A simulation study of bipedal walking robots: Modeling, walking algorithms, and neural network control

Paul Walker Latham

University of New Hampshire, Durham

Follow this and additional works at: https://scholars.unh.edu/dissertation

Recommended Citation
https://scholars.unh.edu/dissertation/1698

This Dissertation is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Doctoral Dissertations by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact nicole.hentz@unh.edu.
A simulation study of bipedal walking robots: Modeling, walking algorithms, and neural network control

Abstract
The purpose of this thesis is to develop walking algorithms for use with mechanical bipeds. This thesis is comprised of three parts. A two dimensional biped simulator, called WALK, is developed. This simulator is designed to facilitate the evaluation of bipedal walking algorithms. Then, a two dimensional walking algorithm is developed using a simple inverted pendulum model. This algorithm is shown to provide for stable walking using the WALK simulator system. A neurocomputer adaptive controller that is based on the CMAC architecture is added to the inverted pendulum model. The adaptive walking algorithm is not only stable, but provides for accurate control when leg inertia and modeling errors are considered.

Keywords
Engineering, Electronics and Electrical, Artificial Intelligence

This dissertation is available at University of New Hampshire Scholars' Repository: https://scholars.unh.edu/dissertation/1698
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI
University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700   800/521-0600
A simulation study of bipedal walking robots: Modeling, walking algorithms, and neural network control

Latham, Paul Walker, II, Ph.D.

University of New Hampshire, 1992
A SIMULATION STUDY OF BIPEDAL WALKING ROBOTS:
MODELING, WALKING ALGORITHMS, AND NEURAL NETWORK CONTROL

by

Paul W. Latham II
B.S.E.E. University of Minnesota, June 1980

DISSERTATION

Submitted to the University of New Hampshire
in Partial Fulfillment of
the Requirements for the Degree of

Doctor of Philosophy
in
Engineering

September, 1992
This dissertation has been examined and approved.

W. R. MILLER
Dissertation Director, W. Thomas Miller III
Professor of Electrical and Computer Engineering

Filson H. Glanz
Professor of Electrical and Computer Engineering

L. Gordon Kraft
Professor of Electrical and Computer Engineering

David E. Limbert
Professor of Mechanical Engineering

Lee L. Zia
Associate Professor of Mathematics

Date 5/14/92
DEDICATION

To Colleen

iii
ACKNOWLEDGEMENTS

I would like to thank Dr. T. Miller, Dr. G. Kraft, Dr. D. Limbert, and Dr. L. Zia for their suggestions and support. I would to thank Diane Knaefler, Jenny Pier, and Kristann Moody for help in proofreading. I would also like to thank my wife, Colleen, helping make all of my graduate studies possible.
# TABLE OF CONTENTS

DEDICATION ........................................ iii
ACKNOWLEDGEMENTS ................................. iv
LIST OF FIGURES .................................... viii
LIST OF TABLES ...................................... x
LIST OF SYMBOLS .................................... xi
ABSTRACT ........................................... xiv

Chapter 1
ISSUES IN BIPEDAL WALKING ...................... 1
   Introduction ..................................... 1
   Why Build an Artificial Biped ................ 1
   The Bipedal Walking Problem .................. 2
   Overview ....................................... 5

Chapter 2
REVIEW OF BIPEDAL WALKING RESEARCH .......... 8
   Walking Algorithms ............................. 8
   Walking Methods ................................ 8
   Deterministic Bipedal Walking Algorithms .... 10
   Adaptive Bipedal Walking Algorithms ........ 11
   Conclusion ..................................... 12

Chapter 3
TWO DIMENSIONAL DYNAMIC SIMULATOR ........ 13
   Introduction ................................... 13
   Simulator Design Philosophy .................. 14
Biped Simulation Model .................................................. 15
Two Dimensional Continuous Time Model .......................... 20
Two Dimensional Discretized System ................................. 31
Computer Algorithm ..................................................... 42
Simulation Results ....................................................... 43
Conclusion ................................................................. 48

Chapter 4

DETERMINISTIC WALKING ALGORITHM ............................... 49
Introduction ............................................................. 49
Bipedal Robot Structure ................................................. 49
Background on Walking .................................................. 51
Walking Algorithm ....................................................... 59
Model Mismatching Analysis ............................................. 70
Simulation Results ....................................................... 73
Conclusion ................................................................. 84

Chapter 5

ADAPTIVE WALKING ALGORITHM ..................................... 85
Introduction ............................................................. 85
CMAC Walking Algorithm ............................................... 85
Simulation Results ....................................................... 89
Conclusion ................................................................. 97

Chapter 6

SUMMARY ................................................................. 98
Two Dimensional Simulator .............................................. 98
Deterministic Walking Algorithm ..................................... 99
Adaptive Walking Algorithm ........................................... 99

vi
Practical Biped Walkers .................... 100
Future Directions ......................... 101
REFERENCES .............................. 104
Appendix A
THREE DIMENSIONAL DYNAMIC SIMULATOR ................ 109
   Introduction ......................... 109
   Three Dimensional Simulator ........... 109
Appendix B
WALK USERS MANUAL ....................... 122
   General Information .................. 122
   WALK Simulator Files ................. 123
   Command Reference ................... 124
   Default Settings .................... 132
Appendix C
BIOLOGICAL REFERENCE ................... 134
   Introduction ......................... 134
   Biological Sensors .................. 134
   Bipedal walking .................... 147
   Control in Biological Walking Systems .... 151
   Conclusion .......................... 154
   Reprinted Figures ................... 157
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Biped Stick Diagram</td>
<td>17</td>
</tr>
<tr>
<td>3.2</td>
<td>Room Coordinate System</td>
<td>18</td>
</tr>
<tr>
<td>3.3</td>
<td>Inverse Model</td>
<td>25</td>
</tr>
<tr>
<td>3.4</td>
<td>Floor Impact</td>
<td>41</td>
</tr>
<tr>
<td>3.5</td>
<td>Falling Example</td>
<td>44</td>
</tr>
<tr>
<td>3.6</td>
<td>Jumping Example</td>
<td>45</td>
</tr>
<tr>
<td>3.7</td>
<td>Walking Example</td>
<td>46</td>
</tr>
<tr>
<td>3.8</td>
<td>Time Step Adjustment</td>
<td>47</td>
</tr>
<tr>
<td>4.1</td>
<td>Idealized Biped</td>
<td>50</td>
</tr>
<tr>
<td>2.2</td>
<td>Center of Force</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Inverted Pendulum</td>
<td>57</td>
</tr>
<tr>
<td>4.4</td>
<td>Walking Process</td>
<td>60</td>
</tr>
<tr>
<td>4.5</td>
<td>Deterministic Walking Algorithm</td>
<td>70</td>
</tr>
<tr>
<td>4.6</td>
<td>Mismatch Model</td>
<td>72</td>
</tr>
<tr>
<td>4.7</td>
<td>Reference Biped Trajectories</td>
<td>74</td>
</tr>
<tr>
<td>4.8</td>
<td>Reference Biped Body Velocity</td>
<td>75</td>
</tr>
<tr>
<td>4.9</td>
<td>Reference Biped Step Length</td>
<td>77</td>
</tr>
<tr>
<td>4.10</td>
<td>Body Velocity vs. Body/Leg Mass Ratio</td>
<td>78</td>
</tr>
<tr>
<td>4.11</td>
<td>Step Length vs. Body/Leg Mass Ratio</td>
<td>79</td>
</tr>
<tr>
<td>4.12</td>
<td>Body Velocity vs. Step Length Errors</td>
<td>80</td>
</tr>
<tr>
<td>4.13</td>
<td>Step Length vs. Step Length Errors</td>
<td>81</td>
</tr>
<tr>
<td>4.14</td>
<td>Body Velocity vs. Leg Length Errors</td>
<td>82</td>
</tr>
<tr>
<td>4.15</td>
<td>Body Velocity vs. Joint Angle Errors</td>
<td>83</td>
</tr>
<tr>
<td>5.1</td>
<td>CMAC/Deterministic Walking Algorithm</td>
<td>89</td>
</tr>
</tbody>
</table>
Figure 5.2 Body Velocity for CMAC Learning Transient  . 90
Figure 5.3 Step Length for CMAC Learning Transient  . 91
Figure 5.4 Body Velocity Step Response for CMAC Controller  . 92
Figure 5.5 Balancing with and without CMAC Controller  . 93
Figure 5.6 CMAC Only Walking Algorithm Trials  . 94
Figure 5.7 Body Velocity for Converged CMAC Only Algorithm  . 95
Figure 5.8 Step Length for Converged CMAC Only Algorithm  . 96
Figure C.1 Semicircular Canals  . 157
Figure C.2 Semicircular Duct  . 158
Figure C.3 Bode Plot of Semicircular Canals  . 159
Figure C.4 Step Response of Semicircular Canals  . 160
Figure C.5 Gravireceptor Organs  . 161
Figure C.6 Bode Plot of Otolithic Organs  . 162
Figure C.7 Step Response of Otolithic Organs  . 163
Figure C.8 Walking Force Diagram  . 164
Figure C.9 Mechanoreceptor Responses  . 165
Figure C.10 Detection Thresholds for Cutaneous Receptors  . 166
Figure C.11 Horizontal Body Accelerations for Walking  . 167
Figure C.12 Vertical Body Accelerations for Walking  . 168
Figure C.13 Pelvis Displacement for Walking  . 169
Figure C.14 Body Velocity for Walking vs. Speed  . 170
Figure C.15 Tension/Length Diagram for Muscles  . 171
LIST OF TABLES

Table 3.1 Discrete Equation Usage ............... 39
Table 4.1 Reference Biped ....................... 73
**LIST OF SYMBOLS**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta t$</td>
<td>Time Increment</td>
</tr>
<tr>
<td>$\delta x_0$</td>
<td>X Axis Body Position Increment</td>
</tr>
<tr>
<td>$\delta q_j$</td>
<td>Unknown Position Increments</td>
</tr>
<tr>
<td>$\delta u_j$</td>
<td>Known Joint Angle Input Increments</td>
</tr>
<tr>
<td>$\delta y_0$</td>
<td>Y Axis Body Position Increment</td>
</tr>
<tr>
<td>$\delta y_f$</td>
<td>Y Axis Foot Position Increment</td>
</tr>
<tr>
<td>$\delta x_f$</td>
<td>X Axis Foot Position Increment</td>
</tr>
<tr>
<td>$\delta \theta_0$</td>
<td>$\theta$ Body Rotation Increment</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Inverted Pendulum Frequency</td>
</tr>
<tr>
<td>$\alpha_b$</td>
<td>Physical Inverted Pendulum Frequency</td>
</tr>
<tr>
<td>$\Delta t_1, \Delta t_2$</td>
<td>Current and Passed Simulation Time Steps</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>Initial Simulation Time Step</td>
</tr>
<tr>
<td>$\theta_{con}$</td>
<td>Body Control Angle</td>
</tr>
<tr>
<td>$\theta_j$</td>
<td>$j^{th}$ Link Total Rotation Angle</td>
</tr>
<tr>
<td>$\theta_o$</td>
<td>Body Orientation Angle</td>
</tr>
<tr>
<td>$\ddot{\theta}_{ri}$</td>
<td>Relative Angular Acceleration of the $i^{th}$ Link</td>
</tr>
<tr>
<td>$\theta_{ref}$</td>
<td>Body Reference Angle</td>
</tr>
<tr>
<td>$\lambda_k$</td>
<td>Constraint Force</td>
</tr>
<tr>
<td>$\tau$</td>
<td>Body Angle Filter Time Constant</td>
</tr>
<tr>
<td>$\varphi_1, \varphi_2, \varphi_3$</td>
<td>Right Leg Joint Angles</td>
</tr>
<tr>
<td>$\varphi_4, \varphi_5, \varphi_6$</td>
<td>Left Leg Joint Angles</td>
</tr>
<tr>
<td>$c_k$</td>
<td>Active Floor Constraints</td>
</tr>
<tr>
<td>$\text{CMAC}_i$</td>
<td>$i^{th}$ CMAC Output</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CMAC_{final}  Final CMAC Output
\( e_i \)  Position and Orientation of the \( i^{th} \)
Center of Mass
\( F_p \)  External Forces
\( g \)  Gravity
\( h \)  Body Height
\( h_s \)  Step Height
\( I \)  Identity Matrix
\( I_i \)  \( i^{th} \) Link Inertia
\( K_j \)  \( j^{th} \) Link Center of Mass
\( L \)  Lagrangian (KE - PE)
\( M_i \)  Inertia Matrix for the \( i^{th} \) Link
\( m_i \)  Mass of the \( i^{th} \) Link.
\( p(j) \)  Link Index Selection Function
\( Q_\theta \)  Generalized Force in the \( \theta \) Direction
\( Q_y \)  Generalized Force in the \( y \) Direction
\( Q_x \)  Generalized Force in the \( x \) Direction
\( q_j \)  Unknown Positions
\( S_p \)  Position Vector
\( S_i \)  \( i^{th} \) Deterministic Output
\( t \)  Time
\( t_0 \)  Initial Time
\( T_s \)  Walking Step Time Interval
\( u_j \)  Known Joint Angle Inputs
\( \dot{x}_0 \)  Body Velocity
\( \dot{x}_d \)  Desired Body Velocity
\( \dot{x}_{\text{final}} \)  
Final Body Velocity

\( x_{n+1} \)  
Next Step Position

\( x_p \)  
Passive Leg X Axis Position

\( \ddot{x}_{ri} \)  
Relative X axis Acceleration of the \( i^{th} \) Link

\( x_a \)  
Active Leg X Axis Position

\( x_0, y_0 \)  
Body Position

\( \dot{x}_{i+1} \)  
Desired End-of-Step Velocity

\( \dot{x}_1 \)  
Last End-of-Step Velocity

\( x_i, y_i \)  
\( i^{th} \) Link Center of Mass

\( x_{pi} \)  
Initial Passive Leg X Axis Position

\( x_{pi+1} \)  
Final Passive Leg X Axis Position

\( x_l, y_l \)  
Left Leg Position

\( x_r, y_r \)  
Right Leg Position

\( \dot{y}_{ri} \)  
Relative Y Axis Acceleration of the \( i^{th} \) link

\( x_{rf} \)  
X Axis Body Relative Foot Position

\( y_{rf} \)  
Y Axis Body Relative Foot Position
ABSTRACT

A SIMULATION STUDY OF BIPEDAL WALKING ROBOTS:
MODELING, WALKING ALGORITHMS, AND NEURAL NETWORK CONTROL

by

Paul W. Latham II

University of New Hampshire, September, 1992

The purpose of this thesis is to develop walking algorithms for use with mechanical bipeds. This thesis is comprised of three parts. A two dimensional biped simulator, called WALK, is developed. This simulator is designed to facilitate the evaluation of bipedal walking algorithms. Then, a two dimensional walking algorithm is developed using a simple inverted pendulum model. This algorithm is shown to provide for stable walking using the WALK simulator system. A neurocomputer adaptive controller that is based on the CMAC architecture is added to the inverted pendulum model. The adaptive walking algorithm is not only stable, but provides for accurate control when leg inertia and modeling errors are considered.
Chapter 1

ISSUES IN BIPEDAL WALKING

Introduction

The main purpose of this thesis is to develop a walking algorithm for use with mechanical bipeds. To verify the walking algorithm, a bipedal walking simulator is developed. The simulator exposed several problems with a number of preliminary walking algorithms. The resulting algorithm described in this thesis is the result of many refinements discovered by the use of the simulator. The simulator and walking algorithm can be used together with a proposed biped design. A proposed biped can be checked out before construction of a physical biped.

This thesis is presented in three sections: a two dimensional biped simulator, an analytic (deterministic) bipedal walking algorithm, and an adaptive bipedal walking algorithm. Each of these sections are covered in chapters three, four, and five.

Why Build an Artificial Biped

Bipedal walking machines are of interest because they have some of the flexibility that humans in moving about in a
large variety of environments. Bipedal machines are also more compatible in an environment built for two legged systems (humans). For example, inside buildings two legged machines may find it easier to move about on stairs and ladders, and in closets.

The Bipedal Walking Problem

Bipedal walking is walking on two legs. From practical experience as a human, bipedal walking requires good balance to avoid falling down. Whereas, animals with more than two legs have an relatively easy time with stability, balancing with only two legs poses a more complex control problem. A successful walking controller requires detailed knowledge of the walker's structure and dynamics.

Leg control in a biped requires precise positioning and timing to achieve coordinated movement. To be useful, the biped must not only remain upright, but must possess several other desirable attributes. For the biped's body to provide for a "stable" platform, the body's height and orientation must be controlled. The leg trajectories chosen should be such that the legs do not collide with each other, the floor, or anything else. Good body dynamics such as high speed, rejection of external forces, and precise position or velocity control are desirable. This thesis concerns itself with the dynamics problem and does not address many of the leg trajectory issues.

2
The Bipedal Walking Control Problem

The control of a biped's dynamics is complicated by mass and inertia of the legs. Like all mechanical systems, the biped is a complex system of distributed masses with flexible structural components. With the presence of joints, even a simplified rigid body model for the biped has to include several coupled masses or links.

To accurately model the biped with high speed movements, a complex multi-body or multi-link rigid body model is needed. The effects of leg inertia will have a significant effect on the body dynamics. To compensate for these coupled dynamics, complex control algorithms need to be applied if accurate results are desired. These controllers involve high order systems of non-linear coupled differential equations (Hemami, 1984).

To apply analytical methods to the design of bipedal walking algorithms requires the knowledge of several structural parameters. These include the inertia, mass, and geometry of all the links used to model the biped. Some of the biped's parameters may change. For example, the biped's body mass will change when the biped is carrying a payload. Many of these parameters are difficult to determine. Adaptive methods are needed for any practical biped walking algorithm to overcome this problem.
The Biped Simulator

A simulator is useful in analyzing potential walking algorithms. Simulators can be used to test the strengths and weaknesses of an algorithm. For example, leg masses can be varied to test the effects of coupling. With a simulator, very high speed movements can be tested due to unlimited actuator forces. In a simulator, parametric changes can be made in seconds. On the other hand, physical bipeds have many engineering issues that are outside the scope of walking algorithm development. As such, real bipeds take a long time to build and debug. Thus, making an analysis of parametric changes in the biped structure is difficult.

The limitations of a biped simulator are determined by their underlying assumptions. All simulators have differences between their results and reality. Building a physical biped is the final proof of a good walking algorithm. A good simulator can save a lot of time in algorithm debugging and biped walker design. For this reason, a major part of this thesis is on simulator design.

Biped Model Design

The simulator that is described in Chapter 3 is designed to easily test walking algorithms. Several simplifications are made to allow for ease of use and fast simulation time. The biped model consists of a seven-link (mass) model in two dimensions. The two dimensional plane that is used is the
sagittal plane (up-down and forward-backwards). The inputs to the simulator are joint angles, not torques.

The assumption of two dimensions is made to simplify the computations required by the simulator and to ease the algorithm design. For most walking problems the body orientation is fixed or slowly moving. The low body orientation velocities limit the coupling between the frontal (facing) and sagittal (along side) planes. Thus, the three dimensional problem can be decomposed into two, two-dimensional problems.

The inputs to the simulator are joint angles. This input choice defers the design of the joint angle controllers. By assuming that accurate joint angle controllers are possible, only the step placement algorithm has to be designed. The step placement algorithm is the subject of this thesis.

**Overview**

This thesis is divided into three main sections: the design of a two dimensional simulator called WALK, the development of a simple analytic walking control algorithm, and an extension to include the addition of an adaptive controller.

Chapter 2 is a short account of the previous research on bipedal walking. The walking algorithms include both deterministic (analytic) and adaptive methods.
Chapter 3 describes the development of a two dimensional simulator called WALK. This C language program is used for the design and development of the walking algorithms described in the rest of this thesis. The WALK simulator's modeling techniques and algorithm provides for the unique combination of high speed simulations with ground impacts effects.

Chapter 4 develops a simple walking algorithm based on an inverted pendulum model. This walking algorithm controls the peak body velocity by a proportional feedback gain. This walking algorithm improves on prior work by combining simplicity with tolerance to leg masses.

Chapter 5 adds an adaptive controller to the deterministic algorithm in chapter 4. The combination of both controllers provides for accurate and robust walking velocity control. The adaptive controller developed in Chapter 5 is based on a neurocomputer. Unlike many neurocomputer walking algorithms, the neurocomputer in Chapter 5 does not have any prior knowledge of the biped's structure.
Appendix A gives the three dimensional extension to the WALK simulator described in Chapter 3. Appendix B is the WALK simulator users manual. Appendix C is an overview of biological research that is helpful in the design of mechanical bipeds.
Chapter 2

REVIEW OF BIPEDAL WALKING RESEARCH

Walking Algorithms

Most walking research is centered on deterministic algorithms for walking (gait generation). In deterministic algorithms, the walking control has built-in knowledge of the biped's structure. The walking gait is generated by solving for the needed foot locations by using the dynamic equations of motion for the biped. Another method of gait generation is some form of adaptive control system that learns the biped's dynamics and geometry. Adaptive control systems do not require as much a priori knowledge as deterministic controllers do. Adaptive walking controllers do not perform well until the biped dynamics and geometries are automatically determined.

Walking Methods

Walking algorithms include both static and dynamic walking types. Static walking is when the walking machine, or walker, is in static equilibrium at all times. During static walking, the walker can stop at any time and be stable. In static walking, the center of mass of the walker is above the area inscribed by the supporting foot. The center of mass
must be above the sole of the supporting leg. Static walking algorithms are easier to design because only the walker geometry (kinematics) needs to be considered. The center of mass location requirements limits the step size and speed for biped walkers.

In dynamic walking, the projected center of mass is allowed outside of the area inscribed by the feet. In this case, the walker may be falling during some part of the gait cycle. To achieve a limited form of stability, called dynamic stability, the walking gait is designed to have feet move in a manner that catches the walker when it is falling. Dynamic walking provides for a significant advantage. Dynamic walking can be much faster because the step lengths are larger. For bipedal walkers, dynamic walking permits smaller foot soles. Static bipedal walkers have very large feet or walk very slowly.

According to Todd, 1985 and Raibert, 1986, the first biped walkers appeared in the late 1960’s. Other multi-leg (four and six legged) walkers had research that started much earlier. The delay in the development of biped walker is due to problems of balance. The issue of balance is avoided with static walkers and crawlers. Many of the early biped walkers used only static algorithms due to the limitations of controller technology. These bipeds moved slowly and/or had large feet. Stability with these bipeds is achieved
passively. Active balance allows for stability with smaller feet and for dynamic walking and running.

**Deterministic Bipedal Walking Algorithms**


Miura uses a complex linearized dynamic model that includes the leg masses. Miura's method models the effect of coupling of the leg masses on the body trajectory. Miura computes the end of step position from the linearized state equation for a three or seven link biped.

The Zheng SD-2 biped robot is capable of both static and dynamic walking. Zheng's method is to design a feedback controller that causes the center of mass trajectory to be optimized from a stability margin point of view. In addition, Zheng has extended his algorithm to allow for walking on sloping surfaces by determining the slope from the biped kinematics. Zheng uses a complex multi-link model for the biped dynamics.

Raibert uses an extension of his hopping monoped to achieve bipedal running. Raibert's method uses a multi-link
dynamic model for the end of step prediction. The model has a mass and fixed inertia for each leg and a body mass. The biped's body attitude, body height, and foot placement are determined by decoupled control systems.

Hemami discusses methods for the analysis of bipeds with complex multi-link structures. These methods involve Lagrangian dynamic models of multi-link rigid bodies. The joint angle stability of coupled multi-link bipeds is also analyzed.

**Adaptive Bipedal Walking Algorithms**

Adaptive control of bipedal walking was investigated by Wang, 1989 and Kitamura, 1990. Both of these methods are based on some form of neuro-computer.

Wang's walking algorithm is based on using several hidden layer feedforward neuro-networks. One network is used for each walking function (height, body attitude, and stepping). The neuro-controllers are trained by using a multi-link model deterministic controller as reference.

Kitamura's walking controller is based on a Hopfield network in combination with a simple single mass dynamic model. The optimization (potential) function for the Hopfield network uses the biped's geometry and kinematic constraints. All of the neuro-controllers have some built-in knowledge about the biped's structure. This knowledge is either from a
deterministic controller as a reference or in the learning algorithm.

**Conclusion**

Most of the current research uses a complex dynamic model for the biped walker. These models are composed of a high order set of equations for predicting the biped motion for a given foot placement. Even the neuro-controller approaches use a similar dynamic model for learning. Many of these methods provide for point contact feet. Ankle torques are not considered. Thus, only dynamic walking is possible for many of these algorithms.

**Comparison of Thesis to Previous Techniques**

In comparison to passed methods, the simulator is this thesis is unique in several respects. First, the input to the simulator is position, not force or torque. Second, the floor interactions are model by inequality constraints or by assumed (forced) constraints. Third, variable time steps allow for more accurate floor impact modeling.

In contrast to these previous techniques, the biped walking algorithms in this thesis use only a simple single mass model of the biped. Detailed kinematic and kinetic information about the biped robot is not needed. The adaptive CMAC neural network algorithm is adjusted without any prior knowledge is the biped’s structure.
Chapter 3

TWO DIMENSIONAL DYNAMIC SIMULATOR

Introduction

This chapter describes a computer simulation program that was developed to facilitate the analysis of bipedal walking. The simulation program is called WALK. The WALK program is used to model the dynamic motion of a seven-link bipedal structure which is possibly in ground contact.

The WALK program models a simple two legged robot in the sagittal plane (forward/backward and up/down). The model for the biped are seven rigid links that are joined to form the body and the two legs (Figures 3.1 and 3.2). The body is one link and each leg is formed by three links. Each link is characterized by a length, mass, and moment of inertia.

This idealized model of a physical bipedal structure is used to test the interactions of biped dynamics on walking algorithms. Of particular interest is the effect of leg mass and inertia on a given walking algorithm. This effect and many others can be evaluated using the WALK simulator.

The WALK simulator is formulated in continuous time by using Lagrangian dynamics. The simulator solves the continuous time dynamic problem by evaluating the discretized continuous time model.
Simulator Design Philosophy

In the design of a dynamic system simulator, many tradeoffs need to be considered. These tradeoffs include: accuracy of the model, model complexity, simulation time, and ease of development. To choose the right tradeoffs for a given task, several points need to be reviewed. First, the important dynamic characteristics need to be identified. Second, the allowable simplifications need to be understood. Third, the appropriate inputs and outputs of the simulator need to be chosen. The performance of the simulator is directly tied to these issues.

For the problem of the walking algorithm analysis, it is most important to simulate the global behavior of a biped. The walking algorithms under study in this thesis assumes that leg positions can be controlled. This assumption defers the design of the leg position control system and concentrates the study on walking algorithms. The leg position assumption eliminates the need for modeling the internal forces required to sustain the leg position. As a result, the intra- and inter-link constraint forces are not required to be known explicitly. By eliminating these internal forces, the number of variables to be modeled is greatly reduced.

Several additional assumptions are made to simplify the simulator design and improve computer execution speed. The biped limbs and body are modeled as rigid masses. The angles
of the joints are rigid and do no deform with applied forces. The floor is modeled as an infinitely hard surface. All of
the ground impacts are modeled as perfectly plastic (no bouncing).

The distributed foot-ground constraint forces are modeled as a single equivalent force. The magnitude of these foot
forces constrain the feet to be above the ground surface. The location of these equivalent forces determines the ankle
torques.

By the preceding assumptions, the input variables to the WALK simulator model are the leg positions and location of the
equivalent feet constraint forces. The output from the model is the biped position, orientation, and ground constraint
forces. The position input and constrain force assumptions differ from prior simulators.

**Biped Simulation Model**

The biped model is an approximation used to simulate a physical biped. The model assumes that the biped has seven
connected rigid bodies, or links. Six of the links form the right and left legs. The remaining link is the body (see
Figure 3.1). The legs have knees that control each leg length. Each point of contact between the links forms a joint
with a varying angle. These joint angles are inputs to the simulator.
The foot model is simplified by using ankle torques instead of ankle positions. By using ankle torques, the issues of compliant feet and ground are deferred. The simulator assumes that when a foot is in ground contact, the foot is flat on the ground. The angles are chosen to keep the feet parallel to the ground at all times.

Each link in the biped model has a given geometry, mass and moment of inertia. The limb angles are determined from the desired foot positions by the link geometries. The walking algorithm under test provides for the desired foot positions from which the joint angles are determined. A simple stick diagram of the biped in the sagittal plane is shown in Figure 3.1.
Where:

- $L_i$ are the link lengths.
- $K_i$ are the locations of the center of masses.
- $X_0, Y_0, \theta_0$ are the body positions.
- $\phi_1, \phi_2, \phi_3$ are the right leg joint angles.
- $\phi_4, \phi_5, \phi_6$ are the left leg joint angles.

**Room Coordinate System**

The simulator uses an inertial coordinate system that is attached to the ground. This coordinate system is called the room coordinate system. In the sagittal plane, up and down is labeled $y$, and right and left is labeled $x$. Gravity is downwards in the minus $y$ direction. The biped's hip is located by the vector $\{x_0, y_0\}$ and the body rotation is $\phi_0$. The
feet are located in room coordinates by the vectors \( \{x_l, y_l\} \) and \( \{x_r, y_r\} \). The room coordinate system is shown in Figure 3.2.

![Figure 3.2 Room Coordinate System](image)

Where:

- \( \theta_0 \) is the body orientation.
- \( X_0, Y_0 \) is the body position.
- \( X_r, Y_r \) is the right leg position.
- \( X_l, Y_l \) is the left leg position.

**Floor Constraints**

There are several ways to model the effects of ground contact. Two methods are particularly attractive. The first method is to constrain the foot's locations. The second method is to model the ground as a damped mass-spring system.
By using the constraint method, the foot-floor constraints are modeled as a requirement that a foot remains fixed in position when the foot is in ground contact. The foot constraint is valid only when the foot is in compression. When the constraint forces cause the ground-foot interface to be in tension, the foot position constraint is relaxed (removed).

The constraint method provides for two effects: The floor is modeled as an infinitely hard surface. All foot impacts with the floor are perfectly plastic so there is no bouncing.

The mass-spring method models the floor as a spring and dashpot between the foot and floor. The model is invoked for compression force only. Any tension force is modeled as zero force.

The mass-spring method allows for finite floor compressions and bouncing. The degree of bouncing is determined by the damping used.

Both the mass-spring and constraint methods provide for similar results when the floor impacts are soft (i.e. vertical velocity is near zero when the foot impacts the floor). The mass-spring method requires shorter simulation time steps due to the higher speed of ground impulses. For that reason, the constraint method is adopted for the WALK simulator.
Two Dimensional Continuous Time Model

The simplest model for biped walking is to model the biped in a two dimensional plane. The plane that is chosen is a "side view" of walking - the sagittal plane. Many walking algorithms solve the walking problem by considering walking as two decoupled two dimensional problems. Balance in the sagittal and frontal planes are solved independently. The two dimensional model simplification cannot test the validity of the decoupling assumption. From the dynamic equations, the frontal and sagittal coupling appears as angular velocity product terms (see appendix A). If these terms are small they can be neglected. Most walking algorithms try to keep all the angular velocities small. As a result the decoupling assumption is normally valid.

The two dimensional model has to be solved for only three unknowns (two translations and one angle), while the three dimensional model requires six unknowns (three translations and three angles). In addition, the geometric transforms required for a multi-link model in three dimensions are much more complex to evaluate than are those for two dimensions. The two dimensional model is much faster to simulate due to its reduced dimensionality.

Kinematics

Any point on the biped's structure can be determined by the sum of the vectors for each link and the body position
vector. In two dimensions, the projection of a link on to the x and y coordinates is determined by the total link rotation. The total x or y coordinate of any position is the sum of all the corresponding links' projections and the body position. The center of mass for the \( i \)th link is given by:

\[
X_i = X_0 - \sum_{j \neq p(i)} L_j \sin(\theta_j) + K_i \sin(\theta_i)
\]

\[
Y_i = Y_0 + \sum_{j \neq p(i)} L_j \cos(\theta_j) + K_i \cos(\theta_i)
\]

\[
\theta_j = \sum_{k \neq p(i)} \phi_k + \theta_0
\]

Where:

\[
p(i) = [0 \text{ for } 0 < i \leq 3] \\
        [4 \text{ for } 3 < i \leq 6]
\]

\( L_j \) is the \( j \)th link length.

\( K_j \) is the \( j \)th link center of mass.

\( \theta_j \) is the \( j \)th link total rotation.

\( \phi_k \) is the \( k \)th link joint angle.

**Velocities and Accelerations**

The acceleration equations for each link will be required to solve the equations of motion. The velocity and acceleration equations can be computed by definition of the total derivative. The velocity is equal to the sum of the
partial derivatives of the link position with respect to each
degree of freedom, multiplied by their corresponding time
derivatives. Similarly, the acceleration can be calculated
from the first and second partial derivatives and time
derivatives.

\[
\dot{x}_i = \dot{x}_0 + \sum_j \frac{\partial x_i}{\partial \theta_j} \dot{\theta}_j
\]

\[
\dot{y}_i = \dot{y}_0 + \sum_j \frac{\partial y_i}{\partial \theta_j} \dot{\theta}_j
\]

\[
\dot{\theta}_i = \dot{\theta}_0 + \sum_j \dot{\phi}_j
\]

\[
\ddot{x}_i = \ddot{x}_0 + \sum_j \frac{\partial^2 x_i}{\partial \theta_j^2} \dot{\theta}_j^2 + \sum_j \frac{\partial x_i}{\partial \theta_j} \ddot{\theta}_j
\]

\[
\ddot{y}_i = \ddot{y}_0 + \sum_j \frac{\partial^2 y_i}{\partial \theta_j^2} \dot{\theta}_j^2 + \sum_j \frac{\partial y_i}{\partial \theta_j} \ddot{\theta}_j
\]

\[
\ddot{\theta}_i = \ddot{\theta}_0 + \sum_j \ddot{\phi}_j
\]

The biped's degrees of freedom are the body position,
body orientation, and joint angles. The time derivatives are
the relative velocity and acceleration of the legs and body.

The inputs to the simulator are the joint angles. These inputs are positions as a function of time. Joint angle
velocities and acceleration are not explicitly given and must
be computed by a discrete approximation. One method of calculating the link accelerations is by using discrete approximations of the joint velocities and accelerations, and then multiplying these values by the required partial derivatives. A second method is to directly calculate the link accelerations by discretizing the segment positions. The latter method eliminates the need for all of the partial derivatives with respect to the joint angles. The only partial derivatives that remain are with respect to the body angles.

\[ \ddot{x}_i = \ddot{x}_0 + \frac{\partial x_i}{\partial \theta_0} \dot{\theta}_0^2 + \frac{\partial^2 x_i}{\partial \theta_0^2} \dot{\theta}_0^4 + \ddot{x}_i \]  

(3.10)

\[ \ddot{y}_i = \ddot{y}_0 + \frac{\partial y_i}{\partial \theta_0} \dot{\theta}_0^2 + \frac{\partial^2 y_i}{\partial \theta_0^2} \dot{\theta}_0^4 + \ddot{y}_i \]  

(3.11)

\[ \ddot{\theta}_i = \ddot{\theta}_0 + \ddot{\theta}_i \]  

(3.12)

Where:
- \( \ddot{x}_{ri} \) is the relative x axis acceleration of the \( i^{th} \) link to the body.
- \( \ddot{y}_{ri} \) is the relative y axis acceleration of the \( i^{th} \) link to the body.
- \( \ddot{\theta}_{ri} \) is the relative angular \( \theta \) acceleration of the \( i^{th} \) link to the body.

**Automated Partial Derivative Computation**

Partial derivatives of link positions with respect to each degree of freedom must be calculated for the evaluation of the dynamic equations. The partial derivatives can be
calculated by either finite difference or analytically. The analytical method provides for fewer computations and better accuracy. For these reasons, the analytical method is chosen. The first and second partial derivatives are evaluated for each center of mass position. The floor constraints require partial derivatives of the foot positions. These partial derivatives are calculated in a like manner.

The partial derivatives of the link center of mass positions with respect to the body orientation are given by:

\[
\frac{\partial X_i}{\partial \theta_0} = -\sum_{j \in p(i)} \left( L_j \cos(\theta_j) + K_i \cos(\theta_i) \right) \tag{3.13}
\]

\[
\frac{\partial Y_i}{\partial \theta_0} = -\sum_{j \in p(i)} \left( L_j \sin(\theta_j) + K_i \sin(\theta_i) \right) \tag{3.14}
\]

\[
\frac{\partial^2 X_i}{\partial \theta_0^2} = \sum_{j \in p(i)} \left( L_j \sin(\theta_j) + K_i \sin(\theta_i) \right) \tag{3.15}
\]

\[
\frac{\partial^2 Y_i}{\partial \theta_0^2} = -\sum_{j \in p(i)} \left( L_j \cos(\theta_j) + K_i \cos(\theta_i) \right) \tag{3.16}
\]

**Inverse Kinematic Model**

The walking algorithm provides for the desired foot locations. These feet locations, in body relative coordinates, are used to calculate the joint angles. This calculation task is usually called the inverse problem (Paul,
1981, Coiffet, 1983). Figure 3.3 shows a kneed leg. The inputs to the inverse calculation algorithm are: $\theta_0$, $\theta_3$, $x_{rf}$, $y_{rf}$, $x_0$, and $y_0$. Note that $\theta_3$ is the total rotation of the ankle and is chosen such that the ankle is parallel to the floor (0.0 rads). This notation is for the right leg. The left leg notation is the same, but for the substitution of the corresponding joint angles.

![Figure 3.3 Inverse Model](image-url)
The foot position equations for each foot are:

\[ L_1 \sin(\theta_1) + L_2 \sin(\theta_2) + L_3 \sin(\theta_3) = x - x_0 \quad 3.17 \]

\[ L_2 \cos(\theta_1) + L_2 \cos(\theta_2) + L_3 \cos(\theta_3) = y - y_0 \quad 3.18 \]

\[ \theta_1 = \phi_0 + \phi_1 \quad 3.19 \]

\[ \theta_2 = \phi_0 + \phi_1 + \phi_2 \quad 3.20 \]

\[ \theta_3 = \phi_0 + \phi_1 + \phi_2 + \phi_3 \quad 3.21 \]

Where:

- \( L_1, L_2, L_3 \) are the link lengths.
- \( \phi_1, \phi_2, \phi_3 \) are the joint angles.
- \( \theta_1, \theta_2 \) are the total link rotations.
- \( \theta_0 \) is the body orientation.
- \( \theta_3 \) is the total ankle rotation.

The above equations can be solved for the required unknowns, \( \phi_1, \phi_2, \) and \( \phi_3 \). Equations 3.17 and 3.18 can be squared and added to each other. This results in the solution for \( \phi_2 \) (Equations 3.22, 3.23, 3.24). Equations 3.17 and 3.18 can be rewritten using a trigonometric identity and solved for \( \phi_1 \) (Equation 3.25). The remaining unknown, \( \phi_3 \) can be solved using the total rotation equations (Equations 3.26). The results are:

\[ \cos(\phi_2) = \frac{(x - L_3 \sin(\theta_3))^2 + (y + L_3 \cos(\theta_3))^2}{2L_2 L_3} \quad 3.22 \]
\[
\sin(\varphi_2) = \sqrt{1 - \cos(\varphi_2)^2}
\]

\[
\varphi_2 = \tan^{-1} \frac{\sin(\varphi_2)}{\cos(\varphi_2)}
\]

\[
\varphi_1 = -\theta_0 + \tan^{-1} \left( \frac{L_2 \cos(\varphi_2) + L_1 (x - L_2 \sin(\theta_3))}{L_2 \sin(\varphi_2) (x - L_3 \sin(\theta_3))} \right) - \frac{L_2 \sin(\varphi_2) (y + L_3 \cos(\theta_3))}{(L_2 \cos(\varphi_2) + L_1) (y + L_2 \cos(\theta_3))}
\]

\[
\varphi_3 = \theta_3 - \varphi_2 - \varphi_1 - \theta_0
\]

**Lagrangian Dynamic Model**

There are several ways to form the dynamic equations of motion for mechanical systems (D'Souza, 1984, Keith 1961). One method is to apply Newton-Euler equations for each link and to solve this system of equations using the constraints that the links are connected. This method results in solving for all of the link constraint forces and link dynamics. This type of solution provides for a very large number of unknowns that are not of interest. A second method of solving the dynamics problem is to find an expression for the total kinetic and potential energy for the biped. Then Lagrangian dynamics are used to solve for the desired unknowns from the energy equations. Lagrangian dynamics are convenient because the unknowns can be chosen to be the body position and orientation. The energy equations can be chosen to be in any convenient set of coordinates.
\[
\frac{d}{dt} \frac{\partial x}{\partial q_j} - \frac{\partial L}{\partial q_j} = \sum_k \lambda_k \frac{\partial c_k}{\partial q_j} + \sum_p \bar{F}_p \frac{\partial \bar{r}_p}{\partial q_j} 
\]

3.27

\[
L = \sum \frac{1}{2} m_i (\dot{x}_i^2 + \dot{y}_i^2) + \frac{1}{2} I_i \dot{\theta}_i^2 - m_i g y_i 
\]

3.28

Where:

- \( m_i \) is the mass of the \( i^{th} \) link.
- \( I_i \) is the inertia of the \( i^{th} \) link.
- \( q_j \) are the unknowns \( x_0, y_0, \) and \( \theta_0 \).
- \( c_k \) are the floor constraints (if active).
- \( \lambda_k \) are the constraint forces.
- \( x_i, y_i \) are the link center of masses.
- \( L \) is the Lagrangian (KE - PE).
- \( m_i \) is the \( i^{th} \) link mass.
- \( I_i \) is the \( i^{th} \) link inertia.
- \( g \) is gravity.
- \( \bar{r}_p \) are the vectors locating each force.
- \( \bar{F}_p \) are the external forces.

The two right-hand terms in Equation 3.27 are the generalized forces (not including gravity). The first right-hand term is the notation for a constraint. The second right-hand term is for an external force that is not a conservative force included in the potential energy. The external forces for modeling the ground are applied in the same manner as the constraint forces for modeling the ground. Due to this equivalence, the right-hand side vector dot product(s) and the partial derivatives result in the same equations for this application. As a result, the forces of constraint and the external forces are the same, (i.e. the components of \( F \) are equal to the corresponding \( \lambda \).s.
The joint angles in the biped are inputs and are not unknowns. In the two dimensional case, there are only three positional unknowns: $x_0$, $y_0$, and $\theta_0$. The following equations are the continuous time equations of motion for the biped in two dimensions:

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{x}_0} - \frac{\partial L}{\partial x_0} = \sum_I m_I \ddot{x}_I = \sum_k \lambda_k \frac{\partial c_k}{\partial x_0} + Q_x \tag{3.29}
\]

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{y}_0} - \frac{\partial L}{\partial y_0} = \sum_I m_I (\ddot{y}_I + g) = \sum_k \lambda_k \frac{\partial c_k}{\partial y_0} + Q_y \tag{3.30}
\]

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_0} - \frac{\partial L}{\partial \theta_0} = \sum_I m_I (\ddot{x}_I \frac{\partial c_k}{\partial \theta_0} + (\ddot{y}_I + g) \frac{\partial y_I}{\partial \theta_0}) + I_I \ddot{\theta}_I = \sum_k \lambda_k \frac{\partial c_k}{\partial \theta_0} + Q_\theta \tag{3.31}
\]

Where: $Q_x$ is the generalized force in the $x$ direction. $Q_y$ is the generalized force in the $y$ direction. $Q_\theta$ is the generalized force in the $\theta$ direction.

**Dynamics Matrix**

The acceleration (3.7, 3.8, and 3.9) and partial derivative (3.13, 3.14, 3.15, and 3.16) expressions can be substituted into the dynamics equations (3.29, 3.30, and 3.31). The result is a system of nonlinear differential equations. The unknowns are the body’s linear and angular accelerations and the angular velocity. The resulting equations can be summarized in matrix form as follows.
\[ M\dddot{\mathbf{s}} + C\dddot{s}^2 = Q\dddot{\mathbf{F}} + K \]

\[ M = \begin{bmatrix}
\sum_i m_i & 0 & \sum_i m_i \frac{\partial x_i}{\partial \theta_0} \\
0 & \sum_i m_i & \sum_i m_i \frac{\partial y_i}{\partial \theta_0} \\
\sum_i m_i \frac{\partial x_i}{\partial \theta_0} & \sum_i m_i \frac{\partial y_i}{\partial \theta_0} & \sum_i m_i \left[ \left( \frac{\partial x_i}{\partial \theta_0} \right)^2 + \left( \frac{\partial y_i}{\partial \theta_0} \right)^2 \right] + I_i
\end{bmatrix} \]

\[ C = \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & \sum_i m_i \left[ \frac{\partial x_i}{\partial \theta_0} \frac{\partial^2 x_i}{\partial \theta_0^2} + \frac{\partial y_i}{\partial \theta_0} \frac{\partial^2 y_i}{\partial \theta_0^2} \right]
\end{bmatrix} \]

\[ Q = \begin{bmatrix}
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
\frac{\partial x_r}{\partial \theta_0} & \frac{\partial y_r}{\partial \theta_0} & \frac{\partial x_i}{\partial \theta_0} & \frac{\partial y_i}{\partial \theta_0}
\end{bmatrix} \]

\[ K = \begin{bmatrix}
-\sum_i m_i \dddot{x}_{x_i} \\
-\sum_i (m_i \dddot{y}_{y_i} + g) \\
-\sum_i m_i \left[ \frac{\partial x_i}{\partial \theta_0} \dddot{x}_{x_i} + \frac{\partial y_i}{\partial \theta_0} (\dddot{y}_{y_i} + g) + I_i \theta_{x_i} \right]
\end{bmatrix} \]

where: \[ s = [x_0, y_0, \theta_0] \]
\[ \lambda = [\lambda_{xx}, \lambda_{yx}, \lambda_{x\theta}, \lambda_{y\theta}] \]
Two Dimensional Discretized System

To simulate continuous time equations on a computer, the continuous equations must be discretized (D'Souza, 1984, Smith, 1977). There are many ways to form a discrete time approximation to a continuous time system. These methods provide for a trade off between discretization errors and computation time. The method that is chosen for the WALK simulator is the central difference method.

The walking problem is a very nonlinear problem due to the modeling of floor impacts. High order discrete approximations provide for improved accuracy if the continuous time problem has continuous derivatives. The floor impacts are discontinuous. Low order approximations with shorter time steps are more appropriate. The central difference approximation provides for a stable model with good phase fidelity.

The central difference approximation for acceleration and the forward difference approximation for the velocity are shown in Equations 3.37, 3.38, 3.39, and 3.40.
\[ \dot{X}_0(t) = \frac{X_0(t + \Delta t_1) - 2X_0(t) + X_0(t - \Delta t_2)}{\Delta t_1 \Delta t_2} \quad 3.37 \]

\[ \ddot{Y}_0(t) = \frac{Y_0(t + \Delta t_1) - 2Y_0(t) + Y_0(t - \Delta t_2)}{\Delta t_1 \Delta t_2} \quad 3.38 \]

\[ \dddot{\theta}_0 = \frac{\theta_0(t + \Delta t_1) - 2\theta_0(t) + \theta_0(t - \Delta t_2)}{\Delta t_1 \Delta t_2} \quad 3.39 \]

\[ \dot{\theta}_0(t) = \frac{\theta_0(t) - \theta_0(t - \Delta t_2)}{\Delta t_2} \quad 3.40 \]

Where:

\[ \Delta t_1, \Delta t_2 \] are current and passed simulation time steps.

The body relative accelerations also must be discretized. The accelerations are calculated by storing the past values of the joint angles. These joint angles are used with the current body state to calculate the relative limb accelerations. The discretization is done by the central difference method.
\[
\ddot{X_i}(t+\Delta t) = \frac{X_i(t^+) - 2X_i(t) + X_i(t^-)}{\Delta t_1 \Delta t_2} 
\]

3.41

\[
\ddot{Y_i}(t+\Delta t_2) = \frac{Y_i(t^+) - 2Y_i(t) + Y_i(t^-)}{\Delta t_1 \Delta t_2} 
\]

3.42

\[
\ddot{\theta_i}(t+\Delta t_1) = \frac{\theta_i(t^+) - 2\theta_i(t) + \theta_i(t^-)}{\Delta t_1 \Delta t_2} 
\]

3.43

Where:

\[
X_i(t^+) = X_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t+\Delta t_1), \ldots) 
\]

\[
Y_i(t^+) = Y_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t+\Delta t_1), \ldots) 
\]

\[
\theta_i(t^+) = \theta_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t+\Delta t_1), \ldots) 
\]

\[
X_i(t) = X_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t), \ldots) 
\]

\[
Y_i(t) = Y_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t), \ldots) 
\]

\[
\theta_i(t) = \theta_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t), \ldots) 
\]

\[
X_i(t^-) = X_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t-\Delta t_2), \ldots) 
\]

\[
Y_i(t^-) = Y_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t-\Delta t_2), \ldots) 
\]

\[
\theta_i(t^-) = \theta_i(X_0(t), Y_0(t), \theta_0(t), \theta_1(t-\Delta t_2), \ldots) 
\]

The final dynamics equation for the biped robot is found by substituting the discretization Equations 3.41, 3.42, and 3.43 into the dynamics Equation 3.32. This result is shown in Equation 3.44.
\[ M \delta s + \Delta^2 \lambda = M(s(t) - (t - \Delta t_1)) - C(s(t) - s(t - \Delta t_2))^2 \frac{\Delta t_1}{\Delta t_2} + \Delta^2 K \]

\[ M \delta s - \dot{\lambda} = \dot{K} \]

Where:
\[ \delta S = S(t) - S(t - \Delta) \]
\[ S = [\delta x_0, \delta y_0, \delta \theta_0]^T \]

\[ \delta x_0 \] is the x axis body position increment.
\[ \delta y_0 \] is the y axis body position increment.
\[ \delta \theta_0 \] is the \( \theta \) axis body rotation increment.
\[ \Delta^2 = \Delta t_1 \Delta t_2 \]

**Floor Constraints**

The dynamic equations are solved with the constraint that any foot must not pass through the floor. This is an inequality constraint and is solved by simulating the system in a piecewise manner (Flechter, 1981). The constraints are either active or inactive. An active constraint is when the inequality is replaced by an equality constraint. An inactive constraint is a constraint that is removed from consideration. The active constraint equations are formed by requiring that the active foot does not move (the active foot is the supporting foot).

The constraint equations are derived by the Taylor series approximation of the foot locations. Because the constrained feet are stationary, the foot position increment is then equal to zero. Using the same method as for the acceleration calculations (Equations 3.10, 3.11, and 3.12), the internal
movements are evaluated by discrete methods. The constraint

equations are:

\[ \delta X_f = \delta X_0 + \frac{\partial X_f}{\partial \theta_0} (t) \delta \theta_0 + \delta x_{rf} = 0 \]  

\[ \delta Y_f = \delta Y_0 + \frac{\partial Y_f}{\partial \theta_0} (t) \delta \theta_0 + \delta y_{rf} = 0 \]

Where:

- \( \delta X_f \) is x axis foot position increment.
- \( \delta Y_f \) is y axis foot position increment.
- \( \delta x_{rf} \) is the x axis body relative foot position increment.
- \( \delta y_{rf} \) is the y axis body relative foot position increment.

By combining the x and y axis constraint equations for both the right and left legs, the matrix Equation 3.47 is obtained.

\[ Q^T \delta s = P \]

\[ P = \begin{bmatrix}
X_{f_r} - X_{rf_1} \\
Y_{f_r} - Y_{rf_1} \\
X_{f_l} - X_{rf_1} \\
Y_{f_l} - Y_{rf_1}
\end{bmatrix} \]

Where:

The \( r \) and \( l \) subscripts indicate the right and left legs respectively.

Equation 3.47 is for active constraints for both legs. The constraints are always applied in x and y pairs. There are four possible combinations of active and passive states.
for two legs. When the constraints are inactive, the Equation 3.47 is replaced by Equation 3.48.

$$I\lambda = 0$$  \hspace{1cm} 3.48

Where: I is the identity matrix.

The Equation 3.48 requires that the constraint force are equal to zero. Combinations of Equations 3.47 and 3.48 are used when a single leg (right or left) is active.

**Discrete Matrix Equation Solution**

The discretized system of Equations (3.44, and 3.47/8) are solved to compute the body position and constraint forces at the next time step. This system of equations is overdetermined for two feet (in two dimensions), since there are four constraint equations and three position unknowns. Three methods for solving this over determined system have been reviewed. These three methods are: using mass-spring forces instead of hard constraints, using singular value decomposition, and using row reduction.

The mass-spring method allows for small displacements in the floor-foot interface. These displacements give rise to the applied force. The forces are uniquely determined by having the floor displacements as unknowns. Note, there is one displacement unknown for each constraint.
The singular value decomposition method can solve an over determined system of equations. However, the singular value algorithm is much slower in computer execution speed.

The row reduction method is done by linearly transforming the four constraint equations into three equations. These methods result in three constraints and three position unknowns. As a result of the transformation from four forces into three unknowns, the four forces cannot be uniquely determined. The y axis forces are tangential to the floor and are convenient for determining if the legs are in tension or compression. Thus, the transformation is used simply to sum the x axis constraints together. The Ω' matrix must be
modified to account for the x axis force only once. The row reduced matrix equations are:

\[
\begin{bmatrix}
\Delta_1 \Delta_2 & 0 & 0 \\
0 & \Delta_1 \Delta_2 & \Delta_1 \Delta_2 \\
\Delta_1 \Delta_2 \left( \frac{\partial x_{rf}}{\partial \theta_0} + \frac{\partial x_{lt}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \left( \frac{\partial y_{rf}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \left( \frac{\partial y_{lt}}{\partial \theta_0} \right)
\end{bmatrix}
\begin{bmatrix}
\delta x_0 \\
\delta y_0 \\
\delta \theta_0
\end{bmatrix}
= \begin{bmatrix}
\dot{x}_r + x_{r_t} - x_{r_l} - x_{rf} \\
y_r - y_{rf} \\
y_{r_t} - y_{lt}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\Delta_1 \Delta_2 & 0 & 0 \\
0 & \Delta_1 \Delta_2 & \Delta_1 \Delta_2 \\
\Delta_1 \Delta_2 \left( \frac{\partial x_{rf}}{\partial \theta_0} + \frac{\partial x_{lt}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \left( \frac{\partial y_{rf}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \left( \frac{\partial y_{lt}}{\partial \theta_0} \right)
\end{bmatrix}
\begin{bmatrix}
\delta x_0 \\
\delta y_0 \\
\delta \theta_0
\end{bmatrix}
= \begin{bmatrix}
\dot{x}_r \left[ \begin{array}{c}
\dot{x}_r \\
y_r \left[ \begin{array}{c}
lx \\
\lambda y
\end{array} \right]
\end{array} \right]
\end{bmatrix}
\]

\[
\begin{bmatrix}
\Delta_1 \Delta_2 & 0 & 0 \\
0 & \Delta_1 \Delta_2 & \Delta_1 \Delta_2 \\
\Delta_1 \Delta_2 \left( \frac{\partial x_{rf}}{\partial \theta_0} + \frac{\partial x_{lt}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \left( \frac{\partial y_{rf}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \left( \frac{\partial y_{lt}}{\partial \theta_0} \right)
\end{bmatrix}
\begin{bmatrix}
\delta x_0 \\
\delta y_0 \\
\delta \theta_0
\end{bmatrix}
= \begin{bmatrix}
\dot{x}_r \left[ \begin{array}{c}
\dot{x}_r \\
y_r \left[ \begin{array}{c}
lx \\
\lambda y
\end{array} \right]
\end{array} \right]
\end{bmatrix}
\]

3.50

3.51

38

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[
\begin{bmatrix}
\Delta_1 \Delta_2 & 0 & 0 \\
0 & \Delta_1 \Delta_2 & \Delta_1 \Delta_2 \\
\Delta_1 \Delta_2 \left( \frac{\partial x_{rf}}{\partial \theta_0} + \frac{\partial x_{ff}}{\partial \theta_0} \right) & \Delta_1 \Delta_2 \frac{\partial y_{rf}}{\partial \theta_0} & \Delta_1 \Delta_2 \frac{\partial y_{ff}}{\partial \theta_0} \\
1 & 0 & \frac{\partial x_{rf}}{\partial \theta_0} \\
0 & 0 & 0 \\
0 & 1 & \frac{\partial y_{rf}}{\partial \theta_0}
\end{bmatrix}
\begin{bmatrix}
\frac{\delta \lambda_0}{\delta x_0} \\
\frac{\delta \lambda_0}{\delta y_0} \\
\delta \theta_0
\end{bmatrix} = \begin{bmatrix}
k \\
\hat{k} \\
\lambda x \\
\lambda y \\
0 \\
Y_{rf} - Y_{ff}
\end{bmatrix}
\]

The following table summarizes the correct usage for Equations 3.49 through 3.52:

<table>
<thead>
<tr>
<th>Right leg</th>
<th>Left leg</th>
<th>Equation used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Active</td>
<td>3.49</td>
</tr>
<tr>
<td>Active</td>
<td>Passive</td>
<td>3.50</td>
</tr>
<tr>
<td>Passive</td>
<td>Active</td>
<td>3.51</td>
</tr>
<tr>
<td>Passive</td>
<td>Passive</td>
<td>3.52</td>
</tr>
</tbody>
</table>

The non-singularity of Equation 3.49 can be tested by calculating the determinant of Equation 3.49. The condition for non-singularity is shown in Equation 3.53.

\[
\frac{\partial y_{rf}}{\partial \theta_0} - \frac{\partial y_{ff}}{\partial \theta_0} \neq 0
\]

Equation 3.49 is for a fully constrained system. This is seen by noting that the mass and inertia terms are missing and only the constraint terms remain. The non-singularity of Equation 3.52 can be tested in a similar manner by calculating.
the determinant of Equation 3.52. The non-singularity condition is shown in Equation 3.54.

\[ \sum_{i} m_i \left[ \sum_{j} m_j \left( \frac{\partial X_i}{\partial \theta_j} \right)^2 \right] + I_i = \left( \sum_{j} m_j \frac{\partial X_i}{\partial \theta_j} \right)^2 \left( \sum_{j} m_j \frac{\partial X_i}{\partial \theta_j} \right)^2 \] 3.54

As expected, Equation 3.54 indicates the total dependance of the dynamic system on the mass and inertia. The other two solutions are linear combinations of Equations 3.49 and 3.52.

**Floor Intercept Correction**

The discrete nature of the simulation algorithm has difficulty with modeling ground impacts with constant time increments. A ground impact may not occur at time sampling times of the simulation. Due to this effect, the constraints would become active inside the ground.

To eliminate this problem, variable time steps are used. First the next time step is calculated. Then, if any constraints change from inactive to active, a new shorter time step is calculated. This new time step is the incremental time required until ground impact. This time interval is
approximated by linear interpolation. The following diagram shows a ground impact (see Figure 3.4).

![Diagram of ground impact](image)

Figure 3.4 Floor Impact

The linear interpolated time increment is calculated by Equation 3.55.

\[
\delta t = \Delta t \frac{g_i - Y_i}{Y_{i+1} - Y_i - g_{i+1} + g_i}
\]

3.55

Where:
- \( \delta t \) is the new time increment.
- \( \Delta t \) is the base time step.
**Computer Algorithm**

The flow chart for the simulation algorithm is:

1. **Start New Iteration**
2. **Compute New Feet Locations**
   (Walking Algorithm & Equations 3.22 to 3.26)
3. **Compute Partial Derivatives**
   (Equations 3.13 to 3.16)
4. **Compute Internal Accelerations**
   (Equations 3.31 to 3.43)
5. **Setup Continuous System**
   (Equations 3.33 to 3.36)
6. **Setup Discrete System**
   (One of Equations 3.49 to 3.52)
7. **Remove Constraints for Non-contacting Feet**
8. **Remove Constraints for Singular Feet (Equal Position)**
9. **Start Tension Test Loop**
10. **Solve System for Next Position & Forces**
11. **Remove Constraints for Legs in Tension**
    - yes
      - **Constraints Changed ?**
    - no
12. **Linear Interpolate Ground Impact, if Needed**
13. **Store Results**
14. **Done**

42
Simulation Results

The best verification of the WALK simulator is its application with the deterministic walking algorithm in Chapter 4. The walking algorithm was independently derived from basic mechanics just as the WALK simulator was. The operation of both programs together helps them verify each other. The examples included here are a samples of the tests used to check out the WALK simulator.

In this section, some examples are shown to demonstrate the WALK simulator. The first example is shown in Figure 3.5. Figure 3.5 shows the biped falling at an angle onto the floor. The two plots in Figure 3.5 demonstrate the effects of ground impact interpolation. Without interpolation, the discretization of time causes the biped to pass through the floor. With interpolation, the simulator forces a time step to occur coincident with the floor impact.
Internal accelerations cause changes in the angular momentum of the biped with respect to the body or internal reference frame. The total acceleration is the sum of the accelerations of each link with respect to the body plus the acceleration of the body with respect to inertial space. In the next example, Figure 3.6 shows the effects of modeling internal accelerations. The biped jumps into the air with a constant acceleration. When the legs reach their maximum programmed extension, the legs stops their acceleration trajectory. This sudden stop provides for a change in the angular momentum of the legs.
The two plots are with and without internal accelerations. The plot without internal accelerations maintains a constant orientation. The plot with internal accelerations changes its orientation due to the leg's change in angular momentum.

Figure 3.6 Jumping Example
In the next example, constant height walking is demonstrated. The walking algorithm used is explained in Chapter 4.

Figure 3.7 Walking Example
In the last figure, Figure 3.8, the walking example in Figure 3.7 is repeated for several simulation time steps. The time step axis is in powers of 2 from 1 ms to 128 ms. With larger simulation time steps, the dynamics of walking change. Valid time steps should not change the results for small variations around the time step that is used. The time step used for most of the simulations in this thesis is 10 ms.
Conclusion

Chapter 3 has described the WALK biped robot simulator. This simulator is used in the following chapters to study walking dynamics. Like any simulator, this program requires an understanding of its algorithms to enable the best use of it. The correct time step for a given simulation should be determined by comparing the results for several different time steps. The author has found that smaller time steps are almost always more accurate and always more time consuming.

The WALK simulator can be extended to three dimensions. A derivation for the required algorithm is shown in Appendix A.
Chapter 4

DETERMINISTIC WALKING ALGORITHM

Introduction

Chapter 4 describes a control algorithm for implementing a walking biped controller in two or three dimensions. The algorithm's objective is to control the body velocity and height of the biped. This objective is achieved by a feedback controller. The controller functions by measuring the state of the biped at discrete time intervals and by finding the required foot positions and ankle torques to achieve dynamically stable walking. The walking algorithm described in this chapter is similar to the hopping algorithms in Raibert, 1986.

Bipedal Robot Structure

The biped robot is a two-legged structure which supports a platform. The platform is considered the biped's body. The legs are articulated in a manner that can be used to position either of the legs at any desired location relative to the body. The only leg position limitation is the leg length and possible leg collisions. At the end of each leg is a rigid foot. The angle of the foot can be controlled by applying
ankle torques. An idealized form of a biped can be seen in Figure 4.1.

The leg length is then determined by the angle of its knee, or by the amount of extension. For the purposes of the algorithm, the leg is assumed to be of the extension variety. A kneed leg can be considered a virtual extension leg. The leg length is assumed to be continuously adjustable from some minimum to maximum value. The leg length limits are determined by the mechanics of the particular embodiments that are used.
**Background on Walking**

The body velocity (or position) of a biped is determined by the combined effects of gravity and ground reaction forces. The ground reaction forces are transmitted to the body by the feet and legs. The ground reaction forces can be modeled as a single force acting on the sole of the foot. The location of the force vector in the foot is determined by the ankle torque. By applying different ankle torques, the center of force in the foot can be moved anywhere inside the area that is spanned by the sole of the foot. The center of force concept is shown in the free body diagram in Figure 4.2.

![Figure 2.2 Center of Force](image)

Ankle torque must be limited for the foot to remain flat on the ground. If the ankle torque is too large, the foot will rotate up on one side. Any additional ankle torque,
beyond the flat foot limits, will not provide any additional change in the body forces. Flat foot contact is desirable for two reasons. First, flat foot contact provides for some mechanical stability. Second, when the foot is in flat contact, the kinematics of the supporting leg and foot allow for the derivation of the body position and velocity.

Walking is characterized by the alternating transfer of weight between each leg. Stability and body velocity control are done by the correct positioning of the feet and control of the ankle torques. During walking, one foot (the active foot) supports the body weight and the other (the passive foot) is moving to the next step. The control inputs are the position of the next foot and the continuously adjustable ankle torques.

Static walking is when the body velocity can be continuously controlled by ankle torques that are less than the flat foot limits. During static walking the body velocity can be stopped at anytime without violating the flat foot torque limit and causing the body to fall down. Static walking has several advantages. Small errors in the predicted dynamics of the biped do not affect the body velocity. In static walking, the body velocity can be continuously controlled. The main disadvantage of static walking is that the maximum velocity is limited by the foot size.

Dynamic walking does not rely on continuous ankle torque adjustments to provide stability. Stability and velocity are
controlled, on average, by controlling the feet positions. The biped's motion is determined by the dynamics of the body during each step of dynamic walking. The one and only control input is the next active foot position. A control system based on only next-step positions has a slow control cycle. Stability requires the prediction of the biped movement over the entire step time interval. For the control algorithm to be successful, the controller must have an accurate model of the biped dynamics.

Dynamic walking and balance has an additional property. The infrequent control cycle does not permit arbitrary velocity trajectories during each step. However, the correct passive foot positioning can choose a desirable trajectory. The velocity (and thus the stability) of a biped can be achieved, on average, by choosing a desirable body trajectory. The average control of body velocity is termed dynamic balance. The main advantage of dynamic walking is that the maximum velocity is determined by the maximum leg length and minimum step time interval. Dynamic walking velocity is, in general, larger than static walking velocity.

Dynamic and static walking provide for complimentary advantages and disadvantages. It is desirable for a biped to be able to do both. The proposed walking algorithm implements both dynamic and static walking with the same algorithm.
Algorithm Objective

A successful walking algorithm must be capable of keeping the biped robot upright. Falling over provides uncontrolled movement and is undesirable. A minimum requirement for a walking controller is to prevent the biped from tipping over.

To allow the biped to move to any desired location, the body velocity (and position) must be controlled. It is also desirable for the biped to start moving and stop moving as needed. The ability to maintain stability under the influence of external forces or unknown dynamics is also very desirable.

The height of the biped body is an additional control objective. Various biped height trajectories can be created. The proposed algorithm is designed to provide for constant height. A constant height provides a stable platform for manipulators or instrumentation such as navigation sensors.

A constant height may also help reduce energy requirements for walking. Many bipeds may have mechanics that do not possess conservative leg forces. Unnecessary vertical movement consumes leg actuator energy.

The proposed walking algorithm's objective is to provide for constant body height and controlled horizontal body velocities. To allow high speed body velocities, dynamic walking is provided. For fine control of low velocity (and position), static walking is also provided. The proposed walking algorithm provides both static and dynamic walking.

54
behavior by the coordinated use of both ankle torques and next-step positions.

**Algorithm Assumptions**

To simplify the biped dynamics, several assumptions are made. The walking controller is only concerned with body translations. Body orientation is not controlled by the step control algorithm and is assumed to be fixed. The fixed orientation simplifies leg kinematics. Leg positions do not have to take into account varying body rotations. This assumption is equivalent to assuming that the body's moment of inertia is very large.

The legs are assumed to have negligible mass when compared to the body. Any leg movements have little effect on body motion. The biped can be considered to be a point mass, even though a biped is not a single rigid mass. One of the control objectives is constant height. The walking algorithm assumes that this objective is approximately met. The dynamic model for the biped assumes the net vertical acceleration is zero.

To simplify the computation of the body dynamics it is assumed that one and only one foot is in contact with the ground at any time. The transition from one leg to the other leg during a step is assumed to be instantaneous.

The walking algorithm assumes that the actuator forces can be instantaneously varied. This assumption requires that
the force actuators (motors) have a bandwidth that is much larger than the frequency of the body dynamics.

Biped Sensors

The biped structure has force sensors in the bottom of each foot (on the sole). These sensors are used to determine when a foot contacts the floor. The force sensors are also used to determine ankle torque. One possible force sensor arrangement is eight force sensors, one for each corner of each foot. The eight force sensors can be used to provide for the ankle torque measurement on each foot.

The passive foot height above the floor (altitude) is controlled during each step. The altitude is used to determine when the foot is to come in contact with the floor. The foot altitude can be determined by either the biped kinematics or by direct measurement. Foot altitude sensors may be added to allow for direct measurement. Direct measurement of foot altitude helps minimize the effects of kinematic inaccuracies. The foot altitude sensors should also allow for moderated ground level changes. The foot altitude sensor can be considered a limited navigation sensor for unknown terrain.

To predict body dynamics, the orientation of gravity is required. The gravity direction can be deduced from the body dynamics and ankle torques, or by direct measurement. Direct measurement of gravity direction simplifies the algorithm.
Body accelerometers, or tilt meters, can be added to permit direct measurement of gravity direction.

**Inverted Pendulum**

A constant height inverted pendulum is the basic model for the body dynamics used in the biped walking algorithm. For that reason, the dynamic model for an inverted pendulum is reviewed here.

The constant height inverted pendulum is a mass supported by an extending link from a frictionless position constraint (hinge). This can be seen in Figure 4.3. This figure is a two dimensional inverted pendulum. The two dimensional results can be generalized into three dimensions easily. In Euclidean space, Newton’s laws of motion for a point mass are decoupled.

![Diagram of Inverted Pendulum](image)

**Figure 4.3 Inverted Pendulum**

57
The length of the link is varied such that the height of the pendulum is constant for any position. The forces on the mass can be summed in the vertical (y) and horizontal (x) directions. The acceleration in the vertical direction is zero due to the constant height. The vertical component of the link force is equal in magnitude to the force on the mass caused by gravity. From the geometry of the inverted pendulum, the horizontal force thus can be calculated (see Equation 4.1).

\[ \sum F_x = F \sin(\theta) = mx \]
\[ \sum F_y = F \cos(\theta) - mg = m\ddot{y} = 0 \]
\[ \frac{F \sin(\theta)}{F \cos(\theta)} = \frac{m\ddot{x}}{mg} \rightarrow \tan(\theta) = \frac{X}{h} \]
\[ \ddot{x} - \frac{g}{h} x = 0 \]

By equating the horizontal force to the horizontal acceleration, a second order differential equation can be derived. Unlike a constant length pendulum, the dynamics of a constant height pendulum are described by a linear equation, which can be easily solved. The solution is shown in equation 4.2. The state (position and velocity) of the mass at any time in the future can be calculated from the present state by the hyperbolic equations shown. This result is the basis for the next step calculation algorithm.
\[ X(t) = X_0 \cosh(\alpha(t-t_0)) + \frac{\dot{X}_0}{\alpha} \sinh(\alpha(t-t_0)) \]

\[ \dot{X}(t) = \dot{X}_0 \cosh(\alpha(t-t_0)) + X_0 \alpha \sinh(\alpha(t-t_0)) \]

Where:

- \( \alpha \) is \( \sqrt{g/h} \).
- \( t_0 \) is the initial time.
- \( t \) is time.

**Walking Algorithm**

The walking algorithm provides for the coordinated positioning of both legs. The process of bipedal walking involves the transfer of weight back and forth between each of the legs. Each leg has a support phase and a movement phase. The leg that is supporting the weight is the active leg. The moving leg is the passive leg. One leg is in ground contact at all times. Each of the legs are used in a complementary manner: one is passive and the other is active.
Let us consider the following example of a step. The biped is moving with an initial velocity. The direction of the velocity is in the forward direction (see Figure 4.4). A step is taken by positioning the active leg in front of the body as it moves. When the body moves, the leg length is adjusted to provide for a constant body height. The body then slows as the reaction forces though the leg push against the body. During the first part of the step, the leg actuators absorb part of the body's kinetic energy. As the body passes over the active leg's foot, the active leg's force pushes the body faster. The kinetic energy of the body is now increased by the active leg. At the end of each step, the final step
velocity is reached. There is a relationship between the active leg's initial position, the initial body velocity, and the final body velocity. This relationship forms a discrete equation that can be used to control walking.

The passive leg should position the passive leg's foot to the correct position by the end of the current step. This new position becomes the new active leg position for the next step. A physically realizable trajectory requires finite time to move the passive foot from its old position to its new position. While the passive leg is moving along its trajectory, the biped's body will also be moving. The terminal position of the passive leg is determined by the position of the body at the end of each step. It is then necessary to predict the body position at the step end. This prediction is used to determine the passive leg trajectory.

The step duration for the biped walking algorithm is a constant. To provide for the highest possible walking velocity, best velocity control, and best disturbance rejection, the walking algorithm should use the shortest step interval that the mechanics will allow. The shortest step interval is limited by the maximum hip torques and the requirement that the feet are in compression. If the legs accelerate too quickly, the leg momentum may pull the feet from contact with the ground. At least one foot must be in ground contact for correct attitude and position control.

61
Active Leg Control

The next step position is derived from the hyperbolic equation for constant height inverted pendulum velocity as shown in Equation 4.2. The Equation 4.2 is the solution for the unforced second order inverted pendulum differential equation. For next step control, the position, \( x \), in Equation 4.2 is the input. The use of a state (position) as an input reduces the order to a first order system (bottom row of Equation 4.2). By knowing the desired body velocity, the required foot position can be calculated from the body's current state (Equation 4.3).

\[
X_{t+1} = \frac{\dot{X}_{t+1} - \dot{X}_t \cosh \alpha T_s}{\alpha \sinh \alpha T_s}
\]

4.3

Where:
\( X_{t+1} \) is the next step position.
\( T_s \) is the walking time step interval.
\( \dot{X}_{t+1} \) is the desired end-of-step velocity.
\( \dot{X}_t \) is the last end-of-step velocity.

Equation 4.3 is a proportional feedback control law. The new step position is the scaled velocity error.

The foot position given by the next step equation, Equation 4.3, relates the body position to the center of force in the foot. This foot position is called the effective foot position. This location is the result of both the physical leg position and the ankle torques.

The ankle torque and active leg position are determined by the following. If the effective next foot location is
inside the current foot sole, the ankle torque is used totally to provide the next step. Or, if the effective next foot location is outside the current foot sole, the ankle torque is used to provide as much of the effective next step as possible. The remaining portion of the step is provided by the next active foot physical location.

When only ankle torque is used, the next torque adjustment can be made at the next control cycle, not after a step interval. This high speed feedback provides for continuous like control of stationary balance. The ankle torques automatically permit the biped to start and stop as quickly as possible.

Passive Leg Control

The passive leg trajectory is modeled after a sinusoidal pendulum. A pendulum is a mechanical oscillator that requires a minimal energy input to overcome frictional losses. This is an ideal passive leg. To minimize the energy required in walking, the passive leg trajectory should be the natural movement of the passive leg as it moves under the force of gravity. The average moment of inertia of the passive leg then sets the ideal step time interval. The passive leg trajectory chosen is a sinusoidal trajectory (in inertial space, not body relative). The sinusoidal trajectory chosen can be seen in Equations 4.4.
\[
X_p(t) = \frac{1}{2} (X_{p1} + X_{p1+1}) + \frac{1}{2} (X_{p1} - X_{p1+1}) \cos(\frac{\pi}{T_s} t) \\
Y_p(t) = h_s (1 - \cos(2\pi \frac{t}{T_s}))
\]

Where:

- \(X_p\) is the passive leg X axis position.
- \(X_{p1}\) is the initial passive leg X axis position.
- \(X_{p1+1}\) is the final passive leg X axis position.
- \(Y_p\) is the passive leg Y axis position.
- \(h_s\) is the step height.

To simplify the passive leg trajectory calculation, several assumptions are made. These assumptions only need to be valid for minimization of energy, they are not required for stable walking. The assumptions are: The velocity of the body is approximately constant over each step. The moment of inertia of the passive leg around the hip is constant.

The passive leg trajectory is designed to move sinusoidally from the last leg position to the next leg position (see Equation 4.4). These positions are relative to the ground and not the body. The horizontal position changes correspond to 180 degrees of a cosine function. The vertical changes correspond to 360 degrees of a cosine function. These trigonometry functions provide for zero passive leg velocities at the terminal points in the trajectory. Zero velocity helps reduce the foot impact on the ground during each step.
End of Step State Prediction

The next step equation, Equation 4.3, is calculated at the end of each step cycle. The algorithm is improved if the next step foot position is continuously calculated at each control cycle. To provide for this feature, desired foot position is determined by first predicting the biped state at the end of the current step cycle and then calculating the next active foot position from Equation 4.3. The next state prediction is solved by applying Equation 4.2. The state transition matrix is shown in Equation 4.5.

\[
\begin{bmatrix}
X_{i+1} \\
\dot{X}_{i+1}
\end{bmatrix} =
\begin{bmatrix}
\cosh\alpha(T_s - t) & \frac{1}{\alpha}\sinh\alpha(T_s - t) \\
\frac{1}{\alpha}\sinh\alpha(T_s - t) & \cosh\alpha(T_s - t)
\end{bmatrix}
\begin{bmatrix}
X_i \\
\dot{X}_i
\end{bmatrix}
\]  

4.5

Where:  
$t$ is the current time for each step.  
$T_s$ is the walking step time.

Next Step Leg Momentum Coupling

The next step equation, Equation 4.3, assumes that the leg momentum has a minimal effect on the body velocity. This assumption may not be valid. If the leg masses and the passive leg accelerations are large enough, the passive leg momentum changes will affect the body velocity. The body velocity effect can cause an instability in the next step algorithm. When the passive leg accelerates forward, the body velocity slows. The lower body velocity retards the next leg position which, in turn, lowers the passive leg acceleration.
This feedback loop can be unstable. To stabilize this system, the leg momentum change and the body velocity need to be decoupled. Because of the low bandwidth requirements for the next step prediction, the decoupling can be accomplished by bandlimiting the next step position.

The bandlimiting is done with a first order filter. The time constant of the lowpass filter is empirically chosen to provide the desired trade-off between stability and body velocity error. For the bipeds that are shown in the simulation section, the time constant is chosen to be about 20 percent of the step interval time.

Biped Height Control

Biped height control is done by predicting the new body position for the next control cycle. From this prediction, the required leg length is then adjusted for the new body position (see Equation 4.6). The body height control is an open loop controller based on the known biped kinematics. Due to the assumed lack of absolute body height sensors, closed loop control does not make any sense in this application. Body height control is accomplished by feeding the predicted feet positions to the leg position control system. Leg position control is a feedback controller based on joint ankles and leg extensions. Leg position control is assumed by the walking algorithm and is not considered here.
\[ l(t + \delta t) = \sqrt{h_s^2 + (x(t) + \dot{x}(t) \cdot \delta t)^2} \]  

Where:

- \( l \) is the active leg length.
- \( x \) is the active leg x axis position.
- \( \dot{x} \) is the body velocity.
- \( \delta t \) is the controller time increment.
- \( t \) is time.
- \( h_s \) is the step height.

**Biped Attitude Control**

The attitude of the biped's body is controlled by the active leg hip angle. The body should be continuously adjusted to maintain a constant orientation to level ground. This control can assume that the ground is level and can use biped kinematics to keep the body orientation constant with respect to the active foot orientation. Alternatively, this control can be achieved by keeping the body orientation parallel to gravity by a tilt sensor or accelerometer. A possible control structure is shown in Equation 4.7.

\[ \theta_{con}(t + \delta t) = (1 - e^{-\tau/\tau_s})(\theta_{ref}(t) - \theta(t)) + e^{-\tau/\tau_s}\theta_{con}(t) \]  

Where:

- \( \theta_{ref} \) is the body angle reference.
- \( \theta_{con} \) is the body transformation angle.
- \( \tau \) is the body angle filter time constant.
The control loop described by Equation 4.7 both measures and affects position (body angle). Without the first order filter included in Equation 4.7, the control bandwidth is unbounded. This high bandwidth would cause large forces to develop and could lift the active leg. To help reduce this undesired effect and its instability, the first order filter is used to limit the command (transformation angle) acceleration.

**Foot Coordinate Transformation**

The equation for the active and passive foot position are in inertia coordinates (room relative). The biped simulator (and a physical biped) needs to have the foot position in body relative coordinates. Equation 4.8 provides for this transformation and also implements Equation 4.6.

\[
\begin{bmatrix}
    x_{rf} \\
    y_{rf}
\end{bmatrix} =
\begin{bmatrix}
    \cos(\theta_{con}) & \sin(\theta_{con}) \\
    -\sin(\theta_{con}) & \cos(\theta_{con})
\end{bmatrix}
\begin{bmatrix}
    x-v_b t \\
    -h_u
\end{bmatrix}
\]

Equation 4.8

Where:

\( x_{rf} \) is the body relative foot x coordinate.
\( y_{rf} \) is the body relative foot y coordinate.
\( \theta_{con} \) is the body transformation angle.
\( X \) is the desired step position.

**Active Foot Torque Control**

The foot torque control is done by the foot force sensors. Torque is applied to move the net force center to
any desired location in the active foot sole. The feedback is done by measuring the difference between opposing foot sensors and by comparing the result to the desired value. The inertia for this control loop is the moment of inertia of the biped around the active foot.

**Passive Foot Torque Control**

A passive foot is not in ground contact. Feedback is not though the foot force sensors (they read zero). Passive foot control is done by position control. The desired position is to keep the foot parallel to the floor. The inertia for this control loop is just the moment of inertia of the foot and ankle.

**Walking Algorithm Operation**

The proposed walking algorithm computes the feet positions and ankle torques in the following manner. For each controller cycle the procedure shown in Figure 4.5 is implemented.
Model Mismatching Analysis

The effect of modeling errors can be analyzed by viewing the deterministic walking algorithm as a feedback controller. This discrete time first order system is event driven by the end of each step. The next step proportional feedback controller is shown in Equation 4.3. Equations 4.2 can be rewritten to give the biped's body velocity vs. body x axis length transfer function. The biped's transfer function is shown in Equation 4.9.
\[
\frac{\dot{x}}{x} = \frac{a_b \sinh(a_b T_s)}{Z - \cosh(a_b T_s)}
\]  \hspace{1cm} (4.9)

Where:
- \(\dot{x}\) is the body velocity.
- \(x\) is the body x axis projection.
- \(a_b\) is the biped real pendulum frequency.
- \(T_s\) is the walking step time interval.

The \(\cosh()\) is greater than one, which shows that Equation 4.9 is unstable. The instability is expected since an unforced or uncontrolled biped is unstable.

As seen in Equation 4.9, the transfer function for body velocity from body projection is first order. The dynamics of an inverted pendulum is second order. The order reduction that is apparent in Equation 4.9 is due to the end of step sampling which is part of the walking algorithm. The oscillatory envelope of the walking velocity has a first order response. Note, that classical damped second order systems have first order decay envelopes.

Equations 4.3 and 4.9 can be combined to model the closed loop system created by the proportional feedback controller described by Equation 4.3. The inverted pendulum frequencies \(\alpha\) and \(a_b\) are used to model a mismatch between the walking controller (Equation 4.3) and the biped (Equation 4.9). The closed loop system can be seen in Figure 4.6.
The total closed loop transfer function for the controller and biped is shown in Equation 4.10.

\[
\frac{\dot{x}}{x_{\text{ref}}} (Z) = \frac{\alpha_b \sinh(\alpha_b T_s)}{\alpha \sinh(\alpha T_s)} \frac{\alpha_b \sinh(\alpha_b T_s) \cosh(\alpha T_s)}{Z - \cosh(\alpha_b T_s) + \alpha_b \sinh(\alpha_b T_s) \cosh(\alpha T_s)}
\]

4.10

Where:
\(\dot{x}\) is the current body velocity.
\(x_{\text{ref}}\) is the desired body velocity.

If \(\alpha\) and \(\alpha_b\) are close in value, the two constant terms in the denominator cancel and the transfer function becomes \(Z^{-1}\). \(Z^{-1}\) is the ideal result, which attains the desired velocity in a single step. A small mismatch in \(\alpha\) and \(\alpha_b\) causes an exponential step response. This exponential result is shown in the simulation results section. Also, note that the
transfer function has a DC gain error (gain ≠ 1.0) for model mismatches.

**Simulation Results**

Several simulations of the deterministic walking algorithm have been run using the WALK simulator in Chapter 3. The results of these simulations demonstrate the performance of the walking algorithm. The biped used for the simulations has the following characteristics, unless otherwise stated. This biped is called the reference biped.

<table>
<thead>
<tr>
<th>Table 4.1 Reference Biped</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Body Mass</strong></td>
</tr>
<tr>
<td><strong>Thigh Mass</strong></td>
</tr>
<tr>
<td><strong>Shin Mass</strong></td>
</tr>
<tr>
<td><strong>Foot Mass</strong></td>
</tr>
<tr>
<td><strong>Body Inertia</strong></td>
</tr>
<tr>
<td><strong>Thigh Inertia</strong></td>
</tr>
<tr>
<td><strong>Shin Inertia</strong></td>
</tr>
<tr>
<td><strong>Foot Inertia</strong></td>
</tr>
<tr>
<td><strong>Thigh Length</strong></td>
</tr>
<tr>
<td><strong>Shin Length</strong></td>
</tr>
<tr>
<td><strong>Ankle Height</strong></td>
</tr>
</tbody>
</table>
The units used in the following plots are MKS units. Position is in meters. Velocity is in meters per second. Time is in seconds.

The reference biped simulation results can be seen in Figure 4.7. The upper part of Figure 4.7 shows the leg and center of mass trajectories. Note that the center of mass trajectory has a slight variation due to the leg masses. The passive leg trajectories are arcs due to their sinusoidal trajectory. The lower part is a strobe light like plot showing the leg moments.
Figure 4.7 Reference Biped Trajectories
In Figure 4.8 the reference biped is walking at a constant velocity (m/s). The desired velocity is 1.0 m/s. Ideally, the end of step velocity, which is the maximum velocity, should be equal to the desired velocity. The large error is due to three effects. The leg masses cause the center of mass not to be located at the biped body center of mass, and the body center of mass is used for the step calculations. The passive leg momentum changes cause the body momentum to change. The floor impacts are not perfectly soft, so that some of the leg momentum is transferred to the floor.
In Figure 4.9 the step trajectory can be seen (m). For a constant walking velocity the step trajectory should be a constant. The step length should be equal to the average body velocity times the step time interval. The step variation is mainly due to the leg mass coupling.
In Figure 4.10 the effect of leg masses can be seen. The body-to-leg-mass ratios are (top to bottom): 3.33, 6.67, 13.37, and infinity. The lower the body-to-leg-mass ratio, the larger the velocity error (desired velocity is 1.0 m/s).
Figure 4.11 Step Length vs. Body/Leg Mass Ratio

Figure 4.11 shows the step trajectories for the same body-to-leg-mass ratios as in Figure 4.10. The upper trajectory shows the beginning of instability due to the leg momentum coupling problem. The leg mass decoupling filter needs to be increased for the body-to-leg-mass-ratio of 3.33 case.
Figure 4.12 Body Velocity vs. Step Length Errors

In Figure 4.12 the total step length has errors introduced in the biped kinematics. These errors are used to model the non-ideal effects of a physical biped's mechanics. From the top to bottom traces, the errors are: +5 percent, 0 percent, and -5 percent. The errors are in the x axis direction only. The vertical direction should not have placement errors due to the flat floor assumption used in this thesis.
Figure 4.13 Step Length vs. Step Length Errors

Figure 4.13 shows the step length trajectories for the step errors used in Figure 4.12. From the top to bottom traces, the errors are +5%, 0%, and -5%. Some instability can be seen for the -5 percent case.
Figure 4.14 demonstrates the effects of leg length error. In this figure, the leg length used for the velocity estimate calculations are distorted by constant errors of +5 percent, 0 percent and -5 percent (top to bottom traces). As can be seen, the effect of leg length errors is to provide for a constant error in the velocity.
In Figure 4.15, the effect of constant joint angle errors is tested. From the top to bottom traces, the errors are +0.05 radians, 0.0 radians, and -0.05 radians. The angle errors provide for a varying velocity error estimate.
Conclusion

A simple walking algorithm has been presented. Simulation results show that the algorithm provides for successful walking. The simulation results also show that leg masses and geometrical errors can significantly degrade performance. Even with geometrical errors and leg masses, however, stable walking is possible.
Chapter 5

ADAPTIVE WALKING ALGORITHM

Introduction

Chapter 5 describes an algorithm to implement adaptive control of walking for a biped. This algorithm is an extension of the deterministic walking algorithm described in Chapter 4. By using both the deterministic and adaptive algorithms together, initial operation and learned optimization are simultaneously achieved. The adaptive algorithm that is chosen is the Cerebellar Model Articulation Controller (CMAC) (Albus, 1975, Miller, 1987, Miller 1991, An, 1991).

CMAC Walking Algorithm

The CMAC walking algorithm is composed of two parts. The first part is the deterministic walking algorithm used in Chapter 4. The deterministic algorithm's response provides for the bulk of the total response. The CMAC algorithm is used to adjust the deterministic algorithm's output. The adjustment is done to change the step lengths to improve the body velocity tracking performance.

85
**CMAC Overview**

The CMAC is a non-linear adaptive algorithm that maps an integer input vector into an integer output vector. The mapping is determined by a set of training data that is used to program the CMAC. The CMAC used in this chapter is described in An, 1991. The operation of the CMAC is as follows:

The input vector selects a group of weights or values, from the CMAC memory. This selection is done such that any two input vectors that are close to each other share some of the other’s weights.

The selected weights are summed up and the result is a scaler output. Vector outputs are created by repeating this process, as needed, with different memories.

Learning is accomplished by adjusting the selected weights by an increment which is proportional to the difference between the CMAC output and its desired value.

**CMAC Walking Algorithm Structure**

The input vector to the CMAC is comprised of four values. The values are: the desired velocity, the body position, the body velocity, and the time remaining for the current bipedal step. The desired velocity input is used to allow for command feedforward. The position and velocity inputs are used to provide for state feedback. The time-to-go input permits the correct interpretation of the body state and desired velocity.
The desired velocity is compared to the velocity at the end of the step. The time to go distinguishes between the beginning and end of the step.

The adaptive strategy used to program the CMAC is a batch mode method. During each step, the inputs to the CMAC are stored. At the end of that step, the CMAC is adjusted for all of the control inputs used during the preceding step. The CMAC is conditioned such that the total step output is equal for all inputs used during the step. The step output value is also adjusted to provide for the correct velocity at the step end. The constant step length goal is chosen to provide for minimum passive leg accelerations. With a constant step length, the passive leg accelerations are only determined by passive leg trajectory equations. The passive leg accelerations are not affected the changing step length. The final step length value, at ground contact, determines the body velocity for the next step cycle. The optimization of final step length optimizes the body velocity.

The CMAC adjustment method is shown in Equation 5.1. Equation 5.1 is for the $i^{th}$ simulator time step and is repeated for all of the time steps at the end of the step.
\[ \text{CMAC}_i = \text{CMAC}_{\text{final}} + (S_{\text{final}} - S_i) + (\dot{X}_{\text{final}} - \dot{X}_i) T_s \]  \hspace{1cm} (5.1)

Where:
- \( \text{CMAC}_i \) is the \( i \)th CMAC output.
- \( \text{CMAC}_{\text{final}} \) is the final CMAC output.
- \( S_i \) is the \( i \)th deterministic output.
- \( S_{\text{final}} \) is the final deterministic output.
- \( \dot{X}_{\text{final}} \) is the final body velocity.
- \( \dot{X}_i \) is the desired body velocity.
- \( T_s \) is the step time interval.

The first term in Equation 5.1 causes the CMAC training to implement an integral action. The next CMAC output is an increment on the last CMAC output, thus forming a discrete integrator. The second term in Equation 5.1 adjusts the CMAC to have a constant output, or step length, for all of the input values used during the current step. The last term in Equation 5.1 adjusts the step length to provide for the correct body velocity. The \( T_s \) scale factor is used to scale the step length increment to have the same step length to body velocity ratio as steady state walking. The average velocity is equal to the step length divided by the step time interval.

**Walking Algorithm Operation**

The adaptive walking algorithm computes the foot positions and ankle torques in the following manner. For each control cycle the procedure shown in Figure 5.1 is implemented.
At the end of each step, the CMAC is trained using the recorded data of all states achieved during the last step.

Simulation Results

Several simulations of the adaptive walking algorithm have been run using the WALK simulator in Chapter 3. The biped geometry used is the reference biped described in Chapter 4.
Figure 5.2 Body Velocity for CMAC Learning Transient

In Figure 5.2, constant velocity walking is shown. The upper trace shows the biped velocity without the aid of the CMAC. The lower trace shows the adaptive algorithm. The CMAC is initialized to zero and is trained during this simulation. The CMAC corrects for the leg mass induced velocity error (desired velocity is 1.0 m/s). The body-mass-to-leg-mass-ratio is 3.33.
Figure 5.3 Step Length for CMAC Learning Transient

Figure 5.3 shows the step length that corresponds to the velocities in Figure 5.2. The lower trace is for the adaptive CMAC algorithm. The CMAC has made the step prediction both more constant and shorter than the deterministic walking algorithm alone. The step prediction shown in Figure 5.3 is correct for the body velocity of 1.0 m/s and a step rate of 500 ms.
In Figure 5.4, the biped has been trained to follow a pulsed desired velocity. The upper trace is without the CMAC being used. The middle velocity trace uses the CMAC but without training on the desired vs. body velocity difference. Thus, the middle trace does not correct for body velocity errors. The lower trace uses the complete training of Equation 5.1. The complete training case shows the best command following performance.
In Figure 5.5, balancing is demonstrated. In the balancing simulation, the desired velocity is steps periodically from 0.0 to 1.0 m/s. When the body is at zero velocity, the biped robot should balance statically. The upper trace is the body velocity without the CMAC. The lower trace is the combined controller body velocity after 20 trials. Without the CMAC, the body velocity is disturbed by the leg masses and never stops. With the CMAC, the body stops moving.
In Figure 5.6, CMAC only walking is demonstrated. In this simulation, the deterministic algorithm is disabled. The desired velocity is a constant 1.0 m/s. The simulation is repeated 120 trials. For about the first 100 trials, the biped falls down. After 100 tries, the biped started to walk for more than one or two steps. At about 120 tries, the CMAC only algorithm demonstrated about the same performance as the combined CMAC and deterministic algorithms. The CMAC only control algorithm is similar to the algorithms described in Miller, 1991.
In Figure 5.7, the velocity is shown for the CMAC only walking algorithm case.
In Figure 5.8, the step length prediction is shown for the CMAC only case. The step output has more noise than the combined algorithms case.
Conclusion

An adaptive walking algorithm was presented. Simulation results show that the algorithm provides for successful and robust walking. Even with the deterministic part disabled, the biped can still learn to walk. The CMAC controller keeps the step length constant over each step. This helps limit the effects of leg masses. The adaptive control also provides for the better body velocity error performance (both statically and transiently) when compared to the deterministic controller alone.
Chapter 6

SUMMARY

This chapter summarizes the results of the three parts of this thesis and suggests some areas of future study. The three parts of this thesis are: a two dimensional simulator, a deterministic walking algorithm, and an adaptive walking algorithm.

Two Dimensional Simulator

The simulator is based on a discretized Lagrangian dynamics model for a seven-link biped. The foot/floor interactions are modeled by point inequality constraints on each foot. The WALK simulator is designed to provide for simplified walking algorithm evaluation. By using joint angle position inputs, the problems of joint angle controllers are deferred. The foot placement algorithm is the only requirement for the WALK simulator.

The WALK simulator was used to test and debug several walking algorithm concepts. Robustness of the different walking algorithms is readily tested by changing the bipedal model parameters.

98
**Deterministic Walking Algorithm**

A simple inverted pendulum walking algorithm is described. This algorithm demonstrates stable walking in the presence of modeling errors. The deterministic algorithm allows large errors when the seven-link biped is modeled poorly by a single mass. Leg momentum variations will cause changes in the body velocity. These body velocity changes can cause instability in the next step prediction. The effects of leg masses are mitigated by using a lowpass filter to decouple the step prediction control from the passive leg trajectory. The lowpass filter time constant is empirically adjusted so that the movement of the non-supporting leg will not cause instability.

The walking algorithm demonstrated dynamic walking, static walking, and static balancing. The ankle torque control methods allow for a smooth transition from dynamic walking to static balancing.

**Adaptive Walking Algorithm**

A CMAC based adaptive controller is used to aid the invented pendulum controller. The combined controller generates the next step prediction by summing the results of a deterministic controller with the results of a CMAC controller. The combined controller algorithm allows for large modeling errors and still provides for accurate control. The deterministic part of the combined adaptive algorithm provides
for fast learning and initial stability. The CMAC portion of the combined controller generates incremental adjustments to compensate for modeling errors.

The CMAC controller can be used alone, without the deterministic controller. The CMAC adaptive controller then provides for all of the next step prediction. When the CMAC controller is used alone, the biped’s walking is initially unstable. The CMAC controller lets the biped fall down until the next step prediction has converged close to the required value. Due to the initial poor walking performance, the CMAC only controller has slow convergence.

**Practical Biped Walkers**

Physical biped walkers have a number of additional considerations that are not covered in this thesis. One of the most important issues is the passive (non-supporting) leg trajectory. For good walking performance, the passive leg must not impact the ground with any remaining velocity. If there is any remaining velocity, a large force will be developed when the foot touches the ground. This force will disturb the biped’s momentum and cause next step prediction errors. This problem is familiar to most readers as toe stubbing and tripping.

Kinematic errors, ground compliance, and ground slope variations will make controlling the passive leg height from the supporting (active) leg problematic. To avoid passive leg
height errors, either adaptive control or direct foot height measurement is needed.

Body orientation control is critical, due to the large body inertia. Abrupt changes in the body angle may lift the active foot and cause a fall. The determination of the body angle kinematically from the active foot is error prone. Gyros or accelerometers (tilt indicators) should be considered as a better body orientation reference.

The projection of the vector from the constrained foot to the center of mass in the plane that is normal to gravity is one of the main inputs for next step prediction. Direct measurement is difficult due to kinematic errors, ground compliance, and ground slope. A better way to determine the body’s projection is indirectly from the body’s acceleration. The body’s acceleration can be measured by an accelerometer. The body’s projection (x, y axis position) is found from the inverted pendulum equation in Chapter 4.

**Future Directions**

The WALK simulator has joint angles as inputs. This input choice assumes that the joint angle controllers are perfect. The effects of not making this assumption should be tested. There are two ways to test the effects of the position controller dynamics.

The WALK simulator can be rewritten to have joint torques as inputs instead of positions. The walking algorithm will
then have to provide for the feedback control of joint angles from joint torques. This approach models the joint angle dynamics directly and is the most accurate method.

The second method is to filter the joint angle positions with a filter that approximates the joint angle controller dynamics. This method is much simpler than the joint controller method. The WALK simulator does not have to be rewritten and the joint angle controller does not have to be designed. The leg position filter method does not correctly model the dynamic effects of floor impacts.

Balancing is more accurately achieved by position feedback than velocity feedback. Small errors will cause drifts in body velocity measurements. Proportional velocity controllers, such as the deterministic walking controller described in Chapter 4, have steady state errors in velocity control. Position feedback balancing will solve this problem. When balancing is needed, the ankle torque(s) should be controlled by position feedback, not velocity feedback.

The passive leg and body trajectories could be optimized for better walking performance. The body height can be controlled for optimum walking efficiency. This result will be dependent on the energy recovery ability of the leg actuators. The passive leg trajectory was chosen ad hoc to be sinusoidal in inertia space. Better efficiency and lower leg forces may be achieved with other trajectories.
The three dimensional simulator should be written. This simulator would allow testing of the coupling between the sagittal and frontal planes. This may be needed to accurately model high speed behavior such as running. The algorithmic basis for the three dimensional simulator is included in Appendix A.
REFERENCES


Herman, Grillner, Stein, and Stuart, Advances in Behavioral Biology, Vol. 18.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

Craik, Herman, and Finley, Human Solutions for Locomotion: Interlimb Coordination.  
Herman, Grillner, Stein, and Stuart, Advances in Behavioral Biology, Vol. 18.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

William J. Davis, Organization Concepts in the Central Motor Networks of Invertebrates.  
Herman, Grillner, Stein, and Stuart, Advances in Behavioral Biology, Vol. 18.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

A. Frank D’Souza and Vijay K. Garg, Advanced Dynamics  
ISBN 0-13-011312-3

Herman, Grillner, Stein, and Stuart, Advances in Behavioral Biology, Vol. 18.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

Edgerton, Grillner, Sjostrom, and Zangger, Central Generation of Locomotion in Vertebrates.  
Herman, Grillner, Stein, and Stuart, Advances in Behavioral Biology, Vol. 18.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

Roger M. Enoka, The Simple Joint System, Part II  
Neuromechanical Basis of Kinesiology.  
Champaign: Human Kinetics Publishers Inc. 1988  
ISBN 0 87322 179 6

R. Flechter, Practical Methods of Optimization Vol. 2  
Constrained Optimization  
New York: John Wiley & Sons 1981  


105

Herman, Wirta, S. Bampton, and F. R. Finley, Human Solutions for Locomotion: Single Limb Analysis.
Herman, Grillner, Stein, and Stuart, Advances in Behavioral Biology, Vol 18.
New York: Plenum Press 1976
ISBN 0 306 37918 X

A. Iggo, Cutaneous Sensory Mechanisms. Chapter 17
Barlow and Mollon, The Senses.
New York: Cambridge University Press 1982
ISBN 0 521 24474 9

S. Kitamura, Y. Kurematsu, Autonomous Motion Planning and Learning Control of a Biped Locomotive Robot, IFAC Congress 1990

E. R. Lewis, Inertial Motion Sensors. Chapter 5.2
Bolis, Keynes, and Maddrell, Comparative Physiology of Sensory Systems.
Cambridge: Cambridge University Press 1984
ISBN 0 521 25002 1


Cambridge Massachusetts: The MIT Press 1981

Christopher Plait, Cellular Bases for Gravistatic Reception by Invertebrates and Vertebrates. Chapter 5.1 Bolis, Keynes, and Maddrell, Comparative Physiology of Sensory Systems.
Cambridge: Cambridge University Press 1984
ISBN 0 521 25002 1
Marc H. Raibert, *Legged Robots That Balance*  
Cambridge Massachusetts: The MIT Press 1986  
ISBN 0-262-18117-7

Herman, Grillner, Stein, and Stuart, *Advances in Behavioral Biology, Vol 18*.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

Herman, Grillner, Stein, and Stuart, *Advances in Behavioral Biology, Vol 18*.  
New York: Plenum Press 1976  
ISBN 0 306 37918 X

Schone, *Orientation to Gravity and Angular Acceleration*.  
*Chapter 3.10, Spatial Orientation*.  
ISBN 0 691 08363 0

Schone, *Orientation by Touch and Feel*.  *Chapter 3.9 Spatial Orientation*.  
ISBN 0 691 08363 0

Schone, *Orientation as a System of Interacting Components*.  
*Chapter 2.4 Spatial Orientation*.  
ISBN 0 691 08363 0

Schone, *Orientation Without External Directing Cues*.  
*Chapter 11, Spatial Orientation*.  
ISBN 0 691 08363 0

Jon M. Smith, *Mathematical Modeling and Digital Simulation for Engineers and Scientists*  
New York: John Wiley & Sons 1977  
ISBN 0-471-80344-8

Keith R. Symon, *Mechanics*  
London England: Addison-Wesley 1961  
Library of Congress Catalog Card No. 60-5164
D. J. Todd, *Walking Machines, an Introduction to Legged robots*
New York: Chapman and Hill 1985

David L. Tomko, Coding the Movements and Position of the Head in Space. Chapter 13
Lund, *Sensory Processing in the Mammalian Brain.*
Oxford: Oxford University Press 1989
ISBN 0 19 504554 8


Mary C. Wetzel and D. G. Stuart, Activation and Coordination of Vertebrate Locomotion. Chapter 5
London: Chapman and Hall Ltd. 1977
ISBN 0 470 99185 2


Yuan F. Zheng, Fred R. Sias, Jr., Design and Motion Control of Practical Biped Robots, *IASTED International Symposium on Robotics and Automation, 1987*
Appendix A

THREE DIMENSIONAL DYNAMIC SIMULATOR

Introduction

This appendix describes the three dimensional extension to the WALK simulator that is described in Chapter 3. The basic method used for the WALK simulator can be used for the three dimensional case. Only the kinematics and the Lagrangian need to be modified for three dimensions. The ease in which the three dimensional case can be solved is due, in a large part, to the generality of Lagrangian dynamics. The derivation in this chapter is similar to derivations in Paul, 1981.

Three Dimensional Simulator

The three dimensional biped robot problem has six unknown positional degrees of freedom. These can be the x, y, and z axis in a Cartesian space and three rotations about the axis. These rotations can be chosen in any order, but must be applied in the same order for each use. The rotations do not commute. The forces of constraint for the two legs are six additional unknowns, one unknown for each translation axis for each leg.
**Kinematics**

One method to solve for a location on a complex geometry in three dimensions is to use a simple coordinate system to measure the desired position and transform it to the final coordinate system that is needed. For example, it is possible to pick a Cartesian coordinate system with its origin at the center of mass on a link in a bipedal structure. The desired position of the center of mass, in the room coordinate system, can be solved by a series of translations and rotations to follow the connected biped links back to the room coordinate system.

These translations and rotations may be formulated in matrix form. The multiplication of these matrices transforms a position vector in one coordinate system to another coordinate system. These matrices are called homogeneous transforms.

Homogeneous transforms work just like any three by three coordinate rotation matrices with the addition of a row to provide for linear translations. The position vectors, then, have four components, three components for position and one constant component (which is normally equal to 1.00). The homogeneous transform equations are given in Equations A.1 through A.5.

110
\[
S = \begin{bmatrix} x \\ y \\ z \\ 1.0 \end{bmatrix}
\]

A.1

\[
Rot(x, \theta) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & \cos(\theta) & -\sin(\theta) & 0 \\
0 & \sin(\theta) & \cos(\theta) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A.2

\[
Rot(y, \phi) = \begin{bmatrix}
\cos(\phi) & 0 & \sin(\phi) & 0 \\
0 & 1 & 0 & 0 \\
-\sin(\phi) & 0 & \cos(\phi) & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A.3

\[
Rot(z, \psi) = \begin{bmatrix}
\cos(\psi) & -\sin(\psi) & 0 & 0 \\
\sin(\psi) & \cos(\psi) & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A.4

\[
Trans(a, b, c) = \begin{bmatrix}
1 & 0 & 0 & a \\
0 & 1 & 0 & b \\
0 & 0 & 1 & c \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

A.5

Where:

- \(S\) is the form of a position vector with coordinates \(x\), \(y\), and \(z\).
- \(Rot(x, \theta)\) is the \(x\) axis rotation matrix.
- \(Rot(y, \phi)\) is the \(y\) axis rotation matrix.
- \(Rot(z, \psi)\) is the \(z\) axis rotation matrix.
- \(Trans(a, b, c)\) is the translations matrix, the values \(a\), \(b\), and \(c\) are the \(x\), \(y\), and \(z\) translations, respectively.
The room coordinate position of the center of mass of any biped link or foot constraint can be computed as the product of the correct matrices. The room coordinate position of any point on the bipedal structure can be expressed as Equation A.6. Note, that the j index on \( T_{ij} \) can cause the transformation matrices to be different for each link.

\[
S_j = \prod_j T_{ij} U
\]

\[
\begin{bmatrix}
0 \\
0 \\
U_d \\
1
\end{bmatrix}
\]

Where:

\( T_{ij} \) is either a rotation or translation matrix.

Equation A.6 provides for the position of any link in the biped. The acceleration and velocity can be computed by the definition of the total derivative. This is done for Equations 3.10 through 3.12. The general form is shown in Equations A.7 and A.8.

\[
\dot{S}_j = \sum_i \frac{\partial S_i}{\partial v_i} \dot{v}_i
\]

\[
\ddot{S}_j = \sum_i \frac{\partial S_i}{\partial v_i} \ddot{v}_i + \frac{\partial^2 S_i}{\partial v_i^2} \dot{v}_i^2
\]

Where:

\( S_j \) is the position of the \( j^{th} \) link.

\( \dot{v}_i \) is the \( i^{th} \) degree of freedom.

112
The $v_i$ has two types of components $v=[q_i \ldots q_i, u_1 \ldots u_j]$. The $q$'s are the unknown position and orientation coordinates. The $u$'s are the known joint angles that are inputs to the simulator.

**Automated Partial Derivatives Computation**

The partial derivatives of the link position with respect to joint angles and translational position can be calculated by using the chain rule. The first and second partial derivatives are shown in Equations A.9 through A.11. The Equations A.9 through A.11 use the fact that each degree of freedom, $v_i$, occurs only once in each equation.

\[
\frac{\partial S_i}{\partial v_i} = \prod_{k=1}^{i-1} T_{kj} \frac{\partial T_{ij}}{\partial v_i} \prod_{i+1}^{n} T_{lj} \tag{A.9}
\]

\[
\frac{\partial^2 S_i}{\partial v_i^2} = \prod_{k=1}^{i-1} T_{kj} \frac{\partial^2 T_{ij}}{\partial v_i^2} \prod_{i+1}^{n} T_{lj} \tag{A.10}
\]

\[
\frac{\partial^2 S_i}{\partial v_i \partial v_p} = \prod_{k=1}^{i-1} T_{kj} \frac{\partial T_{ij}}{\partial v_i} \prod_{i+1}^{p-1} T_{lj} \frac{\partial T_{pj}}{\partial v_p} \prod_{p+1}^{n} T_{lj} \tag{A.11}
\]

Equation A.10 and A.11 are second partial derivatives with respect to the same and different unknowns, respectively. Because the transform matrices are well behaved, the order of differentiation in A.11 does not matter.
Thus, there is no loss of generality in assuming that the i variable transform matrix occurs before the p variable matrix.

**Lagrangian**

The continuous time model of the biped is derived using Lagrangian dynamics. The Lagrangian is the difference between the kinetic and potential energy. The coordinate system used for the biped problem is Cartesian, with the body position, body orientation, and joint angles as degrees of freedom. The Lagrangian, for this case, is shown in Equation A.12.

$$L = \frac{1}{2} \sum_{i=1}^{n} \dot{e}_i^T M_i \dot{e}_i - g^T \sum_{i=1}^{n} M_i e_i$$  

A.12

Where:

- $e_i$ is the position and orientation of the center of mass of the $i^{\text{th}}$ link in Cartesian coordinates.
- $M_i$ is the inertia matrix for the $i^{\text{th}}$ link.
- $g$ is the gravity vector.
- $n$ is the number of links.

The $M_i$ matrix has a simple form when $e_i$ locates the center of mass for each link. The inertia terms for translation and rotation decouple in this equation. This can be seen by noting that $M_i$ has the form shown in Equation A.13.
\[
M_i = \begin{bmatrix}
m_i & 0 & 0 & 0 & 0 & 0 \\
0 & m_i & 0 & 0 & 0 & 0 \\
0 & 0 & m_i & 0 & 0 & 0 \\
0 & 0 & 0 & I_{11} & I_{12} & I_{13} \\
0 & 0 & 0 & I_{12} & I_{22} & I_{23} \\
0 & 0 & 0 & I_{13} & I_{23} & I_{33}
\end{bmatrix}
\]

Where:
- \(m_i\) is the mass of the \(i^{th}\) link.
- \(I_{xxi}\) is the inertia tensor for the \(i^{th}\) link.

**Three Dimensional Continuous Time Model**

The dynamics equation for the continuous time system is shown in Equation 3.27, which is reprinted here as Equation A.14. In Equation A.14, only constraint form is used. For the three dimensional system, it is assumed the floor is modeled by constraints and not external forces.

\[
\frac{d}{dt} \frac{\partial \mathbf{x}}{\partial \mathbf{v}_j} - \frac{\partial L}{\partial \mathbf{v}_j} = \sum_k \lambda_k \frac{\partial c_k}{\partial \mathbf{v}_j}
\]

Where:
- \(v_j\) are the degrees of freedom (known and unknowns).
- \(c_k\) are the floor constraints (if active).
- \(\lambda_k\) are the constraint forces.
- \(e_i\) are the link center of mass vectors.
- \(L\) is the Lagrangian (KE - PE).
- \(M_i\) is the \(i^{th}\) mass tensor.
- \(g\) is gravity.

115
Equation A.14 is solved by first computing the partial derivatives of $L$ with respect to the body position and velocity unknowns. The partial derivatives are shown in Equations A.15 and A.16.

\[
\frac{\partial L}{\partial v_j} = \sum_{i=1}^{n} \sum_{k=1}^{P} \sum_{l=1}^{P} \frac{\partial e_i^T}{\partial v_k} M_i \frac{\partial e_i}{\partial v_j} \dot{q}_k \dot{q}_j - g \sum_{i=1}^{n} M_i \frac{\partial e_i}{\partial v_j} 
\]

\[
\frac{\partial L}{\partial \dot{v}_j} = \sum_{i=1}^{n} \sum_{k=1}^{P} \frac{\partial e_i^T}{\partial v_k} M_i \frac{\partial e_i}{\partial \dot{v}_j} \dot{q}_k 
\]

By the use of the definition of the total time derivative, the final dynamics equation can be derived. The final Equation is A.17.

\[
\frac{d}{dt} \frac{\partial L}{\partial \dot{v}_j} - \frac{\partial L}{\partial v_j} = \\
\sum_{i=1}^{n} \sum_{k=1}^{P} \frac{\partial e_i^T}{\partial v_k} M_i \frac{\partial e_i}{\partial \dot{v}_j} \dot{q}_k + \sum_{k=1}^{P} \sum_{l=1}^{P} \sum_{i=1}^{n} \frac{\partial^2 e_i^T}{\partial v_k \partial v_l} M_i \frac{\partial e_i}{\partial \dot{v}_j} \dot{q}_k \dot{q}_l \\
+ g \sum_{i=1}^{n} M_i \frac{\partial e_i}{\partial \dot{v}_j} + \sum_{k=1}^{r} \lambda_k \frac{\partial c_k}{\partial \dot{v}_j} 
\]

Equation A.17 is solved as a simultaneous system of six equations, one for each position and orientation unknown. The $q_j$ components of $v_j$ are these unknowns. The Equation A.17 can be written in matrix form as Equation A.18.
\[
\sum_{k=1}^{n} A_{jk} \ddot{q}_k + \sum_{k=1}^{m} B_{jk} \ddot{u}_k + \sum_{k=1}^{n} \sum_{l=1}^{n} C_{jkl} \ddot{q}_k \dot{q}_l + \frac{1}{2} \sum_{k=1}^{n} \sum_{l=1}^{n} D_{jkl} \ddot{q}_k \dot{u}_l + \sum_{k=1}^{n} \sum_{l=1}^{m} F_{jkl} \dot{u}_k \dot{u}_l + H_j = \sum_{k=1}^{r} \lambda_k \frac{\partial c_k}{\partial q_j}
\]

Where:

\[u = [u_1, u_2, u_3, \ldots u_m]\]
\[\lambda = [\lambda_1, \lambda_2, \lambda_3, \ldots \lambda_r]\]

\[A_{jk} = \sum_{i=1}^{p} \frac{\partial e_i^T}{\partial q_k} M_i \frac{\partial e_i}{\partial q_j} \]

\[B_{jk} = \sum_{i=1}^{p} \frac{\partial e_i^T}{\partial u_k} M_i \frac{\partial e_i}{\partial u_j} \]

\[C_{jkl} = \sum_{i=1}^{p} \frac{\partial e_i^T}{\partial q_k \partial q_l} M_i \frac{\partial e_i}{\partial q_j} \]

\[D_{jkl} = \sum_{i=1}^{p} \frac{\partial e_i^T}{\partial q_k \partial u_l} M_i \frac{\partial e_i}{\partial q_j} \]

\[F_{jkl} = \sum_{i=1}^{p} \frac{\partial e_i^T}{\partial u_k \partial u_l} M_i \frac{\partial e_i}{\partial q_j} \]

\[H_j = g^T \sum_{i=1}^{p} M_i \frac{\partial e_i}{\partial q_j} \]

\[q_j\] are the n unknown position states.
\[u_j\] are the m known joint angle inputs.
\[\lambda_k\] are the r constraint forces.
\[c_k\] are the r ground constraints.
\[p\] is the number of links.

**Discretized System**

The central difference method can be applied to Equation A.18 in the same manner as in Chapter 3. The Equation A.18 is linear in acceleration and quadratic in velocity. To solve for the next time step either of the following three methods can be used: solve for the non-linear solution, solve for the
linearized solution around the current time, or solve the linear problem formed by using a past, and hence known, value of the velocity in the velocity product terms.

The non-linear method can be done by the using Newton-Raphson iterative techniques. It was experimentally found that for the two dimensional case, with the small time steps required for accurate ground contact modeling, the non-linear velocity terms have little effect. Newton-Raphson requires additional computation time relative to the linearized methods.
The linearized system method and the use of past velocity values are almost equivalent. Due to the shorter computation delays, the past velocity value method shown is used. The matrix Equation A.19 shows this result:

$$P\delta q_{t+\Delta_t} - \mathbf{w}_{t+\Delta_t} = K$$  \hspace{1cm} A.19

Where:

$$P_{jk} = \sum_{k=1}^{n} \frac{A_{jk}}{\Delta_1 \Delta_2}$$

$$w_{jk} = \frac{\partial c_k}{\partial q_j}$$

$$K_j = \sum_{k=1}^{n} \frac{A_{jk}}{\Delta_1 \Delta_2} q_k(t-\Delta_2) - q_k(t) + \sum_{k=1}^{m} \frac{B_{jk}}{\Delta_1 \Delta_2} u_k(t+\Delta_1) - 2u_k(t) + u_k(t-\Delta_2) + \sum_{k=1}^{n} \sum_{l=1}^{n} \frac{C_{jkl}}{\Delta_1^2} (q_k(t) - q_k(t-\Delta_2))(q_l(t) - q_l(t-\Delta_2)) + 2 \sum_{k=1}^{n} \sum_{l=1}^{m} \frac{D_{jkl}}{\Delta_1^2} (q_k(t) - q_k(t-\Delta_2))(u_l(t) - u_l(t-\Delta_2)) + \sum_{k=1}^{m} \sum_{l=1}^{m} \frac{F_{jkl}}{\Delta_1^2} (u_k(t) - u_k(t-\Delta_2))(u_l(t) - u_l(t-\Delta_2)) + H_j$$

Where:

$$\delta q_j$$ are the n unknown position states increments.
$$\delta u_j$$ are the m known joint angle inputs increments.

**Ground Constraints**

As in Chapter 3, the ground constraints are used to limit the feet to being above the ground. These are inequality constraints. The inequality constraints are solved
iteratively by choosing different sets of constraints to be active and others to be inactive. The active constraints are forced to be equality constraints, and the inactive constraints are not used. The set of active constraints is adjusted until both the feet are above the ground and both legs are not in tension.

The linearized constraint equations are formulated by the Taylor series expansion of the nonlinear constraint equations. The equality constraint equations are set up to force the feet to be in ground contact. The constraint equation expansion is shown is Equation A.20

\[
c_k(t+\Delta t)-c_k(t) = \sum_{j=1}^{n} \frac{\partial c_k}{\partial q_j} \delta q_j + \sum_{j=1}^{m} \frac{\partial c_k}{\partial u_j} \delta u_j = \sum_{j=1}^{n} \frac{\partial c_k}{\partial q_j} \delta q_j - \sum_{j=1}^{m} \frac{\partial c_k}{\partial u_j} \delta u_j
\]

**Final Implementation**

The Equations A.19 and A.20 need to be solved simultaneously. This system of \( n + r \) equations can be viewed in matrix form as Equation A.21.

\[
\begin{bmatrix}
P & -\hat{W}^T \\
\hat{W} & 0
\end{bmatrix}
\begin{bmatrix}
\delta q \\
\lambda
\end{bmatrix}
= \begin{bmatrix}
k \\
Z
\end{bmatrix}
\]

As indicated in the section on constraints, the desired set of active constraints needs to be determined (Fletcher, 1981). This can be done by the algorithm used in the two
dimensional simulator described in chapter 3. The method
starts with a feasible set of constraints by using the current
foot position. The correct set is determined by eliminating
the constraints that are associated with legs which are in
tension.

Unlike the two dimensional case, Equation 3.49 through
3.52, the system of equations in A.21 is never over
determined. There are six constraints for two legs. The
biped position and orientation has six degrees of freedom.

To insure that Equation A.21 is non-singular, the
inactive constraints should be replaced by equations that
evaluate the corresponding forces, λ, to zero. Also one of
the legs needs to be set inactive when the feet positions
become too close. If the feet are too close, the constraint
equations become numerically singular.
Appendix B

WALK USERS MANUAL

General Information

Appendix B is the user's manual for the WALK biped robot simulator described in Chapters 3, 4, and 5. The walk simulator is written in Microsoft's QuickC 2.5 programming language. System requirements for the WALK simulator are any IBM PC or compatible personal computer running DOS 3.0 or higher. The computer should have 500KB of free DOS memory, 1MB of extended memory, and a Hercules, CGA, or VGA graphics adapter. If the Hercules graphics adapter is used, Microsoft's MSHERC.COM graphics driver must be installed first.

The WALK simulator uses UNH Robotics Lab's DOS extended memory CMAC. The CMAC TSR must be installed before the WALK simulator is executed. If the CMAC is not installed, the simulator will hang up the computer. WALK is executed by typing WALK at the DOS command line prompt.

When WALK is executed, the ASCII file DEFAULT.INP is read in. This file provides for all of the variable initialization for the WALK program. Without the DEFAULT.INP, the simulator must have all of the variables manually set.
Unpredictable results will happen if the internal variables are not initialized.

The DEFAULT.INP file is a macro file written in the WALK command language. Macro files provide for the same affect as typing in commands at the cmd> prompt.

**WALK Simulator Files**

The WALK simulator is compiled by using Microsoft's QuickC 2.5 compiler. The modules required for the WALK simulator are:

CMD.C DYNAM.C SOLVE.C INPUT.C PLOT.C LCMAC.LIB

The simulator requires the large memory model with 4096 bytes of stack space. The make file for setting up QuickC is WALKQC.MAK. The resulting executable file is WALK.EXE.

The CMD.C module contains the main() routine and the command line parser. SOLVE.C contains the simultaneous equation solver and biped geometry functions. DYNAM.C provides the dynamic simulator. PLOT.C has the graphics functions. INPUT.C has the deterministic and CMAC walking algorithms.

123
Command Reference

The following is a list of the commands that the WALK command line interpreter will accept.

System Commands

datafile filename
   The datafile command opens the output data files for the state data that can be stored for each simulation time step. If the file name is set to "null", the WALK program provides no output file.

transfile filename count
   The transfile command changes the simulation by reading in a macro file during a simulation. The file is read in when the <count> iteration is reached. If the file name is set to "null", the WALK program ignores this command.

do filename <count>
   Read in a macro file <count> number of times. The <count> parameter is optional.

list
   Lists all of the commands.

help
   Displays of help file called HELP.TXT.

display
   Displays all variables

print filename
   Prints all variables to file.

run <optional parameters>
   Starts a simulation. The parameters are:
   off no graphics
   short limited display

chart <filename>
   Executes the charting utility program CHART.EXE. The chart program allows for graphing the information in a data file created by WALK. The optional filename is used to initialize chart to that data file.

strobe datafile
   Displays a data file with "strobe effect".

124
animate datafile <optional parameters>
   Displays a data file output file. The optional parameters are:
       short  limited display
       trace  trace cm, etc
edit <filename> Run MS DOS 5.0 editor, EDIT.EXE.
dos string  Executes a DOS command line
shell      Starts a DOS COMMAND.COM shell
exit       Terminates program
quit       Terminates program
input      Selects simulator input type
       syntax:  string
       manual  Manual foot placement
       jump   Jumping mode
       walk   Deterministic walking mode
find       Reposition biped to just in ground contact
pause      Wait for key to be pressed
echo string Display a string from a macro.

Graphics Commands

body_d    float
       Body graphic depth, distance below the hip.

body_h    float
       Body graphic height, distance above the hip.

body_w    float
       Body graphic width, distance from hip to each side.

plot_scale float
       Number of meters of the y axis.

plot_step integer
       Number of simulation steps per plot update.
Simulator Control Commands

stop_time float Final simulation time.
time_step float Simulator maximum time step.
min_step float Minimum time step.

floor_adjust boolean
Automatic correction for foot penetration of the floor.

   True The biped is repositioned if either foot is below the ground. This is done moving the last three states, thus the simulation algorithm calculates zero acceleration.
   False don’t do the above

forced_active boolean
Active foot calculation method

   True Active feet are determined by the input algorithm
   False Active feet are determined by the foot compression and position relative to the floor.

ground_contact float
Tolerance for a foot to be considered to be in ground contact.

ground_inter boolean
Ground interpolation mode

   True Use ground interpolation
   False Don’t use ground interpolation

internal_accel boolean
Internal acceleration switch

   True Use internal accelerations
   False Don’t use internal accelerations

matrix_check float
Matrix solver test. This value is proportional to the maximum round off error.

matrix_zero float Matrix singularity test
min_foot float
Minimum foot separation for both feet being active
This is used to limit singular constraints.

min_force float Minimum positive force
init_tho float Initial value for orientation
init_xo float Initial value for x axis position
init_yo float Initial value for y axis position
init_vtho float Initial value for orientation velocity
init_vxo float Initial value for x axis velocity
init_vyo float Initial value for y axis velocity
fall_angle float Terminate run if body angle changes by this value
fall_height float Terminate run if body height changes by this value

**Biped Geometry and Parametrics**

body_cm float Body center of mass location
body_mass float Body mass
body_inertia float Body moment of inertia
thigh_cm float Center of mass for thighs
thigh_inertia float Thigh moments of inertia
thigh_length float Thigh lengths
thigh_mass float Thigh mass
shin_cm float Center of mass for shins
shin_inertia float Shin moment of inertia
shin_length float Length of shins

127
shin_mass float 
foot_cm float 
foot_inertia float 
foot_length float 
foot_mass float 
foot_width float 
gravity float 
ground float 

Simulator Input Control

manual_foot_offset float 
manual_lx float 
manual_ly float 
manual_rx float 
manual_ry float 

Input Jumping: (input jumping)

jump_accel float 
jump_begin float 
jump_end float 
jump_foot_offset float 
jump_spread float

Mass of shins
Foot center of mass
Foot moment of inertia
Height of ankles
Mass of a foot
Distance from ankle to either edge of the sole
Value of gravity in MKS units
Y axis position of ground
Manual mode foot offset
Manual mode left leg x axis position
Manual mode left leg y axis position
Manual mode right leg x axis position
Manual mode right leg y axis position
Jumping mode acceleration
Jumping mode initial position
Jumping mode final position
Jumping mode effective foot offset
Jumping mode foot x axis distance from body
**Input Walking:** (input walking)

body_height float Walking algorithm body height

alphamode boolean
   Alpha mode control for deterministic walking,
   Note: alpha = sqrt(gravity/height)

   False     Body height used
   True      Center of mass used

desired_v inputtype
   Walking algorithm desired velocity
   none
   constant value
   sine delay stop minimum maximum frequency
   step delay minimum maximum (period = 2 delay)
   ramp delay stop minimum maximum

walk_height float Walking algorithm passive foot step height

walk_time float Walking algorithm step interval

walk_torque boolean
   Walking algorithm ankle torque switch
   True      Use ankle torques
   False     Don’t use ankle torques

walk_end boolean
   Walking algorithm step end switch
   True      Use walk_time for end of step
   False     Use ground impact of end of step

walk_steptc float Walking algorithm step filter time constant (TC)

walk_bodyref float Walking algorithm body angle servo reference

walk_bodytc float Walking algorithm body angle servo filter TC

walk_foottc float Walking algorithm foot angle servo filter TC
walk_aerror float  Angle error used to corrupt velocity estimate
walk_lerror float  Length error used to corrupt velocity estimate
walk_serror float  Step position gain error used to corrupt foot x axis

Input CMAC Walking: (input walking)
Cmac_alloc  Allocate or re-allocate CMAC memory
Cmac_beta integer  CMAC learning gain in bit shift
Cmac_cell integer  Number of CMAC cells or generalization
Cmac_dealloc  Deallocates CMAC memory
Cmac_forget  Clears CMAC memory

Cmac_inc float
  Gain for end of step CMAC adjustments
  CMAC(n+1) = CMAC(n) - Cmac_inc*(desiredv-vest)*step_time

Cmac_learn boolean
  CMAC learning control switch
  True: learn
  False: hold memory

Cmac_remember boolean
  CMAC output control switch
  True: add in CMAC output to step
  False: don’t use CMAC

Cmac_memory integer  CMAC memory size in 1K blocks
Cmac_usage  Displays the number of 1K blocks used by CMAC
Cmac_read filename  Read CMAC memory from file
Cmac_write filename  Write CMAC memory to file
External Inputs:

**force_th inputtype**

External moment about the tho axis syntax
none
constant value
sine delay stop minimum maximum frequency
step delay minimum maximum (period = 2 delay)
ramp delay stop minimum maximum

**force_x inputtype**

External force along the x axis
none
constant value
sine delay stop minimum maximum frequency
step delay minimum maximum (period = 2 delay)
ramp delay stop minimum maximum

**force_y inputtype**

External force along the y axis
none
constant value
sine delay stop minimum maximum frequency
step delay minimum maximum (period = 2 delay)
ramp delay stop minimum maximum
Default Settings

This is in the DEFAULT.INP file that is used in initialize the WALK simulator. The DEFAULT.INP file is read in each time the simulator is run. The WALK simulator can be re-initialized by the command:

```plaintext
do default.inp
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>alphamode</td>
<td>false</td>
</tr>
<tr>
<td>body_cm</td>
<td>0.0000000e+000</td>
</tr>
<tr>
<td>body_d</td>
<td>1.0000000e-001</td>
</tr>
<tr>
<td>body_h</td>
<td>2.0000000e-001</td>
</tr>
<tr>
<td>body_height</td>
<td>1.0000000e+000</td>
</tr>
<tr>
<td>body_inertia</td>
<td>2.0000000e-001</td>
</tr>
<tr>
<td>body_mass</td>
<td>1.0000000e+001</td>
</tr>
<tr>
<td>body_w</td>
<td>1.0000000e-001</td>
</tr>
<tr>
<td>cmacl_beta</td>
<td>5</td>
</tr>
<tr>
<td>cmacl_cell</td>
<td>32</td>
</tr>
<tr>
<td>cmacl_inc</td>
<td>1.0000000e+000</td>
</tr>
<tr>
<td>cmacl_learn</td>
<td>false</td>
</tr>
<tr>
<td>cmacl_memory</td>
<td>1000</td>
</tr>
<tr>
<td>cmacl_remember</td>
<td>false</td>
</tr>
<tr>
<td>desiredv</td>
<td>constant 1</td>
</tr>
<tr>
<td>fall_angle</td>
<td>2.0000000e-001</td>
</tr>
<tr>
<td>fall_height</td>
<td>2.0000000e-001</td>
</tr>
<tr>
<td>floor_adjust</td>
<td>false</td>
</tr>
<tr>
<td>foot_cm</td>
<td>5.0000000e-002</td>
</tr>
<tr>
<td>foot_inertia</td>
<td>0.0000000e+000</td>
</tr>
<tr>
<td>foot_length</td>
<td>1.0000000e-001</td>
</tr>
<tr>
<td>foot_mass</td>
<td>5.0000000e-001</td>
</tr>
<tr>
<td>foot_width</td>
<td>1.0000000e-001</td>
</tr>
<tr>
<td>force_th</td>
<td>none</td>
</tr>
<tr>
<td>force_x</td>
<td>none</td>
</tr>
<tr>
<td>force_y</td>
<td>none</td>
</tr>
<tr>
<td>forced_active</td>
<td>true</td>
</tr>
<tr>
<td>gravity</td>
<td>9.8100000e+000</td>
</tr>
<tr>
<td>ground</td>
<td>0.0000000e+000</td>
</tr>
<tr>
<td>ground_contact</td>
<td>1.0000000e-003</td>
</tr>
<tr>
<td>ground_inter</td>
<td>false</td>
</tr>
<tr>
<td>init_tho</td>
<td>0.0000000e+000</td>
</tr>
<tr>
<td>init_vtho</td>
<td>0.0000000e+000</td>
</tr>
<tr>
<td>init_vxo</td>
<td>8.0000000e-001</td>
</tr>
<tr>
<td>init_vyo</td>
<td>0.0000000e+000</td>
</tr>
<tr>
<td>init_xo</td>
<td>-2.0000000e+000</td>
</tr>
<tr>
<td>init_yo</td>
<td>1.0000000e+000</td>
</tr>
<tr>
<td>input</td>
<td>walk</td>
</tr>
<tr>
<td>internal_accel</td>
<td>true</td>
</tr>
</tbody>
</table>

132
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>jump_accel</td>
<td>5.000000e+00</td>
</tr>
<tr>
<td>jump_begin</td>
<td>4.000000e-01</td>
</tr>
<tr>
<td>jump_end</td>
<td>1.000000e+00</td>
</tr>
<tr>
<td>jump_foote</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>jump_spread</td>
<td>3.000000e-01</td>
</tr>
<tr>
<td>manual_foote</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>manual_lx</td>
<td>5.000000e-01</td>
</tr>
<tr>
<td>manual_ly</td>
<td>-8.000000e-01</td>
</tr>
<tr>
<td>manual_rx</td>
<td>-5.000000e-01</td>
</tr>
<tr>
<td>manual_ry</td>
<td>-8.000000e-01</td>
</tr>
<tr>
<td>matrix_check</td>
<td>1.000000e-04</td>
</tr>
<tr>
<td>matrix_zero</td>
<td>1.000000e-09</td>
</tr>
<tr>
<td>min_foot</td>
<td>1.000000e-02</td>
</tr>
<tr>
<td>min_force</td>
<td>1.000000e-01</td>
</tr>
<tr>
<td>min_step</td>
<td>1.000000e-02</td>
</tr>
<tr>
<td>plot_scale</td>
<td>5.000000e+00</td>
</tr>
<tr>
<td>plot_step</td>
<td>1</td>
</tr>
<tr>
<td>shin_cm</td>
<td>5.000000e-01</td>
</tr>
<tr>
<td>shin_inertia</td>
<td>4.166666e-02</td>
</tr>
<tr>
<td>shin_length</td>
<td>1.000000e+00</td>
</tr>
<tr>
<td>shin_mass</td>
<td>5.000000e-01</td>
</tr>
<tr>
<td>walk_height</td>
<td>2.000000e-01</td>
</tr>
<tr>
<td>walk_time</td>
<td>5.000000e-01</td>
</tr>
<tr>
<td>walk_torque</td>
<td>false</td>
</tr>
<tr>
<td>stop_time</td>
<td>2.000000e+00</td>
</tr>
<tr>
<td>thigh_cm</td>
<td>5.000000e-01</td>
</tr>
<tr>
<td>thigh_inertia</td>
<td>4.166666e-02</td>
</tr>
<tr>
<td>thigh_length</td>
<td>1.000000e+00</td>
</tr>
<tr>
<td>thigh_mass</td>
<td>5.000000e-01</td>
</tr>
<tr>
<td>time_step</td>
<td>1.000000e-02</td>
</tr>
<tr>
<td>walk_aerror</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>walk_bodyref</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>walk_bodytc</td>
<td>5.000000e-02</td>
</tr>
<tr>
<td>walk_end</td>
<td>false</td>
</tr>
<tr>
<td>walk_foottc</td>
<td>2.000000e-01</td>
</tr>
<tr>
<td>walk_lerror</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>walk_seerror</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>walk_steptc</td>
<td>1.000000e-01</td>
</tr>
<tr>
<td>datafile</td>
<td>null</td>
</tr>
<tr>
<td>tranfile</td>
<td>null</td>
</tr>
</tbody>
</table>

133

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Appendix C

BIOLOGICAL REFERENCE

Introduction

This appendix describes the results of some of the research done in the biological sciences that relates to walking. The purpose for reviewing biological research on walking is to assist in the development of artificial bipeds. The subject area of animal locomotion is large. This paper is restricted to those areas the author believed would be of the greatest utility in simple bipedal systems.

Biological Sensors

This section describes some of the sensing mechanisms that biped animals use in walking. The senses that are discussed are limited to the vestibular system and mechanoreceptors. The vestibular system is a set of organs in the inner ear and is used to measure the motion of the head. Mechanoreceptors are located throughout the body. Mechanoreceptors measure the position, velocity, and forces in various parts of the body.

Vestibular and mechanoreceptor senses are chosen because the author believes that they lend themselves to simple implementation. For example, vision would be extremely useful
to any autonomous system. Vision could be used to determine body orientation, body position, and terrain topography; however, vision requires complex processing to extract information from the image. For this reason, the senses of vision, hearing, smell, etcetera, will not be reviewed.

**Vestibular System**

The vestibular system is used to deduce information about movement and orientation relative to gravity. Vestibular organs function by measuring the inertial forces on a constrained mass. The amplitude of the forces provides an indication of acceleration. The structure of the sense organ determines its directional sensitivity (Benson, 1982, Schone, 1984, Lewis, 1984, Platt, 1984, Tomko, 1989, Roberts, 1976). Unlike most senses, vestibular force sensing has an approximately linear relationship between force and response.

**Semicircular Canals**

The semicircular canals are common to vertebrates. These organs are used to sense angular accelerations and angular velocities.

Semicircular canals consist of toroidal chambers filled with a low viscosity fluid called endolymph (Benson, 1982). The toroidal ducts are obstructed by a compliant diaphragm-like structure called the cupula (see Figures
C.1, C.2). The cupula is innervated to provide output that indicates the cupula displacement.

The toroidal ducts are used to sense the angular movement about its axis of symmetry. Angular acceleration causes the inertia of the endolymph to deflect the cupula; cupula deflection is transduced to provide the sense output.

The dynamics of the semicircular canals form a damped mass-spring system. The inertia of the endolymph is the mass, the viscosity of the endolymph causes fiction, and the restoring force of the cupula is the spring. The cupula restoring force is small when compared to viscous forces. The resulting dynamics of the canals is an overdamped second order system. The system poles are real, with the lower pole having a time constant of about 20 seconds (in man).

Most of the angular accelerations that occur in man have a frequency content that is much faster than the lower time constant of the semicircular canals. The lower time constant functions like a leaky integrator. The integration converts angular acceleration to angular velocity over most of the frequencies of interest.

Typical semicircular canal responses can be seen in Figures C.3 and C.4. Figure C.3 shows a Bode plot of the sensed velocity. Regular and irregular units refer to two types of cupula sensing cells. Figure C.4 shows the step response of the semicircular canals.
There are three semicircular canals positioned orthogonally and each provides a component of the total angular velocity. The complete angular velocity vector is thus determined both in direction and magnitude by each of the semicircular canals.

Statolith and Statocysts Organs

Gravistatic sensing organs are common to many animals from arthropods to vertebrates. Two gravireceptive organs that are well studied are statocysts and statolith (otolith) organs. Each type of organ has a mass and some method to determine the force exerted upon it. The sensing organs consist of a number of hair-like structures called cilia. These cilia respond to shear forces applied normally to the hair axis. This response forms the output of the sense organ (Platt, 1984) and measures the linear acceleration of the organ.

Statocysts have a mass that is free-floating inside a fluid filled chamber (Figure C.5). The chamber walls are lined with the cilia force sensing structures. The magnitude of any acceleration is determined by the reaction force of the chamber wall against the mass. The direction of the acceleration is determined by the location of the mass contact on the cilia.

Statolith organs are similar to statocysts in their structure. The main difference is that the mass of statolith
organs is connected to the ends of the cilia structures by a gelatinous membrane (Figure C.5). The acceleration is determined by the amplitude and the sign of the force measured by the cilia. The spatial orientation of the acceleration vector is determined by the use of multiple statolith organs oriented in different directions. In man, there are two statolith organs: the saccular and utricular maculae (Figure C.1). The saccular macula is sensitive in the direction of gravity (upright posture) and is less sensitive than the utricles. The utricular macula is sensitive in the direction of normal forward walking (sagittal plane).

The typical dynamics of statolith organs can be seen in the Bode plot in Figure C.6. The innervated cilia have varying dynamics. The "regular units" respond to the amplitude of the force. The "irregular units" respond to a change in force amplitude. The irregular units respond to change in acceleration, or jerk. Figure C.7 shows a typical step response of the statolith organs.

**Dipteran Haltere**

The haltere is a small club-shaped wing found on dipterous insects. The haltere functions by vibrating at a rapid rate. Sense organs at the base of the haltere detect the force required to constrain its movement. Because the velocity is large, any change in the direction of the momentum will require a large force (Newton's second law). This effect
makes the haltere a sensitive indicator of angular acceleration.

It is interesting to note that Honeywell Corporation manufactures a vibrating wire gyroscope which operates on the same principle as the haltere. Like the haltere, this gyroscope is both rugged and small. One of the gyroscopes applications is in artillery shells.

Semicircular Canals and Statolith Organ Processing

Information from the semicircular canals and the statolith organs are processed together to provide separate information about gravity orientation and linear acceleration. The semicircular canals sense angular orientation. Thus, the semicircular canals can be used to determine the statolith axis. The statolith organs sense the total linear acceleration. The linear acceleration is the vector sum of both the organ acceleration and gravity. By assuming that the force of gravity is constant, gravity can be decoupled from linear accelerations.

The vestibular processing can be demonstrated by motion sickness. Motion sickness can be caused by inconsistent orientation information provided by the semicircular canals and the statolith organs. A common way to induce motion sickness is to subject the individual to Coriolis accelerations. The angular velocities that man is normally subjected to are such that Coriolis accelerations can be

139
neglected. When Coriolis accelerations are significant, the processing of the semicircular canals indicate the incorrect angular velocity. To induce motion sickness, the head is accelerated about one axis while rotating at a constant velocity about another axis. If the axes are chosen correctly, the perceived angular velocity indicates one orientation, while gravity indicates something else. This results in motion sickness.

Vestibular Functions in Walking

Orientation information is necessary for proper control of walking. In man, this information can be measured directly by the vestibular system, or indirectly by other senses such as vision. Anyone can test this assertion by first spinning around in such a manner that they become dizzy and then by closing their eyes. These two actions temporally remove orientation information. Most individuals find it difficult to walk or even stand under these conditions.

Man normally uses both the vestibular system and vision when walking. The loss of the vestibular system increases the difficulty in balancing while walking on irregular or compliant surfaces. Gravistatic orientation is more difficult to visually access in an irregular environment.

The vestibular system is also used to stabilize visual perception when the head is moving. This is called the vestibular-ocular response. Without the vestibular system, it
is difficult to distinguish between a moving image and a moving head.

**Applications of the Vestibular System in Artificial Biped**

In any artificial biped which is capable of walking, the orientation of gravity is of prime importance. The dynamics of the biped are determined by the relationship between gravity and points of support. Specifically, the acceleration of the biped body is determined by the projection of the center of mass to the points of support. The projection is in a plane that is tangential to gravity (Figure C.8).

The center of mass of the biped is both time variant and load dependent. One method to measure the size of the projection is by measuring the magnitude of the body acceleration in the direction being walked.

The vestibular system can be used to control the posture of the biped body. The position of the legs are determined by the difference in the hip angles; the hip angle determines the body orientation. This angle could be controlled by an artificial vestibular system. The control variable for posture control could be gravity, angular acceleration, or both.

In an artificial biped, linear accelerometers could be coupled with either angular accelerometers or gyroscopes. This system could provide both body posture and center of mass location.
Mechanoreceptors

Mechanoreceptors provide a measure of the mechanical state of the body (Iggo, 1982, Schone, 1984a). These senses can provide: touch, position, velocity, and force information.

Cutaneous Mechanoreceptor Classes

The sense of touch is one of the most basic senses that any organism can use to deduce information about its environment. This section reviews the touch sensors in glabrous skin (the surface of the hands and feet).

All mechanoreceptors provide sensory information about the deflection in the skin (or other organs). The response of a mechanoreceptor is nonlinear, and follows a power law relationship (Iggo, 1982).

There are two general classes of cutaneous mechanoreceptors. One class, called rapidly adapting mechanoreceptors, responds to the velocity of the skin deformation. With a constant deformation, these sensors provide no response. Rapidly adapting sensors have a maximum sensitivity at certain vibration rates. The variability within this class provides for differing peak sensitivities. Rapidly adapting sensors are useful to sense vibration.

The second class of mechanoreceptors is called slowly adapting mechanoreceptors. These sensors respond to static deformations in the skin. Due to the elasticity of the skin,
deformations are proportional to the pressure applied to the skin. Slowly adapting sensors are useful for indicating forces and pressures.

Figure C.9 shows the response of slow and rapid adapting mechanoreceptors to a step input. Pacinian corpuscles are classed as rapidly adapting sensors. C-fibre mechanoreceptors are classed as slowly adapting. Figure C.10 shows the frequency response of various mechanoreceptors.

**Cutaneous Mechanoreceptor Application**

In the author’s research, little information was found on utilization of touch in walking. The lack of information is most likely due to the complexity of walking behavior and the ease of microprobing skin cells. The proposed uses of the touch sense in walking is conjecture; no supporting research is used.

There are many ways in which touch sensing relates to walking. One of the most obvious uses for touch sensing is in the coordination of weight transfer from one leg to the next. During a biped walking gait, the biped’s weight is transferred from one leg to the other. Before weight can be transferred to a leg, the leg must have achieved ground contact. The sense of touch can be used to initiate the weight transfer. To illustrate this utility, consider walking in the dark on irregular terrain. It is possible to use touch to indicate when to take the next step.
Coordinated walking requires careful control of leg trajectories. For example, it is important to land a biped's free (not supporting) leg on the ground with low velocity. "Toe stubbing" or "stomping" is the result of landing a foot with high velocity. Touch can provide an indication of ground contract. This ground contact indication can be used to modify the free leg trajectory so that future steps can be improved. Touch can be used to implement predictive control of leg trajectories.

Ankle torque helps control a biped's walking velocity. Accurate ankle torque measurements are needed for precise control and stability in walking. The touch sensors on the sole of a biped's feet can be used to measure ankle torque. This measurement can be done by subtracting the toe and heel forces. Ankle torque measured in this manner provides for the total torque impressed upon the biped. This includes foot and leg inertial forces.

Leg extension forces are also important to control a biped's velocity. Leg extension forces can be measured by summing the toe and heel forces. This method provides the same advantage as in the case of ankle torque. Touch sensing provides for a measure of the total reaction force. Agile walking requires the understanding of ground conditions. Most individuals walk differently on loose sand and hard roads. Ground compliance can be assessed by transducing the force verses displacement characteristics of ground impacts.
Ground irregularities and slope can also be determined by measuring when, in a leg trajectory the foot impacts the ground. This type of surface detection can compensate for gradual slope changes and small irregularities. This type of surface determination can be observed by walking in the dark.

**Kinesthesia**

Kinesthesia is the sense of an individual’s own movements (Schone, 1984c). Walking requires the precise placement of legs and feet, and the control of muscle forces. The transduction of muscle tension and joint angles are needed for feedback control of leg trajectories. These measurements are done by mechanoreceptors found in the tissues of the legs. There are three classes of mechanoreceptors in the limbs. These receptors are the muscle spindles, tendon receptor, and joint receptors.

**Muscle Systems**

Muscles are specialized cells that convert chemical energy into tension (Enoka, 1988). Skeletal muscles are grouped into antagonistic pairs around each of the skeletal joints. Each muscle group provides the forces necessary to move a joint in one direction. Muscles are classed into three groups: slow-contracting fatigue resistant (type S), fast-contracting fatigue resistant (type FR), and fast-contracting fast-to-fatigue (type FF). These three
muscle classes provide a trade-off between speed, force, and duration. These trade-offs can be seen in Alexander, 1977.

From a design point of view, muscles have several characteristics that are significant. First, muscles and their connective tissue are elastic. This elasticity can be useful in recovering energy used by negative work during locomotion. A consequence of the elasticity is that the body and limb masses now form a distributed mass spring system. To optimize energy recovery, this distributed system should have low frictional losses and, thus, be under damped.

Distributed, under-damped systems can prove to be difficult to control. Constant monitoring and correction are needed for stable and robust locomotion. Nature has addressed this issue by creating a hysteresis in muscle tissue. Muscle hysteresis is illustrated in Figure C.15.

Muscle spindles are distributed throughout muscle tissue. Muscle spindles consist of a small, innervated group of muscle fibers. The innervation provides for the sensing of spindle elongation and spindle muscle fiber activation. The spindle muscle fibers diminish the elongation of the spindle. By this method, spindle muscle fibers are used to provide negative feedback to the elongation sensing. This negative feedback is called the stretch reflex. By using the negative feedback path, the muscle organs have a position control input.
Tendons consist of strands of collagen fibers that connect muscle tissue to the skeleton. Tendons are innervated to provide an indication of tension, which is measured by the compression of nerve fibers between the collagen strands.

Joint receptors consist of a large variety of mechanoreceptors located in the joint capsule, ligaments and surrounding connective tissue. These receptors are used to monitor the joint position, velocity, and force.

**Bipedal walking**

Several bipedal gaits are possible. Some of them are classed as walking, running, and hopping (Chao, 1986, Alexander, 1977, Alexander, 1982, Eberhart, 1976, Ralston, 1976, Alexander, 1977a, Cavagna, 1977). Walking is accomplished by the cyclic movement of each leg from a support phase (active) to a relocation phase (passive) and back again. Walking is characterized by having at least one foot in ground contact at all times during the gait. The gait duty cycle is the ratio of active foot time to passive foot time. For walking, this ratio must be at least one half.

There are two general classifications to walking: the stiff walk and the compliant walk. The stiff walk is when the leg length is approximately fixed. In stiff walking the body moves in circular arcs during each step. In compliant walking, the leg length is shortened as the body passes over
the active foot. Compliant walking reduces the vertical movement.

Humans normally use a stiff walk. A detailed analysis of walking in humans indicates that each individual has his own particular walking gait, and only a general description is possible.

**Force and Velocity Patterns in Walking**

The biped's body accelerates and decelerates by the horizontal component of the active foot's ground reaction forces. These forces can be seen in the free-body diagram in Figure C.8.

The body forces result in periodic body accelerations. Typical accelerations are indicated in Figures C.11 and C.12. The corresponding body position changes can be seen in Figure C.13.

A detailed statistical study of human walking shows the periodic behavior of body velocity in human walking. Typical plots of body velocity are illustrated in Figure C.14.

**Energy in Walking**

Biologists have studied the energy expended in animals during different gaits. These studies usually involve the use of force platform to record the force on the ground while moving. From the forces, the energy used in moving is
determined. Energy storage can be observed by monitoring metabolic energy and comparing this to the total work done.

During bipedal walking, the pendulous nature of the walking gait transforms energy between potential and kinetic energy. In stiff walking, the potential energy is mainly gravimetric. In compliant walking, the potential energy is mainly elastic. Many biological systems can recover energy from negative work. Some of the energy lost in decelerating the body can be recovered and used to accelerated the body at the end of the stride.

The out-of-phase relationship between kinetic and potential energy can be seen in Alexander, 1977. The kinetic energy plotted is only the external kinetic energy. External kinetic energy is the energy associated with the movement of the center of mass; internal kinetic energy is the energy caused by relative movements of the body segments.

Theoretical Models for Walking Energy Consumption

R. McNeil Alexander has derived a theoretical model for the energy consumption used in stiff and compliant walking. The derivation assumed that all of the bipeds mass is in the body, and both feet are never on the ground simultaneously.
For stiff walking the energy consumption is:

\[ T_w = \left( \frac{\beta^3}{256 H^2 N} \right) \coth^2 \left( \frac{\beta}{4 \mu \left( \frac{g}{H} \right)^{\frac{1}{3}}} \right) \]  \hspace{1cm} (C.1)

Where:

- \( \beta \) is the length.
- \( H \) is the height of the hip.
- \( N \) is the efficiency.
- \( \mu \) is the mean velocity.
- \( g \) is gravity.

**Maximum Walking Velocity**

Biped walking can be approximated by assuming that all of the biped's mass is in the body and by assuming that the leg lengths are fixed. With these assumptions, a simple relationship can be derived that determines the maximum walking velocity. This relationship is formed by equating the centripetal acceleration with gravity.

The result is:

\[ |V_{\text{max}}| = \sqrt{GL} \]  \hspace{1cm} (C.2)

Where:

- \( G \) is gravity.
- \( L \) is the distance from ground contact to the center of mass.
- \( V_{\text{max}} \) is the maximum velocity.

These assumptions imply stiff walking, but the results are approximately true for compliant walking.
Control in Biological Walking Systems

Walking requires a complex pattern of muscle activations to provide the necessary coordinated limb movements. These muscle activations are caused by distributed processing throughout the nervous system. The most studied parts of the locomotion system is the periphery system (Schone, 1984b, Wetzel, 1977, Craik, 1976, Cook, 1976, Roberts, 1976, Edgerton, 1976).

Control Organization

Several studies have shown that the lower level motor control systems have different function subsystems. Four of these subsystems have been identified. The four neuron subsystem are: command neurons, oscillator neurons, coordinating neurons, and motoneuron (Davis, 1976).

Command Neurons

Command neurons are identified as those neurons that can illicit complex behaviors over several muscle groups. Simulation of these neurons produce basic behaviors such as swimming, walking, etcetera. Single neurons have been found to cause these higher level actions.

Oscillator Neurons

Oscillator neurons generate the basic timing for locomotion. These oscillators are phase locked together to
provide the desired timing. Higher level inputs control the phase and amplitude of these oscillations. These oscillator neurons have been observed in both inter and intra limb coordination. Neural oscillators have been studied by using animals in which higher level function are removed (decorticated). In these cases, the animals are still capable of gait generation. Examples of this gait generation can be seen in several animal studies.

**Coordination Neurons**

Coordination neurons provide coupling between different motor systems. Coordination neurons may cause the suppression of one motor act to allow others to be performed. For example, antagonistic muscle pairs should be used singularly.

**Motoneurons**

Motoneurons provide control over each of the muscle groups. Their activation causes the contraction of muscles. Either force or position inputs are possible.

**Locomotion Control Organization**

Each of the neuron classes can be stimulated and their respective responses observed. This result lead to the conclusion that the motor neuron system is organized in a hierarchial system. Recent data indicates, however, that the hierarchial architecture assumption is incorrect. A more
recent view is that neurons may be able to perform multiple
tasks, and that the interconnectivity of the neurons may be
more symmetrical.

The new organization has a structure characterized by
several attributes. Neural functions are distributed
throughout the motor control system. Control is accomplished
by a consensus of neural activity. Basic responses are an
emergent property of the system, not individuals units.
Information flows freely without regard to hierarchy.

The new architecture has several properties. First, the
new system is redundant by the equivalence of its neurons.
Positive feedback is possible and can produce reverberations
and spontaneous responses. Greater flexibility in learned
behavior is also afforded by this new system view.

Computer Modeling of Bipedal Motion

Kinematic and kinetic analyses of biological systems has
some similarity to the analysis of artificial bipeds. Kinesiologist are interested in the ground reaction and joint
forces in animal locomotion. These forces can be determined
by measuring the ground reactions and using computer models to
predict the internal forces.

The kinetic simulation programs for biological and
artificial bipeds have the same physical basis (Chao, 1986). Chao uses Lagrangian formulations to solve for the dynamics of
locomotion. Lagrangian equations permit the solution of
complex interconnected dynamic systems without the calculation of all of the internal forces. This is a great advantage for a system with a large number of degrees of freedom. A survey of biological modeling literature may be of use in writing biped simulators.

Conclusion

This section will review some observations that may be of use in constructing artificial bipedal systems. Some of these observations are the author's opinion after reviewing the references.

Artificial vestibular apparatus can be used to provide for body posture and relative center of mass location. Angular accelerometers combined with average (DC) corrections from gravity orientations (linear accelerometers) could provide the needed reference for body posture control. Linear accelerations measured with a known orientation can be used to measure the horizontal force component of walking. The horizontal force is a measure of the position of the relative center of mass.

Walking behavior can be generated by the interactions of small subsystems. The implementation of the control system used by nature in solving the bipedal walking problem is a distributed system of smaller interacting components. This type of problem decomposition may be a useful artificial architecture. Separate control systems could control: body
posture, body height, body velocity, leg position, leg trajectory planning, ankle torque, etcetera. Feedforward control is used in biological systems and is useful in several aspects of bipedal control. For example, feedforward force input to limb position control loops would improve trajectory errors when ground reaction forces are present. Feedforward control can be used in the posture control loop to compensate for walking induced posture disturbances. Feedforward control could also be used to improve the body velocity control loop performance.

Nature makes use of high pass filters (velocity) in sensor transduction. It is interesting to note that many natural sensor systems employ a kind of "AC coupling". The author suggests that nature uses highpass filtering to help limit the effects of sensor offsets.

Hierarchal control is not designed-in. System hierarchy appears to be a learned phenomena. In the process of learning to walk, the low level motor control system trains its computational units to specialize.

Both stiff and compliant walking occur in nature. There is a large variety of animals that walk with differing gaits. This indicates that several different gaits can provide useful locomotion.

Observations of the sensor bandwidths and sensitivities that nature uses in the vestibular and mechanoreceptor systems can help indicate the requirements for artificial systems.
These measurements are provided by the figures that are included.

Acceleration, velocity, and position profiles of bipedal walking are given. These profiles are useful for comparison to a proposed walking algorithm. Useful walking algorithms and trajectories might be suggested by the comparison.

Simple energy/velocity relationships have been calculated and tested and they apply to both natural and artificial bipeds. These relationships can be used as a test for achievable velocity trajectories.

In nature, walking has a duty cycle of less that one half. There may be some advantage in allowing a proposed walking algorithm to permit simultaneous foot contact. This may make weight transfer in walking occur more smoothly.
Diagram to show the principal structures of the membranous labyrinth and its neural connections.
(After Lindeman, 1969.)

Figure C.1 Semicircular Canals

From Benson, 1982
Cut-away view of an ampulla of a semicircular duct.

(After Lindeman, 1969.)

Figure C.2 Semicircular Duct

From Benson, 1982
Bode plot of primary afferent activity of two semicircular canal units, one "regular" (solid line) the other "irregular" (interrupted line).

Figure C.3 Bode Plot of Semicircular Canals

From Benson, 1982
Change of activity of 'regular' and 'irregular' semicircular canal primary afferents in response to a brief angular acceleration to a constant velocity of 300/sec and subsequent deceleration. (Data from Goldberg & Fernandez)

Figure C.4 Step Response of Semicircular Canals

From Benson, 1982
Gravireceptor organs. Schematic sections show receptors (stippled) in relation to dense mass (shaded). (A) Statocyst. Statoliths if free to fall can stimulate sensory cells on the lowest side of the sphere. (B) Otolith organ. Statolith displacements are limited, and coupled to sensory macula by a gelatinous membrane (dashed outline). (C) Pendant statolith organ. Deflection of pendant mass stimulates receptors that surround it, or its own receptors. (D) Tricholith organ. Deflections of pendant mass bends a stalk that stimulates an internal receptor.

Figure C.5 Gravireceptor Organs

From Lewis, 1984
Bode plots of primary afferent activity of two otolithic units, one 'regular' (solid line), the other irregular (interrupted line). Both gain and phase are referred to the acceleration of the forcing function. Solid and open circles identify values obtained during excitatory and inhibitory sine waves respectively.

Figure C.6 Bode Plot of Otolithic Organs

From Benson, 1982
Change in primary afferent activity of otolithic units in response to excitatory (on left) and inhibitory (on right) linear acceleration stimuli having a trapoidal trajectory with onset and offset durations of 4-5s. The graphs show the mean normalised responses of 21 'regular' units (solid line) and 14 'irregular' units (interrupted line); normalisation was based on the initial response to the excitatory stimulus.

Figure C.7 Step Response of Otolithic Organs

From Benson, 1982

Figure C.8 Walking Force Diagram

From Eberhart, 1976
Afferent discharge in different types of cutaneous mechanoreceptor in response to identical displacements of the skin indicated in the top tracing. The Pacinian corpuscle (PC) is excited by the initial deflection at the start of the ramp displacement and at stimulus frequencies of 100 and 300 Hz, whereas the other rapidly adapting receptors (RA) fire repetitively during the ramp, are silent like the PC at constant displacement and are excited by low frequency (20 Hz) and middle frequency (100 Hz) vibration.

The slowly adapting (SA) receptors have a resting discharge, and respond during steady displacement as well as during the ramp, and their responses are synchronised by low and middle frequencies of vibration. The C-mechanoreceptor (C-Mech) is less sensitive than the SA receptors, which have myelinated axons, and fires at a low rate during displacement. It is not synchronised by the vibratory stimuli.

Figure C.9 Mechanoreceptor Responses

From Iggo, 1982
Detection thresholds for three kinds of cutaneous receptors (SA, slowly-adapting, RA rapidly-adapting, PC Paacinian corpuscles) at various frequencies of sinusoidal stimulation.

Figure C.10 Detection Thresholds for Cutaneous Receptors

From Iggo, 1982
Horizontal acceleration and horizontal reaction, fore and aft

RHC: right heel contact, LHC: left heel contact

Figure C.11 Horizontal Body Accelerations for Walking

From Eberhart, 1976
Vertical acceleration and reaction, RHC: right heel contact, LHC: left heel contact.

Figure C.12 Vertical Body Accelerations for Walking

From Eberhart, 1976
Pelvis displacements, cm, walking 110 steps/min, in the transverse and vertical planes. RHC: right heel contact, LHC: left heel contact.

Figure C.13 Pelvis Displacement for Walking

From Eberhart, 1976
Typical tachometer traces showing velocity as a function of time for four different speeds of walking. The interval from one peak to the next represents one step cycle. H.S and I.O. designate single limb occurrences of heel strike and toe-off, respectively.

Figure C.14 Body Velocity for Walking vs. Speed

From Herman, 1976
Tension/length diagram for a reflexly active extensor muscle (soleus) in a decerbrate cat. The tension is varied sinusoidally over various ranges with a period of about 3 sec. The hysteresis loops are traced in a clockwise direction. The effect resembles that of distributed simple friction.

Figure C.15 Tension/Length Diagram for Muscles

From Roberts, 1976