ADAPTIVE CONTROL OF A ONE-LEGGED HOPPING ROBOT THROUGH DYNAMICALLY EMBEDDED SPRING-LOADED INVERTED PENDULUM TEMPLATE

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND

ELECTRONICS ENGINEERING

AND THE GRADUTE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

İsmail Uyanık

August 2011

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. Orhan Arıkan

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. Melih Çakmakçı

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of Graduate School of Engineering and Science

ABSTRACT

ADAPTIVE CONTROL OF A ONE-LEGGED HOPPING ROBOT THROUGH DYNAMICALLY EMBEDDED SPRING-LOADED INVERTED PENDULUM TEMPLATE

İsmail Uyanık
M.S. in Electrical and Electronics Engineering
Supervisor: Prof. Dr. Ömer Morgül
August 2011

Practical realization of model-based dynamic legged behaviors is substantially more challenging than statically stable behaviors due to their heavy dependence on second-order system dynamics. This problem is further aggravated by the difficulty of accurately measuring or estimating dynamic parameters such as spring and damping constants for associated models and the fact that such parameters are prone to change in time due to heavy use and associated material fatigue.

In the first part of this thesis, we present an on-line, model-based adaptive control method for running with a planar spring-mass hopper based on a once-per-step parameter correction scheme. Our method can be used both as a system identification tool to determine possibly time-varying spring and damping constants of a miscalibrated system, or as an adaptive controller that can eliminate steady-state tracking errors through appropriate adjustments on dynamic system parameters.

We use Spring-Loaded Inverted Pendulum (SLIP) model, which is the mostly used, effective and accurate descriptive tool for running animals of different sizes

and morphologies, to evaluate our algorithm. We present systematic simulation studies to show that our method can successfully accomplish both accurate tracking and system identification tasks on this model. Additionally, we extend our simulations to Torque-Actuated Dissipative Spring-Loaded Inverted Pendulum (TD-SLIP) model towards its implementation on an actual robot platform.

In the second part of the thesis, we present the design and construction of a one-legged hopping robot we built to test the practical applicability of our adaptive control algorithm. We summarize the mechanical, electronics and software design of our robot as well as the performed system identification studies to calibrate the unknown system parameters. Finally, we investigate the robot's motion achieved by a simple torque-actuated open loop controller.

Keywords: Spring-Mass Hopper, Adaptive Control, System Identification, Legged Locomotion, Spring-Loaded Inverted Pendulum (SLIP), Hybrid System

ÖZET

TEK BACAKLI ZIPLAYAN BIR ROBOTUN DINAMIK OLARAK GÖMÜLMÜŞ YAYLI TERS SARKAÇ ŞABLONU İLE ADAPTIF KONTROLÜ

İsmail Uyanık
Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Ömer Morgül
Ağustos 2011

Model tabanlı dinamik bacaklı davranışların pratik olarak gerçeklenmesi ikinci derece sistem dinamiklerine aşırı bağlılıkları nedeniyle statik kararlı davranışlara göre oldukça zordur. Bu sorun ilgili modellerde yer alan yay ve sönümlenme sabiti gibi dinamik parametreleri tam olarak ölçmekteki zorluklar ve bu parametrelerin malzeme yorulması ve aşırı kullanım sonucunda zamanla değişme eğilimi göstermesi nedeniyle daha kötü bir hal almaktadır.

Bu tezin ilk bölümünde, adım başına bir kez parametre güncelleme mantığına dayalı olarak düzlemsel bir yay-kütle zıplayanı koşusu için çevrimiçi, model tabanlı bir adaptif kontrol metodu sunulmaktadır. Bu metot gerek kalibre edilmemiş bir sistemin muhtemelen zamanı bağlı yay ve sönümlenme sabitlerini belirleyen bir sistem tanımlama aracı olarak, gerekse dinamik sistem parametreleri üzerinde uygun ayarlamaları yaparak kararlı hal takip hatalarını gideren bir adaptif kontrolcü olarak kullanılabilir.

Algoritmayı değerlendirebilmek için çok farklı boyut ve morfolojideki koşan canlıları etkili ve güvenilir bir biçimde tanımlayan ve çokça kullanılan Yaylı

Ters Sarkaç (YTS) modeli kullanılmaktadır. Metodun hem güvenilir takip hem de sistem tanımlama görevlerini bu model üzerinde başarılı bir şekilde yapabildiğini göstermek amacıyla sistematik simülasyon çalışmaları sunulmaktadır. Ayrıca, metodun fiziksel bir robot platformunda gerçeklenmesine adım olarak simülasyon çalışmaları tork tahrikli tüketimli yaylı ters sarkaç (TT-YTS) modeline genişletilmiştir.

Tezin ikinci bölümünde adaptif kontrol algoritmasının pratik olarak uygulanabilirliğini test etme amaçlı üretilen tek bacaklı zıplayan robotun tasarım ve üretimi anlatılmaktadır. Robotun mekanik, donanımsal ve yazılımsal tasarımı ile birlikte bilinmeyen sistem parametrelerini kalibre etme amaçlı gerçekleştirilen sistem tanımlama çalışmaları da özetlenmektedir. Son olarak, robotun basit tork tahrikli açık döngü kontrolcüsüyle ortaya çıkan hareketi sorgulanmaktadır.

Anahtar Kelimeler: Yay-Kütle Zıplayanı, Adaptif Kontrol, Sistem Tanımlama, Bacaklı Hareket, Yaylı Ters Sarkaç (YTS), Melez Sistem

ACKNOWLEDGMENTS

First, I would like to thank my supervisors, Ömer Morgül and Uluç Saranlı, for their guidance, encouragement and tremendous support throughout my study. They were always there to listen and to give advice when I needed. I am very grateful to them for their patience during our research meetings and all the help they've given me to control my interest. Additionally, I would especially want to thank Uluç Saranlı for helping me discover and grow the creativity and enthusiasm that I did not know I had.

The initial ideas of this thesis was formed during the "Adaptive Control Theory" course I have taken from Melih Çakmakçı. I thank him for this wonderful course and his stimulating discussions. I thank Mustafa Mert Ankaralı for his helps on the way of developing adaptive control strategies for this study. I also thank Ömür Arslan for our productive conversations, where I have always been inspired by his expertise. This thesis would not have been possible without their contributions.

Additionally, there are a number of people from my research group, Bilkent Dexterous Robotics and Locomotion (BDRL), who helped me along the way. I am very thankful to Güneş Bayır, Utku Çulha, Özlem Gür, Ali Nail İnal, Sıtar Kortik, Tolga Özaslan, Bilal Turan and Tuğba Yıldız for our wonderful late night studies and discussions.

One of the most important aspect of my studies was the opportunity to work with BDRL undergraduate student members. They pushed me to study harder and harder with their endless energy to work. I thank Murat Aslan, Bahadır Çatalbaş, Mustafa U. Daloğlu, Bilgesu Erdoğan, Murat Kırtay, Gizem Tabak and Burak Yücesoy for their support on preparing experimental setup and accompanying me during late night experiments.

Outside the laboratory, there are some friends who directly or indirectly contributed to my completion of this thesis. I thank my office friends Veli Tayfun Kılıç, Deniz Kerimoğlu, Naci Saldı and Mahmut Yavuzer for their understanding and support. I also thank my friends Hasan Hamzaçebi, Emrullah Korkmaz, Görkem Seçer, Seyit Can Birlik and Emre Akgün for always being there to listen and motivate.

I am also appreciative of the financial support from the Scientific and Technical Research Council of Turkey (TÜBİTAK).

Finally, but forever I owe my loving thanks to my parents, Ayhan and Meryem Uyanık, my brother, Ali Uyanık, my sister, Serpil Tiryaki and my beloved fiancée, Anıl Türel for their undying love, support and encouragement.

Contents

List of Figures

List of Tables

to my family and to my beloved fiancée

Chapter 1

INTRODUCTION

This thesis concerns the design of model-based adaptive control methods for running with planar one-legged hopping robots, as well as the development of such a robot platform. In contrast to existing control algorithms which assume perfect knowledge of dynamic system parameters, our algorithm tries to achieve high tracking accuracy through on-line estimation of these parameters. In addition to that, we also describe the design and construction of a mechanical spring-mass hopper to illustrate the practical applicability of our algorithms under different scenarios, aiming to show that our approach not only works in simulation, but also allows realization in a physical robot platform.

1.1 Motivation and Background

The utility of legged morphologies and associated dynamic behaviors for robust and efficient locomotion across rough terrain has long been established [2, 3]. One of the primary advantages of legged designs is that a legged robot can reach a great portion of the earth's land mass unlike other wheeled and tracked platforms. They do not suffer from restrictions in directions forces can be applied

to the robot body [4], an inevitable problem experienced by wheeled platforms. In addition to that, legs can also be used as sensors or manipulators like animals do in nature. By using legs, it may be possible to sense the position, moveability and structure of objects in the environment. An example use of legs as sensors, in which haptic information from the robot leg is used to detect the weight and moveability of an object, was illustrated in [5].

Legged robots also admit challenging locomotory behaviors which are problematic or sometimes impossible for wheeled or tracked platforms. For instance, RISE, a biologically inspired hexapedal climbing robot, is able to locomote both on level ground and different vertical surfaces such as walls and trees [6, 7]. Another challenging environment for robots is sandy terrain, wherein wheeled robots usually get stuck. However, SandBot, a bioinspired hexapedal robot, can traverse over sand with its typical leg design [8].

Based on aforementioned advantages, one can intuitively conclude that legs are better than wheels for robotic platforms due to their wider range of environment accessibility, their possible usage as sensors and their wide range of locomotory behaviors in scansorial environments. Despite these advantages, there are also substantial difficulties associated with legged robots such as the control problem, gait generation and locomotion which are much easier to handle with wheeled robots. One of the main reasons behind these difficulties is that it is not possible to find a general mathematical model describing numerous legged morphologies with different sizes. Consequently, the main focus of this thesis is limited only to the control of monopedal robot platforms. Since we constrain ourselves to monopedal robots, we will be using the well-known Spring-Loaded Inverted Pendulum (SLIP) model (see Fig. 1.1) which is frequently used as a fundamental template to analyze and estimate animal locomotion.

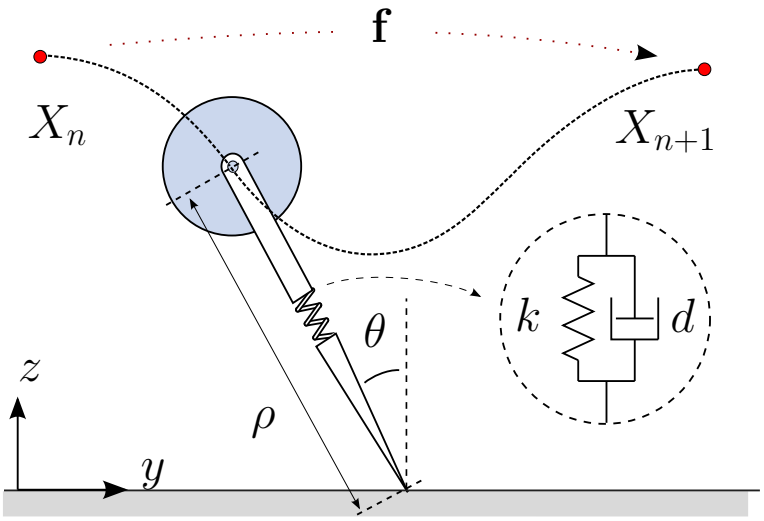


Figure 1.1: The Dissipative Spring-Loaded Inverted Pendulum (SLIP) model. Dashed curve illustrates a single stride from one apex event to the next, defining the return map $X_{n+1} = \mathbf{f}(X_n, \mathbf{u}_n)$.

Nevertheless, despite the discovery of simple mathematical models [9–11] and associated analytical solutions [12–14] that can accurately describe biological runners and support the design of hierarchical controllers for complex legged morphologies [15–18], physical realization of dynamic legged behaviors has mostly been based on intuition and manual tuning [3, 19–21] with a few notable exceptions [22, 23]. More recently, however, there has been increasing interest in using model-based analysis and control methods in this context [24, 25], with experimental success for some behaviors [26].

However, even though dynamic models for which we have a sufficiently good analytic understanding can support physically relevant controller designs, the measurement and estimation of particularly the dynamic parameters, such as spring and damping constants for flexible components of a robotic platform, is still a challenging problem. This problem is further aggravated by the possibly time-varying and unpredictable nature of these parameters for autonomous platforms that may remain operational for extended durations of time. Fortunately, this issue is not confined to the control of legged locomotion and received considerable attention from the adaptive control community [27, 28]. Motivated by

work in this area, this thesis presents a new model-based adaptive control method for running with the SLIP template and its variants, emphasizing on-line estimation of unknown or miscalibrated dynamic system parameters. The adaptive control method we use in this study is a variant of the model reference adaptive control (MRAC a.k.a. MRAS) method in which the desired performance is expressed in terms of a reference model. The objective of the MRAC is to adjust the reference model parameters such that the model response converges to the actual system response [27, 29].

1.2 Existing Work

In contrast to the approach mentioned in the previous section, existing research on adaptive control of legged locomotion almost exclusively focuses on how cyclic behaviors of the mechanical locomotor dynamics can be tuned through their coupling with independently running internal clocks (Central Pattern Generators, CPGs) whose dynamics can then be controlled at lower bandwidth [30–33]. These methods mirror established principles from neurobiology, where groups of neurons in simple organisms were found to remain functional in isolation, producing cyclic control signals even without any high-level control authority [34]. Similar to controller designs based on neural networks and learning [35, 36], such approaches are advantageous in their ability to operate without accurate models, increasing their robustness under unknown environmental conditions such as rough terrain. On the other hand, their structure is often not suitable for incorporating accurate mathematical models when they are in fact available.

The very first experimental studies on actively balanced hopping robots began with Matsuoka [37] who built a planar one-legged hopping robot to study repetitive hopping in human. However, Raibert led the field of dynamically stable legged locomotion by using SLIP as a basic dynamic model for hopping robots

[19]. He built many running robots with different structures and number of legs based on the SLIP model, starting with a planar one-legged machine [38]. The important thing about the Raibert's hoppers is that he uses a modified version of the control algorithm developed for the original one-legged machine in all of his robots. In addition to Raibert's hoppers, Papantoniou [39, 40] also uses a control algorithm based on Raibert's controller in his planar hopper actuated by two electric motors.

Sato also built a one-legged hopping machine, the Sato Hopper [41], to investigate the feasibility of the SLIP model in a robot platform with a spring in the leg and only one actuator at the hip. He achieved experimental validation of the SLIP model and demonstrated a robust hopping robot with only one actuator at the hip. The ARL Monopod I - II are the two other one-legged machines to study the energetics of autonomous dynamically stable legged locomotion based on Raibert's control laws [42, 43]. For example, the ARL Monopod I was the fastest electrically actuated legged robot to date with its top running speed of 1.2 m/s. Among all these robots, the Bow Leg hopper has the closest design to the SLIP model with its actuated compliant leg [44–46]. The springy leg of the Bow Leg Hopper is a bow-shaped fiberglass sheet and it is pre-loaded during the flight phase to store energy. Then, the leg is released to inject the stored energy to the system for the control purposes.

1.3 Methodology

As discussed in Section 1.1, the measurement and estimation of dynamic system parameters is still a challenging problem. Therefore, the assumptions of perfect parameter knowledge and time-invariant parameter values can be quickly violated in actual robotic platforms. In the first part of this thesis, we introduce

a novel adaptive control method that extends on one of the recent control algorithms for leg-length controlled SLIP model and achieves high accuracy even for highly inaccurate initial system parameter estimates.

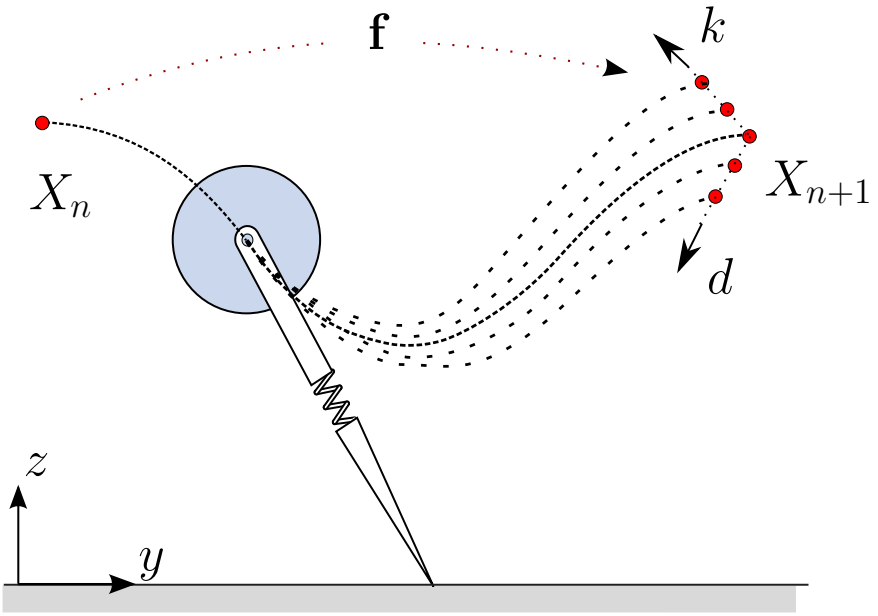


Figure 1.2: Impact of miscalibrated dynamic parameters on SLIP trajectory predictions. Arrows indicate directions of change in the apex as a result of increasing k and d. The curve in the middle shows the unperturbed trajectory while the dotted curves show trajectories with different parameter values.

Our adaptive control method is based on recently proposed analytic approximations to SLIP dynamics [14]. Similar to previous studies, we use a once-per-step deadbeat control strategy that relies on the inversion of an approximate return map for this system. However, unlike previous controllers which assume perfect knowledge of dynamic system parameters (spring and damping constants in particular), and ignore the effects of miscalibrated parameters illustrated in Fig. 1.2, our adaptive controller explicitly considers and compensates for such errors. In order to ensure the realization of our algorithm in an actual robotic platform, we also extend this algorithm to the torque actuated SLIP (TD-SLIP) model described in [1]. We present systematic simulation studies to show that our method performs well in both of these models.

Finally, we designed and built a planar spring-mass hopper platform based on the SLIP model whose details are given in Chapter 4. Since the measurement of ground truth robot state is very important for motion control and planning, we also designed a high accuracy measurement system in the form of a planarizer.

1.4 Contributions

The primary contribution in this thesis is a novel adaptive control algorithm to achieve accurate gait control and system identification for running with a planar hopper (SLIP) model. Our studies show that model-based estimation of leg spring and damping constants can either be used to eliminate tracking errors or to achieve accurate system identification. We provide simulation results in Chapter 3, systematically evaluating the performance of our method in the presence of parameter and modeling errors.

In addition to this primary contribution, our algorithm also eliminates steady-state tracking error resulting from the approximate nature of available return maps. Even with perfect knowledge of dynamic system parameters, there is always a steady-state tracking error since available return maps describing SLIP trajectories are all approximate due to the non-integrability of actual SLIP trajectories. Our algorithm compensates such steady-state errors by making proper adjustments on estimated system parameters.

The final contribution in this thesis is the design and construction of a planar spring-mass hopper robot platform. The leg design of our hopper tries to mimic the SLIP template. In addition to this hopper robot, the planarizer system designed in this thesis can also be used for testing different robot platforms.

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 two existing approximate stance maps for both the SLIP and TD-SLIP models. In Section 3.1, we propose a novel adaptive control algorithm for running with the SLIP template to increase the tracking accuracy and system identification performance in the presence of a parameter uncertainty. Subsequently, in Section 3.2 and Section 3.3, we present the results of systematic simulation studies we performed to evaluate the performance of our algorithm on both the SLIP and TD-SLIP models.

The second part of this thesis begins in Chapter 4 with a summary of the design and construction steps of the one-legged hopper robot we built to test the practical applicability of our adaptive control algorithm on the actual robotic platforms. Section 4.1 details the mechanical, electronics and software design of this robot platform. Then, in Section 4.2, we present the manual system identification studies we performed to calibrate some system parameters of the robot, meaning the spring and damping constant and the effective robot mass at the hip. Then, Section 4.3 details the torque-actuated open loop controller we implemented for our robot and illustrates an example running performance with necessary motion analysis. Finally, in Chapter 5, we conclude the thesis with a review of our work and a summary of open research topics.

Chapter 2

BACKGROUND: THE PLANAR SPRING-MASS HOPPER

This chapter introduces necessary background for the spring-mass hopper as well as a summary of the two important control methods on this model and their analytical approximate maps to the stance trajectories.

2.1 The SLIP Model

The discoveries in biomechanics research revealed that the Spring-Loaded Inverted Pendulum (SLIP) model can be used as a descriptive metaphor for running animals [47]. Based on this fact, the SLIP model was established as a simple and accurate descriptive tool to analyze dynamic locomotion in running animals of widely differing sizes and morphologies [48–50].

Subsequent results from several one-legged hopping platforms based on the SLIP model such as Raibert's hoppers [19], the ARL-Monopods [42], the Bow-Leg

design [44] and the BiMasc [51] platform strengthened the idea of its use an an explicit control target. However, the fact that the stance dynamics of the SLIP model under the effect of gravity are nonintegrable [52] led to the development of analytical approximations to its nonlinear model dynamics to support the analysis of associated behaviours and the design of different controllers [10, 12, 13, 15, 53].

This section continues with the basic SLIP template and introduces its dynamics with the associated terminology.

2.1.1 The SLIP Template

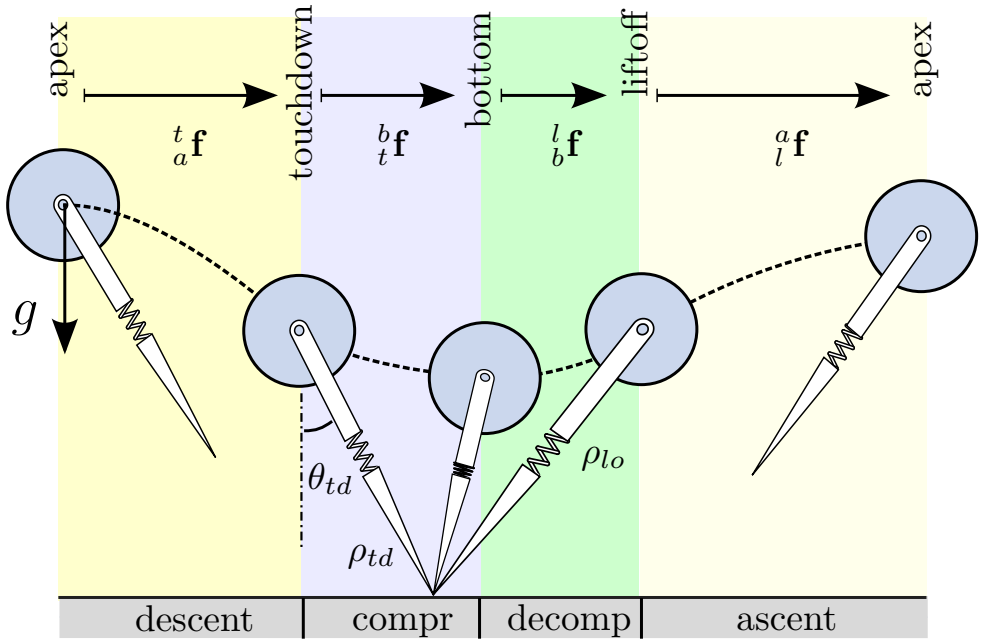


Figure 2.1: SLIP locomotion phases (shaded regions) and transition events (boundaries)

The SLIP model, illustrated in Fig. 1.1, consists of a point mass attached to a massless leg with a linear spring k and viscous damping d. During running, this model alternates between *stance* and *flight* phases with the toe fixed on the ground during the former and the body following a ballistic trajectory during the latter. Moreover, the flight phase is divided into two subphases, *ascent* and

descent according to the vertical velocity. Similarly, the stance phase is also split into two subphases as *compression* and *decompression* according to the sign of the rate of change of the leg length. Fig. 2.1 illustrates a single stride from an apex state and labels all relevant phases, subphases and transition events whose properties will be explained shortly in the next paragraphs. Furthermore, Table 2.1 details the notation associated with the basic SLIP model. Note that, some derivations use non-dimensional parameters to eliminate redundant parameters. These non-dimensional parameters are represented with a dash sign on them. The transformation between the original and non-dimensional parameters are briefly defined in the places where they are first mentioned. More details about these transformations can be found in [1].

Flight : The period when the robot has no contact with the ground. The body follows a simple ballistic trajectory in this phase.

- **Ascent :** The subperiod when the robot moves upwards, meaning that the robot has a decreasing positive vertical velocity.
- **Descent :** The subperiod when the robot moves downwards, meaning that it accelerates in the negative direction for the vertical velocity and increases in magnitude.

Stance : The period when the toe of the robot is in contact with the ground. As told before, the system dynamics in this phase includes nonintegrable terms due to the gravitational acceleration.

- **Compression :** The subperiod when the rate of change of leg length is negative, meaning that the spring is being compressed and stores energy.
- **Decompression :** The subperiod when the rate of change of leg length is positive, meaning that the spring is being decompressed and releases energy.

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

SLIP States, Event States and Control Inputs	
ρ, θ	Leg length and leg angle
$\dot{\rho}, \dot{\theta}$	Leg compression and swing rates
R	Body state vector in polar coordinates, $R = [\theta\ \dot{\theta}\ \rho\ \dot{\rho}]^T$
p_θ	Angular momentum around the toe
$\rho_{td}, \theta_{td}, t_{td}$	Touchdown leg length, angle and time
ρ_b, θ_b, t_b	Bottom leg length, angle and time
$\rho_{lo}, \theta_{lo}, t_{lo}$	Liftoff leg length, angle and time
y, z	Horizontal and vertical body positions
$\dot{y}, \dot{z}$	Horizontal and vertical body velocities
$z_a, \dot{y}_a$	Apex height and velocity
X	Body state vector in cartesian coordinates, $X = [y\ \dot{y}\ z\ \dot{z}]^T$
τ	Hip torque command during stance
SLIP Parameters	
m, g	Body mass and gravitational acceleration
ρ_o	Leg rest length
k, d	Actual values of spring and damping constants
$\hat{k}, \hat{d}$	Estimated values of spring and damping constants
$F_g(x)$	Ground function. It returns the ground height for a given position x
$\mathbf{E}$	Total mechanical energy
Return maps	
$\mathbf{f}$	Exact plant model for SLIP and TD-SLIP
$\hat{\mathbf{f}}$	Analytical approximate solution for SLIP and TD-SLIP
${}^t_a\mathbf{f}$	Apex to touchdown map
${}^b_t\mathbf{f}$	Touchdown to bottom map
${}^l_b\mathbf{f}$	Bottom to liftoff map
${}^a_l\mathbf{f}$	Liftoff to apex map

In addition to these phases, our model also includes several transition events triggering the phase changes during locomotion. The accurate detection of these events is very crucial for simulation or actual locomotion since they strongly impact the locomotion characteristics. Now, we focus on general characteristics of these events.

Apex : This event triggers the state change from the ascent to the descent subphase during flight. This event occurs when the SLIP body reaches its maximum height with the maximum gravitational potential energy. It can be detected by checking the zero crossing of the following function during

the flight phase:

$$\mathbf{h}_a(t) := \dot{z}(t), \tag{2.1}$$

Touchdown : This event triggers the state transition between flight and stance phases. It occurs when the robot touches the ground while in descent. This event can be detected by checking the zero crossing of the following function when $\dot{z} < 0$:

$$\mathbf{h}_{td}(t) := z(t) - (\rho_{td} cos(\theta_{td}) + F_g(y_{td})), \tag{2.2}$$

Bottom : This event triggers the transition from compression to decompression during stance. This event occurs when the spring potential energy reaches its maximum value and can be detected by checking the zero crossing of the following function during stance:

$$\mathbf{h}_b(t) := \dot{\rho}(t), \tag{2.3}$$

Liftoff : This event triggers the stance to flight transition. This event occurs when the toe of the robot leg leaves the ground. This event can be detected by checking the zero crossings of the following function when $\dot{z} > 0$:

$$\mathbf{h}_{lo}(t) := -k(\rho(t) - \rho_o) - d\dot{\rho}(t), \tag{2.4}$$

As it can be understood from the transition event definitions, the highest point in the flight phase is defined as *the apex* point for each stride, with the associated state of the system for the n^{th} stride defined as

$$X_n := [\ z_n,\ \dot{y}_n\ ]^T\ . \tag{2.5}$$

We will also find it convenient to collect relevant dynamic parameters of the system in a single vector as

$$\mathbf{p} := [\ k,\ d\ ]^T\ . \tag{2.6}$$

2.1.2 SLIP Dynamics

As previously stated, the SLIP model has hybrid locomotion characteristics which requires separate consideration of flight and stance dynamics. This section provides the equations of motion for all phases of the SLIP model. These equations will subsequently be converted into non-dimensional coordinates to simplify further analysis.

Flight Dynamics

The SLIP model has a point-mass to represent the whole robot body. During flight, this point-mass follows a ballistic flight trajectory under the effect of gravity. The equations below show the state vector $\mathbf{X}$ is defined as

$$\mathbf{X} := [\ y,\ \dot{y},\ z,\ \dot{z}\ ]^T, \tag{2.7}$$

and the flight dynamics are

$$\dot{\mathbf{X}} := [\ \dot{y},\ 0,\ \dot{z},\ -g\ ]^T, \tag{2.8}$$

Stance Dynamics

During the stance phase, the robot leg is assumed to be rotating around a frictionless revolute joint fixed at the contact point until liftoff occurs. Since the body mass follows an arc-like trajectory, polar coordinates are best suited for the derivation of the stance dynamics. The Lagrangian equation of the SLIP model during the stance phase in polar coordinates can hence be written as

$$L = \frac{m}{2}(\dot{\rho}^2 + \rho^2\dot{\theta}^2) - \frac{k}{2}(\rho_o - \rho)^2 - mg\rho\cos(\theta). \tag{2.9}$$

Then, the equations of motion can be derived as

$$m\ddot{\rho} = m\rho\dot{\theta}^2 + k(\rho_o - \rho) - mg\cos(\theta), \tag{2.10}$$

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

Using the state vector, R, defined in polar coordinates as

$$R := \begin{bmatrix} \theta, & \dot{\theta}, & \rho, & \dot{\rho} \end{bmatrix}^T, \tag{2.12}$$

as well as (2.10) and (2.11), the stance dynamics in polar coordinates are given by

$$\dot{R} = \begin{bmatrix} \dot{\theta} \\ -\frac{g\sin(\theta)}{\rho} - \frac{2\dot{\rho}\dot{\theta}}{\rho} \\ \dot{\rho} \\ \frac{k(\rho_o - \rho)}{m} + \rho\dot{\theta}^2 - g\cos(\theta) \end{bmatrix}. \tag{2.13}$$

2.1.3 Control of SLIP Locomotion

In this section, we focus on the control problem for the SLIP model and briefly explain different possible control modes. Control of SLIP locomotion generally seeks to regulate apex states of the model during locomotion through discrete control inputs at each stride.

Equation (2.7) defines the SLIP state $\mathbf{X}$ in cartesian coordinates. Now, given the control input vector $\mathbf{u}$ (which will be specified shortly), a Poincaré section at apex with $\dot{z} = 0$ enables us to define a discrete apex return map as

$$X_{n+1} = \mathbf{f_p}(X_n, \mathbf{u}_n)\ . \tag{2.14}$$

where the dependence of the map on the dynamic parameters $\mathbf{p}$ is explicitly shown.

Unfortunately, the stance dynamics of the SLIP model are not integrable in closed form, making it impossible to find exact analytic expressions for the apex return map [52]. Consequently, we use analytic approximations to model the actual return map and define the new approximate return map as

$$\hat{X}_{n+1} = \hat{\mathbf{f}}_\mathbf{p}(X_n, \mathbf{u}_n)\ . \tag{2.15}$$

Now, suppose that we want to reach the desired apex state,

$$\mathbf{X}^* = \begin{bmatrix} \dot{y}_a^* \\ z_a^* \end{bmatrix}. \tag{2.16}$$

The control objective is to identify the sequence of control inputs $\mathbf{u}$ by using the above return map to asymptotically converge to the desired apex state, X^*.

There are two main control parameters that are common to all controllers based on the SLIP model; the touchdown leg angle θ_{td} and the amount of change in the total mechanical energy ΔE. The implementation of the control of touchdown leg angle, θ_{td}, is relatively simple since the only goal is to make sure that the leg reaches the desired leg angle at the instant of touchdown. In contrast, energy control of SLIP hopping can be achieved with a variety of different control inputs [1, 10, 15, 44]. In the course of this thesis, we will mainly focus on the two following control methods.

Leg Length Control - LLC : This control mode assumes the presence of a second actuator which controls the length of the robot leg during flight. This mode is generally used in the systems which assume fixed leg stiffness during the locomotion. The main principle in this control mode is to control the total mechanical energy in the system through leg length, meaning leg compression. For instance, it is possible to inject energy into the system by choosing touchdown leg length smaller than the liftoff leg length. This way, the stored energy in the leg spring can be transferred to the system in an effective way. The BowLeg hopping robot and the ParkourBot are example physical robot platforms to use this principle [44, 45, 54].

Torque Actuated Control - TAC : Most legged robot platforms, including the Scout family of quadrupeds [55], the RHex hexapod [20] as well as a number of monopedal platforms [41, 56] use only a single, rotary actuator for each leg (see Fig. 2.2 for an illustration) making it impossible to use

LLC type control modes. In this type of robots, a torque actuation at the hip is used to control the mechanical energy in the system. For example, the TD-SLIP model of [1] and the CT-SLIP model of [57] use only torque actuation to inject energy into the system.

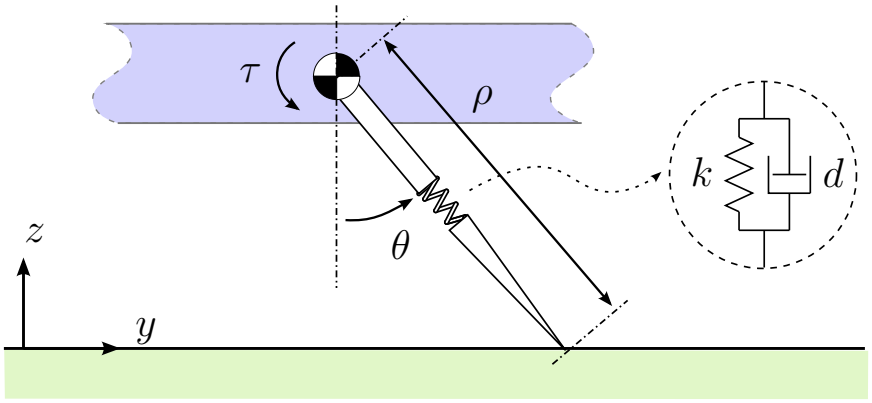


Figure 2.2: A torque actuated SLIP model with a single rotary actuator at the hip (reprinted with permission from [1]).

Note that, there are several other ways of controlling the total mechanical energy in a SLIP-like system such as the Leg Stiffness Control (LSC) and Two-Phase Stiffness Control (TPSC). However, LLC and TAC are the two control modes used in the SLIP and TD-SLIP models on which we build our adaptive control strategy. The following section describes the analytical approximate return maps of these models derived in [14] and [1], respectively.

2.2 Analytical Approximate Maps for SLIP and TD-SLIP Models

Before full dynamic analysis of the SLIP locomotion became available, previous research implemented simple controllers based on intuition [58] or the interpolation of previously observed gaits [59]. Even though these methods exhibit good stabilization performance, their implementation was expensive and they lacked tracking accuracy. It is clear that we need a better, analytical understanding of the return map in (2.14) to design high performance controllers for SLIP locomotion. As mentioned in Section 2.1.3, analytic solutions to the flight dynamics of the SLIP model are easy to obtain but stance dynamics under the effect of gravity are non-integrable [52]. Therefore, the best solution is to use analytical approximate maps for the stance phase. Currently, there are several available analytical approximate stance maps that support the analysis of SLIP and SLIP-like locomotion and the design of associated controllers in literature [10, 12, 13, 15, 53].

The general idea behind approximate stance maps is the estimation of the next apex state, X_{n+1}, from the current apex state, X_n, by using the chosen control input $\mathbf{u}$. The following sections will discuss the approximate analytical map for the SLIP and TD-SLIP models by Ankarali et. al. [1, 14]. The reason why we choose Ankarali's approximations in our studies is that they can successfully incorporate the effect of damping which is inevitable in actual robotic platforms.

2.2.1 Approximate Analytic Solutions to Stance Trajectories of the Passive SLIP with Damping

This section briefly reviews the approximation method used in [14]. Recall from (2.10) and (2.11) that the equations of motion for the stance phase of SLIP in polar coordinates are given by

$$\begin{aligned} m\ddot{\rho} &= m\rho\dot{\theta}^2 + k(\rho_o - \rho) - mg\cos(\theta) \\ 0 &= \frac{d}{dt}(m\rho^2\dot{\theta}) + mg\rho\sin\theta. \end{aligned}$$

First of all, a non-dimensionalization is used to eliminate redundant parameters and to provide an efficient way to interpret the approximation results. Redefining time as $\bar{t} := t/\lambda$ where $\lambda := \sqrt{\rho_o/g}$, scaling all distances with the spring rest length ρ_o, dimensionless stance dynamics are given as

$$\ddot{\bar{\rho}} = \bar{\rho}\dot{\bar{\theta}}^2 - \kappa(\bar{\rho} - 1) - c\dot{\bar{\rho}} - \cos(\bar{\theta}), \tag{2.17}$$

$$\ddot{\bar{\theta}} = (-2\dot{\bar{\rho}}\dot{\bar{\theta}} + \sin(\bar{\theta}))/\bar{\rho}. \tag{2.18}$$

Note that $(d/d\bar{t})^n = \lambda^n(d/dt)^n$ and all time derivatives are with respect to the newly defined, scaled time variable.

Rearranging Eq. (2.18) gives a more convenient form for the angular dynamics

$$0 = \frac{d}{d\bar{t}}(\bar{\rho}^2\dot{\bar{\theta}}) - \bar{\rho}\sin\bar{\theta}\ . \tag{2.19}$$

Assumption 1. *If the angular span of the leg $\Delta\bar{\theta}$ is assumed to be sufficiently small, meaning that the leg remains close to the vertical throughout the entire stance phase as in [12], the effect of gravity can be linearized by assuming $\cos\bar{\theta} \approx 1$ and $\sin(\bar{\theta}) \approx 0$.* ■

Under this assumption, Eq. (2.19) simplifies to

$$\frac{d}{d\bar{t}}(\bar{\rho}^2\dot{\bar{\theta}}) = 0\ . \tag{2.20}$$

Now, the resulting conversation of the angular momentum $p_{\bar{\theta}} := \bar{\rho}^2\dot{\bar{\theta}}$ reduces the radial dynamics of (2.17) to

$$\ddot{\bar{\rho}} + c\dot{\bar{\rho}} + \kappa\bar{\rho} - {p_{\bar{\theta}}}^2/\bar{\rho}^3 = -1 + \kappa \; . \tag{2.21}$$

Remark 1. *Assumption 1 may cause a misinterpretation that the angular momentum is conserved during the stance phase which is not true most of the times in real life. This analytical approximate map [14] also considers the effect of gravity to the stance phase to correct the deviations in angular momentum. Since we did not use this compensation in our adaptive control algorithm, we will not give the details of gravity correction. The detailed information about this process can be found in [14, 53].*

Unfortunately, even these reduced dynamics would not result in an analytic solution for the leg length due to the high order terms.

Assumption 2. *If the relative spring compression is assumed to be sufficiently small with $|1 - \bar{\rho}| \ll 1$, the nonlinear term $1/\bar{\rho}^3$ in (2.21) can be approximated by a Taylor series expansion around $\bar{\rho} = 1$ to yield*

$$1/\bar{\rho}^3\big|_{\bar{\rho}=1} \approx 1 - 3\,(\bar{\rho} - 1) + O((\bar{\rho} - 1)^2). \; \blacksquare \tag{2.22}$$

This assumption is valid unless the maximum leg compression exceeds the 75% of the rest length, which is true for must running behaviors. Nevertheless, by using the Taylor series expansion in Eq. (2.22), Eq. (2.21) is simplified to

$$\ddot{\bar{\rho}} + c\dot{\bar{\rho}} + (\kappa + 3p_{\bar{\theta}}^2)\bar{\rho} = -1 + \kappa + 4p_{\bar{\theta}}^2 \; . \tag{2.23}$$

In order to write this equation in the form of a second order ordinary differential equation, which can easily be solved analytically, some new system parameters are defined such as the natural frequency, $\hat{\omega}_0 := \sqrt{\kappa + 3p_{\bar{\theta}}^2}$, the damping ratio,

$\xi := c/(2\hat{\omega}_0)$, the damped frequency, $\omega_d := \hat{\omega}_0\sqrt{1-\xi^2}$ and the forcing term $F := -1 + \kappa + 4p_{\bar{\theta}}^2$. The new form of Eq. (2.23) with newly defined parameters is given below

$$\ddot{\bar{\rho}} + 2\xi\hat{\omega}_0\dot{\bar{\rho}} + \hat{\omega}_0^2\bar{\rho} = F \ . \tag{2.24}$$

Assuming $\xi < 1$, this equation has a solution of the form

$$\bar{\rho}(t) \ = \ e^{-\xi\hat{\omega}_0 t}(A\cos(\omega_d\bar{t}) + B\sin(\omega_d\bar{t})) + F/\hat{\omega}_0^2 \ , \tag{2.25}$$

where A and B are determined by touchdown states as

$$A \ = \ \bar{\rho}_{td} - F/\hat{\omega}_0^2 \ , \tag{2.26}$$

$$B \ = \ (\dot{\bar{\rho}}_{td} + \xi\hat{\omega}_0 A)/\omega_d \ . \tag{2.27}$$

The radial velocity can be derived with a simple differentiation as

$$\dot{\bar{\rho}}(t) = -M\, e^{-\xi\hat{\omega}_0\bar{t}}(\xi\hat{\omega}_0\cos(\omega_d\bar{t}+\phi) + \omega_d\sin(\omega_d\bar{t}+\phi)) \ , \tag{2.28}$$

where $M := \sqrt{A^2+B^2}$ and $\phi := \arctan(-B/A)$. By further manipulations, the simplest form of the radial motion can be derived as

$$\bar{\rho}(\bar{t}) \ = \ M\, e^{-\xi\hat{\omega}_0\bar{t}}\cos(\omega_d\bar{t}+\phi) + F/\hat{\omega}_0^2 \ , \tag{2.29}$$

$$\dot{\bar{\rho}}(\bar{t}) \ = \ -M\hat{\omega}_0\, e^{-\xi\hat{\omega}_0\bar{t}}\cos(\omega_d\bar{t}+\phi+\phi_2) \ . \tag{2.30}$$

where $\phi_2 := \arctan(-\sqrt{1-\xi^2}/\xi)$.

Equations (2.29) and (2.30) gives an analytical approximation to the radial trajectory of the SLIP model. To derive an analytic solution to the angular trajectory, we use the constancy of angular momentum which is a result of Assumption 1, $\dot{\bar{\theta}} = p_{\bar{\theta}}/\bar{\rho}^2$. Based on Assumption 2, $1/\bar{\rho}^2$ term can be linearized around $\bar{\rho} = 1$ to yield

$$1/\bar{\rho}^2\big|_{\bar{\rho}=1} = 1 - 2(\bar{\rho}-1) + O((\bar{\rho}-1)^2) \ , \tag{2.31}$$

with which the analytical solution for the angular velocity of the leg can be derived as

$$\dot{\bar{\theta}}(\bar{t}) = 3p_{\bar{\theta}} - 2p_{\bar{\theta}}F/\hat{\omega}_0^2 - 2p_{\bar{\theta}}Me^{-\xi\hat{\omega}_0\bar{t}}\cos(\omega_d\bar{t}+\phi) \ . \tag{2.32}$$

Then, a simple integration yields an analytical solution for the angular trajectory of the leg as

$$\bar{\theta}(\bar{\rho}) = \bar{\theta}_{td} + X\,\bar{t} + Y(e^{-\xi\hat{\omega}_0\bar{t}}\cos(\omega_d\bar{t} + \phi - \phi_2) - \cos(\phi - \phi_2)) \ ,$$

where $X := 3p_{\bar{\theta}} - 2p_{\bar{\theta}}F/\hat{\omega}_0^2$ and $Y := 2p_{\bar{\theta}}M/\hat{\omega}_0$.

Although the Equations (2.29),(2.30),(2.33) and (2.32) gives the approximate analytical solutions to the stance trajectories of the lossy SLIP model, we still need to solve for the times of bottom and liftoff events to complete the apex return map.

The bottom by definition is the state where the leg reaches its maximum compression during the stance phase which is analytically described as $\dot{\bar{\rho}}(\bar{t}_b) = 0$. Using (2.30), the bottom time can be found as

$$\bar{t}_b = (\pi/2 - \phi - \phi_2)/\omega_d \ . \tag{2.33}$$

The analytical formulation of liftoff time is more complex than the bottom time. By definition, liftoff occurs when the leg looses contact with the ground. For a lossless SLIP with $\xi = 0$, liftoff can be defined as $\bar{\rho}(\bar{t}_{lo}) = \bar{\rho}_{lo}$, which can easily be solved using (2.29). However, in the presence of damping, the liftoff does not only depend on leg length but also depends on the ground reaction force on the toe. Consideration of both the leg length and the ground reaction forces results in two different liftoff conditions.

First condition represents the case when the net force exerted on the toe by spring-mass pair vanishes which can analytically be expressed as

$$\kappa(1 - \bar{\rho}(\bar{t}_{lo}^{c1})) - c\ \dot{\bar{\rho}}(\bar{t}_{lo}^{c1}) = 0 \ . \tag{2.34}$$

Alternatively, another liftoff condition occurs when the leg reaches the desired leg length during its locomotion from decompression to ascent phase which can be derived by using the equation

$$\bar{\rho}(\bar{t}_{lo}^{c2}) = \bar{\rho}_{lo} \ . \tag{2.35}$$

Using (2.34) and (2.35), the actual liftoff time can be found as $\bar{t}_{lo} = \min(\bar{t}_{lo}^{c1},\ \bar{t}_{lo}^{c2})$. Fig. 2.3 illustrates both of these possible liftoff conditions together with the state transitions.

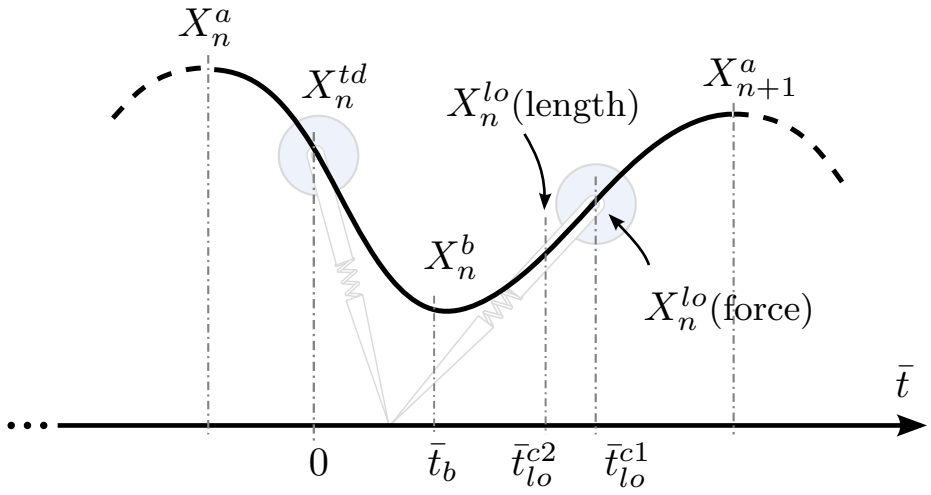


Figure 2.3: An illustration of two possible liftoff conditions based on the force condition of (2.34) and the length condition of (2.35).

Unfortunately, exact analytical solutions of neither (2.34) nor (2.35) is possible. Therefore, a new approximation method is proposed for the exponential term in (2.29) with its value at a specific instant during compression as $e^{-\xi\hat{\omega}_0\bar{t}} \approx e^{-\xi\hat{\omega}_0\gamma\bar{t}_b}$, with $\gamma \geq 1$. Assuming that compression and decompression phases last roughly equal times, $\gamma = 2$ is used in these derivations. By choosing such parameter configurations, the solutions for both conditions are obtained as

$$\bar{t}_{lo}^{c1} \approx (2\pi - \arccos(\kappa(1 - F/\hat{\omega}_0^2)/(\overline{M}Me^{-\xi\hat{\omega}_0\gamma\bar{t}_b})) - \phi - \phi_3)/\omega_d, \quad (2.36)$$

$$\bar{t}_{lo}^{c2} \approx (2\pi - \arccos((\bar{\rho}_l - F/\hat{\omega}_0^2)/(Me^{-\xi\hat{\omega}_0\gamma\bar{t}_b})) - \phi)/\omega_d, \quad (2.37)$$

where we define

$$\overline{M} := \sqrt{(c\hat{\omega}_0)^2 + \kappa^2 - 2\kappa c\hat{\omega}_0\cos(\phi_2)} \quad (2.38)$$

$$\phi_3 := \arctan(\frac{c\hat{\omega}_0\sin(\phi_2)}{c\hat{\omega}_0\cos(\phi_2) - \kappa}). \quad (2.39)$$

Combining the time instants associated with each event with the previously derived radial and angular trajectories, the analytical approximate return map is completed.

2.2.2 Approximate Analytical Return Map for the Torque-Actuated Dissipative SLIP (TD-SLIP) Model

This section briefly reviews the approximation method used in [1] based on the method explained in Section 2.2.1.

The system dynamics of the unforced ($\tau = 0$) TD-SLIP model is equal to the SLIP model explained in Section 2.2.1 except that the TD-SLIP model has no control over the leg lengths. Therefore the solutions for radial and angular trajectories also apply in here.

In the case of forced TD-SLIP model, the applied torque effects the angular momentum which was assumed to be constant as a consequence of Assumption 1. This stance map approximation takes the effect of hip torque into account on the angular momentum and corrects the momentum value at the end of the stance phase. Since the effect on angular momentum depends on the applied torque, we consider a ramp torque profile who has a well-known and simple analytic formulation given below

$$\tau(t) \quad = \quad \begin{cases} \tau_0(1 - \frac{t}{t_f}) & \text{if } 0 \leq t \leq t_f \\ 0 & \text{if } t > t_f \end{cases} \tag{2.40}$$

where τ_0 and t_f chosen prior to touchdown. This profile has three main advantages as listed below

- It is easy to incorporate the effect of ramp torque on the unforced TD-SLIP map due to its simple functional dependence on time.
- Choosing t_f to be the predicted liftoff time results in $\tau(t_{lo}) = 0$ which prevents premature liftoff due to the actuation of the hip.
- As a consequence of the unidirectional action of the hip torque profile, no negative work is done in the sake of locomotion efficiency.

Normally, the direct integration of angular dynamics yields the instantaneous angular momentum around the toe during stance as

$$p_\theta(t) = p_\theta(0) + \int_0^t \tau(\eta)d\eta + \int_0^t mg\rho(\eta)\sin\theta(\eta)d\eta. \tag{2.41}$$

Note that, the inspection of the TD-SLIP dynamics in [1] shows that the effect of hip torque on the angular dynamics is much more dominant to its effect on radial motion. Therefore, an average correction on the angular momentum can capture the effect of hip torque on system trajectories. In [1], Ankarali suggests an update strategy to the constant angular momentum p_θ of Section 2.2.1 which is computed as

$$\hat{p_\theta} = p_\theta(0) + \Delta p_\tau + \Delta p_g \ , \tag{2.42}$$

where Δp_τ and Δp_g represents the time averaged effects of the leg torque and gravitational acceleration.

By choosing $t_f = t_{lo}$ in the hip torque definition of (2.40), we get a very simple solution for the torque correction term as shown below

$$\Delta p_\tau := \frac{1}{t_{lo}} \int_0^{t_{lo}} \left(\int_0^{\eta_1} \tau(\eta_2)d\eta_2 \right) d\eta_1 = \tau_0 \frac{t_{lo}}{3} \ . \tag{2.43}$$

Unfortunately, it is not possible to derive an analytic expression for the gravity correction term Δp_g. Therefore, a linear approximation to the integrand $\rho(\eta)\sin\theta(\eta)$ is used to obtain the following solution

$$\Delta p_g := \frac{mgt_{lo}}{6}(2\rho_o \sin\theta_{td} + \rho_{lo}\sin\theta_{lo}) \ . \tag{2.44}$$

Section 2.2.1 details the derivation of the estimated liftoff time t_{lo}. At this point, the approximate analytical return map for the forced TD-SLIP model can be obtained by substituting $\hat{p_\theta}$ for the constant angular momentum term in all derivations. The important thing here is to notice that this process has an iterative nature since (2.43) and (2.44) depends on the previous estimates of t_{lo} and θ_{lo} respectively. Consequently, starting from the unforced approximation,

more accurate approximations can be obtained by iteratively estimating the new values of angular momentum.

Remark 2. *Actually, this apex return map corrects the angular momentum due to the effect of both the applied torque profile and the effect of gravity in nonsymmetric trajectories. However, in this study we will be only using the correction due to the torque profile since we are not interested in the nonsymmetric locomotion.*

Note that, this chapter focuses more on the background of the SLIP model rather than the TD-SLIP model. The reason is that this study was first introduced for the SLIP model and then transferred to TD-SLIP as a transition phase before the implementation on an actual robot platform. Detailed information about the background of the TD-SLIP model can be found in [60].

Chapter 3

ADAPTIVE CONTROL OF A SPRING-MASS HOPPER

This chapter concerns the design of model-based adaptive control methods for running with planar spring-mass hopper models. In contrast to previous controllers in the literature, the proposed adaptive control algorithm in this chapter allows high tracking performance and accurate system identification for spring-mass hopper templates even in the presence of inaccurate and possibly time varying leg compliance and damping.

This chapter begins by reviewing the proposed adaptive control method with the necessary theoretical formulation in Section 3.1. We build our algorithm based on the presence of a sufficiently accurate deadbeat controller for the SLIP template. This algorithm was introduced in our previous study via simulations in [61]. Section 3.2 reviews the results of this study together with the details of the deadbeat controller of [14] on the same model. Then, Section 3.3 extends this study to a Torque-Actuated Dissipative Spring-Loaded Inverted Pendulum (TD-SLIP) model, towards an implementation on our actual robot platform. Finally,

Section 3.4 concludes the chapter and discusses the possible extensions of this algorithm to different robot platforms with different characteristics.

3.1 Adaptive Control of Spring-Mass Hopper Template

In Section 2.1.3, the control objective of the SLIP model was defined as identifying a sequence of control inputs $\mathbf{u}$ by using the approximate return map definition of (2.15) to asymptotically converge to the desired apex state, X^*. In the presence of a sufficiently accurate system model, gait control of the SLIP model can be achieved through a deadbeat strategy as described in [1, 14]. Given a desired apex state X^*, inversion of the apex return map for the z and $\dot{y}$ components of the state yields the controller

$$\mathbf{u} = \hat{\mathbf{f}}_{\hat{\mathbf{p}}}^{-1}(X^*, X_n) \; . \tag{3.1}$$

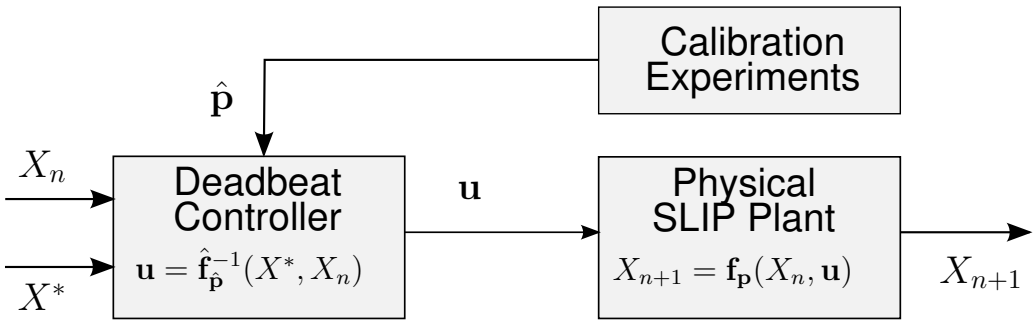


Figure 3.1: Deadbeat SLIP gait control through the inversion of the approximate plant model.

Note, however, that such an approximate return map and hence its inverse must rely on possibly inaccurate parameter estimates $\hat{\mathbf{p}}$ for spring and damping constants. As shown in the block diagram of Fig. 3.1, these estimates are often obtained through calibration experiments on the platform but may not provide sufficiently good accuracy.

The core of our adaptive control algorithm relies on once-per-step corrections to these parameter estimates based on the difference between predicted and measured apex states for each stride. Consequently, we will find it useful to capture the dependence of apex height and velocity coordinates to these parameters through the Jacobian matrices of both the actual and approximate return maps. For both of these models, associated Jacobians are defined as

$$\mathbf{J} \;:=\; \partial\mathbf{f}/\partial\mathbf{p} = \begin{bmatrix} \partial\dot{y}_{n+1}/\partial k & \partial\dot{y}_{n+1}/\partial d \\ \partial z_{n+1}/\partial k & \partial z_{n+1}/\partial d \end{bmatrix}, \tag{3.2}$$

$$\hat{\mathbf{J}} \;:=\; \partial\hat{\mathbf{f}}/\partial\mathbf{p} = \begin{bmatrix} \partial\hat{\dot{y}}_{n+1}/\partial k & \partial\hat{\dot{y}}_{n+1}/\partial d \\ \partial\hat{z}_{n+1}/\partial k & \partial\hat{z}_{n+1}/\partial d \end{bmatrix}, \tag{3.3}$$

where $\mathbf{f}$ and $\hat{\mathbf{f}}$ represents actual and approximate return maps, respectively. We use numerical differentiation to compute both of these Jacobian matrices.

The corrective parameter adjustment strategy we adopt from MRAC method [27, 29] is very similar to how estimation methods such as Kalman filters use innovation on sensory measurement to perform state updates [62].

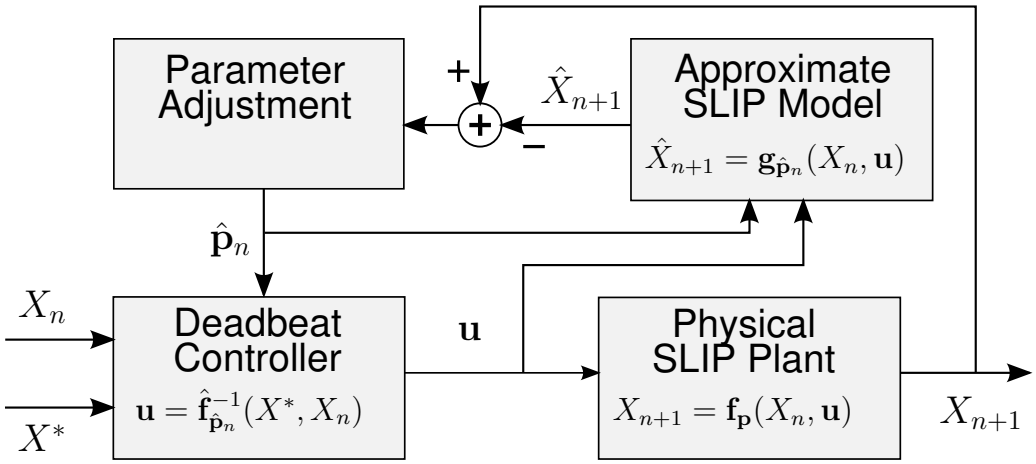


Figure 3.2: The proposed adaptive control strategy. Prediction errors of an approximate plant model $\mathbf{g}$ (computed either using exact plant simulations $\mathbf{f}$ or analytical approximations $\hat{\mathbf{f}}$) are used to dynamically adjust parameter estimates $\hat{\mathbf{p}}_n$.

Fig. 3.2 illustrates the block diagram for the adaptive parameter correction scheme we propose in this study. Our method relies on the availability of an

approximate return map $\mathbf{g}$ that can predict the apex state outcome of a single step, given the apex states of the previous step X_n and associated control inputs $\mathbf{u}_n$. In this study, we consider two alternatives for the approximate predictor model:

1. ***Exact SLIP Model (ESM)***: This alternative uses the numeric simulation of the actual SLIP dynamics to predict the outcome of a single-stride. This corresponds to choosing $\mathbf{g} = \mathbf{f}$ in the block diagram of Fig. 3.2.

2. ***Approximate Analytical Solution (AAS)***: This option uses $\mathbf{g} = \hat{\mathbf{f}}$, adopting the approximate analytical solutions of [14] and [1] as a predictor of SLIP trajectories. We also compute the associated Jacobian through numerical differentiation by using these analytical solutions but it would also be possible to analytically compute Jacobians for a more efficient implementation.

As we will show in Section 3.2 and Section 3.3, the first option is useful for accurate identification of the dynamic parameters of the system, whereas the second option will be useful in eliminating steady-state tracking errors for the gait-level control of SLIP running. Note that the first option brings additional computational burden, so the second option which uses the analytical approximate solutions is much more suitable for real-time implementation on a physical platform particularly if analytic Jacobians are used.

Regardless of which predictor is chosen, an apex state prediction error is computed at every step as

$$\mathbf{e} := X_{n+1} - \hat{X}_{n+1} = \mathbf{f}_{\mathbf{p}}(X_n, \mathbf{u}) - \mathbf{g}_{\hat{\mathbf{p}}_n}(X_n, \mathbf{u}) \; . \tag{3.4}$$

Note that the computation of this error requires measurement of actual apex states X_{n+1} at every stride, which can be accomplished through proper instrumentation and state estimation techniques. In Chapter 4, we talk about the

instrumentation and filtering methods we used in our actual robot platform to measure the apex state. More importantly, however, the predictor is expected to use the *updated estimates* of dynamic system parameters $\hat{\mathbf{p}}_n$ rather than their unknown physical values experienced by SLIP plant, making it relevant in computing corrections on these parameters.

The goal of our adaptive control approach is to bring the steady-state value of this prediction error to zero. In other words, we seek to have

$$\lim_{n\to\infty}(\hat{X}_n - X_n) \;=\; 0\,, \tag{3.5}$$

which will also indirectly yield steady-state parameter estimates as

$$\hat{\mathbf{p}} = \lim_{n\to\infty}(\hat{\mathbf{p}}_n)\,. \tag{3.6}$$

We accomplish both of these goals using a conceptually simple yet effective parameter adjustment strategy based on the Jacobians defined in (3.2) and (3.3). By definition, these Jacobian matrices relate infinitesimal changes in the apex state predictions to infinitesimal changes in the dynamic system parameters with

$$\delta\hat{X}_{n+1} = (\partial\mathbf{g}/\partial\mathbf{p})\big|_{X_n}\delta\mathbf{p}\,. \tag{3.7}$$

Based on this relation and the prediction errors computed at every stride, we propose the parameter update strategy

$$\hat{\mathbf{p}}_{n+1} = \hat{\mathbf{p}}_n + K_{\mathbf{e}}\,(\,\partial\mathbf{g}/\partial\mathbf{p}\,)^{-1}\,\big|_{X_n}\mathbf{e} \tag{3.8}$$

where $K_{\mathbf{e}} < 1$ is a gain coefficient that can be used to tune convergence and prevent oscillatory behavior. This yields an on-line adaptation mechanism that can be used for both predictor choices, with the ESM choice resulting in accurate system identification and the AAS choice yielding adaptive gait control as we will show in following sections. It is important to note that practical applicability of our adaptive control method inevitably depends on the accuracy of the underlying SLIP model. Even though the linear spring model we used in this study was

previously shown to result in reasonable predictive accuracy for biological runners [10], extensions to the model and the associated analytical approximations may be needed for systems with more complex, nonlinear springs.

3.2 Adaptive Control of the SLIP Model

This section discusses the application of the adaptive control algorithm explained in Section 3.1 to the SLIP model presented in [14]. We will first briefly summarize the deadbeat controller of [14] and then present the results of our adaptive control study including the comparison with this nonadaptive approach.

3.2.1 Deadbeat Control of the SLIP Model with Damping

In this section, we review the design of the deadbeat controller of [14] to regulate and stabilize the progression of the apex states of a spring-mass hopper through the analytically formulated approximate return map described in Section 2.2.1.

The fundamental control problem in this study is to derive the appropriate control inputs $\mathbf{u} := [\bar{\theta}_{td}, \bar{\rho}_{td}, \bar{\rho}_{lo}]$ to satisfy

$$X^* = \mathbf{f_p}(X, \mathbf{u}) \ , \tag{3.9}$$

where X and X^* represent the current and desired apex states, respectively.

Normally, Equation (3.1) suggests that these control inputs can be easily obtained through the inversion of the apex return map when you have an integrable analytical approximation. However, in the case of this SLIP model, the associated map involves three coupled variables. Therefore, to simplify the solution of the controllers, they first eliminate the cyclic variable *horizontal position*, $\bar{y}$, from the domain of the controller since the algorithm is primarily interested in sustained, steady-state locomotion. Then, only the apex height $\bar{z}$ and the apex

speed $\bar{\dot{y}}$ remain as variables of interest. However, the solution of the resulting equation with these two coupled variables is not still simple enough, so it requires an iterative procedure.

At the beginning of the iterative approach, they assume that no damping is present in the system and solve the energy balance equation

$$\kappa(\bar{\rho}_{td} - 1)^2 - \kappa(\bar{\rho}_{lo} - 1)^2 = E(\bar{z}^*, \bar{\dot{y}}^*) - E(\bar{z}, \bar{\dot{y}}) \tag{3.10}$$

to find the control inputs $\bar{\rho}_{td}$ and $\bar{\rho}_{lo}$. Note that there is only one unknown in the above equation since either of these control inputs equal to the rest length in dimensionless units when the desired energy change is negative or positive, respectively.

After determining these two control inputs, (3.9) reduces to a one-dimensional equation, whose solution can be formulated as a minimization problem with

$$\bar{\theta}_{td} = \underset{\frac{-\pi}{2} < \bar{\theta} < \frac{-\pi}{2}}{\operatorname{argmin}} \; (\bar{\dot{y}}^* - (\, \pi_{\bar{\dot{y}}} \circ \mathbf{f}(X, \bar{\theta}, \bar{\rho}_{td}, \bar{\rho}_{lo})\,))^2 \,, \tag{3.11}$$

which can be solved numerically since it is a one-dimensional monotonic function [14]. Note that these control inputs are derived for a lossless SLIP system. We need to take the damping losses into account to get better estimates for the control inputs. Therefore, we first estimate the damping losses during a single stride as

$$E_c := \int_0^{\bar{t}_{lo}} c\, \dot{\bar{\rho}}^2(\bar{t}) d\bar{t}. \tag{3.12}$$

The details of how this integration is computed can be found in [14]. Now, we solve the complete energy balance equation to find the new control inputs $\bar{\rho}_{td}$ and $\bar{\rho}_{lo}$. Then, by using these new estimates, we can obtain a new solution for the touchdown angle through (3.11), which now takes the damping into account as well.

3.2.2 Simulation Environment and Performance Criteria

The two related but different goals of our adaptive control are the estimation of unknown or miscalibrated dynamic system parameters and accurate tracking of desired apex states. Both of these goals can be defined as a function of the steady-state behavior of the system. Consequently, we define three different percentage error measures

$$SSE_k := 100 \lim_{n\to\infty} (|\hat{k}_n - k|/k) , \tag{3.13}$$

$$SSE_d := 100 \lim_{n\to\infty} (|\hat{d}_n - d|/d) , \tag{3.14}$$

$$SSE_a := 100 \lim_{n\to\infty} (|| X_n - X^*||/|| X^*||) , \tag{3.15}$$

with SSE_k and SSE_d capturing system identification performance and SSE_a characterizing the tracking performance of the adaptive controller.

Table 3.1: Simulation Apex Goal and Parameter Ranges for the SLIP Model

z_a^* (m)	$\dot{y}_a^*$ (m/s)	k (N/m)	d (Ns/m)	m (kg)
[1.25, 1.75]	[1.25, 2.75]	[800, 2000]	[3, 15]	1

In order to characterize the performance of our adaptive control strategy, we ran a large number of simulations using different apex goal settings X^* as well as different choices of dynamic parameters $\mathbf{p}$ within ranges specified in Table 3.1, chosen to be consistent with biomechanics literature [63] as well as existing legged robots [20] to increase the relevance of our results.

The hybrid SLIP plant dynamics in Figures 3.1 and 3.2 were simulated in Matlab using a fourth-order, adaptive time-step Runge-Kutta integrator with exact detection of touchdown and liftoff events. Simulations were run until steady-state was reached with a tolerance of 10^{-4} in the norm of the apex state. Steady-state trajectories were found to be independent of initial apex states. However, since the convergence behavior of (3.8) depends on the choice of the predictor and the update gain, we will consider different initial parameter estimates $\mathbf{p}_0$ for our simulations.

3.2.3 Accurate Control with the AAS Predictor

In this section, we present apex goal tracking simulations for the SLIP model with the AAS predictor introduced in Section 3.1. Before we proceed with more systematic performance results, however, Fig. 3.3 illustrates an example SLIP simulation started with a nonadaptive controller in the presence of 20% errors for the estimates of both spring and damping constants, with the subsequent activation of our adaptive controller using the AAS predictor around $t = 2s$, finally followed by a step change in the apex goal around $t = 4.55s$.

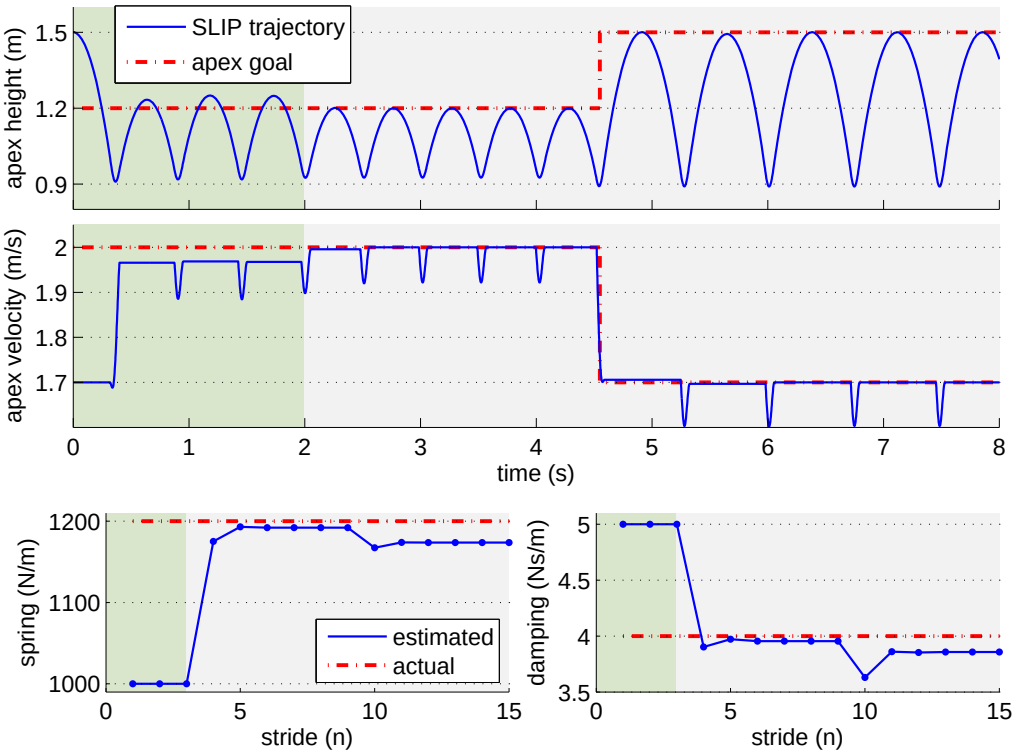


Figure 3.3: An example SLIP simulation started with a non-adaptive controller (dark shaded region) and 20% error in both the spring and damping constants. Our adaptive controller with the AAS predictor was started around $t = 2s$ and a step change in the apex goal was given around $t = 4.55s$.

As expected, using the non-adaptive controller with miscalibrated dynamic parameters results in a substantial steady-state error due to prediction errors in the analytic approximations of [14]. When the adaptive controller is switched on around $t = 2s$, this error is quickly eliminated and estimated values of both the

spring and damping constants quickly converge towards their physical values as shown in Fig. 3.3. The last five steps of the simulation shows that steady-state tracking remains accurate even when a step input with a large magnitude is given to the system.

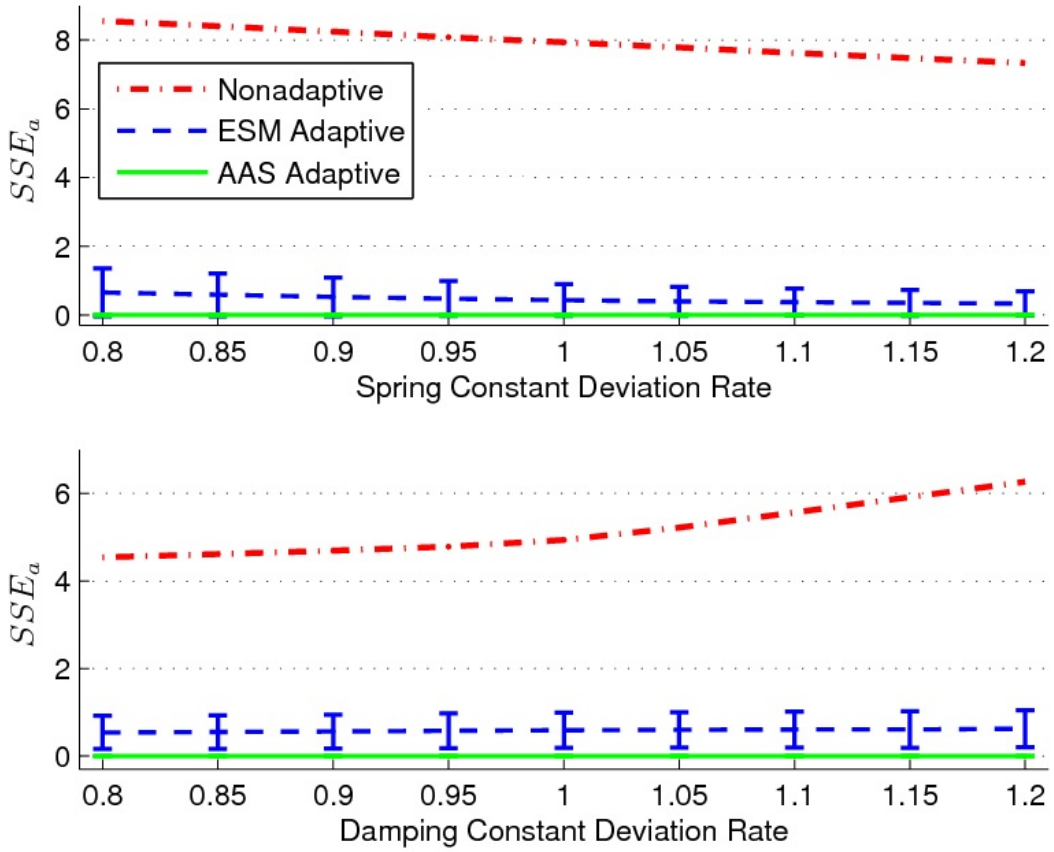


Figure 3.4: Steady-state apex goal tracking errors for the non-adaptive, AAS adaptive and ESM adaptive controllers for the SLIP model. Error measures were averaged across 321489 simulation runs with different goals and initial parameter estimates. Vertical bars show standard deviations and were omitted for the non-adaptive case since they were very large.

More generally, Fig. 3.4 illustrates the average tracking performance of our adaptive controller across the range of apex goals and parameter choices given in Table 3.1 corresponding to 321489 simulation runs. The top and bottom plots respectively show the dependence of average errors and their standard deviations on the initial deviations of the spring and damping constants for a non-adaptive controller as well as our adaptive controller with both the AAS and ESM predictors. As expected, the non-adaptive controller results in large tracking errors

(with very high standard deviations, omitted from the figure for clarity) whereas the *AAS* Adaptive controller reduces the steady state error to zero. Average apex tracking and parameter estimation errors and their standard deviations across all simulations are also given in Table 3.2.

It may be surprising that the AAS predictor outperforms the ESM predictor based on the exact SLIP model for apex goal tracking. However, note that the deadbeat controller of (3.1) is based on the inversion of the AAS analytic approximations. Naturally, when dynamic system parameters are adapted such that the predictions of there approximations are error-free, the resulting controller achieves zero tracking error. This result is expected because our adaptive controller tries to ensure that

$$\hat{\mathbf{f}}_{\hat{\mathbf{p}}}(X_n, \mathbf{u}_n) \longrightarrow \mathbf{f}_{\mathbf{p}}(X_n, \mathbf{u}_n). \tag{3.16}$$

Since both the deadbeat and the adaptive controller uses the same approximate return map, the actual system plant converges to the desired apex state as formalized below

$$\mathbf{f}_{\mathbf{p}}(X_n, \hat{\mathbf{f}}^{-1}_{\hat{\mathbf{p}}}(X^*, X_n)) \longrightarrow X^*. \tag{3.17}$$

In contrast, while the ESM predictor can accurately estimate the dynamic parameters as shown in Section 3.2.4, some prediction errors still remain, leading to the small steady-state tracking errors of Fig. 3.4.

Table 3.2: Percentage Apex Tracking and Parameter Estimation Errors for the SLIP Model

Error Measure:	SSE_a	SSE_k	SSE_d
Non-adaptive	6.56 ± 4.64	10 ± 6.20	10 ± 6.20
AAS Adaptive	0.002 ± 0.001	2.34 ± 1.45	5.53 ± 2.81
ESM Adaptive	0.52 ± 0.45	0.0008 ± 0.0005	0.007 ± 0.005

In addition, Fig. 3.5 shows a comparison of the dynamic tracking performance for the non-adaptive controller and our adaptive controller with the *AAS* predictor. Once again, our controller quickly converges to the desired trajectory,

outperforming the non-adaptive controller which suffers from miscalibrated parameter estimates. These results show that the proposed controller can maintain accurate tracking even for dynamic goal settings and not just for a single static target.

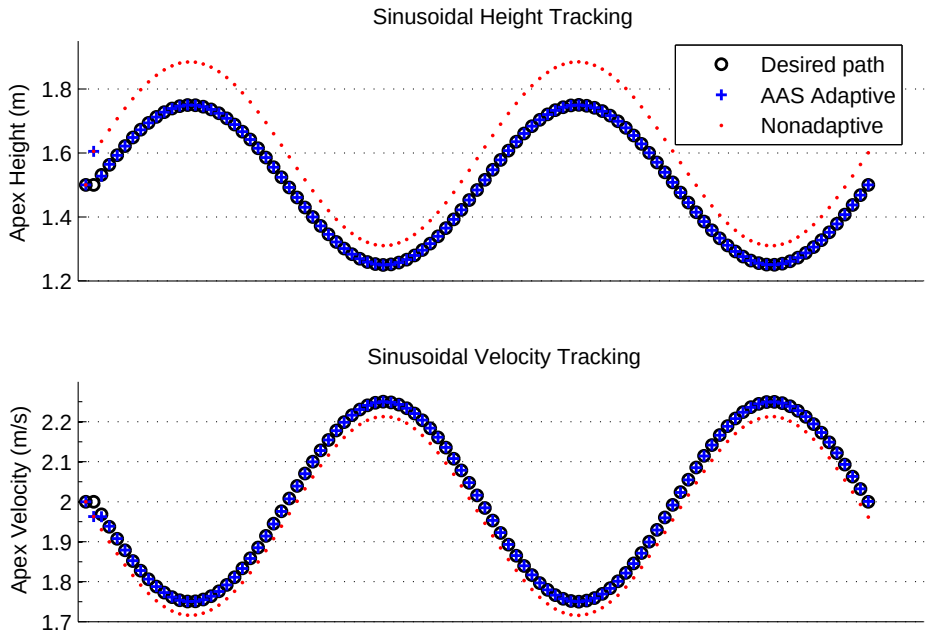


Figure 3.5: Apex height (top) and speed (bottom) tracking performance for a sinusoidal reference trajectory for the SLIP model started with a 20% error in both the spring and damping constants. Each data point corresponds to a single apex event.

Finally, Fig. 3.6 shows a scenario where the robot leg faces with an unexpected breakage during its locomotion, effecting values of the spring and damping constants. In Fig. 3.3, we showed the robustness of our adaptive control algorithm to the step changes in the desired goal settings. In contrast, this scenario aims to show the robustness of our algorithm to step changes in the leg spring and damping constants during locomotion in comparison to a non-adaptive approach. Both the adaptive and non-adaptive control tests start with a 5% error in both spring and damping constants in the dark shaded region. It can be seen

in this region that the adaptive controller follows the desired trajectory accurately and the non-adaptive controller tracks the desired path with a constant steady-state error which can be negligible in most cases. However, at $t = 2.25s$ a sudden breakage occurs in the robot leg, a very likely occurrence in legged robot platforms since they are mostly designed for rough-terrain applications in real environments. Due to this event, the estimation error in both spring and damping constants increases to 20%, which causes a high steady-state tracking error for the non-adaptive controller. In contrast, the adaptive controller with the AAS predictor quickly compensates this error with the help of its on-line estimation capability of leg compliance and damping.

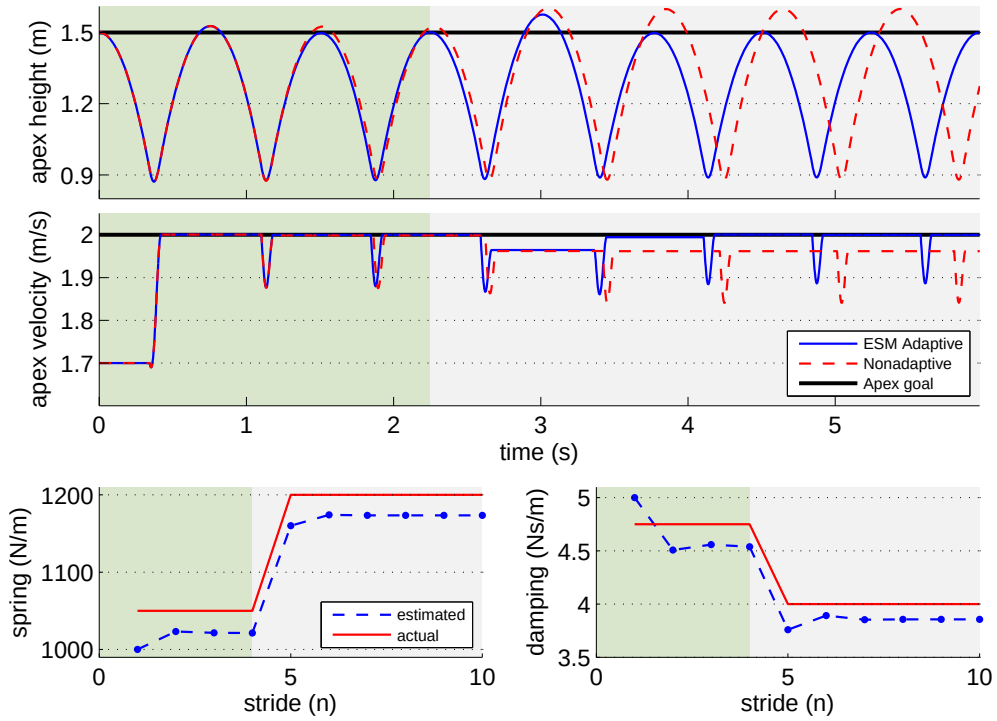


Figure 3.6: Two example SLIP simulations started with 5% error in both spring and damping constants (dark shaded region). One of them starts with a non-adaptive controller and the other uses our adaptive controller with the AAS predictor. An unexpected breakage occurs in the robot leg about $t = 2.25s$ and it increase the estimation error in both spring and damping constants to 20% instantaneously.

3.2.4 System Identification with the ESM Predictor

In this section, we present the system identification performance of our algorithm with the ESM predictor. Fig. 3.7 shows an example SLIP simulation similar to the example of the previous section, but with the ESM predictor instead. Once again, the first three steps were controlled with the non-adaptive strategy, activating the adaptive controller at $t = 2s$ and finally initiating a step change in the apex goal setting at $t = 4.55s$.

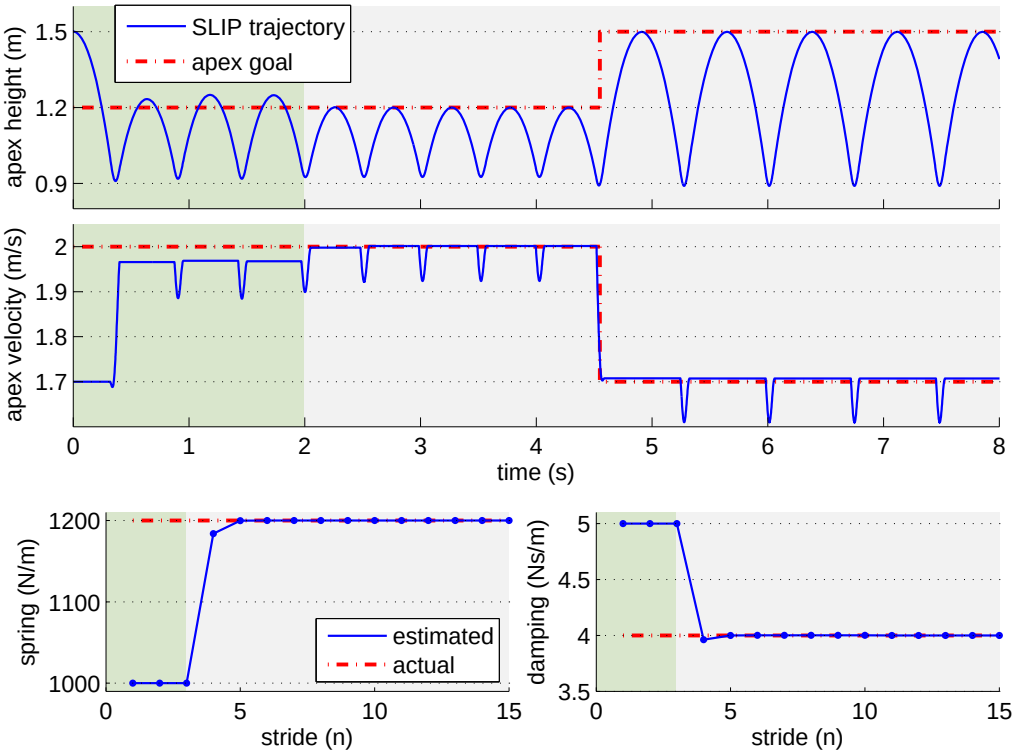


Figure 3.7: An example SLIP simulation started with a non-adaptive controller (dark shaded region) and 20% error in both the spring and damping constants. Our adaptive controller with the ESM predictor was started around $t = 2s$ and a step change in the apex goal was given around $t = 4.55s$.

In contrast to the AAS predictor, the use of the ESM predictor allows better estimation of unknown dynamic parameters at the expense of steady-state tracking accuracy. This can be observed in the bottom two plots of Fig. 3.7 as well as the corresponding columns of Table 3.2 showing an increase in the

average apex tracking error. This result is expected since the elimination of prediction errors for the exact SLIP predictor corresponds to exact identification of the unknown dynamic parameters. For a physical robot, this would be the best way to estimate the spring and damping constants as accurately as possible.

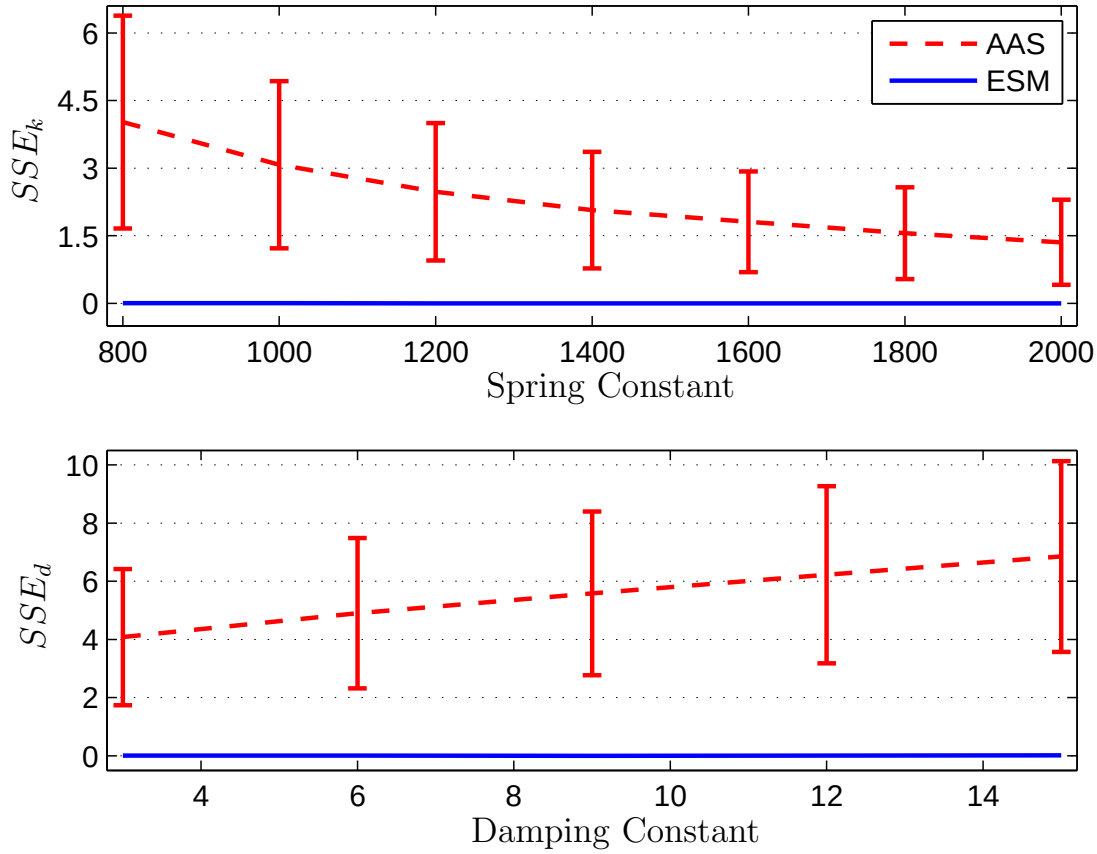


Figure 3.8: Errors in steady-state estimations for the spring (top) and damping (bottom) constants using the AAS adaptive and ESM adaptive controllers for the SLIP model. Error measures were averaged across 321489 simulation runs with different goals and initial parameter estimates. Vertical bars show standard deviations.

Following this isolated example, Fig. 3.8 shows the parameter estimation performance of our adaptive method both with the *ESM* and *AAS* predictors across a larger range of apex goal and parameter settings. Since the non-adaptive controller does not update parameter estimates in any way, we have not included it in the error figures. Our results for both the spring and damping constants show that while the *ESM* predictor perfectly estimates system parameters, the

AAS predictor, which is much more practical and computationally feasible for on-line application due to its analytic nature, also performs very well and yields steady-state parameter estimation errors well below the 10-15% that would be expected from manual calibration alone.

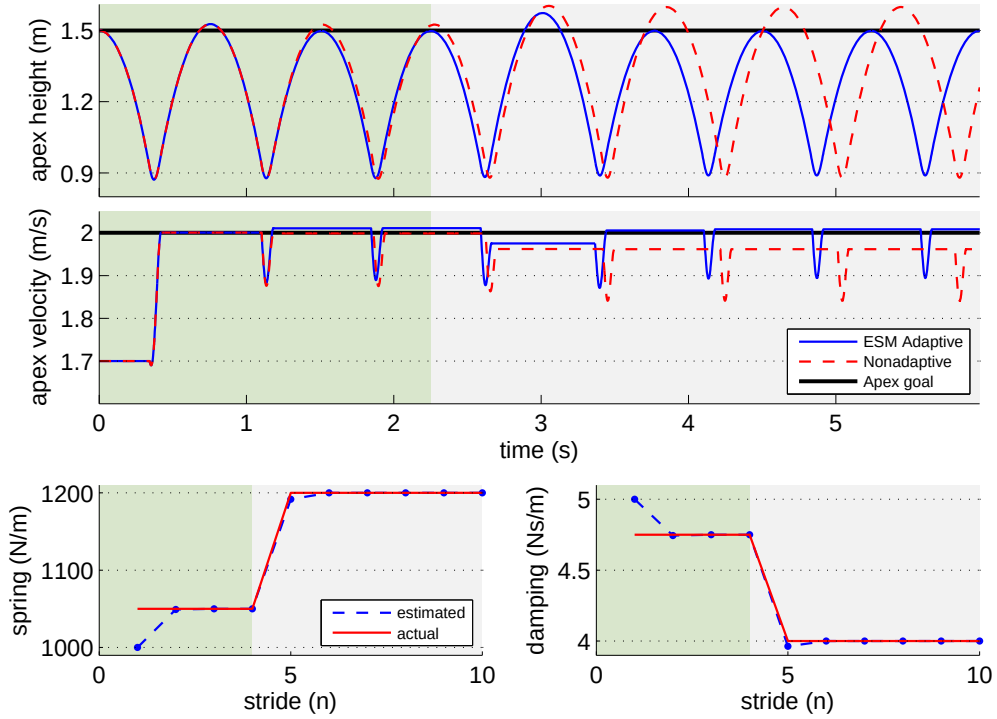


Figure 3.9: Two example SLIP simulations started with 5% error in both spring and damping constants (dark shaded region). One of them starts with a non-adaptive controller and the other uses our adaptive controller with the ESM predictor. An unexpected breakage occurs in the robot leg about $t = 2.25s$ and it increase the estimation error in both spring and damping constants to 20% instantaneously.

As in Section 3.2.3, we design a scenario where the robot leg encounters a breakage problem during its locomotion. We use the same parameter configuration used in the experiment of Fig. 3.6. It can be clearly seen in Fig. 3.9 that both the adaptive and non-adaptive controllers follows the desired trajectory with a negligible steady-state error. However, after the breakage, while the steady-state error in the non-adaptive controller increases, the adaptive controller with the ESM predictor preserves its small tracking error regardless of the sudden changes in spring and damping constants.

3.3 Adaptive Control of the TD-SLIP Model

This section discuss the application of the adaptive control algorithm explained in Section 3.1 to the torque-actuated SLIP (TD-SLIP) model presented in [1]. We will first summarize the deadbeat controller of [1] and then present results with our adaptive control algorithm, including a comparison with the nonadaptive approach.

3.3.1 Deadbeat Control of the TD-SLIP Model

In this part, we briefly summarize the design of the deadbeat controller in [1] to regulate and stabilize the progression of the apex states of a torque-actuated spring-mass hopper through the analytically formulated approximate return map derived in Section 2.2.2. There are a number of studies [15, 64, 65] that inverts the analytic approximate return map to obtain a deadbeat controller for actively stabilizing the system around a desired operating point X^*. Our controller adopts this approach to derive a deadbeat controller for TD-SLIP.

The control problem in this study is to derive the appropriate control inputs $\mathbf{u} := [\tau, \theta]$ to satisfy

$$X^* = \mathbf{f}_p(X, \mathbf{u}), \tag{3.18}$$

where X and X^* represent the current and desired apex states respectively.

In the case of an explicit desired apex state, the system will require no change in the energy level. However, in the case of a dynamic goal setting, the requested energy to reach a desired apex state should be supplied by the hip torque. The total energy dissipated within a single TD-SLIP step is given as

$$E_{loss} = E_d + E_k, \tag{3.19}$$

where E_d represents damping losses with

$$E_d := \int_0^{t_{lo}} d\dot{\rho}^2(\eta)\ d\eta\ , \tag{3.20}$$

and E_k represents the leftover energy in the leg spring when it liftoffs before it is fully extended due to damping with

$$E_k := (\rho_{lo} - \rho_o)^2/2. \tag{3.21}$$

Then, the total energy requested can be defined as

$$E_\tau = \frac{1}{2}m((\dot{y}_a^*)^2 - \dot{y}_a^2) + mg(z_a^* - z_a) + E_{loss}, \tag{3.22}$$

which should be equal to the E_τ, energy supplied by the hip torque. See [1] for details.

After determining the desired torque profile, the only remaining control variable is the touchdown leg angle, θ_{td} which can be obtained through a one-dimensional optimization problem as

$$\theta_{td} = \underset{\frac{-\pi}{2} < \theta < \frac{-\pi}{2}}{\operatorname{argmin}} \ (\dot{y}_a^* - (\ \pi_{\dot{y}_a} \circ \mathbf{P}(\theta_{td}, [z_a, \dot{y}_a]_k)\ ))^2\ , \tag{3.23}$$

whose numerical solution is trivial since we have an analytic approximation of the return map $\mathbf{P}$. This choice of hip torque, τ and touchdown leg angle, θ_{td} yields an effective, one-step deadbeat controller for the TD-SLIP model.

3.3.2 Simulation Environment and Performance Criteria

Similar to Section 3.2.2, our adaptive controller has two different but related goals: the estimation of miscalibrated system parameters and accurate tracking of desired apex states. We define both of these goals as a function of the steady-state behavior of the system. We use the error measures SSE_k and SSE_d defining the system identification performance and SSE_a characterizing the tracking performance whose analytic formulation are given in Section 3.2.2.

In order to characterize the performance of our adaptive control strategy on TD-SLIP, we ran a large number of simulations using different apex goal settings X^* as well as different choices of dynamic parameters $\mathbf{p}$ within ranges specified in Table 3.3, chosen to be consistent with our robot design explained in Chapter 4 and the ranges in [1] for fair comparison.

Table 3.3: Simulation Apex Goal and Parameter Ranges for the TD-SLIP Model

z_a^* (m)	$\dot{y}_a^*$ (m/s)	k (N/m)	d (Ns/m)	m (kg)
[0.25, 0.35]	[1.2, 1.8]	[4200, 5400]	[3, 15]	4

The hybrid TD-SLIP plant dynamics were simulated in Matlab using a fourth-order, adaptive time-step Runge-Kutta integrator with exact detection of touch-down and liftoff events. Similar to Section 3.2.2, simulations were run until steady-state was reached with a tolerance of 10^{-4} in the norm of the apex state and steady-state trajectories were found to be independent of the initial apex states. However, since the convergence behavior of (3.8) depends on the choice of the predictor and the update gain, we will consider different initial parameter estimates $\mathbf{p}_0$ for our simulations up to 20 percent deviations.

3.3.3 Accurate Control with the AAS Predictor

In this section, we extend the apex goal tracking simulations performed for the SLIP model in Section 3.2.3 to the TD-SLIP model with the AAS predictor introduced in Section 3.1. Similar to Section 3.2.3, we give a sample SLIP simulation before proceeding with more systematic performance results. Fig. 3.10 illustrates this simulation started with a nonadaptive controller in the presence of 20% errors for the both spring and damping constant estimates, with the subsequent activation of our adaptive controller using the AAS predictor around $t = 1.4s$, finally followed by a step change in the apex goal around $t = 2.5s$.

As expected, using the nonadaptive controller with miscalibrated dynamic parameters results in a substantial steady-state error due to prediction and approximation errors in the analytic approximation of [1]. Right after the activation of our adaptive controller with the AAS predictor, this error is quickly eliminated and estimated values of both the spring and damping constants quickly converge towards their physical values as shown in Fig. 3.10. Besides, we apply a step change with a large magnitude to the apex goal setting of the system around $t = 2.5s$ and observe that the steady-state tracking still remain accurate.

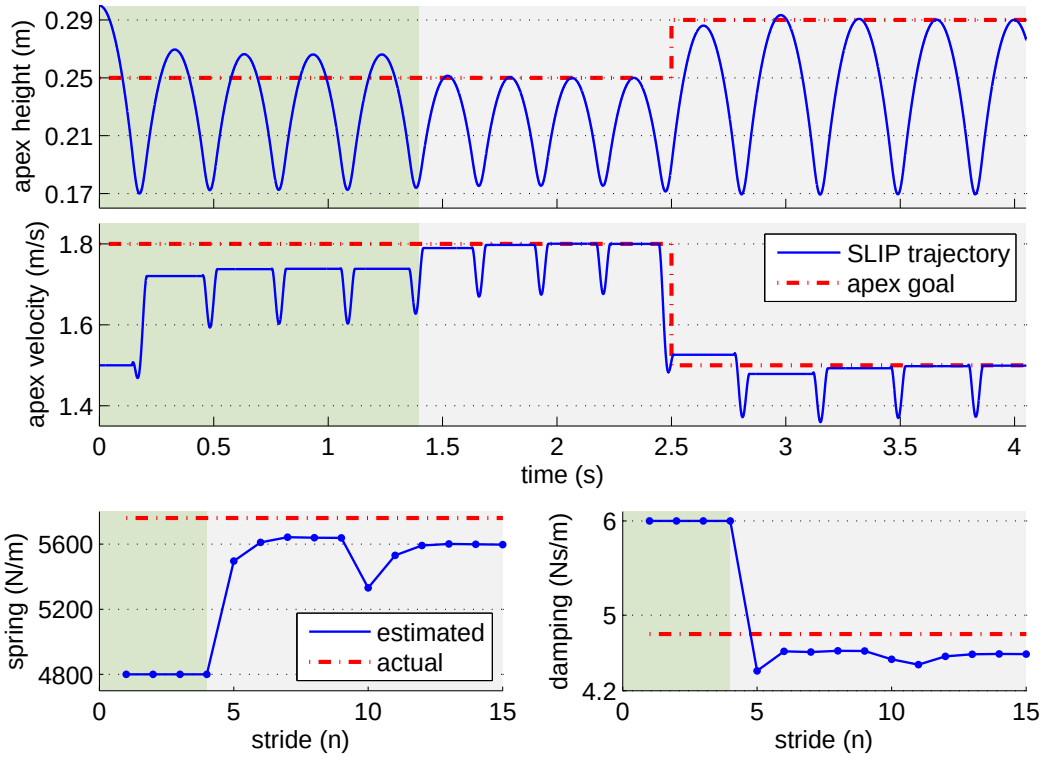


Figure 3.10: An example TD-SLIP simulation started with a non-adaptive controller (dark shaded region) and 20% error in both the spring and damping constants. Our adaptive controller with the *AAS* predictor was started around $t = 1.4s$ and a step change in the apex goal was given around $t = 2.5s$.

More generally, Fig. 3.11 illustrates the average tracking performance of our adaptive controller across the range of apex goals and parameter choices given in Table 3.3 corresponding to 306180 simulation runs. The top and bottom plots respectively show the dependence of average errors and their standard deviations on the initial deviations of the spring and damping constants for a nonadaptive

controller as well as our adaptive controller with both the AAS and ESM predictors. As expected, the nonadaptive controller results in large tracking errors (with very high standard deviations, omitted from the figure for clarity) whereas the AAS Adaptive controller reduces the steady state error to zero (ignoring the negligible numeric computation errors). Unlike SLIP, whose corresponding results are illustrated in Fig. 3.4, the tracking performance of TD-SLIP has less dependence to the deviations in damping constant as shown in the lower plot of Fig. 3.11. This is why the tracking performance of the Nonadaptive and ESM Adaptive methods get closer in the spring constant deviation plot (upper plot) of Fig. 3.11 around 1, corresponding to region without deviation. Average apex tracking and parameter estimation errors and their standard deviations across all simulations are also given in Table 3.4.

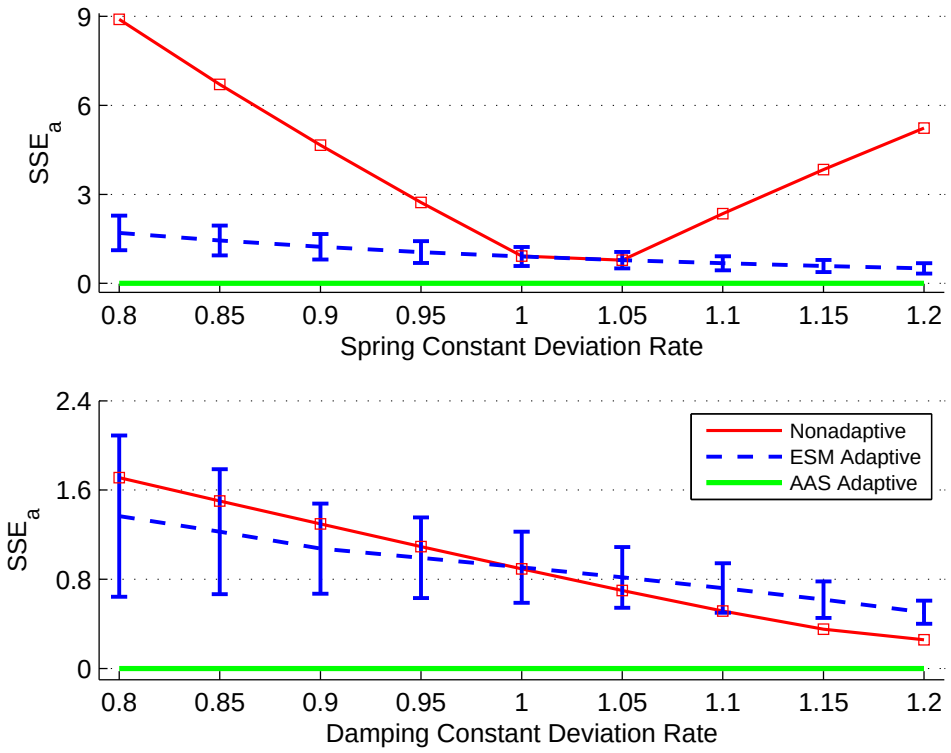


Figure 3.11: Steady-state apex goal tracking errors for the non-adaptive, AAS adaptive and ESM adaptive controllers for the TD-SLIP model. Error measures were averaged across 306180 simulation runs with different goals and initial parameter estimates. Vertical bars show standard deviations and were omitted for the non-adaptive case since they were very large.

Observing Table 3.4, the AAS predictor outperforms the ESM predictor based on the exact TD-SLIP model and the Nonadaptive controller for apex goal tracking. In Section 3.2.3, we analytically showed how the AAS predictor achieves zero tracking error in steady-state for the SLIP model. This result is also valid in TD-SLIP case since both the AAS predictor for the TD-SLIP model and the deadbeat controller of ((3.1)) is based on the inversion of the analytic approximation explained in Section 2.2.2.

In contrast, while the ESM predictor can accurately estimate the dynamic parameters as shown in Section 3.3.4, some prediction errors still remain due to the analytic approximations used for the deadbeat control, leading to the small steady-state tracking errors of Fig. 3.11.

Table 3.4: Percentage Apex Tracking and Parameter Estimation Errors for the TD-SLIP Model

Error Measure:	SSE_a	SSE_k	SSE_d
Non-adaptive	2.47 ± 0.52	10 ± 6.20	10 ± 6.20
AAS Adaptive	0.18 ± 0.12	3.57 ± 1.61	3.89 ± 1.31
ESM Adaptive	0.95 ± 0.35	1.95×10^{-8}	2.48×10^{-8}

Similar to SLIP, we also run a sinusoidal trajectory following experiment to observe the response of our controller to dynamic goal settings. Fig. 3.12 shows a comparison of these tests for the nonadaptive controller and our adaptive controller with the AAS predictor. Once again, our controller quickly converges to the desired trajectory, outperforming the nonadaptive controller which suffers from miscalibrated parameter estimates. These results show that the proposed controller can maintain accurate tracking even for dynamic goal settings and not just for a single static target.

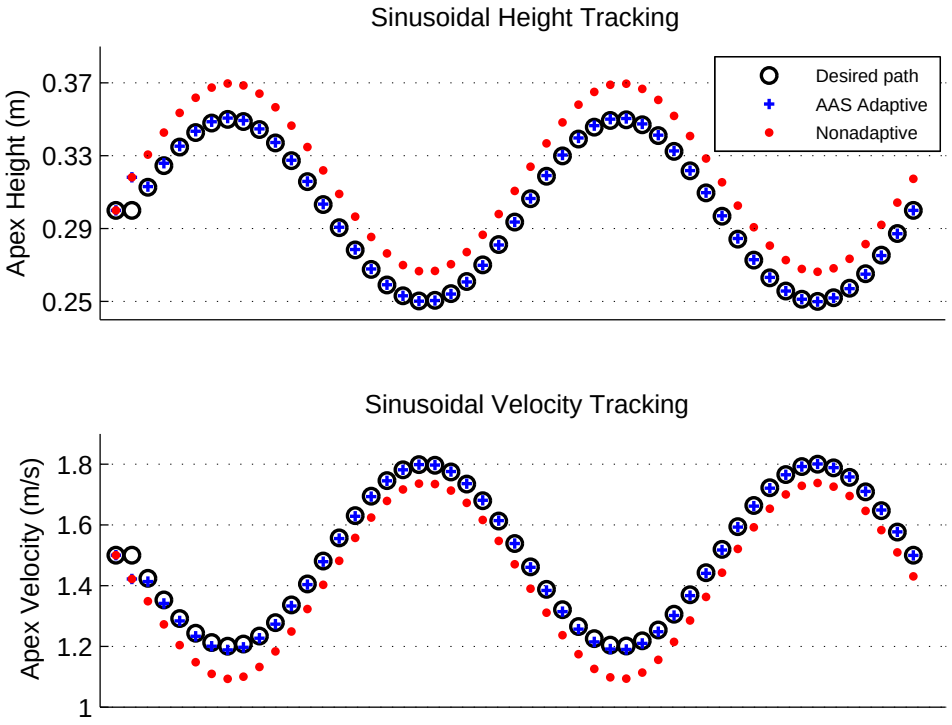


Figure 3.12: Apex height (top) and speed (bottom) tracking performance for a sinusoidal reference trajectory for the TD-SLIP model started with a 20% error in both the spring and damping constants. Each data point corresponds to a single apex event.

Finally, Fig. 3.13 illustrates the case where the robot leg faces with an unexpected breakage during its locomotion, resulting in deviations in leg spring and damping constants. Similar to tests in Section 3.2.3, both the adaptive and nonadaptive tests start with a 5% error in both spring and damping constants in the dark shaded region. It can be seen from Fig. 3.13 that the nonadaptive controller tracks the desired path with a small constant steady-state error while the adaptive controller can accurately follow the same trajectory. At $t = 1.05s$, we apply a step change to the actual values of leg spring and damping constants to simulate a sudden breakage likely to occur for legged platforms. Due to this event, the estimation error in these parameters increase to 20% which results in a high steady-state tracking error for the nonadaptive controller. In contrast, the

adaptive controller with the AAS predictor quickly compensates this error with the help of its on-line estimation capability of leg compliance and damping.

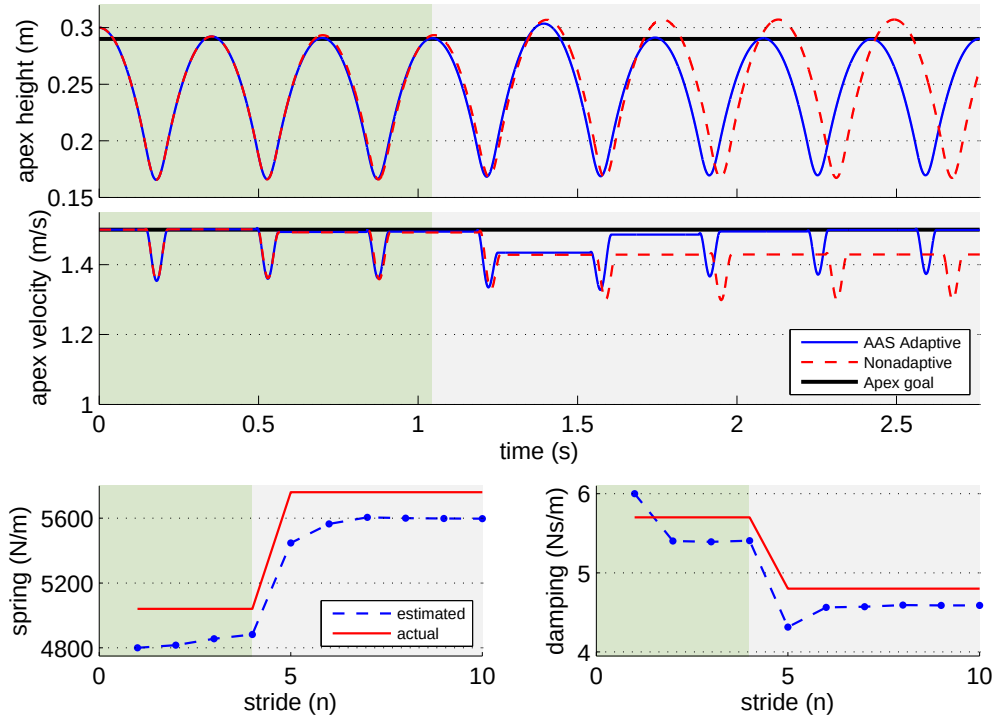


Figure 3.13: Two example TD-SLIP simulations started with 5% error in both spring and damping constants (dark shaded region). One of them starts with a non-adaptive controller and the other uses our adaptive controller with the AAS predictor. An unexpected breakage occurs in the robot leg about $t = 1.05s$ and it increase the estimation error in both spring and damping constants to 20% instantaneously.

3.3.4 System Identification with the ESM Predictor

In this section, we present the system identification performance of our algorithm with the ESM predictor. Fig. 3.14 shows an example TD-SLIP simulation similar to the example of previous section, but with the ESM predictor instead. Similarly, we control the first four steps in the dark shaded region with a nonadaptive controller resulting in a large steady-state tracking error. Then, at $t = 1.4s$ our adaptive controller with the ESM predictor is activated for on-line identification

of system parameters. Finally, we initiate a step change in the apex goal setting around $t = 2.5s$ and observe that our algorithm is robust to these changes.

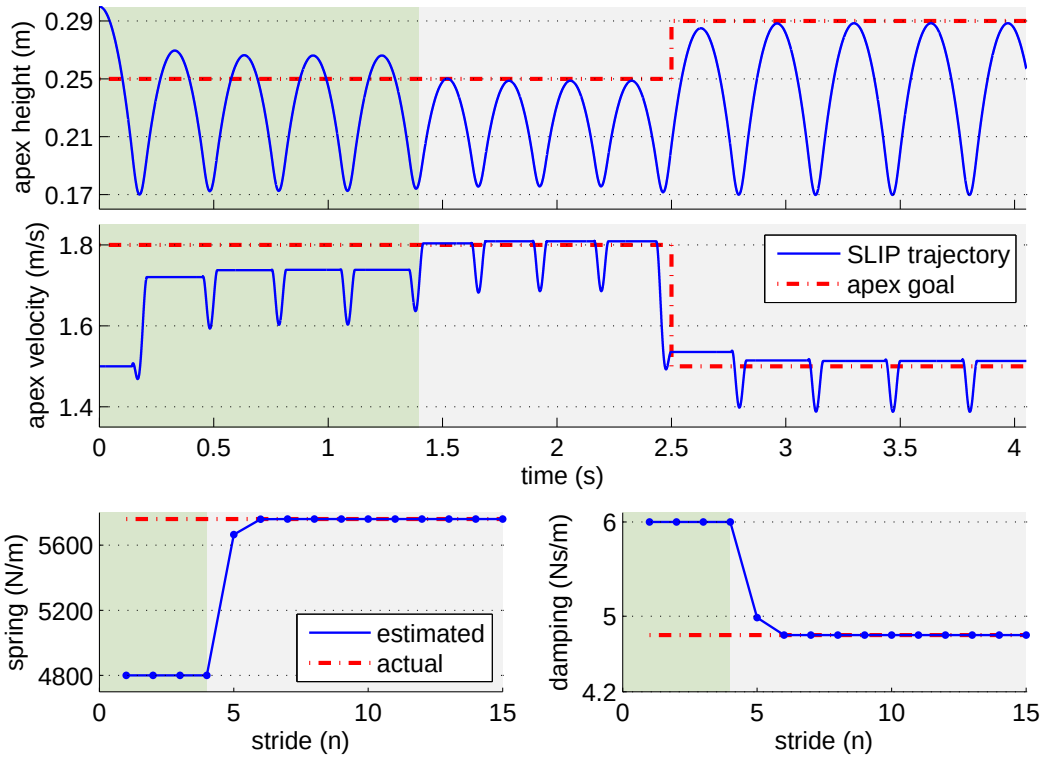


Figure 3.14: An example TD-SLIP simulation started with a non-adaptive controller (dark shaded region) and 20% error in both the spring and damping constants. Our adaptive controller with the ESM predictor was started around $t = 1.4s$ and a step change in the apex goal was given around $t = 2.5s$.

As in SLIP case, the use of ESM predictor allows better estimation of unknown dynamic parameters at the expense of steady-state tracking accuracy. This result can be observed in the bottom plots of Fig. 3.14. Section 3.2.4 gives an intuitional proof of these results.

Following this isolated example, Fig. 3.15 illustrates the system identification performance of our adaptive method with both the ESM and AAS predictors across a large range of apex goal and parameter settings. Since the nonadaptive controller does not estimate the system parameters, we have not included its results in the estimation figures. Our results for both the spring and damping constants show that the ESM predictor perfectly estimates system parameters.

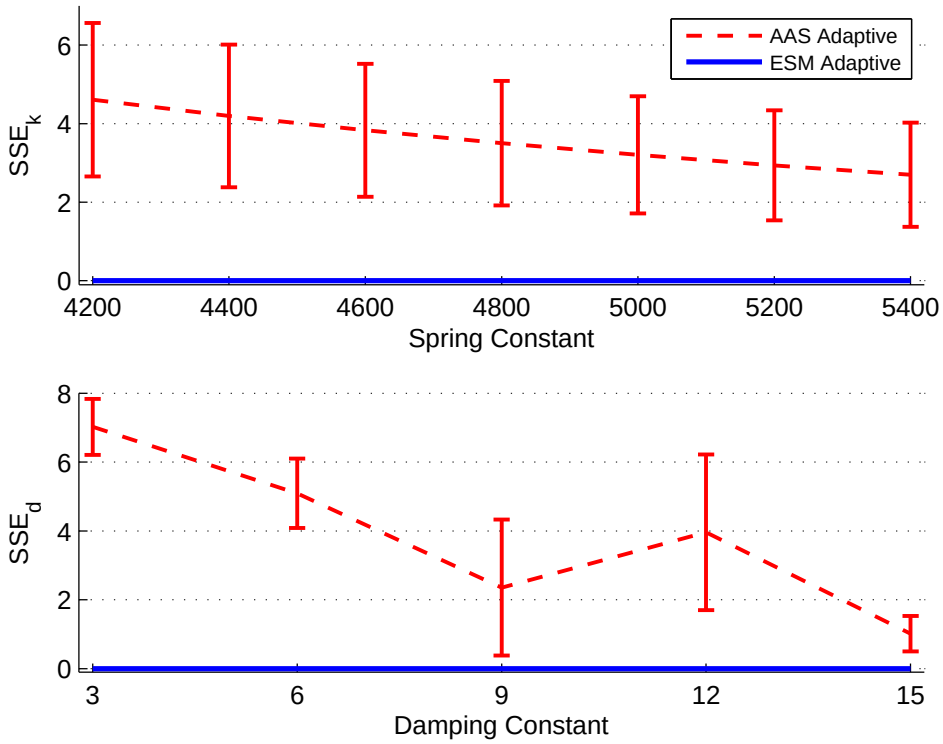


Figure 3.15: Errors in steady-state estimations for the spring (top) and damping (bottom) constants using the AAS adaptive and ESM adaptive controllers for the TD-SLIP model. Error measures were averaged across 306180 simulation runs with different goals and initial parameter estimates. Vertical bars show standard deviations.

In contrast, AAS predictor also performs very well and yields steady-state parameter estimation errors well below the 10-15% that would be expected from manual calibration alone.

As in Section 3.3.3, we prepare a scenario where the robot leg encounters breakage problem during its locomotion. We use the same parameter configuration used in the experiment of Fig. 3.13. It can be seen in Fig. 3.16 that both the adaptive and nonadaptive controller maintains a very small steady-state tracking error in the first region. However, after the breakage event, the steady-state tracking error in the nonadaptive controller increases substantially while the adaptive controller with the ESM predictor preserves its small tracking error. The important thing here is to notice that the adaptive controller can accurately

estimate leg compliance and damping even in the presence of a sudden change in their values.

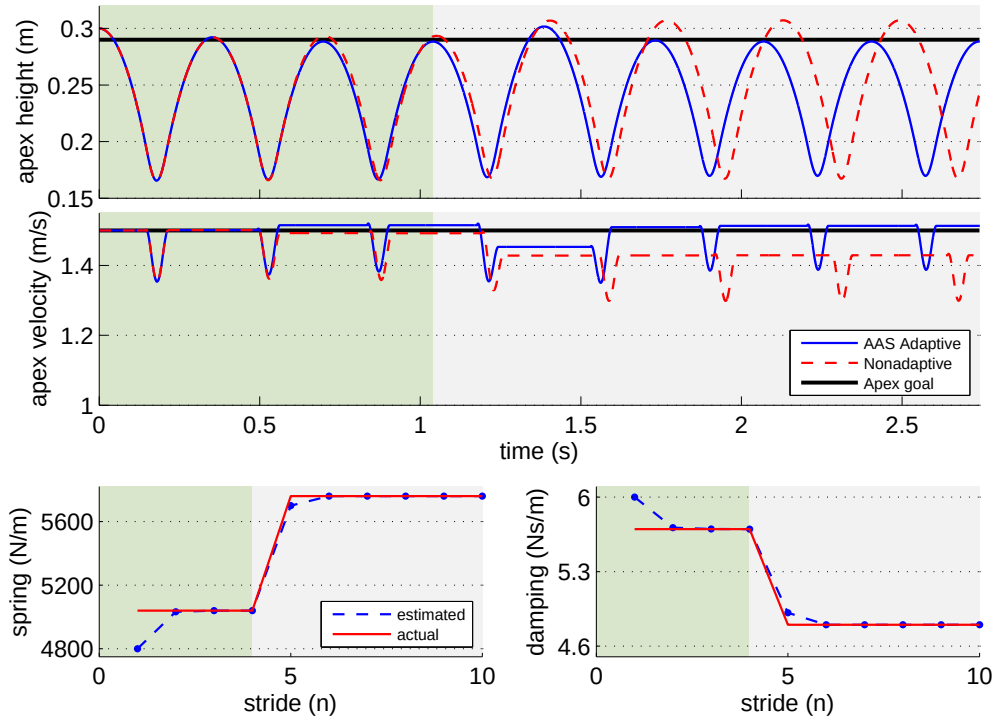


Figure 3.16: Two example TD-SLIP simulations started with 5% error in both spring and damping constants (dark shaded region). One of them starts with a non-adaptive controller and the other uses our adaptive controller with the ESM predictor. An unexpected breakage occurs in the robot leg about $t = 1.05s$ and it increase the estimation error in both spring and damping constants to 20% instantaneously.

Note that, as given in Table 3.3, we used different damping values in the range of [3 15] Ns/m and tested up to 20% deviation from the original values. During the experiments, we observed that $d = 15$ Ns/m is the maximum damping value in which monopod can run successfully. Beyond this value, the locomotion becomes unstable. All of the above mentioned results excludes the results of these unstable runs.

3.4 Discussions and Future Work

In summary, our adaptive controller can be used both as a system identification tool through the use of ESM predictor, or as an accurate gait controller for apex states with the AAS predictor. The latter option is much more suitable for on-line operation on a physical running robot since the approximate solutions and associated Jacobians can be formulated analytically, making them computationally feasible. The ESM predictor, however, not only requires simulated trajectory predictions, but also incorporates numeric differentiation around these simulated trajectories, making it much more suitable for off-line system identification.

Nevertheless, in all cases, our adaptive methods perform much better than the non-adaptive approach both for gait control and system identification. Our contributions with this method clearly illustrate that when analytic solutions to the dynamics of a legged platform are available, their structure and efficiency can be exploited to yield effective solutions both for control and system identification.

Our future work includes extensions of this method to more complex legged models and locomotion controllers as well as their implementation on actual robotic platforms.

Chapter 4

TOWARDS EXPERIMENTAL INQUIRIES

Robotics as a comprehensive field of science and engineering, requires a harmony between the theory and application. As a result, successful ideas and innovations in theory should also show the same performance in real life applications to be a breakthrough study in this field. Motivated by this principle, we built an actual one-legged robot platform towards experimental inquiries of our adaptive control strategy whose success has been proved in extensive simulation studies [61].

The main principle in the design of this robot is to separate the design of the planarizer system from the robot in order to achieve a modular testbench which can be used for different legged robot platforms in the future. The following sections discuss the mechanical, electronics and software design of our robot as well as the performed system identification studies to identify the unknown parameters. Additionally, we investigate the robot's motion achieved by a simple torque-actuated open loop controller.

4.1 Robot Design

4.1.1 Mechanical Design

The concept of a planarizer with a circular boom is a widely used approach in the design of legged robot platforms [19, 41, 44]. One of the main advantages of this method is that the robot can travel long distances in circles without any interruption. However, the boom length should be large enough to treat the robot's trajectory as a planar motion in return.

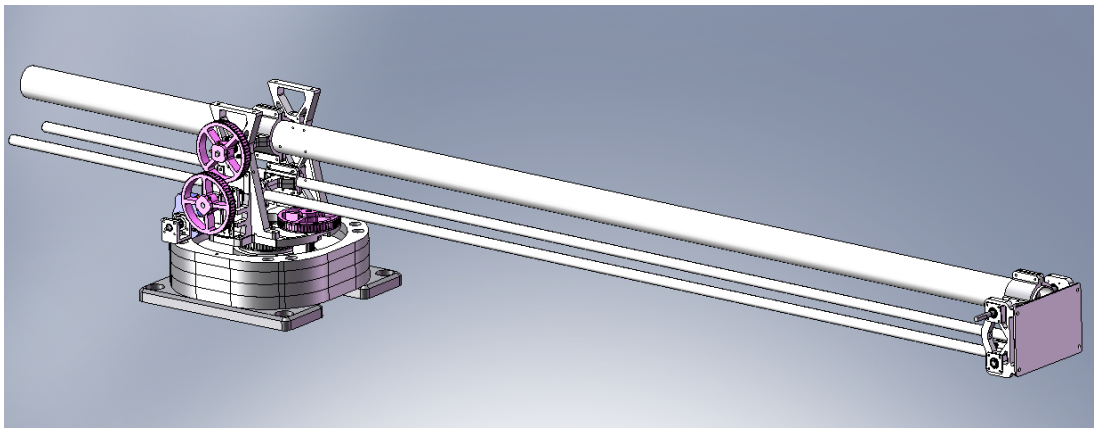


Figure 4.1: The CAD design of the planarizer.

The CAD design of the planarizer and with the boom is depicted in Fig. 4.1. The main goal of the planarizer is to allow free horizontal and vertical motion to the robot leg, which is located at the tip of the planarizer boom, in order to achieve accurate measurement of these motions by optical encoders in the planarizer.

Fig. 4.2 shows the mounted, initial prototype of the robot we have built. The left figure on Fig. 4.2 shows the overall structure of the robot while the right figure more focuses on the planarizer with some fundamental components. Here the configuration of the encoders for the measurement of horizontal and vertical rotation can be clearly seen. In addition to that, a 1x6 pulley system is also seen which is used to increase the resolution of the encoders. The boom length (the

distance between the robot and the planarizer) was chosen to be 1.67 m. in this design to preserve the rigidity of the booms although longer booms are better to imitate planar locomotion.

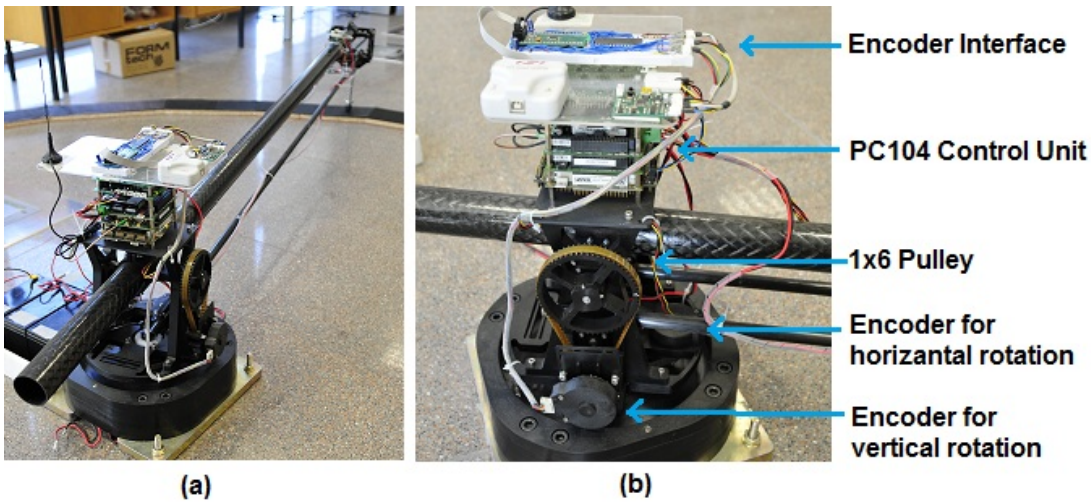
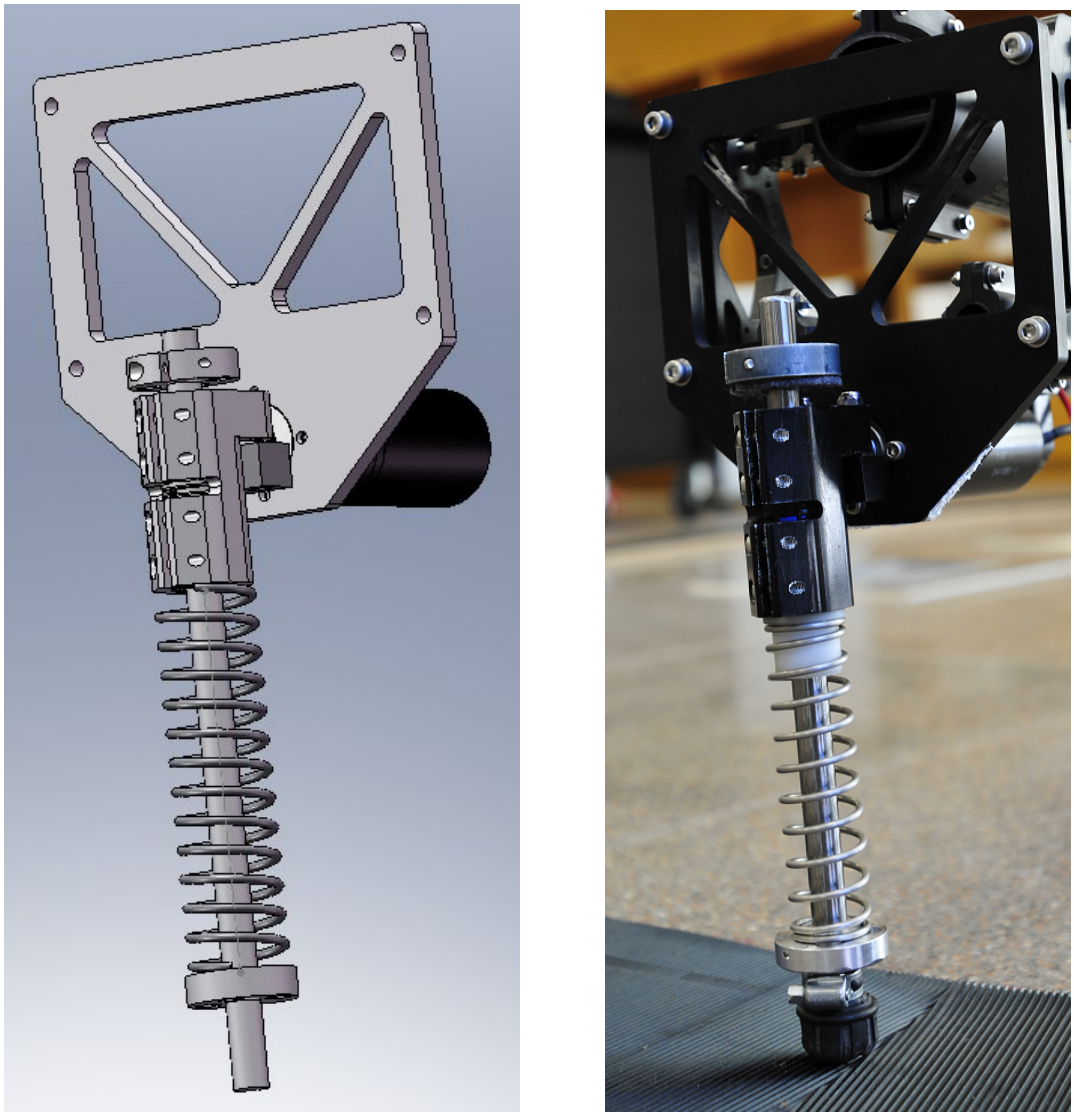


Figure 4.2: The hopper robot. (a) Overall structure of the initial prototype. (b) Planarizer part with a close view of pulley system.

In the aforementioned planarizer system, the measured weight at the tip of the planarizer is considered as a point mass in the connection of the hip motor and the robot leg. Therefore, the combination of this point mass with the connected leg forms the Spring-Loaded Inverted Pendulum (SLIP) model at the hip. Both the CAD design and actual setup of the robot leg is depicted in Fig. 4.3.

Since the SLIP model assumes zero leg mass, the robot leg should be as light as possible. However, we observed in our initial leg designs that materials like aluminium can deform easily with sudden ground collisions (especially in high velocities). Therefore, we used a steel rod to preserve the balance between the weight and rigidity.

A simple bearing part is connected to the hip motor with a locking mechanism to prevent possible dislocations of the leg from the robot body. Linear ball bearings are used inside this bearing part to decrease the friction due to movement of the leg. Additionally, two stopper mechanisms are used to constraint leg movement inside these bearings.



(a) CAD design (b) Actual setup

Figure 4.3: Robot leg including the CAD design and the actual setup.

Finally, the leg stick is surrounded by a spring to implement the SLIP template in the actual platform. This spring acts as a mechanical capacitance in the leg to store and release energy when it is compressed or decompressed respectively. The spring is also chosen after extensive experimental studies to find optimum stiffness and damping.

In this part, the mechanical design of the robot is performed by other group members. My main contribution is the construction of the robot leg.

4.1.2 Electronics - Hardware Design

All computational and motor control hardware of our robot was placed on the planarizer side. The robot leg is a pure mechanical system with no sensor or motor on it. Four 12 V lead-acid batteries are connected to the planarizer from outside through a slip ring to ensure free rotation around the planarizer.

A Cool LiteRunner-LX800 (PC104 single board computer) with an AMD Geode LX800 processor running at 500 MHz is used to perform all the necessary computation and to implement the control algorithms. The robot leg is actuated by a Maxon RE-40-148877 150W brushed DC motor combined with a Maxon GP-42-C two-stage 1:3.5 planetary gear [66]. Although motor is driven by PWM voltage control, approximate torque control can also be implemented with additional back-EMF compensation in software, which allows control of motor and hence the leg angle via PD control loops.

For the sensing part, a two-channel HEDS-5540 A11 optical encoder with 500 lines (2000 counts per revolution) is used to measure the rotation of the hip motor. In addition to that, two two-channel US Digital E3 optical encoders with 2048 counts per revolution is used to measure the horizontal and vertical motions of the robot.

Universal Robot Bus (URB) system is used to support communication inside the robot for data-acquisition and actuation. The URB is a real-time fieldbus architecture that facilitates easy and modular deployment of heterogeneous sensor and actuator nodes that require a central control authority (CPU in our case) in a robot while providing a software abstraction that eliminates the possible problems arising due to the lack of hardware homogeneity [67].

The hopper robot has one actuator node (Hip Node) and one sensor node (Encoder Node) which are connected to the CPU by the URB interface. The

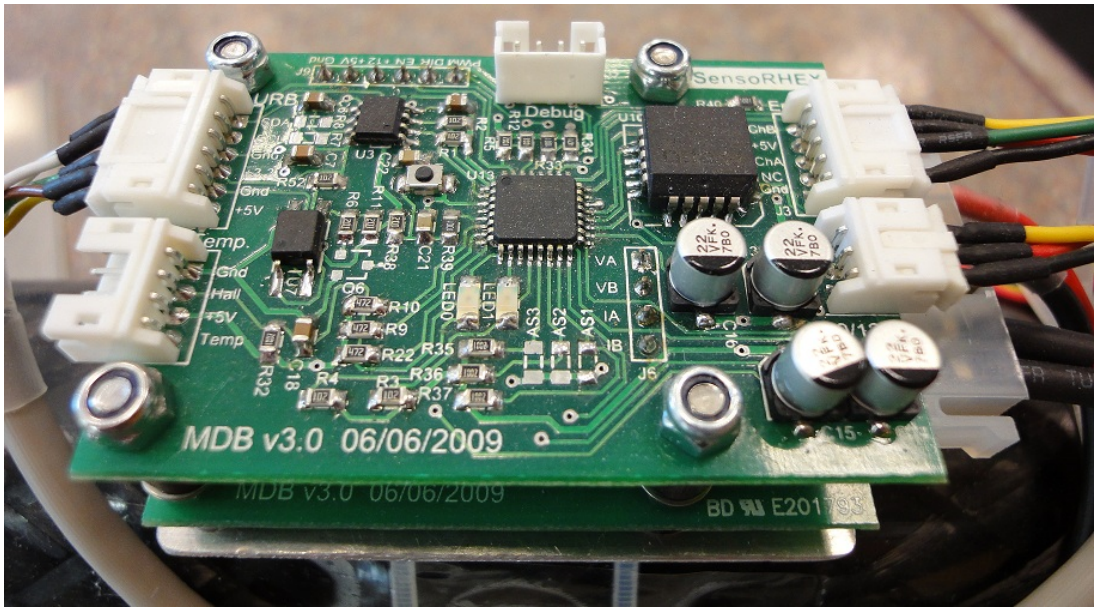


Figure 4.4: The Hip Node. The motor control node to perform local actuation and sensing tasks for the robot leg.

Hip Node (illustrated in Fig. 4.4) is a motor control node which is designed to perform local controllers associated with sensors and actuators on the robot leg. The Hip Node reduces the computational burden on CPU by handling the local control tasks on its own microprocessor. Another primary functionality of the Hip Node is to build an interface to the optical encoder attached to the hip motor to sense the relative hip angle. Hip Node sends the hip angle to the CPU when requested by making the necessary computations with a standard HCTL 2021 encoder counter chip.

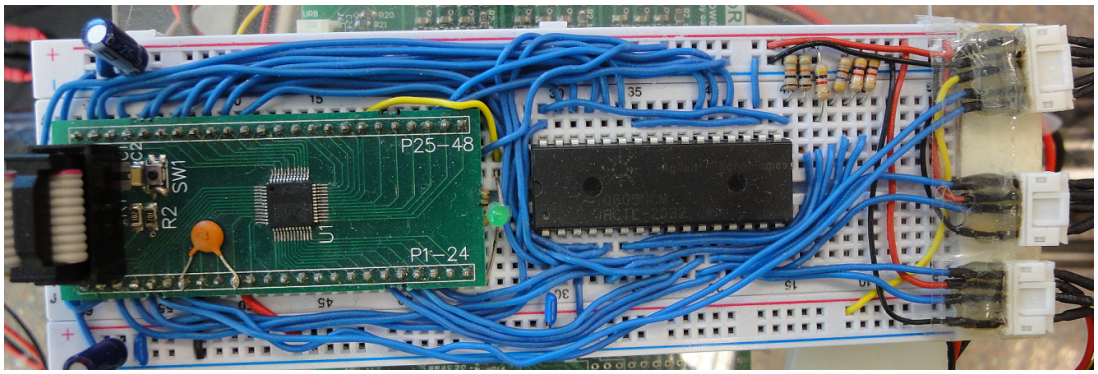


Figure 4.5: The Encoder Node. The planarizer encoder node to measure the vertical and horizontal rotation of the robot body.

Similarly, the Encoder Node (illustrated in Fig. 4.5) performs necessary computations to measure the relative horizontal and vertical angles of the boom by using a HCTL 2032 encoder counter chip.

All hardware nodes in the electronic design are connected to the CPU through a gateway structure called 'The URB Bridge' (illustrated in Fig. 4.6). The main function of the bridge is to enable asynchronous communication between the CPU and the hardware nodes. The bridge buffers the data and asynchronously processes the requests coming from the CPU for the data exchange with hardware nodes.

As explained, the robot has a well-organized electronic system structure through a hierarchical arrangement of the electronic parts. The Fig. 4.7 illustrates this structure with an overview of the electronic components used in the robot.

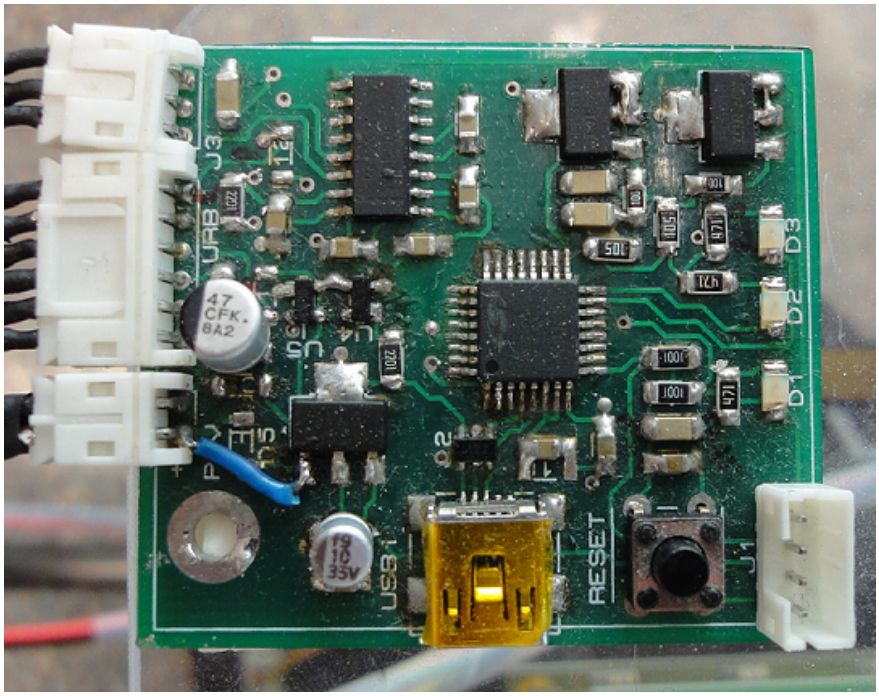


Figure 4.6: The URB Bridge. The gateway between the CPU and the hardware nodes on the bus.

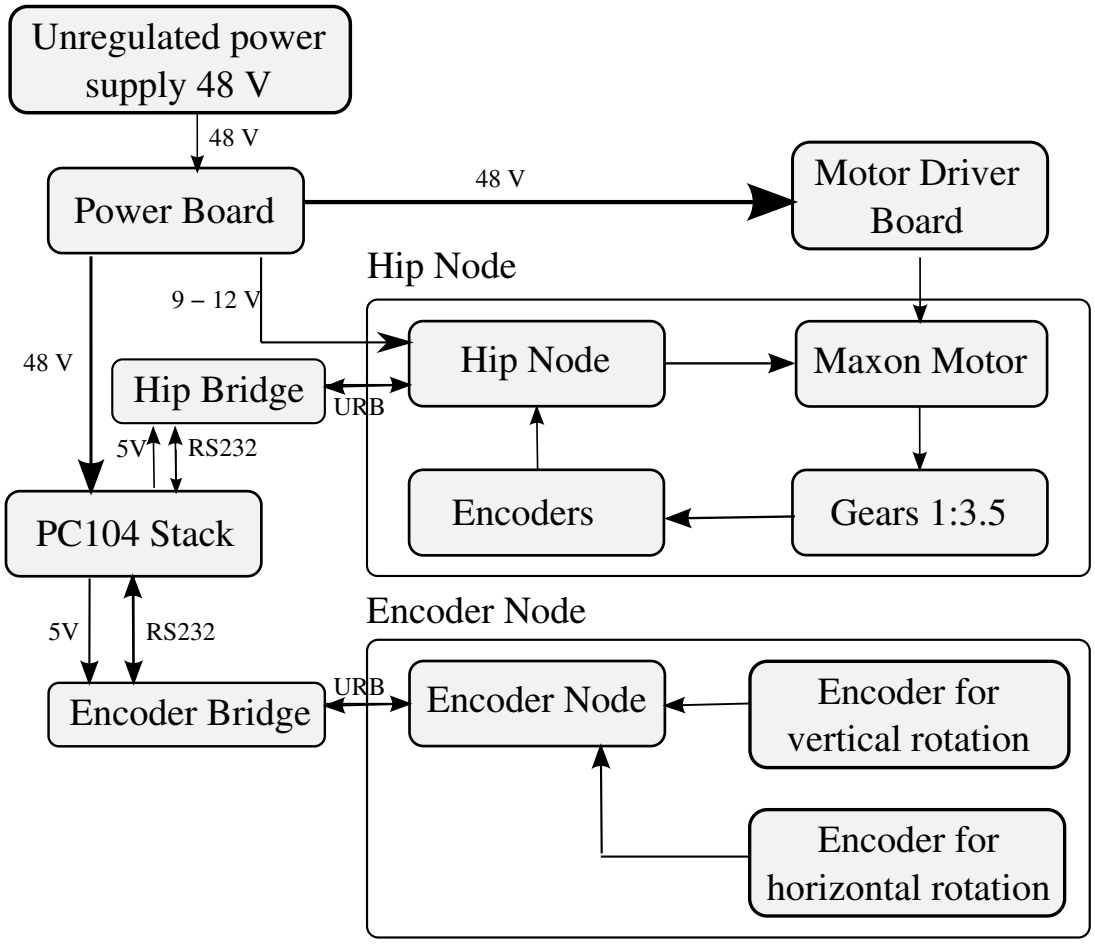


Figure 4.7: The Electronic System Structure.

In this part, the bridge and the hip node components were previously designed and constructed. I directly used these components in the electronics structure of the robot. My main contribution is to design and construct the planar encoder node which is used to measure the horizontal and vertical rotation of the robot by using the encoders in the planarizer. I also implemented the necessary software for this node.

4.1.3 Software Design

The algorithm development of the robot was performed in C++ programming language with the support of RHexLib library. The RHexLib is a software library which is design to ease the algorithm development for robot platforms [15]. The

core of the library relies on the concept of a Module which defines specific tasks that have to be performed periodically such as the reading of encoders and control computations [15]. On the higher level there is a Module Manager which manages the execution, access control and registry of all modules in the system [15]. Finally on the lower level there is an abstract hardware interface layer where the low level hardware access is performed [15]. The details of the RHexLib library can be found in [15].

4.2 System Identification

This section briefly reviews the system identification studies performed to identify the unknown parameters of the robot represented in Section 4.1. Relevant values are the leg spring and damping constants and the point mass attached to the SLIP model.

In this study, we use an off-line manual calibration technique that relies on fitting experimental data to an analytical map which includes all interested parameters in it. We chose vertical hopping as the experiment to ease the analytical return map of a single stride. The following sections describe the vertical hopping experiments, our derived analytical return map and the system identification algorithm to estimate the values of our interested variables.

4.2.1 Vertical Hopping Experiments

Since the experiments only deal with the vertical motion, a module was implemented to prevent the rotation of the legs during the collisions since any change in the leg angle will affect the model characteristics substantially. The stand module performs this task by using a PD controller which ensure that the robot leg stays upright when the module is activated.

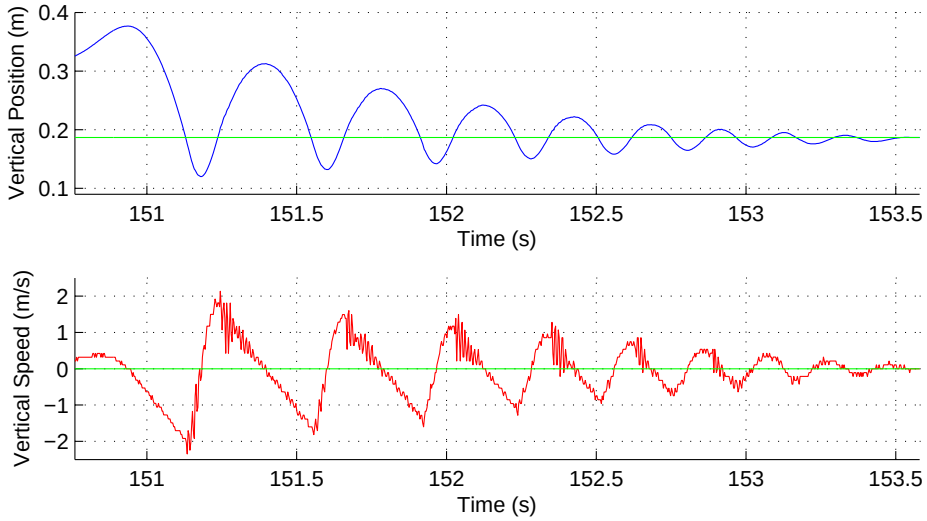


Figure 4.8: A single hopping experiment in the stand mode. The robot was left from an initial height with no vertical speed. The upper and below graphs illustrate the vertical position and speed over time until the robot stabilizes itself in the standing mode.

Then, the robot was thrown upwards and its motion was recorded by using the optical encoders until it stabilizes itself in the stand mode on the ground. The reason why the robot was threw upwards initially is to detect actual apex states by avoiding any affect of the initial launch. We made about 150 hopping tests in which both the vertical position and speed of the robot with respect to time was recorded. Fig. 4.8 shows the results of a single experiment with vertical position and velocity.

As seen in Fig. 4.8 it takes 7 - 8 strides for the robot to loose its initial energy and stabilize itself in the standing mode. Among these multiple strides, we will only deal with the data from the first apex state to the next one to prevent as much noise as we can.

The desired apex states can be found easily by finding the zero crossings of the vertical speed trajectories by using Matlab. However, the vertical speed data contains a non-negligible amount of noise since it is derived by the numeric differentiation of the position data. Because of this reason, some filtering techniques

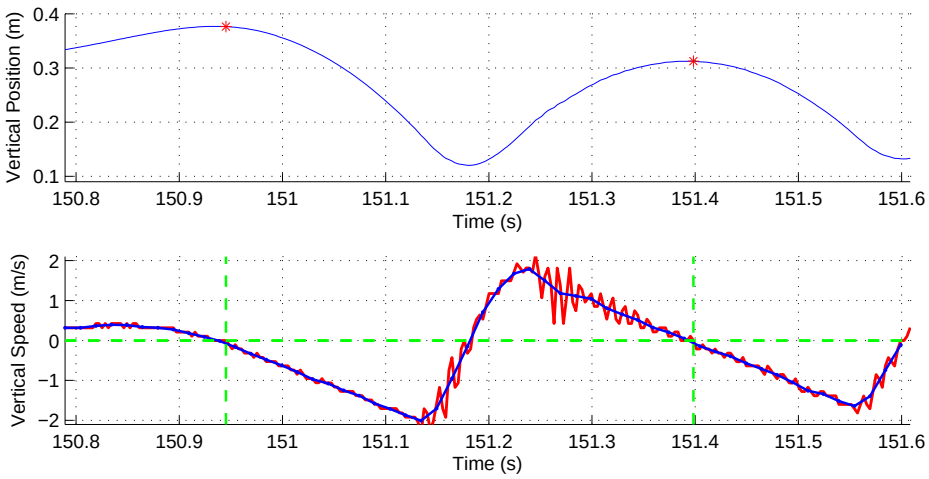


Figure 4.9: Vertical position and speed data filtered with a bidirectional butterworth filter. The resulting apex states detected from the zero crossings of the speed data matches with the summits of the position data.

should be applied to this data before processing it. The important problem here is that using a single filter such as a Butterworth filter causes phase offset, which may result in a time shift in the detection of apex states. To avoid such problems Matlab's bidirectional filtering method filtfilt is used in combination with a butterworth filter. This filtering technique does not create any problems since we are working off-line. The filtfilt method filters the given data in both forward and backward directions to minimize the phase distortion. Fig. 4.9 shows the filtered vertical speed and the detected apex states as a result of this process. As it is seen in Fig. 4.9 the zero crossings of the filtered speed match with the highest points of the position data which shows that the filtering method did not cause any phase distortion in data.

4.2.2 One Dimensional Return Map

By using the experimental results of Section 4.2.1, we derive an input matrix which consists of the initial apex states and an output matrix which includes the resulting apex states. Now, this section tries to build a one dimensional return map from one apex state to another for a single vertical stride.

As previously stated, the SLIP consists of two locomotion phases according to its contact with the ground. The flight dynamics of the SLIP model was explained in Section 4.2.2 and it does not change for vertical hopping experiments. However, the stance dynamics is easier for vertical hopping according to the original dynamics since the leg angle is no longer takes place in the equations of motion.

By using the free body diagram of the SLIP model at the instant when the leg touches the ground, the equations of motion for the vertical components of the state can be derived as

$$m\ddot{z} = k(z_o - z) - d\dot{z} - mg, \tag{4.1}$$

where z_o denotes the initial height. The equation (4.1) is in the form of a second-order non-homogeneous differential equation whose solution can be derived easily as shown below

$$z(t) = \exp((-d/(2m))t)[c_1 \cos(wt) + c_2 \sin(wt)] + z_o - mg/k, \tag{4.2}$$

where $c_1 = mg/k$, $c_2 = (2k\rho + dg)/(2kw)$ and $w = \sqrt{4mk - d^2}/(2m)$.

In addition to the locomotion phases and corresponding transition events represented in Section 4.2.2, another event is defined in our new one dimensional return map to consider the effect of liftoff collision between the robot leg and the point mass. As explained in Section 4.1.1, our robot has a stopper mechanism to constraint the leg's movement inside the linear ball bearings. Right before the liftoff event, the accelerating robot in the decompression phase collides with the leg and lifts it. This is an elastic collision where the two body collides and they move together after the collision. We model the effect of this collision by using the leg velocity as shown below

$$\dot{z}^+ = \dot{z}^- \frac{m_{body}}{m_{body} + m_{leg}}. \tag{4.3}$$

where m_{body} and m_{leg} represents the body and leg masses respectively. Note that the leg mass is only considered to model its effect on liftoff velocity. In all other computations the leg is assumed to be massless.

4.2.3 System Identification Results

In previous sections, the experimental and analytical methodology of our system identification algorithm has been explained. This section gives the results of this algorithm and discuss the validity of the identification results.

We used Matlab's Optimization Toolbox for curve fitting between our experimental data and analytical return map. The optimization toolbox has a data-fitting function called lsqcurvefit which solves the nonlinear curve-fitting problems in least-squares sense. Since our goal in this study is to find the system parameter values which minimizes the least-squares errors between the experimental trajectory of a single stride and the analytical trajectory, lsqcurvefit is the best optimization method to solve our problem.

The lsqcurvefit uses trust-region-reflective algorithm which relies on approximating the desired function with lower degree functions in some specific regions (trust regions) and expanding these regions until an adequate model of the actual function is found [68]. As a result of the initial tests with the lsqcurvefit, the identified parameters are given in Table 4.1.

Table 4.1: System Identification Results with Initial and Final Height Experiments

Parameter:	Value	Unit
k	2430	N/m
d	8.67	Ns/m
m	4.36	kg

Fig. 4.10 shows the trajectories of a sample stride for both experimental data and analytical map with these identified parameters. The first inconsistency with

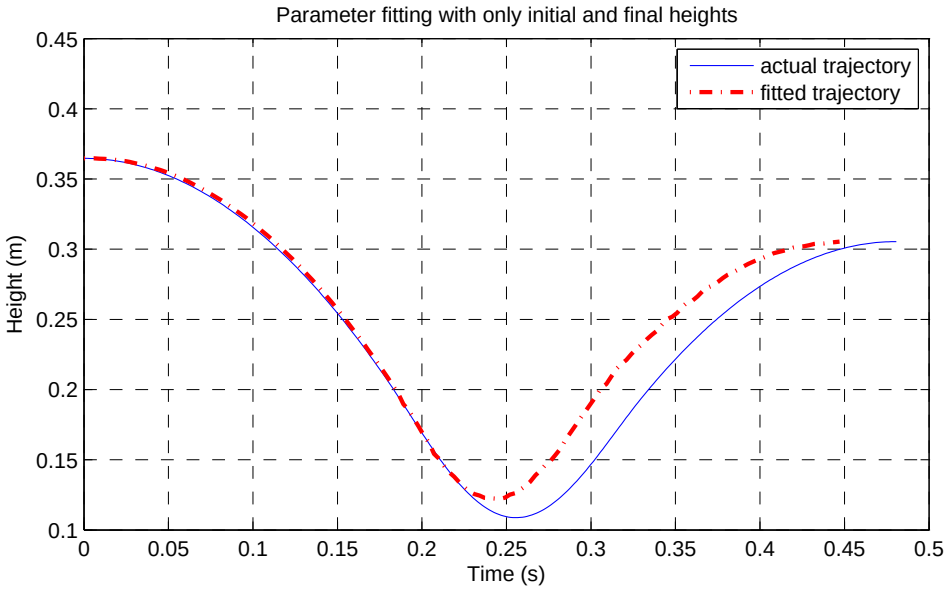


Figure 4.10: The comparison between the actual and fitted trajectories for a single stride for the identified parameter values given in Table 4.1.

the system identification results is that the robot mass is much higher than our manual measurements which is about 2.5 kg including the inertial effects from the boom. The effect of such identification error can be seen in the difference between the actual and fitted trajectories. Especially, the bottom heights of the two trajectories differ a lot from each other. Actually, this result is not surprising for our least-squares problem definition. The lsqcurvefit is fed only with the initial and final apex heights so it tries to minimize the error in the resulting apex height and Fig. 4.10 shows that it accomplishes this task successfully. To get a better fit of the actual trajectory, the bottom height should also be fed to the curve-fitting method since it plays an important role in the stance trajectories which also determines the ascent trajectory.

In this new method, the bottom heights of the experimental data and analytical map is also extracted. The bottom height can be easily found by putting the bottom time t_b into the equation (4.2). However, finding the bottom time requires the analytical differentiation of the equation (4.2) and solving it for zero crossings since the bottom height is a local minima of the vertical position

function. The derived vertical velocity equation is given below

$$\dot{z}(t) = (-d/(2m))\exp((-d/(2m))t)(c_1\cos(wt) + c_2\sin(wt)) + \exp((-d/(2m))t)(-wc_1\sin(wt) + wc_2\cos(wt)). \quad (4.4)$$

The bottom time t_b can be easily found by numerically solving this equation in Matlab. The resulting parameter estimates are given in Table 4.2. Fig. 4.11

Table 4.2: System Identification Results with Initial, Bottom and Final Height Experiments

Parameter:	Value	Unit
k	2350	N/m
d	6.09	Ns/m
m	2.58	kg

shows the actual and fitted trajectories with new parameter estimates. It can be clearly seen that the new parameter estimates give better trajectory fitting to sample experimental data when the bottom height is also considered. Since there is no horizontal velocity in the system the exact fitting of bottom height and the next apex state gives well enough parameter estimates.

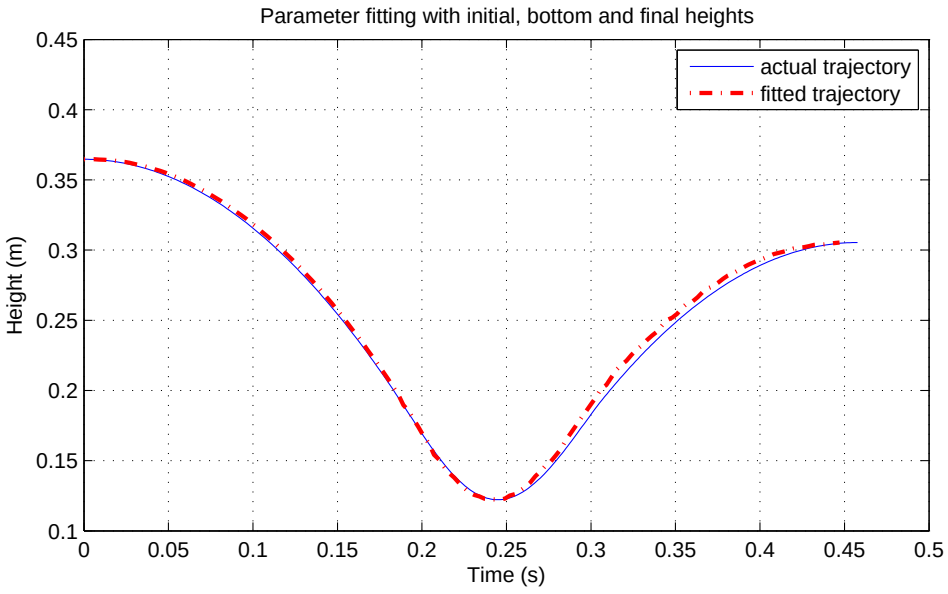


Figure 4.11: The comparison between the actual and fitted trajectories for a single stride for the identified parameter values given in Table 4.2 including the bottom height as a control variable.

4.3 Open Loop Running Experiments

In this section, we discuss the open loop running experiments we performed to observe the performance of our robot and to find the necessary control inputs for stable locomotion. Motivated by the open loop continuous torque strategy of [69], we employ a new open loop torque control algorithm to supply the loss energy to the system at each stride.

The majority of experiments were designed to debug the control software and to tune the control parameters for the most stable locomotion. Our results show that we can obtain a stable running performance for limited number of steps when the open loop control inputs are adjusted properly. In the following sections, we first describe our open loop control algorithm based on a state machine diagram and then present the results of our experimental studies.

4.3.1 Open Loop Control

The controller for our robot was implemented in C++ and RHexLib software library [15]. The control algorithm is based on the Universal Robot Bus (URB) structure explained in Section 4.1.2. All the communication with the actuators and sensors are handled via URB. Our controller is in the form of a state machine diagram (illustrated in Fig. 4.12) which interfaces associated actuators and sensors depending on the active state. This state machine just shows an overview of the controller. We detail the states and associated transition events to explain the underlying control structure.

Start: The initial state of the control algorithm. This state includes the initialization of the system and calibration of the leg position. We throw the

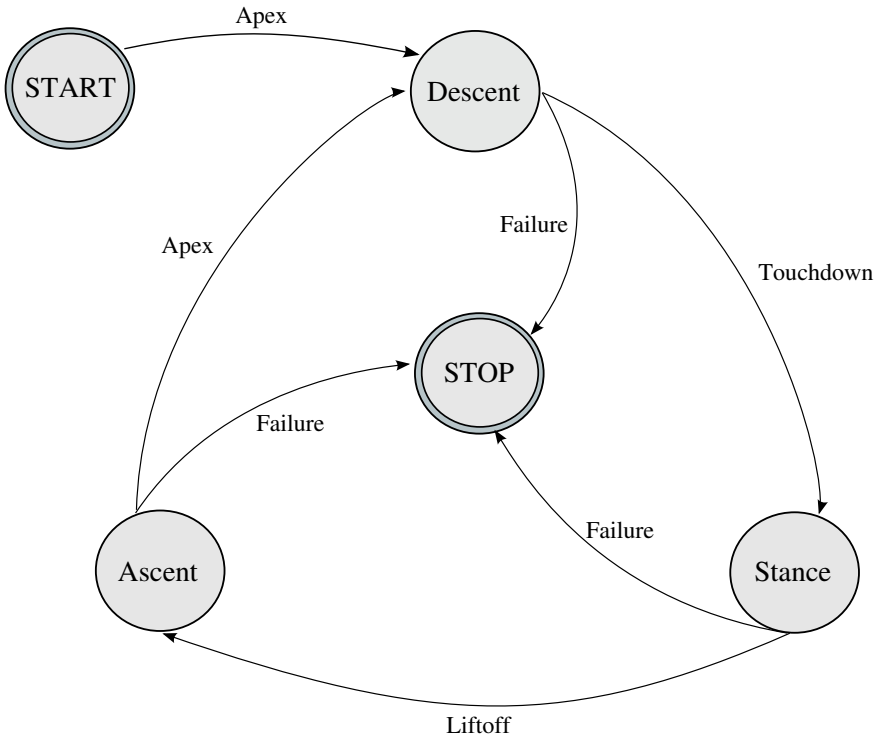


Figure 4.12: State machine diagram of the controller in the entire system loop including the states and the associated transition events.

robot upwards at this state for accurate detection of the apex event and to trigger the state transition to the actual control loop.

Descent: The intuitional explanation of descent phase is given in Section 2.1.1. Here, we explain the behavior of our control algorithm in this state. The descent is that state where the robot sets the touchdown leg angle which has constant value in the case of our open loop control strategy.

After a number of experiments with different values, we decided to use $-25\,\text{deg}$ as the fix touchdown leg angle at each stride. We use PD control to set the touchdown leg angle. At the beginning of the descent phase, the PD controller starts to move the leg towards requested leg angle. The PD gains for this controller is also adjusted experimentally such that the leg reaches desired value before the touchdown occurs.

An important thing about the descent phase is that we do not directly command the requested touchdown leg angle to the PD controller to avoid excessive current demand from the batteries. Instead, we divide the angle

Table 4.3: PD Gains for Leg Position Controller

Gain term:	Value	Unit
K_p	0.22	$A/\deg$
K_d	0.45	$As/\deg$

interval (from the current angle to the requested angle) into discrete steps and command each of these intermediate angles successively.

Stance: The phase where the robot leg is in contact with the ground. In this phase, we inject energy to the system by applying torque via the hip motor. Currently, we apply constant torque during the whole phase which is about $\tau = 7Nm$. However, since we do not use feedback control in these tests, there may be some difference between the requested and applied torque.

Ascent: The phase where the robot starts to ascend during the flight phase. The only objective of this state is to maintain the leg position until the apex event since inertial effects of the torque from the stance phase may cause extreme rotations when the leg is not in contact with the ground. The leg angle is preserved via a PD controller which also uses the same gains given in Table 4.3.

Stop: The phase where the robot disables the hip motor to shut down all the motor activity.

In addition to these states, there are some transition triggering events to handle state changes in the control loop. In Section 2.1.1, we explain how to detect *Apex*, *Touchdown* and *Liftoff* events. We implement the same analytic formulations here to detect these events. Additionally, the *Failure* event detects problems such as the collision with the ground during the locomotion and triggers the *Stop* phase to avoid any damage in the robot.

4.3.2 Experimental Results

In this part, we present the experimental results of our robot. We performed a number of tests for tuning the control parameters to obtain a stable locomotion. In this part, we will mainly focus on one of our final experiments in which our robot preserves its stability for 50 steps which is an acceptable value for an open-loop controlled robot as compared to some other one-legged hopping machines such as Uniroo whose maximum number of hops is about 40 steps [70]. Fig. 4.13 illustrates the results of this experiment. However, from now on we will only use a small portion of this long run in our illustrations for clarity of the figures.

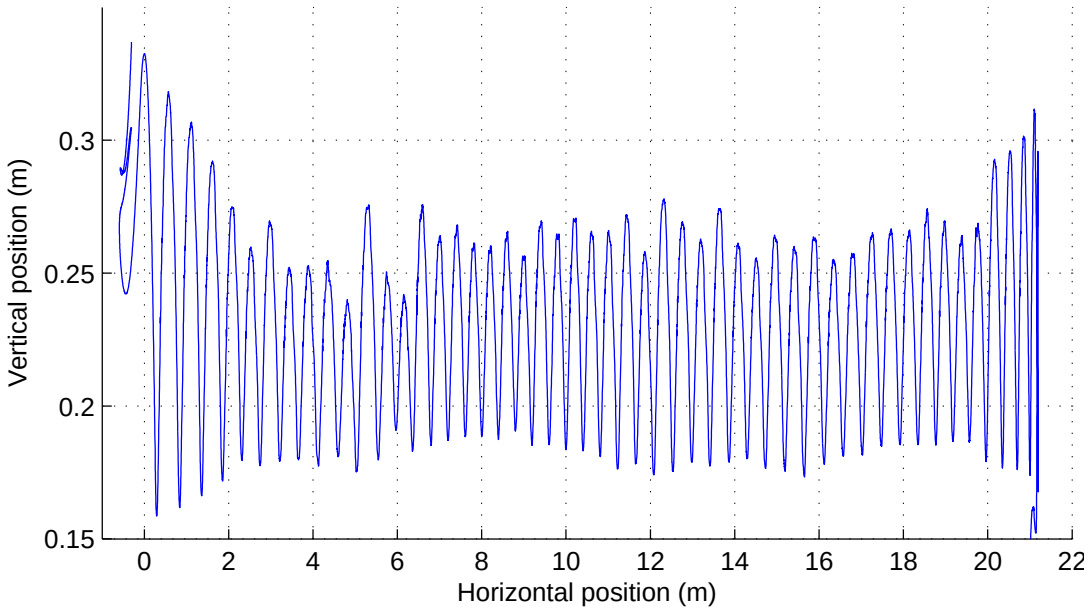


Figure 4.13: Vertical position vs. horizontal position for the COM of our robot. The horizontal position of the first apex event is shifted to zero for better illustration.

Single Stride Video

Fig. 4.14 illustrates the sequential snapshots of the robot during a single stride. The snapshots starts from an apex event and the robot moves towards the right as time progress. The last snapshot illustrates the apex event for the next stride.

The formulations and explanations of each phase and transition event is given in Section 2.1.1. Here, we simply describe what each snapshot corresponds to.

1. Apex Event: the robot is at its maximum height with zero vertical velocity.
2. Descent Phase: the leg is being driven to the touchdown leg angle via PD control.
3. Touchdown Event: the toe of the leg has just touched the ground.
4. Compression Phase: the spring is being compressed. The body rotates around the toe with applied torque.
5. Bottom Event: the instant where the maximum compression occurs. The leg is vertical and the COM is at the lowest point.
6. Decompression Phase: the spring is being decompressed and it releases the stored energy. The body continues to rotate around the toe with hip torque.
7. Liftoff Event: the leg is about the take-off from the ground. The body has its maximum vertical speed.
8. Ascent Phase: the liftoff leg angle is preserved via PD control and the body starts to gain elevation.
9. Apex Event: the robot is at its maximum height with zero vertical velocity.

(a) Apex Event (b) Descent Phase (c) Touchdown Event

(d) Compression Phase (e) Bottom Event (f) Decompression Phase

(g) Liftoff Event (h) Ascent Phase (i) Apex Event

Figure 4.14: Snapshots of hopping motion during a single stride including the phases and associated transition events.

Height vs. Time

Fig. 4.15 illustrates the height of the COM trajectory of our robot over time. The detected apex and touchdown events are also represented with red plus (+) and circle (o) signs, respectively. The desired touchdown height is $\rho_o \cos\theta = 0.225 \cos(25) = 0.204$ (m). The measured values also result in a close touchdown height about 0.21 (m).

The hopping frequency, f_{hop}, is about 3.5 Hz according to our measurements. However, according to the frequency formulation of [71], the hopping frequency should be $f_{hop} = 3.0m^{-0.19} = 2.5$ Hz. One of the reasons behind this difference is that we use stiff spring in our studies as compared to the spring constant formulation of [71].

Finally, the average vertical speed of the robot can be calculated as $2\rho_o f_{hop} = 1.57$ (m/s) which is very high due to our spring stiffness choice as compared to other hopping robots.

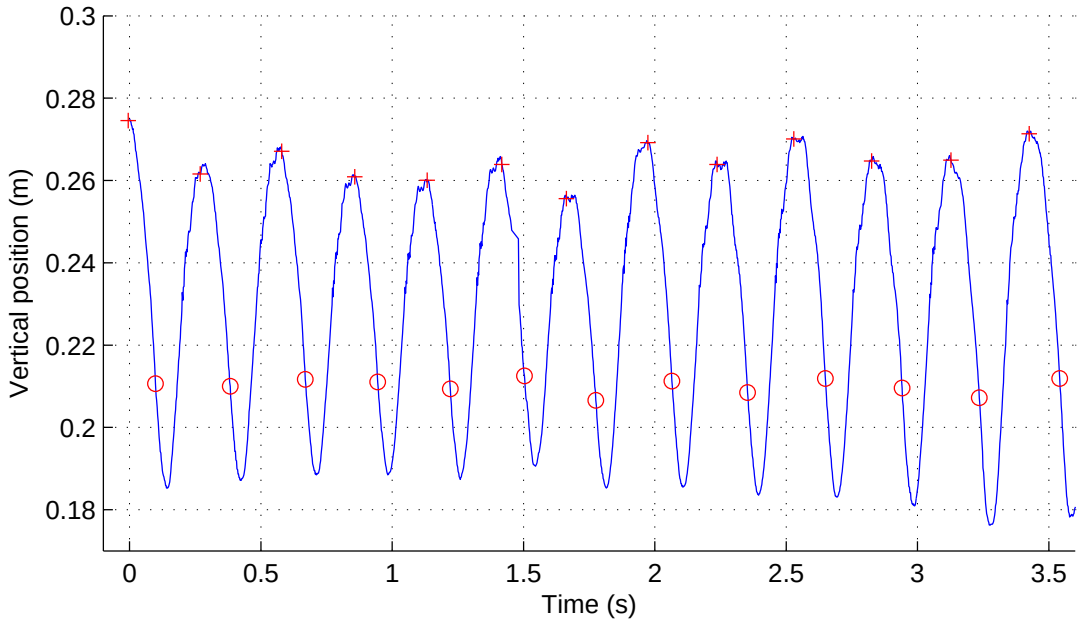


Figure 4.15: Vertical position of the COM of the robot over time including the detected apex (+) and touchdown (o) states.

Horizontal Position vs. Time

Fig. 4.16 illustrates the horizontal position of the robot over time. It is clearly seen that the robot moves forward smoothly. the slope of this plot can be interpreted as the horizontal speed. Therefore, the average horizontal speed of the robot can be calculated from this plot as 1.41 (m/s).

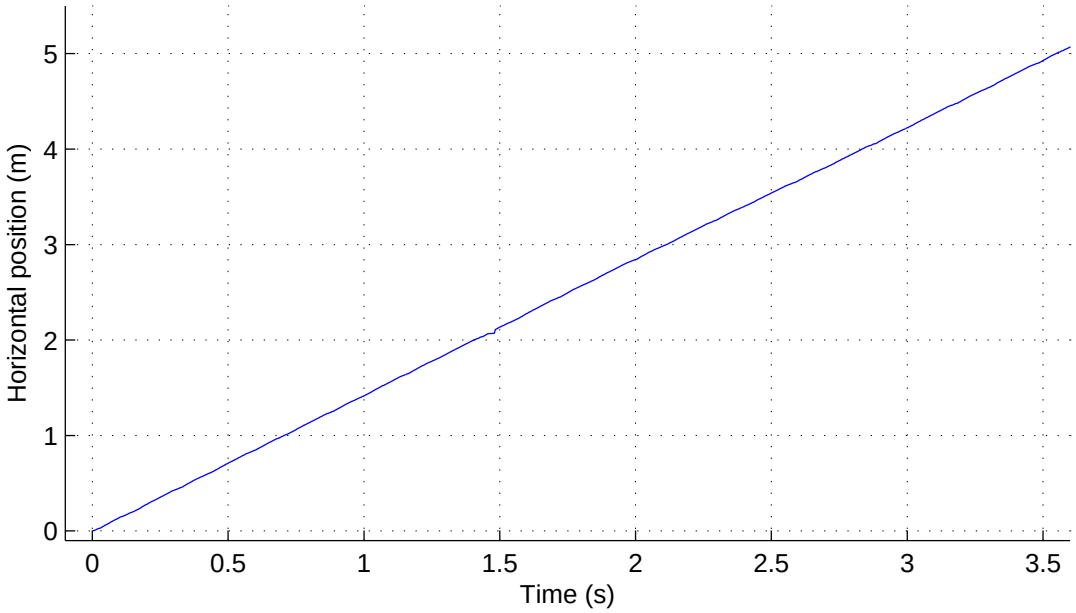


Figure 4.16: Horizontal position of the robot over time.

Height vs. Horizontal Position

Fig. 4.17 illustrates the height of the COM of the robot with respect to horizontal position. The plot shows a smooth, five-meter run of the robot. The robot follows a sinusoidal path as expected. The important thing here is to notice that almost all the apex heights remain above 0.26 m height which easily clears the toe of the leg from the ground. Finally, length of an average stride is measured as 0.4 m from the plot.

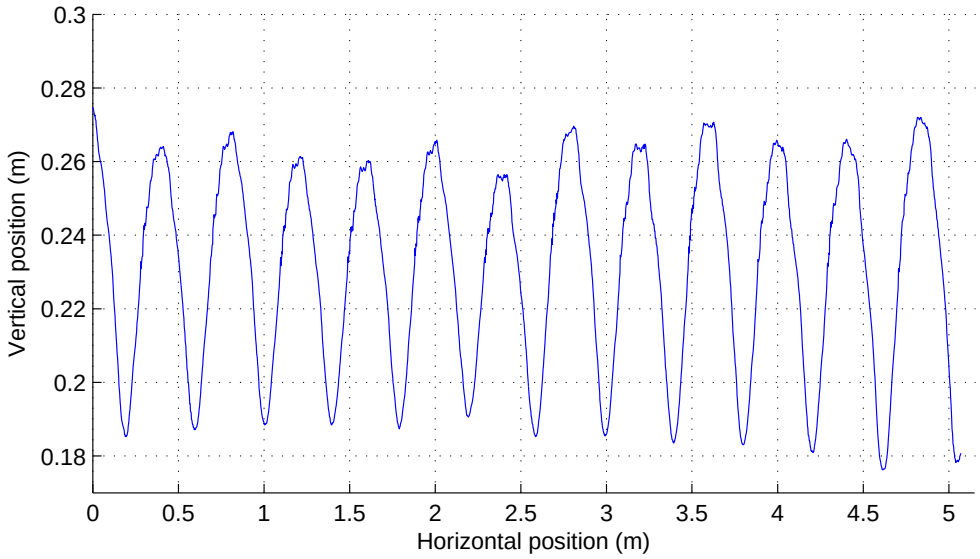


Figure 4.17: Vertical position vs. horizontal position.

Leg Length vs. Time

The leg length equals to rest length during the flight phase. In the stance phase, it can be computed by using the leg angle θ and vertical position z as

$$\rho = \frac{z}{\cos(\theta)}. \tag{4.5}$$

Note that, in some cases the robot may liftoff before the spring is fully extended which results in a smaller leg length at the beginning of the ascent phase. However, we neglect such small errors. Fig. 4.18 illustrates the leg length over time under this assumption.

On the average, the maximum compression is about 0.04 m which corresponds to the %18 of the rest length. This compression results in the following energy

$$E_{spring} = 0.5kx^2 = 1.88J \tag{4.6}$$

where x is the compression length. By using simple potential energy calculations, we can conclude that this energy is enough to elevate the robot body about 0.075 (m) in a lossless system without needing any other energy source. However, we also use torque-actuation to gain extra elevation.

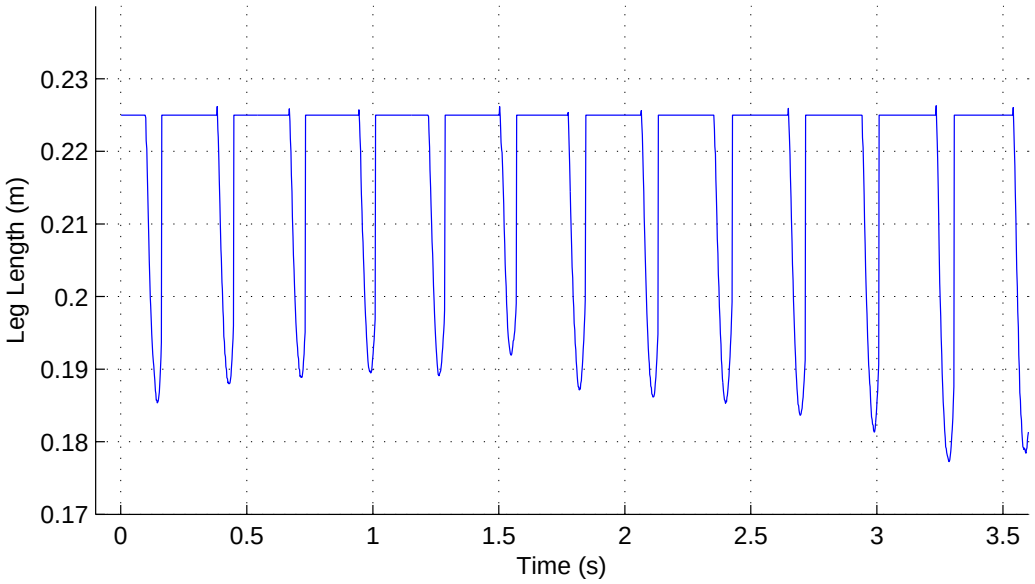


Figure 4.18: Leg length over time.

Duty Cycle

Fig. 4.19 illustrates the average flight and stance times of the locomotion. The duty cycle of the locomotion is computed as the ratio of the stance phase (where the toe of the leg is in contact with the ground) to the duration of the complete stride. In [72], the duty cycle for running in humans is defined as 0.2. Our experiments also results in a close value 0.22 which means that the locomotion of our hopper can be accepted as running according to the running gait definition of [72] based on the duty cycle.

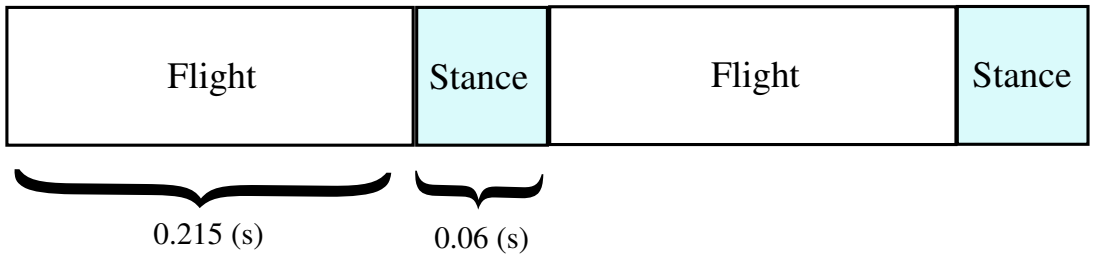


Figure 4.19: Average duty cycle of the locomotion with durations of the flight and stance phases.

4.4 Discussion

The most important goal of our design was the development of a modular planarizer system that can be used for different kinds of robot platforms. Instead of designing measurement and stabilizer units for each robot design, our planarizer enables accurate state measurement for any robot connected to its tip. This results pave the way towards designing more accurate control algorithms and state estimation methods.

The communication inside the whole platform is ensured by Universal Robot Bus (URB). By using such a communication interface, our robot assures easy deployment of extra actuators and sensors on it. Similarly the same design enables modular programming of the robot by using the RHexLib software library. The only disadvantage of using such a system is the high-pitched learning curve but once you learn the system it becomes very easy to implement any algorithm on it.

The results of the system identification studies performed for the robot platform is also consistent with our manual measurements. This kind of parameter calibration technique is widely used in many robot platforms [3, 19–21] but it requires a huge exertion to identify the desired parameters. In addition to that if the desired parameters have tendency to change their values over time, the repetition of this calibration process will create a big problem. As another option, the system will need to work with miscalibrated system parameters. Therefore, we can conclude that the manual calibration technique used in this study may give a better estimation of the system parameters in the initial setup but it is not efficient to use this method to identify the time-varying parameters of the system.

Finally, we implemented a torque-actuated open loop controller to stabilize our robot. In our experiments, we obtained a stable running of 52 steps. We also

made some analysis by using a small portion of our experimental data in which the robot moves in a stable fashion.

4.5 Future Work

One of the important future works about our design is the addition of an extra thrust mechanism to control the compression of the robot leg as in the bowleg hopper [44]. An alternative approach is to use more powerful hip motors which could apply more torque to the robot, so that it can supply the required energy to elevate the robot to desired height.

Another future work direction is to develop the Universal Robot Bus interface, so that faster control loops can be implemented on local nodes and faster communication can be realized between the CPU and nodes. Additionally for the software design, an important future work is to implement some interface library which enables to program the robot with other programming languages such as Matlab which have more computational power than classic C++.

Finally, a closed loop controller is required for a better understanding of the robot's behavior. We need to implement better controllers for stable locomotion of the robot, so that we can perform more complex experiments.

Chapter 5

CONCLUSIONS

In this thesis, we proposed a novel adaptive control algorithm to both support on-line identification of unknown dynamic system parameters and improve steady-state tracking performance of previously proposed control algorithms for the SLIP and TD-SLIP models. Our method used as a system identification tool addresses the practical difficulty of measuring possibly time-varying dynamic system parameters such as spring and damping constants and associated degradation in controller performance when they cannot be correctly estimated. In contrast, our method used as an adaptive controller allows effective elimination of steady-state tracking errors under different types of modeling errors for inverse dynamics controllers.

The choice between these two different modes of operation depends on the choice of a predictor model against which state measurements are compared at each step. We show through systematic simulations that a predictor based on numerical integration of system dynamics is capable of accurate system identification, whereas a predictor based on the analytic approximations proposed in [14] and [1] allows elimination of steady-state tracking errors for a deadbeat controller based on the same approximations. Extensive simulation results for both

predictors show that they successfully realize these objectives and substantially improve on control performance relative to existing non-adaptive controllers.

We also presented the design and construction of a one-legged hopping robot. We performed some manual system identification studies to identify the unknown parameters of our robot. Additionally, we designed a a high accuracy measurement system in the form of a planarizer which can also be used for different robot platforms in the future. We detailed the mechanical, electronics and software design of our robot in the associated sections. Since the adaptive control of running with these platform is beyond the scope of this thesis, we left it as a future work to our studies.

Our longer term goal is to design legged platforms that can successfully negotiate rough terrain. The applicability of mathematical models that are relevant for this purpose critically depends on our ability to accurately estimate associated parameters to be used by model-based planners. Consequently, starting from a direct implementation of the method we propose in this thesis on our monopedal platform, our future work includes extensions to more complex legged models and locomotion controllers. In this context, we believe that our work shows some of the benefits offered by analytic solutions to mathematical models of locomotory behaviors.

Bibliography

[1] M. M. Ankaralı and U. Saranlı, "Stride-to-stride energy regulation for robust self-stability of a torque-actuated dissipative spring-mass hopper," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 3, p. 033121, 2010.

[2] P. Holmes, R. Full, D. Koditschek, and J. Guckenheimer, "The dynamics of legged locomotion: Models, analyses, and challenges," *SIAM Review*, vol. 48, no. 2, pp. 207–304, 2006.

[3] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert, "Autonomous navigation for BigDog," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Anchorage, Alaska), May 3-8 2010.

[4] O. Arslan, *Model Based Methods for the Control and Planning of Running Robots*. M.S. Thesis, Bilkent University, Ankara, Turkey, July 2009.

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

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

[7] 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," *Journal of Field Robotics*, vol. 25, no. 4-5, pp. 223–242, 2008.

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

[9] R. Blickhan, "The spring-mass model for running and hopping.," *Journal of Biomechanics*, vol. 22, pp. 1217–1227, 1989.

[10] W. J. Schwind, *Spring Loaded Inverted Pendulum Running: A Plant Model*. PhD Thesis, University of Michigan, Ann Arbor, MI, USA, 1998.

[11] T. McGeer, "Passive dynamic walking," *International Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990.

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

[13] 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)*, (Istanbul, Turkey), September 2009.

[14] U. Saranlı, O. Arslan, M. Ankaralı, and O. Morgül, "Approximate analytic solutions to non-symmetric stance trajectories of the passive spring-loaded inverted pendulum with damping," *Nonlinear Dynamics*, vol. 62, no. 4, pp. 729–742, 2010.

[15] U. Saranlı, *Dynamic Locomotion with a Hexapod Robot*. PhD Thesis, The University of Michigan, Ann Arbor, MI, USA, September 2002.

[16] U. Saranlı and D. E. Koditschek, "Template based control of hexapedal running," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Taipei, Taiwan), September 2003.

[17] I. Poulakakis and J. W. Grizzle, "Formal embedding of the spring loaded inverted pendulum in an asymmetric hopper," in *Proceedings of the European Control Conference*, 2007.

[18] M. Ankaralı and U. Saranlı, "Control of underactuated planar hexapedal pronking through a dynamically embedded SLIP monopod," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Anchorage, Alaska), May 3-8 2010.

[19] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT Press, 1986.

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

[21] M. Ahmadi and M. Buehler, "Controlled passive dynamic running experiments with the ARL-monopod II," *IEEE Transactions on Robotics*, vol. 22, pp. 974–986, October 2006.

[22] C. Chevallereau, E. R. Westervelt, and J. W. Grizzle, "Asymptotically stable running for a five-link, four-actuator, planar bipedal robot," *The International Journal of Robotics Research*, vol. 24, no. 6, pp. 431–464, 2005.

[23] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[24] I. Poulakakis, E. Papadopoulos, and M. Buehler, "On the stability of the passive dynamics of quadrupedal running with a bounding gait," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 669–687, 2006.

[25] I. Poulakakis and J. Grizzle, "The spring loaded inverted pendulum as the hybrid zero dynamics of an asymmetric hopper," *IEEE Transactions on Automatic Control*, vol. 54, no. 8, pp. 1779 –1793, 2009.

[26] U. Saranlı, A. A. Rizzi, and D. E. Koditschek, "Model-based dynamic self-righting maneuvers for a hexapedal robot," *International Journal of Robotics Research*, vol. 23, pp. 903–918, September 2004.

[27] K. J. Astrom and B. Wittenmark, *Adaptive Control*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1994.

[28] I. D. Landau, R. Lozano, and M. M'Saad, *Adaptive Control*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1998.

[29] I. D. Landau, *Adaptive Control: The Model Reference Approach*. New York: Marcel Dekker, Inc., 1979.

[30] M. Berkemeier and K. Desai, "Control of hopping height in legged robots using a neural-mechanical approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Michigan, USA), May 10-15 1999.

[31] W. Ilg, J. Albiez, H. Jedele, K. Berns, and R. Dillmann, "Adaptive periodic movement control for the four legged walking machine bisam," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Michigan, USA), May 10-15 1999.

[32] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *The International Journal of Robotics Research*, vol. 22, no. 3-4, pp. 187–202, 2003.

[33] G. Brambilla, J. Buchli, and A. Ijspeert, "Adaptive four legged locomotion control based on nonlinear dynamical systems," in *Proceedings of the International Conference on the Simulation of Adaptive Behavior*, vol. 4095, 2006.

[34] S. Grillner, "Neurobiological bases of rhythmic motor acts in vertebrates," *Science*, vol. 228, no. 4696, pp. 143–149, 1985.

[35] R. D. Beer and J. C. Gallagher, "Evolving dynamical neural networks for adaptive behavior," *Adaptive Behavior*, vol. 1, no. 1, pp. 91–122, 1992.

[36] R. L. Tedrake, *Applied Optimal Control for Dynamically Stable Legged Locomotion*. PhD Thesis, Massachusetts Institute of Technology, Boston, MA, September 2004.

[37] K. Matsuoka, "A mechanical model of repetitive hopping movements," *Biomechanisms*, pp. 5:251–158, 1980.

[38] M. H. Raibert, "Dynamic stability and resonance in a one-legged hopping machine," *4th Symposium Theory and Practice of Robots and Manipulators*, 1981.

[39] K. V. Papantoniou, "Control architecture for an electrical, actively balanced multi-leg robot, based on experiments with a planar one leg machine," in *Proceedings of the International Fed. Automatic Control*, (Vienna, Austria), pp. 283–290, 1991.

[40] K. V. Papantoniou, "Electromechanical design of an actively balanced one leg electrically powered robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Osaka, Japan), 1991.

[41] A. Sato and M. Buehler, "A planar hopping robot with one actuator: Design, simulation, and experimental results," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 4, (Sendai, Japan), 2004.

[42] 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.

[43] M. Ahmadi and M. Buehler, "The ARL Monopod II running robot: Control and energetics," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Detroit, MI), May 1999.

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

[45] 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.

[46] 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.

[47] R. M. Alexander and A. S. Jayes, "Vertical movement in walking and running.," *Journal of Zoology, London*, vol. 185, pp. 27–40, 1978.

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

[49] 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, no. 5, pp. 509–517, 1993.

[50] C. T. Farley and D. P. Ferris, "Biomechanics of walking and running: Center of mass movements to muscle action," *Excrcise and Sport Science Rev.*, vol. 26, pp. 253–283, 1998.

[51] J. W. Hurst, J. Chestnutt, and A. Rizzi, "Design and philosophy of the bimasc, a highly dynamic biped," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Roma, Italy), April 10-14 2007.

[52] P. Holmes, "Poincaré, celestial mechanics, dynamical-systems theory and "chaos".," *Physics Reports (Review Sect. of Physics Lett.)*, vol. 193, pp. 137–163, September 1990.

[53] O. Arslan, U. Saranlı, and O. Morgül, "An aproximate stance map of the spring mass hopper with gravity correction for nonsymmetric locomotions," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Kobe, Japan), May 2009.

[54] A. Degani, S. Feng, H. B. Brown, K. M. Lynch, H. Choset, and M. T. Mason, "The parkourbot - a dynamic bowleg climbing robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Shanghai, China), May 2011.

[55] I. Poulakakis, J. A. Smith, and M. Buehler, "Modeling and experiments of untethered quadrupedal running with a bounding gait: The scout II robot," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 239–256, 2005.

[56] N. Cherouvim and E. Papadopoulos, "Control of hopping speed and height over unkown rough terrain using a single actuator," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Kobe, Japan), May 2009.

[57] J. Seipel and P. Holmes, "A simple model for clock-actuated legged locomotion," *Regular and Chaotic Dynamics*, vol. 12, no. 5, pp. 502–520, 2007.

[58] M. Raibert, M. Chepponis, and J. Brown, H., "Running on four legs as though they were one," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 70–82, June 1986.

[59] S. G. Carver, *Control of a Spring-Mass Hopper*. PhD Thesis, Cornell University, Ithaca, NY, USA, January 2003.

[60] M. M. Ankaralı, *Control of Hexapedal Pronking Through A Dynamically Embedded Spring Loaded Inverted Pendulum Template*. M.S. Thesis, Middle East Technical University, Ankara, Turkey, February 2010.

[61] I. Uyanık, U. Saranlı, and O. Morgül, "Adaptive control of a spring-mass hopper," in *Proceedings of the IEEE International Conference on Robotics and Automation*, (Shanghai, China), May 2011.

[62] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley, June 2006.

[63] 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.

[64] S. G. Carver, N. J. Cowan, and J. M. Guckenheimer, "Lateral stability of the spring-mass hopper suggests a two-step control strategy for running," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 2, p. 026106, 2009.

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

[66] A. Interelectric, "Maxon Motor Catalog," April 2005.

[67] U. Saranlı, A. Avcı, and M. Oztürk, "A modular real-time fieldbus architecture for mobile robotic platforms," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, pp. 916–927, March 2011.

[68] T. F. Coleman and Y. Li, "An interior trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on Optimization*, vol. 6, pp. 418–445, 1996.

[69] H. Rad, P. Gregorio, and M. Buehler, "Design, modeling and control of a hopping robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Yokohama, Japan), July 1993.

[70] G. Zeglin, *Uniroo: A One Legged Dynamic Hopping Robot*. B.S. Thesis, Department of Mechanical Engineesing, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1991.

[71] C. T. Farley, J. Glasheen, and T. A. McMahon, "Running springs: Speed and animal size," *Journal of Experimental Biology*, pp. 185:71–86, 1993.

[72] P. Fihl and T. B. Moeslund, *Recognizing Human Gait Types, Robot Vision, Ales Ude (Ed.)*. InTech, 2010.