Stable Control of Jumping in a Planar 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

Matthew S. Hester, B.S.M.E.

* * * * *

The Ohio State University

2009

Master's Examination Committee:

Dr. James P. Schmiedeler, Adviser

Dr. David E. Orin

Dr. Chia-Hsiang Menq

Approved by

____

Adviser
Graduate Program in
Mechanical Engineering

ABSTRACT

The ability to perform high-speed dynamic maneuvers is an important aspect of locomotion for bipedal animals such as humans. Running, jumping, and rapidly changing direction are fundamental dynamic maneuvers that contribute to the adaptability and performance required for bipeds to move through unstructured environments. A number of bipedal robots have been produced to investigate dynamic maneuvers. However, the level of performance demonstrated by biological systems has yet to be fully realized in a biped robot. One limiting factor in achieving comparable performance to animals is the lack of available control strategies that can successfully coordinate dynamic maneuvers. This thesis develops a control strategy for producing vertical jumping in a planar biped robot as a preliminary investigation into dynamic maneuvers. The control strategy was developed using a modular approach to allow adaptation to further dynamic maneuvers and robotic systems.

The control strategy was broken into two functional levels to separately solve the problems of planning and performing the jump maneuver. The jump is performed using a low-level controller, consisting of a state machine for determining the current phase of the jump and motor primitives for executing the joint motions required by the current phase. The motor primitives, described by open- and closed-loop control laws, were defined with numeric control parameters for modifying their performance. The high-level controller performs the task of planning the motion required to achieve the desired jump height. Fuzzy control, an intelligent control approach, was selected for the high-level controller. The fuzzy controller uses heuristic information about the biped system to select appropriate control parameters. This heuristic knowledge was implemented in a training algorithm. The training algorithm uses iterative jumps with error-based feedback to determine the control parameters to be implemented by the fuzzy controller.

The control strategy was developed and validated using a numerical simulation of the experimental biped KURMET. The simulation models the dynamics of the biped system and has demonstrated the ability of the control strategy to produce stable successive jumps with an approximate height of 0.575 m. The control strategy was also implemented on the experimental biped for a simplified case, resulting in stable successive jumps with a range of heights from 0.55 to 0.57 m.

To every professor who has put forth the effort to make a course spectacular.

ACKNOWLEDGMENTS

I would like to thank everyone who has supported me throughout my education. In particular, I would like to thank my adviser, Dr. James P. Schmiedeler, for his advice and ability to give just the right amount of direction. I have learned a great deal from him, both on technical subjects and on how to choose the right course of action. I would also like to thank Dr. David E. Orin for sharing his experience and knowledge and for always offering a thoughtful response.

I would like to thank Brian Knox for his excellent work in designing a powerful new biped robot. His strong engineering skills are evident in every component of the system. I greatly appreciate his suggestions and friendship throughout the project. I'd like to thank Yiping Liu and Pat Wensing for their key contributions to the control electronics and software. I particularly thank Pat for his dedication and time since day one. To Ghassan Bin Hammam, I extend many thanks for help and patience in troubleshooting all things programming.

I also appreciate the support and insight of fellow lab members Satya Ambike, Vivek Yadav, and Ben McCandless. Their fresh perspectives have helped me over many hurdles.

Finally, I thank all of my friends and family who have supported and encouraged me every step of the way. To my parents: thank you for everything!

VITA

October 14, 1984	Born - Mansfield, OH, USA
June - September 2004	Summer Hire, Air Force Research Laboratory, Dayton, OH
August - December 2005	Engineering Co-op, Toyota Technical Center, Ann Arbor, MI
June 2007	B.S. Mechanical Engineering, The Ohio State University, Columbus, OH
October 2007 - September 2008	Distinguished University Fellow, The Ohio State University, Columbus, OH
October 2008 - Present	Graduate Research Assistant, The Ohio State University, Columbus, OH

FIELDS OF STUDY

Major Field: Mechanical Engineering

Studies in:

Robotics	Professor E.R. Westervelt
Mechanical Design	Professor G.L. Kinzel
Control Systems	Professors C.H. Menq ,
	V.I. Utkin
Strength of Materials	Professor H.R. Busby
Measurement Systems	Professor E.O. Doebelin

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Bipedal Robots

The incredible variety of maneuvers performed by humans and other animals has captivated the interest of researchers for many years. Bipedal animals are of particular interest, due to the fact that humans themselves are bipeds. Bipeds also represent the simplest form of a multi-legged animal, reducing the complexity associated with additional joints and limbs while maintaining valuable locomotion capabilities.

The tasks of planning and controlling maneuvers are often completed with little conscious effort on the part of the animal. The internal processes used to complete these tasks give rise to many fundamental questions regarding how they are performed. The primary tools used to investigate legged locomotion include biological experimentation and analysis, numerical simulation of systems, and experimentation with robotic platforms. The use of experimental robotic platforms offers several benefits over other tools including a defined physical structure, easily obtainable physical parameters and state variables, complete access to the motor control process, and real-world demonstration of concepts. These benefits as well as others have led to the creation of a number of bipedal robots for research purposes.

The use of robotic platforms in locomotion research presents a particular challenge with respect to control of the complete system. Legged robots often represent simplified versions of animals, with mechanical links used to mimic body parts including the legs and trunk of an animal while actuators take the place of muscles for providing power to the body parts. Even with a simplified mechanical structure, legged robotic systems can contain many links and actuators. Performing locomotive maneuvers requires that the motion of the robotic links be coordinated. This coordination is built upon a control strategy for achieving the desired locomotion.

1.2 Control Strategies for Legged Locomotion

The control strategy is a critical element for achieving robotic locomotion. The primary function is to plan the manipulation of the robotic links through activation of the actuators in a manner that will achieve the locomotive goals. The motions defined for the links will depend upon the characteristics of the locomotion desired, but the function will remain the same. The difficulty in performing this function is introduced through the requirement of physical stability of the system. Physical stability can be interpreted in a number of ways. A natural interpretation related to this work is the ability of a legged robot or animal to perform a maneuver in a manner such that continued locomotion is possible. While this description of stability is useful for a general understanding of locomotive goals, it lacks the precision necessary for direct integration into a control strategy. More rigorous definitions of stability have been developed and used to create control strategies capable of producing legged locomotion. In the following discussion of control strategies, the term 'system' will be used to refer to a generic bipedal animal or robot unless otherwise specified.

1.2.1 Static Stability and the Zero Moment Point

The most basic posture in which a system can be described as stable or balanced occurs when the links are held in a static configuration such as standing. When this situation occurs, the stability of the system can be described by the projection of the **center-of-mass** (CoM) onto the ground, defined as point P. The contact points between the feet and the ground describe a finite area called the **support polygon**. In a biped with flat rectangular feet, the support polygon will form a shape as depicted in Fig. 1.1. In a statically balanced configuration, the projection of the CoM on the ground will fall within the area defined by the support polygon. The system is statically stable due to the ground reaction forces on the feet balancing the gravitational forces and resultant moments. If the system is disturbed from such a configuration to a position where the point P is outside the support polygon, the ground reaction forces will no longer be capable of counteracting the gravitational forces. The system will fall unless corrective action is taken.

The static stability described by the location of point P within the support polygon is sufficient to develop a basic control strategy for bipedal locomotion. In a system with feet that achieve multiple points of contact with the ground such as those depicted in Fig. 1.1, the control strategy can consist of planning trajectories for the links such that point P remains in the support polygon throughout the maneuver. If the trajectories are performed at a low enough speed, system dynamics introduced by inertial effects can be neglected. The locomotion control process can be performed by transitioning support of the system from foot to foot while maintaining the static stability criteria. This control strategy, while achievable, does not produce locomotion that resembles the natural locomotion of animals. The principles of this strategy

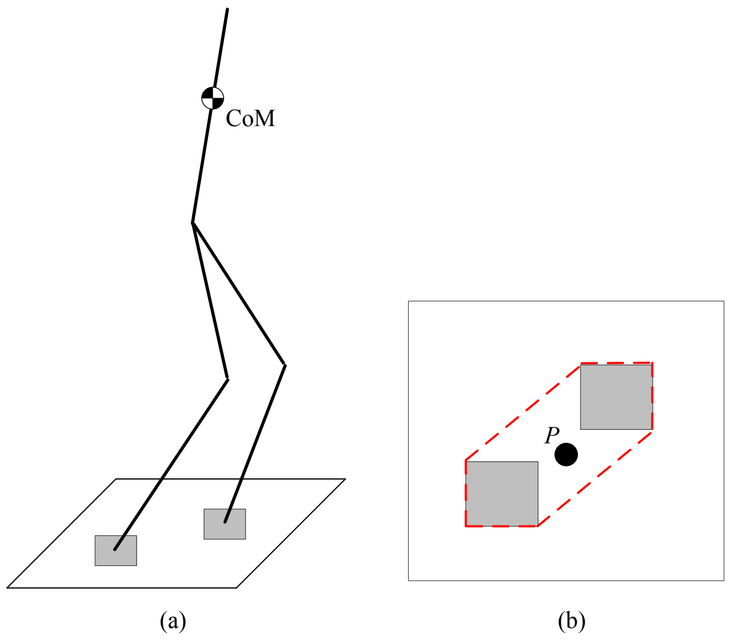

Figure 1.1: Simplified biped schematic for static stability. The link arrangement and CoM are shown in (a). The support polygon is depicted in a planar view in (b) using a dashed red line, with the projected center of mass P represented by the black dot. The rectangular feet are shaded grey in both.

may be extended to provide more flexibility to the control strategy for planning stable motions. The **zero-moment point** (ZMP) was developed in this manner.

Vukobratovic and Borovac [1] describe the ZMP in detail as it corresponds to development of legged locomotion control strategies. The ZMP can be defined as the point on the ground surface about which a net moment of zero is exerted on the system by gravitational, inertial, and ground reaction forces. This point can also be explained as the location on the ground surface where moments introduced by ground reaction forces on the foot (or feet) balance the moments produced by gravity and inertia. The inclusion of inertial effects in the definition of the ZMP allows for use of the point in explaining the stability of non-static situations in which the systems links are moving at an appreciable speed. In order for the system to have stability (also called dynamic balance), the ZMP must fall within the support polygon. The ZMP can be implemented in control strategies for planning locomotion to ensure system balance throughout a maneuver. Maintaining the ZMP within the support polygon can be achieved through control of the link trajectories used to execute a maneuver.

The introduction of the ZMP enabled the development of control strategies capable of performing basic locomotive maneuvers such as walking. The first bipedal robot to demonstrate the effectiveness of the ZMP for describing stability was WABOT-1 [2]. The control strategy for WABOT-1 maintained stability through planning joint trajectories that respected the ZMP location within the support polygon until the time of foot transition. At foot transition during a step, the robot was allowed to fall forward onto the opposite foot, ultimately returning to a new stable posture. This brief instability defines the walking as a quasi-static maneuver. In a quasi-static maneuver, statically-based stability is maintained through the majority of the maneuver with short phases of instability.

A more recent demonstration of the ZMP-based control strategy is given by the humanoid robot ASIMO [3]. ASIMO has demonstrated a variety of maneuvers using this general control strategy, including walking, stair-climbing, and running. A similar level of performance has been achieved with the small entertainment robot QRIO [4]. The maneuvers demonstrated by these systems lack much of the fluid nature present in similar maneuvers performed by animals. The restrictions of a control strategy based on the ZMP cause this lack of fluidity. Achieving more natural-appearing locomotion requires a different approach to the control strategy.

1.2.2 Control Strategies Exploiting Dynamics

At their origin, control strategies based on the ZMP rely on the concept of maintaining static or quasi-static stability. In contrast to this, building a control strategy based upon dynamic considerations can produce a substantially different end result. Fully dynamic motion, in which the system is continuously outside the region of static stability, encompasses a much greater number of maneuvers. Introducing fully dynamic movement into a robot's repertoire of maneuvers expands the design space, offering greater performance potential and flexibility.

The improved performance and flexibility of robotic systems offered by the development of dynamic maneuvers are critical to realizing performance comparable to biological systems. Specifically, robots expected to perform tasks in the diverse unstructured environments frequented by humans must have maneuvering capabilities similar to those of humans. Actions such as running, jumping, stopping, starting, and rapidly changing direction are fundamental dynamic maneuvers exhibited by biological systems. The ability of legged robots to replicate these maneuvers presents a significant challenge.

Raibert sought to exploit the natural dynamics of legged systems to improve stability and performance of maneuvers [5]. A monopod robot was first used to demonstrate the possibility of producing maneuvers in which static balance is not required. The robot consisted of a single prismatic leg and a body housing controllers and actuators. Three dimensional hopping was produced using a relatively simple control strategy that exploits the mechanical structure of the robot. The controller separated the problems of performing a maneuver and maintaining balance. During the ground contact phase, the monopod's leg delivered thrust via a pneumatically-powered, prismatic actuator to drive the system through the following hop. Simultaneous to the delivery of thrust, a servo-controller was used to actively correct the orientation of the robot's body. In this manner, Raibert was able to produce a consistent hopping behavior. Furthermore, control over the characteristics of the hop maneuver including forward velocity and height was demonstrated through control of the leg position and the amount of thrust delivered. This control strategy was extended to a bipedal system to produce running. The biped demonstrated a planar run exhibiting stable, dynamic characteristics.

The robots designed by Raibert, while showing impressive performance and stability, lack realism when compared to legged animals, which rely on articulated legs. The mechanical design of these robots reflects modeling of legged systems using a **spring-loaded inverted pendulum** (SLIP) as described by Full and Koditschek [6]. While the SLIP model has been found useful in analyzing and controlling locomotion [7], it does not include dynamic effects introduced by articulated joints with massive links. Schmiedeler et al. [8] presented evidence that leg mass is an integral part of system dynamics when modeling quadrupedal locomotion. The limitations of prismatic actuators in representing animals become clear when considering maneuvers with extended flight phases where angular momentum becomes significant. Despite the simplifications, the SLIP model has been shown to contribute to developing dynamic running control strategies for an articulated monopod by Poulakakis and Grizzle [9]. The monopod relies on a control strategy using **hybrid zero dynamics** (HZD).

The HZD control strategy was developed by Westervelt et al. [10] for application to a special class of robots consisting of planar bipeds with one more degree-of-freedom (DOF) than the number of actuators. The 'planar' aspect of these bipeds refers to the fact that they operate only in the saggital plane. The dynamics of bipedal walking are modeled in two parts based upon whether one or both feet are in contact with the ground. These phases have a dynamic model associated with each, while the initial contact of the feet with the ground surface is modeled as a discrete, impulsive event occurring in an infinitesimally small time. The control strategy works by reducing the dynamic models to a single DOF hybrid zero dynamic model. The HZD model is then used to determine a stable periodic orbit that describes the behavior of the system during the walking maneuver. The characteristics of the HZD model are such that a stable periodic orbit of the reduced dynamic model correlates to a periodic orbit of the full dynamic model that can be stabilized through feedback control. The resulting controllers produced by the HZD-based control strategy have been demonstrated on multiple robotic systems. The planar biped ERNIE has shown the ability to perform dynamic, stable walking at a variety of speeds [11]. A similar biped robot, RABBIT, used the HZD control approach to perform a sequence of six running steps, revealing the capability of the control strategy to extend beyond walking [12]. The walking and running demonstrated by these systems resemble natural maneuvers much more closely than statically-based approaches.

The results of the HZD-based control strategy support the validity of the technique. However, the intensive mathematical analysis and modeling involved requires a significant time investment to produce locomotion. Additionally, the resulting controllers have restricted application to a particular type of system. The ability of animals to perform complex maneuvers suggests that less analytically intensive control strategies may be viable.

1.2.3 Intelligent Control Strategies

Intelligent control strategies have emerged as an alternative approach to analytically intensive methods. Antsaklis and Passino [13] describe intelligent control systems as a necessary development to address the increasingly complex nature of dynamic systems. They characterize intelligent control methods by inclusion of multiple levels for action planning, the ability to learn from past experience, or the ability to react to threats or challenges. Multiple levels of control suggest the use of a high-level intelligent control method in conjunction with low-level controllers for producing link motion. Neural networks, genetic algorithms, and fuzzy control are some of the primary tools of intelligent control that have seen application to robotic systems.

Neural networks are used to mimic the cognitive processes of animals. The elements of biological neural systems, neurons, are represented through programming objects. These artificial neurons are joined together to form a complex network capable of performing tasks including reinforcement learning and problem solving. Miller [14] applied a neural network to the problem of controlling dynamic walking in a humanoid robot. Given fixed low-level control strategies, the neural network was used to provide an adaptation mechanism for adjusting the maneuvers to produce stability. Continuous walking was not achieved; however, the ability of the neural network to improve the balance of the mechanism while maneuvering was demonstrated.

Genetic algorithms are based on the concept of evolution. Combinations of parameters for controlling a system can be considered as individuals in a population.

An initial set of candidate individuals are evaluated using a performance metric. The best performing individuals are selected to produce a new generation of individuals using combinations of their parameters. The process of performance evaluation and genetic recombination are repeated iteratively to locate a combination of parameters that achieve the desired goal. Genetic algorithms are able to search a large, multi-dimensional design space of parameters in an efficient manner. Curran [15] used a genetic algorithm in this manner to determine actuator command parameters for achieving a maximal height jump in an articulated, hopping leg constrained to vertical motion. Galloping quadrupedal motion was produced by Krasny through use of a genetic algorithm to search for appropriate control parameters to be implemented by low-level motor controls [16].

Fuzzy control allows the integration of heuristic control information into a structured control strategy. Heuristic information can be provided by a user to describe the appropriate control actions for a system for a variety of scenarios. This heuristic information is used to create a set of rules describing the action to be taken by the controller. The fuzzy controller examines the status of the system and implements the control rules that are applicable to the given situation. Some of the key advantages of fuzzy controllers are their ability to incorporate a user's knowledge, their applicability to complex, nonlinear systems, and their ease of implementation [17]. Fuzzy controllers have been implemented in robotic control strategies for determining control parameters for producing locomotion. Palmer [18] successfully used a fuzzy controller for producing trotting in a quadruped in numerical simulation. Marhefka [19] similarly used a fuzzy controller for a quadruped to produce galloping as well as starting and stopping. In these implementations, the fuzzy controller is used as a high-level controller for determining parameters for low-level motor controllers.

The demonstrated ability of intelligent control to produce locomotive behaviors in robotic systems encourages the application of such techniques to additional systems and maneuvers.

1.2.4 Jumping Control Strategies

Many of the bipedal systems already described have emphasized walking or running as the desired locomotion. Some of the systems have demonstrated these maneuvers experimentally. A number of projects have also been created to research the possibility of another basic legged maneuver: jumping. Jumping poses an attractive maneuvering capability for legged robots. The ability to overcome obstacles or rugged terrain is enhanced by the extended flight phase and foot clearance demonstrated in jumps in comparison to walking and running.

The control strategies that have been developed to produce jumping in biped robots typically exhibit one of two shortcomings: either the height of the jump relative to the size of the robot is unimpressive or the mechanical structure of the robot lacks biological realism. Nunez et al. [20] developed a control strategy for stable vertical jumping in a biped through use of the sliding mode control technique. The control strategy incorporates the ZMP concept to maintain the stability of the system by keeping the foot positioned directly below the CoM. The demonstrated results of the control scheme in simulation show the achievement of a jump in which the system's CoM reaches approximately 0.60 m. The system was composed of legs with a segment length of 0.30 m. The jump can be evaluated using the normalized jump height s, defined by Alexander [21] as

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

where h is the hip height at the top-of-flight and L_i is the leg segment length. The normalized jump height for this result is approximately $s = 2.0$. This estimate is a maximum based upon the height of the CoM in place of the height of the hip (which is not provided). This jump height is unimpressive in light of the abilities of animals to achieve normalized jump heights in the range of 2.5-5.5 (based on Alexander's simulations).

A humanoid robot simulator ROCOS was used by Hirano et al. [22] to validate a biped jumping control strategy using active impedance control and a simplified system model. The resulting jumps produced foot heights above ground of 0.05 meters. The normalized jump height cannot be assigned due to a lack of available data. Hosoda et al. [23] achieved dynamic jumping capabilities in a pneumatically actuated biped-type robot through use of a feedforward controller. Two additional legs were included in the robot to provide lateral stability. The robot demonstrated jumps of 0.12 meters with a standing body height of 0.90 meters. However, the control strategy did not provide stability, and the robot was observed to fall forward or backward during the jump maneuver. A similar pneumatic actuation approach was successfully implemented in a robot designed specifically for jumping; Mowgli [24], a bipedal robot, demonstrated impressive jumps using an open-loop motor command control strategy. The robot includes a foot link of similar length to the leg links, providing a large support polygon during ground contact at the beginning and end of the jump. This large foot size does not reflect the structure of animals such as humans and other primates. The control strategy thus has minimal requirements for achieving stability. Despite these and other projects focused on jumping legged robots, jumping with comparable performance to animals has not yet been demonstrated in a biped robot with a biologically realistic mechanical structure.

1.3 Objectives

The primary objective of this work is development of a control strategy capable of driving a biologically realistic biped robot with articulated legs through stable jumping maneuvers. The control strategy is split into two functional levels: planning and performance of the maneuver. The planning level (high-level) of the controller implements a fuzzy controller similar to that used by Palmer and Marhefka and motivated by their results. The performance level (low-level) controller uses open-loop and closed-loop control over the robot's actuators to achieve the necessary link motions.

The biped robot used for modeling and experimentation is KURMET, developed by Knox as an experimental platform for dynamic bipedal maneuvers [25]. Hereafter, references to the 'system' will denote the biped KURMET unless otherwise specified. The control strategy will be formulated with the additional goal of remaining flexible and modular enough to apply to different robotic systems and dynamic maneuvers other than jumping. The primary objective is accomplished through the following intermediate objectives.

- Develop a dynamic model of the system that captures all the critical characteristics affecting performance

- Implement the dynamic model in simulation for development and testing of the control strategy

- Define the link motions required to produce jumping and develop low-level controllers for executing these motions
- Define control parameters sufficient for describing a jump of specified height and forward motion
- Develop a high-level fuzzy controller capable of providing appropriate control parameters
- Implement the control strategy in simulation and in hardware for proof-of-concept

The development of the control strategy, the primary objective, serves as a foundation for investigating more loosely defined concepts related to this work. It is intended that the control development process and resulting experimentation will provide insight into practical aspects of legged locomotion, motor control theory, and bipedal system dynamics.

1.4 Motivation

The motivation for this work is the potential benefit to a variety of applications related to legged locomotion. A thorough understanding of the nature of dynamic maneuvers, from the standpoint of both motor control and physical performance, can be used to guide development of human prosthetics, rehabilitation programs and devices, and more advanced robotic systems. A robot capable of dynamic maneuvers could be used to analyze dynamic locomotion in a systematic way and to explore solutions to challenges in these fields.

The capability of modern medicine to save human lives even in cases of traumatic injuries has increased the demand for prosthetic limbs. The design of prosthetic limbs relies heavily on an understanding of the dynamics and kinematics of human motion. Improvements in the design of these devices directly translates to improved user experience and satisfaction. Examination of the physical characteristics of dynamic locomotion in a robot, including joint speeds and structural forces, could help to precisely define performance requirements for future prosthetic devices.

The field of physical therapy and rehabilitation has seen a large growth in the use of robotic or otherwise mechanical devices to assist in the recovery process. These devices can serve functions such as guiding limbs through defined trajectories to reestablish or improve motor control [26], or counteracting gravitational forces to reduce the muscular strength required for rehabilitation programs [27]. An improved understanding of human motor control processes could be used to refine rehabilitation programs to work in harmony with biological responses to treatment. Kinematic and dynamic descriptions of dynamic legged maneuvers could direct the design of more advanced rehabilitation-assistance mechanisms. The control strategy developed in this work could help to validate theoretical models of motor control, while the resulting dynamic maneuvers supply physical descriptions of legged locomotion.

An improved understanding of practical issues in controlling legged systems supports the development of future robots with improved performance capabilities. As robots approach the performance capabilities of humans, their application can be expanded to include completing tasks hazardous or strenuous for humans. Currently, robots are being developed for use in high risk environments, such as disaster areas, to supplement human efforts in search-and-rescue missions. One example of this is the wheeled robot developed by Tsukagoshi et al. [28]. This wheeled robot uses a pneumatic cylinder to produce a jump maneuver for traversing difficult terrain. The quadrupedal robot BigDog [29], developed for military use in carrying heavy loads through battle environments, has demonstrated robust locomotion over irregular terrain. Rescue robots and legged systems such as BigDog are examples of how robots can be used to reduce risk and toil on the part of humans. Practical problems encountered during the development of the dynamic robot maneuvers could inform the design of systems that share these objectives.

1.5 Organization of Thesis

Chapter 2 describes the development of a mathematical model of the biped robot used in this work. The robot is described with emphasis placed on the unique actuation scheme used. Modeling of the primary elements of the system includes the robotic links, the actuators, and the electronic components. The software used for the numerical simulation is then described. Specific challenges encountered during the model development are found in the chapter's conclusion.

The focus of Chapter 3 is the control strategy. The overall strategy is split into two functional levels, consisting of a high- and low-level controller. The motivation for these functional levels is presented along with the algorithms used for each. The low-level controller is explained with respect to the biological principles used as inspiration. The high-level fuzzy controller is developed from its base principles.

The training algorithm for the fuzzy controller is the emphasis of Chapter 4. The criteria for determining acceptable performance are created to reflect the stability and robustness desired in the resulting controller. The training algorithm is broken into specific cases to reduce the difficulty of determining appropriate changes to the control parameters. The results of the training algorithm are examined. The heuristic knowledge gained from the training algorithm and development of the control strategy is presented. The chapter concludes with interpretations of the fuzzy training algorithm and a brief summary.

The results of the control strategy as implemented in simulation and the experimental biped are presented in Chapter 5.

The final chapter, Chapter 6, contains conclusions on the control strategy described in this thesis and suggestions for future work.

CHAPTER 2

MODELING AND SIMULATION

2.1 Introduction

This chapter begins with an overview of the complete robotic system and a discussion of the unique actuator design used in this work. An explanation of the notation used is then provided. Following this is the development of the modeling for the actuators, electrical system, and environmental interactions, with respect to their role in the complete system model. The software and dynamic engine used to perform the simulation are described, and the chapter concludes with a discussion of several modeling challenges and a brief summary.

2.2 Complete System

This work focuses on application to a specific bipedal experimental platform, KURMET, seen in Fig. 2.1. Developed by Knox [25], the system consists of a 5-degree-of-freedom (DOF) planar bipedal robot with series-elastically actuated, articulated legs. The actuators are physically located within the torso and are connected to their respective links via steel cables and pulleys. The design is based upon previous generations of biped robots and legged systems, including a single-leg prototype system

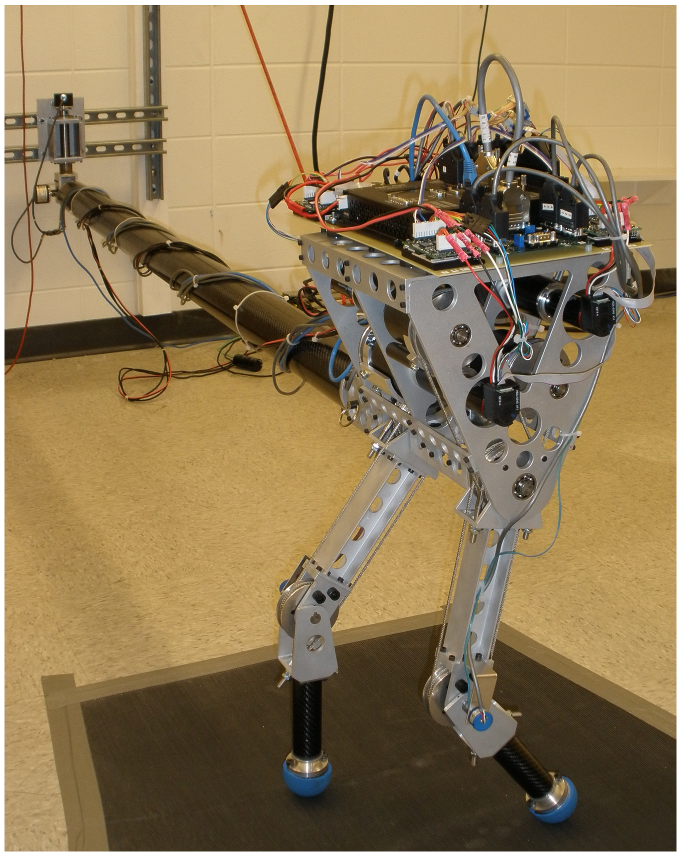

Figure 2.1: Experimental biped robot KURMET. The control electronics, including the control board and power amplifiers, are shown mounted on the top of the torso.

[30] and the biped ERNIE [31]. KURMET was developed to provide the dynamic performance capabilities required for maneuvers such as running and jumping. Control is provided by an on-board Galil Motion Control board working in conjunction with a remote personal computer running real-time Linux.

The robot is mounted on a carbon-fiber boom collinear with the hip axes, set at a height to be horizontal with full leg extension and the feet contacting the ground. The boom functions to restrict the motion of the robot to a spherical surface. The length of the boom (2.08 m), in comparison to the size of the robot (0.5 m leg length), causes

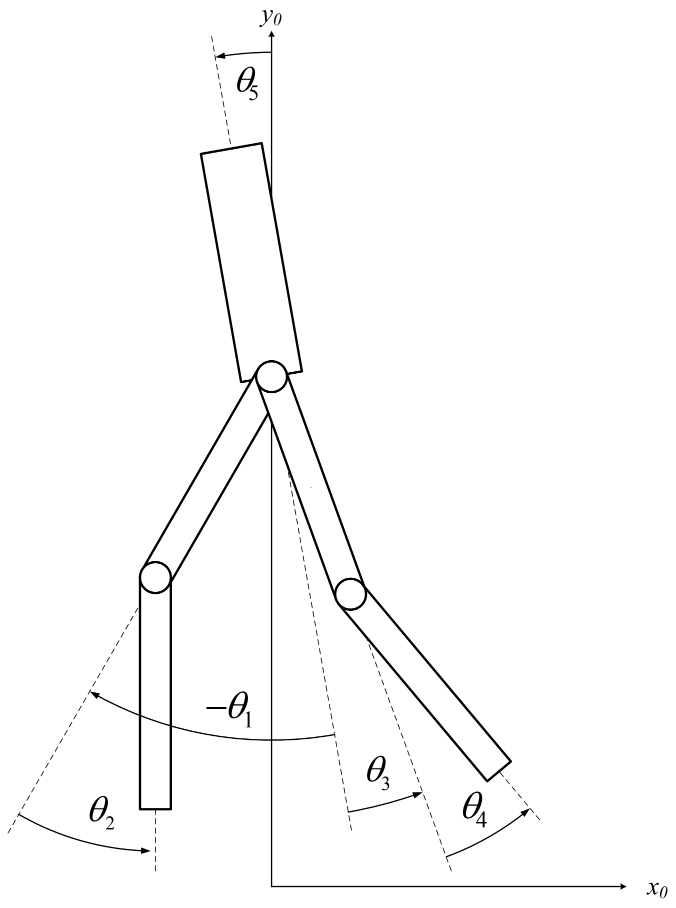

Figure 2.2: System model in the saggital plane with the boom and boom support not included. Not to scale.

the curvature of the spherical surface to be low enough to approximate a restriction to the saggital plane. Revolution of the torso about the boom axis is left free, allowing body pitch. Restriction to the saggital plane, or more precisely the spherical surface to which the saggital plane is always tangent, prevents the robot from yawing and rolling.

The robot is modeled using five rigid links connected by revolute joints: the torso, two thighs, and two shanks, seen in Figs. 2.2 and 2.3. An additional rigid link is

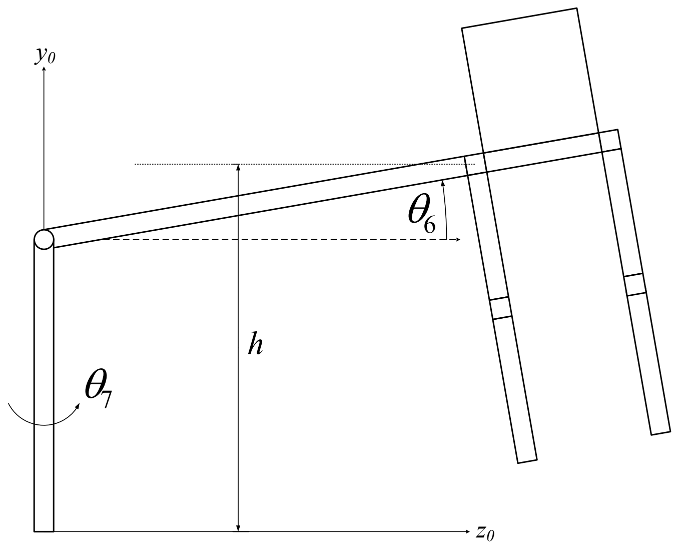

Figure 2.3: System model in the coronal plane. Note that the vertical boom support aligned with the y-axis provides a kinematic connection between the boom and the ground, but is not included in the system dynamics. Not to scale.

used to model the boom. Each of the two legs is composed of geometrically identical thigh and shank links, joined at the knee and attached to the torso at the hip. All of the rigid links include mass, inertia, and geometric properties estimated from CAD models of the experimental hardware, found in Appendix A. Four series-elastic actuators drive the two hip joints and two knee joints. Current-control amplifiers supply the actuators with power. Control of the actuators is performed by specifying the commanded current supplied to each by the amplifiers.

2.2.1 Overview of Actuator Design

The unidirectional series-elastic actuator (USEA) design described in this work was developed to provide the explosive leg power required for performing dynamic maneuvers [25]. Series-elastic actuators (SEA) can generally be described as a traditional actuator (such as a pneumatic piston or DC motor) placed mechanically in series with an elastic element, which is ultimately attached to the driven link. The elastic element serves the purpose of decoupling the torque and speed output of the traditional actuator from the torque and speed delivered to the actuated link. Additionally, the elastic element allows the storage of energy in the form of potential energy, which can be delivered to the actuated link in a manner such that the performance exceeds the capabilities of the traditional actuator acting alone.

The USEA design used here represents a special class of SEA, which only realizes SEA behavior when actuated in one direction. The direction of the actuator chosen for series-elastic behavior is selected to allow deflection of the elastic elements during thrusting of the legs. When driven in the opposite direction, the actuator design causes it to behave as though the actuated link is directly attached to the traditional actuator, at which point the actuator is said to be operating as a direct-drive actuator (DDA).

The USEA consists of three primary components: an elastic element, a brushless DC motor with attached gearbox, and the unidirectional hardware. The elastic element used in this work is a spiral torsion spring. The brushless DC motor and gearbox are commercially available products: the Maxon EC-30 Powermax 200W motor and matching gearbox. The unidirectional hardware consists of a mechanical hardstop pair which is engaged when the actuator is operated opposite the thrust direction. Details of the dynamics of this actuator design will be clarified during the development of the actuator model below.

2.2.2 Notation

Simplified system diagrams, shown in Figs. 2.2 and 2.3, depict the physical arrangement of the robot links with joint angles shown referenced to their zero positions. The variables used to describe the state of the robot are θ_1, θ_2, θ_3, θ_4, θ_5, θ_6, and θ_7, and their associated time derivatives ($\dot{\theta}_1$, $\ddot{\theta}_1 \ldots$). Additional variables used in the simulation and control include the motor positions $\theta_{m,1}$, $\theta_{m,2}$, $\theta_{m,3}$, and $\theta_{m,4}$ and their time derivatives, with subscripts correlated to the joint state variables. The height above ground of the hip axis at its connection point with the boom is represented by h; this value can be kinematically calculated based upon the state variables and link lengths. Physical constants characteristic of the actuators are listed in Table 2.1 with experimental values provided. The equivalent motor inertia and damping, $J_{m,eq}$ and $B_{m,eq}$, are approximations of the combined values for both the motor and gearbox, which are used in modeling for simplification. The motor and gearbox damping cannot be defined precisely, so the equivalent motor damping is set to be the same order of magnitude as the equivalent motor inertia. The commanded motor currents, as determined by the control laws, are represented by $i_{c,1}$, $i_{c,2}$, $i_{c,3}$, and $i_{c,4}$. Torques produced by the actuators are denoted by τ_1, τ_2, τ_3, and τ_4.

This work uses the standard measurement units defined by the MKS system. Unless otherwise noted, the following units are assumed for each type of measure (including state variables and control parameters): length - meter, mass - kilogram, time - second, angle - radians, current - amp, resistance - ohm, velocity - meters per second, angular velocity - radians per second, force - Newton, torque - Newton meter.

Table 2.1: Actuator System Physical Constants

Symbol	Physical Constant	Experimental Value
J_m	Motor Inertia	$3.33\times10^{-6}\text{kg}\cdot\text{m}^2$
B_m	Motor Damping	N.A.
J_g	Gearbox Inertia	1.40×10^{-6} $\text{kg}\cdot\text{m}^2$
B_g	Gearbox Damping	N.A.
$J_{m,eq}$	Equivalent Motor Inertia	4.73×10^{-6} $\text{kg}\cdot\text{m}^2$
$B_{m,eq}$	Equivalent Motor Damping	1.0×10^{-6} $\frac{\text{N}\cdot\text{m}\cdot\text{s}}{\text{rad}}$
R_m	Motor Winding Resistance	0.386 Ω
k_τ	Motor Torque Constant	2.76×10^{-2} $\frac{\text{N}\cdot\text{m}}{\text{A}}$
n_m	Gearbox Ratio	126
η_f	Forward Gearbox Efficiency	0.72
η_b	Backward Gearbox Efficiency	0.36
k_s	Actuator Spring Constant	30 $\frac{\text{N}\cdot\text{m}}{\text{rad}}$
V_{max}	Maximum Voltage	48.0 V
i_{max}	Maximum Current	12.0 A

2.2.3 Sensors

A number of sensors are included in the physical hardware for detection of state changes and to provide feedback to the controller. Shaft encoders attached to the boom and boom support monitor the variables θ_5, θ_6, and θ_7. Potentiometers are attached to all four actuated joints to sense θ_1, θ_2, θ_3, and θ_4. Shaft encoders, attached to the DC motors, are used to detect the motor states $\theta_{m,1}$, $\theta_{m,2}$, $\theta_{m,3}$, and $\theta_{m,4}$. A mechanical switch, integrated in the design of each foot, detects ground contact.

Sensor dynamics are not modeled as part of the simulation. The time scale of the sensor dynamics is assumed to be small enough relative to the control effort that the effects would be negligible. Additionally, the accuracy of the sensors is assumed to be high enough to neglect error introduced through sensor performance. The state variables defined above are used directly in the control algorithm and dynamic engine for simulation purposes.

The foot contact switch is modeled in simulation through detection of foot penetration into the ground. The foot height in simulation is continuously calculated using the kinematic structure of the links and the Denavit-Hartenburg convention as described by Spong [32]. Homogeneous transformation matrices are used to describe the coordinate transformations between the links. The link coordinate frames, as well as two additional coordinate frames labeled with the subscript t and p for the torso and pelvis, respectively, are shown in Fig. A.1 in Appendix A. The link coordinate frames are numbered in correspondence to the joint variables defined in Figs. 2.2 and 2.3. The point $\bar{p}_f$ defining the position of the foot in the coordinate frame of the shank (for both legs) is

$$\bar{p}_f = (l_{link}, 0, 0, 1)^T, \tag{2.1}$$

where l_{link} is the length of the leg links (0.25 m) and the coordinates form a 4 x 1 vector. The fourth element of the point is included to allow the matrix multiplication required and has no physical sense. The homogenous transformation matrix T_b^a used to transform a point in coordinate frame b to coordinate frame a can be defined as

$$T_b^a = \begin{pmatrix} cos(\theta_b) & -sin(\theta_b)cos(\alpha_b) & sin(\theta_b)sin(\alpha_b) & l_b cos(\theta_b) \\ sin(\theta_b) & cos(\theta_b)cos(\alpha_b) & cos(\theta_b)sin(\alpha_b) & l_b sin(\theta_b) \\ 0 & sin(\alpha_b) & cos(\alpha_b) & d_b \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{2.2}$$

where α_b is the angle from z_a to z_b in the plane normal to x_b, l_b is the distance between z_a and the origin of coordinate frame b measured perpendicular to z_a, d_b is the distance from the origin of frame a to the intersection of x_b and z_a, and θ_b is the angle from x_a to x_b in a plane normal to z_a (equal to the joint angle for the numbered joints). All angular measurements follow the right hand rule. The global coordinate frame (x_0, y_0, z_0) is defined at the base of the boom support, with axes as shown in Figs. 2.2, 2.3, and A.1. The cumulative transformation matrix from the shank link coordinate frame to the global coordinate system is then defined as T_2^0 for the right leg and T_4^0 for the left leg. The torso and pelvis coordinate frames are required to calculate the cumulative transformation matrices, as they are used to define the kinematic relationships between the torso link and each of the thigh links. The transformation matrices that define the relationships between the additional coordinate frames are T_t^5, T_p^t, T_1^p, and T_3^p, which can be calculated using Eq. 2.2. The two required transformation matrices can then be calculated by matrix multiplication as

$$T_2^0 = T_7^0\, T_6^7\, T_5^6\, T_t^5\, T_p^t\, T_1^p\, T_2^1 \tag{2.3}$$

and

$$T_4^0 = T_7^0\, T_6^7\, T_5^6\, T_t^5\, T_p^t\, T_4^p\, T_3^4. \tag{2.4}$$

The position of the right foot in the global coordinate frame $\bar{p}_{f,r}$ is then defined as

$$\bar{p}_{f,r} = T_2^0\, \bar{p}_f. \tag{2.5}$$

Similarly, the left foot position in the global coordinate frame $\bar{p}_{f,r}$ is

$$\bar{p}_{f,l} = T_4^0\, \bar{p}_f. \tag{2.6}$$

The coordinates of $\bar{p}_{f,r}$ and $\bar{p}_{f,l}$ are of the form $(x_0, y_0, z_0,1)^T$ in the global coordinate frame, where the fourth element is included as a result of the required matrix multiplication. The foot is considered to be in contact with the ground if the foot height y_0 in the global coordinate frame, noted as $h_{f,r}$ and $h_{f,l}$ for the right foot and left foot respectively, is less than or equal to zero.

2.3 Actuator Model

Consistency between the numerical simulation and the physical hardware requires an accurate model of the actuators to reveal the dynamics and performance limitations. The unidirectional series-elastic actuators used in this work present unique modeling challenges.

The actuator modeling and dynamic simulation are performed within the control algorithm to allow for explicit manipulation of the many characteristics unique to unidirectional series-elastic actuators. The dynamics are simulated with a second-order Euler integration scheme performed at every control step. The control step size is specified as 0.1 millisecond.

2.3.1 Parallel Actuation

The location of all actuators within the torso of the robot necessitates use of a parallel actuation scheme. In a typical application, actuators driving revolute joints produce equal and opposite torques between the two links comprising the joint, as in Fig. 2.4(a). This method of actuation is referred to as serial actuation because torques act between serially connected links. Parallel actuation, as used in this work, does not by default produce torque between serially connected links. The torque produced by a hip actuator is developed between the torso and the thigh link as in serial actuation; however, the torque produced by a knee actuator is developed between the shank link and the torso, as opposed to the shank link and the thigh link as with serial actuation. The parallel actuation scheme is depicted in Fig. 2.4(b). This subtle difference necessitates the use of absolute angles of the leg links relative to the torso for calculating the actuator dynamics, as described in the following sections.

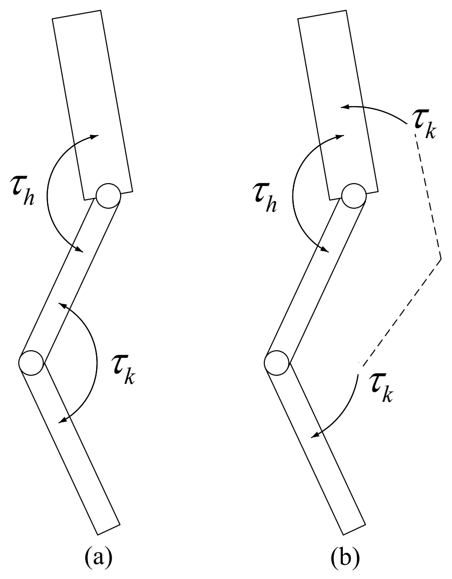

Figure 2.4: Serial (a) vs parallel (b) actuation.

The simulation software used, **RobotBuilder**, was designed assuming a serial actuation scheme. As a result, it is necessary to manipulate the torques produced by the actuators before supplying them to **RobotBuilder** for inclusion in the dynamic simulation. The calculated joint torques are modified to the equivalent values for a serially actuated mechanism $\tau_{\mathrm{i},ser}$ as

$$\tau_{\mathrm{i},ser} = \begin{cases} \tau_{\mathrm{i}} + \tau_{\mathrm{i}+1} & \text{for i=1,3} \\ \tau_{\mathrm{i}} & \text{for i=2,4,} \end{cases} \tag{2.7}$$

where i represents the actuated joint number. The effect of this manipulation is that the torque developed by the knee actuators is ultimately applied between the shank links and the torso, as desired.

2.3.2 Basic Series-Elastic Actuation

The torque developed by a series-elastic actuator is calculated based upon the net deflection of the spring and the stiffness of the elastic element k_s. The deflection of the spring is determined from the relevant link position and the output shaft of the gearbox, which is dependent on the motor position and the gearbox ratio n_m. Using the notation of this work, the basic equation for torque developed at the ith joint is

$$\tau_{\text{i}} = \begin{cases} k_s(\frac{\theta_{m,\text{i}}}{n_m} - \theta_{\text{i}}) & \text{for i=1,3} \\ k_s(\frac{\theta_{m,\text{i}}}{n_m} - \theta_{\text{i}} - \theta_{\text{i}-1}) & \text{for i=2,4.} \end{cases} \tag{2.8}$$

It is important to note that the $\theta_{\text{i}-1}$ term is included in the knee actuators to calculate the torque based on the absolute angle of the shank link relative to the torso (to which the knee actuator is rigidly attached).

2.3.3 Unidirectional Characteristics

The inclusion of an elastic element in the actuator, while offering benefits related to power performance, introduces potential issues with respect to position control of the joint angles; the large compliance in the drive train makes position control more difficult than with a less compliant actuator, such as a direct drive. Additionally, a compliant actuator can allow for oscillation in the joint position. To counter these problems, a design was implemented to make the compliant element active only during thrusting of the legs. This unidirectional approach is realized through the inclusion of a hardstop in the actuator, based on a design revision made to the single-leg prototype to address similar issues in Curran's experimental work [33]. The actuator hardstop is designed to place a preload torque τ_p on the elastic element by forcing an initial angular deflection; this prevents additional deflection until the thrust torque exceeds the preload torque. During motion opposite the thrust direction, the hardstop is engaged, making the elastic element inactive. During this time, the USEA behaves as a DDA.

The engagement of the hardstop creates a contact problem in the hardware of the actuator. This contact location is padded with an elastomer to reduce impacts. The contact is modeled using a linear spring and damper model in parallel, with the stiffness k_c of the contact spring set an order of magnitude higher than the actuator's spring element ($300\ \frac{\text{N}\cdot\text{m}}{\text{rad}}$). The damping constant B_c is set to produce approximately critical damping ($4.477\frac{\text{N}\cdot\text{m}\cdot\text{s}}{\text{rad}}$).

The torque in Eqn. 2.8 is modified to include a model of the hardstop contact as well as the preload. The calculation of the torque produced by the actuator is split into the two primary regimes mentioned above: series-elastic actuation and direct drive actuation. The two cases are distinguished by the inequalities

$$\begin{aligned} &\theta_{\text{i}} < \frac{\theta_{m,\text{i}}}{n_m} && \text{for i=1,3} \\ &\theta_{\text{i}} + \theta_{\text{i}-1} > \frac{\theta_{m,\text{i}}}{n_m} && \text{for i=2,4.} \end{aligned} \tag{2.9}$$

If the inequality does not hold, the actuator is considered to be in the DDA regime (i.e. the actuator is being operated opposite the thrust direction). The actuator torque is then calculated as

$$\tau_{\text{i}} = \begin{cases} k_c(\frac{\theta_{m,\text{i}}}{n_m} - \theta_{\text{i}}) + B_c(\frac{\dot{\theta}_{m,\text{i}}}{n_m} - \dot{\theta}_{\text{i}}) & \text{for i=1,3} \\ k_c(\frac{\theta_{m,\text{i}}}{n_m} - \theta_{\text{i}} - \theta_{\text{i}-1}) + B_c(\frac{\dot{\theta}_{m,\text{i}}}{n_m} - \dot{\theta}_{\text{i}} - \dot{\theta}_{\text{i}-1}) & \text{for i=2,4.} \end{cases} \tag{2.10}$$

If the inequality of Eq. 2.9 holds, the actuator is considered to be in the SEA regime. This regime is further divided into two cases to account for the preload torque. The

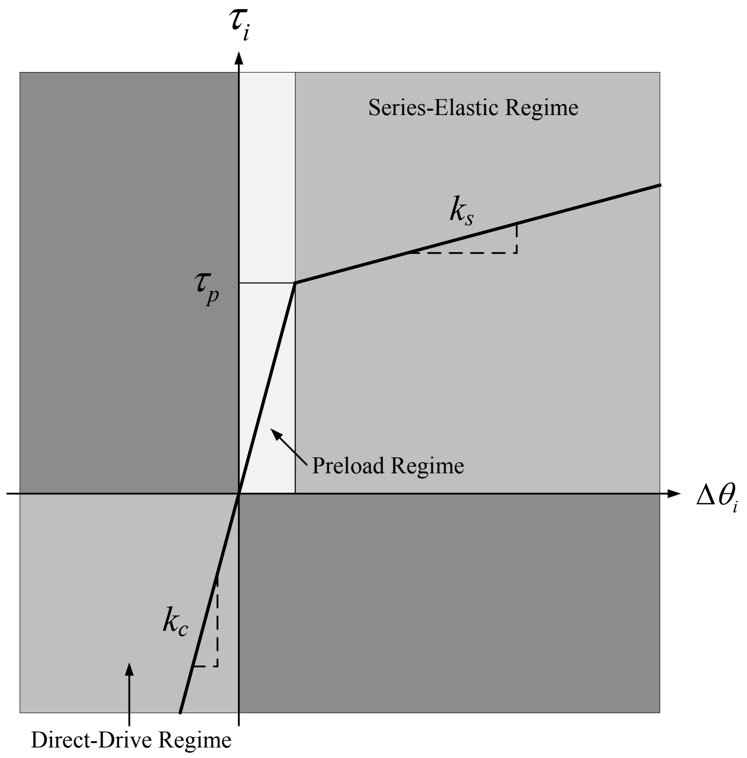

Figure 2.5: Static force-displacement diagram of USEA. The ordinate represents the actuator deflection $\Delta\theta_{\mathrm{i}}$ and the abscissa represents the joint torque τ_{i}.

cases are distinguished by the additional inequalities

$$|\frac{\theta_{m,\mathrm{i}}}{n_m} - \theta_{\mathrm{i}}| < \frac{\tau_p}{k_c} \qquad \text{for i=1,3} \tag{2.11}$$

$$|\frac{\theta_{m,\mathrm{i}}}{n_m} - \theta_{\mathrm{i}} - \theta_{\mathrm{i}-1}| < \frac{\tau_p}{k_c} \qquad \text{for i=2,4.}$$

If the inequality of Eq. 2.11 is valid, the actuator is within the preload range of the actuator. This is modeled by using a spring constant equal to the contact stiffness for the torque calculation until the preload value is exceeded, as depicted in the force-displacement diagram of Fig. 2.5.

The large magnitude of the contact stiffness relative to the spring constant of the elastic element prevents large deflections of the elastic element until the preload torque is overcome. The damping coefficient B_c is included in the preload regime. The torque in the preload range is thus calculated as

$$\tau_{\mathrm{i}} = \begin{cases} k_c(\frac{\theta_{m,\mathrm{i}}}{n_m} - \theta_{\mathrm{i}}) + B_c(\frac{\dot{\theta}_{m,\mathrm{i}}}{n_m} - \dot{\theta}_{\mathrm{i}}) & \text{for i=1,3} \\ k_c(\frac{\theta_{m,\mathrm{i}}}{n_m} - \theta_{\mathrm{i}} - \theta_{\mathrm{i}-1}) + B_c(\frac{\dot{\theta}_{m,\mathrm{i}}}{n_m} - \dot{\theta}_{\mathrm{i}} - \dot{\theta}_{\mathrm{i}-1}) & \text{for i=2,4} \end{cases} \tag{2.12}$$

until the inequality of Eq. 2.11 is invalid, at which point the torque equation becomes

$$\tau_{\mathrm{i}} = \begin{cases} k_s(\frac{\theta_{m,\mathrm{i}}}{n_m} - \theta_{\mathrm{i}}) + \tau_p & \text{for i=1,3} \\ k_s(\frac{\theta_{m,\mathrm{i}}}{n_m} - \theta_{\mathrm{i}} - \theta_{\mathrm{i}-1}) - \tau_p & \text{for i=2,4.} \end{cases} \tag{2.13}$$

The actuator range modeled by Eq. 2.13 matches the original series-elastic torque of Eq. 2.8, with the addition of the preload torque τ_p.

2.3.4 Joint Limits

The physical system incorporates mechanical stops on the joint angles θ_1 through θ_4 in both the positive and negative directions to prevent damage to components. The angular positions of these limits, represented by θ_j^s where the superscript s is the sign [+/-] and the subscript j is the joint [h-hip,k-knee], are included in Table 2.2. The mechanical limits include an elastomer pad for reduction in the impact force associated with contacting the joint limits, similar to the actuator hardstop discussed above. An identical contact model to the actuator hardstop is used to determine the contact torque $\tau_{c,\mathrm{i}}$ developed during joint limit contact, with modifications to include the angular joint limit,

$$\tau_{c,\mathrm{i}} = k_c(\theta_j^s - \theta_{\mathrm{i}}) - B_c\dot{\theta}_{\mathrm{i}}, \tag{2.14}$$

Table 2.2: Joint Limits

Limit	Value
θ_h^+	1.232 rad
θ_h^-	-1.232 rad
θ_k^+	2.175 rad
θ_k^+	-0.103 rad

where the appropriate value of θ_j^s is taken for the ith joint. It is important to note that the contact torque $\tau_{c,\mathrm{i}}$ is developed directly between serially connected links in all four joints, in contrast to and independent of the previously defined actuator-based joint torque.

2.3.5 Electrical Components

The two electrical components of the actuator that are included in the model are the motors and the current-control power amplifiers. The modeling of these elements is identical to that developed for actuator simulation by Curran [15], to which the reader is referred for derivation and detailed explanation. A brief description of the governing equations is included here.

The current and voltage limitations of the components require attention to ensure that physical capabilities of the electronics are not exceeded within the simulation. The amplifier's terminal voltage v_t required to realize a commanded current $i_{c,\mathrm{i}}$ is calculated for the ith actuator as

$$v_t = i_{c,\mathrm{i}} R_m + k_\tau \dot{\theta}_{m,\mathrm{i}}, \tag{2.15}$$

where R_m is the winding resistance of the motor and k_τ is the torque constant. The actual terminal voltage $v_{t,a}$ produced by the amplifier with consideration of physical limitations is then described as

$$v_{t,a} = \begin{cases} -v_{max} & \text{if } i_{c,\mathrm{i}} < 0 \text{ and } v_t < -v_{max} \\ v_{max} & \text{if } i_{c,\mathrm{i}} > 0 \text{ and } v_t > v_{max} \\ v_t & \text{otherwise,} \end{cases} \tag{2.16}$$

where v_{max} is the maximum voltage the electrical system can deliver. The effect being modeled by Eqns. 2.15 and 2.16 is that of saturation of the amplifier due to back-emf. If the required terminal voltage v_t exceeds the voltage limit v_{max}, the current actually supplied by the amplifier $i_{a,\mathrm{i}}$ will not reach the commanded value $i_{c,\mathrm{i}}$ but rather will be restricted to

$$i_{a,\mathrm{i}} = \frac{v_{max}\mathrm{sign}(i_{c,\mathrm{i}}) - k_\tau \dot{\theta}_{m,\mathrm{i}}}{R_m}. \tag{2.17}$$

Furthermore, it is not possible for the amplifier to draw current in excess of the maximum current i_{max}. The actual current supplied by the amplifier is thus delivered as

$$i_{a,\mathrm{i}} = \begin{cases} -i_{max} & \text{if } i_{a,\mathrm{i}} < -i_{max} \\ i_{max} & \text{if } i_{a,\mathrm{i}} > i_{max} \\ i_{a,\mathrm{i}} & \text{otherwise.} \end{cases} \tag{2.18}$$

The model for the DC motors used in the actuators is based on a second-order equation of motion that captures the relevant electromechanical dynamics. The model includes the effective motor torque $\bar{\tau}_{m,i}$ and the effective damping $\bar{B}_m$ and is dependent on the critical speed ω_n^*.

$$J_{m,eq}\ddot{\theta}_{m,\mathrm{i}} = \bar{\tau}_{m,\mathrm{i}} - \bar{B}_{m,eq}\dot{\theta}_{m,\mathrm{i}} - \frac{\tau_\mathrm{i}}{\eta_{eff} n_m}, \tag{2.19}$$

where

$$\bar{\tau}_{m,\mathrm{i}} = \begin{cases} k_\tau i_{a,\mathrm{i}} & \text{for } \dot{\theta}_{m,\mathrm{i}} \le \omega_n^* \\ \dfrac{v_{max} k_\tau}{R_m} & \text{for } \dot{\theta}_{m,\mathrm{i}} > \omega_n^*, \end{cases} \tag{2.20}$$

$$\bar{B}_m = \begin{cases} B_m & \text{for } \dot{\theta}_{m,\mathrm{i}} \leq \omega_n^* \\ B_m + \dfrac{k_\tau^2}{R_m} & \text{for } \dot{\theta}_{m,\mathrm{i}} > \omega_n^*, \end{cases} \tag{2.21}$$

and

$$\omega_n^* = \frac{\mathrm{sign}(i_{a,\mathrm{i}})v_{max} - i_{a,\mathrm{i}}R_m}{k_\tau}. \tag{2.22}$$

The effective efficiency of the gearbox η_{eff} is calculated based upon the direction of the motor relative to the torque as

$$\eta_{eff} = \begin{cases} \eta_f & \text{for } \dot{\theta}_{m,\mathrm{i}}\tau_\mathrm{i} \geq 0 \\ \dfrac{1}{\eta_b} & \text{for } \dot{\theta}_{m,\mathrm{i}}\tau_\mathrm{i} < 0, \end{cases} \tag{2.23}$$

which includes the forward and backward gearbox efficiencies, η_f and η_b. The backward gearbox efficiency cannot be assigned from the physical hardware, so the value is taken as $\eta_b = \eta_f/2$.

This compact model of the motor dynamics allows for easy implementation in simulation and was experimentally verified to accurately describe the behavior of the current-control amplifier/DC motor system [33].

2.4 Environment Model

The environment model consists of ground contact and gravity. Additional environmental effects, including air resistance, are neglected for simplicity. The ground-foot contact is modeled at a single, axially located point on the end of each shank link using a linear spring and damper in the normal and planar directions, as well as constant coefficients of static and kinetic friction. The single-point contact model between the foot and the ground is used as the simplest representation of the behavior of the hemispherical foot. The values of the contact parameters are provided in

Table 2.3: Environmental Contact Parameters

Parameter	Value
Normal Stiffness	$7.5\times10^4\ \frac{\mathrm{N}}{\mathrm{m}}$
Normal Damping	$2.0\times10^3\ \frac{\mathrm{N\cdot s}}{\mathrm{m}}$
Planar Stiffness	$7.5\times10^4\ \frac{\mathrm{N}}{\mathrm{m}}$
Planar Damping	$2.0\times10^3\ \frac{\mathrm{N\cdot s}}{\mathrm{m}}$
Coefficient of Static Friction	0.75
Coefficient of Kinetic Friction	0.60

Table 2.3. The ground is modeled using a level, even surface. A more detailed description of the environment model can be found in the simulation software documentation provided by Orin and Rodenbaugh [34].

An inherent trade-off exists in contact simulation between the integration step size and the stiffness of the contact. Small integration steps are required to capture the behavior of contacts with high stiffness, while lower stiffness contact may allow unrealistic penetration of the contact pair and produce inaccurate results in the simulation. Excessively small integration step size reduces the speed of simulation and is thus undesirable. The parameters used in the simulation were selected to produce realistic behavior while not requiring simulation times of unmanageable length. The friction coefficients were chosen to represent the interaction between rubber and concrete.

2.5 Software/Dynamic Engine

The numerical simulation is performed in the **RobotBuilder** application developed by Rodenbaugh, which provides dynamic simulation of multi-body physics,

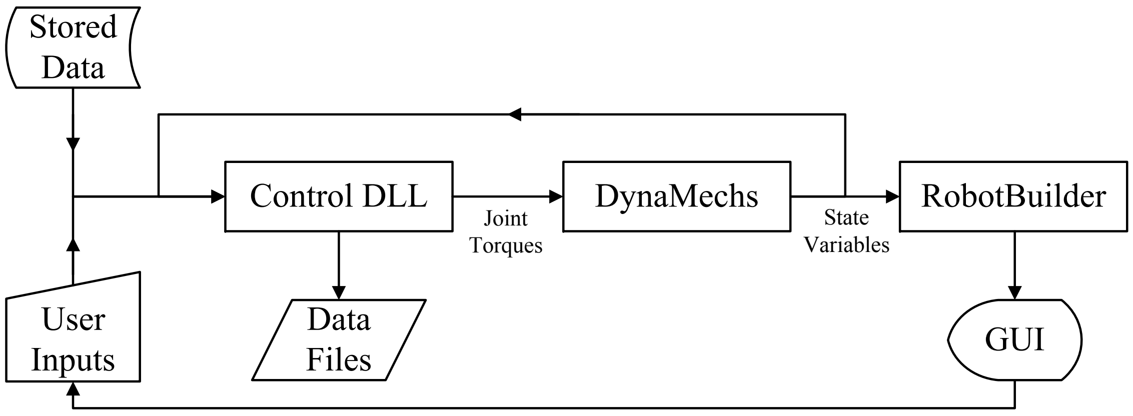

Figure 2.6: Basic file interaction diagram for simulation. Not all data transfers are included. Stored data and user inputs can include values such as control parameters or initial joint angles.

specifically robotic simulation [35]. The application itself offers a graphical user interface (GUI) that interacts with **DynaMechs**, a dynamic engine developed by McMillan [36], to numerically solve the equations of motion for the system and return the state variables of the system for graphical depiction in **RobotBuilder**. Control of the robot is implemented in the Visual C++ programming language, which is ultimately compiled and supplied to **RobotBuilder** as a dynamic link library (DLL) file. During simulation, the control DLL is called at every control step to calculate the control effort and the resulting actuator dynamics, to determine torques for the actuated joints, and to supply these to the dynamic engine. A basic flow chart of the simulation process involving the applications and control files is shown in Fig. 2.6.

2.6 Explanation of Specific Modeling Challenges

The development of the system model required investigation of several specific problems for which multiple solutions were considered. In particular, the contact problem presented by engagement of the hardstop in the unidirectional elements of the actuators was examined. Another significant problem that was discovered and addressed involved limitations on simulation run-time inherent to the software package used.

The engagement of the hardstop was identified during the design of the hardware and control strategy as a possible source of impact loading to the legs during operation. The impact force is reduced through the addition of elastomer pads to the metal contact points. The elastomer serves to distribute the contact more evenly across the mating parts and to increase the time of contact, reducing the peak force due to the impact. Additionally, the increased contact damping introduced by the elastomer reduces energy transfer between the mating parts.

The inclusion of the elastomer pad in the hardware, while improving the physical performance, does not remove the issue of modeling the contact for simulation. Two models were considered for capturing the effect of the contact on the system dynamics. The first model represented the contact with a linear spring and nonlinear damper in parallel, based upon the work of Orin [37] and Sung [38]. The contact torque developed in the hip actuator is determined as

$$\tau_{\mathrm{i}} = \lambda(\theta_{\mathrm{i}} - \frac{\theta_{m,\mathrm{i}}}{n_m})^N(\dot{\theta}_{\mathrm{i}} - \frac{\dot{\theta}_{m,\mathrm{i}}}{n_m}) + k_c(\theta_{\mathrm{i}} - \frac{\theta_{m,\mathrm{i}}}{n_m}), \tag{2.24}$$

where λ is the nonlinear damping coefficient and N is a constant. The benefit offered by the nonlinear model is in the initial torque calculated upon engagement of the

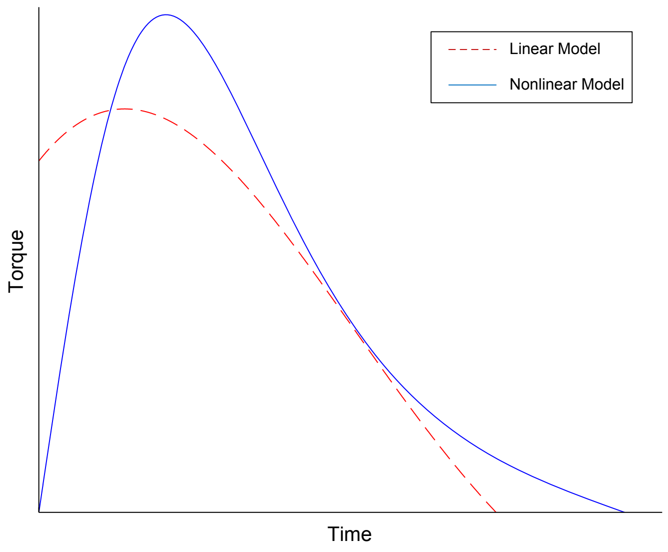

Figure 2.7: Qualitative comparison of contact torques for nonlinear and linear models. The dashed red line demonstrates the high contact torque developed upon initial impact in the linear model, while the solid blue line shows the smooth increase in the contact torque characteristic of the nonlinear model.

hardstop. The inclusion of the relevant angles (θ_{i} and $\theta_{m,\mathrm{i}}$) in the damping term causes the torque to increase in magnitude from zero with increasing depth of penetration, whereas exclusion of these angles would result in a non-zero torque being calculated immediately upon contact due to the damping term. The resulting impact torque profiles for the contact using the nonlinear model have a smooth rise to the peak value, as depicted qualitatively in Fig. 2.7 along with the linear model. The nonlinear term introduces an additional problem, however, for position control of the joints. The effective damping is low around the zero point of the actuator (where $\theta_{\mathrm{i}} \approx \frac{\theta_{m,\mathrm{i}}}{n_m}$) and reduces the controllability of the joint. This controllability issue is manifested when performing PD control of the joint angles, where low amplitude oscillations can arise as a result of the low effective damping.

The controllability problem presented by the nonlinear contact model was determined to counteract the benefit offered by low initial impact torque, and so a linear spring and damper contact model is used in Eq. 2.10. The high value of the impact torque at initial hardstop engagement, while not considered to accurately represent the physical system, occurs at a time during maneuvering when only the gross system dynamics are necessary. The linear contact model thus offers the required level of accuracy without introduction of further modeling or control problems. The damping coefficient B_c is included in the preload regime, described by Eq. 2.12, to further address the issue of joint oscillation around the zero point of the actuator.

The selection of the contact parameters k_c and B_c for the actuator hardstops and joint limits with the given linear contact model was based on the desired characteristics of the contact behavior. Penetration of the contact pairs was desired to be kept small relative to the range of motion of the joints, which drove the selection of the contact spring constant as an order of magnitude higher than the actuator spring constant. An excessively large contact spring constant could introduce complications due to the specified integration step size of 0.1 milliseconds for the actuator dynamics. The selected value of 300 $\frac{\mathrm{N{\cdot}m}}{\mathrm{rad}}$ is considered to balance these two considerations. The corresponding damping constant B_c was specified as 4.477 $\frac{\mathrm{N{\cdot}m{\cdot}s}}{\mathrm{rad}}$ to produce critical damping using an early approximation of the link inertia. The lack of more accurate determination of contact parameters is caused by the difficulty of predicting values for irregular geometries in the contact pair combined with the nonexistence of available experimental data for the elastomer material used.

Additional modeling challenges are introduced through the limitations inherent to the **DynaMechs/RobotBuilder** software. These challenges include a finite precision on the time variable used in the simulation and restricted integration options for calculating the system dynamics. The variable used within the software to represent the running time t is limited in precision to six digits. The integration step size used for the actuator dynamics is dependent on the precision of the running time, calculated as the difference in time between calls of the control algorithm. Sufficient precision is required in the integration calculation to maintain consistent behavior of the system dynamics. Loss of precision in the integration calculations can cause the system behavior to become unpredictable, due to the stiff contact pairs present in the actuators. A local time variable was introduced in the control algorithm to address this issue. The local time variable can be reset at convenient points during simulation to avoid loss of precision. The control strategy and the actuator dynamics use this local time variable in their associated algorithms.

The integration options provided by the software for calculation of system dynamics include Euler integration with fixed integration step sizes (both second and third order) and Runge-Kutta $4^{th}/5^{th}$ order adaptive integration. The Euler integration schemes, as implemented by the dynamic engine, have a minimum integration step size of one millisecond, below which the integrator fails. This limitation is insufficient for accurately calculating the system dynamics when modeling the stiff environmental contacts between the feet and the ground. The Runge-Kutta adaptive integration scheme is thus used to avoid the limitation of the Euler integrators and allow accurate simulation of the stiff contact pairs.

2.7 Summary

This chapter describes the dynamic system model used to simulate the robot for development of the control strategy. The complete system structure was desribed, followed by the notation for the state variables and physical constants quantifying the system. The characteristics unique to this system, those of a unidirectional series-elastically actuated robot using parallel actuation, were examined in depth. The software used to perform the simulation was then discussed. Finally, several specific problems encountered during the model development were presented and their solutions described.

CHAPTER 3

CONTROL STRATEGY

3.1 Introduction

The control strategy developed in this work is split into two functional levels, each of which has an associated algorithm. The functional levels are used to separate the tasks of planning and performing maneuvers so that each may be addressed independently. High-level control, also referred to as supervisory control, is used to determine the values of control parameters necessary to achieve the desired maneuver objectives using a fuzzy controller. The determined control parameters are then supplied to the low-level controller for implementation. The low-level controller consists of a state machine which segments the desired maneuver into discrete sequential phases and motor primitives that supply control laws for each phase. This chapter describes both the high-level and low-level control in detail, including discussion of the biofidelity of the techniques used, and concludes with a summary of the material presented.

3.2 Low-Level Controller: State Machine and Motor Primitives

The low-level controller provides real-time, continuous control of the actuators to achieve defined motion objectives. A state machine is used to detect the occurrence of specific events and implement the control law appropriate for the current phase of motion. The phases of motion and associated control laws are described using motor primitives.

3.2.1 Motivation for Motor Primitive Control

The inherent complexity of multi-joint legged systems gives rise to a significant control problem. Possible joint trajectories and their resultant multi-body motions create a design space of behaviors for producing a maneuver. The size of such a design space is so immense, due to the large potential number of variables describing the maneuver, that it is not computationally feasible to examine all of the space. However, the design space can be reduced in size by representing a maneuver as a finite sequence of parameterized motions, making the control problem much more tractable. The sequential motions, implemented in this work through motor primitives, provide a framework for controlling the desired maneuver that is defined by a finite number of parameters.

The concept of motor primitives has its roots in biological motor control theory. Experimental and theoretical work both support the concept that the central nervous system of animals controls the performance of complex behaviors through concatenation of multiple simple motions. Bizzi et al. [39] examined motor primitives experimentally using different locomotive behaviors of frogs and ultimately concluded that linear combinations of modular motions offer an experimentally and analytically sound model for motor control.

The application of such biological motor control theory to the control of robotic systems has precedent. The quadrupedal locomotion control developed by Palmer [18], Marhefka [19], and Krasny [16] is based on a comparable concept, where complex maneuvers such as running or galloping are segmented into simpler motions. Of higher relevance to this work, Schaal has suggested that control strategies implementing modular motor control offer a promising approach to the control of humanoid robots [40].

The modular nature of the motor primitive control approach offers adaptability to the overall control strategy developed in this work. Modification or substitution of the motor primitives used to achieve jumping could yield viable control for additional dynamic maneuvers. In particular, dynamic maneuvers similar in nature to the jumping performed with the motor primitives defined in this work may require only minor modifications. Such expansion of the dynamic repertoire of robotic systems could greatly improve performance capabilities with minimal investment in control development.

3.2.2 Structure of the State Machine/Motor Primitives

The state machine monitors a number of physical variables, as well as state variables, to determine the appropriate motor primitive at any given time. Each leg is handled independently to counteract ground contact timing discrepancies, which are introduced by the uneven foot contact as a result of the spherical restriction of the boom mount. The boom is mounted at a vertical height of 0.5 m, which corresponds to equal left and right foot contact with the ground at full leg extension and the torso upright ($\theta_1 : \theta_5 = 0$). When the boom is less than horizontal, $\theta_6 < 0$, the two feet are no longer equidistant from the ground for equal left and right hip and knee joint angles. Driving the left and right leg joints through the same joint trajectories will thus produce ground contact at different times. The control of each leg is handled independently to avoid commanding inappropriate control actions based on the position of the other leg.

The state machine operates in a cyclic manner, being reset at the top-of-flight (TOF) for every jump. The top-of-flight is sensed through a change in sign of $\dot{\theta}_6$. The actuated joints are commanded to specific angular positions at the top-of-flight, represented by $\theta_{h,tof}$ and $\theta_{k,tof}$ for the hip and knee joints. These angular positions were chosen to provide a defined starting and ending configuration for every jump to ensure consistent system states at the time of control parameter selection via the fuzzy controller.

The sequential motor primitives used in the state machine are shown in Fig. 3.1 with descriptions of the events used to detect transition between states. Many of these motor primitives were defined based on joint motions observable in the jumping of bipedal animals to which KURMET is structurally similar, particularly humans. The jumps performed by human subjects in [41] were examined to determine the qualitative goals of joint motions during the manuever. The simple motion goals of positioning the leg for ground contact, slowing the vertical motion of the body through compression of the legs, delivery of energy to the body through leg extension, and retraction of the legs to clear an obstacle are the basis of the motor primitives used in this work. These motor primitives provide the control objectives necessary for

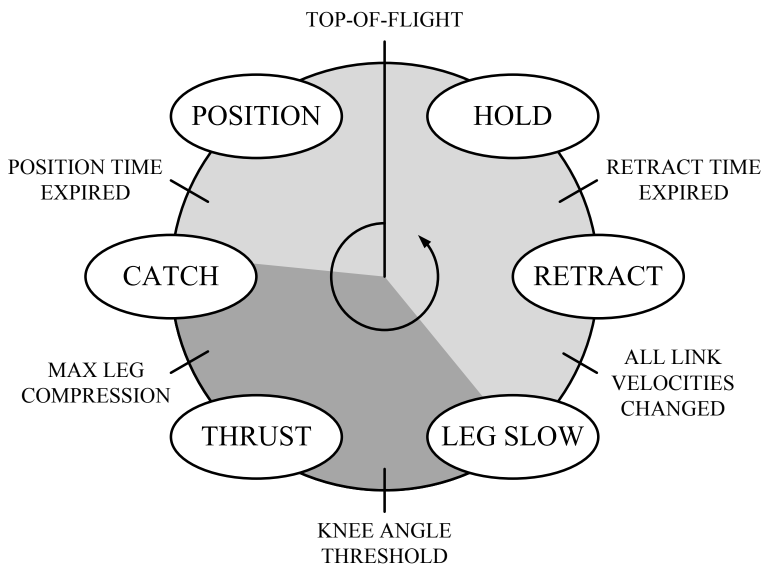

Figure 3.1: State machine diagram including motor primitives. The light grey portion of the state machine approximates the flight phase, while the dark grey approximates the ground contact phase.

completing the jump maneuver. Additional motor primitives were added as needed to counteract problems observed in the performance of the jumping in simulation.

3.2.3 Motor Primitive: Position

The **position** motor primitive serves the function of driving the hip and knee joints to the angular positions desired at ground contact during the descent of the robot in flight. The primitive begins at the detection of top-of-flight, determined by the state machine as a change in sign of $\dot{\theta}_6$, and is terminated upon the expiration of the motor primitive period $\Delta t_{p,1}$. The motor primitive period for **position** is calculated as

$$\Delta t_{p,1} = 0.75\sqrt{\frac{2(h_{max} - 0.30)}{g}}, \tag{3.1}$$

where h_{max} is the torso height at top-of-flight and g is the gravitational acceleration. This equation is based on the simple physics of a body falling under gravity, modified to account for an approximate leg length at ground contact (0.30 m) and reduced (by 25 percent) to ensure completion of the **position** primitive at the approximate time of ground contact. The control parameters required are the desired hip and knee joint angles, $\theta_{h,d}$ and $\theta_{k,d}$, respectively. The desired joint angles are transformed to the corresponding desired motor angle θ_d as

$$\theta_d = \begin{cases} \theta_{h,d}\, n_m & \text{for the hip joint} \\ (\theta_{h,d} + \theta_{k,d}) n_m & \text{for the knee joint.} \end{cases} \tag{3.2}$$

The **position** control law calculates a trajectory $\theta_{m,pos}$, as well as the associated velocity $\dot{\theta}_{m,pos}$, for the motor corresponding to the controlled joint. The trajectory is defined by equal periods of constant acceleration and deceleration to reduce the magnitudes of both values required to perform a given trajectory. The trajectory is timed to be completed in the period $\Delta t_{p,1}$, using additional inputs of the primitive start time t_0, the current time t, and the motor position at the start of the primitive θ_{m,t_0}.

$$\theta_{m,pos} = \begin{cases} \theta_{m,t_0} + \dfrac{a}{2}\,(t_{fun})^2 & \text{for } 0 \le t_{fun} < \Delta t_b \\ \theta_d - \dfrac{a}{2}(\Delta t_{p,1} - t_{fun})^2 & \text{for } \Delta t_b \le t_{fun} < \Delta t_{p,1} \end{cases} \tag{3.3}$$

$$\dot{\theta}_{m,pos} = \begin{cases} a\, t_{fun} & \text{for } 0 \le t_{fun} < \Delta t_b \\ 2\, a\, t_b - a\, t_{fun} & \text{for } \Delta t_b \le t_{fun} \le \Delta t_{p,1}, \end{cases} \tag{3.4}$$

where the function time t_{fun}, blend time Δt_b, and acceleration a are defined as

$$t_{fun} = t - t_0 \tag{3.5}$$

$$t_b = \frac{\Delta t_{p,1}}{2} \tag{3.6}$$

$$a = \frac{4(\theta_d - \theta_{m,t_0})}{(\Delta t_{p,1})^2}. \tag{3.7}$$

The calculated motor trajectory is achieved through proportional control on the motor position and velocity. The control law determines the command current for the ith motor as

$$i_{c,\mathrm{i}} = k_{pos}(\theta_{m,pos} - \theta_{m,\mathrm{i}}) + k_{vel}(\dot{\theta}_{m,pos} - \dot{\theta}_{m,\mathrm{i}}), \tag{3.8}$$

where k_{pos} and k_{vel} are the experimentally tuned proportional position and velocity feedback gains.

3.2.4 Motor Primitive: Catch

The **catch** motor primitive, following **position**, provides control for the initial portion of ground contact. The associated control law is simple; open loop motor currents represented by the control parameters $i_{h,cat}$ and $i_{k,cat}$ are commanded for both the hip and knee motors. The command currents are

$$i_{c,\mathrm{i}} = \begin{cases} i_{h,cat} & \text{for i=1,3} \\ i_{k,cat} & \text{for i=2,4.} \end{cases} \tag{3.9}$$

The open loop currents are used to slow the vertical velocity of the torso and store the kinetic energy of the falling body as potential energy in the actuator spring elements or to dissipate the energy through the resistance of the motors and energy loss of the impact. The primitive begins upon expiration of $\Delta t_{p,1}$ during the previous primitive and is terminated after ground contact of the foot and upon the knee joint angular velocity of either knee dropping below an experimentally tuned zero threshold $\dot{\theta}_{lim}$. The zero threshold is expressed as

$$\dot{\theta}_{\mathrm{i}} \leq \dot{\theta}_{lim} \text{ for i=2,4.} \tag{3.10}$$

Fod et al. [42] suggest the use of similar thresholds for segmentation of motions into movement primitives. Changes in joint direction or dwells at a constant position generally indicate a change in the immediate low-level objective. This termination criteria is used to detect the approximate time at which the knee actuator ceases to store the kinetic energy and begins delivering the stored energy via thrust.

3.2.5 Motor Primitive: Thrust

The **thrust** motor primitive encapsulates the phase during which the actuators inject energy into the jumping maneuver. A similar control law to that used in the **catch** primitive is implemented for **thrust**. The open loop command currents for the hip and knee motors are represented by the control parameters $i_{h,thr}$ and $i_{k,thr}$. The control law of Eq. 3.9 is then changed to

$$i_{c,\mathrm{i}} = \begin{cases} i_{h,thr} & \text{for i=1,3} \\ i_{k,thr} & \text{for i=2,4.} \end{cases} \tag{3.11}$$

The primitive begins upon completion of **catch** and is terminated upon either knee joint angle becoming less than an experimentally tuned position threshold θ_{lim}.

$$\theta_{\mathrm{i}} < \theta_{lim} \text{ for i=2,4.} \tag{3.12}$$

This termination criteria is formulated to achieve two specific objectives. The first objective is reduction of the impact of the joint limits upon foot liftoff. The use of foot liftoff itself as the termination criteria would lead to the rapid expulsion of any remaining potential energy stored in the spring elements. This would cause actuated links to be forced into the joint limits at high velocity, producing large impact forces. Large impact forces are undesirable both for hardware longevity and system stability. The preemptive termination of the **thrust** allows time to implement the **leg slow** primitive, which is intended to counteract this effect. The second objective is the reduction of torque developed between the shank and the torso as the leg approaches a singular configuration. As singularity is approached, the torque produced by the knee actuator contributes less to the vertical velocity of the system (as desired for height performance) and more to the rotational velocity of the torso. The high sensitivity of the torso to torque developed in the knee joints will be discussed in Chapter 4 as an undesirable effect of the articulated joint design of the legs.

3.2.6 Motor Primitive: Leg Slow

The purpose of the **leg slow** primitive, as discussed above, is to prevent the undesirable occurrence of the links being driven into the joint limits. The termination criteria for **thrust**, the knee joint angle becoming less than the threshold value, is used to detect the start of the **leg slow** primitive. The control law uses open loop command currents for the motors in a manner similar to the previous two control laws. The command current for each motor is set to the maximum value i_{max} in the direction opposite of thrust until the velocity of the joint changes sign. The commanded motor current is then set to zero until termination of the primitive. The command currents are

$$i_{c,\mathrm{i}} = \begin{cases} -i_{max} & \text{for i=1,3 and } \dot{\theta}_\mathrm{i} > 0 \\ 0 & \text{for i=1,3 and } \dot{\theta}_\mathrm{i} \le 0 \\ i_{max} & \text{for i=2,4 and } \dot{\theta}_\mathrm{i} < 0 \\ 0 & \text{for i=2,4 and } \dot{\theta}_\mathrm{i} \ge 0. \end{cases} \tag{3.13}$$

The termination criteria used is the detection of a sign change in the velocity of both the hip and knee joints for both legs, the occurrence of which is marked by the binary flag f_{ls}, defined as

$$f_{ls} = \begin{cases} 1 & \text{if } (\dot{\theta}_1 < 0) \text{ and } (\dot{\theta}_2 > 0) \text{ and } (\dot{\theta}_3 < 0) \text{ and } (\dot{\theta}_4 > 0) \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

The inclusion of both legs in the termination criteria is unique to this primitive and is used to synchronize the leg motion for the start of the following primitive. Separate termination criteria for the two legs were observed to elicit behaviors involving foot slip across the ground.

3.2.7 Motor Primitive: Retract

The termination of the **leg slow** primitive signifies the start of the **retract** primitive, noted as start time t_0. The **retract** primitive is similar in structure and objective to **position**. The hip and knee motors are driven from their positions θ_{m,t_0} at the completion of **leg slow** to angular positions corresponding to the desired top-of-flight joint angles $\theta_{h,tof}$ and $\theta_{k,tof}$. The primitive is timed to be completed in the primitive period $\Delta t_{p,5}$. The primitive period uses an adjusted model of the physics, similar to Eq. 3.1, using the desired jump height h_d in place of h_{max}. The primitive period is

$$\Delta t_{p,5} = 0.55\sqrt{\frac{2(h_d - 0.30)}{g}}. \tag{3.15}$$

The leading constant (0.55) for this model was experimentally tuned to ensure complete retraction of the legs prior to achieving top-of-flight. The desired motor positions described in Eq. 3.2 can be modified suitably to

$$\theta_d = \begin{cases} \theta_{h,tof}\, n_m & \text{for the hip joint} \\ (\theta_{h,tof} + \theta_{k,tof})n_m & \text{for the knee joint.} \end{cases} \tag{3.16}$$

The motors are driven through a triangular velocity trajectory similar to that described by Eq. 3.3 with a modification to address difficulties introduced by the elastic elements of the actuators. During the **retract** primitive, the links are moved opposite the direction used for **thrust**. As a result of this, the actuators remain in the DDA regime during the initial part of the trajectory, during which time the joints closely track the desired trajectories. However, during the deceleration phase of the trajectories, the actuators are effectively in the SEA regime, as they are during the **catch** and **thrust** primitives. The momentum of the links cause the elastic elements to be engaged, and the links no longer track the desired trajectories as closely. Any potential energy stored in the springs as angular deflection will be delivered to the joints upon the motors reaching the desired positions, leading to overshoot. To decrease the effect of actuator spring deflection and overshoot, the magnitude of the deceleration can be reduced. The trajectory specified for the motors in Eq. 3.3 is changed so that the deceleration portion is completed over twice the time period of the initial acceleration, reducing the deceleration required. The revised trajectory is defined as

$$\theta_{m,pos} = \begin{cases} \theta_{m,t_0} + \frac{a}{2}\left(t_{fun}\right)^2 & \text{for } 0 \leq t_{fun} < \Delta t_b \\ \theta_d - \frac{a}{4}(\Delta t_{p,5} - t_{fun})^2 & \text{for } \Delta t_b \leq t_{fun} < \Delta t_{p,5} \end{cases} \tag{3.17}$$

$$\dot{\theta}_{m,pos} = \begin{cases} a\, t_{fun} & \text{for } 0 \leq t_{fun} < \Delta t_b \\ a\, t_b - \frac{a}{2}(t_{fun} - t_b) & \text{for } \Delta t_b \leq t_{fun} \leq \Delta t_{p,5}, \end{cases} \tag{3.18}$$

where the function time t_{fun}, blend time Δt_b, and acceleration a are

$$t_{fun} = t - t_0 \tag{3.19}$$

$$t_b = \frac{\Delta t_{p,5}}{3} \tag{3.20}$$

$$a = \frac{6(\theta_d - \theta_{m,t_0})}{(\Delta t_{p,5})^2}. \tag{3.21}$$

The trajectory is achieved through feedback control of the motor position and velocity using the same control law as for **position**, Eq. 3.8, with the same experimentally tuned gains.

3.2.8 Motor Primitive: Hold

The expiration of the previous primitive is used to mark the transition from **retract** to **hold**. The **hold** primitive is used to maintain the desired top-of-flight joint angles $\theta_{h,tof}$ and $\theta_{k,tof}$ during the remainder of the ascending phase of the jump. The termination criteria used is the detection of the top-of-flight. Top-of-flight is sensed through a change in sign of $\dot{\theta}_6$ between two successive control steps and denoted with the binary flag f_{tof}. The binary flag is set to

$$f_{tof} = \begin{cases} 1 & \text{if } \dot{\theta}_6 < 0 \text{ and } \dot{\theta}_{6,p} \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{3.22}$$

where $\dot{\theta}_{6,p}$ is the value of $\dot{\theta}_6$ at the previous control step. When f_{tof} changes value, the state machine is reset, and the **position** is implemented. The control law used is simple proportional-derivative (PD) feedback control performed on the motor position. Integral feedback control, implemented in many applications to eliminate steady-state error (SSE), is not required in this work. The short time periods spent at a given position, in combination with finite precision limits due to mechanical backlash, make the elimination of SSE unnecessary. The PD control law determines the command current for the ith actuator as

$$i_{c,\mathrm{i}} = k_p(\theta_{m,tof} - \theta_{m,\mathrm{i}}) + k_d \frac{d}{dt}(\theta_{m,tof} - \theta_{m,\mathrm{i}}), \tag{3.23}$$

where the desired top-of-flight motor position $\theta_{m,tof}$ is defined as

$$\theta_{m,tof} = \begin{cases} \theta_{h,tof}\, n_m & \text{for the hip joint} \\ (\theta_{h,tof} + \theta_{k,tof}) n_m & \text{for the knee joint} \end{cases} \tag{3.24}$$

and the rate of change of the difference in desired and actual motor position $\frac{d}{dt}(\theta_{m,tof} - \theta_{m,\mathrm{i}})$ is calculated as

$$\frac{d}{dt}(\theta_{m,tof} - \theta_{m,\mathrm{i}}) = \frac{(\theta_{m,tof} - \theta_{m,\mathrm{i}}) - (\theta_{m,tof} - \theta_{mp,\mathrm{i}})}{\Delta t}. \tag{3.25}$$

The control law requires the motor position at the previous control step $\theta_{mp,\mathrm{i}}$ and the time elapsed since the last control step Δt.

3.2.9 Revisions to State Machine

The final structure of the state machine and related motor primitives, listed in Table 3.1 with the defined start and termination criteria, is the result of several revisions to the original structure. Trial and error was necessary to determine an appropriate set of motor primitives for performing the desired jumping maneuvers. Several dynamic characteristics of the system unique to the unidirectional series elastic actuators necessitated creation, elimination, or modification of certain primitives.

- The **leg slow** primitive was introduced to reduce the impact of the links with the joint limits, a problem that is amplified by the release of the remaining energy stored in the elastic elements at foot lift-off (making the problem more severe than for a direct drive actuator). It was also introduced for the beneficial

Table 3.1: State Machine and Motor Primitives

Motor Primitive Index	Motor Primitive Name	Starting Event	Ending Event
1	Position	$f_{tof} = 1$	$\Delta t_{p,1}$ Expired
2	Catch	$\Delta t_{p,1}$ Expired	$\dot{\theta}_{\mathrm{i}} < \dot{\theta}_{lim}$
3	Thrust	$\dot{\theta}_{\mathrm{i}} < \dot{\theta}_{lim}$	$\theta_{\mathrm{i}} < \theta_{lim}$
4	Leg Slow	$\theta_{\mathrm{i}} < \theta_{lim}$	$f_{ls} = 1$
5	Retract	$f_{ls} = 1$	$\Delta t_{p,5}$ Expired
6	Hold	$\Delta t_{p,5}$ Expired	$f_{tof} = 1$

effect of stabilizing the joint positions prior to entering the trajectory defined in the **retract** primitive.

- An additional **hold** primitive (similar to the one used just prior to the top-of-flight) was originally implemented upon completion of the **position** primitive and before the **catch** primitive. It was intended to maintain the desired joint positions until foot contact was sensed. The primitive was eliminated to allow for an increased primitive time period for **position**, reducing the accelerations required for the desired joint trajectories. High accelerations were observed to cause the actuator to leave the direct drive and preload regimes, introducing difficulty for the control.

- The primitive time periods calculated from approximate falling time, Eq. 3.1 and 3.15, were modified through changes to the leading constant to ensure that the corresponding primitives terminate at appropriate points during the jumping maneuver while maintaining as long of time periods as possible for performing joint motions.

- The joint trajectories described for **position** and **retract** originally used trapezoidal velocity profiles with three segments of equal time for acceleration, constant velocity, and deceleration. The change to triangular velocity profiles was made in both primitives to reduce the required accelerations for achieving the desired joint positions. The reduced acceleration and deceleration were found to produce trajectories that are easier to track with less disturbance introduced to the overall system dynamics.

3.3 Supervisory Controller: Fuzzy Control

The supervisory controller serves to define the value of the control parameters required by the motor primitives described above for completing the desired maneuver. The high-level control serves no function in the actual implementation of the parameters. The supervisory controller consists of fuzzy control performed once per jump, at the top-of-flight.

3.3.1 Introduction to Fuzzy Control

The method of intelligent control known as fuzzy control offers a strategy for creating a mapping between control inputs, typically state variables, and control actions for complex, nonlinear systems. The elements of fuzzy control will be discussed in detail below; however, a general understanding of the basic principle will inform this discussion. In general terms, a fuzzy controller contains a data set consisting of control actions appropriate for defined instances of the control inputs in various combinations. Given a number of control inputs, the fuzzy controller consults its data set to determine which stored instances most closely represent the given input. The

Table 3.2: Fuzzy Control Notation

Symbol	Variable Name
n	Number of Control Inputs
m	Number of Control Outputs
n_{mf}	Number of Input Membership Functions
i	Input Index, $1 \leq \mathrm{i} \leq n$
j	Output Index, $1 \leq \mathrm{j} \leq m$
k	Membership Function Index $1 \leq \mathrm{k} \leq n_{mf}$
z	Fuzzy Rule Number
x_{i}	ith Control Input
y_{j}	jth Control Output
$u_{z,\mathrm{j}}$	jth Fuzzy Consequent
$c_{\mathrm{i,k}}$	kth Membership Function Center of ith Input
$\mu_{\mathrm{i,k}}$	Certainty of kth Membership Function for ith Input
$\mu_{z,prem}$	Certainty of Premise for zth Rule

fuzzy controller then produces control outputs that represent the stored control actions for the applicable input instances, with the output contribution of each instance weighted by a metric of how relevant the stored instance is to the given input.

The notation used for the following discussion of fuzzy control is shown in Table 3.2. More precise explanations of the variables are provided as they become relevant.

Fuzzy control can be divided into three primary processes referred to as **fuzzification**, **inference**, and **defuzzification**. A flow chart for the control processes can be seen in Fig. 3.2. Each of these three processes is discussed below, in order.

3.3.2 Fuzzy Control: Fuzzification

The first process of fuzzy control, **fuzzification**, serves to interpret the numeric values of the control inputs $x_1 : x_n$. Every control input x_{i} has a total number of n_{mf} membership functions associated with it. The membership functions divide

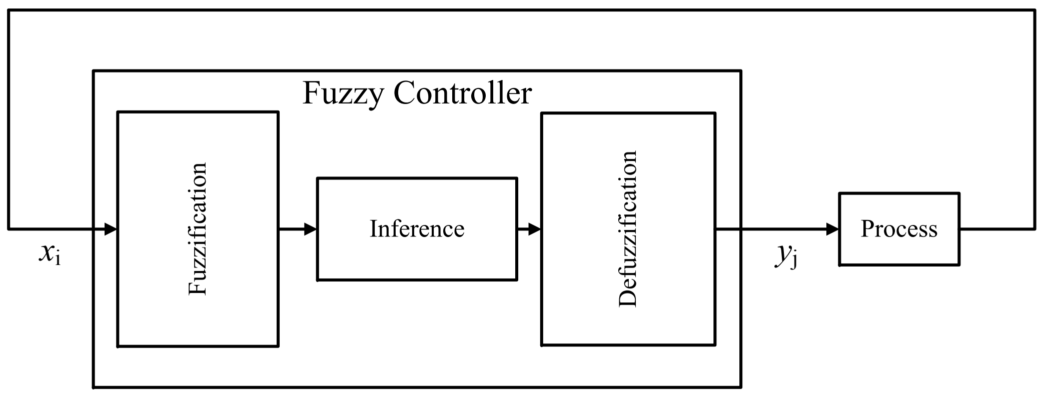

Figure 3.2: Fuzzy control flow chart. In this work, the 'Process' is a complete jumping cycle of the biped.

the control input range into n_{mf} segments centered around the membership function centers $c_{i,k}$. It should be noted that in traditional fuzzy control, as described by Passino [17], the membership function centers are associated with linguistic values, whereas strictly numeric values are used here. This work uses triangular membership functions, depicted in Fig. 3.3. A membership function for the ith control input, identified by the membership function index k, describes the certainty $\mu_{i,k}$ that the numeric input value can be represented by the membership function center $c_{i,k}$. The certainty $\mu_{i,k}$ ranges in value from 0 to 1 as a function of the distance of the numeric input from the relevant membership function center. As an example, for a given numeric input x_i where $x_i = c_{i,k}$, the certainty will be calculated as $\mu_{i,k} = 1$. This example represents the case where the numeric input value lies directly at the center of a membership function, and so the certainty that the membership function center represents the input value is maximum. As the numeric value moves away from a membership function center, the certainty decreases. An important exception to this

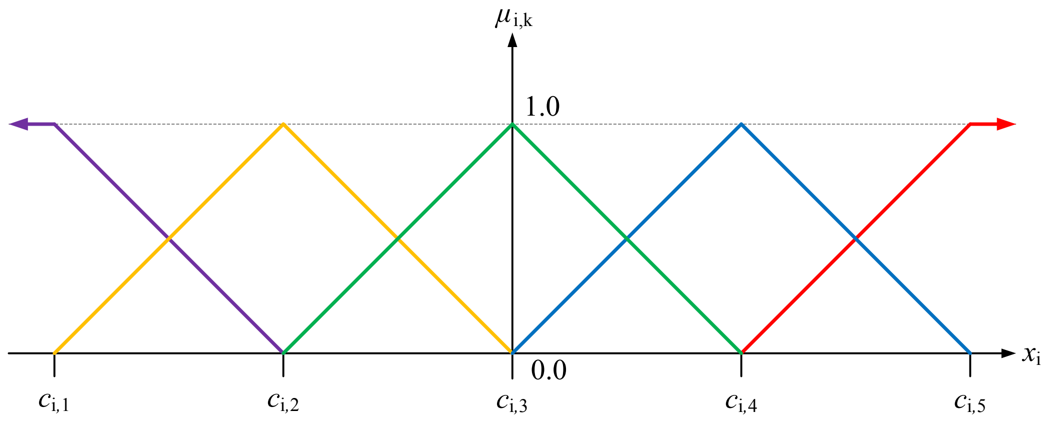

Figure 3.3: Triangular fuzzy membership functions with end saturation

is the end membership functions. These membership functions include saturation, such that input values outside of the range of the membership function centers, i.e. $x_\text{i} < c_{\text{i},1}$ or $x_\text{i} > c_{\text{i},n_{mf}}$, are still represented by the numerically closest membership function center.

The use of triangular membership functions that terminate at the adjacent membership function centers, as shown in Fig. 3.3, ensures that a maximum of two membership functions will have a non-zero certainty. This information is useful for reducing the calculations required by the fuzzy control process.

Upon completion of calculating the certainty of the inputs based on membership functions, the fuzzy controller moves to the **inference** process.

3.3.3 Fuzzy Control: Inference

The **inference** process of fuzzy control functions to match appropriate control outputs to the given inputs; this requires the use of fuzzy rules. Fuzzy rules are used to specify control outputs appropriate to particular values of the inputs. A fuzzy rule is created for every combination of the control input membership function centers previously defined. Assuming an equal number n_{mf} of membership functions for each of the n control inputs, the fuzzy controller contains $(n_{mf})^n$ total rules. These rules are collectively referred to as the fuzzy rule-base and are numbered from 1 to $(n_{mf})^n$, represented by the index z. Associated with each of the fuzzy rules are appropriate control outputs, called consequents, represented by $u_{z,\mathrm{j}}$ where j continues to represent the control output index.

Linguistically, a fuzzy rule can be expressed as a conditional statement.

$$\begin{aligned} &\textit{If } x_1 = c_{1,k} \textit{ and } x_2 = c_{2,k} \textit{ and} \ldots \textit{ and } x_n = c_{n,k} \\ &\textit{then } y_1 = u_{z,1} \textit{ and } y_2 = u_{z,2} \textit{ and } \ldots \textit{ and } y_m = u_{z,m}. \end{aligned} \tag{3.26}$$

The rule consists of two elements. The conditional statement is called the premise of the rule, which states when the rule applies. The specified control outputs, previously defined as the consequents, collectively comprise the consequence of the rule.

The membership function certainty for each of the inputs is used to determine the applicability of a rule to the given inputs. This applicability is defined as the certainty of the premise $\mu_{z,prem}$ for the zth rule. This certainty reflects the certainty that each matches the value stated in the premise. The certainty of the premise can be calculated in many ways as a function of certainty of the membership functions. In this work, the product of the certainties of the input membership functions is used as the certainty of the premise.

$$\mu_{z,prem} = \mu_{1,k} \cdot \mu_{2,k} \cdot \cdots \cdot \mu_{n-1,k} \cdot \mu_{n,k}, \tag{3.27}$$

where the membership function index k for each $\mu_{i,k}$ matches the index of the ith input for rule number z. This definition of $\mu_{z,prem}$ maintains the influence of all input values on the resulting outputs.

As mentioned above, the use of overlapping triangular membership functions assures that for each input a maximum of two membership functions will have a non-zero certainty. This property dictates that a maximum of 2^n rules will be applicable for any combination of inputs. The reduction of the number of rules that are potentially applicable minimizes the required computations, which due to the exponential growth of the rule-base, can offer substantial savings in terms of computational efficiency.

3.3.4 Fuzzy Control: Defuzzification

The final stage of fuzzy control, the **defuzzification** process, is responsible for determining specific values for each of the control outputs. This work uses singleton membership functions for the control outputs. Singleton membership functions can be represented by impulse functions at every value on the range. Effectively, this means that the control outputs described by the fuzzy rule consequents are not modified from their numerical values. The determination of the specific values to be output is based upon the applicable rules and the respective certainty of the premise for each. A center-average defuzzification scheme is used in this work. This scheme defines each control output as a weighted average of the consequents for the active rules. The jth control output is calculated as

$$y_{\mathrm{j}} = \frac{\displaystyle\sum_{z \in S} \mu_{z,prem} u_{z,\mathrm{j}}}{\displaystyle\sum_{z \in S} \mu_{z,prem}}, \tag{3.28}$$

where S is the set of active rule numbers. The formula is applied for every control output y_{j}, for $1 \leq \mathrm{j} \leq m$.

The control output calculation of Eq. 3.28 does not account for the possibility of fuzzy rules for which no appropriate control parameters are specified by the rule-base. Allowing the inclusion of inappropriate control parameters in the fuzzy control process can cause otherwise acceptable jumps to be performed incorrectly. This possibility is counteracted through the inclusion of an additional variable in the fuzzy controller. The training satisfaction flag $f_{sat,z}$ represents whether an appropriate combination of control parameters are known for the z^{th} rule. The flag is set equal to 1.0 if appropriate control parameters were found by the training algorithm and 0.01 if they were not. The ratio is set to 0.01 in place of 0.00 for unsatisfactory rules to ensure a set of non-zero control parameters will be provided by the fuzzy controller regardless of the ability to perform the desired jump. The defuzzification calculation of Eq. 3.28 is then modified to include this satisfaction ratio. The fuzzy control output becomes

$$y_{\mathrm{j}} = \frac{\displaystyle\sum_{z \in S} (\mu_{z,prem} u_{z,\mathrm{j}}) f_{sat,z}}{\displaystyle\sum_{z \in S} (\mu_{z,prem}) f_{sat,z}}. \tag{3.29}$$

The use of the training satisfaction flag reduces the effect of control parameters that will potentially disturb the system.

3.3.5 Motivation for Fuzzy Control

The division of the control architecture into two levels, low and supervisory, improves the tractability of the control problem. However, the complexity of the system still presents challenges to the supervisory controller. The selection of fuzzy control as the supervisory controller was made to address several of the main challenges associated with the system, including the multi-body dynamics, the nonlinear nature of the system, the multiple inputs and outputs, and the desired flexibility in the implementation of the controller.

The multi-body dynamics of the system pose a potentially large challenge for the controller. Many control approaches require accurate models of the system dynamics. Accurate modeling of a multi-body system is analytically intensive and is highly reliant on the knowledge of the physical model parameters. Deviation between the model and the physical hardware can produce non-functioning controllers. Additionally, changes to the physical system or application of the control approach to a new physical system requires that the development of the model be repeated. Fuzzy controllers, in contrast to this, do not require an accurate system model. The fuzzy controller works around the necessity of a dynamic model by associating control outputs directly with the relevant control inputs. This association exists within the rule-base. The relationships between the control inputs and outputs stored within the rule-base also solve the challenge of the nonlinearity of the system. The discrete points in the rule-base described by each individual rule form an approximation of the nonlinear, multi-dimensional surface that relates the inputs and outputs.

The multiple control inputs and outputs present an additional challenge. The nature of the physical system produces coupling between the various control parameters and state variables. Similar to the dynamic model, the fuzzy rule-base is capable of storing and managing an inherent understanding of this phenomenon without requiring an explicit description.

The emphasis of this work is the control of jumping in a biped robot. The control approach, however, is desired to be applicable to a variety of systems and maneuvers. In particular, it is desired that the control approach could be adapted to bipedal maneuvers including running, starting, and stopping. The flexibility of fuzzy control, when used in conjunction with modular motor primitives, allows the adaptation required for new maneuvers or systems.

3.3.6 Variation from Traditional Fuzzy

The implementation of fuzzy control used in this work varies somewhat from traditional fuzzy control methods. The principal difference is the way in which heuristic information provided by the user is applied to the development of the controller. More traditional approaches to fuzzy control implement heuristic information about how to control the system through the development of the rule-base. Typically, consequents for each premise are selected by the user based on past experience with determining appropriate control actions. This work implements heuristic information in the process of training the fuzzy rule-base, where control outputs are varied to find appropriate values. More detailed discussion of the use of heuristic information is found in Chapter 4.

3.3.7 Structure of Fuzzy

The rule-base of the fuzzy controller can be visualized as an n-dimensional space, where n is the previously defined number of control inputs. The range of values in the space is defined for each dimension as the minimum and maximum membership function center of the associated input. The space contains discrete points at the intersection of the input membership function centers; these points represent each

Table 3.3: Control Input Membership Function Centers

Symbol	Input	Membership Function Centers
x_1	Torso Angle at TOF, rad	-0.340 , -0.320 , -0.30 , -0.280 , -0.260
x_2	Torso Angular Rate at TOF, $\frac{\text{rad}}{\text{s}}$	0.50 , 0.55 , 0.60 , 0.65 , 0.70
x_3	Current Jump Height, m	0.550 , 0.565 , 0.575 , 0.585 , 0.600
x_4	Forward Angular Rate at TOF, $\frac{\text{rad}}{\text{s}}$	-0.03 , 0.015 , 0.00, 0.015 , 0.03

individual fuzzy rule. The fuzzy controller in this work includes four control inputs - torso angle at top-of-flight $\theta_{5,tof}$, torso angular rate at top-of-flight $\dot{\theta}_{5,tof}$, current jump height h_{max}, and forward angular rate at top-of-flight $\dot{\theta}_{7,tof}$ - represented by x_1, x_2, x_3, and x_4, respectively. Each of the four control inputs has five triangular membership functions, with centers as described in Table 3.3. The total number of rules is thus $5^4 = 625$.

3.3.8 Revisions to Fuzzy Structure

The structure of the fuzzy controller described above is the result of several revisions to the original structure. The control inputs describing the torso state at the top-of-flight (angular position and rate) as well as forward angular rate were not included in the original structure. The values selected for the membership function centers were also modified throughout the development of the control strategy. The method of determining the certainty of the premise of a fuzzy rule proved to be a key aspect of the controller and was changed from the initial strategy to the final result.

The inclusion of the torso state variables (θ_5 and $\dot{\theta}_5$) in the fuzzy controller was necessitated by two problems: the difficulty of training control parameters to return the torso to a specific state and the sensitivity of the system to perturbations in the torso state. The original design of the fuzzy controller assumed that it would be possible to select control parameters for any desired jump within the performance limitations that would return the state of the robot (including the torso) to precisely the same state from which it started the jump. Simulations intended to find these control parameters revealed difficulty in returning the torso state to the desired angular position and rotational speed. It was either not possible to achieve this goal with the limitations introduced through the motor primitive control architecture or computationally infeasible to find such precise control parameters. In simulation, it was observed that sequential jumps intended to maintain a specific height could not be performed with a constant set of control parameters, due to variation in the top-of-flight torso state causing the system to become unstable. This observation, that small perturbations in the torso state could produce instability, suggested that the torso state was a necessary input to the fuzzy controller for determining appropriate control parameters.

The addition of the forward angular rate at top-of-flight followed a similar reasoning to that of the torso state. It was observed during implementation of previous fuzzy controllers that small perturbations to the forward angular rate of the system could cause otherwise acceptable jumps to become unstable. Inclusion of the forward angular rate in the fuzzy controller also allowed for a more direct definition of the allowable forward motion for a jump in the training algorithm (discussed in Chapter 4).

Selection of the membership function centers for the fuzzy controller required iteration to achieve satisfactory performance. The space described by the fuzzy controller inputs represents the working range of the controller and offers conflicting goals related to the size of the space as described by the membership function centers. A large working range is desirable for the controller to allow for more varied physical performance of the robot and to minimize the difficulty in selecting control parameters that will return the state of the robot to the working range of the controller. Resolution issues present a direct conflict with the desire for a large working range. The interpretation of the rule-base as an approximation of the nonlinear mapping between the fuzzy inputs and outputs indicates that the accuracy of the approximation is dependent upon maintaining a fine enough resolution among the inputs to capture the important features of the nonlinear relationships. The final values selected for the membership function centers were determined through trial and error to be a compromise between these two conflicting goals.

The calculation of the certainty of the premise for a fuzzy rule of Eq. 3.27 initially used a minimum function, of the form

$$\mu_{z,prem} = min\{\mu_{1,k}, \mu_{2,k}, \ldots, \mu_{n-1,k}, \mu_{n,k}\}. \tag{3.30}$$

This method of calculating the certainty of the premise was based on a simple idea: the controller can be no more certain of a fuzzy rule than it is of any of the elements of the premise of that rule. While this method of determining the relevance of fuzzy rules can be successfully implemented for some control systems, it introduces problems when applied to the fuzzy controller described in this work. The ability of the fuzzy controller to select control parameters for returning the system to the desired top-of-flight state is dependent on appropriately combining the relevant stored values in the fuzzy rule-base. The difference in the minimum and product functions can be explained by the nature of the resulting fuzzy outputs when the controller receives inputs between membership function centers. A generic fuzzy controller with two inputs (n=2) and a single output (m=1) can be used to demonstrate this. The fuzzy output can be interpreted as a surface that is a function of the two control inputs across a range of values between the membership function centers. The surface produced using the certainty calculation of Eq. 3.30 has a key difference from that produced when using Eq. 3.27. The surface produced by the minimum function will show zero-order continuity only, while the surface produced by the product function will show zero- and first-order continuity. This difference between the two surfaces represents the problem introduced by using the minimum function. The minimum function is unable to capture the relevance of multiple fuzzy inputs in a manner that produces smoothly continuous fuzzy outputs with varied input values. When used in the fuzzy controller described in this work, this problem introduced by the minimum function causes the biped system to become unstable, while the product function maintains stability.

3.3.9 Control Parameters

The control parameters required by the motor primitive control laws are stored as the consequents of the fuzzy rules. The motor primitives used for low-level control dictate the number of control parameters that must be specified for performing the desired maneuver. In this work, the required control parameters are: desired hip and knee angles, $\theta_{h,d}$ and $\theta_{k,d}$, hip and knee motor open loop catch currents, $i_{h,cat}$ and $i_{k,cat}$, and the hip and knee motor open loop thrust currents, $i_{h,thr}$ and $i_{k,thr}$; these are represented by the fuzzy control outputs y_1, y_2, y_3, y_4, y_5, and y_6.

3.4 Summary

This chapter describes the control approach used to achieve the desired maneuver. The low-level control laws, motivated by the concept of motor primitives, are developed in detail and presented as phases of a state machine. The supervisory controller is described and contrasted with traditional fuzzy controllers. Emphasis is placed on the revisions to the controller, both high- and low-level, required to produce stable jumping maneuvers.

CHAPTER 4

FUZZY TRAINING

4.1 Introduction

The control strategy described in the previous chapter lays out the framework required for achieving jumping. However, the method for determining the control outputs for every rule in the rule-base remains to be explained. The task of determining appropriate control outputs for every scenario described in the rule-base presents a significant challenge. The size of the rule-base, 5^4=625 rules, combined with the number of required control outputs (six) produces a total of 3750 control outputs that must be specified for the rule-base to be complete. This challenge is addressed with an algorithm used to train the fuzzy controller in the numerical simulation through iterative jumps with error-based feedback. This chapter describes the fuzzy training algorithm, specific difficulties encountered, and results of the completed training. Heuristic knowledge derived from the training results and control strategy development are then presented. The chapter concludes with interpretations of the fuzzy training algorithm and a summary.

4.2 Training Satisfaction Criteria

The purpose of training the fuzzy controller is to find values of the control outputs y_1:y_6 that will achieve the desired performance of the jumping maneuver for every rule z. In order to determine when a rule is considered to be trained, criteria that the jump must satisfy are needed. These are referred to as the training satisfaction criteria. The three criteria used in this work emphasize the state of the system at the top-of-flight. The ability of the controller to produce stable, successive jumps requires that any control action specified by parameters in the rule-base results in the system state at the end of a jump falling within the range of the fuzzy controller with respect to all inputs. This will ensure that appropriate control actions will be available for the following jump. This can also be explained as a requirement that the controller must map the system state back to within its own working range to provide stability and robustness.

The most directly formulated satisfaction criteria are based on the gross characteristics of the jump. The height and forward angular rate of the system at the top-of-flight are evaluated based upon the desired behavior of the robot. The jump height h is desired to achieve a specific value, while the forward angular rate $\dot{\theta}_{7,tof}$ is desired to be small enough that the system maintains motion primarily in the vertical direction. These two variables are assigned an acceptable amount of deviation, δ_1 and δ_2 for the height and forward motion, from central desired values h_d and $\dot{\theta}_{7,d}$. The allowable deviation, or control error tolerance, and central desired value for the two control errors are based upon the values of the membership function centers for the fuzzy control inputs associated with the criteria: current jump height and forward angular rate at top-of-flight (x_3 and x_4, respectively). The control error tolerance is selected for each error to ensure that the training algorithm will produce jumps with both a height and forward angular rate that fall within the working range of the fuzzy controller.

The control errors corresponding to jump height and forward angular rate at top-of-flight are ε_1 and ε_2, calculated as

$$\varepsilon_1 = h - h_d \tag{4.1}$$

and

$$\varepsilon_2 = \dot{\theta}_{7,d} - \dot{\theta}_{7,tof}. \tag{4.2}$$

The values of the desired height and forward angular rate are set equal to the middle membership function center of their associated fuzzy control input.

$$h_d = c_{3,3} = 0.575\ m \tag{4.3}$$

and

$$\dot{\theta}_{7,d} = c_{4,3} = 0.00\ \frac{rad}{s}. \tag{4.4}$$

These criteria are formulated such that a negative control error for height represents a jump that is too short, while a negative control error for the forward angular rate represents a jump in the backward direction (based on the previously defined sign conventions and state variables). The training satisfaction criteria can then be expressed by the inequalities

$$-\delta_1 \leq \varepsilon_1 \leq \delta_1 \tag{4.5}$$

and

$$-\delta_2 \leq \varepsilon_2 \leq \delta_2. \tag{4.6}$$

The acceptable range of values in these two criteria cover only part of the range of the fuzzy controller. This formulation of the satisfaction criteria is designed to cause the training algorithm to find control parameters that will return the system to the central region of the fuzzy controller's range. When the trained rule-base is implemented with the fuzzy controller, this encourages stability by causing the state of the system to be returned from points near the edge of the rule-base back to the central region.

The achievement of an acceptable top-of-flight torso state is also required for control stability and robustness. Consider a planar subspace of the complete rule-base, shown in Fig. 4.1, that is formed by holding the current jump height and forward angular rate fixed. The two dimensions of the plane represent the pitch and pitch rate of the torso, where the range of each is specified by the membership function centers. The torso state at top-of-flight must fall within the shaded area of this plane to allow the fuzzy controller to select appropriate control parameters for the following jump. The tight coupling between the pitch and pitch rate of the torso does not allow the use of individual control errors for each, but necessitates the use of a single control error metric that incorporates both parameters. This is accomplished through use of a modified version of the planar subspace, as seen in Fig. 4.2. The torso state at the end of the training jump is located on the (x_1,x_2) plane at a point p, with coordinates defined by the variable θ_5 and its time derivative $\dot{\theta}_5$. A new coordinate system (X_1,X_2) is added to the plane with the origin at the center (defined by the membership function centers $c_{1,3}$ and $c_{2,3}$) and a coordinate transformation is used to

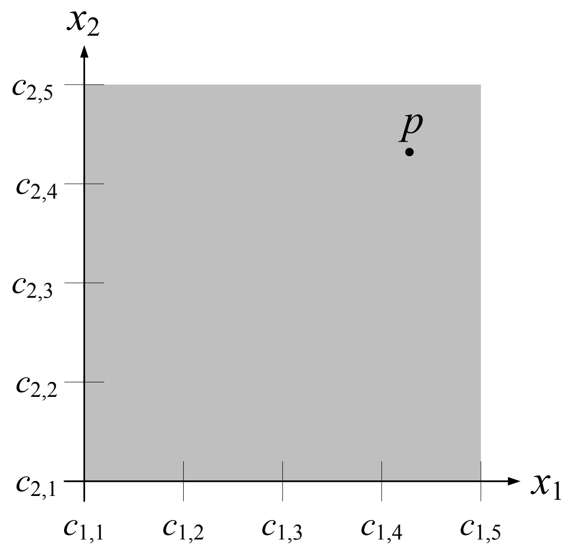

Figure 4.1: Planar subspace of the fuzzy rule-base. Point p is an arbitrary point with coordinates (x_1, x_2).

scale the pitch error and define the origin as zero.

$$X_1 = (x_1 - c_{1,3})(\frac{c_{2,5} - c_{2,3}}{c_{1,5} - c_{1,3}}) \tag{4.7}$$

$$X_2 = x_2 - c_{2,3}. \tag{4.8}$$

The transformation can be used to map the point p to the (X_1,X_2) equivalent P. The radial distance r of point P from the (X_1,X_2) origin is used in the third control error metric. A maximum allowable radius r_{max} defines a small, circular region centered in the plane, described by the x_1 and x_2 membership function centers in the (X_1,X_2) plane, calculated as

$$r_{max} = c_{2,4} - c_{2,3}. \tag{4.9}$$

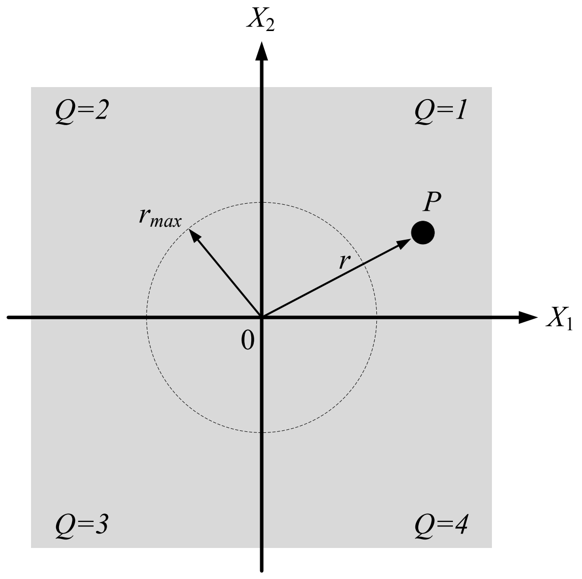

Figure 4.2: Modified planar subspace for use in training criteria. Q represents the Cartesian quadrant within the plane.

The control error ε_3, control error tolerance δ_3, and training satisfaction criteria are then defined directly as

$$\varepsilon_3 = r, \tag{4.10}$$

$$\delta_3 = r_{max}, \tag{4.11}$$

and

$$\varepsilon_3 \leq \delta_3. \tag{4.12}$$

The maximum allowable radius is defined in this manner to produce trained jumps that return the torso state to a central region of the working range of the fuzzy controller. In a similar manner to the previous satisfaction criteria, this provides the system with a margin of stability in relation to the system state, which can be graphically interpreted as the shaded region of the plane in Fig. 4.2 outside of the circle. When the torso state is within this shaded region, the fuzzy controller will select control parameters that attempt to return the state to the circular region.

The location of the point P is further defined using the standard notation for quadrants in a Cartesian plane, as shown in Fig. 4.2, represented by the variable Q. The quadrant Q is used in the training algorithm to identify the nature of failure in a training jump. The error metrics ε_1 and ε_2 represent error of a one dimensional nature, and as a result, the direction of change needed for improved jump performance can be described as positive or negative. In contrast to this, the torso state error metric ε_3 is used to account for error in two dimensions and requires more discretion to determine the direction needed to improve jump performance. As an example, a jump that fails to satisfy the torso state satisfaction criteria and is located in quadrant 3 typically represents a case where the torso pitch and pitch rate both need to be increased to reach the acceptable range.

4.3 Training Cases

The error feedback provided by the training satisfaction criteria and associated metrics defined by Eqs. 4.1, 4.2, and 4.10 is used to diagnose the nature of failure observed for a training jump. The values of the error are then supplied to training laws that serve to update specific control parameters in an attempt to reduce the error during the following training jump. The selection of the training law appropriate to the given failure mode is performed through the use of training cases, which integrate heuristic information with a structured training scheme.

Table 4.1: Training Case Structure

Case Number	ε_1	ε_2	Q
1	OK	OK	1
2	OK	OK	2
3	OK	OK	3
4	OK	OK	4
5	OK	+	n/a
6	OK	-	n/a
7	+	OK	n/a
8	+	+	n/a
9	+	-	n/a
10	-	OK	n/a
11	-	+	n/a
12	-	-	n/a

The training cases are intended to observe the failure mode of the training jump, determine the most significant aspect of the failure, and supply an appropriate training law. The training laws are formulated to modify the control parameters through small changes that will cause the training to converge to an acceptable jump. The cases are defined by twelve specific combinations of the error metrics and are dependent on the sign of the error, as described in Table 4.1.

The combined pitch and pitch rate error metric ε_3 is included only in the first four training cases. These four cases represent situations in which the jump height and forward angular rate are in the acceptable range and only the torso state fails to satisfy the training criteria. The additional cases ignore the torso state and focus only on the jump height and forward angular rate errors. The purpose of this structure is to focus on the gross performance of a jump before examining the satisfaction of the torso state criteria. The first four cases serve to fine tune the control parameters once the remaining training cases have performed the task of generating an approximately acceptable jump.

4.3.1 Training Laws

The training laws, each associated with a particular case, are the repository for heuristic information about control of the system. Selection of which control parameters and errors to use in the training law is based upon knowledge of the system developed by the user. The laws modify control parameters in proportion to the observed error of the previous training jump. Training gains are used to specify the proportion, represented by $k_{J,R}$, where J is the joint [h-hip,k-knee] and R is the type [c-current,a-angle]. The gains are selected to provide stable, convergent training behavior. Excessively high training gains can cause the training to diverge from stable, acceptable jumping behaviors. Conservative training gains can require large numbers of iterative training jumps to achieve a desired result; however, the stability offered to the training outweighs the impact of a slow training algorithm. A maximum number of training iterations n_{max} is set for any given jump so that fuzzy rules for which no acceptable combination of control parameters may exist will not stop the algorithm from further training the rule-base.

The training laws for the twelve cases are provided in Table 4.2. The conditional statements included in training cases one and ten were found to be necessary additions to the training. These cases were observed to exhibit an oscillatory behavior during training, with training jumps repeatedly switching back and forth between the two. The conditional statements provide more discretion to the training algorithm and help to eliminate this observed oscillation. The numerical values defining the conditional

Table 4.2: Training Laws

Training Case	Law	Addl. Conditions
1	$\Delta i_{h,thr} = k_{h,t} \cdot \varepsilon_3$	if $i_{h,thr}$ <11.95
	$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_3$	if $i_{h,thr}$ <11.95
	$\Delta i_{h,cat} = k_{h,c} \cdot \varepsilon_3$	if $i_{h,thr}$ >11.95 and $i_{h,cat}$ <11.95
	$\Delta i_{k,cat} = k_{k,c} \cdot \varepsilon_3$	if $i_{h,thr}$ >11.95 and $i_{h,cat}$ <11.95
	$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_3$	otherwise
2	$\Delta i_{h,thr} = k_{h,t} \cdot \varepsilon_3$	
	$\Delta i_{k,cat} = -k_{k,c} \cdot \varepsilon_3$	
3	$\Delta i_{h,thr} = -k_{h,t} \cdot \varepsilon_3$	
	$\Delta i_{k,thr} = -k_{k,t} \cdot \varepsilon_3$	
4	$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_3$	
	$\Delta i_{k,cat} = k_{k,c} \cdot \varepsilon_3$	
5	$\Delta \theta_{h,d} = -k_{h,a} \cdot \varepsilon_2$	
6	$\Delta \theta_{h,d} = -k_{h,a} \cdot \varepsilon_2$	
7	$\Delta i_{h,cat} = -k_{h,c} \cdot \varepsilon_1$	
	$\Delta i_{k,cat} = k_{k,c} \cdot \varepsilon_1$	
8	$\Delta \theta_{h,d} = -k_{h,a} \cdot \varepsilon_2$	
9	$\Delta \theta_{h,d} = -k_{h,a} \cdot \varepsilon_2$	
10	$\Delta i_{h,thr} = -k_{h,t} \cdot \varepsilon_1$	if $i_{h,thr}$ <11.75
	$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_1$	if $i_{h,thr}$ <11.75
	$\Delta i_{h,cat} = -k_{h,c} \cdot \varepsilon_1$	if $i_{h,thr}$ >11.75 and $i_{h,cat}$ <11.75
	$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_1$	if $i_{h,thr}$ >11.75 and $i_{h,cat}$ <11.75
	$\Delta i_{k,thr} = k_{k,t} \cdot \varepsilon_1$	otherwise
11	$\Delta \theta_{h,d} = -k_{h,a} \cdot \varepsilon_2$	
12	$\Delta \theta_{h,d} = -k_{h,a} \cdot \varepsilon_2$	

statements are used to determine how close the current control parameters are to the physical limitations of the system. Training case 1 uses a limit of 11.95 Amps on the hip command currents for **catch** and **thrust** to determine if small changes to these values may be made that respect both the physical limitations and improve the torso state. Training case 10 uses a lower limit of 11.75 Amps on the same control parameters in an attempt to address other control parameters for correcting the jump height error and leave more range in the hip command currents for fine tuning the torso state. Adding these conditional statements prevents training cases 1 and 10 from manipulating the same control parameter and switching back forth from torso state error to jump height error.

4.4 Physical Limitations

The training of the fuzzy rule-base must respect the physical limitations of the robot for the control approach to be realized in hardware. Many of the physical limits are incorporated directly into the model of the system, such as joint limits. However, some of the limits must be expressed directly in the training algorithm. In particular, the command currents must be closely monitored.

The maximum current i_{max} that the amplifiers are capable of delivering is included in the modeling of the electrical system. However, the primary action of many of the above training cases involves modification of the various command current control parameters. The training laws, implemented without consideration of the amplifier limits, can potentially increase the command currents beyond the realizable values in an attempt to train for an acceptable jump. This behavior can allow the training algorithm to modify the control parameters beyond the region in which an acceptable jump exists. To counteract this effect, command currents (for both **catch** and **thrust**) are examined after the training laws are applied for each jump. The command currents are modified to fall within the limits of the amplifiers.

$$i_{c,\mathrm{i}} = \begin{cases} i_{max} & \text{for } i_{c,\mathrm{i}} > i_{max} \\ i_{c,\mathrm{i}} & \text{for } -i_{max} \leq i_{c,\mathrm{i}} \leq i_{max} \\ -i_{max} & \text{for } i_{c,\mathrm{i}} < -i_{max}. \end{cases} \tag{4.13}$$

The use of this limit after the training law has been enacted allows for alteration of any control parameters for which physical limits have not been exceeded.

4.5 Seeding the Training

The training algorithm, while capable of varying the control parameters required to perform a specific desired jump, offers the best performance when provided with approximate starting values of the control parameters. The set of starting values for the control parameters are considered as a **seed**. The seed serves to reduce the number of iterations required for training a given rule by initializing the training in the approximate region of the design space that will yield appropriate control parameters. The technique of seeding the training algorithm parallels the use of initial values in numerical solution algorithms.

The initial seed for the rule-base is trained manually through manipulation of the control parameters and observation of the performance. The human-in-the-loop nature of the manual seed determination offers flexibility to the process that avoids more laborious searches of the entire design space. The initial seed used in this work is provided in Table 4.3. This seed is used as the starting point for the training of the first rule in the training order.

The training begins with the rule located at an edge of the rule-base and proceeds out through the rule-base in a dictated order. The rule is identified by the membership function centers used to determine the fuzzy control inputs, with $x_1 = c_{1,5}$, $x_2 = c_{2,5}$, $x_3 = c_{3,3}$, and $x_4 = c_{4,1}$. The rules subsequent to the first rule are then provided with trained control parameter from previously trained rules to use as seeds. This method of seeding serves two distinct purposes. Primarily, as mentioned above, seeding the

Table 4.3: Initial Training Seed

Control Parameter	Value
Desired Hip Angle	-0.82 rad
Desired Knee Angle	1.28 rad
Hip Catch Current	9.56 A
Knee Catch Current	-3.67 A
Hip Thrust Current	12.0 A
Knee Thrust Current	-7.31 A

training of a rule reduces the number of iterations required for success, reducing the training time for the entire rule-base while avoiding the necessity of manually creating a seed for every rule. Additionally, using previously trained rules as seeds for subsequent rules serves to maintain consistent behavior between adjacent rules in the rule-base. Drastic changes in the control parameters between adjacent rules could evolve different jumping behaviors and introduce complications into the fuzzy control algorithm.

The order of training the rules in the rule-base and the method of seeding subsequent rules are closely tied. The rules must be trained in an order such that appropriate seeds are available from previously trained rules. The seeding should be performed in a manner that elicits as much consistency of behavior across the rule-base as possible. These two considerations led to the training order and seeding pattern used in this work.

Consider again a planar subspace of the rule-base, shown in Fig. 4.3, formed by holding x_3 and x_4 fixed at values equal to their respective fuzzy input membership function centers. The two remaining inputs x_1 and x_2 form the axes of the planar

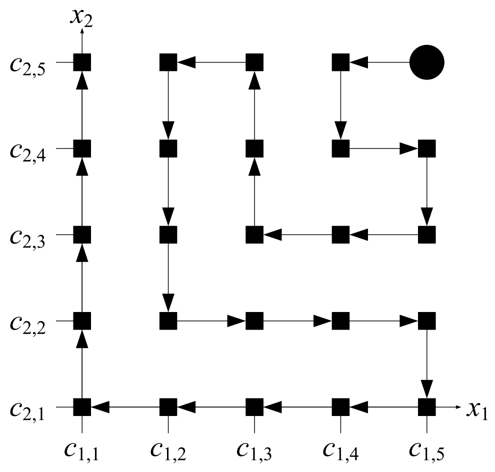

Figure 4.3: Progression of rules for training, (x_1,x_2) plane

subspace. The fuzzy rules in this plane exist at the intersection of the membership function centers for the two inputs, producing 25 rules in the plane ($n_{mf}^2 = 5^2 = 25$) shown as discrete points in the figure. The training algorithm begins in the plane at the top, right corner rule (depicted as a circle) and progresses from rule to rule as depicted by the arrows in the figure.

A planar space of the type depicted in Fig. 4.3 exists for every combination of the input x_3 and x_4 membership function centers, producing a total of 25 planes. The training proceeds from plane to plane in a specified pattern, with every plane starting at the corner rule and following the progression shown in Fig. 4.3 before moving on to the next plane. The progression between the planes is depicted in Fig. 4.4. The seeding is performed within the (x_1, x_2) plane starting from the top, right corner rule following the pattern shown in Fig. 4.5.

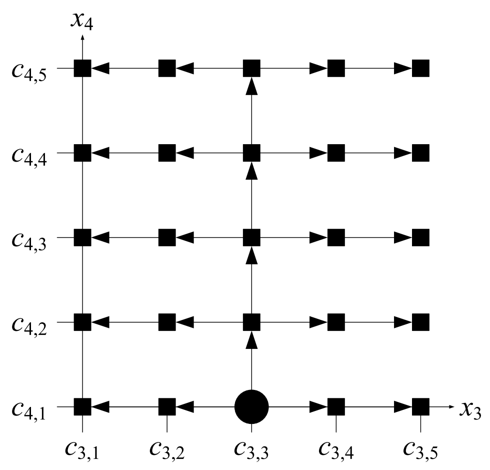

Figure 4.4: Progression of rule training and seeding, (x_3,x_4) plane

The top, right rule of every (x_1, x_2) plane, subsequent to the initial manually seeded rule, is seeded with trained control parameters from the corner of an adjacent plane, using the same pattern as the training progression depicted in Fig. 4.4.

4.6 Training in Parallel

The computational requirements of the simulation software place a high demand on the processing resources of the computer used to run the training algorithm. While the software is not optimized for use on computer systems utilizing multi-core processors, such systems can reduce the time required to complete the training algorithm in comparison to single-core systems. Multiple instances of the simulation software can be run in parallel on multi-core systems, which allows for each instance of the software to make use of a single processor core.

The linear order of the training progression requires some manipulation to allow for multiple instances of the algorithm to run simultaneously. The training order

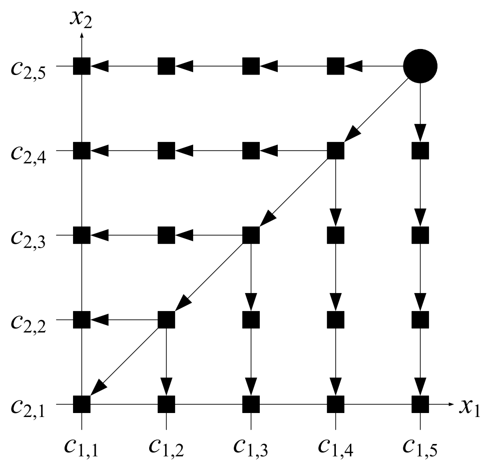

Figure 4.5: Progression of rule seeding, (x_1,x_2) plane

described above is manually overridden to determine the control parameters for a rule closer to the center of the rule-base, identified as rule number z_{ss} (in this case, where $x_1 = c_{1,5}, x_2 = c_{2,5}, x_3 = c_{3,3},$ and $x_4 = c_{4,3}$). This fuzzy rule is located approximately 40% of the way through the training progression. The training algorithm is then run simultaneously within two instances of the simulation software. One instance begins with the initial seed for the first rule in the training progression and proceeds up to rule number z_{ss}. The second instance of the simulation begins at rule number z_{ss} and proceeds through the training progression to the end of the rule-base. The division of the rule base in this manner is used to improve the training time on a dual-core computer. The two segments of the rule-base trained by each instance of the simulation are not of equal size. This reduces some of the time improvement possible through training in parallel; however, it allows for easier reconstruction of the complete rule-base from the two distinct parts produced by the separate training instances.

The results of the training algorithm run in parallel are identical to those that would be achieved by a single instance of the training simulation. The seeding progression described above is maintained for the training algorithm even when the training order is manually overridden. This method of dividing the rule-base into multiple segments can be extended for use on computers with larger numbers of processor cores or even used on multiple computer systems for further improvement of the training time.

4.7 Training Challenges

Satisfactory completion of the fuzzy rule-base required a number of large changes to the original version of the training algorithm presented above. In particular, the training satisfaction criteria, seeding pattern, and training case structure and control laws are the final result of many iterative attempts that were unsuccessful in completing the training as desired. The specific problems encountered and the resulting changes are described below.

The initial training satisfaction criteria did not include the combined torso pitch and pitch rate error ε_3. In place of this combined error metric, the torso pitch and pitch rate were each considered as separate metrics and were given acceptable ranges (similar to δ_1 and δ_2). The training cases and associated training laws were structured to handle the two metrics separately. In practice, it proved impossible to make an appreciable reduction in one of the error metrics without greatly affecting the other. This reflects on the tight coupling between the two. The revision to the training algorithm to handle the torso pitch and pitch rate as a single error metric was implemented along with a revised structure of the fuzzy controller to include the two as fuzzy control inputs.

The forward angular rate was also a necessary addition to the training algorithm. Early versions of the training algorithm attempted to restrict the forward motion of the system based entirely on the angular value of θ_7 at top-of-flight, as opposed to its time derivative. It proved difficult to define control error tolerances on the angular value that would ensure that θ_7 would fall within the working range of the fuzzy controller. To address this issue, the forward angular rate was included directly in the training algorithm in place of the angle.

The seeding pattern used within the (x_1,x_2) plane, as shown in Fig. 4.5, was created in response to a hysteretic behavior observed in the training algorithm using a previous seeding pattern. The original seeding and training pattern began in the center of the (x_1,x_2) plane and worked outwards in a radial fashion. This pattern was intended to produce more consistent fuzzy rules across this plane by minimizing the average distance of any given rule and the initial seed of the plane. However, the direction of seeding was observed to influence the order in which the training laws were used to find an acceptable combination of control parameters. Rules that were seeded from previous control parameters for more negative values of $\theta_{5,tof}$ and $\dot{\theta}_{5,tof}$ were found to quickly reach current limits for the open-loop command currents. Changing the training and seeding order allows the rules to be seeded in a consistent direction, which eliminates the hysteretic effect.

The training case structure was reduced greatly in size through the development of a working training algorithm. The original structure included a case for every combination of the error metrics (jump height, forward motion, pitch, and pitch rate) divided into sections as positive, acceptable, and negative. Extra training cases were included to account for the possibility of foot slip during the **catch** and **thrust** primitives. Implementation of this training case structure revealed several important characteristics. Many of the control laws for various cases were ultimately selected in identical forms, suggesting that the large number of training cases were unnecessary. It also proved futile to attempt to correct torso state errors in the same training jump that corrections for forward motion and jump height were applied, as the scale of changes for the jump height and forward motion often overshadowed any changes intended to affect the torso state (which is reflected in the final training case structure described above). Finally, the foot slip training cases were eliminated altogether, upon the observation that jumps including foot slip could be eliminated from the training through careful seed selection and conservative training gains.

The training laws of Table 4.2 do not include the desired knee angle at ground contact $\theta_{k,d}$. This control parameter was originally included in the training laws, but was finally removed upon observation that the training algorithm did not make significant changes to the value of the parameter for the different fuzzy rules. Additionally, it was discovered during the development of the controller that the value of the knee angle at ground contact was critical to avoiding contact with the joint limits. Maintaining a static value for this parameter, determined by the initial training seed, allows the training algorithm to select appropriate values for the remaining control parameters that do not violate the joint limits. The desired knee angle at ground contact, however, is kept as a control parameter for the motor primitives to maintain the flexibility of the controller for application to maneuvers that may require variation in this value.

The control laws associated with each of the training cases were modified many times before the final training algorithm was considered acceptable. The laws reflect the knowledge of the user about the dynamic behavior of the robot in reaction to small changes to specific control parameters. Through the development and revision of the training algorithm, the user's knowledge of the system was continually improved through observation and analysis of simulation results. This allowed for corresponding improvements to the training laws. Training laws that were observed to drive a jump away from convergence to an acceptable jump were examined in detail to determine what modifications would be required for convergent training behavior; this could include variation of the training gains or selection of different control parameters and error metrics for inclusion in the laws. Small changes to the model were found to require revision of the training algorithm, specifically the control laws. The heuristic knowledge contained in the training laws includes information required to successfully implement the control strategy described in this work. This knowledge is summarized following a brief description of the training results.

4.8 Training Results

The completion of the training algorithm resulted in the successful determination of control parameters for 607 of the 625 rules in the fuzzy rule-base. The average number of iterative training jumps required for a successfully trained rule was 189. A diagram of the rule-base can be seen in Fig. B.1 of Appendix B. This figure demonstrates the fuzzy rules for which the training algorithm failed.

The training algorithm took a total of 12 hours to complete. The training simulation was performed on a desktop PC with an Intel Core 2 Duo 3.0 Ghz processor and 2 Gb of RAM.

4.9 Heuristics Developed and Observed

The motivation for using fuzzy control as a supervisory controller emphasizes the inclusion of heuristic knowledge and the resulting ‘functional model’ of the system dynamics. This ‘functional model’ is contained within the control strategy at two locations: the fuzzy rule-base and the training laws. The topology of the control parameters in the rule-base throughout the space of the fuzzy controller serves as a mapping of the nonlinear relationships between control parameters and inputs. This mapping can be used to extract heuristic information about the system. Many of the same relationships are also present in more qualitative terms in the training laws used to determine appropriate control outputs for every rule. The development of the control strategy itself offered insight into the system dynamics and limitations that can be expressed in the form of heuristic statements. The following heuristic information was drawn from these three areas. This information captures details of the system dynamics that are critical for successfully implementing the control strategy described in this work.

- The primary factor restricting maximum jump height is hip actuator power. Higher jump heights approach hip power in excess of the physical limit described by i_{max}, while the knee actuators do not approach this limit.

- The torso state at top-of-flight described by both angle and angular rate is a critical factor in producing stable jumping. The inability to determine acceptable control parameters for a given torso state reflects the sensitivity of the system to these two state variables.

- The torso state at top-of-flight is directly affected by all control parameters. The magnitude of change in control parameters required to address forward motion or jump height has a significant impact on the torso state, typically requiring further refinement of the parameters to correct for the error introduced.

- Small amounts of energy stored in the actuator springs can negatively affect the ability of the fuzzy controller to accurately determine the torso state at top-of-flight. The state of all links at top-of-flight must be static enough to allow for determination of appropriate control parameters.

- The torso angular rate and position at the top-of-flight can negate one another. Fuzzy rules located on or near the negative diagonal passing through the center of the (x_1,x_2) plane typically train more rapidly than others.

- Foot position relative to the torso at ground contact has a large effect on forward motion. Training laws implemented to reduce error in the forward motion manipulate the horizontal distance between the feet and torso at ground contact through modification of the desired hip angle at contact (see training laws for cases 5, 8, and 11 for examples). This heuristic, observable in the training laws, is also an integral part of the decoupled forward motion and height control strategy implemented by Raibert [5] with monopodal and bipedal robots.

- Initial forward motion of the robot, when unaccounted for in the control parameter selection process, can cause otherwise valid control parameters to be ineffective in producing the desired jump and lead to immediate control failure in the form of falling.

- Jump height is controlled primarily through the magnitude of the hip and knee **catch** and **thrust** command currents. The jump height is particularly sensitive to the commanded knee currents, as shown by the relative size of the hip and knee current training gains seen in the training laws emphasizing jump height correction (including cases 7 and 10).

- The range of jump height described for the controller must respect the ability of the robot to achieve the necessary joint positions during all phases of a maneuver. Excessively low jump heights may not allow time for positioning of the legs during the **position** or **retract** motor primitive.

- Jumps described by lower desired jump height typically offer a higher possibility of acceptable performance. Lower height jumps require control parameters that do not push the limits of the electronics and actuators, leaving more flexibility in the control parameters for achieving an acceptable torso state at top-of-flight.

- Large torques in the knee actuators are undesirable as the robot approaches liftoff. The mechanical coupling between the knee actuators and the torso (due to the parallel actuation scheme) causes a direct transfer of torque from the shank links to the torso. This mechanical coupling creates a high sensitivity of the torso state to the torque developed by the knee actuators and can easily drive the robot out of the working range of the fuzzy controller.

- The top-of-flight joint angles should approximate a midpoint between joint angles at liftoff and touchdown. The transitions between the **hold** and **position** motor primitives will occur with less disturbance to the system as a result.
- The varied failure modes possible during a maneuver make universal training laws infeasible. Training laws must be tailored to address specific failure modes.
- Joint trajectories described by the minimum acceleration required to reach the end state are desirable for reducing disturbances to the system dynamics.

4.10 Fuzzy Training as a Design Space Search

The reduction of the design space for producing a maneuver from an effectively infinite space to the six-dimensional space described by the fuzzy control outputs improves the tractability of selecting appropriate control parameters. However, this reduced space still requires searching to find the necessary values. The training algorithm makes small changes to the control parameters via the training laws, resulting in incremental steps through the design space in search of acceptable performance. This incremental search process follows a pattern similar to that present in gradient search methods used in numerical optimization techniques such as those described by Arora [43]. Rather than using a formula to describe the cost function as with optimization techniques, this work uses the simulated jump results to evaluate the performance of the given point in the design space. The training laws replace the topology of the optimization surface in deciding the direction and dimension through which to proceed for improved performance.

Description of the training algorithm as a design space search routine is also useful when considering the motivation for using fuzzy control in this work. A traditional search of the design space using numerical search techniques would rely on an accurate analytical model of the system dynamics to describe the performance of points in the design space. The avoidance of an accurate analytical model is made possible in this work through the use of simulated jumps and user-based heuristic knowledge for improving jump performance.

4.11 Fuzzy Training as a Learning Algorithm

The training algorithm for the fuzzy controller finds control parameters that will produce the desired jump performance. The determination of these parameters can be interpreted as a learning process. Schaal [40] describes an approach to developing control of humanoid robots that implements imitation learning. The imitation learning process is performed through observation of a demonstrated maneuver. The controller attempts to represent the observed maneuver as a sequence of motor primitives already known by the system. If sufficient motor primitives are not available to the system, existing primitives may be modified or new ones created to address the shortcoming. The controller then implements the motor primitives and refines the control parameters or sequence based on feedback to improve the performance of the motion.

The training algorithm of this work follows a routine similar to this, with the motor primitive identification process removed. Refinement of the performance is completed solely through modification of the control parameters based on error feedback, essentially skipping over the identification of appropriate primitives. The identification of the appropriate motor primitives for the desired jump was performed in this work by the user and maintained as a constant throughout the training. The use of error feedback in motor control learning has been an issue of some debate, with questions raised as to the validity of the approach. In his discussion of the biological plausibility of computational cognition methods, O'Reilly [44] states that error-driven learning does not conflict with available information on biological processes.

The concepts of imitation learning and error-driven feedback support the interpretation of the training algorithm as a learning process. This interpretation helps to create a parallel between the control approach described in this work and the motor control theories used to model biological systems.

4.12 Summary

This chapter describes the training algorithm used to select control parameters for the fuzzy rule-base, including the training cases and control parameter update laws. The training algorithm represents much of the user's heuristic knowledge about control of the robot. The heuristics gained from the training algorithm and the control development process are summarized. Interpretations of the fuzzy training are provided in support of the control approach described in this work. The trained fuzzy controller produced by this training algorithm will next be implemented to validate the control strategy.

CHAPTER 5

CONROL STRATEGY RESULTS

5.1 Introduction

The trained rule-base for the fuzzy controller provides the necessary parameters for performing jumping maneuvers. To verify the capability of the control strategy to produce stable sequential jumps, the trained rule-base must be implemented. The numerical simulation was initially used to verify the stability of the controller before implementing the control strategy in the experimental biped. This chapter begins with the implementation and results of the control strategy in the numerical simulation, with emphasis on the assumptions and modifications required to produce stable jumping. The application of the control strategy to a simplified case of the bipedal system is then introduced for use on the experimental system. The chapter concludes with a discussion of the difficulties preventing implementation of the full control strategy and a number of potential solutions to these issues.

5.2 Implementing the Control Strategy in Simulation

Implementation of the control strategy in the numerical simulation is a simple process. The thigh and shank links of the robot are initially placed in the top-of-flight configuration with zero joint velocity. The boom angle θ_6 is set at a position to provide the desired initial height for the robot to fall from, while the value of $\dot{\theta}_7$ is set to the desired initial forward angular rate. The torso state is then set using a pitch and pitch rate that are within the range of the fuzzy controller. Upon the start of the simulation, the fuzzy controller is called to evaluate the current state of the robot and provide the control parameters for executing the initial jump. The state machine is then started from the beginning of the **position** motor primitive and cycles through the remaining primitives as defined in Chapter 3. Upon reaching the top-of-flight, the fuzzy controller is again called and the state machine is reset. The process of calling the fuzzy controller and executing a cycle of the state machine is repeated until the desired number of jumps have been completed or the system becomes unstable.

5.2.1 Assumptions Required for Stable Performance

The mathematical model presented in Chapter 2 includes a number of assumptions regarding the nature of the bipedal system KURMET. During the process of developing and implementing the control strategy in simulation, some of these assumptions proved problematic for stable sequential jumps. The final simulation results presented here and in Chapter 4 for the 5-DOF system model include changes to these assumptions, particularly in the contact model presented for the unidirectional elements of the series-elastic actuators.

The unidirectional contact model of Eq. 2.10 includes a linear spring and damper acting in parallel between each link and actuator when in the DDA and SEA preload regimes. The spring stiffness and damping ratio of the contact model were defined assuming a stiffness one order of magnitude higher than that of the actuator spring and a damping ratio of approximately 1.0. These values were selected to reflect the predicted behavior of the physical system. This contact model, while expected to capture the most relevant aspects of the system, introduces difficulty to the control strategy described in this work.

One of the heuristics offered in Chapter 4 describes the importance of having the entire system exhibit a smooth, consistent behavior at the top-of-flight. Due to the tight coupling between the legs and the torso of the robot, small deviations from the desired TOF state of the legs (zero velocity and defined positions) can easily change the torso state. When the fuzzy controller is called at TOF, it is critically important for the torso state to exhibit the same behavior as shown during the training algorithm. If the position and velocity of all leg links do not closely match the desired TOF states, the torso angle will not follow the trajectories expected for the following jump. As a result of this, the control parameters selected by the fuzzy controller may not be effective in returning the system to the desired jump height, forward angular rate, and torso state at the next TOF. Failure to return the system to within the bounds of the fuzzy controller quickly leads to instability.

The short period of time available for the **retract** motor primitive requires the use of trajectories that quickly return the motor positions to the desired TOF values. With closed-loop feedback control available for the motor positions, this task introduces no significant problems. However, when the compliance of the USEAs is considered, the goal of quickly returning to a desired position and achieving zero velocity in the joint angles at TOF is no longer trivial. The compliance of the actuators (both in the series-elastic and direct drive regimes) allows the joint position to oscillate around desired position after experiencing a large acceleration or deceleration. The magnitude of oscillation observed in the system when using the previously defined values for the unidirectional contact parameters is significant enough to rapidly cause instability in the system when the fuzzy controller is implemented. To counteract this instability, the contact parameters were modified to a higher stiffness and damping ratio to reduce the magnitude and settling time of these oscillations in the leg joint angles. Increasing the contact stiffness to 900 $\frac{\text{N}\cdot\text{m}}{\text{rad}}$ and the damping coefficient to 10.0 $\frac{\text{N}\cdot\text{m}\cdot\text{s}}{\text{rad}}$ was found to allow the oscillations to be reduced enough to avoid instability. In addition to the increased unidirectional contact stiffness and damping, the preload torque of the actuators was set to a value of 6.0 N·m. This allows the damping included in the preload range of the actuator to further improve the response of the leg joints to the high speed trajectories in the **retract** primitive.

The consequences of the modifications to the contact model are discussed later in this chapter as they apply to implementation of the simulation results to the experimental hardware.

5.2.2 Simulation Results

The implementation of the fuzzy controller in the numerical simulation allows for an examination of the capabilities of the control strategy to produce the desired jump performance. To this end, the simulation was used to perform multiple jumping

Table 5.1: Summarized Simulation Results

Sequence	θ_5 (rad)	$\dot{\theta}_5$ ($\frac{\text{rad}}{\text{s}}$)	h_0 (m)	$\dot{\theta}_{7,tof}$ ($\frac{\text{rad}}{\text{s}}$)	Summary
1	$c_{1,3}$	$c_{2,3}$	$c_{3,1}$	$c_{4,1}$	stable
2	$c_{1,3}$	$c_{2,3}$	$c_{3,3}$	$c_{4,1}$	stable
3	$c_{1,3}$	$c_{2,3}$	$c_{3,5}$	$c_{4,1}$	stable
4	$c_{1,3}$	$c_{2,3}$	$c_{3,1}$	$c_{4,3}$	stable
5	$c_{1,3}$	$c_{2,3}$	$c_{3,3}$	$c_{4,3}$	unstable after 13 jumps
6	$c_{1,3}$	$c_{2,3}$	$c_{3,5}$	$c_{4,3}$	stable
7	$c_{1,3}$	$c_{2,3}$	$c_{3,1}$	$c_{4,5}$	unstable after 1 jump
8	$c_{1,3}$	$c_{2,3}$	$c_{3,3}$	$c_{4,5}$	stable
9	$c_{1,3}$	$c_{2,3}$	$c_{3,5}$	$c_{4,5}$	stable
10	$c_{1,1}$	$c_{2,1}$	$c_{3,3}$	$c_{4,3}$	stable
11	$c_{1,5}$	$c_{2,1}$	$c_{3,3}$	$c_{4,3}$	stable
12	$c_{1,1}$	$c_{2,5}$	$c_{3,3}$	$c_{4,3}$	stable
13	$c_{1,5}$	$c_{2,5}$	$c_{3,3}$	$c_{4,3}$	unstable after 3 jumps
14	$\frac{c_{1,1}+c_{1,2}}{2}$	$\frac{c_{2,1}+c_{2,2}}{2}$	$c_{3,3}$	$c_{4,3}$	stable
15	$\frac{c_{1,4}+c_{1,5}}{2}$	$\frac{c_{2,1}+c_{2,2}}{2}$	$c_{3,3}$	$c_{4,3}$	stable
16	$\frac{c_{1,1}+c_{1,2}}{2}$	$\frac{c_{2,4}+c_{2,5}}{2}$	$c_{3,3}$	$c_{4,3}$	stable
17	$\frac{c_{1,4}+c_{1,5}}{2}$	$\frac{c_{2,4}+c_{2,5}}{2}$	$c_{3,3}$	$c_{4,3}$	stable

sequences for several scenarios. These scenarios describe different initial conditions for the state of the system.

The different initial conditions in the simulation are used to verify the ability of the controller to stabilize the system from different regions of the fuzzy rule-base. The different jumping sequences resulting from these initial conditions are tabulated and summarized in Table 5.1. The variables used as inputs to the fuzzy controller were initialized at values located both on and between the centers of the fuzzy membership functions.

5.2.3 Discussion of Simulation Results

The general characteristics of the stable jump sequences offer insight into the behavior of both the biped system and the control strategy. The simulation data provided by sequence 1 from Table 5.1 is discussed here as representative of the stable jump sequences detailed in the table.

The leg joint and motor positions follow similar trajectories during all cycles of the state machine. The initial cycle of the state machine from sequence 1 demonstrates these trajectories. Figure 5.1 depicts the joint positions and corresponding motor position for this cycle with the active motor primitive noted. The **position** and **retract** primitives demonstrate the close agreement between the motor and link positions during the flight phase, while the deviation between these values during **catch** and **thrust** reveals the deflection of the spring elements in the actuators. The position of the leg links at the desired angles at top-of-flight demonstrates the smooth behavior required for stability, with negligible difference between the motor and link positions. The slight disparity between the left and right leg throughout the state machine cycle is a result of the asymmetry introduced by approximating a planar restriction using a spherical surface, as discussed in Chapter 2.

The torso height h, shown in Fig. 5.2, follows a smooth trajectory throughout the cycle. The trajectory does not appear parabolic toward the end of the cycle, as would be expected of ballisic flight. The influence of the leg retraction on the net momentum of the system explains this effect. The final height of 0.58 m for the given

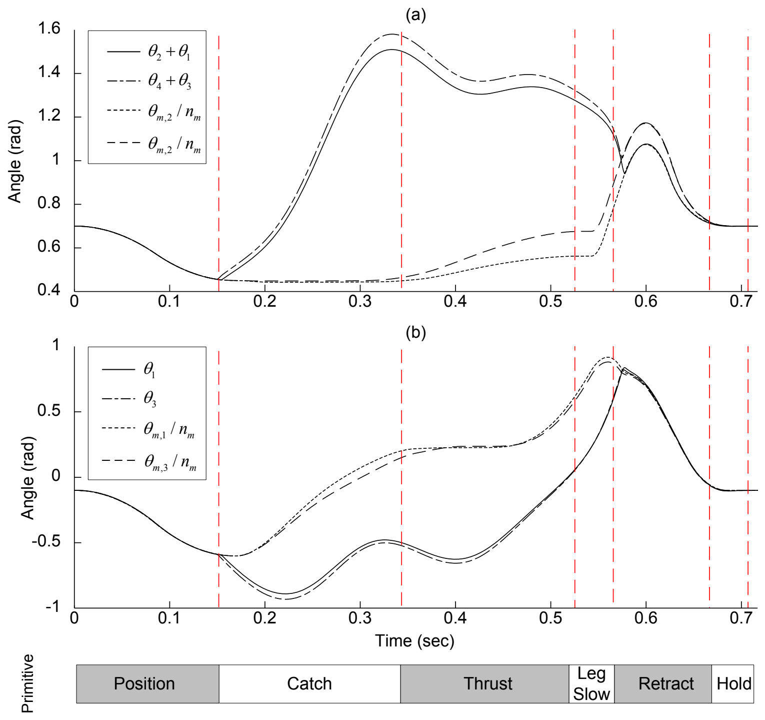

Figure 5.1: Simulated leg joint and motor positions for a single cycle of the state machine. (a) shows the behavior of the shank links and motors using the angle of the link relative to the torso, while (b) shows the thigh links and motors. Motor positions are displayed as the angle of the gearbox output. Motor primitive transitions are marked with dashed red vertical lines and noted at the bottom of the figure.

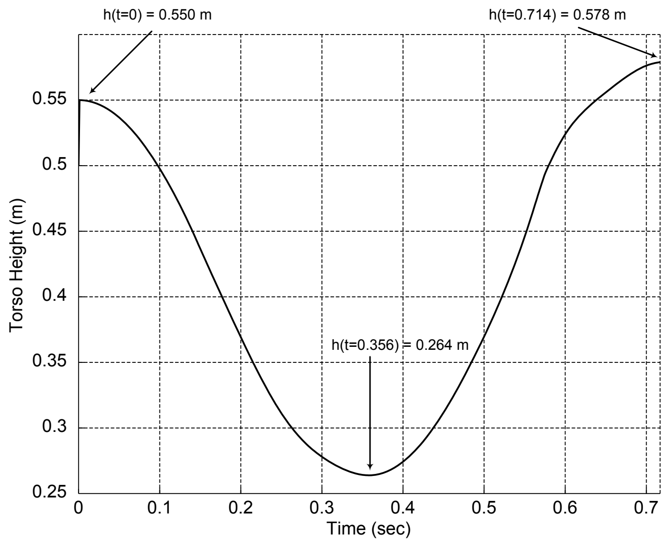

Figure 5.2: Simulated torso height h for one cycle of the state machine. The initial and final height are noted, as well as the minimum height.

cycle deviates from the desired height of 0.575 m by 0.005 m. An error in the final height is a feature common to the jumps produced by the simulation. Despite the error from the desired final height, this value falls within the range of the membership function centers for the fuzzy input of x_3. The training algorithm, as described in Chapter 4, includes an error tolerance of 0.02 m on the final height of the jump to improve the convergence of the training algorithm.

The behavior of the torso during a jump cycle is also significant to the control strategy. Figure 5.3 shows the torso angle θ_5 and angular rate $\dot{\theta}_5$ through the jump cycle. The minimum and maximum values of the fuzzy control input membership functions are included to show the return of these variables to the desired range at the top-of-flight. The oscillation of the torso state, both the angle and angular rate, are not of concern at times other than the top-of-flight. However, the value of

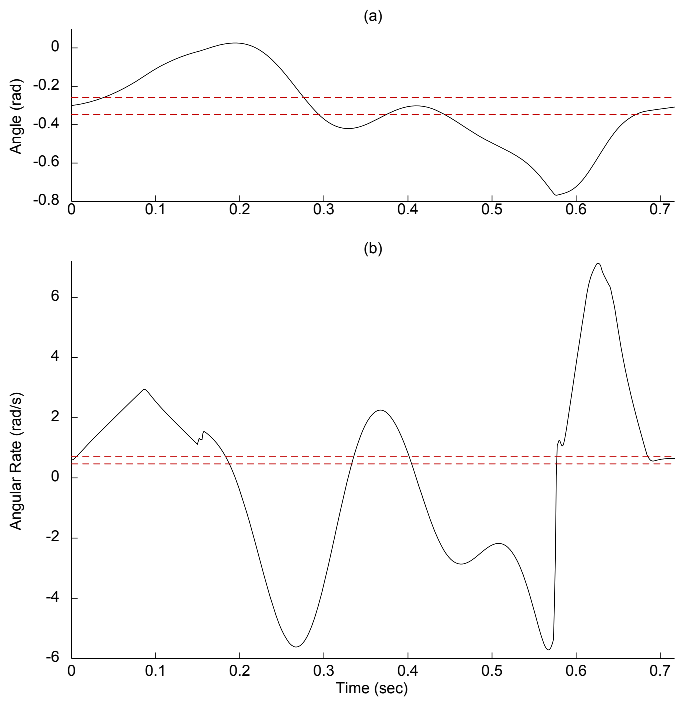

Figure 5.3: Simulated torso state (a) θ_5 and (b) $\dot{\theta}_5$ for one cycle of the state machine. The boundaries of the fuzzy control inputs are shown for each as dashed red horizontal lines.

these variables at the top-of-flight must be consistent enough to allow for the fuzzy controller to select appropriate control parameters for the following cycle. As the cycle nears the top-of-flight, both the torso angle and angular rate exhibit a smooth behavior.

The final input to the fuzzy controller, the forward angular rate, follows the trend displayed by the torso angle and angular rate. The value of $\dot{\theta}_7$ varies throughout the cycle, but ultimately returns to the desired range (defined by the minimum and maximum membership function centers for x_4). Figure 5.4 demonstrates this characteristic of a stable cycle of the state machine. It is important to note that while the forward angular rate at the top-of-flight is measured by the fuzzy controller, the

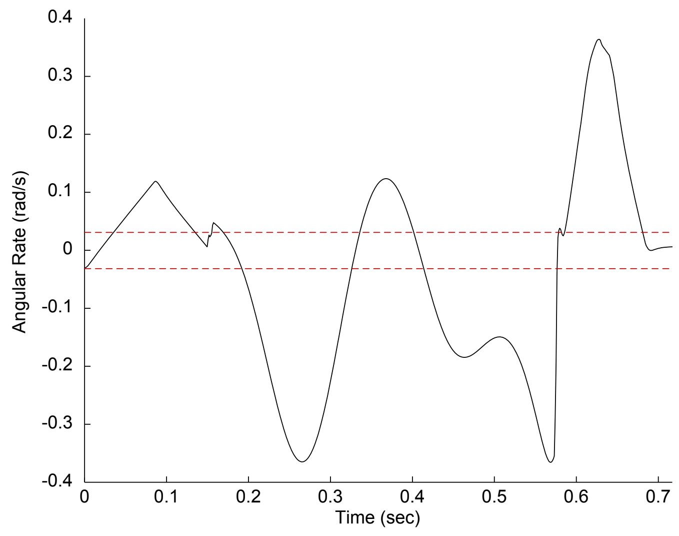

Figure 5.4: Simulated forward angular rate $\dot{\theta}_7$ for one cycle of the state machine. The boundaries of the fuzzy control inputs are shown as dashed red horizontal lines.

forward angular position is not. As a result of this, the system shows a tendency to drift in the forward or backward direction during a jump sequence. The fuzzy controller attempts to correct error in the forward angular rate by returning the system to $\dot{\theta}_7 = 0$, but does not attempt to correct the forward position of the robot. Deviation from the initial forward position of θ_7 can accumulate over sequential jumps, causing the drift observed in the simulation jump sequences.

The stable jump sequences described in Table 5.1 exhibit the characteristics described in detail for the single cycle of the state machine. In particular the sequences demonstrate

- a consistent return of the variables representing fuzzy control inputs x_1:x_4 to within the range of the relevant membership functions, that is $c_{i,1} < x_i < c_{i,5}$ for $i = 1 : 4$,

- a consistent, smooth behavior of these variables around top-of-flight,
- and a negligible difference between the position of the actuated links and their corresponding motors around top-of-flight.

These characteristics are discussed in Section 5.4.1 as they relate to implementation of the control strategy on the experimental hardware.

While the stable jump sequences demonstrate the desirable characteristics of the control strategy, the unstable sequences offer more detail regarding the limitations. Three of the seventeen jump sequences detailed in Table 5.1 became unstable. One of these three, sequence 7, represents a jump for which the failure can be directly attributed to divergence of the training algorithm. The fuzzy rule defined by the input membership function centers $c_{1,3}, c_{2,3}, c_{3,1}$, and $c_{4,5}$ was not successfully trained. As a consequence of this, the control parameters specified by this fuzzy rule do not return the state of the system to within the desired range of the fuzzy controller. The left and right membership functions defined for the fuzzy inputs include saturation, which allows the fuzzy controller to determine the most applicable fuzzy rules for top-of-flight states outside of the desired range. However, the system is unable to recover from the end state of this initial jump, and falls immediately. The two remaining failed jump sequences, numbers 5 and 13, do not share this problem of initialization on an untrained fuzzy rule.

The first jump of sequence 13 successfully returns the state of the system to within the range of the fuzzy controller. Despite this, the control strategy fails to maintain stability. A similar problem occurs here to that described for sequence 7. The state of the system at the top-of-flight of the first jump, while within the range of the fuzzy controller, exists at a point in the fuzzy rule base that contains fuzzy rules that

Table 5.2: Top-of-flight States for Sequence 13

Jump Number	θ_5 (rad)	$\dot{\theta}_5$ ($\frac{\text{rad}}{\text{s}}$)	h (m)	$\dot{\theta}_{7,tof}$ ($\frac{\text{rad}}{\text{s}}$)
Init	-0.300	0.600	0. 550	0.030
1	-0.312	0.641	0.558	0.003
2	-0.281	0.742	0.580	-0.001
3	-0.177	1.04	0.571	-0.080

failed to train. The influence of these rules on the following jump, while reduced by including the training satisfaction flag in Eq. 3.29, is significant enough to cause the system to become unstable. The controller is able to produce an additional two jumps before completely failing to maintain an upright configuration of the robot. The deviation of the torso state fuzzy control inputs from the desired range increases during these additional jumps as noted in Table 5.2, until there is no chance for recovery.

The instability demonstrated by sequence 5 does not exhibit a problem with untrained fuzzy rules as observed for sequences 7 and 13. The initial stability of the sequence, as demonstrated by the first 10 repetitive jumps, suggests that the system is in a stable mode similar to that demonstrated by the remaining jump sequences. The sudden instability, observable as a top-of-flight torso angular rate outside the range of the fuzzy controller at the end of the eleventh jump, suggests a different issue with the sequence. To verify the ability of the control strategy to stabilize the system at the top-of-flight conditions observed at the end of jump number 10, the simulation was initialized using these values, listed in Table 5.3. The resulting jump sequence exhibits stable performance. This result suggests that the instability of

Table 5.3: Top-of-flight States for Sequence 5

Jump Number	θ_5 (rad)	$\dot{\theta}_5$ ($\frac{\text{rad}}{\text{s}}$)	h (m)	$\dot{\theta}_{7,tof}$ ($\frac{\text{rad}}{\text{s}}$)
Init	-0.300	0.600	0. 575	0.000
1	-0.314	0.635	0.590	0.008
2	-0.306	0.641	0.585	0.004
3	-0.308	0.637	0.584	0.004
4	-0.307	0.635	0.585	0.003
5	-0.304	0.657	0.586	0.005
6	-0.308	0.645	0.582	0.004
7	-0.304	0.668	0.581	0.004
8	-0.307	0.648	0.580	0.002
9	-0.302	0.676	0.580	0.004
10	-0.310	0.633	0.579	0.001
11	-0.294	0.708	0.583	0.005
12	-0.281	0.724	0.581	-0.013
13	-0.232	0.857	0.565	-0.045

jump sequence 5 is a numerical anomaly of the simulation. Further support for this conclusion is demonstrated by varying the initial conditions of sequence 5 and observing the resulting performance. The varied conditions are shown in Table 5.4 with a summary of the resulting stability. These results demonstrate that minor changes to the initial conditions result in stable jump sequences, indicating that the instability of sequence 5 is a flaw in the numerical simulation as opposed to the control strategy.

The simulated jump sequences described above demonstrate the capability of the control strategy to produce stable dynamic maneuvers using the five degree-of-freedom model of the bipedal system. The control strategy is able to successfully produce continuous vertical jumping with an approximate normalized jump height s of 2.3. The jump sequences observed to be unstable indicate that the performance

Table 5.4: Varied Sequence 5 Results

Sequence	θ_5 (rad)	$\dot{\theta}_5$ ($\frac{\text{rad}}{\text{s}}$)	h_0 (m)	$\dot{\theta}_{7,tof}$ ($\frac{\text{rad}}{\text{s}}$)	Summary
5	$c_{1,3}$	$c_{2,3}$	$c_{3,3}$	$c_{4,3}$	unstable after 14 jumps
5.1	$c_{1,3} \times 1.001$	$c_{2,3}$	$c_{3,3}$	$c_{4,3}$	stable
5.2	$c_{1,3} \times 0.999$	$c_{2,3}$	$c_{3,3}$	$c_{4,3}$	stable
5.3	$c_{1,3}$	$c_{2,3} \times 1.001$	$c_{3,3}$	$c_{4,3}$	stable
5.4	$c_{1,3}$	$c_{2,3} \times 0.999$	$c_{3,3}$	$c_{4,3}$	stable

of the simulated system is ultimately reliant upon the quality of control parameters selected by the fuzzy training algorithm.

5.3 Implementing the Control Strategy in Hardware

The results presented above demonstrate the capability of the control strategy for producing stable jumping under the ideal conditions modeled by the simulation. When the experimental hardware is used in place of the simulation, a number of issues arise that prevent the implementation of the full control strategy. To address these issues, a simplified case of the bipedal system is used for the experimental hardware.

5.3.1 Simplified Bipedal System

The simplification of the bipedal system consists of eliminating one of the five degrees of freedom. The mechanical design of KURMET includes a removable pin which, when in place, locks the torso into an upright configuration ($\theta_5 = 0.05$ rad) and eliminates the freedom of the robot to rotate about this axis. Elimination of this degree of freedom simplifies the control strategy by avoiding stability issues related to the torso pitch and pitch rate.

The inclusion of the pin is modeled in simulation through placing a proportional-derivative (PD) controller on θ_5 in the form of

$$\tau_5 = k_p(\theta_{5,d} - \theta_5) + k_d \frac{d}{dt}(\theta_{5,d} - \theta_5), \tag{5.1}$$

where τ_5 is the torque applied to the joint, $\theta_{5,d}$ is the desired torso angle, and k_p and k_d are the proportional and derivative gains. The rate of change of the difference in desired and actual torso angle $\frac{d}{dt}(\theta_{m,tof} - \theta_{m,\mathrm{i}})$ is calculated as

$$\frac{d}{dt}(\theta_{5,d} - \theta_5) = \frac{(\theta_{5,d} - \theta_5) - (\theta_{5,d} - \theta_{5,p})}{\Delta t}, \tag{5.2}$$

where $\theta_{5,p}$ is the torso angle at the previous control step and Δt is the time between control steps. The gains are manually tuned to stiff values to maintain a low error from the desired value.

5.3.2 Modifications to the Control Strategy

The control strategy was modified both at the supervisory and motor primitive level for implementation on the simplified biped system. In particular, the structure of the fuzzy controller was changed with respect to the control inputs, and the motor primitives were modified to address issues observed during implementation on the experimental hardware. The following changes focus on these two areas; however, the general approach of the full control strategy described in Chapter 3 remains the same.

- The revised fuzzy control structure consists of three inputs, the current jump height h_{max}, the desired jump height h_d, and the average forward angular rate $\dot{\theta}_{7,ave}$ measured from the termination of the **thrust** primitive to the top-of-flight, identified as fuzzy inputs $\bar{x}_1$, $\bar{x}_2$, and $\bar{x}_3$ respectively. Each input is

Table 5.5: Control Input Membership Function Centers, Simplified Case

Symbol	Input	Membership Function Centers
$\bar{x}_1$	Current Jump Height, m	0.550 , 0.5625 , 0.575 , 0.5875 , 0.600
$\bar{x}_2$	Desired Jump Height, m	0.550 , 0.5625 , 0.575 , 0.5875 , 0.600
$\bar{x}_3$	Average Forward Angular Rate, $\frac{\text{rad}}{\text{s}}$	-0.10 , -0.05 , 0.00, 0.05 , 0.10

described using five triangular membership functions as shown in Fig. 3.3 with membership function centers provided in Table 5.5. The total number of fuzzy rules in the resulting rule-base is thus 5^3=125.

- The average forward angular rate is used as a fuzzy input in place of $\dot{\theta}_{7,tof}$ due to the level of noise present in the numerically differentiated encoder signal used for θ_7. Using the average forward angular rate allows the fuzzy controller to receive a single value that describes the current behavior of the complete system. This also avoids delay issues introduced by filtering the signal during real-time operation.

- The rule-base was trained in simulation using the training algorithm described in Chapter 4, with several changes corresponding to the revised fuzzy structure. The control error ε_3 and training cases one through four are eliminated, as they are no longer applicable. The training algorithm uses the desired height h_d defined by the membership function centers of $\bar{x}_2$ when calculating the error metric ε_2 in Eqn. 4.1 in place of a constant value. This allows the fuzzy rules to be trained to reach the varied jump heights described by the range of values for $\bar{x}_2$ in Table 5.5. The error tolerance for the height is set at 0.001 m to

Table 5.6: Initial Training Seed, Simplified Case

Control Parameter	Value
Desired Hip Angle	-0.55 rad
Desired Knee Angle	0.80 rad
Hip Catch Current	5.00 A
Knee Catch Current	-3.00 A
Hip Thrust Current	10.00 A
Knee Thrust Current	-8.00 A

improve the height accuracy of the controller for the simplified case. The seed used for the training algorithm for the simplified case can be found in Table 5.6. Acceptable control parameters were found for all 125 rules in the fuzzy rule-base for the simplified case.

- The state machine was modified to include an additional **hold** primitive between the **position** and **catch** primitives. This additional primitive was introduced in response to the inability to precisely determine the point during the jump cycle that top-of-flight is achieved. Identical control laws to those defined in Section 3.2.8 are used for executing this primitive, with the desired values changed to the fuzzy control outputs of $\theta_{h,d}$ and $\theta_{k,d}$ in place of the top-of-flight angles. The angular position signal produced by the shaft encoder placed on the joint of θ_6 includes significant noise when numerically differentiated to determine $\dot{\theta}_6$. As a result, the determination of the top-of-flight is imprecise when using a change in sign of $\dot{\theta}_6$. The precision is acceptable for determining the height at top-of-flight, but is problematic when the following cycle of the state machine is performed. Performing the following **position** primitive and terminating the primitive after the completion of $\Delta t_{p,1}$ allows the open-loop control currents of the **catch** primitive to be implemented at inconsistent points relative to the time of ground contact. This inconsistency can drastically change the behavior of the following jump cycle. Introducing the additional **hold** primitive allows for the leg joints to be kept at the desired angles until ground contact is sensed. The implementation of the **catch** command currents then produces consistent behavior between jumps.

- The point of ground contact was originally determined from the mechanical switches integrated into the feet of the robot. However, it was observed during experimentation that the delay introduced between the sensing of ground contact and the beginning of the **catch** motor primitive allowed for the actuator behavior to deviate from that predicted in simulation. To counteract this, the **hold** motor primitive was changed to terminate upon either the left of right foot height, $h_{f,l}$ and $h_{f,r}$ respectively, becoming less than an experimentally tuned threshold. This ensures that the open-loop command currents of the **catch** primitive are implemented in a timely manner.

- The **leg slow** motor primitive was eliminated entirely. This primitive was found necessary in the full control strategy to slow the leg joint velocities and avoid striking the joint limits after lift-off. When the torso is locked into the upright position, the leg configurations used to produce jumping do not require this primitive to avoid striking the joint limits. Removing the **leg slow** primitive allows for more time to be allocated to the **retract** primitive, improving the response of the joint trajectories by decreasing the magnitudes of the accelerations required.

- The calculation of the primitive periods for the **catch** and **retract** primitives were changed to ensure completion of the primitives before ground contact and top-of-flight, respectively. The **position** time period $\Delta t_{p,1}$ uses a leading coefficient of 0.60 in place of 0.75, while the leading coefficient for the **retract** period of $\Delta t_{p,5}$ was changed from 0.45 to 0.60. These reduced values ensure that the primitives are completed in a satisfactory time, allowing the following **hold** motor primitives to fill the remaining time between their completion and the desired event.

- The top-of-flight joint angles were modified during the implementation. Oscillation in the joint angles during the flight phase of the jump maneuvers was found to result in inconsistent leg positions at the time of ground contact. The top-of-flight angle for the thigh links was changed to -0.30 rad from -0.40 rad. This modification causes the hip motor trajectories defined by the **position** motor primitive to include larger angle changes. As a result, the positions of both the hip and knee joints show a tendency to overshoot the desired ground contact values. When oscillations are present in the leg joints, this overshoot reduces the possibility of experiencing ground contact at joint angles less than the desired values. Jump cycles that experience ground contact with the joint angles less than the desired position were observed to produce a visibly different motor behavior during the subsequent ground contact phase. While overshoot

Table 5.7: Modified State Machine and Motor Primitives, Simplified Case

Index	Motor Primitive Name	Starting Event	Ending Event
1	Position	$f_{tof} = 1$	$\Delta t_{p,1}$ Expired
2	Hold	$\Delta t_{p,1}$ Expired	$h_{f,l} < 0.01$ or $h_{f,r} < 0.01$
3	Catch	$h_{f,l} < 0.01$ or $h_{f,r} < 0.01$	$\dot{\theta}_{\mathrm{i}} < \dot{\theta}_{lim}$
4	Thrust	$\dot{\theta}_{\mathrm{i}} < \dot{\theta}_{lim}$	$\theta_{\mathrm{i}} < \theta_{lim}$
5	Retract	$\theta_{\mathrm{i}} < \theta_{lim}$	$\Delta t_{p,5}$ Expired
6	Hold	$\Delta t_{p,5}$ Expired	$f_{tof} = 1$

of the desired angles may interfere with precise control of the forward angular rate, the overall stability of the system is improved by the more consistent motor behaviors observed during the ground contact phase.

- The assumed value of the back efficiency of the gearbox η_b was modified from the initial value of $\frac{\eta_f}{2}$ to a value of 0.58. The initial value was based on an approximation made by Curran [15] to address the inability to reliably assign a value based on the manufacturer's specifications. Comparisons made between simulated motor positions and experimentally measured motor positions during jump cycles indicated that the back efficiency was in fact higher than initially assumed. The final value of $\eta_b = 0.58$ was manually tuned until similar behavior was observed between experimental and simulated motor data.

The revised state machine used for the simplified biped system is summarized in Table 5.7. A more detailed description of the programming and control electronics used to implement this strategy on the experimental system may be found in [45].

Table 5.8: Summarized Experimental Results

Experiment	h_0 (m)	h_d (m)	n	$\bar{\epsilon}_h$ (m)	$\dot{\theta}_{7,mean}$ ($\frac{\text{rad}}{\text{s}}$)	Summary
1	0.549	0.550	10	0.010	0.079	stable
2	0.557	0.560	10	0.006	0.093	stable
3	0.567	0.570	10	0.008	0.086	stable
4	0.549	0.550	20	0.012	0.082	stable

5.3.3 Experimental Results

The modified control strategy was implemented on the experimental hardware to produce varied jump sequences in a manner similar to the simulation sequences discussed above. The constraint placed on the torso by the inclusion of the mechanical pin eliminates the possibility of the robot falling forward or backward. The performance of the experimental system is therefore evaluated based upon the ability of the controller to maintain the desired height h_d of the jump and keep the average forward angular rate $\dot{\theta}_{7,ave}$ within the range of the fuzzy controller.

The jump sequences performed by the experimental system are detailed in Table 5.8. The table includes the starting and desired height of the system h_0 and h_d, the number of jumps performed n, the mean height error $\bar{\epsilon}_h$, the mean value of $\dot{\theta}_{7,ave}$ noted as $\dot{\theta}_{7,mean}$, and a brief description. The robot was dropped from the starting height with an approximate initial forward angular rate of 0.00 $\frac{\text{rad}}{\text{s}}$. The mean height error is

$$\bar{\epsilon}_h = \frac{\sum_{i=1}^{n} |h_d - h_i|}{n}, \tag{5.3}$$

where n is the number of jumps performed and h_i is the height achieved by the i^{th} jump. The mean value of $\dot{\theta}_{7,ave}$ is

$$\dot{\theta}_{7,mean} = \frac{\sum_{i=1}^{n} \dot{\theta}_{7,i}}{n}, \tag{5.4}$$

where $\dot{\theta}_{7,i}$ is the value of $\dot{\theta}_{7,ave}$ for the i^{th} jump. The experimental data used for these calculations can be found in Appendix C.

5.3.4 Discussion of Experimental Results

The experimental results produced using the biped system KURMET for the simplified case can be generalized in much the same way as the simulation results. The varied jump sequences described in Table 5.8 exhibit common characteristics, including the behavior of the links and motors during the state cycle, the torso height h through the jump cycle, and the forward angular rate $\dot{\theta}_7$. Theses characteristics will be examined for a cycle of the state machine taken from the third jump in experiment 2 of Table 5.8.

The leg joint and motor positions are shown in Fig. 5.5 for the jump cycle of interest. The joint angles follow the general trend of the relevant motor positions during the **position**, **hold**, and **retract** motor primitives. The joint positions do, however, oscillate about the desired position. The oscillations can be seen to begin when the motor and link positions cross after the large actuator deflections exhibited during ground contact. These oscillations are an undesirable behavior of the unidirectional hardware in the actuators. The motor positions during the beginning of the **retract** primitive display unusual trajectories that deviate from the desired trajectories described by Eq. 3.17. The trajectories described by Eq.3.17 assume an initial motor velocity of zero, which can be seen in Fig. 5.5 to not be applicable. The

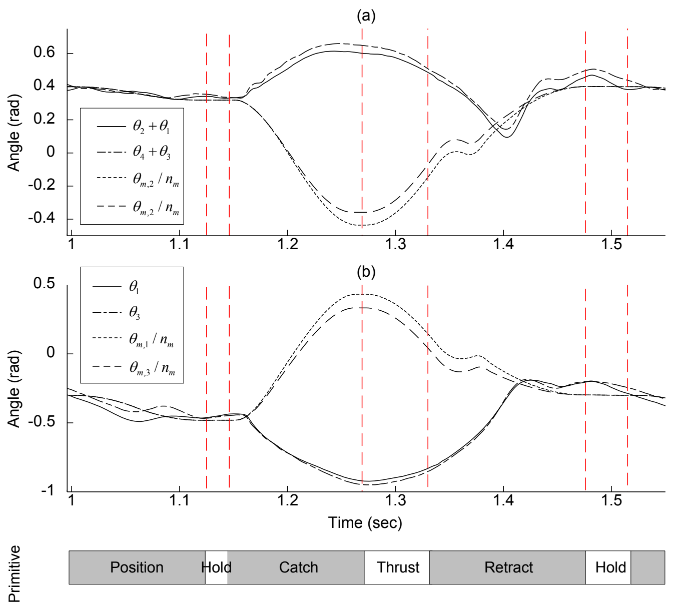

Figure 5.5: Experimental joint and motor positions for a single state machine cycle, third jump of experiment 2. (a) shows the shank links (relative to the torso) and motor positions while (b) shows the thigh links and motor positions. Motor primitives are indicated at the bottom of the figure with dashed red vertical lines marking the transitions.

non-trivial initial velocities of the motors at the beginning of the primitive cause the desired trajectories to not be tracked closely during the beginning of the primitive. The motor positions begin to follow the desired path during the deceleration phase of the trajectories, approximately one third of the way into the primitive period.

The torso height during the jump cycle is shown in Fig. 5.6. The figure demonstrates a relatively smooth behavior of the variable h throughout the state machine cycle, with minor inconsistencies due to impacts with the ground as well as the leg oscillations during the flight phase. The desired jump height h_d of this particular jump cycle is 0.56 m. The initial and final height noted in the figure both show a positive error from this value. The 50 jump cycles summarized in Table 5.8 and detailed in

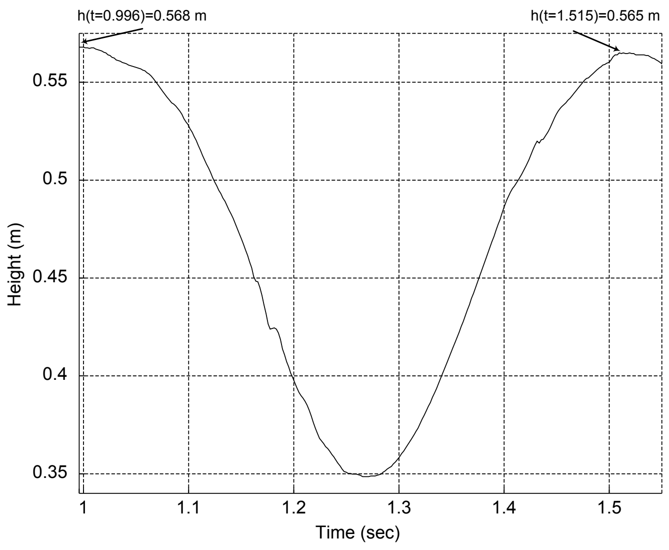

Figure 5.6: Experimental torso height h for a single state machine cycle, third jump of experiment 2. The initial and final height are indicated.

Appendix C include only 2 cycles that resulted in a final jump height less than the desired height. The system consistently overshoots the desired jump height. While this reflects on a discrepancy between the mathematical model of the system and the actual physical hardware, the resultant effect is beneficial in this particular instance. The jump sequences described in Table 5.8 all represent a desired height that falls within the bottom half of the membership functions for $\bar{x}_2$. The positive error on the final jump height results in the fuzzy input $\bar{x}_1$, the current jump height, falling within the range of the membership function centers for this input. The alternative, final jump heights that consistently fall short of the desired height, would result in saturation of the lowest membership function for $\bar{x}_1$, indicating that the fuzzy controller would be extrapolating for a region beyond the trained range instead of interpolating within the trained range.

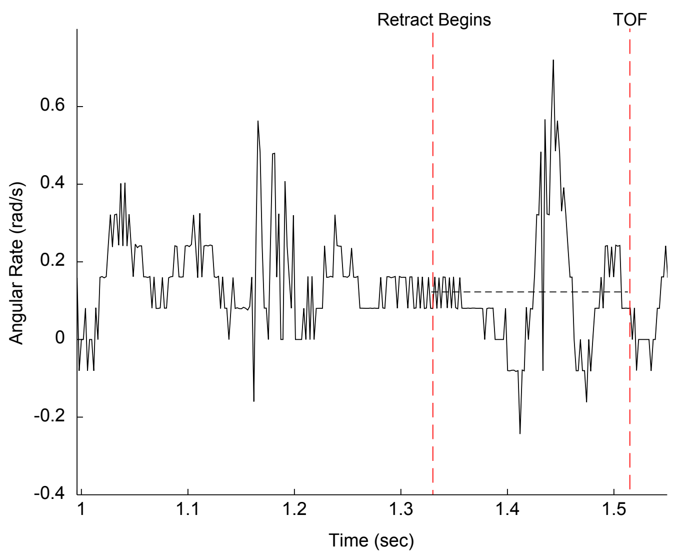

Figure 5.7: Experimental forward angular rate $\dot{\theta}_7$ (unfiltered) for a single state machine cycle, third jump of experiment 2. The dashed red vertical lines indicate the beginning and end of the averaging period for calculating $\dot{\theta}_{7,ave}$. The calculated value of $\dot{\theta}_{7,ave}$ is shown as a dashed black line across this period in the figure.

The final input to the fuzzy controller, the average forward angular rate $\dot{\theta}_{7,ave}$, is shown in Fig. 5.7 along with the unfiltered value of $\dot{\theta}_7$ for the state machine cycle. The calculated average values of $\dot{\theta}_7$ show a tendency to fall in the upper range of the input membership functions for $\bar{x}_3$. As a result, the experimental system moves in the backward (positive x_0) direction during the jump sequences. This behavior is partially the result of changes to the control strategy required to produce consistent sequential jumping with the experimental system, including the modified top-of-flight joint angles discussed above. Additionally, the tendency of the system to drift in the forward or backward direction during sequential jumps is a feature common to both the simulated and experimental results due to the exclusion of θ_7 in the control strategy.

The experimental results summarized in Table 5.8 demonstrate the application of the general control strategy described in this work to a simplified case of the bipedal system. The results show the ability to achieve normalized jump heights of approximately 2.3 in a stable, repetitive manner.

5.4 Experimental Difficulties

This chapter describes performance results of the control strategy described in this work for two distinct applications, the simulated five degree-of-freedom biped system and the simplified experimental biped KURMET. The restricted application of the control strategy to only the simplified bipedal system for the experimental biped reflects a number of issues discovered during work with the experimental system. Many of the issues contradict the desired behavior of the system described in Section 5.2.3 as necessary requirements for producing stability. These issues are discussed here, and possible solutions are provided to address the problems that currently prevent the implementation of the full control strategy to the five degree-of-freedom experimental biped.

5.4.1 Issues in Experimental Implementation

The issues discovered with implementing the full control strategy on the experimental system primarily consist of the limited ability to accurately sense variables critical to the stability of the controller and the non-ideal behavior of the actuators when in the direct-drive regime.

The sensors used to measure the variables of θ_5, θ_6, and θ_7 consist of optical shaft encoders. The encoders provide a high-resolution signal of the angular position of these variables. While this is sufficient to instill confidence in the accuracy of their

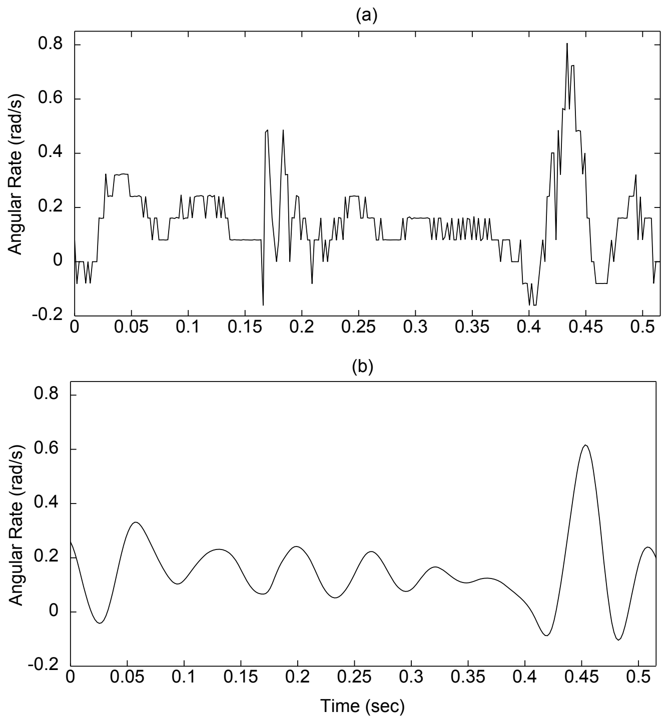

Figure 5.8: Sample of experimental forward angular rate $\dot{\theta}_7$ signal, unfiltered (a) and filtered (b) using a digital 4^{th} order Butterworth filter with a cutoff frequency of 25 Hz. The approximate delay with this filter is 20 milliseconds.

signals, the results require numerical differentiation to produce the time derivatives of the variables. The discrete nature of the sampled data signals from the sensors causes the numerical differentiation to produce noise in the resulting angular rates. Additionally, vibrations in the entire experimental system are excited by the impact forces with the ground as well as the rapid movements of the legs. The combined effect of the noise and vibration results in values for $\dot{\theta}_5$, $\dot{\theta}_6$, and $\dot{\theta}_7$ that exhibit large perturbations. A sample of the experimental data collected for $\dot{\theta}_7$ can be seen in Fig. 5.8 with no filtering applied and with a digital filter applied. As a result of these perturbations, the state of the system can be difficult to accurately identify.

Filtering of these signals can help to address the perturbed behavior as demonstrated in Fig. 5.8. The use of low-bandwidth digital filters on these signals introduces a delay on the order of tens of milliseconds. The high speed nature of the dynamic jumps described in this work require accurate sensing with minimal delay. In particular, the fuzzy controller produced for the full five-degree-of-freedom model has a small range of acceptable values for the torso angular rate $\dot{\theta}_5$. The presence of noise or vibrational perturbations in this signal would quickly lead to instability of the system.

The direct-drive regime of the actuators is modeled in this work using a linear spring and damper operating in parallel, as described by Eq. 2.10. The simulation results presented above for the full five-degree-of-freedom model were found to require a large assumed value for the contact stiffness, damping ratio, and preload torque in order to achieve stability. The observed behavior of the experimental system during jump cycles for the simplified case indicates that the contact model over-estimates the values of the contact stiffness and damping. The oscillatory behavior of the legs during the flight phase, shown in detail in Fig. 5.9 for a segment of a state machine cycle, reveals that the damping and stiffness do not approach the magnitude modeled in the numerical simulation. The tight coupling between the biped's legs and torso would result in this oscillatory behavior being transferred in part to the torso state if the mechanical pin locking the torso in place were removed. Oscillation of the torso state around the top-of-flight would introduce large problems into the selection of appropriate control parameters by the fuzzy controller. Additionally, a consistent top-of-flight leg configuration is required by the full control strategy to ensure that the system will behave as expected during the time between top-of-flight and ground contact.

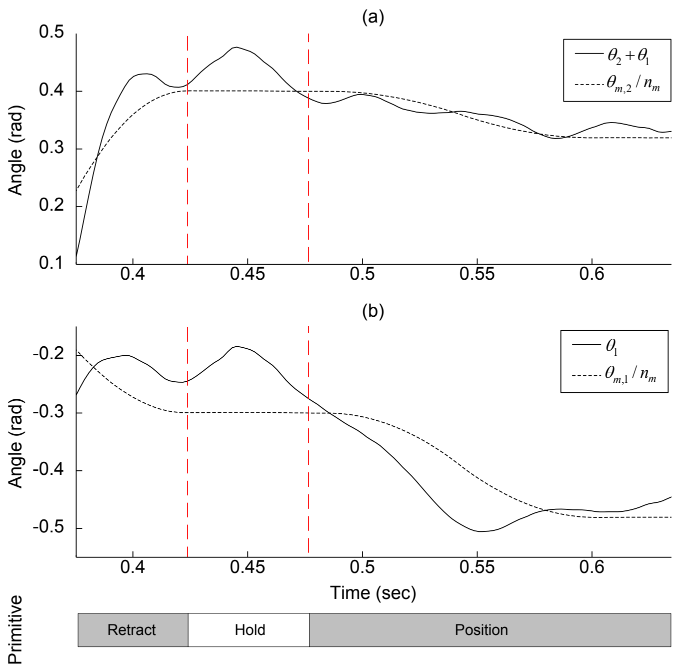

Figure 5.9: Leg oscillation during part of the flight phase for the right leg only. The oscillation of the shank link (shown relative to the torso) about the motor position can be seen in (a), while the thigh link and motor position can be seen in (b). The motor primitives in effect are noted at the bottom of the figure, with vertical dashed red lines used to indicate the transitions.

5.4.2 Suggested Solutions

The following items are suggested for addressing the oscillations observed in the leg joints during flight.

- The elastomer material used on KURMET to pad the contact points in the unidirectional actuator hardware was not selected based on performance characteristics. Alternative materials, including viscoelastic materials with large damping capabilities, could reduce the amplitude and settling time of the joint oscillations.

- Friction could be added to the leg joints themselves to increase the damping of the actuated joints, reducing their sensitivity to small torques. This would, however, negatively impact the height performance of the system due to energy loss to friction during the jump cycle.

- The spiral torsion springs used in the design of KURMET assume a nearly linear force-displacement curve. Modifying the spring design to produce a nonlinear force-displacement curve with a large initial stiffness could improve the position control of the links during the flight phase.

- The low-level feedback control laws used to achieve the desired motor positions during the **position**, **retract**, and **hold** primitives do not monitor the actual link positions. More advanced low-level controls utilizing feedback of the actual link positions could improve the system's ability to eliminate joint oscillations.

The difficulties encountered in accurately sensing the state of the robot with noise and vibrational perturbations present could be improved by the following.

- A model of the system dynamics could be used to develop a state estimator for the biped KURMET. Pat Wensing has made a preliminary investigation into the use of Kalman filters for the system, which could be extended and improved to provide an accurate and timely state estimator for the system state while avoiding noise in the sensor signals.

- The sensitivity of the control strategy to errors in the values of these variables could be reduced through modifications to the low-level controls, in particular the termination criteria for the motor primitives. The inclusion of the additional

hold motor primitive in the state machine for the simplified case is a good example of this, where it is used to improve the control's robustness to sensing error.

5.5 Summary

This chapter describes the results of the control strategy when implemented both in the numerical simulation and on a simplified case of the bipedal system KURMET. The ability of the control strategy to produce stable sequential jumps is examined for both implementations. The modifications to the control strategy required to produce jumping on the experimental system were discussed in detail. Issues preventing the implementation of the full control strategy on the experimental system were discussed and a number of possibles solutions were presented.

CHAPTER 6

SUMMARY AND FUTURE WORK

6.1 Summary

The impressive maneuvering abilities demonstrated by legged animals has largely remained outside the grasp of robotic systems. While a number of remarkable bipedal robot designs have emerged, the value of these systems remains limited until control strategies are developed that are capable of successfully producing fully dynamic maneuvers. The inherent difficulty of controlling these systems is rooted in the complexity of multi-DOF mechanisms with nonlinear dynamic relationships. In order for these robotic systems to be used to their maximum potential, control issues must be addressed and the performance limits pushed higher. This work developed a control strategy for producing vertical jumping in a particular biped robot, which represents a first step into the realm of further dynamic maneuvers. The control strategy is structured using a modular approach to allow for adaptation to additional maneuvers and robotic systems.

The control strategy development relied on a numerical simulation of the 5-DOF experimental biped KURMET. The simulation models the system dynamics, with emphasis placed on the modeling of the unique unidirectional series-elastic actuators used. The simulation model includes the mass and inertial effects of all elements of the system, the effects introduced by the electrical components, and the environmental interactions. It was intended to capture all the dynamic characteristics relevant to the performance of locomotive maneuvers. The simulation is performed in the **RobotBuilder** software application in conjunction with the **DynaMechs** dynamic engine, allowing the implementation of the simulation on a personal computer. Future control development for additional dynamic maneuvers can implement the modeling and simulation described in this work.

The control strategy was split into two functional levels to address the problems of planning and performing the jump maneuver. The low-level controller performs the motion control and uses a state machine to detect specific events and define transitions between the phases of the maneuver. The jump maneuver was defined using six phases, described as motor primitives and labeled as **position**, **catch**, **thrust**, **leg slow**, **retract**, and **hold**. The motor primitives are executed using open- and closed-loop control laws, which perform the joint motions necessary to achieve defined objectives. The use of motor primitives is based upon biological concepts of motor control theory that suggest concatenation of simple motions to achieve complex maneuvers. The high-level control is provided by a fuzzy controller, a class of intelligent control algorithms that allows inclusion of heuristic information in the controller. The fuzzy controller is called once during every cycle of the state machine, at the top-of-flight. The fuzzy controller examines the current state of the robot and then selects control parameters to be included in the low-level control laws to achieve the desired jump height at the next top-of-flight. The fuzzy control inputs include the torso angle and angular rate, which proved to be critical in determining appropriate control parameters. The additional fuzzy control inputs are the current height and the desired jump height. The desired jump height is defined as 0.575 m, which represents a normalized jump height of 2.3

The control parameters selected by the fuzzy controller were determined through the use of a training algorithm in the numerical simulation. The training algorithm implements the low-level control laws to drive the system through a jump maneuver. Upon completion of the jump, the training algorithm examines the performance and selects from a total of 12 failure modes to described the jump. These failure modes are used to select a training law that will modify the control parameters to produce a jump with performance closer to that desired. The jump maneuver is repeated iteratively until the performance is acceptable or the number of iterations exceeds a defined limit. The training algorithm was successfully able to determine control parameters for 607 of the 625 varied jump maneuvers defined in the fuzzy controller.

The control strategy, using the control parameters for the fuzzy controller selected by the training algorithm, was implemented on the simulated robotic system. In simulation, the control strategy demonstrated the ability to produce stable sequential jumping with an approximate height of 0.575 m. A simplified case of the experimental biped was also used to demonstrate the control strategy. The experimental biped KURMET was used to demonstrated stable sequential jumps with heights ranging from 0.55 to 0.57 m.

6.2 Future Work

The control strategy presented in this work was demonstrated to be capable of producing sequential vertical jumping in an experimental, planar biped robot. Many improvements could be made to various parts of the strategy and simulation to improve the performance of the control strategy. Additionally, the groundwork provided in this thesis could be extended to a number of projects in locomotion research to examine additional dynamic maneuvers. The following suggestions for future work focus on these subjects.

- The issues preventing implementation of the full control strategy on the experimental biped, as discussed in Chapter 5, should be examined in detail to investigate the future application of the full control strategy to the experimental system.

- The numerical simulation used to develop the control strategy has not been optimized for performance. Improvements to the source code structure for the control DLL, including conversion to an object-oriented programming approach, could improve the performance of the simulation and reduce the required time for computationally intensive algorithms such as the fuzzy training algorithm. Additionally, the use of super-computing facilities for performing time intensive simulations should be considered.

- The jumping produced in this work was limited to vertical jumps. The structure of the fuzzy controller should be modified to include two additional dimensions of control inputs to account for desired forward motion and desired height. The fuzzy training algorithm has demonstrated the ability to correct forward motion error and could be easily modified to train for a desired forward motion and account for a varied desired height in determining appropriate control parameters. Expanding the fuzzy controller in this way will increase the number of fuzzy rules exponentially. The expansion should therefore be performed in conjunction with the simulation optimizations discussed above.

- The jumps produced in this work do not represent the maximum height performance predicted for the experimental biped. The development of the control strategy indicated a trade-off between maximal performance and stability of the system, with the stability of the system given precedence in this work. The control strategy should be modified to produce maximal height jumps. This could be achieved in a manner that interfaces cohesively with the controller used to produce jumping in this work. An additional level of control above the high-level fuzzy controller could be used to select between strategies for achieving performance or stability, producing a system with greatly increased flexibility.

- The robustness of the controller described in this work is limited by the discrete nature of the supervisory fuzzy controller. A continuous controller could be added to the strategy to monitor the torso state throughout the maneuver and apply corrections to the control parameters to counteract disturbances. The heuristic knowledge for what control parameters to modify could be taken from the training algorithm. The robustness could be greatly improved by the ability to apply corrections between executions of the fuzzy controller.

- The linear contact model used to represent the joint hardstops and the unidirectional elements of the actuators should be examined in more detail. With the experimental hardware now available, a more accurate model of the contact pairs could be developed and experimentally verified.

- Additional dynamic maneuvers should be examined using this control strategy. In particular, maneuvers which are similar in nature to the simple jumping described in this work should be examined, including forward jumps, jumps starting from a standing or crouching position, and ultimately running jumps. A similar control strategy for running is currently being investigated by Yiping Liu using the principles described in this work. The results of the running strategy and other investigations should be verified using the experimental biped KURMET.

- Much of the complexity of the USEA design modeled in this work is intended to remove compliance from the actuators in situations where it is undesirable. An alternative approach using a transmission with variable compliance could offer considerable value, allowing low compliance for precise position control when desired and high compliance for energy storage. Actuator designs of this nature have been examined by Ghorbani and Wu [46] and continue to be a topic of research in robotic locomotion. This type of actuator design could be highly useful in investigating further dynamic maneuvers in legged systems where the objective of delivering explosive leg power currently conflicts to some degree with the need for precise position control.

- The difficulty in controlling the torso state at top-of-flight demonstrates the importance of the angular momentum of the system during flight. The possibility of influencing the system dynamics during flight through addition of passive or actuated links to the torso should be explored. Additional links, or perhaps actuated flywheels used to mimic their angular momentum, could improve the stability of the system.

- A widely accepted metric of stability for dynamic locomotion in robotic system remains undefined. The dynamic, repetitive jumps produced in this work should be examined in detail to determine what characteristics define the performance of a stable maneuver in quantifiable terms. These characteristics should be compared to applicable theories of stability criteria, including angular-momentum-based criteria such as that suggested by Goswami and Vinutha [47].

In conclusion, this work has developed a control strategy capable of producing vertical jumping in a planar biped robot with articulated legs. The dynamic nature of the jumping and the jump heights demonstrated in both simulation and hardware represent an improvement in the locomotive capabilities of robots currently available. Of greater interest to the field of legged robotics, this work offers a control approach that can be adapted for various robotic systems and maneuvers. The insight gained from the development process can serve to guide future control strategies that realize the maximum potential of robotic systems.

APPENDIX A

SYSTEM PARAMETERS: MASS, INERTIA, AND GEOMETRY

Table A.1: Mass Properties of Links

Link	Mass/Link	Number of Links	Total Mass
Boom	2.70 kg	1	2.70 kg
Torso	12.1 kg	1	12.1 kg
Thigh Link	0.81 kg	2	1.62 kg
Shank Link	0.63 kg	2	1.26 kg
Total Biped	-	-	14.98 kg

Table A.2: Center of Mass in Link Coordinate Frame, m

Link	x	y	z
Boom	0.00	-1.0	0.00
Torso	0.138	0.0008	0.0849
Right Thigh Link	0.0784	-0.0001	-0.0037
Left Thigh Link	0.0784	-0.0001	0.0015
Shank Link	0.0964	-0.0003	-0.0002

Table A.3: Inertia Tensor Format in Link Coordinate Frame, kg·m^2

I_{xx}	I_{xy}	I_{xz}
I_{yx}	I_{yy}	I_{yz}
I_{zx}	I_{zy}	I_{zz}

Table A.4: Torso Inertia Tensor, kg·m^2

0.218	1.76×10^{-3}	0.159
1.76×10^{-3}	0.470	4.28×10^{-3}
0.159	4.28×10^{-3}	0.334

Table A.5: Thigh Inertia Tensor, kg·m^2

3.90×10^{-4}	-1.60×10^{-5}	-2.10×10^{4}
-1.60×10^{-5}	1.27×10^{-6}	0.00
-2.10×10^{4}	0.00	1.27×10^{-6}

Table A.6: Shank Inertia Tensor, kg·m^2

2.38×10^{-4}	-2.0×10^{-6}	-1.00×10^{-6}
-2.0×10^{-6}	1.16×10^{-2}	0.00
-1.00×10^{-6}	0.00	1.16×10^{-2}

Table A.7: Boom Inertia Tensor, kg·m^2

3.60	0.00	0.00
0.00	4.10×10^{-3}	0.00
0.00	0.00	3.60

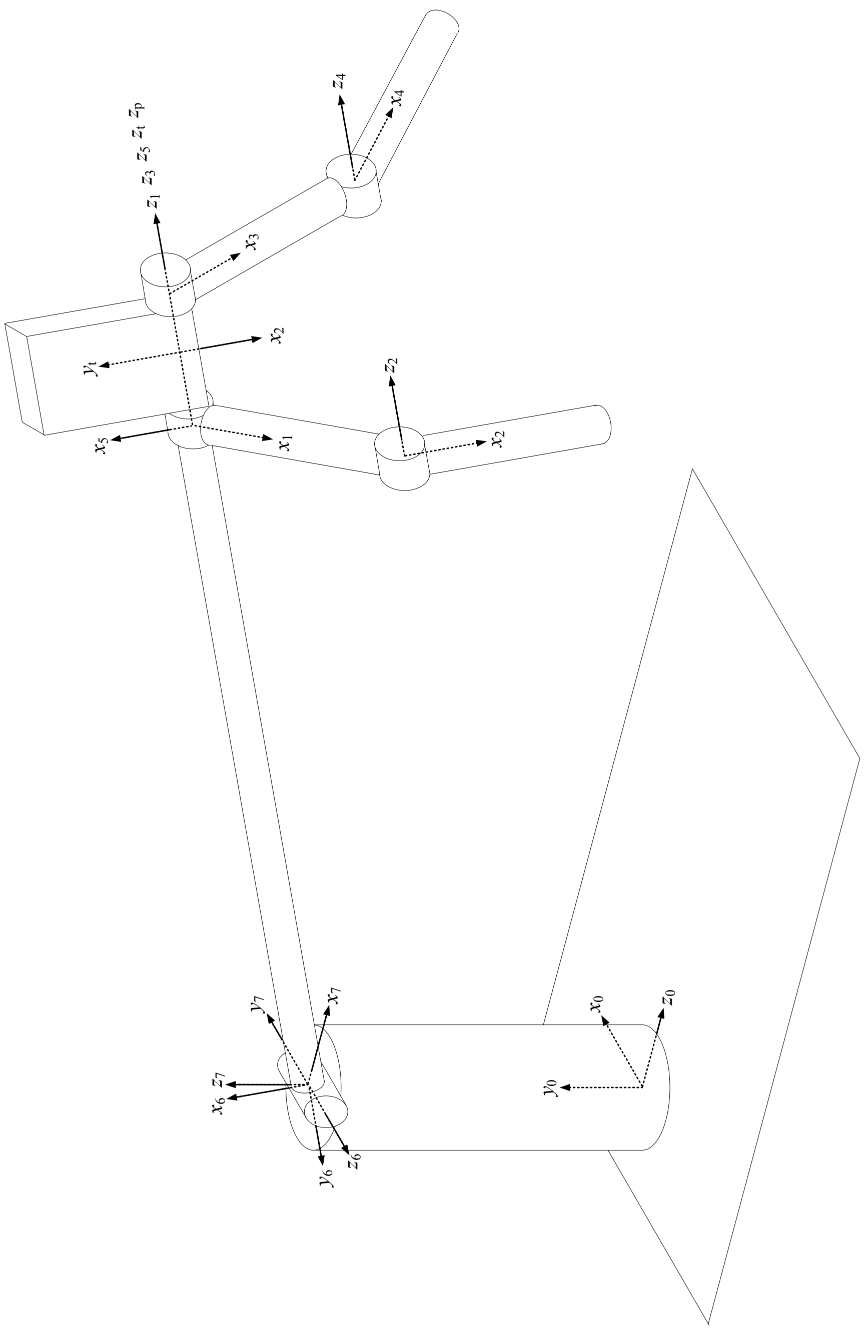

Figure A.1: Coordinate frame definitions for modeling. Not all axes are included for clarity. Remaining axes follow right-hand rule. Not to scale.

APPENDIX B

FUZZY TRAINING RESULTS

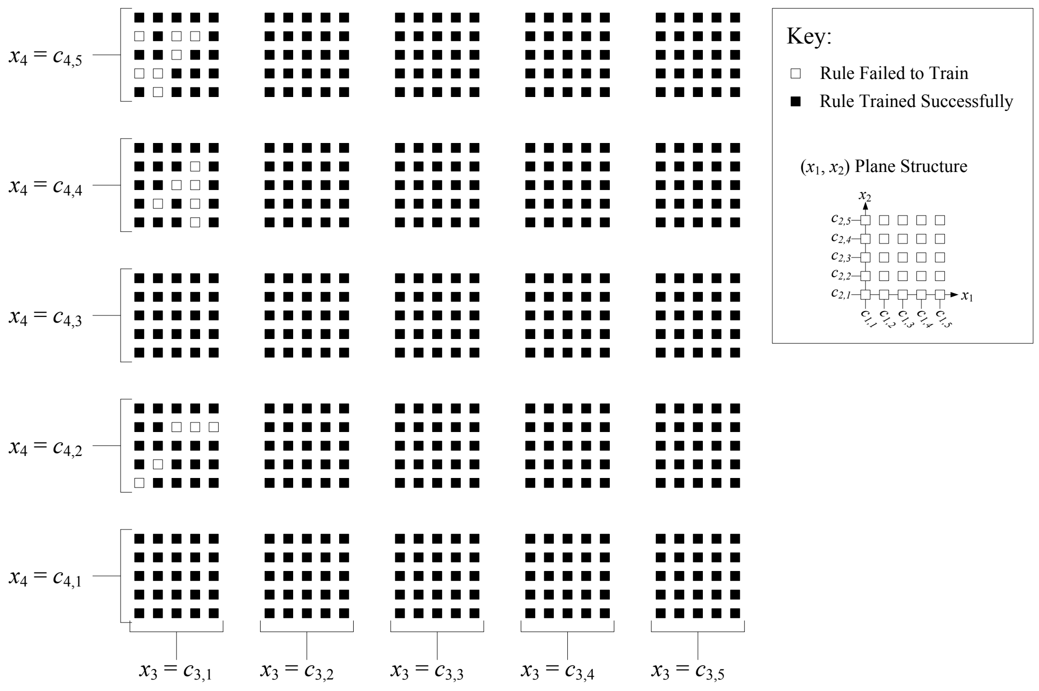

Figure B.1: Training algorithm results

APPENDIX C

EXPERIMENTAL DATA

Table C.1: Data for Experiment 1

Jump Number	$h_d\ (m)$	$h\ (m)$	$h - h_d\ (m)$	$\dot{\theta}_{7,ave}\ (\frac{rad}{s})$
1	0.55	0.542	-0.008	0.083
2	—	0.569	0.019	0.095
3	—	0.556	0.006	0.075
4	—	0.562	0.012	0.083
5	—	0.566	0.016	0.076
6	—	0.558	0.008	0.067
7	—	0.557	0.007	0.081
8	—	0.559	0.009	0.076
9	—	0.556	0.006	0.089
10	—	0.559	0.009	0.068

Table C.2: Data for Experiment 2

Jump Number	h_d (m)	h (m)	$h - h_d$ (m)	$\dot{\theta}_{7,ave}$ $(\frac{rad}{s})$
1	0.56	0.568	0.008	0.130
2	—	0.568	0.008	0.139
3	—	0.565	0.005	0.118
4	—	0.568	0.008	0.093
5	—	0.559	-0.001	0.040
6	—	0.573	0.013	0.117
7	—	0.564	0.004	0.064
8	—	0.566	0.006	0.067
9	—	0.564	0.004	0.065
10	—	0.569	0.009	0.099

Table C.3: Data for Experiment 3

Jump Number	h_d (m)	h (m)	$h - h_d$ (m)	$\dot{\theta}_{7,ave}$ $(\frac{rad}{s})$
1	0.57	0.571	0.001	0.104
2	—	0.584	0.014	0.090
3	—	0.575	0.005	0.093
4	—	0.579	0.009	0.091
5	—	0.576	0.006	0.055
6	—	0.573	0.003	0.095
7	—	0.581	0.011	0.061
8	—	0.573	0.003	0.108
9	—	0.582	0.012	0.075
10	—	0.582	0.012	0.090

Table C.4: Data for Experiment 4

Jump Number	h_d (m)	h (m)	$h - h_d$ (m)	$\dot{\theta}_{7,ave}$ $(\frac{rad}{s})$
1	0.55	0.570	0.020	0.069
2	—	0.556	0.006	0.094
3	—	0.562	0.012	0.087
4	—	0.554	0.004	0.086
5	—	0.563	0.013	0.080
6	—	0.561	0.011	0.085
7	—	0.563	0.013	0.064
8	—	0.566	0.016	0.101
9	—	0.561	0.011	0.058
10	—	0.561	0.011	0.096
11	—	0.565	0.015	0.074
12	—	0.558	0.008	0.056
13	—	0.567	0.017	0.088
14	—	0.564	0.014	0.096
15	—	0.558	0.008	0.063
16	—	0.567	0.017	0.101
17	—	0.557	0.007	0.078
18	—	0.562	0.012	0.087
19	—	0.560	0.010	0.082
20	—	0.556	0.006	0.092

BIBLIOGRAPHY

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

[2] I. Kato and H. Tsuiki. The hydraulically powered biped walking machine with a high carrying capacity. In *The Fourth International Symposium on External Control of Human Extremities on Advances in Robotic Kinematics*, pages 410–421, 1972.

[3] 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 vol.3, 2002.

[4] K. Nagasaka, Y. Kuroki, S. Suzuki, Y. Itoh, and J. Yamaguchi. Integrated motion control for walking, jumping and running on a small bipedal entertainment robot. *IEEE International Conference on Robotics and Automation*, 4:3189–3194 Vol.4, 26-May 1, 2004.

[5] M.H. Raibert. *Legged robots that balance*. MIT Press, 1986.

[6] R.J. Full and D.E. Koditschek. Templates and anchors: neuromechanical hypotheses of legged locomotion on land. *Journal of Experimental Biology*, 202(23):3325–3332, 1999.

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

[8] J.P. Schmiedeler, R. Siston, and K.J. Waldron. The signficance of leg mass in modeling quadrupedal running gaits. In G. Bianchi, J.C. Guinot, and C. Rzymkowski, editors, *ROMANSY 14: Theory and Practice of Robots and Manipulators*, pages 481–488. Springer, 2002.

[9] I. Poulakakis and J.W. Grizzle. Monopedal running control: SLIP embedding and virtual constraint controllers. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 323–330, 29 2007-Nov. 2 2007.

[10] 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.

[11] 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.

[12] B. Morris, E. R. Westervelt, C. Chevallereau, G. Buche, and J. W. Grizzle. Achieving bipedal running with RABBIT: Six steps toward infinity. *Fast Motions in Biomechanics and Robotics*, pages 277–297, 2006.

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

[14] W.T. Miller III. Real-time neural network control of a biped walking robot. *IEEE Control Systems Magazine*, 14(1):41–48, Feb 1994.

[15] 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.

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

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

[18] 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.

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

[20] V. Nunez, S. Drakunov, N. Nadjar-Gauthier, and J.C. Cadiou. Control strategy for planar vertical jump. *International Conference on Advanced Robotics*, pages 849–855, July 2005.

[21] R. McN. Alexander. Leg design and jumping techniques for humans, other vertebrates, and insects. *Philosophical Transactions: Biological Sciences*, (347):235–248, 1995.

[22] T. Hirano, T. Sueyoshi, and A. Kawamura. Development of ROCOS (robot control simulator)-jump of human-type biped robot by the adaptive impedance control. *International Workshop on Advanced Motion Control*, pages 606–611, 2000.

[23] K. Hosoda, T. Takuma, A. Nakamoto, and S. Hayashi. Biped robot design powered by antagonistic pneumatic actuators for multi-modal locomotion. *Robotics and Autonomous Systems*, 56(1):46–53, January 2008.

[24] 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.

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

[26] G. Colombo, M. Joerg, R. Schreier, and V. Dietz. Treadmill training of paraplegic patients using a robotic orthosis. *Journal of Rehabilitation Research and Development*, 37(6):693–700, 2000.

[27] S.K. Agrawal and A. Fattah. Theory and design of an orthotic device for full or partial gravity-balancing of a human leg during motion. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2):157–165, June 2004.

[28] H. Tsukagoshi, M. Sasaki, A. Kitagawa, and T. Tanaka. Jumping robot for rescue operation with excellent traverse ability. *International Conference on Advanced Robotics*, pages 841–848, July 2005.

[29] G. Nelson, K. Blankespoor, and M. Raibert. Walking BigDog: Insights and challenges from legged robotics. *Journal of Biomechanics*, 39:360, 2006.

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

[31] R.A. Bockbrader. Design of a five-link planar bipedal running mechanism. Master's thesis, The Ohio State University, Department of Mechanical Engineering, 2006.

[32] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. John Wiley and Sons, 2006.

[33] 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. ASME.

[34] S.J. Rodenbaugh and D.E. Orin. *RobotBuilder User's Guide*. The Ohio State University, Department of Electrical Engineering, 2003.

[35] 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.

[36] 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.

[37] 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, Nov 1999.

[38] D.K. Sung. Modeling and analysis of a nonlinear elastomer impact model with damping mechanism. *International Conference on Industrial Electronics, Control and Instrumentation*, pages 609–612 vol.1, Oct-1 Nov 1991.

[39] 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.

[40] S. Schaal. Is imitiation learning the route to humanoid robots? *Trends in Cognitive Sciences*, (6):233–242, 1999.

[41] *The Human Motion Show, DVD vol. 1.* Rhino House, 2005.

[42] A. Fod, M.J. Matarić, and O.C. Jenkins. Automated derivation of primitives for movement classification. *Autonomous Robots*, 12(1):39–54, 2002.

[43] J.S. Arora. *Introduction to Optimum Design.* Elsevier Academic Press, 2nd edition, 2004.

[44] R.C. O'Reilly. Six principles for biologically based computational models of cortical cognition. *Trends in Cognitive Sciences*, 2:455–462(8), Nov 1998.

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

[46] R. Ghorbani and Qiong Wu. Closed loop control of an intentionally adjustable compliant actuator. *American Control Conference, 2006*, pages 6:3235–3240, 2006.

[47] A. Goswami and V. Kallem. Rate of change of angular momentum and balance maintenance of biped robots. *International Conference on Robotics and Automation*, 4:3785–3790, 2004.