Thesis

Optimal control of compliant bipedal gaits and their implementation on robot hardware

William C. Martin

CMU-RI-TR-19-25

May, 2019

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Legged animals exhibit diverse locomotion patterns known as gaits, which are capable of robustly traversing terrains of variable grade, roughness, and compliance. Despite the success of legs in nature, wheeled solutions still dominate the field of robotics. State-of-the-art humanoid robots have not yet demonstrated locomotion behaviors that are as robust or varied as their biological counterparts. While humans use a wide range of dynamic motions to ensure stable locomotion, most bipedal robots either reject external disturbances without changing gait or fall. Modern model-based control schemes often target individual gaits rather than realizing multiple behaviors within a single framework. Recently, researchers have proposed using the spring mass model as a compliant locomotion paradigm to create unified controllers for walking and running on bipedal systems. Although initial studies have revealed policies capable of transitioning through different gaits, most of these control laws are designed empirically using heuristics. Furthermore, these controllers have not yet been demonstrated on physical hardware, and thus their utility for real-world machines remains unclear.

This thesis investigates the optimal control of simplified bipedal point-mass models for designing unified walking and running control policies. We examine the theoretically maximum performance these models can achieve and evaluate their utility for controlling higher-order robot hardware. We hypothesize that it is not necessary to hand design these policies using heuristics. Instead, existing numerical optimization tools can generate approximate globally optimal policies, which can be used to find parametric control laws. We then attempt to transfer these low-dimensional plans onto a physical bipedal robot using a model-based controller to embed the underlying simplified model. We show that this control methodology leads to stable locomotion across several different gaits on the ATRIAS biped robot.

Acknowledgments

I would like to thank my advisor Hartmut Geyer and colleague Albert Wu for their help and support with this work. Their dedication to solving complex legged control problems has been a strong motivation for me during this thesis. I would also like to thank my committee members, Chris Atkeson, Stelian Coros, and Jan Peters for their helpful comments and affable conversations. Special thanks to Jonathan Hurst and his entire team at Oregon State for designing an incredible legged system. Working with ATRIAS for several years has given me a special appreciation for robot hardware. I have also had the pleasure of working and interacting with many friends and colleagues throughout graduate school; thank you all. Finally, thank you Akshara, Martha, Mom, and Dad. I cannot emphasize enough how impossible all of this would have been without you.

Contents

Chapter 1

Introduction

1.1 Barriers to progress

Legged animals are capable of executing an extremely diverse range of dynamic locomotion behaviors across a myriad of different environments. For example, many bipedal birds, such as the Indian runner duck (Fig. 1.1), use their legs to walk, run, jump, and swim over both land and sea. Despite the flexibility of legs in nature, wheeled solutions are much more widespread among motorized mobility platforms today, owing to their remarkable efficiency across continuous flat surfaces and high reliability. Indeed, a human consumes less energy per unit distance riding a bicycle than walking or running down a road. It is reasonable then to question the utility of mechanical legged platforms and perhaps even wonder why wheeled animals aren't prevalent. We can assuage these concerns by considering three aspects of locomotion environments which can favor legs over wheels. First, the compliance of a ground surface has a direct impact on the rolling resistance of a wheel. Soft compliant surfaces are inefficient for rolling objects while hard terrains, such as road pavement, are highly efficient. Second, the maximum

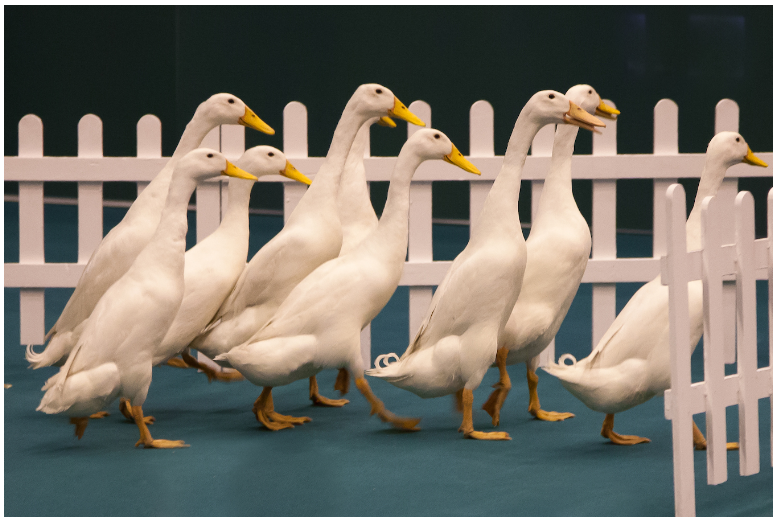


Figure 1.1: Indian Runner ducks display various motion behaviors while locomoting, including walking, running, jumping, and swimming. They are adept at moving through unstructured terrains often found on farms. Photograph available at [99] under a Creative Commons license [35].

vertical height which can be traversed by a rigid wheel is limited to half of its radius. This makes it exceptionally difficult for small wheeled platforms to climb stairs or street curbs. Third, the minimum turning radius of a vehicle is determined by the geometry of its wheel base. Typical wheel layouts prevent exceptionally tight turns that are often required in cluttered environments, such as a dense forest or messy hallway. Legs offer a suitable mobility solution when these three terrain features prove limiting and a small platform footprint is desirable. This is often the case in human environments which cannot always be prearranged for robots and must be traversed with arbitrarily arranged obstacles.

Given the increased diversity of feasible terrains for legged platforms, it may come as some surprise that legs are not commonly found on modern commercial robots. Nature has managed to produce robust legged mobility solutions, but human engineering is still catching up. Several factors make building robotic legged platforms exceedingly difficult. Sophisticated inertial sensors offer highly accurate measurements, but far fewer signals than biological proprioception systems. State-of-the-art electromagnetic and hydraulic actuators have only recently begun to provide high enough power-to-weight ratios for executing dynamic movements on human-size robots. The control-side of the problem isn't any easier; legs introduce significantly higher complexity than wheels. Rather than just one rotating degree of freedom, the average articulated leg possesses several. Legged motions are further complicated by the repeated foot-ground impacts necessary to produce forward motion. One of the most complex features, however, is the capability of legged systems to exhibit distinct dynamic motions using a single physical system. The gait of a legged system can be altered by changing the cyclical pattern of certain system variables, often the sequence of limb phases or the relationship between potential and kinetic energy. This introduces a large number of different behaviors which can be switched between as desired. Several hypotheses have been studied in order to explain why biological systems choose to transition between gaits. Commonly investigated factors include optimization of energy usage [4, 21, 54, 71, 94], decreases in muscle exertion [56, 91], storage of mechanical energy [33, 65], and injury prevention [18, 46]. However, it is most likely that no single aspect alone triggers gait transitions but rather a combination of several determinants [105]. The ability to perform different gaits is an essential component of using legs to locomote across a wide range of speeds and environments.

The success of highly varied gaits on biological systems has not carried over into the world of bipedal robots. State-of-the-art humanoid robots have been unable to match the same levels of gait robustness and diversity as demonstrated by humans and animals. While humans are capable of various highly dynamic recovery maneuvers, most legged robots must either immediately reject an external disturbance or fall. Furthermore, modern model-based control schemes typically target individual walking or running patterns, which makes switching between them difficult and often reliant on heuristics. In order to match the performance of biological systems, bipedal robots must be capable of naturally targeting numerous arbitrary motions within a single controller. The spring mass model (SMM), commonly referred to as the spring loaded inverted pendulum (SLIP), offers a potential framework for studying both walking and running using a single simplified model. This compliant locomotion paradigm has been used abundantly over the last 30 years to design robust running and walking controllers. Studies have revealed that the SMM can exhibit an exceedingly wide range of gaits akin to biological systems. Thus, researchers have recently proposed unified controllers for the SMM that can generate walking, running, and gait transition behaviors.

Gait transition policies that have been proposed for the SMM rely on heuristics and empirical design to simplify the problem. As a result, it is unclear to what extent these current controllers maximize robustness or meet other optimality criteria. Additionally, these gait transition policies have not progressed beyond simulation and have not yet been evaluated on physical hardware. The utility of the SMM for transitioning between gaits on real-world machines is currently uncertain.

1.2 Summary of Goals and Approach

The goals for this thesis fit broadly into two categories: examine the **theoretical limitations** of the bipedal spring mass model and evaluate the **practical utility** of this model for controlling multiple gaits on a physical bipedal robot. At a high level, we aim to understand the maximum locomotion robustness attainable by the bipedal spring mass model and how well this can be transferred to a real-world robot. This boils down to answering the following questions.

1. What is the continuous-time optimal control for a bipedal point-mass model when the system has few constraints?
2. How does the performance of this continuous-time control compare to existing heuristic parametric policies?
3. Do these optimal simplified model controllers transfer usefully to real-world bipedal robot hardware?

By answering these questions we expect to improve the current understanding of gait transitions within the bipedal spring mass model framework and the extent of their practical utility.

1.2.1 Computing an Optimal Control Law

Approaches to solving a continuous-time optimal control typically fall into two categories: Bellman's equation of optimality or Pontryagin's maximum principle. At a mathematical level, Bellman-style approaches involve solving a nonlinear first-order hyperbolic partial differential equation referred to as the Hamilton-Jacobi-Bellman (HJB) equation. Pontryagin-style approaches, on the other hand, aim to find a specific characteristic curve of the related HJB equation. Although both routes pose the same optimization problem, this work focuses on the Bellman approach to compute control laws rather than time-based trajectories. Ultimately, this means our goal is to solve a rather difficult partial differential equation.

In a perfect world our aim would be to find an analytic solution corresponding to the globally optimal control for our specified problem. The desire for continuous-time analytical representations often suggests using Pontryagin's principle, which can solve "academic-strength" optimal control problems [96]. However, nearly all problems of meaningful complexity omit an analytical solution, often due to non-integrable dynamics. Barring new breakthroughs in mathematics, we are resigned to accept numerical techniques for approximating the solution. While in a strictly mathematical context these techniques result in a suboptimal control, we can adjust the degree of approximation and error bounds on the solution. Unfortunately, this limits the complexity of problems which we can satisfactorily solve without introducing large approximation errors. As we increase the dimensionality of a system's state and action space, we must use sparser function approximators in order to keep the problem computationally tractable. This is because the size of these spaces, and therefore required computation time, grows exponentially with problem dimensionality (Fig. 1.2). In this work, we aim to solve a low-dimensional bounded approximation of the globally optimal control using a semi-Lagrangian dynamic programming approach with vast amounts of data.

1.2.2 Comparing Performance Metrics

Our analysis of controller performance aims to evaluate both the limits of the underlying simplified model and how well it will operate on a robot platform. Ideally we wish to measure robustness of the bipedal spring mass model at both theoretical and practical levels. We aim to use metrics that are easily quantified and readily applicable to any control policy.

The control policies in this work are presented as global feedback con-

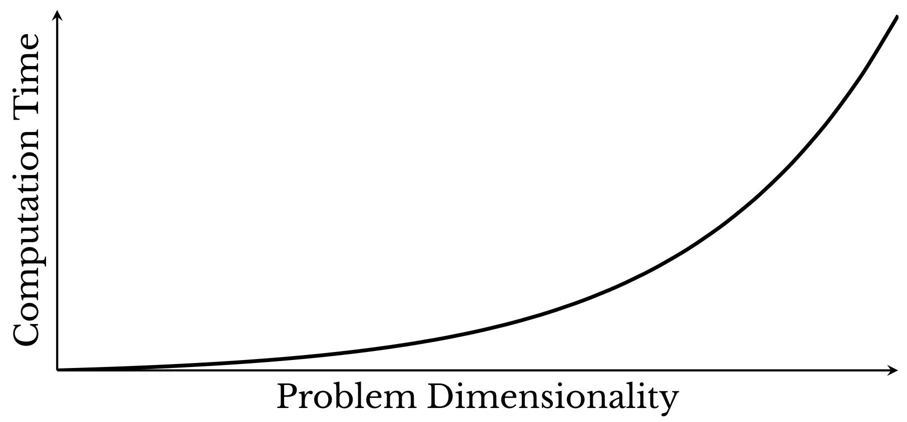


Figure 1.2: In general, the computation time required to solve an optimal control using numerical methods increases exponentially with the model dimensionality. This is due to the increased size of state space which must now be considered. This issue can be mitigated by increasing the coarseness of function approximation, but at the cost of increased solution error.

trollers, which map any given state of the system to a control action. For this reason, the first measure of theoretical robustness we will use is the size of the set of states which can be stabilized to a unique target state. This is often referred to as the backward reachable set, as it includes all states which can reach a target state using admissible controls, or the **basin of attraction** when considering a dynamical system. This metric is easy to measure for arbitrary policies and is a natural outcome of solving minimum-time "target problems". If the backward reachable set happens to include all of state-space, then the control can theoretically reject any state disturbance. However, practically speaking these sets and control laws must be solved over a bounded area of state space. In the context of gait transitions, a natural extension of this measurement is the overlap between basins of attraction for two different gait controllers. For example, we can quantify how much of state-space is capable of spontaneously transitioning from a walking to a running controller (Fig. 1.3). The larger the intersection between stabilizable walking states and stabilizable running states, the more robust we can

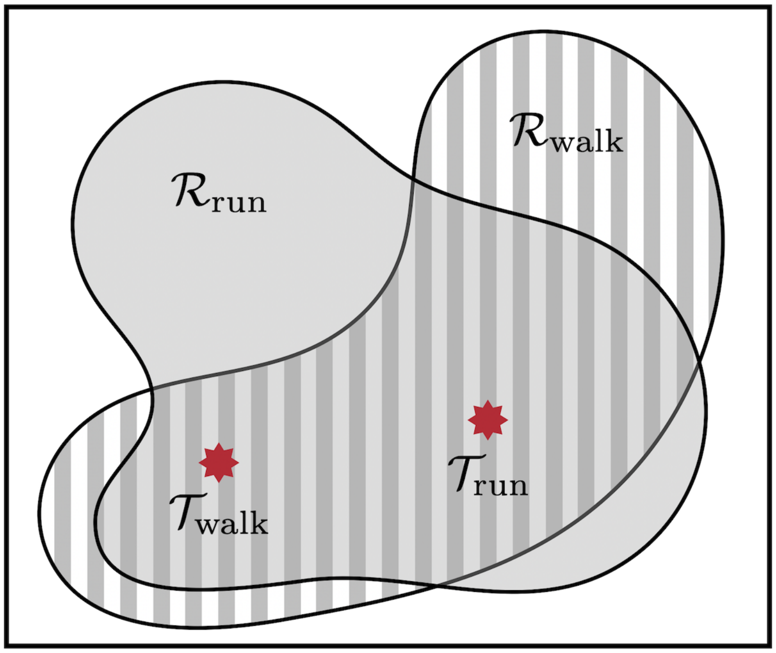

$\mathcal{R}_{\text{walk}}$
$\mathcal{R}_{\text{run}}$
$\mathcal{T}_{\text{run}}$
$\mathcal{T}_{\text{walk}}$

Figure 1.3: The overlap between multiple reachable sets, or basins of attraction, represents a useful robustness measure for gait transition policies. Here is a hypothetical example in state space illustrating a pair of reachable sets for walking, $\mathcal{R}_{\S}$, and running, $\mathcal{R}_{\S\backslash}$. Each policy has a distinct target state, $\mathcal{T}_{\S}$ or $\mathcal{T}_{\S\backslash}$. The larger the overlapping region, the most robust we can consider the gait transition policy.

consider the policy.

Evaluating the practical performance of a theoretical robot controller requires us to consider disturbances that are important to robotics in general. These break down into three primary categories: **sensory input, internal models, and control loop delay**. Sensor measurement noise is a common occurrence on any real system and has a direct impact on the state estimation necessary for feedback control. For a simulated state-space model, this disturbance can be represented as errors in the input state used for control. Internal model uncertainty has a significant effect on model-based robot control because it is often required for both state-estimation and controlling system dynamics predictably. This can be easily recreated on a simulated system by altering a model's inertial parameters. The final area of control loop delay is endemic to digital control systems which run on nearly all modern robots. A combination of loop frequency, digital delay, and plant dynamics have a direct impact on the available actuation bandwidth. Without this disturbance, a model's control inputs can be changed arbitrarily fast. This can be simulated by either introducing intentional time-delays into the control loop or passing the control through a low-pass filter. Each of these three disturbances is important for understanding the practical utility of a theoretical controller for a physical robot.

1.2.3 Transferring to Real-world Bipedal Robot Hardware

The final objective of this thesis is to transfer the computed theoretical optimal controls onto a physical bipedal robot. The hardware platform used for this implementation will be CMU's ATRIAS (Fig. 1.4), a human-size bipedal robot developed by the Dynamic Robotics Laboratory at Oregon State University [58]. In order to translate simplified model behavior onto the machine, we use a model-based control mapping approach that embeds the low-dimensional model on a higher order system. This approach was originally developed for spring mass model running on ATRIAS [129], but is expanded here to include gaits with double support phases as well. Although we keep the approach general enough to be used on other biped robots with light legs, several specific decisions will be motivated by the ATRIAS hardware. To this end, this thesis contains a mixture of ideal control design decisions along with practical considerations for real systems.

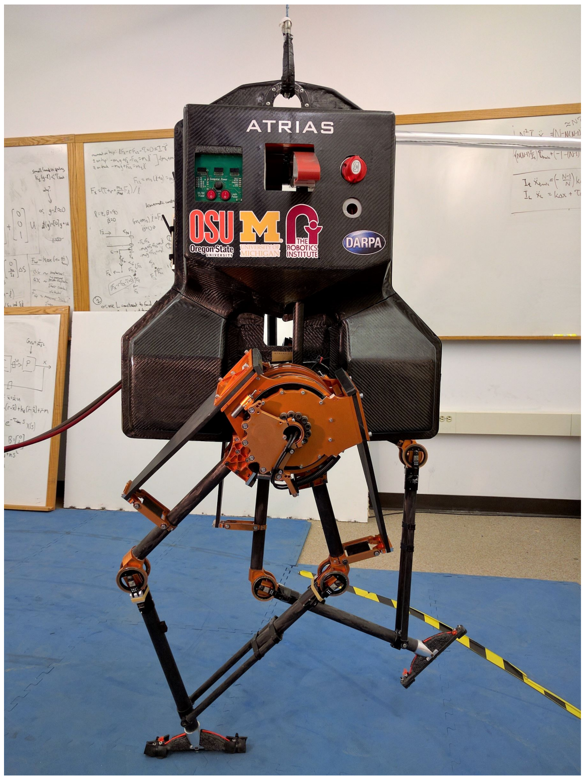


Figure 1.4: We use the ATRIAS biped as a target hardware platform for evaluating our control algorithms. This robot roughly approximates the dynamics of a simplified model due to its light legs and concentrated mass.

1.3 Contributions

The goal of this thesis is twofold. First, we aim to find a theoretically optimal gait transition policy for the bipedal spring mass model using mathematical optimization. Second, we aim to evaluate how this policy translates to real-world legged systems. This process will address the following scientific question. What are the limits of theoretical robustness and practical utility of the bipedal spring mass model for walking and running? To answer this inquiry, we consider the following two hypotheses.

Part 1: Optimal Control of Simplified Models

Hypothesis: It is not necessary to hand design low-dimensional locomotion controllers using heuristics. Existing numerical optimization tools can generate approximate globally optimal policies, which can be used to find parametric control laws.

Major Components:

1. Optimal control of a bipedal point-mass model with continuous time, state, and action using dynamic programming methods to numerically generate approximate globally optimal policies.
2. Design of a low-dimensional Poincare controller able to freely switch between flight, single support, and double support phases for both walking and running gaits without explicit user input.
3. A performance comparison of different controllers generated using dynamic programming, including both discretely and continuously applied control.

Part 2: Implementation on Robot Hardware

Hypothesis: Optimal control laws for the bipedal spring mass model can be transferred to robot hardware using a model-based controller to embed the underlying simplified model. This will lead to stable walking, running, and gait transitions between for the ATRIAS biped robot.

Major Components:

1. Implementation and experimental evaluation of spring mass model deadbeat running on the ATRIAS biped using a model-based controller.
2. Extension of the model-based controller to follow general center-of-

mass plans for multiple gaits including walking, running, and transitions between.

3. Implementation and experimental evaluation of spring mass model walking and running gaits on the ATRIAS using optimal control laws from dynamic programming.

Chapter 2

Background

2.1 The Spring Mass Model

Over the last 30 years the spring mass model has evolved from a simple predictor of human running mechanics to a unified control framework for a variety of bipedal gaits. Although there exist numerous feasible parameterizations for walking and running, the spring mass model is important because it offers a single low-dimensional descriptor of essential degrees-of-freedom for legged locomotion. Furthermore, control policies have been identified within this framework that are both superstable and capable of blindly traversing frequent and large ground height changes. These features have two important implications. First, the spring mass model is simple enough that it can serve as a locomotion template for high degree-of-freedom robot morphologies without limiting extensibility for other high-level goals. Second, the framework lends itself well to mathematical optimization approaches that scale poorly with the size of state or action space. In this section, we will discuss the history of research on the spring mass model in order to describe the current state-of-the-art and highlight future research that remains to be done.

2.1.1 Early Biomechanical Studies

Compliant elements have a long history of providing simple models for complex mechanical phenomena, such as atomic forces and fluid mechanics. Experimental evidence that the center of mass motion during human running could also be described using elastic mechanisms began to emerge in the 1960s and 1970s. Cavagna and colleagues measured the mechanical work performed during human walking and running for more than a decade using force plates and cameras [28, 30, 32]. These studies observed that the variations in potential and kinetic energy of the center of mass were substantially in phase during running, similar to those of a bouncing rubber ball. This conclusion led to further hypotheses about the mechanisms of elastic energy storage in the human body [31]. Concurrently, these results were extended to include bipedal and quadrupedal animals [33], cementing elastic energy storage as a fundamental concept for legged running mechanics.

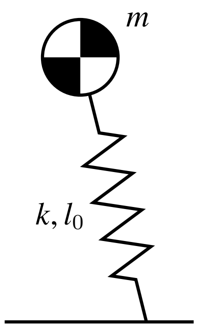
Early attempts to produce a mathematical model of biomechanical running focused on using spring laws to represent vertical ground reaction forces during stance. Alexander and colleagues began this work in the 1970s by modeling the running motions of kangaroos [2] and humans [1, 5] with complex cosine equations which could correspond to springs. Around the same time, McMahon and Greene presented a spring-damper model to describe vertical leg behavior during running [78]. McMahon simplified this model several years later by using a single spring mass system to represent the vertical center of mass motions during mammalian running [76, 79]. This "mass-spring" model was shown to be a good predictor of vertical dynamics during both bipedal and quadrupedal gaits. However, none of these early spring-based models attempted to describe horizontal running dynamics.

It wasn't until the late 1980s that spring mass models were presented as predictors of both horizontal and vertical locomotion dynamics. Although not widely known at the time, van Gurp and colleagues were the first to publish such a spring mass model for describing the hindlimb motion of a walking horse [124]. This was shortly followed by two independent studies that presented the modern day planar spring mass model as a predictor of human running dynamics. Blickhan [19] investigated which model parameters led to biologically plausible running gaits. McMahon and Cheng [77] focused on how spring stiffness varied with different gait features. These two works introduced the two-dimensional spring mass model for running and inspired a decade of biomechanical studies confirming its predictive utility across a wide spectrum of animals [3, 20, 36, 44, 45, 47, 49].

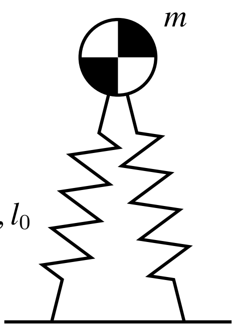
2.1.2 Control of The Spring Mass Model for Running

Before discussing running controls that have been developed for the spring mass model, we will more formally introduce the model. The planar spring mass model for running (Figure 2.1a) consists of a point mass m attached to a massless spring leg with stiffness k and rest length l_0. During the flight phase, the point mass follows a purely ballistic trajectory. The flight dynamics governing the center of mass motion are

$$\begin{aligned} m\ddot{x} &= 0, \\ m\ddot{z} &= -mg, \end{aligned} \tag{2.1}$$



(a) The spring mass model for running.



(b) The spring mass model for walking.

where (x, z) describes the Cartesian coordinates of the point mass and g is the acceleration due to gravity. The system instantaneously transitions into stance when the foot point touches the ground; the model does not consider any slipping of the foot. Once in stance, the system behaves as an inverted pendulum with an embedded spring. This results in the more complex center of mass stance dynamics

$$\begin{aligned} m\ddot{x} &= k\left[l_0\left(x^2+z^2\right)^{-1/2}-1\right]x, \\ m\ddot{z} &= k\left[l_0\left(x^2+z^2\right)^{-1/2}-1\right]z - mg. \end{aligned} \tag{2.2}$$

The model exits stance and returns to flight after rebound once the leg has fully extended back to its rest length.

Although the use of the spring mass model as a predictor of locomotion dynamics has its roots in biomechanics, the model has found widespread use in robotics as a template for designing and controlling running robots. Raibert was the first to recognize that springy mechanisms could be used to construct legged robots capable of running and hopping [93]. Throughout the 1980s, his lab produced a number of successful monopedal, bipedal, and even quadrupedal robots using pneumatic spring legs. Raibert's lab also studied these machines in simulation using both complex spring-embedded rigid body models and the simple planar spring mass model [92]. Despite these simulation studies, all of Raibert's robots utilized an intuitive control policy based on legged locomotion insights. This control scheme focused on regulating three essential running gait quantities: body orientation, apex height, and forward velocity. Body orientation was handled with a proportional-derivative controller to servo the torso in stance. Apex height was regulated by applying a fixed impulse along the leg during stance. And forward velocity was handled using a heuristic foot placement law for the horizontal foot position

$$x_{\text{foot}} = x_{\text{neutral}} + k_{\dot{x}}(\dot{x} - \dot{x}^*), \tag{2.3}$$

where $k_{\dot{x}}$ was an empirically tuned gain, $\dot{x}$ was the robot's forward velocity, and x_{neutral} was the neutral foot point that would generate zero net forward acceleration. This last quantity was estimated as half of the previous stance time multiplied by the forward speed, which approximately equals the horizontal midpoint during a symmetric running stance.

The success of Raibert's legged machines was shortly followed by a number of studies throughout the 1990s that formally investigated the control and stability of spring mass model running. This began with McGeer, who analyzed a three-link rigid body biped model with spring legs and spring joints. He found which mechanical parameters led to passive periodic running and which mechanical parameters could be actively stabilized using hip torque and leg thrust [74]. This analysis involved adjusting model parameters $\boldsymbol{p}$ while numerically calculating the takeoff-to-takeoff return map $\mathbf{R}(\boldsymbol{s}, \boldsymbol{p})$ of the system using a five-dimensional state vector $\boldsymbol{s}$. This technique of analyzing a system's return map is often referred to as Poincaré analysis and is a common method of studying the stability of periodic spring mass model gaits. By defining a Poincaré section as a single event in the gait cycle, the corresponding return map explains how the dynamical system evolves from cycle to cycle. Fixed points on the return map confirm the presence of stable periodic behavior, while points on the return map whose Jacobian eigenvalue lies within the unit circle confirm local asymptotic stability. Ultimately, the shape of a return map is determined by both static model parameters $\boldsymbol{p}$ and changing control inputs $\boldsymbol{u}$. M'Closkey and Burdick used this technique to evaluate the stability of Raibert's planar hopper by modeling a spring mass system with a nonlinear air spring along with Raibert's apex height regulation and foot placement algorithm [75]. Schwind and Koditschek expanded on this analysis by investigating the system with only the foot placement algorithm [103]. They chose the moment of apex during flight as the Poincaré section, which can be fully described with just two state variables: vertical height and forward speed. This study identified a method of increasing the controller's basin of attraction using a modified leg placement algorithm.

Numerous control policies for the spring mass model have been investigated using Poincaré analysis but those which can be formulated using optimal control are of particular interest. When a target Poincaré section state $\boldsymbol{s}^*$ is specified given a current section state $\boldsymbol{s}$, the controller can be formulated as an optimization problem

$$\boldsymbol{u} = \underset{\boldsymbol{u}}{\operatorname{argmin}} \ \left\| \boldsymbol{s}^* - \mathbf{R}(\boldsymbol{s}, \boldsymbol{p}, \boldsymbol{u}) \right\|, \tag{2.4}$$

which attempts to find the control input $\boldsymbol{u}$ that minimize the distance between the target and current state after one cycle. In the case of the running spring mass model with foot placement as input, control strategies have been identified which bring this distance to zero after one step. This is often referred to as deadbeat control and has been used to create optimal running controllers for the spring mass model [101]. Generalizations of this control have led to swing leg retraction policies which remain deadbeat even in the presence of unobserved ground height changes [110, 111]. These controllers have also been extended beyond the sagittal plane to control running and turning for the 3-D spring mass model [25, 26, 130].

2.1.3 Unification with Walking and Gait Transitions

The spring mass model can also be utilized to describe walking by including a second massless spring leg (Figure 2.1b). While the flight and single stance phase dynamics remain identical to those of the running spring mass model, an additional set of dynamics equations is needed to represent the double stance phase

$$\begin{bmatrix} m\ddot{x} \\ m\ddot{z} \end{bmatrix} = \begin{bmatrix} \frac{x-x_{f_1}}{l_1} & \frac{x-x_{f_2}}{l_2} \\ \frac{z}{l_1} & \frac{z}{l_2} \end{bmatrix} \begin{bmatrix} k(l_0 - l_1) \\ k(l_0 - l_2) \end{bmatrix} - \begin{bmatrix} 0 \\ mg \end{bmatrix}, \tag{2.5}$$

where l_i represents the lengths of the two spring legs and x_{f_i} represents the horizontal position of the two foot points. This extension allows the spring mass model to exhibit both running and walking gaits within a single simple mechanical system.

Despite early demonstrations of walking on Raibert's bipedal spring mass robots [55], spring mass model walking was not formally investigated until the mid 2000s. Motivated by the inability of stiff leg models to truly predict walking dynamics [50], Geyer and colleagues examined how a bipedal spring mass model could be used instead [53]. They found that this compliant leg system could demonstrate a wide range of gaits including both walking and running. A later experimental study found that this model is capable of accurately describing human walking at moderate speeds, but deviates at slow and fast speeds [70].

Recently, the ability of the spring mass model to generate both walking and running behaviors has sparked interest in how it can also be used to study transitions between these gaits. Salazar and Carbajal were first to investigate if these transitions existed at a fixed energy level using only foot placement controls [100]. They demonstrated that gait transitions could indeed be achieved over the course of several steps, but did not explore how best to design a gait transition controller. This study also revealed that some of these transitions policies required the system to enter grounded running, a gait observed in humans and animals [79, 97]. Shahbazi and colleagues further examined how gait transitions could be achieved when the bipedal spring mass model was also allowed to change stiffness at certain events in the gait cycle [112, 113]. They utilized optimal control across multiple return maps to select spring stiffnesses and foot positions that quickly achieved desired gait transitions. However, several biologically inspired constraints and mathematical approximations were employed to simplify the optimization problem.

2.2 Optimal Control for Legged Locomotion

The extensive use of Poincaré return maps to develop optimal control policies for the spring mass model can be attributed to both their ease of use and repeatedly successful application. However, the power of Poincaré techniques is limited by two factors. First, unless a closed form solution to the return map is known, an exhaustive state and action space search is required.

As a result, the method extends poorly to systems with high dimensional states or several control inputs. Second, although choosing a Poincaré section collapses one of the state dimensions, it ultimately limits the control policies that can be generated. For example, when midstance is chosen as the point of analysis, the resulting control will never consider actions at different times in the gait cycle. This is acceptable in some cases, such as running, where deadbeat solutions exist for many states, but it can also limit the controller performance when no single step solutions exist.

Ideally, we wish to solve spring mass model control as an optimal control problem using tools that consider all of state space and scale well with problem dimensionality. Reinforcement learning is one tool that addresses how a goal-directed agent can interact with its environment to optimize a numerical reward. In general these methods involve computing a policy based on a continual reward signal in order to maximize the reward in the long run, represented as a value function. These techniques have been successful within the computer graphics community, even with high dimensional state and action spaces [86, 87]. In this section we will address how reinforcement learning can be utilized to develop optimal controls for the spring mass model.

2.2.1 Comparison of Approaches

Although reinforcement learning has become popular in recent years, several other approaches can also be used for attempting to find optimal control policies. Due to the long history of reinforcement learning within the field of artificial intelligence, the classification of many optimal control methods is not well defined. Thus, a wide range of optimal control methods can be considered as part of reinforcement learning. The common root among these methods is the introduction of the Bellman optimality equation in the 1950s by Richard Bellman. Algorithms suitable for solving control problems using this equation became known as dynamic programming algorithms [14]. Dynamic programming is widely used for solving general optimal control problems, but it suffers from what Bellman called the “curse of dimensionality” because its computational complexity grows exponentially with the number of state dimensions. Nevertheless, dynamic programming has been utilized extensively along with many techniques to avoid this dimensionality issue. Dynamic programming and reinforcement learning solve the same type of problems, which makes it difficult to consider them as separate classes.

Dynamic Programming

Before considering how dynamic programming could be used to solve unified spring mass model control, we will first address the feasibility of a brute force search. The bipedal spring mass model (Figure 2.1b) can be fully defined at any point in the gait cycle using a six dimensional state vector $(x, z, \dot{x}, \dot{z}, x_{f_1}, x_{f_2})$ where x_{f_1}, x_{f_2} describe the horizontal foot positions relative to the center of mass. One additional variable is needed to store the gait phase (flight, single stance, or double stance). There are many action parameterizations that can be used, but one example that has worked well for gait transitions is the three dimensional vector defining the two spring stiffnesses and the next leg placement angle. If each of these state and action vector quantities are coarsely discretized into 10 bins, the system will have $n = 10^6$ states and $k = 10^3$ control actions. This corresponds to a total of $k^n = 10^{3000000}$ possible policies that would need to be evaluated.

Fortunately, dynamic programming allows us to solve this problem exponentially faster than an exhaustive search. Popular methods such as policy iteration or value iteration can iteratively find approximate optimal solutions in polynomial time [117]. For instance, the computational complexity of value iteration is $\mathcal{O}(n^2k)$ per iteration. Despite this remarkable decrease in complexity, conventional dynamic programming methods are still not feasible in many cases due to the need to iterate over every state. One popular extension is to use asynchronous methods which visit the states in any order. This can be noticeably more efficient when the optimal solution remains in a small subset of the state space.

Q-learning approaches

Tabular reinforcement learning methods, such as conventional SARSA or Q-learning, can also be used to solve this problem efficiently using temporal difference learning. These algorithms, similar to dynamic programming, iteratively improve their solutions by using previous estimates to generate new solutions, a feature known as bootstrapping. These techniques do not require a model and can be implemented online in an incremental fashion. These tabular algorithms are provably convergent but are mathematically optimal only for truly discrete systems [117, 126]. When discretizing a continuous-time system, these methods produce an approximation of the true optimal solution.

One pertinent disadvantage of the methods discussed so far is that they do not extend well to continuous state and action spaces. Dynamic programming can be improved in this case by randomly sampling actions over a continuous space [7]. Alternatively, approximate dynamic programming methods, such as fitted value iteration, offer compact memory-efficient representations of the value function or policy using function approximators [22].

Function approximators allow value functions to generalize single updates across multiple states, but can introduce a bias and even prevent guarantees of convergence. Nevertheless, function approximators have been widely used to address continuous systems and have demonstrated exceptional performance in practice.

Value function approximation is often accomplished using stochastic gradient descent methods to minimize the mean squared value error (MSVE). When a simple linear function approximator is used this can lead to a single optimum. A typical linear function approximator is $\hat{V}(\boldsymbol{\theta}, \boldsymbol{s}) = \boldsymbol{\theta}^T \phi(\boldsymbol{s})$, where $\boldsymbol{\theta}$ represents the weight vector and $\phi(\boldsymbol{s})$ is a feature vector for state $\boldsymbol{s}$. The simple stochastic gradient descent update rule can be written as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i + \alpha(U_i - \hat{V}(\boldsymbol{\theta}_i, \boldsymbol{s}_i))\nabla\hat{V}(\boldsymbol{\theta}_i, \boldsymbol{s}_i), \tag{2.6}$$

where α is the learning rate and U is the new function estimate. If U is an unbiased estimate of the true value function and the function appoximator is linear, then this algorithm is guaranteed to converge to the discrete problem's global optimum [17]. Unfortunately, bootstrapping methods are biased because they depend on the current weight vector $\boldsymbol{\theta}$. When temporal difference methods are used with a linear function approximator, the gradient descent algorithm instead converges to the TD fixed point, which is within a bound of the optimum [117].

Policy gradients

Policy-gradient algorithms also leverage gradient descent in order to update a functional approximation of the policy based on the gradient of a performance metric [88, 118, 128]. One strong benefit of these methods is that they allow continuous policies and action spaces to be utilized. Actor-critic algorithms arise when policy-gradients are combined with explicit approximations of the performance metric. The policy is referred to as the actor, while the performance metric is the critic. This has the benefit of potentially reducing variance of the gradient estimate compared to full Monte Carlo samples.

One last reinforcement learning algorithm that is worth mentioning in the context of legged locomotion is the continuous actor-critic learning automaton (CACLA) [125]. This algorithm has been particularly successful in demonstrating dynamic legged locomotion behaviors in simulation [86, 87]. CACLA operates similar to policy-gradient actor-critic methods, but rather than updating the actor in policy space it updates the actor in action space. In addition, actor updates are only performed when the sampled action improves the performance metric. This prevents the actor from moving towards unsampled actions.

2.2.2 Applied to Bipedal Robot Hardware

Reinforcement learning has been applied to both simulated and real-world systems for solving a variety complex dynamic tasks. Learning to walk on bipedal robot hardware has been demonstrated numerous times using conventional reinforcement learning techniques. Although the approach taken in this thesis focuses on offline computation of control policies, many successful walking controllers have been learned online by leveraging the model-free ability of reinforcement learning algorithms.

Benbrahim was one of the earliest to demonstrate using modern reinforcement learning methods to walk on a physical biped [16]. His doctoral dissertation involved implementing an actor-critic framework using neural networks on a small biped during the mid-1990s. He reduced the dimensionality of the problem using central pattern generators (CPGs). Although the robot he worked on utilized weak electric actuators and was attached to a walker, it demonstrated a slow walking gait after three hours of online learning.

CPGs have been used several times with reinforcement learning methods on biped hardware [39, 81, 84]. Researchers have utilized this approach to demonstrate slow stable walking gaits on small humanoids after significant numbers of trials. Tedrake and colleagues demonstrated walking from actor-critic reinforcement learning on biped hardware without the use of CPGs, but on a small passive dynamic walker outfitted with just two degrees of actuation [122]. Morimoto and Atkeson demonstrated planar walking on a small five-link biped using reinforcement learning to approximate Poincaré maps and value functions [82]. This system was capable of stable walking within 100 trials.

A notable recent example of gait optimization on robot hardware is the work by Calandra and colleagues using Bayesian optimization [23, 24]. Bayesian optimization is an alternative black-box optimizer that is particularly efficient when policy evaluations are expensive, such as with robot hardware trials. This work is pertinent to this thesis because it addresses automatic gait optimization on a small planar biped with spring legs. Bayesian optimization may potentially learn more efficiently than reinforcement learning on hardware, but the successful performance in this work shows promise for gait optimization on springy robots.

2.3 Control Realization on Hardware

The reality of developing control policies for a virtual model is that it does not directly transfer to hardware. Physical legged robots typically have more degrees of freedom and more dynamics, which can require additional models and layers of control. This section will address how control policies have been realized on humanoid robots in the context of simplified models and gait transitions.

2.3.1 Methods of Embedding Center of Mass Behaviors

Heuristic strategies are some of the least complex yet best performing methods for translating a general center of mass behavior to robot hardware. Raibert's running machines [93] regulate their hopping height, and thus energy, using a fixed leg thrust force during stance coupled with a feedback controller to stabilize body pitch. Similarly, Siavash and colleagues presented a bipedal walking controller on ATRIAS using a spring law with variable rest length for energy regulation and a separate orientation feedback controller [95]. The simple biped control (SIMBICON) framework [132] is another heuristic strategy for walking and running that has been used to demonstrate a variety of gaits on simulated systems. This framework combines decoupled controllers for foot placement, torso regulation, and joint-level position feedback.

Despite the success of heuristic controllers, they do not explicitly compute the motion of the center of mass. Operational space control, also referred to as task space control, uses a full-order model of the system dynamics to find the control inputs which best achieve desired accelerations [63]. This approach allows center of mass motion goals to be defined explicitly using accelerations without having to define individual joint trajectories. Quadratic programming (QP) is a popular and effective way to formulate task space control for high degree-of-freedom humanoid robots. This approach uses numerical optimization to compute joint torques given a set of chosen tasks prioritized for legged locomotion. Feng and colleagues used this approach to track specific center of mass motions and centroidal momentum on an ATLAS robot [48]. Several other groups have also used QP task space control on ATLAS robots to track quantities such as the instantaneous capture point [64] and desired body coordinate frames [67]. This class of full body controllers has even been utilized for tracking spring mass model foot placement targets on simulated 3D humanoid robots [127].

The optimization involved in full body QP task space control typically has high computational costs, and thus is only applied on the current time step. The disadvantage of this approach is that it neglects future dynamics of the system that could be leveraged. One technique for avoiding this problem is to use a less complex, intermediate model with essential degrees-of-freedom that can be efficiently simulated. For instance, Kuindersma and colleagues precede their full body task space control with a time-varying linear quadratic regulator (LQR) to stabilize a simplified model [67]. Centroidal dynamics models [69, 85] are an attractive choice of simplified model for capturing the rotational dynamics of humanoid robots. This approach can provide greater control over the angular momentum of legged systems.

2.3.2 The Lack of Controlled Gait Transitions for Bipedal Robots

Relatively few bipedal robots have demonstrated controlled gait transitions between walking and running. Hodgins presented the earliest example of controlled gait transitions on a planar two-legged machine [55]. This work adopted an event-based heuristic strategy for switching between walking and running gaits. During the middle of the single stance phase, the leg was either lengthened or shortened to transform between vertical walking and running motions. Running was controlled using the classic Raibert hopping approach, while walking used constant length legs and constant foot placement without velocity feedback.

Honda's humanoid ASIMO robot platform has demonstrated 3D walking, running, and transitions between using a model-based approach [119, 120]. Takenaka and colleagues designed ASIMO's walking and running gaits as cyclic trajectories for the zero moment point, feet, and upper body using simplified dynamics models. This approach relies on matching specific boundary conditions between cycles to change gait patterns. Scientists have also demonstrated model-based gait transitions on the MABEL [116] and RABBIT [83] planar bipeds using hybrid zero dynamic frameworks. Walk-to-run transitions are accomplished by blending the virtual constraint at the end of the walking gait with the virtual constraint at the beginning of the running gait. Run-to-walk transitions rely on discretely switching to the walking controller modified with a virtual compliant element. These strategies result in gait transitions within one or two steps.

Gait transitions have also been shown in 3D using the ATRIAS biped platform [57]. Researchers at Oregon State University developed a heuristic control strategy that can transition from walking to running when the desired forward velocity is increased beyond 2.0 m/s. This controller provides a smooth, continuous transition between walking and running gaits, but does not explicitly choose whether it walks or runs.

Despite this handful of gait transition examples, there are a considerable number of humanoid robots that only target single gaits. Since gait transitions have been successfully demonstrated on bipeds, it is reasonable to ask why gait transitions controllers are not more common. The simplest explanation is that many biped platforms do not yet have the necessary power density to run. For example, the ATLAS robot that was used during the 2015 DARPA Robotics Challenge weighs 150 kg. In order to generate running motions similar to a human, it needs to create peak vertical ground reaction forces approximately three times its body weight. This would require greater than 4400 N of force to be generated within fractions of a second. The latest version of this robot from Boston Dynamics is significantly lighter, weighing just 75 kg, due to 3D printing techniques [38]. This robot has demonstrated running and jumping with long flight phases. Bipedal robots and humans alike must create very large ground reaction forces in order to run.

Chapter 3

Optimal Locomotion Policies for Point Mass Models

Scientific advances in legged robot locomotion have led to an increased need for understanding how legged systems transition between different gait patterns. Although a significant amount of work has analyzed the capability of local controllers near individual gaits [61, 89, 90, 93], less work has studied the capability of global controllers spanning all gaits. Transient behavior that occurs while transitioning between steady state gaits can lead to instability and ultimately limits which transitions are feasible for real legged systems. It is therefore important to ask two scientific questions about gait transitions. First, what are the theoretical limits of transition control? This question investigates which gaits can be attained and which disturbances can be tolerated without failure. Second, what are the fundamental rules that govern the control? This question identifies how features of the system state translate into the control policy. Advancements in parallel computing allow us to approach both of these questions using techniques from numerical analysis and optimal control.

Legged systems are capable of a diverse range of dynamic movement patterns during locomotion. Today's bipedal robots have demonstrated many of these gaits and transitions between them using internal models of varying complexity [55, 57, 83, 116, 119, 121]. Despite the ability of these locomotion controllers to transition between walking and running, few studies have investigated the robustness of these transitions to disturbances. Furthermore, it is not yet clear globally which robot states can reach these target gaits. Understanding both of these limitations will improve the reliability of bipedal robot locomotion across multiple gaits by increasing the certainty of transitions.

Evaluating the global set of reachable gaits for a legged system poses complexity issues for high-dimensional robot hardware. A common approach to make global analysis computationally feasible is to use low-dimensional representative models of the system [34]. Bipedal robots are often described using multiple simplified dynamics models, such as the linear inverted pendulum (LIPM) for walking [60] and the spring mass model (SMM) for running [19]. Attempting to transition between different models can lead to a sharp discontinuity in the dynamics. This limits the range of dynamic motions and increases the complexity of designing robust control policies. The SMM, however, is also capable of describing walking [53] and gradual gait transitions using foot placement [73]. Recently, Shahbazi and colleagues have presented a unified SMM controller capable of rapid transitions between walking and running [112, 113]. By altering leg stiffness at specific points during the gait cycle as well as foot placement, the system is capable of transitioning between gaits with different energy levels.

There are two primary limitations to existing SMM gait transition policies. The first is that they rely on a number of heuristic decisions made during the design process which constrain the set of control sequences that can be made. These decisions include when to apply actions during the gait cycle and what intermediate targets should be reached before the final goal. This reduces the policy search space, so biomechanical observations are often used to motivate design choices. The second limitation of SMM gait transition policies is due to the chosen control parameterization. Infrequently varying the linear spring stiffness represents only a small subset of leg force profiles that are potentially useful during gait transitions. Without comparing to more general leg force policies, it is difficult to know how the chosen parameterization affects robustness and state space coverage. In contrast, our goal is to evaluate SMM gait transition policies that make no simplifying control assumptions and thus correspond to the largest possible basin of attraction for a bipedal point mass system.

In this work we globally analyze controllers for optimally transitioning between walking and running on a simple bipedal point-mass system with compliant legs. We focus on the SMM but generalize the control to allow for continuously variable leg stiffnesses while maintaining a low problem dimensionality. We numerically generate approximate globally optimal policies that can achieve transitions in order to analyze their basin of attraction and robustness to disturbances. We compare these policies with realistic parameterized control laws that can be more conveniently implemented on robot hardware.

3.1 Choosing a Dynamics Model

Biomechanical studies have shown that bipedal walking and running are both characterized by periodic vertical excursions of the CoM [33]. Simplified models capable of both gaits must encode these oscillations into their dynamics; leg compliance offers a compact representation. The bipedal SMM uses compliance to reproduce a wide range of basic human locomotion mechanics [53] with a low-dimensional state representation. The system is modeled as a point mass with massless spring legs, which act independently on the CoM during the stance phase. Conventionally, this system is conservative and maintains a constant mechanical energy by only altering its foot placement at each footstep. This restricts the system to states at a fixed energy level and therefore limits the range of achievable motions. Researchers have investigated allowing the system energy to vary by modifying its spring forces with variable stiffness and damping [104, 112]. The larger control dimensionality can greatly increase the size of the policy space, and potentially the policy performance, at the expense of increased problem complexity. Rather than use heuristics to reduce complexity, our goal in this work is to examine a large, unconstrained policy space that can produce a wide range of gait transitions. Therefore, we focus on policies with continuously variable spring stiffnesses that can produce arbitrary force profiles.

We parameterize the bipedal SMM with a mass m and two massless legs of rest length l_0. The coordinates of the point mass are denoted as (x, z) to describe its horizontal and vertical position. We analyze the system during flight using a 3 dimensional state vector, $(z, \dot{x}, \dot{z})$; during the single and double support phases, this state is increased to 4 or 5 dimensions by the horizontal foot positions, Δx_1 and Δx_2 (Fig. 3.1). The dynamics of this model are

$$\begin{bmatrix} \ddot{x} \\ \ddot{z} \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \frac{\Delta x_1}{l_1} & \frac{\Delta x_2}{l_2} \\ \frac{z}{l_1} & \frac{z}{l_2} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} - \begin{bmatrix} 0 \\ g \end{bmatrix}, \tag{3.1}$$

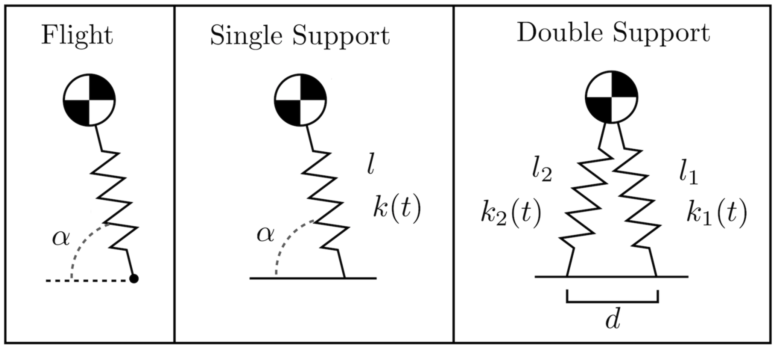


Figure 3.1: A point-mass biped model with spring legs encapsulates three distinct gait phases: flight, single support, and double support. These phases have state vector dimensionalities of 3, 4, and 5 respectively. Although the points of applied ground force change discontinuously after a phase transition, the center of mass state trajectory remains continuous due to the massless limbs and feet. If the leg force profiles are constrained to begin and end at zero, such as with ideal linear springs, the system acceleration will also be continuous during phase transitions.

where $g = 9.81\text{m} \cdot \text{s}^{-2}$ is the gravitational acceleration, $l_i = (\Delta x_i^2 + z^2)^{1/2}$ are the leg lengths and $F_i = k_i(l_0 - l_i)$ are the spring leg forces of stiffness k_i during stance and zero otherwise. Our analysis considers leg policies that control landing angle during swing and continuously vary leg stiffness during stance. The system is allowed to transition between its three distinct gait phases (flight, single support, and double support) based on specific zero crossing events corresponding to touchdown and takeoff. The takeoff events are triggered by the value of $\Delta x_i^2 + z^2 - l_0^2$ while the touchdown events are triggered by the value of $l_0 \sin(\alpha) - z$, where α is a chosen leg landing angle. The tolerance of these zero crossing events is chosen to be 10^{-6}s.

To summarize the above section, for a spring mass model, from a wide range of possible actions, such as virtual stiffness, leg angle, damping, rest length, we choose the following:

1. Virtual leg stiffness: Inspired from biological systems, virtual leg stiffness can be used to regulate the energy of the system in a continuous

fashion.

2. Leg angle : The landing leg angle of attack at the end of swing.
3. Damping : In some experiments, we also include a linear or non-linear damping term that also helps regulate the energy of the system.

For the state representation, we choose the following parametrization, depending on the phase of the system:

1. Flight phase: Height, forward velocity and vertical velocity $(z, \dot{x}, \dot{z})$
2. Single support phase: Height, forward velocity, vertical velocity and horizontal foot position with respect to the center of mass $(z, \dot{x}, \dot{z}, \Delta x_1)$
3. Double support phase: Height, forward velocity, vertical velocity and horizontal foot positions with respect to the center of mass $(z, \dot{x}, \dot{z}, \Delta x_1, \Delta x_2)$

3.2 Choosing Problem Boundaries

The dynamics of the bipedal SMM (equation 3.1) form a set of nonlinear equations which constrain the search for optimal policies. This problem requires a non-convex optimization over a continuous state and action space. It is therefore necessary to bound the search domain with finite limits. If the bounds are too large, the problem may become computationally intractable; if the bounds are too small, the problem may ignore important solutions. Thus, it is crucial to choose these bounds carefully to encapsulate all motions of the system that are pertinent to bipedal locomotion. Here we examine how state and action bounds were chosen in this work using justifications from biology. Although the use of biological insights to direct our search is somewhat antithetical to a heuristic-free approach, we use these guidelines to select generous ranges covering most human walking and running gaits.

3.2.1 Virtual Leg Stiffness

A key parameter of the spring mass model is the stiffness, k, of the virtual spring leg connecting the foot point to the center of mass. It is reported in theoretical simulation studies of the model and biomechanical experiments attempting to fit to the model. However, there are three minor pitfalls with using this data. First, the majority of these works only examine running gaits and do not consider a walking SMM. Second, the range of reported values varies significantly between different studies; it is therefore important to compare dimensionless stiffnesses scaled by mg/l_0. Third, stiffness is commonly presented as either a vertically-aligned spring, leg-aligned spring, or foot-to-CoM-aligned spring, so some care must be taken to understand how these values relate to the SMM.

Blickhan simulated SMM stiffnesses as high as 180 mg/l_0 but estimated human running stiffnesses to be between 14 and 43 mg/l_0 when running at horizontal speeds of 3 to 9 $\mathrm{m} \cdot \mathrm{s}^{-1}$ [19]. McMahon and Cheng examined theoretical leg stiffnesses up to 120 mg/l_0 for extreme velocities, but estimated a much lower stiffness of 15.5 mg/l_0 for typical human running [77]. Farley and Gonzalez performed human running experiments at 2.5 $\mathrm{m} \cdot \mathrm{s}^{-1}$ while varying stride frequency and found leg stiffnesses ranging from approximately 10 to 20 mg/l_0 [45]. Arampatzis et al. performed detailed human running measurements to find slightly higher stiffnesses ranging from approximately 30 to 45 mg/l_0 for horizontal speeds between 2.5 and 6.5 $\mathrm{m} \cdot \mathrm{s}^{-1}$ [6]. Finally, although few studies report walking stiffnesses, Lipfert et al. fit bipedal SMM models to walking data and found stiffnesses of 48 to 34 mg/l_0 at speeds of 1.0 to 2.6 $\mathrm{m} \cdot \mathrm{s}^{-1}$ [70]. Combining all of these results shows that most typical human gaits involve virtual leg stiffnesses between 10 and 50 mg/l_0.

3.2.2 Virtual Leg Touchdown Angle

The touchdown angle, α, that the virtual leg makes with the horizontal ground, sometimes referred to as the angle of attack, has a large impact on the time-evolution of SMM dynamics. When examining biomechanical locomotion data, this quantity should not be confused with the foot-to-hip angle, which is often slightly more oblique given an upright torso. Naively, we can bound this quantity between 0 degrees (i.e. foot directly in front of CoM) and 180 degrees (i.e. foot directly behind the CoM). A brief survey of papers which simulate the SMM for running [77, 110] and for walking [53, 98] show that nearly all forward locomotion solutions use touchdown angles greater than 40 degrees. If we further constrain the range using biomechanical running and walking data from [70], we see that nearly all reported touchdown angles are greater than 70 degrees. Choosing the upper bound to be the lower bound reflected about the vertical, allows for symmetric motions backwards and forwards. If we decrease the upper bound to be closer to 90 degrees, we eliminate the solutions which step backwards or accelerate quickly.

3.2.3 Virtual Leg Compression

The state of the SMM can be conveniently represented in either Cartesian coordinates, using horizontal and vertical foot positions, or polar coordinates, using leg lengths and leg angles. Although both forms are valid, ensuring that kinematic constraints are obeyed during stance is more straightforward when directly storing leg lengths. This avoids having to eliminate foot positions which may exceed the model's rest length, l_0, when building discretized state space grids. This quantity can be bound by specifying a maximum leg compression. Lee and Farley presented center of mass trajectories for human walking and running, which indicated vertical displacements of approximately 8 cm during running and 3 cm during walking [68]. Gard et al. also recorded center of mass trajectories during walking and found vertical excursions of approximately 2 to 5 cm. [51]. These vertical measurements provide a reasonable upper bound for the expected leg compression during typical bipedal gaits.

3.2.4 Horizontal and Vertical Center of Mass Velocity

The translational velocity, $[\dot{x}, \dot{z}]$, of the center of mass is an important quantity for classifying gaits because it defines a point-mass system's kinetic energy. Although the presence of flight or double support phases can be used to label gaits as running or walking, this definition incorrectly labels grounded running, sometimes called Groucho running [79], which has the same limb duty factor characteristics as walking [97]. A dynamical distinction can be used to resolve this by examining the fluctuations of potential and kinetic energy of a biped's center of mass (figure 3.2). Running occurs when both energies are in phase with each other, such as with a bouncing rubber ball [30]. Walking occurs when both energies are out of phase with each other, such as with a rolling egg [29]. This definition even holds up during multi-step gait transitions, where the relative energy phases suddenly switch [107].

Bounding the SMM translational velocities to search over is a simple matter of deciding how fast gaits should move and how high the system center of mass should rise. A reasonable starting point for expected horizontal velocity is the preferred human walking speed of approximately 1.2 $\mathrm{m \cdot s^{-1}}$ [28] and the preferred human run-to-walk transition speed of approximately 2.0 $\mathrm{m \cdot s^{-1}}$ [105]. Running up to 3.0 $\mathrm{m \cdot s^{-1}}$ is often considered a slow jog-

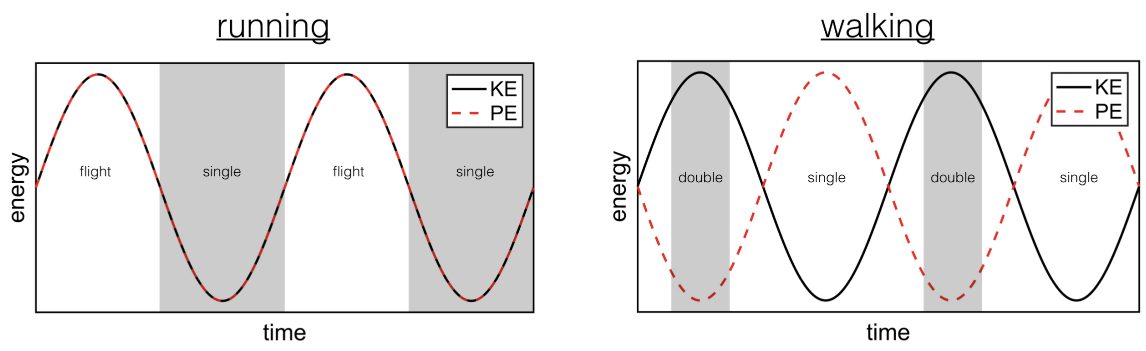


Figure 3.2: The dynamical classification of running and walking gaits is based on fluctuations in the kinetic (KE) and potential (PE) energies of a system's center of mass. A bipedal system is running when these energies are in phase and walking when these energies are out of phase. In this work we identify a characteristic state at the start of single support belonging to each specific gait. Target states which are rising at the onset of single support are considered to be walking, while target states which are falling at the onset of single support are considered to be running.

ging pace and running at 6.0 $\mathrm{m \cdot s^{-1}}$ is a fast running pace [62]. The limits of vertical velocity can be based on the expected apex distance using standard ballistic equations. A point-mass with upward velocity in the absence of external forces will rise a distance of $\dot{z}^2/2g$ before apex. If we combine this with center of mass trajectory data [27, 68], we see that most vertical velocities do not exceed 1.2 $\mathrm{m \cdot s^{-1}}$.

3.2.5 Step Length

The final parameter we must consider is how far apart the model's feet can be placed during the double support phase. The feet locations can alternatively be stored as separate variables, however that option will introduce duplicate and infeasible states of the system. Encoding the state with step length ensures distinct foot holds and consistent ordering of the state variables. The double support SMM state can then be fully determined by combining step length with each leg length to form a triangle. Note that the bounds must be

	Symbol	Typical Human	Search Bounds	Units
Leg Stiffness	k	10 to 50	10 to 90	mg/l_0
Touchdown Angle	α	70 to 110	40 to 140	deg
Leg Compression	Δl	2 to 8	0 to 20	cm
Horizontal Velocity	$\dot{x}$	1.2 to 6.0	-1.0 to 4.0	m/s
Vertical Velocity	$\dot{z}$	-1.2 to 1.2	-2.0 to 2.0	m/s
Step Length	d	0.4 to 1.1	0.25 to 1.30	m

carefully set to obey the triangle inequality so that no two sides are less than or equal to a third side.

There are several biomechanical studies involving human step length. Sekiya et al. conducted experiments to measure the variability of step length during human walking and found lengths ranging from approximately 0.4 to 1.0 m [108]. Donelan et al. measured mechanical work during human walking while testing step lengths from 0.4 to 1.1 m [37]. Segers et al. recorded step lengths during walk-to-run and run-to-walk transitions and measured lengths of approximately 1.1 m [106]. We can combine these results to observe that typical human double support phases do not exceed step lengths of 1.1 m.

3.3 Choosing an Optimization Framework

The optimal controls studied in this work fall into a class of optimization problems involving continuous-time control trajectories applied to nonlinear systems throughout continuous state space. More generally, this represents a non-convex optimization over a continuous domain with nonlinear constraints. Our search for these controls is further complicated by the desire to compute globally optimal solutions rather than more easily obtainable locally optimal solutions. Global optimums represent the best possible control strategies that can be used by our specific model to achieve specific long term goals defined by a cost function. Unsurprisingly, there are relatively few guarantees surrounding the search for this type of solution. This has not discouraged the field of dynamic optimization, which has produced a plethora of numerical methods aiming to compute optimal controls of varying accuracy. While many approaches can guarantee convergence to a local optimum, far fewer can guarantee a globally optimal solution. And even then, only by making certain assumptions within a bounded search space.

3.3.1 Candidate Optimization Methods

An ideal solution to our optimization problem would be an analytical, globally optimal control generated by solving requisite partial differential equations. Indeed, if the underlying Hamilton-Jacobi-Bellman equation were simple enough, we could attempt an analytical solution using Pontryagin's principle. Sadly, modern mathematical approaches have not yet yielded analytic integrations of our model's nonlinear dynamics and constraints. Until this occurs, we are resigned to numerical methods which can approximately compute these control laws.

Trajectory optimization

Trajectory optimization is a common approach to generating locally optimal numerical solutions for nonlinear systems. If the optimization is initialized very close to the global optimum, this class of iterative methods can even produce globally optimal trajectories. However, this feature is not particularly useful as the global solution it not typically known beforehand. The resulting solutions are also in the form of a time-based state or control trajectory rather than a functional mapping from arbitrary states to actions. This restriction coupled with locally optimal results makes trajectory optimization an unideal candidate in the search for globally optimal controls.

Reinforcement Learning

Reinforcement learning (RL) is a powerful optimization approach commonly used for high-dimensional control problems. It can achieve fast convergence and good results for complex problems, without an available model, with the help of function approximators such as neural networks and Gaussian processes. However, except in special cases, the solution obtained with RL with function approximation a local optimum. Moreover, the optimization is not guaranteed to converge, or might converge arbitrarily far away from the global optimum.

Dynamic Programming

Since we have a model of our system, we consider other approaches that might require more samples, but can maintain global convergence guarantees. One such method is Fast marching methods, also known as single pass or ordered upwind methods. These methods model the value function of the optimal policy by dynamic programming, requiring a model and very large amount of data. With a bounded state space and discretized control and state space, these approaches can guarantee convergence to the global solution of the approximate (discretized) problem. The found solution can be made arbitrarily close to the global optimum by making the discretization grid finer.

A special case of problems, represented by the Eikonal partial differential equations, can be solved very efficiently with a single pass through the state space. These problems can be solved with fast marching methods or ordered upwind methods by leveraging the property that the solutions to the PDE lie along some characteristic curves. However, for general systems these characteristic curves are not known, and hence cannot benefit from these approaches. Convergence speed of dynamic programming can still be improved for certain problems for which the characteristic solution is bounded within a certain distance of the state. However, in the most general case, one has to revert to classical dynamic programming, which can take several passes through the state space to converge to the optimal value function. While this might be very slow, it is still guaranteed to reach the globally optimal solution for very general problem.

3.3.2 Semi-Lagrangian Dynamic Programming

Although numerical optimal control schemes which rely on discretization ultimately compute suboptimal solutions for continuous systems, a human designer can balance solution accuracy with computational load. The deciding factor is the problem dimensionality defined by its state and action spaces. As dimensionality increases, computational requirements explode exponentially due to the "curse of dimensionality" [13]. Many optimization methods compensate for this in high dimensions by introducing sparse function approximators at the cost of introducing local optimums. If problem dimensionality is low enough, as is the case with the model used here, linearly interpolated grids can be used to ensure an approximate global optimum. This solution is suboptimal in a strict mathematical sense, but lies within a bound of the true optimal control [17, 43].

Our search for the theoretical limits of bipedal gait transitions requires finding which states are able to reach different target states that correspond to distinct gaits (figure 3.2); this region of state space is the basin of attraction for a specific policy (figure 3.3). Previous work on the SMM has analyzed discrete control inputs to find one-step [101, 130] and multi-step [25, 34] deadbeat policies capable of reaching a target state in the fewest footsteps. This calculation of globally optimal deadbeat policies reveals the basin of attraction for a target set $\mathcal{T}$. This basin of states can be written formally as the n-step discrete (backwards) reachable set,

$$\mathcal{R}_d = \{\boldsymbol{s} \in \mathcal{S} : N(\boldsymbol{s}) \leq n_{\text{max}}\}, \tag{3.2}$$

where $\mathcal{S}$ is the set of all states, $N(\boldsymbol{s})$ is a function that returns the minimum number of foot steps to reach $\mathcal{T}$ from $\boldsymbol{s}$, and n_{max} is the maximum number of considered steps. In comparison, our analysis finds the reachable set for target states using continuous-time control input. This straightforward extension requires a continuous-time (backwards) reachable set,

$$\mathcal{R}_c = \{\boldsymbol{s} \in \mathcal{S} : T(\boldsymbol{s}) < +\infty\}, \tag{3.3}$$

where now $T(\boldsymbol{s})$ defines the minimum time to reach $\mathcal{T}$ from $\boldsymbol{s}$ and the horizon is extended to include all states reachable in finite time. Thus, we have a well-defined optimal control problem to solve; calculate $T(\boldsymbol{s})$, the long term cost, over all of state space to determine the reachable set and the associated optimal control law.

In order to solve for $T(\boldsymbol{s})$, we must define a function that maps from state-

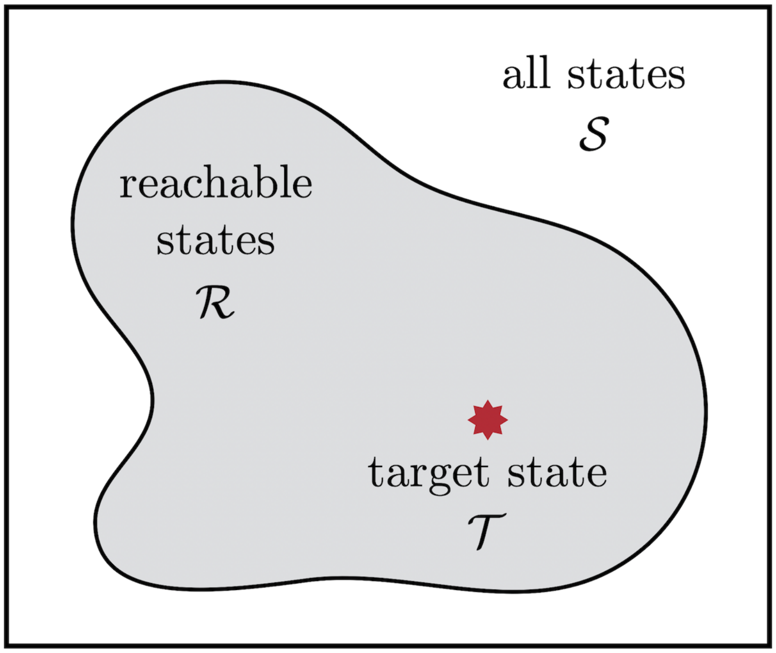


Figure 3.3: The optimal control targeting a specific state or set of states, $\mathcal{T}$, reveals which states, $\mathcal{R}$, in a space, $\mathcal{S}$, are able to reach the target. This is often referred to as the backwards reachable set or basin of attraction for a policy.

policy pairs to their first target arrival time,

$$t(s,\pi) = \begin{cases} \min_t f(s,\pi,t) \in \mathcal{T} & \text{if } \exists t \geq 0 : f(s,\pi,t) \in \mathcal{T}, \\ +\infty & \text{otherwise,} \end{cases}$$

where $f(s,\pi,t)$ returns the state of the system at time t starting from state s and following policy π. The value function can then be defined as

$$T(s) = \inf_{\pi \in \Pi} t(s,\pi), \tag{3.4}$$

where the set Π represents all admissible policies. In order to ensure that $T(s)$ is continuous around the target we must also assume small-time local controllability around $\mathcal{T}$ with the condition

$$\inf_{\boldsymbol{\pi}\in\Pi} \dot{f}(\boldsymbol{s},\boldsymbol{\pi},t)\cdot n(\boldsymbol{s}) < 0 \qquad \forall \boldsymbol{s}\in\partial\mathcal{T}, \tag{3.5}$$

where $n(\boldsymbol{s})$ is the exterior normal to $\partial\mathcal{T}$ at $\boldsymbol{s}$. This ensures the value function is Lipschitz continuous, however the scheme used here will still produce a reasonable approximation even if this condition is not satisfied [42].

Determining the value function is necessary to calculate the reachable set (equation 3.3), but this requires solving a non-convex, constrained optimization problem globally. Fortunately, the low-dimensionality of the problem makes it feasible to apply the Dynamic Programming Principle (DPP) [15] to find a direct numerical solution. The DPP for the minimum time problem provides a recurrence relation for the value function and can be derived [10] to be

$$T(\boldsymbol{s}) = \inf_{\boldsymbol{\pi}\in\Pi}\{t + T(f(\boldsymbol{s},\boldsymbol{\pi},t))\}.$$

This relation leads directly to a nonlinear, first order partial differential equation, known as the Hamilton-Jacobi-Bellman equation, whose solution is the value function of interest. We can use this equation to define the boundary value problem

$$\begin{cases} \sup\limits_{\boldsymbol{\pi}\in\Pi} -\nabla T(\boldsymbol{s})\cdot\dot{f}(\boldsymbol{s},\boldsymbol{\pi},t) = 1, & \boldsymbol{s}\in\mathcal{R}\setminus\mathcal{T}, \\ T(\boldsymbol{s}) = 0, & \boldsymbol{s}\in\mathcal{T}, \\ \lim\limits_{\boldsymbol{s}\to\boldsymbol{s}_0} T(\boldsymbol{s}) = +\infty, & \forall\boldsymbol{s}_0\in\partial\mathcal{R}. \end{cases}$$

This problem can be transformed into a bounded domain without dependence on the reachable set using the Kruzkov change of variables [66],

$$\begin{cases} v(\boldsymbol{s}) = 1 - e^{-T(\boldsymbol{s})} & \text{if } T(\boldsymbol{s}) < +\infty, \\ v(\boldsymbol{s}) = 1 & \text{if } T(\boldsymbol{s}) = +\infty, \end{cases} \tag{3.6}$$

in order to finally arrive at the boundary value problem

$$\begin{cases} v(\boldsymbol{s}) + \sup\limits_{\boldsymbol{\pi}\in\Pi} -\nabla v(\boldsymbol{s}) \cdot \dot{f}(\boldsymbol{s}, \boldsymbol{\pi}, t) = 1, & \boldsymbol{s} \in \mathcal{S} \setminus \mathcal{T}, \\ v(\boldsymbol{s}) = 0, & \boldsymbol{s} \in \partial\mathcal{T}. \end{cases}$$

In order to solve this problem using semi-Lagrangian numerical approximation, we apply time and space discretization along with the Kruzkov transform as described in [43]. This results in the iterative scheme

$$\begin{cases} v_j^{i+1} = e^{-\Delta t} \min\limits_{\boldsymbol{\pi}\in\Pi} I(V^i, f(\boldsymbol{s}, \boldsymbol{\pi}, \Delta t)) + 1 - e^{-\Delta t}, & \boldsymbol{s}_j \in \mathcal{S} \setminus \mathcal{T}, \\ v_j^{i+1} = 0, & \boldsymbol{s}_j \in \partial\mathcal{T}, \end{cases} \tag{3.7}$$

where v_j^{i+1} now represents the value on iteration $i + 1$ for discretized state $\boldsymbol{s}_j$ and $I\left(V^i, f(\boldsymbol{s}, \boldsymbol{\pi}, \Delta t)\right)$ represents an interpolation function which relies on the discrete value table V and the discretization time step Δt. In practice, linear interpolation is often used in combination with a discrete integrator, such as Euler or Runge-Kutta 4 [41]. Finally, we must restrict the problem to a bounded set and introduce a corresponding boundary condition

$$v_j^{i+1} = 1 \qquad \text{if } f(\boldsymbol{s}, \boldsymbol{\pi}, \Delta t) \notin \mathcal{S}_{bounded}, \tag{3.8}$$

where $\mathcal{S}_{bounded}$ is often defined as a hypercube. This effectively assigns infinite arrival time to policies which exit the bounded domain.

Guaranteed convergence of this numerical approximation scheme to the underlying value function has been shown by Barles and Souganidis [11, 12, 115]. The minimum time scheme used here was introduced by Bardi and Falcone along with error and convergence rate estimates [8, 9, 43]. The explicit error estimates can be obtained as

$$v_{\text{error}} \leq C\sqrt{\Delta t}\left[1+\left(\frac{\Delta s}{\Delta t}\right)^2\right], \tag{3.9}$$

where C is a positive constant and Δs is the spacing between discrete states.

3.4 Algorithm Implementation Details

While semi-lagrangian dynamic programming is a very general approach for obtaining globally optimal value functions, there are several problem-specific details that need to be taken care of. For example, the cost used for generating the value function can have a huge impact on the behavior of the optimal policy. We highlight some of the details we observed in the following sections.

3.4.1 Cost Functions

We experimented with different costs that were used for generating the value function in the continuous time and discrete-event setting.

Continuous time setting

In the continuous time setting, we experimented with three different instantaneous costs - minimum time, minimum force and minimum positive mechanical power (Eq. 3.10 - 3.12).

$$cost_{time} = 1 - \exp(-\mu_t \cdot t) \tag{3.10}$$

$$cost_{force} = (\mu_F(F_1 + F_2))^2 + (\mu_{\dot{x}}(\dot{x}_{desired} - \dot{x}))^2 \tag{3.11}$$

$$cost_{power} = (\mu_{power}(\max(F_1\dot{l}_1, 0) + \max(F_2\dot{l}_2, 0)))^2 + (\mu_{\dot{x}}(\dot{x}_{desired} - \dot{x}))^2 \tag{3.12}$$

Here μ_t, μ_F, $\mu_{\dot{x}}$ and μ_{power} are the scaling coefficients for each cost. The minimum time cost penalizes the time of the trajectory spent away from the goal. This cost can lead to very aggressive maneuvers, and might not be practical for implementing the generated polices on a real robot.

The minimum force cost penalizes the forces F_1 and F_2 in each leg, while trying to achieve the target velocity $\dot{x}_{desired}$. This leads to less aggressive gaits than the minimum time policy.

Minimum positive mechanical power cost was designed to mimic biological systems, penalizing the total positive work done during the motion, calculated as the product of the leg velocity $\dot{l}$ and leg force. This cost leads to smooth gaits that can be implemented on real systems.

In the case of discrete events, the cost design becomes per-event instead of per time instant. This is then summed up over the discrete events in the trajectory to get the long-term cost. In this case, the cost is more specific to the discrete event being considered. We consider minimizing impulse, average leg force and work (Eq. 3.13 - 3.15).

$$cost_{impulse} = \mu_{impulse} \cdot (\bar{F}_1 t_1 + \bar{F}_2 t_2)^2 + \mu_{\dot{x}} \cdot (\dot{x}_{desired} - \dot{x})^2 \tag{3.13}$$

$$cost_{legforce} = \mu_{impulse} \cdot (\bar{F}_1 + \bar{F}_2)^2 + \mu_{\dot{x}} \cdot (\dot{x}_{desired} - \dot{x})^2 \tag{3.14}$$

$$cost_{work} = \frac{\int \max(F_1 \dot{l}_1, 0)dt + \int \max(F_2 \dot{l}_2, 0)dt}{(mg \cdot d)} \tag{3.15}$$

The minimum impulse cost penalizes the total impulse per step by integrating the average leg forces in each leg $\bar{F}_1$ and $\bar{F}_2$ over the stance durations t_1 and t_2. It also penalizes distance from a target forward velocity $\dot{x}_{desired}$. The minimum leg force cost penalizes the total average leg force and the distance from a desired forward speed. The minimum work done cost minimizes the total positive work done in one step, normalized by the weight mg and forward distance d covered in the step.

A common problem with discrete events based control is that the resulting control from dynamic programming attempts to switch contacts at a high frequency. As a result, our cost in this case tries to maximize the contact time of each leg, while trying to minimize a desired cost. For example, instead of minimizing the impulse cost $cost_{impulse}$, the cost is normalized by the time spent in contact.

$$cost = \frac{cost_{impulse}}{\mu_t t_{contact}} \tag{3.16}$$

3.4.2 Continuous-Time Algorithm

We combine the choices made in previous sections to arrive at a complete value-iteration algorithm for finding optimal legged gaits. The underlying iterative scheme (equation 3.7) requires solving a substantial number of local optimizations over admissible control actions. We perform these optimizations by random sampling of actions, similar to [7], in order to reduce the computational requirements of each iteration. This involves selecting a random spring stiffness for each stance leg and selecting a random landing time for each swing leg. If the selected action results in an improved long term cost, the stored policy and value are updated.

During each iteration, these optimizations are carried out for all possible states and phases, which are represented as discretized grids over a bounded domain. The system dynamics (equation 3.1) are simulated for a short time horizon using Runge-Kutta 4 for numerical integration. The long-term cost is then evaluated using the backup operation in equation 3.7 along with multilinear interpolation of the currently stored values. This process is repeated until the net change change in stored costs falls below a small threshold. Pseudocode for this implementation is described in algorithm 1.

A simple proof-of-concept example is shown in figure 3.4. In this example, an apex-to-apex height-based swing leg policy is reproduced using the dynamic programming method described in algorithm 1. The ground truth is generated using the technique described in [109] for deadbeat running at 5 $\mathrm{m} \cdot \mathrm{s}^{-1}$ on a typical spring mass model. The resulting average absolute error is 0.59 degrees with a standard deviation of 0.38 degrees. Error can be decreased further by reducing the time and space discretization steps.

3.4.3 Discrete Gait Event Algorithm

Instead of a continuous time control, where the control variables can be changed at each time instance, and alternative could be to only change controls at discrete gait events. This is a less general form of the optimization described above, but can lead to physically realizable solutions on real systems.

Algorithm 1 Value Iteration for Bipedal Locomotion

```
1:  for each $s \in AllStates$ do
2:      $cost\_node(s) \leftarrow 1.0$
3:  end for
4:  $cost\_node(target) \leftarrow 0.0$
5:
6:  repeat
7:      for each $s \in AllStates$ do
8:          $a \leftarrow$RandomAction()
9:          $s' \leftarrow$Simulate($s$,$a$,$\Delta t$)
10:         $c \leftarrow 1 - \exp(-\Delta t)(1-\text{Cost}(s'))$
11:         if $c < cost\_node(s)$ then
12:             $cost\_node(s) \leftarrow c$
13:             $action\_node(s) \leftarrow a$
14:         end if
15:     end for
16:
17:     for each $s \in StatesWithSwingLeg$ do
18:         $t \leftarrow$Random$(0, \Delta t)$
19:         $a \leftarrow$Action$(s)$
20:         $s' \leftarrow$SimulateThenPlaceFoot($s$,$a$,$t$)
21:         $c \leftarrow 1 - \exp(-t)(1-\text{Cost}(s'))$
22:         if $c < cost\_node(s)$ then
23:             $cost\_node(s) \leftarrow c$
24:         end if
25:     end for
26: until net change in cost nodes $< epsilon$
```

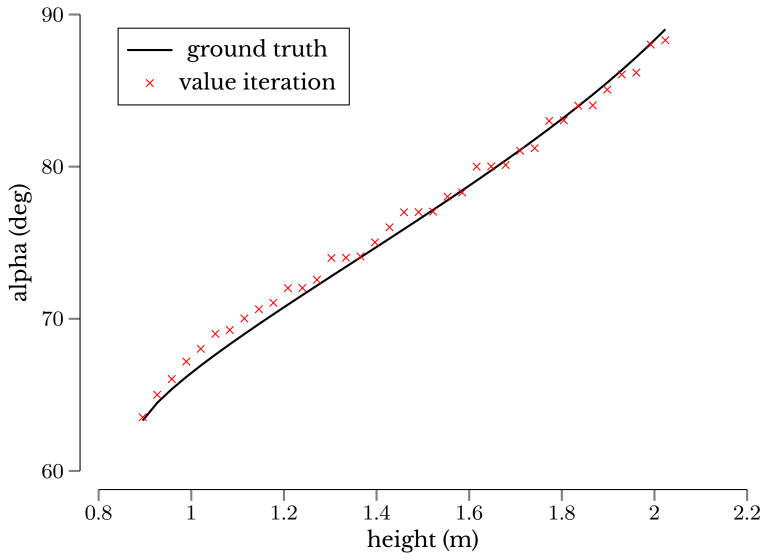


Figure 3.4: Comparison of the value iteration approach to a well-known control law [109] produced using root finding. The policy maps current apex height of the spring mass model to leg landing angles in order to produce running at 5 m $\cdot$ s^{-1} with an apex height of 1 m. The model parameters used are $m = 80$ kg, $l_0 = 1.0$ m, and $k = 20$ kN $\cdot$ m^{-1}.

There can be different gait events that can be used to change the control variables, such as apex, touch down, minimum center of mass height in stance, vertical leg orientation, and minimum or maximum leg compression. The gait events should be chosen in a way that they can seamlessly allow the algorithm to switch between gaits, without affecting the overall optimization process. We choose the start of single support, flight apex, and single support apex as the gait events in our work. Start of single support can occur at the end of flight, as well as end of double support. Flight apex occurs in flight, and single support apex in single support.

3.4.4 Common Pitfalls and Acceleration Methods

In this section, we will describe some of the errors that one might make, and pitfalls one might experience when doing dynamic programming on legged systems. We also describe methods we used for accelerating our computations.

Avoiding Duplicate States

Since dynamic programming depends on visiting every state in every iteration, a very large state space can be prohibitively time consuming to visit. As a result, a large computational overhead can be avoided if one can carefully remove any duplicate states that might occur in the state parametrization.

An example where such a situation occurs in legged locomotion is during double stance. If using foot distances from the center of mass, or leg angles, the same state can represent two different states encodings, or the state encoding can be non-unique. Both the situations are undesirable, and hence the state encoding should be chosen to avoid them. In our work, we use leading leg length and step length as the encoding for state, and introduced bounds to keep it unique.

Interleaving Policy Iteration Loops

Since dynamic programming from scratch can take many iterations, occasional policy evaluations based on the current best policy can sometimes help achieve faster convergence. The frequency of these evaluations is heuristically decided based on the size of the state space. While this heuristic can speed up computations, it does not take away any global convergence guarantees of the overall approach.

Massive Parallelization using GPUs

Recently, Graphics Processing Units (GPUs) have become very popular in reinforcement learning computations. GPUs allow to massively parallelize computations which would take much longer to finish on CPUs on common computers. Since dynamic programming is essentially a parallel computation over multiple states, with a very simple integration step, we can easily use GPUs for this computation. This massively brings down the compute time needed to find the optimal policies for our problem.

Since GPUs come with limited memory, we change the state, actions and rewards to single floating precision to save space and increase speed.

Avoiding Ground Contact Chatter

Ideally, the control decision of entering and leaving contact should be automatically decided based on the relative value functions of the different phases. For example, if the long-term cost of exiting double support and entering single support is less than that of staying in double support, the algorithm should automatically choose to exit double support. However, practically due to a discretized state space and numerical inaccuracies, detecting this change in the sign of the relative values of the different phases can be hard and lead to a chatter in ground contact. Right after the system enters single support from double support, it might choose to re-enter double support, and so on.

Since, such chatter makes the control hard, and infeasible for real systems, we incorporate some constraints into the optimization to reduce the chances of them occurring. First, instead of looking for a zero-crossing in the relative value, we check if the value after the transition is higher than the current value by some threshold. Secondly, we also restrict the search for the next phase transition to be after some minimum specified time, depending on the system and phases concerned.

3.5 Results on Point Mass Models

3.5.1 Continuous-Time Optimal Control Solutions

The approximations of the continuous-time optimal controls computed here use a discrete time step of $\Delta t = 0.010$ s and a zero-crossing tolerance of $t_{\text{tol}} = 10^{-6}$ s for detecting events, such as leg takeoff. The size of the generated discrete lookup table was chosen to balance between solution accuracy and available computational resources. As the table grows in size, memory requirements increase along with the timings for algorithm convergence and policy lookup. The specific bounds and discrete divisions used are shown in table 3.1 and cover a wide range of possible state vectors. In total, all (approximately 10 million) state vectors can be stored in 200 MB of memory when using single precision floating point storage. This allows the code to execute on individual graphic processing units (GPUs), which offer powerful single-instruction-multiple-data (SIMD) processing but often have limited RAM. Although GPUs offer a performance advantage here, this is a problem specific observation and requires benchmarking against CPU implementations. These policies required approximately 10^{11} samples before converging.

Illustrative example trajectories generated using gait transition policies are shown in figure 3.5. Transitions occur very quickly within a few steps in nearly all cases due to the minimum-time objective. Policy rollout requires multilinear interpolation of the stored actions to determine the control at ar-

Table 3.1: State space boundaries for general human-like walking and running gaits.

domain	lower bound	upper bound	units	divisions
α	40	140	$^\circ$	51
l	0.8	1.0	m	21
d	0.25	1.30	m	36
$\dot{x}$	-1.0	4.0	$\text{m} \cdot \text{s}^{-1}$	26
$\dot{z}$	-2.0	2.0	$\text{m} \cdot \text{s}^{-1}$	21
k	10	90	$\frac{mg}{l_0}$	continuous

bitrary states within the bounded domain. Foot placement is determined by searching for the time of value function crossing between swing and stance phases. This requires numerical root finding, which can be computationally intensive and sensitive to contact chattering when the grid spacing is large. The basins of attraction at vertical leg orientation, $\alpha = 90$ degrees, for these policies are shown in figures 3.6 and 3.7. These basins are very large and similar to what can be generated using existing discrete event methods. However, the continuous-time solutions also feature defined controls for all states which lie in between discrete events. This leads to much more complex policies which can generate optimal actions for nearly any given state of the system.

3.5.2 Discrete Gait Event Optimal Control Solutions

Although continuous-time solutions offer powerful control laws and a very large policy space, they require significant computation in order to perform policy rollouts. Even if enough computational resources are available, unrestrained point-mass model trajectories are not necessarily feasible for arbitrary hardware platforms. The biped robot, ATRIAS, used in this work is capable of highly dynamic maneuvers due to its light legs and strong mo-

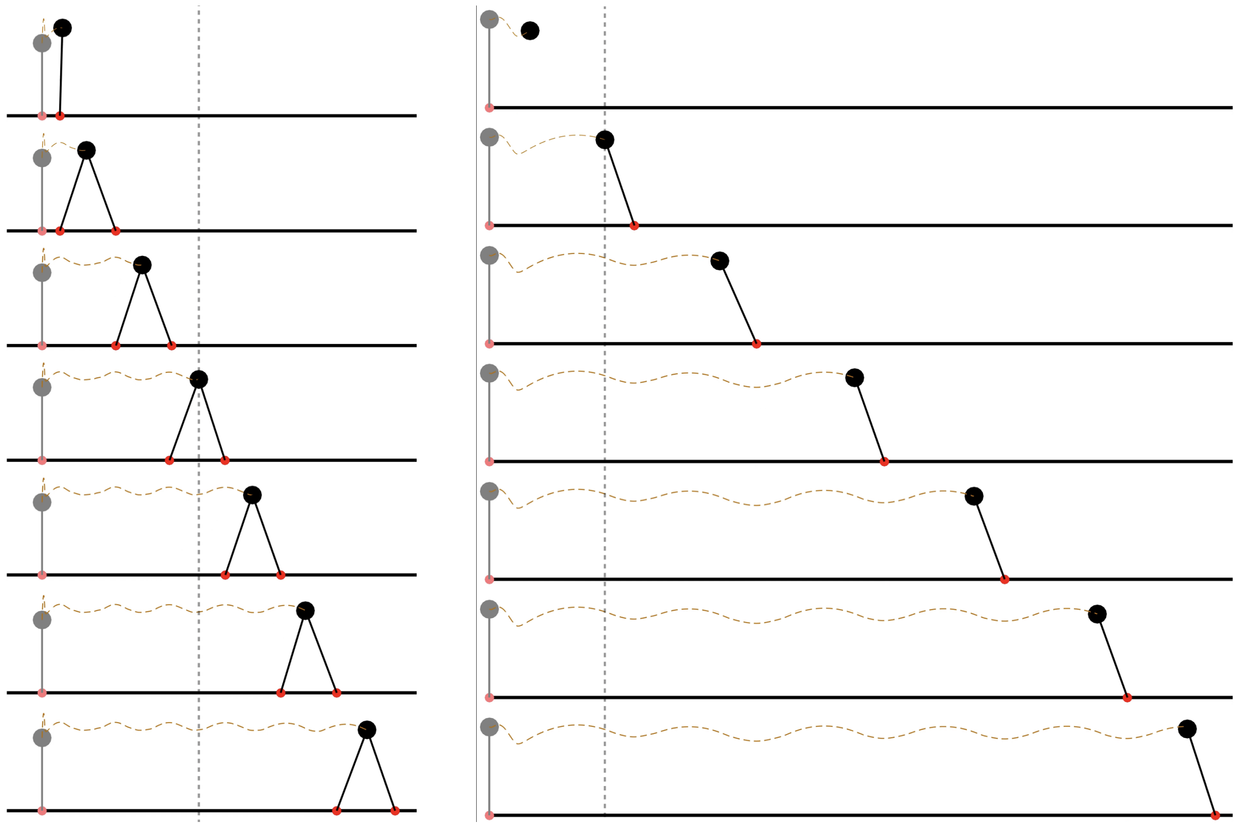


Figure 3.5: Example gait transitions between walking and running generated using continuous-time optimal control. The vertical dashed line indicates when the target state is reached. Left: Run-to-walk transition. Right: Walk-to-run transition.

tors, but has both kinematic and actuation bandwidth limits. This prevents the system from following extreme center of mass motions and necessitates constraining the space of optimal control solutions.

In this section we present a point-mass model Poincaré-style control framework that relies on discrete gait events and can be more readily transferred to the ATRIAS hardware. Similar to many spring mass model controllers in literature, we choose to parameterize the model's leg forces and modulate these parameter values only at specific discrete gait events. We leverage our continuous-time results by using their leg force profiles to inform the discrete control design. This constrains the problem to a much smaller policy

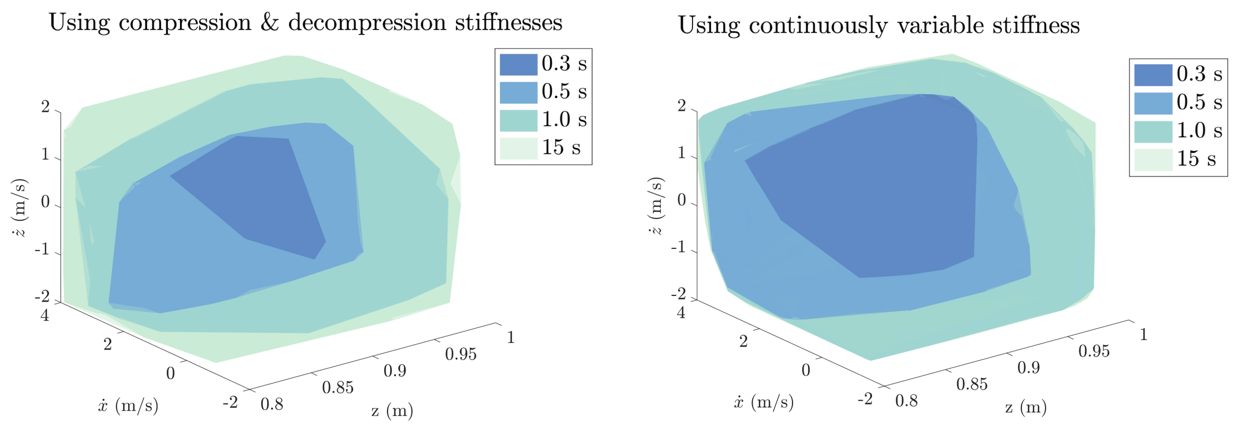


Figure 3.6: Comparison of minimum-time basins of attraction generated using the discrete event scheme described in [113] (left) and the continuous-time approach described here (right). The visualized state space corresponds to vertical leg orientation, $\alpha = 90$ degrees. Colors correspond to the time required to reach a target running state with $\dot{x} = 3.0$ m $\cdot$ s^{-1}, $z = 0.95$ m, $\dot{z} = -1.0$ m $\cdot$ s^{-1}. Although both policies have large basins of attraction, the continuous-time policy is able to achieve a lower cost due to the increased policy space.

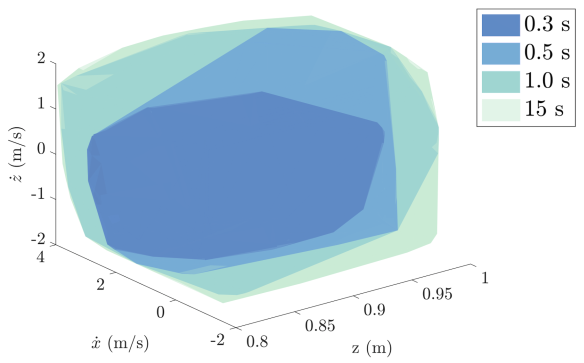


Figure 3.7: Minimum-time basin of attraction generated using the continuous-time approach. The visualized state space corresponds to vertical leg orientation, $\alpha = 90$ degrees. Colors correspond to the time required to reach a target walking state with $\dot{x} = 1.2$ m $\cdot$ s^{-1}, $z = 0.92$ m, $\dot{z} = 0.4$ m $\cdot$ s^{-1}.

space but still produces motions plans close to the continuous-time solutions.

Typical spring mass model controllers that use Poincaré maps, rely on a periodic orbit of state that intersects a single surface corresponding to a specific gait event, such as vertical apex during the flight phase or vertical

Table 3.2: State space boundaries for discrete "ATRIAS-safe" walking and running gaits.

domain	lower bound	upper bound	units
α	76	100	$^\circ$
l	0.94	1.0	m
$\dot{x}$	-0.2	3.2	$\mathrm{m} \cdot \mathrm{s}^{-1}$
$\dot{z}$	-1.2	0.6	$\mathrm{m} \cdot \mathrm{s}^{-1}$
k	20	60	$\frac{mg}{l_0}$

leg orientation during the single support phase. Controllers which target both walking and running gaits sometimes include multiple event maps to increase the diversity of target states. However, this requires a separate entity, such as the user or second optimization, to decide which event surface to target at each step. This manual selection is not ideal, because the initial optimization process is unable to create solutions which move freely through all generated maps. In contrast, our approach finds the optimal control across all event surfaces, which are selected to enable trajectories which span flight, single support, and double support phases.

Our design is based on three state space surfaces shown in figure 3.8 corresponding to the gait events of flight apex (FA), initial single support (SSI), and single support apex (SSA). The model leg forces are computed using a nonlinear spring-damper parameterization, $F = k(l_0 - l) + b(l_0 - l)\dot{l}$, which requires a spring stiffness k and damping coefficient b. We apply the same dynamic programming method (algorithm 1) to find the optimal control for this framework. We also significantly reduce the state space bounds (table 3.2) to enforce the kinematic and actuation limits of the ATRIAS hardware. The resulting control laws are capable of running and walking gaits with similar performance compared to the continuous-time solutions.

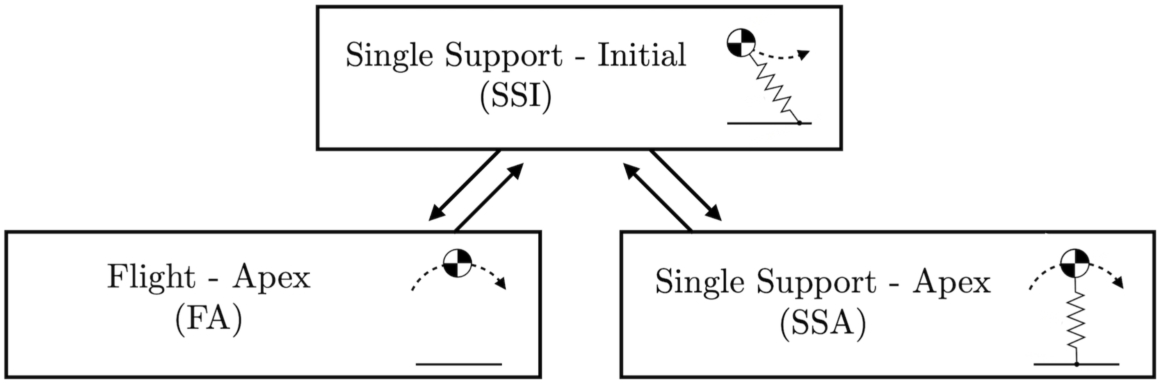


Figure 3.8: The discrete approach looks up a new control whenever the model reaches one of three distinct gait events. The events are selected so that the system can move freely between flight, single support, and double phases. States at FA and SSA are only allowed to transition to SSI, while states at SSI can choose to transition to either FA or SSA.

3.5.3 Comparisons with baselines

We compare our continuous time control approach described in the previous section to a discrete event scheme described in [113]. [113] use four discrete gait events to switch control variables, instead of continuously varying them over the trajectory. These events are apex, vertical leg orientation, maximum virtual leg compression in double support and maximum compression of single support leg. The chosen control variables are the leg stiffness and landing angle. The results of this comparison are shown in Figure 3.6. The original implementation in [113] depends on a user-defined sequence of gait transitions, which limits the applicability of this approach to general gait transitions without human input. We solve the gait transition problem using a dynamic programming approach similar to the one described in the last section, while staying true to the original parametrization of gait events and control variables.

As can be seen in Figure 3.6, while both policies have very large basins of attraction into the desired running state of $\dot{x} = 3.0m/s, z = 0.95m, \dot{z} = -1.0m/s$, the costs incurred while following the baseline are much higher than those from a continuous approach. This highlights the advantage of using a continuously varying control mechanism, which allows for more flexible controllers, and hence solutions that have a lower cost.

Chapter 4

Parametric Insights into Optimal Locomotion Policies

Numerically calculating optimal controls using dynamic programming methods generates large grids of control action values spanning state space. Although the resulting policies can be directly implemented, it is difficult to visualize their behavior due to high problem dimensionality. This makes it particularly challenging to understand and hypothesize about the fundamental processes which govern them. Determining the primary components of optimal policies is important for directing future investigations and discovering which features of state have significant impact on the control. We expect the policies investigated in this work to rely on essential features of bipedal locomotion, because they are generated using a highly general point-mass model with relatively few parameters. However, it is important to note that the cost functions used here result in specific long-term policy goals which may differ from those of biological systems. Nevertheless, inspecting these policies is worthwhile as they are capable of generating stable walking and running gaits for bipedal robots.

4.1 Fitting Parametric Linear Models

In an attempt to get an intuitive explanation for the optimal policies generated by dynamic programming, we attempted to fit parametric linear models to the corresponding value functions. These models let us hypothesize about the underlying essential components that together determine the optimal control actions. While we do not expect to fit one general model to all gaits, we expect correlations between specific state vector features and the long-term costs described by the computed value functions. Ultimately, this boils down into a numerical regression problem which attempts to fit a high-dimensional hyperplane to a precomputed set of data points. While many tools are appropriate for performing this calculation, here we focus on linear ordinary least-squares regression to minimize the sum of squared residuals in the fit model.

Ordinary least squares is a widely used method for solving unknown parameters in a linear model. The expressiveness and accuracy of a linear model generated this way is dependent on providing relevant basis functions that describe the data set. This is useful when domain knowledge is available, but a potential problem when useful basis functions cannot be found. In these cases, more powerful function approximators, such as neural networks or Gaussian processes, may be required to compute accurate models. We opt for the less complex linear least-squares approach here as a way to evaluate the accuracy of models built with different features of system state.

Given a set of basis functions $\boldsymbol{b} = [b_1, b_2, ..., b_n]$ evaluated at some state $\boldsymbol{s}$, we fit the computed value function data to a model of the form

$$\hat{v}_s = w_1 b_1 + w_2 b_2 + ... + w_n b_b + \epsilon_s, \tag{4.1}$$

where $\boldsymbol{w} = [w_1, w_2, ..., w_n]$ are linear weights of the model and ϵ_s is the error for target data point v_s^*. We minimize the sum of squared residuals, $\sum_i (\hat{v}_i - v_i^*)^2$, by solving the normal equations. This is often expressed in matrix form as $\boldsymbol{w} = (\boldsymbol{B}^T \boldsymbol{B})^{-1} \boldsymbol{B}^T \boldsymbol{v}^*$, where now $\boldsymbol{B}$ contains the set of chosen basis functions evaluated at all data points and $\boldsymbol{v}^*$ is a vector containing the corresponding target values. While this relationship holds mathematically, numerical issues can arise while performing the matrix inversion on a computer. In many cases the normal equations can be very close to singular; this commonly happens when two different basis functions provide similar fits to the data. Attempting to numerically invert a matrix that is close to singular often provide erroneous results due to roundoff errors. We avoid these errors by using singular value decomposition (SVD) of the basis function matrix. This allows us to write the matrix decomposition

$$\boldsymbol{B} = UWV^T \tag{4.2}$$

and solve the normal equations with

$$\boldsymbol{w} = VW^+ U^T \boldsymbol{v}^*, \tag{4.3}$$

where $^+$ indicates the psuedoinverse. This will produce a solution that is the best least-squares fit even in the presence of a nearly singular matrix.

We choose to fit a different model to each phase-specific value function in order to take advantage of the different state vectors for flight, single support, and double support phases. Since the number of data points computed for each phase differs significantly (flight: 54,366, single support: 1,141,686, double support: 16,923,816), we limit the number of data points used in the regression to 50,000 by randomly selecting points. Furthermore, we remove outliers in the dataset using robust covariance estimation via the Orthogonalized Gnanadesikan-Kettenring (OGK) method. This prevents us from using points with very high costs that are often outside the reachable set; on average this corresponds to between 20 and 30 percent of a data set. After applying these filters we attempt to find the models with the highest coefficient of determination,

$$R^2 = 1 - \frac{\sum_i (v_i^* - \hat{v}_i)^2}{\sum_i (v_i^* - \text{mean}(v^*))^2}, \tag{4.4}$$

for a fixed complexity defined as the total number of utilized basis function. We approach this task by binary testing each possible combination of candidate basis functions. This requires searching through $2^n - 1$ combinations for n nominated basis functions.

4.1.1 Choice of basis functions

The basis functions chosen for our experiments were based on physical plausibility. For example, system energy and direction of the velocity vector are pertinent quantities when controlling a running gait. We used insights like these to design basis functions describing point mass bipedal systems. These fall in to two categories, those which only involve the center of mass variables and those which involve the foot point locations. Flight phase regression uses center of mass functions, single support phase regression is augmented with functions involving one foot position, and double support regression includes a symmetric set of functions for the second foot position.

Table 4.1: The set of basis functions investigated for fitting linear models.

Description	Symbol
Center of mass basis functions	
Vertical Height	z
Horizontal velocity	$\dot{x}$
Vertical velocity	$\dot{z}$
Vertical height squared	z^2
Horizontal velocity squared	$\dot{x}^2$
Vertical velocity squared	$\dot{z}^2$
Vertical height × horizontal velocity	$z \cdot \dot{x}$
Vertical height × vertical velocity	$z \cdot \dot{z}$
Horizontal velocity × vertical velocity	$\dot{x} \cdot \dot{z}$
Total velocity magnitude	$\sqrt{\dot{x}^2 + \dot{z}^2}$
Velocity angle	$\arctan(\dot{z}/\dot{x})$
Landing angle	$\arcsin(z/l_0)$
Leg basis functions	
Leg length	l
Leg velocity	$\dot{l}$
Horizontal velocity × leg length	$\dot{x}l$
Vertical velocity × leg length	$\dot{z}l$
Vertical height × horizontal position of foot	$z \cdot x_{foot}$
Horizontal velocity × horizontal position of foot	$\dot{x} \cdot x_{foot}$
Vertical velocity × horizontal position of foot	$\dot{z} \cdot x_{foot}$
Leg angle	$\arctan(z/x_{foot})$
Leg length × $\arccos(1 - z/l)$	$l \cdot \arccos(1 - z/l)$
Leg length squared	l^2

4.2 Linear value function models

Our investigation focuses on two nominal gait controllers targeting walking at 1.2 m/s and running at 3.0 m/s. We examine three different cost functions spanning minimum time, minimum force, and minimum positive mechanical power.

4.2.1 Results

The following tables display the maximum coefficients of determination that were found using different numbers of basis functions. The basis functions which generate these scores are marked with non-zero weights, while those not included are marked as zero. In general, as the complexity of a model increases (number of basis functions increases), the model becomes more and more accurate at predicting the optimal policy, as illustrated by the R^2 coefficients. Tables 4.2, 4.4, 4.6 display results for running at 3.0 m/s while minimizing time, leg force, and positive mechanical power. Tables 4.3, 4.5, 4.7 display parallel results for walking at 1.2 m/s.

4.3 Discussion

We can analyze the importance of different basis functions to each policy objective by examining which functions maximize the coefficient of determination, R^2, when model complexity is low. As the number of model parameters is increased, R^2 increases in general. However, there are diminishing returns beyond a point, depending on the specific policy and gait phase. Adding more basis functions stops improving the accuracy of the fit.

Table 4.2: Minimum time - Running at 3.0 m/s.
Basis function weight combinations with the highest coefficients of determination sorted by increasing model complexity.

Flight

R^2	1	z	$\dot{x}$	$\dot{z}$	z^2	$\dot{x}^2$	$\dot{z}^2$	$z\dot{x}$	$z\dot{z}$	$\dot{x}\dot{z}$	$\sqrt{\dot{x}^2+\dot{z}^2}$	$\tan^{-1}(\dot{z}/\dot{x})$	$\sin^{-1}(z/l_0)$
0.42	0.27	0	0	0	-0.13	0	0	0	0	0	0	0	0
0.65	0.39	-0.23	0	0	0	-3.2e-3	0	0	0	0	0	0	0
0.68	0.38	-0.23	0	0	0	-3.2e-3	0	0	6.5e-3	0	0	0	0
0.76	0.41	-0.25	0	-0.082	0	-3.3e-3	0	0	0.096	0	0	0	0
0.77	0.42	-0.26	0	-0.084	0	0	5.9e-3	0	0.099	0	-0.016	0	0
0.78	0.42	-0.26	0	-0.081	0	0	5.9e-3	0	0.099	-2.0e-3	-0.016	0	0
0.78	0.65	-0.8	0	-0.089	0.31	0	6.2e-3	0	0.11	-2.0e-3	-0.016	0	0
0.78	0.65	-0.8	0	-0.09	0.31	0	6.2e-3	0	0.11	-1.6e-3	-0.016	1.4e-3	0
0.79	0.6	-0.71	0.011	-0.086	0.27	0	6.1e-3	-0.013	0.1	-2.1e-3	-0.016	0	0
0.79	0.6	-0.71	0.011	-0.088	0.27	0	6.2e-3	-0.013	0.1	-1.7e-3	-0.016	1.4e-3	0
0.79	0.59	-0.7	0.012	-0.088	0.27	-5.6e-4	5.5e-3	-0.013	0.1	-1.7e-3	-0.013	1.5e-3	0
0.79	0.57	-0.64	0.012	-0.087	0.22	-5.7e-4	5.5e-3	-0.013	0.1	-1.7e-3	-0.013	1.5e-3	7.7e-3

Single Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.65	0.19	0	0	0	0	0	0	0	-0.015	0	0
0.67	0.15	0	0	0	0	0	0	-0.016	0.026	0	0
0.69	0.27	0	0	-0.016	0	0	0	0.026	-0.094	0	0
0.7	0.28	0	0	-0.016	0.025	0	0	-0.17	0.11	0	0
0.7	0.28	0	0	0	-0.016	0	0	0.011	0.028	-0.16	0.11
0.71	0.26	0	3.9e-3	-0.016	0.011	0	0.033	-0.16	0.11	0	0
0.72	0.21	0	3.9e-3	0	-0.016	0	0.053	0.012	0.067	-0.16	0.11
0.72	0.53	0	-0.72	3.8e-3	-0.016	0	0.054	0.013	0.068	-0.16	0.51
0.72	0.21	0.022	-0.016	-0.018	0.063	0	-0.022	0.012	0.071	-0.17	0.12
0.72	0.5	-0.64	0.022	-0.016	-0.018	0.064	-0.021	0.013	0.072	-0.17	0.47

Double Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.71	0.19	0	-0.016	0	0	0	0	0	0	0	0
0.72	0.31	0	-0.13	0	0	0	0	0	0	0	-0.015
0.73	0.38	-0.23	-0.015	0	0	0	0	0	0	-0.078	0
0.74	0.31	0	-0.12	0.015	0	-0.012	0	-0.029	0	0	0
0.74	0.34	0	-0.18	0.012	0	-8.1e-3	0	-0.025	0	0	-0.042
0.74	0.35	0	-0.18	0.014	0	-0.011	0	-0.027	-0.034	0	-0.052
0.75	0.33	0	0.1	-0.25	0	0.015	-0.011	-0.028	-0.058	0	-0.072
0.75	0.33	0	0.11	-0.26	0	0.02	-6.5e-3	-0.028	-9.9e-3	-0.069	-0.079
0.75	0.33	0.12	-0.27	0.022	0	-4.0e-3	-0.038	9.9e-3	-0.014	-0.073	-0.083
0.75	0.33	0.12	-0.27	0.022	-4.5e-3	-0.037	9.1e-3	2.9e-3	-0.016	-0.073	-0.083

Table 4.3: Minimum time - Walking at 1.2 m/s.
Basis function weight combinations with the highest coefficients of determination sorted by increasing model complexity.

Flight

R^2	1	z	$\dot{x}$	$\dot{z}$	z^2	$\dot{x}^2$	$\dot{z}^2$	$z\dot{x}$	$z\dot{z}$	$\dot{x}\dot{z}$	$\sqrt{\dot{x}^2+\dot{z}^2}$	$\tan^{-1}(\dot{z}/\dot{x})$	$\sin^{-1}(z/l_0)$
0.24	0.15	0	0	0	0	0	0	0	0.013	0	0	0	0
0.42	0.14	0	0	0	0	2.3e-3	0	0	0.013	0	0	0	0
0.5	0.21	-0.077	0	0	0	2.1e-3	0	0	0.012	0	0	0	0
0.54	0.21	-0.078	-8.7e-3	0	0	4.6e-3	0	0	0.012	0	0	0	0
0.57	0.27	-0.14	-0.039	0	0	4.7e-3	0	0.034	0.012	0	0	0	0
0.6	0.29	-0.16	-0.042	-0.041	0	4.7e-3	0	0.037	0.057	0	0	0	0
0.62	0.28	-0.16	-0.042	-0.046	0	4.8e-3	3.3e-3	0.036	0.063	0	0	0	0
0.63	0.55	-0.77	-0.046	-0.055	0.35	4.8e-3	3.5e-3	0.041	0.072	0	0	0	0
0.64	0.59	-0.84	-0.046	-0.057	0.39	7.4e-3	8.0e-3	0.044	0.074	0	-0.016	0	0
0.65	0.59	-0.83	-0.045	-0.057	0.38	7.3e-3	8.0e-3	0.043	0.074	0	-0.016	9.0e-4	0
0.65	0.6	-0.85	-0.046	-0.059	0.39	7.3e-3	8.0e-3	0.044	0.074	7.1e-4	-0.016	1.8e-3	0
0.65	0.55	-0.75	-0.046	-0.059	0.32	7.3e-3	8.0e-3	0.044	0.074	7.1e-4	-0.016	1.8e-3	0.013

Single Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l\cdot cos^{-1}(1-z/l)$	l^2
0.55	0.16	0	0	0	0	0	0	0	8.7e-3	0	0
0.57	0.13	0	0	0	0	0	8.7e-3	0	0.018	0	0
0.59	0.13	0	0	0	0	0	4.7e-3	8.7e-3	0.017	0	0
0.6	0.18	0	0	7.1e-3	0	0	4.6e-3	0.027	-0.05	0	0
0.61	0.18	0	0	0	0	0	7.1e-3	4.6e-3	9.1e-3	0.029	-0.051
0.61	0.18	8.2e-3	0	4.7e-3	0	0	-4.7e-3	8.8e-3	0.024	-0.048	0
0.62	0.2	-0.036	0	8.3e-3	0	0	4.7e-3	-5.0e-3	8.9e-3	0.024	-0.035
0.62	0.57	-0.86	0	8.2e-3	4.7e-3	0	-5.0e-3	9.0e-3	0.024	-0.036	0.46
0.62	0.56	-0.89	8.2e-3	4.7e-3	0.014	0	-5.0e-3	9.3e-3	0.033	-0.035	0.47
0.62	0.57	-0.89	7.6e-3	4.7e-3	6.5e-4	0.013	-4.3e-3	9.3e-3	0.033	-0.035	0.47

Double Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l\cdot cos^{-1}(1-z/l)$	l^2
0.59	0.33	0	-0.19	0	0	0	0	0	0	0	0
0.65	0.34	0	-0.2	0	0	0	0	0	0	7.1e-3	0
0.65	0.4	0	-0.24	0	0	9.5e-3	0	0	0	-0.082	0
0.66	0.4	-0.25	7.0e-3	0	0	8.6e-3	0	0	0	-0.06	0
0.66	0.4	-0.26	-7.3e-3	0	0	0.01	0	0.015	0	-0.044	0
0.67	0.43	-0.3	-9.3e-3	0	0	0.011	0	0.018	0	-0.047	-0.027
0.67	0.42	0.084	-0.37	9.1e-3	0	-0.029	0	0.04	0	-0.079	-0.056
0.67	0.43	0.085	-0.37	0	0.011	-0.03	0.042	-1.6e-3	0	-0.079	-0.059
0.67	0.43	0.084	-0.37	0	0.011	-0.03	0.042	-0.027	0.025	-0.08	-0.06
0.67	0.43	0.086	-0.37	5.2e-4	0.011	-0.031	0.043	-0.027	0.025	-0.081	-0.06

Table 4.4: Minimum leg force - Running at 3.0 m/s.
Basis function weight combinations with the highest coefficients of determination sorted by increasing model complexity.

Flight

R^2	1	z	$\dot{x}$	$\dot{z}$	z^2	$\dot{x}^2$	$\dot{z}^2$	$z\dot{x}$	$z\dot{z}$	$\dot{x}\dot{z}$	$\sqrt{\dot{x}^2+\dot{z}^2}$	$\tan^{-1}(\dot{z}/\dot{x})$	$\sin^{-1}(z/l_0)$
0.61	2.7e+3	0	0	0	0	0	0	-811.0	0	0	0	0	0
0.75	9.5e+3	-7.4e+3	-733.0	0	0	0	0	0	0	0	0	0	0
0.76	9.5e+3	-7.4e+3	-999.0	0	0	87.0	0	0	0	0	0	0	0
0.78	9.3e+3	-7.3e+3	-722.0	0	0	0	0	0	0	-78.0	0	188.0	0
0.79	9.4e+3	-7.3e+3	-988.0	0	0	84.0	0	0	0	-76.0	0	188.0	0
0.81	9.9e+3	-7.3e+3	-899.0	0	0	155.0	0	0	0	-73.0	-455.0	199.0	0
0.81	2.1e+4	-3.3e+4	-899.0	0	1.5e+4	155.0	0	0	0	-73.0	-466.0	188.0	0
0.82	2.3e+4	-3.6e+4	-1.6e+3	0	1.6e+4	155.0	0	744.0	0	-71.0	-466.0	177.0	0
0.82	2.3e+4	-3.6e+4	-1.6e+3	0	1.5e+4	166.0	0	733.0	111.0	-99.0	-477.0	122.0	0
0.82	2.5e+4	-4.0e+4	-1.6e+3	-633.0	1.8e+4	155.0	0	800.0	777.0	-92.0	-455.0	133.0	0
0.82	2.3e+4	-3.5e+4	-1.6e+3	-622.0	1.4e+4	155.0	0	800.0	766.0	-92.0	-455.0	133.0	488.0
0.82	2.3e+4	-3.5e+4	-1.6e+3	-611.0	1.4e+4	144.0	-22.0	800.0	744.0	-92.0	-399.0	133.0	499.0

Single Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.73	2.5e+3	0	0	0	0	0	0	0	-755.0	0	0
0.76	944.0	0	0	0	0	0	-799.0	0	1.1e+3	0	0
0.79	811.0	0	0	0	0	0	-799.0	0	433.0	0	1.2e+3
0.8	3.3e+3	0	0	0	0	0	-811.0	1.1e+3	-4.7e+3	0	4.7e+3
0.82	2.8e+3	0	-833.0	0	0	400.0	0	1.7e+3	-5.0e+3	0	4.9e+3
0.82	2.8e+3	0	-833.0	0	0	400.0	0	422.0	1.8e+3	-5.1e+3	4.8e+3
0.82	-533.0	-844.0	3.4e+3	0	0	400.0	0	499.0	4.0e+3	-5.2e+3	4.8e+3
0.82	-644.0	366.0	-833.0	-400.0	0	3.6e+3	0	500.0	4.2e+3	-5.4e+3	5.0e+3
0.82	7.9e+3	-1.9e+4	366.0	-833.0	-400.0	3.7e+3	0	500.0	4.2e+3	-5.4e+3	1.6e+4
0.82	8.2e+3	-2.0e+4	233.0	-833.0	-277.0	3.6e+3	155.0	500.0	4.2e+3	-5.4e+3	1.6e+4

Double Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.79	2.4e+3	-733.0	0	0	0	0	0	0	0	0	0
0.81	1.8e+3	-733.0	0	0	0	0	0	0	0	1.7e+3	0
0.82	-1.3e+3	3.5e+3	0	0	0	-733.0	0	0	0	-1.3e+3	0
0.82	1.5e+3	5.4e+3	-5.3e+3	0	0	0	0	0	0	-744.0	-2.4e+3
0.83	1.6e+3	7.6e+3	-7.4e+3	0	0	0	-744.0	0	0	-1.3e+3	-3.3e+3
0.83	1.6e+3	8.6e+3	-8.4e+3	0	0	-1.3e+3	599.0	0	0	-1.3e+3	-3.4e+3
0.83	1.6e+3	8.7e+3	-8.4e+3	58.0	0	-1.4e+3	655.0	0	0	-1.5e+3	-3.3e+3
0.84	1.4e+3	9.3e+3	-8.6e+3	0	255.0	-1.6e+3	744.0	0	-200.0	-2.0e+3	-3.3e+3
0.84	1.4e+3	9.2e+3	-8.5e+3	222.0	-1.6e+3	722.0	-477.0	0	300.0	-1.9e+3	-3.3e+3
0.84	1.5e+3	9.4e+3	-8.9e+3	288.0	89.0	-1.7e+3	899.0	-577.0	255.0	-2.0e+3	-3.5e+3

Table 4.5: Minimum leg force - Walking at 1.2 m/s.
Basis function weight combinations with the highest coefficients of determination sorted by increasing model complexity.

Flight

R^2	1	z	$\dot{x}$	$\dot{z}$	z^2	$\dot{x}^2$	$\dot{z}^2$	$z\dot{x}$	$z\dot{z}$	$\dot{x}\dot{z}$	$\sqrt{\dot{x}^2+\dot{z}^2}$	$\tan^{-1}(\dot{z}/\dot{x})$	$\sin^{-1}(z/l_0)$
0.23	455.0	0	0	0	0	36.0	0	0	0	0	0	0	0
0.53	544.0	0	-333.0	0	0	133.0	0	0	0	0	0	0	0
0.56	844.0	0	-333.0	0	0	133.0	0	0	0	0	0	0	-255.0
0.63	2.3e+3	-1.9e+3	-1.0e+3	0	0	144.0	0	733.0	0	0	0	0	0
0.63	2.3e+3	-1.9e+3	-1.0e+3	0	0	144.0	0	733.0	0	0	0	28.0	0
0.64	2.4e+3	-1.9e+3	-1.0e+3	0	0	155.0	0	744.0	0	0	-57.0	28.0	0
0.64	2.3e+3	-1.9e+3	-1.0e+3	-57.0	0	144.0	0	755.0	0	24.0	0	58.0	0
0.65	2.4e+3	-2.0e+3	-1.0e+3	-54.0	0	155.0	0	766.0	0	24.0	-56.0	56.0	0
0.65	2.4e+3	-2.0e+3	-1.0e+3	-200.0	0	155.0	0	766.0	166.0	24.0	-51.0	56.0	0
0.65	2.7e+3	-2.5e+3	-1.1e+3	-233.0	0	155.0	0	788.0	199.0	24.0	-50.0	56.0	199.0
0.65	2.7e+3	-2.6e+3	-1.0e+3	-244.0	0	155.0	13.0	788.0	200.0	25.0	-84.0	56.0	200.0
0.65	2.7e+3	-2.6e+3	-1.0e+3	-244.0	24.0	155.0	13.0	788.0	200.0	25.0	-85.0	56.0	200.0

Single Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.58	500.0	0	0	0	0	0	0	0	88.0	0	0
0.63	333.0	0	0	0	0	0	83.0	0	122.0	0	0
0.64	311.0	0	0	84.0	0	0	77.0	0	133.0	0	0
0.65	188.0	0	0	80.0	0	0	64.0	79.0	222.0	0	0
0.66	411.0	0	80.0	70.0	0	81.0	0	244.0	-199.0	0	0
0.67	433.0	0	65.0	80.0	0	-72.0	0	82.0	244.0	0	-211.0
0.67	133.0	0	67.0	78.0	0	-73.0	0	311.0	89.0	444.0	-222.0
0.67	122.0	98.0	79.0	-100.0	0	322.0	0	-36.0	89.0	455.0	-222.0
0.67	3.5e+3	-7.5e+3	66.0	79.0	0	-73.0	333.0	90.0	466.0	-222.0	4.2e+3
0.67	3.4e+3	-7.4e+3	94.0	79.0	-100.0	344.0	-32.0	90.0	466.0	-233.0	4.1e+3

Double Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.65	522.0	0	85.0	0	0	0	0	0	0	0	0
0.66	1.9e+3	0	-1.5e+3	0	0	85.0	0	0	0	0	0
0.66	1.9e+3	0	-1.6e+3	0	-30.0	99.0	0	0	0	0	0
0.67	2.0e+3	0	-1.6e+3	0	-78.0	61.0	0	0	0	0	155.0
0.68	2.5e+3	0	-2.3e+3	0	-122.0	100.0	0	0	199.0	0	-422.0
0.68	2.5e+3	-2.3e+3	233.0	0	-222.0	411.0	0	0	0	-244.0	-511.0
0.69	2.4e+3	-2.3e+3	-87.0	0	166.0	211.0	0	0	-333.0	244.0	-533.0
0.69	2.4e+3	811.0	-3.0e+3	0	233.0	-277.0	477.0	0	-244.0	-344.0	-777.0
0.69	2.4e+3	799.0	-3.0e+3	233.0	-277.0	466.0	-411.0	0	177.0	-344.0	-777.0
0.69	2.4e+3	788.0	-3.0e+3	-4.1	233.0	-266.0	466.0	-400.0	177.0	-333.0	-777.0

Table 4.6: Minimum positive mechanical power - Running at 3.0 m/s. Basis function weight combinations with the highest coefficients of determination sorted by increasing model complexity.

Flight

R^2	1	z	$\dot{x}$	$\dot{z}$	z^2	$\dot{x}^2$	$\dot{z}^2$	$z\dot{x}$	$z\dot{z}$	$\dot{x}\dot{z}$	$\sqrt{\dot{x}^2+\dot{z}^2}$	$\tan^{-1}(\dot{z}/\dot{x})$	$\sin^{-1}(z/l_0)$
0.71	4.2e+4	0	0	0	0	0	0	-1.1e+4	0	0	0	0	0
0.86	1.3e+5	-9.8e+4	-9.6e+3	0	0	0	0	0	0	0	0	0	0
0.9	1.3e+5	-1.0e+5	-9.7e+3	-2.9e+3	0	0	0	0	0	0	0	0	0
0.92	1.4e+5	-1.1e+5	-9.7e+3	-3.3e+4	0	0	0	0	3.3e+4	0	0	0	0
0.94	1.5e+5	-1.1e+5	-7.2e+3	-3.2e+4	0	0	0	0	3.2e+4	0	-4.0e+3	0	0
0.94	1.7e+5	-1.7e+5	-7.3e+3	-3.4e+4	0	0	0	0	3.4e+4	0	-3.9e+3	0	2.2e+4
0.95	3.3e+5	-5.1e+5	-1.8e+4	-3.8e+4	2.1e+5	0	0	1.2e+4	3.8e+4	0	-4.0e+3	0	0
0.95	3.4e+5	-5.3e+5	-1.7e+4	-3.9e+4	2.3e+5	0	1.2e+3	1.2e+4	3.9e+4	0	-5.8e+3	0	0
0.95	3.5e+5	-5.5e+5	-1.7e+4	-3.9e+4	2.4e+5	933.0	2.3e+3	1.2e+4	3.9e+4	0	-1.0e+4	0	0
0.96	3.5e+5	-5.4e+5	-1.7e+4	-3.8e+4	2.4e+5	922.0	2.3e+3	1.2e+4	3.9e+4	-455.0	-1.0e+4	0	0
0.96	3.1e+5	-4.6e+5	-1.7e+4	-3.8e+4	1.8e+5	911.0	2.3e+3	1.2e+4	3.9e+4	-455.0	-1.0e+4	0	8.6e+3
0.96	3.1e+5	-4.6e+5	-1.7e+4	-3.9e+4	1.8e+5	911.0	2.3e+3	1.1e+4	3.9e+4	-366.0	-1.0e+4	300.0	8.6e+3

Single Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.85	5.3e+4	0	0	0	-1.0e+4	0	0	0	0	0	0
0.87	2.0e+5	0	0	0	-1.0e+4	0	0	0	0	-1.1e+5	0
0.89	2.0e+5	0	0	0	-1.0e+4	0	0	0	0	-2.3e+3	-1.1e+5
0.9	1.0e+6	0	-1.9e+6	0	0	0	-1.0e+4	0	0	-1.3e+5	1.0e+6
0.91	2.1e+5	0	2.9e+4	0	0	0	-1.0e+4	0	-3.1e+4	-2.9e+4	-1.2e+5
0.91	2.1e+5	0	3.0e+4	0	0	0	-1.0e+4	-3.3e+4	-3.1e+4	-1.4e+5	3.1e+4
0.92	1.0e+6	0	-1.8e+6	0	3.0e+4	0	-1.0e+4	-3.2e+4	-3.1e+4	-1.4e+5	1.0e+6
0.92	1.1e+6	0	-1.9e+6	3.0e+4	-1.0e+4	0	-3.2e+4	-8.4e+3	-2.9e+4	-1.4e+5	1.1e+6
0.92	1.1e+6	-1.9e+6	3.0e+4	-1.0e+4	-3.3e+4	0	-2.9e+4	3.8e+3	6.2e+3	-1.4e+5	1.1e+6
0.92	1.1e+6	-1.9e+6	3.0e+4	-1.0e+4	-3.3e+4	-1.1e+3	-2.9e+4	3.7e+3	5.5e+3	-1.4e+5	1.1e+6

Double Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.73	6.6e+4	-1.1e+4	0	0	0	0	0	0	0	0	0
0.82	3.0e+5	-2.5e+5	-1.0e+4	0	0	0	0	0	0	0	0
0.83	5.0e+5	-2.6e+5	-2.2e+5	0	0	0	-9.8e+3	0	0	0	0
0.84	5.0e+5	-3.2e+5	-1.8e+5	0	0	0	-9.8e+3	6.0e+4	0	0	0
0.84	4.7e+5	-2.3e+5	-2.1e+5	0	9.0e+3	0	-9.5e+3	-1.9e+4	0	0	0
0.84	4.7e+5	-2.1e+5	-2.4e+5	0	1.8e+4	-5.3e+4	3.5e+4	-1.8e+4	0	0	0
0.85	4.8e+5	-1.9e+5	-2.8e+5	0	1.4e+4	-4.9e+4	3.2e+4	-1.4e+4	0	0	-3.0e+4
0.85	4.9e+5	-1.8e+5	-3.0e+5	0	1.7e+4	7.9e+3	-6.0e+4	4.5e+4	-2.4e+4	0	-4.3e+4
0.85	4.9e+5	-1.8e+5	-3.0e+5	0	1.9e+4	8.8e+3	-6.2e+4	4.7e+4	-1.4e+4	-1.3e+4	-4.3e+4
0.85	4.9e+5	-1.7e+5	-3.0e+5	1.9e+4	9.3e+3	-6.3e+4	4.9e+4	-1.4e+4	-1.3e+4	-5.9e+3	-4.6e+4

Table 4.7: Minimum positive mechanical power - Walking at 1.2 m/s. Basis function weight combinations with the highest coefficients of determination sorted by increasing model complexity.

Flight

R^2	1	z	$\dot{x}$	$\dot{z}$	z^2	$\dot{x}^2$	$\dot{z}^2$	$z\dot{x}$	$z\dot{z}$	$\dot{x}\dot{z}$	$\sqrt{\dot{x}^2+\dot{z}^2}$	$\tan^{-1}(\dot{z}/\dot{x})$	$\sin^{-1}(z/l_0)$
0.2	9.5e+3	0	0	0	-7.4e+3	0	0	0	0	0	0	0	0
0.28	1.0e+4	0	0	-611.0	-8.2e+3	0	0	0	0	0	0	0	0
0.38	1.0e+4	0	-1.7e+3	0	-7.7e+3	455.0	0	0	0	0	0	0	0
0.51	2.9e+4	-2.7e+4	-8.8e+3	0	0	522.0	0	7.6e+3	0	0	0	0	0
0.6	3.1e+4	-2.9e+4	-9.0e+3	-655.0	0	500.0	0	7.8e+3	0	0	0	0	0
0.68	3.6e+4	-3.4e+4	-9.8e+3	-8.1e+3	0	499.0	0	8.7e+3	8.2e+3	0	0	0	0
0.71	8.1e+4	-1.4e+5	-1.1e+4	-9.6e+3	5.7e+4	500.0	0	9.8e+3	9.8e+3	0	0	0	0
0.74	3.9e+4	-3.5e+4	-9.6e+3	-8.5e+3	0	966.0	888.0	9.1e+3	8.6e+3	0	-3.0e+3	0	0
0.77	9.4e+4	-1.6e+5	-1.1e+4	-1.0e+4	6.8e+4	1.0e+3	999.0	1.0e+4	1.1e+4	0	-3.2e+3	0	0
0.79	1.0e+5	-1.7e+5	-1.2e+4	-1.2e+4	7.4e+4	1.0e+3	1.0e+3	1.1e+4	1.2e+4	244.0	-3.2e+3	0	0
0.8	1.0e+5	-1.7e+5	-1.2e+4	-1.2e+4	7.5e+4	1.0e+3	1.0e+3	1.1e+4	1.2e+4	300.0	-3.2e+3	199.0	0
0.8	8.9e+4	-1.4e+5	-1.2e+4	-1.2e+4	5.5e+4	1.0e+3	1.0e+3	1.1e+4	1.2e+4	300.0	-3.2e+3	188.0	2.9e+3

Single Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.67	1.3e+5	0	0	0	0	0	0	0	0	-1.4e+5	0
0.75	9.9e+5	-2.0e+6	0	0	0	0	0	0	0	0	1.0e+6
0.77	1.0e+6	-2.0e+6	-3.4e+4	0	0	0	0	0	0	0	1.1e+6
0.81	1.0e+6	0	-2.0e+6	0	0	0	1.7e+3	0	0	-3.7e+4	1.1e+6
0.83	1.0e+6	0	-2.0e+6	1.6e+3	0	0	2.3e+3	0	0	-3.7e+4	1.1e+6
0.84	1.0e+6	0	-2.0e+6	1.5e+4	0	0	-1.5e+4	-1.5e+4	0	-4.1e+4	1.1e+6
0.84	1.0e+6	0	-2.0e+6	1.5e+4	0	-1.5e+4	-1.5e+4	2.1e+3	0	-4.2e+4	1.1e+6
0.85	1.0e+6	-2.0e+6	1.5e+4	-1.5e+4	0	0	-2.7e+3	-1.4e+4	2.2e+3	-4.2e+4	1.1e+6
0.85	1.0e+6	-2.0e+6	1.5e+4	-255.0	-1.5e+4	-2.9e+3	-1.4e+4	2.2e+3	0	-4.2e+4	1.1e+6
0.85	1.0e+6	-2.0e+6	1.5e+4	-222.0	-1.4e+4	-8.3e+3	-1.4e+4	2.1e+3	-3.8e+3	-4.2e+4	1.1e+6

Double Support

R^2	1	l	$\dot{l}$	$\dot{x}l$	$\dot{z}l$	zx_{foot}	$\dot{x}x_{foot}$	$\dot{z}x_{foot}$	$tan^{-1}(z/x_{foot})$	$l \cdot cos^{-1}(1-z/l)$	l^2
0.5	2.3e+5	-2.2e+5	0	0	0	0	0	0	0	0	0
0.75	4.3e+5	-2.4e+5	-2.0e+5	0	0	0	0	0	0	0	0
0.77	4.3e+5	-2.5e+5	-2.0e+5	0	0	1.2e+3	0	0	0	0	0
0.77	4.3e+5	-2.1e+5	-2.3e+5	0	0	-1.9e+4	0	2.0e+4	0	0	0
0.77	4.3e+5	-2.2e+5	-2.3e+5	0	1.8e+3	-2.2e+4	2.2e+4	0	0	0	0
0.78	4.3e+5	-2.1e+5	-2.3e+5	0	3.1e+3	-2.4e+4	2.3e+4	-1.5e+3	0	0	0
0.78	4.3e+5	-2.2e+5	-2.3e+5	0	6.3e+3	4.9e+3	-3.2e+4	3.2e+4	0	0	-9.1e+3
0.78	4.3e+5	-2.1e+5	-2.4e+5	0	6.1e+3	6.2e+3	-3.3e+4	3.3e+4	0	-1.0e+4	-8.6e+3
0.78	4.3e+5	-2.1e+5	-2.4e+5	6.6e+3	6.5e+3	-3.4e+4	3.4e+4	-1.1e+4	0	-3.2e+3	-1.0e+4
0.78	4.3e+5	-2.0e+5	-2.4e+5	6.8e+3	6.6e+3	-3.4e+4	3.4e+4	-9.4e+3	-1.8e+3	-3.8e+3	-1.1e+4

Amongst the basis functions that we considered in our study, some play a more important role than others, depending on the gait phase. Next, we describe the relative importance of different basis functions to specific policies.

4.3.1 Minimum time

Running

For the flight phase of a running target, the minimum time control policy is strongly correlated with the center of mass vertical height and the product of vertical height and vertical velocity. The square of the horizontal velocity plays an important role in models of 4 or less basis functions, but gets replaced by the square of the vertical velocity beyond models of complexity greater than 4.

In the single support phase, the minimum time control policy is dominated by the leg angle basis function and the product of vertical speed and horizontal foot position. The product of vertical velocity and leg length plays a role in models of complexity greater than 4.

In the double support phase, the leg velocity is an important feature for all models. On the other hand, there is no clear correlation between the control and other features for model complexity less than 4. For complexity 4 and greater, the product of leg length and horizontal velocity, as well as the product of vertical velocity and horizontal foot position become important features. These results can be seen in Table 4.2.

Walking

For the minimum time walking target, the basis functions behave very differently than the running case, which is expected as the control policies are significantly different. In flight, both the product of vertical height and vertical velocity, as well as squared horizontal velocity are correlated with the control policies. These are neatly followed by vertical height and then horizontal velocity. The remaining basis functions do not play a large role here.

For the single support phase, the leg angle is dominant, similar to the running target. The next most active features include the product of horizontal velocity and horizontal foot position, and the product of vertical velocity and horizontal foot position.

In double support, the leg velocity is highly correlated with the control for all model complexities. A second feature that plays a important role is the product of the leg length and the curve described by $\arccos(1-z/l)$, which is related to a cycloid. These results can be seen in Table 4.3.

4.3.2 Minimum leg force

Running

Minimum leg force running shows slightly different trends than the minimum time case. The flight phase shows a strong correlation with vertical height and horizontal velocity. Squared horizontal velocity also shows some significance. The single support phase again is highly reliant on leg angle, but does not show strong correlation with the other basis functions. Double support results show dependence on leg length and the cycloidal curve. These results can be seen in Table 4.4.

Walking

The walking solution flight phase features the horizontal velocity and its squared quantity prominently in the models. Vertical height and its product with horizontal foot position also play a large role in the models. The single support phase shows less clear relationships, aside from leg angle which is included in all model. Double support models have a clear relationship with leg length, the product of vertical height and horizontal foot position, and the product of vertical velocity and leg length. These results can be seen in Table 4.5.

4.3.3 Minimum positive mechanical power

Running

Minimizing the positive mechanical power of the system yields a different set of parametric models. The R^2 scores for this running target cost function are also noticeably higher, indicating better linear fits. During the flight phase, the models are dominated by vertical height, horizontal velocity, and vertical velocity. Single support models only show a strong trend with the cycloidal term, but double support shows a clearer ordering of terms. Leg length has the highest frequency, followed by leg velocity, and then the product of horizontal velocity and horizontal foot position. These results can be seen in Table 4.6.

Walking

The walking results show similar trends. The flight phase models appear to weakly favor horizontal velocity, vertical height, and squared horizontal velocity. The single support models do not show as many consistent trends, but do frequently use the squared leg length. Finally, the double support phase results show more order. Leg length and leg velocity have the strongest relationships, as was the case with the running target. The product of vertical velocity and horizontal foot position follow these quantities in importance.

Our analysis shows that parts of the gait cycle for specific targets can be explained by different gait features. This gives a way of approximating high-dimensional policies from dynamic programming in a concise, parametric manner. This can further be used to reason about the nature of policies being generated, and provide insights about them.

Chapter 5

Deadbeat Spring Mass Model Running on ATRIAS

The simple spring mass model (SMM) describes a point mass rebounding on massless spring legs. Research on this model has led to deadbeat foot placement strategies that produce highly robust SMM running in the presence of large and frequent, unexpected gait disturbances [25, 101, 111, 130]. This theoretical performance goes far beyond what has been demonstrated in running robots [57, 59, 93, 116]. However, these robots are clearly more complex systems than the conceptual SMM. They possess more degrees of freedom leading to additional dynamics, are limited by actuator saturation, and experience sensory noise that produces uncertainty about the actual state of the system. As a result, the utility of SMM theories for the control of complex running robots remains largely unclear.

Addressing this gap in understanding, several researchers have investigated the foot placement strategies of the SMM on more simplified hopping robots. For an early example, Zeglin [133] investigated state space planning algorithms based on the SMM for a bow-legged hopper with a com-

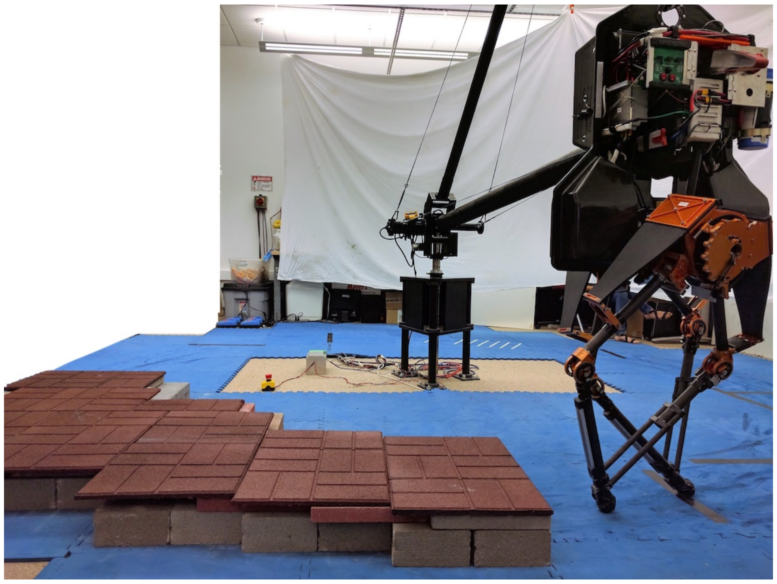


Figure 5.1: CMU's ATRIAS biped shown in the boom testbed during the ground disturbance experiments with unexpected height changes of ±15 cm (discussed in section 5.3.3).

pressible spring and passively stabilized trunk. More recently, Shemer and Degani [114] investigated deadbeat hopping policies for a similar monopod robot with a gyroscopically stabilized trunk in a low gravity environment. They used an analytical approximation of the SMM to compare the effect of constant deadbeat impact angles to swing leg retraction policies. Finally, Uyanık and colleagues [123] quantified the predictive performance of analytical approximations of the SMM in achieving deadbeat behavior using a monopedal spring leg with no attached trunk. All these studies have in common that they were performed with small and specialized one-legged platforms, characterized by prismatic legs, passively stabilized trunk motion in stance, and external sensor measurements. In contrast, we are interested in understanding if the SMM leg placement theories can be transferred to more humanoid robots and attempt the transfer on ATRIAS, a bipedal machine of human scale and weight with an actively controlled trunk and without external sensing (Fig. 5.1).

For the transfer, we focus on rendering the best possible behavior match between the SMM and ATRIAS. To achieve this goal, we use a model-based force control approach during the stance phase of running. Controllers of this type have been implemented successfully on legged robots for tracking desired forces during locomotion [48, 64, 67]. Combined with tracking the deadbeat foot placements of the SMM in flight, ATRIAS should match the behavior and robustness observed in the simplified model. However, we expect deviations from this ideal behavior due to the real world challenges faced by the machine. We perform planar running experiments to quantify these deviations, and thus, the utility of the SMM theories for more complex robots.

The remainder of this section is organized in four parts. We first provide a general overview of our control approach in section 5.1 before detailing its implementation on ATRIAS in section 5.2. We then present the results of the running experiments in section 5.3, which show that the resulting controller achieves velocity tracking that is consistent with deadbeat behavior of the underlying model for velocity changes of $\pm 0.2\,\mathrm{m \cdot s^{-1}}$. For larger velocity changes and ground height disturbances up to ±15 cm, the controller performance degrades, although the robot maintains running. We discuss the reasons for this degradation and highlight in section 5.4 directions to improve on these initial results about SMM-based control for generating robust and versatile behavior in running robots.

5.1 Control Approach

The SMM consists of a point mass m rebounding on a massless spring leg of stiffness k and rest length l_0. This system behaves as a purely ballistic projectile during flight and as a spring-loaded inverted pendulum during stance with

$$\begin{aligned} m\ddot{x} &= k\left[l_0\,(x^2+z^2)^{-1/2}-1\right]x, \\ m\ddot{z} &= k\left[l_0\,(x^2+z^2)^{-1/2}-1\right]z-mg, \end{aligned} \tag{5.1}$$

where (x, z) are the coordinates of the point mass in the horizontal and vertical dimensions. The model does not consider sliding during stance. Stance occurs when the foot point strikes the ground and flight resumes once the leg length reaches l_0 during rebound. The model's trajectory in flight is fully determined by the horizontal speed $\dot{x}$ and the system energy E_s, which is a constant parameter of the model. Given the speed in one flight phase, the model behavior in the ensuing stance and flight phases is controlled by the leg angle α_{TD} at touchdown [111]. This influence of the landing angle on the model behavior can be captured with the apex return map, $\dot{x}_{i+1} = f(\dot{x}_i, \alpha_{\mathrm{TD},i})$, which relates the state of the model between the apexes of two subsequent flight phases (i and $i+1$). Inverting this function yields a deadbeat touchdown angle that takes the system from the current forward velocity $\dot{x}_i$ to a desired forward velocity $\dot{x}_{i+1} = \dot{x}_a^*$ in a single step,

$$\alpha^*_{\mathrm{TD},i} = f^{-1}(\dot{x}_i, \dot{x}_a^*). \tag{5.2}$$

Deadbeat controllers based on this theory have been identified that provide robustness to unobserved rough terrain for the SMM in simulation [25, 111, 130].

Our target platform for translating this theory is CMU's ATRIAS (Fig. 5.1), one of three identical copies of a human-sized bipedal robot developed by the Dynamic Robotics Laboratory at Oregon State University [58]. The centroidal dynamics of ATRIAS has inertial properties similar to that of human locomotion. The robot weighs about 64 kg with its mass concentrated in the trunk, 0.19 m above the pelvis. The trunk's rotational inertia is about $2.2\,\mathrm{kg}\cdot\mathrm{m}^2$. Each leg of this bipedal robot is constructed from four lightweight carbon fiber segments. The proximal segments are driven in the sagittal plane by series elastic actuators (SEA) composed of a fiberglass leaf spring and a geared electric DC motor. The reflected inertia of these hip-anchored motors is about $3.75\,\mathrm{kg}\cdot\mathrm{m}^2$ after gearing. With a power supply stabilized by a 0.12 F electrolytic capacitor, these motors can draw peak currents of 165 A each, which translates into peak torques of about $600\,\mathrm{N}\cdot\mathrm{m}$ per actuator at the joint level. In addition, frontal plane abduction and adduction of each hip is provided by a third DC motor mounted on the trunk. Although ATRIAS is capable of untethered 3-D locomotion, this paper focuses on planar control theory of the SMM; hence, the trunk is attached to a boom but is free to pitch in the sagittal plane. The boom constrains the robot to move in a sphere and has negligible mass and inertia. The boom pivot point moves freely with the robot, and thus does not transmit any significant forces in the sagittal plane that could stabilize the trunk's pitch.

5.1.1 Implicit regulation of system energy

Two points complicate the transfer of control theories developed for the SMM onto legged robots such as ATRIAS. The first point is that the system energy is constant in the model but will change in a robot due to the desire to accelerate and brake as well as internal friction. One way of changing energy in the SMM is to introduce another control input, such as a variable leg stiffness during stance [131]. However, we adopt a different approach. We approximate the SMM dynamics (5.1) around the vertical pose $(x, z) = (0, z_0^*)$

with $z_0^* < l_0$ by

$$m\ddot{x} = k\,(z_0^* - z)\,\frac{x}{z}, \tag{5.3}$$

$$m\ddot{z} = k\,(z_0^* - z) - mg. \tag{5.4}$$

This approximate SMM is similar to the one used in [80]; it has decoupled vertical dynamics, which enables independent control of apex height and horizontal speed achieved during flight, implicitly regulating system energy with more natural gait variables. Specifically, we use (5.4) to prescribe a desired vertical motion $z^*(t)$ for ATRIAS with apex height z_a^* and landing and takeoff height z_0^* (Fig. 5.2). Given this reference, we compute the corresponding return map of the horizontal motion from (5.3). Thus, the updated deadbeat control law for leg placement in flight becomes

$$\alpha^*_{\mathrm{TD},i} = f^{-1}_{\mathrm{approx}}(\dot{x}_i, \dot{x}_a^*, z^*(t)), \tag{5.5}$$

which regulates running speed on ATRIAS.

Besides implicit regulation of system energy, the approximation of the SMM with (5.3) and (5.4) allows us to easily generalize this model from a point mass to a rigid body, which we address in the next section.

5.1.2 Explicit stabilization of trunk orientation

A second point complicating the transfer of SMM theories on to bipedal robots is that they require stabilization of trunk orientation, which is ignored in the SMM. This is a common problem in humanoid walking control based on the linear inverted pendulum model. It is often solved using a nonlinear quadratic program for a full order dynamics model of the robot [48, 64, 67].

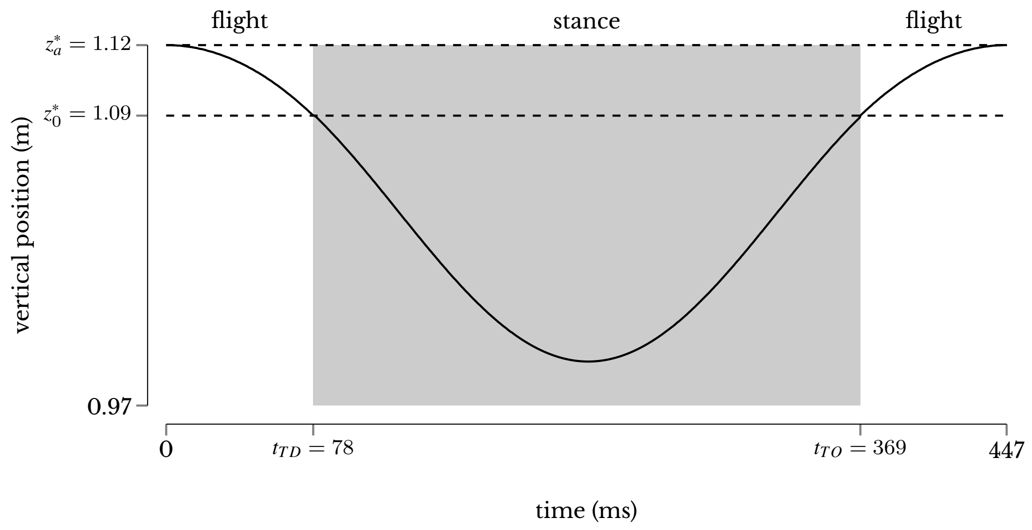


Figure 5.2: Prescribed vertical motion $z^*(t)$ between two apexes. The motion is derived from (5.4) assuming $m = 64\,\mathrm{kg}$ and $k = 16\,\mathrm{kN} \cdot \mathrm{m}^{-1}$ (details on the choice of k provided in Sec. 5.3.1). The motion is used by the ATRIAS controller as a target behavior and re-initiated in every flight phase.

This optimization balances different goals, such as the center of mass (CoM) behavior, trunk orientation, and other constraints on the robot motion. Due to computational costs, the optimization typically applies to only the current time step without taking advantage of future dynamics. In contrast to this approach, we introduce an intermediate model of reduced order (Fig. 5.3b) that allows us to consider the future dynamics of a floating rigid body with orientation θ, inertia I, and dynamics

$$\begin{aligned} m\ddot{x} &= F_x, \\ m\ddot{z} &= F_z - mg, \\ I\ddot{\theta} &= -zF_x + xF_z, \end{aligned} \tag{5.6}$$

using finite-horizon linear quadratic regulation (LQR), as detailed in section 5.2.2. Here F_x and F_z are the ground reaction forces of the approximate SMM model modified by a stabilizing control for the trunk orientation (detailed in section 5.2.2 below, equation 5.9). We assume that the centralized inertia I on ATRIAS is constant, rather than configuration-dependent, because the robot's legs are light relative to its body.

5.1.3 Overview of control flow

Given the approximate spring mass model, the intermediate complexity model distributes translational and rotational motion. However, a third layer of model complexity is required to translate this centroidal motion into robot control. Overall this leads to a three layer control structure.

Figure 5.3 summarizes the flow of this control structure for the transfer of SMM control theory onto the ATRIAS biped. At the highest level, we define a spring mass gait based on desired speed and desired apex height. The corresponding approximate SMM provides the desired CoM trajectory in stance and the desired deadbeat angle in flight (Fig. 5.3a). In stance, the intermediate implementation level then generates GRFs that optimally trade off the desired CoM behavior against the desired trunk orientation (Fig. 5.3b). These GRFs are mapped in the next level by a dynamics model of the ATRIAS robot (detailed in [131]),

$$\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{h} = \boldsymbol{S}\,\boldsymbol{\tau} + \boldsymbol{J}^T\boldsymbol{F}, \tag{5.7}$$

to the required joint torques (Fig. 5.3c), which are finally converted into desired motor velocities for the torque control of ATRIAS's SEAs (Fig. 5.3d).

In flight, the deadbeat angle from the approximate SMM is used to generate a foot point trajectory for the leg that achieves the target angle at a designated touchdown time (Fig. 5.2). This foot trajectory is converted into joint trajectories using a kinematics model of ATRIAS (Fig. 5.3c). The joints

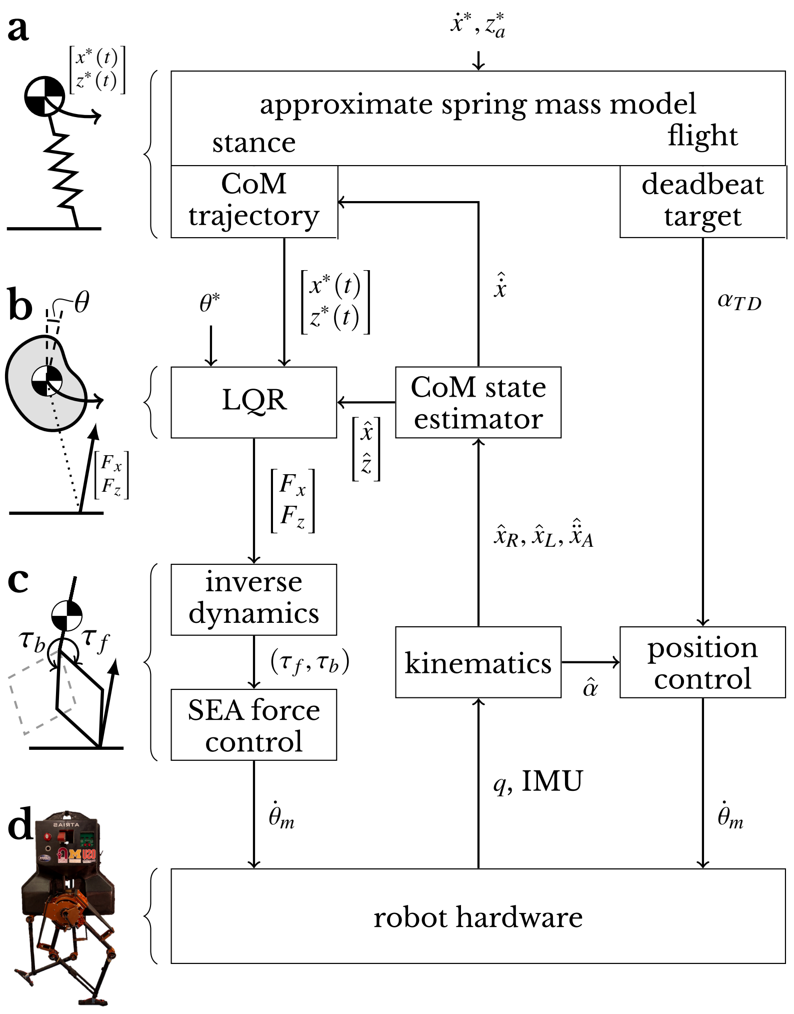


Figure 5.3: Overview of implemented control. See section 5.1.3 for details.

are then position-controlled by sending a velocity command to the robot's SEAs (Fig. 5.3d).

5.2 Implementation

The control implementation on the ATRIAS biped requires more consideration. Besides detailing the individual layers of the control flow presented in the last section, this section explains how we address state estimation, external disturbances, and model inaccuracies. All described control is implemented onboard ATRIAS using MATLAB Simulink Realtime software with an update rate of 1 kHz.

5.2.1 Estimation of CoM and contact states

Tracking the CoM of ATRIAS and knowledge about its ground contact state are prerequisites for implementing the SMM control. For the first, we use two independent but identically structured Kalman filters estimating the horizontal and vertical CoM states. In both filters, the underlying model is a point mass m influenced by an applied force F. For instance, the resulting discrete time process equation of the filter for the horizontal states is

$$\begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \\ \ddot{x}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \\ \ddot{x}_t \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m} \end{bmatrix} (\hat{F}^x_t - \hat{F}^x_{t-1}) + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m} \end{bmatrix} w_t,$$

where $\Delta t = 1\,\text{ms}$ is the time step and w is Gaussian white process noise with covariance $Q = 25\,\text{N}^2$. The force $\hat{F}^x$ is estimated from the measured torques of the hip SEAs and the commanded torques of the lateral motors (ATRIAS has no torque sensing for its lateral motors). This is accomplished by solving for $\boldsymbol{F}$ in equation 5.7 with the constraint $\boldsymbol{J}\ddot{\boldsymbol{q}} = -\dot{\boldsymbol{J}}\dot{\boldsymbol{q}}$, which assumes a static point of contact, yielding the dynamics,

$$\hat{\boldsymbol{F}} = f_{\text{dyn}}(\hat{\boldsymbol{\tau}}, \boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{5.8}$$

The measurement equation of the filter is

$$\begin{bmatrix} \hat{x}_t^R \\ \hat{x}_t^L \\ \hat{\ddot{x}}_t^A \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \\ \ddot{x}_t \end{bmatrix} + \mathbf{v_t},$$

where $\hat{x}^{L/R}$ are estimates of the horizontal distances from the left and right feet to the CoM, respectively, $\hat{\ddot{x}}^A$ is an estimate of the horizontal acceleration of the CoM, and $\mathbf{v}$ is measurement noise. The distances are computed from the kinematics model of ATRIAS using the robot's measured joint angles and trunk orientation. The acceleration is calculated using acceleration measurements from an IMU attached to ATRIAS's trunk. The horizontal filter is initialized using these measurements on every touchdown to account for a changing foot point. The vertical filter is only initialized once on the first touchdown. The covariance matrix

$$R_t = \frac{1}{\Delta t} \begin{bmatrix} R_{mx} - \mu_R(R_{mx} - R_{mn}) & 0 & 0 \\ 0 & R_{mx} - \mu_L(R_{mx} - R_{mn}) & 0 \\ 0 & 0 & R_A \end{bmatrix}$$

of the measurement noise is adaptive. Specifically, the covariance for the distance measurement noise is inversely proportional to the estimated load on each leg in units of body weight, $\mu_{R/L} = 10\,\hat{F}^z_{R/L}/(mg)$ ($\mu_{R/L}$ clamped to $[0,1]$, $R_{mn} = 5 \times 10^{-5}\,\mathrm{m}^2$, $R_{mx} = 1\,\mathrm{m}^2$, and $R_A = 4\,\mathrm{m}^2/\mathrm{s}^4$).

The contact state of each leg is determined from the estimated vertical GRF, $\hat{F}_z$. ATRIAS has no explicit contact or force sensing at its feet; instead, the force estimate (5.8) is used to determine if a leg is in stance. An $\hat{F}_z$ exceeding 50% of body weight triggers the touchdown event and causes the control to enter the stance phase. Conversely, once the vertical CoM velocity $\hat{\dot{z}}$ crosses from negative to positive values, indicating rebound, a drop in $\hat{F}_z$ below the 50% threshold triggers take off and the exit from stance control. This threshold level creates a small delay of approximately 15 ms (about 5% of stance duration) in contact detection.

5.2.2 Stance control

Upon transition of ATRIAS into the stance phase, the approximate SMM layer of the control (Fig. 5.3a) generates a desired CoM trajectory $[x^*(t),\, z^*(t)]$ based on the previous takeoff velocity $\hat{\dot{x}}_i$, the next desired takeoff velocity $\dot{x}^*_{i+1}$, the prescribed vertical motion $z^*(t)$, and equation (5.3) (Sec. 5.1.1). Note, although $x^*(t)$ describes the horizontal motion in stance, it is chosen along with the foot placement based on the system state at the previous takeoff. The layer also generates a corresponding force input $\mathbf{u}^*(t) = [F^*_x(t),\, F^*_z(t)]$ from the GRFs of the approximate SMM with $F^*_z(t) = k(z^*_0 - z^*(t))$ and $F^*_x(t) = F^*_z(t)x^*(t)/z^*(t)$.

In the second control layer (Fig. 5.3b), the force input is modified to account for trunk stabilization. The desired CoM trajectory is combined with a desired trunk orientation $\theta^*(t) = 0$ into a reference motion

$$\boldsymbol{\xi}^*(t) \;=\; [x^*(t),\, \dot{x}^*(t),\, \theta^*(t),\, \dot{\theta}^*(t),\, z^*(t),\, \dot{z}^*(t)]$$

for a floating rigid body (Sec. 5.1.2). We convert the floating rigid body dynamics (equation 5.6) to state space form and linearize the error dynamics around this reference trajectory, which yields,

$$\Delta\dot{\boldsymbol{\xi}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{F_z^*(t)}{I} & 0 & 0 & 0 & -\frac{F_x^*(t)}{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Delta\boldsymbol{\xi} + \begin{bmatrix} 0 & 0 \\ 1/m & 0 \\ 0 & 0 \\ \frac{-z^*(t)}{I} & \frac{x^*(t)}{I} \\ 0 & 0 \\ 0 & 1/m \end{bmatrix} \Delta\boldsymbol{u},$$

where $\Delta\boldsymbol{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}^*$ and $\Delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}^*$. We approximate $F_x^*(t) = 0$ and $x^*(t) = 0$, which decouples the vertical state error dynamics. As this approximate model is linear, finite horizon LQR yields the optimal force control input $\mathbf{u}(t) = \mathbf{u}^*(t) - \mathbf{K}(t)(\boldsymbol{\xi}(t) - \boldsymbol{\xi}^*(t))$ for trading off tracking the SMM behavior against balancing the trunk, with the feedback gain $\mathbf{K}(t) = [\mathbf{K}_x(t), \mathbf{K}_z]$. Thus, the second control layer generates the modified desired GRF $\mathbf{u}(t) = [F_x(t), F_z(t)]$ with components

$$F_x = F_x^* - \mathbf{K}_x \begin{bmatrix} x - x^* \\ \dot{x} - \dot{x}^* \\ \theta - \theta^* \\ \dot{\theta} - \dot{\theta}^* \end{bmatrix}, \quad F_z = F_z^* - \mathbf{K}_z \begin{bmatrix} z - z^* \\ \dot{z} - \dot{z}^* \end{bmatrix}. \tag{5.9}$$

The third control layer (Fig. 5.3c) converts the desired GRFs (5.9) into motor commands of the robot's SEAs in three steps. First, the forces are passed through a safety check to ensure that the foot does not slip on the ground. This requires the vertical force to remain positive, $F_z \geq 0$, and the horizontal force F_x to be inside a friction cone with stiction coefficient of 0.5. It is important to note that these two constraints are almost never active on ATRIAS because stance begins when F_z exceeds 50% of body weight (Sec. 5.2.1) and the stance leg is typically near the vertical. Second, the control compensates for the vertical constraint forces caused by the boom (Fig. 5.1), $F_z^\dagger = F_z \hat{F}_z/(m\hat{\ddot{z}})$, where $\hat{\ddot{z}}$ is provided by the CoM state estimator (Sec. 5.2.1). Third, the modified desired forces, $F_x^\dagger$ and $F_z^\dagger$, are then mapped based on the dynamics model of ATRIAS (5.7) into joint torques using

$$\zeta = \begin{bmatrix} \mathbf{M}_{5\times 5} & -\mathbf{S}_{5\times 2} \\ \mathbf{J}_{2\times 5} & \mathbf{0}_{2\times 2} \end{bmatrix}^{-1} \left(\begin{bmatrix} \mathbf{h}_{5\times 1} \\ -(\dot{\mathbf{J}}\dot{\mathbf{q}})_{2\times 1} \end{bmatrix} + \begin{bmatrix} \mathbf{J}^T_{5\times 2} \\ \mathbf{0}_{2\times 2} \end{bmatrix} \begin{bmatrix} F_x^\dagger \\ F_z^\dagger \end{bmatrix} \right)$$

with $\zeta = [\ddot{x} \;\; \ddot{z} \;\; \ddot{\theta} \;\; \ddot{l} \;\; \ddot{\gamma}_l \;\; \tau_f \;\; \tau_b]^T$ specifying accelerations as well as joint torques, and $\ddot{l}$ and $\ddot{\gamma}_l$ being the leg length and angle accelerations, respectively (detailed in [131]). Although the solution vector ζ contains both accelerations and torques, the accelerations are not used by the controller. Furthermore, because the swing leg is light, we do not account for its accelerations and control it independently as described in the next section. Finally, the resulting joint torques, τ_f and τ_b, are tracked on ATRIAS using the velocity-based SEA controller described in [102].

5.2.3 Flight control

Once ATRIAS transitions out of stance, the SMM prescribes only a desired landing angle $\alpha^*_{\mathrm{TD},i}$ (5.5). The model does not specify which of the robot's two legs is to land or how it shall reach the target. We solve the first problem by introducing a transitory control phase and the second by defining a kinematic trajectory for the foot point.

Figure 5.4 summarizes the state machine of the ATRIAS controller. While one leg follows the SMM stance and flight behaviors, the other leg remains in a mirror control phase. For instance, when the right leg takes off (RTO),

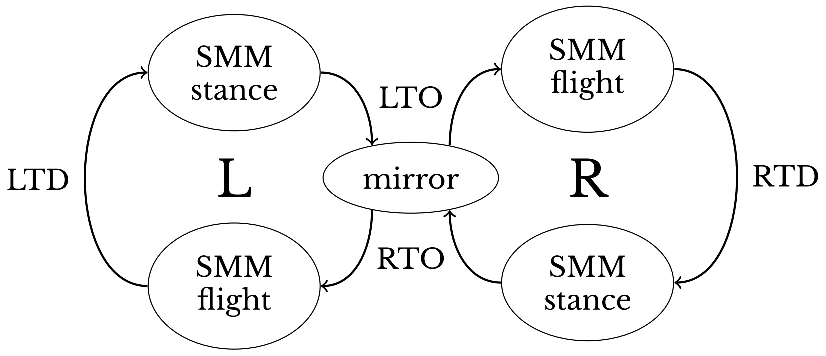


Figure 5.4: State-machine of ATRIAS biped control. While the left (L) or right (R) leg cycles through an SMM flight and stance phase, the other leg remains in the transitory mirror phase. Both legs switch roles when take off occurs. TD/TO: touchdown/takeoff.

it enters this mirror phase and the left leg simultaneously transitions from it into the flight control phase. The right leg remains in the mirror phase while the left leg cycles through an entire step until it takes off (LTO), at which point both legs switch roles.

When a leg is in the mirror control phase, its motion reflects that of the other leg. Raibert [93] introduced this concept of symmetric leg motion, which reduces trunk pitching and prepares the leg for landing. Specifically on ATRIAS, we reflect the foot-to-CoM reference $[x^*(t), z^*(t)]$ from the simplified model (Sec. 5.2.2) across the line $x = 0$. We then negate this reflection to transform it into a CoM-to-foot trajectory,

$$\boldsymbol{r}_{\text{mirror}}(t) = [\, x^*(t)\, ,\, -\, (z^*(t)\, -\, \rho)\,]\, ,$$

where $\rho = 0.2\,\text{m}$ ensures leg retraction for ground clearance.

When a leg switches into the flight control phase at takeoff time t_{TO}, a new CoM-to-foot trajectory is engaged,

$$\boldsymbol{r}_{\text{flight}}(t) = \left[x_{\text{flight}}(t)\, ,\, z_{\text{flight}}(t)\right],$$

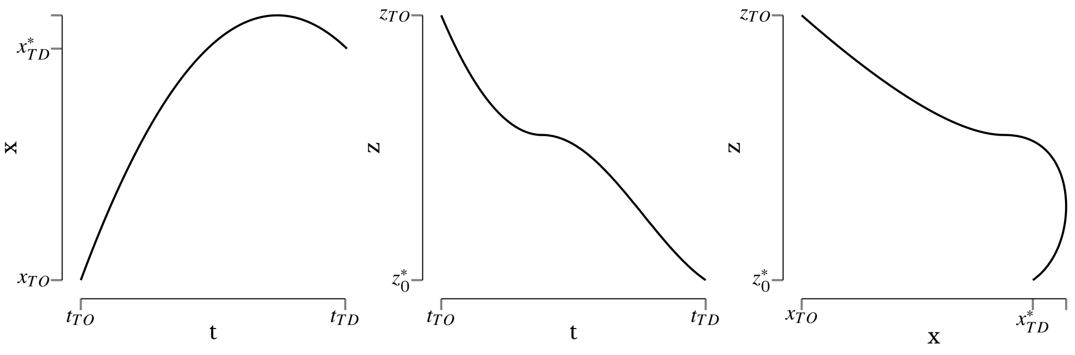


Figure 5.5: Desired foot point trajectory in the flight control phase. The horizontal trajectory is a quadratic function. The vertical trajectory is composed of a quadratic function, which increases z_{flight} to encourage leg retraction, and a cosine function to approach the landing condition.

where $x_{\text{flight}}(t)$ and $z_{\text{flight}}(t)$ are analytic functions that guide the foot to the desired landing condition at a predicted touchdown time $\hat{t}_{\text{TD}}$ (Fig. 5.5). These functions begin at the takeoff mirror position, $\boldsymbol{r}_{\text{mirror}}(t_{\text{TO}})$, and end at the desired deadbeat landing position, $\boldsymbol{r}_{\text{flight}}(\hat{t}_{\text{TD}}) = -[z^*_0/\tan(\alpha^*_{\text{TD},i}), z^*_0]$. The velocity at the expected touchdown time is chosen to match the ground speed, $\dot{\boldsymbol{r}}_{\text{flight}}(\hat{t}_{\text{TD}}) = [-\hat{\dot{x}}_{\text{TO}}, 0]$, based on the estimated horizontal velocity at takeoff. The predicted touchdown time

$$\hat{t}_{\text{TD}} = \frac{\hat{\dot{z}}_{\text{TO}}}{g} + \frac{1}{g}\sqrt{\hat{\dot{z}}^2_{\text{TO}} - 2g(z^*_0 - \hat{z}_{\text{TO}})}$$

is calculated from the expected touchdown state (Sec. 5.1.1) and from the estimated (Sec. 5.2.1) vertical CoM position $\hat{z}_{\text{TO}}$ and velocity $\hat{\dot{z}}_{\text{TO}}$ at takeoff.

Both the mirror and flight foot trajectories are mapped through the kinematics model of ATRIAS into leg joint trajectories $\mathbf{q}(t)$ that are tracked with position control (Fig. 5.3c). Here the compliance of the SEAs is ignored by assuming the motor output shafts are rigidly connected to the joints.

5.2.4 Online adaptation of return map

The final piece of control implementation is the online adaptation of the deadbeat control (5.5) derived from the approximate SMM. To counter small systematic modeling errors and imperfect torque tracking of ATRIAS, the observed error in the return map behavior of ATRIAS is approximated by a linear model

$$\hat{\dot{x}}_{i+1} - \dot{x}_a^* = \epsilon_1 \dot{x}_a^* + \epsilon_0, \tag{5.10}$$

where $\hat{\dot{x}}_{i+1}$ is the observed speed in the flight phase $i+1$ and ϵ_0 and ϵ_1 are obtained online through linear regression,

$$\begin{bmatrix} \epsilon_1 \\ \epsilon_0 \end{bmatrix} = (X^T X)^{-1} X^T Y,$$

with $X_i = [\dot{x}_a^* \; 1]$ and $Y_i = \hat{\dot{x}}_{i+1} - \dot{x}_a^*$. This error is compensated for by adapting the landing angle. For small deviations, the return map of the approximate SMM generates an error

$$\dot{x}_{i+1} - \dot{x}_a^* = \partial_{\dot{x}} f^* \left(\dot{x}_i - \dot{x}_a^*\right) + \partial_\alpha f^* \left(\alpha_{TD,i} - \alpha_{TD,i}^*\right)$$

with the partial derivatives pre-computed from the SMM return map. Hence, the observed error (5.10) is compensated for by the adapted landing angle,

$$\alpha_{TD,i}^\dagger = \alpha_{TD,i}^* - \left(\epsilon_1 \dot{x}_a^* + \epsilon_0 + \partial_{\dot{x}} f^* \left(\dot{x}_i - \dot{x}_a^*\right)\right) / \partial_\alpha f^*.$$

5.3 Hardware Experiments

To evaluate the planar running control developed in sections 5.1 and 5.2, we perform several experiments on undisturbed and disturbed locomotion using the ATRIAS biped attached to the boom (Fig. 5.1 and supplementary video). In this setup, power is supplied to the robot externally; however, all sensing and computation is performed on-board. Each experiment starts with ATRIAS standing still in a reference pose on one leg. A human operator then holds the boom to stabilize the robot while it follows a reference chirp signal for its CoM height. When takeoff occurs, the actual controller engages and the operator releases the boom. Besides the constant apex height target z_a^* (Fig. 5.2), the input provided in each trial by the operator to the ATRIAS controller is a profile of apex velocity targets $\dot{x}_a^*$ indexed by step number. The first and last velocity targets are always zero, and each experiment ends once the robot reaches the last step.

5.3.1 Undisturbed running

First, we evaluate the performance of the proposed controller in undisturbed running over level ground at a speed of $1\,\mathrm{m}\cdot\mathrm{s}^{-1}$. In this gait, about 80% of the available torque of ATRIAS's SEAs is consumed for generating the desired spring mass rebound behavior (eqs. 5.3 and 5.4) with a stiffness $k = 16\,\mathrm{kN}\cdot\mathrm{m}^{-1}$. This stiffness optimally trades off longer stance phases (larger vertical impulses) against the reduced mechanical advantage of ATRIAS's legs with increasing compression [72]. As a result, it enables the largest hopping heights of about 3 cm with appreciable flight times of about 150 ms (Fig. 5.2). The remaining 20% of torque capacity is available for error compensation. ATRIAS utilizes a large amount of torque to achieve this gait due to the low mechan-

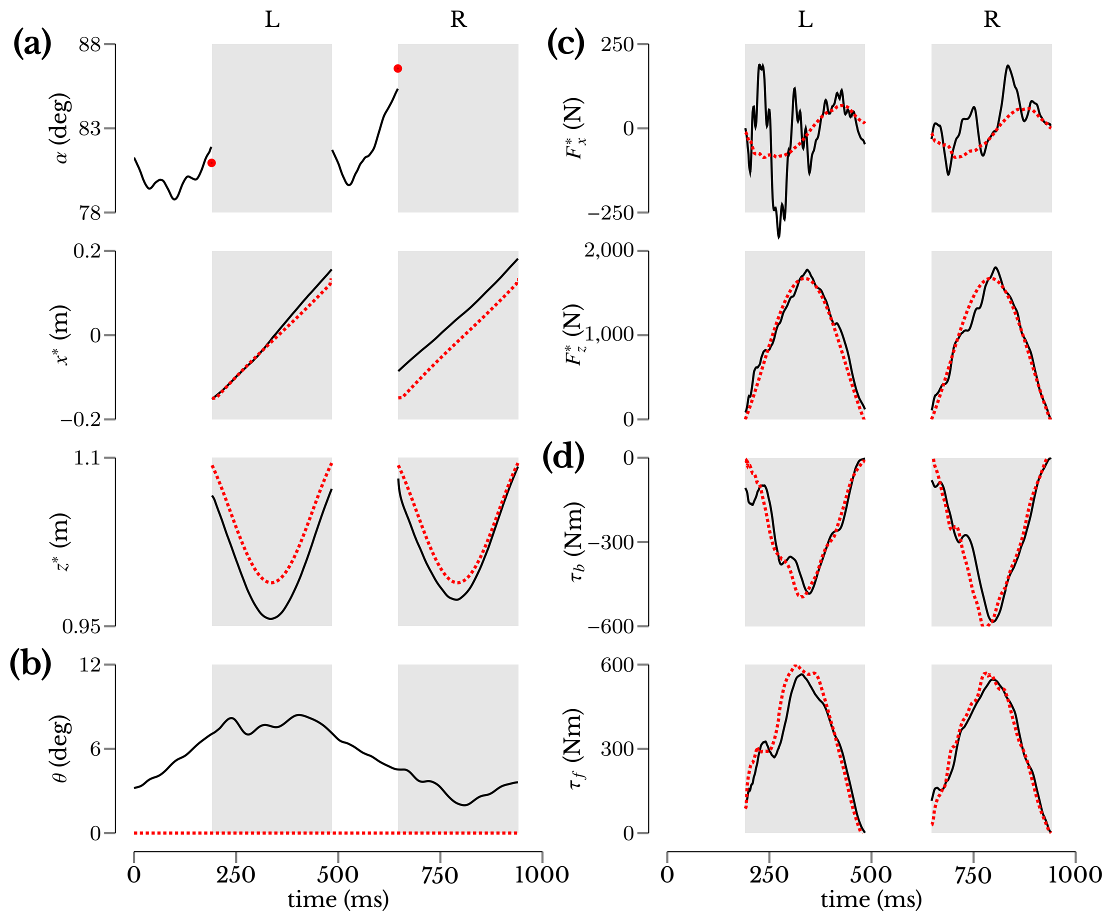


Figure 5.6: Tracking performance of implemented controller for ATRIAS running at $1\,\mathrm{m\cdot s^{-1}}$ over flat ground without gait disturbances. Shown are the desired (red dashed) and observed trajectories (black solid) of key control variables (Fig. 5.3) for two consecutive steps. The asymmetry between the left (L) and right leg (R) occurs due to the boom constraint.

ical advantage in its legs. Other similarly sized robots with different geometries would require different torques, but the ground reaction forces for this spring mass behavior would remain the same.

The tracking performance of the controller is summarized in figure 5.6. At the SMM level, the controller tracks the desired CoM trajectory $[x^*(t), z^*(t)]$ in stance with an error (mean and standard deviation) of 4.5 ± 4.7 cm in x and 2.6 ± 2.0 cm in z, and tracks the target leg angle at touchdown with an error of 0.99 ± 0.80 $^\circ$ (Fig. 5.6a). The model deviations originate from two primary sources. The first source is the GRF tracking error due to ground impacts, delayed contact detection, and limited actuator bandwidth. These force errors are reflected in the tracking of the desired SEA torques (53 ± 68 $\mathrm{N \cdot m}$ error, Fig. 5.6d), which is limited by a 20 Hz closed-loop bandwidth of ATRIAS's SEAs. The second source is the stance feedback control, which creates deviations from the simplified model in order to stabilize the trunk orientation (error of 7.6 ± 6.2 $^{\circ}$, Fig. 5.6b) as shown by the deviation in the GRF from the reference GRF of the SMM (error 110 ± 130 N, Fig. 5.6c).

5.3.2 Tracking SMM deadbeat velocity targets

In a second series of experiments we quantify how closely the implemented controller can realize the deadbeat behavior of the theoretical SMM model when the desired velocity $\dot{x}_a^*$ changes. We perform two sets of five repeated trials, in which ATRIAS runs over flat ground with desired apex velocities that change every five steps (Fig. 5.7a). In the first set, the change is $0.2\,\mathrm{m \cdot s^{-1}}$ with a maximum base velocity of $1.0\,\mathrm{m \cdot s^{-1}}$. In the second set, the change and maximum base velocity are $0.4\,\mathrm{m \cdot s^{-1}}$ and $1.6\,\mathrm{m \cdot s^{-1}}$, respectively. Larger step sizes require deadbeat foot targets beyond the mechanical limits of ATRIAS at high velocities. These limits impose a maximum possible velocity of $2.6\,\mathrm{m \cdot s^{-1}}$ for the chosen spring mass gait.

The observed velocity tracking performance is summarized in figure 5.7b. ATRIAS tracks desired velocity changes of $0.2\,\mathrm{m \cdot s^{-1}}$ (circles) with the average error observed in undisturbed running ($0.05\,\mathrm{m \cdot s^{-1}}$, dashed line; compare Sec. 5.3.1) after one step, indicating spring-mass-like deadbeat behavior within the performance bounds of undisturbed gait. However, the robot requires more steps for tracking $0.4\,\mathrm{m \cdot s^{-1}}$ changes (crosses), caused mainly

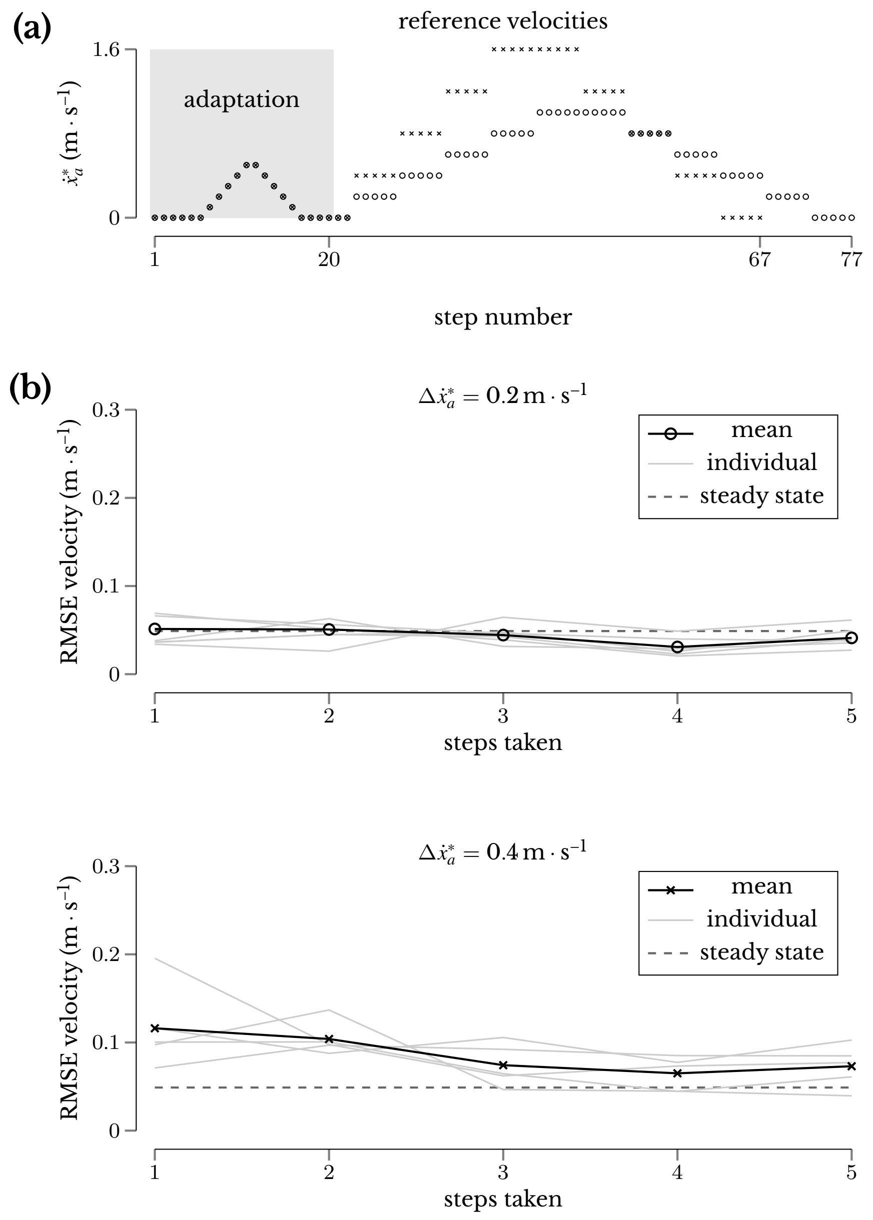


Figure 5.7: Tracking of SMM deadbeat velocity targets. (a) Profile of desired apex velocities $\dot{x}_a^*$ for changes of $0.2\,\mathrm{m \cdot s^{-1}}$ (circles) and $0.4\,\mathrm{m \cdot s^{-1}}$ (crosses). The first 20 steps are used in each trial for the online adaptation of the return map (Sec. 5.2.4) and do not count toward the experiments. (b) Root-mean-square error between the target velocity and the robot's velocity in flight over the number of consecutive steps taken after a change in the velocity target. Averages over all five trials are shown for the entire experiment (black) and separated out based on the different trials (gray). The dashed line indicates the average tracking error in undisturbed locomotion at $1\,\mathrm{m \cdot s^{-1}}$ (Sec. 5.3.1).

by increased ground impacts at the higher horizontal velocities. We measure a peak horizontal impact force of approximately 200 N when running at $1.0\,\mathrm{m \cdot s^{-1}}$. This impact force increases to nearly 300 N when running at $1.6\,\mathrm{m \cdot s^{-1}}$.

As described in section 5.3.1, the deviations from the simplified model are due to force tracking errors and trunk stabilization. For comparison to the hardware results, we quantify these two sources of deviation in simulation. When the simplified model is simulated with the same force errors measured on hardware, we observe similar velocity tracking errors of up to $0.15\,\mathrm{m \cdot s^{-1}}$ at $1.6\,\mathrm{m \cdot s^{-1}}$. When the intermediate complexity model (Fig. 5.3b) is simulated with an initial orientation error of 10°, we observe a velocity error of $0.05\,\mathrm{m \cdot s^{-1}}$ after one step. The system completely recovers after three steps. Thus, errors in force tracking and trunk orientation lead to a substantial performance deterioration compared to the SMM deadbeat control theory. This suggests that performance could be improved by extending the simplified model to account for ground impacts and rotational dynamics of the center body in legged locomotion.

5.3.3 Rejecting unexpected ground changes

With the third series of experiments, we explore how closely the implemented controller follows the deadbeat behavior of the SMM when the robot encounters unexpected changes in ground height. We perform experiments for six different ground height changes of ±6 cm, ±11 cm and ±15 cm, each repeated for three trials. In all trials, ATRIAS encounters the ground disturbance while running at $1.0\,\mathrm{m \cdot s^{-1}}$ with its reference gait. (Sec. 5.3.1).

Figure 5.8 shows the velocity tracking performance of ATRIAS after en-

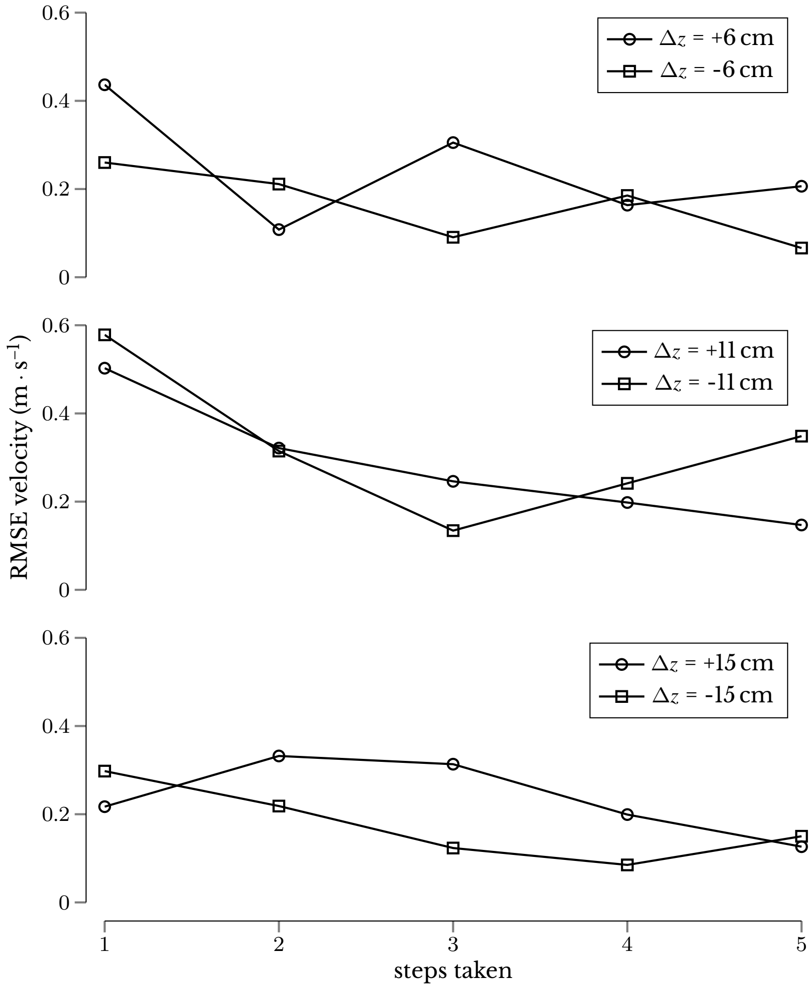


Figure 5.8: Ground disturbance rejection. Shown are the averages over three trials for the root-mean-square error between the desired velocity $x_a^* = 1.0\,\mathrm{m}\cdot\mathrm{s}^{-1}$ and the velocity achieved by the robot during flight over the number of consecutive steps taken after experiencing an unexpected ground height change Δz.

countering a ground height change measured as the error in velocity over the steps taken. Deadbeat behavior would result in an error no larger than the average error of $0.05\,\mathrm{m}\cdot\mathrm{s}^{-1}$ observed in undisturbed running at $1.0\,\mathrm{m}\cdot\mathrm{s}^{-1}$ from the first step on. However, each of the ground height changes results in a substantial velocity error in the first step of about the same size ($0.2\,\mathrm{m}\cdot\mathrm{s}^{-1}$ to $0.4\,\mathrm{m}\cdot\mathrm{s}^{-1}$), which only gradually diminishes in the next steps.

The velocity error and its gradual decay are largely independent of the di-

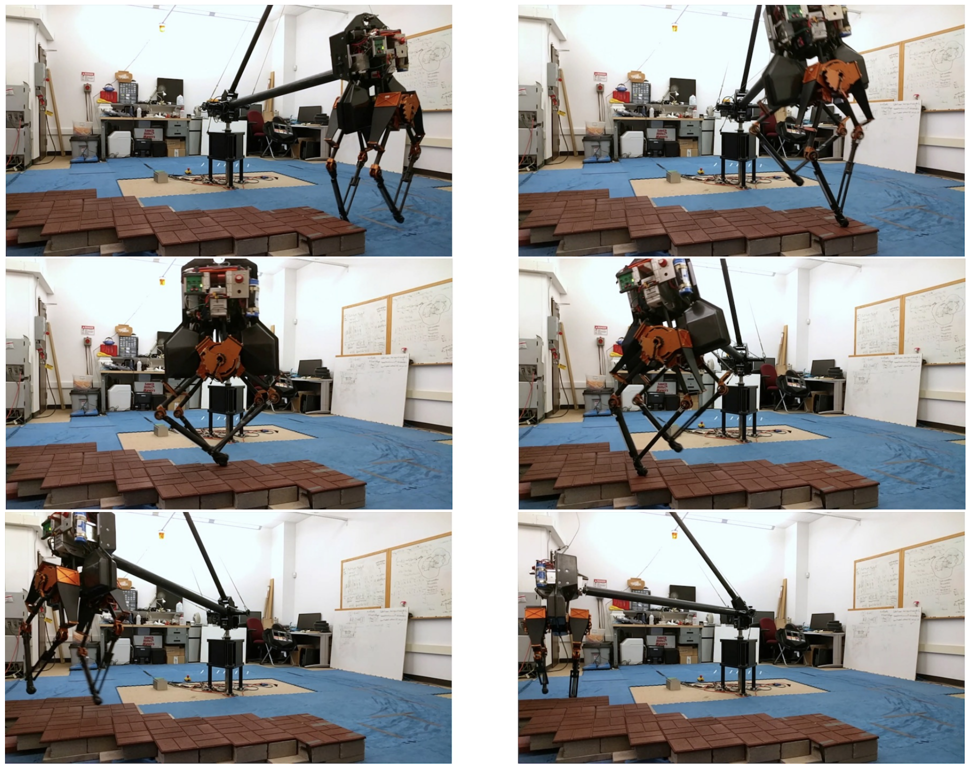


Figure 5.9: Photos of hardware experiments involving unexpected ground height changes for the ATRIAS biped.

rection and size of the ground height change, which seems counterintuitive. For instance, a height drop of 15 cm results in an increase in speed to $2\,\mathrm{m}\cdot\mathrm{s}^{-1}$ if maintaining the same total system energy. Similarly, a height increase of the same amount cannot be achieved without increasing system energy, even with zero speed. Comparing the two cases, it seems they should lead to very different behaviors, and thus velocity errors, after the disturbance.

The reason why the errors behave similarly is because they are due to the increased ground impacts and trunk orientation errors that are common to all of the height changes. The sudden ground height changes are implemented as sheer jumps in the floor surface using concrete blocks (Fig. 5.1). This leads to increased impact forces of nearly 500 N and swing foot impacts with the side of the elevated ground. Thus, most of the observed performance degradation compared to the SMM deadbeat control theory is again due to the increased ground impacts and required trunk stabilization. The detrimental effect of swing leg impacts suggests that hardware implementations should focus on more compliant swing leg motions than stiff kinematic control provides.

5.4 Conclusion

We investigated if the SMM leg placement theory can be transferred to running robots beyond the simplified one-legged test platforms used in previous studies. Specifically, we have evaluated the utility of spring mass theory on a robot of human scale and weight with an actively controlled trunk, articulated legs, and without external sensing. To this end, we focused on the ATRIAS biped platform and implemented on it a controller that transfers the SMM behavior through model-based force control in stance and kinematic control of foot placement in flight. We found that the proposed controller achieves on ATRIAS SMM-like deadbeat performance for velocity changes up to $\pm 0.2\,\mathrm{m \cdot s^{-1}}$. For larger velocity changes and for ground height changes ranging from ± 6 cm to ± 15 cm, the controller performance degraded, albeit without compromising gait robustness. The degradation was in large part due to ground impacts and the incessant need to stabilize the robot's trunk, neither of which are considered in the SMM theory. The results highlight the limited utility of this theory for the control of more complex running machines; on the other hand, they also point to the potential of such an SMM-based control for generating robust and versatile behavior in running robots.

The achieved performance mirrors the performance observed for the implementation of SMM-based deadbeat control strategies on the much simpler robot platforms. The velocity tracking error of 5% on ATRIAS during undisturbed running is in line with the results obtained by Zeglin, who demonstrated deadbeat hopping with a mean velocity error of approximately 15% between steps [133]. The ability of the proposed controller to tolerate unobserved rough terrain of at least 17% of the nominal leg length (0.9 m) is similar to the performance described by Shemer and Degani [114], who demonstrated deadbeat hopping over terrain height changes of about 15% of leg length. In contrast to the previous results, however, the demonstration of these capabilities on ATRIAS with similar performance shows that they generalize to more complex and human-like bipedal robots.

One key advantage of the model-based control framework [48, 64, 67] also pursued in this work is that it is easier to generalize behavior beyond scripted motions. For example, the MABEL robot is capable of planar running using a control framework based on hybrid zero dynamics. However, as the robot encounters perturbations, its controller must adapt speed to maintain stability leading to "considerable variation" in forward velocity [116]. Similarly, Hubicki and colleagues [57] discovered that ATRIAS is capable of 3-D running (although with very short flight phases of about 30 ms) when a heuristic controller designed for walking was commanded higher desired velocities. In contrast, our proposed controller can (within the bounds provided by the torque capacity of the actuators) freely choose the speed at which it runs from step to step, whether on flat ground or after a disturbance, by taking advantage of the underlying gait model and its deadbeat foot placement strategies.

Several research directions will help to further the model-based control framework for running robots. First, the SMM theory remains to be evaluated on robots running in 3-D environments. Second, performance degradation due to force errors and trunk stabilization suggests that the utility of the SMM theory could be increased by extending it to account for ground impacts and the rotational dynamics of a trunk. These force errors could also be mitigated by designing a more compliant swing leg control. Third, the mechanical limits of real robots prevent reaching certain target states in a single step. Robustness could potentially be improved in this case by considering these actuation limits [34, 40]. Finally, the transfer of SMM-based control to walking robots could substantially enlarge the range of robust behaviors that can be addressed. It is our goal to pursue these research directions in order to demonstrate highly robust 3-D running and walking on ATRIAS over uncertain terrain.

Chapter 6

Dynamic Programming Policies on ATRIAS

In this chapter, we present the details of implementing the policies generated from dynamic programming on a full sized bipedal robot, ATRIAS. First, we describe the modifications that need to be made to the controller framework described in the previous chapter to accommodate walking gaits. Second, we explain how to execute the optimal plans generated from a simplified model. Lastly, we demonstrate the generated controllers on bipedal robot hardware.

6.1 Extending the Control Framework for Walking

The deadbeat running framework described in the previous section is capable of robust locomotion (Fig. 6.1), but is limited to a single vertical running trajectory. This prevents the system from executing bipedal gaits which require a double support phase and forces the system to maintain a fixed hopping pattern regardless of desired horizontal velocity. This section will detail what control framework changes were necessary to allow the ATRIAS system

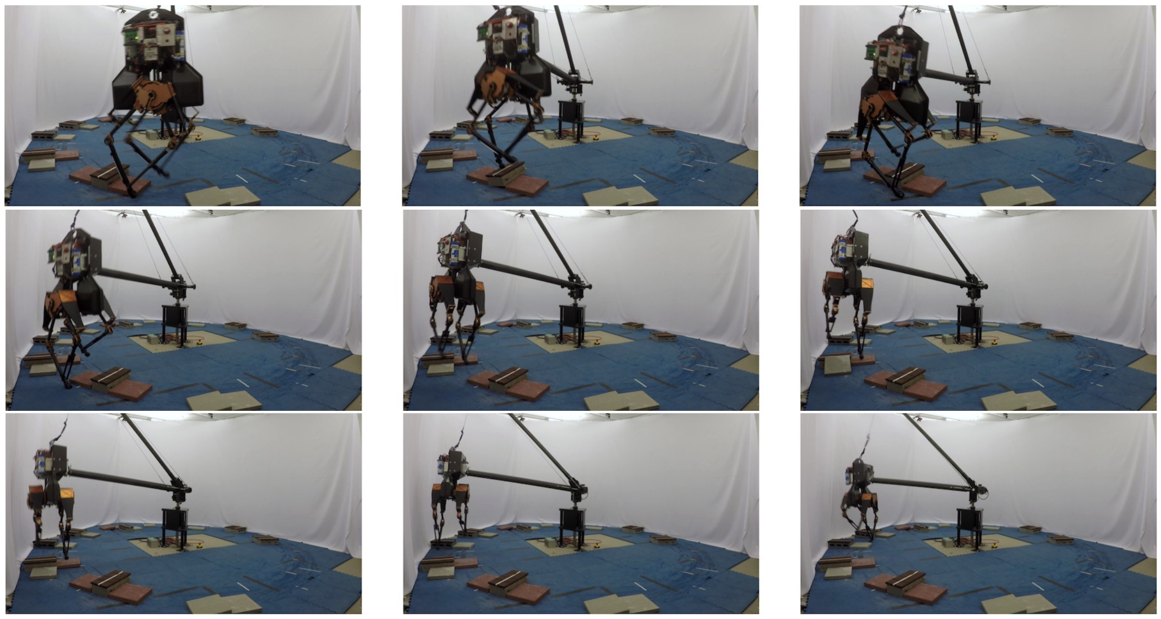


Figure 6.1: Example photos of our deadbeat running controller handling a large number of unexpected ground disturbances. The concrete blocks create changing ground heights and unsteady foot holds which often slide around during locomotion.

to target a range of gaits, including walking and standing, using the optimal controls calculated in chapter 3. These changes fall into three primary categories: system modeling, motion planning, and gait phase detection.

6.1.1 Modeling Double Support

The inclusion of a double support phase is necessary to execute walking plans on a bipedal system. Our heavily model-based controller design requires updated dynamics models in order to successfully plan and recreate behaviors with two feet on the ground. Although the floating point-mass model, used for state estimation (Sec. 5.2.1), does not need to be updated, the floating centroidal model (Eq. 5.6), used for predicting future dynamics, requires the position of the second foot hold. This modification doubles the number of force inputs and results in the dynamics

$$\begin{aligned} m\ddot{x} &= F_{x1} + F_{x2}, \\ m\ddot{z} &= F_{z1} + F_{z2} - mg, \\ I\ddot{\theta} &= -z(F_{x1} + F_{x2}) + xF_{z1} + (x+d)F_{z2}, \end{aligned} \tag{6.1}$$

where F_x and F_z now include indices for each leg and the quantity d defines the horizontal displacement from first to second foot point. We can simplify the dynamics equations slightly by replacing the horizontal force summations with $F_{x(1+2)}$; the horizontal forces do not feature independently in these equations.

Similarly, the full order dynamics equations used for reproducing desired ground forces on ATRIAS (Eq. 5.7), must also be extended to include the effects of a second foot point. We use the same Lagrangian approach detailed in [131] to create a mapping between joint torques and ground reaction forces of the system. Rather than abruptly change dynamics models after foot placement, we leverage underlying continuity of the single support and double support equations. We target zero force at the second foot point during the single support phase, which permits a unified set of inverse dynamics equations for all stance phases. In practice, the increased number of state variables now necessitates numerical matrix inversion rather than a closed-form analytic solution.

6.1.2 Executing Optimal Motion Plans

Following the optimal spring mass model motion plans requires a combination of stance and swing control for each leg. Compared to our initial deadbeat running controller, the system must now follow a significantly larger library of center of mass trajectories. We maintain the same architecture philosophy when designing this new control framework. Control is implemented in a hierarchical fashion using plans from the simplified model to produce the behaviors of an embedded virtual spring mass model.

Stance Control

During single and double support phases, legs in contact with the ground are used to produce desired motions of ATRIAS's center of mass and torso. This control pipeline follows the same broad structure as that used for deadbeat running (Fig. 5.3), but includes modifications to allow for more general motion behaviors. The pipeline contains four major components: (1) motion planner, (2) finite-horizon linear quadratic regulator, (3) inverse dynamics, and (4) series elastic actuator control.

The motion planner module produces desired center of mass trajectories based on previously generated optimal control lookup tables. These trajectories can either be formed online using gridded interpolation during a hardware rollout or offline as a static motion library. We investigated both planning methods on ATRIAS and found that the offline approach had clear advantages over online rollout for our system. This is due to the sensitivity of the underlying simplified model to control interpolation errors. During online forward simulations of the simplified model, important discrete events are occasionally missed unless the control lookup table has an unreasonably high resolution, which results in high memory consumption and lookup times. On the other hand, a static offline motion library requires less memory and provides a grid of reliable optimal trajectories. Although linear interpolation cannot be used between offline trajectories, due to their variable lengths, choosing a nearest neighbor initial state is sufficient for our control. In this work, each offline library contains approximately 30000 unique trajectories and requires 15 MB of memory.

Due to the large number of possible motion plans, finite-horizon LQR gains must now be generated online rather than offline in order to obey our system's memory constraints. We follow roughly the same process as before (Sec. 5.2.2) but now compute the optimal force control plan in real-time on ATRIAS. When a gait event is reached, such as the start of single or double support, the motion planner is queried to find the closest CoM trajectory $[x^*(t),\ z^*(t)]$ based on the current system state. The planner also provides a matching force trajectory $\mathbf{u}^*(t) = [F^*_{x1}(t) + F^*_{x2}(t),\ F^*_{z1}(t),\ F^*_{z2}(t)]$ based on the motion plan accelerations. Multiple feedforward force profiles are possible, so we select forces which do not create angular accelerations of the trunk. The CoM trajectories are combined with a desired trunk orientation $\theta^*(t) = 0$ to generate a complete reference $\boldsymbol{\xi}^*(t) = [x^*(t),\ \dot{x}^*(t),\ \theta^*(t),\ \dot{\theta}^*(t),\ z^*(t),\ \dot{z}^*(t)]$ for a floating rigid body. We use the updated centroidal dynamics (Eq. 6.1) to find the linearized error dynamics around this reference trajectory, which yields,

$$\Delta\dot{\boldsymbol{\xi}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{F^*_{z1}(t)+F^*_{z2}(t)}{I} & 0 & 0 & 0 & -\frac{F^*_{x1}(t)+F^*_{x2}(t)}{I} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \Delta\boldsymbol{\xi} + \begin{bmatrix} 0 & 0 & 0 \\ 1/m & 0 & 0 \\ 0 & 0 & 0 \\ \frac{-z^*(t)}{I} & \frac{x^*(t)}{I} & \frac{x^*(t)+d}{I} \\ 0 & 0 & 0 \\ 0 & 1/m & 1/m \end{bmatrix} \Delta\boldsymbol{u},$$

where $\Delta\boldsymbol{\xi} = \boldsymbol{\xi} - \boldsymbol{\xi}^*$ and $\Delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}^*$. Rather than decoupling the vertical state error dynamics as before, we now run finite-horizon LQR on the full order linear model to yield the optimal force control plan $\mathbf{u}(t) = \mathbf{u}^*(t) - \mathbf{K}(t)(\boldsymbol{\xi}(t) -$

$\boldsymbol{\xi}^*(t)$). This policy trades off between errors in the horizontal, vertical, and angular states of the system.

6.1.3 Contact Detection

Finally, the contact state estimation was modified to improve detection sensitivity and allow for arbitrary phase transitions between flight, single support, and double support. Rather than compare each legs' estimated vertical GRF, $\hat{F}_{z1}$ and $\hat{F}_{z2}$, against a stationary threshold of 50% body weight (Sec. 5.2.1), we now introduce a variable force threshold. We linearly adjust each foot's threshold between 20% and 50% of body weight based on the estimated rate of change of vertical force, $\hat{\dot{F}}_{z1}$ or $\hat{\dot{F}}_{z2}$, and the estimated vertical foot height assuming a flat terrain. We decrease the threshold when the rate of force change is high or the foot is near the ground, and increase the threshold when the rate of force change is low or the foot is far from the ground. This modification prevents large time delays in contact detection that can occur when ATRIAS's ground forces are near body weight, as is the case during walking gaits.

6.2 Demonstrating Optimal Gaits on the ATRIAS Biped

We combine the discrete control scheme described in section 3.5.2 with the framework extensions detailed in section 6.1 in order to realize the low-dimensional optimal gait plans on the ATRIAS biped. The full controller code contains a large number of components that must be carefully debugged when unexpected behaviors occur. Due to the time-intensive nature

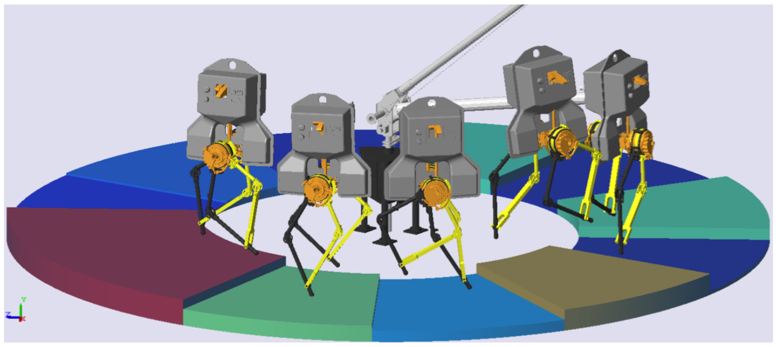


Figure 6.2: Simulation of the ATRIAS testbed used to pre-tune control and provide performance benchmarks for the hardware implementations.

of collecting hardware data, we use a high fidelity simulation of the system to untangle errors in the framework code and rapidly iterate through tunable controller parameters, such as LQR weights. This transition step allows us to deploy the control framework on hardware without having to resolve initial code errors during hardware experiments.

We have developed a detailed simulator of the ATRIAS testbed in MATLAB SimScape Multibody (Mathworks, Natick, MA) for pre-tuning the control and providing performance benchmarks for the hardware implementations (Fig. 6.2). The simulated environment contains 13 degrees of freedom (4 for the four-bar mechanisms, 4 for the SEA motor positions, 2 for the frontal plane motors, 1 for the trunk pitch, and 2 for the boom constraining roll and yaw) and models the main mechanical components (segmented chain, gear stages, springs, motor dynamics) and electrical components (electrical motor dynamics, discrete time control) of the robot testbed. In addition, the simulation includes contact points on the robot's feet, modeling the dissipative ground reaction dynamics as nonlinear spring-dampers with stick-slip transitions [52]. Gaussian sensor noise and joint friction are also included to create a realistic system.

Figure 6.3 shows a simulated ATRIAS experiment involving gait transitions between standing, walking, and running. ATRIAS begins by balancing in a standing double support pose and then commences walking at 0.6 $\mathrm{m \cdot s^{-1}}$ before transitioning into a slow running gait at the same forward velocity. This is followed by progression to a faster running gait at 1.2 $\mathrm{m \cdot s^{-1}}$. Each of these gait transitions is then performed in reverse order to bring the system back to rest. State variable tracking is shown in figure 6.4.

We are able to transfer the same control code to the ATRIAS hardware and rollout the policies in real-time (Fig. 6.5). Although certain features, such as force tracking and contact estimation degrade slightly, the system is able to produce comparable performance to that of simulation. However, the rate at which motion plans can be queried must be reduced from 1000 Hz to 125 Hz to prevent overloading the robot's CPU. This leads to additional delays in phase transitions, but does not notably destabilize the system. Figure 6.6 shows an example of tracking performance on the robot hardware. A new center of mass motion plan is generated at the start of each step, resulting in diverse behaviors of the system. Each plan provides a corrective trajectory that brings the system closer to the targeted gait.

6.2.1 Executing online generated plans on the hardware

One way of implementing dynamic programming policies on hardware is to collect offline generated optimal control actions like leg stiffness, damping, and leg angle for a wide range of initial conditions. Next on hardware, we can interpolate between collected actions and integrate the simplified model dynamics to get a desired trajectory of the center of mass and touch down angles. This target policy can now be executed on the robot using the framework described above. However, in practice, the generated trajectories of-

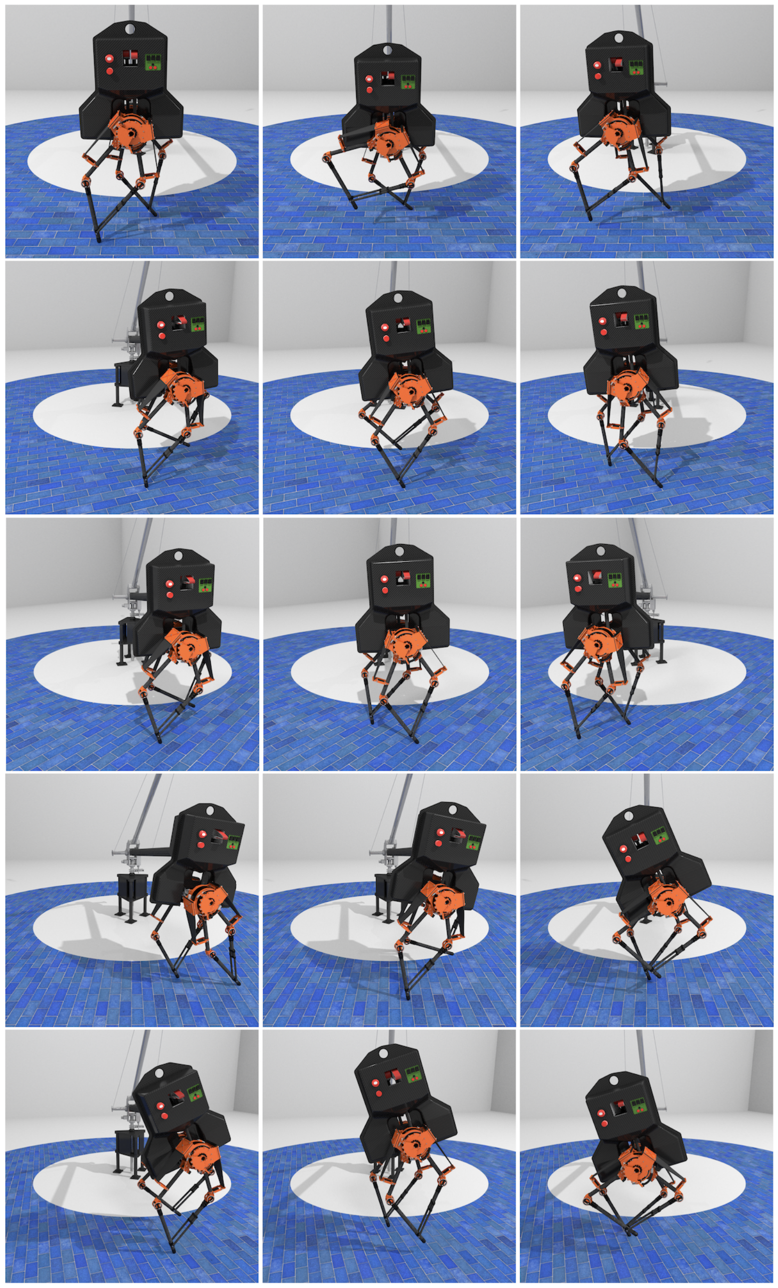


Figure 6.3: Simulation of spring mass model gait transitions on the ATRIAS biped. This experiment incorporates standing, walking, and running behaviors based on optimal motions plans for the system's center of mass. Row 1 & 2: Stand to walk, Row 3: Walking, Row 4 & 5: Walk to run.

ten miss zero-crossings on Poincare events like foot touchdown. In such a case, the simplified model does not detect the next gait event, leading to the robot falling. Also, generating online trajectories with a small integration

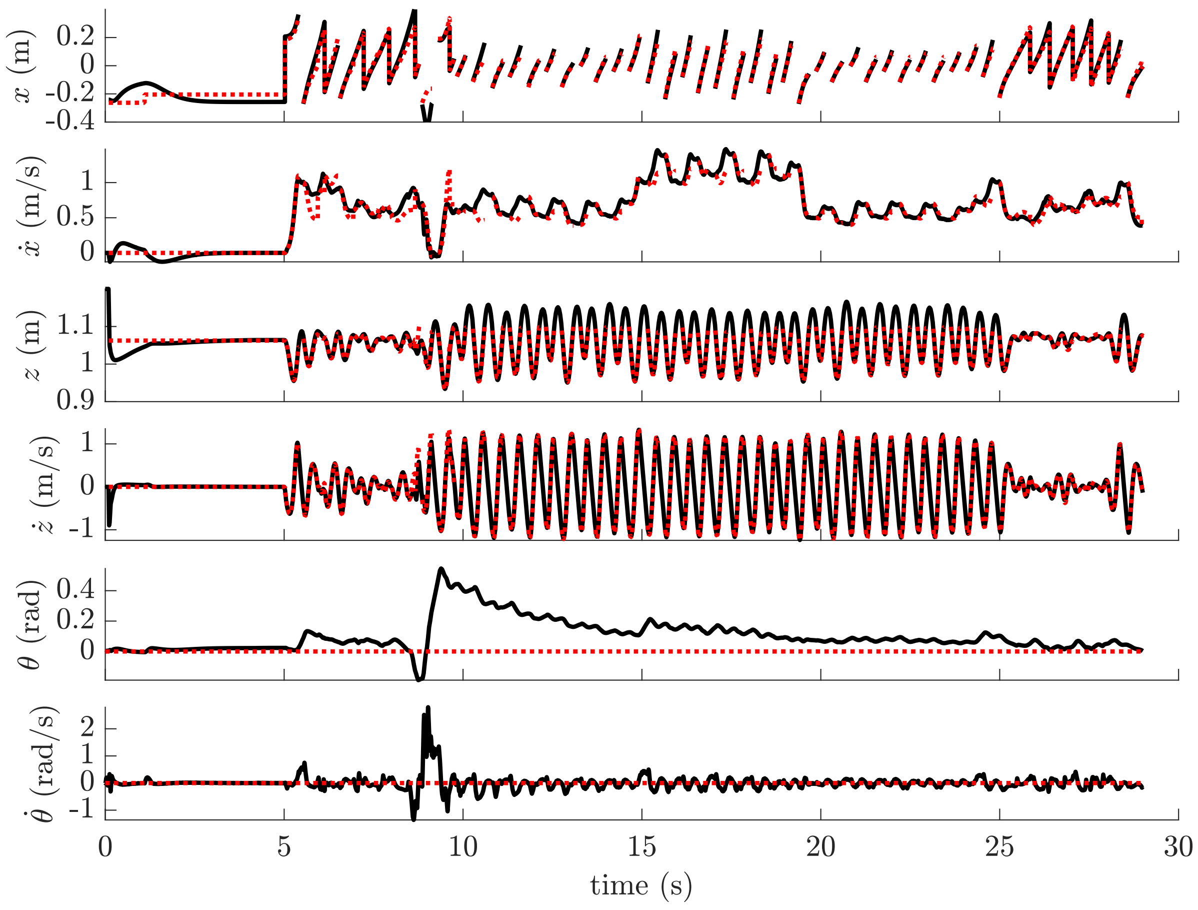


Figure 6.4: Trajectories of regulated state variables during simulated gait transitions on ATRIAS (Fig. 6.3). Most transitions are smooth, but the walk-to-run example here (at approximately 9 seconds) demonstrates a large torso disturbance and recovery. Dotted red lines indicate desired motion plans from the simplified model, while black solid lines indicate measured quantities of ATRIAS.

time-step can be slow. This motivates using offline-generated plans when executing control policies on hardware.

6.2.2 Executing offline plans on the hardware

In an attempt to alleviate the problems described above, we collect integrated trajectories of center of mass motion and touch down angle offline. These dynamic programming plans are generated over a large number of

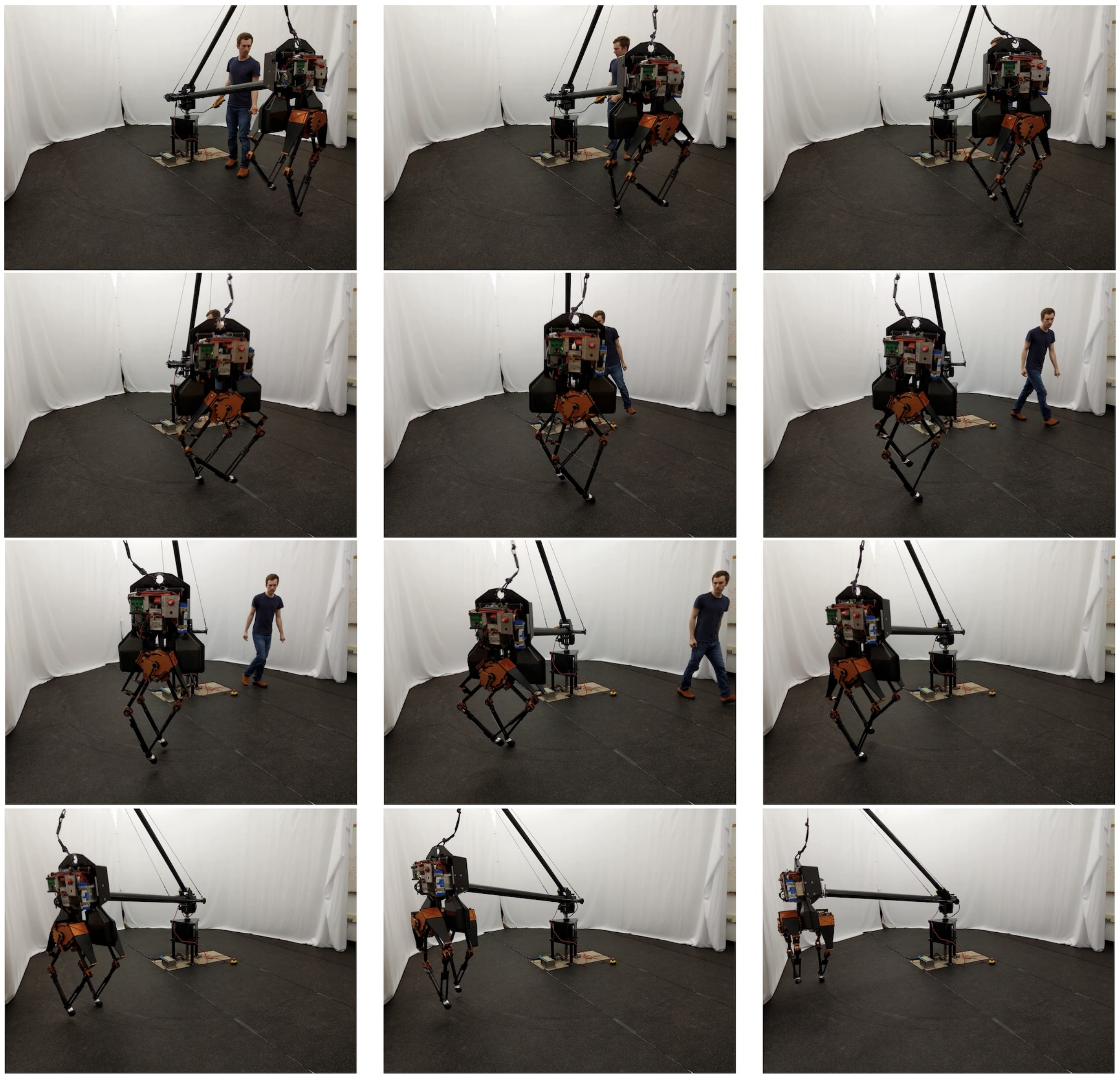


Figure 6.5: Hardware experiments of general spring mass model gaits on the ATRIAS biped over flat terrain. These policies involve over 30,000 center of mass trajectories generated using dynamic programming on a spring mass model.

initial conditions for a spring mass model, and followed on hardware using the frameworks described before. On hardware, we query the pre-generated plans for the nearest initial state at every gait event. This plan now becomes the desired center of mass plan. However, the offline generated plans lead to a very large lookup table that can be difficult to query at 1000Hz. As a result, we bring down the rate of trajectory lookup from 1000 to 250Hz for the

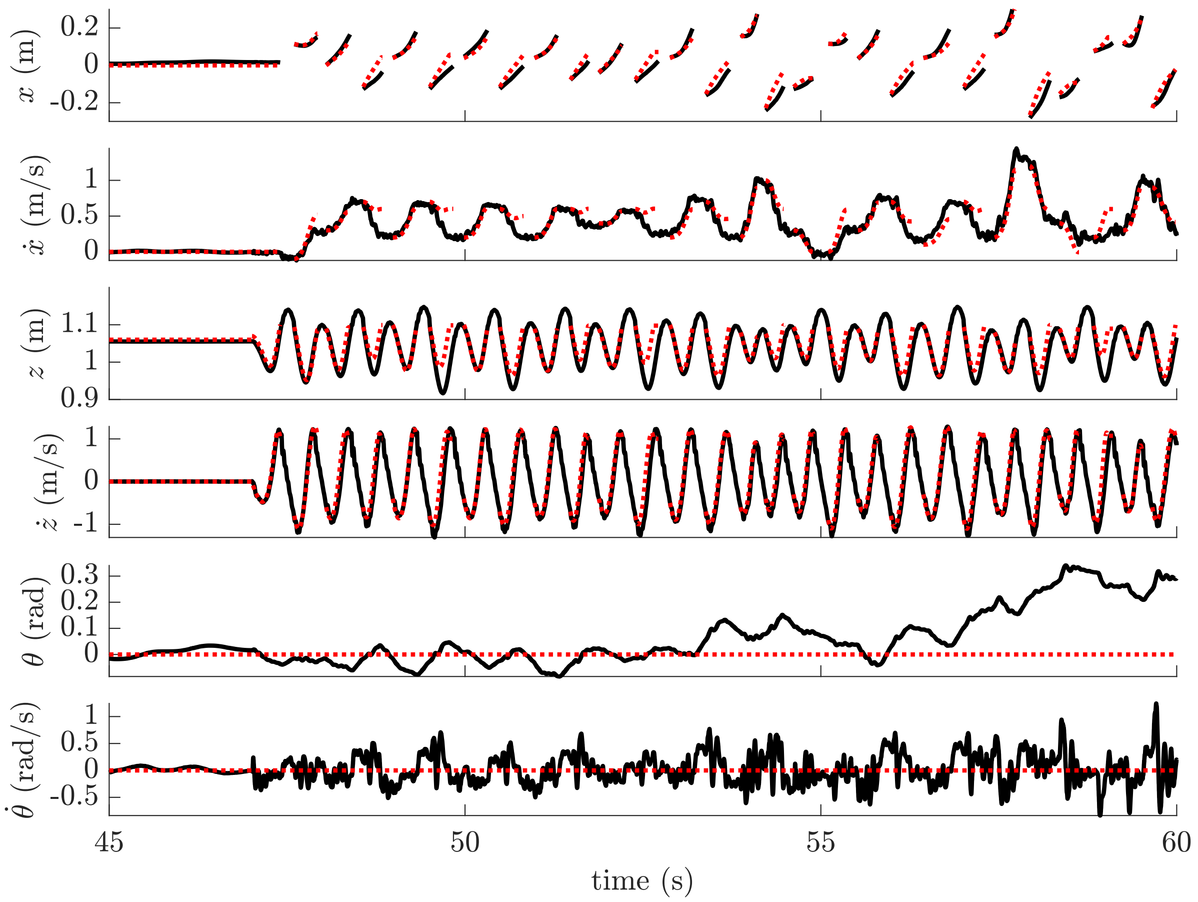


Figure 6.6: Trajectories of regulated state variables during actual hardware rollouts on ATRIAS. Tracking performance is similar to that observed in simulation, but the system is sensitive to tunable controller parameters. Dotted red lines indicate desired motion plans from the simplified model, while black solid lines indicate measured quantities of ATRIAS.

center of mass planner. The inverse dynamics is still conducted at 1000Hz, but the center of mass plan is now updated at a lower frequency.

6.3 Discussion

In this chapter, we described our attempts at executing dynamic programming policies generated on simplified models on the ATRIAS bipedal robot. We started by developing a control framework that can generalize to walking and running gaits, so that the policies generated from dynamic programming can be executed on the robot. With this in place, we could implement dynamic programming control policies for walking and running targets on the ATRIAS simulation and hardware. This is a demonstration of a unified gait transition framework for bipedal systems on a full-size bipedal robot.

In our experiments, we found that the stability of the optimal policies from simulation depended greatly on the accuracy of center of mass motion tracking on hardware. Though the policies were able to reject some disturbance by switching to running from walking, the generated policies were prone to tilting the torso, or slipping at contact. These are physical aspects of the system which were not considered in the simplified model. Hence, it is not surprising that the policies generated are not robust to these disturbances.

Since dynamic programming can generalize to more complex simplified models, albeit at a computational cost, it would be beneficial to study how to incorporate such features into the simplified model. Incorporating more details about the problem, while keeping the problem computationally tractable is an interesting avenue that can be explored in the future.

Chapter 7

Conclusion

7.1 Overview

Collectively, this work provides a better general understanding of how compliant simplified models can be used to realize multiple dynamic gaits on legged robot platforms. Point-mass models provide very low-dimensional locomotion templates, which work well with value-based optimization methods, such as dynamic programming. When using this kind of model, the necessary discrete value function grid can be densely sampled for two reasons: they have relatively small state spaces and dynamics that can be quickly numerically integrated. This enables methods such as value iteration to populate the space with billions of samples using few computational resources. In this context, the utility of locomotion models is a trade-off between how easily their optimal control can be computed and the level of detail that they provide. Although the spring mass model is a coarse representation of a legged system, it captures the essential components of walking and running. This makes it general enough to inform controllers for a wide variety of legged robots. Furthermore, the center of mass level gaits that it can produce have nearly the best possible performance for a point-mass model. Leveraging these policies allows us to not only robustly control simulated models, but also leads to robust walking and running on legged robot hardware.

7.2 Contributions

In this thesis, we made three contributions spanning the theory and practice of compliant models for controlling legged robots.

First, we apply dynamic programming methods to compute approximations of the continuous-time optimal control for a bipedal point-mass model. Although these results are generated numerically, they represent approximations of the underlying globally optimal solutions and are free from heuristic design decisions.

Second, we present an event-based control framework for the spring mass model, which is able to freely transition between flight, single support, and double support phases without relying on secondary optimizations. This enables the model to target standing, walking, running, and gait transitions within a single control structure based on a single simplified model. The resulting motion plans can be generated efficiently for use on real-time robot systems.

Third, we demonstrate that these spring mass model motion plans can be transferred to bipedal robot hardware using a model-based controller to embed the center of mass behavior. This leads to stable standing, walking, and running on the ATRIAS platform.

7.3 Future Research Directions

A necessary shortcoming of optimizing very simple models is their inability to capture all the details of a higher order system. The point-mass models used in this work are capable of achieving many walking and running gaits, but are not guaranteed to respect the constraints of arbitrary robot hardware. For ATRIAS, we primarily encountered constraints on the robot's kinematic configurations and available force bandwidth; the extent of these restrictions ultimately depend on the specific robot hardware. Future work on optimal gait controllers would strongly benefit from finding ways to incorporate these platform-specific limits into the low-order motion plans. Increasing the model dimensionality to include a rotating torso or the center of mass acceleration is a reasonable first step, but comes at the cost of significantly increased computational complexity. For example, when adding two additional dimensions to controllers in this work, solutions converge on the order of days or weeks rather than hours. This can be combated by accepting higher levels of approximation during the optimization process. While we expect legged robot technology to continue to expand in capability, more advanced hardware will likely introduce more degrees of freedom and new problem constraints. For this reason, simplified model based control schemes will need to consider how to deal with these real-world complexities.

Bibliography

[1] R Alexander and AS Jayes. Vertical movements in walking and running. *Journal of Zoology*, 185(1):27–40, 1978. 2.1.1

[2] R Alexander and Alexandra Vernon. The mechanics of hopping by kangaroos (macropodidae). *Journal of Zoology*, 177(2):265–303, 1975. 2.1.1

[3] R McN Alexander. A model of bipedal locomotion on compliant legs. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 338(1284):189–198, 1992. 2.1.1

[4] RM Alexander. Optimization and gaits in the locomotion of vertebrates. *Physiological reviews*, 69(4):1199–1227, 1989. 1.1

[5] RMcn Alexander. Mechanics of bipedal locomotion. *Perspectives in experimental biology*, 1:493–504, 1976. 2.1.1

[6] Adamantios Arampatzis, Gert-Peter Brüggemann, and Verena Metzler. The effect of speed on leg stiffness and joint kinetics in human running. *Journal of biomechanics*, 32(12):1349–1353, 1999. 3.2.1

[7] Christopher G Atkeson. Randomly sampling actions in dynamic programming. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 185–192. IEEE, 2007. 2.2.1, 3.4.2

[8] M Bardi and M Falcone. An approximation scheme for the minimum

time function. *SIAM Journal on Control and Optimization*, 28(4):950–965, 1990. 3.3.2

[9] M Bardi and M Falcone. Discrete approximation of the minimal time function for systems with regular optimal trajectories. In *Analysis and Optimization of Systes*, pages 103–112. Springer, 1990. 3.3.2

[10] Martino Bardi and Italo Capuzzo-Dolcetta. *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008. 3.3.2

[11] Guy Barles. Discontinuous viscosity solutions of first-order hamilton-jacobi equations: a guided visit. *Nonlinear Analysis: Theory, Methods & Applications*, 20(9):1123–1134, 1993. 3.3.2

[12] Guy Barles and Panagiotis E Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. *Asymptotic analysis*, 4(3):271–283, 1991. 3.3.2

[13] R Bellman. Curse of dimensionality. *Adaptive control processes: a guided tour. Princeton, NJ*, 1961. 3.3.2

[14] R. Bellman. *Dynamic Programming*. Dover Books on Computer Science. Dover Publications, 2013. ISBN 9780486317199. URL `https://books.google.com/books?id=CG7CAgAAQBAJ`. 2.2.1

[15] Richard Bellman. Dynamic programming. 1957. 3.3.2

[16] Hamid Benbrahim and Judy A Franklin. Biped dynamic walking using reinforcement learning. *Robotics and Autonomous Systems*, 22(3):283–302, 1997. 2.2.2

[17] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic Programming*. Anthropological Field Studies. Athena Scientific, 1996. ISBN 9781886529106.

URL `https://books.google.com/books?id=WxCCQgAACAAJ`. 2.2.1, 3.3.2

[18] Andrew A Biewener and C Richard Taylor. Bone strain: a determinant of gait and speed? *Journal of experimental Biology*, 123(1):383–400, 1986. 1.1

[19] Reinhard Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22(11-12):1217–1227, 1989. 2.1.1, 3, 3.2.1

[20] Reinhard Blickhan and RJ Full. Similarity in multilegged locomotion: bouncing like a monopode. *Journal of Comparative Physiology A*, 173(5): 509–517, 1993. 2.1.1

[21] Dennis M Bramble and Daniel E Lieberman. Endurance running and the evolution of homo. *Nature*, 432(7015):345, 2004. 1.1

[22] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010. 2.2.1

[23] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. An experimental comparison of bayesian optimization for bipedal locomotion. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1951–1958. IEEE, 2014. 2.2.2

[24] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):5–23, 2016. 2.2.2

[25] Sean G Carver, Noah J Cowan, and John M Guckenheimer. Lateral stability of the spring-mass hopper suggests a two-step control strategy for running. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(2): 026106, 2009. 2.1.2, 3.3.2, 5, 5.1

[26] Sean Goodwin Carver. *Control of a spring-mass hopper*. Cornell University, 2003. 2.1.2

[27] GA Cavagna. The landing–take-off asymmetry in human running. *Journal of Experimental Biology*, 209(20):4051–4060, 2006. 3.2.4

[28] GA Cavagna and M Kaneko. Mechanical work and efficiency in level walking and running. *The Journal of physiology*, 268(2):467, 1977. 2.1.1, 3.2.4

[29] GA Cavagna, FP Saibene, and R Margaria. External work in walking. *Journal of applied physiology*, 18(1):1–9, 1963. 3.2.4

[30] GA Cavagna, FP Saibene, and R Margaria. Mechanical work in running. *Journal of applied physiology*, 19(2):249–256, 1964. 2.1.1, 3.2.4

[31] Giovanni A Cavagna. Storage and utilization of elastic energy in skeletal muscle. *Exercise and sport sciences reviews*, 5(1):89–130, 1977. 2.1.1

[32] Giovanni A Cavagna, H Thys, and A Zamboni. The sources of external work in level walking and running. *The Journal of physiology*, 262(3):639, 1976. 2.1.1

[33] Giovanni A Cavagna, Norman C Heglund, and C Richard Taylor. Mechanical work in terrestrial locomotion: two basic mechanisms for minimizing energy expenditure. *American Journal of Physiology-Regulatory, Integrative and Comparative Physiology*, 233(5):R243–R261, 1977. 1.1, 2.1.1, 3.1

[34] Tom Cnops, Zhenyu Gan, and C David Remy. The basin of attraction for running robots: Fractals, multistep trajectories, and the choice of control. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 1586–1591. IEEE, 2015. 3, 3.3.2, 5.4

[35] Creative Commons. Attribution-NoDerivs 2.0 generic license. URL `https://creativecommons.org/licenses/by-nd/2.0/`. 1.1

[36] Michael H Dickinson, Claire T Farley, Robert J Full, MAR Koehl, Rodger Kram, and Steven Lehman. How animals move: an integrative view. *Science*, 288(5463):100–106, 2000. 2.1.1

[37] J Maxwell Donelan, Rodger Kram, and Arthur D Kuo. Mechanical work for step-to-step transitions is a major determinant of the metabolic cost of human walking. *Journal of Experimental Biology*, 205(23):3717–3727, 2002. 3.2.5

[38] Boston Dynamics, 2019. URL `https://www.bostondynamics.com/atlas`. 2.3.2

[39] Gen Endo, Jun Morimoto, Takamitsu Matsubara, Jun Nakanishi, and Gordon Cheng. Learning cpg sensory feedback with policy gradient for biped locomotion for a full-body humanoid. In *Proceedings of the national conference on artificial intelligence*, volume 20, page 1267. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005. 2.2.2

[40] Michael Ernst, Hartmut Geyer, and Reinhard Blickhan. Extension and customization of self-stability control in compliant legged systems. *Bioinspiration & biomimetics*, 7(4):046002, 2012. 5.4

[41] Marizio Falcone and Roberto Ferretti. Discrete time high-order schemes for viscosity solutions of hamilton-jacobi-bellman equations. *Numerische Mathematik*, 67(3):315–344, 1994. 3.3.2

[42] Maurizio Falcone. Numerical solution of dynamic programming equations. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman equations. Birkhäuser*, 1997. 3.3.2

[43] Maurizio Falcone and Roberto Ferretti. *Semi-Lagrangian approximation schemes for linear and Hamilton-Jacobi equations*, volume 133. SIAM, 2013. 3.3.2, 3.3.2, 3.3.2

[44] Claire T Farley and Daniel P Ferris. 10 biomechanics of walking and running: Center of mass movements to muscle action. *Exercise and sport sciences reviews*, 26(1):253–286, 1998. 2.1.1

[45] Claire T Farley and Octavio GonzalezArampatzis. Leg stiffness and stride frequency in human running. *Journal of biomechanics*, 29(2):181–186, 1996. 2.1.1, 3.2.1

[46] Claire T Farley and C Richard Taylor. A mechanical trigger for the trot-gallop transition in horses. *Science*, 253(5017):306–308, 1991. 1.1

[47] Claire T Farley, James Glasheen, and Thomas A McMahon. Running springs: speed and animal size. *Journal of experimental Biology*, 185(1): 71–86, 1993. 2.1.1

[48] 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. 2.3.1, 5, 5.1.2, 5.4

[49] Daniel P Ferris, Micky Louie, and Claire T Farley. Running in the real world: adjusting leg stiffness for different surfaces. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1400):989–994, 1998. 2.1.1

[50] Robert J Full and Daniel E Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332, 1999. 2.1.3

[51] Steven A Gard, Steve C Miff, and Arthur D Kuo. Comparison of kine-

matic and kinetic methods for computing the vertical motion of the body center of mass during walking. *Human movement science*, 22(6): 597–610, 2004. 3.2.3

[52] Hartmut Geyer and Hugh Herr. A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities. *IEEE Transactions on neural systems and rehabilitation engineering*, 18(3):263–273, 2010. 6.2

[53] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society of London B: Biological Sciences*, 273(1603): 2861–2867, 2006. 2.1.3, 3, 3.1, 3.2.2

[54] Alastair Hanna, Bruce Abernethy, Robert J Neal, and Robin Burgess-Limerick. Triggers for the transition between human walking and running. *Energetics of human activity*, pages 124–164, 2000. 1.1

[55] Jessica K Hodgins. Biped gait transitions. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2092–2097. IEEE, 1991. 2.1.3, 2.3.2, 3

[56] Alan Hreljac, Alan Arata, Reed Ferber, John A Mercer, and Brandi S Row. An electromyographical analysis of the role of dorsiflexors on the gait transition during human locomotion. *Journal of applied biomechanics*, 17(4):287–296, 2001. 1.1

[57] Christian Hubicki, Andy Abate, Patrick Clary, Siavash Rezazadeh, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Ryan Domres, Albert Wu, William Martin, Hartmut Geyer, and Jonathan Hurst. Walking and running with passive compliance: Lessons from engineering a live demonstration of the atrias biped. *IEEE Robotics and Automation*

Magazine, 2016. 2.3.2, 3, 5, 5.4

[58] Christian Hubicki, Jesse Grimes, Mikhail Jones, Daniel Renjewski, Alexander Spröwitz, Andy Abate, and Jonathan Hurst. Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot. *The International Journal of Robotics Research*, page 0278364916648388, 2016. 1.2.3, 5.1

[59] Dong Jin Hyun, Sangok Seok, Jongwoo Lee, and Sangbae Kim. High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. *The International Journal of Robotics Research*, 33(11):1417–1445, 2014. 5

[60] Shuuji Kajita and Kazuo Tani. Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 1405–1411. IEEE, 1991. 3

[61] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 1620–1626. IEEE, 2003. 3

[62] Tony S Keller, AM Weisberger, JL Ray, SS Hasan, RG Shiavi, and DM Spengler. Relationship between vertical ground reaction force and speed during walking, slow jogging, and running. *Clinical biomechanics*, 11(5):253–259, 1996. 3.2.4

[63] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1):43–53, 1987. 2.3.1

[64] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas De Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and Jerry Pratt. Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, 13(01): 1650007, 2016. 2.3.1, 5, 5.1.2, 5.4

[65] Rodger Kram, Antoinette Domingo, and Daniel P Ferris. Effect of reduced gravity on the preferred walk-run transition speed. *Journal of Experimental Biology*, 200(4):821–826, 1997. 1.1

[66] SN Kružkov. Generalized solutions of the hamilton-jacobi equations of eikonal type. i. formulation of the problems; existence, uniqueness and stability theorems; some properties of the solutions. *Sbornik: Mathematics*, 27(3):406–446, 1975. 3.3.2

[67] 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. *Autonomous Robots*, 40(3): 429–455, 2016. 2.3.1, 5, 5.1.2, 5.4

[68] Cynthia R Lee and Claire T Farley. Determinants of the center of mass trajectory in human walking and running. *Journal of experimental biology*, 201(21):2935–2944, 1998. 3.2.3, 3.2.4

[69] Sung-Hee Lee and Ambarish Goswami. Reaction mass pendulum (rmp): An explicit model for centroidal angular momentum of humanoid robots. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 4667–4672. IEEE, 2007. 2.3.1

[70] Susanne W Lipfert, Michael Günther, Daniel Renjewski, Sten Grimmer, and Andre Seyfarth. A model-experiment comparison of system

dynamics for human walking and running. *Journal of Theoretical Biology*, 292:11–17, 2012. 2.1.3, 3.2.1, 3.2.2

[71] R Margaria, P Cerretelli, P Aghemo, and G Sassi. Energy cost of running. *Journal of applied physiology*, 18(2):367–370, 1963. 1.1

[72] William C Martin, Albert Wu, and Hartmut Geyer. Robust spring mass model running for a physical bipedal robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6307–6312. IEEE, 2015. 5.3.1

[73] Harold Roberto Martinez and Juan Pablo Carbajal. From walking to running a natural transition in the slip model using the hopping gait. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 2163–2168. IEEE, 2011. 3

[74] Tad McGeer. Passive bipedal running. *Proceedings of the Royal Society of London B: Biological Sciences*, 240(1297):107–134, 1990. 2.1.2

[75] Robert T M'Closkey and Joel W Burdick. Periodic motions of a hopping robot with vertical and forward motion. *The International journal of robotics research*, 12(3):197–218, 1993. 2.1.2

[76] TA McMahon. The role of compliance in mammalian running gaits. *Journal of Experimental Biology*, 115(1):263–282, 1985. 2.1.1

[77] Thomas A McMahon and George C Cheng. The mechanics of running: how does stiffness couple with speed? *Journal of biomechanics*, 23:65–78, 1990. 2.1.1, 3.2.1, 3.2.2

[78] Thomas A McMahon and Peter R Greene. Fast running tracks. *Scientific American*, 239(6):148–163, 1978. 2.1.1

[79] Thomas A McMahon, Gordon Valiant, and Edward C Frederick. Grou-

cho running. *Journal of Applied Physiology*, 62(6):2326–2337, 1987. 2.1.1, 2.1.3, 3.2.4

[80] Igor Mordatch, Martin De Lasa, and Aaron Hertzmann. Robust physics-based locomotion using low-dimensional planning. In *ACM Transactions on Graphics (TOG)*, volume 29, page 71. ACM, 2010. 5.1.1

[81] Takeshi Mori, Yutaka Nakamura, Masa-Aki Sato, and Shin Ishii. Reinforcement learning for cpg-driven biped robot. In *AAAI*, volume 4, pages 623–630, 2004. 2.2.2

[82] Jun Morimoto and Christopher G Atkeson. Learning biped locomotion. *IEEE Robotics & Automation Magazine*, 14(2):41–51, 2007. 2.2.2

[83] Ben Morris, ER Westervelt, Christine Chevallereau, Gabriel Buche, and JW Grizzle. Achieving bipedal running with rabbit: Six steps toward infinity. In *Fast Motions in Biomechanics and Robotics*, pages 277–297. Springer, 2006. 2.3.2, 3

[84] Yutaka Nakamura, Takeshi Mori, and Shin Ishii. Natural policy gradient reinforcement learning for a cpg control of a biped robot. In *International Conference on Parallel Problem Solving from Nature*, pages 972–981. Springer, 2004. 2.2.2

[85] David E Orin and Ambarish Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 653–659. IEEE, 2008. 2.3.1

[86] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Dynamic terrain traversal skills using reinforcement learning. *ACM Transactions on Graphics (TOG)*, 34(4):80, 2015. 2.2, 2.2.1

[87] Xue Bin Peng, Glen Berseth, and Michiel van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):81, 2016. 2.2, 2.2.1

[88] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008. 2.2.1

[89] Jerry Pratt, Chee-Meng Chew, Ann Torres, Peter Dilworth, and Gill Pratt. Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143, 2001. 3

[90] Jerry Pratt, John Carff, Sergey Drakunov, and Ambarish Goswami. Capture point: A step toward humanoid push recovery. In *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*, pages 200–207. IEEE, 2006. 3

[91] Boris I Prilutsky and Robert J Gregor. Swing-and support-related muscle actions differentially trigger human walk–run and run–walk transitions. *Journal of Experimental Biology*, 204(13):2277–2287, 2001. 1.1

[92] Marc H Raibert. Dynamically stable legged locomotion: progress report: October 1982-october 1983. 1983. 2.1.2

[93] Marc H Raibert. *Legged robots that balance*. MIT press, 1986. 2.1.2, 2.3.1, 3, 5, 5.2.3

[94] Annette J Raynor, Chow Jia Yi, Bruce Abernethy, and Quek Jin Jong. Are transitions in human gait determined by mechanical, kinetic or energetic factors? *Human movement science*, 21(5-6):785–805, 2002. 1.1

[95] 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. 2.3.1

[96] I Michael Ross. *A primer on Pontryagin's principle in optimal control*, volume 2. Collegiate publishers San Francisco, CA, 2015. 1.2.1

[97] Jonas Rubenson, Denham B Heliams, David G Lloyd, and Paul A Fournier. Gait selection in the ostrich: mechanical and metabolic characteristics of walking and running with and without an aerial phase. *Proceedings of the Royal Society of London-B*, 271(1543):1091, 2004. 2.1.3, 3.2.4

[98] Juergen Rummel, Yvonne Blum, and Andre Seyfarth. From walking to running. In *Autonome Mobile Systeme 2009*, pages 89–96. Springer, 2009. 3.2.2

[99] Donna Rutherford. IMG_6077, 2013. URL `https://www.flickr.com/photos/donna_rutherford/8727939197/`. Licensed under Creative Commons BY-ND 2.0. 1.1

[100] Harold Roberto Martinez Salazar and Juan Pablo Carbajal. Exploiting the passive dynamics of a compliant leg to develop gait transitions. *Physical Review E*, 83(6):066707, 2011. 2.1.3

[101] Uluc Saranli, William J Schwind, and Daniel E Koditschek. Toward the control of a multi-jointed, monoped runner. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 2676–2682. IEEE, 1998. 2.1.2, 3.3.2, 5

[102] Alexander Schepelmann, Michael D Taylor, and Hartmut Geyer. Development of a testbed for robotic neuromuscular controllers. *Robotics:*

Science and Systems VIII, 2012. 5.2.2

[103] William J Schwind and Daniel E Koditschek. Control of forward velocity for a simplified planar hopping robot. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 691–696. IEEE, 1995. 2.1.2

[104] Gorkem Secer and Uluc Saranli. Control of hopping through active virtual tuning of leg damping for serially actuated legged robots. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 4556–4561. IEEE, 2014. 3.1

[105] Veerle Segers. *A biomechanical analysis of the realization of actual human gait transition*. PhD thesis, Ghent University, 2006. 1.1, 3.2.4

[106] Veerle Segers, Peter Aerts, M Lenoir, and Dirk De Clercq. Spatiotemporal characteristics of the walk-to-run and run-to-walk transition when gradually changing speed. *Gait & posture*, 24(2):247–254, 2006. 3.2.5

[107] Veerle Segers, Matthieu Lenoir, Peter Aerts, and Dirk De Clercq. Kinematics of the transition between walking and running when gradually changing speed. *Gait & posture*, 26(3):349–361, 2007. 3.2.4

[108] Noboru Sekiya, Hiroshi Nagasaki, Hajime Ito, and Taketo Furuna. Optimal walking in terms of variability in step length. *Journal of Orthopaedic & Sports Physical Therapy*, 26(5):266–272, 1997. 3.2.5

[109] Andre Seyfarth and Hartmut Geyer. Natural control of spring-like running–optimized self-stabilization. In *Proceedings of the Fifth International Conference on Climbing and Walking Robots. Professional Engineering Publishing Limited*, pages 81–85, 2002. 3.4.2, 3.4

[110] Andre Seyfarth, Hartmut Geyer, Michael Günther, and Reinhard Blickhan. A movement criterion for running. *Journal of biomechanics*, 35(5):649–655, 2002. 2.1.2, 3.2.2

[111] André Seyfarth, Hartmut Geyer, and Hugh Herr. Swing-leg retraction: a simple control model for stable running. *Journal of Experimental Biology*, 206(15):2547–2555, 2003. 2.1.2, 5, 5.1, 5.1

[112] M Shahbazi, GAD Lopes, and Robert Babuska. Automated transitions between walking and running in legged robots. *IFAC Proceedings Volumes*, 47(3):2171–2176, 2014. 2.1.3, 3, 3.1

[113] Mohammad Shahbazi, Robert Babuška, and Gabriel AD Lopes. Unified modeling and control of walking and running on the spring-loaded inverted pendulum. *IEEE Trans. Robotics*, 32:1178–95, 2016. 2.1.3, 3, 3.6, 3.5.3

[114] Natan Shemer and Amir Degani. Analytical control parameters of the swing leg retraction method using an instantaneous slip model. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4065–4070. IEEE, 2014. 5, 5.4

[115] Panagiotis E Souganidis. Approximation schemes for viscosity solutions of hamilton-jacobi equations. *Journal of differential equations*, 59 (1):1–43, 1985. 3.3.2

[116] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and JW Grizzle. Embedding active force control within the compliant hybrid zero dynamics to achieve stable, fast running on mabel. *The International Journal of Robotics Research*, 32(3):324–345, 2013. 2.3.2, 3, 5, 5.4

[117] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. 2.2.1, 2.2.1, 2.2.1

[118] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999. 2.2.1

[119] Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1084–1091. IEEE, 2009. 2.3.2, 3

[120] Toru Takenaka, Takashi Matsumoto, Takahide Yoshiike, and Shinya Shirokura. Real time motion generation and control for biped robot-2 nd report: Running gait pattern generation. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1092–1099. IEEE, 2009. 2.3.2

[121] Toru Takenaka, Takashi Matsumoto, Takahide Yoshiike, and Shinya Shirokura. Real time motion generation and control for biped robot-2 nd report: Running gait pattern generation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1092–1099. IEEE, 2009. 3

[122] R Tedrake, TW Zhang, and HS Seung. Learning to walk in 20 min. In *Proceedings of the Yale workshop on adaptive and learning systems*, pages 10–22, 2005. 2.2.2

[123] İsmail Uyanık, Ömer Morgül, and Uluc Saranli. Experimental validation of a feed-forward predictor for the spring-loaded inverted pendulum template. *IEEE Transactions on Robotics*, 31(1):208–216, 2015. 5

[124] M Van Gurp, HC Schamhardt, and A Crowe. The ground reaction force pattern from the hindlimb of the horse simulated by a spring model.

Cells Tissues Organs, 129(1):31–33, 1987. 2.1.1

[125] Hado Van Hasselt and Marco A Wiering. Reinforcement learning in continuous action spaces. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pages 272–279. IEEE, 2007. 2.2.1

[126] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. 2.2.1

[127] Patrick M Wensing and David E Orin. High-speed humanoid running through control with a 3d-slip model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140. IEEE, 2013. 2.3.1

[128] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 2.2.1

[129] Albert Wu. *The Theory, Implementation, and Evaluation of Spring Mass Running on ATRIAS, a Bipedal Robot*. PhD thesis, PhD thesis, Carnegie Mellon University, 2017. 1.2.3

[130] Albert Wu and Hartmut Geyer. The 3-d spring–mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments. *IEEE Transactions on Robotics*, 29(5):1114–1124, 2013. 2.1.2, 3.3.2, 5, 5.1

[131] Albert Wu and Hartmut Geyer. Highly robust running of articulated bipeds in unobserved terrain. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2558–2565. IEEE, 2014. 5.1.1, 5.1.3, 5.2.2, 6.1.1

[132] KangKang Yin, Kevin Loken, and Michiel van de Panne. Simbicon: Simple biped locomotion control. In *ACM Transactions on Graphics (TOG)*, volume 26, page 105. ACM, 2007. 2.3.1

[133] Garth Zeglin. *The bow leg hopping robot*. PhD thesis, Carnegie Mellon University, 1999. 5, 5.4