Winter 2-2017

The Theory, Implementation, and Evaluation of Spring Mass Running on ATRIAS, a Bipedal Robot

Albert Wu
Carnegie Mellon University

The theory, implementation, and evaluation of spring mass running on ATRIAS, a bipedal robot

Albert Wu

February 28, 2017

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

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Hartmut Geyer, Chair
Christopher G. Atkeson
Koushil Sreenath
Jerry Pratt, IHMC

Abstract

We expect legged robots to be highly mobile. Human walking and running can execute quick changes in speed and direction, even on non-flat ground. Indeed, analysis of simplified models shows that these quantities can be tightly controlled by adjusting the leg placement between steps, and that leg placement can also compensate for disturbances including changes in the ground height. However, to date, legged robots do not exhibit this level of agility or robustness, nor is it well understood what prevents them from attaining this performance. This thesis begins to bridge the gap between the theoretical motions of simplified models and the implementation of agile behaviors on legged robots.

The state of the art allows room for improvement at the level of the simplified model, at the level of hardware demonstration, and at the level of theoretical understanding of applying the simplified model to a real system. We make progress on each of these facets of the problem as we work towards leveraging theory from the simplified model to generate effective control for locomotion on robots. In particular, spring mass theory has identified deadbeat stability for planar running, but it must be formulated in 3D to be applicable to a real system. We extend this behavior to 3D, adding deadbeat steering to the tracking of apex height on unobserved terrain. Running robots have yet to demonstrate the agile and robust behavior that the spring mass model describes; existing implementations do not target the deadbeat behavior. We apply state of the art control techniques to map the deadbeat stabilized planar running onto our robot ATRIAS, and we successfully demonstrate tight tracking of commanded velocities and robustness to unobserved changes in ground height. Despite this empirical proof of concept, it remains unclear how exactly the targeted behavior of the simplified model affects the closed loop behavior of the full order system. There are additional degrees of freedom which affect the tracking of original goals and additional layers of control which may offer other sources of stability. Furthermore, the hardware introduces perturbations and uncertainties which detract from the nominal performance of the full order model. To answer these questions, we formulate a framework founded on linear theory, and we use it to examine the contributions of each component of the control and to quantify the expected effects of the disturbances we encounter. This analysis reveals insights for effective control strategies for legged locomotion and presents a tool for scientific iteration between theory-based control design and evidence-based revision of the underlying theory.

Acknowledgments

Most of the work in this thesis was done in close collaboration with my lab mate William Martin. Without his dedication and expertise, none of these results would have been possible. His good nature has made solving potentially frustrating problems very enjoyable. I have learned a lot from him, and can only hope to have such colleagues in the future.

My advisor Hartmut Geyer has been a continuous source of motivation and inspiration. His enthusiasm for his work sparked my interest in this field, he taught me the fundamentals of research, and he always made himself available for discussion. Above all, I truly appreciate how he handled the challenging task of guiding my work to be scientifically relevant while leaving me the freedom to pursue the problems that I found interesting.

My committee members Chris Atkeson, Koushil Sreenath, and Jerry Pratt have all provided invaluable feedback throughout this study. Their collective knowledge in legged systems and controls has been essential in my journey to understand the basic principles in these topics. Likewise, Professors Steve Collins and Drew Bagnell have been very insightful and helpful. I would also like to thank Jonathan Hurst and the team at Oregon State University for providing us with the robot ATRIAS and for the many stimulating discussions on control strategies.

My other lab mates Alex Schepelmann, Seungmoon Song, Ruta Desai, Nitish Thatte, Jessica Austin, Akshara Rai, Yin Zhong, Helei Duan, Ben Shih, and Junlin Wang have also been excellent sources of information and discussion, and I have relied on their critical thinking and their patience to help me get all the nitty gritty details right. Moreover, their friendships have made these years of study truly pleasurable.

Finally, I am deeply grateful for the love and support from my wife Beverly. She has supplied endless positivity and encouragement, and I will always cherish the experience we shared making a life for ourselves on the East Coast. We truly appreciate the patience and support from our families during this extended period away from home.

Contents

1 Introduction **1**
- 1.1 Problem Statement . . . 1
- 1.2 Approach . . . 3
- 1.3 Contributions . . . 10
- 1.4 Thesis Overview . . . 10

2 Background **13**
- 2.1 Fundamentals of legged systems . . . 14
  - 2.1.1 Contact wrenches and leg placement . . . 14
  - 2.1.2 Stability and balance . . . 14
  - 2.1.3 Full actuation versus under-actuation . . . 16
- 2.2 Strategies for fully actuated systems . . . 19
  - 2.2.1 Position control . . . 19
  - 2.2.2 Inverse dynamics and force control . . . 20
- 2.3 Strategies for under-actuated systems . . . 23
  - 2.3.1 Optimal control for challenging dynamics . . . 24
  - 2.3.2 Leg placement as an additional input, and periodic stability . . . 25
- 2.4 Leg placement and simplified models . . . 27
  - 2.4.1 Leg placement on running robots . . . 27
  - 2.4.2 Inverted and Linear inverted pendulum model . . . 29
  - 2.4.3 Spring mass model . . . 31
  - 2.4.4 Other models for compliant legs . . . 34
- 2.5 Performance on real machines . . . 35
- 2.6 Summary . . . 36

3 Deadbeat spring mass running in 3D **39**
- 3.1 Background . . . 39
- 3.2 Method and results . . . 41
  - 3.2.1 Dynamics and Problem Formulation . . . 42
  - 3.2.2 Control Strategy Derivation . . . 46
  - 3.2.3 Control Performance in Simulation . . . 53
  - 3.2.4 Summary . . . 63
- 3.3 Conclusions and future work . . . 64

4 Overview of the ATRIAS robot **67**
4.1 Physical properties . . . 67
4.2 Fundamental assumptions, overall approach, and related background . . . 70
4.2.1 Sagittal plane instead of 3D . . . 70
4.2.2 Series Elastic Actuators (SEAs) as torque sources . . . 71
4.2.3 Optimal control for the higher order system . . . 74
4.3 Model and dynamics . . . 76
4.3.1 Floating base robot and dynamics . . . 76
4.3.2 Stance dynamics . . . 78
4.3.3 Flight dynamics . . . 80
4.3.4 Actuation and series elasticity . . . 81
4.4 Spring-mass gait design within ATRIAS's limits . . . 83
4.4.1 Maximum torque output . . . 84
4.4.2 Current-induced velocity limits . . . 87
4.4.3 Kinematic limits from hard-stops . . . 87
4.5 Summary . . . 88

5 Transfer of deadbeat spring mass running onto ATRIAS **91**
5.1 Background . . . 92
5.1.1 Spring mass theory in robots . . . 92
5.1.2 Technical implementation . . . 92
5.1.3 Problem formulation and decomposition . . . 93
5.2 Overview of control architecture . . . 93
5.3 Planning with the simplified model . . . 96
5.3.1 Modified spring-mass model . . . 96
5.3.2 Feasible hopping pattern . . . 97
5.3.3 Deadbeat speed control . . . 97
5.4 Stance leg control . . . 98
5.4.1 Model-based optimal control . . . 98
5.4.2 Inverse dynamics . . . 102
5.4.3 SEA control . . . 103
5.5 Swing leg control . . . 104
5.6 Estimation . . . 104
5.6.1 Kalman filtering . . . 105
5.6.2 Contact detection, stance vs. swing . . . 106
5.7 Model adaptation . . . 108
5.8 Experimental results and simplified analysis . . . 109
5.8.1 Undisturbed running . . . 110
5.8.2 Tracking SMM deadbeat velocity targets . . . 110
5.8.3 Simple analysis via simulation . . . 113
5.8.4 Rough terrain experiments . . . 114
5.9 Conclusions and unanswered questions . . . 117

6 Stability and performance through the combined action of the flight and stance controllers 119
6.1 Background . . . 120
6.1.1 Poincare analysis . . . 120
6.1.2 Optimal control for stability . . . 121
6.1.3 Hybrid zero dynamics for stable embedding of lower order model . . . 123
6.1.4 Summary and contributions . . . 125
6.2 Linear model of closed loop error dynamics . . . 126
6.2.1 Definition of states and errors . . . 126
6.2.2 Error dynamics in stance . . . 127
6.2.3 Error dynamics across flight . . . 128
6.2.4 Closed loop convergence . . . 130
6.3 Infinite horizon optimal control of stance phase . . . 131
6.3.1 Iterative re-design . . . 132
6.4 Applied examples for vertical system . . . 133
6.4.1 LTV-LQR in stance . . . 133
6.4.2 Using the landing height in flight . . . 136
6.5 Applied examples for the x-and-pitch system . . . 139
6.5.1 Finite horizon versus infinite horizon LQR . . . 140
6.5.2 LTV-LQR without leg placement . . . 143
6.5.3 Coupled states and potential revision of leg placement policy . . . 145
6.5.4 Open loop continuous control . . . 149
6.6 Discussion and conclusions . . . 150
6.6.1 Summary of contributions . . . 150
6.6.2 Relationship with HZD . . . 151
6.6.3 Relevance to other robots . . . 153
6.6.4 Unclear practicality . . . 153

7 Expected performance in the presence of noise and uncertainty 155
7.1 Overall approach . . . 156
7.2 Assumptions . . . 156
7.2.1 Bounded estimation error . . . 157
7.2.2 Bounded force error . . . 159
7.2.3 Bounded nonlinearities and other modeling errors . . . 161
7.3 Drift accumulated in one step . . . 162
7.3.1 Comparison of theory to experimental data . . . 165
7.3.2 Example applied to vertical states . . . 166
7.4 Deviations in flight . . . 169
7.5 Total state error accumulated across multiple steps . . . 171
7.5.1 Tools from linear theory: norms, weighting matrices, and the discrete Lyapunov equation . . . 172
7.5.2 Derivation of the invariant and attractive region . . . 173
7.5.3 Example applied to vertical states over multiple steps . . . 175
7.6 Summary and discussion . . . 177

8 Conclusions **179**
8.1 Big picture 179
8.2 Contributions 180
8.3 Potential tasks for future research 181

Bibliography **183**

List of Figures

1.1 Robust and agile running . . . 2
1.2 Conceptual diagram of performance via simplified models . . . 4

2.1 Contact wrench limitations . . . 15
2.2 Examples of full and under-actuation, wrench constraints. . . . 17
2.3 Acrobot model . . . 18
2.4 Inverted pendulum and linear inverted pendulum . . . 29
2.5 Spring mass model . . . 31
2.6 Deadbeat leg placement for planar SMM . . . 33

3.1 Definition of variables for 3D spring mass model . . . 43
3.2 Behavior function of 3D SMM parameterized by leg placement . . . 47
3.3 Deadbeat control law for 3D SMM . . . 50
3.4 Simulated running on rough terrain with 3D SMM . . . 54
3.5 Performance analysis of 3D SMM . . . 55
3.6 Stumbling margin of SMM . . . 57
3.7 Energetic analysis of sensitivity of 3D SMM . . . 60
3.8 Performance of deadbeat leg placement and classical leg placement . . . 62

4.1 bipedal robot ATRIAS . . . 68
4.2 Rigid body model for equations of motion. . . . 77
4.3 Centroidal dynamics . . . 81
4.4 Actuation system . . . 82
4.5 Rotor torque profile . . . 84
4.6 Tracking performance of actuators . . . 86
4.7 Domain and range of deadbeat running . . . 88

5.1 Overview of control architecture . . . 94
5.2 Prescribed vertical motion . . . 97
5.3 Kalman filtering of vertical state . . . 107
5.4 Photo of rough terrain experiment . . . 109
5.5 Baseline tracking performance: undisturbed, steady running at 1 m/s. . . . 111
5.6 Tracking of step changes in velocities. . . . 112
5.7 Sensitivity to force errors . . . 115
5.8 Coupling of states . . . 116

5.9 Ground height disturbance rejection 116

6.1 Closed loop evolution of state error 126
6.2 LTV-LQR design for vertical system. 134
6.3 Landing height adjustment. 137
6.4 Gains from finite and infinite horizon LQR. 141
6.5 Simulation of infinite-horizon LQR tracking deadbeat law 144
6.6 Simulation of LTV-LQR tracking of single non-adaptive limit cycle 146
6.7 Adjusting the deadbeat law for the full order model 148
6.8 Open loop continuous control with DLQR leg placement 150

7.1 Bounded Estimation error. 158
7.2 Force errors 160
7.3 Saturation of commanded GRFs for contact wrench constraints 160
7.4 Torque perturbations. 162
7.5 Maximum drift $|\boldsymbol{\Delta}_s|^{\max}(t_s)$ from one step 166
7.6 Tracking of vertical trajectory from experimental data 168
7.7 Convergence of vertical errors at takeoff. 175

List of Tables

3.1 Sensitivity of deadbeat leg placement and classical leg placement 58

Chapter 1

Introduction

1.1 Problem Statement

Hypothetically, legged robots can be highly mobile; running and walking behaviors can exhibit agile changes in speed and direction in a single step (Figure 1.1a), even on non-flat ground. To date, real robots have not yet achieved this level of performance. They often fall; they can only walk at low speeds and with conservative motions ; few can get off the ground to run or hop, and those that do can only do so within a limited range of gaits.

Part of the problem is that legged locomotion is fundamentally not a straightforward control task. The motion of the system is an outcome of the net forces and moments, or the net wrench, applied by the feet against the ground. The nature of interaction between feet and the ground limits the domain of wrenches that are possible. The normal (perpendicular to the surface) forces can only push away from and never pull towards the ground. Based on the shape of the contact surface and the available actuation, this phenomenon also translates into a maximum rotational moment applied to the foot [59]. Furthermore, forces directed along the plane of contact are generated through friction and are limited by the coefficient of friction. These limitations on the net wrench constrain the range of possible behaviors. Finally, legged locomotion is characterized by intermittent contacts; each foot is periodically lifted off the ground and re-positioned. Reasoning

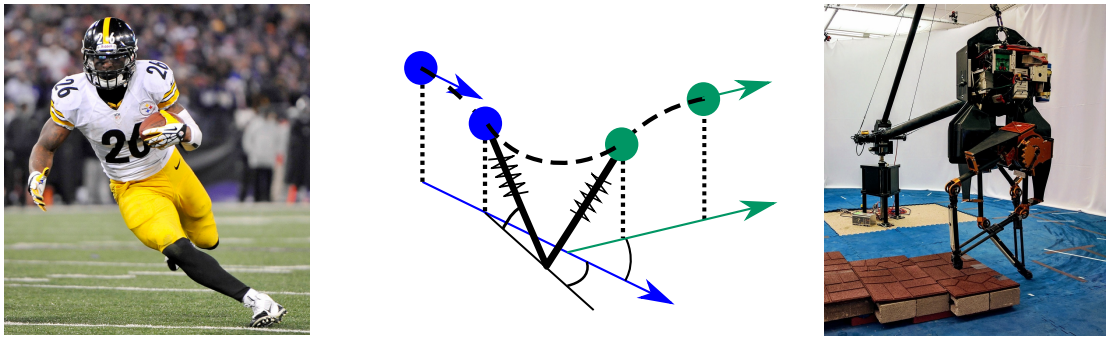


Figure 1.1: Agile and robust running. In (a), an athlete exemplifies how running can include agile changes in speed and direction in a single step. Indeed, in (b) the spring-mass model (an abstraction described by a point mass bouncing on a massless spring) can reproduce these behaviors by controlling the placement of the leg at each step. This thesis explores how such behaviors developed at the level of the simplified model can be implemented to control running on a higher order robot like ATRIAS in (c). Photograph in (a) used under fair use conditions.

effectively about the intermittent contacts and the resulting constraints on the net wrench poses a challenging planning problem.

Simplified models are a popular strategy for approaching this challenge (Figure 1.1b). These models use simplifying assumptions that implicitly obey the wrench limitations to approximately describe the dynamics of the system using a reduced number of states and parameters. This process allows for a more tractable planning problem. For example, representing the system as a single mass attached to a rigid leg or to a linear spring resolves the future motion of the system from any given initial condition. The leg placement can then be chosen accordingly to generate desired motions from step to step. However, for this behavior to carry over to a full-order robot, the robot must reproduce the assumed dynamics on the actual system while sufficiently addressing any modes omitted by the simplified model. This control task is made even more challenging by the disturbances and uncertainty encountered in hardware.

Specifically, the planar spring mass model can use leg placement to nearly perfectly track running gaits on unobserved, uneven terrain [47, 169, 171]. While bipedal robots like ATRIAS (Figure 1.1c) with light, compliant legs have been built [61, 149] to approximately match the

characteristics of this idealized system, controllers designed around the simplified model have yet to translate into hardware demonstrations of the modeled behavior, nor is it well understood to what degree the behavior is limited by the omitted modes and by the realities of the hardware platform.

This thesis begins to bridge the gap between the theory of simplified models and the implementation of effective locomotion on legged robots. Here, we extend the theoretical capabilities of the spring mass model to 3D; we implement established control techniques to demonstrate planar spring mass running on a real robot; and we analyze the resulting system to understand the effects of the spring mass based leg placement, of the closed loop feedback, and of the errors and uncertainties encountered in hardware. In particular, the framework we develop sets up quantitative comparisons between theoretical expectations and experimental data, which allows iteration between theory-based control design and evidence-based revision of the underlying theory.

1.2 Approach

We want to determine to what extent the simplified spring-mass model can be leveraged to achieve robust and agile running on the real robot. This query implies three separate questions:

1. What performance is possible within the simplified model?
2. What performance does that translate into for a full order system with perfect control?
3. How is this performance limited by the noise and uncertainty of the hardware platform?

In pursuing these questions, we develop insights that advance the overall understanding of how to achieve robust and agile locomotion on legged systems and what fundamentally limits their performance. In this section, we first present a conceptual diagram in Figure 1.2 to illustrate the context of these three questions and the overall problem statement. Then, we elaborate more on our approach to answering each question.

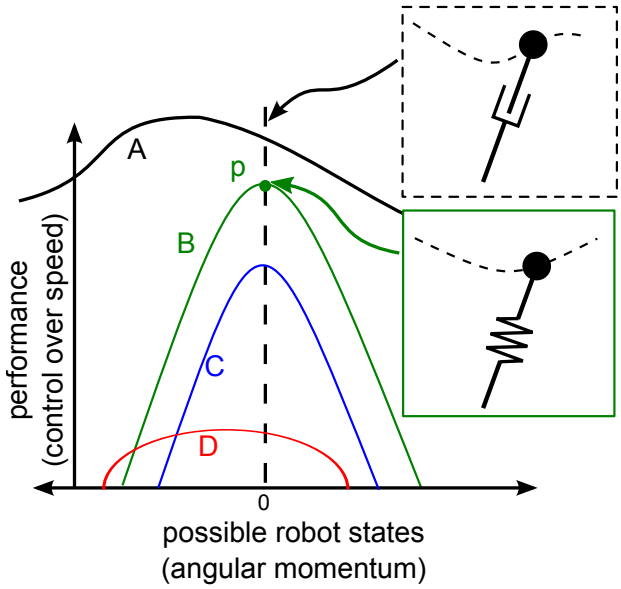


Figure 1.2: Conceptual diagram of exploring the performance of the robot via simplified model. Solving for the theoretical maximum performance A is intractable. A simplified model can offer a solution p for a subset of the full space. Control that tracks the simplified model results in performance B in a perfect system, but noise and uncertainty limits the performance on real hardware to curve C. The current state of the art D in hardware falls short of the point p known in literature, but the gap is not well understood. We propose to identify C for a specific simplified model and validate it in hardware to elevate D.

Decomposing the problem

Figure 1.2 visualizes these components of the problem statement. The vertical axis measures the performance of a controller; for example, this could be the ability of the robot to regulate its speed within physical limits, measured as the amount of correction it can apply in a single step. The horizontal axis represents the possible states of the full-order robot. To simplify the visualization, we sketch a single parameter for these states, which could be the angular momentum of the robot about the center of mass (CoM). The ability of a controller to regulate the robot's speed generally varies across the possible states.

With unlimited computation, a perfect model predictive controller would give the best possible performance from any state, represented by curve A. Finding and achieving this performance is currently computationally intractable. However, identifying control for a simplified model provides some non-optimal solution p for a specific subset of robot states. For example, any point-mass model only describes behaviors with zero angular momentum about the center of mass, which limits the analysis to the dashed vertical line. Imposing additional simplifying assumptions (for example, a specific force profile) results in lower performance than the general optimal solution within these states, which is typically still too hard to find. In our first question, we seek to find the height of point p for the spring mass model. Because it exists only on the dashed line, point p alone does not offer a sufficient description of how well the controller works on the robot. In our second question, we seek to extend the point p into the curve B, which describes how well the controller functions away from the nominal conditions. This curve peaks at point p if the controller is designed to track the behavior of the simplified model, even though the hypothetical optimal performance A may peak elsewhere. Finally, under realistic assumptions about noise and uncertainty, the performance curve B is lowered to curve C. In our third question, we seek to define C theoretically and to validate it experimentally on our hardware platform. This demonstration results in elevating D.

In the existing literature, various controllers proposed for simplified models have offered

promising behavior in terms of point p, while real running has been demonstrated only at much lower levels (curve D). However, the current literature has not sought to quantify how applicable these strategies are by identifying curves B or C. The state of the art hardware performance D exists empirically, but these techniques have not offered a comparison to a theoretical expectation. This gap between the theory and the hardware makes it difficult to reason about the steps required towards attaining the next level of performance.

Behavior of the simplified model

We first explore the theoretical capabilities of leg placement control within the simplified model of a spring mass system, which is suitable for our robot with light, compliant legs and passive ankle joints. Existing literature has shown that leg placement policies can be designed for the planar spring mass system to generate highly robust running on rough terrain [169]. To describe a freely running system with no external support, we need to extend the behavior to 3D. Using the return map formulation frequently applied to examine the stability of periodic orbits [185], we compute the stride-to-stride transitions of the point-mass's motion in 3D as parameterized by the placement of the leg. Interpolating over the transition map produces a policy for matching a target speed and a target direction motion in a single step, which we then extend to reject disturbances from unobserved uneven terrain. Physical limits such as the maximum torque, maximum motor speed, and the coefficient of friction are translated into conditions on the dynamics of the simplified model (as is done for the planar model in [47]), and we identify the feasible domain of the proposed policy.

Transferring to the higher order system

Because the reduced order model does not account for all the degrees of freedom in the higher order system, additional control must be designed to address these modes. In our case, the inertial effects of the robot's light legs are negligible, and the key omission of the simplified model is

the robot's rotational motion. As a point mass model, the spring mass system describes the translational motion of the robot's center of mass (CoM) while intrinsically implying constant angular momentum.

Control based on stabilizing the hybrid zero dynamics of a system [135, 182, 206] offers a formal solution to this type of problem via design of virtual constraints and applying nonlinear control. Instead of applying this methodology, we use a conceptually similar approach that leverages powerful results from linear theory, allows us to use well established control techniques for reference following, and provides more explicit access to the deadbeat-stabilized simplified model. We interpret the output of the closed loop simplified model as a reference motion given by a specific leg placement in flight, a corresponding CoM trajectory in stance, and a fixed body orientation with zero rotational motion. We then apply optimal control to follow this reference in the stance phase; after linearizing the dynamics about the reference trajectory, we solve for the minimum changes in control input that best drive down the weighted error across all the degrees of freedom. More detailed formulations can include actuator limits as well as contact wrench limits as constraints in the optimization. Such techniques have been widely applied on walking robots [51, 68, 101, 104, 125] but have not yet been used to achieve running. We use the structure of the closed loop dynamics and the emergent cost function to quantify the theoretical behavior of the higher order system in terms of the overall rate of convergence as well as how the errors are traded off between different degrees of freedom.

Practical limitations

Lastly, we quantify the performance we expect to attain given our theoretical understanding of the problem and the properties of the real hardware. The nominal convergence of the tracking error is based on an assumed model of the full robot's dynamics and a perfectly generated control input. In reality, the true dynamics of the system do not perfectly match the assumed equations of motion, and the control input (in our case, ground reaction forces) cannot be directly ma-

nipulated. These effects can be treated as perturbations to the nominal closed-loop system. By assuming bounds on the sources of error based on theoretical analysis as well as empirical data, we can then compute the associated bounds on the expected resulting state error in each step as well as the bounds on the accumulated state error after multiple steps.

This process results in a quantified theoretical benchmark for performance. Comparing the computed bounds with the observed state error in experimental data allows us to evaluate the validity and applicability of the theory-based control design and to judge the success or failure of the hardware implementation. It also highlights the most promising next steps, whether we most crucially need a more descriptive model, improved actuator control, or a revised control architecture. While the conservative nature of these bounds may limit their descriptiveness of the actual performance, this formulation nonetheless serves as a first step towards analytically comparing theory and empirical results.

Framework built upon trajectory tracking and linear theory

Linear theory facilitates understanding how the continuous feedback in stance, the leg placement in flight, and any encountered disturbances or uncertainty all contribute to the nominal and observed behaviors of the full order system. Specifically, we study the evolution of the state error with respect to the dynamically defined reference trajectory. In stance, the continuous feedback linearizes the error dynamics, and this linear time-varying system can be integrated into a discrete linear propagation of the state error at touchdown to the ensuing takeoff. Bounded perturbations integrate into a bounded drift in this transition model. Likewise, in flight, the re-planning of the reference and the associated leg placement also define a discrete transition between the state error at takeoff and at the ensuing touchdown. By linearizing this error transition in flight and propagating the effects of non-linearities and other disturbances, we can then directly compose the expected step-to-step error evolution of the real system using matrix multiplication.

This formulation provides a clear framework for deriving stability and evaluating perfor-

mance: it allows us to integrate a variant of the Ricatti equation to derive infinite-horizon optimal control in stance, it factorizes the Poincare stability analysis into flight and stance components, and it allows us to apply known techniques to define attractive and invariant regions in terms of state error as a function of the assumed bounded perturbations. We use these results to quantify comparisons between theory and experimental data and to form hypotheses that generalize to the effective control of legged systems.

In principle, our approach shares many underlying concepts with the hybrid zero dynamics (HZD) framework that has demonstrated and formalized stable running on under-actuated hardware [135, 182, 206]. Both formulations focus on deriving stability by leveraging model reduction techniques and applying local feedback to track the reference behavior, and both formulations utilize a combination of discrete and continuous feedback controllers. In terms of significant differences, here we track target trajectories while HZD embeds target dynamics, and here we start with the simplified model and discrete control while HZD starts with the full order system and the continuous control. Throughout this thesis, we analyze the implications of the similarities and differences between the two perspectives.

Summary

Based on previous studies of the spring mass model, we (and other researchers) hypothesize that the spring mass model can translate into agile locomotion on robots with light and compliant legs. However, this behavior has yet to be demonstrated on real robots, and furthermore, there are still important gaps in the theoretical understanding of how the simplified model applies to a real system. In this thesis, we approach the problem from both directions. On the hardware side, we demonstrate the agility and robustness of planar spring mass running on our robot ATRIAS. On the theory side, we decompose the problem into three separate investigations: we extend the applicability of the simplified model to 3D, we examine the implications of additional degrees of freedom and additional layers of control on a full order model, and we quantify the expected

effects of bounded perturbations to our nominal system. We conduct the latter two studies in a simple framework that leverages powerful tools for linear systems. In the end, this gives us an improved understanding of how to design effective control, what kind of performance we can realistically expect on a real system, and which directions of future work may deliver the largest improvements to the state of the art implementations.

1.3 Contributions

Our goal is to bridge the gap between the demonstrated performance of running robots and the agile theoretical behaviors derived from simplified models. Specifically, this work makes the following contributions to the state of the art:

- extend the capabilities of the spring mass system to generate robust and agile running in 3D on unobserved, uneven terrain;
- implement planar spring mass running on hardware to demonstrate deadbeat tracking of target velocities and robustness to unobserved, uneven terrain;
- clarify the theory behind how the design of the full order control generates an expected closed loop behavior;
- formulate quantified theoretical expectations of performance on a noisy system that are then compared to experimental data.

1.4 Thesis Overview

Chapter 2 reviews the existing literature on controlling legged robots, with an emphasis on stability and on running gaits. Chapter 3 presents the extension of the deadbeat stability of the spring mass model to running in 3D. Chapter 4 introduces the ATRIAS robot as our hardware platform and examines its theoretical capability of executing spring mass gaits. Chapter 5 details

our control implementation for planar running with deadbeat velocity tracking and presents our experimental results. Chapter 6 then presents a theoretical analysis of how the stability of the system is a product of the continuous feedback applied in stance and the discrete motion planning performed in flight. Chapter 7 presents a theoretical analysis of how the noise and uncertainty we observe affect the expected performance. Finally, Chapter 8 summarizes the results of these studies and explores the implications for future work towards achieving agile running on robots.

Chapter 2

Background

This thesis builds upon the existing work of many researchers in robotics. In this chapter, we present a broad review of the relevant studies that precede our work. First, Section 2.1 sets up the rest of the discussion by characterizing the basic components of legged locomotion, introducing the concept of stability, and distinguishing between full and under-actuation. Section 2.2 reviews control strategies commonly used by fully actuated robots, including position control and force-based momentum control. Section 2.3 then presents methods that are appropriate for stabilizing under-actuated systems (which are also highly applicable to and often used on fully actuated systems), including optimal control and intermittent control for periodic systems. In particular, in this context, leg placement emerges as a highly effective control strategy. Section 2.4 then reviews the use of leg placement in control design, with an emphasis on placement strategies that have emerged from studies of simplified models of the system dynamics. Finally, Section 2.5 discusses existing treatments on how model-based control designs carry over to real hardware implementation. The main motivation for the work in this thesis is the gap between the agile and robust running the simplified models can produce and the behaviors that have been demonstrated on real hardware systems.

2.1 Fundamentals of legged systems

2.1.1 Contact wrenches and leg placement

One way to interpret legged locomotion is to view it through two behaviors; the feet are positioned into contact with the ground, and forces and moments are then applied to the system by pushing off the ground through these contacts. The limited nature of these contacts makes control a challenging problem: typical feet can only push away from the ground and can never pull towards it; the lateral forces they can apply are limited by friction; and the torques they can apply are limited by the size of the foot and the availability of ankle actuation (see Figures 2.1).

In the book [173], Siciliano et al characterize this challenge a little more technically: "... the global displacement and orientation of the robot can only be realized through contact forces. Moreover, this motion is necessarily associated with a change of posture. More generally, the system can achieved a desired movement if and only if the total wrench of gravity and the contact forces is equal to the dynamic wrench of the robot. For a given control scheme, the question is then to check whether this property is satisfied by the robot under control."

The total change in linear and angular momentum of a system is equal to the net wrench (net force and net moment) applied, and at any moment in time, only a limited set of net wrenches is possible, depending on the relative position of the feet to center of mass. Due to this dependence, motion generation is a product of choosing appropriate footing, and then applying feasible wrenches to achieve desired accelerations.

In this thesis, we study how agile and robust running is a product of control strategies for both leg placement and wrench generation subject to the contact constraints.

2.1.2 Stability and balance

Stability is a central concept throughout this thesis and features prominently in this background chapter. Here, we will first briefly define what it means to stabilize a standing, walking, or

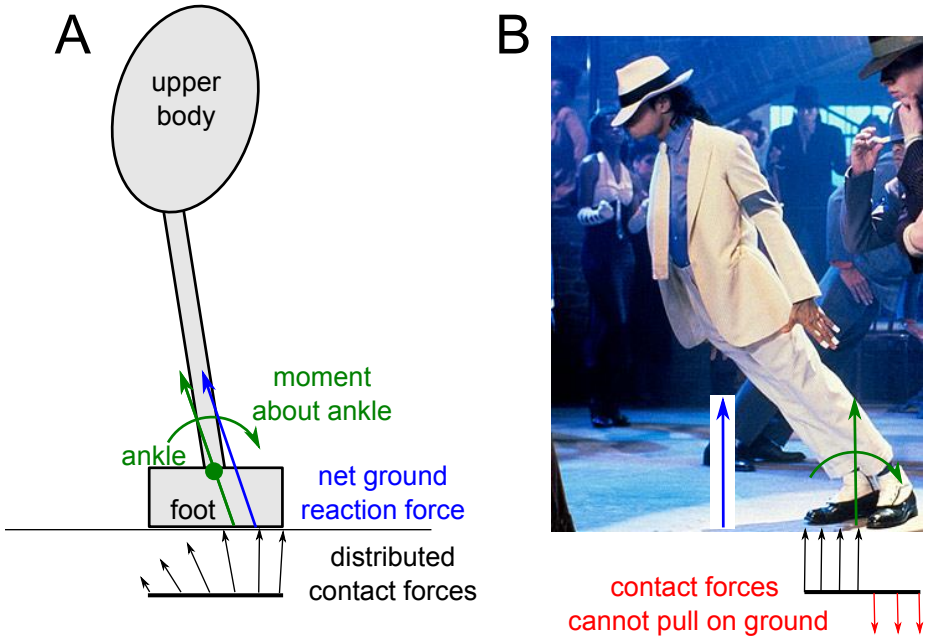


Figure 2.1: (A). Contact wrench limitations. The net ground reaction force (blue vector) is the sum of all the forces (black vectors) across the surface area of the foot. These forces can only have positive vertical components; it is impossible to pull on the ground. The origin of the blue ground reaction force (GRF) is placed at the center of pressure, which is the point at which the single GRF vector equivalently acts with a moment couple that is exactly zero about the horizontal axes. Moving the GRF to a different point on the ground implies a non-zero moment couple (in green, putting the GRF through the ankle comes with a non-zero ankle torque). The center of pressure is always aligned with the ankle if the ankle joint is passive and produces no torque. The unilaterality of the contact forces imply that the center of pressure never leaves the footprint. Friction constraints imply that the GRF vector never tilts past a certain angle from the vertical. (B). An example of a violation of the contact wrench limits. The dancer has leaned past his feet, and this pose can only be held by applying a high torque at his ankles. Equivalently, the center of pressure for the blue moment-free GRF is placed under his center of mass. The required torque can only be generated if his heels pull on the ground, which is usually impossible. Here, the dancer has fasteners in his shoes that do in fact pull on the ground to generate the red GRFs. Legged systems usually cannot violate this constraint. Photograph used under fair use conditions.

running robot.

Stability is most intuitive when a robot standing still. The robot is balanced if it stays upright and does not fall over. This can only happen when the net forces and torques acting on it all cancel to zero; the robot is in equilibrium. The robot is stably balanced if recovers this equilibrium when perturbed by external pushes. That is, if state errors get smaller over time, the robot is stable. If instead errors tend to grow, the robot is unstable and eventually falls after a perturbation.

For a robot in motion, stability can still be viewed in terms of errors, despite not being in an equilibrium condition. For example, if we want the robot to walk or run at a certain speed and hold a certain posture, we can examine how errors in these states evolve. If the robot goes faster and faster or leans farther and farther forward, the behavior is unstable; the errors will build up until the system fails. On the other hand, if the robot can bring its speed and orientation to the desired values from perturbed conditions, the overall motion is stable.

Having defined stability, we can now talk about robustness and agility. Robustness refers to the ability of the robot to recover from large perturbations; if the robot steps off of a curb, we want to study how quickly it can bring the state errors back to zero. Similarly, agility refers to the robot's ability to track changes in the desired motion; we may ask the robot to speed up or to turn, and we are interested in how quickly it can respond to these commands and bring the relative errors back to zero. Overall, this thesis focuses on how theories developed from the dynamics of a simplified model can be translated into highly robust and agile running on a real machine.

2.1.3 Full actuation versus under-actuation

By definition [79, 178, 193], in a fully actuated system, each degree of freedom can be independently controlled. As a result, it relatively straightforward to design stabilizing feedback controllers that drive the system to the desired configuration, thereby reducing all errors to zero. Using special shoes, the dancer in Figure 2.2A can use his ankle muscles to bring himself to an arbitrary lean angle. To a more limited extent (determined by the contact wrench constraints),

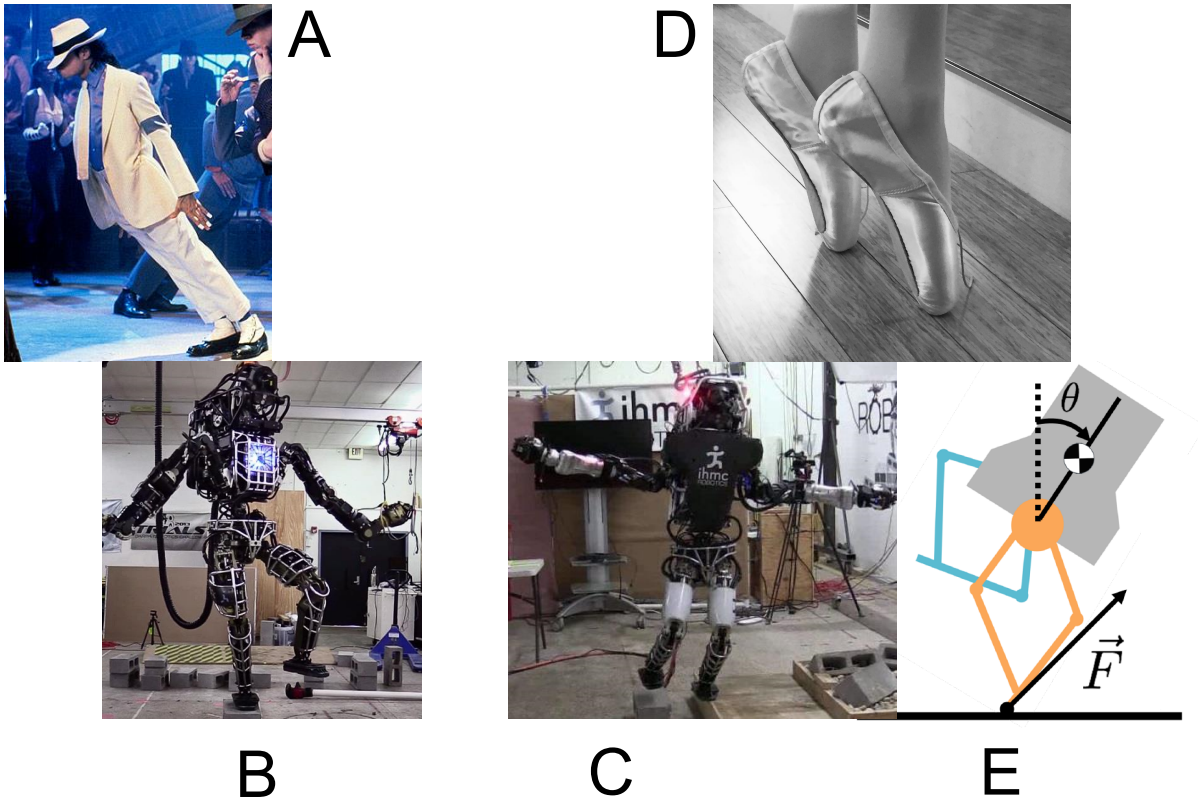


Figure 2.2: Examples of full actuation and under-actuation. The dancer in A and the IHMC Atlas robot in B are both fully actuated. The dancer's shoes are fixed to the stage and eliminate the contact wrench constraints, while the robot's "full actuation" must still respect these limits. Local balance strategies take advantage of the available ankle torque as long as the center of pressure is kept within the footprint. The ballerina in the en pointe pose in D, the same IHMC Atlas robot (designed and built by Boston Dynamics) standing on an edge in C, and the ATRIAS robot in E are all under-actuated. These systems cannot generate torques against the ground and must rely on more complex strategies for maintaining stability. Photograph in A used under fair use conditions. Photographs B and C used with permission. Photograph D used with permission; model is Presley Parsons, photo by Laura Parsons.

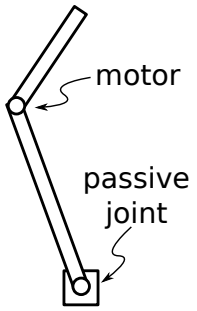


Figure 2.3: The Acrobot model consists of a series of two linked bars attached to the ground with a passive pin joint. The angle between the bars is driven by a motor. This system is under-actuated due to the lack of a motor at the pin joint with the ground. Despite the under-actuation, balance about the upright equilibrium can be maintained with a variety of strategies, including linear optimal control.

Institute for Human and Machine Cognition's (IHMC) Atlas robot (designed and built by Boston Dynamics) in Figure 2.2B can use actuators in each of its joints to reach a desired posture. Many of the successful walking and running robots explicitly or implicitly rely on full actuation in their control design. We first briefly review these control strategies, as some of the concepts encountered carry over to control of under-actuated systems.

In contrast, some platforms (including ours, the ATRIAS robot [61, 149]) are under-actuated. Specifically, ATRIAS is under-actuated at its feet; the ankles generate no torque (Figure 2.2E). Similarly, the ballerina in the en pointe pose in Figure 2.2D has no means of generating torques against the ground. Likewise, IHMC's Atlas in Figure 2.2C is standing on a line and cannot apply torques in the frontal plane. This makes stability a more challenging control problem [79, 178, 193]. Naturally, the strategies used for stabilizing under-actuated systems are still highly relevant for improving the robustness and the performance on fully actuated robots, and indeed many of the fully actuated robots demonstrate effective use of these strategies (as evidenced by Atlas in Figure 2.2C). Nonetheless, in this chapter we use the distinction between full and under actuation to organize the review of the relevant literature.

2.2 Strategies for fully actuated systems

2.2.1 Position control

Specifically, many robots – including the runners QRIO [121], ASIMO [188, 189, 190, 191], HRP-2LR[86, 87], HUBO [30, 129], and the Toyota robot [187] – are actuated by high accuracy joint servos that ultimately track individual joint positions and velocities with PID control. While the low level feedback control is straightforward, maintaining balance then becomes more of a motion planning problem. In higher layers, the controller is tasked with computing reference joint trajectories describing a desired motion that is consistent with the dynamics of the system. That is, specifying all the joint trajectories fully defines a hypothetical motion of the system, where the associated linear momentum, angular momentum, and ground contact wrench all need to be consistent. If the computed zero moment point (ZMP) – or equivalently the foot rotation indicator (FRI) [59] – ever leaves the boundaries of the foot, the ground contact cannot apply a large enough rotational moment to match the inertial effects of the specified motion. Thus, the foot would roll onto an edge, thereby breaking the assumed static contact, and the nominal joint trajectories then imply a different overall motion, which often corresponds to the robot falling over despite all the joint errors being driven to zero. (Note that the ZMP as commonly defined [201] is equivalent to the center of pressure, and physically it can never leave the footprint. The ZMP stability criteria seen in the literature [83, 187, 188] refers to the computed ZMP associated with the desired motion).

As a result, many of the position controlled robots operate around ensuring that the computed ZMP of the reference joint trajectories stay near the center of the foot. These trajectories are often generated by solving inverse kinematics after obtaining a ZMP-consistent CoM motion from a simplified model [83]. In place of general inverse kinematics, resolved momentum control [84] can be used to map center of mass and foot references to individual joint motions [30, 84, 121, 187, 188]. However, directly tracking these joint references with decentralized PID control in

each joint is often insufficient for stabilizing the robot. These computations only account for the dynamics of the nominal motion plan and do not provide any information about the feasibility of corrective motions – the actual wrenches applied to the foot while the joint trajectories are being driven to the references differ from the nominal wrenches, and FRI may leave the foot and destabilize the system. Keeping the computed ZMP away from the edges of the foot provides margin for small corrective actions.

Many tools and extensions [30, 83, 86, 87, 129, 187, 188, 189, 190, 191] have been applied to improve the stability of position-based control in legged systems and have led to demonstrations of stable running gaits on fully actuated robots. In principle, these strategies generally modify the planned reference motion with more detailed considerations of the full system's dynamics such that errors and disturbances are handled more explicitly before being passed to the local joint servos.

2.2.2 Inverse dynamics and force control

More recently, many roboticists have applied inverse dynamics and force control techniques to improve the balancing capabilities of humanoid robots [37, 68, 116, 153, 154, 155, 167, 179, 203]. As opposed to position control methods which only explicitly consider the dynamic feasibility of the reference motion, force control and inverse dynamics continuously solve the equations of motion, thereby providing constant access to the ground contact wrenches. These equations explicitly map between the forces and torques applied at each joint and the resulting rigid body accelerations of the system, and this then allows dynamically consistent feedback control to be generally posed as a constrained optimization problem – one can search for the applied inputs that best achieve desirable accelerations within the feasible contact constraints.

More specifically, optimization-based inverse dynamics control has proved to be quite successful in balancing and walking tasks on bipedal robots [51, 68, 101, 104, 125]. Operational space control [92] (or task space control) defines tasks as desired accelerations and solves the

equations of motion of the full system to find the torques that best achieve the tasks. For example, these accelerations can be a combination of feed-forward and feedback for tracking a specified motion of the CoM or of the swing foot. Using the centroidal momentum [122] matrix further allows angular momentum tracking to be expressed as accelerations in the task space. The task space formulation simplifies the expression of motion control; compared to needing to design coordinated references trajectories for every joint, only the explicit motion goals are expressed as tasks, while the optimization distributes the motion to the individual joints.

Many variations in the detailed formulation of the problem have been employed. Tasks can be layered in hierarchies such that lower priority ones are addressed only in the null space of higher priority ones [35, 116, 130, 131, 153, 168]. Alternatively, they can all be assigned weights to trade off between objectives [35, 50, 51]. [68] reduce the size of the problem by using the structure of the equations of motion to eliminate the joint torques from the set of decision variables. Without considering the limitations of the ground contact wrench, the inverse dynamics can be solved efficiently as an unconstrained optimization [77, 116, 152]. Instead, the wrench constraints can be approximated with linear inequalities, and the optimization problem is reformulated as a constrained quadratic program (QP): tasks are tracked as closely as possible within feasible dynamics [36, 102, 156]. [204] gives an alternative formulation as a conic optimization with reduced variables and constraints.

The QP task space formulation of force control was used extensively by successful robots (MIT, CMU, and IHMC Atlas robots, among others) at the DRC finals [51, 101, 104]. It serves as a general tool that computes the joint actuation given a set of tasks to optimize, and the "art" of the control design becomes choosing and prioritizing the right tasks for stable locomotion. IHMC [101] tasks include tracking of the instantaneous capture point and the centroidal momentum of the system. CMU [51] tasks include tracking of the centroidal momentum and compensating for a dynamically estimated disturbance force. MIT [104] tasks include tracking body frames attached to various points on the robot. For simulated running, prioritized task space control on a

simulation of a 3D humanoid in [203] tracks foot trajectories to embed the 3D-SLIP derived leg placement at a higher priority than other stabilizing tasks and demonstrates stable running with good disturbance rejection.

The centroidal dynamics [122, 123] of a system describe the acceleration of the center of mass and the rate of change of the total angular momentum. Many control designs ([33, 51, 67, 101, 103, 104, 107], among others) have leveraged the straightforward relationship between the centroidal dynamics and the net external contact wrench, as this mapping remains valid regardless of the specific robot parameters or configuration. At its core, the task of balancing a robot can be expressed as stabilizing the centroidal dynamics, while optimization and inverse dynamics algorithms for whole-body-control must address how exactly the centroidal motion is distributed to each joint and individual rigid bodies. Controllers can thus be designed to prioritize generating contact wrenches that feedback stabilize the center of mass motion and total angular momentum.

Finally, at the level of defining feedback policies to stabilize the centroidal dynamics, these controllers implicitly rely on the full actuation of the robot. That is, the desired net wrench is typically expressed as independent PD laws on each of the 6 floating base degrees of freedom, or subset of these if some of the modes are close to stationary (most implementations assume the angular momentum is zero, and thus command a desired net wrench with no rotational moment). Even when some of the degrees of freedom are zeroed, the underlying assumption they can all be set independently remains the same. This assumption is explicitly abandoned when the contact wrench constraints are active, but it is still typically implicit in the feedback design.

[208] demonstrates a departure from this implicit assumption when using angular momentum strategies on Atlas to restore balance when necessary (as measured through the instantaneous capture point – see Section 2.4.2 – versus the footprint). This mode of operation gives high priority to the angular momentum tasks, effectively removing the "full actuation" of the linear momentum. This control allows Atlas to temporarily solve the under-actuated problem of balancing on a line in Figure 2.2(C). [71] also applies inverse dynamics and motion planning for

an under-actuated system, formulating the concept of flow tubes to design stabilizing policies within the dynamic limitations.

Virtual model control

Instead of solving the full equations of motion for inverse dynamics, virtual model control [39, 141, 184] techniques uses the Jacobian matrices from the kinematic relationship between actuator and end effector to approximate the mapping between applied forces and joint torques. The end effectors can then be made to re-create the behavior of virtual elements (like springs and dampers) that would lend stability to the system. This design process can generate simple and intuitive stabilizing feedback for individual modes, but since the virtual components must be consistent with the feasible outputs of the real actuators, virtual model control on its own often implicitly relies on full actuation.

2.3 Strategies for under-actuated systems

Under-actuated robots must use more involved methods to remain stable. For example, if a robot on point feet were to track the COM position with PD control, it then has no way of controlling the angular momentum. In particular, ankle-less robots are fundamentally similar to the Acrobot model [49, 106, 177, 193], described by a double-link pendulum with an actuator between the links and a passive pin joint at the base (Figure 2.3). Despite the under-actuation, this system is controllable – it can balance itself about the upright equilibrium pose, and the coupling between the degrees of freedom plays a critical role its stabilization. The horizontal location of the center of mass is responsible for maintaining the orientation of the robot, and single-instant-in-time optimization (task space control) of independently designed tasks in these two modes can easily lead to unstable control. The collection of works in [19, 20, 49] present comprehensive studies on the balancing problem for this class of robot and includes a comparison of linear and non-linear control strategies.

2.3.1 Optimal control for challenging dynamics

Model-based optimal control [3, 9, 26] can address this issue of stability of an under-actuated system. In general, optimal control poses the problem as a search for the execution that brings the system to the desired equilibrium configuration or to the desired reference trajectory while incurring the minimum total cost. The cost is typically defined as the weighted total error across all states and the total applied actuation, both integrated on a potentially infinite time horizon that looks into the future. If an infinite-horizon solution exists, the full system will be stabilized. On a finite horizon, the resulting control minimizes the total state error and effort applied. In this discussion, there is an important distinction between "optimal control" and the optimization used by the task-space algorithms discussed earlier. The optimal control formulation explicitly considers the future states of the system, where the task-space optimization finds the best match for the defined tasks at a single instant in time. The key observation here is that long term stability would need to then come from appropriate choices of "tasks," and full actuation facilitates this step of the design process.

For example, linear quadratic regulation (LQR, the optimal control solution for unconstrained linear systems with quadratic costs) is solved by integrating the Ricatti equation, which returns the optimal control as a linear feedback law. This controller can also be extended to locally stabilize nonlinear systems about the reference condition. For example, LQR applied to Acrobot about its upright equilibrium automatically generates a linear feedback law that guarantees local stability, where the design process implicitly handles the coupling between the horizontal position and the dynamics of the orientation. In addition to stabilizing equilibrium poses, LQR readily extends to tracking reference trajectories in a feed-forward plus feedback architecture [17]. Beyond stabilizing challenging under-actuated systems, optimal control is a powerful tool for effective control design even on fully actuated systems [51, 104]. For example, despite the availability of ankle torque, identifying policies that intentionally minimize the required torques can still be beneficial. Low ankle torques correspond to motions that keep the center of pressure

away from the boundaries of the foot, thereby leaving a margin against instability.

While this "search for the fastest and cheapest convergence of all the state errors given the dynamics of the system" is very powerful as a generalizable problem statement, the solution generally cannot be expressed as a closed form feedback law. Model predictive control [3] iteratively re-solves the problem numerically, but this process is computationally expensive and slow for high order models. A very wide variety of techniques have been developed to improve the tractability of solving the optimal control problem, and within the legged locomotion community, popular strategies include using reduced order models for motion planning [66, 83, 183, 187], solving approximations to the optimal costs and policies that leverage linear or quadratic structure [29, 32, 51, 62, 79, 103, 104, 182, 194, 207], and handling the constraints in a separate step (applying QP-based constrained inverse dynamics) after identifying an approximate optimal cost function that ignores the constraints [51, 104]. In particular, the strategies used in [51, 104] show that optimal control and task space optimization for inverse dynamics are not mutually exclusive. [79] linearized the dynamics of a walking gait on ATRIAS to solve a stabilizing continuous-LQR tracking problem, but found that the non-linearities of the system resulted in limited local stability. [29, 62, 182, 207] show that stability from optimal control and strategies that leverage periodicity (discussed next) are also not mutually exclusive. In work in this thesis, we hypothesize that the step-to-step behavior designed with deadbeat control offers faster convergence and improved mobility than the discrete LQR designs, and we explore the application of continuous LQR for tracking the deadbeat stabilized motion plans.

2.3.2 Leg placement as an additional input, and periodic stability

Conceptually, another technique for handling the stability of under-actuated systems is to take advantage of additional control inputs. Without needing to add actuators at the ankles, legged robots differ from the Acrobot model in that they can lift and reposition their feet. This act of taking a step can be a source of control; modifying the parameters (for example, timing and

positioning of the step) allows for feedback and error correction. These adjustments can only be made intermittently and are discrete actions, compared to the continuous joint torques discussed earlier. In effect, this takes a more relaxed perspective to the overall goals – instead of trying to constantly enforce a desired behavior, it may be possible to satisfy the original goal over the course of multiple steps.

In practice, this commonly means that walking, running, or hopping robots can use changes in stride-length to regulate their translational speed. The robot may then be loosely described as no longer under-actuated – there are as many inputs as degrees of freedom, allowing for direct and effective feedback control policies. This phenomenon is the key principle behind the long-reigning success of Raibert's hopping control [146, 147]. The system is fully parameterized by its vertical motion, its horizontal motion, and its angular orientation. As long as all three modes track their targeted behaviors, stability of the full order robot is achieved. Raibert applied simple and robust policies to independently stabilize each of these three modes: thrusting force delivered along the axis of the leg controls the vertical energy, linear feedback for leg placement controls the forward speed from step to step, and PD servoing of the hip while the robot is in stance controls the trunk orientation.

While this strategy does not provide any control authority for converging to target positions and velocities within each stride, it does allow tracking of velocities across multiple steps; any accumulated error of the horizontal position or velocity within individual strides can be reset between strides. Stability of systems that use intermittent inputs can thus be defined with less stringent requirements than a continuous system without discrete inputs; states may be allowed to exhibit temporary divergent behaviors with increasing errors, as long as the intermittent control provides sufficient correction. More formally, the local stability of repeating limit cycles [185] in periodic systems can be studied with Poincare analysis, which examines how the states defined at a once-per-cycle event change in sequential steps. In Chapter 6, we present an in-depth study of how control design choices contribute to the structure of the Poincare map, and we generalize

the analysis of error dynamics to apply to behaviors other than periodic limit cycles.

"Template-anchor" approaches [7, 13, 31, 52, 53, 132, 200] to implementing stable gaits leverage the same underlying principles. A template model (like a rim-less wheel or a leg-placement stabilized spring-mass system) that has inherent periodic stability in its dynamics is identified, and the full order system (the "anchor") is controlled to embed the dynamics of the template. The available feedback on the full order system is then directed towards stabilizing the extra degrees of freedom that are missing from the template model, such that the remaining un-actuated modes are the template-stabilized ones.

2.4 Leg placement and simplified models

2.4.1 Leg placement on running robots

In the existing literature, running robots have demonstrated a wide variety of leg placement policies, ranging from empirical approaches like Raibert's work to more model-based studies that offer theoretically more precise control of the motions as well as more formal analyses of the stability of periodic trajectories. In [85, 86, 87], the HRP-2LR robot does not adapt the leg placement to the current motion of the robot. Rather, the footholds are pre-planned based on a specified long-term behavior, and the robot relies on other the continuous feedback and whole-body motion planning to stabilize the motion given the specified placements. Similarly, HUBO [30] does not reactively choose its leg placement. The hopping controllers studied in [19, 20] likewise treat leg placement as a constraint.

As described earlier, Raibert's pioneering work [146, 147] uses feedback on velocity to determine the leg placement x_f:

$$x_f = x_n - k(\dot{x}_0 - \dot{x}).$$

After solving for the neutral point x_n at which a nominal velocity $\dot{x}_0$ is in equilibrium, the controller targets an adjustment that depends linearly on the velocity error (extend the leg farther to

slow down and bring it closer to speed up), thus controlling the forward hopping speed. Due to its simplicity and its success, this policy has been adopted ubiquitously in simulated running and walking systems [82, 214] as well as real running and walking systems [1, 2, 70, 140, 151].

On the MABEL robot, [181, 182], the leg placement policy is identified as part of a discrete linear quadratic regulator (DLQR) that stabilizes the nominal gait. The authors identify a nominal limit cycle in the space of the full robot model for a running gait using numerical optimization tools. The resulting cycle is unstable on its own, but a linearization of the stride-to-stride return map allows a set of adjustable parameters (including leg placement and trunk orientation at transitions) to serve as control inputs. Applying DLQR directly yields a stabilizing solution, derived from minimizing the summed one-per-stride state errors and the applied discrete adjustments for all future steps.

The Toyota robot in [187] uses a predictive model to adapt the planned motion based on the robot's current configuration. Specifically, the authors specify a desired CoM position and velocity two steps in the future. The control propagates the measured robot state through the current step, and then solves a boundary value problem such that the target state is reached at the end of the second step. The solution of the boundary-value problem includes the desired leg placement of the upcoming step.

On ASIMO [188, 189, 190, 191], adjustments in leg placement are used as a last resort for maintaining balance. That is, the primary source of stabilizing feedback comes from modifying the wrench at the foot. When the desired corrective moment would exceed the moment that the foot can apply, the controller accepts the extra uncorrected motion, which accumulates as an growing error. The upcoming leg placement is then adjusted such that the accumulated error can be addressed in future steps.

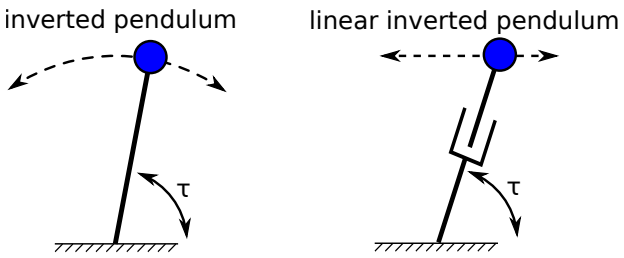


Figure 2.4: The inverted pendulum and linear inverted pendulum are often used to model standing in place and walking for humanoid robots. With a rigid leg, the inverted pendulum is constrained to move on a circle in 2D or a sphere in 3D. With a telescoping leg that keeps the mass at a constant height, the linear inverted pendulum is constrained to move on a line in 2D or a plane in 3D. Without ankle actuation, both systems are unstable around the upright equilibrium, but repositioning the feet periodically resets the motion and can provide stability. Many controllers integrate the un-actuated dynamics of these simplified models to plan the stabilizing leg placement.

Simplified models for leg placement

Simplified models represent the most crucial aspects of a system's dynamics using a low number of parameters and variables. Applied to legged systems, they have been a useful tool for capturing the relationship between the leg placement and the subsequent motion of the system. This structured relationship can be exploited to find highly effective leg placement strategies for achieving specific motions. Leg placement strategies derived from these models have been successfully implemented for walking and balancing tasks, and this approach appears promising for improving the agility and robustness of running.

2.4.2 Inverted and Linear inverted pendulum model

The inverted pendulum describes the system with a single rigid leg pinned to the ground at a stationary foot, and all the mass concentrated at a fixed length l above the foot. The motion of the CoM is thus constrained to lie on a circle (in 2D) or a sphere (in 3D) of radius l centered on the foot. Without an ankle actuator, the system is fully passive. It has an unstable equilibrium point when perfectly upright; it will fall over if slightly perturbed.

Many legged systems do not maintain a constant leg length, as the leg is actively articulated or

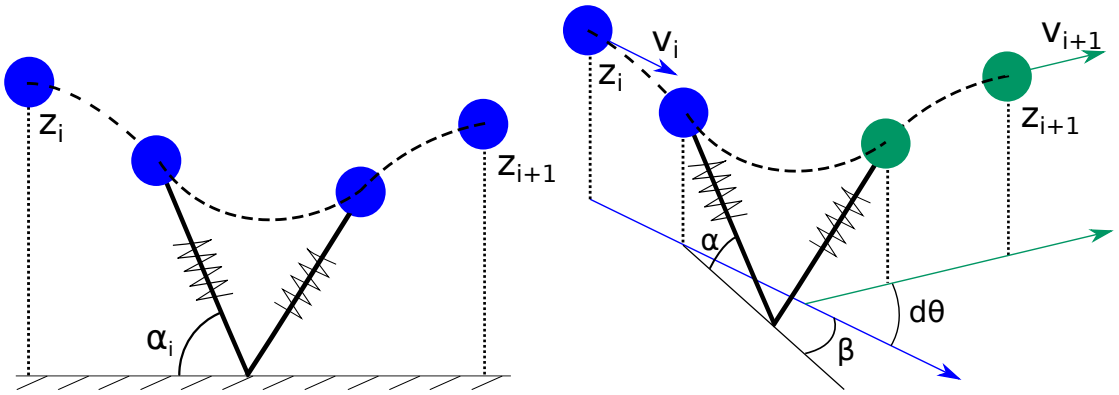
incorporates some other actuated telescoping mechanism. The linear inverted pendulum model (LIPM) modifies the rigid inverted pendulum with the assumption that the leg length varies with the leg orientation such that a constant vertical height z_0 is always maintained; the motion of the CoM now lies on a line or plane instead of a circle or sphere. The name "linear" comes from the fact that these dynamics match the linearization about the upright position of the rigid inverted pendulum. This model is frequently used to represent walking [80, 83, 88, 188, 189, 190, 191], as the body is typically held at an approximately constant height in walking gaits [209].

The equations of motion for the linear inverted pendulum are given by

$$m\ddot{x} = \frac{mgx}{z_0} + \frac{l}{z}\tau_x,$$
$$m\ddot{y} = \frac{mgy}{z_0} + \frac{1}{z}\tau_y.$$

relating the translational acceleration of the CoM to the applied torques in the ankle. Like the inverted pendulum, the LIPM has an unstable equilibrium at the upright configuration, and ankle torques would render the system fully actuated. If no ankle torques are applied, the system is passive and gives a predictive model between initial conditions and the rest of single support, which is often used [64, 80, 81, 105, 121, 186, 187] for identifying sequences of leg placements that bring the system exactly to a desired state after a number of steps. Similarly, in [43, 100, 143, 144], the authors use the model to identify the leg placement corresponding to the capture point, which brings a non-stationary initial condition back to rest in a single step. This analysis is extended to account for angular motion with the extended linear inverted pendulum plus flywheel model. [183] uses this model to solve the model-predictive control problem of trajectory generation for push recovery.

The bipedal linear inverted pendulum is an extension [128] that explicitly considers two legs that gradually shift the vertical load from one leg to the other during double support. Adding the dynamics of the double support phase allows for more direct reasoning of how to manipulate the leg placement to track target speeds and reject disturbances.



(a) SMM in 2D (b) SMM in 3D

Figure 2.5: Spring mass model for running. The point mass bounces on a massless linear spring. The flight phase of the system can be characterized by its apex height, as the speed at the apex can then be deduced from the fixed energy of the conservative system. The transition between sequential flight phases is fully determined given the landing angle of the leg. Thus, the simplified model is described by a mapping between apex states that is parameterized by the leg placement. In 3 dimensions, the flight phase has the direction of motion as an additional parameter, while the leg placement is further characterized by an out of plane splay angle. The stride-to-stride transition is thus a two-to-two mapping.

2.4.3 Spring mass model

The constant vertical height of the LIPM implies constant vertical forces of exactly mg. This does not agree with observations of animal locomotion, which instead exhibit compliant leg behavior [5, 24, 40, 115, 209]; the leg force gradually loads and unloads in a spring-like way. The spring mass model (SMM), or equivalently, spring loaded inverted pendulum (SLIP), is a point mass model that assumes the leg behaves like a linear spring with axial leg force $F = k(l_0 - l)$. This force points along the leg while gravity constantly pulls down on the point mass, and there is no ankle actuation.

The equations of motion of the system are

$$\frac{d}{dt}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{k}{lm}\begin{bmatrix} x \\ y \\ z \end{bmatrix}(l_0 - l) + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix},$$

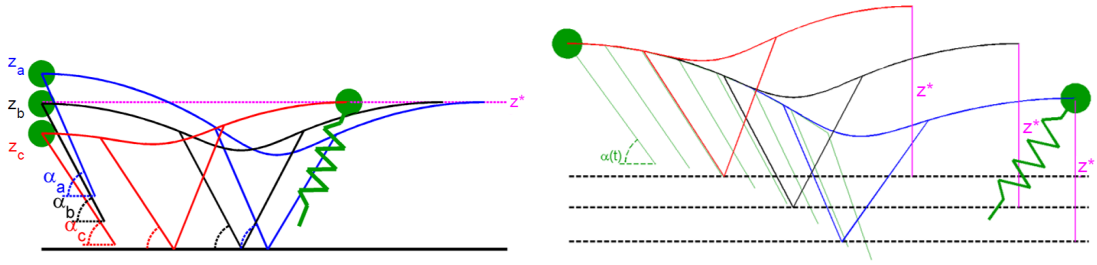
where

$$l = \sqrt{x^2 + y^2 + z^2},$$

and like the inverted pendulum and LIPM systems without ankle torques, these dynamics can be forward integrated from any initial condition to generate the rest of single support. The equation is fairly simple, but no analytic solution for the general trajectories are known while approximations have been found [54, 158, 165]. Even without an analytical form, the model conceptually describes the entire single support phase.

Running and hopping are captured by the SMM as sequential single support phases separated by ballistic flight phases, whose dynamics are $\ddot{x} = \ddot{y} = 0$, $\ddot{z} = -g$. Stance transitions to flight when the leg length reaches l_0, and flight transitions back to stance when the leg again touches the ground at length l_0. The orientation of the leg at touchdown can be freely chosen, however, and it becomes a once-per-stride control input. Using return map analysis [185], numerous studies have explored the effects of specific assumptions of the touchdown angle: constant touchdown angles results in some locally stable gaits in 2D [6, 8, 56, 72, 166, 170], "retraction" of the leg angle improves the stability [11, 16, 47, 169, 171], and while constant angles cannot stabilize the 3D system [166], stabilizing local feedback can be constructed [134, 166].

More generally, the leg angle is a parameter that governs the stride-to-stride transitions assuming SMM dynamics in stance; and control design is simply choosing the parameter that produces the most desirable transition. Indeed, this interpretation of the dynamics for the planar system gives rise to deadbeat running control [27, 47, 157, 170, 171] (Figure 2.6 as well as two-step deadbeat walking control [199]. Conceptually, this is the same as the strategy used by some LIPM policy walkers [80, 187] for a different set of states. In the spring mass system, the discrete once-per-stride states are described by the apex height in flight, and the deadbeat control identi-



(a) Deadbeat leg placement

(b) Implemented as a time law for unobserved terrain

Figure 2.6: Deadbeat leg placement for the planar SMM. The apex-to-apex mapping parameterized by the landing angle α can be inverted to find the leg placement that maps from any current apex (within a specific domain) to the desired one. This provides deadbeat stability of the running gait. If the ground height is unknown, each of the deadbeat landing angles can be incorporated in a time-based policy $\alpha(t)$ such that whenever the system contacts the ground, the current angle is the appropriate one. This transformation into a time policy is possible because the time of landing and leg angle uniquely determine the original apex height. The result is a robust running gait that is insensitive to unobserved changes in ground height.

fies the touchdown angle from any initial condition that returns the system to the target apex in a single step. The domain on which such a solution exists depends on the friction cone as well as the length of the leg. Finally, this deadbeat policy can be rendered robust to unobserved changes in ground height by implicitly inferring the ground height through the time elapsed since apex [47, 170, 171]. In Chapter 3, we extend this terrain-blind deadbeat behavior to include steering in 3D.

In [203, 205], the authors linearize the dynamics of a more generalized model that allows asymmetric spring profiles for a 3D compliant-leg system and solve discrete LQR for stabilizing control over multiple outputs (including system energy and direction of motion) using multiple inputs (including leg placement and stiffness parameters). Similarly, in [21], the spring stiffness is used as an additional control input for stabilizing the spring mass model in 3D. In [27], the authors argue that deadbeat solutions exist for the 3D SMM and propose an iterative computational approach for finding them.

To apply the SMM to walking, the model has been extended with a second mass-less spring leg to describe the dynamics in double support. Similar to the case of running, researchers have analyzed the stability of specific transition policies into double support with varied success [10, 113]. Leaving the leg placement as an arbitrary but manipulable parameter of the stride-to-stride dynamics of the bipedal spring mass model reveals two-step deadbeat solutions for gait tracking [199].

To account for the non-trivial dynamics associated with the hip and the center of mass not coinciding on a more realistic system, [135, 136] map the dynamics of this asymmetrical spring loaded inverted pendulum (ASLIP) back to the simple SLIP by defining a virtual constraint enforced by actuation at the hip. This then allows SMM leg placement strategies to be used directly for the ASLIP model.

2.4.4 Other models for compliant legs

Other formulations similar to the SMM have also been applied to describe locomotion on compliant legs. In [117], the authors first model the vertical dynamics of the system with the spring mass model (independent of the horizontal states) and constrain the net GRF to act along the axis of the leg, thereby resolving the horizontal component of the force. This results in a convenient one way decoupling of the vertical dynamics, and the qualitative behavior of this model is largely the same as the original SMM as long as the leg is nearly vertical. The concept of deadbeat leg placement is unaffected by this modification to the details of the equations of motion.

In [45], the authors define a vertical force profile as a polynomial function of time. They solve the boundary-condition problem to identify the deadbeat force input and resulting state trajectory between the touchdown and desired takeoff conditions. Qualitatively, the resulting force profile retains a spring-like shape while it automatically encodes deadbeat behavior on the vertical states. The horizontal forcing and leg placement is then designed based on this vertical behavior.

2.5 Performance on real machines

Transitioning from a theoretical controller to a hardware implementation is non-trivial. Most model-based running robots end up running at significantly different gaits from the nominally designed one. QRIO's flight phase is 40% of the nominal duration [121], HRP-2LR's running speed is 64% of the nominal speed [85, 86], and MABEL's running speed is 145% of the nominal speed [182]. These differences largely stem from actuator saturation and un-modeled dynamics including collisions, foot slip, and cable stretch.

Pure inverse dynamics based control tends to be unstable on hardware – especially when the generated torque cannot be closed in a feedback loop, and most implementations augment the control with kinematic feedback [50, 51, 101, 104, 107] through integration to reject errors that stem from model errors and torque tracking. Adding these terms distorts the theoretical behavior of the optimized inverse dynamics control.

[18, 51, 101] discuss the effects of various other challenges encountered on the Atlas platform. Joint deformations lead to positional deviations, and these effects are mitigated by approximating the deflection with a linear model. The sensory information is noisy, and depending on the filtering applied, the stability of the system is highly sensitive to feedback gains on joint velocities. The feedback gains are further limited by the latency in the communication, which had to be estimated empirically. [51, 101] report the filter parameters and gains that they ultimately settle on, commenting that the lower gains had "an acceptable cost to tracking performance."

Perhaps due to diversity of the challenges encountered on hardware and diversity of the practical solutions applied towards addressing them, not much work has been done towards quantifying or validating the convergence expected by model-based control design. Sufficiently overcoming the modeling errors, disturbances, and uncertainties to demonstrate stability in real hardware is achievement enough, and at the DRC, reliable repeatability may have been the most salient measure of performance. However, in the interest of reproducing highly agile and robust behaviors predicted by simplified models, it is important to understand at a quantitative level to what

degree this theory is limited by these realities of hardware systems, especially given the large body of potentially applicable control developed around simplified models.

Theoretical treatment of a proposed controller's practical performance is typically evaluated through simulation-based studies of responses to specific perturbations. Many studies simulate uneven terrain [108, 195] or externally applied limited-duration push disturbances [63, 203]. These approaches provide a useful measure of the ability of a system to recover from potentially destabilizing events, but they do not fully describe how the tracking performance of a theoretically derived model-based control would be validated in a comparison to noisy data from an experiment with persistent sources of error.

Meanwhile, in the context of linear systems, there exist methods for quantifying the effects of bounded perturbations to the nominal system [15, 22, 23, 74, 90]. For example, in [22], the authors derive the invariant region for the closed loop evolution of a linear system whose disturbances are limited by finite convex polyhedral bounds. Similarly, [57] formulates the control design problem to preserve set invariance subject to bounded perturbations. Furthermore, many of the issues discussed above can be represented as disturbance forces to a nominal closed loop system that is agnostic to these effects.

In Chapter 7 of this thesis, we quantify the disturbances and uncertainties we encounter empirically and apply numerical techniques and Lyapunov theory to formulate the theoretical performance in terms of expected bounds. We find that the tracking demonstrated in our experiments in Chapter 5 indeed lie within the expected range. While the conservative nature of these bounds can limit their descriptiveness of the actual performance, this formulation nonetheless serves as a first step towards analytically comparing theory and empirical results.

2.6 Summary

Maintaining balance and stability in legged systems is a challenging problem. Among other strategies for feedback, leg placement offers a source of intermittent control. Many existing

implementations have leveraged this control input to demonstrate stable gaits, but analyses of simplified models suggest that higher performance control may be possible, especially for running gaits.

However, robots have not yet demonstrated these highly agile and robust behaviors. Furthermore, the theory does not yet clearly extend past an idealized low order model. The theoretical behavior derived from a low order system may be limited by the requirements of stabilizing the additional modes on a full order robot, and it may also be compromised by the unavoidable errors and uncertainties associated with real hardware. The toll of these effects is currently unclear, so the value of these theories remains unknown. The work in this thesis begins to bridge the gap between hardware and theory for spring-mass running.

Chapter 3

Deadbeat spring mass running in 3D

This chapter presents a study of agile and robust running in 3D within the dynamics of the spring mass model (SMM), and the results show that the low order dynamics of this model have very promising properties for agile and robust running. Despite the ground contact wrench constraints and the under-actuation of the foot, the model is nonetheless capable of executing highly dynamic motions. Leg placement allows the system to tightly control the speed and direction of running, even on unobserved, uneven terrain.

Most of this material is published in the journal paper [210]. Section 3.1 reviews the background and the motivation for this work. In Section 3.2, we derive the deadbeat control for bounce height and direction of motion, extend it to perform on uneven terrain, and compare the tracking performance to classical heuristic leg placement strategies [146]. Finally, Section 3.3 frames the relevance of these results in the context of designing and implementing control for legged robots.

3.1 Background

Over the past three decades, the spring mass model has developed into the basic behavior model for studying the running gait in animals and robots. Established in the 1980's [24, 115, 146, 148],

the model has helped in early work to develop locally stable feedback controls for hopping and running robots [146, 215] and to understand the leg function in animal and human runners [5, 24, 25, 40, 48, 115]. More recently, the focus on this model has shifted to developing approximate solutions [54, 158, 165] and to understanding global running stability by analyzing the model's hybrid dynamics with Poincaré maps. This analysis method has proved especially productive [72]. It has helped to reveal and understand the passive stabilization of insect running in the horizontal plane [162, 163, 164], and of robot and human running in the sagittal plane [6, 8, 56, 170]. In addition, it has resulted in control strategies that promise to largely improve the stability in running robots [11, 13, 16, 47, 161, 169, 171] (see Section 2.4.3 for a detailed discussion), to the extent that they could one day outperform their animal competitors.

While the planar spring-mass model has greatly advanced the understanding of running dynamics and control in the horizontal and sagittal planes, few attempts have been made to transfer this knowledge to three-dimensional locomotion. Seipel and Holmes [166] investigated running stability with Poincaré analysis of the 3D spring-mass model. They showed that symmetric gaits with a constant in-flight leg orientation defined with respect to the desired direction of motion are unstable. By introducing linear feedback control derived from a local stability criterion, they found stable running for motions sufficiently close to the sagittal plane. Peuker, Maufroy and Seyfarth [134] expanded these results in a recent numerical study. Inspired by insights from the work on insect running in the horizontal plane [164], they explored running stability using constant leg angle policies with respect to the direction of motion and observed locally stable gaits for a large range of leg angles. Beyond local stability analysis, Carver, Cowan and Guckenheimer [27] investigated deadbeat stepping policies that drive the 3D spring mass model into desired system states using the least number of steps. In particular, after deriving existence and uniqueness conditions, they proposed an iterative algorithm to compute these policies and demonstrated it in examples where the model's horizontal position is perturbed in flight.

Here we show that identifying deadbeat policies for stability and steering in 3D spring-mass

running can be largely simplified and that the resulting controller can in part be embedded in a feed-forward manner generating highly robust locomotion over rough terrain. First, by parameterizing the full model dynamics in a motion-aligned frame, we reduce the essential behaviors of stability and steering to a compact representation. We then use simple interpolation over this representation to pre-compute the deadbeat policies for attaining all possible running behaviors. Finally, we map these policies into the time after apex, and show in a simulation example that the resulting control enables navigation to waypoints over rough terrain with large, frequent and unknown ground disturbances (up to 30% of leg length in the example). Our results suggest that compact and efficiently computed dead-beat policies for the 3D spring-mass model can be combined with time-embedding to provide a powerful alternative for leg placement strategies in highly maneuverable running robots.

3.2 Method and results

A few comments on notation. In this chapter, θ refers to the direction of motion (Fig. 3.1). The rest of this thesis focuses more on whole-body control in the sagittal plane, and θ is re-defined to describe the orientation of the robot (Fig. 4.2).

Throughout the thesis, we use vector representations of states and control inputs. Unless explicitly denoted in the matrix equations, these vectors are column vectors. When written out in text form (for example, $\boldsymbol{q} = [x,\ y,\ z]$, $\boldsymbol{s} = [\boldsymbol{q}, \dot{\boldsymbol{q}}]$), we sometimes omit the transpose operations to reduce clutter. In this example, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are a 3-by-1 column vectors, and $\boldsymbol{s}$ is a 6-by-1 column vector.

3.2.1 Dynamics and Problem Formulation

We model 3D spring-mass running similar to previous work [27, 56, 133, 166, 170] (Fig. 3.1). The flight phase is modeled as a point mass m following a ballistic trajectory with

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}, \tag{3.1}$$

where the coordinates are given in the apex frame. The apex describes the highest point in flight and the apex frame aligns the x-axis with the direction of motion $\boldsymbol{v}$ at that point (with z pointed up). In addition, the frame's origin is located at the ground level. After the apex event, the angle of attack α and the splay angle β parameterize the swing leg orientation. The swing leg has a constant length l_0, and the model transitions to stance when the point mass reaches the touchdown height $z_{TD} = l_0 \sin(\alpha)$.

During stance, the point mass rebounds on a massless spring with stiffness k and rest length l_0, assuming that the foot point stays fixed on the ground. The dynamics of the point mass are governed by the gravitational force $m\boldsymbol{g}$ and the spring force $\boldsymbol{F} = k(l_0/|\boldsymbol{l}| - 1)\boldsymbol{l}$ acting along the leg axis $\boldsymbol{l}$,

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = k(l_0/|\boldsymbol{l}| - 1) \begin{bmatrix} x - x_f \\ y - y_f \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}, \tag{3.2}$$

where $[x\, y\, 0]_f^T$ is the location of the foot point and $|\boldsymbol{l}|^2 = (x-x_f)^2 + (y-y_f)^2 + z^2$. The resulting trajectory of the point mass determines the spatial orientation and length of the spring. When the spring length returns to l_0 during rebound, the model transitions back to the flight phase.

The model describes running dynamics with alternating stance and flight phases of the left and right legs as long as four conditions are fulfilled. First, the model reaches an apex in flight. Second, at apex the height of the point mass exceeds the landing height, $z_a > l_0 \sin(\alpha)$. Third, the point mass remains above ground throughout stance, $z > 0$. And fourth, the horizontal leg

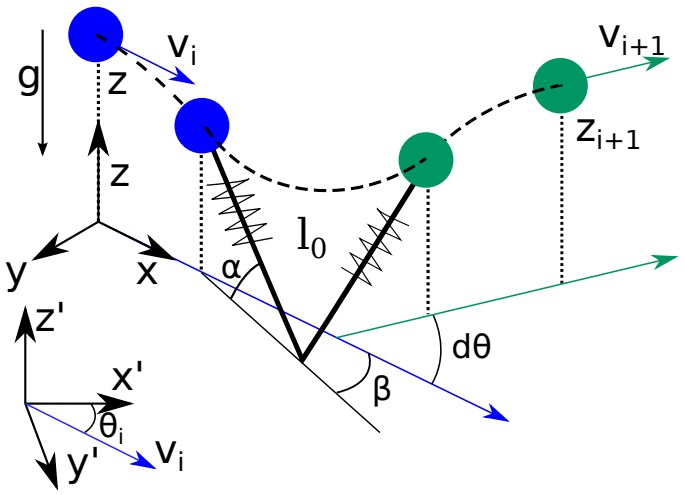


Figure 3.1: The 3D spring-mass running model. The model describes a sequence of flight and stance phases. The ith flight is characterized by a ballistic trajectory of a point mass m. The apex velocity vector $\boldsymbol{v}_i$ and the vertical axis define an apex frame (x, y, z) and a heading θ_i with respect to the world frame $(x, y, z)'$. The leg of length l_0 is positioned with an angle of attack α and a splay angle β, both measured in the apex frame. In stance the point mass rebounds on a massless spring leg (stiffness k, rest length l_0). The model's net behavior can be captured by a behavior function f which maps the apex height $z_{i+1}(z_i, \alpha_i, \beta_i)$ and the change in heading $d\theta(z_i, \alpha_i, \beta_i)$. Due to the symmetry of the model, f equally captures left and right leg behavior of the alternating stance phases in running. g: gravitational acceleration, $\boldsymbol{v}_{i+1}$: velocity vector at apex $i + 1$.

forces stay within the friction limit, $\sqrt{F_x^2 + F_y^2} < \mu F_z$, where μ is the static friction coefficient. (We do not consider slipping.) The last condition translates into a minimum angle of attack $\alpha_{min} = \text{atan}(\mu^{-1})$. Smaller landing angles will always lead to a violation of the friction condition in stance.

Behavior Function and Control Objective

The behavior of the spring-mass model is captured by the apex return map, which is a discrete mapping of the system state at apex between two consecutive flight phases. Poincare analysis [185] is often been applied to test the stability of a passive system [95, 114]; here we use a similar thought process to actively generate stable control. The complete map tracks the state vector $[x\ \dot{x}\ y\ \dot{y}\ z\ \dot{z}]_a$, where the subscript a denotes apex for each variable. However, to characterize the model behavior, it suffices to track only a pair of variables, the apex height z and the deflection $d\theta$ in heading. Similar to the planar case, $\dot{z}_a = 0$ by the definition of the apex. Furthermore, the stance dynamics depend on the relative position of the foot with respect to the point mass and its direction. This position is parameterized by α and β, making $[x\ y]_a$ unnecessary for computing the motion between apexes (just as x can be discarded given α in the planar case). Note, however, that these coordinates are required to track the absolute position of the system within the global frame. Among the remaining variables, the magnitude of the horizontal velocity is coupled to the height z_a by the constant system energy E_s with $\dot{x}_a^2 + \dot{y}_a^2 = \frac{2E_s}{m} - 2gz_a$. Therefore, the model behavior is captured by

$$(z_{i+1}, d\theta) = f(z_i, \boldsymbol{p})$$

where $d\theta$ is the horizontal deflection after the stance phase, f is a function of the model dynamics, and $\boldsymbol{p}$ is the vector of model parameters. That is, given $\boldsymbol{p}$, the previous apex height z_i is the only state needed for deriving the *relative* motion of the next flight phase. From this representation, the apex return map R that characterizes running stability and global steering of the 3D spring-mass

model can be formalized as

$$(z, \theta)_{i+1} = R(z_i, \theta_i, \boldsymbol{p})$$

with θ_i describing the global heading in the ith step (Fig. 3.1) and $\theta_{i+1} = \theta_i + d\theta$.

The model behavior can be controlled by manipulating the return map, for which a goal behavior is equivalent to a target state $(z, \theta)^*$. The control problem consists of finding ways to drive the model to the target state from arbitrary initial conditions, and deadbeat control is achieved if a parameter policy $\boldsymbol{p}(z_i, \theta_i, z^*, \theta^*)$ exists that enforces $(z, \theta)_{i+1} = (z, \theta)^*$ for all possible $(z, \theta)_i$. As a result, the general problem of controlling the model behavior resolves for each initial state i to the minimization problem

$$\boldsymbol{p}_i = \underset{\boldsymbol{p}}{\operatorname{argmin}}\, d(\ (z, \theta)^*, R(z_i, \theta_i, \boldsymbol{p})\ ) \tag{3.3}$$

where $d(.,.)$ is an appropriate measure of distance (the Euclidean norm, for example). If a vector $\boldsymbol{p}_i$ exists that brings the minimum to zero, then deadbeat control is achieved; otherwise, the state $(z, \theta)_{i+1}$ will be as close to $(z, \theta)^*$ as possible.

In the 3D spring-mass model, the vector

$$\boldsymbol{p} = [E_s\ m\ g\ k\ l_0\ \alpha\ \beta]$$

contains seven parameters, several of which are suitable for control. In particular, we choose the two angles that define the swing leg orientation as the control inputs, $\boldsymbol{u} = [\alpha\ \beta]$, to demonstrate how leg placement in the flight phase generalizes the deadbeat control of planar spring-mass running to 3D locomotion and steering. The remaining parameter vector $\hat{\boldsymbol{p}}$ is fixed to $E_s = 1785$ J, $m = 80$ kg, $g = 9.81$ ms^{-2}, $k = 15$ kNm^{-1} and $l_0 = 1$ m; these values fit a human-like system running at about 5ms^{-1} [170]. The model behavior function f is thus written as

$$(z_{i+1}, d\theta) = f(z_i, \boldsymbol{u}, \hat{\boldsymbol{p}}) \tag{3.4}$$

and the control problem reduces to

$$[\alpha\ \beta]_i = \underset{[\alpha\ \beta]}{\operatorname{argmin}}\, d(\ (z,\theta)^*, R(z_i,\theta_i,\boldsymbol{u},\hat{\boldsymbol{p}})\ )$$

$$[\alpha\ \beta]_i = \underset{[\alpha\ \beta]}{\operatorname{argmin}}\, d(\ (z^*,\theta^*-\theta_i), f(z_i,\boldsymbol{u},\hat{\boldsymbol{p}})\ ) \tag{3.5}$$

Note that more parameters can be included in solving (3.3). The added degrees of freedom in the control input can then be used to consider further performance metrics (compare [47] for a planar example), or to embed lower priority tasks as in redundant manipulators [93]. In this work, matching the degrees of freedom between the control targets $(z,\theta)^*$ and inputs (α,β) avoids this redundancy and simplifies the demonstration of deadbeat control.

3.2.2 Control Strategy Derivation

The behavior function $f(z_i,\boldsymbol{u},\hat{\boldsymbol{p}})$ is nonlinear. To solve (3.5) for arbitrary target states, we first perform a numerical scan to generate a discrete representation of f. We then define a particular measure d and use interpolation to derive the control inputs $[\alpha\ \beta]_i$ that project from the initial states into arbitrary target states. In the final part of this section, we apply the transformation between apex height and falling time [169] to derive a control strategy that does not require knowledge about the actual ground level to produce deadbeat control of stability and steering.

Discrete Representation of Model Behavior

We scan the model behavior $f(z_i,\boldsymbol{u},\hat{\boldsymbol{p}})$ over the inputs α and β for all possible initial apex heights z_i. As control input ranges, we choose $\alpha \in [45°, 90°]$ and $\beta \in [0°, 180°]$. The lower bound α_{min} is governed by the static friction limit assuming $\mu = 1$. Due to the reflection symmetry of the model with respect to the sagittal plane, it suffices to scan β within $0°$ and $180°$ to capture the model behavior for all possible splay angles. The initial height is limited to $z_i \in [0.71, 2.27]$ m. The minimum corresponds to the landing height defined by $z_{\text{min}} = l_0 \sin\alpha_{\text{min}}$; the maximum $z_{\text{max}} = E_s/(mg)$ results from converting all system energy into potential energy. Us-

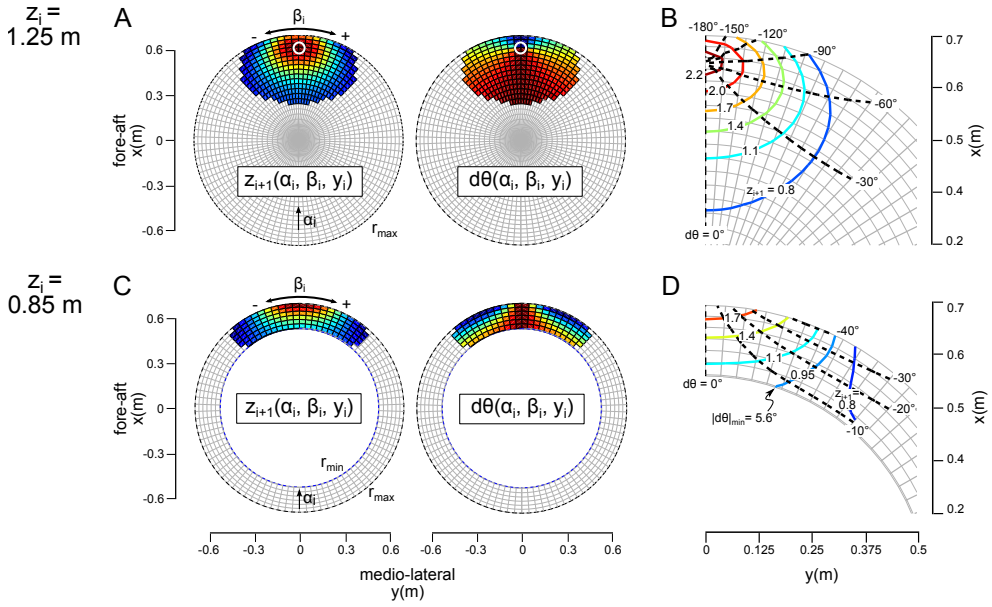


Figure 3.2: Numerical representations of model behavior for two different initial apex heights $z_i = 1.25$ m (A-B) and $z_i = 0.85$ m (C-D). (**A**) Apex height $z_{i+1}[\alpha, \beta, z_i]$ and deflection $d\theta[\alpha, \beta, z_i]$ as functions of foot placement projected onto horizontal plane (Fig. 3.1) for $z_i = 1.25$ m. The radius r_{max} describes the friction limit. Maps are color coded from low (blue) to high values (red). Empty regions indicate that the model does not return to an apex in the next flight phase. The white circle highlights the stopping point at which all energy is directed into vertical motion. (**B**) Closeup section with apex height and deflection behaviors overlaid as contour plots. Contour lines show inputs $[\alpha\ \beta]_i$ that produce the same apex height z_{i+1} (ring sections) or deflection $d\theta$ (rays). (**C,D**) For $z_i = 0.85$ m, the landing condition limits the maximum angle of attack to $\alpha_{\text{max}} = \text{asin}(z_i/l_0)$. This constraint results in a minimum radius r_{min} for the feasible foot positions, reducing the domain of the behavior map projection to an annulus.

ing these ranges, we generate the discrete functions $z_{i+1}[\alpha, \beta, z_i]$ and $d\theta[\alpha, \beta, z_i]$ that represent $f(z_i, \boldsymbol{u}, \hat{\boldsymbol{p}})$ (3.4) with an α-β-z_i resolution of 22x45x150 by numerically integrating the model dynamics (Eq. 3.1) and (Eq. 3.2) between two apexes (computation time less than a day on a modern desktop PC).

Figure 3.2 shows the discretized model behavior for two different initial heights z_i. The control inputs α and β are represented as foot point locations relative to the point mass and projected onto the x-y plane. The projection forms a disc centered on the origin, which corresponds to $\alpha = 90°$ with the foot directly below the point mass. The distance from the origin is given by $l_0 \cos(\alpha)$, and β describes the angular position relative to the direction of motion. The friction limit defines the maximum disc radius $r_{\text{max}} = l_0 \cos \alpha_{\text{min}}$ (Fig. 3.2A). The discrete functions $z_{i+1}[\alpha, \beta, z_i]$ and $d\theta[\alpha, \beta, z_i]$ are well behaved and centered about a characteristic point (white circle) lying on the $\beta = 0°$ axis. This characteristic point is the spring-mass model's equivalent to the capture point of the linear inverted pendulum walking model [80, 143, 196]. Landing with the foot positioned at this point stops the motion in the x-y plane as all the system energy is re-directed into vertical motion with $z_{i+1} = z_{\text{max}}$. Landing past this point leads to backward motion ($d\theta = 180°$). Foot positions resulting in equal apex heights z_{i+1} form concentric rings about the stopping point with decreasing heights, eventually interrupted by the r_{max} given by the friction limit. By contrast, configurations with equal deflections $d\theta$ form rays that originate from the stopping point. The closeup in figure 3.2B shows that the inputs $[\alpha\, \beta]_i$ can reach a large range of target behaviors $(z, d\theta)^*$.

The range of target behaviors that can be reached diminishes when the initial height drops below the rest length $l_0 = 1$ m of the spring. Figure 3.2C-D shows an example with $z_i = 0.85$ m. The disc of foot positions reduces to an annulus with an inner radius $r_{\text{min}} = \sqrt{l_0^2 - z_i^2}$ that corresponds to a landing angle $\alpha_{\text{max}} = \text{asin}(z_i / l_0)$ (Fig. 3.2C). Steeper angles of attack cannot exist due to ground contact. As a result, individual behavior goals may not always be achievable simultaneously. For instance, a target height of $z^* = 0.95$ m cannot be achieved with a target

deflection $d\theta^* = 0^\circ$ from this initial apex height (Fig. 3.2D).

Deadbeat Control Policy

For controlling the model behavior, we prioritize robustness against ground disturbances over heading directions and require that the apex height goal is achieved whenever possible. To realize this constraint, we split (3.5) into two stages. The first stage identifies the set of control inputs

$$U_{z^*}(z_i) = \{[\alpha\ \beta] :\ z_{i+1}(\alpha, \beta, z_i) = z^*\} \tag{3.6}$$

that enforce the desired apex height z^*. The second stage then solves the minimization problem

$$[\alpha\ \beta]_i = \underset{U_{z^*}(z_i)}{\operatorname{argmin}} |\theta^* - \theta_i - d\theta(\alpha, \beta, z_i)| \tag{3.7}$$

to find the control input $[\alpha\ \beta]_i \in U_{z^*}(z_i)$ that best aligns the heading with the target heading θ^*. Given a target apex height, the two stages are straightforward to implement using bilinear interpolation on the discrete functions $z_{i+1}[\alpha, \beta, z_i]$ and $d\theta[\alpha, \beta, z_i]$ for the range of initial apex heights z_i.

The constrained optimization on the behavior function yields the tables $\alpha[z_i, d\theta^*, z^*]$ and $\beta[z_i, d\theta^*, z^*]$, which contain the control policy that projects initial states $(z, \theta)_i$ into target states $(z, \theta)^*$. The domain of this policy is limited to the set of initial heights $\{z_i | U_{z^*}(z_i) \neq \emptyset\} \subset [l_0 \sin(\alpha_{\text{min}}), E_s/mg]$ for which the apex constraint can be satisfied. If z_i falls outside of this domain, it is impossible to project to the target z^* in one step and the control policy picks the nearest neighbor height from the domain.

As an example, figure 3.3A shows as contour plots the resulting policy for achieving a target height of $z^* = 1.1$m. For this target height, all possible initial heights $z_i \in [0.71, 2.27]$m can be projected to $z_{i+1} = z^*$ in a single step. The policy is characterized by a divider, which is defined by the friction limit and the touchdown condition (black curve). Above (and to the right of) the divider is the deadbeat region, where any target deflection can be achieved along with the

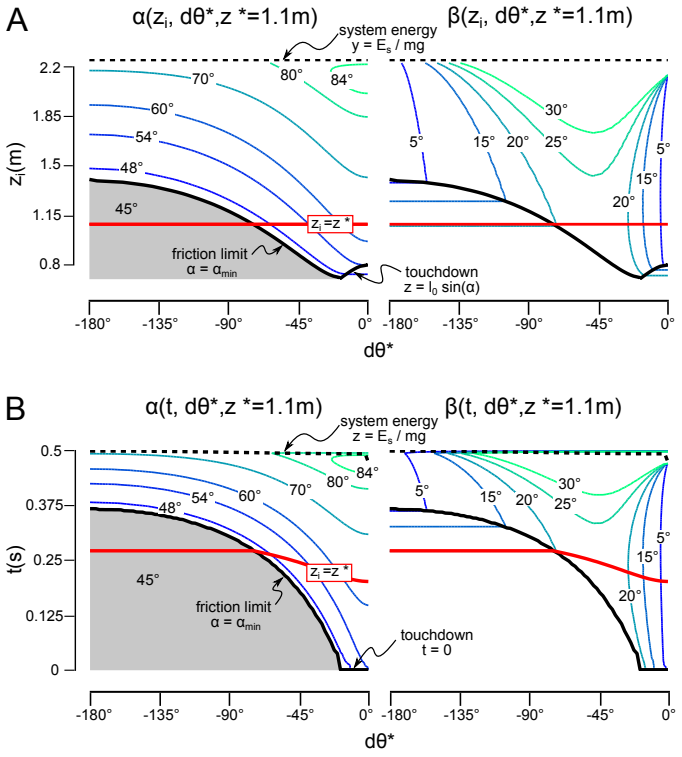


Figure 3.3: Control policy for achieving $z^* = 1.1$ m. (**A**) Control tables $\alpha[z_i, d\theta^*, z^*]$ and $\beta[z_i, d\theta^*, z^*]$ are shown as contour plots for all possible initial heights z_i with ranges of target deflections $d\theta^*$. The black curve describes the friction and touchdown limits. The region of deadbeat control for stability and steering is above this divider and below the system energy limit. Below the deadbeat region, the control policy clips the parameters to the ones on the divider to minimize $|d\theta - d\theta^*|$ while enforcing $z_{i+1} = z^*$. The red line indicates the steady state control $z_i = z_{i+1} = z^*$ (**B**) Same as (A) except that control tables are transformed to the time after apex.

target height, realizing deadbeat control for both behavior goals. Below the divider, however, target deflections cannot be reached and the control policy returns the closest alternative. For instance, the spring-mass model can achieve steady state running with $z_i = z^* = 1.1$ m for target deflections $d\theta^* = \theta^* - \theta_i$ up to $\pm 74°$, where the control input is $[\alpha\ \beta] = [45°, \mp 20°]$ (intersection of red line and black curve in each panel). For larger target deflections, the control input will remain at $[45°, \mp 20°]$, maintaining deadbeat control of the apex height but clipping the directional tracking.

Transformation into Time Policy

Previous work on the planar spring-mass model has shown that deadbeat control using the angle of attack $\alpha(z_i)$ can be transformed into a time-based control $\alpha(t)$ during flight if the time is measured from the apex event, $t \geq t_{\text{apex}}$ [47, 169, 171]. This transformation fundamentally changes the control. Instead of selecting a single leg placement based on a single height at apex, the time-indexed sequence of leg placements embeds the deadbeat behavior across a range of apex heights. In effect, ground height variations in the upcoming step are automatically accommodated as they correspond to different falling times. We generalize this transformation to include the splay angle β and generate a control policy that does not require knowledge about the actual ground level to produce a time-based deadbeat control of stability and steering.

The time transformation takes advantage of the free-fall dynamics after the apex to replace z_i with a function of time in the control tables. The height of the point mass at touchdown is given by $z_{TD} = l_0 \sin\alpha$. Given that time t has passed between apex and touchdown, this apex must have been at a height $z_{\text{apex}} = l_0 \sin\alpha + \frac{g}{2}t^2$. The correspondence can be used to transform the table $\alpha[z_i, d\theta^*, z^*]$ into a time-based table

$$\alpha[z_i, d\theta^*, z^*] \rightarrow \alpha[t_i, d\theta^*, z^*] \tag{3.8}$$

which defines the angle-of-attack control as a time policy after the apex event. At the time

$$t_i = \sqrt{\frac{2}{g}(z_i - l_0 \sin \alpha[z_i, d\theta^*, z^*])}$$

the angle of attack assumes the value $\alpha[z_i, d\theta^*, z^*]$. If the model lands, the apex height must have been z_i; otherwise, the flight phase continues and the system automatically prepares for stance at a later time corresponding to a different pair of apex height and angle of attack. Since $\alpha[z_i, d\theta^*, z^*]$ belongs to a joint control input for α and β, the control table of the splay angle transforms in the same way,

$$\beta[z_i, d\theta^*, z^*] \rightarrow \beta[t_i, d\theta^*, z^*]. \tag{3.9}$$

In effect, (3.8) and (3.9) realize deadbeat control of stability and steering without actually sensing the ground level.

Figure 3.3B shows the resulting time policy for the previous example of achieving a target height of $z^* = 1.1$m (Fig. 3.3A). The policy shares features with the original control policy. A divider (black curve) again separates between the deadbeat control region that achieves target deflections along with the target heights (above divider) and the region in which the deflections are clipped to the closest alternative (below divider). In the time policy, the divider is solely defined by the friction limit as the landing condition maps onto $t = 0$. At that time, only deflections that are smaller than $\pm 20^\circ$ can be realized. As time progresses, larger target deflections become possible with the same minimum landing angle ($\alpha_{\text{min}} = 45^\circ$), allowing sharper turns that in steady-state (red curve with $z_i = z^*$) can reach up to $\pm 74^\circ$ per step. Reducing the friction coefficient μ enforces larger maximum landing angles and leads to smaller maximum deflections.

For a given target deflection $d\theta^*$, the time policy describes an increasing angle of attack $\alpha(t)$ that retracts the leg in the present example. Although leg retraction has been associated with stabilizing running [34, 69], in general, it is not the cause of stability. Rather, deadbeat control requires a specific sequence of landing angles, which can appear as retraction for some model parameters but not for others.

3.2.3 Control Performance in Simulation

We test the performance of the time-based control strategy (3.8) and (3.9) for running and steering in a 3D simulation environment. Specifically, we focus on locomotion robustness when encountering large, frequent and unknown changes in the ground height, and on the sensitivity of the control to changes in model parameters. In addition, we provide in these tests a comparison to the leg placement strategy of the classical Raibert controller [146] as a baseline for assessing the quality of the proposed strategy.

Robustness to Changes in Ground Height or System Energy

In the robustness test, the simulated environment consists of flat square tiles (1x1m^2) which define random height levels that are uniformly distributed between $\pm\frac{\Delta z_{\max}}{2}$ (Fig. 3.4), with $\Delta z_{\max} = 25$cm (25% of the model's nominal leg length) as the maximum step change that can be encountered going up or down. The spring-mass model is launched into this environment at an initial apex height of $z_0 = 1$m with a velocity of 5ms^{-1} (model parameters as defined in section 3.2.1). The target height is set to $z^* = 1.1$m. In addition, we assume that waypoints define heading targets, where the target deflection $d\theta^*$ is computed as the change required to redirect the motion in flight towards the current waypoint. Once the model gets within 0.5m of this waypoint, the target heading switches to the next one. (We forego advanced path planning algorithms for simplicity.)

In the simulation, the time-based control shows large robustness to changes in the ground level. The model navigates robustly through the 3D environment, maintaining running stability while making sharp turns toward the way-points (Fig. 3.4). The control performance is summarized in Fig. 3.5 for a random navigation trail with a total of 500 steps (Fig. 3.5A). The vertical steps encountered on the trail cover the full range from -25cm to +25cm (Fig. 3.5B). Whenever touchdown occurs at a ground height $z'_{g,i} = 0$, the target height is met exactly ($z_{i+1} - z^* = 0$, Fig. 3.5C). Ground deviations $y'_{g,i} \neq 0$ on the other hand can lead to subsequent apex heights z_{i+1} with errors $\epsilon_z = z_{i+1} - z^*$ up to -10/+20cm and to errors in deflection $\epsilon_{d\theta} = d\theta_{i+1} - d\theta^*$ of up

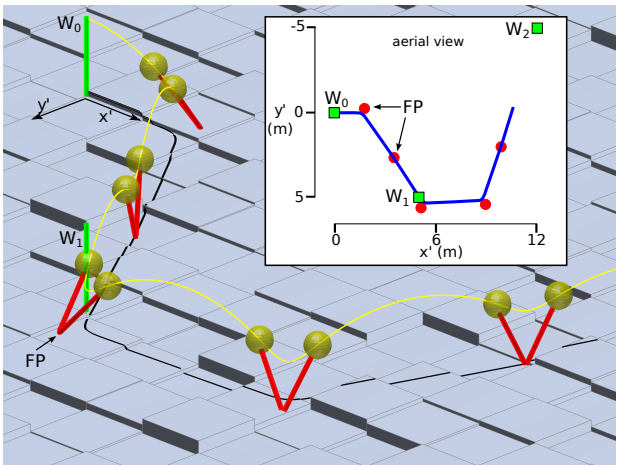


Figure 3.4: Time-based control of stability and steering in 3D environments with large, frequent and unknown changes in the ground height. The spring-mass model is launched at 5ms^{-1} over random ground with step changes up to 25cm. In flight, the leg's attack and splay angles are updated after the apex event based on the time policy (3.8) and (3.9) with a target height of $z^* = 1.1$m (Fig. 3.3B). Target deflections are generated by waypoints W_i (green poles). The resulting trajectory of the point mass is shown (yellow, projection onto horizontal plane in black) with the leg sketched at touchdown and takeoff (red). The inset provides an aerial view with the point mass trajectory in blue and the resulting foot placements FP in stance marked as red circles.

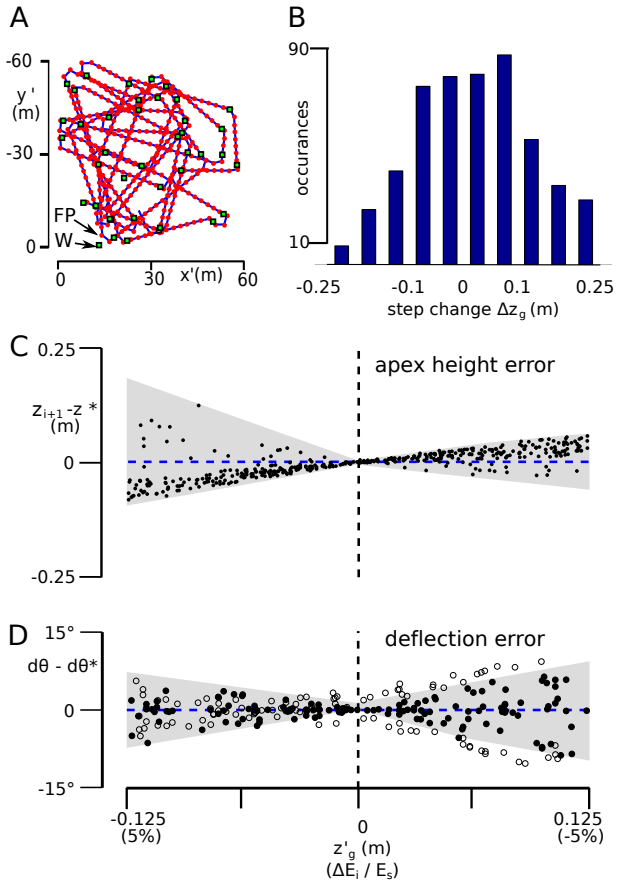


Figure 3.5: Performance of time-based control with target height $z^* = 1.1$ m. (**A**) Aerial view of random navigation trail with 500 steps. Waypoints (W), resulting point mass trajectory and foot points (FP) shown as green squares, blue curve, and red circles, respectively. (**B**) Histogram of encountered step changes. (**C**) Error in apex height $\epsilon_z = z_{i+1} - y^*$ as functions of ground level z'_g and relative disturbance in system energy. The grey envelop is used in later figures for comparisons. (**D**) Corresponding deflection errors $\epsilon_{d\theta} = d\theta - d\theta^*$. If target deflections fall outside of the friction limit divider line (Fig. 3.3B), $\epsilon_{d\theta}$ is measured with respect to the clipped target deflections. In contrast to the non-clipped cases (filled circles), the clipped ones (open circles) can have deflection errors at $z'_g = 0$ due to interpolating the divider line.

to $\pm 10^{\circ}$. These errors fall within the deadbeat region of the control (section 3.2.2 and Fig. 3.3) such that they are corrected in the next step and do not accumulate. In effect, the system keeps robustly navigating through the environment despite the large ground disturbances.

The tracking errors are caused by changes in system energy, which are induced by the changing ground levels. We derived the time-based deadbeat control from the behavior function f assuming a particular system energy $E_{sys} = \frac{m||\boldsymbol{v}_0||^2}{2} + mgz_0$ given by the initial conditions $||\boldsymbol{v}_0|| = 5\text{ms}^{-1}$ and $z_0 = 1\text{m}$. In the environment with nonzero landing heights, however, the effective system energy changes from step to step by $\Delta E_i = -mgz'_{g,i}$ (between $\pm 5\%$ of the total energy), which alters the system behavior and reduces the precision of the applied deadbeat control. (Note that changes in ground level or system energy thus express the same disturbance to the system behavior as indicated by the horizontal axis in Fig. 3.5C).

In general, we observe similar ranges of errors when encountering disturbances of similar *relative* changes in energy. For instance, at system energies corresponding to slower running speeds of 2.5ms^{-1} and 3.75ms^{-1} (at apex height $z_a = 1\text{m}$), we find similar error ranges to the ones we observed for running at 5ms^{-1} if we limit the ground changes to levels of 6.5cm and 8.5cm, which correspond to the same 5% changes in energy.

Although energetic disturbances can be corrected by stabilizing the system energy in stance, such corrections reduce the energetic efficiency of locomotion when performed from step to step. Because the spring mass model is conservative, it cannot correct for changes in system energy. To overcome this limitation, several extensions of this model have been proposed in theory and application [12, 55, 96, 146, 161, 215]. These extended systems can effectively cope with energetic disturbances ranging from dissipative losses to sloped terrain by regulating system energy during stance. However, the control proposed here demonstrates that a tight regulation of system energy from step to step is not required on rough terrain that is flat on average. Nor would such regulation be energetically effective, since it would brake in one step only to re-accelerate in the next. In contrast, the proposed control strategy tolerates local, step-to-step changes in system

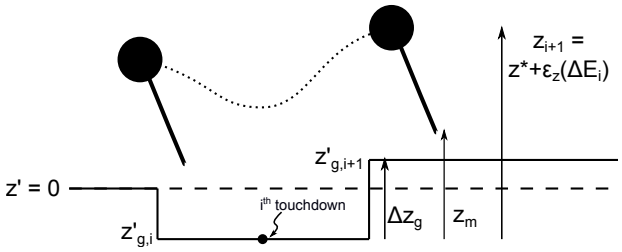


Figure 3.6: Stumbling margin. Variable ground levels $y'_{g,i}$ introduce energetic disturbances $\Delta E_i = -mgz'_{g,i}$ resulting in the tracking error $\epsilon_z(\Delta E)$ that is measured with respect to the ground at the previous touchdown. The stumbling margin y_m describes the maximum height of the ground change Δz_g that the system can tolerate without stumbling in the next step.

energy or ground level without consuming power. An extension of this control with an explicit method for stabilizing system energy over a longer horizon of multiple steps could thus combine robust and energy efficient running and steering in sloped environments.

Margin of Stability against Stumbling

While the model did not fail, the margin of stability against stumbling was only $\Delta z_{\text{max}} = \pm 19$cm in this example. The stability margin z_m is given by the vertical distance

$$z_m = z_{i+1} - l_0 \sin \alpha[0, d\theta^*, z^*]$$

between the foot and the ground at apex (Fig. 3.6). Stumbling occurs if this margin vanishes and the leg protrudes into the ground ($z_m < \Delta y_g$). In steady state locomotion with $z_{i+1} = z^* = 1.1$m, the time-based control has a margin of at least 29cm (at $t = 0$, $\alpha \leq 54°$, Fig. 3.3B). Since the errors in system energy produce apex height tracking errors, the actual margin deviates from the steady state value depending on the disturbance encountered in the previous step. In the worst case, the previous step causes the maximum observed undershoot of $\epsilon_z = -10$cm (Fig. 3.5C), reducing the margin for the next step to

$$z_m = z^* + \epsilon_z - l_0 \sin \alpha[0, d\theta^*, z^*] \approx 19\text{cm}. \tag{3.10}$$

Systematic Error	Deadbeat / Constrained Deadbeat / Classical														
	steady state ϵ_z (cm)			RMS ϵ_z (cm)			max ϵ_z (cm)			RMS $\epsilon_{d\theta}$ (deg)			max $\epsilon_{d\theta}$ (deg)		
0	-0.3	**-0.3**	-1.2	0.3	**0.3**	6.7	0.5	**0.5**	19.8	0.9	**0.0**	3.8	3.3	**0.0**	11.2
$\Delta E = -10\%$	+15.2	+15.2	**-0.6**	15.7	15.1	**5.2**	27.4	16.0	**15.6**	4.0	3.6	**2.4**	15.4	**6.9**	8.8
-5%	+5.9	+5.9	**-1.0**	5.7	**5.8**	6.0	6.8	**6.1**	17.6	3.2	**1.5**	3.2	6.0	**2.6**	9.1
+5%	-4.2	-4.2	**-1.5**	4.0	**4.0**	7.6	4.9	**5.3**	23.1	2.9	**1.2**	4.4	9.8	**1.9**	12.9
+10%	-7.4	-7.4	**-2.2**	7.0	**7.1**	8.6	35.9	**12.9**	25.3	4.6	**2.3**	5.2	19.3	**3.5**	15.8
$\Delta k = -10\%$	-7.5	-7.5	**-1.5**	7.8	7.5	**6.9**	9.8	**7.9**	26.4	1.5	**0.6**	5.1	4.8	**1.0**	14.6
-5%	-3.7	-3.7	**-1.4**	3.9	**3.7**	6.8	4.8	**3.9**	22.8	1.1	**0.2**	4.3	5.2	**0.5**	13.0
+5%	+3.6	+3.6	**-1.3**	3.7	**3.9**	6.8	4.6	**4.1**	17.7	0.3	**0.3**	3.3	1.4	**0.5**	9.9
+10%	+7.2	+7.2	**-1.2**	7.4	7.7	**6.9**	8.4	**8.4**	16.4	0.6	**0.4**	2.9	2.7	**0.8**	8.5
$\Delta\alpha = -2^\circ$	+10.9	+10.9	**-1.5**	10.1	9.8	**7.1**	16.8	**11.0**	16.8	2.4	**2.3**	4.2	7.6	**3.9**	13.7
-1°	+5.0	+5.0	**-1.4**	4.9	**4.5**	6.8	6.2	**5.1**	24.2	0.9	**1.0**	4.0	3.1	**1.8**	12.4
$+1^\circ$	-4.4	-4.4	**-1.3**	4.2	**3.9**	6.7	5.0	**4.4**	17.2	2.6	**1.0**	3.6	8.9	**1.6**	9.9
$+2^\circ$	-8.1	-8.1	**-1.3**	7.6	6.9	**6.8**	9.8	**8.0**	15.3	3.9	**2.2**	3.3	12.2	**3.1**	8.7
$\Delta\beta = \pm 1^\circ$	-0.1	**-0.1**	-1.3	2.8	**1.7**	6.3	7.1	**2.7**	22.2	1.7	**0.8**	4.2	5.2	**1.0**	11.5
$\pm 2^\circ$	+0.3	**+0.3**	-1.4	5.0	**3.3**	7.8	14.4	**4.9**	27.7	2.3	**1.6**	4.2	5.5	**1.9**	15.4
$\Delta t = 15$ ms	+3.9	+3.9	-	3.9	4.9	-	4.9	6.4	-	1.5	1.5	-	5.1	2.8	-
30 ms	+9.3	+9.3	-	9.2	9.7	-	13.3	10.9	-	2.2	3.2	-	10.1	5.3	-

Table 3.1: Sensitivity of tracking on flat ground. Maximum and root-mean-square errors in tracking height (ϵ_z) and deflection ($\epsilon_{d\theta}$) are shown for the deadbeat, turn-rate constrained deadbeat (Sec. 3.2.3), and classical controllers (Sec. 3.2.3). "Steady state" refers to straight-line motion with $z_i = z_{i+1}$. Smaller errors between constrained deadbeat and classical control are in bold.

The model would have stumbled in this next step if there had been a rise in ground level of $\Delta z_g > 19$cm. Thus, (3.10) defines the actual stability margin in the example.

Besides increasing the target height to create a bouncier gait, the stability margin can also be improved by decreasing the initial angle of attack. In fact, at $t = 0$ the control tables allow a minimum of $\alpha = 45^\circ$ that aligns with the friction limit (Fig. 3.3). As $l_0 \sin 45^\circ = 0.71$m, a modified time policy that starts with this angle will, at the expense of an unavoidable deflection in the largest upward steps, increase the stability margin by 10cm to $\Delta z_{max} \sim \pm 29$cm, or about 30% of the leg length.

Sensitivity to Changes in Dynamic and Control Parameters

In further simulation experiments, we test the sensitivity of the time-based control strategy to systematic errors in model parameters. Like the system energy E_{sys}, a change in any other element of the parameter vector $\hat{\boldsymbol{p}}$ causes a deviation in the behavior function $f(z_i, \boldsymbol{u}, \hat{\boldsymbol{p}})$, producing height and deflection errors. Although $\hat{\boldsymbol{p}}$ includes five parameters, we take advantage of the fact that they collapse to only two independent parameter groups in dimensional analysis (dimensionless system energy $\tilde{E}_{sys} = \frac{E_{sys}}{mgl_0}$ and stiffness $\tilde{k} = \frac{kl_0}{mg}$) [54], and we only consider changes in system energy ΔE and leg stiffness Δk to characterize the sensitivity of the leg placement strategy to systematic errors in the dynamic parameters. In addition, we consider systematic errors in the control with regard to the positioning accuracy of the leg ($\Delta\alpha$ and $\Delta\beta$) and the update frequency of the control loop (time delay Δt). For each parameter, we first repeat the random waypoint navigation experiment over flat ground to isolate its effect on the control performance and then investigate its influence in rough terrain locomotion.

Table 3.1 summarizes the results of the sensitivity analysis on flat terrain. Without any parameter disturbances, very small tracking errors in height and deflection persist due to the discrete representation of the behavior function f (maximum errors $\leq$0.5cm and $< 3.5^\circ$ for the sharpest turns). With parameter disturbances, the model is most sensitive to systematic changes in system energy and the angle of attack, producing maximum errors of 36cm and 20° ($\Delta E = 10\%$) and 17cm and 12° ($\Delta\alpha = 2^\circ$). On average, however, tracking errors stay much smaller. Moreover, if the controller is constrained to make wider turns, the maximum errors are reduced substantially (commanded deflections of $|d\theta^*| > 20^\circ$ are truncated to a maximum turning rate of $d\theta^* = \pm 20^\circ$ per step for 'constrained deadbeat' in Tab. 3.1), demonstrating that accuracy in modeling and tracking matter the most for aggressive turning maneuvers.

As a characteristic example of the influence of systematic parameter changes on control performance during rough terrain locomotion, Figure 3.7 shows the combined effects of systematic errors in energy estimation and energy disturbances from rough ground. As mentioned in sec-

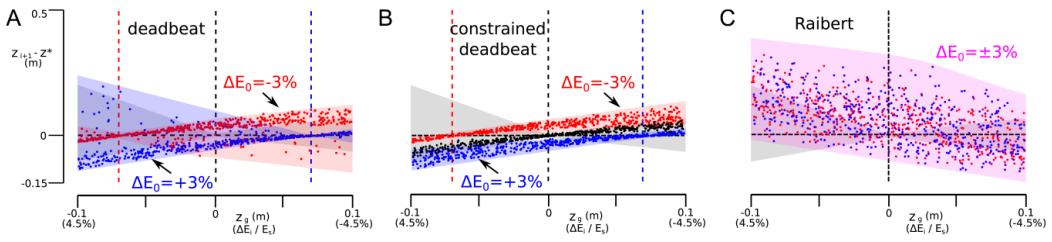


Figure 3.7: Sensitivity of tracking apex height on rough ground to systematic changes in energy $\Delta E = \pm 3\%$ (blue/red dots). Tracking errors are shown as a function of ground level. (**A**) Time-embedded deadbeat control. The gray envelop outlines the tracking error of the unperturbed model (Fig 3.5C). Systematic model errors produce systematic shifts in tracking error including the zero error height (vertical lines). (**B**) Same control limited to maximum turning rate of 20 deg per step. The black dots show the tracking error of the corresponding unperturbed model. (**C**) Classical Raibert-style leg placement control (limited to 20 deg per step to avoid falling). Tracking errors remain large but nearly unaffected by systematic modeling errors (compare to red dots in Fig. 3.8C). Similar effects are observed in the deflection errors $\epsilon_{d\theta}$ (not shown).

tion 3.2.3, changing the initial system energy has the same effect as changes in ground height; it re-centers the tracking errors about a new ground height where the step change ΔE_i in energy cancels the systematic error, $\Delta E_i = -\Delta E$. For example, with a systematic change of $\Delta E = \pm 3\%$, the control is perfect at a ground height of $z'_{g,i} = \pm 7$cm and the energetic errors encountered now range in $[-1.5\%, 7.5\%]$ (blue dots in Fig. 3.7A,B) and $[-7.5\%, 1.5\%]$ (red dots) instead of $[-4.5\%, 4.5\%]$ (shaded gray area in A and black dots in B). Apart from this horizontal shift along the height or energy axis, the shape of the original tracking error function (gray regions in Figs. 3.5C and 3.7) remains unchanged. (Other systematic parameter disturbances have similar effects on rough terrain locomotion, systematically deforming the original error function. Their plots are omitted here.)

The results of the sensitivity analysis show that for an implementation in running robots, automatic model calibration may be required as demonstrated by [16] for a planar hopper that derives its control from the deadbeat control of the planar spring-mass model. The clearly systematic trends in the steady state errors for each disturbance suggest that such tuning will be effective.

Comparison to Classical Leg Placement Strategy

Classical approaches to the control of hopping or running robots often rely on the Raibert controller [146, 215]. This controller has three components including a thrust law for energy regulation in stance, a stance-phase attitude control for the robot center body, and a leg placement control in flight for the generation of gait and turning. To compare the quality of the control proposed here to existing control approaches used in legged robotics, we focus on the leg placement component of the Raibert controller (the spring mass model is energetically conservative and has a point mass body without pitch) and redo the robustness and sensitivity tests.

The leg placement of the classical controller uses the horizontal velocity $\dot{\boldsymbol{x}} = [\dot{x}\ \dot{y}]^T$ and the duration T_s of the previous stance to estimate a "neutral foot point" for a symmetric stance phase, and then applies a linear correction about this point based on the desired velocity $\dot{\boldsymbol{x}}_d = [\dot{x}\ \dot{y}]_d^T$. The resulting leg placement is given by

$$\boldsymbol{x}_f = \frac{\dot{\boldsymbol{x}} T_s}{2} + k_{\dot{x}}(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_d), \tag{3.11}$$

where $\boldsymbol{x}_f = [x\ y]_f^T$ is the horizontal distance vector of the foot point from the hip joint, and $k_{\dot{x}}$ is a proportional gain. Within the theoretical framework developed in this paper, $\boldsymbol{x}_f$ is equivalent to the control input $\boldsymbol{u} = [\alpha\ \beta]$, $\dot{\boldsymbol{x}}$ describes the x-direction of the apex frame and an initial apex height $z_i = \frac{E_{sys}}{mg} - \frac{|\dot{\boldsymbol{x}}|}{2g}$, and the pair ($|\dot{\boldsymbol{x}}_d|$, $\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}$) translates into the desired state $(z^*, d\theta^*)$. Thus, (Eq. Eq. 3.11) can be interpreted as a policy for approximating the behavior function f by linearizing it about the points where $z_{i+1} = z_i$ along the zero turning axis ($\beta = d\theta = 0$, Fig. 3.2). While the neutral point adapts based on the previous stance phase, the slope $k_{\dot{x}}$ of the linearization remains constant, requiring a single compromised approximation at different locations of f.

The comparison of the two leg placement strategies shows that the time-embedded leg placement strategy enables much sharper turns and tracks the target state faster and more accurately (Fig. 3.8). By design, the time-embedded deadbeat control (black dots) generates virtually no deviation from the target state for running and turning on flat ground (Fig. 3.8B). In contrast,

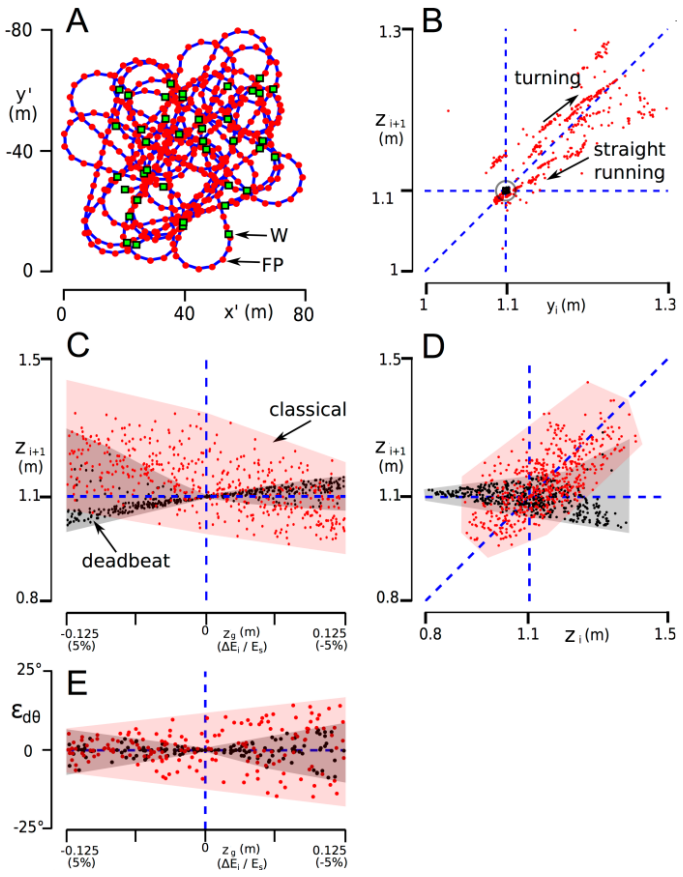


Figure 3.8: Comparison to classical leg placement strategy. (**A**) Random navigation trail example generated by the leg placement strategy of the Raibert controller with limit on turning rate of 20 deg per step to avoid falling. (**B**) Apex return maps generated from running and turning on flat ground for the classical strategy (red dots) and the time-embedded deadbeat strategy (black dots in a very tight cluster). Steady state locomotion corresponds to dots on the diagonal line ($z_{i+1} = z_i$). (**C**) Errors in apex height for running and turning in rough terrain as a function of ground height or system energy. (**D**) Corresponding rough-ground return maps. **(E)** Errors in deflection over rough terrain as a function of ground height or energy.

the classical approach (red dots) needs to be constrained to a maximum turning rate of 20 deg per step to avoid falling (compare Figs. 3.8A and 3.5A), gets attracted away from the target state during turns because the linearization gain $k_{\dot{x}}$ cannot simultaneously fit turning and straight-line running, and only slowly returns to the target state after turns due to a slanted return map (Fig. 3.8B). This difference in performance also shows on rough ground where the deviations from the target state are large and the return to the target state remains slow (Figs. 3.8C,D) for the classical control. Likewise, the directional tracking is also looser with the classical control (Fig. 3.8E).

For systematic errors in the model parameters, the performance of the classical control does not change significantly. In general, it is still worse than that of the time-embedded strategy with the same constraint of 20 deg per step on the maximum turning rate (rms and max error values in Tab. 3.1) due to the slower speed of recovery in the return map. However, unlike the time-embedded strategy, the classical control shows no large drift from the target state in steady locomotion (steady state errors in Tab. 3.1) due to the feedback properties of (Eq. Eq. 3.11), which tend to correct for unmodeled behavior.

3.2.4 Summary

Using a compact representation of the 3D spring mass model's dynamics, we have derived and demonstrated in simulation a time-based leg placement strategy for highly robust running and steering in uncertain 3D environments. The identified control strategy generalizes the time-based control derived for the planar spring mass model [169, 171] and allows a legged system to navigate very rough terrain (up to 30% of leg length in the presented example) without feedback about ground disturbances (Fig. 3.4). It is nearly deadbeat with respect to stabilizing the apex height and heading direction, as long as the relative disturbances in system energy and the other model parameters stay small (Figs. 3.5 and 3.7, Table 3.1). Moreover, it outperforms a classical leg placement strategy in terms of turning rate and disturbance rejection (Fig. 3.8).

The compactness and performance of the proposed control strategy stems in large part from separating the control into feedback detection of the direction of motion and feedforward compensation for unknown ground disturbances. Like previous work on the control of 3D spring mass running, the leg placement strategy developed here requires state feedback about the direction of motion. In [166] and [27] this feedback information is used explicitly. Although [134] find that control relative to the velocity vector in flight yields inherent stability without the need for corrective strategies, it depends on measuring this state at least intermittently. The control developed here requires a similar intermittent feedback. At every apex during flight, the direction of motion needs to be known to align the apex frame. However, the robustness to ground level disturbances is realized with feedforward control. Instead of correcting for past disturbances, the time-indexed control adapts the leg placements to accommodate ground variations in the upcoming step by taking advantage of the relationship between falling time and falling height. The resulting control strategy reveals how highly robust running and steering could be achieved in uncertain environments.

3.3 Conclusions and future work

In this chapter, we extended the theory of deadbeat spring-mass model (SMM) running to 3D. Once per stride, we use the 3D SMM to compute the optimal leg placement that matches the next apex state as closely as possible in terms of direction of motion and apex height. If the terrain height is unknown, arranging the different solutions from different initial conditions by their time of touchdown yields a time law for leg placement that provides near deadbeat stability on unobserved uneven terrain. Prior to this work, deadbeat leg placement had only been shown for the 2D SMM.

In addition, we analyze the sensitivity of this system to various disturbances and compare the performance to classical leg placement policies. Simulation confirms that the deadbeat control law converges significantly faster than heuristic linear feedback that many running robots

currently use [2, 146], even when the latter is optimally tuned to exactly match the slope of the behavior map. This is because the behavior map is not well described with a linearization, though it is low dimensional and easily stored in a look-up table. Conceptually, this demonstrates that the role of leg placement in the dynamics of legged system is non-linear, and that numerical tabulation is an effective way to account for this non-linearity.

In the bigger picture of designing control for legged robots, we use the simplified model to plan motions that satisfy our gait objectives and are likely to obey the dynamics of the actual robot. Using models with as few parameters and degrees of freedom as possible generally makes the planning problem easier and more intuitive. However, there are two distinct costs to making such simplifying reductions.

First, by constraining the simplified model to a limited set of behaviors, the planner may be overlooking potentially better solutions. A force-controlled system like ATRIAS (details in Chapter 4) can produce a wider range of force profiles than the fixed spring stiffness described by the spring mass model. However, the results in this chapter show that even within the overly restrictive terms of this simplified model, highly agile and robust running can be attained; we do not need a more descriptive and computationally expensive model to produce highly dynamic behaviors.

Secondly, a simplified model is often a lower order representation that omits modes from the full order system. In this case, the spring mass model does not describe angular momentum or changes to total system energy. The behavior of these modes then have to be addressed by supplementary control, which may interfere with assumed dynamics of the simplified model. Chapter 6 is largely dedicated to studying these interactions. Alternatively, the model can be redefined more descriptively to explicitly account for the additional dynamics with additional authority (which makes the planning problem harder). In [203, 205], the authors show that using different spring stiffnesses for each half of stance provides good control authority over the system energy, and in [45], the authors effectively deadbeat stabilize the energy by solving the boundary-

value problem for a polynomial force profile. The costs and benefits of using these alternative models can be an interesting topic for future work.

In the context of the overarching questions asked in the beginning of this thesis, these results show that the low order dynamics of the 3D spring mass system have very promising properties for agile and robust running. Despite the constraints imposed by the fundamental contact wrench limitations of legged systems, the model is nonetheless capable of executing highly dynamic motions; the two free parameters of leg placement in each flight phase can be mapped to deadbeat tracking of apex height and direction of motion. Our analysis shows that friction limitations and geometric limitations leave a reasonably large basin of attraction for deadbeat control (Figure 3.3), and even outside of this basin, the apex height can still be deadbeat stabilized while tracking less steep turn angles.

The remaining chapters in this thesis attempt to answer how well these "promising properties" carry over to real legged robot.

Chapter 4

Overview of the ATRIAS robot

The ATRIAS robot is our vehicle for testing the theory of spring-mass running. This chapter presents its basic properties and their implications on our problem formulation and our overall approach to control design.

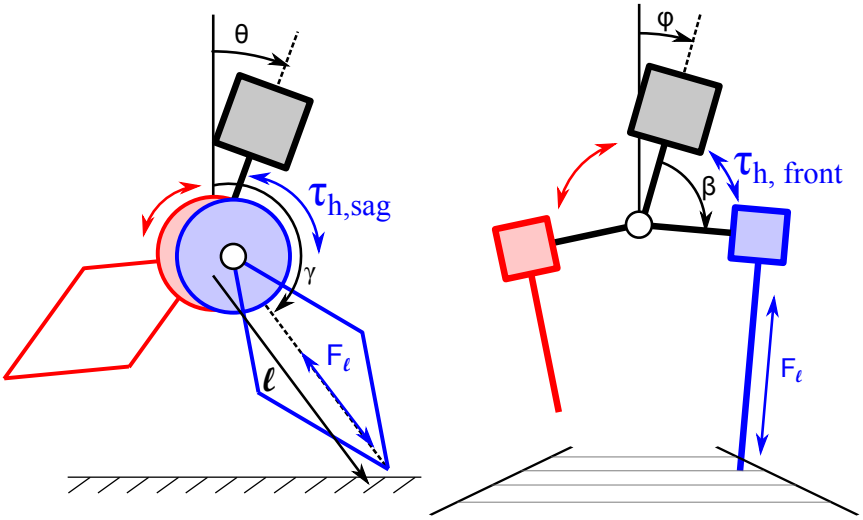
Section 4.1 first describes the construction of the robot. Section 4.2 explains high level choices that we make based on ATRIAS's properties. Specifically, we limit the current implementation to the saggital plane, we decouple the force-controllable rigid body dynamics from the actuator dynamics by formulating series elastic actuation, and we define our problem as an optimal control problem in the space of the centroidal dynamics. Section 4.3 then presents the equations of motion for the robot and for the actuators, which are used throughout the remaining chapters of this thesis. Section 4.4 derives the feasible spring-mass behaviors from ATRIAS's power and kinematic limits. Finally, Section 4.5 gives a concise chapter summary.

4.1 Physical properties

We work with a copy of the ATRIAS robot [61, 149], designed and built by Dr. Jonathan Hurst's lab at Oregon State University. The name ATRIAS stands for Assume This Robot Is A Sphere; the robot was designed to closely match the simplified spring mass model with the intention

(a) photo of ATRIAS from the side.

(b) photo of 5-bar mechanism and leaf springs in leg.



(c) diagram of ATRIAS.

Figure 4.1: Bipedal robot ATRIAS. ATRIAS [61, 149] is a bipedal robot with light, compliant legs. The five-bar mechanism allows the actuation to be located at the hip, minimizing the mass and inertia of the legs. Serial springs in the five-bars provide torque sensing and allow closed-loop torque control in the saggital plane. Each leg has actuated three degrees of freedom with respect to the trunk, but no ankle actuation.

of demonstrating the robust and energetically efficient locomotion that the theoretical model exhibits [46, 97, 180]. Towards this end, ATRIAS's mass (68 kg total) is concentrated near the hip and the legs are kept very lightweight (2.3 kg each). Furthermore, serial springs in the legs add compliance to the system.

Each leg is composed of carbon fiber tubes arranged in a five-bar structure (17 and 95 cm at min and max extension). See Figures 4.1(a) and (b). (Note that this structure of four links is termed a "five-bar" due to the independent actuators at the trunk, which makes the trunk a fifth bar with zero length). This structure allows the actuation and gearing to be placed at the hip, minimizing the rotational inertia about the center of mass. A pair of 200 A DC brush-less motors sits at each hip. When the motors are driven in the same direction, the leg is swung forward or backward (relative angle $\gamma - \theta$ in Fig. 4.1) in the sagittal plane. When they are driven in opposing directions, the leg is lengthened or retracted (length l in Fig. 4.1). Each motor is linked to a 50:1 harmonic drive that is connected to a leaf spring. Since the springs are in series with the motors, they do not provide the passive torques and forces that parallel springs would. Instead, they can be viewed as the compliant elements within series elastic actuators (SEAs) [137] (see Section 4.2.2 for a more detailed review of SEAs). The deflections in the springs describe the torques applied to the five-bar, and the geared output of the motors serve as inputs that drive the spring deflections. This closed loop force control is made possible by high accuracy encoders measuring the joint angles on either side of each spring and Hall effect sensors inside the motors. Through the geometry of the five-bar, the coupled output of the pair of spring deflections in each leg maps to an axial force F_l (Fig. 4.1) along the leg and a rotational torque $\tau_{h,sag}$ between the leg and the hip. This transmission of torques and forces is detailed in Section 4.3.4.

Laterally, each hip is actuated by a 60 A DC brush-less motor connected to a 57:1 belt drive system located in the trunk. Each hip's range of motion spans from $-80 \leq \beta \leq +110$ (Fig. 4.1) degrees measured from the trunk. Since the pair of motors and pair of harmonic drives provide a total of 20 kg centered on each hip at 17 cm from the pelvis, their inertial effects are

non-negligible in the frontal plane.

The legs terminate at rounded, point feet. When on the ground, this constrains the GRF to act through the point contact with no applied rotational moments. For 3D locomotion, they can been augmented with passive ankles and 13 cm feet, which serve to resist yaw moments. Besides the yaw resistance, the passive ankle joint renders the rest of the contact dynamics of the foot identical to those of the original point foot.

4.2 Fundamental assumptions, overall approach, and related background

Based on ATRIAS's construction, we first make three high level choices that define our overall problem formulation.

4.2.1 Sagittal plane instead of 3D

We develop the control theory towards that eventual goal of achieving agile running in 3D. In Chapter 3, we derive stabilizing leg placement for the dynamics of the 3D system. The process of using floating base equations of motion in Section 4.3 to compute inverse dynamics is equally valid in 2D or 3D, and the state-space optimal control we use in Chapters 5 and 6 for designing feedback for an under-actuated system readily accepts the degrees of freedom and control inputs of the full 3D system.

However, within this thesis, we focus on 2D locomotion on the hardware and use a boom to constrain robot to move in only the sagittal plane. This decision was based on primarily on two observations of ATRIAS's frontal plane capabilities. In particular, without springs on the lateral motors, we cannot use SEA control to close the loop on desired torques at these joints, while relying on open loop motor torques would significantly lower the effectiveness of a pure force control formulation. Furthermore, the relatively weak structure of the leg with respect to lateral

loads (the knees have built-in fuses that break in this direction, which also introduce unmodelled compliance) significantly limits the application of active control for dynamic motions in the frontal plane.

For these reasons, our experimental demonstration of agile spring-mass running is limited to the sagittal plane. We also accordingly focus the theoretical analysis of the closed loop dynamics on this 2D system such that we can anchor the analysis in our experimental data. Nonetheless, the techniques we develop in this thesis for tracking the desired closed-loop reference trajectory with optimal control, analyzing the full order error dynamics, and propagating the theoretical expectations of disturbances generalize to robot running in 3D.

4.2.2 Series Elastic Actuators (SEAs) as torque sources

Series Elastic Actuator (SEA) control [137, 142, 145] has become an increasingly popular and effective method for implementing closed loop force and torque control on robotic systems, and ATRIAS's construction is well suited to take advantage of this approach. This then allows us to model idealized joint torques when formulating the dynamics of the robot. (We do not expect perfect torque generation by the SEA control, and Chapter 7 is largely dedicated to studying the effects of these tracking errors).

Highly geared electric motors like the ones on ATRIAS suffer from high impedance from reflected inertia and high stiction [142], which would effectively adds significant actuator dynamics and complicate the concept of directly applying joint forces or torques if the geared output were directly connected to the leg. Series elastic actuation [137, 142, 145] significantly dimishes these effects and has allowed roboticists to apply "ideal" force and torques sources as control inputs at the joint level. In an SEA, the motor and the load (i.e, the two sides of the joint) are separated by a compliant element such that Hooke's law yields the force or torque applied across the joint. An inner loop controller is designed for using the motor to close a loop around the spring deflection, thereby resulting in a joint-level actuator limited only by the relatively high bandwidth and

tracking accuracy of the inner loop SEA control.

Over the last two decades, the performance of SEA control has seen continuous improvement, and SEAs have been widely utilized as high fidelity torque-sources on legged robots including Spring Flamingo [140], M2V2 [139], Hume [176], COMAN [197], ScarlETH [76], TORO [44], THOR [73], and Valkyrie[127]. In these implementations, the outer loop that commands forces or torques and the inner loop SEA control must be designed to interact stably. [73] provides a comprehensive description of the state of the art, and here we summarize some of the key results. [76, 160, 192, 198, 212, 213] each closed a loop around the motor velocity to reduce sensitivity to stiction and backlash, [198] studied passivity for stability guarantees, [73, 99, 126] each applied disturbance observers to further improve the inner loop performance, [160] implemented a multi-input formulation that uses a filtered estimate of load velocity as a feed-forward signal.

By design, the resulting closed loop SEA system approximates an ideal actuator such that at the joint and rigid body level, the dynamics of the robot are separated from the internal dynamics of the SEA – the robot equations of motion treats joint torques as inputs. There are many reasons to use this decoupling as long as the approximation is valid.

By assuming ideal joint torques, the equations of motion take on the standard form used ubiquitously on humanoid robots. This allows the direct use of the force control and inverse dynamics formulations detailed in Section 2.2.2, which have proven highly effective for tracking the centroidal dynamics of the robot and demonstrating stability. In particular, since these second-order equations of motion take joint torques as the input, the contact wrench is also at the input level. Therefore, the input can be mapped directly to the GRFs, which makes it easy to formulate control that obeys the contact wrench constraints; [36, 102, 156, 204] each implement efficient constrained optimization algorithms that explicitly preserve the stability of the ground contact. If the equations of motion were to describe a higher order control input with non-negligible actuator dynamics, any contact wrench constraints would then need to be applied in a significantly more expensive constrained trajectory optimization problem, which is much more challenging to

implement as a real-time controller. Similarly, these second order inputs facilitate a direct translation of the spring-mass model, which is naturally formulated as a force-driven second order system.

This decoupling assumption allows us to study the theory of spring-mass running in a less platform-specific manner. All robots have centroidal dynamics and ground contact wrenches; most modern humanoid robots are force controllable; but few robots share ATRIAS's specific leg construction. Using the SEA formulation replaces this last level with a generic torque source, allowing our results to directly generalize to other robots. Similarly, here our goal is to study the general transfer of spring-mass theory onto higher order systems, and separating the actuator control from the remaining robot dynamics clarifies the scope of this study.

On a practical level, separating the rigid body dynamics from the actuator control into cascaded systems allows a potentially high bandwidth inner loop to effectively eliminate the effects of uncertainties and disturbances in the actuator dynamics. Just as velocity-based SEA control [160, 192, 198, 212] have leveraged Hall effect sensors and high bandwidth embedded controllers to reduce the sensitivity of the motor control to stiction and backlash, high bandwidth inner loop SEA control reduces the sensitivity of the outer loop robot behavior to any modeling errors of the motor-to-spring-to-load dynamics. Finally, also on a practical level, we can update the architecture of the SEA control based on the newest studies to improve our performance without needing to revise the structure of our overall theory.

On the other hand, some researchers [63, 79, 119, 182] have chosen to directly model any serial compliance in the full equations of motion without formulating a separate "actuator" control. Thus, the control inputs are motor velocities or motor torques, instead of idealized joint torques. Fundamentally, the biggest advantage of this approach lies in avoiding the potentially erroneous assumption that the robot dynamics and the motor approximately decouple across the spring. That is, if the closed loop tracking bandwidth of the SEA control is not significantly higher than the desired bandwidth of the outer loop control, these two systems can have destabilizing inter-

actions. Effective control design would need to account for the slow dynamics of the "actuator" in conjunction with the rigid-body dynamics of the robot.

In our work so far, we have only explored control designs with limited outer loop gains that avoid interaction with the inner loop SEA control. Comparing these results to controllers derived from a non-cascaded model of the full order system can be an interesting avenue for future research.

4.2.3 Optimal control for the higher order system

It is clear that ATRIAS has more degrees of freedom and wider range of available actuation than the simplified spring-mass model – ATRIAS has rotational inertia, non-constant system energy, and can produce GRFs that do not match the virtual linear spring. How to handle the degrees of freedom omitted by the simplified model is a recurrent question in the literature on model-based walking and running control, and the existing solutions can be approximately partitioned into three schools of thought for this issue.

Many roboticists have applied heuristic based "template-anchor" arguments [7, 13, 31, 52, 53, 132, 200]. The simplified model is treated as the "template," which defines a set of desired dynamics. The "anchor" is a higher complexity model that more accurately describes the specific real system, and intuitive control policies are applied such that the dynamics of the anchor are made to embed that of the template. These policies typically directly stabilize the extra degrees of freedom while the desired properties of template are approximately retained. Through design of these embedding policies, researchers have successfully demonstrated stabilized systems in simulation and on hardware. However, quantifying the performance or formalizing the convergence of these systems is often challenging.

Our early work [211] towards embedding deadbeat spring mass running on ATRIAS was in fact closely aligned with this approach. We reasoned that the two main modes omitted by the spring mass model were the total system energy and the pitching of the trunk. We added a virtual

damper to the virtual spring force of the SMM to allow the injection or removal of energy in each step, and we augmented the torque control at the hip with PD feedback to stabilize the pitch of the trunk. While this produced reasonable results in simulation [211] and gave us our first stable running experiments, we could not make theoretical claims of performance or stability, and we did not have access to the ground contact constraints (due to the additional torques at the hips).

In the second school of thought, control based on computing the hybrid zero dynamics (HZD) of a system offers a formal analysis of local stability [28, 135, 136, 182, 206] of the full order system. This framework provides a mathematical structure well suited for studying the empirically demonstrated stability of the "template-anchor" implementations. The dynamics of the lower order system define the zero dynamics, which are parameterized by virtual constraints applied to the higher order system. These constraints can be chosen to embed a simplified system like the spring mass model [135, 136], or they can be designed through optimization [182]. Input-output linearization yields continuous feedback control that enforces these constraints, and a discrete control (that includes the leg placement) is designed to stabilize the zero dynamics. This outer layer of control can be motivated by heuristics including Raibert-style velocity control [136] or coupling torso pitch to velocity [182], or it can be designed through optimal control with discrete LQR. Finally, local stability of the resulting system is checked using Poincare analysis (discrete LQR guarantees stability if it is used). [28] showed that under certain conditions, Poincare stability on the reduced order model implies local asymptotic stability of the full order system.

While this formulation is closely related to our study of spring mass running on a higher order robot, on its own it does not provide a complete picture of the full capabilities of a force controlled robot. The continuous dynamics of the full order system may be coupled in a way that allows more effective error rejection than defining and enforcing virtual constraints while implicitly reserving other error modes to be stabilized by the discrete control. That is, it may be possible to apply a combination of continuous and discrete feedback strategies to better track the multi-modal dynamics of the system.

Continuous optimal control (reviewed in Section 2.3.1) describes the third school of thought for how to address the "extra degrees of freedom." Given a model of the dynamics of the full order system, the control problem is explicitly posed as a search for the fastest and cheapest convergence of all the state errors, with no explicit distinction between the modes represented in and those omitted by the simplified model. Solving the the optimal tracking problem thus provides an explicit answer to how the full order system can best produce the desired motion. For linear systems with quadratic costs, the Ricatti equation returns the optimal control as a linear feedback law (LQR), while nonlinear systems can be locally stabilized by applying LQR about the reference condition. This widely used result and the proven success of other optimal control techniques throughout the locomotion literature (Section 2.3.1) lead us to hypothesize that we can design effective control that exploits the behavior of the closed-loop simplified model while stabilizing all the modes of the full order system by formulating and solving the optimal control problem.

4.3 Model and dynamics

In this section, we present our model of ATRIAS, the actuators, and their equations of motion. The resulting equations are the foundation of our model-based control design; from these derivations, we extract both the simplified centroidal dynamics that we use for optimal control as well as numerically computed matrices that we use for inverse dynamics. Furthermore, these equations are required for deriving ATRIAS's capabilities as a function of power limitations (Section 4.4.1).

4.3.1 Floating base robot and dynamics

We define a planar robot model (Fig. 4.2) that captures the essential characteristics of ATRIAS. It has a trunk of mass $m = 63.4$ kg and inertia $I_t = 2.2$ kg m^2 about the center of mass, which

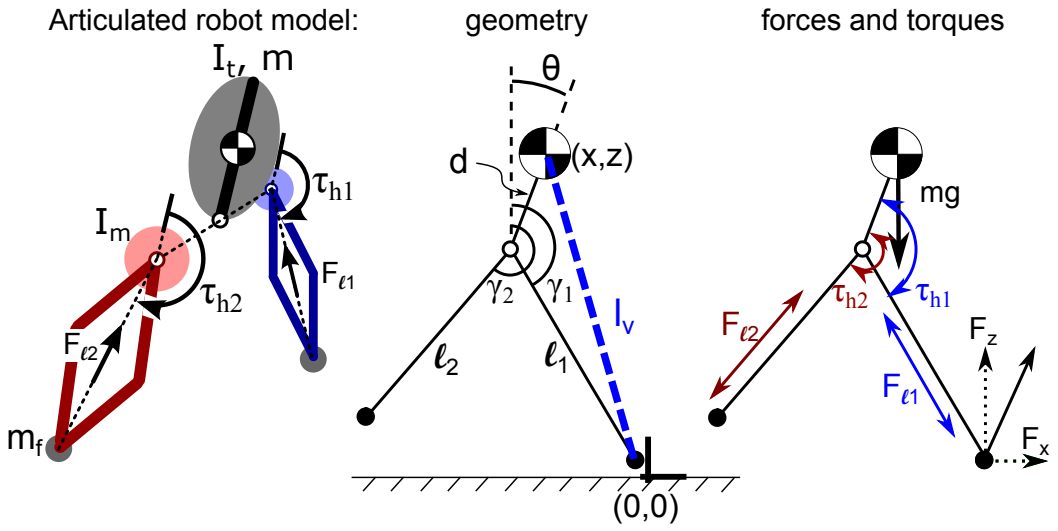


Figure 4.2: Rigid body model for equations of motion. The planar robot is represented by a rigid body trunk above the hips, rotary actuators $\tau_{h,i}$ at each hip, linear actuators $F_{l,i}$ in each leg, a small inertia I_m in each hip, and a small mass m_f in each foot. A virtual leg can be described between the trunk CoM and the stance foot. External forces F_x and F_z applied to the foot map to the robot states through a Jacobian matrix.

is located distance $d = 0.19$ away from the hip joint. The effect of each five-bar leg mechanism is represented by two components: a rotary actuator with outputting torque $\tau_{h,i}$ on disc with a small inertia $I_m = 0.3$ kg m^2 at the hip, and a linear actuator outputting force $F_{l,i}$ that extends the leg with a small point mass $m_f = 2.4$ kg at the foot.

This system has three degrees of freedom for the floating base [154] and two more for each leg. Here, the position vector $\mathbf{q} = [x, z, \theta, \gamma_1, \gamma_2, l_1, l_2]$ consists of the coordinates x, z and θ describing the global position and orientation of the trunk, along with γ_i and l_i describing the global orientation and length of each leg (Fig. 4.2). The concatenation $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$ gives the full state of the system. These floating base coordinates make no assumptions about ground contact. A leg in stance would introduce two translational constraints that reduce the model to a five degree of freedom system.

Through defining the kinetic energy, potential energy, and the generalized forces and torques acting on the system, we apply the method of Lagrange [58, 120] to produce the equations of

motion

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{h}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}\boldsymbol{\tau} + \boldsymbol{J}(\boldsymbol{q})^T \boldsymbol{F}$$

where $\boldsymbol{M}$ is the mass matrix, vector $\boldsymbol{h}$ accounts for Coriolis, centrifugal, and gravitational forces, the selection matrix $\boldsymbol{S}$ assigns control inputs $\boldsymbol{\tau}$ to $\boldsymbol{q}$, and the Jacobian $\boldsymbol{J}$ maps external forces $\boldsymbol{F}$ to $\boldsymbol{q}$. The control vector $\boldsymbol{\tau}$ is composed of the hip torque $\tau_{h,i}$ and linear force $F_{l,i}$ of each leg, and the vector $\boldsymbol{F}$ is the external force. The geometry dependent Jacobian $\boldsymbol{J}$ of the foot is given by

$$\boldsymbol{J} = \begin{bmatrix} J_x \\ J_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -d\cos\theta & l\cos\gamma & 0 & \sin\gamma & 0 \\ 0 & 1 & d\sin\theta & -l\sin\gamma & 0 & \cos\gamma & 0 \end{bmatrix}.$$

In running, at most one foot is in contact at a time, so we drop the subscripts i in the Jacobian equation for brevity. We generate the matrices $\boldsymbol{M}$, $\boldsymbol{h}$, $\boldsymbol{S}$, and $\boldsymbol{J}$ symbolically in Matlab.

4.3.2 Stance dynamics

Without loss of generality, we define the origin to be at the foot, such that $x_{\text{foot}} = z_{\text{foot}} = 0$. For a leg in stance, we assume the standard constraint of the ground providing the exact force that prevents motion of the foot. Written as a function of coordinates $\boldsymbol{q}$, this requirement of $\dot{x}_{\text{foot}} = 0$ and $\dot{z}_{\text{foot}} = 0$ implies the acceleration constraints

$$\boldsymbol{J}\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}\dot{\boldsymbol{q}} = \boldsymbol{0}.$$

Combining the floating base equations of motion and constraints into a single system of equations yields

$$\begin{bmatrix} \boldsymbol{M}_{7\times7} \\ \boldsymbol{J}_{2\times7} \end{bmatrix} \ddot{\boldsymbol{q}} + \begin{bmatrix} \boldsymbol{h}_{7\times1} \\ (\dot{\boldsymbol{J}}\dot{\boldsymbol{q}})_{2\times1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{S}_{7\times4} \\ \boldsymbol{0}_{2\times4} \end{bmatrix} \begin{bmatrix} \tau_{h,1} \\ \tau_{h,2} \\ F_{l,1} \\ F_{l,2} \end{bmatrix} + \begin{bmatrix} (\boldsymbol{J}^T)_{7\times2} \\ \boldsymbol{0}_{2\times2} \end{bmatrix} \begin{bmatrix} F_x \\ F_z \end{bmatrix}, \tag{4.1}$$

which relates robot accelerations, actuator inputs, and reaction forces.

Compared to a derivation from a manipulator perspective [120] for a five degree of freedom system, the constrained floating base equations of motion include two extra degrees of freedom that define the two redundant modes. Eq. 4.1 uses larger matrices to compute the same mapping between actuator inputs and robot accelerations, but in doing so it also explicitly computes the GRFs as the constraint forces.

Buried within Eq. 4.1 are the centroidal dynamics:

$$\begin{aligned} m_{\text{tot}}\ddot{x}_{\text{com}} &= F_x, \\ m_{\text{tot}}\ddot{z}_{\text{com}} &= F_z - m_{\text{tot}}g, \\ I_t\ddot{\theta} + I_m\ddot{\theta}_1 + I_m\ddot{\theta}_2 &= -z_{\text{com}}F_x + x_{\text{com}}F_z, \end{aligned}$$

where the center of mass coordinates $[x_{\text{com}},\ z_{\text{com}}]$ are computed by taking the mass-weighted average of the trunk and foot coordinates, and the small hip inertias I_m contribute minimally to the total angular momentum.

By intentional design [61, 149], the post-actuator mass m_f and inertial I_m are very small. (This is not to be confused with the non-negligible reflected inertia $N^2 I_r \approx 3.75$ kg m^2 for SEA control with rotor inertia I_r and gearing N. That is, very little spring torque is required to accelerate the load, but significant motor torque is required to accelerate the rotor behind the spring.) To use force control on the swing leg, we would need derive gains based on non-zero representations of m_f and I_m; force control is not defined for inertia-less dynamics (Section 5.5 explains the position control we apply to the swing leg). However, when we consider the net linear momentum and angular momentum of the robot, the inertial forces from $m_f \ll m$ and $I_m \ll I_t$ have negligible effects. This effectively allows us to drive the swing leg independently of the centroidal dynamics and to directly relate the pitch of the trunk to the net angular momentum.

After eliminating m_f and I_m, the swing leg is independently driven to control $[\gamma_2, l_2]$ using the inputs $[\tau_{h,2}, F_{l,2}]$, such that the coordinates of the robot are reduced to $\boldsymbol{q} = [x, z, \theta, \gamma_1, l_1]$ and a the inputs are reduced to $\boldsymbol{\tau} = [\tau_{h,1}, F_{l,1}]$ for just the stance leg. The dynamics of the reduced

states are given by the system of equations

$$\begin{bmatrix} m & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 \\ 0 & 0 & I_t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \\ \ddot{\gamma} \\ \ddot{l} \end{bmatrix} + \begin{bmatrix} 0 \\ mg \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_h \\ F_l \end{bmatrix} + \boldsymbol{J^T} \begin{bmatrix} F_x \\ F_z \end{bmatrix},$$

$$\boldsymbol{J\ddot{q} + \dot{J}\dot{q} = 0}.$$

where the indices differentiating the legs have been dropped, l is the length of the stance leg, and $F_{x,z}$ are the horizontal and vertical GRFs (Fig. 4.2).

Finally, re-arranging terms in this system of equations yields the intuitive results of

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_t \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -z & x \end{bmatrix} \begin{bmatrix} F_x \\ F_z \end{bmatrix}, \tag{4.2}$$

$$\begin{bmatrix} \tau_h \\ F_l \end{bmatrix} = \begin{bmatrix} l\cos\gamma & -l\sin\gamma \\ -\sin\gamma & -\cos\gamma \end{bmatrix} \begin{bmatrix} F_x \\ F_z \end{bmatrix}, \tag{4.3}$$

where Eq. 4.2 gives the centroidal dynamics and Eq. 4.3 gives the inverse dynamics as a 1-to-1 invertible mapping between the virtual inputs $[\tau_h, F_l]$ and the GRFs. See Figure 4.3. The accelerations $\ddot{\gamma}$ and $\ddot{l}$ can be recovered through the remaining equations, which resolve the constraints. On a more general platform with significant multi-body mass and inertia distributions, forward and inverse dynamics would instead need to be solved from Eq. 4.1.

The derivation in this section thus shows that for ATRIAS, the full rigid body dynamics are nearly equivalent to the centroidal dynamics.

4.3.3 Flight dynamics

When the system is in flight, there are no external forces and no geometric constraints. The equations of motion are simply $\ddot{x} = 0$, $\ddot{z} = -g$, and $\ddot{\theta} = 0$. Each leg can positioned independently

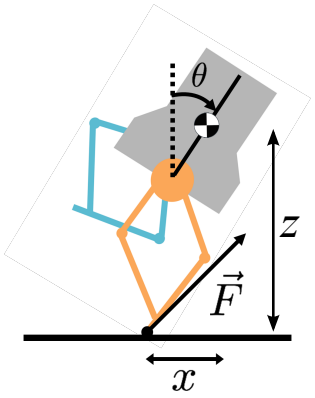


Figure 4.3: In the sagittal plane, ATRIAS's full order dynamics are approximately equivalent to the centroidal dynamics (by neglecting the masses and inertias below the hip). The inverse dynamics is reduced to a mapping that generates $[F_x, F_z]$ in stance using joint torques, and the full order robot model consists of the coordinates $\boldsymbol{q} = [x,\ z,\ \theta]$. See Equations 4.3 and 4.2.

with $[\tau_{h,i}, F_{l,i}]$ driving $[\gamma_i, l_i]$, which does not affect the centroidal dynamics.

4.3.4 Actuation and series elasticity

The virtual inputs $F_{l,i}$ and $\tau_{h,i}$ are directly related to the torques in the two leaf springs on each leg. Neglecting the very low inertia of the carbon-fiber segments, this mapping is determined by the geometry of the five-bar mechanism,

$$\tau_{\text{SEA}} = \frac{\tau_{h,i}}{2} \pm F_{l,i} l_s \sin(\frac{\rho}{2}), \tag{4.4}$$

where $l_s = 0.5$ m is the length of the five-bar segments and ρ is the splay angle between the two proximal segments (Fig. 4.4A).

In each spring (see Figure 4.4B), the joint torque acting between the motor and the load is given by the deflection scaled by the spring constant $k_{\text{SEA}} \approx 3600$ Nm/rad:

$$\tau_{\text{SEA}} = k_{\text{SEA}}(\theta_m - \theta_l).$$

The dynamics of the load angle are the result of a geometric mapping of the robot dynamics derived in Section 4.3. The motor-side angle θ_m is driven by a geared output of the electric

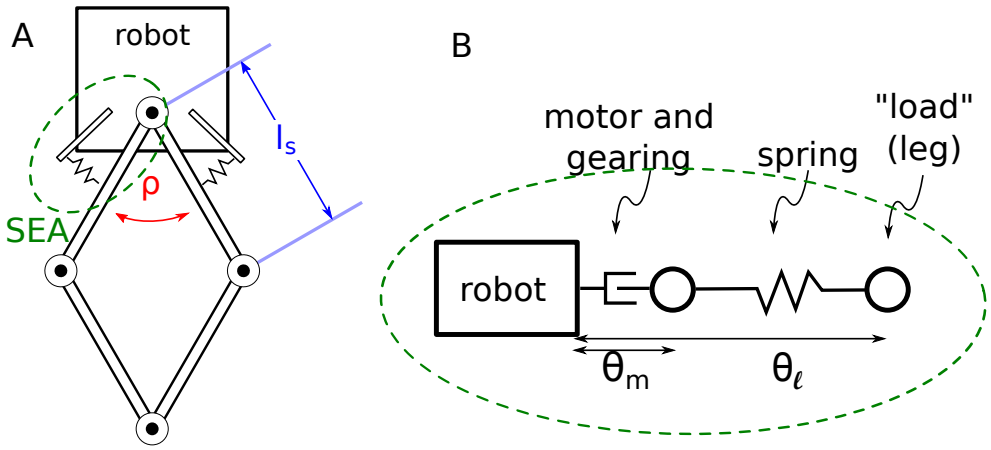


Figure 4.4: Actuation system. The virtual inputs F_l and τ_h have a 1-to-1 mapping to and from the pair of torques applied between the robot body and each the top two segments of the five-bar (A) (Section 4.3.4). This structure with four links is called a five-bar (due to the two independent actuators at the hip) as the robot body defines a link of zero length. The torque between the robot and each of the upper leg segments is given by the deflection in the rotational spring between these two bodies. This system is sketched as a linear spring in (B), and we treat it as a series elastic actuator (SEA) (Section 4.3.4). The geared output of the electric motor is the joint angle θ_m defined with respect to the body (the trunk) of the robot. The spring separates this motor angle from the load angle θ_l, such that the torque between the robot and the load is given by $\tau_{\text{SEA}} = k_{\text{SEA}}(\theta_m - \theta_l)$.

motors, with the harmonic drives providing a gear ratio of $N = 50 : 1$. The dynamics of this system are approximately

$$N^2 I_r \ddot{\theta}_m = \mu N \tau_r - \tau_{\text{SEA}}, \tag{4.5}$$

where the effective motor inertia is dominated by the reflected rotor inertia $N^2 I_r \approx 3.75$ kg m^2, and the effective torque is amplified by the gearing of the harmonic drive but reduced by drive efficiency μ. This efficiency is dynamic; it is 77% when the motor is stalled and loaded, reduces to 58% when the output moves at 2π rad/s, and falls as low as 36% when the load has very low inertia [65].

We take advantage of a high frequency (10 kHz vs 1 kHz) controller on-board the motor amplifier servo drives [41] to close a loop on motor velocities. [76, 198, 212, 213] have shown that this approach can significantly reduce sensitivity to friction, backlash, and modeling error of the actuator dynamics. On ATRIAS, this on-board controller uses an auto-tuner [42] that empirically identifies the plant dynamics between rotor velocity and applied currents. Thus, instead of working directly with the motor dynamics in Eq. 4.5, we effectively see the system

$$\frac{d}{dt}\tau_{\text{SEA}} = k_{\text{SEA}}(\dot{\theta}_m - \dot{\theta}_l), \tag{4.6}$$

where we can command a motor velocity $\dot{\theta}_m^*$ at 1 KHz. Nonetheless, whatever motion is generated is ultimately sourced by applied currents and rotor torques that obey the dynamics of Eq. 4.5.

4.4 Spring-mass gait design within ATRIAS's limits

ATRIAS's power limits and joint angle limits translate into a range of feasible deadbeat running behaviors.

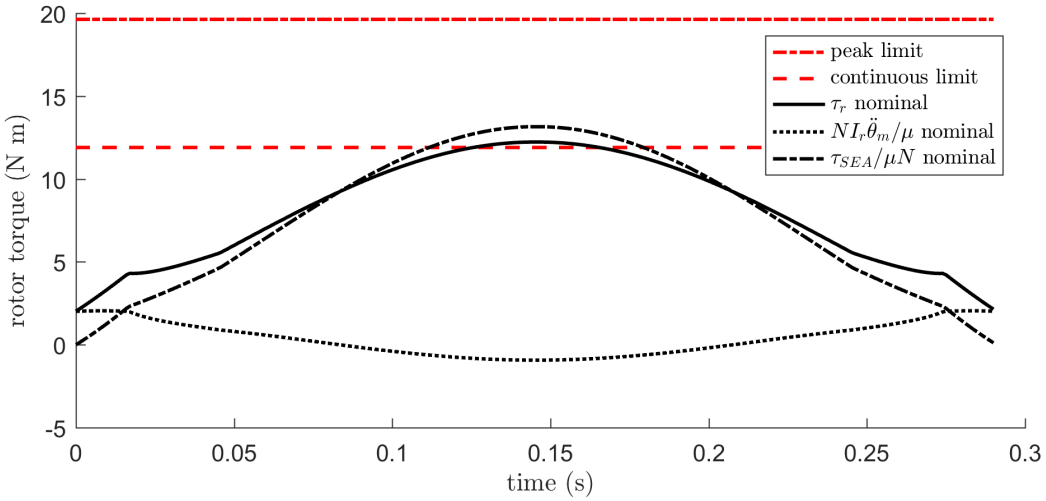


Figure 4.5: A nominal rotor torque profile for a moderate spring-mass gait with $k = 16000$ N/m, $z_0^* = 1.09$ m, and $z_a^* = 1.12$. The solid black line represents Equation 4.7 solved for τ_r and the two dashed black lines are the corresponding inertial and SEA components. The red lines show the continuous and peak rotor torques that can be applied based on motor current limitations.

4.4.1 Maximum torque output

The on-board current control must apply the effective torque $\mu N \tau_r$ to achieve the specified motor velocity $\dot{\theta}_m^*$. In effect, the motor current limits that bound rotor torque τ_r then impose maximum joint torques τ_{SEA} that can be achieved. From the actuator dynamics in Equation 4.5, we can compute what rotor torques must be applied to follow a prescribed motion and corresponding torque profile in the SEA,

$$\mu N \tau_r = N^2 I_r \ddot{\theta}_m + \tau_{\text{SEA}}. \tag{4.7}$$

The applied rotor torque must overcome the current SEA torque and also accelerate the motor angle to create the desired motion. The motor acceleration is equal to the sum of the load acceleration $\ddot{\theta}_l$ and the scaled acceleration of the spring torque $\ddot{\Delta}_{\text{spring}} = \frac{\ddot{\tau}_{\text{SEA}}}{k_{\text{SEA}}}$. For a nominal motion (i.e., the sinusoidal center of mass trajectory of a spring-mass system), the trajectory of the load angle can be computed through kinematics. Likewise, we can directly compute the acceleration of the nominal force profile. Thus everything on RHS of Eq 4.7 is known.

Figure 4.5 shows the nominal rotor torque τ_r profile for a moderate gait parameterized by a virtual spring stiffness of 16000 N/m, a virtual spring rest length of 1.09 m, and an apex height of 1.12 m. The required motor torques are dominated by the SEA torque τ_{SEA} that must be opposed; the inertial effects from the dynamics of the actuator contribute less than 8% to the required motor torques. More explicit optimization routines may be applied [79, 182] to identify gaits that minimize the applied torque. The rotor torque limits shown are based on the continuous (100 A) and peak (165 A) current limits of the motor.

In addition to the feedforward torque computed above for a nominal spring-mass behavior, we leave room for feedback control. The gap between the maximum rotor torque and the nominal rotor torque trajectory (Fig. 4.5) describes how much actuation is available for generating corrective GRFs. We heuristically choose to reserve 40% of our theoretical maximum rotor torque for feedback. After accounting for the geometry-dependent mechanical advantage in the five-bar (Equation 4.4), we use numerical techniques to identify a spring-mass gait that nominally consumes 60% of the available rotor torque at zero forward velocity to produce a maximum hopping height of 0.03 m with an apex height of 1.12 m and an appreciable flight phase of 156 ms.

Experimental evaluation of this design

Experimentally, direct measurements of the applied motor current show that the actuators are often saturated during stance control (Fig. 4.6, bottom) even when the net desired spring torques are within nominal bounds (Fig. 4.6, top). Despite our assumption that only 60% of the peak rotor torque is required for producing the nominal hopping motion, the control utilizes much higher torques in practice. This effect is partially caused by the need to oppose off-nominal torques in the springs as well as the extra torques required for the VSEA feedback control. Furthermore, friction and errors in the assumed efficiency model can cause the motors to saturate more frequently than the above analysis would expect (i.e, the green curves in the bottom plots go to torques of higher magnitude, leaving less room for feedback before saturation occurs).

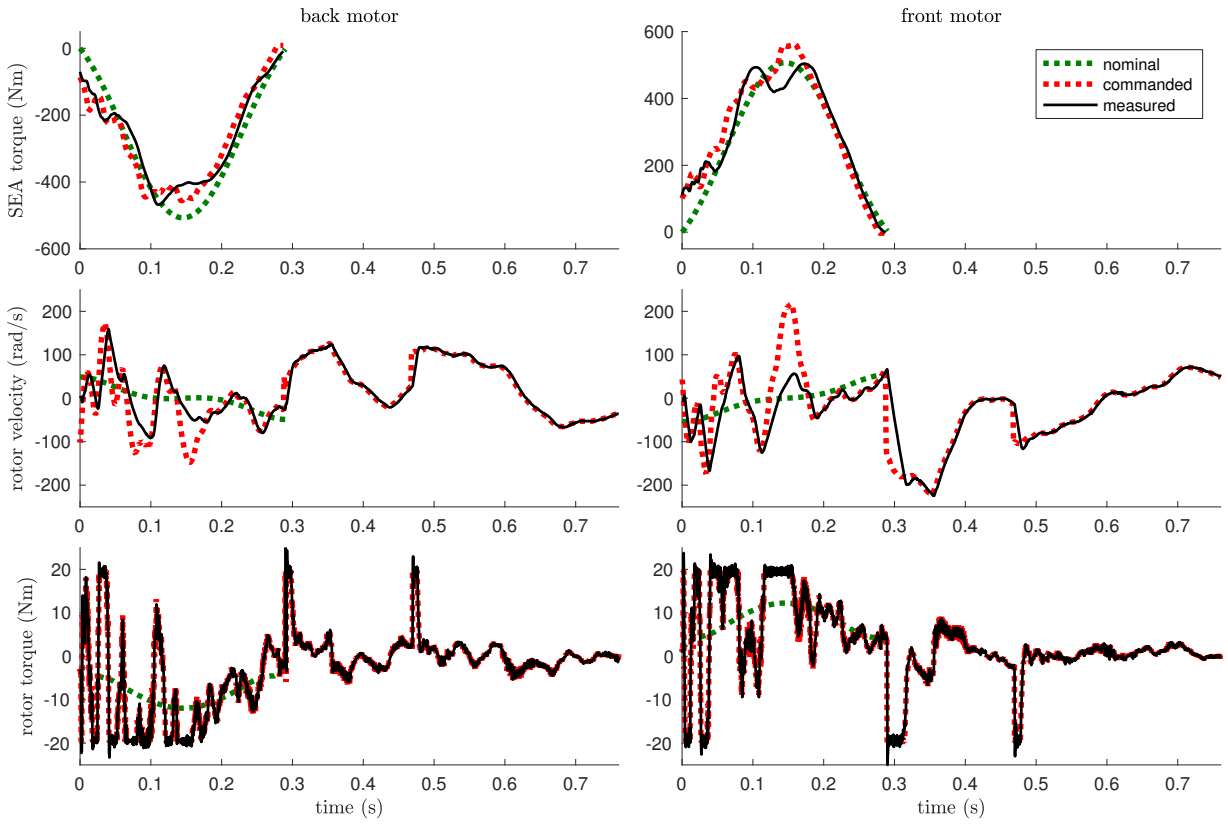


Figure 4.6: Tracking performance for different levels of the actuator control. Left and right columns correspond to the two SEAs in the left leg during a stance-swing cycle. Top row reports SEA torques measured by spring deflections; middle row reports rotor velocities measured by incremental encoders; bottom row reports rotor torques measured by current sensors.

The closed-loop SEA control in Equation 4.6 theoretically creates first-order exponential convergence of the spring torque error, but the saturations in motor torque compromise the tracking of the rotor velocities and invalidate the assumption behind expected convergence. Thus, the errors in spring torque can grow while the motor currents are saturated. In Chapter 7, we analyze the effects of these errors in spring deflection.

4.4.2 Current-induced velocity limits

The servo drive amplifier regulating the motor current has a nominal operating voltage of 50 V, which translates into a maximum swing speed of 7.9 rad/s for the hip angle and a maximum rate of length change of 5.6 m/s for the unloaded leg at a retracted length of 70 cm. These limits hypothetically restrict the leg placement behaviors that are feasible, but on our system these velocity limits are largely irrelevant; the swing leg has sufficient time during the full swing-stance cycle of the opposing leg to be moved into position.

4.4.3 Kinematic limits from hard-stops

The 5 bar mechanism of the leg and the maximum range of travel of the hip joints restrict the relative motion of the leg. We impose tighter virtual limits to avoid harmful impacts on the hardware. This results in a maximum horizontal extension of the leg at touchdown (given a landing height), which in turn restricts the domain of SMM leg placement policies (Chapter 3) that ATRIAS can implement.

Given a landing height z^*_{TD} = 1.09 m (derived in Section 4.4.1) and a vertical trunk orientation, ATRIAS's geometry allows for a leg placement of $|x^*_{TD}| \leq 0.36$ m. For the derived spring stiffness, the limited range on leg placement directly translates into a limited range of speeds $\dot{x}_{i+1}$ that can be reached in the next step (through integrating the dynamics of the simplified model). The range of feasible speeds varies with the initial speed, and this domain and range of deadbeat behaviors is shown in Figure 4.7. In particular, the maximum sustainable speed is 2.6 m/s and

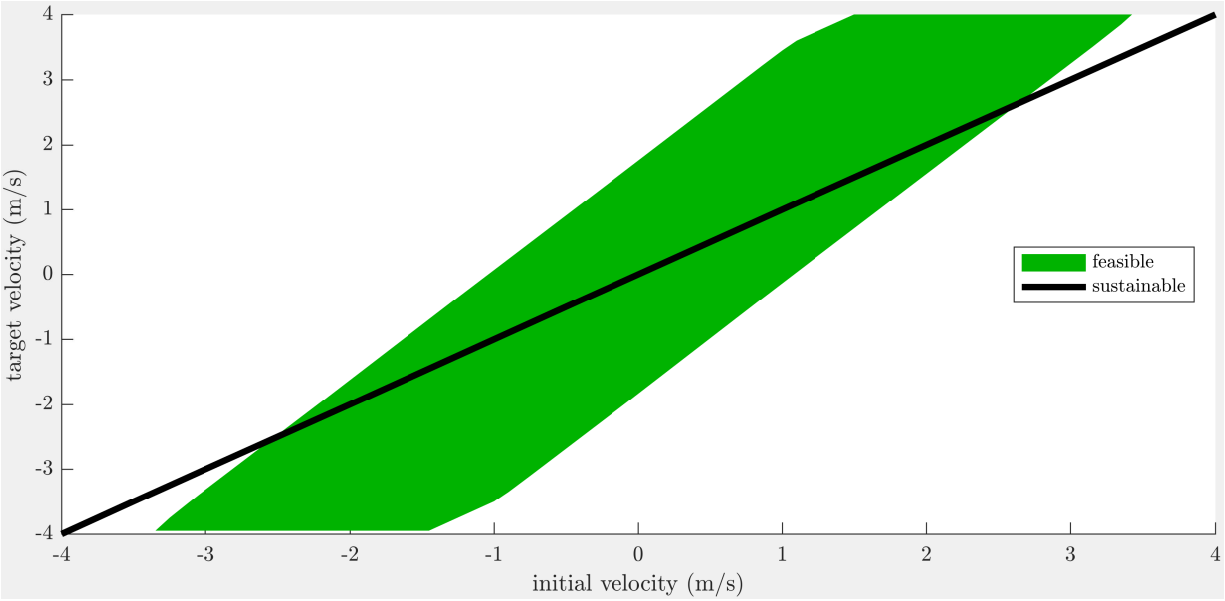


Figure 4.7: Domain and range of deadbeat running. The kinematic limits of the leg restrict the leg placement to $|x^*_{TD}| \leq 0.36$m. Given the spring stiffness derived in Section 4.4.1, this range defines limits on the velocity transitions as a function of initial velocity. The x axis shows $\dot{x}_i$ and the y axis shows the feasible $\dot{x}_{i+1}$ within the kinematic limits. The diagonal line of $\dot{x}_i = \dot{x}_{i+1}$ gives the sustainable gaits; speeds for which this line is not included in the reachable range cannot be maintained.

has no ability to accelerate, and from a hopping in place initial condition, the theoretical control allows for speed changes of up to ± 2 m/s.

In theory, the domain of deadbeat control is also limited by GRFs staying within the friction cone. Here, the geometric maximum leg placement is a tighter constraint (since the GRF acts through the virtual leg), automatically satisfying friction coefficients greater than 0.33. We estimate a friction coefficient of at 0.5 in our setup.

4.5 Summary

We presented the key features of the ATRIAS robot and argued that it is a suitable platform for testing optimal control applied to the centroidal dynamics of the robot. This is facilitated by formulating force control and inverse dynamics while using SEA control to close the loop on

desired joint torques. We use this chapter to set up the associated equations of motion behind these techniques, and we hypothesize that ATRIAS is capable of planar deadbeat spring mass running within a domain defined by its power and kinematic limits. This hypothesis is validated in Chapter 5.

Chapter 5

Transfer of deadbeat spring mass running onto ATRIAS

The results in Chapter 3 show that the spring mass model (SMM) is capable of highly agile and robust running via leg placement, and the analysis in Chapter 4 suggests that ATRIAS is capable of demonstrating these behaviors in the sagittal plane. However, this theoretical performance far exceeds what has been demonstrated in running robots [75, 78, 146, 182]. Like ATRIAS, these robots are clearly more complex systems than the conceptual SMM, and experiments have not sought to reproduce the maximum agility and robustness available to the simplified model. As a result, the utility of these SMM theories for the control of complex running robots had remained largely unclear.

In this chapter, we describe our implementation of the control on ATRIAS for planar spring-mass running with deadbeat velocity tracking and present our experimental results. Much of this material has been reported in [111, 112, 211]. Section 5.1 discusses the relevant state of the art in literature. Section 5.2 first provides an overview of the control architecture before the subsequent sections explain the implementation of each module in greater detail. Section 5.8 presents our experimental results for tracking changing speeds on flat terrain and for running at a constant speed over rough terrain. Finally, Section 5.9 motivates the more detailed theoretical analysis

that we develop in Chapters 6 and 7 to better understand the contributions of leg placement and continuous feedback as well as the effects of perturbations and uncertainties.

5.1 Background

5.1.1 Spring mass theory in robots

While experimental studies on human-scale, self-contained robots have yet to explore the behaviors allowed by spring mass theory, several researchers have investigated the foot placement strategies of the SMM on more simplified hopping robots. For an early example, Zeglin [215] investigated state space planning algorithms based on the SMM for a bow-legged hopper with a compressible spring and passively stabilized trunk. More recently, Shemer and Degani [172] 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 [118] quantified the predictive performance of analytical approximations of the SMM in achieving deadbeat behavior using a monopedal spring leg with no attached trunk. These studies were all 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; here we implement deadbeat SMM strategies on ATRIAS, a bipedal machine of human scale and weight with an actively controlled trunk and without external sensing (Chapter 4).

5.1.2 Technical implementation

Implementing closed loop control on hardware systems is a very well studied problem in robotics and across engineering. At a high level, we propose a specific decomposition of the overall

task of implementing deadbeat SMM running into sub-problems that we then individually tackle with established techniques, and we hypothesize that these choices lead to effective results. In this work, we do not offer significant innovation to these algorithms, but our structured implementation has allowed us to demonstrate the novel result of deadbeat spring-mass running on a full-bodied robot.

In particular, we formulate deadbeat control of the simplified model by building on the body of work reviewed in Section 3.1. We use tools from linear systems and optimal control to stabilize the full order robot. The books [9, 14, 15, 26, 90] provide thorough theoretical derivations and guidelines for practical application. We solve constrained inverse dynamics for a torque controlled system in line with the formulations posed by many other roboticists, as detailed in Section 2.2.2. We control our joint torques by building upon the SEA studies developed in [124, 137, 160, 198, 212, 213] (reviewed in Section 4.2.2) and use standard kinematic [173, 174] algorithms to execute position control of the swing leg. Finally, we apply Kalman filtering [60, 89, 202] throughout our system for extracting state estimates from the sensor data.

5.1.3 Problem formulation and decomposition

Beyond the direct implementation of these algorithms, the challenge of building an effective controller for the full order system lies in how we choose to represent and deconstruct the problem. Here, we took an approach that leverages linear theory and optimal control as much as possible. The context, motivation, and outcomes of this choice of architecture are discussed in detail in Chapter 6.

5.2 Overview of control architecture

This section first gives an overview of the control architecture. See Figure 5.1. The desired running behavior is described by an externally specified velocity $\dot{x}_i^*$ and apex height $z_{a,i}^*$ for

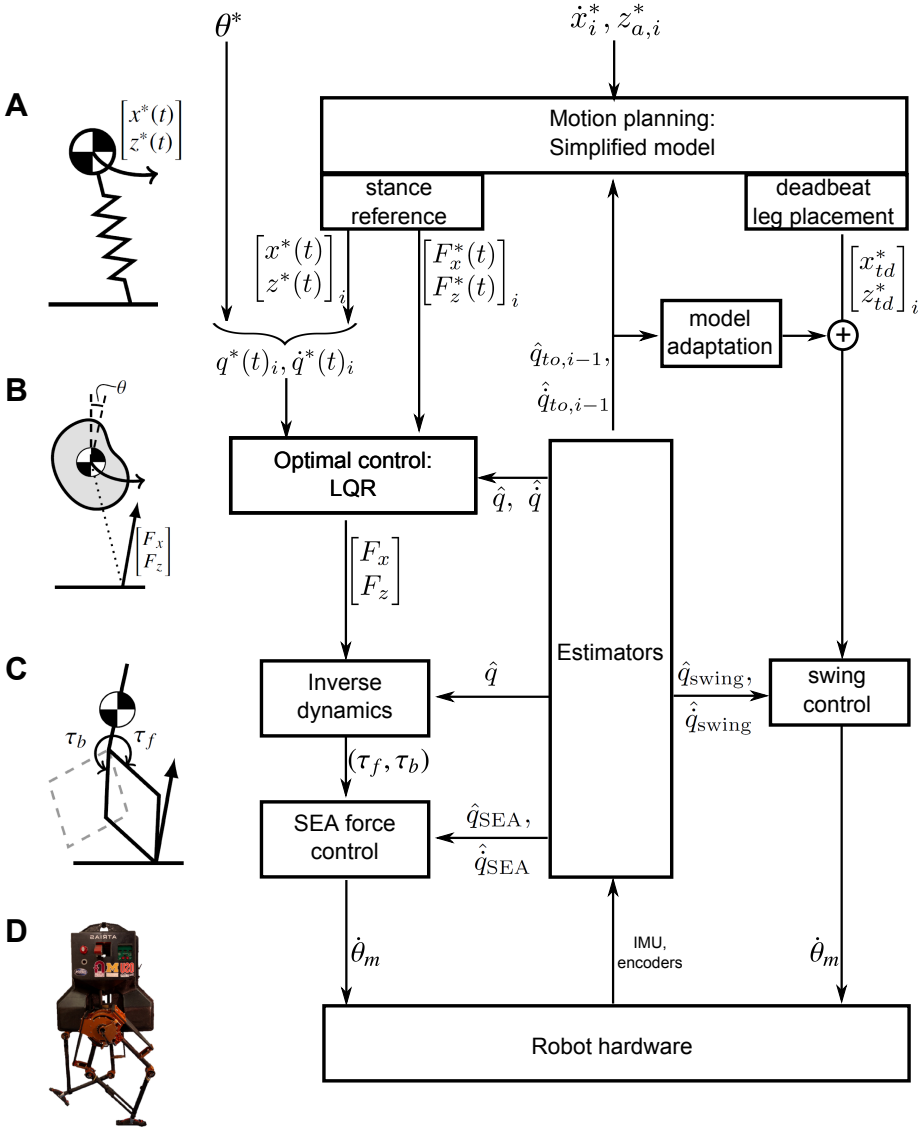


Figure 5.1: Overview of control architecture, see beginning of Section 5.2 for a full explanation. Individual modules can be sorted into layers accordingly to the level of detail of the underlying model. Layer A uses the simplified model (translational dynamics only) to do motion planning. Layer B adds the rotational dynamics to define a full order model. The motion plan is tracked in the full order space using optimal control. Layer C accounts for the leg geometry and the SEAs to generate the desired GRFs and to track the leg placement. Layer D is the actual robot hardware, which takes motor velocity commands and outputs sensor information. The sensor signals are passed into estimation routines, and a model-adaptation block uses the error across multiple steps to adjust the leg placement.

each flight phase (parameters of the planar SMM running gait; see section 2.4.3), along with a stationary pitch θ^* for the trunk.

At the highest level (A), we use the reduced-order simplified model to do motion planning at the end of each stance phase. Given the state of the robot at takeoff (denoted with subscript $_{to}$ and index $i-1$) and assuming the dynamics of the simplified model, we identify the leg placement and COM trajectory in next stance phase (index $_i$) that will most closely match the desired gait parameters in the ensuing flight phase. Specifically, this defines the touchdown (denoted with subscript $_{td}$) condition $[x_{td},\ z_{td}]_i$, the nominal COM trajectory during stance $[x^*(t), z^*(t)]$, and the nominal GRFs $[F_x^*(t),\ F_z^*(t)]_i$ during stance.

At the second level (B), a full order representation of the robot (with states $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$ composed of the orientation θ along with the COM coordinates $[x, z]$) is driven by the moment-free GRFs acting through the point foot of the leg in stance. We apply LQR to solve the optimal control formulation of the trajectory tracking problem. The output of this control block is the desired GRF vector $[F_x, F_z]$, which is a sum of the feedforward force profile $[F_x^*(t), F_z^*(t)]$ and the LQR-derived feedback.

At the third level (C), we take into account the specific properties of ATRIAS's construction to generate these GRFs in stance and to track the desired touchdown condition with the swing leg. Inverse dynamics maps the GRFs into joint torques $[\tau_f, \tau_b]$ for the pair of series elastic actuators at the hip of the stance leg, and we command motor velocities $\dot{\theta}_m$ to close the loop around the corresponding target spring deflections. For a leg not in stance, we plan a trajectory that swings it into position for the upcoming stance phase while maintaining sufficient ground clearance and minimizing the anticipated impact. We then command motor velocities $\dot{\theta}_m$ to track this trajectory with position based control.

The hardware level (D) accepts motor velocity commands and passes back raw sensor data. Each control block in layers B and C needs continuous state information, which is provided by estimation routines (Kalman filters) that act on data from the on-board IMU and joint encoders.

The simplified model updates the motion plan at end of each stance phase and only uses discrete state estimates. Specifically, in this implementation, the simplified model only needs the estimated forward velocity $\hat{\dot{x}}_{i-1}$.

Finally, over the course of multiple steps, we apply an adaptive learner to identify and compensate for systematic errors. This routine augments the simplified model leg placement $[x^*_{td}, z^*_{td}]$ with an adjustment before the desired touchdown condition is passed to the swing control.

The following sections will describe the implementation of each module in greater detail.

5.3 Planning with the simplified model

We use a modified spring-mass model to plan a feasible hopping pattern and a leg placement that deadbeat stabilizes the horizontal speed at each step. The final output of the simplified-model based motion planning layer is three objects. It provides a reference trajectory $[x^*(t), z^*(t)]_i$ (where $t = 0$ starts at touchdown) for the upcoming stance phase to arrives exactly at the desired apex in the next flight phase. The initial condition of this trajectory $[x^*_{TD}, z^*_{TD}]_i$ defines the target leg placement. Lastly, the planner also returns the nominal force profile $[F_x(t)^*, F^*_z(t)]_i$ along this reference trajectory.

5.3.1 Modified spring-mass model

First, we modify the original spring-mass model using the same change made in [117] to describe a "vertical spring." This modification decouples the vertical dynamics to enable independent control of apex height and horizontal speed achieved during flight, and it implicitly defines a non-constant system energy with more natural gait variables for hopping at a fixed height. For gaits with mostly vertical leg angles, the two models are virtually indistinguishable. More importantly, from a theoretical standpoint, this change does not affect the concept of deadbeat stability through leg placement – there is still a 1-to-1 mapping between the discrete control input of leg placement

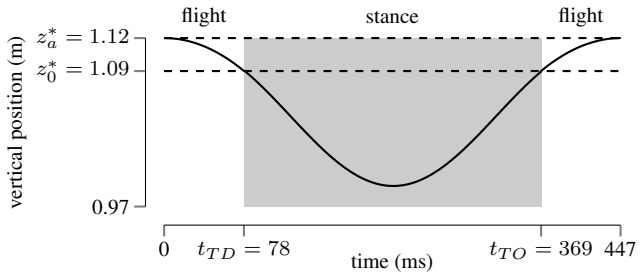


Figure 5.2: Prescribed vertical motion $z^*(t)$ between two apexes. The motion is derived from Eq. 5.2 assuming $m = 64$kg and $k = 16$ kN/m. The motion is used by the ATRIAS controller as a target behavior and re-initiated in every flight phase.

and the discrete output of velocity in the next flight phase. the modified SMM dynamics are given by by

$$m\ddot{x} = k\,(z_0^* - z)\,\frac{x}{z}, \tag{5.1}$$

$$m\ddot{z} = k\,(z_0^* - z) - mg. \tag{5.2}$$

The vertical force is a function of only the vertical states, and the horizontal force is scaled such that the total GRF is still directed along the axis of the leg.

5.3.2 Feasible hopping pattern

Based on ATRIAS's actuation limits, we numerically optimize for a gait that achieves maximum hopping height while reserving 40% of the available actuation for feedback (Section 4.4.1). This yields the vertical reference motion $z^*(t)$ in Figure 5.2 with a 3 cm hopping height an a 156 ms flight phase.

5.3.3 Deadbeat speed control

Given the vertical reference, we compute the corresponding return map (see Chapter 3) of the horizontal motion by integrating the dynamics in Eq. 5.1 from an array of initial conditions

(parameterized by speed $\dot{x}_i$ and touchdown position $x_{TD,i}$):

$$\dot{x}_{i+1} = f_{\text{return map}}(\dot{x}_i, x_{TD,i}, z^*(t)).$$

The deadbeat control law for leg placement in flight to regulate running speed is then

$$x^*_{TD,i} = f^{-1}_{\text{return map}}(\dot{x}_i, \dot{x}^*_a, z^*(t)) \tag{5.3}$$

where $\dot{x}^*_a$ is the desired speed $\dot{x}_{i+1}$ for the next flight phase. The domain and range of this tabulated control policy is limited by ATRIAS's maximum leg extension and is shown in Figure 4.7; the top sustainable speed is 2.6 m/s, and the control can make one-step changes in velocity of $\pm$2m/s from hopping in place.

Just as the original SMM implicitly assumes a fixed energy, this simplified model implicitly assumes exact tracking of the profile $z^*(t)$. Under this assumption, the leg placement in Eq. 5.3 then provides 1-to-1 deadbeat control of the horizontal velocity $\dot{x}_i$ at each step within the domain defined above.

5.4 Stance leg control

We solve finite-horizon optimal control for tracking the reference motion with GRFs as the control input. These GRFs are passed through constrained inverse dynamics to yield joint torques, which are then passed on to the SEA control.

5.4.1 Model-based optimal control

First, the 2 degree-of-freedom reference trajectory $[x^*(t), z^*(t)]_i$ from the simplified model must be converted to a properly defined reference in the 3 degree-of-freedom state space of the full order robot, where the coordinate vector $\boldsymbol{q} = [x, z, \theta]$ is composed of the CoM coordinates and the trunk orientation (Section 4.3). Note that for the model of ATRIAS presented in Section 4.3, the full order dynamics and the centroidal dynamics are equivalent.

Because the simplified model applies no net moment about the center of mass, any full order trajectory with $\ddot{\theta} = 0$ is consistent with simplified motion described by $[x^*(t), z^*(t)]_i$ and $\boldsymbol{v}_i^*(t) = [F_x^*(t), F_z^*(t)]_i$ (we use $\boldsymbol{v}$ here for the GRFs to reserve $\boldsymbol{u}$ as the change in GRFs in Eq. 5.5). For this experiment, we choose a stationary upright torso with $\theta^* = \dot{\theta}^* = 0$.

The equations of motion (Eq. 4.2) derived in Section 4.3 yield the non-linear state-space dynamics of s:

$$\dot{\boldsymbol{s}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{v}) = \begin{bmatrix} \dot{x} \\ \frac{F_x}{m} \\ \dot{\theta} \\ \frac{xF_z - zF_x}{I_t} \\ \dot{z} \\ \frac{F_z}{m} - g \end{bmatrix}, \tag{5.4}$$

where the only non-linearity occurs in the angular acceleration.

The state error $\boldsymbol{\xi} = \boldsymbol{s} - \boldsymbol{s}^*(t)$ is defined with respect to the time-varying reference $\boldsymbol{s}^*(t)$. Linearizing these dynamics about the reference then produces the error dynamics

$$\begin{aligned} \dot{\boldsymbol{\xi}} &= \boldsymbol{f}(\boldsymbol{s}, \boldsymbol{v}) - \dot{\boldsymbol{s}}^*, \\ \dot{\boldsymbol{\xi}} &= \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{F_z^*(t)}{I_t} & 0 & 0 & 0 & -\frac{F_x^*(t)}{I_t} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A(t)} \underbrace{\begin{bmatrix} x - x^* \\ \dot{x} - \dot{x}^* \\ \theta - \theta^* \\ \dot{\theta} - \dot{\theta}^* \\ z - z^* \\ \dot{z} - \dot{z}^* \end{bmatrix}}_{\boldsymbol{\xi}} + \underbrace{\begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ -\frac{z^*(t)}{I_t} & \frac{x^*(t)}{I_t} \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}}_{B(t)} \underbrace{\begin{bmatrix} F_x - F_x^* \\ F_z - F_z^* \end{bmatrix}}_{\boldsymbol{u}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{I_t} \\ 0 \\ 0 \end{bmatrix}}_{\boldsymbol{B_{nl}}} \delta_{nl}(\xi, u), \end{aligned} \tag{5.5}$$

with

$$\delta_{nl}(\xi, u) = (x - x^*)(F_z - F_z^*) - (z - z^*)(F_x - F_x^*).$$

Perturbation term $\boldsymbol{B_{nl}}\delta_{nl}$ captures the non-linearities in the dynamics, and dropping this term results in the approximation of linear, time-varying (LTV) error dynamics:

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A(t)\xi} + \boldsymbol{B(t)u}. \tag{5.6}$$

The relative magnitude of the nonlinearities compared to other sources of error is examined in Chapter 7.

In these LTV error dynamics, there is a one-way decoupling that separates the vertical states $\boldsymbol{s_z} = [z, \dot{z}]$ and input F_z from the remaining states $\boldsymbol{s}_x = [x, \dot{x}, \theta, \dot{\theta}]$ and input F_x. That is, the dynamics of $[z, \dot{z}]$ depend only on these states $\boldsymbol{s_z}$ and on F_z, while the dynamics of the remaining states depend on the full set of states and inputs. If we further approximate the system about the upright condition with $x^* = 0, F_x^* = 0$, the error dynamics become fully decoupled:

$$\dot{\boldsymbol{\xi}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{F_z^*(t)}{I_t} & 0 & 0 & 0 & \cancelto{0}{-\frac{F_x^*(t)}{I_t}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A_{\text{decoupled}}(t)} \begin{bmatrix} x - x^* \\ \dot{x} - \dot{x}^* \\ \theta - \theta^* \\ \dot{\theta} - \dot{\theta}^* \\ z - z^* \\ \dot{z} - \dot{z}^* \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ -\frac{z^*(t)}{I_t} & \cancelto{0}{\frac{x^*(t)}{I_t}} \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}}_{B_{\text{decoupled}}(t)} \begin{bmatrix} F_x - F_x^* \\ F_z - F_z^* \end{bmatrix}. \tag{5.7}$$

In effect, this assumption pushes more modeling error into the term δ_{nl} in Eq. 5.5, but it allows us to derive, implement, and validate simpler control laws in lower-dimensional systems. Throughout this thesis (especially Chapters 6 and 7), we rely on the two-dimensional subsystem $\boldsymbol{s}_z$ to visualize and interpret results, and we rely on the subsystem $\boldsymbol{s}_x$ to study the fundamental challenges of under-actuation and the unstable mode of inverted-pendulum type models. The math we use and the controllers we derive have no issues with going from vectors of length 2 or 4 to vectors of length 6; and the de-coupling assumption can be easily removed for a final implementation.

Over the course of each stance phase, the goal is to keep state error $\boldsymbol{\xi}$ small with feedback control $\boldsymbol{u}$ to track the reference motion $\boldsymbol{s}^*(t)$. We formulate a finite-horizon optimal control problem to minimize the total cost J composed of quadratic state costs $\boldsymbol{H} \geq 0, \boldsymbol{Q} \geq 0$ and

control costs $\boldsymbol{R} > 0$, assuming linear error dynamics:

$$J(\boldsymbol{\xi}, \boldsymbol{u}, t) = \boldsymbol{\xi}^T(t_f)\boldsymbol{H}\boldsymbol{\xi}(t_f) + \int_{t=0}^{t=t_f} \boldsymbol{\xi}^T(t)\boldsymbol{Q}(t)\boldsymbol{\xi}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}(t)\boldsymbol{u}(t)dt,$$

$$J^{\text{opt}}(\boldsymbol{\xi}, t) = \min_u J(\boldsymbol{\xi}, \boldsymbol{u}, t),$$

$$\boldsymbol{u}^{\text{opt}}(\boldsymbol{\xi}, t) = \operatorname{argmin} J(\boldsymbol{\xi}, \boldsymbol{u}, t),$$

subject to

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A}(t)\boldsymbol{\xi} + \boldsymbol{B}(t)\boldsymbol{u}.$$

This is a finite horizon, linear quadratic regulation (LQR) problem for a time-varying system, which is solved by back-integrating the Ricatti equation

$$-\dot{\boldsymbol{P}}(t) = \boldsymbol{Q}(t) - \boldsymbol{P}(t)\boldsymbol{B}(t)\boldsymbol{R}^{-1}(t)\boldsymbol{B}^T(t)\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{A}(t) + \boldsymbol{A}^T(t)\boldsymbol{P}(t)$$

from the terminal condition $\boldsymbol{P}(t = t_f) = \boldsymbol{H}$ to yield the time-varying cost function $J^{\text{opt}}(\boldsymbol{\xi}, t) = \boldsymbol{\xi}^T\boldsymbol{P}\boldsymbol{\xi}$ and the optimal feedback law

$$\begin{aligned} \boldsymbol{K}(t) &= \boldsymbol{R}(t)^{-1}\boldsymbol{B}^T(t)\boldsymbol{P}(t) \\ \boldsymbol{u}^{\text{opt}} &= -\boldsymbol{K}(t)\boldsymbol{\xi}, \\ \boldsymbol{v}^{\text{opt}} &= \boldsymbol{v}^* + \boldsymbol{u} = \boldsymbol{v}^* - \boldsymbol{K}(t)\boldsymbol{\xi}. \end{aligned} \tag{5.8}$$

Applying the decoupling assumption in Eq. 5.7results in two independent optimization problems that yield

$$F_x^{\text{opt}} = F_x^* - \boldsymbol{K_x}(t)(\boldsymbol{s_x} - \boldsymbol{s_x^*}),$$

$$F_z^{\text{opt}} = F_z^* - \boldsymbol{K_z}(t)(\boldsymbol{s_z} - \boldsymbol{s_z^*})$$

where the gain vector $\boldsymbol{K_z}(t)$ has 2 terms and the gain vector $\boldsymbol{K_x}(t)$ has 4 terms.

We experimentally tuned the terminal cost $\boldsymbol{H}$ and the integrated costs $\boldsymbol{Q}$ and $\boldsymbol{R}$ to keep the feedback gains low (to avoid unstable interaction with the higher frequency SEA dynamics) and for the observed tracking of trunk pitch, running velocity, and apex height. Chapter 6 thoroughly analyzes the theory in terms of stability and trade-off between states.

5.4.2 Inverse dynamics

The inverse dynamics layer of the stance control solves for the spring torques that best match the optimal GRFs $[F_x^{\text{opt}}, F_z^{\text{opt}}]$ within the ground contact constraints.

In our implementation, we use a boom with the intention of planarizing the system. Instead, the actual motion travels on a sphere. The difference goes to zero as the boom radius increases, but in our setup, it produces a non-negligible asymmetry in the stance dynamics. The authors in [182] also take note of this discrepancy in their experiments. This phenomenon is equivalent to the wheelbarrow effect, where the outer leg sees a small mechanical advantage while the inner leg sees a small mechanical disadvantage, thereby scaling the net force acting on the system.

Instead of explicitly modeling the boom constraint and the additional forces, we design our control towards the eventual goal of unconstrained locomotion. Therefore, while the robot is constrained to the boom, the commanded GRFs are compensated with the appropriate scaling to recover the desired net force. We implement a disturbance observer that estimates the scaling factor through comparing measured spring deflections and estimated CoM accelerations (see Section 5.6). In [182], the authors scale the virtual compliance of the leg by a factor of 10%. To avoid over-cluttering the notation, we do not define a new variable to represent the scaled GRF commands; this operation is applied directly to $[F_x^{\text{opt}},\ F_z^{\text{opt}}]$.

The applied GRFs must obey the ground contact wrench constraints (Section 2.1.1). The ground can only push with $F_z \geq 0$, and the coefficient of friction μ_f requires $|F_x| \leq \mu_f F_z$ (here we model $\mu_f = 0.5$). Thus, the force commands $[F_x^{\text{opt}}, F_z^{\text{opt}}]$ are saturated with respect to these bounds. Note that the moment-free constraint is implicitly enforced by the representation of the net contact wrench as $[F_x, F_z]$. Next, the 1-to-1 mapping in Eq. 4.3 converts the scaled, saturated GRFs to the virtual force and torque pair $[F_l, \tau_h]$, which is finally converted to a desired pair of spring torques $[\tau_f, \tau_b]$ as a function (Equation 4.4) of the configuration of the five-bar.

The process used here equivalently solves a specific formulation of what the hierarchical quadratic programming (QP) approach to inverse dynamics (reviewed in detail in Section 2.2.2)

more generally solves as a constrained optimization problem. Here, the only objective is producing the LQR-computed GRF, and maintaining ground contact is imposed as a higher priority constraint. We use the structure of our equations of motion to limit the complexity of the "inverse dynamics" steps to a 1-to-1 inversion preceded by saturations. In effect, this leaves the "optimization" to a higher layer of the control architecture that uses model-based prediction (Chapter 6) to reason about the under-actuation of the system. The effects of the saturation on the optimality (and thus stability) of the control are empirically observed to be very small (Figure 7.3 in Chapter 7).

5.4.3 SEA control

To accurately generate the desired GRFs, we apply SEA control to close a feedback loop around the desired spring torques $[\tau_f, \tau_b]$. We implement the control formulated in [160], which uses the measured load velocity to do full-state feedback (Equation 4.6) and create a first order exponentially stable error. The velocity-sourced formulation takes advantage of high on-board bandwidth to reduce tracking sensitivity to friction and efficiency. Despite the structural simplicity of Equation 4.6, the implicit motor dynamics and estimator dynamics can create performance issues. Our empirical data (Figure 4.6 in Section 4.4.1) suggests we that may be able to improve the tuning of this control, and many state of the art techniques apply frequency-space design to address these issues and improve performance [94, 212, 213].

This study focuses on the higher level behavior of the robot, and we treat SEA tracking errors as perturbations to the system (analyzed in detail in Chapter 7). The SEA performance we attained was sufficient for proceeding with our implementation, but improvement to this layer of the control may be a promising avenue of future work.

5.5 Swing leg control

Any leg in swing is controlled via position control. Targets positions for the feet relative to the CoM are mapped through the kinematics of the five-bar to define target hip joint angles θ_l^* (Figure 4.4B). The target joint angles are then tracked with position control that assumes the spring forms a rigid coupling between the motor angle and the load angle:

$$\theta_m^* = k_p(\theta_l^* - \hat{\theta}_m) + \dot{\theta}_l^*.$$

This decision is based upon the very low post-spring inertia of the leg (Section 4.3) and the high stiffness of the spring, which together would require very high bandwidth force control. Meanwhile, position control works reasonably well since these properties closely approximate a system in which the motor output shafts are rigidly connected to the joints. However, as a result of this control design, the positional accuracy of the foot is compromised by the uncontrolled, naturally damped oscillations in the spring. The effects of leg placement error are studied in Chapter 7.

The deadbeat motion plan defines a desired leg placement $[x_{TD}^*, z_{TD}^*]_i$ based on the dynamics of the simplified model and the state of the robot at takeoff. Upon takeoff, we define a kinematic trajectory for the primary leg (i.e, the next one to touch down) that meets a minimum vertical clearance (of 20 cm), arrives at the desired leg placement, and matches the horizontal ground velocity to reduce impact forces. During the stance phase, the swing leg mirrors the stance leg as a heuristic preparation for leg placement, and the secondary leg mirrors the primary leg in flight.

5.6 Estimation

The controller requires state information in the form of CoM positions and velocities, pitch and pitch rate, and actuator states. In addition, it needs to know when each leg is in contact with the ground. The CoM positions are estimated via Kalman filtering, contact detection is handled with

a state machine, and the other values are taken directly from high confidence sensor readings and their filtered derivatives.

5.6.1 Kalman filtering

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 $\Delta t = 1\,\mathrm{ms}$ is the time step and w is Gaussian white process noise with covariance $Q = 25\,\mathrm{N}^2$. This is a third order model so that we can estimate accelerations and compute the scaling that the boom effectively applies (Section 5.4.2) to the net forces acting on the system. 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 4.1 which assumes a static point of contact, yielding the dynamics,

$$\hat{\boldsymbol{F}} = f_{\text{dyn}}(\hat{\boldsymbol{\tau}}, \boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{5.9}$$

The measurement equation of the filter is

$$\begin{bmatrix} \hat{x}^R_t \\ \hat{x}^L_t \\ \hat{\ddot{x}}^A_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \\ \ddot{x}_t \end{bmatrix} + \mathbf{v_t},$$

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 3D model of ATRIAS with three rigid bodies (Figure

4.1c), 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

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

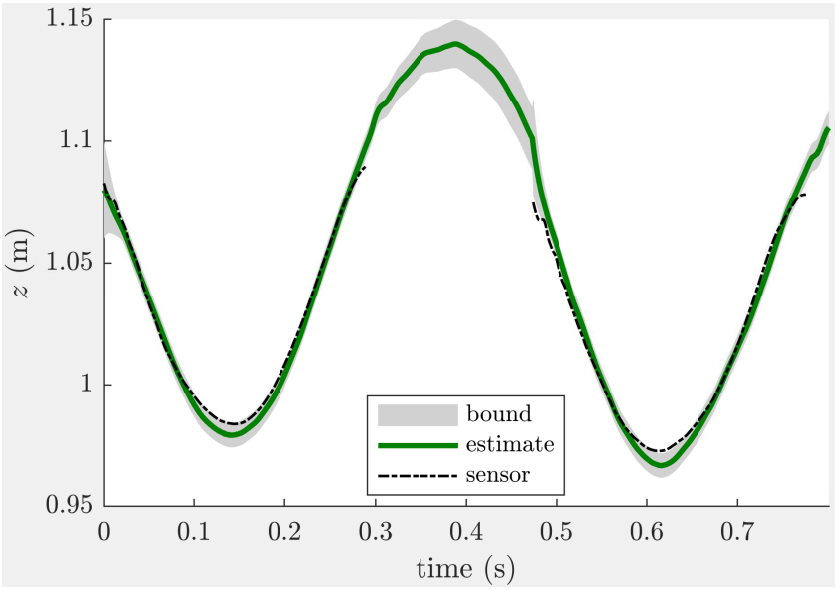
of the measurement noise is adaptive. Specifically, the covariance for the distance measurement noise is inversely proportional to the estimated load on each leg in units of body weight, $\mu_{R/L} = 10\ \hat{F}^z_{R/L}/(mg)$ ($\mu_{R/L}$ clamped to $[0, 1]$, $R_{mn} = 5 \times 10^{-5}\,\mathrm{m}^2$, $R_{mx} = 1\,\mathrm{m}^2$, and $R_A = 4\,\mathrm{m}^2/\mathrm{s}^4$). This continuous scaling inflates the uncertainty on an unloaded leg's CoM measurement by R_{mx} such that this measurement provides negligible influence to the overall estimate of the CoM from the point of contact.

This Kalman filter thus performs sensor fusion, produces a cleaner velocity signal than raw differentiation and is less delayed than low pass filtering. See Figure 5.3.

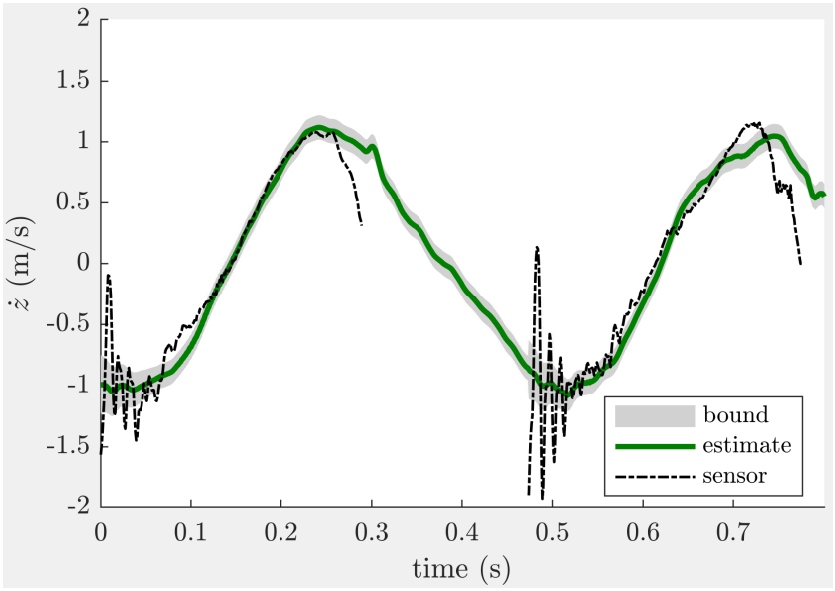
Similarly, we designed a Kalman filter for estimating the load angle in the SEA. Initial experiments suggest this is an effective means of removing the delay introduced by the aggressive filtering required for removing high frequency vibrations in the measured load angle. However, we did not implement this filter for this set of experiments.

5.6.2 Contact detection, stance vs. swing

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 (5.9) 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



(a) Kalman filtering of CoM coordinate z.



(b) Kalman filtering of CoM velocity $\dot{z}$.

Figure 5.3: Kalman filtering of vertical state. The coordinates of the CoM cannot be measured, but they are inferred through measured joint angles and a model of ATRIAS mass distribution. There is no useful kinematic information when the system is in flight. The Kalman filter combines these kinematic "sensor" readings with accelerometer data and spring deflection measurements. The differentiated velocity "measurements" in (b) are particularly noisy, but the Kalman filter outputs a much smoother estimate. There is no available measurement of ground truth for validating these estimates. See Chapter 7 for a discussion on the gray region labeled "bound," which is an empirically-formed assumption about estimation error.

triggers take off and the exit from stance control. This threshold level creates a small delay of approximately $15\,\mathrm{ms}$ (about 5% of stance duration) in contact detection.

5.7 Model adaptation

The final piece of control implementation is the online adaptation of the deadbeat control (Equation 5.3) derived from the simplified model. To counter small systematic modeling errors (like an offset in the estimated CoM location) and repeatable perturbations (like repeatable impact forces from touchdown), the observed error in the return map behavior of ATRIAS is approximated by a linear model

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

where $\dot{\hat{x}}_{i+1}$ is the observed speed in the flight phase $i+1$ and $\dot{x}_a$ is the targetted apex speed. 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 $Y_i = \dot{\hat{x}}_{i+1} - \dot{x}_a$. This error is then anticipated and compensated for by adapting the leg placement $x_{TD,i}$. For small deviations, the return map of the approximate SMM generates an error

$$\dot{x}_{i+1} - \dot{x}_a = \frac{\partial f_{\text{return map}}}{\partial \dot{x}}(\dot{x}_i - \dot{x}_a) + \frac{\partial f_{\text{return map}}}{\partial x_{TD}}(x_{TD,i} - x^*_{TD,i})$$

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,

$$x^{\text{new}}_{TD,i} = x^*_{TD,i} - (\epsilon_1 \dot{x}_a + \epsilon_0 + \frac{\partial f_{\text{return map}}}{\partial \dot{x}}(\dot{x}_i - \dot{x}_a)) \left(\frac{\partial f_{\text{return map}}}{\partial x_{TD}} \right)^{-1}.$$

This algorithm fits an error model based on the parameters of desired change in speed and initial speed, and it was empirically successful in reducing the asymmetries between the right and left legs. Chapter 6 reveals a stronger formulation for a multi-state model (with matrix $\boldsymbol{T_s}$) of the error dynamics based on the closed loop gains. See Section 6.5.3.

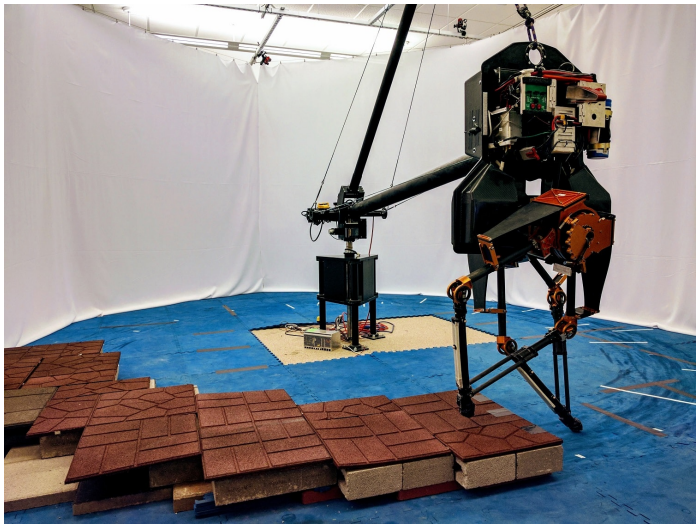


Figure 5.4: CMU's ATRIAS biped shown in the boom testbed during the ground disturbance experiments with unexpected height changes of ± 15 cm (discussed in section 5.8.4).

5.8 Experimental results and simplified analysis

Here, we present our experimental demonstration of deadbeat spring-mass running. In terms of analyzing the results, here we first present a brief overview of the demonstrated behaviors, while Chapters 6 and 7 are largely dedicated to formally studying the expected and the observed performance of the controller. First, we run the system on level ground at a constant speed to establish a baseline for performance. We then test the controller's ability to accurately change running speeds between steps on level ground to assess the transfer of the spring mass model's (SMM) nominal agility. Finally, we experiment with trials of constant speed running on rough terrain to examine the sensitivity of the closed loop system to these disturbances. (Unlike the SMM gaits examined in Chapter 3, the planning strategy (Section 5.3.1) used in this set of experiments is not designed for rough terrain robustness).

All in all, we achieve highly accurate tracking of the running speeds for spring-mass gaits, we successfully execute one-step transitions between gaits of different speeds, and we demonstrate substantial tolerance for step changes in ground height going both up and down.

5.8.1 Undisturbed running

First, we evaluate the performance of the proposed controller in undisturbed running over level ground at a target speed of $1\,\mathrm{m\,s^{-1}}$, which we use as a "best-case" for the subsequent experiment of targeting changing running speeds. The gait converges to the target speed with an average error of 0.05 m/s, the target apex height is tracked with an average error of 2.6 cm, and the pitch of the trunk converges to a lean angle of $\approx 7.6°$. Figure 5.5 shows the evolution of the key variables over one stride in each leg. At the level of the simplified model, the motion differs from the nominal CoM trajectory $[x^*(t),\, z^*(t)]$ in stance with an error (mean and standard deviation) of $4.5 \pm 4.7\,\mathrm{cm}$ in x and $2.6 \pm 2.0\,\mathrm{cm}$ in z and tracks the nominal leg placement (parameterized here by the virtual leg angle at touchdown) with an error of $0.99 \pm 0.80\,°$ (Fig. 5.5a). The tracking of the CoM cannot be considered on its own; the optimal control is simultaneously regulating trunk orientation, which has an error of $7.6 \pm 6.2\,°$ (Fig. 5.5b). This active feedback is applied using deviations in the GRFs from the reference GRFs of the simplified model (difference of $110 \pm 130\,\mathrm{N}$, Fig. 5.5c). At the actuator level, the desired SEA torques are tracked within an error of ($53 \pm 68\,\mathrm{N\,m}$, Fig. 5.5d), which is limited by the observed $20\,\mathrm{Hz}$ closed-loop bandwidth of the SEA control.

5.8.2 Tracking SMM deadbeat velocity targets

In a second series of experiments we test how closely the implemented controller can realize the deadbeat behavior of the theoretical SMM model when the desired apex velocity $\dot{x}_i^*$ 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. 5.6a). In the first set, the change is $0.2\,\mathrm{m\,s^{-1}}$ with a maximum base velocity of $1.0\,\mathrm{m\,s^{-1}}$. In the second set, the change and maximum base velocity are $0.4\,\mathrm{m\,s^{-1}}$ and $1.6\,\mathrm{m\,s^{-1}}$, respectively. Larger step sizes require deadbeat foot targets beyond the mechanical limits of ATRIAS at high velocities (Figure 4.7). These limits impose a maximum possible velocity of $2.6\,\mathrm{m\,s^{-1}}$ for the chosen spring mass gait.

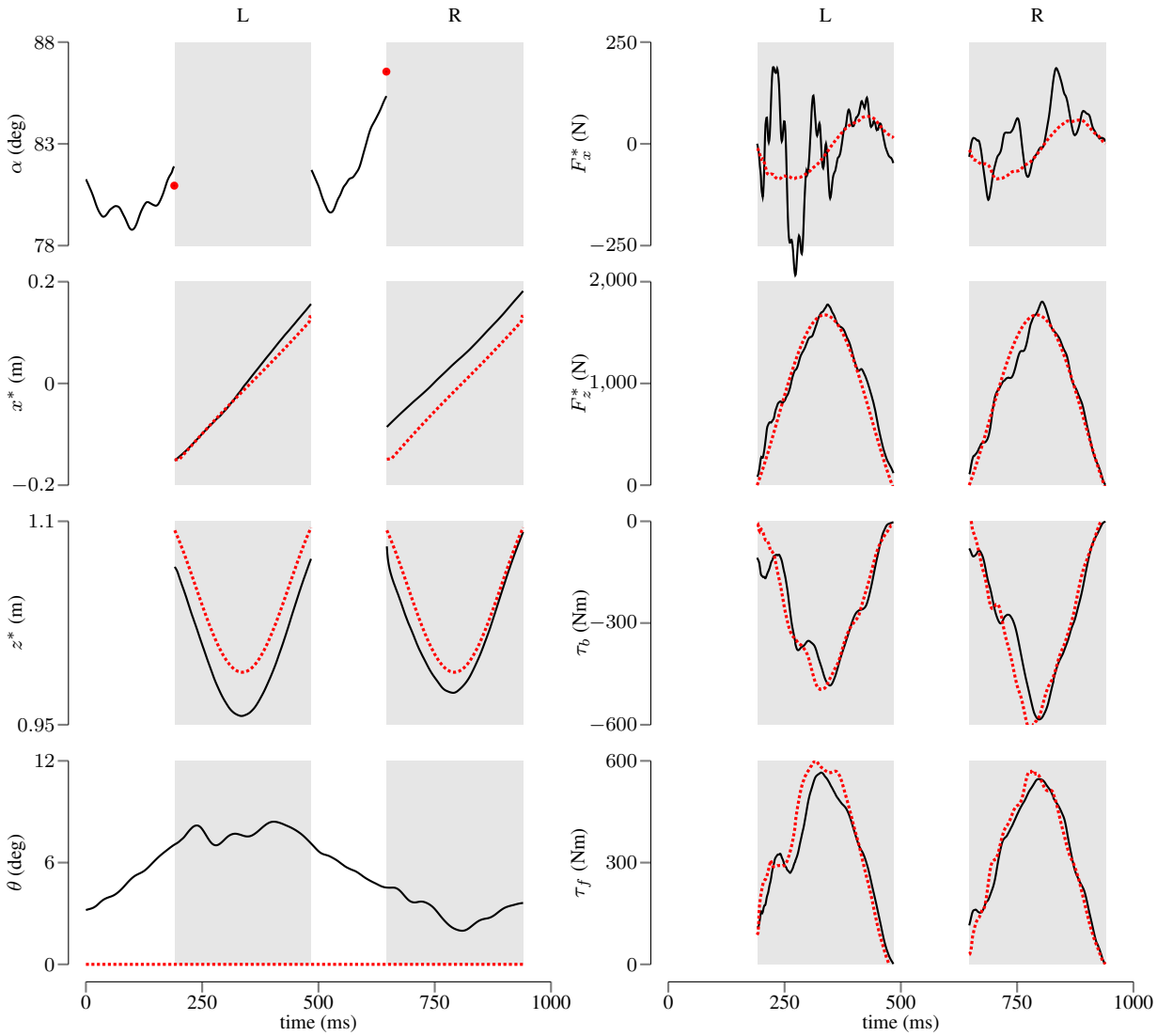


Figure 5.5: Tracking performance of implemented controller for ATRIAS running at $1\,\mathrm{m\,s^{-1}}$ over flat ground without gait disturbances. Shown are the desired (red dashed) and observed trajectories (black solid) of key control variables (Fig. 5.1) for two consecutive steps. The asymmetry between the left (L) and right leg (R) occurs due to the boom constraint.

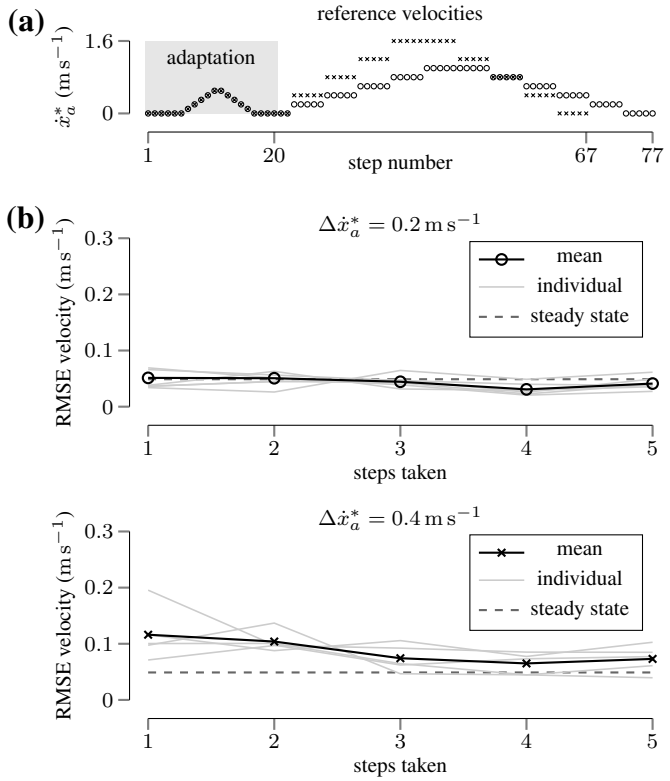


Figure 5.6: Tracking of SMM deadbeat velocity targets. (a) Profile of desired apex velocities $\dot{x}_a^*$ for changes of $0.2\,\mathrm{m\,s^{-1}}$ (circles) and $0.4\,\mathrm{m\,s^{-1}}$ (crosses). The first 20 steps are used in each trial for the online adaptation of the return map (Sec. 5.7) and 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 $1\,\mathrm{m\,s^{-1}}$ (Sec. 5.8.1).

The observed velocity tracking performance is summarized in figure 5.6b. ATRIAS tracks desired apex velocity changes of $0.2\,\mathrm{m\,s^{-1}}$ (circles) with the average error observed in undisturbed running ($0.05\,\mathrm{m\,s^{-1}}$, dashed line; compare Sec. 5.8.1) after one step, indicating deadbeat tracking within the performance expectations defined by the tracking of the undisturbed gait. However, the robot requires more steps for tracking $0.4\,\mathrm{m\,s^{-1}}$ changes (crosses), caused mainly by increased ground impacts at the higher horizontal velocities. We measure a peak horizontal impact force of approximately $200\,\mathrm{N}$ when running at $1.0\,\mathrm{m\,s^{-1}}$. This impact force increases to nearly $300\,\mathrm{N}$ when running at $1.6\,\mathrm{m\,s^{-1}}$.

5.8.3 Simple analysis via simulation

Here we perform a rudimentary numerical study of the expected performance of the transfer of SMM velocity tracking to a noisy, higher order system; Chapters 6 and 7 present more detailed analyses. From a high-level perspective, differences between the real system and the theoretical model can come from unintentional force errors and from intentional changes to the force profile for stabilizing the extra degrees of freedom. We simulate the simplified model (Equations 5.1 and 5.2) subject to the force errors measured on the hardware (i.e, difference between desired and measured spring deflections), and we observe similar velocity tracking errors, with an RMS tracking error of $0.06\,\mathrm{m\,s^{-1}}$ (compare the velocity errors in Figure 5.7) and a peak error of to $0.15\,\mathrm{m\,s^{-1}}$ at $1.6\,\mathrm{m\,s^{-1}}$. The disturbance forces are largest at the beginning of stance due to an initial impact, but the velocity errors grow significantly throughout stance. This tendency is an inherent property of both the hardware and the simplified system (discussed in Chapter 6).

We also simulate the intermediate complexity model (Fig. 5.1b) with an initial orientation error of 10° and observe a velocity error of $0.05\,\mathrm{m\,s^{-1}}$ after one step (compare to the pitch error in Figure 5.5). The simulated system completely recovers after two steps, while the empirical system exhibits steady state error due to ongoing perturbations and modeling error. Nonetheless, this simulated motion reveal consistent scaling between the pitch errors and the translational

errors we see empirically.

These observations suggests that tracking performance could be improved by extending the simplified model to account for ground impacts and for the angular momentum of the robot in running gaits.

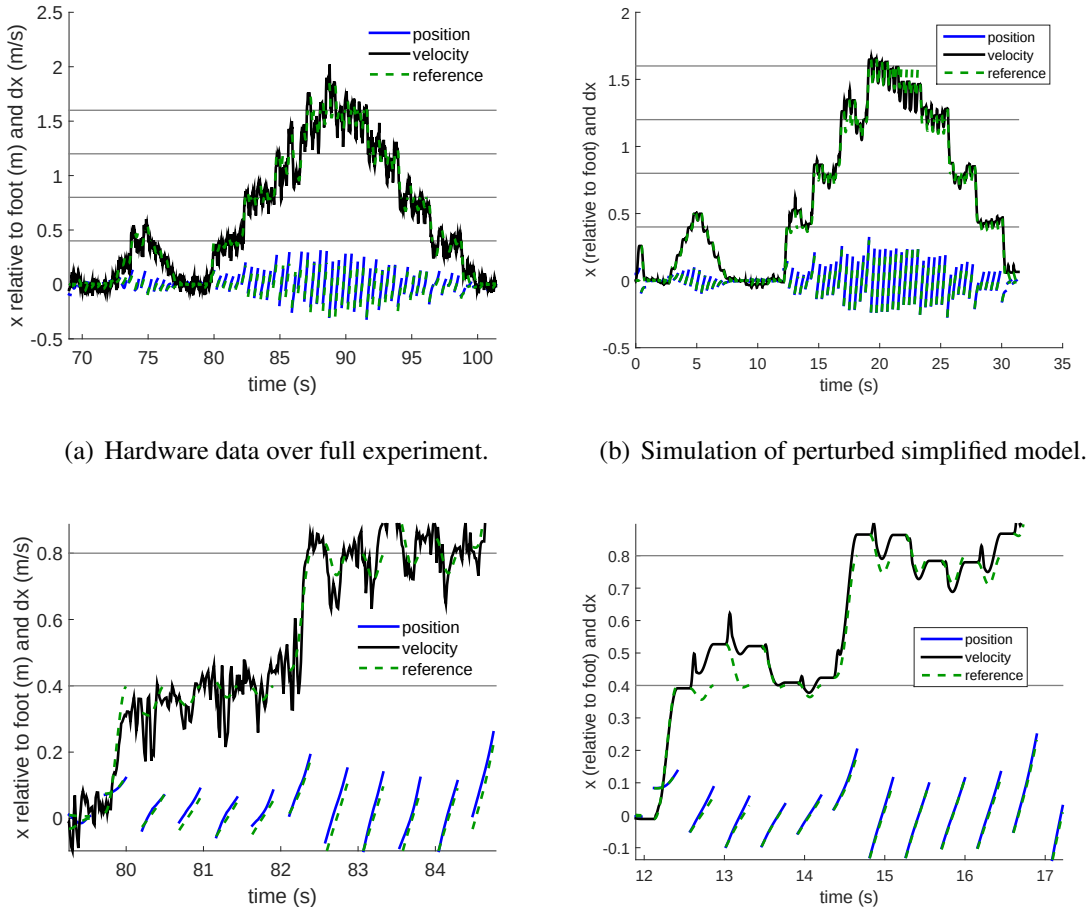
5.8.4 Rough terrain experiments

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 $\pm 6\,\mathrm{cm}$, $\pm 11\,\mathrm{cm}$ and $\pm 15\,\mathrm{cm}$, each repeated for three trials. In all trials, ATRIAS encounters the ground disturbance while running at $1.0\,\mathrm{m\,s^{-1}}$ with its reference gait. (Sec. 5.8.1).

Figure 5.9 shows the velocity tracking performance of ATRIAS after encountering 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 $0.05\,\mathrm{m\,s^{-1}}$ observed in undisturbed running at $1.0\,\mathrm{m\,s^{-1}}$ 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 ($0.2\,\mathrm{m\,s^{-1}}$ to $0.4\,\mathrm{m\,s^{-1}}$), which only gradually diminishes in the next steps.

The velocity error and its gradual decay are largely independent of the direction and size of the ground height change, which seems counterintuitive. For instance, a height drop of $15\,\mathrm{cm}$ results in an increase in speed to $2\,\mathrm{m\,s^{-1}}$ 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.

One potential explanation is that these errors are primarily due to the increased ground impacts and trunk orientation errors, which 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



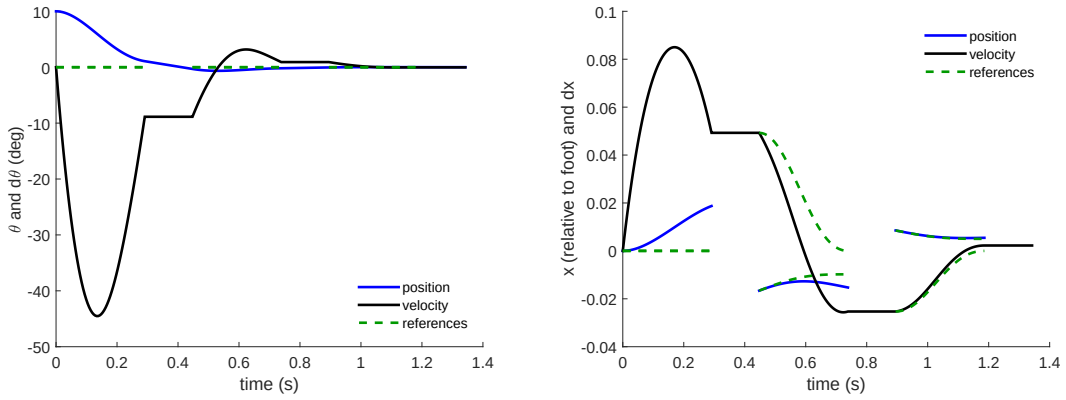
(a) Hardware data over full experiment.

(b) Simulation of perturbed simplified model.

(c) Subset of hardware data.

(d) Subset of simulation.

Figure 5.7: Sensitivity to force errors. Hardware tracking of horizontal position and velocity versus simulation of spring-mass model subject to force disturbances, using force error data measured from the SEAs in the hardware data. CoM velocity plotted in black, and the flat grey lines show incremental target apex velocities (while the dashed green reference velocities change throughout stance). The similar quality of the velocity tracking in simulation shows that our implementation recovers the deadbeat tracking of the simplified model within the sensitivity of the simplified model to the force disturbances we encounter.



(a) Simulated recovery from perturbation to θ

(b) Effects of θ on x and $\dot{x}$

Figure 5.8: Coupling of the pitch and the horizontal states. Simulation from an initial pitch of 10 degrees under ideal control. This shows that a error in pitch results in errors at the velocity level, which partially explains the observed tracking error of our desired velocities. In particular, the scaling of the pitch-to-velocity relationship is consistent with our experimental data. Without perturbations, this error converges in 2 steps.

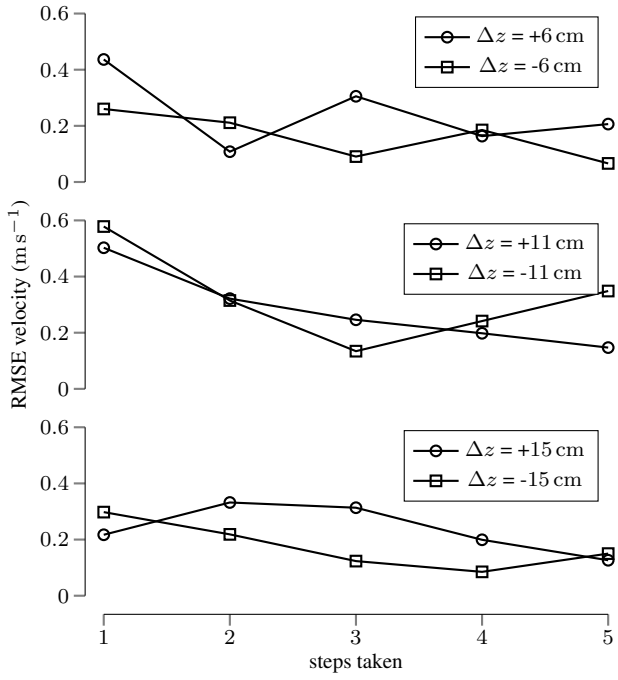


Figure 5.9: Ground height disturbance rejection. Shown are the averages over three trials for the root-mean-square error between the desired apex velocity $x^* = 1.0\,\mathrm{m\,s^{-1}}$ and the velocity achieved by the robot during flight over the number of consecutive steps taken after experiencing an unexpected ground height change Δz.

(Fig. 5.4). This leads to increased impact forces of nearly $500\,\mathrm{N}$ and swing foot impacts with the side of the elevated ground. These disturbances and the subsequent required trunk stabilization may explain the reduced accuracy of the tracking of target speeds on rough terrain. The detrimental effect of swing leg impacts suggests that hardware implementations may be improved through compliant swing leg motions than what the stiff kinematic control (Section 5.5) provides.

5.9 Conclusions and unanswered questions

These results show that despite the numerous sources of error and uncertainty and the additional degrees of freedom on ATRIAS, agile behaviors derived from a simplified model can be transferred to the real, full-order robot. In this study, we limited the implementation to the deadbeat tracking of speeds in a planar running, while the theoretical study in Chapter 3 showed that the spring mass model can further describe deadbeat steering in 3D as well as feedforward robustness to changes in terrain height. The remainder of this thesis focuses on the theory behind how the low order behavior is stabilized in the higher order system, and on how to extend the theory with quantified qualifications of the expected performance on real hardware. The hardware demonstration of the full capabilities predicted for the spring mass model is left for future research.

The results obtained here motivate a number of fundamental questions. The controller consists of two main parts: a discrete component that executes motion planning and leg placement once per flight phase (theory from simplified models can inform the design of this component), and a continuous component that uses force feedback for trajectory tracking throughout each stance phase. It is not immediately clear how much of the agility and stability is attributed to each of these control blocks – does the continuous control have enough authority to stabilize running motions without needing precise motion plans and accurate leg placement from the motion planner? Would it be capable of executing agile changes? On the other side, we have already argued that the lower order model alone does not stabilize the full order system (unless the un-

addressed modes are passively stable); thus it was necessary to augment the spring-mass control with continuous feedback. However, it is still unclear how much the overall stability and tracking relies on the control provided by the reference following, and whether high performance spring mass running can be achieved with minimal bandwidth or power provided to the continuous control. Furthermore, it is unclear how these two sources of control would be best used in conjunction. Chapter 6 addresses these questions with a linear study of the closed loop error dynamics.

In addition, there is a significant gap between the theoretical design and expected behavior and the observed performance. Multiple assumptions (including full state feedback, ideal actuation, and perfect models of the dynamics) are clearly violated in the experimental setup, but these violations should not immediately invalidate all of the theory. Instead of the deadbeat controller promising an exact match of the target velocity and the reference following guaranteeing exponential convergence under unrealistic assumptions, the theory should have a meaningful interpretation in realistic settings. We would like to have a more rigorous answer than simulating a specific initial condition or a specific disturbance to compare with the data. Intuitively, higher performance controllers with faster expected convergence would lead to better tracking of the desired motion, and better tracking from the actuators would lead to less deviation from the nominal behavior. Chapter 7 uses bounded assumptions on the sources of error to extend the theory with quantified bounds on the expected tracking.

Chapter 6

Stability and performance through the combined action of the flight and stance controllers

As motivated in Section 5.9, we want a clearer understanding of how the closed-loop dynamics of the full order system are generated by the combined effects of the discrete, simplified-model based control applied in flight and the continuous reference-tracking control in stance. In this chapter, we study the step-to-step error propagation as the composition two linear mappings. This allows us to solve infinite-horizon optimal control in stance with proper consideration of the error rejection provided by the leg placement, to modify the simplified model's deadbeat law with optimized adjustments, and to develop fundamental insights about the control design of legged systems.

Section 6.1 discusses the relevant literature. Section 6.2 derives the linear closed loop error dynamics, which are then used in Section 6.3 to properly formulate and solve the infinite-horizon optimal control problem for tracking the stabilized motion of the simplified model. Section 6.4 applies these results to analyze and propose improvements to the control of the vertical subsystem, and Section 6.5 similarly presents a set of examples applied to the horizontal-and-rotational

subsystem. Finally, Section 6.6 summarizes the main contributions of this work.

6.1 Background

6.1.1 Poincare analysis

Poincare analysis has been heavily utilized for studying and deriving stability of periodic systems [185]. A Poincare section is defined by describing a discrete event that occurs once per cycle (i.e, apex of the flight phase); this reduces the continuous representation of the system to a discrete model. The Poincare map is the function that describes how the state on the Poincare section changes from cycle to cycle. The local stability of a limit cycle is then given by the Jacobian matrix of this mapping function. If all its eigenvalues have magnitude less than one, nearby states on the Poincare section will converge to the limit cycle, while any eigenvalue with magnitude greater than one implies divergence.

In Section 3.1, we reviewed how Poincare analysis has been applied to examine the stability of various leg placement strategies in spring mass running, given a pre-specified description of the closed loop dynamics. Furthermore, this formulation has also informed the design of actively stabilizing discrete control policies, including the deadbeat 3D leg placement control that we develop in Chapter 3. In general, the method we presented and other constructive approaches describe the return map with parameters that can be treated as discrete control inputs, thereby allowing the direct manipulation of the structure of the closed loop return map. We used tabular lookup to achieve deadbeat stability (with Poincare eigenvalues of 0) for a simplified model; other techniques have used simple feedback [166] or discrete, linear optimal control [31, 182] to create asymptotic stability. In [194] learning algorithms are applied on hardware experiments to identify the control parameters that minimize the magnitudes of the eigenvalues of the return map.

In this chapter we apply similar reasoning to analyze the stability of the combined stance

and flight controllers. While many studies have focused on the local stability of a single limit cycle [32, 136, 182] such that periodicity becomes a requirement in the control design, here we use a slightly more general formulation that examines the error dynamics at discrete events. As a result, the control generalizes to tasks other than converging to a steady-state limit cycle; for example, we can track transitions between different gaits and have access to a library of motions described by the simplified model.

In addition, the factorization presented here offers new insight to the fundamental properties of running gaits and their implications for control. Most existing studies numerically check the eigenvalues of the Poincare map to confirm stability or use it as a tool for automatically generating stable discrete controllers as one of the final steps in the control design. We present the individual matrices as the explicit error dynamics under closed loop control, and thus these structures take on a central role throughout the design process as well as the performance analysis. They clarify the effects of each layer of control, they directly represent the dynamic coupling between different modes of the closed loop system, and they facilitate the analysis the system's response to perturbations (Chapter 7).

6.1.2 Optimal control for stability

Continuous optimal control (see Section 2.3.1) can provide another perspective on stability by working explicitly with the dynamics of the actuated system. Since the problem is formulated as a search for the minimum total accrued state error while applying the least amount of total control to required to bring the system to the desired reference, the infinite-horizon solution is necessarily stabilizing if it exists; otherwise the cost being minimized would be unbounded. For a linearized system, the solution for minimizing integrated quadratic costs (LQR) is described by a linear feedback law paired with an emergent quadratic Lyapunov function (frequently called the "cost-to-go"). The stability guarantee is reflected in non-positive derivative of the Lyapunov function. This formulation is particularly useful for finding efficient, stabilizing solutions on systems with

challenging properties like under-actuation. In the previous discussion on Poincare analysis, the constructive techniques that apply discrete LQR on the transition model similarly have guaranteed stability (assuming controllability of the linear model) – not only do the Poincare states exponentially converge with eigenvalues of magnitude less than one, they do so in a sequence that minimizes the total summed error.

After linearizing the time-varying dynamics about a reference trajectory (in place of a stationary equilibrium point), LQR can be applied to the trajectory-following problem. However, the time-dependent dynamics required for an infinite horizon formulation are often undefined. There are multiple ways the stability guarantee can be recovered. If the dynamics can be properly represented for unbounded time (for example, time-invariant or periodic dynamics), then the Ricatti equation can still be integrated until convergence. Alternatively, a receding-horizon formulation applies a terminal state constraint to the finite-horizon optimization problem to guarantee stability [150], though these methods often still require significant online computation and can produce unbounded gains. The finite horizon problem that we solved in Section 5.4.1 is unconstrained and does not guarantee stability on its own in terms of a continuously decreasing Lyapunov function. Instead, we use Poincare analysis to confirm the stability of combined controllers in terms of decreasing error at discrete events.

Specifically, for periodic running and walking gaits, infinite-horizon linear time-varying LQR (LTV-LQR) has been formulated for continuously stabilizing reference behaviors. [79] uses trajectory optimization techniques to identify an energy-optimal limit cycle for walking on ATRIAS and applies LTV-LQR to track this reference motion. Since the state at the end of the cycle matches the state at the beginning, the backwards integration of the Ricatti equation is propagated from step to step by directly carrying over the cost function. The integration eventually converges to return the infinite horizon cost and associated feedback control law. As a local policy that tracks the fixed limit cycle, this control does not take any advantage of other mechanisms for feedback, including leg placement.

In [32], the authors apply LTV-LQR to stabilize a compass-walking limit cycle on an underactuated robot. The limit cycle itself is simultaneously optimized to reduce the cost passed to the LQR; this defines a much more complex optimization problem that is approached with probabilistic nonlinear techniques. In effect, this formulation then includes leveraging stabilizing effects from parameters of the periodic gait (like leg placement); when applied to the spring mass model [32], this optimization reproduces the deadbeat swing leg trajectory (discussed in Chapter 3) that provides feedforward robustness to terrain changes. However, the periodicity constraint of a limit cycle does not allow for any stabilizing behaviors described by a feedback law.

In [104], the authors formulate LTV-LQR to stabilize a reduced-order model of the center of mass motion on ATLAS. If the desired motion includes a flight phase (during which there are no GRFs and control authority is thus lost), the ballistic motion describes a predictable propagation of the system, which then allows the optimal cost to be explicitly mapped across the discontinuity. The "jump-Ricatti equation" can then be integrated on time scales "long enough to approximate the infinite horizon solution." In this work, the authors do not use any stabilizing feedback in the flight phase.

All these formulations rely on solving how the motion is propagated across multiple cycles such that the Ricatti equation can be backwards integrated to converge to the optimal and guaranteed stable control law. In our work presented here, the closed-loop leg placement of the simplified model directly encodes stabilizing effects in this transition between steps. As a result, we are able to formulate LTV-LQR to design an optimal, continuously-stabilizing feedback control that explicitly accounts for the stability transferred from the deadbeat spring mass model, thereby directly answering one of the central questions of this thesis.

6.1.3 Hybrid zero dynamics for stable embedding of lower order model

Hybrid zero dynamics [28, 135, 136, 182, 206] offers a closely related perspective for designing stable control on the full order system. The primary distinction lies in how the "reference" is

defined. Instead of a trajectory described by a specific state evolution as a function of time, the dynamics of the lower order system define a manifold on which the states evolve. As a surface instead of a trajectory, this manifold incorporates some of the degrees of freedom of the system, and the control that drives the full order dynamics towards the surface may not need to handle the challenges of under-actuation; the design of the virtual constraints is often matched to the available sources of active control. As a result, HZD-based control can be made to have a time-invariant structure [28, 182], which may be beneficial in that the resulting motion is not clock-driven. This difference is examined in more detail in Section 6.6.2.

On its own, the zero dynamics (i.e, the behavior of the lower order system on the constrained manifold) are not guaranteed stable. The energy-efficient running limit cycle optimized in [182] is passively unstable, and it is made stable by an outer loop control that uses leg placement and other discrete parameters are a control input. This feedback law is solved with discrete LQR given the linearized Poincare dynamics of the closed continuous loop system. In effect, this design procedure inverts the trajectory-tracking process we solve here with optimal control: the authors of [182] first design continuous feedback laws to reduce the order of the higher order system, and then stabilize the resulting dynamics using the available discrete inputs; we first model highly stable behavior in the simplified system and then stabilize the full order system with the available continuous control inputs.

One of the central questions we try to answer in this thesis is "how do the stable behaviors of the simplified model transfer to the full order system?" This question is more directly addressed from the optimal control perspective; we minimize the total error and control effort, where the controller must identify the correct compromises to stabilize all the degrees of freedom. The stability imparted by the deadbeat leg placement is explicitly exploited by solving LTV-LQR using the dynamics of the full order system. From the HZD perspective, we would instead answer "can we stably embed the dynamics of our desired behavior within the higher order system?" The stability of the deadbeat leg placement can only be implicitly leveraged by choosing virtual

constraints that omit the speed of the robot, thereby putting this degree of freedom in the zero dynamics and reserving it to be regulated by the deadbeat controller. The final step of solving the discrete LQR would then be redundant with the proposed deadbeat leg placement.

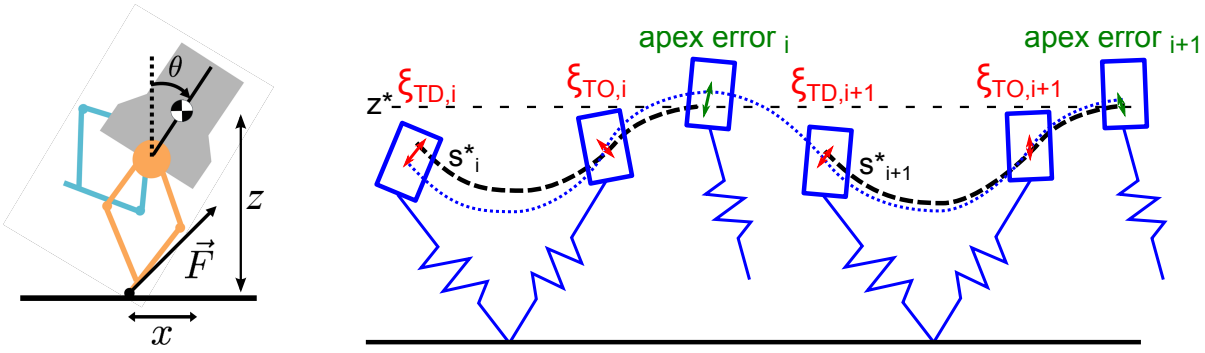
Despite these subtle differences in formulation, the framework we use here and the HZD framework are founded on the same fundamental concepts that describe the closed loop motion of a periodic system, and the main results may be translated between the two perspectives with a little effort.

6.1.4 Summary and contributions

In the current literature, numerical Poincare analysis is the primary method of studying the stability of a closed loop, periodic system. In this work, we factor the return map for running into two components – one that comes from leg placement and motion planning in the flight phase, and one that comes from trajectory following in the stance phase. This decomposition provides clear insights about how control design decisions of legged systems influence gait stability. In particular, using this structure, we can augment the deadbeat leg placement from the simplified model with a full order feedback term that recovers deadbeat stability on the full order system. This generalizes deadbeat spring-mass theory to higher order systems.

Optimal control is a powerful tool that has been used to generate feedback policies to stabilize otherwise unstable limit cycles. Here, we solve LTV-LQR to track reference motions (that need not be periodic limit cycles) of a closed loop simplified model such that the continuous feedback explicitly accounts for the stabilizing properties of the underlying model. This is achieved by modeling the error dynamics of flight phase given the closed loop leg placement.

Hybrid zero dynamics offers a formal framework for stably embedding the dynamics of a lower order model in a higher order system. While this existing theory is highly relevant to our study of transferring deadbeat SMM running to the full order robot, the conceptually similar approach we develop in this work allows us to apply optimal control and linear theory for a



(a) States and inputs

(b) Discrete propagation of error

Figure 6.1: Closed loop evolution of state error. The states of the system are defined in stance and given by $\boldsymbol{s} = [\boldsymbol{q},\ \dot{\boldsymbol{q}}]$, where coordinate vector $\boldsymbol{q} = [z,\ x,\ \theta]$ consists of the CoM coordinates and the trunk orientation. The state error $\boldsymbol{\xi} = \boldsymbol{s} - \boldsymbol{s}^*$ is defined with respect to reference $\boldsymbol{s}^*$. This reference is re-computed in each flight phase (based on the simplified model) and determines the leg placement, which effectively allows a minimization of the error $\boldsymbol{\xi}_{TD}$ by generating a motion plan from the expected touchdown state $\boldsymbol{s}_{TD}$ to the desired apex. In each stance phase, the continuous control acts to reduce state error $\boldsymbol{\xi}$. The error dynamics of the total system can be expressed as a recursive mapping between errors defined at discrete events.

more direct understanding of how the different layers of control interact and ultimately generate stability and agility on the full system. Furthermore, this approach directly extends to stably tracking arbitrary motion plans using the available sources of control, without relying on the periodicity of a nominal limit cycle.

6.2 Linear model of closed loop error dynamics

The results in this chapter and Chapter 7 all stem from framing the control problem in terms of the error dynamics.

6.2.1 Definition of states and errors

We proceed with the definitions used in Section 5.4.1; See Figure 6.1.

The task of agile and robust running on the full order robot can be described as minimizing

the difference between the apex parameters $\boldsymbol{p}_i = [\theta_a,\ \dot{\theta}_a,\ \dot{x}_a,\ z_a]_i$ at each step and the desired values $\boldsymbol{p}^* = [\theta_a^*,\ 0,\ \dot{x}_a^*, z_a^*]$. Poincare return map analysis 6.1.1 yields the local stability of how the error vector $\boldsymbol{p}_i - \boldsymbol{p}^*$ changes from step to step under closed loop control. This analysis is typically performed at the apex event to minimize the number of independent parameters. By construction, the deadbeat reference trajectory $\boldsymbol{s}^*(t)_i$ exactly matches $\boldsymbol{p}^*$ at the apex of the ensuing flight phase. In this chapter, our goal is to study the convergence of the state error $\boldsymbol{\xi} = \boldsymbol{s} - \boldsymbol{s}^*$, which in turn implies the convergence of $\boldsymbol{p}_i$ to $\boldsymbol{p}^*$. To explicitly examine the errors of the gait parameters at the apex, we would derive the nonlinear function $\boldsymbol{p} - \boldsymbol{p}^* = \boldsymbol{f}_{\text{apex}}(\boldsymbol{\xi}_{TO})$ using the equations of motion for a ballistic system in flight (Section 4.3.3). However, since the continuous force control defines the error dynamics $\dot{\xi}$ in the stance phase, it is more convenient to study the evolution of the full state error vector $\boldsymbol{\xi}$ defined in stance.

6.2.2 Error dynamics in stance

The general solution of LTV systems is known ([14]). For any system of the form

$$\dot{\boldsymbol{x}}_g = \boldsymbol{A}_g(t)\boldsymbol{x}_g + \boldsymbol{B}_g(t)\boldsymbol{u}_g(t)$$

(where we use subscript $_g$ to denote a generic variable) with a known initial condition $\boldsymbol{x}_g(t_0)$, the unique solution at any future time $t_1 > t_0$ is given by

$$\boldsymbol{x}_g(t_1) = \boldsymbol{\Phi}(t_1, t_0)\boldsymbol{x}_g(t_0) + \int_{t_0}^{t_1} \boldsymbol{\Phi}(t_1, t)\boldsymbol{B}_g(t)\boldsymbol{u}_g(t)dt, \tag{6.1}$$

where the state transition matrix $\boldsymbol{\Phi}(t_1, t_0)$ is solved by forward integrating the matrix differential equation

$$\frac{\partial}{\partial t_1}\boldsymbol{\Phi}(t_1, t_0) = \boldsymbol{A}_g(t_1)\boldsymbol{\Phi}(t_1, t_0)$$

from the initial condition of $\boldsymbol{\Phi}(t_0, t_0) = \boldsymbol{I}$. (For reference, if $\boldsymbol{A}_g(t)$ is constant, transition matrix $\boldsymbol{\Phi}(t_1, t_0)$ is the matrix exponential $e^{\boldsymbol{A}_g(t_1 - t_0)}$).

In the stance phase, given a time-varying linear feedback law $\boldsymbol{u} = -\boldsymbol{K}(t)\boldsymbol{\xi}$, the linearized error dynamics form the LTV system

$$\dot{\boldsymbol{\xi}} = \underbrace{(\boldsymbol{A}(t) - \boldsymbol{B}(t)\boldsymbol{K}(t))}_{\boldsymbol{A}_{cl}(t)}\boldsymbol{\xi}(t).$$

Thus, the evolution of the error from any time t_0 to later time t_1 in the stance phase is described by the linear mapping

$$\boldsymbol{\xi}(t_1) = \boldsymbol{\Phi}(t_1, t_0)\boldsymbol{\xi}(t_0),$$

where $\boldsymbol{\Phi}(t_1, t_0)$ is integrated using the nominal closed loop dynamics $\boldsymbol{A}_{cl}(t)$. Specifically, in each stance phase the error at takeoff is a linear mapping of the error at touchdown:

$$\boldsymbol{\xi}_{TO,i} = \underbrace{\Phi(t_{TO,i}, t_{TD,i})}_{\boldsymbol{T}_s}\boldsymbol{\xi}_{TD,i}, \tag{6.2}$$

thereby defining stance transition matrix $\boldsymbol{T}_s$ (with subscript $_s$ for stance).

6.2.3 Error dynamics across flight

The state error at the next touchdown $\boldsymbol{\xi}_{TD,i+1}$ is governed by the three things that happen in the flight phase. The states are propagated from the initial condition $\boldsymbol{s}_{TO,i}$, a new reference trajectory $\boldsymbol{s}^*(t)_{i+1}$ is computed, and the swing leg is moved into the desired landing configuration $\boldsymbol{u}_{f,i} = [x^*_{TD,i+1}, z^*_{TD,i+1}]$. We label the leg placement as $\boldsymbol{u}_{f,i}$ (with subscript $_f$ for flight and step index i) since it serves as an input to be assigned by the controller during the i^{th} flight phase. Conceptually, the job of the controller in this phase is to use the simplified model to choose the reference $\boldsymbol{s}^*(t)_{i+1}$ and associated leg placement $u_{f,i}$ that minimizes the anticipated error $\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{s}_{TD,i+1} - \boldsymbol{s}^*_{TD,i+1}$ at touchdown. Here, we will analyze the error dynamics of the flight phase for the control we implemented in Chapter 5.

Defining $t_f = t - t_{TO,i}$ as the time elapsed since takeoff, the motion of the system in flight is

given by

$$\begin{aligned}
\dot{x}(t_f) &= \dot{x}_{TO}, \\
z(t_f) &= z_{TO} + \dot{z}_{TO} t_f - \frac{g}{2} t_f^2, \\
\dot{z}(t_f) &= \dot{z}_{TO} - g t_f, \\
\theta(t_f) &= \theta_{TO} + \dot{\theta}_{TO} t_f, \\
\dot{\theta}(t_f) &= \dot{\theta}_{TO}.
\end{aligned} \tag{6.3}$$

The system transitions to the next stance phase when the foot contacts the ground, and we label the flight time t_f at which this event occurs as $t_{c,i}$ (with subscript $_c$ for contact). Assuming contact occurs after apex (which resolves the two-to-one nature of the t_f^2 term in $z(t_f)$), there is a one-to-one mapping between the parameters $z_{TD,i+1}$ and $t_{c,i}$ given by

$$\begin{aligned}
z_{TD,i+1} &= z(t_{c,i} | z_{TO,i}, \dot{z}_{TO,i}), \\
t_{c,i}(z_{c,i+1} | z_{TO,i}, \dot{z}_{TO,i}) &= \frac{\dot{z}_{TO,i} + \sqrt{\dot{z}_{TO,i}^2 + 2g(z_{TO,i} - z_{c,i+1})}}{g}.
\end{aligned} \tag{6.4}$$

Then, using equations 6.3 along with the reset condition of $x_{TD,i+1} = x^*_{TD,i+1}$, we can solve for the touchdown state as a function of the previous takeoff state and the leg placement:

$$\boldsymbol{s}_{TD,i+1} = f_{\text{flight}}(\boldsymbol{s}_{TO,i}, \boldsymbol{u}_{f,i}).$$

Given the current velocity $\dot{x} \in \boldsymbol{s}_{TO,i}$ and target velocity $\dot{x}^*$, the simplified model planning layer (Section 5.3.1) generates a reference $\boldsymbol{s}^*(t)_{i+1}$ for the next stance phase that exactly produces the necessary change in velocity in one step (while also matching the rest of the apex parameters in $\boldsymbol{p}^*_i$). That is, $\boldsymbol{s}^*_{TD,i+1}$ has $\dot{x}^*_{TD,i+1} = \dot{x}_a$ and $\dot{x}_{TO,i+1} = \dot{x}^*_a$. Unlike $x^*(t)_i$, $z^*(t)_i$ is defined by a fixed limit cycle (Section 5.3.1) independent of the take-off state $\boldsymbol{s}_{TO,i}$; there is no feedback on the vertical states at the planning level. The x and z coordinates of the initial condition $s^*_{TD,i+1}$ of this reference define the target leg placement $\boldsymbol{u}_{f,i} = [x^*_{TD,i+1}, z^*_{TD,i+1}]$, thus describing a tabulated, nonlinear controller $\boldsymbol{u}_{f,i}(\boldsymbol{s}_{TO,i})$.

Finally, this allows us to describe the touchdown error $\boldsymbol{\xi}_{TD,i+1}$ of the upcoming stance phase as a function of the takeoff state $\boldsymbol{s}_{TO,i}$ from the current stance phase:

$$\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{s}_{TD,i+1} - \boldsymbol{s}^*_{TD,i+1} = \boldsymbol{f}_{\text{flight}}(\boldsymbol{s}_{TO,i}, \boldsymbol{u}_{f,i}(\boldsymbol{s}_{TO,i})) \tag{6.5}$$

In particular, the x and z terms of this error vector $\boldsymbol{\xi}_{TD,i+1}$ are 0 by position-based leg placement, and the velocity term $\dot{x}$ is 0 by design of deadbeat control. We can linearize about the nominal takeoff condition this mapping to express

$$\boldsymbol{\xi}_{TD,i+1} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & t_c & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{g}{\dot{z}^*_{TD}} & \frac{\dot{z}^*_{TO}}{\dot{z}^*_{TD}} \end{bmatrix}}_{\boldsymbol{T_f}} \boldsymbol{\xi}_{TO,i} + \Delta_{nl,f} \tag{6.6}$$

where $\boldsymbol{T_f}$ (with subscript $_f$ for flight) is the linear transformation of the state error in flight, and the extra term $\Delta_{nl,f}$ captures any nonlinearities in the original mapping in Eq. 6.5. We show in Chapter 7 that $\Delta_{nl,f}$ is negligible? for the gait we implement.

6.2.4 Closed loop convergence

Having derived linear models for how the state error evolves both in stance and in flight, we can now compose a recursive mapping of the error between steps:

$$\boldsymbol{\xi}_{TO,i+1} = \boldsymbol{T_s}\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{T_s}\boldsymbol{T_f}\boldsymbol{\xi}_{TO,i}.$$

Equivalently, we can also express

$$\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{T_f}\boldsymbol{T_s}\boldsymbol{\xi}_{TD,i}.$$

The product $\boldsymbol{T_s}\boldsymbol{T_f}$ is the linearized return map, or Poincare matrix, of the error at takeoff, while the product $\boldsymbol{T_f}\boldsymbol{T_s}$ is the linearized return map of the error at touchdown. These maps have

the same eigenvalues, which determine the local stability of the closed loop system. Beyond describing the stability of a steady-state limit cycle under closed loop control, these matrices also describe the convergence to arbitrary running patterns generated by the simplified model from Section 5.3.1 under closed loop control.

6.3 Infinite horizon optimal control of stance phase

Having constructed the error transition matrix $\boldsymbol{T_f}$ for the flight phase, we can now properly formulate the infinite-horizon LTV-LQR tracking controller discussed in Section 6.1.2. Instead of a defining the finite-horizon problem for a single stance phase posed in Section 5.4.1 with cost function

$$J_i(\boldsymbol{\xi}, \boldsymbol{u}, t) = \boldsymbol{\xi}_{TO,i}^T \boldsymbol{H} \boldsymbol{\xi}_{TO,i} + \int_{t=t_{TD,i}}^{t=t_{TO,i}} \boldsymbol{\xi}^T(t) \boldsymbol{Q}(t) \boldsymbol{\xi}(t) + \boldsymbol{u}^T(t) \boldsymbol{R}(t) \boldsymbol{u}(t) dt,$$

with integrated state cost $\boldsymbol{Q}(t) \succeq \boldsymbol{0}$, integrated control cost $\boldsymbol{R}(t) \succ \boldsymbol{0}$, and terminal state cost $\boldsymbol{H} \succeq \boldsymbol{0}$, we now define the infinite horizon problem with cost function

$$J_i^{\infty}(\boldsymbol{\xi}, \boldsymbol{u}, t) = \sum_{k=i}^{\infty} J_i(\boldsymbol{\xi}, \boldsymbol{u}, t),$$

where the costs are summed starting from the current step i and propagated indefinitely into the future. Matrix $\boldsymbol{H}$ is no longer a one-time terminal cost, but now serves the role of a once-per-stride penalty on state errors measured at takeoff ($\boldsymbol{H}$ could be made to penalize apex errors w.r.t target apex parameters $\boldsymbol{p}$). The solution to this optimal control problem is found by back-integrating the jump Ricatti equation [110], where

$$-\dot{\boldsymbol{P}}(t) = \boldsymbol{Q}(t) - \boldsymbol{P}(t)\boldsymbol{B}(t)\boldsymbol{R}^{-1}(t)\boldsymbol{B}^T(t)\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{A}(t) + \boldsymbol{A}^T(t)\boldsymbol{P}(t) \tag{6.7}$$

describes the propagation of the cost matrix $\boldsymbol{P}$ within a continuous stance phase, and

$$\boldsymbol{P}_{TO,i-1} = \boldsymbol{T}_{\boldsymbol{f}}^T \boldsymbol{P}_{TD,i} \boldsymbol{T}_{\boldsymbol{f}} + \boldsymbol{H} \tag{6.8}$$

gives the propagation of the cost matrix across flight phases. Intuitively, the "jump" part of this equation describing the discontinuous flight phase propagation comes from substituting $\boldsymbol{\xi}_{TD,i} =$

$\boldsymbol{T_f}\boldsymbol{\xi}_{TO,i-1}$, such that the cost-to-go from the touchdown state

$$\boldsymbol{\xi}_{TD,i}^T \boldsymbol{P}_{TD,i} \boldsymbol{\xi}_{TD,i} = \boldsymbol{\xi}_{TO,i-1}^T (\boldsymbol{T_f}^T \boldsymbol{P}_{TD,i} \boldsymbol{T_f}) \boldsymbol{\xi}_{TO,i-1}$$

can be represented in terms of the previous takeoff. The addition of $\boldsymbol{H}$ in each "jump" comes from accumulation of one-per-stride error upon the $(i-1)^{\text{st}}$ takeoff.

When the backwards integration of this system converges such that $J_i^\infty(\boldsymbol{\xi}, t) \approx J_{i-1}^\infty(\boldsymbol{\xi}, t)$, it means that all future costs are properly represented in cost matrix $\boldsymbol{P}(t)$: given any current error $\boldsymbol{\xi}$ the optimal feedback control will exactly accumulate finite cost $\boldsymbol{\xi}^T \boldsymbol{P}(t) \boldsymbol{\xi}$ while converging to the reference motion. Note that time-varying matrix $\boldsymbol{P}(t)$ is periodic once the back integration of $J^\infty(\boldsymbol{\xi}, t)$ has converged, so the numerical representation of $\boldsymbol{P}(t)$ only needs to span the duration of one stance phase. Finally, the infinite-horizon optimal feedback law is then

$$\boldsymbol{u}^{\text{opt}} = -\boldsymbol{K}(t)\boldsymbol{\xi},$$

where

$$\boldsymbol{K}(t) = \boldsymbol{R}(t)^{-1} \boldsymbol{B}^T(t) \boldsymbol{P}(t).$$

See the blue lines in Figure 6.2(a) for an illustration of the back-integration and convergence. Stability is guaranteed by using the cost function J^∞ as a Lyapunov function, and it can be re-confirmed by examining the eigenvalues of resulting return map $\boldsymbol{T_s}\boldsymbol{T_f}$.

We tune this controller by choosing weights $\boldsymbol{Q}(t)$, $\boldsymbol{R}(t)$, and $\boldsymbol{H}$ to keep the feedback gains in $\boldsymbol{K}(t)$ low enough to avoid coupling with the actuator dynamics and to keep the time-varying gains from changing too quickly. This requires a different set of weights from the ones used in the finite horizon problem solved in Section 5.4.1.

6.3.1 Iterative re-design

This LTV-LQR formulation automatically designs the continuous feedback design as an optimal control problem given the closed loop discrete dynamics. As commonly done in HZD formulations, the discrete control can be automatically designed as an optimal control problem given the

closed loop continuous dynamics. Iteration between these two steps in the style of expectation-maximization (EM) algorithms [38] in the machine learning community then converges to a locally optimal combined control that best exploits the interaction between the two layers of control. Such a process may be an efficient alternative to the highly expensive combined optimization problem formulated in [32].

6.4 Applied examples for vertical system

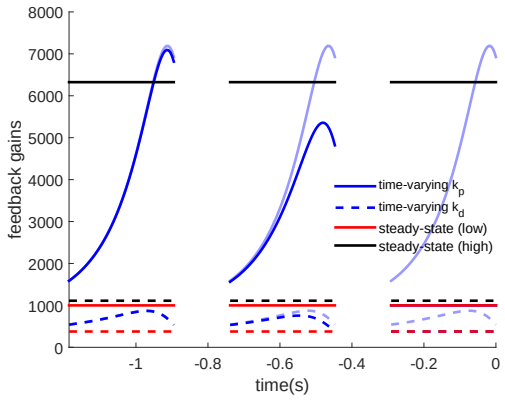
In this section, we use the linear error dynamics derived in Section 6.2 and the infinite-horizon optimal control derived in Section 6.3 to work through a series of examples for the de-coupled vertical system with state $[z, \dot{z}]$ and input F_z.

In Section 6.4.1, we show the stability of the control we applied in the experiments of Chapter 5, we show the instability of a lower-gain stance controller, and we use LTV-LQR to derive a low-gain, guaranteed stable, optimal control with good tracking.

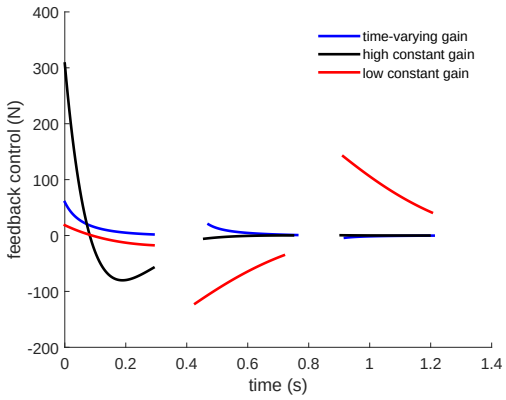
In Section 6.4.2, we use the structure of the linear error dynamics to derive a landing-height law that stabilizes the vertical states without high gain continuous feedback. Existing spring-mass based controllers have not leveraged this parameter as an available control input. We then further demonstrate that iteration between the optimal design of stance and flight control converges to a design that uses both strategies in tandem.

6.4.1 LTV-LQR in stance

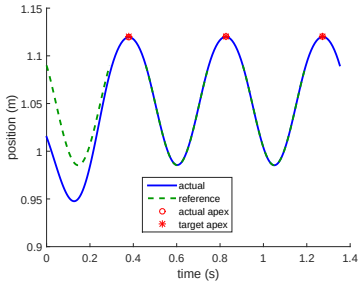
The control we implemented in our experiments in Section 5.8 was originally designed for simplicity. The reference motion is a fixed spring-loaded hop of constant height for each step, and we chose the vertical leg placement to match the nominal landing height of this trajectory. This



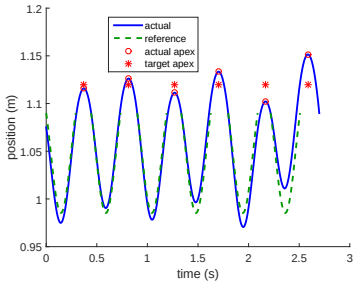
(a) vertical gains



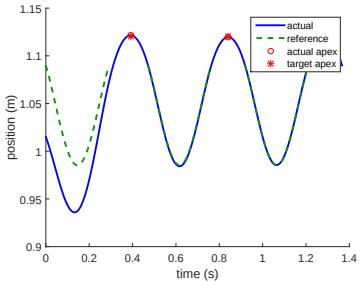
(b) applied control



(c) stable high gain feedback



(d) unstable low gain feedback



(e) time varying feedback

Figure 6.2: LTV-LQR design for vertical system, see discussion in Section 6.4.1. We implemented the black gains by ignoring the effects of T_f, using high enough feedback gains to make T_s sufficiently strong. Higher control cost on R gives red gains when ignoring T_f, which generates an unstable system. Using the high cost R but accounting for T_f, infinite horizon optimal control in blue necessarily stabilizes system. The back-integration of the jump Ricatti equation converges in about 3 cycles.

results in the flight transition matrix

$$\boldsymbol{T_f} = \begin{bmatrix} 0 & 0 \\ -\frac{g}{\dot{z}^*_{TD}} & \frac{\dot{z}^*_{TO}}{\dot{z}^*_{TD}} \end{bmatrix}$$

from Equation 6.6.

Since the stance error dynamics are described by a force-driven second order system, we chose constant PD gains tuned with infinite-horizon LQR (for a system assumed to permanently stay in stance) to be high enough to stably track the reference. Low gain feedback can "stabilize" $\boldsymbol{T_s}$ while leaving the composition $\boldsymbol{T_s T_f}$ unstable (a known property of hybrid dynamical systems [90, 159]). In theory, increasingly high gain feedback drives $\boldsymbol{T_s}$ to $\mathbf{0}$, thereby stabilizing the composition $\boldsymbol{T_s T_f}$. In practice, our feedback gains are limited by the bandwidth of the actuator control, and we observe oscillations in the force tracking as the gains are increased. We tuned the system experimentally to identify the black gains in Figure 6.2(a) ($k_p \approx 6200,\ k_d \approx 1100$), which yields Poincare eigenvalues eig($\boldsymbol{T_s T_f}$) of $[0,\ 0.09]$ and stabilizes the simulated system in Figure 6.2(c).

Applying the same constant LQR design with a 40 times larger quadratic control cost R produces the lower feedback gains in red. This results in unstable Poincare eigenvalues of $[0,\ 1.4]$, and the simulation in Figure 6.2(d) diverges with larger and larger apex errors.

However, applying the LTV-LQR formulation derived in Section 6.3 that explicitly accounts for the flight phase yields time-varying feedback gains that are guaranteed to be stable. Here, we keep the inflated control cost $40R$ and back-integrate the jump Ricatti equation 6.8 until convergence. In Figure 6.2(a), the stance phase farthest to the right (which is integrated first due to the backwards-time nature of the Ricatti equation) starts with the same low-gain feedback as the red system; the lines are over-laid and the dark blue gains are hidden. In the preceding stance phase, the optimal control identifies higher time-varying gains (dark blue) that account for the propagated cost of the flight phase. This process is continued until the time-varying gains no longer change in sequential steps; here, it takes approximately 3 steps for the gains to converge.

The converged gains are shown as a light blue reference in each of the plotted stance phases to illustrate the evolution of these gains.

The key result here is that by explicitly accounting for the flight dynamics $\boldsymbol{T_f}$, infinite-horizon optimal control automatically shapes the initially unstable gains to achieve stability. The Poincare eigenvalues of the time-varying control law are $[0,\ 0.1]$, and the simulation in Figure 6.2(e) demonstrates the convergence. In particular, the tracking accuracy is similar to the high-gain system in black, but significantly lower feedback forces are applied (Figure 6.2(b)) as the gains are on average much lower (Figure 6.2(a)).

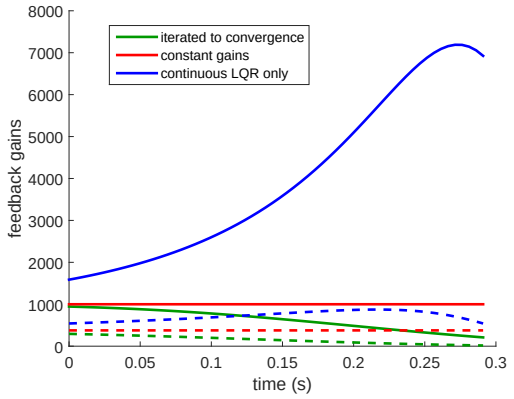
This simulated study thus illustrates the effectiveness of applying the techniques from Section 6.3 to properly formulate optimal control design of the continuous feedback given the dynamics of $\boldsymbol{T_f}$ – stability is guaranteed, and the solution features good tracking and low control costs.

6.4.2 Using the landing height in flight

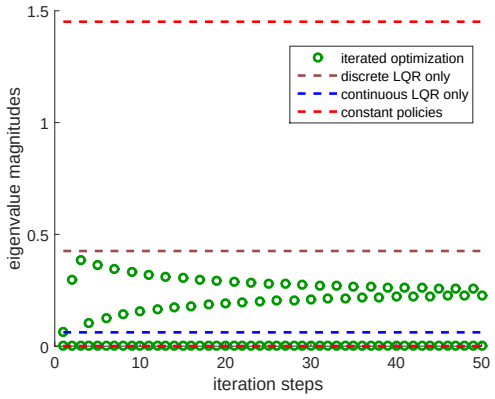
We return to the unstable, low constant gain system in red (Figure 6.2, and Figure 6.3) from the previous study as a motivating example. If the hardware does not have sufficient actuator bandwidth to apply the stabilizing continuous feedback derived by LTV-LQR (blue gains in Figure 6.3(a)), the Poincare matrix $\boldsymbol{T_s T_f}$ may still be stabilized through modifying $\boldsymbol{T_f}$, given the closed loop structure of $\boldsymbol{T_s}$. This is the design procedure typically applied in HZD approaches.

In particular, the vertical leg placement $z^*_{TD,i+1}$ parameterizes the evolution of the system and may be used as a control input in the flight phase. Spring-mass leg placement strategies (see Section 2.4.3) have used a single parameter to represent touchdown (classically leg angle α, which is analogous to $x^*_{TD,i+1}$ here) and maintained constant leg length l (analogous to $z^*_{TD,i+1}$). Our implementation in Chapter 5 followed this convention. However, from the flight error propagation model based on Equations 6.3 and 6.4, we can derive control that leverages this previously unused input.

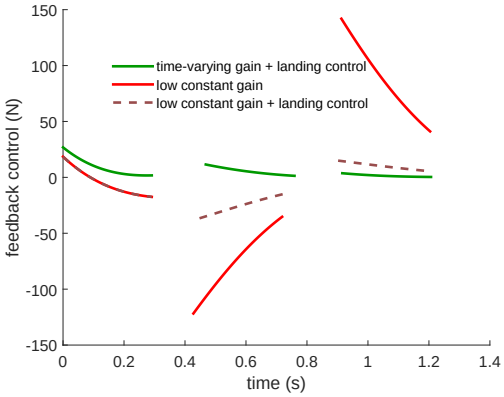
We linearize the flight error propagation around the original condition of a fixed landing



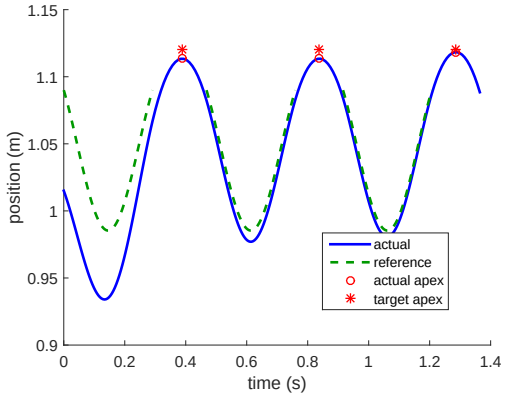
(a) Feedback gains in stance



(b) Eigenvalues of $\boldsymbol{T_s T_f}$.



(c) Applied feedback control.



(d) Simulation of converged iterated control design.

Figure 6.3: Landing height adjustment in flight, see discussion in Section 6.4.2. Red system uses constant gains and no leg placement; it is unstable. Blue system solves LTV-LQR, guarantees stability, requires higher gains. Brown system keeps the red gains, but uses landing height adjustment. Green system is designed by iterating between LTV-LQR design and landing-height design. Simulated trajectory is of this iteratively optimized controller.

height $z^*_{TD,i+1} = z^*_0$ and model the changes in touchdown error $\boldsymbol{\xi}_{z,TD,i+1}$ in response to a change δz_{TD} in the landing height:

$$\boldsymbol{\xi}_{z,TD,i+1} = \begin{bmatrix} 0 & 0 \\ \frac{g}{\dot{z}^*_{TD}} & \frac{\dot{z}^*_{TO}}{\dot{z}^*_{TD}} \end{bmatrix} \boldsymbol{\xi}_{z,TO,i} + \begin{bmatrix} 1 \\ -\frac{g}{\dot{z}^*_{TD}} \end{bmatrix} \delta z_{TD},$$

$$\boldsymbol{\xi}_{z,TO,i+1} = \boldsymbol{T_s} \begin{bmatrix} 0 & 0 \\ \frac{g}{\dot{z}^*_{TD}} & \frac{\dot{z}^*_{TO}}{\dot{z}^*_{TD}} \end{bmatrix} \boldsymbol{\xi}_{z,TO,i} + \underbrace{\boldsymbol{T_s} \begin{bmatrix} 1 \\ -\frac{g}{\dot{z}^*_{TD}} \end{bmatrix}}_{\boldsymbol{B}_{zTD}} \delta z_{TD}.$$

This equation describes a discrete linear system with state $\boldsymbol{\xi}_{z,TO}$ and input δz_{TD}, and discrete LQR returns a feedback policy of the form $\delta z_{TD} = -\boldsymbol{K}_{zTD}\boldsymbol{\xi}_{z,TO,i}$, thereby generating the new Poincare matrix $\boldsymbol{T}_{\text{total}} = \boldsymbol{T_s}(\boldsymbol{T_f} - \boldsymbol{B}_{zTD}\boldsymbol{K}_{zTD})$ under closed loop landing height adjustment.

Without re-tuning the low, constant stance gains in red (Figure 6.3(a)), this discrete control (in brown) stabilizes the Poincare matrix to eigenvalues of $[0,\ 0.45]$ (Figure 6.3(b)) while applying low forces for feedback correction (Figure 6.3(c)).

Finally, as proposed in Section 6.3.1, we can iterate between the optimal design of the discrete control and the optimal design of the continuous control. That is, after solving the LTV-LQR problem yielding a new $\boldsymbol{T_s}$ for the closed loop stance dynamics, we can solve the DLQR problem for leg height adjustment, thereby adjusting the flight dynamics with $\boldsymbol{T_f} - \boldsymbol{B}_{zTD}\boldsymbol{K}_{zTD}$. In Figure 6.3(b), we repeat this process 25 times to converge to a control (green lines) that uses both landing height adjustment and optimized time-varying gains. The result is a controller that uses much lower gains in the stance phase. The off-nominal landing height used by this controller can be seen in Figure 6.3(d), where the blue trajectory and the green reference are not perfectly matched on touchdown.

To take advantage of the available actuator bandwidth on ATRIAS – for which the original constant gains in black (Figure 6.2) were feasible – we run the iterative optimization with the original control cost R without the inflation introduced at the beginning of Section 6.4.1. The resulting gains are shaped similarly to the iterated optimal control Figure 6.3(a), but at the scale

of the original gains we implemented. In practice, higher gains improve the convergence (i.e, eigenvalues closer to 0) of $\boldsymbol{T_s}$, which improves the disturbance rejection studied in Chapter 7.

In this simulation study, we applied discrete LQR to design a feedback law that adjusts the landing height to improve the tracking of the vertical states. This is analogous to adjusting the length of the leg in flight for the spring-mass model, which is a previously un-exploited source of control. We showed that the leg placement design is compatible with the LTV-LQR for the continuous control in stance and allows for drastic reduction of feedback gains.

6.5 Applied examples for the x-and-pitch system

Section 6.4 examined control for stabilizing the vertical states $[z,\ \dot{z}]$; this section presents studies on the control for the system with the remaining states $[x,\ \dot{x},\ \theta,\ \dot{\theta}]$ controlled by input F_x. Intuitively, this system is where the under-actuation is prominently visible – both the pitch θ and the horizontal position x are driven by a single force input in stance. Raibert control and other heuristic methods [7, 13, 31, 52, 53, 132, 200] rely on introducing additional input via leg placement between stance phases to reduce the responsibilities of the stance feedback, HZD methods optimizes the use of the discrete input and formalizes the local stability of these strategies, while pure trajectory-tracking approaches locally solve the under-actuated stabilization problem.

Our implementation in Chapter 5 tested and confirmed the hypothesis that (simplified) model-based design of the leg placement can generate highly precise tracking of desired velocities. Here we use the tools developed earlier in this chapter to clarify the implications of this strategy on the stability of the full order system. In particular,

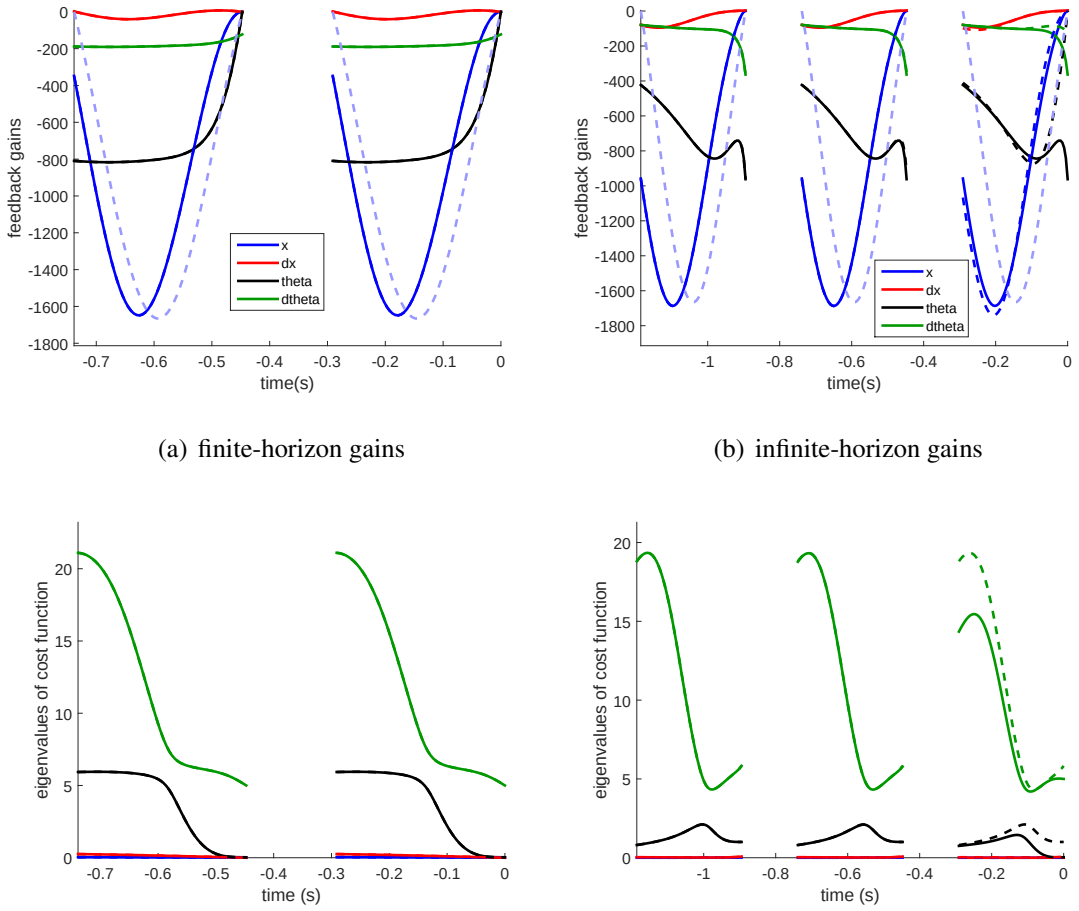
- Properly tuned finite horizon LQR is stable, but we can re-derive this behavior more formally using infinite horizon optimal control.
- Adding the closed-loop stability of the simplified model to the infinite horizon LQR significantly improves its feasibility and its performance.

- $\boldsymbol{T_s}$ gives the coupling between states and allows for modification to the leg placement policy.

6.5.1 Finite horizon versus infinite horizon LQR

First, we show that LTV-LQR around the deadbeat reference model yields similar behavior to what we tuned by hand using finite horizon LQR. See Figure 6.4. The feedback gains $\boldsymbol{K}_x$ are plotted in the top row, while the eigenvalues of $\boldsymbol{P}$ are plotted in the bottom row. The finite-horizon solution on the left does not propagate any information between steps; it resets to a pre-defined terminal cost at each touchdown state and integrates the Ricatti equation for the known time-varying dynamics. Experimentally, we found that choosing a terminal cost that emphasizes orientation error lead to stable running, matching the principles governing heuristically design controllers. The dashed light blue lines in Figure 6.4(a) and 6.4(b) correspond to the x feedback that would re-direct the GRF (given an assumed F_z^* and z^*) through the center of mass, thereby imparting no net moment to the system. This can be viewed as a neutral reference to help understand the behavior of the control – the fact that it is always negative reveals the natural instability of x in an inverted-pendulum system: an axial GRF amplifies any positional error as the system falls over. The black and green lines correspond to corrective PD gains on the orientation.

In Figures 6.4(b) and 6.4(d), we solve the infinite horizon LTV-LQR by integrating the jump-Ricatti equation until convergence. The transient solution during this integration (from right to left for decreasing time) is plotted in dashed lines, and the converged solution is over-laid in solid lines; the optimization converges in roughly 2 steps.



(a) finite-horizon gains

(b) infinite-horizon gains

(c) finite-horizon costs

(d) infinite-horizon costs

Figure 6.4: (A)Finite-horizon gains were tuned empirically on hardware. The dashed blue line is $-\frac{F_z^*(t)}{z^*}$, which is the effective gain on x required such that F_x is directed through the center of mass. Intuitively, it can be thought of as the baseline compensation for creating no pitching moment. The fact that it is always negative (as is the optimal x gain in blue) shows the unstable closed loop tendencies of the horizontal behavior – holding the trunk upright implies unstable feedback of x. This is a fundamental property of the inverted pendulum model. (B) The infinite horizon problem propagates costs across the flight phase and computes gains accordingly.

The closed loop error propagation in stance $\boldsymbol{T_s}$ of the two controllers are nearly the same:

$$\boldsymbol{T_s}_{\text{,finite-horizon}} = \begin{bmatrix} 1.68 & 0.37 & 0.09 & 0.03 \\ 4.82 & 1.65 & 0.25 & 0.15 \\ 0.16 & 0.03 & 0.29 & 0.01 \\ 2.41 & 0.49 & -0.89 & 0.00 \end{bmatrix}, \quad \text{eig}(\boldsymbol{T_s}_{\text{,finite-horizon}}) = \begin{bmatrix} 3.06 \\ 0.34 \\ 0.23 \\ 0.00 \end{bmatrix}$$

$$\boldsymbol{T_s}_{\text{,infinite-horizon}} = \begin{bmatrix} 1.70 & 0.38 & 0.11 & 0.03 \\ 4.88 & 1.66 & 0.28 & 0.15 \\ 0.06 & -0.01 & 0.11 & 0.02 \\ 2.62 & 0.54 & -0.89 & -0.07 \end{bmatrix}, \quad \text{eig}(\boldsymbol{T_s}_{\text{,infinite-horizon}}) = \begin{bmatrix} 3.10 \\ 0.33 \\ -0.01 + 0.08i \\ -0.01 - 0.08i \end{bmatrix}. \tag{6.9}$$

They both let the translational errors grow exponentially during stance, relying on the flight control with

$$\boldsymbol{T_f} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.16 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

to address these modes. Their total closed loop dynamics are given by the Poincare matrices

$$\boldsymbol{T_s}_{\text{,finite-horizon}}\,\boldsymbol{T_f} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.54 & 0.11 & 0.15 & 0.015 \\ 2.41 & 0.49 & -0.99 & 0.00 \end{bmatrix}, \quad \text{eig}(\boldsymbol{T_s}_{\text{,finite-horizon}}\,\boldsymbol{T_f}) = \begin{bmatrix} 0.08 + 0.09i \\ 0.08 - 0.08i \\ 0 \\ 0 \end{bmatrix},$$

$$\boldsymbol{T_s}_{\text{,infinite-horizon}}\,\boldsymbol{T_f} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.46 & 0.08 & -0.03 & 0.01 \\ 2.62 & 0.53 & -0.89 & 0.07 \end{bmatrix}, \quad \text{eig}(\boldsymbol{T_s}_{\text{,infinite-horizon}}\,\boldsymbol{T_f}) = \begin{bmatrix} -0.05 + 0.07i \\ -0.05 - 0.07i \\ 0 \\ 0 \end{bmatrix}.$$

Solved as an explicit optimization problem with a model of the flight dynamics, the infinite-horizon problem achieves slightly faster convergence with slightly lower gains. More importantly, on a theoretical level, this approach is guaranteed stable and returns a continuous Lyapunov function that is always decreasing.

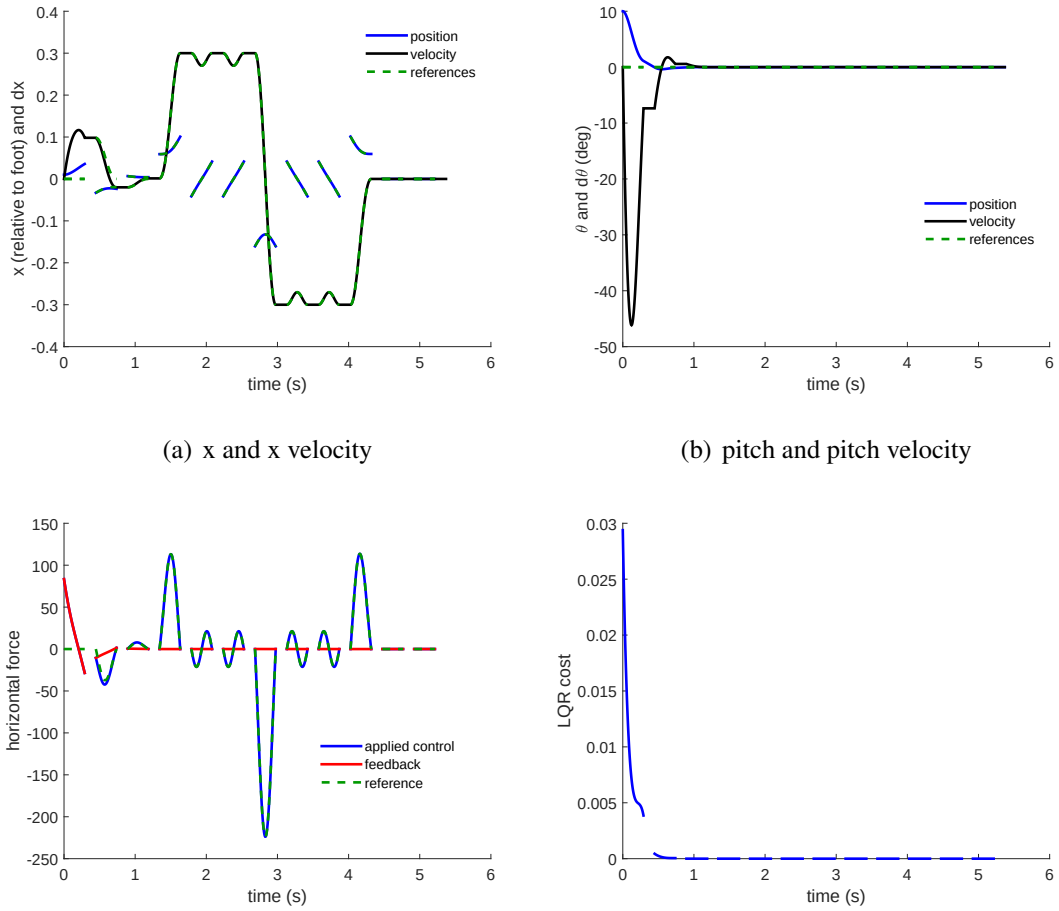
We simulate the system to track a desired velocity profile that takes 3 steps at each at 0, +0.3, -0.3, and 0 m/s. The total behavior of the controllers presented are extremely similar, and here we only plot the infinite-horizon simulation in Figure 6.5. We start from an initial condition with a 10 degree error the trunk and a 1 cm offset in x. The orientation is corrected in the stance phases of the first two steps, and the initial error in x leads to a growth in the velocity error

during the first step. The deadbeat simplified model generates a reference that corrects this error, and the total state tracking is nearly perfect after the second step. The cost-to-go is continuously decreasing as it converges to 0.

6.5.2 LTV-LQR without leg placement

Next, we consider stabilizing fixed limit cycles without using leg placement to track the velocity. In this case, the 2nd entry on the diagonal on $\boldsymbol{T_f}$ becomes 1 instead of 0, since any error in speed at takeoff gets directly carried over to the next touchdown. Designing a stabilizing stance controller by hand would be difficult, but LTV-LQR (keeping the same costs Q and R from the deadbeat LTV-LQR design) eventually converges to a theoretical solution which uses very high gain feedback on x synchronized with quickly changing gains in the other states(Figure 6.6(a)). These high gains match the fact that the cost function (Figure 6.6(b) is two orders of magnitude larger than deadbeat stabilized cost (Figure 6.4(d)). If we were to attempt to balance the robot on point feet (the Acrobot stabilization problem), we would solve the infinite horizon LQR problem for a constant height and no flight transitions. This produces the straight-line gains in Figure 6.6(a).

This theoretically stabilized system is not feasible on ATRIAS; the high gain feedback far exceeds the observed bandwidth of our SEA control. In simulation, it demonstrates very limited tracking capabilities of reference velocities. In Figure 6.6(c), we ask for 5 steps each at target speeds of 0, 0.05, and 0 m/s. The initial perturbation of 1 cm and 10 degrees leads to substantial velocity and orientation errors that are gradually reduced over the course of 5 steps; the tracking performance of even a very small change in velocity is poor. Figure 6.6(e) shows that the applied control is dominated by the required feedback for balancing the under-actuated system. When the desired speed is changed, the controller experiences a new instantaneous perturbation, and re-starts a gradual convergence to the new limit cycle. The theoretical guarantee of a decreasing cost function is maintained (Figure 6.6(f)) and only increases upon the unmodeled perturbation



(a) x and x velocity

(b) pitch and pitch velocity

(c) applied GRFx

(d) cost function

Figure 6.5: Simulation of infinite-horizon LQR tracking deadbeat behavior of simplified model. The initial condition has a 10 degree error in θ and 1cm error in x. (a) shows the tracking of the reference positions and velocities (in dashed green) from the deadbeat stabilized simplified model. Relative errors in these states grow exponentially, as is most evident in the first step. (b) shows the orientation, (c) show the applied horizontal GRF, and (d) shows the LQR cost, which is guaranteed to be decreasing as a function of time.

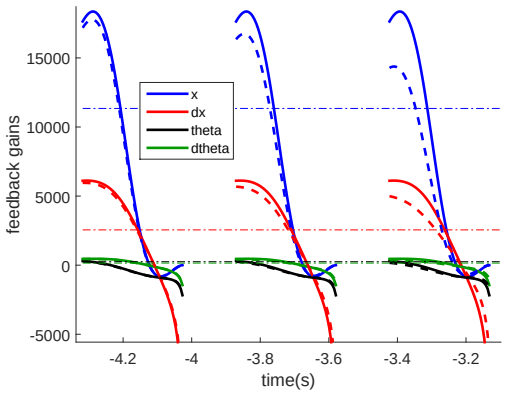
of a change in desired velocity.

The high sensitivity of this controller stems from having no means of independently controlling the two degrees of freedom, and x must be prioritized over θ. This is especially problematic on a low rotational inertia robot like ATRIAS, where generating F_x comes from applying hip torques, which create large accelerations on the orientation. Thus, without accounting for the stabilization available through the discrete control, LTV-LQR techniques like the one proposed in [79] are limited to overly aggressive strategies that are stable in simulation but not applicable to hardware due to limited bandwidth and limited disturbance rejection.

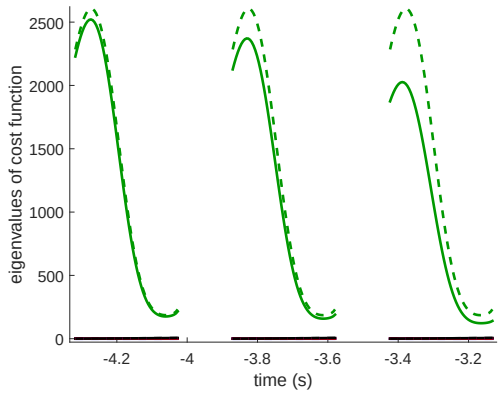
6.5.3 Coupled states and potential revision of leg placement policy

Earlier in this thesis (Chapter 1), we raised the question of how the extra degrees of freedom and the associated feedback control affect the assumed behaviors of the states described by the simplified model. With linear closed-loop error dynamics through $\boldsymbol{u} = -\boldsymbol{K}(t)\boldsymbol{\xi}$, the inter-dependency of the different states of the system is now explicitly known; the multi-state error vector evolves through $\boldsymbol{\xi}_{TO} = \boldsymbol{T_s}\boldsymbol{\xi}_{TD}$. For example, from the [2,3] entry of the infinite-horizon closed loop transition matrix in Equation 6.9, a 0.5 radian (29°) error in pitch θ at touchdown will lead to a 0.14 m/s error in horizontal velocity $\dot{x}$ at takeoff. While the math is not new and the result is highly intuitive, it does not seem to be a commonly examined perspective in the literature, despite the ubiquity of trade-offs between dynamically coupled states in legged systems.

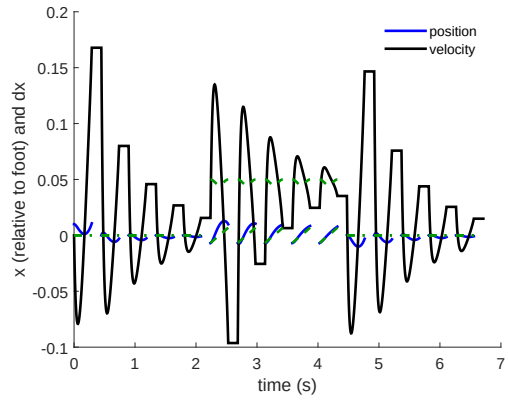
One possible conclusion is that these coupling terms are small – we accept that non-zero pitch θ and pitch velocity $\dot{\theta}$ introduce minor perturbations to the desired deadbeat velocity tracking, and we keep the deadbeat leg placement from the low order model that is agnostic to these rotational states. The smallness of the coupling terms comes from the low rotational inertia I_t of the robot. In 6.5.2, the low inertia prevents us from balancing in place; here, it can justify ignoring the rotational dynamics effects on the translational dynamics.



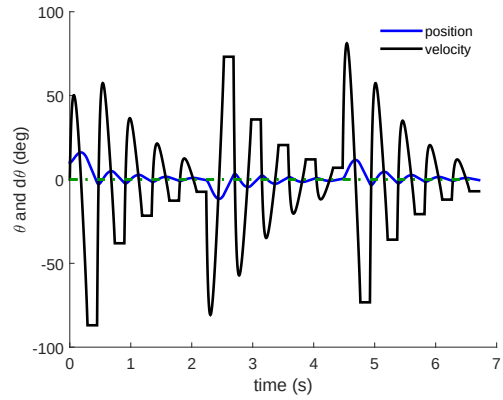
(a) fixed limit cycle gains



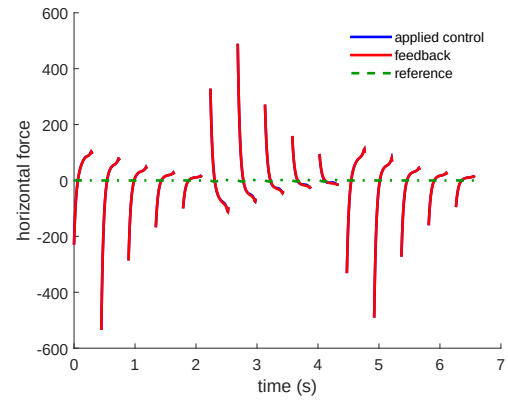
(b) fixed limit cycle costs



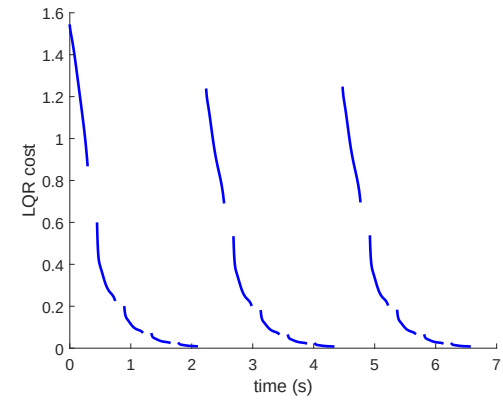
(c) x and x velocity



(d) pitch and pitch velocity



(e) applied GRFx



(f) cost function

Figure 6.6: Simulation of LTV-LQR tracking of single non-adaptive limit cycle. LTV-LQR derives stability by using x and gravity to control the pitch. These gains are not realistic for the hardware; they exceed the actuator bandwidth. Rejection of errors is slow despite large feedback forces. Convergence is theoretically guaranteed.

Alternatively, just as we designed the vertical leg placement in Section 6.4.2, we can use $\boldsymbol{T_s}$ to revise the original leg placement policy. The state errors have the discrete dynamics of

$$\boldsymbol{\xi}_{TD,i+1} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & t_c \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{T_f}} \boldsymbol{\xi}_{TO,i} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta x_{TD},$$

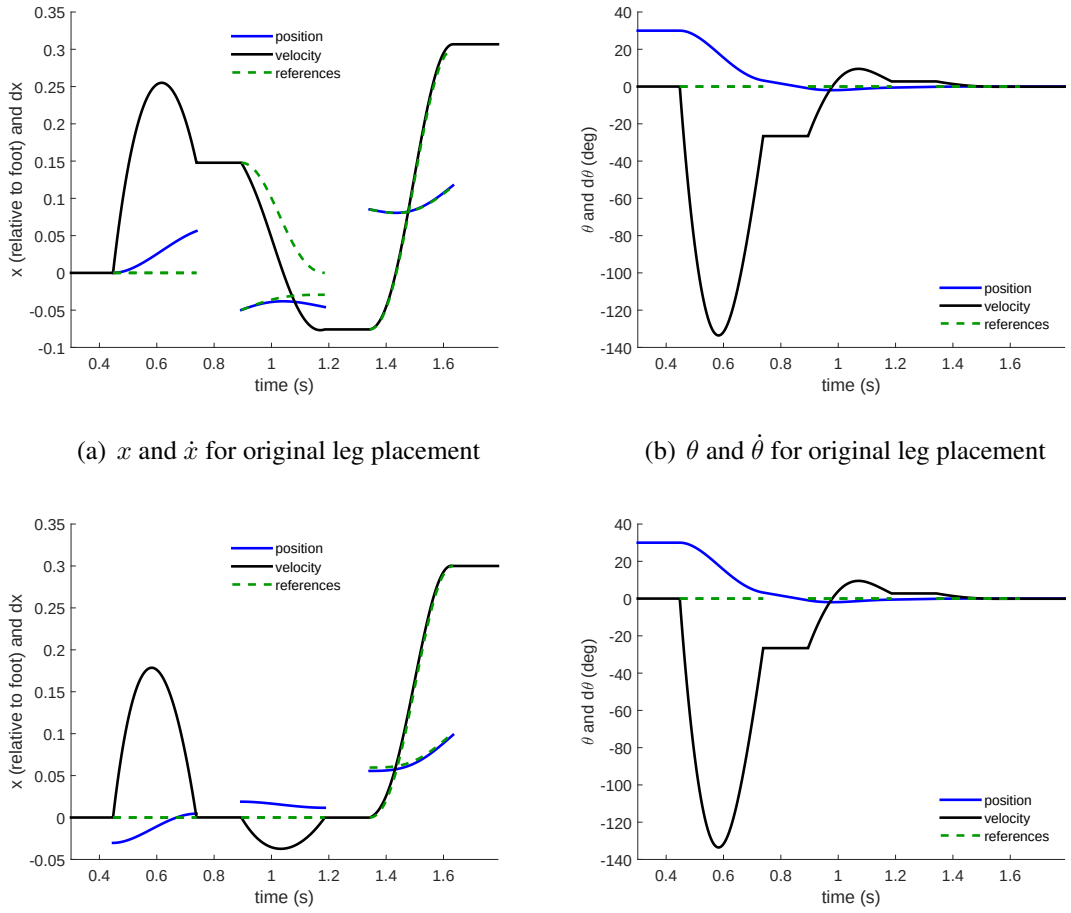
$$\boldsymbol{\xi}_{TO,i+1} = \boldsymbol{T_s} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & t_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{\xi}_{TO,i} + \boldsymbol{T_s} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta x_{TD} \tag{6.10}$$

Thus, to re-formulate deadbeat velocity control that takes the rotational states into account, we could derive

$$\begin{aligned} \delta x_{TD} &= -\frac{1}{\boldsymbol{T}_{s,[1,2]}} \left(\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \boldsymbol{T_s T_f} \right) \boldsymbol{\xi}_{TO,i}, \\ &= -\begin{bmatrix} 0 & 0 & 0.06 & -0.04 \end{bmatrix} \boldsymbol{\xi}_{TO,i}. \end{aligned} \tag{6.11}$$

where $\boldsymbol{T}_{s,[1,2]} = 4.88$ is the adjusted leg placement's effect on the velocity error in flight and selection matrix $\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$ singles out the velocity dynamics in transition matrix $\boldsymbol{T_s T_f}$. This computation yields gains that exactly cancel out the pitch-induced errors on $\dot{x}_{TO}$. See Figure 6.7.

Instead of targeting perfect velocity control at the expense of the other states, DLQR yields a feedback law that uses leg adjustments to trade between long term errors in all the states. We observed that DLQR produces nearly the same policy as the adjusted deadbeat velocity law; this is outcome of the pitch-stabilization in stance being largely independent of the leg placement. Iterating the continuous and discrete optimizations as described in Section 6.3.1 likewise has very little additional effect.



(a) x and $\dot{x}$ for original leg placement

(b) θ and $\dot{\theta}$ for original leg placement

(c) x and $\dot{x}$ for adjusted leg placement

(d) θ and $\dot{\theta}$ for adjusted leg placement

Figure 6.7: Adjusting the deadbeat law for the full order model. Given closed loop stance dynamics $\boldsymbol{A_s}$, we can modify the leg placement to compensate for velocity errors induced by θ and $\dot{\theta}$, predicted from the takeoff state. This yields a linear modification to the original deadbeat leg placement. The plots in the top row use the simplified model leg placement policy, the plots in the bottom row use the modified leg placement policy. Both continuous controllers use the same LTV-LQR. The system is simulated from a takeoff state with 30° of pitch error. 4 flight and 3 stance phases are plotted; the blue x position is plotted only in stance. Leg placement adjustments are the positional offsets (difference between blue and dashed-green) at each touchdown in Figure 6.7(c). The velocity in black thus returns to the desired value by the end of stance. Without the adjustment, the initial 30° error leads to a 15 cm/s velocity error in the first flight phase in Figure 6.7(a). Notice the target leg-placement for the next stance phase is the deadbeat correction of this 15 cm/s velocity error. The adjusted deadbeat law keeps a nominal reference of 0 in the second step due to having no velocity error in flight, but it makes a 2 cm adjustment in leg placement to predictively account for the pitch errors.

6.5.4 Open loop continuous control

In Section 5.9, it was unclear how well the simplified-model-based discrete control could function without continuous feedback. Here, we show that discrete feedback in the leg placement alone is sufficient to stabilize the under-actuated x-θ dynamics and can execute the deadbeat transitions of the simplified model.

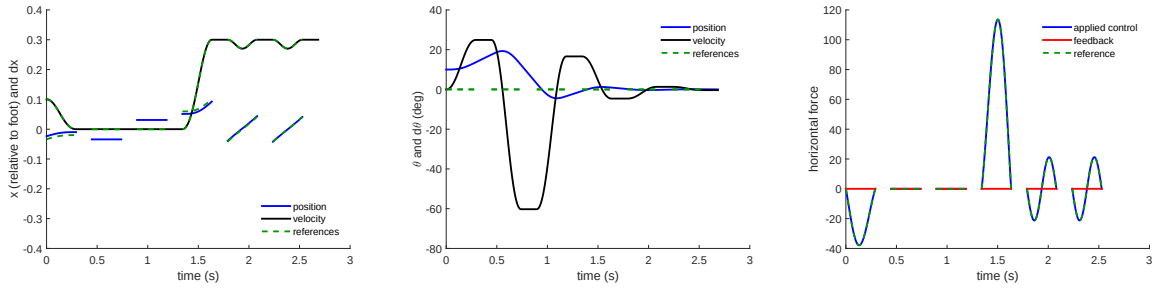
We zero out the continuous force feedback gain $-\boldsymbol{K}(t) = 0$, integrate $\boldsymbol{T_s(t)}$ from the continuous open-loop dynamics, and solve discrete LQR for the discrete error dynamics in Equation 6.10. This process returns a leg adjustment policy of

$$\delta x_{TD} = -\boldsymbol{k}_{x,TD}\boldsymbol{\xi} = \begin{bmatrix} 0 & 0 & 0.065 & 0.043 \end{bmatrix} \boldsymbol{\xi}$$

that is added to the original deadbeat leg placement. Simulation shows that this discrete-only control demonstrates good error rejection from a perturbed initial condition and executes deadbeat velocity tracking (Figure 6.8). The adjusted leg placements can be seen in Figure 6.8(a) to be reasonable.

This perhaps non-intuitive result means that as long as the stance dynamics are predictable (i.e, $\boldsymbol{T_s}$ can be computed), leg placement can offer sufficient control authority to exponentially stabilize the pitch while deadbeat stabilizing the translational states. Note that this control is formulated as a feedback modification of the original deadbeat control; the linear structure of the error dynamics makes deriving such a policy quite straightforward. The unknown practicality of this hypothetical strategy defines a strong motivation for the analysis we present in Chapter 7.

Note that if F_x^* were not defined as a function of time but defined instead as the state-based behavior $F_x^*(\boldsymbol{s}, F_z) = \frac{x}{z}F_z$ (i.e, keep the GRF pointing through the CoM), then $\boldsymbol{T_s}$ would yield a uncontrollable system in Equation 6.10 and DLQR fails. Intuitively, this definition of "feedforward" leaves no control authority over the angular momentum.



(a) x and $\dot{x}$ (b) θ (c) applied forces F_x

Figure 6.8: Open loop continuous control with DLQR leg placement. Adjustments to the deadbeat law that correct the pitch from step to step stabilize the system from initial error of 10 cm/s in velocity, 1 cm in position, and 10 degrees in pitch. The initial velocity error lies directly on top of the velocity reference – the first step chooses a leg placement and plans a motion that perfectly cancels this error.

6.6 Discussion and conclusions

6.6.1 Summary of contributions

In this chapter, we used linear theory to define the closed loop error dynamics of the full order system as a factorized Poincare mapping $\boldsymbol{\xi}_{TO,i+1} = \boldsymbol{T_s}\boldsymbol{T_f}\boldsymbol{\xi}_{TO,i}$. The matrix $\boldsymbol{T_f}$ encodes the deadbeat stability of the closed loop simplified model, and we are thus able to formulate infinite horizon LTV-LQR for the continuous controller with explicit knowledge of the transferred stability. This optimal control approach guarantees stability, identifies non-intuitive but potentially effective control strategies that exhibit good tracking with low feedback gains, and remains compatible with optimal control formulations often applied to design the discrete controller. In addition, the linear system $\boldsymbol{\xi}_{TO,i} = \boldsymbol{T_s}\boldsymbol{\xi}_{TD,i}$ clearly gives the dynamic coupling between state errors. This can be leveraged to augment the simplified model with additional feedback in the full order space; in particular, we can re-formulate the deadbeat law in the full order model. Finally, these linear error dynamics set up a useful framework for the sensitivity analysis developed in Chapter 7.

Applying these methods to our decoupled systems reveals new insights about control of the

centroidal dynamics on a under-actuated runner. In z, the touch-down height (or leg length) is a under-utilized control input of the spring-mass model that can contribute to stability and reduce the required feedback gains. Continuous feedback may not be necessary, even in the under-actuated x-and-θ subsystem. In the x-and-θ subsystem, overall performance is drastically improved by allowing the use of leg placement for velocity control; this comes from the limited control authority of a low rotational-inertia system. As a result, ATRIAS is well suited for decoupled control design (including the point-mass spring-mass model control we use) and poorly suited for standing in place.

6.6.2 Relationship with HZD

As covered in Section 6.1.3, one of the key distinctions between the framework used here and the HZD formulation is the order in which the continuous and discrete control are designed. The theme of this thesis is more closely aligned with starting from the discrete system such that we can explicitly leverage the stability derived from the discrete control of the simplified model. The examples in Sections 6.4.2 and 6.5.3 show that these design processes are not mutually exclusive and that they can even be performed iteratively. However, the HZD perspective does not easily access optimal control for the design of the continuous control based on the properties of the discrete control.

The other primary distinction between the two frameworks is defining the reference as a trajectory versus as a set of dynamics. Due to ATRIAS's inertial properties, x and θ are forced into nearly independent control strategies where x is addressed discretely and θ is handled continuously, and there are no issues with relying on the discrete control to generate a stabilizing solution based on the reduced order reference dynamics. However, on systems with more rotational inertia, it becomes increasingly important to solve the under-actuated balancing problem using the dynamics of the full order model. In the framework we develop here, optimal control naturally transitions to these momentum-based balancing strategies as they become more rele-

vant. In addition, formulating the control as trajectory following allows the smooth transition between different gaits and behaviors without being constrained to periodic limit cycles.

Time dependence

LTV-LQR uses a time-varying model to take advantage of the available theory for optimal control of linear systems. In place of optimal time-varying gains, HZD often returns time-invariant feedback laws. This allows the control to not lock onto a pre-defined trajectory and have to catch up to (or slow down to) some reference state. In the trajectory following formulation, re-planning at every step alleviates this issue, and we do not observe significant problems with this effect in our experiments. Formulating the control in terms of transverse dynamics [109] is also a possible solution, but the math for this technique is complicated. [98] presents a detailed study on time-based control of state-based constraints.

Furthermore, it is not clear that time-invariant formulations are in fact more practical. For example, consider the spring force generated by the virtual leg. This reference behavior could be implemented as the time-invariant state feedback $F_z = k_{\text{spring}}(z - z_0)$, or it could be implemented as a $F_z(t) = z^*(t)$ function of time. Note that the different formulations would lead to different $\boldsymbol{A}$ matrices in the linearized error dynamics. Since ATRIAS does not have a parallel spring in the leg, this virtual spring force F_z is generated through closed loop control. Using a time invariant virtual spring results in $F_z^{\text{cmd}} = k_{\text{spring}}(z - z_0) - \boldsymbol{K\xi}$, which puts k_{spring} into the feedback loop. Time varying control results in $F_z^{\text{cmd}} = F_z^*(t) - \boldsymbol{K(t)\xi}$. Here, $k_{\text{spring}} = 16000$, which is much larger than the position gain for feedback correction. In practice, high gain feedback can cause issues with lower level control loops, though this effect can be alleviated by careful estimator design.

6.6.3 Relevance to other robots

This chapter presents a comprehensive theoretical study on the stabilizing the centroidal dynamics of a hybrid legged system using linear optimal control. These results directly generalize to the balancing problem for multi-link robots with significant inertia using the existing techniques that solve inverse dynamics with a hierarchical quadratic program or with weighted task space priorities (Section 2.2.2). In particular, the formulations used in [104] or [51] could directly use the cost function returned by the LTV-LQR in the QP inverse-dynamics. This results in model-predictive balancing that accounts for trade-offs between linear and angular momentum and leverages leg placement. These strategies are required for under-actuated robots like ATRIAS, and remain pertinent to fully actuated robots with limited ankle actuation imposed by contact wrench constraints.

6.6.4 Unclear practicality

The potential improvements to the control derived in Sections 6.4 and 6.5 have highly promising convergence properties, but unknown practical value. Realistically, low continuous gains may respond poorly to perturbations in stance, and pure leg placement strategies may be overly sensitive to model inaccuracies and leg placement errors. Without quantified measures of sensitivity to realistic conditions, these proposed designs carry little weight. Chapter 7 works towards extending the theoretical description of the control to account for these effects.

Chapter 7

Expected performance in the presence of noise and uncertainty

The existing literature in legged locomotion lacks theoretical studies on the interpretation of experimental data (see Section 2.5). In principle, the advantage of model based design over empirical techniques are the guarantees of stability and the potential for more precise behaviors as well as optimality. In practice, the attainment of these goals cannot be meaningfully evaluated without a quantified extension of the theory. Convergence proofs guarantee perfect tracking of a nominal limit cycle, but this never achieved on hardware. The various reasons why perfect tracking is impossible are well understood, but without quantifying these effects, there is no measure of "how close" should the produced behavior be. As a result, the value of the formal theory remains unknown. Similarly, studies of simplified models can propose deadbeat tracking, but there is no theoretical benchmark for judging imperfections in the experimental implementation. In this chapter, we derive an extension to the theory that allows meaningful evaluation of experimental results obtained in the presence of bounded uncertainties and perturbations.

7.1 Overall approach

In this study, we are not trying to derive nor prove stable interactions between actuator dynamics, estimator dynamics, and outer loop control. This challenge is often approached at a theoretical level using passivity [4, 138, 198] arguments or H_∞ [91, 175] control. In our implementation, we used the widely practiced techniques [17] of separating bandwidths and applying filtering to avoid instabilities. Given the performance we achieve through tuning our inner loops, we now study the performance at the higher level of tracking the desired spring mass running gaits. We make coarse estimates that quantify our actuator and estimator performance, and we want to analyze how the theory performs subject to these errors compared to how our hardware actually performed.

The linear framework we set up in Chapter 6 provides a platform that naturally extends to analyzing the effects of bounded error. In this chapter, we modify the equations of the nominal error dynamics to explicitly account for linearization errors, force tracking errors, and leg placement errors, which all present as disturbances to the assumed convergence of the closed loop system. As a result, instead of theoretical convergence to perfectly follow the reference trajectory, we derive an attractive and invariant region in terms of state errors. The system is guaranteed to converge to and then stay within this envelope, thereby defining theoretical expectation that can be validated by experimental data. Furthermore, from breaking down the sources of error into quantified components, we can make predictive statements about how various improvements to the lower level systems would affect the final higher level performance. Throughout this chapter, we base our analysis on data from the experiments we reported in Section 5.8.

7.2 Assumptions

We start with basic assumptions about the lower level systems and any external disturbances. Specifically, we assume magnitude bounds on our estimation error, our force tracking error,

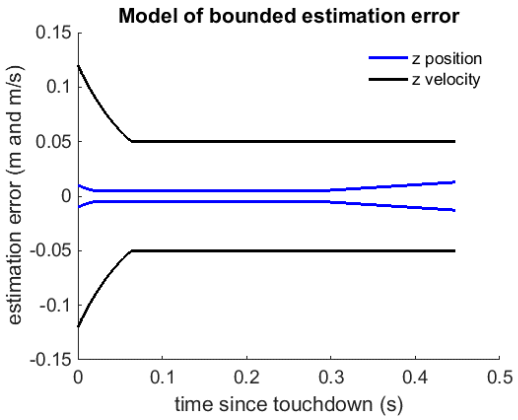
and our leg placement error. The goal is not to formulate precise, robust guarantees based on cautious models of the sources of error; rather we are simply constructing quantified handles on the perturbations to allow the theoretical study of the propagation of errors.

7.2.1 Bounded estimation error

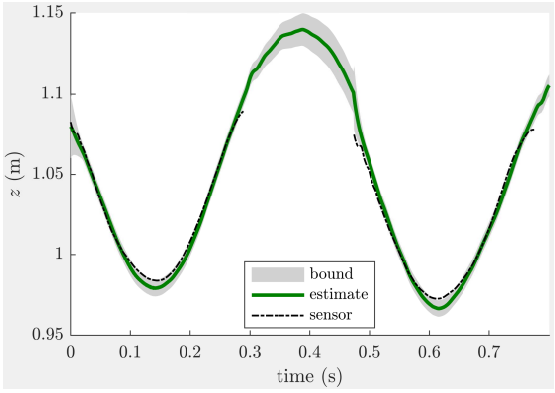
The true, unobservable state vector is $\boldsymbol{s}$, while all the control acts on the output $\hat{\boldsymbol{s}}$ of the Kalman filter (Section 5.6.1). We compute an estimated a state error of $\hat{\boldsymbol{\xi}} = \hat{\boldsymbol{s}} - \boldsymbol{s}^*$, while the actual state error is $\boldsymbol{\xi} = \boldsymbol{s} - \boldsymbol{s}^*$. The difference between the estimated vectors and the real vectors is uncertainty $\boldsymbol{\sigma} = \boldsymbol{\xi} - \hat{\boldsymbol{\xi}} = \boldsymbol{s} - \hat{\boldsymbol{s}}$.

Kalman filtering does not guarantee bounded estimation errors; we are imposing an artificial assumption. We have no means of extracting ground truth for our states $\boldsymbol{s}$, but we can approximate the uncertainty based on comparing filter data to raw sensor data (Figure 7.1). Furthermore, the covariance of the Kalman filter represents of the uncertainty in the state estimation, and we define a model of the estimation error that has a similar behavior. Figure 7.1(a) shows the assumed magnitude bounds $|\sigma_z|^{\max}(t)$ and $|\sigma_{\dot{z}}|^{\max}(t)$. The position uncertainty converges quickly in stance through measuring the joint angles of the stance leg, while the velocity uncertainty converges more slowly due to the required filtering of the noisy position signals. In flight, there are no informative sensor readings. The velocity uncertainty does not grow significantly since the acceleration of the COM is known with fairly high confidence for the ballistic trajectory. The position uncertainty grows due to integrating the uncertain velocity at takeoff.

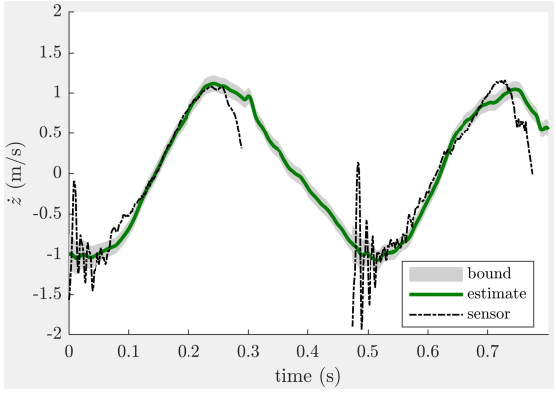
Comparing the resulting envelope (gray region in Figures 7.1(b) and (c)) around the state estimate to the unfiltered encoder data provides a qualitative assessment of the appropriateness of these bounds.



(a) assumed error bounds



(b) estimated z vs raw sensors



(c) estimated $\dot{z}$ vs differentiated and filtered sensors

Figure 7.1: Model of bounded estimator error. (a) shows the assumed bounds, which are motivated by what can be seen in (b) (c) as well as the dynamics for propagating uncertainty in a Kalman filter. (b,c) The Kalman filter state estimate is padded by an envelope (shaded gray) for the model of maximum uncertainty. The coordinates of the CoM cannot be measured, but they are inferred through measured joint angles and a model of ATRIAS mass distribution. There is no available measurement of ground truth for validating these estimates.

7.2.2 Bounded force error

There are three primary sources of disagreement between the GRFs produced on the machine and the nominal closed loop GRFs $\boldsymbol{v} = \boldsymbol{v}^* - \boldsymbol{K}(t))\boldsymbol{\xi}$ assumed in the linear model of the closed-loop error dynamics $\boldsymbol{\xi}_{TO} = \boldsymbol{T}_s\boldsymbol{\xi}_{TD,i}$ (Equation 6.2). First, the feedback controller acts on the estimated state error $\hat{\boldsymbol{\xi}}$ in place of the ideal state error $\boldsymbol{\xi}$, creating a mismatch of $-\boldsymbol{K}(t)\boldsymbol{\sigma}$. Given the assumed estimator uncertainty bounds $|\sigma_z|^{\max}(t)$ and $|\sigma_{\dot{z}}|^{\max}(t)$ as functions of time, the corresponding time-varying bounds on the scalar mismatches $-\boldsymbol{K}(t)\boldsymbol{\sigma}$ for F_x and for F_z are each solved as a linear programs. This component of the force error is shown as the difference between the red and blue lines in Figure 7.2.

Secondly, any saturations imposed by contact wrench constraints modify the originally commanded GRF (Section 5.4.2). These effects are recorded and observed to be both small and short duration (Figure 7.3). This observation shows that the trajectory following problem for a spring-mass trajectory is not significantly hindered by the contact constraints (i.e, the nominal trajectory has good margins with respect to the contact wrench limits), and applying LQR in place of constrained MPC is an appropriate approximation to the constrained MPC optimal solution. Note that the saturations still provide an important safeguard against destabilizing the contact condition, but they do not significantly perturb the performance of the trajectory tracking.

Finally, the SEA controllers do not perfectly track the desired spring deflections that correspond to the commanded GRFs. We map the measured spring deflections from the experimental data to equivalent "measured" GRFs, which are then compared to the commanded GRFs. At the motor control level, whenever the desired rotor velocities are being properly matched by the on-board motor controller, the SEA control nominally produces first-order exponential convergence of any spring deflection error. However, the experimental data shows that the motors are often saturated at their peak current and cannot provide sufficient torque to match the rotor velocities commanded by the SEA controllers (Section 4.4.1). As a result, the recorded GRF errors do not decay exponentially, as shown in Figure 7.2.

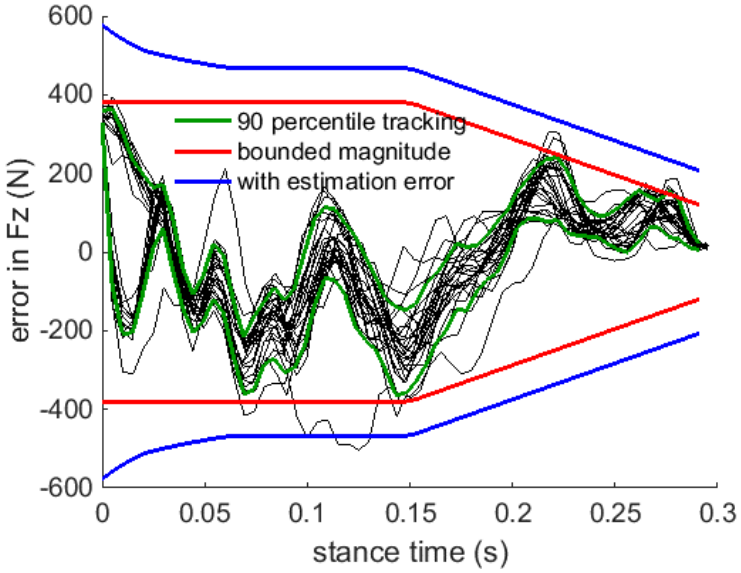


Figure 7.2: Force errors. Black lines are the recorded differences between measured force and commanded force. Green lines are the 90-percentile bounds (for removing outliers), and the red line is a resulting magnitude bound that is fit to the data. The difference between red and the blue comes from adding the effects of $\boldsymbol{K\sigma}$ for bounded $\boldsymbol{\sigma}$.

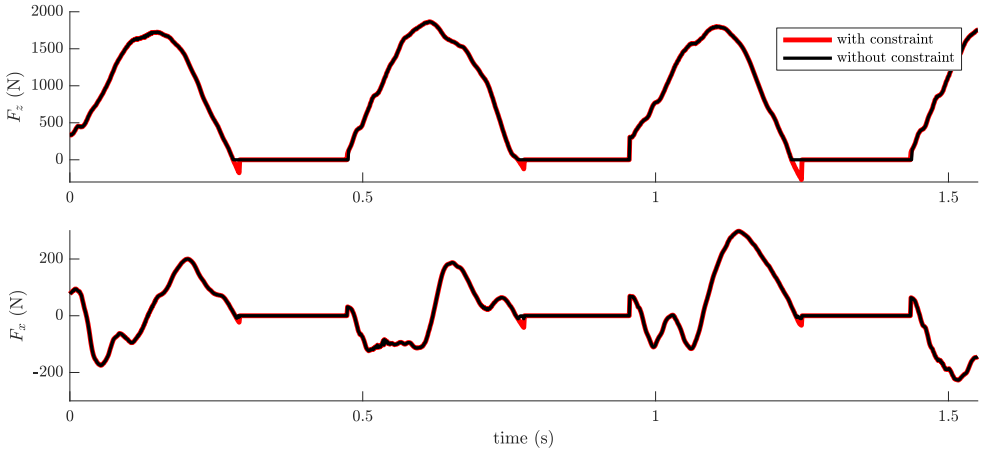


Figure 7.3: Hardware data of saturation of commanded GRFs for contact wrench constraints. To ensure the foot stays in stable contact, F_z^{cmd} is constrained to be non-negative, and F_x^{cmd} is constrained to friction cone described by $|F_x^{\text{cmd}}| \leq \mu_f F_z|$ where $\mu_f = 0.5$. These constraints saturate the original red forces from LQR to the black values. This plot shows that these effects are small and short duration; the forces are nearly always identical.

As discussed in Section 5.4.2, the boom also provides unmodeled forces, but these effects are already accounted for by a disturbance observer that scales the force commands accordingly.

These analyses show that the primary source of force error in these experiments is the SEA tracking. Applying feedback to potentially incorrect state estimates and constraining our control to the feasible contact wrench have much smaller effects.

The observed force tracking errors closely follow a repeatable pattern, so we define two different models of total force error. As a more practical assumption for treating the closed loop SEA control as a black box, we define a symmetric magnitude bound with the blue envelope in Figure 7.2. For evaluating how well the ensuing analyses in this chapter describe the observed data, we define a tighter envelope with the green bounds, which are defined by the 10 and 90 percentiles of the recorded force errors.

7.2.3 Bounded nonlinearities and other modeling errors

From Equation 5.5, the linear model of the closed loop error dynamics drops the term

$$\delta_{nl}(\xi, u) = (x - x^*)(F_z - F_z^*) - (z - z^*)(F_x - F_x^*),$$

This omission describes a disturbance torque (with subscript $_{nl}$ for nonlinear). When the dynamics are further simplified to decouple the control design by assuming $x^* = F_x^* = 0$ in the $A_{\text{decoupled}}$ and $B_{\text{decoupled}}$ matrices in Equation 5.7, an additional disturbance torque of

$$\delta_{dc} = -(F_x^*)(z - z^*) + (x^*)(F_z - F_z^*)$$

(with subscript $_{dc}$ for decouple) is introduced.

These disturbance torques and are plotted in Figure 7.4 to examine the practical cost of our simplifying assumptions. Compared to the disturbances imparted by the force tracking errors $-z^*(t)\delta_x + x^*(t)\delta_z$ in Equation 7.1, the non-linearities and decoupling-induced errors are indeed negligible. We then fit empirical bounds to each of these sources, as illustrated in Figure 7.2 for the empirical bounds on the generalized force errors.

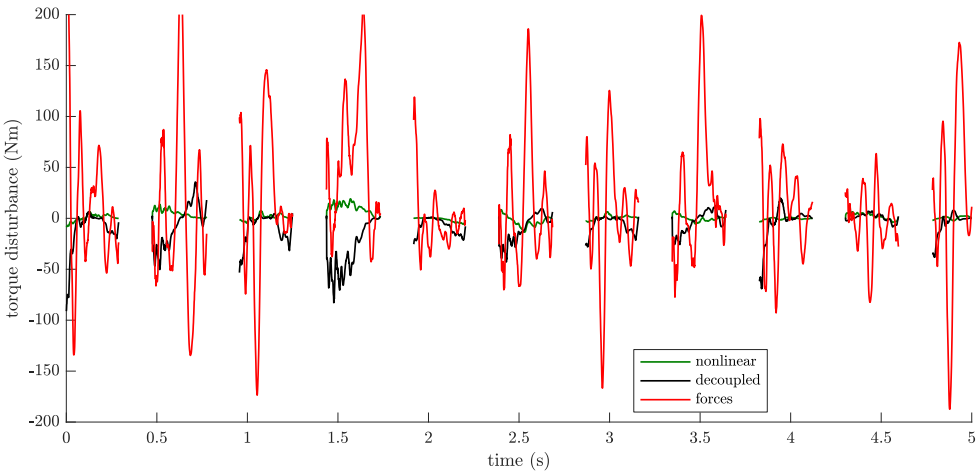


Figure 7.4: Torque perturbations. The non-linearities, de-coupling assumptions, and force tracking errors all contribute perturbing torques to the pitch of the system. This plot shows that the non-linearities and de-coupling assumptions are not significant compared to the force-tracking errors. The centroidal dynamics are very nearly linear, and almost de-coupled.

7.3 Drift accumulated in one step

In this section, the theoretical toll that these disturbance forces and torques take on the tracking performance of the nominal controller is computed. We are then able to derive the requirement for "the theory to be validated by the data," subject to the assumptions made in Section 7.2.

Defining the time elapsed in stance $t_s = t - t_{TD}$, the perturbed error dynamics in stance are

$$\boldsymbol{\xi}(t_s) = \boldsymbol{\Phi}(t_s, 0)\boldsymbol{\xi}_{TD} + \Delta_s(t_s),$$

where the "drift" term $\Delta_s(t_s)$ describes the accumulated deviation from the assumed linear error dynamics (Equation 6.2) given by state transition matrix $\Phi(t_s, 0)$.

The decoupled error dynamics (Equation 5.7) augmented with the corresponding perturba-

tions are given by

$$\dot{\boldsymbol{\xi}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{F_z^*(t)}{I_t} & 0 & 0 & 0 & \cancelto{0}{-\frac{F_x^*(t)}{I_t}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{A_{\text{decoupled}}(t)} \begin{bmatrix} x - x^* \\ \dot{x} - \dot{x}^* \\ \theta - \theta^* \\ \dot{\theta} - \dot{\theta}^* \\ z - z^* \\ \dot{z} - \dot{z}^* \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & 0 \\ -\frac{z^*(t)}{I_t} & \cancelto{0}{\frac{x^*(t)}{I_t}} \\ 0 & 0 \\ 0 & \frac{1}{m} \end{bmatrix}}_{B_{\text{decoupled}}(t)} \begin{bmatrix} F_x - F_x^* \\ F_z - F_z^* \end{bmatrix}$$

$$+ \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 \\ \frac{1}{I_t} & -\frac{z^*(t)}{I_t} & \frac{x^*(t)}{I_t} \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} \end{bmatrix}}_{\boldsymbol{B_p}(t)} \underbrace{\begin{bmatrix} \delta_{nl} + \delta_{dc} \\ \delta_x \\ \delta_z \end{bmatrix}}_{\boldsymbol{\delta_p}}, \tag{7.1}$$

where the ideal GRFs $[F_x, F_z]$ come from linear feedback on $\boldsymbol{\xi}$. $\boldsymbol{B_p\delta_p}$ describes all the perturbations, which consist of GRF mismatches δ_x, δ_z and modeling errors δ_{nl} and δ_{dc}. Substituting in the linear feedback for the GRFs yields the closed loop dynamics of

$$\dot{\boldsymbol{\xi}} = \boldsymbol{A_{cl}}(t)\boldsymbol{\xi} + \boldsymbol{B_p}(t)\boldsymbol{\delta_p}$$

with closed loop matrix

$$\boldsymbol{A_{cl}}(t) = \boldsymbol{A}(t) - \boldsymbol{B}(t)\boldsymbol{K}(t).$$

The general solution to a linear time-varying system (Eq. 6.1) solves the evolution of the perturbed error dynamics:

$$\boldsymbol{\xi}(t_s) = \boldsymbol{\Phi}(t_s, 0)\boldsymbol{\xi}_{TD} + \int_0^{t_s} \boldsymbol{\Phi}(t_s, t)\boldsymbol{B_p}(t)\boldsymbol{\delta_p}(t)dt$$

This yields the affine transformation $\boldsymbol{\xi}(t_s) = \boldsymbol{\Phi}(t_s, 0)\boldsymbol{\xi}_{TD} + \boldsymbol{\Delta_s}(t_s)$ of the state error $\boldsymbol{\xi}$ from touchdown to any later time in the stance phase. Recall from Section 6.2.2 that the state

transition matrix $\boldsymbol{\Phi}(t_s, 0)$ is numerically integrated as a function of the nominal closed loop dynamics $\boldsymbol{A_{cl}}(t_s)$. $\Delta_s(t_s)$ is the drift accumulated during the stance phase, and we now compute its bounds based on the bounded assumptions of the perturbation forces and torques in $\boldsymbol{\delta_p}$.

To formulate boundedness in terms of state errors $\boldsymbol{\xi}$, we first define a distance metric using the weighted L_2 norm $|\ |_{\boldsymbol{W}}$ for a symmetric, positive-definite weighting matrix $\boldsymbol{W}$:

$$|\boldsymbol{\xi}| \equiv \boldsymbol{\xi}^T \boldsymbol{W} \boldsymbol{\xi}.$$

$\boldsymbol{W}$ set to a semi-definite (not a proper norm) rank one matrix isolates the measurement of a single state. Choosing $\boldsymbol{W}$ to be the cost-to-go from the LQR design of the continuous feedback defines error as the expected total cost of the control. $\boldsymbol{W}$ can also be generated by solving the discrete Lyapunov function for the stabilized periodic system (see Section 7.5.1).

Applying the triangle inequality of integrals yields

$$|\Delta_s(t_s)| = |\int_0^{t_s} \boldsymbol{\Phi}(t_s, t)\boldsymbol{B_p}(t)\boldsymbol{\delta_p}(t)dt|,$$
$$|\Delta_s(t_s)| \leq \int_0^{t_s} \underbrace{|\boldsymbol{\Phi}(t_s, t)\boldsymbol{B_p}(t)\boldsymbol{\delta_p}(t)|}_{|\dot{\Delta}_s|(t_s,t)} dt.$$

Thus, a bound of the drift can be solved through maximization of the integrand. Maximizing this expression subject to linear constraints on δ_p is a non-convex quadratic program:

$$|\dot{\Delta}_s|^{\max}(t_s, t) = \max \boldsymbol{\delta_p}(\boldsymbol{B_p}^T(t)\boldsymbol{\Phi}(t_s, t)^T \boldsymbol{W}^T \boldsymbol{\Phi}(t_s, t)\boldsymbol{B_p})\boldsymbol{\delta_p}, \tag{7.2}$$

subject to

$$\delta_x^{\min} \leq \delta_x \leq \delta_x^{\max},$$
$$\delta_z^{\min} \leq \delta_z \leq \delta_z^{\max},$$
$$\delta_m^{\min} \leq \delta_m \leq \delta_m^{\max},$$

where modeling error $\delta_m = \delta_{nl} + \delta_{dc}$ is the sum of the torque errors induced by non-linearities and decoupling assumptions.

This optimization problem is generally NP-hard, but it does not need to be solved in real time. We use Matlab's multi-start algorithm to search for the global maximum. Note that if

a rank one weighting matrix $\boldsymbol{W}$ is used for isolating the drift along a single coordinate (i.e, $\boldsymbol{W_x} = \texttt{diag([1 0 0 0 0 0])}$), then the non-convex QP reduces to two linear programs that search for the maximum and minimum of the drift rate. Technically, a semi-definite $\boldsymbol{W}$ does not properly define a norm operator, but this is not an issue for the arguments applied in this section. A semi-definite $\boldsymbol{W}$ cannot be used for the derivation of the attractive and invariant region (Section 7.5).

By discretizing the stance phase in the interval $0 < t < t_s$, numerically solving Eq. 7.2 to find $|\dot{\Delta}_s|^{\max}(t_s, t)$, and then numerically integrating this function from $t = 0$ to $t = t_s$, the maximum drift $|\Delta_s|^{\max}(t_s)$ over a finite period t_s is computed (Figure 7.5). This is the largest weighted drift that can accumulate from the perturbations experienced up through time t_s:

$$|\Delta_s(t_s)| \leq |\Delta_s|^{\max}(t_s). \tag{7.3}$$

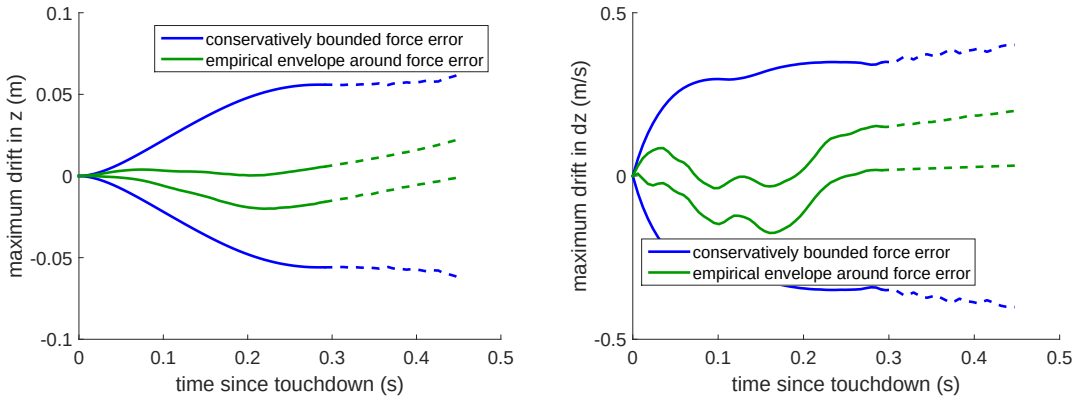
Note that the full computation must be repeated for different values of t_s; the numerically computed integrands cannot be re-used since they explicitly depend on the end time t_s.

7.3.1 Comparison of theory to experimental data

In the presence of these bounded uncertainties and perturbations, estimated state error $\hat{\boldsymbol{\xi}}$ extracted from experimental data can be decomposed into four components:

$$\begin{aligned}
\hat{\boldsymbol{\xi}}(\boldsymbol{t_s}) + \boldsymbol{\sigma}(t_s) &= \boldsymbol{\xi}(t_s) \\
\hat{\boldsymbol{\xi}}(\boldsymbol{t_s}) + \boldsymbol{\sigma}(t_s) &= \boldsymbol{\Phi(t_s, 0)}\boldsymbol{\xi}_0 + \boldsymbol{\Delta}_s(t_s) \\
\hat{\boldsymbol{\xi}}(\boldsymbol{t_s}) + \boldsymbol{\sigma}(t_s) &= \boldsymbol{\Phi(t_s, 0)}(\hat{\boldsymbol{\xi}}_0 + \boldsymbol{\sigma}_0) + \boldsymbol{\Delta}_s(t_s) \\
\hat{\boldsymbol{\xi}}(\boldsymbol{t_s}) + \boldsymbol{\sigma}(t_s) &= \boldsymbol{\Phi(t_s, 0)}\hat{\boldsymbol{\xi}}_0 + \boldsymbol{\Phi(t_s, 0)}\boldsymbol{\sigma}_0 + \boldsymbol{\Delta}_s(t_s)..
\end{aligned} \tag{7.4}$$

- $\boldsymbol{\sigma}(t_s)$ is the uncertainty in the current measurement,
- $\boldsymbol{\Phi(t_s, 0)}\hat{\boldsymbol{\xi}}_0$ is the expected closed loop error evolution from the estimated initial condition $\boldsymbol{\xi}_0$,



(a) z drift (b) $\dot{z}$ drift

Figure 7.5: Maximum drift $|\boldsymbol{\Delta}_s|^{\max}(t_s)$ accumulated in the vertical states over one step. Stance is plotted in solid, flight is plotted in dashed. The algorithm integrates the worst-case disturbances forces subject to the closed loop dynamics. The blue bounds come from using the looser force bounds from section 7.2.2, which sums SEA tracking errors as well as estimation induced feedback errors and any applied saturations. The green bounds make a less realistic assumption, integrating the repeatable profile of the SEA tracking errors that we observe in the data.

- $\boldsymbol{\Phi(t_s, 0)\sigma_0}$ is the propagation of any estimation error $\boldsymbol{\sigma}_0$ of the initial condition, and
- $\boldsymbol{\Delta}_s(t_s)$ is the accumulated drift.

For the experimental results to validate the theoretical expectations, any measurement $\hat{\boldsymbol{\xi}}(t_s)$ must lie within a feasible range of these four factors. Here, we presented the decomposition with the state at touchdown $t_{TD} = 0$ as an initial condition, but this formulation can also be expressed from an arbitrary choice of initial condition.

7.3.2 Example applied to vertical states

This section shows that the experimental results from Chapter 5 agree with the expected performance, but the derived bounds seem overly conservative.

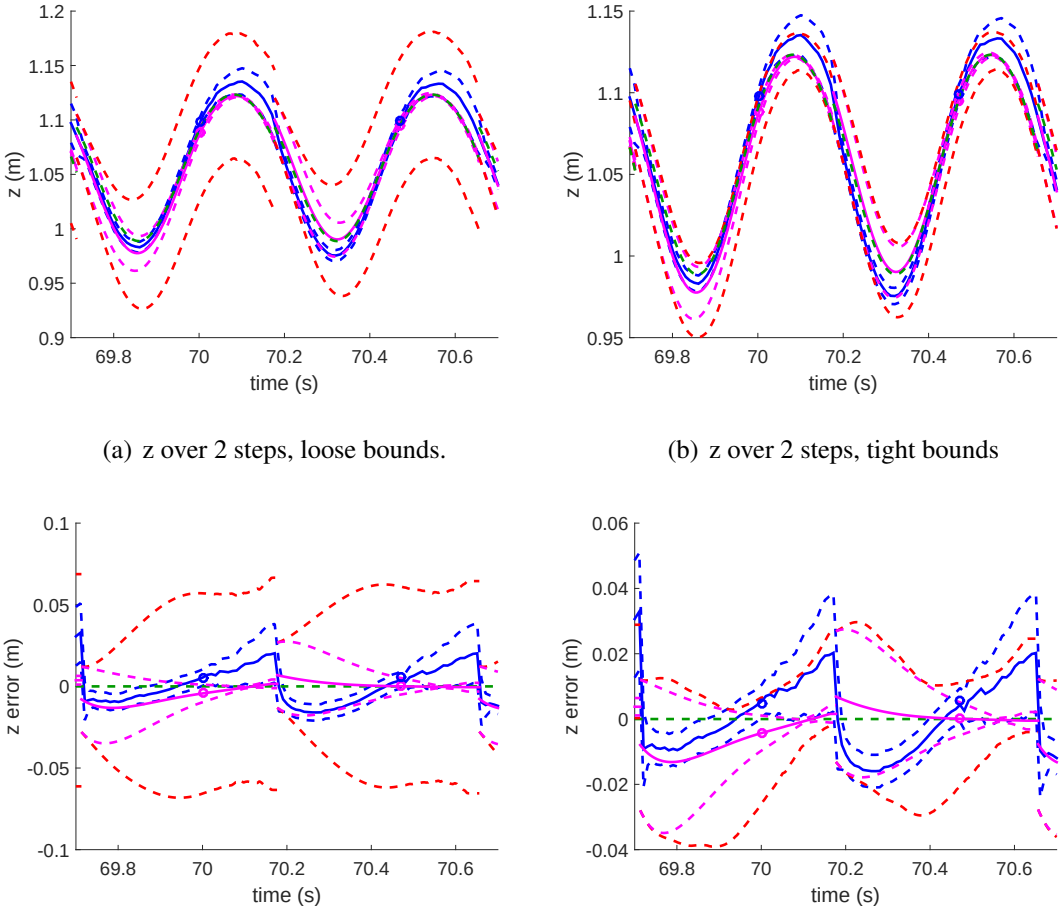
We compute the maximum drift in z and in $\dot{z}$, where we isolate these coordinates by using rank one weighting matrices $\boldsymbol{W}$ to define the norm function $|\ |$. See Figure 7.5.

Maximum drift $|\boldsymbol{\Delta}_s|^{\max}(t_s)$ accumulated in the vertical states over one step is computed by

integrating Equation 7.3 through solving the constrained optimization problem in Equation 7.2. Stance is plotted in solid lines, and flight is plotted in dashed lines (where the drift in flight is integrated from for a uncontrolled system to a small perturbations). This algorithm integrates the worst-case disturbance forces subject to the closed loop dynamics for a given horizon t_s. The blue bounds come from using the loose, symmetrically-bounded force profile from Section 7.2.2 (blue envelope in Fig. 7.2), which sums SEA tracking errors as well as estimation induced feedback errors and any applied saturations. The green bounds make a less realistic assumption, integrating the repeatable profile of the SEA tracking errors that we observe in the data (green envelope in Fig. 7.2).

Now we can examine the tracking we observe in experimental data using the decomposition presented in Equation 7.4. See Figure 7.6. The heights z are plotted in absolute coordinates in the top row for a sense of scale and plotted relative to the reference trajectory in the bottom row for clarity. Two steps are plotted, with touchdowns occurring near $t = 69.7$ and $t = 70.2$. The takeoff event is marked with an open circle. The reference trajectory is the dashed green line. The estimated state $\hat{z}$ is the solid blue line, and it is enveloped by the dashed blue line for maximum measurement uncertainty σ_z (from the assumed model in Figure 7.1). We take each touchdown to be a new initial condition z_0, and the expected closed-loop propagation of z_0 is shown in pink. The closed-loop propagation of the maximum uncertainty of each touchdown defines the pink envelope. Intuitively, if the touchdown state is mis-estimated, the actual state would converge along some trajectory within the envelope, if there were no external perturbations. Finally, the maximum drift $|\Delta_{s,z}|^{\max}$ grows the pink envelope out to the red envelope and accounts for all the error can be accumulated due to the bounded perturbation forces. If the state estimation were perfect, theory expects the blue line to stay within the red envelope. Allowing for estimation errors, theory expects the blue envelope to maintain overlap with the red envelope; the real state must be described by a feasible, perturbed trajectory from some unknown initial condition.

Applying high gain feedback control shrinks both the pink and red envelopes by making state



(a) z over 2 steps, loose bounds.

(b) z over 2 steps, tight bounds

(c) z relative to reference, loose bounds.

(d) z relative to reference, tight bounds.

Figure 7.6: Tracking of vertical trajectory from experimental data. See discussion in Section 7.3.2.

transition matrix $\Phi(t_s, 0)$ strongly contractive. Having accurate state estimation reduces the blue envelope and contributes to reducing the red envelope via reducing the force error associated with erroneous feedback control. Having accurate force tracking collapses the red envelope.

This examination shows that for the loose force bounds that we assume, the trajectory tracking has mediocre guarantees (approximately 5 cm of error in each stance). However, the actual tracking stays well within these bounds, showing the assumption to be overly conservative. This is related to our actual force errors being approximately zero mean, despite having large magnitudes. Instead, by integrate the empirically observed tight force bounds, the theoretically guaranteed tracking better matches the observed errors, thus showing that the hardware does follow the modeled dynamics. To improve performance, we could improve the convergence of $\Phi(t_s, 0)$ (increasing the feedback gains to improve the convergence typically results in larger force tracking errors) or improve the accuracy the SEA control. The red bounds scale linearly with the maximum force errors, so twice as accurate SEA tracking would approximately half the maximum state error.

7.4 Deviations in flight

Just as the deviation of the stance error dynamics from the nominal linear system with bounded drift term $\Delta_s(t_{TO} - t_{TD})$ in the previous section, this section derives the deviation Δ_f in flight error propagation:

$$\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{T_f}\boldsymbol{\xi}_{TO,i} + \Delta_f,$$

where $\boldsymbol{T_f}$ is the linearization derived in Section 6.2.3.

Here, there are primarily two sources of deviation. First, the leg placement policy is based on the state estimate at takeoff $\hat{\boldsymbol{s}}_{TO}$ and is sensitive to estimation error $\boldsymbol{\sigma}_{TO}$. For the policy (Section 5.3) used in the experiments, the horizontal leg placement nominally cancels the speed error based on the speed at takeoff by matching $\dot{x}^*_{TD,i+1} = \hat{\dot{x}}_{TO,i}$. Estimation error on the takeoff speed thus leads to an equal and opposite deviation in the speed error on landing: $\dot{x}_{TD,i+1} -$

$\dot{x}^*_{TD,i+1} = -\sigma_{\dot{x},TO,i}$. The target landing height z^*_{TD} is a fixed policy with no state feedback, so this component has no sensitivity to estimation error $\boldsymbol{\sigma}_{TO}$.

Secondly, the position-based servoing of the swing leg can miss the target $[x^*_{TD,i+1}, z^*_{TD,i+1}]$, where we denote the leg placement error as $[\epsilon_x,\ \epsilon_z]$. These errors come from intentionally ignoring the compliance of the spring (by applying position control directly to the rotor angles, see Section 5.5), imprecise tracking of the desired rotor angles, and any unexpected changes in terrain height. Empirically, we observe errors of approximately 1 cm in each coordinate due to the first two effects. In addition, the kinematic uncertainty $[\sigma_x,\ \sigma_z]$ is added to $[\epsilon_x,\ \epsilon_z]$ due to the state estimation used in closing the position loop on leg placement. In our rough terrain experiments (Section 5.8.4), the steps up and down introduce additional vertical errors of up to 15 cm.

These placement errors $[\epsilon_x,\ \epsilon_z]$ perturb the assumed flight transition. ϵ_x directly becomes a horizontal position deviation $\Delta_{f,x} = x_{TD} - x^*_{TD}$ at touchdown. ϵ_z not only becomes a vertical position deviation $\Delta_{f,z}$ on touchdown, but it also changes the time of contact $t_{c,i}$ (Section 6.2.3). From the ballistic equations describing the flight phase, this results in touchdown deviations in vertical velocity $\Delta_{f,\dot{z}}$ and $\Delta_{f,\theta}$, which can be solved algebraically.

The total effect of estimation error $\boldsymbol{\sigma}_{TO}$ and leg placement error $[\epsilon_x,\ \epsilon_z]$ on the flight error transition is thus given by

$$\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{T_f}\boldsymbol{\xi}_{TO,i} + \boldsymbol{\Delta}_f,$$

where

$$\boldsymbol{\Delta}_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{\sigma}_{TO,i} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & \frac{\dot{\theta}_{TO,i}}{\dot{z}^*_{TO,i}} \\ 0 & 0 \\ 0 & 1 \\ 0 & \frac{g}{\dot{z}^*_{TO,i}} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_z \end{bmatrix}. \tag{7.5}$$

Lastly, there are potentially non-linearities $\Delta_{nl,f}$ (from Equation 6.6) in the nominal takeoff-to-touchdown error mapping that are neglected by matrix $\boldsymbol{T_f}$. These effects can be shown to be negligible.

Applying the assumptions on bounded errors in terms of $|\sigma_{\dot{x},TO,i}| \leq \sigma_{\dot{x}}^{\max}$ (Section 7.2), $|\epsilon_x| \leq \epsilon_x^{\max}$, and $|\epsilon_z| \leq \epsilon_z^{\max}$, Equation 7.5 is used in solving a constrained maximization problem structurally identical to Equation 7.2, thereby yielding the maximum deviation $|\Delta_f|$ in a single flight phase.

7.5 Total state error accumulated across multiple steps

Sections 7.3 and 7.4 derived the bounded drift $\boldsymbol{\Delta}_s$ accumulated in each stance phase and the bounded deviation $\boldsymbol{\Delta}_f$ experienced in each flight phase for the affine mappings

$$\boldsymbol{\xi}_{TO,i+1} = \boldsymbol{T_s}\boldsymbol{\xi}_{TD,i+1} + \Delta_{s,i+1},$$
$$\boldsymbol{\xi}_{TD,i+1} = \boldsymbol{T_f}\boldsymbol{\xi}_{TO,i} + \Delta_{f,i}.$$

This results in the recursive affine mapping

$$\boldsymbol{\xi}_{TO,i+1} = \underbrace{\boldsymbol{T_s}\boldsymbol{T_f}}_{\boldsymbol{A}_d}\boldsymbol{\xi}_{TO,i} + \underbrace{\boldsymbol{T_s}\Delta_{f,i} + \Delta_{s,i+1}}_{\Delta_{\text{tot},i+1}}, \tag{7.6}$$

where subscript $_d$ denotes "discrete."

If the Poincare matrix $\boldsymbol{A}_d = \boldsymbol{T_s}\boldsymbol{T_f}$ has eigenvalues of magnitude less than 1, the system is stable (Chapter 6); state errors are not amplified from step to step. Here we show that given the error reduction provided through $\boldsymbol{A_d}$, the bounded discrete disturbance in each step $|\boldsymbol{T_s}\boldsymbol{\Delta}_{f,i} + \boldsymbol{\Delta}_{s,i+1}| \leq |\boldsymbol{\Delta}_{\text{tot}}|^{\max}$ can only sum to some maximum error $|\boldsymbol{\xi}_{TO}|^{\max}$ on the infinite horizon, thereby defining an invariant and attractive region in the space of errors $\boldsymbol{\xi}_{TO}$ measured at takeoff. That is, based on the assumptions made in Section 7.2, it is guaranteed that $|\boldsymbol{\xi}_{TO}|$ shrinks in each step if it is larger than $|\boldsymbol{\xi}_{TO}|^{\max}$, and that once it is within this bound, it cannot leave. Note that

$\boldsymbol{A_d}$ has no dependency on the assumptions made in Section 7.2; it depends only on the nominal control design.

7.5.1 Tools from linear theory: norms, weighting matrices, and the discrete Lyapunov equation

Recall that the weighted L2 distance for a vector is defined by symmetric, positive-definite weighting matrix $\boldsymbol{W}$, with $|\boldsymbol{\xi}|_{\boldsymbol{W}}^2 = \boldsymbol{\xi}^T\boldsymbol{W}\boldsymbol{\xi}$ (throughout other sections in this chapter, the subscript $_{\boldsymbol{W}}$ is implicit when using the norm operator $|\ |$). Any symmetric, positive definite $\boldsymbol{W}$ has a symmetric, positive definite matrix square root $\boldsymbol{V} = \boldsymbol{W}^{\frac{1}{2}}$ such that $\boldsymbol{V}\boldsymbol{V} = \boldsymbol{W}$. Then, the distance function $|\boldsymbol{\xi}|_{\boldsymbol{W}} = \sqrt{\boldsymbol{\xi}^T\boldsymbol{W}\boldsymbol{\xi}}$ can be equivalently written as $|\boldsymbol{\xi}|_{\boldsymbol{W}} = |\boldsymbol{V}\boldsymbol{\xi}|_{\boldsymbol{I}}$.

The induced matrix norm $|\boldsymbol{A_d}|_{\boldsymbol{W}}$ (by definition) is the maximum of $\frac{|\boldsymbol{A_d}\boldsymbol{\xi}|_{\boldsymbol{W}}}{|\boldsymbol{\xi}|_{\boldsymbol{W}}}$ for any $\boldsymbol{\xi}$. It can be shown that for the weighted L2 vector norm with positive semi-definite $\boldsymbol{W}$, the induced matrix norm is given by the largest singular value of $\boldsymbol{V}\boldsymbol{A_d}\boldsymbol{V}^{-1}$. The largest singular value of a real square matrix $\boldsymbol{M}$ is the square-root of the largest eigenvalue of the product $\boldsymbol{M}^T\boldsymbol{M}$ (where the transpose is replaced with the complex conjugate for complex matrices $\boldsymbol{M}$).

Given that $\boldsymbol{A_d}$ has stable eigenvalues, there exists a unique, symmetric, positive definite solution $\boldsymbol{W}$ to the discrete matrix Lyapunov equation for any symmetric positive definite matrix $\boldsymbol{Q}$:

$$\boldsymbol{A_d}^T\boldsymbol{W}\boldsymbol{A_d} - \boldsymbol{W} = -\boldsymbol{Q}. \tag{7.7}$$

This theorem is proved in [15]. As a result, if we temporarily ignore the deviation Δ_{tot}, we can pick any positive definite $\boldsymbol{Q}$ to achieve

$$\begin{aligned}|\boldsymbol{\xi}_{TO,i+1}|^2 - |\boldsymbol{\xi}_{TO,i}|^2 &= \boldsymbol{\xi}_{TO,i}^T\boldsymbol{A_d}^T\boldsymbol{W}\boldsymbol{A_d}\boldsymbol{\xi}_{TO,i}^T - \boldsymbol{\xi}_{TO,i}^T\boldsymbol{W}\boldsymbol{\xi}_{TO,i}\\ &= -\boldsymbol{\xi}_{TO,i}^T\boldsymbol{Q}\boldsymbol{\xi}_{TO,i}.\end{aligned}$$

More intuitively, given a target convergence behavior described by arbitrary matrix $\boldsymbol{Q}$, we can solve the discrete Lyapunov function for the transformation $\boldsymbol{V} = \boldsymbol{W}^{\frac{1}{2}}$ and the associated weighting matrix $\boldsymbol{W}$ such that the unperturbed, re-mapped system converges with scaling $\boldsymbol{Q}$.

We can use this equation to generate weighting matrices $\boldsymbol{W}$ such that the induced matrix norm $|\boldsymbol{A_d}|_{\boldsymbol{W}}$ is less than one (which is not generally guaranteed by the stability property of eigenvalues of $\boldsymbol{A_d}$ having magnitudes less than one).

7.5.2 Derivation of the invariant and attractive region

Assuming the induced matrix norm $|\boldsymbol{A_d}| < 1$,

$$|\boldsymbol{\xi}_{TO}| \leq \frac{1}{1-|\boldsymbol{A_d}|}|\boldsymbol{\Delta}_{\text{tot}}|^{\max} = |\boldsymbol{\xi}_{TO}|^{\max} \tag{7.8}$$

describes an invariant and attractive region.

First we prove invariance: $|\boldsymbol{\xi}_{TO,i}| \leq |\boldsymbol{\xi}_{TO}|^{\max}$ implies

$$\begin{aligned}
|\boldsymbol{\xi}_{TO,i+1}| &\leq |\boldsymbol{A_d}\boldsymbol{\xi}_{TO,i}| + |\boldsymbol{\Delta}_{\text{tot}}|^{\max} && (7.9)\\
&\leq |\boldsymbol{A_d}|\left(\frac{1}{1-|\boldsymbol{A_d}|}|\boldsymbol{\Delta}_{\text{tot}}|^{\max}\right) + |\boldsymbol{\Delta}_{\text{tot}}|^{\max} && (7.10)\\
&= \left(\frac{|\boldsymbol{A_d}|}{1-|\boldsymbol{A_d}|} + \frac{1-|\boldsymbol{A_d}|}{1-|\boldsymbol{A_d}|}\right)|\boldsymbol{\Delta}_{\text{tot}}|^{\max} = |\boldsymbol{\xi}_{TO}|^{\max}. && (7.11)
\end{aligned}$$

Line 7.9 applies the triangle inequality to the recursive dynamics in Equation 7.6. Line 7.10 applies the definition of induced matrix norm and substitutes the definition from Equation 7.8. Line 3 is algebraic manipulation. Thus, we have shown that if step i started within this region, the next step $i+1$ will also stay within it.

Next we prove attractiveness, where $|\boldsymbol{\xi}_{TO,i}| \geq |\boldsymbol{\xi}_{TO}|^{\max}$ implies $|\boldsymbol{\xi}_{TO,i+1}| - |\boldsymbol{\xi}_{TO,i}| < 0$.

$$\begin{aligned}
|\boldsymbol{\xi}_{TO,i+1}| - |\boldsymbol{\xi}_{TO,i}| &\leq |\boldsymbol{A_d}\boldsymbol{\xi}_{TO,i}| + |\boldsymbol{\Delta}_{\text{tot}}|^{\max} - |\boldsymbol{\xi}_{TO,i}| && (7.12)\\
&\leq |\boldsymbol{A_d}||\boldsymbol{\xi}_{TO,i}| + |\boldsymbol{\Delta}_{\text{tot}}|^{\max} - |\boldsymbol{\xi}_{TO,i}| && (7.13)\\
&= (|\boldsymbol{A_d}| - 1)|\boldsymbol{\xi}_{TO,i}| + |\boldsymbol{\Delta}_{\text{tot}}|^{\max}. && (7.14)
\end{aligned}$$

Given $|\boldsymbol{A_d}| < 1$, the first term is negative. Therefore,

$$|\boldsymbol{\xi}_{TO,i+1}| - |\boldsymbol{\xi}_{TO,i}| \leq (|\boldsymbol{A_d}| - 1)\frac{1}{1-|\boldsymbol{A_d}|}|\boldsymbol{\Delta}_{\text{tot}}|^{\max} + |\boldsymbol{\Delta}_{\text{tot}}|^{\max} = 0. \tag{7.15}$$

Line 7.12 applies the triangle inequality to the recursive dynamics in Equation 7.6. Line 7.13 applies the definition of induced matrix norm, and line 7.14 is algebra. In line 7.15, the introduced fraction is less than one, thereby reducing the magnitude of a negative number. Thus, we have shown that if step i is not within the region, the next step $i+1$ will get closer to the region.

These two proofs hinge upon identifying a matrix norm for $|\boldsymbol{A}_d|_{\boldsymbol{W}}$ that is less than one (i.e, showing the unperturbed sequence of state errors to be contractive). Intuitively, using the cost-to-go from the optimal control applied in the control design or using the discrete matrix Lyapunov equation to construct a decreasing cost-to-go provides viable options for such a weighting matrix $\boldsymbol{W}$. In particular, solving the discrete Lyapunov equation (7.7) using $\boldsymbol{Q} = \boldsymbol{I}$ always guarantees $|\boldsymbol{A}_d|_{\boldsymbol{W}} < 1$:

$$|\boldsymbol{A}_d|_{\boldsymbol{W}}^2 = \text{max eig}\left((\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^{\boldsymbol{-1}})^T(\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^{\boldsymbol{-1}})\right) \tag{7.16}$$

$$= \text{max eig}\left(\boldsymbol{V}^{\boldsymbol{-1}}\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{V}\boldsymbol{V}\boldsymbol{A}\boldsymbol{V}^{-1}\right) \tag{7.17}$$

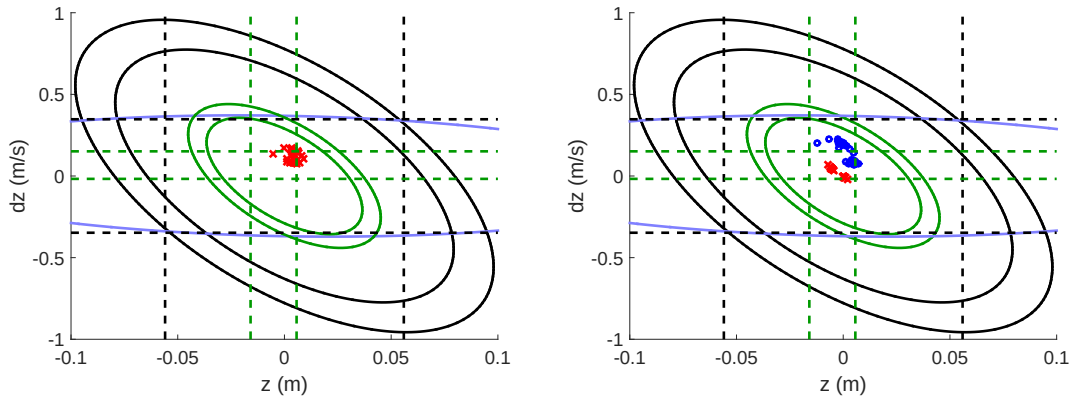
$$= \text{max eig}\left(\boldsymbol{V}^{\boldsymbol{-1}}(\boldsymbol{W}-\boldsymbol{Q})\boldsymbol{V}^{-1}\right) \tag{7.18}$$

$$= \text{max eig}\left(\boldsymbol{I}-\boldsymbol{W}^{-1}\right) < 1. \tag{7.19}$$

Line 7.16 comes from the definition of the induced matrix norm (using the matrix square root $\boldsymbol{V} = \sqrt{\boldsymbol{W}}$), and line 7.17 is algebra (using the symmetry of $\boldsymbol{V}$). Line 7.18 applies the discrete matrix Lyapunov equation, and line 7.19 distributes the multiplication and substitutes $\boldsymbol{Q} = \boldsymbol{I}$. The maximum eigenvalue is less than 1 because $\boldsymbol{W}$ is guaranteed positive definite.

It is known ([15]) that "better" choices of $\boldsymbol{Q}$ than the identity matrix $\boldsymbol{I}$ can provide "better" proofs of convergence, where the matrix norm induced by $\boldsymbol{W}$ is smaller. This results in a tighter attractive and invariant region. We hypothesize that using $\boldsymbol{W}$ from the LQR construction of the feedback control results in a contraction mapping closely related to the actual behavior of the system.

The proofs presented here are not new; they are fundamental tools from linear theory. Here, we applied them to derive quantified, theoretical performance metrics that are suitable for examining experimental results.



(a) drifts in each step

(b) expected errors and observed errors

Figure 7.7: Convergence of vertical errors at takeoff. See discussion in Section 7.5.3.

7.5.3 Example applied to vertical states over multiple steps

Here we the examine the experimental convergence the error of the vertical states $[z, \dot{z}]$ at takeoff, as this 2D system (decoupled from the horizontal and rotational states) can be easily visualized in 2D figures.

First, we derive bounds on the maximum total perturbation $\boldsymbol{\Delta}_{\text{tot}} = \boldsymbol{T_s}\boldsymbol{\Delta}_f + \boldsymbol{\Delta}_s(T_{TO} - T_{TD})$ acquired in one step. Numerical evaluation of the matrices in Equations 7.5 and 7.6 show that leg placement error ϵ_z has negligible effect given the closed loop dynamics described by T_s; $\boldsymbol{\Delta}_{\text{tot}}$ is dominated by the drift in stance. We then evaluate the stance drift in a variety of combinations.

See Figure 7.7. From Section 7.3.2, we can identify the maximum single-step drift along z and along $\dot{z}$ using rank one weighting matrices. The maximum value at takeoff is computed assuming both the tight force bounds (straight dashed lines in black) and the loose force bounds (straight dashed lines in green). In theory, these describe limits on the difference between takeoff errors $\boldsymbol{\xi}_{TO}$ and their predicted values $\boldsymbol{T_s}\boldsymbol{\xi}_{TD}$, which are plotted as the red data points in Figure 7.7A. This does not account for any estimator uncertainty $\boldsymbol{\sigma}$, which would widen each set of bounds.

We also compute $|\boldsymbol{\Delta}_s|^{\max}$ using the LQR cost from our control design (Section 5.4.1) and

from the guaranteed Lyapunov function construction using $\boldsymbol{Q} = \boldsymbol{I}$. The cost-to-go creates the inner black and green ellipses for loose and tight force bound assumptions, while the choice of $\boldsymbol{Q} = \boldsymbol{I}$ generates the light blue ellipse assuming loose force bounds. Like the straight lines described above, these boundaries correspond to the differences between observed and anticipated error, plotted as red points in the left figure A.

Over the course of multiple steps, the drifts can accrue to drive the errors away from the origin, while the nominal closed loop behavior pulls them back in. In figure B on the right, the expected errors are plotted in red, while the observed errors are plotted in blue (their difference yields the data in figure A). The straight black and green lines do not describe theoretical bounds on these measurements as they do not account for accumulated errors across steps. Rather, the matrix norm applied to the LQR bound $|\boldsymbol{A_d}| = 0.19$ grows the black and green ellipses by a factor of 123%, such that each outer ring gives the invariant and attractive region under loose and tight force assumptions – the red and blue data points in figure B are theoretically guaranteed to stay within these constraints. Using $\boldsymbol{Q} = \boldsymbol{I}$ to compute a contractive mapping yields $|\boldsymbol{A_d}| = 0.869$, which means the guaranteed region is an ellipse 763% the size of the one plotted. This guarantee is far more conservative and not practically relevant.

Overall, we see that these experimental vertical errors validate the theoretical expectations of our model based control design. The tracking is not perfect, but it is within the expectations based upon the assumptions made in Section 7.2.

This analysis visualizes the effects of the continuous and discrete feedback controllers. The discrete control can help reset the error between steps, thereby reducing the matrix norm responsible for how errors persist from step to step. If deadbeat control is available (i.e, for the forward velocity $\dot{x}$, the corresponding entries of the matrix $\boldsymbol{T_f}$ can be set to exactly 0 such that there is no growth of the inner ellipse to form the outer ellipse. However, even with perfect deadbeat planning in the discrete control, the performance of the continuous control still determines the size of the original inner ellipse. Specifically, deadbeat control does not imply zero state errors

in the presence of perturbations; it implies nothing is carried over to the next step. On the other hand, higher and higher performance of the continuous control can theoretically approach perfect tracking by collapsing the inner ellipse, even without requiring discrete control. However, we observed in Chapter 6 that the under-actuation limits what the continuous control can achieve on its own, and the continuous control greatly benefits from the stability that the discrete feedback can generate.

7.6 Summary and discussion

We started with assumed models that bound the estimation error and the force tracking error. In addition to these two sources of uncertainty, we account for the modeling errors introduced by the linearization of the centroidal dynamics and by the decoupling of the vertical subsystem from the horizontal and rotational subsystem, which are shown to be relatively minor. Linear theory allows the integration of these effects into a bounded region that describes the expected tracking performance, which can then be compared to noisy hardware data. On our experiments, this analysis reveals that improving the force tracking in the SEA control offers the most promise for improving the tracking performance.

Then, by modeling the leg placement error in addition to the estimation error, we propagate the deviations from the nominal error dynamics in the flight phase. Composing the two perturbed systems together produces a discrete affine model of the perturbed error dynamics from step to step, from which we derive the expected long term closed loop error evolution. This analysis results in a theoretical invariant and attractive region in terms of state error that can be compared to hardware results. Having a Lyapunov function in the form of "cost-to-go" from the control design facilitates the computation of an accurate and meaningful region.

As a result, the theoretical expectations of a control design are validated with two types of comparisons to the experimental data. The short term closed loop behavior is checked by comparing the propagation of an uncertain initial condition to the bounded drift integrated over

a finite horizon. The long term convergence is checked by comparing the guaranteed invariant region with the observed errors over multiple steps. The next step would be to perform the analyses described here for the x and θ states, where the current results have been limited to z.

Based on this analysis, a theoretically promising controller can be judged by its short term error propagation and by its invariant and attractive region. A potentially fruitful avenue of future work would be to apply these analyses to the new control strategies identified in Chapter 6 and validate the expected performance with hardware experiments.

This process begins to bridge the current gap between theoretical studies of model based control and their hardware implementations. We believe this a crucial step, and without quantified comparisons between the expected results and the observed results, there is only limited value in designing high performance control or formulating proofs of convergence. The conservative nature of describing worst case bounds limits the realism of this approach, and more refined formulations can be an interesting topic for future research.

Chapter 8

Conclusions

8.1 Big picture

On the whole, we come away with an clearer overall understanding of the basic underlying mechanics behind the control of legged systems. Upright systems have a tendency to tip over, which is an unstable mode whether the system is standing still or in motion. Fighting this effect is one task, while orienting the trunk is another. Under-actuation means they cannot be addressed independently in one stance phase, though systems with sufficient inertia can solve them together. Leg placement can allow intermittent stabilization of the tipping problem, and the continuous actuation can thus be dedicated to orienting the trunk. Raibert control and other heuristic methods leverage this concept but stop short of maximizing the performance of either component of the control. Instead, the spring mass model or other predictive models of the continuous dynamics can maximize performance of the discrete control to achieve deadbeat stability. Likewise, a model of the discrete dynamics allows for optimization of the continuous control. More generally, model-based optimal control does not require a separation of the rotational dynamics and the translational dynamics; there is simply the potential to use both sources of control for stabilizing the system.

8.2 Contributions

In this thesis, we make four key contributions.

In Chapter 3, we extend the deadbeat agility and robustness of the simplified model to include steering in 3D on unobserved, uneven terrain. The results show that the mapping between the inputs of leg placement and the outputs of steering angle and apex height are non-linear, as the deadbeat law significantly outperforms the optimized classical linear control in terms of tracking and allows for more aggressive turns.

We demonstrate a novel level of performance for 2D running in Chapter 5. By tracking the motion of the deadbeat stabilized spring mass model, we achieve high accuracy tracking of the desired running speed, even across one-step changes in the target speed. We also demonstrate robustness to steps up and down, but these experiments do not exhibit the tracking expected by the deadbeat simplified model, perhaps due to the large impact forces. This potential explanation should be examined using the techniques developed in Chapter 7.

In Chapter 6, we present a linear decomposition of the closed loop error dynamics, which lends insight to the control design process. In particular, we use it to formulate optimal control in a way that automatically takes advantage of the stability exhibited by the simplified model and accounts for any other discrete dynamics. This decomposition also produces the closed-loop dynamic coupling between state errors, which can be then used to modify the original leg placement to recover full-order deadbeat velocity or optimize for other performance measures.

In Chapter 7, we develop a formulation in which the theoretical tracking error is a quantified function of the perturbations to the nominal system. This allows us to validate the model-based control design with comparisons to experimental data, and to propose promising avenues of future work. Specifically, non-linearities in the centroidal dynamics are indeed negligible, deadbeat stability alone does not guarantee good tracking, and that our current implementation primarily suffers from errors in the closed loop actuator control. However, the conservativeness of using bounds leaves room for improvement with more accurate treatments of the effects of perturba-

tions and uncertainty.

8.3 Potential tasks for future research

The results from this thesis motivate interesting avenues for future research. The model-based techniques we use can be adapted to include more of the actuator dynamics in the full order model of the robot, such that the inputs are motor velocities and the equations of motion include the spring torques. Just as we mapped hip torques to Cartesian GRFs to avoid linearizing the geometry of the leg, the motor velocities may be mapped to rates of change of the GRFs. This modification would be a departure from the general study of spring mass motion within the centroidal dynamics of a system, but it may be an effective means of controlling ATRIAS, as our results suggest that the actuator dynamics may indeed be significant. It may be very interesting to compare the performance of such an "un-cascaded" control architecture, which would then clarify how appropriate it is to treat human-scale robots with geared electric drives as force-controlled systems. In turn, this would also offer insight to the relevance of second-order simplified models of legged locomotion on such robots.

We showed that model-based control can transfer the targeted behavior of the simplified model to the real robot, using deadbeat velocity control as an example behavior. The full capabilities of the spring mass system allow for additional behaviors. First, the time-based leg placement that lends feed-forward robustness to uneven terrain specifies a specific swing leg trajectory, where the current implementation designs the swing leg motion to minimize impacts with the ground. The trade-off between these two effects depends on the actuator dynamics, which may be explicitly modeled as described above. Secondly, the forces and control required for the highly agile 3D running described by the spring mass model seem to be outside of ATRIAS's capacities, but the implementation of this behavior may be an exciting project for a different robot. Lastly, similar to the leg length adjustment we identified in Chapter 6, other works have use modified models (Section 2.4.3) with additional parameters (for example, changing the spring

stiffnesses) that effectively increase the capabilities of the motion planning in the discrete control. An exploration of transferring these strategies to hardware using the techniques derived in this thesis could be interesting.

Bibliography

[1] Mojtaba Ahmadi and Martin Buehler. The arl monopod ii running robot: Control and energetics. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 1689–1694. IEEE, 1999. 2.4.1

[2] Mojtaba Ahmadi and Martin Buehler. Controlled passive dynamic running experiments with the arl-monopod ii. *Robotics, IEEE Transactions on*, 22(5):974–986, 2006. 2.4.1, 3.3

[3] Mazen Alamir. *A Pragmatic Story of Model Predictive Control: Self-Contained Algorithms and Case-Studies*. CreateSpace Independent Publishing Platform, 2013. 2.3.1

[4] Alin Albu-Schäffer, Christian Ott, and Gerd Hirzinger. A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *The International Journal of Robotics Research*, 26(1):23–39, 2007. 7.1

[5] R. Alexander. Three uses for springs in legged locomotion. *The International Journal of Robotics Research*, 9(2):53–61, 1990. 2.4.3, 3.1

[6] R Altendorfer, RM Ghigliazza, P Holmes, and DE Koditschek. Exploiting passive stability for hierarchical control. In *Fifth International Conference on Climbing and Walking Robots (CLAWAR)*, pages 177–184, 2002. 2.4.3, 3.1

[7] Richard Altendorfer, Ned Moore, Haldun Komsuoglu, Martin Buehler, H Benjamin Brown Jr, Dave McMordie, Uluc Saranli, R Full, and Daniel E Koditschek. Rhex: a biologically inspired hexapod runner. *Autonomous Robots*, 11(3):207–213, 2001. 2.3.2, 4.2.3, 6.5

[8] Richard Altendorfer, Daniel E Koditschek, and Philip Holmes. Towards a factored analysis of legged locomotion models. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 37–44. IEEE, 2003. 2.4.3, 3.1

[9] Brian DO Anderson and John B Moore. *Optimal control: linear quadratic methods*. Courier Corporation,

2007. 2.3.1, 5.1.2

[10] Emanuel Andrada, Christian Rode, and Reinhard Blickhan. Grounded running in quails: simulations indicate benefits of observed fixed aperture angle between legs before touch-down. *Journal of theoretical biology*, 335:97–107, 2013. 2.4.3

[11] Ben Andrews, Bruce Miller, John Schmitt, and Jonathan E Clark. Running over unknown rough terrain with a one-legged planar robot. *Bioinspiration & biomimetics*, 6(2):026009, 2011. 2.4.3, 3.1

[12] M Mert Ankarali and Uluç Saranli. Stride-to-stride energy regulation for robust self-stability of a torque-actuated dissipative spring-mass hopper. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(3): 033121, 2010. 3.2.3

[13] M Mert Ankaralı and Uluc Saranlı. Control of underactuated planar pronking through an embedded spring-mass hopper template. *Autonomous Robots*, 30(2):217–231, 2011. 2.3.2, 3.1, 4.2.3, 6.5

[14] Panos Antsaklis and Anthony N Michel. *Linear systems*. Springer Science & Business Media, 2006. 5.1.2, 6.2.2

[15] Panos J Antsaklis and Anthony N Michel. *A linear systems primer*, volume 1. Birkhäuser Boston, 2007. 2.5, 5.1.2, 7.5.1, 7.5.2

[16] Ömur Arslan, Uluc Saranli, and Ömer Morgül. Reactive footstep planning for a planar spring mass hopper. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 160–166. IEEE, 2009. 2.4.3, 3.1, 3.2.3

[17] Karl Johan Aström and Richard M Murray. *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010. 2.3.1, 7.1

[18] CG Atkeson, BPW Babu, N Banerjee, D Berenson, CP Bove, X Cui, M DeDonato, R Du, S Feng, P Franklin, et al. What happened at the darpa robotics challenge, and why? 2.5

[19] Morteza Azad. *Balancing and hopping motion control algorithms for an under-actuated robot*. PhD thesis, Ph. D. Thesis, The Australian National University, School of Engineering, 2014. 2.3, 2.4.1

[20] Morteza Azad and Roy Featherstone. Angular momentum based balance controller for an under-actuated planar robot. *Autonomous Robots*, 40(1):93–107, 2016. 2.3, 2.4.1

[21] Sylvain Bertrand. *A control strategy for high-speed running within unknown environments for a 3D bipedal robot*. PhD thesis, Versailles-St Quentin en Yvelines, 2013. 2.4.3

[22] Franco Blanchini. Feedback control for linear time-invariant systems with state and control bounds in the

presence of disturbances. *IEEE Transactions on automatic control*, 35(11):1231–1234, 1990. 2.5

[23] Franco Blanchini. Ultimate boundedness control for uncertain discrete-time systems via set-induced lyapunov functions. In *Decision and Control, 1991., Proceedings of the 30th IEEE Conference on*, pages 1755–1760. IEEE, 1991. 2.5

[24] R. Blickhan. The spring-mass model for running and hopping. *Journal of biomechanics*, 22(11-12):1217–1227, 1989. 2.4.3, 3.1

[25] Reinhard Blickhan and RJ Full. Similarity in multilegged locomotion: bouncing like a monopode. *Journal of Comparative Physiology A*, 173(5):509–517, 1993. 3.1

[26] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975. 2.3.1, 5.1.2

[27] 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. 2.4.3, 3.1, 3.2.1, 3.2.4

[28] Christine Chevallereau, ER Westervelt, and JW Grizzle. Asymptotically stable running for a five-link, four-actuator, planar bipedal robot. *The International Journal of Robotics Research*, 24(6):431–464, 2005. 4.2.3, 6.1.3

[29] Christine Chevallereau, Jessy W Grizzle, and Ching-Long Shih. Asymptotically stable walking of a five-link underactuated 3-d bipedal robot. *IEEE Transactions on Robotics*, 25(1):37–50, 2009. 2.3.1

[30] Baek-Kyu Cho, Sang-Sin Park, and Jun-ho Oh. Controllers for running in the humanoid robot, hubo. In *Humanoid Robots, 2009. Humanoids 2009. 9th IEEE-RAS International Conference on*, pages 385–390. IEEE, 2009. 2.2.1, 2.4.1

[31] Behnam Dadashzadeh, Hamid Reza Vejdani, and Jonathan Hurst. From template to anchor: A novel control strategy for spring-mass running of bipedal robots. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2566–2571. IEEE, 2014. 2.3.2, 4.2.3, 6.1.1, 6.5

[32] Hongkai Dai and Russ Tedrake. Optimizing robust limit cycles for legged locomotion on unknown terrain. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 1207–1213. IEEE, 2012. 2.3.1, 6.1.1, 6.1.2, 6.3.1

[33] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 295–302. IEEE, 2014. 2.2.2

[34] Monica A Daley, James R Usherwood, Gladys Felix, and Andrew A Biewener. Running over rough terrain: guinea fowl maintain dynamic stability despite a large unexpected change in substrate height. *Journal of Experimental Biology*, 209(1):171–187, 2006. 3.2.2

[35] Martin De Lasa and Aaron Hertzmann. Prioritized optimization for task-space control. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5755–5762. IEEE, 2009. 2.2.2

[36] Martin de Lasa, Igor Mordatch, and Aaron Hertzmann. Feature-based locomotion controllers. In *ACM Transactions on Graphics (TOG)*, volume 29, page 131. ACM, 2010. 2.2.2, 4.2.2

[37] Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale. Prioritized motion–force control of constrained fully-actuated robots:task space inverse dynamics. *Robotics and Autonomous Systems*, 63: 150–157, 2015. 2.2.2

[38] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. 6.3.1

[39] Ruta Desai, Hartmut Geyer, and Jessica K Hodgins. Virtual model control for dynamic lateral balance. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 856–861. IEEE, 2014. 2.2.2

[40] 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. 2.4.3, 3.1

[41] *Gold Line Panel Mounted Servo Drive Hardware Manual*. Elmo Motion Control Ltd., Sep. 2014. Ver. 1.005. 4.3.4

[42] *Elmo Application Studio II User Guide*. Elmo Motion Control Ltd., Feb. 2015. Ver. 2.300. 4.3.4

[43] Johannes Englsberger, Christian Ott, Maximo Roa, Alin Albu-Schäffer, Gerhard Hirzinger, et al. Bipedal walking control based on capture point dynamics. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4420–4427. IEEE, 2011. 2.4.2

[44] Johannes Englsberger, Alexander Werner, Christian Ott, Bernd Henze, Maximo A Roa, Gianluca Garofalo, Robert Burger, Alexander Beyer, Oliver Eiberger, Korbinian Schmid, et al. Overview of the torque-controlled humanoid robot toro. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 916–923. IEEE, 2014. 4.2.2

[45] Johannes Englsberger, Pawel Kozlowski, and Christian Ott. Biologically inspired dead-beat controller for bipedal running in 3d. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 989–996. IEEE, 2015. 2.4.4, 3.3

[46] M. Ernst, H. Geyer, and R. Blickhan. Spring-legged locomotion on uneven ground: A control approach to keep the running speed constant. In *International Conference on Climbing and Walking Robots (CLAWAR)*, pages 639–644, 2009. 4.1

[47] M. Ernst, H. Geyer, and R. Blickhan. Extension and customization of self-stability control in compliant legged systems. *Bioinspiration and Biomimetics*, 2012. 1.1, 1.2, 2.4.3, 2.4.3, 3.1, 3.2.1, 3.2.2

[48] Claire T Farley, James Glasheen, and Thomas A McMahon. Running springs: speed and animal size. *Journal of experimental Biology*, 185(1):71–86, 1993. 3.1

[49] Roy Featherstone. Quantitative measures of a robots physical ability to balance. *The International Journal of Robotics Research*, page 0278364916669599, 2016. 2.3

[50] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization based full body control for the atlas robot. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 120–127. IEEE, 2014. 2.2.2, 2.5

[51] Siyuan Feng, X Xinjilefu, Christopher G Atkeson, and Joohyung Kim. Optimization based controller design and implementation for the atlas robot in the darpa robotics challenge finals. *Humanoid Robots (Humanoids)*, 2015. 1.2, 2.2.2, 2.3.1, 2.5, 6.6.3

[52] 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. 2.3.2, 4.2.3, 6.5

[53] Gianluca Garofalo, Christian Ott, and Alin Albu-Schäffer. Walking control of fully actuated robots based on the bipedal slip model. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1456–1463. IEEE, 2012. 2.3.2, 4.2.3, 6.5

[54] H. Geyer, A. Seyfarth, and R. Blickhan. Spring-mass running: simple approximate solution and application to gait stability. *Journal of theoretical biology*, 232(3):315–328, 2005. 2.4.3, 3.1, 3.2.3

[55] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Positive force feedback in bouncing gaits? *Proceedings of the Royal Society of London B: Biological Sciences*, 270(1529):2173–2183, 2003. 3.2.3

[56] RM Ghigliazza, R. Altendorfer, P. Holmes, and D.E. Koditschek. A simply stabilized running model. 2003. 2.4.3, 3.1, 3.2.1

[57] J Glover and F Schweppe. Control of linear dynamic systems with set constrained disturbances. *IEEE Transactions on Automatic Control*, 16(5):411–423, 1971. 2.5

[58] Herbert Goldstein. *Classical mechanics*. Pearson Education India, 1965. 4.3.1

[59] Ambarish Goswami. Postural stability of biped robots and the foot-rotation indicator (fri) point. *The International Journal of Robotics Research*, 18(6):523–533, 1999. 1.1, 2.2.1

[60] Mohinder S Grewal. *Kalman filtering*. Springer, 2011. 5.1.2

[61] Jesse A Grimes and Jonathan W Hurst. The design of atrias 1.0 a unique monopod, hopping robot. In *Proceedings of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, volume 23, page 26. World Scientific, 2012. 1.1, 2.1.3, 4.1, 4.3.2

[62] K Akbari Hamed, N Sadati, WA Gruver, and GA Dumont. Exponential stabilisation of periodic orbits for running of a three-dimensional monopedal robot. *IET control theory & applications*, 5(11):1304–1320, 2011. 2.3.1

[63] Kaveh Akbari Hamed and Jessy W Grizzle. Robust event-based stabilization of periodic orbits for hybrid systems: Application to an underactuated 3d bipedal robot. In *American Control Conference (ACC), 2013*, pages 6206–6212. IEEE, 2013. 2.5, 4.2.2

[64] Kensuke Harada, Shuuji Kajita, Kenji Kaneko, and Hirohisa Hirukawa. An analytical method for real-time gait planning for humanoid robots. *International Journal of Humanoid Robotics*, 3(01):1–19, 2006. 2.4.2

[65] *Cup Type Component Sets & Housed Units*. Harmonic Drive, LLC, 2015. Rev. 7-14. 4.3.4

[66] Bernd Henze, Christian Ott, Maximo Roa, et al. Posture and balance control for humanoid robots in multi-contact scenarios based on model predictive control. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3253–3258. IEEE, 2014. 2.3.1

[67] Alexander Herzog, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal. Momentum-based balance control for torque-controlled humanoids. *environment*, 7(8):1, 2013. 2.2.2

[68] Alexander Herzog, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal. Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 981–988. IEEE, 2014. 1.2, 2.2.2

[69] Daan GE Hobbelen and Martijn Wisse. Swing-leg retraction for limit cycle walkers improves disturbance rejection. *Robotics, IEEE Transactions on*, 24(2):377–389, 2008. 3.2.2

[70] Jessica K Hodgins and Marc H Raibert. Adjusting step length for rough terrain locomotion. *Robotics and Automation, IEEE Transactions on*, 7(3):289–298, 1991. 2.4.1

[71] Andreas G Hofmann. *Robust execution of bipedal walking tasks from biomechanical principles*. PhD thesis, Massachusetts Institute of Technology, 2005. 2.2.2

[72] Philip Holmes, Robert J Full, Dan Koditschek, and John Guckenheimer. The dynamics of legged locomotion: Models, analyses, and challenges. *Siam Review*, 48(2):207–304, 2006. 2.4.3, 3.1

[73] Michael A Hopkins, Stephen A Ressler, Derek F Lahr, Alexander Leonessa, and Dennis W Hong. Embedded joint-space control of a series elastic humanoid. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3358–3365. IEEE, 2015. 4.2.2

[74] Tingshu Hu, Zongli Lin, and Ben M Chen. An analysis and design method for linear systems subject to actuator saturation and disturbance. *Automatica*, 38(2):351–359, 2002. 2.5

[75] Christian Hubicki, Andy Abate, Patrick Clary, Siavash Rezazadeh, Mikhail Jones, Andrew Peekema, Johnathan Van Why, Ryan Domres, Albert Wu, William Martin, et al. Walking and running with passive compliance. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, page 1, 2016. 5

[76] Marco Hutter, C David Remy, Mark A Hoepflinger, and Roland Siegwart. Efficient and versatile locomotion with highly compliant legs. *IEEE/ASME Transactions on Mechatronics*, 18(2):449–458, 2013. 4.2.2, 4.3.4

[77] Sang-Ho Hyon, Rieko Osu, and Yohei Otaka. Integration of multi-level postural balancing on humanoid robots. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1549–1556. IEEE, 2009. 2.2.2

[78] 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

[79] Mikhail S Jones. *Optimal control of an underactuated bipedal robot*. PhD thesis, 2014. 2.1.3, 2.3.1, 4.2.2, 4.4.1, 6.1.2, 6.5.2

[80] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kazuhito Yokoi, and Hirohisa Hirukawa. The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 239–246. IEEE, 2001. 2.4.2, 2.4.3, 3.2.2

[81] Shuuji Kajita, Osamu Matsumoto, and Muneharu Saigo. Real-time 3d walking pattern generation for a biped robot with telescopic legs. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pages 2299–2306. IEEE, 2001. 2.4.2

[82] Shuuji Kajita, Takashi Nagasaki, Kazuhito Yokoi, Kenji Kaneko, and Kazuo Tanie. Running pattern generation for a humanoid robot. In *ICRA*, pages 2755–2761, 2002. 2.4.1

[83] 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. 2.2.1, 2.3.1, 2.4.2

[84] Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, Kiyoshi Fujiwara, Kensuke Harada, Kazuhito Yokoi, and Hirohisa Hirukawa. Resolved momentum control: Humanoid motion planning based on the linear and angular momentum. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 2, pages 1644–1650. IEEE, 2003. 2.2.1

[85] Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, Kazuhito Yokoi, and Kazuo Tanie. A hop towards running humanoid biped. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 629–635. IEEE, 2004. 2.4.1, 2.5

[86] Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, Kazuhito Yokoi, and Kazuo Tanie. A running controller of humanoid biped hrp-2lr. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 616–622. IEEE, 2005. 2.2.1, 2.4.1, 2.5

[87] Shuuji Kajita, Takashi Nagasaki, Kenji Kaneko, and Hirohisa Hirukawa. Zmp-based biped running control. *IEEE robotics & automation magazine*, 2(14):63–72, 2007. 2.2.1, 2.4.1

[88] Shuuji Kajita, Mitsuharu Morisawa, Kanako Miura, Shin'ichiro Nakaoka, Kensuke Harada, Kenji Kaneko, Fumio Kanehiro, and Kazuhito Yokoi. Biped walking stabilization based on linear inverted pendulum tracking. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 4489–4496. IEEE, 2010. 2.4.2

[89] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 5.1.2

[90] Hassan K Khalil. *Noninear Systems*. Prentice-Hall, New Jersey, 1996. 2.5, 5.1.2, 6.4.1

[91] Pramod P Khargonekar, Ian R Petersen, and Kemin Zhou. Robust stabilization of uncertain linear systems: quadratic stabilizability and h/sup infinity/control theory. *IEEE Transactions on Automatic Control*, 35(3): 356–361, 1990. 7.1

[92] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53, 1987. 2.2.2

[93] Oussama Khatib, Luis Sentis, and Jae-Heung Park. A unified framework for whole-body humanoid robot control with multiple constraints and contacts. In *European Robotics Symposium 2008*, pages 303–312. Springer, 2008. 3.2.1

[94] Coleman Knabe, Bryce Lee, Viktor Orekhov, and Dennis Hong. Design of a compact, lightweight, electromechanical linear series elastic actuator. In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V05BT08A014–V05BT08A014. American Society of Mechanical Engineers, 2014. 5.4.3

[95] Daniel E Koditschek and Martin Bühler. Analysis of a simplified hopping robot. *The International Journal of Robotics Research*, 10(6):587–605, 1991. 3.2.1

[96] Devin Koepl and Jonathan Hurst. Force control for planar spring-mass running. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 3758–3763. IEEE, 2011. 3.2.3

[97] Devin Koepl, Kevin Kemper, and Jonathan Hurst. Force control for spring-mass walking and running. In *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*, pages 639–644. IEEE, 2010. 4.1

[98] Shishir Kolathaya, Ayonga Hereid, and Aaron D Ames. Time dependent control lyapunov functions and hybrid zero dynamics for stable robotic locomotion. In *American Control Conference (ACC), 2016*, pages 3916–3921. IEEE, 2016. 6.6.2

[99] Kyoungchul Kong, Joonbum Bae, and Masayoshi Tomizuka. Control of rotary series elastic actuator for ideal force-mode actuation in human–robot interaction applications. *IEEE/ASME transactions on mechatronics*, 14(1):105–118, 2009. 4.2.2

[100] Twan Koolen, Tomas De Boer, John Rebula, Ambarish Goswami, and Jerry Pratt. Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models. *The International Journal of Robotics Research*, 31(9):1094–1113, 2012. 2.4.2

[101] Twan Koolen, Sylvain Bertrand, Gray Thomas, Tomas de Boer, Tingfan Wu, Jesper Smith, Johannes Englsberger, and J Pratt. Design of a momentum-based control framework and application to the humanoid robot atlas. *preparation for International Journal of Humanoid Robotics*, 2015. 1.2, 2.2.2, 2.5

[102] Shunsuke Kudoh, Taku Komura, and Katsushi Ikeuchi. The dynamic postural adjustment with the quadratic programming method. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2563–2568. IEEE, 2002. 2.2.2, 4.2.2

[103] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An efficiently solvable quadratic program for stabilizing dynamic locomotion. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 2589–2594. IEEE, 2014. 2.2.2, 2.3.1

[104] 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*, pages 1–27, 2015. 1.2, 2.2.2, 2.3.1, 2.5, 6.1.2, 6.6.3

[105] Ryo Kurazume, Tsutomu Hasegawa, and Kan Yoneda. The sway compensation trajectory for a biped robot. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 1, pages 925–931. IEEE, 2003. 2.4.2

[106] John Leavitt, James E Bobrow, and Athanasios Sideris. Robust balance control of a one-legged, pneumatically-actuated, acrobot-like hopping robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 4240–4245. IEEE, 2004. 2.3

[107] Sung-Hee Lee and Ambarish Goswami. A momentum-based balance controller for humanoid robots on non-level and non-stationary ground. *Autonomous Robots*, 33(4):399–414, 2012. 2.2.2, 2.5

[108] Yiping Liu, Patrick M Wensing, David E Orin, and Yuan F Zheng. Trajectory generation for dynamic walking in a humanoid over uneven terrain using a 3d-actuated dual-slip model. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 374–380. IEEE, 2015. 2.5

[109] Ian R Manchester. Transverse dynamics and regions of stability for nonlinear hybrid limit cycles. *IFAC Proceedings Volumes*, 44(1):6285–6290, 2011. 6.6.2

[110] Ian R Manchester, Mark M Tobenkin, Michael Levashov, and Russ Tedrake. Regions of attraction for hybrid limit cycles of walking robots. *IFAC Proceedings Volumes*, 44(1):5801–5806, 2011. 6.3

[111] William C Martin, Albert Wu, and Hartmut Geyer. Robust spring mass model running for a physical bipedal robot. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015. 5

[112] William C Martin, Albert Wu, and Hartmut Geyer. Experimental evaluation of deadbeat running on the atrias biped. In *IEEE Robotics and Automation Letters (RA-L)*. IEEE, 2017. 5

[113] Harold Roberto Martinez Salazar and Juan Pablo Carbajal. Exploiting the passive dynamics of a compliant leg to develop gait transitions. *Phys. Rev. E*, 83:066707, Jun 2011. doi: 10.1103/PhysRevE.83.066707. URL `http://link.aps.org/doi/10.1103/PhysRevE.83.066707`. 2.4.3

[114] Tad McGeer. Passive dynamic walking. *the international journal of robotics research*, 9(2):62–82, 1990. 3.2.1

[115] Thomas A McMahon and George C Cheng. The mechanics of running: how does stiffness couple with speed? *Journal of biomechanics*, 23:65–78, 1990. 2.4.3, 3.1

[116] Michael Mistry, Jonas Buchli, and Stefan Schaal. Inverse dynamics control of floating base systems using orthogonal decomposition. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3406–3412. IEEE, 2010. 2.2.2

[117] 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. 2.4.4, 5.3.1

[118] Ömer Morgül, Uluc Saranli, et al. Experimental validation of a feed-forward predictor for the spring-loaded inverted pendulum template. *IEEE Transactions on robotics*, 31(1):208–216, 2015. 5.1.1

[119] B Morris and JW Grizzle. Hybrid invariance in bipedal robots with series compliant actuators. In *Decision and Control, 2006 45th IEEE Conference on*, pages 4793–4800. IEEE, 2006. 4.2.2

[120] Richard M Murray, Zexiang Li, S Shankar Sastry, and S Shankara Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994. 4.3.1, 4.3.2

[121] Ken'ichiro Nagasaka, Yoshihiiro Kuroki, Shinya Suzuki, Yoshihio Itoh, and Jinichi Yamaguchi. Integrated motion control for walking, jumping and running on a small bipedal entertainment robot. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3189–3194. IEEE, 2004. 2.2.1, 2.4.2, 2.5

[122] David E Orin and Ambarish Goswami. Centroidal momentum matrix of a humanoid robot: Structure and properties. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 653–659. IEEE, 2008. 2.2.2

[123] David E Orin, Ambarish Goswami, and Sung-Hee Lee. Centroidal dynamics of a humanoid robot. *Autonomous Robots*, 35(2-3):161–176, 2013. 2.2.2

[124] Christian Ott, Alin Albu-Schaffer, Andreas Kugi, and Gerd Hirzinger. On the passivity-based impedance control of flexible joint robots. *Robotics, IEEE Transactions on*, 24(2):416–429, 2008. 5.1.2

[125] Christian Ott, Maximo Roa, Gerd Hirzinger, et al. Posture and balance control for biped robots based on contact force optimization. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 26–33. IEEE, 2011. 1.2, 2.2.2

[126] Nicholas Paine, Sehoon Oh, and Luis Sentis. Design and control considerations for high-performance series elastic actuators. *IEEE/ASME Transactions on Mechatronics*, 19(3):1080–1091, 2014. 4.2.2

[127] Nicholas Paine, Joshua S Mehling, James Holley, Nicolaus A Radford, Gwendolyn Johnson, Chien-Liang Fok, and Luis Sentis. Actuator control for the nasa-jsc valkyrie humanoid robot: A decoupled dynamics

approach for torque control of series elastic robots. *Journal of Field Robotics*, 32(3):378–396, 2015. 4.2.2

[128] Federico Parietti and Hartmut Geyer. Reactive balance control in walking based on a bipedal linear inverted pendulum model. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5442–5447. IEEE, 2011. 2.4.2

[129] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. Mechanical design of humanoid robot platform khr-3 (kaist humanoid robot 3: Hubo). In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, pages 321–326. IEEE, 2005. 2.2.1

[130] Jaeheung Park and Oussama Khatib. Contact consistent control framework for humanoid robots. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1963–1969. IEEE, 2006. 2.2.2

[131] Juyong Park, Jaeyoung Haan, and Frank C Park. Convex optimization algorithms for active balancing of humanoid robots. *Robotics, IEEE Transactions on*, 23(4):817–822, 2007. 2.2.2

[132] Andrew T Peekema. Template-based control of the bipedal robot atrias. 2015. 2.3.2, 4.2.3, 6.5

[133] F. Peuker and A. Seyfarth. Adjusting legs for stable running in three dimensions. In C. T. Lim, J. C. H. Goh, and Ratko Magjarevic, editors, *6th World Congress of Biomechanics (WCB 2010). August 1-6, 2010 Singapore*, volume 31 of *IFMBE Proceedings*, pages 3–6. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-14515-5. 3.2.1

[134] Frank Peuker, Christophe Maufroy, and André Seyfarth. Leg-adjustment strategies for stable running in three dimensions. *Bioinspiration & biomimetics*, 7(3):036002, 2012. 2.4.3, 3.1, 3.2.4

[135] Ioannis Poulakakis and Jessy W Grizzle. The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper. *IEEE Transactions on Automatic Control*, 54(8):1779–1793, 2009. 1.2, 1.2, 2.4.3, 4.2.3, 6.1.3

[136] Ioannis Poulakakis and JW Grizzle. Monopedal running control: Slip embedding and virtual constraint controllers. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 323–330. IEEE, 2007. 2.4.3, 4.2.3, 6.1.1, 6.1.3

[137] Gill Pratt, Matthew M Williamson, et al. Series elastic actuators. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 399–406. IEEE, 1995. 4.1, 4.2.2, 5.1.2

[138] Gill A Pratt, Pace Willisson, Clive Bolton, and Andreas Hofman. Late motor processing in low-impedance robots: Impedance control of series-elastic actuators. In *American Control Conference, 2004. Proceedings*

of the 2004, volume 4, pages 3245–3251. IEEE, 2004. 7.1

[139] Jerry Pratt and Ben Krupp. Design of a bipedal walking robot. In *SPIE defense and security symposium*, pages 69621F–69621F. International Society for Optics and Photonics, 2008. 4.2.2

[140] Jerry Pratt and Gill Pratt. Intuitive control of a planar bipedal walking robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 3, pages 2014–2021. IEEE, 1998. 2.4.1, 4.2.2

[141] 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. 2.2.2

[142] Jerry Pratt, Ben Krupp, and Chris Morse. Series elastic actuators for high fidelity force control. *Industrial Robot: An International Journal*, 29(3):234–241, 2002. 4.2.2

[143] 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. 2.4.2, 3.2.2

[144] Jerry Pratt, Twan Koolen, Tomas De Boer, John Rebula, Sebastien Cotton, John Carff, Matthew Johnson, and Peter Neuhaus. Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower body humanoid. *The International Journal of Robotics Research*, page 0278364912452762, 2012. 2.4.2

[145] Jerry E Pratt and Benjamin T Krupp. Series elastic actuators for legged robots. In *Defense and Security*, pages 135–144. International Society for Optics and Photonics, 2004. 4.2.2

[146] Marc H Raibert. *Legged robots that balance*. MIT press, 1986. 2.3.2, 2.4.1, 3, 3.1, 3.2.3, 3.2.3, 3.2.3, 3.3, 5

[147] Marc H Raibert, H Benjamin Brown, and Michael Chepponis. Experiments in balance with a 3d one-legged hopping machine. *The International Journal of Robotics Research*, 3(2):75–92, 1984. 2.3.2, 2.4.1

[148] Marc H Raibert, H Benjamin Brown Jr, Michael Chepponis, Jeff Koechling, Jessica K Hodgins, Diane Dustman, W Kevin Brennan, David S Barrett, Clay M Thompson, John Daniell Hebert, et al. Dynamically stable legged locomotion (september 1985-septembers1989). 1989. 3.1

[149] Alireza Ramezani and JW Grizzle. Atrias 2.0, a new 3d bipedal robotic walker and runner. In *Proceedings of the 2012 International Conference on Climbing and walking Robots and the Support Technologies for Mobile Machines*, pages 467–474, 2012. 1.1, 2.1.3, 4.1, 4.3.2

[150] James B Rawlings and Kenneth R Muske. The stability of constrained receding horizon control. *IEEE transactions on automatic control*, 38(10):1512–1516, 1993. 6.1.2

[151] 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 *the ASME Dynamic Systems and Control Conference (ASME/DSCC), to appear*, 2015. 2.4.1

[152] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Inverse dynamics with optimal distribution of ground reaction forces for legged robots. In *Emerging Trends in Mobile Robotics-Proceedings of the 13th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 580–587. World Scientific, 2010. 2.2.2

[153] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Control of legged robots with optimal distribution of contact forces. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 318–324. IEEE, 2011. 2.2.2

[154] Ludovic Righetti, Jonas Buchli, Michael Mistry, and Stefan Schaal. Inverse dynamics control of floating-base robots with external constraints: A unified view. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1085–1090. IEEE, 2011. 2.2.2, 4.3.1

[155] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3): 280–298, 2013. 2.2.2

[156] Joseph Salini, Vincent Padois, and Philippe Bidaud. Synthesis of complex humanoid whole-body behavior: a focus on sequencing and tasks transitions. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1283–1290. IEEE, 2011. 2.2.2, 4.2.2

[157] 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. 2.4.3

[158] Uluc Saranlı, Ömür Arslan, M Mert Ankaralı, and Ömer Morgül. Approximate analytic solutions to non-symmetric stance trajectories of the passive spring-loaded inverted pendulum with damping. *Nonlinear Dynamics*, 62(4):729–742, 2010. 2.4.3, 3.1

[159] S Shankar Sastry. *Nonlinear systems: analysis, stability, and control*, volume 10. Springer Science & Business Media, 2013. 6.4.1

[160] Alexander Schepelmann, Michael D Taylor, and Hartmut Geyer. Development of a testbed for robotic neuromuscular controllers. *Robotics: Science and Systems VIII*, 2012. 4.2.2, 5.1.2, 5.4.3

[161] J Schmitt and J Clark. Modeling posture-dependent leg actuation in sagittal plane locomotion. *Bioinspiration & biomimetics*, 4(4):046005, 2009. 3.1, 3.2.3

[162] John Schmitt and Philip Holmes. Mechanical models for insect locomotion: dynamics and stability in the horizontal plane i. theory. *Biological cybernetics*, 83(6):501–515, 2000. 3.1

[163] John Schmitt and Philip Holmes. Mechanical models for insect locomotion: dynamics and stability in the horizontal plane–ii. application. *Biological cybernetics*, 83(6):517–527, 2000. 3.1

[164] John Schmitt, Mariano Garcia, RC Razo, Philip Holmes, and Robert J Full. Dynamics and stability of legged locomotion in the horizontal plane: a test case using insects. *Biological cybernetics*, 86(5):343–353, 2002. 3.1

[165] William J Schwind and Daniel E Koditschek. Approximating the stance map of a 2-dof monoped runner. *Journal of Nonlinear Science*, 10(5):533–568, 2000. 2.4.3, 3.1

[166] J.E. Seipel and P. Holmes. Running in three dimensions: Analysis of a point-mass sprung-leg model. *The International Journal of Robotics Research*, 24(8):657–674, 2005. 2.4.3, 3.1, 3.2.1, 3.2.4, 6.1.1

[167] Luis Sentis and Oussama Khatib. Control of free-floating humanoid robots through task prioritization. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 1718–1723. IEEE, 2005. 2.2.2

[168] Luis Sentis, Jaeheung Park, and Oussama Khatib. Compliant control of multicontact and center-of-mass behaviors in humanoid robots. *Robotics, IEEE Transactions on*, 26(3):483–501, 2010. 2.2.2

[169] A. Seyfarth and H. Geyer. Natural control of spring-like runningoptimized self-stabilization. In *Proceedings of the Fifth International Conference on Climbing and Walking Robots. Professional Engineering Publishing Limited*, pages 81–85, 2002. 1.1, 1.2, 2.4.3, 3.1, 3.2.2, 3.2.2, 3.2.4

[170] A. Seyfarth, H. Geyer, M. Günther, and R. Blickhan. A movement criterion for running. *Journal of Biomechanics*, 35(5):649–655, 2002. 2.4.3, 2.4.3, 3.1, 3.2.1, 3.2.1

[171] 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. 1.1, 2.4.3, 2.4.3, 3.1, 3.2.2, 3.2.4

[172] Natan Shemer and Amir Degani. Analytical control parameters of the swing leg retraction method using an instantaneous slip model. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International*

Conference on, pages 4065–4070. IEEE, 2014. 5.1.1

[173] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer Science & Business Media, 2008. 2.1.1, 5.1.2

[174] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010. 5.1.2

[175] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006. 7.1

[176] M Slovich, N Paine, K Kemper, A Metger, A Edinger, J Weber, and L Sentis. Building hume: A bipedal robot for human-centered hyper-agility. In *Dynamic Walking Meeting*, 2012. 4.2.2

[177] Mark W Spong. The swing up control problem for the acrobot. *IEEE control systems*, 15(1):49–55, 1995. 2.3

[178] Mark W. Spong. *Underactuated mechanical systems*, pages 135–150. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-40913-7. doi: 10.1007/BFb0015081. URL `http://dx.doi.org/10.1007/BFb0015081`. 2.1.3

[179] Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot modeling and control*, volume 3. wiley New York, 2006. 2.2.2

[180] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *The International Journal of Robotics Research*, 30(9):1170–1193, 2011. 4.1

[181] Koushil Sreenath, Hae-Won Park, and JW Grizzle. Design and experimental implementation of a compliant hybrid zero dynamics controller with active force control for running on mabel. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 51–56. IEEE, 2012. 2.4.1

[182] 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. 1.2, 1.2, 2.3.1, 2.4.1, 2.5, 4.2.2, 4.2.3, 4.4.1, 5, 5.4.2, 6.1.1, 6.1.3

[183] Benjamin Stephens. *Push recovery control for force-controlled humanoid robots*. PhD thesis, Carnegie Mellon University Pittsburgh, Pennsylvania USA, 2011. 2.3.1, 2.4.2

[184] Benjamin J Stephens and Christopher G Atkeson. Dynamic balance force control for compliant humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1248–

1255. IEEE, 2010. 2.2.2

[185] Steven Strogatz, Mark Friedman, A John Mallinckrodt, Susan McKay, et al. Nonlinear dynamics and chaos: With applications to physics, biology, chemistry, and engineering. *Computers in Physics*, 8(5):532–532, 1994. 1.2, 2.3.2, 2.4.3, 3.2.1, 6.1.1

[186] Tomomichi Sugihara and Yoshihiko Nakamura. A fast online gait planning with boundary condition relaxation for humanoid robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 305–310. IEEE, 2005. 2.4.2

[187] Ryosuke Tajima, Daisaku Honda, and Keisuke Suga. Fast running experiments involving a humanoid robot. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1571–1576. IEEE, 2009. 2.2.1, 2.3.1, 2.4.1, 2.4.2, 2.4.3

[188] Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1084–1091. IEEE, 2009. 2.2.1, 2.4.1, 2.4.2

[189] Toru Takenaka, Takashi Matsumoto, and Takahide Yoshiike. Real time motion generation and control for biped robot-3 rd report: Dynamics error compensation. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1594–1600. IEEE, 2009. 2.2.1, 2.4.1, 2.4.2

[190] Toru Takenaka, Takashi Matsumoto, Takahide Yoshiike, Tadaaki Hasegawa, Shinya Shirokura, Hiroyuki Kaneko, and Atsuo Orita. Real time motion generation and control for biped robot-4 th report: Integrated balance control. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1601–1608. IEEE, 2009. 2.2.1, 2.4.1, 2.4.2

[191] 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. 2.2.1, 2.4.1, 2.4.2

[192] Michael David Taylor. *A compact series elastic actuator for bipedal robots with human-like dynamic performance*. PhD thesis, Citeseer, 2011. 4.2.2

[193] Russ Tedrake. Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for mit 6.832). downloaded on 1/20/2017 from http://underactuated.mit.edu/. 2.1.3, 2.3

[194] Russell L Tedrake. *Applied optimal control for dynamically stable legged locomotion*. PhD thesis, Massachusetts Institute of Technology, 2004. 2.3.1, 6.1.1

[195] Nitish Thatte and Hartmut Geyer. Toward balance recovery with leg prostheses using neuromuscular model

control. *IEEE Transactions on Biomedical Engineering*, 63(5):904–913, 2016. 2.5

[196] Miles A. Townsend. Biped gait stabilization via foot placement. *Journal of Biomechanics*, 18(1):21 – 38, 1985. ISSN 0021-9290. doi: 10.1016/0021-9290(85)90042-9. URL `http://www.sciencedirect.com/science/article/pii/0021929085900429`. 3.2.2

[197] Nikos G Tsagarakis, Stephen Morfey, Gustavo Medrano Cerda, Li Zhibin, and Darwin G Caldwell. Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 673–678. IEEE, 2013. 4.2.2

[198] Heike Vallery, Ralf Ekkelenkamp, Herman Van Der Kooij, and Martin Buss. Passive and accurate torque control of series elastic actuators. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3534–3538. IEEE, 2007. 4.2.2, 4.3.4, 5.1.2, 7.1

[199] Hamid Reza Vejdani, Albert Wu, Hartmut Geyer, and Jonathan W Hurst. Touch-down angle control for spring-mass walking. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 5101–5106. IEEE, 2015. 2.4.3

[200] Hamid Reza Vejdani Noghreiyan. Control of spring-mass running robots. 2013. 2.3.2, 4.2.3, 6.5

[201] Miomir Vukobratović and Branislav Borovac. Zero-moment pointthirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157–173, 2004. 2.2.1

[202] Greg Welch and Gary Bishop. An introduction to the kalman filter. 1995. 5.1.2

[203] Patrick M Wensing and David Orin. High-speed humanoid running through control with a 3d-slip model. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5134–5140. IEEE, 2013. 2.2.2, 2.4.3, 2.5, 3.3

[204] Patrick M Wensing and David Orin. Generation of dynamic humanoid behaviors through task-space control with conic optimization. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3103–3109. IEEE, 2013. 2.2.2, 4.2.2

[205] Patrick M Wensing and David Orin. 3d-slip steering for high-speed humanoid turns. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4008–4013. IEEE, 2014. 2.4.3, 3.3

[206] Eric R Westervelt, Jessy W Grizzle, and Daniel E Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE transactions on automatic control*, 48(1):42–56, 2003. 1.2, 1.2, 4.2.3, 6.1.3

[207] Eric R Westervelt, Gabriel Buche, and Jessy W Grizzle. Experimental validation of a framework for the

design of controllers that induce stable walking in planar bipeds. *The International Journal of Robotics Research*, 23(6):559–582, 2004. 2.3.1

[208] Georg Wiedebach, Sylvain Bertrand, Tingfan Wu, Luca Fiorio, Stephen McCrory, Robert Griffin, Francesco Nori, and Jerry Pratt. Walking on partial footholds including line contacts with the humanoid robot atlas. In *Humanoid Robots (Humanoids), 2016 IEEE-RAS 16th International Conference on*, pages 1312–1319. IEEE, 2016. 2.2.2

[209] David A Winter. Human balance and posture control during standing and walking. *Gait & posture*, 3(4): 193–214, 1995. 2.4.2, 2.4.3

[210] 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 Trans on Robotics, submitted*, 2013. 3

[211] Albert Wu and Hartmut Geyer. Highly robust running of articulated bipeds in unobserved terrain. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2558–2565. IEEE, 2014. 4.2.3, 5

[212] Gordon Wyeth. Control issues for velocity sourced series elastic actuators. In *Proceedings of the Australasian Conference on Robotics and Automation 2006*. Australian Robotics and Automation Association Inc, 2006. 4.2.2, 4.3.4, 5.1.2, 5.4.3

[213] Gordon Wyeth. Demonstrating the safety and performance of a velocity sourced series elastic actuator. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 3642–3647. IEEE, 2008. 4.2.2, 4.3.4, 5.1.2, 5.4.3

[214] 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. 2.4.1

[215] Garth Zeglin and Ben Brown. Control of a bow leg hopping robot. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 1, pages 793–798. IEEE, 1998. 3.1, 3.2.3, 3.2.3, 5.1.1