# Toward Enhancing Stability and Agility of Dynamically Balancing Legged Robots: A Data-Driven Approach

by

Ali Zamani
B.S., K. N. Toosi University of Technology, 2008
M.S., K. N. Toosi University of Technology, 2011

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:
Pranav Bhounsule, Chair and Advisor
Milos Zefran
James Patton
Myunghee Kim
Heejin Jeong

This thesis is dedicated to my beloved parents, brother and sister for their endless love, support, and encouragement.

## ACKNOWLEDGMENT

I would like to express my gratitude to all the individuals that provided their encouragement, advice, and support over the years. If not for their efforts, my work would not have been possible. First and foremost, I would like to thank my advisor Prof. Pranav Bhounsule for being my mentor and guide throughout this process. Whenever challenges arose, his guidance and support were pivotal to my success in overcoming obstacles and developing new ideas.

I also offer my sincerest thanks to my other committee members, Prof. Milos Zefran, Prof. James Patton, Prof. Myunghee Kim, and Prof. Heejin Jeong for dedicating their time and accepting my offer to be on the committee. Im deeply grateful for their insight and constructive feedback.

Prof. Ahmad Taha at the University of Texas at San Antonio also provided invaluable advice and guidance, I appreciate the time he took to assist me. I would also like to thank my former labmates Eric, Robert, Christopher, Christian, Abhishek, Steven, and Joseph, and my current labmates Jeremy and Ernesto from the Robotics and Motion Lab for the great discussions and friendships we created. And to my good friends Arman, Nafiseh, and Brandi, I truly appreciate your support and encouragement over the past few years.

And of course, a tremendous thanks to the people who got me where I am today, my family. I could not have gotten here without them. Thank you all for the unending support along the course of this journey.

## CONTRIBUTIONS OF AUTHORS

**Chapter 1**: Ali Zamani wrote this chapter and Pranav Bhounsule advised and edited.

**Chapter 2**: A significant portion of this chapter is reproduced from a published paper with the following citation: [Bhounsule, P. A. and Zamani, A.: A discrete control lyapunov function for exponential orbital stabilization of the simplest walker. Journal of Mechanisms and Robotics, 9(5):051011, 2017.] in which Pranav Bhounsule and Ali Zamani equally contributed to conceiving the idea, carrying out the simulations, and writing the manuscript.

**Chapter 3**: A significant portion of this chapter is reproduced from a published paper with the following citation: [Zamani, A. and Bhounsule, P. A.: Foot placement and ankle push-off control for the orbital stabilization of bipedal robots. In 2017 IEEE International Conference on Intelligent Robots and Systems (IROS), 2017.] Ali Zamani and Pranav Bhounsule conceived the idea; Ali Zamani carried out the simulations and wrote the manuscript; and Pranav Bhounsule advised and edited.

**Chapter 4**: A significant portion of this chapter is reproduced from two published papers with the following citations: [Zamani, A. and Bhounsule, P. A. Control synergies for rapid sta- bilization and enlarged region of attraction for a model of hopping. Biomimetics, 3(3):25, 2018.] in which Ali Zamani and Pranav Bhounsule equally contributed to conceiving the idea, carrying out the simulations, and writing the manuscript; and [Bhounsule, P. A., Zamani, A., Krause, J., Farra, S., and Pusey, J.: Control policies for a large region of attraction for dynamically balancing legged robots: a sampling- based approach. Robotica, pages 116, 2020.)]

## CONTRIBUTIONS OF AUTHORS (Continued)

in which Pranav Bhounsule, Ali Zamani, and Jeremy Krause conceived the idea and wrote the manuscript; Pranav Bhounsule, Ali Zamani, Jeremy Krause, and Steven Farra carried out the simulations; and Jason Pusey edited the manuscript.

**Chapter 5**: A significant portion of this chapter is reproduced from a published paper with the following citation: [Bhounsule, P., Zamani, A., and Pusey, J.: Switching between limit cycles in a model of running using exponentially stabilizing discrete control lyapunov function. In American Control Conference (ACC), 2018.] in which Pranav Bhounsule and Ali Zamani conceived the idea; carried out the simulations; and wrote the manuscript. Jason Pusey edited the manuscript.

**Chapter 6**: A significant portion of this chapter is reproduced from a published paper with the following citation: [Zamani, A., Galloway, J. D., and Bhounsule, P. A.: Feedback motion planning of legged robots by composing orbital Lyapunov functions using rapidly exploring random trees. In 2019 International Conference on Robotics and Automation (ICRA), 2019] in which Ali Zamani and Pranav Bhounsule conceived the idea; and Ali Zamani, Joseph D. Galloway, Pranav Bhounsule carried out the simulations and wrote the manuscript.

**Chapter 7**: A significant portion of this chapter is reproduced from an accepted manuscript with the following citation: [Zamani, A. and Bhounsule, P. A.: Nonlinear model predictive control of hopping model using approximate step-to-step models for navigation on complex terrain. accepted to IEEE International Conference on Intelligent Robots and Systems (IROS), 2020.] in which Ali Zamani and Pranav Bhounsule conceived the idea; Ali Zamani carried out the simulations; Ali Zamani and Pranav Bhounsule wrote the manuscript.

## CONTRIBUTIONS OF AUTHORS (Continued)

**Chapter 8**: Ali Zamani wrote this chapter and Pranav Bhounsule advised and edited.

## TABLE OF CONTENTS

## TABLE OF CONTENTS (Continued)

**CHAPTER** **PAGE**

## TABLE OF CONTENTS (Continued)

**CHAPTER** **PAGE**

## TABLE OF CONTENTS (Continued)

**CHAPTER** **PAGE**

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF FIGURES (Continued)

## LIST OF FIGURES (Continued)

## LIST OF FIGURES (Continued)

## LIST OF FIGURES (Continued)

## LIST OF FIGURES (Continued)

**FIGURE** **PAGE**

## ABSTRACT

For legged robots, to be practical, they need to be stable and agile. Stability is the ability to withstand perturbations, and agility is the ability to rapidly accelerate. Both this metrics are challenging to meet for dynamically balancing legged robots which are machines that have point feet or small feet and hence are statically unstable but need to constantly move to achieve dynamical stability. The most well-known method of stabilizing such robots is to generate a periodic or steady motion, also known as a limit cycle and then stabilizing the limit cycle. However, current techniques only provide small perturbation stability for the limit cycle limiting its usefulness. To generate agile gaits, non-steady motion needs to be planned, but is computationally challenging. Although steady state gaits are computationally simple to evaluate, such gaits are not agile. This thesis provides a framework for large perturbation stability of the limit cycle and composition of limit cycles to achieve agile locomotion. The framework is based on assuming a candidate Lyapunov function for step-to-step or orbital stability of the limit cycle. Then a data-driven approach is used to find a nonlinear control policy to stabilize the limit cycle and to estimate the set of initial states that can be stabilized also known as the region of attraction. These regions of attraction are composed using heuristics and sampling-based techniques to achieve agile motion. Finally, the thesis explores a computational approach for real-time planning on complex terrains such as stepping stones. The approach consists of using data-driven techniques to generate and model the step-to-step dynamics around the limit cycle. The resulting model is simple enough to enable real-time nonlinear optimization. Demonstra-

tion of these approaches is done using the spring-loaded inverted pendulum model of hopping and compass gait model of walking.

# CHAPTER 1

# INTRODUCTION

## 1.1 Motivation

While dynamically balancing legged systems, which are required to travel continuously to stay balanced, have been considered as a replacement for traditional vehicles to operate on difficult terrain, their functionally has been far from satisfactory. Because stability is key to the success of such systems, it is typical to exploit existing techniques in formal control theory to design controllers. Although the research community has made substantial progress in developing techniques for designing controllers with stability guarantees for smooth systems such as aerial, underwater, and wheeled systems, it is difficult to apply such techniques to dynamically balancing legged robots due to several sources of complexity including but not limited to limb coordination (the robot's degree of the freedom is higher than that of the task), hybrid dynamics (continuous dynamics punctuated by discontinuous dynamics during support transfer), underactuation (more degrees of freedom than actuators), and inherent unstable modes (projectile motion for running and inverted pendulum for walking). Therefore, there is a growing need for developing state-of-the-art stability techniques for dynamically balancing legged robots. The continued existence of such need restricts the performance of legged robots. Once the stability issue is successfully tackled, it is possible to effectively handle high-level

behaviors such as motion and task planning. Fulfilling this need will allow legged robots to serve in many roles such as assisting first responders, carrying out search and rescue missions, or as industrial workers.

The vast majority of researches have looked at planning of locomotion that repeats itself after every step, also known as periodic or steady locomotion. Though the steady locomotion is computationally simpler to evaluate, it is not agile since there is no change in speeds and/or direction between steps. Extending techniques for synthesizing stable steady gaits to include non-steady or agile gaits is computationally challenging because it increases the dimensionality of the search space by an order for each added degree of freedom and the need for stabilization of each non-steady gait adds to the computational burden. The goal of this thesis is to create efficient computational techniques for motion control and balance and then extend these techniques to include motion planning. We use best practices from two fields—machine learning and formal control theory—in a unique way to create better control algorithms for legged systems.

## 1.2 Contributions

The contributions of this thesis to the stability and agility of dynamically balancing legged robots can be summarized as follows.

### 1.2.1 Designing Exponentially Stabilizing Controllers for Steady State Gaits

The first contribution of this thesis is to design exponentially stabilizing controllers for steady state gaits. The role of the stabilizing controller is to bring a set of initial conditions back to the steady state gait. This set is called the region of attraction (ROA). The methodology

is to use an orbital control Lyapunov function (OCLF) to find a combination of controls that would yield a large region of attraction. We define the OCLF as a quadratic function with the difference between the actual and nominal states as its arguments. This methodology is original because previous methods are based on a linear controller which limits the size of the ROA, but we rather use nonlinear programming to find controls which maximize the ROA. We then generate a compact representation of each control as a function of the states. We make no assumption on any specific relation, but use data-driven techniques to find the best relation between control and state that enlarges the ROA. This contribution is addressed in chapters 2–4.

### 1.2.2 Developing Efficient Techniques for Creating Agile Gaits Using Controllers for Steady State Gaits

The second contribution of this thesis is to present a computationally efficient framework for synthesizing agile gaits. Agility includes non-steady motion specified by the speed and/or direction change over a step. A simple way to synthesize agile gaits is to construct controllers for every possible change, but this method is computationally expensive. There is a necessity for developing a computationally efficient technique to switch between controllers. The technique is to efficiently compose steady motions to synthesize agile (non-steady state) gaits with stability guarantees. Specifically, we use the region of attraction at the Poincaré section in conjunction with motion planning tools such as rapidly-exploring random tress (RRT) to switch between two steady state gaits. The rationale behind using this method is that it is at least an order

of magnitude computationally cheaper to synthesize steady state gaits than agile gaits. This contribution is addressed in chapters 5 and 6.

### 1.2.3 Creating Approximate Models of Step-to-Step Dynamics and Using them with Formal Control Tools for Motion Planning

The third contribution of this thesis is to use data-driven techniques to build approximate models of the step-to-step dynamics or Poincaré map. This involves two stages. First, we generate the data from sampling the initial conditions and controls and use the robot simulation. Second, we use a suitable regression model to fit the data. This approach is original as it takes advantage of the fact that the step-to-step dynamics can be approximated by smooth functions despite the non-smoothness of the instantaneous dynamics of locomotion. Once the approximate model of the step-to-step dynamics is obtained, we can use a broad range of formal control tools available for smooth systems. This contribution is addressed in chapter 7.

## 1.3 Organization of the Thesis

The remainder of this thesis is organized as follows.

Chapter 2 presents an orbital control Lyapunov function (OCLF) to exponentially stabilize a bipedal robot walking down a slope. The performance of the OCLF-based controller is compared with a one-step dead-beat controller and eigenvalue-based controller in terms of robustness, energy-efficiency and sensitivity to modeling errors.

Chapter 3 investigates the role of two different control strategies, foot placement control and ankle-push off control, in the stabilization of bipedal gaits using two stability criteria, the one-step dead-beat stabilization and the OCLF-based stabilization developed in chapter 2.

Chapter 4 provides a data-driven technique to find nonlinear control policies that enlarge the region of attraction of the limit cycle of legged robots. The approach starts by assuming an ROA and then finds nonlinear control policies by carrying out trajectory optimization on sampled initial conditions. The approach is implemented on a model of hopping robot followed by verifying the control policies and performing robustness checks.

Chapter 5 deals with creating agile gaits by sequentially composing steady state gaits using heuristics. This is realized by funneling the robot from the start state to goal state by using the region of attraction of successive, overlapping steady state limit cycles at the Poincaré section. The OCLF-based controller is used to provide fast transitioning between two limit cycles.

Chapter 6 presents a framework to create agile gaits by sequentially composing steady state gaits using sampling-based techniques. The approach is to fill the entire entire state space with the regions of attraction and association controllers and then use the rapidly-exploring random trees (RRT) algorithm for feedback motion planning.

Chapter 7 provides a data-driven technique to build an approximate model of the step-to-step dynamics and use this model within a model predictive control (MPC) for navigation on stepping stones.

Finally, chapter 8 provides a summary and discusses possible directions for future work.

# CHAPTER 2

# ORBITAL CONTROL LYAPUNOV FUNCTION FOR EXPONENTIAL STABILIZATION OF BIPEDAL ROBOTS

(A significant portion of this chapter is reproduced from a published paper with the following citation:

Bhounsule, P. A. and Zamani, A.: A discrete control lyapunov function for exponential orbital stabilization of the simplest walker. Journal of Mechanisms and Robotics, 9(5):051011, 2017.)

## 2.1 Introduction

Passive dynamic robots that walk downhill are highly energy-efficient because they rely only on their mass distribution and geometry(McGeer, 1990; Owaki et al., 2010; Steinkamp, 2017). These robots are gravity powered and do not use external control or power. Consequently, such robots are highly energy-efficient. But no control means that these robots cannot correct their motion when disturbed, thus they fall down due to even the slightest disturbance (Schwab and Wisse, 2001; Hobbelen and Wisse, 2007a). The lack of stability limits the practical use of such robots.

The compass gait model is a well studied model of walking (Goswami et al., 1998). The model consists of a point mass torso and two legs connected to each other by a pin joint. One

leg pivots about the ground while the other leg, actuated by a hip actuator, swings about the grounded leg. The legs exchange their roles after the swinging leg contacts the ground. The model is challenging to control because of non-linearity and under-actuation (more degrees of freedom than actuators). Iida et al. (Iida and Tedrake, 2010) controlled the compass gait model by creating an open-loop hip actuator profile that effectively "phase locks" the swing leg to the gait cycle. Manchester et al. (Manchester et al., 2011) used a closed-loop controller in which the swing leg is coupled to the motion of the stance leg using a high gain feedback controller. In this chapter, we consider the simplest walking model, which is a special case of the compass gait model: the ratio of the leg to torso mass is zero. This feature decouples the swing leg from the stance leg. Consequently the hip actuator is not able to influence the motion of the stance leg in the swing phase of motion (that is, the system *cannot be stabilized*). Thus, a controller that does step-to-step or orbital stabilization (as opposed to local stabilization) is more appropriate for this model. Here, the hip actuator controls the step length in order to modulate the stance leg velocity between successive steps.

We develop a stabilizing controller using an orbital control Lyapunov function (OCLF). The Lyapunov function is chosen at the Poincaré section at mid-stance. The mid-stance is defined as the position when the stance leg (grounded leg) is normal to the ramp. Next, a control law is chosen that results in exponential stabilization of the system between steps at the Poincaré section. The key idea is to use the actuated degree of freedom (the swing leg) to control the un-actuated degree of freedom (the stance leg) between steps. Although the continuous control Lyapunov function (CCLF) (Ames et al., 2014) has been used for gait stabilization, the use of

OCLF is novel in this area. The major difference between the two is that OCLF does orbital or step-to-step stabilization while CCLF does local stabilization (see Dingwell and Kang (Dingwell and Kang, 2007) for difference between orbital and local stabilization).

The organization of the chapter is as follows. The background and related work is presented in Section 2.2. The details about the simplest model including equations of motion are discussed in Section 2.3. The details about the OCLF technique are provided in Section 2.4. The results and the discussion are in Sections 2.5 and 2.6, respectively.

## 2.2 Background and Related Work

There are broadly two notations of bipedal robot stability: (1) stability is the ability to not fall down, and (2) stability is the ability to follow a given reference trajectory. Next, we discuss some metrics associated with these notions of stability.

Viability, mean first-passage time, and gait sensitivity norm are some metrics that quantify the robots ability to not fall down. Viability is the set of all states from which the robot can avoid falling down (Wieber, 2008). This is a computationally expensive metric to compute and almost intractable for high dimensional systems. The mean first-passage time is the number of steps the robot can take before falling down (Byl and Tedrake, 2009). This definition is a probabilistic one and is evaluated by simulating the robot in a variety of different disturbances conditions and tends to be computationally expensive as the system dimension increases. The gait sensitivity norm (GSN) is the two norm of the ratio of a gait indicator (e.g., step time, velocity) to a disturbance (e.g., terrain height, push) (Hobbelen and Wisse, 2007a). This is easy

to compute but is sensitive to the choice of a good gait indicator that correlates with falling. Moreover, as GSN is based on linearization, it works well only for small disturbances.

Basin of attraction, the largest eigenvalue of the limit cycle, and Lyapunov function are some metrics that quantify the robots ability to follow a given reference trajectory. The basin of attraction is the set of all initial states that will converge to the given reference trajectory (Schwab and Wisse, 2001; Strogatz, 1994). This measure is computationally expensive even for the simplest system. The magnitude of the largest eigenvalue of the Poincaré map indicates how fast a perturbation in the state would converge back to the reference trajectory (McGeer, 1990). The closer the value is to zero, the faster will be the convergence, while values greater than 1 will diverge and lead to instability. This is a computationally simple measure but is only valid for small perturbations. A Lyapunov function is a positive definite function whose time derivative along any state trajectory of system decreases with time. It is non-trivial to find a Lyapunov function but the recent use of sum of squares optimization provides a generalizable numerical technique (Tedrake et al., 2010). The issue with all these metrics is that they are used to check stability after the controller has been designed.

The eigenvalues of the Poincaré map can be controlled using feedback. We call this controller as an eigenvalue-based controller. The key idea is to linearize the Poincaré map with respect to the state and suitable control action (e.g., step length, push-off). Then pole placement is used to set the eigenvalues of the linearized equation. For example, McGeer (McGeer, 1993) used the instantaneous push-off as the control action to stabilize a 2D model while Kuo (Kuo, 1999)

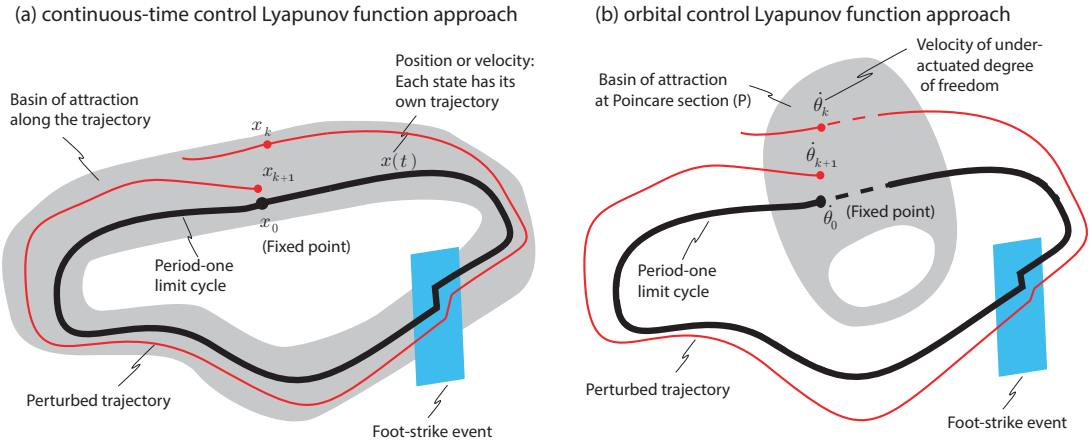


Figure 1: Difference between continuous and orbital control Lyapunov function methodologies: (a) Continuous control Lyapunov function (CCLF) constructs a controller that keeps the trajectory within a tube *along the trajectory*. (b) Orbital control Lyapunov function (OCLF) constructs controller that keeps the trajectory in the basin of attraction only *at the Poincaré section*.

used lateral foot placement to control lateral stability of a 3D model of walking. We describe the eigenvalue-based controller in section 2.4.5 and also compare it with our controller.

The control Lyapunov function provides a generalizable method to design a stable control law (Slotine et al., 1991). For example, Ames et al. (Ames et al., 2014) used continuous-time control Lyapunov functions (CCLF) to guarantee exponential stabilization of the hybrid zero dynamics of the system. A conceptualization of their methodology is shown in Figure 1(a). The thick black line shows the reference trajectory and the thin red line shows a perturbed trajectory. While the CCLF is chosen to keep the trajectory within the gray tube, the discontinuous foot-strike event at the section shown by blue rectangle, tends to push the system out of the gray

tube. However, by choosing suitable tuning parameters, it is possible to keep the perturbed trajectory within the gray tube in spite of the discontinuous foot-strike event.

We take an alternate approach by using an orbital control Lyapunov function (OCLF). This is shown in Figure 1(b). Here we try to keep the perturbed trajectory of the un-actuated degree of freedom (e.g., stance leg of bipedal robot) within the gray region *only* at the Poincaré section. This is done by using the actuated degrees of freedom, that are fully controllable in a step, to affect the un-actuated degree of freedom over a complete step (e.g., using foot placement). In this view, OCLF does not think of the foot-strike event as a disturbance, but rather an essential quantity to modulate the un-actuated degree of freedom. Another difference is that OCLF provides exponential orbital stability while CCLF can only provide asymptotic orbital stability, although CCLF is able to provide exponential stabilization of the hybrid zero dynamics.

In this chapter, we illustrate an application of OCLF using the simplest walking model (Garcia et al., 1998), but with the addition of an actuator between the two legs. The system is nonlinear and under-actuated. Since the point mass torso is heavy compared to the legs, the swing leg motion cannot affect the stance leg motion in the swing phase, and the system is thus decoupled. However, the swing leg (actuated degree of freedom) motion can influence the stance leg (un-actuated degree of freedom) motion by appropriate foot-placement (e.g., big step reduces the stance leg velocity (Bhounsule, 2015)), which is exploited in the OCLF approach.

## 2.3 Model

### 2.3.1 Model description

Figure 2 shows a cartoon of the simplest walker (Garcia et al., 1998). The model has mass $M$ at the hip and point mass $m$ at each of the feet. Each leg has length $\ell$. Gravity $g$ points downwards. The leg in contact with the ramp is called the stance leg while the other leg is called the swing leg. The angle made by the stance leg with the normal to the ramp is $\theta$ and the angle made by the swing leg with the stance leg is $\phi$. The hip torque is $\tau$ and the ramp slope is $\gamma$.

A single step of the walker is given below:

$$\underbrace{\overset{\text{mid-stance}}{\overset{\frown}{\longrightarrow}}\ \text{Single Stance}\ \overset{\text{collision}}{\overset{\frown}{\longrightarrow}}\ \text{Foot-strike} \rightarrow\ \text{Single Stance}}_{\text{single step}}\ \overset{\text{mid-stance}}{\overset{\frown}{\longrightarrow}} \tag{2.1}$$

A single step consists of two phases; a single stance phase where the swing leg pivots about a stationary stance foot and foot-strike phase where there is support transfer and the legs exchange their roles. These phases are connected through two switching events; a mid-stance event where the stance leg is normal to the ramp and a collision phase where the leading leg touches the ground. Note that we have chosen a single step to start and end at mid-stance unlike the usual convention of using the instant just after foot-strike. The use of mid-stance as opposed to foot-strike for the Poincaré section will become clear in the section on Methods (see Sec. 6.4). Next, we describe the equations that describe the phases and events.

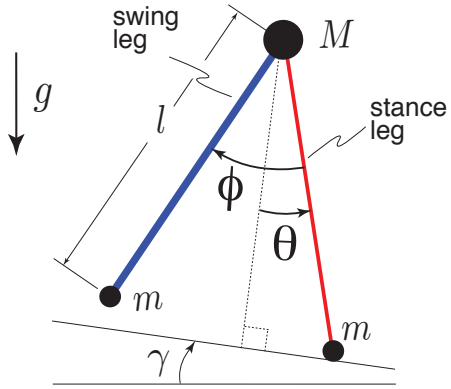


Figure 2: Model: The simplest slope walking model analyzed by Garcia et al. (Garcia et al., 1998).

#### 2.3.1.1 Mid-stance event:

The mid-stance event is the logical condition that indicates the stance leg is normal to the ramp and is given by

$$\theta = 0. \tag{2.2}$$

#### 2.3.1.2 Single stance phase (continuous dynamics):

In this phase of motion, the stance leg pivots and rotates about the stance foot contact point while the swing leg pivots and rotates about the hinge connecting the two legs. Our assumptions are: the stance foot does not slip (infinite friction between stance foot and the ground), there is no hip hinge friction, and foot scuffing is ignored. We obtain Equation 2.3 and Equation 2.4 defined below by taking moments about the stance foot contact point and hip

hinge respectively, and non-dimensionalizing time with $\sqrt{\ell/g}$ and applying the limit, $m/M \to 0$. In (Equation 2.4), $\tau$ is the non-dimensional torque obtained by dividing the dimensional torque by $Mg\ell$. The equations are:

$$\ddot{\theta} = \sin(\theta - \gamma), \tag{2.3}$$

$$\ddot{\phi} = \sin(\theta - \gamma) + \{\dot{\theta}^2 - \cos(\theta - \gamma)\}\sin(\phi) + \tau. \tag{2.4}$$

#### 2.3.1.3 Collision event:

The collision event is the logical condition that indicates the leading leg is about to make contact with the ground. We introduce a step down of height $h$ at the collision. This is taken to be zero, except for testing the robustness of the control approach. The collision event is given by

$$\cos(\phi - \theta) - \cos(\theta) = h. \tag{2.5}$$

#### 2.3.1.4 Foot-strike phase (discontinuous dynamics):

In this phase of motion, the legs exchange their roles, that is, the current swing leg becomes the new stance leg and the current stance leg becomes the new swing leg. There is an instantaneous plastic collision (no slip and no bounce) of the swing leg. The swapping of legs is expressed by Equation 2.6 and Equation 2.7. The angular rates of the legs after support exchange are given by Equation 2.8 and Equation 2.9 and are obtained by applying conserva-

tion of angular momentum about stance foot contact point and hip hinge respectively, followed by non-dimensionalizing time with $\sqrt{\ell/g}$ and applying the limit, $m/M \to 0$. In the algebraic equations below, the state variables before and after collision are denoted using the superscript $-$ and $+$ respectively.

$$\theta^+ = \theta^- - \phi^-, \tag{2.6}$$

$$\phi^+ = -\phi^-, \tag{2.7}$$

$$\dot{\theta}^+ = \dot{\theta}^- \cos\phi^-, \tag{2.8}$$

$$\dot{\phi}^+ = \dot{\theta}^- \cos\phi^-(1 - \cos\phi^-). \tag{2.9}$$

### 2.3.2 Failure modes:

There are two failure modes for the simplest walker and they are described below. These lead to two conditions on the state of the system and are checked at each integration step. Violation of any of these conditions is interpreted as system failure.

1. *Falling Backwards:* Falling backwards is detected when the angular velocity of the stance leg is positive (note that forward velocity is indicated by a negative angular velocity). Thus the condition for failure is: $\dot{\theta} \geq 0$.

2. *Flight phase:* Flight phase is detected when the vertical ground reaction force, $R_y = \cos(\theta - \gamma) - \dot{\theta}^2 \leq 0$. Thus the condition for failure is:

$$\dot{\theta}^2 - \cos(\theta - \gamma) \geq 0 \tag{2.10}$$

## 2.4 Methods

### 2.4.1 Overview of control technique

From the single stance phase Equation 2.3 and Equation 2.4, we see that: (1) the hip torque can be used to control the swing leg and (2) the motion of the swing leg does not affect the motion of the stance leg. Thus, by the controllability definition (Antsaklis and Michel, 2006), although the swing leg motion, $\phi(t)$, is fully controllable, the motion of the stance leg, $\theta(t)$, is not controllable *within a step*. However, we can find a function, $F$, that maps the mid-stance between consecutive steps and is indexed by step number, $k$. Thus

$$\dot{\theta}_{k+1} = F(\dot{\theta}_k, \phi^-) \tag{2.11}$$

where $\phi^-$ is swing leg angle at foot-strike and is related to the step length. Given the measurement at step k, $\dot{\theta}_k$, the hip torque can be used to modulate $\phi^-$ to control $\dot{\theta}_{k+1}$. Thus, the stance leg is fully controllable *between steps*. Based on these observations, we use a hierarchical control approach: *the stance leg velocity is controlled between steps using foot placement obtained from OCLF, while the swing leg is controlled using a trajectory tracking controller based on the foot placement angle.*

### 2.4.2 Mid-stance to mid-stance map for stance leg:

In this section we present equations that can be used to numerically solve for the mid-stance to mid-stance map, $F$, given by Equation 2.11.

We do an energy balance for the stance leg between the mid-stance and the instant just before foot-strike and then again from just after foot-strike to the next mid-stance to get Equation 2.12 and Equation 2.13 respectively

$$\frac{(\dot{\theta}_k)^2}{2} + \cos\gamma = \frac{(\dot{\theta}^-)^2}{2} + \cos\left(\frac{\phi^-}{2} - \gamma\right), \tag{2.12}$$

$$\begin{aligned}\frac{(\dot{\theta}_{k+1})^2}{2} + \cos\gamma &= \frac{(\dot{\theta}^+)^2}{2} + \cos\left(\frac{\phi^+}{2} - \gamma\right), \\ &= \frac{(\dot{\theta}^- \cos\phi^-)^2}{2} + \cos\left(\frac{\phi^-}{2} + \gamma\right).\end{aligned} \tag{2.13}$$

To get the Equation 2.13 we have used the foot-strike conditions given by Equation 2.7 and Equation 2.8. We have also assumed that controller is not going to be aware of the step down disturbance, so we set $h = 0$ (see Equation 2.5). This leads to the condition $\phi^- = 2\theta^-$ and is used to write the cos expression on the right side of Equation 2.13 in terms of $\phi^-$.

### 2.4.3 Orbital control lyapunov function (OCLF)

OCLF was explained in the last paragraph of section 2.2 and illustrated in Figure 1. We provide mathematical details next.

First, we need a period-one limit cycle so that we are able to construct an OCLF to stabilize it. A period-one limit cycle is specified by setting $\dot{\theta}_{k+1} = \dot{\theta}_k = \dot{\theta}_0$ in Equation 2.11 and solving for $\phi^- = \phi_0$ to get the following

$$\dot{\theta}_0 = F(\dot{\theta}_0, \phi_0) \tag{2.14}$$

Second, we define the OCLF as follows

$$V(\Delta\dot{\theta}_k) = \Delta\dot{\theta}_k^2 = (\dot{\theta}_k - \dot{\theta}_0)^2, \tag{2.15}$$

where $V(0) = 0$ and $V(\Delta\dot{\theta}_k) > 0$ at the mid-stance event, $\theta = 0$. For the system to be asymptotically stable, the following condition needs to be satisfied

$$V(\Delta\dot{\theta}_{k+1}) - V(\Delta\dot{\theta}_k) = (\dot{\theta}_{k+1} - \dot{\theta}_0)^2 - (\dot{\theta}_k - \dot{\theta}_0)^2 < 0 \tag{2.16}$$

However, we are interested in exponential stability so we set the following condition

$$V(\Delta\dot{\theta}_{k+1}) - V(\Delta\dot{\theta}_k) = -cV(\Delta\dot{\theta}_k), \tag{2.17}$$

where $c$ is a user chosen positive constant such that, $0 < c < 1$. Thus the condition for exponential stability can be written as

$$\begin{aligned} \Delta\dot{\theta}_{k+1}^2 - (1-c)\Delta\dot{\theta}_k^2 &= 0 \\ \implies \Big(F(\dot{\theta}_k, \phi^-) - \dot{\theta}_0\Big)^2 - (1-c)\Big(\dot{\theta}_k - \dot{\theta}_0\Big)^2 &= 0. \end{aligned} \tag{2.18}$$

The stance leg velocity at mid-stance, $\dot{\theta}_k$, is measured and for given $c$ and $\dot{\theta}_0$, the swing leg angle just before foot-strike, $\phi^-$, is solved using the above equation. The choice of $c$ determines

the rate of decay of a perturbation in the mid-stance velocity; a larger value of $c$ indicates faster convergence. When $c = 1$ there is full correction of disturbances in a single step, also known as one-step dead-beat control (Antsaklis and Michel, 2006). Dead-beat control gives the fastest convergence, "dead-beat" convergence (Gauthier and Boulet, 2005).

Third, we need to define a controller for the hip torque based on the computed swing leg angle at foot-strike. We use 2 third-order, time-based trajectories for the swing leg; one for the instant from mid-stance to foot-strike and another one from instant after foot-strike to mid-stance. Each of the third order polynomials has four coefficients that are computed based on the initial and final values of the position and velocity which are completely known. One also needs the time from mid-stance to instant just before foot-strike ($T_{\text{mid-fs}}$) and from instant after foot-strike to mid-stance ($T_{\text{fs-mid}}$). These can be computed as follows

$$T_{\text{mid-fs}} = \int_{\theta=\frac{\phi^-}{2}}^{\theta=0} \frac{d\theta}{\sqrt{(\dot{\theta}_k)^2 + 2(\cos\gamma - \cos(\theta - \gamma))}} \tag{2.19}$$

$$T_{\text{fs-mid}} = \int_{0}^{\theta=\frac{\phi^-}{2}} \frac{d\theta}{\sqrt{(\dot{\theta}^+)^2 + 2(\cos(\frac{\phi^-}{2} - \gamma) - \cos(\theta - \gamma))}} \tag{2.20}$$

The time-based trajectory is open-loop but we have a proportional derivative controller on the position at the end of the time-based trajectory. The controller comes into effect only when the time from mid-stance to foot-strike is greater than the predicted time using Equation 2.19 (e.g., during a step-down disturbance).

### 2.4.4 Why use mid-stance position for Poincaré map?

We use mid-stance position, defined as $\theta = 0$, for the Poincaré section. In general, any section such as $\theta =$ constant will work just as well. This particular choice of Poincaré section enables us to choose $V$ to be a function of only the stance leg velocity, (e.g., $V(\Delta\dot{\theta}_k) = \Delta\dot{\theta}_k^2$). This allows us to do a hierarchical control: the swing leg is controlled in the continuous sense to affect the stance leg velocity in the orbital sense or between steps. On the other hand, a section before or after foot-strike would be a function of stance leg and swing leg angle, $\cos(\phi - \theta) - \cos(\theta) = 0$. In this case, $V$ needs to be a function of stance leg and swing leg position and velocity, that is, $V(\Delta\theta_k, \Delta\dot{\theta}_k, \Delta\phi_k, \Delta\dot{\phi}_k)$. This complicates the controller design. Moreover, in a practical sense, the instant just before or after foot-strike is most prone to modeling errors (e.g., mis-estimation of ground height) and noise (e.g., two sensors, stance and swing leg angles, are needed and each of them would have their own noise levels). Perhaps the best choice for the Poincaré section is the instant after foot-strike but following the decay of vibrations due to foot-strike because that instant gives the swing leg the entire step for adjusting the step length.

### 2.4.5 Eigenvalue-based controller

The eigenvalue-based controller imparts orbital stability in the linearized sense. The Poincaré map is linearized based on the state and suitable control action. Then the eigenvalues are modulated/placed using the control action. We describe the mathematical details next.

First, we linearize the Poincaré map given by Equation 3.9 about the passive limit cycle:

$$\Delta\dot{\theta}_{k+1} = A\Delta\dot{\theta}_k + B\Delta\phi^-, \tag{2.21}$$

where $A = \dfrac{\partial\dot{\theta}_{k+1}}{\partial\dot{\theta}_k}$ and $B = \dfrac{\partial\dot{\theta}_{k+1}}{\partial\phi^-}$.

Next, we assume a linear controller, $\Delta\phi^- = -K\Delta\dot{\theta}_k$ and substitute it in Equation 2.21 to obtain

$$\Delta\dot{\theta}_{k+1} - (A - BK)\Delta\dot{\theta}_k = 0. \tag{2.22}$$

Finally, we compare Equation 2.22 with Equation 2.18 to compute the gain $K$. We have

$$A - BK = \sqrt{1-c} \quad \Longrightarrow K = \frac{A - \sqrt{1-c}}{B}. \tag{2.23}$$

This allows use to compare the OCLF controller with the eigenvalue-based controller in an objective way. Note that $A$, $B$, and $K$ are scalars in our problem and we have used the positive root from Equation 2.18 (that is, $\sqrt{1-c}$) to ensure monotonic decay in the perturbed state, $\Delta\dot{\theta}_{k+1}$.

## 2.5 Results

All computations are done using MATLAB. We used *ode113* for integrating the equations of motion and *fsolve* to find the limit cycle. To find the passive limit cycle, we chose $\gamma = 0.009$ and

set the controller off ($\tau = 0$). Our passive limit cycle has the fixed point; $\theta_0 = 0$, $\dot{\theta}_0 = -0.0593$, $\phi_0 = -0.0532$, and $\dot{\phi}_0 = -0.3397$. The swing leg position and velocity just before foot-strike are; $\phi^- = -0.4006$ and $\dot{\phi}^- = 3.5 \times 10^{-4} \sim 0$. We used central differencing with step size of $10^{-5}$ to compute $A = 0.3711$ and $B = -0.4026$. We also found the Jacobian of the Poincaré section using central difference. The largest eigenvalue of the Jacobian is 0.5891, which is less than 1, indicating that the passive limit cycle is stable

### 2.5.1 Stability

We use the basin of attraction metric to compare the stability of the uncontrolled case (passive dynamic walking) with our control approach. The basin of attraction is defined as the set of initial conditions that converge to the limit cycle as time goes to infinity (Strogatz, 1994). To compute the basin of attraction we proceed as follows. We perturb the mid-stance velocity from its nominal value and do forward simulations of the walker for 50 steps. We find the upper and lower limits of the mid-stance velocity that allows the walker to complete 50 steps without falling down. The limits give the boundary of the basin of attraction. Note that 50 steps are large enough to be able to see the effect of the perturbation but small enough that we are able to obtain results in a short period of time. We repeat the above procedure for different ramp slopes and with and without control.

Figure 3 shows the basin of attraction for the simplest walker with and without control as a function of ramp slope. The white region shows the basin of attraction for the passive or uncontrolled case. The gray region (including the white region) shows the basin of attraction for three values of $c$. The basin of attraction is substantially improved using our control approach.

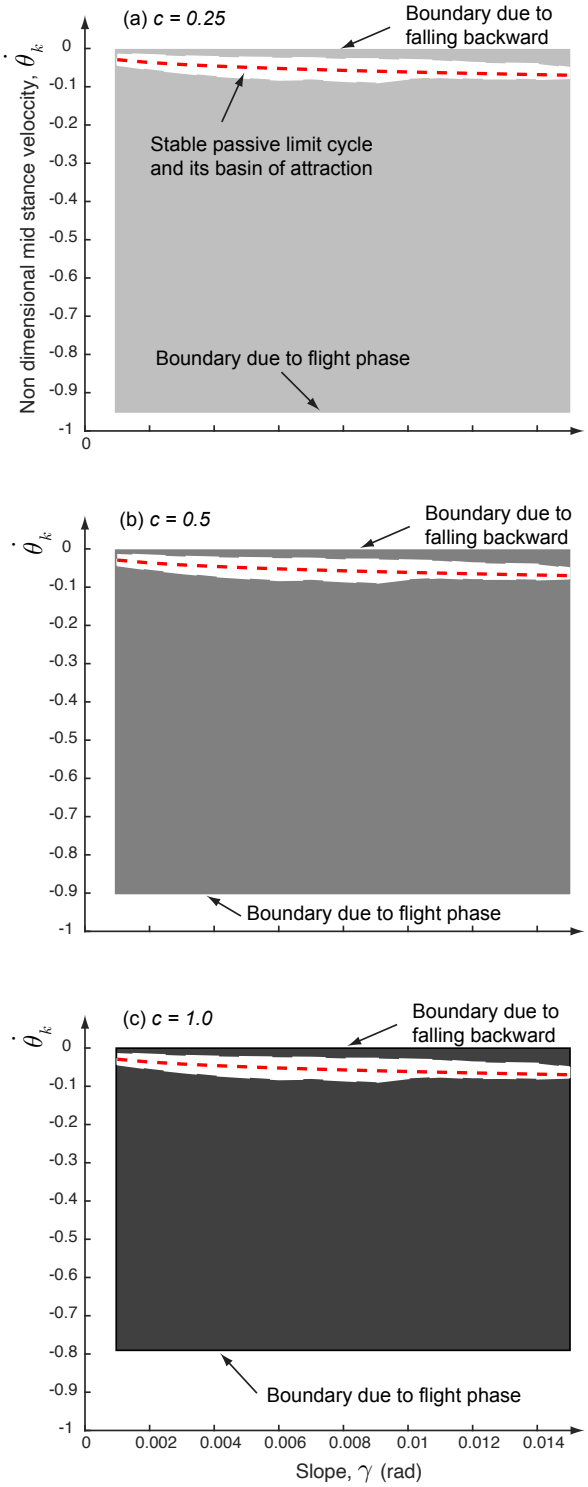


Figure 3: Basin of attraction: Uncontrolled/passive dynamic case (white region) and OCLF (gray region includes the white region for the uncontrolled case) for three values of $c$. The red dashed line shows the passive limit cycle.

We compute the area of the basin of attraction for the uncontrolled case and the three control cases with different $c$ values using numerical quadrature. Then we find the ratio of areas for a specific $c$ value to the uncontrolled case. We found ratios of 19.56, 18.60, 16.4 for $c = 0.25$, $c = 0.5$, $c = 1$ respectively. This indicates that our controller was able to increase the region of attraction by an order of magnitude. Further, we note that $c = 1$ has the smallest basin of attraction and this increases as $c$ decreases. This can be explained as follows: A larger $c$ value corresponds to a faster convergence to the limit cycle. For example, when $c = 1$, the convergence is in a single step. This is achieved by taking a bigger than nominal step as reduction in velocity between steps is directly proportional to the step length (Bhounsule, 2015; Ruina et al., 2005). But a bigger step will lead to flight phase at a lower velocity (see Equation 2.10). As $c$ decreases, the controller chooses shorter steps, leading to higher velocities for flight phase. Consequently, as $c$ decreases, the basin of attraction increases.

Next, we demonstrate the exponential stabilization provided by the OCLF. We perturb the mid-stance velocity to $\dot{\theta}_k = -0.5$ at a slope of $\gamma = 0.009$ and plot the mid-stance velocity as a function of step number to obtain Figure 4. For $0 < c < 1$ there is exponential stabilization (dashed and dash-dot lines) and a larger value of $c$ gives faster convergence to the nominal mid-stance velocity. However, $c = 1$ (dotted line) leads to the condition $\dot{\theta}_{k+1} = \dot{\theta}_0$ (see Equation 2.15 and Equation 2.17) or a "dead-beat" convergence in a single step. This is faster than exponential convergence.

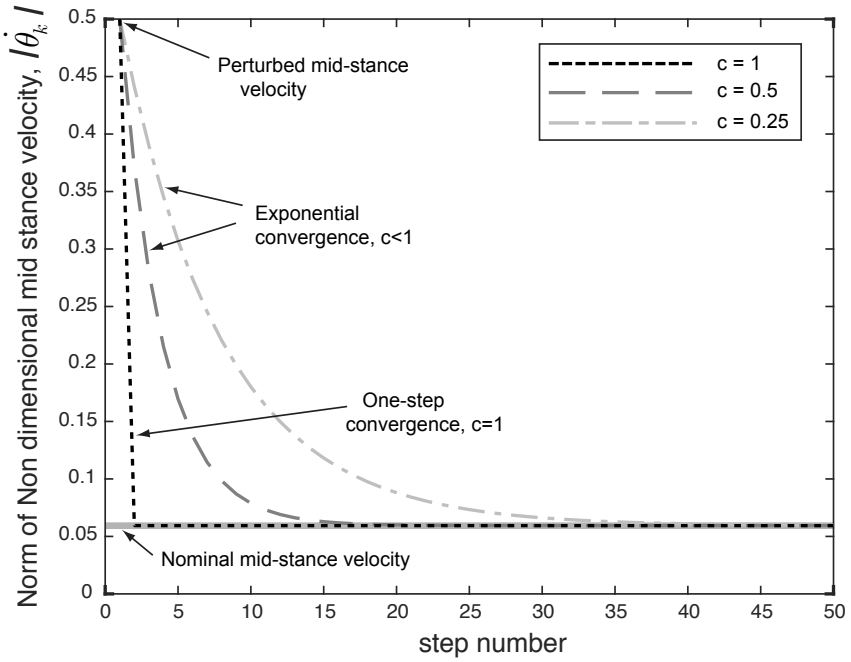


Figure 4: Rate of stabilization: The convergence is exponential for $0 < c < 1$ and "dead-beat" for $c = 1$ for state perturbation $\dot{\theta} = -0.5$ for $\gamma = 0.009$.

### 2.5.2 Robustness

We evaluate robustness by computing the average number of steps that the controller can withstand without the flight phase or falling backward on uneven terrain chosen from a random distribution with maximum height of $\sigma$ (Byl and Tedrake, 2009). Note that $\sigma$ can be interpreted as a maximum step down normalized against the leg length. We create 10 terrains with 400 steps each, selected from a random distribution with maximum height $\sigma$. For each terrain, we do forward simulations of the system for a given value of $c$. We evaluate the average number of steps and average energy used per step for the 10 terrains for different values of $\sigma$ and $c$.

Figure 5(a) shows results for robustness as a function of $\sigma$ for different values of $c$ for a slope of $\gamma = 0.009$. The average number of steps is infinity for $\sigma = 0$ which corresponds to no disturbance. As $\sigma$ increases, the average number of steps decreases steadily as shown in the figure. The average number of steps to failure is almost the same for each value of $c$. The average number of steps to failure is 393 at $\sigma = 0.005$ and decreases to 40 at $\sigma = 0.05$. We found that for very low values of $c$, $0 < c \leq 0.005$, the robustness dropped appreciably with increase in step height. We also did the robustness test without any control. We observed that the average number of steps to failure is 4 at $\sigma = 0.005$ and 0 at $\sigma = 0.05$.

Figure 5(b) demonstrates the average energy used per step as a function of $\sigma$. We obtain the energy usage by integrating the absolute value of mechanical work done by the hip actuator as the robot walks on the terrain and dividing it by the total number of successful steps. For a given controller specified by $c$, the average energy per step increases with $\sigma$. This is because a larger $\sigma$ leads to a larger deviation from the limit cycle and consequently more energy is needed to get back to the nominal trajectory. For a given $\sigma$, the average energy per step is the lowest for $c = 1$ and increases as $c$ decreases. This can be explained as follows: larger value of $c$ implies a faster convergence to the limit cycle and hence lower energy usage since the limit cycle is passive (energy usage of zero).

Figure 5(c) illustrates the average step length, the control strategy, as a function of $\sigma$. Each data point for a particular $\sigma$ and $c$ was obtained by averaging the step length for the 10 simulation runs. For a given maximum step down, $\sigma$, the largest average step length is for $c = 1$ and decreases as $c$ decreases. This is because larger $c$ values correspond a greater

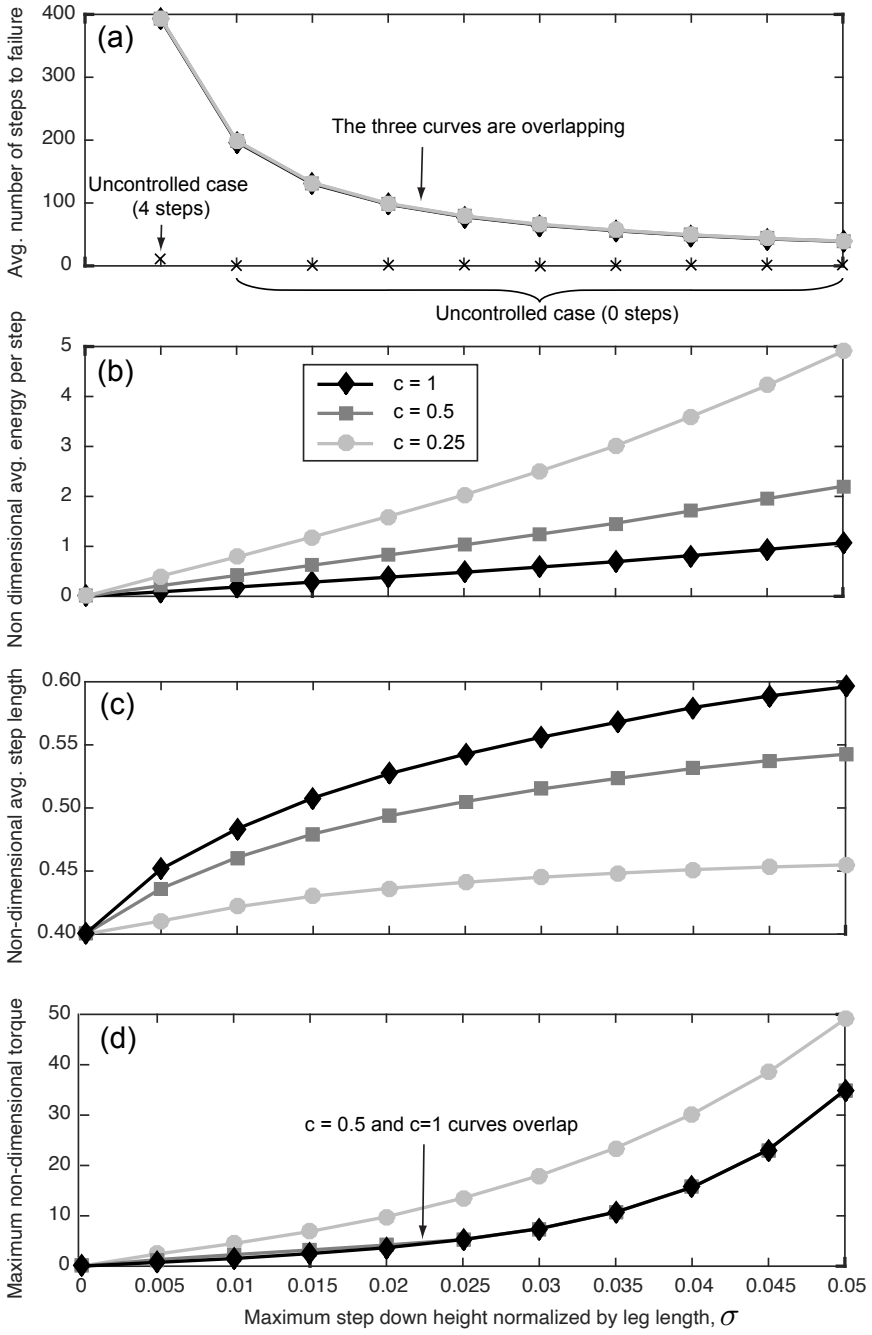


Figure 5: Robustness to step-down disturbance for different $c$ values: (a) Average number of steps to failure, (b) average energy (absolute value of mechanical work done by the hip actuator) used per step, (c) average step length, and (d) maximum torque needed.

decrease of the OCLF, which is possible by taking a bigger step length. Also, the average step length increases as the step down, $\sigma$, increases. Since a larger step down leads to a larger deviated mid-stance velocity, the average step length needs to consequently increase to regulate the mid-stance velocity at the subsequent step.

Finally, Figure 5(d) depicts the maximum torque needed as a function of $\sigma$. This was obtained by searching for maximum torque across the 10 simulations runs for a given $\sigma$. The maximum torque is almost the same for $c = 0.5$ and $c = 1$ but is greater for $c = 0.25$ for a given $\sigma$. This indicates that low values of $c$ require more torque, consequently, a larger actuator for stabilization. The maximum torque increases as the maximum step down increases for a given $c$ because bigger step down requires bigger step length and hence, higher torque.

Figure 6 compares the sensitivity of the OCLF controller to modeling errors. To generate these simulations, we provided the OCLF controller with a slope of $\gamma = 0.01$, which is 10% greater than the actual value of $\gamma = 0.009$, a modeling error. The robustness measured by the average number of steps to failure for $c = 0.25$ and $c = 0.5$ is virtually unchanged (compare Figure 5(a) with Figure 6(a)). However, the robustness decreases substantially for $c = 1$. This can be explained as follows: a larger than actual $\gamma$ in the model, leads to a bigger step and to a greater energy lose during collision, and consequently the model falls backwards. The average number of steps walked before failure for $c = 1$ is about 9 at $\sigma = 0.005$ and it increases to about 40 at $\sigma = 0.025$. This is because an increase in step down disturbance nullifies the effect of this particular modeling error: bigger than actual $\gamma$ in the model leads to a bigger than the actual step but this regulates the walking speed better as the step down disturbance is increased.

Figure 6(b) shows that $c = 1$ is more energy-efficient, followed by $c = 0.5$ and $c = 0.25$. The maximum torques for $c = 1$ and $c = 0.5$ are the same while those for $c = 0.25$ are much higher.

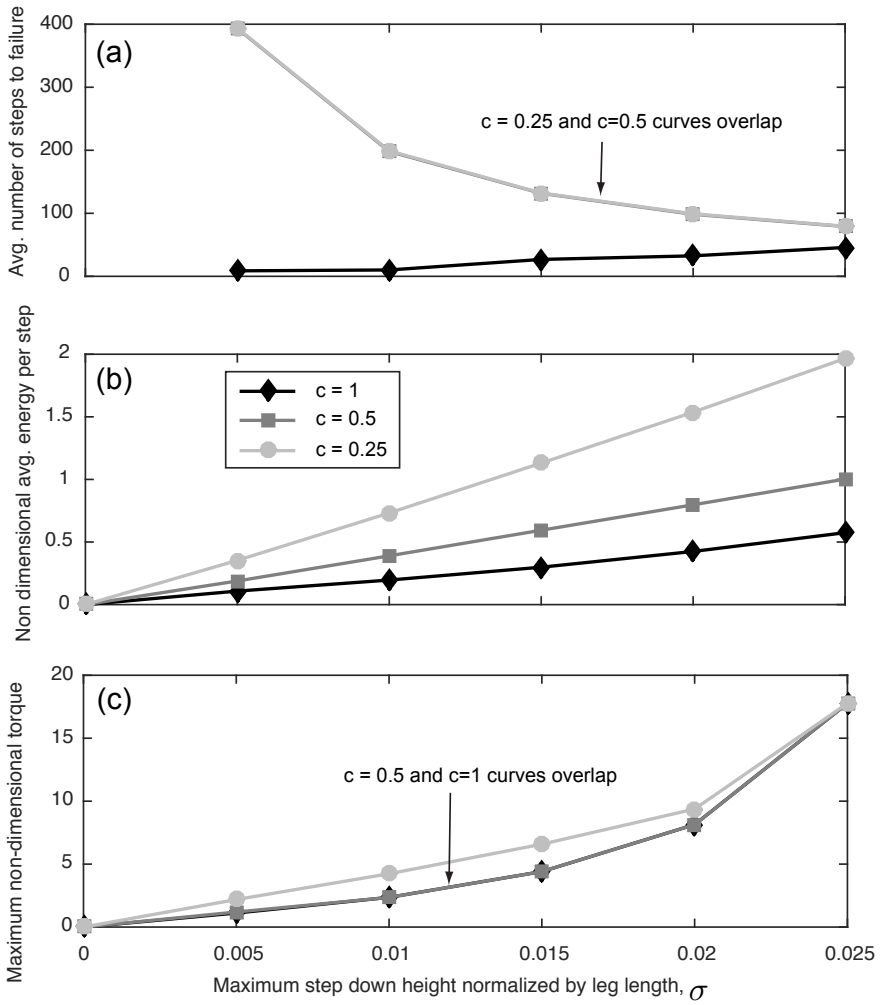


Figure 6: Sensitivity of OCLF to modeling errors: (a) Average number of steps to failure, (b) average energy (absolute value of mechanical work done by the hip actuator) used per step, (c) maximum torque needed. The slope, $\gamma$, is 1.1 of its actual value.

Figure 7 compares the OCLF controller with the eigenvalue-based controller for $c = 0.9$, which corresponds to an eigenvalue of 0.32. Figure 7(a) demonstrates that both the controllers have similar robustness. However, the OCLF controller is more energy-efficient (see Figure 7(b)) and requires a lower maximum torque Figure 7(d). The main difference between the two controllers is that eigenvalue-based controller uses a linearization while OCLF is based on the actual model, which is non-linear. In this case, the linearization leads the eigenvalue-based controller to over correct by taking longer than ideal steps as shown in Figure 7(c).

## 2.6 Discussion

We have presented the orbital control Lyapunov function (OCLF) to exponentially stabilize the simplest walking model with a hip actuator. Our control approach is able to enlarge the basin of attraction by an order of magnitude and increase the average number of steps to failure by two orders of magnitude over the passive dynamic walking. We compared the OCLF controller with a one-step dead-beat controller and eigenvalue-based controller. We found that all three controllers had similar robustness. However, the dead-beat controller was the most energy-efficient and required lower maximum torque followed by the OCLF controller and finally the eigenvalue-based controller.

Theoretically, one-step dead-beat stabilization is better than exponential stabilization in terms of convergence rate and final value achieved. While one-step dead-beat stabilization converges fully to the nominal value in a single step, there will always be some finite (usually very small) error between the actual and nominal state for exponential stabilization. Also, the

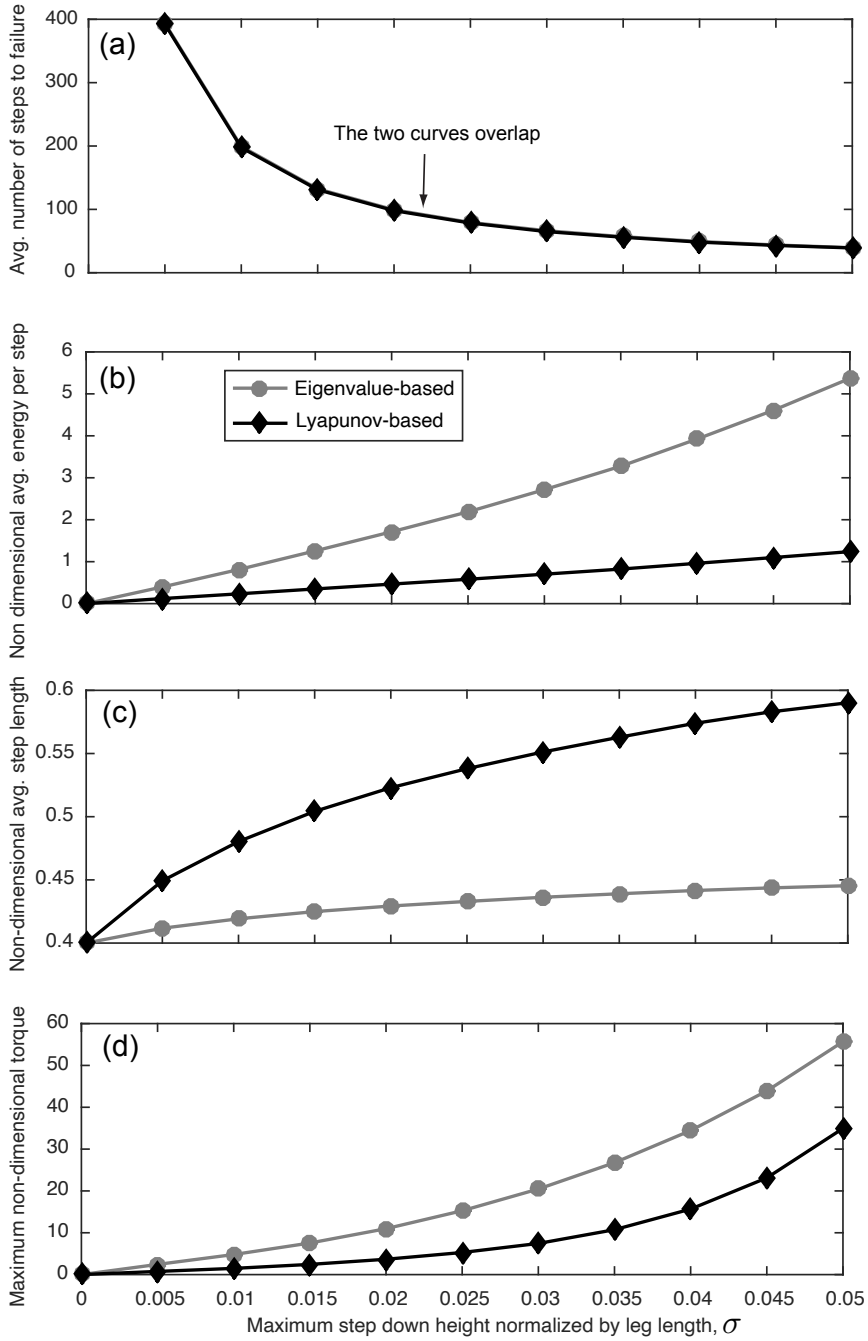


Figure 7: Robustness to step-down disturbance comparing Eigenvalue-based with Lyapunov based stability: (a) Average number of steps to failure, (b) average energy (absolute value of mechanical work done by the hip actuator) used per step, (c) average step length, and (d) maximum torque needed. All plots are for $c = 0.9$ which corresponds to an eigenvalue of $\sqrt{1-c} = 0.32$.

one-step dead-beat stabilization leads to lower energy usage and lower maximum torque than exponential stabilization suggesting its superior performance. However, the robustness of the one-step dead-beat stabilization reduces quickly in the presence of modeling errors. Thus, in practice, an exponential stabilized controller is preferred.

The most common stability metric used in legged locomotion is the maximum eigenvalue of Jacobian of the Poincaré map (McGeer, 1990). This is a linear stability measure that holds true for small state perturbations only. On the other hand, our measure is a non-linear measure that works for large state perturbations. Some authors have optimized the maximum eigenvalues to create stable walking gaits (Mombaur et al., 2005; Chevallereau et al., 2009). The issue with this approach is that the maximum eigenvalue sometimes is a non-smooth function of the robot and control parameters (Goswami et al., 1998), which will lead to numerical issues when used in conjunction with gradient-based optimization methods. In contrast, our stability measure is a function of the velocity of the under-actuated degree of freedom only. Although velocity is discontinuous at foot-strike, the optimization can be made smooth by using multiple shooting method (Betts, 1998). This makes it possible to apply gradient-based methods which lead to faster convergence to the optimal solution for smooth problems. Note that multiple shooting method allows one to handle discontinuities in the physics of the problem (e.g., due to foot-strike), and not in the control parameters.

The eigenvalues of the Jacobian of the Poincaré map can be manipulated using a feedback controller as was done in this chapter. We used pole placement to place the eigenvalues but one can also use a discrete linear quadratic regulator, DLQR (e.g., (Bhounsule et al., 2015)).

In the feedback approach, eigenvalues of the closed-loop system are manipulated using pole placement or DLQR while in the optimization approach discussed in the previous paragraph, the eigenvalues of the open-loop system are modified. The feedback control method does not require smoothness of the eigenvalues and is computationally faster than the optimization-based method. However, since the the eigenvalue-based feedback controller is based on linearization of the system, it usually performs poorly in the presence of large perturbations (disturbances, sensor noise, and modeling errors) as we have seen in this chapter.

One advantage of the OCLF is that we are able to construct a controller for a given stability bound during the design phase. This is in contrast to the more common approach of checking stability after control design has taken place, which is time consuming and provides limited stability guarantees. Specifically, in our approach Equation 2.17 gives the stability condition and can be incorporated easily into the control design framework (e.g., optimization). One can incorporate stability in the controller design by maximizing the mean first-passage time using a stochastic optimization (Byl and Tedrake, 2009). Another approach is to find a heuristic control strategy that imparts gait stability such as swing leg retraction (swing leg moves backwards just before foot-strike) (Wisse et al., 2005). But recent results have shown that under certain conditions (big slopes and dead-beat stabilization), swing-leg protraction can also impart gait stability (Bhounsule and Zamani, 2017).

The OCLF approach uses a single measurement, the stance leg velocity at mid-stance, to adjust the foot step location. A practical way to achieve this would be to place an inertial measurement unit (IMU) on the torso. The IMU serves two purposes: one, it detects mid-stance

event using the accelerometer; and two, the forward velocity using the gyroscope. Another feature of OCLF is that it is predictive, that is, the control technique is able to calculate the desired foot placement at mid-step, giving ample time for the hip actuator to perform the required control action.

# CHAPTER 3

# INVESTIGATING THE ROLE OF FOOT PLACEMENT AND PUSH-OFF CONTROL IN ORBITAL STABILIZATION OF BIPEDAL ROBOTS

(A significant portion of this chapter is reproduced from a published paper with the following citation:

Zamani, A. and Bhounsule, P. A.: Foot placement and ankle push-off control for the orbital stabilization of bipedal robots. In 2017 IEEE International Conference on Intelligent Robots and Systems (IROS), 2017.)

## 3.1 Introduction

Dynamic walking robots rely mainly on foot placement control (stepping in the direction of fall) for orbital or step-to-step stability to achieve natural-looking locomotion. However, push-off control (regulating the ankle motion before or after foot-strike) has also emerged as a promising alternative for balance control of dynamic locomotion. Understanding the role of these two strategies for gait stabilization is expected to help in the development of robust dynamic walking robots. In this chapter, we investigate the role of foot placement control and ankle push-off control in gait stabilization of a dynamic walking model by considering uneven terrain and two notions of stability — one-step dead-beat stabilization and exponential orbital stabilization.

## 3.2 Background and Related Work

Foot placement control was used as early as the 1980's by Marc Raibert to regulate the forward speed of his dynamically balancing hopping robots (Raibert, 1986). He defined the horizontal travel distance of the robot as the CG print. The normal, symmetric gait is created by placing the foot in the middle of the CG print. To increase robot speed, it needs to place its foot slightly before the middle of the CG print and vice versa to decrease its speed. More recently, Pratt et. al. (Pratt et al., 2006) formalized foot placement control by defining the capture point. Capture point is the location that the robot needs to place its foot in order to come to a complete stop when pushed. It is straightforward to extend this idea to generate and control steady locomotion.

Push-off control was first shown to explain the energetics of human locomotion by Kuo (Kuo, 2002). He showed that ankle push-off just before foot-strike is four times energy-efficient than push-off after foot-strike or hip actuation. Consequently, several dynamically balanced legged robots have used push-off control as means to create energy-efficient walking motions(Collins and Ruina, 2005; Bhounsule et al., 2014). Hobbelen and Wisse (Hobbelen and Wisse, 2008) have shown that ankle push-off is also able to improve the robustness while maintaining the energy-efficiency.

Goswami et al. (Goswami et al., 1997) and Spong (Spong, 1999) have investigated the role of ankle torque and hip torque individually in stabilizing the compass gait model. The key idea is to use either actuator to track a reference energy level obtained from a passive dynamic limit cycle (Goswami et al., 1998), but they have not compared the two control strategies. Byl

and Tedrake (Byl and Tedrake, 2008) compared the performance of foot placement with push-off. They used the value iteration algorithm to find one-step control policy that maximizes probability of not falling on rough but 'known' terrain. They found that constant push-off control and adjustable foot placement leads to the most robust controller. Our study differs from theirs in that we assume the terrain profile is unknown.

More recently, a study by Kim and Collins (Kim and Collins, 2017) investigated the role of foot placement and ankle push-off using a three-dimensional model of human walking with single and double support phase. They found that ankle push-off control is twice as effective as foot placement control to recover from disturbances for changing terrain with step up and step down. Our study investigates the role of two strategies with a simple sagittal model and using two different stability criteria: one-step stabilization and exponential stabilization. Our motivation is to check if Kim and Collins' conclusions hold true for different stability measures and for different perturbations, namely step up and step down considered separately. In doing so, we are able to find which strategy works best for a given terrain disturbance and determine how the stability criteria influence the robustness and energy usage. These principles should aid in making design choices and controller design for bipedal robots and artificial devices.

## 3.3 Model

### 3.3.1 Model description

Figure 8 shows a model of the biped. The model has a mass $M$ at the hip and point mass $m$ at each of the feet. Each leg has length $\ell$. Gravity $g$ points downwards. The leg in contact

with the ground is called the stance leg while the other leg is called the swing leg. The angle made by the stance leg with the normal to the ground is $\theta$ and the angle made by the swing leg with the stance leg is $\phi$. The hip torque is $T$. There is a torsional spring with spring constant $K$ between the two legs (not shown). The rest length of the spring is zero and corresponds to the position when both legs are parallel. There is a linear actuator which is used to apply an impulsive push-off, $P$, just before foot-strike.

A complete walking step of the model is similar to that of the simplest walker defined by Equation 2.1.

### 3.3.2 Equations of motion

#### 3.3.2.1 Single stance phase (continuous dynamics)

Similar to section 2.3.1.2, we obtain Equation 3.1 and Equation 3.2 defined below by taking moments about stance foot contact point and hip hinge respectively, and non-dimensionalizing

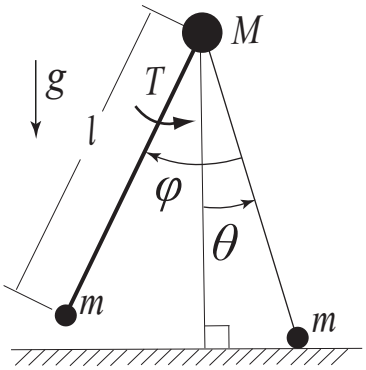


Figure 8: Model: The simplest walking model on level ground (Kuo, 2002).

time with $\sqrt{\ell/g}$ and applying the limit, $m/M \to 0$. In Equation 3.2, $\tau$ is the non-dimensional torque obtained by dividing the torque, $T$, by $Mg\ell$. The non-dimensional spring constant is $k$ and is obtained by dividing K by $Mg\ell$. The equations are;

$$\ddot{\theta} = \sin(\theta), \tag{3.1}$$

$$\ddot{\phi} = \sin(\theta) + \{\dot{\theta}^2 - \cos(\theta)\}\sin(\phi) - k\phi + \tau. \tag{3.2}$$

#### 3.3.2.2 Foot-strike phase (discontinuous dynamics)

Similar to section 2.3.1.4, in this phase of motion, the legs exchange their roles. The swapping of legs is expressed by Equation 3.3 and Equation 3.4. The angular rates of the legs after support exchange are given by Equation 3.5 and Equation 3.6. In the equations below, the state variables before and after collision are denoted using the superscript $-$ and $+$ respectively.

$$\theta^+ = -\theta^-, \tag{3.3}$$

$$\phi^+ = -\phi^-, \tag{3.4}$$

$$\dot{\theta}^+ = \dot{\theta}^- \cos\phi^- + P\sin\phi^-, \tag{3.5}$$

$$\dot{\phi}^+ = (1 - \cos\phi^-)(P\sin\phi^- + \dot{\theta}^- \cos\phi^-). \tag{3.6}$$

#### 3.3.2.3 Mid-stance event

The mid-stance event occurs when the stance leg is normal to the ground and is given by,

$$\theta = 0. \tag{3.7}$$

#### 3.3.2.4 Collision event

Similar to section 2.3.1.3, the collision event is given by,

$$\cos(\phi^- - \theta^-) - \cos(\theta^-) = h. \tag{3.8}$$

#### 3.3.2.5 Failure modes

The failure modes for the walker are similar to those of the simplest walker described in section 2.3.2.

## 3.4 Methods

### 3.4.1 Overview of control technique

From Equation 3.1 and Equation 3.2, we observe that: (1) the swing leg can be controlled by the hip torque and (2) the motion of the stance leg is independent of the motion of the

swing leg. We can find a function, $F$, that maps the mid-stance between consecutive steps and is indexed by step number, $k$. Thus

$$\dot{\theta}_{k+1} = F(\dot{\theta}_k, P_k, \phi_k^-) \tag{3.9}$$

where $\phi_k^-$ is swing leg angle at foot-strike at step k and is related to the step length, and $P_k$ is impulsive push-off at step k. Given the measurement at step k, $\dot{\theta}_k$, the control variables, $\phi_k^-$ and $P_k$, can be used to modulate $\dot{\theta}_{k+1}$.

### 3.4.2 Mid-stance to mid-stance map for stance leg:

Doing an energy balance for the stance leg between the mid-stance and the instant just before foot-strike and then again from just after foot-strike to the next mid-stance leads to Equation 3.10 and Equation 3.11 respectively

$$\frac{(\dot{\theta}_k)^2}{2} + 1 = \frac{(\dot{\theta}^-)^2}{2} + \cos\left(\frac{\phi^-}{2}\right), \tag{3.10}$$

$$\begin{aligned} \frac{(\dot{\theta}_{k+1})^2}{2} + 1 &= \frac{(\dot{\theta}^+)^2}{2} + \cos\left(\frac{\phi^+}{2}\right), \\ &= \frac{(\dot{\theta}^- \cos\phi^- + P\sin\phi^-)^2}{2} + \cos\left(\frac{\phi^-}{2}\right). \end{aligned} \tag{3.11}$$

To get the Equation 3.11, we assumed that the controller unaware of the step up/down disturbance, so we set $h = 0$.

### 3.4.3 OCLF and one-step dead-beat control

We use the orbital control Lyapunov function (OCLF) developed in chapter 2 to create an exponential stabilizing controller. The key idea is to create a control Lyapunov function at an event (e.g., mid-stance) and use it to regulate the velocity of the stance leg at a predefined convergence rate.

A nominal limit cycle is specified by setting $\dot{\theta}_{k+1} = \dot{\theta}_k = \dot{\theta}_0$, $P_k = P_0$, and $\phi_k^- = \phi_0^-$ in Equation 3.9 to get

$$\dot{\theta}_0 = F(\dot{\theta}_0, P_0, \phi_0^-) \tag{3.12}$$

The OCLF is defined as follows

$$V(\Delta\dot{\theta}_k) = \Delta\dot{\theta}_k^2 = (\dot{\theta}_k - \dot{\theta}_0)^2, \tag{3.13}$$

The following condition is required to make the system exponentially stable

$$V(\Delta\dot{\theta}_{k+1}) - V(\Delta\dot{\theta}_k) = -cV(\Delta\dot{\theta}_k), \tag{3.14}$$

where $c$ is a positive constant such that, $0 < c < 1$ for exponential stabilization and $c = 1$ for one-step dead-beat stabilization. Using Equation 3.13 in Equation 3.14 yields

$$\Big(\dot{\theta}_{k+1} - \dot{\theta}_0\Big)^2 - (1-c)\Big(\dot{\theta}_k - \dot{\theta}_0\Big)^2 = 0. \tag{3.15}$$

### 3.4.4 Two control techniques

#### 3.4.4.1 Foot-placement control with fixed push-off

We use the foot-placement to control the stance leg velocity between consecutive steps by fixing the impulsive push-off at $P_0$. Thus Equation 3.9 can be written as

$$\dot{\theta}_{k+1} = F(\dot{\theta}_k, P_0, \phi_k^-) \tag{3.16}$$

Substituting Equation 3.16 for $\dot{\theta}_{k+1}$ in Equation 3.15 leads to the condition for exponential stability

$$\Big(F(\dot{\theta}_k, P_0, \phi_k^-) - \dot{\theta}_0\Big)^2 - (1-c)\Big(\dot{\theta}_k - \dot{\theta}_0\Big)^2 = 0. \tag{3.17}$$

The stabilization problem for this case can be stated as follows: For the limit cycle parametrized by the stance leg nominal velocity at mid-stance, $\dot{\theta}_0$, a fixed rate of convergence specified by $c$, the stance leg velocity measured at mid-stance $\dot{\theta}_k$, and impulsive push-off maintained at its nominal value, $P_0$, find the foot-placement angle $\phi_k^-$ so that the Equation 3.17 is met.

#### 3.4.4.2 Push-off control with fixed foot placement

We use the impulsive push-off to control the stance leg velocity between consecutive steps by fixing the foot-placement at $\phi_0^-$. Thus Equation 3.9 can be written as

$$\dot{\theta}_{k+1} = F(\dot{\theta}_k, P_k, \phi_0^-) \tag{3.18}$$

Substituting Equation 3.18 for $\dot{\theta}_{k+1}$ in Equation 3.15 leads to the condition for exponential stability as

$$\Big(F(\dot{\theta}_k, P_k, \phi_0^-) - \dot{\theta}_0\Big)^2 - (1-c)\Big(\dot{\theta}_k - \dot{\theta}_0\Big)^2 = 0. \tag{3.19}$$

The stabilization problem for this case can be stated as follows: For the limit cycle parametrized by the stance leg nominal velocity at mid-stance, $\dot{\theta_0}$, a fixed rate of convergence specified by $c$, the stance leg velocity measured at mid-stance $\dot{\theta_k}$, and foot-placement angle maintained at its nominal value $\phi_0^-$, find the control action $P_k$ so that the Equation 3.19 is met.

### 3.4.5 Hip torque control for foot placement

The controller for the hip torque is defined similar to the one described in section 2.4.3. The time from mid-stance to instant just before foot-strike ($T_{\text{mid-fs}}$) and from instant after foot-strike to mid-stance ($T_{\text{fs-mid}}$) are computed as follows

$$T_{\text{mid-fs}} = \int_{\theta=\frac{\phi^-}{2}}^{\theta=0} \frac{d\theta}{\sqrt{(\dot{\theta}_k)^2 + 2(1-\cos(\theta))}}$$

$$T_{\text{fs-mid}} = \int_{0}^{\theta=\frac{\phi^-}{2}} \frac{d\theta}{\sqrt{(\dot{\theta}^+)^2 + 2(\cos(\frac{\phi^-}{2}) - \cos(\theta))}}$$

The time-based trajectory is supplemented with a proportional derivative controller to ensure good tracking performance.

## 3.5 Results

### 3.5.1 Nominal limit cycle

We choose average human walking kinematics to create the nominal limit cycle. The non-dimensional average human speed is 0.41, step length is 0.73, and consequently the step time is 1.78 (Bhounsule, 2014). The spring constant $k$ is chosen to be 2.11 in order to correspond to average human swing frequency. Next, we find the impulsive push-off and fixed point that give a passive leg swing. The nominal limit cycle values are: mid-stance robot state, $\{\theta_0, \dot{\theta}_0, \phi_0, \dot{\phi}_0\} = \{0, -0.3686, 0.0063, -1.2495\}$, impulsive push-off, $P_0 = 0.22$, and foot placement angle, $\phi_0^- = -0.7802$.

### 3.5.2 Rate of convergence

To demonstrate the rate of convergence we perturb the mid-stance speed, $\dot{\theta}_k = -0.7$. Then we run the one-step dead-beat controller, $c = 1$, and OCLF controller for $c = 0.25$ and $c = 0.5$. Figure 9 shows a plot of the mid-stance velocity as a function of the steps. As expected, the one-step dead-beat controller is able to regulate the velocity in a single step while the OCLF controller shows exponential convergence to the nominal limit cycle.

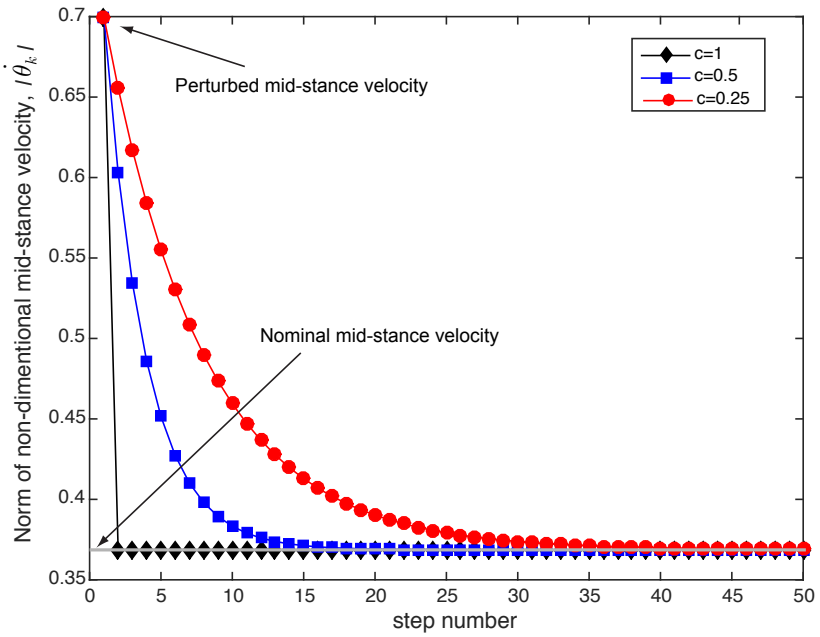


Figure 9: Rate of stabilization: The convergence is exponential for $0 < c < 1$ and "one-step dead-beat" for $c = 1$.

### 3.5.3 Robustness

#### 3.5.3.1 Simulation details

In order to assess robustness of our controller, we compute the average number of steps that the robot can withstand using either controllers, foot-placement control with fixed push-off and push-off control with fixed foot placement, without falling down on uneven terrain chosen from a random distribution with a deviation of $\sigma$. Note that $\sigma$ can be interpreted as a maximum step down ($+\sigma$) or step up ($-\sigma$ ) normalized by the leg length. We create 10 terrains with 400 steps, selected from a random distribution with standard deviation of $\sigma$. For each terrain, we do forward simulations of the system for three values of $c$. We evaluate the average number of steps to failure; control inputs $\phi^-$ and $P$; and average Cost of Transport (COT) defined as the energy used per unit weight per unit distance travelled.

#### 3.5.3.2 Average number of steps to failure

Figure 10 shows the average number of steps walked before failure as a function of deviation in terrain height $\sigma$ for three values of $c$. The top plot corresponds to foot placement control, $\phi^-$, with constant push-off, $P_0$ (see section 3.4.4.1), while the bottom plot corresponds to push-off control, $P$, with foot placement held constant, $\phi_0^-$ (see section 3.4.4.2). Since we limited the steps to 400 in the simulation, walking for 400 steps corresponds to maximum robustness. As $\sigma$ increases from 0 to 0.05 (step down) or decreases from 0 to $-0.05$ (step up), the average steps to failure decreases as expected. The plots indicate the robustness is same for different $c$ values for step down but marginally better for higher $c$ values for step up. The similar robustness for different $c$ values is because we did not enforce actuator limits. We expect that when actuator

limits are enforced, more aggressive control (higher $c$ values) will show lower robustness. The average steps profile is not symmetric about the y-axis; the drop in average steps walked is more gradual for step-down and more dramatic for step-up. This is due to the asymmetry in the hip controller: during step down the model has enough time to complete the step as planned but during step up the robot takes a shorter step because the robot takes a premature step increasing the chance of failure.

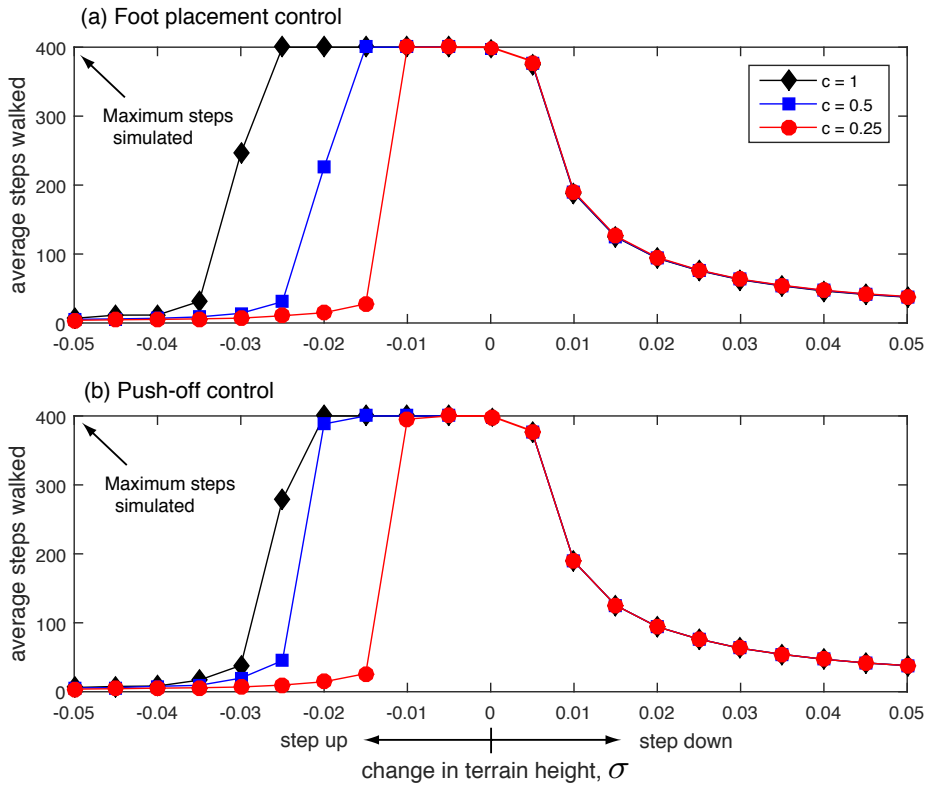


Figure 10: Robustness to step-down and step-up disturbance: (a) Average steps to failure using foot placement control with push-off held constant, and (b) Average steps to failure using push-off control with foot placement held constant.

#### 3.5.3.3 Control strategy

Figure 11 depicts the control strategy used for the robustness test shown in Figure 10. The top plot shows the swing-leg angle at foot-strike for foot placement control strategy with push-off held constant to its nominal value while the bottom plot shows the impulsive push-off for push-off control strategy with swing-leg at foot-strike held constant at its nominal value.

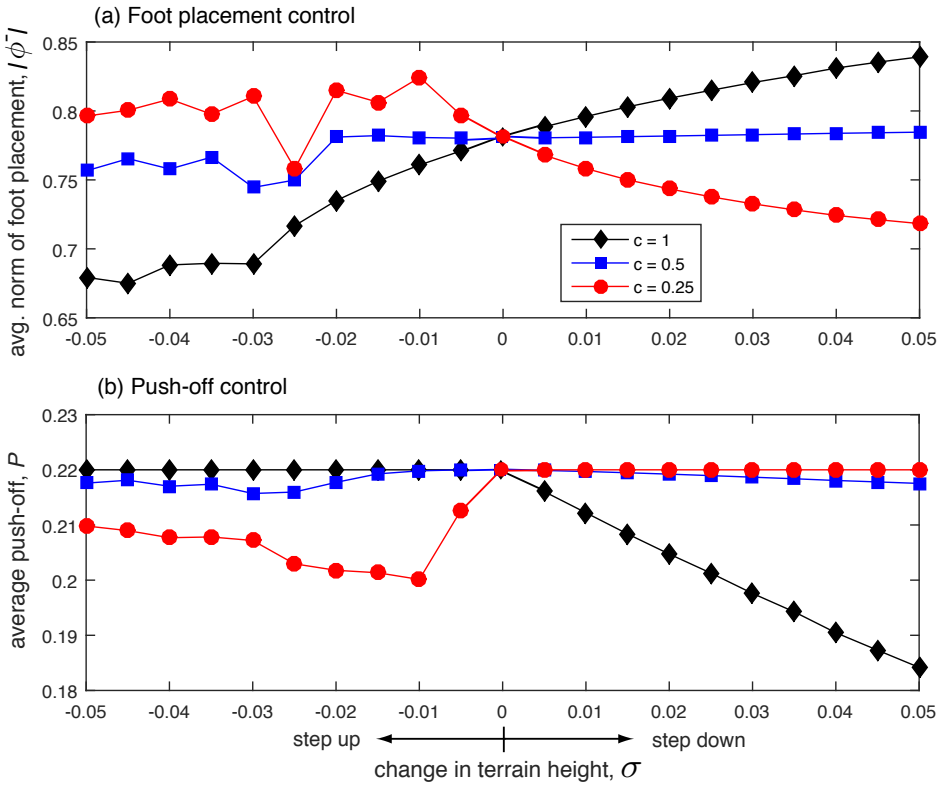


Figure 11: Control strategy for robustness test: (a) Average norm of swing-leg angle at foot-strike for foot placement control with push-off held constant. (b) Average push-off with swing-leg angle at foot-strike held constant. The average is taken over all the 10 runs and across the complete terrain for given deviation, $\sigma$.

We explain these plots by looking at the response for $c = 1$ or one-step dead-beat control (black diamond). For foot placement control, the swing leg angle changes from smaller than nominal to larger than nominal as the terrain height changes from step up to step down. This can be understood as follows. The model's energy is lower than nominal for step up and higher than nominal for step down. To regulate the energy for a fixed push-off, the swing leg angle needs to monotonically increase. Indeed, energy loss at foot-strike is directly proportional to foot placement angle (Ruina et al., 2005). However, for push-off control for $c = 1$, the push-off is almost constant for step up and it decreases monotonically for step down. Since push-off work is directly proportional to added energy, a decrease in the push-off allows the robot to maintain its speed while walking down. Thus we would expect the push-off to increase for step up but it remains constant. This is because the step up introduces an unexpected disturbance and the model takes a short step which has the effect of supplying energy to the system (note that foot placement angle is directly proportional to the energy loss).

The plots for $c = 0.25$ and $c = 0.5$ can be understood from the plot for $c = 1$. For example, consider the foot placement control for step down (top-right figure). The swing-leg angle is almost constant for $c = 0.5$ (blue squares) and decreasing for $c = 0.25$) (red circles). These values of $c$ require the robot to dissipate the excess energy gained through step down more gradually than $c = 1$. The result for $c = 0.5$ is that the model needs to just maintain its nominal swing leg angle and for $c = 0.25$ the model needs to decrease the swing leg angle in order to dissipate the energy more slowly to satisfy the rate of convergence given by $c = 0.25$.

#### 3.5.3.4 Energy usage

Figure 12 and Figure 13 show the average Cost Of Transport defined as the energy used per unit weight per unit distance traveled based on hip work and push-off work, respectively. We used the absolute value of mechanical work for the energies for the hip and ankle COT. Since the hip is massive and legs are light it is not meaningful to sum up the COT of hip and push-off work.

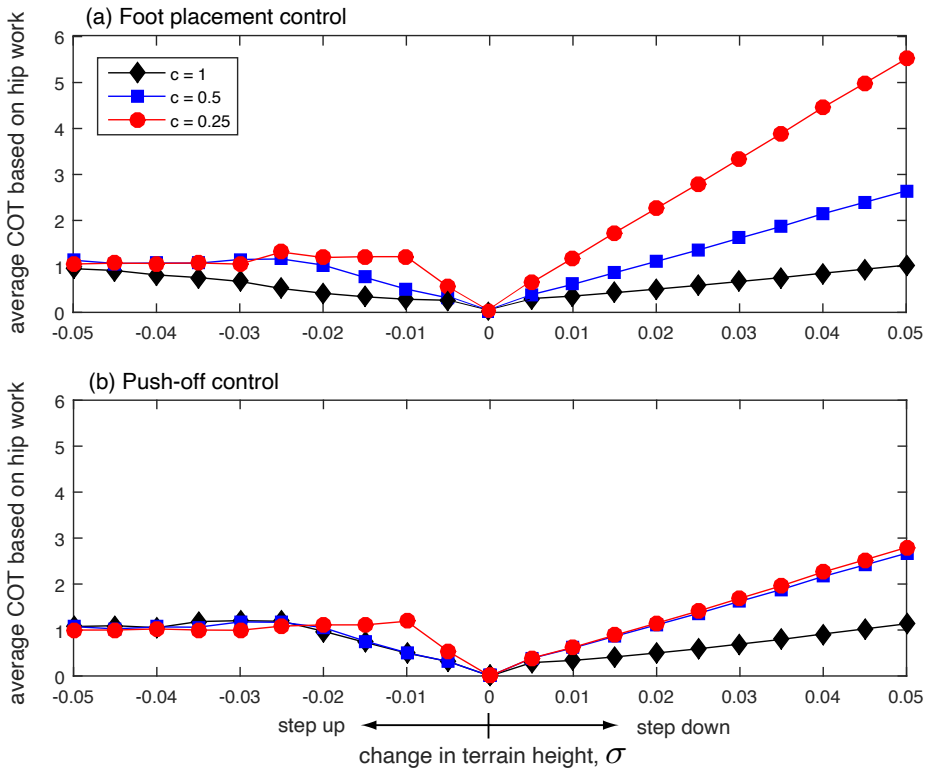


Figure 12: Average COT for robustness based on hip work: (a) Foot placement control (b) Push-off control.

The COT for the hip work (Figure 12) increases as $\sigma$ deviates from 0. Note that the COT for hip work for $\sigma = 0$ is zero because the nominal limit cycle has a passive leg swing. The increase in COT with $\sigma$ on the top plot for foot placement is obvious; the swing-leg angle needs to change for foot placement control, thus torque needs to be supplied and work needs to be done. The increase in COT with $\sigma$ on the bottom plot for push-off control for constant swing-leg angle is slightly more subtle. When the model is subjected to varying terrain heights, the speed changes but so does the time to foot-strike. The hip torque then needs to do work to move the leg faster or slower as needed even though the final swing-leg angle is the same, thus requiring more work.

The COT for push-off work (Figure 13) shows that in general, push-off COT decreases for step down and increases for step up with some exceptions. For $c = 0.25$, step down, foot placement control (top plot, right side, red circles), the COT for push-off increases. As noted earlier, this is to regulate the energy decay at a slower rate than the natural decay rate. A similar trend and reasoning apply for $c = 0.25$, step up, push-off control (bottom plot, right side, red circles).

## 3.6 Discussion

We have presented an analysis of two disparate control strategies, foot placement and ankle push-off, to regulate robot velocity between steps. We compared one-step dead-beat stabilization with exponential stabilization. The approach was tested by doing a forward simulation

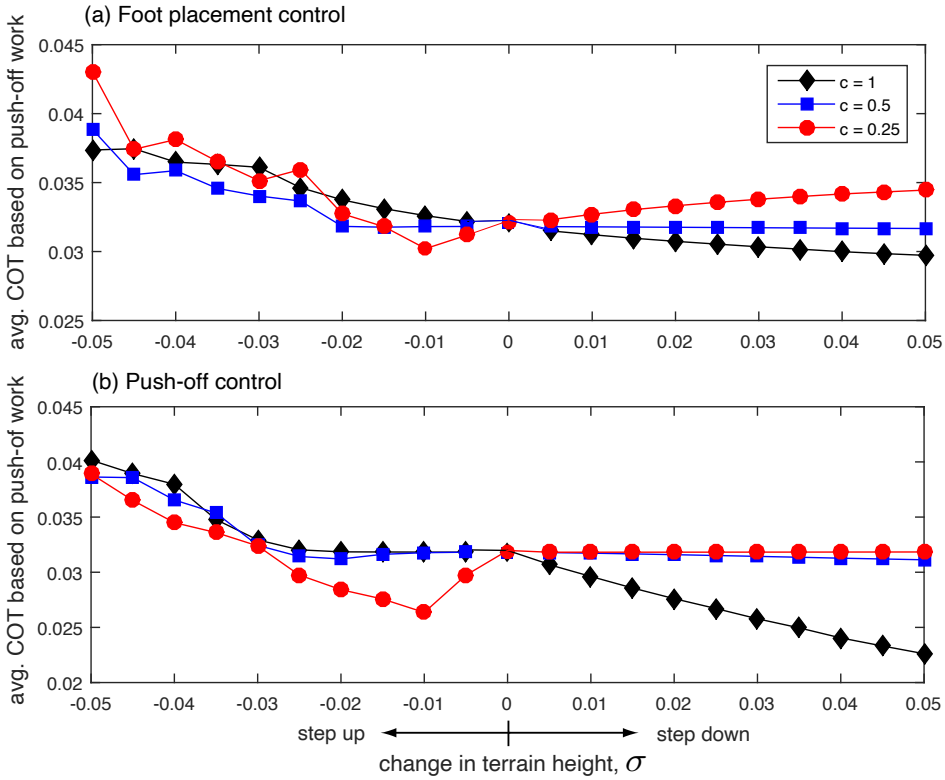


Figure 13: Average COT for robustness test based on push-off work: (a) Foot placement control (b) Push-off control.

with the terrain height changing monotonically, i.e., either step up or step down. Our findings are as follows:

1. Average number of steps to failure is the same for both control strategies and stabilization protocols. This is because of lack of actuator limits.

2. In general, average number of steps walked for step down disturbance is more than that for step up. This is because in our simulation, step up leads to premature stepping, thus throwing off the planned foot placement location.

3. Control strategy for fast convergence to the nominal for step down is to increase the foot placement angle or to decrease ankle push-off, while for step up is to decrease foot placement or maintain same ankle push-off.

4. For both control strategies, the mechanical energy usage (as measured by COT) for ankle push-off and foot placement is almost the same, except for $c = 0.25$ where hip COT for push-off control is half of that of foot placement control for step down. However, ankle COT for step down is less than step up while hip COT for step up is less than step down. Thus energy-wise, foot placement control is ideal for step up and ankle push-off for step down.

5. Overall, one-step dead-beat stabilization is more energy-efficient in terms of hip work and ankle work than exponential stabilization.

We found that ankle push-off and foot placement have similar robustness as measured by the steps to failure on changing terrain. This is in contrast to the work by Kim and Collins (Kim and Collins, 2017) who found that push-off is at least twice as effective as foot placement. We suspect that this is in part due to the fact that we have no actuator limits in our simulation allowing the robot to do drastic corrections to regulate walking. Similarly we found that the energy usage is similar for both control strategies except for slow convergence to the nominal

limit cycle. However, it is more economical to use ankle push-off for step down and foot placement for step up.

We also compared full stabilization of disturbances in a single step, also called one-step dead-beat control ($c = 1$), with exponential stabilization of disturbances ($0 < c < 1$), and found that although both have similar robustness as measured by average steps to failure, full correction of disturbances is more energy-efficient. The latter is due to the fact that although fast convergence to the nominal limit cycle is more expensive in the short-term (due to faster control actions), it has long-term advantages of being on the energy-efficient nominal limit cycle.

# CHAPTER 4

# NONLINEAR CONTROL POLICIES USING DATA-DRIVEN TECHNIQUES FOR ENLARGING REGION OF ATTRACTION OF LIMIT CYCLE OF LEGGED ROBOTS

(A significant portion of this chapter is reproduced from two published papers with the following citations:

Zamani, A. and Bhounsule, P. A. Control synergies for rapid sta- bilization and enlarged region of attraction for a model of hopping. Biomimetics, 3(3):25, 2018.

Bhounsule, P. A., Zamani, A., Krause, J., Farra, S., and Pusey, J.: Control policies for a large region of attraction for dynamically balancing legged robots: a sampling- based approach. Robotica, pages 116, 2020.)

## 4.1 Introduction

Dynamically balancing legged robots are characterized by a small footprint and thus have to constantly move to stay balanced. The most well-studied approach to controlling these robots is to first find a periodic trajectory (Hobbelen and Wisse, 2007b), and second, develop a feedback control policy to stabilize the periodic motion. Also, one may estimate the set of system states that converge back to the periodic motion, known as the region of attraction (ROA), to provide formal stability certificates.

The conventional approach for finding the ROA of a periodic motion is shown in Figure 14 (a). Given a periodic motion, a linear control policy (e.g., linear quadratic regulator (Tedrake, 2009)) is used for stabilization. The use of a linear control policy allows one to use convex optimization tools such as sum-of-squares optimization to find a Lyapunov function and the corresponding ROA (Prajna et al., 2002). One issue with the approach is that a linear control policy may be too restrictive and lead to a small region of attraction, this is especially true for highly nonlinear systems.

We propose a different approach that involves inverting the conventional approach and is shown in Figure 14 (b). We assume a candidate Lyapunov function and a ROA (typically a big region) and then find a control policy (usually a non-linear one) by performing trajectory optimization for sampled initial conditions. We demonstrate that a higher order polynomial or a neural network-based control policy is able to guarantee the ROA. Another advantage of our method, specifically when applied to periodic systems, is that we recast the trajectory tracking problem into a regulation problem by defining the ROA at the Poincaré section, thus keeping the method computationally inexpensive. The net result is an enlarged region of attraction with reasonable computational cost. Once the regions of attraction are computed for multiple cyclic or steady-state gaits, they may be sequenced together by reasoning about overlapping regions to create non-steady or agile gaits using heuristics (see chapter 5) or more systematic motion planning algorithms such as rapidly exploring random trees (see chapter 6).

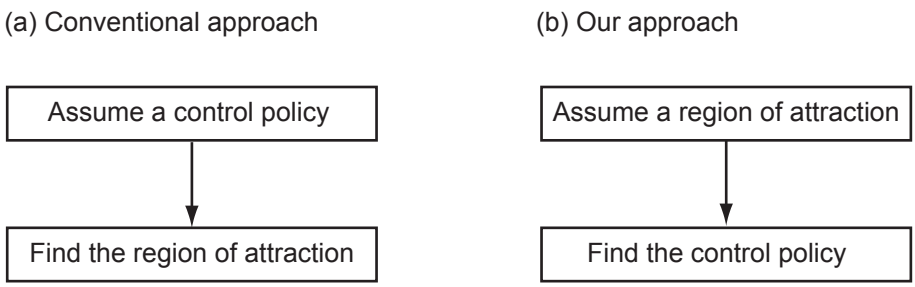


Figure 14: Types of control approaches: (a) The conventional approach assumes a control policy (usually a linear policy) followed by optimization to find the largest region of attraction (e.g., using sum-of-squares optimization to find the largest level set for the Lyapunov function). (b) In our approach, we assume a region of attraction followed by optimization to find the control policy (a higher order polynomial or neural network control policy gives good results for non-linear systems). Our method uses sample-driven trajectory optimization followed by non-linear regression provide an enlarged region of attraction even for highly nonlinear systems.

## 4.2 Background and related work

The most simple dynamically balancing legged robots are those created by McGeer in the early 1990s (McGeer, 1990; McGeer, 1991). McGeer showed that by tuning the natural dynamics (mass distribution, geometry, foot shape, etc.), it is possible to get natural-looking walking motion on a shallow incline. This concept is known as passive dynamic walking. Garcia et al. (Garcia et al., 1998) simplified McGeer's model by using a point mass hip, massless feet, and straight legs. The resulting model, called the simplest walker, has a single free parameter, the ramp slope. They demonstrated that as the ramp slope increased, the model displayed period-doubling leading to chaos, a phenomenon seen in many other non-linear dynamical systems (Strogatz, 1994).

Although passive dynamic walkers are energetically efficient (energy cost of movement is zero), they are extremely fragile and knocked down by the slightest disturbance. For a dynamical system, the set of initial conditions that converge back to the periodic motion is called the basin of attraction (BOA) [A]. Schwab and Wisse (Schwab and Wisse, 2001) used the cell mapping method (Hsu, 2013) to find the BOA of the simplest walker. The cell mapping method relies on extensive sampling and forward simulations to estimate the BOA of the system. First, the state space is divided into cells. Then initial conditions are chosen in the middle of the cells. For each initial condition, the system is simulated to check if it converges back to the periodic motion. The process is repeated for all initial conditions chosen on the cells. The resolution of the cells determines the accuracy of the method. A finer resolution increases accuracy but also increases the computational cost. It was found that the BOA for the simplest walker is very narrow, which is understandable given that the system is completely passive. A similar brute force approach was used by Heim and Sprowitz (Heim and Spröwitz, 2019) to find the BOA of the spring-loaded inverted pendulum (SLIP) model of hopping. They have shown that a non-linear spring increases the BOA over the linear spring model. However, unlike the passive dynamic walkers, the SLIP model uses active control of the foot placement angle before touchdown.

A broader definition of stability is captured by the viability theory, which defines the viability kernel defined as the set of all states from which it is possible to avoid falling down (Wieber,

---

[A] ROA is an estimate of the basin of attraction.

2008). Unfortunately, it is computationally challenging to find this set even for simplest legged systems. A computationally tractable approach was taken by Pratt et al. (Pratt et al., 2006) by defining a capture region, which is the region where the robot needs to step to come to a complete stop in one or more steps. The use of simple models of locomotion such as linear inverted pendulum model enables easy computation of the capture region and subsequent implementation in hardware (Pratt et al., 2012). Zaytsev et al. (Zaytsev et al., 2018) did a broader analysis using a simple model of locomotion that generalizes capture regions to a broader set of targets rather than just standing still and in the spirit of viability theory, computes the states and controllers that avoid falling.

A global method for finding control policy and region of attraction is to use dynamic programming. Dynamic programming returns a global policy and the value function or the cost of executing the control policy. The issue with this approach is the method suffers from the curse of dimensionality, that is, as the system complexity grows, the storage and computations grow exponentially. The common way to deal with this issue is to reduce the system to a simple abstraction, also known as a template (Koditschek and Robert, 1999), and use it for controller design. For example, Whitman (Whitman, 2013) simplified a humanoid robot into three simple independent models; a sagittal, a lateral, and a frontal model. Controllers were developed for the separate models independently, but combined during implementation using time as the phase variable. Another simplification approach taken by Mandersloot et al. (Mandersloot et al., 2006) is to use dynamic programming to choose states and control actions at the Poincaré section instead of the time trajectory.

Simulation-based tabulation of controllers followed by the online implementation is one way of getting past the computational burden. Raibert (Raibert and Wimberly, 1984) used simulations to tabulate the controls as a function of the system state. Then the polynomial surface was used to fit the table. This allowed efficient storage of the table and for real-time implementation. In a similar vein, Da et al. (Da et al., 2017) used simulation to create controllers as functions of system states and terrain height. The data was then inputted into a supervised learning framework to learn a policy, which was then implemented in hardware. Our method is similar to the latter; we generate state, control pairs using optimization followed by learning a function from the states to the controls using regressors such as a deep neural network.

Local control methods involve creating a local control policy for the stabilization of the periodic motion. For example, Tedrake et al. (Tedrake, 2009) used a Linear Quadratic Regulator (LQR) to stabilize the periodic motion. Then, using the LQR cost as a value function in combination with sum-of-squares optimization (Prajna et al., 2002), they found a Lyapunov function and the corresponding region of attraction. The work by Manchester et al. (Manchester et al., 2010) split the dynamics into transverse and tangential components and then searched a Lyapunov function and region of attraction on the transverse dynamics using sum-of-squares optimization. These approaches lead to the creation of the region of attraction as a tube along the trajectory.

In our work, we are primarily interested in stabilizing a periodic motion. Our work is different from past work in several ways:

1. We define the Lyapunov function and the region of attraction at the Poincaré section (see OCLF in chapter 2). Thus we are solving a regulation problem and the region of attraction is a single surface of dimension $n-1$ ($n$ is the number of states of the system) at the Poincaré section compared to the tracking problem and the region of attraction for $t$ samples along the trajectory, which will be $n \times t$ states which define a tube along the trajectory (see Tedrake (Tedrake, 2009)).
2. The control policies we find allow exponential convergence to the periodic motion using an orbital Lyapunov function. Previous methods based on LQR generate asymptotic convergence to the periodic motion, which is relatively slower.
3. Our method can work with black-box models of the system to find a nonlinear control policy. This is in contrast to past approaches that need a specific structure in the model and/or controllers (e.g., polynomial models for sum-of-squares optimization).
4. Our method is able to find a relatively larger region of attraction compared to past approaches, but at the expense of finding a more complex control policy (e.g., nonlinear control policy).

## 4.3 Methods

The details of our approach are shown in Figure 15 and are elaborated in this section. First, as shown in Figure 15 (a), we define preliminaries needed for analysis of cyclic gaits. Second, as shown in Figure 15 (b), we assume a region of attraction and use an exponential control orbital

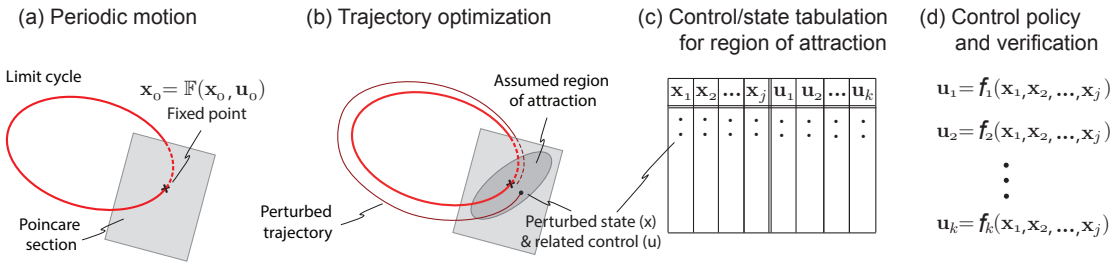


Figure 15: Overview of our sampling-based optimization approach: (a) A gait that is periodic repeats itself in one or more steps, (b) a region of attraction is assumed at the Poincaré section and control actions (set-points, gains, amplitudes, etc) are found that lead to exponential convergence to the periodic motion at the section, (c) trajectory optimization is repeated for sampled points in the region of attraction to generate a look-up table for initial states and corresponding control actions, (d) regression is performed to fit a control policy for each control action as a function of state followed by verification.

Lyapunov function within a trajectory optimization framework for controller design. Third as seen in Figure 15 (c), the results of trajectory optimization on samples chosen in the region of attraction are tabulated. Finally, as shown in Figure 15 (d), we obtain a control policy from the tabulated data and followed by extensive verification of the control policy by simulating the system under external disturbances and modeling errors.

### 4.3.1 Periodic motion

In this paper, we are interested in a one-step periodic motion, which is a movement pattern that repeats itself after a single step (see Figure 15 (a)). A Poincaré section is an instant in the

gait cycle (e.g., mid-stance, foot-strike). A Poincaré map $F$ is a function that maps the state at the Poincaré section to itself after one step and is given by

$$\mathbf{x}_{i+1} = F(\mathbf{x}_i, \mathbf{u}_i), \tag{4.1}$$

where $i$ is the step number, $\mathbf{x}_i$ are the states at step $i$, and $\mathbf{u}_i$ are the control actions at step $i$. In order to find a periodic motion, for a given nominal state $\mathbf{x}_0$ there is a corresponding nominal control $\mathbf{u}_0$ such that

$$\mathbf{x}_0 = F(\mathbf{x}_0, \mathbf{u}_0). \tag{4.2}$$

### 4.3.2 Trajectory optimization

First, we used the stability metric called the orbital control Lyapunov function developed in chapter 2 and then we formulate the trajectory optimization problem.

#### 4.3.2.1 Exponential convergence using orbital control Lyapunov function

We define a Lyapunov function ($V$) at the Poincaré section as follows

$$V(\Delta\mathbf{x}_i) = (\Delta\mathbf{x}_i)^T \mathbf{S}_0 \Delta\mathbf{x}_i = (\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \tag{4.3}$$

where $\mathbf{x}_0$ is the fixed point of the periodic motion, $\mathbf{x}_i \neq \mathbf{x}_0$ is a system state at the Poincaré section that needs to be stabilized, and $\mathbf{S}_0$ is a positive definite matrix that determines the shape of the region of attraction. The condition for exponential stabilization is

$$V(\Delta\mathbf{x}_{i+1}) - V(\Delta\mathbf{x}_i) \quad \leq -\alpha V(\Delta\mathbf{x}_i), \tag{4.4}$$

where $0 < \alpha < 1$ is the rate of decay of the Lyapunov function between steps. Thus, the condition for exponential stability can be rewritten in terms of control using Equation 6.3 and Equation 4.4 as follows

$$\begin{aligned}
& V(\Delta\mathbf{x}_{i+1}) - (1-\alpha)V(\Delta\mathbf{x}_i) \leq 0, \\
\implies \quad & (\mathbf{x}_{i+1} - x_0)^T \mathbf{S}_0 (\mathbf{x}_{i+1} - \mathbf{x}_0) - (1-\alpha)(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq 0, \\
\implies \quad & \Big(F(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_0\Big)^T \mathbf{S}_0 \Big(F(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_0\Big) - (1-\alpha)(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq 0.
\end{aligned} \tag{4.5}$$

Equation 4.5 is the condition on the orbital Lyapunov function for exponential orbital stabilization (step-to-step stabilization). Specifically, we select $\mathbf{u}_i$ such that the above condition is met. The variable $\alpha$ is set to 0.9 in all simulations. The rationale is that a value of 0.9 gives a good compromise between rate of convergence and robustness to modeling errors.

#### 4.3.2.2 Trajectory optimization problem formulation

The trajectory optimization is done for a given initial condition at the Poincaré section $\mathbf{x}_i \neq \mathbf{x}_0$ (see Figure 15 (b)) as follows

$$\underset{\mathbf{u}_i}{\text{minimize}} \qquad \text{Mechanical Cost Of Transport (MCOT)} = \frac{\text{Mechanical Work Per Step}}{\text{Weight} \times \text{Step Length}} \tag{4.6}$$

$$\text{subject to:} \qquad \mathbf{x}_{i+1} = F(\mathbf{x}_i, \mathbf{u}_i) \tag{4.7}$$

$$\Big(F(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_0\Big)^T \mathbf{S}_0 \Big(F(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_0\Big) - (1-\alpha)(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq 0. \tag{4.8}$$

In the above problem, the control actions $\mathbf{u}_i$ are control parameters that are set once-per-step. Some examples of control actions are: feedback gains, set-points, amplitude of suitable time-based functions. All these control actions, the number and type of actions, are designer's choice (also see Figure 17).

The optimization problem defined by Equation 4.6 - Equation 4.8 may be solved using parameter optimization using collocation or shooting methods (for a review see ref. (Betts, 1998)). We use a direct shooting method in all optimizations.

### 4.3.3 Control/state tabulation for the region of attraction

The Region Of Attraction (ROA) of the controller is the set of all initial conditions $\mathbf{x}_i$ that would converge to the fixed point, $\mathbf{x}_0$. As noted earlier, in our method, we start by assuming a region of attraction. Then, we sample the region of attraction to generate a set of initial conditions $\mathbf{x}_i$ as follows. We choose a level set of the Lyapunov function, $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) = c$, where $c$ is a constant and has a small value to start with. We choose $n$ equally spaced points

on this level set. This process is repeated for multiple level sets of increasing value for $c$ to generate the set of initial conditions $\mathbf{x}_i$. The control/state tabulation is done as follows. For each initial condition $\mathbf{x}_i$, we solve the trajectory optimization problem given in section 4.3.2.2 to obtain the corresponding control action $\mathbf{u}_i$. The control/state pairs are saved in a tabular format (see Figure 15 (c)).

### 4.3.4 Control policies and verification

Based on the control/state tabulation, we are interested in fitting a control law for each control action as follows

$$u_k = f_k(x_1, x_2, x_3, ..., x_J) \qquad \text{k = 1, 2,3,..,K} \tag{4.9}$$

where $k$ denotes the index for each control action (total $K$ actions) and $j$ indicates the index for each state (total $J$ states). The function $f$ is a designer's choice. We use linear, quadratic, and a neural networks in our results and compare the fit with each other by doing additional simulation using additional randomly chosen initial conditions. In addition, we simulate the system to disturbances and modeling errors to check the robustness of the control policy (see Figure 15 (d)).

### 4.3.5 Model of hopping

We demonstrate our method on a model of hopping shown in Figure 16 (a). The model consists of a point mass body with mass $m = 80\ kg$ and maximum leg length $\ell_0 = 1\ m$. Gravity points downwards and is denoted by $g = 9.81\ m/s^2$. There is a prismatic actuator that can

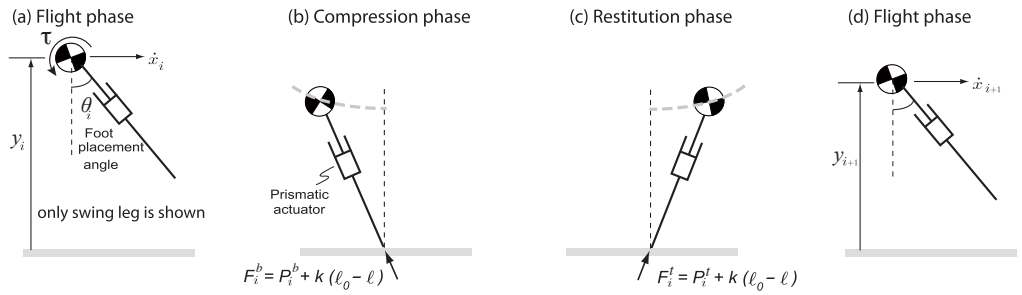


Figure 16: A complete step for the hopping model: The model starts in the flight phase at the apex position (vertical velocity is zero), followed by the stance phase, and finally ending in the flight phase at the apex position of the next step. The hopping model has a prismatic actuator that is used to provide an axial braking force $F^b$ in the compression phase and axial thrust force $F^t$ in the restitution phase, and a hip actuator (not shown) that can place the swinging leg at an angle $\theta$ with respect to the vertical as the leg lands on the ground.

generate an axial force $F$ along the leg and a hip actuator that can place the swing leg at an angle $\theta$.

The states of the model are given by $\{x, \dot{x}, y, \dot{y}\}$ where $x, y$ are the x- and y- position of the center of mass and $\dot{x}, \dot{y}$ are the respective velocities. A single step of the hopper shown in Figure 16 (a)-(d) is given by the following equation:

$$\underbrace{\text{Flight} \overset{\text{apex}}{\longrightarrow} \text{Flight} \overset{\text{touchdown}}{\longrightarrow} \text{Stance compression} \rightarrow \overset{\text{mid-stance}}{\longrightarrow} \text{Stance restitution} \rightarrow \overset{\text{takeoff}}{\longrightarrow} \text{Flight}}_{\text{one step/ one-step periodic motion}} \overset{\text{apex}}{\longrightarrow} \text{Flight}$$

We explain the above equation in detail next. The model starts at the apex (see Figure 16 (a)) where the state vector is, $\{x_i, \dot{x}_i, y_i, 0\}$. The model then falls under gravity,

$$\ddot{x} = 0, \qquad \ddot{y} = -g \tag{4.10}$$

till contact with the ground is detected by the condition $y - \ell_0 \cos(\theta) = 0$, where $\theta$ is the foot placement angle and measured relative to the vertical. Thereafter, the ground contact interaction is given by (see Figure 16 (b), (c)),

$$m\ddot{x} = F\frac{(x - x_c)}{\ell}, \qquad m\ddot{y} = F\frac{y}{\ell} - mg \tag{4.11}$$

where $x_c$ is the ground-foot contact point that needs to be set at every step depending on the ground-foot contact point at touchdown, $F > 0$ is the linear actuator force along the leg. For the first half of the stance phase from touchdown to mid-stance (defined by $\dot{\ell} = 0.$), called the compression phase, the actuator force is a braking force $F = F^b = P^b + k(\ell_0 - \ell)$. For the second half of the stance phase from mid-stance to take-off, called the restitution phase, the actuator force is a thrust force $F = F^t = P^t + k(\ell_0 - \ell)$. $P^b$ and $P^t$ are constant control forces during compression and restitution respectively. In the above equations $\ell = \sqrt{(x - x_c)^2 + y^2}$ is the instantaneous leg length measured relative to the contact point and $k$ is constant (fixed) gain analogous to the spring constant. In all simulations we take $k = 32,000\ N/m$. The take-off phase occurs when the leg is fully extended, that is, $\ell - \ell_0 = 0$. Thereafter, the model has a flight phase and ends up in the next apex state, $\{x_{i+1}, \dot{x}_{i+1}, y_{i+1}, 0\}$ (see Figure 16 (d)).

Figure 17 shows the block diagram of our controller. There are two control loops, an inner loop that does fast continuous control and an outer loop that does slow once-per-step control. The inner loop has a tracking controller that tracks the swing leg angle during flight phase as a function of time and the stance leg force during stance phase as a function of leg length.

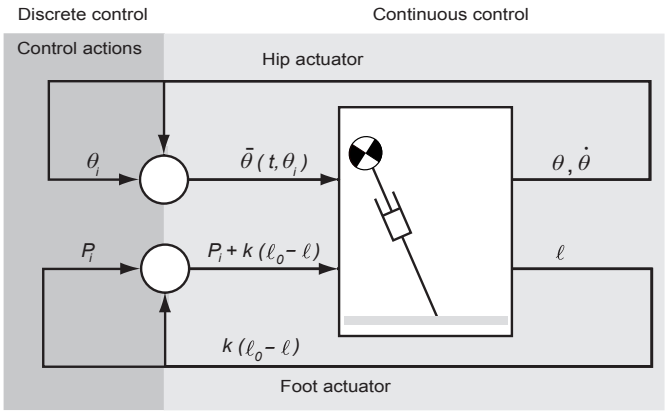


Figure 17: Block diagram of the controller: There are two control loops. A high bandwidth continuous control inner loop that does a time-based position control of the swing leg and force control of the stance leg, and a low bandwidth (once-per-step) discrete control outer loop that sets the foot placement angle ($\theta_i$) and thrust/braking force ($P_i = \{P_t, P_b\}$).

The slow loop sets the foot placement angle ($\theta_i$) and the stance leg force ($P_i = \{P^t, P^b\}$) once-per-step. This paper focuses exclusively on the outer, slow controller.

## 4.4 Results

### 4.4.1 Trajectory optimization for sampled points in the assumed ROA

We define the Poincaré map, $F$, at the apex. The apex is defined by the condition, $\dot{y} = 0$. Thus, the state at the apex at step $i$, $\mathbf{x}_i = \{\dot{x}_i, y_i\}$, and the control actions that are tuned once-per-step as discussed in section 4.3.5 are, $\mathbf{u}_i = \{\theta_i, P_i^b, P_i^t\}$ (where $\theta_i$, $P_i^b$ and $P_i^t$ are foot placement angle with respect to vertical, the constant forces during compression and restitution respectively (see Equation 4.11)). The step-to-step dynamics are given by $\mathbf{x}_{i+1} = F(\mathbf{x}_i, \mathbf{u}_i)$.

We do not have an analytical expression for $F$, but we build a simulation of the system using MATLAB's ordinary differential equation solver *ode113* with an integration tolerance of $10^{-9}$.

We arbitrarily chose the apex state for the periodic motion to be $\mathbf{x}_{i+1} = \mathbf{x}_i = \mathbf{x}_0 = \{5.0, 1.3\}$ (units of m/s and m respectively) and numerically evaluated the control $\mathbf{u}_i = \mathbf{u}_0 = \{\theta_0, P_0^b = 0, P_0^t = 0\}$ necessary to ensure periodic motion given by the condition $\mathbf{x}_0 = F(\mathbf{x}_0, \mathbf{u}_0)$. The control necessary to achieve the periodic motion is $\mathbf{u}_0 = \{0.3465, 0, 0\}$

Next, we show that using only foot placement control will limit the range of perturbation that can be stabilized. Figure 18 shows a plot of the vertical height at the apex ($y_i$) versus the horizontal velocity at apex ($\dot{x}_i$). The fixed points $\mathbf{x}_0$ is shown as point (a). Since the system is conservative (no dissipation), each fixed point lies on a constant total energy (TE) line that is found using the sum of potential and kinetic energy at the apex, $\text{TE}_i = 0.5m\dot{x}_i^2 + mgy_i$. The constant energy line corresponding to the fixed point is shown as $\text{TE}_0$. Because the system is conservative, only initial states on the constant total energy line converge back to the fixed point. The initial states on the constant $\text{TE}_0$ line that start at a higher height ($y_1 > y_0$) and a lower speed ($\dot{x}_1 < \dot{x}_0$) compared to nominal (such as point (b)), can converge to the fixed point by decreasing the foot placement angle ($\theta_1 < \theta_0$). The initial states on the constant $\text{TE}_0$ line that start at a lower height ($y_2 < y_0$) and a higher speed ($\dot{x}_2 > \dot{x}_0$) compared to nominal (such as point (c)), can converge to the fixed point by increasing the foot placement angle ($\theta_2 > \theta_0$). The initial states not on the constant $\text{TE}_0$ line (such as point (d)) cannot converge to the fixed point because there is no means of changing the total energy of the system. From this analysis, we observe that the foot placement angle is able to convert kinetic energy to potential energy

or vice versa while maintaining the total energy of the system. This limits the range of initial conditions that can be stabilized to be on the total energy line corresponding to the fixed point. Thus, the foot placement angle has a very narrow region of attraction.

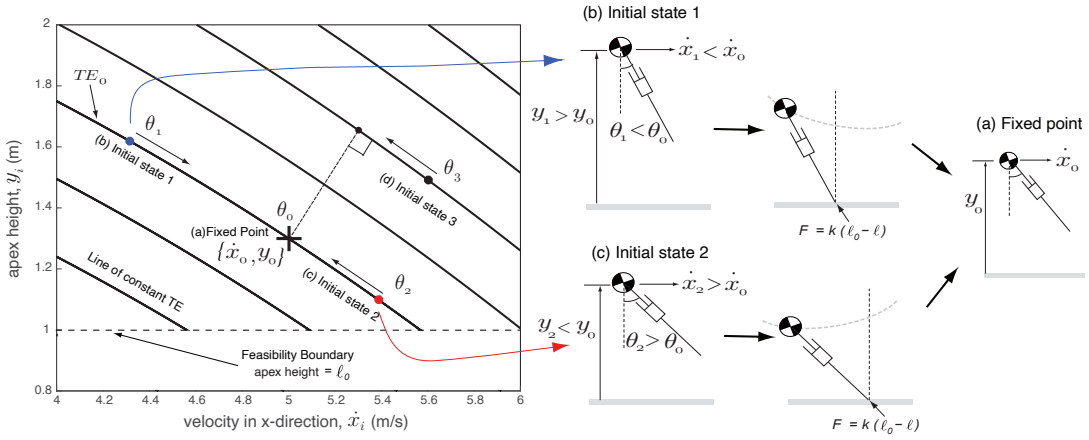


Figure 18: Control of foot placement angle: The plot shows the velocity in the $x$-direction ($\dot{x}_i$) versus the vertical height ($y_i$) at the Poincaré section, which is at the apex in the flight phase. (a) The fixed point is $\{\dot{x}_0, y_0\} = \{5, 1.3\}$, which corresponds to the foot placement angle $\theta_0 = 0.3465$ rad. (b) Initial states such as $\{\dot{x}_1, y_1\}$ need to decrease the foot placement angle $\theta_1$ to converge to the limit cycle. (c) Initial states such as $\{\dot{x}_2, y_2\}$ need to increase the foot placement angle $\theta_2$ to converge to the limit cycle. (d) Initial conditions such as $\{\dot{x}_3, y_3\}$ which are not on the $TE_0$ line cannot converge to the fixed point $\{\dot{x}_0, y_0\}$ because there is no means to change the total energy of the system (drawing not shown).

Next, for the trajectory optimization we chose the positive definite matrix for the Lyapunov function to be $\mathbf{S}_0 = diag\{1, 11.1\}$, exponential convergence rate $\alpha = 0.9$, and the region of attraction (ROA) to be $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) = 1$. These are design variables and the rationale for these choices is as follows: The $S_0$ in combination with the region of attraction $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}^i - \mathbf{x}_0) = 1$ ensures that speed variations within $\pm 1$ m/s and height variation $\pm 0.3$ m may be stabilized. The choice for $\alpha$ determines how fast the initial condition at the apex decays to the periodic motion. For $\alpha = 0.9$, all initial conditions would reduce by a factor of 90%.

We use the energy metric called the Mechanical Cost Of Transport (MCOT) defined as energy used per unit weight per unit distance travelled

$$\begin{aligned} \text{MCOT} &= \text{MCOT}^k + \text{MCOT}^b + \text{MCOT}^t \\ &= \frac{E^k}{mgD} + \frac{E^b}{mgD} + \frac{E^t}{mgD} \\ &= \frac{\int |k(\ell_0 - \ell)\dot{\ell}| dt}{mgD} + \frac{\int |P^b \dot{\ell}| dt}{mgD} + \frac{\int |P^t \dot{\ell}| dt}{mgD} \end{aligned} \tag{4.12}$$

where $|x|$ is the absolute value of $x$, $D$ is the step length, and $\dot{\ell} = \dfrac{(x - x_c)\dot{x} + y\dot{y}}{\ell}$. The absolute value is a non-smooth function of its argument, so we smooth it using square root smoothing (Srinivasan, 2006). That is, $|x| = \sqrt{x^2 + \epsilon^2}$ where $\epsilon$ is a small number (we set $\epsilon = 0.01$).

Finally, the optimization problem given by Equation 4.6 - Equation 4.8 is solved from initial conditions sampled on the region of attraction and described in detail in section 4.4.2.

### 4.4.2 Control/State tabulation

We generated 451 data points within the ellipse given by $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) = 1$ (see section 4.4.2) and performed the trajectory optimization for each sampled data point (see section 4.3.2.2). For the trajectory optimization, we recast the problem as a parameter optimization problem on the control actions and used a single shooting method to evaluate the cost function and system state at the end of a single step. The parameter optimization problem was solved using constrained optimization software called SNOPT (Gill et al., 2002), which is based on solving sequential quadratic programs. The total optimization time for 451 initial conditions was 131 minutes on a laptop (circa 2012). Thus, it took about 17 seconds for each optimization to complete.

The controls/state combinations are tabulated for further processing, but are presented here as plots. Figure 19 shows the three control actions as a function of apex state, the height $y$ and the horizontal velocity $\dot{x}$. The total energy at the apex is given by $\mathrm{TE}_i = 0.5m(\dot{x}_i)^2 + mgy_i$. The corresponding value for the periodic motion is given by $\mathrm{TE}_0 = 0.5m(\dot{x}_0)^2 + mgy_0$. Next, we find the total energy curve corresponding to the fixed point (shown as the dashed black line) by solving for $\dot{x}_i$ and $y_i$ on the curve $TE_0 = 0.5m(\dot{x}_i)^2 + mgy_i$. This curve divides the ellipsoids into two halves: top right half has higher total energy than the nominal, $\mathrm{TE}_i > \mathrm{TE}_0$, (where $i$ is an initial condition in the top right half); and bottom left half, has lower total energy than the nominal, $\mathrm{TE}_i < \mathrm{TE}_0$. Thus for top-right half the braking force $P^b$ is non zero and serves to brake or extract energy from the system. Similarly, for the bottom left half the control strategy is to apply a constant thrust force to add energy to the system. The foot

placement angle maintains the total energy of the system, but converts the potential energy to the kinetic energy and vice versa. The three control actions perform three distinct roles: foot placement cannot change the total energy, but can convert the potential energy to the kinetic energy and vice versa, the braking force can only decrease the total energy, and thrust force can only increase the total energy.

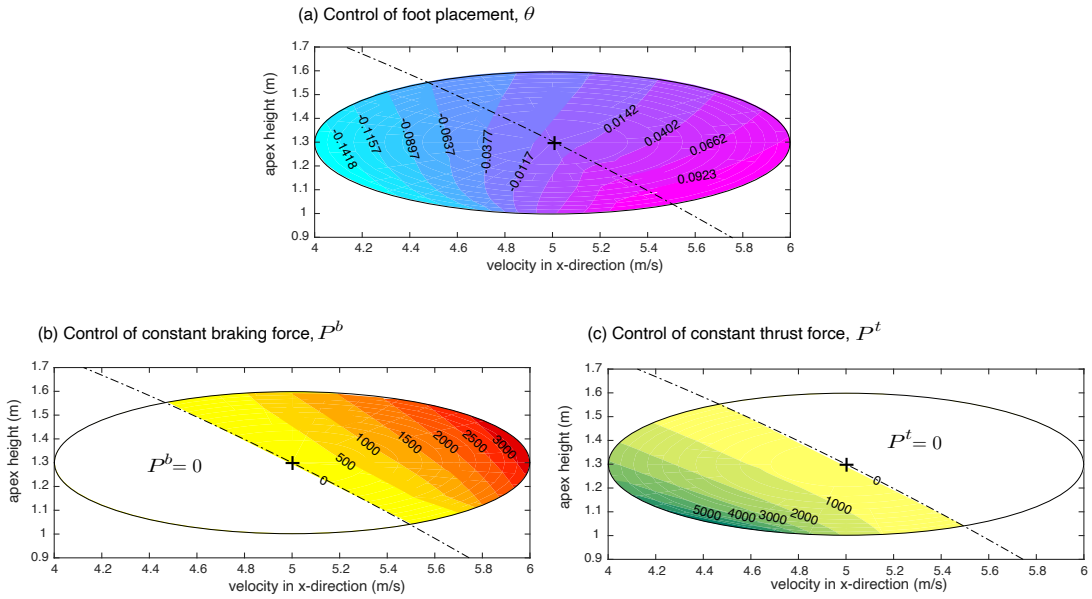


Figure 19: Contour plots for control actions as a function of horizontal velocity and apex height: (a) foot placement angle; (b) constant braking force in stance phase; and (c) constant thrust force in the restitution phase.

Figure 20 shows the associated Mechanical Cost Of Transport (MCOT) (see Equation 4.12) for individual control actions and the total. Figure 20 (a) shows the total MCOT. The total MCOT for the fixed point (shown by the + sign) is 0.39. The MCOT is mostly flat along the x-axis or the horizontal velocity axis, but increases monotonically along the y-axis or the vertical height axis. Figure 20 (b) - (d) plots the contribution of individual control actions while (a) is the sum of (b), (c), and (d).

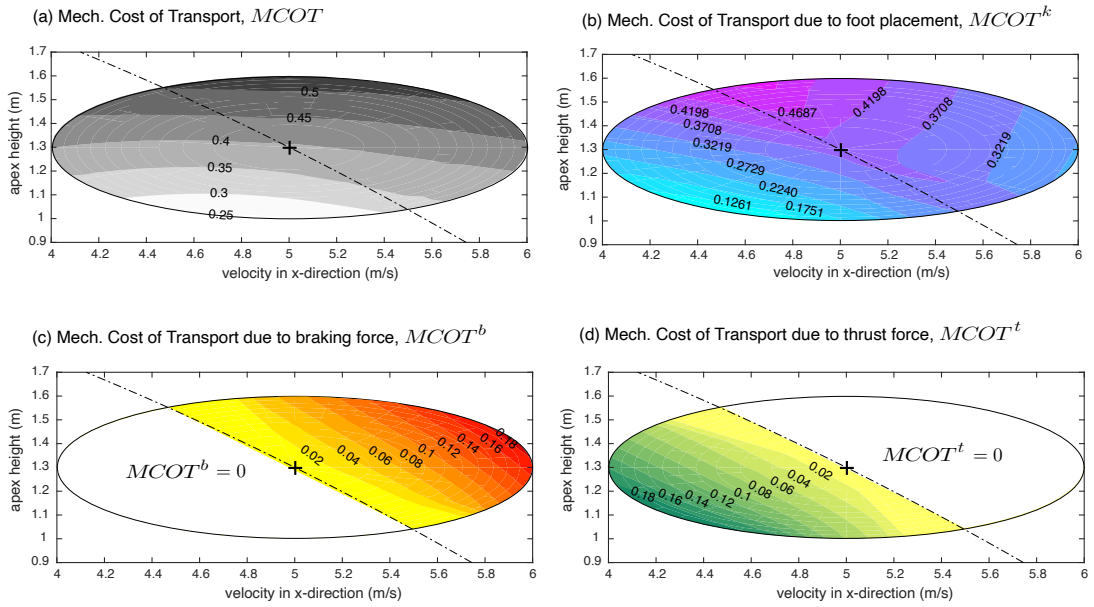


Figure 20: Contour plots for MCOT function of horizontal velocity and apex height: Mechanical Cost Of Transport MCOT (a) total, (b) due to springy leg force $k$ and foot placement angle $\theta$, (c) due to braking force $P^b$, and (d) due to thrust force $P^t$.

### 4.4.3 Finding control policies and verification

Next, we find a control policy ($u_k = f_k(\mathbf{x}_i)$, one for each control action $k$) from the tabulated data section 4.4.2. The rationale is that such a control policy offers a compact representation of the controls and is easier to store for hardware implementation.

We use the observation from Figure 19 that we can segregate force control ($P^b$ and $P^t$) based on the location of the initial condition and the total energy. We have the following three parameterizations in the increasing level of complexity (as given by the number of parameters).

#### 4.4.3.1 Linear policy

The linear policy has 9 parameters as shown below.

$$\theta_i = a_0 + a_1 \Delta y + a_2 \Delta \dot{x}$$

$$P_i^b = \begin{cases} 0 & \mathrm{TE}_i \leq \mathrm{TE}_0 \\ b_0 + b_1 \Delta y + b_2 \Delta \dot{x} & \text{otherwise} \end{cases}$$

$$P_i^t = \begin{cases} 0 & \mathrm{TE}_i \geq \mathrm{TE}_0 \\ c_0 + c_1 \Delta y + c_2 \Delta \dot{x} & \text{otherwise} \end{cases}$$

#### 4.4.3.2 Quadratic policy:

The quadratic policy has 18 parameters as shown below.

$$\theta_i = a_0 + a_1\Delta y + a_2\Delta\dot{x} + a_3(\Delta y)^2 + a_4(\Delta\dot{x})^2 + a_5\Delta y\Delta\dot{x}$$

$$P_i^b = \begin{cases} 0 & \text{TE}_i \leq \text{TE}_0 \\ b_0 + b_1\Delta y + b_2\Delta\dot{x} + b_3(\Delta y)^2 + b_4(\Delta\dot{x})^2 + b_5\Delta y\Delta\dot{x} & \text{otherwise} \end{cases}$$

$$P_i^t = \begin{cases} 0 & \text{TE}_i \geq \text{TE}_0 \\ c_0 + c_1\Delta y + c_2\Delta\dot{x} + c_3(\Delta y)^2 + c_4(\Delta\dot{x})^2 + c_5\Delta y\Delta\dot{x} & \text{otherwise} \end{cases}$$

#### 4.4.3.3 Neural network policy:

Each neural network in the policy has 12 hidden layers and the total parameters are 484.

$$\theta = \text{Neural Net 1}$$

$$P_i^b = \begin{cases} 0 & \text{TE}_i \leq \text{TE}_0 \\ \text{Neural Net 2} & \text{otherwise} \end{cases}, \quad P_i^t = \begin{cases} 0 & \text{TE}_i \geq \text{TE}_0 \\ \text{Neural Net 3} & \text{otherwise} \end{cases}$$

where $\Delta y = y_i - y_0$, $\Delta\dot{x} = \dot{x}_i - \dot{x}_0$, $a$'s, $b$'s, and $c$'s are all constants which were fit using the data from section 4.4.2. The linear and quadratic policies were fit using non-linear least squares function *lsqnonlin* in MATLAB and the neural network policy was trained using Levenberg-Marquardt using the Deep Learning Toolbox in MATLAB.

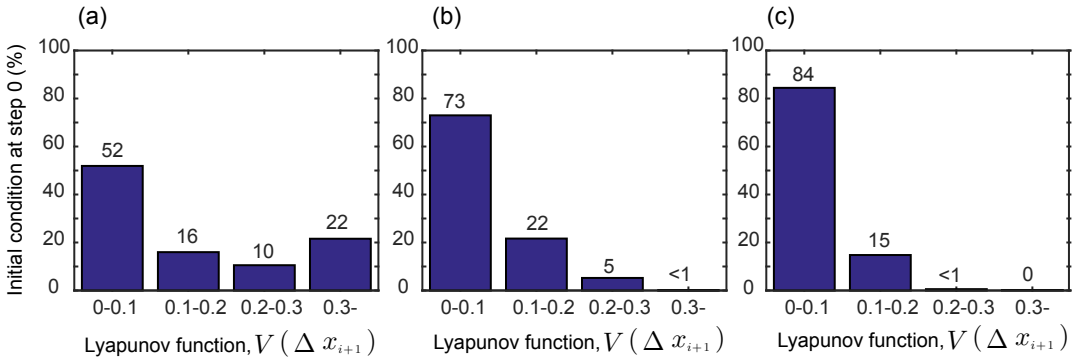


Figure 21: **Verification for different control policies.** A histogram showing Lyapunov function after one step ($V(\Delta \mathbf{x}_{i+1})$) for randomly chosen initial conditions ($\mathbf{x}_i$) in the region of attraction. Each figure represents a different fit between control actions and initial conditions at the apex. The fits are based on: (a) linear, (b) quadratic, and (c) neural networks.

#### 4.4.3.4 Verification

Using the three control fits discussed above, we do the verification as follows. We choose about 1500 initial conditions inside the region of attraction $V(\Delta \mathbf{x}_i) \leq 1$ (note that our fit is based on only 451 points). Next, for each of the initial conditions, we did a forward simulation for one step using each control policy. We plotted the histogram of the Lyapunov function after one step for each of the three fits as shown in Figure 21. We have also indicated the percentages above each bar of the histogram. It can be seen that 52%, 73%, 84% of the initial conditions are in the range $0 < V(\Delta \mathbf{x}_{i+1}) < 0.1$ for the linear, quadratic, and neural network fit respectively. Also, almost 99% of initial conditions are within the range $0 < V(\Delta \mathbf{x}_{i+1}) < 0.3$ for the quadratic fit and within the range $0 < V(\Delta \mathbf{x}_{i+1}) < 0.2$ for the neural network fit.

These results indicate that the neural network provides the best fit for the data followed by the quadratic fit and finally, the linear fit.

#### 4.4.3.5 Robustness

We choose the quadratic control policy for additional robustness checks. The rationale behind using the quadratic policy over the neural network was that the former uses a few parameters while providing a comparable fit as demonstrated in Figure 21. To check robustness we looked into effects such as unmodeled dynamics, noisy sensors/actuators, and external disturbances. These are discussed next.

To check the robustness to unmodeled dynamics, we added a damping force ($F_v = -c_v\dot{\ell}$) in addition to the spring force on the stance leg. Then we simulated the system for 1414 randomly chosen points in the region of attraction, $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq 1$. Figure 22 shows the Lyapunov function (averaged over the 1414 simulations) for 5 consecutive steps for two damping constants (a) $c_v = 100$ Ns/m and (b) $c_v = 200$ Ns/m. These damping values have the effect of reducing the total energy of the periodic motion at the next step by 6% and 10% respectively. It can be seen that for $c_v = 100$, all initial conditions decay to and stay within the range $V(\Delta\mathbf{x}_{i+2}) \leq 0.1$, that is, in 2 steps. However, when damping constant is doubled to $c_v = 200$, all initial conditions are within the range $V(\Delta\mathbf{x}_{i+2}) \leq 0.3$.

We did stochastic simulations to ascertain robustness to several other factors: noisy actuators (foot placement angle and stance forces), noisy sensors (apex height and horizontal velocity), and external disturbance (step change in height at touchdown). We consider each of these 5 factors separately. That is, only one factor was varied by keeping the other four fixed to

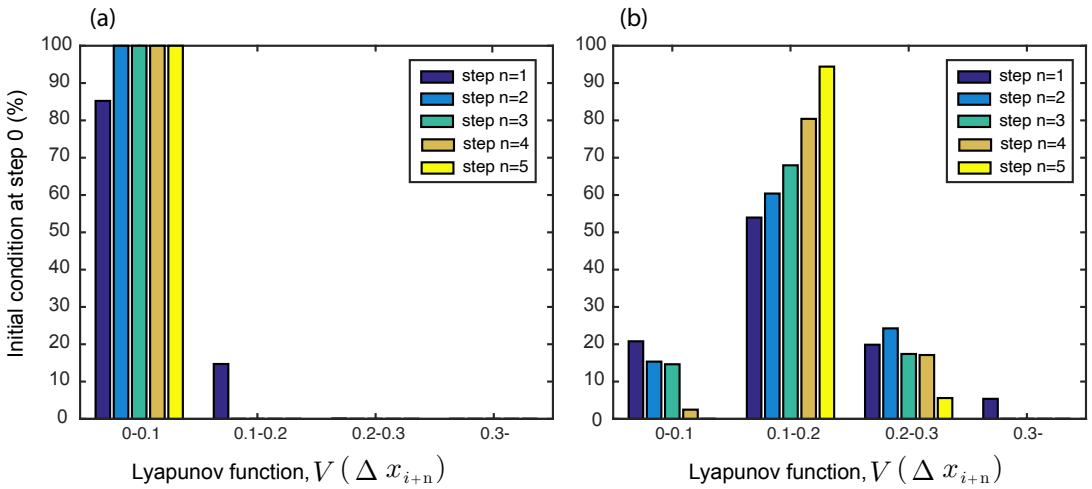


Figure 22: Effect of simulating the system with damping. Lyapunov function $V(\Delta \mathbf{x}_{i+n})$ for $n = 1, 2, 3, 4, 5$ steps starting from randomly chosen initial conditions within the region of attraction for damping factor of (a) $c_v = 100$ (b) $c_v = 200$.

the nominal value. For each factor, we choose a standard deviation $\sigma_d$ (where $d = \theta, P, y, \dot{x}, h$ are foot placement angle, stance forces, apex height, apex horizontal speed, step disturbance, respectively). The standard deviation was varied from zero to a maximum value in certain increments appropriate to the specific factor. For a given $\sigma_d$, we created 10 runs, each of 100 steps. In each of these runs, the noise or disturbance was applied *every step.* For sensor noise, foot placement control actuation noise, and step height disturbance we used uniform random distribution $\{-\sigma_d, \sigma_d\}$ and for stance force noise we used uniform distribution $\{0, \sigma_d\}$ to create perturbations over the nominal values for each of these parameters. For each of these runs, the hopper started at the same initial state, $\mathbf{x}_0$.

Figure 23 shows the results for the stochastic simulations. Each row corresponds to a single factor. The column corresponds (from left to right) the number of steps successfully taken, the average foot placement angle, and the average push-off force. The robustness is best explained using the first column, the number of steps successfully taken: the maximum deviation $\sigma_d$ for which the hopper can take average 100 steps (the maximum number of steps simulated) indicates no failure. This is achieved for foot placement angle $\sigma_\theta \approx 0.04$ rad $= 2.3^\circ$ (about 11% of nominal $\theta$), for stance force $\sigma_P \approx 800$ N, for apex horizontal velocity $\sigma_{\dot{x}} \approx 0.4$ m/s (about 8% of the nominal apex horizontal velocity), for apex height $\sigma_y \approx 0.12$ m (about 9.2% of the nominal apex height), and step disturbance $\sigma_h < 0.01$ m $= 1$ cm. The control actions $\theta$ (column 2) and stance forces $P^b$ and $P^t$ (column 3) for values that indicate no failure (that is, steps traversed is 100) provide an indication of stabilization strategies. In particular, for increasing $\sigma$ for push-off (row 2), apex horizontal speed (row 3), and apex height (row 4), the foot placement angle decreases, but the push-off force increases to stabilize the system. However, increasing $\sigma$ for foot placement angle (row 1) does not change either the push-off or foot placement angle (column 2 and 3) thus indicating that the system relies on the stability properties of the nominal gait for stabilization. Finally, there is no clear trend for an increase in $\sigma$ for step height (row 5) because of the low tolerance for rejecting step height disturbances. Our main conclusion is that the system is most robust to noisy push-off forces, moderately robust to noise in apex height and velocity and to noise in foot placement angle, and least robust to step height disturbance.

Figure 24 gives a better understanding of the evolution of the Lyapunov function across multiple steps for an increasing standard deviation for each of the five parameters. To generate

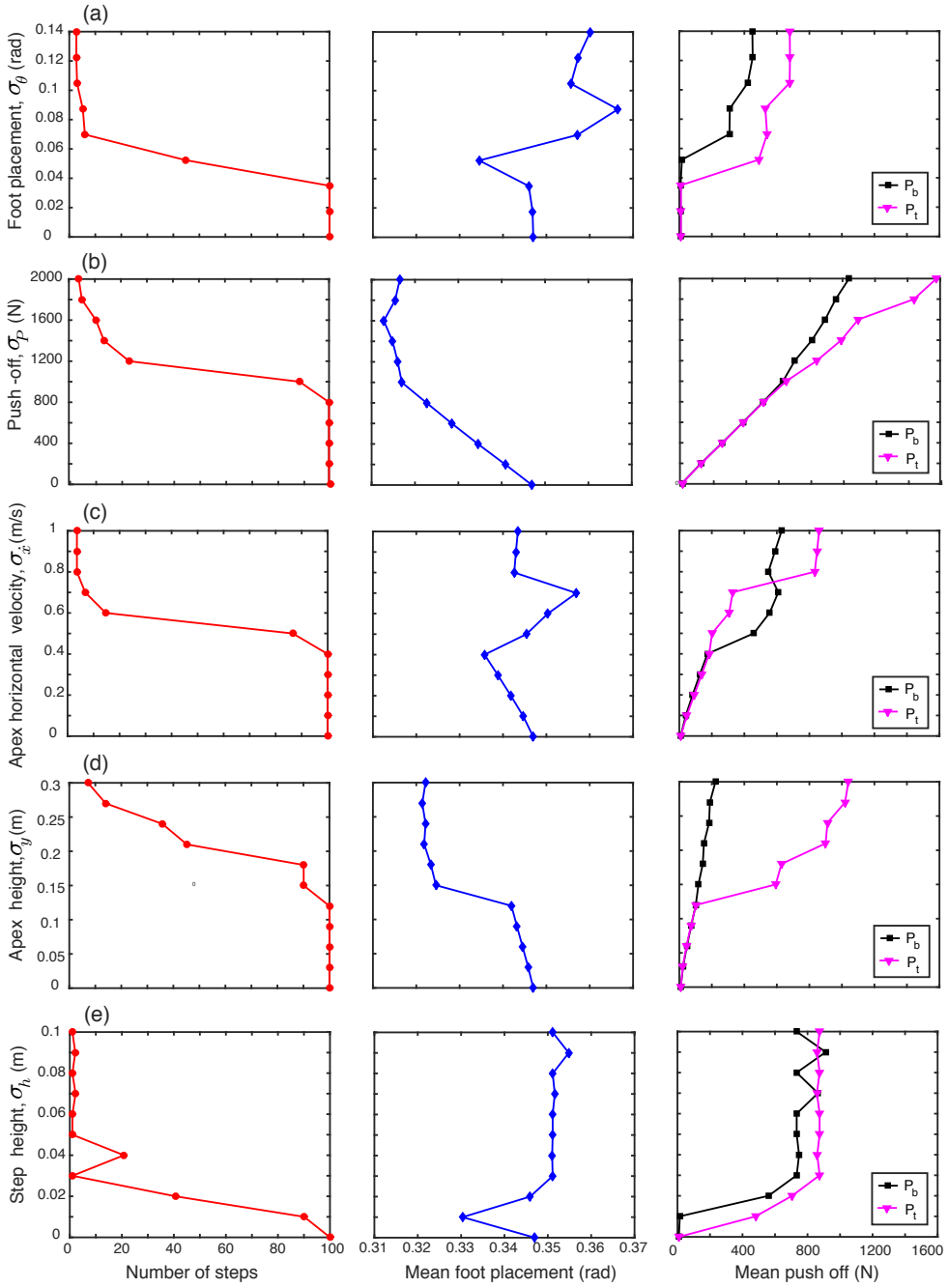


Figure 23: Robustness to noisy actuators (a-b), noisy sensors (c-d) and external disturbance (e). Each row corresponds to a single parameter and the corresponding effect on the number of steps traversed (column 1), foot placement angle (column 2), and force (column 3).

each plot, we started the hopper from the same initial condition, but a different $\sigma$ based on the parameter that we considered, and simulated the system for multiple steps.

Each dot in the figure represents the Lyapunov function at subsequent steps. The light gray ellipse is the region of attraction (that is, $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq 1$) and the dark gray ellipse in the middle corresponds to $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq 0.1$ or the region inside which all initial conditions will decay for a perfect model, perfect sensors, and no external disturbances. The columns are arranged in the increasing order of $\sigma$ from left to right; we chose small, medium, and large model uncertainty/disturbance that the controller is able to sustain. Each row corresponds to a specific external parameter and each column corresponds to increasing standard deviation for the particular parameter. The rows and columns in that order are: (a) foot placement control for $\sigma_\theta$ of $1^\circ$, $2^\circ$, and $3^\circ$, (b) push-off control for $\sigma_P$ of 500, 1000, and 1500 all in N, (c) apex velocity sensor for $\sigma_{\dot{x}}$ of 0.1, 0.25, and 0.5 all in m/s, (d) apex height sensor for $\sigma_y$ of 5, 10, 20 all in cm, (e) step height disturbance for $\sigma_h$ of 2, 5, and 8 all in cm. It can be seen that as the deviation $\sigma$ increases the Lyapunov function over multiple steps stays within the region of attraction, but not inside the ellipse with a boundary defined by 0.1. The effect is most prolonged for the large disturbance (rightmost column), especially for the apex height variation, where the hopper is on the verge of moving out of the assumed region of attraction, but not necessarily falling.

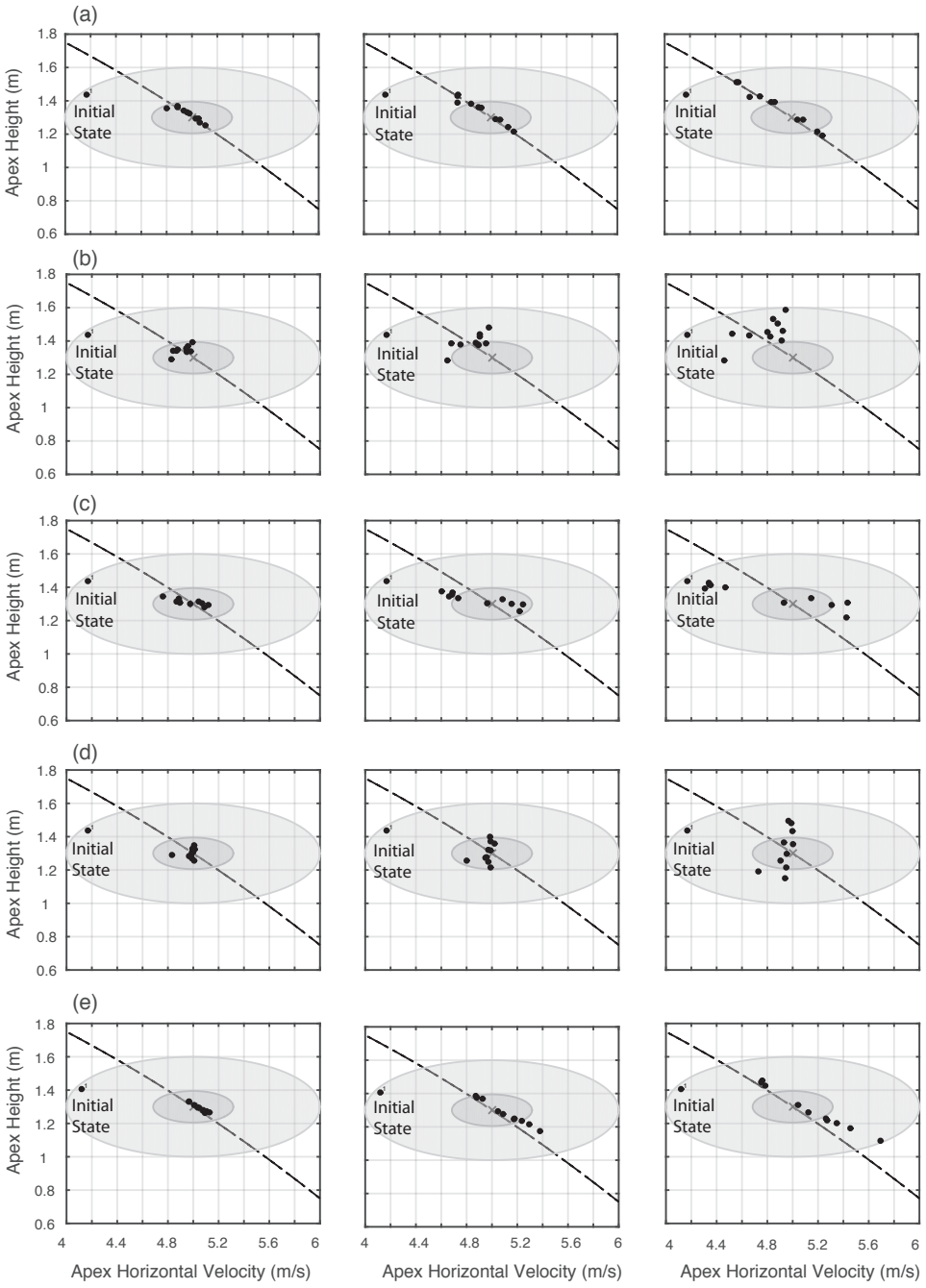


Figure 24: Effect of size of standard deviation ($\sigma$) of various parameters to the evolution of the Lyapunov function. The hopper starts from the same initial conditions for all runs.

## 4.5 Discussion

In this paper, we have presented a sampling-based framework to find control policies for an assumed region of attraction and demonstrated the approach on a model of hopping. The methodology was as follows. We performed a coarse-sampling of the initial conditions at the Poincaré section and used trajectory optimization on each initial condition to find control actions (e.g., gain, amplitudes, set-points) that ensured a reduction of a suitably defined Lyapunov function after a single step. The result was a tabulation of the initial conditions and the corresponding control actions. We fitted each control action as a function of the initial conditions, assuming various functional representations (e.g., linear, quadratic, neural network) to find the control policy. Finally, we verified the control policy on a finely sampled set of initial conditions and did additional robustness checks.

The traditional formal control approach started by assuming a control policy (e.g., linear quadratic regulator, LQR) and then found the largest region of attraction (e.g., using sum-of-squares optimization). Our sampling-based approach worked in the reverse: we started by assuming a region of attraction and then found the control policy using extensive simulations and regression. The formal control approach leads to a simple policy (e.g., the linear policy) at the expense of the possibility of generating a small region of attraction, while our approach guarantees a large region of attraction at the expense of more complicated control policy. For linear systems, the formal control approach based on a linear control policy would be more effective computation- and performance-wise, but for nonlinear systems where a simple linear

controller could potentially produce a small region of attraction, our method would potentially lead to a larger region of attraction.

Another major difference between our and prior work is in the way we define the region of attraction. Tedrake (Tedrake, 2009) and Manchester (Manchester et al., 2010) considered the region of attraction along the trajectory while we considered the region of attraction at the Poincaré section. In these prior works, the system was linearized about the trajectory and the time varying LQR controller was used for tracking purposes. The region of attraction was estimated by finding multiple Lyapunov functions along the trajectory using the sum-of-squares optimization guaranteeing local stability along the trajectory. In our case, there was a single Lyapunov function at the Poincaré section and gave the region of attraction for orbital or step-to-step stability for the nominal cyclic motion. In contrast to the trajectory tracking in the prior work, we solved a computationally simpler regulation problem. In other words, using the Poincaré map-based Lyapunov function, we created a discrete controller that operates in a step-to-step fashion, a common practice in the control of machines exhibiting periodic motion (Hobbelen and Wisse, 2007b).

Our approach blended control theory and machine learning; more specifically, control theory-based stability metrics such as the control Lyapunov function and machine learning-based methods such as sampling and regression (polynomials and deep neural networks) for designing control policies. In this regard, it is important to note that control theory approaches based on the sum-of-squares optimization (Tedrake, 2009) may only work with polynomial model

and control policies. In contrast, our method worked with generic models, including black-box models.

The control actions and Lyapunov function are the designer's choices that affected the final outcomes. For the hopper, our choice of control actions was foot placement angle, braking force, and thrust force while the Lyapunov function was an ellipse. Our choice of control actions not only led to a large region of attraction, but it also led to further simplification as only two of the three control actions were needed to stabilize the system based on the system energy. We chose a quadratic Lyapunov function with principal axes along the speed (x-axis) and height (y-axis) and this allowed us to stabilize a relatively wide range of horizontal speeds (range of $\pm 1$ m/s) for the same apex height, the rationale being that regulating the speed is one of the goals of legged locomotion in the presence of disturbances. In comparison, the work by Heim and Sprowitz (Heim and Spröwitz, 2019) used a single control action, the foot placement angle, which limited the region of attraction to a relatively small area.

We have two comments about our chosen region of attraction. One, we chose our region of attraction to be $(\mathbf{x}_i - \mathbf{x}_0)^T \mathbf{S}_0 (\mathbf{x}_i - \mathbf{x}_0) \leq c$, with $c = 1$. The choice for $c$ is arbitrary. In principle one could choose a large value of $c$, perhaps encompassing the entire state space. Then do optimization for sampled points in the entire space followed by regression to fit a control policy ($\mathbf{u}_i = f(\mathbf{x}_i)$). We hypothesize that the resulting control policy will be highly non-linear and could potentially need a large number of free parameters to describe it adequately. Often a simple control policy (e.g., quadratic or neural network with few parameters) is ideal. Second, although we chose the region of attraction $c = 1$ and found the control policy for that region,

the actual region of attraction is likely much larger. It would be fairly straightforward to find the actual region of attraction by doing additional simulations over larger level sets $c > 1$. After finding the actual region of attraction, one can repeat the control policy evaluation for the new $c$ and repeat the process until no further expansion of the region of attraction is possible.

We advocated offline trajectory optimization followed by online implementation. The average time for the trajectory optimization for the simple model is about 17 seconds (circa 2012 laptop), which is too slow for online implementation. Since the trajectory optimization is done for a range of initial conditions, the resulting data (initial conditions and corresponding control actions) may be saved as a lookup table for online implementation. This simple strategy worked for a few regions of attraction, but as the number of regions of attraction grows (e.g., for different gaits and fixed points and as the system dimensions scale up), the lookup table might be too large. Thus, we advocate mining the data to find a simple functional representation, a control policy of the form, $\mathbf{u} = \mathbf{f}(\mathbf{x})$, which is easier to store and use for real-time control.

# CHAPTER 5

# CREATING AGILE GAITS BY SEQUENTIALLY COMPOSING PERIODIC MOTIONS USING HEURISTICS

(A significant portion of this chapter is reproduced from a published paper with the following citation:

Bhounsule, P., Zamani, A., and Pusey, J.: Switching between limit cycles in a model of running using exponentially stabilizing discrete control lyapunov function. In American Control Conference (ACC), 2018. IEEE, 2018.)

## 5.1 Introduction

The ability to quickly switch between periodic motions or limit cycles allows legged robots to demonstrate agility or the ability to quickly change velocity or direction (Duperret et al., 2016). Very little work has been done in synthesizing agile (non-periodic) gaits, even though a great amount of literature exists on the creation of periodic or steady state gaits. This work provides a technique for constructing non-steady or agile gaits by sequentially composing steady-state gaits, thereby capitalizing on the extensive work done on steady state gaits.

## 5.2 Background and Related Work

A straightforward technique to create agile gaits is to create individual controllers for periodic motion as well as for all possible transitions. For example, Santos and Matos (Santos and Matos, 2011) tuned nonlinear oscillators to generate different gaits and gait transitions in a quadrupedal robot. Haynes and Rizzi (Haynes and Rizzi, 2006) used a heuristic approach in which transitions were created by sequentially changing the controllers for pairs of leg from start to goal gait while maintaining static stability (center of mass projection is within the support polygon of the legs). Byl et al. (Byl et al., 2017) pre-computed transition controllers between successive apex states and used these controllers to map out the reachable state space. The disadvantage of the method is that it requires additional transition controllers to switch between limit cycles.

Transition controller can be avoided entirely if one can find common states between two limit cycles and switch controllers when that particular state is reached. However, it is often very difficult to find common states between two limit cycles. As an alternative, it is relatively easier to use the region of attraction (range of states that converges to the limit cycle) (Strogatz, 1994) to switch. We demonstrate the approach in Figure 25 (a). Each of the ellipsoids are the regions of attraction of a particular limit cycle. Initially we use the controller $C_1$ to get the system to move toward the fixed point corresponding to the first limit cycle. Once the state is inside the region of attraction of the next limit cycle, the controller $C_2$ is switched on and so on. This way, the system can funnel from one limit cycle to another (Burridge et al., 1999).

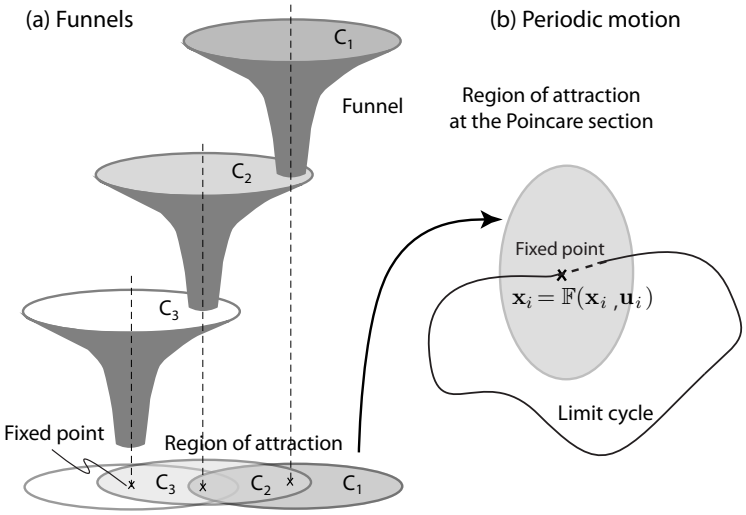


Figure 25: Relation between funnels and limit cycles: (a) Use of funnels to sequentially compose motion (Burridge et al., 1999). (b) Running motion is analyzed using fixed point of the limit cycle at the Poincaré section. Switching controllers are created by composing limit cycles using the region of attraction to create funnels.

Funnel-based switching has been used in the past and is also the main focus of this chapter. Bhounsule et al. (Bhounsule et al., 2016) considered transition of a torso-actuated rimless wheel robot, a simple one degree of freedom system. The key idea was to use a one-step dead-beat control to switch from one fixed point to another in a single step. The switching was done at the mid-stance position using the measured velocity of the leg in contact with the ground. The region of attraction was not estimated but switching was attempted by trial and error. Westervelt (Westervelt et al., 2003) considered switching of a four degree of freedom, walking robot with knees and torso. To deal with the high dimensionality of the system, the switching was done by considering the zero dynamics of the system which is of dimension 1, simplifying

the search for the region of attraction. Cao et al. (Cao et al., 2015) considered the gait transition of a quadruped robot by switching in the mid-flight phase using the reduced state of the body position and velocity (that is, legs were excluded). The region of attraction was estimated by using forward simulation, which may be time consuming. Tedrake and coworkers (Tedrake, 2009; Tedrake et al., 2010) used sum of squares to estimate the region of attraction and used Linear Quadratic Controllers to stabilize gaits, but the same idea can be extended to switch between limit cycles. Veer et. al. (Veer et al., 2017) used an exponential control Lyapunov function (ECLF) to switch between a continuum of limit cycles to generate variable speed walking. In their work, the ECLF ensures exponential local stabilization but in this work we use exponential orbital stabilization, which leads to a much faster transition.

In this chapter, we use funnel-based approach for switching between limit cycles for a running model. The switching is done at the mid-stance position as shown in Figure 25 (b). In contrast to past approaches, we use an orbital control Lyapunov function approach developed in chapter 2 to: (1) estimate the region of attraction, and (2) enable exponential convergence to the fixed points. The latter allows for fast transitioning between limit cycles, typically in a single step, and is the main novelty of our approach.

## 5.3 Model

We use the hoping model described in section 4.3.5 of chapter 4.

## 5.4 Methods

Next, we describe our methodology for creating agile gaits. First, we create multiple limit cycles. Second, we define an orbital Control Lyapunov Function (OCLF) for exponential decay of Lyapunov function between steps and estimate the region of attraction for each limit cycle. Third, we use the idea of funnels to transition from a start state to an end state by sequentially composing the limit cycles.

### 5.4.1 Poincaré map and limit cycle

We define the Poincaré map, $\mathbb{F}$, at the apex. The apex is defined by the condition, $\dot{y} = 0$. Given the state at the apex at step $k$, $\mathbf{x}_k = \{\dot{x}, y\}$, and the control, $\mathbf{u}_k = \{\theta, P^b, P^t\}$ (where $P^b$ and $P^t$ are constant braking and thrust forces during compression and restitution respectively (see Equation 4.11)), we compute the state at the next step,

$$\mathbf{x}_{k+1} = \mathbb{F}(\mathbf{x}_k, \mathbf{u}_k). \tag{5.1}$$

There is no closed form for the map $\mathbb{F}$. In this chapter, it is found numerically by integrating the equations of motion. The $i$th limit cycle is found by fixing $\mathbf{x}_{k+1} = \mathbf{x}_k = \mathbf{x}_i$ and searching for $\mathbf{u}_k = \mathbf{u}_i = \{\theta, P^b = 0, P^t = 0\}$ such that

$$\mathbf{x}_i = \mathbb{F}(\mathbf{x}_i, \mathbf{u}_i). \tag{5.2}$$

The stability of the limit cycle can be found by evaluating the largest eigenvalue of the Jacobian of $\mathbb{F}$, that is, $J = \dfrac{\partial \mathbb{F}}{\partial \mathbf{x}}|_{(\mathbf{x}_i, \mathbf{u}_i)}$ (Strogatz, 1994). For the runner one eigenvalue is always 1 (a

conservative system when $P^b = P^t = 0$) (Poulakakis et al., 2006) while the second eigenvalue is used to evaluate the stability. An eigenvalue less than 1 indicates a stable limit cycle and unstable otherwise.

### 5.4.2 Orbital Control Lyapunov Function (OCLF)

We use the OCLF derived in section 4.3.2.1 of chapter 4. The condition for exponential stability is repeated here

$$
\begin{aligned}
& V(\Delta \mathbf{x}^i_{k+1}) - (1-\alpha) V(\Delta \mathbf{x}^i_k) \leq 0 \\
\Longrightarrow & (\mathbf{x}_{k+1} - \mathbf{x}_i)^T \mathbf{S}_i (\mathbf{x}_{k+1} - \mathbf{x}_i) - (1-\alpha)(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S}_i (\mathbf{x}_k - \mathbf{x}_i) \leq 0, \\
\Longrightarrow & \Big( F(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_i \Big)^T \mathbf{S}_i \Big( F(\mathbf{x}_k, \mathbf{u}_k) - \mathbf{x}_i \Big) - (1-\alpha)(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S}_i (\mathbf{x}_k - \mathbf{x}_i) \leq 0. \qquad (5.3)
\end{aligned}
$$

### 5.4.3 Region Of Attraction (ROA)

The Region Of Attraction (ROA), $\mathcal{R}$, of the controller is the set of all initial conditions $\mathbf{x}_k$ that would converge to the corresponding limit cycle, $\mathbf{x}_i$. In our case, we are interested in all $\mathbf{x}_k$ for which we can find $\mathbf{u}_k$ such that Equation 5.3 is satisfied. To find the ROA, $\mathcal{R}_i$ for a given limit cycle we need to find level set, $(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S}_i (\mathbf{x}_k - \mathbf{x}_i) = c$ such that Equation 5.3 is satisfied. We restrict ourselves to $c \leq 1$.

First, we find $\mathbf{S}_i = diag\{s_{i1}, s_{i2}\}$ such that level $(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S}_i (\mathbf{x}_k - \mathbf{x}_i) = 1$ intersects the state constraint $y = \ell_0$ (a conservative estimate that prevents the leg from stubbing the ground at the Poincaré section, the flight phase apex position, assuming the leg is vertical and at its nominal length $\ell_0$).

Second, we find maximum value of $c$ such that $(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S}_i (\mathbf{x}_k - \mathbf{x}_i) \leq c$ and exponential stabilization conditions given by Equation 5.3 are satisfied. This is done numerically as follows: (1) fix $c$ to a small value and compute $\mathbf{x}_k$'s on the level set, (2) check if a $\mathbf{u}_k$ exists such that Equation 5.3 holds using nonlinear optimization, (3) increase $c$ and repeat, (4) stop when at least one $\mathbf{x}_k$ is infeasible or when $c = 1$.

### 5.4.4 Minimizing Energy Cost

When finding a controller for each state within the ROA, we minimize the Mechanical Cost Of Transport (MCOT) defined as energy used per unit weight per unit distance travelled

$$\begin{aligned} E_{net} &= E^k + E^b + E^t \\ &= \int_{\text{step}} \Big( |k(\ell_0 - \ell)\dot{\ell}| + |P^b \dot{\ell}| + |P^t \dot{\ell}| \Big) dt \\ \text{MCOT} &= \frac{E_{net}}{mgD} \end{aligned} \tag{5.4}$$

where $E^k$, $E^b$, and $E^t$ are mechanical work done by the spring force due to foot placement, constant braking force, and thrust force respectively. The optimization problem is to minimize MCOT (Equation 5.4) subject to the exponential OCLF condition (Equation 5.3) for the given initial condition $\mathbf{x}_0$.

### 5.4.5 Transitioning using funnels

The key idea behind transitioning between limit cycles with stability guarantees is shown in Figure 25 and inspired by (Burridge et al., 1999). We choose limit cycles such that the fixed point of one limit cycle is in the region of attraction of the next limit cycle. When the

system state is in the region of attraction of the next limit cycle, the corresponding controller is switched on. The algorithm is given in Algorithm 1.

**Algorithm 1** Transition(initial state $\mathbf{x}_{init}$, goal state $\mathbf{x}_{goal}$)

**Input:** initial state $\mathbf{x}_{init}$, goal state $\mathbf{x}_{goal}$.
**Output:** Set of control actions for each step, $u_k$, where step index is $k = 1, 2, ..., n$.

1: Compute sufficiently large number of limit cycles (say $m$) and their associated controller ($u$) between init and goal state such that their ROA's ($\mathcal{R}$) overlap (See Sections 5.4.1 and 5.4.3).
2: SORT($\mathcal{R}_p$), $p = 1, 2, ..., m$ either in increasing or decreasing order of speed of fixed point.
3: Initialize step number $k \leftarrow 0$ and state $\mathbf{x}_0 \leftarrow \mathbf{x}_{init}$
4: **while** $|x_{k+1} - \mathbf{x}_{goal}| \geq \delta$ **do** {$\delta$ is a small number}
5: FIND($\mathbf{x}_j$) all the fixed points $\mathbf{x}_j$ where $j = 1, 2, ..., N$ such that $\mathbf{x}_k \in \mathcal{R}_j$. {NOTE: $\mathbf{x}_j$ is a subset of the $m$ limit cycles computed earlier.}
6: Choose fixed point $\mathbf{x}_j$ that is closest to the goal state (say $\bar{\mathbf{x}}_j$) and set the corresponding controller $u_k = \bar{\mathbf{u}}_j$.
7: Apply the control $u_k$ and compute the state at the next apex, $\mathbf{x}_{k+1}$.
8: $k \leftarrow k + 1$
9: **end while**

## 5.5 Results

In all the following results we assume that mass is $m = 80\ kg$, nominal leg length is $\ell_0 = 1\ m$, gravity $g = 10\ m/s^2$, and spring constant $k = 32000\ N/m$. For all simulations, we choose the exponential decay rate $\alpha = 0.9$ (see Equation 5.3).

### 5.5.1 Example 1: Exponential Stabilization

To illustrate the exponential stabilization of the controller we proceed as follows. First, we found foot placement $\theta$ needed to achieve a fixed point of $\mathbf{x}^* = \{5.0, 1.3\}$. The corresponding value is $\theta^* = 0.3465$ rad. The maximum eigenvalue was 1.33 indicating an unstable gait. We found the positive definite matrix for the Lyapunov function to be $\mathbf{S}^* = diag\{1, 11.1\}$ as described in section 5.4.3. To test the controller, we started with an initial condition at step 0, $\mathbf{x}_0 = \{4.20, 1.48\}$, that is different from the fixed point. The value of the Lyapunov function for the initial condition is $V(\Delta\mathbf{x}_0) = (\Delta\mathbf{x}_0^*)^T\mathbf{S}_i^*\Delta\mathbf{x}_0^* = 1$. Subsequently, the controller ensures exponential decay of the Lyapunov function, $V(\Delta\mathbf{x}_1) = 0.1$ and $V(\Delta\mathbf{x}_2) = 0.01$.

### 5.5.2 Example 2: Robustness to height variation

Figure 26 and Table I demonstrate robustness of the OCLF. First, we found foot placement $\theta$ needed to achieve a fixed point of $\mathbf{x}^* = \{\dot{x}^*, y^*\} = \{2.0, 1.3\}$. The corresponding value is $\theta^* = 0.1603$. To test the controller, we introduced a ditch of height 0.2 $m$. Subsequently, the state of the system at the apex (step 0) is $\mathbf{x}_0 = \{1.957, 1.5085\}$. Figure 26 shows the evolution of the state at subsequent steps. The black dashed line indicates the constant total energy line ($\text{TE}^* = mgy^* + 0.5m\dot{x}^{*2}$) passing through the fixed point (black cross). Initially at step 0 the TE is greater than $\text{TE}^*$, thus energy needs to be dissipated. OCLF achieves this by using a non-zero force in compression phase ($P^b$) (see Table I, row 1). Thereafter, on step 1, the TE reaches the constant TE line passing through the fixed point as shown in Figure 26. The controller then adjusts only foot placement to further reduce the Lyapunov function (see Table I, row 1).

TABLE I: Transition for example 2: Robustness to height change. The fixed point of the target limit cycle is $\{2, 1.3\}$. The nominal foot placement angle is 0.1603 rad. A ditch of height 0.2 m is introduced due to which the system starts with the initial condition, $\mathbf{x}_0 = \{1.957, 1.5085\}$. The mechanical work done by the actuator due to spring force for the limit cycle is $E_i^k = 795.4332$ $J$ and $\text{MCOT}_i = 0.7533$.

| $k$ | $i$ | $\mathbf{x}_k$ | $\theta$ | $P^b$ | $P^t$ | $E^k - E_i^k$ | $E^b$ | $E^t$ | MCOT |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | $\{1.957, 1.5085\}$ | 0.14823 | 941.2727 | 0 | 54.4924 | 153.3961 | 0 | 0.89718 |
| 1 | 1 | $\{1.8279, 1.3412\}$ | 0.13238 | 0 | 0 | 49.7985 | 0 | 0 | 0.81926 |
| 2 | 1 | $\{1.9578, 1.3166\}$ | | | | | | | |

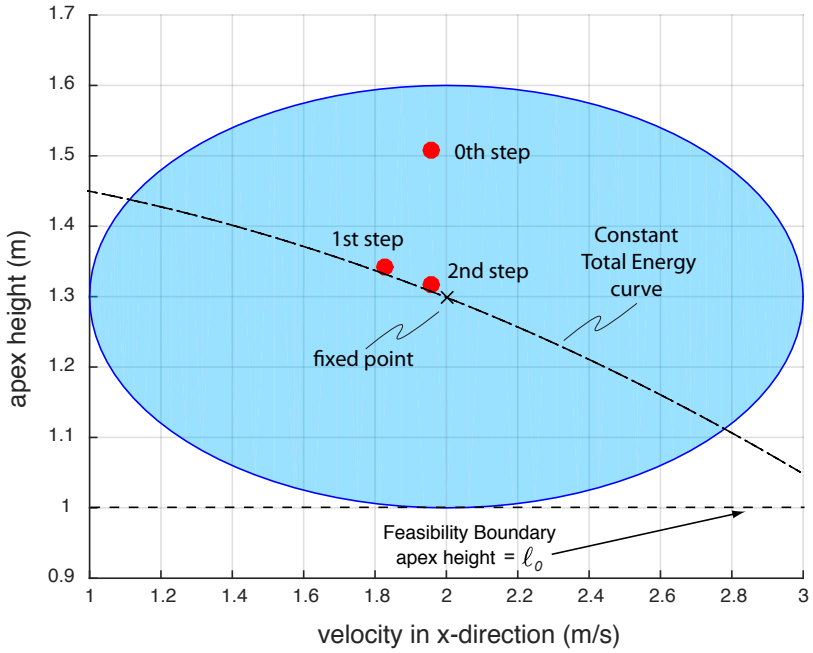


Figure 26: Example 2, robustness to step down. A step down leads to increase in apex height because the height is measured relative to the ground. The black cross indicates the fixed point. The dashed line indicates the constant total energy, sum of kinetic and potential energies, that passes through the fixed point. The blue region indicates the ROA. The red dots indicate the system state at each step. The figure demonstrates how OCLF is able to tackle an external disturbance.

### 5.5.3 Example 3: Transitioning between limit cycles with and without actuator bounds

Figure 27 illustrates how to switch between limit cycles using funnels. The model starts at the fixed point $\{2, 1.2\}$ and needs to end at the fixed point $\{5, 2\}$. We create limit cycles for these two fixed points and estimate their ROA. We also create additional limit cycles and estimate their ROA. We need a total of 3 more limit cycles such that fixed point of one limit cycle is in the ROA of the subsequent limit cycle so that transitions are possible. Table II shows the 5 limit cycles, the foot placement angle, associated energy, and mechanical cost of transport.

We considered two cases: (a) no actuator bounds and (b) with actuator bound of $12mg$ in the prismatic actuator. Figure 27 shows the evolution of the state at the apex between successive steps while Table III shows in addition, the control strategy and energy usage. We can see that for steps 0-2 the state, control, and energy are identical (compare rows 1, 2, and 3 in Table III (a) with those in (b)). Thereafter, the actuator limit kicks in and the convergence for (b) (with actuator limits) is slower than (a) (no actuator limits). Overall, it takes only 7 steps for (a) but 9 steps for (b) to be sufficiently close to the fixed point of the target limit cycle. It is interesting to note that for both cases only the thrust force is applied (i.e., $P^t > 0$) in the first 4 steps to enable fast transition between limit cycles, as this has the effect of adding energy and speeding up the runner. The MCOT does not change appreciably in the entire transition.

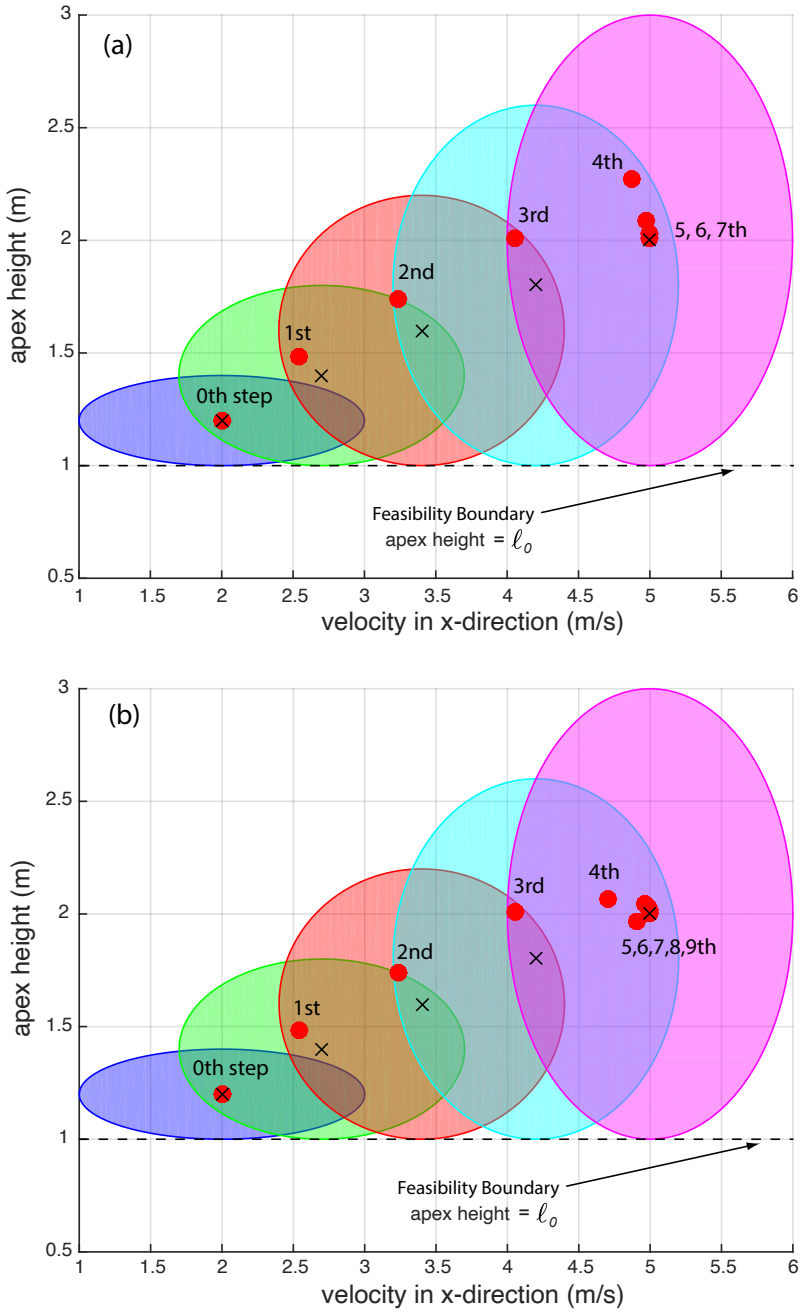


Figure 27: Transitioning between limit cycles for example 3 for (a) no actuator limits, (b) with actuator limits. The black cross indicates the fixed point. The shaded region indicates the ellipsoid estimating the ROA for each limit cycle. The red dots indicate the system state at each step.

TABLE II: Limit cycles for example 3, transition controllers

| limit cycle, # $i$ | $\mathbf{x}^i$ | maximum eigenvalue | $\theta$ | $\{s_{i1}, s_{i2}\}$ | $E^i_\theta$ | MCOT |
|---|---|---|---|---|---|---|
| 1 | $\{2, 1.2\}$ | 1.5958 | 0.16328 | $\{1, 25\}$ | 588.9249 | 0.6394 |
| 2 | $\{2.7, 1.4\}$ | 1.7246 | 0.20813 | $\{1, 6.25\}$ | 1040.5214 | 0.6564 |
| 3 | $\{3.4, 1.6\}$ | 1.8223 | 0.25237 | $\{1, 2.7778\}$ | 1504.095 | 0.6446 |
| 4 | $\{4.2, 1.8\}$ | 1.8846 | 0.30156 | $\{1, 1.5625\}$ | 2003.4739 | 0.6187 |
| 5 | $\{5, 2\}$ | 1.9269 | 0.34897 | $\{1, 1\}$ | 2533.3107 | 0.5987 |

TABLE III: Transition for example 3: The source state is $\{2, 1.2\}$ and target state is $\{5, 2\}$.
(a) No actuator limits

| step #, $k$ | limit cycle, # $i$ | $\mathbf{x}_k$ | $\theta$ | $P^b$ | $P^t$ | $E_\theta - E^i_\theta$ | $E_{P^b}$ | $E_{P^t}$ | MCOT |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | $\{2, 1.2\}$ | 0.10619 | 0 | 2561.2099 | $-516.3278$ | 0 | 327.0972 | 0.6985 |
| 1 | 3 | $\{2.5369, 1.4871\}$ | 0.14125 | 0 | 1960.9835 | $-409.3977$ | 0 | 362.1299 | 0.73876 |
| 2 | 4 | $\{3.239, 1.737\}$ | 0.19017 | 0 | 2033.0225 | $-385.6915$ | 0 | 456.4176 | 0.71919 |
| 3 | 5 | $\{4.0495, 2.0121\}$ | 0.24826 | 0 | 1895.4544 | $-279.4241$ | 0 | 502.4145 | 0.69664 |
| 4 | 5 | $\{4.8733, 2.2726\}$ | 0.32461 | 363.5763 | 0 | 325.6946 | 108.6778 | 0 | 0.66728 |
| 5 | 5 | $\{4.9688, 2.0898\}$ | 0.34128 | 137.2408 | 0 | 176.4281 | 39.9361 | 0 | 0.63788 |
| 6 | 5 | $\{4.9912, 2.0287\}$ | 0.34654 | 45.8814 | 0 | 129.3415 | 13.2341 | 0 | 0.62852 |

(b) With actuator limits.

| step #, $k$ | limit cycle, # $i$ | $\mathbf{x}_k$ | $\theta$ | $P^b$ | $P^t$ | $E_\theta - E^i_\theta$ | $E_{P^b}$ | $E_{P^t}$ | MCOT |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | $\{2, 1.2\}$ | 0.10619 | 0 | 2561.21 | $-516.3278$ | 0 | 327.0972 | 0.6985 |
| 1 | 3 | $\{2.5369, 1.4871\}$ | 0.14125 | 0 | 1960.9766 | $-409.3978$ | 0 | 362.1286 | 0.73876 |
| 2 | 4 | $\{3.239, 1.737\}$ | 0.19017 | 0 | 2033.0298 | $-385.6945$ | 0 | 456.4188 | 0.71919 |
| 3 | 5 | $\{4.0495, 2.0121\}$ | 0.25399 | 0 | 1017.2232 | $-252.895$ | 0 | 271.3035 | 0.68289 |
| 4 | 5 | $\{4.7062, 2.0638\}$ | 0.31812 | 0 | 0 | 91.0704 | 0 | 0 | 0.64186 |
| 5 | 5 | $\{4.9116, 1.965\}$ | 0.34044 | 0 | 286.4697 | $-3.8926$ | 0 | 80.539 | 0.62526 |
| 6 | 5 | $\{4.9883, 2.0277\}$ | 0.34996 | 0 | 0 | 171.1558 | 0 | 0 | 0.63364 |
| 7 | 5 | $\{4.9607, 2.0414\}$ | 0.34373 | 19.6883 | 0 | 154.9589 | 5.7058 | 0 | 0.63215 |
| 8 | 5 | $\{4.9949, 2.0173\}$ | 0.34752 | 27.9249 | 0 | 120.6562 | 8.0413 | 0 | 0.62679 |

## 5.6 Discussion

We have presented a method for switching between limit cycles based on two key ideas: (1) use the region of attraction at the Poincaré section to create funnels, and (2) use exponentially stabilizing orbital control Lyapunov function for fast switching between limit cycles. We demonstrate the approach on a model of running with an axial actuator to control the ground interaction forces and a hip actuator to control the foot placement position. We show that a fairly wide range of initial perturbations can converge to within 1% of the fixed point in a maximum of 2 steps. We also demonstrate the runner can transition from the limit cycle with a speed of 2 $m/s$ to 5 $m/s$ (a change of 150%) in about 5 steps even with actuator limits.

The use of three control actions ($P^b$, $P^t$, $\theta$) substantially increases the region of attraction of the model (e.g., the ellipsoid shown in Figure 26). When the two constant forces are assumed to be zero (i.e., $P^b = P^t = 0$) the region of attraction shrinks to a curve (e.g., the total energy dashed line shown in Figure 26). This special case in which energy is conserved between steps is called the Spring Loaded Inverted Pendulum (SLIP) model of running (Schwind, 1998).

The use of an orbital control Lyapunov function (OCLF) with exponential stabilization accelerates the convergence to the fixed point, thus enabling fast switching between limit cycles. This is a distinct advantage over controllers that impart asymptotic convergence, which is slower (Grizzle et al., 2001). Another approach is to use a one-step dead-beat control (full correction of disturbance in a single step) as it gives the fastest switching between limit cycles. However, we have found in chapter 2 that OCLF can handle a larger range of modeling errors than a one-step dead-beat control and is thus preferred, especially when implementing on hardware.

# CHAPTER 6

# CREATING AGILE GAITS BY SEQUENTIALLY COMPOSING PERIODIC MOTIONS USING SAMPLING-BASED TECHNIQUES

(A significant portion of this chapter is reproduced from a published paper with the following citation:

Zamani, A., Galloway, J. D., and Bhounsule, P. A.: Feedback motion planning of legged robots by composing orbital Lyapunov functions using rapidly exploring random trees. In 2019 International Conference on Robotics and Automation (ICRA), pages 14101416. IEEE, 2019.)

## 6.1 Introduction

For legged robots to achieve mainstream applications, motion planning or planning movement over multiple steps is essential. However, the complex dynamics (e.g., unstable modes, hybrid dynamics) and the under-actuation (more degrees of freedom than actuators) of legged robots make motion planning conceptually and computationally quite challenging. In this context, one approach is to have a hierarchical control: first, multiple low-level dynamic controllers are developed for within-a-step balance control and then, the low-level controllers are combined to achieve high-level behaviors over multiple steps. Our control framework is based on the same idea: we use an orbital Lyapunov function (a Lyapunov function defined at the Poincaré section

and is indicative of the step-to-step or orbital stability) to achieve stability of periodic motions and then combine these periodic motions using rapidly-exploring random trees for planning.

## 6.2 Background and Related Work

One of the earliest techniques for motion planning was by Raibert and Wimberly (Raibert and Wimberly, 1984) who used a computer simulation to generate a table of control actions as a function of the robot states. But such a table was prohibitively large for storage and search purposes, so they approximated the table with high order polynomials and demonstrated their approach on a hopping robot. Similarly, Da et al. (Da et al., 2017) used a computer simulation to create controllers as a function of robot states and terrain height. The data was then input into a supervised learning framework to learn a control policy, which was then implemented on a bipedal robot. One issue with these past works are that they do not explicitly minimize an objective function.

Dynamic programming (DP) is a method for optimal motion planning. The technique relies on the discretization of the state space and defining a suitable cost function. The prime issue with this method is that the quality of the results depends on the fineness of discretization, but for a fine discretization and high degrees of freedom system, the control synthesis becomes quickly intractable due to high computation and storage cost. The common techniques for tackling this issue is to make simplifications to the system. For example, Whitman (Whitman, 2013) simplified a humanoid robot into three simple independent models; a sagittal, a lateral, and a frontal model. Controllers were developed for the individual models and combined during

run-time using time as the phase variable. Mandersloot et al. (Mandersloot et al., 2006) discretized the system from one-step to another using the Poincaré section and used this step-to-step discretization within the dynamic programming framework. Because of the extensive computations, DP is mostly an offline method.

Model predictive control (MPC) is an online optimization technique that relies on repeatedly computing the optimal control policy for a given planning time, then implementing only a part of the planned control policy, followed by replanning as new information becomes available. Park et al. (Park et al., 2015) considered the problem of a quadruped jumping over randomly placed obstacles. They first created stable low-level bounding controllers for the quadruped. Then using the MPC framework, the robot planned apex velocity and step length based on terrain information, which was then input to the low-level controller. Rutschmann et al. (Rutschmann et al., 2012) considered MPC but in the context of tracking a foothold position while maintaining balance on rough terrain for a hopper model.

Our method capitalizes on combining multiple steady state gaits to create motion plans as illustrated next. Figure 28 (a) shows a limit cycle. Associated with the limit cycle is a fixed point, an initial condition picked at the Poincaré section (an instance of motion such as apex or touchdown) that maps onto itself at the Poincaré section of the next step. Let the pink ellipse denote the region of attraction (ROA) of the fixed point. The ROA represents all the initial conditions at the Poincaré section that will converge to the fixed point after one or more steps. Figure 28 (b) shows how multiple limit cycles may be combined to create motion plans. For example, the fixed point of the ROA $E_1$ is inside the ROA $E_2$. Similarly, the fixed point of

ROA $E_2$ is inside the ROA of $E_3$. When the system is near the fixed point of $E_1$, switching the controller to that of $E_2$ will cause the system to move towards the fixed point of $E_2$. Thus, switching the controllers based on ROA's is a straightforward method to create motion plans. This technique was used by Veer et al. (Veer and Poulakakis, 2018; Veer et al., 2017; Veer and Poulakakis, 2019) to achieve variable speed walking on a bipedal robot and by Cao et al. (Cao et al., 2015) to achieve gait transition for a quadruped robot. In both works, extensive numerical simulations were used to estimate the ROA at the Poincaré section. Sampling-based methods may be used with ROAs to achieve feedback motion planning. For example, Tedrake (Tedrake, 2009) used a linear quadratic regulator (LQR) to stabilize limit cycles and to estimate the ROA of the LQR controller. For a random point chosen on the state space, LQR-trees inspired by rapidly-exploring random trees (RRT) were grown towards the limit cycle. The ROAs were estimated using the sum of squares optimization (Prajna et al., 2002).

In this chapter, we switch controllers at the Poincaré section as done before. However, unlike previous works, we use an orbital Lyapunov function developed in chapter 2 to assume a candidate ROA and use trajectory optimization to guarantee exponential convergence of all initial conditions with the ROA. In chapter 5, we explored composing limit cycles using heuristics to achieve agile gaits. This chapter presents two improvements over our past approach: (1) use of deep neural networks to learn control policies for fixed points and their ROAs, and then generalize across multiple fixed points without additional computations, and (2) adopt the RRT algorithm to reason with ROAs to achieve agile gaits.

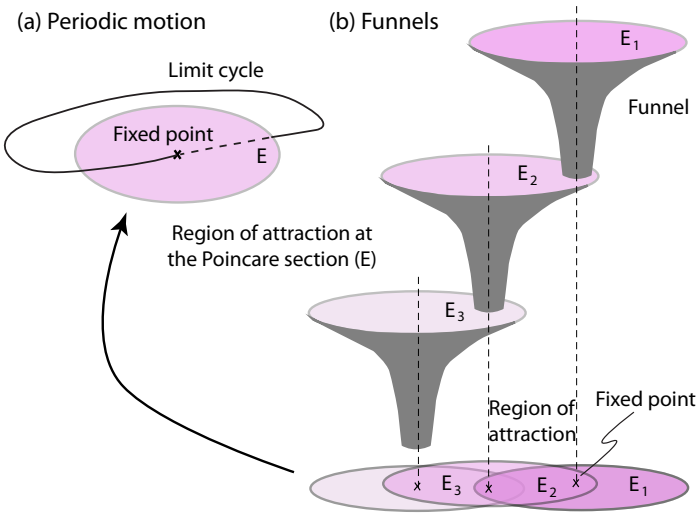


Figure 28: Relation between funnels and limit cycles: (a) A steady state solution or limit cycle and the corresponding region of attraction at the Poincaré section. (b) Feedback motion planning by composing regions of attraction based on common overlapping areas to funnel the systems from one limit cycle to another.

## 6.3 Model

We use the hoping model described in section 4.3.5 of chapter 4.

## 6.4 Preliminaries

### 6.4.1 Poincaré map and limit cycle

Given the state at the apex at step $k$, $\mathbf{x}_k = \{\dot{x}, y\}$, and the control actions, $\mathbf{u}_k = \{\theta, P^b, P^t\}$, we compute the state at the next step,

$$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k). \tag{6.1}$$

The $i$th limit cycle is found by fixing $\mathbf{x}_{k+1} = \mathbf{x}_k = \mathbf{x}_i$ and searching for $\mathbf{u}_k = \mathbf{u}_i = \{\theta, P^b = 0, P^t = 0\}$ such that

$$\mathbf{x}_i = F(\mathbf{x}_i, \mathbf{u}_i). \tag{6.2}$$

### 6.4.2 Orbital Lyapunov Function (OLF)

We define a Lyapunov function for the $i$th limit cycle at the Poincaré section ($F$) as follows

$$V(\Delta\mathbf{x}_k^i) = (\Delta\mathbf{x}_k^i)^T \mathbf{S} \Delta\mathbf{x}_k^i = (\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S} (\mathbf{x}_k - \mathbf{x}_i) \tag{6.3}$$

where the positive definite matrix $\mathbf{S} = diag\{\frac{1}{s_1^2}, \frac{1}{s_2^2}\}$.

An exponentially decaying condition on the Lyapunov function can be expressed as

$$V(\Delta\mathbf{x}_{k+1}^i) - V(\Delta\mathbf{x}_k^i) \leq -\alpha V(\Delta\mathbf{x}_k^i), \tag{6.4}$$

where $0 < \alpha \leq 1$ is the rate of decay of the Lyapunov function between steps.

### 6.4.3 Region Of Attraction (ROA)

The Region Of Attraction (ROA), $\mathcal{R}$, of the controller is the set of all initial conditions $\mathbf{x}_k$ that would converge to the corresponding limit cycle $\mathbf{x}_i$ over one or more steps. The region of attraction is defined by the level set $c$ from the equation $(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S} (\mathbf{x}_k - \mathbf{x}_i) - c =$. In general,

the value of $c$ is constrained by actuator limits or kinematic limits. We choose the constant $c = 1$ (a design choice) to ensure that the biggest switch in velocity is $2c = 2$ m/s.

## 6.5 Methods

### 6.5.1 Trajectory optimization for a given initial condition

For a given initial condition $\mathbf{x}_k \neq \mathbf{x}_i$ at the Poincaré section, we solve the following trajectory optimization problem

$$\underset{\mathbf{u}_k}{\text{minimize}} \qquad \text{MCOT} \tag{6.5}$$

$$\text{subject to:} \qquad \mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k) \tag{6.6}$$

$$V(\Delta \mathbf{x}^i_{k+1}) - (1-\alpha) V(\Delta \mathbf{x}^i_k) \leq 0 \tag{6.7}$$

where MCOT is the Mechanical Cost Of Transport and is defined as the energy used in a step ($E_{step}$) per unit weight ($mg$) per unit distance traveled in one step ($D$)

$$\begin{aligned} \text{MCOT} &= \frac{E_{step}}{mgD} \\ E_{step} &= E^k + E^b + E^t \\ &= \int_{\text{step}} \Big( |k(\ell_0 - \ell)\dot{\ell}| + |P^b \dot{\ell}| + |P^t \dot{\ell}| \Big) dt. \end{aligned} \tag{6.8}$$

where $E^k$, $E^b$, and $E^t$ are mechanical work done by the axial actuator to simulate a springy leg, constant compression force, and restitution force respectively. We solve the optimization problem defined by Equation 6.5-Equation 6.7 using single shooting method (Betts, 2010). One

caveat in the optimization problem formulation is that there is no formal guarantee that the optimization problem will converge to a feasible solution, but our experience has been that when the control actions (e.g., foot placement, brake force, thrust force) have a fairly independent effect on the system state, which is a measure of the controllability of the system in sense, the optimization problem generally converges to a feasible solution without any issues.

### 6.5.2 Overview of the feedback motion planning approach

#### 6.5.2.1 Trajectory optimizations for sampled initial conditions within the ROA for multiple limit cycles

First, we solve the trajectory optimization problem defined in section 6.5.1 for a given limit cycle $\mathbf{x}_i$ and for multiple initial conditions chosen within the ROA $(\mathbf{x}_k - \mathbf{x}_i)^T \mathbf{S} (\mathbf{x}_k - \mathbf{x}_i) - 1 < 0$. We also store the fixed point of the limit cycle $\mathbf{x}_i$, the initial conditions $\mathbf{x}_k$, and the corresponding control actions $\mathbf{u}_k$. This process is repeated to generate control actions for ROAs for multiple limit cycles as shown in Figure 29 (a).

#### 6.5.2.2 Filling state space with regions of attraction and associated stabilizing controllers followed by feedback motion planning using rapidly-exploring random trees

We use the stored data to find a control policy for each control action as a function of the fixed point and initial conditions, $u_j = f_j(\mathbf{x}_i, \mathbf{x}_k)$ where $j = 1, 2, 3$ correspond to the three control actions for the hopper. We use deep neural networks to estimate the functions $f_j$. Chapter 4 informs the choice of neural networks for the $f_j$. We can use the functions $f_j$ to predict the control actions for a randomly chosen fixed point at the Poincaré section. The

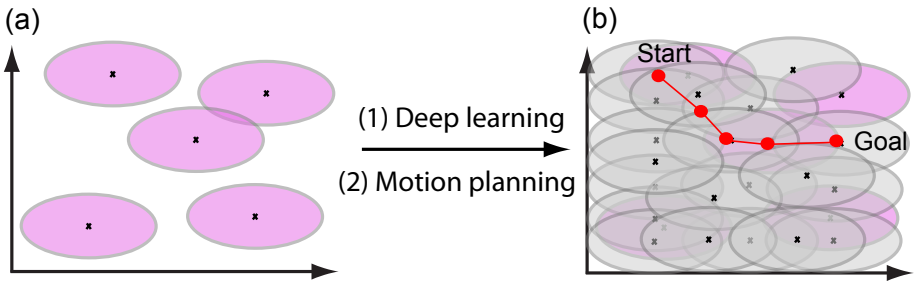


Figure 29: Our control framework: (a) generate controllers to create regions of attraction for multiple fixed points using trajectory optimization; (b) use deep learning to develop control policies for other fixed points (grey ellipses) to fill the state space and then use overlaps between ROAs for motion planning from start point to goal point.

resulting function $f_j$ is validated for ROAs of multiple fixed points as shown by grey ellipses in Figure 29 (b). Next, we plan transitions using rapidly-exploring random trees (RRT). The key idea is to switch at the Poincaré section (i.e., at the apex for the hopper model) by reasoning with the overlap between the ROA of two limit cycles.

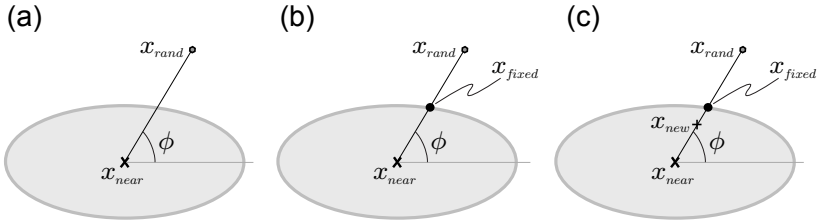


Figure 30: Pictorial depiction of the RRT algorithm

**Algorithm 2** GENERATE_RRT($\mathbf{x}_{init}, \mathbf{x}_{goal}, \delta\dot{x}, \delta y, N_P$)

1: T.init($\mathbf{x}_{init}$);
2: **for** $p = 1$ to $N_P$ **do**
3: $\mathbf{x}_{rand} \leftarrow$ RANDOM_STATE();
4: $\mathbf{x}_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathbf{x}_{rand}, T$);
{NOTE: $\mathbf{x}_{near} = \mathbf{x}_k$}
5: $\mathbf{x}_{fixed} \leftarrow$ NEW_FIXED_POINT($\mathbf{x}_{near}, \mathbf{x}_{rand}, \delta\dot{x}, \delta y$);
{NOTE: $\mathbf{x}_{fixed} = \mathbf{x}_i$}
6: $\mathbf{u}_{near} \leftarrow$ SELECT_INPUT($\mathbf{x}_{fixed}, \mathbf{x}_{near}$)
{NOTE: $\mathbf{u}_{near} = \mathbf{u}_k$, SELECT_INPUT is$f_j$,
$u_j = f_j(\mathbf{x}_{fixed}, \mathbf{x}_{near})$, $j = 1, 2, 3$.}
7: $\mathbf{x}_{new} \leftarrow$ SIMULATION(model,$\mathbf{x}_{near}$,$\mathbf{u}_{near}$)
{NOTE: $\mathbf{x}_{new} = \mathbf{x}_{k+1}$}
8: T.add_vertex($\mathbf{x}_{new}$);
9: T.add_edge($\mathbf{x}_{near}$,$\mathbf{x}_{new}$);
10: **if** $|\mathbf{x}_{new} - \mathbf{x}_{goal}| < \min(\delta\dot{x}, \delta y)$ **then**
11: break;
12: **end if**
13: **end for**

We present the modified RRT algorithm (LaValle, 1998) in Algorithm 2 and a pictorial depiction is in Figure 30. We restrict the algorithm to the hopper model and to the region of attraction described by the major and minor axes on the x and y directions, but is easy to adapt to other cases. The inputs to the models are: the initial state $\mathbf{x}_{init}$, the goal state $\mathbf{x}_{goal}$, the major and minor axes of the ellipse as given by $\delta\dot{x} = s_1\sqrt{c}$ and $\delta y = s_2\sqrt{c}$, and maximum number of nodes allowed $N_p$. The tree is denoted by $T$. At item number 3, we generate a random state $\mathbf{x}_{rand}$ using the function RANDOM_STATE(). At item number 4, we use the function NEAREST_NEIGHBOR() to search for the nearest state $\mathbf{x}_{near}$ on the existing tree using the Euclidean distance. We compute the direction from the nearest point on the

tree to the random point as $\phi = \tan^{-1}((y_{rand} - y_{near})/(\dot{x}_{rand} - \dot{x}_{near}))$ (see Figure 30 (a)). Next at item number 5, the function NEW_FIXED_POINT() is used to choose a fixed point $\mathbf{x}_{fixed} = \mathbf{x}_{near} + \Delta\mathbf{x}_{near}$ where $\Delta\mathbf{x}_{near} = \{\delta\dot{x}\cos(\phi), \delta y \sin(\phi)\}$ (see Figure 30 (b)). Next at item number 6, we use the SELECT_INPUT() function, which is the same as the neural networks identified in section 6.5.2.2 to choose control actions $\mathbf{u}_{near}$. Next at item number 7, we use the function SIMULATION(), which is a forward simulation of the model, to grow the tree to the point $\mathbf{x}_{new}$ (see Figure 30 (c)). At item numbers 8 and 9, we add $\mathbf{x}_{new}$ as a vertex and connect it to $\mathbf{x}_{near}$ to form an edge on the tree T. We terminate the algorithm if the $\mathbf{x}_{goal}$ is within the region of attraction of $\mathbf{x}_{new}$ as shown at item number 10. In summary, the key modifications to the original RRT algorithm are: (1) at item number 5 where the randomly chosen direction is used to choose a fixed point, and (2) at item number 7 where forward simulation is used to update $\mathbf{x}_{new}$. Unlike the static RRT, popular for path planning without dynamics, we do feedback motion planning because $\mathbf{x}_{new}$ is obtained from $\mathbf{x}_{near}$ (feedback) and using the system dynamics. Once the tree T is built, we search it backwards from the goal to the start to find a feasible path.

## 6.6 Results

We present results for the hopper model described in section 6.3. As mentioned earlier, the Poincaré section is at the apex. There are two state variables at the apex $\mathbf{x}_k = \{\dot{x}, y\}$ and three control actions $\mathbf{u}_k = \{\theta, P^b, P^t\}$. The parameters for the orbital Lyapunov function are $s_1 = 1$ and $s_2 = 0.3$, and the rate of convergence $\alpha = 0.9$.

### 6.6.1 Trajectory optimizations for ROAs of multiple limit cycles

We choose the following 7 limit cycles characterized by fixed points $\mathbf{x}_i$s; $\{2, 1.3\}$, $\{2, 1.6\}$, $\{3, 1.5\}$, $\{4, 1.4\}$, $\{4, 1.6\}$, $\{5, 1.4\}$, and $\{5, 1.6\}$. We chose 105 initial conditions in the ROA for each limit cycle, totaling $105 \times 7 = 735$ points. For each point, we solve the optimization problem specified in section 6.5.1. We use the non-linear optimization software SNOPT (Gill et al., 2002) and single shooting method using *ode113* in MATLAB. On average each optimization took about 15 seconds on a laptop (circa 2012). Clearly, the optimization speed is too slow for real-time implementation. This necessitates offline computation of the control policy as given next.

### 6.6.2 Training the control policy

We use the data from the 7 limit cycles to train three neural networks, each corresponding to a control action ($\theta$, $P^b$, and $P^t$). Each neural network has 12 hidden layers and is trained using Levenberg-Marquardt algorithm with mean squared error as the performance criterion. The inputs to the neural networks are the limit cycles $\{\dot{x}_i, y_i\}$ $(i = 1, 2, 3, ..., 7)$, the $k$ perturbed initial conditions at the Poincaré section $\{\dot{x}_k, y_k\}$, and the corresponding control actions $\mathbf{u}_k$, where $k = 1, 2, 3, ..., 735$. Note that we could have also used a single neural network to map the three control actions (outputs) to the two states (inputs).

### 6.6.3 Testing the control policy

We describe how we test the control policy using the forward simulation. We choose a random limit cycle $\mathbf{x}_i$ and 451 random initial conditions $\mathbf{x}_k$ within the ROA for that limit cycle. Then, using the control law found earlier, we simulate the system for a single step starting from

each initial condition. Figure 31 shows the results for 3 limit cycles as a histogram of the percentage of points (total of 451) that lie within the various level sets of the orbital Lyapunov function after one step. Note that only (a) is within our training range but (b) and (c) are outside the training range, yet the test shows that all initial conditions are reduced to within $V(\mathbf{x}_k) < 0.3$. We also check that none of the initial conditions lead to an increase in $V$ at the subsequent steps. This verifies (using limited samples) that the chosen control policy is able to perform satisfactorily.

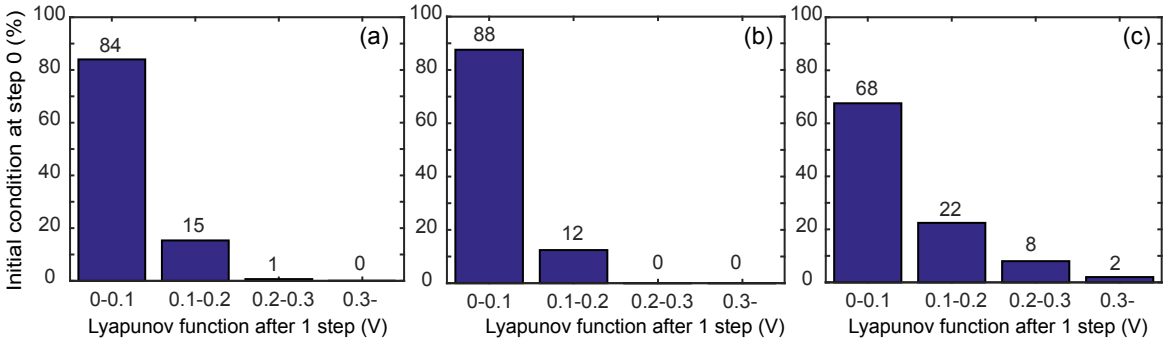


Figure 31: Testing the neural networks on 451 initial conditions inside the region of attraction for randomly chosen limit cycles. The histogram shows the percentage of the initial conditions that are within the ellipse after one step $V(x_k) = c$, (a) $\{3.5, 1.3\}$ (b) $\{2, 1.7\}$, (c) $\{7, 1.3\}$.

### 6.6.4 Feedback motion planning

We present the results for the feedback motion planning using the RRT discussed in Algorithm 2. We considered two scenarios: changing only the velocity as shown in Figure 32 and changing the velocity and the height as shown in Figure 33. We discuss these next.

First, we consider the problems of keeping the start and goal apex heights constant, but changing the start and goal apex horizontal velocities as shown in Figure 32. The cross denotes the start state and the solid circle indicates the goal state. The algorithm terminates when $V(\mathbf{x}_k) < 0.1$ for the fixed point at the goal state. Both these scenarios take 5 steps for the transition. The corresponding control actions $\mathbf{u}_k$ are shown in Figure 32 (c) and (d) with forces $P^t$ and $P^b$ multiplied by 0.001. To increase the velocity, the restitution force $P^t$ is non-zero while the compression force $P^b$ is zero. This is expected because to increase the velocity, energy needs to be supplied to the system, which comes from providing a non-zero restitution force. The roles of the forces are swapped for decreasing the velocity as expected by following the reverse logic. The foot placement angle shows a steady increase for increasing apex forward velocity (c). This can have the effect of increasing the apex height, while reducing the apex forward velocity without changing the total energy of the system (Hodgins and Raibert, 1991). However, from (a) it can be seen that the apex height is decreasing. This suggests that the foot placement angle control is compensating for the increased restitution force, which may increase both the apex horizontal velocity as well as the apex height. A decreased foot placement angle in (d) can now be reasoned by reversing the logic.

Second, we consider the problems of changing both the start and goal apex heights and horizontal velocities. In Figure 33 (a), the objective is to increase the apex horizontal velocity and apex height while in Figure 33 (b), the objective is to decrease the apex horizontal velocity and increase apex height. The first objective is achieved in 5 steps, while the second in 6 steps as shown. The trends in forces are similar to those in Figure 32 and follow similar logic as before. The increase in foot placement angle in (c) provides a means to increase the apex height, but the decrease in foot placement angle in (d) is a mechanism to compensate for the increased compressive force $P^b$ as noted earlier too.

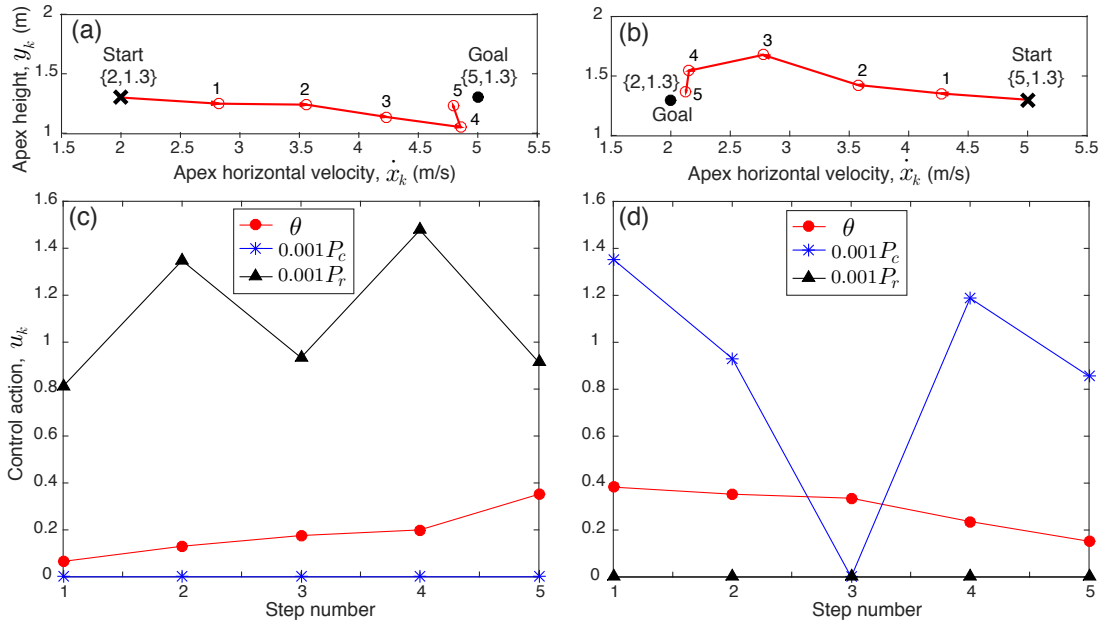


Figure 32: Results for feedback motion planning: (a,c) increasing apex horizontal velocity and (b,d) decreasing apex horizontal velocity, both for constant start and goal apex heights. Note that the angle is in radians and force is in N.

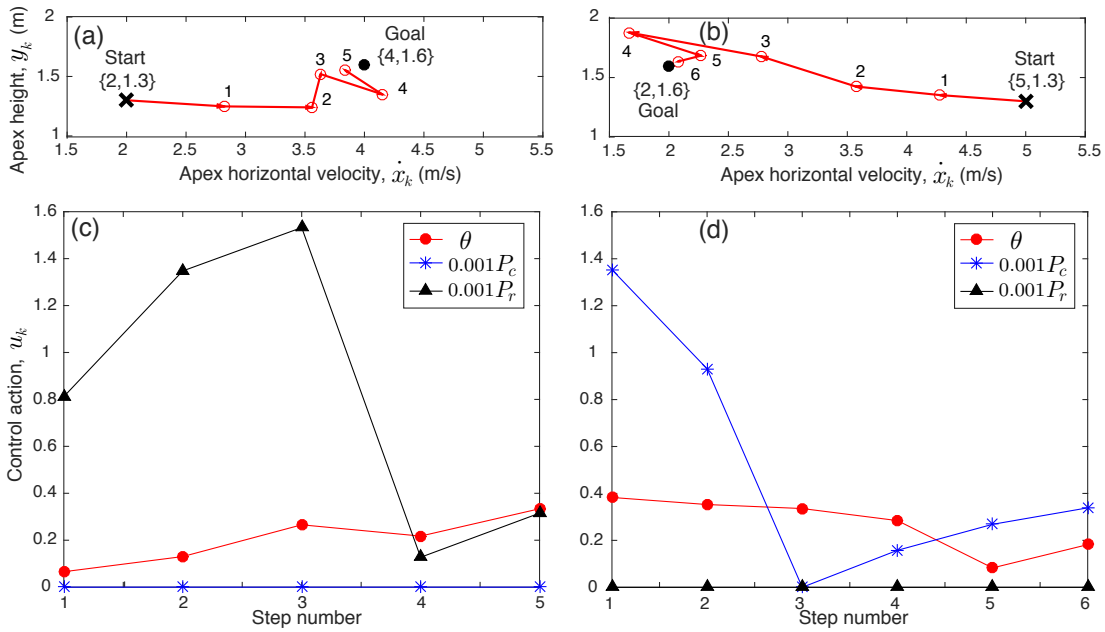


Figure 33: Results for feedback motion planning: (a,c) increasing apex horizontal velocity and apex height, and (b,d) decreasing apex horizontal velocity and increasing apex heights. Note that the angle is in radians and force is in N.

## 6.7 Discussion

We have presented a framework for feedback motion planning of legged robots by composing periodic gaits using regions of attraction and rapidly-exploring random trees. The efficacy of the approach was demonstrated using a hopper model in tasks involving increasing/decreasing the apex horizontal velocity and apex height.

The key benefit of our sampling-based approach for motion control is that we start by assuming a Lyapunov function and region of attraction and then find a control policy that

validates our assumption. This is in sharp contrast to existing approaches that start with a control policy, usually a linear one (e.g., linear quadratic regulator (Tedrake, 2009)), and then estimate the largest region of attraction. Depending on the assumed control policy and the nonlinearity in the dynamics, the existing methods may lead to a small region of attraction, which may significantly affect the motion planning. Furthermore, we are also able to ensure quick transitions by enforcing an exponential convergence condition on the orbital Lyapunov function.

Existing approaches of using a moving Poincaré section and estimating the region of attraction along the trajectory (Manchester, 2011) solve a trajectory tracking problem. In contrast, we are able to convert the traditional trajectory tracking problem into a regulation problem by doing step-to-step control using the Poincaré section. The latter is computationally cheaper and simpler than the former.

The average time taken by the trajectory optimization is 15 seconds per initial condition. This is too slow for online optimization. However, by using offline optimization for the control actions over multiple regions of attraction followed by regression to find a control policy, we are able to create a compact policy that is easy to store and use on hardware.

# CHAPTER 7

# A COMPUTATIONAL APPROACH FOR ONLINE PLANNING OF HOPPING MODEL ON STEPPING STONES USING APPROXIMATE STEP-TO-STEP DYNAMICS

(A significant portion of this chapter is reproduced from an accepted manuscript with the following citation:

Zamani, A. and Bhounsule, P. A.: Nonlinear model predictive control of hopping model using approximate step-to-step models for navigation on complex terrain. In 2020 IEEE International Conference on Intelligent Robots and Systems (IROS), 2020.)

## 7.1 Introduction

The ability to use discrete footholds to move around makes legged robots superior to wheeled robots on terrains with obstacles and ditches. However, planning and control of legged locomotion on such terrain presents a formidable computational challenge because of the complexity of locomotion dynamics (i.e., continuous and discontinuous dynamics) and the complexity of the terrain. Furthermore, if an objective function such as a cost metric is to be optimized, there is an added computational burden of evaluating multiple feasible solutions to determine the best one. In this work, we address the problem of navigating a series of stepping stones, a

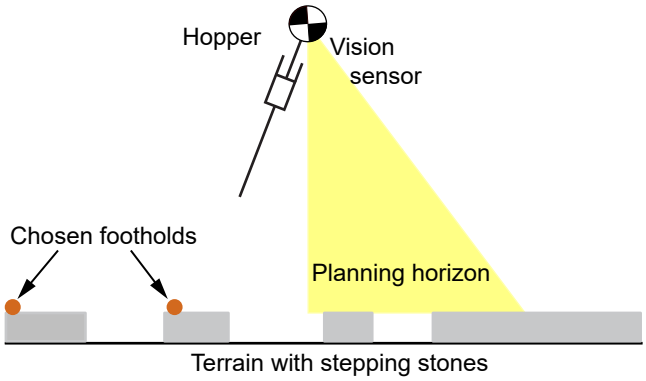


Figure 34: Conceptualization of the problem: The hopper has to negotiate a terrain consisting of stepping stones (gray rectangles). At mid-flight, the hopper plans the optimum strategy and the optimal steps for a planning horizon (yellow patch), then executes the optimum strategy for the first step until the next mid-flight. Then the hopper replans as before continuing the process until it finishes crossing the terrain.

benchmark problem in dynamic legged locomotion, while optimizing an objective (e.g., energy, time) while taking into account the robot dynamics during the planning phase.

We consider a realistic scenario that a robot might be subjected to and is shown in Figure 34. Here the robot can see a fixed distance ahead with an onboard vision sensor. The robot then plans the optimum number of steps and control strategy for that fixed distance ahead (planning). Next, the robot implements the control strategy for a few steps (control). Subsequently, the robot previews the next horizontal distance and the process continues until the robot reaches the end of the terrain. This formulation of the problem is well known as receding horizon control or model predictive control

## 7.2 Background and related work

We limit the review to work done in the area of legged locomotion on complex terrain consisting of obstacles or stepping stones.

A computationally simple approach is to follow a two-tier approach. First, a planner chooses footsteps that enable the robot to go from start to goal and second, a controller that controls the robot to move on those footholds. This method is computationally attractive because the planner only works with the complexity of the terrain and does not consider robot dynamics while the controller is only concerned with the robot dynamics and not the terrain complexity. Chestnutt et. al. (Chestnutt et al., 2005) used an A-star planner that minimizes heuristics for effort, risk, and/or number and complexity of steps taken to plan footsteps from start to goal. Huang et. al. (Huang et al., 2013) minimized the energy usage which they parameterized by step length, step width, and step steering in addition to the terrain profile within an A-star planning approach and then solved the footstep planning problem.

The main issue with this two-tiered approach is that the footstep planner does not consider the dynamics of the locomotion and hence there is no guarantee that the plan can actually be implemented. Moreover, since the planner is based on heuristics rather than actual costs (e.g., energy, time), the resulting solution is always sub-optimal.

A more complex but favorable approach is to find all feasible solutions considering the dynamics of the robot. These feasible solutions are enumerated as discrete control actions. Then using these discrete control actions and with a suitable planner, footholds are planned. Paris et. al. (Paris et al., 2016) considered the dynamics of the quadruped to map out the

reachable states from one step to the next. These reachable states were then searched within an A-star framework to plan jumps on rolling terrain. Campana and Laumond (Campana and Laumond, 2016) used a similar approach to first map out the feasible motion from step to step followed by footstep planning using probabilistic roadmaps (PRM). The advantage of this approach is that the footstep planner uses feasible solutions only. The disadvantage of using a sampling-based approach like A-star or PRM is that they do not handle boundary conditions that well (e.g., final state specified).

Some other works have focused on finding optimal controllers for a given foot placement locations. Rutschmann et. al. (Rutschmann et al., 2012) considered the problem of navigating a hopper on a series of pre-assigned footholds using model predictive or receding horizon control. The hopper plans for two steps in succession, but implements only one step and then replanning. Thereafter, the process continues until the hopper reaches the end of the terrain. Nguyen et. al. (Nguyen et al., 2016) considered the problem of a biped robot walking on a series of pre-assigned footholds using a control Lyapunov function for biped stability and control barrier function for proper foot placement.

Continuous optimization-based methods overcome boundary value problems. These are optimization formulations where the states and control actions are continuous variables. Continuous optimizations are generally used for trajectory optimization but may be extended to incorporate terrain profile and obstacles. Deits and Tedrake (Deits and Tedrake, 2014; Aceituno-Cabezas et al., 2016) considered the problem of a humanoid robot navigating a series of stepping stones. The continuous optimization formulation was based on kinematic reachability and avail-

ability of footholds. The resulting problem is a mixed-integer problem because each foothold needs to be placed only on one of the possible stepping stones. Furthermore, all the non-linearities were convexified, thus the problem could be solved relatively fast. However, their formulation does not consider the dynamics of the system. More recently Ding et. al. (Ding et al., 2018) considered the problem of legged robot jumping on an obstacle course. They also formulated a mixed-integer convex program and took into account the reachable space, dynamics, obstacles, and ground reaction forces by suitable approximations to speed up computations.

Our approach is also based on continuous optimization. Like some of the past approaches, we use the step-to-step model to first ascertain the reachable space. This step-to-step model is then approximated by a polynomial model to enable fast computation. The optimization formulation avoids integer variable by specifying a terrain cost that is added onto the cost of locomotion (i.e., the cost of transport, energy used per unit distance moved per unit weight). The resulting optimization problem is solved within a model predictive control framework where the robot scans a fixed distance ahead, then plans footholds, controls, and number of steps, implements the solution for the first step and continues the process until it reaches the end of the terrain. Our novel contributions are: (1) model predictive control framework that incorporates the terrain as a cost avoiding integer constraints, and (2) approximating the step-to-step dynamics with polynomials to enable fast calculations.

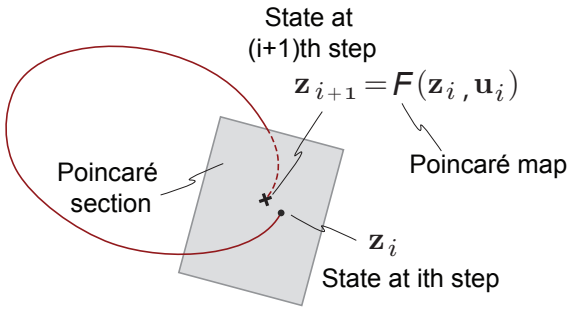


Figure 35: Dynamical systems tools use for analysis: A Poincaré section is an instant in the locomotion cycle (e.g., mid-flight phase for hopper). The state at $i$th step, $\mathbf{z}_i$ chosen at the Poincaré section. After applying controls $\mathbf{u}_i$ during the step, the system ends at the state $\mathbf{z}_{i+1}$ after one step. There is a function $\mathbf{F}$ that relates the states and controls at step $i$ to the state at step $i+1$.

## 7.3 Methods

### 7.3.1 Poincaré section and Poincaré map

We define some preliminaries that are used to analyze and consequently develop controllers for legged systems. Figure 35 (a) (red solid line) shows the trajectory of the robot in state space. We define the Poincaré section as an instant or event in the gait (e.g., the apex event occurs when the vertical velocity is zero during the flight phase for a hopper). We choose an initial condition at the Poincaré section at step $i$, $\mathbf{z}_i$, and trace its movement on the application of control $\mathbf{u}_i$ for a single step. At the Poincaré section at step $i+1$, the state of the robot is

$\mathbf{z}_{i+1}$. There is a function $\mathbf{F}$, known as the step-to-step map or Poincaré map that relates robot state between steps given by

$$\mathbf{z}_{i+1} = \mathbf{F}(\mathbf{z}_i, \mathbf{u}_i). \tag{7.1}$$

In most cases, there is no analytical solution to the Poincaré map $\mathbf{F}$. Hence once resorts to numerical simulation to obtain $\mathbf{z}_{i+1}$ (output) given the state $\mathbf{z}_i$ and control $\mathbf{u}_i$ (inputs). The numerical integration leads to a computational bottleneck for real-time control.

### 7.3.2 Approximation of the Poincaré map

One of the contributions of this paper is to create and then use an approximation of the Poincaré map ($F$) for control. There are two stages as shown in Figure 36 to find an approximation of the Poincaré map. First, for data generation, we choose a range of initial states on the Poincaré section ($\mathbf{z}_i$) and a range of control actions in the step ($\mathbf{u}_i$). Next, we use the forward simulation of the system given by $\mathbf{z}_{i+1} = \mathbf{F}(\mathbf{z}_i, \mathbf{u}_i)$, to obtain the numerical value for the system state at the next step $\mathbf{z}_{i+1}$. We show these points as blue dots in Figure 36 (a). Some of these inputs will lead to a failure, i.e., the state at the next step $\mathbf{z}_{i+1}$ is not defined. We ignore these initial conditions in our curve fitting. Second, for data fitting, we use a suitable defined regression model ($F$) to fit the data as shown by the gray plane in Figure 36 (b). Thus, our approximate model is

$$\mathbf{z}_{i+1} = F(\mathbf{z}_i, \mathbf{u}_i). \tag{7.2}$$

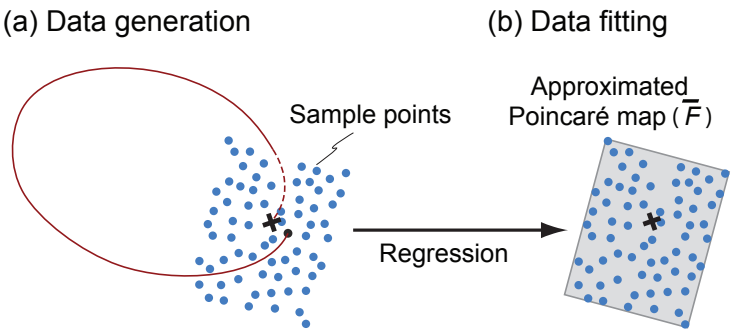


Figure 36: Regression to approximate the Poincaré map function $F$: (a) Data generation: For a range of values of $(\mathbf{z}_i, \mathbf{u}_i)$ at the Poincaré section, we use the forward simulation to generate the robot state at the next step $\mathbf{z}_{i+1}$, shown by blue dots on the left plot. (b) Data fitting: Using an assumed regression model (e.g., 2nd order polynomial), we fit an approximate function to the Poincaré map $\overline{\mathbf{F}}$, i.e., $\mathbf{z}_{i+1} = \overline{\mathbf{F}}(\mathbf{z}_i, \mathbf{u}_i)$, shown by the gray plane.

### 7.3.3 Stepping stones terrain

Another contribution of the paper is the incorporation of the stepping stones terrain with the optimization. Figure 37 (a) shows an example terrain consisting of stepping stones. The gray areas are the allowed foothold positions. Previous formulations stipulated the stepping areas as feasible regions and then found foot locations within those feasible regions, thus converting the problem to a mixed-integer problem.

Here we avoid specifying the stepping stones as a constraint, but specify a terrain cost as shown in Figure 37 (b). We specify a high cost for stepping into the gaps between the stepping stones. We fit a cubic spline to the cost. This formulation is computationally efficient because

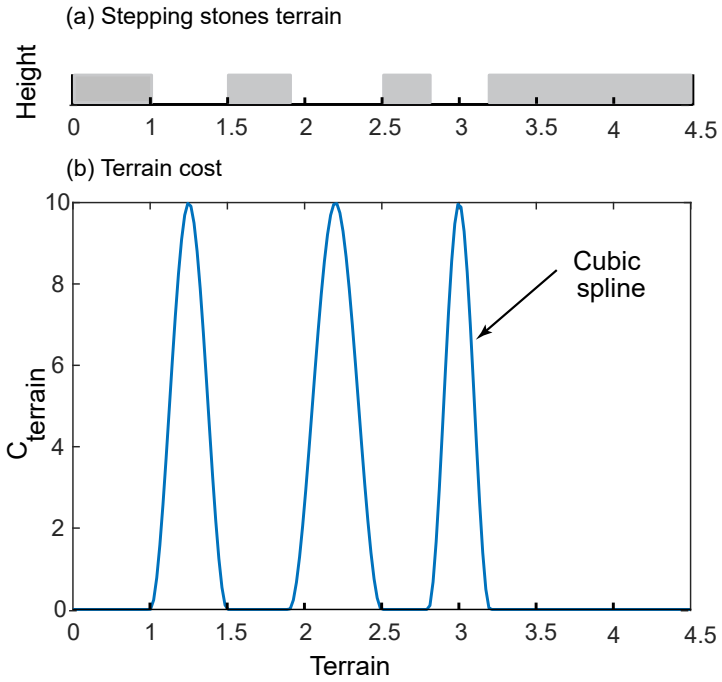


Figure 37: Incorporating the terrain profile as a cost: (a) Stepping stones are shown with gray rectangles and ditches are in the white spaces between the stepping stones. For feasible movement all footholds should be on the stepping stones. (b) We model the terrain with piecewise cubic polynomials. The cost on the stepping stones is zero and increases in the ditches.

it does not need to use branch-and-bounds as needed in mixed-integer problems and also allows us to use nonlinear optimization solvers.

### 7.3.4 Non-linear model predictive control problem formulation

The nonlinear model predictive control (MPC) optimization problem uses a nonlinear cost function consisting of the mechanical cost of transport (MCOT) and the terrain cost

$$\min_{N,\mathbf{u}_i,x_{ci}} \frac{\sum_{i=1}^{i=N} E_i(\mathbf{u}_i)}{mg \times x_N} + \sum_{i=1}^{i=N} C_{\text{ terrain}}(x_{ci})$$

$$\text{subject to: } \mathbf{z}_{i+1} = \mathbf{F}(\mathbf{z}_i, \mathbf{u}_i) \text{ (or } F) \tag{7.3}$$

$$\mathbf{u}_{\text{ min}} < \mathbf{u}_i < \mathbf{u}_{\text{ max}}$$

$$x_0 = 0;$$

$$x_N = d_{\text{ horizon}};$$

$$\dot{x}_0, y_0 \text{ are specified.}$$

where $i = 1, 2, ..., N$ is the planning horizon up to $N$ steps noting that $N$ is a decision variable, there are three controls at each step, $\mathbf{u}_i = \begin{bmatrix} \theta & P^b & P^t \end{bmatrix}_i$ thus a total of $3N$ decision variables, and the foot locations $x_{ci}$, a total of N decision variables. Also, the mechanical energy per step is $E_i = \int (|k(\ell - \ell_0)d\ell| + |P^t d\ell| + |P^b d\ell|)$, $C_{\text{ terrain}}(x_{ci})$ is the terrain cost as explained in section 7.3.3, and $d_{\text{ horizon}}$ is distance the robot can see and thus plan.

This problem can be solved as a parameter optimization problem. Since the decision variables depend on $N$, it is not possible to simultaneously optimize $N$ as well as other decision variables. Thus, the optimization has two loops. In the outer loop we optimize $N$ and in the inner loop (for a given $N$) we optimize the other decision variables. Since planning horizon

($d_{\text{horizon}}$) is short and there are bounds on the maximum and minimum step length, this often leads to solving for only a few number of steps (typically 2 or 3) so it can be done quite fast. We use SNOPT (Gill et al., 2002) to solve the problem either with the exact model **F** or with approximate model $F$. Then we choose the control for the first step and implement on the forward simulation and repeat the calculation until we reach the end of the terrain.

### 7.3.5 Model

We use the same model of hopping described in section 4.3.5 of chapter 4. We non-dimensionalize by dividing the variables with the terms given as follows: distance and leg length by $\ell_0$, time by $\sqrt{\ell_0/g}$, velocity by $\sqrt{g\ell_0}$, acceleration by $g$, and force by $mg$. Thus, the equations of motion in the flight phase are

$$\ddot{x} = 0, \qquad \ddot{y} = -1 \tag{7.4}$$

and in the stance phase (in polar coordinates) are

$$\ddot{\ell} = \ell\dot{\theta}^2 - \cos\theta + F \qquad \ell\ddot{\theta} = -2\dot{\ell}\dot{\theta} + \sin\theta \tag{7.5}$$

## 7.4 Results

### 7.4.1 Computer simulator

We built the forward simulator using MATLAB 2018b. It involves simulating a single step using phases/event stated in Figure 16 and Equation 7.4 and Equation 7.5. We integrate these

equations using dop853 (Hairer et al., 2000), a higher order Runge-Kutta method with an adaptive step size. The integrator is written as a C file and called by MATLAB through the MEX interface. This makes the simulation reasonably fast. The integrator tolerance is set $10^{-13}$. The integrator also has an in-built function *events* to detect a change in phase. Since we treat the simulator as a black-box, we also put checks to detect simulation failure. We consider the robot has failed if it meets any of the following conditions: (1) the horizontal velocity ($\dot{x}$) is negative indicating falling backward; (2) the height of the point mass ($y$) is below the ground; (3) during take-off from the ground, the vertical velocity is negative ($\dot{y}_{\text{take-off}} < 0$); and (4) at the apex of the flight phase it meets the following condition, $y_{\text{apex-of-flight-phase}} < \ell_0$. The last condition ensures that there is sufficient ground clearance for the swing leg. The only free parameter is $\beta = \dfrac{kl_0}{mg} = 40$.

### 7.4.2 Polynomial approximation of the Poincaré map and others

#### 7.4.2.1 Data generation

As mentioned in section 7.4.2 we need to approximate the Poincaré map. Our data range for the inputs are: apex horizontal velocity in the range $0.5 \leq \dot{x}_i \leq 2$ in 0.1667 increments; apex height in the range $1.01 \leq y_i \leq 1.5$ in 0.05 increments; foot placement angle in the range $2^\circ$ (0.035 rad) $\leq \theta_i \leq 45^\circ$ (0.7854 rad) in $4.77^\circ$ (0.0833 rad) increments; constant braking force in the range $0 \leq P_i^b \leq 7$ in 0.7778 increments; and constant thrust force in the range $0 \leq P_i^t \leq 7$ in 0.7778 increments. This generates $100,000$ input data points $\{\dot{x}_i, y_i, \theta_i, P_i^b, P_i^t\}$. For each input data point, we run a forward simulation (see section 7.4.1) and save the output data,

$\{\dot{x}_{i+1}, y_{i+1}\}$. Out of these $100,000$ data points, the robot failed $68,260$ times and successfully took a step $31,740$ times. This means that we only have $31,740$ data points to fit a closed-form expression. We use 75% or $23,805$ successful steps for training and the rest 25% or $7,935$ for testing. Each of the two outputs $\dot{x}_{i+1}$ and $y_{i+1}$ is curve fitted to the inputs, $\mathbf{z}_i = \{\dot{x}_i, y_i\}$ and $\mathbf{u}_i = \begin{Bmatrix} \theta_i & P_i^b & P_i^t \end{Bmatrix}$.

We also save other auxiliary data such as mechanical energy per step $E_i$, time from apex to touchdown, time for stance, time from takeoff to apex, and step length.

#### 7.4.2.2 Data fitting

We use a second order polynomial to fit the Poincaré map for each of the two outputs $\dot{x}_{i+1}$ and $y_{i+1}$. Each polynomial has 21 constants. We use MATLAB function *lsqnonlin* to fit each polynomial. In our testing, we found that 94.2% and 96.1% of the fit for the horizontal velocity and height at the apex, respectively, were within 90% accuracy. It is accepted that 90% accuracy is reasonably good for approximating the model for high fidelity control (Schwind and Koditschek, 2000).

In addition, we fit the auxiliary data with a third order polynomial. In our testing, we found that 93.1% fit for mechanical energy, 97% fit for time from apex to touchdown, 93.2% fit for time for stance, 92% for time from takeoff to apex, and 93.7% of the step length was within 90% accuracy.

### 7.4.3 Optimizations

For all optimizations, the robot starts at apex with initial forward velocity and height of $\dot{x}_0 = 1$ and $y_0 = 1.1$. The stepping stones were randomly assigned as shown in Figure 38. We did three optimization runs.

#### 7.4.3.1 Baseline optimization

We performed the optimization in section 7.3.4 but with $x_N = d_{\text{length}}$ and with the exact Poincaré map $\mathbf{F}$ (i.e., obtained from integration). This is the baseline simulation that we can use to compare our model predictive control results.

#### 7.4.3.2 Model predictive control

We did two model predictive control optimizations. Both of these with $x_N = d_{\text{horizon}} = 4$ m. The only difference is one used the exact Poincaré map $\mathbf{F}$ (MPC with exact model) and the other used the approximate Poincaré map $F$ (MPC with approximate model).

#### 7.4.3.3 Optimization results

Figure 38 shows results for baseline in (a), MPC with exact model in (b), and MPC with approximate model in (c). Each plot shows (1) the number of steps (2) the footstep locations, (3) the vertical height, and (4) the horizontal velocity for the terrain. All three runs produced the same number of steps and approximately the same step locations. However, these solutions differed in the kinematics; the baseline and MPC with exact model had faster speeds and lower jump height compared to MPC with approximate model.

Figure 39 shows the braking force in (a), the thrust force in (b), the foot placement angle in (c), and the Mechanical Cost of Transport in (d) as a function of step number. The braking

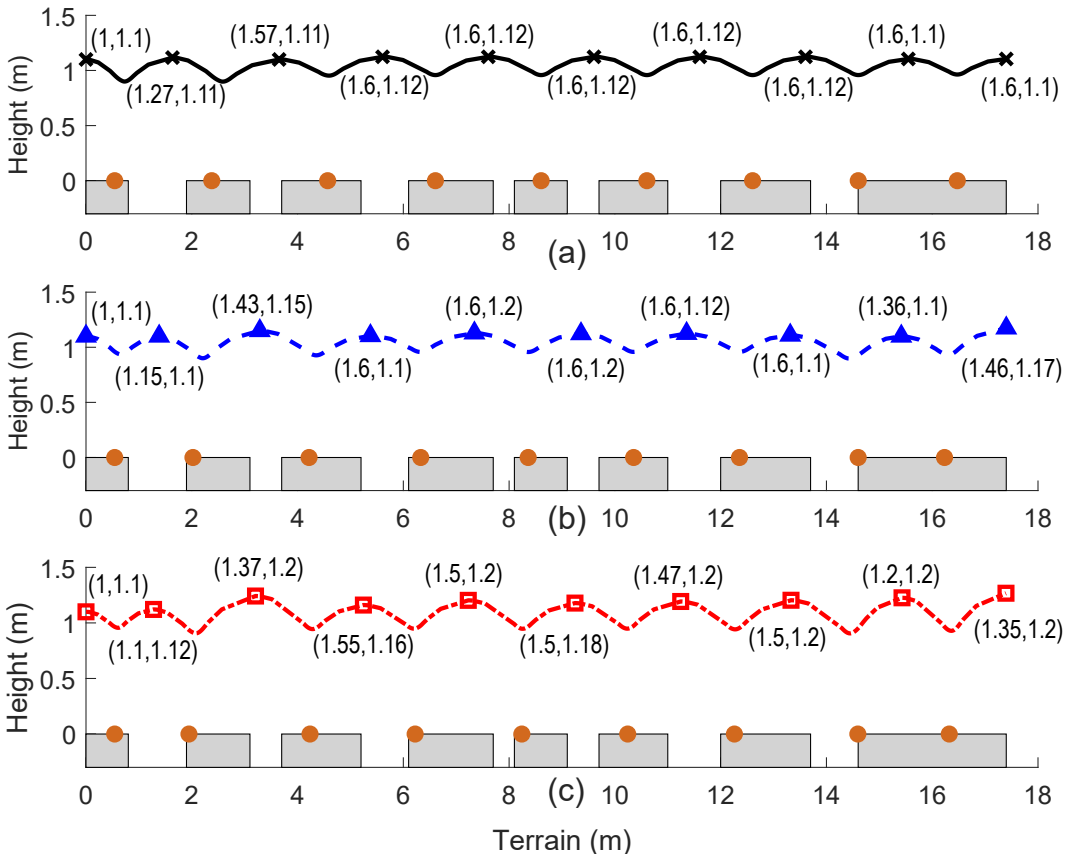


Figure 38: Kinematic data for the stepping stones terrain: (a) Baseline with exact model (b) MPC with exact model, and (c) MPC with approximate model. The stepping stones terrain is shown as gray blocks. The foot placement location is shown as a brown dot on the gray blocks. The trajectory is shown for each of the optimization cases with the apex velocity and apex height at step $i$ marked as $(\dot{x}_i, y_i)$.

force is responsible for extracting energy from the system. It can be seen that MPC with approximate model has a higher average force than baseline which is greater than MPC with exact model. The thrust force is responsible for adding energy to the system. It can be seen that baseline provides on average a greater thrust force than the MPC with approximate model which is greater than MPC with exact model. The foot placement angle is almost the same except at the end where MPC has a larger foot placement angle compared to the baseline. Finally, the MCOT indicates that the baseline is the cheapest after MPC with exact model and finally MPC with approximate model. However, the MPC with approximate model is within 25% of the baseline energetics. These results indicate that MPC formulations give results close to the baseline. Furthermore, the approximate model does reasonably well in terms of control and energetics when compared with baseline and exact models.

Overall, the MCOT for baseline is 0.085, MPC with exact model is 0.092, and MPC with approximate model is 0.11. These results indicate with limited planning such as in MPC the solution is 10% worst than the baseline solution where the entire terrain is known in advance. Also, the MPC with approximate model with 10% modeling error is within 20% of the exact model. We found that optimization time for MPC with approximate model is always faster than MPC with exact model. The speedup is between 2 and 4. This can be notable when real-time planning using MPC is desired.

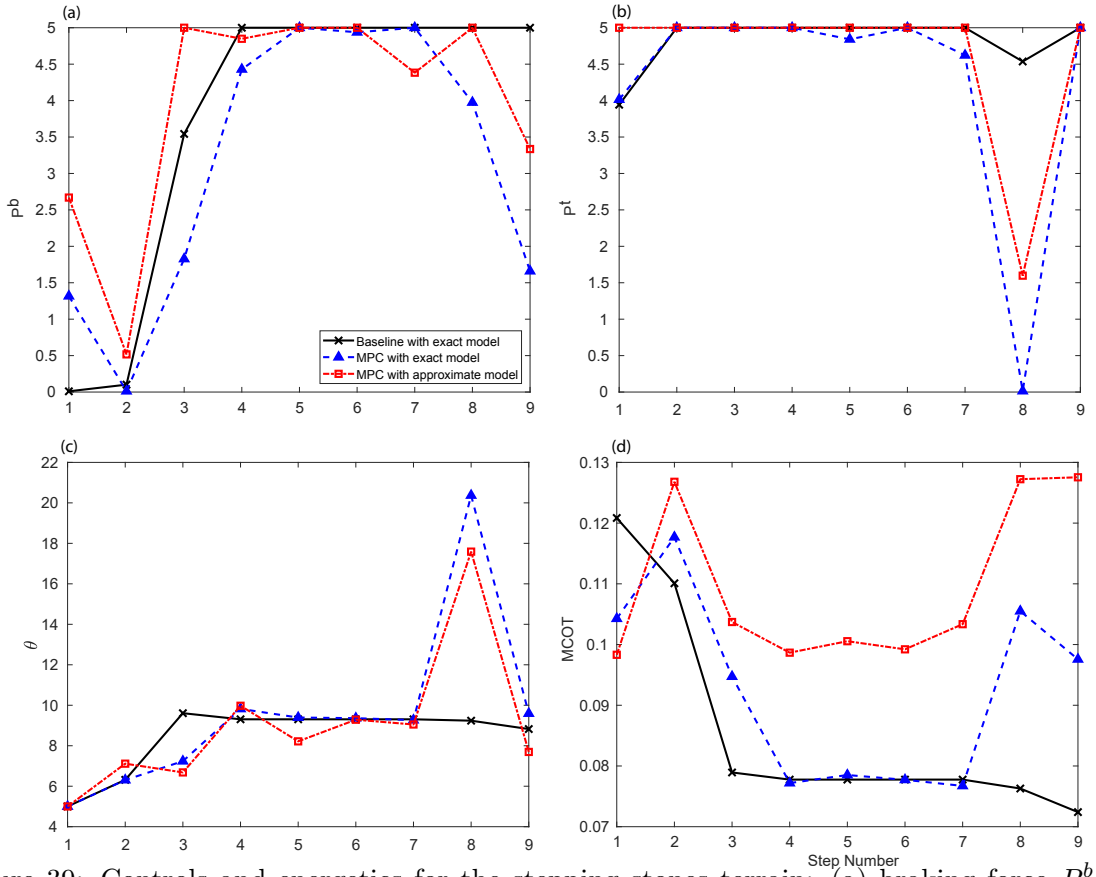


Figure 39: Controls and energetics for the stepping stones terrain: (a) braking force $P_i^b$, (b) thrust force $P_i^t$, (c) foot placement angle $\theta_i$, and (d) Mechanical Cost of Transport $\text{MCOT}_i$.

## 7.5 Discussion

In this paper, we have shown that model predictive control (MPC) framework is a viable approach for control of legged locomotion on stepping stones terrain; MPC performs as well as a baseline (ideal) controller that plans the entire terrain before execution in terms of energetics and optimal solution. Furthermore, approximating the step-to-step dynamics with low order polynomials enables faster planning than using the exact model based on integrating the equations of motion.

Our MPC formulation is unique in a few ways. We plan a fixed distance ahead compared to a few steps ahead. The former is the natural choice when we are optimizing the Cost of Transport which is based on distance rather than number of steps. We incorporate the terrain as a cost compared to other formulations where it is incorporated as a constraint. In our cases, the resulting optimal control problem has all real numbers while in the latter there are integers and real values making it a relatively harder optimization (Deits and Tedrake, 2014).

Using a continuous-time based formulation we are able to satisfy the boundary conditions (e.g., position, velocity constraint) which is not possible with sampling-based formulation (e.g., A-star, Rapidly exploring random trees). However, one caveat is that we need to do multiple optimizations for different step numbers to find the optimal step number. This is usually not a problem with a small preview window as we have in MPC formulations.

MPC is an attractive approach for movement on stepping stones due to its anticipatory nature. However, it is paramount that the computations are done quickly since it is an online method. Since planning the control over continuous time is time consuming we use step-to-step

models, computed offline, for online control. These models help to map the reachable space over one step. Another advantage is that these maps are continuous functions of the states and control and thus can be approximated using low order polynomials as done here. Once we have these low order polynomials, real-time planning is feasible. Also, since the plans are computed once per step, they can be evaluated at a lower speed, about half step time, by modest computational resources.

# CHAPTER 8

# CONCLUSIONS AND FUTURE WORKS

This chapter presents a summary of the author's effort toward increasing stability and agility of legged robots and possible directions for future work.

## 8.1 Summary

We presented the orbital control Lyapunov function (OCLF) for exponential stabilization of the simplest walker in chapter 2. Our control framework could significantly enlarge the basin of attraction and enhance the robustness over passive dynamic walking. We also designed a one-step dead-beat controller and an eigenvalue-based controller and compared their performance to the OCLF controller. Our results showed that all three controllers led to the same robustness as checked by calculating the average number of steps the model could successfully walk without failure. The dead-beat controller was more energy-efficient and had lower maximum torque than the OCLF controller, but was more sensitive to modeling errors. The OCLF controller performed better than the eigenvalue-based controller for the same rate of convergence in terms of energy efficiency and maximum torque. Our findings demonstrated that the OCLF controller with proper rate of convergence can offer a good compromise among energy-efficiency, robustness, and sensitivity to modeling errors.

In chapter 3, we modified the simplest walking model described in chapter 2 by adding a hip spring and a telescoping linear actuator. We examined how foot placement control and

ankle push-off control affect the stabilization of bipedal gaits considering two stability criteria, the one-step dead-beat stabilization and the OCLF stabilization. To this end, we did a forward simulation of the model walking on uneven terrain consisting of either step up or step down. Our results showed that the model was able to successfully walk more steps for step down disturbance than step up disturbance. For step down disturbance, the control strategy was to decrease the ankle push-off or to increase the foot placement angle, but for step up disturbance the control strategy was to maintain the same ankle push-off or to decrease the foot placement angle. We also found that it is more energy-efficient to utilize the ankle push-off control for step down disturbance and the foot placement control for step up disturbance.

Chapter 4 presented a data-driven framework to find control policies for an assumed region of attraction at the poincaré section that guarantees exponential convergence to the nominal motion using an orbital Laypunov function. We used a model of hopping to test our framework. We performed trajectory optimization for sampled points in the region of attraction to find control actions followed by tabulating the initial conditions and associated control actions. We then used regression and deep learning neural networks to fit a control policy for each control action as a function of initial states. The control policy was verified using a finer grid, and robustness checks to unmodeled dynamics, noisy sensors and actuators, and external disturbance were carried out. Our framework was able to find control polices, mostly nonlinear ones, for the assumed region of attraction. This is particularly useful for nonlinear systems where having a linear control policy results in a small region of attraction.

In chapter 5, we proposed a method for creating agile gaits by sequentially composing limit cycles using heuristics. The approach was based on two main ideas: first, we created funnels using the region of attraction at the Poincaré section and second, we used the OCLF to ensure fast switching between limit cycles. The fast switching among limit cycles was possible if there existed enough overlap between the region of attraction of the limit cycles. We tested the method on a model of hopping. The model was able to go from the given initial states to the goal states in a number of steps determined by the user (heuristics).

In chapter 6, we presented a framework to create agile gaits by sequentially composing limit cycles using rapidly-exploring random trees (RRT). In fact, we probabilistically filled the state space with regions of attraction and associated stabilization controllers developed in chapter 4, and then used the RRT algorithm for feedback motion planning. The framework was demonstrated on a hopping model to realize height and velocity changes between steps.

Finally, in chapter 7 we proposed a continuous-time based optimization method within model predictive control framework for navigation of the hopping model on stepping stones. Unlike discrete search based approaches like A-star, our approach could easily handle the boundary value problem. In order to deal with computational complexity of computing controls online, we used a data-driven technique to approximate the step-to-step dynamics with low-order polynomials and used the approximate models in our optimization. Also, we avoided specifying the stepping stones as constraints which would lead to a mix-integer problem. Rather, we specified a high cost for placing the foot on the gaps between the stepping stones. This allowed us to

use nonlinear optimization solvers. Our findings demonstrated that our approach is a viable solution for control of legged robots on stepping stones.

## 8.2 Future Work

The step-to-step control approach developed in this thesis may fail in the face of relatively large disturbances. In our approach, the system state measurement is made at the Poincaré section followed by choosing controls which remain constant until the next Poincaré section. If relatively large disturbances occur between steps, the system may fail before reaching the next Poincaré section. A potential solution to this issue is to consider multiple sections along the system trajectory within a step and update the controls more than once at each step.

We chose simple Lyapunov functions to construct elliptical regions of attraction of similar size and shape and probabilistically filled the state space with these regions. An extension of this work is to choose complex Lyapunov functions to create regions of attraction of different size and shape based on the task objectives. For example, for tasks requiring fast speed changes between steps, the regions of attraction should be directed along the velocity axis to quickly switch speeds, or for tasks involving jumping over obstacles, the regions of attraction should be directed along the height axis.

We used rapidly-exploring random trees (RRT) to find switching for tasks involve changing in velocity or height. An extension of this work is to use optimality variants of RRT such as RRT-star, RRTX, and bidirectional RRT-star and consider various tasks such as gait switching, obstacle avoidance, and accelerating/decelerating while optimizing minimum time, steps, energy, etc.

Perhaps the hardware evaluation of the frameworks presented in this thesis is the most important future work. A hopping robot is currently being developed in our lab. The robot's leg is a symmetric five-bar linkage and has two direct-drive motors at the hip, reducing the leg mass. The robot is mounted on a boom to impart lateral stability. To implement the frameworks on this robot, the mathematics transforming the axial force of the model, studied in chapter 4, into joint torques need to be derived.

## CITED LITERATURE


Aceituno-Cabezas, B., Cappelletto, J., Grieco, J. C., and Fernández-López, G.: A generalized mixed-integer convex program for multilegged footstep planning on uneven terrain. arXiv preprint arXiv:1612.02109, 2016.

Ames, A. D., Galloway, K., Sreenath, K., and Grizzle, J. W.: Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. IEEE Transactions on Automatic Control, 59(4):876–891, 2014.

Antsaklis, P. and Michel, A.: Linear systems. Birkhauser, 2006.

Betts, J. T.: Practical methods for optimal control and estimation using nonlinear programming, volume 19. Siam, 2010.

Betts, J.: Survey of numerical methods for trajectory optimization. Journal of Guidance control and dynamics, 21(2):193–207, 1998.

Bhounsule, P. A.: Foot placement in the simplest slope walker reveals a wide range of walking solutions. IEEE Transactions on Robotics, 30(5):1255–1260, 2014.

Bhounsule, P. A.: Control of a compass gait walker based on energy regulation using ankle push-off and foot placement. Robotica, 33(06):1314–1324, 2015.

Bhounsule, P. A., Ameperosa, E., Miller, S., Seay, K., and Ulep, R.: Dead-beat control of walking for a torso-actuated rimless wheel using an event-based, discrete, linear controller. In ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, pages V05AT07A042–V05AT07A042. American Society of Mechanical Engineers, 2016.

Bhounsule, P. A., Cortell, J., Grewal, A., Hendriksen, B., Karssen, J. D., Paul, C., and Ruina, A.: Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge. International Journal of Robotics Research, 2014.

Bhounsule, P. A., Ruina, A., and Stiesberg, G.: Discrete-decision continuous-actuation control: balance of an inverted pendulum and pumping a pendulum swing. Journal of Dynamic Systems, Measurement, and Control, 137(5):051012, 2015.

Bhounsule, P. A. and Zamani, A.: Stable bipedal walking with a swing-leg protraction strategy. Journal of Biomechanics, 51:123–127, 2017.

Burridge, R. R., Rizzi, A. A., and Koditschek, D. E.: Sequential composition of dynamically dexterous robot behaviors. The International Journal of Robotics Research, 18(6):534–555, 1999.

Byl, K., Strizic, T., and Pusey, J.: Mesh-based switching control for robust and agile dynamic gaits. In American Control Conference (ACC), 2017, pages 5449–5455. IEEE, 2017.

Byl, K. and Tedrake, R.: Approximate optimal control of the compass gait on rough terrain. In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pages 1258–1263. IEEE, 2008.

Byl, K. and Tedrake, R.: Metastable walking machines. The International Journal of Robotics Research, 28(8):1040–1064, 2009.

Campana, M. and Laumond, J.-P.: Ballistic motion planning. In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1410–1416. IEEE, 2016.

Cao, Q., Van Rijn, A. T., and Poulakakis, I.: On the control of gait transitions in quadrupedal running. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 5136–5141. IEEE, 2015.

Chestnutt, J., Lau, M., Cheung, G., Kuffner, J., Hodgins, J., and Kanade, T.: Footstep planning for the honda asimo humanoid. In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, pages 629–634. IEEE, 2005.

Chevallereau, C., Grizzle, J., and Shih, C.: Asymptotically stable walking of a five-link underactuated 3-d bipedal robot. IEEE Transactions on Robotics, 25(1):37–50, 2009.

Collins, S. and Ruina, A.: A bipedal walking robot with efficient and human-like gait. In Proceeding of 2005 International Conference on Robotics and Automation, Barcelona, Spain, 2005.

Da, X., Hartley, R., and Grizzle, J. W.: Supervised learning for stabilizing underactuated bipedal robot locomotion, with outdoor experiments on the wave field. In Robotics and Automation (ICRA), 2017 IEEE International Conference on, pages 3476–3483. IEEE, 2017.

Deits, R. and Tedrake, R.: Footstep planning on uneven terrain with mixed-integer convex optimization. In 2014 IEEE-RAS international conference on humanoid robots, pages 279–286. IEEE, 2014.

Ding, Y., Li, C., and Park, H.-W.: Single leg dynamic motion planning with mixed-integer convex optimization. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1–6. IEEE, 2018.

Dingwell, J. B. and Kang, H. G.: Differences between local and orbital dynamic stability during human walking. Journal of biomechanical engineering, 129(4):586–593, 2007.

Duperret, J. M., Kenneally, G. D., Pusey, J., and Koditschek, D. E.: Towards a comparative measure of legged agility. In Experimental Robotics, pages 3–16. Springer, 2016.

Garcia, M., Chatterjee, A., Ruina, A., and Coleman, M.: The simplest walking model: Stability, complexity, and scaling. ASME J. of Biomech. Eng., 120:281–288, 1998.

Gauthier, G. and Boulet, B.: Convergence analysis of terminal ilc in the z domain. In Proceedings of the American Control Conference, volume 1, page 184, 2005.

Gill, P., Murray, W., and Saunders, M.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Journal on Optimization, 12(4):979–1006, 2002.

Goswami, A., Espiau, B., and Keramane, A.: Limit cycles in a passive compass gait biped and passivity-mimicking control laws. Autonomous Robots, 4(3):273–286, 1997.

Goswami, A., Thuilot, B., and Espiau, B.: A study of the passive gait of a compass-like biped robot: symmetry and chaos. The International Journal of Robotics Research, 17(12):1282–1301, 1998.

Grizzle, J., Abba, G., and Plestan, F.: Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. IEEE Transactions on Automatic Control, 46(1):51–64, 2001.

Hairer, E., NORSETT, S., and Wanner, G.: Solving Ordinary, Differential Equations I, Nonstiff problems/E. Hairer, SP Norsett, G. Wanner, with 135 Figures, Vol.: 1. Number BOOK. 2Ed. Springer-Verlag, 2000, 2000.

Haynes, G. C. and Rizzi, A. A.: Gaits and gait transitions for legged robots. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 1117–1122. IEEE, 2006.

Heim, S. and Spröwitz, A.: Beyond basins of attraction: Quantifying robustness of natural dynamics. IEEE Transactions on Robotics, 35(4):939–952, 2019.

Hobbelen, D. G. and Wisse, M.: Ankle actuation for limit cycle walkers. The International Journal of Robotics Research, 27(6):709–735, 2008.

Hobbelen, D. and Wisse, M.: A disturbance rejection measure for limit cycle walkers: The gait sensitivity norm. IEEE Transactions on robotics, 23(6):1213–1224, 2007.

Hobbelen, D. and Wisse, M.: Limit cycle walking. Humanoid Robots Human-like Machines, pages 277–294, 2007.

Hodgins, J. K. and Raibert, M.: Adjusting step length for rough terrain locomotion. IEEE Transactions on Robotics and Automation, 7(3):289–298, 1991.

Hsu, C. S.: Cell-to-cell mapping: a method of global analysis for nonlinear systems, volume 64. Springer Science & Business Media, 2013.

Huang, W., Kim, J., and Atkeson, C. G.: Energy-based optimal step planning for humanoids. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 3124–3129. IEEE, 2013.

Iida, F. and Tedrake, R.: Minimalistic control of biped walking in rough terrain. Autonomous Robots, 28(3):355–368, 2010.

Kim, M. and Collins, S. H.: Once-per-step control of ankle push-off work improves balance in a three-dimensional simulation of bipedal walking. IEEE Transactions on Robotics, 33(2):406–418, 2017.

Koditschek, D. and Robert, J.: Templates and anchors: Neuromechanical hypotheses of legged locomotion on land. The Journal of Experimental Biology, 2(12):3–125, 1999.

Kuo, A.: Energetics of actively powered locomotion using the simplest walking model. Journal of Biomechanical Engineering, 124:113–120, 2002.

Kuo, A. D.: Stabilization of lateral motion in passive dynamic walking. The International journal of robotics research, 18(9):917–930, 1999.

LaValle, S. M.: Rapidly-exploring random trees: A new tool for path planning. 1998.

Manchester, I. R.: Transverse dynamics and regions of stability for nonlinear hybrid limit cycles. IFAC Proceedings Volumes, 44(1):6285–6290, 2011.

Manchester, I. R., Tobenkin, M. M., Levashov, M., and Tedrake, R.: Regions of attraction for hybrid limit cycles of walking robots. arXiv preprint arXiv:1010.2247, 2010.

Manchester, I., Mettin, U., Iida, F., and Tedrake, R.: Stable dynamic walking over uneven terrain. The International Journal of Robotics Research, 30(3):265–279, 2011.

Mandersloot, T., Wisse, M., and Atkeson, C. G.: Controlling velocity in bipedal walking: A dynamic programming approach. In Humanoid Robots, 2006 6th IEEE-RAS International Conference on, pages 124–130. IEEE, 2006.

McGeer, T.: Passive dynamic walking. The International Journal of Robotics Research, 9(2):62–82, 1990.

McGeer, T.: Passive dynamic biped catalogue. In In Proc. of 2nd International Symposium on Experimental Robotics, pages 465–490, 1991.

McGeer, T.: Dynamics and control of bipedal locomotion. Journal of Theoretical Biology, 163(3):277–314, 1993.

Mombaur, K., Longman, R., Bock, H., and Schlöder, J.: Open-loop stable running. Robotica, 23(01):21–33, 2005.

Nguyen, Q., Hereid, A., Grizzle, J. W., Ames, A. D., and Sreenath, K.: 3d dynamic walking on stepping stones with control barrier functions. In 2016 IEEE 55th Conference on Decision and Control (CDC), pages 827–834. IEEE, 2016.

Owaki, D., Koyama, M., Yamaguchi, S., Kubo, S., and Ishiguro, A.: A two-dimensional passive dynamic running biped with knees. In International Conference on Robotics and Automation, Anchorage, Alaska, USA, pages 5237–5242, 2010.

Paris, V., Strizic, T., Pusey, J., and Byl, K.: Tools for the design of stable yet nonsteady bounding control. In 2016 American Control Conference (ACC), pages 4822–4828. IEEE, 2016.

Park, H.-W., Wensing, P. M., and Kim, S.: Online planning for autonomous running jumps over obstacles in high-speed quadrupeds. In Proceedings of Robotics Science and Systems (RSS), Rome, Italy, 2015.

Poulakakis, I., Papadopoulos, E., and Buehler, M.: On the stability of the passive dynamics of quadrupedal running with a bounding gait. The International Journal of Robotics Research, 25(7):669–687, 2006.

Prajna, S., Papachristodoulou, A., and Parrilo, P. A.: Introducing sostools: A general purpose sum of squares programming solver. In Decision and Control, 2002, Proceedings of the 41st IEEE Conference on, volume 1, pages 741–746. IEEE, 2002.

Pratt, J., Carff, J., Drakunov, S., and Goswami, A.: Capture point: A step toward humanoid push recovery. In 2006 6th IEEE-RAS international conference on humanoid robots, pages 200–207. IEEE, 2006.

Pratt, J., Koolen, T., Boer, T. D., Rebula, J., Cotton, S., Carff, J., Johnson, M., and Neuhaus, P.: Capturability-based analysis and control of legged locomotion. part 2: Application to m2v2, a lower-body humanoid. International Journal of Robotics Research, to appear, 2012.

Raibert, M. H. and Wimberly, F. C.: Tabular control of balance in a dynamic legged system. IEEE Transactions on systems, man, and Cybernetics, (2):334–339, 1984.

Raibert, M.: Legged robots that balance. MIT press Cambridge, MA, 1986.

Ruina, A., Bertram, J., and Srinivasan, M.: A collisional model of the energetic cost of support work qualitatively explains leg sequencing in walking and galloping, pseudo-elastic leg behavior in running and the walk-to-run transition. Journal of theoretical biology, 237(2):170–192, 2005.

Rutschmann, M., Satzinger, B., Byl, M., and Byl, K.: Nonlinear model predictive control for rough-terrain robot hopping. In 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1859–1864. IEEE, 2012.

Santos, C. P. and Matos, V.: Gait transition and modulation in a quadruped robot: A brainstem-like modulation approach. Robotics and Autonomous Systems, 59(9):620–634, 2011.

Schwab, A. and Wisse, M.: Basin of attraction of the simplest walking model. In Proceedings of the ASME design engineering technical conference, volume 6, pages 531–539, 2001.

Schwind, W. J. and Koditschek, D. E.: Approximating the stance map of a 2-dof monoped runner. Journal of Nonlinear Science, 10(5):533–568, 2000.

Schwind, W. J.: Spring loaded inverted pendulum running: A plant model. Doctoral dissertation, University of Michigan, 1998.

Slotine, J.-J. E., Li, W., et al.: Applied nonlinear control. prentice-Hall Englewood Cliffs, NJ, 1991.

Spong, M.: Passivity based control of the compass gait biped. In In Proc. of IFAC World Congress, Beijing, China, 1999.

Srinivasan, M.: Why walk and run: energetic costs and energetic optimality in simple mechanics-based models of a bipedal animal. Doctoral dissertation, Cornell University, 2006.

Steinkamp, P.: A statically-unstable passive hopper: design evolution. Journal of Mechanisms and Robotics, 9(1):011006–011006–7, 2017.

Strogatz, S.: Nonlinear dynamics and chaos. Addison-Wesley Reading, 1994.

Tedrake, R.: LQR-Trees: Feedback motion planning on sparse randomized trees. In Proceedings of Robotics Science and Systems (RSS), Zaragoza, Spain, 2009.

Tedrake, R., Manchester, I. R., Tobenkin, M., and Roberts, J. W.: Lqr-trees: Feedback motion planning via sums-of-squares verification. The International Journal of Robotics Research, 29(8):1038–1052, 2010.

Veer, S., Motahar, M., and Poulakakis, I.: Generation of and Switching among Limit-Cycle Bipedal Walking Gaits. In Proceedings of the 56th IEEE Conference on Decision and Control, Melbourne, Australia, 2017.

Veer, S. and Poulakakis, I.: Ultimate boundedness for switched systems with multiple equilibria under disturbances. arXiv preprint arXiv:1809.02750, 2018.

Veer, S. and Poulakakis, I.: Safe adaptive switching among dynamical movement primitives: Application to 3d limit-cycle walkers. In 2019 International Conference on Robotics and Automation (ICRA), pages 3719–3725. IEEE, 2019.

Westervelt, E. R., Grizzle, J. W., and De Wit, C. C.: Switching and pi control of walking motions of planar biped walkers. IEEE Transactions on Automatic Control, 48(2):308–312, 2003.

Whitman, E. C.: Coordination of Multiple Dynamic Programming Policies for Control of BipedalWalking. Doctoral dissertation, Carnegie Mellon University, 2013.

Wieber, P.-B.: Viability and predictive control for safe locomotion. In 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1103–1108. IEEE, 2008.

Wisse, M., Atkeson, C., and Kloimwieder, D.: Swing leg retraction helps biped walking stability. In Proceedings of 2005 IEEEE-RAS International Conference on Humanoid Robots, Tsukuba, Japan, 2005.

Zaytsev, P., Wolfslag, W., and Ruina, A.: The boundaries of walking stability: Viability and controllability of simple models. IEEE Transactions on Robotics, 34(2):336–352, 2018.

# APPENDIX

Reprint Permissions

Chapter 2, Journal of Mechanism and Robotics (Publisher: ASME):

## Rights and Permissions

### ASME Journals Digital Submission Tool Guidelines and Information

#### Rights and Permissions

**Assignment of Copyright**

ASME requests that authors/copyright owners assign copyright to ASME in order for a journal paper to be published by ASME. Authors exempt from this request are direct employees of the U.S. Government, whereby papers are not subject to copyright protection in the U.S., or non-U.S. government employees, whose governments hold the copyright to the paper.

For more information on copyright, please view the Copyright Transfer information page.

**Retained Rights of Authors**

Authors retain all proprietary rights in any idea, process, procedure, or articles of manufacture described in the Paper, including the right to seek patent protection for them. Authors may perform, lecture, teach, conduct related research, display all or part of the Paper, and create derivative works in print or electronic format. Authors may reproduce and distribute the Paper for non-commercial purposes only. Non-commercial applies only to the sale of the paper per se. For all copies of the Paper made by Authors, Authors must acknowledge ASME as original publisher and include the names of all author(s), the publication title, and an appropriate copyright notice that identifies ASME as the copyright holder.

**Permissions**

Once your paper has been published by ASME, you may wish to submit it for inclusion in a non-ASME publication or to incorporate some or all of its elements in another work. Since ASME is the legal holder of copyright for its papers, it will be necessary for you to secure the permission of the copyright holder to have its material published in another source.

In this case, for permission to have your paper - in whole or in part, as is or adapted - published elsewhere, please submit your request here.

**Questions**

The ASME Publishing staff is available to discuss current ASME policy on permissions and rights. Please feel free to contact us with any questions or comments at: permissions@asme.org.

# APPENDIX (Continued)

Chapter 3, International Conference on Intelligent Robots and Systems (Publisher: IEEE):

# APPENDIX (Continued)

Chapter 4, Biomimetics (Publisher: MDPI):

## Copyrights

### Copyright and Licensing

For all articles published in MDPI journals, copyright is retained by the authors. Articles are licensed under an open access Creative Commons CC BY 4.0 license, meaning that anyone may download and read the paper for free. In addition, the article may be reused and quoted provided that the original published version is cited. These conditions allow for maximum use and exposure of the work, while ensuring that the authors receive proper credit.

In exceptional circumstances articles may be licensed differently. If you have specific condition (such as one linked to funding) that does not allow this license, please mention this to the editorial office of the journal at submission. Exceptions will be granted at the discretion of the publisher.

### Reproducing Published Material from other Publishers

It is absolutely essential that authors obtain permission to reproduce any published material (figures, schemes, tables or any extract of a text) which does not fall into the public domain, or for which they do not hold the copyright. Permission should be requested by the authors from the copyrightholder (usually the Publisher, please refer to the imprint of the individual publications to identify the copyrightholder).

Permission **is required** for:

1. Your own works published by other Publishers and for which you did not retain copyright.
2. Substantial extracts from anyones' works or a series of works.
3. Use of Tables, Graphs, Charts, Schemes and Artworks if they are unaltered or slightly modified.
4. Photographs for which you do not hold copyright.

Permission **is not required** for:

1. Reconstruction of your *own* table with data already published elsewhere. Please notice that in this case you must cite the source of the data in the form of either "Data from..." or "Adapted from...".
2. Reasonably short quotes are considered *fair use* and therefore do not require permission.
3. Graphs, Charts, Schemes and Artworks that are completely redrawn by the authors and significantly changed beyond recognition do not require permission.

### Obtaining Permission

In order to avoid unnecessary delays in the publication process, you should start obtaining permissions as early as possible. If in any doubt about the copyright, apply for permission. MDPI cannot publish material from other publications without permission.

The copyright holder may give you instructions on the form of acknowledgement to be followed; otherwise follow the style: "Reproduced with permission from [author], [book/journal title]; published by [publisher], [year].' at the end of the caption of the Table, Figure or Scheme.

# APPENDIX (Continued)

Chapter 4, Robotica (Publisher: Cambridge):



# APPENDIX (Continued)

Chapter 5, American Control Conference (Publisher: IEEE):



Copyright Clearance Center RightsLink®

Home | Help | Email Support | Ali Zamani

IEEE Requesting permission to reuse content from an IEEE publication

**Switching between Limit Cycles in a Model of Running Using Exponentially Stabilizing Discrete Control Lyapunov Function**

**Conference Proceedings:** 2018 Annual American Control Conference (ACC)
**Author:** Pranav A. Bhounsule
**Publisher:** IEEE
**Date:** Jun. 2018

*Copyright © 2018, IEEE*

**Thesis / Dissertation Reuse**

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

# APPENDIX (Continued)

Chapter 6, International Conference on Robotics and Automation (Publisher: IEEE):



Copyright Clearance Center RightsLink®

Home | Help | Email Support | Ali Zamani

IEEE Requesting permission to reuse content from an IEEE publication

**Feedback motion planning of legged robots by composing orbital Lyapunov functions using rapidly-exploring random trees**

**Conference Proceedings:** 2019 International Conference on Robotics and Automation (ICRA)
**Author:** Ali Zamani
**Publisher:** IEEE
**Date:** May 2019

*Copyright © 2019, IEEE*

**Thesis / Dissertation Reuse**

**The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:**

*Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:*

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

*Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:*

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

# APPENDIX (Continued)

Chapter 7, International Conference on Intelligent Robots and Systems (Publisher: IEEE):

- **Does IEEE require individuals working on a thesis or dissertation to obtain formal permission for reuse?**

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you must follow the requirements listed below:

**Textual Material**

Using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

**Full-Text Article**

If you are using the entire IEEE copyright owned article, the following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

# VITA

**NAME** Ali Zamani

**EDUCATION**
Ph.D., Mechanical Engineering, University of Illinois at Chicago, Chicago, IL, 2020
M.S, Mechanical Engineering, K. N. Toosi University of Technology, Tehran, Iran, 2011
B.S, Mechanical Engineering, K. N. Toosi University of Technology, Tehran, Iran, 2008

**TEACHING** Teaching Assistant, University of Illinois at Chicago, 2020

**RESEARCH**
Research Assistant, University of Illinois at Chicago, 2019-2020
Research Assistant, University of Texas at San Antonio, 2015-2019

**AWARDS**
Graduate Student Award for Exceptional Research Promise, University of Illinois at Chicago, 2020
Graduate Student Professional Development Awards, University of Texas at San Antonio, 2018
Harnek S. and Malkit Gill Endowed Scholarship, University of Texas at San Antonio, 2017 and 2018
Competitive Valero Research Scholarship, College of Engineering, University of Texas at San Antonio, 2015-2016
Competitive Mechanical Engineering PhD Fellowship, University of Texas at San Antonio, 2015-2016

**PUBLICATIONS**
Zamani, A. and Bhounsule, P. A. "Receding Horizon Control for a 2D Point-Mass Hopping Model Navigating on Terrain With Stepping Stones and Stairs." accepted to In Proceeding of the ASME Dynamic Systems and Control Conference (DSCC), 2020.

Zamani, A. and Bhounsule, P. A. "Nonlinear model predictive control of hopping model using approximate step-to-step models for navigation on complex terrain." accepted to In Proceeding of International Conference on Intelligent Robots and Systems (IROS), 2020.

Bhounsule, P. A., Zamani, A., Krause, J., Farra, S., and Pusey, J. "Control Policies for a Large Region of Attraction for Dynamically Balancing Legged Robots: A Sampling-Based Approach." Robotica, 2020.

Zamani, A., Galloway, J. D., and Bhounsule, P. A. "Feedback motion planning of legged robots by composing orbital lyapunov functions using rapidly-exploring random trees." In Proceeding of International Conference on Robotics and Automation (ICRA), 2019.

Zamani, A. and Bhounsule, P. A. "Control synergies for rapid stabilization and enlarged region of attraction for a model of hopping." Biomimetics, 3(3):25, 2018.

Bhounsule, P. A., Zamani, A., and Pusey, J. "Switching between limit cycles in a model of running using exponentially stabilizing discrete control lyapunov function." In Proceeding of American Control Conference (ACC), 2018.

Zamani, A. and Bhounsule, P. A. "A discrete control lyapunov function for exponential orbital stabilization of the simplest walker." Journal of Mechanisms and Robotics, 9(5):051011, 2017.

Zamani, A. and Bhounsule, P. A. "Foot placement and ankle pushoff control for the orbital stabilization of bipedal robots." In Proceeding of International Conference on Intelligent Robots and Systems (IROS), 2017.

Bhounsule, P. A. and Zamani, A. "Stable bipedal walking with a swing-leg protraction strategy." Journal of Biomechanics, 51:123-127, 2017.

Zamani, A., Bhounsule, P. A., and Taha, A. "Planning Bipedal Locomotion on Patterned Terrain Using Energy-Efficient, Dynamic Primitives." In Proceedings of SPIE, Unmanned Systems Technology XVIII, 2016.

Massah, A. B., Zamani, A., Salehinia, Y., Aliyari, M., and Teshnehlab, M. "A Hybrid Controller Based on CPG and ZMP for Biped Locomotion." Journal of Mechanical Science and Technology, 27(11) 34733486, 2013.

F. Alambeigi, F., Zamani, A., Vossoughi, G., and Zakerzadeh, M. R. "Robust Shape Control of Two SMA Actuators Attached to a Flexible

Beam Based on DK Iteration.“ In Proceeding of the 12th International Conference on Control, Automation, and Systems (ICCAS), 2012.

Moosavian, S. A. A., Khorram, M., Zamani, A., and Abedini, H. “PD Regulated Sliding Mode Control of a Quadruped Robot.“ In Proceeding of the IEEE International Conference on Mechatronics and Automation (ICMA), 2011.

Zamani, A., Khorram, M., and Moosavian, S. A. A. “Dynamics and Stable Gait Planning of a Quadruped Robot“ In Proceeding of the 11th International Conference on Control, Automation, and Systems (ICCAS), 2011.