2-D Passive Compass Biped Walker

Analysis and Robustness of Stable Gait

Torleif Anstensrud

Master of Science in Engineering Cybernetics
Submission date: August 2013
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

The project is aimed at analyzing the properties of the walker dynamics in a vicinity of its stable gait. In particular, we expect to investigate the characteristics such as speed of contraction to the gait, its region of attraction and sensitivity of the gait to perturbations given either as small changes of the slope or impulse-like external forces of small amplitude and duration along the gait. It is planned that the delivered results will be scalable for analysis of the similar periodic trajectories of hybrid mechanical systems with more than two degrees of freedom. And at the end of the project we plan to present the procedure for computing these characteristics in general.

Assignment given: 10. January 2013
Supervisor: Anton Shiriaev, ITK

Preface

This report documents the work performed in connection with my Master's thesis during the spring semester of 2013. It represents one full semester of work and completes my five year Master of Science program in Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU).

Legged locomotion and humanoid robotics have always been fields that fascinated me growing up, and provided the motivation for choosing the compass-gait biped as my object of research for my thesis. There are still countless unanswered problems in this field of research, and I appreciate having been given the chance to contribute to such an important subject now and in the future.

I would like to thank my supervisor at NTNU, Professor Anton Shiriaev, for providing insight and research material, as well as generously sharing his time throughout the semester. Our frequent meetings resulted in many fruitful discussions and provided much needed inspiration for overcoming obstacles. I would also like to take this opportunity to thank my fellow student colleagues for all the hardships and laughter we have shared, especially Anders for tolerating me through countless laboratory exercises and Kristian for our many peculiar discussions. Lastly I would like to express my gratitude towards my closest family for all the encouragement, motivation and support they have provided, and especially my brother, Ole-Petter, for leading the way all these years and convincing me that five years can pass in an instant.

Trondheim, August 2013

Torleif Anstensrud

Abstract

The compass-gait biped is a deceptively simple walking machine that is often used as a standard benchmark for testing new concepts and methods in legged locomotion. This study will focus on developing a procedure for finding passive gaits of this specific biped by employing the notion of virtual holonomic constraints. First an introduction to the mathematical foundations behind the basic theory is given, then a rigorous treatment of the hybrid dynamics of the biped and the reduction of the system complexity using virtual holonomic constraints are presented. The reduced dynamics of the biped are used to find relations among the gait parameters leading to the formulation of a minimization problem that yields limit cycle solutions of the hybrid system when solved. The stability of the detected limit cycles is assessed using the notion of transverse linearization, and the procedure for deriving an auxiliary linear system for determining orbital stability of the passive gaits is presented. All major results are visualized and the MATLAB and Maple implementation code is enclosed in either the relevant chapters or the appendixes.

Contents

List of Tables

List of Figures

1 Introduction

1.1 Background and Motivation

The study of passive walking devices is a fascinating field that gained wide recognition following the seminal paper by McGeer in 1990 [12]. McGeer introduced the concept of the rimless wheel traveling down an inclined slope to illustrate how rolling can be transformed into walking by a simple metamorphosis.

Wheeled locomotion is one of the most efficient modes of transportation due to the minimal energy required to keep the motion going; an ideal wagon wheel on a downward slope will continue rolling steadily along the incline indefinitely. This rolling motion can be transformed into a walking motion by removing the rim from the wheel, leaving the spokes acting as a set of static legs, as can be seen in Figure 1.1. Unlike the ordinary wheel, the rimless

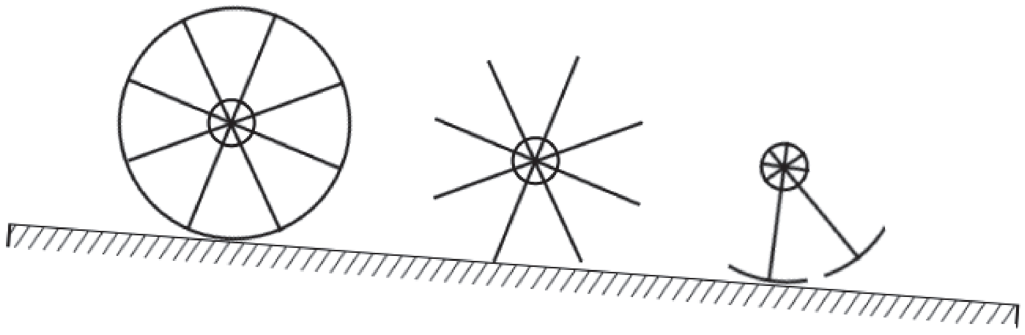

Figure 1.1: Illustration of metamorphosis from wagon wheel to biped walker described by McGeer [12] in 1990.

wheel loses energy with each impact with the ground and the modification is clearly less efficient than the original configuration.

Instead of removing the rim completely, imagine splitting the rim halfway between each spoke, and then remove all but two of the spokes. These remaining spokes act as a pair of legs with semicircular feet that can swing freely by introducing a pin joint between them in the hub, effectively creating a hip joint. This new synthetic wheel is one of the most simple legged robots, that has the property of walking like a biped while rolling like a wheel.

This sort of biped is known as a passive walker due to it's ability to settle into a steady gait on a downward slope without requiring any active control or energy input. Comparing this motion with that of the wheel it becomes clear that this walking gait is highly energy efficient, and it is desirable to study these passive gaits to gain a deeper understanding of how to design efficient walking patterns for other actuated legged robots.

The robot studied in this thesis is a planar two-link walker, commonly known as the compass-gait biped due to the fact that the locomotion produced is analogous to the movement of a pair of compasses or dividers. A simple illustration of the biped is depicted in Figure 1.2. The compass-gait

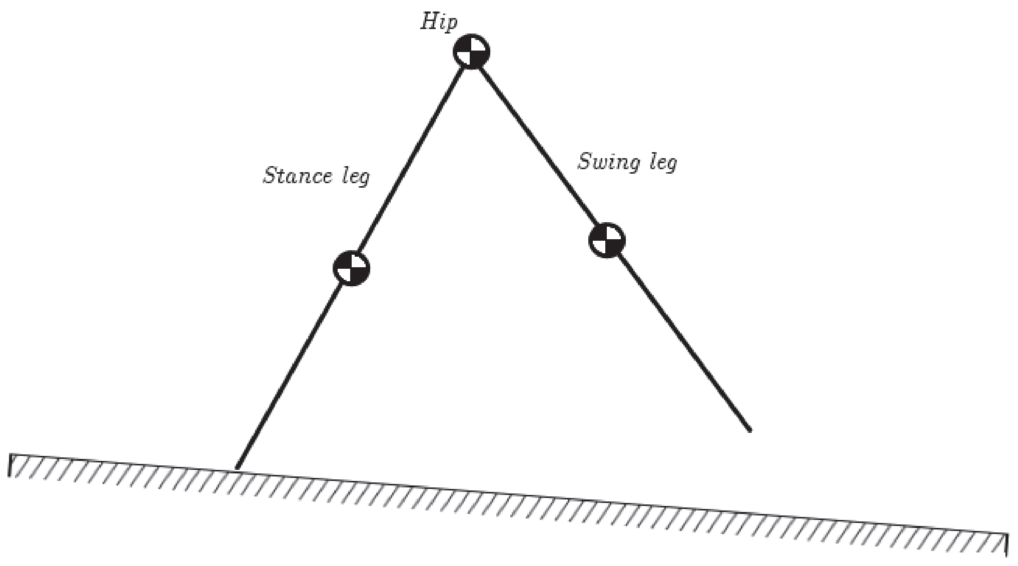

Figure 1.2: Illustration of the compass-gait biped studied in this thesis.

biped resembles the synthetic wheel, but it lacks the semicircular feet that allows for a smooth rolling motion during the gait cycle. This introduces impulsive impact forces that causes discontinuous effects in the dynamics of the robot each time a new leg hits the ground, making analytical arguments for the existence of periodic gaits for this configuration difficult. There was a series of publications following the initial results by McGeer that proposed different strategies for finding and analyzing passive gaits for a variety of walking devices. The important paper [8] by Goswami et al. published in 1998 demonstrated the existence of stable periodic gaits for the compass-gait biped for a variety of slope angles. The paper utilized numerical computational methods that relied on approximations of the full dynamics of the biped, instead of attempting analytical arguments, stating

> *For a general nonlinear system, the analytical demonstration of the existence of a limit cycle, it's local orbital stability, and the analytical procedure to find it still remain a challenge.* [8, p. 1287]

In the last decade there has been extensive research into developing new techniques for analyzing nonlinear systems such as the compass-gait biped. This resulted in a series of publications e.g. [5, 17, 18, 19] on the concept of virtual holonomic constraints[1], which outlined a new approach for finding and characterizing limit cycles of hybrid dynamic systems. The introduction of virtual holonomic constraints enabled the use of analytical arguments for the existence of limit cycles, such as periodic gaits, and the development of tools for finding and characterizing such periodic solutions.

The paper [5] by Freidovich et al. demonstrated how to apply the theory of virtual holonomic constraints to the problem of finding passive gaits for the compass-gait biped. The aim of this thesis is to verify and expand upon the results achieved in [5], and hopefully give a comprehensive introduction to the principle of virtual holonomic constraints and their application to hybrid dynamic systems. In addition, the concept of transverse linearization as a tool for assessing stability of passive gaits will be presented along with obtained results, in the context of the compass-gait biped. Relevant procedures and algorithms will be implemented using MATLAB or Maple, and will be listed either in the relevant chapters or in the appendixes.

1.2 Structure of Thesis Report

The thesis report is organized as follows. Chapter 2 gives a brief introduction to mathematical concepts, theory and notation that are relevant to solving the task at hand. Chapter 3 presents terminology and assumptions made regarding the compass-gait biped, and proceeds with deriving equations of motion using the Euler-Lagrange formalism and a set of discrete update laws that describe the dynamics at impact. These parts are combined into a hybrid dynamics system and a MATLAB script for simulating the complete dynamics of the biped is presented. In Chapter 4 the concept of virtual holonomic constraints are utilized to form relations between the generalized coordinates that are valid along a cycle, allowing for a reduction in system dimension and formulating the reduced dynamics of the system. A general integral for finding specific linear combinations of the reduced dynamics is also proposed. Chapter 5 exploits certain properties of the reduced dynamics of the system to derive a differential equation for the virtual holonomic constraint that allows for a compact formulation of the gait search algorithm. A step-by-step procedure for finding periodic gaits based on this algorithm is then stated along with a MATLAB script implementing the procedure. Chapter 6 introduces the concept of transverse linearization of the system dynamics

[1]The precise definition of these terms will be presented in Chapter 2.

along a periodic trajectory. This linearization is used to define an auxiliary linear system than can be used to determine the stability of the proposed periodic gait for the compass-gait biped. Chapter 7 presents some important results of the gait search procedure developed in Chapter 5 and analyzes the stability of the detected periodic gaits by utilizing the results of transverse linearization from Chapter 6. Important properties of the developed search procedure are discussed in the context of the numerical results. Chapter 8 summarizes the results obtained in the thesis and presents a brief conclusion and final remarks.

2 Mathematical Foundations

This chapter aims to give a brief introduction to some important results from classical mechanics, differential calculus and control theory.

2.1 Euler-Lagrange Equations

The 2-DOF[2] planar biped is a mechanical system that, despite its simple construction, exhibits highly nonlinear dynamics. The derivation of a mathematical model for the motion the biped experiences during its gait cycle is therefore relevant for studying the physical properties of the system. A general procedure for deriving a set of differential equations that describe the time evolution of a mechanical system subject to holonomic constraints is presented in [20], and will be summarized here.

Holonomic constraints

Consider a system consisting of k particles free to move unconstrained relative to each other. Deriving a set of differential equation that describe the motion of these particles is trivial using Newton's 3rd Law for each particle i as

$$m_i \ddot{r}_i = F_{ext} \tag{2.1}$$

where r_i is the position vector of the particle and F_{ext} is the sum of the external forces applied to it. In the case of a kinematic chain, such as a robot manipulator, each of these particles are restricted in their movements relative to each other due to physical links present in the system. These restrictions can be modeled as massless rods pairwise connecting the particles to form a rigid chain. Using the position notation from (2.1) such a rod between particle i and $i+1$ can be mathematical represented as a constraint on the form

$$(r_i - r_{i+1})^T (r_i - r_{i+1}) = l^2 \tag{2.2}$$

where l is the length of the rod. The restriction (2.2) can be rewritten as $h_i(r_i, r_{i+1}) = 0$. The notation can easily be expanded to constraints on the full k coordinates of the system as

$$h_i(r_1, \ldots, r_k) = 0, \qquad i = 1, \ldots, m \tag{2.3}$$

[2]The degrees of freedom denote how many independent parameters must be defined to uniquely identify the configuration of the system. In the case of the biped, with assumptions discussed later, only the angles of the two legs are needed to fully specify its position and orientation.

where m is the number of constraints present in the system. Equality constraints on the form (2.3) are called holonomic constraints and motivates the introduction of generalized coordinates when modeling the system.

Generalized coordinates

If there exists an n-dimensional column vector $q(t) = [q_1(t), \dots, q_n(t)]^T$ such that the position vectors of the k particles subjected to the constraints (2.3) can be expressed as

$$r_i = r_i(q_1(t), \dots, q_n(t)), \qquad i = 1, \dots, k \tag{2.4}$$

where $q_1(t), \dots, q_n(t)$ are all independent, then $q(t)$ are called the generalized coordinates of the system.

The holonomic constraints restrict the motion of the particles such that the system has m fewer degrees of freedom relative to the unconstrained system. This reduction in dimension is maintained by constraint forces[3] that ensure the equality constraints are satisfied. The dynamics outlined in (2.1) must be modified to include these new internal forces by introducing the added term F_{int} on the right-hand side,

$$m_i \ddot{r}_i = F_{ext} + F_{int} \tag{2.5}$$

The internal forces F_{int} greatly increase the complexity of computing the dynamics of the system, and finding a representation of the coordinates r_i such that these forces vanish is desired.

In fact, by applying the Principle of Virtual Work and d'Alembert's Principle to (2.3) - (2.5) it can be shown [3, 20] that explicit computation of the constraint forces F_{int} can be avoided if the position vectors are represented using generalized coordinates. Finding a representation of the system in n independent generalized coordinates $q(t)$ is therefore the first step in deriving the complete dynamics of the mechanical system.

The Lagrangian and equations of motion

In a kinematic chain, such as a biped robot, each link of the mechanical system can be approximated as a particle with mass m_i located at the center of mass of the link. Each particle is then connected using holonomic constraints on the form $h_i(q_1, \dots, q_n) = 0$ where $q_1, \dots, q_n$ are generalized coordinates

[3]In the case of particles linked with rods, the constraint forces are exerted by the rod on the particles directed along $r_i - r_{i+1}$. Without these forces the rod could be freely compressed or stretched by the motion of the particles.

measuring the orientation of each link. One possible choice of generalized coordinates is the absolute angle of each link measured from an axis defined in the inertial frame.

Once the appropriate generalized coordinates have been assigned to the system, the holonomic constraints on the form (2.3) become trivial. The kinetic $\mathcal{K}(q,\dot{q})$ and potential $\mathcal{P}(q)$ energy of the robot can then easily be derived using the expressions for the generalized coordinates. The Lagrangian of the mechanical system is then formed as

$$\mathcal{L}(q,\dot{q}) = \mathcal{K}(q,\dot{q}) - \mathcal{P}(q) \tag{2.6}$$

where $\mathcal{K}(q,\dot{q}) = \frac{1}{2}\dot{q}^T M(q)\dot{q}$. Here $q \in \mathbb{R}^n$ and $\dot{q} \in \mathbb{R}^n$ are the vectors of generalized coordinates and velocities and $M(q) \in \mathbb{R}^{n\times n}$ is the inertia matrix of the system. The Lagrangian can be used to obtain a set of differential equations that describe the time evolution of the system by substituting it into the expression

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial \mathcal{L}(q,\dot{q})}{\partial \dot{q}}\right] - \frac{\partial \mathcal{L}(q,\dot{q})}{\partial q} = B(q)u \tag{2.7}$$

where $B(q) \in \mathbb{R}^{n\times m}$ is a full rank input matrix and $u \in \mathbb{R}^m$ is a vector of independent control inputs. Assuming that the matrix $M(q)$ is symmetric and positive definite for each $q \in \mathbb{R}^n$ and that $\mathcal{P} = \mathcal{P}(q)$ is independent of $\dot{q}$, (2.7) can be written in a compact matrix form (see [20]) known as the robot equations of motion

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)u \tag{2.8}$$

where $C(q,\dot{q}) \in \mathbb{R}^{n\times n}$ is a matrix of centrifugal and Coriolis terms and $G(q) \in \mathbb{R}^n$ is the gravity vector. Using the previously stated assumptions about the Lagrangian, the elements of the matrices on the left-hand side of (2.8) can be computed as

$$m_{kj}(q) = \frac{\mathrm{d}}{\mathrm{d}\dot{q}_k}\left[\frac{\mathrm{d}}{\mathrm{d}\dot{q}_j}\mathcal{K}(q,\dot{q})\right] \tag{2.9}$$

$$c_{kj}(q,\dot{q}) = \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} + \frac{\partial m_{ij}}{\partial q_k}\right)\dot{q}_i \tag{2.10}$$

$$g_k(q) = \frac{\partial \mathcal{P}(q)}{\partial q_k} \tag{2.11}$$

where the subscript k is the row and j is the column of the appropriate matrix, and q_i, q_j and q_k is the appropriate elements of the generalized coordinate vector q.

When (2.8) is used for simulating the dynamics of the mechanical system the equations of motion are usually reformulated as a state space model with the $2n$ - dimensional state vector $x = [q, \dot{q}]^T$. The system can then be stated as system of $2n$ first order differential equations

$$\begin{aligned} \dot{x} &= f(x) + g(x)u \\ y &= h(x) \end{aligned} \tag{2.12}$$

where y is the measured output.

2.2 Hybrid Dynamic Systems

The Euler-Lagrange equations of motion (2.8) describe the unrestricted continuous dynamics of the physical system, and do not take into consideration constraints imposed by the environment where the motion occurs. In the case of a walking biped, one such restriction is the ground impact that occurs when the foot of the robot impacts with the floor during a step. During this phase of the walking cycle an exchange of stance and swing leg should occur, and the generalized coordinates $q(t)$ should experience a transformation to reflect the new state of the system.

These dynamics are not continuous in nature and can not be modeled using only Euler-Lagrange equations. A complete mathematical model of the walking biped must therefore involve a set of discrete dynamics that describe the instantaneous change in state during the impact. A system that exhibits behavior of both continuous and discrete dynamics is called a hybrid dynamic system and can be described by three important properties:

1. One or more ordinary differential equations of the form $\dot{x} = f(x)+g(x)u$ that describe the unrestricted continuous dynamics of the mechanical system.

2. One or more hypersurfaces $\mathcal{S}(x)$ that define boundary conditions for the continuous dynamics such that the configuration $x^- \in \Gamma^-$ indicate an impact on the surface $\mathcal{S}$.

3. One or more discrete mappings $F : \Gamma^- \to \Gamma^+$ that maps the solution of the continuous dynamics at impact $x^- \in \Gamma^-$ to the updated coordinates $x^+ \in \Gamma^+$ after the impact on $\mathcal{S}$. The transformation F is an instantaneous discrete transition to new coordinates, which initializes the next differential equation and the motion continues until the next impact on $\mathcal{S}$.

The superscripts $-, +$ indicate the configuration of the system just before and just after impact, respectively, and can be mathematically expressed as

$$\begin{aligned} x^- &= \lim_{t \to T-} x(t) \\ x^+ &= \lim_{t \to T+} x(t) \end{aligned} \tag{2.13}$$

where $t = T$ is the time of impact.

The continuous dynamics of the 2-DOF biped are identical before and after impact due to the symmetric construction of the robot. The hybrid dynamic system therefore only needs one differential equation that describes the time evolution of the generalized coordinates during motion, which can be expressed as Euler-Lagrange equations on the form (2.8). The only boundary condition applicable to these dynamics is the impact that occurs during heel strike with the floor, at which point the configuration of the robot should experience an instantaneous mapping to updated generalized coordinates and velocities. Due to the static nature of the inclined floor, the hypersurface can be uniquely defined as $\mathcal{S} = \mathcal{S}(q)$ independent of the generalized velocities $\dot{q}$. The full hybrid dynamics of the biped can then be stated as (see [13])

$$\begin{aligned} &M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B(q)u, \qquad && q \notin \mathcal{S} \\ &F(q^-, \dot{q}^-) : \Gamma^- \to \Gamma^+, && q^- \in \mathcal{S} \end{aligned} \tag{2.14}$$

where Γ^- and Γ^+ indicate the full configuration of the biped before and after impact, respectively. Algorithm 1 gives a brief outline of the procedure for simulating the hybrid dynamic system defined in (2.14).

Passive walkers

The 2-DOF planar biped studied in this thesis is a hybrid dynamic system on the form (2.14) with the exception that the motion of the robot is unaffected by a control input u. The robot is placed on a shallow slope allowing for gravity to be the only driving force of motion, which means that the system is not controllable in the sense of control theory. The hybrid model formulation that will be utilized in the rest of this chapter will therefore be changed to

$$\begin{aligned} &M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = 0, \qquad && q \notin \mathcal{S} \\ &F(q^-, \dot{q}^-) : \Gamma^- \to \Gamma^+, && q^- \in \mathcal{S} \end{aligned} \tag{2.15}$$

where all parameters are as described in (2.14). Such a system is commonly described as a passive walker and is known to possess walking gaits that are highly energy-efficient due to the lack of a control torque (see [12]). These walking gaits can be formulated as periodic solutions of the hybrid system (2.15) and are of particular interest in this thesis.

Algorithm 1: Simulating the hybrid dynamic system defined in (2.14)

```
IN is a vector of user-defined inputs.
OUT is a matrix containing the simulation data returned by the
routine

Set initial conditions: $q_0 \leftarrow IN(1), \quad \dot{q}_0 \leftarrow IN(2)$
Set predefined input sequence: $u \leftarrow IN(3)$

while simulation not finished do
    Initialize: $q_0 \leftarrow q^+, \quad \dot{q}_0 \leftarrow \dot{q}^+$
    repeat
        Integrate: $[q, \dot{q}] \leftarrow M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)u$
    until $q \in \mathcal{S}$;

    Compute update: $\Gamma^+ \ni [q^+, \dot{q}^+]^T \leftarrow F(q^-, \dot{q}^-), \qquad [q^-, \dot{q}^-]^T \in \Gamma^-$;
end

$OUT \leftarrow q, \dot{q} \quad \forall t$
```

2.3 Periodic Cycles

Periodic cycles is an important class of solutions for general dynamics systems of the form $\dot{x} = f(x)$. These cycles define oscillations of the solution around an equilibrium point and are also referred to as periodic solutions.

Definition

The most general definition of a nontrivial periodic solution is given in [10] as

$$x(t) = x(t+T), \qquad \forall t \geq 0 \tag{2.16}$$

where $T > 0$ define the period of oscillation. This solution is called a non-trivial periodic cycle to exclude constant solutions of the form $x(t) = C$, which corresponds to equilibrium points of the dynamic system. Although such a trivial solution can be viewed as an oscillation with period $T = 0$, these equilibrium points are not of interest when studying dynamics motions and will not be explored further in this text.

Periodic solutions are recognized in phase portraits as closed trajectories of the system, which are commonly referred to as periodic orbits or closed orbits. This behavior is easily observed in linear systems on the form

$$\dot{x} = Ax \tag{2.17}$$

where $A \in \mathbb{R}^{2\times 2}$ is a system matrix with complex eigenvalues $\pm j\beta$. The solution of this system after transformation to real Jordan form is given by

$$z_1(t) = r_0 \cos(\beta t + \theta_0), \qquad z_2(t) = r_0 \sin(\beta t + \theta_0) \tag{2.18}$$

where z_1, z_2 are the respective axis in the phase plane and

$$r_0 = \sqrt{z_1^2(0) + z_2^2(0)}, \qquad \theta_0 = \tan^{-1}\left[\frac{z_2(0)}{z_1(0)}\right]$$

The expressions in (2.18) are easily recognized as the equations for a circle with radius r_0, meaning that the phase portrait of the system will consist of infinitely many closed orbits arranged as concentric circles in the plane. There are two important properties that this solution illustrates. Firstly, the amplitude of oscillation is determined uniquely by the radius of the circle $r_0 = r_0(z_1(0), z_2(0))$ which is dependent on the initial conditions of the system. Secondly, choosing the initial conditions $[z_1(0), z_2(0)]^T = [0, 0]^T$ gives $r_0 = 0$ and reduces the periodic solution (2.18) to the trivial case of an equilibrium point at the origin. Such an equilibrium point is called a center and all trajectories around this point are closed orbits with constant amplitude. The phase portrait of a linear system on the form (2.17) is shown in Figure 2.1, initialized with three different initial conditions.

Limit cycles

Contrary to the periodic solutions of linear systems (2.18), there exist nonlinear systems that possess periodic solutions which amplitude are independent of initial conditions. These systems may exhibit a single isolated periodic orbit in their phase portrait, instead of the continuum of closed orbits that characterizes a linear oscillator. Such an isolated periodic orbit is known as a limit cycle, and has some interesting properties. To illustrate such a limit cycle, consider the nonlinear Van der Pol equation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + \left(1 - x_1^2\right) x_2 \end{aligned} \tag{2.19}$$

The phase portrait of (2.19) is shown in Figure 2.2 for three different initial conditions. The phase portrait shows that there is a closed trajectory that attracts all other trajectories starting off the orbit. This is a property of

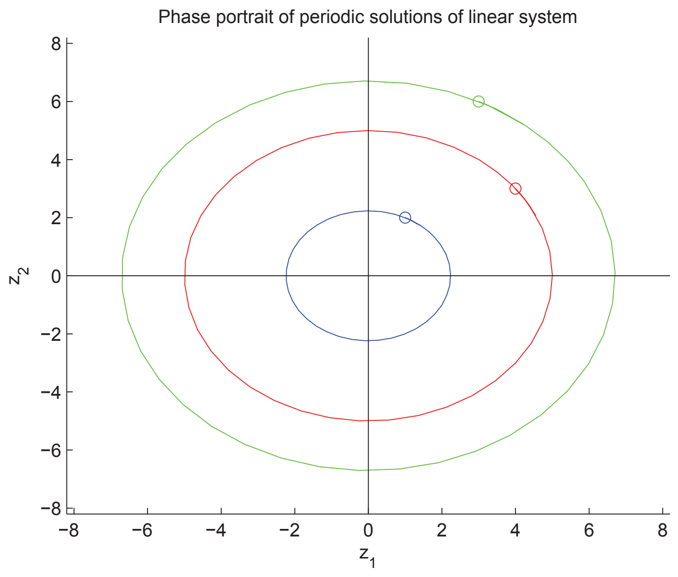

Figure 2.1: Phase portrait of a linear system on the form (2.17)*, initial conditions are depicted as circles on the closed orbits.*

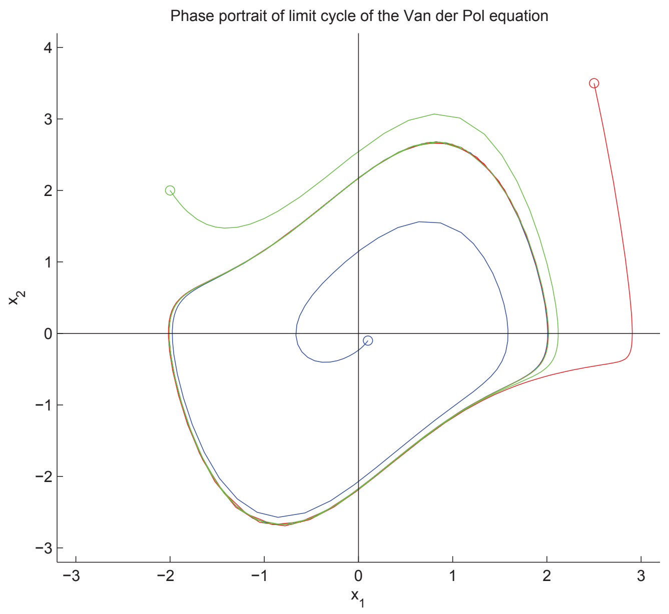

Figure 2.2: Phase portrait of the Van der Pol equation with stable limit cycle, initial conditions are depicted as circles on the three trajectories.

stable limit cycles, all trajectories in the vicinty of the limit cycle ultimately tend toward the closed orbit as $t \to \infty$. The amplitude at steady state is in other words completely independent of the initial conditions.

Finding limit cycles analytically in general is a very difficult problem, although there are ways of proving existence of periodic orbits. The Poincaré-Bendixson Criterion (see [10]) can assure the existence of a periodic orbit in a closed bounded subset of the plane M, but not its uniqueness. Meaning that the nature and quantity of limit cycles in M can not be determined, but the criterion assures that there are at least one periodic orbit in M.

Gait cycles

Limit cycles are especially interesting in the study of gaits of walking robots. The dynamics (2.15) of these walking machines are usually highly nonlinear and possess multiple degrees of freedom. Finding walking gaits for such mechanical systems is equivalent to finding limit cycles among the generalized coordinates and velocities such that the configuration of the robot is periodic with period equal to the time between steps. Using the notation introduced for hybrid dynamical systems, this can be stated as

$$\Gamma_i^+ \ni [q_i, \dot{q}_i]^T = [q_{i+1}, \dot{q}_{i+1}]^T \in \Gamma_{i+1}^+ \tag{2.20}$$

where the subscript indicates step i and Γ^+ is the hypersurface containing the updated state vector after impact.

In addition to finding a limit cycle such that (2.20) is satisfied, the limit cycle should ideally be stable meaning that small perturbations from the periodic orbit should not cause the robot to topple over. If such a stable limit cycle can be found in the dynamics, the robot can be started with initial conditions away from the periodic orbit and will gradually converge to the stable gait. This property is very important for physical testing since small uncertainties related to measuring and manufacturing of the walking machine will always result in perturbations of the desired orbit.

2.4 Stability of Periodic Orbits

Determining the stability of dynamic systems is one of the main problems in control theory, and the literature on the subject is vast (e.g. [2, 10]). Generally the stability of a system describes how solution trajectories behave as $t \to \infty$, but there exist numerous definitions detailing different types of stability. A comprehensive treatment of these concepts are beyond the scope of this text, but some important results relevant to this thesis will be discussed.

Stability of linear discrete-time systems

A discrete linear system on the form

$$x[k+1] = Ax[k] \tag{2.21}$$

excited by non-zero initial conditions x_0 is said to be marginally stable or stable in the sense of Lyapunov if every finite initial state x_0 excites a bounded response. The system (2.21) is asymptotically stable if every finite initial state x_0 excites a bounded response which, in addition, approaches $x = 0$ as $k \to \infty$.

Theorem 1. *[2, Theorem 5.D4]*

1. *The equation* (2.21) *is marginally stable if and only if all eigenvalues of A have magnitudes less than or equal to 1 and those equal to 1 are simple roots of the minimal polynomial of A.*

2. *The equation*(2.21) *is asymptotically stable if and only if all eigenvalues of A have magnitudes less than 1.*

Proof. The proof is given in detail in [2] and will not be repeated here. □

Similar definitions can be stated for continuous-time linear systems and are readily available in [2]. These results are not needed in this thesis and will not be expanded upon here.

Orbital stability

The notion of stability for equilibrium points can be extended to periodic solutions, with the use of coordinate substitution and linearization. Suppose that $x_\star(t)$ is a nontrivial periodic solution to the nonlinear system

$$\dot{x} = f(x) \tag{2.22}$$

and the behavior of other solutions $x(t)$ in neighbor of the periodic orbit is of interest. The stability of $x_\star(t)$ can be investigated by defining

$$y = x - x_\star(t) \tag{2.23}$$

so that the origin $y = 0$ becomes an equilibrium point for the nonlinear system

$$\begin{aligned} \dot{y} &= \dot{x} - \dot{x}_\star(t) \\ \dot{y} &= f(y - x_\star(t)) - f(x_\star(t)) \\ \dot{y} &= F(y, x_\star(t)) \end{aligned} \tag{2.24}$$

The behavior of this equilibrium can be investigated by checking the eigenvalues of the matrix $A(t)$ of the linearization

$$\dot{z} = A(t)z, \qquad A(t) = \left(\frac{\partial F(y, x_\star(t))}{\partial y} \right) \bigg|_{y=0} \tag{2.25}$$

If the equilibirum $y = 0$ is stable, the periodic solution $x_\star(t)$ is stable since the behavior of solutions near $x_\star(t)$ is equivalent to the behavior of solutions of (2.24) near $y = 0$. The system (2.24) can never be asymptotically stable in the sense of Lyapunov since $z = x_\star(t)$ is a non-vanishing solution [19].

By extending the notion of stability in the sense of Lyapunov from stability of an equilibrium to stability of an invariant set, it is possible to classify the stability properties of periodic orbits. Let $M \subset D$ be a closed invariant set of (2.22) where f is a continuously differentiable map from domain $D \subset \mathbb{R}^n$ into $\mathbb{R}^n$. Define an ϵ-neighborhood of M by

$$U_\epsilon = \{x \in \mathbb{R}^n \,|\, \mathrm{dist}(x, M) < \epsilon\} \tag{2.26}$$

where $\mathrm{dist}(x, M)$ is the minimum distance from x to a point in M; that is

$$\mathrm{dist}(x, M) = \inf_{y \in M} \|x - y\| \tag{2.27}$$

Definition 1. *[10, Definition 8.1] The closed invariant set M of* (2.22) *is*

1. *stable, if for each $\epsilon > 0$, there is $\delta > 0$ such that*

$$x(0) \in U_\delta \Rightarrow x(t) \in U_\epsilon, \quad \forall t \geq 0 \tag{2.28}$$

2. *asymptotically stable if it is stable and δ can be chosen such that*

$$x(0) \in U_\delta \Rightarrow \lim_{t \to \infty} \mathrm{dist}(x(t), M) = 0 \tag{2.29}$$

A periodic solution $x_\star(t)$ of (2.22) can be regarded as an invariant set M. The closed orbit can then be described as the invariant set

$$\gamma = \{x \in \mathrm{R}^n \,|\, x = x_\star(t), \quad 0 \leq t \leq T\} \tag{2.30}$$

where $t = T$ is the period of the solution. Orbital stability of the periodic solution can then be defined as

Definition 2. *[10, Definition 8.2] A nontrivial periodic solution $x_\star(t)$ of* (2.22) *is*

1. *orbitally stable if the closed orbit γ generated by $x_\star(t)$ is stable.*

2. *asymptotically orbitally stable if the closed orbit γ generated by $x_\star(t)$ is asymptotically stable.*

Asymptotically stable periodic orbits is commonly referred to as stable limit cycles, which are important in the study of walking gaits.

Poincaré first return map

A standard tool for determining the stability properties, such as orbital stability, of nontrivial periodic orbits is the Poincaré first return map (see [13, 19]). The initial step is to construct a $(n-1)$-dimensional hypersurface S, known as a Poincaré section, which is transversal to the flow of the periodic orbit γ in state space at all points. Such a hypersurface is illustrated in Figure 2.3.

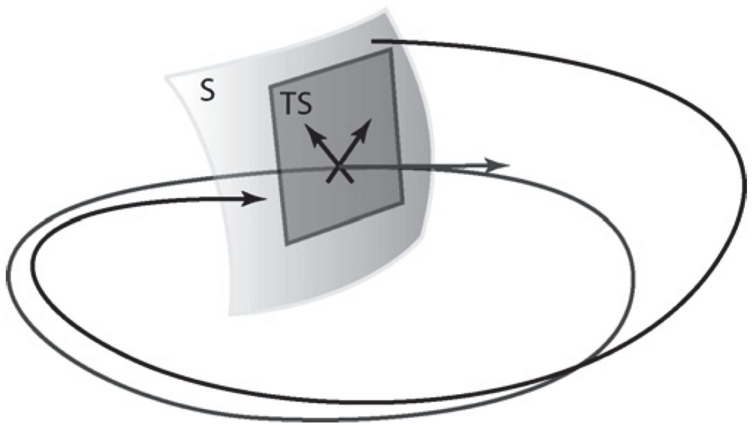

Figure 2.3: Poincaré surface S and linearization TS with periodic orbit γ (grey) and another solution converging to the orbit (black). Illustration © 2008 Elsevier. Reproduced, with permission, from Annual Reviews in Control, vol. 32, no. 2, A. Shiriaev, L. Freidovich, and I. Manchester, "Can we make a robot ballerina perform a pirouette? Orbital stabilization of periodic motions of underactuated mechanical systems," pp. 200 211, 2008.

The behavior of the system is studied on a subset $S_0 \subset S$ that must contain the intersection with the periodic trajectory γ. Let the map $P : S_0 \to S$ be defined by the first hit rule, such that P maps the initial solution of γ belonging to S_0 into the points where the solutions intersects S again for the first time. The corresponding time-discrete system

$$x_\perp[k+1] = P\left(x_\perp[k]\right), \quad x_\perp \in \mathbb{R}^{n-1} \tag{2.31}$$

is known as the Poincaré first return map. The map has a fixed point $P(x_\perp^*) = x_\perp^*$ defined by the periodic orbit γ. If the map is contracting, the periodic solution $x_\star(t)$ is asymptotically stable and the resulting periodic orbit is a stable limit cycle. Contraction can be verified by linearizing the map P around the periodic solution γ as

$$\delta x_\perp[k+1] = \left.\frac{\mathrm{d}P}{\mathrm{d}x_\perp}\right|_{x_\perp = x_\perp^*} \cdot \delta x_\perp[k], \qquad \delta x_\perp = x_\perp - x_\perp^* \tag{2.32}$$

forming the mapping $dP : TS \to TS$ which acts on the plane TS tangent to S at the point of intersection between γ and S. The periodic orbit γ is locally stable if the eigenvalues of dP is strictly inside of the unit circle, as in the case of a linear time-discrete system. The rate of convergence for the solution can be estimated by the absolute value of the eigenvalue of dP closest to the unit circle.

2.5 First Integrals of Dynamic Systems

Definition

Consider an nth order system of ordinary differential equations

$$\dot{x}_i = f_i(t, x_1, x_2, \ldots, x_n), \qquad i = 1, 2, \ldots n \tag{2.33}$$

where f_i are continuous differentiable functions defined on a domain $\mathcal{D} \in \mathbb{R}^{n+1}$. The system can be written in more compact form as

$$\dot{X} = f(t, X) \quad \text{where} \quad X = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \quad f = \begin{bmatrix} f_1(t, X) \\ f_2(t, X) \\ \vdots \\ f_n(t, X) \end{bmatrix} \tag{2.34}$$

Define another differentiable function $F(t, X)$ on the same domain $\mathcal{D}$ and calculate the Lie derivative (see [10]) of this function along the trajectories of f

$$L_f F(t, X) = \frac{\partial F(t, X)}{\partial t} + \sum_{i+1}^{n} \frac{\partial F(t, X)}{\partial x_i} f_i(t, X) = \frac{\mathrm{d}F(t, X)}{\mathrm{d}t} \tag{2.35}$$

If the function $F(t, X)$ is non-constant and satisfies the relation

$$L_f F(t, X) = 0, \quad \forall X \in \mathcal{D} \tag{2.36}$$

then $F(t, X)$ is called a first integral of the system (2.34). In other words $F(t, X)$ is a quantity that keeps its value along all solution trajectories of the system.

Reducing order of dynamics

First integrals are one of the main methods for finding solutions of nonlinear dynamics, since a conserved quantity on the form $F(t, X) = C$ define an implicit relation between the degrees of freedom of the system. Through a nontrivial coordinate transformation, the first integral can be used to substitute out one of the degrees of freedom and thus reduce the dimension of the system.

$$F(t, x_1, x_2, \ldots, x_n) = C \quad \rightarrow \quad x_1 = \phi(t, x_2, x_3, \ldots, x_n, C) \tag{2.37}$$

This transformation can then be substituted into the remaining f_i of the dynamic system and removing the dependence on x_1

$$f_i(t, x_1, x_2, \ldots, x_n) = f_i(t, \phi, x_2, \ldots, x_n) = f_i(t, x_2, x_3, \ldots, x_n) \tag{2.38}$$

The order of the system after substituting the first integral is then $(n-1)$, a reduction in order of 1.

The first integral (2.37) is not unique, there may exist several independent first integrals for a given dynamical system. If $F_1(t, X), \ldots, F_n(t, X)$ are functionally-independent first integrals, then any other first integral $F(t, X)$ can be found as

$$F(t, X) = \Phi(F_1(t, X), \ldots, F_n(t, X)) \tag{2.39}$$

where $\Phi(\ldots)$ is a differentiable function defined on the domain $\mathcal{D}$.

Proof.
Let $F_1(t, X), \ldots, F_n(t, X)$ be independent first integrals of the system (2.34) defined on domain $\mathcal{D} \in \mathbb{R}^{n+1}$ and $\Phi(F_1(t, X), \ldots, F_n(t, X))$ a differentiable function defined on the same domain then

$$L_f F(t, X) = L_f \Phi(F_1, \ldots, F_n) = \sum_{i+1}^{n} \frac{\partial \Phi}{\partial F_i} \frac{\mathrm{d}F_i}{\mathrm{d}t} \tag{2.40}$$

since $F_i(t, X)$ is a first integral

$$\frac{\mathrm{d}F_i}{\mathrm{d}t} = 0 \quad \rightarrow \quad \sum_{i+1}^{n} \frac{\partial \Phi}{\partial F_i} \frac{\mathrm{d}F_i}{\mathrm{d}t} = \quad \rightarrow \quad L_f F(t, X) = 0 \tag{2.41}$$

$F(t, X)$ is therefore a first integral of the system (2.34). □

Finding k independent first integrals for the n dimensional system (2.34) enables reduction of the order of the system down to $(n-k)$. If $k = n$ a general integral of the system can be obtained without integration of the nonlinear dynamics.

In the case of mechanical systems such as the biped walker, an intuitive choice of first integral is the total energy $\mathcal{E} = \mathcal{K} + \mathcal{P}$ of the robot. Since energy is a conserved quantity for a physical system, $\mathcal{E}$ should keep its constant value along the solution trajectories of (2.15). The equations of motion for the biped are formed using the Euler-Lagrange equations, and the Lagrangian (2.6) can therefore be used to define a first integral for the biped as

$$\begin{aligned}\mathcal{E} &= \sum_{i=1}^{n} \dot{q}_i \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \mathcal{L} \\ \mathcal{E} &= 2\mathcal{K} - (\mathcal{K} - \mathcal{P}) \\ \mathcal{E} &= \mathcal{K} + \mathcal{P}\end{aligned} \tag{2.42}$$

Unfortunately, the reduction of order for a mechanical system using a conserved quantity is in general an unsolved problem as stated in [5].

2.6 Reduced Dynamics

First integrals are one important method for reducing the order of general dynamic systems, but not all systems are fully integrable and possess a first integral. In this section another method for reducing the dimensions of dynamic systems not dependent on finding first integrals is introduced. The following procedure is documented in full or partially in [13, 16, 17, 18] and interested readers are refered to these publications for a comprehensive treatment of the subject.

Virtual holonomic constraints

Any nontrivial periodic trajectory $q_\star(t)$ of the system (2.15) is a solution of the differential equations in (2.15). This means that the solution can be defined on a finite interval of time as $q_\star(t) = q_\star(t+T), \forall t$ where T is the period of the cycle. The time evolution of the generalized coordinates is not the only way of describing this periodic trajectory, the motion can actually be fully described using any scalar variable $\theta_\star$ that uniquely defines a particular point on the trajectory during the specified time interval.

The periodic cycle can then be re-parametrized in terms of the new measure of progress θ as

$$q_{1\star} = \phi_1(\theta), \ldots, q_{n\star} = \phi_n(\theta), \quad \theta = \theta_\star(t), \quad \forall t \in [0, T] \tag{2.43}$$

where the functions $\phi_1(\theta), \ldots, \phi_n(\theta)$ are referred to as virtual holonomic constraints. Similiarly to holonomic constraints previously discussed, virtual holonomic constraints serve to reduce the dimension of the state space by imposing certain relations between the generalized coordinates. However, unlike physical constraints such as steel rods linking particles together, virtual holonomic constraints are relations between coordinates kept invariant by a control law or the product of certain feasible motions of the system.

The shape of the functions $\phi_1(\theta), \ldots, \phi_n(\theta)$ depend on the choice of the scalar variable θ and how it parameterizes points along the trajectory of the cycle. To ensure certain properties of the constraint functions, such as being invertible, it is desired that $\theta(t)$ is a monotonic function of time. One obvious choice (see [17]) for θ that fulfills this criteria is

$$\mathcal{O}\left(q_\star\right) = \left\{[q, \dot{q}] \in \mathbb{R}^{2n} : q = q_\star(t), \dot{q} = \dot{q}_\star(t), t \in [0, T]\right\} \tag{2.44}$$

which describe the distance of travel along the periodic orbit. Parameterizing the trajectory in terms of arc length is often a complicated task, but for many systems simpler choices of θ exist.

The virtual holonmic constraints given in (2.43) describes the n-dimensional system using $(n+1)$ parameters, meaning that one of the generalized coordinates may be chosen as the scalar variable θ. In the case of the planar biped, the time evolution of the stance foot[4] is described by the generalized coordinate $q_1(t)$ which is monotonic throughout the motion. A possible choice of virtual constraints for the general n-dimensional system is therefore

$$q_{2\star} = \phi_2(\theta), \ldots, q_{n\star} = \phi_n(\theta), \quad q_{1\star} = \theta(t), \quad \forall t \in [0, T] \tag{2.45}$$

where $\phi_2(\theta), \ldots, \phi_n(\theta)$ are scalar smooth functions that have continuous first and second derivatives.

$\alpha\beta\gamma$ - equations

The expressions for the scalar functions $\theta_\star(\cdot)$ and $\phi_1(\cdot), \ldots, \phi_n(\cdot)$ are unknown quantities, but it is possible to derive equations with respect to these variables using the equations of motions defined in (2.15). An important property of the equations of motion

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = 0 \tag{2.46}$$

is that the matrix $C(q, \dot{q})$ depends linearly on $\dot{q}$, as seen in (2.10). The virtual holonomic constraints of the system and their derivatives can be described in vector notation as

$$q(t) = \begin{bmatrix} \phi_1(\theta) \\ \phi_2(\theta) \\ \vdots \\ \phi_n(\theta) \end{bmatrix} = \Phi(\theta), \quad \dot{q}(t) = \Phi'(\theta)\dot{\theta}, \quad \ddot{q}(t) = \Phi''(\theta)\dot{\theta}^2 + \Phi'(\theta)\ddot{\theta} \tag{2.47}$$

[4]The stance foot is the foot of the robot that stays in contact with the ground during a step.

where $\Phi'(\theta) = \frac{\mathrm{d}\Phi(\theta)}{\mathrm{d}\theta}$ and the time dependence of $\theta = \theta(t)$ has been skipped for readability. Substituting for these relations in (2.46) yields the expression

$$M(\Phi(\theta))\left[\Phi''(\theta)\dot{\theta}^2 + \Phi'(\theta)\ddot{\theta}\right] + C\left(\Phi(\theta), \Phi'(\theta)\dot{\theta}\right)\Phi'(\theta)\dot{\theta} + G(\Phi(\theta)) = 0 \quad (2.48)$$

which can be rewritten on the form

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0 \quad (2.49)$$

where

$$\begin{aligned} \alpha(\theta) &= M(\Phi(\theta))\Phi'(\theta) \\ \beta(\theta) &= M(\Phi(\theta))\Phi''(\theta) + C\left(\Phi(\theta), \Phi'(\theta)\dot{\theta}\right)\Phi'(\theta) \\ \gamma(\theta) &= G(\Phi(\theta)) \end{aligned} \quad (2.50)$$

The equations described in (2.49) is referred to as the $\alpha\beta\gamma$-equations or the reduced dynamics of the system. An interesting property of the reduced system dynamics is that linear combinations of arbitrarily chosen $\alpha\beta\gamma$-equations are on the form

$$\begin{aligned} (\mu_1(\theta)\alpha_1(\theta) + \mu_2(\theta)\alpha_2(\theta))\,\ddot{\theta} + (\mu_1(\theta)\beta_1(\theta) + \mu_2(\theta)\beta_2(\theta))\,\dot{\theta}^2 \\ + (\mu_1(\theta)\gamma_1(\theta) + \mu_2(\theta)\gamma_2(\theta)) = 0 \end{aligned} \quad (2.51)$$

where $\mu_1(\theta), \mu_2(\theta)$ are θ-dependent weights and $\alpha_i(\cdot), \beta_i(\cdot), \gamma_i(\cdot)$, $i \in [1, 2]$ are coefficients of two arbitrarily chosen equations on the form (2.49). It is easily verifiable that (2.51) is another expression for the reduced dynamics of the original system (2.46).

Due to the lack of control input to the system, each of the n equations can be integrated and used to describe the full evolution of $\theta, \dot{\theta}$ and $\ddot{\theta}$ assuming the relations (2.47) are kept invariant. This means that the reduced dynamics are only valid if initial conditions are chosen to satisfy the constraints $q(0) = \Phi(\theta(0))$ and $\dot{q}(0) = \Phi'(\theta(0))\dot{\theta}(0)$. When searching for periodic motions of the system it is assumed that these synchronization functions between the generalized coordinates do in fact exist such that the virtual holonomic constraints are kept invariant along the target trajectory.

The main advantage of utilizing virtual holonomic constraints is that an n-dimensional system can be reduced to a scalar equation on the form (2.49) which greatly simplifies trajectory planning and stability analysis of the full system.

Conserved quantity for reduced dynamics

The behavior of the reduced system dynamics is much easier to classify than the original system. An important property of (2.49) is the existence of a conserved quantity that keeps its value along the target trajectory defined by the virtual holonomic constraints.

Theorem 2. *[18, Theorem 1]: Along any solution $\theta(t)$ of the equation*

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0$$

provided $\alpha(\theta) \neq 0$ the integral function

$$I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0) = \dot{\theta}^2 - \psi(\theta, \theta_0)\left[\dot{\theta}_0^2 - \int_{\theta_0}^{\theta} \psi(\theta_0, s)\frac{2\gamma(s)}{\alpha(s)}\mathrm{d}s\right] \tag{2.52}$$

where

$$\psi(x, \theta_0) = \exp\left\{ -2\int_{\theta_0}^{x} \frac{\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau\right\} \tag{2.53}$$

preserves its zero value.

Proof. The proof of this theorem is readily available in [13, 15, 18] among others and will not be repeated here. □

The integral in (2.52) is not a true first integral for the system (2.46) since it only preserves it's value along a specific trajectory defined by the virtual holonomic constraints. In other words, the conserved quantity is only conserved along those trajectories that keep the virtual holonomic constraints invariant. This property is useful in the sense that (2.52) can be used as a measure of discrepancy between the target orbit defined by (2.49) and the current configuration of the robot in state space. In order to facilitate such an argument, knowledge of time derivative of the function $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$ is needed.

Theorem 3. *[18, Theorem 2]: With x and y being some constants, the time derivative of the function $I(\theta, \dot{\theta}, x, y)$ defined by (2.52), calculated along a solution $[\theta(t), \dot{\theta}(t)]$ of the system*

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = W \tag{2.54}$$

can be computed as

$$\frac{\mathrm{d}}{\mathrm{d}t}I = \dot{\theta}\left\{\frac{2}{\alpha(\theta)}W - \frac{2\beta(\theta)}{\alpha(\theta)}I\right\} \tag{2.55}$$

Proof. The time derivative of $I = I(\theta, \dot\theta, x, y)$ along a solution of (2.49) is

$$\frac{\mathrm{d}}{\mathrm{d}t} I = \dot\theta \frac{\partial}{\partial\theta} I + \ddot\theta \frac{\partial}{\partial\dot\theta} I \tag{2.56}$$

where

$$\begin{aligned}\frac{\partial}{\partial\theta} I = &- \left[\dot\theta_0^2 - \int_{\theta_0}^{\theta} \psi(\theta_0, s)\frac{2\gamma(s)}{\alpha(s)}\mathrm{d}s\right] \frac{\partial}{\partial\theta}\psi(\theta,\theta_0) \\ &+ \psi(\theta,\theta_0)\frac{\partial}{\partial\theta}\left[\int_{\theta_0}^{\theta} \psi(\theta_0, s)\frac{2\gamma(s)}{\alpha(s)}\mathrm{d}s\right]\end{aligned} \tag{2.57}$$

which using the Fundamental Theorem of Calculus can be simplified to

$$\begin{aligned}\frac{\partial}{\partial\theta}\psi(\theta,\theta_0) &= \frac{\partial}{\partial\theta}\exp\left\{ -2\int_{\theta_0}^{\theta}\frac{\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau\right\} \\ &= -2\,\psi(\theta,\theta_0)\frac{\partial}{\partial\theta}\int_{\theta_0}^{\theta}\frac{\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau \\ &= -\,\psi(\theta,\theta_0)\frac{2\beta(\theta)}{\alpha(\theta)} \\ \frac{\partial}{\partial\theta}\left[\int_{\theta_0}^{\theta}\psi(\theta_0,s)\frac{2\gamma(s)}{\alpha(s)}\mathrm{d}s\right] &= \psi(\theta_0,\theta)\frac{2\gamma(\theta)}{\alpha(\theta)} = \frac{1}{\psi(\theta,\theta_0)}\frac{2\gamma(\theta)}{\alpha(\theta)}\end{aligned} \tag{2.58}$$

leading to the expressions of the partial derivatives on the form

$$\begin{aligned}\frac{\partial}{\partial\theta} I &= \left[\dot\theta_0^2 - \int_{\theta_0}^{\theta} \psi(\theta_0, s)\frac{2\gamma(s)}{\alpha(s)}\mathrm{d}s\right]\psi(\theta,\theta_0)\frac{2\beta(\theta)}{\alpha(\theta)} \\ &\quad + \frac{\psi(\theta,\theta_0)}{\psi(\theta,\theta_0)}\frac{2\gamma(\theta)}{\alpha(\theta)} \\ &= -\left[I - \dot\theta^2\right]\frac{2\beta(\theta)}{\alpha(\theta)} + \frac{2\gamma(\theta)}{\alpha(\theta)} \\ \frac{\partial}{\partial\dot\theta} I &= 2\dot\theta\end{aligned} \tag{2.59}$$

The time derivative can then be calculated using the fact that $\alpha(\theta) \neq 0$ as

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t} I &= \dot\theta\left\{\frac{2\gamma(\theta)}{\alpha(\theta)} - \frac{2\beta(\theta)}{\alpha(\theta)}\left[I - \dot\theta^2\right]\right\} + \frac{\Big(W - \beta(\theta)\dot\theta^2 - \gamma(\theta)\Big)}{\alpha(\theta)}2\dot\theta \\ &= \dot\theta\left\{\frac{2}{\alpha(\theta)}W - \frac{2\beta(\theta)}{\alpha(\theta)}I\right\}\end{aligned} \tag{2.60}$$

□

It is trivial to verify that the time derivative of the integral (2.52) is indeed zero along the target trajectory ($W = 0, I = 0$), and that I keeps its zero value along the orbit.

Energy formulation for reduced dynamics

The integral (2.52) is not the only conserved quantity for the reduced system. Lagrangian systems such as the planar biped possess first integrals for their equations of motion on the form (2.42), and it's reasonable to assume that similar quantities exist for the reduced system.

Lemma 1. *[5, Lemma 3]: Along any solution $\theta(t)$ of the nonlinear system*

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0 \tag{2.61}$$

the energy function, if well-defined for some constant x,

$$E_x(\theta, \dot{\theta}) = \frac{1}{2}\Psi_x(\theta)\dot{\theta}^2 + \int_x^\theta \frac{\gamma(\tau)}{\alpha(\tau)}\Psi_x(\tau)\mathrm{d}\tau \tag{2.62}$$

where

$$\Psi_x(\theta) = \exp\left\{\int_x^\theta \frac{2\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau\right\} \tag{2.63}$$

preserves its value $E_x(\theta(t), \dot{\theta}(t)) \equiv E_x(\theta(0), \dot{\theta}(0))$. In particular, we have

$$E_x(\theta(0+), \dot{\theta}(0+)) \equiv E_x(\theta(T-), \dot{\theta}(T-)) \tag{2.64}$$

for the time moments right after an impact and right before the next impact.

Proof. The idea behind the proof is to restate (2.61) in a form that is reminiscent of the Euler-Lagrange equations of motion by exploiting the Lagrangian $\mathcal{L}(\theta, \dot{\theta}) = \frac{1}{2}M_\theta(\theta)\dot{\theta}^2 - \mathcal{P}(\theta)$. First multiplying (2.61) with the scalar integrating factor $\mu(\theta)$ yields

$$\mu(\theta)\alpha(\theta)\ddot{\theta} + \mu(\theta)\beta(\theta)\dot{\theta}^2 + \mu(\theta)\gamma(\theta) = 0 \tag{2.65}$$

where $\mu(\theta)$ will be chosen such that (2.65) correspond with

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{\partial\mathcal{L}(q,\dot{q})}{\partial\dot{q}}\right] - \frac{\partial\mathcal{L}(q,\dot{q})}{\partial q} = M_\theta(\theta)\ddot{\theta} + \frac{1}{2}\frac{\partial M_\theta(\theta)}{\partial\theta}\dot{\theta}^2 + \mu(\theta)\gamma(\theta) \tag{2.66}$$

where the last term is due to the fact that

$$\frac{\partial\mathcal{P}(q)}{\partial q} = G(q) \leftrightarrow G(\Phi(\theta)) = \bar{\gamma}(\theta) = \mu(\theta)\gamma(\theta) \tag{2.67}$$

according to (2.50). Comparing terms between (2.65) and (2.66) yields the following relations

$$\begin{aligned} \mu(\theta)\alpha(\theta) &= M_\theta(\theta) \\ \mu(\theta)\beta(\theta) &= \frac{1}{2}\frac{\partial\left(\mu(\theta)\alpha(\theta)\right)}{\partial\theta} = \frac{1}{2}\left(\frac{\partial\mu(\theta)}{\partial\theta}\alpha(\theta) + \mu(\alpha)\frac{\partial\alpha(\theta)}{\partial\theta}\right) \end{aligned} \tag{2.68}$$

Rearranging the equation and integrating using separation of variables gives

$$\begin{aligned} 2\int_x^\theta \frac{\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau &= \int_{\alpha(x)}^{\alpha(\theta)} \frac{1}{\alpha(\tau)}\mathrm{d}\tau + \int_{\mu(x)}^{\mu(\theta)} \frac{1}{\mu(\tau)}\mathrm{d}\tau \\ \exp\left\{2\int_x^\theta \frac{\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau\right\} &= \frac{\exp\left\{\ln\alpha(\theta) + \ln\mu(\theta)\right\}}{\exp\left\{\ln\alpha(x) + \ln\mu(x)\right\}} \\ \mu(\theta) &= \frac{\mu(x)\alpha(x)}{\alpha(\theta)}\exp\left\{2\int_x^\theta \frac{\beta(\tau)}{\alpha(\tau)}\mathrm{d}\tau\right\} = \frac{\mu(x)\alpha(x)}{\alpha(\theta)}\Psi_x(\theta) \end{aligned} \tag{2.69}$$

Substituting $\mu(\theta)$ (with $\mu(x)\alpha(x) = 1$) into the energy expression yields

$$\begin{aligned} E_x(\theta(t),\dot{\theta(t)}) &= \frac{1}{2}M_\theta(\theta)\dot{\theta}^2 + \mathcal{P}(\theta) = \frac{1}{2}\mu(\theta)\alpha(\theta)\dot{\theta}^2 + \int_x^\theta \mu(\tau)\gamma(\tau)\mathrm{d}\tau \\ &= \frac{1}{2}\Psi_x(\theta)\dot{\theta}^2 + \int_x^\theta \frac{\gamma(\tau)}{\alpha(\tau)}\Psi_x(\tau)\mathrm{d}\tau \end{aligned} \tag{2.70}$$

□

Verifying that (2.62)is a conserved quantity for (2.61) is easily checked by time differentiation of the integral along the trajectories of the reduced dynamics resulting in $\frac{\mathrm{d}}{\mathrm{d}t}E_x(\theta(t),\dot{\theta}(t)) = 0$.

Similarly to (2.52) the energy-like function for the reduced dynamics is not a true first integral since it does not keep its value for all possible initial conditions of (2.46).

2.7 Numerical Optimization

Searching for limit cycles of mechanical dynamic systems such as the compass-gait biped in (2.14) is a daunting task. The sheer number of variables, and the relative small region of attraction for the periodic orbit make manual searches next to impossible to perform with any degree of success.

In order to obtain meaningful results in a reasonable amount of time, the search must be structured mathematically. One way of doing this is transforming the problem from finding limit cycles to finding minimizers of an objective, and then employing numerical optimization algorithms to find such a minimizer.

The first step in creating this new problem formulation is defining a criterion which should be minimized. In the case of a limit cycle, this could intuitively be chosen as the error between initial conditions of the gait and the configuration of the biped after impact. If these two state vectors are identical, a periodic orbit has been found. This can be formulated as

$$\min_{q_0,\dot{q}_0} f(q_0,\dot{q}_0) = \|q^+ - q_0\|^2 + \|\dot{q}^+ - \dot{q}_0\|^2 \tag{2.71}$$

where $q_0, \dot{q}_0$ is the initial conditions of the gait and $q^+, \dot{q}^+$ is the updated state vector after impact caluclated using (2.14).

After defining the optimization criterion for the problem, a choice of algorithm for the search procedure must be made. Numerical searches performed in this thesis will be implemented in MATLAB and will utilize the built-in nonlinear optimization function `fminsearch`. The algorithm is based on the derivative-free Nelder-Mead method that attempts to find the minimum of a nonlinear unconstrained optimization problem, specified by an objective function $f(x)$ and an initial estimate x_0. The inner-workings of the implementation of the optimization routine is beyond the scope of this text, but a comprehensive treatment of the Nelder-Mead method can be found in [14] and documentation of the `fminsearch` function is readily available in [11].

Algorithm 2 below illustrates how optimization can be used to define a search for limit cycles. The approach described is most likely extremely computationally intensive and a minimizer might not be found, as such it is just meant as an example of the flow in an optimization routine.

Algorithm 2: Searching for periodic orbits using numerical optimization

```
IN contains the user-defined initial guess
OUT contains the appropriate parameters returned by the routine

Set initial guess: $q_{iter} \leftarrow IN(1), \quad \dot{q}_{iter} \leftarrow IN(2)$
Set termination criteria: $tolerance \leftarrow IN(3)$

**while** $f > tolerance$ **do**
    Compute next iteration: $[q_{iter}, \dot{q}_{iter}] \leftarrow \mathbf{fminsearch}(f, q_{iter}, \dot{q}_{iter})$;

    Initialize: $q_0 \leftarrow q_{iter}, \quad \dot{q}_0 \leftarrow \dot{q}_{iter}$
    **repeat**
        Integrate: $[q, \dot{q}] \leftarrow M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = 0$
    **until** $q \in \mathcal{S}$;

    Compute update: $\Gamma^+ \ni [q^+, \dot{q}^+]^T \leftarrow F(q^-, \dot{q}^-), \qquad [q^-, \dot{q}^-]^T \in \Gamma^-$;

    Compute state mismatch: $f \leftarrow \|q_{iter} - q^+\|^2 + \|\dot{q}_{iter} - \dot{q}^+\|^2$;
**end**

$OUT \leftarrow [q_{iter}, \dot{q}_{iter}]$
```

3 Modeling the Compass-gait Biped

This chapter is concerned with the techniques employed for modeling the continuous and discrete dynamics of the compass-gait biped. The Euler-Lagrange equations of motion for the system will be presented along with the full hybrid dynamics describing the walker. In order to the make the subsequent discussion more comprehensible, terminology relevant to the physical composition of the planar biped will first be presented.

3.1 Terminology

Stance leg

The term *Stance leg* refers to the balancing leg of the biped during a step, and is not generally connected to a specific leg of the robot.

Swing leg

The term *Swing leg* refers to the leg that experiences detachment from the ground and travels from behind the robot to the front during a complete step.

Impact

After the swing leg has reached the end of the swing phase, it will hit the slope that the biped travels along. This is referred to as the *Impact* the robot experiences when transitioning to the next step.

Heel-strike

The *Heel-strike* is the specific impact that occurs during a normal walking-cycle, meaning that the swing leg has hit the slope in front of the robot while the swing foot is descending.

3.2 Physical Description

The compass-gait biped is one of the simplest walking devices and is a standard benchmark for studying periodic gaits. The robot consists of two links, or legs, connected by a revolute hip joint that allows for rotational motion in the 2-D plane. Figure 3.1 illustrates the planar biped along with some important parameters. The orientation of the stance leg and swing leg are described, respectively, by the absolute angles q_1 and q_2 with positive direction defined counter-clockwise. The biped is traveling down a shallow slope with constant angle ψ measured from the horizontal. Each leg link is identical with total length $l = a + b$ from hip joint to feet. The center of mass of each of the legs is located at the distances a and b from the hip and feet,

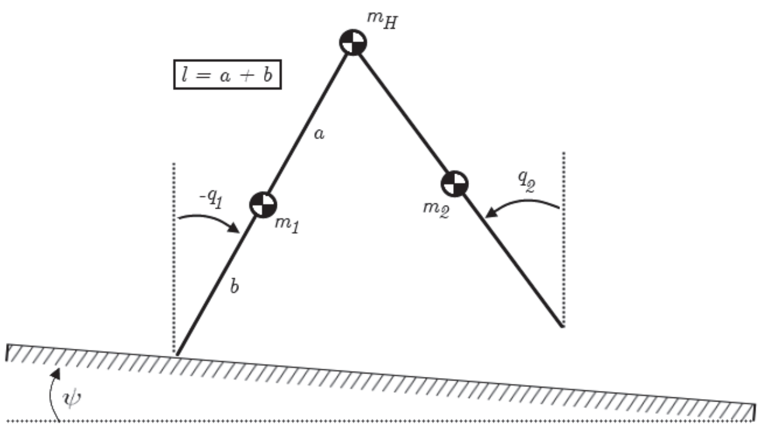

Figure 3.1: Schematic of the planar compass-gait biped on a shallow slope with angle ψ.

respectively. $m_1 = m_2$ is the mass of the legs and m_H is the mass of the hip joint connecting each leg.

To simplify the derivation of the dynamics of the biped, some assumptions have been made regarding the physical composition of the robot and the physics of the impact.

- The legs and the hip are considered to be point masses and the moment of inertia around their centers of mass equals zero.
- The stance leg is rigidly connected to the ground during a complete step of the walking gait, meaning that each point of the robot can be uniquely described by the two angles q_1, q_2.
- The robot has point feet that does not experience slipping or sliding on the slope.
- The swing leg only experiences an impact with the ground during heel-strike. Practically, the swing leg would trespass the surface of the slope (and experience an impact) when it passes through the stance leg due to the symmetry of the legs. The biped can be modified to prevent this (see e.g [12]), but this does not affect the dynamics and will not be treated in this thesis.
- The impact forces the foot of the swing leg experience during the collision with the ground can be represented by impulses, and the impact itself takes place over an infinitesimally small period of time.

- Due to the impulsive nature of the impact forces, the robot configuration q remains unchanged during the collision with the ground. An argument for the validity of this assumption can be found in [1].

Having defined the assumptions and physical parameters connected to the planar biped, the equations of motion can be derived using the Euler-Lagrange approach.

3.3 Euler-Lagrange Equations of Motion

The first step to deriving the Lagrangian of the biped is to define the homogeneous transfer matrices that describe the orientation and position of each link of the robot. One natural placement of local coordinate frames is indicated in Figure 3.2, where each frame is rigidly attached to the appropriate point mass. The origins of each of these frames describe the following

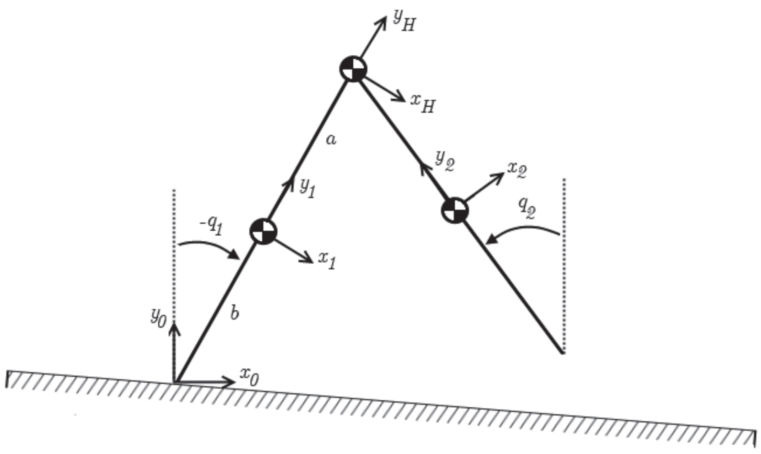

Figure 3.2: Assignment of origins for the coordinate frames of the biped.

important points on the planar biped

o_0 - Global coordinate frame fixed to the ground.

o_1 - Coordinate frame fixed to the center of mass for the stance leg.

o_H - Coordinate frame fixed to the center of mass of the hip joint

o_2 - Coordinate frame fixed to the center of mass for the swing leg.

These coordinate frames are used to form homogeneous transformation matrices on the form

$$H_j^i = \begin{bmatrix} R_j^i & p_j^i \\ 0_{1\times 3} & 1 \end{bmatrix} \tag{3.1}$$

where $R^i_j \in \mathbb{R}^{3\times 3}$ is the rotation matrix from frame j to frame i and $p^i_j \in \mathbb{R}^3$ is the distance between the origins of the respective frames expressed in frame i. Finding rotation matrices is generally complicated, requiring a parametrization of the total rotation of each frame in suitable coordinates (see [20]). This process is greatly simplified by the fact that the planar biped only experiences motion in a 2-D plane. The transformation matrices can then be found by simple trigonometry and are stated for verification below:

$$\begin{aligned}
H^0_1 &= \begin{bmatrix} \cos(-q_1) & -\sin(-q_1) & 0 & -b\sin(-q_1) \\ -\sin(-q_1) & \cos(-q_1) & 0 & b\cos(-q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
H^1_H &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
H^0_H &= H^0_1 H^1_H \\
H^0_2 &= \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & l\sin(-q_1) + a\sin(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & l\cos(-q_1) - a\cos(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.2}$$

The matrices (3.2) can now be employed to determine the position $p^{(0)}_{\dots}$ and velocity $v^{(0)}_{\dots}$ of the center of mass of each point mass in xy coordinates in the global frame o_0 as

$$\begin{aligned}
p^{(0)}_1 &= \begin{bmatrix} I_{2\times 2} & 0_{2\times 2} \end{bmatrix} \cdot H^0_1 \cdot \begin{bmatrix} 0_{3\times 1} \\ 1 \end{bmatrix}, & v^{(0)}_1 &= \frac{\mathrm{d}}{\mathrm{d}t} p^{(0)}_1 \\
p^{(0)}_H &= \begin{bmatrix} I_{2\times 2} & 0_{2\times 2} \end{bmatrix} \cdot H^0_H \cdot \begin{bmatrix} 0_{3\times 1} \\ 1 \end{bmatrix}, & v^{(0)}_H &= \frac{\mathrm{d}}{\mathrm{d}t} p^{(0)}_H \\
p^{(0)}_2 &= \begin{bmatrix} I_{2\times 2} & 0_{2\times 2} \end{bmatrix} \cdot H^0_2 \cdot \begin{bmatrix} 0_{3\times 1} \\ 1 \end{bmatrix}, & v^{(0)}_2 &= \frac{\mathrm{d}}{\mathrm{d}t} p^{(0)}_2
\end{aligned} \tag{3.3}$$

where $I_{2\times 2}$ is the identity matrix and $p^{(0)}_{\dots}, v^{(0)}_{\dots} \in \mathbb{R}^2$.

In order to form the Lagrangian of the system and compute the Euler-Lagrange equations, the kinetic and potential energy of the system must be determined. The potential energy is the sum of the potential energy at the center of mass for each point mass and can be expressed as

$$\mathcal{P} = \left(m_1 h^{(0)}_1 + m_H h^{(0)}_H + m_2 h^{(0)}_2 \right) g \tag{3.4}$$

where g is the gravitational constant and $h^{(0)}_{\dots} = p^{(0)}_{\dots} \cdot \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ is the height of the respective centers of mass expressed in the global frame o_0. The kinetic energy is simply the sum of the translational energy of each point mass and can be expressed as

$$\mathcal{K} = \frac{1}{2}\left(m_1 v_1^2 + m_H v_H^2 + m_2 v_2^2\right) \tag{3.5}$$

Using the expressions (3.4) and (3.5) for the potential and kinetic energy, the Euler-Lagrange equations of motion can be calculated

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_j}\right) - \frac{\partial \mathcal{L}}{\partial q_j} = 0 \quad j \in [1,2] \tag{3.6}$$

where $\mathcal{L}$ is the Lagrangian of the system derived from (3.4) and (3.5) as

$$\mathcal{L} = \mathcal{K} - \mathcal{P}$$

Using the expressions (2.9) - (2.11) for the elements of the respective matrices, the equations of motion for the passive compass-gait biped can be formulated as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = 0 \tag{3.7}$$

where

$$M(q) = \begin{bmatrix} p_1 & -p_2\cos(q_1 - q_2) \\ -p_2\cos(q_1 - q_2) & p_3 \end{bmatrix} \tag{3.8}$$

$$C(q,\dot{q}) = \begin{bmatrix} 0 & -p_2\dot{q}_2\sin(q_1 - q_2) \\ p_2\dot{q}_1\sin(q_1 - q_2) & 0 \end{bmatrix} \tag{3.9}$$

$$G(q) = \begin{bmatrix} -p_4\sin(q_1) \\ p_5\sin(q_2) \end{bmatrix} \tag{3.10}$$

with the constant parameters $p_1 = m_H l^2 + m_1 b^2 + m_2 l^2$, $p_2 = m_2 l a$, $p_3 = m_2 a^2$, $p_4 = (m_1 b + m_2 l + m_H l)\, g$, $p_5 = m_2 a g$. The equations of motion (3.7) describe the continuous dynamics of the biped during the periodic gait and is independent of the walking surface. In order to describe what happens to the robot during heel-strike, an impact map must be formulated to prevent trespassing of the surface slope during motion.

3.4 Formulating Impact Map

When the swing foot impacts the surface of the slope, an update of the angular velocities $\dot{q}_1, \dot{q}_2$ should occur to prevent the biped from falling through the floor. This update can be formulated as a mapping between the velocities just before and just after the collision with the ground on the form

$$\begin{bmatrix} \dot{q}_1^+ \\ \dot{q}_2^+ \end{bmatrix} = \Delta(q) \cdot \begin{bmatrix} \dot{q}_1^- \\ \dot{q}_2^- \end{bmatrix} \tag{3.11}$$

where the $-$ and $+$ signs denote the time instant right before and right after impact, respectively. An important property of this impact mapping is the assumption that the configuration of the biped, the generalized coordinates q_1, q_2, remains unchanged during ground impact. This is due to the fact that the impact forces F the biped experiences during impact are impulsive in nature. An illustration of the impact forces and reference points for the impact maps is depicted in Figure 3.3. There are multiple ways of calculating

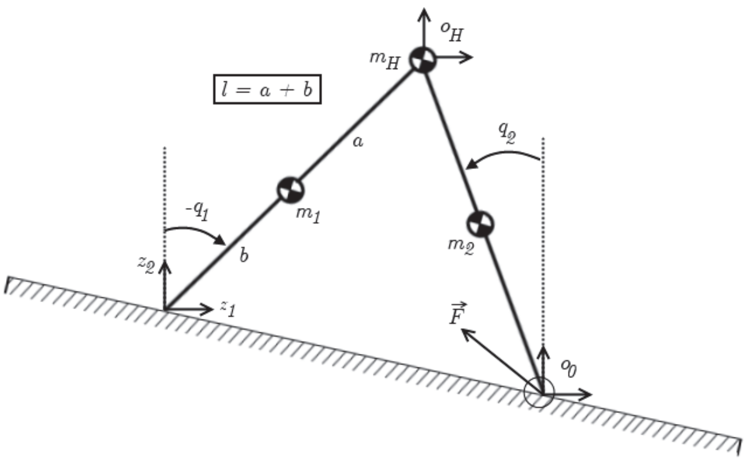

Figure 3.3: Assignment of reference points for impact maps and impact forces.

the velocity updates of the biped. Presented below are two distinct methods that exploit different properties of the impact to derive an impact map for the collision on the form (3.11).

Conservation of angular momentum

Since the impact forces F are the only external forces affecting the biped, the angular momentum about the impacting foot is conserved before and after the collision for the system. The angular momentum L of a point mass can be stated as

$$L = r \times mv \tag{3.12}$$

where r is the position of the mass relative to a given reference point, m is the mass and v is the velocity of the particle. Given that the biped is a system of point masses, the angular momentum of the robot about the impacting foot is given by

$$L^{(0)}_{Biped} = \sum_i r_i^{(0)} \times m_i v_i^+ = \sum_i r_i^{(0)} \times m_i v_i^-, \quad i \in \{1, 2, H\} \tag{3.13}$$

where the reference point is the origin o_0 (see Figure 3.3]), and the position vectors $r_i^{(0)}$ relative to this point is given by

$$\begin{aligned} r_2^{(0)} &= \begin{bmatrix} -b\sin(q_2) \\ b\cos(q_2) \\ 0 \end{bmatrix} \\ r_H^{(0)} &= \begin{bmatrix} -l\sin(q_2) \\ l\cos(q_2) \\ 0 \end{bmatrix} \\ r_1^{(0)} &= r_H^{(0)} + \begin{bmatrix} -a\sin(-q_1) \\ -a\cos(-q_1) \\ 0 \end{bmatrix} \end{aligned} \tag{3.14}$$

The translational velocities $v_i^\pm$ are independent of the reference point and can be expressed using the angular velocities $\dot{q}^\pm$ as

$$\begin{aligned} v_H^- &= \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1^- \end{bmatrix} \times \begin{bmatrix} l\sin(-q_1) \\ l\cos(-q_1) \\ 0 \end{bmatrix}, & v_H^+ &= \begin{bmatrix} 0 \\ 0 \\ \dot{q}_2^+ \end{bmatrix} \times \begin{bmatrix} -l\sin(q_2) \\ l\cos(q_2) \\ 0 \end{bmatrix} \\ v_1^- &= \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1^- \end{bmatrix} \times \begin{bmatrix} b\sin(-q_1) \\ b\cos(-q_1) \\ 0 \end{bmatrix}, & v_1^+ &= v_H^+ + \begin{bmatrix} 0 \\ 0 \\ \dot{q}_1^+ \end{bmatrix} \times \begin{bmatrix} -a\sin(-q_1) \\ -a\cos(-q_1) \\ 0 \end{bmatrix} \\ v_2^- &= v_H^- + \begin{bmatrix} 0 \\ 0 \\ \dot{q}_2^- \end{bmatrix} \times \begin{bmatrix} a\sin(q_2) \\ -a\cos(q_2) \\ 0 \end{bmatrix}, & v_2^+ &= \begin{bmatrix} 0 \\ 0 \\ \dot{q}_2^+ \end{bmatrix} \times \begin{bmatrix} -b\sin(q_2) \\ b\cos(q_2) \\ 0 \end{bmatrix} \end{aligned} \tag{3.15}$$

Substituting (3.14) and (3.15) into (3.13) and computing the crossproducts, yields one equation for the two unknown velocities $\dot{q}_1^+, \dot{q}_2^+$. This means that another equation is needed to solve the system.

The only forces that the pre-impact swing leg experiences during the collision is the constraint force acting on it from the hip joint. This means that the angular momentum of this leg about the hip is conserved through the impact, yielding another equation for the updated velocities on the form

$$\begin{aligned} L_{Swing}^{(H)} &= r_1^{(H)} \times m_1 v_1^+ = r_1^{(H)} \times m_1 v_1^- \\ r_1^{(H)} &= \begin{bmatrix} -a\sin(-q_1) \\ -a\cos(-q_1) \\ 0 \end{bmatrix} \end{aligned} \tag{3.16}$$

where the reference point is the origin o_H and the velocities $v_1^{\pm}$ is given in (3.15). Equations (3.13) and (3.16) combined results in the linear system

$$\begin{aligned} \begin{bmatrix} L_{Biped}^{(0)} \\ L_{Swing}^{(H)} \end{bmatrix} &= \mathcal{Q}_+ \dot{q}^+ = \mathcal{Q}_- \dot{q}^- \\ \begin{bmatrix} L_{Biped}^{(0)} \\ L_{Swing}^{(H)} \end{bmatrix} &= \begin{bmatrix} p_8 - p_7 c_{12} & -p_7 c_{12} + p_6 \\ p_8 & -p_7 c_{12} \end{bmatrix} \begin{bmatrix} \dot{q}_1^+ \\ \dot{q}_2^+ \end{bmatrix} = \begin{bmatrix} p_{10} c_{12} & -p_{11} \\ -p_9 & 0 \end{bmatrix} \begin{bmatrix} \dot{q}_1^- \\ \dot{q}_2^- \end{bmatrix} \end{aligned} \tag{3.17}$$

where $c_{12} = \cos(q_1 - q_2)$ and the parameters $p_6 = m_1 l^2 + m_H l^2 + m_2 b^2$, $p_7 = m_1 a l$, $p_8 = m_1 a^2$, $p_9 = m_1 a b$, $p_{10} = m_2 l b + m_H l^2 + m_1 l b$, $p_{11} = m_2 a b$. Solving the system for $\dot{q}^+$ by inverting the matrix $\mathcal{Q}_+$ gives and impact map on the form (3.11)

$$\dot{q}^+ = \left[\mathcal{Q}_+^{-1} \cdot \mathcal{Q}_-\right] \dot{q}^- \tag{3.18}$$

Integration of dynamics

Computing the impact map by conservation of angular momentum results in a compact mapping that only contains information about the updated velocities $\dot{q}^+$. Another quantity of interest is the impact forces F that occur at the end of the swing leg as a result of the impact, since the assumption that the biped does not slide or trespass the ground just after the collision is dependent on these forces. It might therefore be of interest to calculate the magnitude and direction of F to ensure that the assumption holds.

A procedure for finding an impact model that includes the unknown forces $F = [F_x,\ F_y]^T$ in addition to the update velocities $\dot{q}^+$ is utilized in [9] and will be presented here for the biped. The first step of the procedure is to express the continuous dynamics of the biped in excessive coordinates

$$x = \begin{bmatrix} q_1, & q_2, & z_1, & z_2 \end{bmatrix}^T \tag{3.19}$$

where z_1, z_2 are Cartesian coordinates added to the foot of the stance leg as indicated in Figure 3.3. The main idea of the procedure is then to "integrate" these new continuous dynamics over the infinitesimally small duration of the impact to obtain

$$M_x(q)\left(\dot{x}^+ - \dot{x}^-\right) = F^{ext} \tag{3.20}$$

where $F^{ext} = \int_{t-}^{t+} \delta F^{ext}(\tau)\mathrm{d}\tau$ is the impulse forces affecting each coordinate in x integrated over the duration of the impact and $M_x(q) \in \mathbb{R}^{4\times 4}$ is the inertia matrix of the excessive coordinate dynamics. Finding these dynamics is identical to the procedure described previously with the exception that z_1, z_2 must be included in the transformation matrices yielding

$$\begin{aligned}
H_1^0 &= \begin{bmatrix} \cos(-q_1) & -\sin(-q_1) & 0 & z_1 - b\sin(-q_1) \\ -\sin(-q_1) & \cos(-q_1) & 0 & z_2 + b\cos(-q_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
H_H^1 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
H_H^0 &= H_1^0 H_H^1 \\
H_2^0 &= \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & z_1 + l\sin(-q_1) + a\sin(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & z_2 + l\cos(-q_1) - a\cos(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{3.21}$$

The calculation then proceeds as normal with the new state vector x, resulting in the inertia matrix

$$M_x(q) = \left[\begin{matrix} (m_H l^2 + m_1 b^2 + m_2 l^2) & -m_2 a l\cos(q_1 - q_2) \\ -m_2 a l\cos(q_1 - q_2) & m_2 a^2 \\ -\cos(q_1)(m_H l + m_1 b + m_2 l) & m_2 a\cos(q_2) \\ -\sin(q_1)(m_H l + m_1 b + m_2 l) & m_2 a\sin(q_2) \end{matrix}\right. \cdots$$

$$\cdots \left.\begin{matrix} -\cos(q_1)(m_H l + m_1 b + m_2 l) & -\sin(q_1)(m_H l + m_1 b + m_2 l) \\ m_2 a\cos(q_2) & m_2 a\sin(q_2) \\ m_1 + m_2 + m_H & 0 \\ 0 & m_1 + m_2 + m_H \end{matrix}\right] \tag{3.22}$$

The linear system (3.20) has eight unknowns $\dot{x}^+, F^{ext}$ and four equations, and must therefore be augmented with equations that describe what happens to each contact end of the biped during impact. It is assumed that the stance leg detaches from the ground without interaction after the collision, and thus F^{ext} need only consider external forces at the end of the swing leg. This point has the Cartesian coordinates

$$\Upsilon = \begin{bmatrix} z_1 + l\sin(-q_1) + l\sin(q_2) \\ z_2 + l\cos(-q_1) - l\cos(q_2) \end{bmatrix} \tag{3.23}$$

measured in a global frame with the same orientation as the z_1, z_2 frame. The external forces F^{ext} affecting the end of the swing leg can then be described in terms of the horizontal F_x and vertical F_y components of the impact force F as

$$F^{ext} = E^T \begin{bmatrix} F_x \\ F_y \end{bmatrix} \tag{3.24}$$

where

$$E = \frac{\partial \Upsilon}{\partial x} = \begin{bmatrix} -l\cos(q_1) & l\cos(q_2) & 1 & 0 \\ -l\sin(q_1) & l\sin(q_2) & 0 & 1 \end{bmatrix} \tag{3.25}$$

The stance leg is assumed to act as a pivot before impact, meaning that $\dot{z}_1^- = \dot{z}_2^- = 0$. Right after impact the former swing leg becomes the new stance leg, and assuming no slipping or rebounding of this new pivot we have the relation

$$\frac{\mathrm{d}\Upsilon}{\mathrm{d}t} = \frac{\partial \Upsilon}{\partial x}\dot{x}^+ = E\dot{x}^+ = 0 \tag{3.26}$$

Combining equations (3.20), (3.24) and (3.26) results in a system linear in $\dot{x}^+$ and F, with six equations and six unknowns which can be stated as

$$\begin{bmatrix} M_x & -E^T \\ E & 0_{2\times 2} \end{bmatrix} \begin{bmatrix} \dot{x}^+ \\ F \end{bmatrix} = \begin{bmatrix} M_x\dot{x}^- \\ 0_{2\times 1} \end{bmatrix} \tag{3.27}$$

It is verified in [9] that an unique solution of the system always exist, and can be written

$$\begin{bmatrix} \dot{x}^+ \\ F \end{bmatrix} = \begin{bmatrix} M_x & -E^T \\ E & 0_{2\times 2} \end{bmatrix}^{-1} \begin{bmatrix} M_x\dot{x}^- \\ 0_{2\times 1} \end{bmatrix} \tag{3.28}$$

By extracting the updated velocities $\dot{q}_1^+, \dot{q}_2^+$ from $\dot{x}^+$, the system reduces to the previously found impact map using conservation of angular momentum.

3.5 Hybrid System Formulation

After calculating both the continuous and the discrete dynamics of the compass-gait biped, the full hybrid dynamic system can be formulated.

Defining impact surface

The previously found impact maps calculated the change in angular velocities that occur when the biped impacts with the ground. In order for this update to correctly be applied when the swing foot strikes the slope, the configurations of the robot that results in an impact must be determined. These configurations correspond to the hypersurface $\mathcal{S}$ known as the impact surface or switching surface. Referring to Figure 3.4, a configuration of the

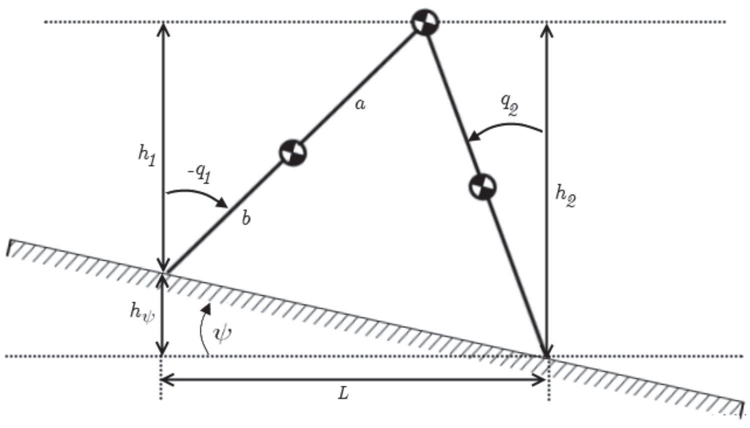

Figure 3.4: Illustration showing the different quantities used when defining the impact surface of the biped.

biped that leads to impact with the slope must satisfy the relation

$$H(q) = h_1(q) + h_\psi(q) - h_2(q) = 0 \tag{3.29}$$

where

$$\begin{aligned} h_1 &= l\cos(-q_1) \\ h_2 &= l\cos(q_2) \\ h_\psi &= L\tan(\psi) \\ L &= l\sin(-q_1) + l\sin(q_2) \end{aligned} \tag{3.30}$$

are found by simple trigonometry. Substituting these expressions and simplifying using trigonometric identities leads to the following derivation

$$\begin{aligned} H(q) &= l\cos(-q_1) + [l\sin(-q_1) + l\sin(q_2)]\frac{sin(\psi)}{\cos(\psi)} - l\cos(q_2) = 0 \\ H(q) &= \underbrace{\cos(q_1)\cos(\psi) - \sin(q_1)\sin(\psi)} - \underbrace{[\cos(q_2)\cos(\psi) - \sin(q_2)\sin(\psi)]} = 0 \\ H(q) &= \qquad \cos(q_1 + \psi) \qquad - \qquad \cos(q_2 + \psi) \qquad = 0 \end{aligned}$$

The switching surface $\mathcal{S}$ is defined as all configurations q of the biped that satisfies the above relation and can be stated in set notation as

$$\mathcal{S} = \left\{q \in \mathbb{R}^2 : \quad H(q) = \cos(q_1 + \psi) - \cos(q_2 + \psi) = 0\right\} \tag{3.31}$$

Transition during impact

When the foot of the swing leg hits the switching surface $\mathcal{S}$ an instantaneous transition occurs where the stance and swing leg switch. After this transition the biped will be in the exact same configuration as shown in Figure 3.1 with the exception that the generalized coordinates q_1 and q_2 has switched places. Instead of calculating new continuous dynamics for the case where the stance leg is parameterized by q_2 instead of q_1, a trivial permutation of the position variable can be performed

$$\begin{bmatrix} q_1^+ \\ q_2^+ \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q_1^- \\ q_2^- \end{bmatrix} = P \begin{bmatrix} q_1^- \\ q_2^- \end{bmatrix} \tag{3.32}$$

such that (3.7) is still valid after impact. The permutation of the generalized coordinates means that the impact map must reflect this switch as well. Applying the change in coordinates to the impact map derived using conservation of angular momentum yields the new mapping

$$\begin{bmatrix} \dot{q}_1^+ \\ \dot{q}_2^+ \end{bmatrix} = P_q(q) \begin{bmatrix} \dot{q}_1^- \\ \dot{q}_2^- \end{bmatrix} \tag{3.33}$$

where

$$\begin{aligned}
P_q(q) &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \mathcal{Q}_+^{-1} \mathcal{Q}_- \\
P_q(q) &= \left(\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}^{-1} \right)^{-1} \mathcal{Q}_+^{-1} \mathcal{Q}_- \\
P_q(q) &= \left(\mathcal{Q}_+ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right)^{-1} \mathcal{Q}_- \\
P_q(q) &= \begin{bmatrix} -p_7 c_{12} + p_6 & p_8 - p_7 c_{12} \\ -p_7 c_{12} & p_8 \end{bmatrix}^{-1} \begin{bmatrix} p_{10} c_{12} & -p_{11} \\ -p_9 & 0 \end{bmatrix}
\end{aligned} \tag{3.34}$$

The update of the impact model obtained by integrating the dynamics is similarly trivial and will not be stated here.

Hybrid dynamics

Using the continuous dynamics (3.7), together with the switching surface defined by (3.31) and the algebraic transition equations (3.32) and (3.33), the full hybrid dynamics of the compass-gait biped can be stated as

$$\begin{array}{lll} M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = 0, & \text{as long as} & q \notin \mathcal{S} \\ q^+ = Pq^- \quad \text{and} \quad \dot{q}^+ = P_q(q)\dot{q}^- & \text{whenever} & q^- \in \mathcal{S} \end{array} \tag{3.35}$$

3.6 MATLAB Simulation Script

The dynamics associated with the periodic gait of the planar biped are difficult to visualize based on the equations of motion alone, or plots of their respective trajectory through state space. When investigating properties of different gaits, an animation that shows the corresponding motion of the biped is extremely helpful. Using the hybrid mechanical system derived in (3.35), a simulation script was written in MATLAB to give visual feedback during the simulation studies. The relevant parts of the main simulation loop are reproduced below.

```
%% Simulate periodic gait

x0 = [q0;Dq0];

MAP_SELECTOR = 1;
% 1 = Conservation of angular momentum
% 2 = Integration of dynamics

SIMULATION_TIME = 30;
DRAW_INTERVAL = 0.05;  % Interval in seconds
                       % between state space plotting

REPEATS = 0;           % Animation repeats
FPS = 24;              % Animation playback fps

T0 = 0;
tspan = [T0 SIMULATION_TIME];

state_space = [];
time = [];
impacts = [];
last_impact = 0;
gait_period = [];

current_time = T0;

while(current_time < SIMULATION_TIME)

    options = odeset('Events', @impact_event);
    [tout, xout, event_time, event_state, event_id] = ode45(...
        @equations_of_motion, tspan, x0, options);

    if ~isempty(event_id) && event_time(end) == tout(end)
        impact_time = tout(end);
        impact_index = length(time)+length(tout);

        impacts = [impacts; impact_time, impact_index];
        gait_period = [gait_period; impact_time - last_impact];
        last_impact = impact_time;
        x0 = impact_map(xout(end,:), MAP_SELECTOR);

        fprintf('************************************************\n');
        fprintf('Impact surface hit at time = %0.2f\n', impact_time);
        fprintf('Index = %d\n', impact_index);
        fprintf('q1 = %0.2f\n', xout(end,1)*180/pi);
        fprintf('q2 = %0.2f\n', xout(end,2)*180/pi);
        fprintf('************************************************\n');
    end

    time = [time; tout];
    state_space = [state_space; xout];

    tspan = [time(end) max(SIMULATION_TIME, time(end))];
    current_time = time(end);
end
```

Implementation

After the hybrid dynamic system (3.35) is derived using Maple, a conversion to MATLAB syntax is performed to simplify implementation. The continuous dynamics (3.7) are then implemented in state space form as the MATLAB function `equations_of_motion(t,x)`:

```
function [Dx] = equations_of_motion(t,x)
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    q1 = x(1);
    q2 = x(2);
    Dq1 = x(3);
    Dq2 = x(4);
    Dq = [Dq1;Dq2];

    m11 = p1;
    m12 = -p2 * cos(q1 - q2);
    m21 = -p2 * cos(q1 - q2);
    m22 = p3;
    c11 = 0;
    c12 = -Dq2 * sin(q1 - q2) * p2;
    c21 = Dq1 * sin(q1 - q2) * p2;
    c22 = 0;
    g1 = -sin(q1) * p4;
    g2 = sin(q2) * p5;

    M = [m11, m12; m21, m22];
    C = [c11, c12; c21, c22];
    G = [g1; g2];

    DDq = M\(-C*Dq-G);
    Dx = [Dq; DDq];
end
```

The state space representation is employed in order to utilize the numerical differential solvers included in MATLAB, where the most widely used is the default solver `ode45()`. The following code excerpt shows an implementation for numerical simulating the previous mentioned `equations_of_motion.m`:

```
[t,x,te,ye,ie] = ode45(@equations_of_motion,tspan,x0,options);
```

The code instructs MATLAB to simulate `equations_of_motion.m` over the time horizon `tspan` with initial condition `q0`. The output from the function is the time vector `t`, state vector `q` and three parameters related to the `options` parameter. The problem with this implementation is that it only simulates the equations of motion over a predefined time period and without detecting crossings of the switching surface $\mathcal{S}$. The solution to these problems can be found in the `ode45` parameter `options`.

```
options = odeset('Events', @impact_event);
```

The important parameter here is the event function `impact_event.m`, which is an implementation of (3.31) for the numerical solver.

```
function [value, isterminal, direction] = impact_event(t,x)
     global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    q1 = x(1);
    q2 = x(2);

    Dq2 = x(4);

    value = cos(q1+psi)-cos(q2+psi);
    direction = -1;

    if Dq2 < 0
        isterminal = 1;
    else
        isterminal = 0;
    end
end
```

Event functions are used to detect zero-crossings of certain algebraic expressions in the solver, in this case it notifies when $H(q)$ in (3.29) crosses zero and $\dot{q}_2 < 0$ which signifies a heel-strike. When `ode45` receives this notification, the integration is stopped and relevant data is returned.

The state space data from `ode_45` is then used to calculate the updated positions and velocities of the biped according to (3.33) implemented in the `impact_map(x,MAP_SELECTOR)` function.

```
function [x_plus] = impact_map(x_minus, map_selector)
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    q1 = x_minus(1);
    q2 = x_minus(2);
    Dq1 = x_minus(3);
    Dq2 = x_minus(4);

    switch map_selector
        case 1
            %% Impulse map using conservation of angular momentum

            Qp = [-cos(q1 - q2) * p7 + p6 p8 - cos(q1 - q2) * p7; ...
                -cos(q1 - q2) * p7 p8;];
            Qn = [cos(q1 - q2) * p10 - p9 -p11; -p9 0;];

            Dq_plus = Qp\Qn*[Dq1; Dq2];

        case 2
            %% Impulse map using method from Grizzle, Abba, Plestan 2001

            [mH, m1, m2, a, b, l, g] = physical_parameters;

            A = [mH * l ^ 2 + b ^ 2 * m1 + l ^ 2 * m2 -a * l * m2 ...
                * cos(q1 - q2) -cos(q1) * (mH * l + b * m1 + l * m2)...
                -sin(q1) * (mH * l + b * m1 + l * m2) l * cos(q1) l ...
                * sin(q1); -a * l * m2 * cos(q1 - q2) a ^ 2 * m2 a ...
                * m2 * cos(q2) a * m2 * sin(q2) -l * cos(q2) -l ...
                * sin(q2); -cos(q1) * (mH * l + b * m1 + l * m2) a ...
                * m2 * cos(q2) m1 + m2 + mH 0 -1 0; -sin(q1) ...
                * (mH * l + b * m1 + l * m2) a * m2 * sin(q2) 0 m1...
                + m2 + mH 0 -1; -l * cos(q1) l * cos(q2) 1 0 0 0; ...
                -l * sin(q1) l * sin(q2) 0 1 0 0;];
            B = [(mH * l ^ 2 + b ^ 2 * m1 + l ^ 2 * m2) * Dq1 ...
                - m2 * l * a * cos(q1 - q2) * Dq2 -m2 * l * Dq1 ...
                * a * cos(q1 - q2) + a ^ 2 * m2 * Dq2 -cos(q1) ...
                * (mH * l + b * m1 + l * m2) * Dq1 + a ...
                * m2 * cos(q2) * Dq2 -sin(q1) ...
                * (mH * l + b * m1 + l * m2) * Dq1 ...
                + a * m2 * sin(q2) * Dq2 0 0];

            z = A\B';
            Dq_plus=[z(2);z(1)];
    end

    %% Updated state vector

    q_plus = [q2; q1];
    x_plus = [q_plus; Dq_plus];

end
```

This function returns $q^+, \dot{q}^+$ which are set as new initial conditions for the numerical solver and the simulation continues.

The loop iterates as long as the time returned from the solver is less than the specified simulation time, at which point the simulation terminates and returns a matrix containing the complete state space of the system (`state_space`), a vector containing information about the time interval between each step (`gait_period`), a matrix containing the impact times and indexes (`impacts`) and the simulation time (`time`). The simulation data are then utilized to plot relevant data and display an animation of the compass-gait biped traveling down the slope, see Figure 3.5.

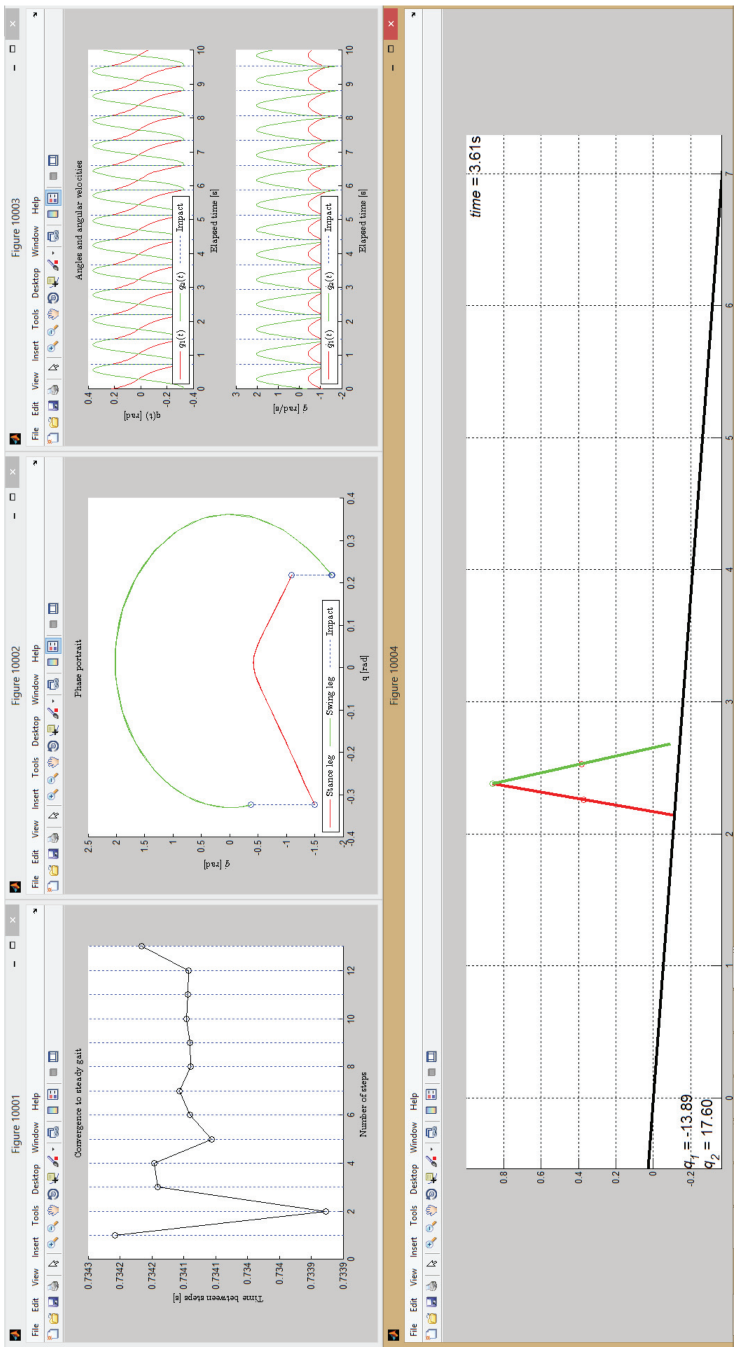

Figure 3.5: Screenshot of output from MATLAB *simulation script. The three plots show convergence of gait period, phase portrait of swing and stance leg, and time-evolution of swing and stance leg. The largest figure window plays an animation of the biped traveling down the slope based on the time-evolution of q_1 and q_2.*

4 Reduced Dynamics Formulation

In this chapter the reduced dynamics of (3.7) will be derived using the notion of virtual holonomic constraints. The reduced dynamics of the biped is an useful tool when searching for periodic gaits and will be used extensively throughout the thesis.

4.1 Virtual Holonomic Constraints

The virtual holonomic constraints of the biped can be stated as

$$q_1(t) = \phi_1(\theta(t)), \quad q_2(t) = \phi_2(\theta(t)) \tag{4.1}$$

where $\theta(t)$ is a scalar variable that uniquely defines a particular point on the continuous sub-arc of the periodic trajectory of (3.7), and $\phi_i(\theta(t))$ are smooth functions which shape depends on the choice of θ. There are many possible choices for the parameterization $\theta(t)$, which serves as a measure of progress throughout the periodic motion. For the compass-gait biped there exists a natural choice for $\theta(t)$ which simplifies the reduced dynamics considerably.

Consider one complete step of the biped as illustrated in Figure 4.1. The

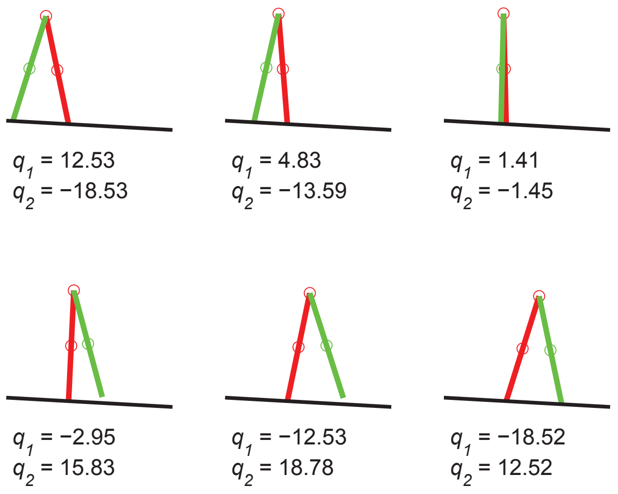

Figure 4.1: Illustration of a complete step of a periodic gait. The values q_1, q_2 are the angles in degrees of the stance leg (red) and swing leg (green) respectively.

stance leg experiences a monotonic decrease in angle q_1 throughout the complete step. This means that there exist an one-to-one correspondence between

q_1 and the distance the system has traveled along its periodic orbit. The coordinate of the stance leg can therefore be chosen as the progress variable $\theta(t)$ resulting in the reparameterization of the orbit as

$$q_1 = \theta, \quad q_2 = \phi(\theta) \tag{4.2}$$

where the dependence on time has been dropped for convenience and the derivatives are calculated as

$$\begin{aligned} \dot{q}_1 &= \dot{\theta}, & \ddot{q}_1 &= \ddot{\theta} \\ \dot{q}_2 &= \phi'(\theta)\dot{\theta}, & \ddot{q}_2 &= \phi''(\theta)\dot{\theta}^2 + \phi'(\theta)\ddot{\theta} \end{aligned} \tag{4.3}$$

4.2 Reduced Dynamics

Substituting the expressions for the virtual holonomic constraints from (4.2) - (4.3) into the equations of motion (3.7) and collecting similar terms, yields two new differential equations for the variable θ on the form

$$\alpha_1(\theta)\ddot{\theta} + \beta_1(\theta)\dot{\theta}^2 + \gamma_1(\theta) = 0 \tag{4.4}$$

$$\alpha_2(\theta)\ddot{\theta} + \beta_2(\theta)\dot{\theta}^2 + \gamma_2(\theta) = 0 \tag{4.5}$$

where

$$\begin{aligned} \alpha_1(\theta) &= -p_2\cos(\theta - \phi(\theta))\phi'(\theta) + p_1 \\ \beta_1(\theta) &= -p_2\sin(\theta - \phi(\theta))\left(\phi'(\theta)\right)^2 - p_2\cos(\theta - \phi(\theta))\phi''(\theta) \\ \gamma_1(\theta) &= -p_4\sin(\theta) \\ \alpha_2(\theta) &= -p_2\cos(\theta - \phi(\theta)) + p_3\phi'(\theta) \\ \beta_2(\theta) &= p_2\sin(\theta - \phi(\theta)) + p_3\phi''(\theta) \\ \gamma_2(\theta) &= p_5\sin(\phi(\theta)) \end{aligned} \tag{4.6}$$

These differential equations are known as the reduced dynamics of the system and describe the synchronization between the generalized coordinates q_1, q_2 along a periodic trajectory. They both possess the common solution $\theta = \theta(t)$ and are integrable with conserved quantities described by the equations (2.52) and (2.62).

Solving for the conserved quantities analytically is a difficult, if not impossible task, for some systems due to the nested integrals in the expressions. The planar biped is no exception, but due to the fact that it is a mechanical system, an analytic expression for the first integral exist in the form of the total energy

$$\mathcal{E} = \dot{q}^T \frac{1}{2} M(q, \dot{q})\dot{q} + \mathcal{P}(q) \tag{4.7}$$

which restated in virtual holonomic constraints can be written

$$E = \frac{1}{2} M_\theta(\theta)\dot{\theta}^2 + \mathcal{P}(\theta) \tag{4.8}$$

where $M_\theta(\theta)$ is a scalar representation of the inertia matrix $M(q, \dot{q})$. Looking at the derivation of (2.62) it is clear that (4.8) must be the conserved quantity for some equation on the form

$$\alpha_0(\theta)\ddot{\theta} + \beta_0(\theta)\dot{\theta}^2 + \gamma_0(\theta) = 0 \tag{4.9}$$

where

$$\begin{aligned} \alpha_0 &= \omega_1(\theta)\alpha_1(\theta) + \omega_2(\theta)\alpha_2(\theta) \\ \beta_0 &= \omega_1(\theta)\beta_1(\theta) + \omega_2(\theta)\beta_2(\theta) \\ \gamma_0 &= \omega_1(\theta)\gamma_1(\theta) + \omega_2(\theta)\gamma_2(\theta) \end{aligned} \tag{4.10}$$

are linear combinations of the coefficients of (4.4) and (4.5) using θ-dependent weights $\omega_i(\theta)$. The challenge now is to find such weights that it is possible to restore the original energy of the Euler-Lagrange system as a conserved quantity of (4.9).

4.3 Finding Weights to Restore Energy

Deriving integral formula

The energy expression (4.8) can be differentiated to yield

$$\frac{\mathrm{d}}{\mathrm{d}t}E = \frac{\partial E}{\partial \theta}\dot{\theta} + \frac{\partial E}{\partial \dot{\theta}}\ddot{\theta} = \dot{\theta}\left(\frac{1}{2}\frac{\partial M_\theta}{\partial \theta}\dot{\theta}^2 + \frac{\partial \mathcal{P}}{\partial \theta} + M_\theta \ddot{\theta}\right) \tag{4.11}$$

where the fact that E is a conserved quantity and $\dot{\theta} \neq 0$ on the periodic orbit means that the expression in parenthesis can be stated as

$$M_\theta \ddot{\theta} + \frac{1}{2}\frac{\partial M_\theta}{\partial \theta}\dot{\theta}^2 + \frac{\partial \mathcal{P}}{\partial \theta} = 0 \tag{4.12}$$

Comparing this equation to the generic reduced dynamics system

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0 \tag{4.13}$$

motivates the following equalities among the parameters

$$\begin{aligned} M_\theta &= \alpha(\theta) \\ \frac{1}{2}\frac{\partial M_\theta}{\partial \theta} &= \beta(\theta) \\ \frac{\partial \mathcal{P}}{\partial \theta} &= \gamma(\theta) \end{aligned} \tag{4.14}$$

The α, β and γ parameters can be represented as weighted combinations of (4.4) and (4.5) on the form (4.10). These expressions contain the two unknowns ω_1, ω_2, which can be reduced to a single unknown using normalization

$$\frac{\omega_2(\theta)}{\omega_1(\theta)} = \mu(\theta) \tag{4.15}$$

without loss of generality due to the fact that (4.9) is identically equal to zero along the periodic orbit. This normalization leads to the system of equations

$$\begin{aligned} M_\theta &= \alpha_1(\theta) + \mu(\theta)\alpha_2(\theta) \\ \frac{1}{2}\frac{\partial M_\theta}{\partial \theta} &= \beta_1(\theta) + \mu(\theta)\beta_2(\theta) \end{aligned} \tag{4.16}$$

where M_θ and μ are unknowns. In further computations the explicit dependence on θ will be dropped from the equations for readability. The system (4.16) can easily be reduced to a single equation in the unknown μ by differentiating the first expressions with respect to θ resulting in

$$\frac{\partial M_\theta}{\partial \theta} = \frac{\partial \alpha_1}{\partial \theta} + \frac{\partial \mu}{\partial \theta}\alpha_2 + \mu\frac{\partial \alpha_2}{\partial \theta} \tag{4.17}$$

which can be equated with the expression for $\frac{\partial M_\theta}{\partial \theta}$ with respect to β_1, β_2 giving the single differential equation

$$\frac{\partial \alpha_1}{\partial \theta} + \frac{\partial \mu}{\partial \theta}\alpha_2 + \mu\frac{\partial \alpha_2}{\partial \theta} = 2\left[\beta_1 + \mu\beta_2\right] \tag{4.18}$$

Rearranging the equation by collecting μ-terms on the left-hand side and generic parameters on the right-hand side yields a first-order non-homogeneous differential equation in μ on the form

$$\begin{aligned} \frac{\partial \mu}{\partial \theta} + \left(\frac{\frac{\partial \alpha_2}{\partial \theta} - 2\beta_2}{\alpha_2}\right)\mu &= \frac{2\beta_1 - \frac{\partial \alpha_1}{\partial \theta}}{\alpha_2} \\ \frac{\partial \mu}{\partial \theta} + f(\theta)\mu &= g(\theta) \end{aligned} \tag{4.19}$$

where the functions $f(\theta), g(\theta)$ are introduced to clearly illustrate the composition of the differential equation. Straightforward computations using the integrating factor $e^{\left(\int_{\theta_0}^{\theta} f(\tau)\mathrm{d}\tau\right)}$ yields

$$\mu(\theta) = \Psi(\theta, \theta_0)\mu(\theta_0) + \Psi(\theta, \theta_0)\int_{\theta_0}^{\theta} \Psi(\theta_0, \tau)\frac{2\beta_1 - \frac{\partial \alpha_1}{\partial \tau}}{\alpha_2}\mathrm{d}\tau \tag{4.20}$$

where

$$\Psi(\theta,\theta_0) = \exp\left(-\int_{\theta_0}^{\theta} \frac{\frac{\partial \alpha_2}{\partial \tau} - 2\beta_2}{\alpha_2} \mathrm{d}\tau\right) \tag{4.21}$$

Unfortunately, (4.20) is difficult to solve analytically since the last term once again contains nested integrals of highly nonlinear functions. On the other hand, this last term does not contain any expressions for μ. This means that if it is possible to find a weight that restores the Euler-Lagrange energy of the original system without solving the integrals, this solution can be used in linear combination with (4.20) to obtain a simpler expressions for the weight $\mu(\theta)$.

Deriving weight by inspection

The fact that the compass-gait biped is a mechanical system means that the kinetic energy of the robot is readily available, and in fact already calculated when the Euler-Lagrange equations of motion for the system was derived. Inserting the virtual holonomic constraints (4.2) into the expression for kinetic energy (3.5) gives the expression

$$\begin{aligned} K &= \left(\frac{p_1}{2} - p_2 \cos(\theta - \phi(\theta))\phi'(\theta) + \frac{p_3}{2}(\phi'(\theta))^2\right)\dot{\theta}^2 \\ K &= \frac{1}{2} M_\theta(\theta,\dot{\theta})\dot{\theta}^2 \end{aligned} \tag{4.22}$$

Utilizing the equation for M_θ and μ in terms of α_1, α_2 from (4.16), μ can easily be solved for by exploiting the knowledge of the kinetic energy

$$\mu(\theta) = \frac{\frac{2K}{\dot{\theta}^2} - \alpha_1}{\alpha_2} \tag{4.23}$$

which after substitution of the appropriate expressions and simplifying reduces to

$$\mu(\theta) = \phi'(\theta) \tag{4.24}$$

Finding family of weights

Substituting the equality $\mu(\theta) = \phi'(\theta)$ into the integral formula (4.20) gives a new relation for $\phi'(\theta)$

$$\phi'(\theta) = \Psi(\theta,\theta_0)\phi'(\theta_0) + \Psi(\theta,\theta_0)\int_{\theta_0}^{\theta} \Psi(\theta_0,\tau)\frac{2\beta_1 - \frac{\partial \alpha_1}{\partial \tau}}{\alpha_2}\mathrm{d}\tau \tag{4.25}$$

where $\phi'(\theta_0)$ is a constant that satisfies the equality, and the second term of the equation is identical to the expression in (4.20). Creating a linear combination of (4.20) and (4.25) allows for a simplification of the generic weight $\mu(\theta)$ as

$$\begin{aligned}\mu(\theta) - \phi'(\theta) &= \Psi(\theta, \theta_0)(\mu(\theta_0) - \phi'(\theta_0)) \\ \mu(\theta) - \phi'(\theta) &= \exp\left(-\int_{\theta_0}^{\theta} \frac{\frac{\partial \alpha_2}{\partial \tau} - 2\beta_2}{\alpha_2} \mathrm{d}\tau\right)(\mu(\theta_0) - \phi'(\theta_0))\end{aligned} \tag{4.26}$$

where the integral can be further simplified by the relation

$$\exp\left(-\int_{\theta_0}^{\theta} \frac{\frac{\partial \alpha_2}{\partial \tau} - 2\beta_2}{\alpha_2} \mathrm{d}\tau\right) = \frac{\alpha_2(\theta_0)}{\alpha_2(\theta)} \exp\left(2\int_{\theta_0}^{\theta} \frac{\beta_2}{\alpha_2} \mathrm{d}\tau\right) \tag{4.27}$$

The weights $\mu(\theta)$ that restore the Euler-Lagrange energy conserved quantity for the reduced dynamics (4.9) of the compass-gait biped are solutions of the equation

$$\mu(\theta) = \frac{\alpha_2(\theta_0)}{\alpha_2} \exp\left(2\int_{\theta_0}^{\theta} \frac{\beta_2}{\alpha_2} \mathrm{d}\tau\right)(\mu(\theta_0) - \phi'(\theta_0)) + \phi'(\theta) \tag{4.28}$$

where θ_0 is an arbitrarily chosen initial condition that describes the zero point of motion along the desired periodic trajectory. This expression shows that it is possible to find weights such that second-order differential equation (4.9) conserves an important property of the full dynamics of the biped (3.7), namely the total energy of the system. The conserved quantity (4.8) will prove to be an important tool when searching for a differential equation for the unknown function $q_2 = \phi(\theta)$.

5 Searching for Periodic Gaits

In this chapter a general procedure for finding periodic gaits of the compass-gait biped will be presented.

5.1 Parameters of the Periodic Gait

The nontrivial symmetric gaits of the compass-gait biped can be uniquely defined by the parameter vector

$$p_\star = [a, b, c, d, e, f, g, h, T]^T \in \mathbb{R}^9 \tag{5.1}$$

where T is the time between consecutive impacts of the swing foot and the other parameters[5] are defined by

$$\begin{array}{lclcl} q_\star(0+) & = & [q_{1\star}(0+), q_{2\star})0+)]^T & = & [a, e]^T \\ \dot{q}_\star(0+) & = & [\dot{q}_{1\star}(0+), \dot{q}_{2\star}(0+)]^T & = & [b, f]^T \\ q_\star(T-) & = & [q_{1\star}(T-), q_{2\star}(T-)]^T & = & [c, g]^T \\ \dot{q}_\star(T-) & = & [\dot{q}_{1\star}(T-), \dot{q}_{2\star}(T-)]^T & = & [d, h]^T \end{array} \tag{5.2}$$

The eight parameters listed in (5.2) are not independent quantities due to the fact that they are connected by the switching law (3.32) and (3.33). This mapping between the parameters is valid at the switching surface defined by (3.31), and will be exploited to reduce the number of parameters in the search procure.

Relations among position parameters

The switching surface is hit whenever the equality

$$\cos(c + \psi) - \cos(g + \psi) = 0 \tag{5.3}$$

is satisfied, where c, g are given by (5.2) and ψ is the angle of the shallow slope the biped is traveling along. The trivial solution to this equation is $c = g$ which corresponds to a configuration of the robot where the legs are in the exact same position. This is a static configuration meaning that the biped is about to fall if switching occurs at this point and is therefore not of interest.

The impact of interest should occur when c and g have opposite signs, indicating a compass-like configuration of the biped as shown in Figure 3.3.

[5]The parameters a, b, g are not related to the physical parameters defining the leg length of the biped or the gravitational constant presented in Chapter 3, but are simply a convenient notation for use in the rest of the text.

Exploiting the fact that $\cos(x) = \cos(-x)$ means that a nontrivial solution of (5.3) can be stated as

$$c = -g - 2\psi + 2k\pi, \quad k \in \mathbb{Z} \tag{5.4}$$

where $k = 0$ since a feasible configuration of the biped must have $c \in (-\pi, 0)$ for the range of slopes $\psi \in (0, \frac{\pi}{2})$. The relation between the angles at impact can therefore be simply stated as

$$c = -g - 2\psi \tag{5.5}$$

There also exists an explicit relation between the position variables just before and just after impact, which is given by the mapping (3.32) as

$$\underbrace{\begin{bmatrix} a \\ e \end{bmatrix}}_{q_\star(0+)} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} c \\ g \end{bmatrix}}_{q_\star(T-)} = \begin{bmatrix} g \\ c \end{bmatrix} \tag{5.6}$$

Using the relations (5.5) and (5.6) reduces the number of independent parameters of the periodic gait to six, and further reduction is possible by defining relations between the generalized velocities.

Relations among velocity parameters

The impact map (3.33) derived using conservation of energy maps the generalized velocities just before impact to the generalized velocities just after impact on the form

$$\underbrace{\begin{bmatrix} b \\ f \end{bmatrix}}_{\dot{q}_\star(0+)} = P\left(\underbrace{\begin{bmatrix} c \\ g \end{bmatrix}}_{q_\star(T-)}\right) \underbrace{\begin{bmatrix} d \\ h \end{bmatrix}}_{\dot{q}_\star(T-)} \tag{5.7}$$

Making the substitutions $g = a$ and $c = -a - 2\psi$ from (5.5) - (5.6) and writing out the expressions yields

$$\begin{aligned} b &= \frac{[(p_8p_{10} - p_7p_9)\cos(2a + 2\psi)]\, d - p_8p_{11}\, h}{p_6p_8 - p_7^2\cos^2(2a + 2\psi)} \\ f &= \frac{[p_7p_{10}\cos^2(2a + 2\psi) - p_6p_9]\, d - [p_7p_{11}\cos(2a + 2\psi)]\, h}{p_6p_8 - p_7^2\cos^2(2a + 2\psi)} \end{aligned} \tag{5.8}$$

which are functions of the parameters a, d, h.

The virtual holonomic constraints (4.2) express $q_2 = \phi(\theta)$ in terms of $q_1 = \theta$. It is therefore convenient to express the gait parameters e, f, g, h as functions of a, b, c, d, which is already the case for the position parameters. By solving the first equation of (5.8) for h and substituting into the second equation gives two new relations for f, h as

$$\begin{aligned} f &= \frac{p_7 \cos(2a + 2\psi)\, b - p_9\, d}{p_8} \\ h &= \frac{[p_7^2 \cos^2(2a + 2\psi) - p_6 p_8]\, b + [(p_8 p_{10} - p_7 p_9)\cos(2a + 2\psi)]\, d}{p_8 p_{11}} \end{aligned} \tag{5.9}$$

which are functions of a, b, d and the physical parameters $p_7, p_8, p_9, p_{10}, p_{11}$ are given by (3.17).

Describing all relations in terms of a, b

The linear combination of (4.4) and (4.5) with the weights $\mu_1(\theta) = 1$ and $\mu_2(\theta) = \phi'(\theta)$ possesses the conserved quantity

$$\begin{aligned} E(\theta, \dot\theta) = E_0 = &\Big(\frac{p_1}{2} - p_2 \cos(\theta - \phi(\theta))\phi'(\theta) + \frac{p_3}{2}(\phi'(\theta))^2\Big)\, \dot\theta^2 \\ &+ p_4 [\cos(\theta) - 1] + p_5 [1 - \cos(\phi(\theta))] \end{aligned} \tag{5.10}$$

where $E_0 = E(\theta(0+), \dot\theta(0+))$ due to the fact that E is a conserved quantity along the periodic trajectory defined by (4.4) and (4.5).

Assuming that the velocities of q_1 before and after impact is nonzero, meaning $b \neq 0$ and $d \neq 0$, the parameters of the gait can be rewritten in the terms of the virtual holonomic constraints as

$$\begin{array}{llll} \theta_\star(0) = a, & \dot\theta_\star(0) = b, & \theta_\star(T) = c, & \dot\theta_\star(T) = d \\ \phi(a) = e, & \phi'(a) = f/b, & \phi(c) = g, & \phi'(c) = h/d \end{array} \tag{5.11}$$

Substituting these expressions into the function (5.10) yields two equivalent representations of the conserved energy

$$\begin{aligned} E(a, b) =& \frac{p_1 b^2}{2} - p_2 \cos(a - e) f b + \frac{p_3 f^2}{2} \\ &+ p_4 [\cos(a) - 1] + p_5 [1 - \cos(e)] \\ E(c, d) =& \frac{p_1 d^2}{2} - p_2 \cos(c - g) h d + \frac{p_3 h^2}{2} \\ &+ p_4 [\cos(c) - 1] + p_5 [1 - \cos(g)] \end{aligned} \tag{5.12}$$

where both functions equal E_0 on the periodic orbit. Equating the two functions and substituting the parameters c, e, f, g, h for the expressions (5.5), (5.6) and (5.9), and collecting terms involving d yields

$$A(a,b)d^2 + B(a,b)d + C(a,b) = 0 \tag{5.13}$$

where

$$\begin{aligned}
A(a,b) =& -\frac{(p_8\,p_{10} - p_7\,p_9)\,(\cos(2\,a + 2\,\psi))^2\,p_2}{p_8\,p_{11}} \\
&+ \frac{1}{2}\,\frac{(p_8\,p_{10} - p_7\,p_9)^2\,(\cos(2\,a + 2\,\psi))^2\,p_3}{{p_8}^2{p_{11}}^2} \\
&+ \frac{1}{2}\,p_1 - \frac{1}{2}\,\frac{{p_9}^2 p_3}{{p_8}^2} \\
B(a,b) =& -\left(-\frac{p_7\,\cos(2\,a + 2\,\psi)\,p_9\,p_3}{b{p_8}^2} + \frac{p_9\,\cos(2\,a + 2\,\psi)\,p_2}{p_8\,b}\right) b^2 \\
&- \left(\frac{(\cos(2\,a + 2\,\psi))^2\,{p_7}^2}{p_8\,p_{11}} - \frac{p_6}{p_{11}}\right) b\cos(2\,a + 2\,\psi)\,p_2 \\
&+ \frac{1}{p_8 p_{11}}\left(\frac{(\cos(2\,a + 2\,\psi))^2\,{p_7}^2}{p_8\,p_{11}} - \frac{p_6}{p_{11}}\right) b\,(p_8\,p_{10} - p_7\,p_9)\cos(2\,a + 2\,\psi)\,p_3 \\
C(a,b) =& \frac{1}{2}\left(\frac{(\cos(2\,a + 2\,\psi))^2\,{p_7}^2}{p_8\,p_{11}} - \frac{p_6}{p_{11}}\right)^2 b^2 p_3 \\
&- \left(-\frac{p_7\,(\cos(2\,a + 2\,\psi))^2\,p_2}{p_8} + \frac{1}{2}\,p_1 + \frac{1}{2}\,\frac{{p_7}^2\,(\cos(2\,a + 2\,\psi))^2\,p_3}{{p_8}^2}\right) b^2 \\
&+ \cos(a + 2\,\psi)\,p_4 - \cos(a)\,p_5 + \cos(a + 2\,\psi)\,p_5 - \cos(a)\,p_4
\end{aligned} \tag{5.14}$$

This is a quadratic equation with respect to d, and at best it has two real solutions for a given values of a and b.

Examining the relations (5.5), (5.6), (5.9) and (5.13) it can easily be seen that the original nine parameters given by (5.2) only contain three independent quantities, namely a, b and T. The period of the gait is not relevant since the reparameterization of the gait in terms of virtual holonomic constraints have removed the explicit dependence of time. This means that the only parameters which must be determined through a numerical search is the initial conditions for the stance leg $q_{1\star}(0+) = \theta_{1\star}(0+)$ and $\dot{q}_{1\star}(0+) = \dot{\theta}_{1\star}(0+)$.

5.2 Deriving a Differential Equation for $\phi(\theta)$

The reduced dynamics (4.4) and (4.5) can be considered as linear algebraic equations in the variables $\ddot{\theta}_\star$ and $\dot{\theta}_\star^2$. Restating these equations on matrix form yields the system

$$\begin{bmatrix} \alpha_1(\theta_\star) & \beta_1(\theta_\star) \\ \alpha_2(\theta_\star) & \beta_2(\theta_\star) \end{bmatrix} \begin{bmatrix} \ddot{\theta}_\star \\ \dot{\theta}_\star^2 \end{bmatrix} = \begin{bmatrix} -\gamma_1(\theta_\star) \\ -\gamma_2(\theta_\star) \end{bmatrix} \tag{5.15}$$

which is solvable assuming that the determinant $\alpha_1(\theta_\star)\beta_2(\theta_\star) - \alpha_2(\theta_\star)\beta_1(\theta_\star)$ is separated from zero on a sub-arc of the periodic cycle[6]. This assumption guarantees that the matrix is invertible and the system can be solved with respect to $\ddot{\theta}_\star$ and $\dot{\theta}_\star^2$ as follows

$$\ddot{\theta}_\star = \frac{\beta_1(\theta_\star)\gamma_2(\theta_\star) - \beta_2(\theta_\star)\gamma_1(\theta_\star)}{\alpha_1(\theta_\star)\beta_2(\theta_\star) - \alpha_2(\theta_\star)\beta_1(\theta_\star)} \tag{5.16}$$

$$\dot{\theta}_\star^2 = \frac{\alpha_2(\theta_\star)\gamma_1(\theta_\star) - \alpha_1(\theta_\star)\gamma_2(\theta_\star)}{\alpha_1(\theta_\star)\beta_2(\theta_\star) - \alpha_2(\theta_\star)\beta_1(\theta_\star)} \tag{5.17}$$

Another equation for the unknown $\dot{\theta}_\star^2$ can be found by rewriting the relation on energy (5.10) as

$$\dot{\theta}^2 = \frac{E_0 - p_4\left[\cos(\theta) - 1\right] - p_5\left[1 - \cos(\phi(\theta))\right]}{\frac{p_1}{2} - p_2\cos(\theta - \phi(\theta))\phi'(\theta) + \frac{p_3}{2}(\phi'(\theta))^2} \tag{5.18}$$

where E_0 is the initial total energy of the system which is kept constant along the target trajectory. Using this relation in conjunction with (5.17) and then collecting terms involving $\phi''(\theta_\star)$ results in a second order differential equation for the unknown function $\phi(\cdot)$ given by

$$\phi''(\theta_\star) = f_0(a, b, \theta_\star, \phi(\theta_\star), \phi'(\theta_\star)) \tag{5.19}$$

where

$$\begin{aligned} f_0(a,b,\theta,\phi(\theta),\phi'(\theta)) = \big[&-2(-\sin(\theta-\phi(\theta))p_2(-\cos(\theta-\phi(\theta))p_2\phi'(\theta) - p_2\cos(\theta-\phi(\theta))\phi'(\theta)^2 + \\ &p_3\phi'(\theta)^3 + p_1))(E_0 - p_4\cos(\theta) - p_5 + p_5\cos(\phi(\theta)) + p_4) - (-\sin(\theta)p_4p_3\phi'(\theta) + \\ &\sin(\theta)p_4p_2\cos(\theta-\phi(\theta)) + \sin(\phi(\theta))p_5\cos(\theta-\phi(\theta))p_2\phi'(\theta) - \sin(\phi(\theta))p_5p_1)(p_1 + p_3\phi'(\theta)^2 - \\ &2\cos(\theta-\phi(\theta))p_2\phi'(\theta))\,\big] \,/\, \left[2\,(-p_1p_3 + {p_2}^2\cos^2(\theta-\phi(\theta)))(E_0 - p_4\cos(\theta) - p_5 + p_5\cos(\phi(\theta)) + p_4)\right] \end{aligned}$$

with $E_0 = E(a, b)$ from (5.12).

[6] An argument proving $\alpha_1(\theta_\star)\beta_2(\theta_\star) - \alpha_2(\theta_\star)\beta_1(\theta_\star)$ is always separated from zero on the periodic orbit of the compass-gait biped is presented in [5], and will not be discussed here.

Examining (5.19) it is clear that $\phi(\cdot)$ is uniquely defined by a nonautonomous differential equation with explicit dependence on the independent variable θ. Finding an analytic expression for $\phi(\theta)$ using (5.19) is a difficult task, but the possibility of solving for the unknown synchronization function numerically is an important tool when searching for periodic gaits of the biped.

5.3 Description of Search Procedure

Having defined the explicit relations among the parameters of the periodic gait and formulated a differential equation describing the evolution of the virtual holonomic constraints along the target trajectory, the problem of finding a periodic cycle for the compass-gait biped can be stated.

Problem 1. *Find a, b such that $\bar{\phi}(c) = g$ and $\bar{\phi}'(c) = h/d$ with the algebraic relations* (5.5)*,* (5.6)*,* (5.9) *and* (5.13) *satisfied and $\bar{\phi}(\theta)$ being the solution to the differential equation* (5.19) *initiated at $\bar{\phi}(a) = e$ and $\bar{\phi}'(a) = f/b$.*

Finding a solution to the above problem requires searching for only two independent parameters a, b and solving a single second-order differential equation with the proposed initial conditions. This reduction in problem scope is one of the advantages of utilizing virtual holonomic constraints when formulating a search procedure.

The first step in deriving a structured procedure for solving Problem 1 is to redefine it as an optimization criteria. A reasonable choice is to minimize the value of the objective function

$$F = \left(\bar{\phi}(c) - g\right)^2 + \left(\bar{\phi}'(c) - \frac{h}{d}\right)^2 + \left(\bar{\theta}(T) - c\right)^2 \tag{5.20}$$

where $\bar{\phi}(c), \bar{\phi}'(c)$ are the solutions of (5.19) at impact, $\bar{\theta}(T)$ is the value of the explicit time-like variable θ used in (5.19) at time $t = T$, and c, d, g, h are defined by the algebraic relations (5.5), (5.6), (5.9) and (5.13). It is straightforward to verify that (5.20) achieves it's minimum value $F = 0$ when Problem 1 is solved. Applying a nonlinear optimization algorithm, such as `fminsearch` in MATLAB, to the objective function (5.20) with optimization variables a, b results in the following procedure for finding periodic gaits of the compass-gait biped.

Step 1 Specify an initial guess for the optimization variables a, b, such that the dynamics of the biped is in vicinity of a periodic gait.

Step 2 Compute the parameter d by solving the quadratic equation (5.13) and choosing either of the real solutions. If no real solutions exist, set $F = C$ where C is an arbitrarily large number and go to Step 7.

Step 3 Solve for the remaining parameters c, e, f, g, h using the algebraic relations (5.5), (5.6), (5.9) and (5.13).

Step 4 Compute the initial energy of the biped E_0 using either of the expressions from (5.12).

Step 5 Solve the differential equation (5.19) numerically over the interval $\theta \in [a, c]$ yielding the boundary values $\bar{\phi}(c), \bar{\phi}'(c)$ and $\bar{\theta}(T)$.

Step 6 Calculate the value of the objective function (5.20).

Step 7 If the value of O is greater than some specified tolerance, generate a new set of initial parameters a, b with the nonlinear optimization algorithm and return to Step 2. Otherwise, the search procedure is finished and the parameters a, b, c, d, e, f, g, h describes a periodic gait for the compass-gait biped.

The procedure above outlines a systematic approach to searching for periodic gaits of the biped and can be easily implemented in e.g. MATLAB.

5.4 MATLAB Implementation of Search Procedure

The following MATLAB function shows a possible implementation of the search procedure defined by Problem 1 by calculating the value of (5.20) based on a, b.

```
function value = objective_function(x, quad_solution_selector)
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    a = x(1);
    b = x(2);

    %% Solving for d, return large number if solution is imaginary
    [d, error_flag] = solving_E0_for_d(a,b,quad_solution_selector);
    if error_flag == 1
        value = Inf;
        return;
    end

    %% Solving for remaining parameters and assigning new names
    c = -a - 2 * psi;
    e = c;
    f = (b * p7 * cos((2 * a + 2 * psi)) - d * p9) / p8;
    g = a;
    h = (p7 ^ 2 / p8 / p11 * cos((2 * a + 2 * psi)) ^ 2 ...
        - p6 / p11) * b + (p8 * p10 - p9 * p7)...
        / p8 / p11 * cos((2 * a + 2 * psi)) * d;

    theta0 = a;
    thetaT = c;
    Dtheta0 = b;
    DthetaT = d;

    phi0 = e;
    phiT = g;
    Dphi0 = f/b;
    DphiT = h/d;

    %% Solving phi'' = f0 for inital conditions
    E0 = -f * cos(a - e) * p2 * b + f ^ 2 * p3 / 0.2e1 + ...
        p1 * b ^ 2 / 0.2e1 + cos(a) * p4 + p5 - cos(e) * p5 - p4;

    options = odeset('reltol',1e-9,'abstol',1e-6);
    [thetaout, yout] = ode45(@(theta,y) f0(theta,y,E0), ...
                              [theta0; thetaT], ...
                              [phi0; Dphi0], options);

    %% Returning objective function value
    value = (thetaout(end) - thetaT)^2 ...
            + (yout(end,1) - phiT)^2 + (yout(end,2) - DphiT)^2;
end
```

Implementation

The function `objective_function(x,quad_solution_selector)` calculates the value of the objective function F defined by (5.20), using $x = [a, b]^T$ as an input. The additional parameter `quad_solution_selector` $\in \{1, 2\}$ determines which real solution of the quadratic equation (5.13) is utilized in the search. This calculation is performed by the following function call

```
[d, error_flag] = solving_E0_for_d(a,b,quad_solution_selector);
```

where `error_flag` $= 1$ if the solution to $\sqrt{B^2 - 4AC}$ is imaginary. If that is the case, $F =$ `value` is assigned an arbitrarily large number and further computations are aborted. This ensures that values of a, b that are not feasible for the motion is quickly discarded, and the optimization algorithm continues the search in another direction. Having solved for d, all other parameters can be found using the previously defined algebraic relations.

The numerical integration of (5.19) is performed using `ode45` (see Chapter 3.6) by utilizing θ as the time parameter in the function call. This integration yields the vector `thetaout` and `yout` which respectively contain the values of θ and the corresponding $\phi(\theta), \phi'(\theta)$ for each iteration of the solver. The objective function value can then easily be calculated and returned to the optimization algorithm for evaluation.

A reasonable choice of optimization algorithm is the predefined MATLAB function `fminsearch` which has the following argument list:

```
xout = fminsearch(@(x) objective_function(x, quad_...),x0)
```

The optimization algorithm attempts to find the local minimizer `xout` of `objective_function` with the specified `quad_solution_selector`, starting the search at `x0`. An implementation of the complete search procedure is listed below for reference. The parameters c, d, e, f, g, h and E_0 are recalculated for convenience.

```
function [a,b,c,d,e,f,g,h,E0] = run_gait_search(...
                                    a0,b0,quad_solution_selector)

    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    run physical_parameters;
    fprintf('*********************************************\n');
    fprintf('SEARCHING FOR PERIODIC GAIT OF PASSIVE WALKER\n');
    fprintf('*********************************************\n');
    fprintf('Starting search with initial guess:\n');
    fprintf('q10 = %f\nDq10 = %f\n', a0,b0);
    fprintf('*********************************\n');

    x0 = [a0; b0];

    % Searching for intitial values q10 = a, Dq10 = b
    xout = fminsearch(@(x) objective_function(x, ...
                      quad_solution_selector),...
                      x0, optimset('Display','iter','TolFun',1e-10));

    a = xout(1);
    b = xout(2);

    % Finding corresponding values dq20 = e, Dq20 = f
    c = -a - 2 * psi;
    e = c;

    d = solving_E0_for_d(a,b,quad_solution_selector);
    f = (b * p7 * cos((2 * a + 2 * psi)) - d * p9) / p8;

    g = a;
    h = (p7 ^ 2 / p8 / p11 * cos((2 * a + 2 * psi)) ^ 2 - p6 / p11)...
        * b + (p8 * p10 - p9 * p7) / p8 / p11...
        * cos((2 * a + 2 * psi)) * d;

    E0 = -f * cos(a - e) * p2 * b + f ^ 2 * p3 / 0.2e1 + p1 *...
        b ^ 2 / 0.2e1 + cos(a) * p4 + p5 - cos(e) * p5 - p4;

    % Complete initial conditions for gait
    fprintf('*********************\n');
    fprintf('Optimization results:\n');
    fprintf('*********************\n');
    fprintf('a = %f\nb = %f\nc = %f\nd = %f\ne = %f\n',a, b, c, d, e);
    fprintf('f = %f\ng = %f\nh = %f\nE0 = %f\n', f, g, h, E0);
    fprintf('******************\n');
    fprintf('q10 = %f\nq20 = %f\nDq10 = %f\nDq20 = %f\n', a,e,b,f);
    fprintf('******************\n\n');

end
```

6 Stability of Periodic Gaits

This chapter will introduce a method for deriving an analytic expression for the Poincaré first return map for the hybrid dynamic system (3.35). Central to this formulation is the notion of transverse linearization of the dynamics of the system along its periodic orbit. A comprehensive treatment of this subject is given in [4, 6, 13, 17, 18, 19], and presented in this thesis by applying the theory to the specific case of the compass-gait biped.

6.1 Transverse Linearization

Transverse linearization is short-hand for the linearization of the transverse dynamics of a system. The essence of the method is to substitute the state vector $x(t) = [q, \dot{q}]^T \in \mathbb{R}^{2n}$ with a new set of coordinates $[\varphi(t), x_\perp(t)]^T$ where $x_\perp(t) \in \mathbb{R}^{2n-1}$ are coordinates transverse to the periodic trajectory, and $\varphi(t)$ is a scalar variable that travels along the orbit.

The transverse coordinates are defined such that $x_\perp(t) = 0$ corresponds to the periodic orbit $x_\star(t)$, meaning that $x_\perp(t)$ can be regarded as a measure of the deviation of $x(t)$ from the target orbit $x_\star(t)$. This can be exploited to define a family of Poincaré sections $\{S(t)\}_{t\in[0,T]}$ that move along the target orbit, such that for every point $\varphi(t)$ there is an associated member of the family present. A visualization of the family of hypersurfaces $S(t)$ is shown in Figure 6.1. Each Poincaré section $S(t)$ is a $(2n-1)$-dimensional hypersurface transversal to the orbit $x_\star(t)$ at each point, and the transverse coordinates $x_\perp(t)$ are coordinates defined on the section with origin in $x_\star(t)$. Due to the periodic nature of the target orbit, $S(0) = S(T)$ and the union of all the surfaces $S(t)$ cover a neighborhood in the vicinity of the orbit. This means that the transverse coordinates $x_\perp(t)$ are defined along the entire periodic orbit $x_\star(t)$, and the dynamics of $x_\perp(t)$ in a vicinity of the desired trajectory give rise to an auxiliary linear system on the form

$$\dot{z}(t) = A(t)z(t), \quad A(t+T) = A(t) \tag{6.1}$$

where $z(t) \in \mathbb{R}^{2n-1}$ is the vector from the tangent space $TS(t)$. The difficult problem of determining orbital stability of the periodic motion is thus reduced to the much simpler problem of determining asymptotic stability of the auxiliary linear system since $z(t) \to 0 \Rightarrow x_\perp(t) \to 0$ which is equivalent to $x(t) = x_\star(t)$.

The problem of computing such an auxiliary linear system for the compass-gait biped will be treated in detail in this chapter, starting with the transverse linearization of the continuous-time dynamics.

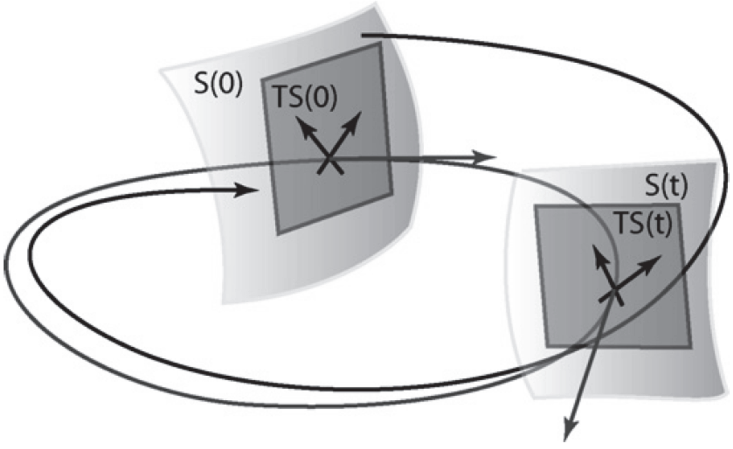

Figure 6.1: Visualization of moving Poincaré surfaces $S(\cdot)$ and transverse linearization $TS(\cdot)$ of periodic orbit (grey) and another solution converging to the orbit (black). Illustration © 2008 Elsevier. Reproduced, with permission, from Annual Reviews in Control, vol. 32, no. 2, A. Shiriaev, L. Freidovich, and I. Manchester, "Can we make a robot ballerina perform a pirouette? Orbital stabilization of periodic motions of underactuated mechanical systems," pp. 200 211, 2008.

6.2 Computing Linearization of Continuous Dynamics

Periodic gaits of the compass-gait biped are defined by the relation between the generalized coordinates q_1, q_2 imposed by the virtual holonomic constraint $\phi(\theta)$. Deviations from the periodic orbit

$$x_\star(t) = \begin{bmatrix} q_{1\star} \\ q_{2\star} \\ \dot{q}_{1\star} \\ \dot{q}_{2\star} \end{bmatrix} = \begin{bmatrix} \theta_\star \\ \phi(\theta_\star) \\ \dot{\theta}_\star \\ \phi'(\theta_\star)\dot{\theta}_\star \end{bmatrix} \tag{6.2}$$

can therefore be described by the error y between q_2 and $\phi(\theta)$ as

$$y = q_2 - \phi(\theta), \quad \theta = q_1 \tag{6.3}$$

with time derivatives

$$\begin{aligned} \dot{y} &= \dot{q}_2 - \phi'(\theta)\dot{\theta}, & \dot{\theta} &= \dot{q}_1 \\ \ddot{y} &= \ddot{q}_2 - \phi''(\theta)\dot{\theta}^2 - \phi'(\theta)\ddot{\theta}, & \ddot{\theta} &= \ddot{q}_1 \end{aligned} \tag{6.4}$$

Expressing $\ddot{\theta}, \ddot{y}$ on matrix form gives the following relation

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\phi'(\theta) & 1 \end{bmatrix} \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} - \begin{bmatrix} 0 \\ \phi''(\theta)\dot{\theta}^2 \end{bmatrix} \tag{6.5}$$

where $\ddot{q}_1, \ddot{q}_2$ are defined by (3.7). Substituting for the equations of motion yields two new differential equations for θ and y on the form

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -\phi'(\theta) & 1 \end{bmatrix} \left(-M^{-1}\left(C(q,\dot{q})\dot{q} + G(q)\right)\right)\Big|_{\substack{q_1=\theta,\, q_2=y+\phi(\theta) \\ \dot{q}_1=\dot{\theta},\, \dot{q}_2=\dot{y}+\phi'(\theta)\dot{\theta}}} - \begin{bmatrix} 0 \\ \phi''(\theta)\dot{\theta}^2 \end{bmatrix} \tag{6.6}$$

which can be restated as

$$\ddot{\theta} = \hat{g}(\theta, \dot{\theta}, y, \dot{y}) \tag{6.7}$$

$$\ddot{y} = h(\theta, \dot{\theta}, y, \dot{y}) \tag{6.8}$$

The differential equation (6.7) describes the time evolution of θ as a function of the error coordinate y, but does not take into account the already defined dynamics of the θ variable from (4.4) and (4.5). One way of incorporating these dynamics are by first forming an arbitrary linear combination of equations (4.4) and (4.5) with the weights

$$\begin{aligned} \mu_1(\theta) &= p_3 \\ \mu_2(\theta) &= p_2 \cos(\theta - \phi(\theta)) \end{aligned} \tag{6.9}$$

resulting in the new reduced dynamics of (3.7) on the form

$$\alpha_0(\theta)\ddot{\theta} + \beta_0(\theta)\dot{\theta}^2 + \gamma_0(\theta) = 0 \tag{6.10}$$

where

$$\begin{aligned} \alpha_0(\theta) &= p_1 p_3 - p_2 \cos^2(\theta - \phi(\theta)) \\ \beta_0(\theta) &= p_2 \sin(\theta - \phi(\theta)) \left(p_2 \cos(\theta - \phi(\theta)) - p_3(\phi'(\theta))^2\right) \\ \gamma_0(\theta) &= p_2 p_5 \cos(\theta - \phi(\theta)) \sin(\phi(\theta)) - p_3 p_4 \sin(\theta) \end{aligned} \tag{6.11}$$

The choice of weights $\mu_1(\theta), \mu_2(\theta)$ made in (6.9) has the benefit of removing the implicit dependence on $\phi''(\theta)$ from the parameters $\alpha_0(\theta), \beta_0(\theta), \gamma_0(\theta)$ and at the same time obtaining the shortest expressions. The dynamics (6.7) and (6.10) can be easily combined using simple multiplication and addition of the parameters (6.11), resulting in the system

$$\begin{aligned} \alpha_0(\theta)\ddot{\theta} + \beta_0(\theta)\dot{\theta}^2 + \gamma_0(\theta) &= g(\theta, \dot{\theta}, y, \dot{y}) \\ \ddot{y} &= h(\theta, \dot{\theta}, y, \dot{y}) \end{aligned} \tag{6.12}$$

where

$$g(\theta, \dot{\theta}, y, \dot{y}) = \alpha_0(\theta)\hat{g}(\theta, \dot{\theta}, y, \dot{y}) + \beta_0(\theta)\dot{\theta}^2 + \gamma_0(\theta) \tag{6.13}$$

The right-hand sides of (6.12) have the important property that they simultaneously vanish on the periodic orbit $\theta(t) = \theta_\star(t) \Rightarrow y = 0$ meaning

$$g(\theta, \dot{\theta}, 0, 0) = h(\theta, \dot{\theta}, 0, 0) = 0 \tag{6.14}$$

The mechanical system (3.7) can therefore be rewritten as (6.12) in a vicinity of the periodic orbit. The transverse coordinates of the system is then natural chosen as

$$x_\perp = \begin{bmatrix} I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0) \\ y \\ \dot{y} \end{bmatrix} \tag{6.15}$$

where $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$ is defined by (2.52). $I(\cdot)$ is added to the vector of transverse coordinates to ensure that $x_\perp \in \mathbb{R}^3$ span the Poincaré sections that are transversal to the periodic orbit $x_\star(t) \in \mathbb{R}^4$. The integral function qualifies as a transverse coordinate since, first, it preserves its zero value along the period orbit and, second, quantifies the distance orthogonal to the vector field along the orbit from any point in its vicinity (see [19, Property 2 p. 206]).

Computing the transverse linearization of the dynamics of (6.15) requires linearization of (6.12) in terms of the transverse coordinates. Using the fact that $g(\cdot), h(\cdot)$ are zero on the periodic orbit, as shown in (6.14), the first-order Taylor approximation in the transverse coordinates of the expressions around $x_\star(t)$ can be calculated as

$$g(\theta, \dot{\theta}, y, \dot{y}) = g_I(\theta_\star, \dot{\theta}_\star)\Delta I + g_y(\theta_\star, \dot{\theta}_\star)\Delta y + g_{\dot{y}}(\theta_\star, \dot{\theta}_\star)\Delta \dot{y} \tag{6.16}$$

where

$$\begin{aligned} g_y(\theta_\star, \dot{\theta}_\star) &= \left.\frac{\partial g(\theta, \dot{\theta}, y, \dot{y})}{\partial y}\right|_{\substack{\theta=\theta_\star,\, \dot{\theta}=\dot{\theta}_\star \\ y=0,\, \dot{y}=0}} \\ g_{\dot{y}}(\theta_\star, \dot{\theta}_\star) &= \left.\frac{\partial g(\theta, \dot{\theta}, y, \dot{y})}{\partial \dot{y}}\right|_{\substack{\theta=\theta_\star,\, \dot{\theta}=\dot{\theta}_\star \\ y=0,\, \dot{y}=0}} \end{aligned} \tag{6.17}$$

and $g_I(\theta_\star, \dot{\theta}_\star)$ is the directional derivative of g with respect to I taken in a direction n that is orthogonal to the trajectory given by (6.10). The directional derivative can be calculated as

$$\begin{aligned} g_I(\theta, \dot{\theta}) &= \frac{\partial g(\theta, \dot{\theta}, 0, 0)/\partial n}{\partial I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)/\partial n} = \frac{\nabla g(\theta, \dot{\theta}, 0, 0) \cdot n}{\nabla I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0) \cdot n} \\ g_I(\theta_\star, \dot{\theta}_\star) &= \frac{\left.\left(-\frac{\partial g(\theta, \dot{\theta}, 0, 0)}{\partial \theta}\ddot{\theta}_\star + \frac{\partial g(\theta, \dot{\theta}, 0, 0)}{\partial \dot{\theta}}\dot{\theta}_\star\right)\right|_{\theta=\theta_\star,\, \dot{\theta}=\dot{\theta}_\star}}{2\left(\dot{\theta}_\star^2 + \ddot{\theta}_\star^2\right)} \end{aligned} \tag{6.18}$$

where the last equality is due to the fact that $n = [-\ddot{\theta}_\star,\ \dot{\theta}_\star]^T$ is orthogonal to the flow $\varphi(\theta_0, \dot{\theta}_0, t_0, T) = [\dot{\theta}_\star,\ \ddot{\theta}_\star]^T$ of (6.10) at each point, and the partial derivatives of $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$ (see (2.59)) evaluated on the orbit are given by

$$\begin{aligned} \left.\frac{\partial I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)}{\partial \theta}\right|_{\substack{\theta=\theta_\star \\ \dot{\theta}=\dot{\theta}_\star}} &= -2\ddot{\theta}_\star \\ \left.\frac{\partial I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)}{\partial \dot{\theta}}\right|_{\substack{\theta=\theta_\star \\ \dot{\theta}=\dot{\theta}_\star}} &= 2\dot{\theta}_\star \end{aligned} \tag{6.19}$$

Analogous expressions are easily obtained for $h(\theta, \dot{\theta}, y, \dot{y})$ giving the linearization of $\ddot{y}$ transverse to the periodic orbit as

$$h(\theta, \dot{\theta}, y, \dot{y}) = h_I(\theta_\star, \dot{\theta}_\star)\Delta I + h_y(\theta_\star, \dot{\theta}_\star)\Delta y + h_{\dot{y}}(\theta_\star, \dot{\theta}_\star)\Delta \dot{y} \tag{6.20}$$

The validity of rewriting (6.12) as (6.16) and (6.20) follows from Hadamard's Lemma as discussed in [18, Appendix I-C].

The time derivative of the conserved quantity $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$ away from the periodic trajectory is stated in (2.55) where $W = g(\theta, \dot{\theta}, y, \dot{y})$ follows from (6.12). Since expressions for the time deriviatives of all the transverse coordinates in terms of perturbations transverse to the orbit are found, the transverse linearized dynamics can be stated as

$$\dot{\zeta} = A(t)\zeta, \quad A(t) = \begin{bmatrix} \frac{2\dot{\theta}_\star(g_I(\theta_\star,\dot{\theta}_\star)-\beta_0(\theta_\star))}{\alpha_0(\theta_\star)} & \frac{2\dot{\theta}_\star g_y(\theta_\star,\dot{\theta}_\star))}{\alpha_0(\theta_\star)} & \frac{2\dot{\theta}_\star g_{\dot{y}}(\theta_\star,\dot{\theta}_\star))}{\alpha_0(\theta_\star)} \\ 0 & 0 & 1 \\ h_I(\theta_\star, \dot{\theta}_\star)) & h_y(\theta_\star, \dot{\theta}_\star)) & h_{\dot{y}}(\theta_\star, \dot{\theta}_\star)) \end{bmatrix} \tag{6.21}$$

where $\alpha_0(\theta_\star), \beta_0(\theta_\star), \gamma_0(\theta_\star)$ are given by (6.10), and $\zeta = [\Delta I(\cdot), \Delta y, \Delta \dot{y}]^T$ gives the linear parts of the deviations from zero for the components of $x_\perp = [I(\cdot), y, \dot{y}]^T$. All the elements of $A(t)$ from (6.21) can be computed analytically in a straightforward way, yielding an accurate description of the transverse dynamics without using numerical approximations. The rather long expressions are not presented explicit in this text in order to not waste space, but a Maple worksheet showing all the calculations can be found in Appendix B.5.

6.3 Computing Linearization of Discrete Dynamics

The discrete dynamics of the compass-gait biped consists of the update laws (3.32) and (3.33), which can be restated as a nonlinear update map

$$\begin{bmatrix} q^+ \\ \dot{q}^+ \end{bmatrix} = F(q^-, \dot{q}^-) = \begin{bmatrix} Pq^- \\ P_q(q^-)\dot{q}^- \end{bmatrix} \tag{6.22}$$

where $F(q^-, \dot{q}^-) \in \mathbb{R}^4$, and the $-, +$ superscripts indicate the time just before and just after impact, respectively. The first step in finding a transverse linearization of the discrete dynamics is computing the Jacobian dF of this mapping for the periodic orbit $x_\star(t)$ under study.

$$\begin{aligned} dF_{(x_\star(t=T))} &= \frac{\partial F(q^-, \dot{q}^-)}{\partial\, [q^-, \dot{q}^-]} = \left.\begin{bmatrix} P & 0_{2\times 2} \\ \frac{\partial P_q(q^-)}{\partial q^-}\dot{q}^- & P_q(q^-) \end{bmatrix}\right|_{\substack{q^- = q_\star^- \\ \dot{q}^- = \dot{q}_\star^-}} \\ \delta x^+ &= dF_{(x_\star(t=T))}\, \delta x^- \end{aligned} \tag{6.23}$$

The subscript $(x_\star(t = T))$ indicates that the Jacobian is computed using the states $x_\star$ at time of impact $t = T$. The transformation (6.23) acts on the vectors from the plane tangent to $\Gamma^- \subset \mathbb{R}^4$, the embedding of the impact surface $\mathcal{S} \subset \mathbb{R}^2$, defined by (3.31), into $\mathbb{R}^4$. This tangential plane is given by

$$T\Gamma^- = \{\delta x^- \in \mathbb{R}^4 : (n^-)^T \delta x^- = 0\} \tag{6.24}$$

where n^- is the gradient of the switching surface $\mathcal{S}$ just before impact

$$n^- = \begin{bmatrix} -\sin\left(q_1^- + \psi\right) \\ \sin\left(q_2^- + \psi\right) \\ 0 \\ 0 \end{bmatrix} \tag{6.25}$$

The transformation (6.23) maps vectors $\delta x^- \in T\Gamma^-$ into the plane

$$T\Gamma^+ = \{\delta x^+ \in \mathbb{R}^4 : (n^+)^T \delta x^+ = 0\} \tag{6.26}$$

where n^+ is the gradient of the switching surface $\mathcal{S}$ just after impact, which can easily be computed by making the substitution $q = q^+ = Pq^-$ into $H(q)$ defined in (3.31) and then calculating the gradient as in the case of n^-, leading to

$$n^+ = \begin{bmatrix} \sin\left(q_1^- + \psi\right) \\ -\sin\left(q_2^- + \psi\right) \\ 0 \\ 0 \end{bmatrix} \tag{6.27}$$

The linear operator $dF : T\Gamma^- \to T\Gamma^+$ relates the state of the biped just before impact δx^- to the state right after impact δx^+, but the linearized continuous-time dynamics is described in terms of $\zeta(t) = [\Delta I, \Delta y, \Delta \dot{y}]$. Hence, a linerization of the relations

$$\begin{aligned} y &= q_2 - \phi(q_1) \\ \dot{y} &= \dot{q}_2 - \phi'(q_1)\dot{q}_1 \end{aligned} \tag{6.28}$$

and $I = I(q_1, \dot{q}_1)$ must be obtained in a vicinity of the desired trajectory. This linearization can be regarded as a mapping between the linear parts of increment of the transversed coordinates in $\mathbb{R}^3$ and the linear parts of generalized coordinates in $\mathbb{R}^4$ on the form

$$\begin{bmatrix} \Delta I \\ \Delta y \\ \Delta \dot{y} \end{bmatrix} = L(t) \begin{bmatrix} \Delta q \\ \Delta \dot{q} \end{bmatrix} \tag{6.29}$$

where $L(t)$ is computed as

$$L(t) = \left. \begin{bmatrix} \frac{\partial I}{\partial [q,\dot{q}]} \\ \frac{\partial y}{\partial [q,\dot{q}]} \\ \frac{\partial \dot{y}}{\partial [q,\dot{q}]} \end{bmatrix} \right|_{\substack{q_1=\theta_\star(t) \\ \dot{q}_1=\dot{\theta}_\star(t)}} = \begin{bmatrix} -2\ddot{\theta}_\star(t) & 0 & 2\dot{\theta}_\star(t) & 0 \\ -\phi'(\theta_\star(t)) & 1 & 0 & 0 \\ -\phi''(\theta_\star(t))\dot{\theta}_\star(t) & 0 & -\phi'(\theta_\star(t)) & 1 \end{bmatrix} \tag{6.30}$$

using the result from (6.19) to calculate $\frac{\partial I}{\partial [q,\dot{q}]}$. In order to derive an inverse transformation from $\Delta x_\perp$ to Δx, the matrix $L(t) \in \mathbb{R}^{3\times 4}$ must be augmented to make it invertible. An important property of the moving Poincaré sections $S(t)$ shown in Figure 6.1 is that they are transverse to the periodic orbit $x_\star(t)$ at each point, this also applies to the planes $TS(t)$ which are tangent to $S(t)$ at the intersection with $x_\star(t)$. Since $\zeta(t) \in TS(t)$, the transverse coordinates are always orthogonal to the flow of the periodic solution $x_\star(t)$ which can be represented by the vector

$$m(t) = \begin{bmatrix} \dot{q}_{1\star}(t) \\ \dot{q}_{2\star}(t) \\ \ddot{q}_{1\star}(t) \\ \ddot{q}_{2\star}(t) \end{bmatrix} = \begin{bmatrix} \dot{\theta}_\star(t) \\ \phi'(\theta_\star)\dot{\theta}_\star(t) \\ \ddot{\theta}_\star(t) \\ \phi''(\theta_\star)\dot{\theta}_\star^2(t) + \phi'(\theta_\star)\ddot{\theta}_\star(t) \end{bmatrix} \tag{6.31}$$

The orthogonality of $m(t)$ and $x_\perp$ is preserved by the transformation (6.30) such that the relation

$$m(t)^T \cdot \Delta x(t) = 0 \tag{6.32}$$

always hold along the orbit. The complete transformation between the linearized transverse coordinates $\zeta(t)$ and the linearized generalized coordinates

$\Delta x(t)$ can then be stated as

$$\begin{aligned}
\mathbb{R}^3 \cong TS(t) \ni \zeta(t) &= \begin{bmatrix} \Delta I(t) \\ \Delta y(t) \\ \Delta \dot{y}(t) \end{bmatrix} = L(t) \begin{bmatrix} \Delta q_1(t) \\ \Delta q_2(t) \\ \Delta \dot{q}_1(t) \\ \Delta \dot{q}_2(t) \end{bmatrix} \\
\mathbb{R}^4 \supset TS(t) \ni \Delta x(t) &= \begin{bmatrix} \Delta q_1(t) \\ \Delta q_2(t) \\ \Delta \dot{q}_1(t) \\ \Delta \dot{q}_2(t) \end{bmatrix} = \begin{bmatrix} L(t) \\ m(t)^T \end{bmatrix}^{-1} \begin{bmatrix} \Delta I(t) \\ \Delta y(t) \\ \Delta \dot{y}(t) \\ 0 \end{bmatrix}
\end{aligned} \tag{6.33}$$

Unfortunately, the tangent planes $TS(t)$ defined as

$$TS(t) = \{\Delta x(t) \in \mathbb{R}^4 : \, m(t)^T \Delta x(t) = 0\} \tag{6.34}$$

at $t = T^-$ and $t = T^+$ are different from the tangent planes $T\Gamma^-$ and $T\Gamma^+$ defined by (6.24) and (6.26). This means that a nontrivial projection of $TS(t)$ onto the appropriate surface $T\Gamma$ is needed in order to compute the updated coordinates ζ^+ using the mapping (6.23), which acts on δx^- and not Δx^-. Deriving these projection operators are beyond the scope of this text, and the expressions will be stated without proof[7]. The projection operator that maps the transverse coordinates before impacts $\zeta^- \in TS(T^-)$ to the state vector $\delta x^- \in T\Gamma^-$ is given by

$$\begin{aligned}
\delta x^- &= P^-_{n^-} \zeta^- \\
P^-_{n^-} &= \left(I_4 - \frac{m(T^-)(n^-)^T}{(m(T^-))^T n^-} \right) \begin{bmatrix} L(T^-) \\ m(T^-)^T \end{bmatrix}^{-1} \begin{bmatrix} I_3 \\ 0_{1\times 3} \end{bmatrix}
\end{aligned} \tag{6.35}$$

where I_i is the identity matrix of dimension $(i \times i)$. The corresponding mapping from the updated state vector $\delta x^+ \in \Gamma^+$ back to the transverse coordinates after impact $\zeta^+ \in TS(T^+)$ is defined by the simpler expression

$$\begin{aligned}
\delta x^+ &= P^+_{n^+} \zeta^+ \\
P^+_{n^+} &= L(T^+) \left(I_4 - \frac{m(T^+)(n^+)^T}{(m(T^+))^T n^+} \right)
\end{aligned} \tag{6.36}$$

[7] An approach for deriving expressions for the projection operators for a more complex hybrid system is outlined in [4, p. 10170]

In the case of the compass-gait biped the projection operators $P^-_{n^-}, P^+_{n^+}$ are calculated as

$$P^-_{n^-} = \begin{bmatrix} 0 & \frac{\dot q_{1\star}(T^-)\sin\left(q_{2\star}(T^-)+\psi\right)}{\dot q_{1\star}(T^-)\sin(q_{1\star}(T^-)+\psi)-\dot q_{2\star}(T^-)\sin(q_{2\star}(T^-)+\psi)} & 0 \\ 0 & \frac{\dot q_{1\star}(T^-)\sin\left(q_{1\star}(T^-)+\psi\right)}{\dot q_{1\star}(T^-)\sin(q_{1\star}(T^-)+\psi)-\dot q_{2\star}(T^-)\sin(q_{2\star}(T^-)+\psi)} & 0 \\ \frac{1}{2\dot q_{1\star}(T^-)} & \frac{\ddot q_{1\star}(T^-)\sin\left(q_{2\star}(T^-)+\psi\right)}{\dot q_{1\star}(T^-)\sin(q_{1\star}(T^-)+\psi)-\dot q_{2\star}(T^-)\sin(q_{2\star}(T^-)+\psi)} & 0 \\ \frac{\dot q_{2\star}(T^-)}{2\dot q_{1\star}(T^-)^2} & \frac{\ddot q_{2\star}(T^-)\sin\left(q_{2\star}(T^-)+\psi\right)}{\dot q_{1\star}(T^-)\sin(q_{1\star}(T^-)+\psi)-\dot q_{2\star}(T^-)\sin(q_{2\star}(T^-)+\psi)} & 1 \end{bmatrix} \tag{6.37}$$

$$P^+_{n^+} = \begin{bmatrix} -2\ddot q_{1\star}(T^+) & 0 & 2\dot q_{1\star}(T^+) & 0 \\ -\frac{\dot q_{2\star}(T^+)}{\dot q_{1\star}(T^+)} & 1 & 0 & 0 \\ \frac{\dot q_{2\star}(T^+)\ddot q_{1\star}(T^+)-\dot q_{1\star}(T^+)\ddot q_{2\star}(T^+)}{\dot q_{1\star}(T^+)^2} & 0 & -\frac{\dot q_{2\star}(T^+)}{\dot q_{1\star}(T^+)} & 1 \end{bmatrix}$$

where $t = T^-$ and $t = T^+$ refer to the time instances just before and just after impact, respectively, with the switching surface $\mathcal{S}$.

Defining the projected linearized update law $\left(d^{TS}F\right) : TS(T^-) \to TS(T^+)$ using (6.23), (6.35) and (6.36) as

$$\left(d^{TS}F\right)_{(x_\star(t=T))} = P^+_{n^+}\, dF_{(x_\star(t=T))}\, P^-_{n^-} \tag{6.38}$$

allows for stating the transverse linearized discrete dynamics that maps the linearized transverse coordinates just before and just after impact on the convenient form

$$\zeta^+ = \left(d^{TS}F\right)_{(x_\star(t=T))} \zeta^- \tag{6.39}$$

The expressions for dF and $\left(d^{TS}F\right)$ are rather long and are therefore not presented in this text for convenience, but a Maple worksheet showing all the calculations involved can be found in Appendix B.6.

6.4 Verifying Stability Using Auxiliary Linear System

The transverse linearization for the system (3.35) is a T-periodic linear auxiliary hybrid system defined by the continuous and discrete dynamics given by (6.21) and (6.39) which can be stated as

$$\begin{array}{lll} \dot\zeta(t) = A(t \bmod T)\zeta(t) & \text{for} & 0 < t < T \\ \zeta^+ = \left(d^{TS}F\right)_{(x_\star(t=T))} \zeta^- & \text{for} & t = T \end{array} \tag{6.40}$$

where $\zeta(0) = \zeta^+$ after each impact at time $t = T$. The solution of the auxiliary linear system (6.40) at the end of a cycle $t = T$, initiated at $\zeta(0)$ is given by the solutions of the discrete-time system

$$\zeta^+ = \Xi(x_\star^-, x_\star^+)\zeta(0) \tag{6.41}$$

where

$$\begin{aligned} \Xi(x_\star^-, x_\star^+) &= \left(P_{n^+}^+ \, dF_{(x_\star(t=T))} \, P_{n^-}^-\right) \Phi(T) \\ \dot{\Phi}(t) &= A(t)\Phi(t), \quad \Phi(0) = I_3 \end{aligned} \tag{6.42}$$

The matrix $\Phi(t)$ is called the state transition matrix of (6.21) (see [2, Definition 4.28]) with the important property

$$\Phi(T)\zeta(0) = \zeta(T) \tag{6.43}$$

where $\zeta(T)$ is the deflection of the linear parts of the transverse coordinates at impact. Interestingly, the eigenvalues of the transition matrix $\Xi(x_\star^-, x_\star^+)$ are equal to the eigenvalues of the 3-dimensional first return Poincaré map for the impulsive system (3.35). Determining orbital asymptotically stability for the periodic orbit $x_\star(t)$ is therefore reduced to checking if the system (6.40) is exponentially stable, which is the case when the eigenvalues of $\Xi(x_\star^-, x_\star^+)$ are strictly inside the unit circle. The procedure for computing the eigenvalues can be summarized as follows:

Step 1 Determine the parameters $[a, b, c, d, e, f, g, h, T]$ of the periodic gait using the approach described in Chapter 5.

Step 2 Find the values for $\phi(\theta_\star), \phi'(\theta_\star)$ for $\theta_\star \in [a, c]$ by solving (5.19) with initial conditions $e, f/b$.

Step 3 Find the values for $\theta_\star(t), \dot{\theta}_\star(t)$ for $t \in [0, T]$ by solving (6.10) with initial conditions a, b.

Step 4 Solve (6.21) using the three different initial conditions $\zeta_1(0) = [1, 0, 0]^T$, $\zeta_2(0) = [0, 1, 0]^T$, $\zeta_3(0) = [0, 0, 1]^T$ resulting in the three vectors $\zeta_1(T)$, $\zeta_2(T)$ and $\zeta_3(T)$.

Step 5 Form the state transition matrix at impact $\Phi(T) = [\zeta_1(T), \zeta_2(T), \zeta_3(T)]$ and then calculate the numerical value of $\Xi(x_\star^-, x_\star^+)$ by substituting in the appropriate parameters.

Step 6 Determine the eigenvalues of $\Xi(x_\star^-, x_\star^+) \in \mathbb{R}^3$ using any method (e.g `eig` function in MATLAB) and check stability.

A straightforward MATLAB implementation of this procedure can be found in Appendix C.4 and will not be discussed in detail here. One important detail is the use of the numerical solver `ode45` to yield numerical values for Step 2 and Step 3, which are then employed when solving Step 4 and Step 5 by interpolating between the data in the appropriate vectors. The eigenvalues are determined by calling the MATLAB function `eig` on the resulting transition matrix $\Xi(x_\star^-, x_\star^+)$.

7 Numerical Results of Gait Search

The procedure outlined in Chapter 5 for finding periodic gaits of the compass-gait biped, can be implemented in MATLAB to yield numeric values for the gait parameters a, b, c, d, e, f, g, h given a predefined slope angle ψ. In this chapter, results from such a search is presented and discussed. In addition, the stability of the periodic gaits are assessed using the notion of transverse linearization presented in Chapter 6.

7.1 Physical Parameters of the Compass-gait Biped

The physical parameters of the compass-gait biped can be freely defined, since the robot under study in this thesis is not based on a physical machine. However, for comparison purposes, the parameters assigned to the biped are similar to those found in e.g. [5, 8, 21], and are given in Table 1. All parameter values are provided in appropriate SI-units. The parameters a, b, g are not associated with the parameters of the periodic gait that are to be determined by the search procedure, and will not be used anywhere in the rest of the chapter.

Mass	Leg length	Gravity
$m_1 = 5.0$	$a = 0.5$	
$m_2 = 5.0$	$b = 0.5$	$g = 9.81$
$m_H = 10.0$	$l = a + b = 1.0$	

Table 1: Physical parameters of the compass-gait biped used in the search for periodic gaits.

7.2 Stable Periodic Gait for $\psi = 3.00$ deg

The existence of symmetric gaits of the compass-gait biped for slopes with angles $\psi \in [0, \approx 4.4]$ degrees is demonstrated in [8] using conventional search techniques. The incline angle $\psi = 3.00$ degrees was chosen in this thesis to exploit the knowledge that periodic gaits exist for this slope, allowing for verification that the search procedure presented in Chapter 5 yields correct results.

The search procedure requires three additional choices in addition to the value of ψ; the initial guesses $a = q_1(0), b = \dot{q}_1(0)$, and the choice of real solution for the quadratic equation (5.13) yielding the parameter $d = \dot{q}_1(T^-)$. Choosing a reasonable initial guess for the gait parameters a, b is a difficult task without prior knowledge of the dynamics of the biped, but the paper [5] gives the following initial values for a stable periodic gait

$$\begin{aligned} q_{1\star}(0) &= 0.21689, & q_{2\star}(0) &= -0.31708 \\ \dot{q}_{1\star}(0) &= -1.08428, & \dot{q}_{2\star}(0) &= -0.39728 \end{aligned} \tag{7.1}$$

where the slope angle was set to $\psi = 2.87$ degrees. The initial guess for the periodic gait parameters a, b in the search procedure was therefore set at

$$a_0 = 0.20, \quad b_0 = -1.0 \tag{7.2}$$

since it is reasonable to assume that a small variation in ψ results in corresponding small variations in the initial values of the gait. The choice of real solution of d was made as

$$d = \frac{-B(a,b) - \sqrt{B(a,b)^2 - 4A(a,b)C(a,b))}}{2A(a,b)} \tag{7.3}$$

where the coefficients A, B, C are given in (5.14), this corresponds to the parameter value `quad_solution_selector`=2 in the MATLAB-script described in Chapter 5.4.

Initiating the search procedure with these parameters yielded convergence of the `fminsearch` algorithm after 48 iterations, resulting in the following gait parameter values.

$$\begin{aligned} a &= 0.218669, & e &= -0.323389 \\ b &= -1.092386, & f &= -0.377377 \\ c &= -0.323389, & g &= 0.218669 \\ d &= -1.494206, & h &= -1.805654 \end{aligned} \tag{7.4}$$

where the initial configuration of the biped is given by

$$\begin{aligned} q_{1\star}(0) &= 0.218669, & q_{2\star}(0) &= -0.323389 \\ \dot{q}_{1\star}(0) &= -1.092386, & \dot{q}_{2\star}(0) &= -0.377377 \end{aligned} \tag{7.5}$$

The stability of the periodic gait defined by (7.4) can be assessed by calculating the eigenvalues of the Poincaré first-return map by using transverse linearization as shown in Chapter 6. The corresponding eigenvalues of the transition matrix $\Xi(x_\star^-, x_\star^+)$ are computed using the MATLAB-script in Appendix C.4, yielding

$$\begin{aligned} \lambda_1 &= 0.123 \\ \lambda_2 &= 0.127 - 0.586i \\ \lambda_3 &= 0.127 + 0.586i \end{aligned} \tag{7.6}$$

where λ_2, λ_3 are complex conjugates. These eigenvalues are plotted in the complex plane in Figure 7.1, and can be seen to all lie inside the unit circle indicating that the periodic gait is asymptotically orbitally stable.

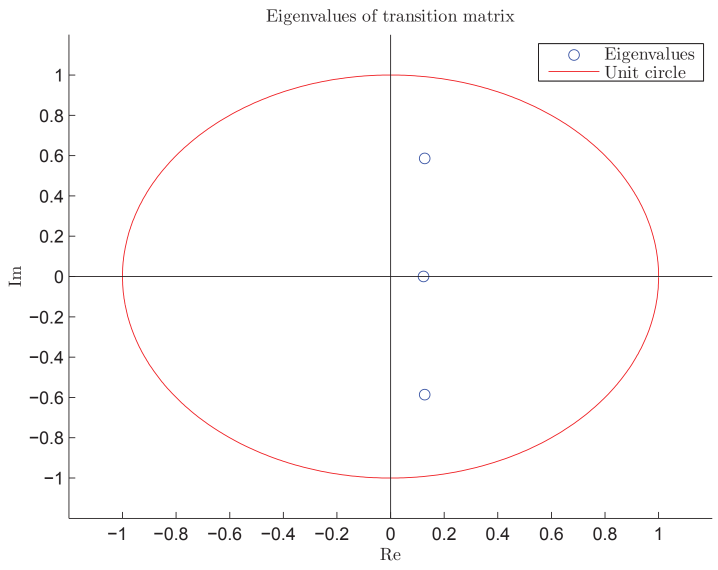

Figure 7.1: Eigenvalues of the transition matrix $\Xi(x^-_\star, x^+_\star)$ for the stable periodic gait with $\psi = 3.00$ degrees.

Simulating stable gait without perturbation

Simulating the compass-gait biped using the MATLAB-script shown in Chapter 3.6 with initial conditions given by (7.5) results in the time evolution of $q, \dot{q}$ shown in Figure 7.2. The generalized coordinates q_1, q_2 describing the angles of the stance and swing leg, respectively, are seen to be continuous during impact with ground. This is in agreement with the assumption that the impact forces are impulsive, made when modeling the biped in Chapter 3. The velocities $\dot{q}_1, \dot{q}_2$ however are discontinuous at impact due to the non-trivial impact map describing the relationship between $\dot{q}^-$ and $\dot{q}^+$. The discontinuities are a direct consequence of the impact forces the biped experiences during heel-strike with the slope, and without such forces the biped would simplify fall through the floor during the motion. Another interesting feature of Figure 7.2 is the switch between stance leg and swing leg during impact. The switch can easily be seen in the $q(t)$ - plot where the angles q_1, q_2 switch value during impact, giving rise to the discontinuous coloring of the plot.

Figure 7.3 depicts the passive limit cycle that describe the periodic gait in the phase plane. The actual trajectory evolves in a four-dimensional space, so the phase portrait shown only depicts one of the legs as it switches from stance leg (red) to swing leg (green). Note the two points of discontinuity resulting from the ground impacts, causing a jump in velocities $\dot{q}$. Another

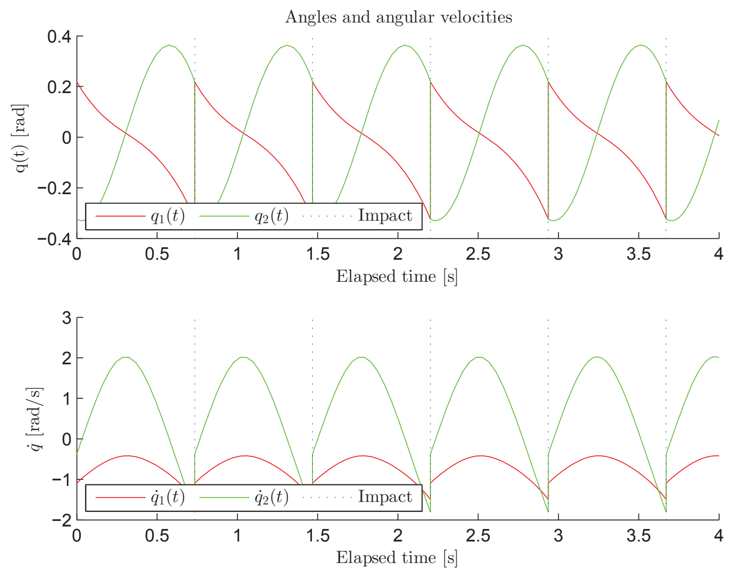

Figure 7.2: Time evolution of $q, \dot{q}$ *for unperturbed stable gait at* $\psi = 3.00$.

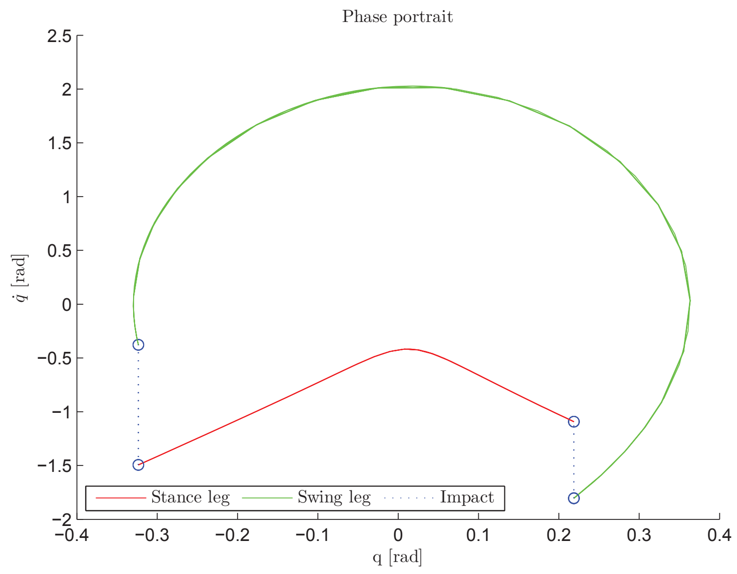

Figure 7.3: Phase portrait of the unperturbed stable gait at $\psi = 3.00$.

indication of the stability of the gait is shown in Figure 7.4, which depicts the time between consecutive steps of the gait after each step of the cycle. The half-step period can be seen to be stable with $T \approx 0.735$ seconds between each impact.

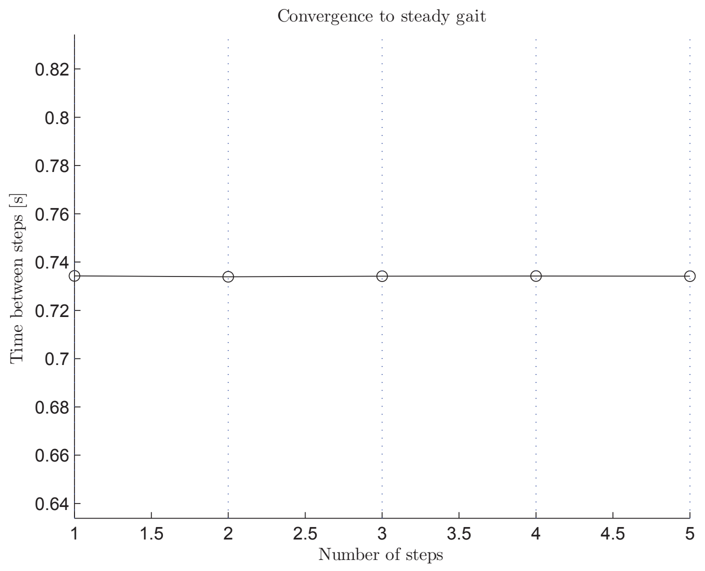

Figure 7.4: Time between consecutive steps for the unperturbed stable gait at $\psi = 3.00$.

Simulating stable gait with perturbation

The stable properties of the gait described by (7.4) can be directly observed by introducing a small perturbation ϵ to the initial conditions of the gait (7.5), giving the new initial conditions

$$\begin{aligned} &q(0) = q_\star(0) + \epsilon = [0.218669, -0.323389]^T + [0.01, 0.01]^T \\ &\dot{q}(0) = \dot{q}_\star(0) + \epsilon = [-1.092386, -0.377377]^T + [0.01, 0.01]^T \\ &q_1(0) = 0.228669, \qquad q_2(0) = -0.313389 \\ &\dot{q}_1(0) = -1.082386, \quad \dot{q}_2(0) = -0.367377 \end{aligned} \tag{7.7}$$

The phase portrait for the perturbed gait is shown in Figure 7.5, where the outermost cycle corresponds to the initial conditions $q(0), \dot{q}(0)$. Looking at the time evolution of $q, \dot{q}$ in Figure 7.6 it is clear that the swing leg is most affected by the perturbation to the initial conditions, and the amplitude of the swing can be seen oscillating around the stable value that is observed in

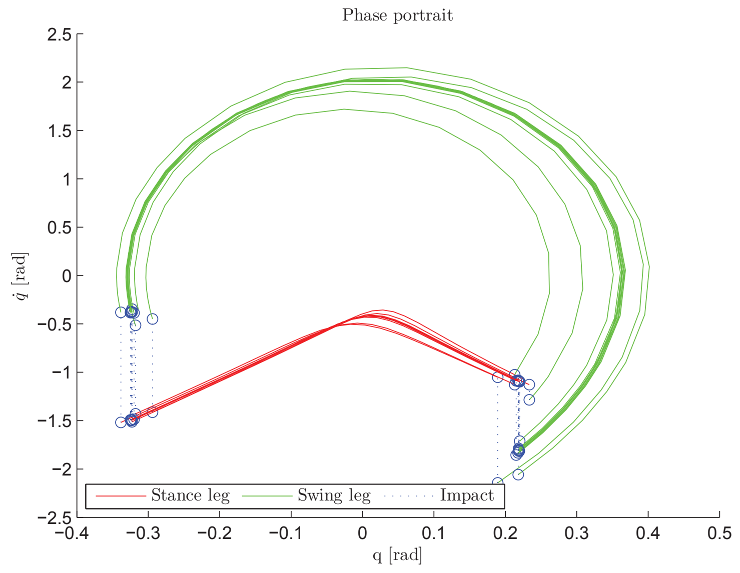

Figure 7.5: Phase portrait of the stable gait with perturbation $\epsilon = [0.01, 0.01]^T$ *in initial conditions at* $\psi = 3.00$.

Figure 7.2 for the unperturbed gait. After approximately 7 seconds the perturbed gait has converged[8] to the steady gait described by (5.2). Examining the time between consecutive steps in Figure 7.7, the convergence to a stable limit cycle occurs after approximately 10 steps.

7.3 Unstable Periodic Gait for $\psi = 3.00$ deg

Initiating the search procedure with the initial guess (7.2), but choosing the other real solution for d given by

$$d = \frac{-B(a,b) + \sqrt{B(a,b)^2 - 4A(a,b)C(a,b))}}{2A(a,b)}, \tag{7.8}$$

which corresponds to the parameter `quad_solution_selector`=1 in the MATLAB-script, yields convergence of the `fminserach` algorithm after 55 iterations. The resulting periodic gait is described by the gait parameters

$$\begin{aligned} a &= 0.204676, & e &= -0.309396 \\ b &= -1.210571, & f &= -0.712484 \\ c &= -0.309396, & g &= 0.204676 \\ d &= -1.395725, & h &= -0.087282 \end{aligned} \tag{7.9}$$

[8]Convergence should be understood to mean that the simulated trajectory have become close to the steady gait with a certain accuracy.

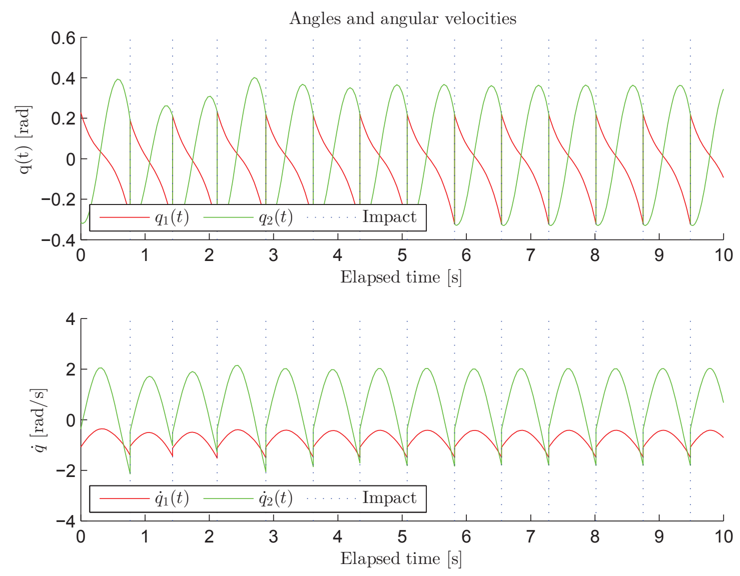

Figure 7.6: Time evolution of $q, \dot{q}$ *for stable gait with perturbation* $\epsilon = [0.01, 0.01]^T$ *in initial conditions at* $\psi = 3.00$.

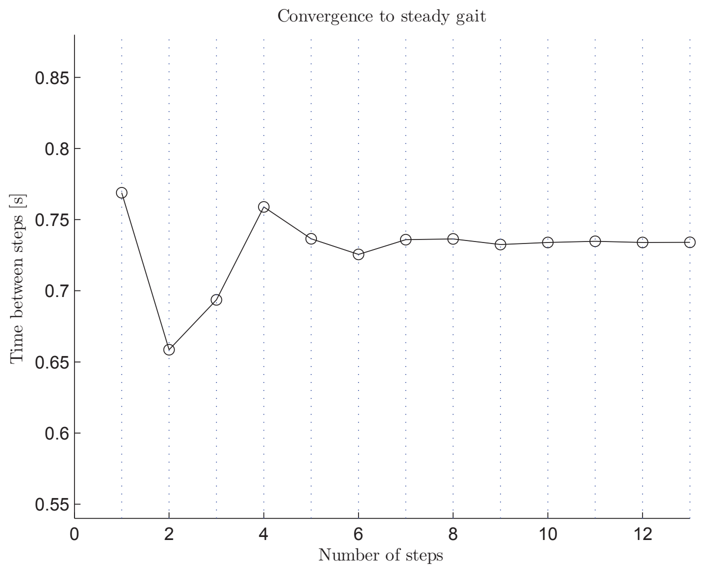

Figure 7.7: Time between consecutive steps for stable gait with perturbation $\epsilon = [0.01, 0.01]^T$ *in initial conditions at* $\psi = 3.00$.

where the initial conditions are given by

$$\begin{aligned} q_{1\star}(0) &= 0.204676, & q_{2\star}(0) &= -0.309396 \\ \dot{q}_{1\star}(0) &= -1.210571, & \dot{q}_{2\star}(0) &= -0.712484 \end{aligned} \tag{7.10}$$

The stability of the gait is assessed, as in the previous case, by calculating the eigenvalues of the new transition matrix $\Xi(x_\star^-, x_\star^+)$ using transverse linearization, which yields the eigenvalues

$$\begin{aligned} \lambda_1 &= 0.066 \\ \lambda_2 &= 0.408 \\ \lambda_3 &= 4.857 \end{aligned} \tag{7.11}$$

By examining the numerical values of $\lambda_1, \lambda_2, \lambda_3$, as well as checking the complex plot of the eigenvalues in Figure 7.8, it is clear that λ_3 lies outside the unit circle. This means that the periodic gait described by the parameters

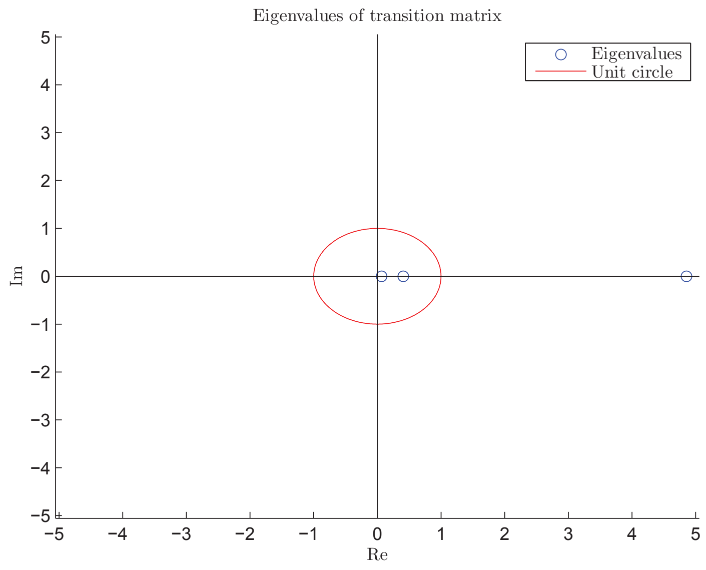

Figure 7.8: Eigenvalues of the transition matrix $\Xi(x_\star^-, x_\star^+)$ for the unstable periodic gait with $\psi = 3.00$ degrees.

(7.9) is an unstable limit cycle, and all trajectories initiated in a neighborhood of the periodic orbit will tend away from it as $t \to \infty$.

Simulating the gait in MATLAB with initial conditions (7.10) demonstrates the unstable nature of the limit cycle. The phase portrait in Figure 7.9 shows divergence from the unstable limit cycle (innermost cycle) and the convergence to the stable limit cycle described by (7.4). The divergence

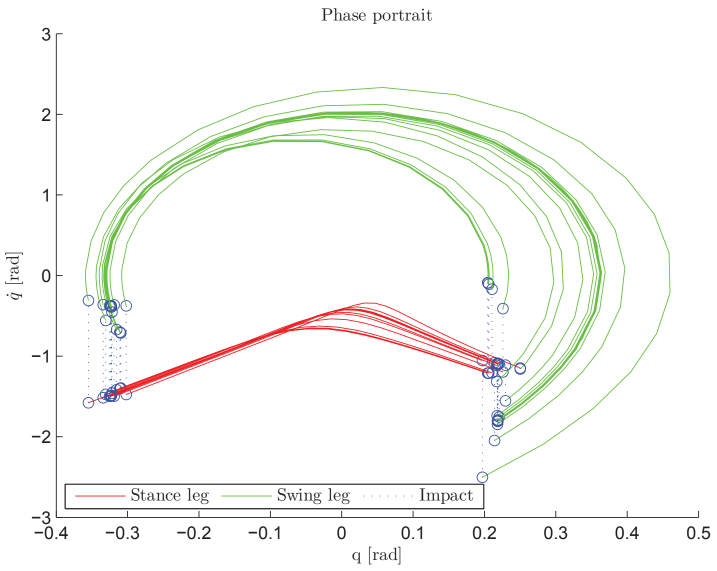

Figure 7.9: Phase portrait of the unstable gait at $\psi = 3.00$.

from the cycle occurs even without any explicitly defined perturbation of the initial values (7.10) due to the inaccuracies in the numeric solver `ode45`. These small numerical errors are unnoticed in stable periodic gaits due to the simple fact that the stable nature of the cycle causes all trajectories close to the orbit to tend to it, this is of course not the case for unstable orbits which consequently causes the observed divergence.

Examining the time between consecutive steps of the gait shown in Figure 7.10 reveals an exponential divergence from the gait from step 2 to step 7, at which point the convergence to the stable limit cycle begins. The divergence of the gait followed by convergence to a stable gait is also clearly visible in the time evolution of $q, \dot{q}$ depicted in Figure 7.11, where convergence occurs after approximately 8 seconds.

7.4 Bifurcation Affecting Periodic Gait for $\psi = 5.00$ deg

The stable and unstable periodic gaits described by (7.4) and (7.9), respectively, are both examples of symmetric gait cycles. A symmetric gait cycle has the property that the time intervals between two consecutive impacts are identical, such that the relation

$$T_p = T_i + T_{i+1}, \qquad T_i = T_{i+1} = \frac{T_p}{2} \tag{7.12}$$

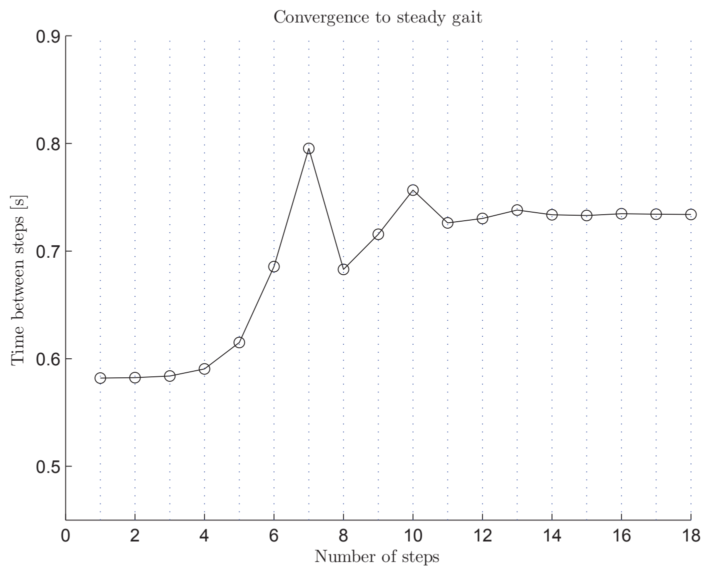

Figure 7.10: Time between consecutive steps for the unstable gait at $\psi = 3.00$.

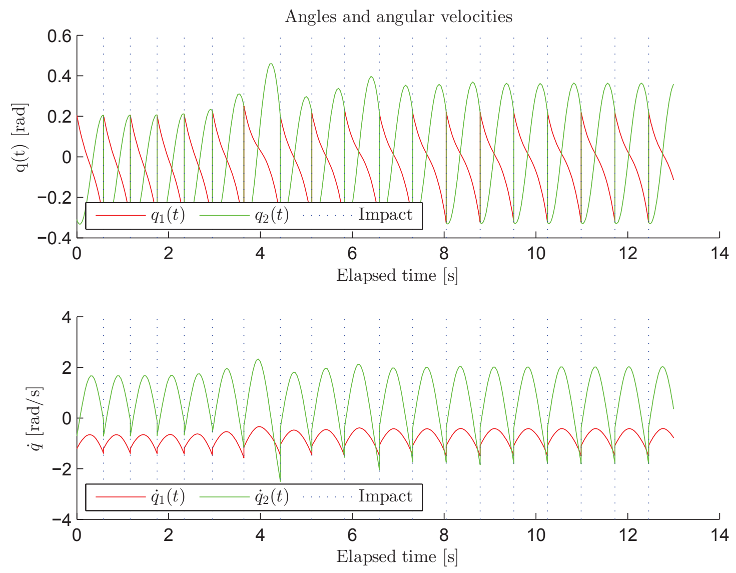

Figure 7.11: Time evolution of $q, \dot{q}$ *for unstable gait at* $\psi = 3.00$.

holds at the steady gait, where T_p is the period of the gait, and T_i, T_{i+1} are the time interval between two consecutive impacts. The period of the gait is defined as the time between two consecutive impacts, since a complete cycle requires that the configuration of the biped at the end of the cycle is identical to the initial configuration. The leg designated the stance leg at the beginning of the cycle must therefore also be the stance leg just after the end of the cycle, a property that requires two consecutive impacts with the ground in order to switch legs twice.

The paper [8] demonstrates the existence of symmetric gaits only for the interval of angles $\psi \in [0, \approx 4.4]$, indicating that asymmetric gait cycles should exist for $\psi > 4.4$ degrees. Choosing $\psi = 5.00$ degrees and keeping the initial guess (7.2) with real solutions of d (7.3), the search procedure returns the following gait parameters

$$\begin{array}{ll} a = 0.235356, & e = -0.409888 \\ b = -1.168938, & f = -0.047946 \\ c = -0.409888, & g = 0.235356 \\ d = -1.819906, & h = -2.328662 \end{array} \tag{7.13}$$

where the initial conditions are given by

$$\begin{array}{ll} q_{1\star}(0) = 0.235356, & q_{2\star}(0) = -0.409888 \\ \dot{q}_{1\star}(0) = -1.168938, & \dot{q}_{2\star}(0) = -0.047946 \end{array} \tag{7.14}$$

This is a four-period asymmetric steady gait cycle, meaning that the gait is shaped by 4 consecutive intervals of continuous dynamics and the period of the gait can be written as

$$T_p = T_i + T_{i+1} + T_{i+2} + T_{i+3} \tag{7.15}$$

where i is the step number. To complete a full gait cycle the biped have to experience four consecutive ground impacts in order to return to its initial configuration. This special periodicity can easily be observed in Figure 7.12, where there are four distinct values for the step period at steady gait, which occurs after approximately 17 steps. Each step period correspond to a distinct cycle in the phase portrait as shown in Figure 7.13, where the difference in amplitude is clearly visible in the time evolution of q shown in Figure 7.14. Examining the time plot, it can be seen that steady gait is achieved after approximately 10 seconds.

Assessing the stability of the post-bifurcated gait is more difficult than the case for the symmetric gaits. Looking at Figures 7.12 and 7.14, there are indications of the presence of a symmetric gait that diverges towards the asymmetric gait after approximately three impacts with the ground. This

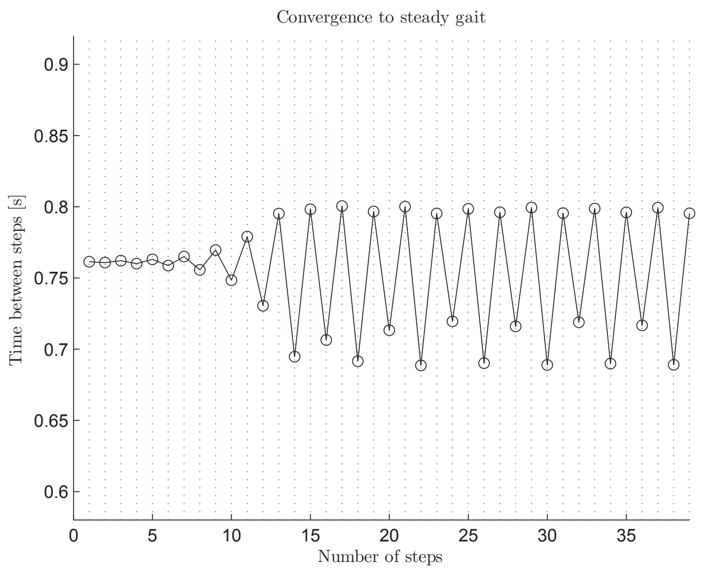

Figure 7.12: Time between consecutive steps for asymmetric gait at $\psi = 5.00$.

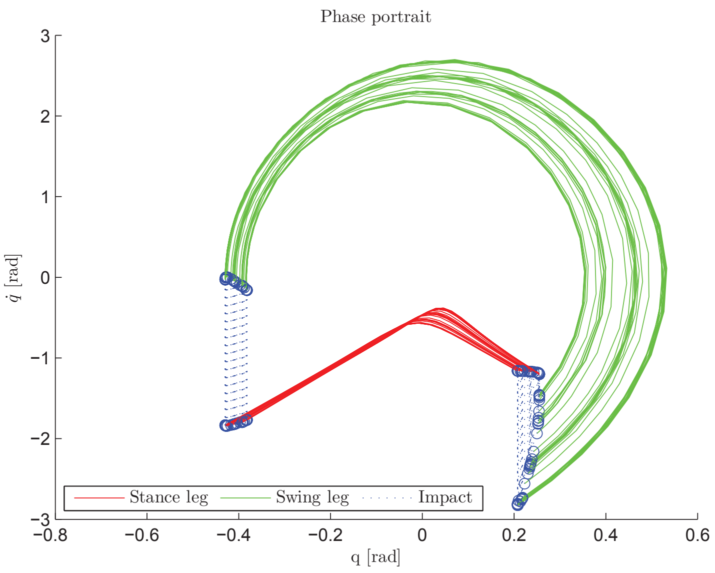

Figure 7.13: Phase portrait of the asymmetric gait at $\psi = 5.00$.

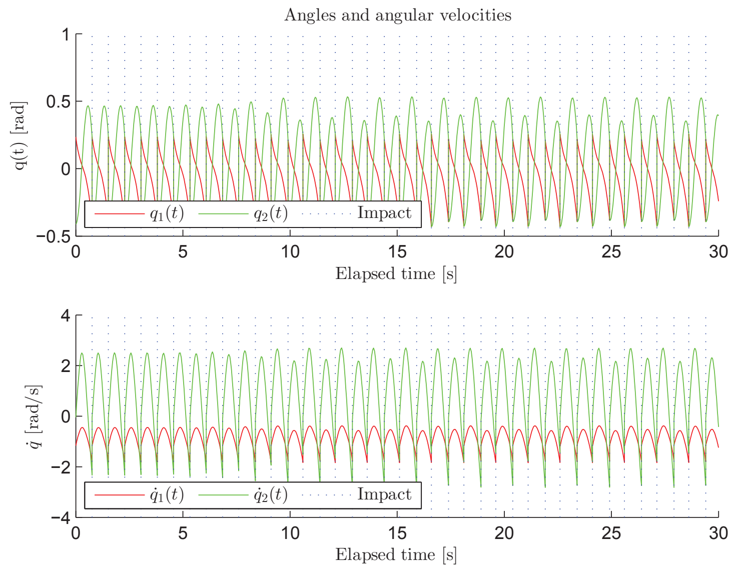

Figure 7.14: Time evolution of $q, \dot{q}$ for asymmetric gait at $\psi = 5.00$.

symmetric gait is clearly unstable, but the asymmetric gait itself appears asymptotically orbitally stable. Computing the eigenvalues of the transition matrix $\Xi(x_\star^-, x_\star^+)$ yields

$$\begin{aligned} \lambda_1 &= 0.092 \\ \lambda_2 &= -0.346 - 0.429i \\ \lambda_3 &= -0.346 + 0.429i \end{aligned} \tag{7.16}$$

which are inside the unit circle as indicated by Figure 7.15, confirming that the asymmetric gait is indeed asymptotically orbitally stable even though the pre-bifurcation gait is unstable. The initial parameters (7.14) describing the unstable symmetric gait can therefore be regarded as a perturbation of the initial parameters describing the four-periodic stable gait.

The four-period gait is not the only asymmetric gait cycle the compass-gait biped exhibits, in fact there exist a number of asymmetric gaits with periodicity 2^k, $k \in \mathbb{N}$. This is known as period-doubling bifurcation and k increases with increasing ψ, starting at $\psi \approx 4.4$. For a threshold value of ψ, the biped starts to exhibit chaotic gaits which is an extreme case of the asymmetric gait. During a chaotic gait the step periods of the biped never completely repeat themselves, leading to apparently random patterns in the gait. The concept of bifurcation and chaos for dynamic systems is a vast field of study, and the details are beyond the scope of this thesis. For a more detailed description of the phenomenon for the compass-gait biped, the

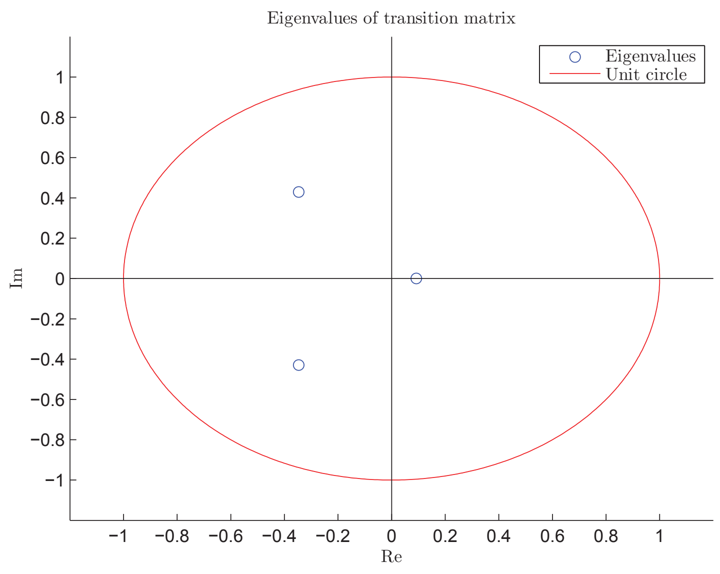

Figure 7.15: Eigenvalues of the transition matrix $\Xi(x_\star^-, x_\star^+)$ for asymmetric gait at $\psi = 5.00$.

interested reader is referred to [8] for an extensive treatment of the subject.

7.5 Remarks on Search Procedure

The search procedure presented in this thesis yields periodic gaits for the compass-gait biped that are in agreement with the other literature (see e.g. [8, 12, 21]) on the subject. In addition, the procedure has some useful properties that are remarked upon below.

Conventional search procedure

The conventional search procedure for finding periodic gaits for the compass-gait biped can be summarized (see [5]) as

Problem 2. *Find a, b, d, T such that $\bar{q}_1(T) = c, \bar{q}_2(T) = g, \dot{\bar{q}}_1(T) = d, \dot{\bar{q}}_2(T) = h$, with algebraic relations* (5.5), (5.6) *and* (5.9) *satisfied and $\bar{q}(t)$ being the solution of the differential equations* (3.7) *initiated at $\bar{q}_1(0) = a, \bar{q}_2(0) = e, \dot{\bar{q}}_1(0) = b, \dot{\bar{q}}_2(0) = f$.*

The procedure above requires solving four first-order differential equations per iteration of the optimization algorithm, in order to search for the four free parameters a, b, d, T. Compare this to the search procedure described by

Problem 1 in Chapter 5, which only requires determining the two independent parameters a, b and is based on solving two first-order differential equations (5.19) instead of the full biped dynamics (3.7). The computational burden of solving this problem is reduced by half compared to the conventional approach described by Problem 2 that is widely utilized in the literature (see e.g. [8] for results).

This reduction in complexity is due to the introduction of the synchronization function $\phi(\theta)$ between the generalized coordinates q_1, q_2, that removes the explicit dependence on time. The resulting second-order differential equation (5.19) describing the evolution of $\phi(\theta)$ along the periodic orbit, has the added benefit that the occurrence of the ground impact is described by the known geometric relation (3.29) instead of the unknown time parameter T. The solution interval of the dynamics can therefore be explicitly determined using the known boundary conditions for the scalar variable $\theta(t)$ describing the motion along the target trajectory.

Non-feasible gaits

Considering that the periodic gaits found by solving Problem 1 are consistent with results obtained using the conventional search algorithm described by Problem 2 (see e.g. [8]), the following proposition is motivated

Proposition 1. *[5, Proposition 1] The set of solutions for Problem 1 contains all the solutions of Problem 2.*

Examining the derivations of the search procedure in Chapter 5.4 it trivially follows that any solution to Problem 2 is also a solution to Problem 1. The reverse, however, is not generally true, since the virtual holonomic constraint approach may yields non-feasible periodic gaits. Initiating the search procedure with

$$a_0 = 0.20, \quad b_0 = 1.0, \quad \psi = 3.00 \tag{7.17}$$

and choosing the minus sign in the solution to the quadratic equations for d, yields a gait with the following initial conditions

$$\begin{aligned} q_{1\star}(0) &= 0.204676, & q_{2\star}(0) &= -0.309396 \\ \dot{q}_{1\star}(0) &= 1.210571, & \dot{q}_{2\star}(0) &= 0.712484 \end{aligned} \tag{7.18}$$

The parameters follow after 55 iterations of `fminsearch` converging to an objective function value $F \approx 9.4 \cdot 10^{-12} \approx 0$, and are clearly a valid solution to Problem 1. However, this is a non-feasible gait since it requires that the compass-gait biped exhibits a walking motion moving upward the slope (from right to left in Figure 3.1), as can be seen by the fact that the velocities $\dot{q}_\star(0)$ are positive.

This rather bizarre result is due to the removal of the explicit dependence of time from the full dynamics of the system. Whereas time has a defined one-way direction that corresponds to observation of physical phenomena, such as gravity causing objects to fall downwards not upwards, the new parameterization of the gait using the geometric quantity θ has no such distinction. Traversing a typical limit cycle describing a periodic gait (such as Figure 7.3) using time as the measure of progress yields a definite sense of direction, either clockwise or counter-clockwise, that is lost when utilizing the parameter θ instead. The search procedure might therefore yield periodic gaits that correspond to solutions of the dynamics (3.7) in negative time, and thus give the impression that the biped is defying gravity and traveling back up the slope.

These non-feasible gaits are extraneous for the passive walker discussed in this thesis, but the existence of them as solutions of Problem 1 might be of interest when actuation of the biped is available.

Unstable limit cycles

There are indications [5] that the unstable limit cycle described by the initial conditions (7.10) can not be found using the conventional search procedure defined by Problem 2. This limit cycle is part of a family of periodic orbits for $\psi \in (0, 6]$ deg that are all unstable, and can easily be found by choosing the plus sign in the solution to the quadratic equations for d when solving Problem 1.

There are few mentions of this limit cycle outside the literature discussing virtual holonomic constraints, although the paper [7] showed the existence of an unstable cycle for their compass-gait biped on small slopes. The approach described consisted of computing the first and second order approximations of the hybrid dynamics (3.35), and then using a perturbation method to derive analytic expressions for the step period and initial angles and velocities. Due to the approximation in the dynamics, these expressions are only valid for small values of ψ and can not be used to find all the unstable cycles for the full range $\psi \in (0, 6]$ deg. The advantage of the virtual holonomic constraint approach is that no such approximation must be performed, and the dynamics used in the search procedure are exact.

These limit cycles are of limited use for the passive walker due to their unstable nature. However, if actuation is available, these periodic gaits can be stabilized and used to generate new walking motions for the actuated biped. A stabilizing controller for this purpose can be designed using transverse linearization of the reduced dynamics as detailed in e.g. [4, 6, 17].

An interesting property of the limit cycles is that the initial conditions for the stable (7.5) and the unstable (7.10) periodic gaits are comparably close to each other. This indicates that the region of attraction for the stable gait is quite small, and makes it difficult to obtain numeric estimates for its size.

8 Conclusion

The aim of this thesis was to study the passive gaits of a 2-DOF planar walking device, commonly referred to as the compass-gait biped, on a downwards slope. The passive gaits are of special interest in the field of legged robotics since the motive force driving the locomotion comes solely from the conversion of the robots gravitational potential energy as is travels down the slope.

The passive gaits studied are limit cycle solutions of the hybrid dynamics of the compass-gait biped, consisting of 2-DOF continuous-time equations of motion and an update map that models the instantaneous update of states after impact with the ground. The equations of motion were modeled using standard Euler-Lagrange dynamics with generalized coordinates chosen as the absolute angles of the legs. Two different approaches for deriving the impact map were considered. The first derivation was based on conservation of angular momentum during the collision, and resulted in a simple map between the velocities just before and just after impact. The second approach exploited the continuous-time dynamics to yield a map that in addition to the velocities also gave an expression for the impact forces the biped experiences during the collision. Both update maps were verified through numerical simulations and yielded identical results for the velocity updates. The combination of the Euler-Lagrange dynamics and the update map constitute the hybrid dynamics of the compass-gait biped and a MATLAB script was developed to visualize the motion of the robot as it travels down the slope.

Finding passive gaits for hybrid systems is a difficult task due to the combination of continuous and discrete elements in the dynamics. In this thesis an approach for finding limit cycles for an unactuated 2-DOF hybrid system is presented. The approach exploits geometric relations that must hold between the two generalized coordinates during the continuous-time sub-arc of the periodic trajectory. These relations are called virtual holonomic constraints and lead to the formulation of a minimization problem for searching for periodic gaits of the biped. This optimization problem requires finding two parameters and is based on solving a second-order differential equation for the computation of the introduced constraints. Implementation of the search procedure in MATLAB has verified that the proposed approach does work, and has yielded both stable and unstable limit cycles for the compass-gait biped. Numerical results of the search have been presented and are found to be consistent with other literature on the subject.

The introduction of virtual holonomic constraints allows for the computation of an auxiliary linear system that describes the dynamics of the biped away from the target periodic trajectory. This linear system is the result of computing the transverse linearization of the reduced dynamics of the hybrid system orthogonal to the periodic orbit, and in that way automatically generating a family of moving Poincaré sections. Verifying the stability of the limit cycle is therefore reduced to the simple problem of computing the eigenvalues of the discrete auxiliary linear system, where exponential stability ensures that the periodic orbit is a fixed point of each Poincaré map. Computation of the transverse linearization and the resulting linear system is carried out in Maple, and a MATLAB script for determining orbital stability of passive gaits has been implemented. Some numerical results of this algorithm is presented in the text along with full derivations of the procedure.

The main contribution of this thesis is presenting a structured procedure for finding passive gaits and assessing their stability for a simple walking machine such as the compass-gait biped. The introduction of virtual holonomic constraints is a powerful tool that enables the use of analytical arguments when analyzing the properties of different gait cycles. The preliminary results obtained for the stability assessment of the periodic orbits are important foundations for further research into topics such as region of attraction, and the sensitivity of the gait to external forces and perturbations of parameters. These are fascinating areas of study that are central to further research into legged locomotion, and passive gaits may prove to be the most efficient choice for controlling biped robots in complex environments.

Bibliography

[1] Bernard Brogliato. *Nonsmooth mechanics: models, dynamics, and control*. Springer, 1999.

[2] Chi-Tsong Chen. *Linear system theory and design*. Oxford University Press, Inc., 2009.

[3] Olav Egeland and Jan Tommy Gravdahl. *Modeling and simulation for automatic control*, volume 76. Marine Cybernetics Trondheim, Norway, 2002.

[4] Leonid Freidovich, Anton Shiriaev, and Ian Manchester. Stability analysis and control design for an underactuated walking robot via computation of a transverse linearization. In *Proc. 17th IFAC World Congress, Seoul, Korea*, pages 10166–10171, 2008.

[5] Leonid B Freidovich, Uwe Mettin, Anton S Shiriaev, and Mark W Spong. A passive 2-dof walker: hunting for gaits using virtual holonomic constraints. *Robotics, IEEE Transactions on*, 25(5):1202–1208, 2009.

[6] Leonid B Friedovich and Anton S Shiriaev. Transverse linearization for an underactuated compass-like biped robot and analysis of the closed-loop system. Review copy, 2010.

[7] Mariano Garcia, Anindya Chatterjee, Andy Ruina, Michael Coleman, et al. The simplest walking model: Stability, complexity, and scaling. *J BIOMECH ENG TRANS ASME*, 120(2):281–288, 1998.

[8] Ambarish Goswami, Benoit Thuilot, and Bernard Espiau. A study of the passive gait of a compass-like biped robot symmetry and chaos. *The International Journal of Robotics Research*, 17(12):1282–1301, 1998.

[9] Jesse W Grizzle, Gabriel Abba, and Franck Plestan. Asymptotically stable walking for biped robots: Analysis via systems with impulse effects. *Automatic Control, IEEE Transactions on*, 46(1):51–64, 2001.

[10] H. K. Khalil. *Nonlinear Systems*. Prentice Hall, international 3rd edition, 2002.

[11] MathWorks. Documentation of MATLAB algorithm `fminsearch`. `http://www.mathworks.se/help/matlab/ref/fminsearch.html`. Accessed: 2013-07-10.

[12] Tad McGeer. Passive dynamic walking. *the international journal of robotics research*, 9(2):62–82, 1990.

[13] Uwe Mettin. *Principles for planning and analyzing motions of underactuated mechanical systems and redundant manipulators.* PhD thesis, Umeå University, 2009.

[14] Jorge Nocedal and Stephen J Wright. *Numerical Optimization.* Springer, 2nd edition, 2006.

[15] J Perram, Anton Shiriaev, C Canudas-de Wit, and F Grognard. Explicit formula for a general integral of motion for a class of mechanical systems subject to holonomic constraint. In *Proceedings of the 2nd IFAC Workshop Control Methods for Lagrangian and Hamiltonian Systems*, pages 87–92, 2003.

[16] A. Shiriaev, A. Robertsson, J. Perram, and A. Sandberg. Periodic motion planning for virtually constrained (hybrid) mechanical systems. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 4035–4040. IEEE, 2005.

[17] A. S. Shiriaev and L. B. Freidovich. Transverse linearization for impulsive mechanical systems with one passive link. *Automatic Control, IEEE Transactions on*, 54(12):2882–2888, 2009.

[18] Anton Shiriaev, John W Perram, and Carlos Canudas-de Wit. Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach. *Automatic Control, IEEE Transactions on*, 50(8):1164–1176, 2005.

[19] Anton S Shiriaev, Leonid B Freidovich, and Ian R Manchester. Can we make a robot ballerina perform a pirouette? orbital stabilization of periodic motions of underactuated mechanical systems. *Annual Reviews in Control*, 32(2):200–211, 2008.

[20] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot modeling and control.* John Wiley & Sons, Inc., 2006.

[21] Mark W Spong, Jonathan K Holm, and Dongjun Lee. Passivity-based control of bipedal locomotion. *Robotics & Automation Magazine, IEEE*, 14(2):30–40, 2007.

A Contents of CD

The following folders are included on the CD accompanying this master thesis:

Report Contains all the required LaTeXsource files and graphics required to successfully compile the thesis report into a pdf-file.

Maple Contains all the Maple worksheets created to solve analytic problems in the thesis, including the worksheets posted in the appendix below. Files are written in Maple 16.

Matlab Contains all the needed MATLAB scripts and functions to simulate the compass-gait biped (`main_simulation_script.m`), search for periodic gaits (`run_gait_search.m`) and assess the stability of the discovered gaits (`main_stability_script.m`). Files are written using the editor in MATLAB 2013a.

Papers Contains all cited articles that are available digitally through NTNU as pdf-files. Filenames are identical to article titles for convenience.

B Maple Worksheets

The following appendix contains the relevant Maple worksheets used to derive the hybrid dynamic system describing the compass-gait biped, the relations used in formulating the gait search procedure and the computations for the transverse linearization of the dynamics.

B.1 Finding Euler-Lagrange Equations of Motion

```
restart

#Initialize linear algebra
with(LinearAlgebra) :

#Vector of generalized coordinates
q := Vector( [q1, q2]);
Dq := Vector( [Dq1, Dq2]) :
DDq := Vector( [DDq1, DDq2]) :

#Homogenous transformation matrices
H01 := Matrix( [ [cos( -q1), sin( -q1), 0, sin( -q1)·b], [ -sin( -q1), cos( -q1), 0, cos( -q1)·b], [0, 0,
      1, 0], [0, 0, 0, 1]]);
H1h := Matrix( [ [1, 0, 0, 0], [0, 1, 0, a], [0, 0, 1, 0], [0, 0, 0, 1]]);
H02 := Matrix( [ [cos(q2), -sin(q2), 0, l·sin( -q1) + a·sin(q2)], [sin(q2), cos(q2), 0, l·cos( -q1)
      - a·cos(q2)], [0, 0, 1, 0], [0, 0, 0, 1]]);

 H2e := Matrix( [ [1, 0, 0, 0], [0, 1, 0, -b], [0, 0, 1, 0], [0, 0, 0, 1]]) :

#Coordinate frame origins
#Center of mass first link:
o1 := MatrixVectorMultiply(H01, Vector( [0, 0, 0, 1])) :
o1 := combine(o1, trig) :
o1x := o1[1];
o1y := o1[2];
#Center of mass hip:
H0h := MatrixMatrixMultiply(H01, H1h) :
oh := MatrixVectorMultiply(H0h, Vector( [0, 0, 0, 1])) :
oh := combine(oh, trig) :
ohx := oh[1];
ohy := oh[2];

#Center of mass second link:
 o2 := MatrixVectorMultiply(H02, Vector( [0, 0, 0, 1])) :
 o2 := combine(o2, trig) :
 o2x := o2[1];
 o2y := o2[2];
#End of swing leg:
H0e := MatrixMatrixMultiply(H02, H2e) :
 oe := MatrixVectorMultiply(H0e, Vector( [0, 0, 0, 1])) :
 oe := combine(oe, trig) :
 oex := oe[1];
 oey := oe[2];

#Linear velocities of centers of mass:
#First link:
 Do1x := diff(o1x, q1)·Dq1;
 Do1y := diff(o1y, q1)·Dq1;
#Hip:
```

$Dohx := diff(ohx, q1) \cdot Dq1;$
$Dohy := diff(ohy, q1) \cdot Dq1;$

#Second link:
$Do2x := diff(o2x, q1) \cdot Dq1 + diff(o2x, q2) \cdot Dq2;$
$Do2y := diff(o2y, q1) \cdot Dq1 + diff(o2y, q2) \cdot Dq2;$

#Kinetic energy of system

#First link:
$K1 := \frac{m1}{2} \cdot \left(Do1x^2 + Do1y^2\right) :$
$K1 := collect(combine(K1, trig), Dq1);$

#Hip:
$Kh := \frac{mH}{2} \cdot \left(Dohx^2 + Dohy^2\right) :$
$Kh := collect(combine(Kh, trig), Dq1);$

#Second link:
$K2 := \frac{m2}{2} \cdot \left(Do2x^2 + Do2y^2\right) :$
$K2 := collect(combine(K2, trig), Dq2);$

#Total kinetic energy T(q,Dq):
$K := K1 + K2 + Kh :$
$K := collect(K, \{Dq1, Dq2\});$
$K :$
#Simplify
$par1 := mH \cdot l^2 + b^2\, m1 + l^2\, m2 :$
$par2 := m2\, l\, a :$
$par3 := m2\, a^2 :$

$K0 := algsubs\left(m1\, b^2 + m2\, l^2 + mH\, a^2 + 2 \cdot mH\, a\, b + mH\, b^2 = par1, K\right) :$
$K0 := algsubs(par1 = p1, K0) :$
$K0 := algsubs(par2 = p2, K0) :$
$K0 := algsubs(par3 = p3, K0) :$

$K0;$

#Computing inertia matrix

$m11 := algsubs\left(2\, a\, b + b^2 + a^2 = l^2, collect(sort(diff(K0, Dq1, Dq1)), [mh, m1, m2])\right);$
$m12 := sort(diff(K0, Dq1, Dq2));$
$m21 := m12;$
$m22 := sort(diff(K0, Dq2, Dq2));$
$M := Matrix([[m11, m12], [m21, m22]]);$

#Potential energy of system

```
#First link:
P1 := m1·g·o1y;
#Hip:
Ph := mH·g·ohy;

#Second link:
P2 := m2·g·o2y;
#Total potential energy:
 P := P1 + P2 + Ph :
 P := combine(P, trig) :
 P := collect(P, [sin, cos]) :
 P + const;
 #Simplify

 par4 := (m1·b + m2·l + mH·l)·g;
 par5 := m2 a·g;

 P0 := algsubs((m1·b + m2·l + mH·a + mH·b)·g = par4, P);
 P0 := algsubs(par4 = p4, P0) :
 P0 := algsubs(par5 = p5, P0) :

 P0;
```

#Lagrangian of the dynamical system:

```
L := K0 − P0;
#First link:
L1 := diff(L, Dq1) :
G1 := diff(L, q1) :
F1 := diff(L1, q1)·Dq1 + diff(L1, Dq1)·DDq1 + diff(L1, q2)·Dq2 + diff(L1, Dq2)·DDq2 − G1 :
F1 := collect(F1, {Dq1, Dq2, DDq1, DDq2, });

#Second link:
L2 := diff(L, Dq2) :
G2 := diff(L, q2) :
F2 := diff(L2, q1)·Dq1 + diff(L2, Dq1)·DDq1 + diff(L2, q2)·Dq2 + diff(L2, Dq2)·DDq2 − G2 :
F2 := collect(F2, {Dq1, Dq2, DDq1, DDq2});
```

#Coriolis and centrifugal torque matrix C(q,Dq):

$F := F1 :$

$c11 := collect\left(simplify\left(diff(F, Dq1, Dq1)\cdot\left(\frac{Dq1}{2}\right) + diff(F, Dq1, Dq2)\cdot\left(\frac{Dq2}{2}\right)\right), [\sin, \cos]\right) :$

$c12 := collect\left(simplify\left(diff(F, Dq2, Dq1)\cdot\left(\frac{Dq1}{2}\right) + diff(F, Dq2, Dq2)\cdot\left(\frac{Dq2}{2}\right)\right), [\sin, \cos]\right) :$

$F := F2 :$

$c21 := collect\left(simplify\left(diff(F, Dq1, Dq1)\cdot\left(\frac{Dq1}{2}\right) + diff(F, Dq1, Dq2)\cdot\left(\frac{Dq2}{2}\right)\right), [\sin, \cos]\right) :$

$c22 := collect\left(simplify\left(diff(F, Dq2, Dq1) \cdot \left(\frac{Dq1}{2}\right) + diff(F, Dq2, Dq2) \cdot \left(\frac{Dq2}{2}\right)\right), [\sin, \cos]\right)$:
$Cc := Matrix([[c11, c12], [c21, c22]]);$

#Gravitational torque vector G(q):

$G := Vector([F1, F2]) - MatrixVectorMultiply(M, DDq) - MatrixVectorMultiply(Cc, Dq)$:
$G := collect(G, [DDq1, \sin(q1), mh])$:
$G := algsubs(a + b = l, G, 'exact')$:
$G := simplify\left(algsubs\left(a^2 + 2\,a\,b + b^2 = l^2, G, 'exact'\right)\right);$
# Finding value of constant term in potential energy
$P0 := int(G(1), q1 = 0..q1) + int(G(2), q2 = 0..q2);$

#Conversion to Matlab Code:

$with(CodeGeneration)$:

```
Matlab(o1x, resultname = "o1x");
Matlab(o1y, resultname = "o1y");
Matlab(ohx, resultname = "ohx");
Matlab(ohy, resultname = "ohy");
Matlab(o2x, resultname = "o2x");
Matlab(o2y, resultname = "o2y");
Matlab(oex, resultname = "oex");
Matlab(oey, resultname = "oey");

Matlab(M[1, 1], resultname = "m11");
Matlab(M[1, 2], resultname = "m12");
Matlab(M[2, 1], resultname = "m21");
Matlab(M[2, 2], resultname = "m22");

Matlab(c11, resultname = "c11");
Matlab(c12, resultname = "c12");
Matlab(c21, resultname = "c21");
Matlab(c22, resultname = "c22");

Matlab(G[1], resultname = "g1");
Matlab(G[2], resultname = "g2");


Matlab(par1, resultname = "p1");
Matlab(par2, resultname = "p2");
Matlab(par3, resultname = "p3");
Matlab(par4, resultname = "p4");
Matlab(par5, resultname = "p5");
```

#Storing matrices for other worksheets:

save M, Cc, G, K0, P0, `eom.m`;

B.2 Computing Impact Map $P_q(q^-)$

restart

$with(LinearAlgebra):$
$with(VectorCalculus):$
$with(CodeGeneration):$

# Coordinate system origins

A - Origin at inital pivot point at the end of Link 1
B - Origin at Hip
C - Origin at impact point at the end of Link 2

# Distance and velocities for Link 1 about Hip

$r1_B := Vector([-a\cdot\sin(-q1), -a\cdot\cos(-q1), 0]):$
$r1_A := Vector([b\cdot\sin(-q1), b\cdot\cos(-q1), 0]):$
$rH_C := Vector([-l\cdot\sin(q2), l\cdot\cos(q2), 0]):$
$v1_- := CrossProduct(Vector([0, 0, Dq1_-]), r1_A):$
$v1_+ := CrossProduct(Vector([0, 0, Dq1_+]), r1_B) + CrossProduct(Vector([0, 0, Dq2_+]), rH_C):$

$BasisFormat(false):$

#Distance and velocities for complete biped about Impact

$r2_C := Vector([-b\cdot\sin(q2), b\cdot\cos(q2), 0]):$
$r1_C := rH_C + r1_B:$
$rH_A := Vector([l\cdot\sin(-q1), l\cdot\cos(-q1), 0]):$
$r2_B := Vector([a\cdot\sin(q2), -a\cdot\cos(q2), 0]):$

$vh_- := CrossProduct(Vector([0, 0, Dq1_-]), rH_A):$
$v2_- := CrossProduct(Vector([0, 0, Dq2_-]), r2_B) + vh_-:$

$v2_+ := CrossProduct(Vector([0, 0, Dq2_+]), r2_C):$
$vh_+ := CrossProduct(Vector([0, 0, Dq2_+]), rH_C):$

#Angular momentum

$AMt := CrossProduct(r1_B, m1\cdot(v1_+)) = CrossProduct(r1_B, m1\cdot(v1_-)):$
$AMr := CrossProduct(r1_C, m1\cdot(v1_+)) + CrossProduct(r2_C, m2\cdot(v2_+)) + CrossProduct(rH_C, mH \cdot(vh_+)) = CrossProduct(r1_C, m1\cdot(v1_-)) + CrossProduct(r2_C, m2\cdot(v2_-)) + CrossProduct(rH_C, mH\cdot(vh_-)):$

#Simplify expressions

$AMtsimp := lhs(AMt(3)) - rhs(AMt(3)):$

$AMtsimp := simplify(AMtsimp,'trig')$:
$AMtsimp := collect(algsubs(\sin(q1)\cdot\sin(q2) + \cos(q1)\cdot\cos(q2) = \cos(q2 - q1), AMtsimp),$
$[Dq1_+, Dq2_+, Dq1_-, Dq2_-])$;

$AMrsimp := lhs(AMr(3)) - rhs(AMr(3))$:
$AMrsimp := simplify(AMrsimp,'trig')$:
$AMrsimp := collect(algsubs(\sin(q1)\cdot\sin(q2) + \cos(q1)\cdot\cos(q2) = \cos(q2 - q1), AMrsimp),$
$[Dq1_+, Dq2_+, Dq1_-, Dq2_-])$;

#Formulating map

A0 := Matrix(2, 2) :
B0 := Matrix(2, 2) :

for i **from** 1 **to** 2 **do**
$A0(1, i) := subs(Dq1_+ = \mathrm{i}$ **mod** $2, Dq2_+ = \mathrm{i} - 1$ **mod** $2, Dq1_- = 0, Dq2_- = 0, AMrsimp)$:
$A0(2, i) := subs(Dq1_+ = \mathrm{i}$ **mod** $2, Dq2_+ = \mathrm{i} - 1$ **mod** $2, Dq1_- = 0, Dq2_- = 0, AMtsimp)$:
end do:

for i **from** 1 **to** 2 **do**
$B0(1, i) := -subs(Dq1_- = \mathrm{i}$ **mod** $2, Dq2_- = \mathrm{i} - 1$ **mod** $2, Dq1_+ = 0, Dq2_+ = 0, AMrsimp)$:
$B0(2, i) := -subs(Dq1_- = \mathrm{i}$ **mod** $2, Dq2_- = \mathrm{i} - 1$ **mod** $2, Dq1_+ = 0, Dq2_+ = 0, AMtsimp)$:
end do:

#Inserting relabeling of legs

$Perm := Matrix([[0, 1], [1, 0]])$;
$Qp := Q_+ = MatrixMatrixMultiply(A0, Perm)$;
$Qn := Q_- = B0$;

#Simplify

$par6 := m1\, l^2 + l^2\, mH + b^2\, m2$:
$par7 := a\, m1\, l$:
$par8 := a^2\, m1$:

$Qp0 := algsubs(par6 = p6, Qp)$:
$Qp0 := algsubs(par7 = p7, Qp0)$:
$Qp0 := algsubs(par8 = p8, Qp0)$:

$Qp0$;

$par9 := a\, m1\, b$:
$par10 := b\, m2\, l + l^2\, mH + m1\, b\, l$:
$par11 := b\, m2\, a$:

```
Qn0 := algsubs(par9=p9, Qn) :
Qn0 := algsubs(par10=p10, Qn0) :
Qn0 := algsubs(par11=p11, Qn0) :

Qn0;
```

#Creating Matlab code

```
Matlab(rhs(Qp0), resultname='Qp');
Matlab(rhs(Qn0), resultname='Qn');

Matlab(par6, resultname='p6');
Matlab(par7, resultname='p7');
Matlab(par8, resultname='p8');
Matlab(par9, resultname='p9');
Matlab(par10, resultname='p10');
Matlab(par11, resultname='p11');


save Qp0, Qn0, "impact_map.m";
```

B.3 Deriving Algebraic Relations for a, b, c, d, e, f, g, h

```
restart ;

#Initialize linear algebra
with(LinearAlgebra) :
with(CodeGeneration) :
```

#Import expressions from other worksheets:

```
read "../Impact_map/impact_map.m";
read "../Equations_of_motion/eom.m";
```

#Solve for relations:

```
g0 := g = a;
c0 := c =-a − 2·psi;
e0 := e =-a − 2·psi;

update := MatrixInverse(rhs(Qp0)).rhs(Qn0) :
update := subs(q1 = c, q2 = g, update) :
update := Vector([b,f]) = simplify(subs(c = rhs(c0), g = rhs(g0), update.Vector([d, h])));


h0 := collect(simplify(h = solve(simplify(update(1)), h)), [b, d, cos(2 a + 2 ψ)]);
f0 := simplify(subs(h = rhs(h0), simplify(update(2))));
```

#Insert into energy:

$E := K0 + P0 :$

$$Ev := E0 = collect\Big(subs(q1 = \mathrm{theta}(t), q2 = \mathrm{phi}(\mathrm{theta}(t)), Dq1 = diff(\mathrm{theta}(t), t), Dq2 = diff(\mathrm{phi}(\mathrm{theta}(t)), t), E), \Big[\Big(\frac{d}{dt}\,\theta(t)\Big)^2\Big]\Big);$$

$$E0ab := \Big(subs\Big(\frac{d}{dt}\,\theta(t) = b, \mathrm{D}(\phi)(\theta(t)) = \frac{f}{b}, \theta(t) = a, \phi(\mathrm{a}) = e, rhs(Ev)\Big)\Big);$$

$$E0cd := \Big(subs\Big(\frac{d}{dt}\,\theta(t) = d, \mathrm{D}(\phi)(\theta(t)) = \frac{h}{d}, \theta(t) = c, \phi(\mathrm{c}) = g, rhs(Ev)\Big)\Big);$$

#Reducing energy parameters:

```
E0d1 := subs(h = rhs(h0), c = rhs(c0), g = rhs(g0), E0cd) :
E0d2 := subs(f= rhs(f0), e = rhs(e0), g = rhs(g0), E0ab) :
```

$E0d := collect(E0d1 - E0d2, [d^2, d]);$
$A2 := coeff(E0d, d^2);$
$B2 := coeff(E0d, d);$
$C2 := subs(d = 0, E0d);$

#Creating Matlab code

```
Matlab(rhs(g0), resultname ='g');
Matlab(rhs(c0), resultname ='c');
Matlab(rhs(f0), resultname ='f');
Matlab(rhs(h0), resultname ='h');

Matlab(E0ab, resultname ='E0ab');
Matlab(E0cd, resultname ='E0cd');
Matlab(E0d, resultname ='E0d');

Matlab(A2, resultname ='A2');
Matlab(B2, resultname ='B2');
Matlab(C2, resultname ='C2');
```

B.4 Finding the Differential Equation for $\phi(\theta)$

restart

#Initialize linear algebra
with(*LinearAlgebra*) :
with(*CodeGeneration*) :

#Import expressions from other worksheets:

read "../Equations_of_motion/eom.m";

#Inserting for virtual holonomic constraint

$E := K0 + P0 :$

$$Ev := E0 = collect\left(subs(q1 = \text{theta}(t), q2 = \text{phi}(\text{theta}(t)), Dq1 = diff(\text{theta}(t), t), Dq2 = diff(\text{phi}(\text{theta}(t)), t), E), \left[\left(\frac{d}{dt} \theta(t) \right)^2 \right] \right);$$

$$eom := M.Vector([DDq1, DDq2]) + Cc.Vector([Dq1, Dq2]) + G :$$

$$aby1 := collect\left(subs(q1 = \text{theta}(t), q2 = \text{phi}(\text{theta}(t)), Dq1 = diff(\text{theta}(t), t), Dq2 = diff(\text{phi}(\text{theta}(t)), t), DDq1 = diff(diff(\text{theta}(t), t), t), DDq2 = diff(diff(\text{phi}(\text{theta}(t)), t), t), eom(1)), \left[\frac{d^2}{dt^2} \theta(t), \left(\frac{d}{dt} \theta(t) \right)^2 \right] \right);$$

$$aby2 := collect\left(subs(q1 = \text{theta}(t), q2 = \text{phi}(\text{theta}(t)), Dq1 = diff(\text{theta}(t), t), Dq2 = diff(\text{phi}(\text{theta}(t)), t), DDq1 = diff(diff(\text{theta}(t), t), t), DDq2 = diff(diff(\text{phi}(\text{theta}(t)), t), t), eom(2)), \left[\frac{d^2}{dt^2} \theta(t), \left(\frac{d}{dt} \theta(t) \right)^2 \right] \right);$$

$$aby1 := collect\left(aby1, \left[\frac{d^2}{dt^2} \theta(t), \left(\frac{d}{dt} \theta(t) \right)^2 \right] \right) :$$

$$aby2 := collect\left(aby2, \left[\frac{d^2}{dt^2} \theta(t), \left(\frac{d}{dt} \theta(t) \right)^2 \right] \right) :$$

$alpha1 \coloneqq coeff\left(aby1, \frac{d^2}{dt^2}\,\theta(t)\right);$

$alpha2 \coloneqq coeff\left(aby2, \frac{d^2}{dt^2}\,\theta(t)\right);$

$gamma1 \coloneqq subs\left(\left(\frac{d}{dt}\,\theta(t)\right)^2 = 0, \frac{d^2}{dt^2}\,\theta(t) = 0, aby1\right);$

$gamma2 \coloneqq subs\left(\left(\frac{d}{dt}\,\theta(t)\right)^2 = 0, \frac{d^2}{dt^2}\,\theta(t) = 0, aby2\right);$

$beta1 \coloneqq subs\left(\frac{d^2}{dt^2}\,\theta(t) = 0, \left(\frac{d}{dt}\,\theta(t)\right)^2 = 1, aby1\right) - gamma1;$

$\beta 2 \coloneqq subs\left(\frac{d^2}{dt^2}\,\theta(t) = 0, \left(\frac{d}{dt}\,\theta(t)\right)^2 = 1, aby2\right) - \gamma 2;$

$abyEq \coloneqq collect\left(solve\left(subs\left(\left(\frac{d}{dt}\,\theta(t)\right)^2 = placeholder, diff(\mathrm{theta}(t), t)^2 = \frac{\alpha 2 \cdot \gamma 1 - \alpha 1 \cdot \gamma 2}{\alpha 1 \cdot \beta 2 - \alpha 2 \cdot \beta 1}\right),\right.\right.$
$\qquad \left.\left. \mathrm{D}^{(2)}(\phi)(\theta(t))\right), placeholder\right);$

$energyEq \coloneqq solve\left(subs\left(\left(\frac{d}{dt}\,\theta(t)\right)^2 = placeholder, Ev\right), placeholder\right);$

$f \coloneqq subs(placeholder = energyEq, abyEq);$

#Creating Matlab code

$f0 \coloneqq algsubs(\mathrm{D}(\phi)(\theta(t)) = Dphi, f):$
$f0 \coloneqq algsubs(\mathrm{phi}(\mathrm{theta}(t)) = \mathrm{phi}, f0):$
$f0 \coloneqq algsubs(\mathrm{theta}(t) = \mathrm{theta}, f0):$

Matlab(*f0*, *resultname* ='*f0*');

save *f*, "diff_eq_phi.m";

B.5 Transverse Linearization of Continuous Dynamics

restart

#Initialize linear algebra
with(*LinearAlgebra*) :
with(*VectorCalculus*) :
with(*ArrayTools*) :
with(*CodeGeneration*) :

#Import expressions from other worksheets:

read "../Equations_of_motion/eom.m";
read "diff_eq_phi.m";

DDphi_sol := *f*:

#Inserting for virtual holonomic constraint:

$q1sub := \mathrm{theta}(t)$:
$q2sub := \mathrm{phi}(\mathrm{theta}(t))$:

$eom := M.Vector([DDq1, DDq2]) + Cc.Vector([Dq1, Dq2]) + G$:

$$aby := collect\left(subs(q1 = q1sub, q2 = q2sub, Dq1 = diff(q1sub, t), Dq2 = diff(q2sub, t), DDq1 = diff(q1sub, t, t), DDq2 = diff(q2sub, t, t), eom), \left[\frac{d^2}{dt^2}\,\theta(t), \left(\frac{d}{dt}\,\theta(t)\right)^2\right]\right):$$

$aby1 := aby(1)$;
$aby2 := aby(2)$;

$$aby1 := collect\left(aby1, \left[\frac{d^2}{dt^2}\,\theta(t), \left(\frac{d}{dt}\,\theta(t)\right)^2\right]\right):$$

$$aby2 := collect\left(aby2, \left[\frac{d^2}{dt^2}\,\theta(t), \left(\frac{d}{dt}\,\theta(t)\right)^2\right]\right):$$

$$alpha1 := coeff\left(aby1, \frac{d^2}{dt^2}\,\theta(t)\right);$$

$$alpha2 := coeff\left(aby2, \frac{d^2}{dt^2}\,\theta(t)\right);$$

$$gamma1 := subs\left(\left(\frac{d}{dt}\,\theta(t)\right)^2 = 0, \frac{d^2}{dt^2}\,\theta(t) = 0, aby1\right);$$

$$gamma2 := subs\left(\left(\frac{d}{dt}\,\theta(t)\right)^2 = 0, \frac{d^2}{dt^2}\,\theta(t) = 0, aby2\right);$$

$$beta1 := subs\left(\frac{d^2}{dt^2}\,\theta(t) = 0, \left(\frac{d}{dt}\,\theta(t)\right)^2 = 1, aby1\right) - gamma1;$$

$$\beta 2 := \mathit{subs}\left(\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,\theta(t) = 0,\ \left(\frac{\mathrm{d}}{\mathrm{d}t}\,\theta(t)\right)^2 = 1,\ \mathit{aby2}\right) - \gamma 2;$$

#Inserting for error coordinates:

$\mathit{q1sub} := \mathrm{theta}(t)$:
$\mathit{q2sub} := y(t) + \mathrm{phi}(\mathrm{theta}(t))$:

$\mathit{eom\_y} := \mathit{subs}(\mathit{q1} = \mathit{q1sub}, \mathit{q2} = \mathit{q2sub}, \mathit{Dq1} = \mathit{diff}(\mathit{q1sub}, t), \mathit{Dq2} = \mathit{diff}(\mathit{q2sub}, t), \mathit{DDq1} = \mathit{diff}(\mathit{q1sub}, t, t), \mathit{DDq2} = \mathit{diff}(\mathit{q2sub}, t, t), \mathit{eom});$

$$\mathit{temp} := \mathit{solve}\left(\{\mathit{eom\_y}[1], \mathit{eom\_y}[2]\}, \left\{\frac{\mathrm{d}^2}{\mathrm{d}t^2}\,\theta(t), \frac{\mathrm{d}^2}{\mathrm{d}t^2}\,y(t)\right\}\right) :$$

$\mathit{DDtheta\_sol} := \mathit{temp}[1];$
$\mathit{DDy\_sol} := \mathit{collect}\left(\mathit{temp}[2], \left[\mathrm{D}^{(2)}(\phi)(\theta(t))\right]\right);$

#Inserting phi"=f0:

$\mathit{DDtheta\_sol\_DDphi\_sub} := \mathit{subs}\left(\mathrm{D}^{(2)}(\phi)(\theta(t)) = \mathit{DDphi\_sol}, \mathit{DDtheta\_sol}\right);$
$\mathit{DDy\_sol\_DDphi\_sub} := \mathit{subs}\left(\mathrm{D}^{(2)}(\phi)(\theta(t)) = \mathit{DDphi\_sol}, \mathit{DDy\_sol}\right);$

#Simplifying aby with weights:

$\mathit{mu1} := \mathit{p3}$:
$\mathit{mu2} := \mathit{p2}\cdot\cos(\mathrm{theta}(t) - \mathrm{phi}(\mathrm{theta}(t)))$:

$\mathit{sub\_out\_time} := (f) \rightarrow \mathit{subs}(\mathit{diff}(\mathrm{theta}(t), t, t) = \mathit{DDtheta}, \mathit{diff}(\mathrm{theta}(t), t) = \mathit{Dtheta}, \mathrm{theta}(t) = \mathrm{theta}, \mathit{diff}(y(t), t, t) = \mathit{DDy}, \mathit{diff}(y(t), t) = \mathit{Dy}, y(t) = y, f)$:
$\mathit{sub\_out\_error} := (f) \rightarrow \mathit{subs}(\mathit{diff}(y(t), t) = 0, y(t) = 0, \mathit{Dy} = 0, y = 0, f)$:

$\mathit{alpha0} := \mathit{sub\_out\_time}(\mathit{simplify}(\mathit{mu1}\cdot\mathit{alpha1} + \mathit{mu2}\cdot\mathit{alpha2}));$
$\mathit{beta0} := \mathit{sub\_out\_time}\left(\mathit{simplify}\left(\mu 1\cdot\mathit{beta1} + \mathit{mu2}\cdot\mathit{beta2}\right)\right);$
$\mathit{gamma0} := \mathit{sub\_out\_time}(\mathit{simplify}(\mathit{mu1}\cdot\mathit{gamma1} + \mathit{mu2}\cdot\mathit{gamma2}));$

# Verify that equation is 0 on orbit (y=0,Dy = 0)
$\mathit{DDtheta\_sol\_on\_orbit} := \mathit{simplify}(\mathit{sub\_out\_time}(\mathit{sub\_out\_error}(\mathit{rhs}(\mathit{DDtheta\_sol\_DDphi\_sub}))));$
$\mathit{verification} = \mathit{simplify}\left(\mathit{DDtheta\_sol\_on\_orbit}\cdot\mathit{alpha0} + \mathit{beta0}\cdot\mathit{Dtheta}^2 + \mathit{gamma0}\right);$

#Defining right-hand side of perturbated aby equation:

$\mathit{g} := \mathit{sub\_out\_time}\left(\mathit{collect}\left(\mathit{alpha0}\cdot\mathit{rhs}(\mathit{DDtheta\_sol\_DDphi\_sub}) + \mathit{beta0}\cdot\mathit{diff}(\mathrm{theta}(t), t)^2 + \mathit{gamma0}, \left[\mathit{diff}(\mathrm{theta}(t), t, t), \mathit{diff}(\mathrm{theta}(t), t)^2\right]\right)\right);$
$\mathit{g0} := \mathit{sub\_out\_error}(g)$:

$verification = simplify(sub\_out\_error(g));$

$$g\_I := simplify\left(\frac{-diff(g0, \mathrm{theta}) \cdot DDtheta + diff(g0, Dtheta) \cdot Dtheta}{2 \cdot \left(DDtheta^2 + Dtheta^2\right)}\right);$$

$g\_y := simplify(sub\_out\_error(diff(g, y)));$

$g\_Dy := sub\_out\_error(diff(g, Dy));$

#Creating linearized system matrix for transverse coordinates:

$h := collect\left(sub\_out\_time(rhs(DDy\_sol\_DDphi\_sub)), [Dtheta^2]\right);$
$h0 := sub\_out\_error(h) :$

$$h\_I := \frac{-diff(h0, \mathrm{theta}) \cdot DDtheta + diff(h0, Dtheta) \cdot Dtheta}{2 \cdot \left(DDtheta^2 + Dtheta^2\right)};$$

$h\_y := sub\_out\_error(diff(h, y)) :$
$h\_y := collect\left(numer(factor(h\_y)), \{diff, Dtheta^2, DDtheta, \sin, \cos\}\right) / denom(factor(h\_y));$

$h\_Dy := sub\_out\_error(\ diff(h, Dy));$

$A := Matrix(3, 3, 'fill = 0') :$

$$\mathrm{mu} := \frac{2 \cdot Dtheta}{alpha0} :$$

$A_{1,\,1} := \mathrm{mu} \cdot (g\_I - beta0) :$
$A_{1,\,2} := \mathrm{mu} \cdot g\_y :$
$A_{1,\,3} := \mathrm{mu} \cdot g\_Dy :$

$A_{2,\,3} := 1 :$

$A_{3,\,1} := h\_I :$
$A_{3,\,2} := h\_y :$
$A_{3,\,3} := h\_Dy :$

#Creating Matlab code:

$sub\_out\_theta := (f) \rightarrow subs\left(\mathrm{D}^{(3)}(\phi)(\theta) = DDDphi, \mathrm{D}^{(2)}(\phi)(\theta) = DDphi, \mathrm{D}(\phi)(\theta) = Dphi,\right.$
$\left.\mathrm{phi}(\mathrm{theta}) = \mathrm{phi}, f\right) :$

$Matlab(sub\_out\_theta(alpha0), resultname = 'alpha0');$
$Matlab(sub\_out\_theta(beta0), resultname = 'beta0');$
$Matlab(sub\_out\_theta(gamma0), resultname = 'gamma0');$

$A0 := sub\_out\_theta(A) :$

$Matlab\left(A0_{1,\,1}, resultname='a11'\right)$:
$Matlab\left(A0_{1,\,2}, resultname='a12'\right)$:
$Matlab\left(A0_{1,\,3}, resultname='a13'\right)$:
$Matlab\left(A0_{2,\,1}, resultname='a21'\right)$:
$Matlab\left(A0_{2,\,2}, resultname='a22'\right)$:
$Matlab\left(A0_{2,\,3}, resultname='a23'\right)$:
$Matlab\left(A0_{3,\,1}, resultname='a31'\right)$:
$Matlab\left(A0_{3,\,2}, resultname='a32'\right)$:
$Matlab\left(A0_{3,\,3}, resultname='a33'\right)$:

B.6 Transverse Linearization of Discrete Dynamics

```
restart

#Initialize linear algebra
with(LinearAlgebra) :
with(VectorCalculus) :
with(ArrayTools) :
with(CodeGeneration) :
```

```
#Import expressions from other worksheets:
read "../Impact_map/impact_map.m";
```

```
 #Compute Jacobian of impact map:
# Impact map for velocities
Pq := MatrixInverse(rhs(Qp0)).rhs(Qn0);
 # Map for positions
P := Matrix([[0, 1], [1, 0]]);

#Computing Jacobian of complete updating law
 qn := Vector([q1n, q2n]) :
 Dqn := Vector([Dq1n, Dq2n]) :

 qp := P.qn :
 Dqp := subs(q1 = qn(1), q2 = qn(2), Pq).Dqn :

 dF := simplify(Jacobian([qp(1), qp(2), Dqp(1), Dqp(2)], [qn(1), qn(2), Dqn(1), Dqn(2)]));
```

```
#Define impact surface gradients:
 BasisFormat(false) :
 S := cos(q1 + psi) − cos(q2 + psi) = 0 ;
 Sn := subs(q1 = qn(1), q2 = qn(2), lhs(S));
 nn := Vector([diff(Sn, qn(1)), diff(Sn, qn(2)), 0, 0]);
 Sp := subs(q1 = q2n, q2 = q1n, lhs(S));
 np := Vector([diff(Sp, q1n), diff(Sp, q2n), 0, 0]);
```

```
#Define transformation matrix L:
 #Using linearization of y, Dy, I expressions
```

$$L := Matrix\left(\left[\left[-2\cdot DDq1, 0, 2\cdot Dq1, 0\right], \left[-\frac{Dq2}{Dq1}, 1, 0, 0\right], \left[-\frac{\left(DDq2 - \frac{Dq2}{Dq1}\cdot DDq1\right)}{Dq1^2}\cdot Dq1, 0, -\frac{Dq2}{Dq1}, 1\right]\right]\right);$$

```
Lp := subs(q1 = q1p, q2 = q2p, Dq1 = Dq1p, Dq2 = Dq2p, DDq1 = DDq1p, DDq2 = DDq2p, L);

Ln := subs(q1 = q1n, q2 = q2n, Dq1 = Dq1n, Dq2 = Dq2n, DDq1 = DDq1n, DDq2 = DDq2n, L);
```

#Define flow vector m:

$m := Vector([Dq1, Dq2, DDq1, DDq2]);$
$mp := subs(Dq1 = Dq1p, Dq2 = Dq2p, DDq1 = DDq1p, DDq2 = DDq2p, m);$
$mn := subs(Dq1 = Dq1n, Dq2 = Dq2n, DDq1 = DDq1n, DDq2 = DDq2n, m);$

#Compute projection matrices:

$$Pn := simplify\left(\left(IdentityMatrix(4) - \frac{mn.Transpose(nn)}{Transpose(mn).nn}\right).MatrixInverse(Concatenate(1, Ln, Transpose(mn))).\ Concatenate\left(1, IdentityMatrix(3), Vector_{row}([0, 0, 0])\right)\right);$$

$$Pp := simplify\left(Lp.\left(IdentityMatrix(4) - \frac{mp.Transpose(np)}{Transpose(mp).np}\right)\right);$$

#Complete linearization of impact map:

$dF\_TS := Pp.dF.Pn;$

#Creating Matlab code:

$Matlab\left(dF\_TS_{1,\,1}, resultname = 'df\_ts11'\right):$
$Matlab\left(dF\_TS_{1,\,2}, resultname = 'df\_ts12'\right):$
$Matlab\left(dF\_TS_{1,\,3}, resultname = 'df\_ts13'\right):$

$Matlab\left(dF\_TS_{2,\,1}, resultname = 'df\_ts21'\right):$
$Matlab\left(dF\_TS_{2,\,2}, resultname = 'df\_ts22'\right):$
$Matlab\left(dF\_TS_{2,\,3}, resultname = 'df\_ts23'\right):$

$Matlab\left(dF\_TS_{3,\,1}, resultname = 'df\_ts31'\right):$
$Matlab\left(dF\_TS_{3,\,2}, resultname = 'df\_ts32'\right):$
$Matlab\left(dF\_TS_{3,\,3}, resultname = 'df\_ts33'\right):$

C MATLAB Code

The following MATLAB functions are not included in the main text due to their length, or trivial nature. They're included in this appendix to demonstrate generic methods for solving certain mathematical problems in MATLAB, as well as a reference for understanding the procedure employed for finding periodic gaits and assessing their stability.

C.1 Physical Parameters

```
function [mH, m1, m2, a, b, l, g] = physical_parameters
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    mH = 10;     % Mass of hip [kg]
    m1 = 5;      % Mass of first link [kg]
    m2 = 5;      % Mass of second link [kg]
    a = 0.5;     % Length of lower leg [m]
    b = 0.5;     % Length of upper leg [m]
    l = a + b;   % Leg length [m]
    g = 9.81;    % Gravity [m/s^2]

    p1 = mH * l ^ 2 + m1 * b ^ 2 + m2 * l ^ 2;
    p2 = m2 * l * a;
    p3 = m2 * a ^ 2;
    p4 = (m1 * b + m2 * l + mH * l) * g;
    p5 = m2 * a * g;

    p6 = m1 * l ^ 2 + l ^ 2 * mH + b ^ 2 * m2;
    p7 = a * m1 * l;
    p8 = a ^ 2 * m1;
    p9 = a * m1 * b;
    p10 = b * m2 * l + l ^ 2 * mH + m1 * b * l;
    p11 = b * m2 * a;

    psi = 3*pi/180; % Incline of slope [radians]
%     psi = 5*pi/180; % Incline of slope [radians]
```

C.2 Animation of Compass-gait Biped

Plotting animation

```
function [movie_storage,winsize] = animate_walker(time,...
                                        impacts,draw_interval,q,fig)
    global psi;

    q1 = q(:,1);
    q2 = q(:,2);

    % Calculate origins of walker frames
    static_origins = walker_origins(q1,q2);
    offset_origins = travel_offset(impacts, static_origins);

    o1x = offset_origins(:,1);
    o1y = offset_origins(:,2);
    ohx = offset_origins(:,3);
    ohy = offset_origins(:,4);
    o2x = offset_origins(:,5);
    o2y = offset_origins(:,6);
    oex = offset_origins(:,7);
    oey = offset_origins(:,8);
    ocx = offset_origins(:,9);
    ocy = offset_origins(:,10);

    % Select which points to draw based on time vector
    if draw_interval == 0
        time_indexes = 1:length(time);
    else
        time_indexes(1) = 1;
        index = find(time > 0,1);
        i = 1;

        while ~isempty(index)
            time_indexes(i) = index;
            index = find(time >= i*draw_interval,1);

            i = i+1;
        end
    end

    max_x = max(max([ocx, oex, o1x, ohx, o2x]));
    max_y = max(max([ocy, oey, o1y, ohy, o2y]));
    min_x = min(min([ocx, oex, o1x, ohx, o2x]));
    min_y = min(min([ocy, oey, o1y, ohy, o2y]));

    axis_scale = [min_x, max_x min_y max_y];

    winsize = get(fig, 'Position');
    winsize(1:2) = [0 0];
    numframes = length(time_indexes);
    movie_storage = moviein(numframes,fig,winsize);
    set(fig,'NextPlot','replacechildren')

    set(0,'defaulttextinterpreter','Tex')
```

```matlab

    for j=1:numframes
        index = time_indexes(j);

        clf;

        hold on;
        axis(axis_scale);
        axis manual;

        % Frame origins
        plot(o1x(index), o1y(index), 'or');
        plot(ohx(index), ohy(index), 'or');
        plot(o2x(index), o2y(index), 'og');

        % Link indicators
        line([ocx(index) ohx(index)],[ocy(index) ohy(index)],...
            'LineWidth',3, 'Color', 'r');
        line([ohx(index) oex(index)],[ohy(index) oey(index)],...
            'LineWidth',3, 'Color', 'g');

        % Slope indicator
        line([axis_scale(1); axis_scale(2)], ...
            [-axis_scale(1)*tan(psi) -axis_scale(2)*tan(psi)],...
            'LineWidth',3, 'Color', 'k');

        % Time instants
        time_instant = ['\ittime \rm= ' num2str(time(index), '%.2f') 's'];
        text(axis_scale(2),axis_scale(4), time_instant,...
           'VerticalAlignment','top',...
           'HorizontalAlignment','right',...
           'FontSize',16)

       % Angles
       model_data = {sprintf('\\itq_1 \\rm= %.2f',q(index,1)*180/pi);...
           sprintf('\\itq_2 \\rm= %.2f', q(index,2)*180/pi)};
       text(axis_scale(1),axis_scale(3), model_data,...
           'VerticalAlignment','bottom',...
           'HorizontalAlignment','left',...
           'FontSize',16)

       grid on;
       hold off;

        movie_storage(:,j) = getframe(fig,winsize);
    end
end
```

Computing frame origins in Cartesian coordinates

```matlab
function origins = walker_origins(q1,q2)
    [mH, m1, m2, a, b, l, g] = physical_parameters;

    o1x = -sin(q1) * b;
    o1y = cos(q1) * b;
    ohx = -sin(q1) * a - sin(q1) * b;
    ohy = cos(q1) * a + cos(q1) * b;
    o2x = -l * sin(q1) + a * sin(q2);
    o2y = l * cos(q1) - a * cos(q2);
    oex = sin(q2) * b - l * sin(q1) + a * sin(q2);
    oey = -cos(q2) * b + l * cos(q1) - a * cos(q2);

    o1 = [o1x, o1y];
    oh = [ohx, ohy];
    o2 = [o2x, o2y];
    oe = [oex, oey];

    origins = [o1, oh, o2, oe];
end
```

Adjusting frame origins for travel along slope

```matlab
function offset_origins = travel_offset(impacts, static_origins)

% Adding contact point for stance leg
    offset_origins = [static_origins, zeros(size(static_origins,1),2)];

    for index = 1:size(impacts,1)
        impact_index = impacts(index,2);

        offset_origins(impact_index+1:end,1:2:end) = ...
            offset_origins(impact_index+1:end,1:2:end) ...
            + static_origins(impact_index,7);

        offset_origins(impact_index+1:end,2:2:end) = ...
            offset_origins(impact_index+1:end,2:2:end) ...
            + static_origins(impact_index,8);
    end

end
```

C.3 Searching for Periodic Gaits

Implementation of $\phi''(\theta) = f_0(E_0, \theta, \phi(\theta), \phi'(\theta))$

```
function Dx = f0(theta, x, E0)
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    phi = x(1);
    Dphi = x(2);

    DDphi = -0.1e1 / (-p1 * p3 + cos(-theta + phi) ^ 2 ...
        * p2 ^ 2) * (-cos(-theta + phi) * p2 ^ 2 ...
        * sin(-theta + phi) * Dphi + p3 * sin(-theta + phi) ...
        * p2 * Dphi ^ 3 - cos(-theta + phi) * p2 ^ 2 ...
        * sin(-theta + phi) * Dphi ^ 2 + p1 * sin(-theta + phi) * p2)...
        - 0.1e1 / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2)...
        / (E0 - cos(theta) * p4 - p5 + cos(phi) * p5 + p4) ...
        * (-sin(theta) * p4 * p3 * Dphi + sin(theta) * p4 * p2 ...
        * cos(-theta + phi) + sin(phi) * p5 ...
        *cos(-theta + phi) * p2 * Dphi - sin(phi) * p5 * p1)...
        * (p1 + p3 * Dphi ^ 2 - 0.2e1 * p2 ...
        * cos(-theta + phi) * Dphi) / 0.2e1;

    Dx = [Dphi, DDphi]';

end
```

Quadratic equation solver for finding d using E_0

```
function [d, error_flag] = solving_E0_for_d(a,b,quad_solution_selector)
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    error_flag = 0;

    %% Solving for d = Dq1T using the quadratic formula
    % E0d = A2*d^2 + B2*d + C2

    A2 = (p8 * p10 - p9 * p7) ^ 2 / p8 ^ 2 / p11 ^ 2 *...
        cos((2 * a + 2 * psi)) ^ 2 * p3 / 0.2e1 - p9 ^ 2 ...
        / p8 ^ 2 * p3 / 0.2e1 + p1 / 0.2e1 - (p8 * p10 - p9 * p7)...
        / p8 / p11 * cos((2 * a + 2 * psi)) ^ 2 * p2;
    B2 = -p9 / p8 * cos((2 * a + 2 * psi)) * p2 * b ...
        - (p7 ^ 2 / p8 / p11 * cos((2 * a + 2 * psi)) ^ 2 ...
        - p6 / p11) * b * cos((2 * a + 2 * psi)) * p2 ...
        + (p7 ^ 2 / p8 / p11 * cos((2 * a + 2 * psi)) ^ 2 ...
        - p6 / p11) * b * (p8 * p10 - p9 * p7) / p8 / p11 ...
        * cos((2 * a + 2 * psi)) * p3 +...
        b * p7 * cos((2 * a + 2 * psi)) * p9 / p8 ^ 2 * p3;
    C2 = -p1 * b ^ 2 / 0.2e1 - cos(a) * p4 + ...
        (p7 ^ 2 / p8 / p11 * cos(0.2e1 * a + ...
        (2 * psi)) ^ 2 - p6 / p11) ^ 2 * b ^ 2 ...
        * p3 / 0.2e1 - cos(a) * p5 + b ^ 2 * p7 ...
        * cos(0.2e1 * a + (2 * psi)) ^ 2 / p8 * p2...
        + cos(a + (2 * psi)) * p4 + cos(a + (2 * psi)) ...
        * p5 - b ^ 2 * p7 ^ 2 * cos(0.2e1 * a ...
        + (2 * psi)) ^ 2 / p8 ^ 2 * p3 / 0.2e1;


    % Standard solution of the quadratic equation
    % d = (-B2 +- sqrt(B2^2-4*A2*C2))/(2*A2)
    D2 = B2^2-4*A2*C2;

    if D2 < 0 % Imaginary solution
        error_flag = 1;
        d = NaN;
        return;
    end

    % Switching between the 2 possible solutions of the
    % quadratic equation
    switch quad_solution_selector
        case 1
            d = (-B2 + sqrt(D2))/(2*A2);
        case 2
            d = (-B2 - sqrt(D2))/(2*A2);
        otherwise
            d = NaN;
            error_flag = 1;
    end
end
```

C.4 Checking Stability Using Transverse Linearization

Main function that returns the eigenvalues of $\Xi(x_\star^-, x_\star^+)$

```
function eigenvalues = run_stability_check(a,b,c,d,e,f,g,h,E0)
    run physical_parameters;

    fprintf('************************************\n');
    fprintf('CHECKING STABILITY OF PERIODIC GAIT\n');

    %% Solving DDphi = f0 and alpha*DDtheta + beta*Dtheta^2 + gamma = 0
    %  for variable values along target trajectory

    fprintf('*****************************************************\n');
    fprintf('Computing required parameter values along trajectory: ');

    options = odeset('reltol',1e-9,'abstol',1e-9);
    step_size = 0.0001;

    % Finding phi and associated values
    [time_phi, xout] = ode45(@(theta,x) f0(theta,x,E0),...
                                  a:-step_size:c, [e; f/b], options);
    phi = xout(:,1);
    Dphi= xout(:,2);

    % Removing possible NaN results
    for i=1:length(time_phi),
        if (~isnan(phi(i))) && (~isnan(Dphi(i))),
            theta_inputs_to_phi(i) = time_phi(i);
            phi_evaluated_at_theta(i) = phi(i);
            Dphi_evaluated_at_theta(i) = Dphi(i);
        else
            break
        end
    end

    theta_inputs_to_phi(1) = a;
    theta_inputs_to_phi(end) = c;

    phi_evaluated_at_theta(1) = e;
    phi_evaluated_at_theta(end) = g;

    Dphi_evaluated_at_theta(1) = f/b;
    Dphi_evaluated_at_theta(end) = h/d;

    % Finding period of gait, theta and associated values using aby -
    % equation
    liberal_period_estimate = 2;
    options = odeset('reltol',1e-9,'abstol',1e-9,'Events',...
                          @(t,x) aby_weighted_event(t,x,c));

    [time_aby, xout] = ode45(@(t,x) aby_weighted(t,x, ...
        theta_inputs_to_phi, phi_evaluated_at_theta, ...
        Dphi_evaluated_at_theta), ...
        0:step_size:liberal_period_estimate, [a; b], options);
    theta = xout(:,1);
    Dtheta = xout(:,2);

    % Removing possible NaN results
    for i=1:length(time_aby),
        if (~isnan(theta(i))) && (~isnan(Dtheta(i)))
            time_inputs_to_theta(i) = time_aby(i);
            theta_evaluated_at_time(i) = theta(i);
```

```
                Dtheta_evaluated_at_time(i) = Dtheta(i);
            else
                break
            end
        end

        gait_period = time_aby(end);

        time_inputs_to_theta(1) = 0;
        time_inputs_to_theta(end) = gait_period;

        theta_evaluated_at_time(1) = a;
        theta_evaluated_at_time(end) = c;

        Dtheta_evaluated_at_time(1) = b;
        Dtheta_evaluated_at_time(end) = d;

        fprintf('DONE!\n');
        fprintf('*********************************************************\n');

        %% Calculating eigenvalues of transition matrix
        % and assessing stability of perioid gait

        % Integrating linearized transversal dynamics until impact
        transverse_dynamics_shortened = @(t, zeta) ...
            transverse_linearization_continuous(t, zeta, ...
            theta_inputs_to_phi, phi_evaluated_at_theta, ...
            Dphi_evaluated_at_theta, time_inputs_to_theta, ...
            theta_evaluated_at_time, Dtheta_evaluated_at_time, E0);

        options = odeset('reltol',1e-9,'abstol',1e-9);
        fprintf('Integrating linearized transverse coordinates\n');
        fprintf('First integration: ');

        [t,zeta_n] = ode45(transverse_dynamics_shortened,...
            [0 gait_period],[1,0,0]',options);
        zeta_n_1 = zeta_n(end,:)';

        fprintf(' DONE!\n');
        fprintf('Second integration: ');

        [t,zeta_n] = ode45(transverse_dynamics_shortened,...
            [0 gait_period],[0,1,0]',options);
        zeta_n_2 = zeta_n(end,:)';

        fprintf('DONE!\n');
        fprintf('Third integration: ');

        [t,zeta_n] = ode45(transverse_dynamics_shortened,...
            [0 gait_period],[0,0,1]',options);
        zeta_n_3 = zeta_n(end,:)';

        fprintf(' DONE!\n');

        % Calculating update matrix
        transverse_update_shortened = @(zeta_n) ...
            transverse_linearization_discrete(zeta_n, theta_inputs_to_phi,...
            phi_evaluated_at_theta, Dphi_evaluated_at_theta,...
            time_inputs_to_theta, theta_evaluated_at_time,...
            Dtheta_evaluated_at_time, gait_period, E0);

        update_matrix = [transverse_update_shortened(zeta_n_1), ...
            transverse_update_shortened(zeta_n_2), ...
            transverse_update_shortened(zeta_n_3)];

        % Checking eigenvalues of update matrix
        eigenvalues = sort(eig(update_matrix));

        fprintf('*****************************************************\n');
        fprintf('The found periodic gait is ');
        if max(abs(eigenvalues)) < 1
            fprintf('STABLE');
        else
            fprintf('UNSTABLE');
        end
        fprintf(' with eigenvalues\n');

        for i=1:3
            fprintf('%d. Eigenvalue = ',i);

            if real(eigenvalues(i)) >= 0
                fprintf(' ');
            end

            fprintf('%0.3f',real(eigenvalues(i)));
            if imag(eigenvalues(i)) < 0
                fprintf(' - ');
            else
                fprintf(' + ');
            end
            fprintf('%0.3fi', abs(imag(eigenvalues(i))));

            if abs(eigenvalues(i)) < 1
                fprintf('   OK!\n');
            else
                fprintf('   Outside unit circle!\n');
            end
        end
        fprintf('*****************************************************\n');
end
```

Implementation of $\alpha_0(\theta)\ddot{\theta} + \beta_0(\theta)\dot{\theta}^2 + \gamma_0(\theta) = 0$

```
function Dx = aby_weighted(t,x, theta_inputs_to_phi,...
                phi_evaluated_at_theta, Dphi_evaluated_at_theta)
    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    theta = x(1);
    Dtheta = x(2);
    phi = interp1(theta_inputs_to_phi,...
        phi_evaluated_at_theta,theta,'spline');
    Dphi = interp1(theta_inputs_to_phi,...
        Dphi_evaluated_at_theta,theta,'spline');

    alpha0 = p1 * p3 - cos(-theta + phi) ^ 2 * p2 ^ 2;
    beta0 = sin(-theta + phi) * p2 ...
        * (Dphi ^ 2 * p3 - p2 * cos(-theta + phi));
    gamma0 = -sin(theta) * p4 * p3 ...
        + p2 * cos(-theta + phi) * sin(phi) * p5;

    Dx = [Dtheta; -(beta0*Dtheta^2 + gamma0)/alpha0];
end
```

Implementation of $\dot{\zeta}(t) = A(t)\zeta(t)$

```
function Dzeta = transverse_linearization_continuous(t, zeta, ...
                theta_inputs_to_phi, phi_evaluated_at_theta, ...
                Dphi_evaluated_at_theta, time_inputs_to_theta, ...
                theta_evaluated_at_time, Dtheta_evaluated_at_time, E0)

    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    % Solving for virtual holonomic constraints
    theta = interp1(time_inputs_to_theta,theta_evaluated_at_time, ...
                    t,'spline');
    Dtheta = interp1(time_inputs_to_theta,Dtheta_evaluated_at_time, ...
                    t,'spline');
    temp = aby_weighted(t,[theta; Dtheta], theta_inputs_to_phi, ...
                         phi_evaluated_at_theta, Dphi_evaluated_at_theta);
    DDtheta = temp(2);

    phi = interp1(theta_inputs_to_phi,phi_evaluated_at_theta,...
                    theta,'spline');
    Dphi = interp1(theta_inputs_to_phi,Dphi_evaluated_at_theta,...
                    theta,'spline');
    temp = f0(theta, [phi; Dphi], E0);
    DDphi = temp(2);

    % Linearized transversal dynamics
    a11 = -0.2e1 * Dtheta / (p1 * p3 - cos(-theta + phi) ^ 2 * p2 ^ 2)...
        * sin(-theta + phi) * p2 * (Dphi ^ 2 * p3 - p2 ...
        * cos(-theta + phi));
    a12 = 0.2e1 * Dtheta / (p1 * p3 - cos(-theta + phi) ^ 2 * p2 ^ 2) ...
        * p2 * (-0.2e1 * Dtheta ^ 2 * p2 ^ 2 * cos(-theta + phi) ...
        * Dphi ^ 2 * p3 + Dtheta ^ 2 * cos(-theta + phi) ^ 3 * ...
        p2 ^ 2 * Dphi ^ 2 * p3 + Dtheta ^ 2 * p2 ^ 3 *...
        cos(-theta + phi) ^ 2 + Dtheta ^ 2 * p1 * p3 ^ 2 ...
        * cos(-theta + phi) * Dphi ^ 2 + Dtheta ^ 2 * p1 * p3 ...
        * p2 - 0.2e1 * Dtheta ^ 2 * p1 * p3 * p2 *...
        cos(-theta + phi) ^ 2 + 0.2e1 * p2 * cos(-theta + phi)...
        * sin(-theta + phi) * sin(theta) * p4 * p3 - p2 ^ 2 ...
        * cos(-theta + phi) ^ 2 * sin(-theta + phi) * sin(phi) ...
        * p5 - p5 * p1 * p3 * sin(-theta + phi) * sin(phi) ...
        + p5 * p1 * p3 * cos(-theta + phi) * cos(phi) - p5 ...
        * cos(-theta + phi) ^ 3 * p2 ^ 2 * cos(phi)) ...
        / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2);
    a13 = 0.4e1 * Dtheta ^ 2 / ...
        (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2)...
        * sin(-theta + phi) * p2 * Dphi * p3;
    a21 = 0;
    a22 = 0;
    a23 = 1;
    a31 = (-((-cos(theta) * p4 * p3 * Dphi - sin(theta) * p4 * p3...
        * DDphi + cos(theta) * p4 * p2 * cos(-theta + phi)...
        + sin(theta) * p4 * p2 * sin(-theta + phi) + sin(phi)...
        * p5 * DDphi * cos(-theta + phi) * p2 + sin(phi) ...
        * p5 * Dphi * sin(-theta + phi) * p2) ...
        / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2) ...
        / (E0 - cos(theta) * p4 - p5 + cos(phi) * p5 + p4) ...
        * (-0.2e1 * Dphi * cos(-theta + phi) * p2 + Dphi ^ 2 ...
        * p3 + p1) * Dtheta ^ 2 / 0.2e1 - (-sin(theta) * p4 ...
        * p3 * Dphi + sin(theta) * p4 * p2 * cos(-theta + phi)...
        + sin(phi) * p5 * Dphi * cos(-theta + phi)...
        * p2 - sin(phi) * p5 * p1) / (-p1 * p3 ...
        + cos(-theta + phi) ^ 2 * p2 ^ 2) ^ 2 / (E0 - cos(theta)...
        * p4 - p5 + cos(phi) * p5 + p4) * (-0.2e1 * Dphi ...
        * cos(-theta + phi) * p2 + Dphi ^ 2 * p3 + p1) ...
        * Dtheta ^ 2 * cos(-theta + phi) * p2 ^ 2 ...
        * sin(-theta + phi) - (-sin(theta) * p4 * p3 * Dphi ...
        + sin(theta) * p4 * p2 * cos(-theta + phi) + sin(phi)...
        * p5 * Dphi * cos(-theta + phi) * p2 - sin(phi) * p5 * p1)...
        / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2) ...
        / (E0 - cos(theta) * p4 - p5 + cos(phi) * p5 + p4) ^ 2 ...
        * (-0.2e1 * Dphi * cos(-theta + phi) * p2...
        + Dphi ^ 2 * p3 + p1) * Dtheta ^ 2 * sin(theta) ...
        * p4 / 0.2e1 + (-sin(theta) * p4 * p3 * Dphi ...
        + sin(theta) * p4 * p2 * cos(-theta + phi) + sin(phi)...
        * p5 * Dphi * cos(-theta + phi) * p2 - sin(phi) * p5 * p1)...
        / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2) ...
        / (E0 - cos(theta) * p4 - p5 + cos(phi) * p5 + p4) ...
        * (-0.2e1 * DDphi * cos(-theta + phi) * p2 - 0.2e1 ...
        * Dphi * sin(-theta + phi) * p2 + 0.2e1 * Dphi * p3 ...
        * DDphi) * Dtheta ^ 2 / 0.2e1 - (-cos(theta) * p4 * p3 ...
        * Dphi - sin(theta) * p4 * p3 * DDphi + cos(theta) * p4 ...
        * p2 * cos(-theta + phi) + sin(theta) * p4 * p2 ...
        * sin(-theta + phi) + sin(phi) * p5 * DDphi ...
        * cos(-theta + phi) * p2 + sin(phi) * p5 * Dphi ...
        * sin(-theta + phi) * p2) / (-p1 * p3 ...
        + cos(-theta + phi) ^ 2 * p2 ^ 2) + 0.2e1 * (-sin(theta)...
        * p4 * p3 * Dphi + sin(theta) * p4 * p2 * cos(-theta + phi)...
        + sin(phi) * p5 * Dphi * cos(-theta + phi) * p2 ...
        - sin(phi) * p5 * p1) / (-p1 * p3 + cos(-theta + phi) ^ 2 ...
        * p2 ^ 2) ^ 2 * cos(-theta + phi) * p2 ^ 2 ...
        * sin(-theta + phi)) * DDtheta + (-sin(theta) * p4 * p3 ...
        * Dphi + sin(theta) * p4 * p2 * cos(-theta + phi) + sin(phi)...
        * p5 * Dphi * cos(-theta + phi) * p2 - sin(phi) * p5 * p1)...
        / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2) ...
        / (E0 - cos(theta) * p4 - p5 + cos(phi) * p5 + p4) ...
        * (-0.2e1 * Dphi * cos(-theta + phi) * p2 + Dphi ^ 2 ...
        * p3 + p1) * Dtheta ^ 2) / (0.2e1 * DDtheta ^ 2 ...
        + 0.2e1 * Dtheta ^ 2);
    a32 = ((Dphi * p2 ^ 4 + p2 ^ 4 * Dphi ^ 2) * Dtheta ^ 2 ...
        * cos(-theta + phi) ^ 4 + (-p2 ^ 3 * cos(phi) * Dphi * p5 ...
        + (-p3 * Dphi ^ 3 * p2 ^ 3 - p2 ^ 3 * p1) * Dtheta ^ 2) ...
        * cos(-theta + phi) ^ 3 + (cos(phi) * p5 * p1 * p2 ^ 2 ...
        - p2 ^ 3 * sin(-theta + phi) * sin(theta) * p4 ...
        + (Dphi * p2 ^ 4 + p2 ^ 4 * Dphi ^ 2) * Dtheta ^ 2 ...
        * sin(-theta + phi) ^ 2 - Dphi * p2 ^ 3 * sin(-theta + phi) ...
        * sin(phi) * p5 + (-Dphi * p2 ^ 2 * p1 * p3 - p2 ^ 2 ...
        * Dphi ^ 2 * p1 * p3) * Dtheta ^ 2) * cos(-theta + phi) ^ 2 ...
        + (p2 * cos(phi) * Dphi * p5 * p1 * p3 + 0.2e1 * p2 ^ 2 ...
        * sin(-theta + phi) * sin(theta) * p4 * p3 * Dphi + (-0.2e1 ...
        * p2 ^ 3 * p1 - 0.2e1 * p3 * Dphi ^ 3 * p2 ^ 3) * Dtheta ^ 2 ...
        * sin(-theta + phi) ^ 2 + 0.2e1 * p2 ^ 2 * sin(-theta + phi) ...
        * sin(phi) * p5 * p1 + (p2 * p1 ^ 2 * p3 + p3 ^ 2 * Dphi ^ 3 ...
        * p2 * p1) * Dtheta ^ 2) * cos(-theta + phi) ...
        - cos(phi) * p5 * p1 ^ 2 * p3 - p2 * sin(-theta + phi) ...
        * sin(theta) * p4 * p1 * p3 + (Dphi * p2 ^ 2 * p1 * p3 ...
        + p2 ^ 2 * Dphi ^ 2 * p1 * p3) * Dtheta ^ 2 ...
        * sin(-theta + phi) ^ 2 - Dphi * p2 * sin(-theta + phi) ...
        * sin(phi) * p5 * p1 * p3) / (-p1 * p3 ...
        + cos(-theta + phi) ^ 2 * p2 ^ 2) ^ 2;
    a33 = -(0.2e1 * p3 * Dphi ^ 2 * sin(-theta + phi) * p2 ...
        * Dtheta - 0.2e1 * Dphi * p2 ^ 2 * cos(-theta + phi)...
        * Dtheta * sin(-theta + phi)) ...
        / (-p1 * p3 + cos(-theta + phi) ^ 2 * p2 ^ 2);

    A = [a11 a12 a13; a21 a22 a23; a31 a32 a33];

    Dzeta = A*zeta;
end
```

Implementation of $\left(d^{TS}F\right)_{(x_\star(t=T))} = P^+_{n^+}\, dF_{(x_\star(t=T))}\, P^-_{n^-}$

```
function zeta_p = transverse_linearization_discrete(zeta_n, ...
    theta_inputs_to_phi, phi_evaluated_at_theta, ...
    Dphi_evaluated_at_theta, time_inputs_to_theta, ...
    theta_evaluated_at_time, Dtheta_evaluated_at_time, gait_period,E0)

    global p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 psi;

    %% Right before impact t=T
    thetaT = interp1(time_inputs_to_theta, theta_evaluated_at_time, ...
        gait_period, 'spline');
    DthetaT = interp1(time_inputs_to_theta, Dtheta_evaluated_at_time, ...
        gait_period, 'spline');
    temp = aby_weighted(gait_period, [thetaT,DthetaT], ...
        theta_inputs_to_phi, phi_evaluated_at_theta, ...
        Dphi_evaluated_at_theta);
    DDthetaT = temp(2);

    phiT = interp1(theta_inputs_to_phi, phi_evaluated_at_theta, ...
        thetaT, 'spline');
    DphiT = interp1(theta_inputs_to_phi, Dphi_evaluated_at_theta, ...
        thetaT, 'spline');
    temp = f0(thetaT,[phiT,DphiT], E0);
    DDphiT = temp(2);

    %% Right after impact t=0
    theta0 = interp1(time_inputs_to_theta, theta_evaluated_at_time, ...
        0, 'spline');
    Dtheta0 = interp1(time_inputs_to_theta, Dtheta_evaluated_at_time,...
        0, 'spline');
    temp = aby_weighted(0, [theta0,Dtheta0], theta_inputs_to_phi,...
        phi_evaluated_at_theta, Dphi_evaluated_at_theta);
    DDtheta0 = temp(2);

    phi0 = interp1(theta_inputs_to_phi, phi_evaluated_at_theta,...
        theta0, 'spline');
    Dphi0 = interp1(theta_inputs_to_phi, Dphi_evaluated_at_theta,...
        theta0, 'spline');
    temp = f0(theta0,[phi0,Dphi0], E0);
    DDphi0 = temp(2);

    %% Solving for generalized coordinates
    q1p = theta0;
    Dq1p = Dtheta0;
    DDq1p = DDtheta0;

    q2p = phi0;
    Dq2p = Dphi0*Dtheta0;
    DDq2p = DDphi0*Dtheta0^2 + Dphi0*DDtheta0;

    q1n = thetaT;
    Dq1n = DthetaT;
    DDq1n = DDthetaT;

    q2n = phiT;
    Dq2n = DphiT*DthetaT;
    DDq2n = DDphiT*DthetaT^2 + DphiT*DDthetaT;

    %% Linearized impact map
    df_ts11 = -Dq1p * cos(q1n - q2n) * (p8 * p10 - p7 * p9) ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2) / Dq1n ...
        + Dq1p * p8 / (-p8 * p6 + cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2) * p11 * Dq2n / Dq1n ^ 2;
    df_ts12 = -0.2e1 * Dq1p * sin(q1n - q2n) ...
        * (Dq1n * p8 * cos(q1n - q2n) ^ 2 * p7 ^ 2 ...
        * p10 + Dq1n * p8 ^ 2 * p10 * p6 - Dq1n * p7 ...
        * p9 * p8 * p6 - Dq1n * p7 ^ 3 * p9 ...
        * cos(q1n - q2n) ^ 2 - 0.2e1 * p8 * p11 * Dq2n ...
        * cos(q1n - q2n) * p7 ^ 2) / (p8 ^ 2 * p6 ^ 2 ...
        - 0.2e1 * p8 * p6 * cos(q1n - q2n) ^ 2 * p7 ^ 2 ...
        + cos(q1n - q2n) ^ 4 * p7 ^ 4) * sin(q2n + psi) ...
        * Dq1n / (Dq1n * sin(q1n + psi) - Dq2n ...
        * sin(q2n + psi)) + (-(2 * DDq1p) + 0.2e1 ...
        * Dq1p * sin(q1n - q2n) * (Dq1n * p8 ...
        * cos(q1n - q2n) ^ 2 * p7 ^ 2 * p10 + Dq1n ...
        * p8 ^ 2 * p10 * p6 - Dq1n * p7 * p9 * p8 * p6 ...
        - Dq1n * p7 ^ 3 * p9 * cos(q1n - q2n) ^ 2 ...
        - 0.2e1 * p8 * p11 * Dq2n * cos(q1n - q2n) ...
        * p7 ^ 2) / (p8 ^ 2 * p6 ^ 2 - 0.2e1 * p8 * p6 ...
        * cos(q1n - q2n) ^ 2 * p7 ^ 2 + cos(q1n - q2n) ^ 4 ...
        * p7 ^ 4)) * Dq1n * sin(q1n + psi) / (Dq1n ...
        * sin(q1n + psi) - Dq2n * sin(q2n + psi)) ...
        - 0.2e1 * Dq1p * cos(q1n - q2n) * (p8 * p10 ...
        - p7 * p9) / (-p8 * p6 + cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2) * sin(q2n + psi) * DDq1n / (Dq1n ...
        * sin(q1n + psi) - Dq2n * sin(q2n + psi)) + 0.2e1 ...
        * Dq1p * p8 / (-p8 * p6 + cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2) * p11 * sin(q2n + psi) * DDq2n / (Dq1n ...
        * sin(q1n + psi) - Dq2n * sin(q2n + psi));
    df_ts13 = 0.2e1 * Dq1p * p8 ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2) * p11;
    df_ts21 = 0;
    df_ts22 = sin(q2n + psi) * Dq1n ...
        / (Dq1n * sin(q1n + psi) - Dq2n ...
        * sin(q2n + psi)) - Dq2p / Dq1p * Dq1n ...
        * sin(q1n + psi) / (Dq1n * sin(q1n + psi) ...
        - Dq2n * sin(q2n + psi));
    df_ts23 = 0;
    df_ts31 = (Dq2p / Dq1p * cos(q1n - q2n)...
        * (p8 * p10 - p7 * p9) / (-p8 * p6 ...
        + cos(q1n - q2n) ^ 2 * p7 ^ 2) ...
        - (cos(q1n - q2n) ^ 2 * p7 * p10 - p9 * p6) ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2)) ...
        / Dq1n / 0.2e1 + (-Dq2p / Dq1p * p8 ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2) ...
        * p11 + cos(q1n - q2n) * p7 / (-p8 * p6 ...
        + cos(q1n - q2n) ^ 2 * p7 ^ 2) * p11) ...
        * Dq2n / Dq1n ^ 2 / 0.2e1;
    df_ts32 = (Dq2p / Dq1p * sin(q1n - q2n) ...
        * (Dq1n * p8 * cos(q1n - q2n) ^ 2 * p7 ^ 2 ...
        * p10 + Dq1n * p8 ^ 2 * p10 * p6 - Dq1n * p7 ...
        * p9 * p8 * p6 - Dq1n * p7 ^ 3 * p9 ...
        * cos(q1n - q2n) ^ 2 - 0.2e1 * p8 * p11 ...
        * Dq2n * cos(q1n - q2n) * p7 ^ 2) / (p8 ^ 2 ...
        * p6 ^ 2 - 0.2e1 * p8 * p6 * cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2 + cos(q1n - q2n) ^ 4 * p7 ^ 4) ...
        - sin(q1n - q2n) * p7 * (0.2e1 * cos(q1n - q2n) ...
        * p6 * Dq1n * p8 * p10 - 0.2e1 * cos(q1n - q2n) ...
        * p6 * Dq1n * p7 * p9 - p11 * Dq2n * p8 * p6 ...
        - cos(q1n - q2n) ^ 2 * p7 ^ 2 * p11 * Dq2n) ...
        / (p8 ^ 2 * p6 ^ 2 - 0.2e1 * p8 * p6 ...
        * cos(q1n - q2n) ^ 2 * p7 ^ 2 ...
        + cos(q1n - q2n) ^ 4 * p7 ^ 4)) * sin(q2n + psi) ...
        * Dq1n / (Dq1n * sin(q1n + psi) - Dq2n ...
        * sin(q2n + psi)) + ((-DDq2p * Dq1p ...
        + Dq2p * DDq1p) / Dq1p ^ 2 - Dq2p / Dq1p ...
        * sin(q1n - q2n) * (Dq1n * p8 * cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2 * p10 + Dq1n * p8 ^ 2 * p10 * p6 ...
        - Dq1n * p7 * p9 * p8 * p6 - Dq1n * p7 ^ 3 ...
        * p9 * cos(q1n - q2n) ^ 2 - 0.2e1 * p8 * p11 ...
        * Dq2n * cos(q1n - q2n) * p7 ^ 2) / (p8 ^ 2 ...
        * p6 ^ 2 - 0.2e1 * p8 * p6 * cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2 + cos(q1n - q2n) ^ 4 * p7 ^ 4) ...
        + sin(q1n - q2n) * p7 * (0.2e1 * cos(q1n - q2n) ...
        * p6 * Dq1n * p8 * p10 - 0.2e1 * cos(q1n - q2n) ...
        * p6 * Dq1n * p7 * p9 - p11 * Dq2n * p8 * p6 ...
        - cos(q1n - q2n) ^ 2 * p7 ^ 2 * p11 * Dq2n) ...
        / (p8 ^ 2 * p6 ^ 2 - 0.2e1 * p8 * p6 ...
        * cos(q1n - q2n) ^ 2 * p7 ^ 2 + cos(q1n - q2n) ^ 4 ...
        * p7 ^ 4)) * Dq1n * sin(q1n + psi) ...
        / (Dq1n * sin(q1n + psi) - Dq2n * sin(q2n + psi)) ...
        + (Dq2p / Dq1p * cos(q1n - q2n) * (p8 * p10 - p7 * p9) ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2) ...
        - (cos(q1n - q2n) ^ 2 * p7 * p10 - p9 * p6) ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2)) ...
        * sin(q2n + psi) * DDq1n / (Dq1n * sin(q1n + psi) ...
        - Dq2n * sin(q2n + psi)) + (-Dq2p / Dq1p * p8 ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2) * p11 ...
        + cos(q1n - q2n) * p7 / (-p8 * p6 + cos(q1n - q2n) ^ 2 ...
        * p7 ^ 2) * p11) * sin(q2n + psi) * DDq2n ...
        / (Dq1n * sin(q1n + psi) - Dq2n * sin(q2n + psi));
    df_ts33 = -Dq2p / Dq1p * p8 / (-p8 * p6 ...
        + cos(q1n - q2n) ^ 2 * p7 ^ 2) * p11 ...
        + cos(q1n - q2n) * p7 ...
        / (-p8 * p6 + cos(q1n - q2n) ^ 2 * p7 ^ 2) * p11;


    dF_TS = [df_ts11 df_ts12 df_ts13; df_ts21 ...
        df_ts22 df_ts23; df_ts31 df_ts32 df_ts33];

    zeta_p = dF_TS*zeta_n;
end
```