University of California
Los Angeles

Modeling, Simulation, and Control of Two-Legged Walking

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Alexis Paul Wieland

1994

To my son Christopher
who learned to walk both
more quickly and more proficiently
than any of these computer programs.

Table of Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{A}$, $\mathbf{B}$	Arbitrary matrices
$\mathbf{A}^{-1}$	Inverse of the matrix $\mathbf{A}$
$\mathbf{A}^{+}$	Pseudo (Moore-Penrose) inverse of the matrix $\mathbf{A}$
$\mathbf{c}$	Constraints
$\mathcal{C}$	Set of contact points
$e(t)$	Continuous time error
e_t	Discrete time error
$fric_i$	Nonconservative forces, i.e., friction
$fric_{\mathrm{b,i}}$	Bearing friction
F_n	Normal force
F_t	Tangential force
g	Gravitation acceleration, i.e., 9.8 m/s^2
I	Inertia
$\mathbf{J}$	Jacobian matrix
K_p, K_i, K_d	PID control parameters
l	Length
m	Mass
M	Number of active constraints
$\mathbf{M}$	Modal matrix
N	Number of coordinates, i.e., 9
$\mathbf{q}$	Vector of generalized coordinates q_i

$\mathbf{Q}$	Vector of generalized forces Q_i
$\mathcal{S}$	Set of segments
t	Time
T	Kinetic energy
$\mathbf{u}$	Vector of control signals u_i
$\mathbf{U}$	Vector of actuation bounds U_i
V	Potential energy
x, y	Horizontal and vertical coordinates
$\mathbf{x}$, $\mathbf{y}$	Arbitrary vectors
Δt	Simulation time step, i.e., $\frac{1}{1000}$ second
$\boldsymbol{\theta}$	Vector of joint angles θ_i
λ_i	Lagrangian multipliers
$\boldsymbol{\Lambda}$	Diagonal matrix diag$[\lambda_{i,i}]$
$\mu_{\mathrm{b1},i}$	Bearing friction coefficient, proportional
$\mu_{\mathrm{b2},i}$	Bearing friction coefficient, constant
μ_d	Foot friction coefficient, dynamic
μ_s	Foot friction coefficient, static
$\sigma(x)$	Sigmoid transfer function
$\boldsymbol{\phi}$	Vector of segment angles ϕ_i
ξ	Angle of force into the ground
$\boldsymbol{\psi}$	Vector of constraints ψ_i
ω	Angular velocity

ACKNOWLEDGMENTS

It is with a profound sense of gratitude that I acknowledge and thank some of the many people that have helped me during my years here at UCLA.

It seems appropriate to start with my committee. The sum total of their knowledge and wisdom is enormous, it has been an honor to learn at their feet. I feel certain that my advisor, Walter Karplus, does not appreciate how important his quiet persistence, encouragement, and very sage advice have been.

A number of present and former students had a strong influence on me: Rob Collins the artificial life wizard, Greg Werner the dreamer and confidant (who showed me the WDI advertisement), Adam King the software magnate and philosopher, Lorraine Fesq the astronaut that beat me out of here, Edmond Mesrobian who graduated first but is still here, and Trent Lange and Ron Sumida the AI neural net guys and keepers of the board.

I probably would not have graduated without the friendship and assistance from the department staff. It was Verra Morgan who always knew the *real* answers, Shelly Backlar with whom I swapped stories about our great kids, and Doris Sublete who always smiled, even when the teas disrupted her archive.

Outside work has been a constant motivation and distraction. It is my friend and colleague at MITRE, Russell Leighton, to whom I *still* turn to when I simply can't figure something out. Now at WDI all sorts of new possibilities are arising, in large part because of the stewardship of Joshua Wachman.

Probably the greatest part of moving back to Los Angeles is living near my parents and in-laws again. The Mercers, when they weren't trotting around the world, were gladly baby sitting Christopher, taking us camping, feeding the three of us Sunday brunch, and generally providing warmth and support of a close knit family.

I am humbled by the task of trying to sum up the contributions that my parents have made to my education, even just in the last few short years. The superficialities: helping baby sit, the dinners, help with proofreading, come readily to mind. But it's the interest in learning and education instilled by Mary Jane, and the serenity, strength, and importance of humility instilled by Paul, that, to use my mother's words, "just blows my mind."

And last, but most definitely not least, my family. Without the affection, tolerance, good humor, support (both emotional and financial), and abundant and unconditional love that I received from my wife Annette, I would never have been able to begin, let alone finish, this endeavor. And finally our son Christopher

who, in two short years, has had as profound an influence on my thinking as the preceding almost three decades in academics.

Abstract of the Dissertation

Modeling, Simulation, and Control of Two-Legged Walking

by

Alexis Paul Wieland
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 1994
Professor Walter Karplus, Chair

This dissertation describes the modeling, simulation, and control of a two-legged two-dimensional walking robot. A mathematical model of walking was derived, a computer simulation of that model was created, target trajectories for the joints during walking were developed, and a control system capable of causing the walker to follow the trajectory was produced.

Because of the veracity of the model and the resulting computer simulation in this study, details of walking that have previously been overlooked became readily apparent, some of which significantly affected the resulting walking motion. The results of these simulations showed that the simplifying assumptions made by past studies about the walking motion and its eventual control were not valid for step lengths and speeds of a normal walking gait. Based on these findings, new target trajectories based on the consistent motion of the walker's body, instead of the standard inverted pendulum model, were developed.

The eventual successful conclusion of this study was a simulated two-legged robot that could produce a sustained walking motion with step lengths and speeds comparable to a normal walking gait. Unique contributions of this study include a better understanding of what is important to a realistic computer simulation of walking, particularly regarding the effects of impacts to the motion, and a new target walking trajectory, needed for the larger joint angles and higher walking speeds found in the present study.

CHAPTER 1

Introduction

> Despite excellence in using our own legs for locomotion, we are still at a primitive stage in understanding the control principles that underlie walking and running.
>
> M. Raibert, [Rai86]

This dissertation describes the modeling, computer simulation, and control of a two-dimensional two-legged walking robot. At the beginning of the study a mathematical model of a roughly anthropomorphic walking robot is derived from first principles. The model, as it is presented herein, contains details of mass, physical extent, friction, and impacts which have been ignored or simplified in the past. A computer simulation of that model is then developed – a simulation which must take into account the large number of possible constraint states, the underlying stiff differential equations, and the often intricate interactions which resulted from sliding and impacts in the mathematical model. Using this computer simulation as a testbed, target walking trajectories and controllers capable of creating those target trajectories are then developed. In the end this study was successful at creating a walker that could produce a sustained walking motion with a step length and speed comparable to a normal human walking gait.

1.1 Outline of Work

The work in this study can be divided into four logically separate parts: (1) developing a mathematical model of walking, (2) simulating that model on a computer, (3) finding appropriate target trajectories for the walker's joints to follow, and (4) controlling the walker so that its joints follow the target trajectories. This same four-part structure is reflected in this dissertation itself. Each of the four tasks, and the contents of the chapter describing it, is briefly mentioned below.

1.1.1 Walking Model

The walking robot, or *walker*, is composed of seven segments connected by six revolute joints, shown in Figure 1.1. Each segment is modeled as having mass and physical extent. Each of the joints can be actuated, all have friction retarding their motion, and each has a limited range of motion bounded by inelastic stops. The walker's two feet can be lifted up from and set down upon the ground at arbitrary times, and, given enough tangential force, will slide.

The motion of walker is described by differential equations. Different sets of constraints restrict the motion of the walker based on what, if any, part of each foot is on the ground, whether the joints are locked, and whether the feet are sliding. There are 3,136 different constraint states possible, each with related but different equations of motion.

Chapter 2 describes the equations for the motion of the walker and for the impacts, and how these equations are reconfigured to model the possible constraint states.

1.1.2 Simulating Walking

Using a set of differential equations to compute the motion of a system is a standard task in scientific computing, but the properties inherent to walking complicated this task for the current study. The system of equations describing the motion was stiff, the constrained equations were unstable, the stabilization of the constraints tended to obscure the forces used to determine when constraints were to be changed, and, impacts can interact in complex ways which, if not handled carefully, produce paradoxical behavior.

Chapter 3 describes how the motion of the walker was simulated on a computer, how the difficulties mentioned above were handled, and how efficiency and accuracy were maintained.

1.1.3 Target Trajectories

In order to make the simulated walking robot walk, it was necessary to specify the desired path for the robot's joints to follow through the step, i.e., the target trajectory. Traditionally, walking trajectories have been developed based on the assumption that the motion of the walker was approximately that of an inverted pendulum. However this study demonstrates that the inverted pendulum was not a good approach, particularly for the relatively long steps and fast velocities that this study considered. Instead, a trajectory based on the assumption that

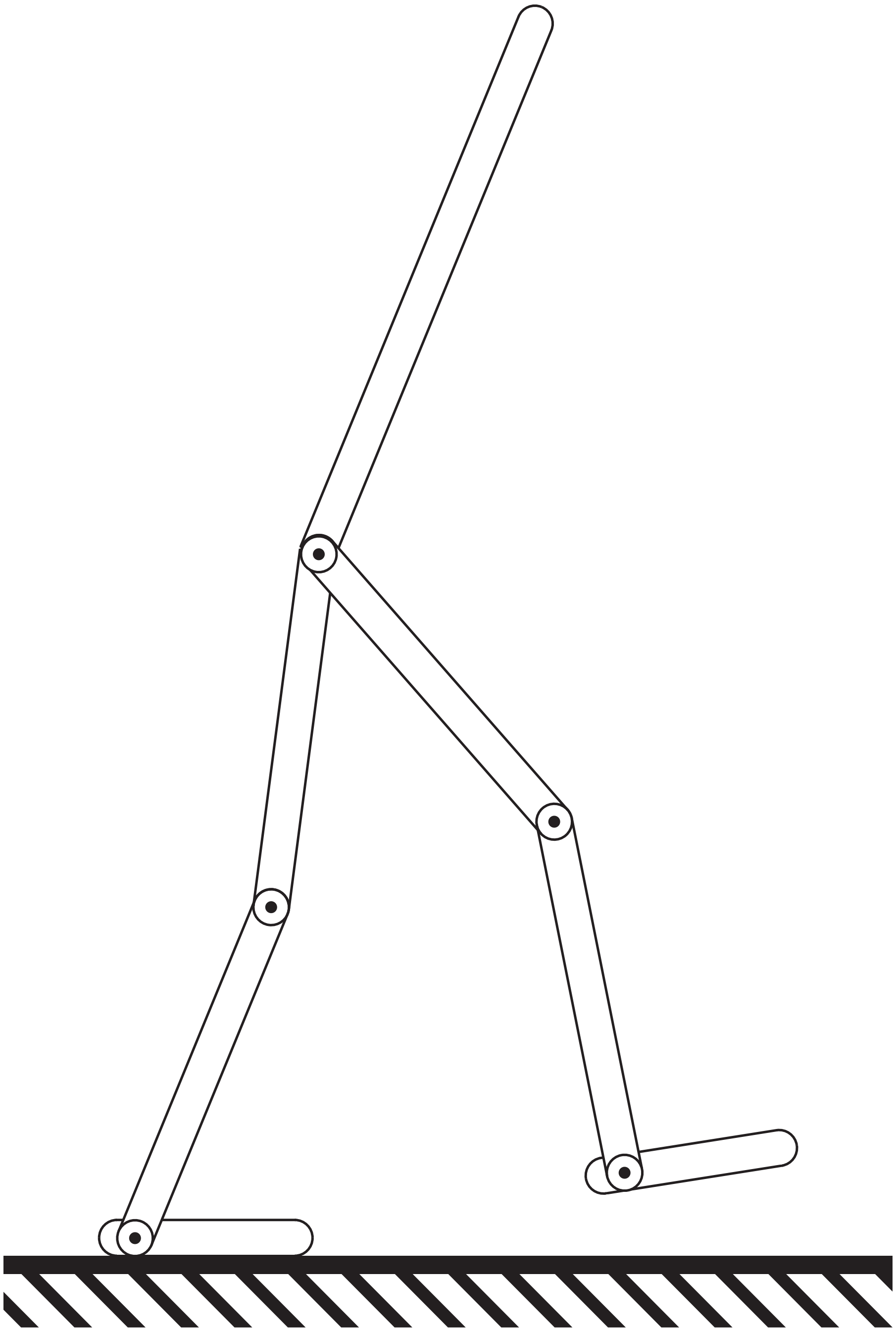

Figure 1.1: *Physical model of the* walker

the walker's body should move at a roughly constant horizontal velocity was developed.

Chapter 4 describes trajectories based on both the standard inverted pendulum model of walking and the constant velocity model developed here.

1.1.4 Controlling Walking

Given the simulation and the target trajectory, the remaining task was to find a controller capable of prescribing the torques to apply to the joints in order to cause the walker to follow the target trajectory. This study used a modified PID controller.

Chapter 5 describes PID control and the modifications that were used in this study. It also describes a neural network that was developed which could produce the same control signals as the modified PID controller and was thus also able to control the walker.

1.1.5 Conclusions

Chapter 6 presents the results and major conclusions of the study, as well as areas for future study.

1.2 Past Work

There has been an abundance of work related to walking, considering the problem from every perspective. Certainly the contrast between the apparent ease with which animals walk and the difficulties encountered in the studies of walking makes it an intriguing area of research. The following is an overview of some of the more relevant past studies.

Past analytic studies on the walking motion have generally assumed that the motion of the walker could be adequately modeled as an inverted pendulum [MS84, SMM91]. The study by Miura and Shimoyama, [MS84], is particularly of interest since it involves one of the few dynamically stable walkers that have been developed.

Other past studies have considered stable and quasi-stable walking gaits, e.g., [FS90]. There have also been studies that focused on devices for handicapped people [VBS90] and on many-legged walking devices [Bro89, KTT88, Don87]. Further studies have included the work on passive walking by McGeer at Cornell, [McG90] and the hopping robots of Raibert, [RJC84, Rai86, Rai90].

Additionally there have been some recent studies into the effects of impacts on walking, [ZH84, Zhe87], and how impacts can lead to the onset of chaos in an otherwise linear system [SS89]. Similarly, the effects of constraints and closed kinematic loops on walking gaits have been studied, [Hur92].

Also related to this study are treatises on equations modeling walking, [Wit77, SMM91], and methods for making resulting simulations efficient, [OO81, OO86, Fea87].

Probably the study closest to the one presented here was a recent monograph based on a dissertation by Manko at Carnegie-Mellon University, [Man92], describing a general simulation environment for studying multi-legged walking. While Manko's study included compliance, something *not* included here, this was necessary because of the minimal attention which his study paid to impacts.

An interesting, but only peripherally related study, was on the use of genetic algorithms to "evolve" walkers, [deG89, deG90]. Unfortunately the model of walking used was not realistic, as it assumed that the joints immediately went to and stayed at whatever angle was requested.

CHAPTER 2

Modeling

At the core of this study is a computer program that simulates an idealized model of a two-dimensional two-legged walking robot or walker. While this model is intended to be reasonable and similar to both anthropomorphic models and past walking robots, it does not precisely model either. Instead the model described here is intended to provide the important features of walking in an idealized and controlled framework.

While there is no physical counterpart to the model or the subsequent computer simulation, considerable effort was spent in making the model correspond to an idealized walking robot. The resulting model includes details of mass, physical extent, friction, impacts, and sliding which past studies have either simplified or ignored entirely. In general these past simplifications were made not only to ease the development of the simulation but also, and more significantly, to make the creation of controllers pragmatically possible.

While a simplified set of equations describing the walker is presented in Chapter 5, the general goal of the current chapter is to present the complete model. This model contains many of the nonlinearities and other attributes that make analyzing and controlling walking difficult. As a result, this model and the resulting simulation are well suited to serve as a testbed for control methods.

The remainder of the current chapter is arranged as follows: Section 2.1 is an introduction to and overview of related past work on modeling. Section 2.2 outlines the idealized model of walking considered by this study, and in Section 2.3 the chapter ends by discussing the deficiencies of the model.

2.1 Previous Work

The practice of developing mathematical models for walking and complex physical systems in general, and then solving the resulting equations to determine the resulting motion, has a long and rich tradition that stretches back hundreds of years. Certainly the basis for what will be described in the remainder of this chapter would have been familiar centuries ago to the intellectual giants who founded the field: Newton, Lagrange, and Euler. But the availability of increasingly more powerful computers has made it possible to derive ever more complex equations and, equally importantly, made it possible to simulate the resulting complex equations in a reasonable amount of time. This increased ability has caused a flurry of work relevant to this study.

The derivation of the mathematical model of two-legged walking discussed in this chapter and the simulation of that mathematical model, presented in the next, were not intended to break new ground. Instead, this work forms a solid basis on which the remainder of the research was built. The only intended "new" component is the degree of completeness, efficiency, and flexibility involved. It was both a surprise and a source of consternation to discover to what degree this kind of modeling and the subsequent simulation remain an active area of research.

First let us briefly survey the relevance of past work, providing both a background and a context for the following discussion.

2.1.1 Dynamic Formulation

A variety of techniques exist for deriving a system of equations which will describe the dynamics of the walker. With few exceptions, the techniques all produce functionally equivalent results, that is, the resulting equations describe the same motion. These techniques are distinguished from one another by the simplicity and extensibility of the derivation and by the efficiency and flexibility of the resulting equations. As will be discussed in Section 2.2.1, the equations that were used to govern the motion of the walker simulation were derived from *Lagrangian Dynamics* because of the resulting mixture of flexibility, extensibility, and relative efficiency. This mixture was needed to efficiently accommodate the walker's large number of possible constraint states. Some alternate methods of dynamic formulation are briefly described below along with the principal reasons why they were not used.

A second common technique of formulating the equations of motion is based on *Newton-Euler Dynamics*, see for example [AS86]. One distinguishing feature of the equations derived from Newton-Euler Dynamics, as compared to those based on Lagrangian Dynamics, is that Newton-Euler Dynamics produces *all* of the forces and moments involved in the motion, both internal and external, whereas Lagrangian Dynamics computes only the external forces. Since internal forces are generally not explicitly needed, the effort required to produce them is unnecessary. A *recursive*, and therefore computationally attractive, form of the

Newton-Euler algorithm also exists for systems with tree structures, [LWP80], but these could not be used in the not uncommon case where the walker has more than one foot's contact point on the ground, causing the walker to not have a tree structure.

Equations of motion can also be derived from *D'Alembert's Principle*, which briefly states that a system is at equilibrium when all of the forces are balanced, see for example [Can67]. The main difficulty with this approach is that it can be cumbersome or even impractical to use in formulating the equations of motion because of the difficulty of determining the centrifugal and coriolis forces for complex systems. Often in these cases such forces are assumed to be insignificant and are left out of the formulation, but for the walker, while these forces are indeed difficult to determine, they can strongly influence the resulting motion and therefore could not reasonably be ignored.

Yet another approach that has recently become popular is based on *Kane's Dynamics*, see [KL83, KL85, Lee87], an approach which introduces *generalized speeds* to produce a set of concise first-order equations for the motion. This is in marked contrast to all of the other methods that have been described, each of which result in a system of second-order equations. The disadvantage of Kane's approach, as far as this study is concerned, is the amount of effort required to formulate the equations and the specificity of the results. As will be explained below, because there are more than a thousand different constraint states it was impractical to derive a distinct optimized system of equations for each of these states.

An important consideration in formulating the dynamic equations for walking is the possibility of having *closed kinematic chains* (or loops), Figure 2.1. Closed kinematic chains for the walker occur whenever more than one foot (or to be precise, more than one contact point) is on the ground at one time. Such multiple contacts restrict the possible motion of the walker, and restrictions are modeled as holonomic constraints[1] and traditionally solved with the equations of motion using the calculus of variations, see Section 2.2.1.

2.2 Physical Model of Walking

This section provides an overview of the idealized physical model of walking that forms the basis for the simulation and of the way such an idealized model was used to derive the equations of motion. Appendix A contains a sample of the

[1] That is, the constraints are on the position, as opposed to the velocity, of the system.

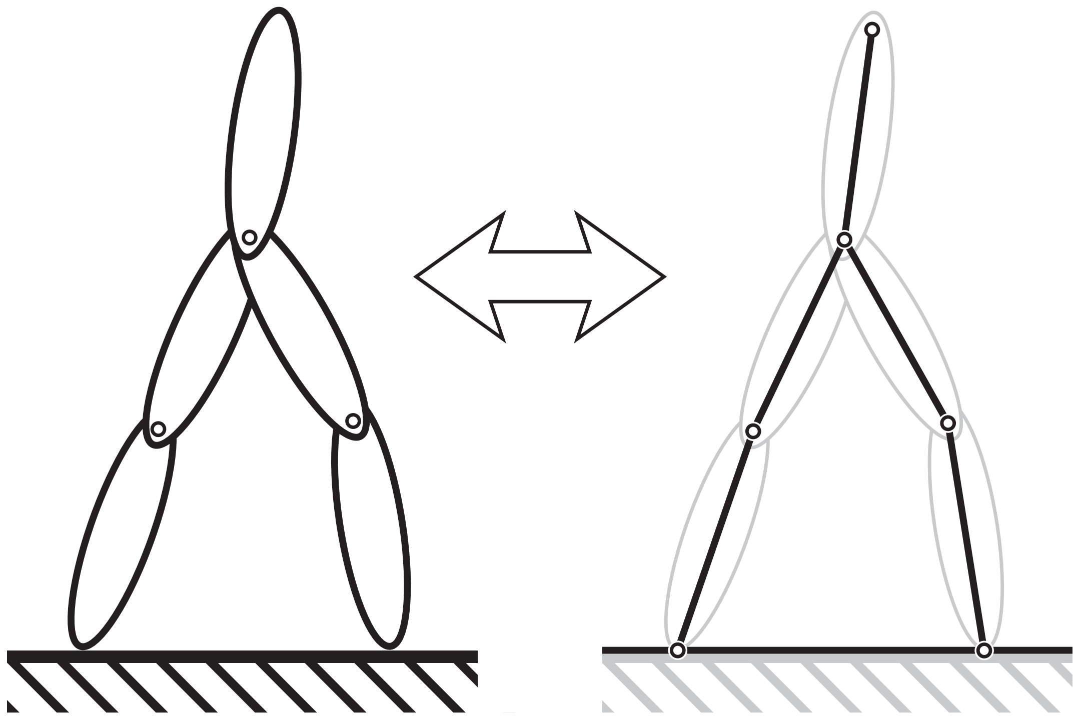

Figure 2.1: *Closed kinematic chain.*

resulting equations and Appendix B contains the values of the parameters used in the simulation such as the length of the segments and the friction coefficients.

The walker consists of seven segments: i.e., a body and two legs each composed of a thigh, a calf, and a foot, Figures 1.1 and 2.2. Each segment is assumed to be an idealized inflexible thin rod; that is, the segments neither bend nor compress and the inertia of the segment is computed by assuming that the mass is uniformly distributed along an infinitely thin line down its center. There are six actuated joints corresponding to two ankles, two knees, and two hips. All six of these joints are revolute (or hinge-type), contain bearing friction and have limited ranges of motion constrained by inelastic stops. All except the ankle joints connect together the ends of two segments; the ankles connect the end of the calf to a point on the foot that is $\frac{1}{10}^{\text{th}}$ the way from the heel to the toe. The feet may make and break contact with the ground at arbitrary times and, if there is sufficient tangential force, will slide. The feet experience both static and dynamic friction. That is, as long as the foot is stationary there is a friction force that works to keep the foot stationary, and when the foot is sliding there is a different (generally smaller) frictional force retarding the motion.

The two-dimensional walker has nine degrees-of-freedom (DOF). In addition to the six DOF that come from the six joints, the walker has three DOF (x, y and θ) which determine its position in the plane relative to the ground. At each step of the simulation, one of the feet is designated as "$foot_1$", and the walker's horizontal, vertical, and rotational position are specified in reference to that foot's toe, Figure 2.2.[2]

These nine DOF form a nine-dimensional vector of generalized coordinates $\mathbf{q}$ that are used to specify the position of the walker

$$\mathbf{q} \quad = \quad \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \\ q_9 \end{pmatrix} \quad = \quad \begin{pmatrix} x \\ y \\ \theta_{\text{foot1}} \\ \theta_{\text{hip1}} \\ \theta_{\text{hip2}} \\ \theta_{\text{knee1}} \\ \theta_{\text{knee2}} \\ \theta_{\text{ankle1}} \\ \theta_{\text{ankle2}} \end{pmatrix}$$

The *motion* of the walker is described by the corresponding velocity vector $\dot{\mathbf{q}}$. The *state* of the walker is defined by its position $\mathbf{q}$, velocity $\dot{\mathbf{q}}$, and the walker's

[2]The assignment of $foot_1$ is made so as to simplify the equations of motion or impulsive forces by reducing the size of the associated array, see Section 2.2.1.

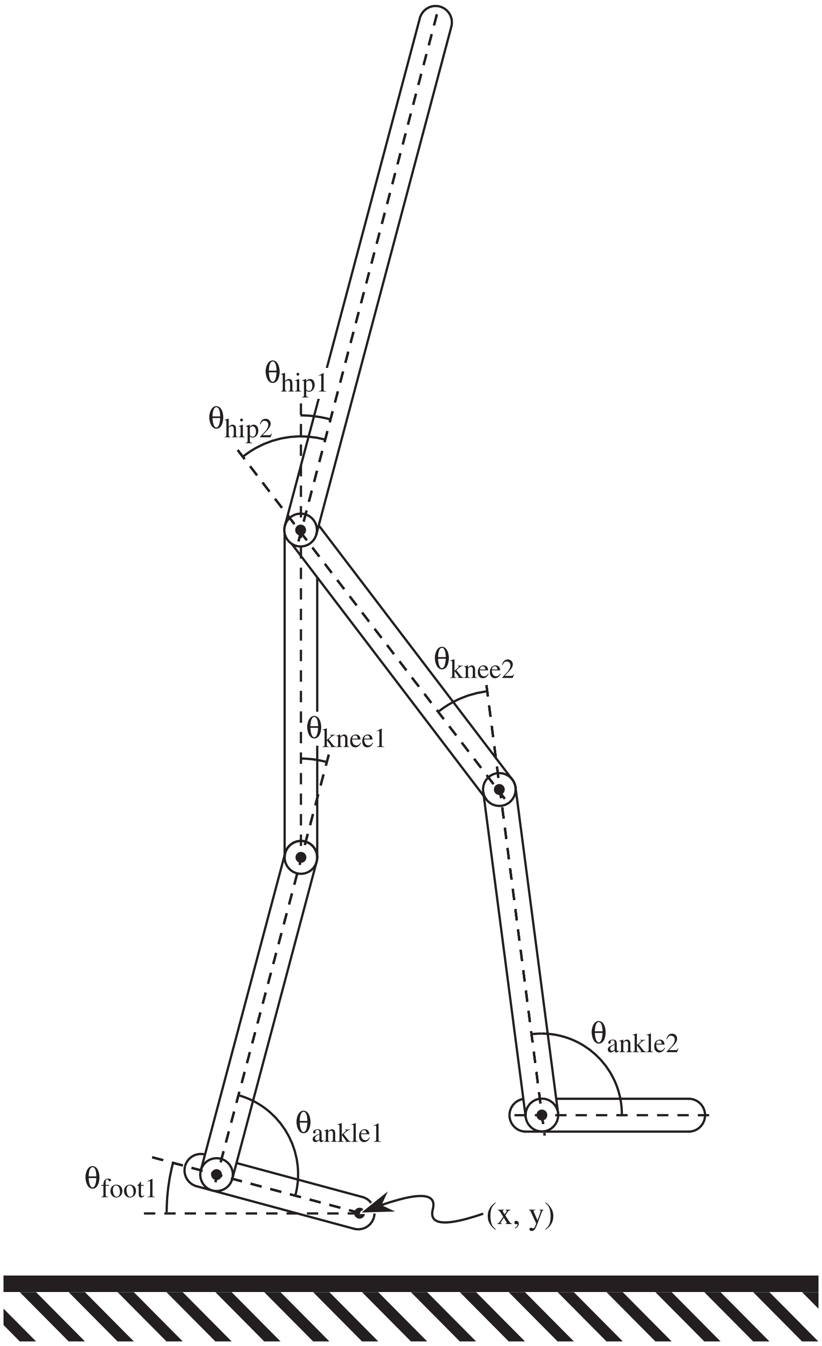

Figure 2.2: *The nine generalized coordinates that specify the walker's location.*

1	heel up	toe up	—
2	heel up	toe down	sliding
3	heel up	toe down	stationary
4	heel down	toe up	sliding
5	heel down	toe up	stationary
6	heel down	toe down	sliding
7	heel down	toe down	stationary

Table 2.1: *Seven constraint states for each foot*

constraint state **c**, a boolean vector describing which of the possible constraints are currently in effect.

Feet are unique in the model in that they are allowed to have sustained contact with the ground. Feet are considered to have precisely two contact points, one located at the toe and the other at the heel. Without loss of generality, the forces and accelerations that are derived in modeling the motion of the foot are considered to be exerted only through these two points.

A foot may be in midair or have one or both of its contact points touching the ground. If a foot is touching the ground, then it is additionally described by whether it is stationary or sliding. Either foot can therefore take on one of seven distinct constraint states shown in Table 2.1. The condition of the two feet together therefore places the walker in one of $7^2 = 49$ constraint states, of which 28 are distinct after considering symmetry.[3] Contacts with the ground are modeled as constraints on the possible motion and therefore affect the form of the equations of motion.

In addition to the constraints imposed by the feet, there are constraints that can be imposed by the six joints. Each of the walker's joints has a limited range of motion, with inelastic "stops" at the extremes. For example, the knees are restricted to a range of 0° to -135°, Figure 2.3. If the robot kicks forward hitting the knee's "max-stop" at 0°, that momentum is transferred to the remainder of

[3] A constraint state that is a mirror image of another constraint state whose equations of motion are known can be addressed by simply changing the assignment of $foot_1$.

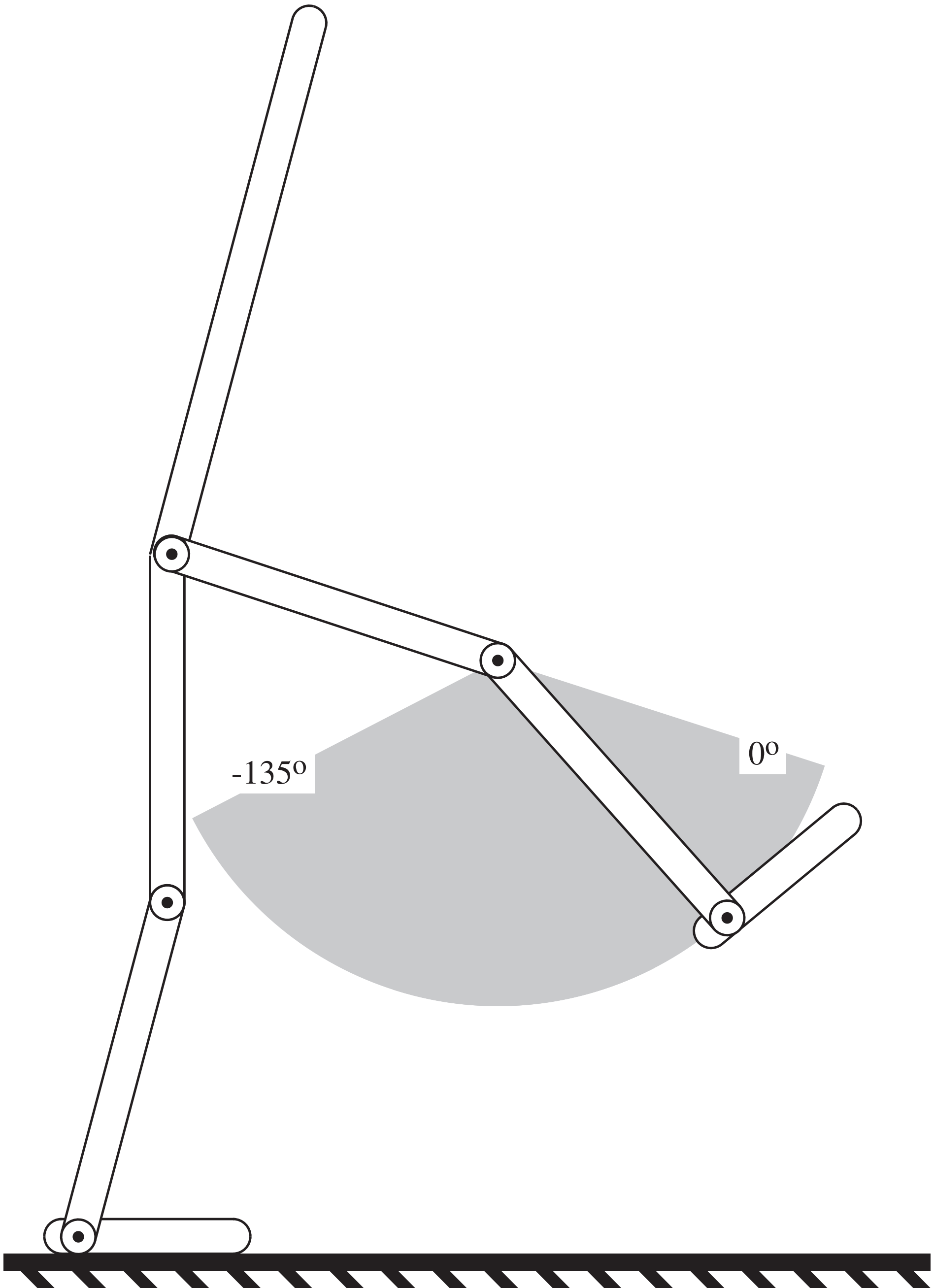

Figure 2.3: *Knee joints are limited to the range 0° to -135°.*

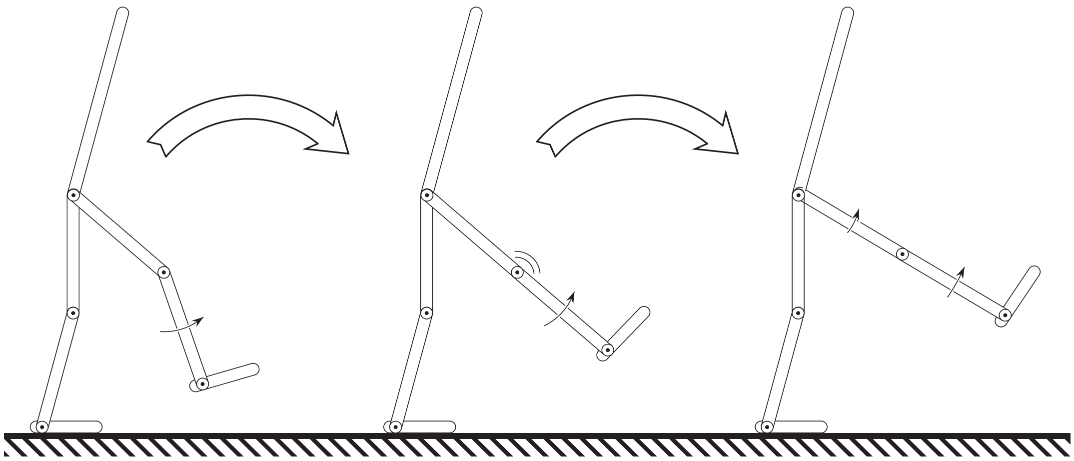

Figure 2.4: *Transfer of momentum from a knee hitting its max-stop.*

the walker and to the ground, with the majority of the momentum in this case going into the corresponding thigh, Figure 2.4. The knee will continue to be locked at the max-stop as long as the torque at the knee pushing against the stop is greater than or equal to zero.

Since each of the joints can be in one of two states, either locked (at either the max-stop or the min-stop) or unlocked, the six joints taken together place the walker in one of $2^6 = 64$ states. Combined with the possible constraint states of the feet, the walker can be in any of $7^2 \times 2^6 = 3{,}136$ constraint states, of which 1,596 are symmetrically distinct. The motion of the walker in each of these 1,596 distinct constraint states is described by a different set of differential equations.

2.2.1 Equations of motion

Because of the large number of constraint states, it was more reasonable to develop a single large software system that could be adapted at run-time to accommodate any set of constraints than to develop optimized routines for each of the constraint states individually. Development of the single system was achieved by using Lagrangian Mechanics with constraints being added using the calculus of variations.

The equations of motion for the walker were derived from the Lagrangian formulation of the mechanics of the walker. That is, second-order differential equations were derived by symbolic manipulation of Lagrange's equation of motion

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_k} - \frac{\partial T}{\partial q_k} = Q_k \tag{2.1}$$

where T is the kinetic energy of the walker, q_k is the k^{th} generalized coordinate (e.g., a joint angle), and Q_k is the generalized force exerted at the generalized coordinate q_k.

Because the robot is composed entirely of idealized thin rods, the kinetic energy T takes a particularly simple form. The kinetic energy for the entire walker is simply the sum of the kinetic energy of each segment

$$T = \sum_{s \in \mathcal{S}} T_s \tag{2.2}$$

where $\mathcal{S}$ is the set of the walker's seven segments. The kinetic energy of each segment T_s has components from both the rotation and translation of the segment, which for an idealized thin rod simply equals

$$T_s = \frac{m_s}{6}\left(\mathbf{v}_{s,1}\cdot\mathbf{v}_{s,1} + \mathbf{v}_{s,1}\cdot\mathbf{v}_{s,2} + \mathbf{v}_{s,2}\cdot\mathbf{v}_{s,2}\right) \tag{2.3}$$

where $\mathbf{v}_{s,1}$ and $\mathbf{v}_{s,2}$ are the velocities of the two ends of segment s.

Q_k can be further broken up into conservative, nonconservative, and actuation forces and torques:

$$Q_k = -\frac{\partial V}{\partial q_k} + fric_k + u_k \tag{2.4}$$

where the conservative forces (e.g., gravity) equal $-\frac{\partial V}{\partial q_k}$ where V is the potential energy for the walker. Nonconservative forces (i.e., friction) are denoted $fric_k$ and the u_k are the actuations applied to the joints.

There are two distinct kinds of friction in the system, bearing friction in the joints, $fric_{\mathrm{b}}$, and Coulomb friction between the feet and the ground, $fric_{\mathrm{C}}$. The details of these friction forces are discussed in Section 2.2.5 below. Since joint angles correspond directly to generalized coordinates, the friction in a joint is manifest only in the corresponding Q_k. By contrast, the Coulomb friction on the feet $fric_{\mathrm{C}}$ is potentially exerted at the four contact points, of which only the toe of $foot_1$ corresponds directly to a generalized coordinate. Instead, these friction forces must be distributed throughout the Q_k. The component of Q_k coming from Coulomb friction is given by

$$Q_k(fric_{\mathrm{C}}) = \sum_{i \in \mathcal{C}} \left(fric_i \frac{\partial x_i}{\partial q_k} \right) \tag{2.5}$$

where $\mathcal{C}$ is the set of the four contact points, $fric_i$ is the Coulomb friction force at the i^{th} contact point, and x_i is the location of the i^{th} contact point.

These equations of motion (as well as the subsequent equations for impacts) were derived and solved by the symbolic manipulation computer programs MapleV [CGG91] and Mathematica [Wol91]. These programs allowed the equations to be entered in a form similar to the above equations. Terms in the equations, such as the kinetic energy T, were defined in terms of variables that made their formulation straightforward, e.g., $\mathbf{v}_{s,1}$ in Equation 2.3. These variables were then defined by other variables that eventually were defined in terms of the generalized coordinates. MapleV was used to generate optimized computer code, in the computer language `C`, to implement the derived equations. Samples of the resulting equations are presented in Appendix A.

The motion of the walker is almost always constrained by having one or more of the walker's four feet-contact points on the ground. There are fourteen possible constraints that can affect the walker: six come from the six joints which can be either free or locked (at either their maximum or minimum values), and the remaining eight come from the feet: the x or y position of either the heel or toe of either foot may be fixed.[4] The constraints, denoted ψ_j, are all autonomous and holonomic, that is they can be written in the form

$$\psi_j(\mathbf{q}) \;=\; 0. \tag{2.6}$$

In particular, note that these constraint do not explicitly depend on time t or the velocities $\dot{\mathbf{q}}$.

Constraints are incorporated into the equations of motion by using the calculus of variations method of undetermined, or Lagrangian, multipliers λ_j. The equations of motion for the walker are thus derived from

$$\frac{d}{dt}\frac{\partial T}{\partial \dot{q}_k} - \frac{\partial T}{\partial q_k} \;=\; Q_k + \sum_{j=1}^{M} \lambda_j \frac{\partial \psi_j}{\partial q_k} \tag{2.7}$$

The constraints restrict the motion of $\mathbf{q}$ to lie along a surface defined by $\psi_j(\mathbf{q})=0$. The resulting λ_j are a measure of the force required to maintain the corresponding constraint. Thus the λ_j are the forces on the constraints and, when properly formulated, give the reaction forces needed to determine when a constraint should be released.

As an example, a foot starts to lift off the ground when the y component of the foot's force (the normal force) changes from pushing down on the ground to zero. If the constraint were to remain as λ_j passed through zero, then the ground would be exerting a force to hold the foot down onto the ground. Since

[4]The y component is locked whenever there is contact with the ground. The x component is locked when the foot is on the ground and is not sliding.

$\lambda_j = 0$ when the release occurs and the foot's acceleration is proportional to the force, the acceleration will be zero at the instant of the release and the foot will then proceed to accelerate smoothly away from the floor. This can be seen, for example, in the sixth plot in Figure 2.6 on page 22.

Equation 2.7 produces N equations in $N + M$ unknowns, where $N = 9$ is the number of generalized coordinates and M is the number of constraints currently in effect. Traditionally the additional M equations are provided by differentiating each of the M constraint equations

$$\frac{d^2\psi_j}{dt^2} \;=\; 0. \tag{2.8}$$

Baumgarte showed that the resulting system of equations can be numerically unstable in the sense of Lyapunov [Bau72], and suggested instead that the additional equations be provided by the equation of a damped oscillator

$$\frac{d^2\psi_j}{dt^2} + 2\alpha\frac{d\psi_j}{dt} + \beta^2\psi_j \;=\; 0, \qquad \alpha > 0 \tag{2.9}$$

where α and β determine the form of the oscillations. Such equations were used in this study, and, following Baumgarte, the parameters were set so $\beta = \alpha$, forming an aperiodic, or critically-damped, oscillation. This effectively creates critically damped springs that attempt to pull constrained parameters back to their proper constrained values. Finally, α was chosen to be large so that errors would be removed quickly, given the restriction that $\alpha \geq 1/\Delta t$ creates a high frequency response that would again make the equations unstable.

At each instant of time, and for a given set of constraints, there is a linear relationship between the generalized forces and the accelerations of the generalized coordinates.[5] That is, Equations 2.7 and 2.9 can be expressed together by the single equation

$$\mathbf{A}_m(\mathbf{q}, \dot{\mathbf{q}}; \mathbf{c}) \cdot \mathbf{x}_m \;=\; \mathbf{b}_m(\mathbf{q}, \dot{\mathbf{q}}; \mathbf{c}). \tag{2.10}$$

where $\mathbf{A}_m$ is an array, the vector of unknowns $\mathbf{x}_m$ is given by

$$\mathbf{x}_m \;=\; \begin{pmatrix} \ddot{q}_1 \\ \vdots \\ \ddot{q}_9 \\ --- \\ \lambda_1 \\ \vdots \\ \lambda_M \end{pmatrix} \tag{2.11}$$

[5]This does *not* mean that the equations of motion for the walker are linear.

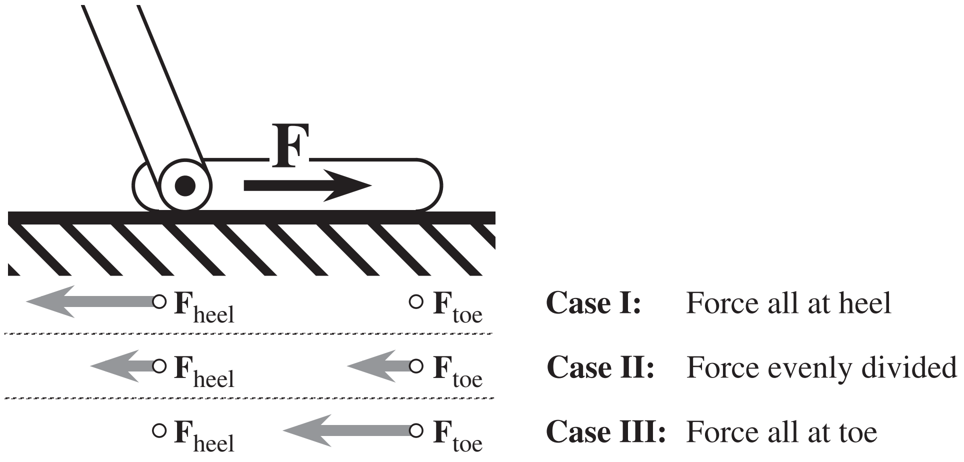

Figure 2.5: *Pairs of reaction forces that produce equivalent motion*

where the $\ddot{q}_i$ are the accelerations of the generalized coordinates and the M different λ_j's are the Lagrangian multipliers (corresponding to the unknown constraint forces), and where

$$\mathbf{b}_m = \begin{pmatrix} Q_1 \\ \vdots \\ Q_9 \\ --- \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{2.12}$$

Note again that these equations are autonomous, that is, $\mathbf{A}_m$ and $\mathbf{b}_m$ do not explicitly depend on the variable t. It is also important to appreciate the dependence of this expression on $\mathbf{c}$. The constraint vector $\mathbf{c}$ determines not only the values but also the sizes of $\mathbf{A}_m$, $\mathbf{x}_m$, and $\mathbf{b}_m$, since it determines the number of active constraints M.

Unfortunately it is not unusual for $\mathbf{A}_m$ to be either singular or nearly singular, that is, for there not to be a distinct solution to Equation 2.10. As an example, if a foot has both its heel and its toe touching the ground, there is an ambiguity as to what fraction of the tangential reaction force is experienced by the heel and what fraction is experienced by the toe, Figure 2.5. Since the foot is not allowed to expand or contract, the resulting motion of the foot is not affected by which one of the infinite number of ways the force is divided. This ambiguity manifests itself as a singularity in $\mathbf{A}_m$.[6] The approach used to solve this possibly

[6]This and a small number of other easily detected situations that always result in a singu-

singular system of linear equations is based on computing the pseudo-inverse of A using singular value decomposition, and is discussed in the following chapter on simulation of the equations of motion in Section 3.6.

2.2.2 Impacts

Impacts occur when a foot touches the ground or when a joint reaches a stop, and are modeled as being perfectly inelastic; that is, there is no bounce or recoil. Mathematically, impacts are modeled by an instantaneous *impulsive force* that precisely cancels the components of the velocity moving into the constraint. Impacts typically also result in a discontinuous change in the velocities of all the other generalized coordinates.

The equations for modeling impacts are derived in a manner similar to the equations of motion, using Lagrange's equation for impulsive forces. Impulsive forces are assumed to occur in an infinitesimally short time period, causing the forces from gravity, actuation, and friction to become negligible. Lagrange's equation for impulsive forces is

$$\Delta\left(\frac{\partial T}{\partial \dot{q}_k}\right) \;=\; P_k \tag{2.13}$$

where the P_k are the generalized impulsive forces. Constraints are incorporated in a manner identical to that used for the equations of motion

$$\Delta\left(\frac{\partial T}{\partial \dot{q}_k}\right) \;=\; P_k + \sum_{j=1}^{M} \lambda_j \frac{\partial \psi_j}{\partial q_k} \tag{2.14}$$

where the additional M equations are provided by differentiating the constraints[7]

$$\frac{d\psi_j}{dt} \;=\; 0. \tag{2.15}$$

At each instant of time and for a given set of constraints there is a linear relationship between the changes in velocities and the impulsive forces. This is directly analogous to Equation 2.10 above. Equations 2.14 and 2.15 can therefore be expressed as

$$\mathbf{A}_p \cdot \mathbf{x}_p \;=\; \mathbf{b}_p \tag{2.16}$$

larity are handled before the system of linear equations are solved. In this case the convention is that all of the force is experienced at the toe.

[7]Since these equations are used to model an instantaneous change in velocities instead of a continuous motion, the concerns about instability that led to Equation 2.9 do not apply here.

where $\mathbf{A}_p$ is an array, the vector $\mathbf{x}_p$ is given by

$$\mathbf{x}_p = \begin{pmatrix} \Delta\dot{q}_1 \\ \vdots \\ \Delta\dot{q}_9 \\ --- \\ \lambda_1 \\ \vdots \\ \lambda_M \end{pmatrix} \tag{2.17}$$

where the $\Delta\dot{q}_i$ are the changes in the velocities of the generalized coordinates as a result of the impact, the λ_j's are the Lagrangian multipliers corresponding to the impulsive constraint forces, and where

$$\mathbf{b}_p = \begin{pmatrix} P_1 \\ \vdots \\ P_9 \\ --- \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.18}$$

in which the P_k are the impulsive forces applied to the coordinates q_k.

In general we only know a fraction of the P_k that make up $\mathbf{b}_p$, but for the remainder of the generalized coordinates we know the $\Delta\dot{q}_k$. An impact is used to change the velocity of the generalized coordinate involved in the impact to zero; that is, for an impact at time t_{impact} the generalized coordinate q_k has $\Delta\dot{q}_k = -\dot{q}_k(t_{\text{impact}})$.

Similarly, all generalized coordinates whose positions are locked have $\Delta\dot{q}_k = -\dot{q}_k(t_{\text{impact}})$.[8] For all of the remaining generalized coordinates – coordinates that are neither constrained nor being impacted – we know that they exert no impulsive force, $P_k = 0$.

Since in Equation 2.16 part of $\mathbf{x}_p$ and part of $\mathbf{b}_p$ are unknown, it is necessary to rearrange the terms to form a new system of equations

$$\tilde{\mathbf{A}}_p \cdot \tilde{\mathbf{x}}_p = \tilde{\mathbf{b}}_p. \tag{2.19}$$

such that $\tilde{\mathbf{x}}_p$ contains all of the unknown terms and all of the elements of $\tilde{\mathbf{b}}_p$ are known. Once this newly rearranged linear system of equations has been formed,

[8] Because of the stabilization of the constraints in the equations of motion, Equation 2.9, generalized coordinates q_k that are locked will still generally have $\dot{q}_k \neq 0$. Since the impact can make these velocities inappropriate, the impact is used to reset them to zero.

it is solved given the same caveat that $\tilde{\mathbf{A}}_p$ may be singular or nearly singular. The result $\tilde{\mathbf{x}}_p$ of solving Equation 2.19 are the unknown $\Delta\dot{q}_k$'s and P_k's. That is, for all the $\Delta\dot{q}_k$ that are known, solving Equation 2.19 produces the corresponding P_k's, and vice versa. If the resulting P_k's are inconsistent with the constraint state $\mathbf{c}$ then the constraints must be modified and the impact recomputed. This is discussed further in the following chapter on simulation, Section 3.5.

2.2.3 Other State Changes

Impacts are events that accompany the addition of most new constraints and are marked by discontinuous changes in $\dot{\mathbf{q}}$. But the addition of one of the constraints and the removal of all constraints occur without the need for impacts. In particular, when a sliding foot becomes stationary, a constraint is added without the need for an impact since the foot is, by the nature of the event, already stationary.

This constraint is added by moving to the time that the x velocity of the foot goes to zero and then simply changing the state of the foot to "stationary." Since the foot's velocity is already zero, there is no need to create an impact. In a similar manner, constraints are released when the corresponding constraint force goes to zero by moving the simulation to that time and then changing the constraint state.

The one constraint that is released when the corresponding constraint is *not* zero is when a stationary foot starts to slide. As discussed in Section 2.2.5, feet start to slide when the tangential force exceeds a threshold. Unlike the releasing of the other constraints, this is a breaking action that causes a discontinuous change in the acceleration of foot (and generally the rest of the walker) as a result.

It is helpful at this point to consider an example of the motion of the walker as it exhibits a number of state changes. Figure 2.6 shows plots (generated by the computer simulation described in Chapter 3) of some of the positions and some of the velocities of the walker as it transfers weight from one foot to the other.[9] In the time covered by these plots the events shown in Table 2.2 occurred. In particular, in these plots note the discontinuities in the velocities when impacts occur and compare that to the smooth transitions when constraints are released.

[9] In order to remove the discontinuities that normally arise from changes in the actuation, none of the walker's joints received any actuation during this particular simulation. As a result the walker is actually in the process of falling over.

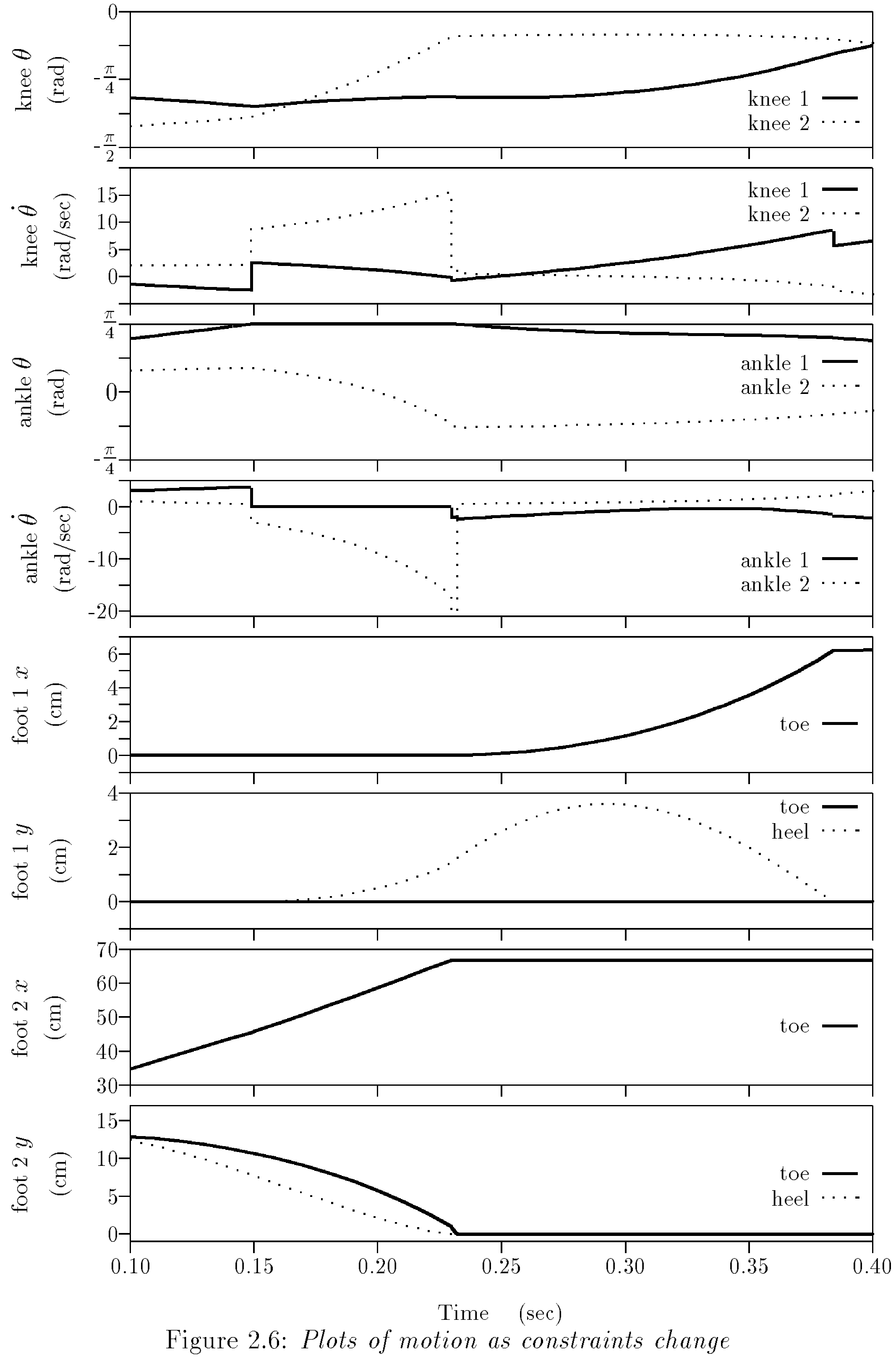

Figure 2.6: *Plots of motion as constraints change*

Time (sec)	*Event*
0.1486	Ankle 1 impacts against its max-stop causing Heel 1 to break contact with the ground
0.2297	Heel 2 makes contact with the ground causing Toe 1 to start sliding right
0.2321	Toe 2 makes contact with the ground
0.3839	Heel 1 remakes contact with the ground

Table 2.2: *Events during sample simulation*

2.2.4 Actuation

Each of the six joints of the walker can be actuated; that is, a torque can be applied to them causing the joint to open or close. Conceptually there is an idealized motor at each joint, and actuating the joint corresponds to turning that motor on. The three other generalized coordinates, x, y, and θ_{foot1}, are not actuated; that is, no additional forces can be applied to these generalized coordinates in order to control walking.

The model assumes a discrete time "computer" controller with a sampling time of 1 msec. The control signals to the robot and the resulting actuations are set at the beginning of every time step Δt and remain fixed at that value until they are set to new values Δt seconds later. This is in contrast to the rest of the simulation where constraints are created and broken, resulting in a change in the equations of motion, at arbitrary times. Thus, the walker simulation models discrete time control of a continuous time process.

The amount of actuation experienced at each joint q_k is specified (by the controller) by a corresponding control signal u_k. The control signals, and therefore the actuations, are bound in range; that is $|u_k| \leq U_k$, where U_k is the bound for that joint. This can be written for all the coordinates at once as $|\mathbf{u}| \leq \mathbf{U}$, where $U_1 = U_2 = U_3 = 0$ are for the three unactuated generalized coordinates x, y, θ_{foot1}. The implementation of these bounds are discussed in Chapter 5 and the values of these bounds are listed, with the other simulation parameters, in Appendix B.

This is a simple model of actuation. In particular there is no delay or slew-rate (the necessarily finite rate at which an actuator can change from one setting to another) associated with actuation. The instant the control signal $\mathbf{u}$ is sent to the walker, the full corresponding torque is experienced at its joints. These discontinuous changes in the torques cause the resulting joint accelerations to change discontinuously, and such discontinuities require that each distinct time step be treated separately when solving the equations of motion. This is one of the main reasons that Predictor-Corrector methods for solving the equations of motion were *not* used, as will be discussed in Chapter 3.

2.2.5 Friction and sliding

There are two distinct idealized models of friction that are used in describing the motion of the walker in this study. The six revolute joints are modeled as having bearing friction and the feet are modeled as experiencing Coulomb friction.

The bearing friction in the joints is modeled as a function of only the angular velocity of the joint. A pragmatic reason for using a friction of this form is that the Lagrangian equations of motion do not readily provide information about the forces experienced within the joints. In the model of bearing friction used in the walker simulation, the friction force is composed of two terms, the first proportional to the velocity and the second a constant force inhibiting the motion

$$fric_{\mathrm{b},i} \;=\; \mu_{\mathrm{b1},i}\,\dot{\theta}_i \;+\; \mu_{\mathrm{b2},i}\,\mathrm{sgn}\left(\dot{\theta}_i\right) \tag{2.20}$$

where $\mu_{\mathrm{b1},i}$ and $\mu_{\mathrm{b2},i}$ are friction coefficients. The bearing friction coefficients for each joint, $\mu_{\mathrm{b,i}}$, are listed in Appendix B.

The feet experience both static and dynamic Coulomb friction. Coulomb friction is a dissipative force between two bodies, in this case the feet and the ground, and is a function of the forces between them. In particular Coulomb friction is *independent* of the relative velocities or the size of the contact area. *Static* Coulomb friction acts when a foot is stationary and works to keep it stationary. *Dynamic* Coulomb friction acts when a foot is sliding and retards the motion.

Coulomb friction on a foot is based on the normal force F_n (i.e., the sum of the y components of a foot's forces against the ground) and the tangential force F_t (i.e., the sum of the x components of a foot), Figure 2.7. These are the constraint forces and are given by the appropriate λ_j in Equation 2.7.

When the foot is stationary, static Coulomb friction will keep it stationary as long as

$$\left|\frac{F_t}{F_n}\right| \;<\; \mu_s, \tag{2.21}$$

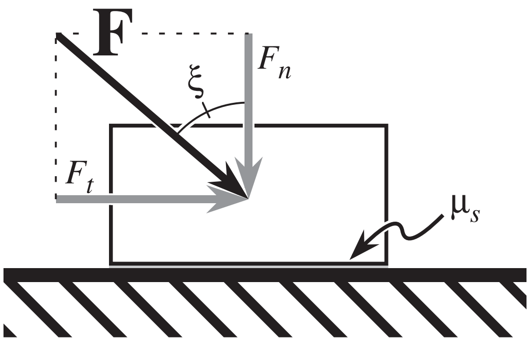

Figure 2.7: *Coulomb friction*

where F_t and F_n are the tangential and normal forces respectively and where the static friction coefficient μ_s serves as a threshold. This is equivalent to saying that the foot will remain stationary as long as the angle ξ of the force from vertical is less than a threshold, $|\tan\xi| < \mu_x$, see Figure 2.7.

At the point that this threshold *is* exceeded, the foot discontinuously switches from being stationary to sliding. Once sliding, the motion of a sliding foot is inhibited by a dynamic Coulomb friction force that is equal to

$$fric_{\mathrm{C}} = \mu_d F_n \qquad (2.22)$$

where μ_d is the dynamic friction coefficient and is in a direction opposite to the foot's motion. The foot continues to slide, regardless of what F_t now becomes, until the velocity of the sliding goes to zero. Typically this occurs when the sliding is about to reverse directions. At this point the foot locks, becoming stationary, and will not slide again until the ratio of forces once again exceeds μ_s.

Computing the value of the dynamic friction is complicated by the need to find the roots of an equation describing the normal forces on the feet. The friction force on a foot is determined by its normal force, but at the same time the normal force is affected by the foot's acceleration which is based in part on the foot's friction force. Hence it is necessary in the simulation of this friction to use an iterative method to resolve this cyclic non-linear relation and find a consistent friction force.

2.3 Deficiencies of the Model

While the walker simulation goes to considerable lengths to describe the motion of a walking biped, it is a *simulation* of walking and as such is necessarily incomplete.[10] This section discusses the most significant of these deficiencies.

A number of related deficiencies come from the lack of compliance in the system. It is assumed that the walker is composed of idealized inflexible rods, that the joints and joint actuators are not compliant, that all of the collisions are inelastic, and that there is no deformation or compliance of the ground at the points of contact. This lack of compliance can result in sudden and jarring impacts that create large transient forces and accelerations.

Modeling and analyzing (let alone controlling) compliant systems, particularly compliant systems containing closed kinematic loops, is an active research area. For example, a model of legged locomotion where the joints contain triaxial springs is in [Shi86], a static model of legged locomotion containing vertical foot-soil interaction is in [KTT88], and a model of dynamic legged locomotion with non-conservative foot-soil interactions is given in [Man92]. From a pragmatic perspective, including some form of compliance into the walker simulation would necessarily have increased the degrees of freedom, thus increasing the complexity of the equations, and resulting in a significant increase in the time required to simulate the equations of motion. The resulting increase in the degrees of freedom and corresponding increase in the basic simulation time were the principal reason that compliance was not originally designed into the walker's model. Also, since the compliant motion could have had high frequency components, there might have been a tendency toward further "stiffening" of the equations, making it more necessary to use stiff methods for solving the ODEs.

In retrospect, for the non-compliant walker described here, when there are energetic impacts, the resulting torques and accelerations can grow large precisely because there is no compliance. If the walker simulation involved a large number of energetic impacts – say for example the walker was hopping instead of walking – it would seem desirable to include some degree of compliance into the model. This would not only make the behavior more closely resemble physical systems but would also reduce the dramatic transient behavior that can occur in the non-compliant walker and thereby make the system easier to simulate.

Another simplification was in the model of actuation. In the walker simulation it is assumed that the control signal is instantaneously computed and the resulting torque instantaneously applied at the joint, and that there is therefore no delay or slew-rate associated with the actuators. This is most significant in the effect

[10] It has recently been suggested that the current propensity for simulating complex systems, as opposed to working with the actual physical objects, has led to a distorted perspective on computational intelligence and control. This provocative position is most strongly advocated by Rodney Brooks and members of his "insect" lab at M.I.T., see [Bro91].

that it has on the ability to create controllers, since delays generally have a destabilizing effect. Unfortunately more accurate models of actuation are highly dependent on the type of actuation being modeled.

A further deficiency in this model is that the floor is always flat and level, and there is some irony in this. One of the reasons that walking is considered to be an interesting area of research is that legs have the potential for traveling through terrain inaccessible to vehicles which use wheels or tracks; the standard example being climbing up a ladder. The floor was kept level and flat in this study to limit the need for feedback to the system.

In general, all of the simplifications described in this section were instituted to make the resulting simulation more efficient. Most of them are not particularly difficult to include in the simulation, and except for the unlevel ground they have all been incorporated to some degree into this software at some point in its development.

While these simplifications may be fertile areas for future research, it is believed that the walker simulation as it is herein presented does provide sufficient accuracy and "realism" to prove an acceptable testbed.

CHAPTER 3

Simulation

Given the mathematical model of walking derived in Chapter 2, the next step is to simulate that model on a computer. Using a set of differential equations to compute the motion of a system is a standard task in scientific computing, called the initial value problem, but features inherent to walking, such as the impacts that result when feet make contact with the ground, the potential interaction of approximately simultaneous impacts, sliding, and closed kinematic loops complicate this task. The present chapter discusses how the motion of the walker was simulated, and in particular how these difficulties were mitigated.

After an initial discussion of general approaches to solving the initial value problem, with an emphasis on the specific methods used here, this chapter presents a series of details of the simulation, including methods used to handle impacts, "hidden" impacts, constraint changes, and the possibly singular linear systems of equations. Sections 3.7 through 3.9 describe how the simulation was made efficient, the tradeoff between accuracy versus stability, and how computational exceptions were dealt with. Finally, Section 3.10 describes how the simulation was validated.

3.1 The Initial Value Problem

The motion of the walker is described by a system of second-order nonlinear ordinary differential equations (ODEs). Given an initial state of the walker (i.e., its position and velocity) and the ODEs describing how the state changes over time, it is possible to estimate what state the walker will be in at some later time. The task of projecting a future state based on an initial state and a set of ODEs is known as an *initial value problem* and is a basic task in scientific computing.

A large number of algorithms have been derived for solving initial value problems. The simplest approach, called *Euler's method*, involves using the derivatives at the initial state to make straight line projections. While conceptually simple, Euler's method is generally too error prone to be used in practice. The popular *Runge-Kutta methods* combine multiple Euler's method-style steps in ways that tend to cancel out the errors. Another general approach, the *Predictor-Corrector methods*, use past values along with the current information to reduce the error. *Extrapolation methods* are another general approach. These methods compute a series of increasingly more accurate estimates and then use these estimates to extrapolate the exact result. The walker simulation presented here used a mixture of Runge-Kutta and extrapolation approaches, described in more detail in Section 3.2.

ODEs of more that one variable and where the variables have widely different characteristic time scales are known as *stiff* systems of equations. Finding numerical solutions to stiff systems of equations presents its own set of challenges. The characteristic high frequency transient behavior of stiff systems can cause the standard methods of solving ODEs to become unstable, producing wildly oscillating values instead of the correct results. Counterparts of each of the standard methods of solving ODEs exist for solving stiff systems. In general these are *implicit* methods; that is, they use estimates of the desired value to iteratively compute more accurate intermediate values which are in turn used to compute new estimates of the desired result. The equations describing the motion of the walker are stiff, a characteristic that is most evident near energetic impacts or motion that transfers energy across many links.

This overview has been intentionally brief. For a more complete treatment of solving ODEs and the initial value problem consult one of the many excellent texts on the subject, e.g., [SWD76, PTV92, SB80, Gea71, DB74].

3.2 Solving the differential equations

Chapter 2 described how the accelerations of the walker's generalized coordinates are derived based on the state of the walker. While this process is generally rather involved and iterative, mathematically it is expressed by the simple autonomous second-order nonlinear ODE

$$\ddot{\mathbf{q}} \;=\; \mathbf{f}_{\mathrm{motion}}(\mathbf{q}, \dot{\mathbf{q}}; \mathbf{c}), \tag{3.1}$$

where $\mathbf{f}_{\mathrm{motion}}(\cdot)$ is a vector-valued function. Following the standard practice used in solving ODEs, this function is converted to an equivalent first-order ODE of twice as many variables (eighteen) of the form

$$\dot{\mathbf{x}} \;=\; \mathbf{F}_{\mathbf{c}}(\mathbf{x}), \tag{3.2}$$

where

$$\mathbf{x} \quad = \quad \begin{pmatrix} q_1 \\ \vdots \\ q_9 \\ --- \\ \dot{q}_1 \\ \vdots \\ \dot{q}_9 \end{pmatrix}. \tag{3.3}$$

Note that the vector of constraints $\mathbf{c}$ has been placed as a subscript of the function $\mathbf{F_c}()$ instead of as a parameter. This is because in solving for the motion of $\mathbf{x}$, a particular $\mathbf{F_c}()$ is used only as long as $\mathbf{c}$ remains constant, see Section 3.3.[1] Determining the walker's motion is therefore simply a series of initial value problems: given a state $\mathbf{x}_a$ at some time t_a and the ODE $\dot{\mathbf{x}} = \mathbf{F_c}(\mathbf{x})$, what is the value $\mathbf{x}_b$ at some future time $t_b > t_a$. A brief overview of the standard methods for solving initial value problems was presented in Section 3.1 above.

Based on the properties of the walker simulation and the desire to limit the number of times $\mathbf{F_c}()$ is evaluated, the present simulation uses primarily a fourth order Runge-Kutta method. For this walking simulation, the actuation of the joints changes discontinuously at each nominal time step (1 msec), and this made predictor-corrector methods, which assume that past values form a consistent trajectory, unreasonable. Further, because the function is not always well behaved (particularly as the system is simulated *past* the constraints), extrapolation methods may require an uncharacteristically large number of function evaluations. While normally Runge-Kutta methods are considered resilient but not a particularly efficient, because of the implicit discontinuities at the time step boundaries and the often non-smooth character of the function, Runge-Kutta methods were determined to be the best suited.

The particular Runge-Kutta method used was a fourth order method by Fehlberg, [Feh70]. This method uses *six* function evaluations, instead of the four normally needed for a fourth order Runge-Kutta method, but in doing so adds the ability to compute a fifth order approximation at the same time. The difference between these two methods is taken as an estimate of the error. This error estimate is traditionally used to adjust the step size, but here, as is described in Section 3.7, it was instead used to determine when to change to the stiff method described next.

[1] Also note that an important side-effect of computing Equation 3.2 is determining the constraint forces, but since this is not directly relevant to solving the ODEs it is not reflected in the notation in this section.

The motion described by Equation 3.2 is *stiff.* That is, high frequency transients in the behavior of the system can at times make numerical simulation difficult. The walker simulation used a semi-explicit extrapolation method by Bader and Deuflhard [BD83] when it was determined that a stiff solver was necessary (see Section 3.7). This method is both robust and efficient, and proved empirically to work well for the walker.

The Bader and Deuflhard method is derived from the extrapolation method for solving non-stiff systems method by Bulirsch and Stoer [SB80], which is a particular implementation of traditional Richardson extrapolation, see for example [DB74]. In Richardson extrapolation, estimates of the future value of the function are repeatedly computed, each time using a shorter time step. These results are then used to extrapolate what the value of the function would be in the limit as the number of steps goes to infinity. The advantage of the Bulirsch and Stoer method is that by using a specific sequence of step sizes the error only involves even powers of the step size and therefore has faster convergence.

Bader and Deuflhard were able to convert the Bulirsch and Stoer algorithm to one suitable for stiff systems of equations by making it semi-implicit. An *implicit* method repeatedly uses an estimate of a value (with additional information) to compute a more accurate estimate of that value. Implicit methods generally have better stability characteristics than non-implicit methods, making them well suited to stiff systems. A method is called *semi*-implicit if is an implicit method based on a linearization of the underlying system. In this case the implicit midpoint rule

$$\mathbf{y}_{n+1} - \mathbf{y}_{n-1} \;=\; 2h\,\mathbf{f}\left(\frac{\mathbf{y}_{n+1} + \mathbf{y}_{n-1}}{2}\right) \tag{3.4}$$

was linearized to

$$\mathbf{y}_{n+1} - \mathbf{y}_{n-1} \;=\; 2h\left[\mathbf{f}(\mathbf{y}_n) \;-\; \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\left(\frac{1}{2}\mathbf{y}_{n+1} \;-\; \mathbf{y}_n \;+\; \frac{1}{2}\mathbf{y}_{n-1}\right)\right] \tag{3.5}$$

where $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$ (which for the walker is $\mathbf{F_c}()$) and where $\frac{\partial \mathbf{f}}{\partial \mathbf{y}}$ is the Jacobian $\mathbf{J}$. See [PTV92] for more details.

Methods for solving stiff ODEs, such as the Bader and Deuflhard method described above, make use of the Jacobian $\mathbf{J}$ of the function $\mathbf{F_c}()$, where the elements of the array $\mathbf{J} = [J_{ij}]$ are defined to be

$$J_{ij} \;\equiv\; \frac{\partial F_i}{\partial x_j}. \tag{3.6}$$

Since $\mathbf{F_c}()$ is not an analytic function and therefore cannot be computed in closed form, it was necessary when using the stiff ODE equation solver to estimate $\mathbf{J}$

numerically. The Jacobian $\mathbf{J}$ was estimated one column at a time using forward differencing. This process of estimating $\mathbf{J}$ alone is approximately 3 times slower than using fourth-order Fehlberg Runge-Kutta method to solve the initial value problem. Empirically computing a time step using this stiff solver was found to take a minimum of 20 times longer than using the corresponding non-stiff solver.

3.3 Time of impact and state changes

Throughout the simulation, events such as impacts and the release of constraints occur at arbitrary and not easily predicted times. However, the methods used to solve ODEs described in Section 3.2 are designed to move the simulation ahead by a prescribed time step Δt. The general approach that is taken to determine when one of these arbitrarily timed events occurs is to step the simulation along by the nominal time step Δt (1 msec) until a constraint condition is violated.

Note that in order to determine the time of a constraint change it is necessary to simulate *past* the change using the existing constraints. Hence the simulation must be able to produce "reasonable" results – that is, motion that does not contain significant discontinuities, even if one or more of the constraints are violated. As an example of where this does not automatically occur, consider a foot that is being lifted nearly horizontally. As the weight of the foot is lifted the foot will start to slide when

$$|\frac{F_t}{F_n}| = \mu_s, \tag{3.7}$$

see Section 2.2.5. Notice that there is a singularity when $F_n = 0$. If the foot is being lifted horizontally, in one time step the simulation can pass from having $F_n > 0$ to having $F_n < 0$. In this case the sliding may be missed entirely, or if $F_n \approx 0$ for a floating point exception to be generated (see Section 3.9). Instead of simply starting sliding when $|F_t| = |\mu_s F_n|$, it is necessary to use the somewhat odd cutoff of $|F_t| \geq |\mu_s F_n|$ with the understanding that the foot is lifted when $F_n \geq 0$.

Given the sequence of states before the violation and the one state after the violation, interpolation is used, first to determine the precise time of the violation (and the resulting addition or removal of a constraint), and then to determine the values of $\mathbf{q}$ and $\dot{\mathbf{q}}$ at that time. Since the derivatives of the state variables $\mathbf{q}$ and $\dot{\mathbf{q}}$ ($\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$) are available as a result of solving the ODEs, it was possible to use *Hermite* interpolation to produce more accurate estimates for those values. By contrast, when interpolating forces and torques where derivative information is *not* available, standard (non-Hermite) interpolation was used. Both cases were implemented using generalized Newton's interpolation and employing a modified divided difference technique, [DB74].

Simulation events, such as impacts and the releasing of constraints, occur at the roots of constraint condition equations; i.e., at the time t such that the constraint $\varphi_c(t) = 0$. For example, when $q_6 \equiv \theta_{\mathrm{knee1}}$ hits its max-stop at $q_{6,\mathrm{max}}$ the impact occurs at the value of t such that

$$\varphi_{\mathrm{knee1,max}}(t) \quad = \quad \Big(q_6(t) \; - \; q_{6,\mathrm{max}}\Big) \quad = \quad 0. \tag{3.8}$$

The constraint condition $\varphi_c(t)$ is a nonlinear function of t, but since it is estimated by interpolation, it is fast to compute. The roots are found by the secant method, which converts to the *regula falsi* method if the estimate leaves the region bounding the estimate, [DB74].

Since the same interpolation is used to find the roots of the constraint equation and to determine the values of $\mathbf{q}$ and $\dot{\mathbf{q}}$ for interpolated states, it is sufficient simply to find the root and move to that time when constraints are based on $\mathbf{q}$ or $\dot{\mathbf{q}}$. Conversely, constraint equations that are based on constraint *forces* cannot simply assume that the estimated time will produce the anticipated constraint force. In this case it is necessary to compute an interpolated state at the estimated time and then compute the constraint forces for that state. If the resulting forces are not sufficiently accurate, this information is used to compute a refined estimate of the time and the process is repeated.

3.4 Hidden impacts

In order to maintain qualitatively reasonable behavior of the walker, it is necessary to detect and correctly respond to "hidden" impacts. Hidden impacts refer in this dissertation to impacts that occur within a time step, but where the values of $\mathbf{q}$ at the time step boundaries (both before and after the impact) do not violate any constraints. As an example, consider the caricature in Figure 3.1 in which the height of a toe passes in and back out of the ground within one time step (going from 1.23 to 1.24 seconds). Since a simple time stepped simulation only considers the state of the system at each of the distinct time steps, it would not detect these impacts, the result being grossly different qualitative behavior. Simply shortening the time step reduces but does not remove the problem; the need to detect and compensate for hidden impacts was found to be necessary for even time steps on the order of $\Delta t = 10^{-8}$ seconds.

Hidden impacts occur most often as a result of multiple impacts or a series of nearly simultaneous impacts. For example, consider two impacts, affecting

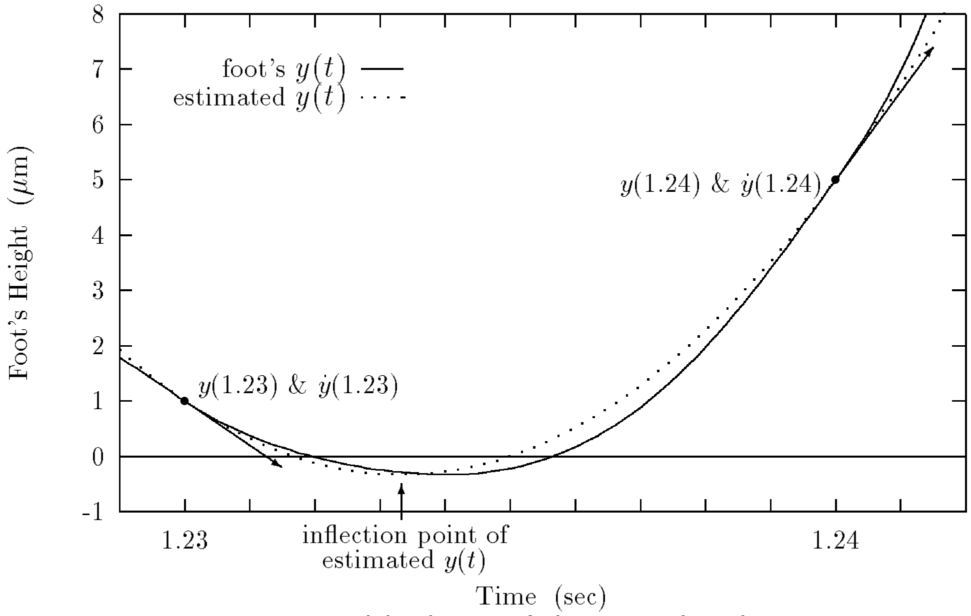

Figure 3.1: *Passing in and back out of the ground within one time step*

two generalized coordinates q_i and q_j, that occur nearly simultaneously.[2] The first impact, say for q_i, will occur when the second coordinate q_j is close to its contact condition. Because of this close proximity, even if the velocity of the second coordinate $\dot{q}_j$ is very small, an impact may still occur. However, in the longer time covered by the time step Δt, the small velocity will be dwarfed by the subsequent motion of the walker. If the gross motion of this foot is away from the ground, at the end of the next time step q_j will have moved back out of the constraint and the impact will be "hidden."

If two impacts occur nearly simultaneously the resulting motion should (generally) be qualitatively similar to when they occur simultaneously.[3] But if the nearly simultaneous impacts result in one or more impacts being hidden and ignored, the resulting gross behavior of the walker may be very different from the simultaneous case. It is therefore important to detect hidden impacts in order to retain even qualitatively reasonable behavior.

Hidden impacts are detected in the walker simulation by using a two-point second-order Hermite interpolation of the trajectory within each time step. The values of each of the $q_i(t)$ and their derivatives $\dot{q}_i(t)$ at the the beginning and end

[2] As is discussed further in Section 3.8, simultaneity is one of the factors in the simulation that is subject to computational error.

[3] Since the walker simulation is a highly nonlinear system it is *not* always the case that small changes result in qualitatively similar behavior. While generally outside the scope of this work, nonlinear dynamics are touched upon again briefly in Section 3.8.

of each time step are used to compute third-order polynomials $\tilde{q}_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$ that approximate $q_i(t)$ within the time step, e.g., the dotted line in Figure 3.1. Inflection points of $\tilde{q}_i(t)$ occur where $\tilde{q}_i'(t) = a_1 + 2a_2 t + 3a_3 t^2 = 0$, that is

$$t_{inflection} = \frac{-a_2 \pm \sqrt{{a_2}^2 - 3a_1 a_3}}{a_1} \tag{3.9}$$

If any inflection points occur within the time step, and $q_i(t_{inflection})$ indicates that an impact may have occurred, a more accurate estimate of $q_i(t_{inflection})$ is computed by making it the destination time of a new time step, and, if appropriate, this is subsequently used to compute the time of the now no-longer-hidden impact.

3.5 Constraint changes during an impact

Impacts are assumed to occur within an infinitesimally short time period, so other forces such as gravity and actuations are negligible. Therefore even a weak impulsive constraint force that pushes a generalized coordinate away from its constraint will succeed. Resolving these constraint violations involves changing $\mathbf{c}$ to remove the constraints and recomputing the results of the impact.

Often more than one impulsive constraint force violates the constraint state during an impact. An iterative process of removing the constraint corresponding to the largest violation and then recomputing the impact is used. Since constraints are only being *removed*, this process cannot cycle.

It is also possible for the changes in velocity resulting from an impact to cause a sliding foot to change the direction in which it is sliding. But, as was described in Section 2.2.5, when a sliding foot changes direction, the result is that it locks and becomes stationary. If there is sufficient tangential force the foot will then unlock and again start to slide. This locking and possible subsequent unlocking all occurs within the infinitesimally short time of the impact.

In a computationally similar manner, when a foot touches and therefore impacts the ground, it is initially assumed that both the x and the y velocity components go to zero as a result of the impact. Again, if the tangential impulsive force is sufficient, the foot will be changed to sliding by releasing the x constraint(s), and the impact will be recomputed. For example, this allows a foot brushing lightly along the ground to have its y velocity component set to zero without the x component necessarily also becoming zero.

This iterative process of resolving constraints during an impact only serves to make the constraints consistent with the walker's new state. In particular, no time passes and no energy is lost as a result. It is common for the discontinuities associated with an impact to result in a substantial change in the constraint state of the walker, and the process described in this subsection simply assures that the forces and constraints in the resulting state are consistent.

3.6 Solving the linear systems of equations

Finding a solution to Equations 2.10 and 2.16 involves solving a linear system of equations in the general form

$$\mathbf{A} \cdot \mathbf{x} \;=\; \mathbf{b}, \tag{3.10}$$

where $\mathbf{A}$ is an array and $\mathbf{x}$ and $\mathbf{b}$ are vectors. This is solved, at least conceptually, by finding the vector $\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$, where $\mathbf{A}^{-1}$ is the inverse of $\mathbf{A}$. In practice the vector $\mathbf{x}$ is computed in the walker simulation by a more efficient method based on the LUP factorization of $\mathbf{A}$, see for example [CLR90].

As was previously mentioned, for both Equation 2.10 and 2.16 it is not unusual for the array $\mathbf{A}$ to be singular or nearly singular. When $\mathbf{A}$ is singular, either no valid solution exists (the overdetermined case) or an infinite number of solutions exist (the underdetermined case), the latter being the more common with the walker simulation. When $\mathbf{A}$ is nearly singular, the value of $\mathbf{x}$ computed using LUP factorization can be dominated by computational errors, producing catastrophically poor results. Fortunately it is possible to detect that a (potential) problem exists while computing LUP factorization by checking the absolute magnitude of the divisors and resorting to a more general method when the divisors are sufficiently small.

When the array $\mathbf{A}$ is singular or nearly singular, the pseudo, or Moore-Penrose, inverse $\mathbf{A}^{+}$ can be used to produce the value $\overline{\mathbf{x}} = \mathbf{A}^{+} \cdot \mathbf{b}$. For the overdetermined case $\overline{\mathbf{x}}$ will be the least mean-squared-error solution, and for the underdetermined case $\overline{\mathbf{x}}$ will be a projection of $\mathbf{b}$ that contains none of the null column space of $\mathbf{A}$ (see [GV89] or [Str80] for a more complete explanation). As an example of the the latter case, if there is an ambiguity as to how to divide the forces between the two symmetrically positioned feet, the resulting $\overline{\mathbf{x}}$ will divide the force equally between them. The pseudo inverse was computed using singular value decomposition (SVD), see [PTV92]. The advantage of SVD is that it produces a numerically stable estimate of $\mathbf{A}^{+}$ (which for non-singular arrays equals $\mathbf{A}^{-1}$); the disadvantage is that it takes a comparatively long time to compute.

Because computing the LUP factorization is considerably faster than computing SVD, the walker simulation always first attempts the LUP factorization. If LUP fails then SVD is used. For typical runs of the walker simulation it was found that approximately 2% of the arrays require SVD, but that SVD took approximately 20 times longer to compute. Thus using the two-approach method required only approximately 7% as long to compute as using SVD alone.

3.7 Making the simulation efficient

The walker simulation described in this chapter is computationally expensive. While the execution speed varies depending on the number and type of constraints and the frequency of impacts, on average the simulation currently runs at approximately $\frac{1}{250}^{\text{th}}$ real time on an otherwise unburdened Sun Microsystems Sparc 10. That is, one second of simulated walking requires approximately four minutes of computer time to compute. Because the walker simulation is both central to this work and computationally expensive, it was important to make its implementation as efficient as reasonably possible.

Profiling executions of the simulation using the `UNIX` utilities `prof` and `gprof` showed that the vast majority of the time in computing the simulation (greater than 85%) was devoted to computing the function $\mathbf{f}_{\text{motion}}()$ in the process of moving the simulation forward. In particular, the iterative process of coordinating impacts and constraints, finding and moving to the times of constraint changes, maintaining full representations of the walker (as opposed to just the generalized coordinates), were not significant contributors to the running time of the simulation[4] – a fact which argued in favor of keeping these portions of the system highly accurate. Typically, these portions of the simulation were maintained at ten to twelve digits of accuracy. Since the errors were themselves effectively perturbations to the system that potentially could dramatically slow the speed at which the simulation could be computed, keeping these errors low was actually found to reduce the time needed to compute the simulated motion, see Section 3.8.

Two general approaches were taken to reduce the time required to compute the walker simulation. The number of times that $\mathbf{f}_{\text{motion}}()$ was computed was reduced and the time for each of those computations was minimized.

Reducing the frequency that $\mathbf{f}_{\text{motion}}()$ was computed was achieved in three ways. First the algorithms for solving the ODEs, described in Section 3.2, were selected because they minimized the number of function evaluations. Second, a general assumption was made that the system was *not* stiff. Since the method used to solve stiff systems of equations required computing the Jacobian, which

[4]These iterative processes *did* add significantly to the time of the simulation before interpolation was extensively used. See Section 3.3.

needed to be approximated numerically (a task that by itself requires nineteen function evaluations) it was more efficient to attempt the non-stiff approach first. Since the non-stiff method used for solving the systems of equations also computed an estimate of the error, this error estimate was used to decide if the stiff solver needed to be used. If either the estimated error became too large or the time step became too small then the system reverted to using the stiff ODE methods.

It should be acknowledged that it is possible for this approach to introduce errors by not switching to the stiff ODE methods early enough. The characteristic behavior that is observed in this case is for a high frequency perturbation to grow until it triggers the transition to the stiff solver. While the stiff solver will then accurately model the newly perturbed system, the perturbation will have already incorporated an error component. Two methods are used to avoid this problem. First, when the decision is made to transfer to the stiff solver the step that caused the transition is recomputed. In other words, the system backs up to a time before an error is detected and simulates the time that caused the difficulty using the more robust stiff simulator. Second, the decision as to when to transition is made conservatively: if the error rises above 10^{-10} or the time step is reduced at all from the nominal time step of $\frac{1}{1000}^{\text{th}}$ of a second then the transition is made. Empirically this approach was found to both perform well and at the same time provide better than a ten-fold speed-up over using the stiff solver alone.

It should be noted that since constraint changes occur at arbitrary times, the time step following a constraint change is always smaller than the nominal simulation time step. The result of this is that the stiff solver is *always* used immediately after a constraint change. While being possibly over-cautious, this was seen as a reasonable precaution.

The third method for reducing the frequency that $\mathbf{f}_{\text{motion}}()$ was called involved maintaining a queue of past function results. Because of the way the ODE's were solved, occasionally $\mathbf{f}_{\text{motion}}()$ was called repeatedly with the precise same values. A circular queue of the arguments and results of the 32 most recent function calls was stored. When the function was called, first the queue was checked to see if the results had already been computed, and if they had the stored results were simply returned.

The actual computing of $\mathbf{f}_{\text{motion}}()$ involves two steps, generating $\mathbf{A}_m$ and $\mathbf{b}_m$ and then using those to find for $\mathbf{x}_m$, Equation 2.10. It was found that the first of these steps requires 70 to 80% of the time involved.

Optimized `C` computer code to generate $\mathbf{A}_m$ and $\mathbf{b}_m$ of Equation 2.10 was generated by MapleV to implement the equations derived from Lagrangian Mechanics. Because of the size and complexity of the equations of motion and of the resulting `C` code generated by MapleV, it was necessary to write a filter program that simplified and rearranged the `C` code generated by MapleV. This filter program processed the `C` code by rearranging terms to allow reduced scoping of variables, converting simple variables to `#define`'d constants, and converting the array's double indexing to direct pointer operations to allow inline optimization. Only then was it possible to compile this code using an optimizing `C` compiler. Even so, the only `C` compiler that was able to successfully compile even the simplified and reduced code was `gcc` by the Free Software Foundation, [Sta93].

While much less significant to the overall performance of the walker simulation, a similar speed-up was achieved in the code that solved $\mathbf{Ax} = \mathbf{b}$. Often one or more rows of the array $\mathbf{A}$ contained only one non-zero element, a phenomenon most common in the lower rows representing the constraint equations. These rows represent equations of the form $a_{ij}\, x_j = b_i$ which immediately provides the result $x_j = b_i / a_{ij}$. By adding code that reduced the size of $\mathbf{A}$ and $\mathbf{b}$ in this manner, a speed-up of more than two was achieved for this code. Note that this speed up applied regardless of whether LUP or SVD was used to compute $\mathbf{x}$, see Section 3.6.

3.8 Accuracy versus stability

The aim throughout the development of this software was to produce a system that was stable and resulted in qualitatively *reasonable* behavior. Extreme numerical precision was not considered important or necessary. Thus the system of equations was stabilized using constraint equations given by Equation 2.9, when necessary linear systems of equations were solved using SVD (Section 3.6), and the ODEs were solved using stiff methods (Sections 3.1 and 3.2).

Errors enter the system in two general ways: computational errors and estimation errors. Computational errors, such as round-off errors, are an inevitable result of the limited resolution of the computer representation of the real-valued numbers being computed. The estimation errors come from the finite resolution of the iterative methods used to find values such as the time and location of impacts and to balance forces on the ground and foot accelerations. Both these errors can be considered to be perturbations to an ideal simulation.

It turns out, somewhat counterintuitively, that since maintaining high numerical precision reduced these perturbations of the underlying nonlinear system, the extra effort required to maintain high accuracy generally produced a system that could be simulated more quickly and that was more stable. For this reason, double precision numbers were used for all real-valued computations, and values were typically estimated to twelve digits of accuracy. This accuracy reduced the occurrences of high frequency transient behavior which had sometimes resulted from perturbing the system, making stiff solution of the equations less likely.

Finally, while generally beyond the scope of this dissertation, the fact that the walker simulation is highly nonlinear has some important ramifications. For a linear system the magnitude of an input determines the magnitude of the response but can have no effect on the *type* of the response. Stated mathematically

$$\begin{aligned} f(\alpha x) &= \alpha f(x) \\ f(x+y) &= f(x) + f(y) \end{aligned}$$

For example, in a linear system if applying a torque of 1 Newton-meter to a joint produces an acceleration of 1 radian/second^2, then applying a torque of 2 Newton-meters to the same joint will produce the same phenomenon, an acceleration, but now the magnitude of that acceleration will be 2 radians/second^2.

This is not necessarily true for nonlinear systems. Many physical systems are at least approximately linear over some limited range of inputs and outputs. In the above example the second acceleration might not be precisely 2 radians/second^2, but if it was adequately close the linear model would likely be a reasonable approximation. Traditional, that is linear, methods of simulation and control rely heavily on this assumption. But in extreme cases, nonlinear systems can not even be approximated by a linear system over the range of interest. In particular, *hard* nonlinearities such as the discontinuities associated with impacts, can cause even the qualitative behavior of a system to be radically different from what would be predicted by a linear model. Because of the hard nonlinearities associated with the impacts of the feet and joints of this study's walker simulation, it is possible for small changes to the inputs to produce dramatically different resulting behavior in the walker. As will be discussed in the following chapters, the final walking motion exhibited an attracting limit cycle, a phenomenon that does not occur for linear equations.

An extreme form of nonlinear behavior is *chaos* in which arbitrarily small perturbations can grow to produce arbitrarily large differences in the resulting states – the so called "butterfly effect." It has been shown that a system with impacts, even if the underlying system is otherwise linear, can produce bifurcations that lead to chaos [SS89]. Good introductions to nonlinear system theory that are nonetheless quite accessible can be found in [Kha92] and [Gri90].

Thus, while it would seem to be a reasonable goal to require the walker simulation to produce qualitatively *correct* behavior, throughout this chapter the emphasis has been on producing *stable* and qualitatively *reasonable* behavior instead. This can be an important distinction. Seemingly subtle changes, such as small changes in the required accuracy of the estimation process, have been observed to produce qualitatively different behavior.[5] While the behaviors may be qualitatively different, both versions are still qualitatively *reasonable*, and both are numerically stable.

3.9 Handling Exceptions

In addition to the constant struggle to maintain numerical precision throughout the computations, on many occasions it was necessary to handle exception conditions. Two classes of exceptions were handled explicitly; timing exceptions and numerical exceptions. The `C/UNIX` signaling mechanism was used to detect and then to act on these conditions.

A timing exception was generated when a simulation had run too long. While care had been taken to assure that the simulation could not get locked endlessly into an infinite loop, because of the potential stiffness of the differential and the multiple optimizations that had to be performed, it was not uncommon for the simulation to effectively stop because the time step had dropped by many orders of magnitude. In each of the many instances that were investigated, it was found that this occurred where an extremely energetic impact or other dramatic event had occurred. In the few instances that this was allowed to be simulated out, these situations inevitably led to the walker quickly falling over or otherwise failing.

To prevent the simulation from periodically becoming locked into simulating what presumably was a difficult but ultimately unproductive event, the operation was timed. If the allotted time expired, then simulation was terminated prematurely.

The other class of exception that needed to be addressed was floating point exceptions. These included underflow, overflow and divide-by-zero errors. As an example, consider the expression used to compute the "sigmoid" cutoff for both the neural networks and the controllers discussed in the following chapters. The

[5] In one extreme example that was observed, changing the estimation accuracy bound from six digits of accuracy to twelve digits of accuracy caused a "hidden" impact in the less accurate simulation to be missed in the more accurate simulation. This resulted in the first simulated walker falling slightly backward and the second falling slightly forward (the walker was not being actuated).

exponentiation in the expression

$$\sigma(x) \;=\; \frac{1-\mathrm{e}^{-x}}{1+\mathrm{e}^{-x}} \tag{3.11}$$

will generate underflow or overflow conditions when the magnitude of x is large and negative or positive, respectively. In degenerate conditions this can return the floating point value "*Not-A-Number*," the default representation of values that are too corrupted to be returned meaningfully. Admittedly this error only occurs when the system is making enormous errors, but it is nonetheless not acceptable to have the simulation break ungracefully in these sorts of instances.

In this example the expression can be converted to the equivalent expression

$$\sigma(x) \;=\; \frac{2}{1+\mathrm{e}^{-x}} \;-\; 1. \tag{3.12}$$

While this expression can still produce both overflow and underflow exceptions, the default way that these exceptions are handled by the computer causes the returned floating point value to be correct. The problem with letting the computer trap and handle exceptions in this way is the inherent inefficiency of the approach. Since execution speed of the program was considered important, such inefficiency was undesirable. In this particular example, if $|x| > 40$ then 1 or -1 was returned.

In general a mixed approach was used for handling floating point exceptions. As with timer exceptions, the `C/UNIX` facility for signaling and trapping on exceptions was used. This was used at first to find the dozens of places, such as the function mentioned above, where floating point exceptions occurred regularly and then to modify the code appropriately. Once all of the problem areas had been addressed, a more resilient set of floating point exception handlers was installed which would allow the simulation to succeed even when large numerical errors had occurred, but would indicate that the results of the simulation should be considered suspect.

It should be noted that a more general and more elegant solution to this problem would have used the `C++` exception handling mechanism, [Str91]. Unfortunately, while this mechanism has been specified, none of the compilers that were available to this project supported this feature.

3.10 Validating the Simulation

A serious concern in developing the new software was finding a method to validate that the software was in fact producing a reasonable simulation of walking.

Normally a computer simulation is constructed to model an existing physical system, so validation amounts to comparing the results computed by the software to the observed actions of the physical system. Because the current work considers a simulation of a physically *feasible* system but one for which there is no physical counterpart, such a comparison is not possible.

The walker simulation was validated by an extensive suite of tests. These tests corresponded to observing the simulated walker in circumstances where the appropriate action could determined independent of the simulation. The successful completion of these tests was taken to indicate that the software provided an acceptable simulation of walking.

A simulation of a system inherently will contain some error, the goal of these tests were to verify that the magnitude of those errors remained within reasonable bounds. The computer code for the walking simulation used 64-bit double precision numbers throughout. While the accuracy of the numerical calculations varies, it is reasonable to expect double precision calculations to produce at least 14 digits of accuracy for relatively complex but straightforward mathematical expressions. Parameters in the walking system whose accuracy was controlled by the simulation, such as parameters that were estimated using iterative techniques, were required to stabilize to twelve digits of accuracy. While it was rare for the error found in the following tests to exceed those twelve digits of accuracy limit, it did occasionally occur. The following tests were considered to be successful if the error was maintained to ten digits of accuracy.

The tests and their characteristics were:

Motion of the center-of-mass:

The motion of the center-of-mass of an object is only affected by external forces such as gravity and contacts with the ground. In particular, the friction and actuation of the walker's internal joints should have no effect on the motion of the center-of-mass itself. The motion of the center-of-mass was checked with the robot in *all* of the constraint states. The motion was correct to within the accuracy of the double precision calculations, that is ± 14 significant digits of accuracy.

Equivalent results for equivalent computations:

The motion of the walker is computed with respect to a base coordinate system assigned to the toe of the foot designated $foot_1$. It was verified that changing the assignment of $foot_1$ produced equivalent motion. When the robot was reversed, that is a "mirror image" of the robot was simulated with the applied torques being similarly reversed, exactly the same results were produced (as would be anticipated); that is, there was *no* error introduced.

In the case in which both of the robot's feet were on the ground, it was possible to arbitrarily assign $foot_1$ to either the front or the back foot. In this case many of the underlying assumptions about motion, the origin of constraint forces, and how constraints affect the walker's motion are significantly changed. Further, since closed kinematic chains which are a natural result of the two foot stance create stiff systems of ODE's, this was arguably the most difficult of the tests when considered from the perspective of maintaining numerical accuracy. This is the one case were errors greater than twelved digits of accuracy were observed, with the greatest observed variance being 2.7×10^{-11} meters within a 100 time step window (that is $\frac{1}{10}^{\text{th}}$ second).

Motion in simplifying situations:

It is possible to effectively reduce the dynamics of the seven segment walker to that of a simpler physical system. By using the symmetry of the robot and by eliminating usable degrees of freedom, either by aligning segments with the applied forces or by locking joints, it was possible to create a system that was simple enough to allow the appropriate action to be computed.[6] The walker was tested with the legs in identical positions (so that in an illustration of the walker the two legs appear as one), as the walker jumped and as it landed,[7] with the walker in a symmetric A-frame type stance,[8] and with all but one portion of the walker vertical. In each of these cases the behavior was correct to within the simulations computation error – that is, twelve significant digits of accuracy.

Reaction to impulses:

Impulsive forces from contacting the ground or hitting a joint stop result in discontinuous changes in the velocity $\dot{\mathbf{q}}$. A series of tests was made to insure that the appropriate change in velocity occurred in the generalized coordinates involved in the impact, that the velocity of the resulting system was consistent with the resulting constraints, and that impacts of joints did *not* affect the motion of the center-of-mass.

In these tests the simulated walker was placed in midair and given random initial trajectories. Torques were then applied to the joints, and the initial

[6]The term "simple" is a relative term here. The appropriate motion was solved using MapleV on the simplified system of equations and typically still required many hours to compute.

[7]This is where the *hidden impacts* discussed in Section 3.4 were originally observed.

[8]The symmetric position required the that joints of one leg bend "backwards" and therefore for the max and min stops of that one leg to be changed.

velocities of joints were set so as to cause the joints to hit their stops, creating impacts. This was done for joints individually as well as for pairs of joints simultaneously (to within the 10^{-12} second accuracy of the simulation). In all cases the resulting motion of the center of mass followed the anticipated parabolic arc to within the simulations twelve digits of accuracy.

Reaction to open-loop control signals:

The action of the walker for simple open-loop control signals was computed and compared to the observed behaviors of the walker simulation. Specifically, the anticipated and observed behavior of applying constant valued torques to individual joints and pairs of joints was compared and found to be within acceptable bounds.

The most extensively studied behaviors were of actuation of the swing foot alone when the support foot was flat on the ground, and the behaviors of actuating either both ankles, both knees, or both hips when both feet were on the ground at identical positions with the calves and thighs parallel. In these later cases the feet were not allowed to either slide or come off the floor; that is, they were "nailed to the floor." In all of these cases the predicted behavior matched the simulated behavior to within the simulation's twelve digits of accuracy.

Reaction to known linear controller:

It was initially assumed that methods which had been used in the past to control simplified walking robots would be sufficient to control the walking simulation described in this dissertation. In particular it was assumed that a controller similar to that of the BIPER robots as described in [MS84] would produce a periodic walking motion. While it was eventually possible to cause the simulation to walk, as will be described in the following chapters, this was never successfully achieved using the simple approach (based on the inverted pendulum model).

This inability was not attributed to errors in the present system but rather to the complexity and high degree of nonlinearity of the walking simulation. Issues involved with controlling the walking motion form the basis of the remainder of this dissertation.

CHAPTER 4

Trajectories

Even the simple two-dimensional simulated walking robot used in this study contains more degrees of freedom than are necessary to produce a walking motion. An important issue in causing the walker to walk is the determining of an appropriate trajectory for all of the parts of the walker to follow during each step. Walking is realized by applying torques to the walker's joints in order to make them follow a desired or target trajectory. A well-designed target trajectory specifies a motion that not only moves the walker forward, but does so in a smooth and controlled manner that is easy to follow in that it minimizes the torques required. One of the most significant contributions of this study was the development of a target trajectory which is simple to compute but at the same time produces substantially better results than can be achieved using the target trajectories used in previous studies.

A "good" target trajectory – a target trajectory that requires that a minimum of torque – is typically derived by prescribing a motion that is close to the "natural" motion of the walker when the joints are not actuated. Because of the complexity of the equations presented in Chapter 2, it is necessary to work with simplified models of the walker's motion when deriving target trajectories. Past analytical work on walking has been based on the assumption that the walker's motion can be adequately approximated by an inverted pendulum – a simplified model based on the assumption that the joint angles and velocities remain small. Unfortunately this assumption is not valid for even moderate stride lengths.

In this study, the inverted pendulum model is rejected, and the trajectory is based on the idea that the body of the walker should move at a constant height and speed. The joint torques are minimized by keeping the motion of the body steady, thereby minimizing the accelerations of the body. Unlike the models in past studies which have assumed that the body has little or no mass, this walker had almost half of its mass in its body. As a result, a trajectory based on moving the body at a constant height and speed produced good results and will be seen to move the center-of-mass along an easily realized trajectory.

A second and related concern in specifying the target trajectory was to provide a trajectory that minimized the number and the strength of impacts. Impacts such as those that occur when a foot makes contact with the floor are an inevitable part of walking, but they have traditionally been ignored as being either insignificant or too difficult to understand. The author found that impacts affect the motion of the entire walker, and that even weak impacts can cause significant and disruptive transient behavior. The *number* of impacts was reduced by keeping the knees always slightly bent. The *strength* of the remaining impacts was reduced by specifying a trajectory in which the feet touched the ground with zero velocity.

The next two sections of this chapter describe the inverted pendulum model and the constant velocity model. The final section presents and the compares the resulting trajectories. One final note before proceeding: while the previous chapters described the motion of the walker using the joint angles θ, it is simpler to describe the trajectories based on the angles ϕ of each segments from vertical.

4.1 Inverted Pendulum Model

Past studies have assumed that the motion of the walker around the support ankle can be approximated by a mass with fixed inertia rotating around a stationary pivot, in other words, an inverted pendulum. Even though it is not accurate, the inverted pendulum model is important. First, since the physics of pendulums are simple and well understood, the insights carried over from our understanding of pendulums are useful. Second, since target trajectories have traditionally been based on the inverted pendulum model, it is important to understand to what degree this model is and is not viable.

A mass with fixed inertia rotates around a stationary pivot according to the equation

$$mgl\sin(\theta) \;=\; \mathrm{I}\,\ddot{\theta} \tag{4.1}$$

where m is the mass, g is the gravitational constant, l is the distance from the pivot to the center-of-mass. I is the mass's inertia around the pivot, and θ is the angle from horizontal from the pivot to the center-of-mass, Figure 4.1. When θ is small, $\sin(\theta) \approx \theta$ and therefore Equation 4.1 can be approximated by the linear equation

$$\theta \;=\; \omega^2\,\ddot{\theta} \tag{4.2}$$

where

$$\omega^2 \;=\; \frac{\mathrm{I}}{mgl} \tag{4.3}$$

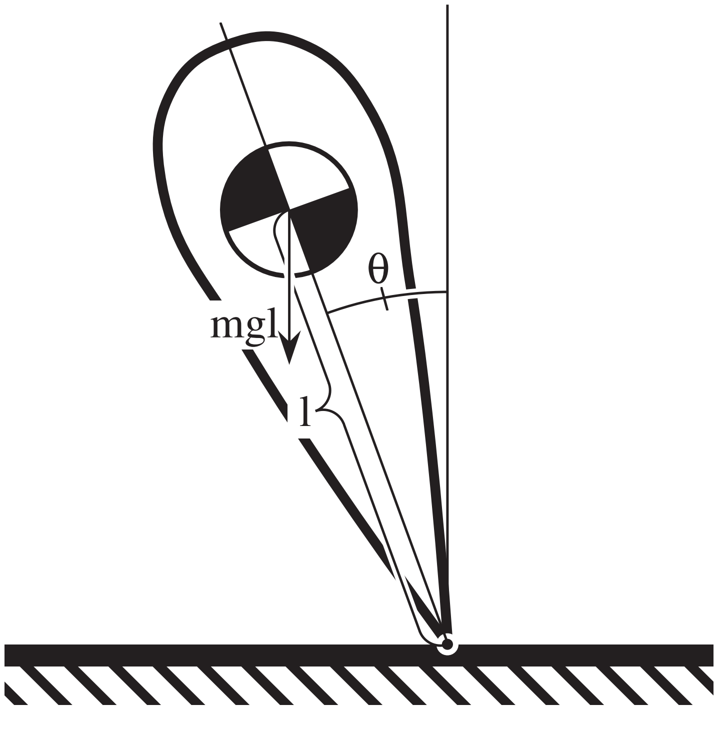

Figure 4.1: *Inverted pendulum model*

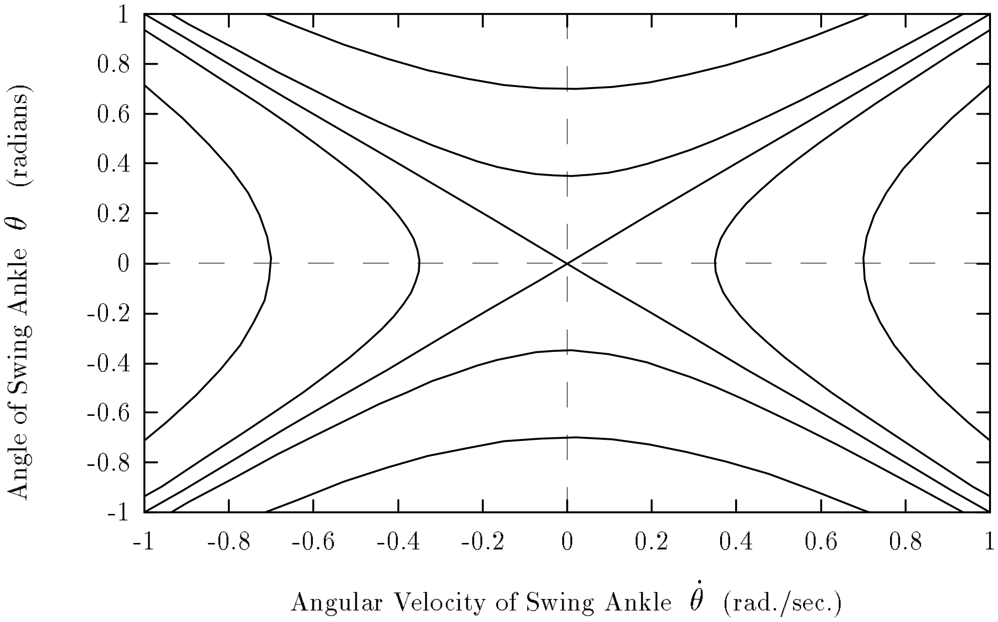

Figure 4.2: *Motion of an inverted pendulum model*

This has the solution

$$\theta(t) \;=\; \frac{1}{2}\left[(\theta_o + \frac{\dot{\theta}_o}{\omega})\,\mathrm{e}^{\omega t} \;+\; (\theta_o - \frac{\dot{\theta}_o}{\omega})\,\mathrm{e}^{-\omega t}\right] \tag{4.4}$$

where $\theta_o = \theta(t_o)$ and $\dot{\theta}_o = \dot{\theta}(t_o)$ are the angle and angular velocity of the inverted pendulum at some time t_o. A plot of the orbits of a simple inverted pendulum are shown in Figure 4.2. The simplicity of this model makes it an ideal foundation for a target trajectory as was done, for example, with the BIPER robots in [MS84]. The BIPER robots were exceptionally successful in that they were able to use the inverted pendulum model as a basis of their dynamic control. For this reason they will be used through the remainder of this chapter as an example of the successful application of the inverted pendulum model.

The target trajectory of the BIPER robot was designed to make the robot's motion as close to an inverted pendulum as possible. Most significantly this was achieved by having the support leg of the robot remain straight, an arrangement which kept the mass of the robot at an approximately fixed distance from the support ankle and thus tended to keep the distribution of the mass, and therefore the inertia, fixed. While this stratagem largely succeeded in keeping the motion of the BIPER robots close to an inverted pendulum model, it also resulted in having the center-of-mass of the BIPER robot move up and down with each step. Figure 4.3 shows the height of the center-of-mass of an inverted pendulum tra-

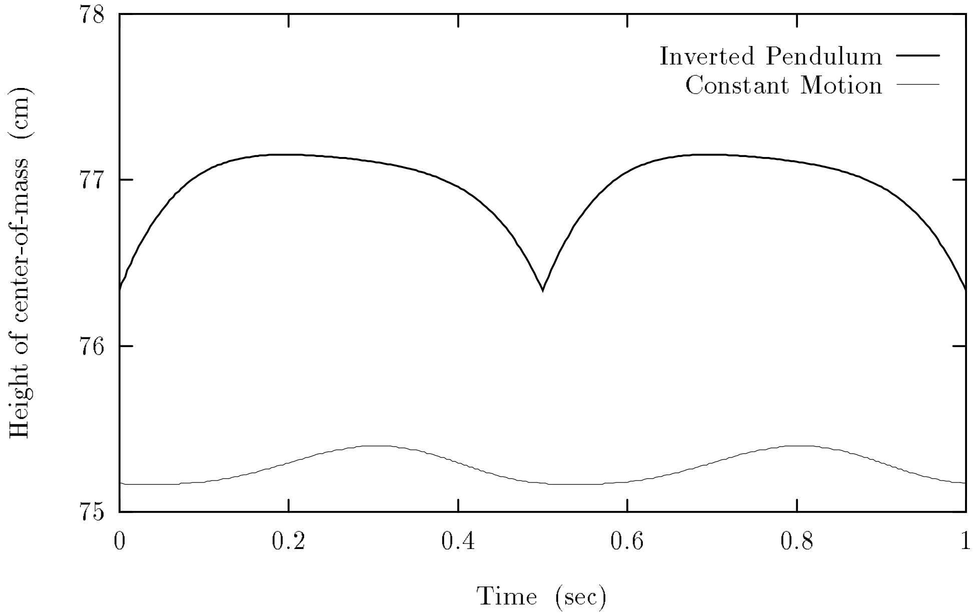

Figure 4.3: *Height of the center-of-mass*

jectory, based on the BIPER robots, compared to the constant motion trajectory described in the next section. The current study found this up-and-down motion of the center-of-mass difficult to maintain, requiring increased torques at the walker's knees and hips.

In the discussion so far, nothing has been specified about the trajectory of the swing leg. In [MS84], Miura and Shimoyama derive fourth-order polynomial trajectories based on a series of boundary conditions. These specify that the angles of the joints of BIPER should be continuous when the robot switches support legs and that the velocities and accelerations of the angles should be continuous for the leg being lifted. Using this technique to derive the trajectory for the swing foot of the walker results, in the present study, in the trajectory shown in Figure 4.4.

There is a problem with this trajectory for the swing foot. As the calf of the swing leg passes through horizontal, the height of the swing foot dips precariously close to the ground, as can be seen in Figure 4.4 at time ≈ 0.35 second and again at ≈ 0.85 second, where the swing foot drops to approximates 0.2 cm from the ground. Small errors in the joint angles are capable of lowering the swing foot and causing it to drag along the ground, which typically leads to the walker pitching slightly forward, an event that eventually leads to the walker stumbling.

Having now specified the trajectory of the body and both legs, it is possible

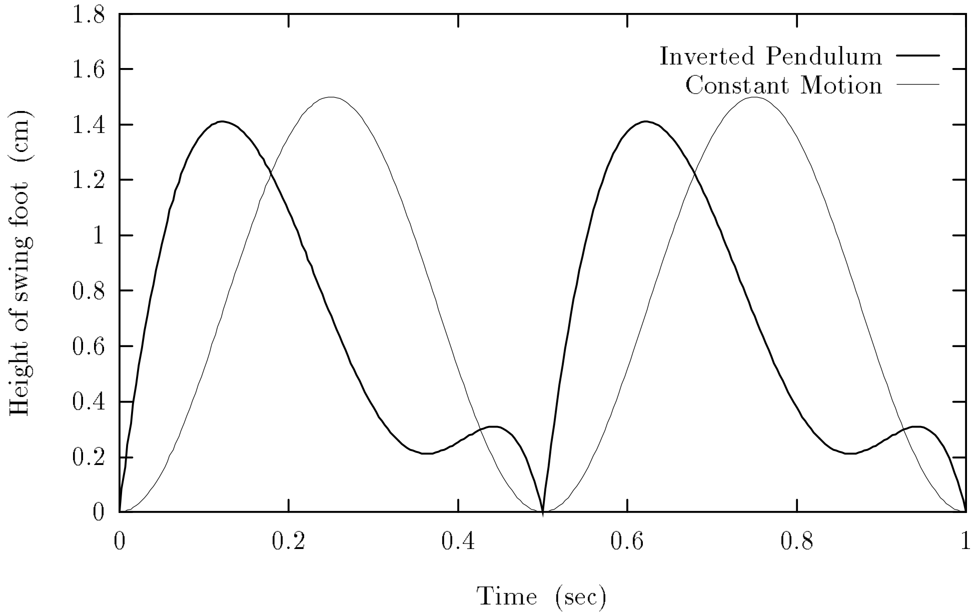

Figure 4.4: *Height of swing foot*

to plot the joint angles (shown in Figure 4.5). Note that the angles of the calf and thigh of the support leg are the same and that the angle for the calf of the swing leg bends further forward before straightening just before being set down. The target angles of the remaining segments are all zero. That is to say, the body is vertical and the two feet are horizontal.

4.2 Trajectories Based on Constant Motion

The target trajectory used in this study is based on moving the body of the walker at a constant rate and at a constant height. It was assumed that by minimizing the accelerations of the body, the amount of torque needed at the joints would likewise be minimized. Further, while other studies typically assume that the mass of the body of the walker is *not* significant, this study assumed that the mass of the body was substantial enough to justify focusing precisely on the motion of the body.

Given the static position of the support ankle and the known constant horizontal trajectory of the body, the angles from the vertical, ϕ, of the calf and thigh of the support leg can be derived. These are given by

$$\phi_{sup,calf}(t) = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_b(t)}{x_b(t)}\right) + \frac{1}{2}\cos^{-1}\left(\sqrt{x_b(t)^2 + y_b(t)^2}\right) \quad (4.5)$$

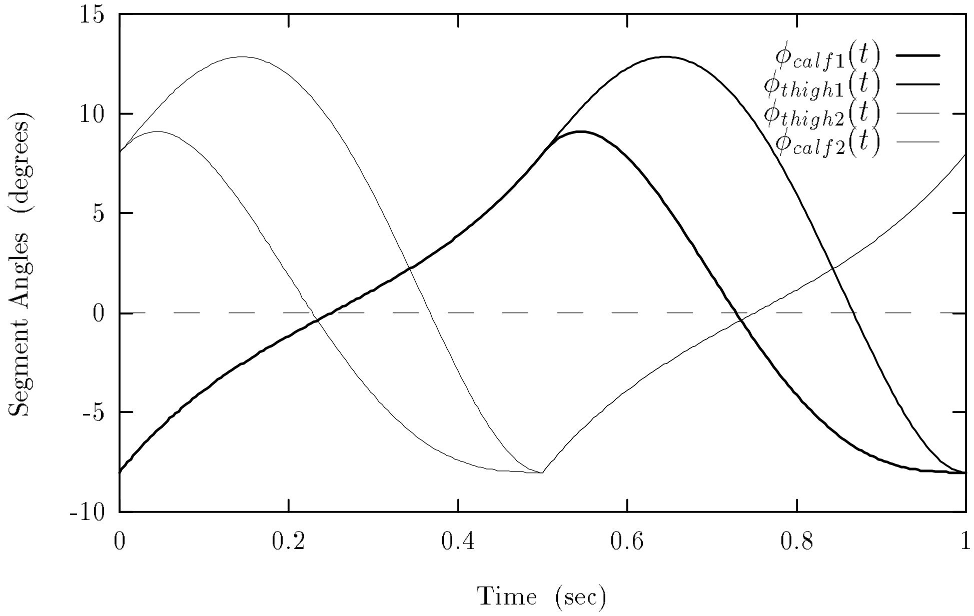

Figure 4.5: *Angle of segments in inverted pendulum trajectory*

$$\phi_{sup,thigh}(t) = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_b(t)}{x_b(t)}\right) - \frac{1}{2}\cos^{-1}\left(\sqrt{x_b(t)^2 + y_b(t)^2}\right) \quad (4.6)$$

where $x_b(t)$ and $y_b(t)$ describe the position of the base of the body with respect to the position of the support ankle:

$$x_b(t) = x_{b,0} + \dot{x}_b t \quad (4.7)$$

$$y_b(t) = y_b \quad (4.8)$$

where $x_{b,0} = \frac{-w_{step}}{2}$ is the x position of the body at the start of the step (where w_{step} is the length of a step), $\dot{x}_b = \frac{w_{step}}{T}$ is the constant x velocity of the body (where T is the period of the step), and y_b is the constant target height of the base of the walker's body.

Even given the motion of the body and the support leg, there is still latitude on the possible trajectories for the swing leg. The desire is for the the swing foot to travel well above the floor, and for the impacts to be made at zero velocity. An initial path that was briefly considered for the swing foot was a simple semi-elliptic arc given by the parametric equations

$$x(t) = -\mathrm{A}\cos(\pi t), \qquad 0 \le t \le 1 \quad (4.9)$$

$$y(t) = \mathrm{B}\sin(\pi t), \qquad 0 \le t \le 1 \quad (4.10)$$

where A is half the stride length and B is the height of the step, and plotted in Figure 4.6. The *advantage* of this path is that it moves the foot quickly away

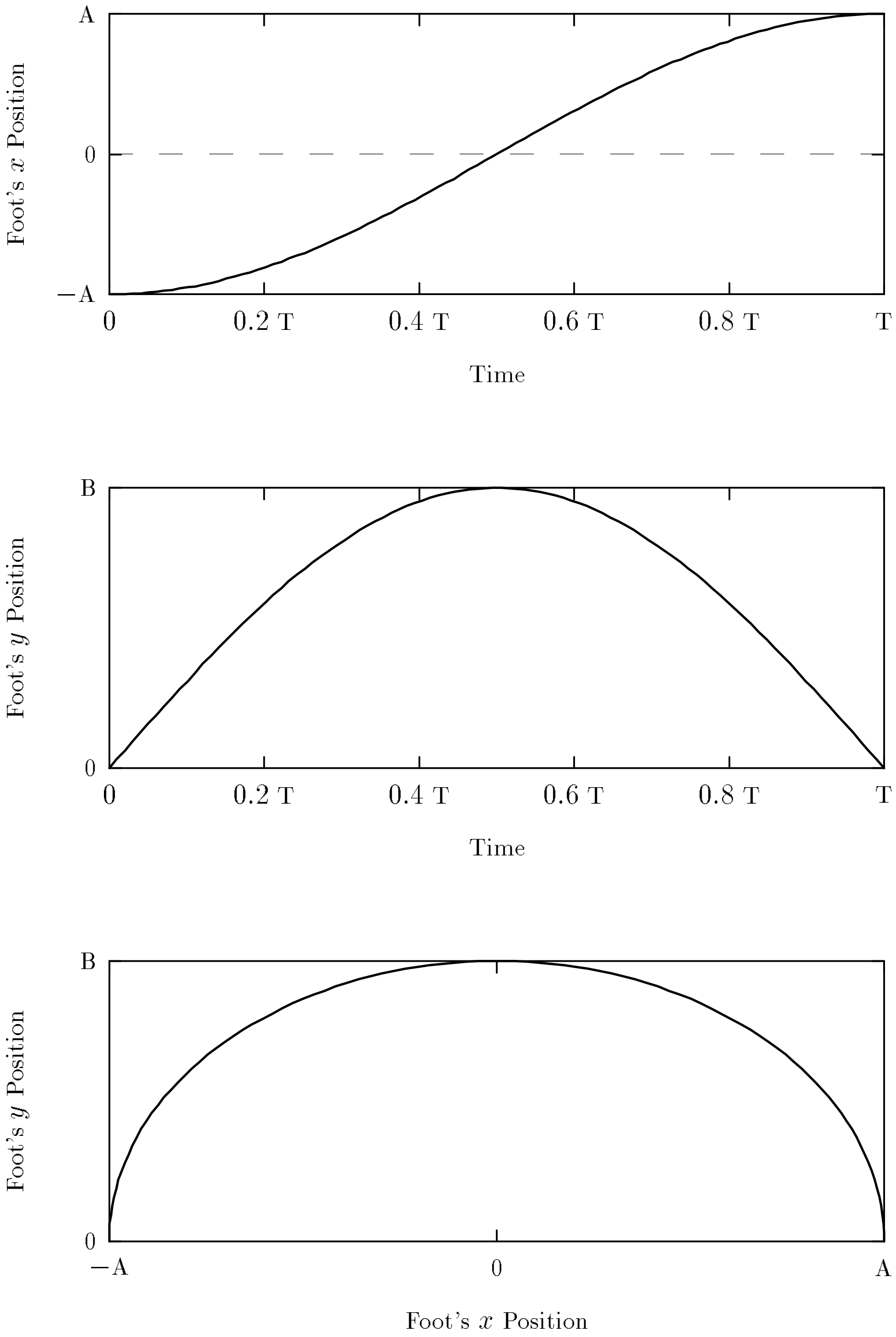

Figure 4.6: *Semi-elliptic foot trajectory*

from the floor and keeps it well clear throughout the step, making it tolerant to high levels of error in the walker's joint angles. The *disadvantage* of this approach is that the vertical component of the swing foot's velocity is at its greatest at the point the foot leaves and impacts the ground. The high velocity at impact causes difficulties from the jarring impact, and the high velocity when the foot leaves the ground require extremely large torques to achieve.

The trajectory of the swing foot that was used for this study is similar to the one in Equations 4.9 and 4.10 above, but has the advantage that the velocity of the feet as they leave and touch the ground is zero. The position of the swing foot is given by the parametric equations

$$x(t) = -A\cos(\pi t), \qquad 0 \leq t \leq 1 \tag{4.11}$$

$$y(t) = \tfrac{1}{2}B\,(1 - \cos(2\pi t)), \qquad 0 \leq t \leq 1 \tag{4.12}$$

where again A is half the stride length and B is the height of the step. These curves are plotted in Figure 4.7 and should be compared to Figure 4.6. This is plotted along with the height of the swing foot back in Figure 4.4.[1]

Having now specified the trajectory of the body and both legs, it is possible to plot the joint angles. These are shown in Figure 4.8, which should be compared to Figure 4.5. Notice that in both plots the angles for the calf and thigh of the support leg tend to increase in value through the step and the calf and thigh of the swing leg tend to decrease, but since the knees in the constant motion model are bent more, the lines are more separated. Notice also that in the constant motion plot, Figure 4.8, the lines for each thigh and calf tend to bow apart and then back together as the height of the body is maintained at a constant level.

4.3 Comparison of Trajectories

The advantage of the constant velocity model over the inverted pendulum model is that it is "easier" for a walking robot to follow. One indication of that ease is that while it was possible to make the walker follow the trajectory based on the constant velocity model for relatively long step lengths, the author was *not* able to create a controller capable of following the trajectory with comparable step lengths based on the inverted pendulum model (see Chapter 4). As a result it is not possible to compare the torques required to control these two trajectories,

[1] In Figure 4.4 the maximum height of the swing foot for the trajectory based on the constant motion model is 1.5 cm; in the remainder of this study the height of the swing foot is 2.5 cm. This change was made in order to make the height comparable to that of the trajectory based on the inverted pendulum model, in which this height can not be directly controlled.

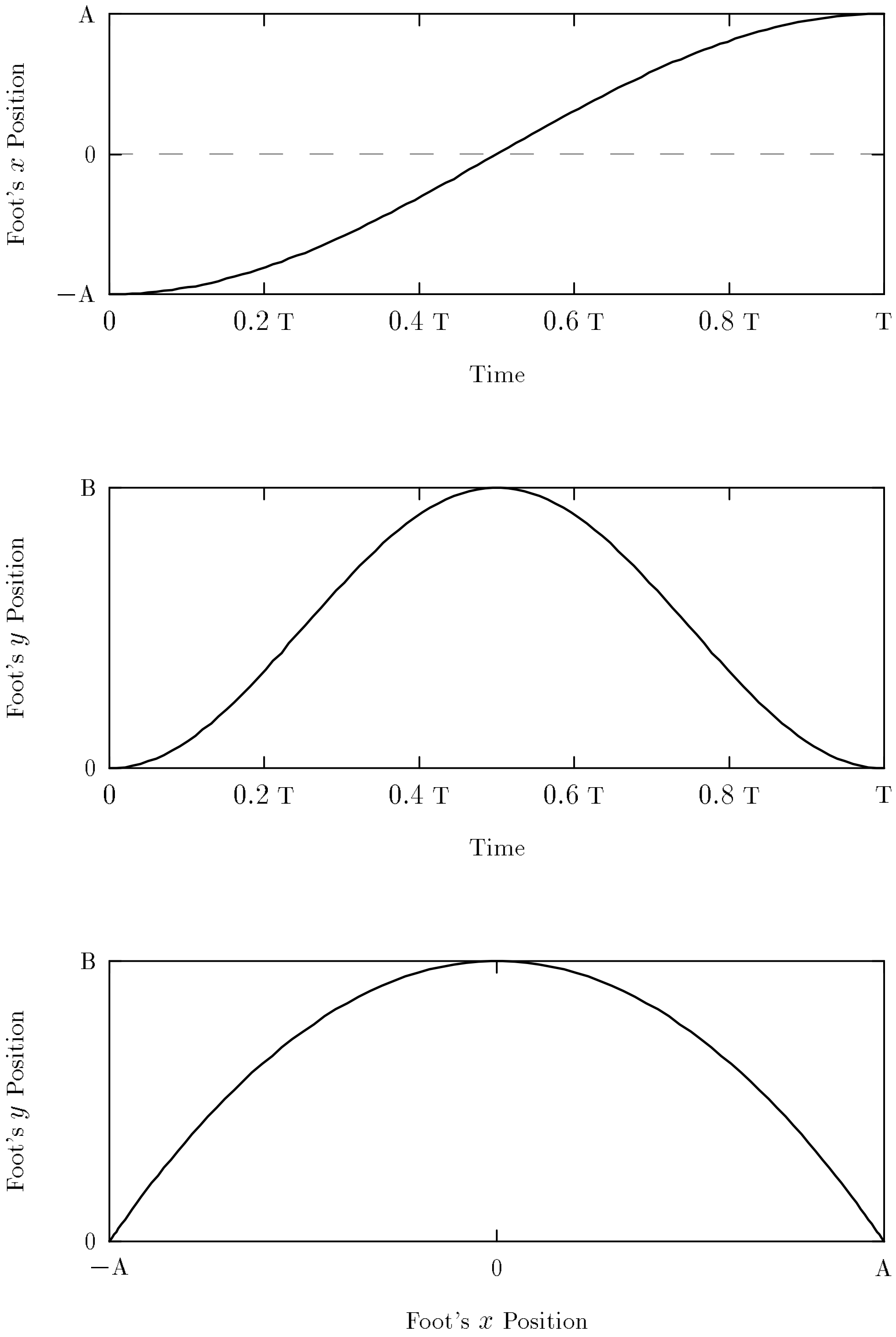

Figure 4.7: *Modified semi-elliptic foot trajectory*

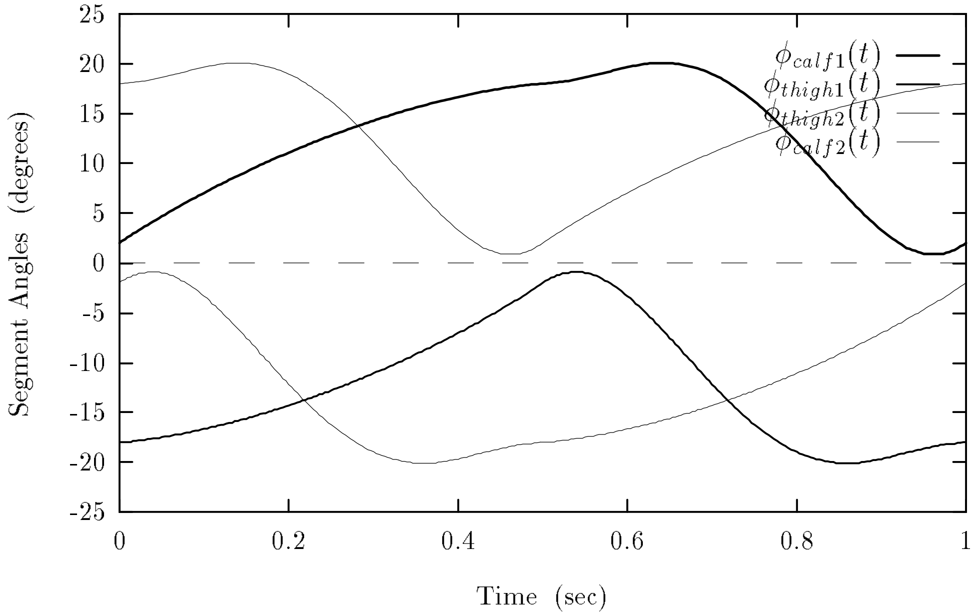

Figure 4.8: *Angle of segments in constant motion trajectory*

since the torques required for the trajectory based on the inverted pendulum model are not known. Therefore, this section will instead compare velocities and accelerations, which are directly related to the torques, by considering the plots of the positions and velocities found in the trajectories.

First consider the heights of the center-of-mass for the two trajectories, shown back in Figure 4.3. Not only is the magnitude of the motion greater for inverted pendulum trajectory (more than three times as large), there is a significant discontinuity at the time of impact when a foot is placed on the ground. In the inverted pendulum trajectory, not only is the swing foot being moved downwards to the ground but the entire body of the walker is following a semi-circular arc downwards at the time of impact. This highly energetic impact is difficult to adequately control. By contrast, the vertical motion of the center-of-mass for the constant motion trajectory is low at this point and is comparatively easy to control, because the height of the body is constant and the vertical velocity of the swing foot is zero at the time of impact.

Second, consider Figure 4.9 which shows the *velocities* of the bodies and the centers-of-mass for the two trajectories. For the inverted pendulum model the velocity of both the center-of-mass and the base of the body is greatest at the time of impact and smallest during the step. While this profile is a necessary and direct consequence of the inverted pendulum model, it is actually difficult

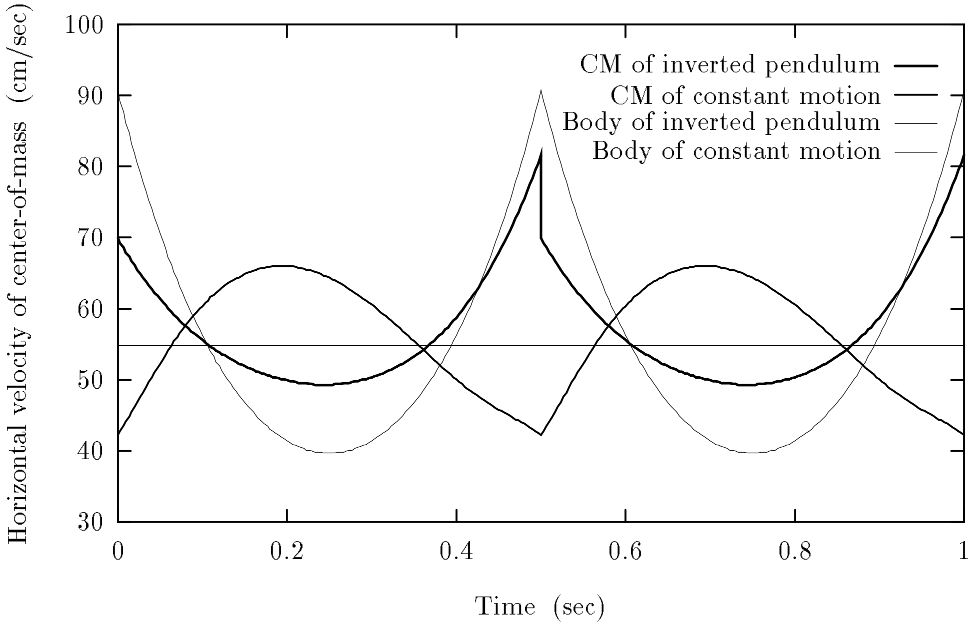

Figure 4.9: *Center-of-mass and body x velocities*

to follow. Since the body of the walker is horizontal throughout the step,[2] the target trajectory dictates that the velocities are too large at the time of impact and too small in the center of each step. Consequently, controlling this trajectory involves *slowing* the robot after a foot fall and accelerating it afterwards; that is, the controller must over compensate for what should happen "naturally." This forces the walker to expend energy to produce a motion that is counterproductive.

In the trajectory based on the constant motion of the walker's body we see the reverse behavior. Since the entire trajectory is based upon the proposition that the horizontal velocity of the walker will be constant, it is edifying to see that this is so. The velocity of the center-of-mass is actually slowest at the time of impact and fastest in the center of the step. This change in velocity comes primarily from the motion of the swing leg, whose horizontal velocity follows the same pattern. The constant motion of the body makes that portion of the trajectory smooth, and while the walker does pull against the support foot this is in order to move the walker forward. While the trajectory based upon the inverted pendulum actually pushes against the ground in order to *slow* the forward motion of the robot during the first half of the step, the horizontal /pagebreak force exerted on the ground during the first half of the constant motion trajectory is used to

[2]In order for the walker to remain closer to an inverted pendulum, the body of the walker should rotate through the step, but this causes there to be a discontinuity in the angle of the body at the time of impact.

move the walker forward.

It is interesting to consider in retrospect why the BIPER robots were able to follow a trajectory based on the inverted pendulum model. The major clue comes from the plots presented in [MS84]. While the length of the robot's step is never explicitly discussed, it can be derived from these plots. The maximum angle of the thighs of the BIPER robot during a step is approximately 0.03 radians $\approx 1.75°$, leading to step length of less that 1 cm. The simulated walker in this study had an $8°$ stride which, because it was a taller robot, lead to a 27.4 cm step length. The period of each step was the same for both, 0.5 seconds, leading to an almost 30-fold difference in maximum velocities between the studies. Apparently it is necessary for a walking robot whose trajectory is based on the inverted pendulum model to take short slow steps leading to small angles and slow angular velocities.

CHAPTER 5

Control

The final task in causing this study's simulated two-legged robot to walk was to determine the torques which must be applied to the six actuated joints in order to make the robot follow the target trajectory. This process is called *controlling* the walker, and the device that performs this function is called the *controller*. The controller used in this study is based on the common proportional-integral-derivative (PID) control paradigm, but has been modified to better handle the transients associated with walking and to better suit it for the adaptive technique that was used to fine-tune the control parameters. Finally, a feed-forward neural network, based on the same PID paradigm and also capable of controlling the walker, was developed.

This chapter first discusses modal decomposition of the linearized model, an approach to control that was considered but eventually rejected, and sets forth the reasons for its failure. Then the traditional PID controller is reviewed, and the modifications that were made to that model in this study are discussed. The method employed to incrementally adjust the control parameters is presented next, followed by sample runs of the successfully controlled walker simulation. Finally the feed-forward neural network that was developed is presented.

5.1 Modal decomposition

In controlling the motion of the walker, it would be typical to assume that each of the joints can be controlled independently.[1] But for the walker in this study, as for most complex systems, simply applying torque to one joint causes not only that joint but typically *all* of the joints to respond. To resolve such control problems, it is in principle possible to uncouple the motion of a complex system into independent components or *modes*. Given these independent modes, it is then possible to estimate what combination of torques on each of the joints need to be applied to cause only a single joint to respond.

[1]This *controllability* property and the related *observability* property are important issues in control theory, but they will only be touched on briefly at the end of this section.

This so-called *modal decomposition* is particularly straightforward for linear systems. However, as was discussed in the last chapter, the motion of the walker is not close enough to linear to make a linear approximation viable. Nonetheless, modal decomposition of the constant coefficient linearized system is presented here for two reasons: it lends insight into the motion of the walker, even if the specifics are only approximately valid and apply for only a small portion of the walking trajectory, and it is an approach that is similar to what has been used elsewhere when studying walking. For example, modal decomposition was used by Canon and his students at Stanford University to study the motion of multiple and jointed inverted pendulums, [Can67, Hig63, Sch65].

Here, as in Chapter 3, the walker will be described using the angles ϕ_j of the segments from their nominal position instead of the joint angles θ_j. This convention provides more insight into the control and resulting motion of the walker. It is sufficient here to note that there is an invertible linear relationship between these two bases, i.e., there exists a $\mathbf{K}$ such that $\boldsymbol{\phi} = \mathbf{K}\boldsymbol{\theta}$.

The equations of motion that were presented in Chapter 2 can be greatly simplified if the assumption is made that the angles and velocity in the system are small, that the system remains in the same constraint state, and friction and impacts are neglected. For small angles we can approximate $\sin(\phi_i) \approx \phi_i$ and $\cos(\phi_i) \approx 1$, and for sufficiently small velocities, terms involving second and higher powers of the angles or velocities can be ignored. These approximations allow the nonlinear differential equations presented in Chapter 2 to be simplified to the second-order *linear* system of differential equations

$$\ddot{\boldsymbol{\phi}} = \mathbf{A}\boldsymbol{\phi} + \mathbf{B}\mathbf{u} + \mathbf{e} \tag{5.1}$$

where the vectors $\boldsymbol{\phi}$ of segment angles and $\mathbf{u}$ of control signals are

$$\boldsymbol{\phi} = \begin{pmatrix} \phi_{sup,calf} \\ \phi_{sup,thigh} \\ \phi_{body} \\ \phi_{swg,thigh} \\ \phi_{swg,calf} \\ \phi_{swg,foot} \end{pmatrix}, \qquad \mathbf{u} = \begin{pmatrix} u_{sup,ankle} \\ u_{sup,knee} \\ u_{sup,hip} \\ u_{swg,hip} \\ u_{swg,knee} \\ u_{swg,ankle} \end{pmatrix} \tag{5.2}$$

Linearizing around $\boldsymbol{\phi} = [0,0,0,0,0,0]^t$ and substituting in the values used in this study, the matrices $\mathbf{A}$ and $\mathbf{B}$ and the vector $\mathbf{e}$ were found to have the numerical values

$$\mathbf{A} = \begin{pmatrix} 197.163 & -189.000 & 8.168 & 9.093 & -1.082 & 0.000 \\ -223.573 & 295.048 & -38.798 & -43.193 & 5.138 & 0.000 \\ 24.759 & -99.420 & 47.091 & 31.968 & -3.803 & 0.000 \\ -33.925 & 136.224 & -39.346 & -98.178 & 34.409 & 0.000 \\ 9.538 & -38.299 & 11.062 & 81.331 & -63.402 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \end{pmatrix} \quad (5.3)$$

$$\mathbf{B} = \begin{pmatrix} 0.830 & -1.770 & 1.045 & -0.247 & 0.183 & -0.040 \\ -0.941 & 2.409 & -1.964 & 1.173 & -0.869 & 0.191 \\ 0.104 & -0.599 & 1.096 & -1.103 & 0.643 & -0.141 \\ -0.143 & 0.821 & -1.180 & 2.043 & -2.818 & 1.277 \\ 0.040 & -0.231 & 0.332 & -1.418 & 3.629 & -2.353 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 32.877 \end{pmatrix} \quad (5.4)$$

$$\mathbf{e} = \begin{pmatrix} 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 64.438 \end{pmatrix} \quad (5.5)$$

Modal decomposition is performed by finding the modal matrix $\mathbf{M}$ such that $\mathbf{M}^{-1}\mathbf{A}\mathbf{M}$ equals the diagonal[2] matrix $\mathbf{\Lambda}$. $\mathbf{M}$ is constructed from the six distinct

[2]The matrix will only be diagonal when, as here, $\mathbf{A}$ is normal and therefore the eigenvalues are distinct. In the general case the ξ_i are *generalized* eigenvectors and $\mathbf{M}^{-1}\mathbf{A}\mathbf{M}$ will be the Jordan normal matrix $\mathbf{J}$ instead of the diagonal matrix $\mathbf{\Lambda}$.

eigenvectors ξ_i of $\mathbf{A}$, that is $\mathbf{M} = [\xi_1 \ldots \xi_6]$. For the values used in this study $\mathbf{M}$ was found to equal

$$\mathbf{M} = \begin{pmatrix} -0.560 & 0.031 & 0.341 & 0.123 & 0.425 & 0.000 \\ 0.767 & 0.082 & 0.245 & 0.166 & 0.467 & 0.000 \\ -0.202 & -0.092 & -0.857 & -0.128 & 0.744 & 0.000 \\ 0.236 & 0.620 & 0.407 & 0.828 & 0.254 & 0.000 \\ -0.034 & -0.810 & 0.153 & 1.288 & 0.209 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix} \tag{5.6}$$

The eigenvectors of $\mathbf{A}$, or equivalently the columns of $\mathbf{M}$, are the independent modes of the system.

By defining the new vector $\mathbf{x} = \mathbf{M}^{-1}(\boldsymbol{\phi} + \mathbf{e})$ and rearranging the terms, Equation 5.1 is transformed to[3]

$$\ddot{\mathbf{x}} = \boldsymbol{\Lambda}\mathbf{x} + \mathbf{B}_n\mathbf{u} \tag{5.7}$$

where $\boldsymbol{\Lambda} = \mathbf{M}^{-1}\mathbf{A}\mathbf{M} = \mathrm{diag}[\lambda_{i,i}]$ (where the $\lambda_{i,i}$ are the eigenvalues of $\mathbf{A}$) and $\mathbf{B}_n = \mathbf{M}^{-1}\mathbf{B}$. For the values in this study this leads to

$$\boldsymbol{\Lambda} = \begin{pmatrix} 455.044 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & -120.870 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 51.157 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & -16.258 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 8.649 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{pmatrix} \tag{5.8}$$

[3]From Equation 5.1 it would appear that $\mathbf{x}$ should be defined as $\mathbf{x} = \mathbf{M}^{-1}(\boldsymbol{\phi} + \mathbf{A}^{-1}\mathbf{e})$. But not only is $\mathbf{A}$ singular, but that singularity is in the last row and column of $\mathbf{A}$, corresponding to the only non-zero component of $\mathbf{e}$.

and

$$\mathbf{B}_n = \begin{pmatrix} -1.324 & 3.138 & -2.292 & 1.035 & -0.638 & 0.081 \\ 0.093 & 0.156 & -0.528 & 2.156 & -4.330 & 2.453 \\ 0.203 & -0.057 & -0.655 & 0.752 & -0.151 & -0.091 \\ 0.029 & 0.003 & 0.019 & -0.034 & -0.003 & -0.014 \\ 0.026 & 0.009 & -0.061 & 0.199 & 0.096 & -0.269 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & -32.877 \end{pmatrix} \tag{5.9}$$

Note that since none of the rows of $\mathbf{B}_n$ are all zeros we can deduce that this linear approximation to the walker system is *controllable*, and since $\mathbf{M}$ has no columns that are all zero the linear approximation to the walker system is *observable*.[4]

The first observation to be made from this short presentation of modal decomposition of the linear approximation comes from noting that the last column of $\mathbf{M}$ has only one non-zero element and the last column $\mathbf{\Lambda}$ is all zeros. Thus, for the linearized model the motion of the swing foot is decoupled not only from the motion of the rest of the walker but even from its own current angle and is found empirically to be a reasonable gross approximation for the full non-linear system as well. This premise is in fact the primary reason for basing this portion of the study on the segment angles ϕ.

The second observation to be made is that the remainder of the linear system is strongly coupled. Aside from the swing foot, the actuation of each joint significantly affects the motion of the other joints of the walker. Even though the particulars of this modal decomposition were *not* found to be useful, the general insight that the system is highly coupled is empirically valid.

5.2 PID Control

Proportional-integral-derivative (PID) control is a standard control paradigm based on the error of the system, the difference between the target value and the system's measured state. As the name implies, the control signal is the sum of three components: a signal proportional to the error (to reduce that error),

[4] For a more complete discussion of controllability and observability consult a standard control text, e.g., [Bro85, pp. 259 ff.].

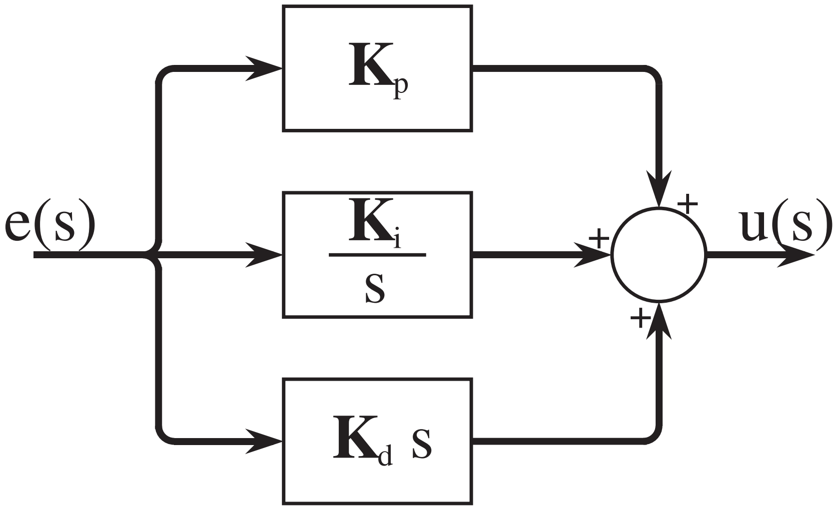

Figure 5.1: *Continuous time PID controller*

a signal proportional to the derivative (to reduce overshoot), and a signal proportional to the integral of the error (to reduce persistent bias). This section will introduce both the continuous and the discrete time versions of the PID controller.

The equation for the continuous time PID controller is given by

$$u(t) \;=\; \mathrm{K}_p\, e(t) \;+\; \mathrm{K}_i \int_0^t e(t)\, dt \;+\; \mathrm{K}_d \frac{d}{dt} e(t) \tag{5.10}$$

where $u(t)$ is the control signal at time t, $e(t)$ is the error, and K_p, K_i, and K_d are constants for the proportional, integral, and derivative terms respectively.

The Laplace transform of Equation 5.10 yields the associated transfer function in the s-domain:

$$\mathrm{G}(s) \;=\; \frac{\mathrm{U}(s)}{\mathrm{E}(s)} \;=\; \frac{\mathrm{K}_d s^2 \;+\; \mathrm{K}_p s \;+\; \mathrm{K}_i}{s} \tag{5.11}$$

The continuous time PID controller is shown in Figure 5.1.

This study models a discrete time controller. A difference equation approximation of Equation 5.10 can be formed by approximating the integral by trapezoidal integration and the derivative by backwards differencing, yielding

$$u_k \;=\; \mathrm{K}_p\, e_k \;+\; \mathrm{K}_i\, \mathrm{T} \sum_{j=1}^{k} \frac{(e_j + e_{j-1})}{2} \;+\; \mathrm{K}_d \frac{(e_k - e_{k-1})}{\mathrm{T}} \tag{5.12}$$

where T is the sampling period.

Eliminating the sum by subtracting u_{k-1} from u_k and rearranging terms yields

$$u_k \;=\; u_{k-1} + \left(\mathrm{K}_p + \frac{\mathrm{K}_i \mathrm{T}}{2} + \frac{\mathrm{K}_d}{\mathrm{T}}\right) e_k + \left(\frac{\mathrm{K}_i \mathrm{T}}{2} - \mathrm{K}_p - \frac{2\mathrm{K}_d}{\mathrm{T}}\right) e_{k-1} + \frac{\mathrm{K}_d}{\mathrm{T}}\, e_{k-2} \tag{5.13}$$

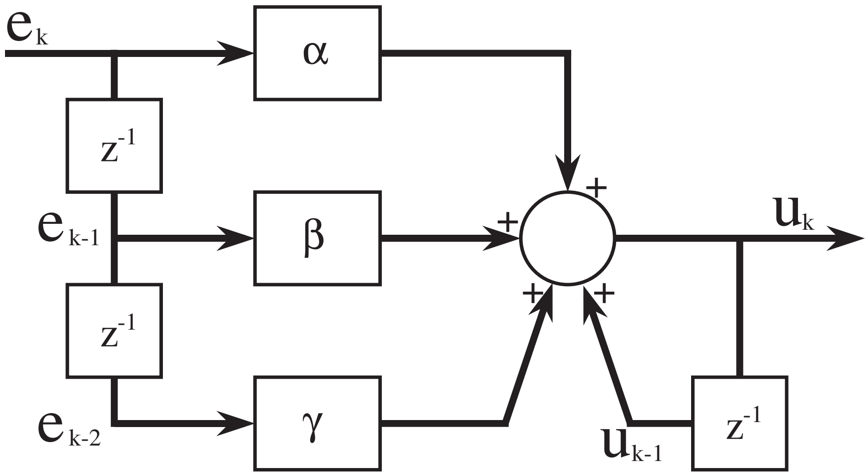

Figure 5.2: *Discrete time PID controller*

Taking the z-transform of both sides yields

$$G(z) \;=\; \frac{U(z)}{E(z)} \;=\; \frac{\alpha \,+\, \beta z^{-1} \,+\, \gamma z^{-2}}{1 - z^{-1}} \tag{5.14}$$

where

$$\begin{aligned}
\alpha &= K_p \,+\, \frac{K_i T}{2} \,+\, \frac{K_d}{T} \\
\beta &= \frac{K_i T}{2} \,-\, K_p \,-\, \frac{2K_d}{T} \\
\gamma &= \frac{K_d}{T}
\end{aligned}$$

The discrete time PID controller is shown in Figure 5.2.

5.3 Modified PID Control

The controller used in this study was not the "pure" PID controller described above but was modified to make it better suited for this particular walking application. An asymptotic bound on the control signal was included to avoid integrator windup and to facilitate the adaptive process used to refine the control parameters. After describing this modification, this section will conclude by explaining why another *ad hoc* modification that is often used to reduce the effects of transients was *not* used.

The simulated walker in the current study was intended to approximate a physically realizable robot; as a result, the range of torques applied to the joints had to be limited. The standard approach when creating such a limit is simply to clip the control signal at some threshold U_k – that is, the clipped control signal $\tilde{u}_k$ would be given by

$$\tilde{u}_k = \begin{cases} U_k, & \text{if} \quad u_k > U_k; \\ u_k, & \text{if} \quad -U_k \leq u_k \leq U_k; \\ -U_k, & \text{if} \quad u_k < -U_k. \end{cases} \tag{5.15}$$

Limiting the output range of a PID controller inherently creates a problem known as *integrator windup*, which occurs when the control output is at its maximum value but the system is still unable to remove an error. Consider, for example, a foot dragging along the ground. The error accumulates in the integrator, but since the controller is at its maximum value, the output of the controller does not change. When the controller is finally able to remove the error, say in this example when the foot is raised back off the ground, the integrator may have "wound up" an enormous amount of error causing the controller to radically overcompensate. The standard solution to integrator windup is to stop further integration when the output is saturated. This process, known as *dumping*, is implicit in the incremental form of the discrete time PID equations, Equation 5.15, given that the control signal is clipped (u_{k-1} is replaced by $\tilde{u}_{k-1}$).

However, for the current application, there is a problem with simply clipping the output of the controller. As discussed in the next section, the control parameters were tuned using an adaptive technique. But if the output is clipped, the system will perform identically for *any* clipped output, making convergence of the adaptive techniques slow. In order to facilitate the process of adjusting the control parameters, an asymptotic bound of the form

$$\tilde{u}_k \;=\; \frac{1 \,-\, \mathrm{e}^{-2\frac{u_k}{U_k}}}{1 \,+\, \mathrm{e}^{-2\frac{u_k}{U_k}}} \tag{5.16}$$

was used, where in this case U_k is the asymptotic bound.

The principal difficulty in controlling the walker in the present study comes from the transients resulting from the impacts. A standard "trick" used to stabilize PID controllers against transients (primarily for when the controller is initially turned on) is to substitute the value being controlled, in this case θ, for the error in the derivative term, [Lei87, p. 42]. Utilizing such a "trick," Equation 5.12 would be replaced by

$$u_k \;=\; \mathrm{K}_p\, e_k \;+\; \mathrm{K}_i\, \mathrm{T} \sum_{j=1}^{k} \frac{(e_j + e_{j-1})}{2} \;+\; \mathrm{K}_d\, \frac{(\theta_k - \theta_{k-1})}{\mathrm{T}} \tag{5.17}$$

This change is based on the assumption that the target value is changing slowly compared to e (typically it is a constant), so that $\dot{e} \approx \dot{\theta}$. The assumption holds unless there is a transient: at which time the derivative term serves to smooth the motion following a transient, the desired effect. For the current walker, though, joint angles can move at speeds comparable to the error term, making the standard approach unacceptable.

5.4 Adjusting the Control Parameters

Given the modified PID control paradigm, the final task in creating the controller for the present study was finding the control parameters. This system has eighteen control parameters, the K_p, K_i, and K_d for each of the six joints, and a combination of two methods was used to find these parameters. First, "hand tuning" techniques were used to quickly find approximate values, and then adaptive computer search techniques were used to refine and optimize the control parameters.

In theory, control parameters can be derived from the mathematical model of the system being controlled. In practice, because models are either imprecise or overly complex, it is common to use more empirical methods to find these control parameters. The most common methods involve simple *ad hoc* approaches that allow "hand crafting" of the values. These methods are quick, but tend to be inaccurate. Another approach frequently used involves incrementally refining values using computer based search.

Methods for finding PID control parameters include tuning based on open-loop step response of the system, [ZN42] and tuning based on the oscillatory behavior of proportional controllers, [Lei87]. Both methods were used to find approximate initial values for the controller in the current study. Unfortunately, because of the coupling and general nonlinearity of the system, these methods were only able to produce very crude initial estimates of the appropriate values.

With the approximate values for the controller parameters in hand, computer search was then used to refine and optimize those values. Unfortunately the underlying general problem, nonlinear multivariate optimization, is a hard problem, which *no* finite method is guaranteed to answer.[5] For equivalent reasons, there

[5] Consider an infinite space where the function being optimized is constant valued except for a finite-sized region containing the minima. Because of the homogeneity of the function, there is no method that performs better than random search, and because the minima is a finite-sized region in an infinite space there is no way to guarantee it will be discovered with a finite algorithm.

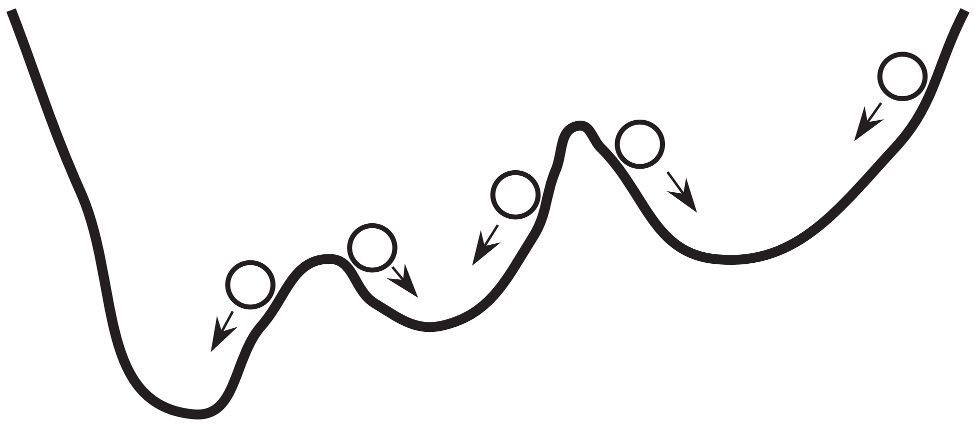

Figure 5.3: *Gradient descent methods may get stuck in local minima*

is no way to assert that for an unknown function, any optima found in a finite amount of time is the true or global optima.

Given this caveat, a number of methods exist for finding a *good*, if not optimal, value of a function. These can be broken into two categories, gradient methods and random (or stochastic) methods. Gradient methods, alternately called hill-climbing or gradient-descent depending on whether the optima corresponds to a maxima or minima, use the direction that the function is changing (its gradient) to iteratively seek out an optima. The advantage of these methods is that they can generally move quickly to a minima, the disadvantage being that this minima may be a local, but not global, minima, Figure 5.3.

Random or stochastic methods combine random "guesses" with existing information about the space. Random searches can in some cases perform considerably better than gradient-based methods, particularly when the space being searched contains many local (sometimes called "parasitic") minima. Two currently popular methods that fall into this category are *genetic algorithms* (see for example [Gol89]) and *simulated annealing* (see for example [KJ83]). For example the author has used the genetic algorithm to develop controllers for a variety of difficult control problems involving balancing jointed and multiple inverted pendulums, [Wie91a, Wie91b], work that has been extended in [CV94].

Gradient descent methods have been used in the past to tune PID control parameters for real world applications, e.g., [LMS69, RMS69]. These past studies used simple univariate techniques, but univariate techniques can perform poorly for multi-variate systems such as the one being considered here. In particular, it is possible that the space being searched contains a long narrow "valley." Simply taking steps along the local gradient can cause the system to "bounce" down the walls of the valley.

In the present study a conjugate gradient method was used in order to reduce this problem. Conjugate gradient methods iteratively rotate the basis of the search after each step, trying to force motion normal to past steps. Since the derivative of the value being optimized is not easily computed for this study (the space was in fact discontinuous), a method by Powell that also iteratively estimates the gradient was used, [Act90], after being modified as described below.

The function being minimized was a weighted sum of the squares of error of the joint angles, the derivative of errors, and the magnitude of the normalized control signals. That is, the function being minimized for a state s, $F(s)$, was given by

$$F(s) \;=\; \sum_{t \in \mathrm{T}} \sum_{j \in \mathrm{J}} \left(\alpha_1 \, {e_j}^2(t) \;+\; \alpha_2 \, \dot{e}_j^2(t) \;+\; \alpha_3 \left(\frac{u_j(t)}{U_j} \right)^2 \right) \qquad (5.18)$$

where T was the time window being considered, J are the joints, $e_j(t)$ is the error of joint j at time t, $u_j(t)$ and U_j are the control signal and its asymptotic bound, and α_1, α_2, and α_3 are the weighting factors, where (typically[6]) $\alpha_1 \gg \alpha_2 > \alpha_3$.

There is a problem with the naive application of gradient methods to adjusting the control parameters because of the limited time window. The initial hand-tuned parameters were unable to control the walker through an impact, and thus the optimization process generally began with a time window of approximately the duration of one step, 0.5 seconds. However, the error for these short time windows can be minimized in ways that do not perform well for continuous walking. For example, increasing the period of an oscillating error signal, even if the magnitude of that oscillation is increased, can sometimes reduce the integrated error, Figure 5.4. One "easy" way of increasing the period is by adjusting K_i so the integration term works to *increase* the error, creating a controller that in the long term is unstable (it will produce an unboundedly large control signal). Thus one way to decrease the error for short time windows is to create an unstable controller.

Because of this difficulty, Powell's method was modified from its standard implementation, e.g. ,[PTV92]. First, since it was desirable to simply "improve" the current value instead of seeking out the minimum for each time window T, Powell's method was allowed to take only a few steps along the current gradient (typically between 1 and 10).[7] Second, since Powell's method was only run for a small number of steps along the gradient, the computational effort that is

[6] The actual weighting factors were adjusted in response to the performance of the controller.

[7] Each step involves estimating the derivative in each of the 18 dimensions and then moving

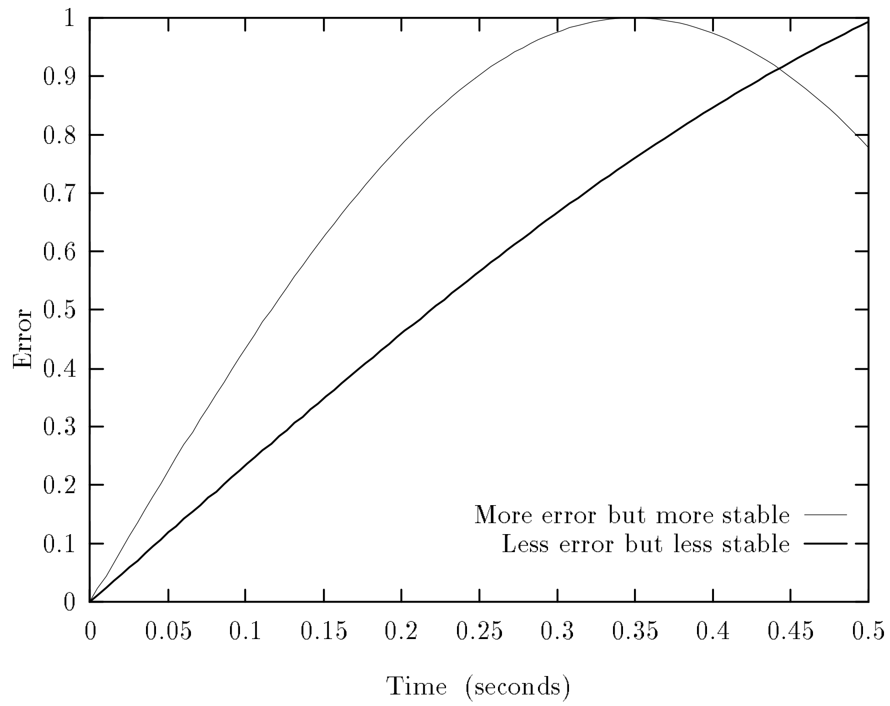

Figure 5.4: *Minimizing the controller error can produce unstable controllers*

typically extended in finding accurate estimates of the gradient (using a method developed by Brent, [Bre73]), was unnecessary and time consuming. Instead of using Brent's algorithm, which iteratively estimates the minimum by assuming that the function being optimized is locally approximately second-order, a single Brent's algorithm-style estimate was considered sufficient. The system as a whole, then, iteratively used the modified version of Powell's method to improve the control parameters and then increase the time window T.

5.5 The resulting control signals

The present study was successful at creating a controller which could cause the simulated two-legged robot to follow the target trajectory, that is, to produce a sustained walking motion. This section presents a series of plots showing the performance of the controller and the walker during this walking motion.

Figures 5.5-5.12 all show two seconds, approximately 4 steps, of walking at the end of a twenty-second session of simulated walking. A vertical line is drawn

along the estimated gradient. This typically required a few dozen executions of the walking simulation that, depending on the length of the time window T, required several hours to compute.

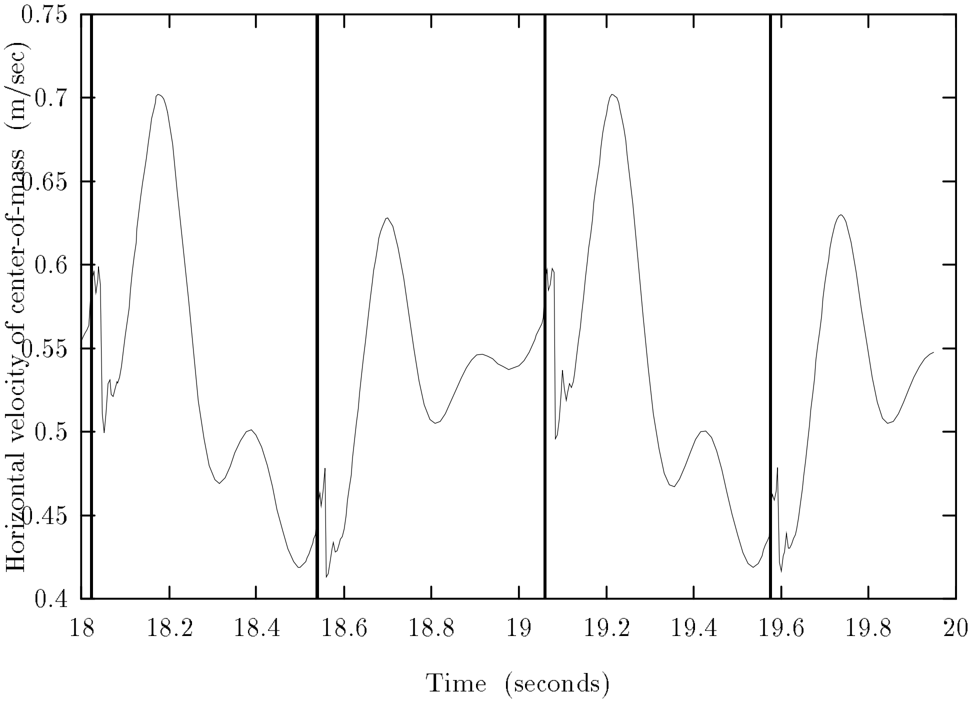

Figure 5.5: *Horizontal motion of the center-of-mass*

at the points where a foot touches the ground. The simulated walker was started at simulated time zero 0.05 seconds into a 0.5 second long step, with all of the walker's joint angles and velocities at their target values. During approximately the first 10 seconds of simulated walking, the walker's motion oscillated away from the target trajectory to a slightly different but nonetheless stable walking motion. These plots, starting after 18 simulated seconds of walking, show the eventual stable walking trajectory and the forces that were applied to produce this motion.

A striking and unexpected result is that the trajectory is not the same for both legs. In particular, on alternate steps the toe of the support leg is raised slightly off the ground, a phenomenon which occurs because the center-of-mass of the walker is in a limit cycle, being alternately ahead of and then behind its target position, shown in Figure 5.5. Figure 5.5 should be compared to the target velocities shown in Figure 4.9. It is when a torque is applied to the support ankle to move the center-of-mass forward, that the toe of the support foot is pulled off the ground.

Figures 5.6-5.11 show the joint angles and torques for the walker's six joints. Note that these plots are for *functional* joints, e.g., support ankle. Since the support ankle switches from one simulated leg to the other at each footfall,

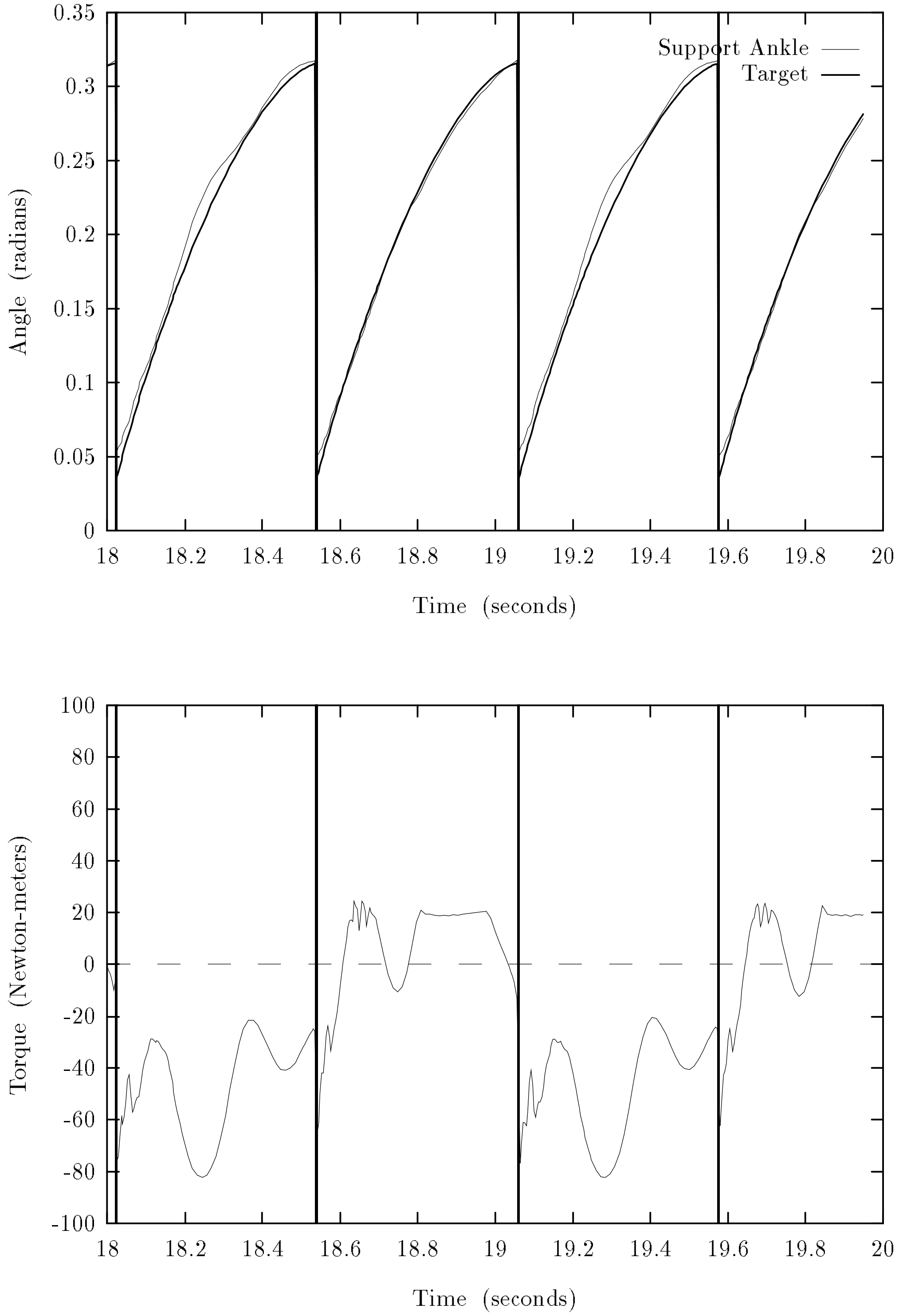

Figure 5.6: *Angles and torque on the support ankle*

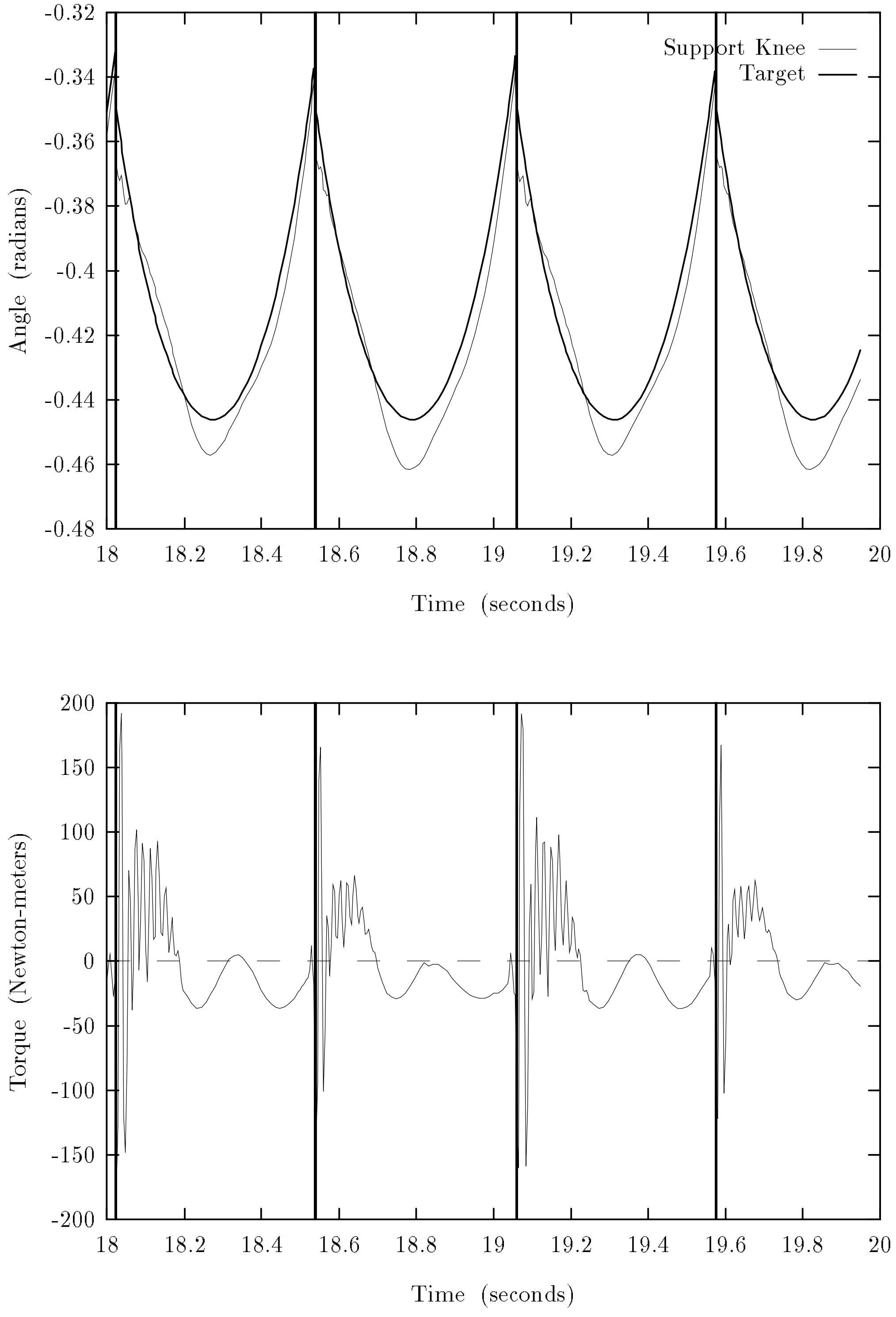

Figure 5.7: *Angles and torque on the support knee*

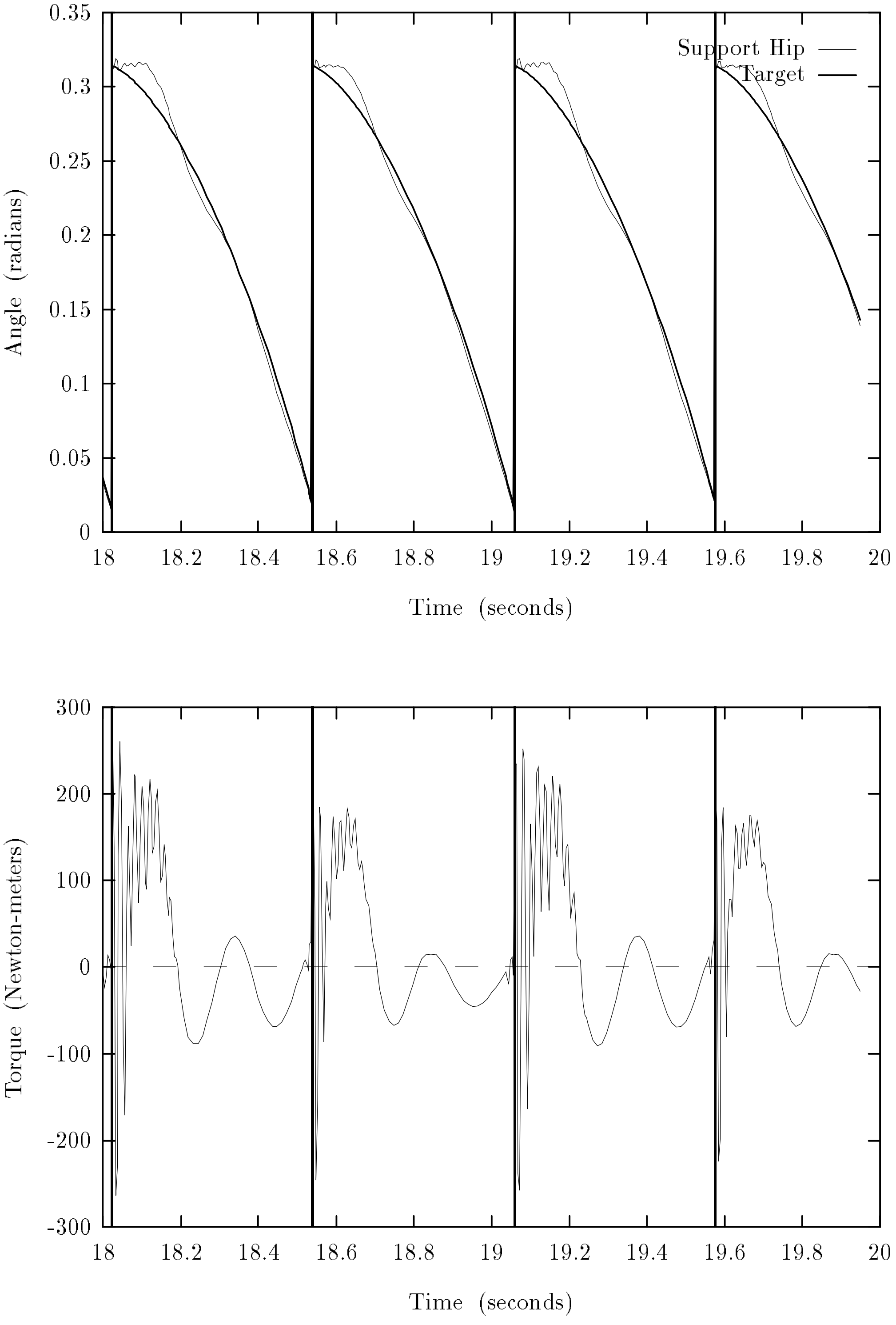

Figure 5.8: *Angles and torque on the support hip*

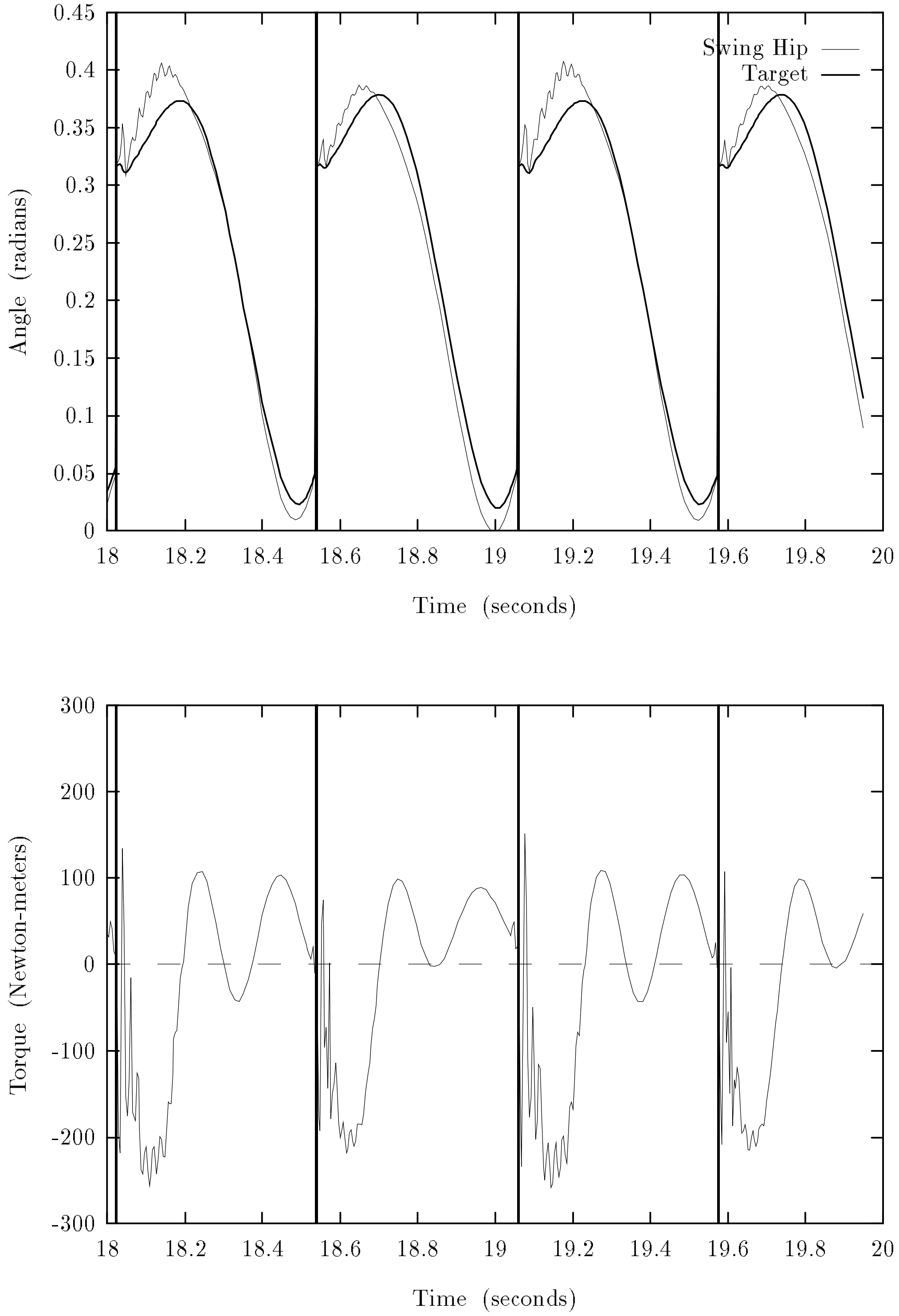

Figure 5.9: *Angles and torque on the swing hip*

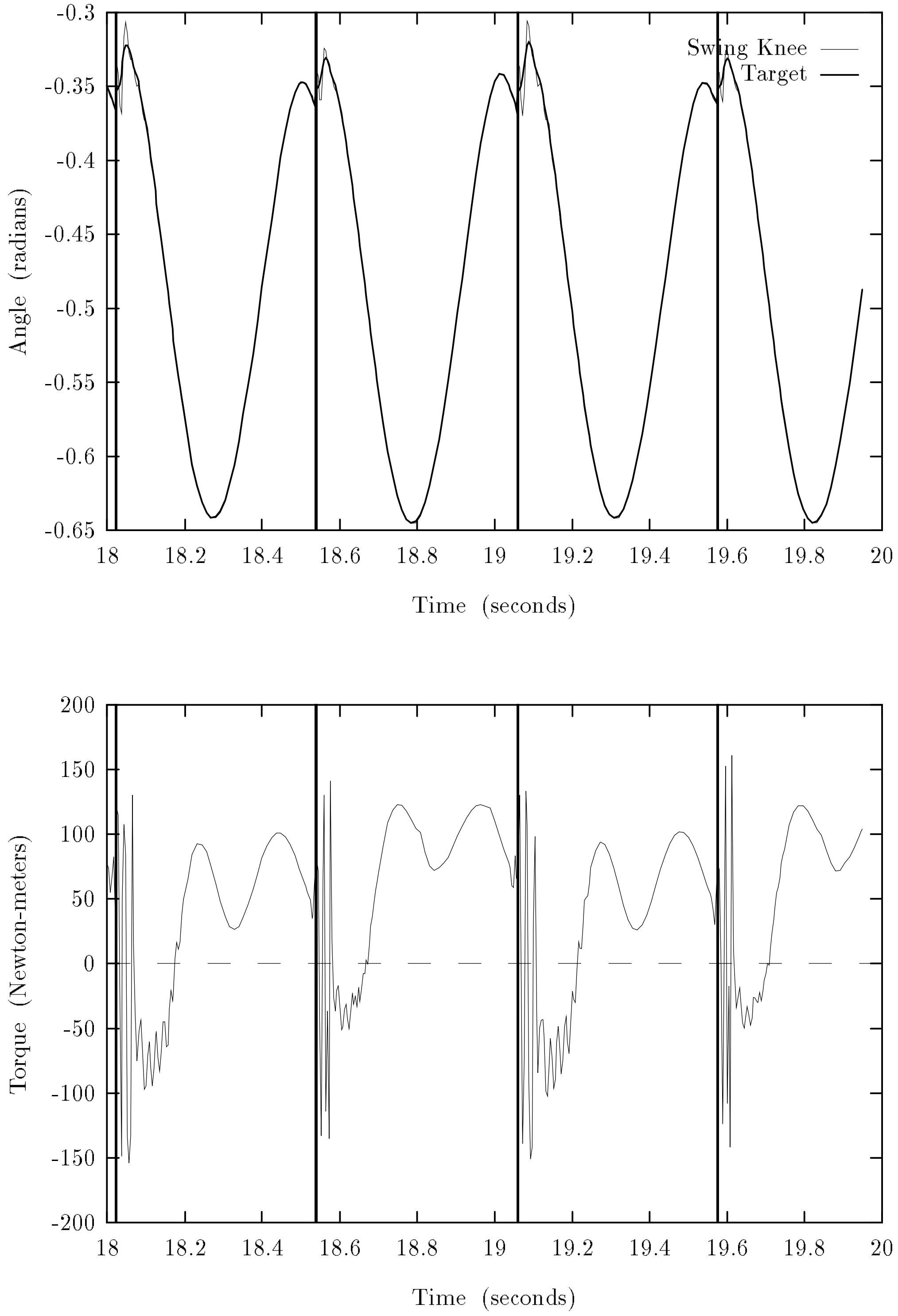

Figure 5.10: *Angles and torque on the swing knee*

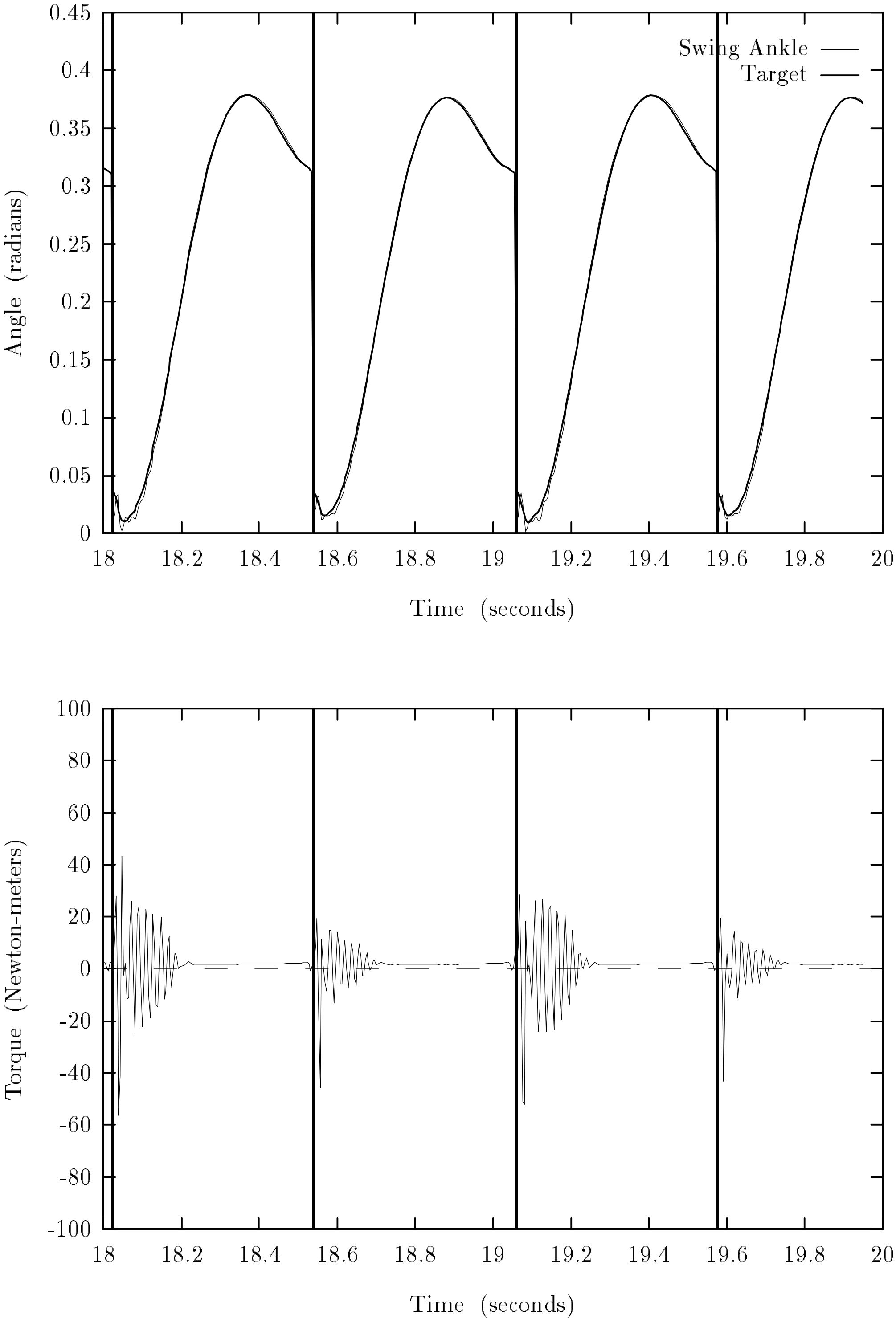

Figure 5.11: *Angles and torque on the swing foot*

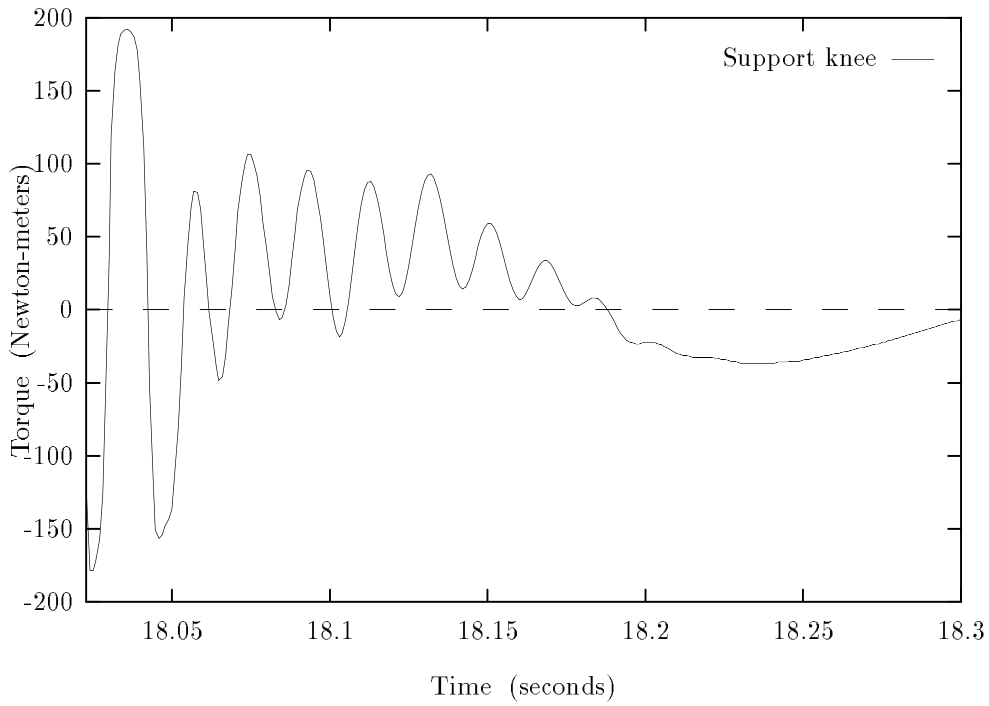

Figure 5.12: *Response of control system to a transient*

the motion and torques are discontinuous at these transitions. Also note that the vertical scale on the torque plots range from the minimum to the maximum possible torques that the joint can produce, i.e., the asymptotic bounds described in Section 5.3.

The asymmetry in the step can also be clearly seen in the torques applied to the the support ankle, Figure 5.6. For the first and third steps plotted, when the center of mass is ahead of its target position, the support ankle is continuously exerting a force to retard the motion of that ankle. By contrast, for the second and fourth steps plotted, the torque quickly becomes positive (the ankle is pushing the walker forward). When this force becomes too large the toe is lifted, causing discontinuity at approximates 18.8 seconds, appearing in the plot as the roughly flat top to the torque curve.

As has been mentioned throughout this dissertation, there were significant transient effects that occurred with each impact. Figure 5.12 shows a zoomed-in version of the walker's response to one of these transients. Two things are significant: while initially the torque applied by the controller is quite large, it damps down quickly, in approximately 0.15 seconds. Second, the oscillation is approximately 50 Hertz. While this is fast, it is slow compared to the simulation time step of $\frac{1}{1000}$ second and it is also slow enough so that it could reasonably be realized by physical actuators.

5.6 Control using neural networks

A feed-forward neural network capable of controlling the robot was also created. The network was designed so as to allow it to mimic the modified PID controller described above. The controller described in Sections 5.2-5.3 ostensibly takes a weighted sum of three error terms and the past output value and clips it using what in neural network parlance is referred to as a *sigmoid* transfer function. Although this is precisely the form that is used by a feed-forward neural network, the present study discovered that it was beneficial to have a "hidden" layer of computing units, nodes between the inputs and the outputs, in order not only to speed the learning, but also to allow better convergence of the neural network's weights for the extreme values.

The single layer feed-forward neural network was trained with the current and previous three joint errors: $e_i(t)$, $e_i(t-1)$, $e_i(t-2)$, and $e_i(t-3)$, and the previous two outputs: $u_i(t-1)$ and $u_i(t-2)$, where $1 \leq i \leq 6$. Notice that the PID controller did not use the $e_i(t-3)$ or $u_i(t-2)$ values, and while the neural network could be trained without them, it learned more quickly with these values included. The target outputs for the neural network were the six control signals generated by the controller described above.

The topology of the neural network is shown in Figure 5.13. The network had 36 inputs, 6 hidden units, and 6 outputs. There are feed-forward weighted connections between each pair of layers: connections from the inputs to both the hidden units and the outputs, and connections from the hidden units to the outputs. Thus the output units each had 43 weighted inputs: 36 from the inputs, 6 from the hidden layer, and a bias term.

The neural network was trained using the common "back-propagation" algorithm, [RHW86, Wer74], using the publicly available Aspirin/MIGRAINES software package, [LW93], developed at The MITRE Corporation by a research effort originally headed by the author. The training data were culled from a twenty-second simulation of walking (20,000 data sets) with the individual values scaled to lie in the range [-1,1].

The back-propagation training algorithm has two user defined parameters, the learning rate and the inertia. For the training session plotted in Figure 5.14 the learning rate was 0.01 and the inertia was 0.05. After every five epochs, or full training passes through the data, the network was tested on all of the data. The test measured the largest mean-squared-error (MSE) for any of the patterns.

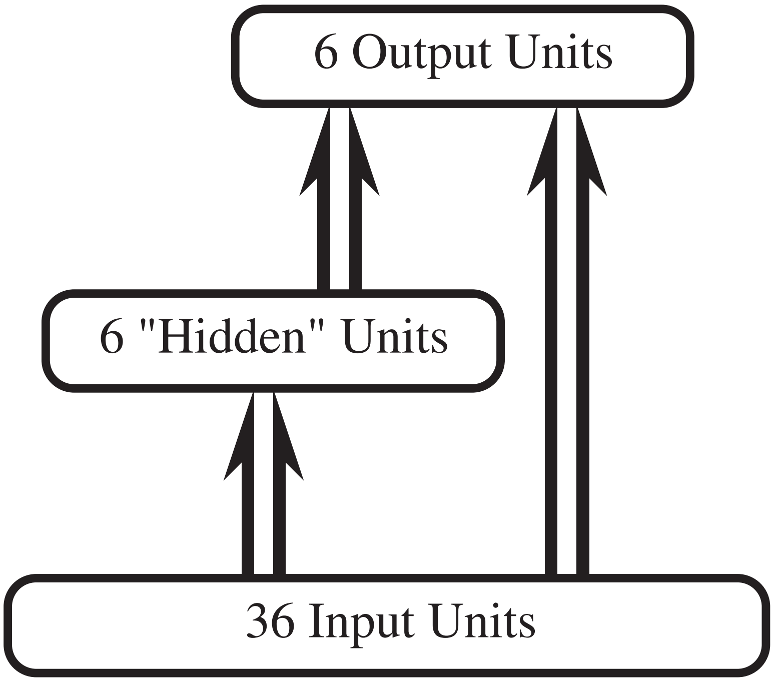

Figure 5.13: *The topology of the neural network used*

Figure 5.14 shows how the error changed as the network was trained. The best performing networking from this training session had a worst case MSE of 0.008 or 0.8%. This neural network was further refined by continuing training, first with a learning rate of 0.001 and inertia of 0.95 and later with a learning rate of 0.1 and inertia of 0.95. The result of all this training was a network for which the maximum MSE error is 0.0002 or 0.02%.

It is interesting to note that even such a small difference between the neural network and the modified PID controller was apparent in the resulting simulations. If simulated walkers were started in identical states and run for extended simulations, commonly 20 seconds, the precise positions of the two simulations drifted in relation to each other. Qualitatively the behavior of the two was the same, certainly both produced a sustained walking motion, but even the small differences in control signals made a difference in the motion of this highly non-linear system.

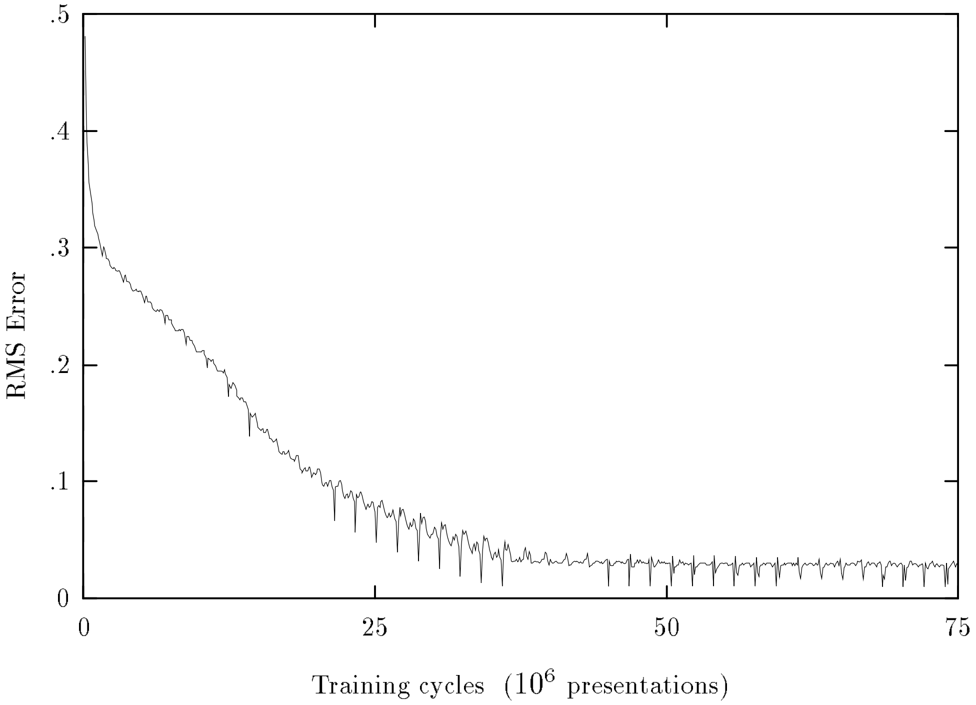

Figure 5.14: *The mean-squared-error of the network as it learned*

CHAPTER 6

Conclusions

This study here presented into the computer modeling, simulation and control of a two dimensional two-legged walking robot is in many ways unique. Walking is a deceptively hard task, both in terms of simulation and control, and while there have been past studies of walking which have considered the problem from a variety of perspectives, the significance of these past studies has in some ways been reduced by the large number of simplifications that were included. In addition to the uniqueness of having successfully created a simulation and a controller for walking, the present study is significant because of the relatively small number of simplifications that were incorporated and the resulting veracity of the results.

This study developed, from first principles, a model of walking which described the motion of the walker in any of its possible constraint states while including details of mass, friction, impacts, and other features ignored or simplified in other studies. The author developed a computer simulation that implemented the model, taking into account (1) that the model's equations are stiff, (2) that the constraints needed to be stabilized, (3) that constraints can change at arbitrary times, and (4) that nearly simultaneous impacts can interact. A target trajectory for the walker's joint angles was derived and its feasibility was verified using the computer simulation. Finally, a controller was developed which was capable of determining the torques for the actuated joints needed to produce sustained walking.

This final chapter summarizes the results and presents the unique contributions of the dissertation. The chapter is arranged as follows: first each of the components of the system, i.e., modeling walking, simulating walking, developing a target trajectory, and controlling walking are discussed in turn. Next the resulting trajectory of the walker is presented and compared to the target path. Finally, possible directions for future study are discussed.

6.1 Modeling Walking

A number of methods for deriving the differential equations governing the motion of complex physical systems exist, see Section 2.1.1. The more recent of these methods, such as the method based on Kane's Dynamics, have focused on producing equations that are efficient to simulate on a computer; clearly a desirable characteristic for a study such as this. This efficiency is gained by initially spending the effort to find optimized forms of the equations that, once derived, form the core of a subsequent fast-running computer program.

Unfortunately it was not pragmatic to use the equations of motion derived from these "modern" methods for the walker in the present study, because of the number of possible constraint states and the resultant number of distinct sets of differential equations needed to model the system's dynamics. For this walker there are 3,136 distinct constraint states. Even after removing symmetrically equivalent states there is a need for 1,596 distinct sets of differential equations. Each of these sets of equations would have to involve all the aspects of mass, friction, and other variables that characterize the walker and would only differ by how the motion was constrained. Even if all these equations were generated, the resulting program (or programs) for the computer simulation would be too large or too numerous to be executed efficiently on existing computers.

Instead, a system was developed which determined, at run time, the constraints on the walker's motion, and then added this information to a common set of equations of motion. This allowed a single relatively small software system to handle any of the possible constraint states. Implementation was achieved by determining the number and types of the constraints, creating an appropriately sized linear system of equations relating joint accelerations and constraint and stabilization forces / torques, and then solving for each time step. While such a procedure is simply an algorithmic realization of a combination of Lagrangian Mechanics (for the equations of motion) and the calculus of variations (for the constraints), the approach provides the flexibility and extensibility required by the current walker. To the best of my knowledge this approach of dynamically determining the appropriate constraints during a computer simulation and explicitly using them to efficiently determine and solve the resulting equations of motion is unique to the study here presented.

The final portion of the model involved impacts, such as when a foot of the walking robot strikes the ground. Impacts, which are an inevitable part of walking, create a myriad of problems for the simulation of the computer model. However, from the aspect of modeling, the equations for impacts are closely related to, but simpler than, the equations of motion. The equations describing impacts determine the changes in velocity and the forces throughout the walker that result from an instantaneous impact at a single distinct point. Since the changes of velocity associated with an impact must be consistent with the walker's constraint state, these impact equations were likewise modified at run time to account for the constraints. To the best of my knowledge, as with the equations of motion, this approach of dynamically determining the appropriate constraints during a computer simulation and explicitly using them to determine the resulting changes associated with an impact is unique to this study.

6.2 Simulating Walking

The model of walking described above provides a linear system of equations that, given the caveat that this system of equations is often singular, can be solved to find the walker's joint accelerations and constraint forces / torques. The computer simulation uses these accelerations, forces, and torques to determine the walker's resulting motion. This is carried out by estimating the path followed by the walker's joints and segments, finding the time that contacts and constraints are made or broken, and modeling impacts as necessary to changing the system state in response to those constraint changes. The unique features of the present simulation are its ability to handle the stiff equations and the constrained and often degenerate motion of walking, its ability to handle the changes of state that are involved with walking and the often violent dynamics that result, and its ability to stabilize constraints and distinguish numerical perturbations of the constrained variables from actual constraint changes.

The equations of motion that describe the walker are stiff; that is, the fundamental frequencies of the component motions vary widely, making simulation difficult and time-consuming. It was necessary to implement one of the existing stiff methods, described in Section 3.2, to solve these equations. Unfortunately, since these stiff methods are iterative methods that include numerically estimating the Jacobian at least once at each time step, they were inherently slow to compute. But the stiff method for solving the equations was only *necessary* when the motion of the walker actively contained high frequency transients. For the major portion of the simulation it was sufficient to use a more standard, and considerably faster, method.

This simulation therefore used both a stiff method (Bader and Deuflhard's semi-explicit extrapolation method) and a non-stiff method (Fehlberg's Runge-Kutta method) for solving the equations of motion. The initial assumption was made that the motion of the walker could be accurately estimated without resorting to the stiff method. Estimates of the error were computed and the frequencies were observed in the resulting motion. If the error became large or the motion contained high frequency components, then the equations of motion were solved a second time using the stiff methods. This allowed the simulation to be computed efficiently for the majority of the time when stiff methods were not necessary, but to switch to the more accurate and numerically stable methods when the need arose.

In addition to being stiff, there is a tendency for systems with multiple interacting constraints, such as the closed kinematic chains associated with walking to be numerically unstable. An example for the walker would be for a foot that is set on the floor to "drift" into the ground or up into the air. It was necessary to stabilize constraints while being able to distinguish the moment when a constraint is being broken (such as a foot being lifted from the ground) from when the motion from the constraint value is simply a computational perturbation. This stabilization was achieved using a method developed by Baumgarte [Bau72] that is numerically equivalent to placing aperiodic virtual springs at the constraints pulling the system back to their correct position. Constraints were *removed* based on the forces (excluding the forces in the virtual springs) experienced at the constraint sites. This combination of constraint stabilization and dynamic changing of constraints appears to be unique to the current study.

Conceptually, changes to the constraint state were carried out by observing the state of the system at both the beginning and end of each time step and noting if any constraints had become newly violated. If they had, interpolation was used to determine an estimate of when the constraint was reached and, when necessary, this revised estimate was recursively used to find a more accurate estimate. An impact was then used to remove the component of motion into the constraint.

Two general problems exist with this simple approach to adding and removing constraints. First, since this approach must move *past* a constraint in order to find the time that it occurred, it is necessary for the model to produce meaningful and consistent results even in this "illegal" range of motion. The example discussed in Section 3.3 involves the singularity in the static friction as the normal force of the foot passes through zero. It is necessary to assure that these singularities do not produce paradoxical or difficult-to-simulate results.

The second general problem involves *hidden* impacts – that is, impacts that occur entirely within a time step and are therefore not apparent if only the ends of the time step are considered. It was necessary, in order to maintain even qualitatively correct behavior, for these hidden impacts to be detected and for appropriate changes to the walker's motion and state to be carried out. Hidden impacts were detected by using two-point second-order Hermite interpolation to determine the path of each state variable within each time step. If any inflection points occurred along the interpolated path within the time step, and if the interpolated state at these inflection points indicated that an impact may have occurred within that time step, then the time step was recomputed with the inflection point as the new end to that simulated time step. Both the need to detect hidden impacts and the method for doing so are unique to this study.

6.3 Developing a Target Trajectory

Given the computer simulation, two steps were involved in making the computer simulated walker walk. First a desired or target trajectory for the walker was developed and then a controller was developed to make the simulated walker follow the target trajectory. The most dramatic improvements in this phase of the project came from providing a trajectory that could be more easily followed. The basis of the trajectory that was eventually used successfully for this study is fundamentally different from what has been commonly used in past studies.

Past studies of walking have assumed that the walker can be adequately modeled as an inverted pendulum. In other words, the motion of the walker can be adequately approximated to be a mass with fixed rotational inertia rotating around an unpowered fixed pivot. This is in fact the mathematical model that is derived when the equations of motion are linearized based on small angle and small velocity assumptions and when friction and impacts are ignored. Unfortunately, for the inverted pendulum model to be adequate, it is necessary for angles and velocities to remain smaller than is appropriate for a normal walking gait. In retrospect it is now clear why that small number of walking robots which have been built and controlled based on these assumptions have been limited to taking very short steps, with strides on the order of one degree.

Trajectories based on the simple inverted pendulum model were found to make unacceptable demands on the current walker's controller. A presumed strength of trajectories based on the inverted pendulum model was that since they were derived assuming an unpowered pivot, the support ankle would require the application of little or no torque. However, because of the motion of the swing leg and the vertical position of the body, the simplified mathematical model has proven not to be sufficiently accurate, and following this trajectory in fact requires considerable torques. For the relatively long strides that were modeled here (eight degrees), the author was never able to develop a controller capable of producing adequate torque to move the center-of-mass to follow an inverted pendulum trajectory, without producing so much torque that either the support foot was pulled out from under the walker or the center-of-mass was thrown too far forward, eventually causing the walker to fall.

The eventually successful target trajectory was based on two different assumptions, that the body of the walker should move at a more constant velocity, and that the number and strength of the impacts should be minimized. The switch from a target trajectory based on the inverted pendulum model to a target trajectory based on constant velocity made an immediate and dramatic change in the ability of the controller of the simulated walker to produce the desired motions. While the torques required to carry out the motion were reduced throughout the step, this change was most obvious just before and just after a footfall.

In addition to the horizontal velocity of the base of the walker's body, the height of the body from the ground was also kept constant. Not only did this minimize the energy required first to raise and then lower the center-of-mass during each step, it also helped in reducing the number of impacts during a simulation. Note again that instead of controlling the height of the center-of-mass, the goal of the target trajectory in this study was for the height of the body to remain constant.[1]

It is important to point out that two parts of the target trajectory involved feedback. First the target trajectory called for the body to remain vertical. This was achieved by setting the error of the support hip equal to the angle of the body from vertical, a feedback loop. Second, the target trajectory for the swing foot dictated that it remain horizontal and that it touch the ground with zero velocity. This was also achieved using feedback, comparing the height and velocity of the walker's foot to the desired one.

For the short steps, small angles, and simplified models used by previous studies, the difference between a trajectory based on an inverted pendulum model and the constant velocity model described here are minimal. For the model in this study however, with its comparatively long step lengths and top-heavy body, the difference between the approaches that have been used in the past and the new approach used by the author was dramatic and central to making walking possible.

[1] Note that since humans effectively compute acceleration and angular displacement of their head's position, basing the trajectory on these factors would also seem to be biologically feasible.

6.4 Controlling Walking

The ultimate goal of this project was to make the simulated robot walk. Given a target trajectory, it was the task of the controller to determine the torques that should be applied to the walker's six actuated joints to cause the walking to occur. The controller was based on the standard proportional-integral-derivative (PID) control model. This was modified slightly to enable the controller to handle the large perturbations caused by the footfalls and to allow for a relatively smooth transition from being supported by one leg to being supported by the other. Finally a simple feed-forward neural network was produced which was able to control the walker.

At each instant of simulated time, the position of each of the walker's six joint angles was compared to the target trajectory, the difference being each joint's error. A control signal was generated for each of the six joints, describing the appropriate actuation for that joint based on a linear combination of that joint's current error and the derivative and integral of that error. This is the standard PID control paradigm. Digital implementations of PID control, such as the one used here, require dumping of the integral term to assure that it can not grow too large and careful calculations of the integral for stability.

The simple PID model had to be modified because of the special demands of walking. The inevitable perturbations caused by footfalls and errors in position that resulted from the transitions from one support leg to the other occasionally caused the simple PID controller to request extremely large torques. For physical feasibility as well as removal of the violent motions that would result from these large torques, the range of possible torques was limited. Since, as discussed below, the control parameters were tuned using adaptive techniques, it was desirable to use an asymptotic bound rather than an absolute bound that simply clipped the control signal. In addition to producing a more realistic and more resilient controller, this asymptotic bound simplified the eventual conversion to a neural network controller.

The creation of a controller tacitly assumes that the system being controlled will consistently react to the control signals. Since the behavior of the walker is different in each of its 3,136 distinct constraint states, it might seem necessary to have a separate controller for each state and to devise a way of making a smooth transition between them. However, not only is this number of controllers impractically large, but the necessary transition between controllers rarely occurs smoothly. Therefore, the walker in this study assumes one control model, nominally designed for the most common constraint state during walking (one foot flat on the floor and the other entirely off the floor). The assumption is that insufficient time is spent in other constraint states to warrant either distinct controllers or the disruption of a transition.

The transition associated with changing between support feet is handled by reinitializing the six controllers to zero. That is, the integral and derivative of the error are reset to zero so that the first control signal after an impact is based solely on the joint errors, and it does produce significant transient behavior at the transition, but it was found, nonetheless, to be satisfactory. This approach has the important advantage of decoupling the motion before and after the transition.

The performance of the modified PID controller was governed by 18 parameters; a set of three parameters – for the error, its derivative, and its integral – for each of the walker's six joints. A central task in developing the controller was finding these 18 parameters. A number of techniques exist for estimating the control parameters for PID controllers, and the author used a simple traditional hand tuning technique to create initial approximations for these parameters, then fine tuned them using a series of multi-variate nonlinear optimization techniques. These were generalized gradient descent paradigms that were modified to meet the particular needs of adjusting PID control parameters (described below).

An interesting property noted while using optimization techniques to tune the control parameters was the tendency for the resulting controllers to be unstable. Since the fundamental frequency of some of the error modes was comparable to the time of an individual step, it was easier to reduce the error of these modes by decreasing the frequency rather than decreasing the magnitude of the error. However, while this approach minimized the error within an individual step, these parameters did poorly on longer simulations, and the highly unstable controllers that were sometimes produced performed catastrophically poorly on intervals that contained significant perturbations.

It thus became necessary to adjust the parameters slightly, determine how much longer a time interval could then be reliably simulated, increase the time window, and repeat. The longer simulations were not possible until reasonably good estimates had been made of the parameters, but the "best" parameters for the shorter simulations did not work well for longer simulations. It was thus necessary to modify the optimization techniques in order to allow gradual refinement at each step rather than initially attempting to find the *best* parameters.

One of the original goals of this study was to use neural networks to control walking, but by the end of the project this had become a secondary concern. Nonetheless, a feed-forward neural network was created that could replicate the performance of the modified PID controller. Note that because of the clipped output of the PID controller it was possible to replicate the performance of the PID controller precisely. And, since the input errors and the output control signals were used as the training signals to the controller, the resulting neural network performed almost precisely the same as the modified PID controller and is therefore of limited interest.

6.5 Areas for Future Work

This project was a success in that a computer simulation of a walking robot was created and a controller capable of causing that simulated robot to walk was developed. In many ways, though, a project like this produces more questions than it answers. At every phase of this project there were additional areas that could have been pursued, and this section presents some of these areas and speculates, based on the last four years' work, upon their relative difficulty and importance, and some probable results.

6.5.1 Modeling and Simulation

It was tempting to make the model and resulting simulation of walking as accurate as possible. Certainly this system, being two dimensional and without any compliance, could have been extended. But it is not always clear that these more physically accurate extensions would have added significantly to the study.

One of the most significant deficiencies of the simulation, and one that had ramifications for the entire study, was that the walker was not compliant. Not including compliance decreased the degrees of freedom in the system and therefore nominally simplified the motion and thus reduced the simulation time. However, impacts in this non-compliant walker were characterized by significant transients, and adding compliance could have softened this effect. In the long term it probably would be a useful extension to add compliance, more because it could be used to minimize the difficulties associated with energetic transients than because it would add significantly to the veracity of the model.

The model of walking considered here is two dimensional. It is not clear that modeling a three dimensional robot would provide any more insight or understanding than modeling a two dimensional robot. Because walking generally involves little motion in the transverse plane (the plane perpendicular to the walking motion), motion and control in this plane *can* be reasonably modeled using a highly simplified linear (that is inverted pendulum) model. Also, the motion in the transverse plane is largely decoupled from the motion in the sagital plane (the plane of motion modeled in this study) so this adds little to the forward motion of the walking robot. It remains my opinion that unless three dimensional motion is being explicitly studied, as for example in the ability of a walking robot to turn, there are more fruitful extensions which should be made before incorporating the third dimension.

Moreover, while adding the third dimension is both conceptually and mathematically straightforward, it also brings with it a host of associated difficulties. For example, the simple system of planar motion and revolute joints is no longer valid, with joints that allow more than one degree of freedom becoming desirable. In addition to adding one more degree of translational motion, the third dimension adds two more degrees of rotational motion. Also a question arises as how to adequately model the feet. In this two dimensional study, feet are modeled as having two contact points, one at the toe and one at the heel. Additional points are required if the foot is to provide any lateral support in the simulated three dimensional world; but this exponentially increases the number of impact interactions that can occur as well as the number of possible singularities in the constraint matrix.

6.5.2 Trajectories and Control

Probably the hardest part of controlling walking is knowing what is a reasonable target trajectory for the walker to follow. The trajectory used in the present study kept the body moving horizontally with a constant velocity, and while the eventual motion of the walker's body is nearly constant, there is periodic swinging. Certainly it seems that humans use the swaying of their bodies as an intentional part of the walking trajectory, and it is reasonable to think that if the "correct" swing could be found and used as the target for the body's motion, a further improvement could be realized.

This walker was only required to walk on a smooth level surface, but clearly it is desirable for a walking robot to be able to navigate more complex terrain. Running, which by definition involves instants where both feet have left the ground, would bring further complications. It is not clear how the trajectory should be extended to handle these cases, but the success brought to this project from improving the target trajectory would indicate that an important issue in any future work along these paths would involve finding an appropriate trajectory.

One option would be to use adaptive techniques to find a good target trajectory. The author has had success using genetic algorithms to find controllers for complex motion, see [Wie91a, Wie91b], and a similar approach could be used to optimize the target trajectory. The results from an adaptive approach like this would produce a good trajectory for a specific walker in a specific walking environment. Ideally results such as these would then be analyzed in an attempt to find a generally good method that could be applied to the wide variety of walking situations.

6.5.3 Physical Realizations

It is enticing to consider creating a physical realization of the walker. Certainly this would be seen as a validation of the results and would be a tangible and entertaining demonstration of the principles involved. However, a walking robot suffers from its own set of limitations, and a computer simulation certainly allows more rapid changes in the basic form and ability of the walker. Such flexibility was important to this study, and would probably be equally important to further work along these lines. It is not clear, when such fundamental questions are still being addressed, that a physical realization would add significant understanding or insight into the problem.

APPENDIX A

Equations of Motion

During the years that this study was underway there have been repeated requests to see the underlying equations-of-motion. However the equations used in this study are large enough and complex enough that the considerable effort and hundreds of pages that would be required to print them would not provide any additional understanding of walking or this study. Nonetheless, this appendix attempts provide a more solid understanding of these equations and how they were generated.

This appendix is divided into two sections. The first section presents two of the initial equations used in formulating the equations of motion: the potential energy V and the kinetic energy T. These are the basic building blocks used in deriving the equations of motion, and at a length of twelve pages give a relatively short and easily understood appreciation of the equations involved. The second section shows the MapleV code that was used to derive the equations of motion. This code shows how the equations in Chapter 2 were formulated on a computer and then solved.

A.1 Sample Equations

The following are the equations for the potential energy V and the kinetic energy T. The symbols and notation used in these equations are the same as is used in Chapter 2 of this dissertation. Appendix B lists the values of the constants.

Note that these equations were formatted by MapleV [CGG91], that is by a computer, with only limited editing by the author. As a result some "obvious" simplifications and rearrangements of terms were not made. As will be pointed out in Section A.2, the computer code that MapleV eventually generated and that was used in the computer simulation of walking was simplified and optimized for computational speed. Unfortunately these "simplifications" do not make the equations easier to read.

$$
\begin{aligned}
V = -\tfrac{g}{2} \Big[& -2\, m_1\, y - 4\, m_2\, y - 4\, m_3\, y - 4\, m_4\, y \\
& + m_1\, l_1\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& + 2\, m_1\, l_3\, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& + 2\, m_1\, l_2\, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& + 4\, m_2\, l_3\, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& + 3\, m_2\, l_2\, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& - m_2\, l_2\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& + 3\, m_3\, l_3\, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& + 2\, m_3\, l_2\, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& - 2\, m_3\, l_2\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& - m_3\, l_3\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
& - m_4\, l_4\, \cos(\theta_{\mathrm{foot1}}) \\
& + 2\, m_4\, l_3\, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& + 2\, m_4\, l_2\, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& - 2\, m_4\, l_2\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& - 2\, m_4\, l_3\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
& - m_4\, l_4\, \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& + 2\, m_4\, \gamma\, l_4\, \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& - 2\, m_1\, \gamma\, l_4\, \cos(\theta_{\mathrm{foot1}}) \\
& - 4\, m_2\, \gamma\, l_4\, \cos(\theta_{\mathrm{foot1}}) \\
& - 4\, m_3\, \gamma\, l_4\, \cos(\theta_{\mathrm{foot1}}) \\
& - 2\, m_4\, \gamma\, l_4\, \cos(\theta_{\mathrm{foot1}}) \Big]
\end{aligned}
$$

$$
\begin{aligned}
T = \tfrac{m_1}{6} \Big[\; \Big\{ & \dot{x} + \gamma\, l_4\, \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& - l_3\, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& - l_2\, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& - l_1\, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \\
& \quad \times \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \Big\}^2
\end{aligned}
$$

$$
\begin{aligned}
+ \ \Big\{ \ & \dot{y} - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_1 \ \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \ \Big\}^2 \\
+ \ \Big\{ \ & \dot{x} + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_1 \ \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \ \Big\} \\
& \times \Big\{ \ \dot{x} + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& \quad - l_3 \ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& \quad - l_2 \ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \\
& \quad\quad \times \ \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \ \Big\} \\
+ \ \Big\{ \ & \dot{y} - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_1 \ \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \ \Big\} \\
& \times \Big\{ \ \dot{y} - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& \quad - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& \quad - l_2 \ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \\
& \quad\quad \times \ \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \ \Big\}
\end{aligned}
$$

$$
\begin{aligned}
&+\Big\{\ \dot{x} + \gamma\, l_4\ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad - l_3\ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad - l_2\ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right)\ \Big\}^2 \\
&+\Big\{\ \dot{y} - \gamma\, l_4\ \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad - l_3\ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad - l_2\ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \\
&\qquad\quad \times\ \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right)\ \Big\}^2\ \Big] \\
+\ \tfrac{m_2}{6}\Big[\ &\Big\{\ \dot{x} + \gamma\, l_4\ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\quad - l_3\ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\quad - l_2\ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right)\ \Big\}^2 \\
&+\Big\{\ \dot{y} - \gamma\, l_4\ \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad - l_3\ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad - l_2\ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right)\ \Big\}^2 \\
&+\Big\{\ \dot{x} + \gamma\, l_4\ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad - l_3\ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad - l_2\ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right)\ \Big\} \\
&\quad \times \Big\{\dot{x} + \gamma\, l_4\ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad\quad - l_3\ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}})\left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right)\ \Big\}
\end{aligned}
$$

$$
\begin{aligned}
&+ \Big\{ \dot{y} - \gamma\, l_4 \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\quad - l_3 \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\quad - l_2 \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big\} \\
&\quad \times \Big\{ \dot{y} - \gamma\, l_4 \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad - l_3 \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big\} \\
&+ \Big(\dot{x} + \gamma\, l_4 \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\quad - l_3 \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big)^2 \\
&+ \Big(\dot{y} - \gamma\, l_4 \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\quad - l_3 \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big)^2 \Big] \\
+ \tfrac{m_2}{6} \Big[&\Big\{ \dot{x} + \gamma\, l_4 \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} - l_3 \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\quad - l_2 \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big\}^2 \\
&+ \Big\{ \dot{y} - \gamma\, l_4 \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} - l_3 \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\quad - l_2 \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big\}^2 \\
&+ \Big\{ \dot{x} + \gamma\, l_4 \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} - l_3 \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\quad - l_2 \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big\} \\
&\quad \times \Big\{ \dot{x} + \gamma\, l_4 \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} - l_3 \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad - l_2 \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad + l_2 \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
&\qquad\quad \times \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}}\right) \Big\} \\
&+ \Big\{ \dot{y} - \gamma\, l_4 \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\quad - l_3 \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\quad - l_2 \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \Big\} \\
&\quad \times \Big\{ \dot{y} - \gamma\, l_4 \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
&\qquad - l_3 \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad - l_2 \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
&\qquad + l_2 \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}})
\end{aligned}
$$

$$
\begin{aligned}
&\qquad\quad \times \ \left(\dot\theta_{\text{hip1}} + \dot\theta_{\text{knee1}} + \dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}} - \dot\theta_{\text{hip2}}\right) \Big\} \\
&+ \Big\{\dot x + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot\theta_{\text{foot1}} \\
&\quad - l_3 \ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right) \\
&\quad - l_2 \ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{knee1}} + \dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right) \\
&\quad + l_2 \ \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
&\qquad \times \ \left(\dot\theta_{\text{hip1}} + \dot\theta_{\text{knee1}} + \dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}} - \dot\theta_{\text{hip2}}\right) \Big\}^2 \\
&+ \Big\{\dot y - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot\theta_{\text{foot1}} \\
&\quad - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right) \\
&\quad - l_2 \ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{knee1}} + \dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right) \\
&\quad + l_2 \ \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
&\qquad \times \ \left(\dot\theta_{\text{hip1}} + \dot\theta_{\text{knee1}} + \dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}} - \dot\theta_{\text{hip2}}\right) \Big\}^2 \ \Big] \\
+ \tfrac{m_3}{6} \Big[\ &\Big(\dot x + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot\theta_{\text{foot1}} \\
&\quad - l_3 \ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right)\Big)^2 \\
&+ \Big(\dot y - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot\theta_{\text{foot1}} \\
&\quad - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right)\Big)^2 \\
&+ \Big(\dot x + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot\theta_{\text{foot1}} \\
&\quad - l_3 \ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right)\Big) \\
&\quad \times \ \left(\dot x + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot\theta_{\text{foot1}}\right) \\
&+ \Big(\dot y - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot\theta_{\text{foot1}} \\
&\quad - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot\theta_{\text{ankle1}} + \dot\theta_{\text{foot1}}\right)\Big) \\
&\quad \times \ \left(\dot y - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot\theta_{\text{foot1}}\right) \\
&+ \left(\dot x + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot\theta_{\text{foot1}}\right)^2 \\
&+ \left(\dot y - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot\theta_{\text{foot1}}\right)^2 \ \Big]
\end{aligned}
$$

$$
\begin{aligned}
+ \tfrac{m_3}{6} \Bigg[\quad & \Big\{ \dot{x} + \gamma\, l_4 \, \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& \quad - l_3 \, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \quad - l_2 \, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \quad + l_2 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \qquad \times \ \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \ \Big\}^2 \\
& + \Big\{ \ \dot{y} - \gamma\, l_4 \, \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& \qquad - l_3 \, \cos(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \qquad - l_2 \, \cos(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \qquad + l_2 \, \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \qquad\quad \times \ \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \ \Big\}^2 \\
& + \Big\{ \ \dot{x} + \gamma\, l_4 \, \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& \qquad - l_3 \, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \qquad - l_2 \, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \qquad + l_2 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \qquad\quad \times \ \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \ \Big\} \\
& \quad \times \ \Big\{ \ \dot{x} + \gamma\, l_4 \, \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& \qquad\quad - l_3 \, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \qquad\quad - l_2 \, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& \qquad\quad + l_2 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \qquad\qquad \times \ \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \\
& \qquad\quad + l_3 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
& \qquad\qquad \times \ \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}}\right) \ \Big\}
\end{aligned}
$$

$$
\begin{aligned}
&+\Big\{\ \dot{y}-\gamma\, l_4\ \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\\
&\qquad -l_3\ \cos(\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}})\left(\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}\right)\\
&\qquad -l_2\ \cos(\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}})\left(\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}\right)\\
&\qquad +l_2\ \cos(\theta_{\mathrm{hip1}}+\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta-\theta_{\mathrm{hip2}})\\
&\qquad\quad \times\ \left(\dot{\theta}_{\mathrm{hip1}}+\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}-\dot{\theta}_{\mathrm{hip2}}\right)\ \Big\}\\
&\quad\times\ \Big\{\ \dot{y}-\gamma\, l_4\ \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\\
&\qquad\quad -l_3\ \cos(\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}})\left(\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}\right)\\
&\qquad\quad -l_2\ \cos(\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}})\left(\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}\right)\\
&\qquad\quad +l_2\ \cos(\theta_{\mathrm{hip1}}+\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta-\theta_{\mathrm{hip2}})\\
&\qquad\qquad \times\ \left(\dot{\theta}_{\mathrm{hip1}}+\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}-\dot{\theta}_{\mathrm{hip2}}\right)\\
&\qquad\quad +l_3\ \cos(\theta_{\mathrm{hip1}}+\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}}-\theta_{\mathrm{hip2}}-\theta_{\mathrm{knee2}})\\
&\qquad\qquad \times\ \left(\dot{\theta}_{\mathrm{hip1}}+\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}-\dot{\theta}_{\mathrm{hip2}}-\dot{\theta}_{\mathrm{knee2}}\right)\ \Big\}\\
&+\Big\{\ \dot{x}+\gamma\, l_4\ \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\\
&\qquad -l_3\ \sin(\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}})\left(\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}\right)\\
&\qquad -l_2\ \sin(\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}})\left(\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}\right)\\
&\qquad +l_2\ \sin(\theta_{\mathrm{hip1}}+\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta-\theta_{\mathrm{hip2}})\\
&\qquad\quad \times\ \left(\dot{\theta}_{\mathrm{hip1}}+\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}-\dot{\theta}_{\mathrm{hip2}}\right)\\
&\qquad +l_3\ \sin(\theta_{\mathrm{hip1}}+\theta_{\mathrm{knee1}}+\theta_{\mathrm{ankle1}}+\theta_{\mathrm{foot1}}-\theta_{\mathrm{hip2}}-\theta_{\mathrm{knee2}})\\
&\qquad\quad \times\ \left(\dot{\theta}_{\mathrm{hip1}}+\dot{\theta}_{\mathrm{knee1}}+\dot{\theta}_{\mathrm{ankle1}}+\dot{\theta}_{\mathrm{foot1}}-\dot{\theta}_{\mathrm{hip2}}-\dot{\theta}_{\mathrm{knee2}}\right)\ \Big\}^2
\end{aligned}
$$

$$
\begin{aligned}
&+ \Big\{ \dot{y} - \gamma\, l_4 \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
&\qquad - l_3 \cos(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
&\qquad - l_2 \cos(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
&\qquad + l_2 \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
&\qquad\quad \times \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \\
&\qquad + l_3 \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
&\qquad\quad \times \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}}\right) \Big\}^2 \ \Big]
\end{aligned}
$$

$$
\begin{aligned}
+ \tfrac{m_4}{6} \Big[\quad & \left(\dot{x} + l_4 \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\right)^2 \\
& + \left(\dot{y} - l_4 \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\right)^2 \\
& + \left(\dot{x} + l_4 \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\right) \dot{x} \\
& + \left(\dot{y} - l_4 \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}}\right) \dot{y} \\
& + \dot{x}^2 + \dot{y}^2 \ \Big]
\end{aligned}
$$

$$
\begin{aligned}
+ \tfrac{m_4}{6} \Big[\ & \Big\{ \dot{x} + \gamma\, l_4 \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& - l_3 \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& - l_2 \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& + l_2 \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \quad \times \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \\
& + l_3 \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
& \quad \times \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}}\right) \\
& + l_4\, (1 - \gamma) \\
& \quad \times \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& \quad \times \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}} - \dot{\theta}_{\mathrm{ankle2}}\right) \Big\}^2
\end{aligned}
$$

$$
\begin{aligned}
+ \ \Big\{ \ & \dot{y} - \gamma\, l_4 \ \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \ \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \ \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& + l_2 \ \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}}\right) \\
& + l_3 \ \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}} - \theta_{\text{hip2}} - \theta_{\text{knee2}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}}\right) \\
& - l_4 \ (1-\gamma) \\
& \quad \times \ \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \ \Big(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} \\
& \qquad\quad - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\Big) \ \Big\}^2 \\
+ \ \Big\{ \ & \dot{x} + \gamma\, l_4 \ \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \ \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \ \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& + l_2 \ \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}}\right) \\
& + l_3 \ \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}}\right) \\
& + l_4 \ (1-\gamma) \\
& \quad \times \ \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \ \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\right)\Big\}
\end{aligned}
$$

$$
\begin{aligned}
\times \Big\{ \; & \dot{x} + \gamma \, l_4 \, \cos(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& - l_3 \, \sin(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& - l_2 \, \sin(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& + l_2 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \quad \times \; \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \\
& + l_3 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}} - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
& \quad \times \; \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}}\right) \\
& + l_4 \, (1 - \gamma) \\
& \quad \times \; \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& \quad \times \; \Big(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} \\
& \qquad \quad - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}} - \dot{\theta}_{\mathrm{ankle2}}\Big) \\
& - l_4 \, \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& \quad \times \; \Big(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} \\
& \qquad \quad - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}} - \dot{\theta}_{\mathrm{ankle2}}\Big) \; \Big\}
\end{aligned}
$$

$$
\begin{aligned}
+ \Big\{ \; & \dot{y} - \gamma\, l_4 \, \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \, \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \, \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& + l_2 \, \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
& \quad \times \; \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}}\right) \\
& + l_3 \, \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}} - \theta_{\text{hip2}} - \theta_{\text{knee2}}) \\
& \quad \times \; \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}}\right) \\
& - l_4 \, (1 - \gamma) \\
& \quad \times \; \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \; \Big(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} \\
& \qquad\quad - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\Big) \; \Big\}
\end{aligned}
$$

$$
\begin{aligned}
\times \Big\{ \; & \dot{y} - \gamma\, l_4 \, \sin(\theta_{\mathrm{foot1}})\dot{\theta}_{\mathrm{foot1}} \\
& - l_3 \, \cos(\theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& - l_2 \, \cos(\theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}}) \left(\dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}}\right) \\
& + l_2 \, \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}}) \\
& \quad \times \; \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}}\right) \\
& + l_3 \, \cos(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta_{\mathrm{foot1}} - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}}) \\
& \quad \times \; \left(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}}\right) \\
& - l_4 \, (1 - \gamma) \\
& \quad \times \; \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& \quad \times \; \Big(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} \\
& \qquad\quad - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}} - \dot{\theta}_{\mathrm{ankle2}}\Big) \\
& + l_4 \, \sin(\theta_{\mathrm{hip1}} + \theta_{\mathrm{knee1}} + \theta_{\mathrm{ankle1}} + \theta - \theta_{\mathrm{hip2}} - \theta_{\mathrm{knee2}} - \theta_{\mathrm{ankle2}}) \\
& \quad \times \; \Big(\dot{\theta}_{\mathrm{hip1}} + \dot{\theta}_{\mathrm{knee1}} + \dot{\theta}_{\mathrm{ankle1}} + \dot{\theta}_{\mathrm{foot1}} \\
& \qquad\quad - \dot{\theta}_{\mathrm{hip2}} - \dot{\theta}_{\mathrm{knee2}} - \dot{\theta}_{\mathrm{ankle2}}\Big) \; \Big\}
\end{aligned}
$$

$$
\begin{aligned}
+ \; \Big\{ \; & \dot{x} + \gamma \, l_4 \, \cos(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \, \sin(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \, \sin(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& + l_2 \, \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
& \quad \times \; \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}}\right) \\
& + l_3 \, \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}} - \theta_{\text{hip2}} - \theta_{\text{knee2}}) \\
& \quad \times \; \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}}\right) \\
& + l_4 \, (1 - \gamma) \\
& \quad \times \; \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \; \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\right) \\
& - l_4 \, \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \; \Big(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} \\
& \qquad\quad - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\Big) \; \Big\}^2
\end{aligned}
$$

$$
\begin{aligned}
+ \Big\{ & \dot{y} - \gamma\, l_4 \, \sin(\theta_{\text{foot1}})\dot{\theta}_{\text{foot1}} \\
& - l_3 \, \cos(\theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& - l_2 \, \cos(\theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}}) \left(\dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}}\right) \\
& + l_2 \, \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}}) \\
& \quad \times \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}}\right) \\
& + l_3 \, \cos(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta_{\text{foot1}} - \theta_{\text{hip2}} - \theta_{\text{knee2}}) \\
& \quad \times \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}}\right) \\
& - l_4 \, (1 - \gamma) \\
& \quad \times \, \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \left(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\right) \\
& + l_4 \, \sin(\theta_{\text{hip1}} + \theta_{\text{knee1}} + \theta_{\text{ankle1}} + \theta - \theta_{\text{hip2}} - \theta_{\text{knee2}} - \theta_{\text{ankle2}}) \\
& \quad \times \Big(\dot{\theta}_{\text{hip1}} + \dot{\theta}_{\text{knee1}} + \dot{\theta}_{\text{ankle1}} + \dot{\theta}_{\text{foot1}} - \dot{\theta}_{\text{hip2}} \\
& \qquad\quad - \dot{\theta}_{\text{knee2}} - \dot{\theta}_{\text{ankle2}}\Big) \Big\}^2 \ \Bigg]
\end{aligned}
$$

A.2 MapleV code

This section provides the MapleV code that was used to generate the equations of motion. While simply presenting computer code generally adds little to the understanding of a system, it was felt that the MapleV code was concise enough enough to provide some helpful information. Further, since it is generally *not* reasonable to present the actual equations of motion because of their large size, this was one of the only ways of presenting the equations in a form that could be easily reproduced.

Short explanations will accompany each of the code segments, in addition to the brief comments in the code itself.

Maple allows for partial differentiation, but to allow *full* differentiation with respect to t it was necessary to find a way of telling MapleV which variables were functions of t. The following segment contains the definitions that were used for two functions, `AddVar()` that adds a new variable to the list of variables that are a function of t and `difft()` that uses that information to compute full differentiation with respect to t, i.e., $\frac{\mathrm{d}}{\mathrm{dt}}$.

```
#
#  Routines for allowing full differentiation with t, d/dt
#

AddVar := proc (x)
    local s:
        if not(type(sub_before, set))
            then sub_before := {}:
        fi:
        if not(type(sub_after,  set))
            then sub_after  := {}:
        fi:
        s := {x(t)            = x,
              x._d(t)         = x._d,
              x._dd(t)        = x._dd,
              diff(x(t),t)    = x._d,
              diff(x(t),t,t)  = x._dd,
              diff(x._d(t),t) = x._dd }:
        sub_after := `union`(s, sub_after):
        s := {x     = x(t),
              x._d  = x._d(t),
              x._dd = x._dd(t) }:
        sub_before := `union`(s, sub_before):
        s
    end:

difft := proc (eqn) :
```

```
        subs(sub_after, diff(subs(sub_before, eqn), t))
    end:
```

The `AddVar()` procedure was then used to define that the nine generalized coordinates were differentiable by t.

```
#
#  define the generalized coord.s that will be used in the Lagrangian
#

AddVar(x):         # x position of one foot's toe
AddVar(y):         # y position of one foot's toe
AddVar(theta):     # angle from vertical (clockwise) of bottom of a leg
AddVar(a1_2a):     # hip angle for leg 'a'  (0 = straight, same as body)
AddVar(a1_2b):     # hip angle for leg 'b'  (0 = straight, same as body)
AddVar(a2a_3a):    # knee angle for leg 'a' (0 = straight leg)
AddVar(a2b_3b):    # knee angle for leg 'b' (0 = straight leg)
AddVar(a3a_4a):    # foot angle for leg 'a' (0 = perpendicular to leg)
AddVar(a3b_4b):    # foot angle for leg 'b' (0 = perpendicular to leg)
```

Switch from joint angles (θ_i in the text) to the segment angles (ϕ_i in the text) and then use these new coordinates to compute the x and y positions of both ends and the center of each segment.

```
#
#  define more intuitive var.s in terms of the generalized coord.s
#

a_4a := theta:
a_3a := a3a_4a + a_4a + Pi/2:
a_2a := a2a_3a + a_3a:
```

```
a_1  := a1_2a  + a_2a:
a_2b := a_1    - a1_2b:
a_3b := a_2b   - a2b_3b:
a_4b := a_3b   - a3b_4b - Pi/2:

# top, bottom, and middle (CM) positions of each segment

# foot 1
s4a_bot_x := x:                              # toe = bot, heel = top
s4a_bot_y := y:                              # base is toe of foot 'a'
s4a_top_x := s4a_bot_x + L4 * sin(a_4a):
s4a_top_y := s4a_bot_y + L4 * cos(a_4a):
s4a_mid_x := (s4a_top_x + s4a_bot_x) / 2:
s4a_mid_y := (s4a_top_y + s4a_bot_y) / 2:

# for (foot_frac = 1) the 'ankle' is at the heal
s4a_ankle_x := s4a_bot_x + foot_frac * L4 * sin(a_4a):
s4a_ankle_y := s4a_bot_y + foot_frac * L4 * cos(a_4a):

# calf 1
s3a_bot_x := s4a_ankle_x:
s3a_bot_y := s4a_ankle_y:
s3a_top_x := s3a_bot_x + L3 * sin(a_3a):
s3a_top_y := s3a_bot_y + L3 * cos(a_3a):
s3a_mid_x := (s3a_top_x + s3a_bot_x) / 2:
s3a_mid_y := (s3a_top_y + s3a_bot_y) / 2:

# thigh 1
s2a_bot_x := s3a_top_x:
```

```
s2a_bot_y := s3a_top_y:
s2a_top_x := s2a_bot_x + L2 * sin(a_2a):
s2a_top_y := s2a_bot_y + L2 * cos(a_2a):
s2a_mid_x := (s2a_top_x + s2a_bot_x) / 2:
s2a_mid_y := (s2a_top_y + s2a_bot_y) / 2:

# body
s1_bot_x := s2a_top_x:
s1_bot_y := s2a_top_y:
s1_top_x := s1_bot_x + L1 * sin(a_1):
s1_top_y := s1_bot_y + L1 * cos(a_1):
s1_mid_x := (s1_top_x + s1_bot_x) / 2:
s1_mid_y := (s1_top_y + s1_bot_y) / 2:

# thigh 2
s2b_top_x := s2a_top_x:
s2b_top_y := s2a_top_y:
s2b_bot_x := s2b_top_x - L2 * sin(a_2b):
s2b_bot_y := s2b_top_y - L2 * cos(a_2b):
s2b_mid_x := (s2b_top_x + s2b_bot_x) / 2:
s2b_mid_y := (s2b_top_y + s2b_bot_y) / 2:

# calf 2
s3b_top_x := s2b_bot_x:
s3b_top_y := s2b_bot_y:
s3b_bot_x := s3b_top_x - L3 * sin(a_3b):
s3b_bot_y := s3b_top_y - L3 * cos(a_3b):
s3b_mid_x := (s3b_top_x + s3b_bot_x) / 2:
s3b_mid_y := (s3b_top_y + s3b_bot_y) / 2:
```

```
# foot 2
s4b_ankle_x := s3b_bot_x:
s4b_ankle_y := s3b_bot_y:
s4b_top_x := s4b_ankle_x + (1-foot_frac) * L4 * sin(a_4b):
s4b_top_y := s4b_ankle_y + (1-foot_frac) * L4 * cos(a_4b):
s4b_bot_x := s4b_top_x - L4 * sin(a_4b):
s4b_bot_y := s4b_top_y - L4 * cos(a_4b):
s4b_mid_x := (s4b_top_x + s4b_bot_x) / 2:
s4b_mid_y := (s4b_top_y + s4b_bot_y) / 2:
```

The positions of the segments are then used to compute the position of the center-of-mass.

```
M_total := M1 + 2 * (M2 + M3 + M4):
xCM := simplify( (M1 * s1_mid_x +
                  M2 * (s2a_mid_x + s2b_mid_x) +
                  M3 * (s3a_mid_x + s3b_mid_x) +
                  M4 * (s4a_mid_x + s4b_mid_x)) / M_total ):
yCM := simplify( (M1 * s1_mid_y +
                  M2 * (s2a_mid_y + s2b_mid_y) +
                  M3 * (s3a_mid_y + s3b_mid_y) +
                  M4 * (s4a_mid_y + s4b_mid_y)) / M_total ):
```

All these positions are then differentiated to give the corresponding velocities.

```
# corresponding velocities


# body
s1_top_x_d := difft( s1_top_x ):
s1_top_y_d := difft( s1_top_y ):
```

```
s1_mid_x_d := difft( s1_mid_x ):
s1_mid_y_d := difft( s1_mid_y ):
s1_bot_x_d := difft( s1_bot_x ):
s1_bot_y_d := difft( s1_bot_y ):

# thigh 1
s2a_top_x_d := difft( s2a_top_x ):
s2a_top_y_d := difft( s2a_top_y ):
s2a_mid_x_d := difft( s2a_mid_x ):
s2a_mid_y_d := difft( s2a_mid_y ):
s2a_bot_x_d := difft( s2a_bot_x ):
s2a_bot_y_d := difft( s2a_bot_y ):

# thigh 2
s2b_top_x_d := difft( s2b_top_x ):
s2b_top_y_d := difft( s2b_top_y ):
s2b_mid_x_d := difft( s2b_mid_x ):
s2b_mid_y_d := difft( s2b_mid_y ):
s2b_bot_x_d := difft( s2b_bot_x ):
s2b_bot_y_d := difft( s2b_bot_y ):

# calf 1
s3a_top_x_d := difft( s3a_top_x ):
s3a_top_y_d := difft( s3a_top_y ):
s3a_mid_x_d := difft( s3a_mid_x ):
s3a_mid_y_d := difft( s3a_mid_y ):
s3a_bot_x_d := difft( s3a_bot_x ):
s3a_bot_y_d := difft( s3a_bot_y ):
```

```
# calf 2
s3b_top_x_d := difft( s3b_top_x ):
s3b_top_y_d := difft( s3b_top_y ):
s3b_mid_x_d := difft( s3b_mid_x ):
s3b_mid_y_d := difft( s3b_mid_y ):
s3b_bot_x_d := difft( s3b_bot_x ):
s3b_bot_y_d := difft( s3b_bot_y ):

# foot 1
s4a_top_x_d := difft( s4a_top_x ):
s4a_top_y_d := difft( s4a_top_y ):
s4a_mid_x_d := difft( s4a_mid_x ):
s4a_mid_y_d := difft( s4a_mid_y ):
s4a_bot_x_d := difft( s4a_bot_x ):
s4a_bot_y_d := difft( s4a_bot_y ):

# foot 2
s4b_top_x_d := difft( s4b_top_x ):
s4b_top_y_d := difft( s4b_top_y ):
s4b_mid_x_d := difft( s4b_mid_x ):
s4b_mid_y_d := difft( s4b_mid_y ):
s4b_bot_x_d := difft( s4b_bot_x ):
s4b_bot_y_d := difft( s4b_bot_y ):
```

Use the formula in Equation 2.3 to compute the kinetic energy of individual segments.

```
#
#  Routine for computing the kinetic energy of an idealized thin bar
#  with uniformly distributed mass
```

```
#

KE := proc (mass, name)
        local vx1, vx2, vy1, vy2:
        vx1 := name._top_x_d:
        vy1 := name._top_y_d:
        vx2 := name._bot_x_d:
        vy2 := name._bot_y_d:
        mass/6 * (vx1*vx1 + vy1*vy1 + vx1*vx2 +
                    vy1*vy2 + vx2*vx2 + vy2*vy2)
    end:
```

Then use this to compute the potential energy V and the kinetic energy T of the walker. These are the equations that are printed in Section A.1 above.

```
# kinetic energy
T := KE( M1, s1  ) +
     KE( M2, s2a ) +
     KE( M2, s2b ) +
     KE( M3, s3a ) +
     KE( M3, s3b ) +
     KE( M4, s4a ) +
     KE( M4, s4b ):


# potential energy
V := M_total * g * yCM:
```

Compute the equations for all of the sixteen possible constraints.

```
c1 := s4a_top_y:  # height of first foot's heel = 0
```

```
c2 := s4a_bot_y:  # height of first foot's toe = 0
c3 := s4b_top_y:  # height of second foot's heel = 0
c4 := s4b_bot_y:  # height of second foot's toe = 0

c5 := AHeelsX - s4a_top_x:  # first foot's heel's X is fixed
c6 := AToesX  - s4a_bot_x:  # first foot's toe's X is fixed
c7 := BHeelsX - s4b_top_x:  # second foot's heel's X is fixed
c8 := BToesX  - s4b_bot_x:  # second foot's toe's X is fixed

c11 := a1_2a - a1_2a_value:    # hip a is locked
c12 := a1_2b - a1_2b_value:    # hip b is locked
c13 := a2a_3a - a2a_3a_value:  # knee a is locked
c14 := a2b_3b - a2b_3b_value:  # knee b is locked
c15 := a3a_4a - a3a_4a_value:  # ankle a is locked
c16 := a3b_4b - a3b_4b_value:  # ankle b is locked
```

Compute all of the possible equations of motion, that is the nine equations that are always used (e1 - e9) plus the sixteen that correspond to the sixteen possible constraints (e11 - e26).

```
# equations of motion from Lagrangian Mechanics
e1  :=  RHS( x )      = Fx:
e2  :=  RHS( y )      = Fy:
e3  :=  RHS( theta )  = Ftheta:
e4  :=  RHS( a1_2a )  = F1_2a:
e5  :=  RHS( a1_2b )  = F1_2b:
e6  :=  RHS( a2a_3a ) = F2a_3a:
e7  :=  RHS( a2b_3b ) = F2b_3b:
e8  :=  RHS( a3a_4a ) = F3a_4a:
e9  :=  RHS( a3b_4b ) = F3b_4b:
```

```
e11 :=  difft(difft(c1)) + 2*alpha*difft(c1) + beta*beta*c1 = 0:
e12 :=  difft(difft(c2)) + 2*alpha*difft(c2) + beta*beta*c2 = 0:
e13 :=  difft(difft(c3)) + 2*alpha*difft(c3) + beta*beta*c3 = 0:
e14 :=  difft(difft(c4)) + 2*alpha*difft(c4) + beta*beta*c4 = 0:

e15 :=  difft(difft(c5)) + 2*alpha*difft(c5) + beta*beta*c5 = 0:
e16 :=  difft(difft(c6)) + 2*alpha*difft(c6) + beta*beta*c6 = 0:
e17 :=  difft(difft(c7)) + 2*alpha*difft(c7) + beta*beta*c7 = 0:
e18 :=  difft(difft(c8)) + 2*alpha*difft(c8) + beta*beta*c8 = 0:

e21 :=  difft(difft(c11)) + 2*alpha*difft(c11) + beta*beta*c11 = 0:
e22 :=  difft(difft(c12)) + 2*alpha*difft(c12) + beta*beta*c12 = 0:
e23 :=  difft(difft(c13)) + 2*alpha*difft(c13) + beta*beta*c13 = 0:
e24 :=  difft(difft(c14)) + 2*alpha*difft(c14) + beta*beta*c14 = 0:
e25 :=  difft(difft(c15)) + 2*alpha*difft(c15) + beta*beta*c15 = 0:
e26 :=  difft(difft(c16)) + 2*alpha*difft(c16) + beta*beta*c16 = 0:
```

The constraint equations above are stabilized, as was shown in Equation 2.9. As was discussed in Chapter 2, following Baumgarte [Bau72], the parameters are set so $\beta = \alpha$.

```
beta := alpha:
```

The forces experienced by each of the nine generalized coordinates that result from contact with the floor are computed.

```
Fx :=
      Fx_toe_a  * diff(s4a_bot_x, x) +
      Fx_heel_a * diff(s4a_top_x, x) +
      Fx_toe_b  * diff(s4b_bot_x, x) +
```

```
      Fx_heel_b * diff(s4b_top_x, x);
Fy :=
      Fx_toe_a  * diff(s4a_bot_x, y) +
      Fx_heel_a * diff(s4a_top_x, y) +
      Fx_toe_b  * diff(s4b_bot_x, y) +
      Fx_heel_b * diff(s4b_top_x, y);
Ftheta :=
      Fx_toe_a  * diff(s4a_bot_x, theta) +
      Fx_heel_a * diff(s4a_top_x, theta) +
      Fx_toe_b  * diff(s4b_bot_x, theta) +
      Fx_heel_b * diff(s4b_top_x, theta);
F1_2a := F_a1_2a +
      Fx_toe_a  * diff(s4a_bot_x, a1_2a) +
      Fx_heel_a * diff(s4a_top_x, a1_2a) +
      Fx_toe_b  * diff(s4b_bot_x, a1_2a) +
      Fx_heel_b * diff(s4b_top_x, a1_2a);
F1_2b := F_a1_2b +
      Fx_toe_a  * diff(s4a_bot_x, a1_2b) +
      Fx_heel_a * diff(s4a_top_x, a1_2b) +
      Fx_toe_b  * diff(s4b_bot_x, a1_2b) +
      Fx_heel_b * diff(s4b_top_x, a1_2b);
F2a_3a := F_a2a_3a +
      Fx_toe_a  * diff(s4a_bot_x, a2a_3a) +
      Fx_heel_a * diff(s4a_top_x, a2a_3a) +
      Fx_toe_b  * diff(s4b_bot_x, a2a_3a) +
      Fx_heel_b * diff(s4b_top_x, a2a_3a);
F2b_3b := F_a2b_3b +
      Fx_toe_a  * diff(s4a_bot_x, a2b_3b) +
      Fx_heel_a * diff(s4a_top_x, a2b_3b) +
```

```
      Fx_toe_b  * diff(s4b_bot_x, a2b_3b) +
      Fx_heel_b * diff(s4b_top_x, a2b_3b);
F3a_4a := F_a3a_4a +
      Fx_toe_a  * diff(s4a_bot_x, a3a_4a) +
      Fx_heel_a * diff(s4a_top_x, a3a_4a) +
      Fx_toe_b  * diff(s4b_bot_x, a3a_4a) +
      Fx_heel_b * diff(s4b_top_x, a3a_4a);
F3b_4b := F_a3b_4b +
      Fx_toe_a  * diff(s4a_bot_x, a3b_4b) +
      Fx_heel_a * diff(s4a_top_x, a3b_4b) +
      Fx_toe_b  * diff(s4b_bot_x, a3b_4b) +
      Fx_heel_b * diff(s4b_top_x, a3b_4b);

# Create matrices
with(linalg):

LMatrix := proc (n)
        genmatrix(map(proc(name,num) Const(num,name) = 0: end,
                        [x, y, theta,
                         a1_2a,  a1_2b,
                         a2a_3a, a2b_3b,
                         a3a_4a, a3b_4b,
                         lambda1,  lambda2,  lambda3,  lambda4,
                         lambda5,  lambda6,  lambda7,  lambda8,
                         lambda11, lambda12, lambda13,
                         lambda14, lambda15, lambda16],
                        n),
          [x_dd, y_dd, theta_dd,
           a1_2a_dd,  a1_2b_dd,
```

```
        a2a_3a_dd, a2b_3b_dd,
        a3a_4a_dd, a3b_4b_dd,
        lambda1,  lambda2,  lambda3,  lambda4,
        lambda5,  lambda6,  lambda7,  lambda8,
        lambda11, lambda12, lambda13,
        lambda14, lambda15, lambda16],
       flag):
     end:

LM1  := LMatrix(  1 ):
LM2  := LMatrix(  2 ):
LM3  := LMatrix(  3 ):
LM4  := LMatrix(  4 ):
LM5  := LMatrix(  5 ):
LM6  := LMatrix(  6 ):
LM7  := LMatrix(  7 ):
LM8  := LMatrix(  8 ):
LM11 := LMatrix( 11 ):
LM12 := LMatrix( 12 ):
LM13 := LMatrix( 13 ):
LM14 := LMatrix( 14 ):
LM15 := LMatrix( 15 ):
LM16 := LMatrix( 16 ):

M_raw := genmatrix([e1, e2, e3, e4, e5, e6, e7, e8, e9,
                    e11, e12, e13, e14, e15, e16, e17, e18,
                    e21, e22, e23, e24, e25, e26],
          [x_dd, y_dd, theta_dd,
           a1_2a_dd,  a1_2b_dd,
```

```
           a2a_3a_dd, a2b_3b_dd,
           a3a_4a_dd, a3b_4b_dd,
           lambda1,  lambda2,  lambda3,  lambda4,
           lambda5,  lambda6,  lambda7,  lambda8,
           lambda11, lambda12, lambda13,
           lambda14, lambda15, lambda16],
          flag):

print(`Completed computing M`):

M_raw := map(simplify, M_raw):

M_width  := coldim(M_raw):
M_height := rowdim(M_raw):

M_all := stack(M_raw,
        LM1,LM2,LM3,LM4,LM5,LM6,LM7,LM8,LM11,LM12,LM13,LM14,LM15,LM16):
all_width  := coldim(M_all):
all_height := rowdim(M_all):

M := extend(eval(M_all), 1, 0, 0):
for i from 1 to all_height do M[i,all_width] := -M[i,all_width] od:

# stick in some "useful" values
M[all_height+1, 1]  :=  simplify(difft(difft(xCM))):  #  xCM_dd
M[all_height+1, 2]  :=  simplify(difft(difft(yCM))):  #  yCM_dd
M[all_height+1, 3]  :=  simplify(difft(s1_top_x_d)):  #  s1_top_x_dd
M[all_height+1, 4]  :=  simplify(difft(s1_top_y_d)):  #  s1_top_y_dd
```

```
M[all_height+1, 5]  :=  simplify(difft(s4a_top_x_d)): #  Heel A's x_dd
M[all_height+1, 6]  :=  simplify(difft(s4a_top_y_d)): #  Heel A's y_dd
M[all_height+1, 7]  :=  simplify(difft(s4a_bot_x_d)): #  Toe A's x_dd
M[all_height+1, 8]  :=  simplify(difft(s4a_bot_y_d)): #  Toe A's y_dd

M[all_height+1, 9]  :=  simplify(difft(s4b_top_x_d)): #  Heel B's x_dd
M[all_height+1,10]  :=  simplify(difft(s4b_top_y_d)): #  Heel B's y_dd
M[all_height+1,11]  :=  simplify(difft(s4b_bot_x_d)): #  Toe B's x_dd
M[all_height+1,12]  :=  simplify(difft(s4b_bot_y_d)): #  Toe B's y_dd

# output M
C(M, optimized, filename = `Maple.out`):
```

APPENDIX B

Model Parameters

This appendix presents a series of tables containing the values of the parameters that specify the details of the walker. These parameters are described in Chapter 2 and used in Appendix A.

Symbol	*Value*	*Description*
m_1	40 kilograms	Mass of the body
m_2	8 kilograms	Mass of each thigh
m_3	7 kilograms	Mass of each calf
m_4	2 kilograms	Mass of each foot

Table B.1: *Mass of each of the segments*

Symbol	*Value*	*Description*
l_1	0.8 meter	Length of the body
l_2	0.5 meter	Length of each thigh
l_3	0.5 meter	Length of each calf
l_4	0.25 meter	Length of each foot

Table B.2: *Length of each of the segments*

Symbol	*Value*	*Description*
$\theta_{\mathrm{hip,max}}$	100°	Largest possible angle for a hip
$\theta_{\mathrm{hip,min}}$	-30°	Smallest possible angle for a hip
$\theta_{\mathrm{knee,max}}$	0°	Largest possible angle for a knee
$\theta_{\mathrm{knee,min}}$	-135°	Smallest possible angle for a knee
$\theta_{\mathrm{ankle,max}}$	45°	Largest possible angle for a ankle
$\theta_{\mathrm{ankle,min}}$	-45°	Smallest possible angle for a ankle

Table B.3: *Maximum and minimum angles for joints*

Symbol	*Value*	*Description*
U_{hip}	300 Newton meters	Bound on hip torque
U_{knee}	200 Newton meters	Bound on knee torque
U_{ankle}	100 Newton meters	Bound on ankle torque

Table B.4: *Maximum torques that each joint can apply*

Symbol	*Value*	*Description*
μ_{b1}	0.02 Newtons / second	Bearing friction proportional to velocity
μ_{b2}	0.05 Newton meters	Bearing friction, constant torque
μ_{s}	0.92	Static friction of feet
μ_{d}	0.48	Dynamic friction of feet

Table B.5: *Friction Coefficients*

References

[Act90] F.S. Acton. *Numerical Methods That Work.* Mathematical Association of America, Washington, 1990.

[AS86] H. Asada and J.-J.E. Slotine. *Robot Analysis and Control.* Wiley, New York, 1986.

[Bau72] J. Baumgarte. "Stabilization of constraints and integrals of motion in dynamical systems." *Computer Methods in Applied Mechanics and Engineering*, **1**(1):1–16, 1972.

[BD83] G. Bader and P. Deuflhard. "A semi-implicit mid-point rule for stiff systems of ordinary differential equations." *Numerische Mathematik*, **41**:373–398, 1983.

[Bre73] R.P. Brent. *Algorithms for Minimization without Derivatives.* Prentice-Hall, Englewood Cliffs, N.J., 1973.

[Bro85] W.L. Brogan. *Modern Control Theory.* Prentice-Hall, Englewood Cliffs, N.J., 1985.

[Bro89] R.A. Brooks. "A robot that walks; emergent behaviors from a carefully evolved network." *Neural Computation*, **1**(2):253–262, 1989.

[Bro91] R.A. Brooks. "New approaches to robotics." *Science*, **253**(5025):1227–1232, 1991.

[Can67] R.H. Cannon, Jr. *Dynamics of Physical Systems.* McGraw-Hill, New York, 1967.

[CGG91] B.W. Char, K.O. Geddes, G.H. Gonnet, B.L. Leong, M.B. Monagan, and S.M. Watt. *MapleV Library Reference Manual.* Springer-Verlag, Berlin, 1991.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[CV94] M.G. Cooper and J.J. Vidal. "Genetic Design of Fuzzy Controllers: The Cart and Jointed Pole Problem." In *Proceedings 1994 IEEE International Conference on Fuzzy Systems*, Orlando, Florida, 1994. IEEE, New York.

[DB74] G. Dahlquist and Å. Björck. *Numerical Methods*. Prentice-Hall, Englewood Cliffs, N.J., 1974.

[deG89] H. de Garis. "WALKER, A Genetically Programmed, Time Dependent, Neural Net Which Teaches a Pair of Sticks to Walk." Technical report, George Mason University, Fairfax, VA, 1989.

[deG90] H. de Garis. "Genetic Programming: Modular Evolution for Darwin Machines." In *International Joint Conference on Neural Networks*, Washington D.C., 1990. IEEE, New York.

[Don87] M.D. Donner. *Real-Time Control of Walking*. Birkhäuser, Boston, MA, 1987.

[Fea87] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer, Boston, 1987.

[Feh70] E. Fehlberg. "Klassische Runge-Kutta-formeln vierter und niedrigerer ordnung mit schrittweiten-kontrolle und ihre anwendung auf wärmeleitungsprobleme." *Computing*, **6**:61–71, 1970.

[FS90] J. Furucho and A. Sano. "Sensor-based control of a nine-link biped." *International Journal of Robotics Research*, **9**(2):83–98, 1990.

[Gea71] C.W. Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Englewood Cliffs, N.J., 1971.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.

[Gri90] R. Grimshaw. *Nonlinear Ordinary Differential Equations*. Blackwell Scientific Publications, Oxford, 1990.

[GV89] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, 1989.

[Hig63] D.T. Higdon. *Automatic Control of Inherently Unstable Systems with Bounded Control Inputs*. PhD thesis, Stanford University, Department of Aeronautics and Astronautics, 1963.

[Hur92] Y. Hurmuzlu. "Outcomes of constrained contact in the locomotion of a five-link biped." In A. Ertas and B. Ovunc, editors, *Dynamics and Vibrations 1992*, volume 44, pp. 71–79. American Society of Mechanical Engineers, New York, 1992.

[Kha92] H.K. Khalil. *Nonlinear Systems*. Prentice-Hall, Englewood Cliffs, N.J., 1992.

[KJ83] S. Kirkpatrick, C.D. Gelatt Jr., , and M.P. Vecchi. "Optimization by Simulated Annealing." *Science*, **220**, 1983.

[KL83] T.R. Kane and D.A. Levinson. "The use of Kane's dynamical equations in robotics." *The International Journal of Robotics*, **2**(3):3–21, 1983.

[KL85] T.R. Kane and D.A. Levinson. *Dynamics: Theory and Applications*. McGraw-Hill, New York, 1985.

[KTT88] M. Kaneko, T. Tanie, and M.N.M. Than. "A control algorithm for hexapod walking machine over soft ground." *IEEE Journal of Robotics and Automation*, **4**(3):294–302, 1988.

[Lee87] S.S. Lee. *Symbolic generation of equations of motion for dynamics/control simulation of large flexible multibidy space systems*. PhD thesis, University of California, Los Angeles, 1987.

[Lei87] J.R. Leigh. *Applied Control Theory*. Peter Peregrinus, Ltd., London, 1987.

[LMS69] A.M. Lopez, P.W. Murill, and C.L. Smith. "Tuning PI and PID digital controllers." *Instruments and Control Systems*, **42**(2):89–95, 1969.

[LW93] R.R. Leighton and A.P. Wieland. "The Aspirin/MIGRAINES Software Package." In J. Skrzypek, editor, *Neural Network Simulation Environments*, chapter 11. Kluwer, Boston, 1993.

[LWP80] J.Y.S. Luh, M.H. Walker, and R.P.C. Paul. "On-line computational scheme for mechanical manipulators." *ASME Journal of Dynamical Systems, Measurement and Control*, **120**(2):69–76, 1980.

[Man92] D.J. Manko. *A General Model of Legged Locomotion on Natural Terrain*. Kluwer, Boston, 1992.

[McG90] T. McGeer. "Passive Dynamic Walking." *International Journal of Robotics Research*, **9**(2):62–82, 1990.

[MS84] H. Miura and I. Shimoyama. "Dynamic walk of a biped." *International Journal of Robotics Research*, **3**(2):60–74, 1984.

[OO81] D.E. Orin and S.–Y. Oh. "Control of force distribution in robotic mechanisms containing closed kinematic chains." *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, **103**(2), 1981.

[OO86] S.–Y. Oh and D.E. Orin. "Dynamic computer simulation of multiple closed-chain robotic mechanisms." In *Proceedings 1986 IEEE International Conference on Robotics and Automation*, pp. 15–20, San Francisco, 1986. IEEE, New York.

[PTV92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C.* Cambridge University Press, Cambridge, 1992.

[Rai86] M. Raibert. *Legged Robots that Balance.* MIT Press, Cambridge, MA, 1986.

[Rai90] M. Raibert. "Forward to the special issue on legged locomotion." *International Journal of Robotics Research*, **9**(2), 1990.

[RHW86] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. "Learning Internal Representations by Error Propagation." In D.E. Rumelhart and J.L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8, pp. 318–362. MIT Press, Cambridge, 1986.

[RJC84] M. Raibert, H.B. Brown Jr., and M. Chepponis. "Experiments in balance with a 3D one-legged hopping machine." *International Journal of Robotics Research*, **3**(2):75–92, 1984.

[RMS69] A.A. Rovina, P.W. Murill, and C.L. Smith. "Tuning controllers for setpoint changes." *Instruments and Control Systems*, **42**(12):67, 1969.

[SB80] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, Berlin, 1980.

[Sch65] J.F. Schaefer. *On the Bounded Control of Some Unstable Mechanical Systems.* PhD thesis, Stanford University, Department of Electrical Engineering, 1965.

[Shi86] L. Shih. *Dynamic modelling and simulation of mechanisms consisting of closed and open kinematic chains with compliance.* PhD thesis, University of Wisconsin-Madison, 1986.

[SMM91] J. Shaw, M.C. Mulder, and C. Meeker. "Mathematical model of a multi-segment, multi-joint biped." In *IEEE International Conference on Robotics and Automation*, pp. 2098 – 2105. IEEE, New York, 1991.

[SS89] J. Shaw and S.W. Shaw. "The onset of chaos in a two-degree-of-freedom impacting system." *Transactions of the ASME*, **56**:168–174, 1989.

[Sta93] R.M. Stallman. *Using and Porting GNU CC*. the Free Software Foundation, Cambridge, MA, 1993.

[Str80] G. Strang. *Linear Algebra and its Applications*. Academic Press, New York, 1980.

[Str91] B. Stroustrup. *The C++ Programming Language*. Addison-Wesley, Reading, 1991.

[SWD76] L.F. Shampine, H.A. Watts, and S.M. Davenport. "Solving nonstiff ordinary differential equations – the state of the art." *SIAM Review*, **18**:376–411, 1976.

[VBS90] M. Vukobratović, B. Borovac, D. Surla, and D. Stockić, editors. *Biped Locomotion: Dynamics, Stability, Control and Application*. Springer-Verlag, Berlin, 1990.

[Wer74] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[Wie91a] A.P. Wieland. "Evolving Controls for Unstable Systems." In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, editors, *Proceedings of the 1990 Connectionist Models Summer School*, pp. 91–102, San Diego, 1991. Morgan Kaufmann, San Mateo, CA.

[Wie91b] A.P. Wieland. "Evolving Neural Network Controllers for Unstable Systems." In *International Joint Conference on Neural Networks*, volume 2, pp. 667–673, Seattle, 1991. IEEE, New York.

[Wit77] J. Wittenburg. *Dynamics of Systems of Rigid Bodies*. B.G. Teubner, Stuttgart, 1977.

[Wol91] S. Wolfram. *Mathematica: A System for Doing Mathematics by Computer*. Addison-Wesley, Reading, 1991.

[ZH84] Y.F. Zheng and H. Hemami. "Impact effects of biped contact with the environment." *IEEE Transactions on Systems, Man, and Cybernetics*, **14**(3):437–443, 1984.

[Zhe87] Y.F. Zheng. "Acceleration compensation for biped robots to reject external disturbances." *IEEE Transactions on Systems, Man, and Cybernetics*, **19**(1):74–84, 1987.

[ZN42] J.G. Ziegler and N.Z. Nichols. "Optimum settings for automatic controllers." *Transactions of the ASME*, **64**(11):759, 1942.