Recurrent Gait of Anthropomorphic, Bipedal Walkers

by

Colleen E. Shannon

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE
in
Mechanical Engineering

Dr. Dankowicz, chair
Dr. Reinholtz
Dr. Robertshaw

May 19, 2003
Blacksburg, Virginia

Key words: Passive Walker, Human Walking, Locomotion,
Dynamical Systems, Feedback Control

Recurrent Gait of Anthropomorphic, Bipedal Walkers

Colleen E. Shannon

(ABSTRACT)

This thesis explores the dynamics of two bipedal, passive-walker models that are free to move in a three-dimensional environment. Specifically, two rigid-bodied walkers that can sustain anthropomorphic gait down an inclined plane with gravity being the only source of energy were studied using standard dynamical systems methods. This includes calculating the stability of periodic orbits and varying the system parameter to create bifurcation diagrams and to address the persistence of a periodic solution under specific parameter variations. These periodic orbits are found by implementing the Newton-Raphson root solving scheme. The dynamical systems associated with these periodic orbits are not completely smooth. Instead, they include discontinuities, such as those produced due to forces at foot contact points and during knee hyper-extension. These discontinuities are addressed in the stability calculations through appropriate discontinuity mappings.

The difference between the two walker models is the number of degrees of freedom (DOF) at the hip. Humans possess three DOF at each hip joint, one DOF at each knee joint, and at least two DOF at each ankle joint. The first walker model studied had revolute joints at the hips and knees and completely locked ankles. To make the walking motion more anthropomorphic, additional degrees of freedom were added to the hip. Specifically, the second walker model has ball joints at the hips.

Two control algorithms are used for controlling the local stability of periodic motions for both walker models. The methods, reference and delay feedback control, rely on the presence of discontinuities in the system. Moreover, it is possible to predict the effects of the control strategy based entirely on information from the uncontrolled system. Control is applied to both passive walker models to try and stabilize an unstable periodic gait by making small, discrete, changes in the foot orientation during gait. Results show that both methods are successful in stabilizing an unstable walking motion for a 3D model with one DOF in each hip and to reduce the instability of the walking motions for the model having more mobility in the hip joints.

Acknowledgments

First of all, I would like to thank my family, friends, and Jason for all their support especially near the end of my work. Secondly, I want to thank Dr. Dankowicz for working as my advisor and sharing his knowledge on this subject. I also want to thank Dr. Reinholtz and Dr. Robertshaw for serving on my committee and helping me through my three years in graduate school. Thirdly, I want to thank Dr. Petri Piiroinen for sharing his knowledge, Maple and Matlab code, and his many jokes. And finally, I greatly appreciate the financial support given by Jeffress Memorial Trust.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Understanding how a mechanical system works is only the beginning. One may also use this knowledge to improve the design of a mechanical system. This provides the rational for theoretical testing for any mechanical device before a final product is built. Why not use this approach for a better understanding of how humans walk?

In the work presented here, theoretical analysis is performed on a family of bipedal walkers modeling human gait. Specifically in this thesis, a continuation method is employed to extend a simple walker model into a more complex mechanism with an increased similarity to human kinematics and thus yielding a better understanding of human gait. In particular, emphasis is placed on passive walker models that can exhibit sustained gait in the absence of actuation by walking down an incline plane. This is based on the idea that an understanding of the natural dynamics of a system with no energy supplied to it allows for more energy efficient actuation in the case of gait on level ground.

Finally, the thesis discusses a semi-passive control strategy with the intent of stabilizing an otherwise unstable walking motion. The knowledge we gain from this and future research, will result in an improved understanding of human gait. This also may translate to building better and more energy-efficient walking robots and possibly improved prosthetic devices.

1.1 Goals

The intent of the work presented here is to investigate the dynamics of passive bipedal mechanisms. Specifically,

- to find a set of extended 2D walker motions,
- to add two extra degrees-of-freedom to each hip joint to make the walkers motions more anthropomorphic,
- to study the existence and stability of periodic gaits of the walker models, and
- to develop control algorithms that will use discrete changes in the system to stabilize an unstable motion.

1.2 Thesis outline

Chapter 2 consists of mathematical concepts that will be used to analyze selected models of bipedal walkers. Chapter 3 presents a brief introduction to the history of walking machines as well as their application to human rehabilitation. It also contains a detailed description of the passive walker models including the kinematics and kinetic modeling upon which they are based. Chapter 4 presents the results of an analysis of a walker with a single degree of rotational freedom at the hip joint. In Chapter 5, a similar study of a new 3D walker model with three degrees of freedom at each hip joint is discussed including various parameter studies and the imposition of feedback control.

Chapter 2

Mathematical tools and concepts

This chapter discusses the mathematical concepts that are used in subsequent chapters to analyze selected models of bipedal walkers. Specifically, simple, low-dimensional examples are used to illustrate concepts and tools that easily generalize to the higher-dimensional model walkers. For a detailed exposition of the various ideas presented here, see Strogatz [37], Adolfsson, Dankowicz & Nordmark [2], Dankowicz & Piiroinen[32], Piiroinen & Dankowicz [35], and Jerrelind and Dankowicz [19].

2.1 Nonlinear Dynamics

Any nonlinear dynamical system can be represented by a set of first-order differential equations of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \tag{2.1}$$

where $\mathbf{x}$ is an $n \times 1$ matrix describing the complete state of the dynamical system and the $n \times 1$ matrix function $\mathbf{f}$ is known as the vector field. The quantity n is the dimensionality of the dynamical system and is a low-level reflection of the complexity of the system. The vector field can be visualized as an infinite collection of arrows based at the points in some open region in state space. From the differential equation (2.1), it follows that the rate of change of the state, at a moment when $\mathbf{x} = \mathbf{x}^*$, is a vector parallel to the arrow based at $\mathbf{x}^*$ and is given by the vector $\mathbf{f}(\mathbf{x}^*)$. In other words, the collection of arrows describe the local state-space velocity and are thus tangential to solution trajectories through state-space. If the vector field $\mathbf{f}$ is continuous and has continuous partial derivatives throughout some region of state space, then there exists a unique solution satisfying the initial condition

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{2.2}$$

for any arbitrary t_0 and $\mathbf{x}_0$ in this region.

The dynamical system (2.1) is said to be autonomous, since the vector field is not dependent on the independent time variable, t. Autonomous dynamical systems are characterized by the uniqueness of their solutions and by the fact that changes in the state depend only on elapsed time and not on absolute time. This implies that solution trajectories cannot cross in state space and are represented by stationary curves in state space. These two

essential properties of an autonomous dynamical system allow for a single graphical representation, known as a state-space diagram. This type of diagram simultaneously displays qualitative information about all the trajectories of the system. In contrast, in an explicitly time-dependent dynamical system, the vector field changes with time making it impossible to represent solution trajectories in terms of stationary curves through state space and thus equally difficult to interpret the system dynamics from a state-space diagram.

A system of n differential equations that depends explicitly on time can be converted to an autonomous dynamical system by the inclusion of the time variable as an additional state variable. Consider, for example, the one-degree-of-freedom, forced harmonic oscillator

$$m\ddot{x} + b\dot{x} + kx = F\cos t. \tag{2.3}$$

From

$$x_1 = x, x_2 = \dot{x}, x_3 = t, \tag{2.4}$$

it follows that

$$\dot{x}_1 = x_2,\ \dot{x}_2 = \frac{1}{m}\left(-bx_2 - kx_1 + F\cos x_3\right),\ \dot{x}_3 = 1, \tag{2.5}$$

corresponding to an autonomous three-dimensional dynamical system that is equivalent to the original time-dependent system.

The behavior of a dynamical system can be studied through a combination of theoretical and numerical analysis. Numerical analysis is generally limited to the approximate solutions of selected initial-value problems. It is critical to appropriately select initial conditions that yield a solution that gives a comprehensive understanding of the system behavior. This is especially important for high-dimensional systems, since it is virtually impossible to numerically approximate the solution trajectories for a significant region of state space. One of the benefits of a theoretical analysis is for guidance in the search for appropriate initial conditions. This typically restricts the focus to local regions of state space that show significant influence on the system behavior.

2.2 The Flow

It is generally impossible to find a closed-form solution to (2.1) that satisfies a given initial condition. Nevertheless, the basic existence and uniqueness result for an autonomous dynamical system implies the existence of a vector function $\mathbf{\Phi}(\mathbf{x}, t)$ that is as smooth as the vector field and is known as the **flow function**, such that

$$\frac{\partial}{\partial t}\mathbf{\Phi}(\mathbf{x}, t) = \mathbf{f}\left(\mathbf{\Phi}(\mathbf{x}, t)\right) \tag{2.6}$$

and

$$\mathbf{\Phi}(\mathbf{x}, 0) = \mathbf{x}, \tag{2.7}$$

for any $\mathbf{x}$ and all t in some interval containing 0. It follows that $\mathbf{\Phi}(\mathbf{x}, t)$ describes the solution trajectory based at the point $\mathbf{x}$ at time $t = 0$. Also, $\mathbf{\Phi}(\cdot, t)$ represents the collection of stationary curves through state space corresponding to arbitrary solution trajectories. The terminology 'flow' originates in considering this collection of stationary curves as the

flow curves of a stationary fluid flow. In this interpretation, the vector field represents the field of velocity vectors of particles of fluid. Although we do not generally have access to $\mathbf{\Phi}(\mathbf{x}, t)$ in closed form, it is a convenient tool in the theoretical analysis of a system.

2.3 Equilibrium Points

A solution trajectory that coincides with a single point $\mathbf{x}^*$ in state space is called an equilibrium point. It follows that

$$\mathbf{\Phi}(\mathbf{x}^*, t) = \mathbf{x}^*, \tag{2.8}$$

for all t. From (2.6) the vector field is

$$\mathbf{f}(\mathbf{x}^*) = \mathbf{f}(\mathbf{\Phi}(\mathbf{x}^*, t)) = \frac{\partial}{\partial t}\mathbf{\Phi}(\mathbf{x}^*, t) = \frac{\partial}{\partial t}\mathbf{x}^* = \mathbf{0}, \tag{2.9}$$

i.e., the equilibrium points correspond to zero points of the vector field.

The local behavior of a dynamical system, in the vicinity of an equilibrium point, can be qualitatively described in terms of the **stability** of the equilibrium point. Specifically, an equilibrium point is said to be **Liapunov stable** if for every neighborhood U of the equilibrium point there exists a smaller neighborhood V, such that the solution trajectory based at any initial condition in V stays in U for all future time. Additionally, if all trajectories approach the equilibrium point as $t \to \infty$, the equilibrium point is said to be **asymptotically stable**. An equilibrium point is said to be **unstable** if it is not stable.

The stability of an equilibrium point may be studied analytically by considering the flow for initial conditions near the equilibrium point. Specifically,

$$\begin{aligned}
\frac{\partial}{\partial t}\left[\mathbf{\Phi}(\mathbf{x}^*+\mathbf{\Delta x}, t) - \mathbf{\Phi}(\mathbf{x}^*, t)\right] &= \mathbf{f}(\mathbf{\Phi}(\mathbf{x}^*+\mathbf{\Delta x}, t)) - \mathbf{f}(\mathbf{\Phi}(\mathbf{x}^*, t)) \\
&= \mathbf{D_x f}(\mathbf{\Phi}(\mathbf{x}^*, t))\left[\mathbf{\Phi}(\mathbf{x}^*+\mathbf{\Delta x}, t) - \mathbf{\Phi}(\mathbf{x}^*, t)\right] + \mathcal{O}(2) \\
&\approx \mathbf{D_x f}(\mathbf{x}^*)\left[\mathbf{\Phi}(\mathbf{x}^*+\mathbf{\Delta x}, t) - \mathbf{\Phi}(\mathbf{x}^*, t)\right],
\end{aligned} \tag{2.10}$$

where $\mathbf{\Delta x}$ is small, $\mathbf{D_x f}$ is the Jacobian matrix of first partial derivatives of the vector field $\mathbf{f}$

$$\mathbf{D_x f} = \begin{pmatrix} \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{f}_1}{\partial \mathbf{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{f}_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{f}_n}{\partial \mathbf{x}_n} \end{pmatrix}, \tag{2.11}$$

and the omitted terms are of order two or higher in the difference

$$\mathbf{\Phi}(\mathbf{x}^*+\mathbf{\Delta x}, t) - \mathbf{\Phi}(\mathbf{x}^*, t) = \mathcal{O}(\mathbf{\Delta x}). \tag{2.12}$$

It follows that the local behavior of the dynamical system is governed to lowest order by the linear, autonomous dynamical system

$$\dot{\mathbf{y}} = \mathbf{D_x f}(\mathbf{x}^*)\,\mathbf{y}. \tag{2.13}$$

From the theory of linear systems of differential equations, we conclude that $\mathbf{y} \to \mathbf{0}$ for all initial conditions if all the eigenvalues of $\mathbf{D_x f}(\mathbf{x}^*)$ have negative real part, i.e., the equilibrium is linearly asymptotically stable. If any eigenvalue has a positive real part, then there exist initial conditions for which the corresponding solution trajectories leave any neighborhood of $\mathbf{0}$, i.e., the equilibrium is linearly unstable. Finally, if all eigenvalues have nonpositive real part, the equilibrium is linearly stable.

That some of the results of the linear analysis carry over to the original nonlinear system follows from the Hartman-Grobman theorem. Specifically, if the equilibrium point is hyperbolic, i.e., if all the eigenvalues of the Jacobian matrix have nonzero real part, the higher-order terms do not affect the prediction of the linear stability analysis. In contrast, for a nonhyperbolic equilibrium point at least one eigenvalue has a zero real part and a nonlinear analysis is necessary to determine the stability properties of the equilibrium point.

2.4 Approximation of equilibrium

2.4.1 Implicit function theorem

The implicit function theorem yields information about the persistence of equilibrium points under small changes in system parameters. Suppose that the vector-valued function $\mathbf{f}(\mathbf{x}, \mu)$ is continuous and is continuously differentiable in some region containing the point $(\mathbf{x}_0, \mu_0)$, such that

$$\mathbf{f}(\mathbf{x}_0, \mu_0) = \mathbf{0}. \tag{2.14}$$

Furthermore, suppose that the Jacobian $\mathbf{D_x f}(\mathbf{x}_0, \mu_0)$ is invertible, i.e.,

$$\det \mathbf{D_x f}(\mathbf{x}_0, \mu_0) \neq 0, \tag{2.15}$$

or, equivalently, that no eigenvalue of the Jacobian equals zero. The implicit function theorem states that there exists a unique function $\mathbf{g}(\mu)$ for all μ near μ_0, such that

$$\mathbf{g}(\mu_0) = \mathbf{x}_0, \tag{2.16}$$

$$\mathbf{f}(\mathbf{g}(\mu), \mu) = \mathbf{0}, \tag{2.17}$$

and

$$\mathbf{D}_\mu \mathbf{g}(\mu_0) = -(\mathbf{D_x f}(\mathbf{x}_0, \mu_0))^{-1} \cdot \mathbf{D}_\mu \mathbf{f}(\mathbf{x}_0, \mu_0). \tag{2.18}$$

In particular, this expression shows that

$$\mathbf{g}(\mu) \approx \mathbf{x}_0 - (\mathbf{D_x f}(\mathbf{x}_0, \mu_0))^{-1} \cdot \mathbf{D}_\mu \mathbf{f}(\mathbf{x}_0, \mu_0) \cdot (\mu - \mu_0), \ \mu \approx \mu_0. \tag{2.19}$$

Thus, if $\mathbf{x} = \mathbf{x}_0$ is an equilibrium point for $\mu = \mu_0$ and if no eigenvalue of the corresponding Jacobian matrix equals zero, then there exists a unique equilibrium point for $\mu \approx \mu_0$ close to $\mathbf{x}_0$. In particular, hyperbolic equilibrium points are persistent under small changes in system parameters.

Example 1 *As an illustration of the implicit function theorem, consider the vector field*

$$f(x, \mu) = \mu - x^2, \tag{2.20}$$

The equilibrium points for the corresponding dynamical system are located at $x^* = \pm\sqrt{\mu}$*, for* $\mu \geq 0$*. In particular for* $(x_0, \mu_0) = (1, 1)$*,*

$$f(x_0, \mu_0) = 0. \tag{2.21}$$

Since

$$D_x f(x, \mu) = -2x, \tag{2.22}$$

it follows that

$$D_x f(x_0, \mu_0) = -2 \neq 0. \tag{2.23}$$

Therefore, the implicit function theorem implies that there exists a unique function $g(\mu)$ *for* μ *near* $\mu_0 = 1$*, such that*

$$\begin{aligned} g(\mu_0) &= x_0 = 1, & (2.24)\\ f(g(\mu), \mu) &= \mu - (g(\mu))^2 = 0 & (2.25) \end{aligned}$$

and

$$D_\mu g(\mu_0) = -\left(D_x f(x_0, \mu_0)\right)^{-1} \cdot D_\mu f(x_0, \mu_0) = \frac{1}{2}. \tag{2.26}$$

Using this information, we find

$$g(\mu_0 + \Delta\mu) \approx g(\mu_0) + \frac{1}{2}\Delta\mu. \tag{2.27}$$

Testing this for $\mu = 1.1$*,* $g(1.1) \approx g(1) + \frac{1}{2} \cdot 0.1 = 1.05$*, whereas, the actual equilibrium point is at* $\sqrt{1.1} \approx 1.0488$*.*

2.4.2 Newton-Raphson method

The implicit function theorem provides a rigorous statement about the persistence of equilibrium points and gives an idea of the rate of change of the location of the equilibrium point under small changes in system parameters. To actually find the new equilibrium point, a different approach is necessary. The Newton-Raphson method is an iterative method for finding equilibrium points, given a good initial guess for their location.

Let $\mathbf{x}_0$ be an initial guess and suppose that there exists a small quantity $\Delta\mathbf{x}$, such that

$$\mathbf{0} = \mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{D_x f}(\mathbf{x}_0)\,\Delta\mathbf{x} + \mathcal{O}(2). \tag{2.28}$$

Suppose that the Jacobian matrix $\mathbf{D_x f}(\mathbf{x}_0)$ is invertible. Then, neglecting terms of order two or higher yields

$$\mathbf{\Delta x} \approx -\left(\mathbf{D_x f}(\mathbf{x}_0)\right)^{-1} \cdot \mathbf{f}(\mathbf{x}_0). \tag{2.29}$$

Thus, a better approximation of the equilibrium point should be

$$\mathbf{x}_0 - \left(\mathbf{D_x f}(\mathbf{x}_0)\right)^{-1} \cdot \mathbf{f}(\mathbf{x}_0) \tag{2.30}$$

Repeating this process, we arrive at the iterative formula

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \left(\mathbf{D_x f}(\mathbf{x}_i)\right)^{-1} \cdot \mathbf{f}(\mathbf{x}_i) \tag{2.31}$$

that can be shown to converge very rapidly to the equilibrium point, provided that the initial guess is sufficiently good.

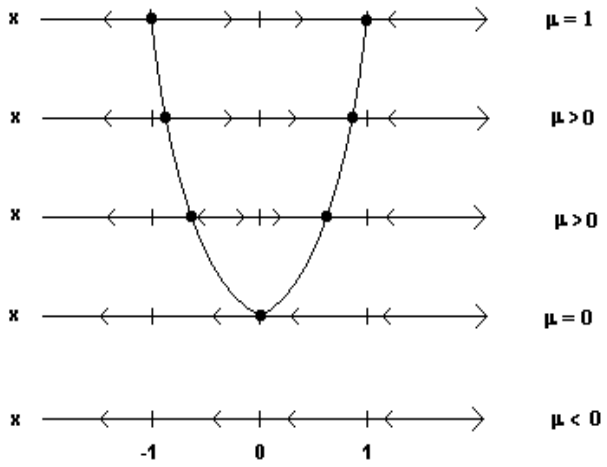

Figure 2.1: Changes in the vector field and the location of equilibrium points as μ is varied.

Example 2 *To incorporate the aforementioned information, return to the one-dimensional dynamical system considered in the previous example*

$$\dot{x} = f(x) = \mu - x^2. \tag{2.32}$$

We previously found two equilibrium points at $x^* = \mp\sqrt{\mu}$ *for* $\mu \geq 0$ *and no equilibria when* $\mu < 0$*. In terms of the flow function, we have*

$$\Phi\left(\mp\sqrt{\mu}, t\right) = \mp\sqrt{\mu},\ \mu \geq 0, \tag{2.33}$$

for all t*. To lowest order, the stability properties of the equilibria are governed by the linear differential equation*

$$\dot{y} = D_{\mathbf{x}} f\left(x^*\right) y = \mp 2\sqrt{\mu}\, y, \tag{2.34}$$

Clearly, as long as $\mu > 0$*, the equilibria are hyperbolic and by the Hartman-Grobman theorem one is asymptotically stable and the other is unstable.*

A graphical representation of the flow for this dynamical system is shown in Figure 2.1. In this figure, each horizontal line represents a different state space diagram thus illustrating changes in the vector field under changes in μ*. As suggested by the theoretical analysis, there are two equilibria for* $\mu > 0$*: a stable equilibrium at* $\sqrt{\mu}$ *and an unstable equilibrium at* $-\sqrt{\mu}$*. Similarly, for* $\mu < 0$*, there are no equilibria. The nonhyperbolic equilibrium point* $x^* = 0$ *at* $\mu = 0$ *fails to satisfy the implicit function theorem. This confirms the observation that there is no unique equilibrium point for* $\mu \approx 0$ *near* $x = 0$*. Instead, the point* $(x^*, \mu) = (0, 0)$ *is a* ***bifurcation point****, also known as a saddle-node bifurcation, at which two branches of equilibria of opposite stability meet and annihilate. We will return to a more detailed discussion of bifurcations in Section 2.8.*

For a more general dynamical system, it would not be possible to carry out the above analysis in closed form, since the equilibria would not be known explicitly. To illustrate the proposed

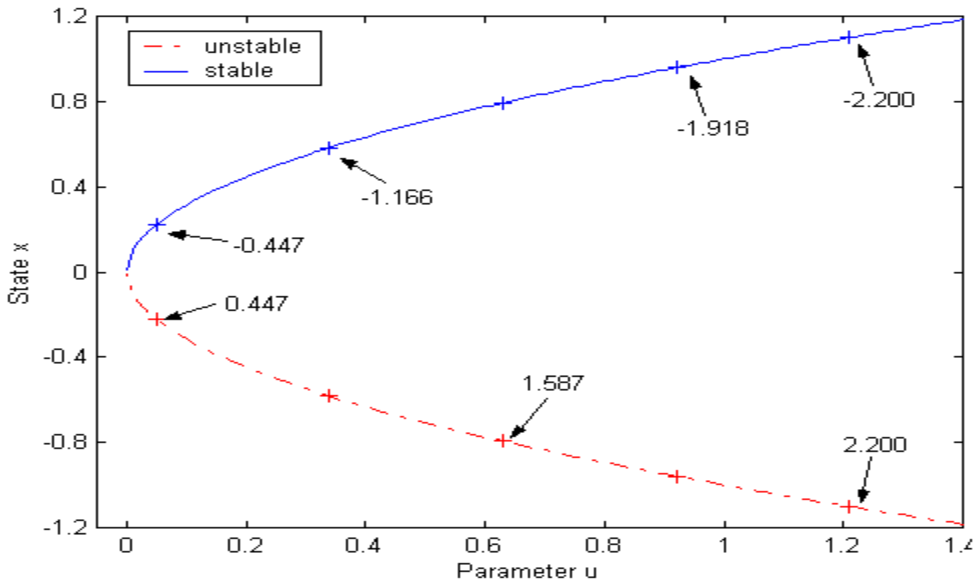

Figure 2.2: Persistence and stability of equilibria under variations on the parameter μ.

methodology in such a situation, we will study the above dynamical system using Matlab, see Appendix A for the code used to generate the following results. Specifically, we will use two Matlab functions to approach this system, namely `vectornewton` *and* `ode45_1DOFsystem`*. Here, the* `vectornewton` *function uses the Newton-Raphson scheme to locate equilibria for the dynamical system given by the vector field specified in* `ode45_1DOFsystem` *over a range of parameter values given a good initial guess for some initial parameter value. The function* `ode45_1DOFsystem` *can be used to compute the vector field and the Jacobian, both of which are used within the* `vectornewton` *function.*

Figure 2.2 shows the results of an application of the Matlab code to numerically locate the equilibria for μ between 0 and 1.4 with a step size of 0.1. This plot also shows the eigenvalues of the Jacobian matrix computed at selected locations along the two branches of equilibria. As can be seen, the bottom branch has all positive eigenvalues, corresponding to unstable equilibria. Similarly, for the top branch all eigenvalues are negative, corresponding to stable equilibria. Clearly, as the saddle-node bifurcation point is approached, the eigenvalues approach 0 from below and above.

2.5 Periodic Orbits

A solution trajectory that forms a closed curve in state space is called a periodic orbit. It follows that there exists a nonzero quantity T, such that

$$\mathbf{\Phi}\left(\mathbf{x}^*, t+T\right) = \mathbf{\Phi}\left(\mathbf{x}^*, t\right), \tag{2.35}$$

for all $\mathbf{x}^*$ on the trajectory and for all t. Periodic orbits are most conveniently studied through the introduction of a Poincaré section and its associated Poincaré map. Suppose

that the periodic orbit intersects the zero-level surface of the function $h(\mathbf{x})$ transversally at a point $\mathbf{x}^*$, i.e.,

$$h(\mathbf{\Phi}(\mathbf{x}^*, T)) = h(\mathbf{x}^*) = 0 \tag{2.36}$$

and

$$D_{\mathbf{x}} h(\mathbf{x}^*) \cdot \mathbf{f}(\mathbf{x}^*) \neq 0. \tag{2.37}$$

Now consider the scalar-valued function

$$g(\mathbf{x}, \tau) = h(\mathbf{\Phi}(\mathbf{x}, \tau)). \tag{2.38}$$

Then,

$$g(\mathbf{x}^*, T) = 0 \tag{2.39}$$

and

$$\begin{aligned} D_\tau g(\mathbf{x}^*, T) &= D_{\mathbf{x}} h(\mathbf{\Phi}(\mathbf{x}^*, T)) \cdot \frac{\partial}{\partial \tau} \mathbf{\Phi}(\mathbf{x}^*, T) \\ &= D_{\mathbf{x}} h(\mathbf{x}^*) \cdot \mathbf{f}(\mathbf{\Phi}(\mathbf{x}^*, T)) \\ &= D_{\mathbf{x}} h(\mathbf{x}^*) \cdot \mathbf{f}(\mathbf{x}^*) \neq 0. \end{aligned} \tag{2.40}$$

The implicit function theorem then implies the existence of a unique function $\tau(\mathbf{x})$ for $\mathbf{x} \approx \mathbf{x}^*$, such that

$$\tau(\mathbf{x}^*) = T, \tag{2.41}$$

$$g(\mathbf{x}, \tau(\mathbf{x})) = h(\mathbf{\Phi}(\mathbf{x}, \tau(\mathbf{x}))) = 0 \tag{2.42}$$

and

$$\begin{aligned} D_{\mathbf{x}} \tau(\mathbf{x}^*) &= -(D_\tau g(\mathbf{x}^*, T))^{-1} \cdot D_{\mathbf{x}} g(\mathbf{x}^*, T) \\ &= -\frac{D_{\mathbf{x}} h(\mathbf{x}^*)}{D_{\mathbf{x}} h(\mathbf{x}^*) \cdot \mathbf{f}(\mathbf{x}^*)} \mathbf{D}_{\mathbf{x}} \mathbf{\Phi}(\mathbf{x}^*, T). \end{aligned} \tag{2.43}$$

This shows that for each trajectory based at an initial point near $\mathbf{x}^*$ there exists a unique elapsed time near T until the trajectory intersects the Poincaré section corresponding to $h(\mathbf{x}) = 0$. Thus, we can define a Poincaré map as

$$\mathbf{P}(\mathbf{x}) = \mathbf{\Phi}(\mathbf{x}, \tau(\mathbf{x})), \tag{2.44}$$

that maps points near $\mathbf{x}^*$ onto the Poincaré section. Specifically,

$$\mathbf{P}(\mathbf{x}^*) = \mathbf{x}^* \tag{2.45}$$

is a fixed point of the Poincaré map.

The local behavior of a dynamical system in the vicinity of a periodic orbit can be qualitatively described in terms of the **stability** of the fixed point $\mathbf{x}^*$of the Poincaré map. Specifically, the fixed point is said to be **Liapunov stable** if for every neighborhood U of the fixed point in the Poincaré section there exists a smaller neighborhood V in the Poincaré section, such that intersections with the Poincaré section of any solution trajectory based

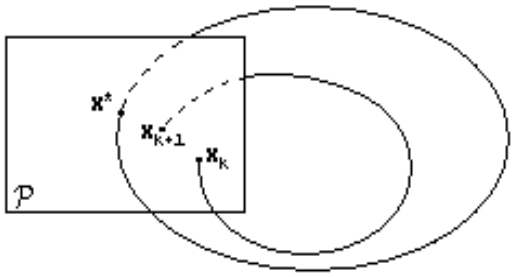

Figure 2.3: Two consecutive intersections ($\mathbf{x_k}$ and $\mathbf{x_{k+1}}$) with the Poincaré section $\mathcal{P}$ and a fixed point of a Poincaré mapping.

at any initial condition in V stay in U for all future time. Additionally, if all intersections approach the fixed point as $t \to \infty$, the fixed point is said to be **asymptotically stable.** A fixed point that is not stable is said to be **unstable**.

The stability of the fixed point $\mathbf{x}^*$of the Poincaré map may be studied analytically by considering the Poincaré map for initial conditions near the fixed point. Specifically,

$$\begin{aligned}
\mathbf{P}\left(\mathbf{x}^*+\Delta\mathbf{x}\right)-\mathbf{P}\left(\mathbf{x}^*\right) &= \mathbf{\Phi}\left(\mathbf{x}^*+\Delta\mathbf{x},\tau\left(\mathbf{x}^*+\Delta\mathbf{x}\right)\right)-\mathbf{\Phi}\left(\mathbf{x}^*,\tau\left(\mathbf{x}^*\right)\right) \\
&= \left(\mathbf{D_x\Phi}\left(\mathbf{x}^*,\tau\left(\mathbf{x}^*\right)\right)+\frac{\partial}{\partial t}\mathbf{\Phi}\left(\mathbf{x}^*,\tau\left(\mathbf{x}^*\right)\right)\cdot D_{\mathbf{x}}\tau\left(\mathbf{x}^*\right)\right)\cdot\Delta\mathbf{x}+\mathcal{O}\left(2\right) \\
&\approx \left(Id-\frac{\mathbf{f}\left(\mathbf{x}^*\right)\cdot D_{\mathbf{x}}h\left(\mathbf{x}^*\right)}{D_{\mathbf{x}}h\left(\mathbf{x}^*\right)\cdot\mathbf{f}\left(\mathbf{x}^*\right)}\right)\cdot\mathbf{D_x\Phi}\left(\mathbf{x}^*,T\right)\cdot\Delta\mathbf{x} \\
&= \mathbf{D_xP}\left(\mathbf{x}^*\right)\cdot\Delta\mathbf{x}, \qquad (2.46)
\end{aligned}$$

where the omitted terms are of order two or higher in $\Delta\mathbf{x}$. It follows that the local behavior of the dynamical system is governed to lowest order by the linear map

$$\mathbf{y}_{i+1}=\mathbf{D_xP}\left(\mathbf{x}^*\right)\mathbf{y}_i. \qquad (2.47)$$

From the theory of linear maps, we conclude that $\mathbf{y}_i\to\mathbf{0}$ for all initial conditions if all the eigenvalues of $\mathbf{D_xP}\left(\mathbf{x}^*\right)$ lie within the unit circle, i.e., the fixed point is linearly asymptotically stable. If any eigenvalue lies outside the unit circle, then there exist initial conditions for which $\mathbf{y}_i$ leaves any neighborhood of $\mathbf{0}$, i.e., the fixed point is linearly unstable. Finally, if all eigenvalues lie on or inside the unit circle, the fixed point is linearly stable.

As with equilibria, some of the results of the linear analysis carry over to the original nonlinear system, again by a version of the Hartman-Grobman theorem. If the fixed point is hyperbolic, i.e., if none of the eigenvalues of $\mathbf{D_xP}\left(\mathbf{x}^*\right)$ lie on the unit circle, then the higher-order terms do not affect the prediction of the linear stability analysis. In contrast, for a nonhyperbolic fixed point, a nonlinear analysis is necessary to determine the stability properties of the fixed point, and hence the associated periodic orbit.

2.6 Approximation of fixed points

As with equilibria, the implicit function theorem applies to the study of the persistence of periodic orbits under parameter variations. Specifically, suppose that $\mathbf{x} = \mathbf{x}_0$ is a fixed point of the Poincaré map for $\mu = \mu_0$. It follows that

$$\mathbf{P}(\mathbf{x}_0, \mu_0) - \mathbf{x}_0 = \mathbf{0}. \tag{2.48}$$

Now consider the vector-valued function

$$\mathbf{F}(\mathbf{x}, \mu) = \mathbf{P}(\mathbf{x}, \mu) - \mathbf{x}, \tag{2.49}$$

which is continuously differentiable as long as the intersection of the corresponding periodic orbit with the Poincaré section is transversal. Then,

$$\mathbf{F}(\mathbf{x}_0, \mu_0) = \mathbf{0} \tag{2.50}$$

and

$$\mathbf{D}_{\mathbf{x}}\mathbf{F}(\mathbf{x}_0, \mu_0) = \mathbf{D}_{\mathbf{x}}\mathbf{P}(\mathbf{x}_0, \mu_0) - Id. \tag{2.51}$$

This matrix is invertible provided that

$$\det(\mathbf{D}_{\mathbf{x}}\mathbf{P}(\mathbf{x}_0, \mu_0) - Id) \neq 0, \tag{2.52}$$

in other words, as long as no eigenvalue of $D_{\mathbf{x}}\mathbf{P}(\mathbf{x}_0, \mu_0)$ equals 1. The implicit function theorem then implies the existence of a unique function $\mathbf{g}(\mu)$ for $\mathbf{x} \approx \mathbf{x}_0$, such that

$$\begin{aligned} \mathbf{g}(\mu_0) &= \mathbf{x}_0, & (2.53)\\ \mathbf{F}(\mathbf{g}(\mu), \mu) &= \mathbf{P}(\mathbf{g}(\mu), \mu) - \mathbf{g}(\mu) = 0 & (2.54) \end{aligned}$$

and

$$\begin{aligned} \mathbf{D}_{\mu}\mathbf{g}(\mu_0) &= -(\mathbf{D}_{\mathbf{x}}\mathbf{F}(\mathbf{x}_0, \mu_0))^{-1} \cdot \mathbf{D}_{\mu}\mathbf{F}(\mathbf{x}_0, \mu_0) \\ &= -(\mathbf{D}_{\mathbf{x}}\mathbf{P}(\mathbf{x}_0, \mu_0) - Id)^{-1} \cdot \mathbf{D}_{\mu}\mathbf{P}(\mathbf{x}_0, \mu_0). \end{aligned} \tag{2.55}$$

Since the fixed point corresponding to the intersection of the periodic orbit with the Poincaré section has been shown to be the solution to an equation $\mathbf{F}(\mathbf{x}) = \mathbf{P}(\mathbf{x}) - \mathbf{x} = \mathbf{0}$, we may again apply the Newton-Raphson method to numerically locate a new fixed point given a good initial guess.

It remains to compute $\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}(\mathbf{x}, t)$ as the other quantities have been solved for previously (2.6, 2.7, and 2.43). Recall from Equations (2.6-2.7) that

$$\frac{\partial}{\partial t}\mathbf{\Phi}(\mathbf{x}, t) = \mathbf{f}(\mathbf{\Phi}(\mathbf{x}, t)) \tag{2.56}$$

and

$$\mathbf{\Phi}(\mathbf{x}, 0) = \mathbf{x}. \tag{2.57}$$

From the first equation, we find

$$\begin{aligned}\frac{\partial}{\partial t}\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}\left(\mathbf{x},t\right) &= \mathbf{D}_{\mathbf{x}}\frac{\partial}{\partial t}\mathbf{\Phi}\left(\mathbf{x},t\right) \\ &= \mathbf{D}_{\mathbf{x}}\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{x},t\right)\right)\cdot\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}\left(\mathbf{x},t\right). \end{aligned} \tag{2.58}$$

Similarly, differentiating the second equation with respect to x gives

$$\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}\left(\mathbf{x},0\right) = Id. \tag{2.59}$$

It follows that $\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}\left(\mathbf{x},t\right)$ can be found by solving the matrix differential equation

$$\dot{\mathbf{X}} = \mathbf{D}_{\mathbf{x}}\mathbf{f}\left(\mathbf{\Phi}\left(\mathbf{x},t\right)\right)\mathbf{X} \tag{2.60}$$

known as the first variational equation together with the initial condition

$$\mathbf{X}\left(0\right) = Id \tag{2.61}$$

for a time t.

Example 3 *Now consider the two-dimensional dynamical system*

$$\dot{\mathbf{x}} = \mathbf{f}\left(\mathbf{x}\right) = \begin{pmatrix} \mu x_1^2 - x_1 x_2 \\ -x_2 + x_1^2 \end{pmatrix}. \tag{2.62}$$

There are two equilibrium points, namely

$$\mathbf{x}^* = \mathbf{0} \text{ and } \mathbf{x}^* = \begin{pmatrix} \mu \\ \mu^2 \end{pmatrix}. \tag{2.63}$$

To lowest order, stability of these equilibria is given by the eigenvalues of the Jacobian matrix

$$\mathbf{D}_{\mathbf{x}}\mathbf{f}\left(\mathbf{x}^*\right) = \begin{pmatrix} 2\mu x_1^* - x_2^* & -x_1^* \\ 2x_1^* & -1 \end{pmatrix}. \tag{2.64}$$

Specifically, when $\mathbf{x}^* = \mathbf{0}$, we find the eigenvalues to be 0 and -1. The equilibrium is linearly stable, but nonhyperbolic and hence a nonlinear analysis is necessary to determine the stability. On the other hand, for $\mathbf{x}^* = \begin{pmatrix} \mu & \mu^2 \end{pmatrix}^T$, the eigenvalues are

$$\lambda_{1,2} = -\frac{1-\mu^2}{2} \pm \sqrt{\frac{(1-\mu^2)^2}{4} - \mu^2}. \tag{2.65}$$

*If the absolute value of μ is greater than one, both eigenvalues have positive real part and thus the equilibrium is unstable. For μ with absolute value less then one and not equal to zero, the real part of the eigenvalues is negative, thus the equilibrium is stable. Nonhyperbolic equilibria exist for $\mu = \pm 1$, where the eigenvalues are $\pm i$, and for $\mu = 0$, for which the eigenvalues are 0 and -1. In fact, the nonhyperbolic equilibrium when $\mu = \pm 1$ is associated with a **Hopf bifurcation**. In conjunction with a change in stability of the equilibrium point, the birth of a branch of periodic orbits occurs.*

We can locate the periodic orbits born at the Hopf bifurcation and compute their stability by a combination of the Newton-Raphson method and a calculation of the eigenvalues of the Jacobian of the Poincaré map $\mathbf{D_x P}(\mathbf{x}^*)$ *at the corresponding fixed point* $\mathbf{x}^*$ *on the Poincaré section. As noted above, it is necessary to find the Jacobian of the flow function* $\mathbf{D_x \Phi}(\mathbf{x}, t)$*, obtained from the first variational equation (recall Equations (2.60-2.61))*

$$\dot{\mathbf{X}} = \mathbf{D_x f}(\mathbf{\Phi}(\mathbf{x}, \mathbf{t}))\,\mathbf{X} \tag{2.66}$$

with initial condition $\mathbf{X}(0) = Id$*. Thus, if we let*

$$\mathbf{X} = \begin{pmatrix} x_3 & x_4 \\ x_5 & x_6 \end{pmatrix}, \tag{2.67}$$

then the variational equation combined with the original dynamical system yields the new dynamical system

$$\dot{\mathbf{x}} = \begin{pmatrix} \mu x_1^2 - x_1 x_2 \\ -x_2 + x_1^2 \\ (2\mu x_1 - x_2)x_3 - x_1 x_5 \\ (2\mu x_1 - x_2)x_4 - x_1 x_6 \\ 2x_1 x_3 - x_5 \\ 2x_1 x_4 - x_6 \end{pmatrix} \tag{2.68}$$

with suitable initial conditions.

To illustrate the methodology proposed above, the following numerical results were obtained using the Matlab functions `vectornewton`*,* `ode45_moreDOFsystem` *and* `Hopf` *(see Appendix B). These functions implement the Newton-Raphson method for finding equilibria as well as periodic orbits. Figure 2.4 shows the location of equilibria in state space under variations of* μ *from* -1.3 *to* 1.3 *in steps of* $.01$*. Here, solid lines are used to represent asymptotically stable equilibria, while dashed lines represent unstable equilibria.*

To locate the periodic orbits born in the Hopf bifurcation at $\mu = \pm 1$*, we introduce a Poincaré section corresponding to the zero-level surface of the function* $h(\mathbf{x}) = x - 1$*. Using the Newton-Raphson method, as implemented in* `Hopf`*, a family of periodic orbits can be found as shown in Figure 2.5. It is clear that the area enclosed by the periodic orbit increases as* $|\mu|$ *increases beyond* 1*. In fact, for* $\mu > 1$*, the distance of the point of intersection on the Poincaré section to the point* $(x_1, x_2) = (1, 1)$*grows as* $\sqrt{\mu - 1}$*, and similarly for* $\mu < -1$ *(see Figure 2.6). Finally, Figure 2.6 also shows the variation in the period* T *of the periodic orbits under variations in* μ*. Again, we see that the period increases as* $|\mu|$ *increases beyond 1. In fact, the period approaches* 2π *as* $|\mu| \to 1$*.*

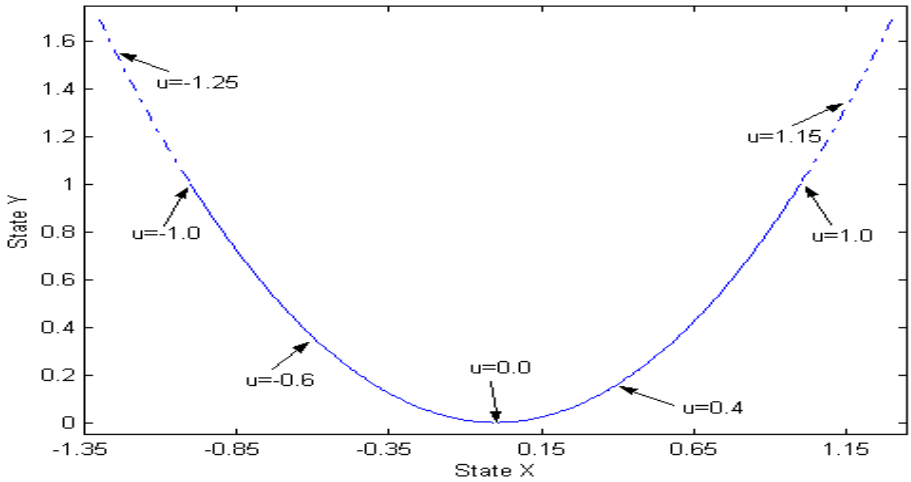

Figure 2.4: A state space plot showing the persistence and stability of the equilibrium points under variations in μ.

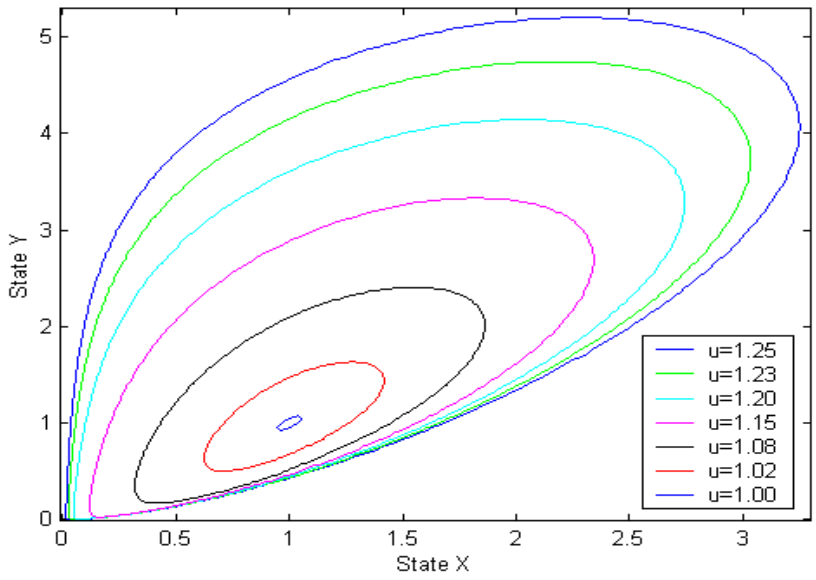

Figure 2.5: A family of periodic orbits born at the Hopf bifurcation for different μ.

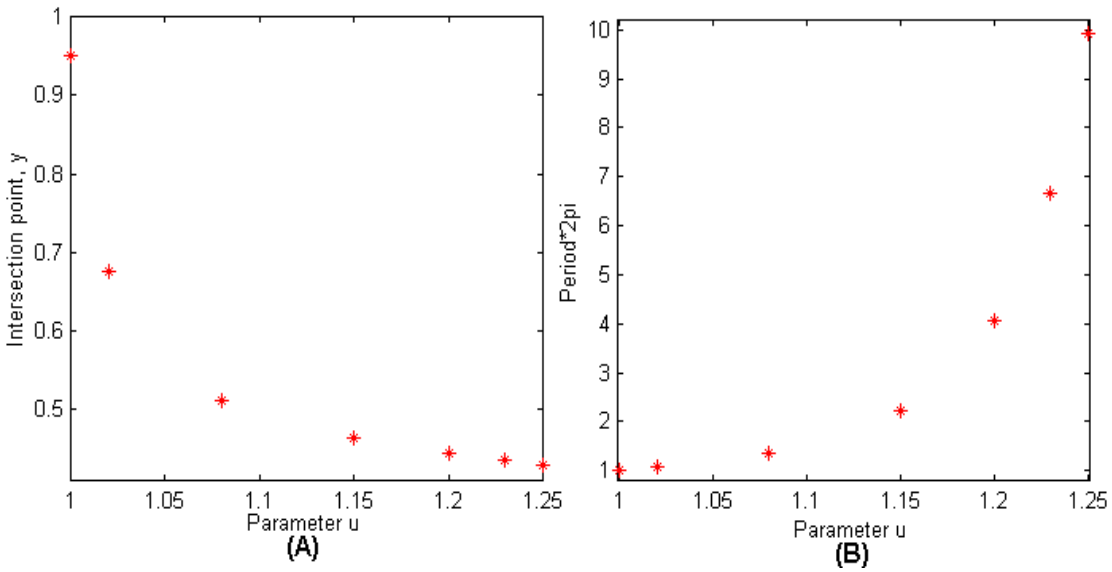

Figure 2.6: The intersection of the Poincaré section $\mathcal{P}$ by the periodic orbits and the period of each periodic orbit under variations in μ.

2.7 Discontinuities

The dynamical systems studied in previous sections were considered smooth; the vector field was assumed to be continuously differentiable and any change in the state variables had to satisfy the corresponding differential equation. The passive walker models that form the core of this thesis, however, are not smooth. Discontinuities in the vector field occur when the knees lock as well as when foot points contact and release from the ground. The latter events are also associated with discrete jumps in selected state variables. As the derivatives of the flow function as well as the vector field are involved in the stability calculations discussed previously as well as in the application of the Newton-Raphson method, it is clear that this theory must be modified to account for system discontinuities. Consider a solution trajectory based at an initial point $\mathbf{x}_0$ at time t_0 that transversally intersects a discontinuity surface, corresponding to the zero-level surface of some function $h(\mathbf{x})$, at the point $\mathbf{x}_{in}$ after elapsed time t_{in}. Suppose that the discontinuity is associated with a discrete jump, given by a jump mapping $\mathbf{g}$, such that $\mathbf{x}_{in}$ is mapped to the point $\mathbf{x}_{out} = \mathbf{g}(\mathbf{x}_{in})$. After the discrete jump, changes in the state variables are again governed by the vector field $\mathbf{f}$ for an elapsed time $t - t_{in} > 0$. See Figure 2.7 for a schematic of a discrete jump.

The expression

$$\mathbf{\Phi}^{vf}\left(\mathbf{g}\left(\mathbf{\Phi}^{vf}\left(\mathbf{x}_0, t_{in}\right)\right), t - t_{in}\right),$$

where $\mathbf{\Phi}^{vf}$ is the flow associated with the vector field $\mathbf{f}$, describes the corresponding solution trajectory. As in the case where the Poincaré section was introduced, the implicit function theorem can be used to show the existence of a unique function $\tau(\mathbf{x})$ for $\mathbf{x}$ near $\mathbf{x}_0$, such that

$$\tau(\mathbf{x}_0) = t_{in}, \tag{2.69}$$

$$h\left(\mathbf{\Phi}^{vf}\left(\mathbf{x}, \tau(\mathbf{x})\right)\right) = 0 \tag{2.70}$$

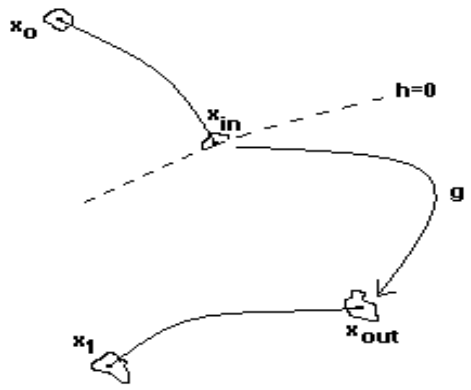

Figure 2.7: A schematic of a discrete jump.

and

$$\begin{aligned} D_{\mathbf{x}}t_{in} &= -\frac{D_{\mathbf{x}}h\left(\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,t_{in}\right)\right)}{D_{\mathbf{x}}h\left(\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,\tau\left(\mathbf{x}_0\right)\right)\right)\cdot\mathbf{f}\left(\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,t_{in}\right)\right)}\cdot\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,t_{in}\right) \\ &= -\frac{D_{\mathbf{x}}h\left(\mathbf{x}_{in}\right)}{D_{\mathbf{x}}h\left(\mathbf{x}_{in}\right)\cdot\mathbf{f}\left(\mathbf{x}_{in}\right)}\cdot\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,t_{in}\right), \end{aligned} \tag{2.71}$$

where the denominator is nonzero by the assumed transversality of the intersection with the discontinuity surface.

Now consider the function

$$\mathbf{\Phi}\left(\mathbf{x},t\right)=\mathbf{\Phi}^{vf}\left(\mathbf{g}\left(\mathbf{\Phi}^{vf}\left(\mathbf{x},\tau\left(\mathbf{x}\right)\right)\right),t-\tau\left(\mathbf{x}\right)\right) \tag{2.72}$$

corresponding to the composite flow function of the dynamical system including the discrete jump due to $\mathbf{g}$. The Jacobian of the composite flow function can be shown to be

$$\begin{aligned} \mathbf{D}_{\mathbf{x}}\mathbf{\Phi}\left(\mathbf{x}_0,t\right)=\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}^{vf}\left(\mathbf{x}_{out},t-t_{in}\right) &\cdot \left(\mathbf{D}_{\mathbf{x}}\mathbf{g}\left(\mathbf{x}_{in}\right)+\frac{\left(\mathbf{f}\left(\mathbf{x}_{out}\right)-\mathbf{D}_{\mathbf{x}}\mathbf{g}(\mathbf{x_{in}})\cdot\mathbf{f}\left(\mathbf{x}_{in}\right)\right)\cdot D_{\mathbf{x}}h(\mathbf{x_{in}})}{D_{\mathbf{x}}h(\mathbf{x_{in}})\cdot\mathbf{f}\left(\mathbf{x}_{in}\right)}\right) \\ &\cdot \mathbf{D}_{\mathbf{x}}\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,t_{in}\right), \end{aligned} \tag{2.73}$$

where $\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}^{vf}\left(\mathbf{x}_0,t_{in}\right)$ and $\mathbf{D}_{\mathbf{x}}\mathbf{\Phi}^{vf}\left(\mathbf{x}_{out},t-t_{in}\right)$ are obtained by solving the variational equations corresponding to the vector field $\mathbf{f}$ along the trajectory starting at $\mathbf{x}_0$ for an elapsed time t_{in} and starting at $\mathbf{x}_{out}$ for an elapsed time $t-t_{in}$, respectively.

Example 4 *To illustrate the validity of these expressions, return to the two-dimensional dynamical system considered in Example 3 and introduce a jump mapping*

$$\mathbf{g}\left(\mathbf{x}\right)=\begin{pmatrix} x_1 \\ 2\mu-x_2 \end{pmatrix}.$$

This corresponds to a discontinuity surface that is coincident with the Poincaré section introduced previously. The resulting system was simulated using the Matlab functions `ode45_moreDOFsystem` *and* `jump_in_y`*, shown in Appendix C. Specifically, Figure 2.8 shows a sample solution trajectory. As the Matlab code simultaneously solves the original differential equations as well*

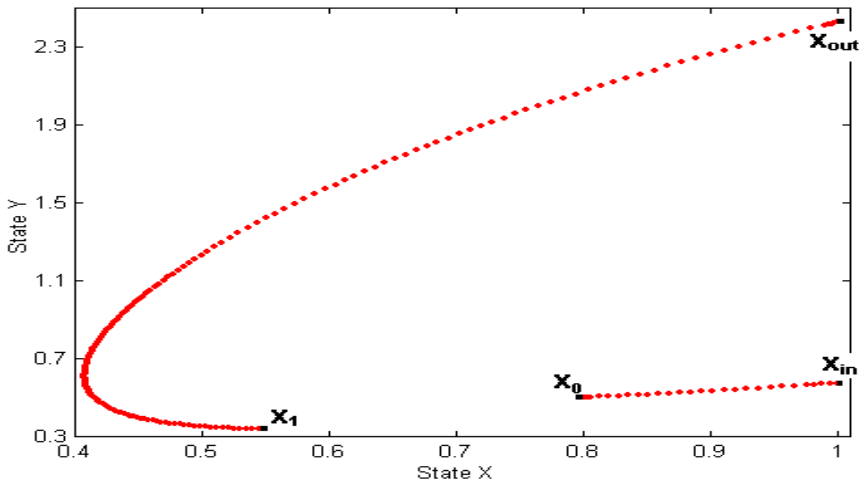

Figure 2.8: A sample solution trajectory including a discrete jump at a discontinuity surface.

as the variational equations, it is possible to compute the Jacobian of the composite flow function. By simulating the original dynamical system with two nearby initial conditions $\mathbf{x}_0$ *and* $\mathbf{x}_0 + \Delta\mathbf{x}$*, the above formula may be verified from*

$$\mathbf{\Phi}\left(\mathbf{x}_0 + \Delta\mathbf{x}, t\right) - \mathbf{\Phi}\left(\mathbf{x}_0, t\right) \approx \mathbf{D}_{\mathbf{x}}\mathbf{\Phi}\left(\mathbf{x}_0, t\right) \cdot \Delta\mathbf{x}, \tag{2.74}$$

where the approximation is valid only for sufficiently small $\Delta\mathbf{x}$*, but not so small that numerical accuracy in the integration of the differential equations becomes a problem.*

2.8 Bifurcations

The qualitative structure of a flow can change as parameters are varied. As seen in previous sections, fixed points and closed orbits can be created or destroyed, or their stability can change. These qualitative changes in the dynamics are called bifurcations, and the parameter values at which bifurcations occur are called bifurcation points. Bifurcation diagrams illustrate these important behavioral changes under variations in system parameters. Five basic types of bifurcations occur in the analysis of the passive walker models that will be discussed in subsequent chapters; namely, the saddle-node bifurcation, the pitchfork bifurcation, the Hopf bifurcation, the period-doubling bifurcation, and grazing bifurcations.

2.8.1 Saddle-node Bifurcation

Suppose that, as a system parameter μ approaches a critical value μ_0, for $\mu < \mu_0$, one originally negative eigenvalue of the Jacobian matrix of the vector field at an equilibrium point approaches zero and equals zero at μ_0. It follows that the conditions for the implicit function theorem are not satisfied at $\mu = \mu_0$ and that existence and uniqueness of a branch of equilibria near the equilibrium point at $\mu = \mu_0$ are not guaranteed. In fact, under the most general conditions, this bifurcation is associated with the existence of two branches of equilibria for $\mu < \mu_0$ and no equilibria for $\mu > \mu_0$. The two branches meet smoothly at the bifurcation point. For equilibria on the two branches near the bifurcation point, all but one

eigenvalue stay relatively unchanged, while the one eigenvalue increases toward zero on one branch and decreases toward zero on the other branch. From its visual appearance, this type of bifurcation is also known as a fold bifurcation or a turning-point bifurcation.

A similar bifurcation may occur when studying fixed points corresponding to periodic orbits. Here, two branches of periodic orbits exist for $\mu < \mu_0$ and no periodic orbits exists for $\mu > \mu_0$. Again, the two branches meet smoothly at the bifurcation point. For the periodic orbits on the two branches near the bifurcation point, all but one eigenvalue of the Jacobian of the Poincaré map stay relatively unchanged, while the one eigenvalue increases toward one on one branch and decreases toward one on the other branch.

2.8.2 Pitchfork Bifurcation

In physical problems with special types of symmetry (for example, the spatial reflection symmetry of a typical bipedal walker), the zero eigenvalue condition discussed previously may be associated with the existence of one branch of equilibria for $\mu < \mu_0$ and three branches for $\mu > \mu_0$. Of the three branches that exist when $\mu > \mu_0$, the outer ones meet smoothly at the bifurcation point, while the middle one meets the single branch that exists when $\mu < \mu_0$, smoothly at the bifurcation point. Again, all but one eigenvalue of the Jacobian of the vector field at the equilibria stay relatively unchanged, while the one eigenvalue decreases toward zero on the middle branch for $\mu > \mu_0$ and increases toward zero on the other three branches.

The scenario described here is for a supercritical pitchfork bifurcation. Alternatively, in a subcritical pitchfork bifurcation, the three branches of equilibria exist for $\mu < \mu_0$ with a single branch for $\mu > \mu_0$. Here, the single varying eigenvalue increases toward zero on the middle branch for $\mu < \mu_0$ and decreases toward zero on the other three branches.

This discussion again carries over to the study of periodic orbits. For example, a periodic orbit on one of the outer branches will, by symmetry, have a symmetrical companion on the other branch. For the passive walker models, a pitchfork bifurcation may correspond to the birth of walking motions that veer down the incline at an angle to the left or, by symmetry, to the right as a parameter is varied.

2.8.3 Hopf Bifurcation

The implicit function theorem only fails if an eigenvalue of the Jacobian of the vector field at an equilibrium point equals zero. If, instead, a complex conjugate pair of eigenvalues with negative real part for $\mu < \mu_0$ have zero real part for some critical parameter value μ_0, the implicit function theorem guarantees the existence of a unique branch of equilibria through this point. Nevertheless, under general conditions, it is possible to show that a branch of periodic orbits is born at the bifurcation point. Again, when the periodic orbits exist for $\mu < \mu_0$, the bifurcation is said to be supercritical, while it is called subcritical if the periodic orbits exist for $\mu > \mu_0$.

In the case of periodic orbits, a Hopf bifurcation is associated with the crossing of a complex conjugate pair of eigenvalues through the unit circle. Again, under general conditions, it can be shown that a branch of torus motions (periodic or quasiperiodic orbits) will be born.

2.8.4 Period doubling

A different type of bifurcation occurs when a single eigenvalue of the Jacobian of the Poincaré map at a fixed point corresponding to a periodic orbit crosses -1 at some critical parameter value μ_0. Here, a second branch of periodic orbits of twice the period of the original orbit are born. Again, the possibility of super- or subcritical bifurcation exists. In terms of the iterated Poincaré map $\mathbf{P}^2$ this corresponds to a pitchfork bifurcation of fixed points.

2.8.5 Grazing Bifurcation

If, for some critical parameter value μ_0, a periodic orbit intersects a discontinuity surface tangentially, the Jacobian of the Poincaré map fails to exist for the corresponding fixed point, since the term

$$D_{\mathbf{x}} h(\mathbf{x_{in}}) \cdot \mathbf{f}\left(\mathbf{x}_{in}\right) \tag{2.75}$$

that appears in the denominator in the computation of the Jacobian of the composite flow function $\mathbf{D_x \Phi}\left(\mathbf{x}_0, t\right)$, equals zero. Under general conditions, it can be shown that this condition is associated with rapid and possibly discontinuous changes in the eigenvalues for nearby parameter values resulting in rapid changes in local stability and system behavior.

2.9 Stabilizing control

In this section two control algorithms are proposed to control the local stability of a periodic orbit corresponding to an initially unstable walking motion of a walking mechanism. For the algorithms proposed here, discrete changes to the swing-foot ankle orientation will be made at specific moments during the gait cycle. These changes will be modeled as discrete jumps in suitable parameters that occur when the flow trajectory encounters a discontinuity surface, $\mathcal{P}_{control}$, locally described by an event function, $H_{control}$. Since the swing phase motion of the model walker will be assumed to be independent of the orientation of the swing-foot ankle, there is no change in the motion of the walker until the foot makes contact with the ground. Therefore, even though it would take some time to change the orientation of the ankle in an actual mechanism, it is allowable to represent the change as a discontinuous jump.

To determine the new orientation of the foot required to locally stabilize the model walker's motion, the difference between the actual state x at the discontinuity surface and a comparison state x^* must be found. Two types of control algorithm will be considered, distinguished by their definition of the comparison state. The comparison state with **reference feedback control** remains constant throughout the motion and is given by the intersection of the reference trajectory with the discontinuity surface. In contrast, with **delay feedback control**, the comparison state is given by the previous intersection of the discontinuity surface by the actual trajectory.

In order to analyze the controlled system consider the augmented state vector

$$\mathbf{y} = \begin{pmatrix} \mathbf{x} \\ \theta \\ \mathbf{x}^* \end{pmatrix}, \tag{2.76}$$

where $\mathbf{x}$ is a vector representing the actual state of the mechanism, θ is a vector of angles describing the orientation of the foot, and $\mathbf{x}^*$ is the comparison state. Away from the discontinuity surface, changes in $\mathbf{y}$ are given by the vector field

$$\mathbf{F}(\mathbf{y}) = \begin{pmatrix} \mathbf{f}(\mathbf{x}, \theta) \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{2.77}$$

since the ankle orientation and comparison state are assumed constant.

As was discussed previously, the discontinuity surface is locally described by the zero-level surface of an event function. The intersection of the reference trajectory with the discontinuity surface is given by the vector

$$\mathbf{y}^{ref} = \begin{pmatrix} \mathbf{x}^{ref} \\ \theta^{ref} \\ \mathbf{x}^{ref} \end{pmatrix}. \tag{2.78}$$

At the discontinuity surface, control is activated with a jump function, $\mathbf{g}_{control}(\mathbf{y})$, which can be represented as

$$\mathbf{g}_{control}(\mathbf{y}) = \begin{pmatrix} \mathbf{x} \\ \theta^{ref} + C(\mathbf{x} - \mathbf{x}^*) \\ \mathbf{x}^* \end{pmatrix} \tag{2.79}$$

for reference feedback control and

$$\mathbf{g}_{control}(\mathbf{y}) = \begin{pmatrix} \mathbf{x} \\ \theta^{ref} + C(\mathbf{x} - \mathbf{x}^*) \\ \mathbf{x} \end{pmatrix} \tag{2.80}$$

for delay feedback control. C is a matrix of gain parameters that describe the linear feedback between the current state $\mathbf{x}$ and the new ankle orientation.

Restrict attention to the case where control is applied only once per step. The Poincaré mapping for a system with control can be written as

$$\mathbf{P}(\mathbf{x}) = \mathbf{\Phi}(\mathbf{g}_{control}(\mathbf{y}), \tau(\mathbf{g}_{control}(\mathbf{y}))), \tag{2.81}$$

where $\mathbf{y}$ is on the discontinuity surface $\mathcal{P}_{control}$, $\mathbf{\Phi}$ is the flow of the vector field $\mathbf{F}$, and τ is the time it takes for the trajectory to come back to the discontinuity surface. In particular,

$$\mathbf{P}(\mathbf{y}^{ref}) = \mathbf{P}\begin{pmatrix} \mathbf{x}^{ref} \\ \theta^{ref} \\ \mathbf{x}^{ref} \end{pmatrix} = \mathbf{\Phi}(\mathbf{y}^{ref}, T) = \mathbf{y}^{ref}, \tag{2.82}$$

where T is the period of the reference trajectory. Supposing that the reference trajectory intersects $\mathcal{P}_{control}$ transversely, the stability of the reference trajectory is given by the Jacobian of the Poincaré mapping

$$\begin{aligned} D_{\mathbf{y}}\mathbf{P}(\mathbf{y}^{ref}) &= \left(Id - \frac{\mathbf{F}(\mathbf{y}^{ref}) \cdot D_{\mathbf{y}}\mathbf{H}_{control}(\mathbf{y}^{ref})}{D_{\mathbf{y}}\mathbf{H}_{control}(\mathbf{y}^{ref}) \cdot \mathbf{F}(\mathbf{y}^{ref})}\right) \\ &\quad \cdot D_{\mathbf{y}}\mathbf{\Phi}(\mathbf{y}^{ref}, T) \cdot D_{\mathbf{y}}\mathbf{g}_{control}(\mathbf{y}^{ref}). \end{aligned} \tag{2.83}$$

Assuming that the event function is independent of the ankle orientation θ and the comparison state $\mathbf{x}^*$, it follows that

$$D_{\mathbf{y}}\mathbf{H}_{control} = (\ D_{\mathbf{x}}\mathbf{H}_{control}\ \ \mathbf{0}\ \ \mathbf{0},\ ) \tag{2.84}$$

$$D_{\mathbf{y}}\mathbf{\Phi}(\mathbf{y}^{ref}, T) = \begin{pmatrix} A & B & \mathbf{0} \\ \mathbf{0} & Id & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & Id \end{pmatrix}, \tag{2.85}$$

$$D_{\mathbf{y}}\mathbf{g}_{control}(\mathbf{y}^{ref}) = \begin{pmatrix} Id & \mathbf{0} & \mathbf{0} \\ C & \mathbf{0} & -C \\ \mathbf{0} & \mathbf{0} & Id \end{pmatrix} \tag{2.86}$$

for reference feedback, and

$$D_{\mathbf{y}}\mathbf{g}_{control}(\mathbf{y}^{ref}) = \begin{pmatrix} Id & \mathbf{0} & \mathbf{0} \\ C & \mathbf{0} & -C \\ Id & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{2.87}$$

for delay feedback. The goal of subsequent analysis is to choice gain parameters in such a way as to improve the stability of a walker's motion.

Chapter 3

3D Passive walker development

3.1 Background

3.1.1 Toys and Walking machines/robots

The first walking machines were developed as toys. There were wind-up toys, toys that could walk down an incline with no energy supplied, and toys that could be pulled along a flat surface by a mass hanging over the edge of the surface connected to the toy by a string. Most of the walking machines/robots that have been developed are able to move due to actuation, thus they are not passive. However, what we learn from a study of passive walkers can be used to make these actuated machines more anthropomorphic or more energy efficient.

Walking Toys

The United States patent, US 376,588 from 1888 shows a walking toy proposed by George Fallis [12]. This toy was meant to simulate the human form by using a combined pendulum and rocker construction so that if placed on an incline plane it would step out and walk down the plane. The legs were constructed as pendulum rods that hung in a parallel plane. The legs were the same length, had no knees, and were made to pivot and rock about a common axis. The feet were turned outward, from the heel out toward the toe, like the human foot. The foot was also curved, to allow for the walker to rock in the lateral direction. When placed on an inclined plane the walker would be given a nudge on its side to get it started. It would lean to the side and pick the other foot off the ground. With the help of gravity this foot would move forward and make contact with the ground. Subsequently the walker would pick up the other foot and take a step forward. This pattern continued as the walker proceeded down the ramp. This idea for a toy was also used for quadrupedal walkers. The wire bipedal walker model that Fallis proposed was constructed in 2002 by Thomas Platzer of Cornell University [10].

To improve the aforementioned toy, a London patent London 7,453 from 1912 was proposed by Balduin Bechstein and Paul Uhlig to make the gait of the toy more human like [4]. This walker still had no knees and the legs rotated about the same axis, but the legs were hinged at the shoulders. The feet were not slanted outward, but they had a sole and a heel like that of a boot. The feet had ridges in different locations to generate the side-to-side

motion. With this new approach, the walker walked down the inclined surface with less swaying.

One other patent of interest (US 1,207,464) for a walking toy was proposed by Harvey Allison in 1916 [3]. The patent was for a toy that could walk down a tight rope that was positioned on an incline. This walker had arms that extended out from its sides holding sticks with masses attached to the ends. This gave the passive walking toy a low center of mass that gravity would be acting on. The toy also had a spring attached between the torso and the legs to keep the torso in the upright position and to help the legs switch position by flipping a lever.

More recently, passive walkers have been made out of tinkertoys. These toys were developed by Michael Coleman at Cornell University [8, 29]. The tinkertoy walker is bipedal, knee-less, and could walk down an incline using only gravity to propel it. This walker, as with the above toy, has masses that project out in the lateral direction it to move the center of mass and make the toy's motion more stable. Due to the construction of this toy, its motion was statically unstable and it could not stand still.

Vintage wind-up and battery operated walking toys from the 1950's to today can be seen on the 'Toy Robots' web page [30]. Wooden toys that walk down an incline and have two feet that remain one in front of the other for all time can be seen at [39]. Some other toys that walk down an incline can be seen at Rampwalkers.net. Most of the toys on this internet site were developed by Louis Marx & Company [36].

Walking Machines

Blueprints for the first walking machines can be found dating back to the eighteenth century. The idea for a mechanical horse was patented in 1893 by L. A. Rygg, but was never built. The first bipedal walking machine was designed in 1893, by Georges Moore, and was called *The Steam Man*. In the 1960's General motors developed a powerful exoskeleton called *Hardyman* that could in principle be used in heavy applications. However, a disadvantage of this robust machine was that the actuation and servo control methods made it impractical [7].

In the late sixties and early seventies Ichiro Kato's team at Waseda University of Japan started developing walking machines that became progressively more complex. Beginning with two machines that had pressure sensors on the soles of their feet to aid in balance and could ascend or descend stairs as well as turn, a significant advance was made in 1973 when the first full scale anthropomorphic robot was built. Apart from its abilities to walk, although quite slow at forty-five seconds a step, its features included a vision and conversation system. It was also able to judge the distance to objects and grasp them. In 1982, a machine was designed that was able to walk backwards and in 1985 Kato's team developed a walker that was capable of taking a step in 13 seconds. However, the Humanoid Robot from Honda Motor Co. [17] is the most notably anthropomorphically advanced walking machine. In 1996 Honda built a robot that looks like a slow-moving astronaut. It has the abilities to walk up and down stairs, open a door, pick up and put down objects, and push a car.

There are many interesting research projects conducted at universities that deal with robotics and walking machines, such as Delft University of Technology [10, 11], MIT [27], Waseda University [6], and Jouhou System Kougaku Laboratory in Japan [20]. *The Walking*

Machine Catalogue [5] also contains an extensive list of different walking machines.

3.1.2 Devices to Aid in Walking

Early research into bipedal walking machines was for researching prosthetic and orthotic aids. The first powered prosthesis were considered by N. A. Bernstein in 1948. This above-the-knee prosthetic device had powered knee joints, but it was not fully developed. In the 1950's, Milikan and Eikan suggested a powered exoskeleton which would not only provide orthotic aid but could also be useful in carrying out heavy work. It was ultimately only developed as a design aid, but was followed by General Motors *Hardyman*, as previously discussed [7].

As cited by Yobotics Inc. [40], today there are 1.7 million people in the United States suffering from weakness in their lower extremities. Debilitation of this nature can have many causes including cerebral vascular trauma (stroke), post-polio syndrome, multiple sclerosis, muscular dystrophy, and aging. In addition 200,000 Americans have no control of their lower limbs due to spinal cord injury and are therefore confined to a wheelchair. There are many devices out there to help those with this immobilizing conditions. One newer idea by Yobotics Inc. [40] is a powered, wearable device called the RoboWalker. If successful, the RoboWalker will augment or replace muscular functions of the lower extremities.

3.1.3 Passive walkers

The term 'passive walker' refers to a mechanism that does not require outside control or actuation to maintain gait. The only source of energy is due to gravity or a torsional spring. The first passive walker was introduced by McGeer [24, 25]. He realized that with the appropriate geometry and mass distribution of the walker, and the inclination of the plane the walker is walking on, the walker would exhibit stable gait. Another type of walking that is related to passive walking is ballistic walking, studied by Mochon [28] and Formal'sky [13]. This type of walking was usually studied on flat/level ground. It is called ballistic because no actuation is used during the swing phase of the leg, meaning that the leg swings freely, like a ballistic pendulum. The reasoning for this walking came from biological studies that show little muscle actuation is needed during the swing phase [18, 38]. To ensure periodic gait, an impulse was added to the walker in its double support phase.

McGeer realized that an impulse does not need to be added to the system. Energy is lost during certain stages of the walking cycle, for example, when the ground is contacted and when the knee locks. To replenish this energy loss, McGeer's model walks down an inclined plane. This allows gravity to act on the walker and add energy to the system so no actuation is required.

3.1.4 Passive walker modifications

The first planar passive walker developed by McGeer [24] had curved feet, no knees, and legs connected by a single hinge joint. With this configuration the walker was constrained from out of plane motion, thus preventing lateral dynamics. McGeer started with a walker that had a very large foot radius, suggestive of a walking wheel. He then reduced the radius of

the foot, resulting in the first straight-legged bipedal passive walker. Since the legs were the same length, foot scuffing occurred at mid swing. The legs were made to contract to prevent this scuffing. Later McGeer added a knee [25] to alleviate this problem.

A three-point-mass planar passive walker was studied by Ruina's group at Cornell University, with one mass representing the torso and the other two modeling the leg masses [14]. Ruina's group was also able to implement planar passive walkers in actual mechanisms with much success, see [15]. To eliminate lateral dynamics, the walkers were constructed with legs in pairs that were physically constrained to move together.

Next, work was done to extend these planar walker configurations into three dimensions while retaining stable gait. McGeer tried extending his walker into 3D, but it resulted in the systems dynamics being unstable [26]. Other researchers (Adolfsson, Dankowicz & Nordmark [2], Dankowicz, Adolfsson, & Nordmark [9], and Piiroinen, Dankowicz, & Nordmark [32]) accomplished the 3D configuration with much success. Kuo [21] added active control to the system to stabilize the swaying motion of the walker. The passive wheel walker described by McGeer was developed as the Tinkertoy walker [8], as discussed previously. This walker was then modeled and its motion was proven to be stable [29]. However, the model failed to be anthropomorphic with extra masses protruding laterally from it.

It was proven by Dankowicz and collaborators [2, 9] that the foot radius could be shrunk to zero. Instead of a single toe point, the feet were subsequently developed with two points at the end of a toe line. With infinitely long toe lines, the walkers motion was constrained to 2D. The motion of this walker could also be found with a walker model with zero hip width but finite toe width, referred to as an extended 2D walker model. However, in the latter case the addition of more degrees-of-freedom allowed for inherently out-of-plane instabilities. Subsequent work was preformed to have the model more closely resemble the geometry and mass distribution of a human [35, 34, 32]. Up to this point, the hip joints were represented by revolute joints, yielding one degree of freedom per hip. In this thesis, we revisit these walker models and attempt to extend the analysis to a walker with two more degrees of freedom in the hip, modeled with ball joints.

An important concept for obtaining the results discussed in the previous paragraph was a continuation method based on the Newton-Raphson method discussed previously for doing parameter studies. The initial guess would be obtained from a known solution for a near-by parameter value.

Parameter studies can than be performed to determine the existence, stability, and bifurcation of recurrent gait under changes of inclination. In addition, a continuation method could be used to reach a desirable selection of model parameters, as was done when reducing the foot radius to zero. When performing a parameter study, it is important to study unstable as well as stable periodic orbits. After all, there is a possibility that a bifurcation to a stable solution from an unstable solution could occur. It is also possible that some unstable solutions can be stabilized using feedback control as discussed previously. A discussion of these issues will resume in subsequent chapters.

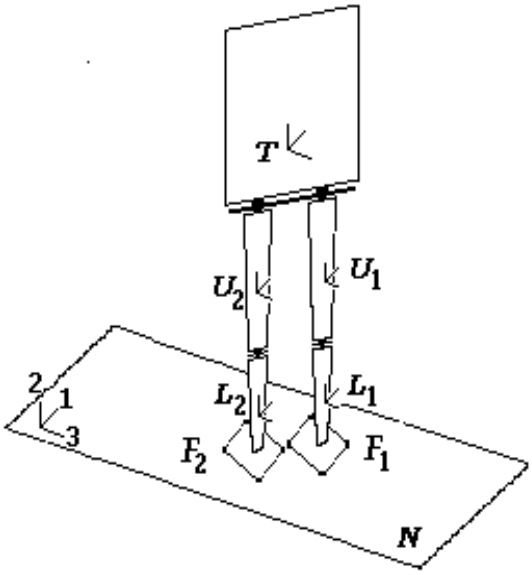

Figure 3.1: A model of a 3D passive walker.

3.2 3D walker model

The first 3D walker model discussed in this thesis is related to the model studied by Piiroinen [33]. An important difference to note is the specified orientation of the reference frames.

3.2.1 Kinematics

The 3D walker consists of five rigid bodies connected by revolute joints: the upper body/torso, T, two upper legs, U_1, U_2, and two lower legs, L_1, L_2; and two rigid bodies are connected to the lower legs by ball joints, F_1, F_2. The ground is assumed to be fixed relative to the inertial reference frame N. This model can be seen in Figure 3.1.

To each rigid body B associate a set of three orthonormal body-fixed basis vectors $\mathbf{b}_1, \mathbf{b}_2$, and $\mathbf{b}_3$, known as a reference triad, and a body-fixed reference point B located at the center of mass of the rigid body. The position of a point P, relative to the rigid body B, will be represented by the position vector $\mathbf{r}^{BP} = x\mathbf{b}_1 + y\mathbf{b}_2 + z\mathbf{b}_3$. The default orientation of the reference triad is such that the $\mathbf{b}_1$-direction is pointing leftward, the $\mathbf{b}_2$-direction is pointing superiorly (upward) and the $\mathbf{b}_3$-direction is pointing anteriorly (forward).

The upper body of the walker consists of only a torso. The center of mass of the torso is located at the center point of the line connecting the hip joints, such that

$$\mathbf{r}^{NT} = q_1\mathbf{n}_1 + q_2\mathbf{n}_2 + q_3\mathbf{n}_3, \tag{3.1}$$

where the reference point N lies on the plane. Furthermore, the torso has three rotational degrees-of-freedom relative to the inertial reference frame as described by the 3-2-1 sequence of Euler angles q_6, q_5, and q_4. For later reference, denote by $\tilde{t}$ the reference triad obtained after the second Euler rotation.

The upper legs U_1 and U_2 are connected to the torso by two revolute joints at the hip with an axis of rotation parallel to the $\tilde{\mathbf{t}}_1$ and angular coordinates q_7 and q_9, respectively.

Similarly, the lower legs L_1 and L_2, are connected to the upper legs by revolute joints at the knee with an axis of rotation parallel to the $\tilde{\mathbf{t}}_1$ and angular coordinates q_8 and q_{10} respectively. Finally, the feet F_1 and F_2 are connected to the lower legs by ball joints at the ankle with orientation given by a 3-2-1 sequence of Euler angles $\theta_1, \theta_2, \theta_3$ and $\theta_4, \theta_5, \theta_6$, respectively.

To complete the description of the kinematics we must describe the velocity and angular velocities for each body. Starting with the torso, the linear velocity of the reference point T relative to the inertial reference frame is given by

$$^N\mathbf{v}^T = u_1\mathbf{n}_1 + u_2\mathbf{n}_2 + u_3\mathbf{n}_3. \tag{3.2}$$

Similarly, the angular velocity of T relative to the inertial reference frame is given by

$$^{\mathbf{n}}\omega^{\mathbf{t}} = u_4\tilde{\mathbf{t}}_1 + u_5\tilde{\mathbf{t}}_2 + u_6\tilde{\mathbf{t}}_3. \tag{3.3}$$

Finally, angular velocities of the upper and lower legs relative to the torso and upper legs, respectively, are given by

$$^{\tilde{\mathbf{t}}}\omega^{\mathbf{u}_1} = u_7\tilde{\mathbf{t}}_1 \tag{3.4}$$

$$^{\tilde{\mathbf{t}}}\omega^{\mathbf{u}_2} = u_9\tilde{\mathbf{t}}_1 \tag{3.5}$$

and

$$^{\mathbf{u}_1}\omega^{\mathbf{l}_1} = u_8\tilde{\mathbf{t}}_1 \tag{3.6}$$

$$^{\mathbf{u}_2}\omega^{\mathbf{l}_2} = u_{10}\tilde{\mathbf{t}}_1. \tag{3.7}$$

It is now possible to write the kinematic differential equations (KDE) relating $\dot{q}_i$ to u_i by

$$\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} = \begin{bmatrix} \cos(q_5)\cos(q_6) & -\sin(q_6) & \cos(q_5)\sin(q_6) \\ \cos(q_5)\sin(q_6) & \cos(q_6) & \sin(q_5)\sin(q_6) \\ -\sin(q_5) & 0 & \cos(q_5) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}, \tag{3.8}$$

$$\begin{bmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \tan(q_5) \\ 0 & 1 & 0 \\ 0 & 0 & \mathrm{cosec}(q_5) \end{bmatrix} \begin{bmatrix} u_4 \\ u_5 \\ u_6 \end{bmatrix}, \tag{3.9}$$

and $\dot{q}_i = u_i, i = 7, 8, 9, 10$.

3.2.2 Forces and Torques

External Forces

The walker is considered to be passive, that is, there is no energy supplied to it other than the energy gained by its motion in the gravitational field. The direction of the gravitational field can be represented as

$$\mathbf{g} = -cos(\alpha)\mathbf{n_2} + sin(\alpha)\mathbf{n_3}, \tag{3.10}$$

where α is the inclination of the plane. As α is increased from zero there is a positive component in the $\mathbf{n_3}$ direction, allowing for motion down the incline.

Energy is dissipated as the walker is moving, for example, through toe contact with the ground and forces or torques that occur in the knee and hip joints. A balance is required between the energy lose and the energy gained by the downhill motion of the walker. These springs and dampers are turned on and off depending on the configuration of the walker, meaning depending on when a discrete state changes, such as knee locking or ground contact.

Ground Contact

The walker model's feet interact with the ground through four toe points, denoted by $P_{i,j}$, representing the j^{th} toe point on the i^{th} leg, where $j = 1, 2, 3, 4$ and $i = 1, 2$. The position of a toe point from the inertial reference point N is

$$P_{i,j} = p_{i,j,1}\mathbf{n}_1 + p_{i,j,2}\mathbf{n}_2 + p_{i,j,3}\mathbf{n}_3. \tag{3.11}$$

When a toe comes in contact with the plane, linear forces are introduced that depend on the position relative to the initial contact point and speed of the toe point. These contact forces are modeled by springs and dampers. The springs and dampers are deactivated when the toe point leaves the ground. It is important to note that the vertical component of viscous damping is only applied when the toe point moves into the plane. See Figure 3.2A for a schematic of the spring and damper configuration with the ground and toe.

The contact forces can be represented as

$$F_{i,j} = F_{i,j,1}\mathbf{n}_1 + F_{i,j,2}\mathbf{n}_2 + F_{i,j,3}\mathbf{n}_3, \tag{3.12}$$

where

$$F_{i,j,1} = \begin{cases} -k_1(p_{i,j,1} - s_{i,j,1} - q_1) - c_1\dot{p}_{i,j,1} & p_{i,j,2} \leq 0 \\ 0 & p_{i,j,2} > 0 \end{cases} \tag{3.13}$$

$$F_{i,j,2} = \begin{cases} -k_2 p_{i,j,2} - c_2\dot{p}_{i,j,2} & p_{i,j,2} \leq 0, \dot{p}_{i,j,2} \leq 0 \\ -k_2 p_{i,j,2} & p_{i,j,2} \leq 0, \dot{p}_{i,j,2} > 0 \end{cases} \tag{3.14}$$

$$F_{i,j,3} = \begin{cases} -k_3(p_{i,j,3} - s_{i,j,3} - q_3) - c_3\dot{p}_{i,j,3} & p_{i,j,2} \leq 0 \\ 0 & p_{i,j,2} > 0 \end{cases} \tag{3.15}$$

where

$$(s_{i,j,1} + q_1)\mathbf{n}_1 + (s_{i,j,3} + q_3)\mathbf{n}_3 \tag{3.16}$$

describes the position of the initial contact point of the toe with the plane relative to the inertial reference point. The variables $s_{i,j,k}$ are governed by the differential equations

$$\dot{s}_{i,j,k} = -\dot{q}_k \tag{3.17}$$

and have discontinuous jumps when a foot point makes ground contact. These variables will be denoted as s_1 through s_{16} rather than $s_{i,j,k}$.

Knee Extension

The knees have springs and dampers, located at the revolute joint between the upper and lower leg, to limit hyper-extension of the knee. To mimic the prevention of collapse of the knee by muscles in human walking [18], our model allows a small amount of hyper-extension during stance [1]. The springs and dampers are activated when the knee is straight and remains activated so as long as the knee is in hyper-extension. See Figure 3.2B for a schematic of the spring and damper configuration in the knee joint. The corresponding torque on the lower limb is shown as

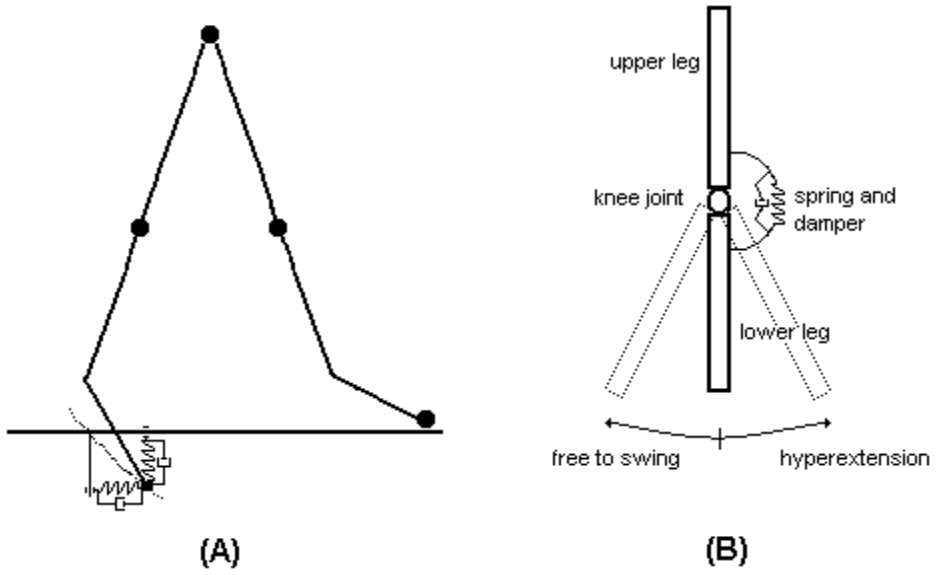

Figure 3.2: Spring and Damper configuration for (A) ground contact and (B) the knee joint. In (A) only one toe point is illustrated and the dotted line is the path the toe takes after it has penetrated the ground.

$$\mathbf{T}_{\text{left knee}} = \begin{cases} (k_{\text{knee}} \cdot q_8 + c_{\text{knee}} \cdot \dot{q}_8)\tilde{\mathbf{t}}_1 & q_8 \leq 0 \\ \mathbf{0} & q_8 > 0 \end{cases} \tag{3.18}$$

$$\mathbf{T}_{\text{right knee}} = \begin{cases} (k_{\text{knee}} \cdot q_{10} + c_{\text{knee}} \cdot \dot{q}_{10})\tilde{\mathbf{t}}_1 & q_{10} \leq 0 \\ \mathbf{0} & q_{10} > 0. \end{cases} \tag{3.19}$$

Torso Rotation

The center of mass of the torso is located at the center of the hip axis. An upright torso, above the hip line, is not studied here because it would require the addition of active control to maintain the upright position, see [16]. For a walking motion close to 2D the rotation of the torso about the hip axis decouples from the motion of the rest of the walker. To prevent the free rotation of the torso about the hip axis damping is added given by

$$\mathbf{T}_{torso} = c_{hip} \cdot (\dot{q}_7 + \dot{q}_9 - 2\dot{q}_4)\tilde{\mathbf{t}}_1, \tag{3.20}$$

$$\mathbf{T}_{leftleg} = -c_{hip} \cdot (\dot{q}_7 - \dot{q}_4)\tilde{\mathbf{t}}_1, \tag{3.21}$$

and

$$\mathbf{T}_{rightleg} = -c_{hip} \cdot (\dot{q}_9 - \dot{q}_4)\tilde{\mathbf{t}}_1. \tag{3.22}$$

Table 3.1: Spring and Damper Parameter values

Location	Spring	Damper
hip	0.00	0.04
knee	500	20
$\text{incline}_{1,3}$	2500	250
incline_2	3800	250

Table 3.2: Description of state variables

Label	Descriptions
q_1, q_2, q_3	Position (side, up, forward) of reference point T
q_4, q_5, q_6	Angles of torso (torso tilt, heading, sway)
q_7, q_9	Left and right hip angles (respectively)
q_8, q_{10}	Left and right knee angles (respectively)
s_1, s_3, s_5, s_7	Initial toe contact points for left foot in 1-direction
s_2, s_4, s_6, s_8	Initial toe contact points for left foot in 3-direction
$s_9, s_{11}, s_{13}, s_{15}$	Initial toe contact points for right foot in 1-direction
$s_{10}, s_{12}, s_{14}, s_{16}$	Initial toe contact points for right foot in 3-direction
u_1, u_2, u_3	Velocity of reference point T
u_4, u_5, u_6	Angular velocities of torso
u_7, u_9	Left and right hip angular velocities (respectively)
u_8, u_{10}	Left and right knee angular velocities (respectively)
$\theta_3, \theta_2, \theta_1$	Angles of left ankle
$\theta_6, \theta_5, \theta_4$	Angles of right ankle

Table 3.1 contains values of the springs and dampers used for modeling the ground contact, knee joints, and hip joints. For a discussion of the effects of using different spring and damper values, see [1]. For a more detailed discussion of the derivation of the equations of motion, the variational equations, (discussed in Section 2.6) and the modeling of the forces and torques needed to describe the dynamics of this 3D passive walker, see Piiroinen [33]. Also, see Table 3.2 for a description of the state variables introduced in the past two sections and see Table 3.3 for a description of the parameters used to describe the center of mass location for each body, its mass, its dimensions, its moment of inertia, etc. All quantities are measured in SI units.

3.2.3 Phases of the Gait Cycle

When we refer to walking we mean a gait where at least one foot is on the ground at any given time. A gait cycle can be divided into two phases: the single-support phase and the double-support phase. They are pretty self-explanatory: if only one foot is on the ground, it is in the single-support phase, but if both feet are touching the ground, it is in the double-support phase. During the single-support phase, the leg that is in contact with the ground is considered the stance leg and the other is called the swing leg.

The gait can be broken up into important discrete events, such as when a toe point comes

Table 3.3: Parameter description

Body part/Number	Descriptions
Torso	
1,2,3	center of mass position
20-25	mass moment of inertia
50	mass
Upper legs	
6-11	center of mass position
26-37	mass moment of inertia
51,52	mass
12,13	length
multicolumn2lLower legs	
14-19	center of mass position
38-49	mass moment of inertia
53,54	mass
105,106	length
Hips	
4,5	position of hip joint
66-68	stiffness and damping
Knees	
69,70	state
63-65	stiffness and damping
Toe Points	
71-102	position and state
Ground	
56	inclination
57-62	stiffness and damping
Other	
103,104	Poincaré section state
55	gravity

in contact with the ground or leaves the ground, and when the knee joints enter and leave hyper-extension. In Table 3.4, we can see the discrete event sequence for a typical step of a passive 3D walker. The X's mark when a toe point is in contact with the ground or when a knee is hyper-extended. Small changes in this event sequence are possible for every walking motion. Knee stretching may occur in a different order or the toe points may lift off or be put down in a different order. Also, in Table 3.4 you can see which leg is the swing or stance leg. Both legs are not listed for each event, but you can tell what the other leg is doing by looking at what support phase the walker is in.

Initially, the walker is in a single-support phase, with the right leg as the stance leg and the left leg as the swing leg. The left leg is in the process of swinging forward and knee stretch subsequently occurs. The leg continues to swing forward until its heel toe points touch the ground and the walker enters the double-support phase. The left knee releases as the foot rolls completely on the ground, following which the right foot's toes start to release from the ground. During this process, there is a switch between which leg is the swing and stance leg, as the weight of the walker switches from the right to left foot and the walker leans slightly forward onto the left foot. The right foot's toes subsequently lift off the ground, following which the right knee releases. The walker is now again in the single-support phase. Both knees are free as the weight of the walker moves forward on the left leg and the right leg swings forward. The left knee stretches and its heel toe points begin to lift off of the ground. At this point the walker has completed a stride and begins to have the same motion as just described starting with the opposite leg.

3.3 New 3D walker model

3.3.1 Kinematics

A novel contribution to this thesis is an attempt to extend the previous analysis to a new 3D walker model with two additional degrees-of-freedom in each hip. This increases the number of state variables from 42 to 50, see Table 3.5 for a description. For convenience the angular coordinates describing the upper leg and foot orientation are expressed relative to the Newtonian reference frame. In addition to the forces and torques discussed previously, springs and dampers are introduced to the hip joints to compensate for an increase in mobility. These springs and dampers are used to return the leg and foot back to the sagittal plane. This increases in the number of parameters from 106 to 110, see Table 3.6 for a description of the parameters and Table 3.7 for the specific spring and damper constants used.

3.4 Software and Code

The equations of motion and variational equations for the passive walker models discussed in this thesis were generated by Petri Piiroinen using the Maple package Sophia (see Lesser [23] and Lennartsson [22]). The Maple/Sophia code for the 3D walker model with one degree-of-freedom in each hip can be found in [33] and the code for the 3D walker model with more degrees-of-freedom in each hip can be found in Appendix D.

Table 3.4: Discrete event sequence for one step for a passive walker

Left Toes				Knees		Right Toes					
1	2	3	4	L	R	4	3	2	1	Leg	Phase
					X			X	X	left swing	single-support
				X	X			X	X	left swing	single-support
			X	X	X			X	X	left swing	double-support
		X	X	X	X			X	X	left swing	double-support
	X	X	X		X			X	X	left stance	double-support
X	X	X	X		X			X	X	left stance	double-support
X	X	X	X		X				X	left stance	double-support
X	X	X	X		X					left stance	single-support
X	X	X	X							left stance	single-support
X	X	X	X	X						right swing	single-support
X	X		X	X						right swing	single-support
X	X			X						right swing	single-support
X	X			X	X					right swing	single-support
X	X			X	X		X			right swing	double-support
X	X			X	X	X	X			right swing	double-support
X	X			X		X	X			right stance	double-support
X	X			X		X	X		X	right stance	double-support
X	X			X		X	X	X	X	right stance	double-support
	X			X		X	X	X	X	right stance	double-support
				X		X	X	X	X	right stance	single-support
						X	X	X	X	right stance	single-support
					X	X	X	X	X	right stance	single-support
					X		X	X	X	right stance	single-support

Table 3.5: State variable description

Label	Descriptions
q_1, q_2, q_3	Position of reference point T
q_4, q_5, q_6	Angles of torso
q_7, q_8, q_9	Left hip angles
q_{11}, q_{12}, q_{13}	Right hip angles
q_{10}, q_{14}	Left and right knee angles (respectively)
s_1, s_3, s_5, s_7	Initial toe contact points for left foot in x-direction
s_2, s_4, s_6, s_8	Initial toe contact points for left foot in y-direction
$s_9, s_{11}, s_{13}, s_{15}$	Initial toe contact points for right foot in x-direction
$s_{10}, s_{12}, s_{14}, s_{16}$	Initial toe contact points for right foot in y-direction
u_1, u_2, u_3	Velocity of reference point T
u_4, u_5, u_6	Angular velocities of torso
q_7, q_8, q_9	Left hip angular velocities
q_{11}, q_{12}, q_{13}	Right hip angular velocities
q_{10}, q_{14}	Left and right knee angular velocities (respectively)
$\theta_3, \theta_2, \theta_1$	Angles of left ankle
$\theta_6, \theta_5, \theta_4$	Angles of right ankle

Table 3.6: Parameter description

Body part/Number	Descriptions
Torso	
1,2,3	center of mass position
22-27	mass moment of inertia
52	mass
Upper legs	
6-11	center of mass position
28-39	mass moment of inertia
53,54	mass
12,13	length
Lower legs	
14-19	center of mass position
40-51	mass moment of inertia
55,56	mass
20,21	length
Hips	
4,5	position of hip joint
68-74	stiffness and damping
Knees	
75,76	state
65-67	stiffness and damping
Toe Points	
77-108	position and state
Ground	
58	inclination
59-64	stiffness and damping
Other	
109,110	Poincaré section state
57	gravity

Table 3.7: Spring and Damper Parameter values

Type	Location	x	y	z
Spring		Values		
	hip	0.0	50	50
	knee	500	0.0	0.0
	incline	2500	3800	2500
Damper		Values		
	hip	0.004	50	50
	knee	20	0.0	0.0
	incline	250	250	250

Chapter 4

Analysis of older 3D model

A novel contribution of this thesis is the demonstration of periodic motions for a new 3D walker model with three degrees-of-freedom in each hip. Here we propose to achieve this goal through the use of a parameter-continuation method based on the Newton-Raphson algorithm. As the success of the Newton-Raphson algorithm is highly dependent on a good initial guess, the purpose of this chapter is to find periodic walking motions that could be used to start the continuation method. Specifically, the construction proposed here relies on walking motions found for an extended 2D walker model that can exhibit purely 2D motion without being physically constrained. In particular, this is achieved with a 3D walker model that has zero hip width, symmetric feet, and is symmetric about the sagittal plane. In order to find a planar motion for an extended 2D walker, a 3D model studied previously by Piiroinen [33] was used. Also in this chapter, low-cost control algorithms are implemented to stabilize an otherwise unstable 3D walking motion.

4.1 Interpreting numerical results

When studying periodic orbits, it is convenient to introduce a Poincaré section and an associated Poincaré map, as discussed in Section 2.5. There are two distinct Poincaré surfaces considered in the analysis of the passive walkers in this chapter, namely $q_7 = 0$ and $q_9 = 0$, respectively, where q_7 and q_9 correspond to the orientation of the upper left and right limbs relative to the torso. The existence and stability of the periodic orbits will be considered in terms of the existence and stability of fixed points of the Poincaré mapping corresponding to the surface $q_7 = 0$.

The local stability of a periodic gait is found by calculating the eigenvalues of the Jacobian of the Poincaré mapping (see section 2.5). As long as the magnitude of the eigenvalues is less than one, the motion is stable. Similarly, in the case of an unstable gait, the eigenvector associated with an eigenvalue greater than one gives qualitative information about the direction of the instability.

By construction, there will always be two eigenvalues of the Jacobian equal to one. They correspond to perturbations made in the q_1 and q_3 directions . Similarly, there will always be a certain number of eigenvalues that equal zero that correspond to perturbations in the contact point coordinates for toe points that are initially off the ground as well as in the

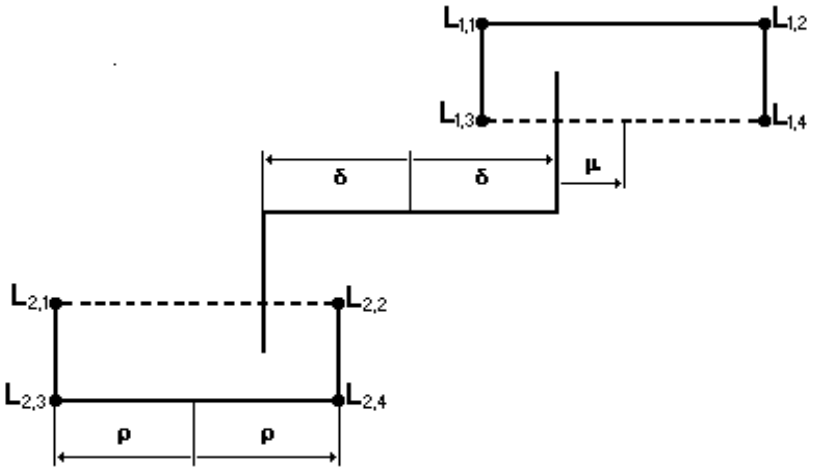

Figure 4.1: The geometry of a walker with asymmetric feet, as seen from above.

direction of the vector field at the Poincaré section.

4.2 Continuation from 3D to extended 2D walker model

This section presents the results of several parameter studies aimed at locating walking motions for an extended 2D walker. To find an extended 2D walking motion, the hip width must be reduced to zero and the feet must be made symmetric about the sagittal plane. As a starting point of this analysis, the 3D passive walker studied previously by Piiroinen was used [33]. Piiroinen's model is a 3D passive walker that has two toe points and two heel points per foot separated by 15 cm in the lateral direction but not symmetric with respect to the legs. Moreover in Piiroinen's model, the single-degree-of freedom-hip joints are separated by 20 cm in the lateral direction.

4.2.1 Hip to zero

A schematic of the walker's geometry as seen from above is shown in Figure 4.1. The reduction of δ from 10 cm to 0 cm was achieved by small decrements in hip width and the application of the Newton-Raphson method using the previously found solution as the initial guess. The results can be seen in Figure 4.2(A). Figure A1 shows a plot of the value of q_9 at the $q_7 = 0$ Poincaré section under variations in hip width. This plot also shows which hip widths correspond to stable motion (crosses) and which correspond to unstable motion (stars). Figure A2 shows the magnitude of the eigenvalues under variations in hip width.

4.2.2 Symmetric feet

The periodic motion for the walker model with zero hip width was used as the starting point for the parameter study performed to achieve symmetric feet. Here the asymmetry parameter μ (see Figure 4.1) was reduced in small decrements from 4.9 cm to 0 cm while leaving ρ equal to 7.5 cm. Figure 4.2B shows the variations in location and stability of the

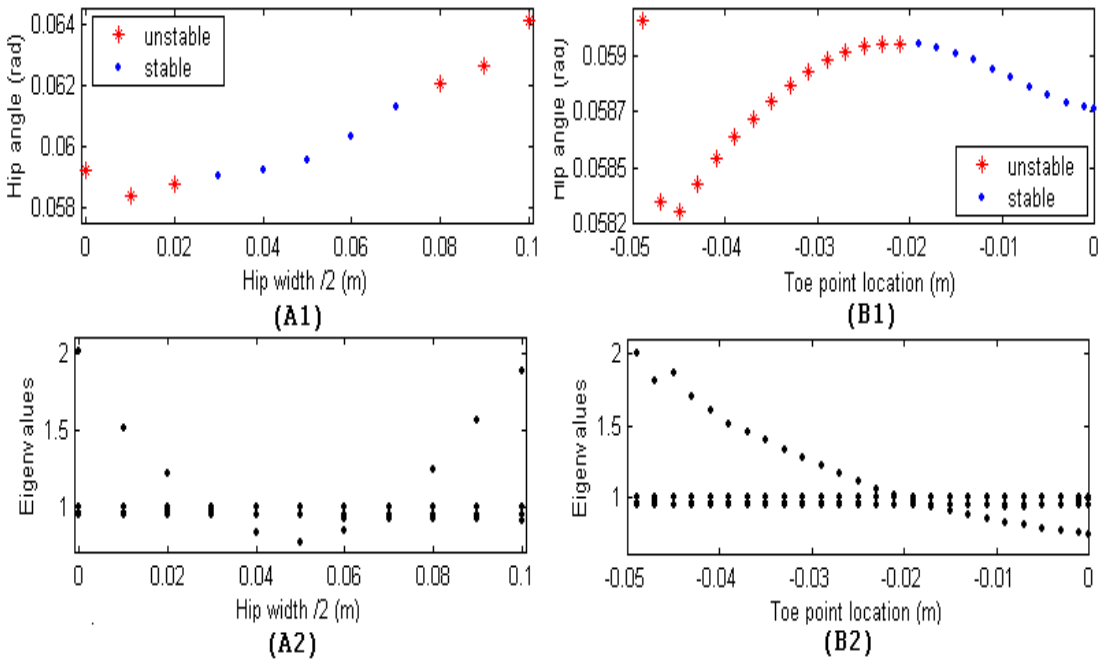

Figure 4.2: Parameter studies: (A) going to zero hip, (B) making the feet symmetric, with (1) the parameter studied versus right hip angle and (2) the eigenvalues corresponding to the parameter.

periodic orbit under variations in μ. Here it is found that as the feet become closer to being symmetric, the walker motion becomes more stable.

The asymmetry in the toe point geometry with respect to the limb results in side-to-side motion of the mechanism. In contrast, for an extended 2D walker, walking motions are possible that show no displacement in the horizontal direction. To illustrate this Figure 4.3 shows the displacement of the center of mass of the torso in the horizontal direction for an asymmetric 3D walker (upper plot) and for an extended 2D walker (lower plot). Note that the variations seen in the second plot are well within the numerical tolerance of the simulation. Figure 4.4 shows the displacement of the center of mass of the torso in both the vertical and horizontal directions for the asymmetric 3D walker. As the walker moves up and down it also sways to the left and right with a peak to peak amplitude of 10 cm.

The walking motion found at the conclusion of this parameter study is that of an extended 2D walker. Tables 4.1, 4.2 and 4.3 provide numerical values for system parameters and initial conditions at the $q_7 = 0$ Poincaré section. Although this walker could be used as the starting point of an analysis of a walking model with more degrees-of-freedom at the hip, it is certainly valuable to find more than one model with extended 2D walker motion. For example, additional walking motions could be found by applying the continuation method to variations in inclination or toe width.

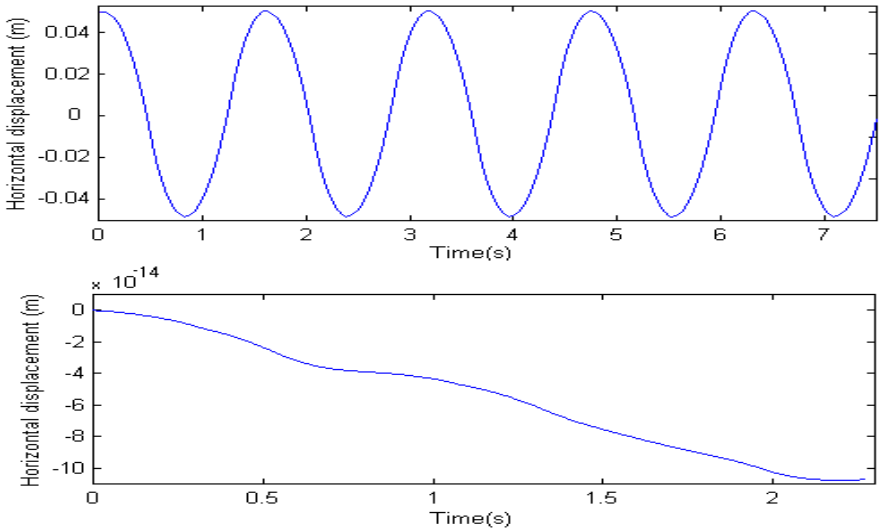

Figure 4.3: Horizontal displacement of the center of mass of the torso for an asymmetric 3D walker (upper plot) and for an extended 2D walker (lower plot).

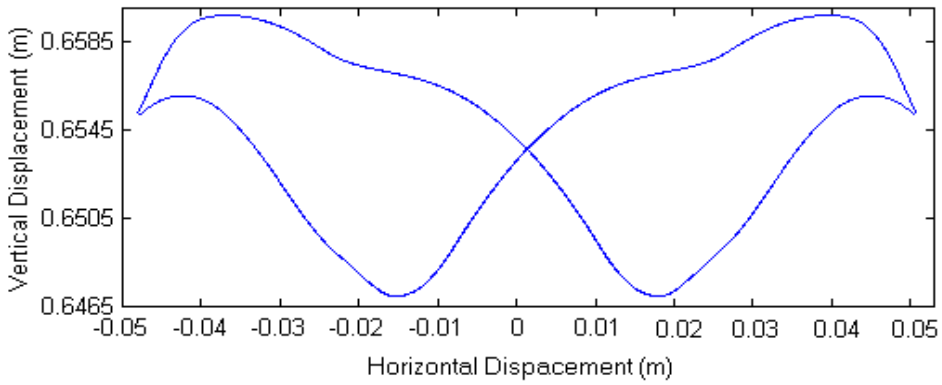

Figure 4.4: 3D hip movement for an asymmetric walker relative to a vertical plane parallel to the gait.

Table 4.1: Initial Conditions used in simulation of walker

State variables	Values	State variables	Values
q_1	0.0	u_1	0.0
q_2	0.6534094640	u_2	-0.0055317806
q_3	0.0	u_3	0.1501207798
q_4	0.0	u_4	0.0004616846
q_5	0.0	u_5	0.0
q_6	0.0	u_6	0.0
q_7	0.0	u_7	-3.9425921403
q_8	0.8090243050	u_8	5.1587075305
q_9	0.0587667066	u_9	0.2344394643
q_{10}	-0.0084559414	u_{10}	-0.0200420565
s_1	0.075	s_2	-0.2778518086
s_3	-0.075	s_4	-0.2778518086
s_5	0.075	s_6	-0.4134948813
s_7	-0.075	s_8	-0.4134948813
s_9	-0.075	s_{10}	0.0665906223
s_{11}	0.075	s_{12}	0.0665906223
s_{13}	-0.075	s_{14}	-0.0690524504
s_{15}	0.075	s_{16}	-0.0690524504
θ_1	0.0	θ_4	0.0
θ_2	0.0	θ_5	0.0
θ_3	0.0	θ_6	0.0

Table 4.2: Center of Mass, Length, and Toe point Parameter values

	Center of Mass		
Point Label	b_1	b_2	b_3
T_{cm}	0.0000	0.0000	0.0000
$U_{cm,1}, U_{cm,2}$	0.0000	-.1010	0.0000
$L_{cm,1}, L_{cm,2}$	0.0000	-.1663	0.0432
	Length		
U_1, U_2	0.0000	-.3400	0.0000
L_1, L_2	0.0000	-.3000	0.0000
	Toe Points		
$P_{1,1}$	0.0750	-.01322	1.050
$P_{1,2}$	-0.0750	-.01322	1.050
$P_{1,3}$	0.0750	-.00302	-0.030
$P_{1,4}$	-0.0750	-.00302	-0.030
$P_{2,1}$	-0.0750	-.01322	1.050
$P_{2,2}$	0.0750	-.01322	1.050
$P_{2,3}$	-0.0750	-.00302	-0.030
$P_{2,4}$	0.0750	-.00302	-0.030
	Ground Parameter values		
Gravity	9.81		
Incline	-.0665rad		

Table 4.3: Mass and Moments of Inertia Parameter values

		Moments of inertia					
Body part	mass	b_1	b_2	b_3	12	13	23
Torso	0.800	2.0000	2.2000	2.2000	0.0	0.0	0.0
Upper leg	2.925	0.0200	0.0010	0.0200	0.0	0.0	0.0
Lower leg	1.013	0.0393	0.0010	0.0100	0.0	0.0	0.0

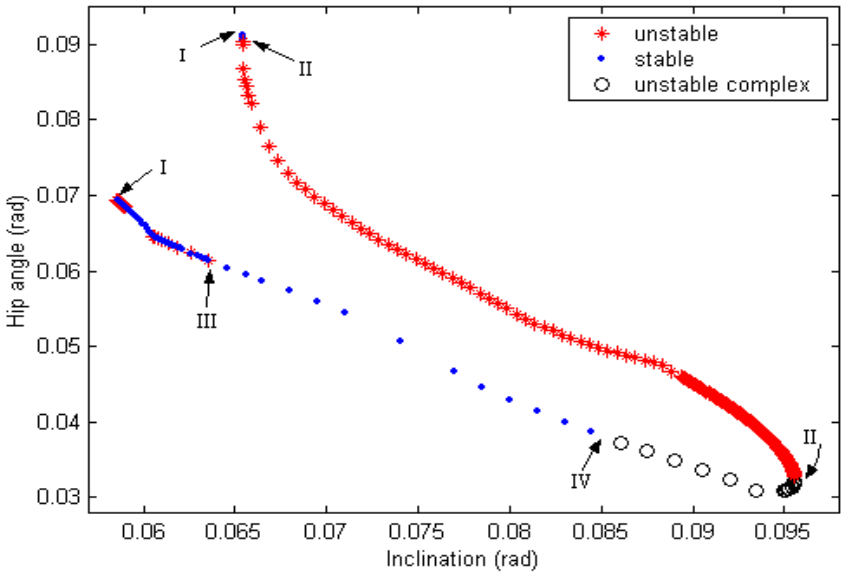

Figure 4.5: A bifurcation diagram for an extended 2D walker with 15 cm foot width under variations in inclination.

4.2.3 Bifurcation Diagrams

With the extended 2D walker a parameter study of the inclination of the ground was performed to construct a bifurcation diagram. This allowed us to see what initial conditions and parameters are needed to develop a walker that can produce periodic gait. We started with the slope parameter of the extended 2D walker motion corresponding to Table 4.1 and varied the slope in the positive and negative direction while applying the Newton-Raphson method. The eigenvalues were calculated for each new set of parameters and initial conditions at the Poincaré section $q_7 = 0$. With these values the bifurcation diagram was constructed, see Figure 4.7. Here the 'crosses' represent unstable periodic orbits while the 'dots' represent stable periodic orbits. At each Roman numeral location a bifurcation occurs. The *I* represents grazing, *II* corresponds to a saddle-node bifurcation, *III* corresponds to a pitchfork bifurcation, and *IV* corresponds to a Hopf bifurcation. See section 2.8 for a more detailed discussion of each type of bifurcation.

Starting at the left of the bottom branch, *I* represents the existence of a grazing bifurcation. The branch is not continued further to the left because the walker's foot scuffs and is unable to maintain gait. It makes sense that for less steep slope the walker will have less momentum forward and would be unable to swing its leg forward without hitting the ground and collapsing, unable to complete its step. At this bifurcation point, there is only one eigenvalue larger than one and it is 1.0283. Moving along the branch to the right this eigenvalue is reduced until it crosses into the unit circle. This happens at *III*, where a super-critical pitchfork bifurcation occurs. Figure 4.6 shows a closer look at the pitchfork bifurcation by plotting the gait's heading q_5 versus the variation in inclination. Here, we see the birth of two new stable branches of walking motions that move down the incline at an angle relative to the nominal vertical plane, i.e. the plane that contains the gravity vector and is perpendicular to the plane. Figure 4.7 shows the horizontal displacement of

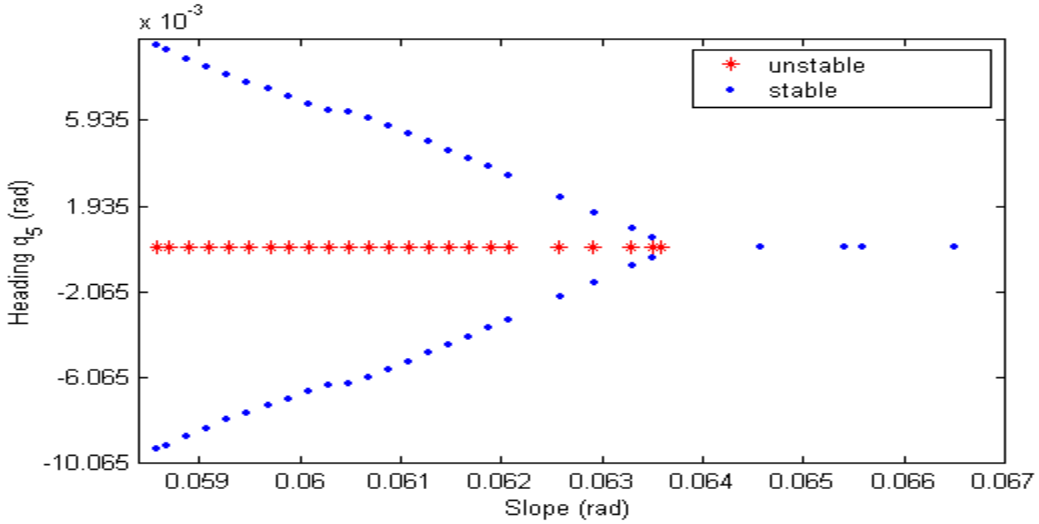

Figure 4.6: A closer look at the super-critical pitchfork bifurcation.

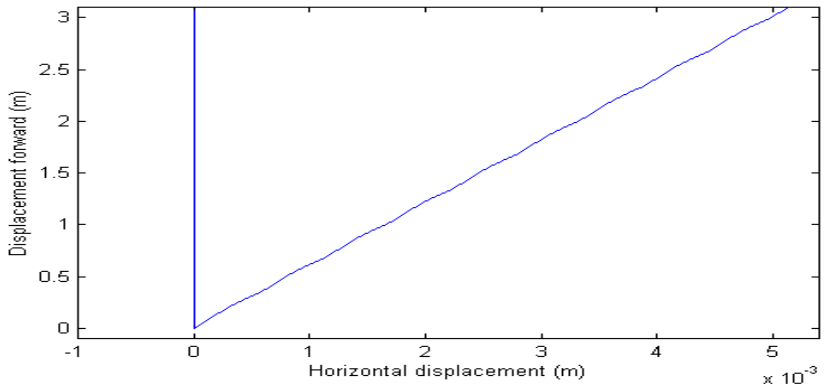

Figure 4.7: A change in heading down the incline due to the pitchfork bifurcation as seen from above for two different extended 2D walking motions.

the torso's center of mass for a walker on the original branch of periodic orbits and a walker on the new branch formed at the pitchfork bifurcation, as they walk down the incline.

Continuing down the stable branch of the bifurcation diagram, a complex conjugate pair of eigenvalues cross out of the unit circle at *IV*. This Hopf bifurcation is associated with the formation of a quasi-periodic motion and the loss of stability of the periodic motion. Figure 4.8A shows the intersection of the quasi-periodic motion with the Poincaré section $q_7 = 0$, projected on the space spanned by the right hip angle and the right knee angle. Here, the 'star' represents the location of the now unstable periodic orbit. Figure 4.8B shows the full system trajectory projected on the space spanned by the left hip angle and the vertical displacement of the center of mass of the torso. It was found that this quasi-periodic motion was entirely 2D, involving no side-to-side motion. In contrast, in Piiroinen's model, quasi-periodic motion was as a result of 3D-gait [31, pages 22-24].

Following along the original branch past the Hopf bifurcation the complex pair of eigenvalues become a real pair and one eigenvalue subsequently enters the unit circle at the saddle node bifurcation point. Thus, on the upper half of the branch there is only one eigenvalue

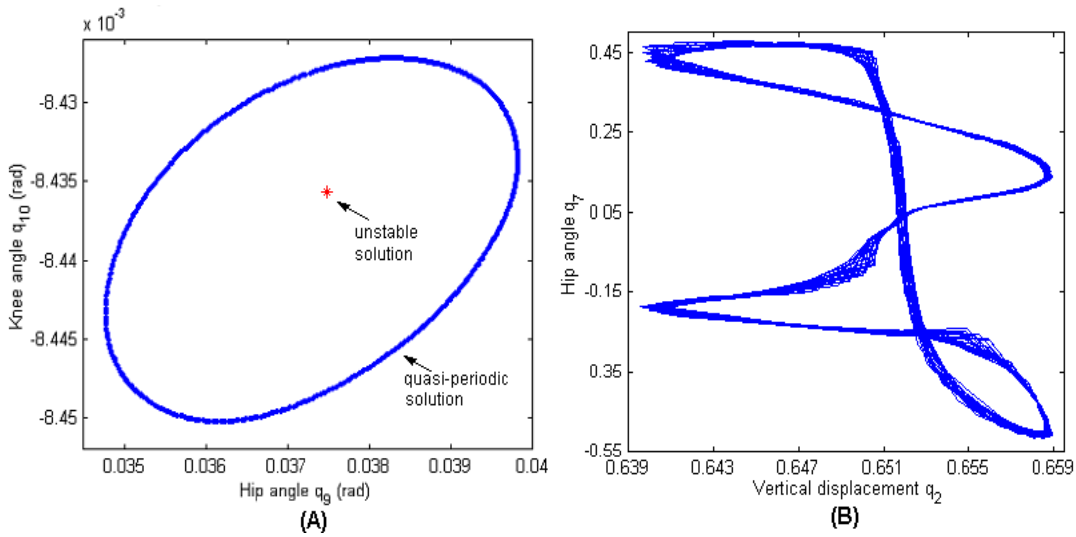

Figure 4.8: Motion towards a quasi-periodic solution formed at the Hopf bifurcation.

larger than one. Moving along the branch to the left, the slope is decreasing and the instability increases. At approximately a slope of 0.0652 radians, the magnitude of the eigenvalue starts to decrease at a fast rate. Then another saddle node bifurcation occurs and the one unstable eigenvalue enters the unit circle. There is coexisting stable gait on the upper and lower branches of the bifurcation diagram. Finally, a grazing bifurcation occurs and the bifurcation diagram is complete.

4.2.4 Different Toe widths

The goal of this section is to study the effects that changes the lateral separation of the toe contact points have on the stability of the 2D motion of an extended 2D walker to find additional 2D walking motions that could possibly be used as starting points for the model with more degrees-of-freedom in the hip. When changing the lateral separation of the toe contact points, the planar motion of the extended 2D walker persists. However inherently out-of-plane instabilities may occur. A parameter study was done to vary the lateral separation of the toe contact points from 2 cm to 17 cm. Figure 4.9 shows bifurcation diagrams for the 2D motion exhibited by four different extended 2D walkers with different toe separation. Here, the 'dots' indicate stable solutions, where as the 'stars','triangles', and 'squares' indicate unstable solutions with one, two, or three real eigenvalues larger than one, respectively. Finally, the 'circles', 'diamonds', and 'crosses' correspond to solutions with a complex conjugate pair of eigenvalues outside the unit circle plus zero, one, or two other eigenvalues outside of the unit circle, respectively. The Roman numeral V indicates bifurcations that result in 3D instabilities. These bifurcations sweep across the solution branches as the toe separation is increased while the bifurcations resulting in 2D motion remain unchanged (e.g., the Hopf and saddle node bifurcation). Note that for sufficiently large toe separation the bifurcation is that of a constrained 2D walker.

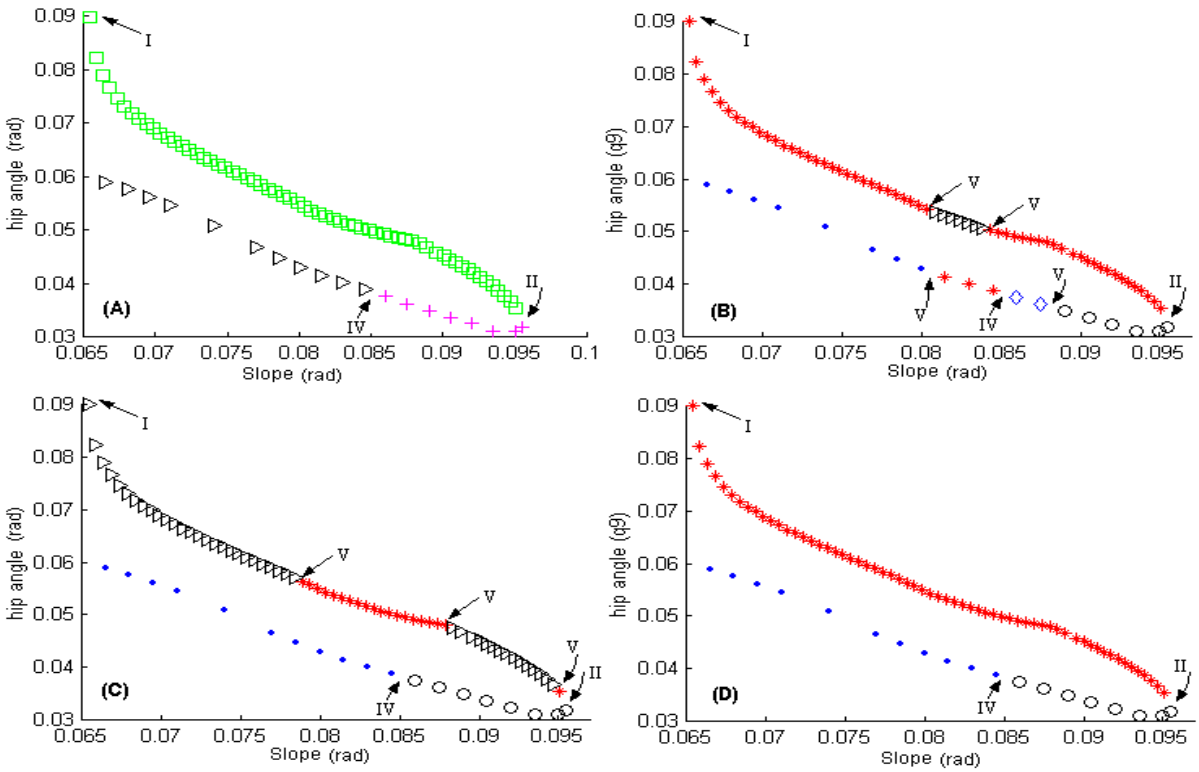

Figure 4.9: Bifurcation diagrams for extended 2D walkers with (A)2, (B)9.5, (C)13, and (D)17 cm foot width.

4.3 Control of passive walkers

As described previously, a passive walker is one that is able to exhibit anthropomorphic gait without actuation and having only gravity as its energy supply. The design of a feedback-based control actuation that affects the stability of the recurrent gait of the walker while involving no exchange of energy with the system would be highly desirable. The passive walkers we are studying are modeled as having abrupt discontinuous changes in the vector field at ground contact or release and knee hyperextension or knee release due to the jumps in damping and restoring forces, see Section 3.2.2. Our approach for control is to make a discrete change in the orientation of the foot while it is not in contact with the ground and keeping the ankle rigid for all other times. Doing this has an indirect effect on the timing and state of the walker at the subsequent foot impact with the ground, thus affecting the local stability properties of a reference periodic gait [33]. Since the foot is mass-less, there is effectively no cost associated with this control.

The inclusion of control introduces the possibility of changing the Jacobian of the Poincaré mapping of a reference trajectory and, consequently, the corresponding eigenvalues. It has been demonstrated that by using reference feedback control, successful stabilization can be accomplished for previously unstable motions of models constrained to two dimensions as well as an unconstrained model able to move in three-dimensions [35]. Other work has been done by Kuo [21] and Formal'sky [13] to implement active feedback control on passive walkers by injecting energy at opportune moments during the gait cycle and to the appropriate degrees of freedom. In Kuo's recent paper, he proposes a low-cost control strategy to control the lateral instability of a knee-less passive walker. Achieved by controlling the affective lateral step width prior to establishing ground contact.

4.3.1 Gain Parameters

When implementing the control to a passive walker's motion, the matrices are formulated slightly differently from those discussed in Section 2.9. As previously discussed we will be using two types of control algorithms: reference and delay feedback. Reference feedback has a constant comparison state, while the delay feedback updates the comparison state every time the actual trajectory intersects the discontinuity surface. Refer to Section 2.9 for a complete discussion of the mathematical concepts and for notation.

Control is activated twice within one periodic orbit, once for each leg. Two control discontinuity surfaces are described coincident with the Poincaré surfaces introduced previously. Specifically, let $\mathcal{P}_{0,control}$ and $\mathcal{P}_{1,control}$ be the zero-level surfaces of the event functions $H_{0,control} = q_7$ and $H_{1,control} = q_9$, respectively. Here, when q_7 or q_9 are equal to zero, the left or right upper leg segment are close to perpendicular to the inclined plane, with the corresponding foot in the air. This ensures that the foot will be off of the ground when the discrete change in the foot's orientation is made, guaranteeing that the control does no work. To accommodate the code used to numerically simulate this system, the augmented state vector is given by

$$\mathbf{y} = \begin{pmatrix} \mathbf{x_1} \\ \theta_L \\ \theta_R \\ \mathbf{x_2} \\ \mathbf{x_L}^* \\ \mathbf{x_R}^* \end{pmatrix}, \tag{4.1}$$

where $\mathbf{x_1}$ has 26 components, θ_L and θ_R have 3 components each

$$\theta_L = (\theta_1\ \theta_2\ \theta_3)^T \tag{4.2}$$

and

$$\theta_R = (\theta_4\ \theta_5\ \theta_6)^T, \tag{4.3}$$

and $\mathbf{x_2}$ has 10 components, describing the 42 state variables, while $\mathbf{x_L}^*$ and $\mathbf{x_R}^*$ have 36 components each, describing the comparison state corresponding to the two different control discontinuity surfaces.

Suppose that the reference trajectory intersects $\mathcal{P}_{0,control}$ and $\mathcal{P}_{1,control}$ transversely at the points $\mathbf{y}_0^{ref}$ and $\mathbf{y}_1^{ref}$, respectively. Then all trajectories based at nearby initial conditions will intersect $\mathcal{P}_{0,control}$ and $\mathcal{P}_{1,control}$ transversely at points near $\mathbf{y}_0^{ref}$ and $\mathbf{y}_1^{ref}$, respectively.

The jump functions as described in section 2.9 will be used to control the stability of a periodic reference walking motion by introducing a discrete correction in θ when a trajectory intersects the control surface corresponding to the swing leg. There are two jump functions for each control scheme associated with the left and right leg. Due to the construction to the augmented state vector, the jump functions are represented as

$$\mathbf{g}^L_{control}(\mathbf{y}) = \begin{pmatrix} \mathbf{x_1} \\ \theta_L^{ref} + C_L\left(\begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} - \mathbf{x}_L^*\right) \\ \theta_R \\ \mathbf{x_2} \\ \mathbf{x}_L^* \\ \mathbf{x}_R^* \end{pmatrix} \tag{4.4}$$

and

$$\mathbf{g}^R_{control}(\mathbf{y}) = \begin{pmatrix} \mathbf{x_1} \\ \theta_L \\ \theta_R^{ref} + C_R\left(\begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} - \mathbf{x}_R^*\right) \\ \mathbf{x_2} \\ \mathbf{x}_L^* \\ \mathbf{x}_R^* \end{pmatrix} \tag{4.5}$$

for the left and right foot, respectively in the case of reference feedback control and

$$\mathbf{g}^{L}_{control}(\mathbf{y}) = \begin{pmatrix} \mathbf{x_1} \\ \theta_L^{ref} + C_L \left(\begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} - \mathbf{x}_L \right) \\ \theta_R \\ \mathbf{x_2} \\ \begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} \\ \mathbf{x}_R^* \end{pmatrix} \tag{4.6}$$

and

$$\mathbf{g}^{R}_{control}(\mathbf{y}) = \begin{pmatrix} \mathbf{x_1} \\ \theta_L \\ \theta_R^{ref} + C_R \left(\begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} - \mathbf{x}_R \right) \\ \mathbf{x_2} \\ \mathbf{x}_L^* \\ \begin{pmatrix} \mathbf{x_1} \\ \mathbf{x_2} \end{pmatrix} \end{pmatrix} \tag{4.7}$$

for the left and right foot, respectively in the case of delay feedback control. In the case of delay control, the Jacobian of the jump function for each control discontinuity surface can be shown in block matrix format as

$$D_{\mathbf{y}}\mathbf{g}^{L}_{control}(\mathbf{y}_0^{ref}) = \begin{pmatrix} Id & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ C_1^L & \mathbf{0} & \mathbf{0} & C_2^L & -C^L & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & Id & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & Id & \mathbf{0} & \mathbf{0} \\ \begin{pmatrix} Id \\ \mathbf{0} \end{pmatrix} & \mathbf{0} & \mathbf{0} & \begin{pmatrix} \mathbf{0} \\ Id \end{pmatrix} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & Id \end{pmatrix} \tag{4.8}$$

and

$$D_{\mathbf{y}}\mathbf{g}^{R}_{control}(\mathbf{y}_0^{ref}) = \begin{pmatrix} Id & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Id & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ C_1^R & \mathbf{0} & \mathbf{0} & C_2^R & \mathbf{0} & -C^R \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & Id & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & Id & \mathbf{0} \\ \begin{pmatrix} Id \\ \mathbf{0} \end{pmatrix} & \mathbf{0} & \mathbf{0} & \begin{pmatrix} \mathbf{0} \\ Id \end{pmatrix} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{4.9}$$

where the matrices are of the size 114x114 because of the components described in the augmented state vector and $C = (C_1\ C_2)$. These matrices can be used for reference feedback control if the -C block is set to zero and the last two rows are zero except for the diagonal having the identity, due to the fact that the comparison state does not change when control is added to the system.

The C block matrices govern the change of the foot orientation, ideally to increase the stability of the walker's motion. The three rows of the C matrices associated with the angles that change due to the discrete jump at the corresponding discontinuity surface contain 16 nonzero elements. These nonzero elements are referred to as gain parameters. There are a total of 36 gain parameters that could have been used. However, only a subset of the state variables were chosen to be used in the feedback control. It is important to choose gain parameters that will optimize the stability characteristics of the reference gait. An algorithm was written in Matlab, utilizing the fminsearch function, to find an optimal choice of gain parameters that will give the largest reduction in the largest eigenvalue of the Jacobian of the Poincaré mapping.

4.3.2 Opening hip

When using gain parameters to increase the stability for a model with zero hip width it is not possible to reduce all of the eigenvalues. This can be explained by looking at the components in the Jacobian of the Poincaré mapping, represented in a block matrix form by

$$D_y\mathbf{P}(\mathbf{y}^{ref}) = \begin{pmatrix} A & \mathbf{0} & B & \mathbf{0} & C & D & E \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & Id & Id & \mathbf{0} \\ F & \mathbf{0} & G & \mathbf{0} & H & I & J \end{pmatrix}, \tag{4.10}$$

where the blocks shown by letters contain a mixture of zero and nonzero elements. When the Jacobian of the Poincaré mapping is multiplied by the Jacobian of the jump function (refer to Equation 4.8), the columns in the blocks C and H will be multiplied by the gain parameters. It is noticed that the C and H blocks contains elements equal to zero in specific locations associated with 11 out of the 16 gain parameters. Thus, these 11 gain parameters have no effect on changing the eigenvalues of the Jacobian of the Poincaré mapping.

To avoid this problem, the hip width was opened until the feet did not overlap. Doing this produces a Jacobian of the Poincaré mapping with no zero elements in the specific locations that would be affected by the gain parameters. A walker motion was chosen from the set found previously by using the continuation method. The walker had symmetric feet, zero hip width, and walked down a 0.068 radian incline. By doing a parameter study, the hip was opened to 13 cm, still retaining an over lap of the feet of 2 cm. The original parameter study was not able to be continued because the walker experiences foot scuffing.

A different parameter study on the variation of the slope of the incline was preformed to see if at a steeper incline whether the hip could be opened more. The inclination was increased in small increments until a new slope of 0.0723 radians was reached. Using this slope, the hip was opened to 16.1 cm using the continuation method. Figures 4.10 and 4.11 show the persistence and stability of the walking motion as a result of the parameter variations.

4.3.3 Symmetry

For a symmetric gait it is possible to formulate a relationship between the elements of the C^L and C^R matrices corresponding to reflection symmetric control of the left and right feet, respectively. See Piiroinen ([31, 33]) for similar testing methods and results for a different

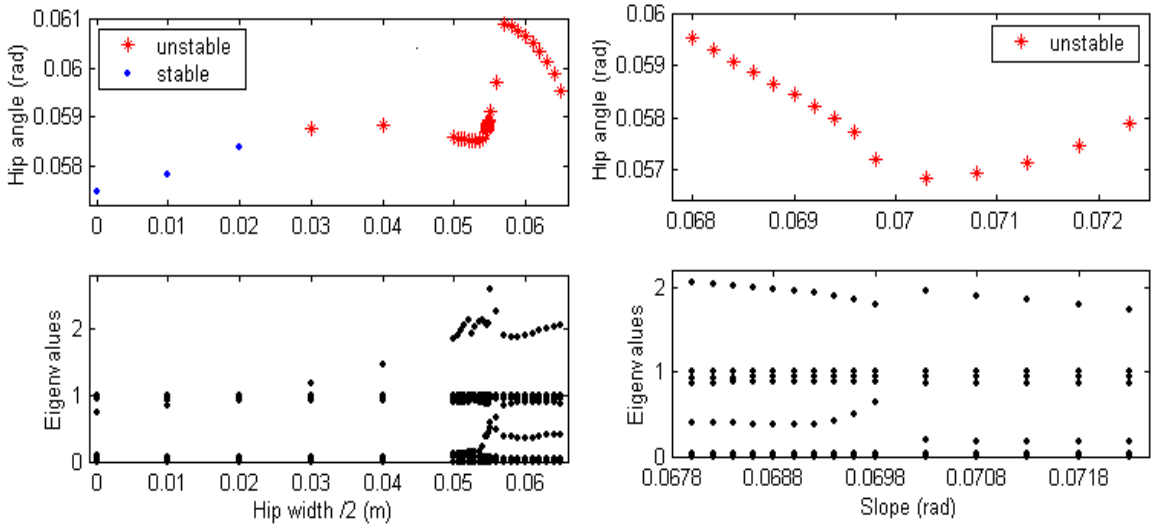

Figure 4.10: Parameter variations for (A) opening hip from 0 to 13 cm and (B) increasing inclination from .068 to .0723rad, with (1) the parameter varied versus the right hip angle and (2) the eigenvalues of the Jacobian at the Poincaré section $q_7 = 0$.

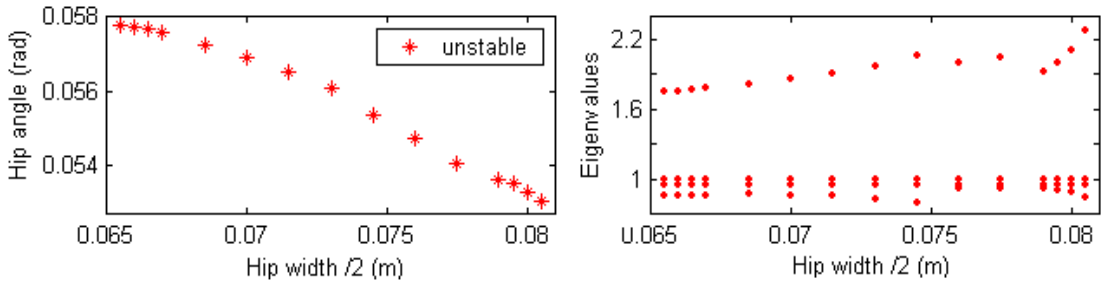

Figure 4.11: Persistence and stability for parameter variations of opening the hip from 13 to 16.1 cm.

walker motion. Specifically, a periodic gait is symmetric, if for each step the motion of the left limb during the first half of the step is a mirror reflection of that of the right limb during the second half of the step and vica versa. For a symmetric walker, a symmetric gait coexists with its mirror image, as obtained by changing the state variables according to

$$\begin{array}{lcll}
q_i & \rightarrow & q_i & i = 2,3,4 \\
q_i & \rightarrow & -q_i & i = 1,5,6 \\
q_i & \rightarrow & q_{i+2} & i = 7,8 \\
q_i & \rightarrow & q_{i-2} & i = 9,10 \\
s_{i,j,1} & \rightarrow & -s_{3-i,j,1} & i = 1,2 \quad j = 1,2,3,4 \\
s_{i,j,3} & \rightarrow & s_{3-i,j,3} & i = 1,2 \quad j = 1,2,3,4 \\
u_i & \rightarrow & u_i & i = 2,3,4 \\
u_i & \rightarrow & -u_i & i = 1,5,6 \\
u_i & \rightarrow & u_{i+2} & i = 7,8 \\
u_i & \rightarrow & u_{i-2} & i = 9,10 \\
\theta_i & \rightarrow & -\theta_{i+3} & i = 1,2 \\
\theta_i & \rightarrow & \theta_{i+3} & i = 3 \\
\theta_i & \rightarrow & -\theta_{i-3} & i = 4,5 \\
\theta_i & \rightarrow & \theta_{i-3} & i = 6
\end{array}$$

As a consequence, for a symmetric gait we obtain the symmetries

$$\begin{array}{lcll}
C^L_{i,j} & = & -C^I_{i+3,j} & i = 1,2 \\
C^L_{i,j} & = & C^I_{i+3,j} & i = 3 \\
C^I_{i,j} & = & C^R_{i,j} & j = 1,2,3,4,28,29,30 \\
C^I_{i,j} & = & -C^R_{i,j} & j = 5,6,27,31,32 \\
C^I_{i,j} & = & C^R_{i,j+2} & j = 7,8,33,34 \\
C^I_{i,j} & = & C^R_{i,j-2} & j = 9,10,35,36.
\end{array}$$

This does not include the toe point symmetries, since the initial toe point state variables are not considered within the 16 gain parameters.

We proceed to check whether the gait found in the previous section satisfies the symmetry requirements. Figure 4.12 shows the time variation of the two hip angles over one and a half periods. A horizontal line has been drawn at the maximum hip angle, demonstrating that both hip angles achieve the same maximum value. In Figure 4.13 the time history of the right hip angle q_9 is shifted half a period along the horizontal. The perfect overlap supports the contention that the walking motion is symmetric about the sagittal plane, allowing for the implementation of the above symmetries in the formulation of the C^L and C^R matrices.

4.3.4 Numerical results of control

In the absence of control, the reference gait has a largest-in-magnitude eigenvalue of 2.27. A limited search with Matlab's fminsearch function was done for both reference and delay feedback control. A local optimal choice of gain parameters was found that reduced the largest-in-magnitude eigenvalue to less than one for both control schemes; 0.41 for reference feedback and 0.93 for delay feedback. Thus, with control the walker is able to sustain stable

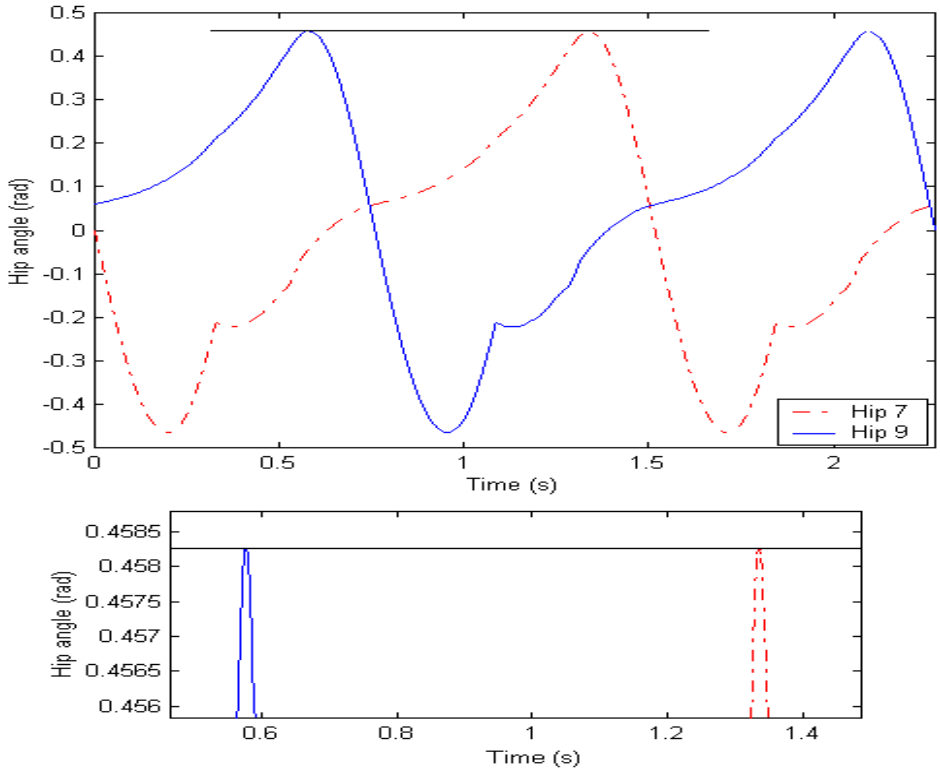

Figure 4.12: The orientation of the left and right hip as a function of time for a period-1 symmetric gait over one and a half strides.

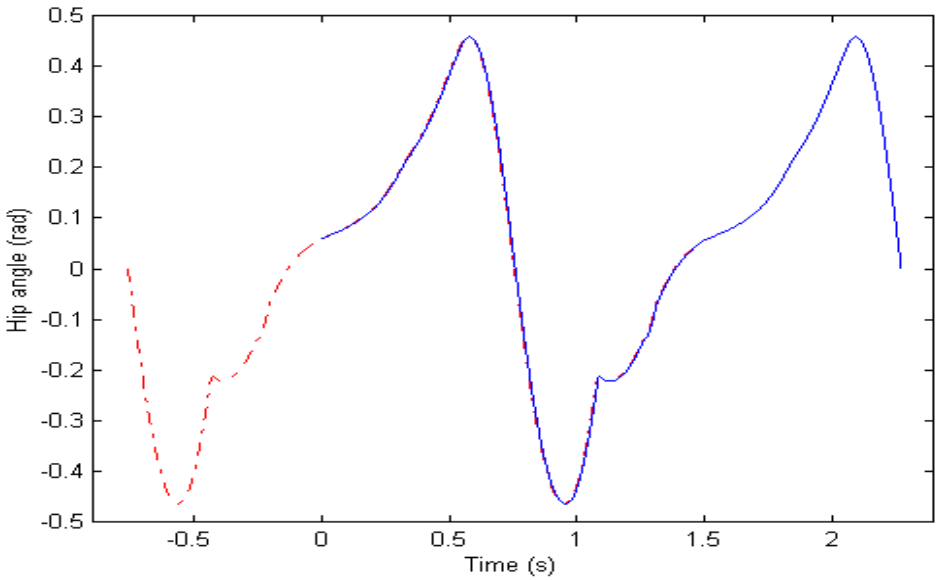

Figure 4.13: The orientation of the left and right hip with the right hip shifted half the time period.

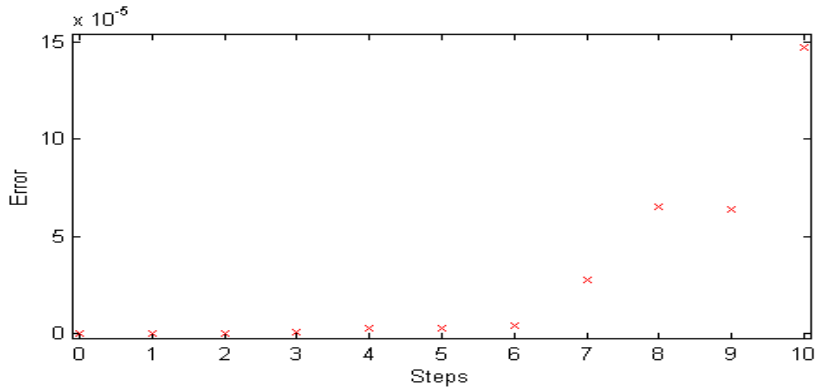

Figure 4.14: The walking motion of a nearby trajectory deviates from the original trajectory showing that the motion is unstable.

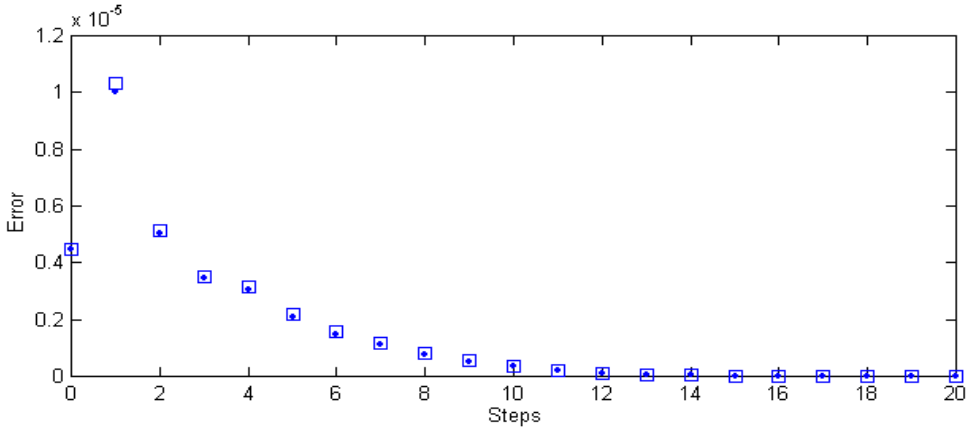

Figure 4.15: Stable walking motion persists for a nearby trajectory by converging on the original trajectory by using reference feedback control.

motion. To verify these predictions, a comparison was made between a direct numerical simulation and the prediction of the linear analysis. In particular, the dynamics of a nearby trajectory were studied with and without control. Figure 4.14 confirms the unstable characteristics of the walking motion persist in the absence of control. In contrast, in Figure 4.15 reference feedback control is used to ensure that the deviation of the actual trajectory from the reference trajectory decays in forward time. Here, the 'dots' indicate results from numerical simulation, while the 'squares' indicate the results of the linear prediction. Similarly, Figure 4.16 presents the same results for the delay feedback scheme.

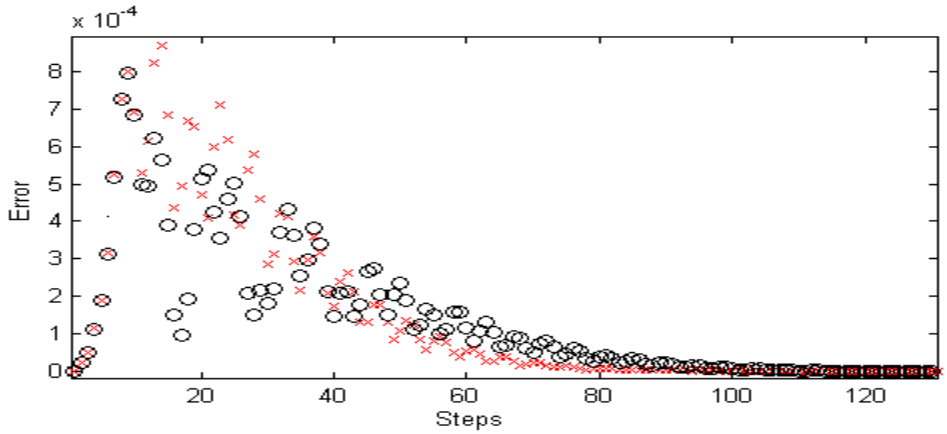

Figure 4.16: Stable walking motion persists for a nearby trajectory by converging on the original trajectory by using delay feedback control. The 'crosses' represent linear prediction and the 'circles' represent direct numerical simulation.

Chapter 5

Analysis of new model

The walker model discussed in this chapter was selected from the set of extended 2D walkers found in the previous chapter. Two additional degrees-of-freedom were added to each hip joint and the continuation method was used to find inherently 3D gait. Feedback control was also be added to a walker with unstable gait motion to try and increase its stability.

All masses and moments of inertia described for each rigid body are the same as the old walker. The location of the centers of mass are also the same, as is the length of the upper and lower limbs. See Tables 4.2 and 4.3 for a review of this information. The initial conditions associated with this new walker can be seen in Table 5.1. It is important to remember that the hip and ankle angles now relate the orientation of the upper limbs and the feet relative to the Newtonian reference frame, whereas previously the upper limb orientation was expressed relative to the $\tilde{t}$ reference basis and the orientation of the feet was expressed relative to the lower limbs. As was done with the previous walker, two zero-level Poincaré surfaces are introduced to study the periodic solutions of the passive walkers, namely at $q_7 = 0$ and $q_{11} = 0$.

5.1 Extended 2D to 3D walker model

With zero hip width and finite toe separation, the model is an extended 2D walker. However, to make the walker's motion more anthropomorphic, we would like to separate the hip points. An initial attempt of using the continuation method failed due to a near singular Jacobian in the Newton-Raphson scheme. It was found the inclusion of large spring and damper constants modeling the torques at the hip joints (see Section 3.3.2) were necessary to apply the method. Additional parameter studies in this area could be done to find the ideal constants that produce periodic gait as well as allow for motion along all three degrees-of-freedom in the hip.

After a good initial guess was found we were able to continue the parameter study and open the hip further. The hip width was increased in small increments to 4.4 cm. However, for a more anthropomorphic walking motion the ideal distance of separation would be 15 cm, insuring that the feet do not overlap. The parameter study could not be continued due to foot scuffing and a different parameter study was done to increase the slope of the inclined plane from .0770 to .0778 radians. In Figure 5.1, the results for these two parameter

Table 5.1: Initial conditions used in simulation of walker

State variables	Values	State variables	Values
q_1	0.0	u_1	0.0
q_2	0.6526262181	u_2	-0.0013783211
q_3	0.0	u_3	0.1963695772
q_4	0.0	u_4	0.0005889622
q_5	0.0	u_5	0.0
q_6	0.0	u_6	0.0
q_7	0.0	u_7	-4.1576567527
q_8	0.0	u_8	0.0
q_9	0.0	u_9	0.0
q_{10}	0.8464187592	u_{10}	5.3769535219
q_{11}	0.0466077008	u_{11}	0.3024320912
q_{12}	0.0	u_{12}	0.0
q_{13}	0.0	u_{13}	0.0
q_{14}	-0.0084259611	u_{14}	-0.022084118
s_1	0.075	s_2	-0.306480381
s_3	-0.075	s_4	-0.306420381
s_5	0.075	s_6	-0.442370156
s_7	-0.075	s_8	-0.442370356
s_9	-0.075	s_{10}	0.074597670
s_{11}	0.075	s_{12}	0.0745976705
s_{13}	-0.075	s_{14}	-0.0613446070
s_{15}	0.075	s_{16}	-0.0613446070
θ_1	0.8464187591	θ_4	0.0381817397
θ_2	0.0	θ_5	0.0
θ_3	0.0	θ_6	0.0

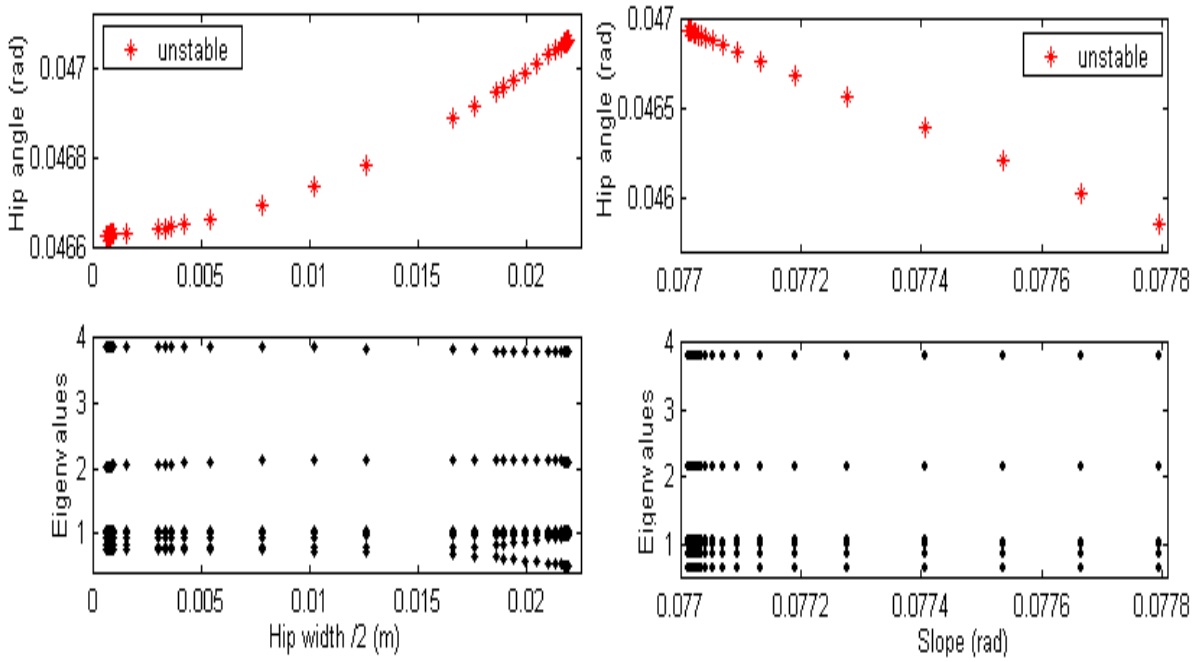

Figure 5.1: Persistence and stability of motion for parameter variations for opening the hip from 0 to 4.4 cm and increasing the inclination from .077 to .0778rad.

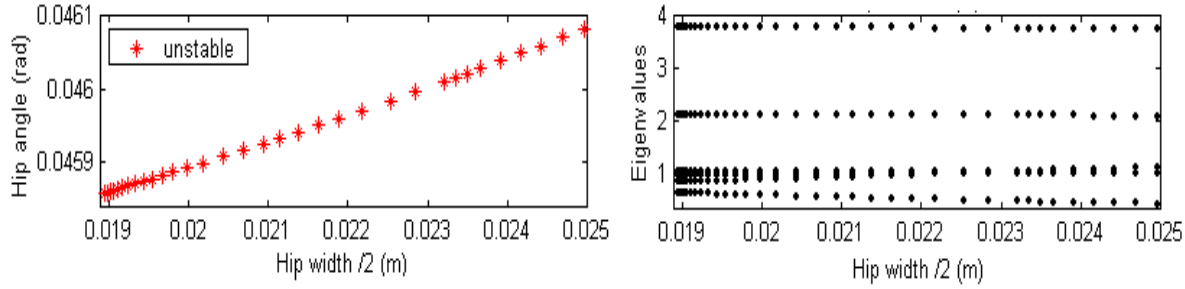

Figure 5.2: Persistence and stability of motion for parameter variations for opening the hip from 3.75 to 5 cm.

studies can be seen. These plots show the values of the right hip angle q_{11} at the Poincaré section $q_7 = 0$ versus the parameter varied. Using the new initial conditions found at the end of the increased-inclination parameter study the hip separation was increased to 5 cm, see Figure 5.2. These parameter studies take an ample amount of time to run (approximately 30 minutes to simulate one step on a Dell Optiplex GX110 with a 500MHz Pentium III processor and an available memory of 260MB RAM) because the equations of motion and variational equations are large and numerous (size of C-files used are 2,040kB). Focus was thus shifted to see if these walkers could be stabilized, as was done for the previous walker using reference and delay feedback control. The walker's motion is unstable for every new parameter set with the largest-in-magnitude eigenvalue slightly less then 4.

5.2 Symmetry and hip movement

As discussed previously, for a symmetric gait it is possible to formulate a relationship between the elements of the C^L and C^R matrices corresponding to reflection symmetric control of the left and right feet, respectively. For a symmetric 3D walker with three degrees of freedom in each hip, a symmetric gait coexists with its mirror image, as obtained by changing the state variables according to

$$
\begin{array}{rcll}
q_i & \rightarrow & q_i & i = 2,3,4 \\
q_i & \rightarrow & -q_i & i = 1,5,6 \\
q_i & \rightarrow & q_{i+4} & i = 7,10 \\
q_i & \rightarrow & -q_{i+4} & i = 8,9 \\
q_i & \rightarrow & q_{i-4} & i = 11,14 \\
q_i & \rightarrow & -q_{i-4} & i = 12,13 \\
s_{i,j,1} & \rightarrow & -s_{3-i,j,1} & i = 1,2 \quad j = 1,2,3,4 \\
s_{i,j,3} & \rightarrow & s_{3-i,j,3} & i = 1,2 \quad j = 1,2,3,4 \\
u_i & \rightarrow & -u_i & i = 1,5,6 \\
u_i & \rightarrow & u_i & i = 2,3,4 \\
u_i & \rightarrow & u_{i+4} & i = 7,10 \\
u_i & \rightarrow & -u_{i+4} & i = 8,9 \\
u_i & \rightarrow & u_{i-4} & i = 11,14 \\
u_i & \rightarrow & -u_{i-4} & i = 12,13 \\
\theta_i & \rightarrow & -\theta_{i+3} & i = 1,2 \\
\theta_i & \rightarrow & \theta_{i+3} & i = 3 \\
\theta_i & \rightarrow & -\theta_{i-3} & i = 4,5 \\
\theta_i & \rightarrow & \theta_{i-3} & i = 6
\end{array}
$$

As a consequence, for a symmetric gait we obtain the symmetries

$$
\begin{array}{rcll}
C^L_{i,j} & = & -C^I_{i+3,j} & i = 1,2 \\
C^L_{i,j} & = & C^I_{i+3,j} & i = 3 \\
C^I_{i,j} & = & C^R_{i,j} & j = 1,2,3,4,7,32,33,34 \\
C^I_{i,j} & = & -C^R_{i,j} & j = 5,6,31,35,36 \\
C^I_{i,j} & = & C^R_{i,j+4} & j = 10,37,40 \\
C^I_{i,j} & = & C^R_{i,j-4} & j = 11,14,41,44 \\
C^I_{i,j} & = & -C^R_{i,j+4} & j = 8,9,38,39 \\
C^I_{i,j} & = & -C^R_{i,j-4} & j = 12,13,42,43.
\end{array}
$$

This does not include the toe point symmetries, since the initial toe point state variables are not considered within the 16 gain parameters.

We proceed to check whether the gait found in the previous section satisfies the symmetry requirements. Figure 5.3 shows the time variation of the two hip angles over one and a half periods with the time history of the right hip angle q_{11} shifted half a period along the horizontal. The gait appears to be symmetric, but at a closer look there is a difference in the fourth decimal place.

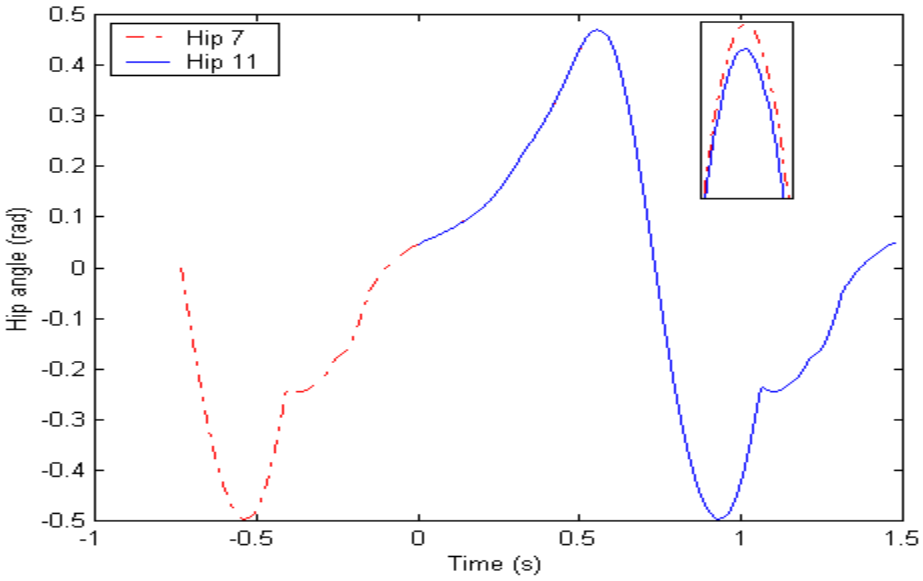

Figure 5.3: The orientation of the left and right hip with the right hip shifted half the time period for one stride.

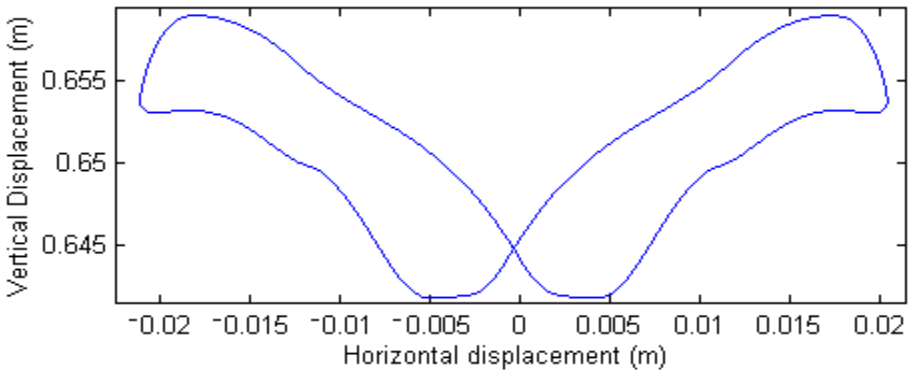

Figure 5.4: Vertical and horizontal motion of the center of mass of the torso, as seen from a frontal view.

Torso movement

A three dimensional walker is one that is allowed to move in all directions. When analyzing the walker found from the parameter studies preformed previously, we wanted to make sure that it indeed was moving in all three directions and that the hips were also moving along all three degrees of freedom. If the walker did not have motion in the hip joint, along all three degrees of freedom that might suggest that the corresponding spring and damper constants are too high.

Figure 5.4 shows the displacement of the torso's center of mass in the lateral plane. The torso is moving side-to-side as well as up and down, i.e., there is motion in the vertical q_2 direction and horizontal direction ($q_6 \neq 0$) showing that the walker moves in all three dimensions.

To check that there is motion in the new degrees of freedom of the hip joint, see Figures

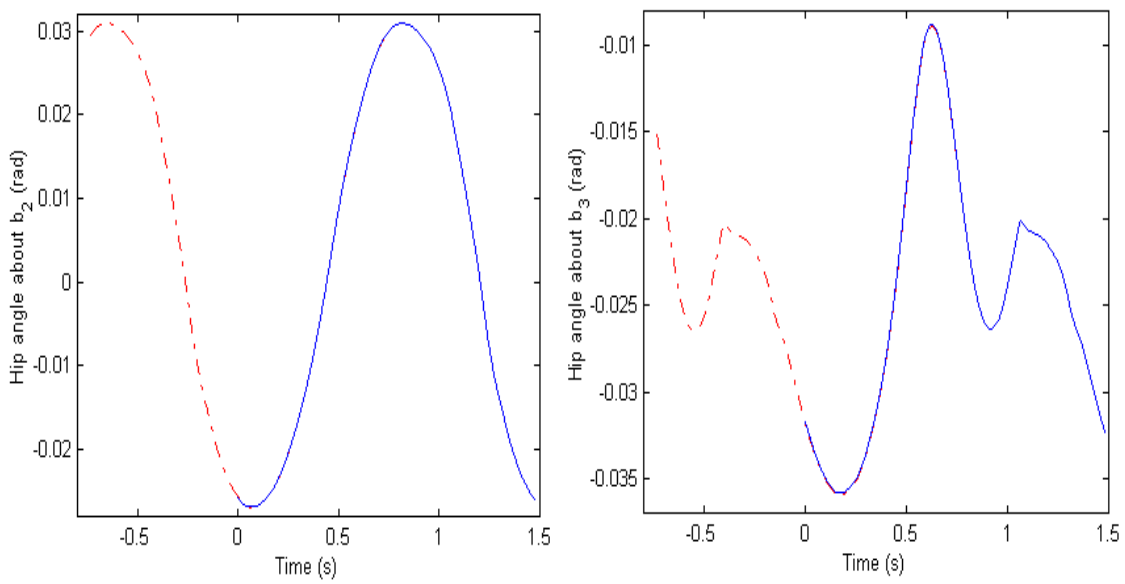

Figure 5.5: Changes in the new hip angles over one step with a time shift of half the period for the left leg's hip angles.

5.5 and 5.6. Figure 5.5 shows plots of the hip angles rotation (first panel) and abduction and adduction (second panel) as the walker takes one step. The left hip angles are again shift back half the time period to show that both hip joints have the same motion. Figure 5.6 is a plot of the motion of the hip projected on the space spanned by the rotation q_8 and q_{10} and abduction/adduction q_9 and $q11$ for both legs. The negative angle for the right hip angles are plotted because the walker is almost symmetric due to reflection of the left and right leg and torso. We see that the motion is almost identical for both hip joints.

5.3 Control of the new 3D walker

Refer to sections 2.9 and 4.4 for the notation and methodology for the control algorithms. The same formulation is used for this walker model except that there are more state variables and gain parameters making the matrices larger. The augmented state vector is of the form

$$\mathbf{y} = \begin{pmatrix} \mathbf{x_1} \\ \theta_L \\ \theta_R \\ \mathbf{x_2} \\ \mathbf{x_L}^* \\ \mathbf{x_R}^* \end{pmatrix}, \tag{5.1}$$

where $\mathbf{x_1}$ now has 30 components, θ has 6 components, and $\mathbf{x_2}$ has 14 components, describing a total of 50 state variables, while $\mathbf{x_L}^*$ and $\mathbf{x_R}^*$ have 44 components, describing the comparison state for each Poincaré section.

The matrices shown in Equations 4.2 and 4.3, for the Jacobian of the jump function used for delay control, have the same block structure, a similar structure also is applied for

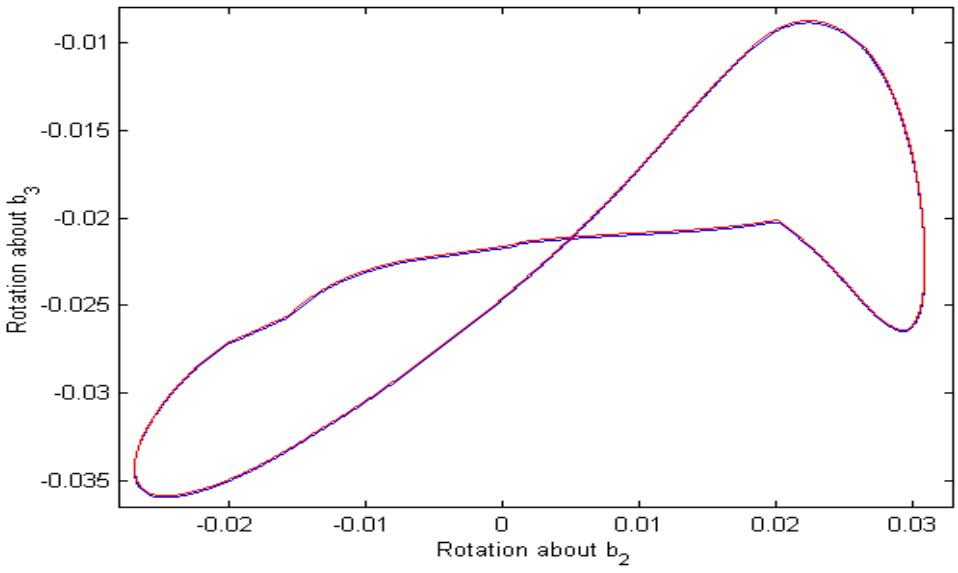

Figure 5.6: The motion of the hip due to the addition of more degrees-of-freedom.

the reference control. However, these matrices are now 138x138 due to the increased size of the augmented state vector. Recall that the ankle angles related the foot orientation to the inertial reference frame and are thus not constant during gait in the absence of control. Here, 24 of the possible 44 elements of each row of the C matrix were assumed non-zero. These nonzero elements are the gain parameters. It is important to choose gain parameters that will optimize the stability characteristics of the reference gait.

When calculating the gain parameters, we again assume that the gait of the walker is symmetric. We notice from the above discussion that the gait is slightly asymmetric. This assumption can still be used though it may be overly restrictive. We assume that the reference gait is symmetric with the right and left legs as well as the torso being identical under reflection. The symmetries listed previously will be taken into account when finding the gain parameters.

5.3.1 Numerical results of control

A walker was studied that in the absence of control had a largest-in-magnitude eigenvalue equaling 3.9. A limited search with Matlab's fminsearch function was done for both reference and delay control. When applying reference feedback control the largest eigenvalue was reduced to 1.22 while when using delay feedback control it was reduced it to 1.46. Figure 5.7 shows the prediction of the linear analysis ('dots') and the direct numerical simulation ('squares') of the walker with zero control ('circles'), reference feedback control (upper panel), and delay feedback control (lower panel). Errors shown on these plots refer to the deviation of the intersection of the actual trajectory with the $q_7 = 0$ discontinuity surface from that of the reference trajectory. Even though a set of gain parameters was not found that could reduce all the eigenvalues to less than one, it can be seen that with control the walking motion does not veer away from the reference trajectory as quickly as when no control is supplied.

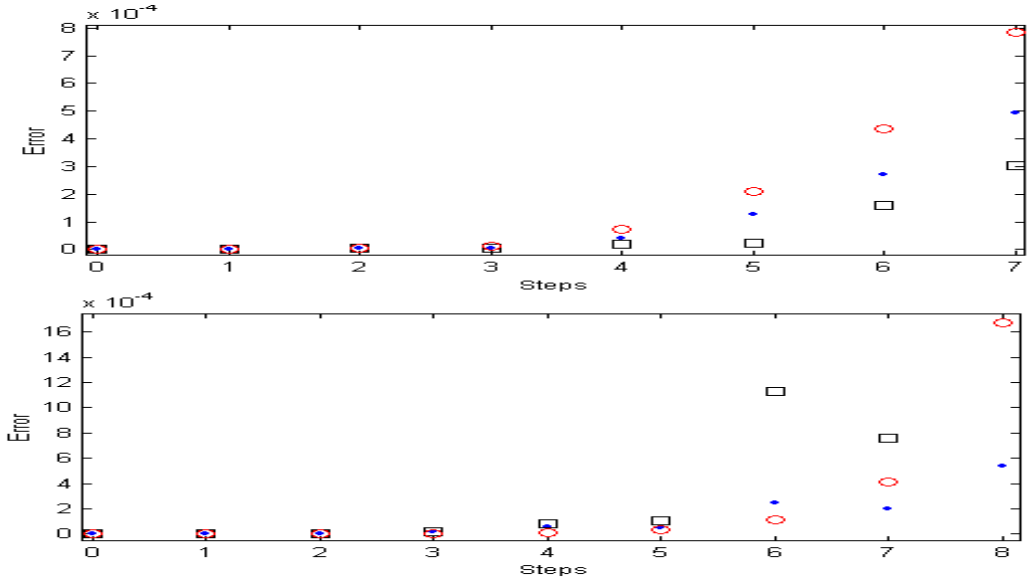

Figure 5.7: Predicted motions of the walker with reference feedback control (top panel) and delay feedback control (bottom panel).

Chapter 6

Conclusion and Recommendations

6.1 Summary

In this thesis, the goal was to find a walking motion of a passive bipedal walker that is more anthropomorphic than those that already exist. With this knowledge it is hoped to gain more information about the natural dynamics of human gait and possibly construct improved prosthetic devices or more energy-efficient walking robots.

The concept of extending a simple system to a more complex system by the use of a continuation method was discussed. Specifically, I considered the extension of an extended 2D walker model into a new 3D walker model with two new degrees-of-freedom in each hip. Starting with a 3D walker developed by Piiroinen, it was possible to find an extended 2D walker by performing parameter studies to reduce the hip width to zero and to make the feet symmetric about the sagittal plane. A set of extended 2D walking motions were found and their persistence and stability were checked for variations in slope and toe separation. In order to make the walker more anthropomorphic more degrees-of-freedom were added to the hip joints and the hip width was increased. We were able to show the existence of periodic gait that allowed for motion in all three dimensions as well as motion of the upper legs relative to the torso about all directions yielding a more human-like gait.

The periodic motions of these walkers were analyzed as periodic orbits of a dynamical system. These systems include discrete jumps necessitating the need for discontinuity mappings. To solve for periodic gaits the Newton-Raphson scheme was implemented.

Two control algorithms were also proposed in this thesis, reference feedback control and delay feedback control. The difference between the two control methods is their definition of a comparison state. The comparison state with reference feedback control remains constant throughout the motion and is given by the intersection of the reference trajectory with a control discontinuity surface. In contrast, with delay feedback control the comparison state is given by the previous intersection of the control discontinuity surface by the actual trajectory. When calculating the gain parameters the walker's motion was assumed to be symmetric about the sagittal plan, although this is not necessary. With both types of low-cost control we were able to increase the stability of walking motions for two walker models.

6.2 Recommendations for Future Research

There is still more work that could be accomplished on this project. The work presented here is a stepping stone along the way for many things that possibly will be achieved in the future. More work can be done with the new walker that has three degrees-of-freedom in each hip. The parameter study of increasing the hip width to make the geometry of the walker more anthropomorphic could be revisited. Also, parameter studies could be performed to find a range of ideal spring and damper constants that will allow for the persistence of gait while maintaining mobility about the new degrees-of-freedom.

The continuation method may be used to reduce the inclination of the ground so the walker will have gait on a flat surface, by possibly adding a spring to the hip area that will propel the leg forward [16]. Also, other extensions of the new 3D walker model may perhaps be incorporated to include a flexible ankle joint.

One could finally try and make an experimental working model of our mathematical model. Dr. Adolfsson tried to find an implementable 3D passive walker without success [1]. Currently, there is a graduate student who plans on building a prosthetic limb that consists of the lower limb and the foot. He plans on using the control algorithms discussed in this thesis to control the ankle for his limb.

Bibliography

[1] Adolfsson, J. *Passive Control of mechanical Systems Bipedal Walking and Autobalancing.* Doctoral Thesis. Royal Institute of Technology. Stockholm, 2001.

[2] Adolfsson, J., H. Dankowicz, A. Nordmark. '3D Passive Walking: Finding periodic gaits in the presence of discontinuities'. *Nonlinear Dynamics* 24, 2001: 205-229.

[3] Allison, H. *United States Patent No. 1,207,464: Toy*, 1916.

[4] Bechstein, B. and P. Uhlig. *London Patent No. 7453: Improvements in and relating to Toys*, 1912.

[5] Berns, K. *The Walking Machine Catalogue* . On line. Internet. 9 April 2003. Available: cognet.mit.edu/MITECS/Entry/boone

[6] *Biped Walking Robot.* On line. Humanoid Robotics Institute, Waseda University Internet. 9 April 2003. Available: humanoid.rise.waseda.ac.jp/booklet/kato04.html.

[7] Clowry, T. A Brief History of Biped Walking Machines. On line. COGS, University of Sussex, Falmer Brighton. Internet. 14 April 2003. Available: cogs.susx.ac.uk/lab/nlp/gazdar/teach/atc/1999/web/thomascl/#Section6

[8] Coleman, M. J. and A. Ruina. 'An Uncontrolled Toy That Can Walk But Cannot Stand Still'. *Physical Review Letters* 80.16, April 1998: 3658-3661.

[9] Dankowicz, J. Adolfsson, and A. B. Nordmark. 'Repetitive gait of passive bipedal mechanisms in a three-dimensional environment'. *Journal of Biomechanical Engineering* 123.1 2001: 40-46.

[10] *Delft Bio-robotics Laboratory.* On line. Delft University of Technology Internet. 9 April 2003. Available: wbmt.tudelft.nl/mms/dbl/dbl_overview_body.htm.

[11] *Delft Bio-robotics Laboratory.* On line. Delft University of Technology Internet. 9 April 2003. Available: wbmt.tudelft.nl/mms/dbl/dbl_research_body.htm.

[12] Fallis, G. *United States Patent No. 376,588: Walking Toy*, 1888.

[13] Formal'sky, A. M. *Ballistic Locomotion of a Bipedal: Design and Control of Two Bipedal Machines.* CISM Advanced School on Modeling and Simulation of Human and Walking Robots Locomotion, Udine 1996.

[14] Garcia, M., A. Chatterjee, A. Ruina, and M. Coleman. 'The Simplest Walking Model: Stability, Complexity, and Scaling'. *ASME Journal of Biomechanical Engineering* 120, 1998: 281-288.

[15] Garcia, M., A. Chatterjee, and A. Ruina. 'Efficiency, Speed, and Scaling of Passive Dynamical Bipedal Walking'. *Dynamics and Stability of Systems*. 15.2, 2000: 75-99.

[16] Howell, G. W. and J. Baillieul. 'Simple Controllable Walking Mechanism which Exhibit Bifurcations' *Processing of the 37th IEEE Conference on Decision and Control* 1998: 3027-3032.

[17] Humanoid Robot. On line. Honda Motor Co. Internet. 9 April 2003. Available: world.honda.com/robot/.

[18] Inman, T. Verne, H. Ralston, and F. Todd. *Human Walking*. Baltimore: Williams & Wilkins, 1981.

[19] Jerrelind, J. and H. J. Dankowicz. 'Low-cost Control of Impact Hammer Performance'. to appear in *Proceedings of ASME DETC'03* 2003.

[20] *Jouhou System Kougaku Laboratory*. On line. Jouhou System Kougaku Internet. 9 April 2003. Available: jsk.t.u-tokyo.ac.jp/

[21] Kuo, A. D. 'Stabilization of Lateral Motion in Passive Dynamic Walking'. *The International Journal of Robotics Research* 18.9, 1999: 917-930.

[22] Lennartsson, A. *Efficient Multibody Dynamics*. PHD thesis, Royal Institute of Technology, Department of Mechanics, Sweden, 1999.

[23] Lesser, M. *The Analysis of Complex Nonlinear Mechanical Systems*. World Scientific Publishing Co. Pte. Ltd., 1995.

[24] McGeer, T. 'Passive dynamic walking'. *International Journal of Robotics Research* 9, 1990: 62-82.

[25] McGeer, T. 'Passive walking with knees'. *Processings of the IEEE Conference on Robotics and Automation* 2, 1990 :1640-1681.

[26] McGeer, T. 'Dynamics and Control of Bipedal Locomotion'. *Journal of Theoretical Biology* 163, 1993: 277-314.

[27] *MIT Leg Laboratory*. On line. MIT Internet. 9 April 2003. Available: ai.mit.edu/projects/leglab/robots/robots.html

[28] Mochon, S. and T. McMahon. 'Ballistic Walking: An Improved Model'. *Mathematical Bioscience* 52, 1980: 241-260.

[29] Mombauer, K., M. Coleman, and A. Ruina 'Prediction of Stable Walking for a Toy That Cannot Stand Still'. *Physical Review E* 64.2, August 2001.

[30] *Neatstuff: Vintage Toys, Collectibles and Antiques.* On line. Internet. 8 April 2003. Available: neatstuff.net/space-robots/Space-robots.html.

[31] Piiroinen, P. *Passive Walking: Transition from 2D to 3D.* Licentiate Thesis. Royal Institute of Technology. Stockholm, 2000.

[32] Piiroinen, P, H. J. Dankowicz, and A. Nordmark. 'On a Normal-Form Analysis for a class of Passive Bipedal Walkers'. *International Journal of Bifurcation and Chaos* 11.9, 2001: 2411-2425.

[33] Piiroinen, P. *Recurrent Dynamics of Nonsmooth systems with Application to Human Gait.* PHD Thesis. Royal Institute of Technology. Stockholm, 2002.

[34] Piiroinen, P., H. J. Dankowicz, and A. Nordmark. 'Breaking Symmetries and Constraints: Transitions from 2D to 3D in Passive Walkers'. to appear in *Multibody system Dynamics.* 2003.

[35] Piiroinen, P. and H. J. Dankowicz. 'Low-cost Control of Repetitive Gait in Passive Bipedal Walkers'. in submission to *International Journal of Bifurcation and Chaos.* 2003.

[36] *Rampwalkers.Net.* On line. Internet. 8 April 2003. Available: angelfire.com/mi/rampwalkers/si.marx.directions.html

[37] Strogatz, S. H. *Nonlinear Dynamics and Chaos.* Cambridge, MA: Perseus Books Publishing, 1994.

[38] Whittle, M. W. *Gait Analysis: an Introduction, 2nd ed.* Butterwort-Heinemann, Oxford, 1996.

[39] Woodard, A. *Woodards Wood Products.* On line. Internet. 8 April 2003. Available: woodardswoodproducts.com/toys/walkers/display.htm

[40] *Yobotics: Creators of Legs.* On line. Yobotics. Internet. 14 April 2003. Available: http://www.yobotics.com/index.html

Appendix A

Code for example 2

vectornewton Function
function [u,x]=vectornewton (func,u0,x0,N,deltau)

% Function for performing Newton-Raphson iterations to locate
% equilibria for the dynamical system given by the vector field
% specified by the function 'func' starting with an initial known
% equilibrium at (u0,x0) and varying u from u0 in N steps with
% stepsize deltau.

% The program returns a vector with the values of u, the values of x.
$u = u0 : deltau : u0 + N * deltau;$
$x = zeros(length(x0), N + 1);$

$for\ i = 1 : N + 1$
$x(:, i) = (i\ \ = 1) * x(:, i - 1 + (i == 1)) + (i == 1) * x0;$
$deltax = ones(length(x0), 1);$
$f = ones(length(x0), 1);$
$while\ norm(deltax, 1) > 10^{-}5|norm(f, 1) > 10^{-}5$
$f = feval(func, 0, x(:, i),'', u(i));$
$dfx = feval(func, 0, x(:, i),' jacobian', u(i));$
$deltax = -inv(dfx) * f;$
$x(:, i) = x(:, i) + deltax;$
end
$x(:, i);$
end

ode45_1DOFsystem Function
function varargout = ode45_1DOFsystem (t,y,flag,p1)

% This file is called to return a scalar function, the jacobian of the function, or
% the initial/default conditions depending on what flag is called when the file is
% called.

switch flag
case '' % Return dy/dt = f(t,y).
varargout1 = f(t,y,p1);
case 'init' % Return default [tspan,y0,options].
[varargout1:3] = init;
case 'jacobian' % Return Jacobian matrix df/dy.

```
varargout1 = jacobian(t,y,p1);
case 'events'                                   % Return [value,isterminal,direction].
[varargout1:3] = events(t,y,p1);
otherwise
error(['Unknown flag ''' flag '''.']);
end
%% ------------------------------------------------------------
function dydt = f(t,y,p1)
dydt = [p1 - y(1)^2; ...                        %vector field
- 2 * y(1) * y(2)];                             %variational equation
%% ------------------------------------------------------------
function dfdy = jacobian(t, y, p1)
dfdy = [-2 * y(1)];
%% ------------------------------------------------------------
function [tspan,y0,options] = init()
tspan = [0 100];                                %time span
y0 = [1;1];                                     %initial values for y(1) and y(2)
options = odeset('OutputSel',1,'RelTol',1e-8,'AbsTol',1e-10,'Events','off');
%% ------------------------------------------------------------
function [value,isterminal,direction] = events(t,y,p1)
value = [y(1)-1];                               %H=0 at y(1)=1
isterminal = [0];                               %terminates after it hits Poincare' section
direction = [1];                                %positive or negative direction
```

Appendix B

Code for example 3

ode45_moreDOFsystem Function
function varargout = ode45_moreDOFsystem (t,y,flag,p1)

% This file is called to return a vector function, the jacobian of the
% function, or the initial/default conditions depending on what flag
% is called when the file is called.

```
switch flag
case ''                                  % Return dy/dt = f(t,y).
varargout1 = f(t,y,p1);
case 'init'                              % Return default [tspan,y0,options].
[varargout1:3] = init;
case 'jacobian'                          % Return Jacobian matrix df/dy.
varargout1 = jacobian(t,y,p1);
case 'events'                            % Return [value,isterminal,direction].
[varargout1:3] = events(t,y,p1);
otherwise
error(['Unknown flag '' flag ''.']);
end
%% ---------------------------------------------------------
function dydt = f(t,y,p1)
dydt = [p1*y(1)^2 - y(1)*y(2); -y(2) + y(1)^2; ...      %vectorfield
(2*p1*y(1) - y(2))*y(3) - y(1)*y(5); (2*p1*y(1) - y(2))*y(4) - y(1)*y(6); ...
2*y(1)*y(3) - y(5); 2*y(1)*y(4) - y(6)];                %variationalequation
%% ---------------------------------------------------------
functiondfdy = jacobian(t, y, p1)
dfdy = [2*p1*y(1) - y(2) - y(1); 2*y(1) - 1];
%% ---------------------------------------------------------
function [tspan,y0,options] = init()
tspan = [0 100];
y0 = [1;1;1;0;0;1];
options = odeset('OutputSel',1,'RelTol',1e-8,'AbsTol',1e-8,'Events','on');
%% ---------------------------------------------------------
function [value,isterminal,direction] = events(t,y,p1)
value = [y(1)-1];
isterminal = [0];
direction = [1];
```

Hopf Function

```
function varargout=Hopf(t,y,flag,p1)

switch flag
case ''                                   % Return dy/dt = f(t,y).
varargout1 = F(t,y,p1);
case 'jacobian'                           % Return Jacobian matrix df/dy.
varargout1 = jacobian(t,y,p1);
case 'Pjacobian'                          % Return Jacobian matrix df/dy.
varargout1 = Pjacobian(t,y,p1);
otherwise
error(['Unknown flag ''' flag '''.']);
end
%%-------------------------
function Poincare=F(t,y,p1)
options = odeset('OutputSel',1,'RelTol',1e-5,'AbsTol',1e-6,'Events','on');
[t,x,te,xe]=ode45('ode45_moreDOFsystem',[0 40],[1;y(2); 1; 0; 0; 1],options,p1);
Poincare=[xe(2,1)-1 ; xe(2,2)-y(2)];
%%-------------------------
function Poincarediff=jacobian(t,y,p1)
options = odeset('OutputSel',1,'RelTol',1e-5,'AbsTol',1e-6,'Events','on');
[t,x,te,xe]=ode45('ode45_moreDOFsystem',[0 40],[1;y(2); 1; 0; 0; 1],options,p1);
hx=[1 0];
phix=[xe(2,3:4);xe(2,5:6)];
tempf=ode45_moreDOFsyste(0,xe(2,:),'',p1);
f=[tempf(1:2)];
Poincarediff=(phix-(f*hx*phix)/(hx*f))-eye(2);
%%-------------------------
function Poincarejakobian=Pjacobian(t,y,p1)
options = odeset('OutputSel',1,'RelTol',1e-5,'AbsTol',1e-6,'Events','on');
[t,x,te,xe]=ode45('ode45_moreDOFsystem',[0 40],[1;y(2); 1; 0; 0; 1],options,p1);
hx=[1 0];
phix=[xe(2,3:4);xe(2,5:6)];
tempf=ode45_moreDOFsyste(0,xe(2,:),'',p1);
f=[tempf(1:2)];
Poincarejakobian=(phix-(f*hx*phix)/(hx*f));
```

Appendix C

Code for example 4

```
function output = jump_in_y(x0,y0,deltax0,deltay0,tend)

options = odeset('OutputSel',1,'RelTol',1e-8,'AbsTol',1e-8,'Events','on');

T=[];
X=[];
tend0=0;
yend0=1.1 ;
gx=[1 0; 0 -1] ;
hx=[1 0];

[t,x,te,ye]=ode45('ode45_moreDOFsyste',[0 tend],[x0;y0;1;0;0;1],options,1.5);
T=[T; t];
X=[X;x];
xin=ye(1);
yin=ye(2);
tau_x0=te;
xout=xin;
yout=2*1.5-yin;
jacin=[ye(3) ye(4); ye(5) ye(6)];
fin= ode45_moreDOFsyste(tau_x0, ye, '',1.5);
fout= ode45_moreDOFsyste(tau_x0, [xout;yout;1;0;0;1], '',1.5);

[t,x,te,ye]=ode45('ode45_moreDOFsyste',[tau_x0 tend],[xout;yout;1;0;0;1],options,1.5);
T=[T; t];
X=[X;x];
jacout=[x(end,3) x(end,4); x(end,5) x(end,6)];

plot(X(:,1),X(:,2),'r.')

phi_y=jacout*(gx+(fout(1:2)-gx*fin(1:2))*hx/(hx*fin(1:2)))*jacin;
[t,x,te,ye]=ode45('ode45_moreDOFsyste',[0 tend],[x0+deltax0;y0+deltay0;1;0;0;1],options,1.5);
[t,x,te,ye]=ode45('ode45_moreDOFsyste',[te tend],[ye(1);2*1.5-ye(2);1;0;0;1],options,1.5);
output=phi_y*[deltax0;deltay0]-x(end,1:2)'+X(end,1:2)';
```

Appendix D

Maple/Sophia code

Maple/Sophia code for walker model with three degrees of freedom at each hip joint.

D.1 Sophia library

```
> restart; ## 02-06-30 ##;

   > read
   > `/afs/mech.kth.se/home/mech/petri/Arb_temp/Everest/Maple/sophiaV5`;
```

D.2 Kde

```
   > &kde(14);
   > dependsTime(seq(p||g,g=1..16));
   > dependsTime(seq(theta||g,g=1..6));
```

D.3 Frame relations

Rotation from inertial body N to upper body B

```
   > chainSimpRot([N,T1,3,q6],[T1,T2,2,q5],[T2,B,1,q4]):
```

Rotation from intermeditiary frame T2 to upper leg UL1 to lower leg LL1

```
   > chainSimpRot([N,T3,1,q7],[T3,T4,3,q8],[T4,UL1,2,q9],[UL1,LL1,1,q10]):
   > ##chainSimpRot([T2,UL1,1,q7],[UL1,LL1,1,q8]):
```

Rotation from intermeditiary frame T2 to upper leg UL2 to lower leg LL2

```
   > chainSimpRot([N,T5,1,q11],[T5,T6,3,q12],[T6,UL2,2,q13],[UL2,LL2,1,q14
   > ]):
```

Rotation from lower leg LL1 foot F1

```
   > chainSimpRot([N,T7,3,theta1],[T7,T8,2,theta2],[T8,F1,1,theta3]):
```

Rotation from lower leg LL2 foot F2

```
   > chainSimpRot([N,T9,3,theta4],[T9,T10,2,theta5],[T10,F2,1,theta6]):
   > R_B_UL1:= simplify(Rmx(B,UL1)):
   > R_B_UL2:= simplify(Rmx(B,UL2)):
   > R_N_UL1:=Rmx(N,UL1):
   > R_N_UL2:=Rmx(N,UL2):
   > ##Rmx(N,F2);
```

```
> matrix([[cos(q12)*cos(q13), -sin(q12), cos(q12)*sin(q13)],
> [cos(q11)*sin(q12)*cos(q13)+sin(q11)*sin(q13), cos(q11)*cos(q12),
> cos(q11)*sin(q12)*sin(q13)-sin(q11)*cos(q13)],
> [sin(q11)*sin(q12)*cos(q13)-cos(q11)*sin(q13), sin(q11)*cos(q12),
> sin(q11)*sin(q12)*sin(q13)+cos(q11)*cos(q13)]]);
```

D.4 Geometry

```
> rbody_hip_center:=N &ev [q1,q2,q3];
> r_torso_cm_rel:=B &ev [rx,ry,rz];
> rleg1_jointpos:=T2 &ev [rl11,0,0];
> rleg2_jointpos:=T2 &ev [rl21,0,0];
> rleg1_upper_cm_rel:= UL1 &ev [rl1u1,rl1u2,rl1u3];
> rleg2_upper_cm_rel:= UL2 &ev [rl2u1,rl2u2,rl2u3];
> rleg1_upper_jointpos_rel:= UL1 &ev [0,l1u,0];
> rleg2_upper_jointpos_rel:= UL2 &ev [0,l2u,0];
> rleg1_lower_cm_rel:= LL1 &ev [rl1l1,rl1l2,rl1l3];
> rleg2_lower_cm_rel:= LL2 &ev [rl2l1,rl2l2,rl2l3];
> rbody_cm:=mkc(rbody_hip_center,r_torso_cm_rel);
> rleg1upper_cm:=mkc(rbody_hip_center,rleg1_jointpos,rleg1_upper_cm_rel
> ):
> rleg2upper_cm:=mkc(rbody_hip_center,rleg2_jointpos,rleg2_upper_cm_rel
> ):
> rleg1lower_cm:=mkc(rbody_hip_center,rleg1_jointpos,rleg1_upper_jointp
> os_rel,rleg1_lower_cm_rel):
> rleg2lower_cm:=mkc(rbody_hip_center,rleg2_jointpos,rleg2_upper_jointp
> os_rel,rleg2_lower_cm_rel):
```

D.5 New kde's

```
> v_body_hipcenter:= T2 &ev [u1,u2,u3];
> omega_body:=T2 &ev [u4,u5,u6];
> omega_UL1:=UL1 &ev [u7,u8,u9];
> omega_UL2:=UL2 &ev [u11,u12,u13];
> v_body_hipcenter_qt:= diffFrameTime(rbody_hip_center,N);
> omega_body_qt:=N &aV B;
> omega_UL1_qt:=N &aV UL1;
> omega_UL2_qt:=N &aV UL2;

> LL1_omega_F1:=N &to (LL1 &aV F1);
> LL2_omega_F2:=N &to (LL2 &aV F2);
> tt1 := {seq(LL1_omega_F1 &c i=0,i=1..3),seq(LL2_omega_F2 &c
> i=0,i=1..3)};
```

Note that the kde for toe contact points are included

```
> ##ikde:={seq((T2 &to v_body_hipcenter &c i) = (T2 &to
> v_body_hipcenter_qt) &c i,i=1..3), seq((T2 &to omega_body &c i) = (T2
> &to omega_body_qt) &c
> i,i=1..3),seq(p.(2*i-1).t=-q1t,i=1..8),seq(p.(2*i).t=-q3t,i=1..8),seq(
> seq(seq(f.i.j.k.t=0,i=1..2),j=1..2),k=[x,y,z])};
> ikde:={seq(((T2 &to v_body_hipcenter) &c i) = (T2 &to
> v_body_hipcenter_qt) &c i,i=1..3),
> seq(((T2 &to omega_body) &c i) = ((T2 &to omega_body_qt) &c
> i),i=1..3),
> seq(((UL1 &to omega_UL1) &c i) = ((UL1 &to omega_UL1_qt) &c
> i),i=1..3),
> seq(((UL2 &to omega_UL2) &c i) = ((UL2 &to omega_UL2_qt) &c
> i),i=1..3),
> seq(p||(2*i-1)||t=-q1t,i=1..8),
> seq((T2 &to omega_body &c i) = (T2 &to omega_body_qt) &c i,i=1..3),
> seq(p||(2*i-1)||t=-q1t,i=1..8),seq(p||(2*i)||t=-q3t,i=1..8),q10t=u10,q
> 14t=u14} union tt1;
> kde_new := simplify(solve(ikde,
>{seq(q||i||t,i=1..14),seq(p||i||t,i=1..16),seq(theta||i||t,i=1..6),se
> q(theta||i||t,i=1..6)}));
> seq(theta||i||t,i=1..6);
```

D.6 Velocities and angular velocities

```
> v_body_cm := ccpt(map(Esimplify,subs(kde_new,cdft(rbody_cm,N)))):
> vleg1upper_cm :=
> ccpt(map(Esimplify,subs(kde_new,cdft(rleg1upper_cm,N)))):
> vleg2upper_cm :=
> ccpt(map(Esimplify,subs(kde_new,cdft(rleg2upper_cm,N)))):
> vleg1lower_cm :=
> ccpt(map(Esimplify,subs(kde_new,cdft(rleg1lower_cm,N)))):
> vleg2lower_cm :=
> ccpt(map(Esimplify,subs(kde_new,cdft(rleg2lower_cm,N)))):
> omega_leg1upper := &simp subs(kde_new,N &aV UL1);
> omega_leg2upper := &simp subs(kde_new,N &aV UL2);
> omega_leg1lower := &simp subs(kde_new,N &aV LL1):
> omega_leg2lower := &simp subs(kde_new,N &aV LL2):
```

D.7 Momentum and angular momentum

```
> pbody:=csm(m_body, v_body_cm):
> pleg1upper:=csm(ml1u,vleg1upper_cm):
> pleg2upper:=csm(ml2u,vleg2upper_cm):
> pleg1lower:=csm(ml1l,vleg1lower_cm):
> pleg2lower:=csm(ml2l,vleg2lower_cm):

> I_body:=EinertiaDyad(I_B11,I_B22,I_B33,I_B12,I_B13,I_B23,B):
> hbody:=I_body &o omega_body:

> I_leg1upper:=EinertiaDyad(I_LU111,I_LU122,I_LU133,I_LU112,I_LU113,I_L
> U123,UL1):
> hleg1upper:=I_leg1upper &o omega_leg1upper:

> I_leg2upper:=EinertiaDyad(I_LU211,I_LU222,I_LU233,I_LU212,I_LU213,I_L
> U223,UL2):
```

```
> hleg2upper:=I_leg2upper &o omega_leg2upper:

> I_leg1lower:=EinertiaDyad(I_LL111,I_LL122,I_LL133,I_LL112,I_LL113,I_L
> L123,LL1):
> hleg1lower:=I_leg1lower &o omega_leg1lower:

> I_leg2lower:=EinertiaDyad(I_LL211,I_LL222,I_LL233,I_LL212,I_LL213,I_L
> L223,LL2):
> hleg2lower:=I_leg2lower &o omega_leg2lower:
```

D.8 Timediff of Mom and ang. mom

```
> ptbody     :=map(Esimplify,subs(kde_new,cdft(pbody,N))):
> ptleg1upper:=map(Esimplify,subs(kde_new,cdft(pleg1upper,N))):
> ptleg2upper:=map(Esimplify,subs(kde_new,cdft(pleg2upper,N))):
> ptleg1lower:=map(Esimplify,subs(kde_new,cdft(pleg1lower,N))):
> ptleg2lower:=map(Esimplify,subs(kde_new,cdft(pleg2lower,N))):

> htbody:=mkc( &simp subs(kde_new,N &fdt hbody) ):
> htleg1upper:=mkc( &simp subs(kde_new,N &fdt hleg1upper) ):
> htleg2upper:=mkc( &simp subs(kde_new,N &fdt hleg2upper) ):
> htleg1lower:=mkc( &simp subs(kde_new,N &fdt hleg1lower) ):
> htleg2lower:=mkc( &simp subs(kde_new,N &fdt hleg2lower) ):
```

D.9 Assembly of K velocity vector and tangent vector creation

```
> vK:=[ v_body_cm, vleg1upper_cm, vleg2upper_cm, vleg1lower_cm,
> vleg2lower_cm,
> mkc(omega_body), mkc(omega_leg1upper), mkc(omega_leg2upper),
> mkc(omega_leg1lower), mkc(omega_leg2lower) ]:
> tauK:=Kctau(vK,[seq(u||i,i=1..14)]):
> ## tauK:=Kctau(vK,[seq(u.i,i=1..10)]):
```

D.10 Toe forces

D.10.1 Common stuff

Procedure for finding out what variables an expression depends on given a list of possible variables

```
> check_args:=proc(expr,args)
> local temp_args,i;
> temp_args:=[]:
> for i in args do
> if depends(expr,i) then temp_args:=[op(temp_args),i]; fi;
> od;
> RETURN(op(temp_args));
> end:
> args:=seq(q||i,i=1..14),seq(theta||i,i=1..6),seq(u||i,i=1..14);
> ##args:=seq(q.i,i=1..10),seq(theta.i,i=1..6),seq(u.i,i=1..10);
```

D.10.2 First foot first toe 11

Position & velocity of toe11

```
> ##rtoe11:=&simp (N &to (rbody_hip_center &++ rleg1_jointpos &++
> rleg1_upper_jointpos_rel &++ (LL1 &ev [f11x,f11y,f11z])) ):
> rtoe11:=&simp (N &to (rbody_hip_center &++ rleg1_jointpos &++
> rleg1_upper_jointpos_rel &++ (LL1 &ev [0,l1l,0]) &++ (F1 &ev
> [xt11,yt11,zt11])) );
> vtoe11:=&simp (N &to subs(kde_new,N &fdt rtoe11)):
```

Find the variables that each coord. & vel. of the toepos depends on

```
> x11_args:=check_args((rtoe11 &c 1)-q1,[args]):
> y11_args:=check_args(rtoe11 &c 2,[args]):
> z11_args:=check_args((rtoe11 &c 3)-q3,[args]):
> vx11_args:=check_args(vtoe11 &c 1,[args]):
> vy11_args:=check_args(vtoe11 &c 2,[args]):
> vz11_args:=check_args(vtoe11 &c 3,[args]):
```

Torque calculated as vector from cm of lower leg to contact point cross applied force

```
> ##toe11_cmll1_cross_f:=&simp ((N &to ((LL1 &ev [f11x,f11y,f11z]) &--
> (LL1 &ev [rl1l1,rl1l2,rl1l3]))) &xx
> ##(N &ev [toefx(stat11,x11(x11_args),vx11(vx11_args),kX,dX,p1),
> ##        toefy(stat11,y11(y11_args),vy11(vy11_args),kY,dY),
> ##        toefz(stat11,z11(z11_args),vz11(vz11_args),kZ,dZ,p2)])):
> toe11_cmll1_cross_f:=&simp ((N &to ((LL1 &ev [0,l1l,0]) &++ (F1 &ev
> [xt11,yt11,zt11]) &-- (LL1 &ev [rl1l1,rl1l2,rl1l3]))) &xx
> (N &ev [toefx(stat11,x11(x11_args),vx11(vx11_args),kX,dX,p1),
> toefy(stat11,y11(y11_args),vy11(vy11_args),kY,dY),
> toefz(stat11,z11(z11_args),vz11(vz11_args),kZ,dZ,p2)])):
```

Common subexpression used later in calculations

```
> toe11_list:=[x11=(rtoe11 &c 1)-q1,y11=rtoe11 &c 2,z11=(rtoe11 &c
> 3)-q3,vx11=vtoe11 &c 1,vy11=vtoe11 &c 2,vz11=vtoe11 &c 3,
> fx11=toefx(stat11,x11(x11_args),vx11(vx11_args),kX,dX,p1),
> fy11=toefy(stat11,y11(y11_args),vy11(vy11_args),kY,dY),
> fz11=toefz(stat11,z11(z11_args),vz11(vz11_args),kZ,dZ,p2),
> tx11=toe11_cmll1_cross_f &c 1,ty11=toe11_cmll1_cross_f &c
> 2, tz11=toe11_cmll1_cross_f &c 3]:
```

Substitutions used in stability calculations

```
> dset11:=[
> D[2](toefx)(stat11,x11(x11_args),vx11(vx11_args),kX,dX,p1)=
> toefx_x(stat11,x11,vx11,kX,dX,p1),
> D[3](toefx)(stat11,x11(x11_args),vx11(vx11_args),kX,dX,p1)=
> toefx_vx(stat11,x11,vx11,kX,dX,p1),
> D[2](toefy)(stat11,y11(y11_args),vy11(vy11_args),kY,dY)=
> toefy_y(stat11,y11,vy11,kY,dY),
> D[3](toefy)(stat11,y11(y11_args),vy11(vy11_args),kY,dY)=
> toefy_vy(stat11,y11,vy11,kY,dY),
> D[2](toefz)(stat11,z11(z11_args),vz11(vz11_args),kZ,dZ,p2)=
> toefz_z(stat11,z11,vz11,kZ,dZ,p2),
> D[3](toefz)(stat11,z11(z11_args),vz11(vz11_args),kZ,dZ,p2)=
> toefz_vz(stat11,z11,vz11,kZ,dZ,p2),
> D[6](toefx)(stat11,x11(x11_args),vx11(vx11_args),kX,dX,p1)=
> toefx_p(stat11,x11,vx11,kX,dX,p1),
> D[6](toefz)(stat11,z11(z11_args),vz11(vz11_args),kZ,dZ,p2)=
> toefz_p(stat11,z11,vz11,kZ,dZ,p2),
> seq(diff(x11(x11_args),i)=dx11d||i,i=x11_args),
> seq(diff(vx11(vx11_args),i)=dvx11d||i,i=vx11_args),
> x11(x11_args)=x11,
> vx11(vx11_args)=vx11,
> seq(diff(y11(y11_args),i)=dy11d||i,i=y11_args),
> seq(diff(vy11(vy11_args),i)=dvy11d||i,i=vy11_args),
> y11(y11_args)=y11,
> vy11(vy11_args)=vy11,
> seq(diff(z11(z11_args),i)=dz11d||i,i=z11_args),
> seq(diff(vz11(vz11_args),i)=dvz11d||i,i=vz11_args),
> z11(z11_args)=z11,
> vz11(vz11_args)=vz11];
```

D.10.3 First foot second toe 12

```
> x12_args:=x11_args:
> y12_args:=y11_args:
> z12_args:=z11_args:
> vx12_args:=vx11_args:
> vy12_args:=vy11_args:
> vz12_args:=vz11_args:
> #from_toe1_to_toe2:={f11x=f12x,f11y=f12y,f11z=f12z};
> #x12_args:=seq(subs(from_toe1_to_toe2,[x11_args])[i],i=1..nops([x11_ar
> gs]));
> #y12_args:=seq(subs(from_toe1_to_toe2,[y11_args])[i],i=1..nops([y11_ar
> gs]));
> #z12_args:=seq(subs(from_toe1_to_toe2,[z11_args])[i],i=1..nops([z11_ar
> gs]));
> #vx12_args:=seq(subs(from_toe1_to_toe2,[vx11_args])[i],i=1..nops([vx11
> _args]));
> #vy12_args:=seq(subs(from_toe1_to_toe2,[vy11_args])[i],i=1..nops([vy11
> _args]));
> #vz12_args:=seq(subs(from_toe1_to_toe2,[vz11_args])[i],i=1..nops([vz11
> _args]));
```

```
> from_toe11_to_toe12_sset:={
> xt11=xt12,yt11=yt12,zt11=zt12,
> rl1l1=rl1l1,rl1l2=rl1l2,rl1l3=rl1l3,
> toefx(stat11,x11(x11_args))=toefx(stat12,x12(x12_args)),
> toefy(stat11,y11(y11_args))=toefy(stat12,y12(y12_args)),
> toefz(stat11,z11(z11_args))=toefz(stat12,z12(z12_args)),
> x11=x12,vx11=vx12,
> y11=y12,vy11=vy12,
> z11=z12,vz11=vz12,
> stat11=stat12,
> p1=p3,p2=p4,
> seq(dx11d||x11_args[i]=dx12d||x12_args[i],i=1..nops([x12_args])),
> seq(dvx11d||vx11_args[i]=dvx12d||vx12_args[i],i=1..nops([vx12_args])),
> seq(dy11d||y11_args[i]=dy12d||y12_args[i],i=1..nops([y12_args])),
> seq(dvy11d||vy11_args[i]=dvy12d||vy12_args[i],i=1..nops([vy12_args])),
> seq(dz11d||z11_args[i]=dz12d||z12_args[i],i=1..nops([z12_args])),
> seq(dvz11d||vz11_args[i]=dvz12d||vz12_args[i],i=1..nops([vz12_args])),
> fx11=fx12,fy11=fy12,fz11=fz12,
> tx11=tx12,ty11=ty12,tz11=tz12
>}:
> rtoe12:=subs(from_toe11_to_toe12_sset,rtoe11):
> vtoe12:=subs(from_toe11_to_toe12_sset,vtoe11):
> toe12_cmll1_cross_f:=subs(from_toe11_to_toe12_sset,toe11_cmll1_cross_f
> ):
> toe12_list:=subs(from_toe11_to_toe12_sset,toe11_list):
> dset12:=subs(from_toe11_to_toe12_sset,dset11):
```

First foot third toe 13

Position & velocity of toe13

```
> ##rtoe13:=&simp (N &to (rbody_hip_center &++ rleg1_jointpos &++
> rleg1_upper_jointpos_rel &++ (LL1 &ev [xt13,yt13,zt13]))):
> rtoe13:=&simp (N &to (rbody_hip_center &++ rleg1_jointpos &++
> rleg1_upper_jointpos_rel &++ (LL1 &ev [0,l1l,0]) &++ (F1 &ev
> [xt13,yt13,zt13]))):
> vtoe13:=&simp (N &to subs(kde_new,N &fdt rtoe13)):
```

Find the variables that each coord. & vel. of the toepos depends on

```
> x13_args:=check_args((rtoe13 &c 1)-q1,[args]):
> y13_args:=check_args(rtoe13 &c 2,[args]):
> z13_args:=check_args((rtoe13 &c 3)-q3,[args]):
> vx13_args:=check_args(vtoe13 &c 1,[args]):
> vy13_args:=check_args(vtoe13 &c 2,[args]):
> vz13_args:=check_args(vtoe13 &c 3,[args]):
```

Torque calculated as vector from cm of lower leg to contact point cross applied force

```
> #toe13_cmll1_cross_f:=&simp ((N &to ((LL1 &ev [xt13,yt13,zt13]) &--
> (LL1 &ev [rl1l1,rl1l2,rl1l3]))) &xx
> #(N &ev [toefx(stat13,x13(x13_args),vx13(vx13_args),kX,dX,p5),
> #        toefy(stat13,y13(y13_args),vy13(vy13_args),kY,dY),
> #        toefz(stat13,z13(z13_args),vz13(vz13_args),kZ,dZ,p6)]):
> toe13_cmll1_cross_f:=&simp ((N &to ((LL1 &ev [0,l1l,0]) &++ (F1 &ev
> [xt13,yt13,zt13]) &-- (LL1 &ev [rl1l1,rl1l2,rl1l3]))) &xx
> (N &ev [toefx(stat13,x13(x13_args),vx13(vx13_args),kX,dX,p5),
> toefy(stat13,y13(y13_args),vy13(vy13_args),kY,dY),
> toefz(stat13,z13(z13_args),vz13(vz13_args),kZ,dZ,p6)]):
```

Common subexpression used later in calculations

```
> toe13_list:=[x13=(rtoe13 &c 1)-q1,y13=rtoe13 &c 2,z13=(rtoe13 &c
> 3)-q3,vx13=vtoe13 &c 1,vy13=vtoe13 &c 2,vz13=vtoe13 &c 3,
> fx13=toefx(stat13,x13(x13_args),vx13(vx13_args),kX,dX,p5),
> fy13=toefy(stat13,y13(y13_args),vy13(vy13_args),kY,dY),
> fz13=toefz(stat13,z13(z13_args),vz13(vz13_args),kZ,dZ,p6),
> tx13=toe13_cmll1_cross_f &c 1,ty13=toe13_cmll1_cross_f &c
> 2, tz13=toe13_cmll1_cross_f &c 3]:
```

Substitutions used in stability calculations

```
> dset13:=[
> D[2](toefx)(stat13,x13(x13_args),vx13(vx13_args),kX,dX,p5)=toefx_x(sta
> t13,x13,vx13,kX,dX,p5),
> D[3](toefx)(stat13,x13(x13_args),vx13(vx13_args),kX,dX,p5)=toefx_vx(st
> at13,x13,vx11,kX,dX,p5),
> D[2](toefy)(stat13,y13(y13_args),vy13(vy13_args),kY,dY)=toefy_y(stat13
> ,y13,vy13,kY,dY),
> D[3](toefy)(stat13,y13(y13_args),vy13(vy13_args),kY,dY)=toefy_vy(stat1
> 3,y13,vy13,kY,dY),
> D[2](toefz)(stat13,z13(z13_args),vz13(vz13_args),kZ,dZ,p6)=toefz_z(sta
> t13,z13,vz13,kZ,dZ,p6),
> D[3](toefz)(stat13,z13(z13_args),vz13(vz13_args),kZ,dZ,p6)=toefz_vz(st
> at13,z13,vz13,kZ,dZ,p6),
> D[6](toefx)(stat13,x13(x13_args),vx13(vx13_args),kX,dX,p5)=toefx_p(sta
> t13,x13,vx13,kX,dX,p5),
> D[6](toefz)(stat13,z13(z13_args),vz13(vz13_args),kZ,dZ,p6)=toefz_p(sta
> t13,z13,vz13,kZ,dZ,p6),
> seq(diff(x13(x13_args),i)=dx13d||i,i=x13_args),
> seq(diff(vx13(vx13_args),i)=dvx13d||i,i=vx13_args),
> x13(x13_args)=x13,
> vx13(vx13_args)=vx13,
> seq(diff(y13(y13_args),i)=dy13d||i,i=y13_args),
> seq(diff(vy13(vy13_args),i)=dvy13d||i,i=vy13_args),
> y13(y13_args)=y13,
> vy13(vy13_args)=vy13,
> seq(diff(z13(z13_args),i)=dz13d||i,i=z13_args),
> seq(diff(vz13(vz13_args),i)=dvz13d||i,i=vz13_args),
> z13(z13_args)=z13,
> vz13(vz13_args)=vz13
> ]:
```

D.10.4 First foot second toe 14

```
> x14_args:=x13_args:
> y14_args:=y13_args:
> z14_args:=z13_args:
> vx14_args:=vx13_args:
> vy14_args:=vy13_args:
> vz14_args:=vz13_args:
```

```
> from_toe13_to_toe14_sset:={
> xt13=xt14,yt13=yt14,zt13=zt14,
> rl1l1=rl1l1,rl1l2=rl1l2,rl1l3=rl1l3,
> toefx(stat13,x13(x13_args))=toefx(stat14,x14(x14_args)),
> toefy(stat13,y13(y13_args))=toefy(stat14,y14(y14_args)),
> toefz(stat13,z13(z13_args))=toefz(stat14,z14(z14_args)),
> x13=x14,vx13=vx14,
> y13=y14,vy13=vy14,
> z13=z14,vz13=vz14,
> stat13=stat14,
> p5=p7,p6=p8,
> seq(dx13d||i=dx14d||i,i=x14_args),
> seq(dvx13d||i=dvx14d||i,i=vx14_args),
> seq(dy13d||i=dy14d||i,i=y14_args),
> seq(dvy13d||i=dvy14d||i,i=vy14_args),
> seq(dz13d||i=dz14d||i,i=z14_args),
> seq(dvz13d||i=dvz14d||i,i=vz14_args),
> fx13=fx14,fy13=fy14,fz13=fz14,
> tx13=tx14,ty13=ty14,tz13=tz14
>}:
> rtoe14:=subs(from_toe13_to_toe14_sset,rtoe13):
> vtoe14:=subs(from_toe13_to_toe14_sset,vtoe13):
> toe14_cmll1_cross_f:=subs(from_toe13_to_toe14_sset,toe13_cmll1_cross_f
> ):
> toe14_list:=subs(from_toe13_to_toe14_sset,toe13_list):
> dset14:=subs(from_toe13_to_toe14_sset,dset13):
```

D.10.5 Second foot first toe 21

```
> from_leg1_to_leg2:=
>{q7=q11,q8=q12,q9=q13,q10=q14,u7=u11,u8=u12,u9=u13,u10=u14,theta1=the
> ta4,theta2=theta5,theta3=theta6}:
> #from_leg1_to_leg2:=
>{q7=q9,q8=q10,u7=u9,u8=u10,theta1=theta4,theta2=theta5,theta3=theta6
>}:
> x21_args:=seq(subs(from_leg1_to_leg2,[x11_args])[i],i=1..nops([x11_arg
> s])):
> y21_args:=seq(subs(from_leg1_to_leg2,[y11_args])[i],i=1..nops([y11_arg
> s])):
> z21_args:=seq(subs(from_leg1_to_leg2,[z11_args])[i],i=1..nops([z11_arg
> s])):
> vx21_args:=seq(subs(from_leg1_to_leg2,[vx11_args])[i],i=1..nops([vx11_
> args])):
> vy21_args:=seq(subs(from_leg1_to_leg2,[vy11_args])[i],i=1..nops([vy11_
> args])):
> vz21_args:=seq(subs(from_leg1_to_leg2,[vz11_args])[i],i=1..nops([vz11_
> args])):
```

```
> from_toe11_to_toe21_sset:={
> xt11=xt21,yt11=yt21,zt11=zt21,
> rl1l1=rl2l1,rl1l2=rl2l2,rl1l3=rl2l3,
> rl11=rl21,
> l1u=l2u,
> l1l=l2l,
> toefx(stat11,x11(x11_args))=toefx(stat21,x21(x21_args)),
> toefy(stat11,y11(y11_args))=toefy(stat21,y21(y21_args)),
> toefz(stat11,z11(z11_args))=toefz(stat21,z21(z21_args)),
> x11=x21,vx11=vx21,
> y11=y21,vy11=vy21,
> z11=z21,vz11=vz21,
> stat11=stat21,
> p1=p9,p2=p10,
> seq(dx11d||x11_args[i]=dx21d||x21_args[i],i=1..nops([x21_args])),
> seq(dvx11d||vx11_args[i]=dvx21d||vx21_args[i],i=1..nops([vx21_args])),
> dx11dq7=dx21dq11,dx11dq8=dx21dq12,
> dvx11dq7=dvx21dq11,dvx11dq8=dvx21dq12,
> dx11du7=dx21du11,dx11du8=dx21du12,
> dvx11du7=dvx21du11,dvx11du8=dvx21du12,
> seq(dy11d||y11_args[i]=dy21d||y21_args[i],i=1..nops([y21_args])),
> seq(dvy11d||vy11_args[i]=dvy21d||vy21_args[i],i=1..nops([vy21_args])),
> dy11dq7=dy21dq11,dy11dq8=dy21dq12,
> dvy11dq7=dvy21dq11,dvy11dq8=dvy21dq12,
> dy11du7=dy21du11,dy11du8=dy21du12,
> dvy11du7=dvy21du11,dvy11du8=dvy21du12,
> seq(dz11d||z11_args[i]=dz21d||z21_args[i],i=1..nops([z21_args])),
> seq(dvz11d||vz11_args[i]=dvz21d||vz21_args[i],i=1..nops([vz21_args])),
> dz11dq7=dz21dq11,dz11dq8=dz21dq12,
> dvz11dq7=dvz21dq11,dvz11dq8=dvz21dq12,
> dz11du7=dz21du11,dz11du8=dz21du12,
> dvz11du7=dvz21du11,dvz11du8=dvz21du12,
> fx11=fx21,fy11=fy21,fz11=fz21,
> tx11=tx21,ty11=ty21,tz11=tz21
>}:
> rtoe21:=subs(from_leg1_to_leg2,subs(from_toe11_to_toe21_sset,rtoe11))
> :
> vtoe21:=subs(from_leg1_to_leg2,subs(from_toe11_to_toe21_sset,vtoe11)):
> toe21_cmll2_cross_f:=subs(from_leg1_to_leg2,subs(from_toe11_to_toe21_s
> set,toe11_cmll1_cross_f)):
> toe21_list:=subs(from_leg1_to_leg2,subs(from_toe11_to_toe21_sset,toe11
> _list)):
> dset21:=subs(from_leg1_to_leg2,subs(from_toe11_to_toe21_sset,dset11)):
```

D.10.6 Second foot second toe 22

```
> x22_args:=x21_args:
> y22_args:=y21_args:
> z22_args:=z21_args:
> vx22_args:=vx21_args:
> vy22_args:=vy21_args:
> vz22_args:=vz21_args:
> #x22_args:=seq(subs(from_leg1_to_leg2,[x12_args])[i],i=1..nops([x12_ar
> gs]));
> #y22_args:=seq(subs(from_leg1_to_leg2,[y12_args])[i],i=1..nops([y12_ar
> gs]));
> #z22_args:=seq(subs(from_leg1_to_leg2,[z12_args])[i],i=1..nops([z12_ar
> gs]));
> #vx22_args:=seq(subs(from_leg1_to_leg2,[vx12_args])[i],i=1..nops([vx12
> _args]));
> #vy22_args:=seq(subs(from_leg1_to_leg2,[vy12_args])[i],i=1..nops([vy12
> _args]));
> #vz22_args:=seq(subs(from_leg1_to_leg2,[vz12_args])[i],i=1..nops([vz12
> _args]));
> from_toe21_to_toe22_sset:={
> xt21=xt22,yt21=yt22,zt21=zt22,
> rl2l1=rl2l1,r21l2=rl2l2,rl2l3=rl2l3,
> toefx(stat21,x21(x21_args))=toefx(stat22,x22(x22_args)),
> toefy(stat21,y21(y21_args))=toefy(stat22,y22(y22_args)),
> toefz(stat21,z21(z21_args))=toefz(stat22,z22(z22_args)),
> x21=x22,vx21=vx22,
> y21=y22,vy21=vy22,
> z21=z22,vz21=vz22,
> stat21=stat22,
> p9=p11,p10=p12,
> seq(dx21d||x21_args[i]=dx22d||x22_args[i],i=1..nops([x22_args])),
> seq(dvx21d||vx21_args[i]=dvx22d||vx22_args[i],i=1..nops([vx22_args])),
> seq(dy21d||y21_args[i]=dy22d||y22_args[i],i=1..nops([y22_args])),
> seq(dvy21d||vy21_args[i]=dvy22d||vy22_args[i],i=1..nops([vy22_args])),
> seq(dz21d||z21_args[i]=dz22d||z22_args[i],i=1..nops([z22_args])),
> seq(dvz21d||vz21_args[i]=dvz22d||vz22_args[i],i=1..nops([vz22_args])),
> fx21=fx22,fy21=fy22,fz21=fz22,
> tx21=tx22,ty21=ty22,tz21=tz22
>}:
> rtoe22:=subs(from_toe21_to_toe22_sset,rtoe21):
> vtoe22:=subs(from_toe21_to_toe22_sset,vtoe21):
> toe22_cmll2_cross_f:=subs(from_toe21_to_toe22_sset,toe21_cmll2_cross_f
> ):
> toe22_list:=subs(from_toe21_to_toe22_sset,toe21_list):
> dset22:=subs(from_toe21_to_toe22_sset,dset21);
```

D.11 Second foot second toe 23

```
> #x23_args:=x21_args:
> #y23_args:=y21_args:
> #z23_args:=z21_args:
> #vx23_args:=vx21_args:
> #vy23_args:=vy21_args:
> #vz23_args:=vz21_args:
> #from_leg1_to_leg2:=
>{q7=q9,q8=q10,u7=u9,u8=u10,f11x=f21x,f11y=f21y,f11z=f21z,f12x=f22x,f1
> 2y=f22y,f12z=f22z}:
> x23_args:=seq(subs(from_leg1_to_leg2,[x13_args])[i],i=1..nops([x13_arg
> s]));
> y23_args:=seq(subs(from_leg1_to_leg2,[y13_args])[i],i=1..nops([y13_arg
> s])):
> z23_args:=seq(subs(from_leg1_to_leg2,[z13_args])[i],i=1..nops([z13_arg
> s])):
> vx23_args:=seq(subs(from_leg1_to_leg2,[vx13_args])[i],i=1..nops([vx13_
> args])):
> vy23_args:=seq(subs(from_leg1_to_leg2,[vy13_args])[i],i=1..nops([vy13_
> args])):
> vz23_args:=seq(subs(from_leg1_to_leg2,[vz13_args])[i],i=1..nops([vz13_
> args])):
> from_toe13_to_toe23_sset:={
> xt13=xt23,yt13=yt23,zt13=zt23,
> rl1l1=rl2l1,rl1l2=rl2l2,rl1l3=rl2l3,
> rl11=rl21,
> l1u=l2u,
> l1l=l2l,
> toefx(stat13,x13(x13_args))=toefx(stat23,x23(x23_args)),
> toefy(stat13,y13(y13_args))=toefy(stat23,y23(y23_args)),
> toefz(stat13,z13(z13_args))=toefz(stat23,z23(z23_args)),
> x13=x23,vx13=vx23,
> y13=y23,vy13=vy23,
> z13=z23,vz13=vz23,
> stat13=stat23,
> p5=p13,p6=p14,
> seq(dx13d||x13_args[i]=dx23d||x23_args[i],i=1..nops([x23_args])),
> seq(dvx13d||vx13_args[i]=dvx23d||vx23_args[i],i=1..nops([vx23_args])),
> seq(dy13d||y13_args[i]=dy23d||y23_args[i],i=1..nops([y23_args])),
> seq(dvy13d||vy13_args[i]=dvy23d||vy23_args[i],i=1..nops([vy23_args])),
> seq(dz13d||z13_args[i]=dz23d||z23_args[i],i=1..nops([z23_args])),
> seq(dvz13d||vz13_args[i]=dvz23d||vz23_args[i],i=1..nops([vz23_args])),
> dx13dq7=dx23dq11,dx13dq8=dx23dq12,
> dvx13dq7=dvx23dq11,dvx13dq8=dvx23dq12,
> dx13du7=dx23du11,dx13du8=dx23du12,
> dvx13du7=dvx23du11,dvx13du8=dvx23du12,
> dy13dq7=dy23dq11,dy13dq8=dy23dq12,
> dvy13dq7=dvy23dq11,dvy13dq8=dvy23dq12,
> dy13du7=dy23du11,dy13du8=dy23du12,
> dvy13du7=dvy23du11,dvy13du8=dvy23du12,
> dz13dq7=dz23dq11,dz13dq8=dz23dq12,
> dvz13dq7=dvz23dq11,dvz13dq8=dvz23dq12,
> dz13du7=dz23du11,dz13du8=dz23du12,
> dvz13du7=dvz23du11,dvz13du8=dvz23du12,
> fx13=fx23,fy13=fy23,fz13=fz23,
> tx13=tx23,ty13=ty23,tz13=tz23
>}:
```

```
> rtoe23:=subs(from_leg1_to_leg2,subs(from_toe13_to_toe23_sset,rtoe13))
> :
> vtoe23:=subs(from_leg1_to_leg2,subs(from_toe13_to_toe23_sset,vtoe13)):
> toe23_cmll2_cross_f:=subs(from_leg1_to_leg2,subs(from_toe13_to_toe23_s
> set,toe13_cmll1_cross_f)):
> toe23_list:=subs(from_leg1_to_leg2,subs(from_toe13_to_toe23_sset,toe13
> _list)):
> dset23:=subs(from_leg1_to_leg2,subs(from_toe13_to_toe23_sset,dset13)):
```

D.12 Second foot second toe 24

```
> x24_args:=x23_args:
> y24_args:=y23_args:
> z24_args:=z23_args:
> vx24_args:=vx23_args:
> vy24_args:=vy23_args:
> vz24_args:=vz23_args:
> from_toe23_to_toe24_sset:={
> xt23=xt24,yt23=yt24,zt23=zt24,
> rl2l1=rl2l1,rl2l2=rl2l2,rl2l3=rl2l3,
> toefx(stat23,x23(x23_args))=toefx(stat24,x24(x24_args)),
> toefy(stat23,y23(y23_args))=toefy(stat24,y24(y24_args)),
> toefz(stat23,z23(z23_args))=toefz(stat24,z24(z24_args)),
> x23=x24,vx23=vx24,
> y23=y24,vy23=vy24,
> z23=z24,vz23=vz24,
> stat23=stat24,
> p13=p15,p14=p16,
> seq(dx23d||i=dx24d||i,i=x24_args),
> seq(dvx23d||i=dvx24d||i,i=vx24_args),
> seq(dy23d||i=dy24d||i,i=y24_args),
> seq(dvy23d||i=dvy24d||i,i=vy24_args),
> seq(dz23d||i=dz24d||i,i=z24_args),
> seq(dvz23d||i=dvz24d||i,i=vz24_args),
> fx23=fx24,fy23=fy24,fz23=fz24,
> tx23=tx24,ty23=ty24,tz23=tz24
>}:
> rtoe24:=subs(from_toe23_to_toe24_sset,rtoe23):
> vtoe24:=subs(from_toe23_to_toe24_sset,vtoe23):
> toe24_cmll2_cross_f:=subs(from_toe23_to_toe24_sset,toe23_cmll2_cross_f
> ):
> toe24_list:=subs(from_toe23_to_toe24_sset,toe23_list):
> dset24:=subs(from_toe23_to_toe24_sset,dset23):
```

D.13 Knee & Hip for leg 1 & 2

```
> knee1_list:=[Tleg1l=knee(statk1,q10,u10,knee_k,knee_d,knee_d_reb)]:
> knee2_list:=[Tleg2l=knee(statk2,q14,u14,knee_k,knee_d,knee_d_reb)]:
> #knee1_list:=[Tleg1l=knee(statk1,q8,u8,knee_k,knee_d,knee_d_reb)]:
> #knee2_list:=[Tleg2l=knee(statk2,q10,u10,knee_k,knee_d,knee_d_reb)]:
```

```
> dset_knees:=[
> diff(knee(statk1,q10,u10,knee_k,knee_d,knee_d_reb),q10) =
> knee_q(statk1,q10,u10,knee_k,knee_d,knee_d_reb),
> diff(knee(statk1,q10,u10,knee_k,knee_d,knee_d_reb),u10) =
> knee_u(statk1,q10,u10,knee_k,knee_d,knee_d_reb),
> diff(knee(statk2,q14,u14,knee_k,knee_d,knee_d_reb),q14) =
> knee_q(statk2,q14,u14,knee_k,knee_d,knee_d_reb),
> diff(knee(statk2,q14,u14,knee_k,knee_d,knee_d_reb),u14) =
> knee_u(statk2,q14,u14,knee_k,knee_d,knee_d_reb)
> ]:
> ##  dset_knees:=[
> ##  diff(knee(statk1,q8,u8,knee_k,knee_d,knee_d_reb),q8)    =
> knee_q(statk1,q8,u8,knee_k,knee_d,knee_d_reb),
> ##  diff(knee(statk1,q8,u8,knee_k,knee_d,knee_d_reb),u8)    =
> knee_u(statk1,q8,u8,knee_k,knee_d,knee_d_reb),
> ##  diff(knee(statk2,q10,u10,knee_k,knee_d,knee_d_reb),q10) =
> knee_q(statk2,q10,u10,knee_k,knee_d,knee_d_reb),
> ##  diff(knee(statk2,q10,u10,knee_k,knee_d,knee_d_reb),u10) =
> knee_u(statk2,q10,u10,knee_k,knee_d,knee_d_reb)
> ##  ]:


> &simp subs(kde_new,N &to ((N &aV UL1) &-- (N &aV B)));
> &simp subs(kde_new,N &to ((N &aV B) &-- (N &aV UL2)));


> Rmx(B, UL1);
> hip1_list1:=[Tleg11=hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,h
> ip1_d)]:
> hip2_list1:=[Tleg21=hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip
> 1_k,hip1_d)];
> hip1_list2:=[Tleg12=hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hi
> p2_d)]:
> hip2_list2:=[Tleg22=hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip
> 2_k,hip2_d)];
> hip1_list3:=[Tleg13=hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hi
> p3_d)]:
> hip2_list3:=[Tleg23=hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip
> 3_k,hip3_d)];
> #hip1_list1:=[Tleg11=hip2(q4,q7,u4,u7,hip_k,hip_delta,hip_d)]:
> #hip2_list1:=[Tleg21=hip2(q4,q11,u4,u11,hip_k,hip_delta,hip_d)]:
> #hip1_list2:=[Tleg12=hip1(q8,u8,hip3_k,hip3_d)]:
> #hip2_list2:=[Tleg22=hip1(q12,u12,hip3_k,hip3_d)]:
> #hip1_list3:=[Tleg13=hip1(q9,u9,hip2_k,hip2_d)]:
> #hip2_list3:=[Tleg23=hip1(q13,u13,hip2_k,hip2_d)]:
> ## hip1_list:=[Tleg1=hip1(q4,q7,u4,u7,hip_k,hip_delta,hip_d)]:
> ## hip2_list:=[Tleg2=hip1(q4,q9,u4,u9,hip_k,hip_delta,hip_d)]:
```

```
> dset_hips:=[
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),q4) =
> hip1_q_1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),q5) =
> hip1_q_2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),q6) =
> hip1_q_3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),q7) =
> hip1_q_4(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),q8) =
> hip1_q_5(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),q9) =
> hip1_q_6(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),u4) =
> hip1_u_1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),u5) =
> hip1_u_2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),u6) =
> hip1_u_3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),u7) =
> hip1_u_4(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),u8) =
> hip1_u_5(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),u9) =
> hip1_u_6(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),q4)
> = hip1_q_1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),q5)
> = hip1_q_2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),q6)
> = hip1_q_3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),q11
> ) = hip1_q_4(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),q12
> ) = hip1_q_5(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),q13
> ) = hip1_q_6(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),u4)
> = hip1_u_1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),u5)
> = hip1_u_2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),u6)
> = hip1_u_3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),u11
> ) = hip1_u_4(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),u12
> ) = hip1_u_5(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),u13
> ) = hip1_u_6(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip1_k,hip1_d),
> diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),q4) =
> hip2_q_1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
> diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),q5) =
> hip2_q_2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
> diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),q6) =
> hip2_q_3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
> diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),q7) =
> hip2_q_4(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
> diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),q8) =
> hip2_q_5(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
> diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),q9) =
> hip2_q_6(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
```

```
>  diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),u4) =
>  hip2_u_1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),u5) =
>  hip2_u_2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),u6) =
>  hip2_u_3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),u7) =
>  hip2_u_4(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),u8) =
>  hip2_u_5(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),u9) =
>  hip2_u_6(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),q4)
>  = hip2_q_1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),q5)
>  = hip2_q_2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),q6)
>  = hip2_q_3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),q11
>  ) = hip2_q_4(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),q12
>  ) = hip2_q_5(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),q13
>  ) = hip2_q_6(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),u4)
>  = hip2_u_1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),u5)
>  = hip2_u_2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),u6)
>  = hip2_u_3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),u11
>  ) = hip2_u_4(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),u12
>  ) = hip2_u_5(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),u13
>  ) = hip2_u_6(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip2_k,hip2_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),q4) =
>  hip3_q_1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),q5) =
>  hip3_q_2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),q6) =
>  hip3_q_3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),q7) =
>  hip3_q_4(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),q8) =
>  hip3_q_5(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),q9) =
>  hip3_q_6(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),u4) =
>  hip3_u_1(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),u5) =
>  hip3_u_2(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),u6) =
>  hip3_u_3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),u7) =
>  hip3_u_4(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),u8) =
>  hip3_u_5(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
>  diff(hip3(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),u9) =
>  hip3_u_6(q4,q5,q6,q7,q8,q9,u4,u5,u6,u7,u8,u9,hip3_k,hip3_d),
```

```
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),q4)
> = hip3_q_1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),q5)
> = hip3_q_2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),q6)
> = hip3_q_3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),q11
> ) = hip3_q_4(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),q12
> ) = hip3_q_5(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),q13
> ) = hip3_q_6(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),u4)
> = hip3_u_1(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),u5)
> = hip3_u_2(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),u6)
> = hip3_u_3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),u11
> ) = hip3_u_4(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),u12
> ) = hip3_u_5(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),
> diff(hip3(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d),u13
> ) =
> hip3_u_6(q4,q5,q6,q11,q12,q13,u4,u5,u6,u11,u12,u13,hip3_k,hip3_d)]:
> Rmx(B,T3);
> &simp (UL1 &to ((B &ev [0,k1*sin(q9),0]) &++(T3 &ev
> [k2*sin(q7),0,0]))):
```

D.14 Forces and torques

```
> Fbody:=N &ev [0, -m_body*g*cos(theta), -m_body*g*sin(theta)]:
> Tbody:= (T2 &ev [Tleg11+Tleg21,0,0]) &++ (T4 &ev [0,Tleg12,0]) &++
> (T3 &ev [0,0,Tleg13]) &++ (T6 &ev [0,Tleg22,0]) &++ (T5 &ev
> [0,0,Tleg23]);


> Fleg1upper:=N &ev [0,-ml1u*g*cos(theta),-ml1u*g*sin(theta)]:
> Tleg1upper:=(T2 &ev [-Tleg11,0,0]) &++(T3 &ev [0,0,-Tleg13]) &++ (T4
> &ev [0,-Tleg12,0]) &++ (UL1 &ev [Tleg1l,0,0]);


> Fleg2upper:=N &ev [0,-ml2u*g*cos(theta),-ml2u*g*sin(theta)]:
> Tleg2upper:= (T2 &ev [-Tleg21,0,0]) &++ (T5 &ev [0,0,-Tleg23]) &++
> (T6 &ev [0,-Tleg22,0]) &++ (UL2 &ev [Tleg2l,0,0]);


> Fleg1lower:=N &ev
> [fx11+fx12+fx13+fx14,fy11+fy12+fy13+fy14-ml1l*g*cos(theta),fz11+fz12+f
> z13+fz14-ml1l*g*sin(theta)]:
> Tleg1lower:=(UL1 &ev [-Tleg1l,0,0]) &++ (N &ev
> [tx11+tx12+tx13+tx14,ty11+ty12+ty13+ty14,tz11+tz12+tz13+tz14]):


> Fleg2lower:=N &ev
> [fx21+fx22+fx23+fx24,fy21+fy22+fy23+fy24-ml2l*g*cos(theta),fz21+fz22+f
> z23+fz24-ml2l*g*sin(theta)]:
> Tleg2lower:=(UL2 &ev [-Tleg2l,0,0]) &++ (N &ev
> [tx21+tx22+tx23+tx24,ty21+ty22+ty23+ty24,tz21+tz22+tz23+tz24]):
```

D.15 Kanes method

```
> pKt:=[ptbody, ptleg1upper, ptleg2upper, ptleg1lower, ptleg2lower,
> htbody, htleg1upper, htleg2upper, htleg1lower, htleg2lower]:

> RKt:=[mkc(Fbody),mkc(Fleg1upper),mkc(Fleg2upper),mkc(Fleg1lower),mkc(
> Fleg2lower),
> mkc(Tbody),mkc(Tleg1upper),mkc(Tleg2upper),mkc(Tleg1lower),mkc(Tleg2lo
> wer)]:
> kane_eq:=map(simplify,ckane(tauK,pKt,RKt)):
```

D.16 Export of dyn eqs to Matlab

D.16.1 Common

Common subexpressions

```
> cse:=[op(toe11_list),op(toe12_list),op(toe13_list),op(toe14_list),
> op(toe21_list),op(toe22_list),op(toe23_list),op(toe24_list),
> op(knee1_list),op(knee2_list),
> op(hip1_list1),op(hip2_list1),op(hip1_list2),op(hip2_list2),op(hip1_li
> st3),op(hip2_list3)]:
```

Translating differentiated functions into the appropriate c-function calls

```
> dset:=[op(dset11),op(dset12),op(dset13),op(dset14),op(dset21),op(dset
> 22),op(dset23),op(dset24),op(dset_knees),op(dset_hips)]:
> uts :=
> [u1t,u2t,u3t,u4t,u5t,u6t,u7t,u8t,u9t,u10t,u11t,u12t,u13t,u14t]:
> kde_new_ordered:=[seq(q||i||t=subs(kde_new,q||i||t),i=1..14),seq(p||i
> ||t=subs(kde_new,p||i||t),i=1..16),seq(theta||i||t=subs(kde_new,theta|
> |i||t),i=1..6)]:

> qts :=
> [q1t,q2t,q3t,q4t,q5t,q6t,q7t,q8t,q9t,q10t,q11t,q12t,q13t,q14t,p1t,p2t,
> p3t,p4t,p5t,p6t,p7t,p8t,p9t,p10t,p11t,p12t,p13t,p14t,p15t,p16t,theta1t
> ,theta2t,theta3t,theta4t,theta5t,theta6t]:
> vars:=[q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14,p1,p2,p3,p4,p5,
> p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,theta1,theta2,theta3,theta4,th
> eta5,theta6,u1,u2,u3,u4,u5,u6,u7,u8,u9,u10,u11,u12,u13,u14]:
```

```
> para:=[
> rx,ry,rz,
> rl11,rl21,
> rl1u1,rl1u2,rl1u3,
> rl2u1,rl2u2,rl2u3,
> l1u,l2u,
> rl1l1,rl1l2,rl1l3,
> rl2l1,rl2l2,rl2l3,
> l1l,l2l,
> I_B11,I_B22,I_B33,I_B12,I_B13,I_B23,
> I_LU111,I_LU122,I_LU133,I_LU112,I_LU113,I_LU123,
> I_LU211,I_LU222,I_LU233,I_LU212,I_LU213,I_LU223,
> I_LL111,I_LL122,I_LL133,I_LL112,I_LL113,I_LL123,
> I_LL211,I_LL222,I_LL233,I_LL212,I_LL213,I_LL223,
> m_body,ml1u,ml2u,ml1l,ml2l,g,theta,
> kX,dX,kY,dY,kZ,dZ,
> knee_k,knee_d,knee_d_reb,
> hip1_k,hip_delta,hip1_d,
> hip2_k,hip2_d,
> hip3_k,hip3_d,
> statk1,statk2,
> xt11,yt11,zt11,stat11,
> xt12,yt12,zt12,stat12,
> xt13,yt13,zt13,stat13,
> xt14,yt14,zt14,stat14,
> xt21,yt21,zt21,stat21,
> xt22,yt22,zt22,stat22,
> xt23,yt23,zt23,stat23,
> xt24,yt24,zt24,stat24,
> poin1,poin2]:

> ##for i from 1 to 103 do lprint(i,cat(‘’‘,para[i],‘’,...‘)); od;

> b:=[
> [
> [y11=rtoe11 &c 2,y12=rtoe12 &c 2,y13=rtoe13 &c 2, y14=rtoe14 &c
> 2,
> y21=rtoe21 &c 2,y22=rtoe22 &c 2,y23=rtoe23 &c 2, y24=rtoe24 &c
> 2],
> [y11,y12,y13,y14,y21,y22,y23,y24,q10,q14,q7,q11]
> ],
> [1,1,1,1,1,1,1,1,1,1,1,1],
> [stat11,stat12,stat13,stat14,stat21,stat22,stat23,stat24,statk1,statk2
> ,poin1,poin2]
> ]:

> inc:=‘includes‘=[‘"walker_3D_ext_abship.c"‘]:


> path:=‘/scratch/petri/C_Walker/Ccode/‘:
> #path:=‘d:\\Petri\\Maple_sophia\\‘:
> #path:=‘F:\\Newtontest\\New_foot_mod\\3D_walker_ankle\\Csource
> \\‘:
```

D.16.2 Simulation

```
> exmex(‘walker_3D_newabship‘,path,[[op(kde_new_ordered)]],[kane_eq,uts
> ],[op(subs(dset,cse))],[op(qts),op(uts)],vars,parameters=para,’v5’=[’e
> vents’=b],inc);
```

D.16.3 Toeposfunc

```
> toe11 := matrix(1,3,[rtoe11 &c 1, rtoe11 &c 2, rtoe11 &c 3]):
> toe12 := matrix(1,3,[rtoe12 &c 1, rtoe12 &c 2, rtoe12 &c 3]):
> toe13 := matrix(1,3,[rtoe13 &c 1, rtoe13 &c 2, rtoe13 &c 3]):
> toe14 := matrix(1,3,[rtoe14 &c 1, rtoe14 &c 2, rtoe14 &c 3]):
> toe21 := matrix(1,3,[rtoe21 &c 1, rtoe21 &c 2, rtoe21 &c 3]):
> toe22 := matrix(1,3,[rtoe22 &c 1, rtoe22 &c 2, rtoe22 &c 3]):
> toe23 := matrix(1,3,[rtoe23 &c 1, rtoe23 &c 2, rtoe23 &c 3]):
> toe24 := matrix(1,3,[rtoe24 &c 1, rtoe24 &c 2, rtoe24 &c 3]):
> exmat('da_toe11_8_toe',path,toe11,vars,para,'matlab');
> exmat('da_toe12_8_toe',path,toe12,vars,para,'matlab');
> exmat('da_toe13_8_toe',path,toe13,vars,para,'matlab');
> exmat('da_toe14_8_toe',path,toe14,vars,para,'matlab');
> exmat('da_toe21_8_toe',path,toe21,vars,para,'matlab');
> exmat('da_toe22_8_toe',path,toe22,vars,para,'matlab');
> exmat('da_toe23_8_toe',path,toe23,vars,para,'matlab');
> exmat('da_toe24_8_toe',path,toe24,vars,para,'matlab');


> exmat('da_BtoN_8_toe',path,Rmx(N,B),vars,'matlab');
> exmat('da_T2toN_8_toe',path,Rmx(N,T2),vars,'matlab');
> exmat('da_UL1toT2_8_toe',path,Rmx(T2,UL1),vars,'matlab');
> exmat('da_UL2toT2_8_toe',path,Rmx(T2,UL2),vars,'matlab');
> exmat('da_LL1toUL1_8_toe',path,Rmx(UL1,LL1),vars,'matlab');
> exmat('da_LL2toUL2_8_toe',path,Rmx(UL2,LL2),vars,'matlab');
> exmat('da_F1toLL1_8_toe',path,Rmx(LL1,F1),vars,'matlab');
> exmat('da_F2toLL2_8_toe',path,Rmx(LL2,F2),vars,'matlab');
> Rmx(N,F2);
```

D.17 Export of dyn eqs to matlab

```
> exmex(‘walker_3D_abship7_ankle2_stab‘,path,[[op(kde_new_ordered)]],[k
> ane_eq,uts],[op(cse)],[op(qts),op(uts)],vars,variationaleqs,parameters
> =para,diffset=dset,'v5'=['events'=b],inc);
```

D.18 Other stuff

```
> ## Constructing the mapping matrices for discontinous forces
> qus:=[seq(q||i,i=1..14),seq(p||i,i=1..16),seq(theta||i,i=1..6),seq(u|
> |i,i=1..14)];
> mxtoe11 := matrix(50,1,[]);
> mxtoe12 := matrix(50,1,[]);
> mxtoe13 := matrix(50,1,[]);
> mxtoe14 := matrix(50,1,[]);
> mxtoe21 := matrix(50,1,[]);
> mxtoe22 := matrix(50,1,[]);
> mxtoe23 := matrix(50,1,[]);
> mxtoe24 := matrix(50,1,[]);
> i:=1;
> for ii in qus do
> mxtoe11[i,1]:=diff(rtoe11 &c 2, ii);
> mxtoe12[i,1]:=diff(rtoe12 &c 2, ii);
> mxtoe13[i,1]:=diff(rtoe13 &c 2, ii);
> mxtoe14[i,1]:=diff(rtoe14 &c 2, ii);
> mxtoe21[i,1]:=diff(rtoe21 &c 2, ii);
> mxtoe22[i,1]:=diff(rtoe22 &c 2, ii);
> mxtoe23[i,1]:=diff(rtoe23 &c 2, ii);
> mxtoe24[i,1]:=diff(rtoe24 &c 2, ii);
> i:=i+1;
> od:
```

```
> exmat('Ntoe11_8_toe_p',path,mxtoe11,qus,para,'matlab');
> exmat('Ntoe12_8_toe_p',path,mxtoe12,qus,para,'matlab');
> exmat('Ntoe13_8_toe_p',path,mxtoe13,qus,para,'matlab');
> exmat('Ntoe14_8_toe_p',path,mxtoe14,qus,para,'matlab');
> exmat('Ntoe21_8_toe_p',path,mxtoe21,qus,para,'matlab');
> exmat('Ntoe22_8_toe_p',path,mxtoe22,qus,para,'matlab');
> exmat('Ntoe23_8_toe_p',path,mxtoe23,qus,para,'matlab');
> exmat('Ntoe24_8_toe_p',path,mxtoe24,qus,para,'matlab');
```

qus;

```
> para;


> G11:=[seq(q||i,i=1..14),(rtoe11 &c 1)-q1,(rtoe11 &c
> 3)-q3,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,seq(theta||i,i=
> 1..6), seq(u||i,i=1..14)]:
> G12:=[seq(q||i,i=1..14),p1,p2,(rtoe12 &c 1)-q1,(rtoe12 &c
> 3)-q3,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,seq(theta||i,i=1..6),
> seq(u||i,i=1..14)]:
> G13:=[seq(q||i,i=1..14),p1,p2,p3,p4,(rtoe13 &c 1)-q1,(rtoe13 &c
> 3)-q3,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,seq(theta||i,i=1..6),seq(u|
> |i,i=1..14)]:
> G14:=[seq(q||i,i=1..14),p1,p2,p3,p4,p5,p6,(rtoe14 &c 1)-q1,(rtoe14 &c
> 3)-q3,p9,p10,p11,p12,p13,p14,p15,p16,seq(theta||i,i=1..6),seq(u||i,i=1
> ..14)]:
> G21:=[seq(q||i,i=1..14),p1,p2,p3,p4,p5,p6,p7,p8,(rtoe21 &c
> 1)-q1,(rtoe21 &c
> 3)-q3,p11,p12,p13,p14,p15,p16,seq(theta||i,i=1..6),seq(u||i,i=1..14)]:
> G22:=[seq(q||i,i=1..14),p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,(rtoe22 &c
> 1)-q1,(rtoe22 &c
> 3)-q3,p13,p14,p15,p16,seq(theta||i,i=1..6),seq(u||i,i=1..14)]:
> G23:=[seq(q||i,i=1..14),p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,(rtoe23
> &c 1)-q1,(rtoe23 &c
> 3)-q3,p15,p16,seq(theta||i,i=1..6),seq(u||i,i=1..14)]:
> G24:=[seq(q||i,i=1..14),p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14
> ,(rtoe24 &c 1)-q1,(rtoe24 &c
> 3)-q3,seq(theta||i,i=1..6),seq(u||i,i=1..14)]:
> qus:=[seq(q||i,i=1..14),seq(p||i,i=1..16),seq(theta||i,i=1..6),seq(u||
> i,i=1..14)];
> Gx11 := matrix(50,50,[]);
> Gx12 := matrix(50,50,[]);
> Gx13 := matrix(50,50,[]);
> Gx14 := matrix(50,50,[]);
> Gx21 := matrix(50,50,[]);
> Gx22 := matrix(50,50,[]);
> Gx23 := matrix(50,50,[]);
> Gx24 := matrix(50,50,[]);
> for i from 1 to 50 do
> for j from 1 to 50 do
> Gx11[i,j]:=diff(G11[i], qus[j]):
> Gx12[i,j]:=diff(G12[i], qus[j]):
> Gx13[i,j]:=diff(G13[i], qus[j]):
> Gx14[i,j]:=diff(G14[i], qus[j]):
> Gx21[i,j]:=diff(G21[i], qus[j]):
> Gx22[i,j]:=diff(G22[i], qus[j]):
> Gx23[i,j]:=diff(G23[i], qus[j]):
> Gx24[i,j]:=diff(G24[i], qus[j]):
> od:
> od;
> evalm(Gx11):
```

```
>  exmat('Gx11_8_toe_p',path,Gx11,qus,para,'matlab');
>  exmat('Gx12_8_toe_p',path,Gx12,qus,para,'matlab');
>  exmat('Gx13_8_toe_p',path,Gx13,qus,para,'matlab');
>  exmat('Gx14_8_toe_p',path,Gx14,qus,para,'matlab');
>  exmat('Gx21_8_toe_p',path,Gx21,qus,para,'matlab');
>  exmat('Gx22_8_toe_p',path,Gx22,qus,para,'matlab');
>  exmat('Gx23_8_toe_p',path,Gx23,qus,para,'matlab');
>  exmat('Gx24_8_toe_p',path,Gx24,qus,para,'matlab');
```

Vita

Colleen E. Shannon

Colleen Elizabeth Shannon was born the first daughter and youngest child of Celestine W. and Darlene E. Shannon on October 15, 1978 in Hagerstown, Maryland. She attended Smithsburg High School, where she graduated in June 1996 in the top 10% of her class. She recieved an academic scholarship to attend Western Maryland College (now known as McDaniel College) and graduated from this institution in May 2000 with a B.A. degree in Physics. The following three years she attended graduate school at Virginia Polytechnic Institue and State University. This thesis completes her M.S. degree in Mechanical Engineering. Following graduation, Colleen will be moving back to Maryland to pursue a career in Mechanical Engineering.