MODEL BASED METHODS FOR THE CONTROL AND PLANNING OF RUNNING ROBOTS

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Ömür Arslan

July 2009

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Ömer Morgül(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Uluç Saranlı(Co-supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Hitay Özbay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Afşar Saranlı

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

MODEL BASED METHODS FOR THE CONTROL AND PLANNING OF RUNNING ROBOTS

Ömür Arslan
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ömer Morgül
July 2009

The Spring-Loaded Inverted Pendulum (SLIP) model has long been established as an effective and accurate descriptive model for running animals of widely differing sizes and morphologies. Not surprisingly, the ability of such a simple spring-mass model to capture the essence of running motivated several hopping robot designs as well as the use of the SLIP model as a control target for more complex legged robot morphologies. Further research on the SLIP model led to the discovery of several analytic approximations to its normally nonintegrable dynamics. However, these approximations mostly focus on steady-state running with symmetric trajectories due to their linearization of gravitational effects, an assumption that is quickly violated for locomotion on more complex terrain wherein transient, non-symmetric trajectories dominate. In the first part of the thesis , we introduce a novel gravity correction scheme that extends on one of the more recent analytic approximations to the SLIP dynamics and achieves good accuracy even for highly non-symmetric trajectories. Our approach is based on incorporating the total effect of gravity on the angular momentum throughout a single stance phase and allows us to preserve the analytic simplicity of the approximation to support research on reactive footstep planning for dynamic

legged locomotion. We compare the performance of our method with two other existing analytic approximations by simulation and show that it outperforms them for most physically realistic non-symmetric SLIP trajectories while maintaining the same accuracy for symmetric trajectories. Additionally, this part of the thesis continues with analytical approximations for tunable stiffness control of the SLIP model and their motion prediction performance analysis. Similarly, we show performance improvement for the variable stiffness approximation with gravity correction method. Besides this, we illustrate a possible usage of approximate stance maps for the controlling of the SLIP model.

Furthermore, the main driving force behind research on legged robots has always been their potential for high performance locomotion on rough terrain and the outdoors. Nevertheless, most existing control algorithms for such robots either make rigid assumptions about their environments (e.g flat ground), or rely on kinematic planning with very low speeds. Moreover, the traditional separation of planning from control often has negative impact on the robustness of the system against model uncertainty and environment noise. In the second part of the thesis, we introduce a new method for dynamic, fully reactive footstep planning for a simplified planar spring-mass hopper, a frequently used dynamic model for running behaviors. Our approach is based on a careful characterization of the model dynamics and an associated deadbeat controller, used within a sequential composition framework. This yields a purely reactive controller with a very large, nearly global domain of attraction that requires no explicit replanning during execution. Finally, we use a simplified hopper in simulation to illustrate the performance of the planner under different rough terrain scenarios and show that it is robust to both model uncertainty and measurement noise.

Keywords: Footstep Planning, Legged Locomotion, Hybrid System, Reactive Control, Approximate Stance Map, Spring-Mass Hopper, Rough Terrain, Non-symmetric Steps

ÖZET

KOŞAN ROBOTLARIN KONTROL VE PLANLAMASI İÇİN MODEL TABANLI YÖNTEMLER

Ömür Arslan
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Ömer Morgül
Temmuz 2009

Yüklü Yay Ters Sarkaç (YYTS) modeli çok değişik boyut ve morfolojideki koşan hayvanların etkili ve güvenilir tanımlayıcı modellenmesinde uzun süredir kullanıldığı gibi değişik zıplayan robotların tasarımı için de bir taban oluşturmaktadır. Bu model üzerindeki ilerleyen araştırmalar integrali alınamayan dinamiği için değişik analitik yaklaşıklamalar ile sonuçlanmıştır. Yerçekimi etkisinin doğrusallaştırılmasından dolayı bu yaklaşıklamalar çoğunlukla kararlı durumdaki simetrik koşmalara yoğunlaşmıştır, ancak bu varsayım geçici rejim ve simetrik olmayan gezingelerin sıklıkla kullanıldığı karmaşık yüzeylerde bozulmaktadır. Bu tezin ilk kısmında, yeni bir yerçekimi etkisini düzeltme yöntemi tanıtılmış ve varolan analitik YYTS dinamiği yaklaşıklamasının güvenilirliği oldukça fazla simetrik olmayan gezingeler için arttırılmıştır. Yaklaşımımız bir adım sırasında yerçekiminin açısal moment üzerine olan toplam etkisinin yaklaşıklamaya dahil edilmesine dayanıyor ve böylelikle de bu kolay analitik yaklaşıklama reaktif adım planlanmasında da kullanılabilir. Yöntemimizin performansı litaratürde varolan iki analitik yaklaşıklama ile karşılaştırılmış, çok sık kullanılan simetrik olmayan gezingeler için diğerlerinden daha iyi olduğu ve simetrik gezingelerde ise aynı performansa

sahip olduğu gözlenmiştir. Buna ek olarak, ayarlanabilir bükülmezlik için analitik yaklaşıklamalar ve bu yaklaşıklamaların hareket kestirme performansı analizi incelenmiştir. Benzer bir şekilde, değişken bükülmezlik yaklaşıklamasının yerçekimi etkisini düzeltme yöntemi ile birlikte kullanıldığında belirgin bir performans gelişimi gözlenmiştir. Bunun yanısıra, YYTS modelinin kontrollünde bu yaklaşmaların olası kullanımını gösterdik.

Bacaklı robotlardaki araştırmaların arkasındaki asıl itici güç bu robotların karmaşık yüzeylerde ve dış ortamlarda yüksek devinim potensiyellerinin olmasıdır. Buna rağmen, bu robotlar için varolan kontrol algoritmalarının çoğu ya ortam ile ilgili katı varsayımlara (düz zemin gibi) yada çok düşük hızlarda kinematik planlamaya dayanmaktadır. Planlamanın geleneksel olarak kontroldan ayrılması genellikle model belirsizliğine ve ortam gürültüsüne karşı sistemin dayanıklılığının azalmasına neden olmaktadır. Bu tezin ikinci kısmında, koşma davranışları için sıklıkla kullanılan dinamik bir model olan basitleştirilmiş düzlemsel yay-kütle sıçrayanı için dinamik ve tamamen reaktif adım planlaması için yeni bir method önerilmektedir. Yaklaşımımız model dinamiğinin dikkatli bir şekilde karakterize edilmesine ve sıralı ardışık bileşim taslağı içerisinde kullanılacak ilgili deadbeat kontrolüne dayanmaktadır. Böylelikle çok geniş, hemen hemen evrensel çekim yöresi olan ve uygulama sırasında tekrar planlama ihtiyacı duymayan tamamen reaktif denetleyici sağlanmış olmaktadır. Son olarak, planlayıcının performansı simülasyon ortamında başitleştirilmiş bir sıçrayıcının üzerinde değişik yüzey koşullarında gösterilmiş ve yöntemin model belirsizliklerine ve ölçme gürültülerine karşı dayanıklı olduğunu gözlemlenmiştir.

Anahtar Kelimeler: Adım Planlaması, Bacaklı Lokomosyon, Karma Sistem, Reaktif Kontrol,Yüklü Yay Ters Sarkaç (YYTS), Analitik YYTS Dinamiği Yaklaşıklaması, Karmaşık Arazi, Simetrik Olmayan Adımlar

ACKNOWLEDGMENTS

First, I would like to thank my supervisors, Ömer Morgül and Uluç Saranlı, for their guidance and support throughout my study. This work is an achievement of their invaluable advice. They were always there to listen and to give advice. I am very grateful to my advisor for their patience during our everlasting exciting research meetings and stimulating discussions. Also, I really appreciate all the help they've given me to control my interest and deeply study a problem.

I took my initial steps into the area of dynamical legged locomotion with our brilliant and helpful discussions during SensoRhex Project. I am thankful to all the members of SensoRhex Project. This is a great opportunity to express my respect to Afşar Saranlı, Uluç Saranlı, Yiğit Yazıcıoğlu and Kemal Leblebici oğlu . Especially, I would like to thank Uluç Saranlı for inspiring me to legged robotics with his endless energy and enthusiasm for legged systems.

Additionally, there are a number of people from my research group, Bilkent Dexterous Robotics and Locomotion (BDRL), who help me along the way. I am very thankful to Akın Avcı (patron - my boss), Tuğba Yıldız, Sıtar Kortik and Cihan Öztürk for our wonderful late night studies and discussions. Further thanks should also go to the member of Robotics Laboratory (RoLab) at Middle East Technical University (METU), Mustafa Mert Ankaralı (my great colleague), Emre Ege and Gökhan Gültekin.

I also extend my thanks to my friends from my department, Onur Tan, Vahdettin Taş, Ahmet Sezgin and Fazlı Kaybal, for their invaluable discussions on science, technology, politics and sports.

I am also appreciative of the financial support from Bilkent University, Department of Electrical and Electronics Engineering and TÜBİTAK, the Scientific and Technical Research Council of Turkey.

Finally, but forever I would like to thank my parents, Ali and Arife Arslan, my brother, Onur Arslan, and my little sister, Arzu Arslan, for their love, support and encouragement.

Contents

List of Figures

List of Tables

Dedicated to my family ♡

Chapter 1

INTRODUCTION

1.1 Components of a Robotic System

There are many different technical and literal descriptions for the term *robotics* in the literature. From our perspective, robotics is an interdisciplinary field of science and technology related to biology and material science, in addition to electrical, mechanical, and software engineering. There are several categorizations of components in a robotics system, but for us, the main components of a robot are *sensors*, a *controller* and *actuators* as in Fig. 1.1.

Figure 1.1: Three main components of a robotic platform: Sensors, Controller, Actuators.

Nowadays, electronic and manufacturing technologies enable us to easily design different robotics platforms with the same sensor and controller capabilities. On the other hand, most distinguishing features of a robot result from its actuation mechanisms. Therefore, we can classify mobile robotic systems in two main groups according to their actuators: wheeled and legged systems. Both of these

actuation systems have advantages according to their purpose and environment, still appealing to researchers with a large number of unanswered questions. We will discuss in the next section, relative advantages and disadvantages of these alternatives.

1.2 Legged or Wheeled Systems

One of the first questions to be answered before a new mobile robotic platform can be designed is what its actuation mechanism should look like; *Is it going to be wheeled or legged?*. The answer to this question significantly effects the capabilities of the robot, limitations on its operation environment and the difficulty of its control.

Before the usage of legs were considered, researchers were using structured environments for their robots such that the desired robot motion could be easily obtained by wheeled mobile robots. However, increasing demand for robots to operate in our daily life began to show the mobility limitations of wheeled systems due to the fact that they have restricted motion capabilities over unstructured and rough surfaces [4]. In [5], the disadvantages of wheels are summarized in three categories: efficiency of wheels is restricted to flat surfaces, they have limited motion capabilities in the presence of vertical obstacles and they have problems in turning within environments with disorganized obstacles.

In contrast to wheeled systems, legged systems started to receive attention somewhat later in mid-1900s. Legged morphologies have since then been considered necessary to achieve dynamic, robust and autonomous traversal of complex, outdoor terrain. Despite effective behaviors and performance demonstrated by tracked vehicles [6] and flexible multi-wheeled platforms [7], the pallet of behaviors realizable with such morphologies inevitably remains limited due to restricted directions in which forces can be applied to the robot body. On the other hand,

while legged designs do not suffer from such limitations [8], their robust and maneuverable control on complex terrain is still a largely unsolved problem. Additionally, in [9], Raibert summarizes advantages of legged systems compared to wheeled ones with two reasons. One reason is that the mobility of legged systems is considerably better than wheeled vehicles. The legged robots can be used in difficult terrains inaccessible to wheeled systems. The second one is that application areas for wheeled vehicles are mostly limited to structured arenas (e.g roads and rails) and some limited natural settings. On the other hand, legged robots can hypothetically reach all regions that animals can travel on foot.

Another useful feature of legs is that they can also be used as manipulators. By using legs, it may be possible to hold, lift, move things in the environment. Similarly, legs can also be used as sensors to sense weight, position, moveability and structure of objects in the environment. For example, [10] illustrates an excellent use of legs as feelers, in which haptic information from a robot's leg is used to characterize several object properties ,such as weight and moveability, by an operator. Such information can not be obtained by visual sensors.

There are a lot of outstanding and challenging examples of locomotion that can be achieved by using legs, but troublesome or sometimes impossible for wheels. For instance, many research results on robots which can climb tree like structures or vertical wall appeared in the literature. An excellent example is the RISE, which is a bioinspired hexapedal climbing robot capable of locomotion on both level ground and different vertical structures such as building surfaces and trees [11, 12]. Furthermore, one of the challenging environments for robots are sandy terrains wherein wheeled robots usually get stuck. However, SandBot, a bioinspired hexapedal robot, can impressively traverse over sand [13].

Therefore, *legs are intuitively better than wheels* since they have a much wider range of different applications. Despite these advantages of legs over wheels, major difficulties with legged systems are the control problem, gait generation and locomotion, which turn out to be more difficult compared to wheeled platforms.

1.3 Motivation and Existing Work

As mentioned above, control of legged systems is among the most important challenges in their effective, real-life adoption. There are numerous legged morphologies with different sizes and it is impossible to find a general mathematical model describing all such systems. Consequently, the main focus of this thesis is limited to the control and planning of monopod and biped robots. We will use the spring-mass hopper (also known as the Spring Loaded Inverted Pendulum, SLIP), which is frequently used as a fundamental model to analyze and estimate human, animal and robotics locomotion.

Several biological observations and experimental verifications show how well the spring-mass hopper describes fundamental characteristics of natural runners with widely varying sizes and morphologies in nature [14–18]. In parallel, a succession of one-legged hopping robots with SLIP-like morphologies such as Raibert's hoppers [9], ARL Monopods [19], the Bow-Leg design [1], the SLIP hopper [20] and the BiMASC leg [21] demonstrated that dynamic locomotion is not only feasible but also has significant energetic and behavioral advantages. These developments led to an increasing belief that the SLIP model may be more than just a descriptive model that fits biological data, but also a control target whose dynamics are an effective and appropriate abstraction for running behaviors [16]. Evidence to this end was provided by Raibert's robots as well as work on active embedding of SLIP dynamics within more complex morphologies [14, 22].

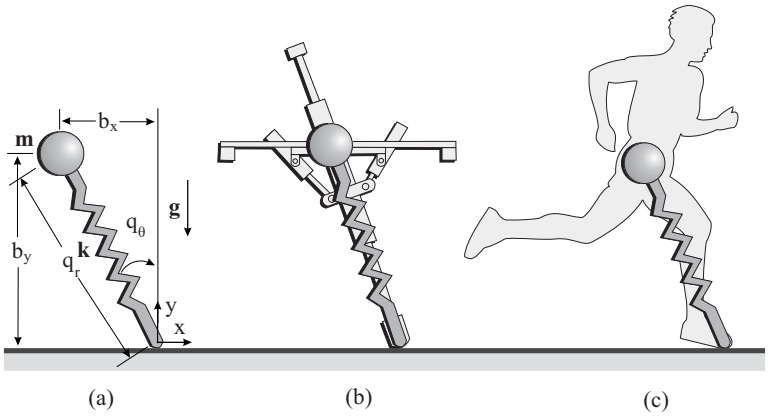

Figure 1.2: (a) The SLIP model, (b) Raibert's hopper, (c) A human runner.

This point of view led to the development of control strategies that explicitly operate on the SLIP model itself. Some of these are based on intuitive observations [9], while others seek accurate and preferably analytic approximations to the nonlinear model dynamics [2, 23]. Such approximations turn out to provide critical tools in the stability analysis of locomotion as well as the design of high performance control and planning algorithms for robot runners. Moreover, these approximations are generally more applicable than numerical alternatives such as the interpolation based alternatives presented in [24].

Since the stance dynamics of SLIP under the effect of gravity are nonintegrable [25], several approximate alternatives have been proposed in the literature. Most notably, Schwind and Koditschek used a generalization of the mean-value theorem to obtain an iterative yet analytic approximation to the stance dynamics [3]. Under certain assumptions, the performance of their approximations is shown to increase with each iteration, eventually converging to true SLIP trajectories. Another alternative is presented by Geyer et al. in [2], wherein a much simpler analytic approximation to the spring mass hopper dynamics is derived based on various assumptions specific to the SLIP model. Both methods focus on steps that are symmetric around the vertical axis, where the effect of gravitational acceleration can be either neglected or linearized and has only a minor impact on accuracy.

In reality, however, humans, animals and robots inevitably need to locomote on a variety of irregular terrain (e.g. grass, gravel, rock field, etc.), for which steady-state is never reached and transient, non-symmetric motions dominate. Under these conditions, assumptions based on either linearization of gravity or conservation of angular momentum are no longer applicable, making most of the currently available approximations inaccurate. Fig. 1.3 illustrates the effect of gravity on the angular momentum of SLIP under different, non-symmetric trajectory conditions. Therefore, this shows the necessity of a more accurate descriptive approximation for nonsymmetric steps and also motivates the first part of this thesis, focusing on analitical models for and control of nonsymmetric locomotion.

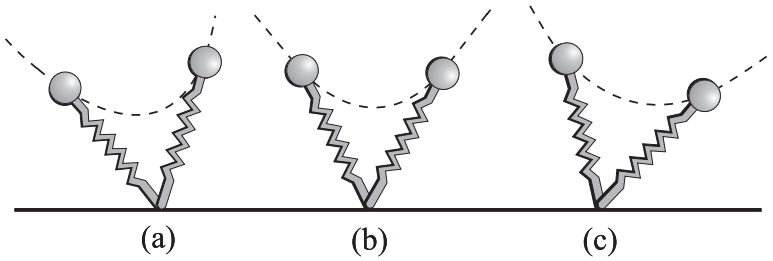

Figure 1.3: The total gravity effect on the angular momentum at the end of the stance phase compared to touchdown instant: (a) decreasing effect on magnitude, (b) the angular momentum is the same since it is a symmetric gait, (c) increasing effect on magnitude.

Ortogonally, motion planning for locomotion on rough terrain has been a topic of interest since the first days of legged robots. With controllers that regulate step-lengths, Raibert's bipeds [9] have been able to traverse both flat terrain with "holes" as well as terrain with significant height variations. Their method relies on preprocessing of the terrain structure to identify specific footholds in the planning step and uses the execution controller to achieve the constructed plan, resulting in an overall controller which is sensitive to modelling uncertainties [26]. A similar planning framework was also investigated in [1], particularly as it applies to the Bow-Leg platform but the proposed solutions still remains non-reactive with explicit replanning performed upon detection of plan failure. More

recently, footstep planning for bipeds in complex environments received considerable attention with the availability of quasi-static but well actuated humanoid robots. As a result of their quasi-static nature, footstep planning for such platforms can rely on a kinematic characterization of their stepping patterns [27]. Since movements of such robots are usually rather slow, discrete abstractions of action sequences combined with search algorithms, possibly with replanning for dynamic or unpredictable environments, suffice to achieve reasonable performance [28, 29]. Unfortunately, for systems that must rely on their second order dynamics, either due to underactuation or to achieve high speed operation, such kinematic methods quickly become inapplicable.

The presence of non-negligible second-order dynamics inevitably brings the need for reactivity since models for such systems are usually much less accurate. This necessity motivates the second part of this thesis in which we concentrate on reactive footstep planning over rough surfaces such as the one illustrated in Fig. 1.4.

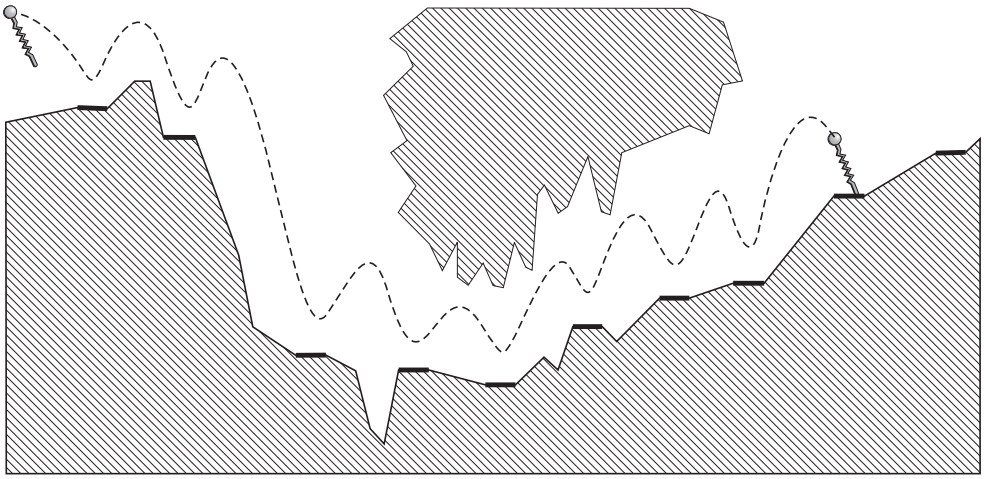

Figure 1.4: An illustrative figure of the spring mass hopper locomotion over an uneven terrain (inspired from [1]).

1.4 Methodology

As discussed in the previous section, the assumed conservation of angular momentum is quickly violated for locomotion on more complex terrain wherein transient, non-symmetric trajectories dominate. In the first part of this thesis, we introduce a novel gravity correction scheme that extends on one of the more recent analytic approximations to the SLIP dynamics and achieves good accuracy even for highly non-symmetric trajectories. Our approach is based on incorporating the total effect of gravity on the angular momentum throughout a single stance phase and allows us to preserve the analytic simplicity of the approximation to support our research on reactive footstep planning for dynamic legged locomotion.

Additionally, we mentioned above, the main difficulty of legged systems: the control problem. In the thesis, we will also introduce a reactive footstep planing algorithm for model based control of monopedal systems over uneven surfaces. Traditional approaches which perform planning and control separately do not perform well in the presence of model uncertainty and measurement noise. On the other hand, existing reactive control methods often make rigid assumptions about their environment (e.g. flat ground or single obstacle of known size) and do not offer the scalability which is necessary for deployment on real-life problems. In the second part of the thesis, we propose a novel algorithm to address these issues for the specific problem of purely reactive control and footstep planning for a simplified planar spring-mass hopper running on rough terrain as illustrated in Fig. 1.5.

One of the most successful methods in integrating deliberate planning with reactivity for dynamically dexterous robots is the Sequential Composition, first introduced in the context of juggling [30] and later applied to other robots such as

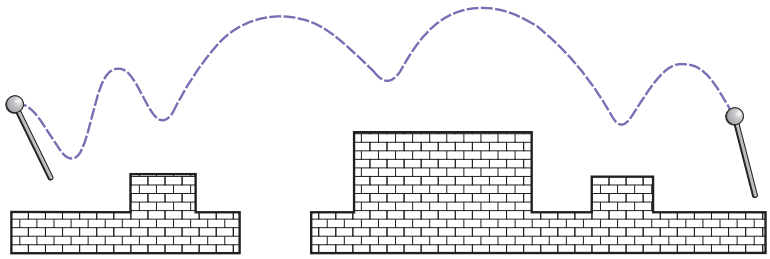

Figure 1.5: A spring-mass hopper running over rough terrain.

planar mobile robots with different actuation modalities [31–33] and the Minifactory [34]. As such, sequential composition characterizes available dynamic behavioral controllers for a given robotic system through their invariant domains and goal sets in the state space, ensuring proper activation order through a prioritization combined with reactive decision-making. Our approach is largely based on these ideas but deviates in our formulation of behavioral primitives and associated domain and goal sets. Among primary contributions of this thesis are the formulation of a general framework for discrete, per-step application of sequential composition to a loosely constrained family of hoppers, as well as the application of resulting ideas to a specific, simplified hopper model and the spring mass hopper supported by an analytical characterization of its apex states reachable from specific regions of allowable footholds.

1.5 Organization of Thesis

In the first part of the thesis, we start in Chapter 2 with the SLIP model and an overview of existing approximations for its stance map. In Section 3.2, we propose a novel gravity correction scheme to increase the accuracy of method in [2] for non-symmetric SLIP trajectories and compare our results with approximations presented in both [2] and [3]. Subsequently, in Section 3.3, we then derive three analytical models for two-phase variable stiffness control by using results of [2], [3], and [35] and compare the prediction performance of these approximations.

The second part of this thesis begins in Chapter 4 with a summary of previous studies on the control and planning of legged locomotion over uneven terrains. We then continue with the design of a position aware deadbeat controller using approximate stance maps in Chapter 3 . In Chapter 6, we introduce a general planning framework for different types of hoppers and continue with our proposed method for reactive footstep planning on a simplified hopper. Finally, in Chapter 7, we conclude the thesis with a review of our work and a summary of open research topics.

In this thesis, we claim that the usage of accurate analytical models of running robots is an efficient way to design robust and reliable gait planning and control algorithms. In summary, the main contribution of this thesis can be listed as follows

- We introduce a novel gravity correction method that extends one of the more recent analytic approximations to the SLIP dynamics and achieves good accuracy even for highly non-symmetric trajectories [35].
- We also derive approximate stance maps for two-phase variable stiffness control based on analytical models in [2], [3] and [35], and show significant prediction accuracy for a good range of SLIP steps.
- We design position aware deadbeat controllers based on approximate models of the SLIP stance trajectory to enable collision free planning and control of the spring-mass hopper.
- We propose a new method for dynamic, fully reactive footstep planning for a general hopper model. We use a simplified hopper in simulation to illustrate the performance of the planner under different rough terrain scenarios and show that it is robust to both model uncertainty and measurement noise. [36].

Chapter 2

BACKGROUND: THE SPRING-MASS HOPPER

This chapter introduces necessary background for the spring-mass hopper as well as a summary of existing methods for the derivation of approximate analytical maps for its stance trajectory.

2.1 The SLIP Model

From the perspective of both biomechanics and robotics, the Spring-Loaded Inverted Pendulum (SLIP) model is the simplest, most effective and accurate descriptive model to analyze dynamic locomotion in humans, animals and robots. This section continues with the SLIP template, its dynamics, and possible modes for its control.

2.1.1 The SLIP Template

The SLIP model consists of a point mass representing the total mass for the system of interest and a massless springy leg, characterizing one or more compliant leg(s) as illustrated in Fig. 2.1(a). SLIP is a hybrid system such that its continuous dynamics change depending on the state of ground contact. During locomotion, the system alternates between stance and flight phases. During stance, the toe remains stationary on the ground with no torque applied whereas in flight, the body follows a ballistic trajectory. Moreover, each of these two phases is also divided into two subphases based on the sign of the rate of change of the leg length for stance phase and sign of the vertical velocity for the flight phase. Fig. 2.1 shows a single stride starting from an apex position and labels relevant phases, subphases and transition events. Furthermore, Table 2.1 details the notation associated with the formal SLIP model we use throughout the thesis. Let us give definitions and general properties of locomotion phases, subphases and transition events.

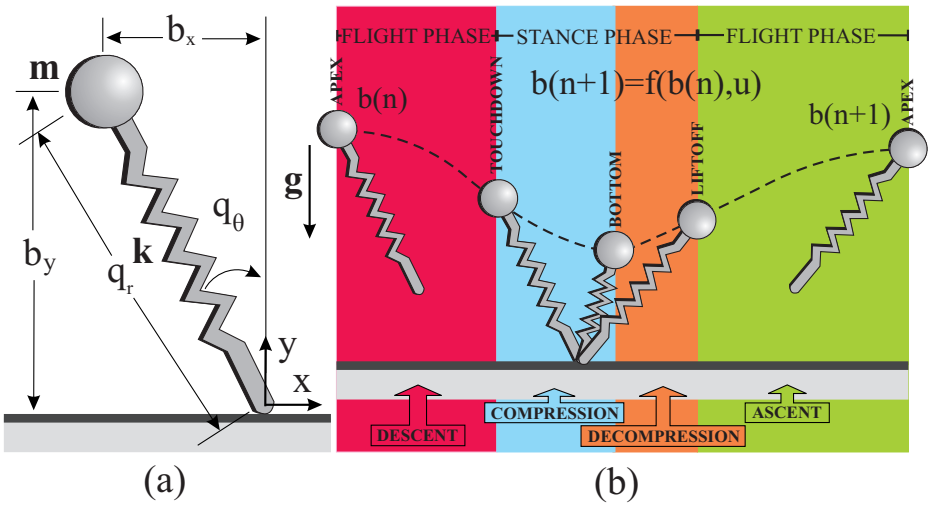

Figure 2.1: The SLIP Model. (a) Coordinates and model parameters. (b) Locomotion phases (shaded regions) and transition events (boundaries).

Table 2.1: Notation associated with the SLIP model used throughout the thesis

SLIP States	
q_r, q_θ	Leg length and leg angle
$q_{\dot{r}}, q_{\dot{\theta}}$	Leg compression and swing rates
$\mathbf{q}$	Body state vector in polar coordinates, $\mathbf{q} = [q_\theta \ q_{\dot{\theta}} \ q_r \ q_{\dot{r}}]^T$
p_r, p_θ	Radial and angular momenta
$q_{r_{td}}, q_{\theta_{td}}, t_{td}$	Touchdown leg length, angle and time
$q_{r_b}, q_{\theta_b}, t_b$	Bottom leg length, angle and time
$q_{r_{lo}}, q_{\theta_{lo}}, t_{lo}$	Liftoff leg length, angle and time
b_x, b_y	Horizontal and vertical body positions
b_{tx}	Horizontal toe position
$b_{\dot{x}}, b_{\dot{y}}$	Horizontal and vertical body velocities
$\mathbf{b}$	Body state vector in cartesian coordinates, $\mathbf{b} = [b_x \ b_{\dot{x}} \ b_y \ b_{\dot{y}} \ b_{tx}]^T$
$b_{y_a}, b_{\dot{x}_a}$	Apex height and velocity
SLIP Parameters	
m, g	Body mass and gravitational acceleration
l_0, k	Leg rest length and leg stiffness
k_c, k_d	Leg stiffness during compression and decompression
E	Total mechanical energy
$F_g(x)$	Ground function. For a given position x, it returns the ground height.
$F_s(q_r, q_{\dot{r}})$	Spring force function. For a given leg length it returns spring force based on the stance phase of SLIP.
$U_s(q_r, q_{\dot{r}})$	Spring potential energy function. For a given leg length it returns stored energy on compliant leg based on the stance phase of SLIP.
Mapping Functions	
$f_{a \mapsto td}(\mathbf{b_a})$	Apex to touchdown map
$f_{td \mapsto lo}(\mathbf{b_{td}})$	Stance map
$f_{lo \mapsto a}(\mathbf{b_{lo}})$	Liftoff to apex map
$t_{c \mapsto p}(\mathbf{b})$	Cartesian to polar coordinate transformation
$t_{p \mapsto c}(\mathbf{q})$	Polar to cartesian coordinate transformation
$\pi_i \circ [x_1 x_2 \cdots x_i \cdots] = x_i$	Projection operator

Flight : This is the period in which the leg does not touch the ground and the body performs ballistic flight, which has a well known dynamics. Depending on the vertical velocity, this phase is divided into two subphases: Ascent and Descent.

Ascent : This is the subperiod of the flight phase where the vertical velocity is positive (upward) and decreasing in magnitude. In this subphase, the gravitational potential energy increases.

Descent : This is the subperiod of the flight phase where the vertical velocity is negative (downward) and increases in magnitude. In this subphase, the gravitational potential energy decreases.

Stance : This is the period in which the model touches the ground. Due to the gravitational pull, stance phase dynamics consist of non-integrable terms. Also, depending on the rate of change of leg length, this phase is divided into two subphases: Compression and Decompression.

Compression : This is the subperiod of the stance phase where the rate of change of leg length is *negative*. In this subphase, the stored energy on the compliant leg increases.

Decompression : This is the subperiod of the stance phase where the rate of change of leg length is *positive*. In this subphase, the stored energy on the compliant leg decreases.

Transition Events : Since our model is a hybrid system, it includes both continuous and discrete dynamics. Transition events are defined as the boundaries between ascent, descent, compression and decompression subphases. To perform a simulation study, these transition events should be checked. Moreover, these transition events have special properties and are important to determine the locomotion characteristics. Now, let us give the general properties of these events.

Apex : This event occurs during the flight phase between the ascent and descent subphases. Coincident with this event, the SLIP body reaches its maximum height (or maximum gravitational potential energy). The zero crossing of the following apex event function identifies this event[1]:

$$f_a(\mathbf{b}) := b_{\dot{y}}, \tag{2.1}$$

[1]Apex occurs if $f_a(\mathbf{b}) = 0$ and SLIP is in flight phase.

Touchdown : This is the flight to stance transition event. In other words, it marks the transition from descent to compression. It occurs when the leg length is equal to the touchdown leg length and the SLIP is going down. This event is identified by the zero crossing of the following touchdown event function[2]:

$$f_{td}(\mathbf{b}) := b_y - (q_{r_{td}} \cos(q_{\theta_{td}}) + F_g(b_{tx})), \tag{2.2}$$

Bottom : This event occurs during the stance phase between the compression and decompression subphases. Coincident with this event, the spring potential energy reaches its maximum value (or the minimum leg length is reached). The zero crossing of the following bottom event function identifies this event[3]:

$$f_b(\mathbf{q}) = q_{\dot{r}}, \tag{2.3}$$

Liftoff : This is the stance to flight transition event. In other words,it marks the transition from decompression to ascent. It occurs when the leg length is equal to the liftoff length and the SLIP is going up. This event is identified by the zero crossing of the following liftoff event function[4]:

$$f_{lo}(\mathbf{b}) := b_y - (q_{r_{lo}} \cos(q_{\theta_{lo}}) + F_g(b_{tx})), \tag{2.4}$$

For every step, the apex return map is defined as a mapping from the current apex state, $b_a[n]$, to the next apex state, $b_a[n+1]$, by using the control input $u[n]$, as illustrated in Fig. 2.2.

[2]Touchdown occurs if $f_{td}(\mathbf{b}) = 0$ and $f_a(\mathbf{b}) < 0$.
[3]Bottom occurs if $f_b(\mathbf{q}) = 0$ and SLIP is in stance phase.
[4]Liftoff occurs if $f_{lo}(\mathbf{b}) = 0$ and $f_a(\mathbf{b}) > 0$.

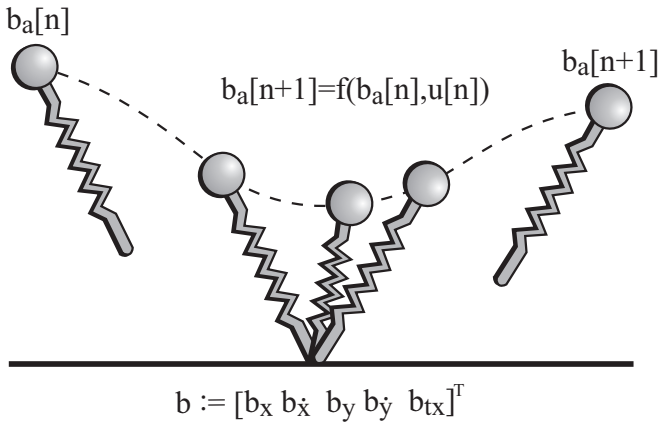

Figure 2.2: The Apex Return Map

2.1.2 SLIP Dynamics

As previously stated, the SLIP model is a hybrid system. Therefore stance and flight phase dynamics must be considered separately. In this section, equations of motion for all SLIP's phases will be provided in both cartesian and polar coordinates to be used for our simulation studies.

Flight Dynamics

During flight, the spring-mass hopper follows a ballistic flight trajectory which has a well known analytical solution. For simplicity, the most suitable coordinates to analyze the flight dynamics are cartesian coordinates. The state vector, $\mathbf{b}$, can be hence defined in cartesian coordinates as

$$\mathbf{b} := \begin{bmatrix} b_x & b_{\dot{x}} & b_y & b_{\dot{y}} & b_{tx} \end{bmatrix}^T,$$

and the flight dynamics are

$$\dot{\mathbf{b}} = \begin{bmatrix} b_{\dot{x}} & 0 & b_{\dot{y}} & -g & b_{\dot{x}} \end{bmatrix}^T.$$

The fifth state variable, b_{tx}, is only defined as a bookkeeping tool for multi-step locomotion. For single stride, locomotion it is not necessary and can be assumed zero. It stays constant during stance and has identical dynamics with

the body position state, b_x, during flight. Since the controller action is assumed to be executed at apex, the toe position b_{tx}, is instantaneously updated at apex with the new touchdown angle independently from its dynamics.

Under these dynamics, the apex to touchdown map, $f_{a \mapsto td}(\mathbf{b_a})$, can easily be derived for a flat surface at zero height as follows,

$$\mathbf{b_{td}} = f_{a \mapsto td}(\mathbf{b_a}) := \begin{bmatrix} b_{x_a} + b_{\dot{x}_a}\sqrt{2(b_{y_a} - b_y)/g} \\ b_{\dot{x}_a} \\ q_{r_{td}}\cos(q_{\theta_{td}}) \\ -\sqrt{2g(b_{y_a} - b_y)} \\ b_x + q_{r_{td}}\sin(q_{\theta_{td}}) \end{bmatrix}. \tag{2.5}$$

In addition, a simple and convenient way to derive the stance dynamics is using polar coordinates. Our state vector hence is defined as

$$\mathbf{q} := \begin{bmatrix} q_\theta & q_{\dot{\theta}} & q_r & q_{\dot{r}} \end{bmatrix}^T.$$

And so, the touchdown body states may be mapped with a transformation, $t_{c \mapsto p}(\mathbf{b_{td}})$, from cartesian to polar coordinates, this map is given by

$$\mathbf{q_{td}} = t_{c \mapsto p}(\mathbf{b_{td}}) := \begin{bmatrix} q_{\theta_{td}} \\ (-b_y b_{\dot{x}} + (b_x - b_{tx})b_{\dot{y}})/q_r^2 \\ q_{r_{td}} \\ ((b_x - b_{tx})b_{\dot{x}} + b_y b_{\dot{y}})/q_r \end{bmatrix}. \tag{2.6}$$

Also, for a given liftoff state in polar coordinates, a transformation, $t_{p \mapsto c}(\mathbf{q_{lo}})$, to cartesian coordinates is given by

$$\mathbf{b_{lo}} = t_{p \mapsto c}(\mathbf{q_{lo}}) := \begin{bmatrix} -q_{r_{lo}}\sin(q_{\theta_{lo}}) \\ -q_{\dot{r}_{lo}}\sin(q_\theta) - q_{r_{lo}}\cos(q_{\theta_{lo}})q_{\dot{\theta}_{lo}} \\ q_{r_{lo}}\cos(q_{\theta_{lo}}) \\ q_{\dot{r}_{lo}}\cos(q_{\theta_{lo}}) - q_{r_{lo}}\sin(q_{\theta_{lo}})q_{\dot{\theta}_{lo}} \\ 0 \end{bmatrix}, \tag{2.7}$$

where, once again, we assume a flat surface with zero height and toe is located at the origin of the global cartesian coordinate frame. Finally, the liftoff to apex map, $f_{lo\mapsto a}(\mathbf{b_{lo}})$, can also be easily derived as

$$\mathbf{b_a} = f_{lo\mapsto a}(\mathbf{b_{lo}}) := \begin{bmatrix} b_{x_{lo}} + b_{\dot{x}_{lo}} b_{\dot{y}_{lo}}/g \\ b_{\dot{x}_{lo}} \\ 0.5 b_{\dot{y}_{lo}}^2/g \\ 0 \\ b_{\dot{x}_{lo}} b_{\dot{y}_{lo}}/g \end{bmatrix}. \tag{2.8}$$

Stance Dynamics

During stance, i.e. when toe of the leg touches the ground, we assume the presence of a frictionless revolute joint at the contact point until the liftoff event occurs. As previously mentioned, polar coordinates are best suited for the derivation of the stance dynamics. As such, the Lagrangian equation during stance in polar coordinates is given by (see Fig. 2.1)

$$L = \frac{m}{2}(\dot{q_r}^2 + {q_r}^2 \dot{q_\theta}^2) - \frac{k}{2}(l_0 - q_r)^2 - mgq_r \cos(q_\theta). \tag{2.9}$$

Equations of motion for the SLIP model can thus be derived as

$$m\ddot{q_r} = mq_r \dot{q_\theta}^2 + k(l_0 - q_r) - mg\cos(q_\theta), \tag{2.10}$$

$$0 = \frac{d}{dt}(m{q_r}^2 \dot{q_\theta}) + mgq_r \sin q_\theta. \tag{2.11}$$

Using (2.10) and (2.11), the stance dynamics in polar coordinates, $\mathbf{q}$, are given by

$$\dot{\mathbf{q}} = \begin{bmatrix} q_{\dot{\theta}} \\ -\frac{g\sin(q_\theta)}{q_r} - \frac{2q_{\dot{r}}q_{\dot{\theta}}}{q_r} \\ q_{\dot{r}} \\ \frac{F_s(q_r, q_{\dot{r}})}{m} + q_r q_{\dot{\theta}}^2 - g\cos(q_\theta) \end{bmatrix},$$

where $F_s(q_r, q_{\dot{r}})$ is the spring force function defined as

$$F_s(q_r, q_{\dot{r}}) = \begin{cases} k_c(l_0 - q_r) & \text{if } q_{\dot{r}} \leq 0 \\ k_d(l_0 - q_r) & \text{if } q_{\dot{r}} > 0 \end{cases} . \tag{2.12}$$

Note that, (2.10) and (2.11) are coupled nonlinear differential equations and a closed form solution with known basic functions can not easily be found. In fact, stance dynamics have several nonintegrable terms due to the presence of gravity [25, 37]. Therefore, there are no exact analytical solutions for this stance map. Nevertheless, there are several studies on approximate solutions of SLIP stance dynamics in literature. We consider two such approximations introduced in [2] and [3] in Section 2.2.

For completeness, we give stance dynamics in cartesian coordinates with

$$\dot{\mathbf{b}} = \begin{bmatrix} \dot{b}_x \\ \dot{b}_{\dot{x}} \\ \dot{b}_y \\ \dot{b}_{\dot{y}} \\ \dot{b}_{tx} \end{bmatrix} = \begin{bmatrix} b_{\dot{x}} \\ \frac{-F_s(q_r, q_{\dot{r}}) \sin(q_\theta)}{m} \\ b_{\dot{y}} \\ \frac{F_s(q_r, q_{\dot{r}}) \cos(q_\theta)}{m} - g_s \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} b_{\dot{x}} \\ \frac{-F_s(\sqrt{(b_x - b_{tx})^2 + (b_y - F_g(b_{tx}))^2}, \frac{(b_x - b_{tx}) b_{\dot{x}} + (b_y - F_g(b_{tx})) b_{\dot{y}}}{\sqrt{(b_x - b_{tx})^2 + (b_y - F_g(b_{tx}))^2}}) \sin(\arctan(\frac{b_y - F_g(b_{tx})}{b_x - b_{tx}}))}{m} \\ b_{\dot{y}} \\ \frac{F_s(\sqrt{(b_x - b_{tx})^2 + (b_y - F_g(b_{tx}))^2}, \frac{(b_x - b_{tx}) b_{\dot{x}} + (b_y - F_g(b_{tx})) b_{\dot{y}}}{\sqrt{(b_x - b_{tx})^2 + (b_y - F_g(b_{tx}))^2}}) \cos(\arctan(\frac{b_y - F_g(b_{tx})}{b_x - b_{tx}}))}{m} - g_s \\ 0 \end{bmatrix} .$$

For simplicity, if we assume a flat ground with zero height (i.e. $F_g(x) = 0$) and that the toe of the leg is located at zero ($b_{tx} = 0$). Then, the stance dynamics in

cartesian coordinates become

$$\dot{\mathbf{b}} = \begin{bmatrix} b_{\dot{x}} \\ \frac{-F_s(\sqrt{{b_x}^2+{b_y}^2}, \frac{b_x b_{\dot{x}} + b_y b_{\dot{y}}}{\sqrt{{b_x}^2+{b_y}^2}}) \sin(\arctan(\frac{b_y}{b_x}))}{m} \\ b_{\dot{y}} \\ \frac{F_s(\sqrt{{b_x}^2+{b_y}^2}, \frac{b_x b_{\dot{x}} + b_y b_{\dot{y}}}{\sqrt{{b_x}^2+{b_y}^2}}) \cos(\arctan(\frac{b_y}{b_x}))}{m} - g_s \\ 0 \end{bmatrix}. \tag{2.13}$$

2.1.3 Possible Modes of Control

Now that we have introduced the system model, we will discuss in this section, possible modes of controlling SLIP locomotion.

Two of the control parameters for the spring-mass hopper system are common to all controllers:

- The touchdown leg angle, q_θ
- The amount of change in the total mechanical energy, ΔE

The first control parameter, q_θ, is an essential and indispensable for most monopod and biped systems. However, there are several ways to control the change in total mechanical energy, ΔE. In the sequel, we classify these possible control modes into three groups based on how total mechanical energy may be physically changed.

Leg Length Control - LLC : This is a mode of control for SLIP-like systems where leg stiffness is fixed during locomotion. Instead, the total mechanical energy is controlled by changing leg lengths at touchdown and liftoff. For example, to increase the total mechanical energy the touchdown leg length is chosen to be smaller than the liftoff leg length. An example physical

platform using the LLC mode of control is the Bow Leg hopping robot, which uses a precompressed leg with stored potential energy during flight to release stored energy during stance [38, 39].

Leg Stiffness Control - LSC : This is another control scheme to modify the total mechanical energy of SLIP-like systems such that the leg stiffness throughout whole stance phase is set to a fixed value and can be modified during flight across subsequent steps. The amount of change in the total mechanical energy is also controllable using only either the touchdown or liftoff leg lengths. The other leg length is kept equal to the rest leg length. For example, adjustments to the touchdown leg length are needed to inject energy while the liftoff leg length is set to be equal to the rest leg length. Similarly, liftoff leg length adjustment is required to take out energy with the touchdown leg length is kept equal to rest leg length.

Two-Phase Stiffness Control - TPSC : This is the last mode of control for adjusting the total mechanical energy such that the touchdown and liftoff leg lengths are kept equal to the rest leg length and energy adjustment is done by separately controlling leg stiffness during compression and decompression subphases. For instance, to increase the total system energy, the compression leg stiffness must be smaller than the decompression leg stiffness. We note that Raibert's hoppers are examples such legged systems and use TPSC to modify the system energy. During locomotion, the hopper detects the bottom instant, then, according to a control strategy, the total system energy is adjusted by controlling the air pressure inside leg actuator to change leg stiffness [9]. Another example is BiMASC, a biped with a mechanically tuneable leg stiffness, that can use this mode of control since it has a tunable leg compliance [21]. One of the practical difficulties of TPSC is the detection of the bottom instance and fast adjustment of stiffness. Moreover, there is another novel leg structure, a half circular leg

with variable compliance, which enables TPSC like mode of control for a RHex like robots or biped as in [40, 41].

2.2 Approximate Stance Maps

Since stance dynamics of the spring-mass hopper are noninegrable [25, 37], approximate analytical models for the stance motion of SLIP are the next best solution. Moreover, approximate stance maps are very useful for the stability analysis of locomotion, the design of control algorithms and motion planning for SLIP and SLIP-like platforms. Before approximate stance maps became available, several controllers were designed based on empirical data captured from running videos of legged runners or animals, or numerical solutions of the SLIP dynamics. For example, in [24], a real-time deadbeat controller was designed by interpolating previously observed gaits. This was an expensive way of designing control laws for monopod runners. Once studies on approximate stance maps became available, several controllers, especially deadbeat controllers, were designed [23, 42, 43] which were applicable to real-time locomotion control and were computationally inexpensive compared to [24].

The general idea behind approximate stance maps is the estimation of the next apex state, $b_a[n+1]$, from the current apex state, $b_a[n]$, with a chosen control set, $u[n]$, (touchdown leg angle and total mechanical energy adjustment - LLC, LSC or TPSC). Also, more detailed maps, i.e. bottom to apex or apex to bottom, may be necessary for other purposes such as support for two-phase variable stiffness. Fig. 2.3 represents these maps for a nonsymmetrical gait.

In the following sections, two previous studies on approximate stance maps, [2] and [3], are examined and summarized with notation consistent with the rest of this thesis.

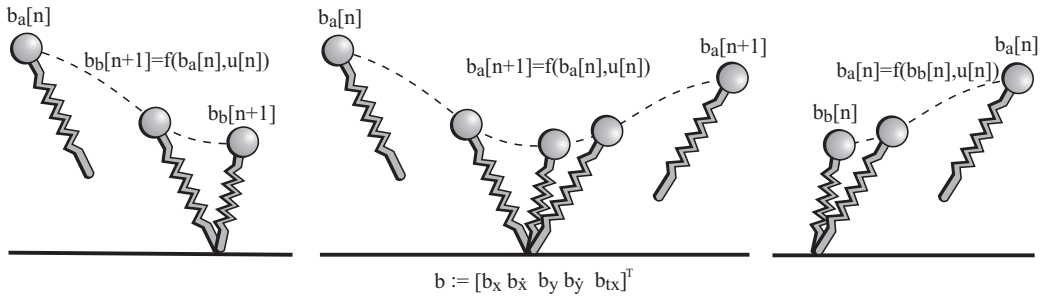

Figure 2.3: *Left*: Apex to Bottom Map. *Middle*: Apex Return Map. *Right*: Bottom to Apex Map.

2.2.1 Simple Approximate Stance Map by Geyer et al.

In this section, we briefly review the approximation method proposed in [2]. Recall from (2.10) and (2.11) that the equations of motion for the stance phase of SLIP in polar coordinates are given by

$$\begin{aligned} m\ddot{q_r} &= mq_r\dot{q_\theta}^2 + k(l_0 - q_r) - mg\cos(q_\theta), \\ 0 &= \frac{d}{dt}(mq_r{}^2\dot{q_\theta}) + mgq_r\sin q_\theta. \end{aligned}$$

Assumption 1. *If a sufficiently small angular span Δq_θ is assumed for the stance phase, the effect of gravity can be linearized by assuming* $\cos(q_\theta) \approx 1$ *and* $\sin(q_\theta) \approx 0$. ■

Under this assumption, the equations of motion simplify to

$$\begin{aligned} m\ddot{q_r} &= mq_r\dot{q_\theta}^2 + k(l_0 - q_r) - mg, \\ \frac{d}{dt}(mq_r{}^2\dot{q_\theta}) &= 0, \end{aligned}$$

which are now integrable since the angular momentum, $p_\theta := mq_r{}^2\dot{q_\theta}$ and the total energy become constants of the motion. Similarly, assuming $\cos(q_\theta) \approx 1$, the total energy can now be written as

$$E := \frac{m}{2}\dot{q_r}^2 + \frac{{p_\theta}^2}{2mq_r{}^2} + \frac{k}{2}(l_0 - q_r)^2 + mgq_r. \tag{2.14}$$

Remark 1. *As a result of Assumption 1, it seems that angular momentum is also conserved during stance phase. However, in reality, angular momentum may change at the end of stance due to gravity. Therefore, the conservation of angular momentum is purely a result of this first assumption.*

Defining the parameters

$$\rho := \frac{q_r - l_0}{l_0} \leq 0,\ \epsilon := \frac{2E}{m{l_0}^2},\ \omega := \frac{p_\theta}{m{l_0}^2} \text{ and } \omega_0 := \sqrt{\frac{k}{m}},$$

and substituting them into (2.14), yields

$$\epsilon = \dot{\rho}^2 + \frac{\omega^2}{(1+\rho)^2} + {\omega_0}^2\rho^2 + \frac{2g}{l_0}(1+\rho). \tag{2.15}$$

Assumption 2. *The above definition of ρ represents the relative spring compression. If we assume that the leg spring is only subjected to small compressions, namely $|\rho| \ll 1$, then the term $1/(1+\rho)^2$ can be approximated by a Taylor series expansion around zero to yield*

$$\left.\frac{1}{(1+\rho)^2}\right|_{\rho=0} = 1 - 2\rho + 3\rho^2 - O(\rho^3). \blacksquare$$

Combining Assumption 2 with the energy equation (2.15) and using further simplifications detailed in [2], an analytical solution to the radial motion $q_r(t)$ can be found as

$$q_r(t) = l_0(1 + a + b\sin(\hat{\omega}_0 t)), \tag{2.16}$$

where we define

$$\begin{aligned}
\hat{\omega}_0 &:= \sqrt{{\omega_0}^2 + 3\omega^2},\\
a &:= (\omega^2 - g/l_0)/\hat{\omega}_0^2,\\
b &:= \sqrt{a^2 + (\epsilon - \omega^2 - 2g/l_0)/\hat{\omega}_0^2}.
\end{aligned}$$

Fig. 2.4 shows one period of this approximate solution for the leg length trajectory during the stance phase. As seen from the figure, it is a sinusoidal

motion with amplitude $l_0 b$, frequency $\hat{\omega}_0$ and offset $l_0(1+a)$. Since it is a solution for only the stance phase, we only care about the part of solution where $q_r(t) \leq l_0$. Also, Δl_{max} represents the maximum spring compression and it is the difference between $l_0 b$ and $l_0 a$ ($\Delta l_{max} = l_0 b - l_0 a$). Additionally, using Assumption 3 and definition of ρ we can conclude that $b - a \ll 1$ ($|\rho| \ll 1$ and $|\rho| = \frac{\Delta l_{max}}{l_0} = b - a$ implying that $b - a \ll 1$).

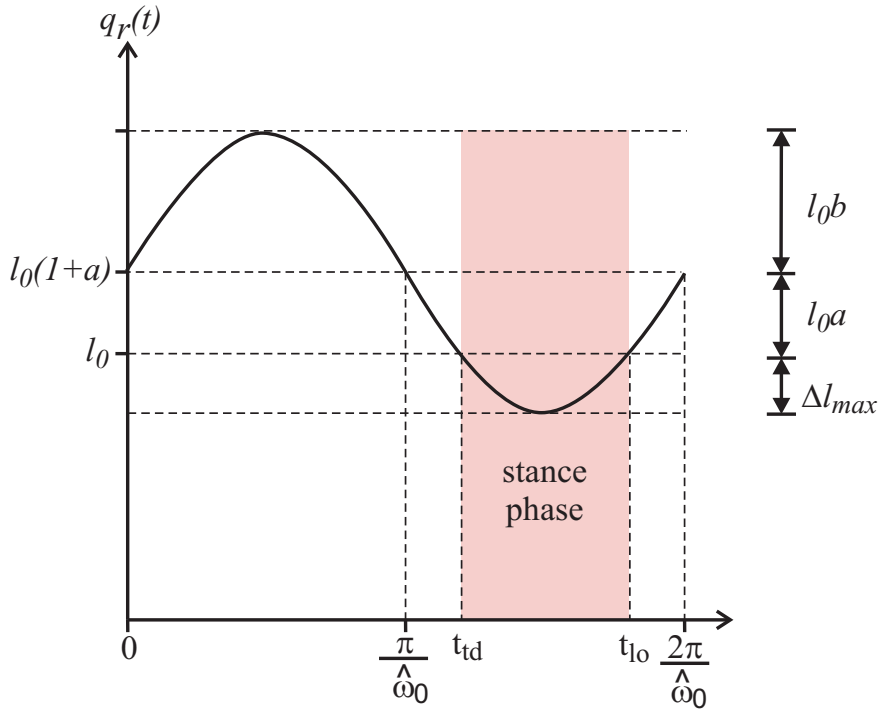

Figure 2.4: General solution for the leg length, $q_r(t)$, during stance. The sinusoidal solution has amplitude $l_0 b$ and frequency $\hat{\omega_0}$ with offset $l_0(1+a)$. Since the solution is only suitable for the stance phase, only the portion where $q_r \leq l_0$ is significant. The parameter a can also be negative, in which case l_0 will be above $l_0(1+a)$. Δl_{max} represents the maximum leg compression. (reproduced from [2])

The equation (2.16) can conveniently be used to determine the times for critical events such as touchdown, bottom and liftoff relative to an unknown time origin, yielding

$$t_{td} = (\pi - \arcsin(-a/b))/\hat{\omega}_0, \tag{2.17}$$

$$t_b = 3\pi/2\hat{\omega}_0, \tag{2.18}$$

$$t_{lo} = (2\pi + \arcsin(-a/b))/\hat{\omega}_0, \tag{2.19}$$

where we assume that the liftoff and touchdown leg lengths, $q_{r_{lo}}$ and $q_{r_{td}}$, are equal to the rest leg length, l_0.

Using this analytical solution for the radial motion, the angular motion of the SLIP can also be derived since angular momentum is assumed to be conserved during the stance phase as a result of Assumption 1. Using ρ and ω, we can write

$$q_{\dot{\theta}} = \frac{\omega}{(1+\rho)^2}. \tag{2.20}$$

Using Assumption 2 once again, (2.20) becomes $q_{\dot{\theta}} = \omega(1-2\rho)$. Recalling that $\rho := (q_r - l_0)/l_0 = a + b\sin(\hat{\omega}_0 t))$, an analytic solution to the angular motion can be found as

$$\begin{aligned} q_\theta(t) &= q_{\theta_{td}} + \int_{t_{td}}^{t} \omega[(1-2a) - 2b\sin(\hat{\omega}_0 t)]dt, \\ q_\theta(t) &= q_{\theta_{td}} + \omega(1-2a)(t-t_{td}) + \frac{2b\omega}{\hat{\omega}_0}[cos(\hat{\omega}_0 t) - cos(\hat{\omega}_0 t_{td})], \end{aligned} \tag{2.21}$$

where $t_{td} \leq t \leq t_{lo}$, and t_{td} and t_{lo} as in (2.17) and (2.19). Moreover, the stance time, t_s, can easily be calculated as

$$t_s = t_{lo} - t_{td} = [\pi + 2\arcsin(-a/b)]/\hat{\omega}_0. \tag{2.22}$$

Moreover, if the touchdown instance is shifted to $t = 0$, the equations for the leg length and the leg angle take the form,

$$\begin{aligned} q_r(t) &= l_0 + l_0[a(1-\cos(\hat{\omega}_0 t)) - \sqrt{b^2-a^2}\sin(\hat{\omega}_0 t)], \\ q_\theta(t) &= q_{\theta td} + (1-2a)\omega t + \frac{2\omega}{\hat{\omega}_0}[a\sin(\hat{\omega}_0 t) + \sqrt{b^2-a^2}(1-cos(\hat{\omega}_0 t))], \end{aligned}$$

while the stance time, t_s, remains the same as in (2.22).

If the predefined parameters a, b, ϵ, ω and ω_0 are replaced by touchdown states and system parameters, and if the touchdown time is selected as zero, the

radial and angular trajectories become

$$\begin{aligned} q_r(t) &= l_0 - \frac{|q_{\dot{r}td}|}{\hat{\omega}_0}\sin(\hat{\omega}_0 t) + \frac{{q_{\dot{\theta}td}}^2 l_0 - g_s}{\hat{\omega}_0^2}(1 - cos(\hat{\omega}_0 t)), \\ q_\theta(t) &= q_{\theta td} + (1 - 2\frac{{q_{\dot{\theta}td}}^2 - g_s/l_0}{\hat{\omega}_0^2})q_{\dot{\theta}_{td}} t \\ &\quad + \frac{2q_{\dot{\theta}_{td}}}{\hat{\omega}_0}[\frac{{q_{\dot{\theta}td}}^2 - g_s/l_0}{\hat{\omega}_0^2}\sin(\hat{\omega}_0 t) + \frac{|q_{\dot{r}_{td}}|}{\hat{\omega}_0 l_0}(1 - cos(\hat{\omega}_0 t))], \end{aligned}$$

where the oscillation frequency is calculated by $\hat{\omega}_0 := \sqrt{k/m + 3{q_{\dot{\theta}_{td}}}^2}$ and the stance time, t_s, and the bottom time, t_b, are given by

$$\begin{aligned} t_s &= \frac{1}{\hat{\omega}_0}\left[\pi + 2\arctan(\frac{g_s - l_0{q_{\dot{\theta}_{td}}}^2}{|q_{\dot{r}_{td}}|\hat{\omega}_0})\right], \\ t_b &= \frac{1}{\hat{\omega}_0}\arctan\left(\frac{|q_{\dot{r}_{td}}|\hat{\omega}_0}{q_{\dot{\theta}_{td}}^2 l_0 - g_s}\right). \end{aligned}$$

2.2.2 Iterative Approximate Stance Map by Schwind et al.

In [3], Schwind and Koditschek proposed an analytic approximation to the non-integrable stance dynamics of SLIP through an iterative application of the mean value theorem for integral operators. They showed that this iterative method converges to the true SLIP dynamics under certain assumptions, which are unfortunately easily violated for non-symmetric trajectories. Furthermore, the presented form of their approximation only gives a map from the bottom to the apex point and does not support the entire apex return map. In this section, we briefly describe and extend their method to enable comparisons with our proposed approximations.

The Hamiltonian function for the stance phase of SLIP is given by

$$H = \frac{1}{2m}\left(p_r^2 + \frac{p_\theta^2}{q_r^2}\right) + \frac{1}{2}k(l_0 - q_r)^2 + mgq_r\cos(q_\theta),$$

Solving the equation $H(q_r, p_r, q_\theta, p_\theta) = E$ for p_r yields

$$p_r = H^\dagger(q_r, q_\theta, p_\theta, E) := \sqrt{2m\left(E - \frac{1}{2}k(l_0 - q_r)^2 - mgq_r\cos(q_\theta)\right) - \frac{p_\theta^2}{q_r^2}}, \quad (2.23)$$

Since

$$\dot{p}_j = -\frac{dH}{dq_j} \ , \ \dot{q}_j = \frac{dH}{dp},$$

the Hamiltonian vector field is given by

$$\mathbf{X_H} = \begin{bmatrix} \dot{q}_r \\ \dot{q}_\theta \\ \dot{p}_r \\ \dot{p}_\theta \end{bmatrix} = \begin{bmatrix} \frac{p_r}{m} \\ \frac{p_\theta}{mq_r^2} \\ \frac{p_\theta^2}{mq_r^3} + k_d(l_0 - q_r) - mg_s\cos(q_\theta) \\ mg_s q_r sin(q_\theta) \end{bmatrix}.$$

Using the Hamiltonian vector field and the conservation of energy[5], the equations of motion can be written as

$$\frac{dt_s}{dq_r}(q_r, q_\theta, p_\theta, E) = \frac{m}{p_r(q_r, q_\theta, p_\theta, E)}, \tag{2.24}$$

$$\frac{dq_\theta}{dq_r}(q_r, q_\theta, p_\theta, E) = \frac{p_\theta}{q_r^2 p_r(q_r, q_\theta, p_\theta, E)}, \tag{2.25}$$

$$\frac{dp_\theta}{dq_r}(q_r, q_\theta, p_\theta, E) = \frac{m^2 g_s q_r sin(q_\theta)}{p_r(q_r, q_\theta, p_\theta, E)}, \tag{2.26}$$

where $p_r(q_r, q_\theta, p_\theta, E)$ is given by (2.23).

Since there is an implicit function for radial momentum, p_r, if the other two states, q_θ and p_θ, are solved then p_r can be calculated by using (2.23). However, (2.24), (2.25) and (2.26) are nonlinear coupled differential equations. The exact analytical solution is unknown but by using the mean value theorem an iterative solution procedure can be derived. In fact, Schwind and Koditschek have shown the following results in [45]:

Theorem 1. *Suppose that the function f is continuous on (a,b] and g is an integrable function on (a,b) with* $g(t) \geq 0 \quad \forall t \in (a,b)$. *Let* $x \in (a,b]$. *If both limits*

$$\lim_{t \to a} \frac{f(t) - K}{(t-a)^r} \qquad \lim_{t \to a} \frac{g(t)}{(t-a)^s}$$

exist and are nonzero for some constant K, some nonzero r, and some $s > -1$ *with* $r + s > -1$, *then*

[5]The ideal SLIP template does not contain any damping. In [44], we also derive an analytical approximation to the stance dynamics of SLIP with damping.

1. there exists a $\xi_x \in (a, x]$ *such that*

$$\int_a^x f(t)g(t)dt = f(\xi_x)\int_a^x g(t)dt \tag{2.27}$$

2. for any such choice of ξ_x

$$\lim_{x\to a}\frac{\xi_x - a}{x - a} = \left(\frac{s+1}{r+s+1}\right)^{\frac{1}{r}} \tag{2.28}$$

Moreover, in [45] a practical observation is presented for the calculation of ξ_x.

Observation 1. *If, motivated by (2.28),* ξ_x *is approximated by*

$$\hat{\xi}_x = a + \left(\frac{s+1}{r+s+1}\right)^{\frac{1}{r}}(x-a) \text{ for } x \text{ near } a, \tag{2.29}$$

and ξ_x *is replaced by* $\hat{\xi}_x$ *in (2.27), an approximation is obtained for the integral as*

$$\int_a^x f(t)g(t)dt \approx f(\hat{\xi}_x)\int_a^x g(t)dt \text{ for } x \text{ near } a. \tag{2.30}$$

Using Theorem 1 and Observation 1 and under reasonable assumptions, ξ_x is found as in [3, Appendix A] for the following integrals.

$$\int_{q_{r_b}}^{q_r}\frac{1}{H^\dagger(\sigma, q_\theta, p_\theta, E)}d\sigma \approx \frac{1}{H^\dagger(\hat{\xi}_x, \hat{q}_\theta(\hat{\xi}_x), \hat{p}_\theta(\hat{\xi}_x), E)}(q_r - q_{r_b}), \tag{2.31}$$

$$\int_{q_{r_b}}^{q_r}\frac{1}{\sigma^2 H^\dagger(\sigma, q_\theta, p_\theta, E)}d\sigma \approx \frac{1}{\hat{\xi}_x^2 H^\dagger(\hat{\xi}_x, \hat{q}_\theta(\hat{\xi}_x), \hat{p}_\theta(\hat{\xi}_x), E)}(q_r - q_{r_b}), \tag{2.32}$$

$$\int_{q_{r_b}}^{q_r}\frac{\sigma}{H^\dagger(\sigma, q_\theta, p_\theta, E)}d\sigma \approx \frac{\hat{\xi}_x}{H^\dagger(\hat{\xi}_x, \hat{q}_\theta(\hat{\xi}_x), \hat{p}_\theta(\hat{\xi}_x), E)}(q_r - q_{r_b}), \tag{2.33}$$

where $\hat{\xi}_r$ is found to be the same for all of the integrals as

$$\hat{\xi}_x = \frac{3}{4}q_{r_b} + \frac{1}{4}q_r. \tag{2.34}$$

In order to derive the full apex return map, we use the iterative bottom to liftoff map given in [3] as

$$\begin{aligned}
\hat{t}_{s_{(n+1)}}(q_r) &= t_b + m(q_r - q_{r_b})/H_n^\dagger, \\
\hat{q}_{\theta_{(n+1)}}(q_r) &= q_{\theta_b} + \hat{p}_{\theta_n}(\hat{\xi}_r)(q_r - q_{r_b})/(\hat{\xi}_r^2 H_n^\dagger), \\
\hat{p}_{\theta_{(n+1)}}(q_r) &= p_{\theta_b} + m^2 g \hat{\xi}_r \sin(\hat{q}_{\theta_n}(\hat{\xi}_r))(q_r - q_{r_b})/H_n^\dagger, \\
\hat{p}_{r_{(n+1)}}(q_r) &= H_{n+1}^\dagger,
\end{aligned}$$

where n is the iteration number, $\hat{\xi}_r := 3q_{r_b}/4 + q_r/4$ and $H_k^\dagger := H^\dagger(\hat{\xi}_r, \hat{q}_{\theta_k}(\hat{\xi}_r), \hat{p}_{\theta_k}(\hat{\xi}_r), E)$. The zeroth (initial) iteration can be any approximate analytical solution for the stance phase as in [3].

Similarly, an approximate touchdown to bottom map can be derived as

$$\begin{aligned}
\hat{t}_{s_{(n+1)}}(q_r) &= t_{td} - m(q_r - q_{r_{td}})/H_n^\dagger, \\
\hat{q}_{\theta_{(n+1)}}(q_r) &= q_{\theta_{td}} - \hat{p}_{\theta_n}(\hat{\xi}_r)(q_r - q_{r_{td}})/(\hat{\xi}_r^2 H_n^\dagger), \\
\hat{p}_{\theta_{(n+1)}}(q_r) &= p_{\theta_{td}} - m^2 g \hat{\xi}_r \sin(\hat{q}_{\theta_n}(\hat{\xi}_r))(q_r - q_{r_{td}})/H_n^\dagger, \\
\hat{p}_{r_{(n+1)}}(q_r) &= -H_{n+1}^\dagger,
\end{aligned}$$

where n is the iteration number, $\hat{\xi}_r := 3q_{r_{td}}/4 + q_r/4$ and $H_k^\dagger := H^\dagger(\hat{\xi}_r, \hat{q}_{\theta_k}(\hat{\xi}_r), \hat{p}_{\theta_k}(\hat{\xi}_r), E)$. As before, the zeroth (initial) iteration can be any approximate analytical solution for the stance phase as in [3].

The overall apex return map is obtained by combining the above approximations under any number of desired iterations for the stance approximations given above with solutions to the flight dynamics. However, the stance components of this formulation have one important unknown: the bottom leg length, q_{r_b}. This parameter is critical since it divides the stance map into two components. Even though an exact solution for the bottom leg length does not exist, several approximate solutions can be used. Under assumptions of symmetry and conservation of angular momentum, one approximate solution is given by the quartic

equation arising from the total energy relation as

$$\frac{k}{2}q_r^4 + (mg - kl_0)q_r^3 + \left(\frac{kl_0^2}{2} - E\right)q_r^2 + \frac{p_\theta^2}{2m} = 0,$$

for which a real analytic solution that is less than or equal to the rest leg length can be found. An alternative approximate solution is given by the approximate stance map of [2] as

$$q_{r_b} = l_0(1 + a - b),$$

where a and b are as defined in Section 2.2.1.

In the following chapter, we will discuss necessity of more accurate analytic approximations for nonsymmetric steps and introduce our proposed gravity effect correction methods. We then continue with approximate stance model for two-phase variable stiffness control.

Chapter 3

MODELLING AND CONTROL OF NONSYMMETRIC SLIP STEPS

3.1 Nonsymmetric Steps During Legged Locomotion

In Section 1.2, we discussed advantages of legged systems over wheeled vehicles. One of the important highlights of legged robots is their increased mobility. Theoretically, legged vehicles can reach all the regions that animals can travel on foot. On the other hand, wheeled systems are mostly used in structured environments and have limited motion capabilities on unstructured and rough surfaces.

Even though control of legged locomotion on structured environments is much simpler than on broken and uneven terrains, it is still difficult compared to

wheeled platforms. Legged systems moving over flat surfaces generally use symmetric steps and existing researches mostly focus on this kind of stable running over even surfaces. However, the main reason why legged robots are appealing to researchers is their increased capability of traversing rough surfaces. Interestingly, locomotion over an uneven terrain predominantly requires nonsymmetric steps to accelerate, decelerate, step up, step down, jump and change the direction of motion. Therefore, the true performance of legged systems may only be achieved with effective and reliable nonsymmetric step control.

3.2 An Approximate Stance Map with Gravity Correction

3.2.1 Analytical Approximation Model

In Section 2.2, we reviewed two leading methods [2, 3] to find analytic approximations to the stance map of SLIP, both of which were based on assumptions of symmetric gaits. In this section, we extend the method presented in [2] with a gravity correction to yield a much larger domain of validity in the presence of non-symmetric trajectories.

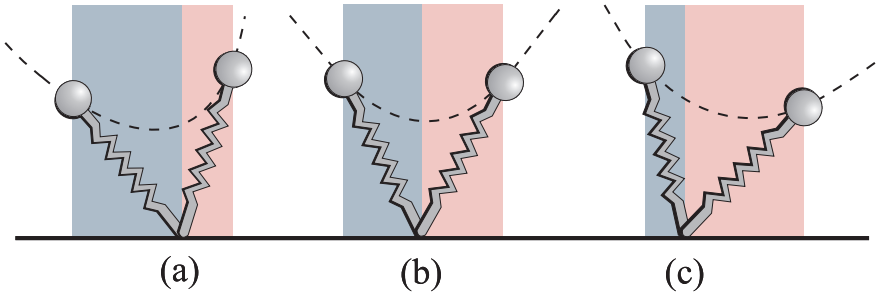

Figure 3.1: The total gravity effect on the angular momentum at the end of the stance phase compared to the touchdown instant (blue and red regions represent decreasing and increasing effects of gravity, respectively). (a) decreasing effect on magnitude, (b) angular momentum stays the same due to the symmetric step, (c) increasing effect on magnitude.

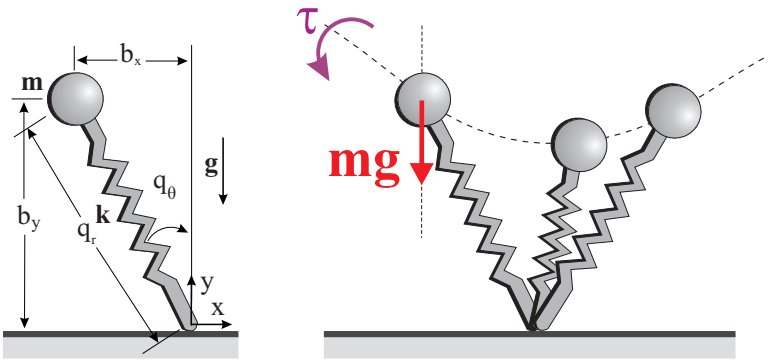

Figure 3.2: The effect of gravity on the angular momentum: τ, gravitational torque on the spring-mass hopper

As illustrated in Fig. 3.1, the angular momentum from touchdown to liftoff is conserved only for symmetric SLIP trajectories. Consequently, while the conservation of angular momentum is a reasonable assumption for symmetric steps, it becomes rather inaccurate for non-symmetric trajectories.

During stance (see Fig. 3.2), the angular momentum around the toe, $P(t)$, can be computed as

$$\begin{aligned} P(t) &= P_{t_0} + \int_{t_0}^{t} \tau(\zeta) d\zeta, \\ \tau(t) &:= mgq_r(t) \sin q_\theta(t), \end{aligned} \tag{3.1}$$

where $\tau(t)$ is the torque applied by the gravitational acceleration around the toe point and P_{t_0} is angular momentum of the SLIP at time t_0. Unfortunately, even with the analytic approximations (2.16) and (2.21), these expressions are too complex for symbolic integration. Consequently, our method involves an n-point approximation to the integral in (3.1), yielding

$$P(t) \approx P_{t_0} + (t - t_0) \left(\frac{1}{n} \sum_{k=1}^{n} mgq_r[n] \sin q_\theta[n] \right). \tag{3.2}$$

Using this approximation, we propose to model the total effect of gravity on the angular momentum from any initial state to any final state during stance with a constant total correction. To this end, we first compute an average leg

length, $q_{r_{av}}(t_i, t_f)$, for the period of interest. We then use this general formulation to compute a correction for the apex return map, using touchdown and liftoff as the initial and final states, respectively.

Using (2.16) and letting t_i and t_f be the initial and final state times such that $t_{td} \leq t_i < t_f \leq t_{lo}$, we have

$$\begin{aligned} q_{r_{av}}(t_i, t_f) &\approx \frac{1}{t_f - t_i} \int_{t_i}^{t_f} l_0(1 + a + b\sin(\hat{\omega}_0 t))dt, \\ &= l_0(1+a) - \frac{b}{\hat{\omega}_0(t_f - t_i)}(\cos(\hat{\omega}_0 t_f) - \cos(\hat{\omega}_0 t_i)), \end{aligned}$$

where a, b and $\hat{\omega}_0$ can be calculated by using related formulas in Section 2.2.1. Using (3.2), the total effect of gravity becomes

$$P_c := \frac{(t_f - t_i) m g_s q_{r_{av}}(t_i, t_f)}{2} \left(\sin q_\theta(t_i) + \sin q_\theta(t_f)\right), \tag{3.3}$$

where $q_\theta(t)$, is as given in Section 2.2.1.

We propose to incorporate P_c into the approximation as a simple correction term added to the original angular momentum, p_θ, otherwise assumed to be constant for the formulations in [2]. This yields a new angular momentum term

$$\hat{p}_\theta = p_\theta + P_c.$$

which replaces p_θ in all derivations. Using touchdown and liftoff times as the initial and final states yields our corrective method for the apex return map.

3.2.2 Performance Analysis

Simulation Environment and Performance Criteria

Our interest in analytic approximations to the stance dynamics of SLIP arises from our need to compute the apex return map for a given set of controls.(i.e. touchdown angle and leg stiffness). Therefore, the most important performance criteria for us is the accuracy of the predicted apex position and velocity. To this end, we use two measures to quantify the prediction performance of both our method and existing methods: Normalized percentage errors in the apex position and liftoff velocity predictions, respectively defined as

$$\begin{aligned} PE_{ap} &= 100\ \frac{||(b_{x_a}, b_{y_a}) - (\hat{b}_{x_a}, \hat{b}_{y_a})||_2}{||(b_{x_a}, b_{y_a})||_2}, \\ PE_{lov} &= 100\ \frac{||(b_{\dot{x}_{lo}}, b_{\dot{y}_{lo}}) - (\hat{b}_{\dot{x}_{lo}}, \hat{b}_{\dot{y}_{lo}})||_2}{||(b_{\dot{x}_{lo}}, b_{\dot{y}_{lo}})||_2}. \end{aligned}$$

We use the liftoff velocity rather than the apex velocity to ensure that normalization is practical even for non-symmetric gaits for which the apex velocity may become zero.

In order to compare different methods of approximation, we ran simulations spanning four different dimensions of initial states and control inputs: the apex height (b_{y_a}), the apex velocity ($b_{\dot{x}_a}$), the spring constant (k) and the relative touchdown angle $q_{\theta_{td_{rel}}} := q_{\theta_{td}} - q_{\theta_{td_n}}$[1]. The ranges considered for these dimensions were determined based on biomechanics literature as well as structural properties of various legged robots. In particular, experiments on humans (with 80kg mass and 1m leg length on average) running at different speeds (in the range 2.5-6.5m/s) reveals that their leg stiffnesses are in the range [10, 50] kN/m [46] . In the robotic domain, the small hexapod robot RHex [22], has an approximate mass of 10kg, leg length of 0.25 m and compliant legs with stiffness of around

[1] $q_{\theta_{td_n}}$ is the neutral touchdown angle, which can be defined as a touchdown angle that results in a symmetric SLIP trajectory and depends on the initial conditions. For each simulation, we numerically calculated this angle to be used as the origin for our plots.

2000N/m for each leg. Based on a dimensionless stiffness measure $\hat{k} = \frac{l_0}{mg}k$, these examples motivate our choice of simulation parameters of Table 3.1 and control input ranges of $k \in$ [125, 1000] N/m and $q_{\theta_{td_{rel}}} \in$ [-0.4, 0.4] rad.

Table 3.1: Simulation Parameters

m (kg)	l_0 (m)	g (m^2/s)	b_{y_a} (m)	$b_{\dot{x}_a}$ (m/s)	k (N/m)	$q_{\theta_{td_{rel}}}$ (rad)
1	1	9.81	1.1 - 1.5	0 - 8	125 - 1000	-0.4 - 0.4

For each of our simulations, we checked whether they satisfy two conditions to ensure that we can support meaningful comparisons of all approximations to the stance map. Firstly, due to the hybrid nature of SLIP locomotion, certain stance trajectories never leave the ground or prevent foot protraction. Consequently, we restricted our domains to exclude simulations where the next apex height is smaller than the rest length of the leg spring, ensuring that there are no limitations on the touchdown angle for the next step. Second, in order to preserve similarity to results presented in both [3] and [2], we restricted the maximum allowable leg compression to a maximum of 25% and excluded trajectories that violate this condition. From among a total of 25500 initial states and control inputs, 10264 were found to satisfy these two conditions.

We computed "ground truth"[2] through numerical integration of SLIP dynamics within MATLAB using a variable time-step, fourth order Runge-Kutta integrator. We then computed approximate estimates of the apex states based on two previously proposed approximations and our new gravity correction scheme and compared estimation performances using the error criteria defined above. For the Schwind approximations, we used the 10th iterate (after which further iterations yield no improvements) to make sure we obtained the best possible performance for their method.

[2]The *ground truth* represents the actual SLIP motion during stance.

Simulation Results

Our general simulation results are illustrated in Fig. 3.3, where we show the mean, standard deviation and maximum values for the apex position and liftoff velocity percentage errors, PE_{ap} and PE_{lov}, across all valid simulations and all three approximation methods. Our results show that there is a notable performance improvement on the average for our proposed gravity correction method compared both to Geyer's and Schwind's approximations.

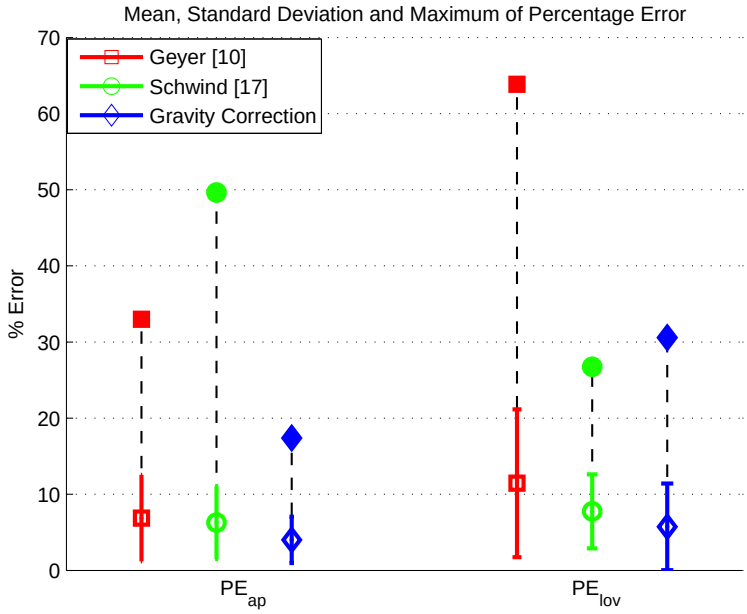

Figure 3.3: Approximation performances for the stance maps of *Geyer* [2], *Schwind* [3] and our proposed *Gravity Correction* method. PE_{ap} (left) and PE_{lov} (right) are apex position and liftoff velocity percentage errors. Empty markers, filled markers and colored vertical bars represent mean, maximum and standard deviations of associated approximations.

A more informative comparison between different approximation alternatives can be achieved by investigating the estimation performance as a function of the relative touchdown angle. Since our method is expected to perform well for non-symmetric trajectories, its error performance should be better than the alternatives for nonzero relative angle values. As illustrated by Figures 3.4 and 3.5, this was indeed the case both error measures, PE_{ap} and PE_{lov}.

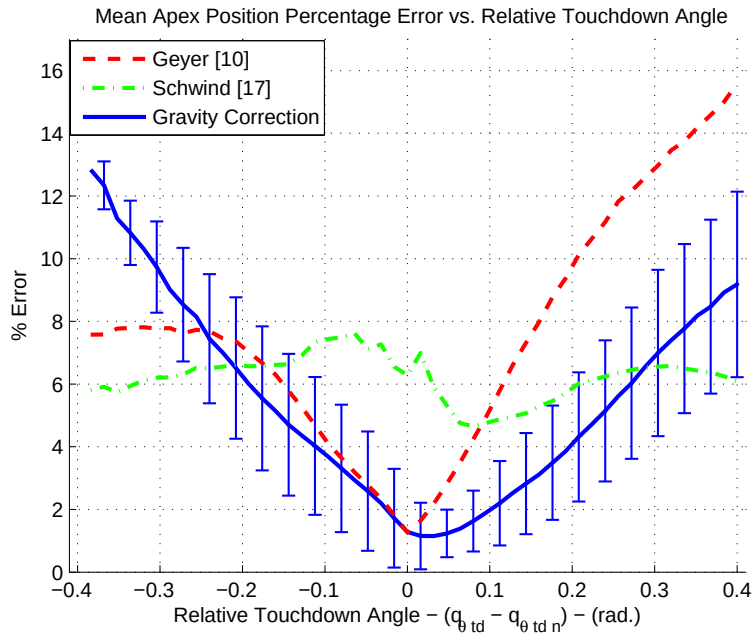

Figure 3.4: Mean Apex Position Percentage Error (PE_{ap}) versus Relative Touchdown Angle ($q_{\theta_{td}} - q_{\theta_{td_n}}$). The vertical bars represent standard deviations for the approximate stance map with gravity correction.

In almost all cases, our approximation performed better than Geyer's method for non-symmetric trajectories. Note that these two approximations are expected to perform identically for symmetric gaits (zero relative touchdown angle), which is also confirmed by their almost identical estimation performance for $q_{\theta_{td_{rel}}} \in [-0.1,\ 0]\ rad$.

On the other hand, Schwind's iterative approximation has an almost uniform performance profile, relatively independent of symmetry. Consequently, it performs better than our approximation for some non-symmetric trajectory regions far out into the touchdown angle spectrum. However, these regions correspond to rather extreme transient conditions unlikely to be observed for locomotion on reasonable terrain. Furthermore, some of their method's performance can be attributed to the fact that we used the 10th iterate of their approximations rather than more reasonable ones such as the first or second iterate for which the analytical nature of the approximations can still be preserved.

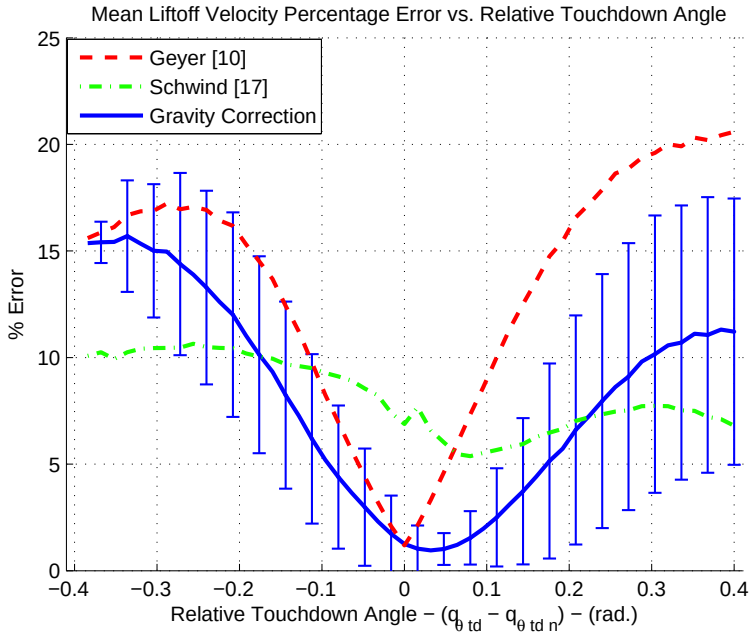

Figure 3.5: Mean Liftoff Velocity Percentage Error (PE_{lov}) versus Relative Touchdown Angle ($q_{\theta_{td}} - q_{\theta_{td_n}}$). The vertical bars represent the standard deviations for the approximate stance map with gravity correction.

Overall, our gravity correction scheme performs best for relative touchdown angles in the range of $[-0.2\ 0.2]$ rad. Fortunately, angles outside this range correspond to very sudden changes in the locomotion and can safely be left unused by a reasonable planner cognizant of the limitations of available approximations.

Finally, Figs. 3.6 and 3.7 illustrate regions in the control input space for which different approximants produce the best error performance. Schwind's method once again, is observed to have good performance far from symmetric gaits. When the relative touchdown angle is in the range $[-0.2,\ 0.2]$ rad, our gravity correction scheme has the best performance for both error measures. In Fig. 3.6, even though Geyer's approximations seem to be better than ours for larger leg stiffnesses and nearly symmetric gaits, their performance is actually almost identical to ours in those areas as can be observed from corresponding regions in Fig. 3.4.

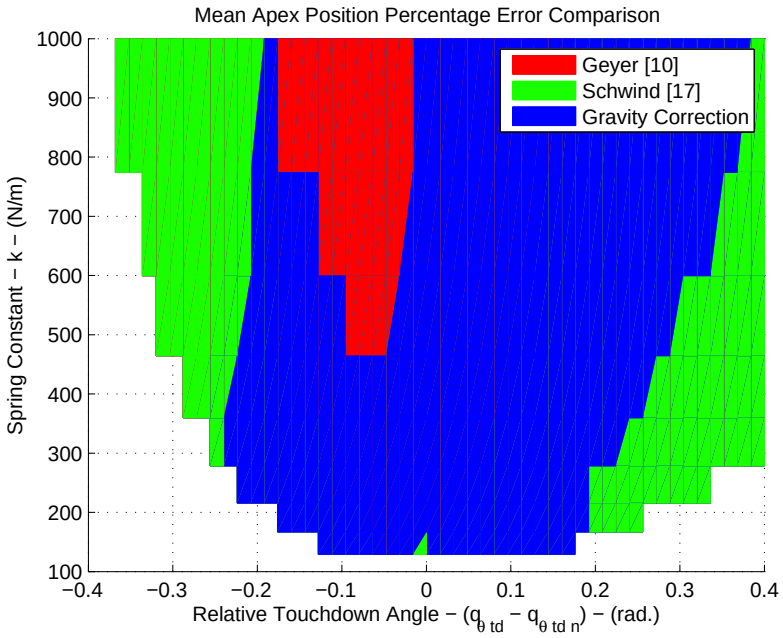

Figure 3.6: Comparison of the mean Apex Position Percentage Errors. Colored regions show where the associated approximation performs better.

As illustrated by these results, our proposed method improves the accuracy of approximations presented in [2]. When compared to Schwind's iterative approximations presented in [3], our approximation also performs better for the most commonly used subset of non-symmetric trajectories. It is important to note that the higher iterates of the Schwind approximations have much more complicated analytical forms, a significant handicap for the design of control algorithms and dynamic locomotion planning for SLIP. Our approximations, even with the gravity correction, have a very simple analytical form that can easily support control algorithms for dynamical locomotion and footstep planning for the SLIP model.

3.2.3 Discussion

In this section, we proposed a novel gravity correction method to improve the performance of previously proposed analytic approximations to the stance dynamics of the SLIP model in [2]. Our method is based on the hypothesis that for non-symmetric locomotion trajectories, the effect of gravity on the total angular

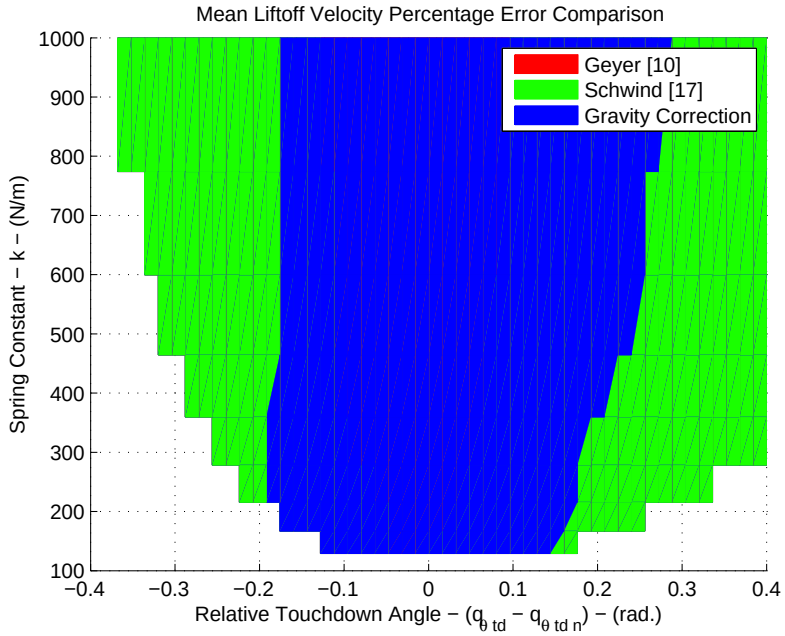

Figure 3.7: Comparison of the mean Liftoff Velocity Percentage Errors. Colored regions show where the associated approximation performs better.

momentum can be summarized with a constant correction term. We demonstrated that this is indeed the case through comparisons of estimation errors for both Geyer's method, our method as well as a different analytic approximation method proposed in [3]. Our method is found to perform best for relative touchdown angles in the range [-0.2 0.2] rad. For gaits that are sufficiently far from symmetric, Schwind's iterative stance map is observed to have better performance but under the condition that it is iterated until convergence, which usually results in unacceptable analytical complexity. Overall, our method seems to present the best combination of accuracy and simplicity for non-symmetric SLIP trajectories and is suitable for the design of footstep planning algorithms that rely on the use of transient stepping behavior.

Even though it is the scope of the following section, our approximations can also be easily applied to using tunable stiffness during stance as a control input as introduced in [9] and also embodied in the passive dynamics of the BiMASC leg [21]. This aspect turns out to be critical for nontrivial planning tasks with the

SLIP model since it allows inducing changes in the total energy of the system, allowing finer control over possible maneuvers.

The goal of the second part of thesis is the design of reactive planning controllers for the SLIP model, which can in turn be applied to more complex legged robots through passive or active embedding of the SLIP model. To this end, we believe that analytic approximations to the dynamic behavior of this model will be invaluable both in the design of controllers that can accurately and efficiently regulate its discrete control inputs as well as in the analytical characterization of such controllers for planning purposes. Our proposed method fills a gap in this area and provides an analytic approximation that remains valid for a large range of control inputs and initial conditions of the SLIP model.

3.3 An Approximate Stance Map for Two-Phase Stiffness Control

As mentioned earlier in Section 2.1.3, the SLIP template has relatively few control parameters since it is one of the simplest models for legged systems. While the touchdown leg angle is the most critical control parameter for stable locomotion, leg compliance parameters are also needed to adjust the total energy of the system such that desired properties of locomotion, apex velocity and height, can be achieved. In general, there are two different approaches to adjust stiffness properties of SLIP for controlling locomotion. First, the spring constant can be assumed constant during stance and the touchdown and liftoff leg lengths can be controlled to inject or take out energy from the system (Leg Length Control - LLC). In [38] and [39], a similar approach was used on an experimental robotic platform, the Bow-Leg hopping robot, such that the total energy of the robot was controlled by compressing the leg during flight to store energy. Also, in [42], a deadbeat gait controller for a biped was designed by adjusting

touchdown and liftoff leg lengths. The second approach is to separately control the spring constant during compression and decompression phases to modify the total system energy (Two-Phase Stiffness Control - TPSC). Simulation and experimental studies on this approach have been done by Raibert in [9] where the control of leg spring constant was done by adjusting air pressure in leg piston by a pneumatic actuator during flight phase to adjust the compression spring constant and at bottom to adjust the decompression spring constant.

In our studies, we will use the latter control approach together with the touchdown angle control for motion planning of SLIP in 2D. Consequently, we need an approximate stance map for the variable stiffness case. During the rest of the thesis, we will refer to the use of different stiffness for the compression and decompression phases as two-phase stiffness control. In the following sections, two different approximate stance maps built on [2] and [3] are introduced. The main difference of TPSC from other alternatives is that the apex return map should be divided into two parts, the apex to bottom map and the bottom to apex map, with different spring constants for each. The general idea for two-phase variable stiffness stance map is represented in Fig. 3.8.

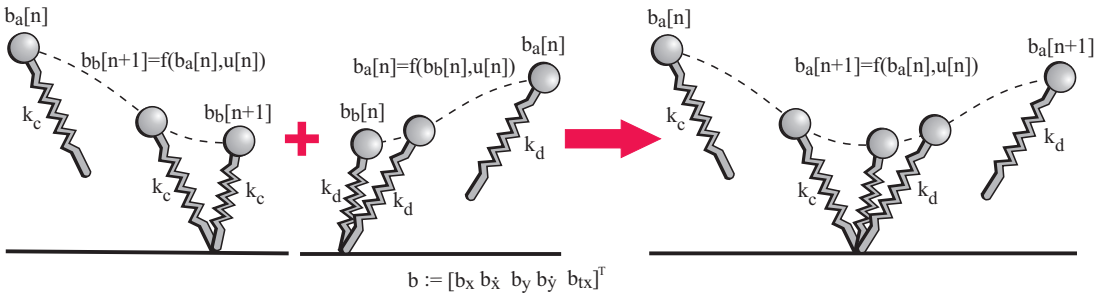

Figure 3.8: SLIP apex return map for two-phase variable stiffness case. The two-phase variable stiffness apex return map is composed of the apex to bottom map with compression phase leg stiffness, k_c, and the bottom to apex map with decompression phase stiffness, k_d.

3.3.1 Simple Approximate Stance Map For Two-Phase Variable Stiffness

In [2], an approximate apex return map was found as a function of model parameters, touchdown states, total energy and angular momentum while the spring constant was assumed to change during the stance phase. Also, since the ideal SLIP template has no damping, the total energy of the system is conserved. However, for the variable compliance case, the total energy of the model can be modified by controlling the spring constant at the bottom instant, so that the total energy changes during stance with a discrete change at bottom. In such cases, the apex return map can be obtained by considering the apex to bottom map and the bottom to apex map separately as in Fig. 3.8.

The Apex to Bottom Map

Using the results of [2], the apex to bottom map can be easily written as

$$q_r(t) = l_0(1 + a_{ab} + b_{ab}\sin(\hat{\omega}_{0_{ab}}t)), \tag{3.4}$$

$$q_\theta(t) = q_{\theta td} + \omega(1 - 2a_{ab})(t - t_{td}) + \frac{2b_{ab}\omega_{ab}}{\hat{\omega}_{0_{ab}}}[cos(\hat{\omega}_{0_{ab}}t) - cos(\hat{\omega}_{0_{ab}}t_{td})], \tag{3.5}$$

where $t_{td} \leq t \leq t_{b_{ab}}$ [3] and t_{td} and $t_{b_{ab}}$ are given by

$$t_{td} = \frac{1}{\hat{\omega}_{0_{ab}}}\left\{\pi - \arcsin(\frac{q_{r_{td}}/l_0 - 1 - a_{ab}}{b_{ab}})\right\}, \tag{3.6}$$

$$t_{b_{ab}} = \frac{1}{\hat{\omega}_{0_{ab}}}\left\{\frac{3}{2}\pi\right\}, \tag{3.7}$$

with the parameters $a_{ab}, b_{ab}, \omega_{ab}$ and $\hat{\omega}_{0_{ab}}$ defined as

$$\epsilon_{ab} = \frac{2E_{ab}}{m{l_0}^2}, \quad \omega_{ab} = \frac{p_\theta}{m{l_0}^2}, \quad \omega_{0_{ab}} = \sqrt{\frac{k_c}{m}}, \quad \hat{\omega}_{0_{ab}} = \sqrt{{\omega_{0_{ab}}}^2 + 3{\omega_{ab}}^2},$$

$$a_{ab} = \frac{{\omega_{ab}}^2 - g_s/l_0}{\hat{\omega}_{0_{ab}}^2},$$

$$b_{ab} = \sqrt{a_{ab}^2 + \frac{\epsilon_{ab} - {\omega_{ab}}^2 - 2g_s/l_0}{\hat{\omega}_{0_{ab}}^2}},$$

[3] Subscript ab stands for apex to bottom mapping.

where E_{ab} is the total mechanical energy at touchdown, conserved during the compression phase, and p_θ is the total angular momentum during stance, assumed to be conserved during the compression phase.

Finally, (3.6) can be simplified for a special case when the touchdown leg length, $q_{r_{td}}$, is kept equal to rest leg length, l_0, to yield

$$t_{td} = \frac{1}{\hat{\omega}_{0_{ab}}} \left\{ \pi - \arcsin(-\frac{a_{ab}}{b_{ab}}) \right\}. \tag{3.8}$$

The Bottom to Apex Map

The bottom to apex map has the same form as the apex to bottom mapping

$$\begin{aligned} q_r(t) &= l_0(1 + a_{ba} + b_{ba}\sin(\hat{\omega}_{0_{ba}}(t + t_{b_{ba}} - t_{b_{ab}}))), & (3.9) \\ q_\theta(t) &= q_{\theta b} + \omega(1 - 2a_{ba})(t - t_{b_{ab}}) \\ &\quad + \frac{2b_{ba}\omega_{ba}}{\hat{\omega}_{0_{ba}}}[cos(\hat{\omega}_{0_{ba}}(t + t_{b_{ba}} - t_{b_{ab}})) - cos(\hat{\omega}_{0_{ba}} t_{b_{ba}})], & (3.10) \end{aligned}$$

where $t_{b_{ab}} \leq t \leq t_{lo}$ [4]. Interestingly, the bottom times, $t_{b_{ab}}$ and $t_{b_{ba}}$, may be different since leg compliance now be diferent for the compression and decompression phases. Hence, we have to use a time shift $(t_{b_{ab}} - t_{b_{ba}})$ for the derivation of the bottom to apex map. As such, $t_{b_{ab}}$ is calculated by (3.7) and $t_{b_{ba}}$ and t_{lo} are given by

$$\begin{aligned} t_{b_{ba}} &= \frac{1}{\hat{\omega}_{0_{ba}}} \left\{ \frac{3}{2}\pi \right\}, & (3.11) \\ t_{lo} &= \frac{1}{\hat{\omega}_{0_{ba}}} \left\{ 2\pi + \arcsin(\frac{q_{r_{lo}}/l_0 - 1 - a_{ba}}{b_{ba}}) \right\} + t_{b_{ab}} - t_{b_{ba}}. & (3.12) \end{aligned}$$

Finally, (3.12) can be simplified for a special case when liftoff leg length, $q_{r_{lo}}$, is equal to rest leg length, l_0,

$$t_{l_0} = \frac{1}{\hat{\omega}_{0_{ba}}} \left\{ (2\pi + \arcsin(-\frac{a_{ba}}{b_{ba}}) \right\} + t_{b_{ab}} - t_{b_{ba}}. \tag{3.13}$$

[4] Subscript *ba* stands for bottom to apex mapping.

Remark 2. *The angular momentum, $p_\theta = mq_r^2 q_{\dot{\theta}}$, is assumed to be separately conserved during the compression and decompression phases. In order to check if the total angular momentum is conserved during the entire stance phase, we must consider the SLIP states just before and after the bottom instant. At bottom, we only change the spring constant, so there are discrete jumps on only the total energy and the spring force. Thus, system accelerations may have discrete jumps but velocity and position states are continuous at bottom. Since the total angular momentum depends only on position and velocity states, it remains continuous and constant during the entire stance phase.*

Remark 3. *At the bottom instant, the leg stiffness is changed based on a control algorithm and some amount of energy is injected or taken out from the springy leg. This additional energy is given by*

$$E_{additional} = \frac{1}{2}(k_d - k_c)(l_0 - q_{r_b})^2, \tag{3.14}$$

where q_{r_b} and t_b can be calculated by (3.4) and (3.7). Therefore, the total energy during the decompression phase, E_{ba}, is given by

$$E_{ba} = E_{ab} + E_{additional} = E_{ab} + \frac{1}{2}(k_d - k_c)(l_0 - q_r(t_b))^2. \tag{3.15}$$

The parameters $a_{ba}, b_{ba}, \omega_{ba}$ and $\hat{\omega}_{0_{ba}}$ are defined for (3.9)-(3.13) as

$$\begin{aligned}
\epsilon_{ba} &= \frac{2E_{ba}}{ml_0{}^2}, \quad \omega_{ba} = \frac{p_\theta}{ml_0{}^2} = \omega_{ab}, \quad \omega_{0_{ba}} = \sqrt{\frac{k_d}{m}}, \quad \hat{\omega}_{0_{ba}} = \sqrt{\omega_{0_{ba}}{}^2 + 3\omega_{ba}{}^2}, \\
a_{ba} &= \frac{\omega_{ba}{}^2 - g_s/l_0}{\hat{\omega}_{0_{ba}}^2}, \qquad (3.16) \\
b_{ba} &= \sqrt{a_{ba}^2 + \frac{\epsilon_{ba} - \omega_{ba}{}^2 - 2g_s/l_0}{\hat{\omega}_{0_{ba}}^2}}. \qquad (3.17)
\end{aligned}$$

Remark 4. *The above formulation of the bottom to apex map may cause discontinuities on the position and velocity states as in Fig. 3.9. This arises from the fact that the parameters of the bottom to apex map are solved without considering the continuity of position and velocity states at bottom. We know that q_{θ_b} and*

$q_{\dot{r}_b}$ are continuous due to the formulation of the solution, so we only need to constrain continuity of $q_r(t)$ and $q_{\dot{\theta}}(t)$ at bottom. Recall that the general expressions for $q_r(t)$ and $q_{\dot{\theta}}(t)$ were given by

$$q_r(t) = l_0(1 + a + b\sin(\hat{\omega}_0)t), \tag{3.18}$$

$$q_{\dot{\theta}}(t) = \omega(1 - 2a - 2b\sin(\hat{\omega}_0)t). \tag{3.19}$$

Figure 3.9: Left: Separate Approximate Stance Map for not variable compliance with spring constants k_c and k_d. Middle: Approximate Stance Map for Variable Stiffness with only parameter updates. A discontinuity is observed on stance map at bottom instance. Right: Approximate Stance Map for variable stiffness with parameter updates and velocity and position state continuity constraints

As indicated by Remark 2, the angular momentum is assumed to be constant during stance, i.e. $\omega_{ab} = \omega_{ba}$, and the continuity constraints can be written as

$$\begin{aligned} l_0(1 + a_{ab} + b_{ab}\sin(\hat{\omega}_{0_{ab}} t_{b_{ab}})) &= l_0(1 + a_{ba} + b_{ba}\sin(\hat{\omega}_{0_{ba}} t_{b_{ba}})), \\ \omega_{ab}(1 - 2a_{ab} - 2b_{ab}\sin(\hat{\omega}_{0_{ab}} t_{b_{ab}})) &= \omega_{ba}(1 - 2a_{ba} - 2b_{ba}\sin(\hat{\omega}_{0_{ba}} t_{b_{ba}})), \end{aligned}$$

Using (3.7) and (3.11), the relation between parameters a_{ab}, b_{ab}, a_{ba} and b_{ba} is found as

$$a_{ab} - b_{ab} = a_{ba} - b_{ba} \tag{3.20}$$

As such, (3.20) is the constraint on a_{ba} and b_{ba} for a continuous stance map with variable stiffness. Similarly, (3.16) and (3.17) are unconstraint parametric solutions for a_{ba} and b_{ba}. To prevent discontinuities at bottom for the stance

map, we assume that the trajectory of the decompression phase is invariant under small shift operations, so an offset term on $q_r(t)$ can be reformulated, this means that b_{ba} is given by (3.17) and the continuity constraint can be used for the calculation of a_{ba} as follows

$$a_{ba} = a_{ab} - b_{ab} + b_{ba}. \tag{3.21}$$

At this point, we can ensure the continuity of the stance map in such a way that a_{ba} is calculated by (3.16) and we calculate b_{ba} from the continuity constraint

$$b_{ba} = b_{ab} + a_{ba} - a_{ab}. \tag{3.22}$$

As a result, the whole stance map for the variable stiffness case is given by

$$q_r(t) = \begin{cases} l_0(1 + a_{ab} + b_{ab}\sin(\hat{\omega}_{0_{ab}}t)), & \text{if } t_{td} \le t \le t_{b_{ab}} \\ l_0(1 + a_{ba} + b_{ba}\sin(\hat{\omega}_{0_{ba}}(t + t_{b_{ba}} - t_{b_{ab}}))), & \text{if } t_{b_{ab}} \le t \le t_{lo} \end{cases} \tag{3.23}$$

$$q_\theta(t) = \begin{cases} q_{\theta td} + \omega_{ab}(1 - 2a_{ab})(t - t_{td}) \\ \quad + \frac{2b_{ab}\omega_{ab}}{\hat{\omega}_{0_{ab}}}[cos(\hat{\omega}_{0_{ab}}t) - cos(\hat{\omega}_{0_{ab}}t_{td})], \text{ if } t_{td} \le t \le t_{b_{ab}} \\ q_{\theta b} + \omega_{ba}(1 - 2a_{ba})(t - t_{b_{ab}}) \\ \quad + \frac{2b_{ba}\omega_{ba}}{\hat{\omega}_{0_{ba}}}[cos(\hat{\omega}_{0_{ba}}(t + t_{b_{ba}} - t_{b_{ab}})) - cos(\hat{\omega}_{0_{ba}}t_{b_{ba}})], \text{ if } t_{b_{ab}} \le t \le t_{lo} \end{cases} \tag{3.24}$$

Furthermore, the continuous stance map can be obtained by vertically shifting the bottom to apex map. However, even though this shifting can guarantee a continuous stance trajectory, it may not handle the continuity of velocity components at bottom.

Note that, as in [2] the derivation of the approximate stance map for the variable stiffness case, (3.23) and (3.24), also assumes conservation of angular momentum. However, this assumption fails for nonsymmetric gaits as explained in Section 3.2. Nevertheless we may still compensate for the effect of gravity on the angular momentum using the idea of gravity correction introduced in Section 3.2. The effect of gravity during the compression, $P_{c_{ab}}$, and decompression, $P_{c_{ba}}$,

phases can be approximated using (3.3) with corresponding initial and final time of subphases. Then, the corrected angular momenta for the compression and decompression phases are given by

$$\begin{aligned} \hat{p}_{\theta_{ab}} &= p_\theta + P_{c_{ab}}, \\ \hat{p}_{\theta_{ba}} &= \hat{p}_{\theta_{ab}} + P_{c_{ba}}, \end{aligned}$$

which replace p_θ in the corresponding derivation step of the simple two-phase variable stiffness approximation.

3.3.2 Iterative Approximate Stance Map For Two-Phase Variable Stiffness

In this section, we present the derivation of an iterative approximate stance map for two-phase variable stiffness control using the methods of [3] and the idea of partitioning the apex return map into the apex to bottom and the bottom to apex maps as in Fig. 3.9. In fact, this form of the apex return map is the same as in Section 2.2.2 with the main difference due to different leg stiffness for the compression and decompression phases. Consequently, we only need to use two different hamiltonian functions and their inverses.

The Apex to Bottom Map

The Hamiltonian function for the compression phase is given by

$$H_{ab} = \frac{1}{2m}\left(p_r^2 + \frac{p_\theta^2}{q_r^2}\right) + \frac{1}{2}k_c(l_0 - q_r)^2 + mgq_r\cos(q_\theta).$$

Solving the equation $H_{ab}(q_r, p_r, q_\theta, p_\theta) = E_{ab}$ for p_r yields

$$p_r = H_{ab}^{\dagger}(q_r, q_\theta, p_\theta, E_{ab}) := \sqrt{2m\left(E_{ab} - \frac{1}{2}k_c(l_0 - q_r)^2 - mgq_r\cos(q_\theta)\right) - \frac{p_\theta^2}{q_r^2}},$$

where E_{ab} is the conserved total mechanical energy of the system during the compression phase. The iterative touchdown to bottom map hence takes the

form

$$\hat{t}_{s_{(n+1)}}(q_r) = t_{td} - m(q_r - q_{r_{td}})/H^{\dagger}_{abn},$$
$$\hat{q}_{\theta_{(n+1)}}(q_r) = q_{\theta_{td}} - \hat{p}_{\theta_n}(\hat{\xi}_r)(q_r - q_{r_{td}})/(\hat{\xi}_r^2 H^{\dagger}_{abn}),$$
$$\hat{p}_{\theta_{(n+1)}}(q_r) = p_{\theta_{td}} - m^2 g\hat{\xi}_r \sin(\hat{q}_{\theta_n}(\hat{\xi}_r))(q_r - q_{r_{td}})/H^{\dagger}_{abn},$$
$$\hat{p}_{r_{(n+1)}}(q_r) = -H^{\dagger}_{abn+1},$$

where n is the iteration number, $\hat{\xi}_r = 3q_{r_{td}}/4 + q_r/4$ and $H^{\dagger}_{abk} := H^{\dagger}_{ab}(\hat{\xi}_r, \hat{q}_{\theta_k}(\hat{\xi}_r), \hat{p}_{\theta_k}(\hat{\xi}_r), E_{ab})$. The zeroth (initial) iteration can be any approximate analytical solution for the stance phase as in [3]. Also, the approximate mapping from apex to bottom in Section 3.3.1 may be a good initial iteration for this approximation.

The Bottom to Apex Map

In this case, the Hamiltonian function for the decompression phase, H_{ba}, and the solution of the equation $H_{ba}(q_r, p_r, q_\theta, p_\theta) = E_{ba}$ for p_r are given by

$$H_{ba} = \frac{1}{2m}\left(p_r^2 + \frac{p_\theta^2}{q_r^2}\right) + \frac{1}{2}k_d(l_0 - q_r)^2 + mgq_r\cos(q_\theta),$$
$$p_r = H^{\dagger}_{ba}(q_r, q_\theta, p_\theta, E_{ba}) := \sqrt{2m\left(E_{ba} - \frac{1}{2}k_d(l_0 - q_r)^2 - mgq_r\cos(q_\theta)\right) - \frac{p_\theta^2}{q_r^2}},$$

where E_{ba} is the conserved total energy during the decompression phase. The relation between E_{ba} and E_{ab} is given by (3.15).

With these definitions, the approximate iterative bottom to apex map is given by

$$\hat{t}_{s_{(n+1)}}(q_r) = t_b + m(q_r - q_{r_b})/H^{\dagger}_{ban},$$
$$\hat{q}_{\theta_{(n+1)}}(q_r) = q_{\theta_b} + \hat{p}_{\theta_n}(\hat{\xi}_r)(q_r - q_{r_b})/(\hat{\xi}_r^2 H^{\dagger}_{ban}),$$
$$\hat{p}_{\theta_{(n+1)}}(q_r) = p_{\theta_b} + m^2 g\hat{\xi}_r \sin(\hat{q}_{\theta_n}(\hat{\xi}_r))(q_r - q_{r_b})/H^{\dagger}_{ban},$$
$$\hat{p}_{r_{(n+1)}}(q_r) = H^{\dagger}_{ban+1},$$

where n is the iteration number, $\hat{\xi}_r = 3q_{r_b}/4 + q_r/4$ and $H^{\dagger}_{bak} := H^{\dagger}_{ba}(\hat{\xi}_r, \hat{q}_{\theta_k}(\hat{\xi}_r), \hat{p}_{\theta_k}(\hat{\xi}_r), E_{ba})$. The zeroth iteration can be any approximate analytical solution for stance phase. In [3], three different initial approximate solutions are used and the approximation in Section 3.3.1 based on [2] may be a good initial iteration for this iterative approximation.

The general formulation of the iterative apex return map for two-phase variable stiffness control can hence be derived with one missing important parameter: the bottom leg length, q_{r_b}. The bottom leg length is crucial since the apex return map is divided into two parts around the bottom instant. Unfortunately, an exact solution of the bottom length cannot be found, but several approximate solutions can be used. The first alternative is obtained by using the symmetric gait assumption with constant leg stiffness , i.e $k_c = k_d$, and the approximate bottom length can be calculated by using the total energy relation below since SLIP is vertical at bottom for symmetric gaits:

$$E_{ab} = \frac{p_\theta}{2mq_r^2} + \frac{k_c}{2}(l_0 - q_r)^2 + mg_s q_r, \tag{3.25}$$

$$\frac{k_c}{2}q_r^4 + (mg_s - k_c l_0)q_r^3 + \left(\frac{k_c l_0^2}{2} - E_{ab}\right)q_r^2 + \frac{p_\theta^2}{2m} = 0. \tag{3.26}$$

Note that (3.26) is a quartic equation for the bottom leg length and the solution to this quartic equation which is real and less than or equal to the rest leg length gives the bottom leg length.

Another way of approximately calculating the bottom leg length is to use the approximate stance map of [2] up to the bottom instant. using this method, the approximate bottom leg length, q_{r_b}, is given by

$$q_{r_b} = l_0(1 + a_{ab} - b_{ab}) \tag{3.27}$$

where a_{ab} and b_{ab} are the same as in Section 3.3.1.

3.3.3 Performances of Proposed Approximations

Simulation Environment and Performance Criteria

Our interest in analytic approximations to the SLIP stance dynamics for variable stiffness arises from one of the possible control modalities, the Two-Phase Stiffness Control (TPSC). We want to design position aware deadbeat controllers which will enables us the control and planning of legged locomotion for a specified high level tasks. Deadbeat controllers that appear in the literature mostly concentrate on apex height and apex velocity to obtain desired gait properties without dealing with the horizontal position of the system in the environment. However, to achieve realistic planning for legged locomotion over rough terrain, we need to consider both gait properties and the robot position. Consequently, position aware deadbeat controllers are necessary for automated legged locomotion over rough surfaces.

Position aware deadbeat controllers require good approximations to SLIP stance dynamics and in this section we investigate how well the two-phase variable stiffness approximation captures dynamical SLIP behavior for the TPSC control mode. In doing this, the most important performance criteria for us is the accuracy of the predicted apex position and velocity as in Section 3.2. To this end, we use two measures to quantify the prediction performance of approximations described above: Normalized percentage errors in the apex position and liftoff velocity predictions, defined respectively as:

$$\begin{aligned} PE_{ap} &= 100\ \frac{||(b_{x_a}, b_{y_a}) - (\hat{b}_{x_a}, \hat{b}_{y_a})||_2}{||(b_{x_a}, b_{y_a})||_2}, \\ PE_{lov} &= 100\ \frac{||(b_{\dot{x}_{lo}}, b_{\dot{y}_{lo}}) - (\hat{b}_{\dot{x}_{lo}}, \hat{b}_{\dot{y}_{lo}})||_2}{||(b_{\dot{x}_{lo}}, b_{\dot{y}_{lo}})||_2}. \end{aligned}$$

As in Section 3.2, we use the liftoff velocity rather than the apex velocity to ensure that normalization is practical even for non-symmetric gaits for which the apex velocity may become zero or very small in magnitude.

In order to compare prediction performance of the proposed approximations, we ran simulations spanning five different dimensions of initial states and control inputs: the apex height (b_{y_a}), the apex velocity ($b_{\dot{x}_a}$), the compression phase spring constant (k_c), the stiffness ratio (k_d/k_c) and the relative touchdown angle ($q_{\theta_{td_{rel}}} := q_{\theta_{td}} - q_{\theta_{td_n}}$[5]). The ranges considered for these dimensions were determined based on experimental observation in the biomechanics literature as well as structural properties of various legged robots as in Section 3.2. Therefore, our choice of simulation parameters and control input ranges are listed Table 3.2.

Table 3.2: Simulation Parameters

m(kg)	$l_0(m)$	g(m^2/s)	b_{y_a} (m)	$b_{\dot{x}_a}(m/s)$	$k_c(N/m)$	k_d/k_c	$q_{\theta_{td_{rel}}}(rad)$
1	1	9.81	1.1 - 1.5	0 - 5	250 - 1000	0.5 - 2	-0.25 - 0.25

For each of our simulations, we checked whether they satisfied conditions presented in Section 3.2.2 to ensure that we can support meaningful comparisons of all approximations and preserve similarity to existing analytical models of SLIP stance dynamics. From among a total of 258570 initial states and control inputs, 216770 were found to satisfy these two conditions.

Then, we computed approximate estimates of the apex states based on three proposed approximations in the previous section and compared estimation performances with "ground truth" using the error criteria defined above. The first approximation is the simple variable stiffness approximation, Simple VS Approximation, described in Section 3.3.1 where a_{ba} is calculated from parameter definitions (3.16) and b_{ba} is found by continuity equation (3.22). We must note that there are several ways of satisfying continuity at bottom instant as mentioned in Section 3.3.1, but the most reliable one is that parameter b_{ba} which is calculated from continuity equation. The second approximation for the variable stiffness

[5] $q_{\theta_{td_n}}$ is the neutral touchdown angle, which can be defined as a touchdown leg angle that results in a vertical SLIP state at bottom instant and depends on the initial conditions. When k_d/k_c is equal to one, then the neutral touchdown angle result in symmetric steps. For each simulation, we numerically calculated this angle and it is used as the origin for our plots.

case is iterative approximation, Iterative VS Approximation, based on [3] and we used the 10th iterate (after which further iterations yield no improvements) to make sure we obtained the best possible performance for this approximation. Finally, the last approximation model uses the gravity correction method and has the same form as the simple variable stiffness approximation. In other words, it is the gravity corrected version of the first approximation.

Simulation Results

Our simulation results are illustrated in Fig. 3.10, where we show the mean, standard deviation and maximum values for the apex position and liftoff velocity percentage errors, PE_{ap} and PE_{lov}, across all valid simulations and all three approximation methods. First two approximations, the simple variable stiffness (VS) and the iterative VS approximations, are derived using the results of [2] and [3]. Third approximation is obtained by modifying the results of [35]. Our results show that there is a notable performance improvement on the average for the gravity correction method compared both to the simple VS and the iterative VS approximations.

A more informative comparison between different approximation alternatives can be achieved by investigating the estimation performance as a function of the relative touchdown angle. Since our gravity correction method is expected to perform well for non-symmetric trajectories, its error performance should be better than the alternatives for nonzero relative angle values. As illustrated by Fig. 3.11, this was indeed the case for both error measures, PE_{ap} and PE_{lov}.

In all cases, our variable stiffness approximation with gravity correction performs better than other approximations for non-symmetric trajectories. In fact it performs best for relative touchdown angles in the range of $[-0.25\ 0.25]\ rad$. On

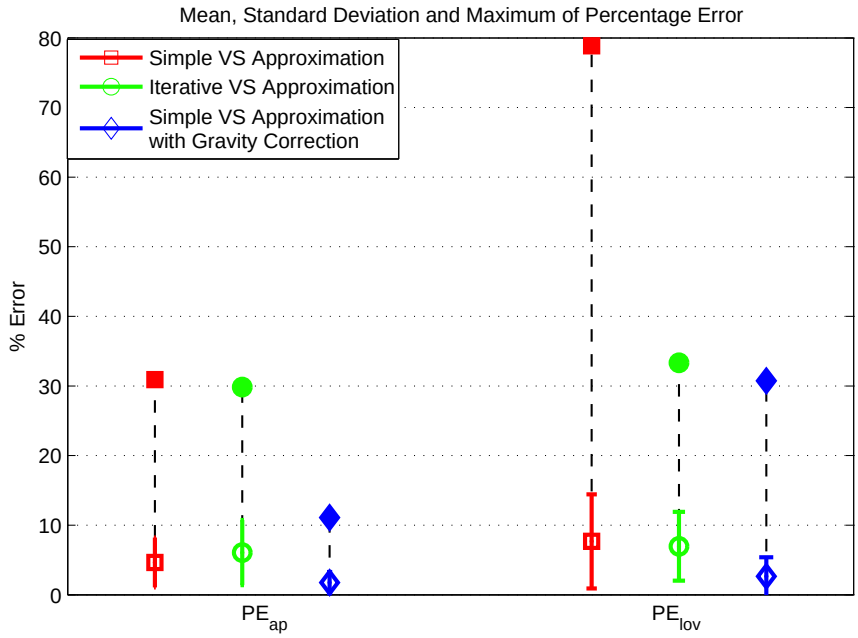

Figure 3.10: Estimation performances for the stance maps of *Simple*, *Iterative* and *Simple Gravity Corrected* Variable Stiffness (VS) Approximations. PE_{ap} (left) and PE_{lov} (right) are apex position and liftoff velocity percentage errors. Empty markers, filled markers and colored vertical bars represent mean, maximum and standard deviations of associated approximations.

the other hand, iterative approximation has an almost uniform performance profile, relatively independent of symmetry. Consequently, it performs better than our approximation for some non-symmetric trajectory regions far out into the touchdown angle spectrum. However, these regions correspond to rather extreme transient conditions unlikely to be observed for locomotion on reasonable terrain, so we limited our simulation studies to frequently used relative touchdown angle range. Furthermore, some of its performance can be attributed to the fact that we used the 10th iterate of iterative approximation rather than more reasonable ones such as the first or second iterate for which the analytical nature of the approximations can still be preserved.

Another detailed comparison between different approximation methods can be achieved by investigating the estimation performance as a function of the stiffness ratio, k_d/k_c. In Fig. 3.12, we present average apex position and liftoff velocity mapping performances of all three approximation and once again our

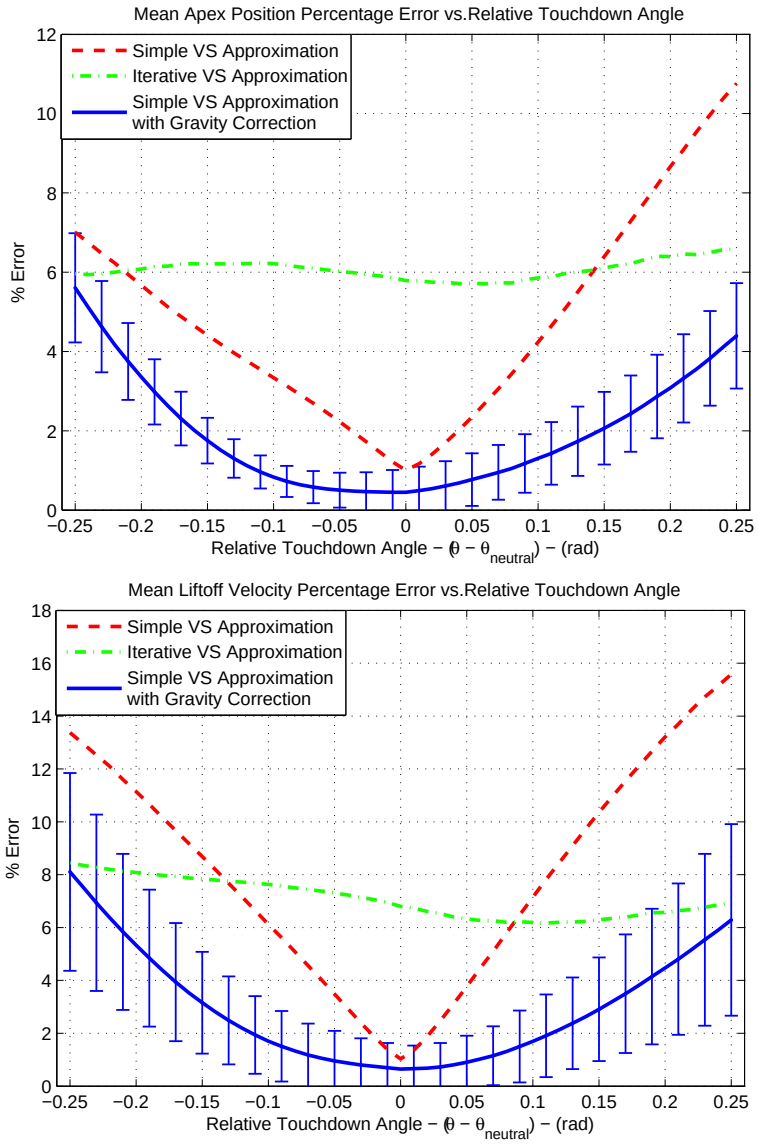

Figure 3.11: Upper:Mean Apex Position Percentage Error (PE_{ap}) versus Relative Touchdown Angle ($q_{\theta_{td}} - q_{\theta_{td_n}}$). Lower: Mean Liftoff Velocity Percentage Error (PE_{lov}) versus Relative Touchdown Angle ($q_{\theta_{td}} - q_{\theta_{td_n}}$). The vertical bars represent standard deviations for the approximate stance map with gravity correction.

simple variable stiffness approximation with gravity correction is the best one for stiffness ratio, k_d/k_c, in the range of [0.5 2]. In reality, this range of stiffness ratio is acceptable and practical since we may suddenly double or halve the spring potential energy, in other words, a significant amount of total system energy can be controllable. Moreover, Fig. 3.12 illustrates that the apex position estimation performance becomes worse with increasing stiffness ratio. In other words, the apex position estimation performance worsen as expected with increasing liftoff energy (or liftoff velocity). On the contrary, the liftoff velocity estimation

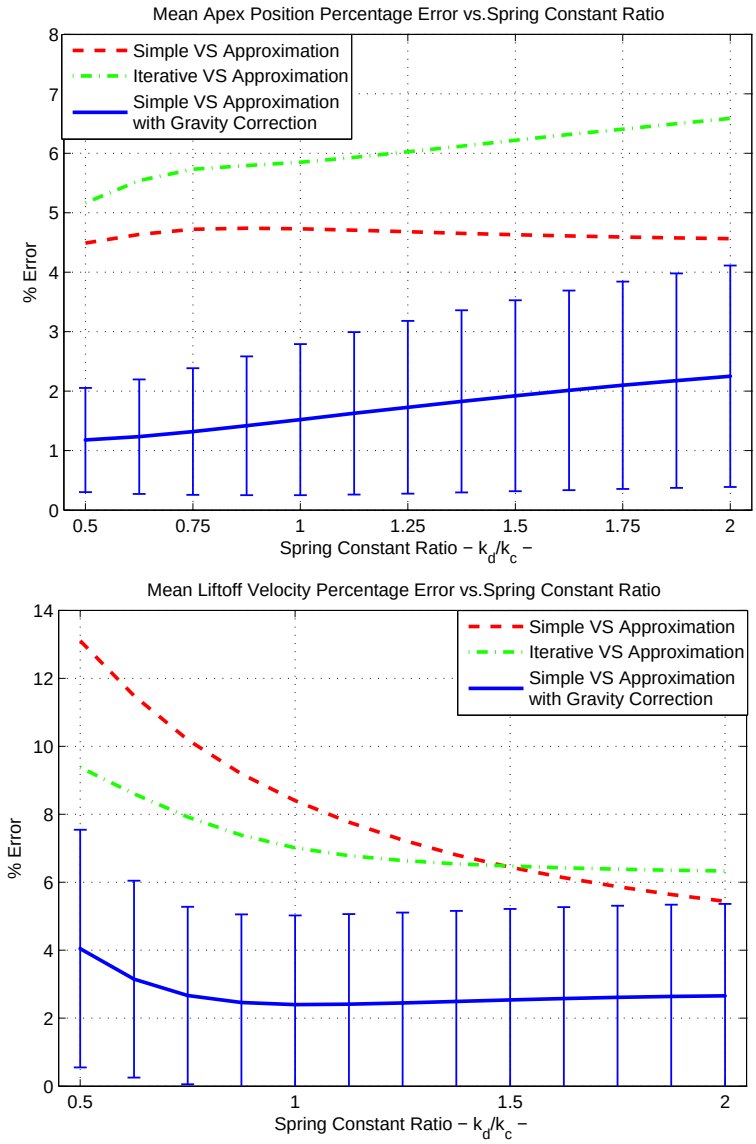

Figure 3.12: Upper:Mean Apex Position Percentage Error (PE_{ap}) versus Stiffness Ratio (k_d/k_c). Lower: Mean Liftoff Velocity Percentage Error (PE_{lov}) versus Stiffness Ratio (k_d/k_c). The vertical bars represent standard deviations for the approximate stance map with gravity correction.

becomes better with increasing stiffness ratio since the expected liftoff velocity increases in this case, making the percentage error become smaller, assuming almost constant absolute prediction error.

On the whole, our approximate map with gravity correction is the best predictor for the behavior of the spring mass hopper on the average for significant and practical ranges of control parameters and initial configurations. In this sense, in Figures 3.13, 3.14, 3.15 and 3.16, we illustrate the relations between the estimation performance and relative touchdown angle, compression leg stiffness,

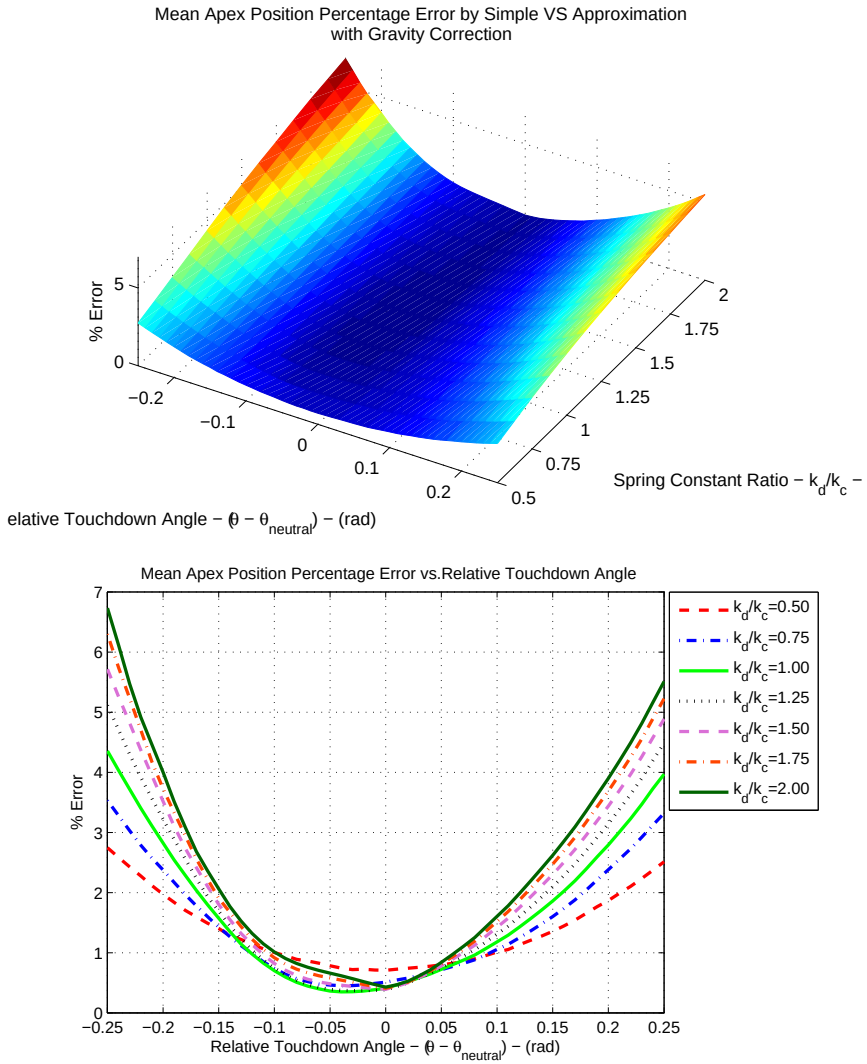

Figure 3.13: Upper:Mean Apex Position Percentage Error vs Stiffness Ratio and Relative Touchdown Angle. Lower: Mean Apex Position Error vs Relative Touchdown Angle at different Stiffness Ratio.

and stiffness ratio for two-phase variable stiffness approximation with gravity correction. As previously observed, apex position prediction performance increases with decreasing stiffness ratio (i.e. decreasing system energy or liftoff velocity) and increasing compression phase leg stiffness. Because increasing leg stiffness decreases the amount of leg compression, and so the actual SLIP motion becomes closer to approximation assumptions. Note that, the accuracy of the liftoff velocity estimation becomes better with increasing stiffness ratio and compression phase leg stiffness. Similarly, the actual motion has very similar characteristics to the assumption of the approximation. Finally, the dependence of both the apex

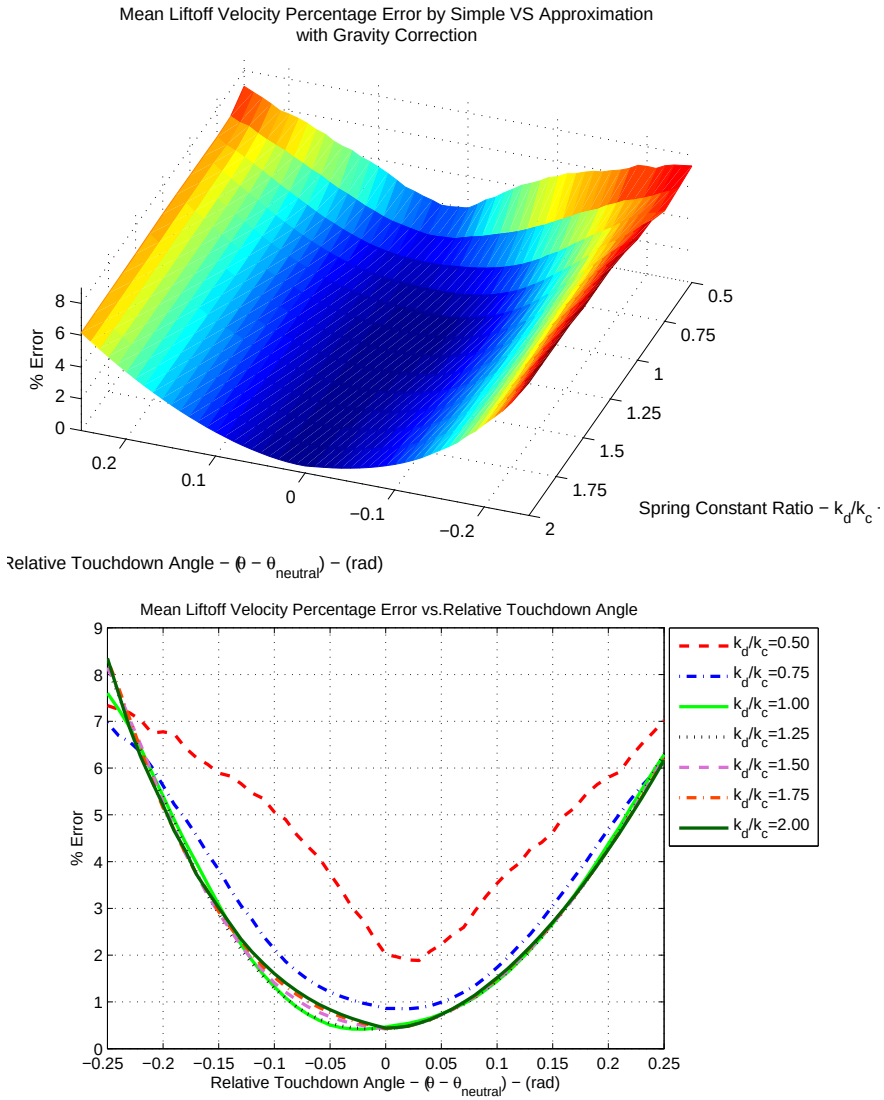

Figure 3.14: Upper:Mean Liftoff Velocity Percentage Error vs Stiffness Ratio and Relative Touchdown Angle. Lower: Mean Liftoff Velocity Error vs Relative Touchdown Angle at different Stiffness Ratio.

position and the liftoff velocity predictions on the relative touchdown angle has a general characteristic due to approximation assumptions such that mapping performance is better around neutral touchdown angle, i.e. $q_{\theta_{td_{rel}}} = 0$. Overall, our gravity correction method performs best for relative touchdown angles in the range of $[-0.25\ 0.25]$ *rad*. Fortunately, angles outside this range correspond to very sudden changes in the locomotion and can safely be left unused by a reasonable planner cognizant of the limitations of available approximations.

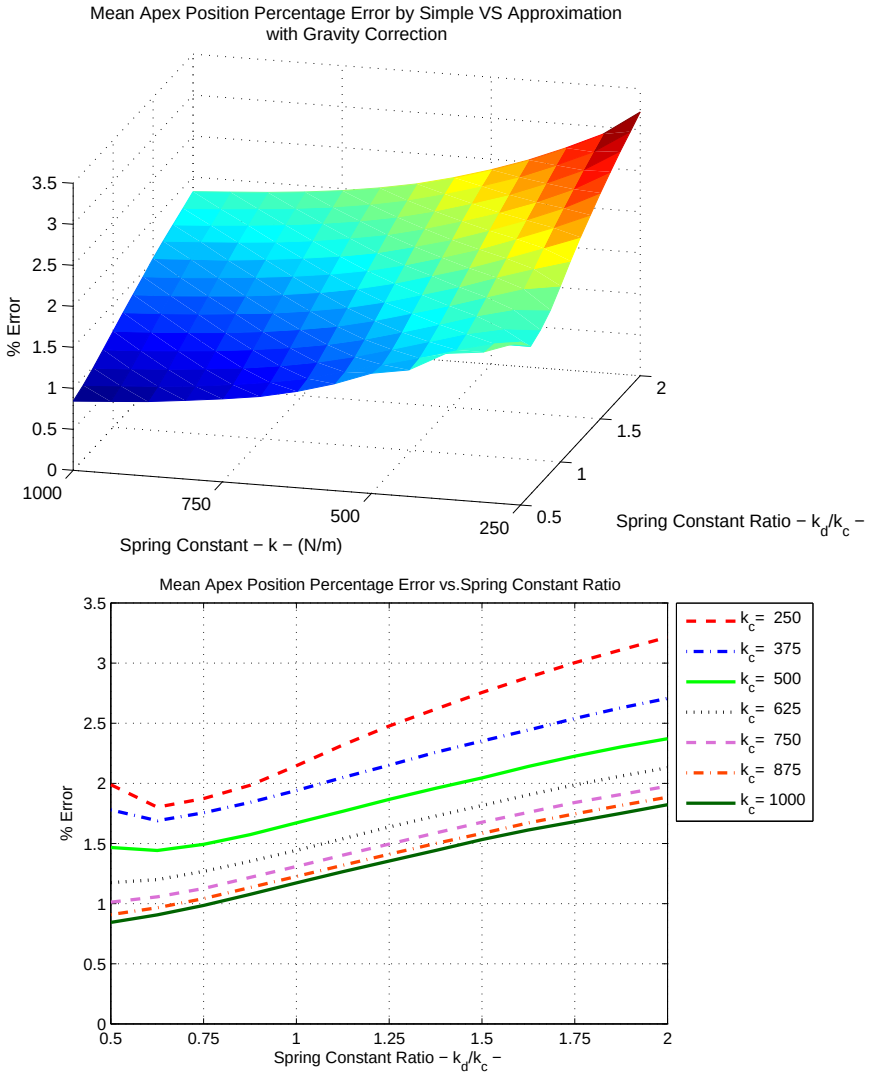

Figure 3.15: Upper:Mean Apex Position Percentage Error vs Stiffness Ratio and Compression Phase Leg Stiffness. Lower: Mean Apex Position Error vs Stiffness Ratio at different Compression Phase Leg Stiffness.

3.3.4 Discussion

In this section of the thesis, we proposed three novel approximation for two-phase variable stiffness control based on [2], [3] and [35] and we illustrate the effect of gravity correction on the performance for estimating SLIP trajectories. The general idea behind the two-phase variable stiffness approximation is division of the apex return map into two parts: the apex to bottom and the bottom to (next) apex maps. As a result, we can separately control leg compliance during compression and decompression to reach a desired apex position and apex speed.

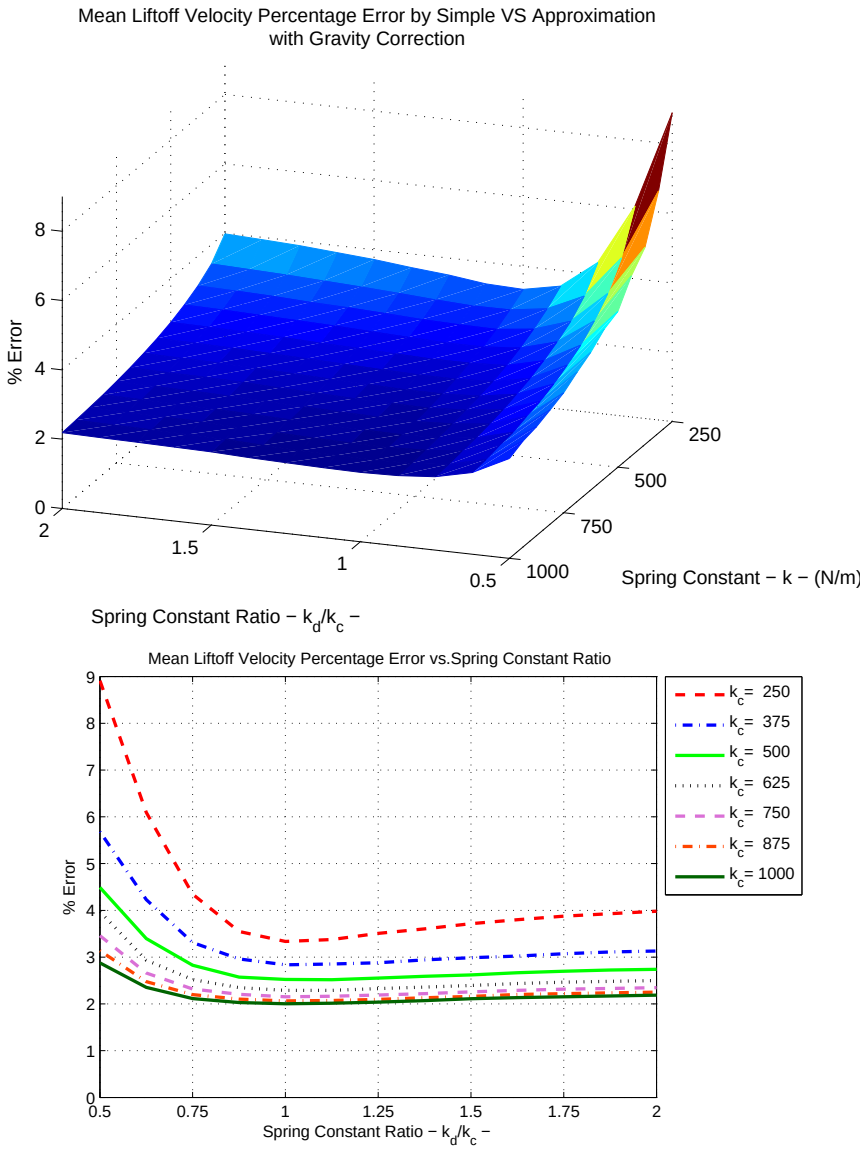

Figure 3.16: Upper:Mean Liftoff Velocity Percentage Error vs Stiffness Ratio and Compression Phase Leg Stiffness. Lower: Mean Liftoff Velocity Error vs Relative Touchdown Angle at different Compression Phase Leg Stiffness.

In fact, position aware-deadbeat controllers resulting from this mode of control differ from previous controllers in a way that previous approaches only deal with gait properties (apex height and velocity) instead of horizontal system position as well. However, legged locomotion planning requires both horizontal system position and gait control to ensure safe, reliable and robust mobility. Therefore, motion prediction performance of a candidate approximation becomes critical.

The average performance of our variable stiffness approximation with gravity correction was found to be highest in both its apex position and liftoff velocity prediction performances for relative touchdown angles in the range $[-0.25\ 0.25]$ *rad.*. For gaits that are sufficiently far from symmetric, iterative stance map for two-phase variable stiffness is observed to have better performance but under the condition that it is iterated until convergence, which usually results in unacceptable analytical complexity. Overall, the gravity corrected simple variable stiffness approximation seems to present the best combination of accuracy and simplicity for non-symmetric SLIP trajectories and is suitable for the design of two-phase stiffness controllers used by footstep planning algorithms that rely on the use of transient stepping behavior for robots likes Raibert's hopper [9] and the BiMASC leg [21]. This aspect turns out to be critical for nontrivial planning tasks with the SLIP model since it allows inducing changes in the total energy of the system, allowing finer control over possible maneuvers.

In the second part of the thesis, we will discuss the design of reactive planning controllers for the SLIP model, which can in turn be applied to more complex legged robots through passive or active embedding of the SLIP model. To this end, we believe that analytic approximations to the dynamic behavior of this model will be important both in the design of position aware deadbeat controllers that can accurately and efficiently regulate its discrete control inputs as well as in the analytical characterization of such controllers for planning purposes. Our proposed method fills a gap in this area and provides an analytic approximation that remains valid for a large range of control inputs and initial conditions of the SLIP model.

Chapter 4

BACKGROUND: CONTROL AND PLANNING OF LEGGED LOCOMOTION OVER ROUGH TERRAIN

Robotics is a comprehensive field of engineering and science with a large number of application areas (e.g. military, manufacturing, automation, rescue, medical). As a result, there are different robot motion planning and control problems in various environments with diverse robot morphologies and capabilities. In this extremely wide research area, we are particularly interested in legged locomotion and especially its control and planning over rough terrain. This chapter starts the second part of the thesis: Reactive footstep planning of legged systems over rough surfaces. We will first summarize previous approaches on footstep planning and control of legged systems in the following sections. According to our observations of studies appeared in the literature, existing approaches can be classified as: either static and quasistatic (e.g. walking and crawling) or dynamic (e.g. running and trotting) legged locomotion. On the other hand, an equally

important problem is the solution of control and planning for running robots. Traditional approaches to this problem divide it into two separate problems: Planning of the solution strategy and the necessary control for its realization. Solving these subproblems individually and combining them later gives the final solution. However, this approach doesn't work very efficiently and is very sensitive to modelling uncertainty and environment noise for complex problems like kinodynamic motion planning and dynamical legged locomotion. A new trend in robotics is the integration of planning, control and high level tasks for more reliable algorithms for autonomous systems and real time applications. An earlier method for simultaneous planning and control is the sequential composition [30]. This chapter continues with the background on static and dynamic legged locomotion and sequential composition.

4.1 Planning and Control of Static and Quasistatic Legged Locomotion

Initial studies on human motion planning concentrated on statically stable locomotion and started with simple motion model by projecting the 3D environment of a human character onto approximate 2D models such as differential drive circular robot approximation since researchers mainly looked for possible collision free locomotion trajectories from a start point to a goal point. In [47], the navigation problem of an animated character in a maze is studied and the character is assumed to be bounded by an appropriate cylinder and a 2D circular differential drive robot model is used to characterize human motion. In fact, a quasi-nonholonomic system assumption is reasonable since experimental observations on human locomotions shows the nonholonomic nature of human locomotion [48]. Also, the human motion planning algorithm in [47] assumes obstacles with infinite height and that the human character cannot take a step

over or onto any obstacle. However, in reality, legged systems are able to step over or onto obstacles which is a unique and distinctive feature of such systems.

Additionally, in [27], an offline approach is introduced in a simulation environment while considering the capability of the system to step over obstacles. In subsequent studies, obstacles are modelled as holes in the floor so that robot can take a step over these holes and several simulation and experimental online planning studies on bipeds are performed for statically stable locomotion [29, 49, 50]. In [28, 51], the problem of biped navigation over complex environments is investigated using full static locomotion capability (i.e. stepping over and upon obstacles).

In contrast, [52] shows a different approach to find more realistic planning trajectories. The algorithm proposed in [52] is based on planning for static locomotion and the dynamical consistency is preserved by using dynamical filtering. To sum up, most of these studies focus on statically stable locomotion and they are not suitable and sufficient for dynamical legged locomotion.

Finally, there is also significant research on footstep planning for quadrupeds and hexapods, mostly concentrating on statically stable locomotion. In contrast, we focus on monopod or biped robots since their morphologies are much more similar to the spring mass hopper.

4.2 Planning and Control of Dynamic Legged Locomotion

Motion planning and control of second order systems, also known as kinodynamic motion planning, has always been challenging and it has not been yet completely solved. Since dynamic analysis and the design of a universal controller for all

possible motion of a specific robot is difficult or may be even impossible, researchers use different approximate solutions with limited capabilities or probabilistic approaches for kinodynamic planning. A good example for gait planning of humanoid robot is presented in [53], where a sampling based approach is used while considering both kinematics and dynamics of the system. Sampling based approaches are excellent ways of solving the control problem for complex and high dimensional systems and are mostly used for offline planning. However, they are not immediately applicable to online planning for highly dynamic locomotion. For this kind of applications, dynamical behavior needs to be characterized well and controllers should be designed according to system dynamics.

In [9], several control structures are proposed for the control of dynamical locomotion, based on good characterizations of system dynamics. These locomotion control algorithms are used to adjust step lengths over rough terrain [26]. These studies concentrate on dynamical locomotion problem over rough surfaces and their methodology separates the navigation and control problems. As a first step, footholds to be used during locomotion are determined and the control policy tries to achieve the resulting offline plan as accurate as possible while ensuring the balance of the hopper. There are several limitations of this approach due to its nonreactive feature against changing conditions.

Furthermore, the bow leg robot is an accomplished monopod robot with compliance which has 2D and 3D versions. There are many dynamical locomotion research over rough surfaces with discrete stones and obstacles [1, 38, 39, 54]. The approaches presented in these studies are based on offline planning of possible foot placements to realize the specified task and stepwise feedback control. In these studies, the attention is also given to the limitation of offline footstep planning for long range locomotion and necessity of integrated planning and control.

4.3 Simultaneous Planning and Control via Sequential Composition

Traditional robot motion planning and control approaches separate these strongly related issues into two independent planning and control problems and tries to solve them individually. The independence of motion planning and control is acceptable for static environments and slow systems. However, this approach has limited capabilities or sometimes does not work for changing conditions and highly dynamic robots. The main reason for insufficiency of this approach is that motion planning need to consider capabilities of existing control policies for dynamic robot behaviors.

On the other hand, an alternative framework for integrating motion planning and control is the composition of controllers [30]. For a specified planning problem, design of a global controller is generally very difficult or sometimes impossible for complex systems. However, the sequential composition method proposes a switching control strategy such that the desired task of the global controller can be achieved by an autonomous switching of existing local controllers. There are several requirements for a valid local control policy such as:

- Safety: the domain of attraction of a policy must be in free configuration space,
- Convergence In Finite Time: if a given state is inside the domain of attraction it must converge to policy goal set in finite time,
- Conditionally Invariant: if a given state is inside the domain of attraction of a local policy, then it must stay in that local policy domain until it reaches to goal set,

- Efficient Inclusion test : there must be simple and efficient test to check if the given state is inside the given local control policy. It is an important policy feature because of practical reasons.

The sequential decomposition method has successfully been applied to many problems and it is known to be robust which respects to uncertainties and changes in the environment. For instance, in [33, 55] navigation and control of a fully actuated point robot is solved by using sequential composition. The sequential composition has also been applied to navigation and control of wheeled systems [31, 32, 56]. These studies show satisfying performance of the method for changing environment conditions. Additionally, sequential composition can be used for safe transition between different walking speed of a bipedal robot as in [57]. Furthermore, sequential composition is a good framework to describe high level tasks using linear temporal logic which enables multi-goal oriented motion planning [32, 58].

In the second part of thesis, we will describe a reactive footstep planning algorithm for dynamical locomotion of a simplified hopper via sequential composition.

Chapter 5

POSITION AWARE DEADBEAT CONTROLLER

In Section 2.1.3, we discussed possible modes of controlling SLIP locomotion. In summary, there are two common control parameters, the touchdown leg angle and the amount of change in the total mechanical energy. We described three different control modes, Leg Length Control (LLC), Leg Stiffness Control (LSC) and Two-Phase Stiffness Control (TPSC), classified according to way of controlling the total system energy. In this section, we will introduce different deadbeat controller[1] based on these controller modes to achieve a desired apex position and velocity relatively specified to the current apex state within a single step period. Hence, we will need a good predictor of the SLIP motion during a single step and as discussed in Chapter 3, we have several approximate stance maps with reasonable accuracy to design such controllers. Once again, we need to emphasize that existing deadbeat controllers in the literature are mostly designed for one period symmetric steps and concentrate on general characteristic of a gait (apex height and speed). However, since we are interested in motion planning of SLIP-like robots, our proposed deadbeat controllers have *position aware* structure to reach

[1] A deadbeat controller gives a discrete stepwise control actions at each apex to control SLIP locomotion.

a desired gait characteristic at a specific horizontal position without collision. In this chapter, we will mention the general form and implementation of these deadbeat controllers and possible applications.

5.1 General Form of Deadbeat Controllers

Existing studies on the control of the SLIP model are generally designed for a desired stable locomotion on flat surfaces and the resultant steps generally symmetric. These approaches mostly based on feedback control and an accurate modelling of SLIP is not critical. Desired motion can be obtained within a finite number of steps. On the other hand, dynamic gait planning over rough surfaces necessitates high performance controllers within single step action. Hence, our deadbeat controllers need an accurate motion predictor of SLIP.

For every SLIP step, the apex return map is a mapping from the current apex state, $\mathbf{b_a}[n]$, to the next apex state, $\mathbf{b_a}[n+1]$, by using the selected control action, $u[n]$:

$$\mathbf{b_a}[n+1] = F_a(\mathbf{b_a}[n], u[n]). \tag{5.1}$$

The apex return map is composition of the apex to touchdown map, $\mathbf{b_{td}}[n] := f_{a \mapsto td}(\mathbf{b_a}[n], u[n])$, the SLIP stance map, $\mathbf{b_{lo}}[n+1] := f_{td \mapsto lo}(\mathbf{b_{td}}[n], u[n])$ and the liftoff to apex map, $\mathbf{b_a}[n+1] := f_{lo \mapsto a}(\mathbf{b_{lo}}[n+1])$. Also, control input, $u[n]$, is a vector of control parameters defined as

$$u[n] := [\ q_{\theta_{td}}\ q_{r_{td}}\ q_{r_{lo}}\ k_c\ k_d\ ]_n.$$

Since we could not have the actual SLIP stance trajectory, we can use any appropriate stance approximation introduced in Chapter 3. Our deadbeat controllers will be based on approximate stance map of the SLIP model. In fact, the gravity corrected version of simple approximate stance map for two-phase variable stiffness in Section 3.3.1 has simple functional form and better accuracy

as compared to other alternatives. Also it is suitable for all three possible modes of control (LLC, LSC and TPSC). In this sense, for the rest of thesis, we generally use simple approximate map for two-phase variable stiffness with gravity correction and we will directly refer it as approximate stance map.

The general form of the deadbeat controllers depends on the inverse of the apex return map in (5.1) and we calculate the control action,$u[n]$, which takes SLIP from specified current apex state, $\mathbf{b_a}[n]$, to next apex state $\mathbf{b_a}[n+1]$. The control action is given by

$$u[n] = F_a^{-1}(\mathbf{b_a}[n], \mathbf{b_a}[n+1]). \tag{5.2}$$

For the time being, analytical form of the inverse of the apex return map is not available. Therefore, (5.2) can be solved as an optimization problem defined as finding $u[n] \in U$ to minimize $||\mathbf{b_a}[n+1] - F_a(\mathbf{b_a}[n], u[n])||$.

For the different modes of control, the optimization parameters will be different. For example, Leg Length Control (LLC) assumes fixed leg stiffness, k, during the entire stance phase. Also, the required change in the total mechanical energy is known since the current and next apex states are specified. Eventually, we have a relation between the touchdown and liftoff leg lengths as follows

$$E(\mathbf{b_a}[n+1]) - E(\mathbf{b_a}[n]) = 0.5k((l_0 - q_{r_{td}})^2 - (l_0 - q_{r_{lo}})^2). \tag{5.3}$$

Consequently, for LLC, we only need to determine two control parameters, the touchdown leg angle, $q_{\theta_{td}}$, and the touchdown leg length, $q_{r_{td}}$, to step from the current apex state, $\mathbf{b_a}[n]$, to the next apex state, $\mathbf{b_a}[n+1]$. The corresponding optimization problem is as follows

" Find $u[n] := [\ q_{\theta_{td}}\ q_{r_{td}}\ q_{r_{lo}}\ k\ k\ ] \in U$ to minimize $||\mathbf{b_a}[n+1] - F_a(\mathbf{b_a}[n], u[n])||$ "

where leg stiffness, k, is constant and the liftoff leg length, $q_{r_{lo}}$, is calculated using (5.3).

On the other side, Leg Stiffness Control (LSC) assumes tunable leg stiffness and it is kept constant during stance. In other words, the compression and decompression leg stiffness are the same and adjustable, $k_c = k_d$. Similarly, we may calculate the energy difference between the current and next apex states to determine the touchdown and liftoff leg lengths. The relation between additional energy to the system energy and leg lengths at the touchdown and liftoff instants is given in Table 5.1. Consequently, for LSC, we only need to determine two

Table 5.1: LSC - Energy and Leg Length Relation for a Chosen Leg Compliance

$\Delta E = E(\mathbf{b_a}(n+1)) - E(\mathbf{b_a}(n)),$

if $(\Delta E > 0)$

$q_{r_{td}} = l_0 - \sqrt{\frac{2\Delta E}{k_c}},$

$q_{r_{lo}} = l_0,$

else

$q_{r_{td}} = l_0,$

$q_{r_{lo}} = l_0 - \sqrt{\frac{2\Delta E}{k_d}},$

end

control parameters, the touchdown leg angle, $q_{\theta_{td}}$, and leg stiffness during stance, $k = k_c = k_d$ to reach from the current apex state, $\mathbf{b_a}[n]$, to the next apex state, $\mathbf{b_a}[n+1]$. The corresponding optimization problem to determine these control parameters can be defined as

" Find $u[n] := [\ q_{\theta_{td}}\ q_{r_{td}}\ q_{r_{lo}}\ k\ k\ ] \in U$ to minimize $||\mathbf{b_a}[n+1] - F_a(\mathbf{b_a}[n], u[n])||$ "

where the touchdown and liftoff leg lengths, $q_{r_{td}}$ and $q_{r_{lo}}$, are calculated using the relation in Table 5.1.

Finally, Two-Phase Stiffness Control (TPSC) is another possible mode of controlling SLIP locomotion and this control scheme assumes that the touchdown and liftoff leg lengths are equal to the rest length, i.e. $q_{r_{td}} = q_{r_{lo}} = l_0$. Also, we assume that we can instantaneously adjust leg stiffness at bottom instant. In a similar manner to previous control modes, the additional amount of change in the total mechanical energy of SLIP is calculated from the current and next apex

states. The compression leg stiffness, k_c, and the decompression leg stiffness, k_d, are determined by using the required change in the total system energy and the bottom leg compression as in Table 5.2.

Table 5.2: TPSC - Energy and Decompression Stiffness Relation for a Chosen Touchdown Leg Angle and Compression Leg Stiffness

$$\Delta E = E(\mathbf{b_a}[n+1]) - E(\mathbf{b_a}[n]),$$

$$q_{r_b} = L_b(\mathbf{b_a}[n], q_{\theta_{td}}, k_c),$$

$$k_d = k_c - (2\Delta E)/(l_0 - q_{r_b})^2.$$

where $L_b(\mathbf{b_a}[n], q_{\theta_{td}}, k_c)$ gives the approximate bottom leg length, q_{r_b}, for the current apex state, $\mathbf{b_a}(n)$ with chosen control inputs $q_{\theta_{td}}$ and k_c. Accordingly, for TPSC, we only need to determine two control parameters, the touchdown leg angle, $q_{\theta_{td}}$, and the compression leg stiffness during stance, k_c to achieve stepping from the current apex state, $\mathbf{b_a}[n]$, to the next apex state, $\mathbf{b_a}[n+1]$. The corresponding optimization problem to determine these control parameters is as follows

" Find $u[n] := [\ q_{\theta_{td}}\ l_0\ l_0\ k_c\ k_d\ ] \in U$ to minimize $||\mathbf{b_a}[n+1] - F_a(\mathbf{b_a}[n], u[n])||$ "

where the touchdown and liftoff leg lengths, $q_{r_{td}}$ and $q_{r_{lo}}$, are kept equal to the rest length of the system and the decompression leg stiffness can be calculated using the relation in Table 5.2.

In the following section, we will describe implementation steps of these control modes and our assumption to reduce computational complexity from two dimensional space search to two nested one dimensional space search.

5.2 Implementation of Deadbeat Controllers

In the previous section, we describe the general form of different control modes of SLIP. For every possible mode of control, there are two control parameters to be found to achieve the desired motion. The touchdown leg angle, $q_{\theta_{td}}$, is a common control parameter for all control approaches designed for the SLIP-like system. Fortunately, for flat surfaces the touchdown leg angle has a monotonic effect on the locomotion characteristic such that for a chosen energy adjustment for the next apex state the increasing touchdown leg angle decreases the apex velocity. Accordingly, we concentrate on rough terrain at discrete height levels with flat surfaces and assume that the toe of SLIP touches the ground at the same height for all possible allowed touchdown angles. In fact, it is possible to characterize a local controller used by a high level gait planner to guarantee that all apex states in its domain touches the same flat ground portion. Also, this monotonicity of touchdown leg angle is independent of other control parameter (touchdown leg angle for LLS, leg stiffness for LSC and compression leg stiffness for TPSC). Therefore, we may implement the two dimensional optimization problems defined in Section 5.1 as two nested one dimensional optimization problems. In the following paragraphs, we give the general implementation procedure for different modes of controlling SLIP steps and the cost functions used during optimization.

First of all, since we will divide our two dimensional optimization problem into two nested one dimensional optimizations, we have two different cost functions used by these one dimensional optimizations. One of these cost functions used by inner optimization is defined to determine that how well the liftoff state is obtained compared to the next apex state. We need to predict the liftoff and apex states which are given by

$$\hat{\mathbf{b}_{\mathbf{lo}}}[n+1] = f_{a \mapsto lo}(\mathbf{b_a}[n], u[n]), \tag{5.4}$$

$$\hat{\mathbf{b}_{\mathbf{a}}}[n+1] = f_{lo \mapsto a}(\mathbf{b_{lo}}[n+1]), \tag{5.5}$$

where $f_{a \mapsto lo}(\mathbf{b_a}[n], u[n])$ and $f_{lo \mapsto a}(\mathbf{b_{lo}}[n+1])$ are the apex to liftoff and the liftoff to apex maps, respectively and also the apex to liftoff map consists of the apex to touchdown and our approximate stance map. Accordingly, the cost function used by inner optimization is defined as

$$\begin{aligned} InnerCostFunction(\mathbf{b_a}[n+1], \hat{\mathbf{b_{lo}}}[n+1], \hat{\mathbf{b_a}}[n+1]) := \\ & w_1 F_{DT}(\mathbf{b_a}[n+1], \hat{\mathbf{b_{lo}}}[n+1])^2 \\ & + w_2(\pi_2 \circ \mathbf{b_a}[n+1] - \pi_2 \circ \hat{\mathbf{b_{lo}}}[n+1])^2 \\ & + w_3 D_{ss}(\mathbf{b_a}[n+1], \hat{\mathbf{b_a}}[n+1]), \end{aligned} \tag{5.6}$$

where $\hat{\mathbf{b_{lo}}}[n+1]$ and $\hat{\mathbf{b_a}}[n+1]$ are estimated liftoff and next apex states given by (5.4) and (5.5),respectively. w_1, w_2 and w_3 are weights of corresponding performance criteria. $F_{DT}(\mathbf{b_a}[n], \hat{\mathbf{b_{lo}}}[n])$ gives the distance of the predicted liftoff state, $\hat{\mathbf{b_{lo}}}[n]$, to flight trajectory ending at the next apex position, $\mathbf{b_a}[n]$ and $D_{ss}(\mathbf{b}_1, \mathbf{b}_2)$ calculates the Euclidian distance between the given SLIP states $\mathbf{b}_1$ and $\mathbf{b}_2$. Also, π is the projection operator.

Furthermore, another cost function for the outer optimization stage characterizes the accuracy of reaching the next apex state and it is given by

$$\begin{aligned} OuterCostFunction(\mathbf{b_a}[n+1], \hat{\mathbf{b_a}}[n+1]) := \\ & w_2(\pi_2 \circ \mathbf{b_a}[n+1] - \pi_2 \circ \hat{\mathbf{b_a}}[n+1])^2 \\ & + w_3 D_{ss}(\mathbf{b_a}[n+1], \hat{\mathbf{b_a}}[n+1]), \end{aligned} \tag{5.7}$$

where the parameters and functions are the same as above. The weights w_2 and w_3 are needed to be constant or have the same ratio as the used values for inner cost function. Thereby, for the ideal case if the outer cost function is equal to zero then the inner one has to be zero as well.

For Leg Length Control (LLC), we have two control parameters, the touchdown leg angle, $q_{\theta_{td}}$, and the touchdown leg length, $q_{r_{td}}$, determined during the

optimization problems. As mentioned above, if we assume that SLIP touches the same flat surface portion for all possible control inputs, then the effect of touchdown angle on gait properties becomes independent of touchdown leg length and has a monotonic behavior. Therefore, the original two dimensional optimization problem can be solved by using two nested one dimensional optimization as described in Table 5.3. Many of the parameters and functions are previously described , only $F_{(5.3)}(\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],q_{r_{td}})$ is a new function, see (5.3).

Table 5.3: LLC Deadbeat Controller Algorithm

Algorithm LLC_Deadbeat_Controller($\mathbf{b_a}[n],\ \mathbf{b_a}[n+1]$)

k : constant
$u[n] := [0\ l_0\ l_0\ k\ k]$
$u[n]$:=OuterOptimizationSearch($\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],\ u[n],\ U$)
{
do
$q_{r_{td}}$:= Selected by optimization search algorithm from allowed range
$q_{r_{lo}} := F_{(5.3)}(\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],\ q_{r_{td}})$
$u[n] := [0\ q_{r_{td}}\ q_{r_{lo}}\ (\pi_4 \circ u[n])\ (\pi_5 \circ u[n])]$
$q_{\theta_{td}}$:= InnerOptimizationSearch($\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],\ u[n],\ U$)
{
do
$q_{\theta_{td}}$:= Selected by optimization search algorithm from allowed range
$u[n] := [q_{\theta_{td}}\ (\pi_2 \circ u[n])\ (\pi_3 \circ u[n])\ (\pi_4 \circ u[n])\ (\pi_5 \circ u[n])]$
$\hat{\mathbf{b}}_{\mathbf{lo}}[n+1] := f_{a \mapsto lo}(\mathbf{b_a}[n], u[n])$
$\hat{\mathbf{b}}_{\mathbf{a}}[n+1] := f_{lo \mapsto a}(\hat{\mathbf{b}}_{\mathbf{lo}}[n+1])$
until Minimize $InnerCostFunction(\mathbf{b_a}[n+1],\ \hat{\mathbf{b}}_{\mathbf{lo}}[n+1],\ \hat{\mathbf{b}}_{\mathbf{lo}}[n+1])$
return $q_{\theta_{td}}$
}
$u[n] := [q_{\theta_{td}}\ (\pi_2 \circ u[n])\ (\pi_3 \circ u[n])\ (\pi_4 \circ u[n])\ (\pi_5 \circ u[n])]$
$\hat{\mathbf{b}}_{\mathbf{a}}[n+1] := F_a(\mathbf{b_a}[n], u[n])$
until Minimize $OuterCostFunction(\mathbf{b_a}[n+1],\ \hat{\mathbf{b}}_{\mathbf{a}}[n+1])$
return u[n]
}

return $u[n]$

Secondly, for Leg Stiffness Control -LSC, there are two control parameters, the touchdown leg angle, $q_{\theta_{td}}$, and leg stiffness during stance, $k = k_c = k_d$, which are to be found during optimization. Using the monotonic behavior of SLIP over flat surface, once again we will solve the original two dimensional

optimization problem as two nested one dimensional optimization problem as described in Table 5.4. The parameters and functions used in LSC Deadbeat Controller algorithm are mostly described above, $F_{T-5.1}(\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],\ k)$ is only a new function referring Table 5.1.

Table 5.4: LSC Deadbeat Controller Algorithm

Algorithm LSC_Deadbeat_Controller($\mathbf{b_a}[n]$, $\mathbf{b_a}[n+1]$)

$u[n] := [0\ l_0\ l_0\ 1\ 1]$
$u[n]$:=OuterOptimizationSearch($\mathbf{b_a}[n]$), $\mathbf{b_a}[n+1]$, $u[n]$, U)
{
  do
    k := Selected by optimization search algorithm from allowed range
    $[q_{r_{td}}\ q_{r_{lo}}] := F_{T-5.1}(\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],\ k)$
    $u[n] := [0\ q_{r_{td}}\ q_{r_{lo}}\ k\ k]$
    $q_{\theta_{td}}$:= InnerOptimizationSearch($\mathbf{b_a}[n]$, $\mathbf{b_a}[n+1]$, $u[n]$, U)
    {
      do
        $q_{\theta_{td}}$:= Selected by optimization search algorithm from allowed range
        $u[n] := [q_{\theta_{td}}\ (\pi_2 \circ u[n])\ (\pi_3 \circ u[n])\ (\pi_4 \circ u[n])\ (\pi_5 \circ u[n])]$
        $\hat{\mathbf{b}}_{\mathbf{lo}}[n+1] := f_{a \mapsto lo}(\mathbf{b_a}[n], u[n])$
        $\hat{\mathbf{b}}_{\mathbf{a}}[n+1] := f_{lo \mapsto a}(\hat{\mathbf{b}}_{\mathbf{lo}}[n+1])$
      until Minimize $InnerCostFunction(\mathbf{b_a}[n+1],\ \hat{\mathbf{b}}_{\mathbf{lo}}[n+1],\ \hat{\mathbf{b}}_{\mathbf{lo}}[n+1])$
      return $q_{\theta_{td}}$
    }
    $u[n] := [q_{\theta_{td}}\ (\pi_2 \circ u[n])\ (\pi_3 \circ u[n])\ (\pi_4 \circ u[n])\ (\pi_5 \circ u[n])]$
    $\hat{\mathbf{b}}_{\mathbf{a}}[n+1] := F_a(\mathbf{b_a}[n], u[n])$
  until Minimize $OuterCostFunction(\mathbf{b_a}[n+1],\ \hat{\mathbf{b}}_{\mathbf{a}}[n+1])$
  return u[n]
}

return $u[n]$

Finally, we have two control parameters, the touchdown leg angle, $q_{\theta_{td}}$, and the compression leg stiffness, k_c, for Two-Phase Stiffness Control (TPSC), and they are determined using the algorithm described in Table 5.5 to solve the corresponding optimization problem. Again, many of the parameters and functions are previously described, only $F_{T-5.2}(\mathbf{b_a}[n],\ \mathbf{b_a}[n+1],\ q_{\theta_{td}},\ k_c)$ is a new function, see Table 5.2.

Table 5.5: TPSC Deadbeat Controller Algorithm

1: **Algorithm TPSC_Deadbeat_Controller**($\mathbf{b_a}[n]$, $\mathbf{b_a}[n+1]$)

2: $q_{r_{td}} := l_0$
3: $q_{r_{lo}} := l_0$
3: $u[n] := [0\ q_{r_{td}}\ q_{r_{lo}}\ 1\ 1]$
4: $u[n]$:=OuterOptimizationSearch($\mathbf{b_a}[n]$, $\mathbf{b_a}[n+1]$, $u[n]$, U)
5: {
6: **do**
7: k_c := Selected by optimization search algorithm from allowed range
9: $u[n] := [0\ (\pi_2 \circ u[n])\ (\pi_3 \circ u[n])\ k_c\ 1]$
10: $u[n]$:= InnerOptimizationSearch($\mathbf{b_a}[n]$, $\mathbf{b_a}[n+1]$, $u[n]$, U)
11: {
12: **do**
13: $q_{\theta_{td}}$:= Selected by optimization search algorithm from allowed range
8: $k_d := F_{T-5.2}(\mathbf{b_a}[n], \mathbf{b_a}[n+1], q_{\theta_{td}}, k_c)$
14: $u[n] := [q_{\theta_{td}}\ (\pi_2 \circ u[n])\ (\pi_3 \circ u[n])\ (\pi_4 \circ u[n])\ k_d]$
15: $\hat{\mathbf{b}}_{\mathbf{lo}}[n+1] := f_{a \mapsto lo}(\mathbf{b_a}[n], u[n])$
16: $\hat{\mathbf{b}}_{\mathbf{a}}[n+1] := f_{lo \mapsto a}(\hat{\mathbf{b}}_{\mathbf{lo}}[n+1])$
17: **until** Minimize $InnerCostFunction(\mathbf{b_a}[n+1],\ \hat{\mathbf{b}}_{\mathbf{lo}}[n+1],\ \hat{\mathbf{b}}_{\mathbf{lo}}[n+1])$
18: **return** $u[n]$
19: }
21: $\hat{\mathbf{b}}_{\mathbf{a}}[n+1] := F_a(\mathbf{b_a}[n], u[n])$
22: **until** Minimize $OuterCostFunction(\mathbf{b_a}[n+1],\ \hat{\mathbf{b}}_{\mathbf{a}}[n+1])$
23: **return** u[n]
24: }

25: **return** $u[n]$

In this section, we mentioned implementation of the possible modes of controlling the SLIP in a much more efficient way by using the monotonic effect of touchdown leg angle on gait properties on flat surfaces. In the following section, we will discuss some possible application of these deadbeat controllers for the gait planning of the SLIP-like systems.

5.3 Applications

There are three possible modes of deadbeat controller to achieve stepping from the current apex state to the next apex state, and in the previous sections, we

mentioned the general form and implementations of these controllers. During our studies we assume that the ground profile is composed of flat ground portions at different levels. In this section, we will consider this type of terrains and traverse over them by reaching previously determined a sequence of apex states. At this point of the thesis, we assume availability of an offline high-level gait planner such that for a given start and goal states with known ground profile it gives a sequence of apex states to reach desired goal location. One of the critical property of this planner is that locomotion planning is performed while considering the limitations on control input and realizability of each SLIP steps. As mentioned previously, our deadbeat controllers are based on approximate stance map of the SLIP model. We generally use simple approximate map for the two-phase variable stiffness control described in Section 3.3.1, and we will directly refer to it as approximate stance map.

Firstly, existing controllers for SLIP are generally designed for symmetric steps over flat surfaces, hence we will begin with symmetric locomotion over flat surfaces. In Figures 5.1, 5.2 and 5.3, we illustrate performance of LLC, LSC and TPSC for a given set of symmetric apex states. These figures also include the "ground truth"[2] for these possible modes of control which are obtained by using the numerically obtained apex return map. From the existing studies and our observations we know that symmetric steps are the most accurately predicted steps by approximate stance map, eventually the performance of the controllers are very close to "ground truth". Also, for a given flat surface we easily know the height of the touchdown point on the ground which is equal to height of flat ground, but for a general ground profile it is not the case and we need to compute that information for each possible control iteration and it also increases the computation complexity drastically. At that point we focus on the worst case and we implement a general algorithm for any ground profile to find that height.

[2]The *ground truth* for each control modes is obtained using the actual apex return map that is numerical solution of the SLIP dynamics.

In fact, for ground profiles with flat ground portion at discrete levels we may use much more efficient algorithms but we want to show the importance of knowing the height of the ground portion where the spring-mass hopper lands. Hence, design of local controllers for selected flat ground portions may be an efficient approach for footstep planning of the SLIP-like systems.

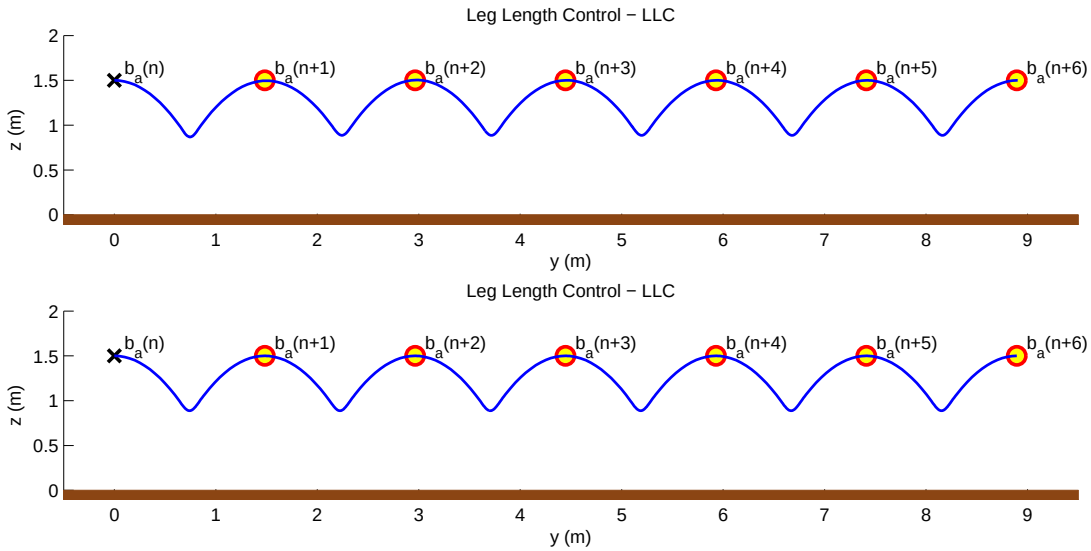

Figure 5.1: Leg Length Control - LLC for Symmetric Steps over a flat surface. *Upper*: The apex return map used by LLC is based on our approximate stance map. Number of iterations during optimization is around 191 and the computation time is around 0.31 secs for each steps. *Lower*:"Ground Truth" for LLC. The apex return map used by LLC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 235 and the computation time is around 6.1 secs for each steps. If we also compute the height of the flat ground portion where the SLIP lands for every iteration, the computation time increases to 5.1 secs and 12.4 secs for the upper and lower cases, respectively.

Another explanatory example is usage of nonsymmetric steps over flat surfaces. We illustrate in Figures 5.4, 5.5 and 5.6 that how well nonsymmetric steps can be generated by using the proposed deadbeat controllers. A set of apex states during the locomotion is assumed to be given by a high-level planner. As seen from the figures, deadbeat controllers based on the approximate stance map improves the performance for nonsymmetric steps as well and they result with similar trajectories as numerically solved "ground truth". As seen from the simulation times, computation time dramatically increases for a known

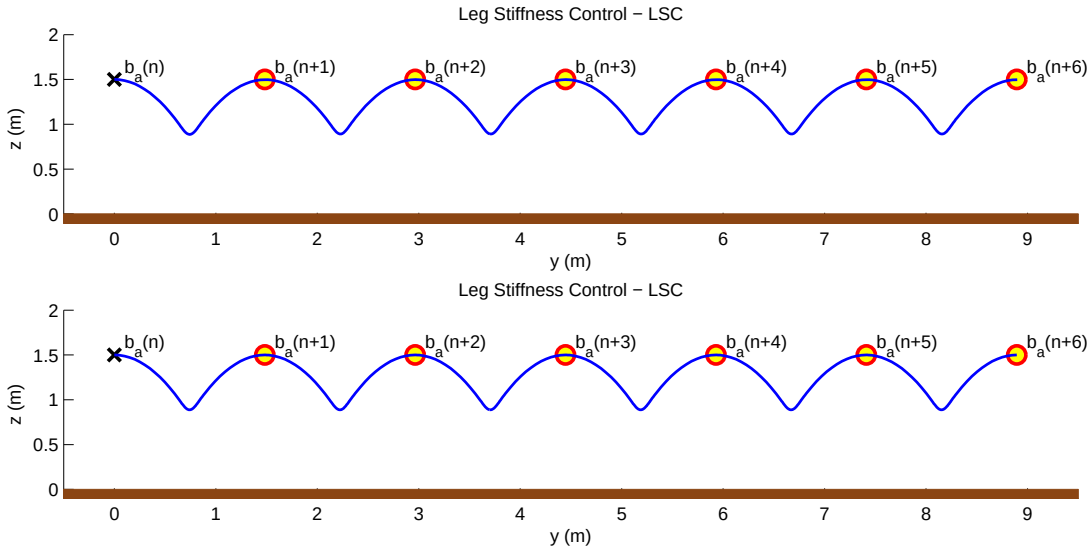

Figure 5.2: Leg Stiffness Control - LSC for Symmetric Steps over a flat surface. *Upper*: The apex return map used by LSC is based on our approximate stance map. Number of iterations during optimization is around 238 and the computation time is approximately 0.32 secs for each steps. *Lower*:"Ground Truth" for LSC. The apex return map used by LSC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 192 and the computation time is around 5.2 secs for each steps. If we also compute the height of the flat ground portion where the SLIP lands for every iteration, the computation time increases to 7.1 secs and 10.6 secs for the upper and lower cases, respectively.

general ground profile since for each iteration during the execution of deadbeat controller, we need to calculate height of the touchdown point for each possible control iterations. This increases the cost of computation. Therefore, as will be discussed in the next chapter, a local controller need to be designed for a flat ground portion with known ground height for real-time applications.

Moreover, another interesting application of LLC,LSC and TPSC is controlling SLIP locomotion over stair like grounds as illustrated in Figures 5.7, 5.8 and 5.9. Once again, we assume that a high level locomotion planner gives a sequence of apex states to traverse over a stair and our deadbeat controllers aim to achieve successful stepping between these apex states. In this case, we couldn't know height of the touchdown point without knowing the current apex state and selected control action. Accordingly, the computation times disables this kind off

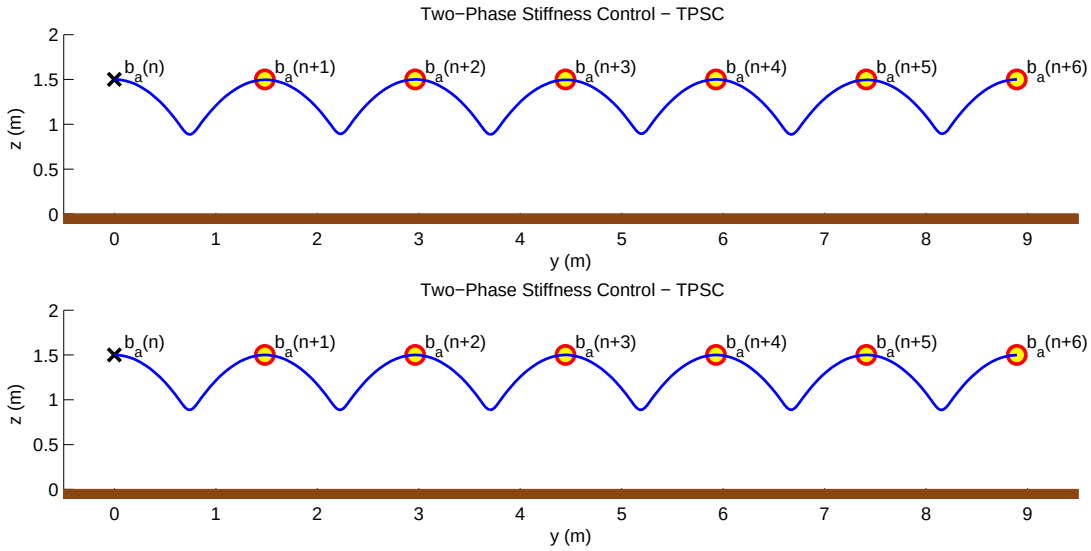

Figure 5.3: Two-Phase Stiffness Control - TPSC for Symmetric Steps over a flat surface. *Upper*: The apex return map used by TPSC is based on our approximate stance map. Number of iterations during optimization is around 215 and the computation time is approximately 0.41 secs for each steps. *Lower*:"Ground Truth" for TPSC. The apex return map used by TPSC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 192 and the computation time is around 8.2 secs for each steps. If we also compute the height of the flat ground portion where the SLIP lands for every iteration, the computation time increases to 5.75 secs and 13 secs for the upper and lower cases, respectively.

approach for real time application. One possible solution is usage of much more computationally efficient algorithms to determine height of toe at the touchdown instant. However, a much more efficient solution is the characterization of local controllers for a specific flat ground portion. For instance, in the next chapter, we propose to design local controllers for a flat ground portion with known ground height to be applicable for real-time applications.

5.4 Discussion

In the previous section, we gave several examples of the applications of LLC, LSC and TPSC for controlling SLIP locomotion by using a set apex states . These apex states are assumed to be generated by a high-level planner such that this

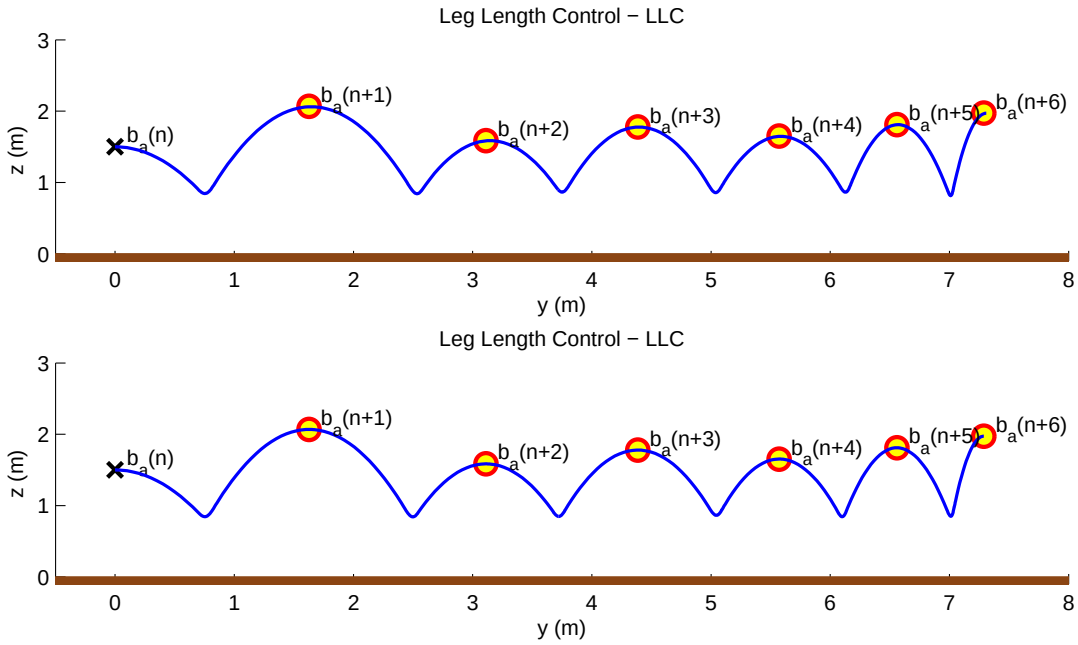

Figure 5.4: Leg Length Control - LLC for Nonsymmetric Steps over a flat surface. *Upper*: The apex return map used by LLC is based on our approximate stance map. Number of iterations during optimization is around 224 and the computation time is around 0.34 secs for each steps. *Lower*:"Ground Truth" for LLC. The apex return map used by LLC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 213 and the computation time is around 5.6 secs for each steps. If we also compute the height of the flat ground portion where the SLIP lands for every iteration, the computation time increases to 7.1 secs and 10.8 secs for the upper and lower cases, respectively.

planer considers limitation of control inputs and it also satisfies that starting from any apex state in the domain of a local controller the SLIP touches the same flat ground segment with all possible control action. For the time being, we do not have this kind of local controllers and high level planner. These are the novel extensions and open research problems related to footstep planning of SLIP-like systems.

One of the critical property of the SLIP model used for deadbeat controller design, which enables us to deal with two nested one dimensional optimization rather than two dimensional one, is monotonicity with respect to the touchdown leg angle over flat surfaces. Another interesting problem is the design of local control policies for the SLIP-like system for a selected flat ground portion. For a

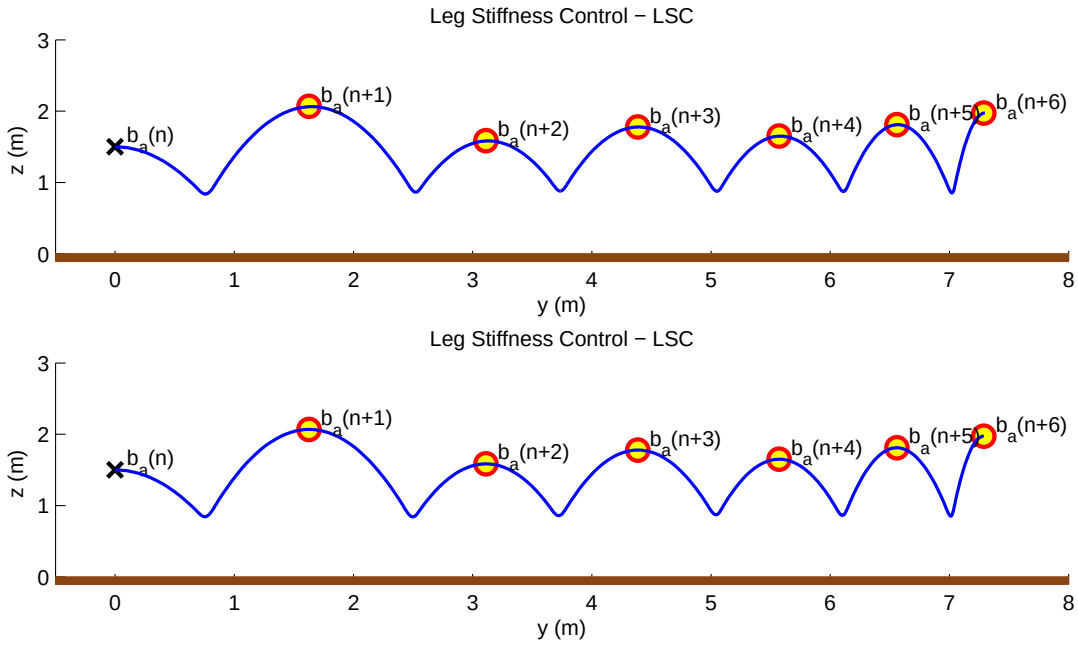

Figure 5.5: Leg Stiffness Control - LSC for Nonymmetric Steps over a flat surface. *Upper*: The apex return map used by LSC is based on our approximate stance map. Number of iterations during optimization is around 258 and the computation time is approximately 0.35 secs for each steps. *Lower*:"Ground Truth" for LSC. The apex return map used by LSC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 215 and the computation time is around 5.6 secs for each steps. If we also compute the height of the flat ground portion where the SLIP lands for every iteration, the computation time increases to 8.3 secs and 13.3 secs for the upper and lower cases, respectively.

given control input sets, determination of the domain and goal regions of these kind of local policies are significant for footstep planning over rough surfaces with flat surfaces at different levels. Also, if we design local controllers for a specific portion of the ground, we easily calculate the touchdown states for a chosen control action with is really important for real-time applications. Similar approaches is used in the next chapter for a simplified hopper to design reactive footstep algorithm and it is an introductory idea for a possible longer term research on reactive footstep planning of the SLIP-like monopods and bipeds.

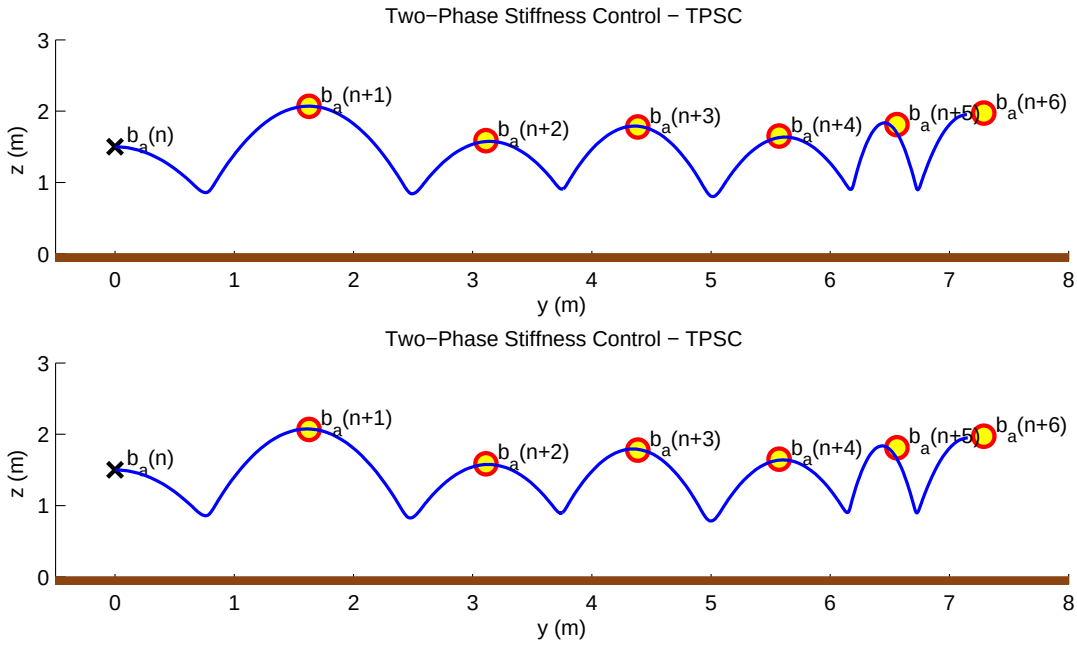

Figure 5.6: Two-Phase Stiffness Control - TPSC for Nonsymmetric Steps over a flat surface. *Upper*: The apex return map used by TPSC is based on our approximate stance map. Number of iterations during optimization is around 282 and the computation time is approximately 0.56 secs for each steps. *Lower*:"Ground Truth" for TPSC. The apex return map used by TPSC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 253 and the computation time is around 8.2 secs for each steps. If we also compute the height of the flat ground portion where the SLIP lands for every iteration, the computation time increases to 8.6 secs and 18.4 secs for the upper and lower cases, respectively.

Orthogonally, the deadbeat controllers proposed here are based on approximate stance map. In reality, there are several limitation on the prediction performance of the SLIP stance approximation and to increase the estimation performance of approximation we may also force deadbeat controller to select control inputs which results with steps with properties close to the assumption for the derivation of the approximate stance maps. Therefore, the closeness of the gait properties to approximation assumption may be included as a part of optimization cost functions. At this moment, we don't need this kind of modification for cost function but it will result in much more reliable motions.

Furthermore, we designed three different deadbeat controllers, LLC, LSC and TPSC, and each of them has good performances compared to the "ground truth"

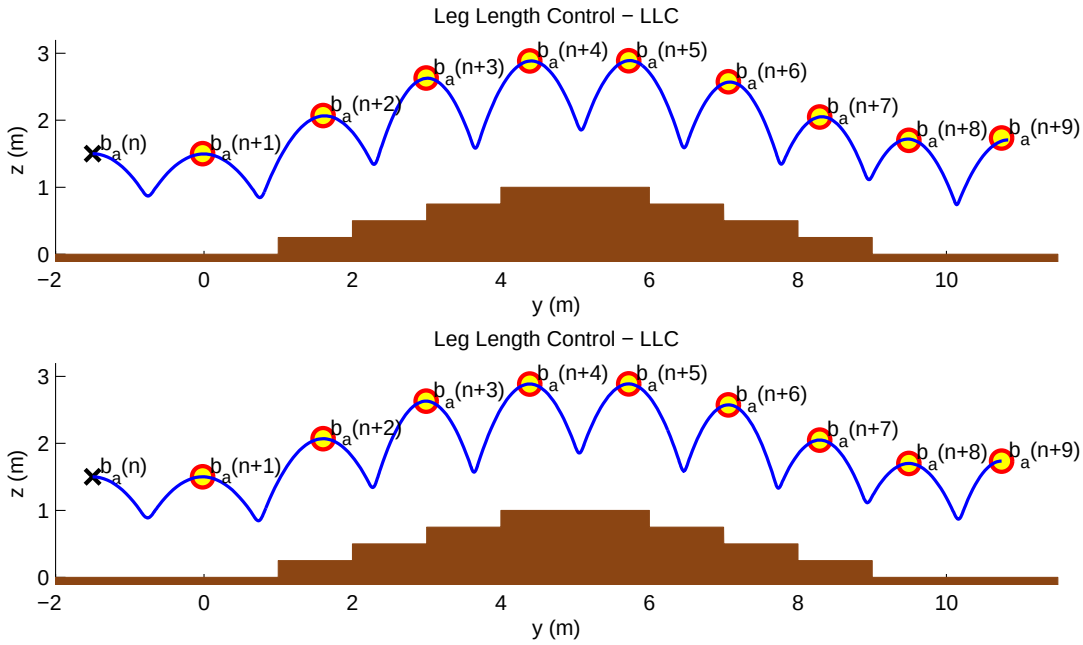

Figure 5.7: Leg Length Control - LLC for Nonsymmetric Steps over a flat surface. *Upper*: The apex return map used by LLC is based on our approximate stance map. Number of iterations during optimization is around 202 and the computation time is around 6.8 secs for each steps. *Lower*:"Ground Truth" for LLC. The apex return map used by LLC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 220 and the computation time is around 12.8 secs for each steps.

obtained by numerically computed apex return map of the SLIP. If we compare these three controller modes, LEG Stiffness Control - LSC seems to the most accurate and computationally efficient one. For instance, LLC sometimes select control inputs which results with large amount of compression during stance but the approximate stance map assumes small leg compression. On the other hand, TPSC has worse computational efficiency since it needs to know bottom leg length to determine the decompression leg stiffness. Also, for LLC and LSC we guarantee that the exact amount of energy is added to or taken out from system, but TPSC uses the estimated bottom leg length to adjust system energy which is not necessarily to be equal to the exact amount. This is another problematic issue which is related with TPSC.

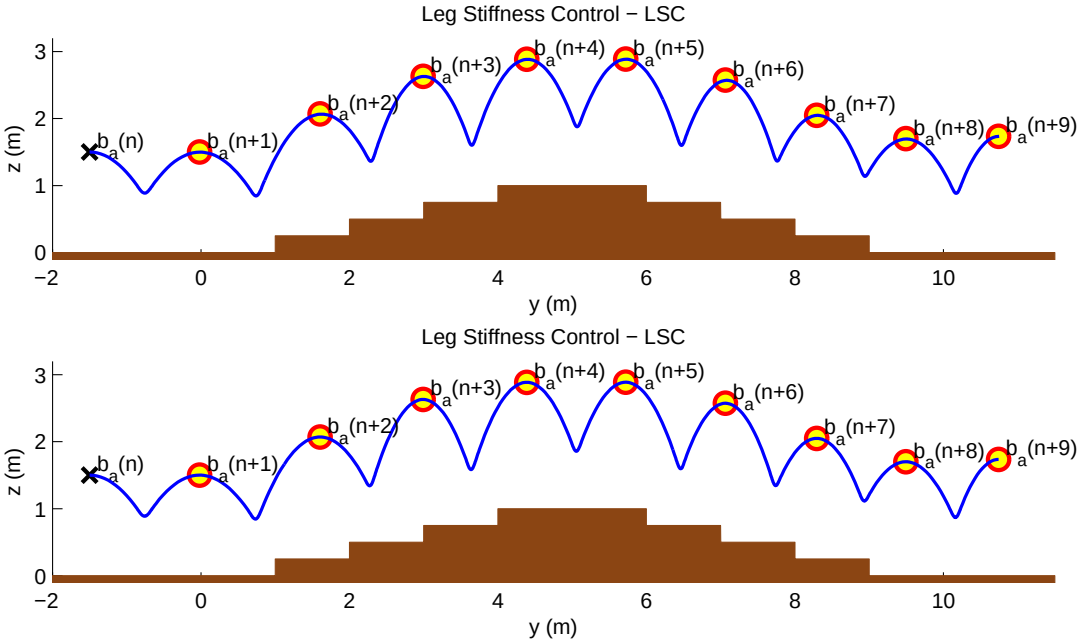

Figure 5.8: Leg Stiffness Control - LSC for Nonymmetric Steps over a flat surface. *Upper*: The apex return map used by LSC is based on our approximate stance map. Number of iterations during optimization is around 258 and the computation time is approximately 10.6 secs for each steps. *Lower*:"Ground Truth" for LSC. The apex return map used by LSC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 210 and the computation time is around 13.7 secs for each steps.

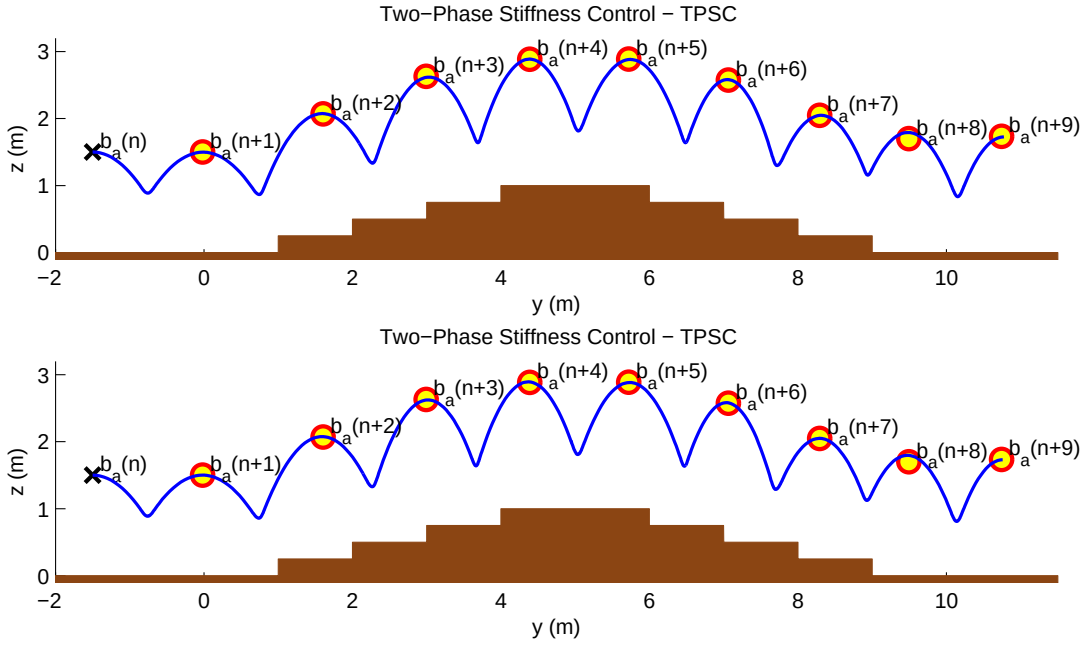

Figure 5.9: Two-Phase Stiffness Control - TPSC for Nonsymmetric Steps over a flat surface. *Upper*: The apex return map used by TPSC is based on our approximate stance map. Number of iterations during optimization is around 234 and the computation time is approximately 8.0 secs for each steps. *Lower*:"Ground Truth" for TPSC. The apex return map used by TPSC is based on numeric solution of the SLIP dynamics. Number of iterations during optimization is around 223 and the computation time is around 18.7 secs for each steps.

Chapter 6

REACTIVE FOOTSTEP PLANNING

The second part of this thesis, reactive footstep planning, will be mainly introduced within this chapter. To make clear the advantage of reactivity, we need to define it. In general, reactive and adaptive behaviors sounds the same and they are commonly confused, but essentially they are different. Adaptive acting is a type of behavior to adjust the current behavior according to several observation from the environment and results of current action. On the other hand, reactive behavior is tendency to show a response in all situations. Accordingly, proposed algorithm guaranties to stay inside the global domain of local policies since the sequential composition method has conditionally invariance property (see Section 4.3) and gives a reasonable control action to reach the goal state for all states insides the global domain. This chapter composes of general planning framework for any type of hopper over an uneven surface as in Fig. 6.1 and illustrative application on a simplified hopper with simulation results.

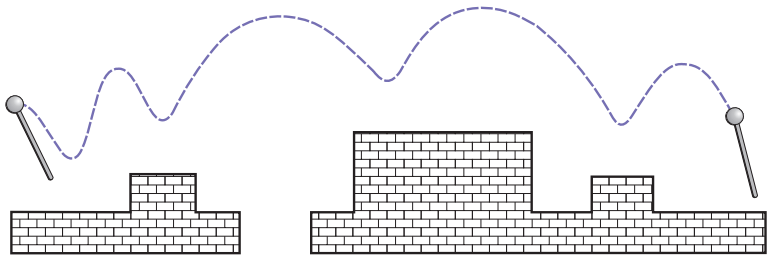

Figure 6.1: A spring-mass hopper running over rough terrain.

6.1 Planning Framework

6.1.1 A Generic Hopper Model

Running trajectories for all one or two legged systems exhibit a common structure: They alternatingly go through *flight* and *stance* phases, separated by *touchdown* and *liftoff* events as the foot comes into contact and leaves the ground, respectively. Most specific examples in the literature also find it useful to define an *apex* event associated with the highest point of the center of mass (COM) during flight, whose height and forward velocity are often used as a representative state vector for the subsequent stride. In this section, we will make as few assumptions as possible about the underlying legged system for such structures in order to ensure general applicability of our reactive planning framework.

Throughout this thesis, we assume that a planar one-legged hopper is running on a piecewise flat ground (such as the example shown in Fig. 6.1), possibly with a number of "holes" on which no foot placement is possible. During flight, we assume that the robot COM follows a ballistic trajectory, whereas during stance, its dynamics are determined by its leg morphology and control which we leave unspecified. We also assume that gait control is achieved with per-step control inputs selected at each apex (but possibly realized throughout the entirety of the following flight and stance phases), allowing independent, possibly limited control of all three degrees of freedom for the next apex. This framework, for which a single stride is illustrated in Fig. 6.2, is consistent with most planar

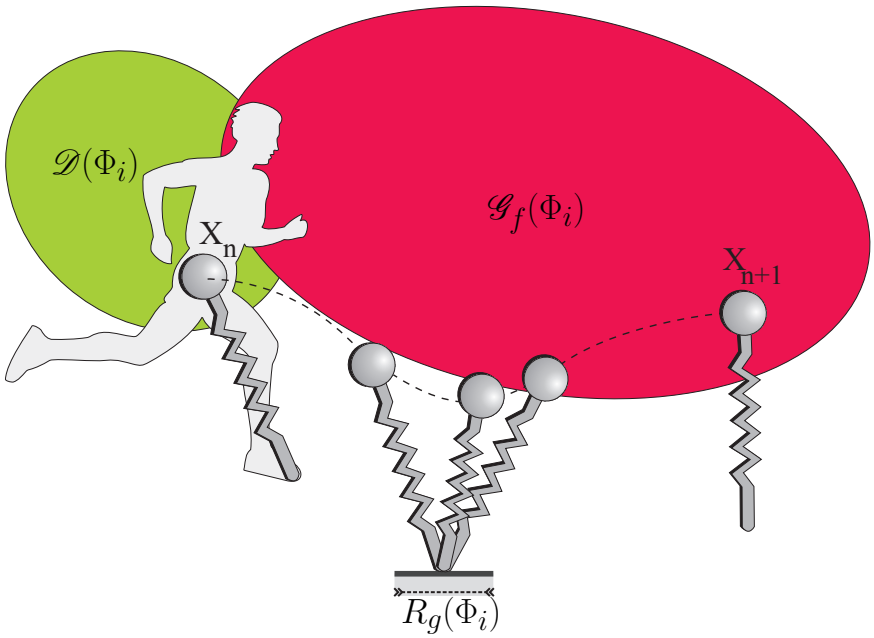

Figure 6.2: An illustration of ground support $R_g(\Phi_i)$, policy domain $\mathscr{D}(\Phi_i)$, and feasible goal $\mathscr{G}_f(\Phi_1)$ regions for the spring mass hopper.

running robot morphologies in the literature ranging from the SLIP model to more complex, multi-jointed leg designs.

We associate with each ground segment, one or more single-step "local" families of control policies Φ_i that use that segment for their foothold during stance, while using control inputs from a constrained set $U(\Phi_i)$ to take the robot to an associated set of possible apex states. Our planning algorithm seeks to find a particular reactive sequencing of these policies to ensure that the robot is driven to a desired goal state from as large a set of initial conditions as possible.

In the spirit of sequential composition, we associate with each family of policies Φ_i, a *domain* $\mathscr{D}(\Phi_i)$, including those apex states from which the corresponding ground segment $R_g(\Phi_i)$ is reachable, as well as a *feasible goal set* $\mathscr{G}_f(\Phi_i)$ including only apex states that are achievable from every state within the domain within a single stride using allowable control inputs. In the sequel, we will use $X_n := \{y_n,\ z_n,\ \dot{y}_n,\ 0\}$ to denote the state of the hopper at the n^{th} apex event,

Table 6.1: Notation associated with reactive footstep planning used throughout the thesis

State Parameters	
y, z	Horizontal and vertical system positions
$\dot{y}, \dot{z}$	Horizontal and vertical system velocities
$X_n = [y, \dot{y}, z, \dot{z}]^T$	State vector representation
$\pi_i \circ X_n$	Gives the i^{th} state variable, i.e. $\pi_3 \circ X_n = z$
Policy Parameters	
Φ_i	A local policy
$\hat{\Phi}_i$	An initiated local policy, i.e. it has a unique goal state
$\mathscr{D}(\Phi_i)$	Domain region of a local policy Φ_i
$\mathscr{G}_f(\Phi_i)$	Feasible goal region of a local policy Φ_i
$\mathscr{G}_s(\hat{\Phi}_i)$	Goal state of an initiated local policy$\hat{\Phi}_i$
u, U	Control input and control input set
$R_E(\Phi_i)$	The apex energy range of a local policy Φ_i
$R_V(\Phi_i)$	The apex velocity range of a local policy Φ_i
$R_g(\Phi_i)$	The portion of ground that a local policy Φ_i can use (The region of attraction of local policy Φ_i on the ground)
Useful Functions	
$y_{f,td}(X_n, u)$	Gives the horizontal touchdown position of system
$F_a(X_n, u)$	Apex return map, gives the next apex state according to the current state and selected control input
$E(X_n)$	Total mechanical energy of a system state
$T_D(X_n, \Phi_i)$	Flight Trajectory from X_n to touchdown instance over a flat surface at policy height
$T_A(X_{n+1}, \Phi_i)$	Flight Trajectory from corresponding liftoff instance to X_{n+1} over a flat surface at policy height
Several Definitions	
$\mathcal{FS}$	Free Configuration Space of Apex States

and $F_a(X_n, u)$ to denote the apex return map under a specific control input u. Formal definitions of the domain and feasible goal sets hence take the form

$$\mathscr{D}(\Phi_i) := \{X_n \mid \dot{y}_n \in R_V(\Phi_i),\ E_n \in R_E(\Phi_i),$$
$$\forall u \in U(\Phi_i).\ y_{f,td}(X_n, u) \in R_g(\Phi_i),$$
$$T_D(X_n, \Phi_i) \subset \mathcal{FS}\ \} \tag{6.1}$$
$$\mathscr{G}_f(\Phi_i) := \{X_{n+1} \mid \forall X_n \in \mathscr{D}(\Phi_i),\ \exists u \in U(\Phi_i).$$
$$X_{n+1} = F_a(X_n, u),$$
$$T_A(X_{n+1}, \Phi_i) \subset \mathcal{FS}\}\ , \tag{6.2}$$

where the sets $R_V(\Phi_i)$ and $R_E(\Phi_i)$ are allowed ranges for the velocity and energy of the initial apex state, $U(\Phi_i)$ is the allowable set of control inputs, and $R_g(\Phi_i)$ denotes the ground segment associated with the policy on which the hopper will land as shown in Fig. 6.2. In order to prevent collisions with the ground we also require descent and ascent trajectories, denoted with $T_D(X_n, \Phi_i)$ and $T_A(X_{n+1}, \Phi_i)$, to be contained in free space $\mathcal{FS}$.

Intuitively, $\mathscr{D}(\Phi_i)$ captures apex states having energy and forward velocity values in the associated ranges $R_V(\Phi_i)$ and $R_E(\Phi_i)$ from where the hopper can reach the corresponding ground segment using available control inputs $u \in U(\Phi_i)$. In contrast, the feasible goal region $\mathscr{G}_f(\Phi_i)$, represents all apex states that are guaranteed to be reachable from the entire domain using whatever control input is necessary from within the allowable set $U(\Phi_i)$.

6.1.2 Reactive Footstep Planning

As described above, computation of the domain and feasible goal regions depends on the system dynamics. Nevertheless, once computed, they present a very convenient abstract interface between planning and control since the above construction ensures by design, the existence of a single-step controller that can take any apex state inside the domain to any subsequent apex state within the

feasible goal region. In this section, we describe how to automatically construct a reactive, provably correct hybrid footstep controller that uses a given set of policies Φ_i.

The Prepares Relation

The sequential composition formalism introduced in [30] defines a relation between local controller policies that captures whether their sequencing is feasible or not. Policy definitions in their domain include only a single goal point, whose inclusion in the domain of another policy is sufficient to ensure the validity of sequencing. Our formulation of the goal set is more general in the sense that the feasible goal set $\mathscr{G}_f(\Phi_i)$ includes an entire range of possible goal points that can be chosen to yield a particular policy instance. Consequently, we define a more general relation *can prepare*, indicating the availability of a goal choice that can guarantee proper sequencing.

Definition 2. *A policy* Φ_i can prepare *another policy* Φ_j*, denoted by* $\Phi_i \succeq_c \Phi_j$*, if and only if the following condition holds,*

$$\mathscr{G}_f(\Phi_i) \cap \mathscr{D}(\Phi_j) \neq \emptyset \, .$$

This freedom in choice for the goal point associated with a policy allows the planner to consider optimality or safety criteria to increase the efficiency and robustness of the final reactive controller. Much like the original sequential composition algorithm, this relation induces a directed, possibly cyclic *can prepare graph* $\mathcal{G} := \{\Phi_i, \succeq_c\}$ between all policies. Before we proceed with the description of our planning algorithm, we will find the following definition useful.

Definition 3. *A* policy instance $\hat{\Phi}_i(X_g)$ *is a controller that will take the hopper from an apex state* $X_n \in \mathscr{D}(\Phi_i)$ *and bring it to a specific, feasible goal point* $X_{n+1} = X_g \in \mathscr{G}_f(\Phi_i)$ *using an allowable control input* $u \in U(\Phi_i)$ *and stepping once on the ground range* $R_g(\Phi_i)$ *associated with the policy.*

Planning by Prioritizing Policies

The formulation of the "can prepare graph" above captures all relevant sequencing constraints between different control policies. However, a robot running across rough terrain must still decide at every step which one of these policies will be used to determine proper control inputs for the next stride. The original sequential composition method divides this problem into two stages. First, the prepares graph is converted into a total order whose top element is chosen as a policy that can take the robot to the desired global goal. The resulting explicit prioritization of policies is then used at runtime to determine which policy should be used from among those whose domains cover the measured robot state. Even though different alternatives such as sequence-based and automata-based planners are possible [31], we adopt the order-based method, adapted to deal with larger, non-point goal sets as well as the discrete nature of our system.

Suppose a global goal is supplied to the planner in the form of a desired apex state X_g. Our algorithm starts by choosing goal policies Φ_j such that $X_g \in \mathscr{G}_f(\Phi_j)$ and instantiates them for the specific desired goal as $\hat{\Phi}_j(X_g)$. The algorithm then proceeds by backchaining on $\mathcal{G}$, incrementally building a total order of instantiated policies until all policy nodes in the graph are traversed. The instantiation of each policy chooses a single goal point which is both in its feasible goal set as well as the domain of the policy that it prepares, using a heuristic cost function that takes into account appropriate safety and efficiency criteria to determine the best candidate from among available goal alternatives.

Table 6.2 gives the detailed planning algorithm that yields the final policy ordering to be used for reactive control. Note that for each instantiated policy, the algorithm computes a specific goal point $\mathscr{G}_s(\hat{\Phi}_j)$, and a priority $\mathscr{P}(\hat{\Phi}_j)$ according to which the final total order will be obtained. The procedure $CostFunction(\Phi_j, \hat{\Phi}_i)$ is expected to return both the best candidate goal point

Table 6.2: Order Based Policy Priority Planner

1: **Algorithm Order_Based_Policy_Planner**(X_g, $\mathcal{G}$)

2: $PolicyFIFO := \emptyset$
3: $\mathcal{GP} := FindGoalPolicies(X_g, \mathcal{G})$
4: for all $\Phi_j \in$ GP do
5: $\mathscr{G}_s(\hat{\Phi}_j) := X_g$
6: $\mathscr{P}(\hat{\Phi}_j) := 0$
7: $Push(PolicyFIFO, \hat{\Phi}_j)$
8: endfor
9: while not(isempty($PolicyFIFO$))
10: $\hat{\Phi}_i := Pop(PolicyFIFO)$
11: $\mathcal{PP} := FindPreparingPolicies(\hat{\Phi}_i, \mathcal{G})$
12: for all $\Phi_j \in \mathcal{PP}$ do
13: $[C_T, X_{g_T}] = CostFunction(\Phi_j, \hat{\Phi}_i)$
14: if $(C_T + \mathscr{P}(\hat{\Phi}_i)) < \mathscr{P}(\Phi_j)$ then
15: $\mathscr{G}_s(\hat{\Phi}_j) := X_{g_T}$
16: $\mathscr{P}(\hat{\Phi}_j) := C_T + \mathscr{P}(\hat{\Phi}_i)$
17: $Push(PolicyFIFO, \hat{\Phi}_j)$
20: endif
21: endfor
22: endwhile

23: **return** $\{\hat{\Phi}_{1,N}\}$

for policy Φ_j and a cost related to how effectively it prepares the instantiated policy $\hat{\Phi}_i$.

6.2 Reactive Foorstep Planning of a Simplified Hopper: Ball Hopper

In this section, we will present a simple example of reactive footstep planning algorithm for a simplified hopper, mainly for a ball hopper, which we also refer to a controllable ball.

6.2.1 Simplified Hopper Model

Efficient, preferably analytic characterization of the domain and particularly the feasible goal regions for even the relatively well-studied SLIP model is a challenging, currently unsolved problem. Despite recent availability of very effective analytical maps for the stance dynamics of this model, numerical solutions are still needed for the characterization of the feasible goal region. Since our primary emphasis in this paper is the reactive planning framework, we will investigate the efficacy of our proposed method in a hopper model whose simplified dynamics will structurally mimic SLIP behavior while admitting analytical characterization of the feasible goal region $\mathscr{G}_f(\Phi_i)$ for individual policies. This simplified model can be best described as a *controllable ball*, which will summarize stance dynamics of the SLIP model with a "bounce" from a virtual surface elevated by a height equal to the SLIP spring rest length, changing the liftoff velocity and position of the body center of mass in a controllable fashion. Fig. 6.3 illustrates this idea for the scenario previously depicted in Fig. 1.5.

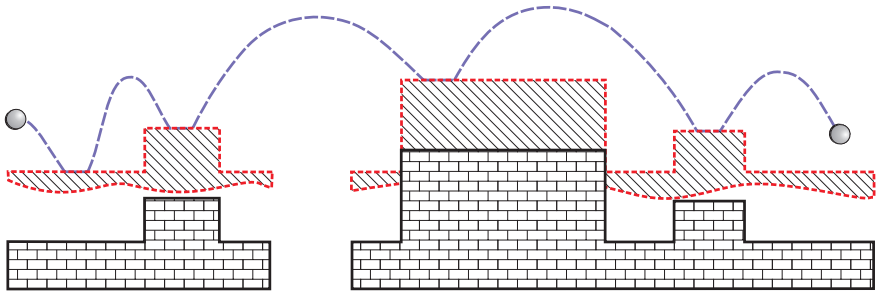

Figure 6.3: An illustration of locomotion trajectories for the simplified "controllable ball" hopper model together with the "virtual ground" constructed from the scenario depicted in Fig. 1.5.

System Dynamics

During flight, the simplified hopper follows an uncontrollable ballistic trajectory, whose dynamics are given by

$$\dot{X} = \begin{bmatrix} \dot{y} & \dot{z} & \ddot{y} & \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{y} & \dot{z} & 0 & -g \end{bmatrix}.$$

Flight dynamics are particularly important in computing both the domain and feasible goal regions associated with a given ground segment.

The stance phase for the SLIP model is governed by the compression and decompression of the leg spring until the liftoff event. For the simplified hopper, we capture this behavior by using a direct, "instantaneous" touchdown to liftoff map, controlled by a horizontal shift Δy, and adjustments θ and k on the angle and normal magnitude of the liftoff velocity with respect to a symmetric gait and an oblique "virtual bounce surface". These control parameters have very close correspondence to control parameters that are frequently used for the SLIP model and associated robot morphologies. The following list details these correspondences.

- The liftoff velocity magnitude gain, k, roughly corresponds to the energy control for the SLIP model through the decompression and compression stiffness ratio k_d/k_c.
- The liftoff velocity angle adjustment closely corresponds to the touchdown leg angle for the SLIP model with respect to the neutral angle, $q_{\theta_t} - q_{\theta_n}$.
- The position shifting control, Δy, corresponds to the average stiffness of the SLIP leg, which can increase or decrease the positional span of the stance phase. In the SLIP model, this displacement nonlinearly depends on other control parameters, but can be independently chosen by adjusting k_c.

For example, a symmetric step can be obtained in the SLIP model by choosing $k_c = k_d$ and $q_{\theta_t} = q_{\theta_n}$ with the horizontal liftoff position independently adjustable through the common stiffness value. For the simplified hopper model, this corresponds to choosing $\theta = 0$ and $k = 1$ with Δy independently adjusting the horizontal displacement during stance.

The resulting stance map for the simplified hopper model is hence given by

$$X_{lo} = AX_{td} + B \tag{6.3}$$

where we define

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1-(1+k)\sin^2(\theta) & 0.5(1+k)\sin(2\theta) \\ 0 & 0 & 0.5(1+k)\sin(2\theta) & 1-(1+k)\cos^2(\theta) \end{bmatrix}$$

$$B = \begin{bmatrix} \Delta y \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Gait Control for the Simplified Hopper

The gait controller associated with each instantiated policy $\hat{\Phi}_i$ is responsible from finding control inputs necessary to bring the robot from any state $X_n \in \mathscr{D}(\hat{\Phi}_i)$ to the selected goal point X_g of $\hat{\Phi}_i$. To this end, we use a simple deadbeat controller for the simplified hopper, similar to those frequently used for the SLIP model, yielding reactive control inputs computed as $u = F_a^{-1}(X_n, X_g)$.

Derivation of the Domain Region

Before we proceed with an analytical representation of the domain region associated with a ground segment $R_g(\Phi_i)$, we add an additional constraint on the minimum apex height h_{min} to ensure that the leg can always clear the ground for protraction. This leads to a redefinition of the domain as

$$\mathscr{D}_{SH}(\Phi_i) := \{X_n | X_n \in \mathscr{D}(\Phi_i), z_n \geq h_{min}\}$$

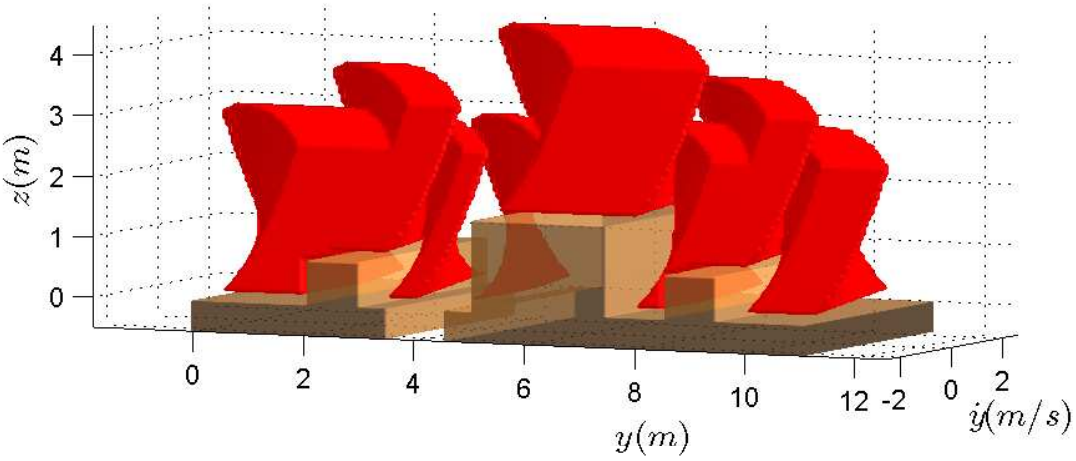

Figure 6.4: Global domain coverage $\mathscr{D}_g := \bigcup_i \mathscr{D}(\Phi_i)$ for a planar rough surface, showing the union of all instantiated policy domains. Note that the depth axis represents the apex velocity.

Apart from this adjustment, policy domains are simply constrained by the selected energy and velocity ranges together with the constraint of landing on the selected ground segment. For a given energy E and velocity $\dot{y}$, the ballistic flight trajectories yield simple expressions for the upper and lower limits of the horizontal position as

$$y_{min}(\dot{y}, E) = y_g - 0.5l_g - \dot{y}\sqrt{\frac{2(E - 0.5m\dot{y}^2 - mgz_g)}{mg}}$$

$$y_{max}(\dot{y}, E) = y_g + 0.5l_g - \dot{y}\sqrt{\frac{2(E - 0.5m\dot{y}^2 - mgz_g)}{mg}}$$

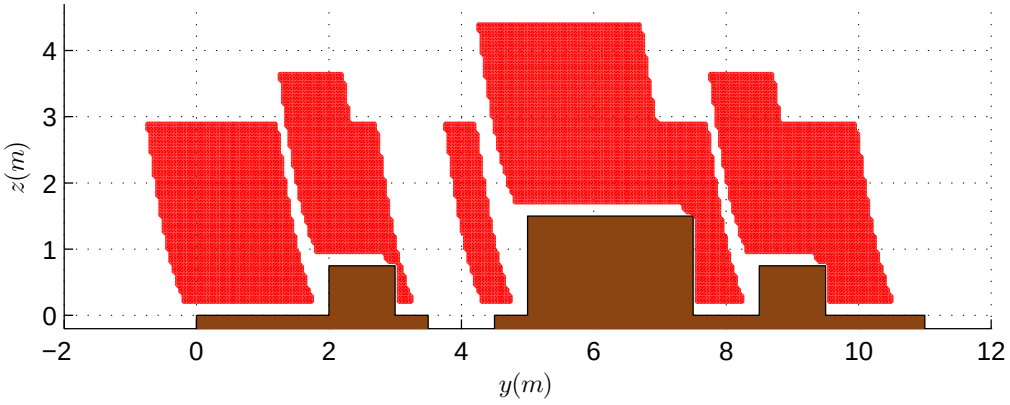

Figure 6.5: A cross section of the global domain $\mathscr{D}_g$ at apex speed $\dot{y} = 1m/s$.

where (y_g, z_g) and l_g are the center and length of ground segment of associated policy.

The resulting simple analytical formulation yields a computationally efficient inclusion test for measured apex states. An illustration of the global domain of attraction $\mathscr{D}_g := \bigcup_i \mathscr{D}(\Phi_i)$ resulting from the deployment of such policies over the complex terrain shown in Fig. 6.3 is illustrated in Fig. 6.4. Similarly, Fig. 6.5 illustrates a cross section of the same domain at $\dot{y} = 1m/s$, showing positions from which the hopper can successfully recover from and find a foothold while traveling at $\dot{y} = 1m/s$.

Derivation of the Feasible Goal Region

The feasible goal region for a policy includes all apex states reachable in a single step from every initial apex state in the domain, using only control inputs from within the allowable set. In deriving a computational representation of this set, we proceed by analyzing each of the three dimensions in the apex state, namely the height z, the horizontal position y and the forward velocity $\dot{y}$. The following steps outline our method for finding the representation of $\mathscr{G}_f(\Phi_i)$ where we will omit analytical details for space considerations.

i) We first find the feasible z range, $[z_{min}, z_{max}]$, using analytical solutions to the equations

$$z_{min} = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmax}} \ (\underset{u \in U(\Phi_i)}{\text{argmin}} \ \pi_2 \circ F_a(X_n, u))$$
$$z_{max} = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmin}} \ (\underset{u \in U(\Phi_i)}{\text{argmax}} \ \pi_2 \circ F_a(X_n, u))$$

ii) Then, for any $z \in [z_{min}, z_{max}]$, we find the feasible y range, $[y_{min}, y_{max}]$, by analytically solving

$$y_{min}(z) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmax}} \ (\underset{z_{n+1}=z, \ u \in U(\Phi_i)}{\text{argmin}} \ \pi_1 \circ F_a(X_n, u))$$
$$y_{max}(z) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmin}} \ (\underset{z_{n+1}=z, \ u \in U(\Phi_i)}{\text{argmax}} \ \pi_1 \circ F_a(X_n, u))$$

iii) Finally, for any $z \in [z_{min}, z_{max}]$ and $y \in [y_{min}, y_{max}]$, we find the feasible velocity range, $[\dot{y}_{min}, \dot{y}_{max}]$, by

$$\dot{y}_{min}(z, y) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmax}} \ (\underset{(z,y)_{n+1}=(z,y), \ u \in U(\Phi_i)}{\text{argmin}} \ \pi_3 \circ F_a(X_n, u))$$
$$\dot{y}_{max}(z, y) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmin}} \ (\underset{(z,y)_{n+1}=(z,y), \ u \in U(\Phi_i)}{\text{argmax}} \ \pi_3 \circ F_a(X_n, u))$$

Note that we have been able to derive fully analytical solutions to these equations, yielding a very simple inclusion test for membership in the feasible goal set. Fig. 6.6 illustrates the resulting goal regions $\mathscr{G}_g := \mathscr{D}_g \bigcap (\bigcup_i \mathscr{G}_f(\Phi_i))$ for the example of Fig. 6.3. Note that feasible goal regions that do not intersect any of the instantiated policy domains can be considered "as good as lost" since, by definition, states that are not in any the domain of any policies correspond to catastrophic situations from which none of the existing controllers are capable of recovering.

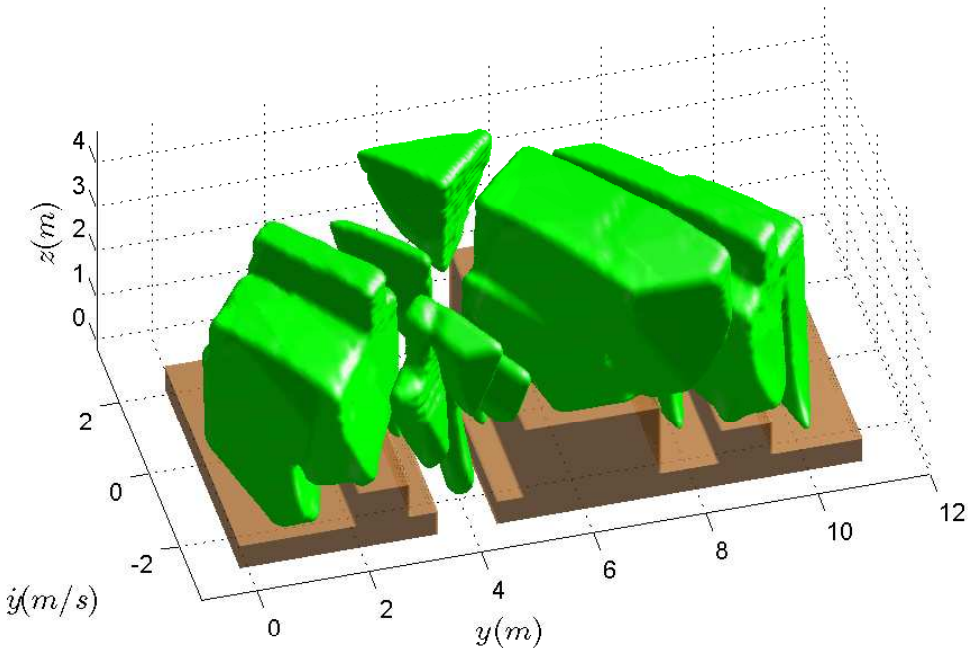

Figure 6.6: Global goal coverage $\mathscr{G}_g := \mathscr{D}_g \bigcap (\bigcup_i \mathscr{G}_f(\Phi_i))$ over a planar rough surface, showing the union of feasible goal regions for all instantiated policies that are also inside the global domain. Note that the depth axis represents the apex velocity.

6.2.2 Simulation Results

Simulation Environment

In order to illustrate the effectiveness of our algorithm, we conducted a range of simulations of the simplified planar hopper on the rough terrain illustrated in Fig. 1.5, with different initial conditions and goal configurations under both ideal models as well as different noise conditions. All simulations were run in Matlab, numerically integrating the equations of motion described in Section 6.2.1 using control inputs selected by the reactive controller deployment.

Policy Generation and Deployment

Our planning algorithm assumes that a map of the environment is available in the form of the locations and heights of each flat ground segment. In practice,

this information can be obtained through exteroceptive sensors such as cameras and range sensors as the robot locomotes over new terrain.

Before our planning algorithm in Table 6.2 can be applied, a collection of local control policies Φ_i must be generated. To this end, we start by coarsely discretizing the known environment map into a set of constant length ($l_g = 0.25m$) sections. We then consider for each such region, four "exiting" policies that allow free transition between forward ($R_V(\Phi_i) = [0, 2.5]m/s$) and backward ($R_V(\Phi_i) = [-2.5, 0]m/s$) locomotion (ff, bb, fb, bf) and two "goal" policies that can stop the robot from either slow forward ($R_V(\Phi_i) = [0, \epsilon]m/s$) or slow backward ($R_V(\Phi_i) = [-\epsilon, 0]m/s$) locomotion (fs, bs). Orthogonal to these choices, we also consider different energy levels by imposing a global constraint on the hopping height as $z \in [0.2, 3]m$ and dividing this range into as many energy levels as necessary to obtain policies whose domain and goal regions exhibit maximal overlap. This results in four different energy levels in our case: very low, low, medium and high, yielding a total of $6*4 = 24$ policies associated with each ground segment.

For each policy, we also impose certain limits on control inputs. First, global limits on the velocity gain and angle adjustment are imposed with $k \in [0.5, 2]$, and $\theta \in [-\pi/2, \pi/2]rad$. Limits on the horizontal shift Δy are designed to ensure close correspondence to trajectories feasible in the SLIP model. For steps resulting in nonzero forward or backward speeds, we require $\Delta y \in [0, 0.5]m$, and $\Delta y \in [-0.5, 0]m$, respectively. In contrast, for stopping controllers, we require smaller displacements with $\Delta y \in [-0.25, 0.25]m$, realized through a two-step controller. As a result of this construction, we assign each policy to their corresponding ground segment $R_g(\Phi_i)$, apex velocity range $R_V(\Phi_i)$, apex energy range $R_E(\Phi_i)$, minimum apex height $F_{h_{min}}(\Phi_i)$, and allowable control set $U(\Phi_i)$.

Once all policies are generated (a total of 960 for the example in Fig. 1.5, corresponding to 40 ground segments), we proceed with the generation of the

prepares graph $\mathcal{G}$. Having the analytical representations and associated inclusion checks for the domain and goal regions described in Sections 6.2.1 and 6.2.1, the construction of the prepares graph is straightforward and can be done offline. This graph, whose representation is very concise with only as many nodes as there are policies, can be reused every time a different apex goal state is supplied.

The deployment of a reactive controller for a specific apex goal involves the application of the algorithm shown in Table 6.2 to obtain a total order of instantiated policies $\hat{\Phi}_i$ from the prepares graph $\mathcal{G}$ constructed above. During execution, the reactive controller measures the system state at every apex, performs a prioritized inclusion test to determine which policy to activate, and then applies a single-step deadbeat controller to select control inputs that will bring the hopper to the goal state associated with the selected policy instance.

Results

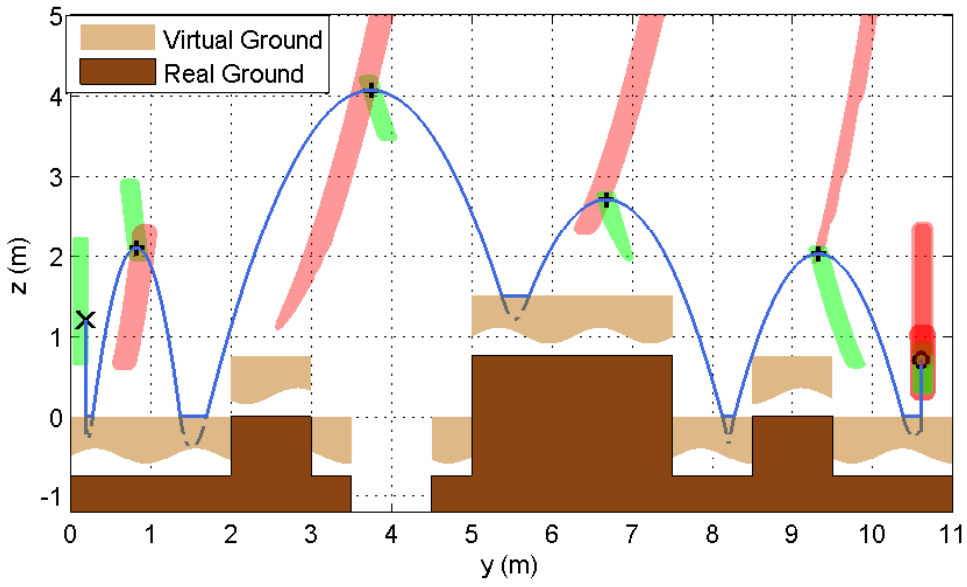

Figure 6.7: A example hopper trajectory over rough terrain with reactive planning, starting from initial state $y = 0.2m, z = 1.2m, \dot{y} = 0$ and going to the goal $y = 10.6m, z = 0.7m, \dot{y} = 0$. Cross sections of domain (green) and feasible goal (red) regions are illustrated at every apex event.

Fig. 6.7 illustrates an example run with reactive control over rough terrain, starting from $y = 0.2m$ and going to a goal state of $y = 10.6m$. At every apex, the reactive controller performs ordered inclusion tests on all policy domains and selects the first match as the policy to apply. The domain of the selected policy is illustrated with the red region in the figure while the feasible goal for the previously used policy is illustrated with the green region. As visible from the figure, the prepares relation is satisfied with nonempty intersections with the feasible goal and domain regions of successive policies. Furthermore, policies are instantiated with goals that are maximally safe, lying as far in the domain of the next policy as possible. Note that only activated policies are shown, but the union of all domains has substantially more coverage as shown in Fig. 6.4. This example illustrates that under ideal conditions with no model or measurement uncertainty, our planner and reactive controller performs as expected.

In contrast to the ideal environment with no model uncertainty, Fig. 6.8 illustrates simulation runs with a constant "wind" force, constantly pushing the hopper to the East. The top figure shows that if control inputs computed offline under an ideal model assumption are applied, hopper trajectories slowly deviate from the generated "plan" and eventually crash into the side of the wall around $y = 8.5m$. In contrast, the application of our reactive control method ensures that proper control policies are selected at each apex, safely taking the hopper across the terrain.

Finally, Fig. 6.9 illustrates a scenario wherein the "sensed" ground (i.e. the ground profile used by the planner, shown with dashed lines) is different than the actual ground profile. This corresponds to a possibly more problematic situation since rather than the gradual noise introduced by the wind disturbance above, surface discrepancies may result in sudden, large disturbances that may quickly invalidate previously constructed plans. Indeed, as show in the figure, the large difference between the sensed ground and the actual ground profile in the

range $y \in [0, 1]m$ causes the application of control inputs computed offline to fail catastrophically, causing a crash into the wall at $y = 5m$. However, our reactive controller, once it finds itself in a new, unexpected apex state, automatically selects the control policy that is guaranteed to eventually drive it to the overall goal, following a completely different plan than what was originally intended.

6.2.3 Discussion

In this part of the thesis, we introduced a novel algorithm for the automated construction of a reactive footstep controller for a planar hopper. Our method is based on a careful characterization of the attracting domain and feasible goal sets of potential footholds on a piecewise flat surface map, combined through backchaining in a sequential composition framework to yield a full reactive control policy that guides the robot to a specified goal point, while providing an almost global region of attraction for the overall behavior.

We demonstrate the performance of our algorithm with a series of simulations of a simplified planar hopper, capturing essential properties of the popular SLIP model while preserving analytical derivability of domain and goal regions for individual control policies and associated deadbeat controllers. We show that even in the presence of significant model and measurement noise, the global controller deployed by our algorithm is capable of successfully reaching the desired goal point, automatically taking alternative paths when the original, ideal plan fails due to unexpected disturbances. Compared to existing, mostly quasi-static footstep planning algorithms, both the ability of our algorithm to handle fully dynamic legged locomotion as well as its robustness against external, large sources of noise represents an important step towards autonomous deployment of legged robots on realistic, rough terrain.

In the near future, we will extend our results and analytical domain and goal representations obtained for the simplified hopper to the more realistic SLIP model. This will ensure that our results are immediately applicable to a large class of legged robots whose morphology and controls closely parallel those of the SLIP model. Also, in the near future, we are planning to demonstrate the experimental applicability of our method through a SLIP-like robot.

Automated deployment with variable length ground segment selection and the incorporation of inclined surfaces are the among possible future extensions to the method proposed above. The consideration of ceiling constraints may also be interesting for indoor or otherwise covered settings. Finally, simultaneous mapping and policy deployment for fully autonomous utilization of the proposed algorithm is among the interesting future directions for this research.

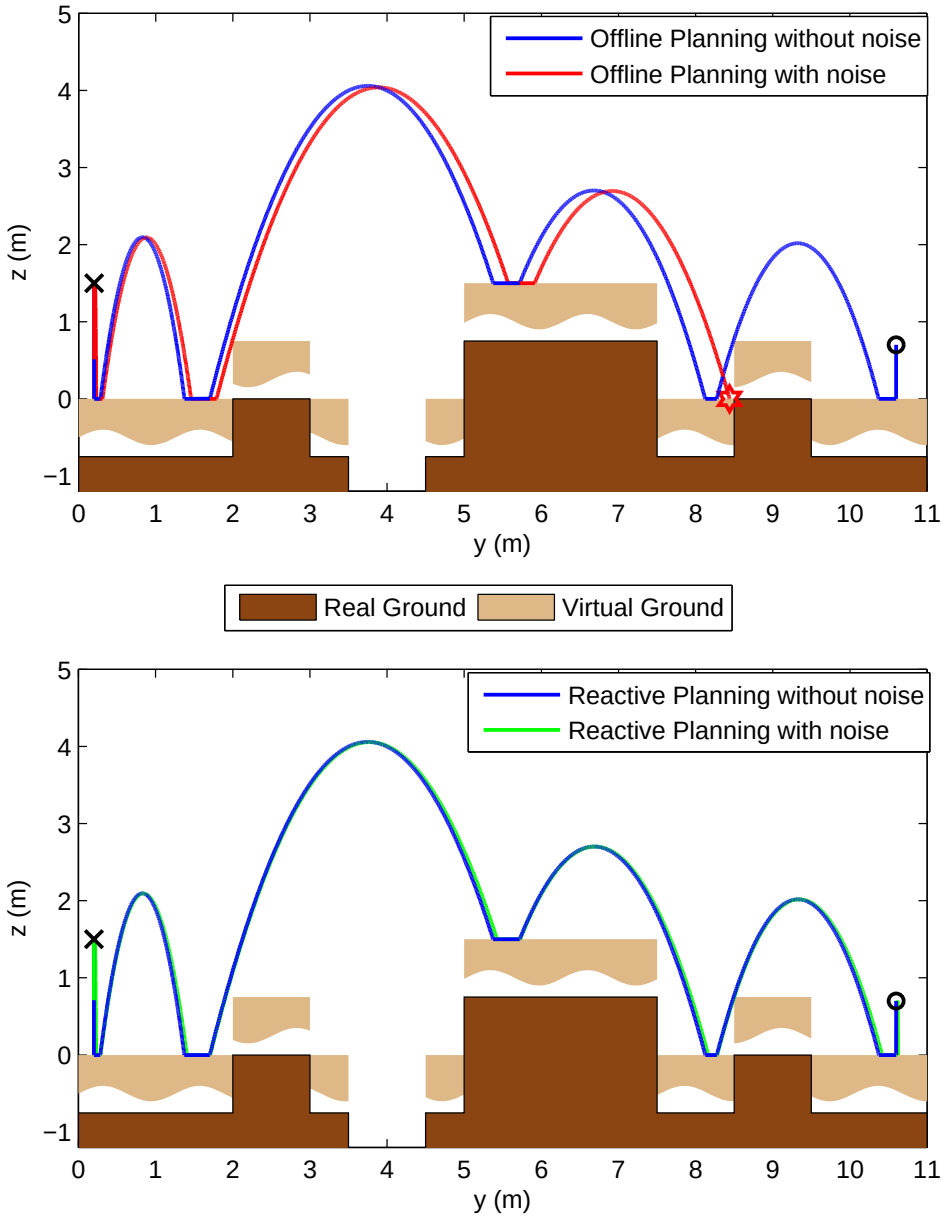

Figure 6.8: Example hopper trajectories under a constant "wind" force of 0.02 N in the East direction. Top figure compares trajectories with no noise (green) to trajectories when control inputs computed offline are directly applied (red). The bottom figure compares trajectories with no noise (green) to trajectories resulting from our reactive control (blue).

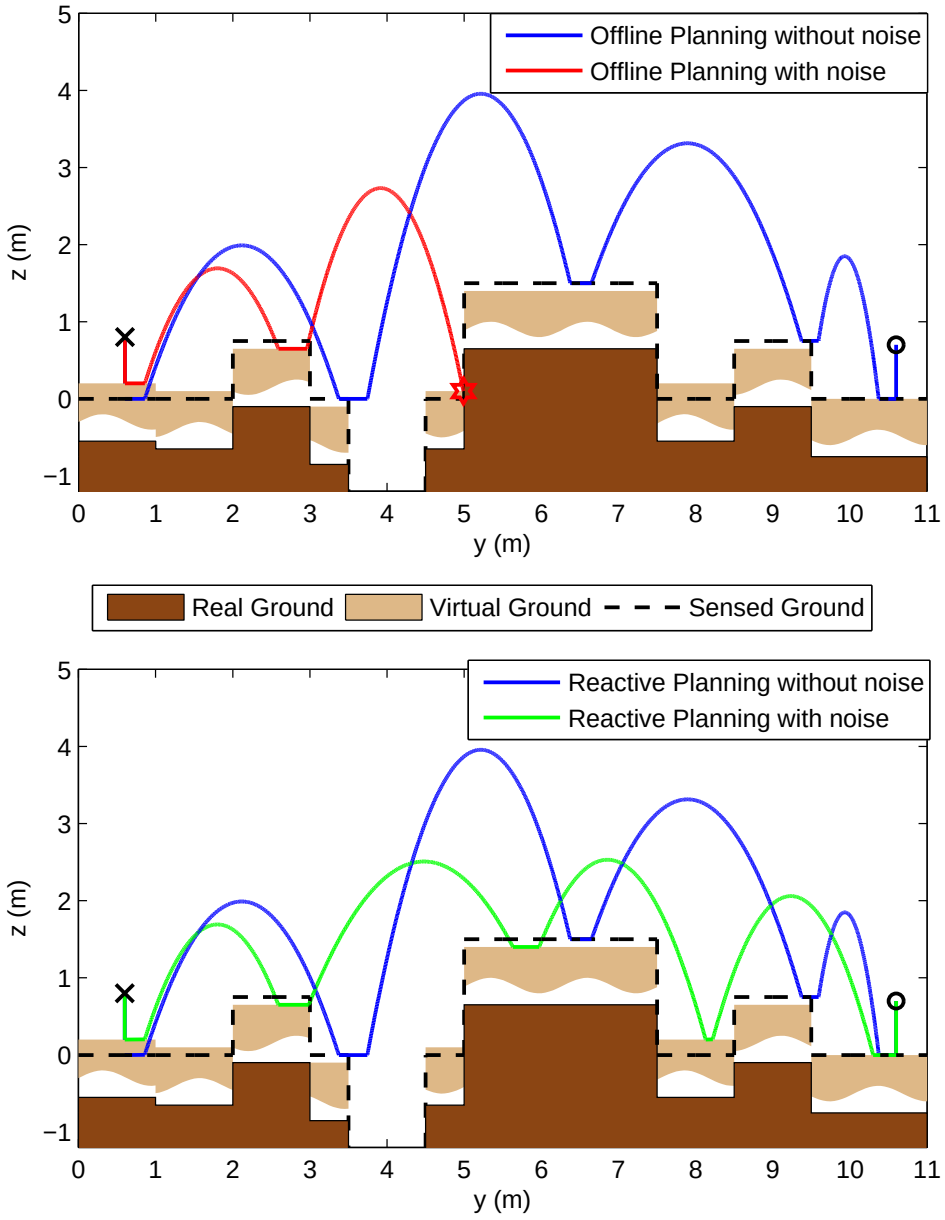

Figure 6.9: Example hopper trajectories under a mismatch between sensed and actual ground heights. Top figure compares trajectories with no noise (green) to trajectories when control inputs computed offline are directly applied (red). The bottom figure compares trajectories with no noise (green) to trajectories resulting from our reactive control (blue).

Chapter 7

CONCLUSIONS

The Spring-Loaded Inverted Pendulum (SLIP) model has long been established as an effective and accurate descriptive model for running animals of widely differing sizes and morphologies. In this thesis, we studied analytical models for and control of nonsymmetric SLIP steps, and reactive footstep planning of a planar spring mass hopper.

The existing studies are generally related to modelling and controlling of symmetric SLIP steps. However, nonsymmetric steps are indispensable for legged locomotion to accelerate, decelerate, step up and step down. Eventually, in the first part of the thesis, we proposed a gravity correction method to compensate the gravity effect on angular momentum to enable much more accurate approximations for the SLIP stance dynamics during nonsymmetric steps. We compared the prediction performance of our approximation with the previous studies on approximating SLIP stance dynamics [2] and [3], and our method was found to perform better for a reasonable range of nonsymmetric steps. Furthermore, we introduced approximate stance maps for two-phase variable stiffness case based on the approximation stance maps in [2] , [3] and [35]. We demonstrated that gravity correction method performs better for two-phase variable stiffness. In

short, with a simple correction term, we obtained more accurate approximate stance map for nonsymmetric steps as well.

In the second part of the thesis, we studied on position aware deadbeat controllers which are different than many existing deadbeat controllers. By using the SLIP properties on flat surfaces, we designed three different deadbeat controller, LLC, LSC and TPSC, based on the approximate stance map with gravity correction for two-phase stiffness control. We illustrated the performance of deadbeat controller for both symmetric and nonsymmetric steps. We also studied on the design of reactive planning controllers for dynamic legged robots. We introduced a novel reactive footstep planning algorithm based on sequential composition and illustrated the performance of the proposed algorithm on a simplified hopper and showed the reactivity and robustness of the algorithm with respect to the environment noise and modelling uncertainty.

We finished this thesis with a lot of interesting open research topics. One of the novel extension of our reactive footstep algorithm is more realistic characterization of the domain and goal region for the planar SLIP model. We may use the undisturbed system approach (zero gravitational acceleration during stance phase) to find approximate analytical forms of the domain and goal region of a local policy. Another interesting problem is simultaneous mapping and policy deployment to achieve much more autonomy. The next stage of this problem may be adding some logical reasoning capabilities for more intelligent behaviors. One of the necessary future work is experimental verification of all these extensions on a real hopper.

Additionally, 3D SLIP Model and 2D SLIP with aptitude pose a number of still unsolved problems. Approximate stance map for these platforms and reactive footstep planning and control of these systems are unanswered problems of dynamical legged locomotion.

Furthermore, the analytical models for the SLIP stance trajectory and position aware deadbeat controllers can also be used with probabilistic approach like the Rapidly-Exploring Random Tree for efficient motion planning SLIP like systems.

All these research questions will fill a significant gap in the field of dynamical legged locomotion.

Bibliography

[1] G. Zeglin, *The Bow Leg Hopping Robot*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, October 1999.

[2] H. Geyer, A. Seyfartha, and R. Blickhanb, "Spring-mass running: simple approximate solution and application to gait stability," *Journal of Theoretical Biology*, vol. 232, pp. 315 –328, February 2005.

[3] W. J. Schwind and D. E. Koditschek, "Approximating the stance map of a 2 dof monoped runner," *Journal of Nonlinear Science*, vol. 10, no. 5, pp. 533–588, 2000.

[4] M. LaBarbera, "Why the wheels won't go," *The American Naturalist*, vol. 121, pp. 395–408, March 1983.

[5] J. Diamond, "Why animals run on legs, not on wheels," *Discover*, vol. 4, pp. 64–67, September 1983.

[6] B. M. Yamauchi, "Packbot: a versatile platform for military robotics," in *Proceedings of SPIE: Unmanned Ground Vehicle Technology VI*, vol. 5422, pp. 228–237, September 2004.

[7] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, E. T. Baumgartner, and E. Tunstel, "Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization," *Autonomous Robots*, vol. 14, pp. 103–126, Mar. 2003.

[8] U. Saranli, M. Buehler, and D. E. Koditschek, "RHex: A simple and highly mobile robot," *International Journal of Robotics Research*, vol. 20, pp. 616–631, July 2001.

[9] M. H. Raibert, *Legged robots that balance*. Cambridge, MA, USA: Massachusetts Institute of Technology, 1986.

[10] H. Tappeiner, S. Skaff, T. Szabo, and R. Hollis, "Remote haptic feedback from a dynamic running machine," in *IEEE International Conference on Robotics and Automation*, (Kobe, Japan), May 12 - 17 2009.

[11] M. J. Spenko, G. C. Haynes, J. A. Saunders, M. R. Cutkosky, A. A. Rizzi, R. J. Full, and D. E. Koditschek, "Biologically inspired climbing with a hexapedal robot," *J. Field Robot.*, vol. 25, no. 4-5, pp. 223–242, 2008.

[12] K. Autumn, M. Buehler, M. Cutkosky, R. Fearing, R. J. Full, D. Goldman, R. Groff, W. Provancher, A. A. Rizzi, U. Saranli, A. Saunders, and D. E. Koditschek, "Robotics in scansorial environments," *Proceedings of the SPIE,*, vol. 5804, pp. 291–302, May 2005.

[13] D. Goldman, H. Komsuoglu, and D. Koditschek, "March of the sandbots," *Spectrum, IEEE*, vol. 46, pp. 30–35, April 2009.

[14] R. Altendorfer, U. Saranli, H. Komsuoglu, D. Koditschek, H. B. Brown, M. Buehler, N. Moore, D. McMordie, and R. Full, "Evidence for Spring Loaded Inverted Pendulum Running in a Hexapod Robot," in *Proceedings of the International Symposium on Experimental Robotics*, (Honolulu, HI), 2001.

[15] R. Blickhan and R. J. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, vol. 173, pp. 509–517, November 1993.

[16] R. J. Full and D. E. Koditschek, "Templates and Anchors: Neuromechanical Hypotheses of Legged Locomotion," *Journal of Experimental Biology*, vol. 202, pp. 3325–3332, 1999.

[17] R. M. Alexander, "Three uses for springs in legged locomotion.," *International Journal of Robotics Research*, vol. 9, no. 2, pp. 53–61, 1990.

[18] C. T. Farley and D. P. Ferris, "Biomecahnics of Walking and Running: Center of Mass Movements to Muscle Action," *Excercise and Sport Science Reviews*, vol. 26, pp. 253–283, 1998.

[19] P. Gregorio, M. Ahmadi, and M. Buehler, "Design, Control, and Energetics of an Electrically Actuated Legged Robot," *Transactions on Systems, Man, and Cybernetics*, vol. 27, pp. 626–634, August 1997.

[20] A. Sato and M. Buehler, "A planar hopping robot with one actuator: design, simulation, and experimental results," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 4, pp. 3540–3545 vol.4, Sept.-2 Oct. 2004.

[21] J. W. Hurst, J. Chestnutt, and A. Rizzi, "Design and Philosophy of the BiMASC, a Highly Dynamic Biped," in *Proc. of the Int. Conf. on Robotics and Automation*, April 2007.

[22] U. Saranli and D. E. Koditschek, "Template based control of hexapedal running," in *Proceedings of the IEEE International Conference On Robotics and Automation*, September 2003. accepted for publication.

[23] W. J. Schwind, *Spring loaded inverted pendulum running: a plant model*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1998. Chair-Daniel E. Koditschek.

[24] S. G. Carver, *Control of a spring-mass hopper*. PhD thesis, Cornell University, Ithaca, NY, USA, January 2003. Adviser-John Guckenheimer.

[25] P. Holmes, "Poincaré, celestial mechanics, dynamical-systems theory and "chaos".," *Physics Reports (Review Section of Physics Letters)*, vol. 193, pp. 137–163, Sept. 1990.

[26] J. Hodgins and M. Raibert, "Adjusting step length for rough terrain locomotion," *Robotics and Automation, IEEE Transactions on*, vol. 7, pp. 289–298, Jun 1991.

[27] J. Kuffner, J.J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue, "Footstep planning among obstacles for biped robots," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, vol. 1, pp. 500–505 vol.1, 2001.

[28] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami, "Planning biped navigation strategies in complex environments," in *Proceedings of the 2003 International Conference on Humanoid Robots*, October 2003.

[29] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade, "Footstep planning for the honda asimo humanoid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[30] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[31] D. C. Conner, H. Choset, and A. A. Rizzi, "Integrated planning and control for convex-bodied nonholonomic systems using local feedback," in *Proceedings of Robotics: Science and Systems II*, (Philadelphia, PA), pp. 57–64, MIT Press, August 2006.

[32] D. C. Conner, *Integrating Planning and Control for Constrained Dynamical Systems*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 2008.

[33] D. C. Conner, H. Choset, and A. A. Rizzi, "Flow-through policies for hybrid controller synthesis applied to fully actuated systems," *Robotics, IEEE Transactions on*, vol. 25, pp. 136–146, Feb. 2009.

[34] A. A. Rizzi, J. Gowdy, and R. L. Hollis, "Distributed coordination in modular precision assembly systems," *The International Journal of Robotics Research*, vol. 20, no. 10, pp. 819–838, 2001.

[35] O. Arslan, U. Saranlı, and O. Morgül, "An aproximate stance map of the spring mass hopper with gravity correction for nonsymmetric locomotions," in *Proc. of the Int. Conf. on Robotics and Automation*, (Kobe, Japan), 2009.

[36] O. Arslan, U. Saranlı, and O. Morgül, "Reactive footstep planning for a planar spring mass hopper," in *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2009), accepted*, (St. Louis, Missouri, USA), October 2009.

[37] E. T. Whittacker, *A treatise on the analytical dynamics of particles and rigid bodies, Fourth ed.* New York: Cambridge University Press, 1904.

[38] H. B. Brown and G. Zeglin, "The bow leg hopping robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Leuven, Belgium), May 1998.

[39] G. Zeglin and H. B. Brown, "Control of a bow leg hopping robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Leuven, Belgium), May 1998.

[40] K. C. Galloway, J. E. Clark, and D. E. Koditschek, "Design of a multi-directional variable stiffness leg for dynamic running," in *2007 ASME International Mechanical Engineering Congress and Exposition*, (Seattle, Washington, USA), November 11-15 2007.

[41] J. Y. Jun and J. Clark, "Dynamic stability of variable stiffness running," in *2009 IEEE International Conference on Robotics and Automation*, (Kobe, Japan), May 12-17 2009.

[42] U. Saranli, *Dynamic locomotion with a hexapod robot*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 2002. Chair-Daniel E. Koditschek.

[43] U. Saranli, W. J. Schwind, and D. E. Koditschek, "Toward the control of multi-jointed, monoped runner," in *Proceedings of the IEEE International Conference On Robotics and Automation*, (Leuven, Belgium), pp. 2676–2682, May 1998.

[44] M. M. Ankaralı, O. Arslan, and U. Saranlı, "An analytical solution to the stance dynamics of passive spring-loaded inverted pendulum with damping," in *12th International Conference on Climbing and Walking Robots and The Support Technologies for Mobile Machines (CLAWAR'09), accepted*, (Istanbul, Turkey), September 2009.

[45] W. J. Schwind, J. Ji, and D. E. Koditschek, "A physically motivated further note on the mean value theorem for integrals," *The American Mathematical Monthly*, vol. 106, pp. 559–564, June-July 1999.

[46] A. Arampatzis, G.-P. Briggemann, and V. Metzler, "The effect of speed on leg stiffness and joint kinematics in human running," *Journal of Biomechanics*, vol. 32, pp. 1349–1353, 1999.

[47] J. J. Kuffner and Jr., "Goal-directed navigation for animated characters using real-time path planning and control," in *In Proceedings of Captech'98*, pp. 171–186, Springer-Verlag, 1998.

[48] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "On the nonholonomic nature of human locomotion," *Auton. Robots*, vol. 25, no. 1-2, pp. 25–35, 2008.

[49] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade, "Vision-guided humanoid footstep planning for dynamic environments," in *Proceedings of the IEEE-RAS Conference on Humanoid Robots (Humanoids'05)*, pp. 13 – 18, December 2005.

[50] J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue, "Online footstep planning for humanoid robots," in *in Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 932–937, IEEE, September 2003.

[51] J. Gutman, M. Fukuchi, and M. Fujita, "Real-time path planning for humanoid robot navigation," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1232–1238, 2005.

[52] Z. Shiller, K. Yamane, and Y. Nakamura, "Planning motion patterns of human figures using a multi-layered grid and the dynamics filter," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 1–8 vol.1, 2001.

[53] K. Harada, M. Morisawa, K. Miura, S. Nakaoka, K. Fujiwara, K. Kaneko, and S. Kajita, "Kinodynamic gait planning for full-body humanoid robots," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1544–1550, Sept. 2008.

[54] G. Zeglin and H. J. Brown, "First hops of the 3d bow leg," in *Proceedings of the 5th International Conference on Climbing and Walking Robots*, pp. 357–364, 2002.

[55] D. C. Conner, A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, vol. 4, pp. 3546– 3551, IEEE, October 2003.

[56] G. Kantor and A. A. Rizzi, *Robotics Research*, vol. Volume 15/2005 of *Springer Tracts in Advanced Robotics*, ch. Feedback Control of Underactuated Systems via Sequential Composition: Visually Guided Control of a Unicycle, pp. 281–290. Springer Berlin / Heidelberg, August 2005.

[57] E. Westervelt and J. Grizzle, "Sequential composition of walking motions for a 5-link planar biped walker." Workshop on Future Directions in Nonlinear Control of Mechanical Systems, October 2 2002. University of Illinois, Urbana-Champaign, IL.

[58] D. C. Conner, H. Kress-Gazit, H. Choset, A. Rizzi, and G. J. Pappas, "Valet parking without a valet," in *Proceedings of 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, (San Diego, CA), pp. 572–577, IEEE, October 2007.