Fuzzy Control of Hopping in a Biped Robot

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

By

Yiping Liu, B.S.E.E.

* * * * *

The Ohio State University

2010

Master's Examination Committee:

Dr. David E. Orin, Adviser

Dr. Yuan F. Zheng

Approved by

Adviser

Electrical and Computer Engineering

ABSTRACT

Current bipedal robots with articulated legs, even the most impressive prototypes to date, still lack the ability to execute dynamic motions such as jumping and running with comparable performance to biological systems. Recently a new biped prototype, KURMET, has been built at OSU to serve as an experimental platform for further investigation into the performance of dynamic movements in bipedal machines with biologically-realistic features. KURMET has series-elastic actuators (SEA) at all leg joints. The presence of SEAs provides the compliance that is needed in dynamic motions, yet also complicates the controller's tasks, especially when combined with articulated legs in a system that is not naturally stable.

This thesis develops a fuzzy control system for hopping with KURMET. With this controller, KURMET can stably hop at varying heights and forward/backward velocities. The control system is arranged into two levels. The low-level control executes the hop motion. It employs a hopping state machine that is specifically designed to accommodate the natural dynamics of the SEAs. The high-level control is a fuzzy controller that is called at discrete instances (every top of flight (TOF)) to regulate the key parameters in the state machine. Through proper selection of these parameters, the desired hop height and velocity can be achieved. The fuzzy rulebase is generated via an iterative training process, which is done off-line through dynamic

simulation using detailed models of the articulated mechanism and the series-elastic actuation. The fuzzy rulebase is later modified by on-line adaptation.

The fuzzy rulebase has fewer than 200 rules; however, the overall fuzzy control system is able to produce robust and accurate hopping performance in KURMET. Experimental data shows that the maximum error of the torso height at TOF is controlled within 1 cm and the maximum error of the torso velocity at TOF is controlled within 5 cm/s.

This thesis also experimentally investigates the high jump potential in KURMET. With a jumping state machine that is modified from the previous hopping state machine, KURMET is able to produce a maximum nominal jump height of 75 cm. When normalized to the length of the biped's link segments (25 cm), this performance is significant relative to human jumps.

To the great national parks.

ACKNOWLEDGMENTS

First I would like to thank my advisor Dr. David Orin for his insightful guidance, understanding and patience throughout the project. Additionally I want to thank him for spending so many extra hours helping me improve my writing skills. I would also like to thank Dr. Jim Schmeideler for providing timely and thoughtful opinions over the weekly video conferences. And I'd like to thank Dr. Yuan F. Zheng for taking the time to serve on my committee.

I want to express my gratitude to my fellow group members, too. First I want to thank Brian Knox for designing KURMET so sturdy that it can survive so many test hops. Also my work in this thesis would not possibly be realized in the physical machine without Pat Wensing's excellent computer engineering skills. I particularly thank Pat for his availability during the process. Ghassan Bin Hammam provided me lots of constructive suggestions on programming and he is so experienced with dynamic simulation. Mountain man Matt Hester helped me understand the mechanical system in many cases and his pioneer control work has served as a valuable reference for my work. I feel so honored working with those guys.

I would also want to thank Kelsey and Julie Stehli for showing their support at my thesis defense.

Finally, I would like to thank my family for their love through the years. Their constant encouragement and support have always been my source of confidence.

VITA

November 4, 1984 ..... Born - Nanjing, China

June 2007 ..... B.S. Electrical and Computer Engineering,
Nanjing University of Science & Technology,
Nanjing, China

January 2009 - June 2009 ..... Graduate Research Associate,
The Ohio State University,
Columbus, Ohio

September 2009 - December 2009 ..... Graduate Research Associate,
The Ohio State University,
Columbus, Ohio

FIELDS OF STUDY

Major Field: Electrical and Computer Engineering

Studies in:

Robotics	Professor D.E. Orin
Computer Engineering	Professors Yuan F. Zheng, D.E. Orin, F. Özgüner, J.E. Degroat
Control Engineering	Professors A. Serrani, V.I. Utkin
Computer Graphics	Professor R. Crawfis

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation and Background

Legged machines, unlike wheeled or tracked vehicles, require only discrete footholds instead of full contact with the surface along the path traveled, and thus have more advantages when traversing unprepared terrains [1]. The potential applications for legged machines include scientific exploration in extraterrestrial locations, search-and-rescue missions in hazardous situations, land mine detection on the military field or forestry in mountainous terrain. Since human beings are bipeds themselves, bipedal robots are of unique interest. Bipedal robots, especially those with articulated legs, can better assist and interact with humans and access the human's immediate environments. An outstanding example is Honda's humanoid robot ASIMO [2].

However, in order to gain the agility offered by legged locomotion and effectively function in realistic environments, the bipedal robot must be able to perform dynamic movements without compromising its stability. ASIMO's bipedal locomotion, based upon the Zero Moment Point (ZMP) criterion [3], is mostly quasi-static. It does not execute the natural, fluid movements typical of humans and other biological bipeds. Its dynamic "running" gaits are actually quite similar to its quasi-static walking gaits.

In recent years, several experimental bipedal robots with articulated legs have been developed to investigate dynamic movements in bipedal locomotion. Mowgli, a frog-like biped, can produce one single high jump and land safely utilizing its special biologically-inspired mechanism: an artificial musculoskeletal system that consists of six pneumatic muscle actuators [4]. Later, Ernie, a five-link *planar* biped with parallel knee compliance was developed at The Ohio State University [5]. It adopted a control strategy called hybrid zero dynamics (HZD) [6] and has shown the ability to perform dynamic, stable walking at a variety of speeds. More recently, Boston Dynamics has developed an impressive anthropomorphic robot prototype, Petman, that walks dynamically like a real person [7]. Petman can balance itself and walk freely, even maintain stability when pushed; It can also walk as fast as 3.2 mi/h. However, despite these and other projects focused on dynamic locomotion in bipedal robots, many dynamic movements such as continuous hopping, high-speed running and many other non-cyclic maneuvers such as sudden start and stop, or sharp turn, with comparable performance to animals, have rarely been demonstrated in a biped robot with biologically realistic mechanical structures.

Thus, propelled by the purpose to gain more insight into the dynamic motions in bipedal robots, a new prototype bipedal experimental platform has been developed at The Ohio State University. The prototype is named KURMET, which is an acronym for **K**inematically **U**nderactuated **R**obot for dynamic **M**aneuver **E**xperimental **T**esting. See Figure 1.1. KURMET is a five-link planar bipedal robot, just like its predecessor Ernie, but is smaller and lighter. It was originally designed and assembled by Knox [8]. Wensing configured and wired the computer control electronics for KURMET [9] and

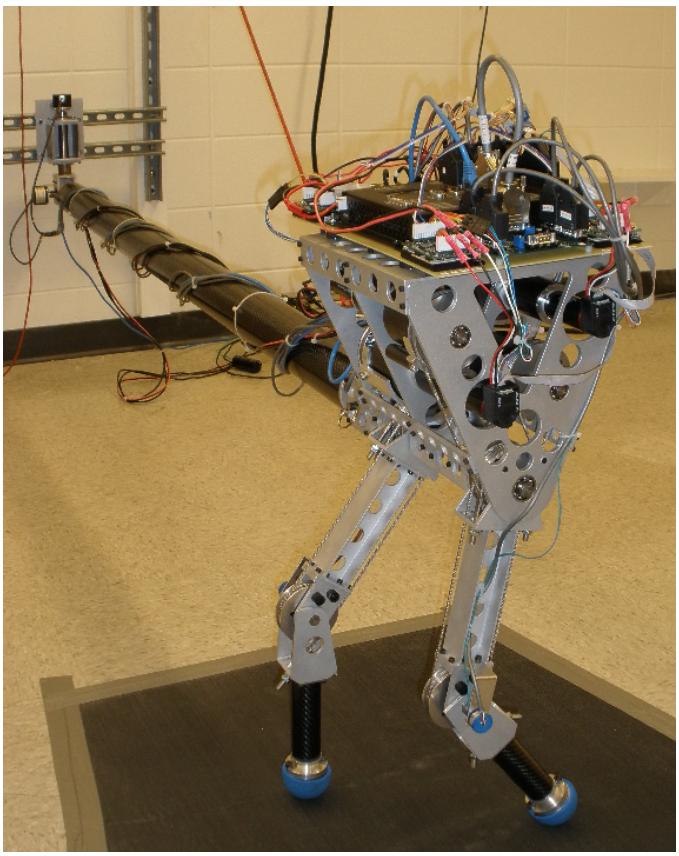

Figure 1.1: KURMET, an experimental biped designed for the study of dynamic movements.

later Hester made several changes to KURMET's mechanical parts to guarantee a more stable performance of the experimental biped.

In order to provide KURMET the capabilities that are required to perform jumping, hopping and running, all its leg joints are designed to be series-elastically actuated. The series-elastic actuation scheme, which is inspired by biological study of muscles and tendons, is believed to be an effective way to provide the explosive leg power necessary for dynamic movements and improve the performance of robot-environment interaction. It has so far been used successfully in a number of

dynamic bipedal robots [10, 11]. Paluska and Herr [12] recently have shown that series-elasticity can amplify the actuator power output over a limited stroke length, and thus is of critical importance for jumping robots. However, they only showed the linear actuator case. Curran studied the prototype leg, Hopper, that was built by Remic [13], to analyze the effects that the series-elastic actuator (SEA) has on the articulated leg [14]. The Hopper is a single articulated leg, with two revolute joints at the hip and knee, constrained to vertical motion by four rails. It only has an SEA at the knee joint; the hip joint uses a direct-drive actuator. Curran used an evolutionary search algorithm to optimize the parameters of the SEA to produce a highest jump. His result demonstrated the need to have an SEA in both the knee and hip joints for a legged jumping machine. The lessons learned from the Hopper provided key insights for the actuator design in KURMET. Later the knee actuator of the Hopper was redesigned by Knox to be a unidirectional-SEA to help reposition the leg in the flight during repetitive jumping [8]. Such a feature is also inherited in the design of KURMET.

Although series compliance has improved the performance for legged machines, it also significantly increases the control difficulties. Series-elastic actuators, combined with the articulated legs and the highly nonlinear contact-flight hybrid dynamics, together make KURMET a very complex system to be controlled using an analytical approach. In recent years, intelligent control strategies have emerged as an alternative way to address such complexities in dynamic systems. Antsaklis and Passino [15] describe intelligent control systems as a necessary development to address the increasingly complex nature of dynamic systems. Unlike the more traditional analytic approaches, intelligent control strategies do not require a complete mathematical

model for the overall system. Instead, the intelligent control strategies help the controller gain the essential control knowledge needed to manage the dynamic system through a progressive learning process.

Previous applications of intelligent control schemes in dynamic legged locomotion at The Ohio State University have shown promising results. Marhefka [16, 17] developed a direct adaptive fuzzy controller to control the galloping and bounding in a simulated planar quadruped with prismatic legs. His fuzzy controller was unique since it started with no control knowledge but only a few heuristics that are derived from the physical understanding of the system to guide its learning. He innovatively designed a training process applying those heuristics to initialize the fuzzy rulebase. After the initial rulebase was acquired, he used an adaptation mechanism to modify the rule outputs in the fuzzy rulebase to improve the galloping/bounding performance. His fuzzy controller outperformed the modified Raibert controller [1].

Krasny [18, 19] explored a practical approach to generate various high-speed motions such as galloping, turning, jumping, and stopping in a simulated quadrupedal model with biologically realistic characteristics. He first developed a series of low-level primitive functions for each leg which can be combined sequentially to create various behaviors. Then he used a multi-objective genetic algorithm (MOGA) to search for parameter values for these functions to acquire the desired motion. The use of the MOGA and control architecture in Krasny's work has resulted in the first biological-mode, fully spatial gallop in an articulated-leg model. Some solutions found by the MOGA, like using sliding to quickly stop from high-speed galloping, was unexpected yet robust, illustrating its effectiveness from another aspect.

Realizing that the previous high-speed quadruped turns were only able to be controlled around a fixed point due to the fact that the quadruped system is significantly more responsive at high speed than low speed, Palmer [20, 21] in his Ph.D. work, carefully studied the relationship between lateral touchdown positions and the resulting motions, and developed a direct adaptive fuzzy controller which was the first to achieve control of high-speed turning at variable speeds in a quadruped system with articulated legs and practical leg mass properties in a simulation environment with realistic friction coefficients and system losses. All the previous results motivate the application of intelligent control strategies for KURMET.

1.2 Objectives

The primary objective in this thesis is to control the hopping motion in the experimental biped KURMET using a fuzzy control strategy. To achieve this goal, first a practical low-level hopping controller will be implemented for KURMET to actually execute the hopping. Then a fully customized fuzzy controller will be built upon it to plan the thrust from hop to hop. With this controller, KURMET can stably perform consecutive two-leg synchronized hops (bunny hops) at various hop heights and velocities, and the hop height and velocity can also be accurately regulated during the hopping. This objective is meaningful considering the following two aspects:

1) Relatively little research has been directed to realize the various dynamic motions in a biologically-realistic bipedal robot. Take the bipedal hopping motion for example. Most previous hopping bipeds used prismatic instead of articulated legs, like Raibert's biped [1]; and in most cases, the hopping velocity cannot be accurately controlled. The work towards the primary objective in this thesis

will help researchers gain more understanding of the dynamic movements in bipedal locomotion.

2) Previous work has shown that intelligent control strategies, including fuzzy control, are particularly suited to the complexities presented in the control of dynamic movements in legged machines. However, most successful results are only verified in simulation. Simulations provide confidence in the control strategies but do not guarantee success in the physical system. Many new problems that are not captured by the simulation can occur in the actual tests. Palmer [22, 23] in his M.S. work developed a fuzzy controller to regulate the hopping velocity in a single articulated-leg. The controller worked well in simulation. However, when it was moved to the hardware, the OSU DASH leg [24], the performance was far from satisfactory. Hester's recent fuzzy controller for KURMET [25], when tested on a physical machine, can only allow it to hop in place, and the continuous hopping has limited stability. The work in this thesis will try to address the practical problems and make the fuzzy controller described at the beginning of this section really work on KURMET.

Like Marhefka's controller, the fuzzy controller developed here will also need a training process to initialize the rulebase. Training on the experimental machine will be a challenge due to the number of test hops involved. So this work will investigate an alternative approach, which will try to adequately train the rulebase in simulation so that when it is first applied to the experimental biped, the hopping can be immediately stabilized and the hop height and velocity be within a reasonable range from the desired values. If this goal can be reached, the initial rulebase will then be moved on to the physical controller for KURMET, and the on-line adaptation mechanism will

be turned on to improve the actual hopping performance when KURMET is hopping in situ.

A secondary objective is to investigate the high jump in KURMET. A normalized jump height s, which is defined as the ratio of the jump height and the leg segment length, or

$$s = \frac{h}{l_i} , \tag{1.1}$$

can be used to evaluate the performance of the high jump [26]. According to Knox's expectation [8], KURMET should be able to jump comparable to a human, with $s = 3.0$. That is to say, the maximum jump height should be around 75 cm. Part of the effort in this thesis will be spent to modify the existing low-level hopping controller to make it optimal for a high jump. This investigation will only focus on optimizing a single high jump and mainly rely on an experimental approach; the fuzzy control strategy will not be involved.

1.3 Organization

The remainder of this thesis is organized as follows: Chapter 2 will focus on modeling of the key elements of the system that are crucial to the reliability of the simulation, including the model for KURMET articulation, actuator model, DC motor model and the ground contact model. The simulation method will also be presented. Tests related to system parameter estimation and calibration will be introduced at the end of the chapter.

Chapter 3 will develop the control strategy for KURMET hopping. First low-level hopping control (hopping state machine) will be described in detail. Certain parameters in the hopping state machine will be given particular consideration. Then

a discussion on optimizing the selection of the inputs/outputs for the higher level fuzzy controller will follow. Structure of the fuzzy controller, along with the important training and on-line adaptation methods, will be presented in the remainder of the chapter.

Chapter 4 will present the hopping control results, both from the simulation and the actual tests on KURMET. Chapter 5 will provide the description of the low-level state machine optimized for a KURMET high jump, along with the corresponding test results. The final chapter, Chapter 6, will summarize the work in this thesis and put forward suggestions for future work.

CHAPTER 2

SYSTEM MODELING AND SIMULATION

2.1 Introduction

The goal of this thesis is to develop the controller to produce the dynamic movement in the experimental bipedal robot KURMET. A fully customized fuzzy logic controller is developed to achieve this goal. For the successful functioning of the fuzzy controller, a reliable rule base and a properly designed low level controller are required.

Development of the rule base can be a complex task. An intensive 'training' process, which usually includes tens of thousands of repetitive trial movements, is necessary to populate the rule base. Clearly it is not practical to have the training done physically: the real machine is not built strong enough to sustain so many impacts, for instance in hopping, and such experimentation would be extremely cumbersome. Therefore the development of the fuzzy rule base is aided by dynamic simulation on the computational models of the real system and environment. If the model is a good approximation to the real system, then the rule base obtained from the off-line simulated training should provide a good reference for the fuzzy controller when it is

applied on the real machine. The virtual rule base can also be refined later through on-line adaptation to obtain better performance.

An additional challenge is that, during a dynamic maneuver where system states change quickly, an ill-conceived low level control can easily cause serious mechanical damage to the experimental system such as breaking the joints or burning up the motors. If a relatively accurate model of the experimental system is available, applying the low level control first on the model provides a safe and efficient way to predict how a certain low level control performs on the real machine. For the above reasons, a considerable effort is spent to establish a proper model for KURMET.

This chapter starts with a brief overall description of the existing bipedal robot system. The model for this system and its environment is then presented in detail. Following that, the dynamic library package and simulator software used to simulate this model are briefly introduced. In the remainder of this chapter, tests pertaining to system parameter estimation and model evaluation are discussed.

2.2 KURMET System Overview

2.2.1 The Mechanical Subsystem

KURMET, the experimental machine used to perform dynamic maneuvers, is an experimental bipedal robot [8]. It has a torso and two legs. Each leg is comprised of a thigh segment and a shank segment. They are all connected by revolute joints.

The thigh and shank are actuated in parallel by identical *Unidirectional Serial-Elastic Actuators* (USEA) [14]. Each USEA includes a Maxon Powermax EC-30 brushless motor, a planetary gearhead with a gear ratio of 126 and a spiral wound torsion spring. In order to reduce the weight on the legs, all the USEA actuators

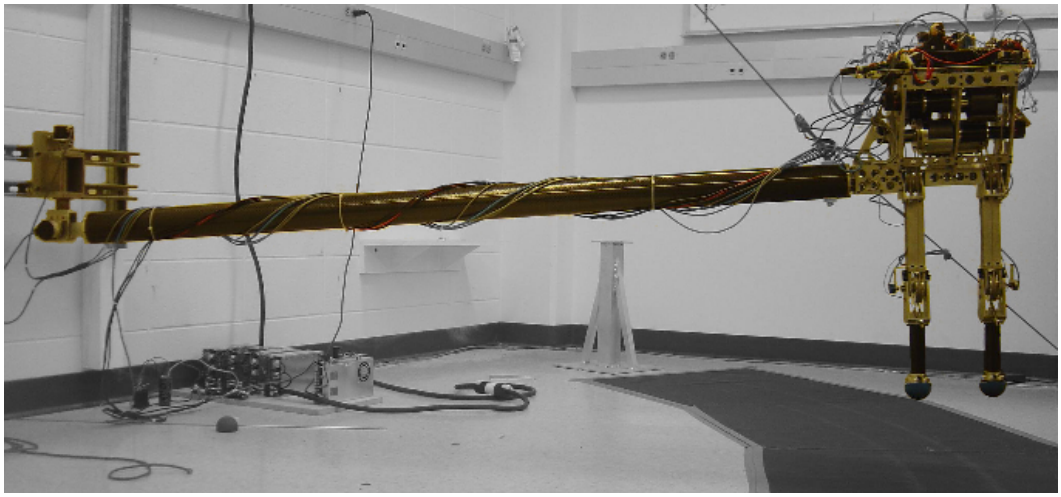

Figure 2.1: KURMET and its boom.

are mounted on the torso and connected to their respective leg segment through pulleys and vinyl coated steel cable drives. KURMET's movement is constrained to approximate 2D motion by a long boom that is anchored to the wall with a 2 degree-of-freedom (DOF) joint (boom pitch and boom yaw). KURMET's torso is attached to the end of the boom via an unactuated revolute joint, whose rotational axis is coincident with the hip axis. Therefore KURMET can only maneuver on a spherical surface centered at the boom's wall anchor point and its sagittal plane is always perpendicular to the boom. See Fig. 2.1. The length of the boom is 2 m and the separation between the two legs is 0.18 m.

However, in order to reduce the complexity of the control, currently, the joint between the torso and boom has been locked by inserting a pin so that the torso's coronal plane is always kept perpendicular to the ground. The height of the boom's anchor point on the wall is adjusted so that when KURMET stands on the ground

with both legs fully extended right under the hip joints, the boom is level. Each leg segment has a length of 0.25 m, so the anchor point of the boom is 0.5 m above the ground. The overall weight of KURMET is 14.98 kg and the weight of the boom is 2.7 kg.

2.2.2 The Electrical Subsystem

The electrical subsystem of KURMET includes power supplies, actuator amplifiers, sensors, a host computer and a Galil DMC-4040 motion controller, as shown in Fig. 2.2. Other than the power supplies, host computer and the encoders for the boom, all electrical hardware is present on-board the biped KURMET .

An array of 220VAC-to-48VDC power supplies is used to power 4 AMC AZBDC20A8 amplifiers which serve as motor drives. Each amplifier can attain a peak current of 20A for 2 s and continuously supply 12 A. Another 110VAC power supply is used to power the Galil motion controller and sensory electronics.

Sensors include shaft encoders attached to the boom and boom support (to sense θ_{B1} and θ_{B2}), a shaft encoder attached to the torso (to sense θ_{torso}), potentiometers attached to the joints on the legs (to detect θ_{lh}, θ_{lk}, θ_{rh} and θ_{rk}), shaft encoders attached to the DC motors (to sense $\theta_{m,lt}$, $\theta_{m,ls}$, $\theta_{m,rt}$ and $\theta_{m,rs}$) and the foot contact switches mounted on the feet (to detect ground contact). The definitions of those symbols are given in Section 2.3.

The Galil DMC-4040 motion controller is mounted on the torso and has a RISC based 68000 processor at its core. It can gather all the sensory information that is needed for a user-customized control. The controller can take 8 encoder inputs (4 primary inputs and 4 auxiliary inputs), 8 optically-isolated digital inputs and

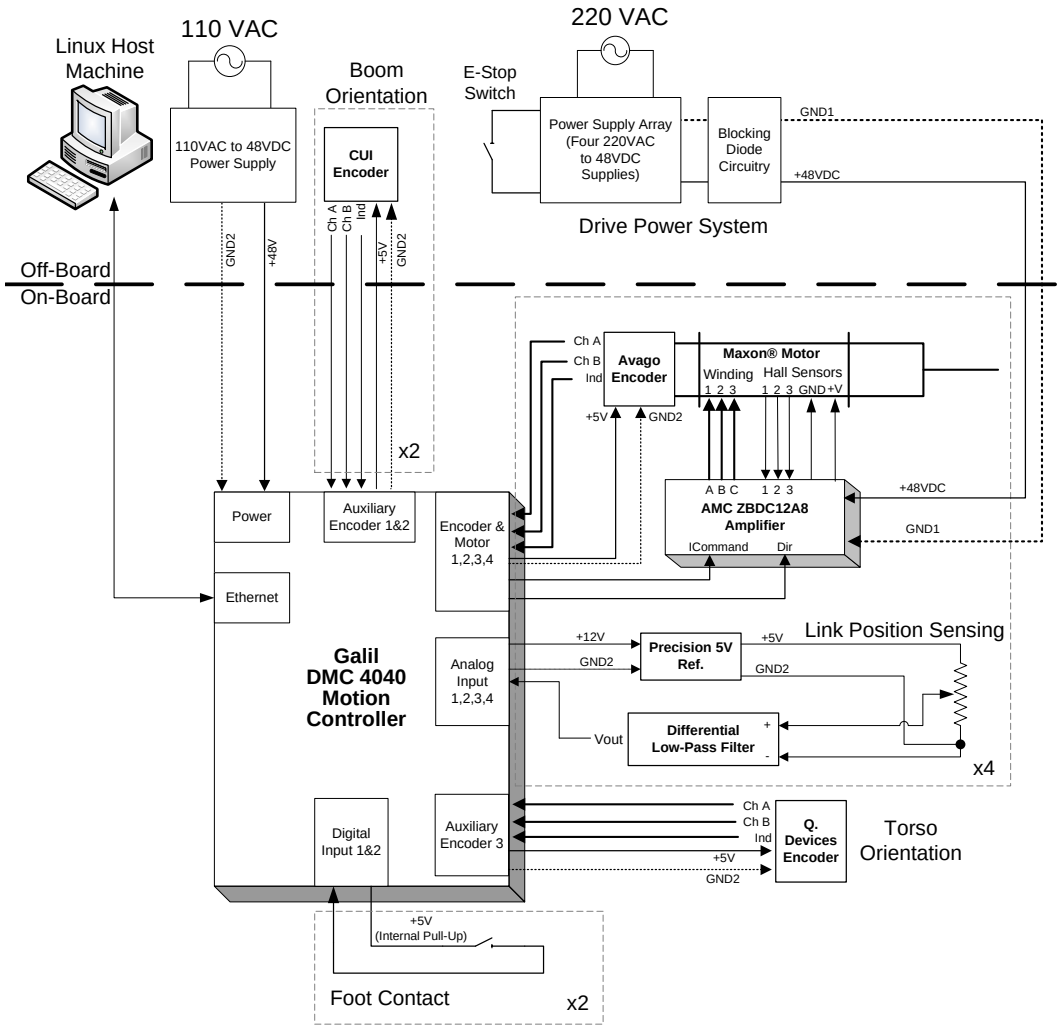

Figure 2.2: The electrical subsystem of KURMET [9]. (Courtesy of Patrick Wensing.)

8 analog inputs which interface with a 12-bit analog-to-digital converter. In the case of KURMET, the 4 primary encoder inputs connect to the encoders on the motors and provide closed-loop position control for the motors based on the feedback from the encoders. Three auxiliary encoder inputs are used for boom and torso

orientation. Four analog inputs are used to receive joint angles from the hip and knee joint potentiometers. Two digital inputs are used for the ground contact signals.

The Galil controller provides a variety of motor trajectory control options which are characterized by 'motion modes'. The 'contour mode' allows an arbitrary position trajectory to be assigned for the motor, and is most applicable when a complex trajectory for the motor is required, or when the trajectory needs to be dynamically retargeted during its execution. The contour is comprised of a series of motor position increments each over a specific time interval. The trajectory of the motor will be smooth if each position increment on the contour is small enough.

The Galil controller also has the ability to communicate with an off-board host computer that runs real-time Ubuntu Linux through a 100 Mbs ethernet connection, which makes the communication delays negligible. During an experiment, first the user-defined motor position trajectory is generated on the host computer and then the setpoints evaluated on the trajectory are sequentially relayed to the Galil motion controller at the specific time interval. The motion controller is set to work in the contour mode and can dynamically update the contour every time a new setpoint is generated (the actual interval is 1 ms).

The electrical subsystem for KURMET was built mainly by Patrick Wensing. More details could be found in his thesis [9].

2.3 System Model

2.3.1 Articulated-Body Model

A combination of 5 rigid links is used to model KURMET and 2 additional rigid links are used to model the boom. Each rigid link includes information on its own

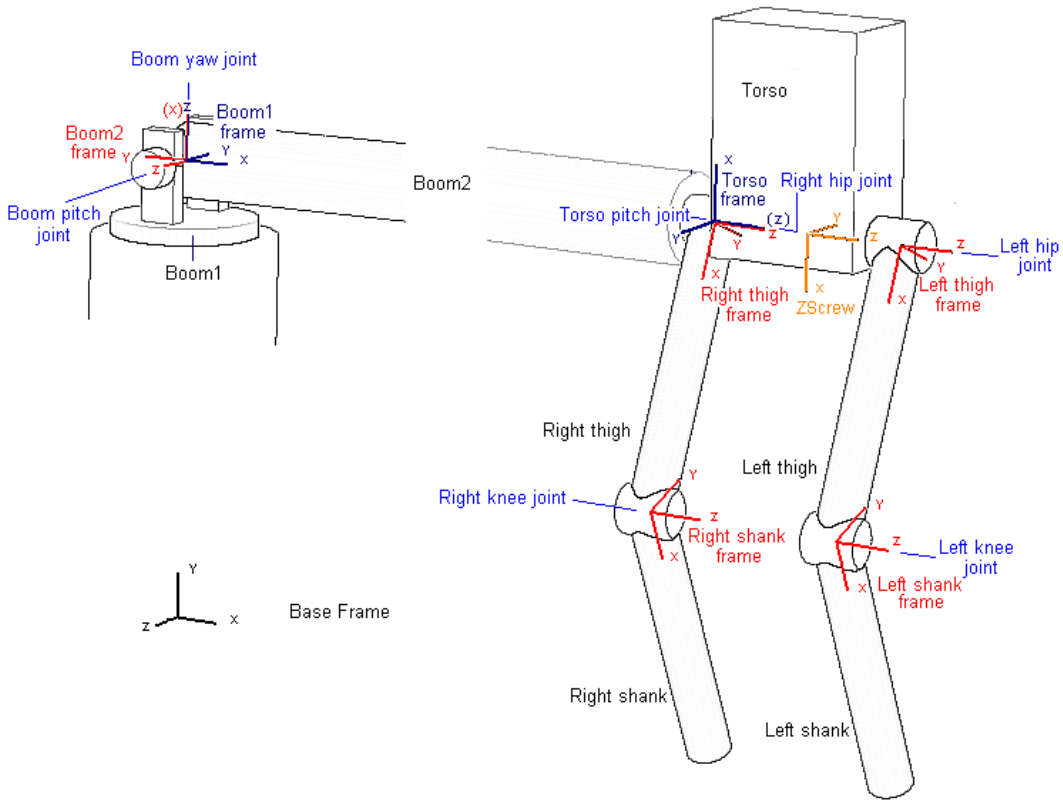

Figure 2.3: The articulated-body model for KURMET and its boom.

mass, center of mass and inertia. All the links are connected with revolute joints. Figure 2.3 shows all of the coordinate frames affixed to each link. Each coordinate frame is properly oriented so that the z-axis of the coordinate frame is always coincident with the joint axis. The relation of the link frames is visualized in the tree structure in Fig. 3.11. An auxiliary frame called a ZScrew is added. The ZScrew frame is a fixed transformation between the torso frame and thigh frames that allows the zero configuration for KURMET to be in an upright standing position. Contact points are placed at the distal end of each shank to model the hemispherical foot.

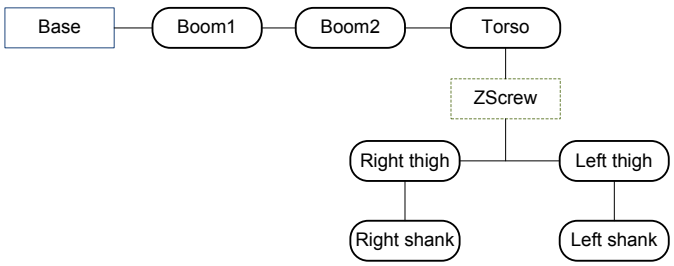

Figure 2.4: Tree structure of articulated links. This figure shows the relationships among all coordinate frames attached to the various links. The auxiliary ZScrew frame is displayed in the dash box, and is used to provide a better "zero" configuration for the system.

The connection of these articulated links can also be described in terms of link parameters (fixed and variable) using the *Denavit-Hartenberg convention* [27]. Table 2.1 provides the detailed link parameters for KURMET and its boom. Note that in Table 2.1, θ_{rh} indicates the right hip joint angle, θ_{rk} indicates the right knee joint angle, etc. .

Table 2.1: Link parameters of KURMET for the D-H convention

Link Frame	Parent	$a\ (m)$	$\alpha\ (rad)$	$d\ (m)$	$\theta\ (rad)$ (Joint Angle)
Base	-	-	-	-	
Boom1	Base	0	$-\frac{\pi}{2}$	0.5	θ_{B1}
Boom2	Boom1	0	$\frac{\pi}{2}$	0	θ_{B2}
Torso	Boom2	0	$\frac{\pi}{2}$	2.08	θ_{torso}
ZScrew	Torso	-	-	0.09	$-\pi$
Right Thigh	ZScrew	0	0	-0.09	θ_{rh}
Right Shank	Right Thigh	0	0	0.25	θ_{rk}
Left Thigh	ZScrew	0	0	0.09	θ_{lh}
Left Shank	Left Thigh	0	0	0.25	θ_{lk}

2.3.2 Parallel Actuation

With both hip actuators and knee actuators mounted on the torso, KURMET uses a parallel actuation scheme. The torque that the hip actuator exerts on the thigh is indicated as τ_h. The torque that the knee actuator exerts on the shank is indicated as τ_k. Since KURMET is parallel actuated, both the hip torque and knee torque are developed relative to the torso; thus, the knee torque will have no direct effect on the thigh.

However, the dynamic engine (*DynaMechs*) used by our simulator (*RobotBuilder*) assumes that all the articulated bodies are serially actuated [28]. If KURMET were serially actuated, then the knee torque would generate a torque on the thigh that has an equal and opposite effect that is, the thigh torque equals the difference of the hip and knee torques (the shank torque still equals knee torque).

During simulation of KURMET, the dynamic engine takes joint torques (namely, torques from the actuators) as inputs from the simulator's front end. When the thigh torque is used in the dynamic calculation, assuming that the system is serially actuated, the dynamic engine will automatically deduct knee torque from the hip torque to obtain the net thigh torque, which is not desired for parallel actuation [29]. In order to adapt the dynamic engine to parallel actuation, simply calculate:

$$\tau_h' = \tau_h + \tau_k \ , \tag{2.1}$$

$$\tau_k' = \tau_k \ . \tag{2.2}$$

The primed values are then furnished to the dynamic engine as inputs. In such a way, the dynamic engine still runs in a serial actuation mode but the difference on the thigh torque has been offset so that the overall effect is that of parallel actuation.

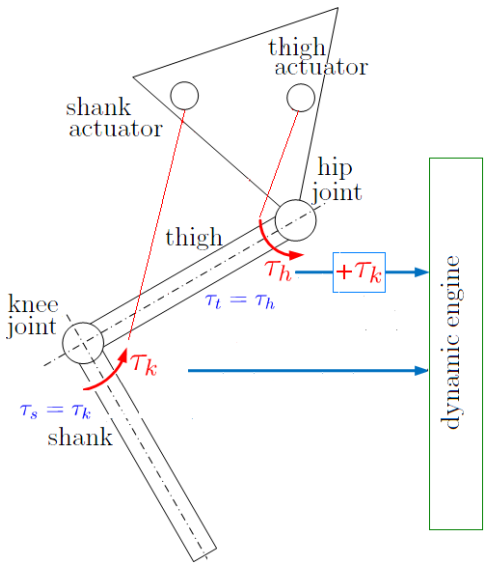

Figure 2.5: Parallel actuation adapted to the dynamic engine. KURMET adopts parallel actuation scheme, while the dynamic engine used to simulate the KURMET model takes a serial actuation scheme as the default. Therefore, during simulation, the knee joint torque (τ_k) is added to the hip joint torque (τ_h), before the latter is fed to the dynamic engine as an input for the dynamic functions.

Figure 2.5 provides a diagram of the parallel actuation scheme and interface to the dynamic engine. The further details of the simulation process are given in Section 2.4.

2.3.3 Unidirectional Series-Elastic Actuator (USEA) Model

USEA: A Basic Idea

Each of the four USEAs mounted on the KURMET torso is powered by a brushless DC motor. The rotor of each motor connects to its corresponding link through a gearbox and spiral torsional spring and cable drive in a serial manner. There are two different working modes for a USEA (Fig. 2.6). See [14] for more details on the design.

In this thesis, the rotor angle of the motor, before the gearbox and with respect to the torso frame, is referred to as the motor position θ_m; its first and second time derivatives are referred to as the motor rate $\dot{\theta}_m$ and motor acceleration $\ddot{\theta}_m$, respectively. The total electromagnetic torque of the motor is denoted as τ_{em}. The torque output from the motor onto the gearbox is denoted as τ_m. Later, subscripts such as rt, rs, lt, ls, t or s will also be added to the above notation when a specific actuator needs to be addressed. The subscript rt means 'right thigh', ls means 'left shank', etc. . θ_L is a general notation for the leg link position relative to the torso. As such, any of the previous subscripts could be substituted for L. The relationships between thigh position and hip joint angle, and shank position and knee joint angle are as follows:

$$\theta_t = \theta_h \ , \tag{2.3}$$

$$\theta_s = \theta_h + \theta_k \ . \tag{2.4}$$

Figure 2.7 gives a graphical illustration of the joint angles, link positions and motor positions. In this figure the positive direction is defined as counter-clockwise about the z-axis.

Ideally, the behavior of a USEA may be described by the equation (thigh USEA)

$$\tau_t = \begin{cases} K_s(\dfrac{\theta_{m,t}}{n_m} - \theta_t) & \dfrac{\theta_{m,t}}{n_m} > \theta_t \\ \eta \ n_m \tau_{m,t} & \dfrac{\theta_{m,t}}{n_m} = \theta_t \\ N.A. & \dfrac{\theta_{m,t}}{n_m} < \theta_t \end{cases} \tag{2.5}$$

or (shank USEA)

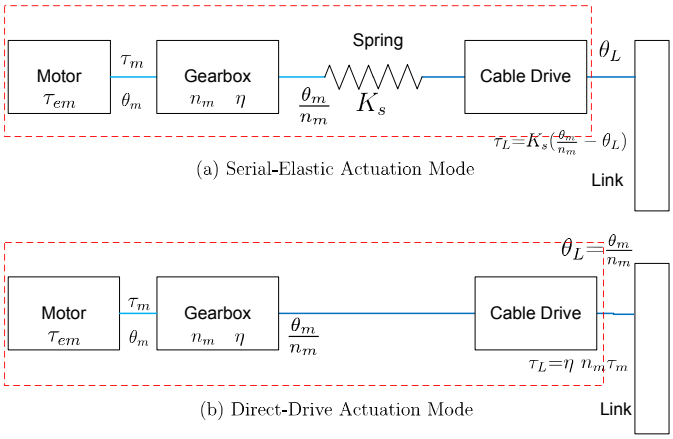

Figure 2.6: Two working modes of the unidirectional series-elastic actuator.

$$\tau_s = \begin{cases} N.A. & \frac{\theta_{m,s}}{n_m} > \theta_s \\ \eta\ n_m \tau_{m,s} & \frac{\theta_{m,s}}{n_m} = \theta_s \\ K_s(\frac{\theta_{m,s}}{n_m} - \theta_s) & \frac{\theta_{m,s}}{n_m} < \theta_s\ , \end{cases} \tag{2.6}$$

where η is the efficiency of the gearbox and n_m is its gear ratio. Take the USEA on the thigh for instance, clearly, when the thigh motor position, after the gearbox ($\frac{\theta_{m,t}}{n_m}$), is more positive than the thigh position θ_t, then the thigh link is driven by the tension torque of the spring, which is proportional to the angular difference between $\frac{\theta_{m,t}}{n_m}$ and θ_t. This is called the *Series-Elastic Actuation Mode*. When the thigh motor position coincides with the thigh position, the thigh torque is equal to the torque output by the thigh motor after the gearbox ($\eta\ n_m \tau_{m,t}$); i.e. the thigh is driven directly by the thigh motor, and the link does not see the spring. This is called the *Direct-Drive Actuation mode*.

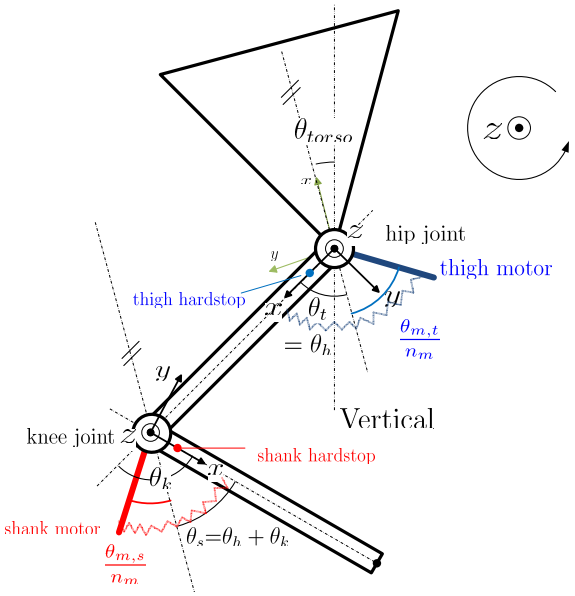

Figure 2.7: Joint angles, link positions and motor positions.

Equations 2.5 and 2.6 also imply that the thigh motor position (after the gear) is limitedd to be larger than thigh position and the shank motor position (after the gear) is limited to be smaller than the shank link position. Mechanical hardstops on the thigh and shank enforce these limits., Thus the name why "unidirectional". It also needs to be noted that during the Series-Elastic Actuation Mode, the deflection of the spring, and thus the link torque, cannot be directly controlled by the motor.

Pretensioning Torque

In the previous USEA model, when the motor engages the hardstop on the link, the angular deflection of the spring is zero, which means that there is no tensioning torque on the spring. On the actual machine, the position of the hardstop has been adjusted so that when the motor engages the hardstop, the spring is still deflected a

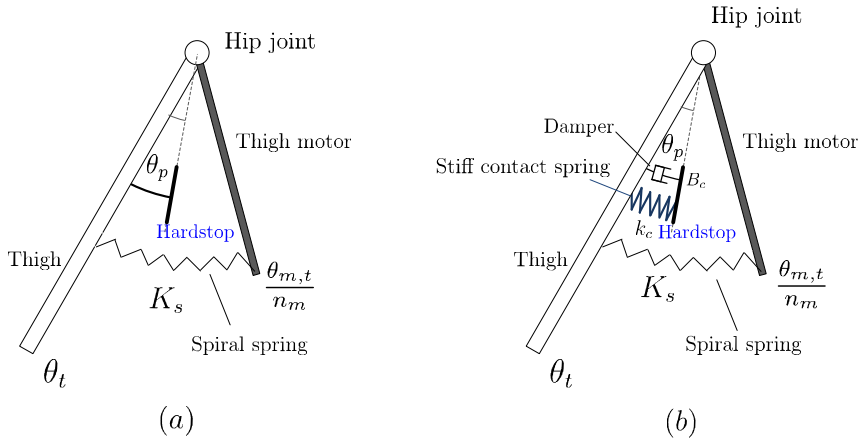

Figure 2.8: Modified USEA models (thigh). (a) shows the SEA with pretensioning torque. (b) adds in the unidirectional hardstop contact model on the basis of (a). In the configurations shown in (a) and (b), the spiral spring has deflection of $(\theta_{m,t} - \theta_t)$; the contact spring has no deflection.

few degrees (i.e. the link position and motor position after the gear do not coincide). See Fig. 2.8 (a). The spring torque generated by this deflection (θ_p) is called the pretensioning torque τ_p. Therefore the model of the thigh USEA (Eq. 2.5) is modified as (suppose θ_p is positive):

$$\tau_t = \begin{cases} K_s(\frac{\theta_{m,t}}{n_m} - \theta_t) & \frac{\theta_{m,t}}{n_m} - \theta_p > \theta_t \\ \eta\ n_m \tau_{m,t} & \frac{\theta_{m,t}}{n_m} - \theta_p = \theta_t \\ N.A. & \frac{\theta_{m,t}}{n_m} - \theta_p < \theta_t \end{cases} \tag{2.7}$$

and the model of the shank USEA (Eq. 2.6) is modified as

$$\tau_s = \begin{cases} N.A. & \frac{\theta_{m,s}}{n_m} + \theta_p > \theta_s \\ \eta\ n_m \tau_{m,s} & \frac{\theta_{m,s}}{n_m} + \theta_p = \theta_s \\ K_s(\frac{\theta_{m,s}}{n_m} - \theta_s) & \frac{\theta_{m,s}}{n_m} + \theta_p < \theta_s\ . \end{cases} \tag{2.8}$$

Unidirectional Hardstop Contact Model

Equations 2.7 and 2.8 have assumed an ideal contact model: as soon as the motor contacts the hardstop on the link, they acts as a single element, which is not realistic on the real machine. In practice, an elastomer element has been placed on the hardstop in order to reduce the impact when the motor bangs into the hardstop. The hardstop is modeled as a linear spring and a nonlinear damper operating in parallel on a rigid body (see Fig. 2.8 (b)). The contact spring has a very large spring constant K_c, where in this model $K_c \geq 10K_s$. The effective damping coefficient of the nonlinear damper (B_c) is linearly dependent on the motor penetration depth into the hardstop, by the constant λ_c. Marhefka has showed that the nonlinear damping contact model resolves a number of weaknesses in the linear damping model which includes discontinuous impact force and a coefficient of restitution that is independent of the impact velocity, and is thus more physically meaningful [30]. The thigh USEA model (Eq. 2.7) is then further modified as:

$$\tau_t = \begin{cases} K_s(\frac{\theta_{m,t}}{n_m} - \theta_t) & \frac{\theta_{m,t}}{n_m} - \theta_p > \theta_t \\ K_s(\frac{\theta_{m,t}}{n_m} - \theta_t) + K_c(\frac{\theta_{m,t}}{n_m} - \theta_p - \theta_t) & \\ \qquad +\lambda_c \cdot |\frac{\theta_{mt}}{n_m} - \theta_t - \theta_p| \cdot (\frac{\dot{\theta}_{m,t}}{n_m} - \dot{\theta}_t) & \frac{\theta_{m,t}}{n_m} - \theta_p \leq \theta_t \end{cases} \tag{2.9}$$

and the shank USEA model (Eq. 2.8) is further modified as:

$$\tau_s = \begin{cases} K_s(\frac{\theta_{m,s} - \theta_s}{n_m}) & \frac{\theta_{m,s}}{n_m} + \theta_p < \theta_s \\ K_s(\frac{\theta_{m,s} - \theta_s}{n_m}) + K_c(\frac{\theta_{m,s}}{n_m} + \theta_p - \theta_s) & \\ \qquad +\lambda_c \cdot |\frac{\theta_{m,s}}{n_m} + \theta_p - \theta_s| \cdot (\frac{\dot{\theta}_{m,s}}{n_m} - \dot{\theta}_s) & \frac{\theta_{m,s}}{n_m} + \theta_p \geq \theta_s \ . \end{cases} \tag{2.10}$$

The characteristic of the USEA on the thigh as described in Eq. 2.9 is also shown in terms of a $\Delta\theta$ vs. τ curve in Fig. 2.9 (damping term not included).

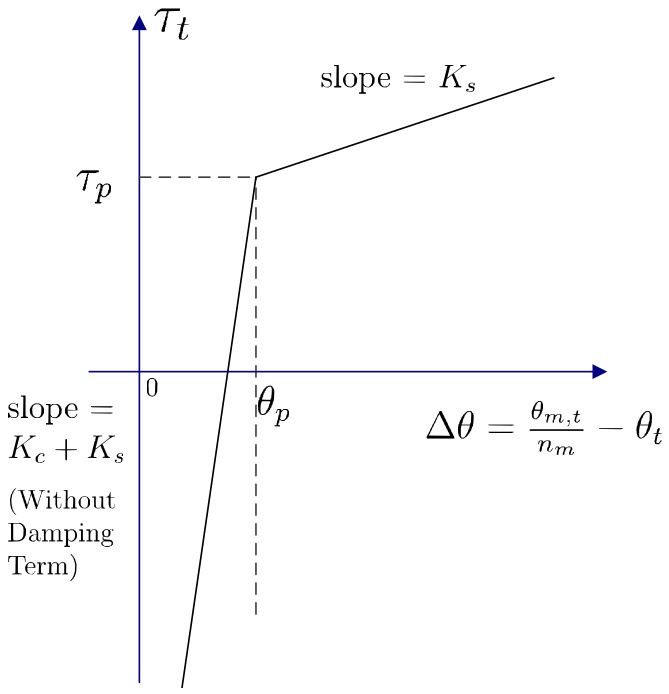

Figure 2.9: $\Delta\theta$ vs. τ curve for the USEA on the thigh. The damping term is not included.

DC Motor Model

The model for the DC motor used in this thesis was originally developed by Simon Curran. The derivation of the equations and more details can be found in his work [31].

The basic motor model can be described in two sets of equations. The first gives the armature circuit equations:

$$i_a = \frac{V - V_b}{R} \ , \tag{2.11}$$

$$V_b = k_b \cdot \dot{\theta}_m \ , \tag{2.12}$$

where i_a is the armature current, V is the terminal voltage across the motor, R is the armature resistance, V_b is the back EMF voltage and k_b is the back EMF constant.

The second set comprises of the motor dynamics equations:

$$k_{\tau m} i_a = J_m \ddot{\theta}_m + B_m \dot{\theta}_m + \tau_m \ , \tag{2.13}$$

$$\tau_m = \frac{\tau_L}{\eta \cdot n_m} \ , \tag{2.14}$$

where $k_{\tau m}$ is the torque constant, J_m and B_m are the inertia and damping of the motor itself, and τ_m is the output torque. It is equal to the load torque seen by the motor through the gearbox. η is the gearbox efficiency, and n_m is the gearbox ratio.

A current-control amplifier is used to provide current to the motor. The armature current i_a will follow the commanded current i_c from the amplifier, i.e. $i_a = i_c$, when the following condition is satisfied:

$$-V_{\max} \leq i_c R + k_b \dot{\theta}_m \leq V_{\max} \ , \tag{2.15}$$

where $V_{\max}$ is the voltage limitation enforced by the DC power supply.

However, if at some moment the above condition fails, the damping effect of the back EMF voltage will then begin to restrict the armature current to follow the commanded current. In that case:

$$i_a = \frac{V_{\max} - k_b \dot{\theta}_m}{R} < i_c \qquad \text{if } i_c, i_a > 0 \tag{2.16}$$

or

$$i_a = \frac{-V_{\max} - k_b\dot{\theta}_m}{R} > i_c \qquad \text{if } i_c, i_a < 0 \ . \tag{2.17}$$

As such, the DC motor model with the limitation of the amplifier can be summarized as follows: (given the fact that $k_b = k_{\tau m}$ in metric units):

$$\frac{k_{\tau m}V_{\max}}{R} = J_m\ddot{\theta}_m + (B_m + \frac{k_{\tau m}^2}{R})\dot{\theta}_m + \frac{\tau_L}{\eta \cdot n_m} \qquad \text{if } i_c > \frac{V_{\max} - k_{\tau m}\dot{\theta}_m}{R} \ , \tag{2.18}$$

$$\frac{-k_{\tau m}V_{\max}}{R} = J_m\ddot{\theta}_m + (B_m + \frac{k_{\tau m}^2}{R})\dot{\theta}_m + \frac{\tau_L}{\eta \cdot n_m} \qquad \text{if } i_c < \frac{-V_{\max} - k_{\tau m}\dot{\theta}_m}{R} \ , \tag{2.19}$$

$$k_{\tau m}i_c = J_m\ddot{\theta}_m + B_m\dot{\theta}_m + \frac{\tau_L}{\eta \cdot n_m} \qquad \text{Otherwise} \ . \tag{2.20}$$

The gearbox efficiency η can take two different values. The forward efficiency η_f applies when the motor drives the load (the motor outputs energy). The backward efficiency $\frac{1}{\eta_b}$ applies when the motor is driven by the load (energy is injected into the motor). For forward drive, the direction of $\dot{\theta}_m$ and the direction of τ_m are the same. For backward drive, the direction of $\dot{\theta}_m$ and the direction of τ_m are the opposite. Thus,

$$\eta = \begin{cases} \eta_f & \text{when } \dot{\theta}_m\tau_m > 0 \ , \\ \dfrac{1}{\eta_b} & \text{when } \dot{\theta}_m\tau_m \leq 0 \ . \end{cases} \tag{2.21}$$

The value of η_f is specified by the manufacturer, while the value of η_b is not, for the motor is not designed to be backdriven. However, this is a natural condition for our system. Note that Curran [31] arbitrarily set $\eta_b = \frac{1}{2}\eta_f$.

2.3.4 Ground Contact Model

The ground contact model is readily implemented in the dynamic engine and automatically integrated in the simulator (refer to Section 2.4). The ground is

modeled as a level surface. Certain points on the foot or other links are designated as possible contact points. The system only interacts with the ground at these contact points. When the simulator detects that a contact point is below the ground level, the foot or other link will start to feel the forces from linear springs and dampers in both the normal and tangential direction (with respect to the ground surface). See Fig. 2.10 (a). The point where the contact point first touches the ground surface is recorded as its "anchor point". This is also the point where the contact point interacts with the ground. When the contact point is below the ground surface, the anchor point is fixed as long as there is no sliding. The deflection of the normal (tangential) ground spring is the position difference between the anchor point and the contact point projected onto the normal (tangential) direction.

The actual hemispherical foot is represented by a group of contact points that are evenly distributed along the arc formed by the foot hemisphere and the xy-plane of the shank frame[1]. See Fig. 2.10 (b). The overall ground contact spring and damper seen by the foot are nonlinear even though the spring and damper seen by each single contact point are linear. The ground is characterized with friction coefficients which can be used to determine sliding. Since the ground spring constant is usually very large, the integration step size in the simulation is set small enough so that the behavior of the contact with high stiffness may be correctly simulated. An improper integration step size can easily cause the simulation to go unstable. More details about the implementation of the ground contact model may be found in the *DynaMechs* library reference manual [29] and Rodenbaugh's thesis [32].

[1]Previously the hemispherical foot is only simply modeled as a single contact point, however, later work has shown that the curved foot model can produce a much better real-world approximation in simulation than the single contact point foot model.

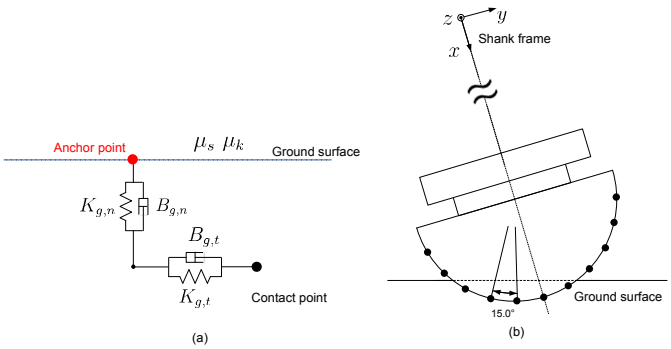

Figure 2.10: Ground contact model. (a) shows the normal and tangential spring/damper between the anchor point and the contact point. (b) shows the multiple contact points arrangement on the foot hemisphere. The 6 configurable parameters of the ground contact are: ground normal spring constant ($K_{g,n}$), ground tangential spring constant ($K_{g,t}$), ground normal damper constant ($B_{g,n}$), ground tangential damper constant ($B_{g,t}$), static friction coefficient (μ_s) and kinetic friction coefficient (μ_k).

2.4 Simulation Method

Dynamic simulation for the KURMET model that is discussed above is developed using *RobotBuilder*, a simulator application with a graphical user interface developed by Rodenbaugh [32]. *RobotBuilder* is built upon the *DynaMechs*, a dynamic engine library which was developed by Scott McMillan [29, 33]. *RobotBuilder* actually provides a very convenient way to utilize *DynaMechs*, so that one can quickly perform dynamic simulation for an articulated mechanism used in *DynaMechs*, without knowing all the details of the hierarchical classes.

Generally, simulation in *RobotBuilder* runs in loops (see Fig. 2.11). Each loop represents a discrete time point and the interval between each loop is set by the system integration step size. *RobotBuilder* embeds a user-implemented control in the simulation loop, which is called at every control step (the control step size is a multiple

of the system integration step size). In this user control, one can initialize/reset/read current system states, read the current foot contact status or change the output to the actuator for each of the articulated joints, just by calling certain functions from *DynaMechs*. And based upon those low level operations, a more sophisticated control strategy may be developed.

2.4.1 Simulation of Articulated-Body Dynamics

At time t, *RobotBuilder* first retrieves the 'next system states', calculated at time $t - \Delta t$, and sets them as the current system states. It then executes the user-implemented control if necessary and collects the current inputs to the system (from external forces, contact with the environment and the actuators) and delivers them to *DynaMechs*. Based on this information, *DynaMechs* calculates the articulated-body dynamic equations to find derivatives of the current states, and uses them, along with the integration step size Δt, to perform numerical integration to get 'next states' for time $t + \Delta t$. This process repeats. The details of the articulated-body dynamic equations that are implemented in *DynaMechs* may be found in [33].

2.4.2 Simulation of Motor Dynamics

DynaMechs does not provide for dynamic simulation of a DC motor model which is integrated into a USEA. It has to be done inside the user-implemented control. Suppose during the simulation loop that the control is called and the current states of the motor are θ_m and $\dot{\theta}_m$, the current commanded current is i_c and the current load torque delivered to the series-elastic element is τ_L. Following Eq. 2.18, $\ddot{\theta}_m$ can be calculated:

$$\ddot{\theta}_m = \begin{cases} \frac{1}{J_m}\left[\frac{k_{\tau m} V_{\max}}{R} - (B_m + \frac{k_{\tau m}^2}{R})\dot{\theta}_m - \frac{\tau_L}{\eta \cdot n_m}\right] & \text{if } i_c > \frac{V_{\max} - k_{\tau m} \cdot \dot{\theta}_m}{R} \\ \frac{1}{J_m}\left[-\frac{k_{\tau m} V_{\max}}{R} - (B_m + \frac{k_{\tau m}^2}{R})\dot{\theta}_m - \frac{\tau_L}{\eta \cdot n_m}\right] & \text{if } i_c < \frac{-V_{\max} - k_{\tau m} \cdot \dot{\theta}_m}{R} \\ \frac{1}{J_m}(k_{\tau m} i_c - B_m \dot{\theta}_m - \frac{\tau_L}{\eta \cdot n_m}) & \text{otherwise .} \end{cases} \tag{2.22}$$

Then the next states of θ_m and $\dot{\theta}_m$ may be calculated using simple Euler integration:

$$\theta_m(t+1) = \theta_m(t) + \dot{\theta}_m(t)\Delta t \ , \tag{2.23}$$

$$\dot{\theta}_m(t+1) = \dot{\theta}_m(t) + \ddot{\theta}_m(t)\Delta t \ . \tag{2.24}$$

The control step size and the system integration step size are both set to be the same (0.1 ms) during KURMET simulation so that the integration of the link states and the integration of motor states can be synchronized. However, other than the simulation for the motors, the rest of the control is updated every 10 control steps to match the control step size (1 ms) for the controller for the physical system. The simulation implemented in *RobotBuilder* also uses Euler integration for consistency, although there are other integration methods available in *DynaMechs*. Finally the integration step size (0.1 ms) has also proved to be small enough to simulate the relatively stiff ground contacts.

2.5 Parameter Estimation and Model Calibration

It is desirable to estimate some of the parameters of the model to reflect the characteristics of the physical system. This is specially important since off-line training of the fuzzy rulebase will be used to provide an initial controller for KURMET. Several relatively simple experiments have been completed on KURMET to coarsely calibrate the model.

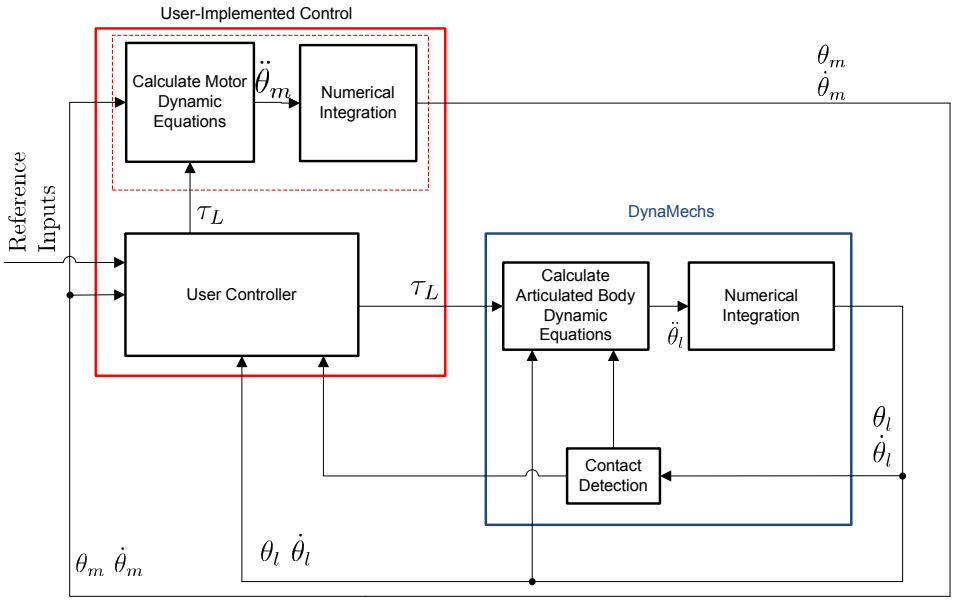

Figure 2.11: *RobotBuilder* simulation loop.

2.5.1 Estimation of the USEA Spring Constant

The USEA spring constant is the key parameter that can greatly influence the actual performance of KURMET. It has a manufacturer specified value of 30 Nm/rad. However, since the spring is fully customized, this value has never been verified. The test described below is used to verify the estimate for the spring constant.

The test is done on the left thigh (lt), the procedure is as follows: First shut off the left thigh motor and then manually push the left thigh in the positive counter-clockwise direction until slightly touches the joint limit. The left thigh motor position and the left thigh position will only differ by the small pretensioning deflection (θ_p). Then turn on the left thigh motor to provide it with the current that is just enough to hold its current position.

Next ramp up the current from the magnitude needed to hold the current motor position (approximately $0.72A$) to $9A$ in about 2.3 seconds ($0.018A/5ms$); then ramp the current down. When the current is ramping up, the left thigh motor is rotating in the positive counter-clockwise direction while the thigh is fixed at the hip joint limit. The left thigh torque is also positive, so the left thigh motor is in the forward-drive mode.

The actual experimental results are shown in the current-deflection curves in Fig. 2.12. The blue (lower) curve corresponds to the current ramp-up, while the red (upper) curve corresponds to the current ramp-down. The large relatively abrupt changes on both curves are actually due to the stick-slip in the gear. Since this is a quasi-static test, the gear performs differently from when KURMET is involved in a dynamic motion. The static friction does make it more difficult to analyze the test results.

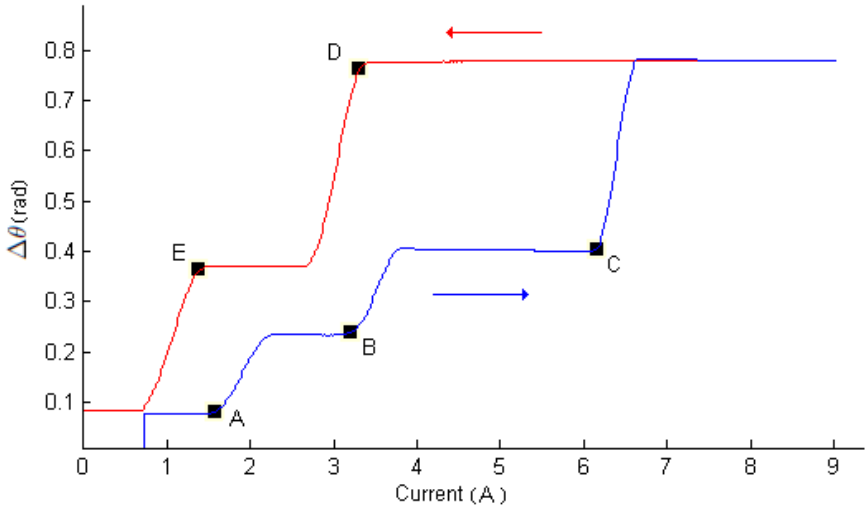

Figure 2.12: Current-deflection curve from the current ramp test.

Note point A on the blue curve in Fig. 2.12. At point A the gear is just about to break loose and switch from stick to the dynamic motion of slip. Let $\phi = \Delta\theta = \frac{\theta_{m,lt}}{n_m} - \theta_{lt}$. Then at point A

$$n_m \eta_f k_{\tau m} i_A = K_s(\phi_A - \theta_p) + \tau_p + \tau_{f,A} \tag{2.25}$$

where θ_p is the pretensioning deflection, τ_p the pretensioning torque and $\tau_{f,A}$ the maximum static gear friction at point A. Points similar to A may be observed on the curve (points B and C). Technically, the maximum static gear friction at each point maybe different. However, assuming that this difference is not a dominant factor in the calculation, points A, B and C are assumed to have the same static gear friction. This leads to

$$n_m \eta_f k_{\tau m} \cdot \Delta i = K_s \Delta\phi \tag{2.26}$$

where Δi and $\Delta\phi$ are the difference between two data points. From this equation the spring constant K_s may be calculated. The calculated value of K_s ranges from 32 to 36 Nm/rad. Considering the assumption made on the static friction and non-ideal conditions of the experiment, the estimate is quite close to the manufacturer specified value (within 10%), which indicates that it is valid to use the manufacturer's value (30 Nm/rad) for the spring constant in the model.

With the spring constant validated, the range for the backward gearbox efficiency (η_b) may also be estimated in the same way:

$$n_m k_{\tau m} \cdot \Delta i' = \eta_b K_s \Delta\phi' \ . \tag{2.27}$$

$\Delta i'$ and $\Delta\phi'$ are the difference between points D and E on the red (upper) curve. K_s is 30 Nm/rad (just validated). The backward gearbox efficiency is estimated to be 0.55, which is larger than the value used by Curran [14].

2.5.2 Model Calibration

The values of some other parameters in the KURMET model are still unknown. They are

USEA pretensioning torque (τ_p)
Unidirectional hardstop contact spring constant (K_c)
Unidirectional hardstop contact nonlinear damping coefficient (λ_c)
Ground contact spring constant (normal $K_{g,n}$ and tangential $K_{g,t}$)
Ground contact damping coefficient (normal $B_{g,n}$ and tangential $B_{g,t}$)
Static friction coefficient (μ_s)
Kinetic friction coefficient (μ_k)

These parameters do not have manufacturer specified values and are difficult to measure. Nominal values have been preliminarily suggested for these unknown parameters, especially for ground contact. However, it is desirable to find more proper values for these parameters so that the overall dynamic performance of the model may be similar to the physical system. A simple comparison test provides a way to quickly set these parameters.

In this comparison test, first on the physical machine, hold KURMET's torso at a nominal height of 60 cm. Then servo the motors so that both legs of KURMET have the following configuration: $\theta_t = -10^\circ$ and $\theta_s = 10^\circ$. Finally, drop the robot from the air with all the motors set to hold their current positions until it settles on the ground. Three key curves generated from the sensor signals during the process are shown in Fig. 2.13.

The same drop is also simulated using the KURMET model developed previously (initially, those unknown parameters take on some nominal values). The corresponding curves based on the simulated data are also plotted. If the simulated curve does not match its real counterpart very well, some of the unknown parameters need to be tuned (within a reasonable range).

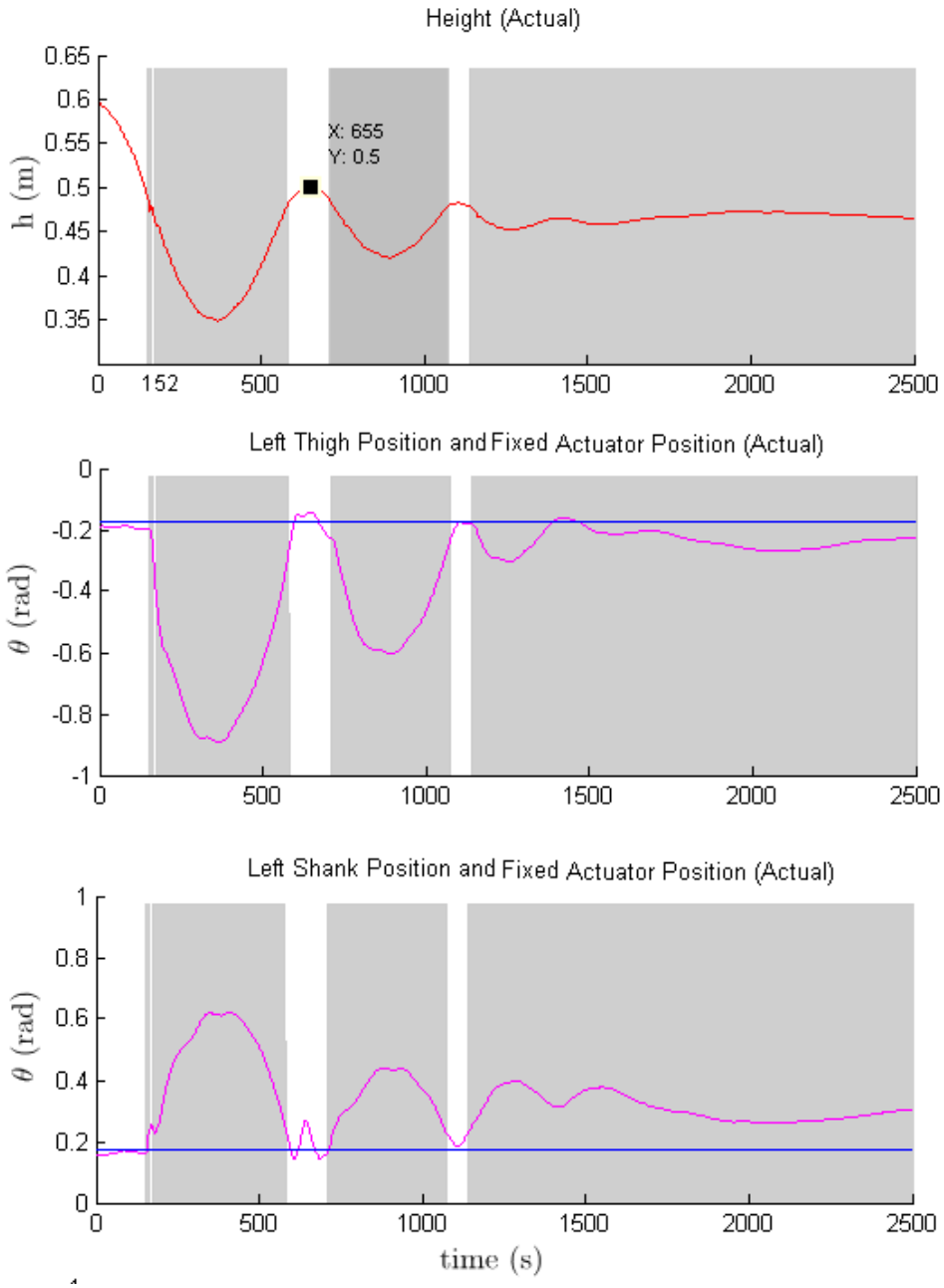

Figure 2.13: Curves from drop test on the physical machine (motor position fixed).

The following guidelines are very useful when tuning these parameters in simulation:

- The natural frequency of the system (i.e. inverse of the interval between two TOFs) is almost solely determined by the spring constant (K_s) in the USEA, and the ground normal contact spring constant ($K_{g,n}$) has little effect on the frequency.

- The height at TOF is mainly affected by both the touchdown configuration and ground friction coefficients (μ_s, μ_k). Change of the ground normal contact damping coefficient ($B_{g,n}$) will not significantly change the TOF height. Decreasing the ground friction coefficients will increase the TOF height. Along with that the whole system will take longer to settle down and three or even four flight periods may be observed.

- The pretensioning torque (τ_p) roughly decides the static deflection between the link and the motor after KURMET settles on the ground.

- The unidirectional hardstop contact spring and damper (K_c, λ_c) can alter the maximum deflection between the link and the motor after the second touchdown and the penetration depth during link and motor impact at life-off.

The first guideline is reasonable. During the drop test, KURMET as a whole acts like a mass on a spring ('KURMET spring'). The ground normal contact spring is much stiffer than the 'KURMET spring' for reasonable touchdown configurations. On touchdown, the two springs are serially connected, so the overall stiffness is dominated by the stiffness of the 'KURMET spring'.

The second guideline need some additional explanation. The effect is due to an interesting condition of the experimental system: The biped and its feet are constrained by the boom to move on a spherical surface and this causes lateral slip at the feet as the legs compress/extend during contact. So during the simulated tests, the feet are always sliding on the ground, no matter how large the ground friction coefficients are set (for example, even if $\mu_s = 10$, $\mu_k = 9$). As a check, it may be noted that for the simulated contact force on the foot, the tangential contact force (f_t) is equal to the normal contact force multiplied by the kinetic coefficient ($\mu_k \cdot f_n$), which indicates sliding. Since the biped feet are sliding anyway, and f_t is forced by the simulator to equal $\mu_k \cdot f_n$, if the kinetic friction coefficient μ_k becomes larger, the system will have to overcome a larger tangential force to slide. Therefore more energy will be lost during the contact. This is true in the simulation; however it might not always be the case in the physical system.

The "Simulate-Compare-Tune-Simulate" process may be repeated several times until a best matching model is found. Figure 2.14 shows the same trajectories, as in Fig. 2.13, except generated from simulated data after the model is calibrated. The final values of the parameters used in the model are listed in Tables A.7 and A.10 in Appendix A.

It may be observed that, even after the model is calibrated, the simulated trajectories still have some mismatches in comparison with the real curves. (Two examples of such places are marked with A and B in Fig. 2.14.) Tuning the above parameters could not account for these mismatches. Actually, all the unsatisfactory behaviors in simulation are related to contact with a surface. It implies that the ground contact

model (including the sliding model[1]) and the unidirectional hardstop contact model are perhaps too simple to capture all the dynamic features of the physical system. However, the overall model is good enough to serve as a foundation upon which the fuzzy control of the next chapter may be developed.

2.6 Summary

In summary, Chapter 2 serves as a foundation for the following chapters. It first introduced KURMET, the bipedal robot system to which the control strategy developed in the later chapters are targeted. An overview of both the mechanical and the electrical subsystems were given. The articulated-body model for the mechanism was developed, followed by a model for the USEA and model for ground contact. Simulation methods using *RobotBuilder* were described. Finally, two experimental tests were discussed: one is aimed at verifying the spring constant; another evaluates the overall dynamic performance of the model through comparison with the results from the experimental system during a drop test. The next chapter will present the hopping control strategy designed for KURMET, the system model developed in this chapter is a key to the successful application of the control to the physical system.

[1]The simulator's sliding model is also problematic, when a contact point starts to slide, the simulator automatically resets the position of its anchor point to be above the current position of the contact point.

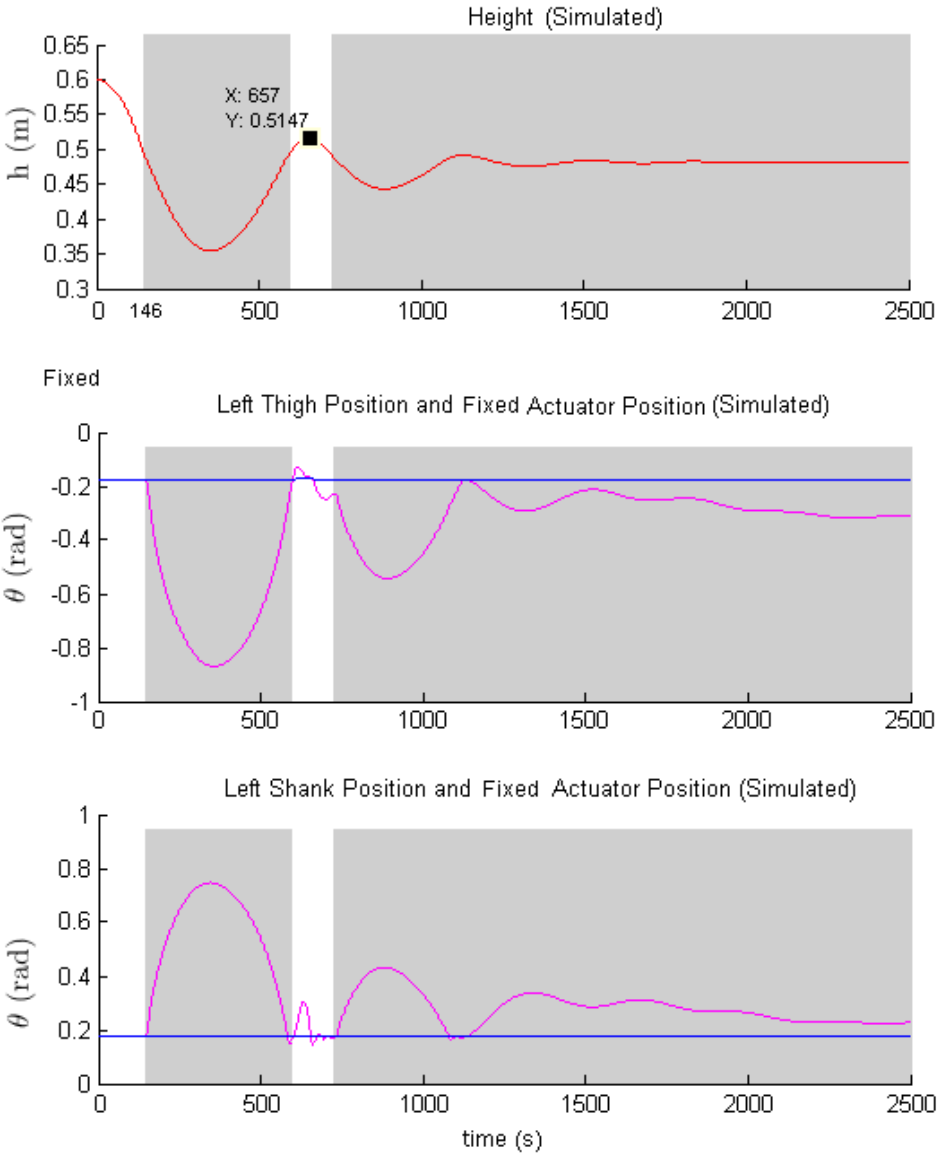

Figure 2.14: Curves from simulated drop test (motor position fixed).

CHAPTER 3

FUZZY CONTROL OF HOPPING

3.1 Introduction

The goal of this chapter is to develop a stable hopping control for KURMET, the biped robot. With the control, KURMET can perform multiple hops each at a designated height and velocity. If a hop is defined as a hopping cycle that starts from a top of flight (TOF) and ends in the next TOF, then the height of a hop is characterized by the torso height at the ending TOF, and the velocity of a hop is characterized by the average horizontal torso velocity from the lift-off to the ending TOF[1]. The formulas used to calculate the torso height and horizontal velocity are given in Appendix C.

The control has two parts: a low level hopping controller and a high level fuzzy controller. The low level hopping controller is designed specifically to accommodate the natural dynamics of the USEAs. It consists of a hopping state machine which decomposes a hop motion into a set of distinct sequential states, and different motor position commands are specified for each of the states. The fuzzy controller supervises

[1]The average velocity from lift-off to TOF is less sensitive to the perturbations caused by the oscillations in the legs and is physically more meaningful than the instantaneous velocity at TOF.

the low level hopping controller. While the low level hopping controller is active at every control step, the fuzzy controller is only called once per hopping cycle at TOF. Once called, the fuzzy controller monitors certain current system state variables and determines the key parameters in the hopping state machine for the next hop. It can be interpreted that the high level control 'plans' a hop, and the low level control actually executes the motion. The fuzzy controller needs to address the complex nonlinearity between the input parameters and the resulting motion as well as the hybrid contact/flight dynamics of the robot. This layered control approach has been used by Marhefka [16], Palmer [21] and Hester [25] and proved to be effective to control dynamic movements with legged machines. This controller is later enhanced through the on-line adaptation process to achieve a better performance on the physical machine. This chapter starts with a description of the implementation of the low level hopping control for KURMET. The inputs and outputs of the fuzzy controller, which are closely associated with the low level hopping control, are subsequently introduced. The section following gives detailed examination of the internal structure of the fuzzy controller. A training process that is used to initialize the fuzzy rulebase for the controller is then described. The method for on-line adaptation is also provided. The next chapter will present the control results from both simulation and experimental tests.

3.2 Low Level Hopping Control

KURMET's low level hopping control is the foundation that ensures effective functioning of the fuzzy controller. It is actually a set of simple but event-sensitive motor position commands that control the legs of the biped robot to achieve a desired

hop. Timing is very important for the low level control, which suggests that the state machine is a very convenient way to coordinate these motor position commands. The hopping state machine decomposes a hop motion into a sequence of discrete and concatenated control states. Each state has its own specific motor position commands and unique exit condition. Research in neuromotor physiology also indicates that the central nervous system of animals uses a linear combination of a number of fixed-pattern muscle synergies to generate a variety of complex movements [34], which lends more credibility to using modular approach for the low level hopping control, such as the state machine. The modular nature of the state machine also simplifies the control design process, allowing the designer to focus on only one control state at a time. The state machine also makes it easier to change or expand the low level control at later point.

3.2.1 Hopping State Machine

Since the hopping state machine will ultimately work with the physical machine, its robustness becomes a very important issue. Robustness includes several aspects:

- Each control state in the state machine monitors certain sensory signals to detect the occurrence of a specific event; however, the actual sensory signals are usually not ideal due to the noise in the sensor circuits and/or the error caused by the play and backlash in the mechanical parts. It is important to find the right triggering event so that the trigger of a state is not very sensitive to the noise in the sensory signals.

- The goal in this chapter is to control the hopping of KURMET at varying height and velocity. It is very likely that a hopping state machine works perfectly when

the biped robot is hopping in place but performs poorly or even fails during high-speed hopping. It is desirable to have one universal state machine that can handle all the different hopping conditions nicely; otherwise the upper-level fuzzy controller may not be able to function properly.

- A well-designed state machine should also take the most advantage of the natural dynamics of the system and be as simple as possible. To reduce any unwanted oscillations in the legs, the state machine should generally avoid commanding a large motor position step in a very short time.

With the above considerations, the following state machine has been proposed and is later shown to be robust. Before the description is given, the points below should be noted:

- The left and right legs are synchronized at all times. The state machine controls both legs as one.

- The hopping state machine only issues motor position commands, and it always keeps the motor positions under closed-loop control. It never gives open-loop motor current commands.

- The hopping state machine never tries to control the link positions. Attempts to control the link positions will be very difficult and oppose the natural dynamics of the system due to the application of series-elastic elements in the actuators.

- Whenever a new motor position is commanded, a cubic-spline desired trajectory will be automatically generated for the transition (see Appendix B). A

trajectory will be prematurely terminated if a new motor position command is issued before the current trajectory is finished.

- The hopping state machine is cyclic to accommodate the periodic nature of repetitive hopping.

The hopping state machine defines the motor actions for each state in a hopping motion from TOF to TOF. Figure 3.1 shows all the states and transitions between those states. Figure 3.2 shows the typical motor/link position trajectories (left leg) in one hop from TOF to TOF as partitioned by the control states. Figure 3.3 shows the associated torso height/horizontal velocity trajectories in the same time period, also partitioned by state.

The hopping state machine starts from the **DROP** state to initiate a hop. In this state the experimental biped is initially held in the air with a certain torso height[1] and no torso velocity. The legs are positioned in a static initial configuration. The biped may be considered to be at a motionless TOF. At some moment, the biped is released in the air. As soon as the vertical velocity of the torso becomes negative, the **PRE_TD** state is entered.

Once the **PRE_TD** state is entered, the motors start to servo the links to the desired touchdown configuration. The touchdown configuration is characterized as the desired virtual leg angle θ_{vl} and the link deflection θ_{defl}. The virtual leg is the imaginary line segment between the hip joint and the distal end of the shank, and its angle relative to the vertical direction is called the virtual leg angle. In this state machine, θ_{vl} is a function of the hop height and velocity of the last hop[2]. The

[1]The height should be in the tractable range of the fuzzy controller; see Section 3.3.

[2]If the control state before **PRE_TD** is **DROP**, then the last hop height is the drop height, and the last hop velocity is zero.

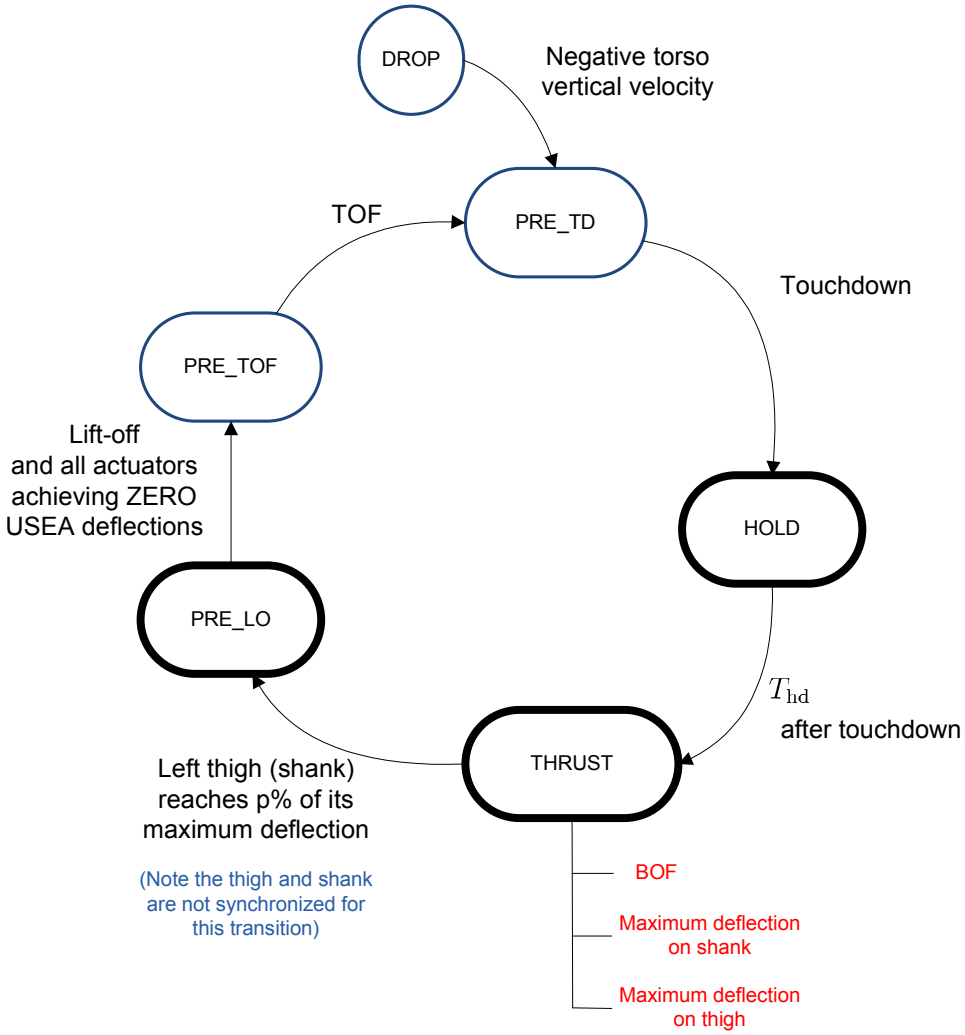

Figure 3.1: Diagram of the hopping state machine.

link deflection is the absolute value of the link position relative to the virtual leg angle, and the thigh and shank always have the same deflection since the links are of equal length(Fig. 3.4). Upon entering the **PRE_TD** state, the thigh motor position command, $(\theta_{\mathrm{vl}} - \theta_{\mathrm{defl}}) \cdot n_m$, and the shank motor position command, $(\theta_{\mathrm{vl}} + \theta_{\mathrm{defl}}) \cdot n_m$, are issued, where n_m is the gear ratio. The desired trajectory time is $T_{\mathrm{pr\_td}}$, meaning that

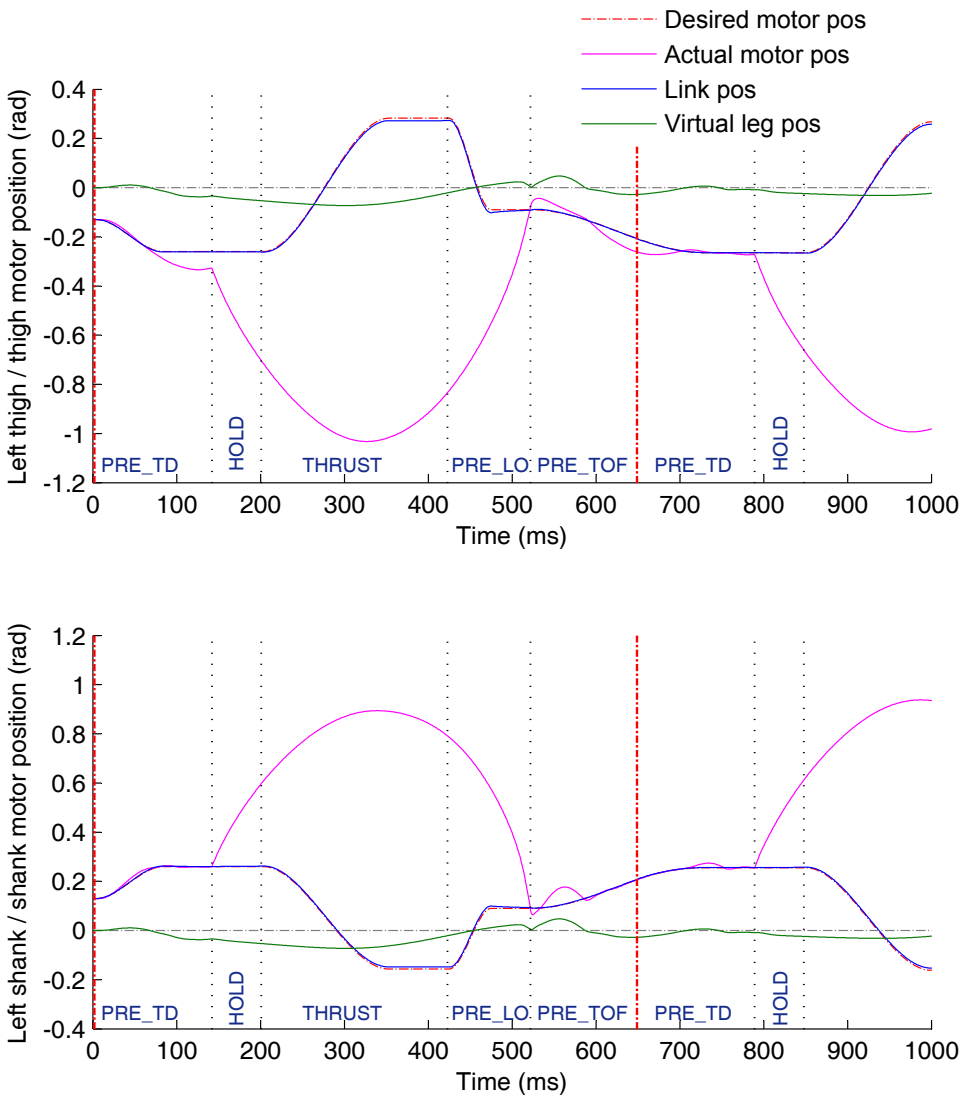

Figure 3.2: Typical motor/link trajectories (left leg) in one hop from TOF to TOF, partitioned by the system states. The motor position is on the link side of the gear.

the thigh/shank motors should arrive at the commanded positions $T_{\text{pr\_td}}$ time after the position commands are sent. The trajectory should be finished before touchdown.

Immediately after the left foot touches down, the state machine transitions to the **HOLD** state. All motors simply hold their current positions (i.e. the desired

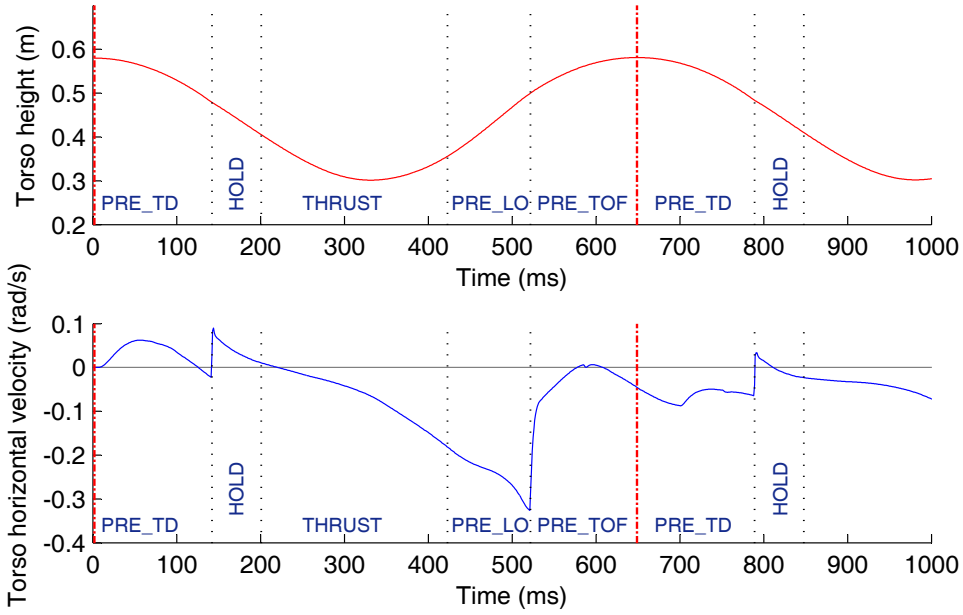

Figure 3.3: Typical torso height/horizontal velocity trajectories in one hop from TOF to TOF, partitioned by the system states.

touchdown positions) for T_{hd} time. The system starts to squat and the leg links are deflected more as the body height going down.

The **THRUST** state begins T_{hd} time after entering the **HOLD** state at touchdown. In this state, energy is injected into the system by driving the motors to new positions in a *timely* manner to cause an extra[1] amount of deflections in the springs in the USEAs. Such an action is called a 'thrust'. The parameters of a thrust include the thrust angle for the thigh motor, $\theta_{\text{thr},t}$, the thrust angle for the shank motor, $\theta_{\text{thr},s}$ and the thrust time T_{thr}. The thrust angle is the absolute value of the change of the motor position during a thrust (Fig. 3.5). The thrust time is just the desired trajectory time. At the beginning of the **THRUST** state, the thigh motor starts to

[1]This is in addition to the deflection increase caused by the natural dynamics.

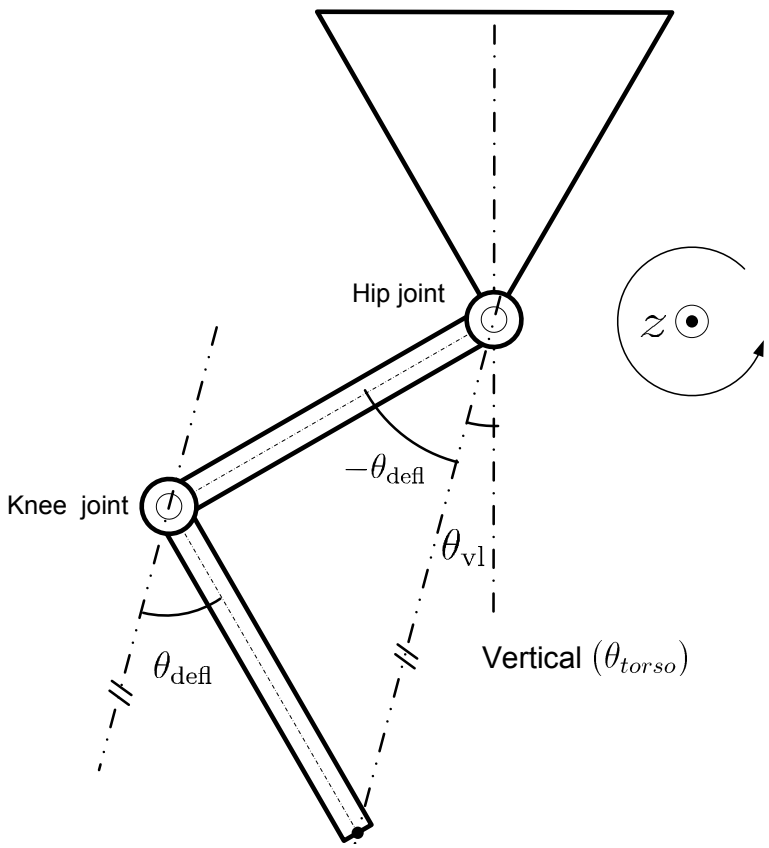

Figure 3.4: Virtual leg and link deflection angles.

move towards position $\theta_{m,t} + \theta_{\text{thr},t}$ and at the same time, the shank motor starts to move towards position $\theta_{m,s} - \theta_{\text{thr},s}$, where $\theta_{m,t}$ and $\theta_{m,s}$ are the actual thigh/shank motor positions when the **THRUST** state begins. The trajectories for all the motors are supposed to be finished before the **THRUST** state is terminated. Both bottom of flight (BOF) and the maximum deflections of the links happen during this state.

After the thrust is given, leaving the motors at their thrust positions waiting for lift-off tends to hyper-extend the legs and probably damage the knee joint. It is important to bring the motors back to the safe side of the leg singularity before the

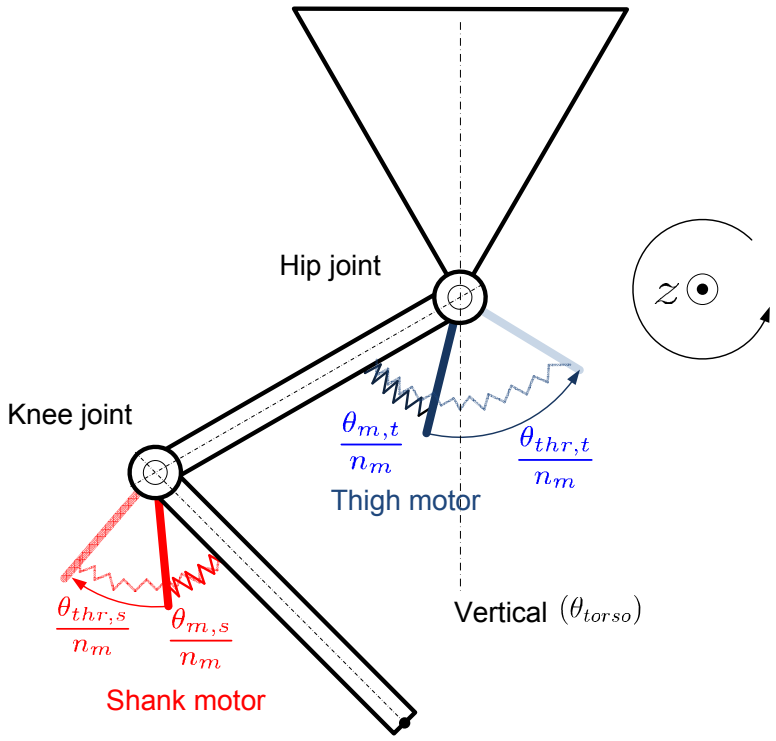

Figure 3.5: Thrust angles for the hip and knee joints.

shank hits hard on the singularity hardstop (Fig. 3.6). During the **THRUST** state, the deflections of links on the left leg are monitored. Once the current deflection of the thigh (shank) link is smaller than $p\%$ of its maximum deflection, the thigh (shank) motor enters the **PRE_LO** state from the **THRUST** state. It should be noted here that usually the thigh and shank arrive at the $p\%$ of their respective maximum deflections at different times, thus the thigh and shank motors enter the **PRE_LO** state at *different* times, too. This is different from the previous states, where the thigh and shank motors switch states at the same time.

When the thigh motors enter the **PRE_LO** state, it is immediately commanded to go to position $-\theta_{\text{pr\_lo}} \cdot n_m$. When the shank motors enter the **PRE_LO** state,

the immediate motor position command is $\theta_{\text{pr\_lo}} \cdot n_m$. The trajectory time for both the thigh and shank motors are $T_{\text{pr\_lo}}$. In the time period shortly before lift-off, the body height is increasing and the links are hurtling towards the motors unidirectional hardstop, the allowed $T_{\text{pr\_lo}}$ is very short, and the motors have to accelerate, then decelerate and finally come to a full stop at the desired positions just before the links slam into the motors so that the oscillations on the links can be minimized. The **PRE_LO** state demands high agility of the motors and thus, large amounts of supply current. The motors will see an especially heavy load during the deceleration period. Figure 3.7 shows the situation when the supply current is insufficient during the **PRE_LO** state.

After all links have hit the motors (all actuators achieving zero USEA deflections) respectively and both feet are off the ground, the link motors enter the **PRE_TOF** state. Upon entering this state, the thigh and shank motors are commanded to go to their last pre-touchdown positions. However, the trajectory time, $T_{\text{pr\_tof}}$, is long enough so that the motors will only be approximately half way toward their commanded positions when the biped reaches TOF.

Immediately after TOF, the **PRE_TD** state is entered again. The desired touchdown virtual leg angle is also updated based on the hop velocity and height just achieved at TOF. The newly commanded touchdown motor positions will terminate the old unfinished motor trajectories from the last state, and generate new trajectories for the motors from their current positions. The state machine then repeats for the next hop. Table 3.1 summarizes the motor actions taken during each state and the trigger to exit each state. The details behind the triggers, such as how TOF and touchdown are detected in the real system, can be found in [9].

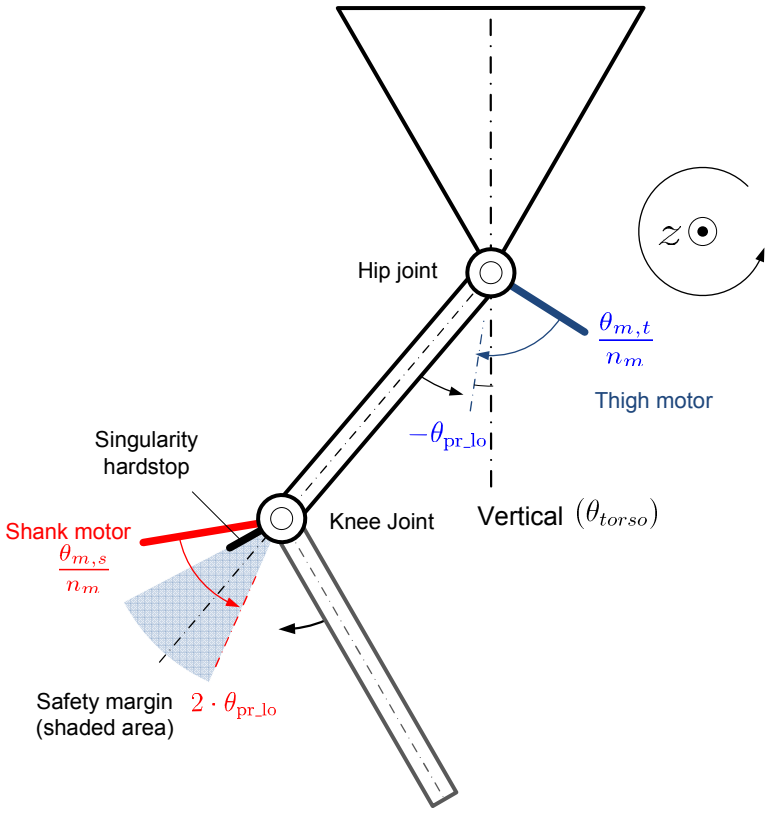

Figure 3.6: The safety consideration in the **PRE_LO** state. The shank motor should always get back to the 'safe' side of the singularity hardstop in time during the **PRE_LO** state so that the shank will not hit impact the singularity hardstop.

3.2.2 Actual Parameters for the Hopping State Machine

Section 3.2.1 has provided a behavioral-level description of the hopping state machine. However, in order to put it into practical use, the many parameters in the state machine need to be supplied with proper values. The parameter values listed in Table 3.2 are the ones that are actually used by KURMET. Simulation and many actual tests have been performed to refine these values. The state machine, as applied with

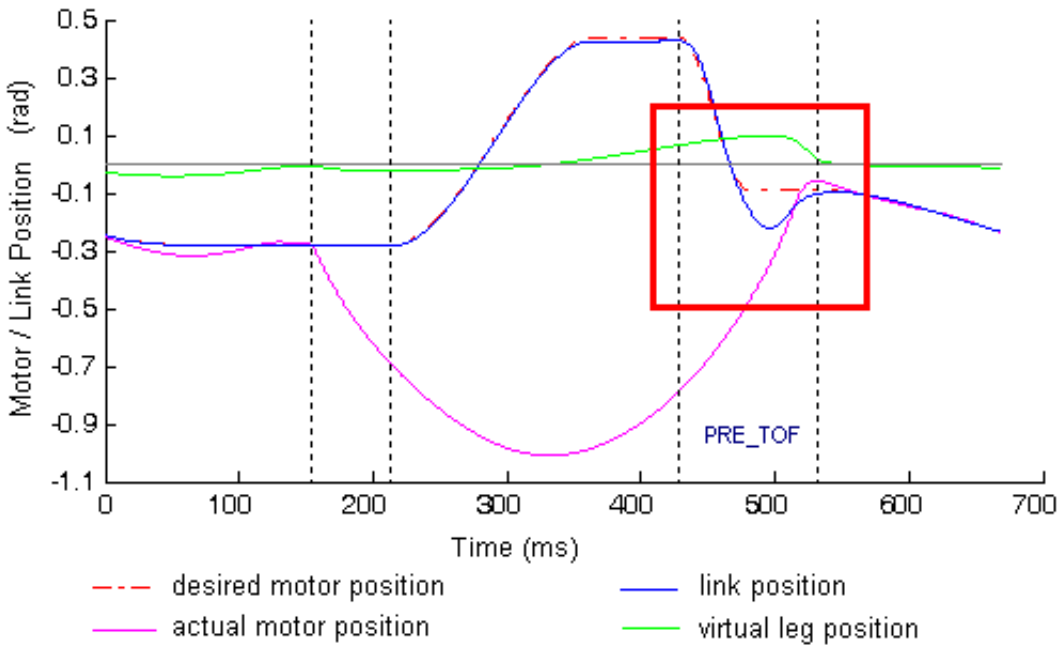

Figure 3.7: Thigh motor position overshoot due to insufficient supply current during the **Pre_LO** state.

the listed parameter values, has been shown to be effective and robust for KURMET. The following guidelines were used in the process of tuning these parameters:

- The desired touchdown virtual leg angle θ_{vl} is a function of the hop velocity and height of the last hop and is designed to assist the biped robot to achieve the desired hopping velocity. The heuristics behind the formula are inspired by Mark Raibert's foot placement algorithm for a one-leg hopping machine [1] and has been modified for KURMET.

- The desired touchdown link deflection, θ_{defl}, proves to be a very sensitive factor that affects the overall hopping performance. The results, from a series of zero-thrust drop tests with different touchdown link deflection ranging from

Table 3.1: Hopping state machine summary

State	**Motor actions**	**Exit trigger**
DROP	Both the thigh and shank motors hold their positions.	Negative torso vertical velocity
PRE_TD	The thigh motors are driven to position $(\theta_{\text{vl}} - \theta_{\text{defl}}) \cdot n_m$, and the shank motors are driven to position $(\theta_{\text{vl}} + \theta_{\text{defl}}) \cdot n_m$, both in $T_{\text{pr\_td}}$ time.	Touchdown
HOLD	Both the thigh and shank motors hold their positions.	T_{hd} time after Touchdown
THRUST	The thigh motors are driven to $\theta_{m,t} + \theta_{\text{thr},t}$, and the shank motors are driven to $\theta_{m,s} - \theta_{\text{thr},s}$, both in T_{thr} time.	The thigh (shank) motors exit the **THRUST** state when the thigh (shank) deflection is less than $p\%$ of the maximum deflection of the left thigh (shank).
PRE_LO	The thigh motors are driven to $-\theta_{\text{pr\_lo}} \cdot n_m$, and the shank motors are driven to $2\theta_{\text{pr\_lo}} \cdot n_m$, both in $T_{\text{pr\_lo}}$ time	Lift-off and all actuators achieving zero USEA deflections.
PRE_TOF	The thigh and shank motors are driven to their previous desired pre-touchdown position in $T_{\text{pr\_tof}}$ time.	TOF

0.05 to 0.35 *rad*, have shown that as θ_{defl} becomes smaller (i.e. as the legs become more singular), the second TOF height becomes lower; meanwhile, the

Table 3.2: Actual parameter values for KURMET's hopping state machine

State	**Parameter values**
DROP	-
PRE_TD	$\theta_{\text{vl}} = \frac{v}{0.08} \cdot 0.12 \cdot \frac{1-(h-0.58)}{0.8}$, where h (m) is the last hop height and v (m/s) is the last hop velocity $\theta_{\text{defl}} = 0.26\ rad$ $T_{\text{pr\_td}} = 80\ ms$
HOLD	$T_{\text{hd}} = 60\ ms$
THRUST	$T_{\text{thr}} = 150\ ms$ $\theta_{\text{thr},t}$ and $\theta_{\text{thr},s}$ are determined by the fuzzy controller; see Section 3.3 $p = 85\%$
PRE_LO	$\theta_{\text{pr\_lo}} = 0.09\ rad$ $T_{\text{pr\_lo}} = 50\ ms$
PRE_TOF	$T_{\text{pr\_tof}} = 200\ ms$

system tends to behave in a 'stiffer' manner (especially when θ_{defl} is smaller then 0.2 rad). This indicates that a more singular touchdown configuration would cause more energy loss at touchdown impact. Generally, a larger θ_{defl} (a less singular touchdown configuration) can better utilize the springs in the

USEA elements to buffer the touchdown impact, and therefore, tends to give more springiness and controllability to the system.

- The touchdown link deflection, θ_{defl} and the hold time, T_{hd}, together determine the maximum link deflections. Larger θ_{defl} and longer T_{hd} can increase the maximum link deflections, which usually have positive effects on the overall hopping performance. However, at the same time the links also tend to hit the joint limits more easily when the torso height approaches bottom of flight (BOF). These two variables should be chosen carefully so that the links can deflect as much as they can but the maximum link deflections never exceed the mechanical limits.

- The hold time T_{hd} also delays the thrust. If there is no hold time and the thrusts are applied immediately after touchdown, the thrusts tend to prevent the legs from full deflections and the process becomes inefficient, since a good portion of the thrust is spent to oppose the natural dynamics of the spring, instead of injecting extra energy into it. It should be noted that the motor positions alone cannot account for the energy injection – it also depends on the link positions. However, since the link positions cannot be controlled directly due to the series-elastic elements in the actuators, it is important to time the motor thrust so it can utilize the natural dynamics to achieve the maximum efficiency. Generally the motor thrusts are more efficient when the link rates are relatively slow. Such a period happens shortly before and after BOF of a hop. Therefore the T_{hd} and T_{thr} are designed to *approximately* fit the motor thrusts into that period so that the motor thrusts can effectively increase the

maximum deflection in the springs in the series-elastic elements. The T_{thr} is usually kept short but should also ensure enough trajectory time for the motor to avoid over-acceleration.

- The **PRE_LO** state is a state that trades off system energy for system safety. The exit trigger for the **THRUST** state, $p\%$ of the maximum deflection, and the pre-liftoff trajectory time $T_{\text{pr\_lo}}$ dictate when and how fast the energy is removed from the system. As long as the shank can return to the safe side of the singularity in time, it is desirable to delay the **PRE_LO** state and reduce the $T_{\text{pr\_lo}}$ time as much as possible to minimize the energy loss. The motor current limitation turns out to be a major consideration for choosing these two state machine variables.

In this hopping state machine, in most cases a desired motor trajectory is required to be finished before the state switches to the next state. However, the **PRE_TOF** state is an exception. Previously, in the **PRE_TOF** state, the motors were only required to hold their positions until TOF. Then in the following **PRE_TOF** state, the motors started to servo the link to the desired touchdown configuration (Fig. 3.8 (a)). Since $T_{\text{pr\_td}}$ is short and the desired motor position changes for that period are usually large, the motors accelerate and decelerate rapidly. Due to the series-elastic elements, quickly altering the motor positions with no loads on the legs inevitably causes substantial link position overshoots.

Therefore, to reduce the overshoot, the pre-touchdown trajectories start early at the beginning of the **PRE_TOF** state (Fig. 3.8 (b)). In such a way the motors have more time to go through the same position changes, and therefore be able to servo the links at a relatively slow rate and reduce the overshoot. A technical

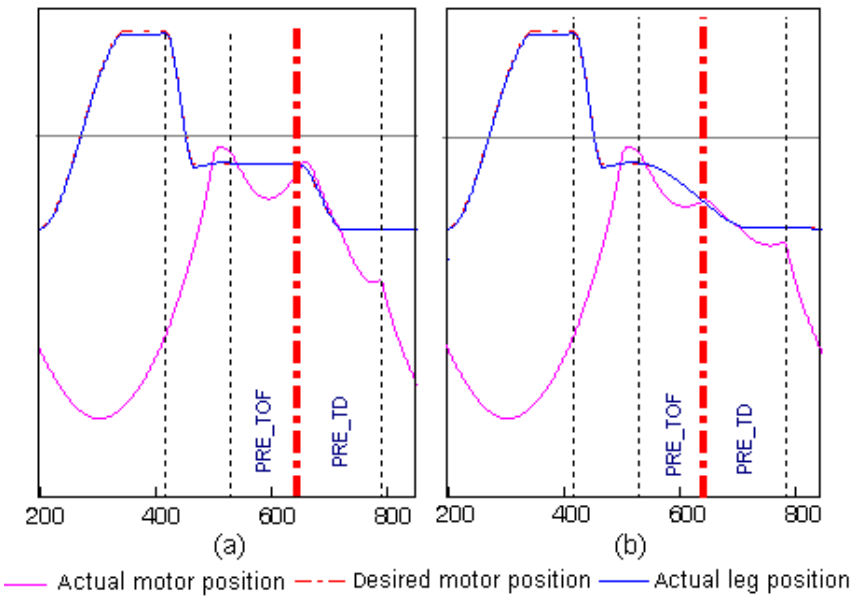

Figure 3.8: Comparison of two different **Pre_TOF** to **Pre_TD** motor position trajectories. Note the change in the desired motor position is commanded before **PRE_TD** in (b).

problem is: the desired touchdown virtual leg position for the next hop is unknown until TOF. To address this problem, at the beginning of the **PRE_TOF** state, the old desired touchdown motor positions are commanded as a temporary substitute, since the desired touchdown virtual leg positions are relatively *close* between two adjacent hops. The 'old' trajectories can lead the motors to move towards the next desired touchdown positions even before TOF is reached. Then at TOF, the motor trajectories are updated and lead to the exact new touchdown positions. In particular, the updated trajectories will not be very much different from the old ones.

3.3 Fuzzy Controller Inputs and Outputs

Choosing the *right* fuzzy controller inputs and outputs, especially the outputs, is a very important step to effectively integrate the fuzzy controller with the low-level hopping state machine. A set of properly chosen fuzzy controller inputs and outputs not only can improve the accuracy, robustness and safety of the overall controller but also can greatly facilitate the training process to initialize the rulebase.

The inputs to the fuzzy controller are relatively easy to choose since they are the direct reflections of the control objectives. They are: the actual hop height at the current TOF, h_o, the actual hop velocity at the current TOF, v_o, the difference between the actual hop height at the current TOF and its desired value for the next TOF, Δh_d, and the difference between the actual hop velocity at the current TOF and its desired value for the next TOF, Δv_d.

The choice of the fuzzy controller outputs needs more consideration. First experience has been borrowed from Hester's work [25]. His focus of work was to develope a fuzzy jumping controller for KURMET. It employed closed-loop motor position control in flight and used open-loop currents to provide the thrust during ground contact. However, two weaknesses of using open-loop current control, rather than closed-loop position control, for thrust were revealed:

- To safely and effectively control the USEA elements, the controller must carefully manipulate the motor positions so that the joint positions can always be constrained within its mechanical limits. This is especially important during the period right before lift-off. However it is difficult for the controller to enforce

these constraints with open-loop currents, since it cannot actively regulate the actual motor positions. Therefore considerable risk to damage the joints exists.

- The effect of the open-loop thrust is heavily influenced by the forward-drive/back-drive model of the gear in simulation and thus more sensitive to modeling errors. Considerable tuning is required to make the controller really work on the physical machine. The closed-loop thrust, on the other hand, would hardly be affected by the inaccuracy of the forward/backward efficiency of the gear.

Given the two points above, the fuzzy controller developed in this thesis avoids outputting open-loop current commands and the low-level hopping state machine is designed correspondingly to accommodate closed-loop motor position control (already introduced in the previous section).

Another challenge faced by Hester's controller was due to the number of fuzzy controller outputs (there are six of them in total). The non-minimal selection of the number of fuzzy outputs gives extra flexibility to the controller and can cause the training to converge to different modes of operation. If multiple modes of operation exist in the fuzzy rule base, the controller may provide poor outputs when inferring between rules. Sophisticated training laws can help but will not be able to fundamentally solve this problem. This experience has suggested that the number of fuzzy controller outputs should be kept as small as possible and only those that have the most direct influence on a control objective are included.

In an unsuccessful early attempt, the desired touchdown virtual leg angle θ_{vl} and the thrust angle θ_{thr} (thigh and shank motors have the same thrust angle) were chosen as the outputs for the fuzzy controller developed in this thesis. The output θ_{vl} was targeted to regulate the hopping velocity and the output θ_{thr} was targeted to maintain

the hopping height. These two outputs worked OK with the low-level hopping state machine for the lower-speed cases. However, when the hopping velocity increased, the output θ_{thr} (i.e. the same thrust on both thigh and shank motors) started to counteract the output θ_{vl} and impede the experimental biped from getting the desired hopping velocity. Moreover, the output θ_{vl} is very difficult to be realized accurately on the physical machine due to the oscillations of the unloaded links, introducing more uncertainties to the control performance.

Synthesizing all the experience above, the final fuzzy controller outputs selected are: the **common-mode thrust angle**, $\theta_{\text{thr,comm}}$, and the **differential thrust angle**, $\theta_{\text{thr,diff}}$. These two fuzzy controller outputs are then converted to the more straightforward motor position commands: thigh and shank motor thrust angles, $\theta_{\text{thr,t}}$ and $\theta_{\text{thr,s}}$, through the following equations:

$$\theta_{\text{thr,t}} = \theta_{\text{thr,comm}} + \theta_{\text{thr,diff}} \ , \tag{3.1}$$

$$\theta_{\text{thr,s}} = \theta_{\text{thr,comm}} - \theta_{\text{thr,diff}} \ . \tag{3.2}$$

The output $\theta_{\text{thr,comm}}$ corresponds mainly to the hop height and the output $\theta_{\text{thr,diff}}$ corresponds mainly to the hop velocity. Instead of being a fuzzy controller output, the desired touchdown virtual leg angle θ_{vl} is approximated by a linear function reflecting certain heuristics to assist $\theta_{\text{thr,diff}}$ to achieve hop velocity control. This selection keeps the number of fuzzy controller outputs minimum but at the same time ensures enough flexibility for the controller to reach its objectives. Both outputs can be realized physically with adequate accuracy and are robust to modeling errors, and they are shown to be only weakly coupled.

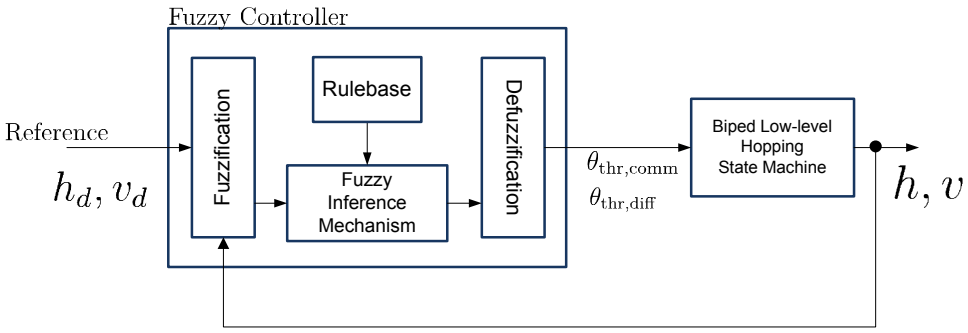

Figure 3.9: Structure of the fuzzy controller.

3.4 Fuzzy Controller Structure

The fuzzy controller essentially provides a mapping between its inputs and outputs. When invoked at the TOF of a hop, it interprets the current controller inputs and consults its knowledge base to determine the $\theta_{\text{thr,comm}}$ and $\theta_{\text{thr,diff}}$ that are to be used by the hopping state machine in the next hop. It is then deactivated until the next TOF. Figure 3.9 illustrates the structure of the fuzzy controller. It is comprised of four functional components: the fuzzification interface, the fuzzy rulebase, the inference mechanism and the defuzzification interface.

The fuzzification interface is a pre-processor of the inputs. The fuzzification interface converts a specific input value into certainties of input membership functions. For simplicity, in this work, identical, equilateral-triangular input membership functions are used for all inputs. The membership function centers (where the certainty of a membership function is 1) are equally-spaced and the certainty of a triangular membership function becomes zero right at the centers of its two neighboring membership functions, so that a membership function will never overlap the centers of

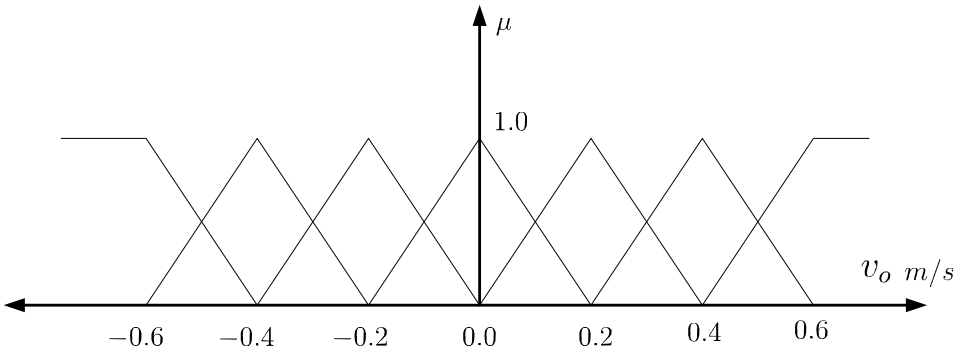

Figure 3.10: Input membership functions for input v_o. The outer most membership functions saturate at 1.0 when $|v_o| > 0.6$.

Table 3.3: Fuzzy controller input membership function centers

Input	**Membership function centers**	Units
h_o	0.54, 0.58, 0.62	m
v_o	-0.6, -0.4, -0.2, 0.0, 0.2, 0.4, 0.6	m/s
Δh_d	-0.02, 0.0, 0.02	m
Δv_d	-0.2, 0.0, 0.2	m/s

other membership functions. Due to this setting, at any given time, any input value will only activate at most two membership functions (i.e. at most two membership functions have non-zero certainties). This reduces the computational complexity for the following process. Figure 3.10 shows the membership functions used to characterize v_o. The membership function centers for all the fuzzy controller inputs are listed in Table 3.3. For example, if the current hop velocity v_o is 0.1 m/s, then the certainties $\mu_{0.0}^{v_o} = \mu_{0.2}^{v_o} = 0.5$, while the certainties of all other membership functions are zero.

The fuzzy rulebase provides control knowledge of the system. It contains a set of rules which specifies the controller outputs for every combination of input membership functions. In this work, the inputs h_o, Δh_d and Δv_d have 3 membership functions each and the input v_o has 7 membership functions so there are $3 \times 7 \times 3 \times 3 = 189$ rules. Each input membership function is represented by the input value at its center[1], so an example rule looks like the following:

Rule **2422**
If $h_o = 0.58\ m$, $v_o = 0.0\ m/s$, $\Delta h_d = 0.0\ m$ and $\Delta v_d = 0.0\ m/s$,
then $\theta_{\text{thr,comm}} = 61.8\ rad$ and $\theta_{\text{thr,diff}} = 7.4\ rad$.

Here 2422 is the unique index of this rule. It is a convenient notation since $0.58\ m$ is the center of the 2^{nd} membership function of h_o, $0.0\ m/s$ is the center of the 4^{th} membership function of v_o, etc. The 'if' part of the rule is called the premise and the 'then' part is called the consequence. A combination of the input membership function centers is thus also referred to as a rule center.

The fuzzy rulebase only specifies the controller outputs for certain input cases. The controller outputs for more general input cases are inferred from the existing rules. The inference mechanism is the component that determines the applicability of each rule to the current set of inputs. In this work, the applicability of a rule, or the certainty for the premise of the rule, is calculated as the product of the certainties of the input membership functions. For example, the certainty of rule 2422 would be calculated as

$$\mu_{2422} = \mu_{0.58}^{h_o} \cdot \mu_{0.0}^{v_o} \cdot \mu_{0.0}^{\Delta h_d} \cdot \mu_{0.0}^{\Delta v_d} \ . \tag{3.3}$$

[1]It does not necessarily have to be the case, though.

Due to the fact mentioned above, that for each of the 4 inputs, only a maximum of two membership functions will be activated at a time (have non-zero certainties), at most $2^4 = 16$ rules can have non-zero certainties for their premises during the inference, which means there are only 16 applicable rules at most for any possible set of inputs, i.e. the control knowledge for any arbitrary set of inputs can be inferred from a maximum of 16 rules instead of the whole rulebase. The reduced number of the potentially applicable rules minimize the computation needed by the inference mechanism and defuzzification. Moreover, it is only a function of the number of the inputs, adding more membership functions to an input will not affect the amount of computation.

The defuzzification interface then converts the conclusion of the inference mechanism into the final fuzzy controller outputs. Since this is a zero-order Sugeno type fuzzy controller (the consequences of a fuzzy rule are constant singletons), the outputs are calculated as the weighted average of the consequences of all the applicable rules:

$$y_i = \frac{\sum_{z \in S} \mu_z \cdot u_{z,i}}{\sum_{z \in S} \mu_z} = \sum_{z \in V} \mu_z \cdot u_{z,i} \ , \tag{3.4}$$

where z is the rule index and S is the set of the indices of all the applicable rules. The subscript i can be 1 or 2, $u_{z,i}$ is the singleton value of the output for rule z, y_1 is the controller output $\theta_{\text{thr,comm}}$ and y_2 is the controller output $\theta_{\text{thr,diff}}$. More theoretical and practical details about the fuzzy controller can be found in [35, 36].

3.5 Fuzzy Controller Training

As shown in the previous section, the fuzzy rulebase is the most crucial component in the fuzzy controller, every fuzzy decision made as a response to a certain set of

inputs is inferred from the applicable rules out of the rulebase. The fuzzy rulebase must be initialized before the fuzzy controller can be put into use. In a more traditional fuzzy controller the consequence of a rule is usually assigned by the user based on previous experience, but for the fuzzy controller developed in this work, the user initially has no knowledge on how to assign the values for outputs $\theta_{\text{thr,comm}}$ and $\theta_{\text{thr,diff}}$ to control the height and velocity of a hop. Therefore the consequence of each rule cannot be decided from the user experience. As such a training process is needed to provide a chance for the fuzzy controller to learn the consequences of each rule by its own in a progressive manner.

Marhefka first used the idea of training to initialize the rulebase for a fuzzy running controller in a quadruped in 2000 [16]. In this work, the content of training is set to be a single-cycle test hop that starts at a TOF and terminates at the next TOF. The detailed training process for one rule is summarized in Table 3.4. The selection of the fuzzy controller outputs, as discussed in Section 3.3, along with the training heuristics, turned out to have a heavy influence on whether the training process for one rule converges and how fast it converges.

Once a rule is trained, training moves on to the next rule until all 189 rules are trained. As discussed in Chapter 2, the training has to be done in simulation due to the large number of test hops required and the potential risk for damage to the machine during test hops. Another practical reason against training with the physical machine lies in the fact that it is very difficult to initialize the physical machine to the desired torso horizontal velocity at the beginning of every test hop. However, the fuzzy rulebase that results from the simulated training process yet faces another challenge that when it is *first* used to control hops in the physical machine, not only

Table 3.4: Training process for one rule

1. Initialize the biped at a TOF with h_o as the torso height, v_o as the torso velocity and the legs in an initial static configuration. h_o and v_o are specified by the rule premise.
2. Initialize $\hat{\theta}_{thr,comm}$, the estimated value for $\theta_{thr,comm}$, and $\hat{\theta}_{thr,diff}$, the estimated value for $\theta_{thr,diff}$, from an adjacent rule.
3. Perform the test hop.
4. Evaluate the hop at the next TOF from the actual hop height h and the actual hop velocity v
   (a) If $|h_o + \Delta h_d - h| < \delta_h$ and $|v_o + \Delta v_d - v| < \delta_v$ then
   $\hat{\theta}_{thr,comm} \Rightarrow \theta_{thr,comm}$ and $\hat{\theta}_{thr,diff} \Rightarrow \theta_{thr,diff}$,
   the rule is considered trained, and the
   training process for this rule terminates.
   (b) Else
      i. Update $\hat{\theta}_{thr,comm}$ and $\hat{\theta}_{thr,diff}$:
      $w_h \cdot (h_o + \Delta h_d - h) + \hat{\theta}_{thr,comm} \Rightarrow \hat{\theta}_{thr,comm}$,
      -$w_v \cdot (v_o + \Delta v_d - v) + \hat{\theta}_{thr,diff} \Rightarrow \hat{\theta}_{thr,diff}$,
      where w_h and w_v are the positive weights used to regulate the update rate;
      ii. Re-initialize the biped at TOF with h_o and v_o and the legs in the initial static configuration. Then repeat the training process from Step 3 while applying the updated $\hat{\theta}_{thr,comm}$ and $\hat{\theta}_{thr,diff}$.

the actual controller performance has to be stable, but it also has to be in close range to the desired rulebase. Otherwise, the on-line adaptation will not even have a working ground. Thus a great effort has been spent to develop a reality-reflecting model for the experimental system, as detailed in Chapter 2. In the simulation, the

training satisfaction range is set to be: $\delta_h = 0.001\ m$, $\delta_v = 0.01\ m/s$; the update weights are set to be: $w_h = 30.0\ rad/m$, $w_v = 0.8\ rad/(m/s)$. Typically the training process for one rule takes about 100 to 200 iterations. The training for the whole rulebase takes approximately 5 hours on a moderately-configured dual-core Thinkpad T400 laptop.

The training performance for one rule can also be greatly improved by setting its initial tentative thrusts with the consequences from one of its adjacent trained rules. The indices of two adjacent rules have only one different digit and are only off by 1. For example, rule 2422 and rule 2421 are considered adjacent rules. However, the initial tentative thrusts for the first rule have to be assigned manually. In this work, the training starts with rule 2422, in which the biped starts falling from a TOF with medium height and zero horizontal velocity and attempts to hop straight back up to the initial height in the next TOF still with zero horizontal velocity. This is the most 'central' rule in the rulebase and relatively easy to get trained. Later the training expands to other rules in a manner similar to tree traversal. Figure 3.11 shows two possible training plan trees (partially shown). The training plan can vary as long as the 'adjacent' principle is satisfied. Several different training plans applying the 'adjacent' principle were generated and used to train the rulebase in the simulation respectively. It turned out that the different training sequences do not make too much difference in the final trained rulebases. Such flexibility and simplicity in generating a training plan and consistency in the trained results benefit directly from the 2-output setup of the fuzzy controller used in this work. Hester's 6-output controller [25], by contrast, has much more difficult training problems to address.

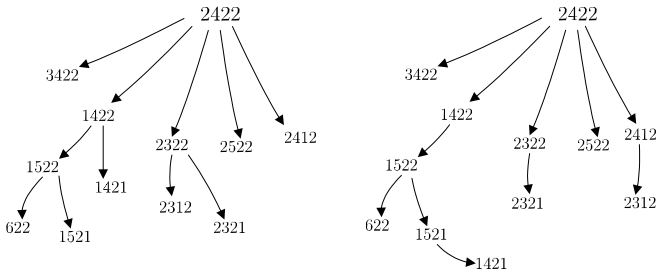

Figure 3.11: Possible training plan trees (partial).

A limitation in the training is: at the beginning TOF of every single test hop, the legs are always initialized to a static configuration, i.e. there are no link motions. However, later when the trained fuzzy rulebase is used to control multiple hops in the biped, the link rates at each TOF (except the first one) are not zero and vary from TOF to TOF. This discrepancy in the initial condition of a hop has certain negative effects on the applicability of the simulated training results. Such a problem can be alleviated later through an on-line adaptation process.

3.6 On-line Adaptation

Due to the modeling error and inexact starting conditions, the fuzzy rulebase that comes out from the simulated training will not produce the exact performance as desired when it is used by the fuzzy controller on the physical machine. An on-line adaptation process can help to fine-tune the fuzzy rulebase in the real scenario. It should be noted that the on-line adaptation can only work if the control performance of the original simulation-trained fuzzy rulebase does not deviate from the desired one too much.

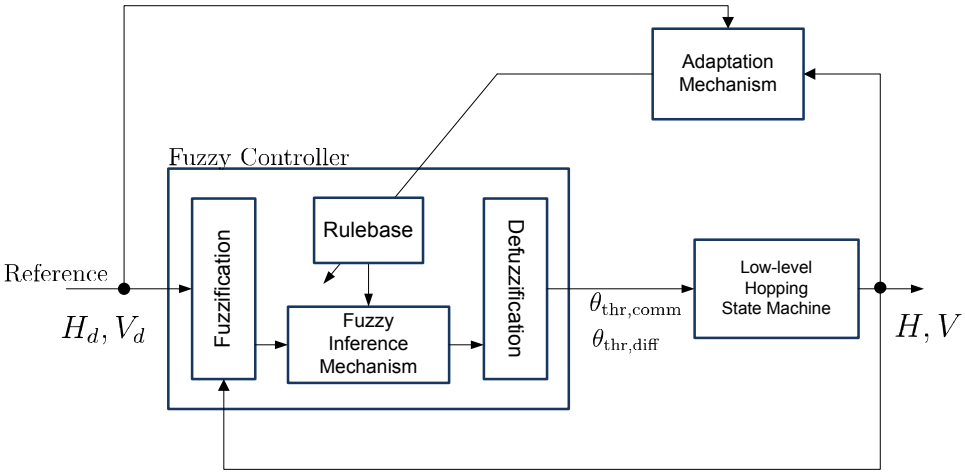

Figure 3.12: Fuzzy controller with on-line adaptation.

At each TOF, the on-line adaptation mechanism is invoked right before the fuzzy controller is called. It evaluates the last hop and modifies the output singletons of the active rules from the last hop, based on the difference between the actual and desired performance. Then the fuzzy controller will be called to select the thrust angles for the next hop using the immediately updated rulebase. Specifically, if at the $(j-1)^{th}$ TOF, the desired hop height for the next TOF is $h_d(j-1)$ and the desired hop velocity for next TOF is $v_d(j-1)$, and later at the j^{th} TOF, the actual hop height is $h(j)$ and the actual hop velocity is $v(j)$, then

$$e_h(j-1) = h_d(j-1) - h(j) \ , \tag{3.5}$$

$$e_v(j-1) = v_d(j-1) - v(j) \ , \tag{3.6}$$

and the rule output singletons are updated as follows:

$$u_{z,1}(j) = u_{z,1}(j-1) + K_1 \cdot \mu_z(j-1) \cdot e_h(j-1) \ , \tag{3.7}$$

$$u_{z,2}(j) = u_{z,2}(j-1) + K_2 \cdot \mu_z(j-1) \cdot e_v(j-1) \ , \tag{3.8}$$

where $z \in S$, and S is the set of indices of rules that are activated at the $(j-1)^{th}$ TOF. The adaptation heuristics are the same as the ones used in the simulated training. K_1 and K_2 are the adaptation gains for the two output singletons, respectively, and are tuned experimentally. The certainty of the rules are used to scale the update size. In particular, the output singletons of the more applicable rules were changed more.

In the actual experiments, the original fuzzy rulebase and the on-line adaptation data are kept in different files; and the old on-line adaptation data is backed up before a new test is performed. In this way, if a bad adaptation happens (due to faulty operations, for example), the previous results will not be affected.

3.7 Summary

This chapter gives a detailed description of the hopping controller designed for the experimental biped, KURMET. KURMET is a complex mechanical system, and considerable effort has been spent on the development of the fully customized low-level hopping state machine and its interface with the higher level fuzzy controller. The fuzzy controller does not look for an analytically formulated control strategy. Instead, it can learn the control knowledge through a training process using simple heuristics derived from a basic understanding of the system. The training process, however, is performed in a simulated environment since it cannot be performed on the physical machine due to several reasons. Therefore, a good estimate of the model for the system is required so that later the fuzzy rulebase that is learned in the simulated

training can be applied to control the physical machine. The on-line adaptation will improve the performance of the simulated fuzzy rulebase. However it may not work unless the original simulated fuzzy rulebase can provide a stable performance with errors limited to a reasonable range in the first place.

It has been shown experimentally that the hopping controller developed in this chapter has a good performance on the physical machine. The detailed control results are presented in Chapter 4.

CHAPTER 4

HOPPING CONTROL RESULTS

4.1 Introduction

This chapter presents the results of the hopping control strategy developed for KURMET in Chapter 3. It first examines the relationships between the inputs and outputs in the simulation-trained fuzzy rulebase. It then compares the controller's performance before and after the on-line adaptation process. Finally, it shows the controller's ability to execute relatively complex hop height/velocity profiles with accuracy after adequate adaptation.

4.2 Input/Output Relationships in the Fuzzy Rulebase

Figure 4.1 shows the $\theta_{\text{thr,comm}}/\theta_{\text{thr,diff}}$ - v_o curves corresponding to different h_o, where Δh_d and Δv_d are maintained at zero. Figure 4.2 shows the $\theta_{\text{thr,comm}}/\theta_{\text{thr,diff}}$ - v_o curves corresponding to different Δv_d, where Δh_d is maintained at zero and h is maintained at 0.58 m. Both $\theta_{\text{thr,comm}}$ and $\theta_{\text{thr,diff}}$ are the direct outputs of the controller.

The training heuristics include the following ideas:

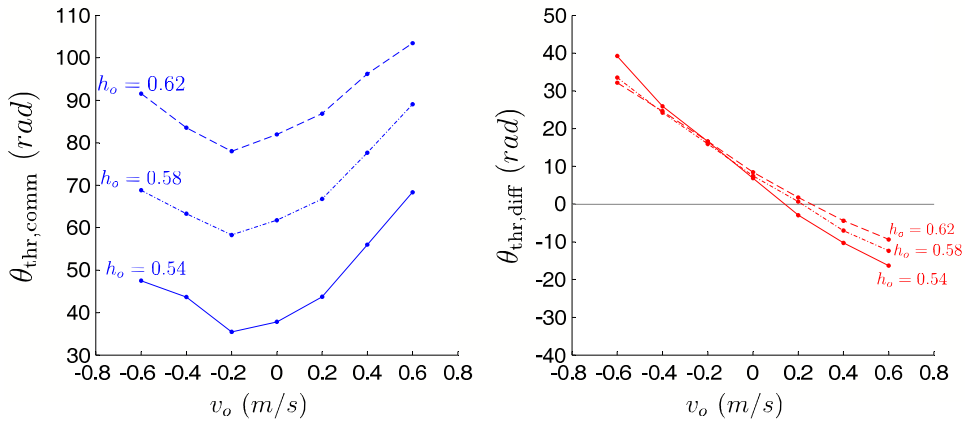

Figure 4.1: $\theta_{\text{thr,comm}}/\theta_{\text{thr,diff}}$ - v_o fuzzy curves corresponding to different h_o with $\Delta v_d = 0\ m/s$ and $\Delta h_d = 0\ m$.

- Increase $\theta_{\text{thr,comm}}$ if the desired change of hop height is positive. Decrease $\theta_{\text{thr,comm}}$ if the desired change of hop height is negative.
- Increase $\theta_{\text{thr,diff}}$ if the desired change of hop velocity is negative. Decrease $\theta_{\text{thr,diff}}$ if the desired change of hop velocity is positive.

The curves in Fig. 4.1 and 4.2 have actually proven the effectiveness of the training heuristics. Strong correlations between $\theta_{\text{thr,comm}}$ and the hop height and between $\theta_{\text{thr,diff}}$ and the hop velocity that are consistent with the training heursitics can be observed from the simulation-trained fuzzy curves.

It can also be observed from Figs. 4.1 and 4.2 that:

- $\theta_{\text{thr,diff}}$ does not vary much with hop height.
- To achieve the same desired hop height, $\theta_{\text{thr,comm}}$ goes up with increasing $|v_o|$.
- The fuzzy curves are not symmetric around a zero hop velocity.

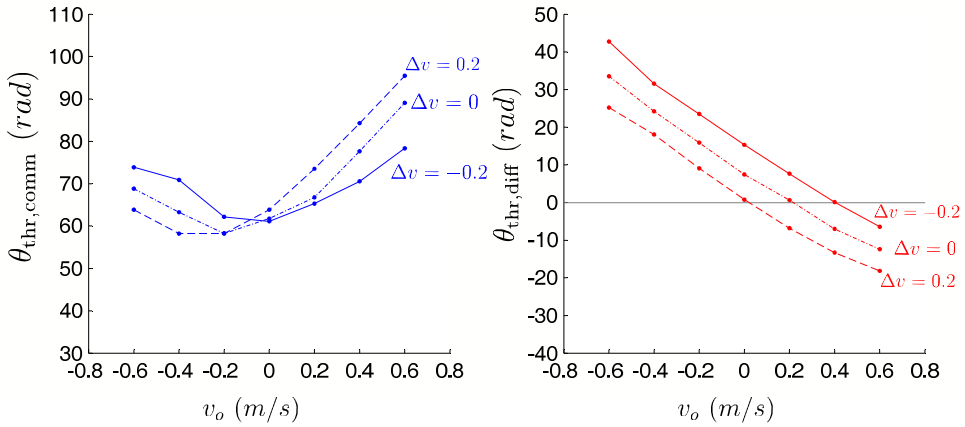

Figure 4.2: $\theta_{\text{thr,comm}}/\theta_{\text{thr,diff}}$ - v_0 fuzzy curves corresponding to different Δv_d with $h_o = 0.58\ m$ and $\Delta h_d = 0\ m$.

The actual outputs of the controller are $\theta_{thr,t}$ and $\theta_{thr,s}$. $\theta_{thr,t}$ and $\theta_{thr,s}$ are related to $\theta_{\text{thr,comm}}$ and $\theta_{\text{thr,diff}}$ through Eqs. 3.1 and 3.2. Figure 4.3 and 4.4 show many of the same characteristics as Fig. 4.1 and 4.2 but $\theta_{\text{thr,comm}}$ and $\theta_{\text{thr,diff}}$ have been converted into $\theta_{thr,t}$ and $\theta_{thr,s}$. Further, they show the highly nonlinear relationships between the inputs and outputs.

4.3 Performance Before and After On-line Adaptation

After the rulebase is populated through off-line training, it is tested on the physical machine. An on-line adaptation process is used to modify the simulation-trained rulebase to achieve a better hopping performance. A simple hopping profile (Table 4.1) is used to examine the effectiveness of the on-line adaptation.

Figure 4.5 shows the controller's performance of executing the profile in Table 4.1 using the simulation-trained rulebase only and with no on-line adaptation. Three curves can be seen in this figure: the desired profile, the fuzzy reference and the

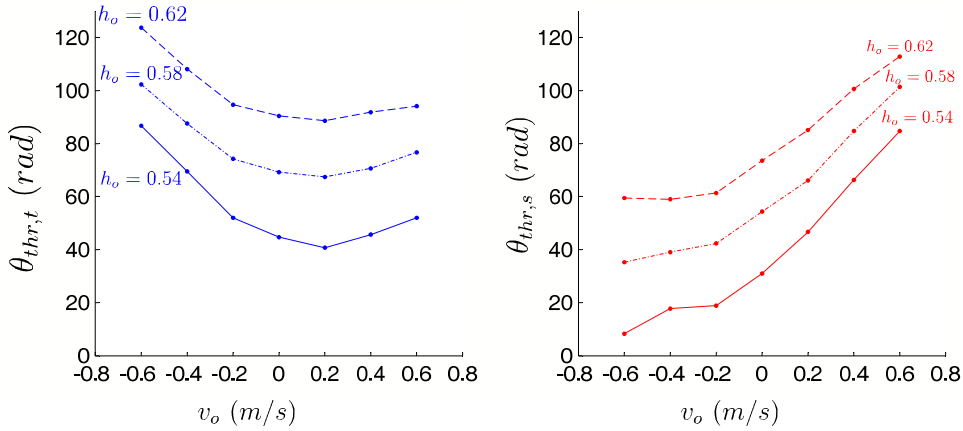

Figure 4.3: $\theta_{thr,t}/\theta_{thr,s}$ - v_o fuzzy curves corresponding to different h_o with $\Delta v_d = 0\ m/s$ and $\Delta h_d = 0\ m$.

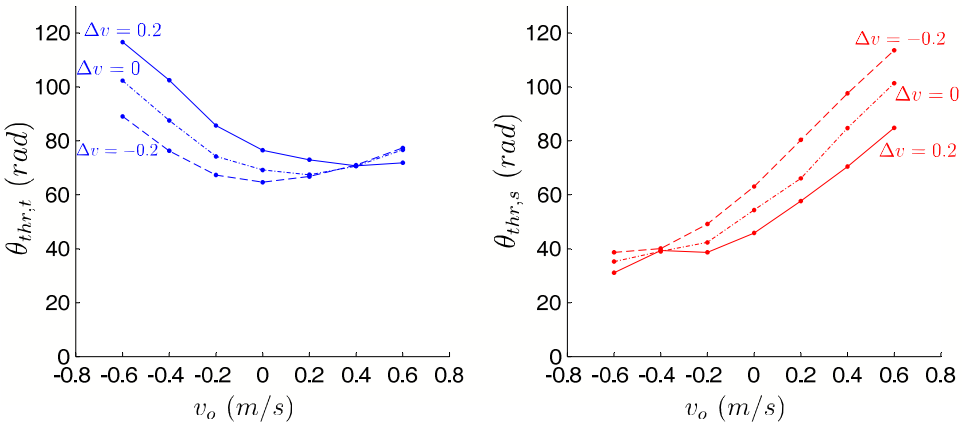

Figure 4.4: $\theta_{thr,t}/\theta_{thr,s}$ - v_o fuzzy curves corresponding to different Δv_d with $h_o = 0.58\ m$ and $\Delta h_d = 0\ m$.

actual profile. The fuzzy references are intermediate position/velocity commands. The fuzzy controller avoids changing the desired position/velocity too much in one hop due to the rulebase limitation. If the desired change of hop height is larger than

Table 4.1: A simple profile used to compare the hopping performance before and after on-line adaptation

Hop Count	Desired hop height h_d (cm)	Desired hop velocity v_d (cm/s)
1-15	58	0
16-30	58	-20

15 cm or the desired change of hop velocity is larger than 15 cm/s, the fuzzy controller automatically limits them to 15 cm and 15 cm/s, respectively. For example, if the desired profile causes the hop velocity to change 40 cm/s in one hop, then the fuzzy controller will actually do it in 3 hops.

It is clear in Fig. 4.5 that the actual hop profiles are not very accurate. However the hopping is still stable. The controller then executes the same profile again with on-line adaptation on. The resulting curves are shown in Fig. 4.6. The actual profiles are moving closer to the desired profiles.

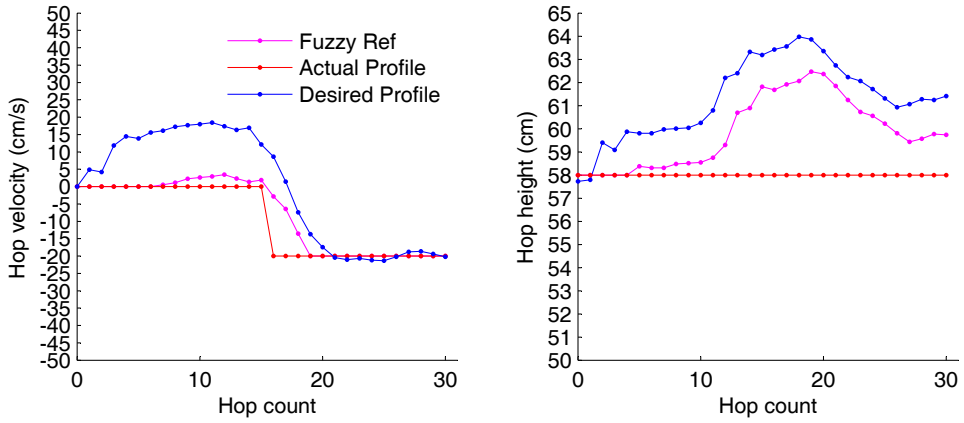

Figure 4.5: The controller's performance of executing the profile in Table 4.1 using the off-line rulebase without any on-line adaptation.

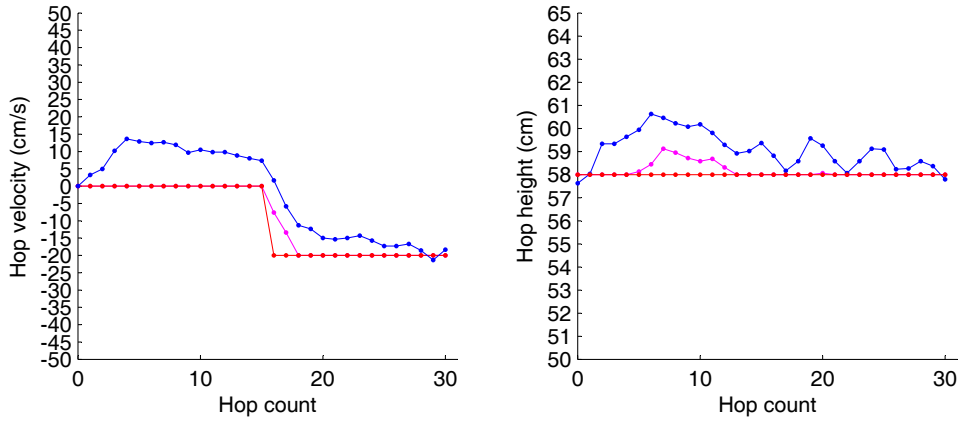

Figure 4.6: The controller's performance of executing the profile in Table 4.1 for the first time with on-line adaptation on. (on-line adaptation data is cumulative.)

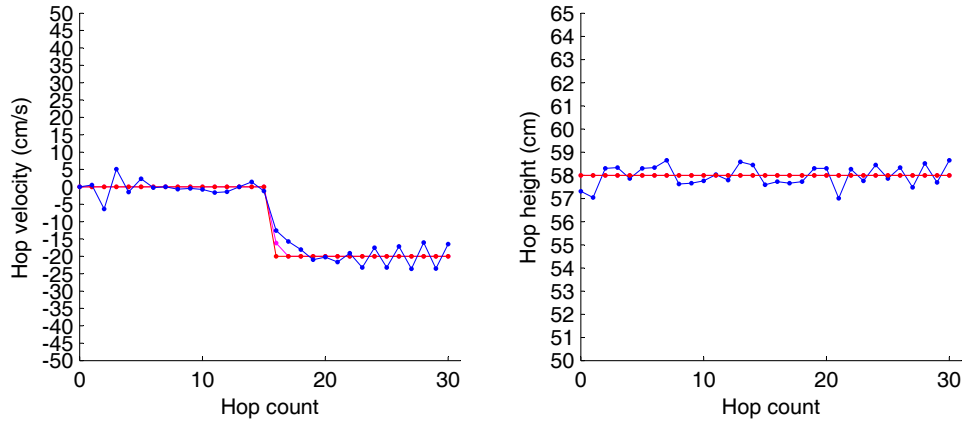

Figure 4.7: The controller's performance of executing the profile in Table 4.1 for the second time with on-line adaptation on.

Figure 4.7 shows the results of executing the same profile for the second time with on-line adaptation on. The actual profiles are already able to follow the desired profiles quite well. However, continuously running the profile with on-line adaptation does

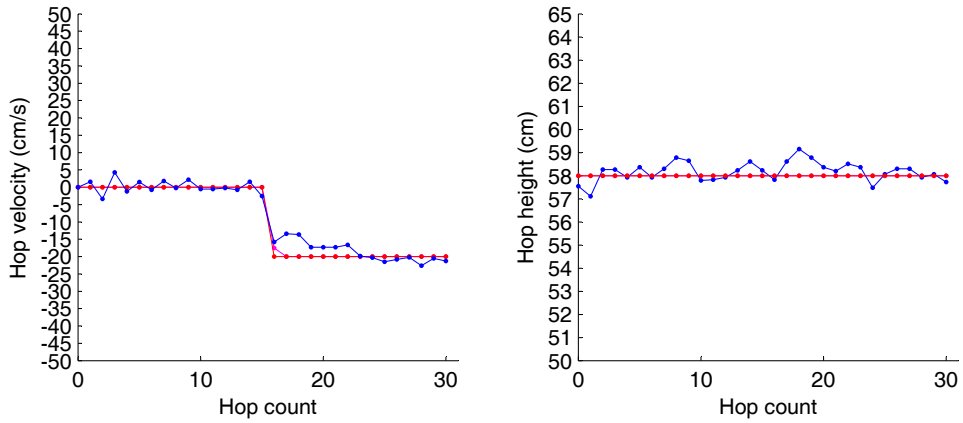

Figure 4.8: The controller's performance of executing the profile in Table 4.1 for the fifth time with on-line adaptation on.

Table 4.2: Hopping performance with on-line adaptation†

	h_d (cm)	$\bar{h}$ (cm)	σ_h (cm)	v_d (cm/s)	$\bar{v}$ (cm/s)	σ_v (cm/s)
2^{nd} ††	58	57.998	0.446	0/-20	-0.329/-20.047	2.417/2.832
5^{th}	58	58.196	0.418	0/-20	0.178/-19.628	1.963/1.996

†† Indicates executing the hopping profile with on-line adaptation on for the 2^{nd} time
† Transitional hops are not counted.

not further improve the controller's performance. Figure 4.8 shows the performance of executing the same profile for the fifth time with on-line adaptation, which is about the same as the second time. See Table 4.2. (Note the on-line adaptation data is cumulative.) This is probably due to the mechanical play in the physical system. The mechanical play introduces certain random factors to the hopping performance, which starts to interfere with the on-line adaptation when the actual values get very close to the desired values and prevent it from producing more accurate results.

Table 4.3: A relatively complex hop profile

Hop Count	Desired hop height h_d (cm)	Desired hop velocity v_d (cm/s)
1-5	58	0
6-10	60	0
11-15	58	0
16-22	58	-40
23-30	58	0
31-37	58	40
38-45	58	0
46-50	60	-20
51-55	58	0

4.4 Execution of a Relatively Complex Profile

After adequate on-line adaptation, the controller is able to robustly follow a relatively complex hopping profile with accuracy. The profile in Table 4.3 has been tested on KURMET and the results are shown in Fig. 4.9.

4.5 Summary

The experimental results for KURMET as presented in this chapter have verified the effectiveness of the control strategy developed in Chapter 3. The simulation-trained rulebase, when applied to the physical machine, is able to produce a stable, continuous hopping motion in KURMET without any adjustment. Based on that, an on-line adaptation process provides significant improvements in the controller's performance through the selective modification of the rule output singletons in the rulebase using a reasonable number of test hops. Finally with the updated rulebase, KURMET is able to stably and accurately follow relatively complex hopping profiles

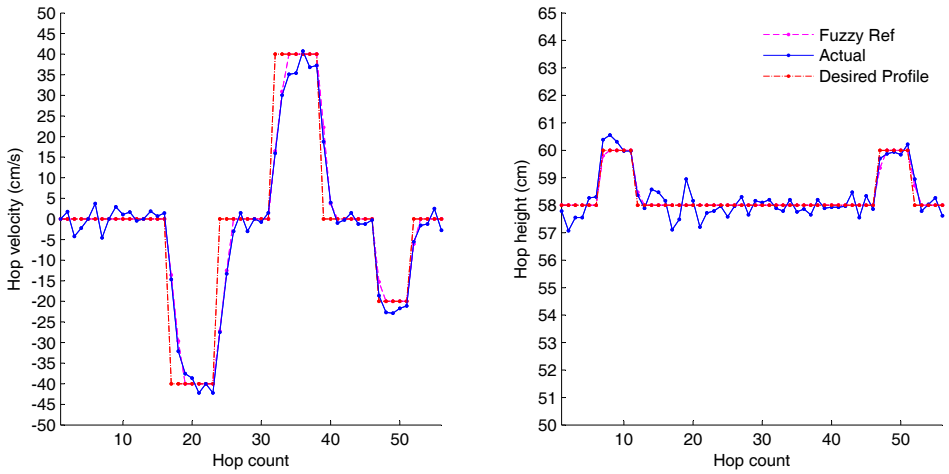

Figure 4.9: The controller's performance of executing the hop profile in Table 4.3 after adequate on-line adaptation. (The adaptation gain has also been turned down.)

over multiple hops. During steady-state conditions, the maximum hop velocity errors are within 2.5 cm/s and the maximum hop height errors are within 1 cm. During transitions the hop velocity errors go up but are still limited to 5 cm/s.

CHAPTER 5

KURMET HIGH JUMP

5.1 Introduction

The controller implemented in the previous chapters has given KURMET the ability to perform consecutive hops in a robust and accurate manner. Another similar dynamic movement in bipedal locomotion, the vertical high jump, is also of interest, and needs to be demonstrated on the KURMET platform. KURMET's mechanical design was specifically optimized for jumping. However, the previous control strategy has not fully exploited this design to produce an optimal jump height. Based on the previous work on KURMET's hopping, this chapter preliminarily explores KURMET's high jump potential to provide insight for further research.

5.2 High Jump State Machine

The investigation of a high jump for KURMET takes advantage of the previously developed hopping state machine. Several parameter values in the hopping state machine are modified to produce an optimal jump height. For convenience, the modified hopping state machine is referred to as high jump state machine. The high jump state machine actually addresses two specific jumps: the first jump (primary jump) aims

to maximize the jump height; the second jump (recovery jump) aims to actively dissipate the energy in the system so that the biped can land safely without damaging the mechanism. As a preliminary investigation, the determination of parameters for the high jump state machine relies heavily on the qualitative understanding of the system and experimental feedback and does not involve any intelligent search method. The experience from tuning the hopping state machine is still applicable. The method to tune the high jump state machine parameter values (for the primary jump) includes:

- Increase the touchdown link deflection;
- Delay the thrust;
- Increase the thrust angle;
- Reduce the thrust time;
- Delay the time to remove energy from the system before lift-off.

Following these, the tuning process gradually pushes the high jump performance to the extreme. Table 5.1 shows the final experimentally-tuned high jump state machine parameter values. The original parameter values for the hopping state machine are also shown for comparison.

It should be noted that, unlike the hopping state machine which addresses a variety of hop heights and velocities, the high jump state machine has only one fixed initial condition (h_o=55 cm, v_o=0 cm/s). All other high jump state machine parameters are specifically optimized to work with this initial condition. Actually 55 cm appears to be an optimal initial drop height for the primary jump. From previous experience, to maximize the jump height, one important way is to maximize the SEA spring

Table 5.1: Comparison of parameter values for the hopping and high jump state machine

	hopping	High Jump		Unit
		primary jump	recovery jump	
h_o range	54∼62	55	last hop height	cm
v_o range	-60∼60	0	last hop velocity	cm/s
θ_{defl}†	0.26	thigh 0.35 shank 0.4	0.25	rad
T_{hd}	60	thigh 85 shank 105	0	ms
T_{thr}	150	145	80	ms
$T_{\text{pr\_lo}}$	50	70	70	ms
$\theta_{\text{thr,comm}}$	fuzzy output	145	68	rad
$\theta_{\text{thr,diff}}$	fuzzy output	5	10	rad

† In the high jump state machine, the link deflection is defined as the difference between the link position and the vertical direction, which is different as defined in the hopping state machine.

deflections (i.e. the energy stored in the springs) at the **BOF**. If the initial drop height is too high, for instance 60 cm or higher, then for safety consideration, the state machine has to adopt a more singular leg touchdown configuration and a shorter hold time, which according to the discussion in Section 3.2.2, will not only significantly increase the stiffness of the system and cause more impact loss, but also reduce the efficiency of the thrust. On the other hand, a much lower initial drop height, for instance 50 cm or lower, allows a less singular touchdown configuration and a longer hold time, however the torso then will not gain enough initial momentum from the drop to adequately compress the legs. In both cases the initial drop heights create certain disadvantages for maximizing the spring deflections at BOF.

It should also be noted that in Table 5.1, for the primary jump, the thigh and shank have different touchdown link deflections. (In the high jump state machine the link deflection is defined as the difference between the link position and the vertical direction, not the difference between the link position and the virtual leg position as defined in the hopping state machine.) In previous tests it has been found that if the thigh and shank have the same touchdown link deflection, then at BOF the thigh will have a larger link deflection than the shank. To balance the link deflections on the thigh and shank at BOF, a slightly larger touchdown link deflection is commanded to the shank.

The actual jump performance produced by the high jump state machine using the parameter values in Table 5.1 are reflected in Figs. 5.1, 5.2, 5.3, 5.4 and 5.5. It can be observed from Figs. 5.1 and 5.2 that, for the primary jump, at BOF, both the thigh and shank get very close to the joint limits, and the thigh and shank motors are pushed as far as possible from the thigh and shank positions, respectively, to maximize the SEA spring deflections. Actually, both the thigh and shank motors have already reached their power limits during the thrust. (The motor currents start to become saturated. See Figs. 5.3 and 5.4.) A larger thrust angle, while significantly increasing the SEA spring deflections at BOF if applied at the right time, can also cause another issue during the **PRE_LO** state. In particular, it takes a longer time to bring the motor back to the safe position from its thrust position due to the maximum motor speed constraint (limited by the power supply voltage (48 V)), which means the energy stored in the springs has to be manually released earlier; i.e. the thrust is used less efficiently. Usually in the high jump the pre-liftoff anti-thrust trajectory time needs at least 70 ms. (In fuzzy controlled hopping this time is shorter, 50 ms.)

70 ms is longer than desirable but is necessary for the current mechanical setup. It can be seen in Figs. 5.1 and 5.2 that the pre-liftoff anti-thrust has been delayed as much as possible and there is barely any time margin left for the motor to come to rest before lift-off.

In contrast to all the effort made above to produce a maximum jump height in the primary jump, the recovery jump uses a much more singular touchdown configuration and applies the thrust immediately after touchdown without any hold time. The thrust is also much faster; it only takes 80 ms. In such a way almost all the thrust is used to prevent the links from deflecting much, and the extra potential energy in the system is safely dissipated.

As a result, the primary jump produces a maximum jump height of 75 cm and the subsequent recovery jump reduces the jump height at the next TOF by 15 cm. See Fig. 5.5. The jump height of 75 cm is substantial producing a normalized jump height (ratio of the jump height and the length of the leg segment) of 3.

5.3 High Jump Initialization

Previously to initialize the high jump, KURMET needed to be manually dropped from 55 cm (torso height). Since the hopping state and the high jump state machine are basically of the same structure, and the starting height of the primary jump and the ending height of the recovery jump are both within the range of the fuzzy hopping controller, it is also possible to integrate the high jump into the fuzzy controlled hopping. Another option is to start the high jump from the ground. The basic idea is to first use a moderate crouching-jump to push KURMET to around 55 cm then

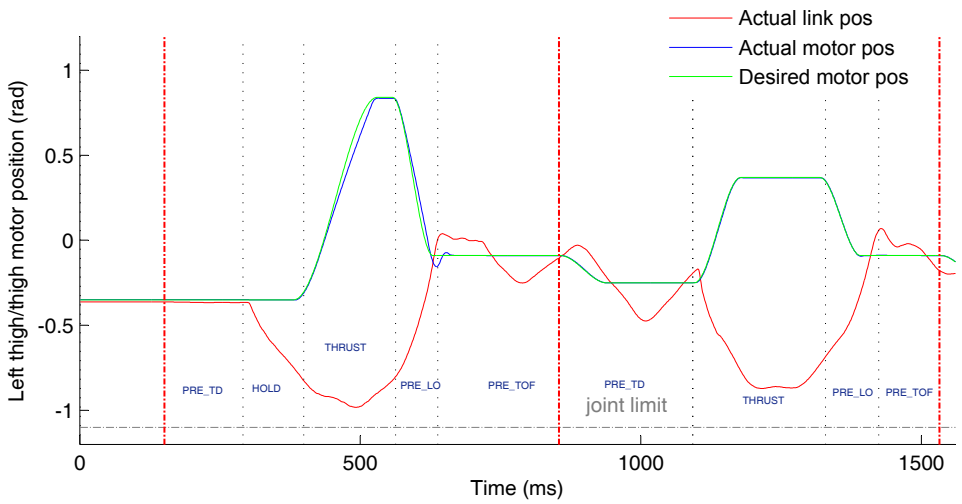

Figure 5.1: KURMET high jump: the left thigh/thigh motor positions for the primary and recovery jumps.

engage the high jump state machine. Both of these features will be implemented for KURMET in the near future.

5.4 Summary

The work in this chapter has preliminarily demonstrated KURMET's ability to perform high jumps. Utilizing a optimized state machine, KURMET has achieved a maximum jump height of 75 cm. Such a jump height, when normalized to the length of the leg segment (3), is comparable to the human's jumping performance. However, the work in this chapter is primarily based on an experimental approach, which has left room for future improvement.

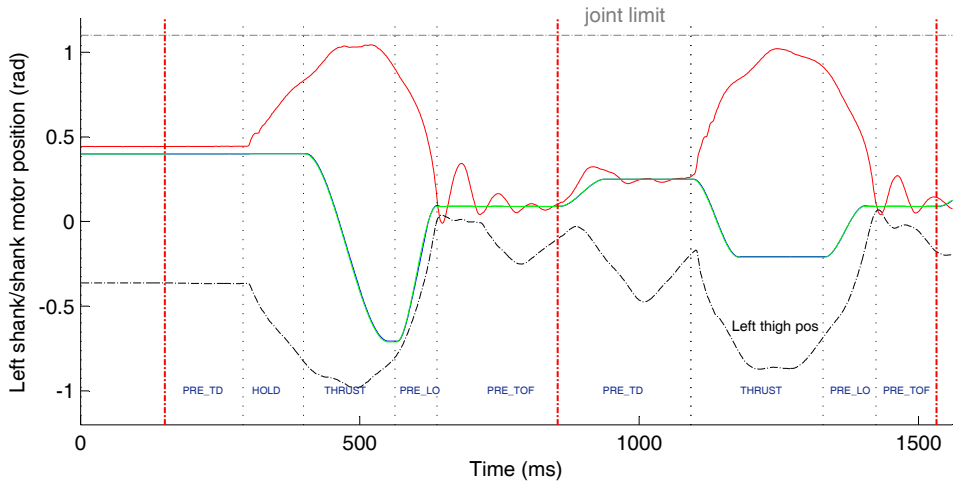

Figure 5.2: KURMET high jump: the left shank/shank motor positions for the primary and recovery jumps.

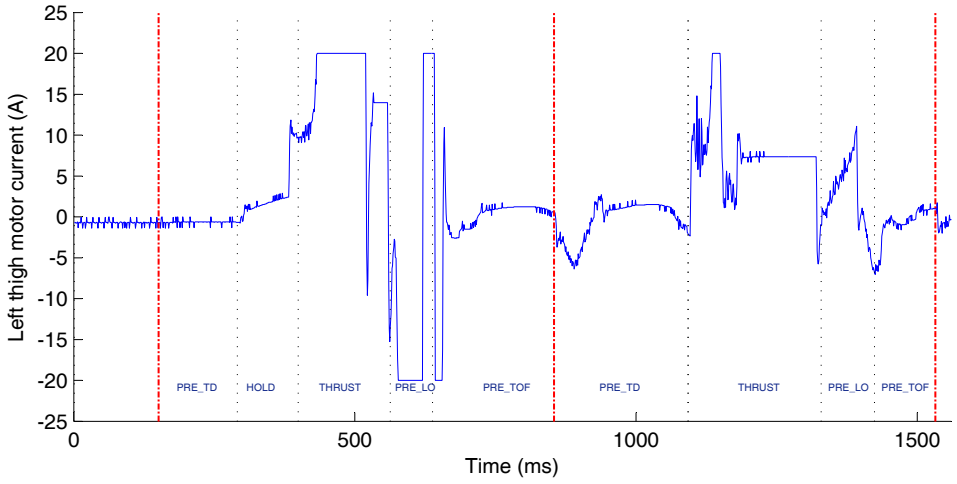

Figure 5.3: KURMET high jump: the left thigh motor current.

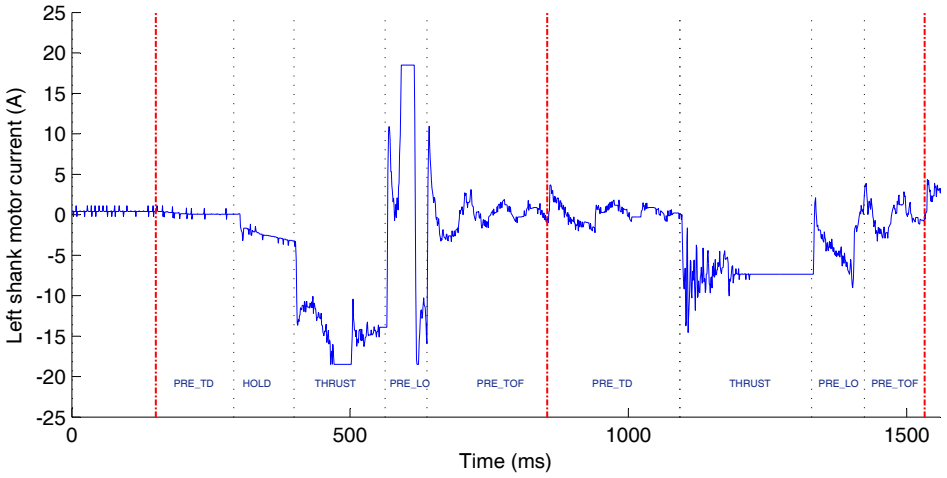

Figure 5.4: KURMET high jump: the left shank motor current.

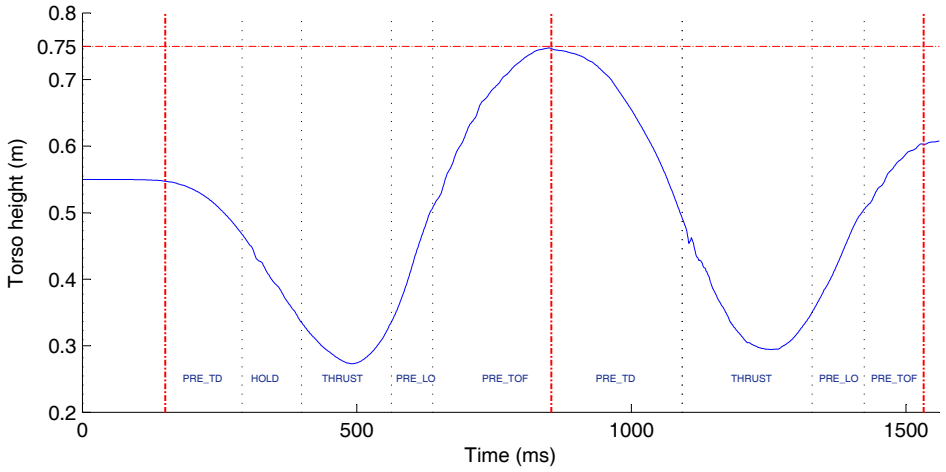

Figure 5.5: KURMET high jump: the torso height.

CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 Summary

This thesis developed an adaptive fuzzy control system for the planar bipedal robot, KURMET, so that it can produce consecutive hops in a tractable manner. KURMET has articulated legs and employs series-elastic actuators (SEA) on all leg joints. The SEA elements provide the compliance needed for dynamic movements but also significantly increase the complexity of the leg dynamics. To manage the complexity, a layered control strategy was adopted.

The low-level control was based on a hopping state machine. Much of the functionality of the state machine was realized by a Galil motion controller and had a control update rate of 1 kHz. The state machine was specifically designed to maximally capture the natural dynamics of the leg compliance so that it was able to efficiently realize a hop in the biped. At the same time, it hid the complexity of the implementation of the hopping motion to the high-level control through parameterization.

With the underlying low-level hopping state machine, a Sugeno-type fuzzy controller was able to perform the role of the high-level control. Unlike the low-level state machine, the fuzzy controller was only called once per hopping cycle at top of

flight (TOF). At each TOF, through regulating the two most important parameters in the low-level state machine, the thigh motor thrust $\theta_{thr,t}$ and the shank motor thrust $\theta_{thr,s}$, using the knowledge from its rulebase, the fuzzy controller was able to control the hop height and velocity at the next TOF.

The fuzzy controller had 4 inputs and 2 outputs. The inputs included the actual hop height and hop velocity at the current TOF, h_o and v_o, and the desired change of the hop height and velocity at the next TOF, Δh_d and Δv_d. All the input variables had minimally overlapping triangular membership functions. The direct output singletons were the common-mode thrust $\theta_{\text{thr,comm}}$ and the differential thrust $\theta_{\text{thr,diff}}$. They were then converted to $\theta_{thr,t}$ and $\theta_{thr,s}$ ($\theta_{thr,t} = \theta_{\text{thr,comm}} + \theta_{\text{thr,diff}}$ and $\theta_{thr,s} = \theta_{\text{thr,comm}} - \theta_{\text{thr,diff}}$). A modest number of rules (189) was used so that the fuzzy controller can be implemented on the physical machine. Also, the minimal number of outputs (2) effectively prevented the training from falling into multiple modes of operation.

The fuzzy controller started with an empty rulebase. An effective training method was developed to populate the rulebase. The training was done through off-line simulation using a full dynamic model of KURMET in *RobotBuilder*. The training process took around 5 hours on a moderately-configured ThinkPad T-400 laptop. The model was pre-calibrated to be a close approximation of the physical system, so that with the immediate rulebase from the training simulation, the fuzzy controller was able to physically stabilize the hopping in KURMET. An on-line adaptation process was further used to modify the fuzzy rule output singletons, which improved the hopping performance on the physical machine.

The results showed that the fuzzy control system developed in this thesis worked very well on KURMET. With this control system, KURMET was able to stably follow any desired hop height within the range from 0.54 m to 0.62 m and any hop velocity within the range from -0.5 m/s to 0.5 m/s during continuous hopping. Under steady-state conditions, the maximum error for the hop velocity is within 2.5 cm/s and the maximum error for the hop height is within 1 cm. This result demonstrated the effectiveness of fuzzy control for a relatively complex system like KURMET.

This thesis also experimentally explored the high jump in KURMET. Through optimizing certain parameters in the hopping state machine (no specialized mechanism was used for the jump), KURMET was able to reach a maximum height of 0.75 m in one single jump. This maximum jump height corresponded to a normalized jump height (the ratio of the maximum jump height to the length of the leg segments) of 3, which was comparable to a human. The high jump required a starting TOF height of 0.55 m, which could be realized either through a crouching hop launched from the ground or through the previous fuzzy controlled hopping. After the high jump, a recovery jump followed to safely dissipate the extra system energy. This high jump test demonstrated KURMET's ability to perform dynamic motions.

6.2 Suggestions for Future Work

As an initial investigation into the full dynamic maneuvers on the experimental biped KURMET, the work done in this thesis has produced positive and encouraging results; it has also revealed a large number of improvements yet to be explored. The following suggestions outline the possible areas for future research:

- The hopping controller currently does not control the pitch of the torso (the torso joint is locked using a removable pin). To allow full control of the pitch action, the difficulty to balance the torso position through series-elastic elements needs to be addressed. The current discrete fuzzy controller is only invoked once per hop cycle at the TOF, and is unable to apply corrections between two fuzzy control calls. This suggests that a continuous-time high-level control may be needed to achieve that objective. The torso pitch control could also be approached from the mechanical side. A modification of the current parallel actuation mechanism could possibly simplify the tasks the controller needs to perform.

- KURMET was designed based on a dynamic model optimized solely for jumping. KURMET's designer hoped a design optimized for vertical jumping could also produce reasonably good performance over a broad range of dynamic motions, such as running. However, simulation and actual tests have both shown, the current SEA springs ($K_s = 30\ Nm/rad$), while ideal for jumping and hopping, are too soft for running. They are not strong enough to provide the desired single-leg support for KURMET during running. The contact leg tends to over-deflect and cause a large torso height variation. In simulation, doubling the current SEA spring constant produces a much more desirable running performance. It suggests that variable compliance may be needed for a biped robot to perform multiple dynamic motions.

- The current method for the detection of TOF is not ideal. Currently, TOF is detected once the vertical boom velocity drops below a certain negative threshold while the system is in the **PRE_TD** state. The vertical boom velocity is not directly sensed in hardware. It is acquired from the numerical differentiation of the vertical boom position signals followed by a low-pass digital filter. The digital filtering, however, brings in a certain amount of delay. On the other hand, the rapid swing of the legs also disturbs the ballistic flight trajectory of the torso, randomly introducing inconsistent rise and fall in the vertical boom position readings. Current filtering method are not able to recognize those perturbations and sometimes falsely detects TOF. Usually this does not harm the stability of the fuzzy control system (extreme cases did exist though) but it does affect the controller's accuracy. In future work, an adaptive filter with certain knowledge of the ballistic trajectory may be developed so that the TOF detection delay can be minimized and the TOF detection method can be more robust to perturbations.

- The KURMET system has sustained high impacts from many test hops. As the experiments goes on, more and more mechanical play has shown up and adds more uncertainties to the control results. In the future work with KURMET, certain mechanical parts, like the motor keyways, the connecting module between KURMET and the boom, cable connections, etc., may be reinforced so that the physical robustness of the system can be improved. Moreover, a mechanism that allows the quick replacement of the easily-worn parts may be included in the future biped system design.

- Due to the current design of the shank series-elastic actuators, the shank motors need to return from their thrust position to the 'safe' side of the singularity hardstop before lift-off so that the extra energy in the spring can be released without damaging the knee joints. Usually it is desirable to delay this energy release process as long as possible so that the motor could stay longer in the thrust position, where the spring potential energy can be maximumly converted into the vertical velocity of the system. However, currently the lack of motor agility prevents this goal to be achieved in a favorable way. Future improvements for the SEA may include adding a spring energy quick release mechanism so that the energy stored in the spring can be more quickly and safely dissipated right before lift-off without being limited by the maximum motor rate. In this way the thrust will be used more efficiently.

- The impact-buffering elastomer pad installed on the unidirectional-SEA hardstop is an easily-consumed part. The condition of this pad has a profound influence on the overall hopping performance. It usually gets damaged in approximately 200 hops and needs to be replaced with a new one. On-line adaptation usually needs almost the same amount of test hops to reasonably adapt most rules in the rulebase. The pads gradually wears out as the on-line adaptation goes, and the effect of the deterioration will be accumulated in the rulebase. So after the pads are replaced, the previously well-adapted rulebase will produce a large error in the subsequent hops, since it has already adapted to the worn pads. The rulebase therefore needs to be re-adapted but the same problem will repeat. Such a dilemma compromises the effectiveness of the on-line adaptation, which is another motivation for a redesign of the SEA.

- The ground contact model in *DynaMechs* should be improved so that it could produce simulation results with higher fidelity.

- Currently the values of most parameters in the low-level hopping/jumping state machine are manually tuned based on simple qualitative analysis of the system dynamics and do not guarantee optimality. In the future a genetic algorithm may be used to search for the optimal combination of parameter values.

In conclusion, the work presented in this thesis has developed an effective and robust fuzzy control system that allows a physical planar biped with articulated legs (KURMET) to execute continuous hopping motion at desired hop heights and velocities. It provides valuable insights on how the full dynamic stability and agility can be achieved on an artificial bipedal structure with biologically-realistic features, an area still not fully investigated. The experimental results from this work have shown the power of intelligent control in exploring that area. This work also accumulated much practical experience about controlling the KURMET platform, which will greatly expedite future research with this platform on other dynamic movements in bipedal locomotion.

APPENDIX A

SYSTEM PARAMETERS

Table A.1: Link mass

Link	Mass (kg)
Boom2	2.70
Torso	12.1
Thigh Link	0.81
Shank Link	0.63
Total Biped	14.98

Table A.2: Center-of-mass for each link*

Boom2	$\begin{bmatrix}0.00 & -1.0 & 0.00\end{bmatrix}$ m
Torso	$\begin{bmatrix}0.138 & 0.0008 & 0.0849\end{bmatrix}$ m
Right thigh	$\begin{bmatrix}0.0784 & -0.0001 & -0.0037\end{bmatrix}$ m
Left thigh	$\begin{bmatrix}0.0784 & -0.0001 & 0.0015\end{bmatrix}$ m
Shank	$\begin{bmatrix}0.0964 & -0.0003 & -0.0002\end{bmatrix}$ m

* Each center-of-mass (COM) vector is relative to its corresponding link frame. All the link frames are displayed in Fig. 2.3.

Table A.3: Inertia matrix for each link†

Boom2	$\begin{bmatrix}3.60 & 0.00 & 0.00\\ 0.00 & 4.10\times10^{-3} & 0.00\\ 0.00 & 0.00 & 3.60\end{bmatrix}$ kg·m^2
Torso	$\begin{bmatrix}0.218 & 1.76\times10^{-3} & 0.159\\ 1.76\times10^{-3} & 0.470 & 4.82\times10^{-3}\\ 0.159 & 4.82\times10^{-3} & 0.334\end{bmatrix}$ kg·m^2
Thigh††	$\begin{bmatrix}3.90\times10^{-4} & -1.60\times10^{-5} & -2.10\times10^{4}\\ -1.60\times10^{-5} & 1.27\times10^{-2} & 0.00\\ -2.10\times10^{4} & 0.00 & 1.27\times10^{-2}\end{bmatrix}$ kg·m^2
Shank	$\begin{bmatrix}2.38\times10^{-4} & -2.0\times10^{-6} & -1.00\times10^{-6}\\ -2.0\times10^{-6} & 1.16\times10^{-2} & 0.00\\ -1.00\times10^{-6} & 0.00 & 1.16\times10^{-2}\end{bmatrix}$ kg·m^2

† Each inertia matrix is relative to its corresponding link frame. All the link frames are displayed in Fig. 2.3.

††The right and left thigh are not identical, so there are two COM vectors listed. However, the differences in the inertia matrices are negligible, so only one matrix is presented [8].

Table A.4: Motor parameters

Name	Value	Unit
Values at nominal voltage		
Nominal voltage	48	V
No load speed	16500	rpm
No load current	422	mA
Nominal speed	15800	rpm
Nominal torque (max continuous torque)	120	mN·m
Nominal current (max continuous current)	4.7	A
Stall torque	430	mN·m
Characteristics		
Winding resistance (R) (between two connections)	0.386	Ω
Winding inductance (L) (between two connections)	0.0653	mH
Torque constant ($k_{\tau m}$)	0.0276	N·m/A
Back EMF coefficient (k_b)	0.00289	V/rpm
	0.0276	V·s/rad
Speed/torque gradient	4.83	rpm/mN·m
Mechanical time constant	1.68	ms
Rotor inertia	33.3	g·cm^2
	3.33×10^{-6}	kg·m^2
Combined rotor and gearbox inertia (J_m)	4.73×10^{-6}	kg·m^2
Motor damping (B_m)	3.5×10^{-6}	N·m·s/rad

Calculation of B_m:

$$b = \frac{1}{\text{Speed/torque gradient}} = \frac{1}{4.83}\frac{\text{mN·m}}{\text{rpm}} = 0.00197708\frac{\text{N·m·s}}{\text{rad}}\,, \tag{A.1}$$

$$B_m = b - \frac{k_{\tau m}^2}{R} = 3.5\times10^{-6}\frac{\text{N·m·s}}{\text{rad}}\,. \tag{A.2}$$

Table A.5: Gearbox parameters

Name	Value	Unit
Forward efficiency (η_f)	72	%
Backward efficiency (η_b)	55	%
Gear Ratio (n_m)	126	-
Gearbox inertia	1.4×10^{-6}	N·m·s/rad

Table A.6: Joint limits

Joint Angle	Range
θ_h	-1.232 to 1.232 rad
θ_k	0.0 to 2.175 rad

Table A.7: USEA parameters

Name	Value	Unit
Spring constant (K_s)	30	N·m/rad
Pretensioning torque (τ_p)	2	N·m
Unidirectional hardstop contact spring constant (K_c)	300	N·m/rad
Unidirectional hardstop contact nonlinear damping (λ_c)	60	N·m·s/rad^2

Table A.8: Constraints by amplifier

Name	Value	Unit
AMC ZBDC12A8		
Range of DC Supply Voltage	16-80	V
Peak Current (max. duration 2s)	±12	A
Max. Continuous Current	±6	A
AMC AZBDC20A8		
DC Supply Voltage	10-80	V
Peak Current (max. duration 2s)	20	A
Max. Continuous Current	12	A

Table A.9: DC power supply parameters

Name	Value	Unit
TDK-Lambda SWS600L-48		
Output voltage (V)	48	V
Peak output current	13	A
Peak output power	624	W

The DC power supply is comprised of 6 Lambda SWS600Ls that are arranged in two separate arrays (3 in each). Each array powers a leg.

Table A.10: Ground contact parameters

Name	Value	Unit
Normal spring Constant ($K_{g,n}$)	17500	N/m
Tangential spring Constant ($K_{g,t}$)	17500	N/m
Normal damping coefficient ($B_{g,n}$)	250	N·s/m
Tangential damping coefficient ($B_{g,t}$)	250	N·s/m
Static friction coefficient (μ_s)	0.4	-
Kinetic friction coefficient (μ_k)	0.3	-

APPENDIX B

CUBIC SPLINE TRAJECTORY FOR THE MOTOR POSITION

The generation of a *cubic spline* trajectory for the motor position takes the following inputs: the motor position θ_{m0} and the motor rate $\dot{\theta}_{m0}$ at time t_0, the desired motor position θ_{md} and the desired motor rate $\dot{\theta}_{md}$ at time $t_0 + T$ (usually $\dot{\theta}_{md}$ is expected to be zero). These are the four constraints for the motor position trajectory:

$$\begin{aligned} \Theta_m(t_0) &= \theta_{m0} \ , \\ \Theta_m(t_0 + T) &= \theta_{md} \ , \\ \dot{\Theta}_m(t_0) &= \dot{\theta}_{m0} \ , \\ \dot{\Theta}_m(t_0 + T) &= 0 \ . \end{aligned} \tag{B.1}$$

Then the desired trajectory for the motor position can be formulated as:

$$\Theta_m(t) = \begin{cases} a_0 + a_1(t - t_0) + a_2(t - t_0)^2 + a_3(t - t_0)^3 & t_0 \le t \le T + t_0 \ , \\ a_0 + a_1 T + a_2 T^2 + a_3 T^3 & T + t_0 < t \ , \end{cases} \tag{B.2}$$

where

$$\begin{aligned} a_0 &= \theta_{m0} \ , \\ a_1 &= \dot{\theta}_{m0} \ , \\ a_2 &= \frac{3}{T^2}(\theta_{md} - \theta_{m0}) - \frac{2}{T}\dot{\theta}_{m0} \ , \\ a_3 &= -\frac{2}{T^3}(\theta_{md} - \theta_{m0}) - \frac{1}{T^2}\dot{\theta}_{m0} \ . \end{aligned} \tag{B.3}$$

The trajectory and its derivatives are also shown in Fig. B.1. More details about cubic spline trajectories can be found in [27].

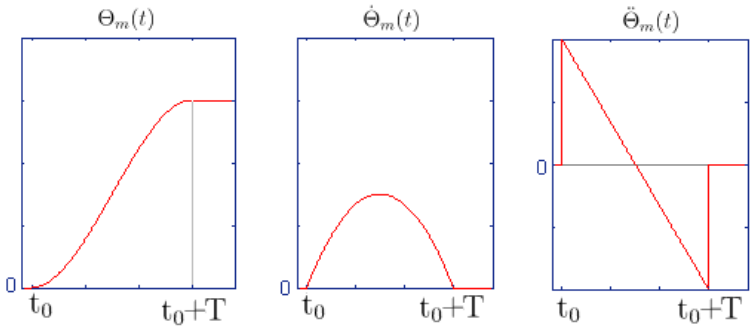

Figure B.1: Cubic spline trajectory of the motor position and its derivatives.

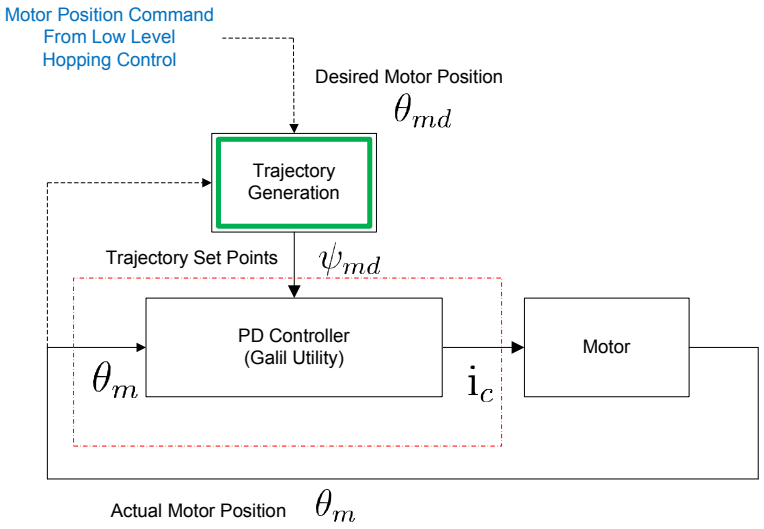

Figure B.2: Trajectory generation and PD controller.

When the desired motor position changes, a cubic-spline trajectory is generated automatically to help the motor smoothly transition to the new position over a period of time without any jerky motions (Fig. B.2). The motor position is sampled by the encoder at a certain frequency (10^3 Hz) and a discrete-time PD controller provides

closed-loop control on the motor position based on these sampled signals to make sure the motor follows the trajectory setpoints. The command current of the motor is the directly manipulated variable.

It should be noted that in this thesis, the motor command current is *always* under closed-loop control. This is different from Hester's jumping controller for KURMET [25]. His controller would command open-loop current on the motor for a certain period of time in a jump.

APPENDIX C

TORSO HEIGHT AND HORIZONTAL VELOCITY

The height, h, and the horizontal component of the velocity, v, at the center point of the torso, along the torso pitch joint axis, may be computed from the boom position and rate. See Figure 2.3 and Table 2.1. The resulting kinematic equations are given as follows:

$$h = s + (L + \frac{l}{2}) \cdot \sin(\theta_{B2} - \frac{\pi}{2}) \ , \tag{C.1}$$

$$v = (L + \frac{l}{2}) \cdot \cos(\theta_{B2} - \frac{\pi}{2}) \cdot \dot{\theta}_{B1} \ . \tag{C.2}$$

s is the height of the boom joint (0.5 m), L is the length of the boom (0.28 m) and l is the width of the torso (0.18 m). θ_{B1} and θ_{B2} are the boom yaw and pitch angles respectively.

BIBLIOGRAPHY

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

[2] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura. The intelligent ASIMO: system overview and integration. *IEEE/RSJ International Conference on Intelligent Robots and System*, 3:2478–2483, 2002.

[3] M. Vukobratović and B. Borovac. Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics*, 1(1):157–173, 2004.

[4] R. Niiyama, A. Nagakubo, and Y. Kuniyoshi. Mowgli: A bipedal jumping and landing robot with an artificial musculoskeletal system. *IEEE International Conference on Robotics and Automation*, pages 2546–2551, April 2007.

[5] T. Yang, E. Westervelt, J. Schmiedeler, and R. Bockbrader. Design and control of a planar bipedal robot ERNIE with parallel knee compliance. *Autonomous Robots*, 25(4):317–330, 2008.

[6] E.R. Westervelt, J.W. Grizzle, and D.E. Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48:42–56, 2001.

[7] Boston Dynamics. Petman - bigdog gets a big brother. `http://bostondynamics.com/robot_petman.html`, 2009.

[8] B.T. Knox. Design of a biped robot capable of dynamic maneuvers. Master's thesis, The Ohio State University, Department of Mechanical Engineering, 2008.

[9] P. Wensing. Real-time computer control of a prototype bipedal system. Undergraduate Honors Thesis. The Ohio State University, Department of Electrical & Computer Engineering, 2009.

[10] J.E Pratt, M.C. Chee, A. Torres, P. Dilworth, and G.A. Pratt. Virtual model control: an intuitive approach for bipedal locomotion. *International Journal of Robotics Research*, 20(2):129–143, 2001.

[11] J. W. Hurst and A. A. Rizzi. Series compliance for an efficient running gait. *IEEE Robotics and Automation Magazine*, 15:42–51, 2008.

[12] Daniel Paluska and Hugh Herr. The effect of series elasticity on actuator power and work output: Implications for robotic and prosthetic joint design. *Robotics and Autonomous Systems*, 54:277–287, Feb. 2006.

[13] J.M. Remic. Prototype leg design for a quadruped robot application. Master's thesis, The Ohio State University, Department of Mechanical Engineering, 2005.

[14] S. Curran, D.E. Orin, B.T. Knox, and J.P. Schmiedeler. Analysis and optimization of a series-elastic actuator for jumping in robots with articulated legs. In *Proc. of 2008 ASME Dynamic Systems and Control Conference*, Ann Arbor, Michigan, October 2008.

[15] P.J. Antsaklis and K.M. Passino, editors. *An Introduction to Intelligent and Autonomous Control*. Kluwer Academic Publishers, 1993.

[16] Duane Winfield Marhefka. *Fuzzy control and dynamic simulation of a quadruped galloping machine*. PhD thesis, Ohio State University, Department of Electrical & Computer Engineering, 2000.

[17] D. W. Marhefka, D. E. Orin, J. P. Schmiedeler, and K. J. Waldron. Intelligent control of quadruped gallops. *IEEE/ASME Transactions on Mechatronics*, 8(4):446–456, December 2003.

[18] D. P. Krasny and D. E. Orin. Generating high-speed dynamic running gaits in a quadruped robot using an evolutionary search. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(4):1685–1696, August 2004.

[19] D.P. Krasny. *Evolving dynamic maneuvers in a quadruped robot*. PhD thesis, Ohio State University, Department of Electrical & Computer Engineering, Columbus, Ohio, 2005.

[20] L.R. Palmer. *Intelligent control and force redistribution for a high-speed quadruped trot*. PhD thesis, Ohio State University, Department of Electrical & Computer Engineering, Columbus, Ohio, 2007.

[21] L.R. Palmer and D.E. Orin. Intelligent control of high-speed turning in a quadruped trot. *Journal of Intelligent and Robotic Systems*, 2009.

[22] L.R. Palmer. Intelligent control of an articulated leg for a quadruped galloping machine. Master's thesis, The Ohio State University, Department of Electrical and Computer Engineering, 2002.

[23] L. R. Palmer, D. E. Orin, D. W. Marhefka, J. P. Schmiedeler, and K. J. Waldron. Intelligent control of an experimental articulated leg for a galloping machine. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 3821–3827, Taipei, Taiwan, September 2003.

[24] J.P. Schmiedeler. *The Mechanics of and Robotics Design for Quadrupedal Galloping*. PhD thesis, Ohio State University, Department of Mechanical Engineering, Columbus, Ohio, 2001.

[25] M. Hester. Stable control of jumping in a planar biped robot. Master's thesis, The Ohio State University, Department of Mechanical Engineering, 2009.

[26] R. McN Alexander. Leg design and jumping technique for humans, other vertebrates, and insects. *Philosophical Transactions: Biological Sciences*, 347:235–248, 2009.

[27] John J. Craig. *Introduction to Robotics: Mechanics & Control*. Addison-Wesley, 1989.

[28] S. McMillan, D.E. Orin, and R.B. McGhee. DynaMechs: An object oriented software package for efficient dynamic simulation of underwater robotic vehicles. In *Underwater Robotic Vehicles: Design and Control*, pages 73–98. TSI Press, 1995.

[29] Scott McMillan. *DynaMechs Simulation Library Reference Manual, v3.0*. The Ohio State University, Department of Electrical Engineering, 2003.

[30] D.W. Marhefka and D.E. Orin. A compliant contact model with nonlinear damping for simulation of robotic systems. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 29(6):566–572, November 1999.

[31] S. Curran. Analysis and optimization of a jump for a prototype leg with series-elastic actuation. Master's thesis, The Ohio State University, Department of Electrical & Computer Engineering, 2007.

[32] S.J. Rodenbaugh. Robotbuilder: a graphical software tool for the rapid development of robotic dynamic simulations. Master's thesis, Ohio State University, Department of Electrical & Computer Engineering, 2003.

[33] S. McMillan. *Computational dynamics for robotic system on land and underwater*. PhD thesis, Ohio State University, Department of Electrical & Computer Engineering, Columbus, Ohio, 1994.

[34] E. Bizzi, V. C. K. Cheung, A. d'Avella, P. Saltiel, and M. Tresch. Combining modules for movement. *Brain Research Reviews*, 57(1):125–133, 2008.

[35] K. Passino and S. Yurkovich. *Fuzzy Control*. Addison Wesley, 1998.

[36] The MathWorks Inc. *Fuzzy Logic Toolbox Users Guide, version 2.0*. The MathWorks, Inc, 1999.