Stability of the Simplest Walker Model

Jarno van Wijk

DCT 2009.021

Traineeship report

Coach(es): Dr. Guangyu Liu

Supervisor: Prof. Iven Mareels

Technische Universiteit Eindhoven
Department Mechanical Engineering
Dynamics and Control Technology Group

Eindhoven, August, 2009

Abstract

For a better understanding of the human walking behavior, its motion is studied by investigating a simplified version of a two-legged walking model. This walker is able to perform a stable walking motion only powered by gravity, this is called passive-dynamic walking. When this model is reduced to the simplest walking model, the stepsize of a stable gait cycle is only dependent on the slope angle it is walking on. Garcia et al. [1] found both analytically as numerically a relationship between them, and checked whether these gait cycles are stable. These results are verified in this thesis, using an own approach. For a better view on the stability, Monte Carlo-like simulations have been applied to the model to investigate which initial condition eventually would lead to a stable gait cycle. This responsive behavior of the dynamics is highly dependent on the slope angle.

Acknowledgements

I'd like to acknowledge some people who helped my in realizing this thesis. First of all my supervisor Dr. Guangyu Liu, with whom I had some interesting discussions about the content and gave good advise about the layout of this thesis. I'd also want to acknowledge Prof. Iven Mareels and for his supervision, Prof Maarten Steinbuch for giving me the opportunity to go abroad for this internship, NICTA for giving the possibility to follow my internship at their research laboratory, and Bram with whom I could discuss some problems I ran into and who provided me with the necessary cup of coffee. Last of all I'd like to thank Mariano Garcia for his coding of the simplest walker model.

Contents

1 Introduction **4**
- 1.1 Overview . . . 4
- 1.2 Objective . . . 4
- 1.3 Background . . . 4
- 1.4 Methodologies . . . 5
- 1.5 Contribution . . . 5
- 1.6 Layout of the Thesis . . . 5

2 Literature Review **6**
- 2.1 Problems . . . 6
- 2.2 Review of previous studies . . . 6
  - 2.2.1 Passive-Dynamic Walking . . . 6
  - 2.2.2 Simplest Walker . . . 7
  - 2.2.3 Responsive Behavior of Dynamics . . . 7
  - 2.2.4 Powered Walking . . . 7
- 2.3 The Simplest Model . . . 7
  - 2.3.1 Description and Assumptions . . . 7
  - 2.3.2 Equations of motion during swing phase . . . 8
  - 2.3.3 Transition Rule . . . 9

3 Methods **11**
- 3.1 Runge-Kutta-Fehlberg method . . . 11
- 3.2 Poincaré Map for Cycle Orbits . . . 12
- 3.3 Newton-Raphson Method . . . 13
- 3.4 Stability of the fixed points . . . 14
- 3.5 Responsive Behavior of Dynamics . . . 14

4 Results **16**
- 4.1 Period-One Gait Cycles . . . 16
- 4.2 Fixed points . . . 16
- 4.3 Stability . . . 17
- 4.4 Responsive Behavior of Dynamics . . . 18

5 Conclusion and Future Work **21**
5.1 Conclusion 21
5.2 Future Work 21

A M-files for simulation **22**
A.1 M-files 22
A.1.1 find_all.m 22
A.1.2 jacob.m 23
A.1.3 search.m 24
A.1.4 stride.m 25
A.1.5 xderivs.m 28
A.1.6 run_grid.m 29
A.1.7 step_sequence.m 31
A.2 M-files of Garcia 36
A.2.1 int_doubpend.m 36
A.2.2 int_doubpend_step.m 41
A.2.3 it_doubpend.m 44
A.2.4 run_doubpend.m 46
A.2.5 yderivs_doubpend.m 48

Chapter 1

Introduction

1.1 Overview

This section provides some brief information about this thesis. First the objectives are stated, then some more background information about the problem is given. After that, the methods which are used in this thesis are briefly described, and the contribution of this research is given. Finally there will be some more information about the layout of the rest of this thesis.

1.2 Objective

The objective of this study is to verify the fixed points of the gait cycle and their stability found by Garcia et al. [1], based on own coding. Next the response of the dynamic system is analysed to get a better understanding of the stability properties.

1.3 Background

It has been shown by McGeer [2] that human-like walking motion can be exhibited by an uncontrolled two-legged walking model powered only by gravity. The coordination of the human stride depends on the dynamical properties of the legs. Garcia et al. [1] found for the most simplified version of McGeer's walker the fixed stepsizes as a function of the slope on which the walker is moving, but this movement is only stable for a small range of slope angles. In extension to this research, Schwab and Wisse [3] investigated which initial conditions will eventually lead to a stable walking motion; this "basin of attraction"[1] is shown to be only a small thin area. With control on the legs, this area can be enlarged so that the walker is more robust against disturbances. Now if one's goal is

[1] the term 'basin of attraction' is used different from the common definition as described in e.g. Khalil [4]

to study active, human-like robotic walking it is natural to take advantage of this passive-dynamic behavior to reduce the energetic and computational cost of coordinating the walker's motion.

1.4 Methodologies

- The Runge-Kutta-Fehlberg method is used in order to integrate the equations of motion of the simplest walking model. Because its use of variable stepsize and the possibility to reject a solution if it doesn't match to a given tolerance, this is an appropriate method to use in this thesis.
- Since the full gait cycle is repetitive, Poincaré mapping is used as a mathematical analogy for a this gait cycle to analyze its behavior.
- The Newton-Raphson method is used to find the states when the initial conditions are exactly the same as the conditions at the end of a step. This is then considered to be the fixed point of the Poincaré map.
- The stability of the fixed point is analyzed by evaluating the eigenvalues of the Jacobian at that point.

1.5 Contribution

The contribution of this thesis firstly is to verify the results found by Garcia et al. [1], based on own coding and computations. Further Monte Carlo simulations are used to investigate the behavior of the walker model.

1.6 Layout of the Thesis

In rest of the thesis, Chapter 2 describes the problem of 2D passive walking. A literature survey is provided and the model that will be used in the rest of the thesis will be described. In Chapter 3 the methodologies described above to investigate this model will be further explained. Then in Chapter 4 the results obtained with these methods are presented, after which in Chapter 5 a conclusion can be drawn and some future work can be given.

Chapter 2

Literature Review

2.1 Problems

The aim of studying the human walking behavior is to gain more insight in the existence of independent walking possibility of two-legged walkers. To find this behavior, a proper model has to be made that mimics the human walking motion. This model however should not be too complicated, because the more parameters can be changed, the more difficult it is to see which factors really influence the walking motion. Although real human walking is performed in a 3D environment, a 2D model is able to capture much of the same behavior and is much simpler to investigate.

2.2 Review of previous studies

2.2.1 Passive-Dynamic Walking

One of the first to study passive-dynamic walking was McGeer [2]. He showed that a class of two-legged walkers exists who, once started on a shallow slope, will settle into a steady gait quite comparable to human walking, without active control or energy input. To reach this conclusion, first two wheel-like devices were studied: the synthetic wheel and a rimless spoked wheel. From these two models he derived a two-legged walker model which included parameters such as the foot radius, leg radius of gyration, leg center of mass height, center of mass offset and hip mass fraction. Simulations with variations of these parameters showed that the walking cycle and its stability are highly dependent on them. These results were verified by McGeer by experiments with a test machine. In extension to this study, knees were added to the model to avoid the swing leg's foot touching the ground during its swing phase (scuffing). Simulations with models which included this adjustment proved that this is a far more efficient way to avoid scuffing than e.g. adding tiny motors which lift the feet.

2.2.2 Simplest Walker

With the research of McGeer as starting point, more research has been done in the field of passive-dynamic walking. Garcia et al. [1] studied the most simplified version of McGeer's walking model, called the simplest walker. This walker has only masses at the hip and feet, rigid legs, and an infinitely small foot radius. For this model the initial conditions and stability estimates for period-one gait limit cycles are found both analytical as numerical. For a slope angle γ between 0 and 0.015 rad two period-one gait cycles exist, of which one is stable. For increasing γ stable cycles of higher periods appear, the walking-like motions apparently become chaotic through a sequence of period doubling.

2.2.3 Responsive Behavior of Dynamics

The "basin of attraction" of the cyclic walking motion for the simplest walking model was investigated by Schwab and Wisse [3]. This investigation of the responsive behavior of the dynamics provides an indication which range of initial conditions will eventually lead to a stable walking cycle. Application of a cell mapping method shows the basin of attraction to be a small thin area. To extend this area, Wisse et al. [5] found out that by using swing leg control the walker can be prevented from falling forwards, which is shown by more initial conditions leading to a stable walking cycle.

2.2.4 Powered Walking

To achieve walking on level ground, the gravital forces have to be replaced by some kind of actuation. Kuo [6] modified the simplest model to study the energetics of walking and the preferred relationship between speed and step length in humans. Two methods of actuation were applied to explore powered walking: applying an impulse at toe-off immediately before heelstrike, and a torque applied to the stance leg. Numerical simulations of both methods show that applying an impulse is four times more efficient than applying a torque, because it decreases the collision loss at heelstrike. Walking energetics can be furthermore improved by adding a hip torque on the swing leg to tune its frequency, because this reduces the collision loss even more.

2.3 The Simplest Model

2.3.1 Description and Assumptions

The model that is used in this thesis is called the simplest model. As the name implies, this is the most simplified model which however still encapsules the passive-dynamic walking behavior. The model, also used by Garcia et al. [1], Schwab and Wisse [3],[5] and Kuo [6] may be viewed as a simplified version of the straight legged walker from McGeer [2] by replacing the round feet by

point masses, and giving it a more general weight distribution. A cartoon of our model is shown in figure 2.1. For this model a set of assumptions have been made.

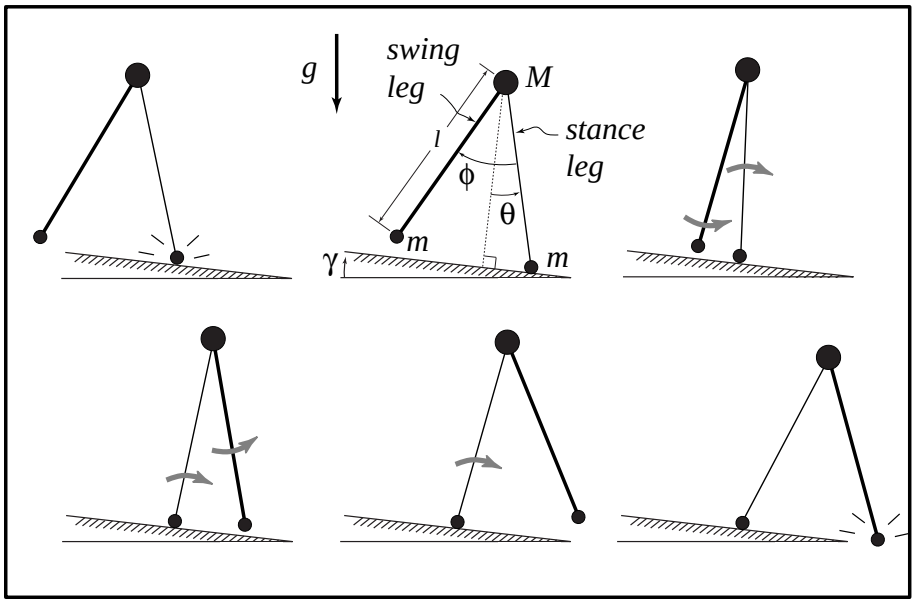

Figure 2.1: A typical passive walking step

- Legs are rigid and massless
- Hip mass M is much larger than foot mass m ($M \gg m$)
- Hip acts as a frictionless hinge
- Collisions are plastic (no-slip, no-bounce)
- Scuffing is ignored

2.3.2 Equations of motion during swing phase

The model acts as a simple double pendulum, with the stance leg as an inverted pendulum pivoting around its stance foot, and the swing leg as a normal pendulum pivoting around the hip. Its motion is now governed by the laws of classical rigid-body mechanics. With Lagrange's equation a set of two coupled second-order differential equations can be derived (2.1), where $\beta = m/M$ and θ, ϕ are dependent of time t.

$$\begin{bmatrix} 1+\beta(1-\cos\phi) & -\beta(1-\cos\phi) \\ \beta(1-\cos\phi) & -\beta \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} + \begin{bmatrix} -\beta\sin\phi(\dot{\phi}^2 - 2\dot{\theta}\dot{\phi}) \\ \beta\dot{\theta}^2\sin\phi \end{bmatrix}$$
$$+ \begin{bmatrix} (\beta g/l)[\sin(\theta-\phi-\gamma) - \sin(\theta-\gamma)] - g/l\sin(\theta-\gamma) \\ (\beta g/l)\sin(\theta-\phi-\gamma) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (2.1)$$

Now we add the assumption that the hip mass is much larger than the foot mass. The result of this is that the motion of the swing leg doesn't influence the motion of the stance leg. This is a realistic simplification, because the mass of the torso of a human in considered to be larger than the mass of a single foot. By setting β to 0 in the first equation of 2.1, and dividing the second one by β, two simpler equations result as in 2.2 and 2.3. Equation 2.2 is also used to further simplify equation 2.3.

$$\ddot{\theta}(t) - \sin(\theta(t)-\gamma) = 0 \quad (2.2)$$
$$\ddot{\theta}(t) - \ddot{\phi}(t) + \dot{\theta}(t)^2\sin\phi(t) - \cos(\theta(t)-\gamma)\sin\phi(t) = 0 \quad (2.3)$$

2.3.3 Transition Rule

To complete the description of the dynamics the impact dynamics have to be clarified: a relationship between angles and momenta just before and just after the heel strike. This transition takes place when the swing and stance foot both touch the ground, which is when the collision condition 2.4 holds. This condition clearly also holds when both legs are nearly parallel ($\theta, \phi \approx 0$), but as already has been mentioned scuffing is ignored so the swing leg can continue its way without any collision.

$$\phi(t) - 2\theta = 0 \quad (2.4)$$

Now when heelstrike occurs, there is an impulse at the swing foot contact point. Assumed is that the former stance leg has no impulsive reaction with the ground, and that the impulsive force is much larger than the non-impulsive forces (like gravity) for the duration of the impact. With these two assumptions angular momentum is conserved through the collision for both the whole mechanism about the swing foot contact point and the former stance leg isolated from the former swing leg about the hip. Now the following jump condition 2.5 can be derived as done by Gomes [7] (in addition to Garcia et al. [1]), where the '+' superscript means 'just after heelstrike', and '-' means just before it.

$$\begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}^{+} = \underbrace{\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & \cos 2\theta & 0 & 0 \\ -2 & 0 & 0 & 0 \\ 0 & \cos 2\theta(1-\cos 2\theta) & 0 & 0 \end{bmatrix}}_{h} \begin{bmatrix} \theta \\ \dot{\theta} \\ \phi \\ \dot{\phi} \end{bmatrix}^{-} \tag{2.5}$$

Chapter 3

Methods

3.1 Runge-Kutta-Fehlberg method

Before any gait cycle can be analyzed, first the ordinary differential equations (ODEs) stated in section 2.3.2 must be solved. To solve the equations of the two-legged walker an integration scheme must be implemented. The standard Matlab integration solvers can't be used on this problem because the equations are not continuously differentiable, which is caused by the transition rule at heelstrike. Now a Runge-Kutta method (denoted RK) could be applied to calculate the motion during the swing phase, but this method uses a fixed stepsize. This can either lead to large calculation times (in case of a too small stepsize) or to a reduced accuracy (in case the stepsize is too large). Therefore an integration method with adjustable stepsize is preferred, in this case is chosen for the Runge-Kutta-Fehlberg method (denoted RKF45). This method compares the approximation of the solution made using a Runge-Kutta method of order 4 (RK4) and a more accurate one made using a Ruge-Kutta method of order 5 (RK5). If the two answers are in close agreement, the approximation is accepted. If the two answers do not agree to a specified accuracy, the step size is reduced. If the answers agree to more significant digits than required, the step size is increased.

To calculate the 4th and 5th order solutions, the equations are evaluated at certain points in space which will lead to a set of function values k_1 to k_6. Since the the equation are not strictly time dependent, all coefficients in 3.1 are, in contrast with the original Runge-Kutta-Fehlberg method, only evaluated at current time t_k.

$$
\begin{aligned}
k_1 &= hf\left(t_k, y_k\right), \\
k_2 &= hf\left(t_k, y_k + \frac{1}{4}k_1\right), \\
k_3 &= hf\left(t_k, y_k + \frac{3}{32}k_1 + \frac{9}{32}k_2\right), \\
k_4 &= hf\left(t_k, y_k + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right), \\
k_5 &= hf\left(t_k, y_k + \frac{439}{216}k_1 - 8k2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right), \\
k_6 &= hf\left(t_k, y_k - \frac{8}{27}k_1 + 2k2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right).
\end{aligned} \tag{3.1}
$$

With these coefficients, first an approximation to the solution of the ODE is made using a RK4 method:

$$
y_{k+1} = y_k + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4101}k_4 - \frac{1}{5}k_5, \tag{3.2}
$$

where the four function values k_1, k_3, k_4, and k_5 are used. Notice that k_2 is not used. A better value for the solution is determined using a RK5 method:

$$
z_{k+1} = y_k + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6. \tag{3.3}
$$

Now the difference between the two solutions can be analyzed by comparing it to a given tolerance. When the difference is smaller than this tolerance, the 5th order approximation is accepted to update the solution, and the new stepsize can be calculated. If not, the new stepsize is calculated and the scheme is followed once again, until the solution can be updated.

The optimal step size sh can be determined by multiplying the scalar s with the current step size h. The scalar s is

$$
s = \left(\frac{\text{tol}\, h}{2|z_k + 1 - y_k + 1|}\right)^{1/5} \approx 0.9\left(\frac{\text{tol}\, h}{|z_k + 1 - y_k + 1|}\right)^{1/5} \tag{3.4}
$$

See [8],[9] for additional information.

3.2 Poincaré Map for Cycle Orbits

The motion of the two-legged walker is assumed to be periodic. Therefore the dynamic behavior can be analyzed by using Poincaré mapping. A full periodic

gait cycle is a trajectory that corresponds to a closed orbit $p(x)$, where $x \in \mathbf{R}^n$ represents the state vector. A sectional plane $\mathbf{U} \in \mathbf{R}^{n-1}$ can be chosen that transverses with the closed periodic orbit at the point $x_i^* \in \mathbf{R}$ is the *Poincaré section* if $\mathbf{U}(x) \cdot n(x) \neq 0$ holds where $n(x)$ is the tangent plane of $p(x)$ at $x_i^* \in \mathbf{R}$. Let $\mathbf{\Sigma} \subset \mathbf{U}$ be a small section locally about $x_i^* \in \mathbf{R}$. Pertubating the closed orbit $p(x)$ starts from initial $x \in \mathbf{\Sigma}$. The pertubated orbit will return to the section $\mathbf{\Sigma}$ again and again, where the first return to the section is referred to as the *Poincaré map* or the first return map that is represented by an operator $S(x)$ with $x^* = S(x^*)$ where $x^* \in \mathbf{\Sigma}$ is called the fixed point of the step-to-step map $S(x)$. In case of the simplest walker the plane can be chosen at just after heelstrike, so the jump equations have already been applied. This corresponds to a plane spanned by $\phi = 2\theta$. Now the fixed point x^* describes the state of the angles θ and ϕ and angular velocities $\dot{\theta}$ and $\dot{\phi}$ just after heelstrike, which now is considered the beginning of the gait cycle. To find these fixed points a Newton-Raphson Methods is applied on the function $S(x)$. (See [10], [11])

3.3 Newton-Raphson Method

The Newton-Raphson Method is an efficient algorithm for finding approximations to the zeros (or roots) of a real-valued multidimensional function $F(x)$. But in this case the aim is not to find the roots of our step-to-step function $S(x)$, but the roots of the difference between this function and its state vector x, so the roots of the function

$$G(x) = S(x) - x \tag{3.5}$$

Now let x_0 be a good estimate of the fixed point x^*, for which $G(x^*) = 0$, and let $x^* = x_0 + h$. Since the true root of $G(x)$ is x^*, and $h = x^* - x_0$, the number h measures how far the estimate x_0 is from the root of the function $G(x)$. Since x_0 is supposed to be a good estimate, h is small number and therefore we can use the linear (tangent line) approximation to conclude that

$$0 = G(x^*) = G(x_0 + h) \approx G(x_0) + J(x_0)h \tag{3.6}$$

where $J(x)$ is the Jacobian matrix of $G(x)$

$$\mathbf{J} = \left[\frac{\partial G(x)}{\partial x}\right] \tag{3.7}$$

and therefore, unless $J(x_0)$ is singular,

$$h \approx -J(x_0)^{-1}G(x_0) \tag{3.8}$$

It follows that

$$x^* = x_0 + h \approx x_0 - J(x_0)^{-1}G(x_0) \tag{3.9}$$

Our new estimate x_1 of x^* is therefore given by

$$x_1 = x_0 - J(x_0)^{-1} G(x_0) \tag{3.10}$$

This scheme is iterated until the new estimate value x_n approaches the fixed point x^* within a given tolerance.

3.4 Stability of the fixed points

Now the fixed points are found, the local stability of the closed orbit $p(x)$ can be determined by the eigenvalues of the Jacobian matrix

$$\mathbf{J} \equiv \left[\frac{\partial S(x)}{\partial x} \right] \Big|_{x=x^*} \tag{3.11}$$

of the step-to-step map $S(x)$. This Jacobian matrix describes locally the relationship between the difference between a point $x \in \mathbf{\Sigma}$ and the fixed point x^*, and the difference between the Poincaré map of that point $S(x)$ and the fixed point x^* by

$$(S(x) - x^*) = \overline{J}\,(x - x^*) \tag{3.12}$$

The eigenvalues of **J** are called characteristic multipliers. **J** is constructed by numerically evaluating $f(x)$ several times n a small neighborhood of x^* in Σ. If the Jacobian **J** has all its eigenvalues λ_i $i = 1, ..., n-1$ inside the unit circle, the map $S(x)$ and its corresponding closed orbit $p(x)$ are asymptotic stable. If the Jacobian **J** has any eigenvalue λ_i outside the unit circle, the map $S(x)$ and its corresponding closed orbit $p(x)$ are unstable. If an eigenvalue λ_i had magnitude of one, the map $f(x)$ and its corresponding closed orbit $p(x)$ is marginally stable. (see [10], [11])

3.5 Responsive Behavior of Dynamics

A way to analyze the responsive behavior of the dynamics of the cyclic walking motion for the simplest walking model is to apply a cell-to-cell mapping method described in Hsu [12] and used by Schwab [3]. The region of all applicable initial conditions are subdivided in a large number (N) of cells, all unapplicable conditions in a small number (z) so called sink cells. Now the cell that contains the fixed point found by the previous methods is indicated as the fixed point cell. The step-to-step function $S(x)$ is now applied to the center of the N applicable cells, which will make them point to other cells. A sequence of these pointers will eventually end in one of the (z) sink cells or in a repetitive cycle. This can be either the fixed point cell or a number of cells (comparable to multi-period walking described by Garcia et al. [1]). When a sequence reaches a

repetitive cycle, all cells in this sequence are labeled stable. In case an unstable sink cell is reached, all cells are labeled unstable. As soon as a known cell is encountered, either stable or unstable, all cells in the sequence are labeled as such. Applying this method results in a list of all cells which will lead to the stable gait cycle. These results are as accurate as the discretization used for making the cells. Increasing or decreasing the resolution can result in small changes of the borders of the region.

Chapter 4

Results

4.1 Period-One Gait Cycles

By applying the RKF45 method to the equations of motion of the simplest walker described in 2.3.2, we can now simulate gait cycles for certain slope angles γ and initial conditions x. A typical plot of the stance an swing angle θ and ϕ over one step is shown in figure 4.1.

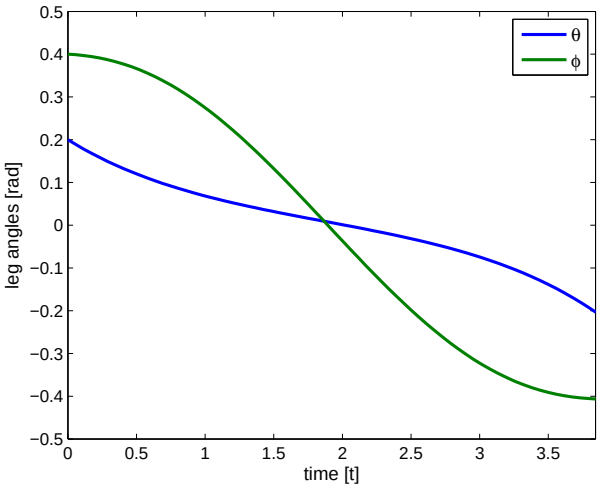

Figure 4.1: Angles of the swing and the stance leg during the swing phase

4.2 Fixed points

The Newton-Raphson methods, described in section 3.3, can be applied to the stride function $S(x)$ to find the fixed points of the simplest model for different

values of γ. For each value of γ two different solutions can be found, this is dependent on the initial estimate of the fixed point. These solutions can bee seen in figure 4.2. The long period solutions have a slightly larger stance angle at the fixed point than the short period solutions.

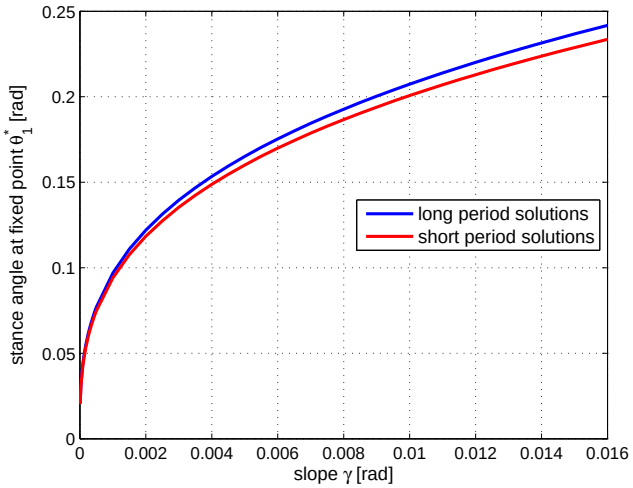

Figure 4.2: Stance angle at fixed point as a function of the slope γ

4.3 Stability

The stability of these fixed points can be analyzed by checking the Eigenvalues of the Jacobian at the fixed points as is described in section 3.2. As mentioned in section 3.2 the stability is checked at heelstrike, so the Poincaré plane is chosen at $\phi = 2\theta$. Applying Poincaré mapping means a reduction of dimensions of one, because the section is lower in dimension than the phase space. In this case however there is an extra loss of one dimension. The physical reason for this is that the swing leg only has a mass at the foot, so it makes no contribution to the angular momentum of the system about the new contact point just before heelstrike. This can also been seen in equation 2.5, where the rank of h is only 2, so the transition rule reduces this problem from 4D to 2D. For the short period solutions, the eigenvalues are shown in figure 4.3. It is clear that for all the slope angles one of the eigenvalues has a magnitude larger than one. Therefore, all the short period solutions are regarded unstable.

For the long period solutions however, for a certain range of the slope γ the magnitude of all Eigenvalues are smaller than one. As can be seen in Figure 4.4, this holds for slopes γ ranging from 0 up to 0.0151 rad.

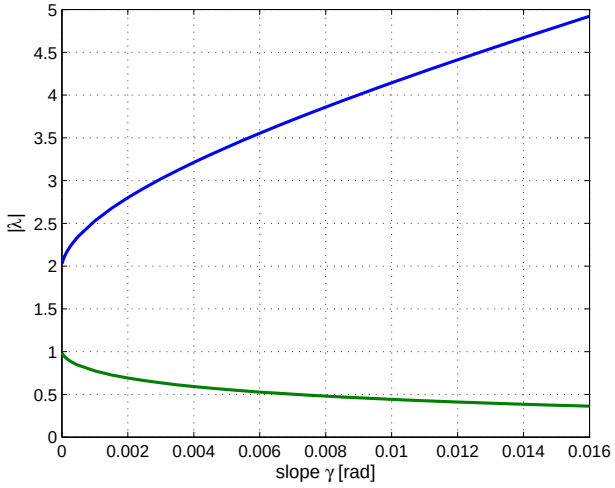

Figure 4.3: Eigenvalues of the short period solutions as a function of the slope γ

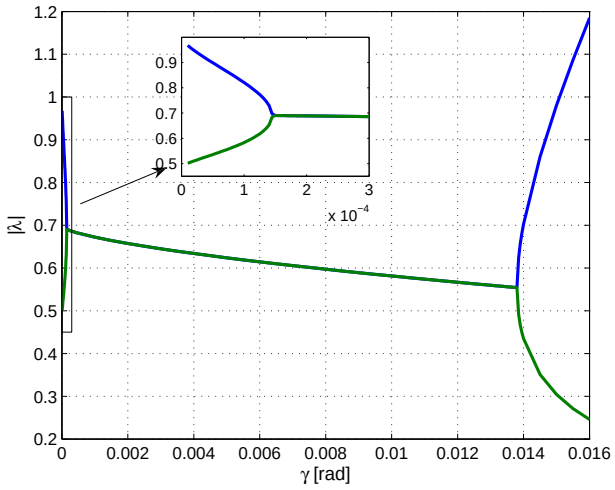

Figure 4.4: Eigenvalues of the long period solutions as a function of the slope γ

4.4 Responsive Behavior of Dynamics

The basin of attraction is known to be a small thin region when viewed with θ and $\dot{\theta}$ as axes. Therefore it can better be visualized zoomed and sheared with θ and $\theta + \dot{\theta}$ as axes. The black area corresponds to all the initial conditions

which eventually will lead to the fixed point, so to the stable gait cycle. When started outside the area the walker will possibly make one or a few steps, but eventually fall either forwards or backwards. Figures 4.5 and 4.6 are obtained with application of the cell mapping method with a discretization of 200 x 250 points ($\Delta\theta = 0.002$ [rad], $\Delta(\theta + \dot{\theta}) = 0.0002$ [rad])

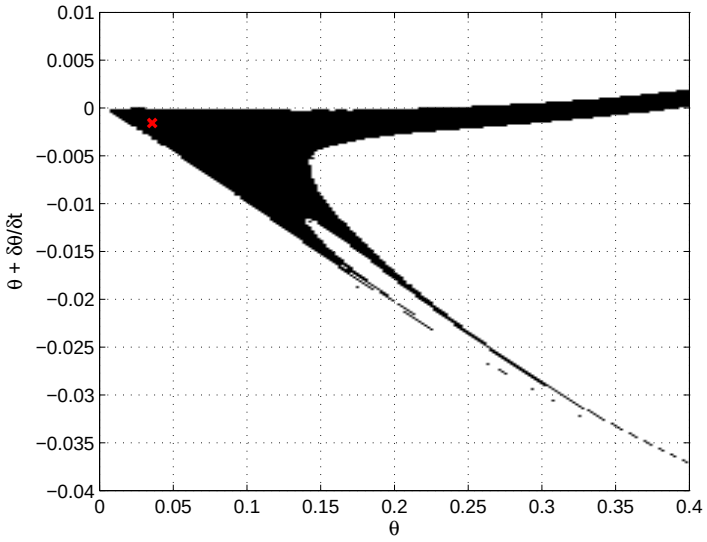

Figure 4.5: Responsive behavior of the dynamics of of the simplest walker on a slope γ of 0.00005 rad. The red 'x' indicates the location of the fixed point.

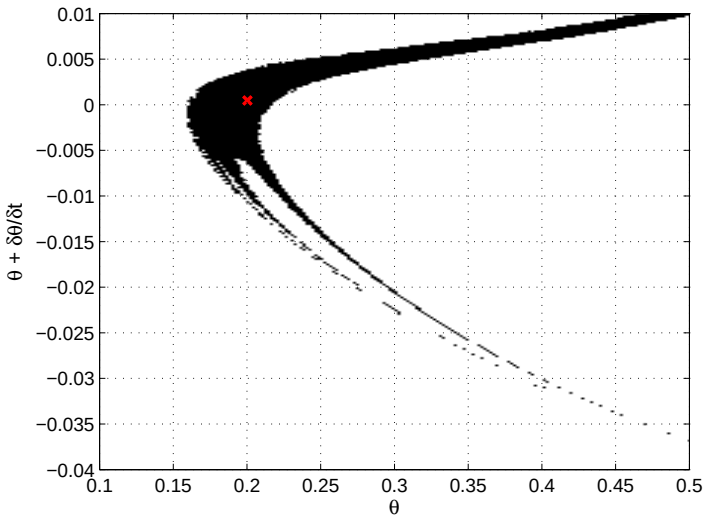

Figure 4.6: Responsive behavior of the dynamics of of the simplest walker on a slope γ of 0.009 rad. The red 'x' indicates the location of the fixed point.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis the results from Garcia et al. [1] are regenerated based on own coding, using a Runge-Kutta-Fehlberg integration method, Poincaré mapping and stability analysis based on eigenvalues. These results show that for a slope angle γ between 0 and 0.015 rad. the simplest walker model has a stable gait cycle, so stable passive-dynamic walking exists within this region. To get a better understanding in the range of stable initial conditions, cell-to-cell mapping is applied which resulted in visualizations of the responsive behavior of dynamics of the system comparable to the results of Schwab [3]. Together these results give more insight in the possibility of a two-legged walker to perform an uncontrolled stable walking motion on a sloped ground.

5.2 Future Work

Now it is proven for the simplest walker that stable gait cycles exist in 2D, this sequence of methods could be used for analyzing the gait cycles of a simplest walker model in a 3D environment. This will need a proper description of the equations of motions during the swing phase, the collision condition and the jump equations. Another interesting study following this research is to investigate the gait cycles of a powered walking model, in order to make the simplest walker be able to walk on level ground and to extend the basin of attraction of the stable gait cycles.

Appendix A

M-files for simulation

A.1 M-files

The following list contains all the m-files that are used for the simulations, with a brief explanation of their functions.

m-files	**description**
find_all.m	finds and plots all fixed points and the magnitude of their eigenvalues for a range of slope angles, calls *search*
jacob.m	returns jacobian for given situation, calls *stride*
search.m	returns the fixed point for given slope angle, calls *jacob* and *stride*
stride.m	returns the state after one step given its initial conditions by applying RKF45 method, calls *xderivs*
xderivs.m	returns derivatives for a given state
run_grid.m	performs *step_sequence* for every cell in the grid and plots the result, calls *search*
step_sequence	return whether a sequence of steps starting with the given initial conditions will result into failure or stable gait cycle, calls *xderivs*

A.1.1 find_all.m

```
clear all
clc

k=1;
x(:,1) = 0.06;              % initial value for theta1
x1(:,1) = -x(1,1)-0.002;    % initial value for theta1d

for gam = 1e-5:stepsize:16e-3       % various stepsizes can be used
```

```
    [th1(:,k+1),th1d(:,k+1),J] = search(th1(:,k),th1d(:,k),gam);
    if th1(:,k+1)==99       % terminates if fixed point can't be found
        error('Fixed point could not be found for gamma = %g',gam)
    end
    lambda(:,k) = eig(J);   % eigenvalues of jacobian at fixed point
    gamma(:,k) = gam;
    k = k + 1;
end

th1  = th1(2:length(th1));    % lose the initial value for theta1
th1d = th1d(2:length(th1d));  % lose the initial value for theta1d

figure(1)
plot(gamma,th1)
xlabel('\gamma [rad]')
ylabel('stance angle at fixed point \theta_1^* [rad]')
grid

figure(2)
plot(gamma,abs(lab(1:2,:)))
xlabel('\gamma [rad]')
ylabel('|\lambda|')
grid
```

A.1.2 jacob.m

```
function [J] = jacob(th1,th1d,gam)
%
% [J] = jacob(theta1,theta1dot,gamma)
%
% M-file to find the Jacobian of the stride function S around the
% point [theta1,theta2,theta1dot,theta2dot] by solving the equation
%
% [S(X+pert) - S(X)] = J*[(X+pert) - X]
%
% Makes use of:
% - stride.m
% - xderivs.m

% Initial conditions
th2 = pi-2*th1;
th2d = -th1d*(1-cos(2*th1));

x = [th1 th2 th1d th2d]';
```

```
tol = 1e-12;        % tolerance
pert = sqrt(tol);   % perturbation

Sx = stride(x,gam);       % step on initial conditions

for i=1:4
    xp(:,i) = x;
    xp(i,i) = x(i)+pert;  % apply perturbation on ith condition

    Sxp(:,i) = stride(xp(:,i),gam); % step on perturbed conditions

    G(:,i) = xp(:,i)  - x;  % diff betw. perturbed and initial cond.
    H(:,i) = Sxp(:,i) - Sx; % diff betw. perturbed and initial step
end

J = H/G; % solves H = J*G
```

A.1.3 search.m

```
function [theta1 theta1d J] = search(th1,th1d,gam)
%
% [theta1* theta1d*] = search(initial theta1,initial theta1d,gamma)
%
% This program searches for the fixed point for the slope gamma,
% starting with the initial conditions (theta1, theta1d), by
% applying the Newton-Raphson iteration method
%
% x(i+1) = x(i) - g(x)/g'(x)
%
% Makes use of:
% - jacob.m
% - stride.m
% - xderivs.m

tol = 1e-12;
k = 0;

% Initial conditions
th2  = pi-2*th1;
th2d = -th1d*(1-cos(2*th1));
x = [th1 th2 th1d th2d]';

dx = [1 1 1 1];   % larger than tol
```

```
while abs(dx(1)) > tol && abs(dx(3)) > tol && k < max_iterate
    Sx = stride(x,gam);
    g = Sx - x;
    J = jacob(Sx(1),Sx(3),gam);
    dx = (J - eye(4))^-1 * g;  % dx = g(x)/g'(x)
    x = x - dx;   % next estimate
    k = k + 1;
end

if k == max_iterate
    theta1=99;
    theta1d=99;
else
    theta1 = x(1);
    theta1d = x(3);
end
```

A.1.4 stride.m

```
function [Sx,xout,tout] = stride(x,gam)
%
% [final state Sx,xout,tout] = stride(initial states x,slope gamma)
%
% M-file that simulates the stride function S of the 2-legged walker.
% It contains the continuous walking motion and the discontinuous
% transition rule at heelstrike
%
% Makes use of:
% - xderivs.m

% Set parameters
l  = 1;     % leg length
m1 = 1;     % hip mass
m2 = 0;     % foot mass
g  = 1;     % gravity
beta = m2/m1;

tol = 1e-10;

param=[beta,gam,g,l];

% The Runge-Kutta-Fehlberg coefficients:
beta  = [ [    1      0      0     0      0    0]/4
```

```
          [    3      9      0     0      0    0]/32
          [ 1932  -7200   7296     0      0    0]/2197
          [ 8341 -32832  29440  -845      0    0]/4104
          [-6080  41040 -28352  9295  -5643    0]/20520 ]';
gamma = [ [902880  0  3953664  3855735  -1371249  277020]/7618050
          [ -2090  0    22528    21970    -15048  -27360]/752400 ]';
pow = 1/5;
f = zeros(length(x),6);

k = 1;
t = 0;
tfinal = 10;
hmax = (tfinal - t)/16;
h = hmax/8;

xout(:,k) = x;
tout(:,k) = t;

% First part before 'scuffing'
while  ((x(2)+h*x(4) <= pi-2*(x(1)+h*x(3))))...
        && t < tfinal && abs(x(1)) < pi/2

    % Compute the slopes
    f(:,1) = xderivs(x,param);
    for w = 1:5
        f(:,w+1) = xderivs(x+h*f*beta(:,w),param);
    end

    % Estimate the error and the acceptable error
    delta = norm(h*f*gamma(:,2),'inf');
    tau = tol*max(norm(x,'inf'),1.0);

    % Update the solution only if the error is acceptable
    if delta <= tau
        t = t + h;
        x = x + h*f*gamma(:,1);
        k = k + 1;
        xout(:,k) = x;
        tout(:,k) = t;
    end

    % Update the step size
    if delta ~= 0.0
        h = min(hmax, 0.9*h*(tau/delta)^pow);
```

```
    end
end

% Terminate if walker has fallen down
if abs(x(1)) > pi/2
    t=tfinal;
end;

% Second part after 'scuffing'
while  ((x(2)+h*x(4) >= pi-2*(x(1)+h*x(3))) ) ...
        && t < tfinal && abs(x(1)) < pi/2

    % Compute the slopes
    f(:,1) = xderivs(x,param);
    for w = 1:5
        f(:,w+1) = xderivs(x+h*f*beta(:,w),param);
    end

    % Estimate the error and the acceptable error
    delta = norm(h*f*gamma(:,2),'inf');
    tau = tol*max(norm(x,'inf'),1.0);

    % Update the solution only if the error is acceptable
    if delta <= tau
        t = t + h;
        x = x + h*f*gamma(:,1);
        k = k + 1;
        xout(:,k) = x;
        tout(:,k) = t;
    end;

    % Update the step size
    if delta ~= 0.0
        h = min(hmax, 0.9*h*(tau/delta)^pow);
    end
end;

% Terminate if walker has fallen down
if abs(x(1)) > pi/2
    t=tfinal;
end;

% Third part starts right before heelstrike
if t<tfinal
```

```
    while abs(pi-2*x(1)-x(2)) > tol/100
        h=(pi-2*x(1)-x(2))/(x(4)+2*x(3));
        f(:,1) = xderivs(x,param);
        for w = 1:5
            f(:,w+1) = xderivs(x+h*f*beta(:,w),param);
        end

        t = t + h;
        x = x + h*f*gamma(:,1);
        k = k + 1;
        xout(:,k) = x;
        tout(:,k) = t;
    end;
end;

% Now heelstrike occurs, jump equations give the
% final states

Sx(1)=-y(1);
Sx(2)=pi-2*y(1);
Sx(3)=(y(3)*cos(2*y(1)))/(1+param(1)*(sin(2*y(1)))^2);
Sx(4)=-y(3)*(1+cos(y(2)));
```

A.1.5 xderivs.m

```
function xdot = xderivs(x,param)
%
% xdot = xderivs(x,parameters)
%
% This m-file calculates the derivatives according to the equations
% of motion of the simplest walker model
%
% [M(x)]xddot + [V(x,xdot)] + [G(x)] = 0

beta = param(1);
gam = param(2);
g = param(3);
l = param(4);

c02 = cos(x(2));
s02 = sin(x(2));

s01g  = sin(x(1)-gam);
s012g = sin(x(1)+x(2)-gam);
```

```
M(1,1) = 1+2*beta*(1+cO2);
M(1,2) = beta*(1+cO2);
M(2,1) = 1+cO2;
M(2,2) = 1;

V(1) = -beta*sO2*x(4)*(2*x(3)+x(4));
V(2) = sO2*x(3)^2;

G(1) = -g/l*(beta*sO12g+sO1g*(1+beta));
G(2) = -g/l*sO12g;

xddot=M\(-V'-G');

xdot(1)=x(3);
xdot(2)=x(4);
xdot(3)=xddot(1);
xdot(4)=xddot(2);

xdot = xdot';
```

A.1.6 run_grid.m

```
function [] = run_grid(th1_0,th1_m,m,...
                                th1d_0,th1d_n,n)

% [A,th1_vec,th1d_vec] =
%       run_grid(th1_0,th1_m,m,th1d_0,th1d_n,n)
%
% th1_0 = minimum for theta1
% th1_m = maximum for theta1
% th1d_0 = minimum for theta1+theta1d
% th1d_n = maximum for theta1+theta1d
% m,n = number of cells

clear A

global A

% Set Parameters
gam = 0.015; % slope angle
l   = 1;     % leg length
m1  = 1;     % hip mass
m2  = 0;     % foot mass
```

```
g   = 1;     % gravity
beta= m2/m1;

tol = 1e-10;

%Cell-to-cell Mapping parameters
A=zeros(n,m);
step_th1 = (th1_m - th1_0) / m;
step_th1d = (th1d_n - th1d_0) / n;

%Determine fixed cell
[th1_fix,th1d_fix]=search(0.23,-0.23,gam);  % Finds the fixed point
j_fix = ceil( (th1_fix - th1_0) / step_th1 );
i_fix = ceil( ((th1d_fix+th1_fix) - th1d_0) / step_th1d );
A(i_fix,j_fix) = 3;     % Fixed cell = '3'

for i = 1:n
    for j = 1:m
        if A(i,j)==0
            theta1 = th1_0 + (j-1/2)*step_th1;
            theta1dot = (th1d_0 + (i-1/2)*step_th1d)-theta1;
            theta2= pi-2*theta1;
            theta2dot=-theta1dot*(1-cos(2*theta1));
            x = [theta1 theta2 theta1dot theta2dot];
            step_sequence(0,50,tol,x,beta,g,gam,l,...
                th1_0,th1_m,th1d_0,th1d_n,m,n);
        end
    end
    i
end

th1_vec = [th1_0:step_th1:th1_m]';
th1d_vec = [th1d_0:step_th1d:th1d_n]';

figure(1)
image(th1_vec,th1d_vec,A);
colormap([0 0 0;1 1 1;1 0 0])
axis xy
grid
xlabel('\theta')
ylabel('\theta + \delta\theta/\deltat')
hold on
plot(th1_fix,th1_fix+th1d_fix,'rx','MarkerSize',7,'LineWidth',2)
```

A.1.7 step_sequence.m

```
function []=step_sequence(t0,tfinal,tol,x,beta,g,gam,l,...
               th1_0,th1_m,th1d_0,th1d_n,m,n)

% funcion step_sequence(t0,tfinal,tol,x,beta,g,gam,l,...
%                th1_0,th1_m,th1d_0,th1d_n,m,n)
%
% th1_0 = minimum for theta1
% th1_m = maximum for theta1
% th1d_0 = minimum for theta1+theta1d
% th1d_n = maximum for theta1+theta1d
% m,n = number of cells
%
% M-file simulates a sequence of steps until the step ends
% in a cell of which the state (stable/unstable) is known
%
% Makes use of:
% - xderivs.m

global A

param=[beta,gam,g,l];

% The Runge-Kutta-Fehlberg coefficients:
beta  = [ [    1      0      0     0      0    0]/4
          [    3      9      0     0      0    0]/32
          [ 1932  -7200   7296     0      0    0]/2197
          [ 8341 -32832  29440  -845      0    0]/4104
          [-6080  41040 -28352  9295  -5643    0]/20520 ]';
gamma = [ [902880  0  3953664  3855735  -1371249  277020]/7618050
          [ -2090  0    22528    21970    -15048  -27360]/752400 ]';
pow = 1/5;
f = zeros(length(x),6);

t = t0;

hmax = (tfinal - t)/16;
h = hmax/8;

tk=[]; xk=[]; i=[]; j=[];
tx=1;           % # of steps, 1 is initial condition
tk(tx,:)=t;     % time of steps
xk(tx,:)=x;     % state-vector of steps
```

```
step_th1 = (th1_m - th1_0) / m;     % cell size theta1
step_th1d = (th1d_n - th1d_0) / n;  % cell size (th1+th1d)

% First part before 'scuffing'
while (t < tfinal)

% If theta or theta1 are outside the boundaries, cycle is unstable
if (th1_0  >= x(1) || x(1) >= th1_m || ...
        th1d_0 >= (x(3)+x(1)) || (x(3)+x(1)) >= th1d_n)
    for w = 1:length(i)
        A(i(w),j(w)) = 2;   % Unstable = '2'
    end
    return
end

% Indeces of the current position
j(tx) = ceil( (x(1) - th1_0) / step_th1 );
i(tx) = ceil( ((x(3)+x(1)) - th1d_0) / step_th1d );

% If current position is known to be unstable, all previous are
% unstable
if A(i(tx),j(tx)) == 2;
    for w = 1:length(i)-1
        A(i(w),j(w)) = 2;
    end
    t=tfinal;
end

% If current position is known to be stable, or the fixed cell
% itself, all previous are stable
if A(i(tx),j(tx)) == 1 || A(i(tx),j(tx)) == 3;
    for w = 1:length(i)-1
        A(i(w),j(w)) = 1;   % Stable = '1'
    end
    t=tfinal;
end

while  ((x(2)+h*x(4) <= pi-2*(x(1)+h*x(3))))...
       && t < tfinal && abs(x(1)) < pi/2

       % Compute the slopes
       f(:,1) = xderivs(x,param);
       for w = 1:5
```

```
          f(:,w+1) = xderivs(x+h*f*beta(:,w),param);
       end

       % Estimate the error and the acceptable error
       delta = norm(h*f*gamma(:,2),'inf');
       tau = tol*max(norm(x,'inf'),1.0);

       % Update the solution only if the error is acceptable
       if delta <= tau
           t = t + h;
           x = x + h*f*gamma(:,1);
       end;

       % Update the step size
       if delta ~= 0.0
            h = min(hmax, 0.9*h*(tau/delta)^pow);
       end
end

% Terminate if walker has fallen down
if abs(x(1)) > pi/2
    for w = 1:length(i)
        A(i(w),j(w)) = 2;
    end
    t=tfinal;
end;

while  ((x(2)+h*x(4) >= pi-2*(x(1)+h*x(3))) ) ...
   && t < tfinal && abs(x(1)) < pi/2

   % Compute the slopes
   f(:,1) = xderivs(x,param);
   for w = 1:5
       f(:,w+1) = xderivs(x+h*f*beta(:,w),param);
   end

   % Estimate the error and the acceptable error
   delta = norm(h*f*gamma(:,2),'inf');
   tau = tol*max(norm(x,'inf'),1.0);

   % Update the solution only if the error is acceptable
   if delta <= tau
       t = t + h;
       x = x + h*f*gamma(:,1);
```

```
   end;

   % Update the step size
   if delta ~= 0.0
       h = min(hmax, 0.9*h*(tau/delta)^pow);
   end
end;

% Terminate if walker has fallen down
if abs(x(1)) > pi/2
    for w = 1:length(i)
        A(i(w),j(w)) = 2;
    end
    t=tfinal;
end;

horig=h;

% Third part starts right before heelstrike
if t<tfinal
    while abs(pi-2*x(1)-x(2)) > tol/100
        h=(pi-2*x(1)-x(2))/(x(4)+2*x(3));
        f(:,1) = xderivs(x,param);
        for w = 1:5
            f(:,w+1) = xderivs(x+h*f*beta(:,w),param);
        end

        t = t + h;
        x = x + h*f*gamma(:,1);
    end;
end;

if t<tfinal

    h=horig;

    % Now heelstrike occurs, jump equations give the
    % final states

    x(3)=(x(3)*cos(2*x(1)))/(1+param(1)*(sin(2*x(1)))^2);
    x(4)=-x(3)*(1+cos(x(2)));

    x(1)=-x(1);
    x(2)=pi-2*x(1);
```

```
    tx=tx+1;
    tk(tx,:)=t;
    xk(tx,:)=x';

end;
end;
```

A.2 M-files of Garcia

The following list contains the m-files from Garcia that simulate the simplest walker's motion, with a brief explanation of their function

m-files	description
int_ doubpend.m	integrates the motion of the two-legged walker including the jump equations for a given time period by applying the RKF45 method, calls *yderivs_ doubpend*
int_ doubpend_ step.m	returns the states after a singel step by integrates the motion of the two-legged walker including the jump equations by applying the RKF45 method, calls *yderivs_ doubpend*
it_ doubpend	finds the fixed point for a given slope angle, calls *int_ doubpend_ step*
run_ doubpend	runs *int_ doubpend* and plots the output
yderivs_ doubpend	returns derivatives for a given state

A.2.1 int_doubpend.m

```
function [tout,yout]=int_doubpend(t0,tfinal,tol,y0,beta,g,gam,l);
% kludge to run mex file- assumes g=l=1, stores [beta gamma] in BETA
betavec=[beta,gam,g,l];
% now there are two different betas floating around

% Does forward integration for a "simplest" walker with
% 2 DOF.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       C.B. Moler, 3-25-87, 10-5-91, 6-3-93.
%       Copyright (c) 1984-93 by The MathWorks, Inc.
%       Modified for walking simulations
%       by Mariano Garcia (garcia@tam.cornell.edu), 1996
%%%%%%%%%%%%%%%%%%%%%%%%%% Program begins  %%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization
% The Fehlberg coefficients:
beta  = [ [    1      0      0     0      0    0]/4
          [    3      9      0     0      0    0]/32
          [ 1932  -7200   7296     0      0    0]/2197
          [ 8341 -32832  29440  -845      0    0]/4104
          [-6080  41040 -28352  9295  -5643    0]/20520 ]';
gamma = [ [902880  0  3953664  3855735  -1371249  277020]/7618050
          [ -2090  0    22528    21970    -15048  -27360]/752400 ]';
pow = 1/5;
```

```
f = zeros(length(y0),6);

if nargin < 5, tol = 0.001; end
t = t0;

hmax = (tfinal - t)/16;
h = hmax/8;
y = y0(:);
chunk = 128;
tout = zeros(chunk,1);
yout = zeros(chunk,length(y));
k=1;
tout(k) = t;
yout(k,:) = y.';
tau = tol * max(norm(y, 'inf'), 1);

clf reset;

axis([-2 2 -2 2]);

% Get coordinates for graphics
% cfx,cfy are coordinates of center of stance foot in a world
% fixed frame
% hip, knee, and end are coordinates of hip, swing knee, and end
% foot center

hipx=-l*sin(y(1));
hipy=l*cos(y(1));
endx=hipx-l*sin(y(1)+y(2));
endy=hipy-l*cos(pi-y(1)-y(2));

%plot graphics initially
walker_data=[0,0;hipx,hipy;endx,endy];
walker_bits=line('xdata',walker_data(:,1),'ydata',walker_data(:,2),...
'erasemode','xor');
drawnow;

%%%%%%%%%%%%%%%%%%%%%%  Main Loop Begins  %%%%%%%%%%%%%%%%%%%%%%%%%%%%
while (t < tfinal)

while  ((y(2)+h*y(4) <= pi-2*(y(1)+h*y(3))))...
 & t < tfinal & abs(y(1)) < pi/2
         % Compute the slopes
     f(:,1) = yderivs_doubpend(y,betavec);
```

```
        for j = 1:5
              f(:,j+1) = yderivs_doubpend(y+h*f*beta(:,j),betavec);
        end

        % Estimate the error and the acceptable error
   delta = norm(h*f*gamma(:,2),'inf');
   tau = tol*max(norm(y,'inf'),1.0);

        % Update the solution only if the error is acceptable
        ts = t;
        ys = y;
        if delta <= tau
           t = t + h;
        y = y + h*f*gamma(:,1);
           k = k+1;
              if k > length(tout)
              tout = [tout; zeros(chunk,1)];
            yout = [yout; zeros(chunk,length(y))];
              end
       tout(k) = t;
    yout(k,:) = y.';

    %Update Graphics
    hipx=-l*sin(y(1));
    hipy=l*cos(y(1));

    endx=hipx-l*sin(y(1)+y(2));
    endy=hipy-l*cos(pi-y(1)-y(2));

    walker_data=[0,0;hipx,hipy;endx,endy];
        set(walker_bits,'xdata',walker_data(:,1),...
                                'ydata',walker_data(:,2));

    drawnow;

    end;

        % Update the step size
      if delta ~= 0.0
      h = min(hmax, 0.9*h*(tau/delta)^pow);
     end
end;

if abs(y(1)) > pi/2
```

```
t=tfinal
end;

while  ((y(2)+h*y(4) >= pi-2*(y(1)+h*y(3))) ) ...
 & t < tfinal & abs(y(1)) < pi/2
         % Compute the slopes
         f(:,1) = yderivs_doubpend(y,betavec);
            for j = 1:5
              f(:,j+1) = yderivs_doubpend(y+h*f*beta(:,j),betavec);
            end

        % Estimate the error and the acceptable error
   delta = norm(h*f*gamma(:,2),'inf');
   tau = tol*max(norm(y,'inf'),1.0);

        % Update the solution only if the error is acceptable
        ts = t;
        ys = y;
        if delta <= tau
           t = t + h;
    y = y + h*f*gamma(:,1);
           k = k+1;
              if k > length(tout)
                  tout = [tout; zeros(chunk,1)];
                  yout = [yout; zeros(chunk,length(y))];
              end
           tout(k) = t;
        yout(k,:) = y.';

        %Update Graphics
        hipx=-l*sin(y(1));
        hipy=l*cos(y(1));

        endx=hipx-l*sin(y(1)+y(2));
        endy=hipy-l*cos(pi-y(1)-y(2));

        walker_data=[0,0;hipx,hipy;endx,endy];
        set(walker_bits,'xdata',walker_data(:,1),...
                                'ydata',walker_data(:,2));
        drawnow;

        end;

        % Update the step size
```

```
      if delta ~= 0.0
      h = min(hmax, 0.9*h*(tau/delta)^pow);
     end
end;

if abs(y(1)) > pi/2
t=tfinal
end;

    horig=h;
%%%%%%%%%%%%% End THree Link Loop %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% now we are less than h seconds away from strike
% need to zero in on collision hypersurface
    q=0;

if t<tfinal
    while abs(pi-2*y(1)-y(2)) > tol/100
      h=(pi-2*y(1)-y(2))/(y(4)+2*y(3));
            f(:,1) = yderivs_doubpend(y,betavec);
            for j = 1:5
              f(:,j+1) = yderivs_doubpend(y+h*f*beta(:,j),betavec);
            end

%don't worry about error computation in here, since h small enough
% assume solution is within tolerable error

      t = t + h;
    y = y + h*f*gamma(:,1);
          k = k+1;

        if k > length(tout)
          tout = [tout; zeros(chunk,1)];
          yout = [yout; zeros(chunk,length(y))];
        end
      tout(k) = t;
      yout(k,:) = y.';
      q=q+1;

end;
end;

if t<tfinal
```

```
%Update Graphics

        hipx=-l*sin(y(1));
    hipy=l*cos(y(1));
    endx=hipx-l*sin(y(1)+y(2));
    endy=hipy-l*cos(pi-y(1)-y(2));
    walker_data=[0,0;hipx,hipy;endx,endy];
        set(walker_bits,'xdata',walker_data(:,1),...
                                'ydata',walker_data(:,2));

        drawnow;

h=horig;

% now we hit strike. AMB about strike point gives

y(3)=(y(3)*cos(2*y(1)))/(1+betavec(1)*(sin(2*y(1)))^2);
y(4)=-y(3)*(1+cos(y(2)));

y(1)=-y(1);
y(2)=pi-2*y(1);

y
t

k=k+1;
tout(k) = t;
yout(k,:) = y.';

end;
end;

tout = tout(1:k);
yout = yout(1:k,:);
```

A.2.2 int_doubpend_step.m

```
function [t,y]=int_doubpend_step(t0,tfinal,tol,y0,beta,g,gam,l)
% kludge to run mex file- assumes g=l=1, stores [beta gamma] in BETA
betavec=[beta,gam,g];
% now there are two different betas floating around

global m1 m2
```

```
%%%%%%%%%%%%%%%%%%%%%%%% Program begins  %%%%%%%%%%%%%%%%%%%%

% Initialization
% The Fehlberg coefficients:
beta  = [ [    1      0      0     0      0    0]/4
    [    3      9      0     0      0    0]/32
    [ 1932  -7200   7296     0      0    0]/2197
    [ 8341 -32832  29440  -845      0    0]/4104
    [-6080  41040 -28352  9295  -5643    0]/20520 ]';
gamma = [ [902880  0  3953664  3855735  -1371249  277020]/7618050
    [ -2090  0    22528    21970    -15048  -27360]/752400 ]';

pow = 1/5;
f = zeros(length(y0),6);
if nargin < 5, tol = 0.001; end
t = t0;
hmax = (tfinal - t)/16;
h = hmax/8;
y = y0(:);

%%%%%%%%%%%%%%%%%%%%  Main Loop Begins  %%%%%%%%%%%%%%%%%%%%%%
while  ((y(2)+h*y(4) <= pi-2*(y(1)+h*y(3))))...
        && abs(y(1)) < pi/2

    % Compute the slopes
    f(:,1) = yderivs_doubpend(y,[betavec,1]);
    for j = 1:5
        f(:,j+1) = yderivs_doubpend(y+h*f*beta(:,j), ...
            [betavec,1]);
    end

    % Estimate the error and the acceptable error
    delta = norm(h*f*gamma(:,2),'inf');
    tau = tol*max(norm(y,'inf'),1.0);

    % Update the solution only if the error is acceptable
    if delta <= tau
        t = t + h;
        y = y + h*f*gamma(:,1);
    end;

    % Update the step size
    if delta ~= 0.0
        h = min(hmax, 0.9*h*(tau/delta)^pow);
```

```
    end
end;

while  ((y(2)+h*y(4) >= pi-2*(y(1)+h*y(3)))) ...
        && abs(y(1)) < pi/2

    % Compute the slopes
    f(:,1) = yderivs_doubpend(y,[betavec,l]);
    for j = 1:5
        f(:,j+1) = yderivs_doubpend(y+h*f*beta(:,j), ...
            [betavec,l]);
    end

    % Estimate the error and the acceptable error
    delta = norm(h*f*gamma(:,2),'inf');
    tau = tol*max(norm(y,'inf'),1.0);

    % Update the solution only if the error is acceptable
    if delta <= tau
        t = t + h;
        y = y + h*f*gamma(:,1);
    end;

    % Update the step size
    if delta ~= 0.0
        h = min(hmax, 0.9*h*(tau/delta)^pow);
    end
end;

horig=h;

% now we are less than h seconds away from strike
q=0;
if t<tfinal
    while q<6
        h=(pi-2*y(1)-y(2))/(y(4)+2*y(3));

        % Compute the slopes
        f(:,1) = yderivs_doubpend(y,[betavec,l]);
        for j = 1:5
            f(:,j+1) = yderivs_doubpend(y+h*f*beta(:,j), ...
                [betavec,l]);
        end
```

```
        % don't worry about error computation in here, since h
        % small enough assume solution is within tolerable error
        t = t + h;
        y = y + h*f*gamma(:,1);

        q=q+1;
    end;
end;

h=horig;

% now we hit strike. AMB about strike point gives

Lvpre = -m1*l*y(3)*sin(y(1));

y(3)=(y(3)*cos(2*y(1)))/(1+betavec(1)*(sin(2*y(1)))^2);
y(4)=-y(3)*(1+cos(y(2)));

y(1)=-y(1);
y(2)=pi-2*y(1);

Lvpost = -m1*l*y(3)*sin(y(1)) + ...
         m2*(-2*l*sin(y(1))*y(3)*cos(y(1))-y(4)*l*sin(y(1)));
```

A.2.3 it_doubpend.m

```
% Created by Mariano Garcia(garcia@tam.cornell.edu)
% Feb 1996 to find fixed points
% for pointfoot walker. Assumes that the file int_doubpend_step.m
% exists to take the initial conditions through one step
% and heelstrike and return the initial conditions for
% the next step.
format long
% leg parameters
global graphics
l = 1;
m1 = 1;
m2= 0;
beta=m2/m1;
graphics = 1;

% slope, foot radius,  and gravity

gam = 0.009;
```

```
g = 1;

%initial conditions

theta1 =  0.12196164739182;
theta2= pi-2*theta1;
theta1dot = -0.12535154656863;
theta2dot=-theta1dot*(1-cos(2*theta1));

%initial conditions gam=0.009
theta1 = 0.20031089581004;
theta2= pi-2*theta1;
theta1dot = -0.19983246430015;
theta2dot=-theta1dot*(1-cos(2*theta1));

y0 = [theta1 theta2 theta1dot theta2dot];

tol = 1e-10;
zeta = sqrt(tol); % good rule of thumb for forward difference
tfinal =12; % not necessary
n=0;
iterations=20;
fixedpoint = 'false';

while (strcmp(fixedpoint,'true')==0) && (n < iterations)

    [t,y]=int_doubpend_step(0,tfinal,tol,y0,beta,g,gam,l); %get g
    gold(1) = y(1)-y0(1);
    gold(2) = y(3)-y0(3);

    yg=y0; % perturb initial conditions, get column 1 of Jac of G
    yg(1)=y0(1)+zeta;
    yg(2)=pi-2*yg(1);
    yg(4)=-yg(3)*(1-cos(2*yg(1)));

    [t,y]=int_doubpend_step(0,tfinal,tol,yg,beta,g,gam,l);
    gnew(1) = y(1)-yg(1);
    gnew(2) = y(3)-yg(3);
    dg(:,1)=(gnew-gold)';
    jac(1)=y(1)-y0(1);
    jac(2)=y(3)-y0(3);
    J(:,1)=(jac./zeta)';

    yg=y0; % perturb thetadot, get second column
```

```
    yg(3)=y0(3)+zeta;
    yg(4)=-yg(3)*(1-cos(2*yg(1)));

    [t,y]=int_doubpend_step(0,tfinal,tol,yg,beta,g,gam,l);
    gnew(1) = y(1)-yg(1);
    gnew(2) = y(3)-yg(3);
    dg(:,2)=(gnew-gold)';
    jac(1)=y(1)-y0(1);
    jac(2)=y(3)-y0(3);
    J(:,2)=(jac./zeta)';

    dgdy=(1/zeta).*dg;

    dely= dgdy\(-gold)'; % get initial guess

    y0(1)=y0(1)+dely(1);
    y0(2)=pi-2*y0(1);
    y0(3)=y0(3)+dely(2);
    y0(4)=-y0(3)*(1-cos(2*y0(1)));

    if abs(dely) < tol/10
        fixedpoint = 'true';
    end;

    n=n+1
end;

I=eye(2); %compare eigenvalues
eig(dgdy+I)
eig(J)
```

A.2.4 run_doubpend.m

```
% All code by Mariano Garcia (garcia@tam.cornell.edu), 1998
% Some documentation of sorts and accompanying figures can
% be found in chaps 2 and 3 of my thesis, which can be downloaded from
% http://tam.cornell.edu/students/garcia/msghomepage.html
% or
% http://www.ccmr.cornell.edu/~ruinalab/
% also read the README.txt file.
% This simulation was used to produce results for the paper
% "The Simplest Walking Model: Stability, Complexity, And Scaling"
% ASME Journal of Biomechanical Engineering Vol 120, No. 2
% pp. 218-288, 1998.
```

```
% Note that theta2 is the exterior angle between the
% legs, not the same as phi in the ASME paper above.
% theta2 = pi - phi

% see the accompanying figure pointfoot.cartoon2.eps

global graphics
graphics=0;
format long
clf reset
clear

% leg parameters
l = 1;   % leg length
m1 = 1;  % hip mass
m2= 0;   % foot mass
beta=m2/m1;

gam = 0.009;  % slope, in rad
g = 1;        % gravity

%initial conditions gam=0.009 long step to 1e-10
% loc  = -0.19120868949940 +/- 0.556333636871061
theta1 = 0.20031090049483;
theta2= pi-2*theta1;
theta1dot = -0.19983247291764;
theta2dot=-theta1dot*(1-cos(2*theta1));

%initial conditions gam=0.009 short step to 1e-10
% loc = 4.000, 0.459
%theta1 = 0.19393736960624;
%theta2= pi-2*theta1;
%theta1dot = -0.20386692731962;
%theta2dot=-theta1dot*(1-cos(2*theta1));

y0 = [theta1 theta2 theta1dot theta2dot]

tol = 1e-10;
tfinal =5;
%tic
[t,y]=int_doubpend(0,tfinal,tol,y0,beta,g,gam,l);
%toc
```

```
% plot output angles
plot(t,-y(:,1),t,pi-y(:,1)-y(:,2));
%plot(t,y(:,3),t,y(:,4));
%plot(t,y(:,1),t,y(:,2));
%plot(t,y(:,2)-pi+2.*y(:,1));
%plot(t,cos(y(:,1))-cos(pi-y(:,1)-y(:,2)));
```

A.2.5 yderivs_doubpend.m

```
function ydot = yderivs_doubpend(y,betavec);

% All code by Mariano Garcia (garcia@tam.cornell.edu), 1998
% Some documentation of sorts and accompanying figures can
% be found in chaps 2 and 3 of my thesis, which can be downloaded from
% http://tam.cornell.edu/students/garcia/msghomepage.html
% or
% http://www.ccmr.cornell.edu/~ruinalab/
% also read the README.txt file.
% This simulation was used to produce results for the paper
% "The Simplest Walking Model: Stability, Complexity, And Scaling"
% ASME Journal of Biomechanical Engineering Vol 120, No. 2
% pp. 218-288, 1998.

% Note that theta2 is the exterior angle between the
% legs, not the same as phi in the ASME paper above.

% This file contains the derivatives of the simplest walker.

beta = betavec(1);
gam = betavec(2);
g = betavec(3);
l = betavec(4);

cO1 = cos(y(1));
sO1 = sin(y(1));

cO2 = cos(y(2));
sO2 = sin(y(2));

sO1g = sin(y(1)-gam);
sO12g=sin(y(1)+y(2)-gam);

M(1,1)=1+2*beta*(1+cO2);
M(1,2)=beta*(1+cO2);
```

```
M(2,1)=1+cO2;
M(2,2)=1;

V(1)=-beta*sO2*y(4)*(2*y(3)+y(4));
V(2)=sO2*y(3)^2;

G(1)=-g/l*(beta*sO12g+sO1g*(1+beta));
G(2)=-g/l*sO12g;

%-V'-G'

Oddot=M\(-V'-G');

ydot(1)=y(3);
ydot(2)=y(4);
ydot(3)=Oddot(1);
ydot(4)=Oddot(2);

ydot = ydot';
```

Bibliography

[1] M. Garcia, A. Chatterjee, A. Ruina, and M. J. Coleman, "The simplest walking model: Stability, complexity, and scaling," *ASME Journal of Biomechanical Engineering*, vol. 120, pp. 281–288, April 1998.

[2] T. McGeer, "Passive dynamic walking," *The International Journal of Robotics Research*, vol. 9, April 1990. (c) 1990 Massachusetts Institute of Technology.

[3] A. L. Schwab and M. Wisse, "Basin of attraction of the simplest walking model," in *Proceedings of ASME Design Engineering Technical Conferences, Pittsburg*, ASME, September 2001. (c) 2001 by ASME.

[4] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2001.

[5] M. Wisse, A. L. Schwab, R. Q. van der Linde, and F. C. T. van der Helm, "How to keep from falling forward: Elementry swing leg action for passive dynamic walkers," *IEEE Transactions on Robotics*, vol. 21, pp. 393–401, June 2005.

[6] A. D. Kuo, "Energetics of actively powered locomotion using the simplest walking model," *Journal of Biomechanical Engineering*, vol. 124, pp. 113–120, February 2002. (c) 2002 by ASME.

[7] M. W. Gomes, "A derivation of the transisition rule at heelstrike which appears in the paper "the simplest walking model: Stability, complexity, and scaling" by garcia et al.." October 1999.

[8] J. H. Mathews and K. K. Fink, *Numerical Methods Using Matlab*. Prentce Hall, 2004.

[9] J. R. Dormand, *Numerical Methods for Differential Equations*. CRC Press, 1996.

[10] G. Liu, I. M. Y. Mareels, M. G. Pandy, R. Evans, and G. N. Nair, "On the complexity of 2d simplest walking models." 2007.

[11] S. Wiggins, *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer Verlag, 1990.

[12] C. S. Hsu, *Cell-to-Cell Mapping; A Method of Global Analysis for Nonlinear Systems*. Springer, 1987.