University of Bath

PHD

# An Empirical Approach for the Agile Control of Dynamic Legged Robot

Hale, Matthew

*Award date:*
2020

*Awarding institution:*
University of Bath

Link to publication

**Alternative formats**
If you require this document in an alternative format, please contact:
openaccess@bath.ac.uk

**General rights**
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

**Take down policy**
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 11. May. 2021

# An Empirical Approach for the Agile Control of Dynamic Legged Robot Locomotion

submitted by

Matthew Frederick Hale

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Mechanical Engineering

September 2019

**COPYRIGHT**

Attention is drawn to the fact that copyright of this thesis rests with the author. A copy of this thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with the author and that they must not copy it or use material from it except as permitted by law or with the consent of the author.

This thesis may be made available for consultation
within the University Library and may be
photocopied or lent to other libraries for the purposes
of consultation with effect from . . . . . . . . . . . . . . . . (date)

Signed on behalf of the Faculty of Engineering and Design . . . . . . . . . . . . . . . . . . . . . . . . . . .

# Abstract

The use of robotic legged locomotion offers great potential in terms of the ability to traverse discontinuous or rough terrain, inaccessible to wheeled or tracked robots, because feet require only intermittent support. In order to cross a complex environment with limited safe foothold positions, a legged robot must be able to accurately control the length of each step independently. For a hopping or running robot, this requires the precise control of both apex height and forward velocity, with demand values varying on every step – an agile gait – which is the challenge this thesis contributes towards.

In particular, the transition between the apex of one step and the next is considered as a discrete problem of selecting the system inputs which result in the desired next apex state, consisting of height and velocity. Despite the near ubiquity of legged locomotion in nature, this task has proved difficult in robotic control. This is due to the need for highly dynamic gaits, away from the simplifying assumptions of static balance, and the use of spring-like mechanics to store and re-use energy between steps. The Spring Loaded Inverted Pendulum model is widely used as an intuitive and simple model which captures the key characteristics of dynamic running and hopping, but even this lacks an exact analytical solution making the control task difficult.

The approach taken in this work is to formulate a simple controller, based on simple analytical analysis, but select the gain values within this controller by using the results of previous steps, either pre-computed or on-line based on the preceding steps. It is shown to be as accurate or better than the available analytical approximations from the literature when applied to the Spring Loaded Inverted Pendulum model, whilst remaining computationally inexpensive – an important advantage over numerical methods for real-time implementation. Furthermore, by tuning the controller on-line based on previous steps, this approach has the benefits of reducing the manual effort in implementation and automatically adapting to changes in the system dynamics during a robot's operation.

The control tasks for apex height and forward velocity are first considered independently, with the proposed approach applied to each in simulation and using physical experimental hopping robots, before finally they are combined to control both aspects simultaneously in a simulation model.

# Contents

**1 Introduction 1**
1.1 Background . . . 1
1.2 Motivation . . . 3
1.3 Research Question . . . 3
1.4 Contributions . . . 4
1.5 Thesis Outline . . . 4
1.6 Publications . . . 5

**2 Literature Review: Hopping Robots 7**
2.1 Introduction . . . 7
2.2 Walking Robots . . . 7
2.3 Modelling of Hopping and Running . . . 9
2.4 Control of Hopping and Running . . . 12
2.4.1 Finding Stable Gaits . . . 13
2.4.2 Using Analytical Approximations . . . 14
2.4.3 Using Numerical Models . . . 16
2.4.4 Modifying the Model Dynamics . . . 17
2.4.5 Approaches Without Solving Model Dynamics . . . 17
2.5 Physical Hopping Robots . . . 18
2.6 Research Gap . . . 20
2.7 Conclusion . . . 20

**3 Control of Apex Height in Vertical Hopping 25**
3.1 Chapter Introduction . . . 25
3.1.1 Chapter Outline and Contributions . . . 26
3.2 System Design and Simulation . . . 27
3.2.1 Design Requirements . . . 27
3.2.2 Overall System Design . . . 28
3.2.3 Simulation Model . . . 30
3.3 Empirical Controller Derivation . . . 32
3.3.1 Controller Formulation . . . 32
3.3.2 Tuning Parameters On-line . . . 33
3.4 Simulation Results . . . 35
3.5 Physical Experiments . . . 38

3.5.1 Transition to Physical Implementation . . . . . . . . . . . . . . . 38
3.5.2 Physical Robot . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
3.5.3 Experiments and Results . . . . . . . . . . . . . . . . . . . . . . 42
3.6 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

**4 Forward Velocity Control of the Spring Loaded Inverted Pendulum 49**
4.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
4.1.1 Chapter Outline and Contributions . . . . . . . . . . . . . . . . . 50
4.2 System Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.2.1 Description of Model . . . . . . . . . . . . . . . . . . . . . . . . 51
4.2.2 The Control Task: Velocity Control . . . . . . . . . . . . . . . . 52
4.2.3 System Dynamics . . . . . . . . . . . . . . . . . . . . . . . . . . 52
4.2.4 Non-dimensional Equations . . . . . . . . . . . . . . . . . . . . . 54
4.2.5 Simulation Parameters . . . . . . . . . . . . . . . . . . . . . . . . 55
4.3 Existing Approximate Analytical Methods . . . . . . . . . . . . . . . . 55
4.4 Empirical Velocity Controller . . . . . . . . . . . . . . . . . . . . . . . 57
4.4.1 Controller Formulation . . . . . . . . . . . . . . . . . . . . . . . . 57
4.4.2 Numerical Verification . . . . . . . . . . . . . . . . . . . . . . . . 59
4.5 Comparison of Empirical and Analytical Controllers . . . . . . . . . . . 60
4.5.1 Controller Performance Comparison . . . . . . . . . . . . . . . . . 61
4.5.2 Effect of Leg Stiffness Value . . . . . . . . . . . . . . . . . . . . . 62
4.5.3 Agile Gait Control . . . . . . . . . . . . . . . . . . . . . . . . . . 62
4.6 Adaptive Gains Through Multiple Linear Regression . . . . . . . . . . . 64
4.7 Adaptive Simulation Results . . . . . . . . . . . . . . . . . . . . . . . . 66
4.7.1 Agile Control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 67
4.7.2 Adaptation to a Change in System Parameter . . . . . . . . . . . 67
4.8 Physical Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
4.8.1 Robot Design . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
4.8.2 Experimental Testing Procedure . . . . . . . . . . . . . . . . . . . 74
4.8.3 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . 75
4.8.4 Limitations and Failure Modes . . . . . . . . . . . . . . . . . . . 76
4.9 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77

**5 Combined Control of Hopping Height and Forward Velocity 81**
5.1 Chapter Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
5.1.1 Chapter Contributions . . . . . . . . . . . . . . . . . . . . . . . . 82
5.2 Simulation Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 83
5.3 Empirical Controllers . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.3.1 Dataset Generation . . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.3.2 Decoupled Controller . . . . . . . . . . . . . . . . . . . . . . . . . 84
5.3.3 Energy Controller . . . . . . . . . . . . . . . . . . . . . . . . . . 85
5.3.4 Term Selection by Stepwise Regression . . . . . . . . . . . . . . . 86
5.4 Simulation Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 87
5.5 Concluding Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
5.5.1 The Decoupling Assumption . . . . . . . . . . . . . . . . . . . . . 93

5.5.2 Utility of Stepwise Regression Controller . . . . . . . . . . . . . . . 93

**6 Conclusions and Future Work** **97**
6.1 Research Question . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
6.2 Findings . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 97
6.3 Closing Comments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 99
6.4 Future Work . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 100

# Chapter 1

# Introduction

## 1.1 Background

Legged locomotion is an interesting area of research because it presents unique opportunities for robotic capabilities, but also many great challenges in order to realise this potential. The potential is clear from the fact that legged animals can thrive in such diverse terrains across the world, while a wheeled robot will struggle with a flight of stairs. For robots to become more useful, they must be able to traverse complex terrains of unpaved outdoor environments as well as indoor environments designed and built for people. A possible approach to achieve this is to adopt a locomotion strategy similar to the way animals (including people) move. This means robots must not only possess legs with similar mechanical capabilities to ours, but also the equivalent control abilities. They must achieve high efficiency and good foot placement accuracy, to operate for significant periods in unstructured terrain, able to precisely control the speed and direction of every step.

These requirements necessitate a gait that is *dynamic*, free from the limitations of pseudo-static movements. While slow-moving robots can make use of static equilibrium to maintain their balance, a dynamic gait allows for high speed movement without such large energy cost penalty, because spring-like legs are able to store and reuse energy between steps. The robot's gait must also be *agile*, meaning the ability to achieve a new desired apex state for every step. This will allow the robot to control the length of every bound independently, and so to traverse difficult terrain where the safe foot landing positions are not evenly spaced or in a straight line. The challenges in this control problem are significant, in addition to the practicalities of the mechanical design. This is well illustrated by the fact that even the simplest model to reasonably capture the dynamics of running, the Spring Loaded Inverted Pendulum (SLIP), lacks a closed form solution from which to create an exact, analytical controller.

To make some progress towards a vision of useful legged robots, the work presented here is focused on the control of a small, single legged, pneumatically actuated robot,

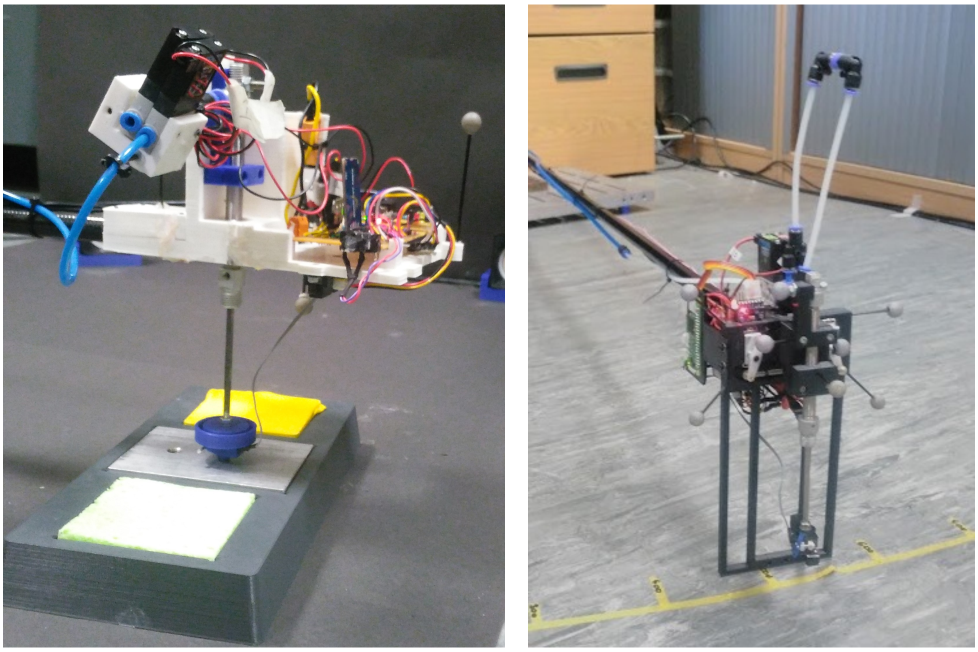

(a) (b)

Figure 1-1: The two custom built pneumatic robotic legged platforms used in this work. (a) A single degree-of-freedom version constrained to hop vertically for 1D experiments. (b) A version with an additional degree of freedom allowing the leg to pivot and constrained to hop sideways in an approximately planar motion for the 2D experiments.

the two versions of which are shown in Figure 1-1. While some way from being able to traverse the same terrain as humans, it has been chosen as a test bed for several reasons. The pneumatic actuation provides a spring-like effect as the air in the cylinder is compressed by impact, while also allowing a convenient method to rapidly inject energy at the desired point in time through opening a valve. The single leg limits the robot to a hopping gait, but this is of little consequence from a control perspective compared to bipedal running, which is also only features a single foot interacting with the ground at a time. The primary difference is that in a biped the each individual leg only participates in alternate stance phases which has the practical advantage of allowing more time for repositioning. Furthermore, animals with more than two legs have also been shown to have similar dynamics, with multiple legs in contact with the ground simultaneously effectively acting as one [1], suggesting wider applicability of results from monoped hoppers.

By definition, hopping and running gaits include a flight phase, during which there is no contact with the ground, and provides a well defined *apex* point of peak height and zero vertical velocity. This makes it convenient to define each hop as transition from one apex state to the next, with the next state determined by the actual dynamics of the system. This allows the control of the continuous dynamics of the system to be abstracted into a discrete problem: at a given apex point, some control input(s) should be determined, to be applied during the upcoming step, in order that the system dynamics result in the desired next apex state. As will be elaborated on later, an apex state is generally defined by its apex height (peak centre of mass distance above ground level) and horizontal velocity.

To address this control problem, the mapping between apex states, as a function of the available inputs, is required. Previous researchers have developed various approximations to the SLIP dynamics for this purpose, but the accuracy of these is limited and deteriorates for more agile gaits. Another approach is to use numerical solutions, which can be made (almost) exact through reducing the calculation timestep, but at the cost of increasing computational expense. The problems with these approaches are compounded by the fact that physical robots will tend to defy the idealised models on which they are based, due to the inherent difficulty in modelling effects such as friction, non-linear actuation, and addition of noise/measurement error.

## 1.2 Motivation

The particular approach taken in this work is motivated by the observation that any controller that is dependent on an idealised model will degrade in performance for an actual system which deviates from this. What is required is a controller which can accurately control these complex dynamics, while remaining computationally lightweight. A further goal would be to avoid being too dependent on the SLIP hopper, but rather able to maintain good performance in the more realistic scenario of dynamics which deviate from the idealised model. To attempt to achieve this, an alternative approach will be taken focusing on a simple controller structure, taking inspiration from the early controllers used by Raibert [2], which will be referred to in this thesis as "empirical" controllers because they do not attempt to solve the dynamics directly. The controller will be formulated through simple approximations to provide the structure of the controller, giving a logical basis in theory, but actual value of the parameters within this controller will be set by examining the results of previously completed steps. The motivation for such a strategy is the assumption that the controller values which would have suitably predicted the preceding hops will make a good choice for controlling subsequent hops. Simulations of simple or fundamental models will be used to develop the controller and compare against the existing analytical approximations. A physical robot will also be developed, to test and demonstrate the controller in more realistic conditions.

## 1.3 Research Question

Given the rational outlined above, the research question to be posed is **"can a discrete feedback controller, derived from approximate dynamic models but tuned**

**on-line, produce accurate hoping control of a small pneumatic robot"**. To answer this question, a self-tuning empirical controller will be developed and applied to the control both apex height and velocity in simulation and a custom built physical experimental hopper. In particular, the objective in developing this controller is to outperform the analytical approximations available, even applied to the SLIP model from which they were derived, while remaining computationally simple enough for on-board implementation on low power hardware. To attempt to achieve this, the controller will be designed to:

- contain only a small number of controller gains, which can be found by fitting previously simulated data, and/or be updated on-line using the preceding steps
- include the ability track rapidly changing (agile) demands, outperforming the available analytical approximations for the SLIP model
- avoid the large computational requirements of numerical approaches through a simple algebraic form

## 1.4 Contributions

- The development of an adaptive empirical controller in the context of the control of a hopping robot
- A demonstration of this controller both for height and velocity control, with a direct comparison against the analytical approximations for controlling the SLIP hopper in simulation
- The design and building of two physical robots, demonstrating one- and two-dimensional hopping control in real time

## 1.5 Thesis Outline

In this thesis, the overall problem of control of a hopping robot is decomposed into controllers for hopping height and forward velocity, first considering each separately and then recombining them. After this introductory chapter and a literature review in Chapter 2, the main structure and contributions are as follows.

**Chapter 3** considers the task of apex height control, often neglected when studying the lossless SLIP model. The primary contribution of this chapter is a simple but accurate controller for the one-dimensional case of vertical hopping which is capable of agile motion with changes in demand for every hop, together with a method for on-line tuning of the control parameters. The controller is demonstrated in simulation and using a physical experimental hopper, tracking large demand variations and automatically adapting to changes in the environment.

In **Chapter 4** a similar approach is applied to velocity control of the archetypal SLIP model for dynamic hopping and running. This contributes a pragmatic novel approach the problem of forward velocity control in running, focusing on agile gaits with random demand values. It combines a low computational cost with good control accuracy, whilst providing benefits in terms of adapting to changes in the system and avoiding the need to manually tune the controller. The proposed method is compared in detail in simulation to existing analytical controller methods, with the comparison of these methods against each other another contribution. Finally a physical experimental robot is built to approximate the 2D SLIP hopper and the empirical controller, including self-tuning, is demonstrated to successfully run on-board in real time.

**Chapter 5** seeks to combine the preceding chapters to control a hopping gait that varies in both apex height and forward velocity, contributing a simulation study into the important question of whether two components can be treated as independent, as is often assumed. It considers, in simulation, three possible approaches: a decoupled controller assuming complete independence, a modified version based on further reasoning about the interacting dynamics, and the extreme case of a controller with a very large number of empirically determined terms.

Finally, **Chapter 6** concludes the thesis, collecting the key findings and discussing possible areas for future work.

## 1.6 Publications

This thesis contains work which has lead to the following publications:

- M. F. Hale, J. L. du Bois, and P. Iravani, “A comparison of analytical and empirical controllers for the slip hopper,” in *18th Annual Conference on Towards Autonomous Robotic Systems, TAROS 2017*, pp. 79–85, Springer Verlag, 2017.
- M. F. Hale, J. L. Du Bois, and P. Iravani, “Agile and Adaptive Hopping Height Control for a Pneumatic Robot,” *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5755–5760, 2018.
- M. F. Hale, and P. Iravani. “An Adaptive, Low Computational Cost, Hopper Controller”, in peer review at time of writing.

# Chapter 2

# Literature Review: Hopping Robots

## 2.1 Introduction

This chapter will examine some of the previous related research. Legged robots can be classified by the number of legs and the type of gait they use. Some multi-legged and static walking robots will be briefly described for context, but the focus of this review will be hopping robots aimed at the dynamic gaits discussed in the previous chapter. Within these dynamic locomotion strategies, there a lot of focus on (asymptotic) stability of steady gaits (e.g. [3–8]), where each step is the same length.

In order to realise the potential benefits of legged robots and traverse more difficult terrain, such as limited, unevenly spaced footholds, the length of each step must be controlled independently. Therefore a new target apex state to be reached in a single hop. A gait that requires a different target state on each hop, as opposed to approaching a steady state over several hops, will be termed "agile", and particular emphasis will be given below to controllers aiming to achieve this.

Therefore, this chapter will be divided into some background in walking robots in Section 2.2, before discussing the modelling of hopping and running in Section 2.3, its control in Section 2.4 and some examples of physical experimental hoppers are given in Section 2.5. The research gap explored in this work is discussed in Section 2.6, with some conclusions in Section 2.7.

## 2.2 Walking Robots

There has long been a desire to create machines which use legs, but the obvious challenge is to how to make them to move without falling over. This problem has been

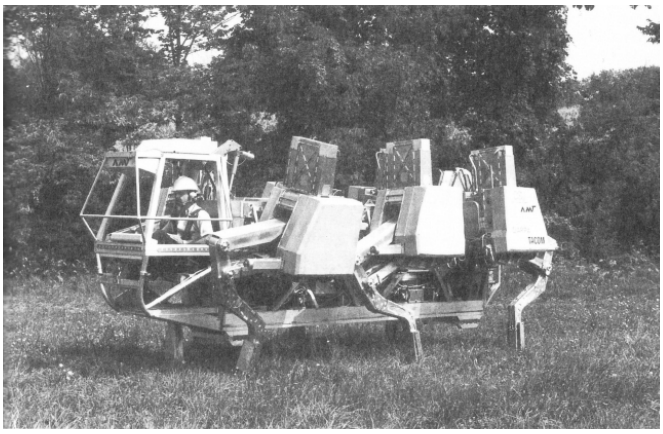


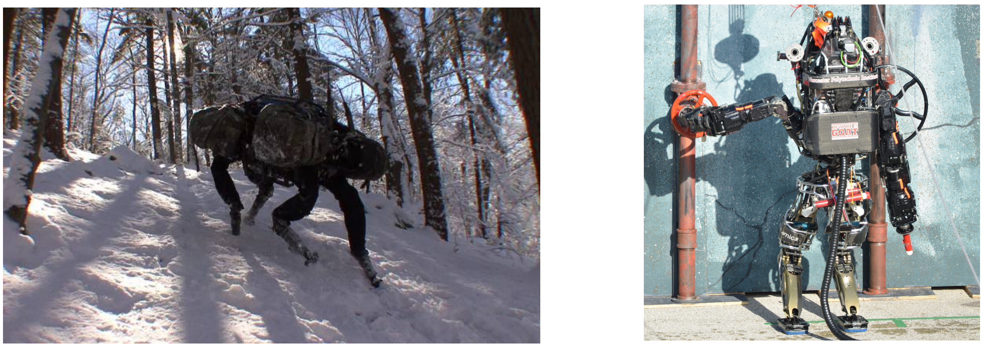
Figure 2-1: The Adaptive Suspension Vehicle which employed a statically stable alternating tripod gait, with direction set by a human driver [12]

under consideration since at least the 1960s, when McGhee and Frank described stable "creeping" gaits for what they term "artificial quadrupeds" but we might consider robots [9]. This involved keeping at least three feet in contact with the ground and the centre of mass over the resulting support polygon at all times, and assuming the movement is slow enough that the inertial forces can be ignored. This kind of gait was employed in early legged robots, such as the Adaptive Suspension Vehicle [10] shown in Fig. 2-1, and still in use in current research, thanks to it's reliable stability (e.g. [11]).

Static gaits are simpler to compute and control, but in order for legged robots to compete with wheeled or tracked methods, greater speed and energy efficiency are required [13]. This can only be achieved by dynamic gaits, defined as those where the movement is such that the inertial forces are significant – so that there are points in the gait where should all the joints suddenly become locked, the robot would fall over.

Work at Honda (which would later became the famous ASIMO robot) was an early attempt to make progress in this direction [14]. Their strategy was to take recorded motion from human demonstrations, and play them back on their humanoid robot, modulated by Zero Moment Point (ZMP) control to help maintain balance. ZMP is an application of techniques from static walking to moderately dynamic motions by including the effects of inertial forces, so that instead of the centre of mass remaining directly above the foot support polygon, the "centre of pressure" must fall within it. This has been successful for reasonable walking speeds and traversing stairs but still requires large feet and high-gain position feedback control which limits the speed and efficiency possible.

Another approach to the control of dynamic walking is that of "Virtual Model Control", employed on the Spring Flamingo [15] and related robots. Forces are computed based on the interaction of virtual components, such as the "granny walker" spring damper system to keep the body upright. The actual forces must be applied through the real legs of the robot, and so the scheme requires constant contact with the ground



(a) (b)

Figure 2-2: Two walking robots made by Boston Dynamics: (a) BigDog [17] and (b) ATLAS used as a research platform and shown competing in the DARPA Robotics Challenge [18]

through large feet. Through various algorithms to select and tune the various virtual components, steady state walking over rough ground and up and down slopes was achieved [16].

Various fully actuated walking robots and their control continue to be an area of active development, notably those build by Boston Dynamics, such as BigDog [17] and ATLAS [18], shown in Fig. 2-2. A focus on practical application to real world has been driven by private investment and the DARPA Robotics Challenge in 2013 and 2015, although challenges in autonomy in terms of energy and control remain. Boston Dynamics have recently produced some exciting videos of what appears to be dynamic walking and trotting, but as a private venture there is little available detail on how this is achieved [19].

## 2.3 Modelling of Hopping and Running

While dynamic walking offers many improvements over statically stable gaits, animals demonstrate much more impressive hopping and running gaits. These are defined by the introduction of a phase without any contact with the ground, and offer the opportunity to take advantage of many of the potential benefits of legged robots in terms of traversing discontinuous terrain with large gaps at high speed, but precise control over velocity in flight is required.

In hopping, as opposed to running, the same foot is used in each stance phase. This makes such gaits by their nature highly dynamic, requiring a long flight phase to reposition the foot in order to make progress, and it is common to also rely only on dynamic stability even during the stance phase, with the foot approximating a point contact that cannot apply any ankle torque. These dynamics are complex, and there is a diverse range of animals (and to some extent robots) that apply these gaits, but a

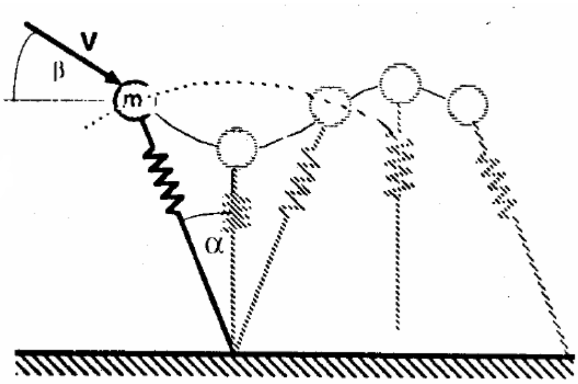


Figure 2-3: The Spring Loaded Inverted Pendulum (SLIP) model [20]

surprisingly simple model has been found which encapsulates much of this complexity. The Spring Loaded Inverted Pendulum (SLIP) [20], shown in figure 2-3, has become the standard model for investigating monopod locomotion in nature and robotics. Inspired by the observation that the legs of running animals compress like a spring, this model consists of a single point mass representing the body, with the leg modelled as a simple massless spring. While not in contact with the ground (flight phase) the body moves along a ballistic path, and when the foot contacts the ground (stance phase) the spring is compressed and the body pivots about the foot. During stance, the only forces on the body are the spring force and gravity.

A review of the literature has found that the SLIP model, or variants of it, is dominant for the simulation of hopping and running robots. The remainder of this section will discuss some of the many studies making use of the model, and the various additions and alterations to the basic model they have made, as summarised in Table 2.1.

One shortcoming of the SLIP model is that it is energy conservative, which is clearly not realistic as any physical system will experience energy losses. A popular way to model this is to add a linear damper into the leg, in parallel with the spring [23, 30, 46, 47]. This is a simple friction model, which is straightforward to implement, adds a single extra parameter and is easy to understand. Losses can also be introduced by including a mass at the foot [35, 36], so the kinetic energy associated with it is lost at the moment of impact when the foot immediately comes to rest. This has some physical plausibility, and the modelled foot mass can be given a realistic value, but perhaps does not encapsulate all the energy losses.

The SLIP model can also be extended by adding actuation, which otherwise acts passively during the stance phase. This can be linked to the introduction of losses, since any non-conservative model will require actuation to compensate for losses and allow the hopping to continue. Actuation can also be added to a lossless model, however, to move the hopper between energy levels or reject disturbances that change the energy level, such as ground height changes. There are various forms such actuation can take. An actuator can be added in series with the leg spring (between it and the

Table 2.1: Various studies which have used the SLIP model, comparing the modifications and additions each has included.

| | **Actuation** | | | | | **Losses** | | |
|---|---|---|---|---|---|---|---|---|
| | Leg force (series) | Leg force (parallel) | Hip joint actuation | Differential leg stiffness | Differential leg length | Linear leg damper | Unsprung foot mass | **Other additions to model** |
| Ahmadi, 1997[21] | ✓ | | | | | | | Body rotation |
| Hyon, 2002[4] | | ✓ | ✓ | | | | | Body rotation |
| Schmitt, 2006[22] | | | | | | | | |
| Seipel, 2007[23] | | | ✓ | | | ✓ | | |
| Schmitt, 2009[24] | | ✓ | | | | | | |
| Carver, 2009[25] | | | | ✓ | | | | Extended to 3D |
| Poulakakis, 2009[26] | | ✓ | | | | | | Body rotation |
| Tamaddoni, 2010[27] | | | | | | | | |
| Karssen, 2011[28] | | | | | | | | Non-linear leg spring |
| Karssen, 2011[29] | | | | | | | | |
| Uyanik, 2011[30] | | | | | ✓ | ✓ | | |
| Vejdani, 2012[31] | | | | | ✓ | | | |
| Vejdani, 2012[31] | ✓ | | | | | | | Leg inertia in flight, limited motor torque |
| Yu, 2012[32] | | | | ✓ | | | | |
| Piovan, 2012[33] | ✓ | | | | | | | |
| Arslan, 2012[34] | | | | ✓ | | | | |
| Rutschmann, 2012[35] | ✓ | | | | | | ✓ | |
| Byl, 2012[36] | ✓ | | | | | | ✓ | |
| Wensing, 2013[37] | | | | | | | | Extended to 3D |
| Wu, 2013[38] | | | | | | | | Extended to 3D |
| Haitao, 2013[39] | | | | | | | | Non-linear leg spring |
| Piovan, 2013[40] | ✓ | | | | | | | |
| Palmer, 2014[41] | | | | | | | | |
| Dadashzadeh, 2014[42] | | ✓ | ✓ | | | | | |
| Shemer, 2014[6] | | | | | | | | |
| Piovan, 2015[43] | ✓ | | | | | | | |
| Piovan, 2016[44] | ✓ | | | | | | | |
| Calisti, 2016[45] | ✓ | | | | | | | Hydrodynamics effects, visous losses |
| Shemer, 2017[46] | | | | | ✓ | ✓ | | |
| Secer, 2018[47] | | ✓ | | | | ✓ | | |

body mass), in which case the input is defined by the displacement of this actuator over time [21, 31, 33, 35, 36, 40, 43–45]. Alternatively, an actuator can be added in parallel with the leg spring, with the additional force specified [26, 47]. In either case, it is easy to envisage the physical application, with either position controlled or force controlled actuators integrated into the leg design. Another method to add energy is to allow the leg to take a different length at touchdown compared to liftoff [30, 31, 46]. This can be achieved in practice by pre-compressing the leg spring during flight, with some form of latching mechanism to allow this energy to be released during the subsequent stance phase. A somewhat less physically plausible method to add energy is to allow the leg spring to take a different stiffness value when compressing compared to decompressing, with an instantaneous change in stiffness at the point of maximum compression [25, 32, 34]. The above actuators have all acted linearly along the length of the leg, but it is also possible to apply a torque at the hip joint during stance in order to add energy, either in combination with a leg actuator [4, 42], or alone [23]. This makes good sense as the leg angle will need to be controlled during flight in any case, in practice requiring actuation, and it may also be mechanically more straightforward than a linear actuator.

The SLIP dynamics can also be extended in several other ways. The point mass of the body is a clear simplification, and the rotation of the body can be added as an additional degree of freedom to be controlled [4, 21, 26]. The spring can also be modified to follow some non-linear dynamics [28, 39], which may change the behaviour and stability of the system. Considering the lateral motion of the robot extends the model into three dimensions, a key consideration for more practical robots [25, 37, 38]. Finally, Calisti *et al.*have taken a unique direction, considering a underwater robot hopping along the bottom by adding hydrodynamic effects, including drag on the body, to the SLIP model [45].

In several cases, the gait of a SLIP hopper has been used in order to control a more complex morphology, where the more complex robot is forced to track the path predicted by the simple model [27, 37, 40, 47]. This is sometimes called using the SLIP as an anchor, and shows the importance of controllers for the model, and their potential for wider utility. Overall, it is clear that this is a very widely used model, the control of which will be important for future robotic applications.

## 2.4 Control of Hopping and Running

The use of spring-like legs is likely to be required in order to achieve the efficiency needed for mobile robot applications, but this same compliance further complicates the control problem. The seminal work work of Raibert [2] in this direction has inspired various strands of research in the subsequent decades, but these robots remain limited to research labs (or, in many cases, to simulation).

Despite the apparent simplicity of the SLIP model, there is no analytical solution to the stance phase dynamics and must be solved numerically. This has been the primary obstacle for controllers, which must predict what input values (usually touchdown angle and/or force input during stance) will produce the liftoff conditions required to achieve a desired trajectory. To simplify the control problem, Raibert proposed a strategy to separate it independent sub-problems, to create the "three part controller" [48]. Firstly, forward velocity is controlled by the placement of the foot relative to the body at touchdown (*i.e.* the touchdown angle). Secondly, the body attitude is corrected by applying a hip torque during stance to remove unwanted angular momentum, Thirdly, the apex height is controlled by leg thrust. The decoupling assumption is that these three problems can be considered can be assumed independent, so they can be derived and computed separately. The majority of work since has continued with this assumption, although it is often not explicitly stated. This is in no small part due to the use of the SLIP model, which is used to investigate forward velocity control, but its lossless nature means height control can be ignored and a point mass body means there is no body orientation to consider. In order be applied to physical robots, velocity a controller for the SLIP model will need to be applied in conjunction with one for apex height and possibly body attitude.

The following subsections will review some proposed controllers for hopping, organised by some overall themes to the approaches that have been taken. These controllers are equally applicable to monopod hopping and biped running, which both feature alternating between a single foot stance and a flight phase. In general, the task these controllers attempt to solve is to select the leg angle at touchdown in order that the stance dynamics (usually assumed to be the SLIP model) lead to the desired forward velocity on the next flight phase.

### 2.4.1 Finding Stable Gaits

Seyfarth *et al.* [49] demonstrated the existence of stable running gaits for certain combinations of leg stiffness and forward speed, where a constant touchdown angle will allow the hopper to naturally recover from small deviations. The stability of running gaits is expressed by computing the apex-to-apex return map, which plots the next apex state as a function of the current apex state, which provides a convenient graphical representation to examine stability, such as those shown in figure 2-4.

These stable gaits have the obvious attraction of not requiring a controller to compute the touchdown angle, as it is the same for each step, however the gaits must be carefully chosen to be one of these particular stable combinations of parameters. Additionally, the initial conditions must not fall outside the stable region, and disturbances must be small enough to be rejected. To achieve gaits free from these restrictions, a controller is required in order to select the appropriate touchdown angle, given the current state.

An extension to this idea is that of "Controlled Passive Dynamic Running", where the robot is designed to hop passively, but feedback control around this point is applied encourage stability as well as make up for energy losses [51].

One method to increase the basin of attraction (the region of state space where the

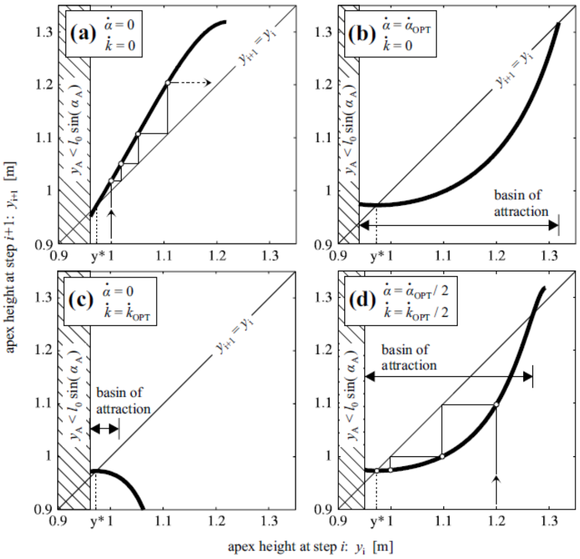


Figure 2-4: Some examples of (a) unstable and (b-c) stable apex return maps [50]

system can be started and will converge to the stable fixed point) is “Swing Leg Retraction” (SLR) (e.g. [50, 52]). In this method, the leg angle is varied as function of time during stance, which effectively results in the angle at touchdown being determined by the apex height, which is directly linked to time in flight phase. This has been shown to not only enlarge the basin of attraction but also produce faster convergence to a stable steady state hopping and stable gaits that otherwise are unstable [52]. The speed at which the leg should be retracted is generally found through numerical simulations, although some work has looked at producing analytical methods for selecting these parameters [6].

### 2.4.2 Using Analytical Approximations

To assist in the development of controllers, multiple researchers have developed approximations to the stance dynamics that do have closed form solutions.

For example, Geyer *et al.* have derived an approximation assuming small angular sweep and small spring compression [53]. Another analytical approximation from Saranl *et al.* [54] assumes conservation of angular momentum, which requires that the effect of gravity on the rotation are neglected.

Yu *et al.* have produced a more accurate approximation using a truncated Taylor series expansion [55]. This again makes the assumption that the spring compression is a relatively small fraction of the leg length (which will make the approximation more applicable to a stiff leg spring), but the more elaborate mathematical treatment is shown through simulation to perform better than previous approximations for larger touchdown angles and in particular asymmetric cases which are inherent to large velocity changes and so required for agile gaits. Subsequently, the same authors applied this approximation to create a controller and tested in simulation, and is shown to successfully vary the forward velocity of a SLIP model on level ground [32].

Another approximation was derived by Shen and Seipel [56] using the a assumptions that "gravity acts mostly along the leg" and "denominators involving leg length can be approximated as constant". This has broken the general chronological pattern of increasing algebraic complexity for more accurate results over time, instead giving a highly simplified set of equations but with the advantage of an entirely linear approximation to the stance phase which capture the qualitative characteristics of the SLIP model. Although the stated aim of this piecewise-linear formulation is to allow for the application of controllers, this has not so far been demonstrated.

There have also been many studies, focused primarily on implementation, that have used control strategies derived from more crude approximations. One approach is to simply neglect the gravity term during stance, making the dynamics analytically solvable [7, 57]. An even simpler approach is the "Ball-Hopper" or "instantaneous SLIP" model, where the leg stiffness is taken as infinite, e.g. [5, 6, 34, 58]. The result is an instantaneous stance phase and the robot rebounds as one would expect of an ball bouncing off a hard, angled surface.

A controller based on the instantaneous impact model was used "with ad hoc but physically motivated corrections" by Zeglin and Brown [59]. This simplified model is used for a graph search to find suitable trajectories, creating a feedforward controller, and is combined with a feedback controller to keep the robot close to the computed trajectory. The graph search, being computationally expensive, is not performed on every hop, but only only if the actual position becomes too far from the desired trajectory, in which case a new trajectory must be computed starting at the current position.

Arslan and Saranl [34] utilised the Ball-Hopper model to create multi-step plans over height varying terrain with gaps, combined with their own more complex approximation [54] in the step-to-step control of touchdown angle for a simulated SLIP hopper. The resulting controller must re-plan on each step to make up for the differences between the models.

Shemer and Degani[46] used the instantaneous SLIP model with the addition of a damping term to create a controller they call "polynomial energy insertion". This is aimed at stabilising a constant forward velocity subject to changes in the ground height, requiring deadbeat control to reject changes in a single step. In their experimental robot, the *ParkourBot*, the amount of energy added during a stance phase is determined by retracting the leg during the preceding flight and they found an approximately linear relationship between apex height and the required leg compression. This results in a polynomial relationship between flight time from apex to touchdown, from which the method takes its name.

All these approximations are most accurate for small touchdown and liftoff angles, and for symmetrical gaits (where these two angles are similar in magnitude). This means they are suitable for controlling steady state gaits at relatively low speeds, but performance rapidly degrades for more agile motions, with large changes in speed requiring large angles and an asymmetric stance phase.

### 2.4.3 Using Numerical Models

Another approach is use a numerical simulation to solve the stance phase dynamics, possibly for several steps into the future, and selecting inputs which minimise some cost function in this simulation. This is known as Model Predictive Control (MPC). For example, Rutschmann *et al.* [35] investigate the critical choice of the number of future steps to simulate, which is a compromise between accuracy of the controller and time to execute because of the computationally intensive numerical simulation. Using a desktop computer, they find the MPC optimisation requires in the order of a few hundred milliseconds (carried out during the flight phase) to achieve foot placement within a centimetre of target.

On-line numerical simulations are used by Piovan and Byl [43] to choose actuator displacement during stance phase for a simulated SLIP hopper, in order to reach a desired liftoff state. Since the controller is to be operated in real time during the stance phase, a low order integration method is used which is fast to compute, accepting a reduction in accuracy.

In a recent extension into a three-dimensional simulation, Wu and Geyer [38] numerically simulated the system to find a discretised solution for the result of all possible landing angles. They state that this computation took "less than a day on a modern desktop PC". This then effectively becomes a lookup table, with the controller interpolating between points to achieve a desired next apex state. Their simulations demonstrated robustness to ground height disturbances of up to 30% of leg length.

Offline simulation of the SLIP dynamics was also used by Wensing and Orin [37, 60], in this case to find periodic gaits which were then applied to a more complex humanoid model in simulation. The robot is stabilised using linearised models around these periodic gaits, again computed numerically offline. An obstacle to the physical implementation of strategies using prior simulations to create lookup tables or linearised controllers is the reliance on a highly accurate simulation in order for the result to be applicable to the real world. So, although still only in simulation, the fact a controller derived from the SLIP model was successfully applied to a much higher degree of freedom biped is in itself an important and promising result, and supports further work on SLIP hopper control.

Another approach for implementing the results of off-line simulation is to fit an approximated function to them, allowing for fast subsequent on-line calculation using this function. Yim and Fearing [61] implement this, using a Taylor series polynomial fitted to a simulation of their physical SALTO robot. This work is notable for considering both the leg force input and touchdown angle in three-dimensional hopping

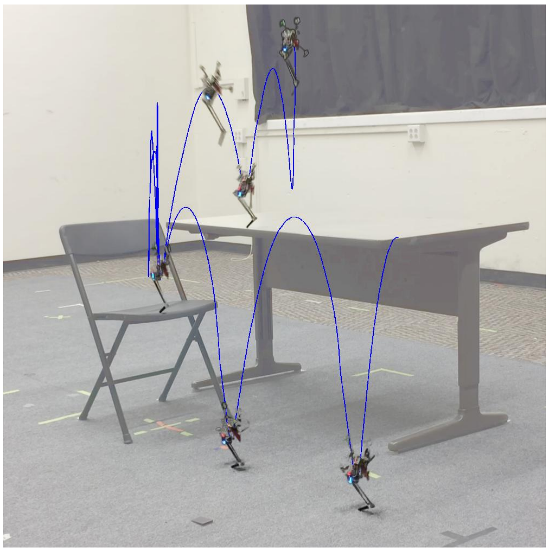


Figure 2-5: Impressive agility demonstrated by the SALTO hopper, controlled based on a function fitted to pre-computed simulation data [61]

and for highly agile hopping, and particularly for the remarkably impressive results demonstrated on a physical robot, such as in Fig. 2-5.

### 2.4.4 Modifying the Model Dynamics

Piovan and Byl [33, 40] have taken an unusual approach of "Enforced Symmetry", whereby instead of attempting to solve the SLIP dynamics, the non-linear term is instead cancelled out by providing an input to produce an identical but opposite force. Then an exact analytical solution for leg length during stance can be found, although the angular displacement must still be approximated. They have applied this approach in simulation to an actuated SLIP hopper, maintaining a steady gait over uneven terrain or after perturbations.

### 2.4.5 Approaches Without Solving Model Dynamics

The approaches above have taken different approaches to the thorny problem that the stance phase SLIP dynamics have no closed form analytical solution, making control of this model difficult. However, Raibert demonstrated computationally simple controllers (a strict requirement given 1980s computer hardware) successfully controlling dynamic gaits in real time without considering the SLIP model dynamics at all – in fact the SLIP model was only proposed by Blickhan [20] a few years after Raibert's classic book [2].

For forward velocity control, this was achieved by estimating the "CG-footprint", the section of ground covered by the centre of mass during stance and placing the food in the middle of this in order to produce a symmetrical stance phase and so maintain a steady velocity. The foot position was moved back or forward from this point by a proportional term acting on the error in velocity, with the gain tuned manually and results showing the velocity converge on the demand within a few steps [2]. Although Raibert's controller has sometimes been applied directly, such as in the recent SALTO robot [62], this overall approach has largely been neglected by researchers in the years since.

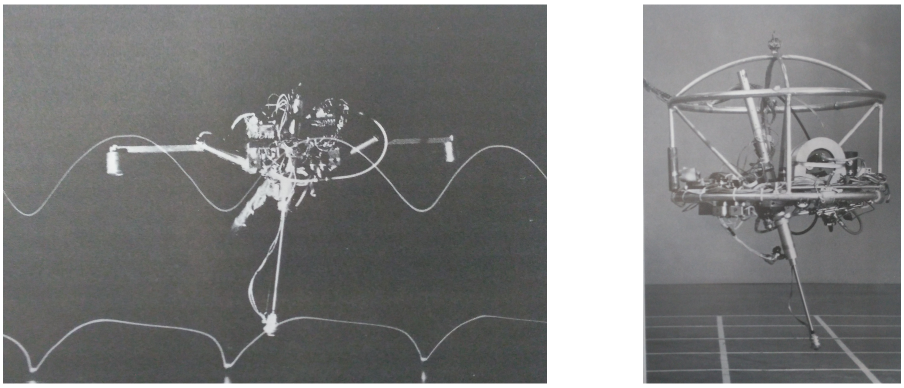
One other technique that has avoided attempting to directly solve the SLIP dynamics is the "Biologically Inspired Deadbead" (BID) controller. Inspired by the overall shape of the time plots for the ground reaction force in biological data, Englsberger *et al.*[63, 64] take the approach of fitting low-order polynomials to the force (and thus motion) profile expected during stance. These polynomials can be solved by setting the boundary conditions based on the desired future states, allowing for an analytical solution. The advantage of this approach is its biological plausibility, and that it avoids the difficult problem of solving the SLIP dynamics. However, this model does not take into account any spring-like action in the leg, but instead the required force profile is determined by the controller and (in simulation) applied entirely by the leg actuators. For practical implementation, this would require very high-speed force control, and does not utilise the efficiency advantage offered by elastic energy storage and release.

## 2.5 Physical Hopping Robots

In order to validate simulations and test control strategies, physical robots must be built, but of course with the addition of real-world effects such as friction and sensor noise. The have been various research robots built for this purpose [65]. Below are described some of the most important which take a similar form to the SLIP hopper model, for the purpose of testing and verifying hopping control strategies.

In his pioneering work, Raibert built two-dimensional and three-dimensional hoppers [48], shown in figure 2-6. These operated using a pneumatic leg, with the two-dimensional version constrained by a long boom. The body was designed to have a large inertia compared to the leg to allow a hip actuator to move the leg while in the air, without too much effect on the body orientation, a feature seen in many later robots also. Power and connection to an external control computer was provided via an umbilical cable.

Brown and Zeglin developed the "Bow Leg Hopper" in the late nineties [58], shown in figure 2-7. In this design the leg comprises a spring, shaped like an archer's bow, in which energy can be stored during the flight phase by a string which is mechanically released upon touchdown. Actuation is provided by electric servos. The robot is attached to the end of a boom, to create a planar motion, and a bungee cord pulling the boom upwards reduces the effective gravity experienced by the robot. This leg actuator design was further developed and integrated into the "Parkourbot" design, which has



(a) (b)

Figure 2-6: Raibert's hoppers, (a) two-dimensional version and (b) three-dimensional version.

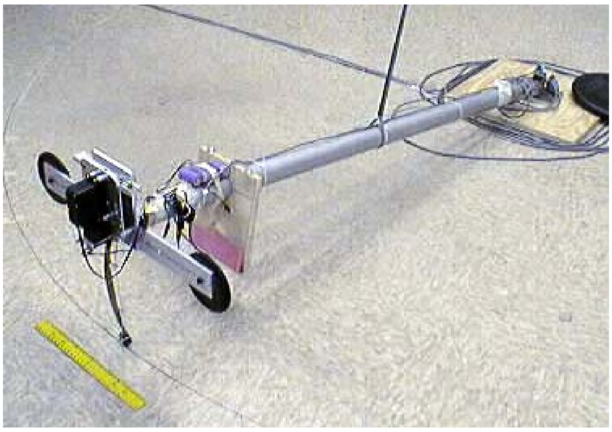


Figure 2-7: The Bow Leg Hopper [58], photo from [66]

been demonstrated operating on an inclined table to create a two dimensional reduced gravity environment [5].

The ARL-monopod [67] and later ARL-monopod II [51, 68], shown in Fig. 2-8, are electrically actuated planar hoppers. The later design was refined using leg and hip springs to dramatically improve the energy efficiency by exploiting the natural dynamics of the robot. The controller attempts to keep the robot in the centre of a treadmill to simulate maintaining a forward running speed. The behaviour for such a system are similar to running on fixed ground for steady state running, but the transient dynamics of changing speed or direction are not well represented.

The ATRIAS monopod is deliberately designed to mimic the SLIP dynamics by concentrating the mass at the body and having a sprung leg. Initially a monopod was build [69], and developed into a biped runner [70, 71].

In just the last few years, the SALTO robot shown in Fig. 2-9 has demonstrated some very impressive developments. The robot is powered in stance by an electric motor connected via an elastic element to a carefully designed leg linkage mechanism which modulates the power output over the stroke [72]. The lightweight design has achieved impressively large hops over 1 m high with a leg length of 14.4 cm [62]. Orientation in flight is controlled via small thrusters and a reaction wheel.

## 2.6 Research Gap

This review has shown that the control of hopping, particularly the SLIP model, is an important area of research in order for legged locomotion in robots to progress. Previously proposed controllers have been based around solutions to the SLIP dynamics, either through approximation or numerical methods. However, such methods rely not only on the accuracy of the approximation, but also the system's adherence to the idealised SLIP dynamics in order to be successful – physical robots will inevitably deviate from this, and controller performance will suffer. This leads to the **research gap** identified for this work: an alternative approach to the controller would be to select or tune controller parameters based on already completed steps. Thus, the **novelty** of this work lies in the development and application, in simulation and hardware, of an adaptive controller like this to the field of dynamic hopping control, which has not been previously investigated.

## 2.7 Conclusion

The SLIP model has been extensively used by many researches because it captures the fundamentals of legged running in a simple model. Despite this simplicity, a complete solution to the dynamics is not available, and approximations are often used. The use-

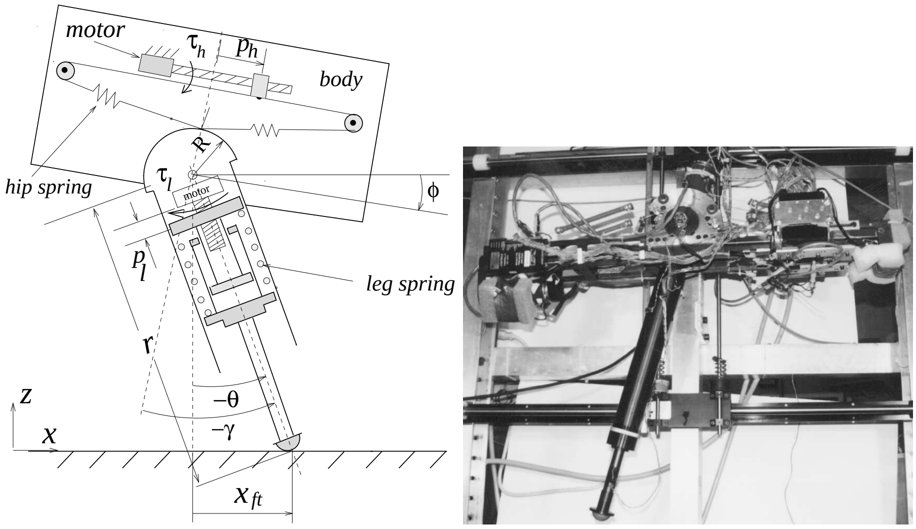

motor
τh
ph
body
hip spring
R
τl
motor
ϕ
leg spring
pl
r
−θ
−γ
z
x
xft

Figure 2-8: The ARL-monopod II [68].

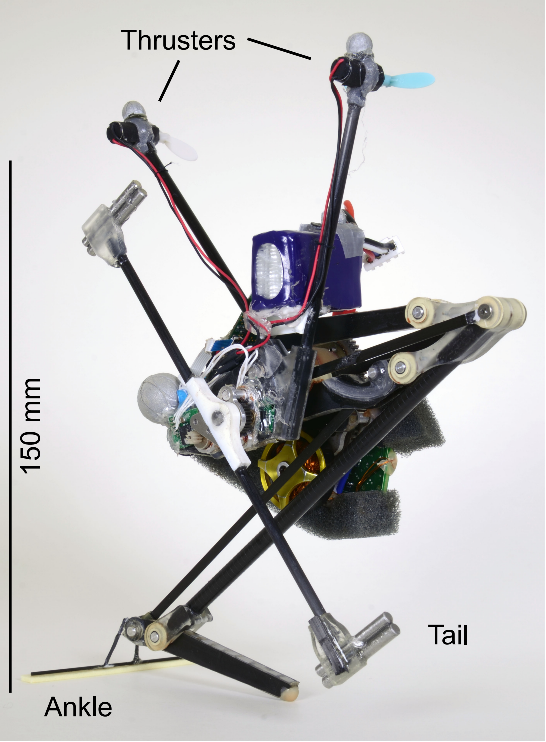

Thrusters
150 mm
Tail
Ankle

Figure 2-9: The SALTO hopper [62]

fulness of this model and the control strategies developed for it have been demonstrated in real hardware, primarily in two-dimensional hoppers.

The available approximations are most accurate for small angles and symmetrical gaits, and most research has been concentrated on maintaining stable steady gaits. For more agile motion, with large angles and an asymmetric stance phase, the non-linear terms become more important and the approximations are less accurate. Recently, improvements in computing power have inspired attempts to perform real-time control using numerical simulations. Difficulty in implementing this using on-board hardware leads to the use of pre-computed simulations saved as lookup tables or fitted functions.

There is scope to further develop control strategies in particular for agile hopping, where the robot can vary its forward speed on every hop. In order to be of use in real systems, these must be simple enough to operate in real time on hardware that can be carried on-board a robot. Building on the work described above, this may be be approached by attempting to find more accurate approximations, or use numerical simulations with larger timesteps. However, the approach in this thesis has been inspired by how much Raibert's hopping controller achieved without this kind of analysis of the SLIP model. Similar simple controllers will be derived, exploring a new approach how they can be tuned and updated in real time. The resulting controller will seek to overcome or avoid the primary weaknesses of the other approaches, namely:

- provide good accuracy, even for agile gaits, to outperform analytical approximation methods
- be low in computation cost to provide a benefit over on-line numerical approaches
- use automatic tuning to make for simpler real-world implementation where system dynamics and parameters are uncertain

# Chapter 3

# Control of Apex Height in Vertical Hopping

Consider yourself (or a robot) running across a rubble-strewn environment, where each footstep position must be controlled to avoid dangerous material or gaps in the surface. In order to do this, the distance between each step must in turn be controlled. For instance, imagine that, while in the air, you see that your next bound needs to clear a particularly large gap. You must ensure your foot strikes the ground at the correct angle and subsequently apply the correct force during the contact (stance phase), in order to cover a greater distance over the next bound (flight phase) compared to the current one. It is easy to imagine that instinctively you are likely to attempt to increase both forward velocity and apex height in order to cross the large gap. This is in contrast to the typical SLIP model, where its energy conservative nature means that increasing speed necessarily causes a decrease in apex height. Therefore, when considering agile motion such as this, it is logical consider not only forward velocity but also apex height control, introducing a model with variable system energy and losses.

This chapter is based around the development and testing of a physical, single degree of freedom robot model, its simulation counterpart, and the application of a novel control strategy to it. This serves as a test-bed for the height control portion of the more general problem of agile legged locomotion.

## 3.1 Chapter Introduction

The chapter, which describes work presented at the IEEE International Conference on Robotics and Automation (ICRA) [73], considers the problem of apex height control in isolation. Thinking generally of any running system, a particular step can be characterised by the horizontal (forward and/or sideways) velocity, and the peak height – the apex height – with the distance travelled in a that step linked to both these parameters. Velocity clearly has a direct relationship to the step length, but velocity control alone is not sufficient. A higher apex will lead to a greater time in flight, causing more distance to be covered and a longer step length, therefore accurate control of apex height will be not only useful, but a requirement for accurately landing on footholds which vary in height and/or separation. Even for flat terrain, changes in footstep length can be controlled through either modulating the forward velocity or the apex height [74]; in practice is is likely that varying both forward velocity and apex height will allow for greater agility and a larger range of step lengths to be achieved. It is therefore important apex height control is not neglected.

The emphasis on control of the lossless SLIP model, while a very useful model of hopping and running, avoids the need to actively control the apex height, because the constant energy level means it becomes a simple consequence of the velocity. In real robots (and non-conservative models) there will be energy losses which would lead to hop height decaying if not compensated for. In order to approximate the idealised lossless dynamics, actuation must be used with the controller estimating the energy loss and attempting to replace it. In addition, for agile motion, the controller must determine the amount of energy to add or subtract to reach the demanded apex for the next hop.

The task of hop height control will be investigated here for single degree-of-freedom, one-dimensional hopping, *i.e.* hopping vertically with no forward motion. These dynamics are clearly simpler than when forward motion is included, however starting with this simple model will allow the control approach to be developed and demonstrated on this sub-problem (of height control) before considering the more complex task of hopping in two dimensions. It is assumed that the height controller can later be combined with a forward velocity controller (such as that in Chapter 4), with independent inputs controlling each; an assumption explored later in Chapter 5. This is sometimes called the "decoupling assumption", and has been a common feature of legged robot control since the Raibert's "three part controller"[48].

The physical experimental robot, shown in Fig. 3-11, comprises a simple pneumatic actuator used to form the leg, constrained on a boom to hop vertically. Compression of the air in the actuator creates a spring-like effect to store and release energy from one hop to the next. A simulation of this robot, with a simple gas compression model, is first created to develop and test the controller.

### 3.1.1 Chapter Outline and Contributions

This chapter contributes the derivation and testing, in simulation and hardware, of an empirical controller applied to the control of apex height in a hopping robot. Furthermore, an on-line adaptive method will be demonstrated to allow the controller to select the parameters with reduced manual effort and automatically update to changes in system dynamics. The remainder of the chapter is laid out as follows:

- Section 3.2 describes the system model and problem under consideration

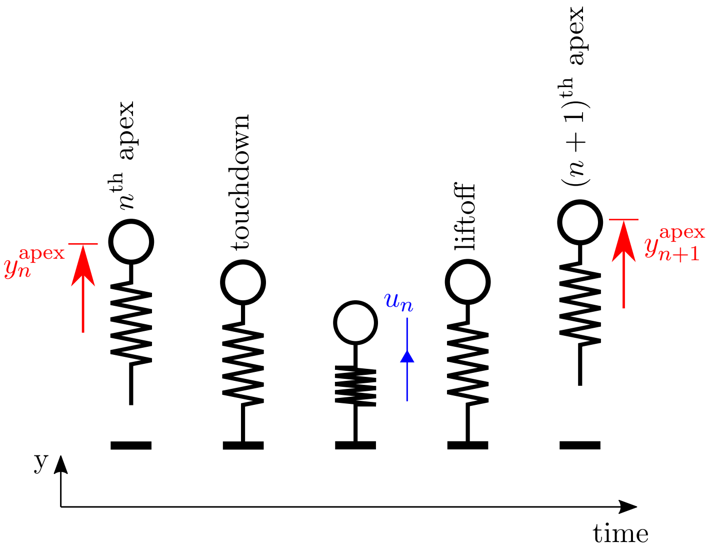


Figure 3-1: The problem formulation: given the $n^{\text{th}}$ apex state and a desired $(n+1)^{\text{th}}$ state (red), choose the system input ($u_n$, blue).

- The proposed empirical controller, with its rationale, formulation and self-tuning method, is presented in Section 3.3.
- The controller is applied and tested in simulation and on an experimental hopper in Sections 3.4 and 3.5 respectively
- Section 3.6 provides some concluding remarks

## 3.2 System Design and Simulation

### 3.2.1 Design Requirements

Running and hopping are dynamically very similar, with both characterised by alternating between *stance* phase, where a single foot is in contact with the ground, and a *flight* phase, with no ground contact. The objective for the design of the robot described in this chapter is to provide a minimal system which will emulate this alternating stance and flight such that the apex height of the system must be controlled. To achieve this, only the vertical motion needs to be considered, leading to a single dimensional robot hopper. The key system state is the apex height, which is the highest point of a particular hop and where the vertical velocity is zero. The apex height of a particular flight phase is dictated by the system dynamics, the value of the preceding apex height and the system input, which is an additional leg force input during the intervening stance phase. The overall system phases and states are summarised in Fig 3-1.

During stance, the robot will have the opportunity to apply a force to the ground in order to control the next apex height; theoretically, this force could be varied arbitrary over the duration of the stance phase, to give a huge choice of possible inputs. However, as the system state is one-dimensional, it should be possible to find a control scheme where the system input is constrained to also be described by a single value. In any case, it is also likely to be impractical to implement complex force modulation over the short duration of a stance phase. The physical parameter to which the input corresponds will vary depending on the design of the robot in question, but must relate in some way to the force or energy input.

Physical space limitations of the laboratory necessitate that the physical robot is smaller than human-scale machines that may be deployed in the future, but this does not prevent the controller demonstration. It does, however, shorten the time of each stance-flight cycle, which adds to the timing precision needed. In particular, the energy input must be done rapidly because of the limited timeframe of the stance phase, and in such a way that it can be accurately controlled.

### 3.2.2 Overall System Design

In order to replicate the vertical motion only of a hopping robot, the system will consist of a single leg constrained to move vertically. The overall system design is shown conceptually in Fig. 3-2. A pneumatic leg actuation system has been chosen where compressed air is injected into the actuator by the timed opening of a valve. The advantage of this is that energy, stored in the compressed air supply, can be injected into the system very quickly, giving a near instantaneous change in energy level to act as the system input. The pneumatic system also has the benefit that the compression of air in the actuator acts as a spring because when the robot hits the ground its momentum compresses the air causing the actuator pressure to rise. This stores and reuses some of the energy from one hop to the next, and a spring-like leg is a key feature of running in biology and for robotics [20].

Vertical motion is achieved my constraining the robot body on the end of a boom, much longer than the hop height, which is pivoted at the other end. As shown in Fig. 3-3, this results in approximately one-dimensional motion, where horizontal movement and rotation can be neglected.

To allow for extended experiments without the need to change batteries, and to simplify the physical design, power for the pneumatics will be provided by a fixed pressure external supply. Between the downward (compression) and upward (extension) parts of the stance phase, an inlet valve is briefly opened between the supply and the actuator to cause the pressure in the actuator rise, adding energy. The duration of this opening is used as the control input to the system – to add more energy, the value remains open for a longer period of time before being closing. Once the actuator has extended and the foot leaves the ground, a second, outlet, valve is opened for long enough to return the actuator pressure to atmospheric, so the system is reset in preparation for the next landing. This pneumatic circuit is shown schematically in Fig. 3-4.

The control task to be considered is to determine the required inlet valve opening time

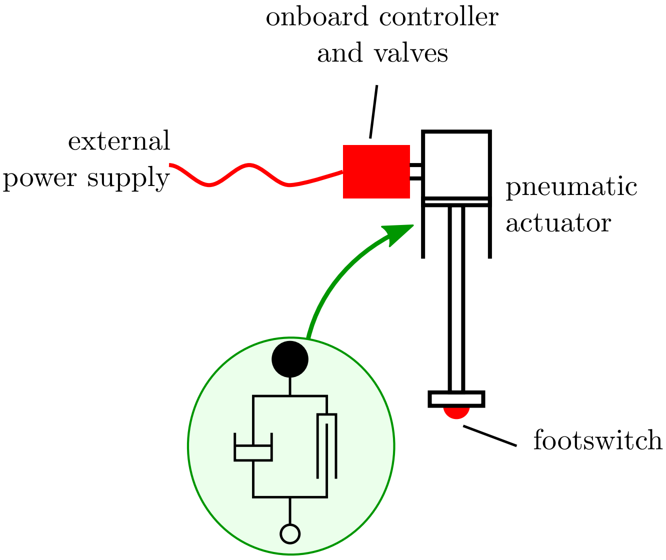

onboard controller
and valves
external
power supply
pneumatic
actuator
footswitch

Figure 3-2: Concept sketch for the one-dimensional system. The pneumatic actuator encapsulates a springy-leg hopping behaviour.

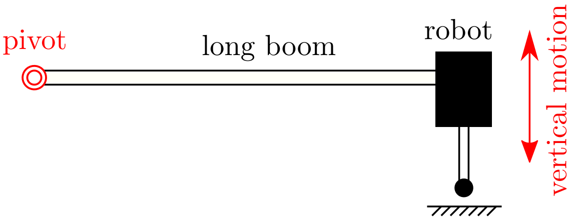

pivot
long boom
robot
vertical motion

Figure 3-3: Illustration of long boom constraining the robot to an approximately vertical motion.

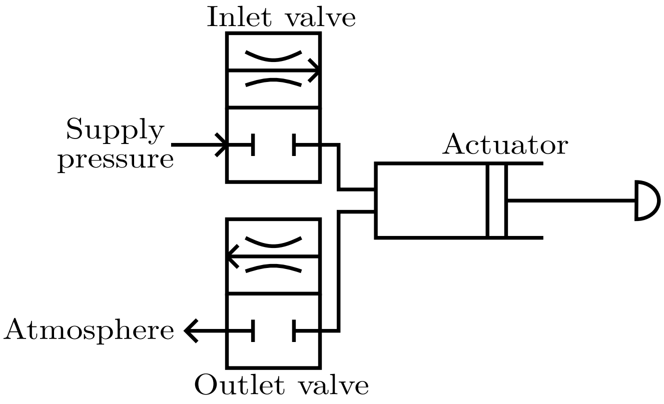


Figure 3-4: Schematic of the pneumatic circuit. Energy is added by briefly opening the inlet valve to allow air from the fixed pressure supply to enter the actuator. The system is reset during flight by opening the outlet valve and venting any remaining pressure in the actuator to atmosphere.

(or in general, the system input $u_n$) to take the system from a particular apex height $y_n^{\text{apex}}$ to the desired next apex height $y_{n+1}^{\text{target}}$. The controller must therefore take the form:

$$u_n = f\left(y_n^{apex}, y_{n+1}^{target}\right) \tag{3.1}$$

### 3.2.3 Simulation Model

Before application on the physical robot, the controller will be tested in simulation. This not only has the advantage of ease of testing without risk of physical damage, but also allows a much wider range of conditions and parameters to be tested. The simulation is implemented in MATLAB using the built-in numerical solver *ode45*.

The model of the dynamic system consists of a mass, damper and actuator as shown in Fig 3-5. Height is measured from the point where the foot is in contact with the ground and the leg is in the fully extended position, such that compression of the leg is a negative displacement, and positive displacements represent the height of the foot from the ground during the flight phase.

In flight, gravity, $-mg$, is the only force being applied, and the motion is simply described by:

$$\ddot{y} = -g \tag{3.2}$$

During stance, the point mass of the body experiences the same force due to gravity and a force exerted by the pneumatic actuator and frictional effects. The instantaneous actuator force is computed using a simple pneumatic model; this means the instantaneous state is described by two variables, the body displacement $y$ and the pressure inside the actuator, $P$, which are linked through the pneumatic model. The volume inside the actuator is approximated from the known geometry, in particular the cross

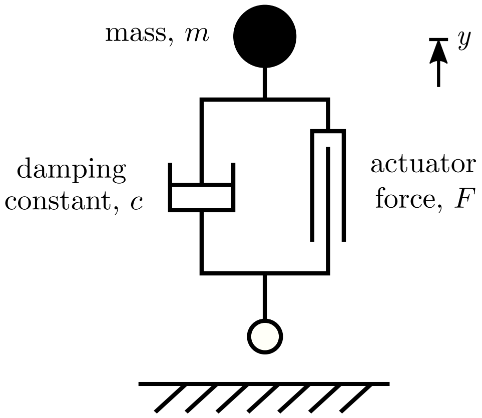


Figure 3-5: The model of the hopper used in simulation. The actuator force is computed by a pneumatic model which provides a spring-like effect as the air is compressed.

sectional area of $A_c$ and an approximated value for the dead volume, including the pipe length between the valve and actuator of $V_0$. The volume is reduced linearly by leg compression, so (noting that $y$ will be negative during stance):

$$V = V_0 + A_c y \tag{3.3}$$

The pressure of the gas inside the actuator is modelled using the equation for adiabatic compression/expansion of an ideal gas, which assumes the process is fast enough to neglect heat exchange:

$$PV^{\gamma} = constant \tag{3.4}$$

where $p$ and $V$ are the instantaneous pressure and volume of the actuator chamber and $\gamma$ is the ratio of specific heats for air, taken as 1.4. Differentiating with respect to time and rearranging yields:

$$\dot{P} = -\frac{\gamma P}{V}\dot{V} \tag{3.5}$$

For an actuator with a cross sectional area of $A_c$, the rate of change of the actuator volume is related to the velocity as $\dot{V} = A_c\dot{y}$, as so the rate of change of pressure due to the motion of the leg is:

$$\dot{P} = -\frac{\gamma P}{V}A_c\dot{y} \tag{3.6}$$

To keep the controller equivalent to that for the physical system, the control input, $u$, is the time, in milliseconds, for which the inlet valve is opened. During this time, an additional term is added to equation (3.6), to account for the extra pressure created by this input. This flow entering the chamber is modelled as the nominal flow stated by the valve manufacturer, scaled proportionally to the pressure difference across the valve. The theoretical volumetric flow, $Q$, through the valve is modelled as:

$$Q = C\left(P_{supply} - P\right) \tag{3.7}$$

where the $C$ value is given on the manufacturer's datasheet. The proposed valve (selected below in Section 3.5.1) has a stated value of $C = 0.2\,\mathrm{l\,s^{-1}\,bar^{-1}}$. This Volumetric flow is converted to a pressure rise by replacing $\dot{V}$ by $Q$ in equation (3.5). It is assumed the supply pressure, $P_{supply}$, is constant at 6 bar, but the flow will decrease as the pressure in the actuator rises.

Losses are introduced in two ways. The first is a viscous damping term in the dynamics, proportional to the vertical velocity, active only during stance, to represent frictional forces. The second consists of removing a fixed percentage of the energy on touchdown (by reducing the velocity), which is somewhat analogous to a coefficient of restitution. These impact losses are likely to occur due to the unmodelled mass of the foot, for which the associated kinetic energy will be lost, and the stiction in the pneumatic cylinder, which will take some energy to overcome. The coefficients for both of these losses have been manually tuned to give similar levels of loss to those of the physical system.

## 3.3 Empirical Controller Derivation

### 3.3.1 Controller Formulation

This section will describe the rationale behind the formulation of a simple controller, and describe the method used to adaptively tune the parameters quickly and accurately, providing a novel method for the control of a hopping machine. Although this chapter is focused on the control of the robot described in Section 3.2.3 as a practical example, the general approach may be applicable to many legged machines, and comprised the stages:

1. Formulate simple (linear or linearised) relationship between steady state output ($y_{ss}^{\text{apex}}$) and input ($u_{ss}$)
2. Formulate linear or linearised function that describes the effect of deviating from $u_{ss}$ by $\Delta u$
3. Collect the parameters from these equations into a small number of constants/gains
4. Update the gain values on the assumption that the values which would have correctly predicted the outcomes of the actual inputs for the previous $p$ hops/steps will also be suitable for future hops/steps

Inherent in the method taken here the assumption that the required input, $u$, to take the system from one apex to the desired next state can be decomposed into:

$$u = u_{ss} + \Delta u \tag{3.8}$$

The first part, $u_{ss}$, describes the input required to overcome losses and maintain a steady state, such that for the n$^{\text{th}}$ hop $y_{n+1}^{\text{apex}} = y_n^{\text{apex}} = y_{ss}^{\text{apex}}$. A simple estimate is to assume losses are proportional to height and energy input is proportional to $u$, so that:

$$u_{ss} = \alpha_1 \, y_{ss}^{\text{apex}} \tag{3.9}$$

The second term, $\Delta u$, describes how a deviation from the steady state input affects the height of the next apex. Let the change in apex state be $\Delta y = y_{n+1}^{\text{apex}} - y_n^{\text{apex}}$ and again assuming proportionality between $u$ and energy supplied:

$$\Delta u = \alpha_2 \, \Delta y \tag{3.10}$$

At the n$^{\text{th}}$ apex, values up to $y_n^{\text{apex}}$ are known[1], and $u_n$ should be chosen to achieve $y_{n+1}^{\text{apex}} = y_{n+1}^{\text{target}}$ at the next apex. Making the substitution that $y_{ss}^{\text{apex}} = y_n^{\text{apex}}$ in (3.9) and combining (3.8)–(3.10), the input to be applied is calculated as:

$$u_n = \alpha_1 y_n + \alpha_2 \left( y_{n+1}^{\text{target}} - y_n \right) \tag{3.11}$$

which provides the controller formula.

### 3.3.2 Tuning Parameters On-line

#### Multiple Linear Regression

When at the n$^{\text{th}}$ apex, all values of $y$ up to $y_n^{\text{apex}}$ are known; $u_n$ should now be selected in order to achieve $y_{n+1}^{\text{target}}$. In order to apply Eq. (3.11), values of $\alpha_1$ and $\alpha_2$ are needed. These could be found by simulation or manual tuning, but here a method will be presented to compute them by considering previous inputs and the resulting apex states. Consider applying (3.11) to the preceding $p$ hops and collecting these equations into a matrix:

$$\hat{\boldsymbol{u}} = \begin{bmatrix} \hat{u}_{n-p} \\ \vdots \\ \hat{u}_{n-1} \end{bmatrix} = \underbrace{\begin{bmatrix} y_{n-p} & (y_{n-p+1} - y_{n-p}) \\ \vdots & \vdots \\ y_{n-1} & (y_n - y_{n-1}) \end{bmatrix}}_{\boldsymbol{Y}} \underbrace{\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix}}_{\boldsymbol{A}} \tag{3.12}$$

Here, $\boldsymbol{Y}$ contains the observed apex heights over the previous $p$ hops, and $\hat{\boldsymbol{u}}$ is the

---

[1] In practice there may be a delay in computing $y_n^{\text{apex}}$, but in any case it should be known before the $u_n$ is required during the stance phase.

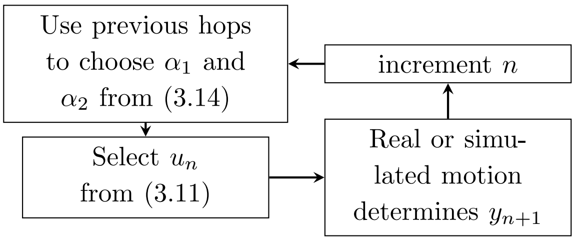

Use previous hops to choose $\alpha_1$ and $\alpha_2$ from (3.14)
increment $n$
Select $u_n$ from (3.11)
Real or simulated motion determines $y_{n+1}$

Figure 3-6: Method for controller update at each apex.

inputs that would have been chosen using a particular set of $\alpha$ values. If it is assumed that the values which would have accurately predicted this previous behaviour will be good values for future use in the controller, then $\boldsymbol{A}$ should be chosen to minimise the error between the predicted ($\hat{\boldsymbol{u}}$) and actual ($\boldsymbol{u}$) input values. This error is defined as $\boldsymbol{\epsilon}$:

$$\boldsymbol{\epsilon} = \boldsymbol{u} - \hat{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{Y}\boldsymbol{A} \tag{3.13}$$

Note all of the values inside $\boldsymbol{Y}$ and $\boldsymbol{u}$ are known at the n$^{\text{th}}$ apex. The values of $\alpha_1$ and $\alpha_2$ to minimise the sum of the squares of the errors, $\boldsymbol{\epsilon}'\boldsymbol{\epsilon}$, are given by the well known multiple linear regression solution:

$$\boldsymbol{A} = \left(\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{Y}\right)^{-1}\boldsymbol{Y}^{\mathsf{T}}\boldsymbol{u} \tag{3.14}$$

At least two previous hops are required in order to provide enough information to set the $\alpha$ values, so $p \leq 2$. In this special case $p = 2$, $\boldsymbol{Y}$ is square and so (3.14) reduces to:

$$\boldsymbol{A} = \boldsymbol{Y}^{-1}\boldsymbol{u} \tag{3.15}$$

The controller sequence is illustrated in Fig. 3-6. Clearly, this strategy for selecting $\alpha$ values cannot work for the first few hops, before $p$ previous results are available. As a simple way to overcome this, some fixed initial values are used, determined from previous simulations or experiments as described below, and will also serve as the values for a fixed gain controller for comparison to the adaptive controller.

### Avoiding Matrix Inversion Singularities

For steady state gaits, the change in height between hops, $\Delta y$, will be close to zero. Intuitively, this will provide little information for selecting $\alpha_2$, which describes the input needed to make such changes. This would be an example of the matrix to be inverted in (3.14) being close to singular, and the resulting $\alpha_2$ value would be very susceptible to any small errors. A simple strategy to avoid this will be employed, whereby $\boldsymbol{Y}$ must contain at least one $\Delta y$ of a minimum magnitude, otherwise $\alpha_2$ is not updated but left

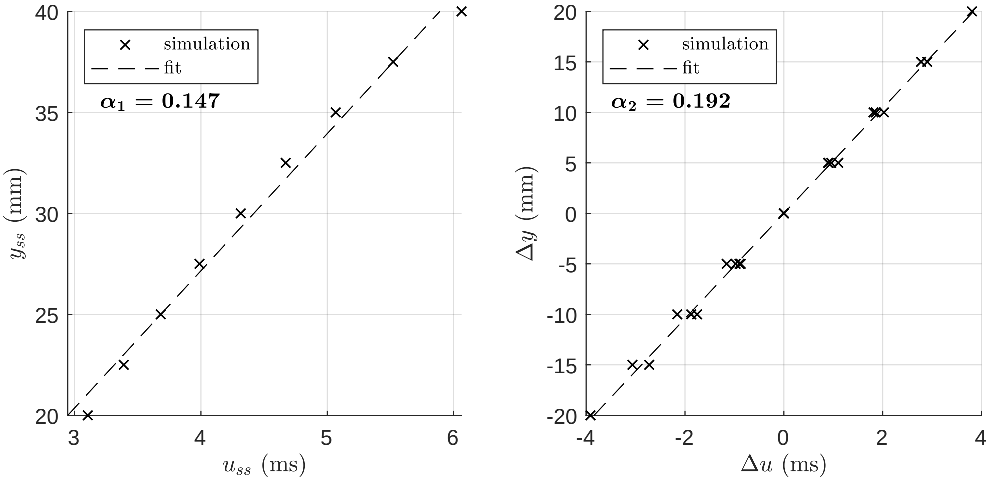


Figure 3-7: Using simulations to estimate $\alpha_1$ (left) and $\alpha_2$ (right)

at its previous value. This is considered acceptable as for a steady state gait, the term including $\alpha_2$ in (3.11) is small anyway.

In the simulation environment, during long runs with a repetitive demand (either steady state or alternating between two values), the system will settle to precisely the same values on each step without the small variations caused by real sensor and actuator inconsistencies. This has been found to lead to errors in the matrix inversion even with the $\Delta y$ check described above, believed to be due to the limits of numerical precision. Therefore, an additional check skips the gain updates altogether when the matrix condition number is very large[2]. This is not expected to be necessary for physical implementation.

## 3.4 Simulation Results

The numerical simulation described in Section 3.2.3 has been used to test the controller with three different settings. Firstly, with fixed, pre-set gains ($\alpha_1$ and $\alpha_2$) which remain unchanged. Secondly, by starting with these gains but then allowing the update algorithm to modify them, with values of $p = 2, 3, 5, 10$. The minimum $\Delta y$ value to update $\alpha_2$ has been set at 3 mm, approximately 10% of a typical hop height.

Before the controller can be applied, the fixed values for the gains are required. Repeated simulation of the apex-to-apex dynamics has been used to find the steady state input to achieve a range of steady state heights, as well as the input required to move between of these heights in a single hop. These have been used to create the linear fits shown in Fig. 3-7 to estimate $\alpha_1$ and $\alpha_2$.

---

[2] Implemented using the MATLAB function *rcond*, see `mathworks.com/help/matlab/ref/rcond.html`

For the tests a set sequence of demand heights was passed to each controller in turn. An example set of demand values has been used to demonstrate the controller's applicability to a range of demands. This begins with a steady state demand near the system's maximum height, then near the minimum, before an alternating demand between the high and low values and a series of random demands illustrate agile changes.

The results using the controller with fixed gains and for $p = 2, 3, 5, 10$ against this set of demands, on "hard" ground representing a firm floor, are shown in Fig. 3-8. It can be seen that while the fixed gains controller suffers a steady state error, the adaptive controllers are able to remove this error by tuning $\alpha_1$, as shown by the updating gain values in the lower plot. The performance is broadly similar over the agile demands. The controller adapts more quickly with low $p$ values, but is prone to occasionally miscalculating the required gain values, *e.g.* at hop 32 for $p = 2$, leading to a larger error in apex height on next step. A higher $p$ value uses more previous steps to reduce such errors and will improve stability for practical applications. This is seen as a slower but less erratic updating of the gains for $p = 5$ and even more so at $p = 10$.

The accuracy of the controllers is shown in more detail in Fig. 3-9, with each of the four types of demand in the illustrative example above considered in turn: a steady state demand of a high and a low hop, alternating on every step between two values, and demand varying randomly between steps. Controllers with fixed gains and values of $p = 2, 3, 5, 10$ are included. For each of the controllers, the absolute error (magnitude of the difference between demanded and actual apex height) has been measured with 100 hops in each run. As the random demand will be different on every run, 100 repeats were conducted in this case, to give an average taken over a total of $10,000$ hops. However, for $p = 2$, the robot failed (becoming "stuck" on the ground after applying too small an input) on 33 of the runs, giving fewer hops overall and clearly an ineffective controller.

For all of the regular demand profiles (constant or alternating values) the adaptive controller finds the appropriate gain values and the error is very small irrespective of $p$ value, while the fixed gains version suffers steady state errors. However, for the random demand the adaptive controller is less accurate when the $p$ value is small. In these cases, the adaptive controller is using only a small sample of hops, which may be different in height or required height change from the next hop, and so not allow for good gain values to be chosen for the upcoming control input. This problem is overcome by larger $p$ values, which by using more previous hops can, on average, select better gain values, with values over $p = 5$ outperforming the fixed gains.

One significant advantage of using adaptive gains is to automatically correct for changes in the system. For example, as a robot moves it may encounter different ground surfaces with varying stiffnesses and energy dissipation. In the next test, the robot experiences a sudden increase in the energy loss on touchdown between one step and the next. The controller is given no warning or indication this has happened, except for the resulting changes in hop height. The result is shown in Fig 3-10, where, as a simple example of an agile gait, the demand height alternates between high and low. The point of change

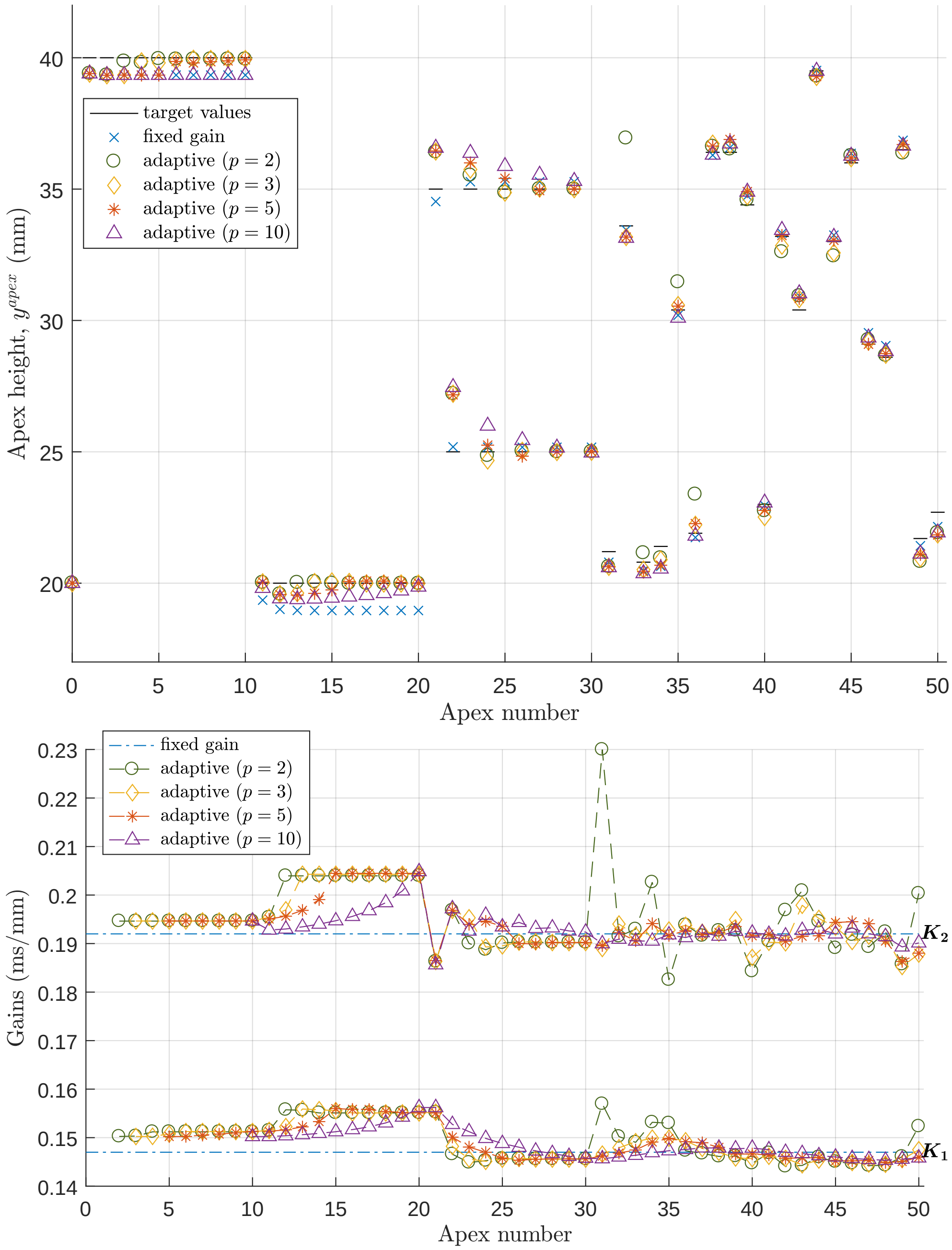


Figure 3-8: Performance of each controller in simulation for a range of demand profile. Reasonable performance is achieved for agile demands, and the adaptive controller is able to eliminate the steady state error. Top: target apex heights and those achieved. Bottom: gain values as they are updated.

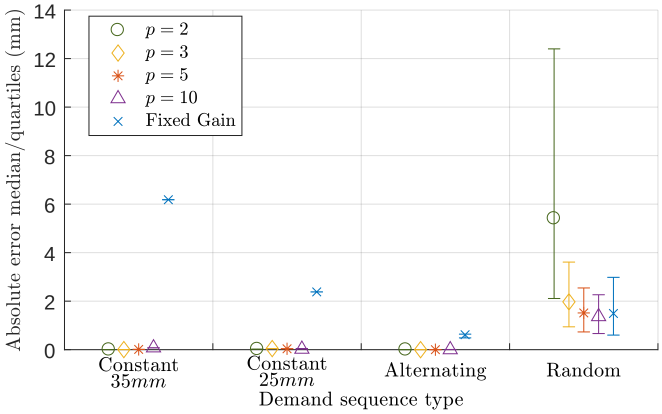


Figure 3-9: Errors suffered by the controller with fixed gains and adapting by considering the preceding $p$ hops, for steady state demands, demands alternating on each step and for a random sequence of demand values.

it denoted by the vertical dashed line. It takes a little more than $p$ hops to adjust the gains and correct for the error; the fixed gain controller does not correct and suffers a steady state error as it is poorly tuned for the new ground properties.

## 3.5 Physical Experiments

### 3.5.1 Transition to Physical Implementation

This section presents results of applying the controller from Section 3.3 to a physical test system, shown in Fig 3-11. The previous simulation results have been used to inform and give greater confidence to the physical robot design in several ways.

The simulations have shown hop heights in the region of 40 mm are possible, which should be suitable for the space available, which allows for a boom length of approximately 1 m. This hop height is also large enough to be accurately tracked by the external tracking cameras (accuracy <1 mm) and on-board timing, with an expected flight time of approximately 90 ms. It should be noted the precise level and nature of frictional losses is difficult to estimate in advance, and notoriously difficult to model. This is very likely to affect the size of actuator required. However, in advance of building the physical system, various frictional values were experimented with in the simulation, including highly conservative values, and found to give acceptable performance by increasing the valve actuation times.

An important aspect of the chosen actuation system is the inlet valve, and in particular its ability to open and close rapidly. The valve actuation time must be significantly shorter than the total valve opening time in order that it can be precisely controlled. In the simulations, the valve opening time computed by the various controller parameters

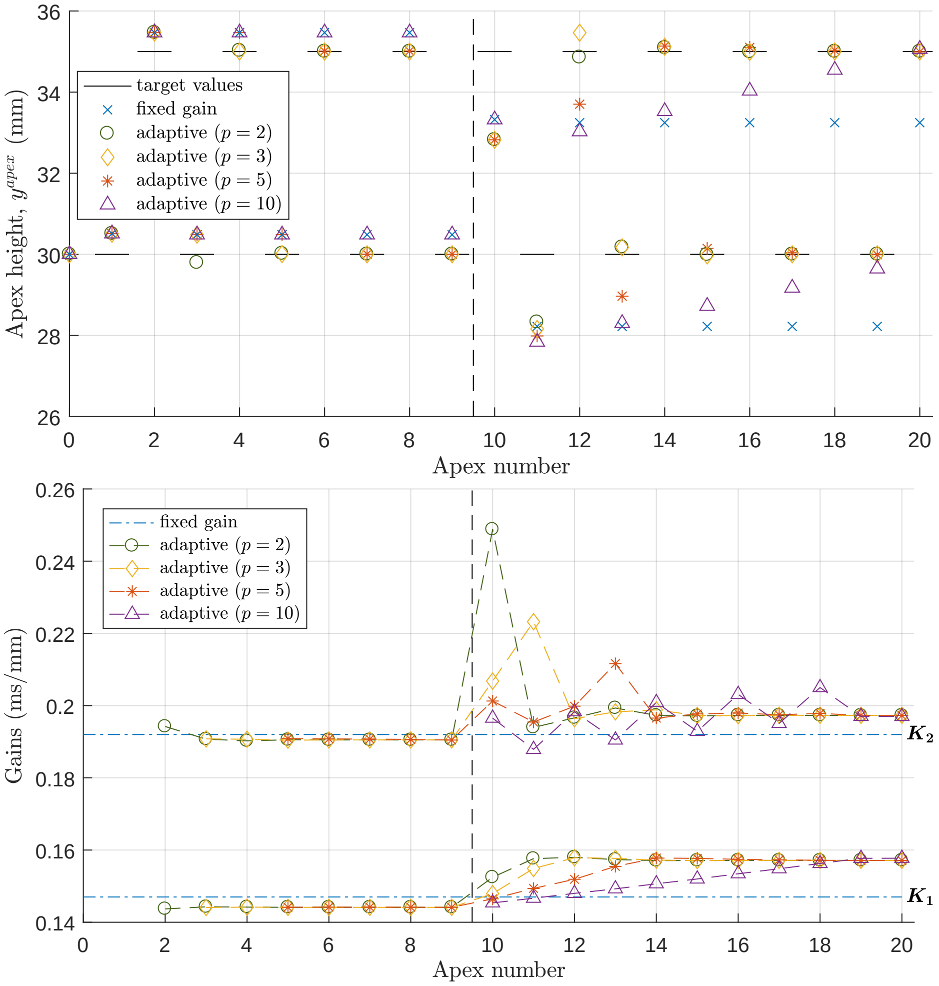


Figure 3-10: Effect of a sudden change in ground surface in simulation. After the change, the fixed gain controller is poorly tuned as suffers significant error, which the adaptive controllers are able to remove. Top: target apex heights and those achieved. Bottom: gain values as they are updated.

varied between 2.1 ms to 6.8 ms, so the selected valve must open and close in less time than this.

Taking these factors into consideration, various parameters of the simulation were adjusted to select the valve and actuator as:

- Festo Fast switching valve MHJ9-QS-4-LF with quoted closing and opening times of 0.7 ms and 0.9 ms respectively[3]
- Festo DSNU cylinder with 8 mm diameter and 60 mm stroke[4]

There are, of course, limitations to what can be learn from the simulations with regard to the effects of noise introduced by measurement error and other variability, which is likely to impair the controller and adaptive scheme. This is expected to mean that higher $p$ values become necessary in physical experiments.

Overall, the successful simulations give confidence that the controller structure is a sensible one, and that a relatively small number of hops will be sufficient to tune the adaptive controller. Testing on a simple system physical system is the next logical step, to verify the controller can be transferred for more practical implementation.

### 3.5.2 Physical Robot

The experimental setup, shown in Fig 3-11, consists of a single pneumatic leg rigidly attached to, and pointing vertically down from, the body of the robot. The robot is mounted on the end of a boom which is pivoted at the other end in the vertical plane; as the boom is much longer than the leg or the hop heights, the movement is constrained to be almost vertical. This also provides a long lever arm, so any friction in the bearing at the base of the boom has a minimal impact on the dynamics of the robot at the end of the boom. The boom is made from a carbon fibre tube so it is both lightweight and stiff to minimise its inertia while keeping the motion constrained. For simplicity of mechanical design, power (in the form of electrical connection and compressed air) are supplied externally through the boom. All control computation is carried out on-board by an Arduino Nano (16 MHz ATmega328, 2 kB RAM); this very low power device was chosen to highlight the low computational load of the algorithm. The robot has a mass of 506 g and a leg length of 150 mm.

For this one-dimensional system, the only sensing that is required is the height of each hop. A simple footswitch is used for this purpose, which detects the transitions between flight and stance. This provides a measure of flight time, allowing calculation of hop height at the moment of touchdown. Neglecting any friction during flight and assuming a purely vertical motion (*i.e.* that the boom is long compared to hop height) then the height, $y^{\text{apex}}$ for a given flight time, $t^{\text{flight}}$, and acceleration due to gravity, $g$,

---

[3] Details available from `www.festo.com/cat/en-gb_gb/products_MHJ?CurrentPartNo=572079`

[4] Details available from `www.festo.com/cat/en-gb_gb/products_DSNU_1?CurrentPartNo=1908250`

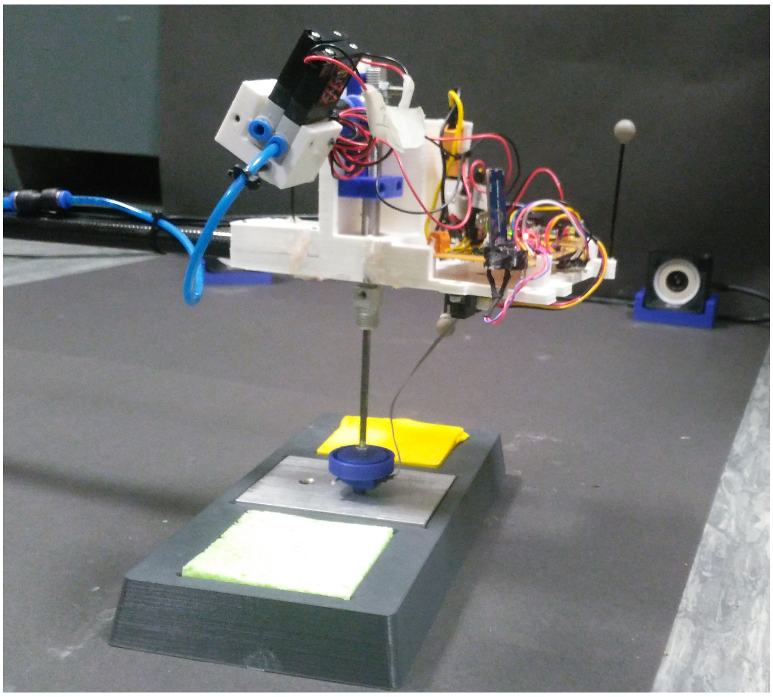


Figure 3-11: The robot used for experimental validation, constrained to hop vertically. The boom, carrying external power supply, can be seen coming from the left. The camera in the background forms part of the vision system, which is used for verification only with all control is on-board based on footswitch timing.

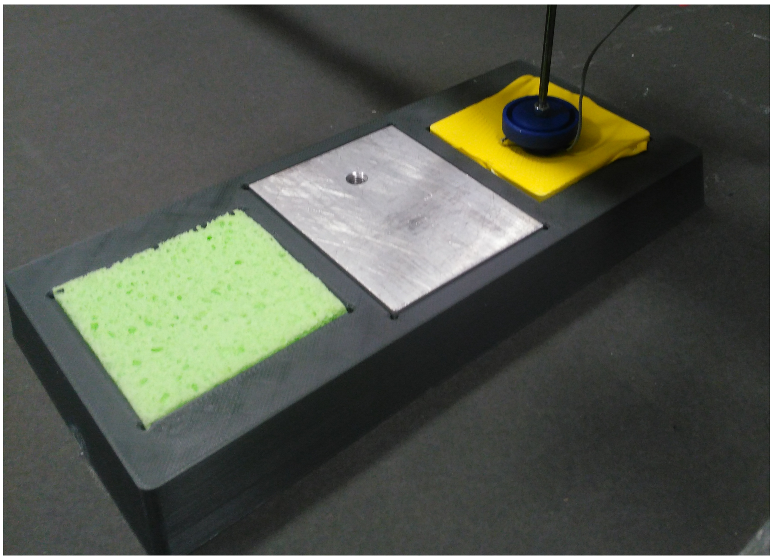


Figure 3-12: The three ground surfaces in their tray, left to right: "soft", "hard" and "springy".

is computed as:

$$y^{\text{apex}} = \frac{1}{2} g \left( \frac{t^{\text{flight}}}{2} \right)^2 \tag{3.16}$$

For the two valves controlling air entering and leaving the actuator, fast acting solenoid values (Festo MHJ9 with switching time <1 ms), have been used, arranged as described above and shown in Fig. 3-4. The supply air is provided by an external compressor with a reservoir much larger than the actuator to allow many hops to be completed with no significant change in the supply pressure, set to 6 bar

Ideally, the input valve opening should occur at the bottom of the stance phase, so as to provide the additional pressure for the entirety of the extension but none of the compression part of the stance. However, it is difficult in practice to detect this point precisely in real time. If the stance forces can be assumed to be dominated by the spring stiffness, it is reasonable to assume the frequency of the oscillation will be constant, and the valve opening can be executed after a fixed delay from the touchdown, an event which is easy to detect via a footswitch. In practice, this delay has been set experimentally to 50 ms by manually determining the value with gives the highest steady-state apex height.

A standard pneumatic cylinder (Festo, DSNU-8) has been used for simplicity, but the friction is significant and in particular the seals cause a stiction effect. This will be a significant source of inefficiency and future applications should implement a lower-friction alternative to reduce losses. However the current setup is sufficient to demonstrate the controller.

In order to test the controller's ability to operate on various ground surfaces, and demonstrate its adaptive nature, three distinct surfaces are used. These surfaces can be seen in Fig 3-12, and will be referred to as "soft", "hard" and "springy" ground. The soft ground is a spongy cellulose material expected to absorb more energy on impact, the hard ground is an aluminium plate and the springy surface has been made by stretching a latex rubber membrane over a frame (like a drum). The three test surfaces are contained in a tray to ensure the tops of the surfaces are at the same level and allow for quick manual changes between surfaces.

A simple test was performed by pressurising the actuator, dropping the robot onto each surface with no actuator input and measuring the height of the rebound using motion tracking cameras. The coefficient of restitution can be estimated as the ratio of apex height from one hop to the next, and was found to be approximately 0.30, 0.22 and 0.33 for the hard, soft and springy surfaces respectively.

### 3.5.3 Experiments and Results

As with the simulation, the first task is to find a set of gains which will be used in the "fixed gain" controller, and as initial gains for the adaptive controller. This was

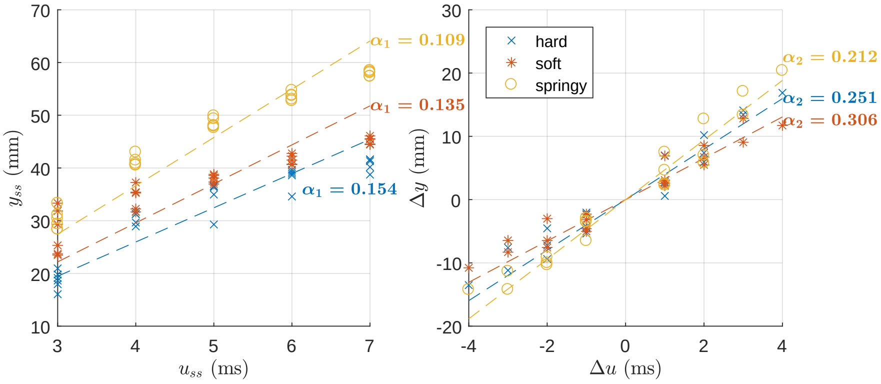


Figure 3-13: Estimates for $\alpha_1$ and $\alpha_2$ have found by fitting to a test sequence of valve opening times ($u$). These values are to be used as initial values and for the fixed gain controller.

done by cycling through a set of open loop valve opening times, allowing the system to settle on a steady state for each. The resulting plots of steady state values, $u_{ss}$, and the effect of a step change starting at steady state conditions, $\Delta u$, are shown in Fig. 3-13 for each of the ground surfaces. It can be seen each surface has distinct characteristics, requiring different gain values.

As for the simulation, a set sequence of demands is used that includes sections of steady state, alternating high and low and random variation in demand height. The time-series for a single run is shown in Fig 3-14, together with the data from motion capture cameras for verification. The robot's internal estimates of apex height (blue crosses) match well the tracked heights.

For each ground surface, the controller has been applied using fixed gains (from Fig. 3-13) and with the adaption implemented with two $p$ values, defining the number of previous hops considered: $p = 2$ (the minimum possible) and $p = 5$. The results of each test are shown in Fig 3-15, with a separate plot for each surface. Each surface shows a similar result, with the fixed gain controller suffering steady state error which are removed by the adaptive scheme. The lower $p$ value should remove such errors more quickly, but also tends to be more erratic, because it uses only a small sample of previous data – in effect, it has a shorter "memory".

The adaptive scheme should be able to overcome changes in the system after a few hops. To test this, the ground surface tray was manually moved during the flight phase between one hop and the next, to mimic the effect of a robot stepping from one surface onto another. The controller is provided with no prior warning this will happen, and must adapt based only on the changes in the relationship between the input and resulting apex heights. Figure 3-16 shows the results of changing from hard to soft (top) and from hard to springy (bottom). After the change, the fixed gain controller suffers an error, and does not correct for it. In contrast, the adaptive scheme is able to update the relevant gains and returns to the demanded heights.

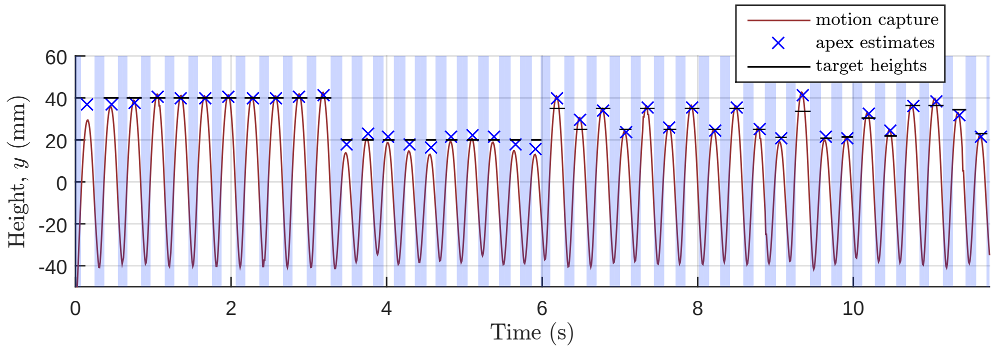


Figure 3-14: Height plotted against time with the motion capture data for verification. Shaded regions are where footswitch is depressed (*i.e.* stance phase).

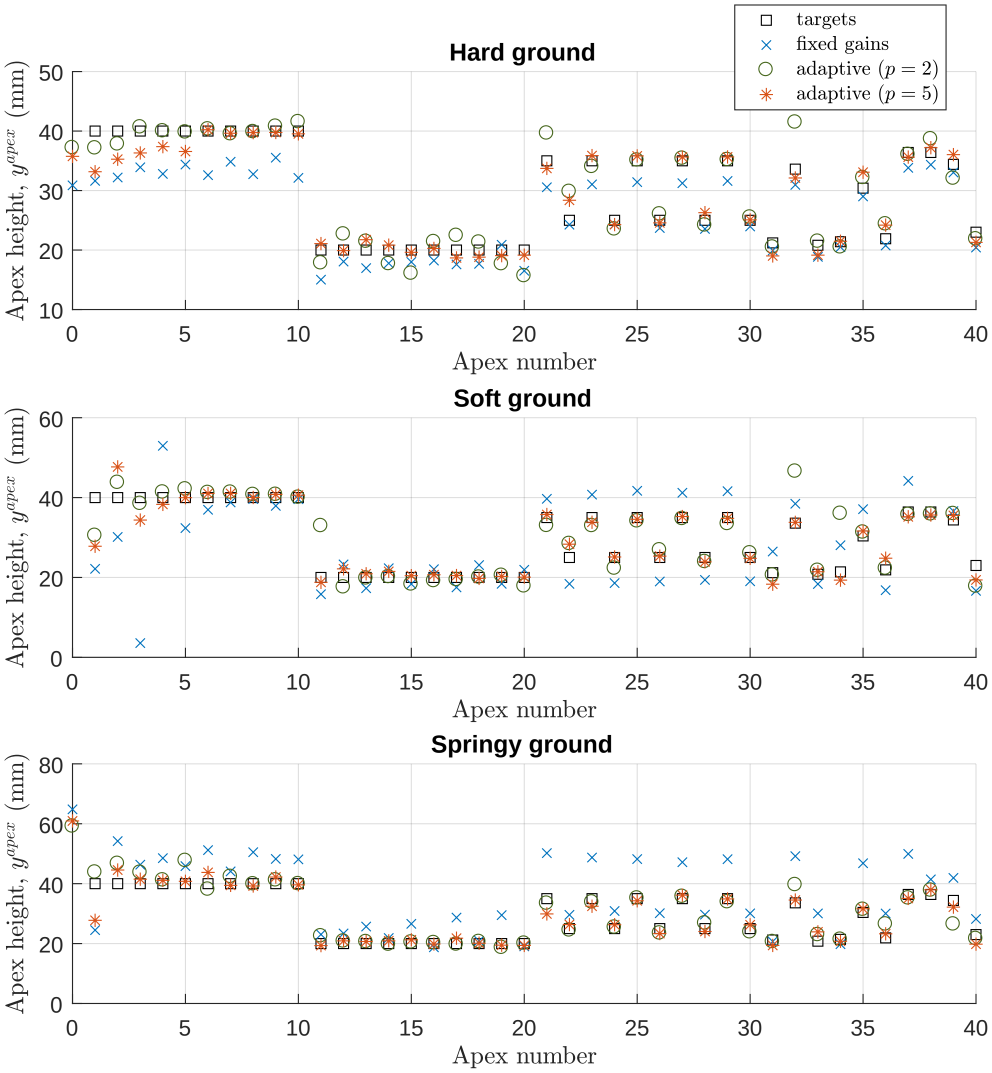


Figure 3-15: Experimental results over different ground types

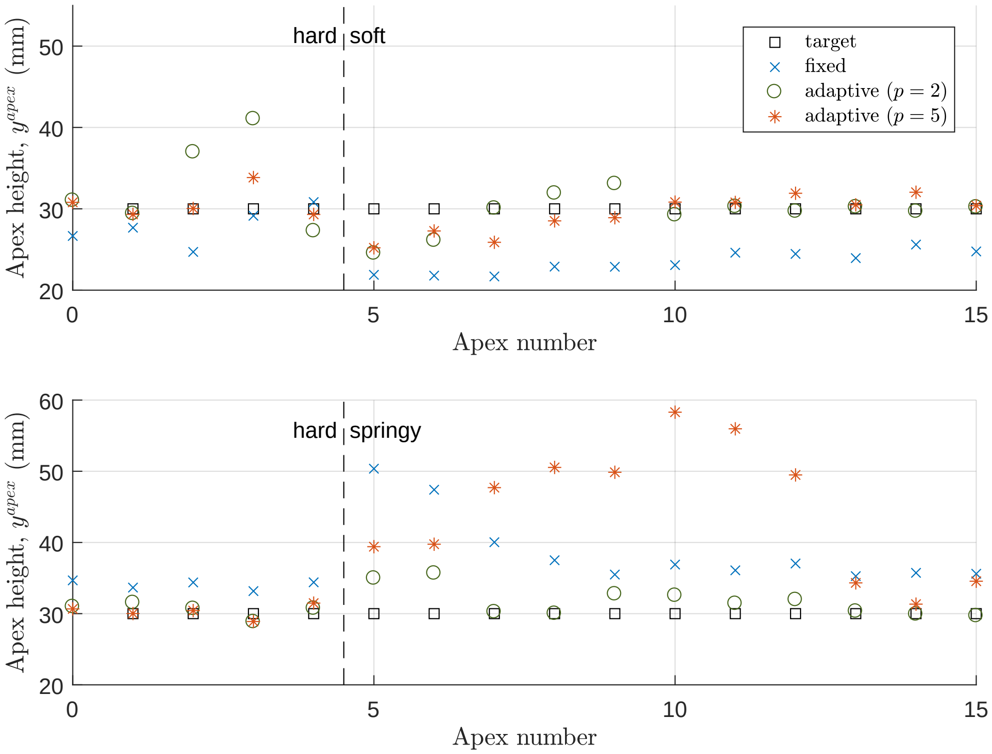


Figure 3-16: Experimental results with changing ground surface at dashed line

## 3.6 Concluding Remarks

This chapter has presented an empirical controller in the context of height control for a one dimensional hopper, with on-line self tuning of the controller parameters based on previous hops. It has been applied to a simulation of a pneumatically actuated robot leg, and its physical incarnation constrained to vertical motion. This computationally simple controller has been shown to be able to dynamically control agile changes in height and provides several benefits:

- fast to compute and implement on low power hardware
- directly computes the system input (here valve opening), eliminating need for lower level controller
- elimination of steady state errors for constant demand heights
- takes into account additional energy needed to change height so can perform agile sequences
- adapts to changes in system or environment

One critical design choice is the selection of the parameter $p$, which selects how many previous steps to consider. A small $p$ creates a controller with a short "memory", which will adapt to changes more quickly, but may change the gains too readily in response to noise or erroneous values. Conversely, a large $p$ results in more stable gain values, which will update more slowly. The choice will depend on the application, and will have to be made on a case by case basis. This will depend upon the physical robot itself, for example the prevalence of measurement noise, but also the environment and task in question, such as the frequency and magnitude of changes to the ground surface.

The height controller is intended to form one part of a controller for hopping or running in two or three dimensions. In the forthcoming chapters, a similar controller for forward velocity in two-dimensional hopping is presented, and then these combined to control both height and velocity.

# Chapter 4

# Forward Velocity Control of the Spring Loaded Inverted Pendulum

When hopping or running, it is possible to change the length of a bound even without changing the overall system energy. This is illustrated in Fig. 4-1, where a hopper traverses unevenly spaced stepping stones, with the first requiring a longer step than the second. The apex of the first is lower and faster with more kinetic energy contained in the forward velocity, while the second follows a higher, more looping trajectory with more energy instead transferred into gravitational potential.

The stance between these two bounds is the critical phase, in which the hopper must achieve the appropriate transfer of this energy, as once in the flight phase it will follow a ballistic trajectory and can no longer adjust. Slowing down, as in this example, requires the hopper to "lean back", so that the angle of the leg at the moment of touchdown is further from the vertical than at lift off. Conversely, "leaning forward" would mean the leg starts closer to vertical, so that by the time of liftoff the leg is more inclined and can be thought of as pushing the body forward to increase horizontal velocity.

This chapter seeks to build a controller around this intuition, to control a sequence of varying demand velocities.

## 4.1 Introduction

This chapter considers the task of controlling the forward velocity at each apex for the commonly used "Spring Loaded Inverted Pendulum" (SLIP) model. Much of the simulation work (excluding the adaptive tuning) was presented at the conference Towards Autonomous Robotic Systems (TAROS) [75], and a further publication including the

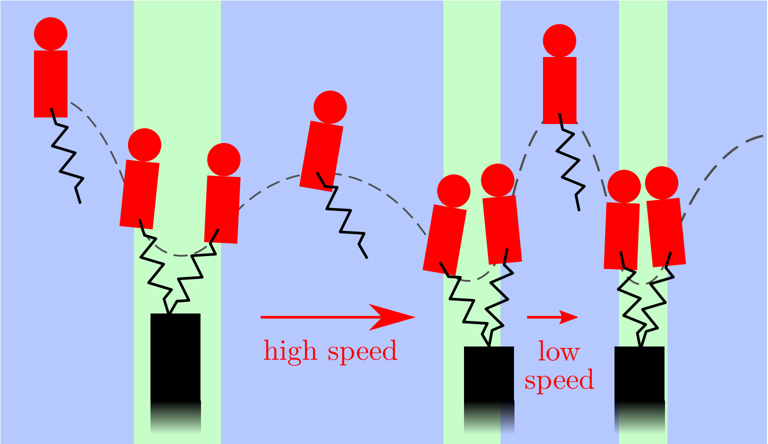


Figure 4-1: Conceptual diagram of hopping between stepping stones. A larger step can be achieved by a higher velocity, so velocity control could be used to cross difficult terrain. The green background indicates the stance phase, while blue is the flight phase, defined by foot contact with the ground.

adaptive approach and experimental validation is under review at the time of writing. Fig. 4-1 shows conceptually the alternating stance and flight phases of the robot's motion. In order to traverse differently spaced footholds (the black platforms), the forward velocity must be accurately changed over a single stance phase – an agile gait.

The robot considered is a two dimensional hopper, moving vertically and in a single horizontal direction, equivalent to the sagittal plane and ignoring the need for the left-to-right balance of a three-dimensional system. Furthermore, the model is energetically conservative (without any frictional or other losses) and so the task of energy regulation can be ignored. Although the apex height will vary from apex to apex, depending on the forward speed such that the total energy is constant, the height is not actively controlled but instead follows as a consequence of achieving the desired velocity, which is the parameter subject to demand values.

As this chapter considers only control of the forward velocity, it makes the inherent assumption that the control developed for this aspect of hopping (or running) can later be applied in conjunction with other controllers (for balance, apex height etc.). This approach has been widely accepted ever since the pioneering “three-part controller” from Raibert[48] made this decoupling assumption.

### 4.1.1 Chapter Outline and Contributions

This chapter will present:

- In sections 4.2 to 4.5, a simulation comparison of existing analytical approximations against a simple, two-term, empirical controller applied to the lossless SLIP model.

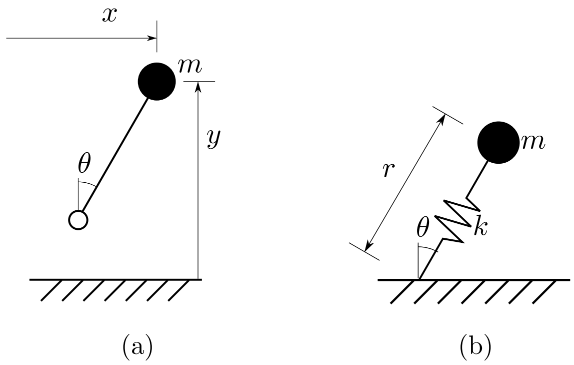


Figure 4-2: System model for (a) flight phase and (b) stance phase.

- In Section 4.6, the used adaptive gains to allow for automatic adjustment of the controller, with simulation results in Section 4.7.
- Finally in Section 4.8, an experimental demonstration on a physical, pneumatically actuated, robot with this empirical controller and adaptive gains operating in real time.

## 4.2 System Model

### 4.2.1 Description of Model

The SLIP model has been very widely used in simulations of dynamic legged locomotion. First proposed by Blickhan [20], it is inspired by the simple observation in biological running that while the foot is in contact with the ground, the leg compresses and then extends like a spring. Thus, the model comprises a point mass (representing the body) and a simple spring (representing the leg). As the mass is concentrated in the body, while the foot is not in contact with the ground, defined as the *flight phase*, the only force present is gravity and the motion is a simple ballistic trajectory. When the foot is in contact with the ground, the *stance phase*, the leg spring creates an additional force, acting along the leg as it pivots about the foot contact point (it is assumed there is no movement of the foot relative to the ground during stance). These two phases are illustrated in Fig. 4-2, with some important variables labelled: $x$ and $y$ are the horizontal and vertical displacement, $\theta$ and $r$ are the leg angle and length, $m$ is the body mass and $k$ is the leg spring constant.

During running or hopping, the system alternates between a stance phase with a single foot in contact with the ground, and a flight phase characterised by ballistic motion; whether the next stance uses the opposite foot (bipedal running) or the same foot (hopping) is of no consequence for this model, and so hopping and running are similar control problems and will be considered equivalent in this analysis. The cyclic nature of the system means it is convenient to consider the apex state, where the body reaches its highest point for a particular flight phase and the vertical velocity is zero. The dynamics of the system are then a mapping of one apex state to the next. The apex state can be defined completely by the horizontal position and velocity and the vertical position of the body (the vertical velocity being defined to be zero) relative to some origin. It is often convenient to omit the horizontal position, as this has no effect on the dynamics and is not relevant if the objective is to control horizontal speed.

Thus, for the two-dimensional form of the SLIP hopper, the apex state will be described by the forward velocity, $\dot{x}^{apex}$ and apex height, $y^{apex}$.

### 4.2.2 The Control Task: Velocity Control

The model is energetically conservative, so not all apex states are reachable for a given system; indeed there is a unique apex height which will result for a given forward velocity and system energy. Therefore, for the problem of velocity control, no apex heights need to be specified but instead are a necessary and natural result of achieving the demanded velocities. The general task considered here is: during a flight phase, given the current ($\mathrm{n}^{\mathrm{th}}$) apex state, select the touchdown angle ($\theta_n^{td}$) at which the natural dynamics of the system will result in the desired value of forward velocity ($\dot{x}_{n+1}^{tgt}$) for the next flight phase, as illustrated in Fig. 4-3.

For a particular task, a sequence of steps must be performed with a set target velocity for the flight phase of each individual step. A steady state demand would consist of the same target velocity on each step. By contrast, the demand will be termed “agile” if there is significant variation such that the target velocity at each flight phase may be different from the previous one, requiring changes in velocity at every stance phase. This is a more difficult task, compared to approaching a steady state demand over multiple steps, but is critical for legged locomotion to traverse complex terrain. This is because it allows the length of each step to be controlled individually, in order to solve tasks such as traversing a series of isolated safe foot landing positions, such as stepping stones.

Therefore accurately computing the required touchdown angles to achieve a given velocity target sequence is required to control foot landing positions, and furthermore to do so quickly could be particularly useful for planning planning the landing positions – for instance finding a sequence that is feasible without exceeding some limits on touchdown angle or velocity might require many possible plans to be evaluated in a short time.

### 4.2.3 System Dynamics

As described above, the SLIP model consists of a point mass body connected to a massless foot by a spring.

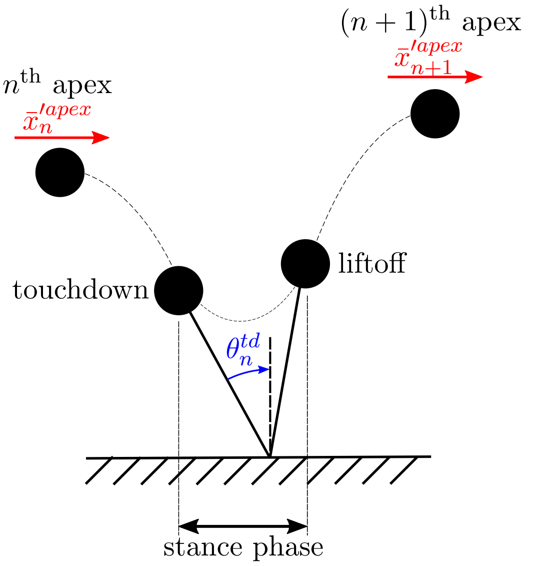


Figure 4-3: The problem being considered: the relationship from the $\mathrm{n}^{\mathrm{th}}$ apex velocity to the $(\mathrm{n}+1)^{\mathrm{th}}$ is a function of the touchdown angle. Note that angles are defined as positive clockwise, so in the example shown $\theta_n^{td}$ is negative, which is typical for positive velocities.

When the foot is not in contact with the ground, during the flight phase, the spring is assumed to take its rest length and the robot acts as a point mass falling under gravity, with zero inertia, and so the foot can be positioned arbitrarily. The angle of the leg is still important, however, to detect when the foot will hit the ground (touchdown). In this case the coordinates are the horizontal and vertical body displacements, $x$ and $y$, and the leg angle, $\theta$, as shown in figure 4-2(a).

During the stance phase, when the foot is assumed to be rigidly attached to the ground, the coordinate system is changed to a rotational system, with $\theta$ the leg angle and $r$ the leg length, as shown in figure 4-2(b).

For this system the equations of motion for the flight phase are:

$$\ddot{x} = 0 \tag{4.1a}$$

$$\ddot{y} = -g \tag{4.1b}$$

and during stance phase:

$$\ddot{r} = \frac{k}{m}(r_0 - r) + r\dot{\theta}^2 - g\cos\theta \tag{4.2a}$$

$$\ddot{\theta} = \frac{g}{r}\sin\theta - \frac{2\dot{r}\dot{\theta}}{r} \tag{4.2b}$$

where $m$ is the body mass, $k$ is the spring stiffness, $g$ is the acceleration due to gravity and $x$, $y$, $r$ and $\theta$ are the dimensions as defined in figure 4-2.

### 4.2.4 Non-dimensional Equations

To allow for easier comparison between systems of different sizes, the equations can be non-dimensionalised [55]. To do this, a reference length (for which $r_0$, the rest length of the leg is used) and an arbitrary time scaling factor, $s$, must be defined; careful choice of $s$ allows the system to be characterised by a single parameter, the "dimensionless stiffness", $k^*$. Other dimensionless quantities will be denoted by the overbar throughout, and are defined as:

$$\begin{aligned}
\bar{x} &= \tfrac{x}{r_0}\ , & \bar{x}' &= \tfrac{s\dot{x}}{r_0}\ , & \bar{x}'' &= \tfrac{s^2\ddot{x}}{r_0} \\
\bar{y} &= \tfrac{y}{r_0}\ , & \bar{y}' &= \tfrac{s\dot{y}}{r_0}\ , & \bar{y}'' &= \tfrac{s^2\ddot{y}}{r_0} \\
\bar{r} &= \tfrac{r}{r_0}\ , & \bar{r}' &= \tfrac{s\dot{r}}{r_0}\ , & \bar{r}'' &= \tfrac{s^2\ddot{r}}{r_0} \\
\bar{\theta} &= \theta\ , & \bar{\theta}' &= s\dot{\theta}\ , & \bar{\theta}'' &= s^2\ddot{\theta}
\end{aligned}$$

Here $()'$ represents differentiation with respect to the non-dimensional time, $\bar{t} = t/s$. The time scale factor is chosen as $s = \sqrt{r_0/g}$. Substituting these coordinates into the equations of motion for the standard SLIP hopper (equations 4.1 and 4.2) during flight phase:

$$\bar{x}'' = 0 \tag{4.3a}$$

$$\bar{y}'' = -1 \tag{4.3b}$$

and during the stance phase:

$$\bar{r}'' = k^*(1 - \bar{r}) - \cos(\bar{\theta}) + \bar{r}(\bar{\theta}')^2 \tag{4.4a}$$

$$\bar{\theta}'' = \frac{1}{\bar{r}}\sin\bar{\theta} - \frac{2\bar{\theta}'\bar{r}'}{\bar{r}} \tag{4.4b}$$

The non-dimensional system is then characterised by a single quantity, the dimension-

Table 4.1: Example system parameters demonstrating how a typical human and a small robot can have the same dimensionless stiffness ($k^*$).

| | **Symbol (units)** | **Human** | **Robot** |
|---|---|---|---|
| Leg stiffness | $k$ ($\mathrm{kN\,m^{-1}}$) | 6.87 | 0.392 |
| Mass | $m$ (kg) | 70 | 0.6 |
| Gravitational acceleration | $g$ ($\mathrm{m\,s^{-2}}$) | 9.81 | 9.81 |
| Leg rest length | $r_0$ (m) | 1 | 0.15 |
| **Time scale factor** | $s = \sqrt{\frac{r_0}{g}}$ | 0.319 | 0.124 |
| **Dimensionless stiffness** | $k^* = \frac{kr_0}{mg}$ | 10.0 | 10.0 |

less stiffness $k^*$, given by:

$$k^* = \frac{kr_0}{mg} \tag{4.5}$$

### 4.2.5 Simulation Parameters

From experimental data, Blickhan and Full found the dimensionless stiffness for animal and human bipedal running to have a typical value of $k^* \approx 9.76$ with a standard deviation of 2.44 [1]; thus a value of $k^* = 10$ has been chosen as a starting point for this simulation study. Table 4.1 shows how this value can be arrived at using typical values for a human, in the first column. As shown in the “Robot” column, it is possible to select a spring constant, $k$, to give the same non-dimensional stiffness parameter, while constrained to realistic values of $g$, $m$, and $r_0$ for a small scale test robot. As such the non-dimensional simulation could equally apply to a human-scale robot or the small prototype planned, with the only difference how the resulting non-dimensional values are interpreted.

## 4.3 Existing Approximate Analytical Methods

The flight phase (Fig. 4-2(a)) is characterised by ballistic motion under gravity, and can be solved analytically. However the stance phase dynamics, with the spring force acting along the leg, do not have an exact analytical solution and therefore require approximation, either by numerical methods or using assumptions and simplifications to obtain an analytical solution. For practical, real-time implementation in a robot controller, this approximation should be both accurate and simple to compute. This section will very briefly outline some previously proposed approximations from the literature; the full derivations and details can be found in the relevant references.

Each of these has been implemented in MATLAB in order to form controllers. Generally, in the original papers, the approximations are formulated to answer the forward problem of predicting the outcome of a stance phase given certain touchdown conditions, however in order to control the system the inverse is required, *i.e.* a prediction of what touchdown condition should be chosen in order to achieve a desired forward velocity at liftoff. So, to carry out this conversion, a simple optimisation algorithm has been implemented, involving repeated executions of the forward approximation to find the touchdown angle which achieves the desired velocity. This has the advantage of simplicity and minimises the possibility of inadvertently introducing implementation errors which would skew the resulting comparison. These multiple evaluations increase the computation load, and so this implementation is very likely not the most computationally efficient. For future applications, a careful algebraic handling could perhaps produce an efficient closed form version for each of these approximations. However, for the purpose of a simulation study on a desktop computer the actual efficiency is of relatively little importance. The results of this will be compared later, in Section 4.5.1.

In this simulation environment, exact values for the system parameters, such as the leg stiffness, are available for these approximations, although of course any physical robot's controller would have the rely on measured or computed values, which would create further inaccuracies.

The full detail of the approximations are available in the relevant papers, and are only very briefly outlined here:

**Schwind (2000):** An iterative approximate solution is found by applying the "mean value theorem", considering the system as an integral describing the spring-mass and a perturbation caused by gravity [76].

**Geyer (2005):** This is a much used approximation, which has the benefit of a simple, easy to calculate algebraic form. An important assumptions are of small leg angles and small leg compression, which limit the accuracy to cases of stiff leg spring and low speeds. These allow the dynamics to be expressed as a Taylor series, solved approximately by truncation. [53].

**Saranli (2010):** Here the approximation based on Geyer's approach is improved with the addition of explicit improvements to compensate for the asymmetric effects of gravity, which should improve performance in non-symmetrical stance phases, *i.e.* where there are changes in horizontal velocity. Also included is a correction for damping, although this are not applicable here as the model is lossless [54].

**Yu (2012):** A more algebraically complex approximation to the stance dynamics, using a perturbation solution. This allows for the weakening of some of the assumptions, in particular taking some account of non-uniform angular momentum when solving the rotational motion [55].

## 4.4 Empirical Velocity Controller

This section will develop a very simple analytical approximation for a controller, using admittedly very crude approximations. However, the simple form will allow an empirical approach to be taken to tune the parameters in following sections. Remember from Section 4.2.2, the aim is to select the n<sup>th</sup> touchdown angle ($\theta_n^{td}$) in order that the forward velocity of the next apex ($\bar{x}'_{n+1}$) equals some target velocity ($\bar{x}'_{tgt}$), which in general will be different from the current velocity($\bar{x}'_n$).

### 4.4.1 Controller Formulation

Firstly, consider the steady state case where $\bar{x}'_n = \bar{x}'_{n+1} = \bar{x}'_{ss}$. This occurs when the stance phase is symmetrical, and so the touchdown and liftoff angles are equal in magnitude, and defined as $\bar{\theta}^{ss} = -\bar{\theta}^{td} = \bar{\theta}^{lo}$. This is sometimes referred to as the "neutral angle" [48].

During stance, if the leg spring force is assumed much larger than the gravity force (which is true if the spring stiffness is high) then gravity can be temporarily neglected. In this case, the system will oscillate at its natural frequency and the dimensionless stance time will be approximately $\pi\sqrt{1/k^*}$. Also assuming a constant horizontal velocity and small angles, the horizontal displacement during stance phase is:

$$\Delta\bar{x} \approx 2\bar{\theta}^{ss} \approx \bar{x}'^{ss}\pi\sqrt{1/k^*} \tag{4.6}$$

Rearranging yields:

$$\bar{\theta}_n^{td} = -\bar{\theta}^{ss} \approx -\left(\frac{\pi}{2\sqrt{k^*}}\right)\bar{x}'^{ss} \tag{4.7}$$

Collecting the constants, it can be seen that, for a steady velocity:

$$\bar{\theta}^{td} = -\bar{\theta}^{ss} \approx -\beta_1\bar{x}'^{ss} \tag{4.8}$$

where $\beta_1$ is a positive constant.

Now the effect of deviating from this neutral angle will be considered, *i.e.* how to adjust the touchdown angle to increase or decrease the horizontal velocity. A simple case is when $\bar{x}' = 0$ then $\bar{\theta}^{ss} = 0$, irrespective of the value of $\beta_1$. A common simplification of the SLIP model is to assume a very stiff spring, which leads to an instantaneous stance phase (*e.g.* [5]). Then the system behaves like a ball bouncing off a hard surface inclined at the touchdown angle; as illustrated in Fig. 4-4, this gives a liftoff angle of $2\theta^{td}$. Using a small angle approximation for $\sin(\bar{\theta}^{td})$, the change in the forward component of the velocity, $\Delta\bar{x}'$, is found to be:

$$\Delta\bar{x}' \approx 2\sqrt{2\bar{y}^{apex}}\,\bar{\theta}^{\Delta} \tag{4.9}$$

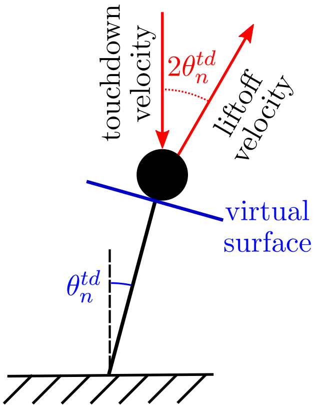


Figure 4-4: If the leg spring is assumed infinitely stiff, the stance phase reduces to the dynamics of a ball bouncing instantaneously off an angled surface (blue). For zero initial horizontal velocity, the touchdown velocity is vertical and the liftoff velocity occurs at an angle of $2\theta^{td}$ (red).

Again, to produce a simple tunable relationship we look to combine the constants into a positive value, $\beta_2$. The greatest simplification can be achieved by assuming that the apex height ($\bar{y}^{apex}$) remains constant between hops and so can be included in the constant ($\beta_{2A}$), to yield:

$$\bar{\theta}^{\Delta} \approx \beta_{2A} \Delta \bar{x}' \tag{4.10}$$

However, as the forward velocity changes from step to step, energy is transferred from potential to kinetic and so the apex height will in fact vary. Therefore, it should be more accurate to not include $\bar{y}^{apex}$ in the constant term, and instead measure the actual value on each hop and use this in the controller. This case, considering the variation in height, will be referred to simple as the “full” version (and the above version, assuming a constant apex height, the “simplified” version), and the new tunable parameter denoted by $\beta_{2B}$:

$$\bar{\theta}^{\Delta} \approx \frac{\beta_{2B}}{\sqrt{\bar{y}^{apex}}} \Delta \bar{x}' \tag{4.11}$$

The disadvantage to including the variation in height is that, in practice, the leg angle needs to be adjusted to its new value during the flight phase. If the actual value of apex height is required, this is likely to only be available after the apex has occurred, leaving only the second half of the flight phase in which to actually carry out the computation and physical re-positioning of the leg. Besides, the apex height can be difficult to measure quickly and accurately. Of course, these disadvantages do not exist in simulation. Therefore, the two possible options will be tested in simulation in order

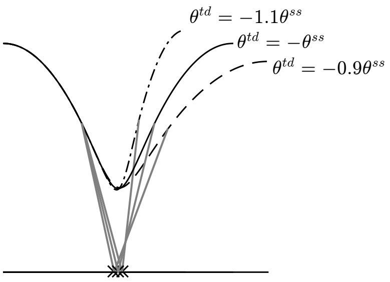

$\theta^{td} = -1.1\theta^{ss}$
$\theta^{td} = -\theta^{ss}$
$\theta^{td} = -0.9\theta^{ss}$

Figure 4-5: Example trajectories at the neutral angle and deviating from it. A more negative angle corresponds to "leaning back" and causes a higher and slower trajectory, and vice versa.

to determine if the increased accuracy from considering the variation in height is likely worth these added implementation complexities.

In equation (4.8) we have an equation which describes the required touchdown angle to maintain the current horizontal velocity, and in (4.10) or (4.11) a description of the change in touchdown angle (relative to the first angle) which will induce a desired change in forward velocity. This is illustrated in Fig. 4-5; at the neutral angle, $\theta^{td} = -\theta^{ss}$ and the next apex state is the same as that which preceded it; deviating from this angle creates a higher and slower or lower and faster apex state. Thus, summing these components gives a control logic to compute the system input ($\theta^{td}$) based on the current state ($\bar{x}'_n$) and demanded next state ($\bar{x}'^{tgt}_{n+1}$), with two tunable parameters, $\beta_1$ and $\beta_2$. With the two options discussed for $\beta_2$, the controller has two possible forms (the "simplified" and "full" versions):

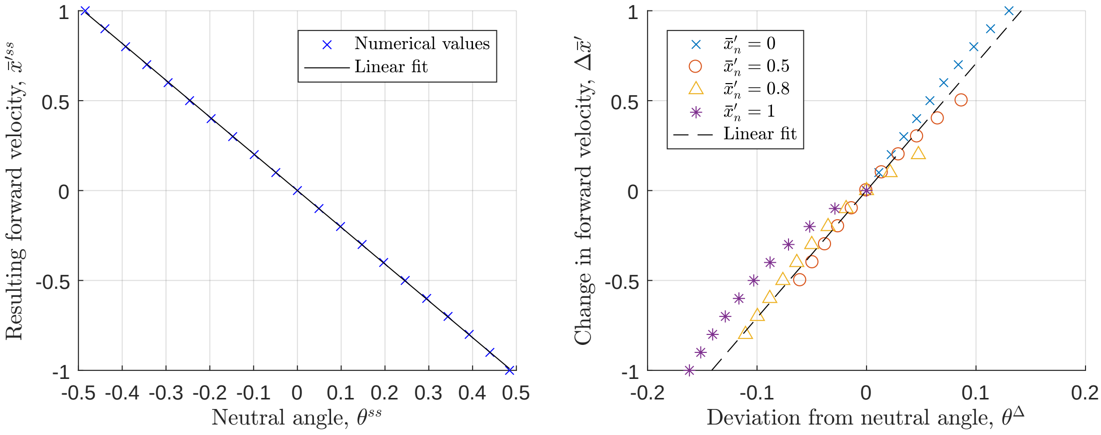
$$\bar{\theta}^{td}_n = \underbrace{-\beta_1 \bar{x}'_n}_{\bar{\theta}^{ss}} + \underbrace{\beta_2(\bar{x}'^{tgt}_{n+1} - \bar{x}'_n)}_{\bar{\theta}^{\Delta}} \tag{4.12a}$$

$$\bar{\theta}^{td}_n = \underbrace{-\beta_1 \bar{x}'_n}_{\bar{\theta}^{ss}} + \underbrace{\frac{\beta_{2B}}{\sqrt{\bar{y}^{apex}_n}}(\bar{x}'^{tgt}_{n+1} - \bar{x}'_n)}_{\bar{\theta}^{\Delta}} \tag{4.12b}$$

### 4.4.2 Numerical Verification

The linear relationships on which the controller is based can be verified in simulation.

Considering first the linear relationship predicted in (4.8), a brute force iterative method of repeated simulations has been used to find the neutral angle ($\theta^{ss}$) for a range of steady state velocities between $\bar{x}'^{ss} = -1$ and 1. The leg stiffness used was $k^* = 10$ with the apex height set to maintain the total dimensionless system energy (found by $E^* = \bar{y}^{apex} + \bar{x}'/2$) as $E^* = 1.5$. The result is shown in Fig. 4-6(a), with a best fit line approximating $\beta_1$ showing a good fit.



(a) (b)

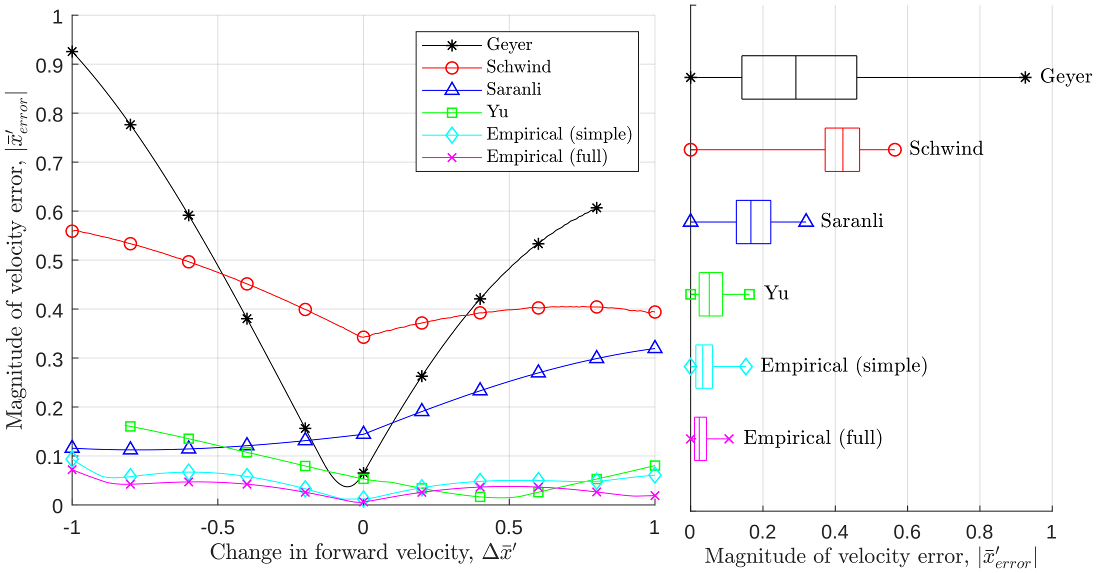
Figure 4-6: Numerical simulation results for verifying (a) the linear relationship between steady state velocity and touchdown angle from (4.8), and (b) how deviation from that angle creates a change in velocity as predicted by (4.10).

For the second relationship, between $\theta^{\Delta}$ and $\Delta\bar{x}'$ from (4.10) or (4.11), similar simulations were used to find the touchdown angle ($\theta^{td}$) which results in a specific change in forward velocity ($\Delta\bar{x}'$), over a range of starting velocities ($\bar{x}'_n$). The actual deviation from the neutral angle was then calculated at each point as $\theta^{\Delta} = \theta^{td} - \theta^{ss}$, with $\theta^{ss}$ from the first set of simulation.

Figure 4-6 shows the values from the numerical simulations. It can be seen that the linear relationships predicted by equation (4.8) holds very well for finding $\beta_1$. In figure 4-6(b) the line shown is for the more simple relationship (4.10), and shows a reasonable fit, although the relationship depends to some extent on the initial velocity. These graphs show that the linear relationships proposed do describe the nature of the system, and therefore hold promise as the basis of the empirical controller.

## 4.5 Comparison of Empirical and Analytical Controllers

Controllers have been implemented based on each of the approaches in Sections 4.3 and 4.4 and used to control a numerical simulation of the full SLIP dynamics using Matlab. These simulations aim to test the controllers' ability to make large changes in forward velocity over a single stance phase. Each run simulates only a single step, with a starting velocity in the range $\bar{x}'_n = [0, 1]$ and target velocities of $\bar{x}'_{n+1} = [0, 1]$ This range corresponds to stopping or starting a jogging pace in a single step for a human (or human-sized robot). In each run, the robot starts at the apex and the controller selects the touchdown angle based on the target velocity, then the full dynamics, defined by (4.3) and (4.4), are simulated until the next apex state, where the achieved velocity is measured, and the absolute value of the difference is recorded.



(a) (b)

Figure 4-7: The average velocity error for each controller, plotted as (a) the mean value for all those that did not fall, for each value $\Delta\bar{x}'$ and (b) as a box plot, showing the minimum, lower quartile, median, upper quartile and maximum of the error for each controller over the full range of conditions tested.

### 4.5.1 Controller Performance Comparison

The results for $k^* = 10$ are compared in Fig. 4-7, plotted against the desired change in velocity ($\Delta\bar{x}' = \bar{x}'_{tgt} - \bar{x}'_n$); the magnitude of this value corresponds to how far from a symmetric stance the motion is. As starting velocity is always positive, positive values $\Delta\bar{x}'$ mean a increase in speed, and negative values slowing down or stopping.

The error value for a particular step is the difference between the target and actual velocity at the $(n+1)^{\text{th}}$ apex:

$$\bar{x}'_{error} = \bar{x}'_{n+1} - \bar{x}'_{tgt} \tag{4.13}$$

However, when considering several steps, taking an average of this value would cause positive (too fast) and negative (too slow) errors to cancel out, giving a misleading average error of close to zero despite a poor performance in both cases. Therefore, for the plots below the average of the magnitude of this error has been plotted, $|\bar{x}'_{error}|$.

Most of the approximations make an assumption of a symmetrical stance phase, conservation of angular momentum or equivalents. Therefore, the larger changes in velocity (*i.e.* more agile gaits) result in larger errors as they breach these assumptions. Both the simplified and full empirical controllers show good performance across the range of $\Delta\bar{x}'$ when compared against the approximation based controllers, with the simplified version showing a slight deterioration in performance, as expected due to the limitation of assuming a constant apex height. The empirical controllers were pre-tuned, off-line, using a set of simulated results over the same range of $\bar{x}'_{tgt}$ and $\bar{x}'_n$ as the test, using the regression method described below in Section 4.6.

### 4.5.2 Effect of Leg Stiffness Value

All the analytical approximations and the motivation for the form of the empirical controller in some way use an approximation of a stiff leg spring. This is equivalent to an instantaneous stance phase, because a very stiff spring would have a very fast natural frequency and therefore rebound in a short time. Therefore, to test the controllers' sensitivity to this approximation the simulations were repeated at various leg stiffness ($k^*$) values, and the results are shown in Fig. 4-8.

### 4.5.3 Agile Gait Control

The previous simulations have been for a single step at a time, with the initial apex conditions pre-set. Of course, for physical experiments and applications, the robot will instead need to carry out a series of steps one after another, with the final state of one step becoming the initial state of the next; such motion will be simulated in this section as preparation for the experimental implementation of the controller later in the chapter. An example of a simulated motion is shown in Fig. 4-9. The lower sub-figure shows how varying the horizontal velocity has a direct effect on the step length, with higher speed leading to longer steps. This may be useful in situations where there are limited safe positions for foot placement.

The simulation environment allows for a large number of steps to be considered. Results will be reported to measure the stability and accuracy of the controllers.

Firstly, the stability of the system will be measured by the mean number of steps the robot can take before falling. This is more challenging when the velocity is higher and when there is larger changes from step-to-step. The demand velocity for each step will be randomly chosen between zero and a maximum value, $\bar{x}'_{max}$; higher values of $\bar{x}'_{max}$ will produce gaits which are faster on average and more agile, *i.e.* have more variation between steps. The maximum number of steps for a run will be 100, and so the "stability score" will be defined as the mean steps to failure with a maximum of 100.

The results, for the two empirical options and the analytical approximations (see Section 4.3) over 100 simulated trails of 100 steps each with $k^* = 10$, are plotted in Fig. 4-10. It can be seen in each case there is a sudden drop as the maximum velocity reaches the point where the controller is no longer able to model the system accurately enough to prevent falling over. The two options for the empirical controller both remain stable at higher velocities than any of the analytical approximation based controllers, but there is little difference between them.

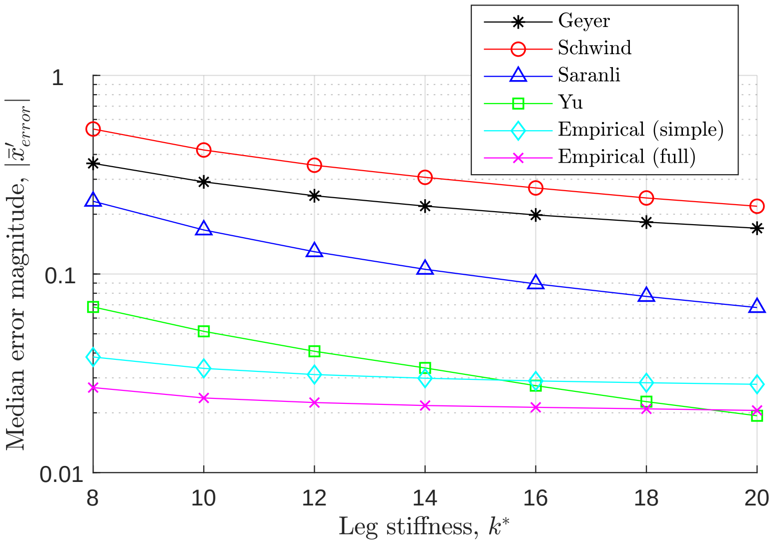


Figure 4-8: Comparison of median error magnitude for each controller considered against a range of leg stiffness values. For each point, the equivalent of the simulations for Fig. 4-7 were performed, with the median of the absolute error is reported, on a logarithmic vertical scale. All the controllers are more accurate when the leg is stiffer, as a stiff leg (equivalent to instantaneous stance phase) is a common assumption, however the empirical controllers suffer this effect much less as they can be re-tuned for each stiffness value.

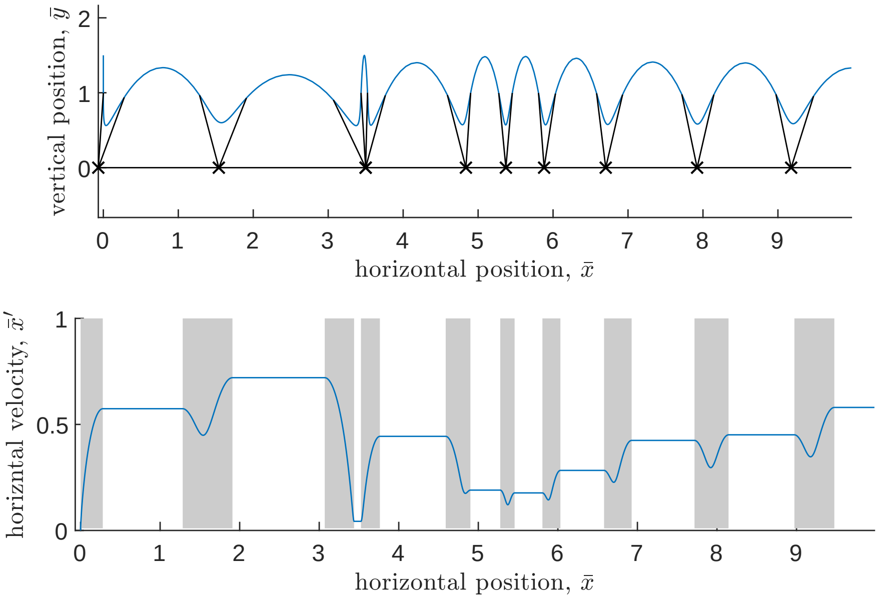


Figure 4-9: A simulation example of agile motion, with large changes in velocity between steps. Changes in forward velocity (lower plot) lead to different length steps (upper plot). Black crosses denote the foot landing positions and grey bars the stance phases.

Secondly, for those controller and $\bar{x}'_{max}$ combinations that are stable, the accuracy can be measured. Again, the demand profile consists of random values between zero and $\bar{x}'_{max}$, and the accuracy measured as the magnitude of the difference between demand and actual velocity, taken as an average over $10,000$ steps. Accuracy has only been considered in the cases where the stability score was 100 as the required steps could not be completed in the unstable cases.

Upon each step the absolute difference between the demand for that step and the actual velocity is recorded. Median and upper and lower quartile values are shown in Fig. 4-11.

Across the range of $\bar{x}'_{max}$ tested, the median error was between 25% and 40% lower for the option "full" empirical controller over the "simplified" version. The disadvantage of using the full version for the physical implementation (Section 4.8) would be the requirement to measure the apex height for use in the controller computation. This would mean the controller could only compute the required angle after the apex has been reached, even assuming instantaneous measurement data being available. This would leave less time for the leg to be physically re-positioned before the touchdown, which is of critical importance. Therefore, with a view to physical implementation, the reduction in performance is considered acceptable and the simpler version of the controller will be used below.

## 4.6 Adaptive Gains Through Multiple Linear Regression

For the empirical control strategy suggested here the two controller parameters, $\beta_1$ and $\beta_2$, need to be selected. This could be done though the analytical approximations above, however due to their very approximate nature the results would be poor. In the results above, the values have been pre-set through simulation by fitting values to large numbers of single-step simulations, with the initial conditions for these steps varied over the range of expected conditions. However, this is not such an attractive approach for physical robots; large numbers of repeated tests would require a lot of time and effort. In this section, an approach will be taken to approximate $\beta_1$ and $\beta_2$ on-line, using data from a small number of preceding steps with arbitrary initial conditions (rather than systematically varied over a range).

To do this, a tuning method similar to the one used for the one-dimensional controller in Section 3.3.2 will again be used, described again here for completeness. Noting that when the robot is at the $\mathrm{n^{th}}$ apex point information will be available from the preceding stance phases, this information can be used to inform the choice of the $\mathrm{n^{th}}$ touchdown angle by selecting the $\beta_1$ and $\beta_2$ values which would have correctly predicted the outcomes of these previous stance phases. Let $p$ be the number of previous steps to be considered for the selection of $\beta_1$ and $\beta_2$ (the "memory" of the controller); if the vector of actual touchdown angles (the system inputs) which were applied are denoted by $\Theta$, let the values that would have been chosen using a particular set of $\beta$ values be

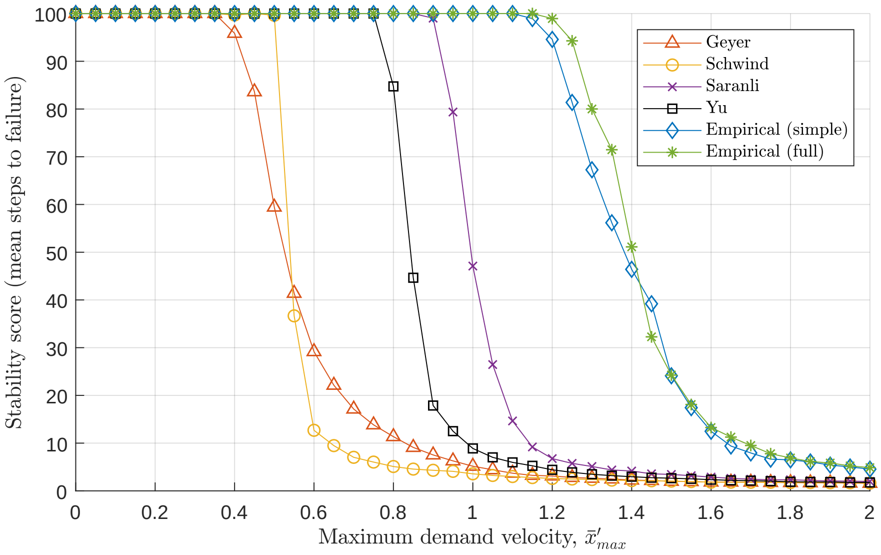


Figure 4-10: Stability for the analytical approximations and the two options for the empirical controller. The "stability score" is the mean number of steps completed out of a maximum of 100, over 100 separate trials. For each step, the demand velocity is chosen randomly varying between zero and $\bar{x}'_{max}$.

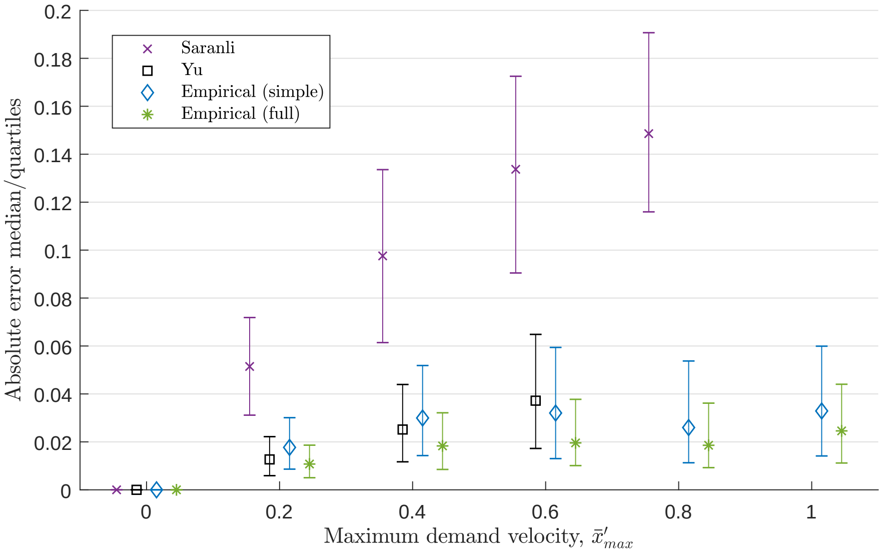


Figure 4-11: The absolute errors recorded over $10,000$ simulated steps, using the most promising analytical controllers ("Yu" and "Saranli") and the two empirical formulations under consideration. The empirical controllers show a slight improvement over the best approximation-based controller, with a small further improvement of the full version over the simplified one. Plotted are the medians of the absolute errors, and the associated upper and lower quartile bounds. Missing bars (at higher velocities) indicate the simulated robot fell before completing all the steps, and corresponds to values after the fall in stability shown in Fig. 4-10.

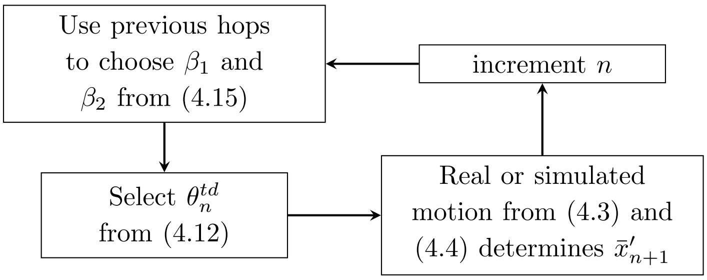

Use previous hops to choose $\beta_1$ and $\beta_2$ from (4.15)
increment $n$
Select $\theta_n^{td}$ from (4.12)
Real or simulated motion from (4.3) and (4.4) determines $\bar{x}'_{n+1}$

Figure 4-12: The adaptive controller update logic.

$\hat{\Theta}$, then:

$$\hat{\Theta} = \begin{bmatrix}\hat{\theta}_{n-p}\\ \vdots \\ \hat{\theta}_{n-1}\end{bmatrix} = \underbrace{\begin{bmatrix} -\bar{x}'_{n-p} & \left(\bar{x}'_{n-p+1} - \bar{x}'_{n-p}\right) \\ \vdots & \vdots \\ -\bar{x}'_{n-1} & \left(\bar{x}'_{n} - \bar{x}'_{n-1}\right)\end{bmatrix}}_{\boldsymbol{X}} \underbrace{\begin{bmatrix}\beta_1\\ \beta_2\end{bmatrix}}_{\boldsymbol{\beta}}$$

The errors to be minimised, $\boldsymbol{\epsilon}$ is:

$$\boldsymbol{\epsilon} = \Theta - \hat{\Theta} = \Theta - \boldsymbol{X}\boldsymbol{\beta} \tag{4.14}$$

The gain values to minimise the sum of the squares of the errors, $\boldsymbol{\epsilon}'\boldsymbol{\epsilon}$, are found by multiple linear regression:

$$\boldsymbol{\beta} = \left(\boldsymbol{X}^{\mathsf{T}}\boldsymbol{X}\right)^{-1}\boldsymbol{X}^{\mathsf{T}}\Theta \tag{4.15}$$

With two gain parameters to determine, the minimum number of previous steps to consider is $p = 2$, in which case:

$$\boldsymbol{\beta} = \boldsymbol{X}^{-1}\Theta \tag{4.16}$$

The controller update sequence is illustrated in Fig. 4-12. The first $p$ hops must be completed before this method can be applied to set the $\beta$ values, and so some initial values are required to control these first few steps. In simulation, the fixed values found above will be used for the initial values, and for the experimental hopper some values are found manually. As long as these are sufficient for the system to complete $p$ hops without falling over, the adaptive controller can be activated, beginning at the $(p+1)^{\text{th}}$ hop.

## 4.7 Adaptive Simulation Results

It was shown in Section 4.5.1 that, with the optimum gain values, the empirical controller performs well. In this section, the adaptive approach will be introduced for selecting the gains; this means the gains will be selected during the simulation, based on limited information, and so will inevitably cause some reduction in performance. The adaptive gains will therefore be compared against a "fixed gain" version of the controller using the same values as above (Section 4.4.2), which were found by fitting to a large sample. For the adaptive controller, the $p$ value sets how long the "memory" is, *i.e.* how many previous steps are considered for selecting the new gain values. A range of values will be compared in order to asses the trade-off between fast adaptation to changes in the system against improved stability and accuracy.

### 4.7.1 Agile Control

Firstly, the adaptive controller will be tested under the conditions used to compare the analytical controllers in Section 4.5.3 above, namely the mean steps to failure ("stability score", with a maximum of 100) and the average of the absolute velocity error over $10,000$ steps.

By using previous steps to select the gains any error or inaccuracy in the linear modelling for these steps will inevitably be amplified, and so the adaptive controllers are less stable (more likely to cause the robot to fall) than when fixed gains are used. In Fig. 4-13 it can be seen smaller $p$ values creates more instability, as small errors have a larger weight in the tuning; whereas when $p$ is large, the average is taken over a larger number of data points and so the result is more stable. The smallest value possible is $p = 2$, which sometimes falls even at low speeds, while at and above $p = 10$ there is little difference from the fixed gain version.

The adaptive controllers use only a limited amount of information from the previous steps, for which the target velocity was randomly varied, and so may not provide a representative sample. It would therefore be reasonable to expect a deterioration in performance compared to the fixed gains which were selected using a large set of steps specifically simulated to evenly cover the targeted space of velocities (in this case, $\bar{x}' = 0$ to 1). However, Fig. 4-14 shows that for most velocity values, all the adaptive controllers outperform the fixed gain controller. This is because for $\bar{x}'_{max}$ values less than 1, the demanded velocities do not actually cover the full range of steps used to determine the fixed gains, and so a better set of gain values can be found by using even a limited number (*i.e.* $p$) of actual steps. This suggests that, whether tuned off-line or on-line, it is important that the conditions used when selecting the gain values match those which will actually be encountered – potentially a significant advantage of tuning on-line based on real data as the robot progresses.

### 4.7.2 Adaptation to a Change in System Parameter

One advantage to a system where the controller gains are changed during a run is adaptation to changes in the robot dynamics. Changes in the dynamics may occur if, for example, the mass changes (such as carrying a payload) or if it moves onto a different terrain with stiffer or softer ground. In the simulation, such changes can be created by changing the value of $k^*$, the dimensionless spring stiffness. In the following tests, $k^*$ is changed instantaneously between two stance phases, and the control receives

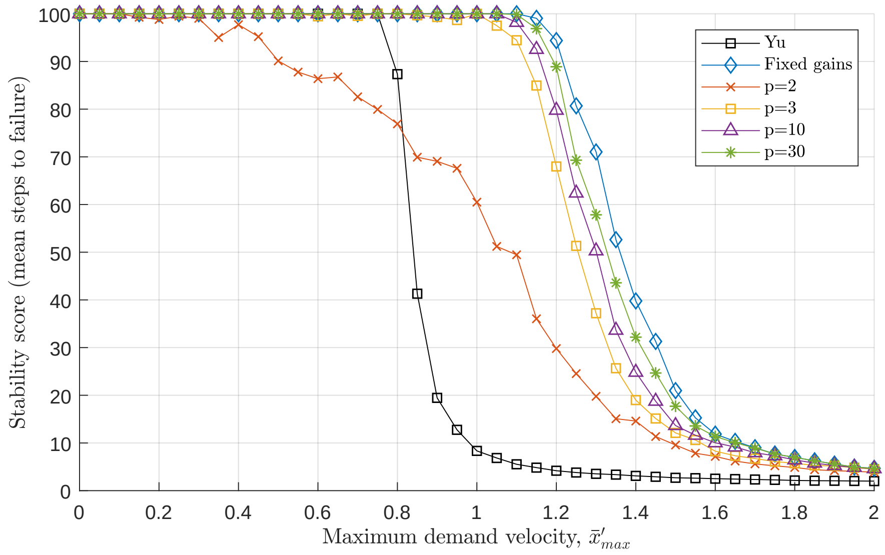


Figure 4-13: Stability for the (simplified) empirical controller, for a range of $p$ values. The "Fixed gain" version is analogous to an infinite $p$ value, and is the same as the "Empirical (simple)" controller from Fig. 4-10. The "stability score" is the mean number of steps completed out of a maximum of 100, over 100 separate trials. For each step, the demand velocity is chosen randomly varying between zero and $\bar{x}'_{max}$.

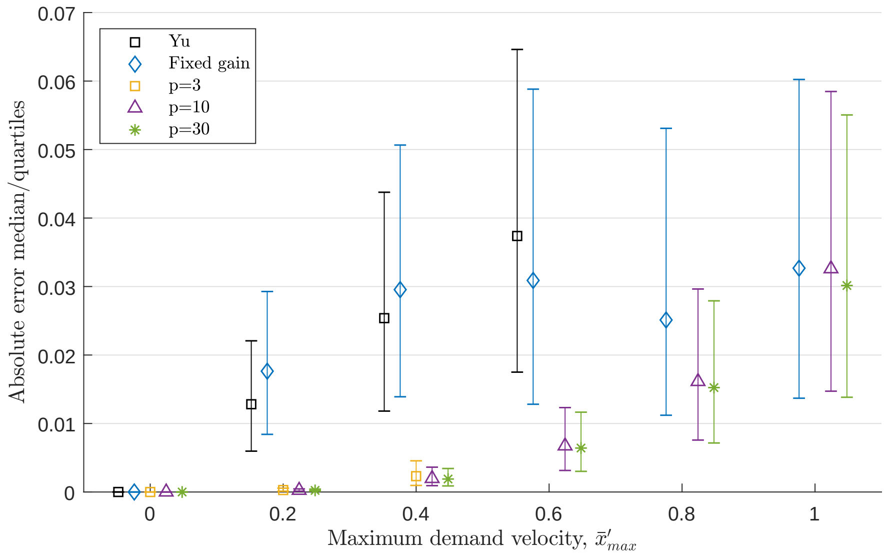


Figure 4-14: The errors in velocity recorded over 10,000 simulated steps for a range of $p$ values and the "fixed gains" empirical controller, which is the same as the 'Empirical (simple)" controller from Fig. 4-11. Plotted are the median and the associated upper and lower quartile of the error, defined as the absolute difference between the demand and actual velocity for each step.

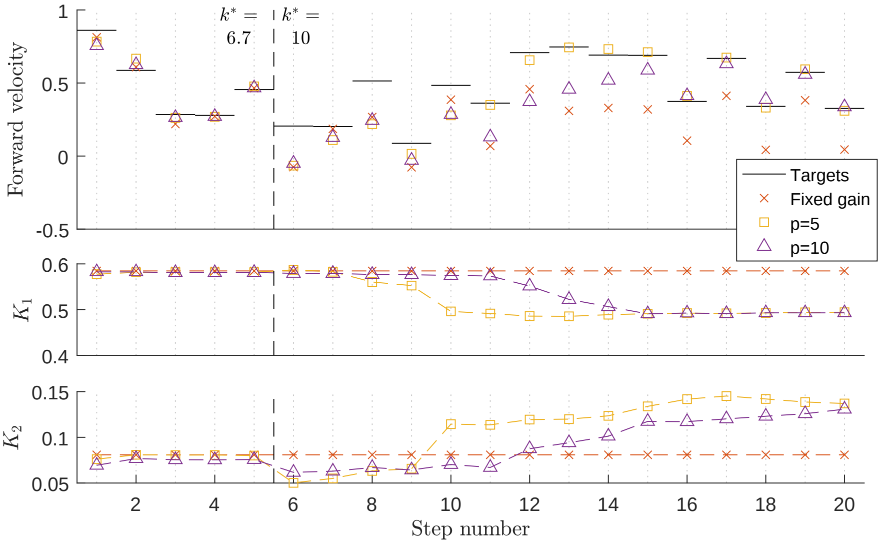


Figure 4-15: An example of one run with a change in $k^*$ between steps 5 and 6. Shown are (top) the achieved velocities compared with those demanded and (bottom) the gains as they are re-tuned by the adaptive controller with $p = 3$ and $p = 10$.

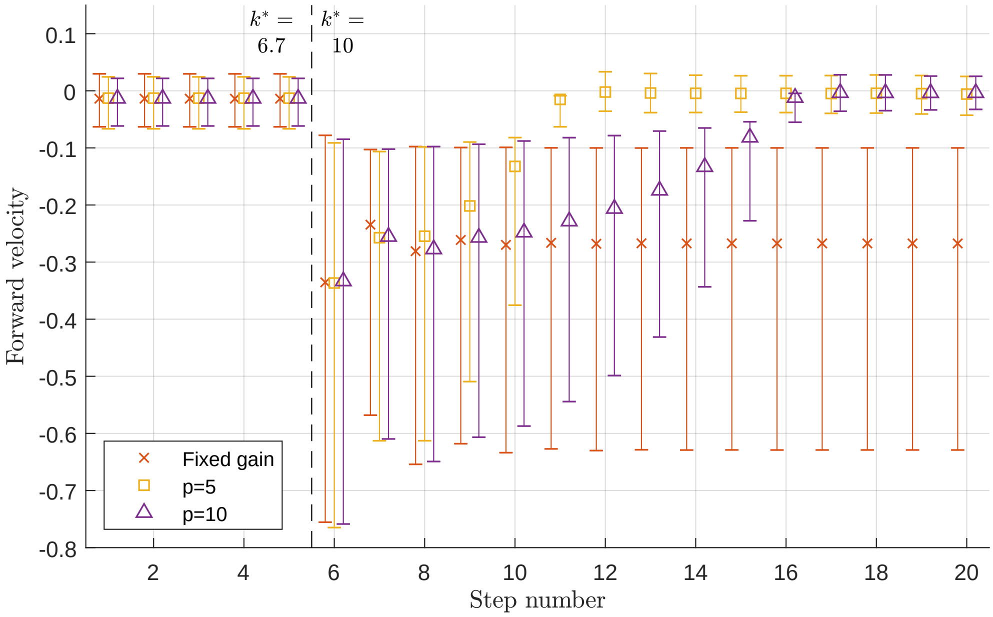


Figure 4-16: The response to a change in system parameter. For hops 1–5, $k^* = 6.7$ and thereafter $k^* = 10$. Shown are the mean and standard deviation of the velocity error over $10,000$ simulations with random velocity demands on each step. When well tuned, the average error should be zero and a more consistent controller will have a smaller standard deviation.

no notification of this change, other than the changes in velocity resulting from the new system dynamics.

An example of one such run if shown in Fig. 4-15 where $k^*$ is changed from 6.7 to 10 between steps 5 and 6. The fixed gain controller is pre-tuned (using the same procedure described above) for the initial value of $k^*$. The adaptive controllers, the gain values are allowed to settle over a number of previous steps, significantly larger than $p$, before the plotted steps. Fig. 4-16 shows aggregated results from $10,000$ such runs, with a new random set of target velocities for each run; the median and quarterlies of the error (defined here as the difference between demand and achieved velocities) on each step are shown. An accurate and consistent controller would produce a bar centred around zero with small error bars. Before the change, all the controllers track the demand fairly closely, but immediately after (step 6) suffer the expected error from changed system dynamics. However, the adaptive controllers are able to re-tune, and recover performance; this takes approximately $p$ steps, as before this some results from the old system dynamics are still included in the multiple linear regression model. Therefore, lower values of $p$ will adjust to such changes more quickly. It was found that, for this particular step change in $k^*$, values below $p = 5$ caused the robot to fall in most cases, while at $p = 5$, eight of the $10,000$ runs (0.1%) caused the robot to fall and are excluded from the plot. No such failures occurred for the fixed gains or $p = 10$ controllers.

## 4.8 Physical Experiments

In order the verify the simulations described above, the proposed controller has been demonstrated on a physical robot. This section details a physical robot made for this purpose, and some experiments implementing the controller with adaptive and fixed gains.

### 4.8.1 Robot Design

In order to carry out experiments, a physical hopper is needed. For two-dimensional hopping, the robot must be allowed to move in two axes, but be contained in the third so it can traverse a plane, but the constraining mechanism must be low mass and friction to avoid introducing significant disturbances into the dynamics. Achieving this on a perfect plane is difficult due to the need for linear rails which add inertia and friction, but can be approximated by attaching the robot to the end of a boom which is pivoted in two directions. This means the robot actually traverses the surface of a sphere, but so long as the boom is long compared to the hop height and length, this surface is approximately planar. The long lever arm of the boom also means any friction in the pivot is more easily overcome. This planned layout is shown conceptually in Fig. 4-17. The horizontal and vertical motion are not directly controlled. Instead, the robot has two further degrees of freedom, over which it has partial control. Firstly, the leg angle, which can be set during the flight phase but is allowed to move freely during stance. Secondly, the leg length which will be at its maximum during flight, and in stance will compress, with control only over the gross energy added through the valve opening time.

The robot used in these experiments is shown in Fig. 4-18. The long boom to constrain the robot to the sagittal plane can be seen, with the leg able to pivot in this plane also. The body is attached rigidly to the boom to prevent rotation. In this way, the robot approximates the SLIP model described in Section 3.2.3, moving in two dimensions. By having only the leg rotate, the rotational inertia is minimised in order to better approximate the point mass of the SLIP model.

The leg is made from a pneumatic actuator with valves for inlet (from a 6 bar supply) and outlet (to atmosphere), with the same actuation method as for the one dimensional hopper described above in Section 3.5. Pneumatic actuation provides a convenient way to rapidly inject energy at a specific time, by simply opening a value. To approximate the lossless SLIP model, the same amount of actuation (*i.e.* the same valve opening time) is applied during each stance phase so the system should settle to an approximately constant energy level.

The leg actuation system must set the leg angle to a desired value during the flight phase, but during the stance phase the leg should be allowed to rotate freely to follow the natural dynamics (with imposed torque as close to zero as possible). The actuation system has been designed to allow this by being able to release the leg when desired. As shown in Fig. 4-19, this system comprises three parts: the non-rotating body (coloured green), the actuating servos and connected arms (red), and the rotating leg (blue). A central shaft connects these parts, allowing the leg and servo arms to rotate about a common axis. Fig. 4-20 shows how the two servo arms can move back and forth together, (a)-(c), to push the pin (grey circle) which protrudes from the leg, driving the leg to a demanded position, *e.g.* during flight phase. In order to allow the leg to swing freely (*e.g.* during stance phase), the servo arms move away from each other, (d), releasing the pin. A physical electronic switch in the foot to detect the moment of impact with the ground (touchdown) via an interrupt pin on the Raspberry Pi minimises the delay in activating this release.

To calibrate the required low-level inputs to the servos a step input test was performed. For this, the robot was suspended above the ground so the leg could be moved back and forth freely, and the servos set to open and then drive the leg to a given angle. Fig. 4-21 show the results of such a test once the low level servo control had been manually tuned. This suggests that moving from the “open” position (grey areas) to a demanded angle can be achieved in approximately 120 ms; this measured time will include the latency in angle measurement from the external tracking system. This must occur during the flight phase, so that the leg is in position ready for the following touchdown, which, in the experiments described below, provides approximately 200 ms to 250 ms for this.

Power (both electrical and pneumatic) are provided through the boom for simplicity. Position estimation is carried out though an on-board IMU and an external tracking

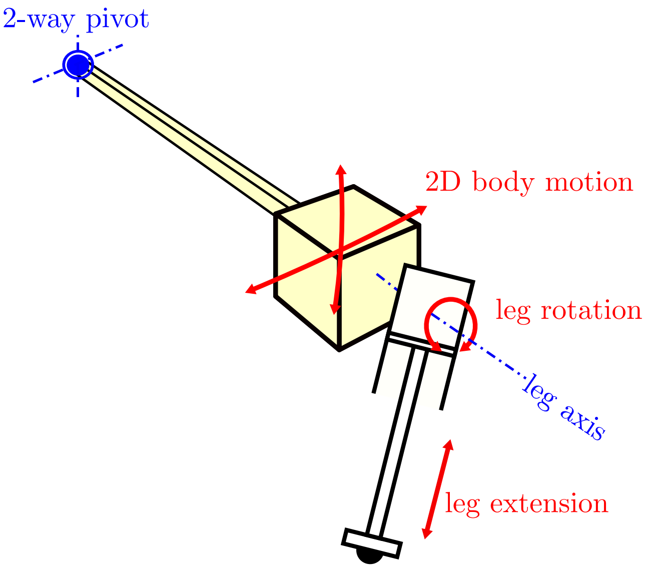

2-way pivot
2D body motion
leg rotation
leg axis
leg extension

Figure 4-17: Concept sketch for the 2D physical robot, showing the degrees of freedom (red) and rotational axes (blue). The horizontal and vertical body displacements are approximated by a 2-way pivot at the other end of a boom (yellow). The leg has an additional two degrees of freedom: its rotation which can be controlled in flight, and its extension along which the input can be applied during stance.

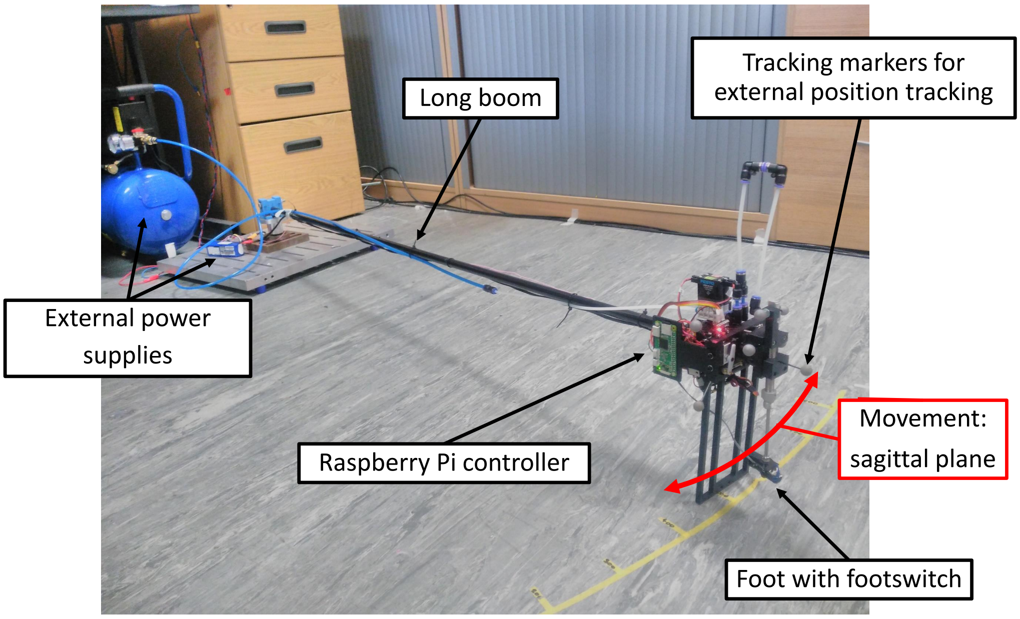

Long boom
Tracking markers for external position tracking
External power supplies
Raspberry Pi controller
Movement: sagittal plane
Foot with footswitch

Figure 4-18: The experimental robot hopper, constrained to the sagittal plane by a long boom.

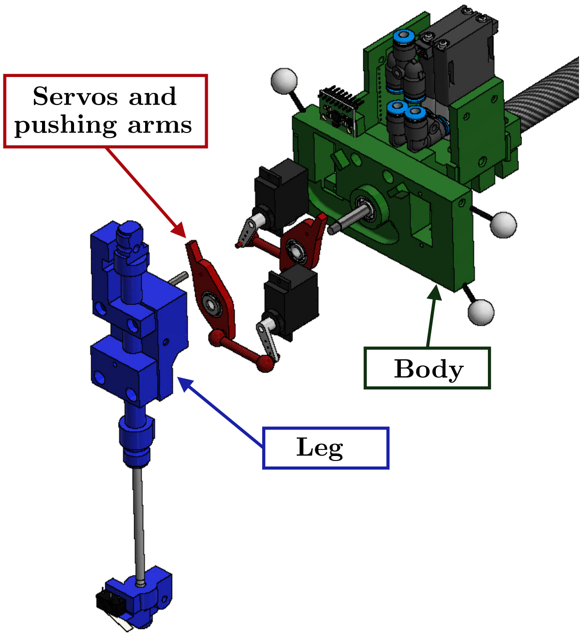

Servos and pushing arms
Body
Leg

Figure 4-19: Exploded 3D view of the design for leg actuation.

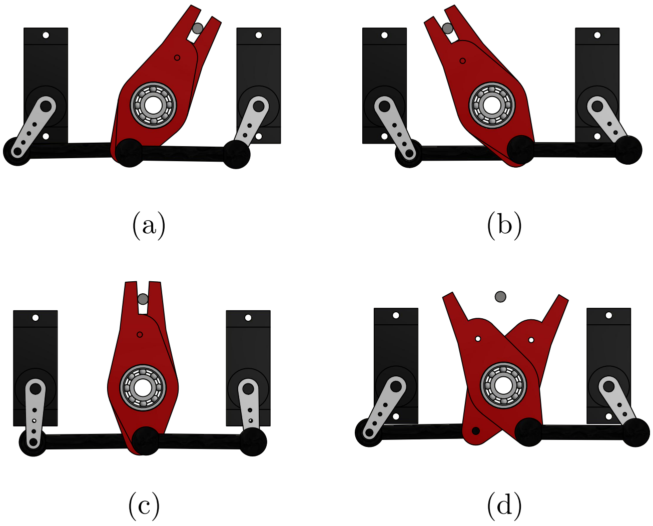


Figure 4-20: The servo actuation method. The pin attached to the leg protrudes into the page between the servo arms, and can be forced to (a) a positive angle, (b) a negative angle, (c) the zero position, or (d) the servos can opened to allow the leg to swing freely (during stance phase).

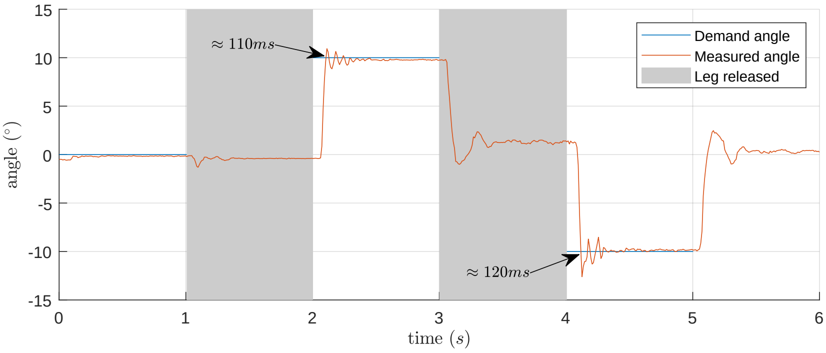


Figure 4-21: Step input to test low-level leg angle control using the servo system described. Reasonable angles can be achieved in approximately 120 ms.

system which runs on a laptop, with these two sources combined using a simple complimentary filter. The control algorithm described in Section 4.4 is carried out on-board on a Raspberry Pi Zero (1GHz, single-core CPU and 512MB RAM)[1].

The robot mass is approximately 600 g and with the actuator extended the leg length (from centre of mass to foot) is 150 mm. The stiffness of the leg is created by the compression of air trapped in the actuator, which makes it non-linear. However an approximate value analogous to the spring stiffness in the model (Section 3.2.3) can be found from the stance time. A typical value is 0.15 s, which, assuming this is half the natural frequency, suggests a stiffness of approximately 260 N m$^{-1}$. Together, these values correspond to a dimensionless stiffness of $k^* = kr_0/mg \approx 6.7$.

### 4.8.2 Experimental Testing Procedure

A fixed pair of $\beta$ gain values were determined manually through trial and error, to allow the robot to successfully hop along the track without falling over, approximately tracking demanded velocities. This is required because the adaptive controller requires data from at least $p$ steps before it can be applied.

To test the tracking of an agile (*i.e.* varying each step) velocity demand, a sequence was generated of 17 hops corresponding to hopping along and back for most of the length of the track, with demand values (initially positive and then negative) varying randomly in magnitude from zero to 400 mm s$^{-1}$ (for comparison to the simulation results, the equivalent non-dimensional value is $\bar{x}' \approx 0.33$). The same fixed sequence was used for each test for comparison purposes. For each test, before applying this sequence, the controller was initialised with the manually selected $\beta$ values and the robot commanded to hop back and forth along the track with random magnitude velocity demands (different each time). Once at least $p$ hops had been completed the adaptive controller was activated (except for the fixed gain controller) and the next time the robot reached the start of the track the test sequence of demands was applied.

---

[1] See `www.raspberrypi.org/products/raspberry-pi-zero-w/`

In this way the adaptive controller was set up with a different random sequence of previous steps for each run, giving a representative range of values from which to tune the controller gains.

This procedure was completed for 15 runs, giving a total of 255 individual steps, for each of three controllers: using the manually determined fixed gains, and for $p = 10, 30$. Values of $p$ below 10 were found to cause the robot to fall without reliably completing the run, and so were not included in the test.

### 4.8.3 Experimental Results

Fig. 4-22 shows the sequence of demand values, with the mean and standard deviation of the measured velocity under each of the controllers (fixed, $p = 10$ and $p = 30$) at each step. Under all three settings, the controller is able to approximately track the demanded velocity. Although there is, inevitably, variation caused by the underlying inaccuracies in measurement and actuation of the robot, there are some meaningful trends that can be discerned.

Firstly, comparing the fixed gain version (blue diamond) to $p = 30$ (green star), the introduction of the adaptive controller generally reduces the error – the mean is closer to the target on most (15/17) of the steps in Fig 4-22. This can be measured by a reduction in the overall mean of the absolute error for all 255 individual steps from $253\,\mathrm{mm\,s^{-1}}$ to $191\,\mathrm{mm\,s^{-1}}$ (significant within 1% under the Mann-Whitney U test[2][77]). The adaptive gains allow the controller to tune accurately to the system dynamics, outperforming the fixed values which were manually selected by trial and error.

Secondly, the adaptive controllers show a larger amount of variability – *i.e.* the standard deviation bars in Fig. 4-22 become larger. The mean standard deviation is $73\,\mathrm{mm\,s^{-1}}$ for the fixed gains, $85\,\mathrm{mm\,s^{-1}}$ at $p = 30$, and $107\,\mathrm{mm\,s^{-1}}$ at $p = 10$. This is as expected from the simulation results, because adapting the gains on-line effectively creates variation in the controller settings, which will lead to variation in the inputs selected and the resulting velocities.

As was seen in the simulation results previously, the value of the update parameter $p$ has an influential effect on the robot stability. When setting up the robot, it was generally found that below $p = 8$ the robot would fall in the majority of runs, and $p = 10$ was required to consistently complete the required steps and so this was chosen as the lowest value for the larger numbers of runs required to compute the data in Fig. 4-22. The higher values of $p$ required compared with the simulation (*c.f.* Fig. 4-13) should be expected as a consequence of the introduction of inaccuracies associated with measurement and actuation in physical systems. The variability increases for lower $p$ values because the selection of controller gain values are also subject to noise in the data, and a larger sample of previous steps ($p$) allows some of this noise to be

---

[2] Implemented through the MATLAB function *ranksum()*, see `www.mathworks.com/help/stats/ranksum.html` for details

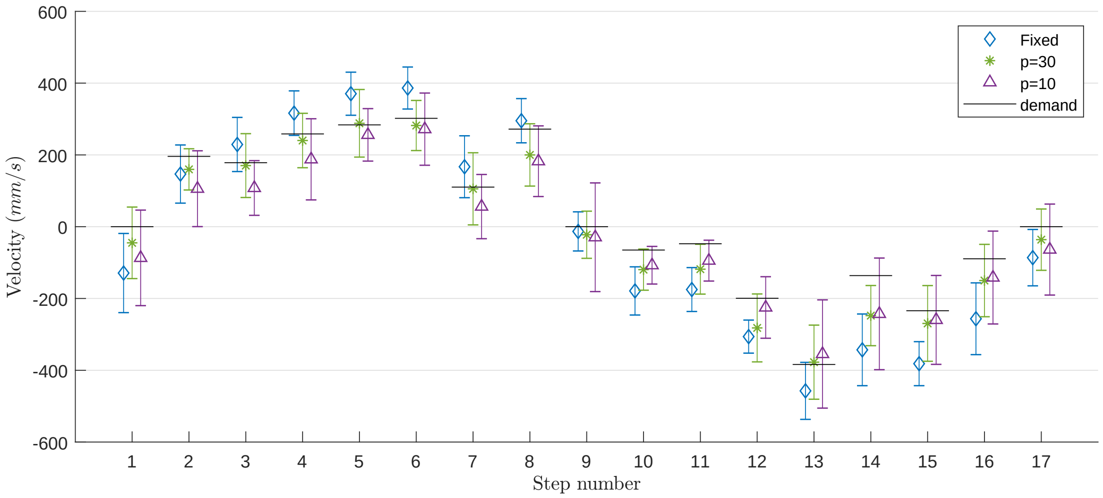


Figure 4-22: Results for velocity tracking of a fixed series of random velocity demands, first positive then negative. Shown are the mean of the velocity for each step in the sequence over 15 trial runs, with bars representing the standard deviation for that step in the sequence. Compared are the controller with fixed (manually determined) gains, and adapting over $p = 30, 10$ previous steps.

smoothed out. In this sense, the fixed gain controller can be considered as the extreme case of infinite $p$.

The choice of $p$ value is then a trade-off; higher $p$ values correspond to using more previous steps to tune the controller and so tends to be more accurate and is likely more robust to noise. However lower $p$ values will respond more quickly to changes in the environment and to remove inaccuracies, using fewer previous steps. In practice, it may not be necessary to update very rapidly; some changes may happen slowly, such as a change in actuation as a battery runs down, or it may be acceptable to adapt more slowly to a new running surface if these changes will be infrequent. In these cases, a $p$ value of 30 or even higher may be adequate. Even a high $p$ value retains a practical benefit in setting up a robot; the improved accuracy for $p = 30$ over the manually chosen fixed gains demonstrates it can be difficult to select the optimum gains by trial and error alone.

### 4.8.4 Limitations and Failure Modes

Even with fixed controller gains, there is still some inaccuracy and variability in the results, due to practical limitations in the robot hardware. In particular, the leg rotation actuator is only just able to move the leg to the demanded position within the flight phase, and there may still be some oscillations in leg position at the moment of touchdown. Additionally, velocity is computed by the derivative of the tracking position data, which will exacerbate any noise. The velocity controller relies on these data and the implementation of the computed leg touchdown angle, and so inaccuracies will be reflected in the velocity control results.

Further to the general challenges of physical implementation, there are some ways the adaptive approach could create or amplify problems.

Noise and measurement error are inevitable for any real system, and will (almost) always degrade performance. However, this controller formulation can exacerbate these problems, because any error does not just affect the current step, but through the selection of gains will influence future steps too. This can generally be mitigated by choosing a larger $p$ value, effectively increasing the sample size used to compute the $\beta$ values. Although these results suggest this has not been the case, and the accuracy has not been significantly impacted in practice, suitable selection of the required $p$ value will be application specific.

The gain computation from Section 4.6 involves a matrix inversion, and when this matrix is close to singular even a small measurement error can create large errors in the selected gain values. Conceptually, this corresponds to cases where the preceding $p$ hops provide insufficient information to determine the best controller gains. For instance, if two consecutive hops have very similar velocity, then $\Delta \bar{x}' \approx 0$ and this provides little information for selecting $\beta_2$. Such issues may be mitigated through techniques to detect when the matrix inversion is poorly scaled, such as checking the condition number and not updating the $\beta$ values if this falls below a threshold.

## 4.9 Concluding Remarks

This Chapter serves to demonstrate the the core ideas behind the thesis, applied to the lossless SLIP model for two dimensional hopping. Firstly, that taking a simple empirical approach can be at least as accurate as even the more complex approximations available, while also having very low computational cost to allow for implementation on low-power hardware. Secondly, that this simple empirical approach can allow the controller to be tuned on-line, using a small number of preceding steps.

An advantage of the empirical approach is the avoidance of convoluted algebraic analysis required to achieve good accuracy, keeping the controller simple to derive, implement and tune. The multiple linear regression approach allows the values for the controller gains to be found quickly and easily. In addition, the empirical approach can still be intuitive (at least to some extent) by formulating the structure of the controller from consideration of the dynamics.

Another benefit of the simple formulation is the introduction of on-line tuning during use. This has the potential to reduce the manual effort in tuning an empirical controller allowing faster and easier implementation. It also allows the controller to adapt to changes in the environment, such as different ground types or changes in leg stiffness or body mass.

When moving from the simulated environment to physical implementation, the behaviour of the system will inevitably deviate from the idealised SLIP dynamics. As this happens, any analytical approximation is likely to only become more inaccurate, with correspondingly deteriorating performance. As the empirical controller can be tuned to the real system, rather than to the idealised model, it should be better able to cope with this mismatch.

The experimental results have demonstrated that this controller formulation can make the transition from simulation to physical implementation. The physical system inherently presents a more challenging control task, with the addition of sensor noise, additional non-linearity of the spring force, frictional effects to be overcome all contributing to errors and inconsistencies not found in the simulation. As a result, the $p$ value of the controller (the number of previous steps considered) had to be increased to smooth out some of these errors, but values between 10 and 30 will still allow for on-line adjustment to changing conditions.

As a final comment, the experimental work presented here reinforces that agile locomotion is an inherently difficult control problem in physical implementation, in ways not easily captured by simulation. A small angular error in the touchdown angle is often amplified in the resulting velocity, requiring a good measurement and actuation accuracy on the leg angle, while the limited time in flight requires the leg swing to be carried out quickly. From this perspective, the agile motion seen in nature is even more impressive.

# Chapter 5

# Combined Control of Hopping Height and Forward Velocity

The previous two chapters have described the importance of, and proposed controllers for, controlling the apex height and the forward velocity of a hopping robot. While an agile gait – where every step is different – can be achieved by accurately varying either of these parameters, that assumes that the other can be simultaneously maintained at a constant value without the need for it to be controlled. Clearly this will not in general be the case as the height of a hopping robot will be affected by energy losses, and the forward velocity generally to be actively controlled as it relied on dynamic stability. Therefore, controllers are required which can control both height and velocity at the same time.

In this chapter, such controllers are formulated and tested in simulation. The simplest approach is to take the controllers from the preceding chapters, and apply them at the same time under the *decoupled* controller assuming they do not affect one another. An improvement can be made by repeating some of the approach from previous chapters, but with some account of the new dynamics of the system by considering the energy transfer between velocity and apex height – refereed to as the *energy* controller. A controller generated by stepwise regression, which result in much greater algebraic complexity, demonstrates that a purely empirical controller is capable of accurately implementing a gait that is agile in both apex height and forward velocity, however both the decoupled controller and energy-based controller suffer degraded performance even if only one of the states (height or velocity) is given agile demand values. The adaptive method of gain updates successfully applied in previous chapters is found to exacerbate these problems, and in particular lead to increased likelihood of the robot falling, due poor gain choices caused by the underlying assumptions. These results have implications if the many controllers based on the lossless SLIP model are to be applied alongside a separate height controller.

## 5.1 Chapter Introduction

The preceding chapters have demonstrated the empirical approach is effective for controlling the height of a one-dimensional hopper, where there is no forward velocity, and for the control of the forward velocity in a lossless SLIP model, where the height does not need to be controlled. This chapter's aim is to combine these two controllers for height and forward velocity in a two-dimensional hopper and so explore, in simulation, the limits of the decoupled approach (where these two parameters are assumed to be independent). The ability to control both apex height and forward velocity in a an agile way will be important in future applications where a robot is required to traverse limited safe foothold positions which vary in both height and separation. If, for example, the addition of height variation has a large impact on the forward velocity controller, this would have serious implications for many controllers developed for the lossless SLIP model when attempting to apply them to this sort of task.

The first controller presented in this chapter will take the most straightforward, decoupled, approach and simply apply the previous controllers in parallel without any modification. This will be refined in the second controller (the "energy controller" below) by taking a similar approach to that which derived the original controllers, namely creating the overall form of the controller by reasoning about the dynamics (and in this case, the coupling effect between the two parameters), before tuning the gain values empirically. Finally, to show the potential power of an empirically tuned controller, a set of terms are chosen by stepwise regression to give a highly accurate, if somewhat more complex, formulation.

### 5.1.1 Chapter Contributions

This will be done by considering an extension to the SLIP model with the introduction of energy loss through a damping term, described in below in Section 5.2, where the task is to control the robot with variation between steps in the demanded values for both apex height and forward velocity. The three controller formulations are presented in section 5.3 and compared in simulation in section 5.4, before some concluding remarks in section 5.5.

Building on the previous ones, in the chapter:

- The proposed controllers from chapters 3 and 4 are further tested in combination.
- A further controller is proposed based on a similar strategy but considering the coupling effect of energy transfer between height and velocity, and the adaptive tuning method again applied.
- As a starting point for possible future work, the stepwise regression method is applied to generate a more accurate, purely empirical, controller.

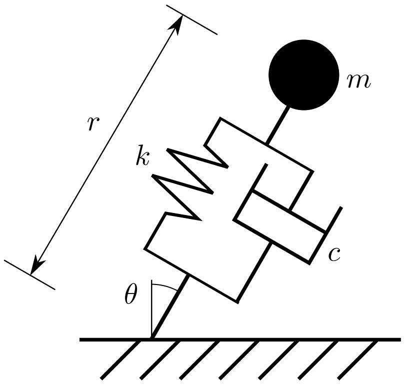


Figure 5-1: The modified stance phase model, with the addition of viscous damping in parallel with the leg spring.

## 5.2 Simulation Model

In order to necessitate an input to maintain height, a viscous damping term will be introduced into the stance phase of the SLIP model previously described in Section 4.2.3, forming the model shown in Fig. 5-1. Although it is still highly simplified, this “lossy SLIP” model adds some realism to the model in the form of an energy loss which must be replaced. The flight phase dynamics are unchanged from equation (4.1), while the stance phase dynamics (*c.f.* equation((4.2)) become:

$$\bar{r}'' = k^*(1-\bar{r}) - \cos(\bar{\theta}) + \bar{r}(\bar{\theta}')^2 - c^*\bar{r}' + F \tag{5.1a}$$

$$\bar{\theta}'' = \frac{1}{\bar{r}}\sin\bar{\theta} - \frac{2\bar{\theta}'\bar{r}'}{\bar{r}} \tag{5.1b}$$

where $F$ is a generic actuator force determined by the controller. In addition to the terms defined in section 4.2.4, a new term $c^*$ is the dimensionless viscous damping, defined in relation to the physical viscous damping coefficient $c$ as:

$$c^* = \frac{cs^2}{mr_0} \tag{5.2}$$

The system inputs are the touchdown angle, $\bar{\theta}^{td}$, as for the lossless SLIP model, and the force $F$ acting in the direction of the leg during the stance phase, but, by definition, there is no contact during flight and so no input during this phase. While the force input could be made to vary arbitrary over the stance phase, it will be restricted to a chosen value ($u_n$) applied while the leg is extending (*i.e.* the upward part of the stance phase) to give a single value for the input on a given step. Note that energy can be removed from the system by setting this input to a negative value.

The system parameters have been chose as $k^* = 10$ to match the simulations in chapter 4, and $c^* = 0.5$. The forward velocity values will again be varied from step to step

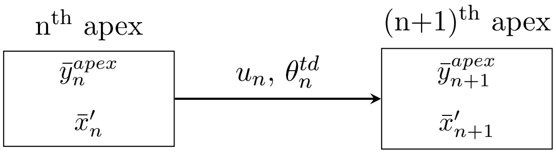


Figure 5-2: Each point in the dataset consists of finding the values of $u_n$ and $\theta_n^{td}$ which will cause the system to move from an apex state of $\bar{y}_n^{apex}$ and $\bar{x}'_n$ to the next apex state of $\bar{y}_{n+1}^{apex}$ and $\bar{x}'_{n+1}$.

between no forward motion to approximately a jogging pace, values $\bar{x}' = 0$ to $\bar{x}' = 1$. The apex height should always be greater than the leg length (*i.e.* greater than 1) to avoid the foot "clipping" the ground, so $\bar{y}^{apex} = 1.2$ to $\bar{y}^{apex} = 2$ has been chosen. The demanded motions will vary each step individually up to this range to create highly agile motion.

## 5.3 Empirical Controllers

### 5.3.1 Dataset Generation

First a dataset of "ground truth" values will be pre-computed, to be used for finding the fixed gain values (and the initial values for the adaptive controllers), and will also be used in developing the stepwise controller as described below. The controller inputs affect the system dynamics, and thus determine the next apex state from the current one, shown conceptually in Fig. 5-2. For each possible combination of current and next apex states, $\bar{y}_n^{apex}$, $\bar{x}'_n$, $\bar{y}_{n+1}^{apex}$ and $\bar{x}'_{n+1}$, the two system inputs, $u_n$ and $\theta_n^{td}$, must be found which will connect these states. This will be done through an iterative method of numerical simulations, each of a single step. Combinations will be considered of values from 1.2 to 2 for $\bar{y}^{apex}$ (in steps of 0.08) and 0 to 1 for $\bar{x}'$ (in steps of 0.1), giving a total of $11^4 = 14\,641$ data points.

Once generated, this dataset will be used to fit each of the empirical controllers described below to provide a good estimate for best gain parameters for overall performance over this range of system states.

### 5.3.2 Decoupled Controller

The decoupled approach involves simply applying the height controller for vertical hopping and the forward velocity controller for the SLIP dynamics from the previous chapters in parallel with one another. In doing this, it is assumed that the problem of controlling the height and that of controlling forward velocity can be considered separate (*i.e.* decoupled), so that the force input into the leg, $u$, affects only the apex height, $\bar{y}^{apex}$, and that the touchdown angle, $\theta^{td}$ affects only the forward velocity, $\bar{x}'$. The height controller from equation (3.11) and the velocity controller from equation (4.12) are combined to form:

$$u_n = \alpha_1 \left(\bar{y}_n^{apex} - 1\right) + \alpha_2 (\bar{y}_{n+1}^{apex} - \bar{y}_n^{apex}) \tag{5.3a}$$

$$\theta_n^{td} = \beta_1 \bar{x}'_n + \beta_2 \left(\bar{x}'^{tgt}_{n+1} - \bar{x}'_n\right) \tag{5.3b}$$

Note slight modifications from (3.11): firstly the addition of the overbar for $\bar{y}$ indicating the dimensionless quantities are now being used, and secondly $y_n^{apex}$ is replaced by $(\bar{y}_n^{apex} - 1)$. This is because the leg length could be ignored for the one dimensional model, and so was set to zero to simplify the equations. However, the leg's rest length is now defined as 1 and the zero energy level is defined as the point where the leg is at this length, vertical, and the foot in contact with the ground, while the height is measured from ground level.

### 5.3.3 Energy Controller

Intuition for the problem suggests the decoupled approach will fail for agile gaits, where there are significant changes in velocity from one apex to the next. This is because adjusting the touchdown angle changes the forward velocity not by adding (or removing) the required energy for this increase (or decrease) in velocity, but rather by transferring energy from gravitational into kinetic, causing a lower and faster (or higher and slower) next apex. It is therefore clear that controlling the velocity in this way will also have the effect of changing the value of the next apex height – the two parameters are coupled.

However, the observation that the touchdown angle should not significantly change the energy level[1] allows for another approach. Instead of the leg force input, $u_n$, being selected to control the required apex height, $y_{n+1}^{apex}$, it instead is selected to achieve the required overall next apex energy level, $\bar{E}_{n+1}$. This energy can then be split between forward velocity and height by the leg angle controller as required.

When the forward velocity controller was developed and tested in sections 4.4 and 4.5, the "simplified" version was selected which assumed a constant apex height. Now, with the apex height actively controlled and with a varying demand, this assumption is likely to no longer be valid and the variation in apex height should be included. The required addition to controller to return to the "full" version, derived from the approximation described in section 4.4, is to divide the second term, which acts on the required change in velocity, by $\sqrt{\bar{y}_n^{apex} - 1}$. This makes intuitive sense when thinking about the energy of the system. When the apex height is lower, there is less energy available to produce the desired change in velocity, and so a larger proportion of it is required to be converted, requiring a larger deviation in the touchdown angle.

---

[1] An effect may be created indirectly, whereby a different touchdown angle causes differing behaviour during the stance phase, which may in some way influence the amount of energy lost to the damping. Any such effect is likely to be small and is ignored for the purpose of formulating this controller.

Defining a new pair of controller gains $\gamma_1$ and $\gamma_2$ to describe the required input to maintain a particular energy level and create a particular change in energy level respectively, and re-introducing the velocity controller from equation (4.12b), the controller takes a very similar form the those previously described:

$$u_n = \gamma_1 \bar{E}_n + \gamma_2 \left( \bar{E}^{tgt}_{n+1} - \bar{E}_n \right) \tag{5.4a}$$

$$\theta^{td}_n = \beta_1 \bar{x}'_n + \beta_{2B} \left( \frac{\bar{x}'^{tgt}_{n+1} - \bar{x}'_n}{\sqrt{\bar{y}^{apex}_n - 1}} \right) \tag{5.4b}$$

with the important new term to describe the (non-dimensional) energy level at the $n^{th}$ apex, defined as:

$$\bar{E}_n = \bar{y}^{apex}_n - 1 + \frac{1}{2} \left( \bar{x}'_n \right)^2 \tag{5.5}$$

This formulation moves beyond the decoupling assumption, as the energy (and so now choice of $u_n$) depends on both the apex height and forward velocity, but retains some of the advantage of the previous controller, without the value of either control input being dependent on the other. It also retains a form based on simple approximations and reasoning about the dynamics, assisting with an intuitive understanding of how it functions. It can be thought of as the leg actuation modulating the total energy of the system, while the touchdown angle determines how this energy is split between height and velocity in the next flight phase.

The previously described adaptive approach through multiple linear regression based on previous steps can also still be applied and will be tested below.

### 5.3.4 Term Selection by Stepwise Regression

The previous controllers have been formulated by first considering the physics of the model, and from this determining a simple relationship, to be empirically tuned. If, instead, the physics is not considered, and completely empirical approach may be taken, it may be possible to find a better set of algebraic terms. Such a controller would lose some appeal due to the lack of intuition for “how” it works, but could potentially be more accurate. The self-tuning approach may still be taken, so long as the controller is restricted to linear coefficients which can then be found through the regression method discussed in previous chapters, although the more terms are included the less practical this becomes.

Stepwise regression is a technique to automate the selection of parameters to be used in a linear model [78, p.307-312]. To create the model, first a wide range of candidate terms will be generated. From these, the stepwise regression algorithm will add and remove individual terms and determine those which have a significant impact on the accuracy of the overall model, finishing with a linear combination of terms that can accurately predict the values in the dataset generated above.

The freedom of the simulation environment allows for a wide range of candidate terms to be provided. Any possible controller will be considered to be made up of a sum of terms, where each term is the product of the four values which define the current ($n^{th}$) and next ($(n+1)^{th}$) apex states, each raised to a power ($i$,$j$,$k$ and $l$ respectively) and with a unique coefficient, $\lambda$ or $\kappa$:

$$u_n = \sum \lambda_{ijkl} \left(\bar{x}'_n\right)^i \left(\bar{x}'_{n+1}\right)^j \left(\bar{y}^{apex}_n - 1\right)^k \left(\bar{y}^{apex}_{n+1} - 1\right)^l \tag{5.6}$$

$$\theta^{td}_n = \sum \kappa_{ijkl} \left(\bar{x}'_n\right)^i \left(\bar{x}'_{n+1}\right)^j \left(\bar{y}^{apex}_n - 1\right)^k \left(\bar{y}^{apex}_{n+1} - 1\right)^l \tag{5.7}$$

Candidate terms will be considered for all possible combinations of the powers in the following ranges, giving a total of 255 candidates in each case.

$$\begin{aligned} i &\in \{0, 1, 2\} \\ j &\in \{0, 1, 2\} \\ k &\in \{-1, -0.5, 0, 0.5, 1\} \\ l &\in \{-1, -0.5, 0, 0.5, 1\} \end{aligned}$$

Note that for all terms containing $(\bar{x}')^2$, this is replaced by $|\bar{x}'|\,\bar{x}'$ in order preserve the sign of the velocity and ensure the controller is agnostic to direction of travel. This allows it to perform equivalently for positive or negative $\bar{x}'$ values.

This gives a a controller formulation where $u_n$ and $\theta^{td}_n$ are each computed by a total of 225 candidate terms, where each term has a unique coefficient denoted by $\lambda$ or $\kappa$ (including where all the powers are zero, giving a constant term). When the stepwise regression method was applied[2] to the dataset described above 49 $\lambda$ terms for selecting $u$, and 56 $\kappa$ terms for selecting $\theta^{td}$, were included as significant in the final model.

## 5.4 Simulation Results

Each of the controllers described in Sections 5.3.2–5.3.4 above has been implemented for comparison in simulation, with the decoupled and energy controllers tested using fixed gain values and adapting over $p = 10$ preceding steps under the scheme already described in detail for the height and velocity components of the controller in sections 3.3.2 and 4.6. Various ranges of demand in velocity and apex height will be tested, *i.e.* requiring different levels of agility, with larger changes in demand from step to step being more challenging to achieve. Results of some example runs are shown in Figs. 5-3 to 5-6. The baseline result, against a steady state demand, is shown in Fig. 5-3 with all the controllers able to achieve this, with a small steady state error for the fixed gain decoupled and energy controllers. In each particular run, variation can be introduced in either the height or forward velocity demand values, with the other set at a steady state demand, or both can be varied at the same time. Variation in either of demand values could introduce errors in the resulting velocities, apex heights, or both.

---

[2] This was done with the MATLAB function *stepwisefit()*, using confidence values to add or remove terms of 0.05

It can be seen in Fig. 5-4 that the decoupled controller performs poorly when the forward velocity is required to change in an agile manner, even for a constant height demand, with the fixed gain version less accurate than the energy controller and the adaptive version unstable. As forward velocity is changed, this removes or adds energy to the height but is not compensated for under the decoupled assumption of the controller. This problem is exacerbated by using the adaptive gains: when the apex height fails to meet the demand, the gains are updated under the assumption this was caused by the height controller, when if fact it was caused by the velocity controller through changing the touchdown angle. This incorrect assumption causes poor gain values to be chosen and the hopper often falls, in this case between steps 10 and 15.

Fig. 5-5 shows the inverse, attempting to maintain a steady velocity while the apex height demand is agile. Although there are some differences between the decoupled and energy controllers, neither shows a clearly improved performance over the other. As before, the adaptive controllers perform noticeably worse achieving a steady-state demand (in this case velocity) compared with the fixed gain equivalents because of the deceptive feedback they receive, in this case by different leg actuations intended to modify the apex height also affecting forward velocity.

Finally, Fig. 5-6 shows a run with changes in both forward velocity and apex height demands. The controllers are all able to track these agile gaits, albeit with some error. Indeed the adaptive decoupled controller which had previously been unstable with a steady state apex height demand performs better now that the demands are changed between steps since these changes yield better information for tuning the controller gains.

The results described so far are some example runs from a larger set, from which averages can be taken to confirm these observations hold more generally. For each dimension (velocity and apex height), the demands have been varied randomly over either the full range used in the dataset generation (maximum agility required), a constant in the centre of this range (steady state), or a range half way between these two. These different demand ranges make up the rows and columns in Fig. 5-7, labelled along the left and bottom, where, for example, $\bar{x}'_{tgt} = 0$ to 1 indicates velocity targets varying randomly between zero and 1 for each step, which would give a maximum possible demanded change of 1. Each controller, for each pair of target ranges, was tested over 1000 runs of 100 steps in each run, and the absolute error (the magnitude of the difference between the achieved and the target values) on each step was recorded.

Results are shown in Fig. 5-7 for (a) the velocity errors and (b) the apex height errors.

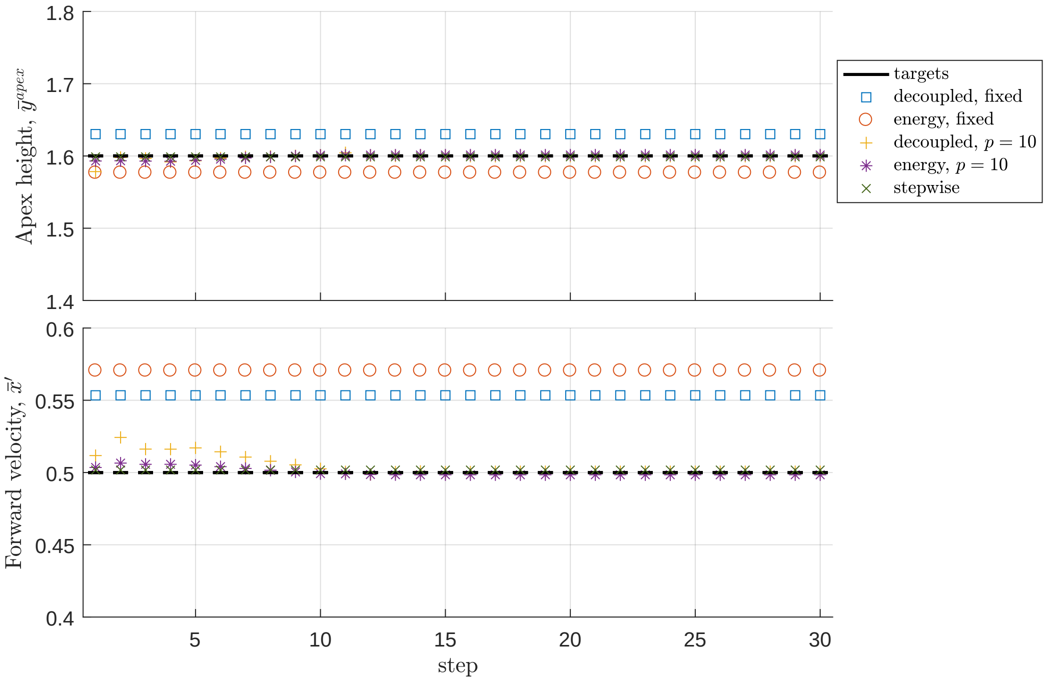


Figure 5-3: An example run with the demand values for both apex height and forward velocity head constant, a steady state gait.

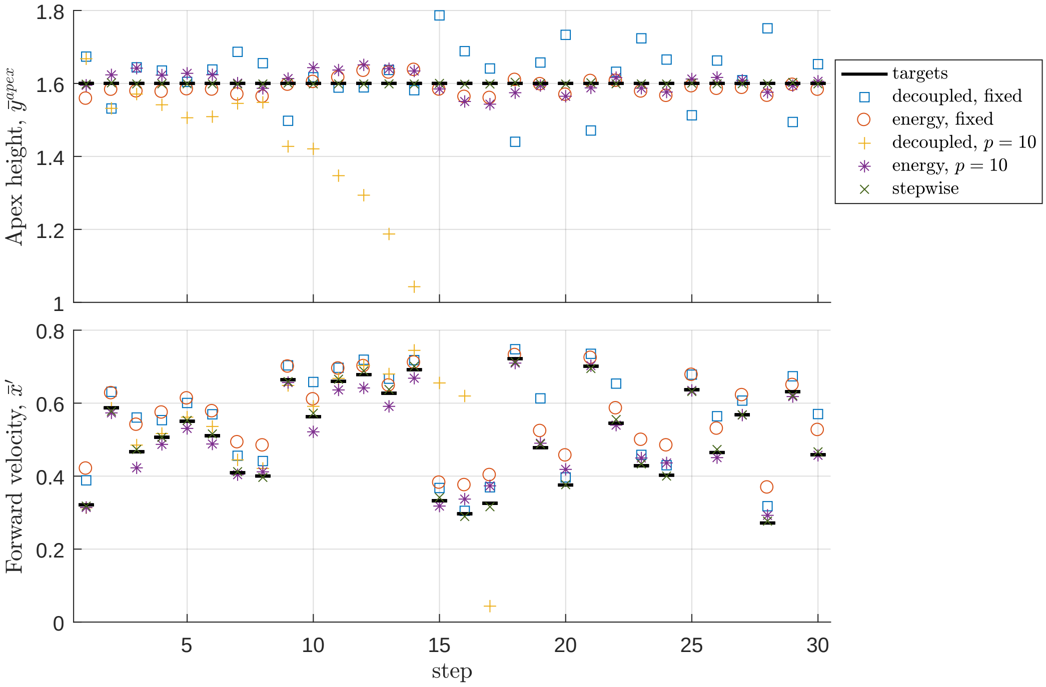


Figure 5-4: An example run where the forward velocity target is varied, while the apex height target is constant, for each controller.

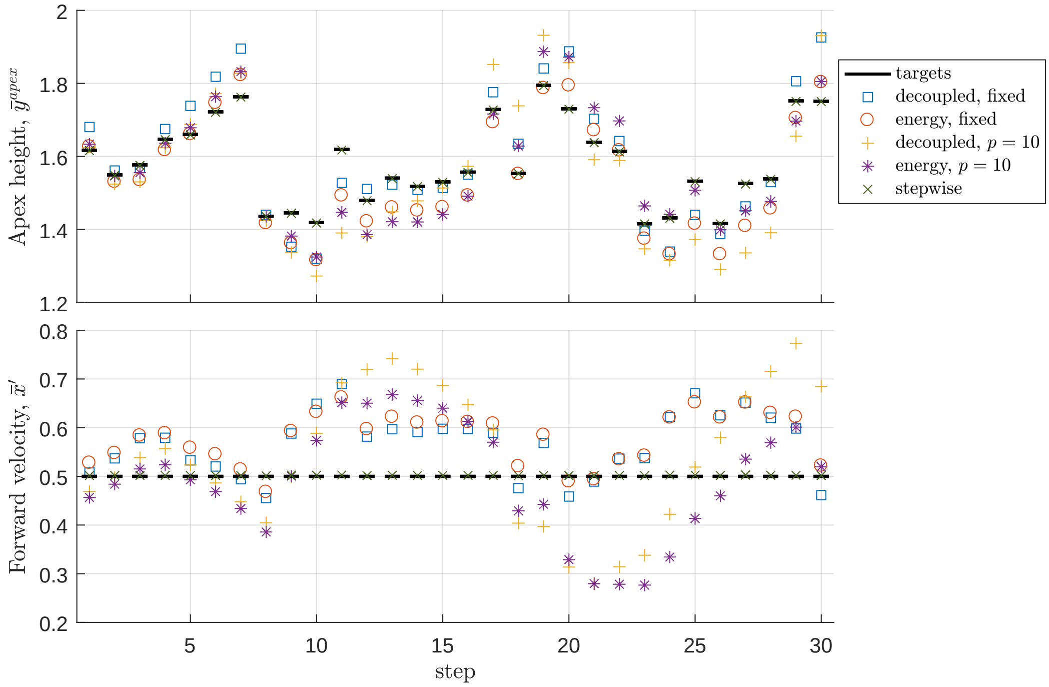


Figure 5-5: An example run where the forward velocity target held constant, while the apex height target is varied, for each controller.

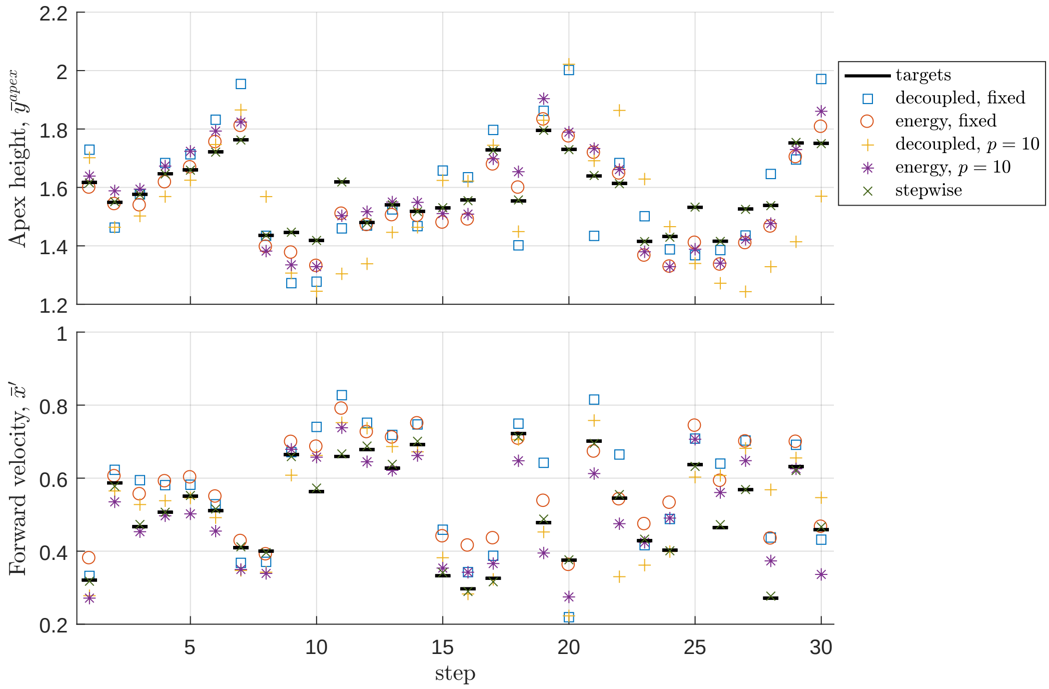


Figure 5-6: An example run of varying both forward velocity and apex height demand values, for each controller.

Shown are the median and quartiles of the absolute error for all the steps completed. For those cases where not every run was completed successfully the "stability score" (the mean steps to failure with maximum of 100) is printed above the bars – these values are very important because any controller that causes the robot to fall is likely to be even less useful than one that is inaccurate but at least remains upright. With each subplot, there are nine sets of target ranges, where top left is steady state demand (the same target every step), and the bottom right most agile with large target variations possible between each step.

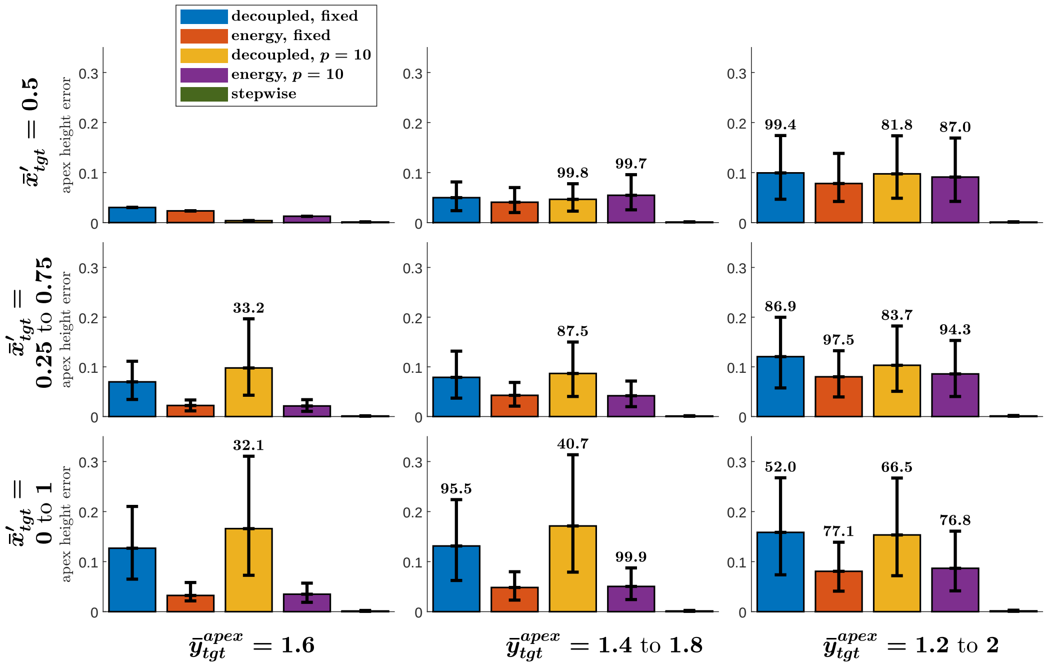
Within the data in Fig. 5-7 there are several notable results and patterns that warrant discussion.

For the case of steady state demand (top left subplot in each (a) and (b)), the error is generally the lowest. In particular, the adaptive controllers ($p = 10$, yellow and purple bars) rapidly tune the relevant parameters to the best values for this particular velocity/height combination, bringing the average error to almost zero.

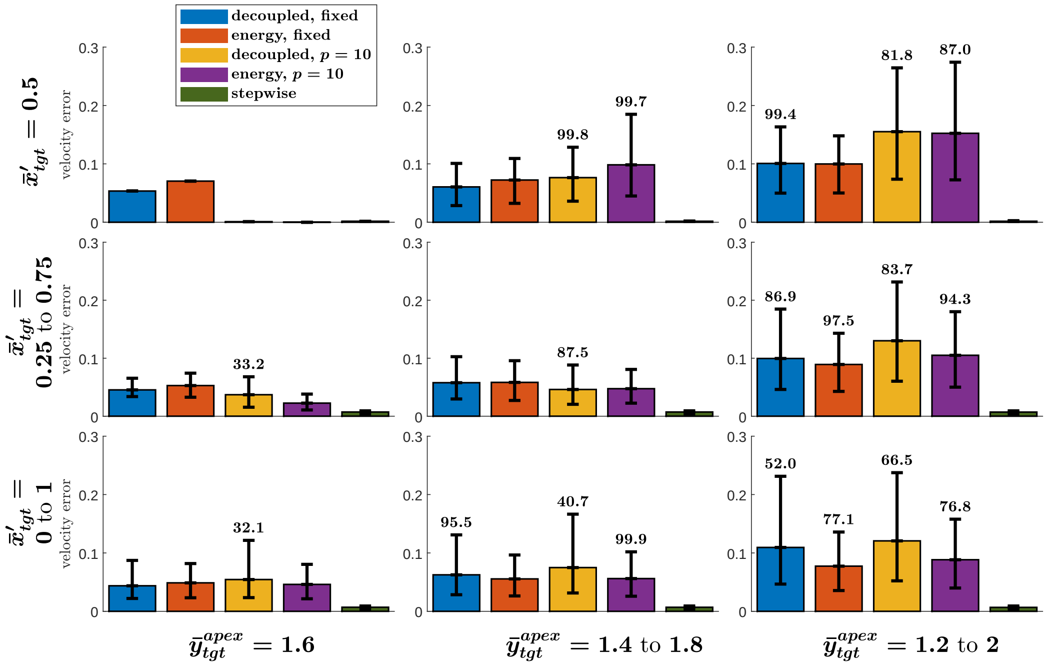
The decoupled and energy controllers (blue/yellow bars and orange/purple bars) have very similar performance in terms of horizontal velocity error (Fig. 5-7(b)), whereas the error in apex height control (Fig. 5-7(a)) shows the energy controller performing better. As the demanded forward velocity is made more agile (moving from top row to bottom row), the energy controller is much better able to compensate, even if the apex height is demanded to be constant (left column). This indicates that the coupling effect of transferring energy between kinetic (velocity) and gravitational (apex height) considered by the energy controller formulation is indeed an important factor.

The introduction of self-tuning for the decoupled controller (compare the blue to the yellow bars) reduces the stability significantly (indicated by lower values of the stability score, printed above each bar when less than 100). This is in contrast to the results from the equivalent controller used on the lossless version of the model in section 4.7, where a value of $p = 10$ had very little detrimental impact on the stability score over the fixed gains. The self-tuning method assumes that the controller parameters can be selected based on previous steps, *i.e.* whatever parameters would have worked for the preceding steps should be useful for this step. In cases where the information from the preceding steps is not applicable, this will lead to erroneous parameters being selected; this can amplify the errors introduced when the decoupling assumption fails to take into account the effect of changes in apex height will have on the forward velocity and vice versa. For instance, if a large increase in velocity on previous step has caused the apex height to drop on that step, the self-tuning would conclude that the $\alpha$ gains for selecting the input, $u$, should have been larger, and so apply a higher input values (compared to the fixed controller), even if the next step requires instead a decrease in velocity (which will add to the apex height).

The controller based on the stepwise regression model dramatically outperforms the alternatives in Fig. 5-7. The variation in demand values makes much less difference than for other controllers, and with errors of less that 1% of the demanded values even for the agile gait demands. This shows the potential power of fitting larger numbers of terms;



(a) Median absolute error in apex height, $\bar{y}^{apex}$



(b) Median absolute error in forward velocity, $\bar{x}'$

Figure 5-7: Simulation results for each controller with demand values varied randomly within the stated ranges.

in particular, this controller is free from the assumptions of linear relationships, and the “decoupling” of touchdown angle controlling velocity and leg force input controlling height or energy level.

## 5.5 Concluding Remarks

### 5.5.1 The Decoupling Assumption

In this chapter, the the controllers previously introduced for hop height (Chapter 3) and forward velocity (Chapter 4) are combined. The two controllers are applied in parallel to a version of the SLIP model, extended to include some energy losses which introduces the need for actively controlling the apex height. When applied independently, the assumption being made is that for determining the required leg actuation, only the apex height needs to be considered, and for the touchdown angle, only the forward velocity The resulting decoupled controller has been shown to be adequate when the demand vales are constant. However, for a gait that is agile (with variation in demand on every step) in even one of the two states creates errors in both. The fact that introducing more variation in apex height induces an error in the velocity, and vice versa, suggests this decoupling does not in fact hold.

The performance of the controller can be improved by removing this assumption, allowing the controller for each input to consider both parameters of the apex state (the height and velocity). The “energy” controller remains in keeping with the approaches from previous chapters, where the controller is formulated based on approximations to the dynamics, but tuned based on completed steps.

In this way, the energy controller successfully reduced the apex height error. However, the remaining error in forward velocity suggests there is a further coupling effect that has not been accounted for, whereby the leg actuation force impacts not just the overall energy level but also its division between forward velocity and height. This effect will also be present for the many controllers based around the lossless SLIP dynamics, implying that subsequently adding a height controller may be detrimental to their efficacy, especially for agile gaits.

### 5.5.2 Utility of Stepwise Regression Controller

The terms used in the previously discussed controllers was determined through reasoning about, and approximations to, the dynamics of the system, with only the parameters tuned empirically. By relaxing the desire for this kind of intuitive reasoning, an empirical approach can also be taken to choosing the terms themselves. As illustrated by the controller created through stepwise regression method, this can provide greatly improved accuracy.

The approach to controller formulation for a physical robot would need to be evaluated on a case by case basis; if a robot conforms closely to the SLIP model, the previously described controllers may be suitable. For other robots, for instance with more complex leg geometry, other simple relationships may be derived by reasoning about their dynamics. However, methods such as the stepwise regression used here, or more complex machine learning systems, could become useful for complex robots; if the terms of the controller can be kept linearly independent (*i.e.* a linear sum of terms each with their own coefficient), tuning and on-line adaptation of the coefficients could still make use of the multiple linear regression method from previous chapters.

The controller selected by stepwise regression in this chapter includes over 100 terms, and is something of an extreme example whereby the simulated dataset is fitted very closely by the empirical controller. Replicating such a large dataset for a physical robot for the selection of these terms and tuning the coefficients is likely to be impractical. Instead, which terms to be included could be selected through a large set of simulation data, as was done here, but with with the actual $\lambda$ and $\kappa$ coefficients tuned based on experimental data, possibly on-line. The additional noise and experimental error in a physical robot means the number of sample steps must be substantially larger than the number of parameters to be selected, and so the more terms included in the controller the longer it will take to tune. For on-line tuning, this corresponds to a high $p$ value, so it would be slow to respond to changes. Even so, this could still be useful for responding to slowly changing parameters, for instance a change in actuator characteristics as the battery runs down in an electrically actuated robot.

Therefore, although the stepwise controller as formulated here is unlikely to be used in practice, the overall technique could be. It would be appealing to always formulate the controller based on knowledge of the robot's dynamics, but if this is not possible or found to perform poorly, a more purely empirical approach could be explored.

# Chapter 6

# Conclusions and Future Work

## 6.1 Research Question

This thesis has explored agile and dynamic legged locomotion for robots, which is challenging from both a mechanical and control perspective. The research question originally posed in the introduction was: **“can a discrete feedback controller, derived from approximate dynamic models but tuned on-line, produce accurate hoping control of a small pneumatic robot”**.

This question has been explored through a combination of simulation and physical experiments, with hopping control subdivided into control over the height of each apex and the velocity of the horizontal motion. The proposed controller has been successfully demonstrated as effective in the one dimensional case of height control, including physical implementation with a pneumatically actuated leg. For the velocity control, the empirical controller was found to outperform the available analytical SLIP hopper controllers in simulation, with some physical verification of its efficacy. Less clear is the ability of the height and velocity controllers to be combined, for cases where both the height and velocity must be varied from step to step. While the controllers could be applied to steady state demands, the accuracy and stability deteriorated for agile gait, with a much more complex empirical controller needed even in simulation.

## 6.2 Findings

The analytical, simulation and experimental work described in this thesis has lead to numerous findings.

In the literature, it was found that the Spring Loaded Inverted Pendulum (SLIP) is the dominant model for dynamic hopping and running, with previous approaches to control mostly split between strategies reliant on numerical simulations or approximated analytical solutions, and focused largely on steady state gaits. Thus, the research gap was identified for a controller using a simple empirical formulation, focussed on agile hopping.

The single degree of freedom hopper of Chapter 3 found that a simple linearised controller can be highly effective, even for the pneumatic system which presents non-linear spring and actuation properties. An adaptive system of gains was shown to be able to eliminate steady state errors and track an agile demand, as well as cope with changes in the external system parameters, requiring no direct sensing of such changes other than the resulting effect on the system apex state. Experimental validation of the simulations showed that the proposed system can overcome the measurement error and noise of a physical system.

Chapter 4 looked at the control of the SLIP model. It was found that a two-term empirical controller, if well tuned, could outperform the approximations found in the literature for agile gaits on a simulated SLIP hopper. The adaptive system of gains was found to be able to successfully predict the required gains for future steps, based on the previous control outputs and apex states. While the gains of the controller can be automatically selected, a key choice remains for the system designer: that of the $p$ value, representing the number of previous steps to be considered. A low $p$ value allows the system to rapidly update to changes in the system or environment, while larger $p$ values have the advantage of greater system stability especially in the face of noise. In simulation, $p$ values could be reduced to the theoretical minimum of $p = 2$ for slow, steady gaits, but for the physical test hopper this had to be increased to at least $p = 10$.

It has been found, in Chapter 5, that the empirical controllers for height and forward velocity from previous chapters can be applied simultaneously, but with sever limitations. Simply assuming the two controllers are independent, so that leg force affects only apex height and touchdown angle only forward velocity, in the decoupled approach was successful for steady state gaits but, for agile gaits the errors were found to be significantly larger than when each had been used in isolation. Furthermore, the adaptive gain system was found to lead to the hopper falling in many cases where either the height or velocity demand was varied. An important reason for this was found to be that the transfer of energy between speed and height was not being accounted for, and by reformulating the controller in terms of energy, instead of height, improved the stability and to some extent the accuracy. Despite this, the controller still did not perform nearly as well as a fully empirical controller, with many terms to account for the coupled dynamics, suggesting there could be a better formulation available.

Overall, these findings have added to the body of work toward legged robotic systems able to traverse complex terrain through agile gaits, by taking a novel control approach. In particular, the results have shown that there is some promise in this more empirical approach, where the controller gains can be set based on a small number of preceding steps, an approach not seen in previous work. This work has contributed to the body of knowledge through the controller formulation, simulations of fundamental models and physical experiments across one and two dimensions.

## 6.3 Closing Comments

The fundamental approach taken throughout this thesis has been to formulate a controller based on simple analytical approximation, but to tune the appropriate gain values by using the results of previous steps. This retains much of the benefit of the analytical approximations over a fully empirical method, in terms of allowing some intuitive reasoning about how the controller is functioning. This contrasts to many “black box” algorithms popular in machine learning, where there is little understanding of how the controller functions and therefore of why and in what circumstances it might fail. Meanwhile by tuning the parameters, performance can be improved by accepting that the approximation cannot faithfully reproduce the full dynamics, especially for a physical robot. It is therefore argued that this kind of approach can provide reasoned but pragmatic solutions to difficult problems in robot control, such as legged locomotion.

Another key feature has been an emphasis on minimising computational cost for the deployed controller. With ever increasing power and reducing cost of computation, it may be tempting to simply accept the computational cost of an approach based on on-line simulation; however in cases where a simpler alternative is available, as well as being more elegant, there are practical advantages. A more lightweight controller allows for smaller, cheaper control hardware and less energy usage (therefore longer battery life), and/or frees up resources for other computationally demanding tasks such as vision or high level planning.

This approach offers the chance to combine the benefits of other methods, having been shown to be more accurate than the available analytical approximations, but is inherently simpler to compute than approaches using on-line numerical simulation, such as model-predictive control.

Even if it does prove to be the case that a more complex controller is required to achieve the desired accuracy, the simple model used here could still find application within the overall controller architecture. For instance, a robot may need to asses the difficulty of multiple potential paths, and in order to do so will need to determine, at least approximately, the touchdown angles required for each path, over many steps. This planning could make use of the fast to compute empirical controller for comparing the options available, before a more complex but accurate controller refines the touchdown angles to use for the chosen path. If only the computationally expensive method is available, such planning problem could rapidly become infeasible.

This thesis has focused on the control of agile motion, allowing for changes in velocity or apex height in a single step. This is in contrast to majority of previous work, which, with a few notable exceptions, have focused on approaching a demand over several steps and maintaining it, perhaps against disturbances. Such steady state gaits are undoubtedly useful to maximise efficiency and/or stability when running over flat terrain, where the precise location of each foot landing is of little importance. However, more complex terrain will require control over foot placement – consider crossing a set of unevenly spaced stepping stones – and so the height and velocity of each step must be adjusted. As robots move into more unstructured environments in the real world, this kind of control will become ever more important.

## 6.4 Future Work

The work from this thesis presents several possible directions to continue the study of robotic locomotion.

The results from Chapter 5 warrant further investigation of the coupling effect between height and velocity control, which is a largely neglected area. The neat-perfect tracking from the complex stepwise-based controller suggests a trade-off between controller complexity and performance. This could be further explored by introducing new terms one at a time to find the most important, and create a controller with good performance but few enough terms to allow for the adaptive gains scheme to be used.

A more capable physical robot platform would be needed to test the combined height and velocity controllers in hardware, which would be an important next step. The friction and lack of fidelity in the actuation has been the limiting factor in preventing this from being achieved thus far. While the pneumatic actuator used was convenient and sufficient for basic height control, a lower friction version would allow for larger hops, making the errors less significant as a proportion. The leg angle actuator may benefit from the addition of an encoder to provide faster position feedback and larder actuation servos to allow for more accurate leg positioning during the flight phase.

While the simulation work here has been focussed on simple models in order to illuminate the fundamentals of the controller's function, another question which could be approached may be how well it may generalise to other leg geometries. As a first implementation, various non-linear leg springs could be implemented and compared. More complex simulation models could be developed to include different leg geometries such as using a knee joint, or the pneumatic model using the the one dimensional simulation could be embedded into the leg of the SLIP hopper. The inclusion of body inertia and hip torque adds another degree of freedom and actuation dimension which must also be controlled, which could be attempted by an extension of the proposed controller method. In this way, controllers could be tested in more detail in the simulation environment, across a wider range of conditions than can reasonably be implemented physically.

While the control of apex height and forward velocity, as investigated here, is an important element of locomotion control, in general it would not be an end in and of itself but rather a tool to achieve foot landing position targeting to allow the traversal of complex terrain. This will add a layer of complexity to the implementation which may have unforeseen consequences. Therefore, the velocity and height controllers should also be integrated into a higher level controller for foot placement, to apply the agile motion to traversing footholds which vary in both horizontal separation and vertical height. This is clearly an important next step toward traversing complex terrain, and will then need to be further combined with perception capabilities for locating the safe footholds and planning for determining the sequence in which to use them.

More generally, there are many challenges remaining for the practical application of these types of controllers, especially in moving from simulation to physical robots. Making this transition may be area where an adaptive controller has a particular advantage, since it can adapt to the system as it is, rather than as it was simulated. This makes for an interesting area of potential study, such as by introducing additional noise or errors into a simulation, or deliberately poorly tuning the controller to replicate the possible difference when applied in practice.

For hopping robots with SLIP-like dynamics, such as considered in this thesis, small errors in touchdown angle can lead to large errors in the resulting forward velocity. This makes precise sensing and actuation, which must occur in the limited time of the flight phase, a critical but difficult task, all too easily overlooked by simulation studies. Space and hardware limitations have restricted this study to a small robot with a relatively short hopping track. The next stages of development should see larger and more unconstrained test robots, able to perform longer runs and better test longer-term stability, and adaptation to changes, for example hopping between different ground surfaces or carrying various payloads. Moving out of the laboratory and into the field, the challenges of sensing will become even more difficult, and work needs to be done to develop internal state estimation, without relying on external tracking cameras.

Overall, the results of this thesis show potential for a simple, fast to compute, method for controlling the complex task of agile legged locomotion, deserving of further investigation.

# References

[1] R. Blickhan and R. J. Full, "Similarity in multilegged locomotion: Bouncing like a monopode," *Journal of Comparative Physiology A*, vol. 173, no. 5, pp. 509–517, 1993.

[2] M. H. Raibert and J. H. B. Brown, "Experiments in Balance With a 2D One-Legged Hopping Machine," *Journal of Dynamic Systems, Measurement, and Control*, vol. 106, no. 1, pp. 75–81, 1984.

[3] H. Geyer, R. Blickhan, and A. Seyfarth, "Natural dynamics of spring-like running: Emergence of selfstability," *International Conference on Climbing and Walking Robots (CLAWAR)*, pp. 1–6, 2002.

[4] S. Hyon and T. Emura, "Quasi-periodic gaits of passive one-legged hopper," *IEEE International Conference on Intelligent Robots and Systems*, no. October, pp. 2625–2630, 2002.

[5] A. Degani, A. W. Long, S. Feng, H. Benjamin Brown, R. D. Gregg, H. Choset, M. T. Mason, and K. M. Lynch, "Design and open-loop control of the parkourbot, a dynamic climbing robot," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 705–718, 2014.

[6] N. Shemer and A. Degani, "Analytical control parameters of the swing leg retraction method using an instantaneous SLIP model," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 4065–4070, 2014.

[7] C. François and C. Samson, "A New Approach to the Control of the Planar One-Legged Hopper," *The International Journal of Robotics Research*, vol. 17, no. 11, pp. 1150–1166, 1998.

[8] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek, "A Simply Stabilized Running Model," *SIAM Journal on Applied Dynamical Systems*, vol. 2, no. 2, pp. 187–218, 2003.

[9] R. B. McGhee and A. A. Frank, "On the stability properties of quadruped creeping gaits," *Mathematical Biosciences*, vol. 3, no. 1-2, pp. 331–351, 1968.

[10] K. J. Waldron and R. B. McGhee, "The Adaptive Suspension Vehicle," *IEEE Control Systems Magazine*, vol. 6, no. 6, pp. 7–12, 1986.

[11] D. Belter and P. Skrzypczyński, "Integrated motion planning for a hexapod robot walking on rough terrain," *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 44, no. 1 PART 1, pp. 6918–6923, 2011.

[12] R. Hoggett, "Cybernetic zoo [online]." `http://cyberneticzoo.com`, last accessed 2019-09-19.

[13] E. Garcia, M. A. Jimenez, P. G. De Santos, and M. Armada, "The evolution of robotics research," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 90–103, 2007.

[14] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka, "The Development of Honda Humanoid Robot," in *IEEE Conference on Robotics and Automation*, pp. 1321–1326, 1998.

[15] J. Pratt, "Virtual Model Control: An Intuitive Approach for Bipedal Locomotion," *The International Journal of Robotics Research*, vol. 20, no. 2, pp. 129–143, 2001.

[16] G. Pratt, "Legged robots at MIT: what's new since Raibert," *IEEE Robotics & Automation Magazine*, no. September, pp. 16–19, 2000.

[17] R. P. Marc Raibert, Kevin Blankespoor, Gabriel Nelson, "BigDog, the Rough-Terrain Quadruped Robot," *OCL - Oleagineux Corps Gras Lipides*, vol. 13, no. 2-3, pp. 143–151, 2006.

[18] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," *IEEE-RAS International Conference on Humanoid Robots*, pp. 120–127, 2014.

[19] Boston Dynamics, "Spotmini autonomous navigation [online]," 2018. `https://www.youtube.com/watch?v=Ve9kWX_KXus`, last accessed 2019-09-04.

[20] R. Blickhan, "The spring-mass model for running and hopping," *Journal of Biomechanics*, vol. 22, no. 11-12, pp. 1217–1227, 1989.

[21] M. Ahmadi and M. Buehler, "Stable control of a simulated one-legged running robot with hip and leg compliance," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 1, pp. 96–104, 1997.

[22] J. Schmitt, "A simple stabilizing control for sagittal plane locomotion," *Journal of Computational and Nonlinear Dynamics*, vol. 1, no. 4, pp. 348–357, 2006.

[23] 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.

[24] J. Schmitt and J. Clark, "Modeling posture-dependent leg actuation in sagittal plane locomotion," *Bioinspiration and Biomimetics*, vol. 4, no. 4, 2009.

[25] 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*, vol. 19, no. 2, 2009.

[26] I. Poulakakis and J. W. 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.

[27] S. H. Tamaddoni, F. Jafari, A. Meghdari, and S. Sohrabpour, "Biped hopping control bazsed on spring loaded inverted pendulum model," *International Journal of Humanoid Robotics*, vol. 7, no. 2, pp. 263–280, 2010.

[28] J. D. Karssen and M. Wisse, "Running with improved disturbance rejection by using non-linear leg springs," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1585–1595, 2011.

[29] J. G. D. Karssen, M. Haberland, M. Wisse, and S. Kim, "The optimal swing-leg retraction rate for running," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4000–4006, 2011.

[30] I. Uyanik, U. Saranli, and O. Morgül, "Adaptive control of a spring-mass hopper," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2138–2143, 2011.

[31] H. R. Vejdani and J. W. Hurst, "Swing leg control for actuated spring-mass robots," *Adaptive Mobile Robotics - Proceedings of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, CLAWAR 2012*, pp. 536–542, 2012.

[32] H. Yu, M. Li, W. Guo, and H. Cai, "Stance control of the SLIP hopper with adjustable stiffness of leg spring," in *2012 IEEE International Conference on Mechatronics and Automation, ICMA 2012*, pp. 2007–2012, 2012.

[33] G. Piovan and K. Byl, "Enforced symmetry of the stance phase for the Spring-Loaded Inverted Pendulum," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1908–1914, 2012.

[34] O. Arslan and U. Saranli, "Reactive Planning and Control of Planar Spring-Mass Running on Rough Terrain," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 567–579, 2012.

[35] M. Rutschmann, B. Satzinger, M. Byl, and K. Byl, "Nonlinear model predictive control for rough-terrain robot hopping," in *IEEE International Conference on Intelligent Robots and Systems*, pp. 1859–1864, 2012.

[36] K. Byl, M. Byl, M. Rutschmann, B. Satzinger, L. Van Blarigan, G. Piovan, and J. Cortell, "Series-elastic actuation prototype for rough terrain hopping," in *2012 IEEE Conference on Technologies for Practical Robot Applications, TePRA 2012*, pp. 103–110, 2012.

[37] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3D-SLIP model," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5134–5140, 2013.

[38] A. Wu and H. Geyer, "The 3-D spring-mass model reveals a time-based deadbeat control for highly robust running and steering in uncertain environments," *IEEE Transactions on Robotics*, vol. 29, no. 5, pp. 1114–1124, 2013.

[39] H. Yu, M. Li, and H. Cai, "Analysis on the performance of the SLIP runner with nonlinear spring leg," *Chinese Journal of Mechanical Engineering*, vol. 26, no. 5, pp. 892–899, 2013.

[40] G. Piovan and K. Byl, "Two-element control for the active SLIP model," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5656–5662, 2013.

[41] L. Palmer and C. E. Eaton, "Periodic springmass running over uneven terrain through feedforward control of landing conditions," *Bioinspiration & Biomimetics*, vol. 9, no. 3, p. 036018, 2014.

[42] B. Dadashzadeh, H. R. Vejdani, and J. Hurst, "From template to anchor: A novel control strategy for spring-mass running of bipedal robots," *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, no. Iros, pp. 2566–2571, 2014.

[43] G. Piovan and K. Byl, "Reachability-based Control for the Active SLIP Model," *International Journal of Robotics Research*, vol. 34 (3), pp. 270–287, 2015.

[44] G. Piovan and K. Byl, "Approximation and Control of the SLIP Model Dynamics via Partial Feedback Linearization and Two-Element Leg Actuation Strategy," *IEEE Trans. on Robotics*, vol. 32, no. 2, pp. 399–412, 2016.

[45] M. Calisti, E. Falotico, and C. Laschi, "Hopping on Uneven Terrains with an Underwater One-Legged Robot," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 461–468, 2016.

[46] N. Shemer and A. Degani, "A flight-phase terrain following control strategy for stable and robust hopping of a one-legged robot under large terrain variations," *Bioinspiration and Biomimetics*, vol. 12, no. 4, 2017.

[47] G. Secer and U. Saranli, "Control of planar spring-mass running through virtual tuning of radial leg damping," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1370–1383, 2018.

[48] M. H. Raibert, *Legged Robots That Balance.* MIT Press, 1986.

[49] A. Seyfarth, M. Geyer, M. Günther, and R. Blickhan, "A movement criterion for running," *Journal of Biomechanics*, vol. 35, pp. 649–655, 2002.

[50] Y. Blum, J. Rummel, and A. Seyfarth, "Advanced swing leg control for stable locomotion," *Autonome Mobile Systeme 2007*, pp. 301–307, 2007.

[51] M. Ahmadi and M. Buehler, "Controlled Passive Dynamic Running Experiment with the ARL Monopod II," *IEEE Trans. on Robotics*, vol. 22, no. 5, pp. 974–986, 2006.

[52] A. Seyfarth, H. Geyer, and H. Herr, "Swing-leg retraction: a simple control model for stable running.," *The Journal of experimental biology*, vol. 206, no. Pt 15, pp. 2547–2555, 2003.

[53] H. Geyer, A. Seyfarth, and R. Blickhan, "Spring-mass running: Simple approximate solution and application to gait stability," *Journal of Theoretical Biology*, vol. 232, no. 3, pp. 315–328, 2005.

[54] U. Saranli, Ö. . Arslan, M. M. Ankarali, and Ö. . 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.

[55] H. Yu, M. Li, P. Wang, and H. Cai, "Approximate Perturbation Stance Map of the SLIP Runner and Application to Locomotion Control," *Journal of Bionic Engineering*, vol. 9, no. 4, pp. 411–422, 2012.

[56] Z. Shen and J. Seipel, "A Piecewise-Linear Approximation of the Canonical Spring-Loaded Inverted Pendulum Model of Legged Locomotion," *Journal of Computational and Nonlinear Dynamics*, vol. 11, no. 1, p. 011007, 2016.

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

[58] G. Brown, Ben Zeglin, "The Bow Leg Hopping Robot," *Proceedings - IEEE International Conference on Robotics and Automation*, no. May, 1998.

[59] G. Zeglin and B. Brown, "Control of a bow leg hopping robot," *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 1, no. May, pp. 793–798, 1998.

[60] P. M. Wensing and D. E. Orin, "3D-SLIP steering for high-speed humanoid turns," *IEEE International Conference on Intelligent Robots and Systems*, no. Iros, pp. 4008–4013, 2014.

[61] J. K. Yim and R. S. Fearing, "Precision Jumping Limits from Flight-phase Control in Salto-1P," *IEEE International Conference on Intelligent Robots and Systems*, pp. 2229–2236, 2018.

[62] D. W. Haldane, J. K. Yim, and R. S. Fearing, "Repetitive extreme-acceleration (14-g) spatial jumping with Salto-1P," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 3345–3351, 2017.

[63] J. Englsberger, P. Kozlowski, and C. Ott, "Biologically Inspired Dead-beat controller for bipedal running in 3D," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 989–996, 2015.

[64] J. Englsberger, P. Kozlowski, and C. Ott, "Biologically inspired deadbeat control for running on 3D stepping stones," *IEEE-RAS International Conference on Humanoid Robots*, vol. 2015-Decem, pp. 1067–1074, 2015.

[65] A. Sayyad, B. Seth, and P. Seshu, "Single-legged hopping robotics researchA review," *Robotica*, vol. 25, no. 05, pp. 587–613, 2007.

[66] G. Zeglin and B. Brown, "The bowleg hopping robot [online]," 2007. `https://www.cs.cmu.edu/~garthz/research/bowleg/bowleg.html`, last accessed 2019-09-19.

[67] P. Gregorio, M. Ahmadi, and M. Buehler, "Design, control, and energetics of an electrically actuated legged robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 27, no. 4, pp. 626–634, 1997.

[68] M. Ahmadi and M. Buehler, "Preliminary experiments with an actively tuned passive dynamic running robot," in *Experimental Robotics V*, pp. 312–324, Springer, 1998.

[69] J. Grimes and J. Hurst, "The design of atrias 1.0 a unique monopod, hopping robot," *Proceedings of the Fifteenth International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pp. 548–554, 2012.

[70] A. Ramezani, J. W. Hurst, K. Akbari Hamed, and J. W. Grizzle, "Performance Analysis and Feedback Control of ATRIAS, A Three-Dimensional Bipedal Robot," *Journal of Dynamic Systems, Measurement, and Control*, vol. 136, no. 2, p. 021012, 2013.

[71] A. Hereid, S. Kolathaya, M. S. Jones, J. Van Why, J. W. Hurst, and A. D. Ames, "Dynamic multi-domain bipedal walking with atrias through SLIP based human-inspired control," *Proceedings of the 17th international conference on Hybrid systems: computation and control - HSCC '14*, pp. 263–272, 2014.

[72] D. W. Haldane, M. Plecnik, J. K. Yim, and R. S. Fearing, "A power modulating leg mechanism for monopedal hopping," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-Novem, pp. 4757–4764, 2016.

[73] M. F. Hale, J. L. Du Bois, and P. Iravani, "Agile and Adaptive Hopping Height Control for a Pneumatic Robot," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 5755–5760, 2018.

[74] J. K. Hodgins and M. H. Raibert, "Adjusting Step Length for Rough Terrain Locomotion," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 289–298, 1991.

[75] M. F. Hale, J. L. du Bois, and P. Iravani, “A comparison of analytical and empirical controllers for the slip hopper,” in *18th Annual Conference on Towards Autonomous Robotic Systems, TAROS 2017*, pp. 79–85, Springer Verlag, 2017.

[76] W. J. Schwind and D. Koditschek, “Approximating the Stance Map of a 2-DOF Monoped Runner,” *Journal of Nonlinear Science*, vol. 10, no. 5, pp. 533–568, 2000.

[77] H. B. Mann and D. R. Whitney, “On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other,” *The Annals of Mathematical Statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[78] J. F. de Carvalho, N. R. Draper, and H. Smith, “Applied Regression Analysis (2nd ed).,” *Journal of the American Statistical Association*, vol. 76, no. 376, p. 1012, 1981.