Advances in Feedback Control for High-dimensional Bipedal Models

by

Omar Harib

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical and Computer Engineering)
in the University of Michigan
2021

Doctoral Committee:

Professor Jessy W. Grizzle, Chair
Assistant Professor Maani Ghaffari
Associate Professor Robert D. Gregg, IV
Assistant Professor Ram Vasudevan



Omar Harib

oharib@umich.edu

ORCID iD: 0000-0001-9594-0967

©Omar Harib 2021

To my family

ACKNOWLEDGMENTS

My work and and my journey during my PhD program would not have been possible without the love and support of my family. To my Mom and Dad and to my siblings: Abdulla, Saeed, Ahmad, Shamma, and Roudha, thank you for always being there for me. I am very proud to be your son and brother. You will always mean the world to me.

I am deeply thankful for the guidance of Professor Jessy Grizzle. My time in his lab had the right mixture of joy and stress to keep me motivated. I am grateful to the members of the Biped Lab. We always worked together to achieve seemingly impossible goals, and yet, we manged to do many of them and even when we did not, we learned a lot.

My thanks go to the senior members of the lab, Brian Buss, Brent Griffin, and Xingye Da, for showing me the ropes when I was new to the lab. Thank you Ross Hartley, Yukai Gong, Eva Mungai, Bruce Huang, Grant Gibson, Oluwami Dosunmu-Ogunbi, Ray Zhang, and Lu Gan for making the lab a fun and exciting place. I am especially grateful to Yukai for all the time we spent conducting experiments and, in general, learning how to make algorithms work on challenging hardware.

I thank, then post-docs, now professors, Ayonga Hereid and Zhenyu Gan for their leadership and collaboration. I also thank Ayush Agrawal and Thomas Gurriet with whom I collaborated on much of the exoskeleton work. Even more broadly, my thanks go to Wandercraft and its amazing team of engineers for allowing me to work in Paris.

I extend my gratitude to the dissertation committee, Professors Maani Ghaffari, Robert D. Gregg, IV, and Ram Vasudevan for their support and valuable feedback.

Lastly, I want to thank some special people in my life whose presence made my time in the States all the more enjoyable: Izzy, Galen, Leo, Luka, Nick, Maddie, and Madeline.

Thank you all from the bottom of my heart!

TABLE OF CONTENTS

Dedication . . . . . **ii**

Acknowledgments . . . . . **iii**

List of Figures . . . . . **vii**

List of Tables . . . . . **xi**

List of Appendices . . . . . **xii**

Abstract . . . . . **xiii**

Chapter

1 Introduction **1**

1.1 Motivation 1
1.2 Contributions of the Dissertation 4
1.3 Outline 5

2 Background on Control Approaches **7**

2.1 Motivation 7
2.2 Model 7
2.3 Virtual Constraints and HZD 8
2.3.1 Virtual Constraints 8
2.3.2 Computer Optimization 10
2.3.3 Stability with One Degree of Underactuation 11
2.3.4 Foot Placement as Means of Achieving Stability with Greater Underactuation 11
2.4 Gait Library 12
2.5 Background into G-HZD 13

3 Speeding up Optimization **16**

3.1 Motivation 16
3.1.1 FROST 18
3.1.2 Robot Behavior Modeling 18
3.1.3 Trajectory Optimization 20
3.2 C-FROST Implementation 21
3.2.1 Formulation of NLP Problems 21
3.2.2 Execution of NLP Problems 22

3.2.3 Parallel Evaluation of NLP Problems . . . 24
3.3 A Case Study: Cassie Blue . . . 25
3.3.1 Testbed Robot Model . . . 25
3.3.2 Optimization Problem . . . 25
3.4 Results . . . 27
3.4.1 Computational Performance Analysis . . . 27
3.4.2 Removing the Velocity Terms . . . 30
3.4.3 Experimental Validation . . . 33
3.5 Conclusions . . . 34
4 Control of an Exoskeleton . . . 35
4.1 Motivation . . . 35
4.2 Model . . . 39
4.2.1 Hardware Description . . . 39
4.2.2 Mathematical Representation . . . 40
4.2.3 Model with Human separately . . . 42
4.3 Feedback for Centralized Control . . . 42
4.3.1 Virtual Constraints . . . 42
4.3.2 Partial Hybrid Zero Dynamics and a Central Controller . . . 43
4.3.3 Gait Generation via Direct Collocation Method . . . 44
4.4 Velocity Regulation . . . 44
4.4.1 Library of Gaits . . . 45
4.4.2 Relaxing local stability of walking speed . . . 46
4.4.3 User Regulation of Walking Speed . . . 46
4.4.4 Simple Model of User Control Actions . . . 47
4.5 Decentralized Controller . . . 50
4.5.1 Control Design Assumptions . . . 50
4.5.2 Decentralized Control Design . . . 52
4.6 Gait Library Results and Discussion . . . 54
4.6.1 Centralized vs Decentralized Implementations with a Fixed Gait . . . 54
4.6.2 Centralized vs Decentralized Implementations Velocity Regulation . . . 56
4.7 G-HZD Controller Architecture, Gait Objectives, and Important Constraints 59
4.7.1 Gait Design Objectives . . . 59
4.7.2 Ground Contact . . . 60
4.7.3 Other Objectives . . . 61
4.7.4 Controller Architecture . . . 61
4.8 G-HZD: Harnessing the Power of Modern Optimization Techniques . . . 64
4.8.1 Remark on Trajectory Design via Optimization . . . 72
4.9 G-HZD Numerical Illustration . . . 73
4.9.1 Velocity Tracking . . . 73
4.9.2 Preliminary Robustness Analysis . . . 73
5 From Partial-GHZD to Full-GHZD in High Dimensional Models . . . 79
5.1 Motivation . . . 79
5.2 Recalling the System Model . . . 80

5.3 New Approach . . . 81
5.3.1 Linearized Model and Assumptions . . . 82
5.4 Illustration: Linear Inverted Pendulum on a Cart . . . 84
5.4.1 Derivation . . . 84
5.4.2 Comparison to original G-HZD approach . . . 88
6 Cassie Implementation . . . 91
6.1 Nominal Orbit and LQR Optimization . . . 91
6.1.1 Nominal orbit and linearizing the model . . . 91
6.1.2 LQR . . . 92
6.1.3 Forming Z_0 surface . . . 93
6.2 Frost Optimizations . . . 94
6.3 Controller . . . 94
6.4 Experiments and Results . . . 95
7 Discussion and Future Works . . . 98
Appendices . . . 99
Bibliography . . . 117

LIST OF FIGURES

FIGURE

1.1 MARLO going up a steep hill at the University of Michigan. 3
1.2 Cassie roaming around in the Dental School at the University of Michigan. . . . 4

2.1 Showing virtual constraints choices on MARLO 10
2.2 Depicts example of a biped walking in-place versus falling forward and having to catch itself using foot placement. (b) has a forward velocity. 12
2.3 Example of a controller design using a gait library approach. 13
2.4 Graphical representation of gait optimization for G-HZD as developed by Da. . 15
2.5 Representations of Z 15

3.1 (a) Shows Cassie Blue, one of the Michigan copies of the Cassie series of robots built by Agility Robotics. (b) Shows the joints and the kinematic model of the robot. 17
3.2 The process of modeling and optimizing dynamic walking gaits in FROST. Given a robot model (e.g., via a URDF file), FROST makes it easy for users to define the desired behavior and provide the necessary physical constraints to formulate a realistic gait optimization problem. 19
3.3 Stereoscopic tiles for stepping up/down at different walking speeds. 27
3.4 The computation time of each components while using different number of threads. 29
3.5 Comparison of the relative contributions of the various components of the dynamic equation (2.1) for forward walking at 1 m/s on flat ground. 31
3.6 Projections of the periodic orbits for the gait in Figure 3.5 over one step on each leg. (a) A typical actuated joint, the left knee, and (b) the only "passive" variable in single support, torso roll. Note that the roll difference is less than one degree. 32
3.7 Comparison of the actuator torques and the vertical ground reaction force for the gait in Figure 3.5 over one step. 33
3.8 Experimental results for a gait-library-based sustainable walking controller on Cassie Blue. (a) The tracking of the commanded forward/backward walking speeds (measured at the CoM position). (b) The periodic phase portrait of an actuated joint (right knee) when stepping in place. 34

4.1 ATALANTE: An exoskeleton designed by Wandercraft for people with paraplegia. 36
4.2 The kinematic diagram of human-exoskeleton lumped system. The patient is secured to the exoskeleton by means of fasteners located at the ankle, the shin, the thigh, the abdomen, and the torso. 40
4.3 The kinematic diagram of human-exoskeleton lumped system where the human is assumed to be a lump of mass attatched to the exo, similar to a rigid mannequin. 41

4.4 Snapshots of walking at a desired speed of 0.3 m/s achieved by our proposed controller on the human-exoskeleton system is shown. An animation video of the simulation is available at `https://youtu.be/VXP7DKY6Trc`. 44
4.5 Feedback diagram illustrating the decentralized controller with velocity regulation. Block in yellow denotes continuous-time, decentralized controller. Block in blue denotes the discrete-time event-based velocity regulation controller, executed at the end of every walking step. Dashed lines denote variables sampled at the end of every step. 51
4.6 Comparison between the Centralized and Decentralized controller in handling a perturbation in the initial conditions of the exoskeleton states; recall that the user has an upright posture. Both controllers are implementing the same periodic orbit. Figures (a) and (b) show the velocity tracking. Figures (c) and (d) show phase portraits converging to periodic orbits. Figures (e) and (f) show output errors converging to zero. 55
4.7 Comparison between Centralized and Decentralized controllers in tracking desired velocities. Figures (a) and (b) show the velocity tracking. Figures (c) and (d) torso tracking at the end of each step. Figures (e) and (f) show ZMP being within the foot's geometry. 57
4.8 Comparison of the effect of arm swinging on the decentralized controller. Figure (a) shows the case with the arms stationary. Figure (b) shows the case with the arm swinging $\pm 7.5°$. In both cases, the exoskeleton controller assumes that the arms are stationary. 58
4.9 Simulation results of walking up a 5° slope. Figure (a) shows a snapshot of the simulation. Figure (b) shows the velocity tracking of the closed-loop system. . . 58
4.10 Exoskeleton with rigid human assumption. 60
4.11 Depicts feet of the exoskeleton, with blue the left foot pink the right foot. L_h and L_t denote the distance from the point of reference to the heel and toe respectively. W_f denotes the width of the foot. Shaded area depicts the desired location of the Zero Moment Point (ZMP). $[F_x, F_y, F_z, M_x, M_y, M_z]$ is the resultant ground reaction force and moment acting on the foot during support, described in the body coordinate of the foot. 62
4.12 Overview of the controller structure. The Guard Checker handles detecting when to swap legs. The Control Policy specifies the desired gait and the means to achieve it. The Low-level Controller translates control policy commands to desired joint-level trajectories and achieves them via PD control. 63
4.13 A collection of trajectories forming a smooth surface; while this is the desired outcome, it is not guaranteed. Direct-collocation-based optimization is used to design the trajectories. 65
4.14 Gedanken Experiment. (a) Assume the system is initialized at ξ^a, with input, $u_{\xi^a}(t)$, being applied, and hence its solution is evolving along $\varphi_{\xi^a}(t)$. Suppose that at time t_d, an "impulsive" disturbance instantaneously displaces the solution to a point $x(t_d)$. What input should be applied? (b) If there exists an $\xi^d \in Z_0$ such that $x(t_d) = \varphi_{\xi^d}(t_d)$, then applying the input $u_{\xi^d}(t)$ for $t_d \leq t < T_p$ will move the system toward the equilibrium in the sense that $||\varphi_{\xi^d}(T_p)) - \xi^*|| \leq c||\xi^d - \xi^*||$. This thought process leads to the "interpolation" or "learning" conditions in (4.21). 67

4.15 Shows forward velocity of the hip resulting from the transition optimizations. Black vertical dashed lines mark the midpoint of each (robot) step in the optimization, whereas the discontinuities (jumps) in velocity between two dashed lines arise from the rigid impact at the end of each step. Squares, circles, and the triangle mark points where the state of the system is in Z_0. The blue squares mark starting states obtained from periodic gaits while the green triangle marks the ending desired state also obtained from a periodic gait. The red circles were made to be in Z_0 through an optimization constraint. The gait trajectory from the blue square to the red circle is what is used as training data for the Supervised Machine Learning. 68
4.16 A vector field is constructed that gives rise to the trajectories, so it is a state-variable realization. Supervised Machine Learning (fancy regression) and model structure are used to "extract" the vector field from the trajectory optimization data. 69
4.17 Surface showing the desired swing knee angle as a function of the gait timing variable τ and the forward hip velocity for stabilizing the step-in-place gait. The red circles are the actual data obtained from optimization while the surface is a regression done using the Neural Network Toolbox in MATLAB. Note that some of the circles are under the surface and are therefore not visible. 71
4.18 The low-dimensional realization (invariant surface and vector field) being rendered attractive in the full order model. 71
4.19 Shows tracking of a desired velocity profile. v_x^a is the actual forward velocity of the pelvis during walking. v_x^d is the commanded average forward velocity. 74
4.20 Shows the gait timing variable τ changing as a function of time while the exoskeleton was following a varying desired speed. 74
4.21 The phase portrait of both the swing and stance sagittal hip joints for when the system was tracking a desired velocity profile. Shows q vs $\dot{q}$ for the two joints. The solid blue line shows the states at the last quarter of the simulation. While the solid red line shows the part of the simulation where the exoskeleton reached a steady state trajectory while walking forward. It can be seen that both cases, the exoskeleton converged to a periodic cycle. 75
4.22 Outtakes from animations of the exoskeleton tracking a desired velocity profile. Below are images from the Gazebo simulation, while above are stick figures with the same motion added for clarity. The first two photos show the step-in-place gait. The third image shows how the system looks like when it is starting to transition. The last two show the exoskeleton walking forward in a periodic gait. 76
4.23 Change in hip velocity due to a perturbation. The control algorithm based on Supervised Machine Learning is able to recover from larger perturbations. Both controllers failed under perturbations larger than those shown. 77
4.24 The swing hip joint angle as a function of time while responding to a perturbation. When experiencing large perturbations, the controller based on Supervised Machine Learning makes more use of its swing foot in order to return to a stepping-in-place gait. 77

4.25 Instantaneous velocity of the hip when system is perturbed for a duration for 10 seconds. The forward push causes the system to drift slightly more while the force is being applied. When force is removed, the velocity converges back to the nominal velocity. 78

5.1 Structure of a controller under new approach. 84
5.2 Pendulum on a cart 85
5.3 Results from testing different controllers on the linear cart and pendulum model 87
5.4 Shows how the state of the system in the Z_0 manifold would look to the same perturbation in the weakly actuated states. 89

6.1 Overview of the controller structure implemented on Cassie. Guard Checker handles swapping the stances. Weakly-actuated state estimator calculates x_w. The neural network computes the desired x_s. The The Low-level Controller tracks the joints using passivity based control. 95
6.2 Picture of Cassie being pushed during the experiment. 96
6.3 Estimated velocity of Cassie during the experiment. 96
6.4 Ground reaction forces on Cassie's feet during experiment. 97
6.5 Input to the neural network during experiment. 97

A.1 Illustration of defect constraints and node distribution of the direct collocation optimization 100

LIST OF TABLES

TABLE

3.1 Time to perform $11^3 = 1,331$ gait optimizations for the Cassie. The same pre-compiled code is run on all platforms. The average compute time for a gait is similar in all cases; the total times essentially inversely scale with the number of threads with slight differences with regards to the CPU frequency. 28

3.2 Time to perform $11^3 = 1,331$ gait optimizations for the Cassie with the Velocity terms removed. 32

4.1 Dominant Eigenvalues at different speeds. 46

4.2 List of variables pertaining to the controller design, velocity regulation, and decentralized controller 49

4.3 Performance comparison between direct-collocation optimization vs classic shooting approaches on a 5-link planar biped (single-shooting) and a 7-link spring-leg planar biped, respectively. 64

4.5 Table showing a mapping between higher-level control objectives and their realization at a lower-level. 70

4.6 The mean-absolute error (MAE) for joint angle tracking when applying external torques on the right knee joint representing a simplified model of spasticity. . . 78

LIST OF APPENDICES

APPENDIX

A How Direct Collocation Works **99**

B Partial Hybrid Zero Dynamics and a Central Controller **101**

C Constraints for exoskeleton optimization **103**

D Linearizing a hybrid model **105**

ABSTRACT

The promise of bipedal machines brings with it the promise of advancements in both the work and personal sectors. The robots are coming to work alongside us at our pace. They are coming to work in environments which we deem too dangerous to send our fellow humans. They are coming to help us recover mobility when we suffer neural or muscular impairments. The machines being prototyped and built for these tasks are complex devices, with many links, joints, and actuators. Moreover, these bipedal machines are meant to exhibit diverse gaits, and many are designed with agility in mind. Taking full advantage of their mechanical capabilities requires advanced optimization and feedback design methods, which are the central theme of this dissertation. While bipedal machines typically refer to two legged robots, we also in our research consider full-assist exoskeletons as well. Our specific hardware targets are (1) an exoskeleton designed by Wandercraft that allows people with paraplegia to walk again without the use of crutches, and (2) a Cassie-series bipedal robot that can stand quietly in place, balance on a Segway, and walk at over a meter per second. The high-dimensional and hybrid natures of the dynamical models of these robots pose challenges in optimal gait design and gait stabilization that we identify and address in this dissertation.

The first topic addressed in this dissertation is trajectory optimization. Specifically, we developed a toolset called C-FROST that speeds up the offline trajectory design process by a factor of six to eight times compared to using FROST. This allows the generation of 1000+ gaits for a 20 degrees of freedom (DOF) biped in under two hours. C-FROST allows FROST to generate a stand-alone C++ executable for running optimizations, thereby allowing for parallelization and easy deployment to the cloud. Benchmarking and practical examples demonstrating the capabilities of the C-FROST toolset are provided. The second topic is gait and feedback control design for the full-assist Wandercraft exoskeleton. This work was done early in the dissertation research and provided prototype software to the Wandercraft team. In addition to the exoskeleton's model being high dimensional (with 36 state variables), its operation poses difficult challenges arising from workspace limitations required for patient safety and torque limits of the hardware. Feedback controllers are designed to meet these challenges using the method of Hybrid Zero Dynamics (HZD), gait libraries, and an extension of HZD by Xingye Da that is called Generalized Hybrid Zero Dynamics, or G-HZD for short.

Work on the Wandercraft exoskeleton revealed a fundamental drawback in the original G-HZD approach, and that motivates the third main topic of the dissertation: a broad extension of the design and stability properties of the hybrid zero dynamics manifold in the G-HZD feedback design approach. In particular, in G-HZD, the manifold is built from trajectories that arise from a boundary-value problem, and thus specifying the proper boundary of the manifold is crucial to the success of the method. This dissertation proposes a novel method for the design of the boundary manifold. The new approach is subsequently demonstrated on an inverted pendulum on a cart and compared to the original approach. We then demonstrate how to implement the new approach on Cassie, an underactuated biped hybrid system, and report experimental results.

CHAPTER 1

Introduction

1.1 Motivation

When dealing with rough terrain, from dense forests to tight factory environments, wheeled vehicles are often ill-equipped at land traversal. Legged animals, however, have little difficulties in any of said environments. That is where legged robots come into play.

The field of legged robots has dedicated significant effort towards the goal of transferring the level of agility, stability, and robustness we see in animals to machines. This work is not only seen in walking bipeds [30, 98, 24, 95], but in exoskeletons [13, 48, 127, 68] and prosthetics [42, 88] as well. Focusing specifically on bipeds, researchers seek to have robots accomplish what humans typically can. The interiors or our homes and factories are designed to be traversed by two legged beings, and thus, research into bipeds is relevant. In Figure 1.1, we see a biped robot called MARLO navigating very steep hills. While in Figure 1.2, we see a biped robot called Cassie roaming around the corridors of a building.

The subject of feedback control of bipedal locomotion has grown into a significant field over the past 20 years, meriting a number of survey articles and books [67, 89, 114, 77]. Instead of reviewing that literature, we'll give a brief overview of the work in the Biped Lab [6] that preceded and supports the work in this dissertation.

A significant breakthrough came in the dissertation of Eric Westervelt, where he introduced the method of Hybrid Zero Dynamics (HZD) for underactuated 2D planar robots. Working directly with the full-order dynamical model, the method used virtual constraints and input-output linearization to induce a low-dimensional model of the same dimension as the robot's degree of underactuation. When the low-dimensional model was all invariant under the impact map, the resulting hybrid system was called the HZD. By combining optimization and HZD, a wide variety of gaits were designed for Rabbit, AMBER2, and Ernie [118, 79, 121, 125].

Walking in 3D presents its own challenges with both added degrees of freedom and

underactuation. Methods that for 2D robots were delivering robust stable locomotion were failing to even stabilize walking gaits at modest speeds for the 3D bipedal robot, MARLO. To tackle this, Brian Buss showed how to parameterize the virtual constraints in a manner that Bilinear Matrix Inequalities could be incorporated with parameter to ensure that the eigenvalues of an orbit's Poincaré map were strictly within the unit circle. This worked well in the laboratory, but faced challenges in outdoor terrain. Brent Griffin contributed the notion of non-holonomic virtual constraints to achieve robust walking in 3D outside of the laboratory [44]. Griffin then set up an optimization that solves for the virtual constraint parameters such that it can reject terrain height and velocity disturbances. MARLO was then able to walk outdoors on grass fields and parking lots at speeds between 0.9-0.98 m/s.

A more common option for achieving stability in 3D-underactuated robots was through "foot-placement" control policies that were often hand-designed [93]. Xingye Da et al. developed an approach using a "periodic gait library" consisting of virtual constraints that realized gaits of different speeds and directions, that when combined with foot-placement, significantly enhanced the basin of attraction on the actual robot [30]. The same paper also analyzed stability of gait library method.

In order to interpolate among the various gaits in his periodic gait library, Da was experimenting with neural networks. Within this framework, he realized that he could also include transition gaits between periodic gaits, leading to a kind of gait transition graph that had connections to Model predictive control (MPC). The design of the transition gaits led him to the idea of stabilizing a periodic orbit by designing trajectories that transition to the orbit in a finite number of states, typically three. In order to get around the curse of dimensionality, he would design the family of trajectories to form a low-dimensional "boundary manifold", and also render it invariant under the impact map. This ultimately led Da to an approach called the Generalized Hybrid Zero Dynamics (G-HZD) that utilized offline optimization and supervised learning techniques to build a control policy to achieve stability in the sagittal plane without the need for foot-placement along that plane [31].

In this thesis, we continue the Biped Lab's legacy of novel contributions to model-based control. We had the opportunity to work with the engineers at Wandercraft [11] early on in their control design for an exoskeleton that allows patients with paraplegia to walk without crutches. The complexity of the Wandercraft exoskeleton led to a very slow design cycle when attempting to apply advanced methods from the HZD and gait library frameworks [13, 51]. The main culprit was the time spent on optimizations and it was clear that applying Da's G-HZD method to the Watercraft Exoskeleton or to Cassie would not be a pleasant task. This motivated us to develop a stand alone C++ implementation of the FROST package designed by Ayong Hereid [55, 58], speeding it up by a factor between six to eight. After this work, we



Figure 1.1: MARLO going up a steep hill at the University of Michigan. Image obtained from [4]

identified a shortcoming in Da's approach to G-HZD. To address the shortcoming required us to develop a new approach to designing Da's boundary manifold and, subsequently, building an invariant manifold and its accompanying control policy. With Covid in full swing, working with our French colleagues at Wandercraft became more challenging, and hence we turned to Cassie for an evaluation of our new "true" G-HZD control design methodology.

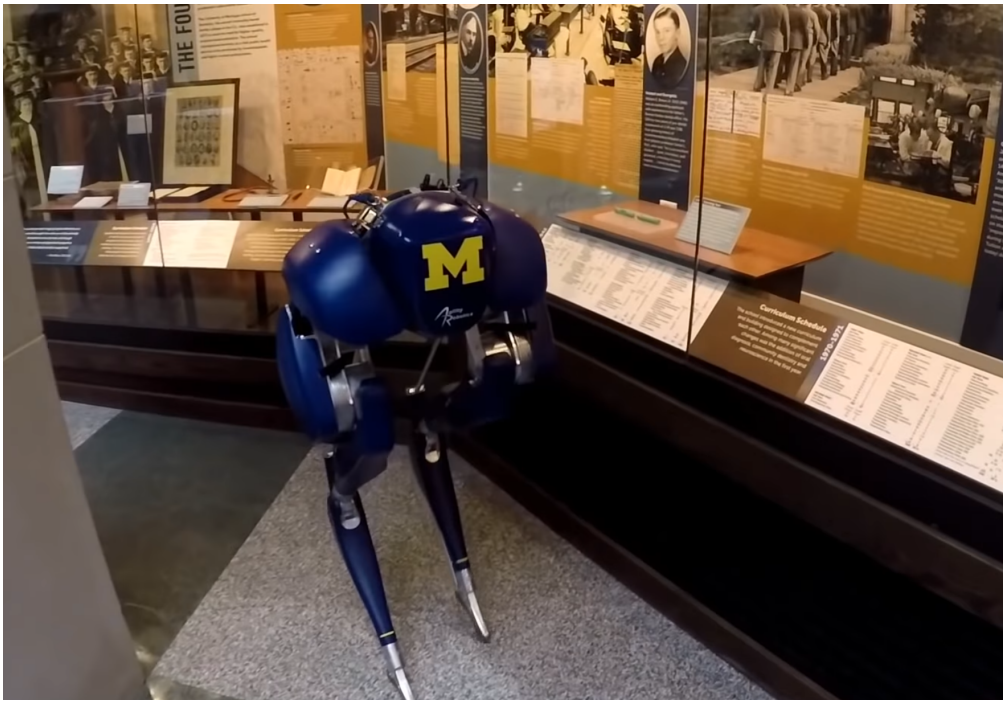


Figure 1.2: Cassie roaming around in the Dental School at the University of Michigan. Image obtained from [8].

1.2 Contributions of the Dissertation

Fast Computation of Optimal Trajectories for High-dimensional Legged Robot Models: The feedback control designs pursued in this dissertation for the Wandercraft Exoskeleton Atlante and for the Cassie biped robot are model based. The floating base model of Atlante has 21 DOF while Cassie model has 20 DOF. The feedback design for Atlante involves a gait library, which requires 7 optimal trajectories. When developing a gait library for Cassie for 11 different sagittal speeds, 11 different lateral speeds, and 11 different ground heights, we get 1331 trajectories that need to be computed. Of course, real-life design involves iteration, and thus the numbers listed above reflect just a few percent of the actual number of trajectories that needed to be computed. In light of this, a package called C-FROST was co-developed with Dr. Ayonga Hereid that expanded his MATLAB-based "Fast Robot Optimization and Simulation Toolkit" (FROST) by allowing it to run optimizations as a native C++ program. C-FROST exploits internal multithreading and ease of scaling to the cloud where you can take advantage of large numbers of cores. This work is published in [58] and the software has been made open-source [1].

Feedback Control of an Exoskeleton for Patients with Paraplegia: While others have

[1] `https://github.com/UMich-BipedLab/C-Frost`

used work from our lab on research other than biped robots, our lab has only focused on bipeds themselves. So the project I worked on was our first attempt at working directly with exoskeletons, which was an exciting venture. The objective is to develop control methods for an exoskeleton that allows patients with paraplegia to walk again without the use of crutches. The exoskeleton was designed and built by the startup, Wandercraft, and they made the decision to use Hybrid Zero Dynamics (HZD) control design methods developed by our lab. We have assisted them in gaining access to our most advanced methods. This work is published in [13, 51].

Systematic Design of the Boundary Conditions for G-ZHD: A powerful generalization of HZD was *initiated* by a previous PhD student, Xingye (Dennis) Da [29]. He called it G-ZHD for Generalized Hybrid Zero Dynamics. Dennis's foundational work left open an important aspect of the method, namely, how to select the boundary condition for the zero-dynamics manifold, and in some sense, even the proper dimension of the manifold. I address these important deficiencies by giving a systematic method to design the boundary conditions which underlay the trajectory optimizations that are fundamental to the method. In essence, my work transforms the general theory behind G-HZD into a practical design method. Initial testing of my ideas are presented on the classical inverted pendulum on a cart.

Implementation of the new G-ZHD approach on Cassie: Moving from theory to practice is always a challenge. I demonstrate the new G-ZHD approach on Cassie, an underactuated biped. I go over the steps of how to translate the approach from the pendulum example to a hybrid system. In addition, I go over the controller design and show experimental results. Work on experiments was significantly delayed due to COVID-19, lockdowns, and trying my best to stay safe.

1.3 Outline

The remainder of this document is organized as follows. Chapter 2 briefly introduces biped control concepts and summarizes key background material needed for the control approaches used in this dissertation. The chapter contains some sections previously published in [13, 51]. Chapter 3 presents the package C-FROST, a "turbo-charger" for the package FROST [55]. By allowing the optimizations in FROST to run as native C++ programs, where multithreading and cloud resources can be exploited, gait designs can be accelerated to where 1000+ gaits can be computed in under two hours on the cloud. The original work was reported in [58, 57]. With fast optimizations out of the way, Chapter 4 contains results previously published in [13, 51] where control approaches were tested on exoskeletons and techniques

like gait library and G-HZD were used. The chapter contains text and figures previously published in [13, 51]. Several extensions are made, however, most text and results remain unchanged. Chapter 5 shows a new approach to G-HZD. We cover limitations in the original G-HZD approach and how the new method overcomes those limitations allowing the method to work for multiple degrees of underactuation. Chapter 6 covers the Cassie implementation of the new G-HZD approach. We cover the optimization approach, the controller design, and include experimental results.

Appendix A describes how a direct collocation problem is formulated. The appendix contains text and figures previously published in [51]. Several extensions are made, however, most text and results remain unchanged.

Appendix B goes over PHZD. The appendix contains text and figures previously published in [13]. Several extensions are made, however, most text and results remain unchanged.

Appendix C contains a list of constraints used in Chapter 4. The appendix contains text and figures previously published in [13]. Several extensions are made, however, most text and results remain unchanged.

Appendix D contains a some equations used to linearize the hybrid model in Chapter 6.

CHAPTER 2

Background on Control Approaches

2.1 Motivation

Since the publication of the book [119], a lot has changed in the world of HZD. While Rabbit [26], MABEL [45], and MARLO [22] used pinned models for the Lagrangian dynamics, for Cassie and the Exoskeleton, we use floating-base models. Instead of the phase-based virtual constraints used in Rabbit [79], MABEL [45], and the initial work on MARLO [45] since the work in [30], the control objectives have become more complete, by including stepping in place for example, where there is no obvious monotonically increasing phase variable. Consequently, the virtual constraints have included a hybrid version of time.

This chapter *briefly summarizes* the key advances highlighted above that are used in this dissertation. No attempt is made to the repeat the details of the published results. *The intent is give a brief overview with references to the most important papers.*

2.2 Model

A two legged robot is modeled as a hybrid system with configuration variables q and inputs u. When one foot is in contact with the ground and one is above the ground, the floating-base equations of motion can be written as

$$D(q)\ddot{q} + H(q,\dot{q}) = Bu + J_c^T(q)F, \tag{2.1}$$

where $D(q)$ is the inertia matrix, $H(q,\dot{q}) = C(q,\dot{q})\dot{q}+G(q)$ is the vector containing the sum of the Coriolis and gravity term, B is the actuator distribution matrix, and $F : T\mathcal{Q}\times U \to \mathbb{R}^{n_c}$ is a vector of contact *wrenches* containing the contact constraint forces and/or moments (see [81]), where n_c is the number of holonomic constraints, and $u \in U$, is the control input. Let $c(q) \equiv 0$ be the condition that renders the holonomic constraint satisfied. J_c is the Jacobian

of the function $c(q)$ with respect to q. The contact wrenches, F, can be obtained by solving the second derivative of the holonomic constraints

$$J_c(q)\ddot{q} + \dot{J}_c(q,\dot{q})\dot{q} = 0 \tag{2.2}$$

and (2.1) simultaneously.

Under a rigid ground assumption, the swing foot's impact with the ground will be considered as plastic (coefficient of restitution is zero) and instantaneous (the impact forces and moments act over an infinitesimal interval of time) impact. During an impact, the generalized position variables of the system remain unchanged. However, the generalized velocities $\dot{q}$ undergo a discrete jump due to the instantaneous change in the generalized momentum. This is captured by a *reset map* Δ which represents the relationship between the pre-impact states x^- with the post-impact states x^+. Let $x = (q, \dot{q}) \in T\mathcal{Q}$ be the states of the system dynamics, where $T\mathcal{Q}$ is the tangent space of $\mathcal{Q}$, the hybrid system model of the flat-footed walking of the robot can be written as

$$\Sigma : \begin{cases} \dot{x} = f(x) + g(x)u, & x \notin S \\ x^+ = \Delta(x^-), & x \in S, \end{cases} \tag{2.3}$$

where S is the guard or switching surface which determines the specific condition (i.e., the swing foot impacting the ground) that triggers the discrete events and the vector fields f, g are from the continuous-time swing-phase dynamics in (2.1). See [46] for a thorough discussion on obtaining dynamical models for bipedal mechanical systems.

Remark 1. *When needed, a hybrid version of time is appended to the model via*

$$\begin{cases} \dot{\tau} = 1, & x \notin S \\ \tau^+ = 0, & x \in S. \end{cases} \tag{2.4}$$

As noted, the guard condition for re-setting the "hybrid time variable", τ, is determined by foot contact, as in the model (2.3).

2.3 Virtual Constraints and HZD

2.3.1 Virtual Constraints

At the core of this controller approach is designing a set of outputs, $y(x)$, known as virtual constraints, such that the zeroing of these outputs leads to the desired robot behavior.

Virtual constraints are functions of the state variables that define how the various links of the robot should be coordinated in order to elicit a desired motion, such as walking. The virtual constraints are imposed by driving them to zero via feedback control rather than through mechanical linkages, and hence the name "virtual" constraints.

Virtual constraints are formulated as the difference between the actual value, y^a, of an important physical quantity, such as leg angle, and its desired value, y^d, as in

$$y = y^a(q) - y^d(\tau, \alpha) = 0, \tag{2.5}$$

where our virtual constraints are relative degree 2, τ is a monotonically increasing gait timing variable, and α is a set of constants defining the trajectory. The controller then is designed to enforce the virtual constraints by driving the outputs y asymptotically to zero. τ can either a function of time, or a function of q such that it is monotonically increasing during a step.

The controller is designed using input-output linearization such that the output dynamics are

$$\ddot{y} + \dot{y}k_d + yk_p = 0, \tag{2.6}$$

where $k_d, k_p > 0$ are tuned by the controller designer. The decoupling matrix,

$$L_g L_f y(q, \dot{q}, \tau, \alpha),$$

must be invertible for the controller to be well defined. Because τ is included in the virtual constraints, it is assumed to be appended to both f and g in the dynamic model, as noted in (2.4).

The set of states where the virtual constraints and holonomic constraints are enforced is called the zero dynamics manifold, i.e.

$$\mathcal{Z} := \{x \in \mathbb{R}^{2n+1} \mid y = 0, \dot{y} = 0\}, \tag{2.7}$$

and the reduced-order dynamics restricted to this set is called the zero dynamics. We note that both Z and the restriction dynamics depend on "time" through τ.

The chosen outputs need not be simply the actuated states. In fact, the dynamical properties of the zero dynamics change and depend on the choice of outputs [27]. For underactuated robots like MARLO in Figure 2.1 , which has 6 actuators, a set of virtual constraints could be [30]:

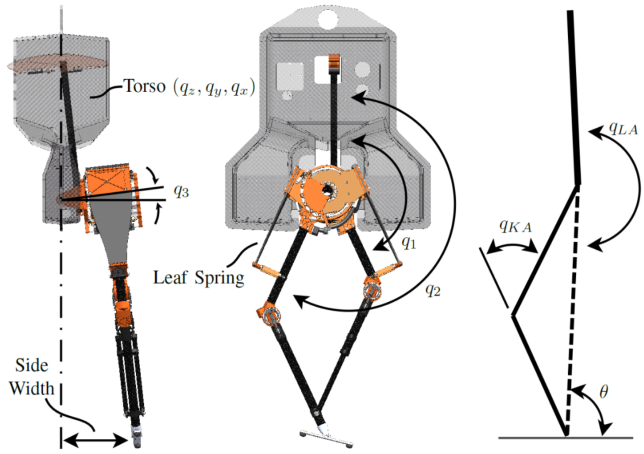


Figure 2.1: Showing virtual constraints choices on MARLO]. q_{KA} refers to the "knee angle". q_{LA} refers to the "leg angle". Figure obtained from [30, Fig. 3].

- Torso Pitch
- Torso Roll
- Stance Knee Angle
- Swing Hip Angle (sideways motion)
- Swing Leg Angle (forward/backward motion)
- Swing Knee Angle

2.3.2 Computer Optimization

Trajectories that define the virtual constraints are obtained via offline trajectory optimizations. We attempt to find a periodic gait that satisfies various constraints while minimizing the cost

$$\mathcal{L} = \int_0^T \|u(t)\|^2 dt, \tag{2.8}$$

where T is the time period of one step. $\|u(t)\|^2$ is treated as a "pseudo energy" cost of the system.

The constraints on the optimization problem are as follows:

- Dynamic feasibility; trajectory must satisfy the dynamics equations
- Joint and torque limits must be satisfied
- Fixed time step
- Swing foot must have a large enough foot clearance during midstep, eg. 10 cm
- Friction cone and ZMP conditions must be satisfied

- Swing foot should have zero horizontal speeds
- Gait must be periodic
- Posture of the robot might be constrained, eg. torso remains upright

We typically use FROST [55] for gait generation which formulates the problem as a direct collocation problem. FROST and trajectory optimization are discussed later in Chapter 3. A more in depth discussion about direct collocation is given in Appendix A.

2.3.3 Stability with One Degree of Underactuation

Assume we have a planar biped with one degree of underactuation. In addition, assume we have a periodic orbit $\mathcal{O}$ where the robot is walking forward. We can extract a function $\tau(q)$ that monotonically increases during a step when following $\mathcal{O}$. We'll choose a set of virtual constraints parameterized by τ such that

$$y = y_a(q) - y_d(\tau, \alpha),$$

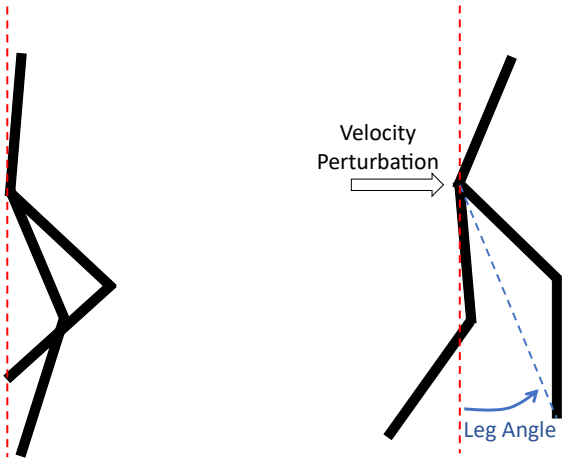
where y_d is trivially chosen to be the actuated coordinates. From [119, Theorem 6.2], we can show that periodic orbit $\mathcal{O}$ is exponentially stable if the vertical component of the center of mass velocity is negative at the end of the step.

2.3.4 Foot Placement as Means of Achieving Stability with Greater Underactuation

With one degree of underactuation, we can prove stability under certain conditions. But for higher degrees of underactuation, implementing the simple virtual constraints discussed earlier often to not yield stable zero dynamics. So extra effort is needed to stabilize the system. One technique is using foot placement. In simple terms, the swing foot position is adjusted to catch the robot from falling depending on how the robot is falling. Figure 2.2 shows an illustration on the robot falling forward and catching itself using foot placement. A simple foot placement policy can be accomplished by estimating the horizontal velocity of the center of mass, then adjust the desired swing leg angle appropriately, such that:

$$y^d_{sw,LA} = \bar{y}^d_{sw,LA}(\tau, \alpha) + k_{fp}(v_{com} - \bar{v}_{com}(\tau)) \tag{2.9}$$

where $y^d_{sw,LA}$ is the desired swing leg angle, $\bar{y}^d_{sw,LA}$ is the nominal trajectory, k_{fp} is a tunable gain, v_{com} is the veclotiy of the center of mass, and $\bar{v}_{com}$ is the expected velocity.



(a) In-place (b) Falling

Figure 2.2: Depicts example of a biped walking in-place versus falling forward and having to catch itself using foot placement. (b) has a forward velocity.

2.4 Gait Library

In this approach, using computer optimizations, we first generate a set of periodic trajectories for different walking speeds. We typically employ the FROST package [55] for this purpose.

Define $A := \{\alpha(v_i) | 1 \leq i \leq n\}$ to be a set of n trajectories generated for different periodic speeds v_i. Next, by interpolating between the gaits, we're able to have a continuous set of gaits for different velocities from v_1 to v_n.

Then, by choosing the appropriate gait based on the current velocity, either on a per step basis [30, 13] or continuously during a step, we obtain a neutrally stable controller. In this situation, disturbances to the robot's velocity will persist, similar to an integrator with an eigenvalue of one. Due to the neutral stability, asymptotic stabilization to an equilibrium gait is easier, such as when using the foot placement approach. Figure 2.3 shows the controller structure.

The combination of gait interpolation (yielding neutral stability) and a foot placement controller results in an increased basin of attraction beyond what foot placement alone can accomplish with a fixed gait. An analysis and comparison has been done in [30].

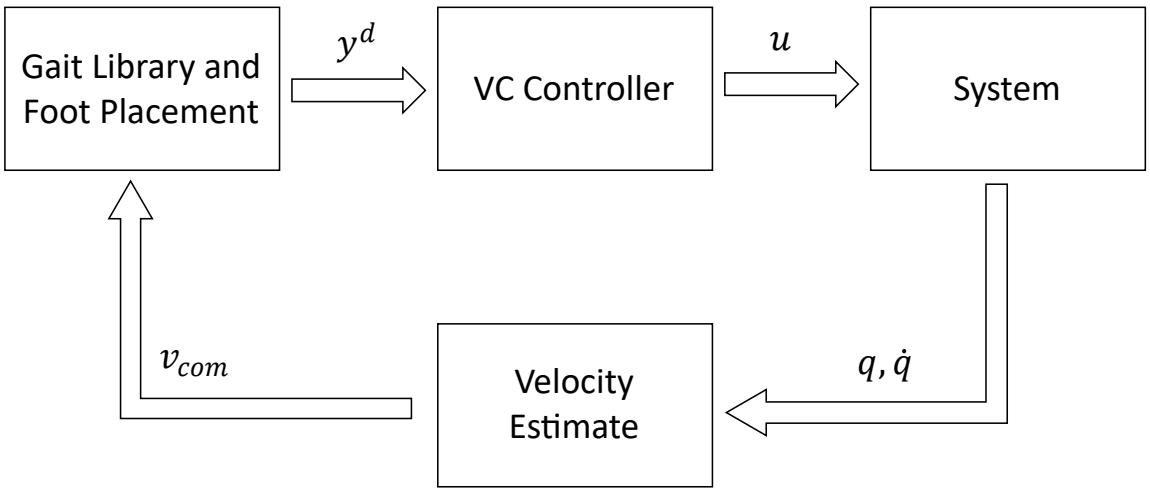


Figure 2.3: Example of a controller design using a gait library approach.

2.5 Background into G-HZD

The gait library method used a heuristic controller to achieve stability. With G-HZD, introduced in [29], we wish to encode disturbance rejecting behavior within a family of trajectories and subsequently design a control policy capable of utilizing such trajectories.

Let's focus on stabilizing a periodic trajectory, x_n. We seek to create a low-dimensional hybrid-invariant surface that is comprised of trajectories that contract toward the periodic trajectory. The contraction property of the trajectories will lead to local exponential stability on the surface. If the surface can also be rendered sufficiently rapidly exponentially attractive, then we can use results in [16] to conclude that the periodic orbit is locally exponentially stabilizable in the full-order dynamics. We know how to render a surface locally exponentially attractive when its transverse dynamics are input-output linearizable.

These ideas set the stage for the G-HZD approach. (a) The trajectories used to build surface are found from a two-point value problem: in order to have hybrid invariance, we need trajectories of the hybrid model that start on a given subset and terminate on the same subset. Here, that subset will be denoted as Z_0, and called the boundary of the invariant surface. (b) We want to therefore design the boundary manifold Z_0 is such a way that the dynamics transverse to Z will be feedback linearizable. The suggestion in [29] is to decompose the system's state variables, x, into weakly actuated states x_w and strongly actuated states x_s and to write

$$Z_0 := \{x =: (x_w; x_s) | x_s = \gamma_0(x_w)\}, \tag{2.10}$$

for a given function γ_0. One of the drawbacks of [29] that we highlight in this dissertation—is that Da does not provide a rigorous method for choosing γ_0. We will remove this drawback

in Chapter 5. For the time being, we follow [29] and define γ_0 by a surface of periodic gaits, as in the gait library method, with the x-y hip velocities of the robot taken as the weakly actuated states.

To "fill out" the surface Z, we run optimizations that start from initial states in Z_0 and end in Z_0 in one step of the robot. Attractivity can be ensured in two ways: (i), by choosing a Lyapunov function and imposing that its value at the end of the step is less than its initial value; and (ii), inspired by finite-horizon control (e.g. MPC), the optimization is extended over three steps and one imposes that the periodic orbit is exactly reached at the end of the third step. Figure 2.4 provides a graphical illustration of the optimization process. The trajectories form a surface Z seen in Figure 2.5a. The dimension of Z is that of Z_0 plus one (for time).

The next step is to construct a vector field from the open loop trajectories defining Z. For perturbations during a step, the vector field can be thought of as providing instantaneous re-planning, guiding the system back to the desired periodic orbit, as seen in Figure 2.5b. We use supervised machine learning to "extract" the vector field from the open-loop trajectories by regressing the strongly actuated states x_s against the weakly actuated states $x_w$and time t. If successful, we end up with a mapping of the form

$$x_s = \nu(x_w, t). \tag{2.11}$$

A more detailed description is given in Chapter 4, based on [51] with an example on an exoskeleton.

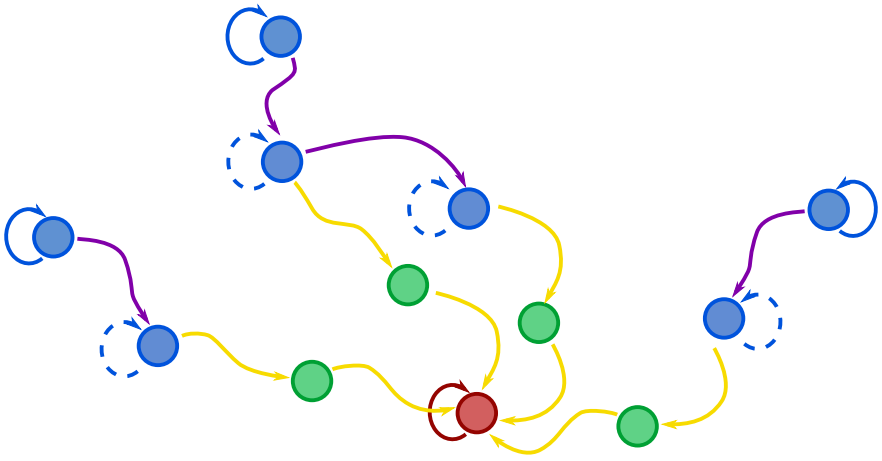
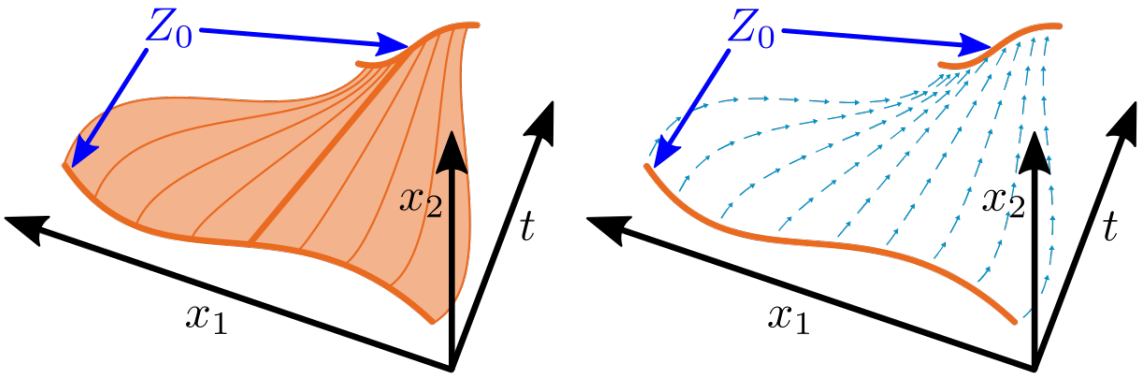


Figure 2.4: Graphical representation of gait optimization for G-HZD as developed in [29]. Each arc represents one step of the robot. Circles with self-loops denote periodic orbits, while those without are transient states. The red circle denotes the goal (periodic) gait. The optimizations are done over three steps, with the end of the first step required to terminate in Z_0, which is parameterized by periodic gaits, and the end of the third step must be the goal state. Since the first step initiates and ends in Z_0, the three-step optimization can continue from the end of a first step and other self-loop-states in the same fashion as described above. The (first-step) trajectories corresponding to the time interval $[0, T_p]$—denoted by the purple arcs—are saved and used as training data for the Supervised Machine Learning. The other data is discarded.



(a) Open Loop Z

(b) Vector Field Z

Figure 2.5: Representations of Z. In (a), surface is a collection of open-loop trajectories that converge to a periodic solution, shown in bold. In (b), we seek a vector field whose solutions are the open-loop trajectories forming Z. We use Machine Learning to extract the vector field from the trajectories, if it exists.

CHAPTER 3

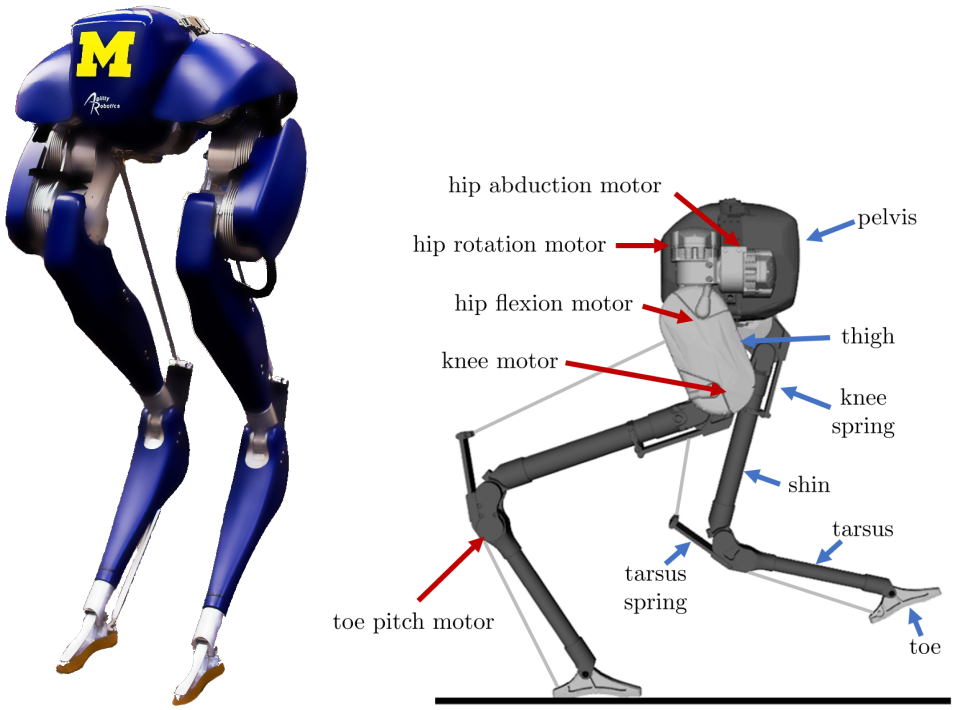
Speeding up Optimization

3.1 Motivation

Low-dimensional inverted pendulum models are by far the most common approach in the literature to getting around the analytic and computational challenges imposed by today's humanoid robots [65, 86]. In this approach, one maps a locomotion task onto the motion of the center of mass of a linear inverted pendulum (LIP), a spring-loaded inverted pendulum (SLIP), or other simplified models [75] and equips the model with a "foot-placement strategy" for stability [66, 90, 87]. The overall simplicity of this approach is one of its uncontested advantages. On the other hand, when designing agile motions, guaranteeing stability in the full-order model and exploiting the full capability of the machine, especially in light of physical constraints of the hardware or environment, are among its main drawbacks.

Because of these drawbacks, numerous groups are pursuing systematic approaches to utilize a high-dimensional mathematical model of a legged robot when designing dynamic behaviors and also when designing controllers that are sufficiently robust to realize these behaviors on the physical hardware. Speaking first to dynamic gait design, there is an increasing trend in the development of optimization-based planning algorithms which directly utilize more complicated dynamic models [23, 32, 59, 74]. These applications typically use general-purpose nonlinear programming solvers, e.g., IPOPT [115] or SNOPT [38], to formulate the motion planning problems. Others are advocating state-of-the-art optimal control toolboxes, such as GPOPS [84], DIRCOL [111] or PSOPT [20], that come with advanced trajectory optimization algorithms. These latter packages, however, either lack an effective framework for users to construct the robot model or are not capable of solving large-scale problems that are very common for today's agile walking robots.

More recently, several open-source robotic toolboxes, including DRAKE [106] and FROST [55], have been developed to provide an integrated development environment for complex robotic systems. Both DRAKE and FROST are capable of optimizing dynamic walking



(a) Cassie Blue (b) Cassie's Coordinates

Figure 3.1: (a) Shows Cassie Blue, one of the Michigan copies of the Cassie series of robots built by Agility Robotics. (b) Shows the joints and the kinematic model of the robot.

gaits for high degree of freedom (DOF) robots using the full-order dynamics of the models. While DRAKE is implemented in a C++ environment for efficiency, FROST is developed as a MATLAB package for rapid prototyping. Compared to DRAKE, which uses automatic differentiation for computing gradient information for the NLP solvers, FROST takes advantage of the symbolic computation to generate the sparsity structure of the trajectory optimization problem. To speed up the computational speed for high-dimensional robotic systems, FROST is equipped with a custom symbolic math toolbox based on the Wolfram Mathematica kernel as the back-end to generate optimized C++ code for computing system dynamics and kinematics functions. Moreover, FROST is capable of computing the analytic derivatives of optimization constraints and cost functions symbolically, and *even more importantly, it can precisely determine the sparsity structure of the nonlinear trajectory optimization problem.* With these features, FROST generates dynamic biped walking gaits using the full-order dynamics for a full-size humanoid robot (e.g., DURUS) in less than 10 minutes [54]. Since its inception, FROST has been used for fast dynamic gait generation of various bipedal robots [51, 2, 92]. Yet, its performance is significantly affected by the computational overhead introduced by MATLAB. This becomes evident particularly in the case when many different gait trajectories are required to be designed [29].

In this section, we present C-FROST, an open-source package that converts the direct collocation trajectory optimization problem constructed in FROST to a more computationally efficient C++ code. By reducing the computational overhead of MATLAB functions, we are able to increase the speed over previous full-order dynamic optimization problems by a factor of six to eight for a single gait. The implementation of C-FROST also makes it easier to deploy parallel computation on cloud servers for the generation of a large set of different gaits, a feature that allows the gait library approach presented in [29] to be practical on high-dimensional 3D bipedal robots and advanced exoskeletons [51]. Beyond generating families of gaits for various walking speeds, directions, and terrain slopes, the ability to generate gaits in parallel for a family of models is also useful for iterative design of legged mechanisms [26, 92].

The remainder of the chapter is organized as follows. Section 3.1.1 overviews the FROST package and the optimization problem being solved here for gait design. Section 3.2 introduces a stand-alone open-source C++ package that carries out the optimization problem formulated in FROST. Section 3.3 presents a case study of the application of C-FROST for a Cassie robot, and its execution speeds are illustrated in Section 3.4 in terms of: (1) optimization in FROST vs. C-FROST, (2) serial execution vs. parallel, and (3) desktop vs. the cloud. Section 3.4 also demonstrates that the gait designs performed in the chapter are viable on a Cassie robot. Conclusions are given in Section 3.5.

3.1.1 FROST

FROST (Fast Robot Optimization and Simulation Toolkit)a is an open source MATLAB toolbox that provides an integrated development framework for mathematical modeling, trajectory optimization, model-based control design, and simulation of complex robotic systems. While initially developed particularly for bipedal robots, FROST has been applied to a wide range of robotic systems, including quadrupeds [49], robot manipulators [70], and the adaptive lane keeping control design of an automated truck model [25]. The most important feature of FROST is the fast trajectory optimization algorithm for high-dimensional hybrid dynamical systems using direct collocation approaches [55].

3.1.2 Robot Behavior Modeling

In FROST, a particular behavior of a robot can be modeled as a hybrid system that exhibits both continuous and discrete dynamics. For instance, legged locomotion often consists of a collection of continuous phases, with discrete events triggering transitions between neighboring phases [46].

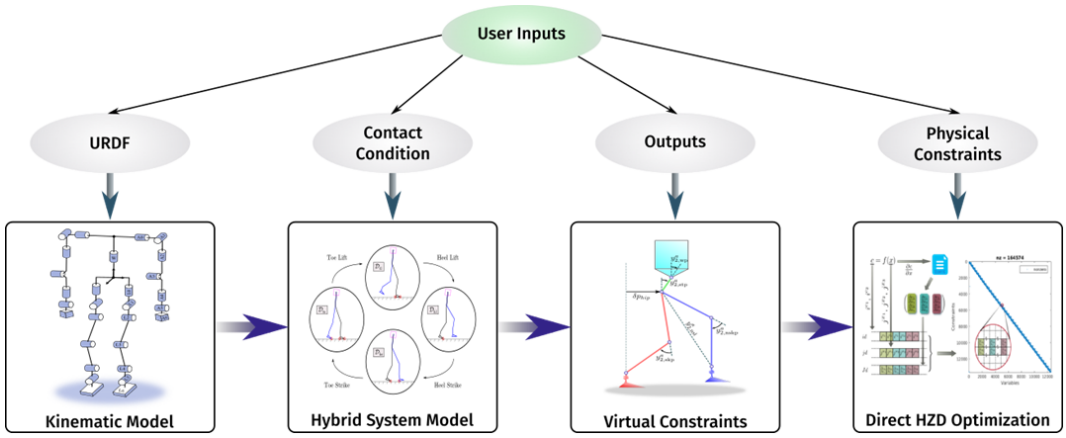


Figure 3.2: The process of modeling and optimizing dynamic walking gaits in FROST. Given a robot model (e.g., via a URDF file), FROST makes it easy for users to define the desired behavior and provide the necessary physical constraints to formulate a realistic gait optimization problem.

Definition 1. *A* hybrid system *is a tuple,*

$$\mathscr{HC} = (\Gamma, \mathcal{D}, \mathcal{U}, S, \Delta, \mathrm{FG}), \tag{3.1}$$

where $\Gamma = \{V, E\}$ *is a* directed graph, $\mathcal{D}$ *is the admissible configuration of the continuous domains,* $\mathcal{U}$ *represents the admissible controllers,* S *determines guard conditions that trigger discrete transitions, and* Δ *and* FG *represent the discrete and continuous dynamics respectively [55].*

In a hybrid system model, each vertex represents an admissible continuous domain determined by the differential equation of motion and physical constraints of the robot. These constraints include the contacts or mechanical constraints such as four-bar links or contacts with the environment, which modeled as *holonomic constraints*, and associated limiting conditions such as the friction cone or swing foot clearance conditions, which can be formulated as a set of *unilateral constraints*. A *holonomic constraint* establish a set of algebraic equations to the continuous system dynamics. Moreover, for each edge in the graph, a *unilateral constraint* determines the guard condition that triggers the associated discrete event. For example, in legged locomotion, the robot switches to different continuous domains when it establishes new contacts or breaks existing contacts. FROST describes the discrete behaviors of a robot by defining these kinematic and dynamic elements of the robot in the framework of a hybrid system model given in (3.1).

Continuous Dynamics. The continuous domain describe the evolution of system states governed by differential algebraic equations (DAEs) on a smooth manifold. Let $x \in \mathcal{Q}$ be

a set of coordinates of the system, the continuous dynamics is governed by the following ordinary differential equations (ODEs):

$$M(x)\ddot{x} = F(x, \dot{x}) + G(x, u) \tag{3.2}$$

subject to a set of algebaric equations $h(x) = 0$, where $M(x)$ is the positive definite mass matrix, $F(x, \dot{x})$ is a set of drift vectors, and $G(x, u)$ is a set of input vectors with u being the system inputs. FROST also allows to use a first-order ODEs to describe the system equations of motion [55, 25].

Discrete Dynamics. When the evolution of a system transitions from one continuous phase into another, its states often undergo an instantaneous discrete change. It can be given expressed in the following manner:

$$x^+ = \Delta_x(x^-), \tag{3.3}$$

where $x^- = \lim_{t \nearrow t_0} x(t)$ and $x^+ = \lim_{t \searrow t_0} x(t)$ with t_0 be the time instant at which the discrete dynamics occurs, and Δ_x represents a reset map at transition. For a second-order system, the first order derivative of x should also be considered, i.e.,

$$\dot{x}^+ = \Delta_{\dot{x}}(x^-)\dot{x}^-, \tag{3.4}$$

where $\dot{x}^- = \lim_{t \nearrow t_0} \dot{x}(t)$ and $\dot{x}^+ = \lim_{t \searrow t_0} \dot{x}(t)$.

Remark 2. *Given a robot's kinematic structure and inertial parameters, FROST provides many useful functions to easily compute the forward kinematics and dynamics of the system [55]. A typical process in FROST for the modeling and gait design of a bipedal robot is shown in Figure 3.2.*

3.1.3 Trajectory Optimization

The key functionality of FROST is to provide a fast and scalable trajectory planning algorithm for the hybrid dynamical systems. Given a hybrid dynamical system model in (3.1), FROST automatically constructs a multi-phase hybrid trajectory optimization problem [54]. In particular, it uses direct collocation methods to solve such a trajectory optimization problem. FROST supports various types of direct collocation methods, including local collocation methods that use Hermite-Simpson or Trapezoidal schemes and global collocation methods that collocate at Legendre-Gauss-Lobatto (LGL) points. For the mathematical formulation of different collocation methods, we refer [54] to the readers for more detail. The result of

such a trajectory optimization problem is a large-scale sparse nonlinear programming (NLP) problem. In general, it can be stated as:

$$\arg\min \sum_{j=1}^{P} \left(\sum_{i=1}^{N_j} w_i \mathcal{L}_j(\cdot) + E_j(\cdot) \right) \tag{3.5}$$

$$\begin{aligned}
\text{s.t} \quad & M_j(x_i)\ddot{x}_i = F_j(x_i, \dot{x}_i) + G_j(x_i, u_i), && \text{(dynamics)} \\
& \delta_j(x_0, \ldots, x_i, \dot{x}_i, \ddot{x}_i, \ldots, \ddot{x}_{N_j}) = 0, && \text{(collocation)} \\
& (x_0^{j+1}, \dot{x}_0^{j+1}) = \Delta_j(x_{N_j}^j, \dot{x}_{N_j}^j), && \text{(reset map)} \\
& C_j(x_i, \dot{x}_i, u_i) \geq 0 && \text{(path constraints)}
\end{aligned}$$

where $\mathcal{L}_j(\cdot)$ represents the running cost and $E_j(\cdot)$ represents the terminal cost defined in the domain $j \in V$ which has P vertices, δ is the collocation constraints, Δ_j is the reset map associated with domain $(j, j+1) \in E$, and $C(\cdot)$ is the collection of physical constraints, such as foot clearance, joint angle/velocity limits and torque limits, etc.

3.2 C-FROST Implementation

This section presents C-FROST[1], an open source package that allows the user to convert and run the trajectory optimization problem in a native C++ environment instead of MATLAB. By reducing the computational overhead in the MATLAB environment, C-FROST noticeably speeds up the computational performances. It also comes with a few added benefits such as:

- using multi-threading techniques to speed up the evaluation of function Jacobians; and
- making it easier to execute multiple optimization problems in parallel on multi-core processors; and
- allowing easier deployment of parallel optimization to remote cloud services.

3.2.1 Formulation of NLP Problems

The use of direct collocation converts all constraints as algebraic constraints. While in more traditional trajectory optimization or optimal control problem, the state trajectories must be obtained by integrating the differential equations of motion. This enables us to construct the optimization problem as a general purpose nonlinear programming (NLP) problem, stated

[1]C-FROST is publically available at `https://github.com/UMich-BipedLab/C-Frost`.

as:

$$
\begin{aligned}
X^* = \arg\min_{X} \quad & \sum_{i=1}^{N_o} \mathcal{L}_i(X_i) \\
\text{s.t} \quad & \mathcal{C}_j^l \leq \mathcal{C}_j(X_j) \leq \mathcal{C}_j^u \quad \forall j \in [1, \ldots, N_c], \\
& X^l \leq X \leq X^u,
\end{aligned} \tag{3.6}
$$

wher $X \in R^{N_x}$ with N_x being the total number of optimization variables, N_o is the total number of cost functions with each cost function $\mathcal{L}_i$ has dependent variables X_i, where X_i is a collection of components of X, N_c is the total number of constraint functions, each constraint $\mathcal{C}_j$ has dependent variables X_j (again, X_j is also a collection of components of X), and $\mathcal{C}_j^l$ and $\mathcal{C}_j^u$, and X^l and X^u, respectively, are the lower and upper bounds of $\mathcal{C}_j$ and X. We also denote $\mathcal{C} = [\mathcal{C}_1; \ldots; \mathcal{C}_{N_c}]$ as the vector of all constraints, and $\mathcal{C}^l$ and $\mathcal{C}^u$ as the lower and upper bound of $\mathcal{C}$, respectively.

A particular advantage of using FROST is its capability to compute the closed-form symbolic expressions of all constraints and cost functions and export them to optimized C++ codes for fast computation. This is realized by the custom designed MATLAB symbolic math toolbox in FROST powered by the Wolfram Mathematica kernel as the backend. While formulating the problem in (3.6), we introduce additional *defect variables* to simplify the constraints expressions in FROST. That enables FROST to compute the analytic derivatives (a.k.a., gradient or Jacobian matrices) of constraint and cost functions with respect to the corresponding dependent variables, as well as to determine the non-zero entries in the derivatives. In C-FROST, we generate two functions to compute the sparse Jacobian matrices: one that returns the row and column indices of all non-zero entries, and another one computes the values of these non-zero entries. Using these two functions, one can formulate the sparse Jacobian matrix of a constraint or the gradient of the cost function.

3.2.2 Execution of NLP Problems

C-FROST contains two main components: a collection of MATLAB functions that convert and export the original optimization problem in MATLAB to a set of C++ functions and JSON configuration files; and a C++ program that uses these exported functions and configuration files to run the optimization on a native Linux environment by calling the IPOPT solver directly.

Exporting the NLP Problem. To export an NLP problem to the C-FROST format, we extract the structure of the problem in (3.6) and export this information to a collection of

Algorithm 1 Evaluation of the NLP Constraints

procedure EVALFUNC(X, `Constraints`) xInitialize $\mathcal{C} \leftarrow 0$
  for all $f_j \in$ `Constraints` **do** x$x_j \leftarrow$ the dependent variables of f_j from X x`aux` $\leftarrow$ the auxiliary constants of f_j x$\mathcal{C}_j \leftarrow$ the evaluation of $f_j(x_j;$ `aux`$)$ x$r_j \leftarrow$ the indices of $\mathcal{C}_j$ values in $\mathcal{C}$ x$\mathcal{C}(r_j) \leftarrow \mathcal{C}_j$
  end forx**return** $\mathcal{C}$
end procedure

Algorithm 2 Evaluation of the sparse Jacobian matrix of NLP Constraints

procedure EVALJAC(X, `Constraints`) xInitialize $G_{nz} \leftarrow 0$, $i_R \leftarrow 0$, $j_C \leftarrow 0$
  for all $f_j \in$ `Constraints` **do** x$x_j \leftarrow$ the dependent variables of f_j from X x`aux` $\leftarrow$ the auxiliary constants of f_j x$J \leftarrow$ the evaluation of $G_j(x_j;$ `aux`$)$ x$(i_r, j_c) \leftarrow$ the evaluation of $Gs_j(x_j;$ `aux`$)$ x$r_j \leftarrow$ the indices of J values in G_{nz} x$G_{nz}(r_j) \leftarrow J_i$, $i_R(r_j) \leftarrow i_r$, $j_R(r_j) \leftarrow j_c$
  end forx**return** sparse(G_{nz}, i_R, j_C) ▷ Construct the sparse matrix
end procedure

JSON files. For each constraint, the following information is required:

- the associated C++ functions, f_j to compute the constraint or cost function, G_j to compute the non-zeros in the Jacobian matrix, and Gs_j to return the indices of non-zero terms;
- the indices of dependent variables X_j in X;
- the indices of constraints $\mathcal{C}_j$ in $\mathcal{C}$;
- the auxiliary constant data `aux` required to compute f_j;
- the indices of non-zero Jacobian entries G_j in G_{nz}.

The above information is constructed as arrays of structured data, `Constraints`, and exported to a JSON file. The same information for the cost functions is also formulated and exported to the same JSON file. We also generate two additional configuration files with one storing the boundary values of constraints and optimization variables and another storing the initial guess for the NLP solver. The C++ functions f_j, G_j and Gs_j are generated automatically using the symbolic math toolbox in FROST.

Evaluating the NLP Problem. Using the generated C++ functions, C-FROST will create a C++ program that calls the interior-point-based NLP solver IPOPT as a shared object to solve the direct collocation optimization problem. The program will first read the exported configuration files to construct a problem for IPOPT, and then call the IPOPT optimization routine, which requires sub-routine functions that compute the cost function and its gradient,

the constraints, and the sparse Jacobian of the constraints. In Procedure 1, we describe how the constraints are computed. The same procedure applies to the cost function. Procedure 2 illustrates the construction of the sparse Jacobian matrix for constraints, using the generated C++ functions and other indexing information. Again, the same procedure is used for calculating the gradient of the cost function. It is important to note that we will only need to compute the non-zero entries in the Jacobian matrix. Considering that the percentage of non-zero entries in the Jacobian of a FROST-generated NLP problem is often far less than 1%, this significantly reduces the overall computational load.

3.2.3 Parallel Evaluation of NLP Problems

To speed up the optimization further, we consider two types of parallelization in FROST. For a single gait optimization, C-FROST provides multi-thread parallelization for the evaluation of Jacobian, whereas to run multiple optimization problems together, we prefer to run them in parallel on a multiple CPU cores platform.

Subroutine Parallelization. The evaluations of the Jacobian of some constraints might be computationally expensive, like in the case of the dynamics equation. Therefore there are benefits to parallelization when evaluating such matrices. The subroutines of each constraint within the for-loop in Procedure 2 are independent of other constraints; hence they can be run in parallel. C-Frost allows the different constraint Jacobians to be evaluated simultaneously by multi-threading Procedure 2. For each constraint f_j, lines 4 till 9 in Procedure 2 are evaluated in available workers. To avoid concurrent manipulation of the shared data G_{nx}, i_R and j_C, the evaluations in line 9 is locked using a monitor class to synchronize access to the shared data. The evaluation of the constraint Jacobian is significantly more computationally expensive than the evaluations of the constraint functions, the cost functions and the gradient of the cost function. And thus, these processes have not been parallelized.

Parallel Optimization. When having multiple optimization problems to be solved, as in the application of generating a library of gaits, running the multiple optimization problems simultaneously is preferred to using multiple threads per problem as that avoids the overhead from thread synchronization in a single program. This can be done by writing a simple bash script that starts new optimizations when a CPU is underutilized or existing third-party tools, such as the GNU Parallel package [105].

The ability to run many different gait optimization in parallel is particularly useful for learning-based control approaches as described in [29, 30], and can also be used in the mechanical design process of a robot, where instead of gait characteristics varying, link lengths, mass distributions, and gear ratios are varied [117, 92].

3.3 A Case Study: Cassie Blue

The efficiency of C-FROST will be illustrated by designing a library of different periodic walking gaits for the Cassie-series bipedal robot shown in Figure 3.1. Applications on other platforms can be found from the official C-FROST GitHub repository.

3.3.1 Testbed Robot Model

This underactuated robot's floating base model has 20 degrees of freedom (DOF). There are seven joints in each leg with five of them actuated by electric motors through gearing and the other two joints being passive where they are realized via a specially designed four-bar linkage with one bar being a leaf spring. When supported on one foot during walking, the robot is underactuated due to the narrow feet. The springs in the four-bar linkages also introduce additional underactuation. For simplicity, we assume these springs have infinite stiffness, i.e, rigid links, in this chapter (see Figure 3.1b). The description of the mechanical model of Cassie used in this chapter is publicly released on GitHub[2].

In this chapter, we will use FROST to model and formulate a series of periodic walking gait optimization problems for the Cassie, and then use C-FROST to generate these gaits in parallel rapidly. The optimization performance will be evaluated on multiple platforms—including remote cloud services—with different numbers of CPU cores. While the present chapter is concerned with optimization for the open-loop model of Cassie, some of the designed gaits are implemented to demonstrate that they are physically realistic, and for that, a control policy is necessary. The control laws used in the simulations and experiments reported here are based on the work in [41]. To enhance our confidence that the presented optimization results are representative of what other users will find on their models, we have also run several of them on an 18-DOF exoskeleton model presented in [51]. These results are not reported here.

3.3.2 Optimization Problem

The primary problem being illustrated in this chapter is to design two-step periodic walking gaits for Cassie Blue. This walking pattern is modeled as a two-domain (a right-stance domain and a left-stance domain) cyclic hybrid system model in FROST. A virtual constraints based feedback controller is also enforced in the gait optimization problem so that the results are not just the open-loop join trajectories but also a class of feedback controllers that can be used to stabilize the periodic motion in the full-order robot dynamics [54].

[2]Available at `https://github.com/UMich-BipedLab/Cassie_Model`.

The optimization problems are formulated in FROST using the structure described in (3.6). When formulating the gait optimization problems, we impose periodicity over two steps for a desired average walking speed and ground inclination. The cost function to be minimized in the optimization is the "pseudo energy", given as

$$\mathcal{L}_j = \|u_i(t)\|^2, \tag{3.7}$$

where $\mathcal{L}_j$ is the integral function for the running cost defined at the domain j, with $j \in \{1, 2\}$ being the index of the right and left stance domain and u_i being the control inputs at each collocation node. The "pseudo energy" in (3.7) is not normalized by step length because the speed and step duration are constrained. In addition, the optimization also captures torque and joint limits of the Cassie robot as well as the following physical constraints that are heuristically chosen to achieve stable walking. These constraints are:

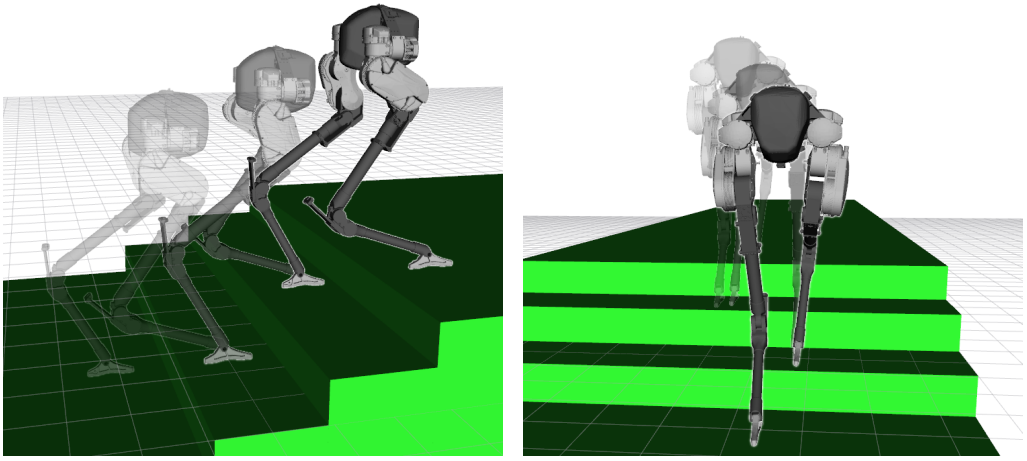
- Fixed step duration T of 0.4 s;
- Swing foot clearance of at least 15 cm at midstep;
- Ground reaction forces respect the friction cone and ZMP condition [46];
- Zero swing foot horizontal speeds at impact;
- For purely sagittal plane walking, require the two steps in a gait to be symmetric;
- Sagittal plane walking gaits have tighter bounds on hip abduction and rotation motors than gaits that include lateral motion; and
- Torso must remain upright within ± 3 degrees pitch and roll limits.

In particular, we use C-FROST to design a library of $11^3 = 1,331$ periodic gaits defined on a cubic uniform grid with different walking speeds and ground inclinations:

- 11 different backward/forward walking speeds ranging from -1 m/s to 1 m/s;
- 11 different left/right walking speeds ranging from -0.3 m/s to 0.3 m/s; and
- 11 different inclinations of the ground surface from -15 cm to 15 cm per step.

The desired average walking speed is enforced via an equality constraint

$$\bar{v}_x = \frac{p_N^x - p_0^x}{T}, \quad \bar{v}_y = \frac{p_N^y - p_0^y}{T} \tag{3.8}$$



(a) 15 cm step up while walking forward at 1 m/s.

(b) 15 cm step down while walking forward at 1 m/s and left at 0.3 m/s.

Figure 3.3: Stereoscopic tiles for stepping up/down at different walking speeds.

where p_N^x, p_0^x, p_N^y, and p_0^y are the (X,Y) position of the hip at the end and beginning of a step, respectively, and $\bar{v}_x$ and $\bar{v}_y$ are the desired sagittal and lateral walking speeds. The desired ground inclination will be enforced by changing the height where the swing foot should impact with the ground. Two of the gaits from this set are illustrated in Figure 3.3.

3.4 Results

This section first presents the computational performance for creating a collection of periodic walking gaits for Cassie Blue under different scenarios using C-FROST[3]. Then we validate the feasibility of the optimized walking gaits on Cassie through experiments.

3.4.1 Computational Performance Analysis

To show how C-FROST can be used to improve the gait optimization process, we first compare its performance with the original MATLAB implementation in FROST, and then secondly, we show the improvement in performance when using parallelization on different platforms, including commercial cloud servers with large numbers of CPU cores.

Comparison versus FROST. In this comparison, we tested running a single gait optimization using FROST on MATLAB and using C-FROST running as a native program on Ubuntu. The optimization is set to design a walk-in-place gait (zero sagittal and lateral

[3]The source code of this work is now publically available on `https://github.com/UMich-BipedLab/Cassie_CFROST`.

Table 3.1: Time to perform $11^3 = 1,331$ gait optimizations for the Cassie. The same pre-compiled code is run on all platforms. The average compute time for a gait is similar in all cases; the total times essentially inversely scale with the number of threads with slight differences with regards to the CPU frequency.

Computing Resource	CPU Cores	OS	Time (min)
Desktop 9-7900X @ 3.30GHz	10 cores (20 threads)	Ubuntu 18.04	401.18
Google GCP Xeon Skylake @ 2.0 GHz	64 vCPUs (64 threads)	Ubuntu 18.04	113.93
Amazon EC2 Xeon Platinum @ 3.0 GHz	72 vCPUs (72 threads)	AMI Linux 4.14	91.11

speeds) on flat ground. Both tests were run on a Thinkpad laptop with a 2.6GHz Intel i7 processor, under identical settings, and with the same initial guess. No parallelization was enabled in this test.

In our test, it took 90.12 seconds to generate the walk-in-place gait when running on MATLAB, whereas, using C-FROST resulted in the gaits being generated in 24.51 seconds. The result shows that using C-FROST speeds up the gait optimization process by approximately 3.5 times for a single gait. This demonstrates that C-FROST can be used to rapidly generate dynamic motions for legged robots without dealing with the computational overhead associated with MATLAB while still allowing one to use the easy-to-prototype environment provided by FROST. Note that, the above comparison is of the time to run the optimizations only. The setup time of defining the optimization problems is essentially the same for both FROST and C-FROST.

Single gait optimization with subroutine parallelization. Further analyzing the result of C-FROST optimization, approximately 80% of the total computational time was spent to evaluate the NLP functions and the rest was used for IPOPT internal computation. Moreover, approximately 85% of the NLP function evaluation time was used to compute the sparse Jacobian matrix of the constraints. It indicates that computing Jacobian matrix becomes the main performance bottle neck.

Here, we report the performance improvements while enabling multi-threading for the computation of Jacobian matrix as described in 3.2.3. In this test, we ran the walk-in-place gait optimization on a Linux virtual machine on the Google Cloud Platform. The results are shown in 3.4. The plot shows that while the time required for IPOPT internal computation and other user-defined NLP functions other than the constraint Jacobian (the different between blue and purple line) remaining the same, the time to compute the constraint Jaco-

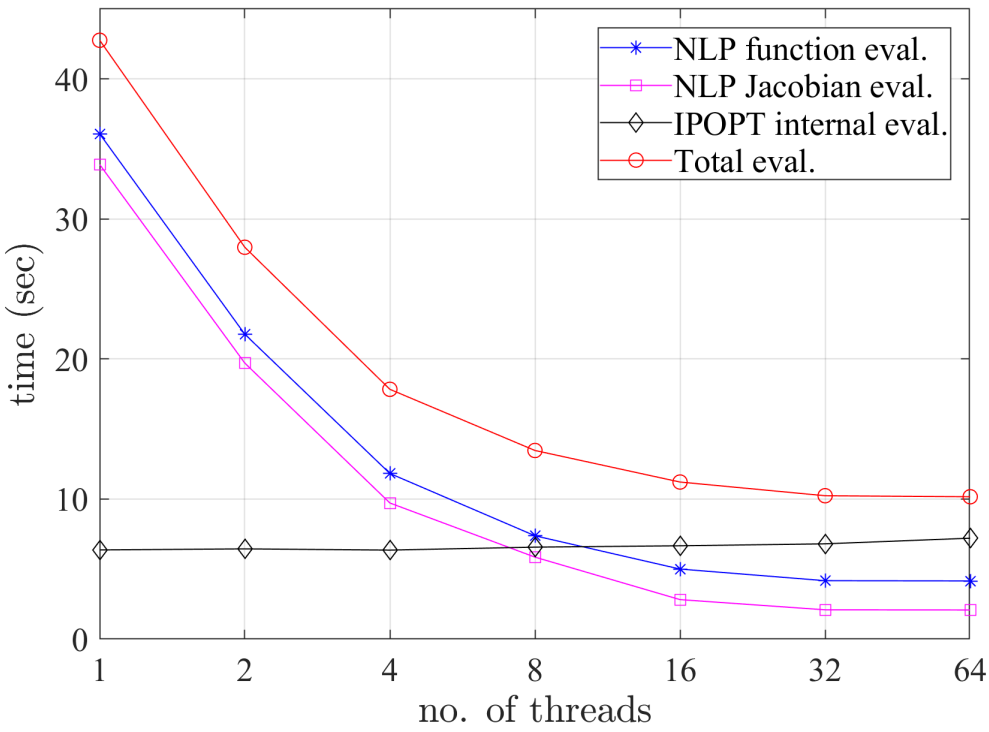


Figure 3.4: The computation time of each components while using different number of threads.

bian (the purple line) reduces significantly when the available number of threads increases. It reduces from 34 (single core) to 2 seconds (64 cores), which yields a 17 times speed-up. When having more than 16 cores, however, the overhead time for synchronization between threads will dominate the overall run time, and the change is barely noticeable. Moreover, the affect on the total evaluation time becomes less when it requires more time for IPOPT internal computation. Overall, the total optimization evaluation sped up from 42.8 seconds to 10.2 seconds when using up to 64 cores.

Generating multiple gaits in parallel. The speed-up factor of the subroutine parallelization is not significant enough if considering the number of CPU cores being used. In need of generating multiple gaits at the same time, running all optimizations in parallel on available CPU cores will be more efficient. In our final test, all $1,331$ gaits were generated in a single batch by running the optimization solvers in parallel using all available cores on various computing platforms. The JSON configuration files that specify the boundary condition for each optimization are generated a priori, and then multiple instances of the same compiled program were set to run on different CPU cores using different boundary conditions under identical settings. When attempting to run multiple optimizations in parallel, we use the GNU Parallel package. Table 3.1 shows the computation times, specific machines, number of cores, and OS. The results show that utilizing all available computing resources of the given platforms, C-FROST can dramatically reduce the total time to compute batches of gaits. This feature is particularly important for learning based walking controllers, which require that one generate a large set of gaits as training data. In particular, users can take advantage of commercial cloud servers, such as Amazon's AWS or Google's GCP to generate a large set of gaits in a much shorter period. For example, using 72 vCPUs on AWS yields an average-per-gait computation time of approximately 4.1 seconds. This result shows that C-FROST noticeably reduces the computation time of trajectory optimization for a high-dimensional hybrid dynamical system without compromising problem formulation with simplified models. It is also important to note that C-FROST only needs to compile once the program even for thousands of gait optimization problems. Different gaits are determined by the boundary conditions defined in JSON files. That is, the setup time of a library of gait will be significantly reduced when using C-FROST.

3.4.2 Removing the Velocity Terms

We investigate the effect of removing the velocity terms from a bipedal robot's dynamics. How much do the velocity terms contribute to a typical walking gait? Will removing the terms significantly reduce the optimization time without significantly compromising the

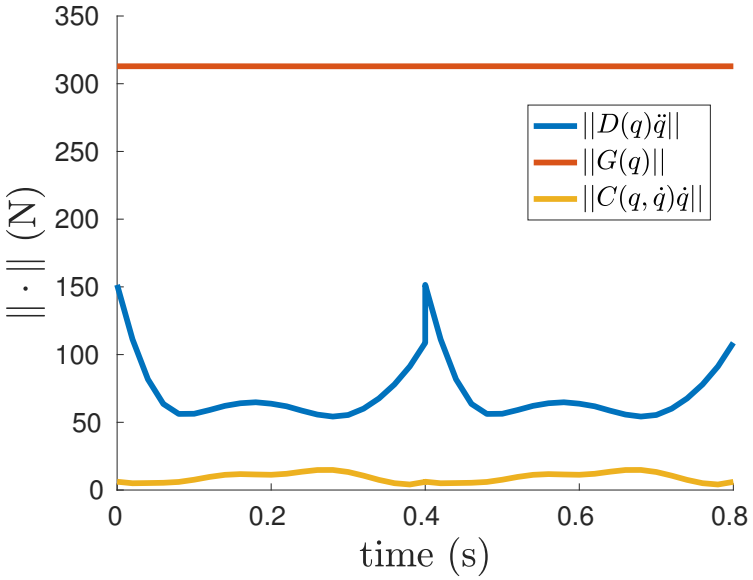
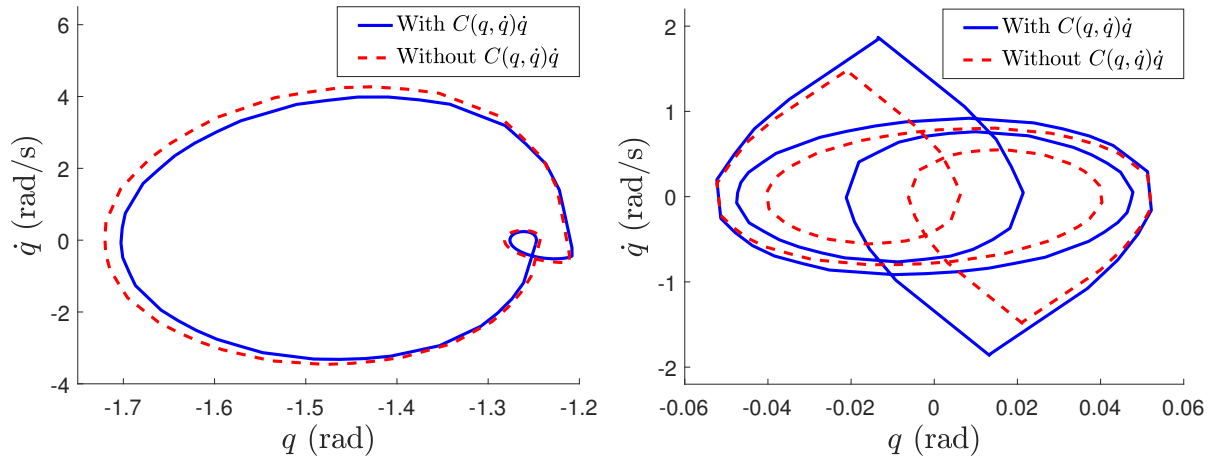


Figure 3.5: Comparison of the relative contributions of the various components of the dynamic equation (2.1) for forward walking at 1 m/s on flat ground.

quality of the trajectory?

It has been a classical maneuver in the control community for manipulator arms to remove the velocity terms from the dynamic model [28, Chap. 10.7]. The intuition is that a manipulator's motions are often sufficiently slow such that the squared velocity terms have very little impact on the system dynamics. To determine the relative contribution of the velocity terms compared to other components (such as the gravity and inertial effects) in the natural dynamics of a biped, we measure quantities, $||D(q)\ddot{q}||$, $||C(q,\dot{q})\dot{q}||$, and $||G(q)||$, along a periodic orbit of a walking gait. Figure 3.5 shows the contributions of the various terms when Cassie is walking forward at 1 m/s. As seen, the contribution of velocity terms is significantly overshadowed by the inertial and gravity terms.

Of course, looking at norms is a crude measure and saving computation time is only interesting if the performance of a gait computed on the basis of a model without the velocity terms is sufficiently similar to the model with the velocity terms. Here, "performance" will be interpreted in terms of the resulting limit cycle, motor torques, and the ground reaction forces. To investigate these, two gaits are generated for forward walking at 1 m/s on flat ground, one including the velocity terms and one dropping them. Figure 3.6 compares the limit cycles for a typical actuated joint and the only passive degree of freedom when in single support, namely the torso roll angle. While not identical, the differences could be not discerned in an experiment. The left knee joint orbits are within two tenths of a degree while the maximum deviation in roll angle is less than one degree. Figure 3.7 shows the torques associated with the two gaits and the normal component of the stance foot ground



(a) Left Knee Angle

(b) Torso Roll Angle

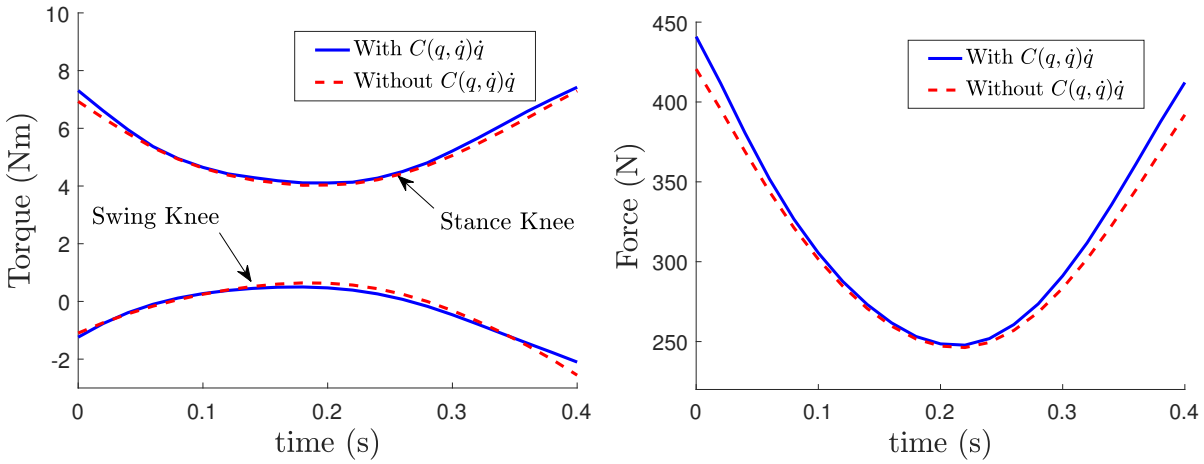
Figure 3.6: Projections of the periodic orbits for the gait in Figure 3.5 over one step on each leg. (a) A typical actuated joint, the left knee, and (b) the only "passive" variable in single support, torso roll. Note that the roll difference is less than one degree.

Table 3.2: Time to perform $11^3 = 1,331$ gait optimizations for the Cassie with the Velocity terms removed.

Computing Resource	CPU Cores	OS	Time (min)
Desktop 9-7900X @ 3.30GHz	10 cores (20 threads)	Ubuntu 18.04	13.83
Google GCP Xeon Skylake @ 2.0 GHz	64 vCPUs (64 threads)	Ubuntu 18.04	16.95
Amazon EC2 Xeon Platinum @ 3.0 GHz	72 vCPUs (72 threads)	AMI Linux 4.14	4.32

reaction force. Once again, the differences seem to be unimportant. It should be noted that the reported differences could come from slightly different local minima due to removing the velocity terms.

Dropping the velocity terms $C(q,\dot{q})\dot{q}$ from (2.1) significantly reduces the time to create and run the gait optimization problem in (3.6) while using FROST with C-FROST. Table 3.2 shows the results of running the optimizations in parallel for the library of 1331 gaits using the same platforms seen in Table 3.1. When looking at the Desktop row, we can see around 30 times improvement in computation time. From our tests, the speed-up is due to fewer iterations being required for convergence as well as less time required to evaluate each iteration when dropping the velocity terms.



(a) Actuator Torques

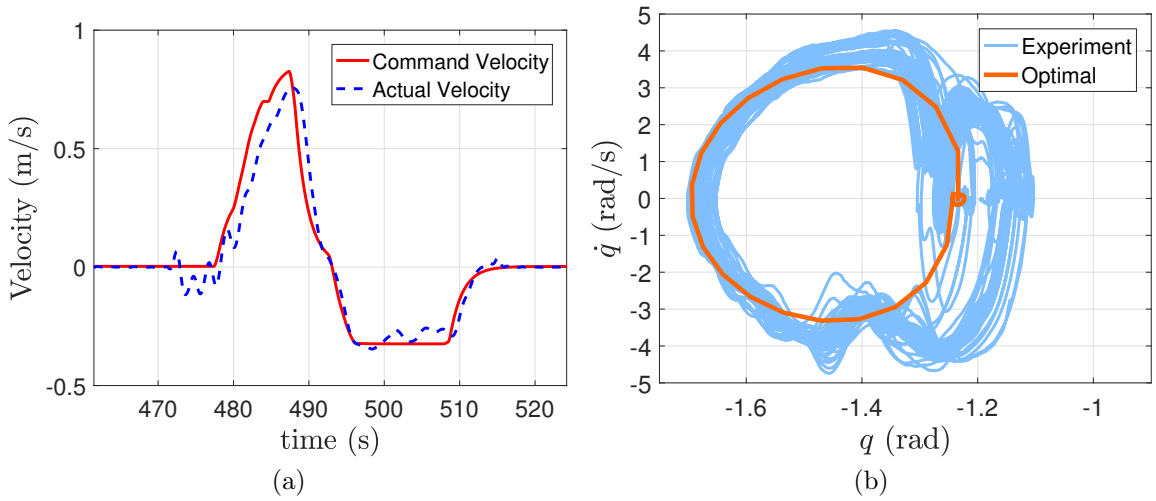
(b) Vertical Ground Force

Figure 3.7: Comparison of the actuator torques and the vertical ground reaction force for the gait in Figure 3.5 over one step.

3.4.3 Experimental Validation

The purpose of the experiments is to assure the reader that the set of constraints imposed during trajectory optimization are physically realistic and were not selected to simplify the designs. The controller that realizes the trajectories on Cassie Blue is based on [30] as detailed in [41]. Note that, only a subset of gaits from the previously generated 1,331 gaits are being utilized in the experiment. Regulating lateral velocity and ground inclination that uses the full set of gaits will be subject to future work.

During these tests, the 11 sagittal gaits with speeds ranging from -1.0 m/s to 1.0 m/s on flat ground were used to generate a gait library [30, 41] for Cassie Blue to regulate the forward and backward walking speeds. A heuristic foot placement controller was also applied to stabilize the lateral motion of the robot. In the experiment, we use a remote controller to command a reference speed profile, and the robot utilized the gait library to follow the commanded speeds. The robot can accelerate from 0 m/s to 0.8 m/s in around 10 seconds and then slow down to walking in place (e.g., 0 m/s) in 5 seconds. The robot is then commanded to walk backward with a nearly constant velocity of 0.3 m/s across the room to return to the starting point. Figure 3.8a shows the tracking of the commanded walking velocity in the experiment. Figure 3.8b compares the similarity in the phase portraits of the right knee angle from optimization and experiment when the robot is stepping in place (i.e., zero forward velocity). Figure 3.8b also shows the periodicity of the sustainable walking motion. A video that illustrates the different walking motions from the gait library built from optimization and the presented experimental results is available at [1].



(a) (b)

Figure 3.8: Experimental results for a gait-library-based sustainable walking controller on Cassie Blue. (a) The tracking of the commanded forward/backward walking speeds (measured at the CoM position). (b) The periodic phase portrait of an actuated joint (right knee) when stepping in place.

3.5 Conclusions

The primary objective of the chapter was the presentation of a method to speed up trajectory design for high-dimensional systems by 3-4 times faster for single trajectories if not using parallelization and up to 9 times faster with parallelization. Moreover, the average computational time for a gait reduced significantly for families of trajectories. This was accomplished through the design of C-FROST, an open-source C++ interface for FROST—a MATLAB-based direct trajectory optimization tool for robotic systems. The significant speed up was illustrated by the design of a library of walking gaits for a Cassie bipedal robot. For a single gait, C-FROST completed an optimization in 24.51 seconds vs 90.12 seconds in FROST. The speedup is much more significant when doing multiple gaits, such as computing a (periodic) gait library [30] or building an invariant surface of open-loop transition trajectories [29]. The chapter documents the parallel computation in C-FROST of $1,331$ gaits for a Cassie-series bipedal robot on a range of platforms. The average computation time is roughly 4.1 seconds per gait while running on an AWS virtual machine. In future work, we will consider using other parallel platforms, such as GPUs, to speed up the trajectory optimization problems further.

CHAPTER 4

Control of an Exoskeleton

4.1 Motivation

This chapter describes the hardware, shown in Figure 4.1, that has been designed to achieve hands-free dynamic walking, the control laws that have been deployed and those being developed to provide enhanced mobility and robustness, as well as the early test results alluded to above. In this chapter, dynamic walking refers to a motion that is orbitally stable as opposed to statically stable.

At present approximately 4.7 million people in the United States would benefit from an active lower-limb exoskeleton due to the effects of stroke, polio, multiple sclerosis, spinal cord injury, and cerebral palsy [33]. Moreover, by 2050, an estimated 1.5 million people in the United States will be living with a major lower-limb amputation [128]. Such individuals expend up to twice the metabolic effort to walk at half the speed of able-bodied persons, experience higher-risk of falls, and have secondary pathological conditions such as osteoarthritis, back pain, and depression [116, 85, 76]. Lower-limb exoskeletons serve as assistive devices by providing support and balance to wheelchair users and enabling them to perform normal ambulatory functions such as standing, walking and climbing stairs. Lower-limb exoskeletons have also been utilized for gait training and rehabilitation purposes.

More importantly, standing and walking with these assistive devices provides exceptional health benefits. For instance, for people with paraplegia, the benefits include improvement of blood circulation, of respiratory, urinary, and intestinal functions, as well as positive psychological effects [72], fundamentally improving their quality of life. For spinal cord injury (SCI) patients, the benefits include improved bone density, cardiorespiratory function, gastrointestinal function, sitting balance, and decreased pain and spasticity [37].

The objective of the work described herein is to translate formal control design methodologies from bipedal robots to exoskeleton systems so as to achieve dynamic hands-free walking. This is a formidable problem as control of bio-mechatronic exoskeleton devices not



Figure 4.1: ATALANTE: An exoskeleton designed by Wandercraft for people with paraplegia.

only share many of the challenges of bipedal robot locomotion but also challenges introduced by the integration of an active human user. These challenges include, nonlinear, high degree-of-freedom hybrid dynamics, workspace limitations, actuator constraints, unilateral ground contact forces, being robust to variations in the user's dynamical parameters such as mass and inertia, being able to handle interaction forces between the user and the device, and enforcing safety-critical constraints for the operation of the exoskeleton.

Over the years, several research groups and companies have begun responding to the need and benefits of exoskeletons. Exoskeletons can be designed for human performance augmentation and as orthotic devices. Exoskeletons for Human Performance Augmentation are designed to enhance the strength and physical capabilities of able-bodied users, to provide fatigue relief and protection to factory and construction workers, soldiers, and disaster relief workers, and to assist them in carrying heavy loads for prolonged periods of time. In contrast, orthotic devices are designed to assist and restore autonomy to individuals with physical impairments causing difficulty in walking. Orthotic devices are also designed for rehabilitation purposes - to provide gait training and therapy. A comprehensive review of the state-of-the art lower-limb exoskeletons can be found in [33, 122, 61].

Early exoskeletons, developed by General Electric [80], the University of Wisconsin [47], and the Mihailo Pupin Institute [112, 60], focused on human augmentation or assistance, supporting multi-task capabilities such as walking, standing up from a seated position, sitting down, stepping over obstacles, and climbing stairs, all through pre-programmed motion patterns that were executed at the user's command. These systems, however, were not very robust. Turning to more recent devices, Ekso Bionics' robotic exoskeleton EksoGT [3] is primarily designed for use in clinical settings for rehabilitation and gait training for stroke and SCI patients. ReWalk [5, 124] is another robotic lower-limb exoskeleton to enable patients with SCI to stand-up, walk, turn and climb stairs. The Hybrid Assistive Limb (HAL) [104] developed at the University of Tsukuba, Japan and Cyberdyne [12] provides locomotion assistance to physically challenged persons. These are high-degree-of-freedom exoskeletons that have multiple actuators at the hip, knee and ankle. A primary limitation, however, of the cited exoskeletons is that they require external support mechanisms such as crutches or canes for the user to maintain balance while walking with the device. While REX bionics' lower-limb exoskeleton provides hands-free functionality, it only allows slow static gaits with velocities on the order of 0.05 m/s.

While modern day hardware for exoskeletons and prosthetics is becoming lighter, stronger, and power-dense, the current approaches to control of powered leg devices are rudimentary and driven by finite-state machines with several phases such as swing, stance, heel-strike and toe-off [64], [103, Fig. 9], each with numerous tunable parameters that are specific to each

user [96, 97], offering no formal guarantees of either stability or safety [42], and typically require the use of additional aids such as crutches to be safely used [102, 19].

A general review of various control strategies for lower-limb assistive robotics is presented in [64, 18, 109]. In particular, low-level control strategies are either position-based [104, 14, 39, 82, 35] or torque / force-based [63, 108, 110, 120, 94], while a higher-level impedance or admittance controller is used to regulate human-device interaction forces. This is in stark contrast to the surge in control technology for highly dynamic bipedal locomotion [101, 34, 73, 32, 71, 107], where tools are being developed that allow rapid design of gaits and model-based feedback controllers, that respect physical constraints of the system such as torque limits and joint speeds, while providing formal guarantees on stability, safety and robustness to uncertainties in the model and in the environment [119, 43, 36, 83, 53]. If the control and design methodologies underlying advanced locomotion strategies for bipedal robots can be successfully translated to powered prostheses and exoskeletons in a holistic and formal manner, the end result promises to be a new generation of wearable robotic devices that deliver the next level of stable, safe, and efficient mobility.

This chapter is divided to to parts showing results from exoskeleton research done in [13, 51]. First we will start by talking about approach in [13] where we demonstrate a nonlinear decentralized controller for a lower extremity exoskeleton using the gait library approach. Section 4.2 presents a hybrid, nonlinear model of the human-exoskeleton system. In Section 4.3 a centralized HZD-based controller is presented. In Section 4.4 a velocity regulation scheme, that allows the human subject to control velocity of the exoskeleton, is presented. Section 4.5 presents a decentralized version of the HZD controller for the human-exoskeleton subsystems. Section 4.6 presents results of numerical simulations and compares the two controllers. The primary contributions with respect to prior work are:

- Generation of natural walking gaits that are dynamically feasible for the human-exoskeleton system that enforce the unilateral ground contact as well as friction and input torque constraints.

- Design of a control architecture allowing the human to actively control and regulate the walking velocity of the human-exoskeleton system using his torso.

- Design of a decentralized controller, with minimal sharing of sensing information between the human and exoskeleton subsystems, that not only preserves the periodic orbit created by a centralized HZD controller, but is also robust to perturbations.

- Numerical validation of the proposed controller on a 21 degrees-of-freedom (DOF) human-exoskeleton system walking at speeds ranging between 0.13 m/s and 0.34 m/s,

and being sufficiently robust to walk on ramps of up to 5° slopes.

In the second part of the Chapter, we focus on results from [51] where we develop a controller using the G-HZD approach. The newer technique harness the power of modern optimization tools and are blazing the way for improved controller designs that deliver multiple walking speeds, turning, and enhanced robustness for exoskeleton locomotion.

To be totally transparent, in terms of "gain tuning", the local controllers at the joint level will be tuned based on a nominal walking motion and then left fixed. In early stages of optimization, the constraints are adjusted to provide adequate foot clearances given the observed tracking errors in the local joint controllers and small errors in calibration. Post-optimization, a constant bias is sometimes added to a commanded joint profile to compensate for model errors or tracking errors. The method presented combines offline trajectory optimization and machine learning to design stabilized gaits that can be robustly tracked online. Preliminary robustness and stability analysis of G-HZD design approach is numerically illustrated in simulation.

4.2 Model

In this section, we first provide a brief description of the hardware and sensors of the exoskeleton hardware. We next derive a mathematical model of walking for the human-exoskeleton system, which can be represented by a hybrid control system. The model developed here is used later to find periodic walking gaits, develop feedback controllers to stabilize these gaits, and perform numerical simulations of the hybrid control system. The most basic of these generated gaits are evaluated in experiments.

4.2.1 Hardware Description

ATALANTE, developed by the French startup Wandercraft, is a fully-actuated lower-limb exoskeleton intended for use in medical centers for rehabilitation of patients with paraplegia. The exoskeleton consists of 12 actuated joints as shown in Figure 4.3: three joints that control the spherical motion of each hip, and in each leg, a single joint for the knee, and two joints for the ankle rotation in the sagittal and frontal plane, respectively. Except for the ankle, where a special mechanism is mounted, each degree of freedom is independently actuated by a brushless DC motor. The displacement and velocity of each actuated joint are measured by a digital encoder mounted on the corresponding motor as well as by three Inertial Measurement Units (IMUs), with one attached to the torso and one on each leg above the ankles. Four 3-axis force sensors attached to the bottom of each foot are used

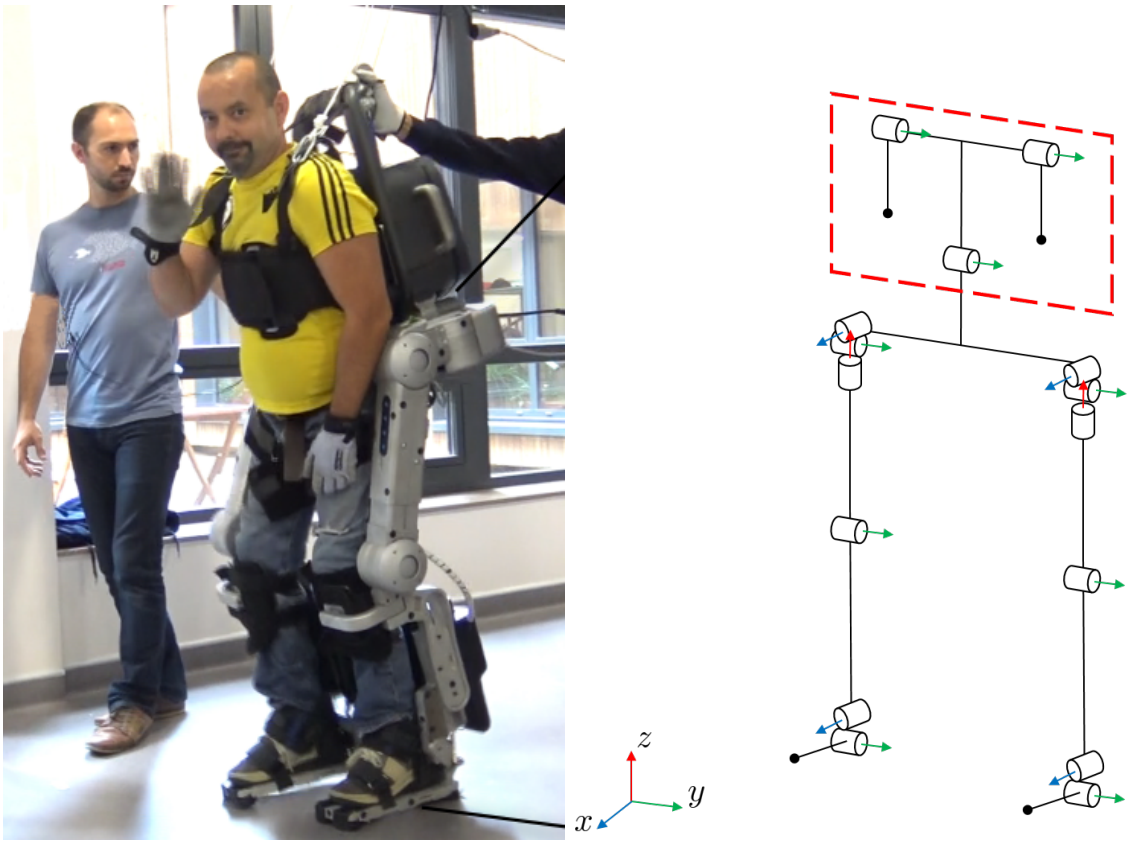


Figure 4.2: The kinematic diagram of human-exoskeleton lumped system. The patient is secured to the exoskeleton by means of fasteners located at the ankle, the shin, the thigh, the abdomen, and the torso.

to detect ground contact. All electrical components of the exoskeleton are controlled by a central processing unit running a real-time operating system and in charge of high-level computations.

The mechanical design of ATALANTE allows the leg length and hip width to be manually adjusted to fit to an individual patient's personal measurements. This presents a challenge from the control design perspective in that the controller should be robust to these physical changes of the model. Because the exoskeleton is designed to fully support the user's weight, the user is securely strapped to the device from the feet up to the abdomen as shown in Figure 4.2.

4.2.2 Mathematical Representation

The goal is to study the dynamical behavior of the human-exoskeleton and to avoid over complicating the model by considering the compliant elements present in the human body and exoskeleton linkages. One approach is to lump human-exoskeleton system is modeled

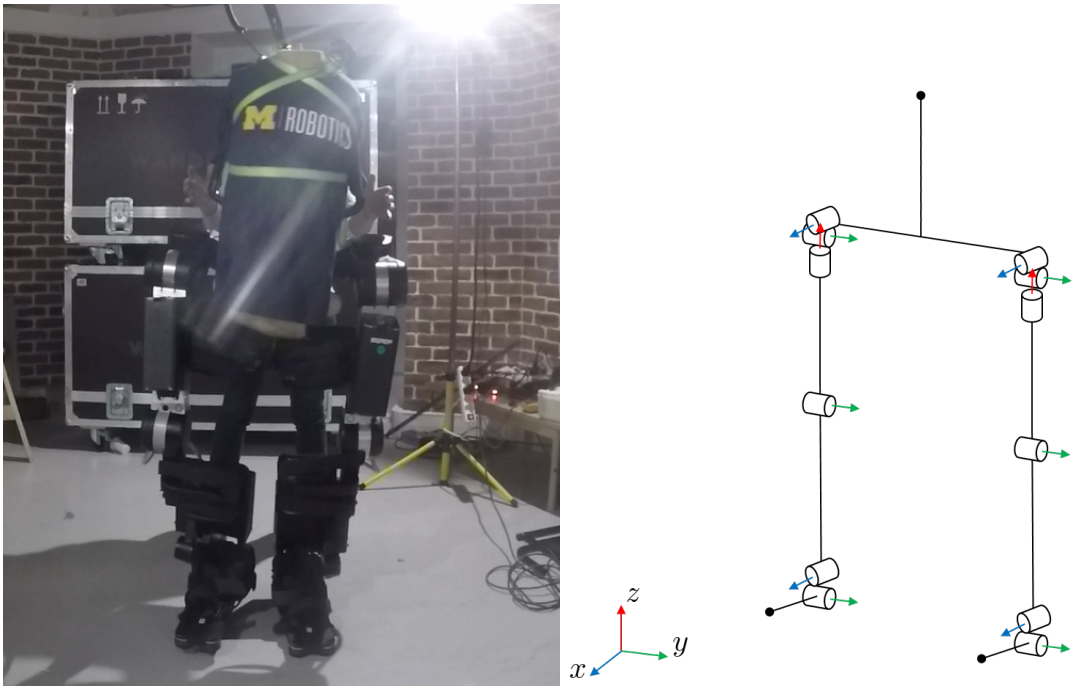


Figure 4.3: The kinematic diagram of human-exoskeleton lumped system where the human is assumed to be a lump of mass attatched to the exo, similar to a rigid mannequin.

as a rigid body system represented by a kinematic tree as shown in Figure 4.3. In other words, the upper body of the human is modeled as a single rigid link attached to the torso of the exoskeleton. In particular, the patient does not provide any actuation, however, the approximate masses and inertias of the patient are combined in the corresponding links of the exoskeleton. Such a model appears to be appropriate for paraplegic patients who have complete loss of motor input in their lower extremity.

Based on the rigid body assumption, the mathematical representation of the system dynamics can be obtained via the Euler-Lagrangian equations of motion of rigid body dynamics. Specifically, a floating-base generalized coordinate system is considered with the coordinate variables defined as

$$q = (p, \phi, q_b) \in \mathcal{Q}, \tag{4.1}$$

where $p \in \mathbb{R}^3$ and $\phi \in SO(3)$ denote the relative position and orientation of the exoskeleton's base frame with respect to the world frame respectively, and $q_b \in \mathbb{R}^{12}$ denotes the relative angles of the actuated joints. The dynamical model from Section 2.2 can used.

4.2.3 Model with Human separately

Alternative representation that adds more complexity to the human model can be seen in Figure 4.2. Particularly, we model the torso and arms of the human as separate rigid links connected by revolute joints to the exoskeleton's torso. Under this model, the floating base model of the overall system has 21 DOF, with six of those corresponding to the floating base. The system has 15 degrees of actuation, with 12 actuators for the exoskeleton and three for the human.

4.3 Feedback for Centralized Control

Having developed the hybrid dynamical model of the human-exoskeleton system, we next present a feedback controller to achieve dynamic walking. In particular, this will be a centralized controller that has been successfully implemented on high DOF underactuated 3D bipedal robots [56, 91, 100]. An identifying feature of the control method is that the various links of the robot are coordinated by a single phase variable that depends on the robot state only and is independent of time. This method has been successfully translated to lower limb prosthesis [42, 126].

4.3.1 Virtual Constraints

At the core of our controller approach is designing a set of outputs, $y(x)$, known as virtual constraints, such that the zeroing of these outputs leads to the desired robot behavior. In other words, virtual constraints are functions of the state variables that define how the various links of the robot should move. These virtual constraints are regulated via feedback control rather than through mechanical linkages, and hence the name "virtual" constraints.

These virtual constraints are formulated as the difference between their actual values, y^a, and desired values, y^d, yielding

$$y = y^a - y^d. \tag{4.2}$$

For the particular case of 3D flat-footed walking, we chose the outputs, y^a, to be a combination of velocity regulating terms, y_1^a and position modulating terms, y_2^a, i.e. $y^a = [y_1^a; y_2^a]$ where y_i^a is a relative degree i output, $i \in \{1, 2\}$.

We choose $y_1^d = v_d \in \mathbb{R}$ to be the corresponding desired velocity and choose

$$y_2^d = y_2^d(\tau(q, \beta), \alpha) \tag{4.3}$$

as the desired relative degree two output described by a set of parameters, α, and *phase vari-*

able, τ, a monotonically increasing function of the configuration variables, q, and parameters, $\beta = (\beta_1, \beta_2)$. Here, the phase variable is defined as

$$\tau(q, \beta) := \frac{\delta p_{hip}(q) - \beta_2}{\beta_1 - \beta_2} \in [0, 1], \tag{4.4}$$

where δp_{hip} is the horizontal hip position relative to the stance foot position in the sagittal plane (see Figure 4.2). Note that τ is normalized to be within $[0, 1]$. The scaling vector (β_1, β_2) is defined as

$$(\beta_1, \beta_2) := (\delta p_{hip}(q_0), \delta p_{hip}(q_f)), \tag{4.5}$$

where q_0 and q_f define the configuration of the human-exoskeleton system at the beginning and end of a step.

For the relative degree two virtual constraints, we use Bézier polynomials to parametrize the desired evolution as a function of the phase variable, τ,

$$y_2^d(\tau, \alpha) := \sum_{k=0}^{M} \alpha[k] \frac{M!}{k!(M-k)!} \tau^k (1-\tau)^{M-k}, \tag{4.6}$$

where M is the order of the Bézier defined by $M + 1$ coefficients. The tuple $\{\alpha, \beta, v_d, x^{\mathcal{O}}\}$ then defines a gait.

Remark 3. *For the human-exoskeleton system, the following virtual constraints are defined in terms of actuated joints and are hence in body coordinates: Stance Knee Pitch, Stance Hip Roll, Stance Ankle Roll, Stance Hip Yaw, Stance Pelvis Pitch, Stance Arm Pitch, Torso Pitch, Swing Arm Pitch, Swing Knee Pitch, Swing Hip Roll and Swing Hip Pitch. The following virtual constraints, on the other hand, are defined in terms of world-frame orientation: Swing Foot Roll, Swing Foot Pitch and Swing Foot Yaw. For the each of these 14 virtual constraints, the actuator torques show up in their second derivatives and hence are said to be relative degree two and are denoted as y_2. The remaining virtual constraint is defined as the horizontal velocity of the hip; the torques show up in the first derivative of this virtual constraint; it is therefore said to be relative degree one and is denoted by y_1. Note that we have defined 15 virtual constraints corresponding to the 15 actuators in the human-exoskeleton system.*

4.3.2 Partial Hybrid Zero Dynamics and a Central Controller

Having established a hybrid dynamical model of walking and presented a set of virtual constraints that needs to be enforced, we use feedback techniques developed in [119] to

enforce these constraints. This creates a partial hybrid zero dynamics surface (partial HZD) [17] embedded in the state space. The developed controller is a centralized controller in the sense that it requires full knowledge of the entire state vector. More details on partial hybrid zero dynamics and the centralized controller have been relegated to Appendix B.

4.3.3 Gait Generation via Direct Collocation Method

For the HZD method of control design, we need to design the desired profile of the virtual constraints that when enforced leads to a stable walking gait. This is done through a nonlinear constrained optimization process using direct collocation [56]. Further details on the direct collocation based optimization are presented in Appendix A, while the constraints that are enforced by the optimization are presented in Appendix C.

The result from the optimization is the set of parameters $\{\alpha, \beta, v_d, x^{\mathcal{O}}\}$ which fully describes an individual gait, where $x^{\mathcal{O}}$ is the evolution of the states of the system along the orbit $\mathcal{O}$ parametrized by τ, while α, β, v_d are as defined previously. The stability of the resulting gait can be inferred as in [119, Thm. 5.3], or as we will do later, by directly computing the Poincaré map.

4.4 Velocity Regulation

Having presented the continuous-time HZD controller and the nonlinear constrained optimization to generate periodic gaits, we now present a method to enable walking velocity regulation based on the desired human velocity. In particular, this section presents a convenient means for the user to vary the walking speed of the exoskeleton, while maintaining stability and satisfying torque, ZMP[1] and friction constraints. The key ideas are: (1) to

[1]The Zero Moment Point (ZMP) is a point in the contact region of the stance foot and the ground where the sum of all moments of active forces with respect to this point is equal to zero. The ZMP constraint

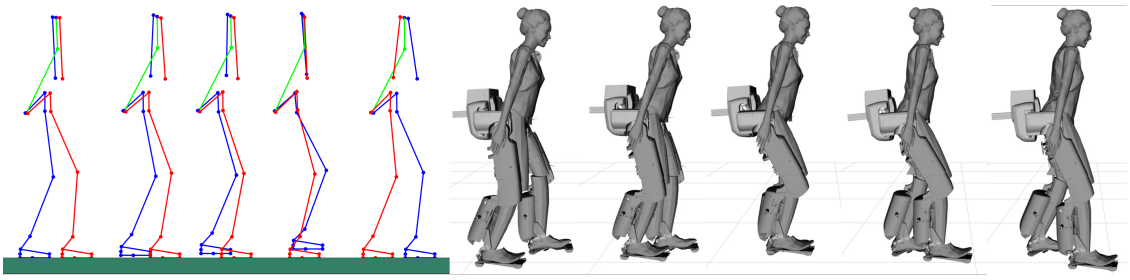


Figure 4.4: Snapshots of walking at a desired speed of 0.3 m/s achieved by our proposed controller on the human-exoskeleton system is shown. An animation video of the simulation is available at `https://youtu.be/VXP7DKY6Trc`.

design a discrete library of gaits and controllers for stable walking at fixed speeds; (2) to interpolate the gaits and controllers to create a continuum of walking gaits and controllers [30]; and (3) for the user to command speed increases or decreases through his or her torso lean angle, similar to how one rides a Segway. Each of these aspects of the speed regulation is addressed in the following.

4.4.1 Library of Gaits

To regulate the velocity of the exoskeleton, a method similar to [30] is used. A finite library of gaits, $\mathfrak{L}$, is generated through optimization for different desired walking speeds v_d. The step duration is restricted to around 0.8 (s) and all physical constraints discussed in Section C are imposed. Each gait in the library $\mathfrak{L}$ is defined by the parameters $\left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}^i := \left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}(v_d^i)$, all of which were defined in 4.3.1. The gaits are ordered by increasing values of v_d^i. Gaits were generated for values of

$$\begin{aligned}\{v_d^1, \cdots, v_d^7\} = \; & \{0.13, 0.15, 0.18, 0.20, \\ & 0.24, 0.29, 0.34\} \text{ m/s}.\end{aligned} \tag{4.7}$$

Gait parameters at an intermediate speed v are generated by linearly interpolating the gait library. Specifically,

$$\zeta = \frac{v - v_d^i}{v_d^{i+1} - v_d^i}, v_d^i \leq v < v_d^{i+1}, \tag{4.8}$$

$$\begin{aligned}\left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}(v) = \; & (1-\zeta) * \left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}^i \\ & + \zeta * \left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}^{i+1},\end{aligned} \tag{4.9}$$

where $\zeta \in [0, 1]$ is the interpolation factor and the parameters for the interpolated gait are $\left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}(v)$ with $v_d = v$.

Remark 4. *To be extra clear, in the above equations the parameters $\{\alpha, \beta, v_d, x^{\mathcal{O}}\}$ vary continuously with the velocity v.*

This then creates a continuum of gaits, each of which is locally exponentially stable and meets the design constraints given in Section C, see [30] for reasons on why this is true. To demonstrate this, we numerically compute the Poincare map as described in [119] and tabulate the modulus of the dominant eigenvalue of the linearized Poincare map in Table 4.1

enforces that the ZMP is within, and not at the boundary of, the foot geometry.

Table 4.1: Dominant Eigenvalues at different speeds.

Forward Velocity (m/s)	Largest Eigenvalue
0.17	0.3023
0.19	0.2150
0.21	0.3606
0.23	0.3313
0.25	0.1891
0.27	0.1764

for different interpolated speeds. Since these values are strictly less than 1, the gaits are locally exponentially stable, see [119] for more details.

4.4.2 Relaxing local stability of walking speed

The above gaits, being asymptotically stable, will reject attempts by the user to change the walking speed through body posture adjustments. We now prepare the controller to more readily accept such posture changes as speed commands from the user.

Let $y_1^a(x^-)$ be the longitudinal velocity of the exoskeleton's hip at the end of the current step. When selecting a gait for the next step, suppose that we set v in (4.8) and (4.9) such that $v = y_1^a(x^-)$. Then, as explained in [30], two things will happen. First of all, we are selecting a gait that respects important physical constraints around the system's current walking speed. Secondly, the closed-loop system becomes "neutrally stable" in the sense that all but one of the eigenvalues of the Jacobian of the Poincaré map are strictly within the unit circle, and the remaining eigenvalue is approximately equal to one. In other words, when looking at the system step to step, the longitudinal speed behaves like an integrator because the controller's setpoint at each step is reset to the speed of previous step. Said another way, if a "disturbance" or "user command" causes the system to increase or decrease walking speed over the course of a step, the new speed will be maintained during the next step. The next subsection will show this to be a highly desirable property for the closed-loop system as it makes it easy for the user to regulate speed through torso pitch angle.

4.4.3 User Regulation of Walking Speed

With the exoskeleton being controlled to walk at a continuum of velocities, but being neutrally stable in the sense described above, a means is now provided for the user to stabilize the walking speed to a desired value. This is accomplished by having the human command

speed changes to the exoskeleton through changes in torso pitch angle.

The basic idea can be inferred from how a human regulates the speed of a Segway: a forward lean of the body accelerates the Segway, while a backward lean decelerates it. Here, a change in the user's torso pitch with respect to the upright position will be interpreted as a desired increment (or decrement) of the walking speed with respect to the current walking speed. Speed increase or decrease over the course of a step is a discrete-time or hybrid analogue to acceleration or deceleration of a Segway. Both allow velocity regulation to be accomplished with a proportional control actions that humans master easily.

The speed regulation policy described above is implemented through a simple modification of the outer loop of the exoskeleton controller. Specifically, at the end of each step, the desired velocity is modified to take into account the user's torso angle,

$$v = y_1^a(x^-) + \delta v \tag{4.10}$$

$$\delta v := K_v * (q_T^a - 0), \tag{4.11}$$

where v is the velocity used to select a gait as seen in (4.8) and (4.9), K_v is a proportional gain, and q_T^a is the user's torso pitch angle at the end of the previous step.

4.4.4 Simple Model of User Control Actions

For the purpose of simulating a user walking in the exoskeleton, we need to model how the user will modify their torso pitch angle step-to-step when signaling desired speed changes to the exoskeleton. The simple proportional controller

$$q_T^d := K_t * (v_{H,d} - y_1^a(x^-)) \tag{4.12}$$

is assumed, where q_T^d is the desired torso angle to be achieved at the end of the next step, K_v is a proportional gain, $v_{H,d}$ is the velocity desired by the human.

The proportional controller could be replaced with a more advanced policy, such as one having integral and derivative terms to better reflect a more highly trained user, though this is not done here. We choose instead to focus on smooth transitions in the desired speeds. The following assumptions are made on how the human moves the torso. These properties could be achieved through training, or modifications could be added to the exoskeleton controller to achieve the same objectives:

- The path of the torso can be captured by a Bézier polynomial that starts at the current torso angle and ends at the desired angle, with a starting and ending slope of zero.

- To avoid large jumps in velocity, q_T^d and δv are saturated. The gains and saturations used in this exoskeleton are as follows:

$$\begin{aligned} -0.06\,\mathrm{m/s} \leq\ & \delta v & \leq 0.06\,\mathrm{m/s} \\ -5^\circ \leq\ & q_T^d & \leq 5^\circ \\ & K_t & = 0.5\,\mathrm{rad\,s/m} \\ & K_v & = 0.7\,\mathrm{m/(s\,rad)}. \end{aligned}$$

Remark 5. *It was found that $K_tK_v \leq 0.5$ results in good velocity regulation with periodic motion in steady state. Lower values result in slower tracking of the desired velocity. Note that the expression K_tK_v arises when we substitute* (4.12) *into* (4.11) *with $q_T^a \approx q_T^d$.*

Table 4.2: List of variables pertaining to the controller design, velocity regulation, and decentralized controller

Variable	Description
y, y_1, y_2, y^a, y^d	y are the set of virtual constraints imposed by the controller, where $y = y^a - y^d$. y^a is the actual values of a set of outputs from the system while y^d is the desired values for the outputs. y_1 and y_2 define relative degree one and two virtual constraints respectively.
α, β, τ	α is a set of Bèzièr polynomial coefficients used to define y_2^d. The polynomials are parametrized by the phase variable $\tau \in [0, 1]$ where τ is a function of the state of the system. β is a set of parameters used to compute τ. During a step τ should be monotonically increasing.
$x^{\mathcal{O}}$	The evolution of the states of the system along a periodic orbit $\mathcal{O}$ as a function of τ.
$\mathfrak{L}$	Ordered set of gait primitives in increasing order of desired hip velocity y_1^d
$\left\{\alpha, \beta, v_d, x^{\mathcal{O}}\right\}^{next}$	Set of gait parameters for the next step.
$y_1^a(x^-)$	The actual linearized hip velocity at the end of the previous step
$v_{H,d}$	The desired speed the **human** wants to achieve
q_T^d	The desired torso angle, chosen by the human, to be achieved by the end of the next step
q_T^a	The actual relative torso pitch angle at the end of the previous step
δv	Change in the desired hip velocity of a gait (away the periodic gait). Used by the exoskeleton to speed up or slow down
K_t	A coefficient denoting the relation between q_T^d, $v_{H,d}$, and $y_1^a(x^-)$.
K_v	Coefficient denoting the relation between δv and q_T^a.
x_G	States corresponding to the global position and orientation (i.e. position and orientation relative to an inertial frame.)
x_H	States corresponding to Human subsystem (torso pitch angle and arm joint angles).
x_E	States corresponding to Exoskeleton subsystem
$\mathfrak{x}_H$	States corresponding to Human subsystem (torso pitch angle and arm joint angles) on the periodic orbit, $\mathcal{O}$, parametrized with the phase variable τ.
$\mathfrak{x}_E$	States corresponding to Exoskeleton subsystem (torso pitch angle and arm joint angles) on the periodic orbit, $\mathcal{O}$, parametrized with the phase variable τ.

4.5 Decentralized Controller

Having presented HZD-based control design for bipedal walking and a means for the human user to control the velocity of the exoskeleton, we now introduce a decentralized control architecture for the human-exoskeleton system. The need for decentralization is motivated by the fact that it is often difficult or impractical to obtain precise state information of the human, which involves attaching sensors to various links and joints of the human pilot. Also, it is possible that the human pilot may not know what the true state of the exoskeleton system is. This motivates us to develop a decentralized controller that minimizes the sensing information that is shared between the human subject and the exoskeleton. Towards this, we first present assumptions we make about the information available to the human and the exoskeleton controllers, followed by the decentralized controller. An overview of the decentralized control design with the velocity regulation approach is presented in Figure 4.5. Note that the two control inputs (u_H, u_E) from the decentralized controller are only needed for simulations in which the human behavior is being represented through simulation. In practice only u_E is implemented on the exoskeleton system with the real human taking over for u_H.

4.5.1 Control Design Assumptions

1. *State Decomposition*: The state of the human-exoskeleton system, x, can be decomposed into states of the human, exoskeleton and global variables,

$$x = [x_H; x_E; x_G],$$

   where, $x_H \in \mathbb{R}^h$ represents the states corresponding to the human configuration variables and are available to the human alone. In the human model, these correspond to the human torso and arm joints. Similarly, $x_E \in \mathbb{R}^e$ represents the states corresponding to the configuration variables of the exoskeleton and are available to the exoskeleton controller alone. The global state variables, $x_G \in \mathbb{R}^g$ are available to both, the human as well as the exoskeleton controller. These pertain to the position and orientation of the system with respect to an inertial frame of reference. Here h, e and g are positive constants such that, $h + e + g = n$.

2. *Existence of Periodic Orbit*: There exists an asymptotically stable periodic orbit, $\mathcal{O}$, for the closed-loop system, Σ in (2.3), with the input-output linearizing controller, u_{IO}.

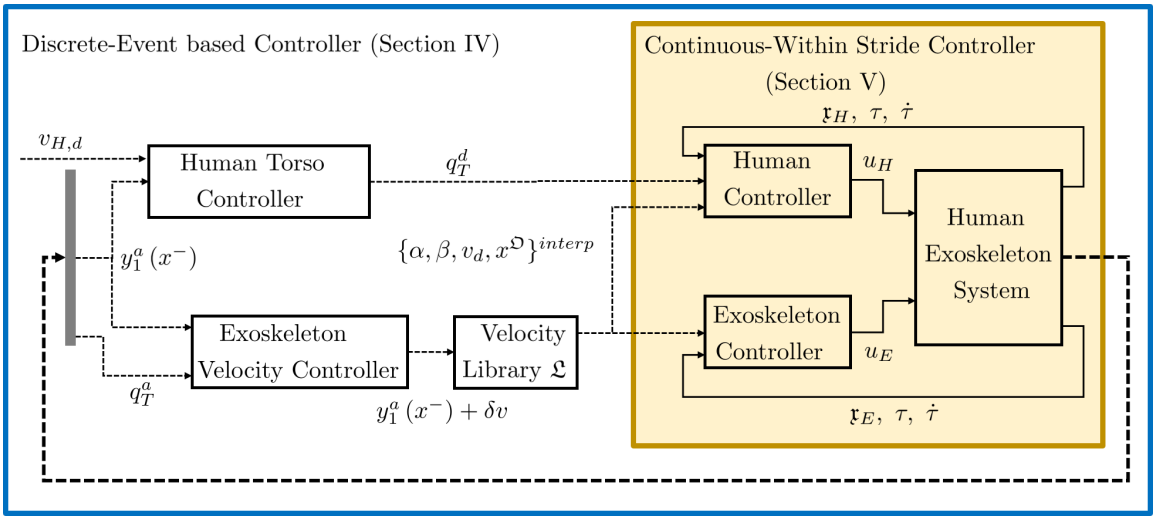


Figure 4.5: Feedback diagram illustrating the decentralized controller with velocity regulation. Block in yellow denotes continuous-time, decentralized controller. Block in blue denotes the discrete-time event-based velocity regulation controller, executed at the end of every walking step. Dashed lines denote variables sampled at the end of every step.

3. *Partial State Information*: With the exception of the assumptions 5a and 5b, it is assumed that the human subsystem has knowledge only about the human and global states, x_H, x_G, and assumes the corresponding exoskeleton states are always on the periodic orbit, $\mathcal{O}$. Similarly, the exoskeleton subsystem has information only about the exoskeleton and global states, x_E, x_G, and assumes that the states corresponding to the human are always on the periodic orbit, $\mathcal{O}$. Therefore, the variables available to the human controller, $\mathfrak{x}_H$, and the exoskeleton controller, $\mathfrak{x}_E$ are given by,

$$\mathfrak{x}_H = [x_H; x_E^{\mathcal{O}}; x_G],$$
$$\mathfrak{x}_E = [x_H^{\mathcal{O}}; x_E; x_G],$$

where, $x_E^{\mathcal{O}} \in \mathbb{R}^e$ and $x_H^{\mathcal{O}} \in \mathbb{R}^h$ represent the states corresponding to the exoskeleton and human joints on the periodic orbit, respectively.

It should be noted that $x_H^{\mathcal{O}}$ is simply the human having an upright stance relative to the exoskeleton with downright arms, since the exoskeleton is unaware of the motion of the human torso in the middle of a step.

Remark 6. *The human would be controlling their torso angle to adjust their speed. In addition, the human is expected to use their arms and not hold it statically. This implies that the true human state $x_H \neq x_H^{\mathcal{O}}$. Still, it is believed that a sufficiently robust controller on the exoskeleton would keep the system stable, despite not having the light*

human upper body on their periodic orbit. In addition, with the exoskeleton accelerating or decelerating, the exoskeleton won't always be on its periodic orbit either. With that, it is assumed that the human would be able to track their desired torso pitch angles regardless of the state of the exoskeleton. Section 4.6 will show how the system behaves under small steady state errors in the torso tracking.

4. *Phase Variable*: The phase variable, τ is available to the exoskeleton controller. If one models the human's controller for posture and arm regulation as being synchronized with the gait, then one may assume that τ is also available to the human. Here, we use τ to generate a feedforward term for regulating the posture of the upper body (torso and arms).

5. *Velocity Regulation*:

   (a) In between steps, for purposes of computing q_T^d, the human has access to $y_1^a(x^-)$, the linearized hip velocity at the end of the previous step before impact.

   (b) In between steps, for purposes of computing δv, the exoskeleton has access to q_T^a, the relative torso angle at the end of the previous step.

6. *System Model*: The human and the exoskeleton controller have knowledge about the actual model of the system.

4.5.2 Decentralized Control Design

Note that, throughout this section, the indexing variable, $i \in \{H, E\}$ corresponds to the human or exoskeleton and the indexing variable, $k \in \{1, 2\}$ denotes relative degree one or relative degree two.

We begin by obtaining the desired outputs, v_d and $y_2^d(\tau, \alpha)$, representing a walking gait through constrained nonlinear optimization using direct collocation methods, detailed in Section 4.3.3, for the complete human-exoskeleton system.

The control equations we develop next closely follow that introduced in Section 4.3, with states $\mathfrak{x}_i$ and outputs y_i. In particular, we compute the vector fields, $f(\mathfrak{x}_i)$ and $g(\mathfrak{x}_i)$, outputs, $y_i = [y_{1,i}; y_{2,i}]$, and the Lie derivatives, $L_f^k y_{k,i}$ and $L_g L_f^{k-1} y_{k,i}$ corresponding to the human states, $\mathfrak{x}_H$, and exoskeleton states, $\mathfrak{x}_E$ where,

$$y_i = \begin{bmatrix} y_{1,i} \\ y_{2,i} \end{bmatrix} := \begin{bmatrix} y_1^a(\mathfrak{x}_i) - v_d \\ y_2^a(\mathfrak{x}_i) - y_2^d(\tau, \alpha) \end{bmatrix}.$$

Note that, to better use the available measurements of $\tau, \dot{\tau}$ as per Assumption 4 in Section 4.5.1, the decentralized controller computes the Lie derivatives as follows:

$$\begin{aligned}
L_f y_{k,i}(\tau, \dot{\tau}, \mathfrak{x}_i) &= L_f y_k^a(\mathfrak{x}_i) - \frac{\partial y_k^d}{\partial \tau}(\tau, \alpha)\dot{\tau}, \\
&= L_f y_k^a(\mathfrak{x}_i) - \dot{y}_k^d(\tau, \dot{\tau}, \alpha), \\
L_f^2 y_{2,i}(\tau, \dot{\tau}, \mathfrak{x}_i) &= L_f^2 y_2^a(\mathfrak{x}_i) - \frac{\partial \dot{y}_2^d(\tau, \dot{\tau}, \alpha)}{\partial \tau}\dot{\tau} - \\
&\quad \frac{\partial \dot{y}_2^d(\tau, \dot{\tau}, \alpha)}{\partial \dot{\tau}} L_f \dot{\tau}(\mathfrak{x}_i), \\
L_g L_f y_{2,i}(\tau, \dot{\tau}, \mathfrak{x}_i) &= L_g L_f y_2^a(\mathfrak{x}_i) - \\
&\quad \frac{\partial \dot{y}_2^d(\tau, \dot{\tau}, \alpha)}{\partial \dot{\tau}} L_g \dot{\tau}(\mathfrak{x}_i).
\end{aligned}$$

Two separate control inputs, for the human (u_H) and for the exoskeleton (u_E), are computed as follows,

$$\begin{aligned}
u_{IO,i} &= \mathcal{A}_i^{-1} \left(\begin{bmatrix} L_f y_{1,i}(\mathfrak{x}_i) \\ L_f^2 y_{2,i}(\mathfrak{x}_i) \end{bmatrix} + \begin{bmatrix} \mu_{1,i} \\ \mu_{2,i} \end{bmatrix} \right) \\
u_i &= T_i \cdot u_{IO,i}.
\end{aligned}$$

Here, $T_i : \mathbb{R}^m \to \mathbb{R}^{n_i}$, is a constant matrix that selects the control input corresponding to the human or exoskeleton joints from the full dimensional input, $u_{IO,i}$, where n_i is the number of actuators corresponding the human or exoskeleton and $n_H + n_E = m$. $\mathcal{A}_i = [L_g y_{1,i}(\mathfrak{x}_i); L_g L_f y_{2,i}(\mathfrak{x}_i)]$ is the decoupling matrix corresponding to the human or the exoskeleton controller and μ_i is a linear feedback controller,

$$\mu_i = \begin{bmatrix} \mu_{1,i} \\ \mu_{2,i} \end{bmatrix} := \begin{bmatrix} -\bar{k}_i^p y_{1,i} \\ -2k_i^p y_{2,i} - k_i^d \dot{y}_{2,i} \end{bmatrix}, \tag{4.13}$$

for $k_i^p, \bar{k}_i^p, k_i^d$ diagonal matrices with positive entries. The control input for the full state is then given by the augmented control input, $u = [u_H; u_E]$.

In the decentralized control design presented above, it can be noticed that the exoskeleton controller does not have information about the true states of the human and vice versa. This is important since, in a practical setting, it may not be feasible to obtain measurements of human states. However, it is unclear if this controller will result in a stable system since we do not use full state information. Thus, Section 4.6 covers numerical analysis using Poincaré maps to test stability and simulations to test robustness.

It is worth noting that the feedback linearizing control model for the human is not the only

controller that can be applied to the human subsystem. In addition, other control designs have been tested, such as using joint level PD controllers on the human subsystem, which resulted in similar results. A feedback linearizing design has been chosen for the human subsystem to maintain consistency and similarity between the centralized controller and decentralized controller approaches. Further, it is assumed that the exoskeleton controller is robust enough that it would remain stable and capable of regulating velocity so long as the human is capable of maintaining a sufficiently decent tracking of their desired torso angle by the end of each step. Section 4.6 will show results of the system being stable despite some steady state error in the human's torso tracking.

4.6 Gait Library Results and Discussion

Numerical simulations are performed to investigate and compare the performance of the stabilizing controllers developed in the previous Sections 4.3 till 4.5. In particular, the centralized and decentralized implementations of a controller for a fixed gait are first compared. The controllers are evaluated for their response to an initial condition that is not on the designed periodic orbit and for changes in the arm profiles adopted by the user. The point of the latter is that the user should not be constrained in how she moves her arms, while the optimization assumed the arms remain down and parallel to the body. Next, the centralized and decentralized implementations of speed regulation are compared over a varying speed profile. Finally, to test robustness of the proposed controller, the exoskeleton is simulated when walking up a ramp of a constant 5° slope. In all simulations, identical gains were used in the centralized and decentralized implementations of the controllers, and the same applies for walking on a flat surface versus a sloped surface.

4.6.1 Centralized vs Decentralized Implementations with a Fixed Gait

In this first simulation, the user is not allowed to adjust the speed setpoint of the closed-loop system. The controller is operated with a fixed gait corresponding to $v_d = 0.15\,\mathrm{m/s}$, which is held constant step to step; moreover the user is assumed to maintain an erect torso, with the arms held fixed in the downward position. The purpose is to compare the centralized controller, that comes from the bipedal robotics literature, to its decentralized implementation, that respects the amount of shared information that can reasonably be expected between the user and the exoskeleton. The controllers are implemented as in Section 4.5, with $q_T^d = 0°$ and the output of the velocity library held constant.

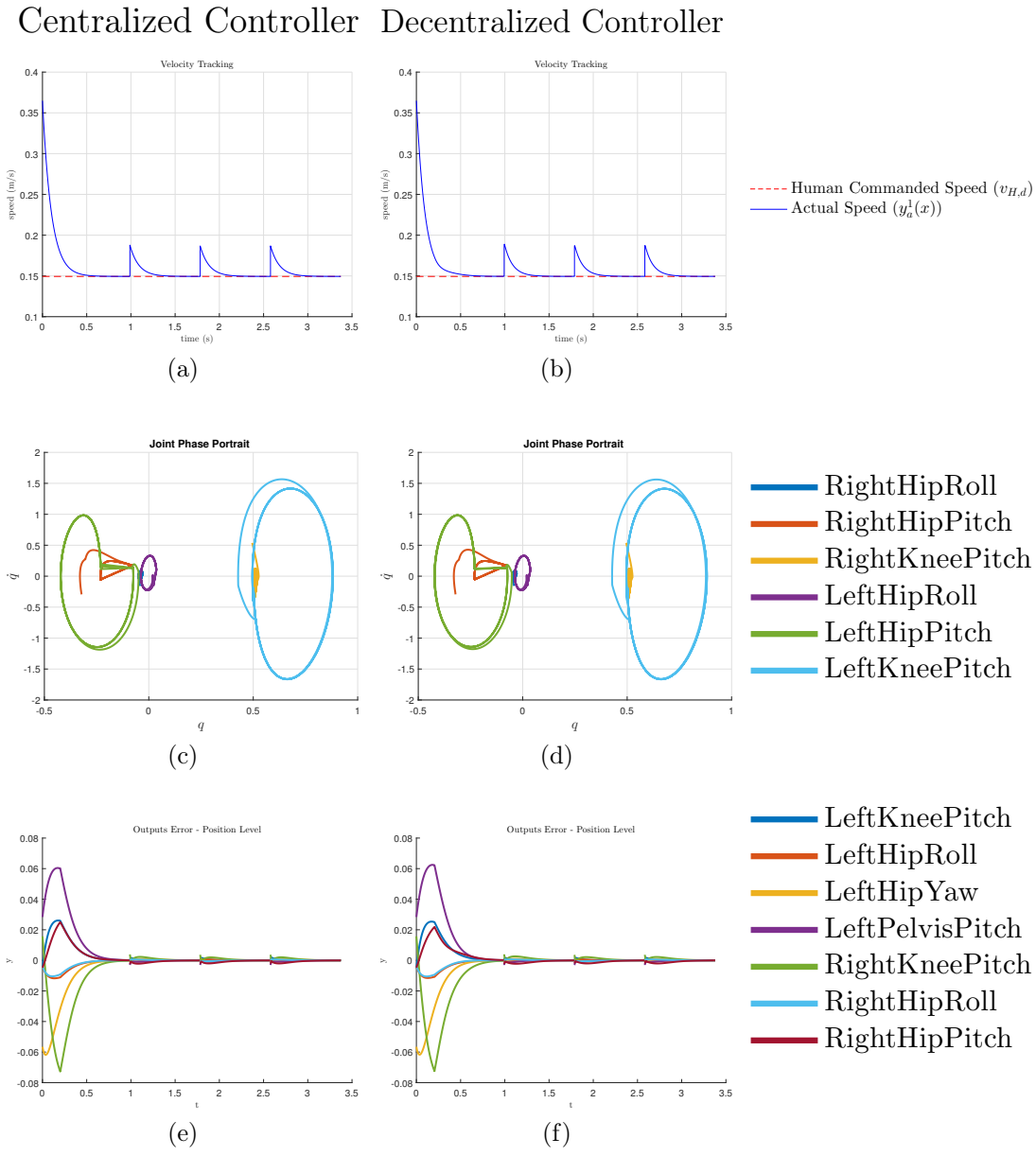


Figure 4.6: Comparison between the Centralized and Decentralized controller in handling a perturbation in the initial conditions of the exoskeleton states; recall that the user has an upright posture. Both controllers are implementing the same periodic orbit. Figures (a) and (b) show the velocity tracking. Figures (c) and (d) show phase portraits converging to periodic orbits. Figures (e) and (f) show output errors converging to zero.

To check the controllers' responses to off-orbit conditions, the initial condition of the exoskeleton is selected to be that of the periodic orbit of the gait for $v_d = 0.24\,\mathrm{m/s}$. Figures 4.6e and 4.6f shows both controllers driving the outputs nearly to zero before the end of each step. In addition, one can observe from the phase portraits in Fig. 4.6c and 4.6d that the periodic orbits of the closed-loop systems are indeed the same. In addition, it can be seen that both the centralized and decentralized controllers lead to a stable solution / behavior. We conjecture that their high similarity is because the perturbations are in the states of the exoskeleton. In addition, we conjecture that the upright posture of the user minimizes the effect of the user on the exoskeleton states.

4.6.2 Centralized vs Decentralized Implementations Velocity Regulation

In this next set of simulations, the user is allowed to adjust the speed setpoint of the closed-loop system. The exoskeleton is walking on flat ground. In the first evaluation, the user's arms are controlled to a downward position, corresponding to the assumption of the gait design, and in the next simulation, they are swinging.

Figures 4.7a and 4.7b show both controllers responding to step changes in desired speed. The centralized and decentralized controllers each converge to a periodic steady state; phase portraits are shown in Figures 4.7c and 4.7d. The steady-state speeds of the two controllers indicate that interpolated gaits are being employed. Figures 4.7e and 4.7f show that a key constraint, the ZMP, is being satisfied by the interpolated centralized and decentralized controllers. The other constraints, such as friction cone and torque bounds are also respected, though they are not shown here.

It is noted in Figure 4.7a that the centralized controller achieves zero steady-state tracking error in the hip velocity, whereas in Figure 4.7b, the decentralized controller sometimes has a non-zero steady-state error. A non-zero tracking error in the velocity occurs when the user's torso has a steady-state orientation error. In the highlighted section, the speed error of $0.028\,\mathrm{m/s}$ corresponds to a torso error of $0.36°$. The steady-state error is of no practical importance because it could be removed by augmenting the simple proportional controller proposed in (4.12) with an integral term, something the user would do through "intuition".

4.6.2.1 Effects of Arm Motion

Next, with the exoskeleton walking on level ground with the decentralized controller, the effects of the arms swinging $\pm 7.5°$ are evaluated; to be clear, the arm positions are unknown to the exoskeleton. The resulting velocity profiles are shown in Fig. 4.8a and 4.8b, where

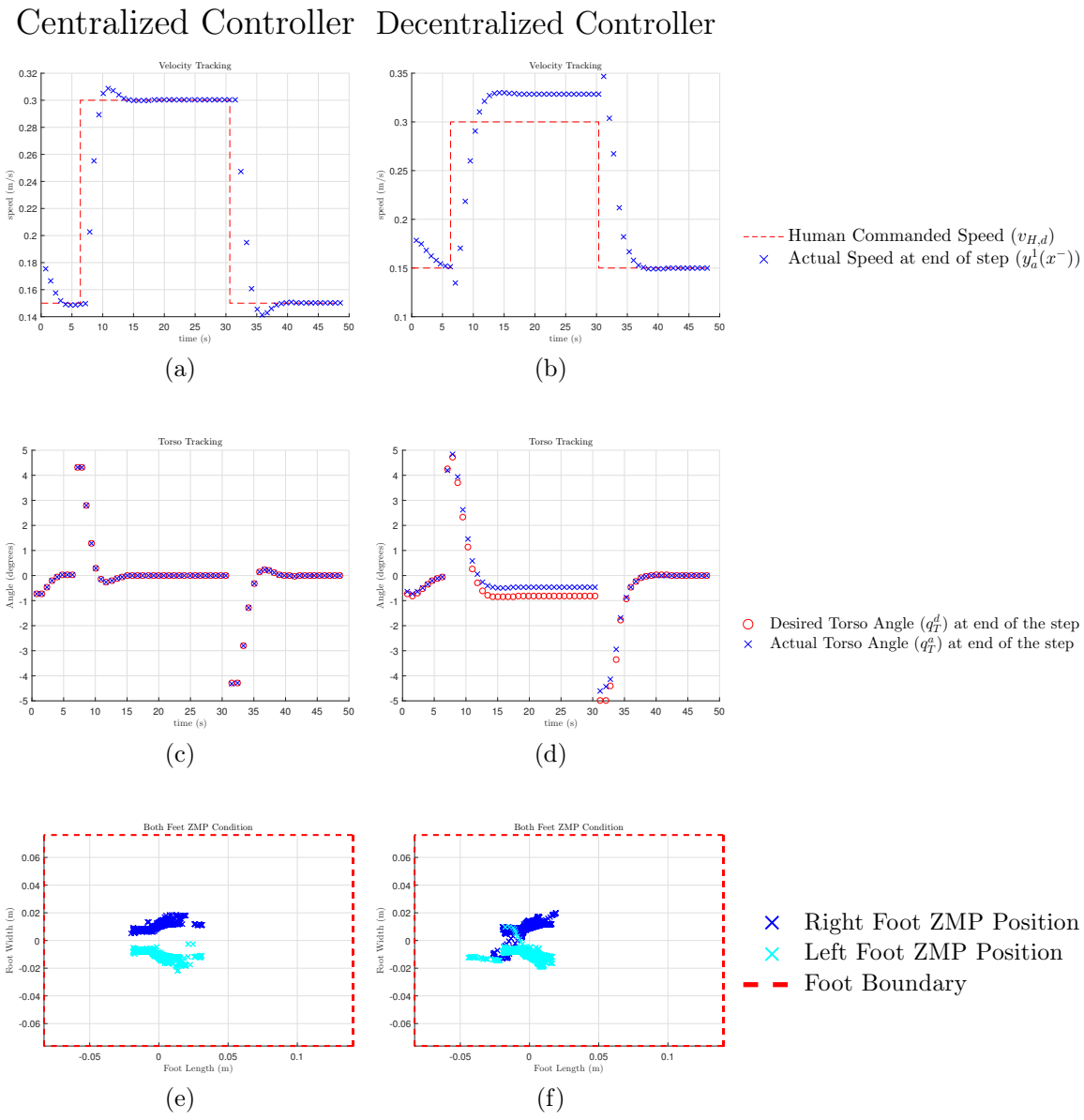
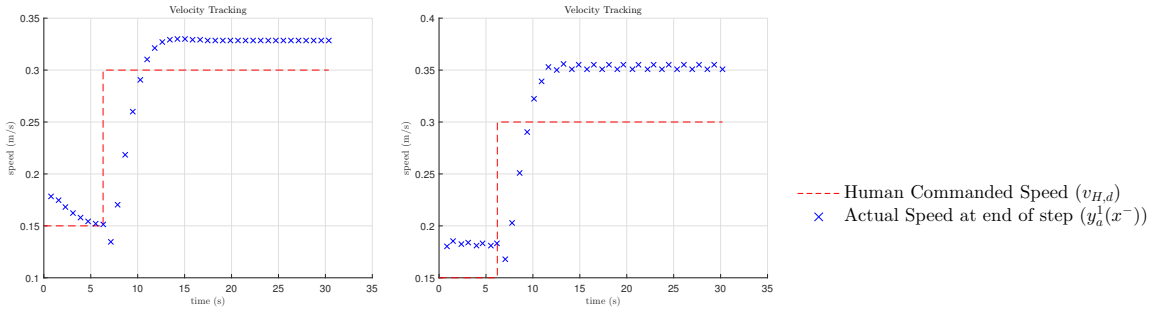
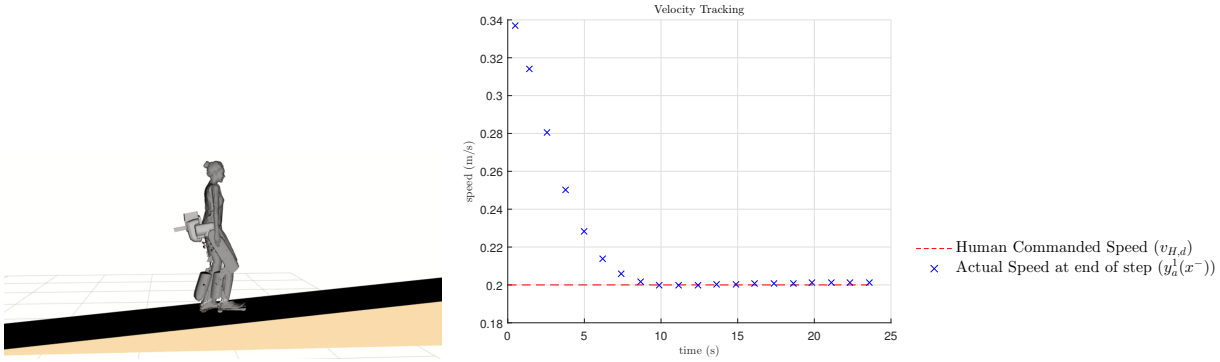


Figure 4.7: Comparison between Centralized and Decentralized controllers in tracking desired velocities. Figures (a) and (b) show the velocity tracking. Figures (c) and (d) torso tracking at the end of each step. Figures (e) and (f) show ZMP being within the foot's geometry.



(a) (b)

Figure 4.8: Comparison of the effect of arm swinging on the decentralized controller. Figure (a) shows the case with the arms stationary. Figure (b) shows the case with the arm swinging $\pm 7.5°$. In both cases, the exoskeleton controller assumes that the arms are stationary.



(a) (b)

Figure 4.9: Simulation results of walking up a $5°$ slope. Figure (a) shows a snapshot of the simulation. Figure (b) shows the velocity tracking of the closed-loop system.

a period-2 oscillation is observed when the arms are swinging in Fig 4.8b. The amount of perturbation seems unlikely to be significant to a user. Though not shown, the action of the arms' motion has increased the steady-state error in the user's torso position. This does increase the steady-state error in the velocity. An experienced user would remove the torso error and cancel the velocity error as well.

4.6.2.2 Walking up a Ramp

Now, with the arms fixed downward and using the decentralized controller, the closed-loop system is challenged with a ramp of 5° incline[2]. The same library of gaits optimized for flat terrain are used here. The only change made is that it is assumed that the exoskeleton can sense ground slope for the purpose of orienting the swing foot parallel to the ground.

Figure 4.9b shows that the exoskeleton is indeed capable of walking up the slope without modifying the controller. In addition, the achieved velocity is close to the desired value. In addition, the simulation satisfies ZMP, friction cone, and torque bound constraints. Walking up the slope has resulted in the step duration increasing from approximately 0.8 s to 1.24 s.

4.7 G-HZD Controller Architecture, Gait Objectives, and Important Constraints

After demonstrating generating a gait library based controller for the exoskeleton with a human in the loop, we want to develop a GHZD controller, from Section 2.5, where we're able to utilize trajectory optimization to capture more stabilizing behavior for the controller to use. For simplicity reasons, the following sections will be using the model of the exoskeleton with a rigid human attached to the torso, seen in Figure 4.10. The embedded control system needs to generate comfortable, robustly stable walking gaits that respect mechanical limits of the exoskeleton, such as joint and torque limits, initiate smooth (not jarring) foot contact with the ground, and satisfy ground contact constraints that avoid slipping. These requirements will be the main focus of the next few sections.

4.7.1 Gait Design Objectives

The gaits designed are destined for testing in a medical facility where an engineer or therapist will provide external commands for walking speed. Gaits will be designed here for walking

[2]According to the 2010 ADA Guideline, ramps are considered part of an accessible route if their slope is no steeper than 1 : 12, with exceptions [10]. 5° is very close to the 4.8° recommended by the guidelines.

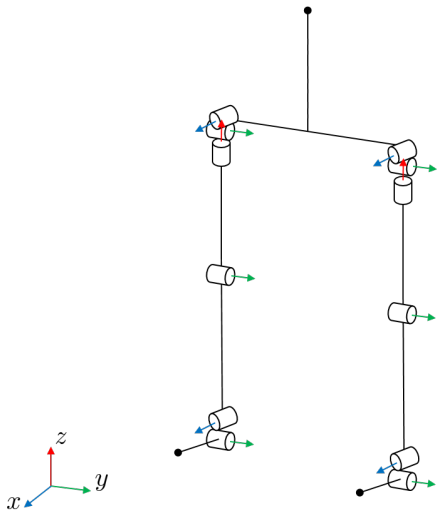


Figure 4.10: Exoskeleton with rigid human assumption.

in a straight line at speeds that vary from -0.3 m/s to 0.3 m/s. For comparison purposes, the relaxed human walking gait is approximately 0.9 m/s.

The time duration of a step will be set to 0.7 s. Our observation is that shorter step times closer to 0.5 s, while easier to stabilize, are uncomfortable for the user. To limit the transmission of vibrations from the exoskeleton to the user, the impact of the swing foot with the ground needs to be carefully regulated. Just before impact, we try to achieve near-zero forward and lateral velocities of the foot with respect to the ground, while the downward velocity of the foot is between -0.3 and -0.1 m/s. The upper bound ensures a transversal intersection of the foot with the ground, a key guard condition in the hybrid controller, while the lower bound is for user comfort.

It is desirable for the user to be able to maintain an upright posture when using the exoskeleton to limit demands on abdominal and dorsal muscles, which may have been weakened through prolonged use of a wheelchair. We have settled on left-right swaying motions that are less than two degrees, and a forward lean angle that is between two and six degrees. For user safety, the knee angles are bounded above five degrees away from straight and the ankles are limited to ± 23 degrees. Joint safety limits are imposed through a combination of hardware limits and software enforced limits.

4.7.2 Ground Contact

The holonomic constraints for modeling ground are taken from [46]. The key things to note are that the ground cannot pull on a foot and a "friction cone" must be respected to avoid

foot slippage, namely,

$$F_z > 0, \tag{4.14}$$

$$F_x^2 + F_y^2 \leq \mu^2 F_z^2, \tag{4.15}$$

where, $[F_x, F_y, F_z]$ is a vector of ground reaction forces acting on the stance foot, as shown in Figure 4.11, and $0 < \mu < 1$ is the coefficient of friction.

For the gaits used in this study, we will simplify the motions of the exoskeleton by imposing that the stance foot remains flat on the ground. As explained in [46], this requires moment constraints so that that foot does not roll about one of its axes. The stance width is set at 27 cm. This relatively wide stance limits rolling about the outer edge of the stance foot, which is typically harder to recover from than rolling inwards on the stance foot, while also promoting lateral stability. To provide additional robustness against foot rotation, we design the gaits so that the Zero Moment Point (ZMP) [113] lies in the shaded areas shown in Figure 4.11; though it is not exactly the same thing, for the purpose of this in this section, the reader can think of the ZMP as being the Center of Pressure (CoP) of the forces distributed on the sole of the foot. In addition, due to the relatively heavy battery pack mounted just behind the user's hips, the center of mass of the exoskeleton is towards the heel of the foot when the leg is straight. Designing gaits with the ZMP towards the forward section of the foot prevents the exoskeleton from rolling backward on its stance foot. In our experience, rolling forward on the foot has not been a concern.

4.7.3 Other Objectives

To reduce the possibility of the swing foot contacting the ground prematurely, gaits are designed with relatively large foot clearance in the middle of the step. The heel is designed to be 10 cm above the ground and the toe to be 5 cm above the ground. Larger foot clearance results in easier handling of terrain irregularities and trip recovery via foot placement. Potential downsides include greater torque requirements and motions that are closer to joint limits. Because flat walking is assumed, gaits need to be designed such that the feet are parallel to the ground during lift off and impact.

4.7.4 Controller Architecture

The overall control structure is shown in Figure 4.12. The Control Policy is responsible for specifying the evolution of key quantities of the exoskeleton, such as torso angle, swing leg angle (imagine a line from the hip to the ankle), and stance leg length (once again, a line

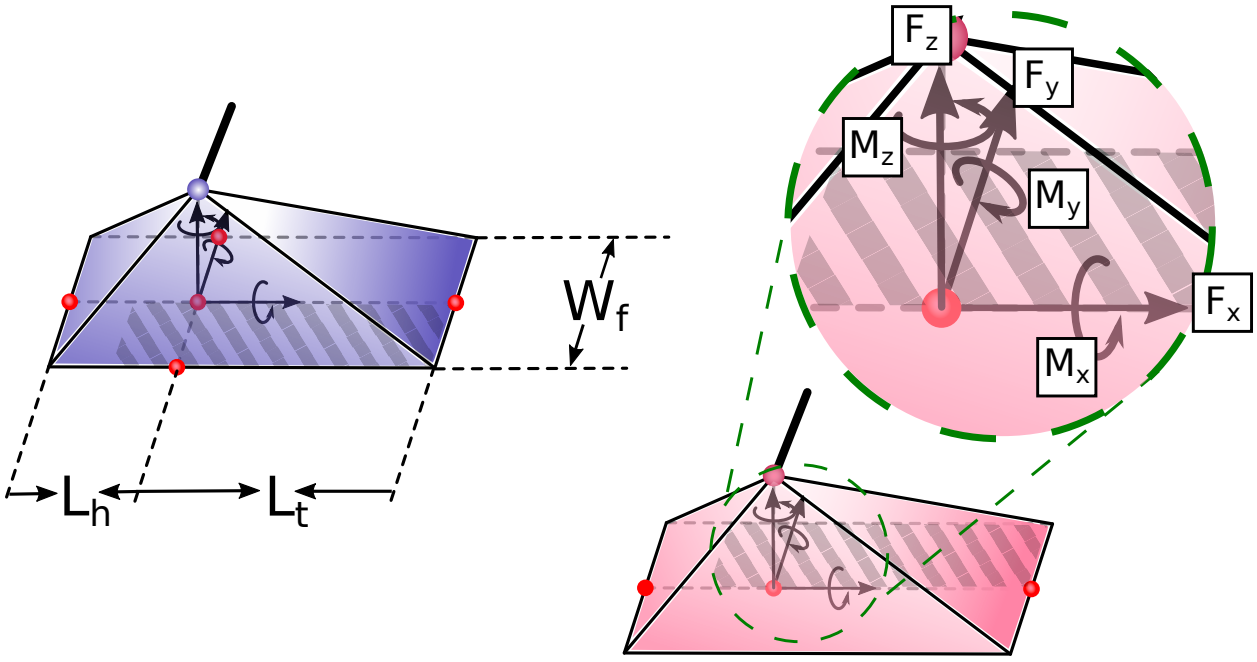


Figure 4.11: Depicts feet of the exoskeleton, with blue the left foot pink the right foot. L_h and L_t denote the distance from the point of reference to the heel and toe respectively. W_f denotes the width of the foot. Shaded area depicts the desired location of the Zero Moment Point (ZMP). $[F_x, F_y, F_z, M_x, M_y, M_z]$ is the resultant ground reaction force and moment acting on the foot during support, described in the body coordinate of the foot.

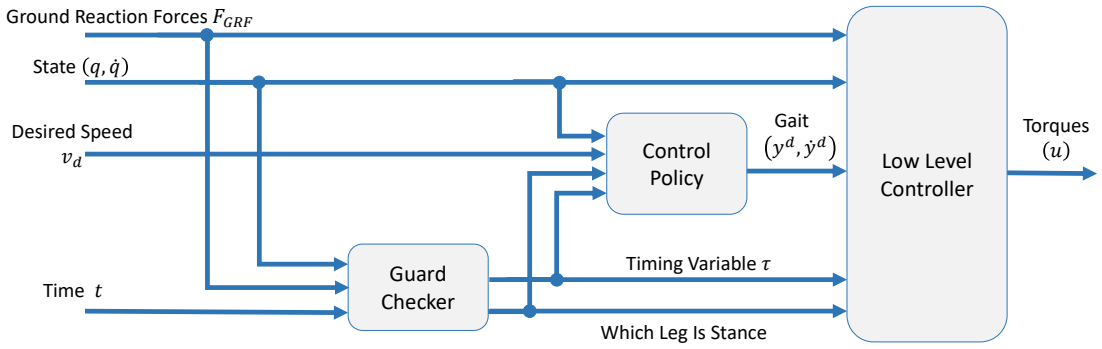


Figure 4.12: Overview of the controller structure. The Guard Checker handles detecting when to swap legs. The Control Policy specifies the desired gait and the means to achieve it. The Low-level Controller translates control policy commands to desired joint-level trajectories and achieves them via PD control.

from the hip to the ankle). These synthetic quantities are often more intuitive for the control engineer and the test engineer to use when specifying and discussing gait designs. It's design follows the approach described in Section 2.5.

The Low-level Controller is responsible for associating the synthetic high-level quantities to the individual actuators of the exoskeleton. During early testing, the simpler the low-level controller, the easier it is to make rapid changes and uncover bugs. The low-level joint controllers assure trajectory tracking with less than two degrees of error.

As its name suggests, the Guard Checker monitors quantities associated with events in a gait. In the real world, leg swapping is a control decision and not a discrete, capturable event as it would appear in an ideal simulator. The guard event for leg swapping is defined here in terms of step duration and measured vertical ground reaction force. In general, a gait timing variable is a strictly monotonically increasing quantity that varies from 0 to 1 over the course of a step. Here we use $\tau = \frac{t-t_0}{T_p}$, where t_0 is the starting time of the current step and T_p is the time duration of a step. The guard for leg swapping is then

$$\begin{aligned}(GRF^z_{swing} > GRF_{min} \text{ and } \tau > 0.5)\\ \text{or } \tau > 1,\end{aligned} \tag{4.16}$$

where GRF^z_{swing} is the measured vertical component of the ground reaction force acting on the swing foot and GRF_{min} is a chosen minimum threshold.

Table 4.3: Performance comparison between direct-collocation optimization vs classic shooting approaches on a 5-link planar biped (single-shooting) and a 7-link spring-leg planar biped, respectively.

Method	CPU time (s)
Single shooting (`fmincon`)	162.59
Direct collocation (`IPOPT`)	1.60
Method	**CPU time (s)**
Multiple shooting (`fmincon`)	5027.45
Direct collocation (`IPOPT`)	41.47

4.8 G-HZD: Harnessing the Power of Modern Optimization Techniques

When the HZD method were created, it would take several hours for the computation of a single periodic orbit. At that time, it was very important to be able to build a controller for rendering the periodic orbit locally exponentially stable directly from the orbit itself, without requiring further optimizations. The situation today is totally different as shown in Table 4.3. So, how to harness this power?

The control loops on the exoskeleton run at 1 KHz and the duration of a walking step is on the order of 500 ms to 750 ms. Hence, online Model Predictive Control (MPC) (i.e., iteratively solving in real time a finite-horizon constrained optimization problem) is simply not possible for models with twenty or more state variables. Explicit MPC is not possible either because one would have to numerically sample the state space, do the optimization offline, and then store the control actions for use online. A sparse uniform grid of ten samples per dimension would require 10^{20} optimizations; random sampling will provide a smaller, more effective discretization of the state space, but not enough is gained to handle $n \geq 20$. So, what to do?

Reference [29] introduces G-HZD, Generalized Hybrid Zero Dynamics, a new approach to control design for a class of high-dimensional nonlinear systems. As with HZD, the design process for G-HZD begins with trajectory optimization to design an open-loop periodic walking motion of the high-dimensional model. It differs from HZD in that it exploits the fact the trajectory optimization can be done rapidly to add to this periodic solution a carefully selected set of additional open-loop trajectories of the model that steer toward the nominal motion, thereby directly building in stability. A drawback of trajectories is that they provide little information on how to respond to a disturbance. To address this shortcoming, Supervised Machine Learning is used to extract a low-dimensional state-variable realization of the open-loop trajectories. The periodic orbit is now an attractor of the low-dimensional

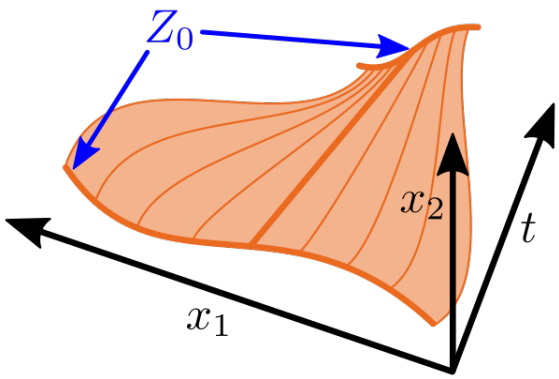


Figure 4.13: A collection of trajectories forming a smooth surface; while this is the desired outcome, it is not guaranteed. Direct-collocation-based optimization is used to design the trajectories.

state-variable model but is not attractive in the full-order system. The special structure of mechanical models associated with bipedal robots is used to embed the low-dimensional model in the original model in such a manner that the desired walking motions are locally exponentially stable.

In the following, we give the main ideas underlying G-HZD for models given by ordinary differential equations; the reader is referred to [29] for the small technical changes required to deal with hybrid models.

Step 1: Constructing Z_0, the boundary of the generalized HZD surface, from a collection of periodic orbits.

The dynamic model of the exoskeleton is decomposed into a low-dimensional weakly actuated portion corresponding roughly to the $x-y$-coordinates and velocities of the hips (in the world frame) and a strongly actuated portion of the model that captures the hips, knees, and swing ankle joints. Specifically, the dynamic portion of the hybrid model (2.2) is decomposed as

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, u_1) \quad x_1 \in \mathbb{R}^{n_1} \\ \dot{x}_2 &= f_2(x_1, x_2, u_1, u_2) \quad x_2 \in \mathbb{R}^{n_2}, \end{aligned} \tag{4.17}$$

where x_1 represents the "weakly actuated" portion of the model, u_1 are the stance ankle torques, x_2 captures the strongly actuated part of the model and the remaining actuators u_2. With this decomposition, $n_1 \ll n_2$; indeed, for the exoskeleton as modeled here in single support, $n_1 = 4$ and $n_2 = 20$.

Next, a library of gaits is constructed by uniformly discretizing a bounded set of initial conditions for x_1. Without loss of generality, each periodic gait is assumed to start from the origin, i.e., hip positions $(x; y)$ are at the origin. As a consequence, the periodic gaits are parametrized by the $(x; y)$-velocities of the hip, corresponding to walking forwards,

backwards, and sideways. Turning is not addressed presently but will be in the near future.

A nonlinear mapping is then constructed between the x_1 and x_2 states for the periodic orbits such that $x_2 = \gamma(x_1)$. The function γ is called an insertion map and can be constructed in several ways. One effective approach is to numerically fit a surface to the initial conditions of the periodic orbits through machine learning tools, resulting in

$$Z_0 := \{x =: (x_1; x_2) | x_2 = \gamma(x_1)\}, \tag{4.18}$$

as illustrated in Figure 4.13. In the ideal case, every point on the surface Z_0 corresponds to an initial condition for a periodic orbit.

Step 2: Constructing Z, the Generalized HZD Surface.

Assume now that one of the periodic orbits has been selected, corresponding to a point $\xi^* \in Z_0$. The process of stabilizing this particular periodic orbit is based on computing a judiciously selected set of open-loop trajectories of the full-order model that "approach" the periodic solution. Specifically, let B be an open ball about ξ^* and select a "contraction factor" $0 < c < 1$. For each initial condition $\xi \in B \cap Z_0$, a solution of the ODE (4.17) with initial condition ξ is sought that satisfies physical constraints, terminates in Z_0 at time $t = T_p$, and approaches the periodic solution as measured by $||\varphi_\xi(T_p) - \xi^*|| \leq c||\xi - \xi^*||$; transient solutions that approach periodic solutions are called transition gaits.

Denote the corresponding input and state trajectories parametrized by the initial condition $\xi \in B \cap Z_0$ as

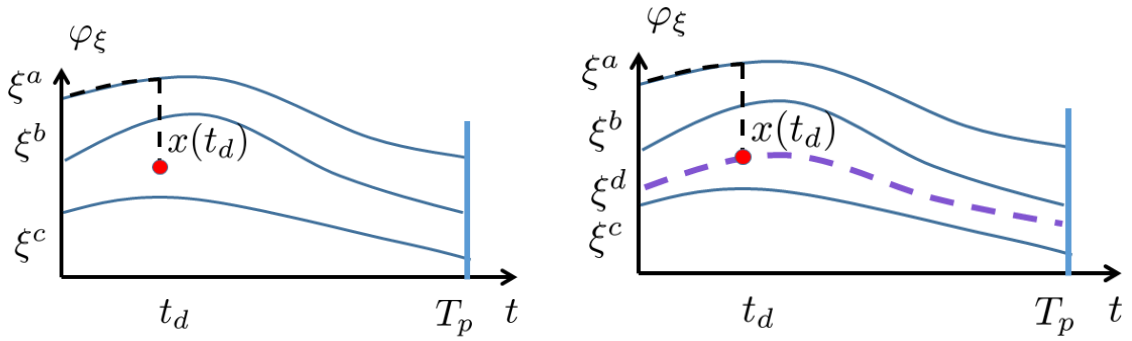
$$\begin{aligned} u_\xi &: [0, T_p] \to \mathbb{R}^m, \\ \varphi_\xi &: [0, T_p] \to \mathbb{R}^n. \end{aligned} \tag{4.19}$$

If all has gone well, the process has resulted in a smooth surface Z of dimension equal to the dimension of Z_0 plus one (due to time), as shown in Figure 4.13. As discussed in [29], such solutions may or may not exist and in principle could be very "ugly" functions of ξ (i.e., non-smooth, etc.).We note that because for all $\xi \in Z_0$, $\varphi_\xi(T_p) \in Z_0$, trajectories that start in Z_0 can be continued indefinitely.

Step 3: State-variable Realization of the Open-loop Trajectories Forming Z.

While the above process results in complete trajectories for the model, we do not directly use these for tracking because: (a) If the system starts on the surface Z, and a perturbation occurs between $t = 0$ and $t = T_p$, it would attempt to converge back to the potentially-far-away original-trajectory $\varphi_x i(t)$ until $t = T_p$, only at which point would a trajectory-based controller update the desired trajectory; and (b), if the system does not start on the surface, there is no obvious choice of a trajectory to follow.

We first address (a) by seeking a means to continuously update the desired evolution of



(a) Disturbance occurs at time t_d

(b) Proposed "course" correction

Figure 4.14: Gedanken Experiment. (a) Assume the system is initialized at ξ^a, with input, $u_{\xi^a}(t)$, being applied, and hence its solution is evolving along $\varphi_{\xi^a}(t)$. Suppose that at time t_d, an "impulsive" disturbance instantaneously displaces the solution to a point $x(t_d)$. What input should be applied? (b) If there exists an $\xi^d \in Z_0$ such that $x(t_d) = \varphi_{\xi^d}(t_d)$, then applying the input $u_{\xi^d}(t)$ for $t_d \leq t < T_p$ will move the system toward the equilibrium in the sense that $||\varphi_{\xi^d}(T_p)) - \xi^*|| \leq c||\xi^d - \xi^*||$. This thought process leads to the "interpolation" or "learning" conditions in (4.21).

the system so as to immediately respond to a perturbation. In fact, we will design a low-dimensional differential equation that evolves on the surface and has the desired periodic orbit as its locally stable and attractive limit cycle. A solution to (b) will be given in Step 5.

Figure 4.14 motivates a condition for "automatically re-planning" so as to respond to a disturbance, namely

$$x(t) = \varphi_\xi(t) \implies u(t, x(t)) = u_\xi(t). \tag{4.20}$$

This is an implicit interpolation condition for specifying a control response at each point of the surface Z in Figure 4.13. As explained further in [29], a solution to (4.20) can be constructed if one can find two T_p-periodic functions, ν and μ, satisfying the following conditions on the trajectory data: for all $0 \leq t < T_p$

$$\begin{aligned} \nu(t, \pi_1 \circ \varphi_\xi(t)) &= \pi_2 \circ \varphi_\xi(t) \\ \mu(t, \pi_1 \circ \varphi_\xi(t)) &= u_\xi(t), \end{aligned} \tag{4.21}$$

where $\pi_i : \mathbb{R}^n \to \mathbb{R}^{n_i}$ are the canonical projections (i.e., $\pi_1(x_1, x_2) = x_1$ and $\pi_2(x_1, x_2) = x_2$). Moreover, it can then be shown that (i) the trajectories of Figure 4.13 are solutions of the reduced-order model

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, \nu(t, x_1), \mu_1(t, x_1)), \\ x_2 &= \nu(t, x_1), \end{aligned} \tag{4.22}$$

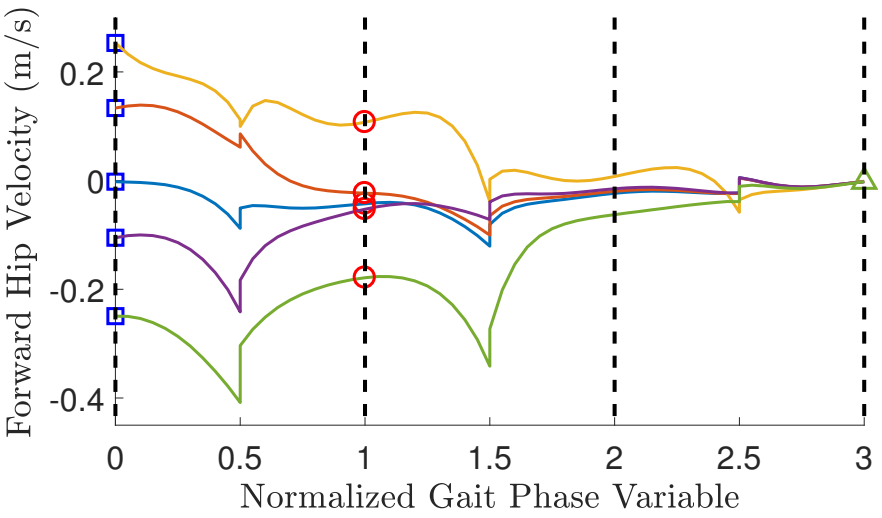


Figure 4.15: Shows forward velocity of the hip resulting from the transition optimizations. Black vertical dashed lines mark the midpoint of each (robot) step in the optimization, whereas the discontinuities (jumps) in velocity between two dashed lines arise from the rigid impact at the end of each step. Squares, circles, and the triangle mark points where the state of the system is in Z_0. The blue squares mark starting states obtained from periodic gaits while the green triangle marks the ending desired state also obtained from a periodic gait. The red circles were made to be in Z_0 through an optimization constraint. The gait trajectory from the blue square to the red circle is what is used as training data for the Supervised Machine Learning.

and that (ii) the periodic orbit $\varphi_{\xi^*} : [0, T_p) \to \mathbb{R}^n$ is locally exponentially stable. In fact, by construction, all initial conditions $\xi \in Z_0$ for which feasible model solutions have been found result in trajectories that converge to the periodic orbit; hence, as one builds the feasible solutions in (4.19), one is constructing the domain of attraction of the periodic orbit in the G-HZD surface, Z.

Figure 4.15 shows the sagittal-plane hip velocity for several initial conditions in Z_0. Two things are important to note: (a) the convergence to the nominal orbit; and (b), the trajectories are feasible solutions of the full-order model.

Next we deal with how to find functions satisfying the conditions in (4.21).

Step 4: Supervised Machine Learning to Extract ν and μ from Optimization Data

The importance of finding a differential equation (i.e., vector field) realization of the trajectories in Figure 4.13 is that the differential equation is an automatic, instantaneous re-planner of the system's evolution when it is perturbed off a nominal motion. Figure 4.16 shows such a realization. Solving for the functions in (4.21) is the key to constructing the vector field. Doing so analytically would be a challenge at best, and this is where Supervised Machine Learning comes into play. Table 4.4 shows a vanilla implementation where the

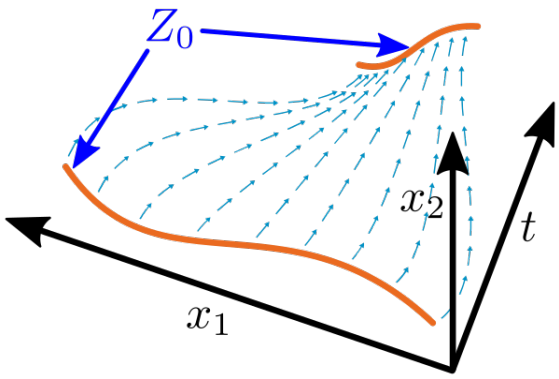


Figure 4.16: A vector field is constructed that gives rise to the trajectories, so it is a state-variable realization. Supervised Machine Learning (fancy regression) and model structure are used to "extract" the vector field from the trajectory optimization data.

features are selected as time and the x_1-states and the *labels or targets* are taken as the inputs u and the x_2-states. In the control implementations simulated here, the features are taken as the Cartesian hip velocities only (the positions are discarded) and the labels are taken as the outputs listed in Table 4.5.

At this point we have all the functions we need to implement a control policy that locally exponentially stabilizes the selected periodic walking gait. Figure 4.17 shows one component of the function ν arising from the Supervised Machine Learning.

Step 5: Feedback Control to Render Z Attractive.

The previous steps created a low-dimensional dynamical model for which the desired periodic orbit with initial condition $\xi^* \in Z_0$ is locally stable and attractive in Z. The next step is to stabilize the orbit in the full model, as illustrated in Figure 4.18, and for this we need to be more specific about the x_2-portion of the model. In the case of the exoskeleton, the strongly actuated part of the model is fully actuated and is therefore feedback linearizable [99]. In particular, the x_2-part of the model (4.17) can be expressed as

$$x_2 = \begin{bmatrix} x_{2a} \\ x_{2b} \end{bmatrix} \text{ and } f_2 = \begin{bmatrix} x_{2b} \\ \alpha(x_1, x_2) + \beta_1(x_1, x_2)u_1 + \beta_2(x_1, x_2)u_2 \end{bmatrix}, \tag{4.23}$$

where β_2 is square and invertible.

In [29], it is shown that for all $\frac{n_2}{2} \times \frac{n_2}{2}$ positive definite matrices K_p and K_d, $\varphi_{\xi^*} : [0, T_p) \to$

Table 4.4: Conceptual arrangement of the optimization data from which appropriate functions for building a stabilizing controller can be determined by "regression". Here ξ^i, $1 \leq i \leq M$ is a discretization of Z_0, t_j, $1 \leq j \leq N$ is a discretization of $[0, T_p]$, $\varphi_\xi(t)$ is the solution of the full model with the initial condition ξ at $t = 0$, π_1 is the projection onto the x_1 coordinate, π_2 is the projection onto the x_2 coordinates, and $u_\xi(t)$ is the input giving rise to the solution $\varphi_\xi(t)$. Standard toolboxes in MATLAB are used to build the functions and even to test for their existence via the quality of fit tools.

Features		**Labels or Targets**	
t_j	$x_1^{j,i} = \pi_1 \circ \varphi_{\xi^i}(t_j)$	$\nu^{j,i} = \pi_2 \circ \varphi_{\xi^i}(t_j)$	$\mu^{j,i} = u_{\xi^i}(t_j)$
$t_0 = 0$	$x_1^{0,1}$	$\nu^{0,1}$	$\mu^{0,1}$
$t_0 = 0$	$x_1^{0,2}$	$\nu^{0,2}$	$\mu^{0,2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_0 = 0$	$x_1^{0,M}$	$\nu^{0,M}$	$\mu^{0,M}$
t_1	$x_1^{1,1}$	$\nu^{1,1}$	$\mu^{1,1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
t_1	$x_1^{1,M}$	$\nu^{1,M}$	$\mu^{1,M}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_N = T_p$	$x_1^{N,1}$	$\nu^{N,1}$	$\mu^{N,1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$t_N = T_p$	$x_1^{N,M}$	$\nu^{N,M}$	$\mu^{N,M}$

Table 4.5: Table showing a mapping between higher-level control objectives and their realization at a lower-level.

Outputs	Actuated Joints
Torso Pitch, Roll, and Yaw orientation	Stance Hip Pitch, Roll, and Yaw Joints
Stance Knee Joint	Stance Knee Joint
Swing Knee Joint	Swing Knee Joint
Stance Pitch Ankle Joint	Stance Pitch Ankle Joint
Stance Henke Ankle Joint	Stance Henke Ankle Joint
Swing Hip Pitch Joint	Swing Hip Pitch Joint
Swing Hip Roll Joint	Swing Hip Roll Joint
Swing Foot Pitch, Roll, and Yaw orientation	Swing Pitch and Henke Ankle Joints, and Swing Hip Yaw Joint

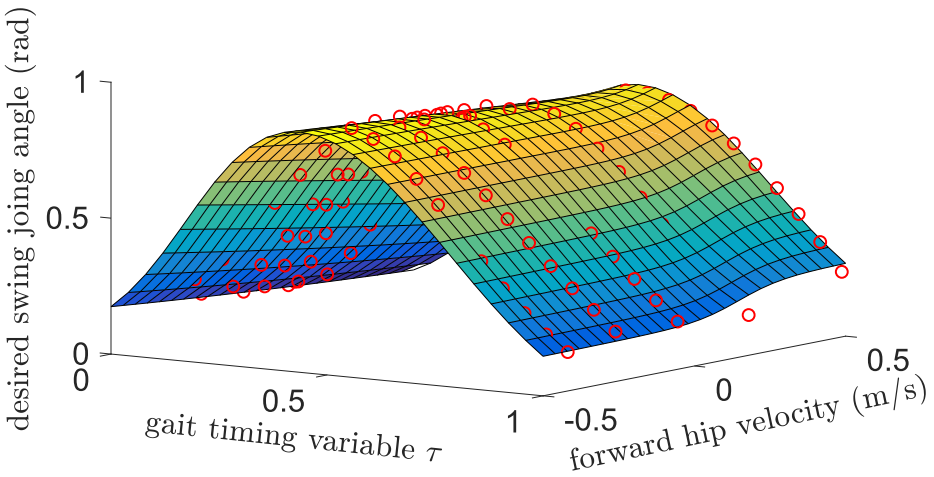


Figure 4.17: Surface showing the desired swing knee angle as a function of the gait timing variable τ and the forward hip velocity for stabilizing the step-in-place gait. The red circles are the actual data obtained from optimization while the surface is a regression done using the Neural Network Toolbox in MATLAB. Note that some of the circles are under the surface and are therefore not visible.

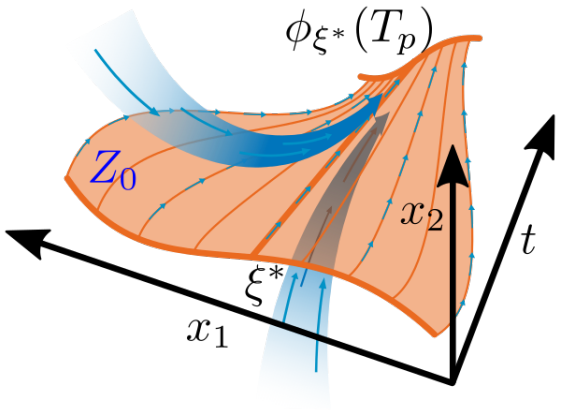


Figure 4.18: The low-dimensional realization (invariant surface and vector field) being rendered attractive in the full order model.

$\mathbb{R}^n$ is a locally uniformly exponentially stable solution of the closed-loop system

$$\begin{aligned}
\dot{x}_1 &= f_1(x_1, x_2, u_1) \\
\dot{x}_2 &= f_2(x_1, x_2, u_1, u_2) \\
u_1 &= v_1 \\
u_2 &= [\beta_2(x_1, x_2)]^{-1} \big(-\alpha(x_1, x_2) - \beta_1(x_1, x_2)u_1 + v_2 \big) \\
\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} &= \mu(t, x_1) - [K_p \;\; K_d]\big(x_2 - \nu(t, x_1)\big).
\end{aligned} \tag{4.24}$$

Step 6 (Optional): Enriching the Control Policy to Handle Multiple Gaits

The final step performed with the ATALANTE model is to repeat the above processes for a grid of periodic gaits corresponding to a range of walking speeds captured in Z_0. The trajectory designs for these new gaits are catenated to Table 4.4, with the only change being that the feature set is augmented to include the designed average (Cartesian) velocity of each gait. Supervised Machine Learning, if successful, then produces a control policy that allows a desired walking speed to be selected.

4.8.1 Remark on Trajectory Design via Optimization

We provide a few details on how to actually generate the trajectories in (4.19) that form the surface in Figure 4.13. We use optimization to determine solutions of the model; see [55] for more details on the optimization method itself. In particular, a cost function of the form

$$\begin{aligned}
J(\xi) =& \min_{u_\xi} \int_0^{3T_p} L(\varphi_\xi(\tau), u_\xi(\tau))d\tau \\
\text{s.t. } \varphi_\xi(t) =& \xi + \int_0^t f(\varphi_\xi(\tau), u_\xi(\tau))d\tau \\
0 \geq& c(\varphi_\xi(t), u_\xi(t)), \;\; 0 \leq t \leq 3T_p \\
\varphi_\xi(T_p) \in& Z_0, \quad ||\varphi_\xi(T_p) - \xi^*|| \leq c||\xi - \xi^*|| \\
\varphi_\xi(3T_p) =& \xi^*
\end{aligned} \tag{4.25}$$

with constraints is posed over a time horizon of three steps such that (i) energy per step taken is penalized, (ii) solutions satisfy the full-order model, (iii) key constraints on ground reaction forces and actuator limits are respected (captured in the function $c \leq 0$), (iv) solution returns to Z_0 in one step so that trajectories can be continued indefinitely, while approaching the desired periodic orbit, and (v) terminates at the nominal periodic orbit in three steps (this could be replaced by a terminal penalty in the cost function). The choice

of three (robot) steps in the optimization is based on the capture-point analysis in [123] and theoretical work in [29]. $L(\varphi_\xi(\tau), u_\xi(\tau))$ defines the running cost while $f(\varphi_\xi(\tau), u_\xi(\tau))$ is the dynamics of the system. Figure 2.4 earlier in Section 2.5 showed a conceptual representation of the gait design process.

4.9 G-HZD Numerical Illustration

This section presents simulation results of the G-HZD controller developed in the previous section. The simulation experiments were done with Gazebo using the full dynamical model of the exoskeleton with a human in it. The ground contact model in Gazebo allows the possibility for the feet to roll and slip. In these tests, the machine learning approach based on G-HZD was used to stabilize walking motions with a longitudinal speed range of -0.3 m/s to 0.3 m/s.

4.9.1 Velocity Tracking

Figure 4.19 shows a simulation where the exoskeleton starts with stepping in-place at $t = 0$ s, then at $t = 3$ s, the exoskeleton is commanded to walk at a speed of 0.15 m/s, until at $t = 10$ s, the exoskeleton is commanded to return to a stepping in place gait. It can be seen that there is a small steady-state error when forward walking is commanded; this can be attributed to a combination of joint tracking errors from the low-level PD-controllers and the compliant (non-rigid) ground being different from the model used for control design. Figure 4.20 shows the gait timing variable exhibiting a typical triangle-wave pattern, with leg swapping happening before $\tau = 1$. It can be seen that regardless of whether the system was in a periodic gait or was transitioning between speeds, τ consistently terminates at around 0.9. This is due to having trained the system on various transition gaits. The phase portrait given in Figure 4.21 shows the periodic nature of stepping in-place, followed by a transient to another periodic condition. Convergence to periodic motion is clear during periods when the speed command is constant. Stick figures showing stepping-in-place, transitioning, and walking forward can be seen in Figure 4.22. An animation of the simulation result can be seen in [9].

4.9.2 Preliminary Robustness Analysis

It is easy to imagine scenarios that could lead to an exoskeleton losing stability and "tripping up", including unexpected contact with objects in the environment, walking over uneven terrain, external force perturbations being applied on the system, or spasticity (i.e., involuntary

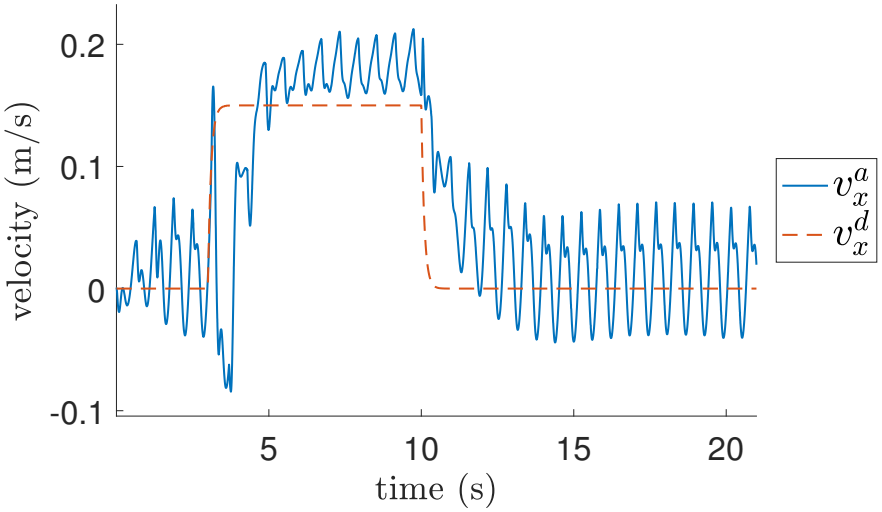


Figure 4.19: Shows tracking of a desired velocity profile. v_x^a is the actual forward velocity of the pelvis during walking. v_x^d is the commanded average forward velocity.

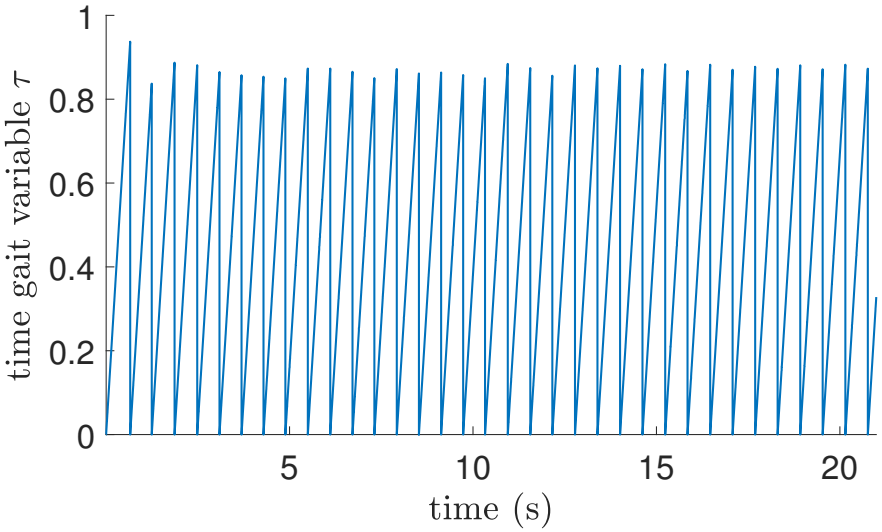
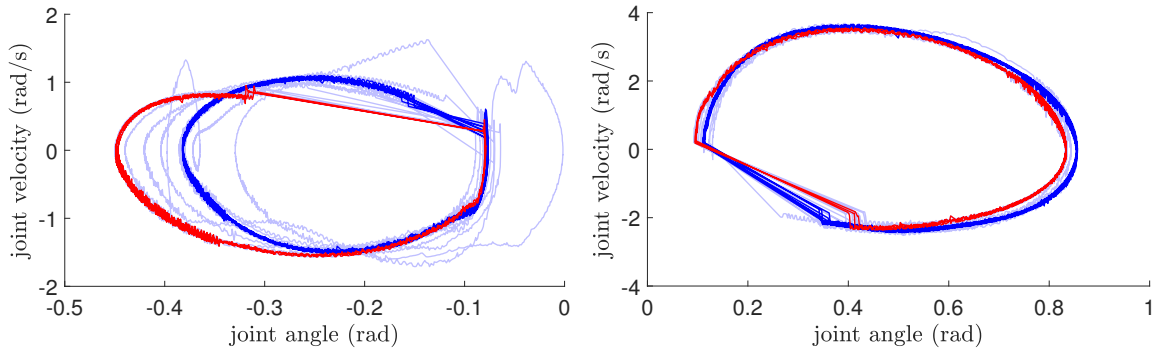


Figure 4.20: Shows the gait timing variable τ changing as a function of time while the exoskeleton was following a varying desired speed.



(a) Swing Sagittal Hip Phase Portrait (b) Swing Knee Phase Portrait

Figure 4.21: The phase portrait of both the swing and stance sagittal hip joints for when the system was tracking a desired velocity profile. Shows q vs $\dot{q}$ for the two joints. The solid blue line shows the states at the last quarter of the simulation. While the solid red line shows the part of the simulation where the exoskeleton reached a steady state trajectory while walking forward. It can be seen that both cases, the exoskeleton converged to a periodic cycle.

muscle resistance to patient's leg motion). In previous work, we evaluated in simulation a PHZD-based controller of the exoskeleton in the face of unexpected slopes and unplanned upper-body motion [13]. Here, we present a preliminary robustness analysis of the machine learning controller under various external and internal perturbations.

Velocity Perturbations. Analysis in [44] shows that a bipedal robot's ability to reject velocity perturbations when using HZD-based control strategies correlates with its ability to reject a set of other perturbations, such as variations in terrain height. Based on this observation, it is posited that velocity perturbation rejection provides a reasonable preliminary test of the robustness of the proposed controller framework. In this test, we induce velocity perturbations by applying impulsive external forces of various magnitudes and directions to the exoskeleton while it is walking. We compare the response of two different closed-loop configurations: one with a controller using the machine learning and another with a fixed single periodic gait. The external force is applied at the hip of the exoskeleton during the second step for 0.1 seconds in either forward or backward directions. When the exoskeleton is commanded to step in place, the results of Gazebo simulation are shown in Figure 4.23 and Figure 4.24. In particular, using the machine learning controller, the exoskeleton is able to recover from up to 750 N force in the forward direction and 650 N force in the backwards direction, whereas, when using a controller for fixed periodic gait, the system lost stability with any forward force larger than 300 N or a backward force larger than 100 N. Figure 4.23 shows the changes in the forward hip velocity of the exoskeleton for both machine learning and fixed-gait methods. It is seen that the Supervised Machine Learning controller is capable of recovering for relatively large velocity disturbances arising from external forces.

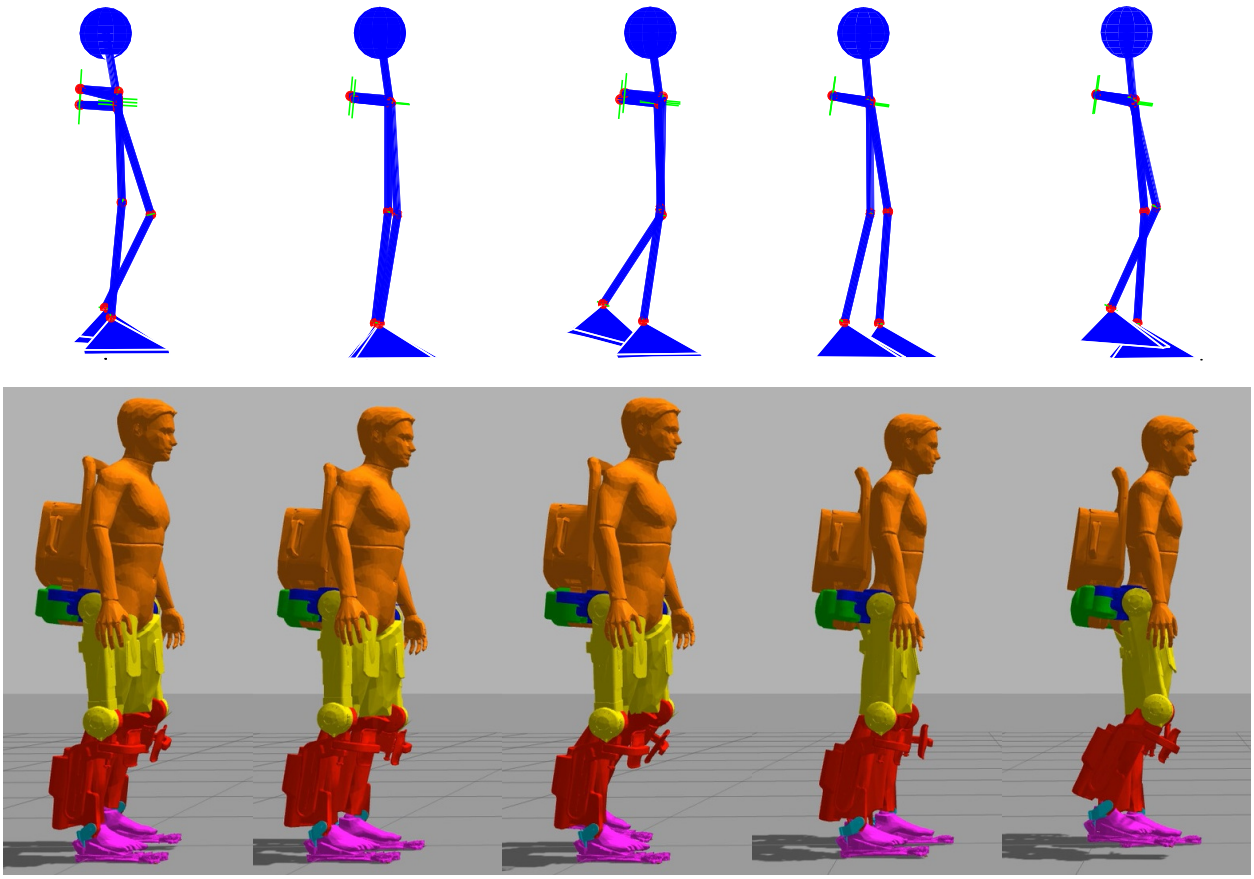
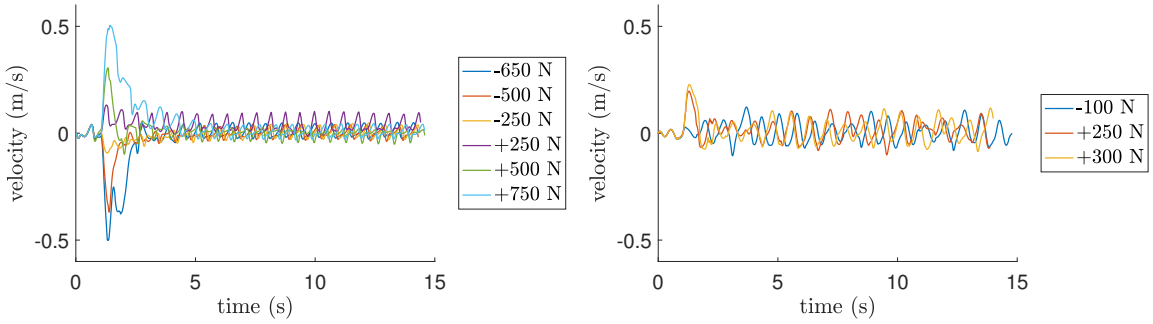
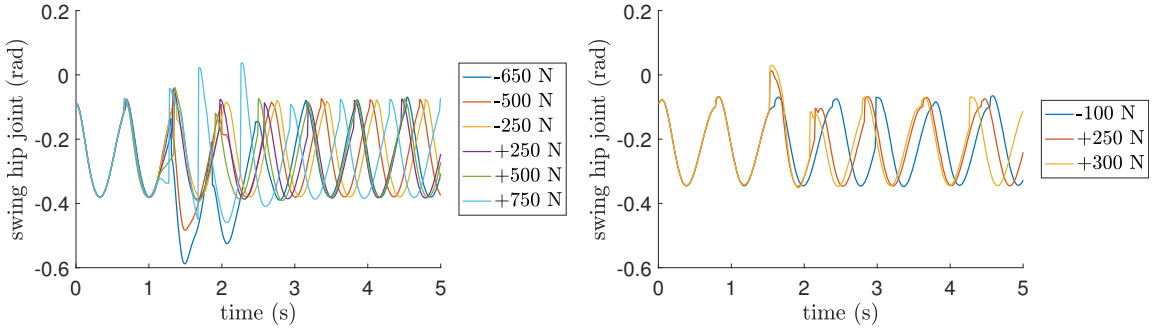


Figure 4.22: Outtakes from animations of the exoskeleton tracking a desired velocity profile. Below are images from the Gazebo simulation, while above are stick figures with the same motion added for clarity. The first two photos show the step-in-place gait. The third image shows how the system looks like when it is starting to transition. The last two show the exoskeleton walking forward in a periodic gait.



(a) Supervised Machine Learning (b) Fixed Gait

Figure 4.23: Change in hip velocity due to a perturbation. The control algorithm based on Supervised Machine Learning is able to recover from larger perturbations. Both controllers failed under perturbations larger than those shown.



(a) Supervised Machine Learning (b) Fixed Gait

Figure 4.24: The swing hip joint angle as a function of time while responding to a perturbation. When experiencing large perturbations, the controller based on Supervised Machine Learning makes more use of its swing foot in order to return to a stepping-in-place gait.

In addition, Figure 4.24 shows how the controllers respond to the external disturbances by changing the desired trajectories of the exoskeleton joints. Specifically, the machine learning controller, when challenged with a large perturbation, extends the swing leg outward by modifying the desired trajectory of the sagittal swing hip joint. This behavior occurs naturally from training the control surface on optimized walking gaits for various speeds and transitions among them. The fixed-gait controller only uses the large feet of the exoskeleton (regulating the ZMP) to reject the perturbation instead of adjusting the step length. An animation of the velocity perturbation simulations in Gazebo can be seen in [9].

Constant Force Perturbation. While studying velocity perturbation under impulses captures short duration events, such as someone or something bumping into the exoskeleton, it is also interesting to investigate the effects of persistent external forces applied to the system. Figure 4.25 shows the average speed when a constant force is applied to the torso,

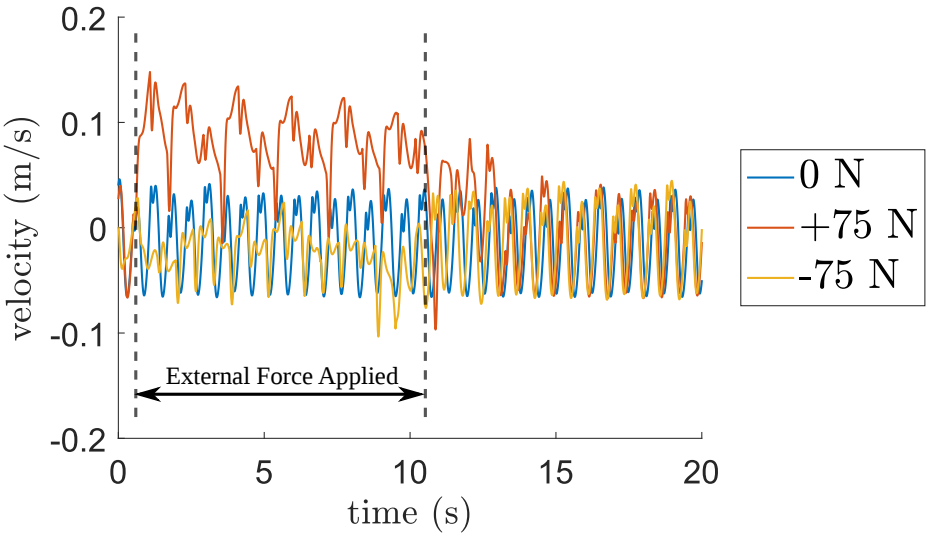


Figure 4.25: Instantaneous velocity of the hip when system is perturbed for a duration for 10 seconds. The forward push causes the system to drift slightly more while the force is being applied. When force is removed, the velocity converges back to the nominal velocity.

Table 4.6: The mean-absolute error (MAE) for joint angle tracking when applying external torques on the right knee joint representing a simplified model of spasticity.

Test	**MAE (rad) at steady-state**
No perturbations	0.0130
Constant +100 Nm torque	0.0166
Constant -100 Nm torque	0.0180
Sinusoidal 100 Nm torque at 1 Hz frequency	0.0147
Sinusoidal 100 Nm torque at 10 Hz frequency	0.0154

0.5 m above the pelvis, under the controller based on Supervised Machine Learning. The exoskeleton is able to withstand a constant force of ± 75 N for a duration of 10 seconds. Small drifts in velocity are noticed with the forces applied. The velocities converge to the nominal velocity after the force is removed. Larger forces result in loss of stability.

Unplanned forces in the user's legs. A simplified model of spasticity is represented by torques at a user's knee joints. While spasticity occurring simultaneously in both legs was investigated, only spasticity in the right leg is reported here. Both constant and sinusoidal torques are applied, and these are added to the joint-side torque provided by the knee motor and gearing. The mean-absolute error (MAE) in joint tracking is computed for the right knee. It is observed that the MAE increases only slightly, as shown in Table 4.6. In addition, no differences in the gaits is visually observed.

CHAPTER 5

From Partial-GHZD to Full-GHZD in High Dimensional Models

5.1 Motivation

One of the limitations of the G-HZD method as introduced in [29], and as described in Section 2.5 and Section 4.8, is that no concrete design rules have been given for determining the boundary manifold, Z_0, from which the design process for constructing the generalized zero dynamic manifold begins! In fact, the existing design method is not specific to the given periodic orbit that one is seeking to stabilize. Instead, the surface is indirectly constructed on the basis of a family of "steady-state" behaviors of the system. The design rule proposed in [29] consists of first constructing a library of periodic orbits, with a common period $T > 0$, parameterized by a physically meaningful parameter; in the case of a biped, the parameter was taken as walking speed. The designer should then select a time $0 \leq t_0 \leq T$ (for bipeds, t_0 corresponds to the middle of the step). Z_0 is then taken as the surface swept out by the periodic orbits at time t_0 while the physical parameter is varied.

Choosing Z_0 on the basis of a family of periodic orbits obliges the controller to return the system every T seconds to a surface traced out by nominally "equilibrium-like" behaviors. In particular, it prevents the system from taking advantage of more aggressive recovery approaches tailored to the particular orbit that one seeks to stabilize. Section 5.4 will highlight this limitation through an example.

A second limitation of the design method of [29] is that for high dimensional systems such as MARLO, the surfaces Z_0 and Z were parameterized using only the robot's hip velocity and ignored torso lean angle, for example. Hence the approach ignores state information that can increase the region of attraction of a given periodic orbit. In [29, Remark 5], the above indirect approach to designing the surface Z_0 was justified by the difficulty of defining an effective insertion map for high dimensional systems, such as Cassie.

We will present an alternative, systematic approach that won't force the closed-loop system to periodically pass through relative equilibrium states while recovering from perturbations. In addition, the new approach will allow a richer set of feedback controllers to be designed. In particular, for a biped, it will allow all of a robot's weakly actuated states (i.e., those not directly associated with powerful actuators, such as CoM position and velocity) to be used when developing "virtual constraints" for the strongly actuated states (i.e., those states directly associated with powerful actuators, such as knee angles).

5.2 Recalling the System Model

As in [29], we consider a nonlinear system

$$\dot{x} = f(x, u). \tag{5.1}$$

In addition, we'll have a periodic system with a period $T > 0$ and an orbit $x_n(t), u_n(t), t \in [0, T]$ and $x_n(t) = x_n(t + T), u_n(t) = u_n(t + T)$.

Referring back to Section 2.5, the model (5.1) should admit a set of coordinates

$$x = \begin{bmatrix} x_w \\ x_s \end{bmatrix}$$

in which it can be expressed as

$$\begin{aligned} \dot{x}_w &= f_1(x_w, x_s, u) \\ \dot{x}_s &= f_2(x_w, x_s, u), \end{aligned} \tag{5.2}$$

where, x_w is the underactuated (or weakly actuated) component of the system, and x_s is the actuated (or strongly actuated) component of the system. Compared to (4.17), u_1 has no elements and $u = u_2$. Moreover, the strongly actuated part of the model must be expressible as

$$x_s = \begin{bmatrix} x_{sa} \\ x_{sb} \end{bmatrix} \text{ and } f_2 = \begin{bmatrix} x_{sb} \\ \alpha(x_w, x_s) + \beta(x_w, x_s)u \end{bmatrix}, \tag{5.3}$$

where $\beta(x_w, x_s)$ is square and invertible.

Letting $v = \alpha(x_w, x_s) + \beta(x_w, x_s)u$, the model can be written as

$$\begin{aligned} \dot{x}_w &= f_1(x_w, x_s, u) \\ \dot{x}_{sa} &= x_{sb} \\ \dot{x}_{sb} &= v. \end{aligned} \tag{5.4}$$

With the model expressed in this form, one may be tempted to think of "backstepping" [69] as a means of designing a stabilizing controller. Indeed, for the special case,

$$f_1(x_w, x_s, u) = f_1(x_w, x_{sa}),$$

backstepping is potentially applicable, with x_{sa} serving as a "virtual control" for the first stage of backstepping and x_{sb} a "virtual control" for the second stage. However, with f_1 also depending on x_{sb} and u, the model (5.4) does not fall into any of the known cases that are treatable by backstepping in the nonlinear control literature.

G-HZD, on the other hand, as described in Section 2.5, does apply to this class of models, using the strongly actuated x_s-states as "virtual controls" for the weakly actuated x_w-states. G-HZD gives sufficient conditions for the existence of a function $\nu(t, x_w)$, which acts as a "virtual control" for the x_s states, and an additional feedback term, $\mu(t, x_w)$, so that the feedback

$$\begin{aligned} u &= [\beta(x_w, x_s)]^{-1} \big(v - \alpha(x_w, x_s)\big) \\ v &= \mu(t, x_w) - [K_p \;\; K_d]\big(x_s - \nu(t, x_w)\big), \end{aligned} \tag{5.5}$$

with $K_p > 0$ and $K_d > 0$ appropriately sized positive definite matrices, when applied to (5.2), renders the periodic orbit, $(x_n(t), u_n(t)), t \in [0, T]$, locally exponentially stable. Moreover, the surface defined by

$$Z_t := \{(x_w, x_s) \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_s} \mid x_s = \nu(t, x_w)\},$$

for $t \in [0, T]$, is invariant under the closed-loop dynamics. See [29, Theorem 1] for the appropriate theorem.

The G-HZD design process begins with the specification of the boundary manifold, Z_0, expressed in the form

$$Z_0 := \{(x_w, x_s) \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_s} \mid x_s = \gamma(x_w)\}. \tag{5.6}$$

In the following, we develop a systematic method for its design, or equivalently, a systematic method for the design of the function $\gamma : \mathbb{R}^{n_w} \to \mathbb{R}^{n_s}$.

5.3 New Approach

Equations (5.5) and (5.6) emphasize that the boundary surface is the initial action (e.g., $t = 0$) of a time-varying feedback control policy. Because $x_s = \gamma(x_w)$ is part of a control policy, it makes sense to use the tools of optimal control to formulate its design. The challenge

is that solving optimal control problems for nonlinear systems with 40 state variables is beyond the current state of the art. On the hand, periodic linear time-varying control problems can be solved quickly for systems with forty state variables. We will formulate a standard LQR problem for the linearization of a system evaluated along a periodic orbit. From the solution of this problem, we will keep the value function and discard the feedback controller.

The value function will be used to propose the design of the surface

$$Z_0 := \{(x_w, x_s) \in \mathbb{R}^{n_w} \times \mathbb{R}^{n_s} \mid x_s = \nu(0, x_w) = \gamma(x_w) = \arg\min_{x_s} V(t_0, (x_w; x_s)\} \tag{5.7}$$

based on the following novel idea: for a given value of x_w, we will select the value (if it exists) of x_s that minimizes the cost to go from the initial condition $x_0 = (x_w; x_s)$.

5.3.1 Linearized Model and Assumptions

We'll express the linearization of the model (5.1) about a nominal periodic trajectory $x_n(t)$ and $u_n(t)$, $t \in [0, T]$, as

$$\delta\dot{x}(t) = A(t)\delta x(t) + B(t)\delta u(t), \tag{5.8}$$

where

$$\begin{aligned} \delta x(t) &= x(t) - x_n(t), \\ \delta u(t) &= u(t) - u_n(t), \\ A(t) &= \frac{\partial f(x,u)}{\partial x}|_{x=x_n(t), u=u_n(t)}, \\ B(t) &= \frac{\partial f(x,u)}{\partial u}|_{x=x_n(t), u=u_n(t)}. \end{aligned} \tag{5.9}$$

Key Assumption: We assume the origin of the linear time-varying model is exponentially stabilizable so that an associated periodically time-varying LQR problem with appropriately selected weights will always have an exponentially stabilizing solution that minimizes the cost

$$J = \int_0^\infty (\delta x(t)^\top Q(t)\delta x(t) + \delta u(t)^\top R(t)\delta u(t))dt, \tag{5.10}$$

where $R(t) \geq \alpha I$ and $Q(t) \geq \beta I$ are tunable positve-definite matrices, with $\alpha > 0$ and $\beta > 0$.

The periodically time-varying Riccati equation then yields both an optimal controller $\delta u(t) = -K(t)\delta x(t)$ and a periodically time-varying cost-to-go function

$$V(t, \delta x) := \delta x^\top P(t)\delta x,$$

with $P(t+T) = P(t)$. We will discard the controller and focus on the cost-to-go function as it will allow us to distinguish surfaces of initial conditions from which it is "easier" (i.e., lower cost) to stabilize the equilibrium of the linearized system versus those from which it is "harder" (i.e, higher cost) to stabilize the equilibrium. In turn, going back to the original nonlinear system, this will give us a concrete idea on how to select the boundary manifold Z_0 in the G-HZD method.

5.3.1.1 Designing the Boundary Manifold

Next, the cost-to-go function $V(t, \delta x)$ is used to define the surface δZ_0 at time $t_0 \in [0, T]$. Indeed, for $\delta x_w \in \mathbb{R}^{n_w}$, define

$$\delta\gamma(\delta x_w) := \underset{\delta x_s \in \mathbb{R}^{n_s}}{\arg\min}\, V(t_0, (\delta x_w; \delta x_s)). \tag{5.11}$$

Partitioning $P(t_0)$ as

$$P(t_0) =: \begin{bmatrix} P_{ww} & P_{ws} \\ P_{sw} & P_{ss} \end{bmatrix}, \tag{5.12}$$

yields

$$\delta\gamma(\delta x_w) = -P_{ss}^{-1} P_{sw} \delta x_w =: L_0 \delta_w. \tag{5.13}$$

The linearized surface δZ_0 can be used to define Z_0 in the nonlinear system in several ways. The simplest way is based on the linearization,

$$\gamma(x_w) = x_{ns}(t_0) + \delta\gamma(x_w - x_{nw}(t_0)), \tag{5.14}$$

where $x_n(t_0) =: (x_{nw}(t_0); x_{ns}(t_0))$. A more sophisticated approach is to define Z_0 as the set of states where, given x_w at time t_0, x_s is chosen to minimize the "cost-to-go" function while still respecting constraints applied to the system, such as joints limits. This alternative approach of finding a mapping between x_s and x_w to construct Z_0 allows important physical constraints to be taken into account.

5.3.1.2 Using the Boundary Manifold

The next step is running optimizations for different initial conditions on the surface Z_0, and ending on Z_0 after one time period such that the "cost-to-go" has been reduced. With many different trajectories, we'll use neural networks to regress the set of trajectories in order to derive the surface Z where we get $x_s^d(t) = L(x_w, t)$ where x_s^d is the desired strongly actuated trajectory. The added benefit is that, compared to [29], we have the means to utilize all

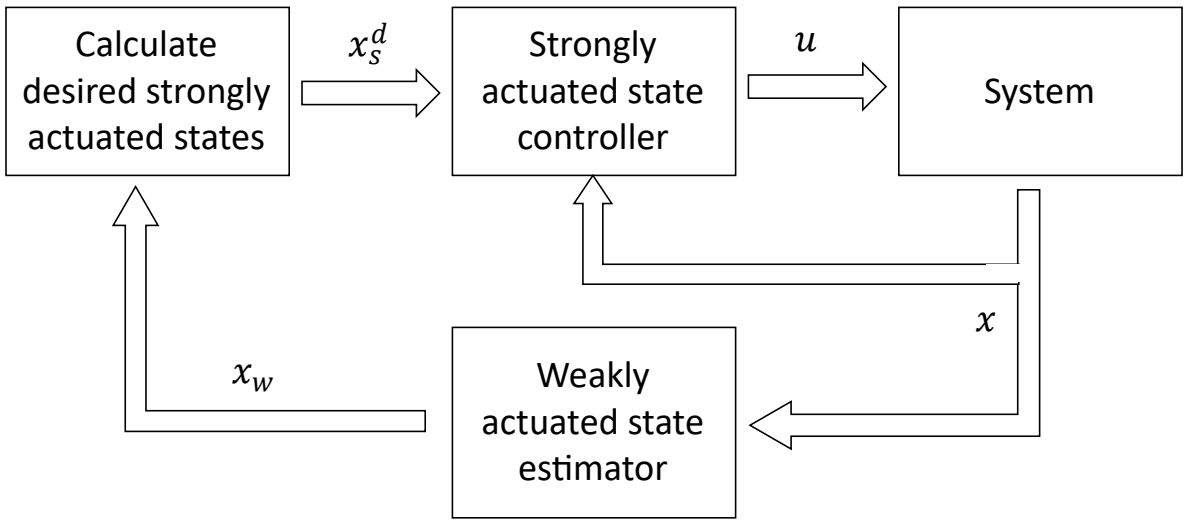


Figure 5.1: Structure of a controller under new approach.

weakly actuated states in the controller design, instead of just the pelvis velocity. In the end, the controller will resemble the structure in Figure 5.1. The inner-loop controller denoted "Strongly actuated state controller" would be a feedback linearizing controller responsible for tracking the desired strongly actuated states.

Remark 7. *One concern in [29, Remark 1-(iii)] was of trajectories where the weakly actuated states "crossed". In other words, the mapping of weakly actuated to strongly actuated states throughout the combination of trajectories wasn't injective. Here, we are hopefully less concerned about such since we're utilizing a higher order set of weakly actuated states in our mapping. In the final document, we hope to yield a formal reasoning as to why the new method would yield less "crossings" compared to the original method.*

5.4 Illustration: Linear Inverted Pendulum on a Cart

5.4.1 Derivation

We'll start with the model for the inverted pendulum on a cart, seen in Figure 5.2. The equations of motion used are obtained from [29, eq. (44)]:

$$\dot{x} = f(x,u) = \begin{bmatrix} \dot{p} \\ \frac{2\sin(\theta)\dot{\theta}^2 - 3g\cos(\theta)\sin(\theta) - 4u}{3\cos(\theta)^2 - 8} \\ \dot{\theta} \\ \frac{3\cos(\theta)\sin(\theta)\dot{\theta}^2 - 12g\sin(\theta) - 6\cos(\theta)u}{3\cos(\theta)^2 - 8} \end{bmatrix}, \tag{5.15}$$

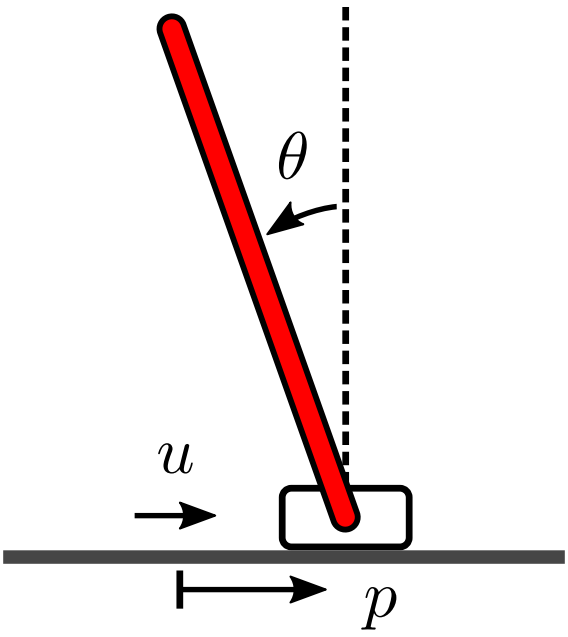


Figure 5.2: Pendulum on a cart. Figure obtained from [29, Figure 6].

where $x = \begin{bmatrix} p & \dot{p} & \theta & \dot{\theta} \end{bmatrix}^T$. We'll define the desired equilibrium to be $x_n = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$ and $u_n = 0$.

We derive the linear model to be in the form

$$\dot{x} = Ax + Bu, \tag{5.16}$$

where $A = \frac{\partial f}{\partial x}|_{x=x_n,u=u_n}$ and $B = \frac{\partial f}{\partial u}|_{x=x_n,u=u_n}$. Since the equilibrium is a point, we get a time invariant system in this example. The values of A and B are

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1.0 & 0 & 0 \\ 0 & 0 & 5.89 & 0 \\ 0 & 0 & 0 & 1.0 \\ 0 & 0 & 23.5 & 0 \end{bmatrix}, \\ B &= \begin{bmatrix} 0 \\ 0.8 \\ 0 \\ 1.2 \end{bmatrix}. \end{aligned} \tag{5.17}$$

Next, we can derive an LQR based controller to minimize the cost function from (5.10). We'll choose $Q = I_{4\times4}$ and $R = I_{1\times1}$. By solving for the standard algebraic Riccati equation, because we have a time invariant problem, we're able to obtain a feedback policy $u = -Kx$,

with

$$K = \begin{bmatrix} -1.0 & -2.67 & 49.6 & 10.9 \end{bmatrix}$$

and the Lyapunov function $V(x) = x^T P x$, where

$$P = \begin{bmatrix} 2.6673 & 3.0572 & -10.8897 & -2.8715 \\ 3.0572 & 7.0445 & -26.1745 & -6.9191 \\ -10.8897 & -26.1745 & 258.7214 & 58.7926 \\ -2.8715 & -6.9191 & 58.7926 & 13.68741 \end{bmatrix}.$$

We'll be using the Lyaponov function as a "cost-to-go". Figure 5.3 shows the results from applying the controller $u = -Kx$ with a perturbation on the weakly actuated states $x_w = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}^T$ and with $x(0) \in Z_0$.

Next is to separate the states into strongly and weakly actuated states where

$$x = \begin{bmatrix} x_w \\ x_s \end{bmatrix},\ x_w = \begin{bmatrix} p \\ \dot{p} \end{bmatrix},\ x_s = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix},$$

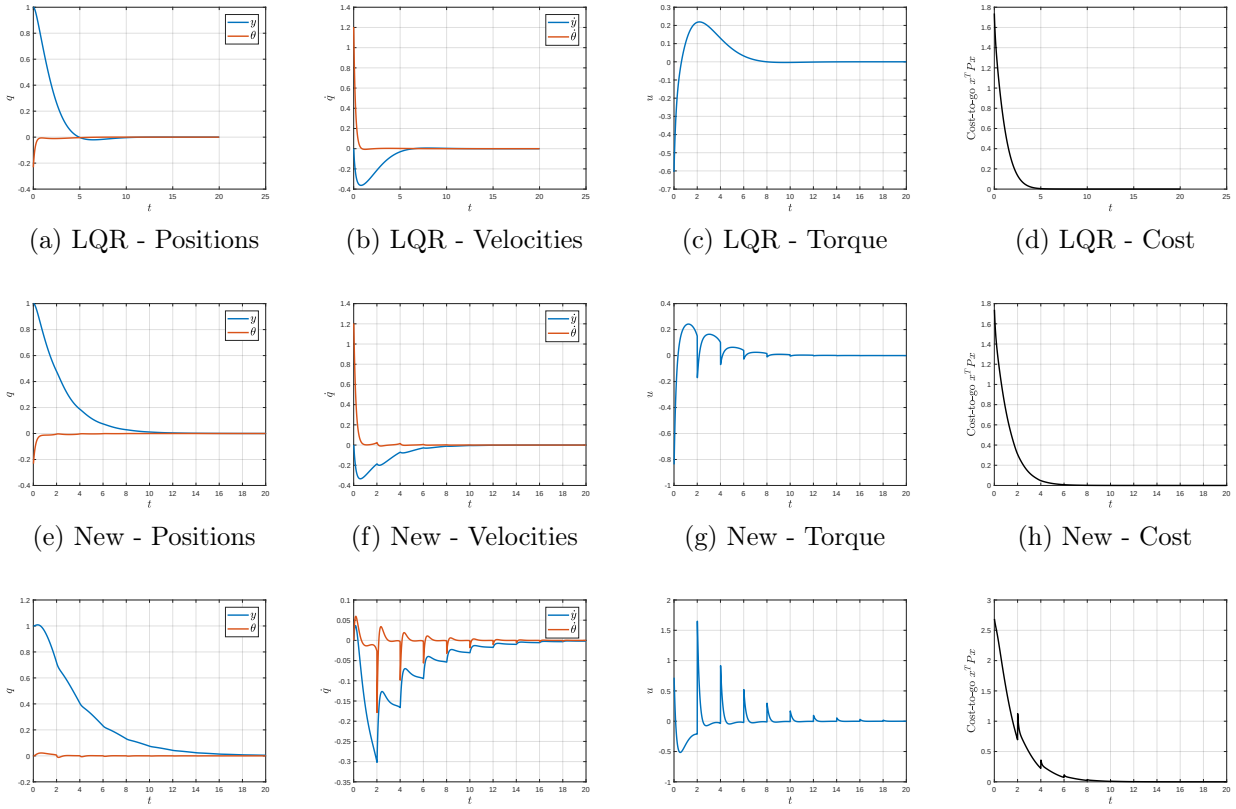
and x_w is the collection of weakly actuated coordinates and x_s is the collection of strongly actuated coordinates.

We'll now define the manifold Z_0 from (5.7). In the manifold, the relationship between the strongly and weekly actuated states are such that for given weakly actuated states, the strongly actuated states are chosen to minimize the "cost-to-go". The mapping between strongly and weakly actuated states at t_0 such that the cost function is minimized is found to be

$$\gamma(x_w) = \begin{bmatrix} -0.232 & -0.569 \\ 1.205 & 2.948 \end{bmatrix} x_w. \tag{5.18}$$

Next, we'll formulate a set of optimizations where the system must start at a point on the manifold Z_0 and must end at Z_0 as well with a chosen time period $T = 2s$ subject to the same cost function in (5.10), but with the integral spanning the period T instead of $[0, \infty)$.

With that, we find a mapping $x_s(t) = L(t, x_w(t))$ by regressing the strongly actuated states using the weakly actuated states for all time $t \in [t_0, t_0 + T]$. Then, we can define the



(a) LQR - Positions (b) LQR - Velocities (c) LQR - Torque (d) LQR - Cost

(e) New - Positions (f) New - Velocities (g) New - Torque (h) New - Cost

(i) Original - Positions (j) Original - Velocities (k) Original - Torque (l) Original - Cost

Figure 5.3: Result from testing different controllers on the linear cart and pendulum model. The "New" plots refer to the new approach using the insertion map shown in (5.18). The "Original" approach refers to the insertion map used in [29], also shown in (5.20). For all cases, the initial weakly actuated states are $x_w = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}$. The initial strongly actuated states of both the "LQR" controller and the "New" approach use (5.18) while the "Original" approach uses (5.20).

subspace

$$\begin{aligned} Z(t) = \{x \in \mathbb{R}^4 \mid x = \Pi^{-1} \begin{bmatrix} x_w \\ x_s \end{bmatrix}, \\ x_w \in \mathbb{R}^2, \\ x_s = L(t, x_w)\}, \end{aligned} \tag{5.19}$$

and $Z(t = 0) = Z(t = T) = Z_0$. Figure 5.3 shows the results with a perturbation on the weakly actuated states $x_w = \begin{bmatrix} 1.0 & 0.0 \end{bmatrix}^T$ with $x(0) \in Z_0$ using the G-HZD controller.

5.4.2 Comparison to original G-HZD approach

In previous work [29, eq. (56)], the insertion map for the surface Z_0 for the same system is

$$\gamma(x_w) = \begin{bmatrix} 0 & 0 \\ 0 & 0.5911 \end{bmatrix} x_w. \tag{5.20}$$

This was obtained by creating a library of periodic motions for different values of p_0 and $\dot{p}_0$ such that each periodic motion follows

$$p(t) = p_0 + \frac{\dot{p}_0}{\pi} \sin(\pi t).$$

For a perturbed case where the system is at the position $p = 1.0\ m$ and with a velocity of $\dot{p} = 0\ m/s$, and where we wish the system to go back to the original state of $\begin{bmatrix} p & \dot{p} \end{bmatrix}^T = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, we get different responses from each approach. In the original approach, the state at t_0 is $x_0^{old} = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$, yet in the new approach, it is $x_0^{new} = \begin{bmatrix} 1.0 & 0.0 & -0.231 & 1.204 \end{bmatrix}$. Figure 5.4 shows the different Z_0 surfaces for the same x_w. The original method limits the controller's behavior by requiring the system to be on a periodic orbit every $T = 2s$ when it is on Z_0. On the other hand, according to the "cost-to-go" function, and based on the chosen values of Q and R in (5.10), x_0^{new} is the system state that best promotes recovery of the weakly actuated states, where $\theta_0 = -0.116$ and $\dot{\theta}_0 = 0.602$. The states on Z_0 need not restrict the system to a periodic orbit. Removing this restriction allows the system to attempt recovery for all t.

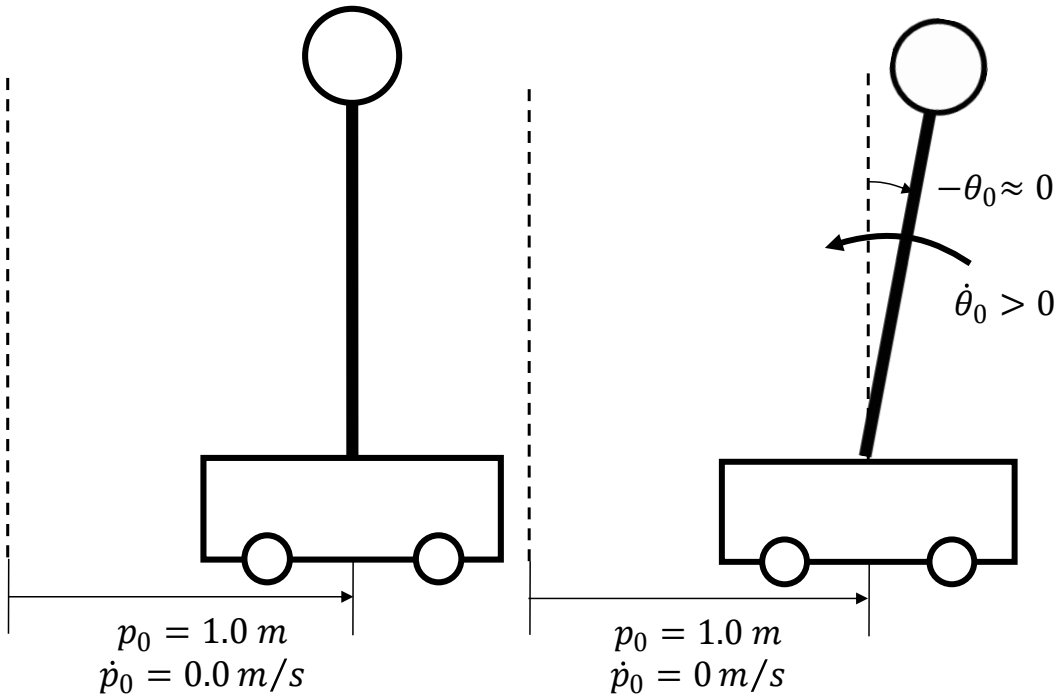
In x_0^{new}, we notice that the ideal recovery state has a negative θ with a large positive $\dot{\theta}$. This might be slightly unintuitive and perhaps undesirable. The behavior can be altered by tuning the cost function in (5.10) with

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 100 \end{bmatrix}$$

and $R = 1$, where we're penalizing heavy deviations in $\dot{\theta}$, in which case we get the behavior

$$\gamma(x_w) = \begin{bmatrix} 0.018 & 0.057 \\ 0.098 & 0.203 \end{bmatrix} x_w.$$

This results in slight tilting in the direction we want the cart to be moving, which is a more



(a) Original approach (b) New approach

Figure 5.4: Shows how the state of the system in the Z_0 manifold would look to the same perturbation in the weakly actuated states.

intuitive behavior without quick angular velocity on the pendulum. Simply put, the new method makes it easier and more intuitive to tune desired recovery behavior of the controller.

In the original approach, optimizations ran for 3 time periods, and the trajectory in the first period was used to construct Z. The motivation was to have a trajectory that eventually converges to the nominal desired state. Now with having a "cost-to-go" function, we're able to limit the optimization to one time period since we have a quantifiable metric on how close we are to the nominal trajectory.

CHAPTER 6

Cassie Implementation

In this chapter, the specifics of implementing the new G-HZD approach on Cassie will be covered. Compared to the four-state pendulum example we covered in Section 5.4, the Cassie robot poses many significant challenges. First of all, the robot's dynamic model is a hybrid system with 20 degrees of freedom (floating base model). Secondly, we're going to stabilize a periodic orbit instead of an equilibrium point. Finally, we must take into consideration numerous constraints on the system involving torque and joint limits.

6.1 Nominal Orbit and LQR Optimization

6.1.1 Nominal orbit and linearizing the model

As in Section 5.3, the controller's design starts by formulating a time-varying LQR problem for the robot's linearization along a nominal orbit. We obtained Cassie's floating based model from its URDF description in FROST. This yields a dynamic model of the form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + J(q)^{\top}F \tag{6.1}$$

$$J(q)\ddot{q} + \frac{\partial J(q)\dot{q}}{\partial q}\dot{q} = 0, \tag{6.2}$$

and a discrete impact map

$$x^{+} = \Delta(x^{-}), \quad x \in S, \tag{6.3}$$

where $x = [q, \dot{q}]^{T}$, and S is the switching surface. For Cassie, $q \in \mathbb{R}^{20}$ and has 9 holonomic constraints, 4 from four bar linkages on the legs and 5 from the stance foot being fixed to the ground and only allowed to roll.

Once the holonomic constraints are taken into account, Cassie can be parameterized by a reduced-dimensional coordinate system $q_{red} \in \mathbb{R}^{11}$ and functions $q_{red} = \Pi(q)$ and $q = \Psi(q_{red})$ that allow us to go back and forth between the reduced model and the floating based model. The reduced dynamic model can be expressed as

$$\Sigma : \begin{cases} \dot{x}_{red} = f(x_{red}) + g(x_{red})u, & x_{red} \notin S \\ x_{red}^{+} = \Delta(x_{red}^{-}), & x_{red} \in S. \end{cases} \tag{6.4}$$

Next we linearize the reduced model around the periodic walk-in-place trajectory, $x^n(t)$, yielding a time-varying model of the form

$$\Sigma : \begin{cases} \delta\dot{x}_{red}(t) = A(t)\delta x_{red}(t) + B(t)\delta u(t), & x_{red} \notin S \\ \delta x_{red}^{+} = F\delta x_{red}^{-}, & x_{red} \in S, \end{cases} \tag{6.5}$$

where $\delta x_{red}(t) = x_{red}(t) - x_{red}^{n}(t)$. For Cassie, the reduced coordinates were selected as

$$q_{red} = \begin{bmatrix} \text{stance foot roll around ground} \\ \text{10 actuated joints} \end{bmatrix} \tag{6.6}$$

The nominal trajectory $x^n(t)$ is obtained through FROST optimizations similar to what has been seen in previous sections. In this case, it is a walk-in-place trajectory for Cassie.

6.1.2 LQR

One of the important steps in this design process is to design weights for the cost function, $Q(t)$ and $R(t)$,

$$J = \int_0^\infty \left(\delta x_{red}(t)^\top Q(t)\delta x_{red}(t) + \delta u(t)^\top R(t)\delta u(t)\right) dt, \tag{6.7}$$

to reflect the behavior that we want to see in the closed-loop system. Since we aren't imposing constraints in the LQR optimization problem, we are obliged to include penalties in the cost function to shape the response. For instance, assuming we want the swing foot to have a certain foot clearance in the middle of the step, we can add a term to $Q(t)$ as follows. Let's

define the function $f(q)$ to be the foot height of the swing leg. We can add the term

$$Q_{\text{foot height}} = k((\frac{\partial \Psi(q_{red})}{\partial q_{red}})|_{q=q^n_{red}(t)})^T((\frac{\partial f(q)}{\partial q})|_{q=q^n(t)})^T \\ (\frac{\partial f(q)}{\partial q})|_{q=q^n(t)}(\frac{\partial \Psi(q_{red})}{\partial q_{red}})|_{q=q^n_{red}(t)} \quad (6.8)$$

to the matrix $Q(t)$ at time range $t = [\frac{1}{2}T - \Delta T, \frac{1}{2}T + \Delta T]$, where k is a tunable weight and Δt is a small value. This will result in a cost term that penalizes foot height deviation from the nominal trajectory at $t = \frac{1}{2}T$.

For the control designs performed in this chapter evaluated experimentally, we've added terms controlling swing foot height in the middle of the step and at the end of the step, the latter to make sure the foot impacts with the ground. In addition, we've added terms for swing foot orientation, torso orientation to remain upright, and stance knee angle to keep the leg length fairly constant.

Now one can solve the LQR problem and obtain the cost-to-go function similar to what was done in Section 5.4.

6.1.3 Forming Z_0 surface

First things first, we need to represent the coordinate system in the form $x_w \in \mathbb{R}^4$ and $x_s \in \mathbb{R}^{18}$. The choices for x_w and x_s affect the controller behavior as it determines what 'signal' to respond to. We select the weakly actuated states to be

$$x_w = \begin{bmatrix} \text{torso forward position relative to stance foot} \\ \text{torso sideways position relative to stance foot} \\ \text{torso forward velocity} \\ \text{torso sideways velocity} \end{bmatrix}. \quad (6.9)$$

The choice of the strongly actuated coordinates is a bit more obvious: they are the actuated coordinates minus the stance ankle. We're keeping the stance ankle passive mainly because the motor at the joint is very weak and we wanted a more underactuated system for this experiment. This results in $[x_w, x_s]^T \in \mathbb{R}^{22}$. We arrived at our choice of weakly actuated states after some experimentation. Originally we've tried using the stance foot and ankle as the weakly actuated states, but the resulting Z_0 surface compensated for having disturbances on those coordinates by having large velocities on the torso to counter it. The next choice was the torso coordinates. The rationale is that the velocity signals can be cleanly obtained from an EKF and that disturbances on the coordinates correspond to what disturbances we

want to recover from.

Next, we want to solve for

$$\delta\gamma(\delta x_w) := \underset{\delta x_s \in \mathbb{R}^{18}}{\arg\min}\, V(t_0, (\delta x_w; \delta x_s)) \tag{6.10}$$

where t_0 is the middle of the step. Unlike using a closed form solution as in Section 5.3.1.1, we'll use a numerical solver so that we can impose constraints on the problem. The constraints are simply joint limits, torque limits, swing foot height during midstep, and torso orientation. We run the optimizations for many different $\delta x_w \in \mathbb{R}^4$, then regress the mapping between δx_s and δx_w using third order polynomials, including all cross terms.

6.2 Frost Optimizations

The FROST optimizations are single step optimizations starting from the middle of the right stance step till the middle of the left stance step where each step takes $T = 0.4s$. The optimization starts on the Z_0 surface and also ends at the Z_0 surface. Other constraints include torso orientation being upright, foot clearance, stance knee being constant, swing foot horizontal velocity being zero before impact, and joint and torque limits. The cost function is the same as (6.7) but with a finite bounded integral.

We run 1225 optimizations for different perturbed δx_w, 7 for sagittal position, 7 for sagittal velocity, 5 for lateral position, and 5 for lateral velocity. We then regress the results of the optimization to map t and x_w onto x_s using a neural network.

Remark 8. *One issue that arose during the implementation was the time it took to evaluate the neural network. Two fixes were done to address that. First, we moved the evaluation of the neural network to a secondary computer such that both computers "talk" through an ethernet cable. Second, we increased the amount of inputs to be various expressions of the original 5 inputs (such as x_1^2 or $sin(x_2)x_1$). The rationale was that these terms could be of use during the regression and could result in decreasing the size of the network. Thus the neural network used has 18 inputs and 33 outputs, and has 6 hidden layers. The activation layers used are $tanh()$.*

6.3 Controller

The controller architecture resembles the description of the exoskeleton controller seen in Section 4.7.4. Figure 6.1 shows the controller architecture. Some notable things are that

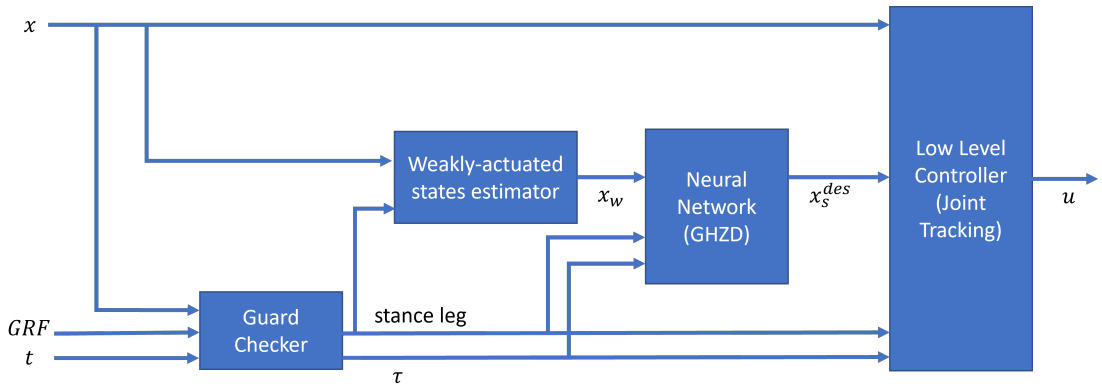


Figure 6.1: Overview of the controller structure implemented on Cassie. Guard Checker handles swapping the stances. Weakly-actuated state estimator calculates x_w. The neural network computes the desired x_s. The The Low-level Controller tracks the joints using passivity based control.

we use invariant-EKF from [52] to estimate the state of the system. The weakly-actuated state estimator internally has a low-pass filter to smooth the signal. In addition, the desired strongly actuated states are evaluated using the neural network mentioned in Section 6.2.

In the low-level controller, the joint tracking itself uses a passivity based controller from [40]. In addition, instead of tracking the stance hip joints, we're directly tracking the torso orientation. Whenever the stance foot is not firmly on the ground, which is determined by the estimated ground reaction forces, x_w^n is fed into the neural network instead of x_w.

6.4 Experiments and Results

The setup involved having a laptop connected to Cassie through an Ethernet cable. The laptop runs the invariant-EKF remotely at a rate of 2000 Hz and evaluates the neural network component at rate of 500 Hz. The on board controller on Cassie handles the low level joint tracking and leg swapping while running at a rate of 2000 Hz. The neural network and invariant-EKF were moved to a secondary computer due to computer performance limitations.

For the experiments, the robot starts with a foot placement controller while tracking the nominal trajectory based on a version of the controller seen in [41]. Then midway in the experiment, the controller switches to the G-HZD controller where foot placement is turned off and the signal x_s^{des} is used instead. Video seen at [7] shows an experiment where the robot was shoved while G-HZD is turned on and Cassie successfully rejecting the disturbances while maintaining walking in place. Figure 6.2 shows a screenshot of the video.

Figure 6.3 shows the velocity signal in the x-y plane. Around the 94 s mark is when the



Figure 6.2: Picture of Cassie being pushed during the experiment. Experiment can be viewed at [7].

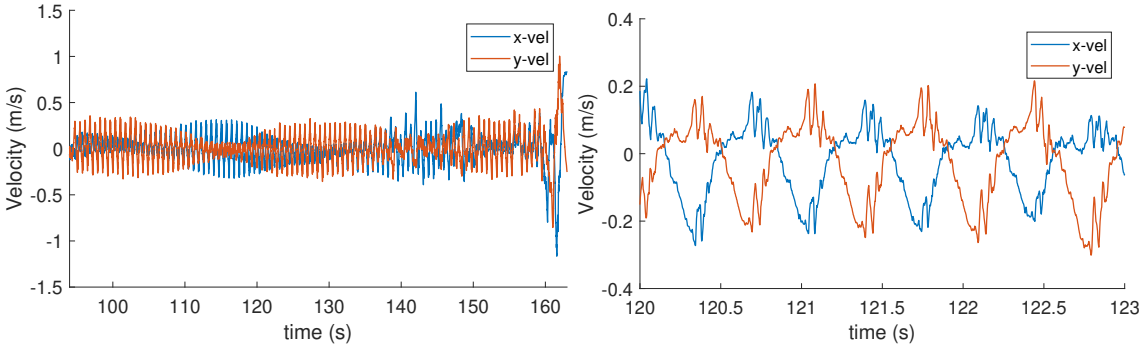


Figure 6.3: Estimated velocity of Cassie during the experiment.

G-HZD controller was enabled. Past the 160 s point is when the robot slipped and fell after the application of a large perturbation. Around the 140 s point we can see some disturbance being rejected. Figure 6.4 shows the ground reaction force. Figure 6.5 shows the input into the neural network.

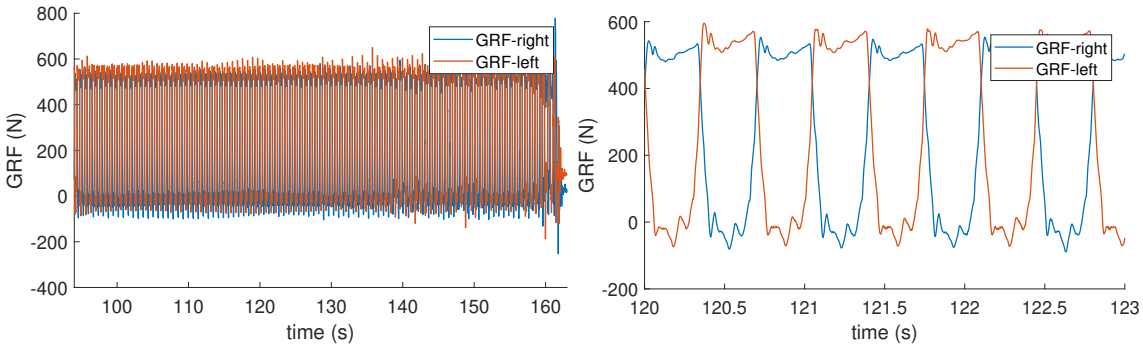


Figure 6.4: Ground reaction forces on Cassie's feet during experiment.

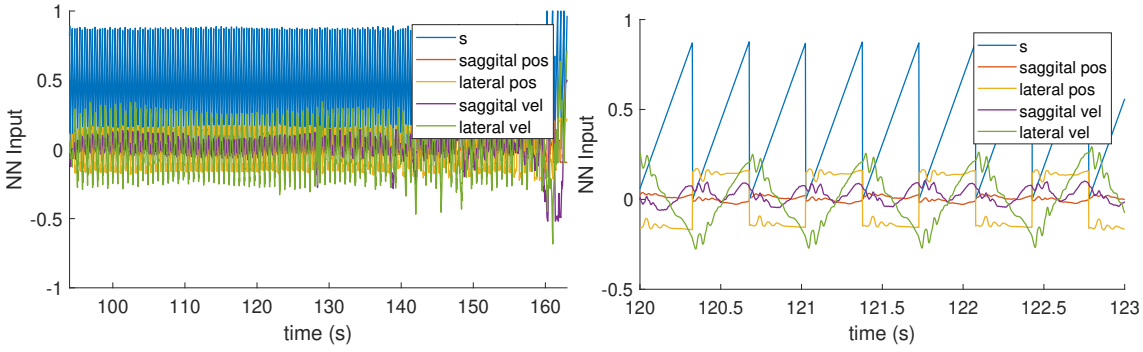


Figure 6.5: Input to the neural network during experiment.

CHAPTER 7

Discussion and Future Works

The focus of this thesis was on progress I made in the research of developing control approaches to bipeds, specifically using trajectory optimizations. We've covered C-FROST, which opened up the doors to rapidly generate 1000+ gaits in less than two hours using cloud computing. When removing the Coriolis terms from the model, the time shrinks to a few minutes [57]. The creation of C-FROST is one tool that opened up the doors to be able to implement the new G-HZD approach, as highlighted by the Cassie implementation that used 1225 trajectories.

The dissertation also covered research I've done in our lab's first venture into lower-limb exoskeletons. We've demonstrated how the gait library and original G-HZD methods could apply to systems with humans in them that are also fully actuated. In the future, applying the new G-HZD approach to a fully actuated system similar to the exoskeleton Atalante is something I would like to do.

The new G-HZD method showcased a pipeline of moving from setting up trajectory optimization problems to control design to experimental implementation while reducing the necessity of human tuning on the actual hardware. That said, the theoretical model used during optimization does not take into consideration compliances and tracking errors by the actual robot. Research is being done in other labs [62] on using machine learning to represent modeling inaccuracies from experimental data and modifying reference signals to compensate for tracking errors. Combining such research with mine would help in further eliminating human hand-tuning of the final feedback policies.

In the future, I'd like to have this work extend beyond stabilizing a single trajectory. In the spirit of combining gait library and G-HZD, we should be able to create a set library of control feedbacks that stabilize different desired trajectories, from different desired speeds to different slopes or terrains. The scope of that would involve one or two orders of magnitude more gaits generated, but with C-FROST and cloud services, that shouldn't be a problem.

APPENDIX A

How Direct Collocation Works

The direct collocation trajectory optimization utilizes the collocation methods for solving ordinary differential equations based on the finite step implicit Runge-Kutta methods [21]. Here, the specific formulation of the Hermite-Simpson algorithm, one of the most commonly used direct collocation schemes, is introduced. Consider a system of the form $\dot{x} = f(x, u)$, a trajectory optimization problem for such system can be stated as

$$\begin{aligned} J(x(t), u(t)) = \min_{u(t)} \quad & \int_0^T L(x(t), u(t))dt \\ \text{st.} \quad & x(t) = \int_0^t f(x(t), u(t))dt \\ & 0 \geq c(x(t), u(t)), \quad 0 \leq t \leq T, \end{aligned} \tag{A.1}$$

where $L(\cdot)$ represent the running cost function, and $c(\cdot)$ represents the path constraints. To solve this problem using direct collocation, a discrete representation of the continuous time solution is introduced. Specifically, the time interval $t \in [0, T]$ is divided into a fixed number of uniformly distributed intervals (see Figure A.1). In particular, the even-numbered nodes (e.g., t_0, t_2, ..., t_N) are called cardinal nodes, and the odd-numbered nodes between every two cardinal nodes are called interior nodes. At each discrete node of $t = t_i$, an approximation of state variables $x_i = x(t_i)$ and control inputs $u_i = u(t_i)$ is introduced as a set of optimization variables to be solved. In this chapter, the approximation of the slope of state variables $\dot{x}_i = \dot{x}(t_i)$ is also introduced as *defect variables* in the optimization.

The Hermite-Simpson methods then use piece-wise continuous cubic interpolation polynomials to approximate the solution of the system over each interval between two neighboring cardinal nodes. This approximation can be fully determined by the approximated state variables and slopes at the cardinal nodes. Hence, if the approximated states x_i and slopes $\dot{x}_i$ at the interior nodes match the interpolation polynomial at time $t = t_i$ (e.g., $\bar{x}_i$ and $\dot{x}_i$ in Figure A.1), then the resulting piece-wise polynomials are considered as an approximated

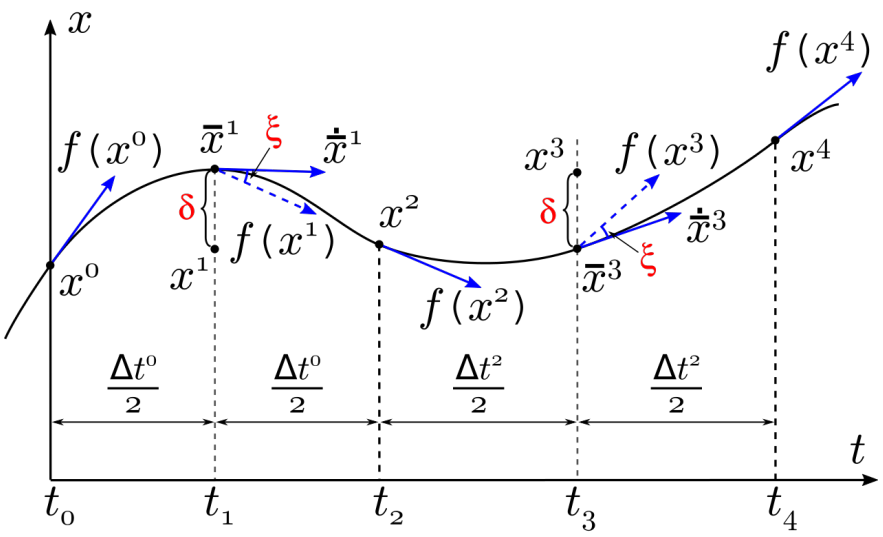


Figure A.1: Illustration of defect constraints and node distribution of the direct collocation optimization [53].

solution of the system [50]. To find this approximated solution, e.g., the discrete representation of the states, the original continuous time trajectory optimization problem can be converted to the following form given by

$$
\begin{aligned}
J(x_i, u_i) = \min_{u_i} \quad & \sum_{i=1}^{N-1} w_i L(x_i, u_i) \\
\text{st.} \quad & \dot{x}_i = f(x_i, u_i) \\
& c(x_i, u_i) \geq 0, \quad 0 \leq i \leq N \\
& \dot{x}_i - 3(x_{i+1} - x_{i-1})/2\Delta t_i + (\dot{x}_{i-1} + \dot{x}_{i+1})/4 = 0 \\
& x_i - (x_{i+1} + x_{i-1})/2 - \Delta t_i(\dot{x}_{i-1} - \dot{x}_{i+1})/8 = 0,
\end{aligned}
\tag{A.2}
$$

where for all $i \in \{1, 3, \cdots, N-1\}$, where $\Delta t_i = t_{i+1} - t_{i-1}$ is the time interval between two cardinal nodes, and w_i is weighting factor of each node determined by the Gaussian quadrature [53]. Specifically, the last two constraints are called *collocation constraints*, which are determined by cubic interpolation polynomials. The above nonlinear programming problem can be solved straightforwardly by existing numerical NLP solvers.

APPENDIX B

Partial Hybrid Zero Dynamics and a Central Controller

With the aim of driving the outputs exponentially to zero, the control law,

$$u_{IO} := \mathcal{A}^{-1}(\begin{bmatrix} 0 \\ L_f^2 y_2(q,\dot{q},\alpha,\beta) \end{bmatrix} + \begin{bmatrix} L_f y_1(q,\dot{q}) \\ k^d L_f y_2(q,\dot{q},\alpha,\beta) \end{bmatrix} + \begin{bmatrix} \bar{k}^p y_1(q,\dot{q},v_d) \\ 2k^p y_2(q,\alpha,\beta) \end{bmatrix}), \tag{B.1}$$

where $k^p, \bar{k}^p, k^d$ are diagonal matrices with positive entries and the *Decoupling Matrix* $\mathcal{A}$ is given by

$$\mathcal{A} = \begin{bmatrix} L_g y_1(q,\dot{q}) \\ L_g L_f y_2(q,\dot{q},\alpha) \end{bmatrix}, \tag{B.2}$$

input-output linearizes the system and yields the exponentially stable linear output dynamics

$$\dot{y}_1 = -\bar{k}^p y_1, \tag{B.3}$$

$$\ddot{y}_2 = -2k^p y_2 - k^d \dot{y}_2. \tag{B.4}$$

Note that $L_f y_1, L_g y_1, L_f y_2, L_f^2 y_2$, and $L_g L_f y_2$ denote various Lie derivative of y_1, y_2 with respect to the vector fields f, g.

The input-output feedback controller, u_{IO}, renders the *zero dynamics* manifold, given by

$$\mathcal{Z} := \{x \in T\mathcal{Q} \mid y_1 = 0, y_2 = 0, L_f y_2 = 0\}, \tag{B.5}$$

invariant in the continuous dynamics, i.e., any solution that starts in $\mathcal{Z}$ remains in $\mathcal{Z}$ throughout the continuous phase. However, since the impact generally involves a jump on the velocity, the post impact velocity modulating output, y_1^+, is non zero, and consequently the zero dynamics manifold, $\mathcal{Z}$, is not impact variant under the choice of these outputs. Therefore, the hybrid invariance condition is enforced only on the position modulating outputs,

resulting in the *Partial Zero Dynamics Surface* [17], defined as

$$\mathcal{PZ} := \{x \in T\mathcal{Q} \mid y_2 = 0, L_f y_2 = 0\}. \tag{B.6}$$

The Partial Zero Dynamics Surface is said to be impact invariant if the post impact states remain in $\mathcal{PZ}$, i.e.

$$\Delta(x) \in \mathcal{PZ}, \qquad \forall x \in S \cap \mathcal{PZ}. \tag{B.7}$$

The Partial Zero Dynamics Surface is hybrid invariant if it is invariant in the continuous dynamics and if it is impact invariant. If there exists a control input, u, for the system, Σ, such that $\mathcal{PZ}$ is hybrid invariant, then the system is said to have a *Partial Hybrid Zero Dynamics* (PHZD) surface. Moreover, by restricting the dynamics of the system on the PHZD surface, the analysis of periodic orbits for the full-order system can be done through those of the lower dimensional PHZD. Further details can be found in [15, 78]. For the considered model and choice of outputs, these dynamics are of dimension 4 and described by the coordinates $(\tau, y_1, \dot{\tau}, \dot{y}_1)$. In the next section, we briefly summarize a gait generation technique, presented in [17], for determining the parameters α and v_d that define the desired outputs y_d.

APPENDIX C

Constraints for exoskeleton optimization

The following constraints are enforced for the optimization of the exoskeleton in Sections 4.3-4.6.

1. *Defect Constraints*: These are constraints on the estimated states from the optimizer and states obtained from the interpolation polynomial. A complete description of defect constraints can be found in [56].

2. *System and linear output dynamics*: The system dynamics, (2.1) and (2.2), and the closed-loop linear output dynamics, (2.6), are imposed as equality constraints.

3. *PHZD Condition*: The relative degree two outputs must be zero at the beginning of each domain, i.e., the condition in (B.7) must be satisfied.

4. *Periodicity of an orbit*: Post impact states at the last node at the current domain must be equal to the first node of the next domain.

5. *Parameter consistency*: The parameters, $(\alpha^i, \beta^i, v_d^i)$ and the time T_I^i, at each time step, i, need to be consistent and constant throughout and between each domain.

6. *Domain of admissibility*: The domain of admissibility constraints, as defined in [46, eq. 31], must be satisfied to ensure holonomic and unilateral constraints are met.

7. *Guard condition*: To ensure that the system reaches the appropriate guard condition, the guard condition is imposed on the last node.

8. *Holonomic constraints*: Ensure that the holonomic constraints are held constant. However, to ensure that they are held at the correct constant, additional set of constraints are required that explicitly enforce this, i.e. the constraint, $\eta - \bar{\eta}^0 = 0$, is enforced at the first node of each domain.

9. *Time parameterization*: According to (4.5), the parameters β_1 and β_2 represent the linearized hip position in the sagittal plane, δp_{hip} at the beginning and at the end of a gait respectively. These conditions must be enforced at the first and the last node of each domain, respectively.

10. *Walking Speed, Step Length and Foot clearance*: A desired forward walking speed, step length and swing foot clearance may be provided to achieve the required walking behavior.

11. *Foot retraction*: The forward velocity of the swing foot before impact must be negative. This reduces impact loses and, when combined with the foot clearance constraint, helps avoid cases of gaits with foot scuffing.

12. *Rigid human*: The human subsystem is constrained to have torso and arm angles fixed, with the torso upright relative to the exoskeleton and the arms fixed downwards.

APPENDIX D

Linearizing a hybrid model

D.1 Preliminaries

Suppose that $y = F(x)$ and that $y_0 + \delta y = F(x_0 + \delta x)$, where $y_0 = F(x_0)$. Then doing a first-order Taylor, expansion we have

$$F(x_0 + \delta x) \approx F(x_0) + \frac{\partial F(x)}{\partial x}\Big|_{x=x_0} \delta x \tag{D.1}$$

so that

$$\delta y = \frac{\partial F(x)}{\partial x}\Big|_{x=x_0} \delta x. \tag{D.2}$$

We now apply this to several cases:

(a) Suppose $x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$ and $x_0 = \begin{bmatrix} q_0 \\ \dot{q}_0 \end{bmatrix}$. Then,

$$\delta y = \frac{\partial F(q, \dot{q})}{\partial q}\Big|_{x=x_0} \delta q + \frac{\partial F(q, \dot{q})}{\partial \dot{q}}\Big|_{x=x_0} \delta \dot{q}. \tag{D.3}$$

(b) Suppose $F(q, \dot{q}) = \begin{bmatrix} f(q) \\ \frac{\partial f(q)}{\partial q} \dot{q} \end{bmatrix}$. Then,

$$
\begin{aligned}
\delta y &= \frac{\partial}{\partial q} F(q,\dot{q})\big|_{x=x_0}\delta q + \frac{\partial}{\partial \dot{q}} F(q,\dot{q})\big|_{x=x_0}\delta\dot{q} \\
&= \frac{\partial}{\partial q}\begin{bmatrix} f(q) \\ \frac{\partial f(q)}{\partial q}\dot{q} \end{bmatrix}\Big|_{x=x_0}\delta q + \frac{\partial}{\partial \dot{q}}\begin{bmatrix} f(q) \\ \frac{\partial f(q)}{\partial q}\dot{q} \end{bmatrix}\Big|_{x=x_0}\delta\dot{q} \\
&= \frac{\partial}{\partial q}\begin{bmatrix} f(q) \\ \frac{\partial f(q)}{\partial q}\dot{q} \end{bmatrix}\Big|_{x=x_0}\delta q + \begin{bmatrix} 0 \\ \frac{\partial f(q)}{\partial q} \end{bmatrix}\Big|_{x=x_0}\delta\dot{q}.
\end{aligned}
\tag{D.4}
$$

(c) Suppose that $y = \begin{bmatrix} z \\ \dot{z} \end{bmatrix}$ and $F(q,\dot{q}) = \begin{bmatrix} f(q) \\ \frac{\partial f(q)}{\partial q}\dot{q} \end{bmatrix}$. Then,

$$
\begin{aligned}
\delta z &= \frac{\partial f(q)}{\partial q}\Big|_{x=x_0}\delta q, \\
\delta\dot{z} &= \frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\Big|_{x=x_0}\delta q + \frac{\partial f(q)}{\partial q}\Big|_{x=x_0}\delta\dot{q}.
\end{aligned}
\tag{D.5}
$$

(d) Suppose there is a third component so that $\ddot{z} = \frac{\partial f(q)}{\partial q}\ddot{q} + \left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}$, $x = \begin{bmatrix} q \\ \dot{q} \\ \ddot{q} \end{bmatrix}$, and $x_0 = \begin{bmatrix} q_0 \\ \dot{q}_0 \\ \ddot{q}_0 \end{bmatrix}$. Then,

$$
\begin{aligned}
\delta\ddot{z} &= \frac{\partial f(q)}{\partial q}\Big|_{x=x_0}\delta\ddot{q} + \frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\ddot{q}\right)\Big|_{x=x_0}\delta q + 2\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\Big|_{x=x_0}\delta\dot{q} \\
&\qquad + \frac{\partial}{\partial q}\left(\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}\right)\Big|_{x=x_0}\delta q \\
&= \frac{\partial f(q)}{\partial q}\Big|_{x=x_0}\delta\ddot{q} + 2\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\Big|_{x=x_0}\delta\dot{q} \\
&\qquad + \left[\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\ddot{q}\right) + \frac{\partial}{\partial q}\left(\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}\right)\right]\Big|_{x=x_0}\delta q.
\end{aligned}
\tag{D.6}
$$

Claim: The differential of $\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}$ with respect to $\dot{q}$ is equal to $2\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\delta\dot{q}$.

Proof: Let $\left[\cdot\right]_k$ denote the k-th row of a column vector and let $\left[\cdot\right]_{k\ell}$ be the $k-\ell$ element

of a matrix. Expanding out the partial derivatives we have that

$$\Big[\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}\Big]_k = \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 f_k(q)}{\partial q_i \partial q_j}\dot{q}_j\dot{q}_i,$$

and from basic calculus, we know that

$$\frac{\partial^2 f_k(q)}{\partial q_i \partial q_j} = \frac{\partial^2 f_k(q)}{\partial q_j \partial q_i}.$$

Hence,

$$\begin{aligned}\frac{\partial}{\partial \dot{q}_\ell}\Big[\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}\Big]_k &= \frac{\partial}{\partial \dot{q}_\ell}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 f_k(q)}{\partial q_i \partial q_j}\dot{q}_j\dot{q}_i\Big] \\ &= \sum_{j=1}^{n}\frac{\partial^2 f_k(q)}{\partial q_\ell \partial q_j}\dot{q}_j + \sum_{i=1}^{n}\frac{\partial^2 f_k(q)}{\partial q_i \partial q_\ell}\dot{q}_i \\ &= 2\sum_{j=1}^{n}\frac{\partial^2 f_k(q)}{\partial q_\ell \partial q_j}\dot{q}_j \\ &= 2\Big[\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\Big]_{k\ell}.\end{aligned} \tag{D.7}$$

Hence,

$$\frac{\partial}{\partial \dot{q}}\Big[\left(\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right)\right)\dot{q}\Big] = 2\frac{\partial}{\partial q}\left(\frac{\partial f(q)}{\partial q}\dot{q}\right).$$

D.2 Linearization of Second-order Model in Floating Base Form

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu + J(q)^\top F \tag{D.8}$$

$$J(q)\ddot{q} + \frac{\partial J(q)\dot{q}}{\partial q}\dot{q} = 0. \tag{D.9}$$

Assumptions

1. $J(q)$, the contact Jacobian, is full rank.

2. For $0 \le t \le T$, $(q_n(t), \dot{q}_n(t), \ddot{q}_n(t), u_n(t), F_n(t))$ form a solution of the model in the obvious way that when inserted into the above equations, we have left-hand side equals right-hand side; in particular, dropping the time argument,

$$\begin{aligned} D(q_n)\ddot{q}_n + C(q_n,\dot{q}_n)\dot{q}_n + G(q_n) &= Bu_n + J(q_n)^\top F_n, \\ J(q_n)\ddot{q}_n + \frac{\partial J(q)\dot{q}_n}{\partial q}\Big|_{q=q_n}\dot{q}_n &= 0 \end{aligned} \tag{D.10}$$

Let's rewrite the model as

$$\begin{bmatrix} D(q) & -J(q)^\top \\ J(q) & 0 \end{bmatrix}\begin{bmatrix} \ddot{q} \\ F \end{bmatrix} + \begin{bmatrix} C(q,\dot{q}) \\ \frac{\partial J(q)\dot{q}}{\partial q} \end{bmatrix}\dot{q} + \begin{bmatrix} G(q) \\ 0 \end{bmatrix} = \begin{bmatrix} B \\ 0 \end{bmatrix} u \tag{D.11}$$

and define

$$D_e(q) := \begin{bmatrix} D(q) & -J(q)^\top \\ J(q) & 0 \end{bmatrix}, \tag{D.12}$$

$$C_e(q,\dot{q}) := \begin{bmatrix} C(q,\dot{q}) \\ \frac{\partial J(q)\dot{q}}{\partial q} \end{bmatrix}, \tag{D.13}$$

$$G_e(q) := \begin{bmatrix} G(q) \\ 0 \end{bmatrix}, \tag{D.14}$$

$$B_e := \begin{bmatrix} B \\ 0 \end{bmatrix}. \tag{D.15}$$

In the following, we continue to drop the time argument and write

$$q := q_n + \delta q \tag{D.16}$$

$$\dot{q} := \dot{q}_n + \delta \dot{q} \tag{D.17}$$

$$\ddot{q} := \ddot{q}_n + \delta \ddot{q} \tag{D.18}$$

$$F := F_n + \delta F \tag{D.19}$$

$$u := u_n + \delta u. \tag{D.20}$$

Applying Taylor's Theorem and keeping only the zeroth- and first-order terms,

$$\begin{aligned} &D_e(q_n + \delta q) \begin{bmatrix} \ddot{q}_n + \delta \ddot{q} \\ F_n + \delta F \end{bmatrix} \approx \\ &\qquad D_e(q_n) \begin{bmatrix} \ddot{q}_n \\ F_n \end{bmatrix} + D_e(q_n) \begin{bmatrix} \delta \ddot{q} \\ \delta F \end{bmatrix} + \frac{\partial}{\partial q} \left(D_e(q) \begin{bmatrix} \ddot{q}_n \\ F_n \end{bmatrix} \right)_{|q=q_n} \delta q, \end{aligned} \tag{D.21}$$

$$\begin{aligned} &C_e(q_n + \delta q, \dot{q}_n + \delta \dot{q}) (\dot{q}_n + \delta \dot{q}) \approx \\ &\qquad C_e(q_n, \dot{q}_n)\dot{q}_n + \frac{\partial}{\partial q} \left(C_e(q, \dot{q}_n)\dot{q}_n \right)_{|q=q_n} \delta q + 2C_e(q_n, \dot{q}_n)\delta \dot{q}, \end{aligned} \tag{D.22}$$

$$\begin{aligned} &G_e(q_n + \delta q) \approx \\ &\qquad G_e(q_n) + \frac{\partial}{\partial q} G_e(q)_{|q=q_n} \delta q. \end{aligned} \tag{D.23}$$

Using the fact (D.10) holds, the zero-order terms cancel and we arrive at the linearized model

$$\begin{aligned} &D_e(q_n) \begin{bmatrix} \delta \ddot{q} \\ \delta F \end{bmatrix} + 2C_e(q_n, \dot{q}_n)\delta \dot{q} \\ &\qquad + \left[\frac{\partial}{\partial q} \left(D_e(q) \begin{bmatrix} \ddot{q}_n \\ F_n \end{bmatrix} \right) + \frac{\partial}{\partial q} \left(C_e(q, \dot{q}_n)\dot{q}_n \right) + \frac{\partial}{\partial q} G_e(q) \right]_{|q=q_n} \delta q = B_e \delta u, \end{aligned} \tag{D.24}$$

or

$$\begin{aligned} &D_e(q_n) \begin{bmatrix} \delta \ddot{q} \\ \delta F \end{bmatrix} + 2C_e(q_n, \dot{q}_n)\delta \dot{q} \\ &\qquad + \frac{\partial}{\partial q} \left(D_e(q) \begin{bmatrix} \ddot{q}_n \\ F_n \end{bmatrix} + C_e(q, \dot{q}_n)\dot{q}_n + G_e(q) \right)_{|q=q_n} \delta q = B_e \delta u. \end{aligned} \tag{D.25}$$

D.3 Linearized Impact Map

Assumptions

1. $J(q)$, the contact Jacobian, is full rank. This is for the same foot used in the ensuing dynamic model

2. $(q_n, \dot{q}_n^+, \dot{q}_n^-, F_n^I)$ form a solution of the nominal impact model in the sense that

$$\begin{bmatrix} D(q_n) & -J(q_n)^\top \\ J(q_n) & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_n^+ \\ F_n^I \end{bmatrix} = \begin{bmatrix} D(q_n)\dot{q}_n^- \\ 0 \end{bmatrix} \tag{D.26}$$

We define the deviations from the nominal solution

$$q := q_n + \delta q \tag{D.27}$$
$$\dot{q}^- := \dot{q}_n^- + \delta\dot{q}^- \tag{D.28}$$
$$\dot{q}^+ := \dot{q}_n^+ + \delta\dot{q}^+ \tag{D.29}$$
$$F^I := F_n^I + \delta F^I \tag{D.30}$$

Applying Taylor's Theorem and keeping only the zeroth- and first-order terms,

$$\begin{aligned} D_e(q_n+\delta q)\begin{bmatrix} \dot{q}_n^+ + \delta\dot{q}^+ \\ F_n^I + \delta F^I \end{bmatrix} \approx \\ D_e(q_n)\begin{bmatrix} \dot{q}_n^+ \\ F_n^I \end{bmatrix} + D_e(q_n)\begin{bmatrix} \delta\dot{q}^+ \\ \delta F^I \end{bmatrix} + \frac{\partial}{\partial q}\left(D_e(q)\begin{bmatrix} \dot{q}_n^+ \\ F_n^I \end{bmatrix}\right)_{|q=q_n} \delta q, \end{aligned} \tag{D.31}$$

$$\begin{aligned} \begin{bmatrix} D(q_n+\delta q)\left(\dot{q}_n^- + \delta\dot{q}^-\right) \\ 0 \end{bmatrix} \approx \\ \begin{bmatrix} D(q_n)\dot{q}_n^- \\ 0 \end{bmatrix} + \begin{bmatrix} D(q_n)\delta\dot{q}^- \\ 0 \end{bmatrix} + \begin{bmatrix} \frac{\partial}{\partial q}\left(D(q)\dot{q}_n^-\right)_{|q=q_n} \\ 0 \end{bmatrix} \delta q. \end{aligned} \tag{D.32}$$

From (D.26), we then arrive at the linearized model

$$D_e(q_n)\begin{bmatrix} \delta\dot{q}^+ \\ \delta F^I \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial q}\left(D(q)\dot{q}_n^-\right)_{|q=q_n} \\ 0 \end{bmatrix} \delta q - \frac{\partial}{\partial q}\left(D_e(q)\begin{bmatrix} \dot{q}_n^+ \\ F_n^I \end{bmatrix}\right)_{|q=q_n} \delta q + \begin{bmatrix} D(q_n) \\ 0 \end{bmatrix} \delta\dot{q}^-, \tag{D.33}$$

which is more conveniently expressed as

$$D_e(q_n)\begin{bmatrix}\delta\dot{q}^+\\ \delta F^I\end{bmatrix} = \frac{\partial}{\partial q}\left(\begin{bmatrix}D(q)\dot{q}_n^-\\ 0\end{bmatrix} - D_e(q)\begin{bmatrix}\dot{q}_n^+\\ F_n^I\end{bmatrix}\right)_{|q=q_n}\delta q + \begin{bmatrix}D(q_n)\\ 0\end{bmatrix}\delta\dot{q}^-. \tag{D.34}$$

D.4 Reduced-second-order Linearized Model

Let's write the generalized configuration coordinates as $q \in Q$ and the generalized configuration variables of the reduced-order model as $q_{red} \in Q_{red}$. Furthermore, let $\Phi : Q \to Q_{red}$ and $\Psi : Q_{red} \to Q$ be such that $\Phi \circ \Psi = I_d$, the identity mapping on Q_{red}. To be useful on Cassie, $\Psi(q_{red}) \in Q$ should result in the stance foot being flat on the ground.

D.4.1 Special case: Reduced coordinates are a projection

For Cassie, the floating-base configuration variables are

$$q = \begin{bmatrix}x\\ y\\ z\\ q_x\\ q_y\\ q_z\\ q_{body}\end{bmatrix}, \tag{D.35}$$

and a natural choice of the reduced coordinates is the `projection`

$$q_{red} = \begin{bmatrix}q_x\\ q_{body}\end{bmatrix}, \tag{D.36}$$

and hence we write $q_{red} = \Pi_1(q)$. Since Π is linear, we will not make the distinction between $\Pi_1(q)$ and $\Pi_1 q$, with Π_1 being a matrix. We note that

$$\delta q_{red} = \Pi_1(\delta q) \tag{D.37}$$

$$\delta \dot{q}_{red} = \Pi_1(\delta \dot{q}) \tag{D.38}$$

$$\delta \ddot{q}_{red} = \Pi_1(\delta \ddot{q}) \tag{D.39}$$

Let's define Π_2 by

$$\Pi_2 \left(\begin{bmatrix} q \\ F \end{bmatrix} \right) = q_{red}. \tag{D.40}$$

For $q \in \Psi(Q_{red})$ and $\dot{q} \in T_{q=\Psi(q_{red})}Q$, we note that

$$\begin{aligned} \delta q &= \frac{\partial \Psi(q_{red})}{\partial q_{red}}\Big|_{q_{red}=\Pi(q_n)} \delta q_{red} \\ \delta \dot{q} &= \frac{\partial \Psi(q_{red})}{\partial q_{red}}\Big|_{q_{red}=\Pi(q_n)} \delta \dot{q}_{red} + \frac{\partial}{\partial q_{red}} \left(\frac{\partial \Psi(q_{red})}{\partial q_{red}} \Pi(\dot{q}_n) \right)\Big|_{q_{red}=\Pi(q_n)} \delta q_{red} \end{aligned} \tag{D.41}$$

The reduced-second-order linearized time-varying dynamic model is now given.

$$\boxed{\delta \ddot{q}_{red} = A_{21}(\cdot)\delta q_{red} + A_{22}(\cdot)\delta \dot{q}_{red} + B_2(\cdot)\delta u,} \tag{D.42}$$

where

$$\begin{aligned} A_{21}(\cdot) &:= -\Pi_2 D_e^{-1}(q_n) \frac{\partial}{\partial q} \left(D_e(q) \begin{bmatrix} \ddot{q}_n \\ F_n \end{bmatrix} + C_e(q, \dot{q}_n)\dot{q}_n + G_e(q) \right)_{|q=q_n} \left(\frac{\partial \Psi(q_{red})}{\partial q_{red}} \right)\Big|_{q_{red}=\Pi(q_n)} + \\ &\quad - \Pi_2 D_e^{-1}(q_n) 2C_e(q_n, \dot{q}_n) \frac{\partial}{\partial q_{red}} \left(\frac{\partial \Psi(q_{red})}{\partial q_{red}} \Pi(\dot{q}_n) \right)\Big|_{q_{red}=\Pi(q_n)} \\ A_{22}(\cdot) &:= -\Pi_2 D_e^{-1}(q_n) 2C_e(q_n, \dot{q}_n) \left(\frac{\partial \Psi(q_{red})}{\partial q_{red}} \right)\Big|_{q_{red}=\Pi(q_n)} \\ B_2(\cdot) &:= \Pi_2 D_e^{-1}(q_n) B_e \end{aligned} \tag{D.43}$$

D.4.2 General case

$$q_{red} = \Phi(q) \tag{D.44}$$

$$\dot{q}_{red} = \frac{\partial \Phi(q)}{\partial q}\dot{q} \tag{D.45}$$

$$\ddot{q}_{red} = \frac{\partial \Phi(q)}{\partial q}\ddot{q} + \frac{\partial}{\partial q}\left(\frac{\partial \Phi(q)}{\partial q}\dot{q}\right)\dot{q} \tag{D.46}$$

From pages one and two of this document, the differentials are

$$\delta q_{red} = \frac{\partial \Phi(q)}{\partial q}\Big|_{q=q_n}\delta q \tag{D.47}$$

$$\delta \dot{q}_{red} = \frac{\partial}{\partial q}\left(\frac{\partial \Phi(q)}{\partial q}\dot{q}_n\right)\Big|_{q=q_n}\delta q + \frac{\partial \Phi(q)}{\partial q}\Big|_{q=q_n}\delta \dot{q} \tag{D.48}$$

and

$$\delta \ddot{q}_{red} = \frac{\partial \Phi(q)}{\partial q}\Big|_{q=q_n}\delta \ddot{q} + 2\frac{\partial}{\partial q}\left(\frac{\partial \Phi(q)}{\partial q}\dot{q}\right)\Big|_{q=q_n}\delta \dot{q} + \frac{\partial}{\partial q}\left[\frac{\partial \Phi(q)}{\partial q}\ddot{q}_n + \frac{\partial}{\partial q}\left(\frac{\partial \Phi(q)}{\partial q}\dot{q}_n\right)\dot{q}_n\right]\Big|_{q=q_n}\delta q. \tag{D.49}$$

The objective is to express the reduced-second-order linearized time-varying dynamic model as before,

$$\boxed{\delta \ddot{q}_{red} = A_{21}(\cdot)\delta q_{red} + A_{22}(\cdot)\delta \dot{q}_{red} + B_2(\cdot)\delta u.} \tag{D.50}$$

From (D.25), we have

$$\begin{aligned}\delta \ddot{q} = &- \Theta D_e^{-1}(q_n) 2C_e(q_n, \dot{q}_n)\delta \dot{q} - \Theta D_e^{-1}(q_n)\frac{\partial}{\partial q}\left(D_e(q)\begin{bmatrix}\ddot{q}_n \\ F_n\end{bmatrix} + C_e(q, \dot{q}_n)\dot{q}_n + G_e(q)\right)_{|q=q_n}\delta q \\ &+ \Theta D_e^{-1}(q_n)B_e\delta u,\end{aligned} \tag{D.51}$$

where here, $\Theta : Q \times \mathcal{F} \to Q$ projects out the "force terms". Using (D.49) and carefully

grouping terms, we obtain

$$\begin{aligned}\delta\ddot{q}_{red} = &\left\{ \frac{\partial}{\partial q}\left[\frac{\partial \Phi(q)}{\partial q}\ddot{q}_n + \frac{\partial}{\partial q}\left(\frac{\partial \Phi(q)}{\partial q}\dot{q}_n\right)\dot{q}_n\right]\right. \\ &\left. - \frac{\partial \Phi(q)}{\partial q}\Theta D_e^{-1}(q_n)\frac{\partial}{\partial q}\left(D_e(q)\begin{bmatrix}\ddot{q}_n \\ F_n\end{bmatrix} + C_e(q,\dot{q}_n)\dot{q}_n + G_e(q)\right)\right\}\Big|_{q=q_n}\delta q \\ &+ \left\{2\frac{\partial}{\partial q}\left(\frac{\partial \Phi(q)}{\partial q}\dot{q}\right)\Big|_{q=q_n} - \frac{\partial \Phi(q)}{\partial q}\Theta D_e^{-1}(q_n)2C_e(q_n,\dot{q}_n)\right\}\Big|_{q=q_n}\delta\dot{q} \\ &+ \left\{\frac{\partial \Phi(q)}{\partial q}\Theta D_e^{-1}(q_n)B_e\right\}\Big|_{q=q_n}\delta u \\ &=: \mathbf{A}(\cdot)_{21}\delta q + \mathbf{A}(\cdot)_{22}\delta\dot{q} + \mathbf{B}(\cdot)_2\delta u.\end{aligned}$$

To arrive at the final form, we must substitute in from (D.41) the terms below,

$$\begin{aligned}A(\cdot)_{21} &:= \mathbf{A}(\cdot)_{21}\frac{\partial \Psi(q_{red})}{\partial q_{red}}\Big|_{q_{red}=\Phi(q_n)} + \mathbf{A}(\cdot)_{22}\frac{\partial}{\partial q_{red}}\left(\frac{\partial \Psi(q_{red})}{\partial q_{red}}\frac{\partial \Phi(q)}{\partial q}\dot{q}_n\right)\Big|_{q_{red}=\Phi(q_n)} \\ A(\cdot)_{22} &:= \mathbf{A}(\cdot)_{22}\frac{\partial \Psi(q_{red})}{\partial q_{red}}\Big|_{q_{red}=\Phi(q_n)} \\ B(\cdot)_2 &:= \mathbf{B}(\cdot)_2.\end{aligned} \tag{D.52}$$

D.5 Reduced-order Impact Map

Since the reduced coordinates before and after impact can be different, we'll define the mappings $\Phi^+ : Q \to Q^+_{red}$, $\Psi^+ : Q^+_{red} \to Q$, $\Phi^- : Q \to Q^-_{red}$ and $\Psi^- : Q^-_{red} \to Q$ where Q^+_{red} is the coordinates after impact and Q^-_{red} is the coordinates before impact. Likewise, while $q^+ = q^- = q$, we cannot generalize it to q^-_{red} and q^+_{red}. Instead, we have

$$\frac{\partial \Psi^+(q^+_{red})}{\partial q^+_{red}}\Big|_{q^+_{red}=\Phi^+(q_n)} \delta q^+_{red} = \frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)} \delta q^-_{red} \tag{D.53}$$

We use the results from the previous sections. Into (D.34) we substitute (D.41) to obtain

$$\begin{aligned} D_e(q_n) \begin{bmatrix} \delta\dot{q}^+ \\ \delta F^I \end{bmatrix} = \frac{\partial}{\partial q}\left(\begin{bmatrix} D(q)\dot{q}^-_n \\ 0 \end{bmatrix} - D_e(q) \begin{bmatrix} \dot{q}^+_n \\ F^I_n \end{bmatrix} \right)_{|q=q_n} \frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)} \delta q^-_{red} + \\ + \begin{bmatrix} D(q_n) \\ 0 \end{bmatrix} \left(\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)} \delta\dot{q}^-_{red} + \frac{\partial}{\partial q^-_{red}} \left(\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}} \dot{q}^-_{red,n} \right)\Big|_{q^-_{red}=\Phi^-(q_n)} \delta q^-_{red} \right), \end{aligned} \tag{D.54}$$

where

$$q^-_{red,n} = \Phi^-(q_n) \tag{D.55}$$

$$\dot{q}^-_{red,n} = \frac{\partial \Phi^-(q)}{\partial q}\Big|_{q=q_n} \dot{q}_n. \tag{D.56}$$

After regrouping of terms,

$$\begin{aligned} D_e(q_n) \begin{bmatrix} \delta\dot{q}^+ \\ \delta F^I \end{bmatrix} = \frac{\partial}{\partial q}\left(\begin{bmatrix} D(q)\dot{q}^-_n \\ 0 \end{bmatrix} - D_e(q) \begin{bmatrix} \dot{q}^+_n \\ F^I_n \end{bmatrix} \right)_{|q=q_n} \\ \times \left(\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}} + \frac{\partial}{\partial q^-_{red}} \left(\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}} \dot{q}^-_{red,n}) \right) \right)\Big|_{q^-_{red}=\Phi^-(q_n)} \delta q^-_{red} \\ + \begin{bmatrix} D(q_n) \\ 0 \end{bmatrix} \frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)} \delta\dot{q}^-_{red}, \end{aligned} \tag{D.57}$$

where $\times$ is matrix multiplication. Or,

$$\begin{aligned}\delta q &= \frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)} \delta q^-_{red},\\ \delta \dot{q}^+ &= \bar{A}\delta q^-_{red} + \bar{B}\delta \dot{q}^-_{red},\end{aligned} \tag{D.58}$$

where,

$$\begin{aligned}\bar{A} &= \Theta D_e^{-1}(q_n)\frac{\partial}{\partial q}\left(\begin{bmatrix} D(q)\dot{q}^-_n \\ 0 \end{bmatrix} - D_e(q)\begin{bmatrix} \dot{q}^+_n \\ F^I_n \end{bmatrix}\right)_{|q=q_n}\\ &\quad\times\left(\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}} + \frac{\partial}{\partial q^-_{red}}\left(\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\dot{q}^-_{red,n})\right)\right)\Big|_{q^-_{red}=\Phi^-(q_n)},\\ \bar{B} &= \Theta D_e^{-1}(q_n)\begin{bmatrix} D(q_n) \\ 0 \end{bmatrix}\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)},\end{aligned} \tag{D.59}$$

and $\Theta : Q \times \mathcal{F} \to Q$ projects out the "force terms".

Next, we need to project down the post-impact states using (D.48). From (D.48), we can state

$$\begin{aligned}\delta q^+_{red} &= \frac{\partial \Phi^+(q)}{\partial q}\Big|_{q=q_n}\delta q\\ \delta \dot{q}^+_{red} &= \frac{\partial}{\partial q}\left(\frac{\partial \Phi^+(q)}{\partial q}\dot{q}_n\right)\Big|_{q=q_n}\delta q + \frac{\partial \Phi^+(q)}{\partial q}\Big|_{q=q_n}\delta \dot{q}^+\end{aligned} \tag{D.60}$$

Combining (D.60) with (D.58), we can write the impact map as

$$\begin{aligned}\delta q^+_{red} &= \frac{\partial \Phi^+(q)}{\partial q}\Big|_{q=q_n}\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)}\delta q^-_{red},\\ \delta \dot{q}^+_{red} &= A\delta q^-_{red} + B\delta \dot{q}^-_{red},\end{aligned} \tag{D.61}$$

where,

$$\begin{aligned}A &= \frac{\partial}{\partial q}\left(\frac{\partial \Phi^+(q)}{\partial q}\dot{q}_n\right)\Big|_{q=q_n}\frac{\partial \Psi^-(q^-_{red})}{\partial q^-_{red}}\Big|_{q^-_{red}=\Phi^-(q_n)} + \frac{\partial \Phi^+(q)}{\partial q}\Big|_{q=q_n}\bar{A}\\ B &= \frac{\partial \Phi^+(q)}{\partial q}\Big|_{q=q_n}\bar{B}.\end{aligned} \tag{D.62}$$

BIBLIOGRAPHY

[1] Video to Illustrate CFROST on the Bipedal Robot Cassie Blue. https://youtu.be/pweV7NLr6JY, . Accessed: 2019-02-27.

[2] Cassie Blue. https://www.youtube.com/watch?v=oyKCBc9CZUA&list=PLFe0SMV3hBCBPg0dDG9FbqOy_x7H9kEDJ, . Accessed: 2018-07-08.

[3] Ekso Bionics. http://eksobionics.com/. Accessed: 2017-11-04.

[4] MARLO vs The Hill (22 degree slope). https://www.youtube.com/watch?v=pTx2pyzxzTM. Accessed: 2018-07-08.

[5] ReWalk 6.0 - Home. http://rewalk.com/. Accessed: 2017-11-04.

[6] Biped Robotics Lab at Michigan. https://www.biped.solutions/. Accessed: 2021-07-27.

[7] New G-HZD on Cassie. https://www.youtube.com/watch?v=IxgB1LXmQBg, . Accessed: 2021-09-16.

[8] Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. https://www.youtube.com/watch?v=UhXly-5tEkc, .

[9] Video of the gazebo simulation results. https://youtu.be/7Ol-U1y_S8I.

[10] *2010 ADA Standards for Accessible Design*, September 2010.

[11] Wandercraft. http://www.wandercraft.eu/en/, Nov 2017. Accessed: 2017-11-07.

[12] HAL for medical use (Lower Limb Type). https://www.cyberdyne.jp/english/products/LowerLimb_medical.html, Nov 2017. Accessed: 2017-11-06.

[13] Ayush Agrawal, Omar Harib, Ayonga Hereid, Sylvain Finet, Matthieu Masselin, Laurent Praly, Aaron D. Ames, Koushil Sreenath, and Jessy W. Grizzle. First steps towards translating HZD control of bipedal robots to decentralized control of exoskeletons. *IEEE Access*, 5:9919–9934, 2017. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2690407.

[14] G. Aguirre-Ollinger, J.E. Colgate, M.A. Peshkin, and A. Goswami. Active-impedance control of a lower-limb assistive exoskeleton. In *IEEE International Conference on Rehabilitation Robotics*, 2007.

[15] A. D. Ames, E. A. Cousineau, and M. J. Powell. Dynamically stable bipedal robotic walking with NAO via human-inspired hybrid zero dynamics. In *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*, pages 135–144. ACM, 2012.

[16] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control (TAC)*, 59(4):876–891, April 2014. ISSN 0018-9286. doi: 10.1109/TAC.2014.2299335.

[17] Aaron D Ames. Human-inspired control of bipedal walking robots. *IEEE Transactions on Automatic Control*, 59(5):1115–1130, 2014.

[18] Khairul Anam and Adel Ali Al-Jumaily. Active exoskeleton control systems: State of the art. *Journal of Procedia Engineering*, 41:988–994, 2012.

[19] R. Angold, N.H. Harding, and H. Kazerooni. Semi-powered lower extremity exoskeleton, 11 2011. URL `https://www.google.com/patents/US8057410`. US Patent 8,057,410.

[20] Victor M Becerra. Solving complex optimal control problems at no cost with PSOPT. In *Computer-Aided Control System Design (CACSD), 2010 IEEE International Symposium on*, pages 1391–1396. IEEE, 2010.

[21] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[22] B. G. Buss, A. Ramezani, K. Akbari Hamed, K. S. Griffin, B. A., Galloway, and J. W. Grizzle. Preliminary walking experiments with underactuated 3D bipedal robot marlo. In *Intelligent Robots and Systems (IROS 2014)*, pages 2529–2536. IEEE, 2014. URL `http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=6942907`.

[23] Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016. URL `https://hal.archives-ouvertes.fr/hal-01203507`.

[24] Xuechao Chen, Zhangguo Yu, Weimin Zhang, Yu Zheng, Qiang Huang, and Aiguo Ming. Bioinspired control of walking with toe-off, heel-strike, and disturbance rejection for a biped robot. *IEEE Transactions on Industrial Electronics*, 64(10):7962–7971, 2017. doi: 10.1109/TIE.2017.2698361.

[25] Yuxiao Chen, Ayonga Hereid, Huei Peng, and Jessy Grizzle. Enhancing the performance of a safe controller via supervised learning for truck lateral control. *Journal of Dynamic Systems, Measurement, and Control*, 141(10):101005, 2019.

[26] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. Canudas-De-Wit, and J. W. Grizzle. Rabbit: a testbed for advanced control theory. *IEEE Control Systems*, 23(5):57–79, Oct 2003. ISSN 1066-033X. doi: 10.1109/MCS.2003.1234651.

[27] C. Chevallereau, J.W. Grizzle, and C-L. Shih. Asymptotically stable walking of a five-link underactuated 3D bipedal robot. *IEEE Transactions on Robotics*, 25(1):37–50, February 2009.

[28] John J Craig. *Introduction to Robotics: Mechanics and Control*. Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 3 edition, 2005.

[29] 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*, July 2019. doi: 10.1177/0278364919859425.

[30] Xingye Da, Omar Harib, Ross Hartley, Brent Griffin, and Jessy W Grizzle. From 2D design of underactuated bipedal gaits to 3D implementation: Walking with Speed Tracking. *IEEE Access*, 4:3469–3478, 2016.

[31] Xingye Da, Ross Hartley, and Jessy W. Grizzle. Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3476–3483, 2017. doi: 10.1109/ICRA.2017.7989397.

[32] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *2014 14th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 295–302. IEEE, 2014.

[33] Aaron M. Dollar and Hugh Herr. Lower extremity exoskeletons and active orthoses: Challenges and state-of-the-art. *IEEE Transactions on Robotics*, 24(1):144–158, 2008.

[34] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization-based full body control for the DARPA Robotics Challenge. *Journal of Field Robotics*, 32(2):293–312, 2015.

[35] A. Frisoli, E. Sotgiu, C. Procopio, M. Bergamasco, B. Rossi, and C. Chisari. Design and implementation of a training strategy in chronic stroke with an arm robotic exoskeleton. In *IEEE International Conference on Rehabilitation Robotics*, 2011.

[36] Kevin Galloway, Koushil Sreenath, Aaron D. Ames, and Jessy W. Grizzle. Torque saturation in bipedal robotic walking through control lyapunov function based quadratic programs. *IEEE Access*, 3:323–332, April 2015.

[37] Paula Richley Geigle and Marni Kallins. Exoskeleton-assisted walking for people with spinal cord injury. *Arhives of Physical Medicine and Rehabilitation*, 98(7):1493—-1495, 2017.

[38] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005. doi: 10.1137/S0036144504446096. URL `http://dx.doi.org/10.1137/S0036144504446096`.

[39] M. Gomes, G. Silveira, and A. Siqueira. Gait pattern adaptation for an active lower-limb orthosis based on neural networks. *Advanced Robotics*, 25(15):1903–1925, 2011.

[40] Yukai Gong and Jessy Grizzle. Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller, 2021.

[41] 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 *American Control Conference (ACC)*, July 2019.

[42] R. D. Gregg, T. Lenzi, L. J. Hargrove, and J. W. Sensinger. Virtual constraint control of a powered prosthetic leg: From simulation to experiments with transfemoral amputees. *IEEE Transactions on Robotics*, 30(6):1455–1471, October 2014.

[43] B. Griffin and J. Grizzle. Walking gait optimization for accommodation of unknown terrain height variations. In *2015 American Control Conference (ACC)*, pages 4810–4817, July 2015. doi: 10.1109/ACC.2015.7172087.

[44] Brent Griffin and Jessy Grizzle. Nonholonomic virtual constraints and gait optimization for robust walking control. *The International Journal of Robotics Research*, 36 (8):895–922, 2017.

[45] J. W. Grizzle, J. Hurst, B. Morris, H. Park, and K. Sreenath. Mabel, a new robotic bipedal walker and runner. In *2009 American Control Conference*, pages 2030–2036, 2009.

[46] Jessy W. Grizzle, Christine Chevallereau, Ryan W. Sinnet, and Aaron D. Ames. Models, feedback control, and open problems of 3d bipedal robotic walking. *Automatica*, 50(8):1955 – 1988, 2014. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2014.04.021. URL `http://www.sciencedirect.com/science/article/pii/S0005109814001654`.

[47] J Grundmann and A Seireg. Computer control of multi-task exoskeleton for paraplegics. *Proceedings of the Second CISM/IFTOMM International Symposium on the Theory and Practice of Robots and Manipulators*, pages 233–240, 1977.

[48] Thomas Gurriet, Sylvain Finet, Guilhem Boeris, Ayonga Hereid, Omar Harib, Matthieu Masselin, Jessy Grizzle, and Aaron D. Ames. Towards restoring locomotion for paraplegics: Realizing dynamically stable walking on exoskeletons. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2018.

[49] Kaveh Akbari Hamed, Wen-Long Ma, and Aaron D. Ames. Dynamically stable 3d quadrupedal walking with multi-domain hybrid system models and virtual constraint controllers. In *American Control Conference (ACC)*, July 2019.

[50] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10 (4):338–342, 1987. doi: doi:10.2514/3.20223. URL `http://arc.aiaa.org/doi/abs/10.2514/3.20223`.

[51] 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, Dec 2018. ISSN 1066-033X. doi: 10.1109/MCS.2018.2866604.

[52] Ross Hartley, Maani Ghaffari Jadidi, Jessy W Grizzle, and Ryan M Eustice. Contact-aided invariant extended kalman filtering for legged robot state estimation. *arXiv preprint arXiv:1805.10410*, 2018.

[53] A. Hereid, C. M. Hubicki, E. A. Cousineau, and A. D. Ames. Dynamic humanoid locomotion: A scalable formulation for hzd gait optimization. *IEEE Transactions on Robotics*, 34(2):370–387, April 2018. ISSN 1552-3098. doi: 10.1109/TRO.2017.2783371.

[54] Ayonga Hereid. *Dynamic Humanoid Locomotion: Hybrid Zero Dynamics based Gait Optimization via Direct Collocation Methods*. PhD thesis, Georgia Institute of Technology, 2016.

[55] Ayonga Hereid and Aaron D. Ames. Frost: Fast robot optimization and simulation toolkit. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, Sep 2017. IEEE/RSJ.

[56] Ayonga Hereid, Eric A. Cousineau, Christian M. Hubicki, and Aaron D. Ames. 3D dynamic walking with underactuated humanoid robots: A direct collocation framework for optimizing hybrid zero dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[57] Ayonga Hereid, Omar Harib, Ross Hartley, Yukai Gong, and Jessy W. Grizzle. Rapid bipedal gait design using c-frost with illustration on a cassie-series robot. *arXiv preprint arXiv:1807.06614v2*, 2018.

[58] Ayonga Hereid, Omar Harib, Ross Hartley, Yukai Gong, and Jessy 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. doi: 10.1109/IROS40897.2019.8968056.

[59] Alexander Herzog, Nicholas Rotella, Stefan Schaal, and Ludovic Righetti. Trajectory generation for multi-contact momentum control. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 874–880. IEEE, 2015.

[60] D Hristic, M Vukobratovic, and M Timotijevic. New model of autonomous' active suit'for distrophic patients. In *Proceedings of the International Symposium on External Control of Human Extremities*, pages 33–42, 1981.

[61] Weiguang Huo, Samer Mohammed, Juan C. Moreno, and Yacine Amirat. Lower Limb Wearable Robots for Assistance and Rehabilitation: A State of the Art. *IEEE Systems Journal*, 10(3):1068–1081, 2016. ISSN 19379234. doi: 10.1109/JSYST.2014.2351491.

[62] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019.

[63] S. Jezernik, G. Colombo, T. Keller, H. Frueh, and M. Morari. Robotic orthosis lokomat: a rehabilitation and research tool. *Neuromodulation: Technology at the Neural Interface*, 6(2):108–115, 2003.

[64] R. Jimenez-Fabian and O. Verlinden. Review of control algorithms for robotic ankle systems in lower-limb orthoses, prostheses, and exoskeletons. *Med. Eng. Phys.*, 34(4): 397–408, 2012.

[65] S Kajita, F Kanehiro, K Kaneko, K Yokoi, and H Hirukawa. The 3D linear inverted pendulum model: a simple modeling for a biped walking pattern generation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 239–246, 2001. doi: 10.1109/IROS.2001.973365.

[66] Shuuji Kajita, Tomio Yamaura, and Akira Kobayashi. Dynamic walking control of a biped robot along a potential energy conserving orbit. *IEEE Transactions on Robotics and Automation*, 8(4):431–438, 1992.

[67] Duško Katić and Miomir Vukobratović. Survey of intelligent control techniques for humanoid robots. *Journal of Intelligent and Robotic Systems*, 37(2):117–141, 2003.

[68] Taeyeon Kim, Mingoo Jeong, and Kyoungchul Kong. Bio-inspired knee joint of a lower-limb exoskeleton for misalignment reduction. *IEEE/ASME Transactions on Mechatronics*, pages 1–1, 2021. doi: 10.1109/TMECH.2021.3099815.

[69] P. V. Kokotovic. The joy of feedback: nonlinear and adaptive. *IEEE Control Systems Magazine*, 12(3):7–17, 1992.

[70] S. Kolathaya, W. Guffey, R. W. Sinnet, and A. D. Ames. Direct collocation for dynamic behaviors with nonprehensile contacts: Application to flipping burgers. *IEEE Robotics and Automation Letters*, 3(4):3677–3684, Oct 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2854910.

[71] Twan Koolen, Jesper Smith, Gray Thomas, Sylvain Bertrand, John Carff, Nathan Mertins, Douglas Stephen, Peter Abeles, Johannes Englsberger, Stephen Mccrory, et al. Summary of team ihmc's virtual robotics challenge entry. In *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, pages 307–314. IEEE, 2013.

[72] H Igo Krebs, Neville Hogan, Mindy L Aisen, and Bruce T Volpe. Robot-aided neurorehabilitation. *IEEE transactions on rehabilitation engineering*, 6(1):75–87, 1998.

[73] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2589–2594. IEEE, 2014.

[74] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot. *Auton. Robots*, 40(3):429–455, March 2016. ISSN 0929-5593. doi: 10.1007/s10514-015-9479-3. URL http://dx.doi.org/10.1007/s10514-015-9479-3.

[75] Tad McGeer. Passive dynamic walking. *International Journal of Robotics Research*, 9 (2):62–82, 1990.

[76] W. Miller, A. Deathe, M. Speechley, and J. Koval. The influence of falling, fear of falling, and balance confidence on prosthetic mobility and social activity among individuals with a lower extremity amputation. *Arch. Phys. Med. Rehab.*, 82(9):1238–1244, 2001.

[77] Katja Mombaur and Karsten Berns. *Modeling, simulation and optimization of bipedal walking*. Springer, 2013.

[78] B. Morris and J. W. Grizzle. Hybrid invariant manifolds in systems with impulse effects with application to periodic locomotion in bipedal robots. *IEEE Transactions on Automatic Control*, 54(8):1751–1764, Aug 2009. ISSN 0018-9286. doi: 10.1109/TAC.2009.2024563.

[79] B. Morris, E.R. Westervelt, C. Chevallereau, G. Buche, and J.W. Grizzle. *Achieving Bipedal Running with RABBIT: Six Steps Toward Infinity*, pages 277–297. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-36119-0. doi: 10.1007/978-3-540-36119-0_13. URL https://doi.org/10.1007/978-3-540-36119-0_13.

[80] Ralph S Mosher. Handyman to hardiman. Technical report, SAE Technical Paper, 1967.

[81] R. M. Murray, Z. Li, S. S. Sastry, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[82] T. Nef, M. Guidali, and R. Riener. Armin iii–arm therapy exoskeleton with an ergonomic shoulder actuation. *Applied Bionics and Biomechanics*, 6(2):127–142, 2009.

[83] Quan Nguyen and Koushil Sreenath. Optimal robust control for bipedal robots through control lyapunov function based quadratic programs. In *Robotics: Science and Systems*, Rome, Italy, July 2015.

[84] Michael Patterson and Anil Rao. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software (TOMS)*, 41(1):1, 2014.

[85] J. P. Pell, P. T. Donnan, F. G. Fowkes, and C. V. Ruckley. Quality of life following lower limb amputation for peripheral arterial disease. *Eur. J. Vasc. Surg.*, 7(4):448–451, 1993.

[86] J. Pratt, T. Koolen, T. de Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid. *The International Journal of Robotics Research*, 31(10):1117–1133, August 2012. ISSN 0278-3649. doi: 10.1177/0278364912452762. URL `http://ijr.sagepub.com/cgi/doi/10.1177/0278364912452762`.

[87] Jerry E Pratt and Russ Tedrake. Velocity-based stability margins for fast bipedal walking. In *Fast Motions in Biomechanics and Robotics*, pages 299–324. Springer, 2006.

[88] D. Quintero, D. J. Villarreal, D. J. Lambert, S. Kapp, and R. D. Gregg. Continuous-phase control of a powered knee-ankle prosthesis: Amputee experiments across speeds and inclines. *IEEE Transactions on Robotics*, pages 1–16, 2018. ISSN 1552-3098. doi: 10.1109/TRO.2018.2794536.

[89] M. Raibert, S. Tzafestas, and C. Tzafestas. Comparative simulation study of three control techniques applied to a biped robot. In *Proceedings of IEEE Systems Man and Cybernetics Conference - SMC*, volume 1, pages 494–502 vol.1, 1993. doi: 10.1109/ICSMC.1993.384792.

[90] Marc H Raibert et al. *Legged robots that balance*, volume 3. MIT press Cambridge, MA, 1986.

[91] A. Ramezani, J. W. Hurst, K. A. Hamed, and J. W. Grizzle. Performance analysis and feedback control of ATRIAS, A three-dimensional bipedal robot. *Journal of Dynamic Systems Measurement and Control*, 136(2), Mar 2014. ISSN 0022-0434. doi: 10.1115/1.4025693.

[92] Jacob Reher, Eric A. Cousineau, Ayonga Hereid, Christian M. Hubicki, and Aaron D. Ames. Realizing dynamic and efficient bipedal locomotion on the humanoid robot DURUS. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[93] Siavash Rezazadeh, Christian Hubicki, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Andy Abate, and Jonathan Hurst. Spring-mass walking with atrias in 3d: Robust gait control spanning zero to 4.3 kph on a heavily underactuated bipedal robot. In *ASME 2015 Dynamic Systems and Control Conference*, pages V001T04A003–V001T04A003. American Society of Mechanical Engineers, 2015.

[94] E. Rocon, J.M. Belda-Lois, A.F. Ruiz, M. Manto, J.C. Moreno, and J.L. Pons. Design and validation of a rehabilitation robotic exoskeleton for tremor assessment and suppression. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 15(3):367–378, 2007.

[95] Juergen Rummel, Yvonne Blum, and Andre Seyfarth. Robust and efficient walking with spring-like legs. *Bioinspiration & biomimetics*, 5(4):046004, 2010.

[96] A. Simon, N. Fey, S. Finucane, R. Lipschutz, and L. Hargrove. Strategies to reduce the configuration time for a powered knee and ankle prosthesis across multiple ambulation modes. In *IEEE Int. Conf. Rehab. Robot.*, Seattle, WA, 2013.

[97] Ann M Simon, Kimberly A Ingraham, Nicholas P Fey, Suzanne B Finucane, Robert D Lipschutz, Aaron J Young, and Levi J Hargrove. Configuring a powered knee and ankle prosthesis for transfemoral amputees within five specific ambulation modes. *PloS one*, 9(6):e99387, 2014.

[98] Ryan W Sinnet and Aaron D Ames. Bio-inspired feedback control of three-dimensional humanlike bipedal robots. *Journal of Robotics and Mechatronics*, 24(4):595, 2012.

[99] M. W. Spong. Partial feedback linearization of underactuated mechanical systems. In *Intelligent Robots and Systems '94. 'Advanced Robotic Systems and the Real World', IROS '94. Proceedings of the IEEE/RSJ/GI International Conference on*, volume 1, pages 314–321 vol.1, Sep 1994. doi: 10.1109/IROS.1994.407375.

[100] K. Sreenath, H. Park, I. Poulakakis, and J. W. Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on MABEL. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011.

[101] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and J. W. Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on MABEL. *International Journal of Robotics Research*, 32(3): 324–345, 2013. doi: 10.1177/0278364912473344. URL `https://doi.org/10.1177/0278364912473344`.

[102] E. Stickland. Good-bye, wheelchair. *IEEE Spectrum*, 49(1):30–32, January 2012.

[103] Frank Sup, Amit Bohara, and Michael Goldfarb. Design and control of a powered transfemoral prosthesis. *The International journal of robotics research*, 27(2):263–273, 2008.

[104] Kenta Suzuki, Gouji Mito, Hiroaki Kawamoto, Yasuhisa Hasegawa, and Yoshiyuki Sankai. Intention-Based Walking Support for Paraplegia Patients with Robot Suit HAL. *Advanced Robotics*, 21(12):1441–1469, 2007. ISSN 0169-1864. doi: 10.1163/156855307781746061. URL `http://www.tandfonline.com/doi/abs/10.1163/156855307781746061`.

[105] O. Tange. Gnu parallel - the command-line power tool. *The USENIX Magazine*, 36 (1):42–47, Feb 2011. URL `http://www.gnu.org/s/parallel`.

[106] Russ Tedrake and the Drake Development Team. Drake: A planning, control, and analysis toolbox for nonlinear dynamical systems, 2016. URL `http://drake.mit.edu`.

[107] Russ Tedrake, Maurice F Fallon, Sisir Karumanchi, Scott Kuindersma, Matthew E Antone, Toby Schneider, Thomas M Howard, Matthew R Walter, Hongkai Dai, Robin Deits, et al. A summary of team mit's approach to the virtual robotics challenge. In *ICRA*, page 2087, 2014.

[108] A. Tsukahara, Y. Hasegawa, and Y. Sankai. Gait support for complete spinal cord injury patient by synchronized leg-swing with hal. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[109] Michael R. Tucker, Jeremy Olivier, Anna Pagel, Hannes Bleuler, Mohamed Bouri, Olivier Lambercy, José del R. Millán, Robert Riener, Heike Vallery, and Roger Gassert. Control strategies for active lower extremity prosthetics and orthotics: a review. *Journal of NeuroEngineering and Rehabilitation*, 12(1):1, Jan 2015. ISSN 1743-0003. doi: 10.1186/1743-0003-12-1. URL `https://doi.org/10.1186/1743-0003-12-1`.

[110] O. Unluhisarcikli, M. Pietrusinski, B. Weinberg, P. Bonato, and C. Mavroidis. Design and control of a robotic lower extremity exoskeleton for gait rehabilitation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[111] Oskar von Stryk. User's guide for DIRCOL version 2.1. *Simulation and Systems Optimization Group, Technische Universität Darmstadt, www. sim. informatik. tu-darmstadt. de/sw/dircol*, 1999.

[112] M Vukobratovic, D Hristic, and Z Stojiljkovic. Development of active anthropomorphic exoskeletons. *Medical and Biological Engineering*, 12(1):66–80, 1974.

[113] Miomir Vukobratović and Branislav Borovac. Zero-moment point—thirty five years of its life. *International journal of humanoid robotics*, 1(01):157–173, 2004.

[114] Miomir Vukobratovic, Branislav Borovac, Dusan Surla, and Dragan Stokic. *Biped locomotion: dynamics, stability, control and application*, volume 7. Springer Science & Business Media, 2012.

[115] Andreas Wächter and T. Lorenz Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2005. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y. URL `http://dx.doi.org/10.1007/s10107-004-0559-y`.

[116]] R. L. Waters, J. Perry, D. Antonelli, and H. Hislop. Energy cost of walking of amputees: The influence of level of amputation. *J. Bone Joint Surg. Amer.*, 58(1): 42–46, January 1976.

[117] E. R. Westervelt and J. W. Grizzle. Design of asymptotically stable walking for a 5-link planar biped walker via optimization. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 3, pages 3117–3122, 2002. doi: 10.1109/ROBOT.2002.1013706.

[118] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, 2003.

[119] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC press Boca Raton, 2007.

[120] E.T. Wolbrecht, D.J. Reinkensmeyer, and J.E. Bobrow. Pneumatic control of robots for rehabilitation. *The International Journal of Robotics Research*, 29(1):23–38, 2010.

[121] Tao Yang, Eric R Westervelt, James P Schmiedeler, and Ryan A Bockbrader. Design and control of a planar bipedal robot ernie with parallel knee compliance. *Autonomous robots*, 25(4):317–330, 2008.

[122] Aaron J Young and Daniel P Ferris. State of the Art and Future Directions for Lower Limb Robotic Exoskeletons. *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, 25(2):171–182, 2017. ISSN 1558-0210 (Electronic). doi: 10.1109/TNSRE.2016.2521160.

[123] P. Zaytsev, S. J. Hasaneini, and A. Ruina. Two steps is enough: No need to plan far ahead for walking balance. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6295–6300, May 2015. doi: 10.1109/ICRA.2015.7140083.

[124] Gabi Zeilig, Harold Weingarden, Manuel Zwecker, Israel Dudkiewicz, Ayala Bloch, and Alberto Esquenazi. Safety and tolerance of the ReWalk ™ exoskeleton suit for ambulation by people with complete spinal cord injury: A pilot study. *The Journal of Spinal Cord Medicine*, 35(2):96–101, 2012. ISSN 1079-0268. doi: 10.1179/2045772312Y.0000000003. URL `http://www.tandfonline.com/doi/full/10.1179/2045772312Y.0000000003`.

[125] H. H. Zhao, W. L. Ma, A. D. Ames, and M. B. Zeagler. Human-inspired multi-contact locomotion with amber2. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*, pages 199–210, April 2014. doi: 10.1109/ICCPS.2014.6843723.

[126] Huihua Zhao, Jonathan Horn, Jacob Reher, Victor Paredes, and Aaron D. Ames. First steps toward translating robotic walking to prostheses: a nonlinear optimization based control approach. *Auton. Robots*, pages 1–18, 2016.

[127] Jinying Zhu, Qining Wang, Yan Huang, and Long Wang. Adding compliant joints and segmented foot to bio-inspired below-knee exoskeleton. In *2011 IEEE International Conference on Robotics and Automation*, pages 605–610, 2011. doi: 10.1109/ICRA.2011.5980054.

[128] K. Ziegler-Graham, E. J. MacKenzie, P. L. Ephraim, T. G. Travison, and R. Brookmeyer. Estimating the prevalence of limb loss in the united states: 2005 to 2050. *Arch. Phys. Med. Rehab.*, 89(3):422–429, 2008.