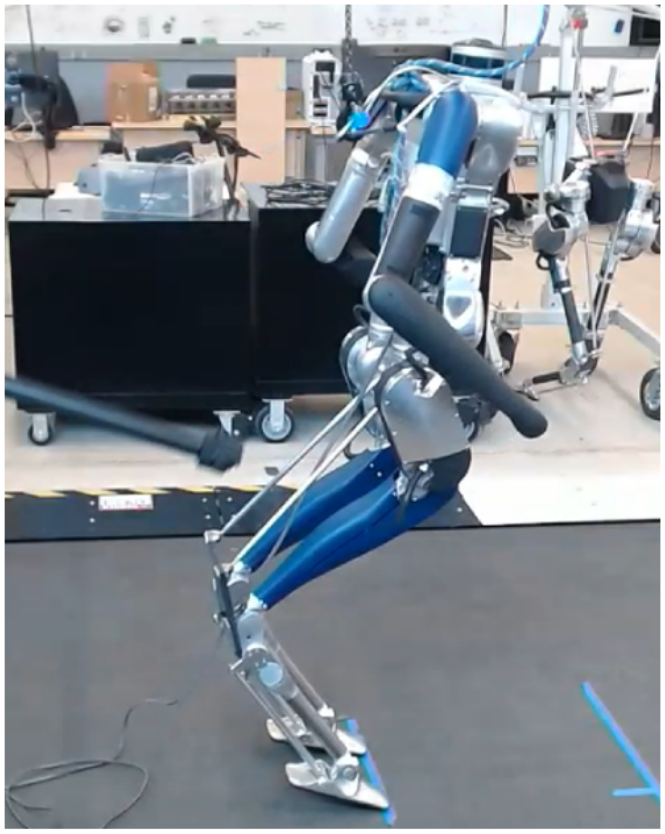
Towards a Fall-Tolerant Framework for Bipedal Robots

by

Margaret Eva Wangari Mungai

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2024



Margaret Eva Wangari Mungai

mungam@umich.edu

ORCID iD: 0000-0001-7011-6912

This dissertation is dedicated to my grandparents and parents, without whom this PhD would not have been possible. I hope that I have made you all proud.

ACKNOWLEDGMENTS

My PhD journey has been lengthy, challenging, yet deeply rewarding, and I couldn't have done it alone. From Nairobi, California, New York, and finally to Michigan, this voyage has been marked by the unwavering support of various individuals. The saying that it takes a village to raise a child could not be truer. Despite any omissions, I'm genuinely thankful to each person who has supported, encouraged, and believed in me along the way. Thengiu muno!

I'd like to start by thanking my parents; their tireless sacrifices, unconditional love, and confidence in my abilities have been the driving force behind my success. Deep gratitude goes to my grandparents, whose encouragement and support were crucial to my pursuit of a Ph.D. degree. To my brothers, Solomon and Ian, thank you for always putting a smile on my face and cheering me on. To the Kenyan community and the rest of my family, words cannot express how truly grateful I am for you all.

To my advisor, Jessy Grizzle, your belief in me, mentorship, and guidance have been pivotal in my academic journey. I am grateful for your patience and the opportunities you provided for personal and professional growth. I'd also like to express my deepest appreciation to my committee for their expertise, advice, and willingness to meet with me individually. Their valuable feedback helped shape this dissertation.

I am immensely fortunate to have been part of a lab with such a collaborative and welcoming environment. I want to thank Bruce, Wami, Grant, Yukai, Omar, Ross, Zhenyu, Ayonga, Maani, Ray, and Lu for their camaraderie and insightful discussions. Especially, thank you to Grant for his constant availability and willingness to help and for the Digit system repository. I'm also grateful to the senior members of the lab, Omar, Ross, Zhenyu, Ayonga, Maani, and Dennis, for their willingness to share their knowledge. To Wami, thank you for always making sure I never miss out on an event. A special thanks is reserved for Gokul for his extraordinary work on the fall prediction project during his Master's degree.

To the Wandercraft team, I want to express my gratitude for an incredible internship and for welcoming me warmly as part of your team. Additionally, I would like to thank Victor, with whom I collaborated on the exoskeleton work, for his time and engaging discussions.

I am grateful to both the Robotics and Mechanical Engineering staff and the Bridge Program team for their willingness to listen and provide the necessary assistance and guidance. I am especially grateful to Samantha Price, Tiffany Porties, Adam Mael, Professor Kevin Pipe, Denise Edmund, Kimberly Mann, Rachel Casanova, Damen Provost, and Dan Newman. I'd also like to thank the Engineering Teaching Consultant Team; it was a pleasure to work beside you all. I'd be remiss if I didn't thank my mentors outside of the University of Michigan, whose support ignited my passion for research and laid the groundwork for my academic pursuits. Especially, thank you to Professor Micheal Amitay, Dr. Haley Dell'Orso, Mrs. Denson-Low, Dr. Michael Conward, Teacher Thomas, Teacher Lucas, Isaack Karanja, and Linda Muiru.

A heartfelt thank you to my friends for always being there, cheering me on, encouraging me, going on adventures, and sharing countless moments of fun and joy. I feel blessed to have you all in my life. To the friends I met at Michigan like Challen, Jon, Yves, Kwesi, Gabrielle, Nosa, Brian, Dan, Prince, Lankeu, Angelar, and Liz, you all made every step of this journey memorable and made Michigan feel like home to me. Meeting you all has truly been an honor for me. To Paul, Wambui, Jonnie, Njeri, Jimmy, Sharon, Maggie, and Sadaf, my childhood friends who have been there since day one, you're not just friends – you're family, and I appreciate every moment of our shared journey. A special shoutout is reserved for Maggie, thank you for always lending an ear. To Rachel, thank you for your check-ins. Last but not least, I would like to express my gratitude towards my partner, Khabane. His unwavering belief in me has propelled me through numerous challenging deadlines. Thank you for being my pillar of strength throughout this journey.

Funding for this work was supported in part by the Rackham Merit Fellowship, the Ford Motor Company, the NSF under Award No. 1808051, and the NSF Graduate Fellowship. However, this dissertation solely reflects the opinions and conclusions of its author and not necessarily of any of the organizations that helped fund it.

TABLE OF CONTENTS

List of Figures

FIGURE

List of Tables

List of Appendices

Abstract

This dissertation focuses on developing a fall-tolerant framework for bipedal robots, aiming to enhance their ability to navigate challenging situations by effectively assessing, adapting, and responding to uncertainties and disturbances. Bipedal robots, with their unique capability to navigate diverse terrains and restore mobility, are ideal for assisting in critical and day-to-day tasks. However, their real-world deployment is limited due to factors like high-dimensional complex dynamics and a smaller support polygon, making it difficult to achieve stable motion, especially in the face of disturbances and uncertainties.

To address these limitations, the dissertation develops robust controllers and reliable fall prediction algorithms. Feedback controllers have been used in the literature to ensure robustness against disturbances and uncertainties. However, the infeasibility of accounting for all disturbances and uncertainties during real-world operations makes falls inevitable. Falls are undesirable as they can prevent a robot from completing its task, result in damage to the surrounding area, or lead to injuries. Therefore, the dissertation emphasizes the importance of implementing robust controllers and employing methods to predict falls.

This research begins by introducing a systematic method to design control objectives for highly constrained systems and concludes by presenting a 1D convolutional neural network fall prediction algorithm capable of not only predicting falls but also estimating the time to react. The effectiveness of the control objectives is demonstrated through robust, comfortable closed-loop sit-to-stand motions for a fully actuated lower-limb exoskeleton, Atalante. The performance of the proposed fall prediction algorithms is evaluated in simulation using a planar-four link robot based on Atalante and in hardware and simulation for the bipedal robot Digit.

Chapter 1

Introduction

1.1 Motivation

The distinct morphology of bipedal robots gives them the unique capability to navigate diverse terrains, from unstructured environments to human-centric spaces, thereby making them ideal candidates for assisting in daily tasks and critical situations. For example, by acting in parallel with the user's limbs and augmenting their joint torques, lower-limb exoskeletons can help users to stand up and ambulate [5]. However, despite their unique capability, the real-world deployment of bipedal robots is limited. Their high-dimensional, hybrid nature, combined with their smaller support polygon–compared to robots with more legs–and occasional stringent constraints, complicates the achievement of stable motion, particularly when confronted with disturbances and uncertainties.

Disturbances and uncertainties can lead to faults, which are defined as unforeseen deviations in one or more operational variables [6]. If left unaddressed, faults can evolve into critical faults, which lead to falls. Falls are undesirable because they can prevent the robot from completing its task, result in irreparable damage to the surrounding environment, and lead to operator and bystander injuries.

A solution is the implementation of a fall-tolerant framework that combines various algorithms to equip the robot with the capabilities to assess, adapt, and respond effectively to uncertainties and disturbances, thereby minimizing the risks and consequences of falling. For instance, a well-designed controller can counteract the effects of disturbances and uncertainties. However, it is infeasible to anticipate every potential disturbance and uncertainty that may be encountered during real-world operations. This unpredictability makes the occurrence of falls inevitable. Therefore, it is imperative to implement fall prediction and recovery algorithms alongside robust controllers. Fall prediction algorithms aim to predict the occurrence of falls with ample lead time, defined as the difference in time between the prediction of a fall and its occurrence. The minimum amount of lead time needed depends on the chosen recovery algorithm, and the robot's dynamics as discussed in [7–10]. The recovery algorithms aim to execute reflexive motions to prevent the fall from happening or make the "landing" less dangerous.

1.2 Objectives and Contributions

This dissertation contributes to a fall-tolerant framework for bipedal robots by developing robust controllers and reliable fall prediction algorithms. Given the dissertation's focus on bipedal robots, the fall prediction algorithms developed and discussed are only for bipedal robots and not for other applications, such as elderly fall prediction. This is due to the disparity in available measurements [11–15].

The effectiveness of the robust controllers is demonstrated through comfortable sit-to-stand motions for the fully-actuated lower limb exoskeleton Atalante designed by Wandercraft [1], which can withstand a class of physically motivated disturbances. Lower-limb exoskeletons provide people who suffer from lower limb impairments with an opportunity to stand up and ambulate. Sit-to-stand is a crucial task for lower-limb exoskeletons as it allows the user to transfer to the exoskeleton from a wheelchair without assistance and can be a precursor to walking. Furthermore, various studies have shown that allowing a user to stand and ambulate has positive psychological and physical benefits [16–19]. Achieving a safe sit-to-stand motion for the exoskeleton

+ user system (exo-system) can be challenging because of the need to balance user comfort while respecting hardware bounds and being robust to changes in the user characteristics and the user's environment. This dissertation successfully achieves safe sit-to-stand motions using constrained optimization to generate two types of dynamic sit-to-stand motions based on hybrid systems. The methods proposed for the sit-to-stand motions in this dissertation can be easily applied to other exoskeletons or bipedal robots. The contributions of these methods in comparison to literature [20–33] are as follows:

- Modeling the exo-system
  - This dissertation: Modeling the sit-to-stand motion using the full 3D exo-system.
  - Literature: Existing literature captures only the sagittal plane portion of the sit-to-stand motion, making it challenging to ensure hardware bounds are met in the frontal and transverse planes.
- Generating the sit-to-stand motions
  - This dissertation: An analysis of two dynamic sit-to-stand motions, each ideal for rejecting specific perturbations.
  - Literature: Generated dynamic motions are for a simplified model of the exo-system and mostly match only one of the two motions developed in this dissertation.
- Executing the motions
  - This dissertation: A novel and systematic way of choosing the control objectives for highly constrained systems such that they are independent of the contact constraints.

  - Literature: No such method exists in the literature to the author's best knowledge.

- Robustness tests

  - This dissertation: Use of physically motivated robustness tests to analyze the two sit-to-stand motions. From these tests, one can assess the range of variations in which the exoskeleton can operate. These ranges can be used to inform new hardware design or to further robustify the controller. Note that the highly non-linear nature of the sit-to-stand problem makes it infeasible to analytically implement stability methods.
  - Literature: Robustness tests do not cover the user or their affordances.

The performance of the fall prediction algorithms introduced in this dissertation are evaluated in simulation using a planar-four link robot based on Atalante, and both in hardware and simulation for the bipedal robot Digit designed by Agility Robotics [2]. As designing a recovery controller is beyond this dissertation's scope, a lead time of 0.2s, which is the lead time required by reflexive algorithms such as [9] and [34], is used as the minimum required lead time. Early fall prediction is a challenging task due to the masking effects of controllers (through their disturbance attenuation actions), the direct relationship between lead time and false positive rates, and the temporal behavior of the faults/underlying factors. As such, this dissertation begins by developing fall prediction algorithms for a planar four-link robot based on Atalante for the task of standing, thereby simplifying the problem while providing a means to scale up to more complex robots and more dynamic motions. The success of these algorithms lead to the development of a fall prediction algorithm capable of detecting critical faults and approximating lead time for the bipedal robot Digit during the task of standing both in simulation and hardware. The proposed algorithm can be easily applied to other full-sized 3D bipedal robots. In comparison to the literature [7–10, 34–50], the contributions of the fall prediction algorithms in this dissertation are as follows:

- Prediction of falls
  - This dissertation:
    - Successful implementation of a fall prediction algorithm capable of predicting falls arising from multiple critical faults, both in simulation and on hardware for a full-sized bipedal robot.
    - Multi-classification-based algorithms capable of detecting multiple critical faults while providing sufficient lead time for corrective motions for the planar four-link robot.
    - Development and comparison of physics-based and data-based algorithms capable of detecting multiple critical faults for the planar four-link robot.
  - Literature: Addresses falls arising from only one type of fault.
- Prediction of lead time
  - This dissertation: Development of a robust method to estimate the lead time.
  - Literature: No such method exists in literature to the author's best knowledge.
- Open-source dataset
  - This dissertation: Open-source dataset of a full-sized 3D bipedal robot comprised of simulation and hardware trajectories with various critical and non-critical faults. The dataset is available here: `https://github.com/UMich-BipedLab/Digit_Fall_Prediction_Dataset`
  - Literature: No such dataset exists to the author's best knowledge.

1.3 Outline

The dissertation is structured as follows: Chapter 2 provides the necessary background information to understand the rest of this document, including a description of the robot platforms used. Chapter 3 covers the work on feedback control design for a lower-limb exoskeleton. Chapter 4 builds on Chapter 3 by implementing closed-loop sit-to-stand motions in two simulators that consider contact dynamics. Chapter 5 discusses the design of data-based and model-based fall prediction algorithms for a planar four-link robot. Chapter 6 introduces a data-based fall prediction algorithm for a full-dimensional bipedal robot that can predict the time left to recover while minimizing false positive rates. Chapter 7 further analyzes the fall prediction algorithm introduced in Chapter 6 by applying it online both in hardware and simulation. Finally, Chapter 8 concludes this dissertation by summarizing the previous chapters and discussing future work.

Chapter 2

Background

This chapter goes over the background knowledge needed to understand the ideas presented in this dissertation.

2.1 Modeling the Robot

Bipedal robots can be viewed as kinematic trees, where each link is connected to another through a joint. Every joint in the kinematic tree connects no more than two links. There are various types of joints, each resulting in different degrees of freedom. For instance, a revolute joint that only allows rotation about the joint axis and hence only has one degree of freedom or a spherical joint that allows rotation about all three spatial axes. The revolute and spherical joints introduce 5 and 3 constraints, respectively, for the motion of a limb in $\mathbb{R}^3$.

The overall degrees of freedom of a robot can be calculated using Grübler's formula:

$$dof \geq m(N-1) - \sum_{i=1}^{J} c_i, \tag{2.1}$$

where $N-1$ is the total number of links minus the ground (the ground is typically counted as a link), J is the number of joints, and c is the number of constraints imposed by a joint. If the joint constraints are independent, Grübler's formula is a strict equality. Otherwise, Grübler's formula only provides a lower bound on the degrees of freedom [51].

2.1.1 Equations of Motion

The dynamic model of a robot can be derived using either the Euler-Lagrange or the Newton-Euler method. Assuming that the only external forces acting on the robot are gravity, motor torques, and contact wrenches, both the Euler-Lagrange and Newton-Euler methods result in the same second-order differential equation

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu, \tag{2.2}$$

where q is the vector of generalized coordinates, u is the torque input vector, and D, C, G, and B are the inertia, Coriolis, gravity, and torque distribution matrices/vectors, respectively. B can be thought of as the Jacobian mapping the control inputs to the generalized coordinates. G appears on the left-hand side even though it's an external force because it's a conserved force. The inertia matrix, D, is symmetric and positive definite. The generalized coordinates are defined as the minimum number of coordinates needed to describe the robot's configuration. Given that the links of the robot are assumed to be rigid, the generalized coordinates are typically taken as the joint angles.

Given the Lagrangian $\mathscr{L}$,

$$\mathscr{L}(q,\dot{q}) = KE(q,\dot{q}) - PE(q), \tag{2.3}$$

where KE is the kinetic energy, and PE is the potential energy, the equation of motion formulated using Euler-Lagrange is expressed as

$$F = \frac{d}{dt}\frac{\partial \mathscr{L}}{\partial \dot{q}} - \frac{\partial \mathscr{L}}{\partial q} \tag{2.4}$$

where $F = \begin{bmatrix} f \\ \tau \end{bmatrix}$ is the generalized wrench composed of the generalized forces (f) and torques (τ). Equation (2.4) can be rewritten in the form (2.2) using the following equations (for simplicity, we have dropped the arguments of each term):

$$\begin{aligned} KE &= \frac{1}{2}\dot{q}^{\intercal} D\dot{q} \\ G &= \frac{\partial PE}{\partial q} \\ C\dot{q} &= (\frac{\partial}{\partial q} D\dot{q})\dot{q} - \frac{1}{2}(\frac{\partial}{\partial q} D\dot{q})^{\intercal}\dot{q}. \end{aligned}$$

For more information on the Lagrangian derivation, please see [51–54].

The Newton-Euler formulation derives the equations of motion based on Newton's second law expressed in terms of linear and angular momentum and builds on the dynamics of a rigid body. More details of the derivation can be found in [51,54]. In comparison to the Newton-Euler method, the Lagrange method is simpler to derive for complex systems, while the Newton-Euler method is more computationally efficient for complex systems. Additionally, the Newton-Euler method is not constrained by the assumption that the configuration of the system can be described by generalized coordinates, whereas the method of Lagrange can only be used after a set of generalized coordinates has been chosen.

The equations of motion can be derived either using a pinned or floating base representation of the robot. A pinned base model assumes that at least one point on the robot is fixed with respect to the world frame, whereas in a floating base model, the robot is not attached to any point in the world frame. The base of the robot can be any of the robot's links and is typically taken as the foot for the pinned base model and the torso for the floating base model. Equation (2.2) displays the equations of motion using a pinned base. When the floating base is chosen, the robot becomes an open chain. As a result, contact forces, such as the ground reaction forces, become external forces, and the generalized coordinates are extended to include the position and orientation coordinates of the robot's base. We rewrite the dynamic equation using the floating base as follows:

$$D_e(q_e)\ddot{q}_e + C(q_e,\dot{q}_e)\dot{q}_e + G(q_e) = B_e u + J^T(q_e)\Gamma \tag{2.5}$$

where q_e is the extended vector of generalized coordinates, D_e, C_e, and G_e are the extended inertia, Coriolis, and gravity matrices/vector, respectively, B_e is the extended torque distribution matrix, J is the Jacobian mapping the contact wrenches to the generalized coordinates, and Γ is the contact wrench.

Unlike (2.2), (2.5) has two unknowns, $\ddot{q}_e$ and Γ. As a result, another equation is needed to solve for both unknowns. This second equation, (2.6), is referred to as the contact acceleration constraint, and it ensures that the contact constraints are met,

$$J(q_e)\ddot{q}_e + \dot{J}(q_e,\dot{q}_e)\dot{q}_e = 0. \tag{2.6}$$

Equations (2.2) and (2.6) together make up the equations of motion for a floating base.

2.1.2 Hybrid System

A robot interacting with the world around it can be described as a hybrid system. Hybrid systems consist of both continuous and discrete states. The discrete states can also be thought of as locations, domains, or phases. Each domain is a set in which the continuous dynamics evolve. Transitions occur either because the dynamics in a domain no longer describe the system due to events such as a loss of contact or because it's simpler to model the system as a collection of simpler systems. For instance, when walking, a model of a 3D bipedal robot consists of three continuous domains: a right stance, a left stance, and a double support domain (which is typically assumed to be instantaneous). The transition from the right stance domain to the left stance domain occurs when the left foot leaves the ground and the right gains contact.

The transition of a walking robot from a left to right stance can be described as a loss of contact with the left foot and a contact gain with the right foot. The states in the transition are discrete states that take values in a countable or finite set. The continuous states take values in $\mathbb{R}^n$.

Hybrid systems can be represented by directed graphs and described using various models, such as the hybrid automaton and the transition system. However, in this overview, we will focus on hybrid automatons. A hybrid automaton H is a tuple $(Q, X, f, Init, Inv, E, G, R)$ where:

- Q: finite set of discrete states, locations, domains, or phase
- $X \subseteq \mathbb{R}^n$: set of continuous states
- $f : Q \times X \to \mathbb{R}^n$: vector field that describes how each of the continuous states changes over time
- $Init \subset Q \times X$: set of initial states
- $Inv : Q \to 2^X$: conditions that have to be met to be in a particular domain
- $E \subseteq Q \times Q$: set of edges that describes the transition relation between continuous domains
- $G : E \to 2^X$: guard condition that determines when to transition between domains
- $R : E \times X \to 2^x$: reset map that maps states accordingly after a transition

Other definitions of hybrid automatons can be found in the literature to account for various modeling conditions [55, 56].

2.2 Generating the Motion: Optimization

There are numerous ways to generate trajectories for the nominal or desired motion, such as basing the motion on previously recorded human movement, using hand-crafted trajectories, or generating trajectories through optimization. This section will focus on obtaining trajectories through parameter optimization.

The optimization problem consists of a cost function and constraints. As seen in Equation (2.7), the cost function is comprised of both a running cost L and terminal cost term F.

$$J := \int_{t_0}^{t_f} L \ \ dt \ \ + F \tag{2.7}$$

The constraints can either be inequality or equality constraints and can be used to ensure the feasibility of the derived motion. For a bipedal robot's derived trajectory to be feasible, it has to satisfy, at minimum, the equations of motions and the hardware bounds, such as joint limits. If it's desired for one or both of the feet to be in contact with the ground, additional constraints such as the friction cone constraints and the zero moment point (ZMP) can be utilized to further ensure dynamic feasibility. For more information on optimization see [57–59]

2.3 Feedback Control

The objective of a feedback or closed-loop controller is to track the desired motion (a.k.a. reference), reject disturbances, ensure robustness against uncertainties, and maintain the stability of the controlled system. This objective is accomplished by constantly monitoring the system's outputs, comparing them to the desired values, and adjusting the control input or signal accordingly.

A feedback control loop for bipedal robots can be represented in block-diagram form, as depicted in Figure 2.1. The loop comprises the robot (plant), a controller, a sensor (state estimator), and a reference input, $h_d(t)$. The reference input specifies the desired control objective or evolution of the robot's states, while u denotes the control input or motor torques. The measured value of the control objective is denoted as $h_0(q^m)$. The tracking error, $y = h_d(t) - h_0(q^m)$, quantifies the discrepancy between the

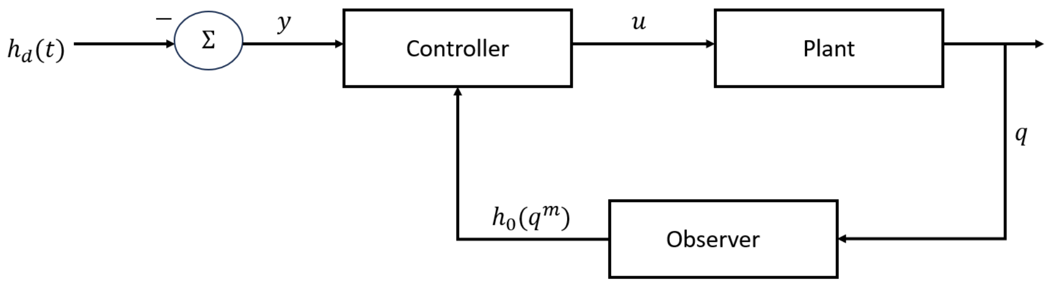


Figure 2.1: Feedback Control Block Diagram

desired and measured control objectives. It is worth noting that if all the generalized states can be measured and there are enough control inputs, then $q^m = q$. [60, 61]

2.3.1 Virtual Constraints

Virtual constraints are kinematic relations among the generalized coordinates of a robot that are enforced by a feedback controller rather than physical constraints. Despite not being physically imposed, they serve as control objectives, defining the desired motion that a legged robot should adhere to. Virtual constraints offer advantages over physical constraints imposed by mechanical connections since they can be dynamically reprogrammed without necessitating modifications to the physical links or the robot's environment [53]. Additionally, their capacity to impose intuitive control objectives such as leg length and torso angle in addition to directly controlling actuated joints reduces the complexity of control and allows for more intuitive and agile motions that can readily adapt to unforeseen disturbances and uncertainties as demonstrated by [62–66].

We assume a virtual constraint of the form

$$y = h_0(q) - h_d(t), \tag{2.8}$$

where $h_0(q)$ is a vector of variables to be "controlled" or "regulated' and $h_d(t)$ is the desired evolution. The control objective is $y(t) = 0$, and thus if $q^*(t)$ is an optimal motion of the robot, we define

$$h_d(t) := h_0(q^*(t)).$$

2.3.2 Computed-torque control

The basic idea of computed-torque control, or input-output linearization, is to design the control input $u = \Gamma(x,t)$ such that

$$\ddot{y} + K_d\dot{y} + K_p y = 0, \tag{2.9}$$

where $K_p > 0$ and $K_d > 0$ are selected so that y converges sufficiently rapidly and "smoothly" to zero [53, 67].

If $h_d(t)$ is at least twice differentiable, computing $\ddot{y}$ is done exactly as if one were imposing a contact constraint,

$$\ddot{y} = J_h(q)\ddot{q} + \dot{J}_h(q,\dot{q})\dot{q} - \ddot{h}_d(t) \tag{2.10}$$

where

$$\begin{aligned} J_h(q) &:= \frac{\partial h_0(q)}{\partial q} \\ \dot{J}_h(q,\dot{q}) &:= \left[\frac{\partial}{\partial q}\left(\frac{\partial h_0(q)}{\partial q}\dot{q}\right)\right]. \end{aligned} \tag{2.11}$$

2.4 Stability Margins

One of the major challenges for bipedal robots is to avoid falls. Therefore, in line with [68, 69], we define a bipedal robot as stable if and only if its current state will not lead to a fall. There are several stability margins that are used to indicate when a bipedal robot will fall. This section will briefly go over some of these stability margins. For a more in-depth look at stability margins and safety regulation for bipedal robots, see [68–81]

2.4.1 Lyapunov Theorem

Lyapunov stability theorem requires one to find a continuously differentiable function V that captures the system's behavior. If V is positive definite and its derivative $\dot{V}$ is negative semi-definite, the system is stable. However, finding a Lyapunov function V for highly non-linear systems is challenging. For more information on Lyapunov functions, see [67].

2.4.2 Control Barrier Funtions

When given an unsafe set $\mathbf{X}_u$, such as all the states that lead a robot to fall, one may seek a barrier function B to prove that the unsafe states are unreachable from any state in the set $\mathbf{X}_0$. $\mathbf{X}_0$ contains all the initial states in which the robot can start. A barrier function is defined as a continuously differentiable function that is positive definite for all states in $\mathbf{X}_u$ and negative semi-definite for all states in $\mathbf{X}_0$. Additionally, it has a negative semi-definite derivative. Similar to the Lyapunov function, the barrier function can be challenging to find for highly non-linear systems. See [82] for more information on control barrier functions.

2.4.3 Poincare Maps

If the system is periodic, the eigenvalues of a Poincare map's Jacobian can be used to determine the local exponential stability of the system. Given a state $x \in \mathbb{R}^n$ and a surface $S \in \mathbb{R}^{n-1}$ that is transverse to the trajectories of the robot (in other words, all trajectories starting on S go through S), the Poincare map, P, is a mapping that maps S to itself $P : S \to S$. If all eigenvalues of the Poincare map's Jacobian are contained within the unit cycle, the periodic system is locally exponentially stable. Note that S is typically taken as the switching surface for walking. The disadvantage of Poincare

Maps is that it assumes periodicity and can thus only be used for periodic motions. For more information on Poincare maps please see [83, 84]

2.4.4 Zero Moment Point, Center of Pressure, and Foot Rotation Index

The center of pressure is defined as the point where the net moment from the contact wrenches is zero about the x and y-axis. If this point is inside the robot's support polygon, then the robot is safe. If the robot is in single support and the foot is stationary, then the zero moment point, center of pressure, and the foot rotation index are the same point. If the foot has rotational acceleration, the zero moment point and the center of pressure are at the edge of the support polygon, while the foot rotation index is outside the support polygon. Note that maintaining the ZMP inside the support polygon is not a necessary or sufficient condition for the stability of the robot. For instance, during non-flat-footed walking, the feet roll [68, 85, 86].

2.5 Fall Prediction

Falls in bipedal robots can often be traced back to faults, which are characterized as unforeseen deviations in one or more operational variables. Based on their temporal behavior, faults can be classified into three categories: abrupt, incipient, and intermittent. Abrupt faults manifest as sudden or rapid changes, incipient faults evolve gradually over time, and intermittent faults appear sporadically [6]. Each of these fault types can arise during real-world operations. For example, abrupt faults might be triggered by unexpected interactions with the environment (e.g., stepping in a hole), incipient faults could stem from discrepancies in the model (e.g., poor trajectory tracking in the operational space of the robot), and intermittent faults might emerge in unpredictable environments laden with obstacles. We term the faults that precipitate a fall as critical faults and the paths leading to a fall as unsafe or faulty trajectories. Figure 2.2 demonstrates the time dependency of the three faults.

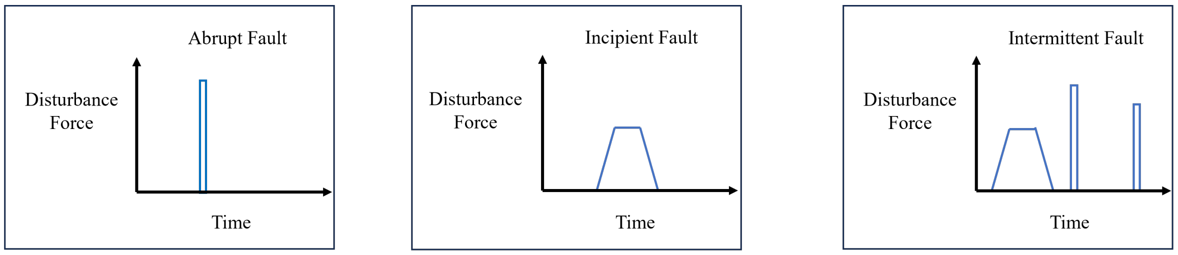


Figure 2.2: The time dependency of faults.

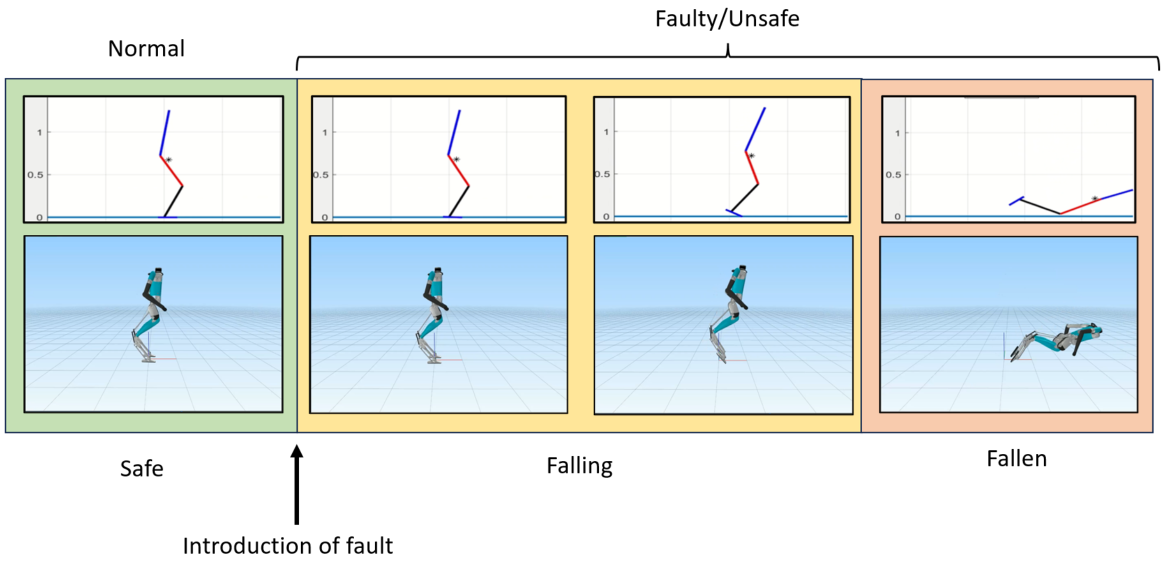


Figure 2.3: The states of a bipedal robot under the influence of faults.

In general, the states of a bipedal robot can be divided into three classes: safe/balanced, falling, and fallen as shown in Figure 2.3. [8] The safe/balanced states are states where it is possible for the robot to avoid falling while under the influence of its nominal feedback controller. These states are, therefore, contained in a subset of the viability kernel [9, 34, 69].

2.5.1 Anomaly Detection

The prediction of falls can be approached through the lens of anomaly detection. Anomaly detection is the problem of identifying data points that significantly deviate from expected patterns. In the context of fall prediction, these anomalous data points are the unsafe or faulty states. It is worth noting that anomalies are referred to by various terms in the literature, such as outliers, and that anomaly detection is closely related to novelty detection, which is the problem of identifying novel patterns. [87–90]

Anomaly detection algorithms can be classified into several categories, such as classification, feature extraction, and nearest-neighbor. Classification models learn a model from labeled training data and classify a data point into a class based on the learned model. These algorithms assume that the feature space learned can preserve the information necessary to distinguish anomalies from normal classes. However, a disadvantage of these algorithms is that they rely on accurately labeled data, which is not always available.

On the other hand, feature extraction algorithms assume that normal data are better represented in a lower dimensional space. The disadvantage of these methods is that the learned feature representations can be suboptimal since the objective of these algorithms is dimension reduction. Moreover, the feature representations can be biased by anomalies in the data.

Nearest-neighbor-based algorithms, especially those based on density, assume that normal data exist in highly dense spaces, whereas the neighborhood of anomalous data is sparse. However, these methods tend to have high false positives if the normal instances do not exist in dense enough neighborhoods [87, 90].

To overcome the disadvantages of individual categories such as those listed in the previous paragraphs, anomaly detection algorithms can be comprised of various methods and thus can belong to multiple categories. Examples of anomaly detection algorithms are [91–94].

2.6 Robot Description

This section contains the descriptions of all the robots used throughout the dissertation.

2.6.1 Atalante

Atalante is a fully actuated hands-free lower-limb exoskeleton developed by Wandercraft for patients with paraplegia [1]. Each leg has six actuated joints:

- Frontal Hip Joint
- Transverse Hip Joint
- Sagittal Hip Joint
- Sagittal Knee Joint
- Sagittal Ankle Joint
- Henke Ankle Joint.

Encoders are located on each actuated joint, and an inertial measurement unit is located at the torso; see Figure 2.4. The adjustable thigh and shank links on Atalante allow it to be worn by users ranging from 1.55 to 1.90 m and 50-90 kg. The Henke axis on Atalante is defined as a 38^o deviation from the horizontal axis of the foot's sagittal plane. There are four force sensors on the corner of each foot that allow for the detection of ground reaction forces (GRF). The exoskeleton is attached to the user via multiple straps on each leg and foot, and a belt/jacket set on the torso [1]. Atalante has been certified for use in the European Union (CE Marking) and is operational in various rehabilitation centers in France.

2.6.2 Four-Link Robot

The four-link robot, Figure 2.5, is a planar robot based on Wandercraft's exoskeleton Atalante [1]. The four links are joined by three actuated revolute joints called the ankle, knee, and hip.

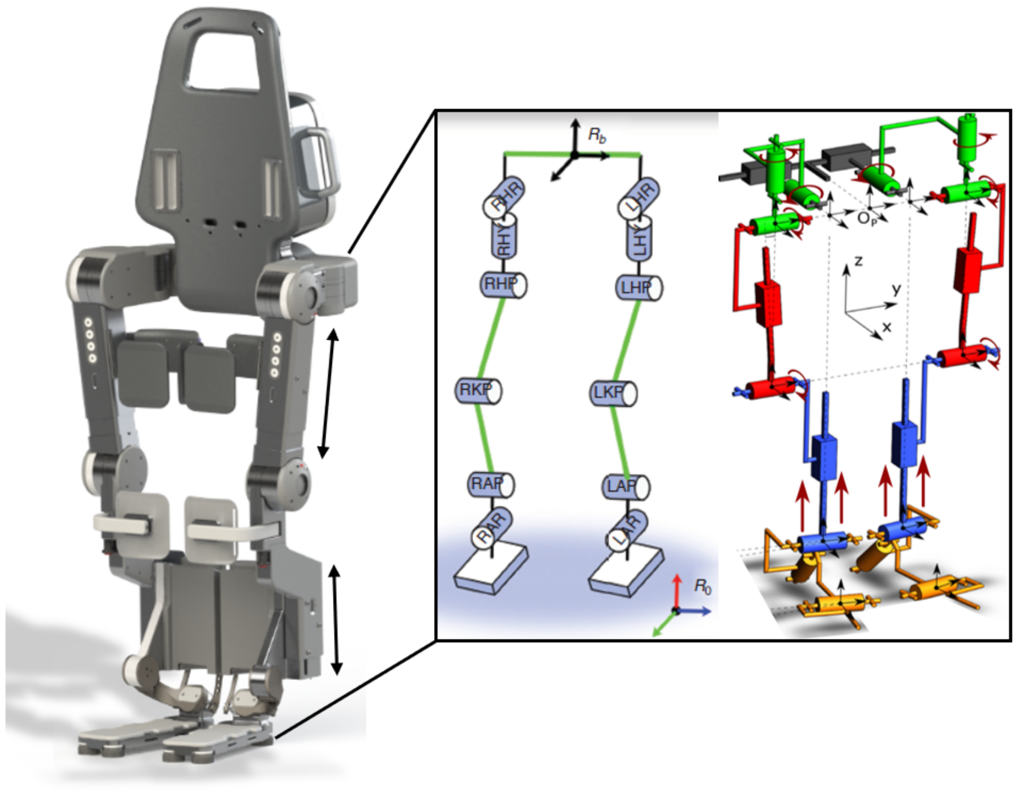


Figure 2.4: Kinematics architecture of Atalante [1]

2.6.3 Digit

Developed by Agility Robotics, Digit is a state-of-the-art bipedal robot [2,3]. While it draws inspiration from Agility Robotic's earlier model, Cassie, Digit distinguishes itself with the addition of a torso and an integrated perception system. Possessing 30 degrees of freedom, Digit has 20 actuated joints. Weighing in at 48kg, its lower limb design is inspired by a Cassowary bird, leading to the unique nomenclature where what would typically be termed "feet" are actually "toes"; we will use the latter terminology. The kinematic architecture of Digit is illustrated in Figure 2.6.

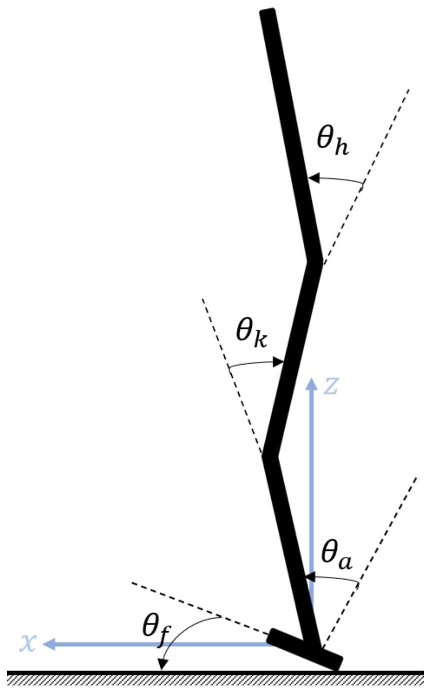


Figure 2.5: Four Link Robot

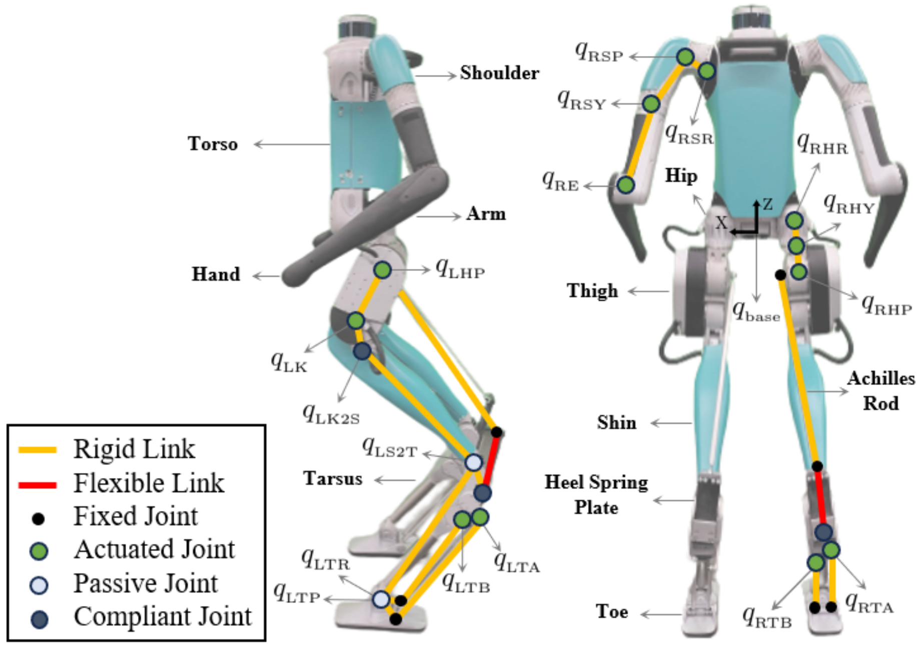


Figure 2.6: Kinematics architecture of the Digit robot by Agility Robotics, [2]. Image Credit: Grant Gibson [3].

Chapter 3

Feedback Control Design for Robust Comfortable Sit-to-Stand Motions of 3D Lower-limb Exoskeletons

3.1 Introduction

This chapter contributes to the implementation of robust controllers by introducing a systematic way of choosing control objectives such that they are not in conflict with contact constraints. This methodology is applied to the task of standing up from a sitting position using Atalante, a fully-actuated lower limb exoskeleton. The work that is presented in this chapter was previously published in [95] with Jessy W. Grizzle as a co-author.

3.1.1 Motivation

Lower-limb exoskeletons are assisting patients with mobility impairments, such as the elderly or people with paraplegia. Mobility restoration is achieved by the exoskeleton acting in parallel with the user's limbs and augmenting their joint torques [5]. This external assistance is allowing patients to carry out day-to-day activities that would be otherwise difficult to achieve in a wheelchair autonomously. Various studies have shown that allowing a user to stand and ambulate has positive psychological and physical benefits [16–19].

Lower-limb exoskeletons can be active or passive, stationary or non-stationary, and crutch-assisted or hands-free (aka, crutchless). Furthermore, the assistance provided by the exoskeleton can be in the form of assist-as-needed or complete assistance. Assist-as-needed exoskeletons require the user to have some mobility in their lower limbs. Most if not all of the lower-limb exoskeletons on the market require the user to have good control over their upper body [96–99].

To enable a user to make the transition from a wheelchair to a lower-limb exoskeleton without any outside assistance, the exoskeleton needs to start from a sitting position. As a result, it's imperative to develop proper trajectories and algorithms that will enable the exoskeleton to stand up in a robust manner. In this study, robustness will be interpreted as (i) insensitivity to variations in user mass and inertia properties, (ii) operability over a range of chair heights, (iii) functions for a range of patient spasticity, and (iv) the ability to handle variations in the torque provided by the powertrain of the exoskeleton.

3.1.2 Literature Review

It is convenient to divide algorithms for sit-to-stand into three main parts: A) modeling the exoskeleton + human system (exo-system), B) generating the motion, and C) executing the motion.

The model of the exo-system can be based either on an approximation of its full dynamics or on a significantly simplified representation. Arguing that the sit-to-stand motion occurs mostly in the sagittal plane, the literature mostly models the exo-system as a planar 3-link inverted pendulum consisting of either the shank, thigh and HAT (head, arm, and trunk) [20–24, 28] or the shank, thigh, HAT, and feet [25–33]. The underlying assumptions for these simplifications are joint symmetry, feet fixed on the ground, and no movement of the neck and head relative to the torso.

While these low degree-of-freedom models simplify the sit-to-stand problem, they typically prevent the assessment of the full capabilities of the exoskeleton. Furthermore, it may not be possible to explicitly apply some design constraints and thus one cannot truly ensure that real hardware requirements are respected. This work will therefore use a high degree-of-freedom 3D model of the exoskeleton and will, for instance, be able to assess the effect of asymmetry in patient spasticity.

When formulating a sit-to-stand controller design, it is very important to determine whether the exoskeleton will be providing complete assistance or only assistance as needed. Trajectory tracking is generally employed to achieve standing motions for complete assistance, whereas, with assistance as needed, it may be required to first estimate the user's intent and then complement the user's effort [25,29,100,101]. This dissertation focuses on complete assistance.

Numerous approaches to generating trajectories for tracking have been followed in the literature, such as basing the motion on previously recorded human movement [102–105], using hand-crafted trajectories [20,27,29,106,107], or trajectories generated through optimization or dynamic movement primitives. Generating trajectories based on human motion is applicable only if the exo-system does not have significantly different kinematic and mass properties in comparison to a human. Hand-crafted trajectories are applicable when only a few joint trajectories are needed, such as with simplified models; of course, they may not take full advantage of the exoskeleton's capabilities. The use of dynamic movement primitives [24] or optimization methods such as the minimum jerk criterion [26,106], constrained optimization [108,109], or genetic algorithms [23,110], have been employed to address the disadvantages of other methods. For instance, a constrained optimization problem can ensure that a designed sit-to-stand motion explicitly accounts for torque bounds. In addition, Zero moment point (ZMP) or Center of Pressure (CoP) bounds can be included to ensure that the generated trajectory is feasible [29,103,107,111] in terms of foot roll.

This dissertation will use constrained trajectory optimization on a high degree-of-freedom model and nonlinear control for implementing the trajectories. The complexity of the resultant trajectory design problem will be addressed through a recent tool, Fast Robotics Optimization and Simulation Toolbox (FROST) [112]. The challenges of the nonlinear control of an over actuated highly constrained motion will be addressed through computed-torque control and a quadratic program (QP) [113–119] for torque distribution while respecting constraints.

Most exoskeletons are not able to support a sit-to-stand motion without outside assistance. In fact, to the authors' best knowledge there are only two hands-free exoskeletons on the market: REX [120], and Atalante [1]. This dissertation focuses on Atalante because a detailed model has been generously shared with the authors. Moreover, because Atalante has been explicitly designed for dynamic walking [63], it is interesting to seek dynamic standing trajectories that can be achieved with minimal user assistance, and no other assistance, or even no user assistance at all. A static sit-to-stand motion requires intermediate poses to be stable throughout the motion, while dynamic motion refers to a continuous trajectory, which, like a dynamic walking gait, does not guarantee stability at intermediate points of time. Even though the "inherent stability" of a static motion appears to be more desirable than a dynamic motion, the severe constraints required by the trajectory are often incompatible with hardware limitations (e.g., joint torque limits). External force from the user, either by pushing downward on the arms of a chair, crutches, or FES [22], have been used to achieve assisted sit-to-stand motions. Allowing for the user to apply an external force can enhance stability of the motion as well as user confidence in the motion.

Counting on an external force, however, comes at the cost of adding complexity to the design of the control system. This dissertation will assess the effects of imperfect application of the user's force and will seek to limit the force demands on the user.

To robustly perform the sit-to-stand motion, it is important that the chosen controller can track a desired trajectory in the presence of disturbances. There are several controllers that have been used in literature such as the computed-torque or input-output feedback linearization controller [20, 23, 32, 100], sliding mode controller [27], fuzzy controller [121], proportional-integral-derivative based controller [29,103,105,122], impedance controller [24], and LQR [20,107].

3.1.3 Contributions

The objective of the present work is to design user-assisted feedback-stabilized dynamic sit-to-stand trajectories for the exoskeleton Atalante, shown in Figure 2.4, using its full dynamic model. This is a challenging design problem due to the complexity of the dynamical system and considerations such as user comfort and safety-critical constraints.

To address this challenge, innovations must be made in the three areas identified in Section 3.1.2, namely modeling the exo-system, generating the motion, and, executing the motion. Our contributions include:

1. Modeling the Exo-System
    - Our Contributions: Modeling the sit-to-stand motion using the full 3D exo-system.
    - Literature: Only the sagittal plane portion of the sit-to-stand motion is captured, which leaves out the torque requirements on actuators in the frontal and transverse planes.
2. Generating the Motion
    - Our Contributions: An analysis of two types of dynamic sit-to-stand motions, chair-to-stand and chair-to-crouch-to-stand, based on hybrid system models and constrained optimization. The two motions are similar when the exo-system is sitting in the chair but differ when the exo-system is off the chair. The chair-to-stand motion simultaneously extends the joints and shifts the CoM forward, while the CoM is first shifted forward and then the joints

extended in chair-to-crouch-to-stand. The chair-to-stand motion is symmetrical and consists of motion mainly in the sagittal plane.

- Literature: The motions are generated for a simplified model of the exo-system. While both quasi-static and dynamic motions are studied, the dynamic motions are most similar to chair-to-stand.

3. Executing the Motion

Control Objectives:

- Our Contributions: We provide a novel and systematic way of choosing the control objectives for highly constrained systems in such a way that the objectives are not in conflict with the contact constraints. Since the resulting closed-loop system is underdetermined (aka, over actuated) and must satisfy real-time constraints on joint limits, torque bounds, and ground reaction forces, we combine quadratic programming with input-output linearization (QP I/O) to (robustly) achieve the sit-to-stand motion and to safely come to a stop.
- Literature: To the best of our knowledge, there currently isn't a way of systematically choosing control objectives for highly constrained systems. When we use the control objectives in [20, 23, 25–27, 29, 32, 104, 123] and a QP-enhanced input-output linearizing controller from [20,29], we find that a 0.02 m increase in chair height results in foot contact violations of over 100 N.

Robustness Tests

- Our Contributions: Physically motivated perturbation tests that help us analyze and compare the two sit-to-stand motions. In our tests, we subject our controller to the following perturbations: different users in the exoskeleton, different chair heights, zero user force, spasticity in the knee joints, and asymmetric motor torque outputs. From these tests we are able to assess ranges of variations in which the exoskeleton can operate. These ranges can be used to inform new hardware design or to further robustify the controller. The main results from the tests are:

  - It is possible to achieve unassisted sit-to-stand motions that meet user comfort constraints. However, we forgo this approach because the inclusion of user force gives the user confidence.
  - The chair-to-stand motion is better at handling changes in the chair height.
  - The chair-to-crouch-to-stand motion is better at rejecting perturbations that result in asymmetry such as spasticity in the knee joints, and asymmetric torque outputs
  - Both motions are equally capable of handling different users in the exoskeleton.

- Literature: The robustness tests do not cover the user or their affordances.

Our method outlined in the contributions above, can be easily applied to other exoskeletons or humanoids. For instance, our novel way of choosing control objectives can be applied in [124] to choose objectives that are not in conflict with contact constraints in the double support phase. Additionally, our formulation for incorporating the user force can be extended to other motions with multiple contact points.

3.1.4 Assumptions used throughout the chapter

For clarity, we list in one place the assumptions used throughout the chapter. The reasons behind individual assumptions are treated as they appear in the chapter. **Assumptions made while generating the exo-system model:**

- rigid links

- rigid drivetrain
- the user does not generate a moment (in the body frame) when applying forces to the arms of the chair
- at least 10 cm of space between the front of the chair and the back of the feet

Assumptions made when generating the optimal motion:

- a friction coefficient of 0.9 between each foot and the ground
- a friction coefficient of 0.5 between the exoskeleton and the chair
- a torsional friction coefficient of 100 between each foot and the ground

Assumptions made in the execution of the motion:

- after using the exoskeleton several times under the supervision of a trained operator, the user will learn to provide the nominal force predicted by optimization
- the controller must be robust to mismatches in the applied user force

3.1.5 Overview: Chapter Organization

We begin by discussing the dynamic model and the hybrid system problem formulation in Section 3.2. We then utilize this information to form a constrained optimization problem in Section 3.3. In Section 3.4, we introduce our method for systematically choosing control objectives. With the control objectives and constrained optimization problem defined, we generate the sit-to-stand motions and present them in Section 3.5. Next, we design our controllers and prove their stability in Section 3.6. Finally, we analyze the robustness of the controllers in Section 3.7, compare our choice of control objectives to the literature in Section 3.8, and discuss our conclusions in Section 3.9.

3.2 Dynamic Model and Problem Formulation

In this section, we summarize our modeling approach. More specifically, we introduce two hybrid systems models along with a floating-base Lagragian model and stability constraints to ensure feasible sit-to-stand motions.

3.2.1 Dynamic Model

The floating base model of the exoskeleton has 18 degrees of freedom: the usual six degrees of freedom for position and rotation in 3D-space, plus the six degrees of freedom for each leg. The links are assumed to be rigid and the drivetrain from each motor to the corresponding joint is assumed to be rigid as well. Motor inertia and gearing are reflected to the associated link using standard methods [125]. In this study, the user is modeled by including additional mass and inertia rigidly attached to the exoskeleton's torso and legs and hence the exo-system has the same number of degrees of freedom as the exoskeleton.

The overall floating-base Lagragian model takes the form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + J^T(q)\Gamma + J_{ext}^{\top}\zeta \tag{3.1}$$

$$J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} = 0, \tag{3.2}$$

where q is the vector of generalized coordinates, u is the torque input vector, D, C, and G are the inertia, Coriolis, and gravity matrices/vector, respectively, B is the torque distribution matrix, J is the Jacobian mapping the contact wrenches to the generalized coordinates, Γ is the contact wrench associated with the exoskeleton's contact with the floor and the seat of the chair, ζ is the force provided by the user, and J_{ext} is the jacobian that maps the provided user force to the generalized coordinates. Equation (3.2) gives the lagrange multipliers (Γ) that are necessary to enforce the contact constraints. Note that (3.1) and (3.2) are similar to the equations presented in Chapter 6 of Murray et al [54].

The generalized coordinates are taken as

$$q = \begin{bmatrix} \text{torso X} \\ \text{torso Y} \\ \text{torso Z} \\ \text{torso yaw} \\ \text{torso pitch} \\ \text{torso roll} \\ \text{left henke ankle} \\ \text{left sagittal ankle} \\ \text{left sagittal knee} \\ \text{left sagittal hip} \\ \text{left transverse hip} \\ \text{left frontal hip} \\ \text{right frontal hip} \\ \text{right transverse hip} \\ \text{right sagittal hip} \\ \text{right sagittal knee} \\ \text{right henke ankle} \end{bmatrix}, \tag{3.3}$$

where the first six elements are the components of the special Euclidean group, $SE(3)$, with respect to a fixed (world) frame, and the next 12 are body coordinates representing the relative angles of the joints comprising the two legs; each of these joints is actuated.

The assistive force from the user is defined to act at the top of the torso in the torso's body coordinate frame with components in the x, y, and z axes. It is assumed that there is no net moment generated by the user.

3.2.2 Hybrid models

We first discuss the chair-to-stand motion, which is based on a hybrid system model with two domains, the *sitting domain* and the *standing domain*. A specific reference point on the exoskeleton, called the *sitting-point*, is used to establish the point of application of the contact wrench from the seat. The contact with the seat is modeled as a point contact and thus the contact wrench from the seat of the chair consists only of the contact force. The sitting domain has three contact points (chair, right foot, and left foot) while the standing domain has two (right foot and left foot). The standing domain is entered when the exoskeleton is no longer in contact with the seat.

Equations (3.1) and (3.2) define the dynamics for both domains with

$$J = J_{stand} := \begin{bmatrix} J_{foot}^{right} \\ J_{foot}^{left} \end{bmatrix} \tag{3.4}$$

in the standing domain, and

$$J = J_{sit} := \begin{bmatrix} J_{stand} \\ J_{chair} \end{bmatrix} \tag{3.5}$$

in the sitting domain, where J_{foot}^{right}, J_{foot}^{left}, and J_{chair} are the geometric contact constraint jacobians for the right foot, left foot, and chair respectively. The guard function that triggers the transition between the two domains is defined as follows

$$F_{chair}^{z} = 0 \tag{3.6}$$

where F_{chair}^{z} is the vertical component of the force from the seat. Since there is no impact during the transition from the sitting domain to the standing domain, the reset map is an identity matrix. The hybrid model for chair-to-stand is depicted in

Figure 3.1.

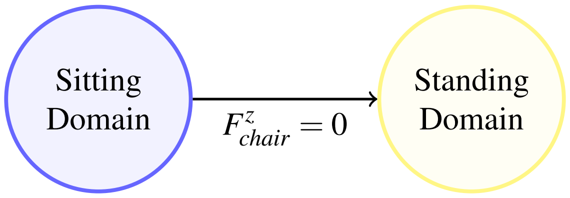


Figure 3.1: The directed acyclic graph for the **chair-to-stand** hybrid system model

We next discuss a dynamic chair-to-stand motion similar to the quasi-static motion used in the literature, where the CoM is first shifted over the feet and then the joints are extended [20, 23]. In this case, the standing domain is divided into two separate domains governed by the same equations of motions, *standing shift* and *standing extend*. Note that the transition between the sitting domain and the stand shift domain is equivalent to the transition between the sitting and standing domain in the chair-to-stand hybrid model. The transition from standing shift to standing extend occurs after the ZMP is in the feet polygon; see Figure 3.3b. The reset map during this transition is the identity. We will refer to this hybrid model as chair-to-crouch-to-stand. The hybrid model for chair-to-crouch-to-stand is given in Figure 3.2. The motions derived from the chair-to-stand and chair-to-crouch-to-stand hybrid system models are later compared for user comfort.

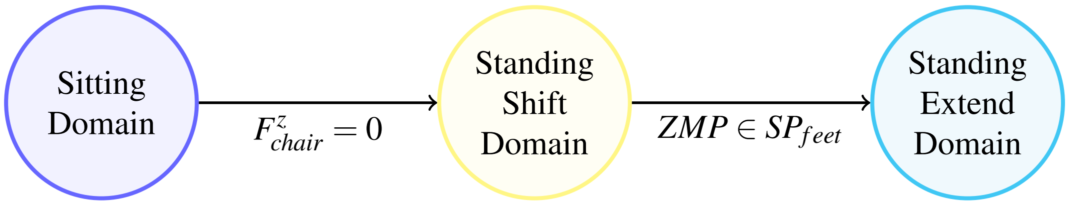


Figure 3.2: The directed acyclic graph for the **chair-to-crouch-to-stand** hybrid system model

3.2.3 Contact Forces and Moments

In all domains, constraints are imposed on contact forces and moments. For example, the vertical components of all contact forces must be non-negative, and in the sitting domain, there must be a minimum amount of weight supported by the feet so that the legs can contribute to a lifting motion, viz

$$F^z_{chair} \geq 0 \tag{3.7}$$

$$F^z_{feet} \geq 0.3 m_{\text{total}} g, \tag{3.8}$$

where F^z_{feet} is the total vertical component of the ground reaction force.

To avoid sliding and yawing, a linear friction cone

$$|F^x_\bullet| \leq \mu_\bullet \frac{F^z_\bullet}{\sqrt{2}} \tag{3.9}$$

$$|F^y_\bullet| \leq \mu_\bullet \frac{F^z_\bullet}{\sqrt{2}} \tag{3.10}$$

and torsional friction constraints are used in all domains

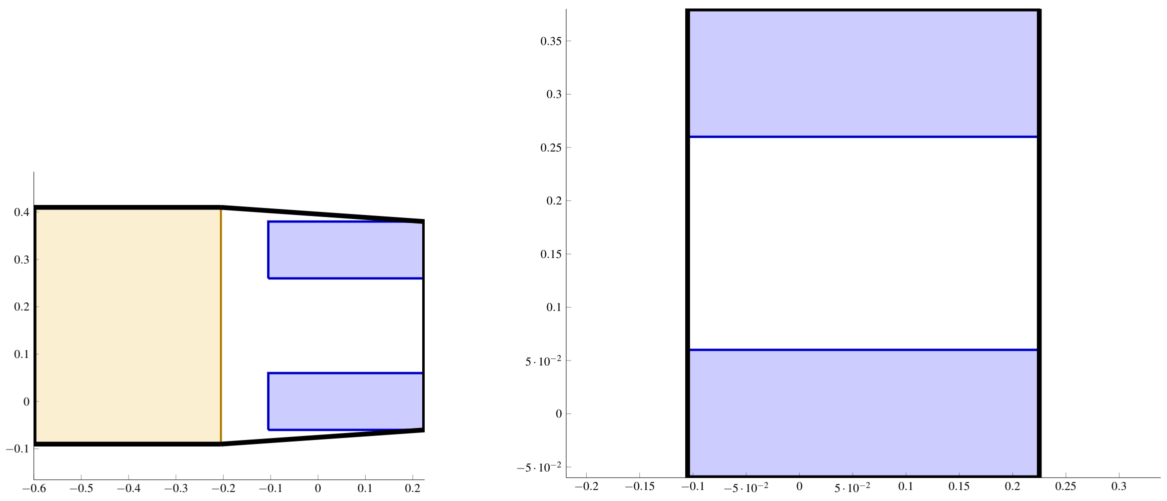
$$|M^z_{feet}| \leq \gamma_{feet} F^z_{feet} \tag{3.11}$$

where $F^x_\bullet$, $F^y_\bullet$ and $F^z_\bullet$ denote the components of the contact forces associated with the chair or the feet, $\mu_\bullet$ is an assumed friction coefficient for the chair or feet, γ_{feet} is a torsional friction coefficient for the feet, and M^z_{feet} and F^z_{feet} are the moments about and forces along the z-axis for the feet.

To prevent the feet from rolling or pitching, the ZMP[1] [85, 86] must lie strictly within the appropriate support polygon (*SP*). When a user force exists or there is contact with the chair, the support polygon is given by the convex hull of the feet of the exoskeleton and those of the chair. Here, there is assumed to be at least 10 cm of open space between the front of the chair and the back of the feet. When there is neither an applied user force nor contact with the chair, the support polygon

[1]The ZMP definition that we use here actually corresponds to the CoP; it is known that the ZMP and CoP are coincident if the contact forces are applied on horizontal surfaces. And even if not, one can define a virtual surface and obtain a pseudo ZMP-CoP definition [86]

is given by the convex hull of the feet. We will refer to the support polygon of the feet and the chair as SP_{both}. The support polygon of the feet will be called SP_{feet}. An example of the two support polygons that will used in this work are depicted in Figure 3.3.



(a) Sitting & Standing Domain (b) Standing Domain: no contact with the chair

Figure 3.3: An example of the support polygons that will be used. The brown rectangle is the chair while the blue rectangles are the feet. Even though the chair is depicted as being wider than the feet in (a), the width of the feet is not predetermined; it will be solved for via optimization. The bold black line in (a) and (b) encompasses the support polygon of the chair and the feet, and just the feet respectively. The chair is set to be 10 cm behind the feet.

To calculate the ZMP, the external wrenches (ground reaction, applied user force, and chair) are converted to the spatial frame using their respective adjoint matrices [51,54]. Once everything is expressed in the spatial frame, the ZMP, denoted P_*, can then be calculated as the point on the ground about which the x and y total moment equals zero [85,86]. The total moment $\tau = [\tau^x, \tau^y, \tau^z]^\top$ is given by

$$\tau := \sum_{i=1}^{N} (P_i - P_*) \times F_i^z + M_{total}, \tag{3.12}$$

where

- N is the number of active contact points

- $M_{total} = [M^x_{total},\ M^y_{total},\ M^z_{total}]^\top$ is the sum of all external moments
- $P_i = [P^x_i,\ P^y_i,\ P^z_i]^\top$, is the point of application of the i-th contact force F_i
- $P_* = [P^x_*,\ P^y_*,\ P^z_*]^\top$ is the unique point on the ground resulting in $\tau = [0,0,\tau^z]^\top$. That is, at P_* the moment is acting about an axis normal to the ground plane.

We note that (3.12) simplifies to (3.13) and (3.14) if $\forall\ i$, P_i is located at the origin (which is the case for us), yielding

$$P^x_* = \frac{-M^y_{total}}{F^z_{total}} \tag{3.13}$$

$$P^y_* = \frac{M^x_{total}}{F^z_{total}}, \tag{3.14}$$

where F^z_{total} is the sum of all the vertical components of the contact forces.

3.3 Motion Generation through Constrained Optimization

Motion generation is posed as a constrained optimization problem for the full-dimensional floating-base hybrid models described in Section 3.2.2. State trajectories, motor torques, and external wrenches are computed with the open-source package FROST [112]. FROST also provides for computing the terms in the Lagrangian model given in (3.1) and (3.2), from the universal robot description file (URDF) of the exoskeleton and user.

Once the cost function, unilateral (i.e., inequality) constraints, and holonomic (i.e, equality) constraints are posed, FROST transcribes this data and the dynamic model into a direct collocation problem with analytic derivatives and solves it with IPOPT. Computation times for the native MATLAB implementation of FROST are discussed in [112]. There is a C++ companion, called C-FROST, that provides for parallel executions and more [126].

3.3.1 Cost Function

The cost function, $J(x,u)$, consists of a running cost, $L(x,u)$, and a final cost that can depend on the domain, terminal cost $F_i(x)$, and the duration of the domain, t_i,

$$J(x,u) := \int_{t_0}^{t_f} L(x(t),u(t),\zeta(t))dt \quad + \sum_{\text{domains}} F_i(x(t_i)).$$

While FROST allows a different running cost to be specified for each hybrid domain, this flexibility was not used. For both the chair-to-crouch-to-stand and chair-to-stand motions, the running cost is

$$L_{sit}(x,u) := k_1||u||_2^2 + k_2||\zeta||_2^2 + k_3(\dot{\theta})^2 + k_4(\ddot{\theta})^2, \tag{3.15}$$

where θ denotes the torso pitch angle and k is a weighting vector with components k_i. The squared Euclidean norms of u and ζ are present to reduce motor effort and user effort, respectively. The inclusion of torso angular velocity and acceleration is for tuning user comfort.

The terminal costs are all zero with one exception: in the chair-to-crouch-to-stand motion, the terminal cost of the standing shift domain (the second domain in chair-to-crouch-to-stand) is the difference between the final and initial knee angles,

$$F_2(x(t_2)) = k_5||q_{\text{Knee}}(t_0) - q_{\text{Knee}}(t_f)||_2^2, \tag{3.16}$$

where q_{Knee} is the vector of the knee angles and k_5 is a weight. The final cost encourages a solution that “rolls” the exo-system from the chair to a crouching position, without extending the legs.

3.3.2 Summary of Physical Constraints for Dynamic Feasibility and User Comfort

In addition to implementing ZMP and friction constraints for dynamic feasibility as discussed in Section 3.2.3, additional constraints are implemented to ensure that the desired optimal motion can be realized. These additional constraints ensure feasibility of the motion in terms of user comfort and hardware limitations, while meeting design specifications. A summary of the constraints is provided here; a more detailed description, including tables with the numerical values of the constraints, can be found in Appendix A.

Anticipating that the feedback controller will need robustness margins, for both motions, the ZMP is constrained to be within a strict subset of SP_{both} and SP_{feet} when the exo-system is in contact with the chair, and no longer in contact with the chair, respectively. The state bounds for the actuated joints are set to be stricter than the hardware bounds to ensure user safety and comfort. Optimization is allowed to use the maximum torque value a motor can output for each actuated joint except the ankle, which is set to a smaller nominal value. These design choices are possible because the real-time quadratic program used in the control implementation is effective at managing torque limits; see Section 3.6.

Both motions are designed to start and end in statically stable positions with zero user force. Therefore, to guarantee a feasible final pose for both motions, the ZMP needs to be within SP_{feet} at the end of the motion. Since the ZMP and CoM are coincident at the end of the motion when the exo-system is static, it is sufficient to constrain the final CoM to be within SP_{feet}. The chair-to-stand motion is constrained to be symmetric while the chair-to-crouch-to-stand motion is not. Therefore, to encourage the optimizer to find a chair-to-crouch-to-stand motion with the CoM near the middle of the feet at the end of the motion, the CoM is constrained to be in a smaller polygon between the feet.

3.4 Designing Control Objectives for a Highly Constrained System

The primary objective here is to introduce a new method for designing control objectives for systems that are highly constrained, such as a full assist exoskeleton in a chair-to-stand or chair-to-crouch-to-stand motion. For a computed-torque controller, we show how to select control objectives $h_0(q)$ that are "orthogonal" to the system's contact constraints and "aligned" with the torque distribution matrix, B. The meaning of the terms "orthogonal" and "aligned" will be clarified in the text.

3.4.1 Virtual Constraints and computed-torque Control

We pose our control objectives in the form of *virtual constraints*, which are relations on the state variables of the robot's model that are achieved through the action of actuators and feedback control instead of physical contact forces and moments. They are called *virtual* because they can be re-programmed on the fly without modifying any physical connections among the links of the robot or its environment. If it is known in advance that virtual constraints will be used, FROST allows them to be designed in parallel with the optimal motion for the mechanism.

We assume a virtual constraint of the form

$$y = h_0(q) - h_d(t), \tag{3.17}$$

where $h_0(q)$ is a vector of variables to be "controlled" or "regulated' and $h_d(t)$ is the desired evolution. The control objective is $y(t) = 0$, and thus if $q^*(t)$ is an optimal motion of the robot, we define

$$h_d(t) := h_0(q^*(t)).$$

The basic idea of computed-torque control, or input-output linearization, is to design the control input $u = \Gamma(x,t)$ such that

$$\ddot{y} + K_d \dot{y} + K_p y = 0, \tag{3.18}$$

where $K_p > 0$ and $K_d > 0$ are selected so that y converges sufficiently rapidly and "smoothly" to zero.

If $h_d(t)$ is at least twice differentiable, computing $\ddot{y}$ is done exactly as if one were imposing a contact constraint,

$$\ddot{y} = J_h(q)\ddot{q} + \dot{J}_h(q,\dot{q})\dot{q} - \ddot{h}_d(t) \tag{3.19}$$

where

$$\begin{aligned} J_h(q) &:= \frac{\partial h_0(q)}{\partial q} \\ \dot{J}_h(q,\dot{q}) &:= \left[\frac{\partial}{\partial q}\left(\frac{\partial h_0(q)}{\partial q}\dot{q}\right)\right]. \end{aligned} \tag{3.20}$$

In the case of contact constraints, the term $J(q)D^{-1}(q)J^{\top}(q)$ is square and invertible if, and only if, $J(q)$ has full row rank. For virtual constraints, the analogous term, $J_h(q)D^{-1}(q)B$, may not be square, and even if $J_h(q)$ is full row rank, $J_h(q)D^{-1}(q)B$ may be rank deficient. Hence, one must choose carefully the controlled variables, $h_0(q)$, so that there indeed exists u satisfying (3.18). This is addressed next for highly constrained motions such as chair-to-stand and chair-to-crouch-to-stand.

3.4.2 Contact Interaction Matrix

Placing the contact and virtual constraints together, and dropping the arguments for compactness of notation, yields

$$\begin{bmatrix} \ddot{c} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} JD^{-1}(C\dot{q} + G - J_{ext}^{\top}\zeta) \\ \dot{J}_h\dot{q} - J_h D^{-1}(C\dot{q} + G - J_{ext}^{\top}\zeta) \end{bmatrix} + \mathscr{I} \begin{bmatrix} \Gamma \\ u \end{bmatrix}, \tag{3.21}$$

where the matrix

$$\mathscr{I}(q) := \begin{bmatrix} J(q)D^{-1}(q)J^{\top}(q) & J(q)D^{-1}(q)B \\ \\ J_h(q)D^{-1}(q)J^{\top}(q) & J_h(q)D^{-1}(q)B \end{bmatrix} \tag{3.22}$$

captures the coupling or interaction of the contact wrenches and the motor torques in achieving the contact and virtual constraints. Hence, we call it the *constraint interaction matrix*.

Setting $\ddot{c} = 0$ to impose the contact constraint and $\ddot{y} = \ddot{y}_d - K_d\dot{y} - K_p y$ to impose the virtual constraints, we have

$$\mathscr{I} \begin{bmatrix} \Gamma \\ u \end{bmatrix} = \begin{bmatrix} JD^{-1}(C\dot{q} + G - J_{ext}^{\top}\zeta) \\ J_h D^{-1}(C\dot{q} + G - J_{ext}^{\top}\zeta) - \dot{J}_h\dot{q} + \ddot{y}_d - K_d\dot{y} - K_p y \end{bmatrix}. \tag{3.23}$$

Equation (3.23) allows us to understand how the motor torques and contact wrenches interact along trajectories of the exo-system. An interesting question is how to select the virtual constraints so that they “do not fight” the contact constraints.

3.4.3 Design Philosophy for the Virtual Constraints

The blocks of $\mathscr{I}$ are generalized inner products with positive-definite weight matrix $D^{-1}(q)$. Fix $q = q_0$, a point along a designed motion, and write

$$\bar{J} := J(q_0)\left(D(q_0)\right)^{-1/2} \qquad n_h \times m \tag{3.24}$$

$$\bar{B}^{\top} := B^{\top}\left(D(q_0)\right)^{-1/2} \qquad n_u \times m \tag{3.25}$$

$$\bar{J}_h := J_h(q_0)\left(D(q_0)\right)^{-1/2} \qquad n_v \times m \tag{3.26}$$

where n_h, n_u, n_v, and m are the number of contact constraints, actuators, virtual constraints, and generalized coordinates respectively; note that $\bar{B} = \left(D(q_0)\right)^{-1/2}B$. With these definitions, the constraint interaction matrix becomes

$$\mathscr{I}(q_0) = \begin{bmatrix} \bar{J}\ \bar{J}^\top & \bar{J}\ \bar{B} \\ \bar{J}_h\ \bar{J}^\top & \bar{J}_h\ \bar{B} \end{bmatrix}. \tag{3.27}$$

The elements forming the top row of A are fixed by the dynamic model and the contact constraints. We propose to design the Jacobian matrix $\bar{J}_h$ arising from the virtual constraints so that $\bar{J}_h$ is orthogonal to $\bar{J}$ and the rows of $\bar{J}_h\ \bar{B}$ are linearly independent. The reasoning is that (a) if the virtual constraints are orthogonal to the contact constraints, then the controller is not acting in the directions of the contact conditions, and (b) if the rows of $\bar{J}_h\ \bar{B}$ are linearly independent, then there exists u such that (3.18) is satisfied. On the other hand, if the columns of $\bar{J}_h\ \bar{B}$ are linearly dependent, then u is not unique and a means of choosing u needs to be provided. The linear dependence of the rows of $\bar{J}_h\ \bar{B}$ arises from the highly constrained nature of a standing motion. In Section 3.6, this aspect of imposing the virtual constraints will be addressed.

In some situations, it would also be beneficial to select virtual constraints that actively fight against undesired contact constraints while respecting desired contact constraints. For instance, in the sitting domain the exo-system is actively trying to break the contact with the chair without slipping in the chair. Therefore, the desired contact constraints in the sitting domain are the feet on the ground and the contact with the chair in the transverse plane. The undesired contact constraint is the chair contact constraint in the z-axis. The question then becomes how to design virtual constraints that respect wanted contact constraints while fighting unwanted contact constraints.

3.4.4 Designing the Virtual Constraints using QR Factorization

In this section, we provide guidelines on how to obtain virtual constraints using QR factorization based on the philosophy presented in Section 3.4.3 . Note that our analysis in this section is dependent on a fixed point q.

3.4.4.1 Non-sitting Domains

Here, J is given by (3.4). Performing a QR-factorization of $[\bar{J}^\top \quad \bar{B}]$ yields

$$\begin{aligned} \begin{bmatrix} \bar{J}^\top & \bar{B} \end{bmatrix} &= QR \\ &= \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}, \end{aligned} \tag{3.28}$$

where the orthonormal matrix Q and the upper triangular matrix R have been partitioned conformally with the size of $\bar{J}^\top$. Then, because R is upper triangular, $R_{21} = 0$ and thus we have

$$\bar{J}^\top = Q_1 R_{11} \tag{3.29}$$

$$\bar{B} = Q_1 R_{12} + Q_2 R_{22} \tag{3.30}$$

Hence, a choice of $\bar{J}_h$ that is orthogonal to $\bar{J}$ and full row rank is

$$\bar{J}_h := Q_2^\top.$$

Indeed, with this choice,

$$A(q_0) = \begin{bmatrix} \bar{J}\ \bar{J}^\top & \bar{J}\ \bar{B} \\ 0 & R_{22} \end{bmatrix} = \begin{bmatrix} R_{11}^\top R_{11} & R_{11}^\top R_{12} \\ 0 & R_{22} \end{bmatrix}. \tag{3.31}$$

The (row) rank of R_{22} gives the number of virtual constraints that can be used in the controller design.

3.4.4.2 Sitting Domain: Actively *Fighting* Against Constraints

In the sitting domain, the exo-system's motion is actively seeking to break contact with the chair by driving the vertical force from the chair to zero. It is desirable that control actions do not promote sliding in the chair, while aiding in lifting from the chair. The virtual constraints should therefore be designed as orthogonal to the x and y components of J_{chair}, the seat Jacobian, but not to its vertical component. We start our analysis by first noting the similarities between the contact and virtual constraint definitions as described in (3.32)

$$\begin{array}{ll} \text{Contact Constraint} & \text{Virtual Constraint} \\ c = c_0 - c_d & y = h_0 - h_d \\ \dot{c} = J\dot{q} & \dot{y} = J_h\dot{q} - \dot{h}_d \\ \ddot{c} = J\ddot{q} + \dot{J}\dot{q} & \ddot{y} = J_h\ddot{q} + \dot{J}_h\dot{q} - \ddot{h}_d \end{array} \tag{3.32}$$

where c_0 and c_d are the current and desired positions of the contact respectively, and $J := \frac{\partial c_0}{\partial q}$. We assume c_d is a constant, and hence $\ddot{c}_d = 0$. As previously demonstrated in Section 3.2.1 and 3.4.1, the contact and virtual constraints, are both implemented at the acceleration level and are achieved when $y \equiv 0$ and $c \equiv 0$. In particular, a contact constraint is not identically satisfied when either $c \neq 0$, $\dot{c} \neq 0$, or $\ddot{c} \neq 0$. Similarly, a virtual constraint is not identically satisfied when either $y \neq 0$, $\dot{y} \neq 0$, or $\ddot{y} \neq 0$. When the virtual constraints are not satisfied, we use control action to mitigate the error. In a similar manner, we can use control action to induce $\ddot{c} \neq 0$, thereby *fighting* the contact constraint. In our case, we want to *fight* against the vertical component of the chair contact constraint to achieve $c_0 > c_d$, which gives $\ddot{c} > 0$.

Therefore, to achieve virtual constraints that are orthogonal to all the contact constraints except for the vertical component of the chair constraint, we modify (3.28) to

$$\begin{aligned}\begin{bmatrix}\bar{J}_r^\top & \bar{J}_v^\top & \bar{B}\end{bmatrix} &= QR \\ &= \begin{bmatrix}Q_1 & Q_2 & Q_3\end{bmatrix}\begin{bmatrix}R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33}\end{bmatrix},\end{aligned}$$

where $\bar{J}_r^\top$ and $\bar{J}_v^\top$ are the Jacobian matrices of the contact constraints we want to be orthogonal to and non-orthogonal to respectively. Similar to before, $R_{21} = 0$, $R_{31} = 0$ and $R_{32} = 0$. Note that Q_2 is in the span of $\bar{J}_v^\top$. Since at the fixed point q, Q_2 is a constant matrix, we can now define

$$\dot{c} = Q_2^\top \dot{q} \tag{3.33}$$

$$\ddot{c} = Q_2^\top \ddot{q} \tag{3.34}$$

The first step for designing control actions that are not only non-orthogonal to the vertical component of the chair constraint but also actively *fight* against it, is to design J_h such that it's orthogonal to $\bar{J}_r^\top$ and aligned with $\bar{B}$. Note that choosing $J_h = Q_3^T$ as was done previously does not work here because n_v will be less than the difference between the DoF of the unconstrained system and the DoF of the constrained system. To properly design J_h, an optimization problem is necessary. The formulation of this optimization problem can be found in Section 3.4.5. Once J_h is found, all that is left to do is to get rid of the undesired contact constraint from the controller and introduce $Q_2^\top \ddot{q} > 0$ as a constraint in the QP when one decides to start "fighting" the contact constraint. We note that since our controller is tracking an optimal trajectory that achieves the sit-to-stand motion, it is sufficient to have control actions that are non-orthogonal to the vertical component of the chair constraint.

3.4.5 From Jacobians to Functions

In this section, we show how to design functions $h_0(q)$ whose Jacobians approximately satisfy

$$\begin{aligned} \frac{\partial h_0(q)}{\partial q} D^{-1}(q) J^\top(q) &= 0 \\ \frac{\partial h_0(q)}{\partial q} D^{-1}(q) B &=: M(q) \quad \text{(well conditioned).} \end{aligned} \tag{3.35}$$

To do this, we propose to include linear and quadratic terms in $h_0(q)$, namely,

$$h_0(q) := H_0 q + \sum_{i=1}^{n} \sum_{j=1}^{i} H_{ij} \quad q_i q_j, \tag{3.36}$$

and to select the coefficients to minimize

$$H^* = \quad \underset{H}{\arg\min} \quad \frac{1}{N} \sum_{i=1}^{N} \left|\left| \, J_h(q) D^{-1}(q) J^\top(q) \Big|_{q=q(t_i)} \right|\right|_2^2 \tag{3.37}$$

subject to

$$\lambda_{min} \left[M(q) M(q)^\top \right]_{q=q(t_i)} \quad \geq 1 \tag{3.38}$$

$$\lambda_{max} \left[M(q) M(q)^\top \right]_{q=q(t_i)} \quad \leq \kappa^2. \tag{3.39}$$

Here, H is the collection of coefficients in (3.36) and $q(t_i)$ are points along a designed motion of the exo-system. The constraint (3.38) is necessary to avoid $H^* = 0$ as a solution to the objective function (3.37), while the tuning parameter $\kappa > 1$ bounds the condition number of $M(q)$; see (3.22).

Remarks:

1. For non-sitting domains, the contact Jacobian, J, in (3.37) is given by J_{stand} in (3.4).

2. For the sitting domain, the contact Jacobian in (3.37) is taken as $J^\top = \left[(J_{\text{stand}})^\top, (J^x_{\text{chair}})^\top, (J^y_{\text{chair}})^\top\right]$, while the z-component is added to $M(q)$ in (3.35), (3.38), and (3.39) per

$$M(q) := \begin{bmatrix} J^z_{\text{chair}}(q) D^{-1}(q) B \\ \\ \frac{\partial h_0(q)}{\partial q} D^{-1}(q) B \end{bmatrix}.$$

   If feasible, the constrained optimization problem posed in this manner will return three virtual constraints that are (approximately) orthogonal to the foot constraints and the horizontal components of the chair constraint, but not to the vertical component of the chair constraint. Hence, the controls may assist in driving the chair's vertical force to zero for transitioning to the next domain. Moreover, through the choice of M, the three virtual constraints will be independent of all 15 contact constraints.

3. The function $h_0(q)$ in (3.36) can be multiplied on the left by an orthonormal matrix without changing[2] the values in (3.37), (3.38), and (3.39).

3.5 Optimization for Nominal Sit-to-Stand and Sit-to-Crouch Behaviors

This section presents the results of performing optimization to design the chair-to-stand and chair-to-crouch-to-stand behaviors, using the cost functions and constraints described in Section 3.3 and the control objectives presented in Section 3.4. The optimal motions are derived using:

- A user who is 1.73 m tall and weighs 73 kg, for a total weight of the exo-system of 147.4 kg.
- A chair height of 0.6 m.
- A friction coefficient of the chair of 0.5. This value was chosen by using the

[2]The cost is quadratic and the eigenvalues of a matrix are invariant under similarity transformations.

friction coefficient of leather and oak for guidance [127]. Even though a friction coefficient of 0.5 is probably smaller than the friction coefficient of an average chair, by using a smaller friction coefficient, we encode robustness to slipping in the optimal trajectory.

- A friction coefficient of the feet of 0.9.

The virtual constraints imposed during the optimization are derived from the optimization problem posed in (3.37) - (3.39). However, because there is relatively little joint displacement in the sitting domain of the chair-to-stand motion, a linear virtual constraint is used. To find the corresponding (constant) matrix, we

1. solve (3.37) - (3.39) for H^*,
2. replace H with H^* in (3.36) and solve for $h_0(q)$ at all the time steps in the trajectory (note that $H_0(q)$ is the first m columns of H^*, and that H_{ij} is evaluated from the remaining columns),
3. iteratively evaluate the condition number of (3.22) with $J_h = h_0(q)$ throughout the entire motion, and
4. choose the h that induces the minimum condition number

It should be noted that for both optimal motions, there is an interplay between the torso pitch acceleration, the user force, and the motor torques. This interplay comes about from (a) the inverse relationship of motor torque and user force: the more the user contributes by pushing with their arms, the less the motors need to contribute; and (b) taking advantage of the exo-system's dynamics (e.g., generating forward momentum through rapid torso pitch displacement) reduces the demands on motor torque and user force. We also note that the figures referenced in this section have been grouped by motion type, but are discussed in order of relevance.

3.5.1 User Comfort

To ascertain user comfort[3], in terms of pitch acceleration, we use maximum and minimum torso pitch acceleration values ($Max: 494\frac{deg}{s^2}$, and $Min: -660\frac{deg}{s^2}$) reported as comfortable by the nominal user. The torso pitch acceleration for both motions, in all domains, is below these thresholds. During the sitting domain, for both chair-to-stand and chair-to-crouch-to-stand, the support of the exo-system transitions from the chair, user, and the feet—with the chair initially supporting the majority of the weight—to the feet and the user. Figure 3.5 and Figure 3.10 depict the chair and feet GRF for chair-to-stand and chair-to-crouch-to-stand respectively. Figures 3.6 and 3.11 show that the user force is always below 25.5 N in each hand, in all domains, for both motions.

3.5.2 Chair-to-stand

The optimal chair-to-stand trajectory respects left-right symmetry for the majority of the motion. The majority of the joint displacement occurs in the sagittal knee and hips, though all of the actuated sagittal joints do contribute. The joint and torque trajectories shown in Figure 3.7 and Figure 3.8, respectively, respect the constraints imposed in the optimization; the henke and sagittal ankle both hit their optimization lower bounds during the motion. The entire chair-to-stand motion takes four seconds to complete, with the majority of the time spent in the standing domain (three seconds). The ZMP, shown in Figure 3.4, stays well within the support polygon throughout the entire motion, and is contained in the support polygon of the feet at the end of the motion. With an absolute maximum torso pitch acceleration of 58.6236 $\frac{deg}{s^2}$ and maximum norm of the user force of 16.6 N for each hand, the chair-to-stand motion is comfortable for the user. Figure 3.6 depicts the spatial user force.

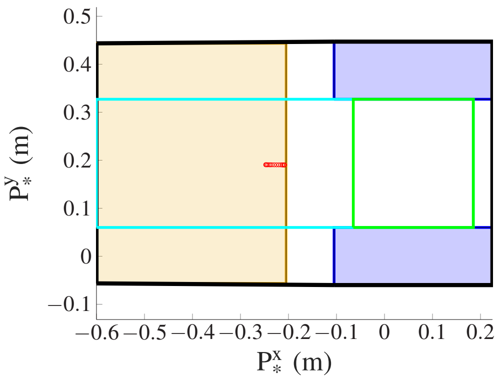
[3]To evaluate user comfort, we use the pitch acceleration of the torso and the user force. These indexes were chosen based on a previous experiment done by one of the authors, Mungai.

3.5.3 Chair-to-crouch-to-stand

In contrast to the chair-to-stand trajectory, the chair-to-crouch-to-stand trajectory was allowed to be asymmetric; see Appendix. A. Chair-to-crouch-to-stand has a longer duration at approximately six seconds. Furthermore, all of the motor joints (not just those in the sagittal plane) contribute a significant amount to the trajectory. Similar to chair-to-stand the joint and torque trajectories for chair-to-crouch-to-stand respect the optimization and hardware bounds. The left frontal hip and sagittal ankles both hit their optimization bounds. The sagittal ankle, left henke ankle, and transverse hip motors hit their respective motor torque bounds imposed in the optimization. However, as previously mentioned, this can be rectified by increasing the user force. The torque, and joint profiles are depicted in Figure 3.12 and Figure 3.13. The asymmetry observed in the joint and torque profiles result in the ZMP, shown in Figure 3.9, being off centered in the support polygon. The ZMP motion along the y-axis observed in the standing extend domain is a result of the CoM constraint implemented at the end of the domain. The maximum norm of the user force per hand (see Figure 3.11) and absolute maximum torso pitch acceleration for the entire motion are 25.37 N and 106.2 $\frac{deg}{s^2}$ respectively. Even though the absolute maximum torso pitch acceleration for the entire chair-to-crouch-to-stand motion is larger than that of chair-to-stand, the chair-to-crouch-to-stand motion is still comfortable for a user.

3.6 Torque Distribution for Over-actuated Systems

Along the nominal (feasible) motions determined through optimization, the conditions in (3.23) for the contact and virtual constraints are satisfied, as well as the design constraints given in Tables A.3, A.4, A.5, and A.6 of Appendix A. To enable the exoskeleton to safely perform a stand up motion and come to a stop, even in off-nominal conditions, two controllers of similar architecture are designed and implemented.


(a) Sitting Domain

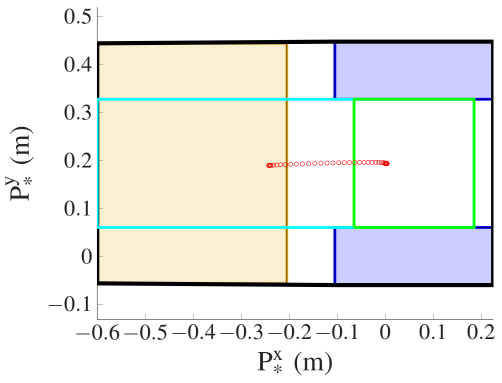
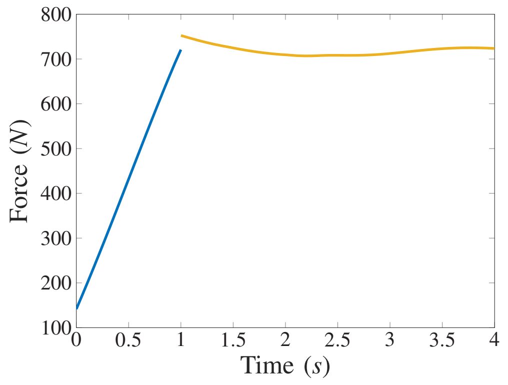

(b) Standing Domain

Figure 3.4: **Chair-to-Stand:** The evolution of the *ZMP (red circles)* from the user being in contact with the chair to letting go. The brown rectangle and the blue rectangles are the chair and feet respectively. The bold black line encompasses the support polygon of the chair and the feet, while the cyan rectangle depicts the support polygon, SP_{both_opt}, that is used in optimization. The green rectangle represents SP_{feet_opt}; it is the target location for the ZMP at the end of the standing domain when the user is no longer in contact with the chair.


(a) Right Foot Vertical GRF

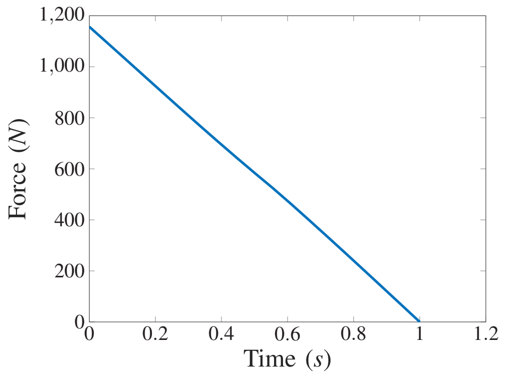
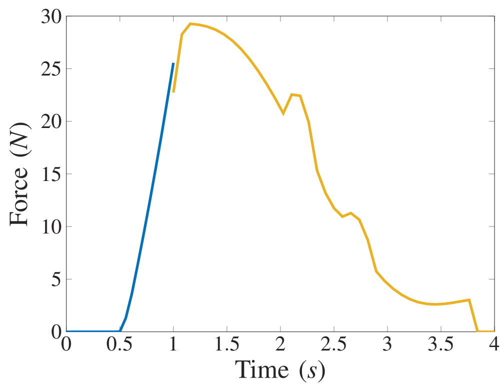
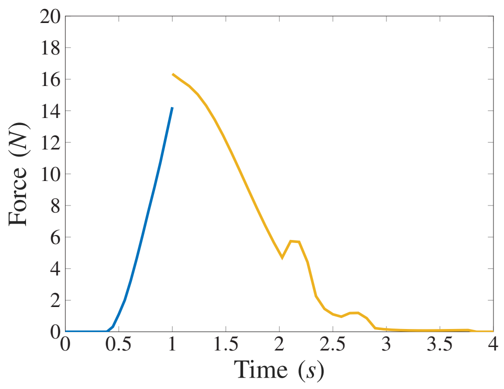

(b) Chair Vertical Force

Figure 3.5: **Chair-to-Stand:** The *optimal vertical contact forces* show a smooth transfer of weight from the chair to the feet. The left foot vertical GRF is omitted because it is similar to the right foot vertical GRF. The blue and yellow lines are for the sitting and standing domain respectively.



(a) Spatial User Force- X



(b) Spatial User Force- Z

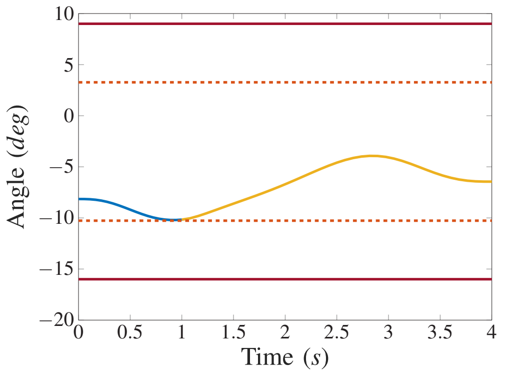
Figure 3.6: **Chair-to-Stand:** The *optimal spatial force* the user needs to provide with both arms along the x and z axes. The user force along the y axis is omitted because it is zero throughout the motion. The blue and yellow lines are for the sitting and standing domain respectively. The maximum spatial user force required for the entire motion along the x and z axes are small, since they amount to about 3 kg and 1.6 kg respectively.

The first controller that tracks the desired virtual constraints obtained from an optimal trajectory during the initial stages of standing was partially designed in Section 3.4.1 and will be completed here; it will be be called the standing up controller, or SU for short. The second controller is active in the final stage of standing and is designed to achieve a constant set-point. This controller has not been discussed yet. It will be called the standing in place controller, or SP for short. We will refer to the final stage of standing where the SP controller is active as the stopping domain.

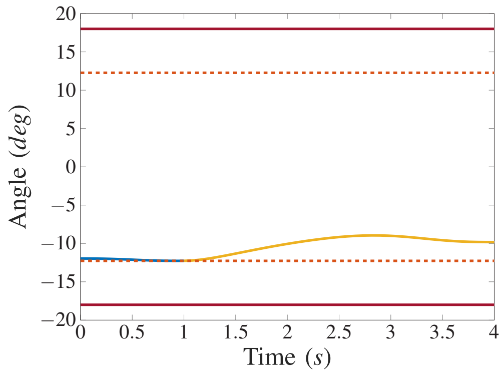
A key issue for each of these two controllers is that due to the large number of contact constraints, the system of equations (3.23) is underdetermined for the motor torques and is hence an over-actuated system. This will be addressed through a real-time quadratic program (QP), as in [116–119]

3.6.1 Design of the SU Controller

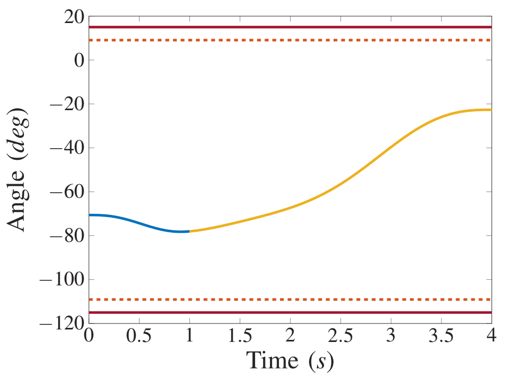
Our objective is to select at each time instance, t, the motor torques $u(t)$ of minimum norm such that:



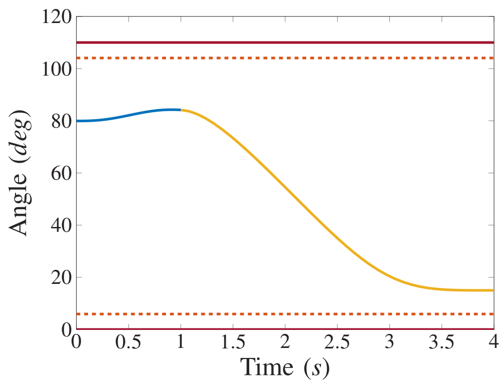
(a) Left Sagittal Ankle



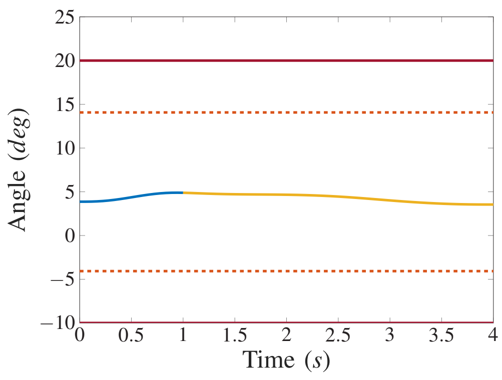
(b) Left Henke Ankle



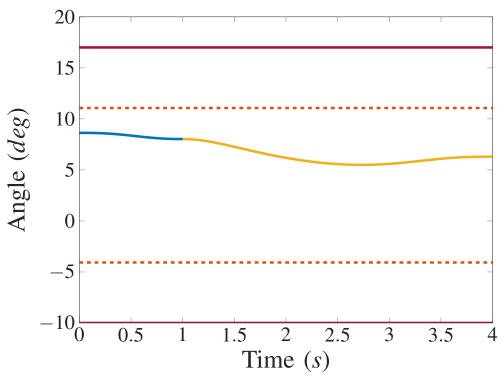
(c) Left Sagittal Hip



(d) Left Sagittal Knee

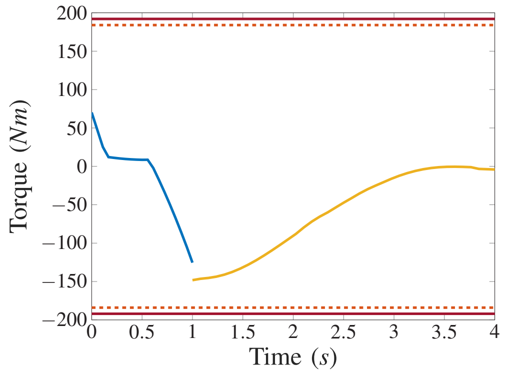


(e) Left Transverse Hip

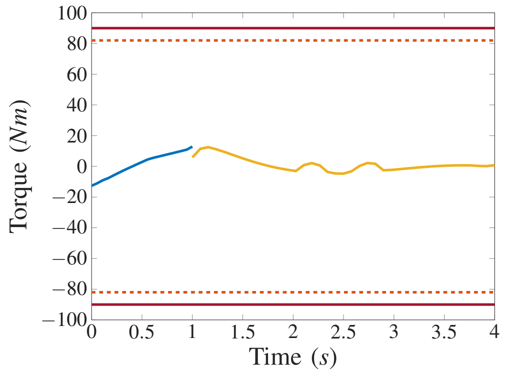


(f) Left Frontal Hip

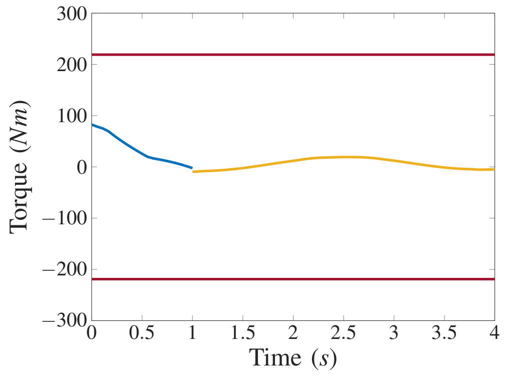
Figure 3.7: **Chair-to-Stand:** The *optimal joint trajectories* show that the joints respect the optimization and hardware constraints throughout the entire motion. The right joint trajectories are omitted because the chair-to-stand motion is symmetric. The blue and yellow lines are for the sitting and standing domain respectively, while the red and dashed orange line depict the hardware and optimization bounds respectively.



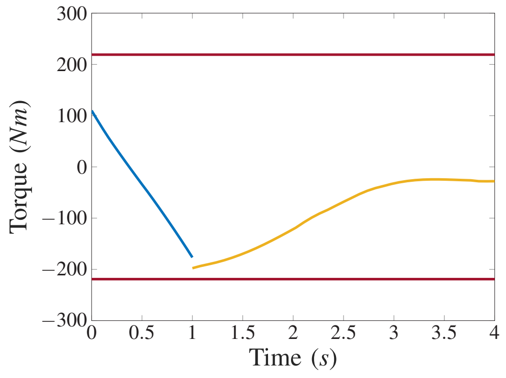
(a) Left Sagittal Ankle



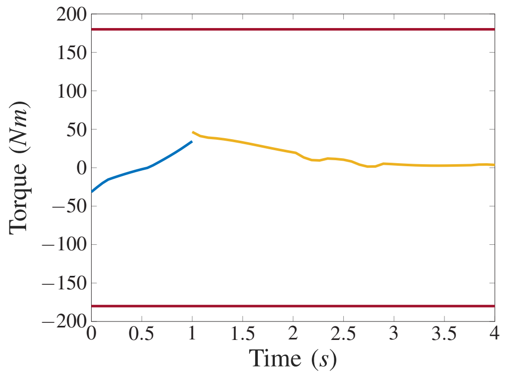
(b) Left Henke Ankle



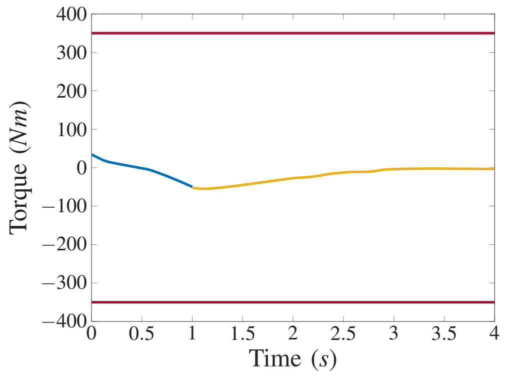
(c) Left Sagittal Hip



(d) Left Sagittal Knee

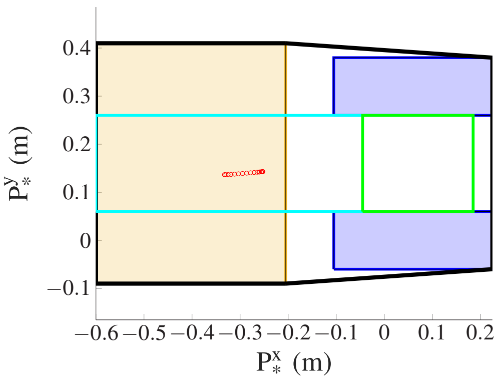


(e) Left Transverse Hip

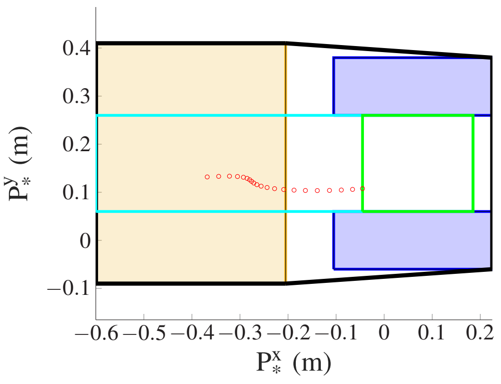


(f) Left Frontal Hip

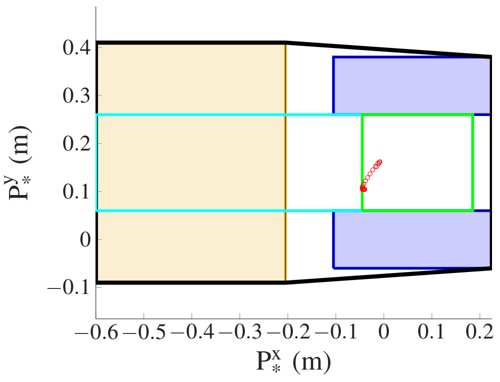
Figure 3.8: **Chair-to-Stand:** The *optimal torque trajectories* show that the motor torque bounds are respected throughout the entire motion. The right torque trajectories are omitted because the chair-to-stand motion is symmetric. The blue and yellow lines are for the sitting and standing domain respectively, while the red line depicts the maximum torque bounds. For the ankle torques, we also include the nominal torque bounds, represented by the dashed orange lines, since those are the bounds used in optimization.



(a) Sitting Domain

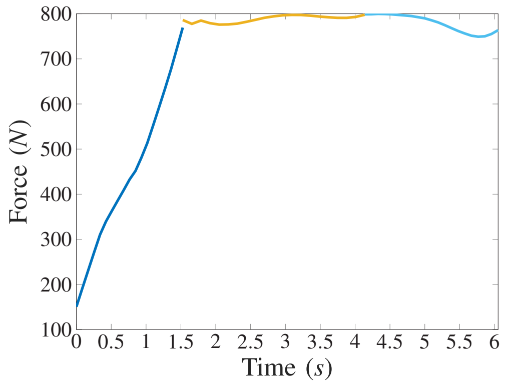


(b) Standing Shift Domain

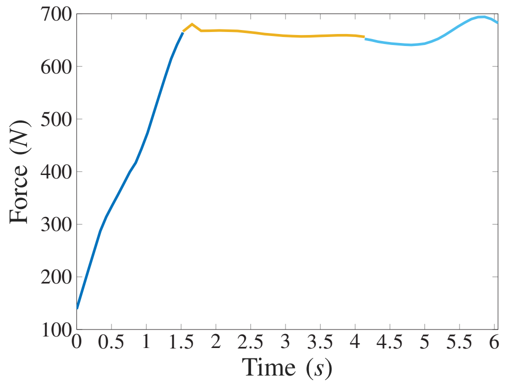


(c) Standing Extend Domain

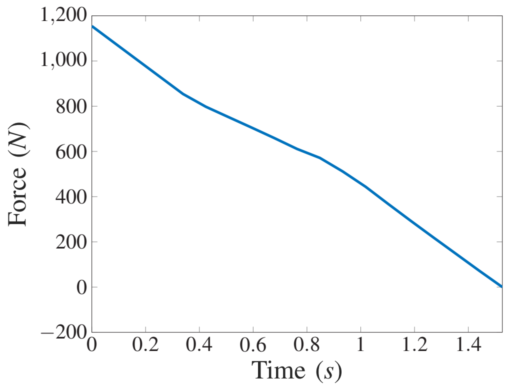
Figure 3.9: **Chair-to-Crouch-to-Stand:** The evolution of the *ZMP (red circles)* from the user being in contact with the chair to letting go. The brown rectangle and the blue rectangles are the chair and feet respectively. The bold black line encompasses the support polygon of the chair and the feet, while the cyan rectangle depicts the support polygon, SP_{both_opt}, that is used in optimization. The green rectangle represents SP_{feet_opt}; it is the target location for the ZMP at the end of the standing shift domain and throughout the standing extend domain. We note that the user force is zero at the end of the standing shift domain and for the entirety of the standing extend domain.



(a) Right Foot Vertical GRF

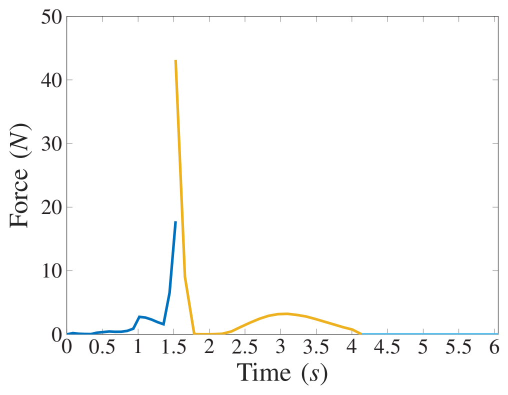


(b) Left Foot Vertical GRF



(c) Chair Vertical Force

Figure 3.10: **Chair-to-Crouch-to-Stand:** The *optimal vertical contact forces* show a smooth transfer of weight from the chair to the feet. The blue, yellow, and light blue lines are for the sitting, standing shift, and standing extend domains respectively.


(a) Spatial User Force- X

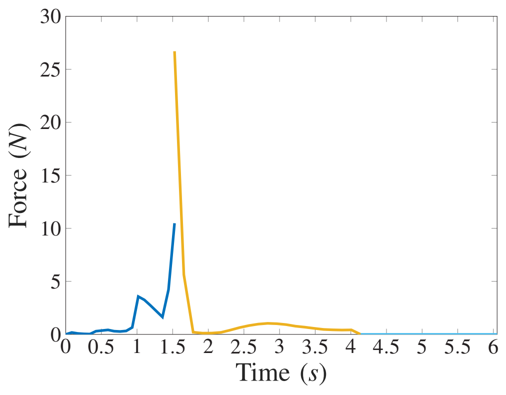

(b) Spatial User Force- Z

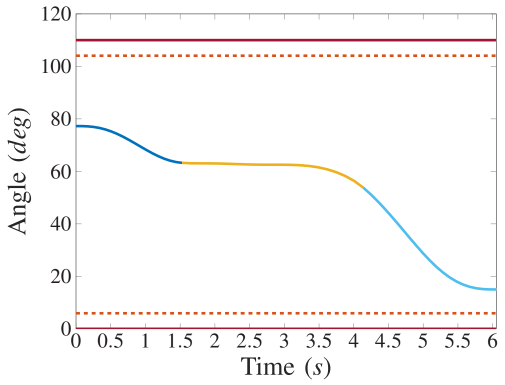
Figure 3.11: **Chair-to-Crouch-to-Stand:** The *optimal spatial force* the user needs to provide with both arms along the x and z axes. The user force along the y axis is omitted because it is zero throughout the motion. The blue, yellow, and light blue lines are for the sitting, standing shift, and standing extend domains respectively. The maximum spatial user force required for the entire motion along the x and z axes are small, since they amount to about 4.4 *kg* and 2.7 *kg* respectively.

(a) $\ddot{y} + K_d\dot{y} + K_p y = 0$, and thus disturbances to the nominal motion are attenuated; and

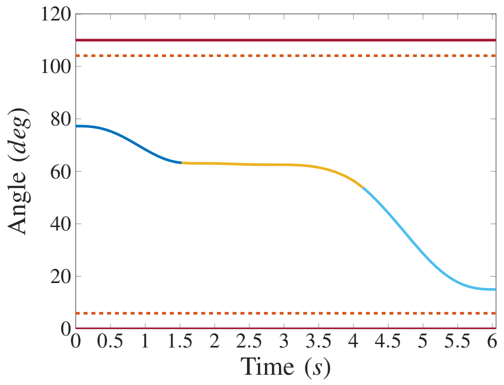
(b) the user force, $\zeta(t)$, that is required to satisfy the contact constraints and the above virtual constraints remains as close as possible to the nominal force, $\zeta^*(t)$, that was determined in optimization, even in off-nominal conditions.

The motivation in (b) is that we do not want the controller relying on the user to make corrections to the trajectory. On the other hand, if torque or joint constraints limit the controller's ability to meet all of the contact constraints, we want the user to be able to contribute, but only if absolutely necessary. To meet the above objectives, we rewrite (3.23) in terms of motor torques u and user force ζ as

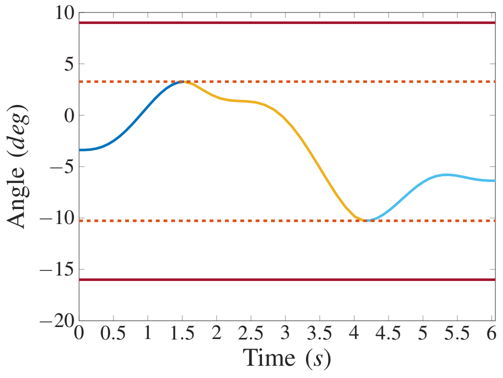
$$A_{eq} \begin{bmatrix} u \\ \zeta \end{bmatrix} = b_{eq}, \tag{3.40}$$



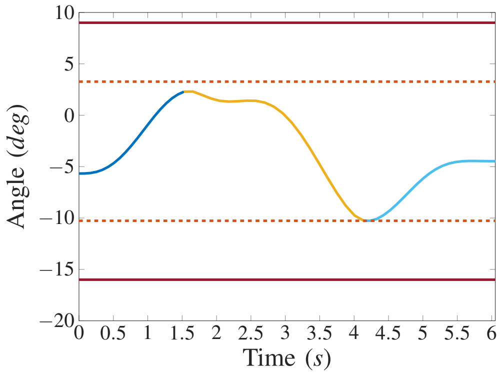
(a) Left Sagittal Knee



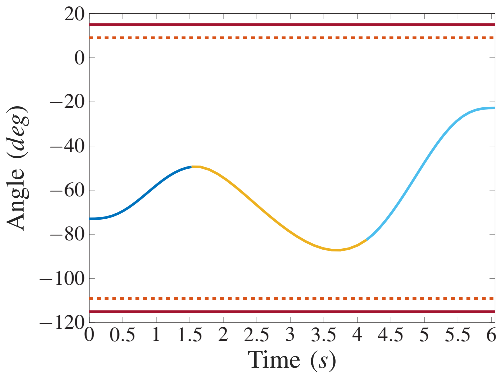
(b) Right Sagittal Knee



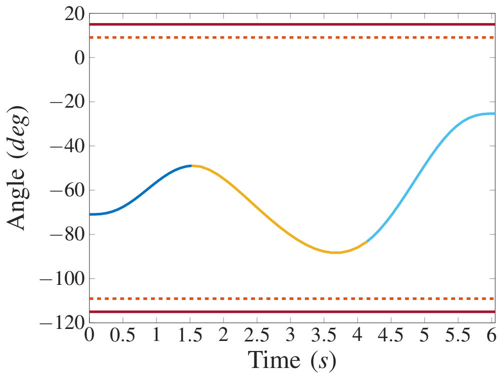
(c) Left Sagittal Ankle



(d) Right Sagittal Ankle

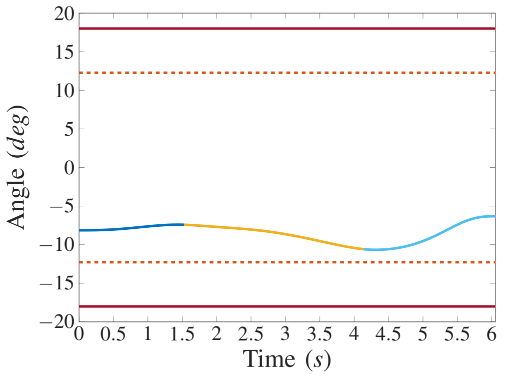


(e) Left Sagittal Hip

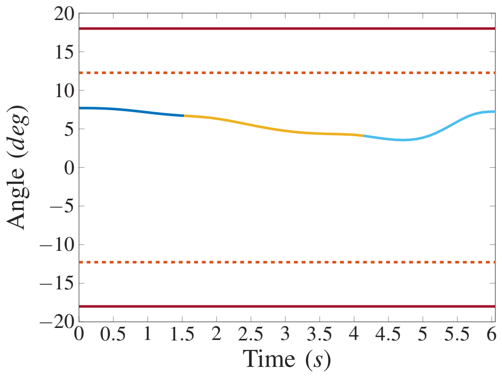


(f) Right Sagittal Hip

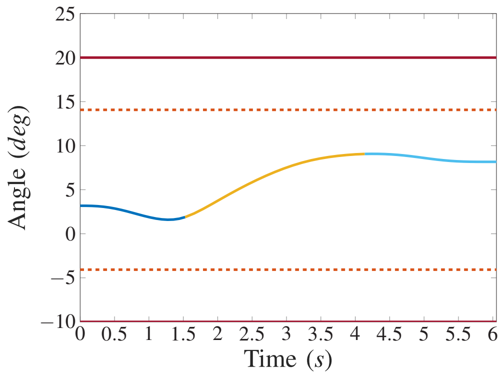
Figure 3.12: **Chair-to-Crouch-to-Stand:** The *optimal joint trajectories* show that the joints respect the optimization and hardware constraints throughout the entire motion. The blue, yellow, and light blue lines are for the sitting and standing shift and standing extend domains respectively, while the red and dashed orange line depict the hardware and optimization bounds respectively.



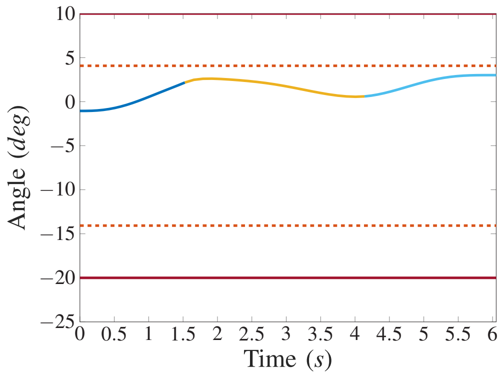
(g) Left Henke Ankle



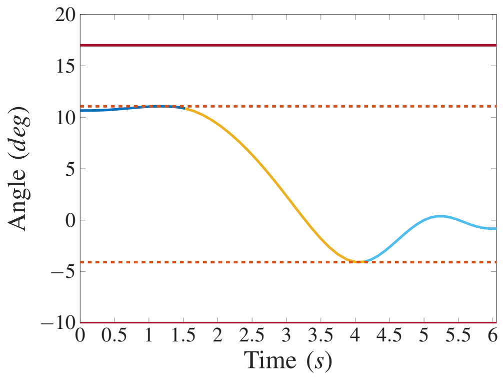
(h) Right Henke Ankle



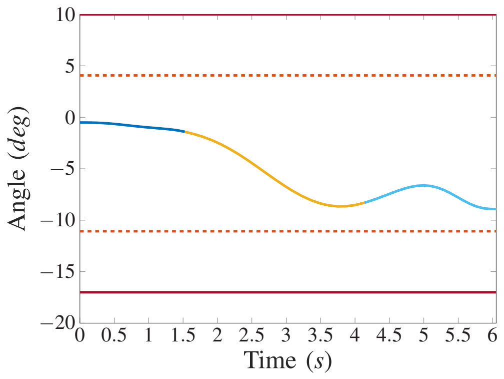
(i) Left Transverse Hip



(j) Right Transverse Hip

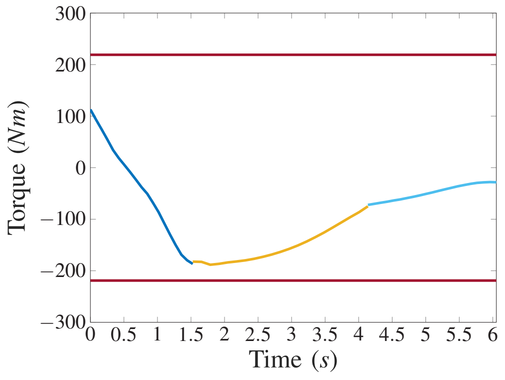


(k) Left Frontal Hip

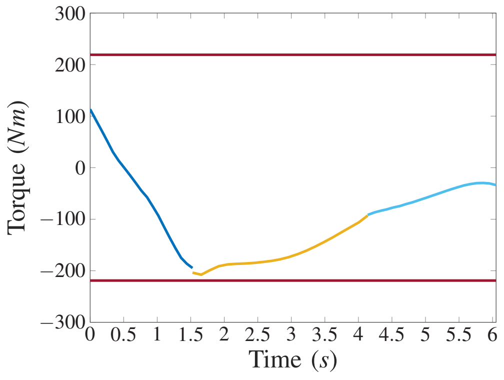


(l) Right Frontal Hip

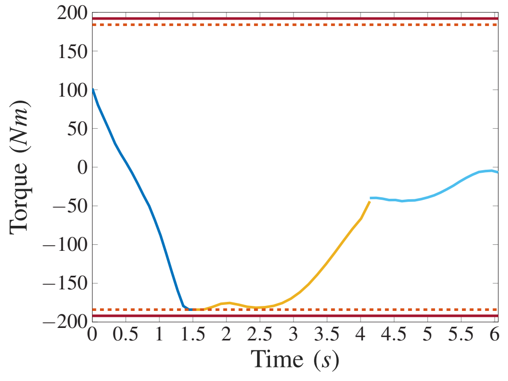
Figure 3.12: **Chair-to-Crouch-to-Stand:** The *optimal joint trajectories* (Cont.)



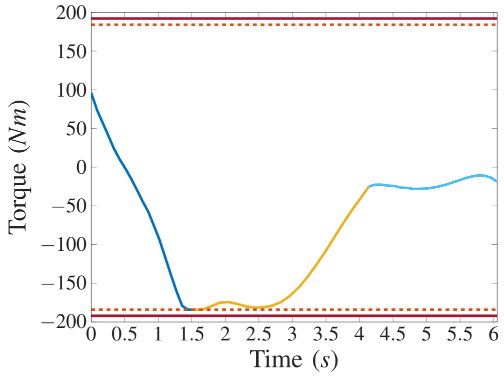
(a) Left Sagittal Knee



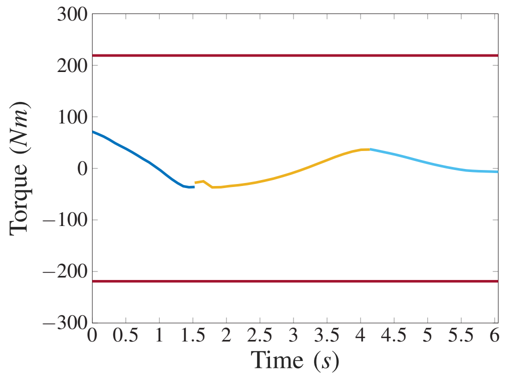
(b) Right Sagittal Knee



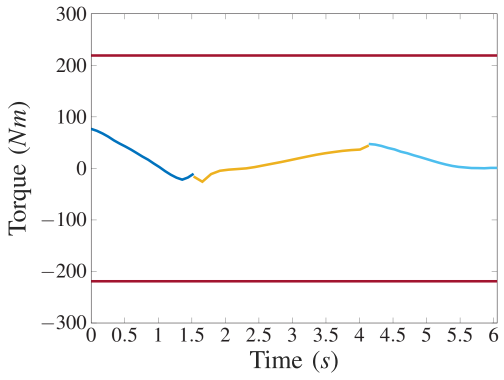
(c) Left Sagittal Ankle



(d) Right Sagittal Ankle

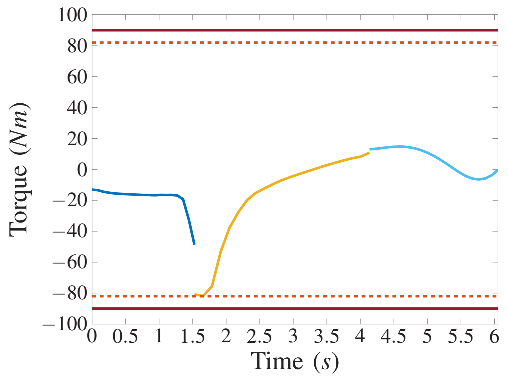


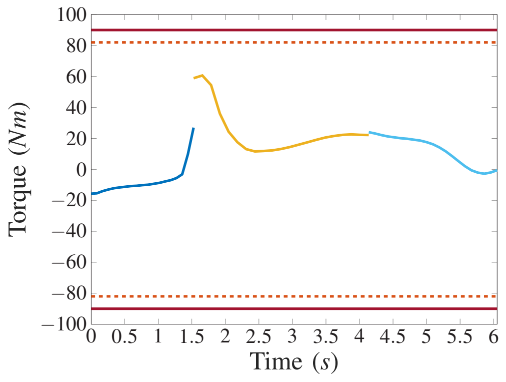
(e) Left Sagittal Hip

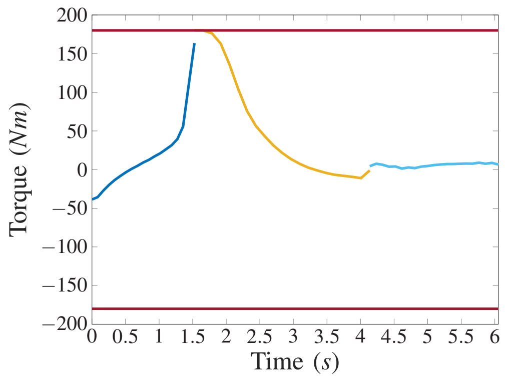


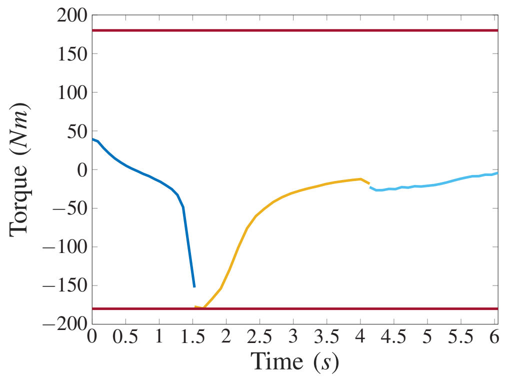
(f) Right Sagittal Hip

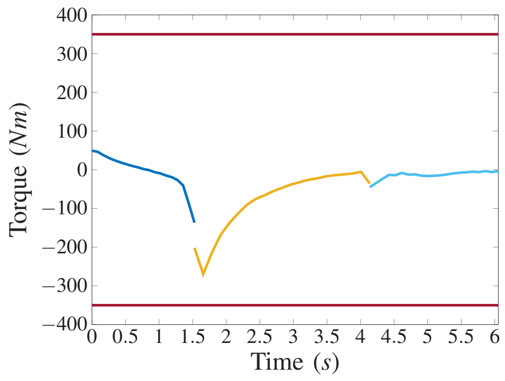
Figure 3.13: **Chair-to-Crouch-to-Stand:** The *optimal torque trajectories* show that the motor torque bounds are respected throughout the entire motion. The blue, yellow, and light blue lines are for the sitting, standing shift, and standing extend domains respectively while the red line depicts the maximum torque bounds. For the ankle torques, we also include the nominal torque bounds, represented by the dashed orange lines, since those are the bounds used in optimization.


(g) Left Henke Ankle


(h) Right Henke Ankle


(i) Left Transverse Hip


(j) Right Transverse Hip


(k) Left Frontal Hip

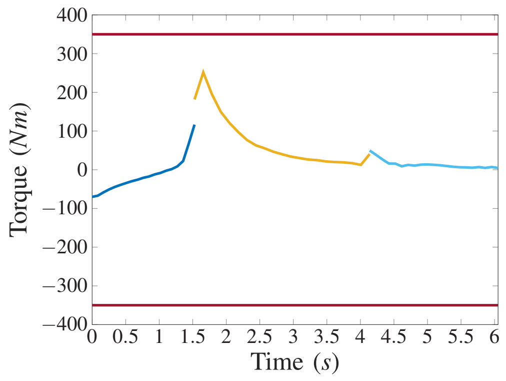

(l) Right Frontal Hip

Figure 3.13: **Chair-to-Crouch-to-Stand:** The *optimal torque trajectories* (Cont.)

where,

$$\begin{aligned}
A_{eq} &= \begin{bmatrix} J_h D^{-1}(\mathbf{I} - J^\intercal \chi J D^{-1})B & J_h D^{-1}(\mathbf{I} - J^\intercal \chi J D^{-1})J_{ext}^\intercal \end{bmatrix} \\
b_{eq} &= -J_h D^{-1}[(\mathbf{I} - J^\intercal \chi J D^{-1})(F_v) - J^\intercal \chi \dot{J}\dot{q}] - \dot{J}_h \dot{q} + Y \\
\chi &= (J D^{-1} J^\intercal)^{-1} \\
F_v &= -C\dot{q} - G \\
Y &= \ddot{y}_d - K_d \dot{y} - K_p y.
\end{aligned}$$

At time $\bar{t}$, let $u^*(\bar{t})$ and $\zeta^*(\bar{t})$ be the control signal and user force along the optimal trajectory. The applied control signal and "estimated user force" for the SU controller will be determined by the following QP,

$$\begin{aligned}
\begin{bmatrix} u(\bar{t}) \\ \zeta(\bar{t}) \end{bmatrix} := \underset{u\in\mathbb{R}^{12},\zeta\in\mathbb{R}^3}{\arg\min} \quad & ||u - u^*(\bar{t})||_2^2 + \alpha||\zeta - \zeta^*(\bar{t})||_2^2 \\
\text{subject to} \quad & \\
A_{eq}(q(\bar{t})) \begin{bmatrix} u \\ \zeta \end{bmatrix} &= b_{eq}(q(\bar{t}), \dot{q}(\bar{t})) \\
\{P_*^x, P_*^y\} &\subseteq SP \\
F_\bullet^z &\geq 0 \\
|F_\bullet^x| &\leq \mu \frac{F_\bullet^z}{\sqrt{2}} \\
|F_\bullet^y| &\leq \mu \frac{F_\bullet^z}{\sqrt{2}} \\
|M_\circ^z| &\leq \gamma F_\circ^z \\
u_{lb} \leq u &\leq u_{ub}
\end{aligned} \tag{3.41}$$

The inclusion of $||\zeta - \zeta^*||_2^2$ in the cost ensures feasibility of the constraints in (3.41), while a large value of $\alpha > 0$ encourages solutions where the user force ζ "*assumed by the controller*" remains close to its designed value. Of course, the user of the exoskeleton has no knowledge of this calculation and will assist in the standing process "as best as they can". We choose to implement the user force in this manner because we assume that by using the exoskeleton several times the user will eventually learn to provide the external force needed at the "right" time. In other words, after several attempts the user provided assistance should be close to the optimal value assumed by the controller. To better mimic the user force from the user, one can take advantage of the various learning algorithms to design a controller for the user force which interacts with QP I/O control; Aroche et al. [20] approximated the user force using an iterative learning control. Robustness of the control actions to varying user force will be checked in the results section. To ensure that the solutions of the QP are feasible even in the presence of perturbations, the ZMP is constrained to be within the appropriate support polygon. We first define SP as the compact set of all the points in the desired support polygon. We then require P_*^x and P_*^y to be inside the set SP at all times; this is done by using half planes.

3.6.2 Design of the SP Controller

The set-point for the SP controller regulates the variables

$$y_d = \begin{bmatrix} \mathrm{CoM_X} \\ \mathrm{CoM_Y} \\ \frac{q_{\mathrm{knee}}^{\mathrm{left}} + q_{\mathrm{knee}}^{\mathrm{right}}}{2} \\ \frac{q_{\mathrm{knee}}^{\mathrm{left}} - q_{\mathrm{knee}}^{\mathrm{right}}}{2} \\ \theta \\ \psi \end{bmatrix} \tag{3.42}$$

to constant values. Here, ψ is the torso yaw angle, $(q_{\mathrm{knee}}^{\mathrm{left}} + q_{\mathrm{knee}}^{\mathrm{right}})/2$ is directly related to the average height of the hips, and $(q_{\mathrm{knee}}^{\mathrm{left}} - q_{\mathrm{knee}}^{\mathrm{right}})/2$ sets the relative height of each hip.

The SP controller is expressed as a QP in the form

$$
\begin{aligned}
\begin{bmatrix} u(\bar{t}) \\ \zeta(\bar{t}) \end{bmatrix} := \underset{u\in\mathbb{R}^{12},\zeta\in\mathbb{R}^{3}}{\arg\min} \quad & ||u||_2^2 + \alpha||\zeta||_2^2 \\
\text{subject to} \quad & \\
A_{eq}(q(\bar{t}))\begin{bmatrix} u \\ \zeta \end{bmatrix} \quad &= \quad \tilde{b}_{eq}(q(\bar{t}),\dot{q}(\bar{t})) \\
\{P_*^x, P_*^y\} \quad &\subseteq \quad SP_{feet} \\
F_\bullet^z \quad &\geq \quad 0 \\
|F_\bullet^x| \quad &\leq \quad \mu\frac{F_\bullet^z}{\sqrt{2}} \\
|F_\bullet^y| \quad &\leq \quad \mu\frac{F_\bullet^z}{\sqrt{2}} \\
|M_\circ^z| \quad &\leq \quad \gamma F_\circ^z \\
u_{lb} \leq \quad & u \quad \leq u_{ub} \\
q_{lb} \leq \quad & q(t+1) \quad \leq q_{ub}
\end{aligned}
\tag{3.43}
$$

Remarks: The SP controller is similar to the SU controller except for the addition of an integral term for the average knee angle virtual constraint and state bound constraints for the exo-system's position, q. These additional constraints guarantee that the exoskeleton stops at the desired height without violating any joint bounds. The bounds for q could have been implemented in the SU controller as well, however, since the SU controller tracks the optimal trajectory via virtual constraints, these constraints were inactive along the nominal trajectory. Note that since the SP controller tracks a set point, $u^*(\bar{t}) = 0$ and $\zeta^*(\bar{t}) = 0$.

3.6.3 Stability Analysis of the Two Standing Controllers

The overall controller for each motion is hybrid, consisting of (short-duration) transient phases of trajectory tracking that guide the exo-system from the chair to a transition point where a standing controller, SP, takes over and must assure steady-state stability. Here, we analyze the local exponential stability of the standing controller's equilibrium point, which we denote by x_{eq}.

For each of the chair-to-stand and chair-to-crouch-to-stand motions, the Jacobian linearization of the closed-loop floating base model about x_{eq} is estimated using symmetric differences. So that the contact constraints are respected, perturbations are only applied in the null space of the contact Jacobians. The control algorithm automatically computes corrections[4] to the nominal control signal when the perturbations are applied to the equilibrium point, x_{eq}. The resulting matrix is then orthogonally projected to the null space of the contact Jacobians, giving us a square matrix A_{lin} corresponding to the Jacobian of the reduced-order model one would obtain if the contact constraints were eliminated. For both sit-to-stand motions, chair-to-stand and chair-to-crouch-to-stand, the eigenvalues of A_{lin} have negative real parts. Therefore, from Theorem 4.6 and 4.7 in Khalil [67], we conclude that x_{eq} is a locally exponentially stable equilibrium point.

The above analysis shows that a quadratic Lyapunov function exists for the equilibrium point, and hence there is an accompanying open set that forms a domain of convergence for x_{eq}. The feedback controller for the transient domains, SU, is based on trajectory tracking via the I/O linearizing controller and a QP. The I/O linearizing controller is differentiable and hence locally Lipschitz continuous. The QPs are feasible for the nominal motions and have positive definite costs and differentiable constraints. Hence, they are locally Lipschitz continuous as well. Therefore, there exists an open set about the nominal starting point for which all initial conditions are steered to the domain of attraction of the standing controller. This completes the stability analysis.

Remark: It is well known that performing the above analysis would provide very conservative estimates of the domain of attraction. One could attempt to use other methods such as reachability analysis or barrier functions, however, with the current

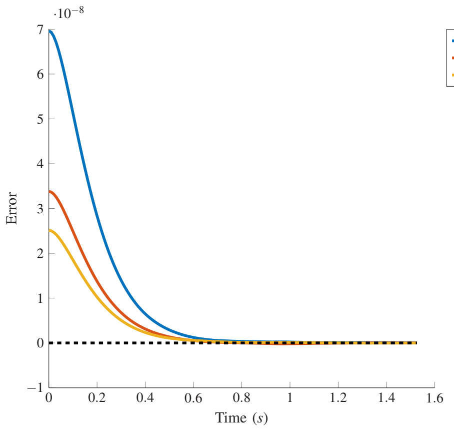
[4]Analytically computing the Jacobian of the floating-base dynamical model would have been straightforward. However, doing so would have been less straightforward for the QP-based controller.

techniques available it is not feasible to analytically calculate the range of operability for high-order nonlinear systems. Therefore, to circumvent these obstacles, we find the range of operability of the proposed closed-loop system using robustness tests.

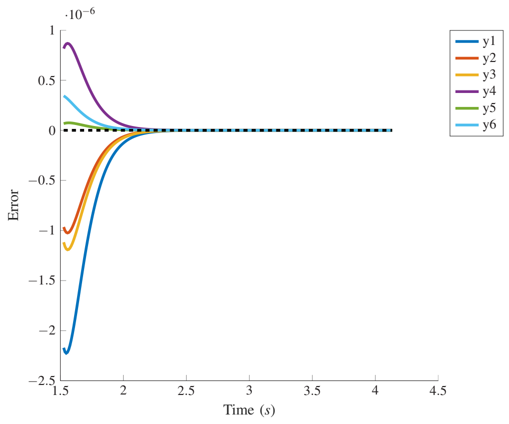
3.6.4 Nominal Simulation of the Two Standing Controllers

Here we present the nominal closed-loop standing behaviors for both chair-to-stand and chair-to-crouch-to-stand of the exo-system under the action of the SU and SP controllers during nominal conditions. The simulation is performed using the ideal simulator in FROST. [5] The SU controller is able to maintain low tracking error and the SP controller is able to achieve the desired set point values. The error plots for chair-to-crouch-to-stand and chair-to-stand can be found in Figure 3.14 and Figure 3.15 respectively.

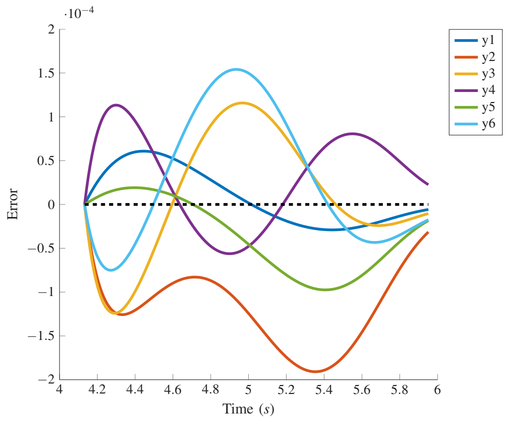
[5]FROST uses Matlab's ode45 to simulate the dynamics of the given hybrid system.



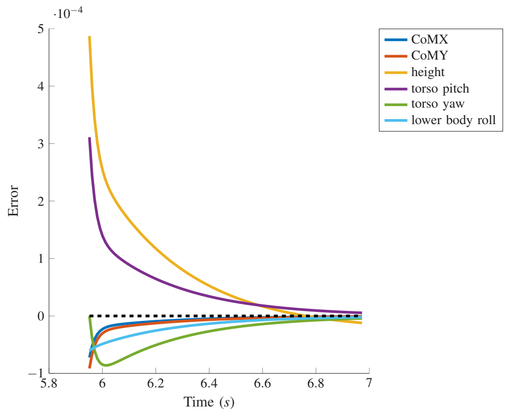
(a) SU Sitting Tracking Error



(b) SU Standing Shift Error

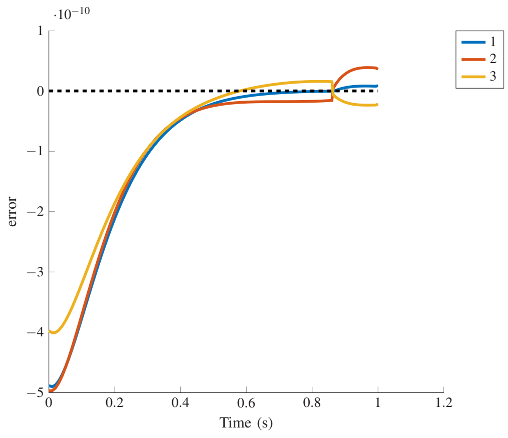


(c) SU Standing Extend Error

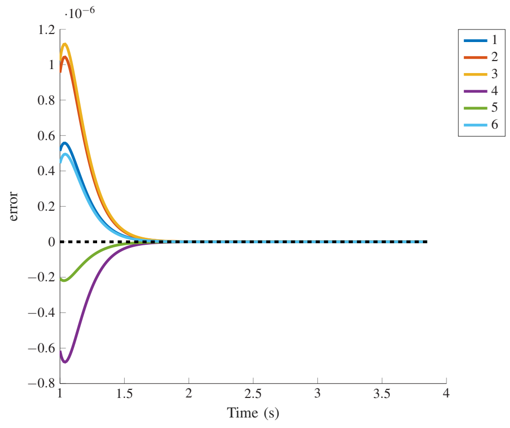


(d) SP Error

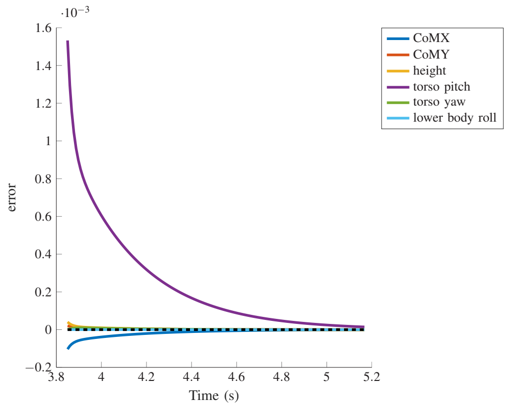
Figure 3.14: **Chair-to-Crouch-to-Stand** closed-loop *nominal tracking error* plots for the standing up (SU) and standing in place controllers (SP). The tracking error of the virtual constraints (y_i) in (a)-(c) are dimensionless. The error of the positions, and angles of the virtual constraints displayed in (d) are measured in meters and radians.



(a) SU Sitting Tracking Error



(b) SU Standing Error



(c) STS SP Error

Figure 3.15: **Chair-to-Stand** closed-loop *nominal tracking error* plots for the standing up (SU) and standing in place controllers (SP). The tracking error of the virtual constraints (y_i) in (a)-(c) are dimensionless. The error of the positions, and angles of the virtual constraints displayed in (d) are measured in meters and radians.

3.7 Robustness Tests

This section assesses the ability of the two closed-loop behaviors to tolerate a range of perturbations. The perturbations are: (1) different users in the exoskeleton (this perturbation can be thought of as a weight and height perturbation. Weight is the primary perturbation we will focus on.), (2) variations to chair height (both higher and lower than the nominal chair height), (3) zero user force which creates a discrepancy between the external force the controller expects and the one that's provided, (4) spasticity in the knee joints, and (5) asymmetric motor torque outputs which will help us understand a controller's sensitivity to asymmetry.

These robustness tests will help us ascertain differences in the two types of standing motions. They will also help us determine how many nominal trajectories one may need in a gait library to expand the controllers' range of operability. To make the controllers more impervious to perturbations and for ease of implementation, the controller and simulation architecture are modified slightly. These modifications are discussed in detail in Appendix B.

For the reader's convenience, we recall the following values:

- Nominal Chair Height: 0.6 m
- Nominal User Weight: 73 kg
- Nominal μ_{chair}: 0.5
- Nominal μ_{feet}: 0.9
- Atalante's designed user range: $1.55-1.90$ m and $50-90$ kg

3.7.1 Criteria for Success and Summary of the Results

The criteria that we use to determine the success of the standing motions under perturbations are: (1) tracking and steady state error of the SU and SP controllers respectively, (2) torso pitch acceleration, (3) user force expected by the controller, (4) ZMP constraint violation, (5) friction constraint violations (we will consider even small violations of the feet friction constraints to be a failure), (6) joint angle limit violation, and (7) motor limit violation. The tracking and steady state error allows us to confirm whether or not the desired standing motion has been achieved by the controller. A relatively low torso pitch acceleration and required user force ensure user comfort. The maximum ($494\frac{deg}{s^2}$) and minimum ($-660\frac{deg}{s^2}$) torso pitch acceleration thresholds used for optimization are utilized here as well. The ZMP and friction constraints, and joint angle and motor limits ensure the feasibility of the motion in terms of stability and hardware limitations. Note that the friction constraint consists of both the friction cone and torsional friction constraint, and that the joint angle limits are only explicitly implemented in the SP controller. The friction cone constraint is calculated using $|F_{\bullet}^{\circ}| - \mu\frac{F_{\bullet}^{z}}{\sqrt{2}} \leq 0$, where $F_{\bullet}^{\circ}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair.

With these criteria in mind, the results that we will show indicate that both motions are equally capable at handling weight variations and user force disparities. However, the chair-to-crouch-to-stand motion is better at handling asymmetric motions and spasticity, while the chair-to-stand motion can handle a broader range of chair heights. Under all of the perturbations, the closed-loop trajectories of both motions respect the ZMP and motor limit constraints. A large friction coefficient between the chair and the exo-system can be used to increase the robustness of the motions against torque asymmetry, spasticity, and variations to the user characteristics. The data that led to these conclusions is presented in the following subsections.

3.7.2 User Characteristics

URDFs were generated for additional users to test the sensitivity of the motions to weight and height. The weight and height of these additional users can be found in

Table 3.1. Note that our analysis will focus on the weight of the user. The various users in the exoskeleton result in various plant models. To enforce this perturbation, we introduce a discrepancy in the plant model used for simulation, while the controller is always based on the nominal model.

Table 3.1: The height and weight of the various users used to study the closed-loop behavior of the chair-to-stand and chair-to-crouch-to-stand motions

Name	Weight (kg)	Height (m)
User 1	54	1.62
User 2	68	1.8
Nominal	73	1.73
User 3	90	1.8

The results for both motions exhibit no joint angle violations. However, both motions violate the friction cone constraint in the sitting domain for some users. This violation occurs, even though the friction constraints are explicitly implemented in the controllers, because of the model error. With the nominal value of $\mu_{chair} = 0.5$, chair-to-stand has a friction constraint violation for User 1 and User 3. Chair-to-crouch-to-stand on the other hand, has friction constraint violations for all the non-nominal users. For User 1, the feet and chair friction constraints are violated for chair-to-stand, while only the chair friction constraint is violated for chair-to-crouch-to-stand. Due to the large chair friction cone constraint violations during chair-to-crouch-to-stand and feet friction cone constraint violation during chair-to-stand (see Table 3.2, Figure 3.16, and Figure 3.17), the SU controller is unable to successfully get the lightest user (User 1) to stand up. Even though there are other users for which the friction constraint is violated, we only consider User 1 a failed case because the violations for the other users are only observed for the chair and are fairly small. In fact, these friction constraint violations can be mitigated by the user exerting the necessary additional force or by using a chair with a higher friction coefficient. For instance, by setting $\mu_{chair} = 0.6$ and $\mu_{chair} = 0.7$ for User 3, we are able to stop the exo-system from slipping in the chair for chair-to-stand and chair-to-crouch-to-stand respectively.

With User 1 out, the maximum torso pitch acceleration for chair-to-stand occurs for User 3 during the standing domain and is 120 $\frac{deg}{s^2}$. The maximum torso pitch acceleration for chair-to-crouch-to-stand is 174 $\frac{deg}{s^2}$ and it is observed for User 2 during the standing shift domain. Even though the chair-to-stand motion has a lower maximum torso pitch acceleration, the results from both motions should be comfortable for the user. As a result, we can conclude that the performance of both motions is equal in terms of torso pitch acceleration. The torso pitch acceleration profiles can be found in Figure 3.18 and Figure 3.19. The tracking and steady state errors for both controllers for both motions are small. The resulting user force from the chair-to-stand motion is close to the nominal values. The user force for chair-to-crouch-to-stand is similar to the nominal value except for User 3 whose results require an additional user force norm of 12 N during the standing shift domain; see Figure 3.20.

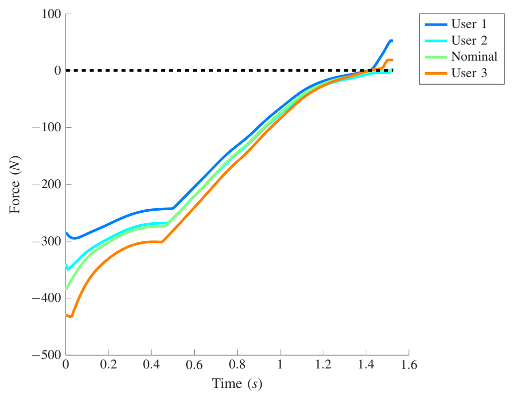
The SU and SP controllers are unable to perform a successful standing motion for users who weigh significantly less than the nominal user. The range of users that can stand up successfully is the same for both controllers, 68 kg to 90 kg. Note that 90 kg is the heaviest user that Atalante can handle. Therefore, we can conclude that both motions perform equally with various plant models. A summary of the results can be found in Table 3.2.

3.7.3 Varying Chair Height

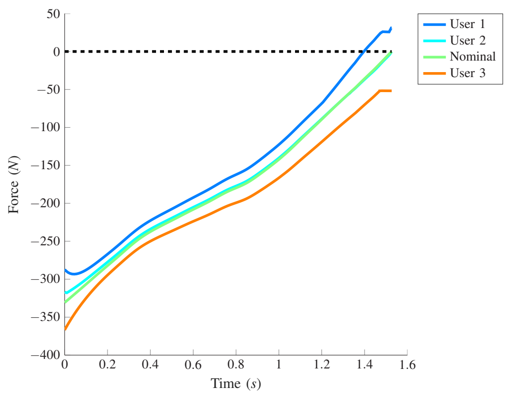
Simulations for chair heights ranging from 0.45 to 0.75 m are run for both motions. The trajectories resulting from the chair-to-stand motion violate the upper bound of the

Table 3.2: **User Characteristics:** A comparison of the closed-loop chair-to-crouch-to-stand and chair-to-stand trajectories obtained from the SU and SP controllers for different users in the exoskeleton. The characteristics of the users are described in Table 3.1. The user force displayed in the table is the total force the user needs to provide using both arms. For all the values of interest, except for the steady state error of SP, the Domain/Virtual constraint gives the domain where the maximum value occurs. For the steady state error of SP, the virtual constraints where the maximum error occurs are instead given in the Domain/Virtual constraint. The errors are displayed as maximum position and angle errors. Recall that the first three components of (3.42) are positions and the rest are angles.

	Chair-to-stand				Chair-to-crouch-to-stand			
Name	Violated	Domain/ Virtual constraint	Max value	Nominal value	Violated	Domain/ Virt Constr	Max value	Nominal value
Joint constraint (deg)	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
Friction constraint (N)	chair	Sitting	52.5 (for User 1)	$\times$	Feet Chair	Sitting	Feet:13.5 Chair:14.2 (for User 1)	$\times$
Torso pitch acceleration ($\frac{deg}{s^2}$)	$\times$	Standing extend	174.3 (for User 2)	106.1	$\times$	Sitting	-144.6 (for User 1)	58.7
User force norm (N)	$\times$	Standing shift	62 (for User 3)	50.7	$\times$	Standing	35.5 (for User 3)	33.1
Tracking error SU	$\times$	Standing extend	0.01 (for User 3)	~ 0	$\times$	Standing	0.01 (for User 3)	~ 0
Steady state error SP	$\times$	Pos : all	~ 0 m	~ 0 m	$\times$	Pos : all	~ 0 m	~ 0 m
	$\times$	Angle : $y_d(4)$	0.02 rad (for User 3)	~ 0 rad	$\times$	Angle : $y_d(4)$	0.02 rad (for User 3)	~ 0 rad



(a) Chair X Friction Constraint

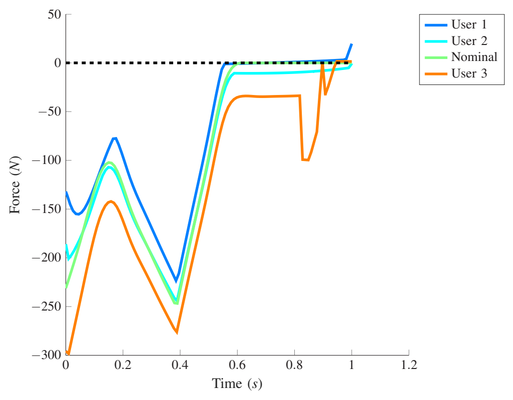


(b) Chair Y Friction Constraint

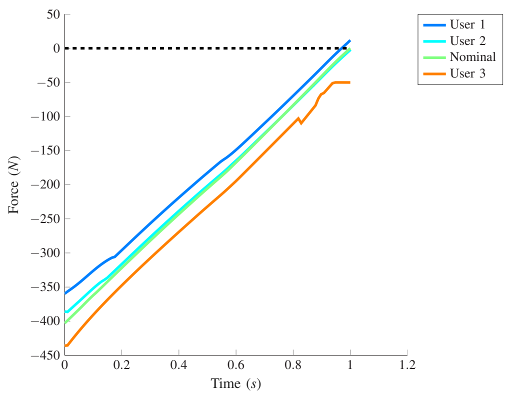
Figure 3.16: **User characteristics for Chair-to-Crouch-to-Stand:** The chair friction cone constraints during the *SITTING* domain for different users in the exoskeleton. The characteristics of the users are described in Table 3.1. The results are calculated using $|F^{\circ}_{\bullet}| - \mu \frac{F^{z}_{\bullet}}{\sqrt{2}} \leq 0$, where $F^{\circ}_{\bullet}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair. Therefore, a value above zero violates the friction constraint.

sagittal knee and hip joints for chair heights below 0.47 m; the maximum constraint violation occurs at 0.45 m and is 4.6 deg. The trajectories obtained from the chair-to-crouch-to-stand motion violate the sagittal ankle joint upper bound at chair heights greater than 0.67 m; the maximum constraint violation is observed at 0.75 m and is 8.6 deg. Therefore, the range of operability for the chair-to-stand and chair-to-crouch-to-stand motions based on joint bounds is 0.47-0.75 m and 0.45-0.67 m, respectively. Even though the chair-to-crouch-to-stand motion has a lower ranger than the chair-to-stand motion, the 2 cm difference is small enough to consider the two motions as having the same lower range. The trajectories for the joints whose limits are violated can be found in Figure 3.21 and Figure 3.22. Note that to conserve space only the respective left joint trajectories are plotted for the symmetric chair-to-stand motion. We continue our analysis using only the results within each motion's range of operability based on joint angle constraints.

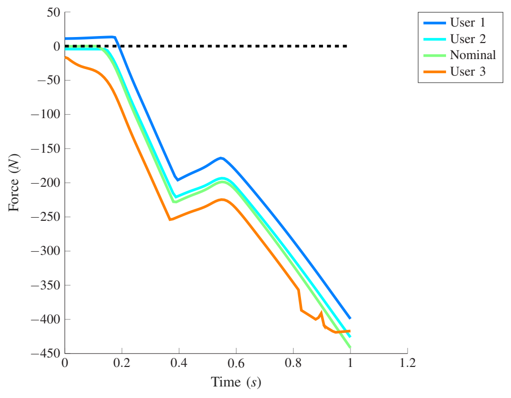
The friction constraint is respected for both motions while the required user force for both motions remains close to the nominal value. For both motions, the results



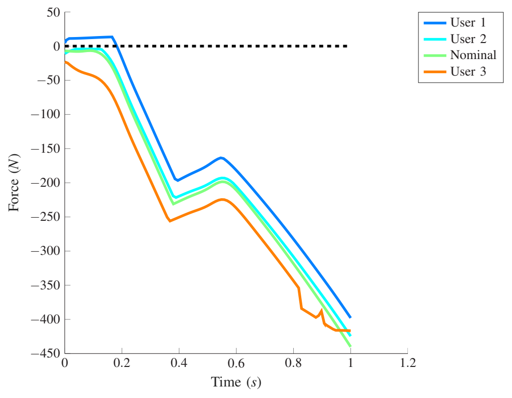
(a) Chair X-axis Friction Cone Constraint



(b) Chair Y-axis Friction Cone Constraint

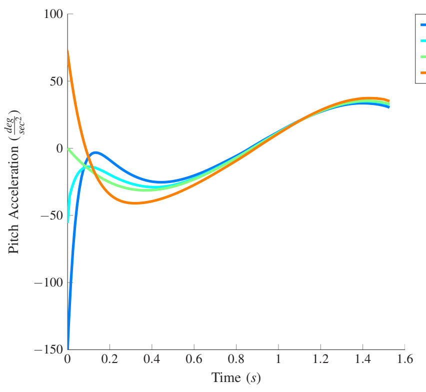


(c) Right Foot X-axis Friction Cone Constraint

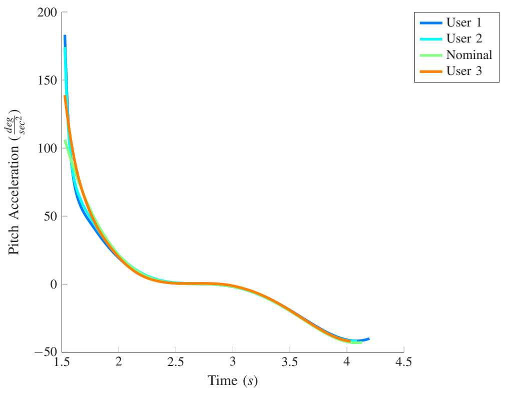


(d) Left Foot X-axis Friction Cone Constraint

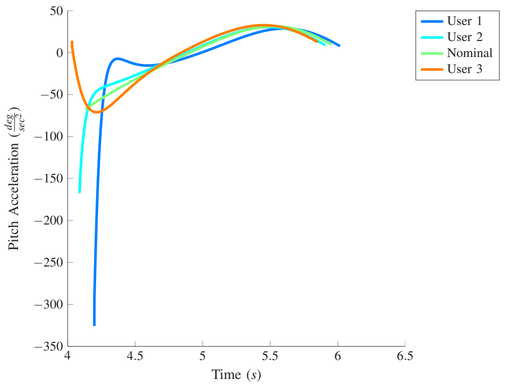
Figure 3.17: **User characteristics for Chair-to-Stand:** The chair and feet friction cone constraints during the *SITTING* domain for different users in the exoskeleton. The characteristics of the users are described in Table 3.1. The results are calculated using $|F_{\bullet}^{\circ}| - \mu \frac{F_{\bullet}^{z}}{\sqrt{2}} \leq 0$, where $F_{\bullet}^{\circ}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair. Therefore, a value above zero violates the friction constraint.



(a) Sitting Pitch Acceleration

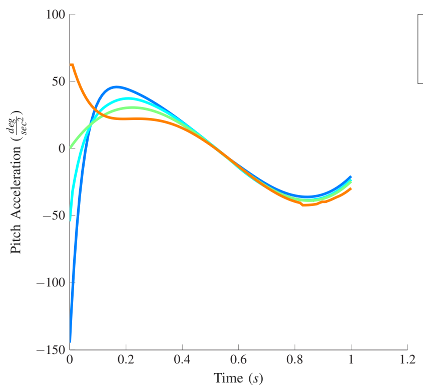


(b) Standing Shift Pitch Acceleration

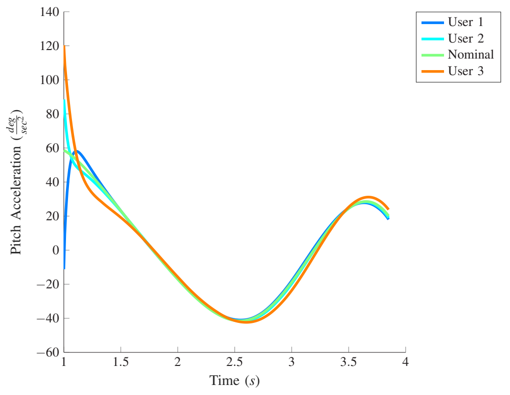


(c) Standing Extend Pitch Acceleration

Figure 3.18: **User characteristics for Chair-to-Crouch-to-Stand:** The torso pitch acceleration during the *SITTING*, *STANDING SHIFT*, and *STANDING EXTEND* domains for different users in the exoskeleton. The characteristics of the users are described in Table 3.1. The jump in acceleration observed between the plots is caused by the transition between domains.

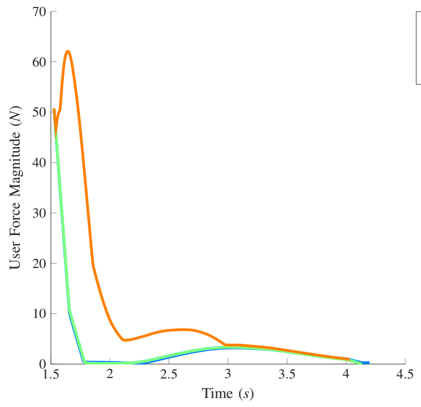


(a) Sitting Pitch Acceleration

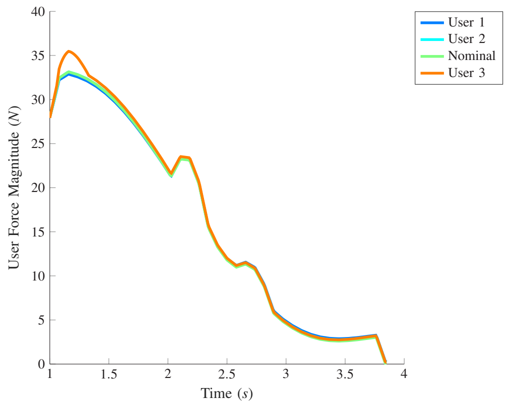


(b) Standing Pitch Acceleration

Figure 3.19: **User characteristics for Chair-to-Stand:** The torso pitch acceleration during the *SITTING* and *STANDING* domains for different users in the exoskeleton. The characteristics of the users are described in Table 3.1. The jump in acceleration observed between the two plots is caused by the transition between the two domains.



(a) Chair-to-Crouch-to-Stand Standing Shift User Force



(b) Chair-to-Stand Standing User Force

Figure 3.20: **User characteristics for Chair-to-Crouch-to-Stand and Chair-to-Stand:** The user force during the *STANDING SHIFT* and *STANDING* domains for different users in the exoskeleton. The characteristics of the users are described in Table 3.1. The depicted user force is the combined force the user would have to apply with both hands.

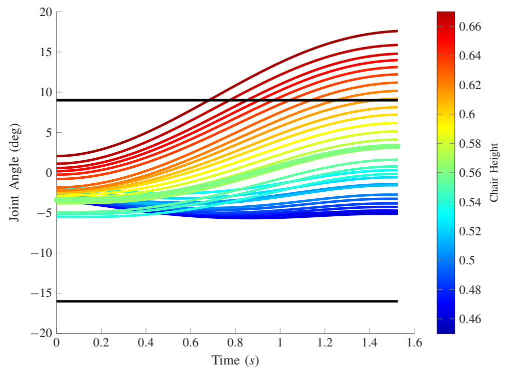
for the torso pitch acceleration from the various chair heights exhibit a mirror like pattern across the x axis with respect to the nominal pitch acceleration values in the sitting domain. The chair-to-stand motion has similar maximum torso pitch acceleration values for its lowest (0.47 m) and highest (0.75 m) chair height. This maximum value occurs in the sitting domain and is 223.8 $\frac{deg}{s^2}$ and -231 $\frac{deg}{s^2}$. The maximum torso pitch acceleration values for chair-to-crouch-to-stand for the lowest and highest chair values occur in the sitting and standing shift domain and are 147.8 $\frac{deg}{s^2}$ and 120 $\frac{deg}{s^2}$. The torso pitch acceleration profiles for the sitting and standing shift domains can be found in Figure 3.23 and Figure 3.24. The torso pitch acceleration values are within the threshold and the tracking and steady state errors are essentially zero for both motions. Therefore, we can conclude that the chair-to-stand motion is better at handling chair height variations than the chair-to-crouch-to-stand motion. More detailed information on the resulting motions can be found in Table 3.3.

3.7.4 Zero User Force

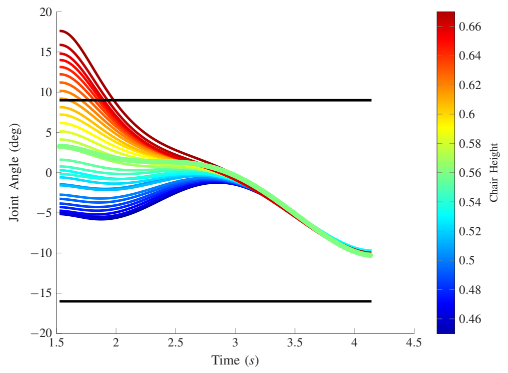
To test the SU and SP controllers' sensitivity to the user force provided, we run a simulation where the user provides no assistance despite the controllers expecting one. The results, which can be found in Table 3.4, show that the controllers are able to achieve both motions with zero user force throughout the entire trajectory. With no state bound and friction constraint violations, minimal tracking error, and absolute maximimum torso pitch acceleration values of 60.3 $\frac{deg}{s^2}$ and 128.4 $\frac{deg}{s^2}$ for chair-to-stand and chair-to-crouch-to-stand respectively, the resulting trajectories for both motions are successful. Therefore, we can conclude that the controllers are not sensitive to the user force and that they are able to compensate for misalignments in the user provided assistance.

Table 3.3: **Various Chair Heights**: A comparison of the closed-loop chair-to-crouch-to-stand and chair-to-stand trajectories obtained from the SU and SP controllers for chair heights ranging from 0.45-0.75m. Recall that the nominal chair height is 0.6m. The user force displayed in the table is the total force the user needs to provide using both arms.For all the values of interest, except for the steady state error of SP, the Domain/Virtual constraint gives the domain where the maximum value occurs. For the steady state error of SP, the virtual constraints where the maximum error occurs are instead given in the Domain/Virtual constraint column. The errors are displayed as maximum position and angle errors. Recall that the first two components of (3.42) are positions and the rest are angles.

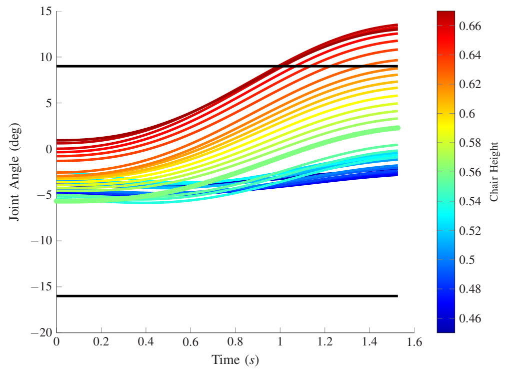
	Chair-to-crouch-to-stand				Chair-to-stand			
Name	Violated	Domain/ Virtual constraint	Max value	Nominal value	Violated	Domain/ Virtual constraint	Max value	Nominal Value
Joint constraint (deg)	0.68 - 0.75m	Sitting Standing shift	8.6 (at 0.75m)	$\times$	0.48-0.45	Sit Standing	4.6 (at 0.45m)	$\times$
Friction constraint (N)	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$
Pitch Acceleration ($\frac{deg}{s^2}$)	$\times$	Sit	147.8 (at 0.45m)	106.1	$\times$	Sit	-230.9 (at 0.75m)	58.7
User force (N)	$\times$	Standing shift	56.5 (at 0.67m)	50.7N	$\times$	Standing	33.4 (at 0.75m)	33.1
Tracking error SU	$\times$	All	~ 0	~ 0	$\times$	All	~ 0	~ 0
Steady state error SP	$\times$	Pos : All	~ 0 m	~ 0 m	$\times$	Pos : All	~ 0 m	~ 0 m
	$\times$	Angle : All	~ 0 rad	~ 0 rad	$\times$	Angle : All	~ 0 rad	~ 0 rad



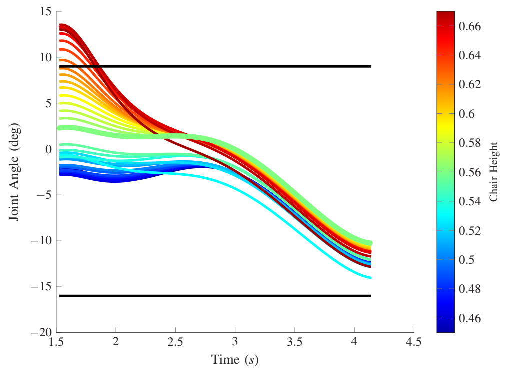
(a) Sitting Left Sag. Ankle



(b) Standing Shift Left Sag. Ankle

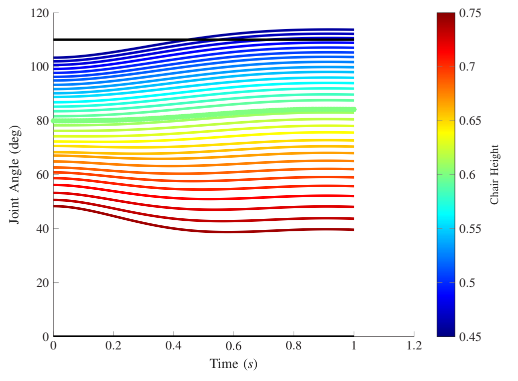


(c) Sitting Right Sag. Ankle

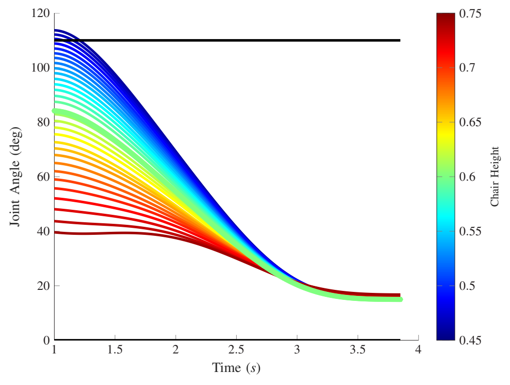


(d) Standing Shift Right Sag. Ankle

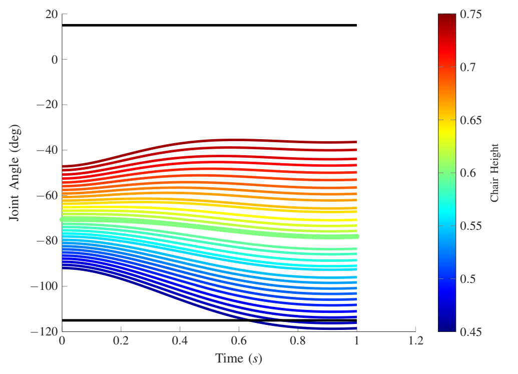
Figure 3.21: **Varying Chair Height for Chair-to-Crouch-to-Stand:** The sagittal ankle joint during the *SITTING* and *STANDING SHIFT* domains for chair heights ranging from 0.45-0.75m. The trajectory resulting from the nominal chair height (0.6m) is depicted by the thick green line, while the black solid lines represent the lower and upper bounds.



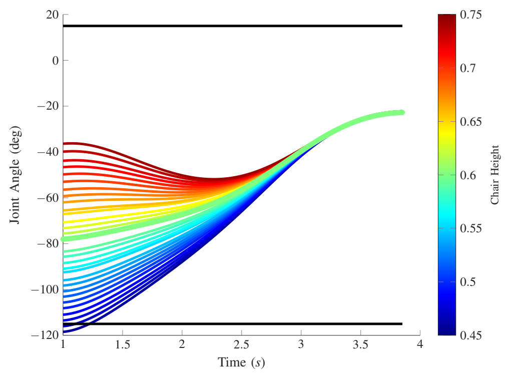
(a) Sitting Left Sag. Knee



(b) Standing Left Sag. Knee

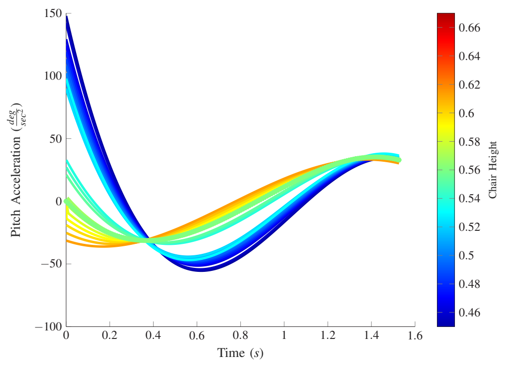


(c) Sitting Left Sag. Hip

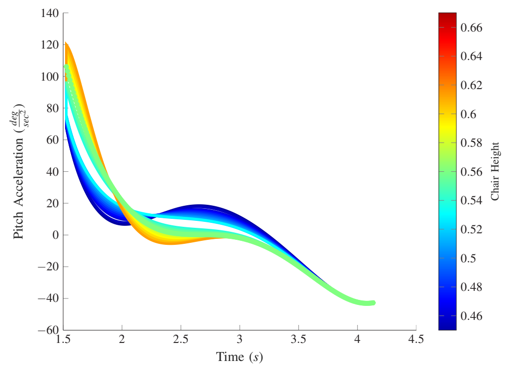


(d) Standing Left Sag. Hip

Figure 3.22: **Varying Chair Height for Chair-to-Stand:** The left sagittal knee and hip joints during *SITTING* and *STANDING* domains for chair heights ranging from 0.45-0.75m. The trajectory resulting from the nominal chair height (0.6m) is depicted by the thick green, while the black solid lines represent the lower and upper bounds.



(a) Sitting Pitch Acceleration



(b) Standing Shift Pitch Acceleration

Figure 3.23: **Various Chair Height for Chair-to-Crouch-to-Stand:** The torso pitch acceleration during the *SITTING* and *STANDING SHIFT* domains for chair heights ranging from 0.45-0.75m. The trajectory resulting from the nominal chair height (0.6m) is depicted by the thick green line. The jump in acceleration observed between the two plots is caused by the transition between the sit and SS domains.

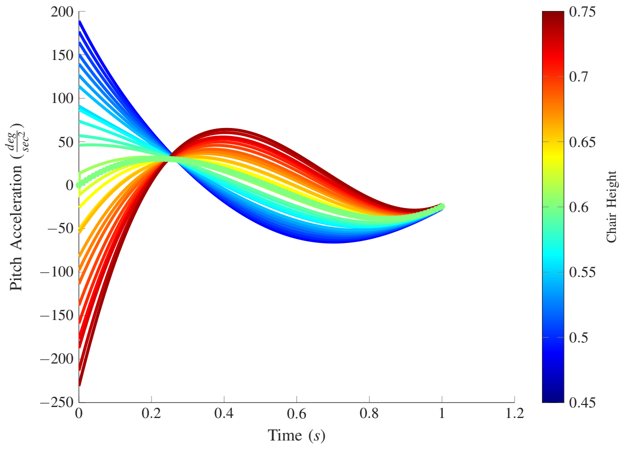


Figure 3.24: **Various Chair Height for Chair-to-Stand:** The torso pitch acceleration during the *SITTING* domain for chair heights ranging from 0.45-0.75m. The trajectory resulting from the nominal chair height (0.6m) is depicted by the thick green line.

Table 3.4: **Zero User Force:** A comparison of the closed-loop chair-to-crouch-to-stand and chair-to-stand trajectories obtained from the SU and SP controllers when the user provided no assistance.For all the values of interest, except for the steady state error of SP, the Domain/Virtual constraint gives the domain where the maximum value occurs. For the steady state error of SP, the virtual constraints where the maximum error occurs are instead given in the Domain/Virtual constraint column. The errors are displayed as maximum position and angle errors. Recall that the first three components of (3.42) are positions and the rest are angles.

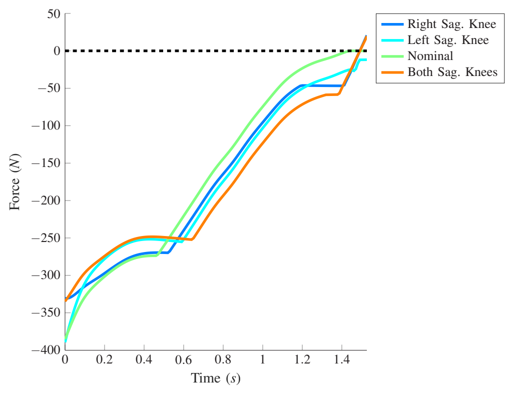
	Chair-to-crouch-to-stand				Chair-to-stand			
Name	Violated	Domain/ Virtual constraint	Max value	Nominal value	Violated	Domain/ Virtual constraint	Max value	Nominal value
Joint constraint (deg)	×	×	×	×	×	×	×	×
Friction constraint (N)	×	×	×	×	×	×	×	×
Pitch acceleration $(\frac{deg}{s^2})$	×	Standing shift	128.4	106.1	×	Stand	60.3	33.1
Tracking error SU	×	Standing extend	0.002	~ 0	×	Stand	0.004	~ 0
Steady state error SP	×	Pos : All	~ 0 m	~ 0 m	×	Pos : All	~ 0 m	~ 0 m
	×	Angle : All	~ 0 rad	~ 0 rad	×	Angle :All	~ 0 rad	~ 0 rad

3.7.5 User Spasticity

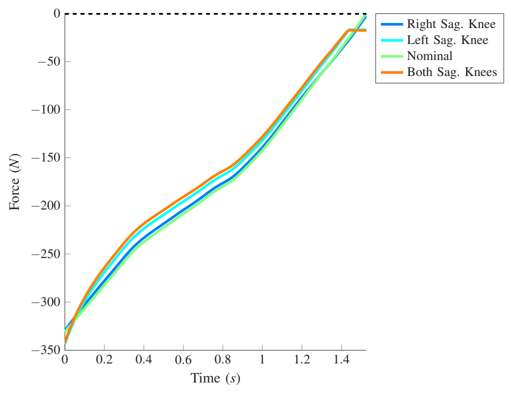
We apply a 15 Nm flexion on individual knee angles and in pairs to simulate spasticity. The value of 15 Nm is chosen based on the study by Franzoi et al [128] where the maximum torque flexion observed for SCI at 60 $\frac{deg}{s}$ was 8 NM. Since the maximum knee velocity for both motions does not exceed 60 $\frac{deg}{s}$ (31.5 $\frac{deg}{s}$ and 30.7 $\frac{deg}{s}$ for the nominal simulation results during chair-to-crouch-to-stand and chair-to-stand respectively), 15 Nm is more than sufficient for testing. For both motions, there are no joint angle and ZMP constraint violations, and the tracking and steady state errors are small. Additionally, the user force for both motions is near the nominal value. See Table 3.5 for more details.

Both motions, however, do exhibit friction cone constraint violations. When the 15 Nm is applied to the right knee and both knees the chair-to-crouch-to-stand motion exhibits chair slipping along the x axis in the sitting domain. This slipping, however, as was discussed before, can be alleviated by using a chair with a higher friction coefficient. The maximum value of the friction constraint violation is 20.7 N, and 19.7 N when the spasticity occurs in the right and both knees respectively. No friction bound constraint violations are observed when spasticity occurs in the left knee. For chair-to-stand the left foot slips for all the spasticity tests. The highest friction constraint violation for the left foot slipping are 0.16 N, 55.3 N, and 59.8 N for spasticity on the right knee, left knee, and both knees respectively. Therefore, since the constraint violation is quite large for the left and both knees we consider the chair-to-stand motion to have failed the spasticity test for both of these situations. The friction cone constraint can be found in Figure 3.18 and Figure 3.19.

We continue our analysis looking at just the chair-to-stand results from spasticity on the right knee and chair-to-crouch-to-stand on the right, left, and both knees. The



(a) Chair X-axis Friction Cone Constraint



(b) Chair Y-axis Friction Cone Constraint

Figure 3.25: **Spasticity for Chair-to-Crouch-to-Stand:** The chair friction cone constraints during the *SITTING* domain for spasticity induced in the left, right, and both sagittal knees. The results are calculated using $|F^{\circ}_{\bullet}| - \mu\frac{F^{z}_{\bullet}}{\sqrt{2}} \leq 0$, where $F^{\circ}_{\bullet}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair. Therefore, a value above zero violates the friction constraint.

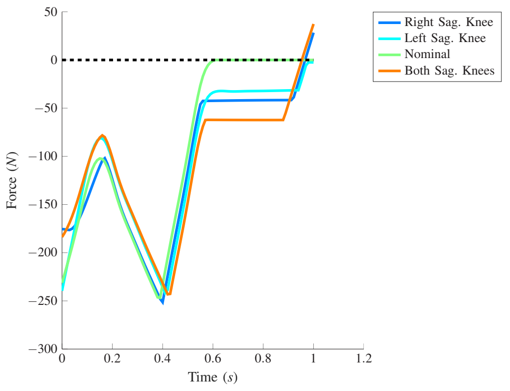
maximum torso pitch acceleration observed for both motions, $-242.72\ \frac{deg}{s^2}$, occurs during the standing extend domain of chair-to-crouch-to-stand for spasticity on both knees and is well below our threshold for user comfort. We can now conclude that the chair-to-crouch-to-stand motion deals with spasticity better than chair-to-stand can.

3.7.6 Asymmetric Motor Torque

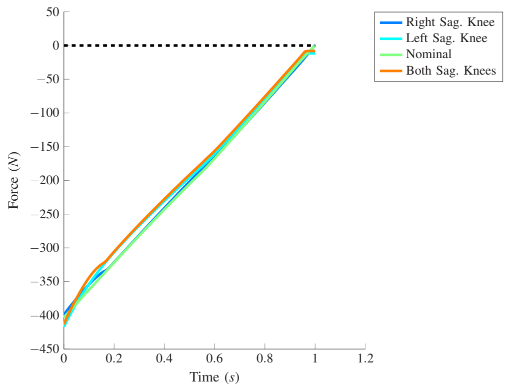
To mimic asymmetry in motor torque, a 15% motor deficiency is introduced in the right sagittal knee, right sagittal knee and hip, and all the right motors. For both motions, there is no joint angle constraint violation. The maximum torso pitch acceleration for both motions, 139.7.4 $\frac{deg}{s^2}$ for chair-to-crouch-to-stand and 64.2 $\frac{deg}{s^2}$ for chair-to-stand, occur with the asymmetry in all the right motors. Since these values are below our threshold, and the resulting user force for both motions is close to the nominal value, both motions should be comfortable for the user. The tracking and steady state error for both motions are small.

Table 3.5: **User Spasticity:**A comparison of the closed-loop chair-to-crouch-to-stand and chair-to-stand trajectories obtained from the SU and SP controllers when spasticity is induced in the right knee, left knee, and both knees. The user force displayed in the table is the total force the user needs to provide using both arms. For all the values of interest, except for the steady state error of SP, the Domain/Virtual constraint gives the domain where the maximum value occurs. For the steady state error of SP, the virtual constraints where the maximum error occurs are instead given in the Domain/Virtual constraint column. The errors are displayed as maximum position and angle errors. Recall that the first three components of (3.42) are positions and the rest are angles.

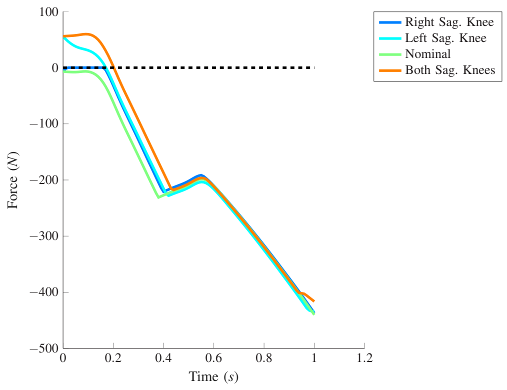
	Chair-to-crouch-to-stand				Chair-to-stand			
Name	Violated	Domain/ Virt Constr	Max Value	Nominal Value	Violated	Domain/ Virt Constr	Max Value	Nominal Value
Joint constraint (deg)	×	×	×	×	×	×	×	×
Friction constraint (N)	Chair	Sitting	Chair x : 20.7 (for right knee) No violation (for left knee) Chair x : 19.7 (for both knees)	×	Feet Chair	Sitting	Chair x : 28.7 left foot x : 0.16 (for right knee) left foot x : 55.76 right Foot x : 0.34 (for left knee) Left foot : 59.8 Chair : 37.4 (for both knees)	×
Pitch acceleration ($\frac{deg}{s^2}$)	×	Standing extend	-204.1 (for right knee) 117.9 (for left knee) -242.7 (for both knees)	106.1	×	Standing Stopping	Stand : 202 (right knee) Stand : 53.7 (for left knee) Stopping : 145 (for both knees)	58.7
User force (N)	×	Standing shift	50.8 (for All)	50.7N	×	Standing	33.2 (for All)	33.1
Tracking error SU	×	Standing extend	0.02 (for left knee)	~ 0	×	Standing	0.01 (for right knee)	~ 0
Steady state error SP	×	Pos : $y_d(1)$	0.001 m (for left knee)	~ 0 m	×	Pos : $y_d(1)$	0.001 m (for left knee)	~ 0 m
	×	Angle : $y_d(6)$	0.06 rad (for both knees)	~ 0 rad	×	Angle : $y_d(6)$	0.05 rad (for both knees)	~ 0 rad



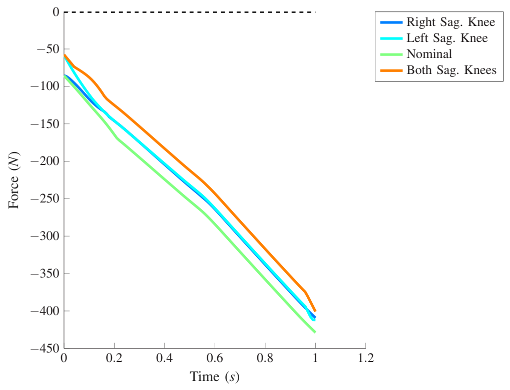
(a) Chair X-axis Friction Cone Constraint



(b) Sit Chair Y-axis Friction Cone Constraint



(c) Left Foot X-axis Friction Cone Constraint



(d) Left Foot Y-axis Friction Cone Constraint

Figure 3.26: **Spasticity for Chair-to-Stand:** The chair and left foot friction cone constraints during the *SITTING* domain for spasticity induced in the left, right, and both knees. The results are calculated using $|F_{\bullet}^{\circ}| - \mu \frac{F_{\bullet}^{z}}{\sqrt{2}} \leq 0$, where $F_{\bullet}^{\circ}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair. Therefore, a value above zero violates the friction constraint.

Slipping is observed in the right foot and the chair along the x axis when the torque reduction is applied on the right knee motor for chair-to-stand. The right foot friction constraint violation of 14 *N* is significant enough for us to consider the chair-to-stand motion to have failed for motor asymmetry in the right knee. The chair-to-stand motion also violates the friction cone constraint of the chair when asymmetry occurs in all motors. However, the chair friction constraint violation is small. When the asymmetry occurs both in the right sagittal knee and hip, the chair-to-stand motion does not violate the friction constraint. In comparison, the chair-to-crouch-to-stand motion exhibits no friction constraint violations when there's torque reduction in the right sagittal knee, and right sagittal knee and hip. Chair-to-crouch-to-stand, however, does experience slipping in the chair along the x and y axis when the torque reduction is present in all right motors. The maximum value of the chair friction cone constraint violation is 42.2 *N*. Even though the violation of the friction cone constraint of the chair can be eliminated by using a chair with higher friction coefficient and by the user exerting the necessary force, a violation of 42.2 *N* is significant enough for us to consider the chair-to-crouch-to-stand motion to have failed the robustness test when the motor deficiency occurs on all right motors.

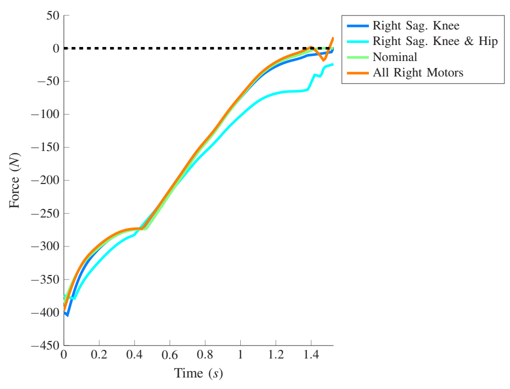
Due to the fact that it's highly unlikely to have asymmetry occur in all motors simultaneously, we can conclude, if we remove the results for asymmetry in all motors, that the chair-to-crouch-to-stand motion is better at handling torque asymmetry. For more details, see Table 3.6, Figure 3.27, and Figure 3.28.

3.8 Comparing Our Controller's Performance to the Literature

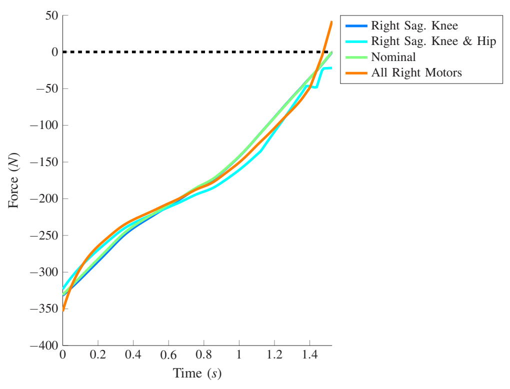
In this section, we compare the SU controller of Section 3.6 to related work in the literature. We seek to highlight the utility of the design philosophy in Section

Table 3.6: A comparison of the closed-loop chair-to-crouch-to-stand and chair-to-stand trajectories obtained from the SU and SP controllers for torque asymmetry in the sagittal (sag.) knee motor, sagittal knee and hip motors, and all the motors.Even though asymmetry in all motors is unlikely, it is an interesting case to study. The user force displayed in the table is the total force the user needs to provide using both arms. For all the values of interest, except for the steady state error of SP, the Domain/Virtual constraint gives the domain where the maximum value occurs. For the steady state error of SP, the virtual constraints where the maximum error occurs are instead given in the Domain/Virtual constraint column. The errors are displayed as maximum position and angle errors. Recall that the first three components of (3.42) are positions and the rest are angles.

	Chair-to-crouch-to-stand				Chair-to-stand			
Name	Violated	Domain/ Virtual constraint	Max value	Nominal value	Violated	Domain/ Virtual constraint	Max value	Nominal value
Joint constraint (deg)	×	×	×	×	×	×	×	×
Friction constraint (N)	Chair	Sitting	No violation (for sag. knee) No violation (for sag. knee & hip) Chair x:16.7 Chair y:42.2 (for all sag. motors)	×	Right foot Chair	Sitting	Right foot x : 14.5 Chair x : 3.47 Chair y : 0.15 (for sag. knee) No violation (for sag. knee & Hip) Chair : 6.5 (for all sag. motors)	×
Pitch acceleration ($\frac{deg}{s^2}$)	×	Standing shift	76.5 (for sag. Knee) -103.34 (for sag. knee & hip) 139.7 (for all sag. motors)	106.1	×	Standing	53.68 (for sag. knee) 54.55 (for sag. knee & hip) 64.2 (for all sag. motors)	58.7
User force (N)	×	Standing shift	50.7 (for all tests)	50.7N	×	Standing	33.4 (for all sag. motors)	33.1
Tracking error SU	×	Standing extend	0.02 (for all sag. motors)	~ 0	×	Sitting	0.01 (for sag. knee & Hip)	~ 0
Steady state error SP	×	Pos :All	~ 0 m	~ 0 m	×	Pos : All	~ 0 m	~ 0 m
	×	Angle : $y_d(6)$	0.02 rad (for sag. knee & hip)	~ 0 rad	×	Angle : $y_d(6)$	0.01 rad (for sag. knee & hip)	~ 0 rad



(a) Chair X-axis Friction Cone Constraint



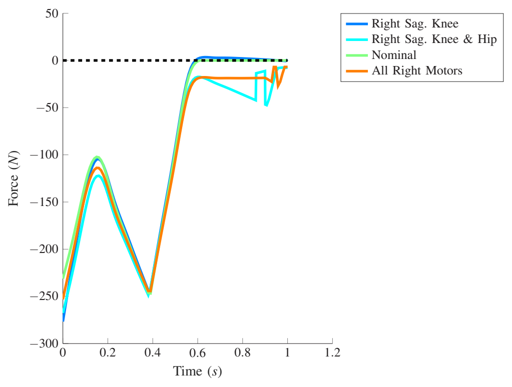
(b) Chair Y-axis Friction Cone Constraint

Figure 3.27: **Asymmetric Torque for Chair-to-Crouch-to-Stand:** The chair friction cone constraints during the *SITTING* domain for torque asymmetry in the sagittal knee motor, sagittal knee and hip motors, and all the motors. The results are calculated using $|F^{\circ}_{\bullet}| - \mu \frac{F^{z}_{\bullet}}{\sqrt{2}} \leq 0$, where $F^{\circ}_{\bullet}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair. Therefore, a value above zero violates the friction constraint.

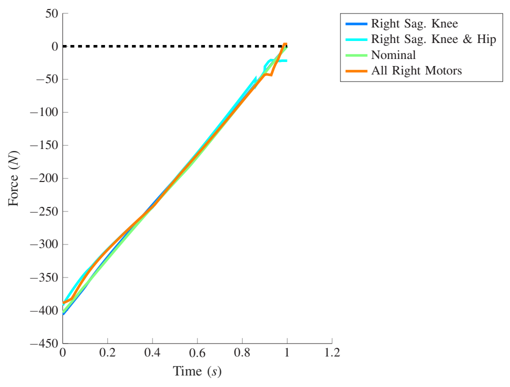
3.4.3, where control objectives were selected to have minimal conflict with the contact constraints of the exo-system, while ensuring that the actuators of the exoskeleton are effectively[6] used to implement the constraints. The comparison will be done with respect to perturbations in the chair height and the user mass-inertia parameters. We selected these two perturbations because they are likely to be encountered in practice. The nominal values used here are the same as those used in Section 3.7.

The controller objectives in the literature consist of a combination of the exo-system's CoM and the (relative) joint angles of the exoskeleton. Moreover, they are applied to planar models. To account for the full 3D setting of our work, we selected

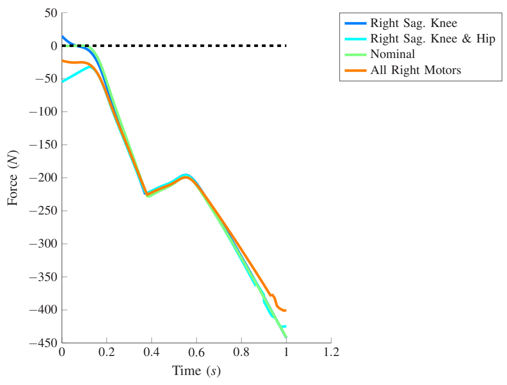
[6]We refer to the condition number constraint on our design process.



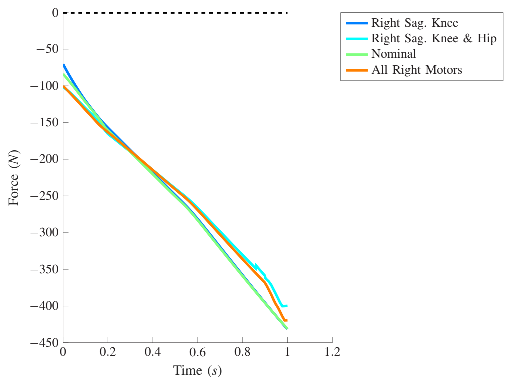
(a) Chair X-axis Friction Cone Constraint



(b) Chair Y-axis Friction Cone Constraint



(c) Left Foot X-axis Friction Cone Constraint



(d) Left Foot Y-axis Friction Cone Constraint

Figure 3.28: **Asymmetric Torque for Chair-to-Stand:** The chair and left foot friction cone constraints during the *SITTING* domain for torque asymmetry in the sagittal knee motor, sagittal knee and hip motors, and all the motors. The results are calculated using $|F^{\circ}_{\bullet}| - \mu \frac{F^{z}_{\bullet}}{\sqrt{2}} \leq 0$, where $F^{\circ}_{\bullet}$ denotes the components of the contact forces along the x or y axis that are associated with the feet or chair. Therefore, a value above zero violates the friction constraint.

two sets of control objectives from [20, 23, 25–27, 29, 32, 104, 123], as follows,

$$y_{\text{lit}_1} = \begin{bmatrix} CoM_X \\ CoM_Y \\ \psi \\ CoM_Z \\ \frac{q_{\text{knee}}^{\text{left}} - q_{\text{knee}}^{\text{right}}}{2} \\ \frac{q_{\text{sag.ankle}}^{\text{left}} + q_{\text{sag.ankle}}^{\text{right}}}{2} \end{bmatrix} \quad y_{\text{lit}_2} = \begin{bmatrix} \theta \\ \varphi \\ \psi \\ \frac{q_{\text{knee}}^{\text{left}} + q_{\text{knee}}^{\text{right}}}{2} \\ \frac{q_{\text{knee}}^{\text{left}} - q_{\text{knee}}^{\text{right}}}{2} \\ \frac{q_{\text{sag.ankle}}^{\text{left}} + q_{\text{sag.ankle}}^{\text{right}}}{2} \end{bmatrix} \tag{3.44}$$

Here θ, φ, and ψ are the torso pitch, roll, and yaw angle respectively.

To implement control objectives in (3.44) we design a QP nput-output linearizing controller based on [20, 29]. The QP minimizes the error between the nominal and applied motor torques, and the nominal and applied user force, similar to our SU controller. The QP also implements the dynamic constraints and torque bounds. We will refer to the two control objectives and the QP-based controller from the literature as the baseline control objectives and the baseline controller, respectively.

For both sit-to-stand motions, our control objectives are better at respecting the contact constraints and they result in lower torso pitch acceleration. We conjecture that the latter is because our control objectives are more effective at deploying the motor torques due to the bound on the condition number. The lower torso pitch acceleration augments user comfort. For brevity, we only present the data for chair-to-stand that led us to these conclusions. All control objectives function correctly under nominal conditions. At a chair height of 0.62 m, that is 0.02 m above the nominal value of 0.60 m, y_{lit_1} results in a force violation at the feet of 134 N, y_{lit_2} results in a violation of 174 N, while our constraints (under the same baseline QP-IO controller as used for $y_{\text{lit}-1}$ and $y_{\text{lit}-2}$), results in a violation[7] of 8 N. When the user model is

[7]To be clear, our objectives experience no violations when we use the QP formulated in Section 3.6.

perturbed, the baseline control objectives are unable to achieve the sit-to-stand motion with any of the users due to large friction constraint violations. Our control objectives, however, are able to successfully achieve the sit-to-stand motion with User 3, with a maximum torso pitch acceleration of only 69.8 $\frac{deg}{s^2}$, which is less than 20% of the allowed upper bound.

We conclude that the baseline QP-IO controller and control objectives are neither robust to variations in chair height nor to users lighter than the nominal user. Therefore, in comparison to the baseline controller, our controller, SU, is able to reject more perturbations.

3.9 Conclusion

We have designed and analyzed two motions, chair-to-stand and chair-to-crouch-to-stand, for the exoskeleton Atalante, that successfully performed fully assisted sit-to-stand motions even in the presence of perturbations. Constrained optimization, performed using FROST, was utilized to ensure that the open-loop behavior of the two motions were feasible. Feasibility was defined in terms of user comfort, dynamic feasibility, and hardware limitations. We derived the dynamic equations using the full dynamic model of the exoskeleton, and incorporated the user force in the equations of motion. The equations of motion for both the, chair-to-stand and chair-to-crouch-to-stand motions were highly constrained, due to the various contact points, and therefore underdetermined with respect to the motor torques. To address this, we developed, for a computed-torque controller, a novel way of systematically designing virtual constraints so that they “do not fight” the contact constraints for highly constrained systems. Along the way, we also introduced a formulation that allowed the control actions to “fight” against a specific contact constraint to induce a domain transition while respecting other desired contact constraints.

To analyze and compare the closed-loop behavior of the two motions, we designed two QP-based computed-torque controllers and conducted physically motivated robustness tests. The choice of a QP-based controller allowed to select the vector of motor torques of smallest norm that satisfied the control objectives, as expressed by a set of virtual constraints. Our results indicated that both motions can equally handle variations to user characteristics and user force disparities. In fact, our analysis showed that it is possible to successfully stand up with no user force under both motions. The chair-to-crouch-to-stand motion, however, was more well equipped to handle asymmetric perturbations, while the chair-to-stand excelled at handling variations to the chair height. To improve the operational range of either motion, for perturbations that result in incorrect contact forces, a chair with a high friction coefficient could be specified or the motion could be redesigned for the new environmental conditions.

To check the effectiveness of our method, we compared our control objectives and sit-to-stand controller to those found in the literature. From our analysis, we found that our control objectives were better at respecting contact constraints and resulted in motions that required less torso pitch acceleration. Additionally, our sit-to-stand controller was better equipped at handling perturbations.

Even though the methods presented in this chapter are illustrated using the sit-to-stand motion and for Atalante, our methodology can easily be adopted for other motions with multiple contact points and other exoskeletons or humanoids.

Chapter 4

Accounting for Contact Dynamics in Robust Comfortable Sit-to-Stand Motions of 3D Lower-limb Exoskeletons

4.1 Introduction

In the previous chapter, we focused on designing and analyzing two motions, namely chair-to-stand and chair-to-crouch-to-stand, for the exoskeleton Atalante. These motions were successfully able to perform fully assisted sit-to-stand movements. Due to the highly constrained nature of the equations of motion, we introduced a method to systematically design virtual constraints for highly constrained systems. To achieve the sit-to-stand motions and safely come to a stop in a standing position, we designed two quadratic program-based computed-torque controllers. Lastly, we analyzed the closed-loop behaviors of the two sit-to-stand motions under the two controllers using physically motivated robustness tests. The analysis, however, was done on closed-loop motions derived in the FROST simulator, which does not consider contact dynamics and, therefore, does not simulate real-world conditions. Recall that FROST is the open-source package utilized to generate the trajectories of the two sit-to-stand motions.

Therefore, to emulate the real-world conditions the exoskeleton would operate in, we continue the analysis of the closed-loop sit-to-stand motions with two simulators MuJoCo [129] [130] and Jiminy [131] that do take contact dynamics into account. This chapter will go over the preliminary closed-loop chair-to-stand motions obtained from the two simulators. Unfortunately, due to the Atalante robot breaking, we were unable to conduct any hardware tests. The work presented here is a collaboration with Prof. Ayonga Hereid and Ph.D. candidate Victor Paredes at The Ohio State University.

4.2 Problem Description

In line with Chapter 3, we use the floating base equations of motion described by (3.1) and (3.2), the generalized coordinates listed in (3.3) and the same definition for the assistive force from the user. Recall that the user force is defined to act at the top of the torso in the torso's body coordinate frame, and it is assumed that there is no net moment generated by the user. The chair-to-stand motion, as described in Section 3.2.2, consists of two domains, the sitting domain and the standing domain. For this motion, the center of mass is simultaneously shifted forward and extended.

In contrast to Chapter 3, where the exoskeleton user weighed 73kg with a height of 1.73m, this chapter uses a mannequin with a weight of 61.5kg and only an upper body. Thanks to the change in users, the nominal chair-to-stand trajectory derived in the previous chapter may no longer be optimal. Therefore, the controller needs to account for the user discrepancy and the ground contact dynamics while tracking the nominal chair-to-stand trajectory. The total weight of the exoskeleton and the mannequin combined is 135.92kg. The chair height is set to 0.6m, and the friction coefficient for both the chair and feet is set to 0.9. Additionally, the torsional friction for the feet is set to 900.

4.3 Controller

Due to the underdetermined nature of the equations of motion for the sit-to-stand motion, as detailed in Chapter 3, we design a quadratic program-based computed-torque controller. Like the SU controller described in Section 3.6.1, the objective of the controller here is, given a user force for each time instance, t, to select motor torques of minimal norm such that disturbances to the nominal motion are attenuated.

To avoid inverting the inertial matrix and, thus, numerical instabilities, we reformulate the equations of motion as follows:

$$\underbrace{\begin{bmatrix} D & -B & -J^{\top} \\ J & 0 & 0 \end{bmatrix}}_{\tilde{A}_{eq}} \begin{bmatrix} \ddot{q} \\ u \\ \Gamma \end{bmatrix} = \underbrace{\begin{bmatrix} -C - G + J_{ext}^{\top}\zeta \\ -\dot{J}\dot{q} \end{bmatrix}}_{\tilde{b}_{eq}} \tag{4.1}$$

where D, B, C, and G are the inertia, torque distribution, Coriolis, and gravity matrices/vectors, respectively. J and J_{ext} are jacobians that map the contact wrenches, Γ, and the user force, ζ, to the generalized coordinates, q, respectively. u is the motor torques. Given this formulation, the optimization variables for the controller are the acceleration of the generalized coordinates, the motor torques, and the contact wrenches. Note that the user force is given as an external input, using the force profile derived in Chapter 3 and depicted in Figure 3.6, as we do not want the controller to rely on the user making corrections to the trajectory. As discussed in Chapter 3, we assume that the user will learn to provide the external force needed at the "right" time.

To account for the infinite number of solutions that are characteristic of under-determined equations, the controller is set to track the nominal motor torques and contact wrenches derived from the optimization in Chapter 3. Lastly, similar to the SU controller, the equations of motions, (4.1), ZMP, torque bounds, and friction constraints are included as constraints to ensure feasibility. The controller is expressed as a QP in the form:

$$
\begin{bmatrix} \ddot{q}(t) \\ u(t) \\ \Gamma(t) \end{bmatrix} := \underset{\ddot{q}\in\mathbb{R}^{18}, u\in\mathbb{R}^{12}, \Gamma\in\mathbb{R}^{15} \text{ or } 12}{\arg\min} \quad \alpha_{\ddot{q}}||\ddot{q}||_2^2 + \alpha_u||u - u^*(t)||_2^2 + \alpha_\Gamma||\Gamma - \Gamma^*(t)||_2^2
$$

$$
\begin{aligned}
&\text{subject to} \\
&\tilde{A}_{eq}(q(t)) \begin{bmatrix} u \\ \zeta \end{bmatrix} = \tilde{b}_{eq}(q(t), \dot{q}(t)) \\
\{P_*^x, P_*^y\} &\subseteq SP \\
F_\bullet^z &\geq 0 \\
|F_\bullet^x| &\leq \mu \frac{F_\bullet^z}{\sqrt{2}} \\
|F_\bullet^y| &\leq \mu \frac{F_\bullet^z}{\sqrt{2}} \\
|M_\circ^z| &\leq \gamma F_\circ^z \\
u_{lb} \leq u &\leq u_{ub}
\end{aligned} \tag{4.2}
$$

where $F_\bullet^x$, $F_\bullet^y$ and $F_\bullet^z$ denote the components of the contact forces associated with the chair or the feet, $\mu_\bullet$ is an assumed friction coefficient for the chair or feet, γ is a torsional friction coefficient, and $M_\bullet^z$ and $F_\bullet^z$ are the moments about and forces along the z-axis for the feet. P_*^x and P_*^y are the x and y components of the ZMP, while SP is the support polygon defined as described Section 3.2.3. $\alpha_{\ddot{q}}$, α_u and α_Γ are the weights for the optimization variables. Recall that the size of the contact wrenches will be 15 and 12 for the sitting and standing domains, respectively.

4.4 Results

This section assesses the preliminary resulting closed-loop behaviors of the chair-to-stand motion in the MuJoCo and Jiminy simulator. To account for any perturbations that might occur at the start of the simulation as a function of the simulators, such as the robot dropping into position, a PD controller is used for the first four seconds to stabilize the exoskeleton.

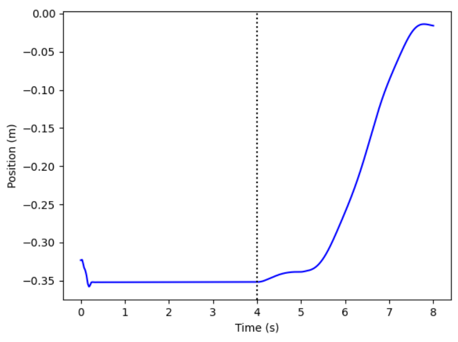
4.4.1 Jiminy

It was challenging to transition from the sitting to the standing domain in Jiminy with a non-zero ankle torque due to how the ground contact is modeled. For instance, when the PD controller applied a constant 90Nm torque to the sagittal ankle during the sitting domain, the feet pitched. To prevent the feet from rolling or pitching, we add 80kg to the mass of the sagittal and henke ankles. With this assistance, the exo+mannequin system successfully completes the chair-to-stand motion as depicted in Figure 4.1.

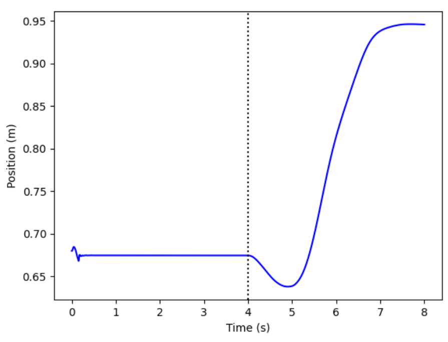
The resulting closed-loop motion respects the joint and torque bounds and closely tracks the nominal torque trajectories. Figures 4.2 and 4.3 show the resulting joint and torque trajectories, respectively. Additionally, the resulting ZMP closely follows the nominal ZMP as depicted in Figure 4.5. The tracking error has a maximum value of about 0.7, as depicted in Figure 4.4. However, considering the closed-loop motion's tracking of the nominal torque and ZMP, the discrepancy in the users, and the fact that the virtual constraints are dimensionless, we can conclude that a value of 0.7 is acceptable.

4.4.2 MuJoCo

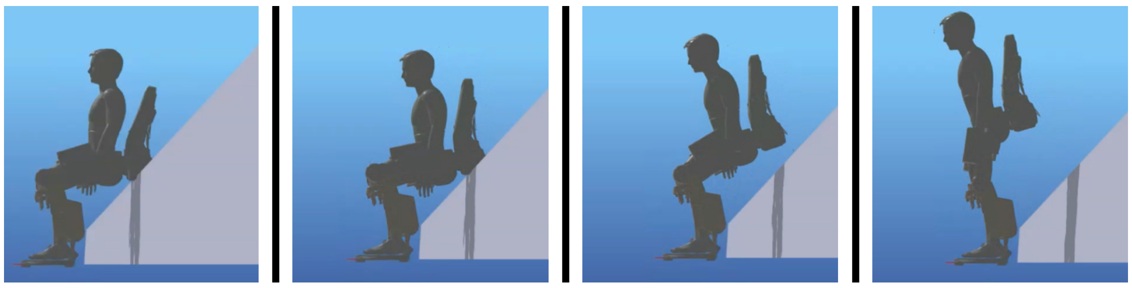
The additional ankle mass we had to apply in Jiminy led us to switch to MuJoCo. The preliminary closed-loop motion in MuJoCo successfully stands up with no extra ankle weight, as shown in Figure 4.6. Similar to the motion in Jiminy, this motion closely follows the nominal torque trajectory and respects joint and torque bounds. However, in the standing domain that begins at 5 seconds, the sagittal ankle is at its bounds, and the torque profiles oscillate. Gaining tuning can potentially mitigate these



(a) Base X

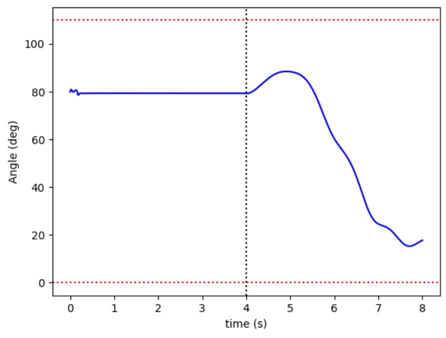


(b) Base Z

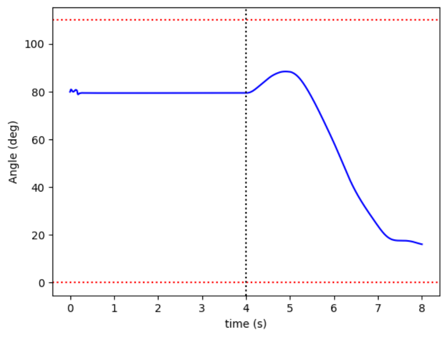


(c) Snapshots of the exo+mannequin system standing

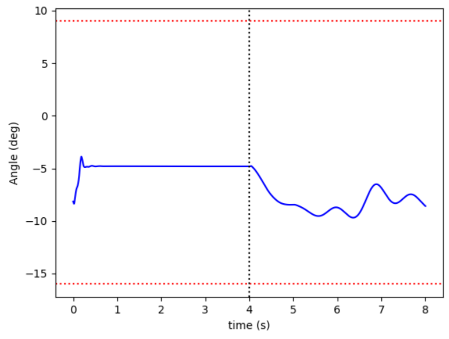
Figure 4.1: **Jiminy:** The x and z trajectories of the base and the snapshots of the exo+mannequin system show the system standing up. The y trajectory is not included as the motion mainly occurs in the sagittal plane. Note that the snapshots depict legs, but the mannequin has no legs.



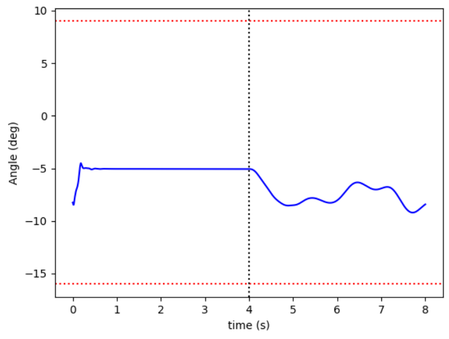
(a) Left Sagittal Knee



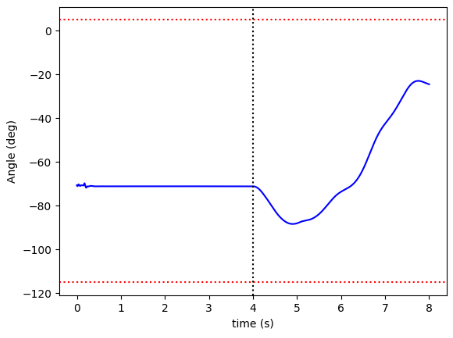
(b) Right Sagittal Knee



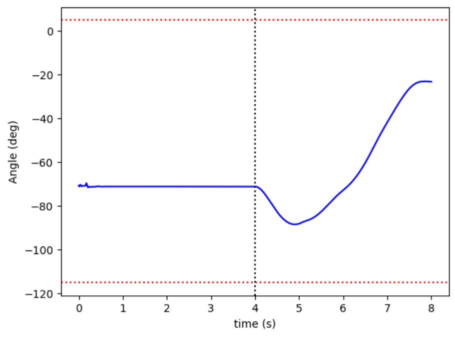
(c) Left Sagittal Ankle



(d) Right Sagittal Ankle

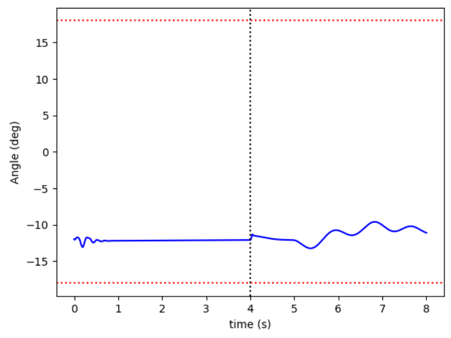


(e) Left Sagittal Hip

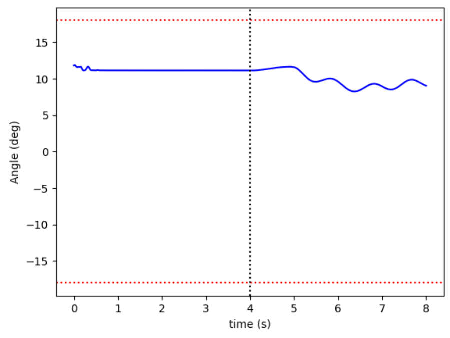


(f) Right Sagittal Hip

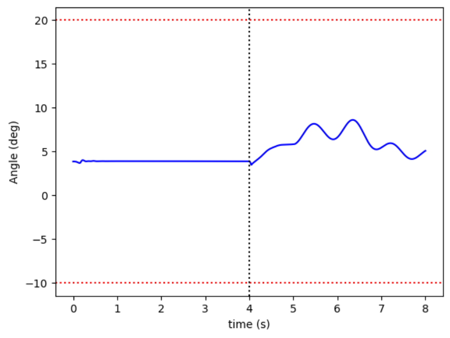
Figure 4.2: **Jiminy**: The chair-to-stand closed-loop trajectories show that the joints respect the joint constraints. Recall that a PD controller is used for the first 4 seconds. The red dashed lines depict the bounds while the black dashed line indicates the end of the PD controller.



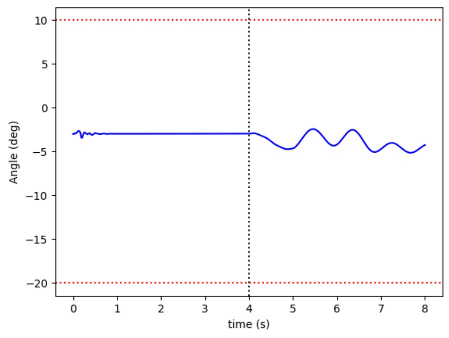
(g) Left Henke Ankle



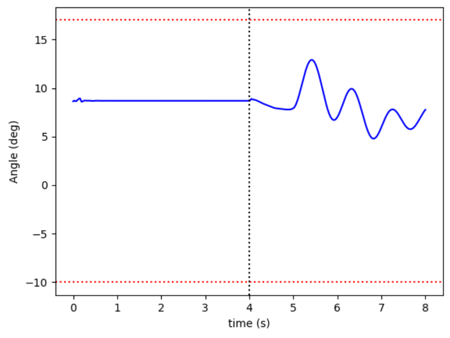
(h) Right Henke Ankle



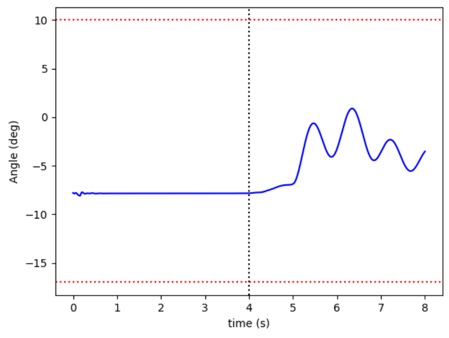
(i) Left Transverse Hip



(j) Right Transverse Hip

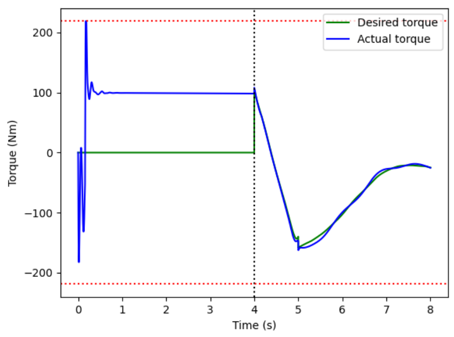


(k) Left Frontal Hip

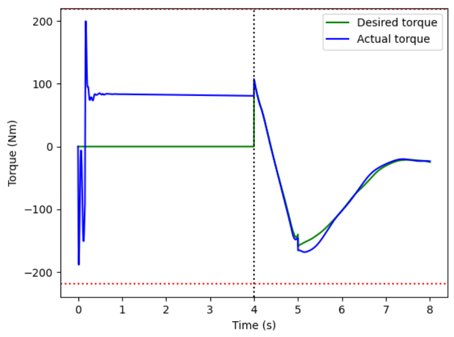


(l) Right Frontal Hip

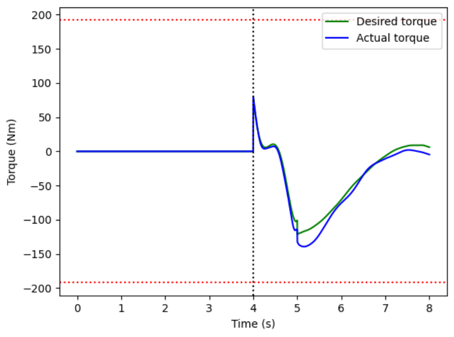
Figure 4.2: **Jiminy**: The chair-to-stand closed-loop joint trajectories(Cont.)



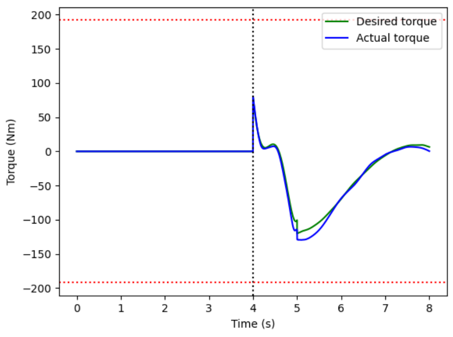
(a) Left Sagittal Knee



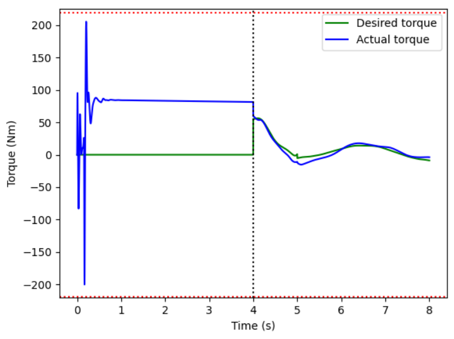
(b) Right Sagittal Knee



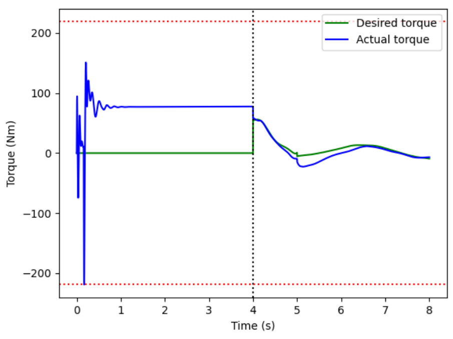
(c) Left Sagittal Ankle



(d) Right Sagittal Ankle

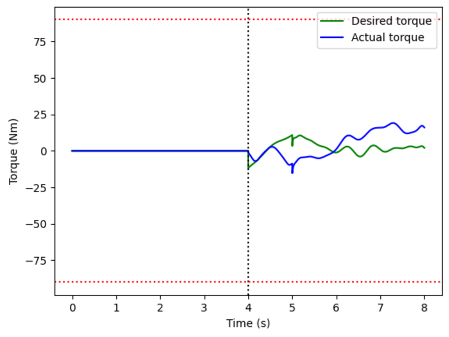


(e) Left Sagittal Hip

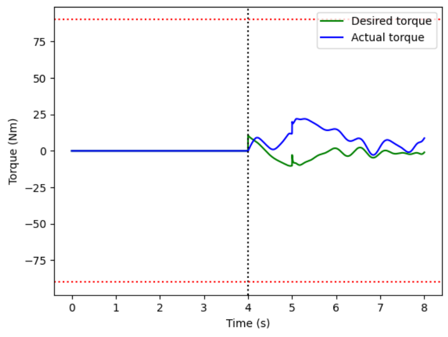


(f) Right Sagittal Hip

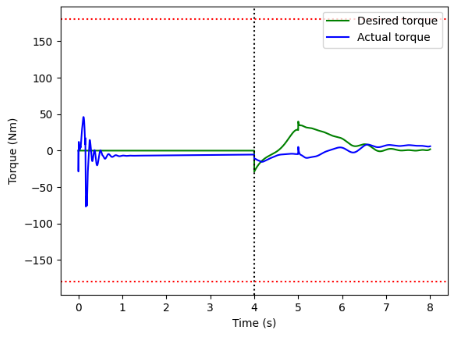
Figure 4.3: **Jiminy**: The chair-to-stand closed-loop trajectories show that the resulting torques respect the torque bounds and closely track the nominal torque trajectories throughout the entire motion. Recall that a PD controller is used for the first 4 seconds. The red dashed lines depict the bounds while the black dashed line indicates the end of the PD controller.



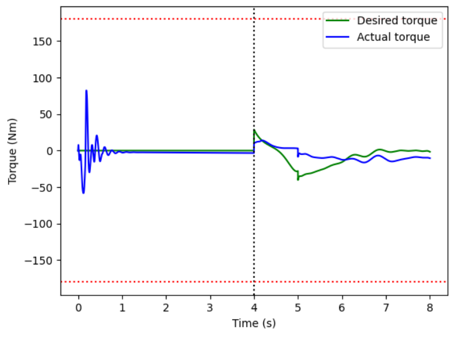
(g) Left Henke Ankle



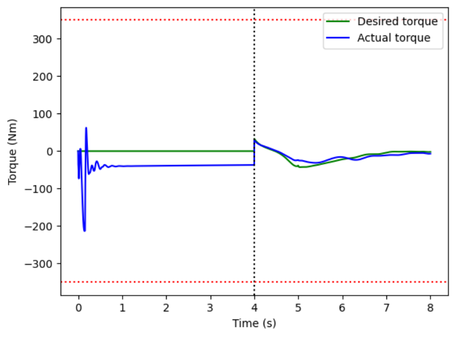
(h) Right Henke Ankle



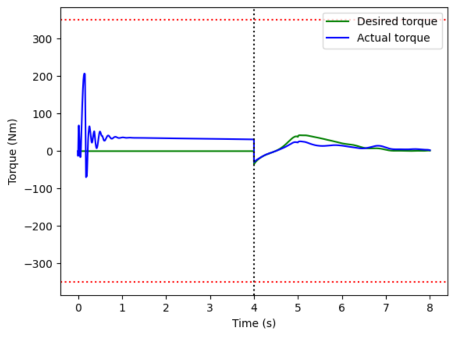
(i) Left Transverse Hip



(j) Right Transverse Hip

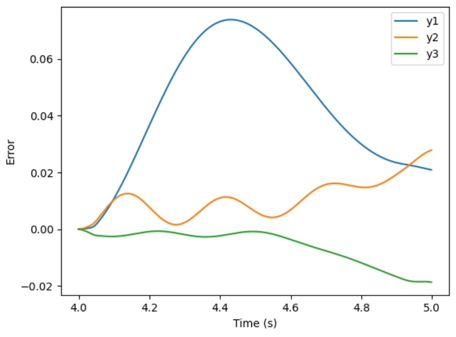


(k) Left Frontal Hip

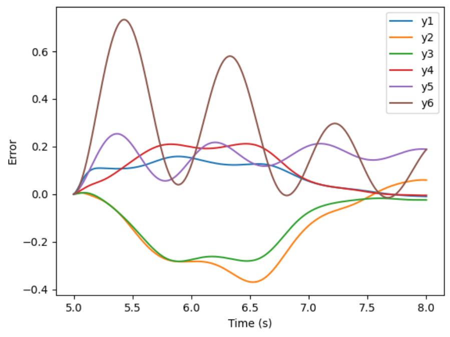


(l) Right Frontal Hip

Figure 4.3: **Jiminy**: The chair-to-stand closed-loop torque trajectories (Cont.)

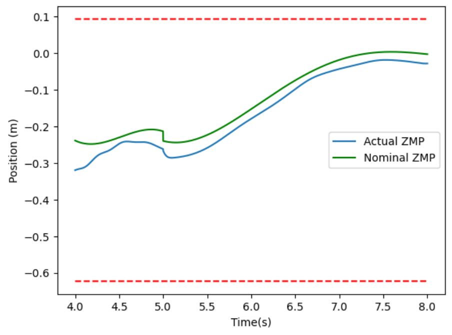


(a) Sitting Tracking Error

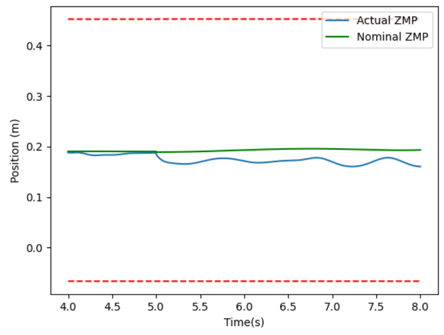


(b) Standing Tracking Error

Figure 4.4: **Jiminy**: Chair-to-standclosed-loop *nominal tracking error* plots. The virtual constraints (y_i) are dimensionless.



(a) ZMP X



(b) ZMP Y

Figure 4.5: **Jiminy**: The resulting ZMP trajectory in comparison to the nominal ZMP trajectory. The red dashed lines are the bounds of the ZMP support polygon.

oscillations and the sagittal ankles operating at their bounds. The joint and torque trajectories are illustrated in Figure 4.7 and Figure 4.8, respectively.

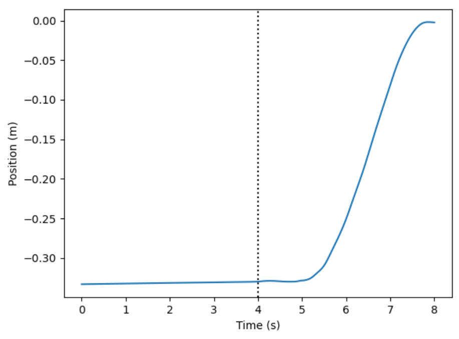
The tracking error is illustrated in Figure 4.9, and similar to the tracking error observed in Jiminy, the maximum value is approximately 0.85. Following our analysis in the previous section, we can conclude that this value is acceptable. Furthermore, it is possible to reduce the tracking error through gain tuning.

Despite not tracking the nominal profile in the x-axis, the ZMP profile remains within the bounds of the support polygon, as demonstrated in Figure 4.10. However, when transitioning from sitting to standing, the feet slide roughly 0.03m in the x direction despite the friction cone constraints being met. Figure 4.11 depicts the friction cone constraints. The sliding phenomenon can, therefore, be attributed to integration drift in the contact constraint. In the next paragraph, we will discuss how to mitigate this drift.

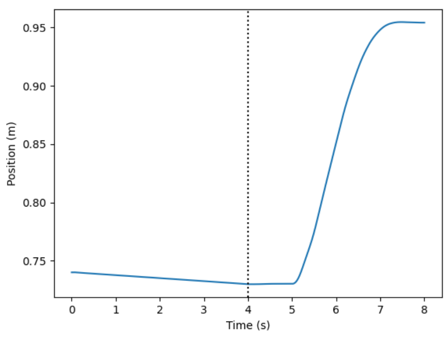
Recall that a contact constraint is defined as follows:

$$\begin{aligned} c &= c_0 - c_d \\ \dot{c} &= J\dot{q} \\ \ddot{c} &= J\ddot{q} + \dot{J}\dot{q} \end{aligned}$$

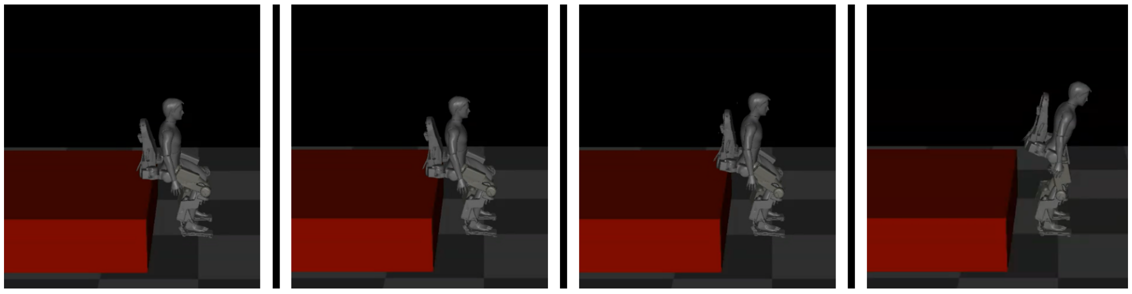
where c_0 and c_d are the current and desired positions of the contact respectively and $J := \frac{\partial c_0}{\partial q}$. We assume c_d is a constant, and hence $\ddot{c}_d = 0$. As discussed in Section 3.2.1, the contact is implemented at the acceleration level and achieved when $c \equiv 0$. $J := \frac{\partial c_0}{\partial q}$. In particular, a contact constraint is not identically satisfied when either $c \neq 0$, $\dot{c} \neq 0$, or $\ddot{c} \neq 0$. Therefore, we correct the accumulated error in c and $\dot{c}$ by defining the acceleration of the contact constraint as:



(a) Base X



(b) Base Z



(c) Snapshots of the exo+mannequin system standing

Figure 4.6: **MuJoCo:** The x and z trajectories of the base and the snapshots of the exo+mannequin system show the system standing up. The y trajectory is not included as it the motion mainly occurs in the sagittal plane. Note that the snapshots depict legs, but the mannequin has no legs.

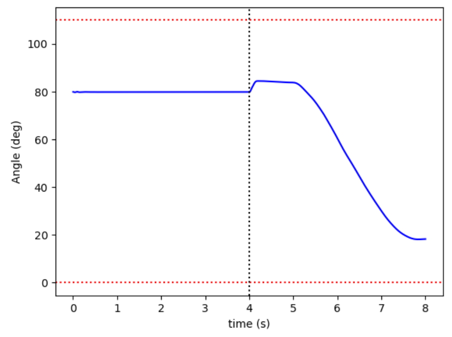
$$J\ddot{q} + \dot{J}\dot{q} = -Kc - D\dot{c} \tag{4.3}$$

where K and D are coefficients that are selected such that $c = 0$.

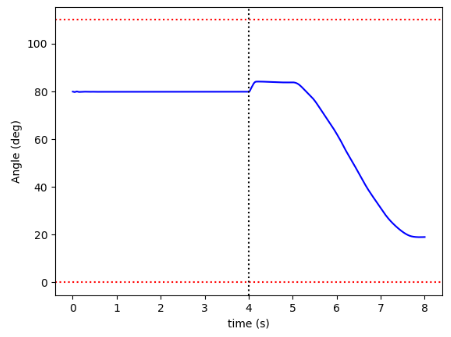
4.5 Conclusion

This chapter extended our analysis from Chapter 3 by simulating closed-loop motions in the MuJoCo and Jiminy simulators which incorporate contact dynamics. The simulations in this chapter involved a switch in the nominal user to a mannequin, differing from our previous approach. Unfortunately, hardware tests could not be conducted due to Atalante breaking.

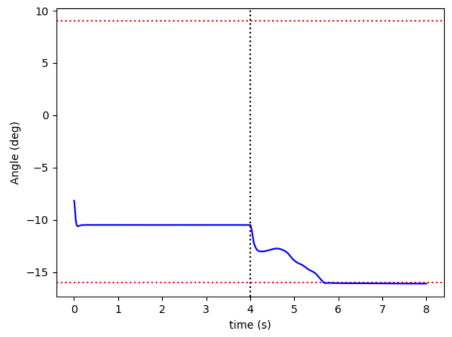
We developed a quadratic-program-based computed torque controller similar to the



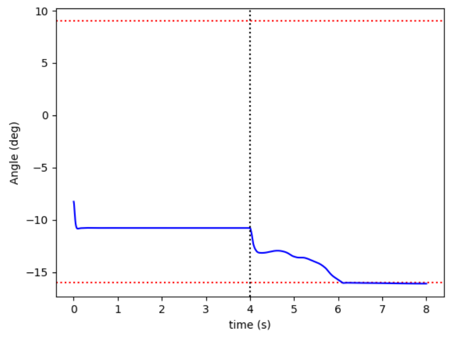
(a) Left Sagittal Knee



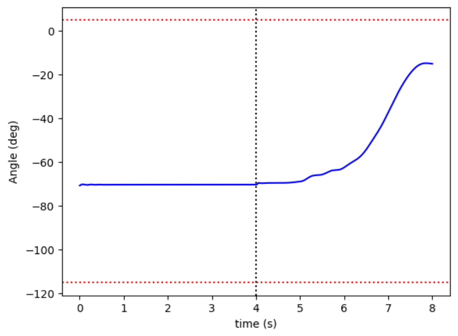
(b) Right Sagittal Knee



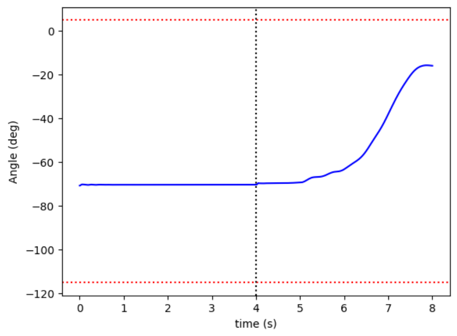
(c) Left Sagittal Ankle



(d) Right Sagittal Ankle

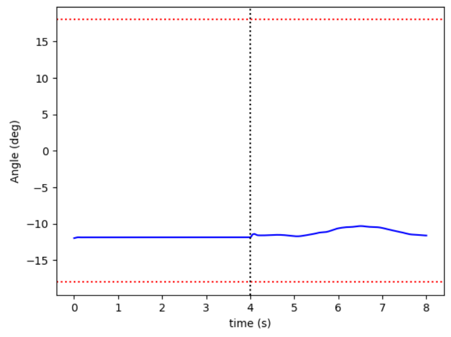


(e) Left Sagittal Hip

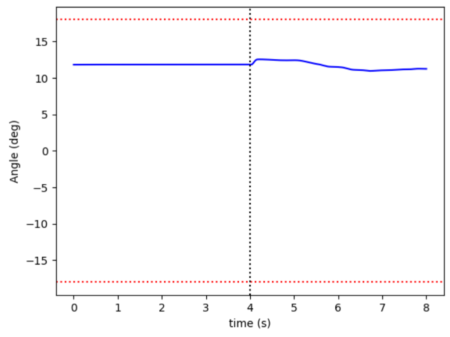


(f) Right Sagittal Hip

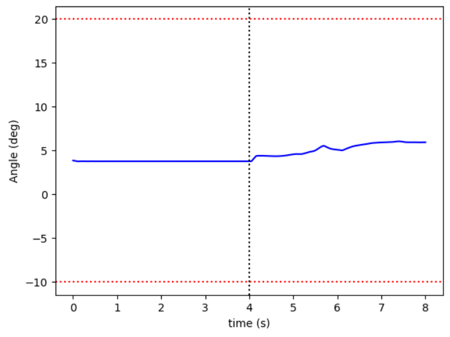
Figure 4.7: **MuJoCo**: The chair-to-stand closed-loop trajectories show that the joints respect the joint constraints. Recall that a PD controller is used for the first 4 seconds. The red dashed lines depict the bounds while the black dashed line indicates the end of the PD controller.



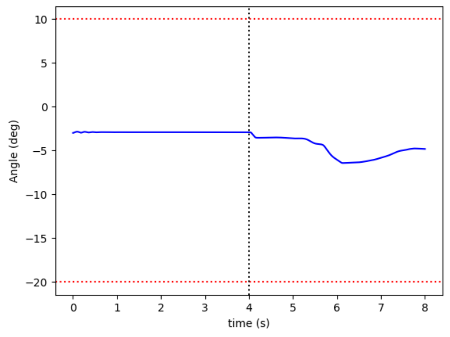
(g) Left Henke Ankle



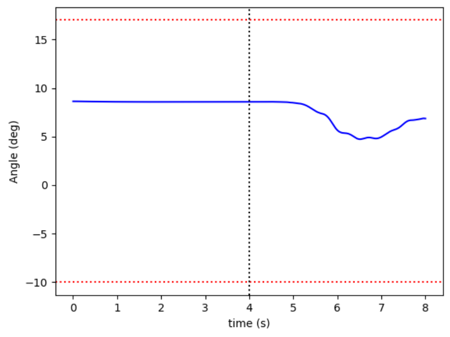
(h) Right Henke Ankle



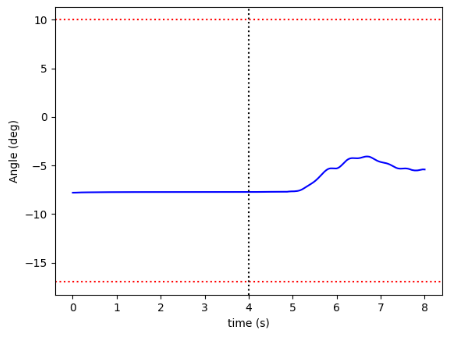
(i) Left Transverse Hip



(j) Right Transverse Hip

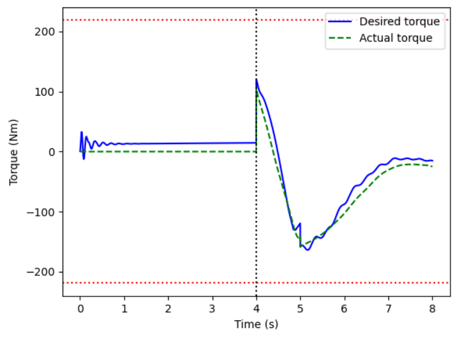


(k) Left Frontal Hip

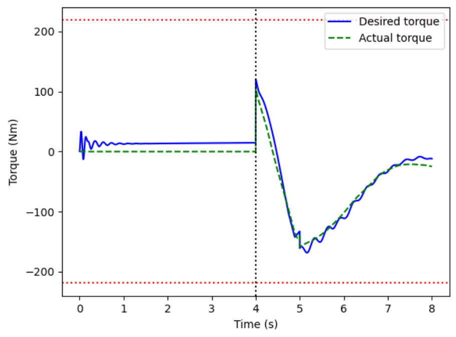


(l) Right Frontal Hip

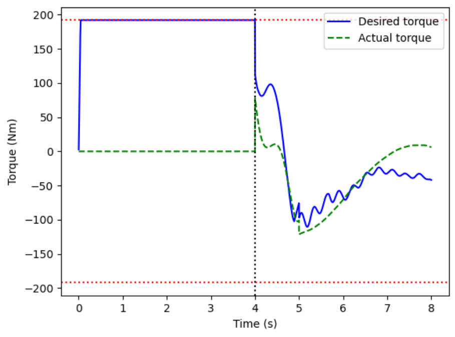
Figure 4.7: **MuJoCo**: The chair-to-stand closed-loop joint trajectories(Cont.)



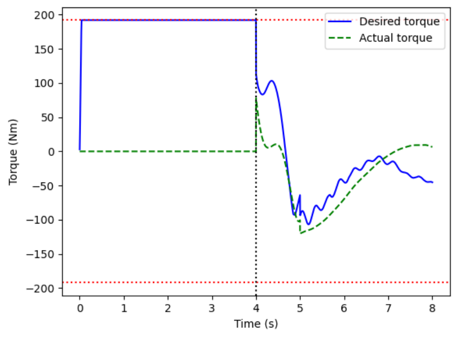
(a) Left Sagittal Knee



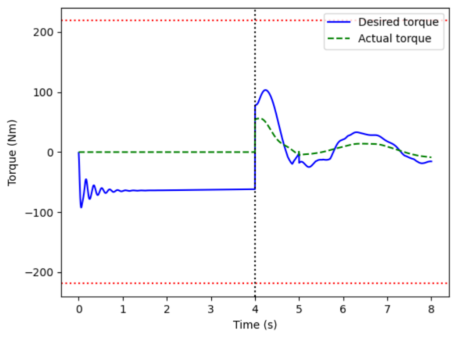
(b) Right Sagittal Knee



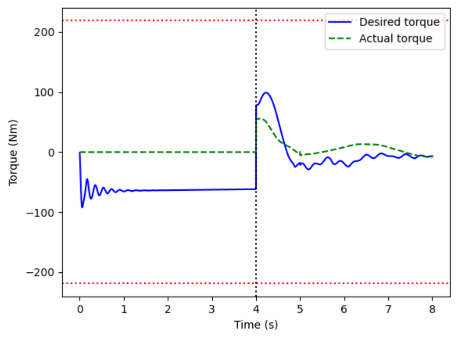
(c) Left Sagittal Ankle



(d) Right Sagittal Ankle

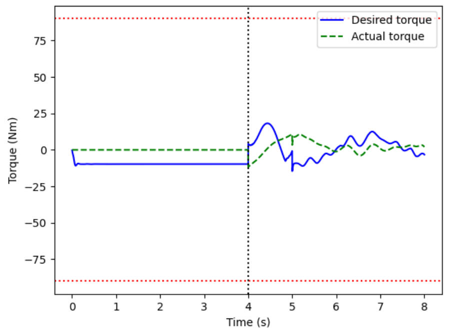


(e) Left Sagittal Hip

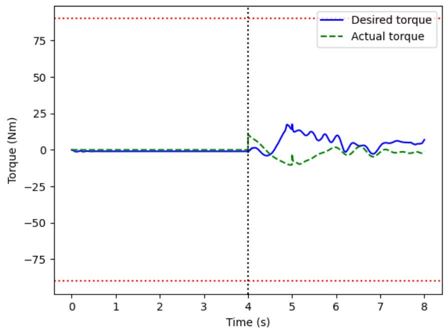


(f) Right Sagittal Hip

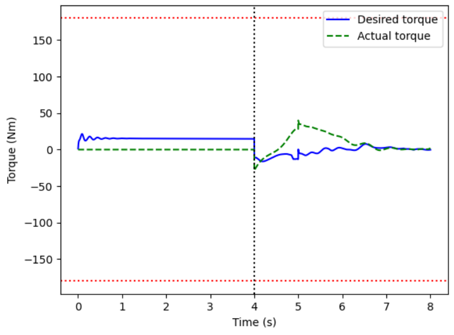
Figure 4.8: **MuJoCo**: The chair-to-stand closed-loop trajectories show that the resulting torques respect the torque bounds and closely track the nominal torque trajectories throughout the entire motion. Recall that a PD controller is used for the first 4 seconds. The red dashed lines depict the bounds while the black dashed line indicates the end of the PD controller.



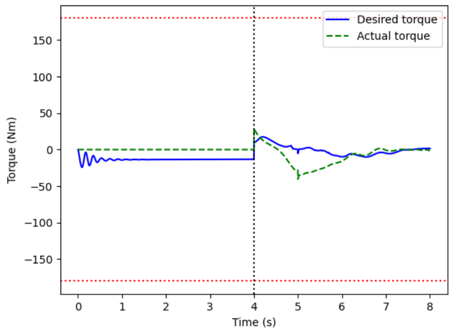
(g) Left Henke Ankle



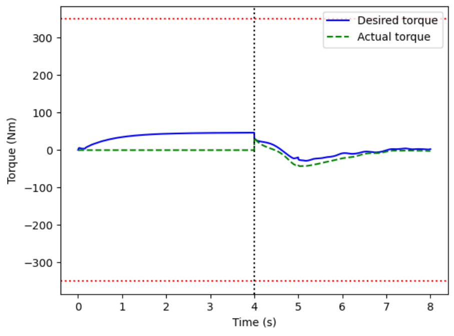
(h) Right Henke Ankle



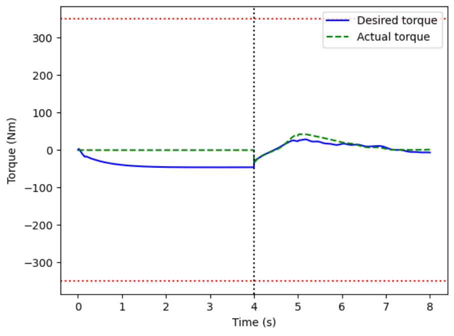
(i) Left Transverse Hip



(j) Right Transverse Hip

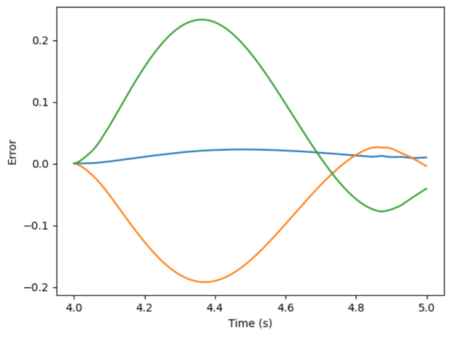


(k) Left Frontal Hip

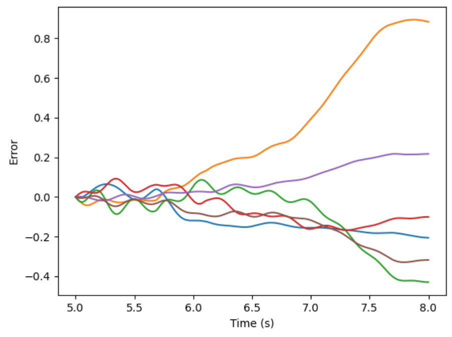


(l) Right Frontal Hip

Figure 4.8: **MuJoCo**: The chair-to-stand closed-loop torque trajectories (Cont.)

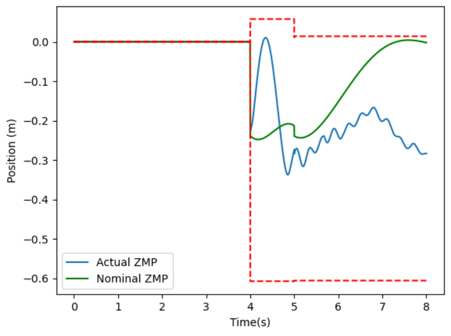


(a) Sitting Tracking Error

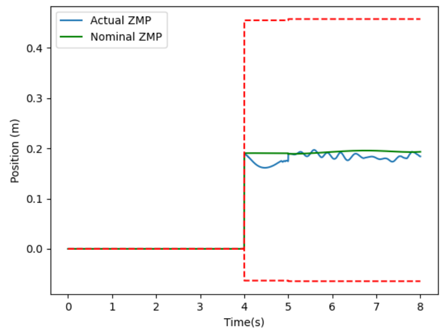


(b) Standing Tracking Error

Figure 4.9: **MuJoCo**: Chair-to-standclosed-loop *nominal tracking error* plots. The virtual constraints (y_i) are dimensionless.

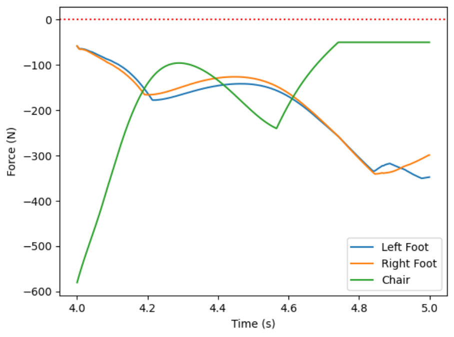


(a) ZMP X

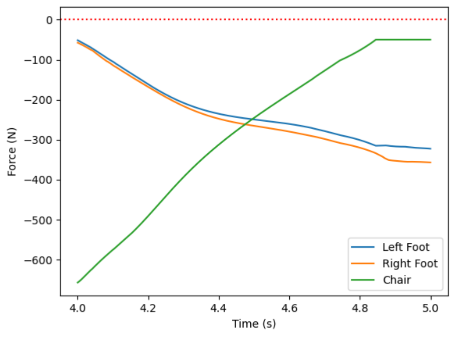


(b) ZMP Y

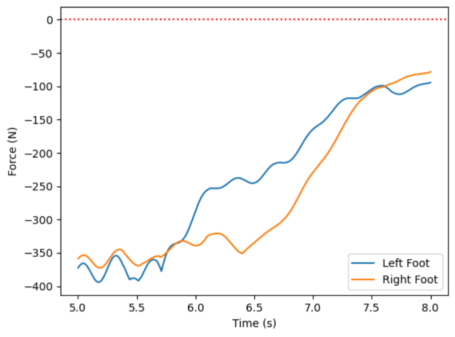
Figure 4.10: **MuJoCo**: The resulting ZMP trajectory in comparison to the nominal ZMP trajectory. The red dashed lines are the bounds of the ZMP support polygon.



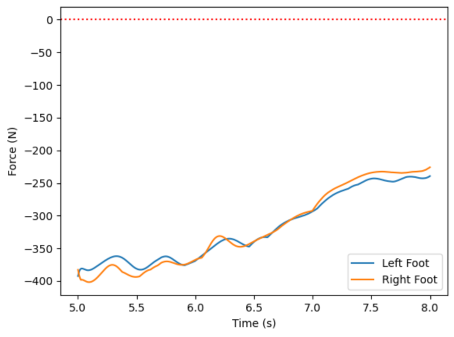
(a) Sitting X-axis Friction Constraint



(b) Sitting Y-axis Friction Constraint



(c) Standing X-axis Friction Constraint



(d) Standing Y-axis Friction Constraint

Figure 4.11: **MuJoCo:** The chair and feet friction cone constraints

SU controller to track the nominal chair-to-stand trajectory outlined in Chapter 3. During implementation in Jiminy, challenges emerged when transitioning from sitting to standing due to the ankle torques causing the feet to roll or pitch. This problem was resolved by adding mass to the ankles. The resulting closed-loop motion respected the joint and torque bounds and closely tracked the nominal torque and ZMP trajectories. Although a peak tracking error of 0.7 was observed, this value was deemed acceptable considering user discrepancies, the controller's tracking of the nominal ZMP and torque trajectories, and the dimensionless virtual constraints.

Switching to MuJoCo due to the ankle mass adjustments in Jiminy, preliminary closed-loop motions showed successful standing without added ankle weight. The resulting motion closely followed nominal torque trajectories, adhered to joint and torque bounds, respected friction cone constraints, and maintained ZMP within support polygon bounds. However, a 0.03m sliding of the feet along the x-axis was noted. The sliding was attributed to integration drift, and a solution was discussed. Oscillations in motor torques, the sagittal ankles operating at their lower bounds, and a maximum tracking error of 0.85 were observed in the standing domain but deemed improvable with gain tuning. Note that the tracking error of 0.85 was also deemed acceptable, in line with the Jiminy results.

Chapter 5

Fall Prediction of a Planar Bipedal Robot

5.1 Introduction

This chapter develops and analyzes physics-based and data-based fall prediction algorithms for the planar four-link robot. The core work presented in this chapter was previously published in [132] and presented as a late-breaking abstract in the poster session and the Robot Safety workshop during IROS 2023. The late-breaking abstract can be found at the following link `https://www.evamungai.com/fall-prediction-planar-four-link`. The co-author of [132] was Jessy W. Grizzle.

5.1.1 Motivation

As discussed in Chapter 1, the objective of a fall prediction algorithm is to detect all critical faults in a timely and accurate manner. However, early fall prediction is challenging due to the masking effects of controllers (through their disturbance attenuation actions), the direct relationship between lead time and false positive rates, and the temporal behavior of the faults. To simplify the fall prediction problem while providing a pathway to scale up to more complex dynamic motions and robots, this chapter explores the development of a fall prediction algorithm for the task of standing with the planar four-link robot. As the task of interest is standing, we define a “fall’ as any link other than the feet coming in contact with the ground [68] or the feet being off the ground.

5.1.2 Literature Review

A fall prediction algorithm has been implemented in the commercial bipedal robot, Digit [2]. However, this does not appear to be the norm. The objective of most bipedal fall prediction algorithms found in literature is to reliably and promptly detect all abrupt faults. Incipient and intermittent faults have not been addressed in the literature.

Fault detection reliability has been addressed using a combination of evaluation terms from the confusion matrix, such as false positive and negative rates [7–9,34–38,47,48]. Thresholds, based on factors such as the center of mass height, have been proposed to minimize the percentage of false negative fault declarations in [8], while the output of the fall prediction algorithm is monitored for a certain number of windows ($N_{monitor}$) to reduce the false positive rate in [8,35,133]. Lead time, defined as the difference between the time of the actual fall and the predicted fall, is used to inform whether or not sufficient time is left for the implementation of recovery/reflexive motions. It is desirable to have a large lead time; however, maximizing lead time can increase false positive rates.

Fall prediction algorithms can either rely on physics-based [39–41] or data-based models [8,9,34,35,38,47,48]. Physics-based models can suffer from model inaccuracies, while data-based models are limited by the amount of data available. For both physics-based and data-based models, the objective is to obtain either a model of the nominal (safe) states and/or of the faulty (unsafe) states. However, neither of these models is straightforward to produce. In practice, it is infeasible to quantify all faults that can lead to a fall, and the faulty states in any given trajectory are irregular and rare. Both of these conditions make it nearly impossible to obtain an accurate model of the anomalies. It is also challenging to obtain a model that accounts for all the safe states of the robot. However, due to advances in the machine learning community, data-based algorithms are becoming more common.

Data-driven detection algorithms can be divided into two subsequent parts: feature engineering and the method used for detection. Feature engineering consists of selecting and transforming raw data into features that can differentiate between faulty and normal states. Even though stability metrics are used to increase the robustness of controllers, they individually do not provide sufficient conditions for falling [35]. A combination of stability metrics from bipedal control theory, such as the angular momentum about the center of mass (L_{com}), and kinematic functions, such as the center of mass position (p_{com}), are typically chosen as features [8, 9, 34, 35, 38, 47, 48].

Classification algorithms, such as that used by [8], attempt to learn a model from labeled training data and then classify a data point into one of the classes based on the learned model. A disadvantage of classification algorithms is that they can output incorrect predictions if the input data is outside the training data parameters (outside distribution). Nearest-neighbor-based algorithms, such as [133], assume that normal data exist in highly dense spaces, whereas the neighborhood of anomalous data is sparse. However, these algorithms can have high false positives if the normal instances do not exist in sufficiently dense neighborhoods. Threshold-based algorithms, such as [35], attempt to use a combination of features to derive a threshold that can be used to separate faulty and normal states.

5.1.3 Objective of the Chapter

For the task of standing and for given **upper bounds on the false positive and false negative rates**, our objective is to detect potential falls caused by either **incipient, abrupt or intermittent faults** while **maximizing the lead time**, that is, the time from fault declaration to the robot entering a fallen state. The objective is challenging due to the crowding phenomenon [134, 135], masking effects of the controller as it tries to mitigate deviations from steady state [6], the direct relationship between lead time and false positive rate, and the sporadic nature of intermittent faults. The crowding phenomenon is the similarity between the normal and incipient faulty data which makes it difficult to separate normal data from faulty data [134, 135].

We address this objective in two parts: first, the detection of critical abrupt and incipient faults, and second, the detection of all three critical faults. To achieve the first task, we design a nearest-neighbor, a physics-based, and a neural-network-based fall prediction algorithm and compare their performance to an existing classification-based detection algorithm. For the second task, we propose a multi-classification algorithm. A threshold-based method was not chosen for any of the tasks because it is difficult to find simple thresholds for systems as complex as bipedal robots.

5.1.4 Contributions

The major contributions of this chapter are as follows:

- An algorithm that maximizes lead time subject to bounds on false positive and negative rates;
- A method of identifying trajectories associated with incipient or abrupt faults;
- A way to label the data based on lead time is proposed;
- A nearest-neighbor classification-based fall prediction algorithm that can detect incipient and abrupt faults;
- A comparison of the proposed nearest-neighbor classification algorithm and an existing classification algorithm;
- A multi-classification algorithm that can detect incipient, abrupt, and intermittent faults; and
- A physics-based method that can detect abrupt and incipient faults

5.2 Robot Description and Data Generation

In this section, we describe the robot model that is used for this study and how the data are generated and prepared.

5.2.1 Equations of Motion and Simulation Environment

The equations of motion are given by (5.1) and (5.2)

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + J^T(q)\Gamma \tag{5.1}$$

$$J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} = 0, \tag{5.2}$$

where q is the vector of generalized coordinates defined by (5.3), u is the torque input vector, D, C, G, and B are the inertia, Coriolis, gravity, and torque distribution matrices/vector, respectively, J is the Jacobian mapping the contact wrenches to the generalized coordinates, and Γ is the contact wrench. The floating-base Lagragian model is given by (5.1) while the contact/acceleration constraint is given by (5.2) [51,54,83].

$$q = \begin{bmatrix} \text{foot x} \\ \text{foot z} \\ \text{foot angle } (\theta_f) \\ \text{ankle angle } (\theta_a) \\ \text{knee angle } (\theta_k) \\ \text{hip angle } (\theta_h) \end{bmatrix}. \tag{5.3}$$

A PD controller to maintain the robot in a standing position was designed and implemented in MATLAB [136]. The PD controller seeks to keep the foot flat on the ground while maintaining the center of mass (CoM) inside the support polygon. The simulation environment uses MATLAB's ODE45 function and compliant ground contact forces represented as a spring-damper.

5.2.2 Data Generation

Four hundred trajectories each are generated for abrupt and incipient faults, with a sampling time of 0.03s and a disturbing force applied to the torso. To emulate disturbances that might cause the robot to oscillate slightly while standing, a random impulse force in the range of 0-159N and lasting for 0.075s is applied at time zero, but only the data after 2 seconds is kept. The abrupt faults last for 0.075 seconds with magnitudes ranging from 0-320N , while the incipient faults last for 1.0 seconds and range from 0-46N. The ranges, based on previous experiments, are chosen such that half the trajectories end in a fall (we'll refer to these trajectories as *faulty trajectories*), and half of the safe (non-falling) trajectories have a heel or toe lift. Similar to [8], the force magnitudes are generated using a uniform distribution. The abrupt force is applied at random between 2.5s and 3.5s, while the incipient fault is applied at random between 2.0s and 3.5s. The application time for the incipient fault is longer because, in order not to include an abrupt deviation in the robot's nominal states, only the data collected after the force is applied is kept.

Forty intermittent trajectories with a random mix of non-continuous incipient and abrupt faults are also generated. Each intermittent trajectory contains two faults. The time of application for each fault is determined by the fault type as described in the previous paragraph. There is a 1s gap between the intervals of application for the two forces.

5.2.3 Data Pre-processing

The features are selected as

$$\begin{bmatrix} L_{cop} - L_{com} \\ p^x_{com} \\ v^x_{com} \\ (p_{toe} - p_{com})^x \\ (p_{heel} - p_{toe})^{xz} \\ (L_{cop} + L_{com}) * sgn(p^x_{com} - p^x_{f_{mid}}) \end{bmatrix} \tag{5.4}$$

where v_{com} is the CoM velocity, p_{toe}, p_{heel}, and $p_{f_{mid}}$ are the position of the toe, heel and middle of the foot, and L_{cop} is the angular momentum about the contact point[1]. These features are chosen based on their correlation with the lead time and other features commonly used in literature. The distance correlation coefficient is used to evaluate the correlations as it is able to capture nonlinear relationships [137].

The features are split into training (60%), validation (20%), and testing (20%) sets using scikit-learn's [4] stratified train_test_split method and k-folds methods with the number of folds set to 5. The stratified methods are chosen because they ensure that each of the splits has the same distribution of normal and faulty data. Scikit-learn's min-max scaler is used to scale the data to a range of $[0, 1]$. To ensure that only transient data is kept for training, only the first 6s of trajectories that are deemed normal are kept.

5.3 Fall Prediction Methods

As a baseline, we use the SVM-based classification algorithm of [4], while the nearest-neighbor classification algorithm is based on the Ward minimum variance method

[1]The contact point is set to the rotation point (toe or heel) when the foot rotates, and the center of pressure otherwise.

[138]. To prioritize recent data points over previous ones, both methods make use of sliding windows. The number of data points in a window is referred to as N_{window}. It is important to note that both methods are supervised algorithms.

5.3.1 SVM Classifier

The radial basis function is chosen as the kernel and the soft margin formulation is implemented for the SVM classifier (SVM classification algorithm). The training data for the classifier is defined as

$$D = \{X_i, y_i\}_{i=1}^{n}$$

where

$$\begin{aligned}
n &= \text{number of windows across all training data} \\
m &= \text{number of time steps in a window} \\
x_{ij} &= \text{features at time step j in window i} \\
X_i &= \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{im} \end{bmatrix}^{\mathsf{T}} \\
y_i &\in \left\{ \begin{array}{cccc} -1 & X_i & \in & \text{faulty trajectory} \\ 1 & X_i & \in & \text{normal trajectory} \end{array} \right\}
\end{aligned}$$

5.3.2 Nearest-Neighbor Classification Algorithm

The selected nearest-neighbor classification algorithm determines distance using the Ward minimum variance and a weighted Euclidean distance. Given two clusters A and B, the Ward minimum variance method calculates the effort, E_{AB}, it takes to join the two clusters together, as determined by the sum of squared errors, specifically,

$$E_{AB} = SSE_{AB} - SSE_A - SSE_B, \tag{5.5}$$

where

$$\begin{aligned}
&SSE_{AB} = (A \cup B - \mu_{A\cup B})^{\intercal} R_{A\cup B}^{-1} (A \cup B - \mu_{A\cup B}) \\
&SSE_{A} = (A - \mu_{A})^{\intercal} R_{A}^{-1} (A - \mu_{A}) \\
&SSE_{B} = (B - \mu_{B})^{\intercal} R_{B}^{-1} (B - \mu_{B}) \\
&R = \text{correlation coefficient matrix} \\
&\mu = \text{mean vector}
\end{aligned}$$

In our application, cluster B contains the features at the current time step while cluster A contains all the features in the previous time steps included in the window. Given, that B is a single data point, the Ward minimum variance simplifies to

$$E_{AB} = SSE_{AB} - SSE_{A}. \tag{5.6}$$

The nearest-neighbor classification algorithm detects a potential fall if the effort it takes to join the two clusters A and B is higher than a threshold determined from the training data. The threshold is calculated offline as the maximum E_{AB} value for the safe data while R is determined using distance correlation. Note that the underlying assumption for the nearest-neighbor classification algorithm is that cluster A only contains safe data points.

5.3.3 Data Labeling

As one of our objectives is to maximize the lead time, we propose the use of a training lead time to label windows in a trajectory. Training lead time is defined as the difference between the time of the actual fall and the time when a sliding window of a trajectory can be labeled as faulty. Therefore, training lead time is a subset of the maximum lead time that can be achieved in a faulty trajectory[2]. While labeling all windows in a faulty trajectory as faulty would achieve the maximum lead time, it would also increase the rate of false positives. For instance, given two faults that are close in magnitude but where one results in a fall and the other is safe, the safe trajectory could be mistaken as a faulty trajectory.

If a trajectory does not contain a fall, all windows derived from the trajectory are labeled as 1. If a trajectory ends in a fall, all windows containing data points after the desired training lead time are labeled as -1. Note that for an abrupt fault, the training lead time is only defined after the push is introduced, and only the data points before a fall are kept for the training data of both faults.

The desired training lead time is determined by a grid search algorithm that trains the algorithm of interest using a range of training lead times from 0 to 2s and evaluates the results on the training and/or validation data. The training data is included in the evaluation process for cases where the algorithm is allowed to make mistakes, such as when using a soft margin in SVM. A training lead time of 2s would label all the data points in a faulty trajectory as faulty [3].

5.3.4 Performance of Fall Prediction Methods

In this section, we analyze the performance of the proposed nearest-neighbor classification algorithm and the baseline SVM classifier. The algorithms are trained and evaluated on testing data across all 5 folds using only abrupt trajectories, only incipient trajectories, and both trajectories together. The evaluation metrics are false positive and negative rates, and the average lead time achieved. The desired false positive and false negative rates are set to 0. The training lead time chosen is the maximum that meets the given bounds on the false positive and false negative rates when evaluated on the training and validation data. Based on previous experiments, we set the values

[2]In other words, the training lead time is less than or equal to the maximum lead time.

[3]All the faulty trajectories fall within 2s.

of the remaining hyper-parameters as $N_{window} = 10$ and $N_{monitor} = 1$.

From Table 5.1 and 5.2 we see that the nearest-neighbor and the SVM classification algorithms perform similarly when trained and evaluated on the abrupt and incipient faults separately. The nearest-neighbor classification algorithm achieves an average lead time of 0.46s and 0.91s, respectively, for the abrupt and incipient faults, while the SVM classification algorithm achieves an average lead time of 0.48s and 0.97s. Because our sampling time is 0.03s, the difference in the performance of both algorithms is 1 and 2 data points for the abrupt and incipient fault, respectively. Figure 5.1 displays the classification results for several trajectories.

When both faults are trained and evaluated together, the SVM classification algorithm achieves an average lead time 0.15s higher compared to the nearest-neighbor classification algorithm. In comparison to its average performance on the abrupt and incipient fault, the SVM classification algorithm achieves an average lead time of 0.08s less when trained on both faults together. Similarly, the nearest-neighbor classification algorithm achieves an average lead time of 0.19s less. As a result, the SVM classification algorithm outperforms the nearest-neighbor classification algorithm when both faults are trained together. However, because both algorithms can achieve lead times higher than the 0.2s, which is the lead time required by reflexive algorithms such as [34] and [9], both algorithms are viable options. As the nearest-neighbor classification algorithm learns the safe/good model, it should be used when faulty data is sparse.

5.3.5 Categorizing Faults

A means to decrease the difference in performance for both algorithms when trained on both faults together vs. separately is to implement a multi-class classification problem. The labels for this multi-class classification can be identified as: abrupt fault safe (AS), abrupt fault fall (AF), incipient fault safe (IS), and incipient fault fall

(IF). Using these labels with the one-vs-one or one-vs-rest multi-class classification techniques typically implemented [139], results in six and four detectors, respectively. However, as one-vs-rest can result in ambiguities and class imbalances and using one-vs-one can result in ambiguities and higher computational times, we seek a different approach [139].

If the problem is decomposed into classifying trajectories first into the incipient versus abrupt categories, and secondly, detecting falls (or not) within these categories of trajectories, the number of detectors needed is only three: a detector for identifying types of trajectories, a second for detecting falls in incipient trajectories, and a third for detecting falls in abrupt trajectories. Furthermore, using this technique resolves the ambiguity problem as the incipient vs. abrupt classifier can be used to determine the operational space (abrupt vs incipient fault).

To achieve this, we propose using SVM to categorize the trajectories into incipient vs abrupt. The training data for this SVM are taken as the joint velocities, and the labels 1 and -1 are used for the incipient and abrupt faults, respectively. For the training data, the windows in abrupt trajectories before a force is applied and windows uniformly distributed throughout the incipient trajectories are labeled as incipient and only the windows containing the force are labeled as abrupt. The remainder of the pre-processing steps are similar to those in Section 5.2.3.

5.4 Multi-class Classification Prediction Method

The proposed multi-class classification fall prediction method, as shown in Figure 5.2, is comprised of three algorithms, one for detecting falls caused by abrupt faults (abrupt fault detector), another for detecting falls caused by incipient faults (incipient fault detector) and a third for identifying the type of fault (fault type identifier).

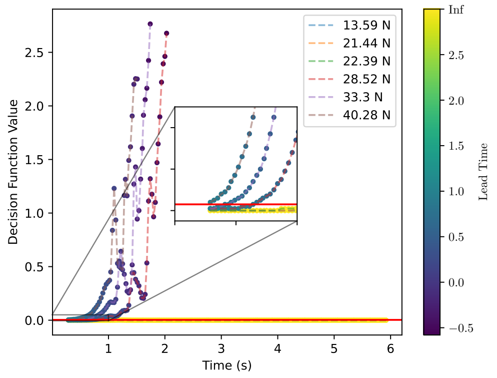
The fault type identifier is first trained and evaluated on the training data. Next, the abrupt fault detector and incipient fault detector are trained using the relevant

Table 5.1: A comparison of the nearest-neighbor classification algorithm's performance when trained with (1) just the abrupt fault, (2) just the incipient fault, and (3) both faults together. Note that the false positive and negative rates are 0.

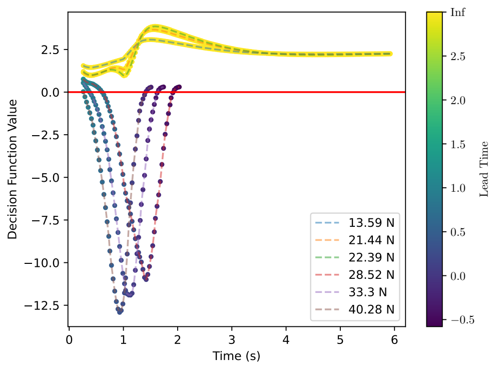
	Abrupt Fault Only	Incipient Fault Only	Both Faults Together
Fold	Average Lead Time	Average Lead Time	Average Lead Time
1	0.48	0.93	0.49
2	0.46	0.91	0.49
3	0.44	0.9	0.51
4	0.43	0.89	0.51
5	0.5	0.89	0.51
Average	0.46	0.91	0.50

Table 5.2: A comparison of the SVM classification algorithm's performance when trained with (1) just the abrupt fault, (2) just the incipient fault, and (3) both faults together. Note that the false positive and negative rates are 0.

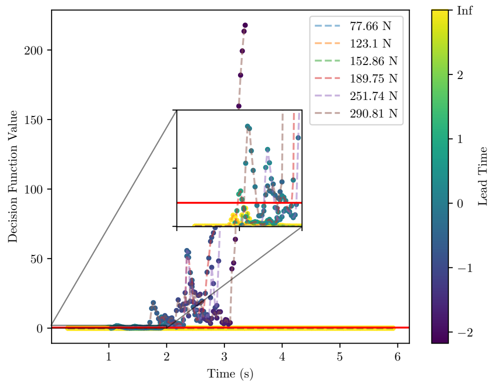
	Abrupt Fault Only	Incipient Fault Only	Both Faults Together
Fold	Average Lead Time	Average Lead Time	Average Lead Time
1	0.5	1.0	0.65
2	0.47	0.96	0.66
3	0.49	0.98	0.66
4	0.44	0.97	0.65
5	0.51	0.96	0.63
Average	0.48	0.97	0.65



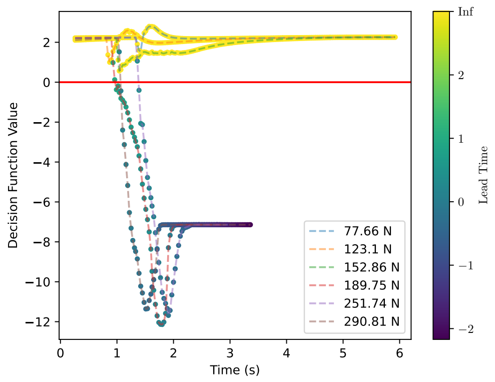
(a) Nearest-Neighbor Incipient Fault



(b) SVM Incipient Fault



(c) Nearest-Neighbor Abrupt Fault



(d) SVM Abrupt Fault

Figure 5.1: Plots displaying the classification results of the nearest-neighbor and SVM classification algorithms for several trajectories. The red solid line is the threshold for the decision function. The dots are the last data point in a window. Positive and negative values for the SVM decision function result in safe and faulty classifications, respectively. On the other hand, values below and above the decision function threshold are classified as safe and faulty for the nearest-neighbor classification algorithm.

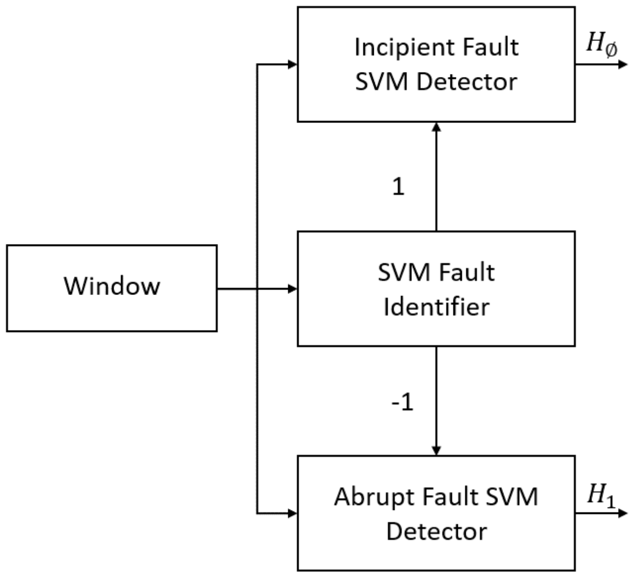


Figure 5.2: The proposed multi-classification algorithm

trajectories and the windows of trajectories misclassified by the fault type identifier. The training lead time is determined similarly as in Section 5.3.4.

When detecting potential faults, we run all three algorithms in parallel. As we initially assume that every trajectory has an incipient fault, the output of the incipient fault detector is utilized by default. However, if the fault identifier classifies a trajectory as containing an abrupt fault, we start using the output of the abrupt fault detector. In other words, our null hypothesis is the incipient fault, while our alternative hypothesis is the abrupt fault. As a result, a delay in the fault identifier only results in a delay in the abrupt fault detector. When an abrupt fault is identified, the fault identifier is no longer used to identify the fault type until it is reset. Inherent in our implementation is that only one fault will be encountered per trajectory.

5.4.1 Results: Critical Abrupt and Incipient Faults

We train and evaluate the multi-class classification algorithm using the same parameters and metrics as described in Section 5.3.4. On average, across all folds,

Table 5.3: Features derived from scikit-learn's [4] sequential forward feature selection

Incipient Fault Features	Abrupt Fault Features
$\begin{bmatrix} \text{knee angle} \\ \text{hip angle} \\ \text{vel hip angle} \\ (L_{cop}+L_{com})*sgn(p^x_{com}-p^x_{f_{mid}}) \end{bmatrix}$	$\begin{bmatrix} p^x_{com} \\ v^x_{com} \\ p^x_{com}-p^x_{heel} \\ \text{foot x} \\ \text{vel foot z} \\ \text{vel hip angle} \end{bmatrix}$

when trained using the features in (5.4), the multi-class classification algorithm achieves 0.06s and 0.05s additional average lead time across all folds for the nearest-neighbor and SVM classification algorithms, respectively. This results in an average lead time difference of 0.13s and 0.03s across all folds for the nearest-neighbor and SVM classification algorithms in comparison to their average when trained with the incipient and abrupt faults separately. Note that the SVM fault identifier has a delay of 0.07s or 3 data points, in detecting abrupt faults across all folds. The results are displayed in Table 5.4 and 5.5.

Even though the multi-class classification algorithm achieved similar results to the binary classification algorithm, an advantage over binary classification is that different features can be used for each detector. Feature selection algorithms such as sequential feature selection can be used to determine the optimal features. For instance, using the features shown in Table 5.3 derived from scikit-learn's [4] sequential forward feature selection results in an average lead time increase of 0.1s over training a binary classification with (5.4). However, to truly take advantage of the multi-class classification algorithm more investigation into optimal feature selection is needed to determine whether the additional average lead time gained can overcome the fault identifier delay.

Table 5.4: A comparison of the maximum average lead time achieved by the binary nearest-neighbor classification algorithm and the multi-class classification algorithm with nearest-neighbor fault detectors for abrupt and incipient faults

Fold	Multi-class Classification Average Lead Time	Binary Nearest-Neighbor Average Lead Time
1	0.56	0.49
2	0.52	0.49
3	0.57	0.51
4	0.56	0.51
5	0.57	0.51
Average	0.56	0.5

Table 5.5: A comparison of the maximum average lead time achieved by the binary SVM classifier and the multi-class classification algorithm with SVM fault detectors for abrupt and incipient faults

Fold	Multi-class Classification Average Lead Time	Binary Classification Average Lead Time
1	0.7	0.65
2	0.73	0.66
3	0.70	0.66
4	0.70	0.65
5	0.67	0.63
Average	0.7	0.65

5.4.2 Results: Critical Intermittent Faults

To evaluate the multi-classification fall prediction algorithm on intermittent faults, we (1) modify the algorithm to allow it to go back and forth between the null (incipient fault) and alternate hypothesis (abrupt fault), (2) train it using abrupt and incipient data, and (3) test its performance on the intermittent fault data. Using the features described in (5.4), we find that the algorithm successfully detects intermittent faults with 0 false positive and false negative rates and an average lead time of 0.82s across all folds. Given that the achieved average lead time exceeds the 0.2s threshold required by reflexive algorithms such as [34], we can conclude that the proposed multi-classification algorithm can successfully detect critical intermittent faults.

5.5 A Shift Towards Neural Networks: Critical Abrupt and Incipient Faults

Given the significant influence of the chosen features on the resulting lead time and false positive rate as demonstrated in Section 5.4.1, we propose a shift to a neural network-based fall prediction algorithm. We specifically choose a 1D convolutional neural network (CNN) due to its equivariance to translation and its ability to recognize local patterns when processing data with a grid-like topology. Note that the trajectory of a bipedal robot can be retrieved by a 1D grid taking samples at fixed time [140].

5.5.1 Results

The 1D CNN model is implemented in Pytorch [141]. It has two 1D CNN layers, a max pooling layer, a drop-out layer, and two fully connected layers. The binary cross entropy with logits loss and ReLu activation functions are used. A regularization function that uses the false positive and negative rates, and the lead time for the validation and training trajectories is implemented.

Using the features displayed in (5.4), the resulting neural network achieves an average lead time of 0.74s when both faults are trained together. This is two data points faster than the multi-classification algorithm with the SVM fault detector and separate features for the abrupt and incipient faults (Table 5.3 lists the features). The CNN model likely achieves similar results to the multi-classification algorithm because the fall prediction problem posed in Chapter 5 for the four-link robot is simple. The results are displayed in Table 5.6.

When the features are increased by including variables such as the states and their velocities as displayed in (5.7), the CNN model trained with both abrupt and incipient faults together attains an average lead time of 0.77s. This is 0.03s and 0.07s faster than the results attained by the CNN model using the features in (5.4) and the multi-classification algorithm, respectively. The results are displayed in Table 5.6.

$$\begin{bmatrix} L_{cop} \\ L_{com} \\ p_{com} \\ p_{cop} \\ v^x_{com} \\ (p_{toe} - p_{com})^x \\ (p_{heel} - p_{toe})^{xz} \\ (L_{cop} + L_{com}) * sgn(p^x_{com} - p^x_{f_{mid}}) \\ q \\ \dot{q} \end{bmatrix} \tag{5.7}$$

5.6 Physics-Based Fall Prediction: Critical Abrupt and Incipient Faults

Data-based methods such as the multi-classification and 1D CNN-based fall prediction algorithms can perform poorly when evaluated on out-of-distribution data. As a result,

Table 5.6: A comparison of the maximum average lead time achieved by the 1D CNN algorithm trained with abrupt and incipient faults, and using the features in (5.4) the features in (5.7).

Fold	Features from (5.4)	Features from (5.7)
1	0.72	0.74
2	0.66	0.75
3	0.67	0.71
4	0.88	0.87
5	0.77	0.76
Average	0.74	0.77

we investigate the use of a physics-based method. The proposed physics-based fall prediction algorithm is based on the contact interaction matrix introduced in Section 3.4.2. Recall that the contact matrix is defined as follows:

$$\mathscr{I}(q) := \begin{bmatrix} J(q)D^{-1}(q)J^{\top}(q) & J(q)D^{-1}(q)B \\ & \\ J_h(q)D^{-1}(q)J^{\top}(q) & J_h(q)D^{-1}(q)B \end{bmatrix}$$

where J_h and J are the control objective and contact wrench Jacobians respectively, B and D are the torque distribution and inertia matrices and q is the generalized coordinates. $\mathscr{I}$ captures the interaction of contact wrenches and motor torques in achieving contact constraints and control objectives [95].

5.6.1 Method and Results

Given that the contact interaction matrix governs the relation between the contact constraints, desired tasks (represented as virtual constraints for us), contact wrenches, and motor torques, we hypothesize that if the robot starts to lose balance, the matrix will be ill-conditioned. If this is the case, the contact interaction matrix can be utilized to detect faults. To test this hypothesis, we employ an algorithm similar to the nearest-neighbor classification algorithm proposed in Section 5.3. Instead of

Table 5.7: Results of the contact interaction matrix-based fall prediction algorithm with just the abrupt fault.

	Training Data		Validation Data		Testing Data	
Fold	Average Lead Time	False Positive Rate	Average Lead Time	False Positive Rate	Average Lead Time	False Positive Rate
1	0.42	0	0.48	0	0.37	0
2	0.43	0	0.43	0	0.38	0
3	0.40	0	0.51	0	0.40	0
4	0.44	0	0.34	0	0.46	0
5	0.41	0	0.37	0	0.49	0
Average	0.42	0	0.43	0.0	0.42	0

Table 5.8: Results of the contact interaction matrix-based fall prediction algorithm with just the incipient Fault.

	Training Data		Validation Data		Testing Data	
Fold	Average Lead Time	False Positive Rate	Average Lead Time	False Positive Rate	Average Lead Time	False Positive Rate
1	0.84	0	0.90	0	0.87	0
2	0.94	0	0.87	0	0.86	0
3	0.85	0	0.84	0	0.89	0
4	0.91	0	0.92	0.03	0.90	0
5	0.85	0	0.86	0	0.88	0
Average	0.88	0	0.88	0.01	0.88	0

using the Ward minimum variance method to calculate the threshold, we use the first derivative of the conditioning number derived from the contact interaction matrix. The same data used in Section 5.3 are used here. We, however, have not used windows. The preliminary results for the abrupt and incipient faults displayed in Table 5.7 and 5.8 respectively show the promise of this method. Even though the false positive rates are not zero for all the folds in the validation data for the incipient fault, the threshold can be adjusted to result in a zero false positive rate for the validation data. However, adjusting the threshold can result in a lower lead time.

Table 5.9: The average lead times achieved by the binary contact-interaction-based and SVM-based fall prediction algorithms

Fault Type	SVM Average Lead Time	Contact-Interaction Average Lead Time
Abrupt	0.48 (s)	0.42 (s)
Incipient	0.97 (s)	0.86 (s)

Next, we compare the performance of the proposed physics-based model on incipient and abrupt test data to the SVM binary classification algorithm introduced in Chapter 5.3. The results are summarized in Table 5.9. For abrupt faults, both algorithms achieve similar results. However, the SVM algorithm results in 0.09s higher lead time for incipient faults. Despite being outperformed, the contact-interaction-based algorithm is viable as it achieves lead times higher than 0.2s.

5.7 Conclusion

The objective of this chapter was to design a fall prediction algorithm for bipedal robots that is capable of detecting all three critical faults while maximizing the lead time and meeting the desired false positive and negative rates. To meet the desired upper bound on the false positive and negative rates, we proposed using training lead time, a subset of lead time, to label the windows in a trajectory. We successfully implemented a nearest-neighbor fall prediction classification algorithm and analyzed and compared its performance to an SVM classification-based algorithm. Using false positive and negative rates and average lead time as metrics, we found that the nearest-neighbor classification algorithm's performance is comparable to the SVM classifier when trained on abrupt and incipient faults separately. However, it detects falls on average 0.15s (results to 5 data points given our sampling time) slower than the SVM classifier when the faults are trained together. Given that the nearest-neighbor classification algorithm still has an average lead time of 0.5s, we conclude that if a sufficient amount of faulty data is unavailable, the nearest-neighbor classification algorithm can be used to detect abrupt and incipient faults simultaneously.

Even though the SVM classification algorithm outperforms the nearest-neighbor classification algorithm, its leading time when trained on both faults together is slightly lower than its average lead time from both faults separately. As a result, we investigate the use of a multi-class classification algorithm to reduce this difference. We find that using the same features with the multi-class classification algorithm slightly increases the average lead time. We briefly investigate using the multi-class classification algorithm with different features for the incipient and abrupt faults, and conclude that the multi-class classification algorithm shows promising results. However, more investigation is needed in feature selection and reduction in the delay time of the fault identifier to truly assess the advantage of using a multi-class classification algorithm.

In addition to successfully detecting critical abrupt and incipient faults, the proposed multi-classification algorithm can also detect critical intermittent faults. However, given that the resulting lead time and false positive rates for the SVM multi-classification algorithm are greatly impacted by user-selected features, we briefly explore the use of a 1D CNN-based algorithm for fall prediction. We also present a physics-based fall prediction algorithm and compare it with the binary classification SVM algorithm.

Chapter 6

Offline Fall Prediction for Digit: The Standing Phase

6.1 Introduction

In the previous chapter, we successfully developed a multi-classification fall prediction algorithm that was capable of detecting critical abrupt, incipient, and intermittent faults for the planar four-link robot. Therefore, we now progress to the development of a fall prediction algorithm for the bipedal robot Digit during the task of standing. The proposed fall prediction algorithm will be evaluated offline in simulation and hardware. The core work presented in this chapter has been previously published in [142] and submitted for publication to ICRA 2024. The co-authors of [142] are Gokul Prabhakaran and Jessy W. Grizzle.

6.1.1 Literature Review

Existing fall prediction algorithms for bipedal robots predominantly target the detection of critical abrupt faults. The primary method consists of establishing a threshold that distinguishes these faults from regular data. Selecting a threshold is a delicate task due to the impracticality of accounting for every potential critical fault and the complexity of capturing (e.g., in a model) all the robot's safe states. Additionally, the presence of faulty states in the data is infrequent.

The overarching aim of fall prediction algorithms is to extend the lead time while curtailing the false positive rate. Consequently, their performance is typically assessed

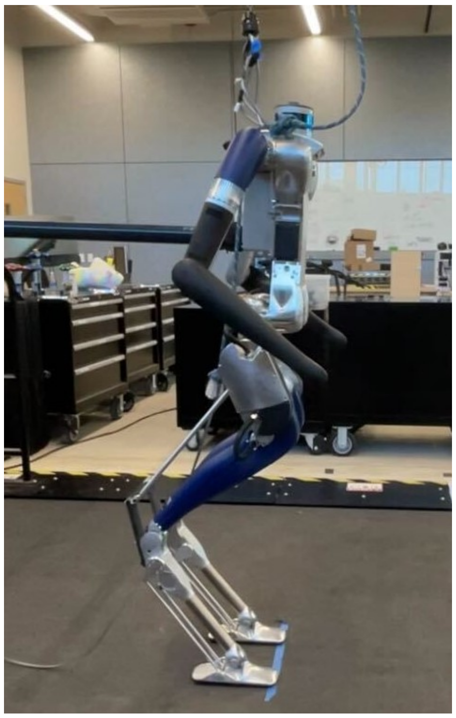


Figure 6.1: The experimental setup that was used to collect hardware data with the Digit robot [2].

based on these two metrics [143]. On the one hand, it's important to note that there's an inherent positive correlation between lead time and false positive rate. On the other hand, the impact of false negatives can be alleviated by setting thresholds on kinematic signals, such as the height of the center of mass, as elaborated in [8]. The minimum lead time deemed acceptable varies depending on the specific robot and the selected recovery algorithm.

In the realm of fall prediction, thresholds are derived from various sources: analytical models like [40, 42, 43, 144, 145], hand-crafted features as seen in [35, 38, 44, 45, 146], or data-driven models such as [8–10, 36, 40, 46–50]. While analytical models offer a structured approach, they can be hampered by model uncertainties and might not fully encapsulate the robot's full dynamics, especially if based on simplifying assumptions. Simple thresholds are elusive for multifaceted systems like bipedal robots. On the other hand, while data-driven models might necessitate extensive data and remain constrained to the data's distribution, their popularity is on the rise, thanks to advancements in machine learning and computational capabilities. Both shallow methods [8–10, 36, 46–48] and deeper neural network-based approaches [49, 50] have found their place in the literature.

6.1.2 Objective of the Chapter

Our goals are as follows:

1. Detect imminent falls caused by abrupt, incipient, and intermittent faults, ensuring an adequate lead time for the standing task.
2. Optimize the trade-off between maximizing lead time and minimizing false positive rates.
3. Accurately estimate the lead time.

The robot is deemed to have fallen if the height of its center of mass descends below 0.12m. Drawing parallels with our prior research [132], we adopt 0.2s as the minimum desired lead time, aligning with the requirements of reflexive algorithms such as those in [9, 34]. Achieving this objective presents challenges, given the controller's masking effects, the crowding phenomenon induced by incipient faults [134], the direct correlation between false positive rates and lead time, the sporadic nature of intermittent faults, and the diminishing number of data points with increasing lead time.

To address these challenges, we propose a 1D convolutional neural network (CNN)-driven fall prediction algorithm, enhanced with several components. Our choice of a deep model is motivated by the significant impact of user-selected features on lead time and false positive rates, as evidenced in our earlier work [132]. Furthermore, extracting these features for complicated bipedal robots, like Digit, is non-trivial. Our choice of a 1D CNN is underpinned by its equivariance to translation and proficiency in discerning local patterns in data with a grid structure. It is noteworthy that a bipedal robot's trajectory can be mapped onto a 1D grid, sampled at consistent (uniform) intervals [140].

6.1.3 Contributions

We present several key contributions in this chapter:

- Introduction of an algorithm capable of detecting abrupt, incipient, and intermittent faults in full-sized robots undertaking a standing task.
- Successful implementation of our fall prediction algorithm, both in simulation and on hardware, tailored for a full-sized humanoid robot.
- Development of a robust method to estimate lead time.

6.2 Data Generation

This section describes our approach to generating data for the Digit robot. The dataset, which includes both simulated trials and hardware tests, can be found at `https://github.com/UMich-BipedLab/Digit_Fall_Prediction_Dataset`.

6.2.1 Simulation Data Generation

We employ Agility's MuJoCo-based simulator in conjunction with a standing controller. This controller is designed to maintain both the center of mass and the zero moment point within the support polygon [3]. We generate 900 trajectories each for abrupt and incipient faults, and 100 trajectories for intermittent faults. These faults are simulated by applying forces of various magnitudes to the robot's torso in the x-direction (i.e., sagittal plane).

Abrupt faults are simulated using impulsive forces with a duration of 0.075s, randomly uniformly distributed within a range of 0 - 414.8N. In the case of incipient faults, their crowding effect is captured through trapezoidal force profiles [147] as

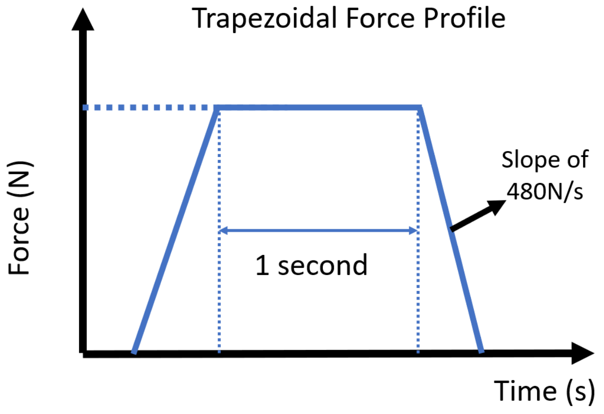


Figure 6.2: The trapezoidal force profile that is used to introduce incipient faults.

depicted in Figure 6.2. These profiles have a slope of $\frac{480N}{s}$ over a varying duration to result in a desired constant amplitude over a time duration of 1s; the resulting force amplitudes of incipient faults are randomly uniformly distributed between 0 - 57.6N. The force ranges for both abrupt and incipient faults are calibrated to ensure an equal distribution of falling and safe trajectories. Emulating the unpredictable nature of intermittent faults, we apply two distinct forces. These forces are designed to mimic either abrupt or incipient faults. The first force's magnitude remains within the safe range, while the second force's magnitude can potentially lead to a fall or maintain stability.

To simulate minor disturbances that might induce slight oscillations in the robot's standing posture, we introduce impulsive forces with a 0.075s duration, ranging from 0 - 202.4N, at the start of each trajectory. However, only data recorded one second after the introduction of this perturbative force is retained [1]. The abrupt and incipient faults are subsequently introduced between 2 - 3.5s following this oscillatory perturbation for all three faults. The time between the application periods for the two faults comprising the intermittent fault is 2s.

[1]Note that only data collected after one second is retained in order to ensure that the robot oscillates slightly even in the presence of perturbing forces towards the high end of the range. The choice of one second diverges from the two-seconds selected in Chapter 5, due to the utilization of a different controller and robot platform.

6.2.2 Hardware Data Generation

To prevent the Digit robot from getting damaged during data collection, the hardware data generation is carried out with Digit attached to a gantry via a slack cable that allows Digit to move about. Additionally, the motor power is "killed" when the robot starts to fall, thereby allowing the gantry to catch it. Impulsive and trapezoidal forces are introduced to the robot's torso by pushing Digit with a pole as depicted in Figure 6.1. To emulate the trapezoidal forces that result in an incipient fault, the pole is first rested on Digit before pushing. The approximate time of force application is obtained by coordinating the push on the robot with a keyboard press.

Twenty-seven (27) safe and 13 unsafe trajectories are collected for abrupt faults, while 26 safe and 15 unsafe trajectories are collected for incipient faults. Out of the abrupt and incipient unsafe trajectories, six and ten trajectories, respectively, exhibited trajectory profiles resembling those observed in simulations characterized by lower falling forces. The remaining falling trajectories exhibited similarities to simulation profiles featuring forces within the mid-range of the falling forces. Figure 6.1 depicts the experimental setup of the hardware data.

6.2.3 Data Pre-processing

From the 1,800 simulation-generated trajectories for abrupt and incipient faults, 200 are reserved for testing. The remaining trajectories are divided into training (80%) and validation sets. The testing set is further supplemented with intermittent fault data and hardware data. We employ scikit-learn's stratified train-test split method for segmenting the simulation data and its min-max scaler to normalize the data within the range $[0, 1]$. It's important to note that only transient data is utilized during training. The initial feature values are subtracted from subsequent data points to adjust for any drift in the hardware data. For both hardware and simulation, the features are transformed to the world coordinate which is placed on the ground between the feet. Lastly,

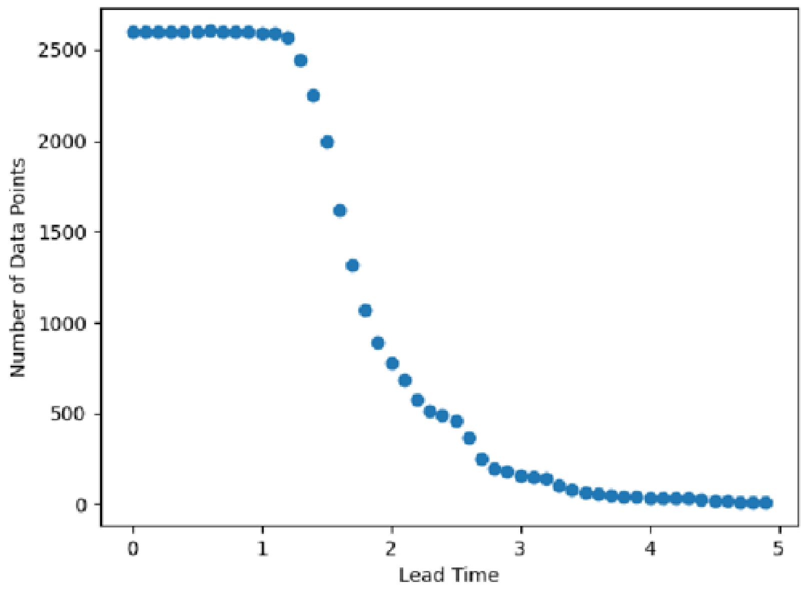


Figure 6.3: The number of data points vs. lead time.

sliding windows are utilized so as to prioritize the most recent data points.

6.3 Fall Prediction Method

The task of detecting critical faults and estimating lead time can be framed as a regression problem where lead time serves as the predicted variable. Given our definitions of lead time and critical faults, data points from unsafe trajectories prior to a critical fault's onset, along with all data points from safe trajectories, are assigned an infinite lead time. Data points following the introduction of a critical fault have a lead time within the range $[0, H]$, where $H > 0$ represents the **maximum prediction horizon** (i.e, maximum interval of time over which fault prediction is attempted). Predicting a lead time less than H can thus indicate the presence of critical faults. However, as illustrated in Figure 6.3, the quantity of data points diminishes exponentially with increasing lead time, leading to an imbalanced regression problem [148–151].

While techniques like the Synthetic Minority Over-Sampling Technique for Regression with Gaussian Noise [151] exist for addressing imbalanced regression data, the direct

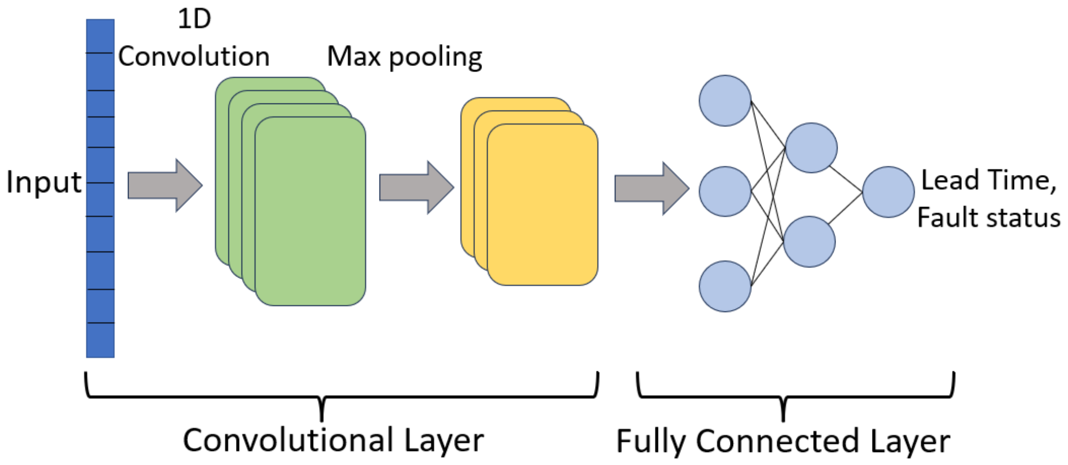


Figure 6.4: 1D CNN architecture of the fault detection network. It consists of a single CNN layer with a max pooling layer followed by 2 fully connected layers.

correlation between lead time and false positive rate complicates achieving maximum lead time with an acceptable false positive rate, as evidenced by studies like [8,50]. **Consequently, we reframe the problem into a combined classification and regression challenge**. Our proposed algorithm consists of three main components: a critical fault classifier, a lead time classifier, and a lead time regressor. All three components share a 1D CNN architecture, featuring a 1D CNN layer, a max pooling layer, and two fully connected layers with the ReLu activation function. The 1D CNN architecture is depicted in Figure 6.4.

The critical fault classifier's objective is to predict critical faults while maximizing lead time and minimizing false positives. The lead time classifier, on the other hand, categorizes windows containing critical faults into three distinct ranges: $[0,1]$, $(1,2]$, and $(2,H]$. Notably, the $(2,H]$ range contains significantly fewer data points, as shown in Figure 6.3. Finally, the lead time regressor predicts the lead time for windows that have a critical fault and a lead time within the range $[0,1]$.

The components of the algorithm interact sequentially. Initially, the critical fault classifier processes a window of data points from the robot. If a critical fault is detected, this window is then relayed to the lead time classifier, which categorizes

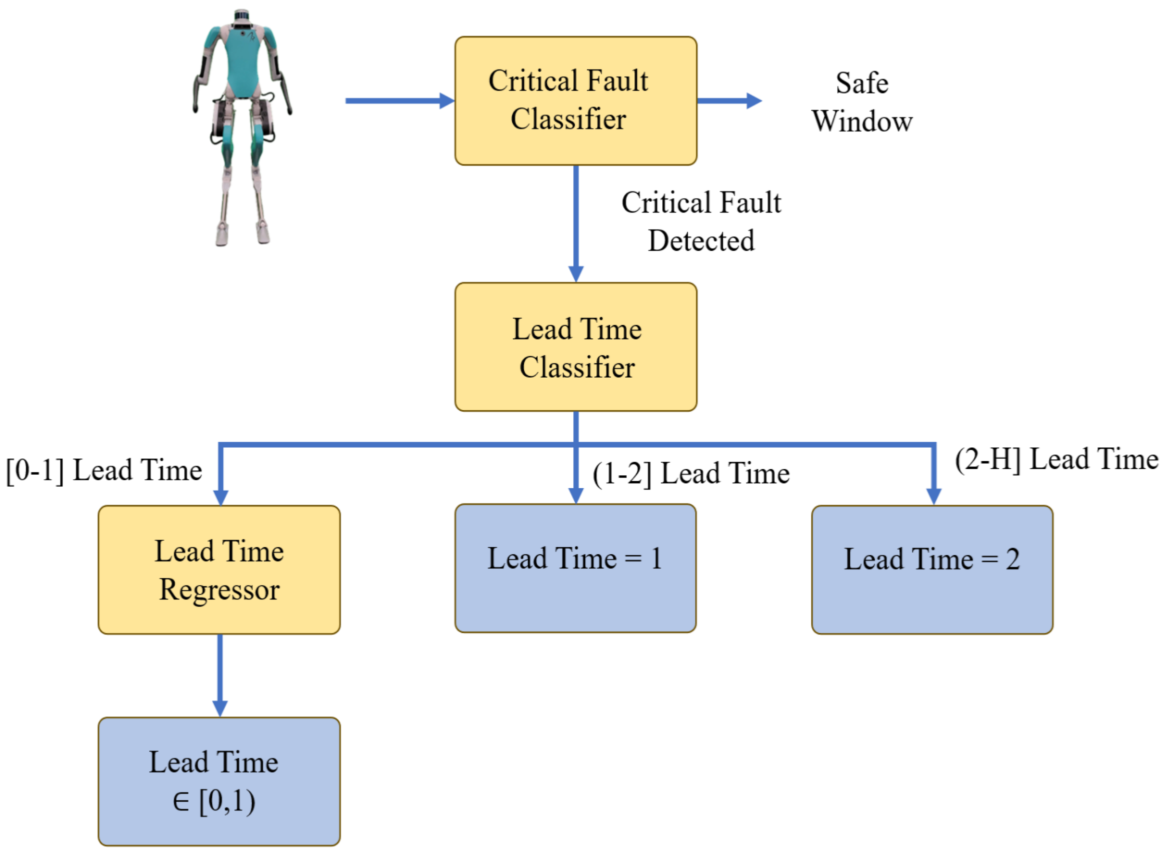


Figure 6.5: The proposed fall prediction algorithm with three components: critical fault classifier, lead time classifier and lead time regressor. The blue boxes depict the output of the entire algorithm.

the window based on the predefined lead time intervals. If the categorized lead time is within the $[0,1]$ interval, the lead time regressor determines the exact predicted lead time. For other intervals, the infimum lead time corresponding to that interval is reported (e.g, 1 for the interval $(1,2]$). The entire workflow of the proposed fall prediction algorithm is depicted in Figure 6.5.

6.4 Critical Fault Classifier

The features for the critical fault classifier are defined as,

$$\begin{bmatrix} (p_{com} - p_{midtoe})^{xz} \\ v_{com}^{xz} \\ q_{\text{sag chosen}} \\ \dot{q}_{\text{sag chosen}} \end{bmatrix} \tag{6.1}$$

where p_{com}, v_{com}, and p_{midtoe} represent the position and velocity of the center of mass and the midpoint of the two toes, respectively. The terms $q_{\text{sag chosen}}$ and $\dot{q}_{\text{sag chosen}}$ denote the torso, knee, hip, and toe pitch angles (in the sagittal plane).

The data for abrupt and incipient faults are structured as,

$$D = \{X_i, y_i\}_{i=1}^{n},$$

where

$$\begin{aligned} n &= \text{number of windows across all training data} \\ m &= \text{number of time steps in a window} \\ x_{ij} &= \text{features at time step j in window i} \\ t_i &= \text{time at time step i} \\ t_{\text{ft}} &= \text{time the fault is introduced} \\ \mathbf{T}_{safe} &= \text{safe trajectories} \\ \mathbf{T}_{unsafe} &= \text{unsafe trajectories} \\ X_i &= \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{im} \end{bmatrix}^{\mathsf{T}} \end{aligned}$$

$$y_i \in \begin{cases} 1 & \left(X_i \in \mathbf{T}_{unsafe} \wedge t_i \geq t_{\text{ft}}\right) \\ 0 & \left(\begin{array}{c} X_i \in \mathbf{T}_{safe} \\ \vee \\ (X_i \in \mathbf{T}_{unsafe} \wedge t_i < t_{\text{ft}}) \end{array}\right) \end{cases}.$$

With the above labeling, achieving correct identification for all windows would yield the maximum possible lead time. For intermittent data, the labeling approach remains similar, but t_{ft} is the time of the first fault's introduction.

The binary cross-entropy loss is employed for training. The classifier's output, when combined with this loss, produces logits. These logits can be transformed into probabilities using the sigmoid function, allowing for flexibility in setting the desired probability threshold for critical fault detection. For instance, a model with a 0.03 false positive rate and a 1.7s lead time for abrupt and incipient faults can achieve a 0 false positive rate with a 1.61s lead time by adjusting the probability threshold from 0.5 to 0.9.

The model's performance is evaluated at each epoch to optimize lead time and minimize false positive rates. A model is saved only if it meets one of the following criteria:

1. A reduction in the false positive rate of validation trajectories.
2. No change in the validation false positive rate, but an increase in validation lead time, accompanied by a decrease in the training data's false positive rate.
3. Both the validation and training false positive rates meet a predefined maximum threshold.

It's worth noting that the false positive rates used in the saving criteria pertain to entire trajectories, not individual windows.

6.4.1 Training

We train on a subset of abrupt and incipient simulation trajectories as described in Section 6.2.3. We use false positive rate, lead, and response time as our evaluation criteria. Response time is defined as the difference in time between the detection of the critical fault and its introduction. Given that the robot was not allowed to fall to the ground during hardware data collection, lead time is not defined for hardware. Similarly, given that intermittent data has 2 disturbances, it is difficult to estimate the response time.

6.4.2 Results

We evaluate on hardware trajectories, as well as on the intermittent and remaining incipient and abrupt trajectories, as defined in Section 6.2.3. The critical fault classifier is able to achieve 0 false positive rate for training and validation data when trained for 4 epochs with 8 filters for the 1D CNN. When evaluated on testing data, the model is able to achieve 0 false positive and negative rates for hardware, intermittent, abrupt, and incipient fault data.

The resulting average lead times are 1.61s and 1.52s for intermittent and abrupt plus incipient data, respectively. The classifier's success in categorizing the intermittent data can be attributed to the sliding window formulation, which allows the algorithm to recognize local patterns. The resulting response time is 0.42s and 1.0s for the abrupt and incipient simulation data, and hardware data, respectively. The 0.58s difference in response time between the hardware and simulation data can be attributed to the lower force profiles applied in the hardware. Furthermore, by applying lower force profiles in simulation, it was observed that the robot's behavior and response times were similar to the ones observed in hardware data. This was characterized by the robot's slight oscillatory motion prior to falling. Given that the critical fault classifier detects incipient, abrupt, and intermittent faults with a lead time greater than 0.2s, it meets our objective of detecting critical faults with sufficient lead time. The results are summarized in Table 6.1.

Table 6.1: Results of the critical fault classifier when trained on abrupt and incipient simulation data and evaluated on (a) abrupt and incipient simulation data, (b) abrupt and incipient hardware data, and (c) intermittent data. Trained for 4 epochs with 8 filters.

Platform	**Fault Type**	**Lead Time (s)**	**Response Time (s)**	**False Positive Rate**
Simulation	Intermittent	1.61	N/A	0.0
Simulation	Abrupt and Incipient	1.52	0.42	0.0
Hardware	Abrupt and Incipient	N/A	1.0	0.0

6.5 Lead Time Prediction

While the hardware data lacks a defined lead time, the similarity in response time between the hardware and simulation data, as indicated in 6.1, suggests that the hardware may have a lead time comparable to that of the simulation data. Nevertheless, there is a critical need for lead time prediction.

Given the absence of a defined lead time for the hardware data, this section evaluates the lead time algorithms using only simulation data. The subsequent section will apply the complete fall prediction algorithm to both hardware and simulation datasets.

The feature set for lead time prediction includes those from (6.1), complemented by the hip, knee, and toe pitch torques, as well as the average position of the contact point[2].

6.5.1 Lead Time Classifier

In this section, we detail our approach to categorizing lead times for potential falls. The categorization of lead times into three distinct intervals, $[0,1]$, $(1,2]$, and $(2,H]$, is treated as a multi-classification challenge. We employ the cross entropy loss for

[2]The contact point position defaults to the rotation point when the toes rotate and to the zero moment point otherwise.

this purpose.

The method of data labeling is analogous to the one adopted for the critical fault classifier. However, in this context, y_i is adjusted to indicate the specific lead time range to which a window is associated. A noteworthy aspect of our data is the exponential decline in the number of data points as lead time increases; recall Fig. 6.3. This implies that the amount of data available for the interval $(2, H]$ is significantly less than for the other intervals. As a result, achieving a high classification accuracy for this range is not our primary objective, but rather, an anticipated challenge.

After training on abrupt and incipient simulation faults using 8 filters, evaluation yields:

- The classifier achieves an accuracy of 1.0 for the interval $[0, 1]$;
- For the interval $(1, 2]$, the accuracy stands at 0.95; and
- The interval $(2, H]$ sees a lower accuracy of 0.80, in line with our expectations.

6.5.2 Lead Time Regressor

In this section, we delve into the methodology and results associated with our lead time regressor. The regressor is specifically trained on abrupt and incipient simulation data that fall within the range of $[0, 1]$. For the loss function, we employ the mean squared error, which is particularly effective for regression problems as it emphasizes larger errors over smaller ones.

Upon evaluation:

- The maximum difference between the predicted and actual lead times is 0.09s.
- The mean difference stands at 0.01s.
- The median difference is also 0.01s.

Given the minimal prediction error, it's evident that the lead time regressor performs well at predicting lead times within the specified range of $[0,1]$.

6.6 Overall Fall Prediction Method Results

In this section, we assess the performance of the fall prediction algorithm depicted in Figure 6.5 using both hardware and simulation data. It should be noted that the lead times and false positive rates presented in Table 6.1 relate to the entire fall prediction algorithm, not only the critical fault classifier. Therefore, when the critical fault classifier identifies a critical fault within the intervals of $(1,2]$ and $(2,H]$, the actual lead time is taken as 1 and 2, respectively. This is done in order to obtain a conservative approximation of the true lead time. The predicted lead time from the bin classifier is derived similarly. When assessed on simulation data, the fall prediction algorithm predicted an average lead time of 1.18s, with a slight difference of only 0.04s from the actual lead time of 1.14s.

Next, we evaluate the algorithm on the hardware data and the simulation data trimmed at 0.95m, which is the minimum center of mass height Digit achieved during experiments. The resulting lead time is 1.07s and 1.16s for the hardware and simulation data, respectively. Given that the fall prediction algorithm outputs similar lead times for the simulation data at 0.95m and the original fall height of 0.12m, we can conclude that for the simulation data, the fall prediction algorithm identifies critical faults at around 0.95m. In comparison to the simulation data trimmed at 0.95m and the actual lead time for simulation, the predicted lead time for the hardware data only differs by 0.09s and 0.07s, respectively. This minute difference demonstrates the algorithm's success in identifying and predicting lead time for both hardware and simulation data. Table 6.2 summarizes the results of the fall prediction algorithm.

Table 6.2: Results of the entire fall prediction algorithm when trained on abrupt and incipient simulation data and evaluated on (a) abrupt and incipient simulation data, and (b) abrupt and incipient hardware data.

Platform	**Fault Type**	**Fall Height (m)**	**Predicted Lead Time (s)**	**Actual Lead Time (s)**
Simulation	Abrupt and Incipient	0.12	1.18	1.14
Simulation	Abrupt and Incipient	0.95	1.16	N/A
Hardware	Abrupt and Incipient	0.95	1.07	N/A

6.7 Conclusion

In conclusion, this chapter aimed to develop an effective fall prediction algorithm for the bipedal robot Digit, considering abrupt, incipient, and intermittent faults while accurately predicting lead time. While a regression approach faces challenges due to data imbalance, we propose a comprehensive algorithm comprising a critical fault classifier, lead time classifier, and lead time regressor.

Our evaluation, based on simulation and hardware data, demonstrates the effectiveness of each component. The critical fault classifier achieves a 0 false positive rate, detecting faults with a minimum lead time of 1.52s and a maximum response time of 0.42s for simulation data. When evaluated on hardware data, a 0.5s time discrepancy in lead time was noted compared to simulation. This variance is attributed to differing force profiles in hardware. The lead time classifier exhibits an accuracy of up to 1.0 for data-rich ranges, and the lead time regressor accurately predicts lead times within a 0.09s difference in simulation data.

Assessing the entire algorithm, we observed a negligible 0.04s difference between predicted and actual lead times in simulation. In hardware evaluation, where true lead times were unknown, only a 0.07s difference in lead time was noted when compared to simulation. Given our algorithm's performance across diverse datasets and the successful prediction of fall events in bipedal robots, we can conclude that our objective has been achieved. Future work involves implementing this algorithm online on Digit.

Chapter 7

Online Fall Prediction for Digit: The Standing Phase

7.1 Introduction

The previous chapter introduced a 1D CNN-based algorithm to predict falls in the bipedal robot Digit while standing. This algorithm consisted of three parts: a critical fault classifier, a lead time classifier, and a lead time regressor. The critical fault classifier identifies faults related to falls, while the lead time classifier and regressor work together to predict the lead time. The lead time classifier divides the lead time into three intervals: $[0,1]$, $(1,2]$, and $(2,H]$. If the lead time is classified into the $[0,1]$ interval, the lead time regressor is used to calculate the lead time. Otherwise, the lead time is considered as the infimum of the interval. Lead time is determined this way due to the exponential decline of data points with respect to the lead time, as shown in Figure 6.3, and as a way to obtain a conservative approximation.

To validate the fall prediction algorithm, we introduced a new dataset on the humanoid platform Digit by Agility Robotics. This dataset is comprised of both simulated trials and hardware tests and can be accessed at `https://github.com/UMich-BipedLab/Digit_Fall_Prediction_Dataset`. Our proposed algorithm successfully detected all three critical faults (incipient, abrupt, and intermittent) offline while maximizing lead time, minimizing false positive rates, and predicting lead time. In other words, our algorithm can notify us when we are outside the region of attraction of the nominal controller.

In this chapter, we continue the analysis of the algorithm by assessing its performance online on simulated trials and hardware tests. The results show that the algorithm successfully differentiates between critical and non-critical faults online, both in hardware and simulation, thereby demonstrating that the assumptions made during the design and training of the proposed fall prediction algorithm, such as the simulation data closely emulating the hardware data (the algorithm is only trained on simulation data), are accurate. The same data pre-processing steps and fault descriptions as detailed in Section 6.2 are followed.

7.2 Simulation

Building upon Chapter 6, we employ the use of Agility's MuJoCo-based simulator with a standing controller designed to maintain the center of mass and zero moment point inside the support polygon [3]. We will refer to this controller as the nominal controller. To demonstrate the effectiveness of our fall prediction algorithm when run online, we run tests showcasing the Digit robot falling when a fall is predicted, but a recovery controller is not activated. As in Chapter 6, the robot is considered to have fallen when the center of mass is below 0.12m. We also show how the robot stabilizes once the recovery controller is triggered for each of the three faults. We have chosen to use Agility's base controller as the recovery controller as implementing a recovery controller is outside the scope of our work and because Agility's base controller already has a fall prevention strategy embedded in it. As a reminder, the nominal controller can stabilize incipient faults up to a force of 28.8N and abrupt faults up to a force of 207.4N. This was discussed in more detail in Section 6.2.

7.2.1 Results

When an abrupt fault of 250.0N is applied to the robot, the fall prediction algorithm detects the critical abrupt fault with a response time and lead time of 0.03s

Table 7.1: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with intermittent faults. The first fault is introduced at 9.5s while the second fault is introduced at 11.5s.

First Fault Type	Second Fault Type	Predicted Lead Time	Actual Lead Time
Abrupt 150 N	Abrupt 250 N	2 (s)	1.9 (s)
Abrupt 150 N	Incipient 35 N	1 (s)	1.18 (s)
Incipient 25 N	Abrupt 250 N	2 (s)	1.93 (s)
Incipient 25 N	Incipient 35 N	1 (s)	1 (s)
Average		1.5 (s)	1.5 (s)

and 2.06s, respectively. The predicted lead time is 2s. The algorithm's response time and lead time for a critical incipient fault of 30.0N are 1.64s and 1.65s, respectively, while the predicted lead time is 1s. Figure 7.1 and 7.2 show how switching to the recovery controller from the nominal controller helps the robot regain stability when the critical abrupt and incipient faults are detected, respectively.

For intermittent faults, we run four tests for each fault pairing: (1) abrupt followed by abrupt, (2) abrupt followed by incipient, (3) incipient followed by abrupt, and (4) incipient followed by incipient. As outlined in Section 6.2, the first fault is selected such that it remains within the safe range. The fall prediction algorithm detects critical faults with an average lead time of 1.5s; the average predicted lead time is also 1.5s. See Table 7.1 for details. As with the abrupt and incipient faults, when the fall prediction algorithm predicts a fall for all four intermittent tests, the robot regains stability upon activating the recovery controller, while it falls otherwise. Figures 7.3, 7.4, 7.5, and7.6 illustrate this phenomenon.

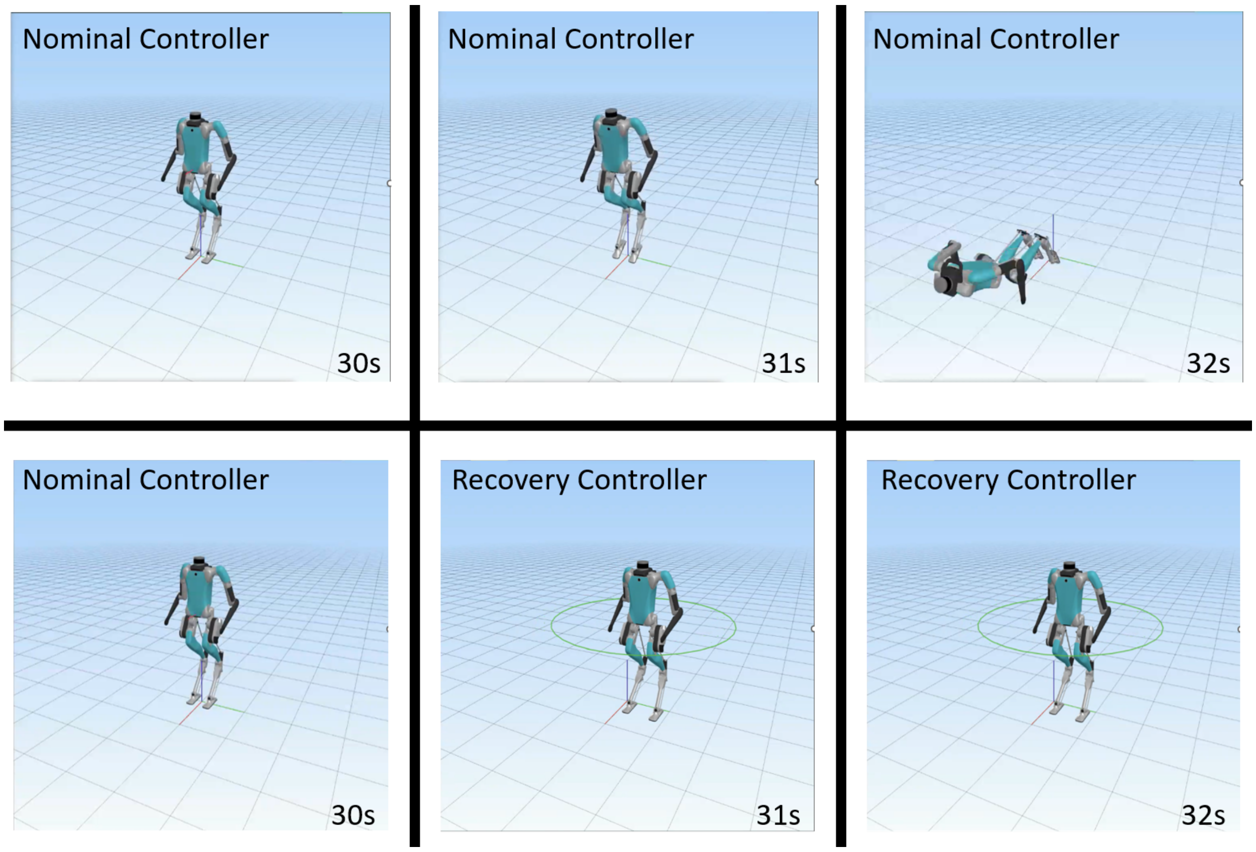


Figure 7.1: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with a critical abrupt fault of 250N. The critical fault is introduced at 30s.

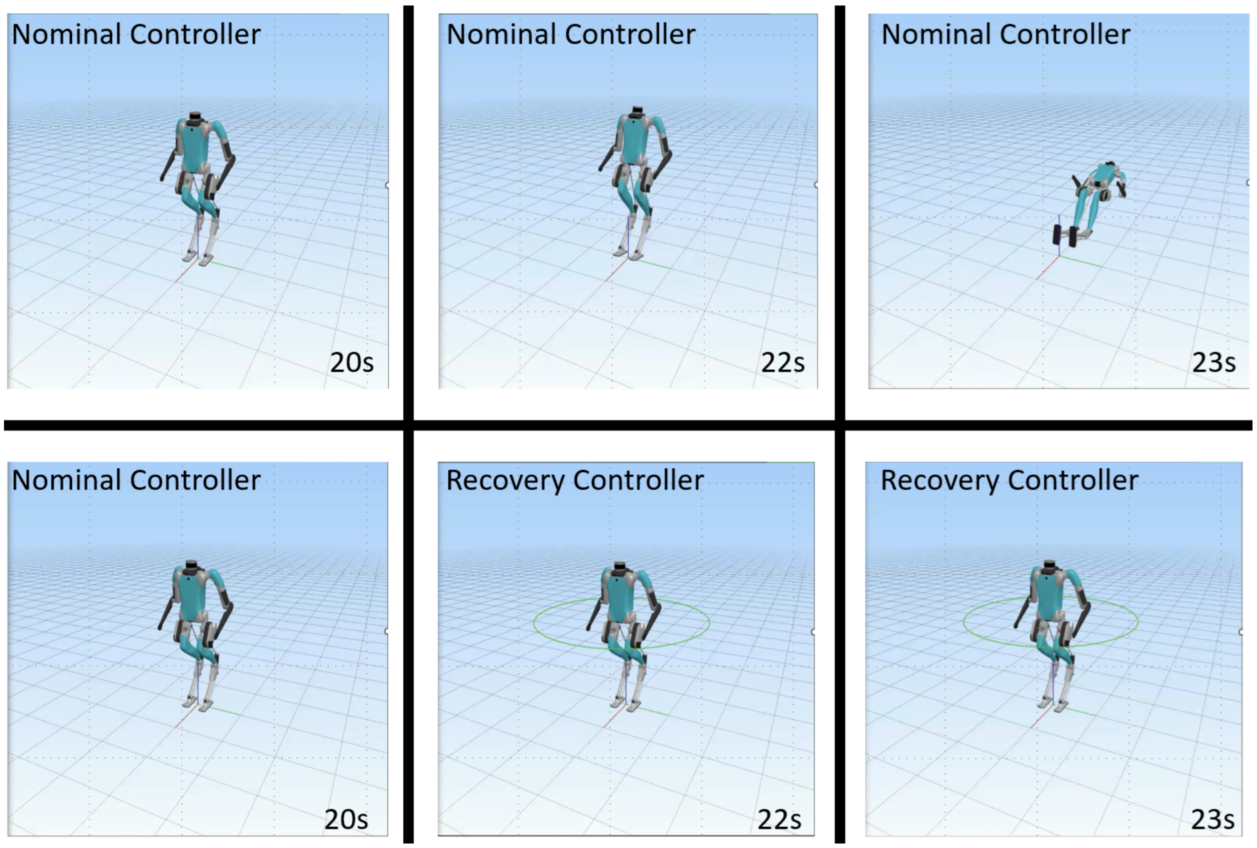


Figure 7.2: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with a critical incipient fault of 30N. The critical fault is introduced at 20s.

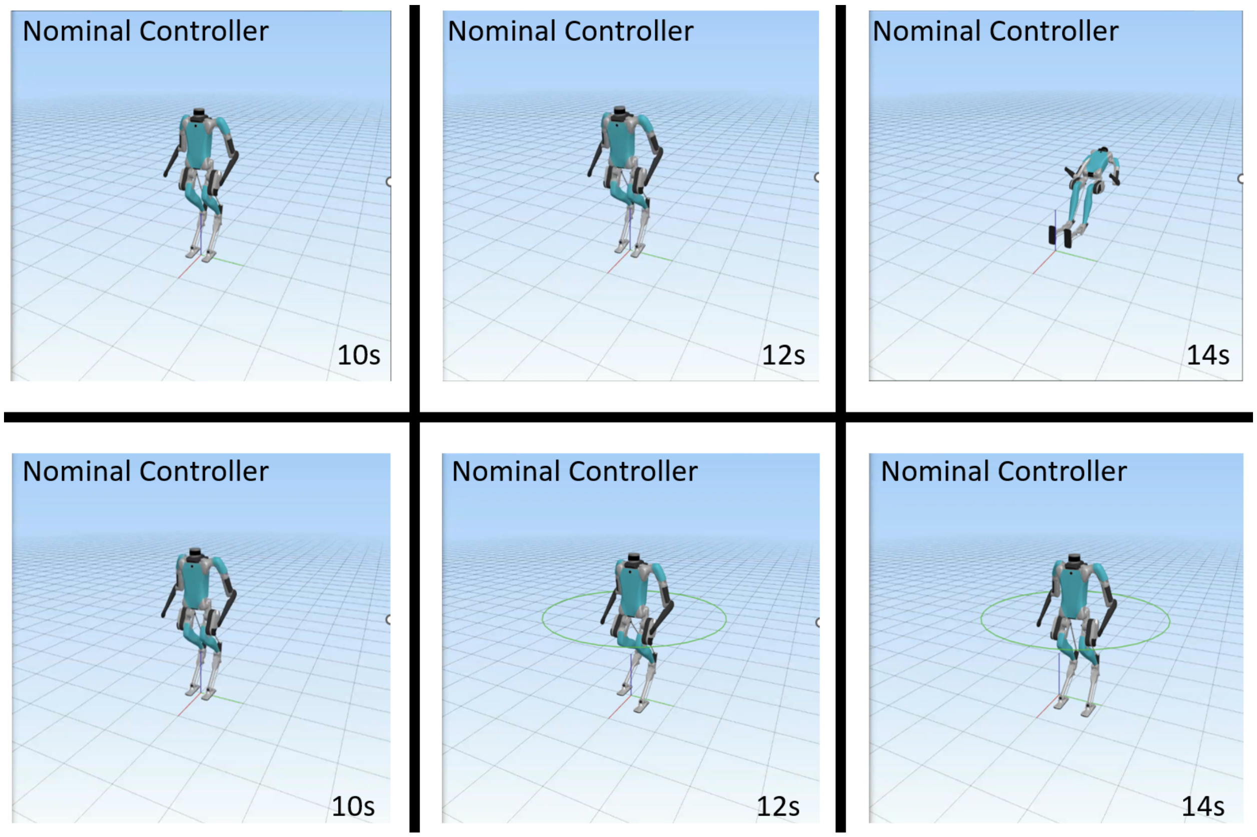


Figure 7.3: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with an intermittent fault comprised of an abrupt fault of 150N followed by a critical abrupt fault of 250N. The first fault is introduced at 9s and the second at 11.5s.

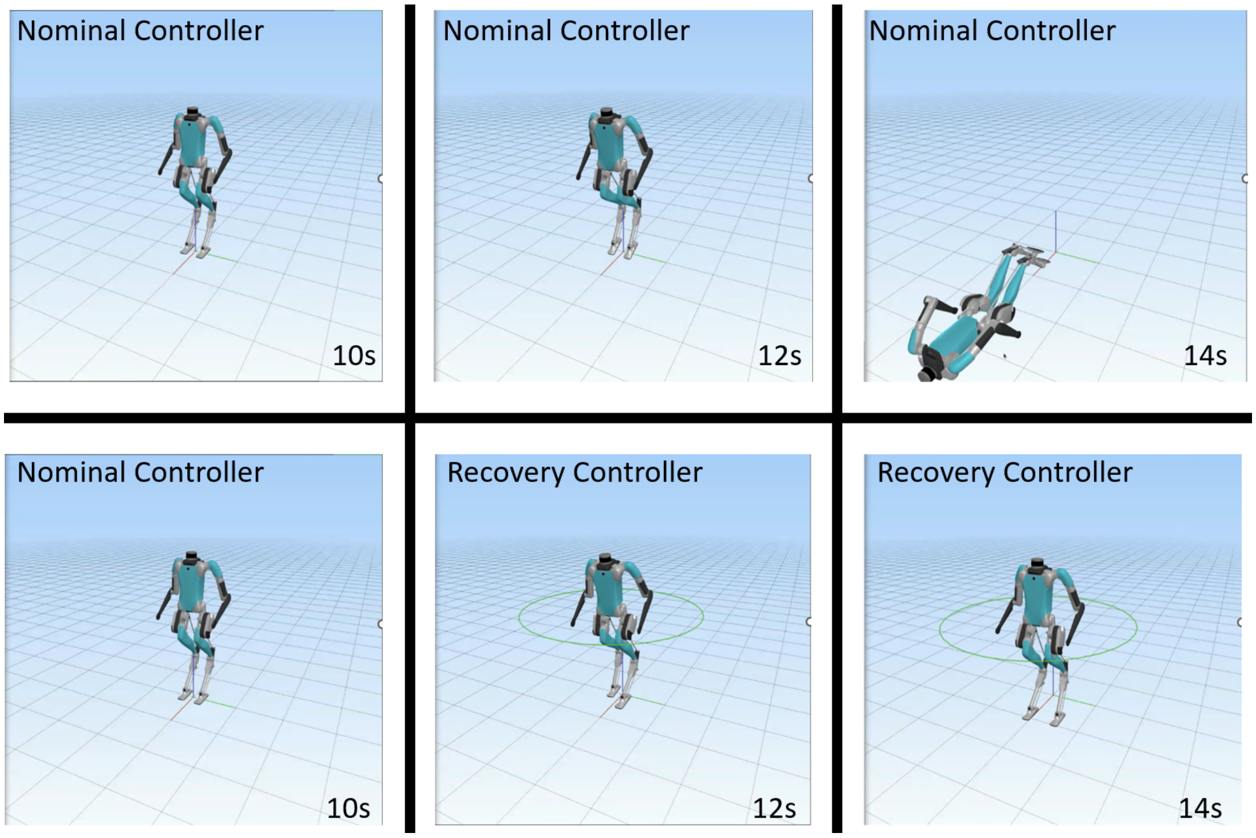


Figure 7.4: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with an intermittent fault comprised of an abrupt fault of 150N followed by a critical incipient fault of 35N. The first fault is introduced at 9s and the second at 11.5s.

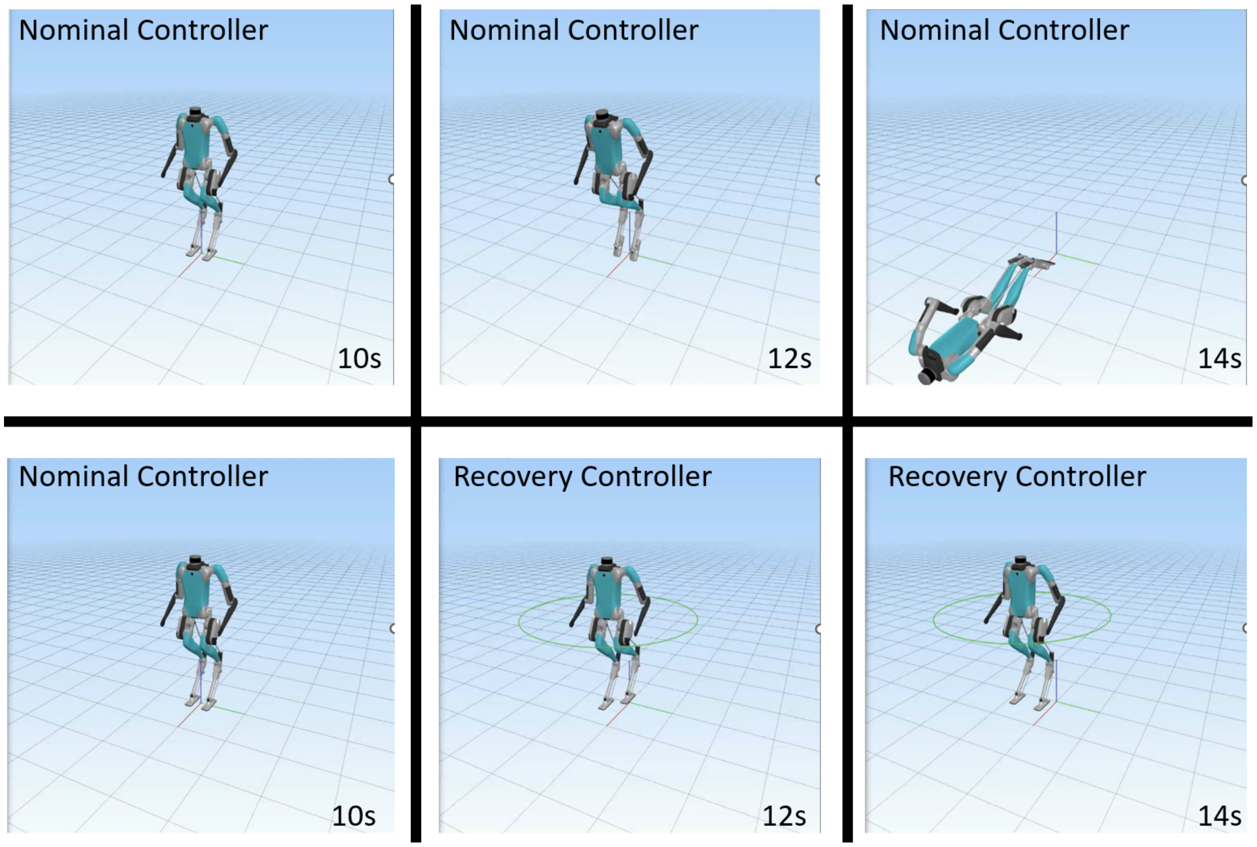


Figure 7.5: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with an intermittent fault comprised of an incipient fault of 25N followed by a critical abrupt fault of 250N. The first fault is introduced at 9s and the second at 11.5s.

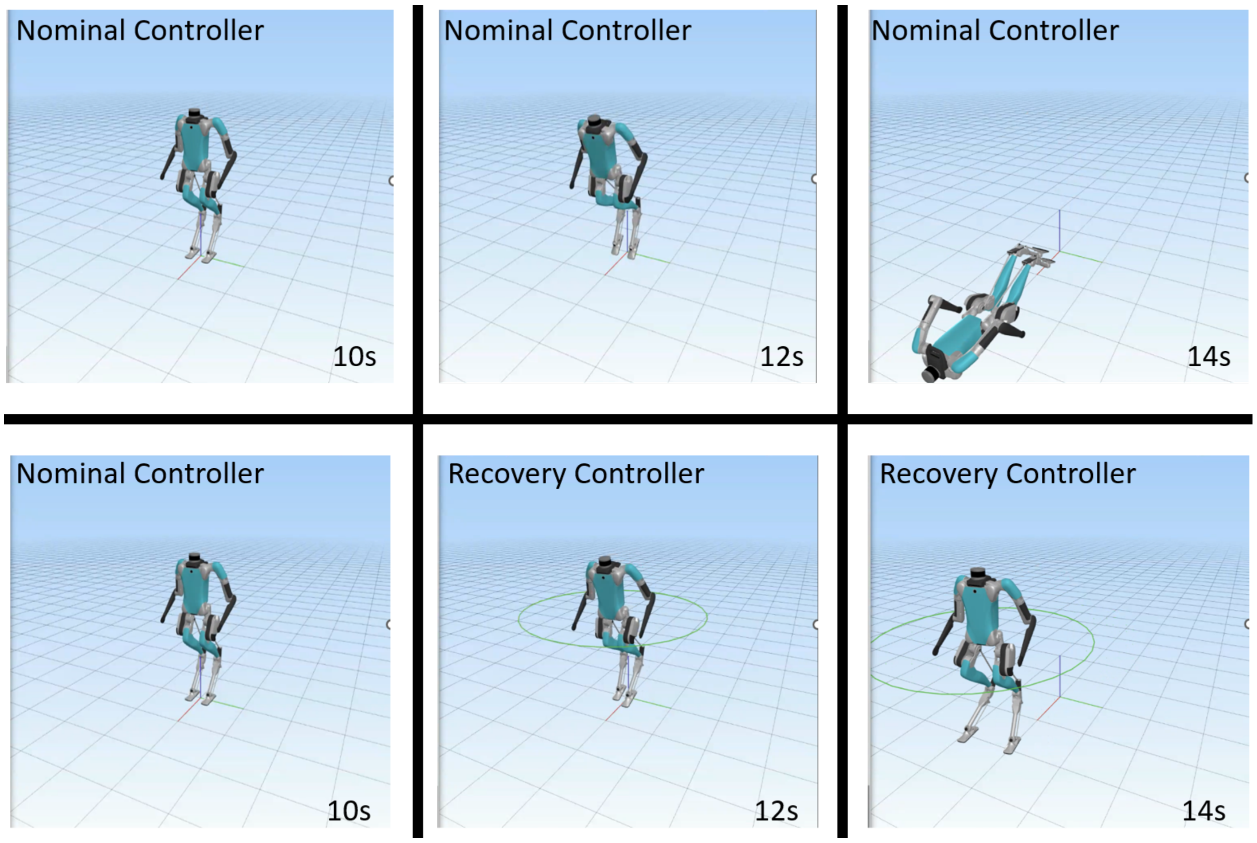


Figure 7.6: The 1D-CNN-based fall prediction algorithm's results when evaluated online in simulation with an intermittent fault comprised of an incipient fault of 25N followed by a critical incipient fault of 35N. The first fault is introduced at 9s and the second at 11.5s.

7.3 Hardware

The same experimental setup described in Section 6.2.2 is utilized here. We demonstrate the efficacy of our algorithm running online in hardware by showcasing its ability to switch from the nominal controller to the recovery controller when a critical fault is detected and to remain in the nominal controller when a non-critical fault is introduced. Due to the complexity of replicating the same fault twice in hardware, we refrain from executing tests that demonstrate the Digit robot falling when a fall is predicted, but the recovery controller isn't activated, as was done in simulation. However, given our experience collecting data for Chapter 6, we emulate the low and medium force profiles inside and outside the region of attraction of the nominal controller for abrupt and incipient faults. As intermittent faults are comprised of abrupt and incipient faults and our algorithm uses windows, the algorithm's capability to accurately detect critical abrupt and incipient faults enables it to detect critical intermittent faults as discussed in Chapter 6. Therefore, in line with Chapter 6, we limit the online hardware tests to abrupt and incipient faults.

7.3.1 Results

From the hardware tests, we find that the fall prediction algorithm is capable of distinguishing critical faults from non-critical faults with a predicted average lead time of 1.14s. Note that this only has a 0.07s difference compared to the offline results in Chapter 6. Figures 7.7, 7.8, 7.9, and 7.10 display the algorithm's ability to switch to the recovery controller only when a critical fault is detected. To emphasize the switch between controllers, we display the cases where the recovery controller had to take a step in order to stabilize the robot. In addition to successfully detecting critical faults introduced by pushing the robot's torso horizontally, the algorithm also detects critical faults introduced by pushing the robot in various ways, such as the angled

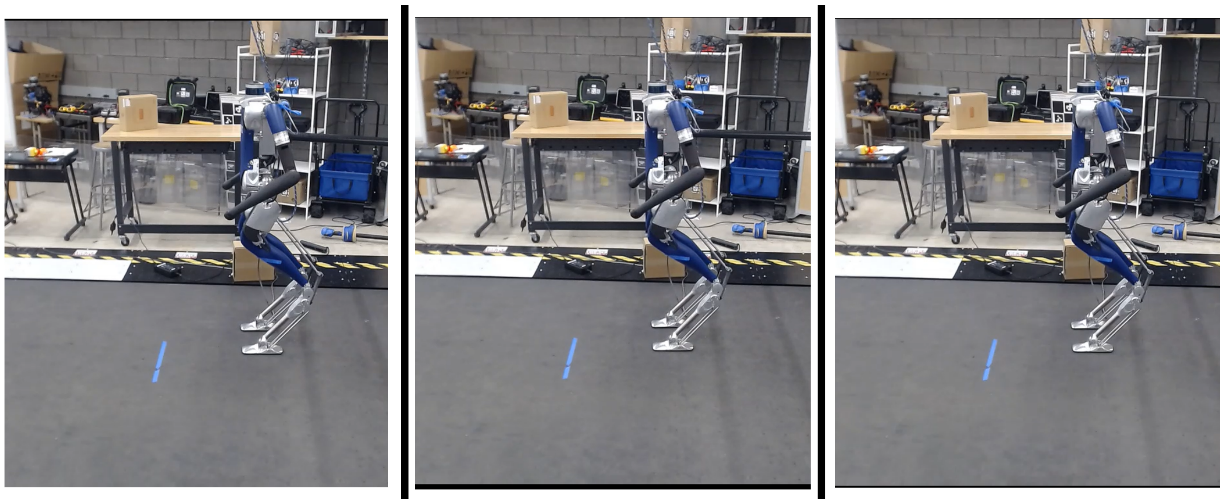


Figure 7.7: The 1D-CNN-based fall prediction algorithm correctly identifies a non-critical incipient fault.

push displayed in Figure 7.11 and the leg push displayed in Figure 7.12. Note that the algorithm is only trained on non-angled torso pushes, and the recovery controller does not take a step for the critical fault introduced by a leg push.

7.4 Conclusion

In this chapter, an evaluation of the 1D-CNN-based fall prediction algorithm's online performance was conducted through simulated trials and hardware tests, complementing the offline evaluations done in Chapter 6. Using Agility's MuJoCo-based simulator, the standing controller from [3] as the nominal controller, and Agility's base controller as the recovery controller, the simulation trials demonstrated the algorithm's ability to detect all three critical faults. Furthermore, they demonstrated that transitioning from the nominal to the recovery controller was pivotal in stabilizing the robot once a critical fault was detected. The online hardware experiments further demonstrated the algorithm's ability to differentiate critical and non-critical faults, with a predicted average lead time that closely mirrored offline results. The 1D-CNN-based fall prediction algorithm's success, both online and offline in hardware and simulation, at detecting critical faults and predicting lead time underscores the algorithm's ability to approximate the region

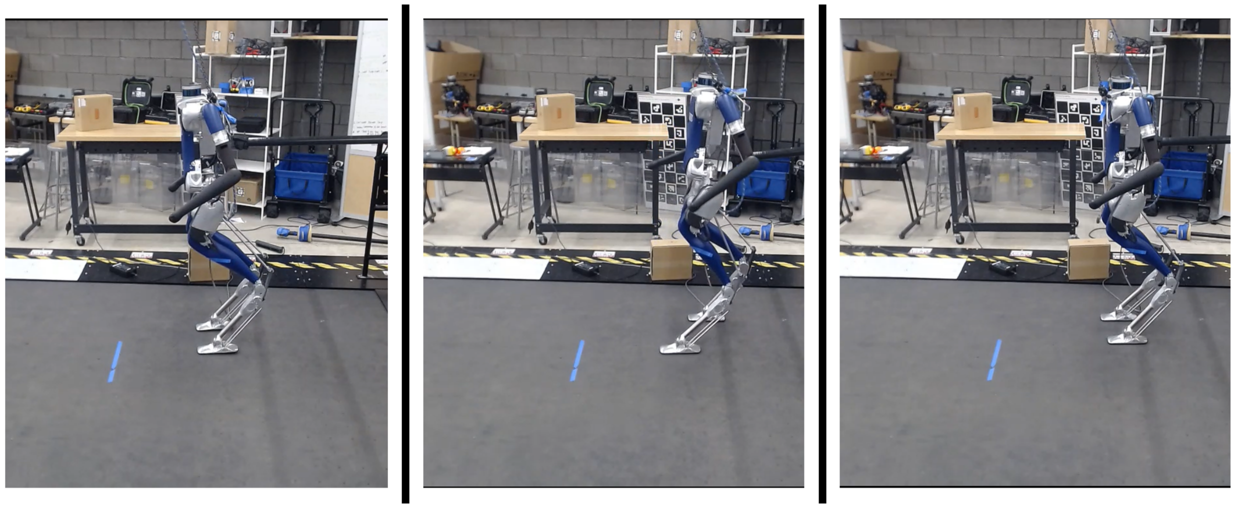


Figure 7.8: The 1D-CNN-based fall prediction algorithm identifies a critical incipient fault, thereby triggering a switch to the recovery controller.

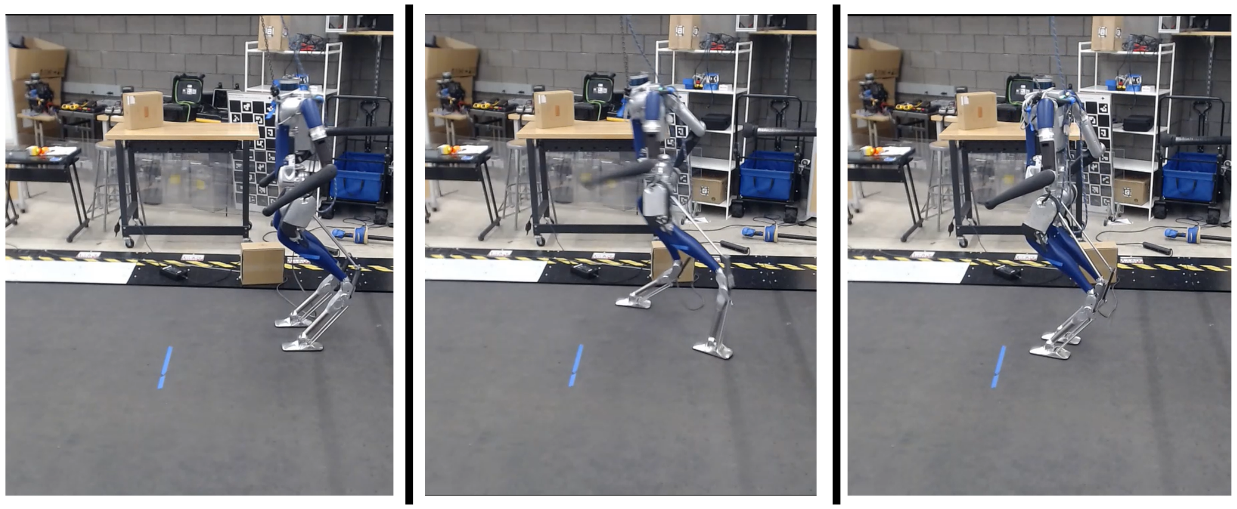


Figure 7.9: The 1D-CNN-based fall prediction algorithm identifies a critical abrupt fault, thereby triggering a switch to the recovery controller.

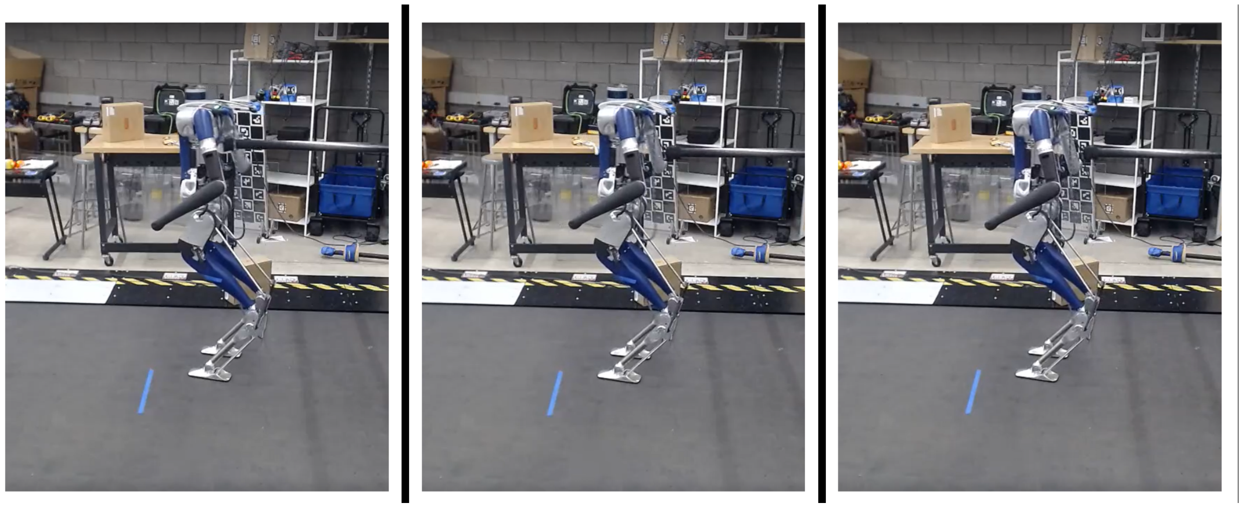


Figure 7.10: The 1D-CNN-based fall prediction algorithm correctly identifies a non-critical abrupt fault.

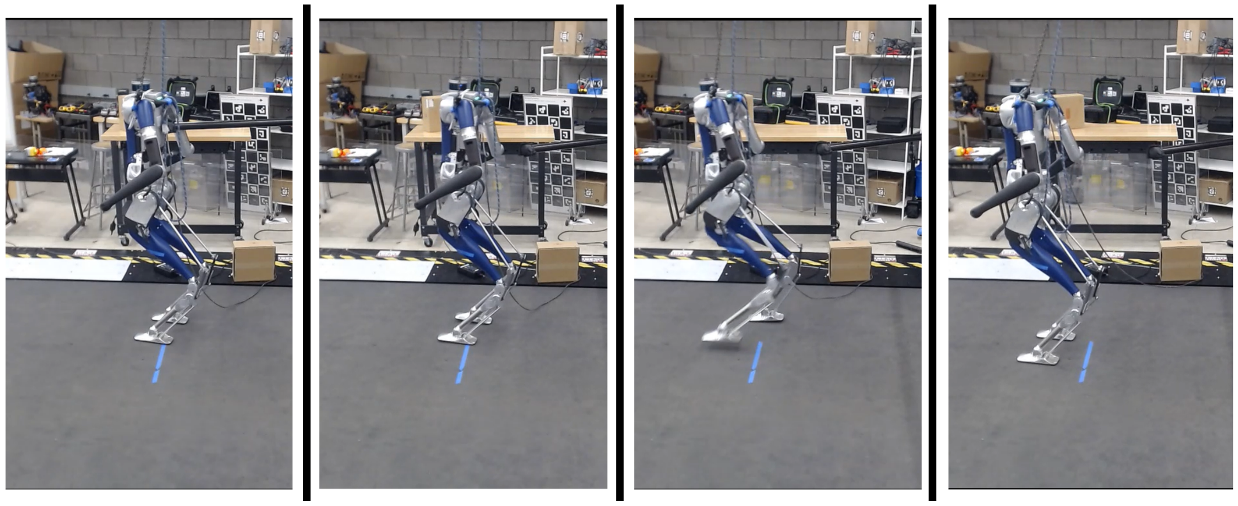


Figure 7.11: The 1D-CNN-based fall prediction algorithm identifies a critical incipient fault introduced by an angled push to the torso, thereby triggering a switch to the recovery controller.

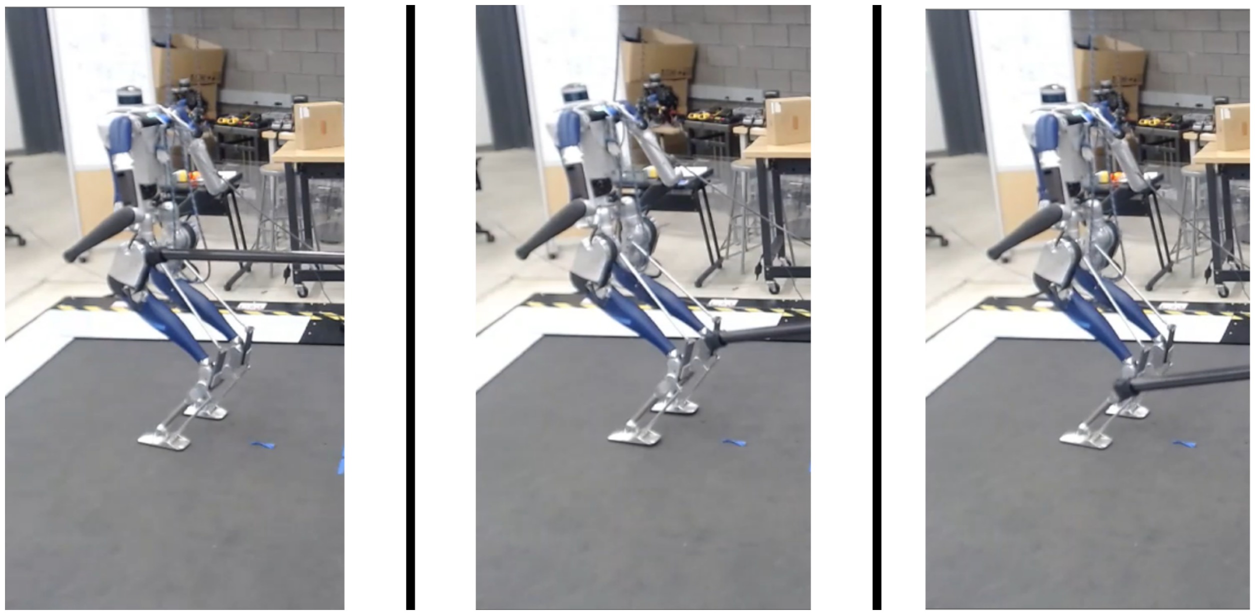


Figure 7.12: The 1D-CNN-based fall prediction algorithm identifies a critical incipient fault introduced by a leg push, thereby triggering a switch to the recovery controller. Note that the recovery controller does not take a step to recover.

of attraction of the nominal controller and makes it a necessity for any fall-tolerant framework of bipedal robots.

Chapter 8

Conclusion and Future Work

The capability of bipedal robots to seamlessly navigate in human environments and their ability to restore mobility demonstrates their potential to improve day to day lives. However, their highly dimensional, hybrid, and, at times, highly constrained nature makes it challenging to achieve stable motions, especially in the face of disturbances and uncertainties. Disturbances and uncertainties can lead to faults, which can be a precursor to a fall. As a result, the real-world deployment of bipedal robots is limited.

For bipedal robots to reach their potential and assist us, it's imperative to implement a fall-tolerant framework that encompasses various algorithms, such as robust control algorithms, reliable fall prediction algorithms, and recovery algorithms. Recall that a fall-tolerant framework aims to equip the robot with the capabilities to assess, adapt, and respond effectively to uncertainties and disturbances, thereby minimizing the risks and consequences of falling. This dissertation has focused on developing and implementing key components of such a framework, with contributions directed towards the enhancement of robust controllers and the creation of dependable fall prediction algorithms.

Chapters 3 and 4 contributed to the implementation of robust controllers by introducing a systematic method to design control objectives for highly constrained systems such that they are independent of the contact constraints. The effectiveness of the control objectives was demonstrated through robust, comfortable closed-loop sit-to-stand motions for the exoskeleton Atalante. Unfortunately, due to the robot breaking, we were unable to conduct hardware experiments. Consequently, a pivotal next step for this project is hardware implementation.

As explained in Chapter 3, our methodology is flexible enough to be applied to other motions involving multiple contact points and different types of exoskeletons or humanoids. Therefore, additional future work involves expanding our methodology to other robot platforms and different motions, such as manipulation or walking with a non-instantaneous double support domain.

In Chapters 5, 6, and 7, reliable fall prediction methods were developed and evaluated for a planar four-link robot and the bipedal robot Digit. These contributions included physics-based and data-driven algorithms, culminating in a 1D-CNN-based fall prediction model exhibiting high accuracy in simulation and hardware evaluations.

The findings from Chapter 5 underscored the importance of employing different features for abrupt and incipient faults within the SVM multi-class classification algorithm. Therefore, the next step for this work is further exploration into optimal feature selection. This exploration should aim to ascertain whether the additional average lead time gained from using different features for the faults can overcome fault identifier delay.

Chapters 6 and 7 demonstrated the effectiveness of the proposed 1D-CNN-based fall prediction algorithm in predicting critical faults for the bipedal robot Digit while standing. However, critical faults can occur in various ways during real-world operations, as mentioned in Section 2.5. Therefore, the next step for the 1D-CNN-based fall prediction algorithm is to extend its application to more diverse environmental conditions and dynamic motions such as walking.

The introduction of a physics-based method in Chapter 5 for detecting critical abrupt and incipient faults presents another opportunity for future work. As discussed in the chapter, data-based methods such as the 1D-CNN-based fall prediction algorithm introduced in Chapter 6 and the SVM multi-class classification algorithm introduced in

Chapter 5, can perform poorly when evaluated on out-of-distribution data. Therefore, it is imperative to extend the proposed physics-based methodology to encompass the detection of all three critical faults for a full-dimensional bipedal robot like Digit.

The projects undertaken in this dissertation focused on developing robust controllers and reliable fall prediction algorithms separately. Yet, considering the comprehensive fall prediction framework outlined in Chapter 1, which encompasses multiple algorithms including recovery controllers, future research directions include the integration of the proposed robust controller and fall prediction algorithm. Furthermore, expanding the implementation scope to include other algorithms, such as recovery controllers and high-level motion planners, stands as a pivotal endeavor to the full realization of a fall-tolerant framework.

In conclusion, the work presented in this dissertation has greatly contributed to the implementation of a fall-tolerant framework for bipedal robots. As a result, bipedal robots are now one step closer to achieving their full potential and assisting us in the real world.

Appendix A

Description of the Physical Constraints for Dynamic Feasibility and User Comfort

Here we present in full detail, the additional constraints that are implemented to achieve the desired motions. For clarity, the constraints are divided into two categories, boundary and path constraints. Boundary constraints are implemented only at the beginning and end of a domain, while path constraints are present throughout the entire domain.

A.1 Path Constraints

Anticipating that the feedback controller will need robustness margins, the ZMP is constrained to be within SP_{both_opt}, a strict subset of SP_{both}, for the entire chair-to-stand motion, and the sit and standing shift domains of chair-to-crouch-to-stand, as illustrated in Figure 3.4 and Figure 3.9. Similarly, during the standing extend domain, the ZMP is constrained to be within a smaller polygon, $SP_{feet\_opt} \subset SP_{feet}$. We note that there is no contact with the chair during the standing extend domain. Motor torque bounds, joint bounds, and a minimum distance between the feet are introduced to respect hardware bounds. The minimum distance between the feet prevents the exoskeleton from colliding with itself. The torque and state bounds are implemented in all domains while the distance between the feet constraint is only implemented in the sitting domain. However, the distance between the feet established in the sitting

Table A.1: Actuated joint limits used for optimization

Name	Lower Bound (deg)	Upper Bound (deg)
Henke Ankle Joint	-12.3	12.3
Sagittal Ankle Joint	-10.3	3.3
Sagittal Knee Joint	5.9	104.1
Sagitaal Hip Joint	-109.1	9.1
Transverse Hip Joint	-4.1	14.1
Frontal Hip Joint	-4.1	11.1

Table A.2: Motor torque limits used for optimization

Name	Maximum Torque (N)	Nominal Torque (N)
Henke Ankle Joint	90	82
Sagittal Ankle Joint	192	184
Sagittal Knee Joint	219	124
Sagitaal Hip Joint	219	124
Transverse Hip Joint	180	124
Frontal Hip Joint	350	198

domain is maintained in all domains due to the feet contact constraint. The state bounds for the actuated joints are set to be stricter than the hardware bounds to ensure user safety and comfort. These joint limits can be found in Table A.1.

Optimization is allowed to use the maximum torque value a motor can output for each actuated joint except the ankle, which is set to the smaller nominal value; see Table A.2. These design choices are possible because the real-time quadratic program used in the control implementation is effective at managing torque limits; see Section 3.6.

In order to generate realistic motions, the rate of change of the chair force component

Table A.3: Equality path constraints for both the chair-to-stand and chair-to-crouch-to-stand motions

Constraint		Chair-to-stand		Chair-to-crouch-to-stand		
Name	Constraint value	Sitting	Standing	Sitting	Standing shift	Standing extend
Equal knee angle (deg)	0	✓	✓	✓	✓	✓
User force (N)	0	×	×	×	×	✓

in the z direction ($\dot{F}^{z}_{chair}$) is bounded. The bound for $\dot{F}^{z}_{chair}$ is defined as follows:

$$max(|\dot{F}^{z}_{chair}|) \leq \frac{F^{z}_{chair_0}}{t^{sit}_{min}}$$

where t^{sit}_{min} is the minimum desired duration of the sitting domain and $F^{z}_{chair_0}$ is the initial vertical force exerted by the chair. To achieve a symmetric motion, a torque difference constraint between the right and left actuated joints is implemented only for chair-to-stand. An equal knee angle and negative knee angle velocity constraints are implemented to prevent the exoskeleton from swaying side to side and oscillating up and down. A minimum knee angle constraint is implemented in all domains for user comfort. As a design choice, the y component of the spatial user force is constrained to zero for the entire motion. The lower and upper bounds for the x and z components are set to zero and the total weight of the user (TW_{user}) respectively. The path constraints are summarized in Table A.3 and Table A.4.

Table A.4: Inequality path constraints for both the chair-to-stand and chair-to-crouch-to-stand motions

Constraint			Chair-to-stand		Chair-to-crouch-to-stand		
Name	Lower bound	Upper bound	Sitting	Standing	Sitting	Standing shift	Standing extend
$ZMP \in SP_{both\_opt}$	×	×	✓	✓	✓	✓	×
$ZMP \in SP_{feet\_opt}$	×	×	×	×	×	×	✓
Joint bounds	see Table A.1	see Table A.1	✓	✓	✓	✓	✓
Motor torque bounds	see Table A.2	see Table A.2	✓	✓	✓	✓	✓
Distance between Feet (m)	0.32	0.5	✓	×	✓	×	×
Minimum knee angle (deg)	15	104.1	✓	✓	✓	✓	✓
User force-spatial frame (N)	$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} \text{TW}_{\text{user}} \\ 0 \\ \text{TW}_{\text{user}} \end{bmatrix}$	✓	✓	✓	✓	×
Knee angle velocity $(\frac{deg}{s})$	-240	0	×	✓	×	✓	✓
Torque difference (N)	-1	1	✓	✓	×	×	×
$\dot{F}^{z}_{chair}$ $(\frac{kg}{s})$	chair-to-stand:-117.9 chair-to-crouch-to-stand:-90.7	chair-to-stand:117.9 chair-to-crouch-to-stand:90.7	✓	×	✓	×	×

A.2 Boundary Constraints

For the exoskeleton to start at the desired sitting pose, for both motions, the chair height is set by constraining the initial *sitting-point* height. In addition, an initial torso pitch and knee angle constraint are implemented. A final torso pitch, and knee angle constraint are implemented at the final domains of both motions. The chair is constrained to support at least 80% of the exo-system's total weight (TW) at the beginning of the sitting domain in both motions.

The chair-to-stand and chair-to-crouch-to-stand motions are both constrained to start and end in a statically stable position. The user force is set to zero at the beginning of the sitting domain for both motions, and at the end of the stand and standing shift domain for the chair-to-stand and chair-to-crouch-to-stand motions respectively. Therefore, to guarantee a stable static final pose for chair-to-stand, the ZMP needs to be within SP_{feet} at the end of the motion. However, since the ZMP and CoM are coincident when the exo-system is static, it is sufficient to constrain the final CoM to be within SP_{feet}; we constrain the final CoM to be in SP_{feet_opt}. Even though at the end of the standing shift domain the exo-system will have no contact with the chair (the user force is constrained to be zero), it is not necessary to explicitly constrain the ZMP or CoM to be inside SP_{feet}. This is because the event that triggers the transition from the standing shift domain to the standing extend domain, $ZMP \in SP_{feet}$, is set up such that it constrains the ZMP to be in SP_{feet_opt} at the end of the standing shift domain. To encourage the optimizer to find a chair-to-crouch-to-stand motion with the CoM near the middle of the feet at the end, the CoM is constrained to be in $SP_{feet\_SS} \subset SP_{feet\_opt}$. The boundary constraints are summarized in Table A.5 and Table A.6.

Table A.5: Equality boundary constraints for both the chair-to-stand and chair-to-crouch-to-stand motions

Constraint			Chair-to-stand		Chair-to-crouch-to-stand		
	Name	Constraint value	Sitting	Standing	Sitting	Standing shift	Standing extend
Initial	Exo-system velocity ($\frac{deg}{s}$)	0	✓	×	✓	×	×
	Exo-system acceleration ($\frac{deg}{s^2}$)	0	✓	×	✓	×	×
	User force (N)	0	✓	×	✓	×	×
	Torso pitch (deg)	0	✓	×	✓	×	×
Final	Exo-system velocity ($\frac{deg}{s}$)	0	×	✓	×	×	✓
	Exo-system acceleration ($\frac{deg}{s^2}$)	0	×	✓	×	×	✓
	User force (N)	0	×	✓	×	✓	×

Table A.6: Inequality boundary constraints for both the chair-to-stand and chair-to-crouch-to-stand motions

Constraint				Chair-to-stand		Chair-to-crouch-to-stand		
	Name	Lower bound	Upper bound	Sitting	Standing	Sitting	Standing shift	Standing extend
Initial	Knee angle (deg)	60	110	✓	×	✓	×	×
	Chair height (m)	0.5	0.6	✓	×	✓	×	×
	Chair support (kg)	0.8TW	2TW	✓	×	✓	×	×
Final	Knee angle (deg)	0	15	×	✓	×	×	✓
	Torso pitch (deg)	0	15	×	✓	×	×	✓
	$CoM \in SP_{feet\_opt}$	×	×	×	✓	×	×	×
	$CoM \in SP_{feet\_SS}$	×	×	×	×	×	×	✓

Appendix B

Enhancing Performance in Off-nominal Conditions

The domain definition for the chair-to-stand motion is modified so that it is compatible with the chair-to-crouch-to-stand motion, allowing a unified control architecture. The hybrid closed-loop system is represented by a directed acyclic graph with four domains as shown in Figure B.1: sitting, stand 1, stand 2, and stopping. For more information on the domains see Section B.1. The SU and SP controllers act in the sitting, stand 1, and stand 2; and stopping domains respectively.

The desired evolution of the virtual constraints for the SU controller, $h_d(t)$ are represented using a Bezier polynomial [53, 152] of 5 th degree for the sitting domain and 6 th degree for the stand 1 and stand 2 domains. The desired virtual constraints are time based and are parametrized using

$$t_{bez} = \frac{t - min(t)}{max(t) - min(t)} \tag{B.1}$$

In order to start at the beginning of the desired virtual constraint profile even with perturbations, (B.1) is modified such that $t_{bez} = 0$ at the beginning of all domains except the stand 2 domain during chair-to-stand; this will be discussed further in Section B.1. To handle a variety of off nominal conditions, the Bezier coefficients at the beginning of each domain are modified such that $y(t_0) = 0$ while still maintaining the same value for $\dot{y}(t_0)$. Additionally, if the number of iterations in the QP exceeds an allowed maximum number of iterations, u^* and ζ^* are replaced with the QP's output and the QP is rerun with the modified objective function. For each motion, the desired torque profile is the torque profile from optimization. It is passed to the controller as a spline and is generated using Matlab's curve fitting tool.

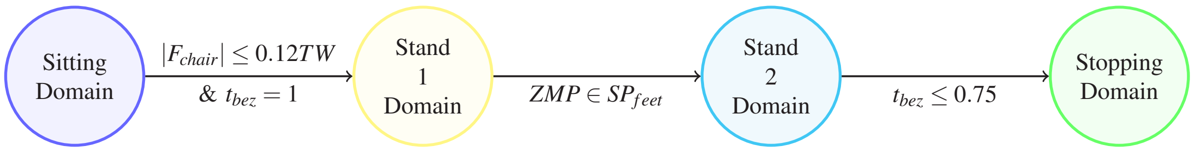


Figure B.1: hybrid system model for the two controllers

B.1 Control Domain Unification

The standing domain from the chair-to-stand optimization is split into three domains, in a similar manner to how the chair-to-crouch-to-stand hybrid model was obtained in Section 3.2.2, stand 1, stand 2, and stopping. This is done for three reasons: (a) to allow both the chair-to-stand and chair-to-crouch-to-stand motions to be addressed with the same simulation and control architecture, (b) to ensure the transition from stand 2 to stopping occurs after the ZMP is within the feet support polygon, and (c) to allow the implementation of additional constraints and modifications to the equations of motion towards the end of the motion that ensure the exo-system safely comes to a stop (see Section 3.6.2). As the stand 1, stand 2, standing shift, and standing extend domains are all governed by the same dynamic equations (the equations of motion during the standing domain), they are all equivalent domains. Therefore, the stand 1 and stand 2 domains can be thought of as the standing shift and standing extend domains respectively during chair-to-crouch-to-stand.

It is important to note that for chair-to-stand the SU controller tracks the same virtual constraints during the stand 1 and stand 2 domains. However, for chair-to-crouch-to-stand the SU controller tracks different virtual constraints. This difference is caused by the original domain definition of the chair-to-stand and chair-to-crouch-to-stand motions which result in 2 optimal virtual constraints for the chair-to-stand motion (for the sitting and standing domains) and 3 for the chair-to-crouch-to-stand motion (for the sitting, standing shift, and standing extend domains). As a result, for chair-to-crouch-to-stand the SU controller tracks the virtual constraints obtained from the standing shift and standing extend domains during the stand 1 and stand 2 domains respectively. On the other hand, for chair-to-stand the SU controller tracks the virtual constraint obtained from the Standing domain during both the stand 1 and stand 2 domains. In fact, t_{bez} is not reset to zero between the the stand 1 and stand 2 domains for the chair-to-stand motion. This allows the SU controller to seamlessly continue tracking the virtual constraint profile in the stand 2 domain.

The transitions among the various domains in Figure B.1 are highlighted here.

- **Sitting to stand 1:** The transition from the sitting domain to the stand 1 domain happens when the chair supports 25 percent or less of the total exo-system weight and $t_{bez} = 1$. In other words, the transition happens when the feet are supporting most of the exo-system weight and when the SU controller reaches the end of the desired virtual constraint trajectory. The chair forces can be estimated from the ground reaction forces or measured directly.

- **stand 1 to stand 2:** The transition from the stand 1 domain to stand 2 domain happens after the ZMP is within the feet support polygon.

- **Control Action in the Subsequent Domains after stand 1:** The ZMP is constrained by the controller to be within the feet support polygon irrespective of whether the user is in contact with the chair or not. In other words, the chair is ignored in the calculation of the support polygon.

- **stand 2 to Stand in Place:** The transition from the stand 2 domain to the stopping domain happens when $t_{bez} = 0.95$. This ensures that the exo-system is still in motion when the stopping domain begins.

- **Dynamics of stand 1 and stand 2:** Since for the chair-to-stand motion the stand 1 and stand 2 domains track the same virtual constraints, and are governed by the same dynamic equations, the transition to the stopping domain can be thought of occuring after the ZMP is within the feet support polygon (and) when $t_{bez} \leq 0.95$.

BIBLIOGRAPHY

[1] Wandercraft. Atalante. Accessed: April 2020.

[2] Agility Robotics. Robots. `https://agilityrobotics.com/robots`.

[3] Grant Gibson. *Terrain-Aware Bipedal Locomotion*. PhD thesis, University of Michigan, 2023.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Hugh Herr. Exoskeletons and orthoses: classification, design challenges and future directions. *Journal of neuroengineering and rehabilitation*, 6:1–9, 2009.

[6] R. Isermann. Process fault detection based on modeling and estimation methods. *IFAC Proceedings Volumes*, 15(4):7–30, 1982. 6th IFAC Symposium on Identification and System Parameter Estimation, Washington USA, 7-11 June.

[7] Reimund Renner and Sven Behnke. Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2967–2973. IEEE, 2006.

[8] Shivaram Kalyanakrishnan and Ambarish Goswami. Learning to predict humanoid fall. *International Journal of Humanoid Robotics*, 8(02):245–273, 2011.

[9] Tong Wu, Zhangguo Yu, Xuechao Chen, Chencheng Dong, Zhifa Gao, and Qiang Huang. Falling prediction based on machine learning for biped robots. *Journal of Intelligent & Robotic Systems*, 103:1–14, 2021.

[10] Hiromichi Suetani, Aiko M Ideta, and Jun Morimoto. Nonlinear structure of escape-times to falls for a passive dynamic walker on an irregular slope: Anomaly detection using multi-class support vector machine and latent state extraction by canonical correlation analysis. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2715–2722. IEEE, 2011.

[11] George Vavoulas, Matthew Pediaditis, Charikleia Chatzaki, Emmanouil G Spanakis, and Manolis Tsiknakis. The mobifall dataset: Fall detection and classification with a smartphone. *Int. J. Monit. Surveillance Technol. Res.*, 2:44–56, 2014.

[12] Lourdes Martínez-Villaseñor, Hiram Ponce, Jorge Brieva, Ernesto Moya-Albor, José Núñez-Martínez, and Carlos Peñafort-Asturiano. Up-fall detection dataset: A multimodal approach. *Sensors*, 19(9):1988, 2019.

[13] Carlos Menacho and Jhon Ordoñez. Fall detection based on cnn models implemented on a mobile robot. In *2020 17th international conference on ubiquitous robots (UR)*, pages 284–289. IEEE, 2020.

[14] Tong Zhang, Jue Wang, Liang Xu, and Ping Liu. Fall detection by wearable sensor and one-class svm algorithm. In *Intelligent Computing in Signal Processing and Pattern Recognition: International Conference on Intelligent Computing, ICIC 2006 Kunming, China, August 16–19, 2006*, pages 858–863. Springer, 2006.

[15] Xiaoqun Yu, Jaehyuk Jang, and Shuping Xiong. A large-scale open motion dataset (kfall) and benchmark algorithms for detecting pre-impact fall of the elderly using wearable inertial sensors. *Frontiers in Aging Neuroscience*, 13:692865, 2021.

[16] Janice J Eng, Stephen M Levins, Andrea F Townson, Dianna Mah-Jones, Joy Bremner, and Grant Huston. Use of prolonged standing for individuals with spinal cord injuries. *Physical therapy*, 81(8):1392–1399, 2001.

[17] Pouran D Faghri, John P Yount, William J Pesce, Subramani Seetharama, and John J Votto. Circulatory hypokinesis and functional electric stimulation during standing in persons with spinal cord injury. *Archives of physical medicine and rehabilitation*, 82(11):1587–1595, 2001.

[18] Robert B Dunn, James S Walter, Yuvone Lucero, Frances Weaver, Edwin Langbein, Linda Fehr, Paul Johnson, and Lisa Riedy. Follow-up assessment of standing mobility device users. *Assistive Technology*, 10(2):84–93, 1998.

[19] Melanie M Adams and Audrey L Hicks. Comparison of the effects of body-weight-supported treadmill training and tilt-table standing on spasticity in individuals with chronic spinal cord injury. *The journal of spinal cord medicine*, 34(5):488–494, 2011.

[20] O. Narvaez Aroche, P. J. Meyer, S. Tu, A. Packard, and M. Arcak. Robust control of the sit-to-stand movement for a powered lower limb orthosis. *IEEE Transactions on Control Systems Technology*, 28(6):2390–2403, 2020.

[21] Abbas Fattah, Sunil K. Agrawal, Glenn Catlin, and John Hamnett. Design of a Passive Gravity-Balanced Assistive Device for Sit-to-Stand Tasks. *Journal of Mechanical Design*, 128(5):1122–1129, 10 2005.

[22] V. Rajasekaran, M. Vinagre, and J. Aranda. Event-based control for sit-to-stand transition using a wearable exoskeleton. In *2017 International Conference on Rehabilitation Robotics (ICORR)*, pages 400–405, 2017.

[23] Ameya S Chamnikar, Gaurav Patil, Mohammadreza Radmanesh, and Manish Kumar. Trajectory generation for a lower limb exoskeleton for sit-to-stand transition using a genetic algorithm. In *ASME 2017 Dynamic Systems and Control Conference*, page V001T36A004. American Society of Mechanical Engineers Digital Collection, 2017.

[24] Kaveh Kamali, Ali Akbar Akbari, and Alireza Akbarzadeh. Trajectory generation and control of a knee exoskeleton based on dynamic movement primitives for sit-to-stand assistance. *Advanced Robotics*, 30(13):846–860, 2016.

[25] Weiguang Huo, S. Mohammed, Y. Amirat, and K. Kong. Active impedance control of a lower limb exoskeleton to assist sit-to-stand movement. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3530–3536, 2016.

[26] E.V. Velu. Design of a sit-to-stand controller for a lower limb exoskeleton and validation by feasible crutch usage identification, December 2018.

[27] S. Mefoued, S. Mohammed, Y. Amirat, and G. Fried. Sit-to-stand movement assistance using an actuated knee joint orthosis. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1753–1758, 2012.

[28] Ronnie Joseph Wong and James Andrew Smith. Regenerative effects in the sit-to-stand and stand-to-sit movement. *Robotica*, 33(1):107–126, 2015.

[29] Giuseppe Menga and Marco Ghirardi. Control of the sit-to-stand transfer of a biped robotic device for postural rehabilitation. *Robotics*, 8(4):91, 2019.

[30] Sergey Jatsun, Sergei Savin, Andrey Yatsun, and Ruslan Turlapov. Adaptive control system for exoskeleton performing sit-to-stand motion. In *2015 10th International Symposium on Mechatronics and Its Applications (ISMA)*, pages 1–6. IEEE, 2015.

[31] Gaurav Patil, Lillian M. Rigoli, Ameya Chamnikar, Amanda Miller, Anca Ralescu, Adam W. Kiefer, Michael J. Richardson, Tamara Lorenz, and Manish Kumar. Control Strategy for an Assistive Exoskeleton for Sit-to-Stand Transition. In *ASME 2017 Dynamic Systems and Control Conference*, page V001T30A005, 2017.

[32] M. H. Qureshi, Z. Masood, L. Rehman, M. Owais, and M. U. Khan. Biomechanical design and control of lower limb exoskeleton for sit-to-stand and stand-to-sit movements. In *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, pages 1–6, 2018.

[33] G Ya Panovko, SI Savin, SF Yatsun, and AS Yatsun. Simulation of exoskeleton sit-to-stand movement. *journal of Machinery Manufacture and Reliability*, 45(3):206–210, 2016.

[34] Oliver Höhn and Wilfried Gerth. Probabilistic balance monitoring for bipedal robots. *The International Journal of Robotics Research*, 28(2):245–256, 2009.

[35] Rajesh Subburaman, Dimitrios Kanoulas, Luca Muratore, Nikos G Tsagarakis, and Jinoh Lee. Human inspired fall prediction method for humanoid robots. *Robotics and Autonomous Systems*, 121:103257, 2019.

[36] Jeong-Jung Kim, Tae-Yong Choi, and Ju-Jang Lee. Falling avoidance of biped robot using state classification. In *2008 IEEE International Conference on Mechatronics and Automation*, pages 72–76, 2008.

[37] JA Gallego, A Forner-Cordero, JC Moreno, A Montellano, EA Turowska, and Jose L Pons. Continuous assessment of gait stability in limit cycle walkers. In *2010 3rd IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 734–739. IEEE, 2010.

[38] Javier Ruiz-del Solar, Javier Moya, and Isao Parra-Tsunekawa. Fall detection and management in biped humanoid robots. In *2010 IEEE International Conference on Robotics and Automation*, pages 3323–3328. IEEE, 2010.

[39] Ons Amri, Majdi Mansouri, Ayman Al-Khazraji, Hazem Nounou, Mohamed Nounou, and Ahmed Ben Hamida. Improved model based fault detection technique and application to humanoid robots. *Mechatronics*, 53:140–151, 2018.

[40] Thomas Muender and Thomas Röfer. Model-based fall detection and fall prevention for humanoid robots. In Hidehisa Akiyama, Oliver Obst, Claude Sammut, and Flavio Tonidandel, editors, *RoboCup 2017: Robot World Cup XXI*, pages 312–324, Cham, 2018. Springer International Publishing.

[41] X. Xinjilefu, Siyuan Feng, and Christopher G. Atkeson. Center of mass estimator for humanoids and its application in modelling error compensation, fall detection and prevention. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 67–73, 2015.

[42] Bassam Jalgha, Daniel Asmar, and Imad Elhajj. A hybrid ankle/hip preemptive falling scheme for humanoid robots. In *2011 IEEE International Conference on Robotics and Automation*, pages 1256–1262, 2011.

[43] Carlotta Mummolo, Luigi Mangialardi, and Joo H Kim. Numerical estimation of balanced and falling states for constrained legged systems. *Journal of Nonlinear Science*, 27:1291–1323, 2017.

[44] Junyun Tay, I-Ming Chen, and Manuela Veloso. Fall prediction for new sequences of motions. In *Experimental Robotics: The 14th International Symposium on Experimental Robotics*, pages 849–864. Springer, 2016.

[45] Duy Hoa Tran, Fred Hamker, and John Nassour. A humanoid robot learns to recover perturbation during swinging motion. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 50(10):3701–3712, 2020.

[46] Petar Kormushev, Barkan Ugurlu, Luca Colasanto, Nikolaos G Tsagarakis, and Darwin G Caldwell. The anatomy of a fall: Automated real-time analysis of raw force sensor data from bipedal walking robots and humans. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3706–3713. IEEE, 2012.

[47] JG Daniël Karssen and Martijn Wisse. Fall detection in walking robots by multi-way principal component analysis. *Robotica*, 27(2):249–257, 2009.

[48] Fernando Marcolino and Jiuguang Wang. Detecting anomalies in humanoid joint trajectories. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2594–2599. IEEE, 2013.

[49] João André, Brígida Mónica Faria, Cristina Santos, and Luís Paulo Reis. A data mining approach to predict falls in humanoid robot locomotion. In Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muñoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, pages 273–285, Cham, 2016. Springer International Publishing.

[50] Dongdong Liu, Hoon Jeong, Aoxue Wei, and Vikram Kapila. Bidirectional lstm-based network for fall prediction in a humanoid. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 129–135, 2020.

[51] Kevin M Lynch and Frank C Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[52] David Morin. The lagrangian method. Accessed: 5-23-2023.

[53] Eric R Westervelt, Jessy W Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC press, 2018.

[54] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A Mathematical Introduction toRobotic Manipulation*. CRC press, 1994.

[55] Thomas A Henzinger. The theory of hybrid automata. In *Proceedings 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292. IEEE, 1996.

[56] John Lygeros. Lecture notes on hybrid systems. In *Notes for an ENSIETA workshop*, 2004.

[57] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017.

[58] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[59] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[60] Gene F Franklin, J David Powell, Abbas Emami-Naeini, and J David Powell. *Feedback control of dynamic systems*, volume 4. Prentice hall Upper Saddle River, 2002.

[61] Kemin Zhou Li Qui. *Introduction to Feedback Control*. Pearson Education, 2010.

[62] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. In *2019 American Control Conference (ACC)*, pages 4559–4566, 2019.

[63] O. Harib, A. Hereid, A. Agrawal, T. Gurriet, S. Finet, G. Boeris, A. Duburcq, M. E. Mungai, M. Masselin, A. D. Ames, K. Sreenath, and J. W. Grizzle. Feedback control of an exoskeleton for paraplegics: Toward robustly stable, hands-free dynamic walking. *IEEE Control Systems Magazine*, 38(6):61–87, 2018.

[64] Grant Gibson, Oluwami Dosunmu-Ogunbi, Yukai Gong, and Jessy Grizzle. Terrain-adaptive, alip-based bipedal locomotion controller via model predictive control and virtual constraints. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6724–6731. IEEE, 2022.

[65] Xingye Da and Jessy Grizzle. Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots. *The International Journal of Robotics Research*, 38(9):1063–1097, 2019.

[66] Oluwami Dosunmu-Ogunbi, Aayushi Shrivastava, Grant Gibson, and Jessy W Grizzle. Stair climbing using the angular momentum linear inverted pendulum model and model predictive control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8558–8565. IEEE, 2023.

[67] Hassan K Khalil. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[68] Jerry E Pratt and Russ Tedrake. Velocity-based stability margins for fast bipedal walking. *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*, pages 299–324, 2006.

[69] Pierre-Brice Wieber. On the stability of walking systems. In *Proceedings of the international workshop on humanoid and human friendly robotics*, 2002.

[70] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous robots*, 35:161–176, 2013.

[71] Elena Garcia, Joaquin Estremera, and Pablo Gonzalez-de Santos. A classification of stability margins for walking robots. *Robotica*, 20(6):595–606, 2002.

[72] Hirohisa Hirukawa, Shizuko Hattori, Kensuke Harada, Shuuji Kajita, Kenji Kaneko, Fumio Kanehiro, Kiyoshi Fujiwara, and Mitsuharu Morisawa. A universal stability criterion of the foot contact of legged robots-adios zmp. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1976–1983. IEEE, 2006.

[73] Sjoerd M Bruijn, OG Meijer, PJ Beek, and Jaap H van Dieen. Assessing the stability of human locomotion: a review of current measures. *Journal of the Royal Society Interface*, 10(83):20120999, 2013.

[74] Ambarish Goswami and Vinutha Kallem. Rate of change of angular momentum and balance maintenance of biped robots. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3785–3790. IEEE, 2004.

[75] Arash Mahboobin, Patrick J Loughlin, Mark S Redfern, Stuart O Anderson, Christopher G Atkeson, and Jessica K Hodgins. Sensory adaptation in human balance control: lessons for biomimetic robotic bipeds. *Neural Networks*, 21(4):621–627, 2008.

[76] Satoshi Ito, Tomohiro Nishigaki, and Haruhisa Kawasaki. Upright posture stabilization by ground reaction force control. *Proc., Int. Sympo. On Measurement, Analysis and Modeling of Human Functions (ISHF2001)*, pages 515–520, 2001.

[77] Hun ok Lim, S.A. Setiawan, and A. Takanishi. Balance and impedance control for biped humanoid robot locomotion. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*, volume 1, pages 494–499 vol.1, 2001.

[78] Ye Xie, Bin Lou, Anhuan Xie, and Dan Zhang. A review: Robust locomotion for biped humanoid robots. In *Journal of Physics: Conference Series*, volume 1487, page 012048. IOP Publishing, 2020.

[79] Nils Smit-Anseeuw, C. David Remy, and Ram Vasudevan. Walking with confidence: Safety regulation for full order biped models. *IEEE Robotics and Automation Letters*, 4(4):4177–4184, 2019.

[80] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 295–302. IEEE, 2014.

[81] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *2006 6th IEEE-RAS international conference on humanoid robots*, pages 200–207. IEEE, 2006.

[82] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

[83] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun Ho Choi, and Benjamin Morris. *Feedback Control of Dynamic Bipedal Robot Locomotion*, chapter 5. CRC Press, 2009.

[84] Russ Tedrake. *Underactuated Robotics*. 2023.

[85] Shuuji Kajita and Bernard Espiau. Legged robot, 2008.

[86] P. Sardain and G. Bessonnet. Forces acting on a biped robot. center of pressure-zero moment point. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 34(5):630–637, 2004.

[87] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[88] Srikanth Thudumu, Philip Branch, Jiong Jin, and Jugdutt Jack Singh. A comprehensive survey of anomaly detection techniques for high dimensional big data. *Journal of Big Data*, 7(1):1–30, 2020.

[89] Lukas Ruff, Jacob R Kauffmann, Robert A Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021.

[90] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[91] Mostafa Rahmani and George K Atia. Coherence pursuit: Fast, simple, and robust principal component analysis. *IEEE Transactions on Signal Processing*, 65(23):6260–6275, 2017.

[92] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[93] Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3379–3388, 2018.

[94] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1705–1714, 2019.

[95] M Eva Mungai and Jessy W Grizzle. Feedback control design for robust comfortable sit-to-stand motions of 3d lower-limb exoskeletons. *IEEE Access*, 9:122–161, 2020.

[96] Akim Kapsalyamov, Prashant K Jamwal, Shahid Hussain, and Mergen H Ghayesh. State of the art lower limb robotic exoskeletons for elderly assistance. *IEEE Access*, 7:95075–95086, 2019.

[97] Alberto Esquenazi, Mukul Talaty, and Arun Jayaraman. Powered exoskeletons for walking assistance in persons with central nervous system injuries: A narrative review. *PM&R*, 9(1):46–62, 2017.

[98] A. Norhafizan, R.A.R. Ghazilla, Vijayabaskar Kasi, Z. Taha, and Bilal Hamid. A review on lower-limb exoskeleton system for sit to stand, ascending and descending staircase motion. In *Engineering and Manufacturing Technologies*, volume 541 of *Applied Mechanics and Materials*, pages 1150–1155. Trans Tech Publications Ltd, 6 2014.

[99] W. Sun, J. Lin, S. Su, N. Wang, and M. J. Er. Reduced adaptive fuzzy decoupling control for lower limb exoskeleton. *IEEE Transactions on Cybernetics*, page 1, 2020.

[100] Jonas Vantilt, Kevin Tanghe, Maarten Afschrift, Amber KBD Bruijnes, Karen Junius, Joost Geeroms, Erwin Aertbeliën, Friedl De Groote, Dirk Lefeber, Ilse Jonkers, et al. Model-based control for exoskeletons with series elastic actuators evaluated on sit-to-stand movements. *Journal of NeuroEngineering and Rehabilitation*, 16(1):65, 2019.

[101] Mohamed Amine Alouane, Weiguang Huo, Hala Rifai, Yacine Amirat, and Samer Mohammed. Hybrid fes-exoskeleton controller to assist sit-to-stand movement. *IFAC-PapersOnLine*, 51(34):296–301, 2019.

[102] M. Mistry, A. Murai, K. Yamane, and J. Hodgins. Sit-to-stand task on a humanoid robot from human demonstration. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 218–223, 2010.

[103] A. Tsukahara, Y. Hasegawa, and Y. Sankai. Standing-up motion support for paraplegic patient with robot suit hal. In *2009 IEEE International Conference on Rehabilitation Robotics*, pages 211–217, 2009.

[104] Gaurav Patil, Lillian Rigoli, Michael J Richardson, Manish Kumar, and Tamara Lorenz. Momentum-based trajectory planning for lower-limb exoskeletons supporting sit-to-stand transitions. *International Journal of Intelligent Robotics and Applications*, 2(2):180–192, 2018.

[105] Bing Chen, Chun-Hao Zhong, Hao Ma, Xiao Guan, Lai-Yin Qin, Kai-Ming Chan, Sheung-Wai Law, Ling Qin, and Wei-Hsin Liao. Sit-to-stand and stand-to-sit assistance for paraplegic patients with cuhk-exo exoskeleton. *Robotica*, 36(4):535–551, 2018.

[106] Diego Felipe Paez Granados, Hideki Kadone, and Kenji Suzuki. Unpowered lower-body exoskeleton with torso lifting mechanism for supporting sit-to-stand

transitions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2755–2761. IEEE, 2018.

[107] S Jatsun, S Savin, and A Yatsun. Motion control algorithm for a lower limb exoskeleton based on iterative lqr and zmp method for trajectory generation. In *International Workshop on Medical and Service Robots*, pages 305–317. Springer, 2016.

[108] Jernej Kuželički, Miloš Žefran, Helena Burger, and Tadej Bajd. Synthesis of standing-up trajectories using dynamic optimization. *Gait & posture*, 21(1):1–11, 2005.

[109] S. Pchelkin, A. Shiriaev, L. Freidovich, U. Mettin, S. Gusev, and W. Kwon. Natural sit-down and chair-rise motions for a humanoid robot. In *49th IEEE Conference on Decision and Control (CDC)*, pages 1136–1141, 2010.

[110] Rahman Davoodi and Brian J Andrews. Optimal control of fes-assisted standing up in paraplegia using genetic algorithms. *Medical engineering & physics*, 21(9):609–617, 1999.

[111] C. Wu, T. Zhang, Y. Liao, C. Wang, G. Wu, and X. Wu. Self-adaptive control strategy for exoskeleton to help paraplegic patients stand up and sit down. In *2016 35th Chinese Control Conference (CCC)*, pages 6189–6194, 2016.

[112] A. Hereid and A. D. Ames. Frost: Fast robot optimization and simulation toolkit. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 719–726, 2017.

[113] Hui Cheng, Yiu-Kuen Yiu, and Zexiang Li. Dynamics and control of redundantly actuated parallel manipulators. *IEEE/ASME Transactions on Mechatronics*, 8(4):483–491, 2003.

[114] D. J. Braun, Y. Chen, and L. Li. Operational space control under actuation constraints using strictly convex optimization. *IEEE Transactions on Robotics*, 36(1):302–309, 2020.

[115] W. Shang, S. Cong, Y. Zhang, and Y. Liang. Active joint synchronization control for a 2-dof redundantly actuated parallel manipulator. *IEEE Transactions on Control Systems Technology*, 17(2):416–423, 2009.

[116] M. W. Oppenheimer, D. B. Doman, and M. A. Bolender. Control allocation for over-actuated systems. In *2006 14th Mediterranean Conference on Control and Automation*, pages 1–6, 2006.

[117] Matthew G Feemster and Joel M Esposito. Comprehensive framework for tracking control and thrust allocation for a highly overactuated autonomous surface vessel. *Journal of Field Robotics*, 28(1):80–100, 2011.

[118] J. H. Plumlee, D. M. Bevly, and A. S. Hodel. Control of a ground vehicle using quadratic programming based control allocation techniques. In *Proceedings of the 2004 American Control Conference*, volume 5, pages 4704–4709 vol.5, 2004.

[119] J. A. M. Petersen and M. Bodson. Constrained quadratic programming techniques for control allocation. *IEEE Transactions on Control Systems Technology*, 14(1):91–98, 2006.

[120] Rex Bionics. Rex technology. Accessed: April 2020.

[121] Taslim Reza, Sharif Muhammad, Norhafizan Ahmad, Imtiaz Ahmed Choudhury, and Raja Ariffin Raja Ghazilla. A fuzzy controller for lower limb exoskeletons during sit-to-stand and stand-to-sit movement using wearable sensors. *Sensors (Basel)*, 14(3):4342–4363, 2014.

[122] Sergey Jatsun, Sergei Savin, Andrey Yatsun, and Andrei Malchikov. Study of controlled motion of exoskeleton moving from sitting to standing position. In *Advances in Robot Design and Intelligent Control*, pages 165–172. Springer, 2016.

[123] Jatsun, Sergey, Savin, Sergei, Lushnikov, Boris, and Yatsun, Andrey. Algorithm for motion control of an exoskeleton during verticalization. *ITM Web of Conferences*, 6:01001, 2016.

[124] Daniel S Williams and Anne E Martin. Does a finite-time double support period increase walking stability for planar bipeds? *Journal of Mechanisms and Robotics*, pages 1–25, 2020.

[125] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot Modeling and Control*. John Wiley & Sons, Ltd., 2006.

[126] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle. Rapid trajectory optimization using c-frost with illustration on a cassie-series dynamic walking biped. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4722–4729, 2019.

[127] Dudley D. Fuller. Coefficients of friction. Accessed: August 2020.

[128] AC Franzoi, C Castro, and C Cardone. Isokinetic assessment of spasticity in subjects with traumatic spinal cord injury (asia a). *Spinal Cord*, 37(6):416–420, 1999.

[129] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[130] Tom Erez, Yuval Tassa, and Emanuel Todorov. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 4397–4404. IEEE, 2015.

[131] Alexis Duburq. jiminy. `https://github.com/duburcqa/jiminy`, 2019.

[132] M Eva Mungai and Jessy Grizzle. Optimizing lead time in fall detection for a planar bipedal robot. *arXiv preprint arXiv:2303.15620*, 2023.

[133] Eliahu Khalastchi, Meir Kalech, Gal A Kaminka, and Raz Lin. Online data-driven anomaly detection in autonomous robots. *Knowledge and Information Systems*, 43:657–688, 2015.

[134] H Safaeipour, M Forouzanfar, and AJJoPC Casavola. A survey and classification of incipient fault diagnosis approaches. *Journal of Process Control*, 97:1–16, 2021.

[135] Hongtian Chen, Bin Jiang, Ningyun Lu, and Wen Chen. *Probability-Relevant PCA-based FDD Methods*, pages 81–98. Springer International Publishing, Cham, 2020.

[136] The MathWorks Inc. Matlab version: 9.6.0 (r2019a), 2019.

[137] Gábor J. Székely and Maria L. Rizzo. Partial distance correlation with methods for dissimilarities. *The Annals of Statistics*, 42(6):2382 – 2412, 2014.

[138] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[139] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Springer New York, NY, 2006.

[140] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[141] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[142] M Eva Mungai, Gokul Prabhakaran, and Jessy W Grizzle. Fall prediction for bipedal robots: The standing phase. *arXiv preprint arXiv:2309.14546*, 2023.

[143] Rajesh Subburaman, Dimitrios Kanoulas, Nikos Tsagarakis, and Jinoh Lee. A survey on control of humanoid fall over. *Robotics and Autonomous Systems*, 166:104443, 2023.

[144] Zhibin Li, Chengxu Zhou, Juan Castano, Xin Wang, Francesca Negrello, Nikos G. Tsagarakis, and Darwin G. Caldwell. Fall prediction of legged robots based on energy state and its implication of balance augmentation: A study on the humanoid. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5094–5100, 2015.

[145] Benjamin Stephens. Humanoid push recovery. In *2007 7th IEEE-RAS International Conference on Humanoid Robots*, pages 589–595, 2007.

[146] Oliver Höhn, J Gačnik, and Wilfried Gerth. Detection and classification of posture instabilities of bipedal robots. In *Climbing and Walking Robots: Proceedings of the 8th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2005)*, pages 409–416. Springer, 2006.

[147] Zhangming He, Yuri AW Shardt, Dayi Wang, Bowen Hou, Haiyin Zhou, and Jiongqi Wang. An incipient fault detection approach via detrending and denoising. *Control Engineering Practice*, 74:1–12, 2018.

[148] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[149] Yuzhe Yang, Kaiwen Zha, Yingcong Chen, Hao Wang, and Dina Katabi. Delving into deep imbalanced regression. In *International Conference on Machine Learning*, pages 11842–11851. PMLR, 2021.

[150] Luís Torgo, Paula Branco, Rita P Ribeiro, and Bernhard Pfahringer. Resampling strategies for regression. *Expert Systems*, 32(3):465–476, 2015.

[151] Paula Branco, Luís Torgo, and Rita P Ribeiro. Smogn: a pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications*, pages 36–50. PMLR, 2017.

[152] Mike Kamermans. A primer on bezier curves. Accessed: September 2020.