Biologically-Plausible Six-Legged Running: Control and Simulation

by

Matthew David Malchano

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2003

Author . . . .
Department of Electrical Engineering and Computer Science
February 4, 2003

Certified by . . . .
Hugh M. Herr
Instructor, MIT/Harvard Division of Health, Science, and Technology
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Biologically-Plausible Six-Legged Running: Control and Simulation

by

Matthew David Malchano

Submitted to the Department of Electrical Engineering and Computer Science
on February 4, 2003, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents a controller which produces a stable, dynamic 1.4 meter per second run in a simulated twelve degree of freedom six-legged robot. The algorithm is relatively simple; it consists of only a few hand-tuned feedback loops and is defined by a total of 13 parameters. The control utilizes no vestibular-type inputs to actively control orientation. Evidence from perturbation, robustness, motion analysis, and parameter sensitivity tests indicate a high degree of stability in the simulated gait. The control approach generates a run with an aerial phase, utilizes force information to signal aerial phase leg retraction, has a forward running velocity determined by a single parameter, and couples stance and swing legs using angular momentum information. Both the hypotheses behind the control and the resulting gait are argued to be plausible models of biological locomotion.

Thesis Supervisor: Hugh M. Herr
Title: Instructor, MIT/Harvard Division of Health, Science, and Technology

Acknowledgments

Hugh Herr, Gill Pratt, and Max Berniker, Joaquin Blaya, Bruce Deffenbaugh, Pete Dilworth, Jon Gonda, Andreas Hofmann, Frank Kirchner, Marko Popovic, and Rebecca Schulman:
Thanks much, y'all.

Mom, Dad, and Zach Malchano:
My love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The difficulties of control in legged locomotion are well documented [28]: (1) the high dimensionality and nonlinearity of the dynamics; (2) the relatively short duration of and limited control provided by the ground reaction forces; (3) the complexity of environmental conditions; (4) mechanical limitations in power, bandwidth, and energy efficiency; and (5) operation in a gravitational field. Yet the promise of legged locomotion is obvious in nature's ability to efficiently walk, run, trot, gallop, leap, and bound over most of the Earth's terrain. The lack of machines capable of reproducing this range of biological behaviors is evidence of the numerous challenges still faced by researchers.

Many of the controllers for six-legged (or hexapod) robots exhibit a high level of computational and conceptual complexity. Inverse kinematics, inverse dynamics, and complex models are common features. The systems are defined by large sets of high-precision parameters. The use of computational search methods, nonlinear optimizations, and machine learning techniques is often necessary. The usual focus is the production of a robust walking gait. Almost all of the resulting "hexapod literature has [up until 2001] concerned static or quasi-static walking machines."[2]

Recent successes in legged locomotion have been reported using control techniques labeled "feedforward", "open-loop", or "preflex" [34, 8, 33]. In these techniques, motions are coordinated by a central oscillator or clock. The oscillator triggers the play-back of trajectories at each joint. Often the control targets a highly simplified morphology with only one actuated degree of freedom (dof) per leg. The oscillator frequency, trajectory, and physical parameters are searched computationally or empirically until a satisfactory form of locomotion emerges.

The algorithm presented in this thesis diverges from both the static walkers and the "feedforward" approaches. In contrast to complicated control techniques used in the walkers, the controller studied consists of only a few hand-tuned position and velocity control loops and a two-state finite state machine. These loops servo joints to simple positions or velocities. The hexapod model runs dynamically.

The controller resembles the "feedforward" methodology in its relative simplicity and its stabilization without the use of vestibular-type information, such as a gyroscope signal. A number of important distinctions separate this work from the "feedforward" class. The morphology, with two actuated degrees of freedom per leg, lies between both the 3-dof insect leg design common in statically-stable walkers and the 1-dof legs found in the "feedfoward" morphologies. The control is not a purely clock-driven system, but rather it predicts approximate ground contact time using force information in order to signal leg retraction. A single parameter value selects from a range of forward velocities. Angular momentum information is used to couple stance and swing legs motions. The controller uses the feedback of a few high-level state variables and on-line calculation of dynamic quantities in generating leg trajectories. Finally, where "functional biomimesis" of cockroach locomotion inspires many "feedforward" algorithms, the basis for this controller is instead a model of horse trotting [15].

The organization of presentation here is as follows:

1. A short introduction of running locomotion, description of the horse trotting model, and survey of hexapedal control is made in Chapter 2.

2. The main goal of this work is the design of a novel hexapedal running control algorithm. It is described in depth in Chapter 3

3. The resulting motions of the gait produced in simulation, the evaluation of stability through perturbation, sensitivity, and robustness testing, and an argument for biological plausibility may be found in Chapter 4

4. Finally, a summary of the main conclusions and future directions of this thesis occurs in Chapter 5

5. I designed three mechanical leg segments to further physical testing, and they are documented in Appendix A. The controller source code is listed in Appendix B.

Chapter 2

Background Material

2.1 A Short Introduction to Running

A run is a fast locomotory mode characterized by specific motions of the center of mass. In particular, these motions feature significantly in-phase sinusoidal kinetic and potential energies. The simplest model of running which exhibits these energy profiles is the monopod. It consists of a single massless linear stiffness leg and a point-mass body. It has been shown that many animals' legs approximate such linear stiffnesses during running gaits [5]. Additionally, many runs feature an aerial phase: a period during which no feet contact the ground [24]. These characteristics hold true for morphologies running with any number of legs [4, 10].

A common gait used by hexapods, both natural and artificial, is the alternating tripod gait [17]. This symmetric and alternating gait moves legs together in groups of three. The front and hind ipsilateral and middle contralateral legs combine to form a tripod. The legs of a given tripod alternate between stance phase (during which they contact the ground) and swing phase (during which they move forward through the air).

Gaits, including running gaits, may be divided into those that are *statically stable* and those that are *dynamically stable.* This terminology was introduced by [23]. The points of ground contact of the stance legs define a support polygon. In a statically stable gait, the center of mass always lies directly above this polygon. In a dynamically stable gait, on the other hand, center of mass projects outside of this polygon for some significant portion of the gait cycle. The gait then relies on its momentum to carry it through these periods of instability. Cockroaches, even though six-legged, move dynamically [12].

Given these above definitions, the hexapod presented in this thesis may be said to move in a dynamically-stable, alternating-tripod running gait.

2.2 Biological Inspiration

A biological model of horse trotting [15] provides the initial basis for the running controller presented in this work. This quadrupedal model has been shown to successfully predict biologically observed metrics across a range of masses [14], and has been further adapted to model horse galloping gaits [16].

Trotting is a gait where diagonal limb pairs alternately strike the ground. This pairing resembles the motions of a hexapedal alternating tripod gait, where limbs are also grouped diagonally. Trotting occurs as a middle speed gait between walking and galloping in a variety of quadrupedal animals. The central ideas posed in the trotting model are:

1. Both diagonal limbs in a pair begin retraction simultaneously.

2. A trotting animal estimates when its feet will first strike the ground and begins limb retraction at this time. Hence, retraction may occur before foot contact. This estimation is performed on-line using the previous stance phase pitch and force information. It is believed that beginning retraction before ground contact may help stabilize those gaits which may be modeled as monopods [35], such as trotting and hexapedal running.

3. Stance leg retraction attempts to match a tangential target speed.

4. Vertebrate limbs behave like linear stiffness springs during ground contact.

5. Hind hip joints provide motive force. Forward hip joints provide braking force.

6. No active control of body orientation is used.

The trotting study was conducted using sagittal planar models. These models possessed 4 radially compliant legs with rotary hip joints; back and neck joints with passive stiffnesses were included to replicate the horse morphology accurately. Stability was evaluated based on the number of gait cycles the simulation locomoted without significant change to pitch, aerial apex height, or forward velocity. Additionally, simulations were subjected to a ground impedance change while trotting to test disturbance rejection.

The above ideas provide both the inspiration and starting framework for this work. Specific documentation of divergences and modifications may be found in Chapter 3.

2.3 Hexapedal Robots

This section seeks to place this thesis' contribution in the context of other work in hexapedal control. The histories of the study of legged locomotion are long and broad, and more detailed accounts of the robotics [31], biomechanics [24], and recent advances [32] may be found elsewhere.

The literature documents numerous approaches specific to hexapedal control. These approaches have focused mainly on producing robust walking gaits in 3-dof per leg morphologies. The advanced concepts and computational methods are applied. Parameter search techniques have been used to reduce the complexity of the resulting controllers. For example, neural nets [3], genetic algorithms [22], and varieties of reinforcement learning [18] have been used to define parameters. The problem has been decomposed into a variety of subproblems, with restricted goals such as gait generation [39], fault tolerance [41], rough terrain navigation [40], and leg coordination [1]. Biological organisms have served as physical [38, 26] as well as behavioral models [11]. Many of these controllers are successful in their chosen objectives or introduce innovative approaches. Yet all of these hexapods, along with many others, locomote in only static or quasi-static regimes. Their focus includes neither fast running gaits nor simplicity.

The MIT Leg Lab has previously developed two six-legged walking simulations similar to the aforementioned systems. The first controller targets a cockroach-sized morphology and uses a technique called "virtual model control" [37]. In virtual model control, desired forces are prescribed to the body link. For example, a "virtual" linear spring, attached between the center of mass and a "virtual" fixed frame, might counteract gravity's force on the body. The leg inverse kinematics and the Jacobian are used to map these "virtual" forces to leg joint torques, which exert real forces against the ground to produce the desired motion. The second hexapod, a simulated automobile-sized walking vehicle, also relies on this technique [27]. Neither controller claims biological plausibility.

More recently, impressive successes have been had in a series of highly simplified approaches. These approaches are unified in their use of a clock to coordinate the playback trajectories on 1-dof actuated legs and their emergent passive stability. They are also unified by their "functional biomimesis" of cockroach locomotion. Below I discuss RHex [2, 34], Sprawlita [8, 6], and Wheg [30], three examples of this approach to control.

The autonomous robot RHex possesses six actively controlled degrees of freedom, one per leg. The actuation is provided by rotary hip joints whose axes lie perpendicular to the sagittal plane. Legs move in complete rotations and group together in alternating tripods. A central oscillator coordinates the hips, which use local feedback loops to follow one of two velocities: a "retraction" velocity for stance phase and a faster "protraction" velocity for swing phase. Buhler et al. report a forward gait, with a speed of one body length per second, emerges after physical tuning of the system mass, leg spring stiffness, target velocities, and loop gains.

Sprawlita is a pneumatically actuated, tethered robot. It's 1-dof legs are pneumatic cylinders attached to the body with passive stiffnesses. In Sprawlita, a central pattern generator again sends a periodic feedforward trajectory to the mechanical system. Cham et al. argue passive stiffnesses and pneumatic cylinders act as purely mechanical feedback (termed "preflexes") which stabilize the gait. Otherwise, no feedback occurs. The controller may also servo the angle of all hips, but does not do so during intra-stride operation. The feedforward trajectory is a simple "bang-bang", extend-or-retract command sent to all legs in a tripod. The frequency and duty cycle of this trajectory were determined empirically. Sprawlita has been shown to move at speeds of 2.5 body lengths per second.

Wheg uses a morphology similar to RHex but with three radial-stiffness spokes per leg instead of one. Additionally, it reduces leg drive to a single motor. Its controller appears similar to RHex, but targets only a single angular rate per leg and switches its legs to an in-phase gait when climbing obstacles. It's forward velocity is undocumented.

Since the initial work on both Sprawlita and RHex, additional research has begun adding high-level adaptive feedback to these systems. In RHex, body orientation and foot contact information has been used to adjust for ground slope conditions [20]. This information is also being used to research adding an aerial phase to RHex's gait. In the above, Komsuoglu et al. state that, "experimental observations suggest higher forward speeds can be achieved provided an aerial phase is introduced....". In Sprawlita, a foot contact signal was used to modify the frequency and duty cycle of the feedforward trajectory to aid slop ascent [7].

Both of these recent incremental successes, I believe, validate the direction of the work pursued in this thesis, which utilizes a different set of high-level feedback terms in a simple control system.

Chapter 3

Hexapedal Running Control Algorithm

3.1 Introduction

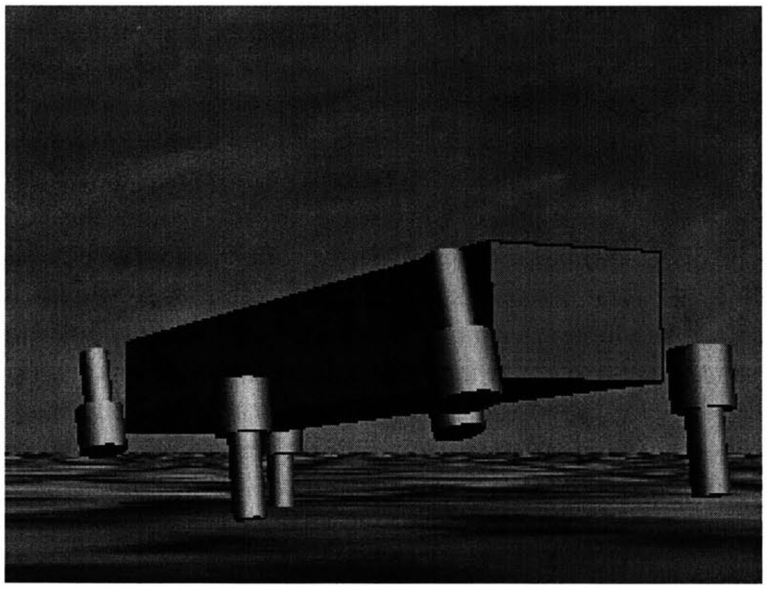


Figure 3-1: The hexapod model.

The control algorithm presented in this chapter generates leg motions in the above 12-dof model (Figure 3-1). These leg motions in turn produce a fast tripod running gait in simulation.

What are the salient features of this control? It is coordinated by a finite state machine. Foot contact switches signal FSM state changes. Proportional gain or proportional-derivative feedback loops mostly servo joints to constant positions or velocities. No active mechanism maintains body orientation; instead, level posture emerges passively. Leg retraction matches the distal leg end velocity to the target forward velocity. Angular momentum information about the stance leg is used to coordinate opposing swing leg motion. Leg retraction occurs based on an on-line prediction of ground contact time. The resulting gait possesses an aerial phase. The rest of this chapter gives precise description of these features.

Simulations of sagittal-planar and fully three-dimensional models were developed for a variety of masses. The results contained in this thesis focus only on the sagittal-planar model. The hexapedal sprawled posture causes most of the interesting dynamics to occur in the sagittal-plane. Reduction of orientation state to only pitch angle and derivatives allows a clear examination of the degree of passive orientation stability. For the three-dimensional simulations, only small changes to leg sprawl and increases in hip gains were necessary to achieve stable locomotion.

Section 3.2 describes the model used in simulation. The control algorithm is discussed in detail in Section 3.3. As previously mentioned, this work draws inspiration from a model of quadrupedal trotting. Documentation of the specific differences between these systems, along with various notes on simulation development, may be found in Section 3.4.

Results analyzing the gait produced by this control follow in Chapter 4. The complete source code for the controller, along with some supporting simulation code, may be found in Appendix B.

3.2 Morphology

The hexapod model possesses 12 actuated degrees of freedom. Each leg has a rotary hip whose axis lies perpendicular to the sagittal plane and a prismatic knee oriented radially from that hip. The hip joints attach the thigh link to the body, with fore and hind hips located at the body's corners and at the vertical level of the body center of mass. Each hip joint acts to protract (swing forward) and retract (swing hindward) a given leg in the sagittal plane. The knee joints attach the thigh to the shin and act to lengthen or shorten the leg. Additionally, during stance phase, these knee joints behave as linear, high stiffness leg springs. Figure 3-2 diagrams these actuated joint motions in the sagittal plane.

An un-actuated joint, co-located with the hip, splays the thighs outward within the frontal plane. A passive 40 N/m stiffness and 20 N-s/m viscosity hold these legs at an angle of 0.1 radians from vertical. The splay and stiffness maintain the necessarily sprawled posture in the 3D simulations.

The hexapod inherits this general form from the quadrupedal trotting model. Similar morphologies also appear in other robots and biomechanical models, such as [31, 25, 36], in part due to common recognition of a tested hypothesis suggesting that the legs of many animals behave like linear-stiffness springs [5].

The physical parameters defining the hexapod are summarized in Table 3.1. The total mass, its distribution, body dimensions, and leg lengths roughly approximate the latest iteration of the Scorpion robot [19]. I chose to model these values on the Scorpion because of prospective plans to conduct physical tests using some of its

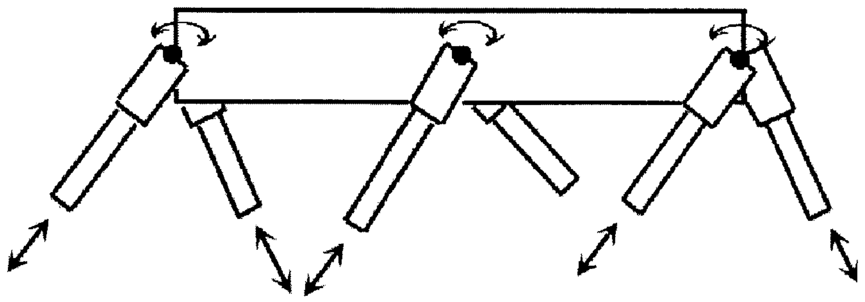


Figure 3-2: Diagram of hexapod model joint configuration, view from sagittal plane

hardware. One consequence of this choice, the legs total nearly 50% of the system mass and hence contribute significantly to the system dynamics.

Parameter	**Symbol**	**Value**	**Unit**
Body mass	–	3.104	kg
Thigh mass	M_{thigh}	0.464	kg
Shin mass	M_{shin}	0.029	kg
Total mass		6.062	kg
Body length	–	0.64	m
Body height	–	0.10	m
Body width	–	0.145	m
Thigh length	L_{thigh}	0.076	m
Thigh radius	–	0.15	m
Shin length	L_{shin}	0.176	m
Shin radius	–	0.024	m
Max Leg Length	–	0.252	m
Min Leg Length	–	0.052	m
Max Hip Angle	–	$\pi/2$	rad
Min Hip Angle	–	$-\pi/2$	rad

Table 3.1: Morphological parameters defining the simulated model.

3.3 Running Control Algorithm

This section discusses the control algorithm which generates the hexapod's running gait. A centralized controller coordinates the motions of the legs. This controller is in one of two states during steady state locomotion: *aerial phase* (Section 3.3.1) or *stance phase* (Section 3.3.2). Each state performs a separate function. In the aerial phase, the hexapod moves in a ballistic trajectory through the air. The controller positions the legs to prepare for ground contact. During the stance state, the hexapod contacts the ground, and the controller moves the legs in order to propel the body forward and upward into the air.

In both states, the legs are grouped into two tripods of three legs each. Each leg in a tripod is commanded similarly. At any time, one of the tripods is designated the *stance tripod* and the other the *swing tripod.* During stance phase, the legs of the stance tripod are in contact with the ground, while the legs of the swing tripod do not touch the ground. When control switches from the stance phase to the aerial phase, the designations for the tripods are swapped. The swing tripod becomes the stance tripod and vice versa. Hence, in aerial phase, the tripod designated as the stance tripod will contact the ground in the subsequent stance phase.

3.3.1 Aerial Phase Control

Aerial phase control positions the legs to prepare the hexapod to enter into stance phase control. Aerial phase control begins when the hexapod's leg leave the ground and ends when ground contact is predicted to occur again. During aerial phase control, the legs are positioned to a ready position for the next stance phase's initiation. Figure 3-3 shows the aerial phase controller for both knee and hip joints in block diagram notation. Table 3.2 lists the numerical values of the controller gains and target positions.

Parameter	**Value**	**Unit**	**Description**
$K_{aerial\ knee}$	2.0	N/m	Aerial knee stiffness
$B_{aerial\ knee}$	1.086	Ns/m	Aerial knee damping
$K_{aerial\ hip}$	40	N/rad	Aerial hip stiffness
$B_{aerial\ hip}$	20	Ns/rad	Aerial hip damping
$L_{knee\ extend}$	0.0	m	knee target compression
$L_{knee\ retract}$	0.2	m	knee target compression
$\phi_{hip\ protract}$	-0.4	rad	hip target position
$\phi_{hip\ retract}$	0.4	rad	hip target position

Table 3.2: Aerial phase control parameters

At the beginning of the aerial phase, the swing tripod starts in a retracted position and with its knees fully extended. The stance tripod starts protracted and with its knees fully compressed. Proportional-derivative position control loops at the hips apply torques to both tripods to brake the inertia from the previous stance phase's motions and to roughly position the tripods for entry into the next stance phase. The target angle for the stance tripod protraction was estimated by considering the return path of a spring-mass monopod dropped from the predicted average height of the hexapod center of mass apex and with the predicted velocity of the hexapod robot. The angle of attack of this monopod was used as the target protraction angle. This value was then hand-tuned until a steady state run in the hexapod model was achieved. The same angle from vertical is used for the swing tripod target retracted position.

Hip loop gains were chosen to assure that legs were brought to a halt before they contacted the hip joint stops. Although it was intended that loop gains would be high enough to assure precise positioning to the desired target positions before the end of aerial phase, this behavior was not observed in the final simulation. Instead, some variation in protracted leg position was observed. Hip damping coefficients were chosen to critically damp the motion of the hips, assuring that no oscillation or overshoot occurs during leg motion.

When aerial phase is entered, the controller estimates the time of the next ground contact, which is when stance phase should start. This estimation is accomplished using the vertical velocity of the center of mass of the hexapod at the moment aerial phase begins. From this, the duration of a parabolic flight path during aerial phase is predicted. A timer maintains the elapsed time from the onset of aerial phase control; when the timer exceeds the predicted duration, aerial phase ends and stance phase begins. The controller also switches into stance phase if a leg contacts the ground at any time during aerial phase motion.

Just before aerial phase control ends, P-D controllers apply forces to the knees to extend the stance tripod to its maximal length and compress the swing tripod by 0.15 meters. Knee gains and damping coefficients were chosen to assure this motion completes before ground contact.

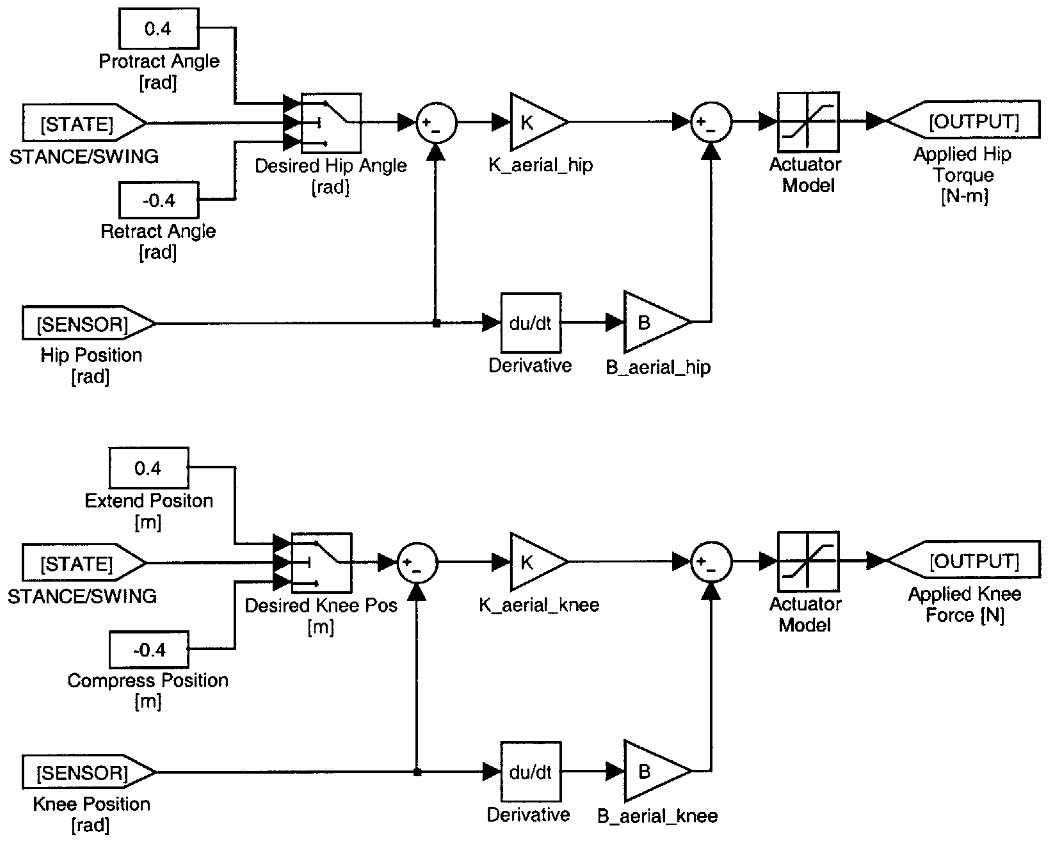


Figure 3-3: Block diagram of aerial phase control

3.3.2 Stance Phase Control

Stance phase control determines the motions of the tripods during ground contact. The stance tripod usually contacts the ground shortly after the onset of stance phase control, and the control ends when that tripod again leaves the ground. During this control, the stance tripod moves from a protracted position to a retracted position and acts against the ground. Simultaneously, the swing tripod moves from a retracted position to a protracted position without contacting the ground. Figure 3-4 shows a control block diagram depiction of the control used during stance phase. Table 3.3 lists the gains and desired velocities used in the controller.

Parameter	**Value**	**Unit**	**Description**
G_{vel}	6.0	N-sec/rad	stance tripod loop gain
G_{vel2}	-0.2	N-sec/rad	swing tripod loop gain
$\dot{x}_{max}$	1.5	m/sec	target forward velocity
$l_{compress}$	0.22	m	swing leg compressed length
Fore/Hind k_{leg}	750	N/m	stance leg stiffness
Middle k_{leg}	1500	N/m	stance leg stiffness
b_{leg}	20	N-sec/m	stance leg damping

Table 3.3: Stance controller parameters

Stance phase control begins when either the predicted duration of the aerial phase expires or when one or more legs contact the ground. In normal steady-state behavior, stance phase initiates slightly before ground contact. This preliminary acceleration results in "leg retraction", which was one of the hypotheses posed in the quadruped model.

At the initiation of stance phase control, knees begin to act as high linear stiffnesses. The front and hind stance legs are set to a 750 N/m stiffness with 20 N-sec/m damping coefficient. The middle leg switches to a 1500 N/m stiffness; this stiffness is doubled to compensate for the asymmetry of the tripod. Without this adjustment, asymmetrical ground reaction forces, a body-roll moment, and insufficient rear leg ground clearance occur during stance. Swing tripod knee stiffness also increase.

The hips of the stance tripod are controlled with proportional velocity loops. These loops attempt to match the distal end of a stance leg's tangential speed to the ground speed for speeds less than the desired forward velocity $\dot{x}_{max}$. The distal end of the leg has a velocity of $v = l_{leg} \cdot \dot{\phi}_{hip}$ where $l_{leg}(t)$ is the length of the leg and $\dot{\phi}_{hip}(t)$ is the angular rate of the hip joint. The torque $\tau_{hip}(t)$ applied is:

$$\tau_{hip}(t) = G_{vel} \cdot \left(\left(l_{leg}(t) \cdot \dot{\phi}_{hip}(t) \right) - \dot{x}_{max} \right) \tag{3.1}$$

The swing leg control operates as follows. Proprioception information provides the compression of the stance leg $\Delta l_{stance}(t)$, along with the velocity of the stance leg hip $\dot{\phi}_{stance}(t)$ and of laterally-opposed swing leg hip $\dot{\phi}_{swing}(t)$. The moment of inertia for the stance leg about the hip joint is calculated:

$$I_{stance} = \frac{1}{3}M_{thigh} \cdot \left(0.5 \cdot L_{thigh}\right)^2 + \frac{1}{3}M_{shin} \cdot \left(0.5L_{shin} + L_{thigh} - \Delta l_{stance}(t)\right)^2 \quad (3.2)$$

Hence, the angular momentum of the leg about the hip axis is

$$L_{hip}(t) = I_{leg}\left(l_{leg}(t)\right) \cdot \dot{\phi}_{hip}. \quad (3.3)$$

The target velocity $\dot{\varphi}_{swing}(t)$ of the stance leg is set to cancel this angular momentum.

$$\dot{\varphi}_{swing}(t) = \frac{I_{stance}(t)}{I_{swing}(t)} \cdot \dot{\phi}_{stance}(t) \quad (3.4)$$

A proportional velocity control loop with negative gain G_{vel2} servos the swing leg speed to follow $\dot{\varphi}_{swing}(t)$. The applied hip torque is

$$\tau_{hip}(t) = G_{vel2} \cdot \left(\dot{\varphi}_{swing}(t) + \dot{\phi}_{swing}(t)\right). \quad (3.5)$$

Since the swing leg length is always shorter than the stance leg length, $I_{swing}(t) < I_{stance}(t)$ and therefore $\dot{\varphi}_{swing}(t) > \dot{\phi}_{stance}(t)$. The swing leg will hence be in position to enter aerial phase. We suspect that angular momentum about the center of mass plays an important role in dynamically stable locomotion. Swing leg control annuls the pitch-axis angular momentum about the hip, but this is only the smaller of the two terms expressing the effects of a leg's contribution to the angular momentum in the body frame. Further research is necessary before these speculations can be confirmed.

The control outlined above for the hip and knee joints is maintained throughout stance phase. Stance phase ends when all feet leave the ground. At the transition, the controller starts the aerial phase timer, records the vertical velocity, and switches stance and swing tripod designations. It then enters aerial phase.

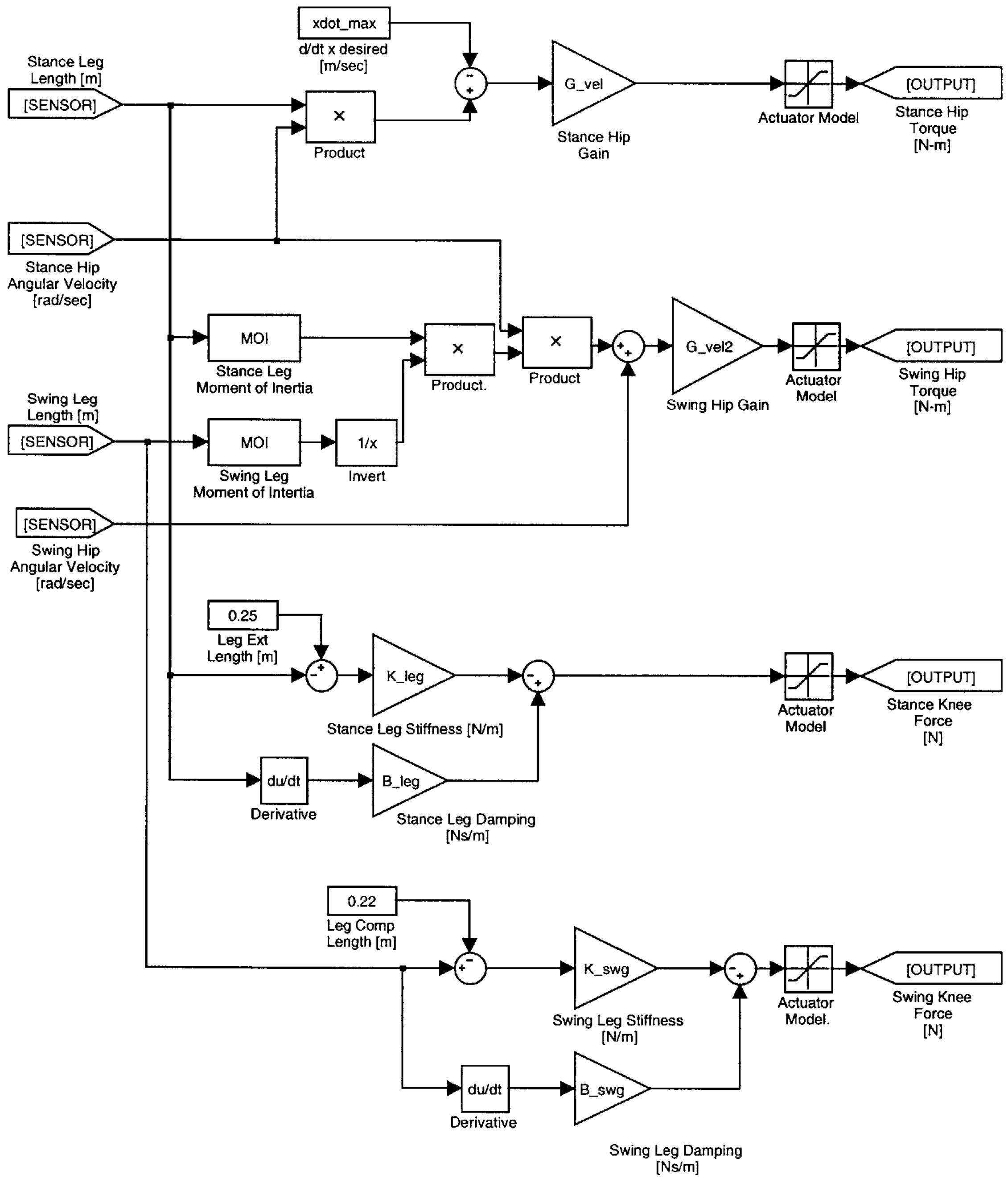


Figure 3-4: Block diagram of stance control

3.4 Details of Methodology

The control presented here uses the quadrupedal model as inspiration, but both morphology and control differ in a number of very significant ways. The hexapod morphology adds two legs, makes the body link rigid, and removes the head link. The mass, mass distribution, and dimensions have all been changed. Viscous damping has been added to the hexapod's stance leg springs to model mechanical losses.

Control differs from the quadruped as follows. Symmetric target velocities and stiffnesses are used for fore- and hindlimbs; in the quadruped these targets are significantly different. The aerial state occurs when all feet leave the ground; in the quadruped it occurs when any single foot loses contact. Only vertical take-off speed is measured to predict ground contact; the quadruped also uses pitch sensor information. The swing leg follows the opposing stance leg's angular momentum; the quadruped's swing leg "mirrors" the position of the stance leg instead. Parameter values differ widely. The quadrupedal model documents that twenty-eight unique velocities, stiffnesses, gains, and feedback targets define its control. The hexapod control is defined by thirteen such quantities.

Simulations were developed in Creature Library [21], an MIT Leg Lab extension to the commercial dynamics simulator SD-Fast [9]. A 4th-order Runge-Kutta integrator integrated the equations of motion generated by SD-Fast with a 200 nsec step size. The ground model was a linear spring and damper system with a stiffness of 400 N/m and a viscosity of 100 N-sec/m.

All simulation results reported relied on hand-tuned control parameters. At one point, I attempted to computationally generate parameters using naive nonlinear optimization. The optimization attempted to minimize the basic cost function

$$V = \sum_{t_o<t<t_f} (\dot{x}(t) - x_d)^2 + (z(t) - z_d)^2 + \theta(t)^2 \tag{3.6}$$

with V evaluated using the simulation. In the case of instability, out of range parameters, or other failure cases, V defaulted to its maximal value. The search space was over only five controller parameters. Computing an analytical gradient for the system appeared too difficult an undertaking, and hence Powell's method [29] was chosen as the nonlinear optimizer. It generates gradient information by sampling cost function values for changes in each parameter. It then line-minimizes along the direction which appears most promising. Initial parameter values were selected from a known-good starting point.

This approach was proved infeasible due to a number of stumbling blocks. The cost function failed to capture the desired behavior. The optimization seemed to become trapped in local minima. Large portions of the parameter space produce no useful optimization information due to the "brick wall" nonlinearities in the cost function for gait failures. The simulation runs far slower than realtime and an evaluation of a parameter set requires on the order of tens of seconds of simulated time. Hence optimization trials took a long time. Numerical optimization was eventually abandoned in favor of hand-tuning.

Chapter 4

Results

A mathematically rigorous, analytical evaluation of system stability is not tractable: the high dimensionality, nonlinearity, and dynamic complexity alone limit the ability to present such a treatment. Even if it were possible, the the value of the analyses obtained would be questionable. Yet, the existence of a stable gait in simulation provides thin proof unless its ability to handle the wide range of environmental conditions found in the real-world is demonstrated.

This chapter presents evidence that argues for the system's convergence to a steady-state periodic gait over a range of initial conditions, in the presence of disturbances, and with an insensitivity to changes in parameter values. These tests offer a compelling argument that the algorithm exhibits long-term stability and is physically realizable.

Simple metrics of the resulting gait (Table 4.1) allow performance comparison to other legged locomotion algorithms. Mean kinematic traces for joints and the center of mass motions describe periodic steady-state behavior (Section 4.1) . Error bars on these kinematic traces provide information as to the typical divergence seen from this mean motion. Torque and force perturbations quantify disturbance rejection (Section 4.2). Robustness testing samples the space of initial conditions to determine those which result in steady-state motion (Section 4.3). Parameter sensitivities measure the necessary precision of the values defining the controller (Section 4.4). Finally, the biological plausibility of the algorithm is discussed (Section 4.5).

Throughout this chapter, a gait is termed *stable* or *in a periodic steady state* if, over some stated period of time, the gait occurs without a gait failure. A gait failure is defined as body ground contact or joint torques exceeding simulation integration limits. The system is considered *stable* if it exhibits a stable gait. Additionally, when initial conditions of a simulation are not given, it may be safely assumed they are standard initial conditions: $\dot{x} = 0.7$ m/s; $z = 0.3$ m; $\theta = 0.0$ rad; $\dot{\theta} = 0.0$ rad/sec; and $t_{est\ contact} = 0.09$ sec.

Parameter	**Value**	**Units**
Mean Velocity	1.48 2.35	m/sec body lengths/sec
Stride Length	0.71	meters
Stride Frequency	2.08	Hz
Tripod Duty Cycle	30.2	%
Mean Mech Power	38.88	Watts
Peak Mech Power	481.0	Watts
Froude Number	0.957	-

Table 4.1: Summary of gait measurements.

4.1 Motion Analysis

The gait analysis plots below display the mean motions of the hexapod during steady-state locomotion over a single gait cycle. This gait cycle includes the stance phases of both tripods along with two aerial phases and lasts an average duration of 0.435 sec. The stance phase for one tripod begins at $t = 0.00$ sec and ends at $t = 0.13$ sec; the opposing tripod's stance phase begins at $t = 0.21$ sec and ends at $t = 0.34$ sec.

Figure 4-1 shows the motions of the center of mass along with the system's kinetic and potential energy and angular momentum in the center of mass frame.

The remaining plots give leg specific information for the tripod initially entering stance. Since the gait is nearly symmetric, only a single tripod's motions needs be displayed. Figure 4-2 shows the ground reaction forces for the front, middle, and rear legs of this tripod. These forces are split into vertical (z-direction) and horizontal (x-direction) components. Positive horizontal forces point in the direction of robot motion. Figure 4-3 shows the hip joint position, velocity, and force for the same legs. Positive values correspond to retraction. Figure 4-4 shows the knee joint position, velocity, and force. Figure 4-5 shows absolute mechanical power output for the individual joints. These power graphs do not account for regeneration through negative work. They also neglect possible passive mechanical implementations. For example, if the knee joint stiffness in stance phase were implemented with a passive compression spring, no net power output would occur. Previous power numbers do take passive knee implementation into account.

These plots were produced in a standard gait plot format. The data were collected over a 10 second simulation. They were then split into samples beginning at the instant of a specific tripod's first ground contact. These samples were resampled to a uniform length and filtered with a 10-point window, low-pass FIR filter. The mean across these samples was then taken and is plotted as the solid line. The two dashed lines delimit plus and minus one standard deviation from this mean. Some minor high-frequency noise is visible at the edges of a few of the plots. This noise is believed to be an artifact of the FIR filter; it should be stressed that it does not result from any system motion or dynamics.

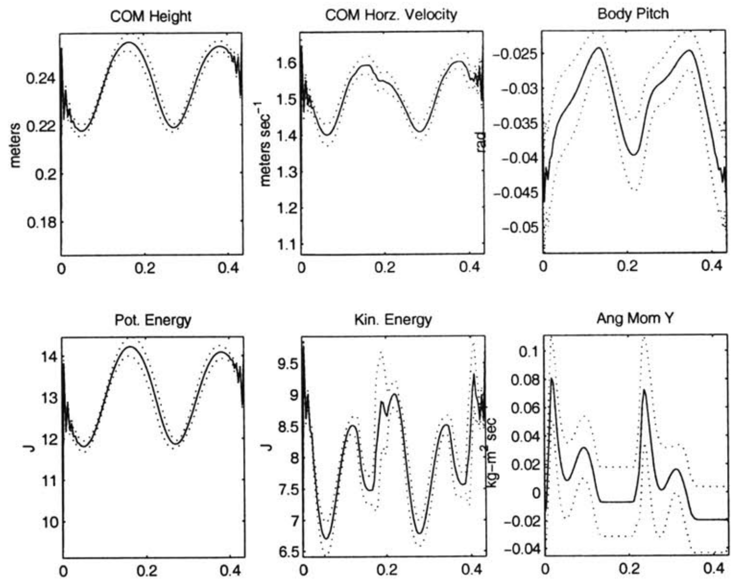


Figure 4-1: Mean kinematic motions of the center of mass, system potential and kinetic energies, and pitch-axis angular momentum.

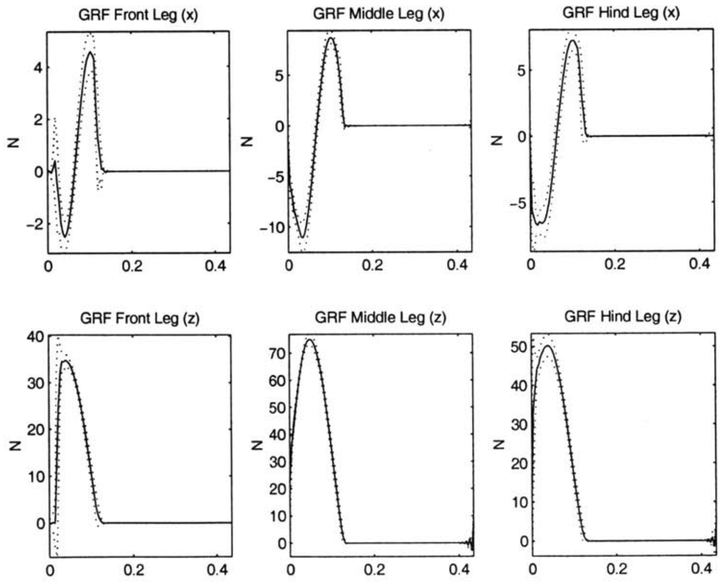


Figure 4-2: Mean ground reaction forces: horizontal and vertical components for front, middle, and rear legs of a tripod.

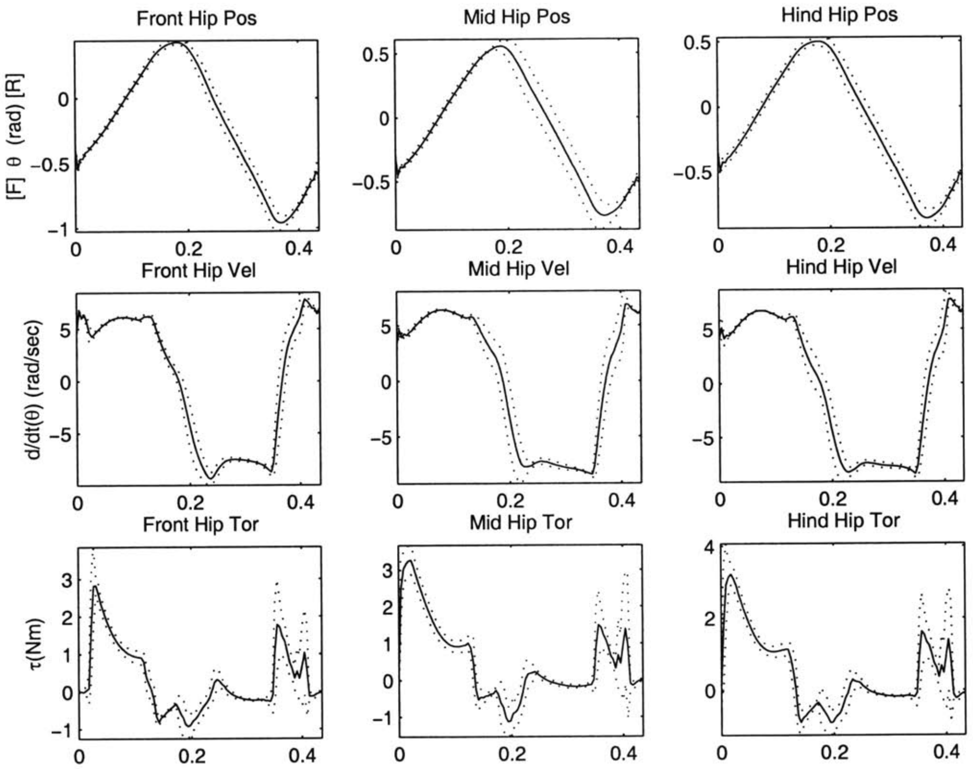


Figure 4-3: Mean hip joint kinematics: angle, angular rate, and torque.

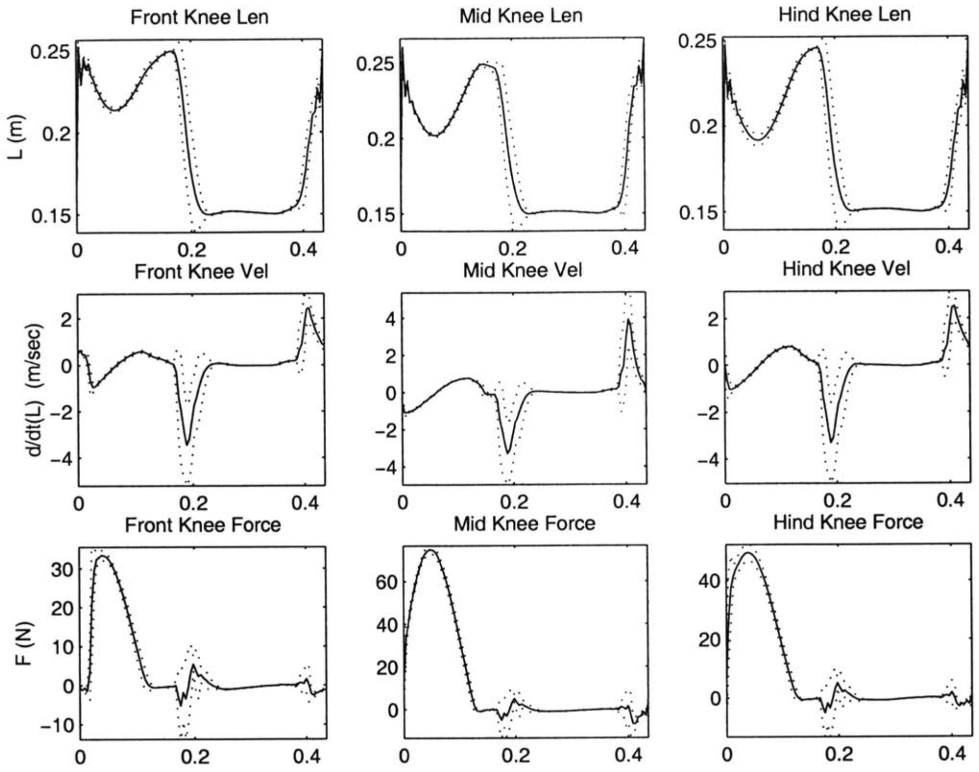


Figure 4-4: Mean knee joint kinematics: leg length, velocity, and applied force.

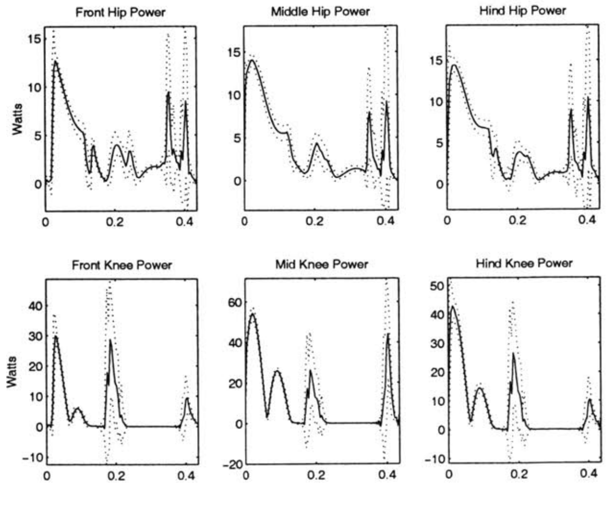


Figure 4-5: Mean mechanical joint power output.

4.2 Perturbation Testing

Perturbation testing measures the influence of environmental disturbances on the stability of a controlled system. In the case of the hexapod, torques and forces are applied to the center of mass during stable locomotion. The resulting linear and angular momentum response is examined. Impulse and step function perturbations are employed to measure the rejection of near-instantaneous and continuous changes to these momenta.

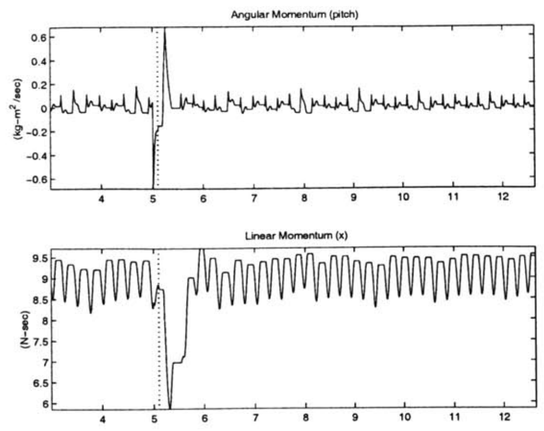
Many detrimental environmental conditions, to a first approximation, might be modeled as unforeseen, sudden changes to system momenta. For example, impacts with small obstacles, abrupt ground height variations, or foot contact slips alter the hexapod's linear and angular momenta. Uneven ground impedance or hip torque mismatch change the angular momentum. Results demonstrating the return of momenta to pre-perturbation levels, along with upper bounds on rejected disturbance magnitudes, provide evidence that stable gait survives environmental disruption.

Short duration (0.8 msec) torques were applied directly to the center of mass. These approximate a zero time duration impulse but, unlike a zero time duration force, cause a non-zero change in system angular momentum. Perturbations were applied at $t = 5.1$ sec, which corresponds to the instant of a stance phase initiation. Figure 4-6 shows the magnitudes of the pitch-axis angular momentum (top row) and x-axis linear momentum (bottom row) over two separate simulation runs. A dashed line delineates the instant of perturbation. In the left plot of this figure, a positive torque "impulse" of magnitude 75 N-m acts on the center of mass, producing a forward pitching motion. In the right plot, a negative torque of magnitude –100 N-m is applied to the center of mass again at $t = 5.1$ sec.

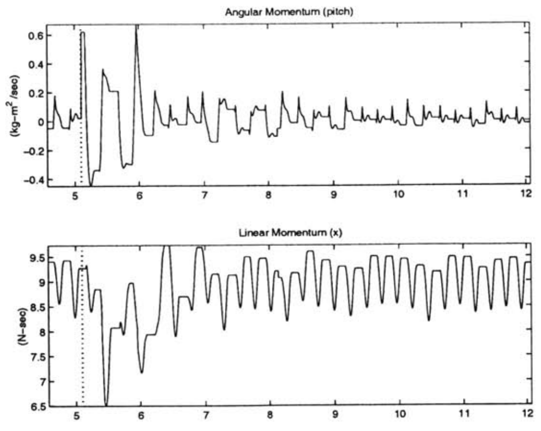
In both cases, the disturbance induces a sudden visible pitching of the hexapod which continues as an oscillation for a number of gait cycles. Eventually, this oscillation dies off and a stable gait continues. As the plots indicate, pre-perturbation angular momentum amplitude initially lie almost entirely within a –0.5 to 0.25 $kg-m^2/sec$ range. At the time of impulse, it peaks sharply to a magnitude of nearly 0.65 $kg-m^2/sec$. Shortly thereafter, it returns to its previous range.

Similar to the use of torques to test dynamic rejection of sudden angular momentum changes, short-duration forces were used to study the effects of sudden changes in linear momentum. These forces acted on the center of mass, again at $t = 5.1$ sec, again for a duration of 0.8 msec , and in each of four directions (upward, downward, forward, and hindward). Figure 4-7 shows the angular momentum (top row) and x-axis linear momentum (bottom row) effects originating from horizontal force perturbations. Figure 4-8 shows the x- and z-axis linear momentum response to vertically-directed forces.

A different class of perturbation tests were then performed. These tests measure the ability of the controller to reject continually applied, or "step" disturbances. At $t = 5.1$ sec, a force or torque is applied to the center of mass for the remainder of the simulation. If the hexapod completes the 10.0 seconds following perturbation without a gait failure, the magnitude of the perturbation is increased. This process is repeated until instability occurs after the step disturbance begins. In this manner, a



(a) 75 N-m at t=5.1sec, 0.8 msec



(b) –100 N-m at t=5.1sec, 0.8 msec

Figure 4-6: Impulse torque perturbations: resulting momenta vs. time for (a.) positive and (b.) negative torques.

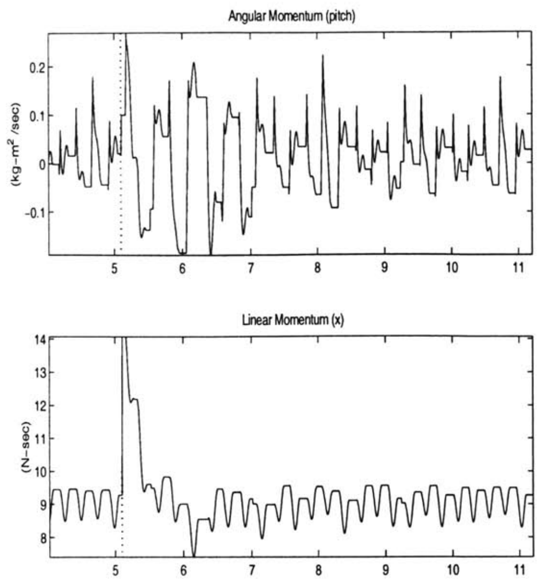
loose upper bound on rejected step magnitudes is established to at least two significant figures. Table 4.2 lists these maximal step magnitudes.

Perturbation	**Direction**	**Max Mag**	**Units**
Torque Step	positive	4.1	N-m
	negative	2.75	N-m
Force Step	forward	17.9	N
	hindward	5.5	N
	upward	21.75	N
	downward	8.1	N

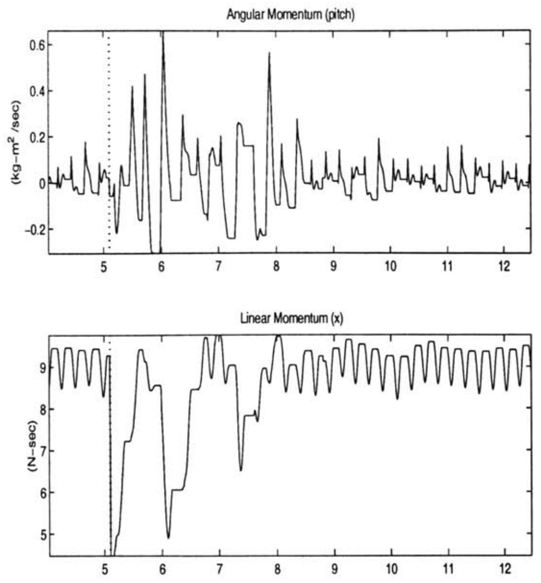
Table 4.2: Maximum magnitude of step perturbations resulting in stable gait, for at least 10.0 sec after perturbation. Perturbations applied for all time after 5.1 sec.

The angular momentum of the hexapod during normal locomotion is relatively low compared with the bounds of disturbance generated momentum peaks. Recent research within the lab has produced preliminary evidence that human walking and other locomotive behavior possesses a near-zero instantaneous system-wide angular momentum magnitude. This may relate to the overall stability occurring in the model. Further investigation is necessary.

What physical conditions can these step perturbations be seen to model? Continual forward and hindward forces occur during hill climbing and could be seen as velocity errors and certain ground conditions. Downward and upward forces also occur on slopes, test resilience to controller ground contact time mis-predictions, and evaluate operation in other gravitational fields. It is more difficult to reasonably interpret step torques; they should be considered as merely quantifying maximum



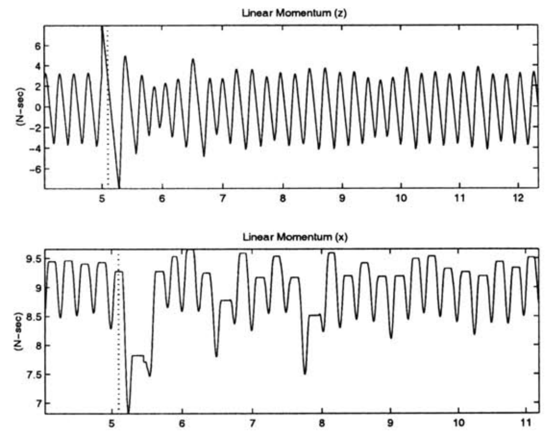
(a) Forward 600N, at t=5.1 sec, 0.8 msec



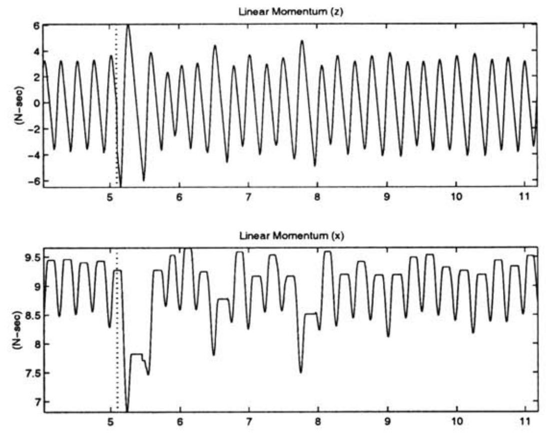
(b) Rearward 600N, at t=5.1 sec, 0.8 msec

Figure 4-7: Horizontal impulse force perturbation: resulting linear and angular momenta.

rejected angular momentum error for lack of a natural cause.



(a) Upward 200N, at t=5.1 sec, 0.8 msec



(b) Downward 400N, at t=5.1 sec, 0.8 msec

Figure 4-8: Vertical impulse force perturbation: resulting horizontal and vertical linear momenta.

4.3 Robustness

To test the robustness of the controller, a subspace of initial dynamic conditions is sampled. The goal is to determine the area inside this subspace for which stable, long term locomotion results. If locomotion is maintained for some defined period of time, then the initial conditions are considered to lead to a steady-state periodic gait. Ideally, the area over which this convergence occurs should be large.

The results of two separate tests are documented. The first test evaluates a subspace of the initial height and forward velocity. The height was varied over the range $0.3 \text{ m} \leq z \leq 0.8 \text{ m}$ in 0.1 m increments. Initial $t_{est\ contact}$ was adjusted appropriately. The velocity was varied over the range $0.2 \text{ m/sec} \leq \dot{x} \leq 2.0 \text{ m/sec}$ in 0.1 m/sec increments. In Figure 4-9, marked areas represent initial conditions from which stable gaits occur for a duration of at least 20.0 seconds. In all sampled cases, either the simulation converged to a periodic cycle or instability occurred within the first 5.0 seconds.

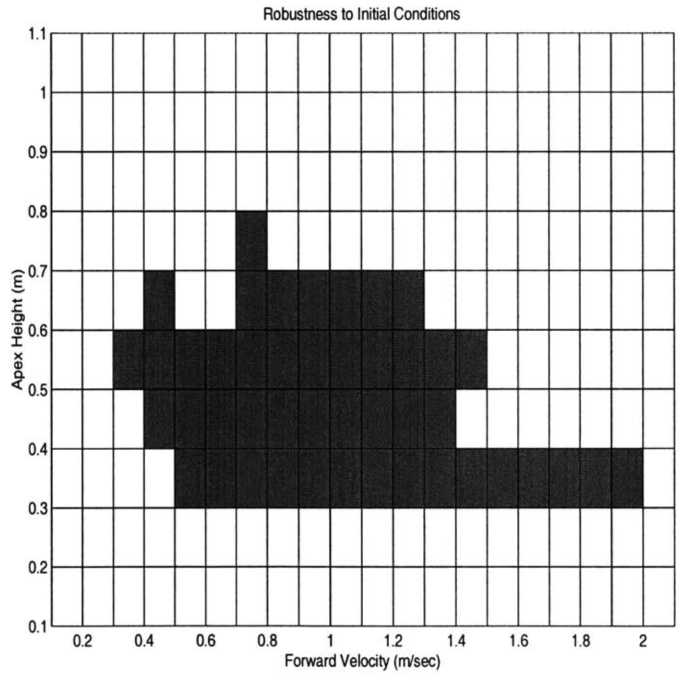


Figure 4-9: Robustness to initial height and velocity: dark areas represent conditions converging to a stable gait with duration of at least 20 seconds.

In the second test, the pitch and pitch derivative are varied to test orientation robustness. Figure 4-10 displays initial conditions which result in stable runs. The test samples initial pitch over the range $-1.3 \text{ rad} < \theta < 1.1 \text{ rad}$, in 0.01 rad increments, and the initial pitch derivative $\dot{\theta}$ over the range $-1.0 \text{ rad/sec} \leq \dot{\theta} \leq 1.0 \text{ rad/sec}$, in 0.5 rad/sec increments. Gaits lasting at least 10.0 sec were denoted as stable.

Somewhat surprisingly, a number of unstable initial pitch and derivative points lie

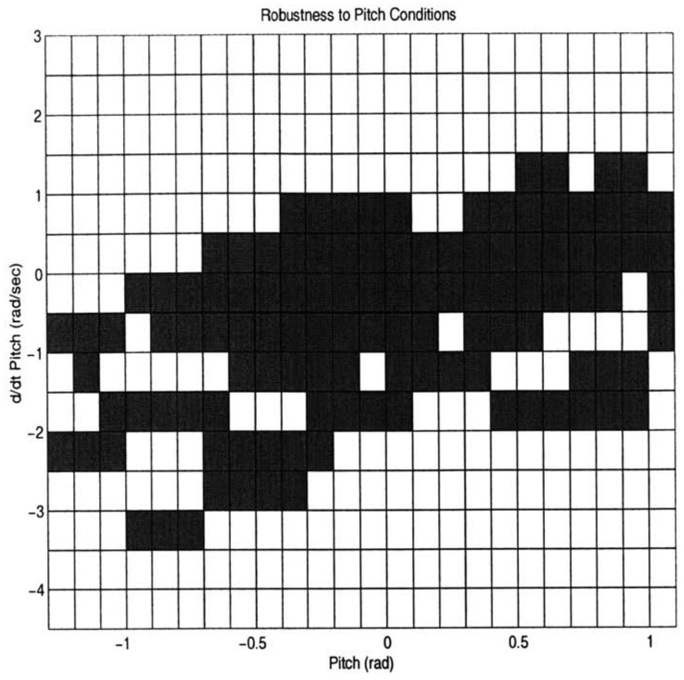


Figure 4-10: Robustness to initial pitch and pitch rate of change: dark areas represent conditions converging to a stable gait with duration of at least 10 seconds.

inside regions surrounded by stable initial conditions. I cannot produce a satisfactory explanation of these unstable points. It may suggest that certain body orientations at the instant of ground contact are more susceptible to instability caused by non-zero angular momentum values. Further investigation of these conditions is necessary.

4.4 Parameter Sensitivity

Parameter sensitivity testing measures the amount which parameter values may vary before the controller ceases to produce a stable long-term running gait. Very precise parameters indicate that using the algorithm on a physical platform will require very precise replication of the simulation and would therefore likely malfunction. Parameter sensitivity also hints at which values have the greatest effect on the stability of the system.

Parameter	Minimum	Nominal	Maximum	Units
k_{leg}	675 (-10%)	750	1100 (+46%)	N/m
G_{vel}	3.5 (-42%)	6.0	13.5 (+125%)	N-sec/m
G_{vel2}	-0.150 (-25%)	-0.2	-0.475 (+137%)	N-sec/m
$\dot{x}_{max}$	1.10 (-27%)	1.50	1.75 (+16%)	m/sec
body mass (leg mass constant)	1.49 (-52%)	3.104	5.23 (+68%)	kg
total mass (leg and body changed)	2.12 (-65%)	6.062	6.79 (+12%)	kg
target $\phi_{hip\ protract}$	0.310 (-23%)	0.400	0.550 (+37.5%)	rad
target $\phi_{hip\ retract}$	0.00 (-100%)	0.4	0.94 (+134%)	rad

Table 4.3: Independent sensitivity of parameter values: minimum, nominal, and maximum values for which a stable gait of more than 10.0 sec results.

A variation is made to an individual parameter value, and then the simulation is tested from standard initial conditions. By varying only a single parameter and holding the other values constant, the effects of that parameter can be isolated.

The parameter under test is reduced until a simulated trial fails to maintain a stable gait for at least 10 seconds. The last value for which a stable gait occurred is recorded. This lower bound is refined to a precision of at least two significant figures by binary search. The test is repeated, increasing the parameter to establish an upper bound. Results are tabulated in Table 4.3.

This test should only be interpreted as a "zeroth-order" evaluation of sensitivities. Due to the nonlinearity of the system, the results are specific to the operating point and conclusions cannot be made about the simultaneous variation of multiple parameters. Furthermore, the results cannot guarantee all parameter values lying within the minima and maxima produce stable gaits. There exists the possibility that some "island of instability" lies between sampled points. Although only anecdotal, it is encouraging to note that no such regions were discovered during testing.

The parameters which permitted the smallest variation were k_{leg} and the total system mass; both parameters changed by around 10%, but in opposite directions. One possible speculation supported by this is decreasing k_{leg} and increasing system mass may produce similar effects on motion in the controller. Body mass could be increased by over 2 kg before instability. Since leg masses are held constant, the overall system mass is increased by 2 kg and the proportion of body-to-leg mass changes. But increase of total mass (while maintaining body-to-leg proportions) causes instability after only 0.73 kg. This argues that the effect of leg mass on stability is more critical than that of body mass.

Additionally, without changing any other parameter value, speeds may be targeted over a 0.65 m/sec range without effecting stability. This single parameter speed selection was impossible in the quadruped trotting model; changing target speed required re-tuning a number of other parameters [13]. Finally, the insensitivity to the target leg protraction angle $\phi_{stance\ protract}$ is surprising in that previous control approaches have required a high degree of precision to properly regulate a trade-off of velocity and apex height [31].

4.5 Biological Plausibility

How can the similarity of biological running gaits and the gait generated here be quantitatively evaluated? Since the morphology of the system models no specific species, a direct comparison is impossible. Instead, cross-species metrics from the literature are calculated for the simulated hexapod gait. These metrics are then compared to values predicted by two biological models. Table 4.4 tabulates these calculations and predictions along with the size of one standard deviation variance. The table also includes brief descriptions of the metrics and their sources.

Gait Metric	**Sim**	**Bio**	**Bio 1 Stdev**	**Units**	**Description**	**Source**
F_{grf}	165	172.9	+49.61 / -38.54	N	peak vertical GRF	[10]
Δl	0.042	0.073	+0.023 / -0.017	meters	peak leg compression	[10]
θ_0	0.40	0.56	+0.10 / -0.86	rad	leg sweep half-angle	[10]
t_c	0.13	0.139	+0.016 / -0.014	sec	ground contact time	[10]
$\frac{T}{2}$	0.089	0.086	+0.008 / -0.007	sec	half resonant period	[10]
P_{spring}	1.18	4.63	+1.44 / -1.10	Watts/kg	leg-spring work rate	[10]
k_{leg}	3.956	2.39	+0.74 / -0.57	kN/m	combined leg stiffness	[10]
k_{vert}	7.50	7.86	+1.56 / -1.31	kN/m	equivalent vertical stiffness	[10]
$\frac{F_{grf}}{mg}$	2.77	1.78	$\pm$0.37	-	unitless peak force measure	[4]
$\frac{F_{grf}}{mg}leg^{-1}$	0.92	0.78	$\pm$0.29	-	unitless peak force per leg	[4]
$\frac{\Delta l}{l}$	0.168	0.096	$\pm$0.02	-	unitless leg comression	[4]
k_{rel}	16.48	18.60	$\pm$1.68	-	relative stiffness	[4]
$k_{rel}leg^{-1}$	5.48	7.82	$\pm$2.26	-	relative stiffness per leg	[4]

Table 4.4: Comparison of simulation measurements with biological model predictions.

These numerical predictions take one of two forms. Either they are logarithmic fits of observed data to functions of organism mass, or they are dimensionless quantities calculated from polypedal trotters. In both cases, they derive from simplified spring-mass models of running, and most are measured across multiple morphologies, gaits, and species. Limitations of space prevent the in-depth discussion of the meaning of the metrics, the assumptions of the models from which they derive, or the procedure by which biological values were obtained; instead, the reader is directed to the cited literature.

The peak ground reaction forces (F_{grf}), ground contact times (t_c), half-resonant period ($T/2$), vertical stiffness (k_{vert}), and normalized force per leg ($F_{grf}\text{leg}^{-1}$) all lie within a standard deviation of biological predictions. A possible conclusion: the "aerial to stance to aerial" bouncing behavior of the system is highly similar to biological systems. All metrics related to the stiffness of the leg lie within two standard deviations, indicating that the system stiffness may only partially replicate that of biological systems.

It is worth noting the value furthest afield, P_{spring}, is not unexpected. The metric measures the energetic equivalence of the leg behavior with a passive linear spring. Since the model of the leg the hexapod is just that, a value near 1.0 is expected. Natural systems do not manage quite this level of mechanical efficiency. The deviation from biological is a consequence of the morphological model.

Chapter 5

Conclusions and Future Work

This work makes the following contributions:

1. A new hexapedal locomotion controller which demonstrates a stable and fast gait in simulation.

2. Results indicating this gait responds favorably to variations in initial conditions and momenta during dynamic locomotion.

3. New numerical evidence suggesting that the class of controllers originating in the quadrupedal trotting model exhibit robust locomotion in hexapedal morphologies.

Although the simulated control algorithm has evaluated favorably, at least three far-reaching challenges must be met before the system described here reaches its full potential.

First, tests on a physical system must be performed. Without this confirmation, there is always the possibility some real-world factor was overlooked in these simulations. Currently, planned testing will be performed on the Scorpion robot. Should this fall through, or turn out mechanically implausible, physical trials will require the design and production of a new robot. In either case, such an evaluation is a necessary prerequisite to acceptance of the work here.

Second, the tedious hand-tuning procedures need to be automated in some fashion. Hand-tuning requires significant effort from an experienced individual with intuition for parameter choices. Additionally, many of the hypotheses made about this algorithm take the form, "This algorithm can produce *some behavior*." The hand-tuning procedure can never disprove such a statement, it can only prove by example. There always exist more unchecked combinations of parameters. Hand-tuning's labor-intensity and inability to negate certain conclusions make its automation a priority.

Finally, a more fundamental understanding of this algorithm needs to be developed. How do the control parameters relate to the morphology? What additional high-level feedback control could enhance stability? Can the algorithm be further simplified? What biological conclusions can be confirmed? A more advanced theoretical framework is necessary to address these questions. Mechanisms such as leg retraction, angular momentum matching, and ground contact prediction need additional theoretical explanation.

I hope the simple algorithm presented in this thesis, one of only a documented few capable of producing dynamically-stable hexapedal running gaits, will encourage and aid further legged locomotion research.

Appendix A

Mechanical Designs

This appendix briefly documents three mechanical designs I completed. The parts detailed here attempt to retrofit the Scorpion, a statically-stable walking hexapod robot, to enable it to operate with the algorithm detailed above. These segments replace the most distal segment of the legs. They increase the stroke length and stiffness to the values required by the simulation.

The initial passive distal segment design provides a stroke length of two inches. An interchangeable helical compression spring sets the stance stiffness, and a linear-stroke potentiometer measures segment compression. Figure A-1 shows CAD drawings of the complete distal assembly.

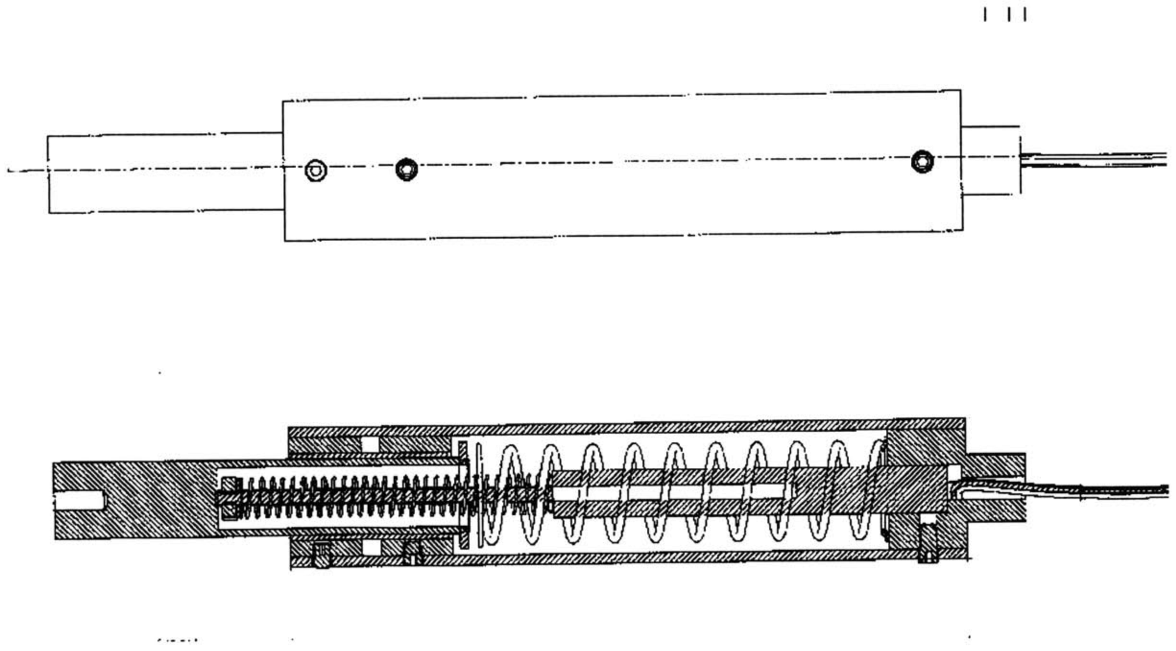


Figure A-1: Assembly drawings and section view of first distal segment design.

Due to external deadlines, time pressure dictated a simple design. Parts were conceived to be produced by a novice machinist with only a week of machining time; nine segments were produced in seven business days by a commercial shop.

The unit is housed in an aluminum tube casing chosen from stock tubing sizes and turned down to a requisite thickness. Two interior rings affix to the inner housing, oriented coaxially, with a small gap between them. These rings hold a polymer thrust bushing. A smaller tube moves in and out of the housing along this bushing. A motion stop, consisting of a washer pressfit onto the slide, holds the slide inside the housing during extension. An additional light compression spring provides return force for the potentiometer.

After production, trials found that the overall weight of the initial design may be too high. Plans were also altered, focusing on the eight-legged model of the Scorpion. This necessitated manufacture of more segments, so I took the opportunity to tune the design.

Figure A-2 shows the revised passive distal segment design. Its function is essentially the same as the initial design. The revision's improvements include:

1. Replacement of exterior screws. These screws may bind on the environment, over-constrain the design, and add a disassembly risk. The screws were replaced with interior threads.

2. Unnecessary metal was removed, reducing weight.

3. Reduced part count and assembly.

4. The bearing holder was eliminated, and instead the bearing is secured directly to the outer distal housing. This reduces weight and part count, rotationally confines the bearing, and presents easier assembly.

5. The washer pressfit to the top of the lower slide was removed. This washer acts as a motion stop. There was concern that the pressfit glue would fail, allowing the slide to separate from the distal segment. It was replaced with a machined lip integral to the slide.

6. Grooves to hold the spring in position were added.

7. A metal sleeve on the potentiometer holder protects the potentiometer from buckling due to misalignment.

8. The interface screw was too long, causing a visible gap between the hip and the distal segment.

Finally, additional work went into a preliminary pneumatic design (Figure A-3). The appeal of this design was its ability to inject energy during stance. Questions had arisen as to the Scorpion's eventual ability to provide the hip power required in the running algorithm. The initial power and energy analyses indicated that a small air tank could provide sufficient energy to operate the legs for the duration of battery life. The design was eventually scrapped for non-technical reasons.

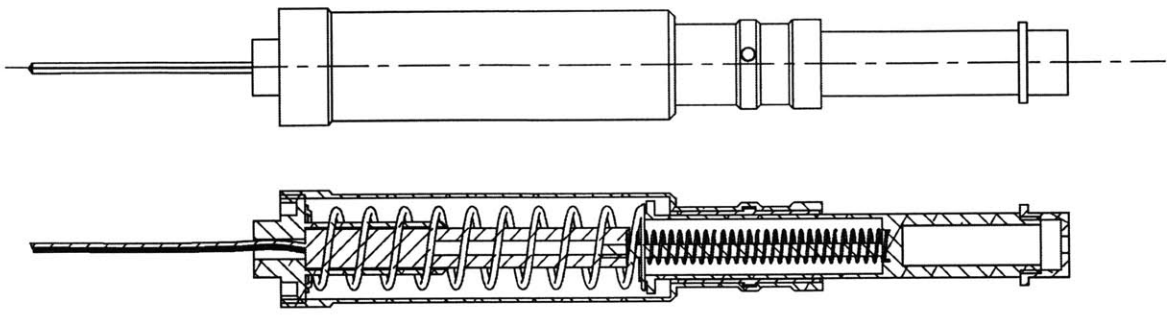


Figure A-2: Assembly drawing and section view of revised distal segment

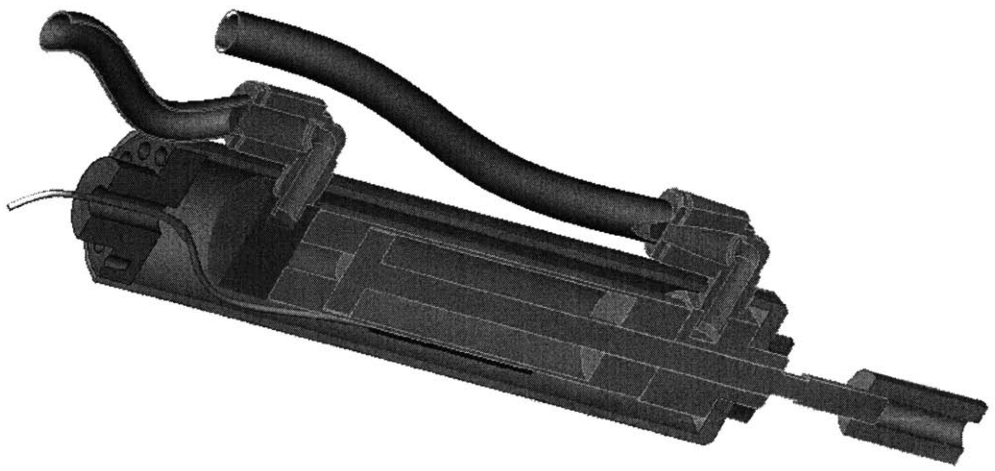


Figure A-3: Sectioned model of preliminary pneumatic distal segment

Appendix B

Control Source Code

The source code contained in this appendix:

8legs.h (Section B.1) is the header file which defines morphological values.

create_hexahop.c (Section B.2) contains the code which generates the hexapod morphology.

bug_hop.h (Section B.3) is the header file for control code.

bug_hop.c (Section B.4) is the code of the control presented in this thesis

Miscellaneous Functions (Section B.5) lists utility functions necessary for a complete understanding of the listed code.

I reduced the code to contain only functions directly relevant to the controller presented in this thesis. Utility functions, data collection, and experimental code has been removed. These files require Creature Library and SD-Fast in order to build the simulation.

Table B.1 describes the identifiers used in the source below, along with correspondences with the variable names used previously in the thesis.

Table B.1: Variable names used in simulation source code and coresponding symbols from text.

Physical State		
ls.t	t	simulation current time
q.x	x	body COM x position (in laboratory frame)
q.z	z	body COM z position (in laboratory frame)
q.pitch	θ	body pitch angle
q.hip*n*_z	ϕ_n	hip n protract/retract angle
q.hip*n*_x		hip n outward splay angle
q.knee*n*	l_n	knee n length of compression
qd.x	$\dot{x}$	body COM x velocity
qd.z	$\dot{z}$	body COM z velocity
qd.pitch	$\dot{\theta}$	body pitch rate of change
qd.hip*n*_z	$\dot{\phi}_n$	hip n protract/retract angular rate
qd.knee*n*	$\dot{l}_n$	knee n compression velocity
tau.hip*n*_z	τ_n	
tau.knee*n*	Fk_n	
Controller State		
ls.current_state		FSM current state (AERIAL or STANCE)
ls.next_state		FSM next state
ls.t_est_contact	$t_{est\ contact}$	estimated time of AERIAL phase end
gc_foot*n*.fs		foot n ground contact foot switch
ls.tripod		current stance tripod
ls.tangvel_d		target leg retraction speed
ls.t_takeoff		time of STANCE phase end
ls.moi_*n*	I_n	moment of intertia of leg n about hip joint
Controller Parameters		
ls.k_aerial_kn	$K_{aerial\ knee}$	AERIAL phase knee position gain
ls.b_aerial_kn	$B_{aerial\ knee}$	AERIAL phase knee position damping
ls.k_aerial_hip	$K_{aerial\ hip}$	AERIAL phase hip position gain
ls.b_aerial_hip	$B_{aerial\ hip}$	AERIAL phase hip position daming
ls.k_leg	k_{leg}	front/rear linear stance leg spring stiffness
ls.b_knee	b_{leg}	front/rear linear stance leg damping
ls.maxvel	$\dot{x}_{max}$	maximum forward velocity
ls.hip_retract	$\phi_{retract}$	AERIAL phase hip retraction target position
ls.hip_extend	$\phi_{protract}$	AERIAL phase hip protraction target position
ls.knee_extend	l_{extend}	AERIAL phase knee extension length
ls.leg_ratio		swing knee retraction distance for STANCE
	$l_{compress}$	defined as $l_{extend} - ls.leg\_ratio$
ls.G_vel	G_{vel}	STANCE phase stance tripod hip velocity gain
ls.G_vel2	G_{vel2}	STANCE phase swing tripod hip velocity gain

continued on next page

continued from previous page

Simulation Parameters	
ls.control_dt	time increment to execute control code
ls.plant_dt	step size for dynamic state evolution
ls.check_dt	time increment to check for dynamic drift
ls.servo_dt	time increment to update servo control
gnd_foot*n*.k_x	ground model stiffness (x direction)
gnd_foot*n*.b_x	ground model viscosity (x direction)
gnd_foot*n*.k_z	ground model stiffness (z direction)
gnd_foot*n*.b_z	ground model viscosity (z direction)
Servo Interface	
ls.k_*joint*	joint position gain
ls.b_*joint*	joint position damping
ls.ff_*joint*	joint feedforward torque
q_d.*joint*	desired joint position
ls.utau_*joint*	joint applied torque upper limit
ls.ltau_*joint*	joint applied torque lower limit
ls.k_stop	joint stop contact model stiffness
ls.b_stop	joint stop contact model viscosity

B.1 6legs.h

```
/* 6legs.h */
/* defined constants for hexahop body morphology */
/* Matt Malchano */

#define NUM_LEGS 6          /* number of legs */

#define BODY_X 0.620     /* body length (sagittal plane) */
#define BODY_Y 0.145     /* body width (frontal plane) */
#define BODY_Z 0.100     /* body height (transverse plane) */

#define SHIN_Z 0.176     /* shin length */
#define SHIN_M 0.0565    /* shin mass */
#define SHIN_RAD 0.024   /* shin cylinder and end-cap radius */

#define PIN_Z 0          /* pin link length (unused) */
#define PIN_M 0.0

#define THIGH_Z 0.076    /* thight link length (m) */
#define THIGH_M 0.0908   /* thigh link mass */
#define THIGH_RAD 0.15   /* thigh cylinder and end-cap radius */

#define BODY_M 3.104     /* body segment mass */

#define TOTAL_M (BODY_M + NUM_LEGS*(THIGH_M + SHIN_M + PIN_M))

#define KNEE_MIN 0.0     /* knee joint stop, min compression */
#define KNEE_MAX 0.2     /* knee joint stop, max compression */

#define HIPX_MIN −PI/2   /* hip joint stop, min angle */
#define HIPX_MAX PI/2    /* hip joint stop, max angle */

#define HIPZ_MIN −PI/2   /* hip joint stop, min angle */
#define HIPZ_MAX PI/2       /* hip joint stop, max angle */

#define SPONGE_Z 0.005   /* ground contact max penetration */
#define CONTACT_Z (SHIN_Z+SHIN_RAD−SPONGE_Z)

// the x positions of hip joint connections to the
// body indexed by hip joint number
// 0th ignored, since legs numbered from 1.
static const double PX[NUM_LEGS+1] = { 0.0,
                                       0.5 * BODY_X,
                                       0.0,
                                       −0.5 * BODY_X,
                                       −0.5 * BODY_X,
                                       0.0,
                                       0.5 * BODY_X };

// the y positions of hip joint connections to body
// indexed by hip joint number
// 0th ignored, since legs numbered from 1.
static const double PY[NUM_LEGS+1] = { 0.0,
                                       −BODY_Y, −BODY_Y, −BODY_Y,
                                       BODY_Y, BODY_Y, BODY_Y, };

// the z positions of hip joint connections to body
// indexed by hip joint number
// 0th ignored, since legs numbered from 1.
static const double PZ[NUM_LEGS+1] = { 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 };
```

B.2 create_hexahop.c

```
#include <stdio.h>
#include <cl_lib.h>
#include "8legs.h"

// function prototypes
void make_leg(int n);
void body_shape();
void thigh_shape(int legno);
void shin_shape(int legno);
void pin_shape();
void subn(char*, int);

int
main(int argc, char* argv[])
{
  int i = 0;
  command_line(argc, argv);
  begin_species("hexahop");

  // confine the model to sagittal plane
  new_link("base");
  set_parent(GROUND);
  new_joint("trainer");
  set_planar_joint("x", "z", "pitch", 'x', 'z', 'y');
  // this servo set limp.
  servo(1, "pitch", PD_FF_SERVO, "ls.k_pitch", "ls.b_pitch", "ls.ff_pitch");

  new_link("body");
  body_shape();

  //set_parent(FREE); // uncomment for 3D simulations
  for(i = 1; i <= NUM_LEGS; i++) make_leg(i);
  end_species();
}

/* make_leg(int n) constructs the morphology of leg n.
 * It perfoms some string replacement to generate the
 * names for the required variables. It then calls all
 * the CreatureLibrary routines nececssary
 */
void
make_leg(int n)
{
  // various labels that appear in final sim application
  char hip_z[12] = "hip#_z";
  char hip_x[12] = "hip#_x";

  char k_hip_z[15] = "ls.k_hip#_z";
  char b_hip_z[15] = "ls.b_hip#_z";
  char ff_hip_z[15] = "ls.ff_hip#_z";
  char utau_hip_z[20] = "ls.utau_hip#_z";
  char ltau_hip_z[20] = "ls.ltau_hip#_z";

  char k_hip_x[15] = "ls.k_hip#_x";
  char b_hip_x[15] = "ls.b_hip#_x";
  char ff_hip_x[15] = "ls.ff_hip#_x";
  char utau_hip_x[20] = "ls.utau_hip#_x";
  char ltau_hip_x[20] = "ls.ltau_hip#_x";

  char knee[12] = "knee#";
  char k_knee[15] = "ls.k_knee#";
  char b_knee[15] = "ls.b_knee#";
  char ff_knee[15] = "ls.ff_knee#";
  char utau_knee[20] = "ls.utau_knee#";
  char ltau_knee[20] = "ls.ltau_knee#";

  char shin[12] = "shin#";
  char foot[12] = "foot#";
  char thigh[12] = "thigh#";
  char pin[12] = "pin#";

  // replace '#' above with leg number
  subn(hip_z, n); subn(hip_x, n); subn(k_hip_z, n); subn(b_hip_z, n); subn(ff_hip_z, n);
  subn(k_hip_x, n); subn(b_hip_x, n); subn(ff_hip_x, n);
  subn(knee, n); subn(k_knee, n); subn(b_knee, n); subn(ff_knee, n); subn(shin, n); subn(foot, n);
  subn(thigh, n); subn(pin,n);
  subn(utau_hip_z, n); subn(ltau_hip_z, n); subn(utau_hip_x, n); subn(ltau_hip_x, n);
  subn(utau_knee, n); subn(ltau_knee, n);

  // create the hip protraction/retraction joint
  joint_pin(hip_z,'y');
  limit(hip_z, HIPZ_MIN, HIPZ_MAX);
  servo(1, hip_z, PD_FF_SERVO, k_hip_z, b_hip_z, ff_hip_z);
  user_servo(5, hip_z, "pd_ff_taulimit_servo", 5, k_hip_z, b_hip_z, ff_hip_z, utau_hip_z, ltau_hip_z);
  set_parent("body");
  set_joint_offset(PX[n], PY[n], PZ[n]);

  // the pin link allows us to seperate the joint axes, but currently
  // its length is 0.
  new_link(pin);
  pin_shape();

  // create the hip sprawl joint
  joint_pin(hip_x,'x');
  limit(hip_x, HIPZ_MIN, HIPX_MAX);
  servo(1, hip_x, PD_FF_SERVO, k_hip_x, b_hip_x, ff_hip_x);
  user_servo(5, hip_x, "pd_ff_taulimit_servo", 5, k_hip_x, b_hip_x, ff_hip_x, utau_hip_x, ltau_hip_x);
  set_joint_offset(0.0,0.0,0.0);

  // generate the thigh link and give it the appropriate shape
  new_link(thigh);
  thigh_shape(n);

  // generate the knee joint
  joint_slider(knee,'z');
  limit(knee, KNEE_MIN, KNEE_MAX);
  servo(1, knee, PD_FF_SERVO, k_knee, b_knee, ff_knee);
  user_servo(5, knee, "pd_ff_taulimit_servo", 5, k_knee, b_knee, ff_knee, utau_knee, ltau_knee);
  set_joint_offset(0.0, 0.0, -THIGH_Z);

  // finally, generate the shin and create a ground contact test point
  // at the tip of the foot.
  new_link(shin);
  shin_shape(n);
  ground_contact(foot, 0.0,0.0,-CONTACT_Z);
}

/* subn(char* s, int n) destructively modifies string s,
 * replacing all occurences of '#' with the single digit
 * number n's character value.
 */
void
subn(char* s, int n)
{
  while(*s != '\0') {
    if(*s == '#') *s = (char)(n+48);
    s++;
  }
}

/* body_shape() makes the body shape, a big rectangular box.
 * It also adds ground contact detectors to the bottom corners.
 */
void
body_shape()
{
  begin_shape();
  use_density(0.25 * 1.76 * ALUMINUM_DENSITY/4.0);
  translate(0.0,0.0,-BODY_Z/2.0);
  shape(SBRICK, BODY_X, BODY_Y, BODY_Z);
  ground_contact("body1", BODY_TOT*0.35,-BODY_Y/2.0,-BODY_Z/2.0);
  ground_contact("body2", BODY_TOT*0.35,BODY_Y/2.0,-BODY_Z/2.0);
  ground_contact("body3", -BODY_TOT*0.35,BODY_Y/2.0,-BODY_Z/2.0);
  ground_contact("body4", -BODY_TOT*0.35,-BODY_Y/2.0,-BODY_Z/2.0);
  end_shape();
}

/* pin_shape() generates a cylindrical pin which connects
 *  the two hip joints together.
 */
void
pin_shape()
{
  begin_shape();
  use_density(ALUMINUM_DENSITY/4.0);
  translate(0.0,0.0,-THIGH_Z/100.0);
  shape(SCYLINDER, THIGH_Z/50.0,THIGH_RAD);
  end_shape();
}

/* thigh_shape(int legno) creates the thigh shape. It
 * is optionally passed a legno to customize the leg's
 * mass (unused).
 */
void
thigh_shape(int legno)
{
  begin_shape();

  if((legno == 2 || legno == 5) && DBL_MID_MASS) {
    use_density(2.0 * 0.25 * ALUMINUM_DENSITY/8.0);
  } else  use_density(0.25 * ALUMINUM_DENSITY/8.0);

  shape(SSPHERE,JOINT_RAD);
  translate(0.0,0.0,-THIGH_Z);
  shape(SCYLINDER, THIGH_Z,THIGH_RAD);
  shape(SSPHERE,JOINT_RAD);
  end_shape();
}

/* shin_shape(int legno) creates the shin shape. */
void
shin_shape(int legno)
{
  begin_shape();

  if((legno == 2 || legno == 5) && DBL_MID_MASS)
    use_density(2.0 * 0.25 * ALUMINUM_DENSITY/8.0);
  else
    use_density(0.25 * ALUMINUM_DENSITY/8.0);

  translate(0.0,0.0,-SHIN_Z);
  shape(SCYLINDER, SHIN_Z,SHIN_RAD);
  identity();
  rotate('y', 180.0);
  translate(0.0,0.0,-SHIN_Z);
  shape(SHEMISPHERE,SHIN_RAD);
  end_shape();
}
```

B.3 bug_hop.h

```
#include <stdio.h>
#include <math.h>
#include "creature.h"
#include "8legs.h"

#ifndef __BUG_HOP_H
#define __BUG_HOP_H

#define AERIAL        00
#define STANCE        20

#define LEG_LEN       (THIGH_Z + PIN_Z+ SHIN_Z)
#define M_THIGH       0.46411
#define M_SHIN        0.0293018

void bug_hop2();

static void aerial_controller();
static void stance_controller();
static void angmo_instrument();
static double angmo(int, double, double, double, double, double*, double*, double*);

#endif
```

B.4 bug_hop.c

```
#include "bug_hop.h"

/* bug_hop2() is the entry function into the control code.
 * It is called once every control_dt seconds, examines the
 * dynamic state of the robot, and sets the relevent servo
 * variables and tau torque output variables to generate the
 * gait. Depending on the FSM state, it calls either an aerial
 * control function or stance control function.
 */
void bug_hop2() {
  ls.current_state = ls.next_state;
  switch((int)ls.current_state) {
  case(AERIAL):
    aerial_controller(); break;
  case(STANCE):
    stance_controller(); break;
  }
}

/* aerial_control() implements AERIAL phase control */
void aerial_controller() {
    ls.k_knee1 = ls.k_aerial_kn;
    ls.b_knee1 = ls.b_aerial_kn * sqrt(ls.k_knee1 * (THIGH_M + SHIN_M));

    ls.k_knee2 = ls.k_knee3 = ls.k_knee4 = ls.k_knee5 = ls.k_knee6 = ls.k_knee1;
    ls.b_knee2 = ls.b_knee3 = ls.b_knee4 = ls.b_knee5 = ls.b_knee6 = ls.b_knee1;

    ls.k_hip1_z = ls.k_aerial_hip;
    ls.b_hip1_z = ls.b_aerial_hip;

    ls.k_hip2_z = ls.k_hip3_z = ls.k_hip4_z = ls.k_hip5_z = ls.k_hip6_z = ls.k_hip1_z;
    ls.b_hip2_z = ls.b_hip3_z = ls.b_hip4_z = ls.b_hip5_z = ls.b_hip6_z = ls.b_hip1_z;

      if(ls.tripod == 0) {
        q_d.hip1_z = q_d.hip5_z = q_d.hip3_z = ls.hip_retract;
        q_d.hip2_z = q_d.hip4_z = q_d.hip6_z = ls.hip_extend;

        q_d.knee1 = q_d.knee5 = q_d.knee3 = ls.knee_extend + ls.leg_ratio;
        q_d.knee2 = q_d.knee4 = q_d.knee6 = ls.knee_extend;
      } else {
        q_d.hip2_z = q_d.hip4_z = q_d.hip6_z = ls.hip_retract;
        q_d.hip1_z = q_d.hip3_z = q_d.hip5_z = ls.hip_extend;

        q_d.knee2 = q_d.knee4 = q_d.knee6 = ls.leg_ratio + ls.knee_extend;
        q_d.knee1 = q_d.knee3 = q_d.knee5 = ls.knee_extend;
      }

    if(ls.t + ls.control_dt >= ls.t_est_contact) {
      // The controller will enter STANCE phase next
      // control execution. Prepare for the transition.

      if(ls.tripod == 1) {
        ls.k_knee1 = ls.k_leg;
        ls.k_knee3 = ls.rk_leg_mult * ls.k_leg;
        ls.k_knee5 = ls.k_knee1 + ls.k_knee3;
        ls.b_knee1 = ls.b_knee5 = ls.b_knee3 = ls.b_knee;

        ls.k_knee6 = ls.k_knee4 = ls.k_knee2 = 0.6 * ls.k_leg;
        ls.b_knee6 = ls.b_knee4 = ls.b_knee2 = 0.3 * ls.b_knee;
      } else {
        ls.k_knee6 = ls.k_leg;
        ls.k_knee4 = ls.rk_leg_mult * ls.k_leg;
        ls.k_knee2 = ls.k_knee4 + ls.k_knee6;
        ls.b_knee2 = ls.b_knee4 = ls.b_knee6 = ls.b_knee;

        ls.k_knee1 = ls.k_knee5 = ls.k_knee3 = 0.6 * ls.k_leg;
        ls.b_knee1 = ls.b_knee5 = ls.b_knee3 = 0.3 * ls.b_knee;
      }

      ls.k_hip1_z = ls.k_hip3_z = ls.k_hip5_z = 0.0;
      ls.k_hip2_z = ls.k_hip4_z = ls.k_hip6_z = 0.0;
      ls.b_hip1_z = ls.b_hip3_z = ls.b_hip5_z = 0.0;
      ls.b_hip2_z = ls.b_hip4_z = ls.b_hip6_z = 0.0;

      ls.tangvel_d = (ls.tripod?ls.maxvel:-ls.maxvel);

      ls.next_state = STANCE;
    }
}

/* stance_controller() implements STANCE phase control */v
oid stance_controller() {
    // depending on the active tripod...
    if(ls.tripod == 1) {
      // stance leg velocity control loops
      ls.ff_hip1_z = -ls.G_vel * (qd.hip1_z * (0.25 - q.knee1) - ls.tangvel_d);
      ls.ff_hip5_z = -ls.G_vel * (qd.hip5_z * (0.25 - q.knee5) - ls.tangvel_d);
      ls.ff_hip3_z = -ls.G_vel * (qd.hip3_z * (0.25 - q.knee3) - ls.tangvel_d);
    // swing leg velocity control loops
    ls.ff_hip4_z = ls.G_vel2 * (qd.hip4_z + (ls.moi_3/ls.moi_4) * qd.hip3_z);
    ls.ff_hip2_z = ls.G_vel2 * (qd.hip2_z + (ls.moi_5/ls.moi_2) * qd.hip5_z);
    ls.ff_hip6_z = ls.G_vel2 * (qd.hip6_z + (ls.moi_1/ls.moi_6) * qd.hip1_z);
  } else {
    // stance leg velocity control loops
    ls.ff_hip4_z = -ls.G_vel  * (qd.hip4_z * (0.25 - q.knee4) + ls.tangvel_d);
    ls.ff_hip2_z = -ls.G_vel  * (qd.hip2_z * (0.25 - q.knee2) + ls.tangvel_d);
    ls.ff_hip6_z = -ls.G_vel  * (qd.hip6_z * (0.25 - q.knee6) + ls.tangvel_d);

    // swing leg velocity control loops
    ls.ff_hip1_z = ls.G_vel2 * (qd.hip1_z + (ls.moi_6/ls.moi_1) * qd.hip6_z);
    ls.ff_hip5_z = ls.G_vel2 * (qd.hip5_z + (ls.moi_2/ls.moi_5) * qd.hip2_z);
    ls.ff_hip3_z = ls.G_vel2 * (qd.hip3_z + (ls.moi_4/ls.moi_3) * qd.hip4_z);
  }

  if(gc_foot1.fs || gc_foot2.fs ||  gc_foot3.fs || gc_foot4.fs ||  gc_foot5.fs || gc_foot6.fs) {
    // a ground contact has occurred during stance phase.
    // this prevents leaving stance phase prematurely.
    ls.gnd_contact = 1.0;
  }

  if(!gc_foot1.fs && !gc_foot2.fs && !gc_foot3.fs
     && !gc_foot4.fs && !gc_foot5.fs && !gc_foot6.fs
     && ls.gnd_contact == 1.0) {
    // All the feet have left the ground after a ground
    // contact had occured, go ahead and switch into aerial phase.
    ls.next_state = AERIAL;
    ls.tripod = !(int)ls.tripod;    // swap the swing/stance tripods
    ls.gnd_contact = 0.0;

    ls.t_takeoff = ls.t;
    // predict next ground contact.
    ls.t_est_contact = ls.G_delay*(qd.z/9.812) + ls.t_takeoff;
  }
}

/* angmo_instrument() calculates the moments of intertia for
 * each leg about the hip axis.
 */
void angmo_instrument() {
  ls.moi_1 = 9.5545e-4 + 7.7044e-5 + 0.0293*sq(0.25 - q.knee1);
  ls.moi_2 = 9.5545e-4 + 7.7044e-5 + 0.0293*sq(0.25 - q.knee2);
  ls.moi_3 = 9.5545e-4 + 7.7044e-5 + 0.0293*sq(0.25 - q.knee3);
  ls.moi_4 = 9.5545e-4 + 7.7044e-5 + 0.0293*sq(0.25 - q.knee4);
  ls.moi_5 = 9.5545e-4 + 7.7044e-5 + 0.0293*sq(0.25 - q.knee5);
  ls.moi_6 = 9.5545e-4 + 7.7044e-5 + 0.0293*sq(0.25 - q.knee6);
}

/* calcuate leg angular momentum in system center of mass frame.
 * currently unused.
 */
double angmo(int leg, double theta, double dtheta, double lknee, double dknee,
             double* amx, double* amy, double*amz)
{

  double thigh_Icm, shin_Icm;
  double s_mrxr_x, s_mrxr_y, s_mrxr_z;
  double t_mrxr_x, t_mrxr_y, t_mrxr_z;
  double l;

  // location of shin com
  l = THIGH_Z + (SHIN_Z/2) - lknee;

  // revolution term moments about com.
  thigh_Icm = (1/12)*M_THIGH*(sq(THIGH_Z) + sq(THIGH_X)) + M_THIGH*(THIGH_Z/2)*(THIGH_Z/2);
  shin_Icm = (1/12)*M_SHIN*(sq(SHIN_Z) + sq(SHIN_X)) + M_SHIN*l*l;
  // M(r X rdot): orbital term for shin and thigh
  s_mrxr_x = M_SHIN*(−PY[leg]*cos(theta)*(−dknee) + PY[leg]*l*sin(theta)*dtheta);
  s_mrxr_y = M_SHIN*(PX[leg]*cos(theta)*(−dknee) − (sq(l) + PX[leg]*l*abs(sin(theta)))*dtheta);
  s_mrxr_z = M_SHIN*(−PY[leg]*sin(theta)*(−dknee) + PY[leg]*l*cos(theta)*dtheta);

  t_mrxr_x = M_THIGH*(−PY[leg]*(THIGH_Z/2)*sin(theta))*dtheta;
  t_mrxr_y = M_THIGH*(sq(THIGH_Z/2) + PX[leg]*(THIGH_Z/2)*abs(sin(theta)))*dtheta;
  t_mrxr_z = M_THIGH*(PY[leg]*(THIGH_Z/2)*cos(theta))*dtheta;

  // angular momentum totals for the leg.
  *amx = s_mrxr_x + t_mrxr_x;
  *amy = thigh_Icm*dtheta + shin_Icm*dtheta;
  *amz = s_mrxr_y + t_mrxr_y + thigh_Icm*dtheta + shin_Icm*dtheta;
  //*amz = s_mrxr_z + t_mrxr_z;
}
```

B.5 Miscellaneous Functions

These utility functions are excerpted from unlisted source files.

```
/* do_pd_ff_servo implements a propotional-dervative-feedforward
 * torque servo. It is never directly called by the control code
 * but instead is called every servo_dt, where it examines the
 * ls state variables q, qd, q_d... and modifies the value of
 * tau as appropriate.
 */
void
do_pd_ff_servo(tau, q, qd, q_d, k, b, ff)
     double *tau, q, qd, q_d;
     double k, b, ff;
{
  *tau = −k*(q − q_d) − b*qd + ff;
}

/* nonlinear_spring_damper_ground handles ground contacts.
 * A PD feedback loop, along with the gnd_contact stiffness
 * and damping values generates unidirectional forces which
 * prevent slipping and pentration.
 */
void
nonlinear_spring_damper_ground(gc, x, y, z, dx, dy, dz)
     basic_gcontact *gc;
     double x, y, z, dx, dy, dz;
{
  if (z > 0.0)
    gc−>fs = 0.0;
  if (z < 0.0)
    if (gc−>fs < 0.5)
      {
        gc−>fs = 1.0;
        gc−>td_x = x;
        gc−>td_y = y;
        gc−>td_z = z;  /* Note that td_z isn't actually used, zero is */
      }
  if (gc−>fs > 0.5)
    {
      /* Foot switch is on, need to apply forces. */
      /* x direction */
      gc−>f_x = (gc−>td_x − x);
      if (x > gc−>td_x) /* f_x is positive, the force should be negative */
        gc−>f_x = −(gc−>k_x)*gc−>f_x*gc−>f_x
          − gc−>b_x*dx;
      else
      gc->f_x = (gc->k_x)*gc->f_x*gc->f_x
        - gc->b_x*dx;

    /* y direction */
    gc->f_y = (gc->td_y - y);
    if (y > gc->td_y) /* f_y is positive, the force should be negative */
      gc->f_y = -(gc->k_y)*gc->f_y*gc->f_y
        - gc->b_y*dy;
    else
      gc->f_y = (gc->k_y)*gc->f_y*gc->f_y
        - gc->b_y*dy;

    /* z direction */
    gc->f_z = (gc->td_z - z);
    /* You know z < gc->td_z or you wouldn't be in this part of the code */
    gc->f_z = (gc->k_z)*gc->f_z*gc->f_z
        - gc->b_z*dz;
    /* Zero z direction if it would result in sticky ground. */
    if (gc->f_z < 0.0) gc->f_z = 0.0;

    gc->t_th = 0.0;
   }
  else
   {
    gc->f_x = 0.0;
    gc->f_y = 0.0;
    gc->f_z = 0.0;
    gc->t_th = 0.0;
   }
}

/* pd_ff_taulimit_servo adds output saturation to
 * the standard proportional-derivative-feedforward
 * servo model
 */
void
pd_ff_taulimit_servo(tau, q, qd, q_d, k, b, ff, utau, ltau)
     double *tau, q, qd, q_d;
     double k, b, ff;
     double utau, ltau;
{
  *tau = -k*(q - q_d) - b*qd + ff;
  if(utau > 0.0 && *tau > utau) *tau = utau;
  if(ltau < 0.0 && *tau < ltau) *tau = ltau;
}


/* some math functions */
double
sq(double a) {
return a * a;
}

double
dmin(double a, double b) {
  if(a < b) return a;
  else return b;
}

double
dmax(double a, double b) {
  if(a >= b) return a;
  else return b;
}
```

Bibliography

[1] Paul Alexandre and Andre Preumont. On the gait control of a six-legged walking machine. *International Journal of System Science*, 27(8):713–721, 1996.

[2] R. Altendorfer, N. Moore, H. Kumsuoglu, M. Buehler, H. B. Brown Jr., D. McMordie, U. Saranli, R. Full, and D. E. Koditschek. RHex: A biologically inspired hexapod runner. *Autonomous Robots*, 11:207–213, 2001.

[3] R. D. Beer, H. J. Chiel, R. D. Quinn, K. Espenschied, and P.Larsson. A distributed neural net architecture for hexapod robot locomotion. In *Neural Computation*, volume 4, pages 356–365. M.I.T. Press, Cambridge, MA, 1992.

[4] Robert Blickhan and Robert J. Full. Similarity in multilegged locomotion: Bouncing like a monopode. *Journal of Comparative Physiology A*, 173:509–517, 1993.

[5] G.A. Cavagna, N. C. Heglund, and C. R. Taylor. Mechanical work in terrestrial locomotion: Two basic mechanisms for minimizing energy expenditure. *American Journal of Physiology*, 233:243–261, 1977.

[6] J. G. Cham, S. A. Bailey, and M. R. Cutosky. Robust dynamic locomotion through feedforward-preflex interaction. In *ASME IMECE Proceedings*, Orlando, Florida, November 2000.

[7] Jorge G. Cham, Jonathan Karpick, Jonathan E. Clark, and Mark R. Cutkosky. Stride period adaptation for a biomimetic running hexapod. In *10th Annual Symposium of Robotics Research*, Lorne, Victoria, Australia, November 2001.

[8] Jonathan E. Clark, Jorge G. Cham, Sean A. Bailey, Edward M. Froehlich, Pratik K. Nahata, Robert J. Full, and Mark R. Cutosky. Biomimetic design and fabrication of a hexapedal running robot. In *IEEE International Conference on Robotics and Automation*, 2001.

[9] Parametric Technology Corporation. SD-Fast. Software.

[10] Claire T. Farley, James Glasheen, and Thomas A. McMahon. Running springs: Speed and animal size. *Journal of Experimental Biology*, 185:71–86, 1993.

[11] C. Ferrell. A comparison of three insect-inspired locomotion controllers. *Robotics and Autonomous Systems*, 16(2-4):135–159, 1995.

[12] Robert J. Full and Michael S. Tu. The mechanics of six-legged runners. *Journal of Experimental Biology*, 148:129–146, 1990.

[13] Hugh M. Herr. personal communications.

[14] Hugh M. Herr, Gregory T. Huang, and Thomas A. McMahon. A model of scale effects in mammalian quadrupedal running. *Journal of Experimental Biology*, 205:959 – 967, 2002.

[15] Hugh M. Herr and Thomas A. McMahon. A trotting horse model. *International Journal of Robotics Research*, 19(6):566–581, June 2000.

[16] Hugh M. Herr and Thomas A. McMahon. A galloping horse model. *International Journal of Robotics Research*, 20(1):26–37, January 2001.

[17] G. M. Hughes. The co-ordination of insect movements. I. the walking movements of insects. *Journal of Experimental Biology*, 29:267–285, 1952.

[18] Winfried Ilg and Karsten Burns. A learning architecture based on reinforcement learning for adaptive control of the walking machine Lauron. *Robotics and Autonomous Systems*, 15:321–334, 1995.

[19] Frank Kirchner. personal communications.

[20] Haldun Komsuoglu, Dave McMordie, Uluc Saranli, Ned Moore, Martin Buehler, and Daniel E. Koditschek. Proprioception based behavioral advances in a hexapod robot. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, May 2001. IEEE.

[21] MIT Leg Lab. Creature Library. Software.

[22] M. Anothony Lewis, Andrew H. Fagg, and George A. Bekey. Genetic algorithms for gait synthesis in a hexapod robot. In Zheng, editor, *Recent Trends in Mobile Robotics*, pages 317–331. World Scientific, New Jersey, 1994.

[23] R. B. McGhee and A. A. Frank. On the stability properties of quadruped creeping gaits. *Mathematical Biosciences*, 3:331–351, 1968.

[24] Thomas A. McMahon. *Muscles, Reflexes, and Locomotion.* Princeton University Press, 1984.

[25] Thomas A. McMahon and G.C. Cheng. The mechanics of running: How does stiffness couple with speed? *Journal of Biomechanics*, 23:65–78, 1990.

[26] G. M. Nelson, R. D. Quinn, Bachmann, W. C. Flannigan, R. E. Ritzmann, and J. T. Watson. Design and simulation of a cockroach-like hexapod robot. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA '97)*, Alburquerque, New Mexico, April 1997.

[27] Gill A. Pratt, Daniel Paluska, Andreas Hofman, and Matt Malchano. Report on the UGCV. Presented to Omnitech Robotics LLC, Colorado, and DARPA, August 2001.

[28] Jerry Pratt. Virtual model control of a biped walking robot. Master's thesis, Massachusetts Institute of Technology, 1995.

[29] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing.* Cambridge University Press, 1993.

[30] Roger D. Quinn, Gabriel M. Nelson, Richard J. Bachmann, Daniel A. Kingsley, John Offi, and Roy E. Ritzmann. Insect designs for improved robot mobility. In Berns and Dillmann, editors, *Proc. 4th Int. Conference On Climbing and Walking Robots*, pages 69–76, 2001.

[31] Mark H. Raibert. *Legged Robots That Balance.* MIT Press, 1986.

[32] Christian Ridderstrm. Legged locomotion control — a literature survey. Technical Report TRITA-MMK 1999:27, Dept. of Machine Design, Royal Institute of Technology, S-100 44 Stockholm, Sweden, November 1999.

[33] Robert Ringrose. *Self-Stabilized Running.* PhD thesis, Massachusetts Institute of Technology, 1996.

[34] Uluc Saranli, Martin Buehler, and Daniel E. Koditschek. RHex: A simple and highly mobile hexapod robot. *International Journal of Robotics Research*, 20(7):616–631, July 2001.

[35] A. Seyfarth, H. Geyer, R. Blickhan, and H. Herr. Leg retraction – a simple control strategy for stable running. (in preparation), 2002.

[36] S. Talebi, I. Poulakais, E. Papadopoulos, and Martin Buehler. *Experimental Robotics VII*, chapter Quaruped Robot Running with a Bounding Gait, pages 281–289. Lecture Notes in Control and Information Sciences; 271. Springer-Verlag, 2001.

[37] Ann L. Torres. Virtual model control of a hexapod walking robot. Technical Report AITR–1582, Massachusetts Institute of Technology AI Lab, 1996.

[38] H. J. Weidemann, F. Pfeiffer, and J. Eltze. A design concept for legged robots derived from the walking stick insect. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Yokohama, Japan, July 1993.

[39] David Wettergreen and Chuck Thorpe. Gait generation for legged robots. In *IEEE International Conference on Intelligent Robots and Systems.* IEEE, July 1992.

[40] David Wettergreen and Chuck Thorpe. Dante II: Technical description, results, and lessons learned. *International Journal of Robotics Research*, 18(7):621–649, 1999.

[41] Jung-Min Yang and Jong-Hwan Kim. Optimal fault tolerant gait sequence of the hexapod robot with overlapping reachable areas and crab walking. *IEEE Transactions on Systems, Man, and Cybernetics A*, 29(2):224–235, March 1999.