NTNU – Trondheim
Norwegian University of
Science and Technology

Stabilization of Gaits for the SemiQuad Robot

Henrik Røst Breivik

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

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

This thesis focuses on studying orbital stability for a biped walking robot called SemiQuad, developed and built at École Centrale de Nantes, France. Using previously designed walking motions, the focus was to develop an exponentially orbital stabilizing controller for the biped.

Assignment given: 20. August 2012
Supervisor: Prof. Anton Shiriaev, ITK

Preface

This thesis is made to document the project performed during the 10th and final semester of the Master of Engineering Cybernetics degree at NTNU. I have chosen this topic because of a personal interest in walking robots. The project scope allowed me to explore this interest, while using theoretical methods that I enjoy working with.

I would like to thank my supervisor Prof. Anton Shiriaev, for giving me this project, and also for the long discussions and encouraging meetings. A thanks also goes to Prof. Sergei Gusev for his help in designing the controller for the linearized system. I would also like to thank my girlfriend for helping me proofread the thesis, and for her support throughout the semester.

Henrik Røst Breivik
Trondheim, January 13, 2013

Abstract

It was desired to make an orbitally stabilizing controller for a series of designed gaits for the SemiQuad robot. The designed gaits had virtual constraints of the second and third order. It was found that the gaits that had virtual constraints of third order broke a design restriction, which made the stance foot lift off the ground. The second order gaits did not break this restriction. A constant gain feedback controller was found to not be able to orbitally stabilize the gait. A linearization was made using the method of transverse linearization for one of the gaits with second degree constraints, and a controller was designed for this linearization. This controller stabilized the linearized system. The controller was not found to be able to stabilize the original nonlinear system. It was concluded that the gaits were of a too complicated nature to stabilize with this controller. A new gait was therefore created, which was of a simpler nature with a prolonged double support phase. This new gait was stabilized with a constant gain controller.

Sammendrag

Det var ønsket å lage en orbitalt stabiliserende regulator for en rekke bevegelser som var designet for roboten SemiQuad. De designede bevegelsene brukte virtuelle beskrankninger av andre og tredje grad. Det ble funnet at bevegelsene med beskrankninger av tredje grad brøt med en designregel, som gjorde at støttefoten ble løftet opp fra bakken underveis i steget. Denne designregelen ble ikke brutt av bevegelsene med beskrankninger av andre grad. Det ble funnet at en regulator med tilbakekopling og konstant forsterkning ikke kunne orbitalt stabilisere gåbevegelsen. En linearisering ble laget for en av bevegelsene med beskrankninger av andre grad ved bruk av metoden transverse linearization. En regulator ble laget for denne lineariseringen, og det ble vist at den stabiliserte det lineariserte systemet. Denne regulatoren ble vist til å ikke kunne stabilisere det ulineære systemet. Det ble konkludert med at de designede gåbevegelsene hadde en for komplisert dynamikk til å kunne stabiliseres med denne regulatoren. Det ble derfor laget en ny bevegelse, som inneholdt en forlenget fase med begge bena i kontakt med bakken. Denne nye bevegelsen ble stabilisert med en regulator med tilbakekopling og konstant forsterkning.

Contents

Chapter 1

Introduction

The science fiction picture of humanoid robots has given a little insight to what robots might be able to do in the future. As of now, the research on robots includes many fields of interest. Fields of study include among others artificial intelligence and biomechanics. The development of these fields are creating a need for understanding the human body, and the way it works. If a robotic prosthetic arm is to be connected to the human body, there is a need to know how an arm works, down to the nerve signals being sent to and from the arm. This study of the body naturally involves how the body moves, which brings focus to legged robotic walkers.

The research interest in legged robotic walkers has increased especially in the last decade, with many different of robots being designed and created. Examples of this include the humanoid ASIMO robot produced by Honda, the RABBIT developed in France[1], as well as LittleDog and BigDog from Boston Dynamics. Walking robots are made mostly to mimic human or animal behaviour. Movement that is done by animals in nature is highly efficient in both speed and energy. Therefore, by replicating this movement, the performance of the robot will become better than if unnatural movement is used. The goal is to develop different control schemes to attain high walking speed, high energy efficiency, and a stable gait. In addition, designing movement over uneven terrain, sloped surfaces and stairs is of great interest.

The robot studied in this thesis is the SemiQuad robot. The SemiQuad seeks to test and develop new control strategies for legged walking robots. It is a five-link biped robot, which is made to model a quadruped gait, called the *curvet* gait. The SemiQuad has point feet, with four DC-motors, each situated on a joint between two links. In this thesis, analysis of some designed movements will be performed, looking at the forces of the stance foot during the gait. Then, a controller will be designed for the walking motion, ensuring orbital stability. The controller is designed with the method of transverse linearization. This method will also be used to analyse the gaits orbital stability properties. To conclude the thesis, a new and much simpler gait will be designed, with a stabilizing controller also designed for the motion.

The methods that were used in the thesis will be presented in chapter 2. This includes discussion on orbital stability analysis, and the method of transverse linearization. The results and simulations will be presented and discussed in chapter 3. Concluding remarks will be presented in chapter 4, with suggested further work in chapter 5.

Chapter 2

Theory

2.1 Introduction

The robot studied in this thesis is the SemiQuad. The SemiQuad is a 5-link robot, four-actuator biped with passive feet that extend into the frontal plane. The SemiQuad is shown in figure 2.1. It is modeled to represent a four-legged robot executing a specific pattern of movement of the limbs, called a gait[2].

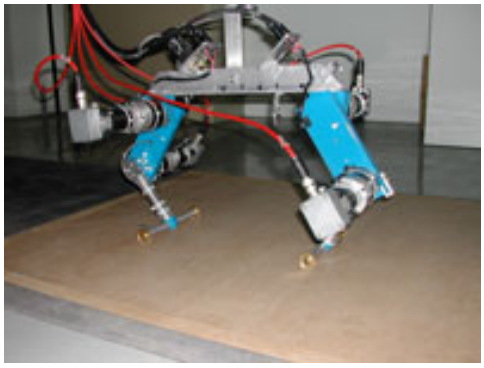


Figure 2.1: *The SemiQuad*

Gait is the pattern of movement of the limbs of animals during locomotion. There are naturally many different types of gaits, for example trot and gallop. The gait that is studied with the SemiQuad is the curvet gait. The curvet gait is when the hind legs both move synchronously, as does the front legs. In the curvet gait, the front legs always stay in front of the hind legs through the whole step sequence[3]. The gait starts with the robot leaning on its back leg, and lifting the front leg. When the front leg is put to the ground, the same movement is done with the back leg lifted in the air. If viewed from the side, a quadruped walking in a curvet gait looks like a biped with one front and one hind leg. Because of the fact that the feet extend in the frontal plane, the robot is stable in that plane. This makes it

possible to design gaits for the robot in the sagittal plane, without the need to take movement in the frontal plane into account. This provides simpler modeling.

The curvet gait is not used by any animal for normal locomotion, as it is not energy efficient compared to other quadruped gaits[3]. Because of the nature of the curvet gait, the design of the gait for a quadruped can be done on a biped, using the theory of virtual legs [4]. Later, this motion can be used on a quadruped. The theory of virtual legs suggests that two legs that are coupled, can be modeled as one leg that is located in the middle of the two real legs. Thus, the SemiQuad is a biped, even though the name suggests otherwise.

The SemiQuad is designed to study the classes of robots that sometimes have less than three points in contact with the ground. These robots are mainly bipeds, tripeds and quadrupeds. These robots are underactuated for some parts of their gait cycle, and will have a free rotation about the contact point with the ground. Underactuation means that they have fewer actuators than degrees of freedom. Hence, they are more challenging to control. This kind of gait, with the free rotation, resembles the kind of gaits that are present in nature. Animals and humans greatly rely on underactuation principles during walking and running[5]. This is mainly because of the greater speed that they can achieve with these kind of gaits, and also because of their energy efficiency[6]. As the gaits are rotations around a contact point, there is no need for actuating every joint to achieve locomotion. This means that less energy is spent during motion[7]. Although the curvet gait is inferior in energy efficiency compared to other gaits, it is a natural gait candidate to study in the underactuation class of robots, because it is simple to model on a biped.

The SemiQuad, due to its gait and build, is made for walking on flat surfaces, as most of the robots that exhibit underactuated gaits. In contrast to humans, these robots do not have actuated ankles, which makes it challenging to traverse uneven ground[6]. These robots are important to study, as their motion on flat surfaces resembles that of humans, which is interesting for development of humanoid robots.

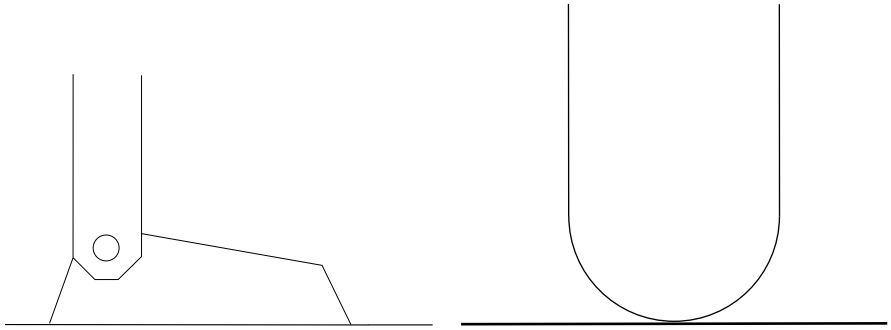
2.2 Limit Cycle Walking

The first walking robots that were made used a control method where the center of mass of the robot never passed outside the support polygon. That is, the center of mass projected on to the walking surface would never lie outside the shape of the feet[6]. This type of constraint strictly limits the motion that the robot performs, which affects the walking speed and disturbance handling negatively[6]. Therefore, a new and less restricting control method was designed, called *Zero Moment Point*, introduced in 1970 by Miomir Vukobratović [8]. This control method is based on calculating the point in space, in which the reaction force of the contact of the foot with the ground does not produce any moment in the horizontal direction. The robot is then controlled such that this point stays within a predefined region. This

is less restrictive than the older method, as it allows the center of mass to travel outside the support polygon, but it does require the stance foot to remain flat against the walking surface at all times during the gait. Thus, the walking motion is less strict, and the robot can therefore walk more efficiently. Although this is an improvement compared to the older method, robots using this control paradigm are still underachieving in terms of speed, energy efficiency and disturbance handling. The appearance of the motion created by these kind of restriction leaves an unnatural looking gait, in comparison to human and animal movement[6]. As a result, new control paradigms are being researched to improve the performance. An example of a robot using ZMP as a control strategy is the ASIMO robot developed by Honda.

If one is to increase the performance of walking robots beyond the ZMP control strategy, it would entail loosening the constraints on the motion even further. Loosening the constraint would have an impact on how the stability measurement is done. Therefore, the concept of limit cycle walking is introduced by Hurmuzlu et al. in 1986[9]. This way of looking at stability takes a periodic motion as a whole, and ensures that the motion is cyclically stable in a periodic orbit. Thus, the system does not need to be stable in every instant of time, as was necessary in the earlier control strategies like the ZMP.

Limit cycle walkers can be divided into two subgroups, one being the passive dynamic walkers, the other being walkers with actuation in their joints. Passive dynamic walkers are robots that do not have any form of actuation, and usually walk down an inclined surface, using gravity as the actuating force. An example of a passive walker has been presented by Goswami et al.[10], where a 2-link robot performing a compass-gait was studied. The other group, which consists of actuated robots, are largely composed of robots with point/arced feet. The robots with point feet do not have any actuation in the ankle joint, which makes them underactuated for parts of, or the whole, gait cycle. These robots rely on the limit cycle principle when walking, which allows them higher speed. In addition, the walking motion is more natural looking[6]. An example of point feet and an actuated ankle is given in figure 2.2.



(a) Robot foot with ankle **(b)** Point foot

Figure 2.2: *Examples of a robot ankle and a point foot*

Another aspect of the limit cycle walkers is that they are very energy efficient compared to other types of robots, such as the robots using Zero Moment Point control[7]. A downside to these point feet robots is that they lose the ability to stand still, as they do not have static stability. This limits the scope of their use, as they can not perform other tasks than to walk or run. Examples of robots with point feet are RABBIT and SemiQuad, both designed by Chevallereau et al.[1, 11, 12, 13], and a biped studied by Grizzle et al.[14]. These robots are classified as hybrid systems since their legs switch roles at some instants of the gait. Such systems have gained much attention the last decade, and a lot of research has therefore been done on application of control strategies for these systems[15].

2.3 Dynamic Modeling

The dynamic model is necessary to simulate the physics in the robot, as well as simulating how the robot will move with actuation [16]. The dynamic equations are also called the equations of motion. These equations are differential equations that explicitly describe the motion of a robot that is subject to holonomic constraints, in addition to the forces exerted on it [16]. The dynamic equations are calculated using the Euler-Lagrange formalism[17]. This method gives the equations of motion in terms of the generalized coordinates. The external forces to the robot are also defined in the generalized coordinates, and are referred to as generalized forces [16]. For a planar revolute robot, these forces will consist of the torques applied at the joints of the robot.

A model for when the robot has one foot in contact with the ground and the other foot in the air, is referred to as the single support model. A single support model can be calculated under the assumption that the robots stance leg is pinned to the ground [18]. That is, the leg cannot slide or lift in any way. Consequently, the SemiQuad resembles a 5-link robotic arm with revolute joints, which are connected to the ground at the point of contact of the stance leg.

To calculate the equations of motion for the robot, the kinetic energy and the potential energy needs to be calculated for every link. This is used to define a lagrangian function, $\mathcal{L}$, which is the difference between kinetic and potential energy, and is defined as follows[17]

$$\mathcal{L}(\dot{q}, q) = K(\dot{q}, q) - P(q) \tag{2.1}$$

where K is the kinetic energy, and P is the potential energy. By using the lagrangian function (2.1), the equations of motion one can be described as shown in theorem 2.1, which is shown by Murray et al.[16]

Theorem 2.1. Euler-Lagrange equations of motion
"The equations of motion for a mechanical system with generalized coordinates $q \in \mathbb{R}^m$ and Lagrangian $\mathcal{L}$ are given by

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \qquad i = 1, \ldots, m \tag{2.2}$$

where τ_i is the external force acting on the i^{th} generalized coordinate"[16] □

This results in a set of m differential equations describing the motion of the robot and its input, where m is equal to the degrees of freedom. The norm in the literature is to order the Euler-Lagrange equations in such a way that they obtain a standard form, as shown in the following equation[17].

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B\tau \tag{2.3}$$

Here, M is the inertia matrix, C is the coriolis matrix, and the matrix G the gravity matrix. The system (2.3) is a set of differential equations defined in the generalized coordinates, which describes the motion of the robot.

2.4 Reduced System Dynamics - Virtual Holonomic Constraints

When creating controllers for underactuated systems, difficulties may arise when it is desired to plan realizable motions. The necessity for a desired behaviour, such as a periodic motion, complicates the controller design. The possibility of controlling every degree of freedom does not exist in underactuated systems, which gives strict restrictions in the design of behaviour of a system[19]. To overcome these challenges, the virtual holonomic constraint approach could be used. The term virtual holonomic constraints refers to constraints imposed on the system, that are not physical constraints. The virtual constraints are algebraic constraints imposed by a controller [20]. In the case of the SemiQuad, the virtual constraints are imposed on the four actuated degrees of freedom. These constraints are constructed to be functions of an independent variable, which in turn can be defined as any physical parameter of the robot [21]. This can for example be the location of the robots

center of mass, or the angle of the ankle in relation to the floor. If this independent variable is chosen as the unactuated state, the virtual holonomic constraints can be inserted into the vector of generalized coordinates[19]. This is shown in the following equation

$$q = \begin{bmatrix} \theta \\ \phi_2(\theta) \\ \phi_3(\theta) \\ \phi_4(\theta) \\ \phi_5(\theta) \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} \dot{\theta} \\ \phi_2'(\theta)\dot{\theta} \\ \phi_3'(\theta)\dot{\theta} \\ \phi_4'(\theta)\dot{\theta} \\ \phi_5'(\theta)\dot{\theta} \end{bmatrix} \qquad \ddot{q} = \begin{bmatrix} \ddot{\theta} \\ \phi_2''(\theta)\dot{\theta}^2 + \phi_2'(\theta)\ddot{\theta} \\ \phi_3''(\theta)\dot{\theta}^2 + \phi_3'(\theta)\ddot{\theta} \\ \phi_4''(\theta)\dot{\theta}^2 + \phi_4'(\theta)\ddot{\theta} \\ \phi_5''(\theta)\dot{\theta}^2 + \phi_5'(\theta)\ddot{\theta} \end{bmatrix} \tag{2.4}$$

where θ is the independent variable, and ϕ_i represents the constraint functions with θ as their parameter. If it is assumed that there exists a controller that holds the constraints in equation (2.4) invariant, the vectors (2.4) can be inserted into the system (2.3) as new state vectors [19]. This means that the system (2.3), with five differential equations and five variables, can be expressed as functions of five equations with *one* variable. The resulting set of equations are known as the reduced dynamics of the system[15], and are shown in the following equations.

$$\alpha_1(\theta)\ddot{\theta} + \beta_1(\theta)\dot{\theta}^2 + \gamma_1(\theta) = 0 \tag{2.5}$$

$$\alpha_i(\theta)\ddot{\theta} + \beta_i(\theta)\dot{\theta}^2 + \gamma_i(\theta) = \tau_i \ , \quad i = 2 \dots 5 \tag{2.6}$$

This reduced system is normally called the α-β-γ-equations. It is shown in equation (2.5) with one unactuated equation, and four actuated equations. The point of interest is the unactuated equation. Its solution describes the evolution of the unactuated state, which is desired stabilized.

A property of the solution to the reduced system (2.5) is given by Shiriaev et al. and is cited in theorem 2.2.[21]

Theorem 2.2.
"Given a system on the form

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \ \gamma(\theta) = 0 \tag{2.7}$$

If the solution $[\theta(t), \dot{\theta}(t)]$ *of* (2.7) *with initial conditions* $\theta(0) = \theta_0$, $\dot{\theta}(0) = \dot{\theta}_0$ *exists and is continuously differentiable, then along this solution the function*

$$I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0) = \dot{\theta}^2 - \psi(\theta_0, \theta)\left[\dot{\theta}_0^2 - \int_{\theta_0}^{\theta} \psi(s, \theta_0)\frac{2\gamma(s)}{\alpha(s)}ds\right] \tag{2.8}$$

with

$$\psi(\theta_0, \theta_1) = \exp\left\{-2\int_{\theta_0}^{\theta_1} \frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\} \tag{2.9}$$

preserves its zero value along the solution."[21]

This can be coupled with the result cited in proposition 2.3, given by Shiriaev et al.[20]. This ensures the existence of a solution.

Proposition 2.3.
"Given the Lagrangian system (2.2)*, with n degrees of freedom and* $n-1$ *actuators, consider the geometrical relations*

$$q_1 = \phi_1(\theta, c), \quad q_2 = \phi_2(\theta, c), \quad \dots \quad , \quad q_n = \phi_n(\theta, c) \tag{2.10}$$

imposed along the generalized coordinates $q_1, \dots, q_n$*, and the new independent variable* θ*. Suppose that there exists a control law* u^* *for* (2.2) *that makes the relations* (2.10) *invariant along the solutions of the closed loop system, then* θ *is a solution of the system*

$$\alpha(\theta, c)\ddot{\theta} + \beta(\theta, c)\dot{\theta}^2 + \gamma(\theta, c) = 0 \tag{2.11}$$

where $\alpha(\theta, c)$*,* $\beta(\theta, c)$ *and* $\gamma(\theta, c)$ *are scalar functions."[20]*

This means that any controller that holds the virtual constraints invariant, makes a solution to the system (2.7). The integral equation (2.8) will then keep its zero value for the whole solution. It is now proven that the system (2.7) with a controller based on the virtual holonomic constraints give a solution to the system.

To make the walking motion a limit cycle, it is necessary to design the virtual constraints in such a way that they result in a periodic motion for the unactuated degree of freedom. A good alternative to designing the constraints is to use bézier curves as the parametrization. With bézier curves, one can design starting and ending configurations for each phase. This ensures that the robot movement can be predictable. The curves can have further intermediate points defined. These points define the movement of the curve between the initial point and the end point[22]. This is useful, since it is possible to determine the velocity of each degree of freedom at both the start and end of a phase by adjusting the intermediate points, which is a valuable property when periodic motion is desired.

2.5 Hybrid Systems

A model of a robot walking in the sagittal plane, with the two feet alternating as swing legs, is a hybrid model [18]. This implies that the solution of the system (2.7) for the SemiQuad, will exhibit a discontinuous jump in the phase portrait for each time the swing foot impacts the ground. When the legs change roles, the model is switched and the generalized coordinates are relabeled. This is what causes the jump in the solution. These switching points of the solution lie on a surface in the phase plane, called switching surface, which appear as C^0 curves in the phase plane [15]. When the solution hits a switching surface, the coordinates will change according to the discontinuous update law defined for the switching surface, and

the next phase initiated with the new coordinates [18].

$$\Gamma_s = \Gamma_- \bigcup \Gamma_+ \tag{2.12}$$

The update law found between the different phases, is defined to act on a switching surface Γ_s. The switching surface consists of two components as given in equation (2.12) [15]. The relation between the two are defined by the update law F. This means that when the solution reaches some point $p_- \in \Gamma_-$, the solution exhibits an instantaneous jump to a new point $p_+ \in \Gamma_+$ of the phase plane, which serves as the reinitializing conditions of the solution in the new phase [15]. Figure 2.3 shows the concept of a hybrid system. It shows a system with 4 phases, and 4 jumps.

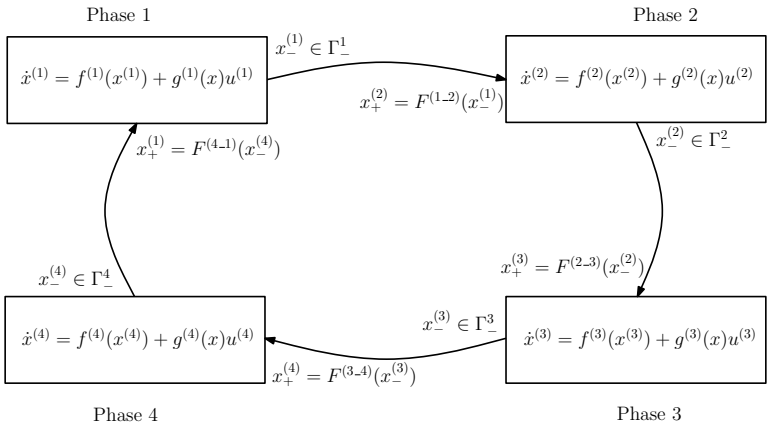


Figure 2.3: *Concept of hybrid switching with 4 phases and 4 jumps*

From figure 2.3 it can be seen that when the solution hits a switching surface, the update of the coordinates are calculated, and the next phase is initiated. Some of the switching surfaces lead to trivial transformations, and others to nontrivial ones. An example of a nontrivial transformation is when the swing foot hits the ground. The gait of a biped walker can have two instances of this, which will lead to a relabeling of the coordinates. The phase plot of the evolution of the gait will therefore not be continuous. Shiriaev et al. have proved the existence of a periodic solution, which is cited in lemma 2.4.[15]

Lemma 2.4.
"Suppose the hybrid system

$$\begin{aligned} \alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \ \gamma(\theta) = 0 \\ [q(t_+), \dot{q}(t_+)] = F(q(t_-), \dot{q}(t_-), t_-) \end{aligned} \tag{2.13}$$

has a periodic solution

$$q(t) = q(t+T) \tag{2.14}$$

of the period T with one non-trivial continuous-time sub-arc and one non-trivial update jump, then it is by necessity a stationary point of the next equations

$$I(q_-, \dot{q}_-, q_+, \dot{q}_+) = 0 \tag{2.15}$$

$$[q_+, \dot{q}_+] = F(q_-, \dot{q}_-) \tag{2.16}$$

where the function I is defined in equation (2.8)*, and $(q_-, \dot{q}_-)$ is a uniquely defined point of the solution that belongs to Γ_-." [15]* □

Lemma 2.4 suggests that the integral to the solution will keep the same value as before the impact, which was 0, and still be a solution to the system (2.5). This is a result that proves the existence of a periodic solution, even though the system is hybrid with more than one discontinuous jumps in the phase plane.

The desired result is a closed hybrid solution in the phase plane, whose existence is proven in lemma 2.4. This means that one can find a solution using the reduced dynamics of the system, while also defining the switching surfaces and the discontinuous update law. When this is done, if the endpoint of the solution coincides with the initial point in the phase plot, a periodic gait is achieved.

2.6 Orbital Stabilization Analysis

The concept of stability for a biped robot with underactuation is different from for example an industrial process. What is stable performance for an industrial plant does not mean stability for a biped robot. That is, converging a controlled state to a given reference value. If a biped is in a phase of its gait where it is standing on one foot, the single support phase, having the angle of the joints close to a reference value does not mean that the robot is stable. The robot is standing on one foot while controllably *falling* towards the floor. Falling is not stable, which is why the concept of orbital stability must be discussed.

Biped walking consists of repeated steps, which optimally will be of periodic nature. Certain constraints can be put on the motion, creating a trajectory which defines an optimal motion for all links. Since the system is of periodic behaviour, it is necessarily time-dependent. Analyzing a time-dependent system for a single instant of time does not provide any meaningful results, because the system is continually changing. However, analyzing the behaviour along an orbitally stable trajectory

will give an indication of stability for the systems motion. Stability in the orbital sense entails convergence of the system towards a reference trajectory. As the system is not necessarily stable at each time instant, the orbital stability ensures that the motion performed by the robot converges towards a periodic orbit, or a limit cycle[23]. This limit cycle was proven to exist in section 2.4. The overall objective when doing the orbital stability analysis, is to design an exponential orbital stabilizing controller which will bring the uncontrolled state to the wanted periodic orbit.

When analyzing orbital stabilization of the class of mechanical systems that have one degree of underactuation, there exists a series of methods. Two methods of analysis that are mentioned in the literature is that of Poincaré first-return map and the method of transverse linearization.

2.6.1 Poincaré First-Return Map

The Poincaré analysis is performed by constructing a surface orthogonal to the flow of a periodic orbit[24]. This surface is called a Poincaré section, and is shown in figure 2.4.

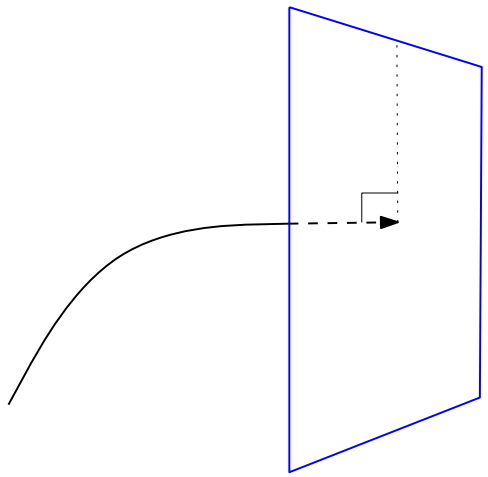
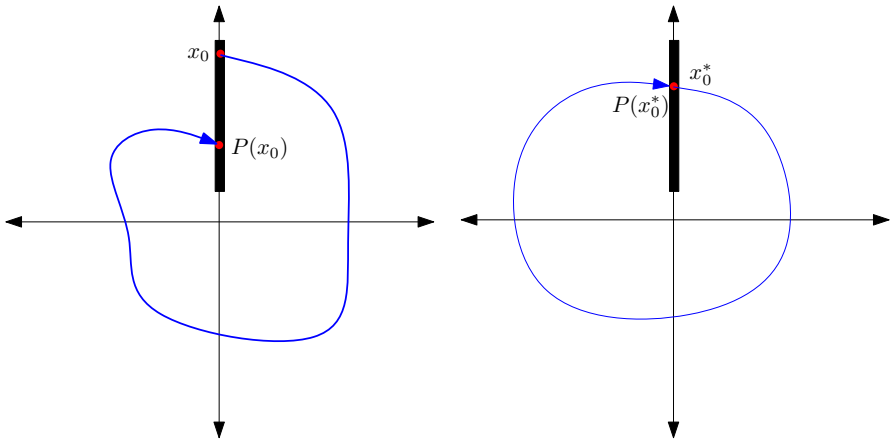


Figure 2.4: *Example of a poincaré section orthogonal to the orbit*

The Poincaré first-return map method starts by selecting a constant Poincaré section at a point along the periodic orbit. This section will consist of a plane, from which a number of points can be used as initial points for simulation of the orbit. The system is simulated with these initial conditions, and the position where the solution reenters the poincaré section is calculated, with the intent to create a relation between the initial point and boundary point. This can be seen in figure 2.5.



(a) Showing an orbit and the relation $x_0 = P(x_0)$

(b) Showing a closed orbit, meaning $x_0 = P(x_0)$

Figure 2.5: *Visualisation of a Poincaré map. The Poincaré section represented by thick a black line. The trajectory is represented by a blue line, going from the initial point x_0, to the point $P(x_0)$*

From figure 2.5, it can be seen that there exists a relation between the initial point, and the reentry point of the poincaré section. This relation is defined as $x[k+1] = P(x[k]) = \Phi(t_k, x[k])$, and is called a Poincaré First Return Map[18]. Analysis of this function can reveal orbital stability characteristics of the system. The analysis of this map can be done using a Lamerey diagram[25] as shown in figure 2.6. By using a Lamerey diagram, information of the stability of the orbit can be extracted. The method takes the first return function $\Phi(t_k, x[k])$, and differentiates it. The resulting function $\Phi'(x[k])$ is then evaluated. The orbit is said to be stable if $\Phi'(x[k]) < 1$, and unstable if $\Phi'(x[k]) > 1$[25]. In essence, this is a way to describe whether the reentry point moves away from, or closer to, the initial point if the initial point is moved on the poincaré section. If $\Phi'(x[k]) = 1$ for a given $x[k]$, it means that $x[k+1] = P(x[k])$. This means that the orbit is a closed orbit, and per definition, is a limit cycle.

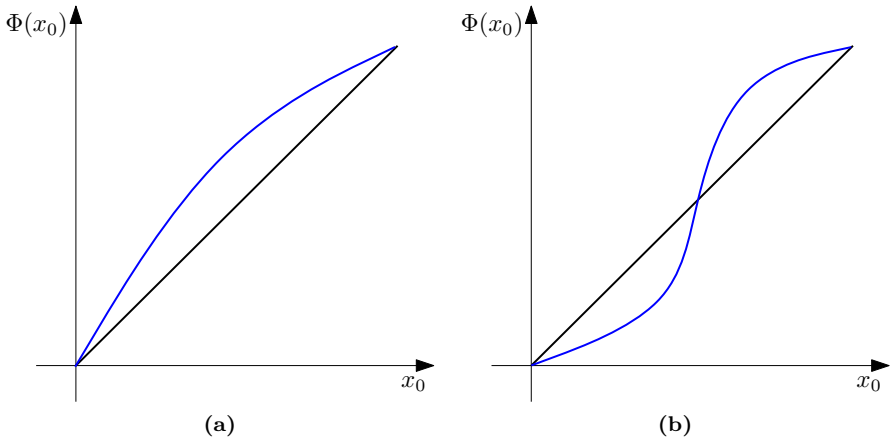


Figure 2.6: *Examples of Lamerey Diagrams. The function $\Phi(x_0)$ is represented by a blue line and is shown in relation to a line with derivative equal to one.*

The Poincaré method consists of simulating the system for a wealth of different initial conditions, followed by analyzing the results to find orbital stability. Consequently, this method requires a large computational capacity, as it requires numerical solutions for a large set of initial conditions[18]. This computational complexity is undesirable, which has led to investigation of alternative methods. An example of use of the Poincaré first return method is the previously mentioned five-link, four-four actuator biped RABBIT with success[11]. RABBIT was designed to be a running biped, which includes flight phases in the gait, and achieved running with a number of different velocities during the tests of the system. The concept of using the Poincaré method on biped robots, specifically systems with impulse effects, was studied in depth by Grizzle et al.[14].

2.6.2 Transverse Linearization

The method of transverse linearization is a different approach than the Poincaré first return method. The transverse linearization method seeks to rewrite the coordinates of the system into a new set of coordinates relating the path of the solution to the path of the optimal solution. These coordinates can be seen as error coordinates. These coordinates lie in a plane transversal to the flow of the solution, namely on a moving poincaré section just as the one for the Poincaré first-return method. The system is then linearized along the orbit in terms of these new coordinates, meaning that a periodic, time-varying linearization transverse to the solution is produced, hence its name[24]. This linearization will give linearized snapshots of the nonlinear system along the optimal trajectory. The reason for introducing the auxiliary linear system is that a stabilizing controller can be made for the linear system, which can be applied to the nonlinear system to obtain orbital stabilization.

A biped gait involves impacts with the floor, and consequently changing the set of virtual constraints along the trajectory. It is therefore a hybrid system, as discussed in section 2.5. The transverse linearization of a hybrid system with underactuation consists of three steps[26].

1. Linearizing the update law $\Gamma_-, \Gamma_+, F(\,\cdot\,)$ around the point where the trajectory intersects the switching surface, given as $[q_*(T_{h-}); \dot{q}_*(T_{h-})] \in \Gamma_-$.

2. Linearizing the transverse dynamics of the system (2.3) along the continuous-in-time- sub-arc of q_*. In other words, along the optimal trajectory.

3. Merging the linearizations of the continuous and discrete parts of the dynamics.

Step 1: Linearize the update law

The first step is to linearize the switching surfaces that exist in the orbit. The switching surfaces exist in $2n$-dimensional space. The linearization of switching surface will be a tangent plane to this surface, with the tangential point being at the intersecting point between the optimal trajectory and the switching surface[26]. This can be found by calculating the jacobian of the switching surface at the given point. An example of the linearization of a switching surface is shown in figure 2.7.

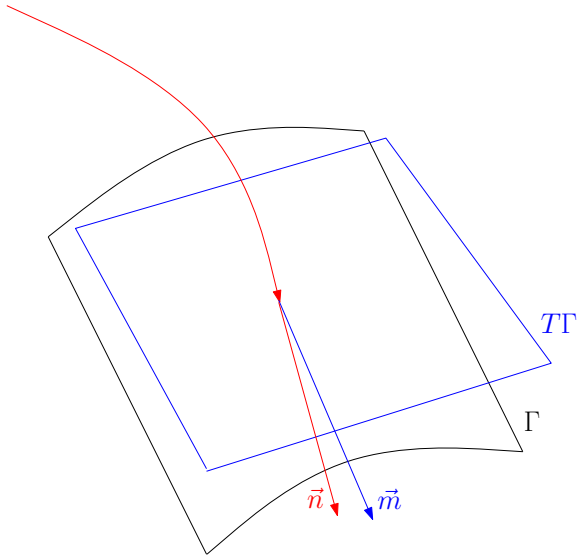


Figure 2.7: *Switching surface* Γ *in black, and its tangent plane* $T\Gamma$ *in blue*

This operation must be done for each of the switching surfaces that may exist on the trajectory.

Step 2: Linearize the continuous-in-time dynamics

To facilitate the linearization of the dynamics, one can use a feedback transformation described in Spong[27]. Given virtual constraints $\phi_i(\theta)$, one can introduce the coordinates[20]

$$Y = q_1 - \phi_1(\theta), \cdots, y_n = q_n - \phi_n(\theta) \tag{2.17}$$

which together with θ give $n+1$ coordinates for the n degrees of freedom system. This is excessive, but it can be seen that one of the coordinates can be described as a function of the others. It is assumed that this can be done with coordinate y_n, which leaves

$$y_i = (y_1, \cdots, y_{n-1}) \quad and \quad \theta \tag{2.18}$$

which can be used as the new coordinates[26]. The last relation, for y_n, is

$$q_n = \phi_n(\theta) + h(y_1, \cdots, y_{n-1}, \theta) \tag{2.19}$$

If a matrix L is introduced as

$$L(\theta, y) = \begin{bmatrix} I_{n-1}, 0_{(n-1)\times 1} \\ \nabla h \end{bmatrix} + \left[0_{n\times(n-1)}, \dot{\Phi}_{(n-1)\times 1}\right] \tag{2.20}$$

where $\dot{\Phi} = \left[\dot{\phi}_1(\theta), \ldots, \dot{\phi}_n(\theta)\right]$, the following definitions can be introduced [28].

$$N(\theta, y) = \left[\mathbf{I}_{(n-1)}, \mathbf{0}_{(n-1)\times 1}\right] L^{-1}(\theta, y) \left[M^{-1}(q)B(q)\right] \tag{2.21}$$

$$\begin{aligned} R(\theta, y) = &\left[\mathbf{I}_{(n-1)}, \mathbf{0}_{(n-1)\times 1}\right] L^{-1}(\theta, y) \\ &\times \left(\left[M^{-1}(q)(-C(q,\dot{q})\dot{q} - G(q))\right] - \dot{L}(\theta, y) \begin{bmatrix} \dot{y} \\ \dot{\theta} \end{bmatrix} \right) \end{aligned} \tag{2.22}$$

These equations make it possible to define the relation [28]

$$u = N(q)^{-1}[v - R(q, \dot{q})] \tag{2.23}$$

where v represents the actuation of the transversal dynamics, and u the actuation of the original system. Thus, it is possible to make a controller for stabilizing the transverse coordinates, v, and transform it into actuation of the original nonlinear system, u.[26]

With a feedback transformation computed, the reduced system dynamics described in equation (2.5) can be rewritten as a linear function of the new transverse coordinates. In the vicinity of the trajectory, the $\alpha - \beta - \gamma$ equations will no longer equal zero. The value is now expressed as a function of θ, $\dot{\theta}$ and $\ddot{\theta}$ as follows.[28]

$$\ddot{y} = v \tag{2.24}$$

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \ \gamma(\theta) = g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y}) \tag{2.25}$$

The right hand side of the system (2.25) can be found by substituting the relations shown in equations (2.26)-(2.28) into the dynamic equations (2.3).

$$q_i = y_i + \phi_i(\theta) \tag{2.26}$$

$$\dot{q}_i = \dot{y}_i + \dot{\phi}_i(\theta)\dot{\theta} \tag{2.27}$$

$$\ddot{q}_i = \ddot{y}_i + \ddot{\phi}_i(\theta)\dot{\theta}^2 + \dot{\phi}_i(\theta)\ddot{\theta} \tag{2.28}$$

It can be seen that when $y_i = 0$, the system (2.25) is equal to the $\alpha - \beta - \gamma$ equations, which are equal to zero when the system is on the optimal trajectory. It can be shown that the right hand side in equation (2.25) can be written as[28]

$$g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y}) = g_y(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y})y + g_{\dot{y}}(\theta, \dot{\theta}, y, \dot{y})\dot{y} + g_v(\theta, \dot{\theta}, y, \dot{y})v \tag{2.29}$$

From $g(\,\cdot\,)$, the functions g_y, $g_{\dot{y}}$, and g_v are defined as follows[28]

$$g_y = \left.\frac{g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y})}{y}\right|_{y=\dot{y}=\ddot{y}=0} = \left.\frac{\partial g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y})}{\partial y}\right|_{y=\dot{y}=\ddot{y}=0} \tag{2.30}$$

$$g_{\dot{y}} = \left.\frac{g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y})}{\dot{y}}\right|_{y=\dot{y}=\ddot{y}=0} = \left.\frac{\partial g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y})}{\partial \dot{y}}\right|_{y=\dot{y}=\ddot{y}=0} \tag{2.31}$$

$$g_v = \left.\frac{g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y})}{\ddot{y}}\right|_{y=\dot{y}=\ddot{y}=0} = \left.\frac{\partial g(\theta_*, \dot{\theta}_*, \ddot{\theta}_*, y, \dot{y})}{\partial \ddot{y}}\right|_{y=\dot{y}=\ddot{y}=0} \tag{2.32}$$

The relations are derived from using L'Hôpital's rule on the indeterminate ${}^0/_0$ situation that arises when $y = \dot{y} = \ddot{y} = 0$.

These linearized functions are valid in the vicinity of the optimal trajectory. These functions can be used in forming a linear system whose behaviour should resemble that of the nonlinear system when it is close to the optimal trajectory[24]. Since the transverse system is defined in a moving plane transverse to the trajectory, the coordinates will have one dimension less than the original system. The transverse plane along the solution is shown in blue in figure 2.8.

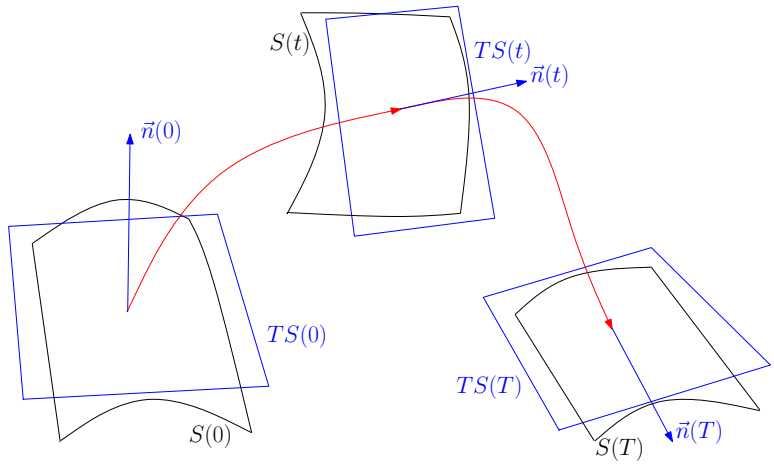


Figure 2.8: *Tangent planes to the solution along the trajectory shown as blue planes at different time instants. The tangent planes normal vector $\vec{n}$ is also shown in blue.*

The $(2n-1)$-dimensional coordinate vector is described as the following equations.[26]

$$X_\perp = \begin{bmatrix} I \\ Y \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} I(\theta, \dot{\theta}, \theta_*(0), \dot{\theta}_*(0)) \\ q_i - \phi_i(\theta) \\ \dot{q}_i - \phi_i'(\theta)\dot{\theta} \end{bmatrix} \tag{2.33}$$

The integral function I is a measure of how far from the optimal trajectory the uncontrolled variable is at the given set $[\theta, \dot{\theta}]$. It is described by Shiriaev et al. [21] as

$$I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0) = \dot{\theta}^2 - \psi(\theta_0, \theta)\left[\dot{\theta}_0^2 - \int_{\theta_0}^{\theta} \psi(s, \theta_0)\frac{2\gamma(s)}{\alpha(s)}ds\right] \tag{2.34}$$

where

$$\psi(\theta_0, \theta_1) = \exp\left\{-2\int_{\theta_0}^{\theta_1} \frac{\beta(\tau)}{\alpha(\tau)}d\tau\right\} \tag{2.35}$$

With the g-functions as well as the coordinate vector defined, it is possible to define the dynamics of the transverse system. The linear system can be written as [24]

$$\dot{X}(t) = A(t)X(t) + B(t)V(t) \tag{2.36}$$

These matrices are periodic in time, meaning that they can described as in the following equation

$$A(t) = A(t+T) \qquad B(t) = B(t+T) \tag{2.37}$$

Here, T is the period of the motion. The system matrices $A(t)$ and $B(t)$ of the linearized system are defined as follows[24].

$$\begin{aligned} B(t) &= \begin{bmatrix} b_1(t) \\ \mathbf{0}_{(n-1)\times(n-1)} \\ \mathbf{I}_{(n-1)\times(n-1)} \end{bmatrix} \\ A(t) &= \begin{bmatrix} a_{11}(t) & a_{12}(t) & a_{13}(t) \\ \mathbf{0}_{(n-1)\times 1} & \mathbf{0}_{(n-1)\times(n-1)} & \mathbf{I}_{(n-1)\times(n-1)} \\ \mathbf{0}_{(n-1)\times 1} & \mathbf{0}_{(n-1)\times(n-1)} & \mathbf{0}_{(n-1)\times(n-1)} \end{bmatrix} \end{aligned} \tag{2.38}$$

Here, $\mathbf{0}$ is a zero-filled matrix and $\mathbf{I}$ is an identity matrix, both of given dimensions. The coefficients of the system matrices are given in the following equations[24]

$$\begin{aligned} b_1(t) &= \dot{\theta}_\star(t)\frac{2g_v(\theta_\star(t),\dot{\theta}_\star(t),0,0)}{\alpha(\theta_\star(t))} \\ a_{11}(t) &= -\dot{\theta}_\star(t)\frac{2\beta(\theta_\star(t))}{\alpha(\theta_\star(t))} \\ a_{12}(t) &= \dot{\theta}_\star(t)\frac{2g_y(\theta_\star(t),\dot{\theta}_\star(t),\ddot{\theta}_\star(t),0,0)}{\alpha(\theta_\star(t))} \\ a_{13}(t) &= \dot{\theta}_\star(t)\frac{2g_{\dot{y}}(\theta_\star(t),\dot{\theta}_\star(t),\ddot{\theta}_\star(t),0,0)}{\alpha(\theta_\star(t))} \end{aligned} \tag{2.39}$$

Here, the functions g are as given in equations (2.30)-(2.32). This periodic time-varying system describes the linearized dynamics of the nonlinear system in the vicinity of the optimal trajectory.

Step 3: Merge the discrete and continuous-in-time parts together

When solving the linearized system, it is clear that when a switching surface is reached, the solution will lie in a plane transverse to the trajectory. It should be noted that this plane not necessarily is equivalent to the tangent plane of the switching surface. This is because the tangent plane $T\Gamma \neq TS(t_-)$. As a result, the linearized system will give a solution on $TS(t)$, which does not equate a solution in the nonlinear system, or $T\Gamma$. This can be rectified by projecting the solution from $TS(t_-)$ onto $T\Gamma$ in a direction parallel to the normal vector $\vec{n}^{(-)}$. This will convert the solution found from the linearized system to a feasible solution of the nonlinear system. As the system changes phases, the virtual constraints will undergo a change, which could make the trajectory exit the surface in a different direction than before the switch. This implies that the tangent plane of the initial conditions for the next phase is not the same as the tangent plane of the switching surface. Again, the tangent plane $T\Gamma$ is not the same as the tangent plane of the linearized solution, $TS(t_+)$. This means that another projection must be made, this time from $T\Gamma$ onto $TS(t_+)$, parallel to the normal vector $\vec{n}^{(+)}$. As a result, the linearized system can be initialized with feasible initial conditions for the next phase. The projections are shown in figure 2.9.

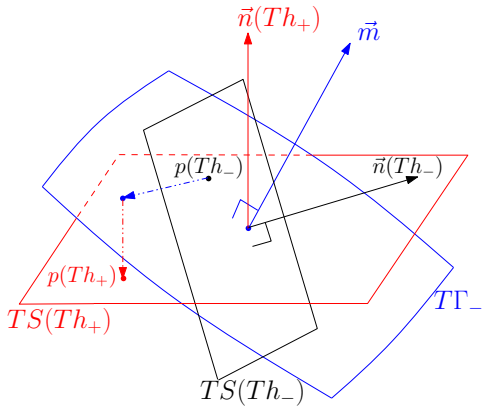


Figure 2.9: *Merging linearized continuous and discrete parts of the linearization. Projections of points $p(Th-)$ in the black plane to $p(Th+)$ in the red plane. The plane $T\Gamma$ is shown in blue along with its normal vector $\vec{m}$.*

In figure 2.9, the transverse system has given a solution $p(Th-)$, which is shown in black. To initialize the next phase, the points $p(Th-)$ need to be transformed into $p(Th+)$, which lie on the tangent plane to the trajectory at the start of the next phase, and is showed in red.

A challenge in relation to these projections is that they are computed in the original coordinates $[q_i, \dot{q}_i]$, while the linear system is calculated in the transverse coordinates. The change from original coordinates into the transverse coordinates is well defined, but the change from the transverse coordinates to the original coordinates is not. The change from transverse coordinates involve changing from a system of $(2n-1)$ coordinates into a system of $2n$ coordinates. This is not trivial, but it can be done. The calculations are presented by Freidovich et al.[29], and start by defining the transformation from the original coordinates to the transverse coordinates.

$$\begin{bmatrix} \Delta I \\ \Delta y \\ \Delta \dot{y} \end{bmatrix} = L(t) \begin{bmatrix} \Delta q \\ \Delta \dot{q} \end{bmatrix} \tag{2.40}$$

The matrix $L(t)$ is given as

$$L(t) = \begin{bmatrix} -2\ddot{\theta} & \mathbf{0}_{1\times(n-1)} & \cdots & 2\dot{\theta} & 0 & 0 \\ -\dot{\phi}_1(\theta_*(t)) & \mathbf{I}_{(n-1)\times(n-1)} & \cdots & \mathbf{0}_{(n-1)\times(n-1)} & \cdots & \cdots \\ \vdots & \vdots & \ddots & \vdots & \ddots & \cdots \\ -\dot{\phi}_{n-1}(\theta_*(t)) & \vdots & \vdots & \vdots & \vdots & \ddots \\ -\ddot{\phi}_1(\theta_*(t))\dot{\theta}_*(t) & \mathbf{0}_{(n-1)\times(n-1)} & \cdots & -\dot{\phi}_1(\theta_*(t)) & \mathbf{I}_{(n-1)\times(n-1)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \cdots \\ -\ddot{\phi}_{n-1}(\theta_*(t))\dot{\theta}_*(t) & \vdots & \ddots & -\dot{\phi}_{n-1}(\theta_*(t)) & \vdots & \ddots \end{bmatrix} \tag{2.41}$$

The vector $\vec{n}$ is the normal vector to the transversal plane of the solution, and is defined as

$$\vec{n}(t) = \begin{bmatrix} \dot{q}(t) \\ \ddot{q}(t) \end{bmatrix} \tag{2.42}$$

The change from transverse coordinates into the original coordinates can then be expressed as

$$\begin{bmatrix} \Delta q \\ \Delta\dot{q} \end{bmatrix} = \begin{bmatrix} L(t) \\ n^T(t) \end{bmatrix}^{-1} \begin{bmatrix} \Delta I \\ \Delta y \\ \Delta\dot{y} \\ 0 \end{bmatrix} \tag{2.43}$$

These equations, (2.41)-(2.43), will yield the solution in original coordinates, which can be projected as shown in figure 2.9. The total change in coordinates and projections can be presented in a matrix η, a map between the transverse coordinates before the switch, to the transverse coordinates after the switch. [26]

$$\eta = \left[P^+_{\vec{n}(0)}\right] dF \left[P^-_{\vec{n}(Th)}\right] \tag{2.44}$$

where

$$P^+_{\vec{n}(0)} = L(0)\left(I_n - \frac{n(0)n^T(0)}{n^T(0)n(0)}\right) \tag{2.45}$$

$$P^-_{\vec{n}(Th)} = \left(I_n - \frac{n(0)n^T(0)}{n^T(0)n(0)}\right) \begin{bmatrix} L(t) \\ n^T(t) \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{I}_{n-1} \\ \mathbf{0}_{1\times(n-1)} \end{bmatrix} \tag{2.46}$$

The matrix dF is the linearization of the relabeling matrix, which relabels the coordinates between two phase switches. The whole process of tying together the discrete and continuous dynamics of the system is shown in figure 2.10.

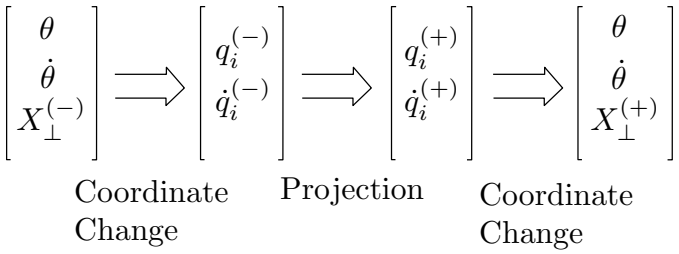


Figure 2.10: *Operations for transforming the coordinates in the switch*

All three operations are compacted into the mapping matrix η. Once the mapping η is found, step number three is complete. This step now handles the phase switches during the simulation of the linearized system.[26]

The linearization has now produced a periodic and linear system, which will be used to make a state feedback controller, where the controller naturally also is periodic. The stabilization of this linearized system will be equal to the exponential orbital stabilization of the nonlinear system to the previously found optimal periodic orbit, as proved by Leonov[30]. This method has been used with much success, with examples given in Shiriaev et al. [20], where a cart-pendulum system was studied. A devil-stick system was also studied using this method by Shiriaev et al. [26].

One can see a big difference in the two methods presented. A big difference is that the transverse linearization introduces a new auxiliary system. If the auxiliary system is stabilized, the stabilizing controller will also stabilize the original system. The Poincaré first return method on the other hand, does not use an auxiliary system. Another point is that for the Poincaré first return method, a controller has to be designed in order to be able to study orbital stability. For the transverse linearization an analysis of the system on its optimal orbit can be done even before the controller is designed. This makes it possible to make several different motions, as it only entails a change in the virtual constraints. This would not be possible for the Poincaré first return method without extensive simulation.

2.7 Controller Design

Orbital exponential stability means that the deviation from the optimal trajectory will decrease throughout the motion. If a system initializes away from the optimal initial conditions, the solution will lie in the vicinity of the optimal trajectory, but not on it. If a series of different initial conditions are simulated, they can form a cylindrical shape around the optimal trajectory. In the transversal plane, they would form a circular or elliptical shape. Orbital stability would mean that this cylinder will be constricting along the trajectory, moving the initially non-optimal solution towards the optimal solution. This constricting cone is showed in figure

2.11, where the distance to the optimal is much smaller at the end than it was for the initial conditions.

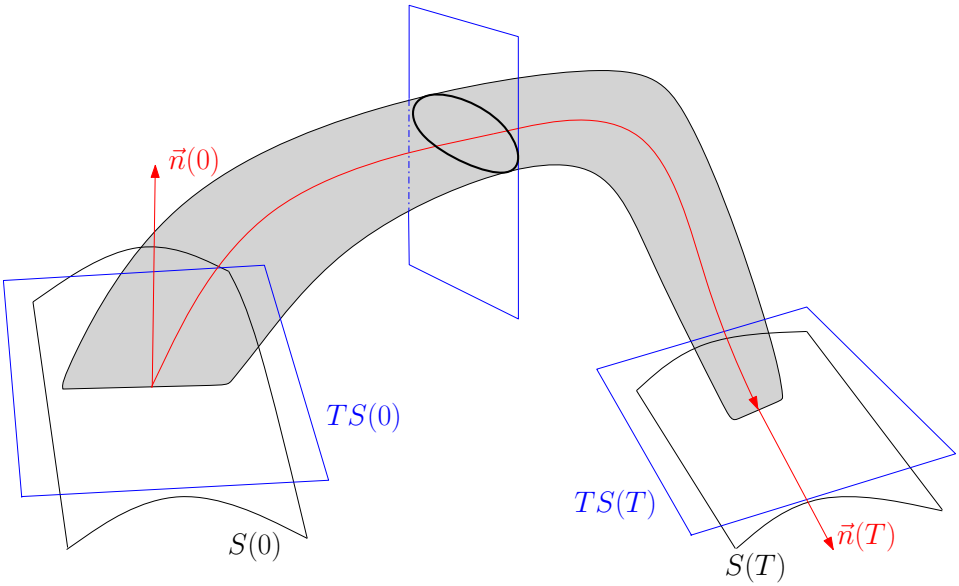


Figure 2.11: *Gradually constricting cone along the trajectory. Elliptical shape on a tangent plane is shown in black.*

For an orbitally stabilizing controller this cone will finally converge to the optimal trajectory, controlling the system along the optimal solution.

As mentioned in section 2.6, a controller that stabilizes the auxiliary linear system around the origin, will also be a orbitally stable controller for the original nonlinear system. The relation between the linear system and the nonlinear system is proven by Shiriaev et al.[28]

Theorem 2.5.
"Given a T-periodic solution $q = q_*(t)$ to the system (2.3), with a C^1-smooth control signal $u = u_*(t)$ and C^2-smooth virtual constraints ϕ_i. Consider the system (2.3) rewritten in the vicinity of the motion as in the system (2.25) with the function (2.29), and the linear T-periodic system (2.36). The following two statements are equivalent.

1. *There is a C^1-smooth T-periodic matrix gain $K(\tau)$ such that the feedback control law*

$$V(t) = K(\tau)\left[I; Y; \dot{Y}\right], \qquad K(\tau) = K(\tau + T) \tag{2.47}$$

stabilizes the origin of the linear system (2.36)

2. *There exists a C^1-smooth time-invariant feedback control law of the form $u = v + U(\theta, \dot{\theta}, y, \dot{y})$ with*

$$v = f(\theta, \dot{\theta}, y, \dot{y}) \tag{2.48}$$

that makes the T-periodic solution $q_*(t) = q_*(t + T)$ of the system (2.3) exponentially orbitally stable." [28]

The two statements in theorem 2.5 state the connection between the linear auxiliary system and the original nonlinear system. With this connection well established, a controller for the linearized system needs to be designed. A well demonstrated method to control linear systems is the linear quadratic controller.

In order to stabilize the linear system (2.36), a controller $V(t)$ needs to be defined. This controller can be formulated as

$$V(t) = -K(t)x_\perp \tag{2.49}$$

When inserted into the system (2.36), this will yield the system

$$\dot{x}_\perp = (A(t) - B(t)K(t))x_\perp \tag{2.50}$$

The matrix gain K is derived from the solution of the Ricatti equation to find an optimal controller. The resulting controller optimizes different performance criteria. As the matrices $A(t)$ and $B(t)$ are periodic, $K(t)$ will also be periodic by necessity. One can define matrices Q, $G(\tau)$, $g(\tau)$ and $\Gamma(\tau)$, and the condition $\Gamma(\,\cdot\,) > 0$ for $\tau \in [0, T]$, such that there exists a solution to the Ricatti equation defined as

$$\begin{aligned}\frac{d}{dt}P(\tau) + A(\tau)^T P(\tau) + P(\tau)A(\tau) + G(\tau) = \\ = \left[P(\tau)B(\tau) + g(\tau)\right]\Gamma(\tau)^{-1}\left[P(\tau)B(\tau) + g(\tau)\right]^T\end{aligned} \tag{2.51}$$

with the boundary condition $P(T) = Q$. If this solution is well defined over the time interval $[0, T]$, the LQR-controller is defined by the following equation[31]

$$V = K(\tau)X = -\Gamma(\tau)^{-1}\left[B(\tau)^T P(\tau) + g(\tau)^T\right]X \tag{2.52}$$

This controller will minimize the performance index, which is given as follows

$$\int_0^T \begin{bmatrix} X(\tau) \\ V(\tau) \end{bmatrix}^T \begin{bmatrix} G(\tau) & g(\tau) \\ g(\tau) & \Gamma(\tau) \end{bmatrix} \begin{bmatrix} X(\tau) \\ V(\tau) \end{bmatrix} d\tau + X(T)^T Q X(T) \tag{2.53}$$

along the solution trajectory of the system (2.36)[31]. The controller can be tuned by varying the performance index (2.53), which is done by changing the matrices Q, $G(\tau)$, $g(\tau)$ and $\Gamma(\tau)$. The matrices penalizes the controller for different response criteria. For the version given in equation (2.53), the matrix Q penalizes the systems ability to reach the desired value at time T. The matrices G and Γ

penalize deviations from the desired value and large actuating signals, respectively. Weighting the error against actuation gives the designer a choice of either quickly removing the error, or penalizing large actuation signals. This controller gives the designer a lot of room for tuning for the desired behaviour. Once a controller $V(t)$ has been found, it can be converted into actuation for the nonlinear system, $u(t)$, by using the feedback transform relation (2.23). This should yield an orbitally stabilizing controller for the nonlinear system.

Chapter 3

Results and Discussion

The first part of this chapter will include a presentation of results gained from the preliminary project, of which this thesis is based upon. This includes the kinematic and dynamic model, as well as a description of the gaits that were created. After the preliminaries, an analysis of these gaits will be presented. This is followed by a discussion of the results of the controller design, covering the results of the transverse linearization. A new gait will be presented in the final part of the chapter. All of the dynamic models and the transverse linearization were created by using the symbolic math software Maple. All simulations were conducted in Matlab/Simulink. The created code will be submitted alongside the thesis.

3.1 Preliminary Results

3.1.1 Kinematic and Dynamic Model

For the gaits that were designed, there were required two kinematic models for the single support phase of the motion. This was because the stance leg would change halfway through the gait. These choices of coordinates for the kinematic model of the first half step is shown in figure 3.1. It can be seen from the figure that angles q_2 to q_5 will have negative values.

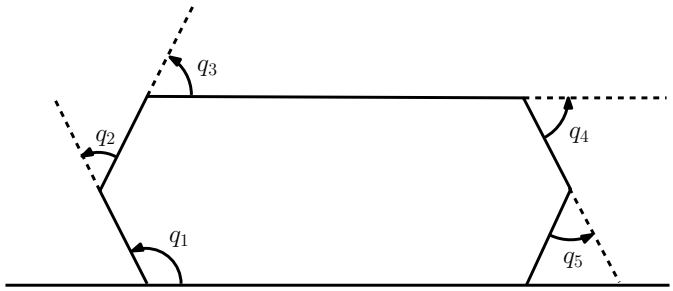


Figure 3.1: *Choice of coordinates for front leg(right) as swing leg*

The coordinates change for the second half step. The coordinates of the second half are shown in figure 3.2. As can be seen in the figure, all of the angles have positive values.

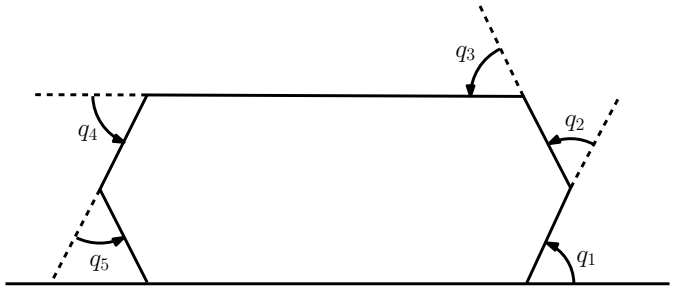


Figure 3.2: *Choice of coordinates for back leg(left) as swing leg*

To make the kinematic model, the physical properties of the robot are needed. These properties are shown in table 3.1, and are given by the makers of the robot.[2]

Table 3.1: *Physical properties of the SemiQuad*

Link number	Mass(kg)	Length(m)	Mass Center(m)	Inertia(kg m^2)
Link 1	$m_1 = 0.404$	$l_1 = 0.15$	$s_1 = 0.083$	$I_1 = 0.0589$
Link 2	$m_2 = 3.211$	$l_2 = 0.15$	$s_2 = 0.139$	$I_2 = 0.0837$
Link 3	$m_3 = 6.618$	$l_3 = 0.45$	$s_3 = 0.225$	$I_3 = 0.3157$
Link 4	$m_4 = 3.211$	$l_4 = 0.15$	$s_4 = 0.139$	$I_4 = 0.0837$
Link 5	$m_5 = 0.404$	$l_5 = 0.15$	$s_5 = 0.083$	$I_5 = 0.0589$

The forward kinematic equations for the x and y coordinates of the points were calculated to be as shown in the following equations:

$$P_{i_x} = \sum_{j=1}^{i} L_j \cos\left(\sum_{k=1}^{i} \theta_k\right) \tag{3.1}$$

$$P_{i_y} = \sum_{j=1}^{i} L_j \sin\left(\sum_{k=1}^{i} \theta_k\right) \tag{3.2}$$

In equation (3.1), P_i represents the point at the end of link i, and θ_i represents the angle in the generalized coordinate i.

The dynamic model for the single support phase of the gait was calculated using the Euler-Lagrange method as described in section 2.3. The final product was given in the standard form as in equation (3.3).

$$M(q)\begin{bmatrix}\ddot{q}_1\\ \vdots\\ \ddot{q}_5\end{bmatrix} + C(q,\dot{q})\begin{bmatrix}\dot{q}_1\\ \vdots\\ \dot{q}_5\end{bmatrix} + G(q) = \begin{bmatrix}0\\ \tau_2\\ \vdots\\ \tau_5\end{bmatrix} \tag{3.3}$$

The matrices M and C are 5x5 matrices, and G is a 5x1 matrix.

3.1.2 Designed Gaits and Switching Surfaces

The initial phase of the gait is when the front leg lifts up and the robot leans backwards in a single support phase. It then reconfigures while still in the air, which makes it possible to reach a desired configuration before impact with the ground. When the swing leg touches the ground, the forces that propagate through the structure makes the stance leg lift off the ground and initiate a new single support phase. The legs roles are switched for the second half step, with the front leg acting as a stance leg. This means that the double support phase is instantaneous, with all of the time of the step spent in the single support phase. The second half step continues in the same fashion, and ends in the same configuration as it had at the start of the first half step. With the configuration of the step being the same in the beginning as in the end, it is possible to initiate further with the same initial conditions as the first step.

The gaits that were designed for the robot were divided into four phases, each with a separate set of virtual constraints. Phase one is when the swing left lifts off the ground, and the robot leans backwards. The second phase is when the motion reverses and the robot starts falling forward until the swing leg hits the ground. The third phase is when the back leg lifts up from the ground, acting as a swing leg. The fourth phase is when the swing leg moves towards the ground again, and completes the step. The step is completed when the swing leg touches the ground. With these motions in mind, the switching surfaces were defined as in the following equations.

$$
\begin{aligned}
\Gamma^{(1\_2)} &= p_- \in \{\dot{\theta} = 0\} \\
\Gamma^{(2\_3)} &= p_- \in \{l_1 \sin(\theta) + l_2 \sin(\theta + q_2) + l_3 \sin(\theta + q_2 + q_3) \\
&\qquad + l_4 \sin(\theta + q_2 + q_3 + q_4) + l_5 \sin(\theta + q_2 + q_3 + q_4 + q_5) = 0\} \\
\Gamma^{(3\_4)} &= p_- \in \{\dot{\theta} = 0\} \\
\Gamma^{(4\_1)} &= p_- \in \{l_1 \sin(\theta) + l_2 \sin(\theta + q_2) + l_3 \sin(\theta + q_2 + q_3) \\
&\qquad + l_4 \sin(\theta + q_2 + q_3 + q_4) + l_5 \sin(\theta + q_2 + q_3 + q_4 + q_5) = 0\}
\end{aligned}
\tag{3.4}
$$

The switching surface between phases 2 and 3 and the switching surface between phase 3 and 4 are equal to the forward kinematics from the stance leg to the endpoint of the swing leg. As discussed in section 3.1.1, the motion requires two models, one for each half step. It was necessary to construct a mapping between the two models. The mappings for the two switching phases that imply impact with the ground are described as in the following equations.

$$
F^{(2\_3)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} q_-^{(2)} + \begin{bmatrix} \pi \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.5}
$$

$$
F^{(4\_1)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} q_-^{(4)} + \begin{bmatrix} -\pi \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.6}
$$

The mappings in equation (3.5) and (3.6) show the relation between the coordinates of the model in figure 3.1 and the model in figure 3.2. These are used between phases 2 and 3, and also between phase 4 and 1. Here, it is possible to switch between the two models by substituting the values of the coordinates for q_-, which will yield the coordinates of the other model. The two other phase switches that occur have a trivial update law, equal to an identity matrix. Between the two models, there was also a need for mapping the generalized velocities, which is given in the following equation.

$$
F_{vel} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \dot{q}_- \tag{3.7}
$$

Several gaits were found, each with different virtual constraints. All the gaits used bézier curves to parametrize the constraints, but the polynomials were of both

second and third order. Several step sizes were made with polynomials of second order, while some were duplicated with third degree polynomials. Table 3.2 shows the second order gaits, and their corresponding walking speed.

Table 3.2: *Different step lengths using second degree polynomials*

Step length (cm)	Time (s)	Speed (cm/s)
2.12	0.8801	2.4088
4.55	1.1137	4.0855
6.37	1.1370	5.6025
6.41	1.1404	5.6208
8.11	1.0593	7.6561
9.26	1.0398	8.9053
inwards knee 2.15	1.0639	2.0209

One of the gaits was made with a different starting configuration, having the first joint bent inwards instead of outwards. This did not impact the performance noticeably. Table 3.3 shows the third order gaits and their corresponding walking speeds.

Table 3.3: *Different step lengths using third degree polynomials*

Step length(cm)	Time (s)	Speed (cm/s)
4.55	0.7777	5.8503
6.41	0.7483	8.5665
8.11	1.1174	7.2581
9.26	0.8404	11.0187
12.12	0.8135	14.8984

The higher order polynomials allowed for optimization of the movement, which clearly shows in the higher speed that was obtained in simulation, as shown in table 3.3.

3.2 Motion analysis

The gaits were designed with specific conditions set on the movement. The stance leg was not to slide, nor lift from the surface. It was to be modeled as a pinned system, with the stance leg being pinned to the ground, acting as an ideal pivot. An analysis of the gaits was needed in order to check if the gaits broke the conditions during the whole step sequence. A study was conducted on the normal force acting on the stance foot from the surface during the motion. An example is shown in figure 3.3, which shows a gait with virtual constraints with third order polynomials.

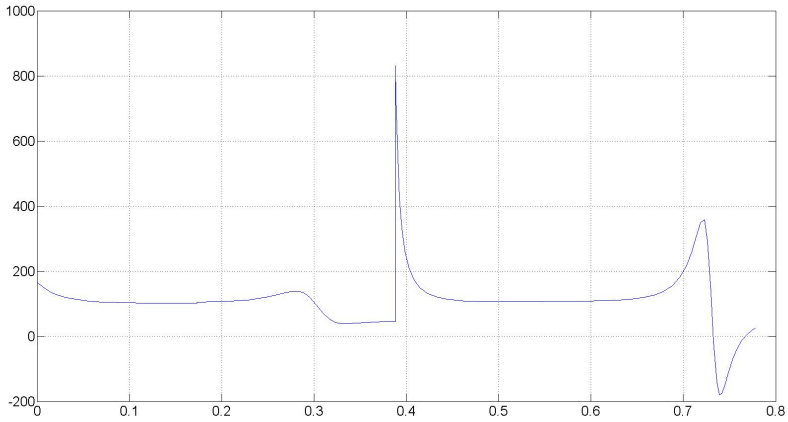


Figure 3.3: *Normal force from the floor acting on the stance foot for an example gait with third degree polynomials as virtual constraints.*

As can be seen from figure 3.3, the normal force was negative during a part of the gait. A negative normal force means that the stance foot would leave the ground during that part of the gait, rendering the constraint broken. This was the case for all of the gaits that were designed with third order constraints. The gaits of the third order would therefore not be realizable, as the robot would enter a flight phase, which was not desirable. The analysis was repeated for the gaits that had virtual constraints with second order polynomials. An example is shown in figure 3.4 for the gait with a step length of $4.55cm$.

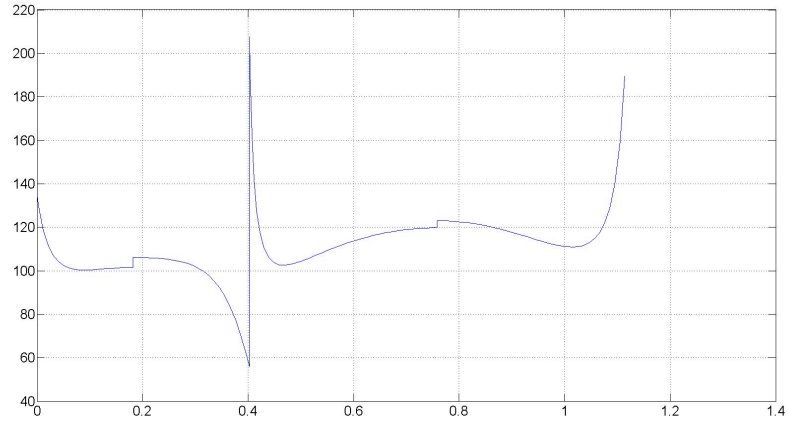


Figure 3.4: *Normal force from the floor acting on the stance foot for an example gait with second degree polynomials as virtual constraints.*

As can be seen from figure 3.4, the normal force was positive throughout the motion. This was the case for all gaits designed with second degree constraints. Consequently, the third degree motions were not used further in the study, due to their infeasibility.

3.3 Implementation

It was desired to make the simulation behave as similar to the physical robot as possible. In order to make the motors in the joints respond realistically to control signals, they needed to have a mathematical model made to represent their behaviour. The four actuators located in the joints of the robot were implemented in the system as DC-motors, as given the specifications from the makers of the robots[2], which is presented in table 3.4.

Table 3.4: *Parameters of actuators*

DC motor	Length (m)	Mass (kg)	Gearbox Ratio	Inertia (kg m^2)	Torque (N m)
Haunch	0.23	2.82	50	$3.25 \cdot 10^{-5}$	0.114
Knee	0.23	2.82	50	$2.26 \cdot 10^{-5}$	0.086

The parameters given in table 3.4 were used in following equation to find the motor dynamics[32].

$$\left(\frac{1}{n^2}J_m + J_L\right)\dot{\omega}_L = \frac{1}{n}T - T_e \tag{3.8}$$

In equation (3.8), n is the gearbox ratio, J_m is the motor inertia, J_L is the load inertia, T is the motor torque and T_e is the load torque.

The calculation of the load torques and inertias was challenging because they changed along with the configuration of the robot. The load torques and load inertias needed to be calculated for each of the motors for each instant in time. The motors all experience different loads depending on what position in they are in. This means that for the first joint, which is located at the junction between link one and link two, the motor will have the remaining links two through five as a load. On the other hand, joint number four will only have link five act as a load. The load inertias of each engine was calculated by using the parallel-axis theorem[33].

$$I = I_{cm} + Mh^2 \tag{3.9}$$

In equation (3.9), I_{cm} is the inertia around the center of mass of the object, M is the object mass, and h is the distance between the center of mass and the connection point of the link. For the motor located at the first joint, the inertias around the joint from the four links to the right were calculated using equation (3.9). These inertias were summed together by the use of the superposition principle. This is shown in figure 3.5. This operation was repeated for the remaining motors. The second motor only had the three remaining links summed together, while the third motor had two, and the fourth motor only had the last link.

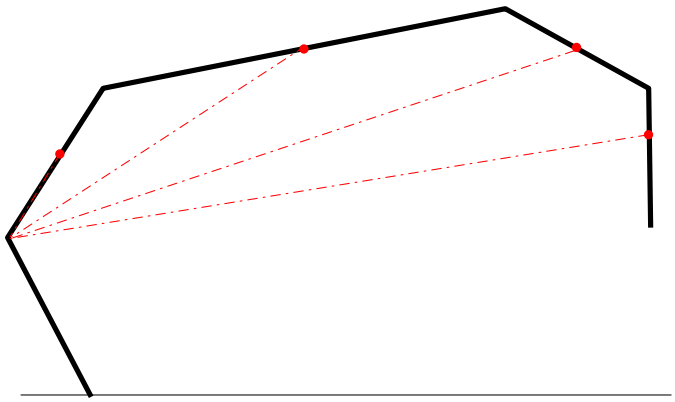


Figure 3.5: *The red dashed lines show the distance from the center of mass for each link to the joint, for which inertia was calculated. The red dots show the centers of mass for each link.*

The torque that each motor received from its load was calculated in a similar fashion. The position of the center of mass of the body that acted as a load to the motor was calculated by the use of the forward kinematics given in equations (3.1)-(3.2). This was further used to calculate the distance from the joint to the point at which the gravitational force acted, which caused torque on the motor. This is shown in figure 3.6.

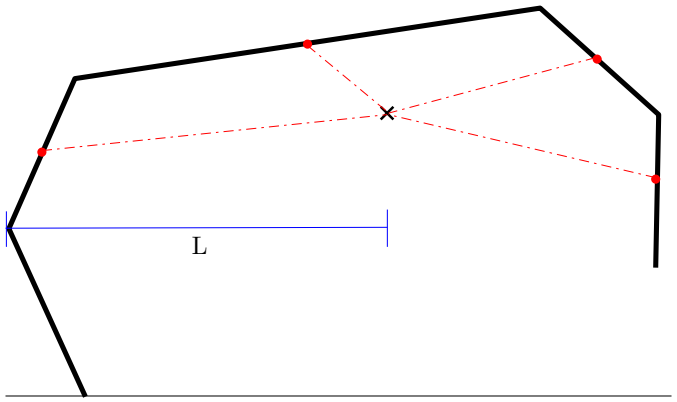


Figure 3.6: *The black cross shows the location of the center of mass for the load body for motor 1. The distance L is used for calculating the load imposed on the motor in joint 1. The red dots show the centers of mass for each link.*

The torque was calculated by the following equation[33].

$$\tau = rF\sin(\theta) \tag{3.10}$$
$$\tau = LgM_i\sin(\theta) \tag{3.11}$$

In these equations, M is the total mass of the body that acts as a load. The torque was calculated for all of the joints. As can be seen from figure 3.6, motor 1 would carry the greatest load of the four motors, as it had four links acting as its load.

A time-saving procedure for implementing the transverse coordinates as shown in equation (2.33) in Matlab/Simulink was utilized. Since the integral value of the transverse coordinates is a function of θ and $\dot{\theta}$, it could be calculated before carrying out the simulation. A two-dimensional lookup table was utilized for this purpose, with the software using interpolation between data points. This was done to reduce the simulation time, as a full simulation without the precalculated integral would take up to 15 hours to complete. The simulation time was reduced to five minutes when the integral was precalculated. Interpolation by using lookup tables did naturally not provide as accurate results as if the integrals were calculated real-time, but it sufficed for the initial simulations.

3.4 State Feedback controller

It was attempted to stabilize the gait with a constant gain state feedback controller, by using a linear quadratic controller with pole placement. Since the linearized system was time-varying, the feedback law was computed using a selected instant in time to find the matrix gain. In this example, the system used was the linear system at time $t = 0$, which is at the start of the optimal trajectory. The system used is given in the following equation

$$\dot{x}_\perp = A(0)x_\perp - B(0)u \tag{3.12}$$

with

$$u = -Kx_\perp \tag{3.13}$$

The poles of the resulting system $(A(0) - B(0)K)$ were placed well into the left hand plane. In the first example, all nine poles were placed evenly between -10 and -5. The results given in figure 3.7 show how this constant gain controller did not cope with the complexity of the system.

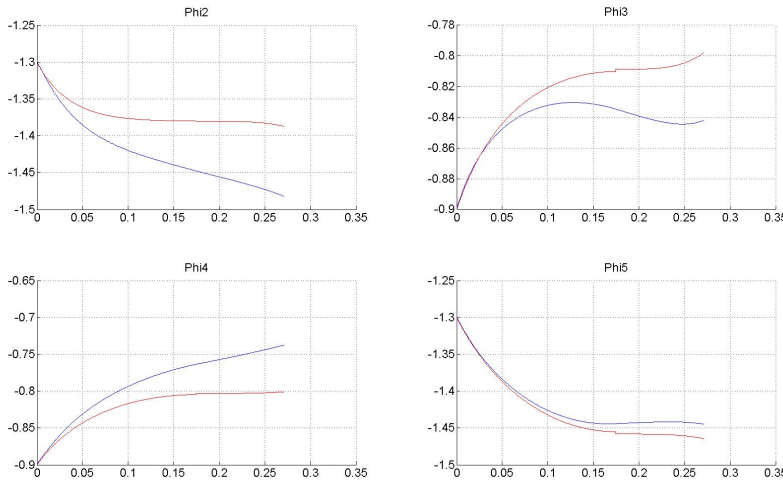


Figure 3.7: *Reference values are shown in red, and values of ϕ_i are shown in blue*

The controller gave a poor following of the reference values for both angle and angular velocity. This can be attributed to the complexity of the system. One of the reasons for these results can be that the load torque and load inertia varied significantly, depending on the configuration of the robot. It can safely be assumed that this controller would not achieve the desired orbital stability.

It was attempted to make another constant gain state feedback controller with a different pole placement. This time, all nine poles of the system (3.12) were placed

evenly between -25 and -20 in the left half plane. The result of this controller showed a significant improvement compared to the previous controller. The actuated states closely followed the reference values throughout the motion. The unactuated variable was therefore close to the optimal. This can be seen in the phase plot of the unactuated variable in figure 3.8, where for the first half step, the reference value and the trajectory overlap each other.

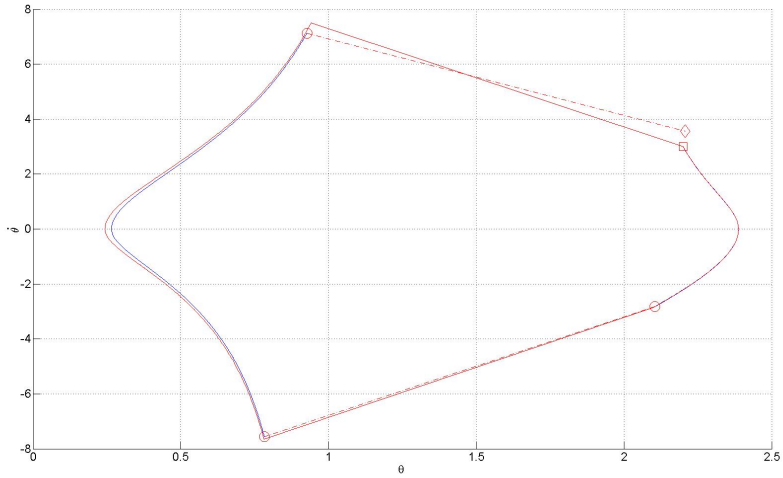


Figure 3.8: *Phase plot of the unactuated variable, related to the optimal trajectory shown in red. The trace of the variable start with a red square in the upper right part of the shape, moving downwards. All of the impacts with the floor are shown as red circles, where the result of the relabeling of the cordinates are shown in dashed lines. The solution of the complete step ends in a red diamond.*

The optimal value and the simulated uncontrolled variable differed during the second half step. The only issue was that the switching surfaces were crossed a little earlier in the trajectory than what would be optimal. That implied that the swing leg would touch the ground before the θ-variable would reach its optimal value. The value of θ at the end of the step sequence is shown with the red diamond, and it showed a notable error from the optimal. The early crossing of the switching surface was a minute problem for the first impact, but the disparity grew for the second impact which corresponds to the upper left red circle of figure 3.8. Altogether, the step executed well with the new controller. All states were close to optimal, as well as the uncontrolled state. The good response is shown as an example in figure 3.9 for the variable q_2.

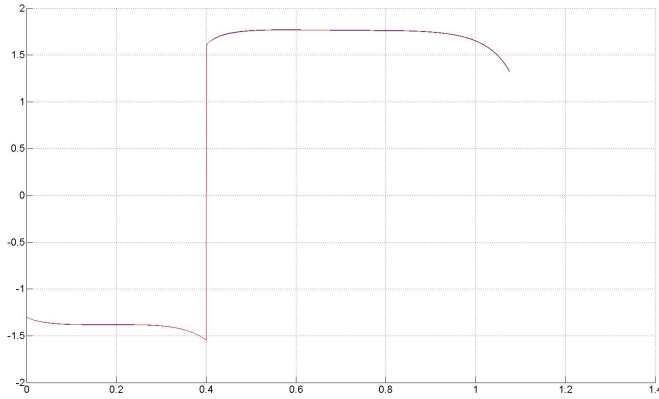


Figure 3.9: *Reference value for ϕ_2 shown in red, and value of q_2 shown in blue*

The variable and the reference value barely differed throughout the full step, even the switch for the first impact gave the controller few problems in following the reference. Also, as seen in the integral value in figure 3.10, it can be seen that the distance to the optimal trajectory was small throughout the step.

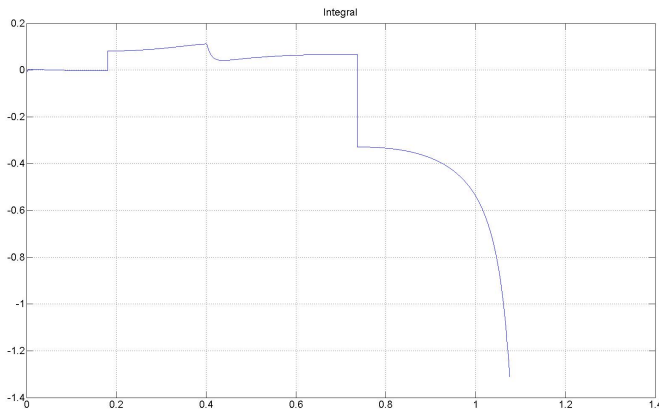


Figure 3.10: *Value of the integral throughout the step*

Because of the promising results, it was attempted to simulate the following step, by using the initial conditions equal to the states at which this first step finished. At the start of the second step, the trajectory changed notably from the optimal. The problem was that the uncontrolled states had a noticeable deviation from the optimal initial conditions. This is showed in figure 3.11, which is a phase plot of the uncontrolled state.

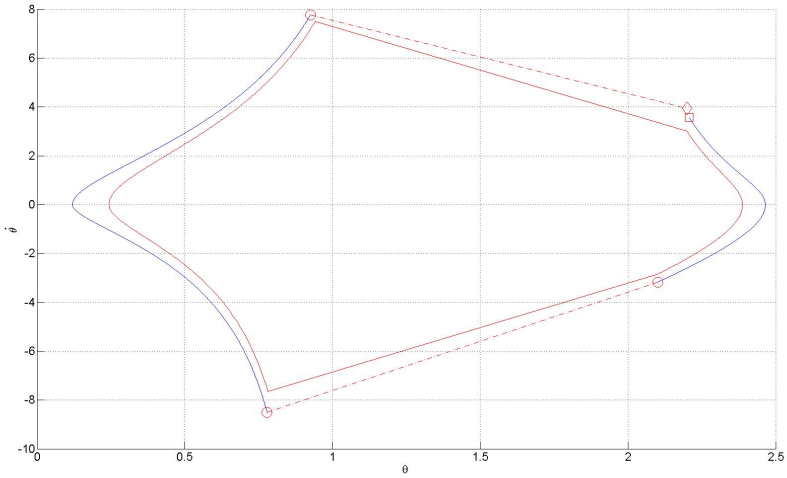


Figure 3.11: *Phase plot of the unactuated variable for step number 2. The optimal trajectory is shown in red, while the simulated trajectory is shown in blue.*

The erroneous initial conditions seemed to escalate the problems through the second step, with the unactuated states drifting further apart from the optimal trajectory. As shown in the phase plot for the unactuated variable in figure 3.11, the solution started a distance away from the optimal, but ended up even further away from the desired trajectory. The controlled variables all followed their references well, as shown for the variable q_2 in figure 3.12.

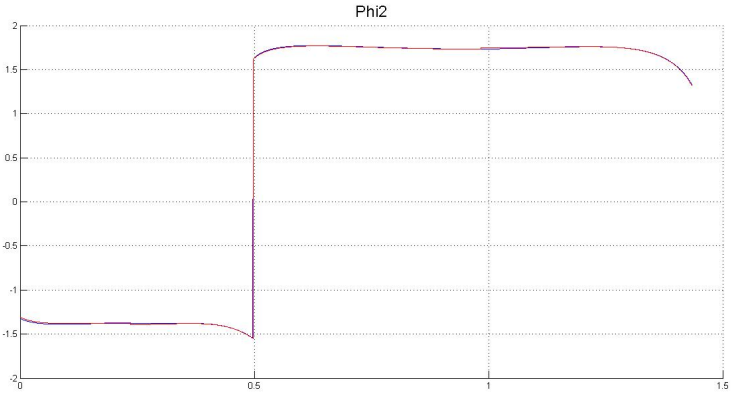


Figure 3.12: *The reference value for ϕ_2 is shown in red, and value of q_2 is shown in blue for step number 2*

Even though the controlled states followed their references well, the controller failed to bring the uncontrolled variable back to the optimal trajectory. This can be seen in the integral value shown in figure 3.13, that did not decrease through the step. It shows an increase towards the end of the step, which means that the uncontrolled variable had drifted even further from the optimal value.

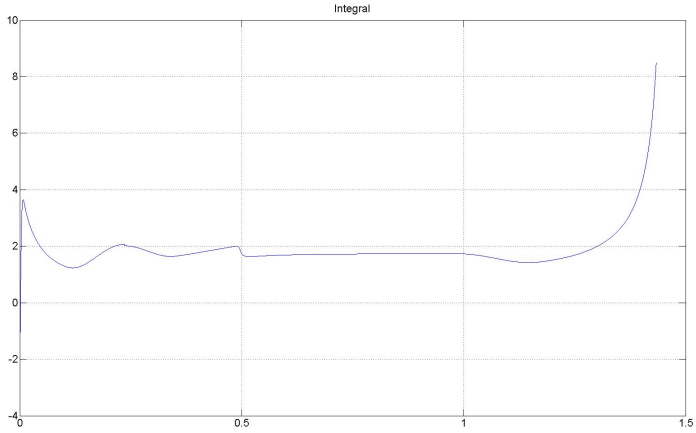


Figure 3.13: *Integral value for step number 2*

A third step was simulated, to uncover if there were any trends in the behaviour of the system. Again, looking at the phase plot for the uncontrolled variable for the third step, it can be seen that the controller could clearly not bring the uncontrolled variable back to the optimal trajectory. This is shown in figure 3.14.

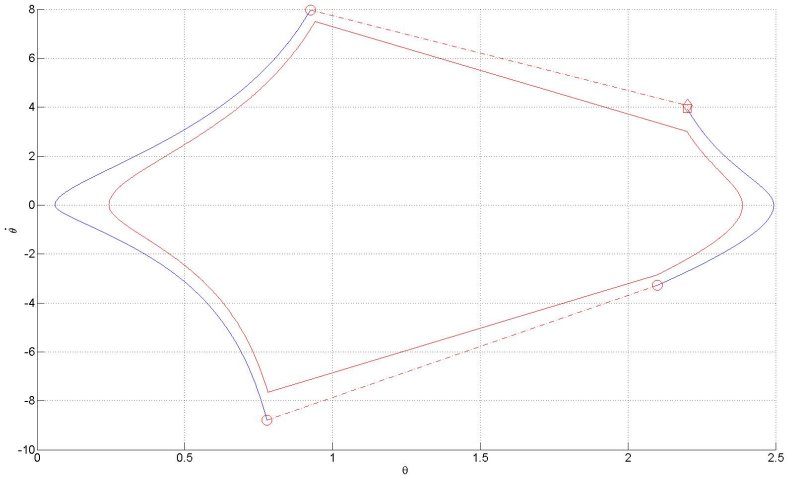


Figure 3.14: *Phase plot for the uncontrolled variable with optimal trajectory shown in red, and simulated trajectory shown in blue for step number 3*

The values at the end of the second half step indicated a trend that the uncontrolled variable drifted further away from the optimal for each step that was taken. The response of the controlled variables for the third step were good, in that they followed their given reference values, as in the two previous steps. The integral value increased even more for this step, as it is a function of how far the system was from the optimal trajectory.

This seemingly good controller did not achieve orbital stability, which can be attributed to the way the reference values were calculated. Following of the reference values was achieved without notable error, but the reference values would need to be adjusted to counter the increasing error of the unactuated variable. This could not done with a constant gain controller, and it could therefore not bring the system back on the optimal trajectory if it did drift off. This was evident during the first phase of the second step. The controlled states followed their reference values almost perfectly, but the uncontrolled values drifted away from the optimal trajectory. This would mean that the calculated reference value were in fact misleading, and would leave the system unable to reach the optimal orbit. Another part of the problem may lie in the fact that this controller is state-dependent, and not dependent on time. The reference values that were calculated depended on the values of the unactuated states θ and $\dot{\theta}$. Consequently, a state-dependent controller will not be a good controller due to the fact that its reference values are not reliable unless the solution is directly on the desired trajectory. Using the four controlled variables to control the motion of the uncontrolled variable would not be possible using a constant gain controller, as even good control of the actuated states did not

equate an orbitally stable system. This example clearly illustrated the challenges of controlling a system with underactuation.

To analyze this result more closely, the solution of the Ricatti equation for this system was analyzed. The solution would vary depending on the chosen matrices defined in equation (2.51), but for all of the simulated values, the result looked similar. The determinant of the solution to the Ricatti equation with constant gain matrices is shown in figure 3.15.

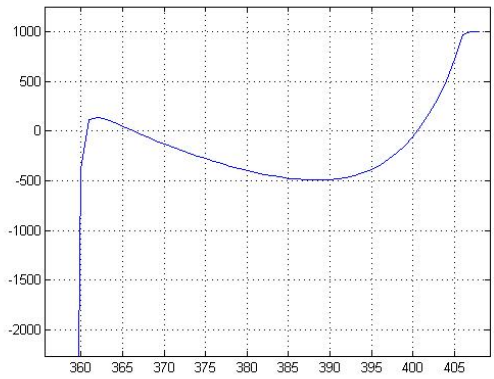


Figure 3.15: *Determinant of the solution P of the Ricatti equation*

To be able to achieve orbital stability, it is desired to have a positive definite determinant of the solution P. As shown in figure 3.15, this was not the case for this system. As discussed in section 2.7, the solutions of the systems gather in a tube around the optimal trajectory. This tube is gradually constricted along the trajectory, until the solutions converge to the optimal trajectory. Transversally to the trajectory this tube has the shape of an ellipsis, or circular shape. When the determinant of P is not positive definite, the shape of the tube that is supposed to constrict the evolution of the solution changes into a non-closed shape. Consequently, gradual convergence could not be showed for this system, and the system therefore could therefore not be orbitally stabilized with a constant gain controller. This result stands for any form of constant gain controller, which is the type of controller that was presented in this section.

Another way of designing a controller, is to make a time-dependent controller. This is done in the method of transverse linearization. This presents additional challenges, as a periodic time-dependent controller in its nature does not seem like a good control strategy for hybrid systems. If there are disturbances in the system, the phase switches will not be made at the designated times. This would mean that the control signals given to the system at a given time instant, is meant to be given for a different place in the orbit than the current position.

3.5 Transverse linearization

As described in section 2.6.2, there were 3 steps necessary to create the transverse system. Step one was to linearize the switching surfaces at the point of intersection with the optimal trajectory, denoted $[q_*, \dot{q}_*]$. This was calculated by taking the jacobian of the switching surface Γ. For switch number one, the surface Γ and the normal vector $\vec{m}$ of its tangent plane $T\Gamma$ were given as

$$\Gamma_1 = p_- \in \left\{\dot{\theta} = 0\right\} \tag{3.14}$$

$$\vec{m}_1 = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0] \tag{3.15}$$

For the more complicated switch number two, the following linearization was defined.

$$\begin{aligned}\Gamma_2 = p_- \in \{&l_1 \sin(\theta) + l_2 \sin(\theta + q_2) + l_3 \sin(\theta + q_2 + q_3) \\ &+ l_4 \sin(\theta + q_2 + q_3 + q_4) + l_5 \sin(\theta + q_2 + q_3 + q_4 + q_5) = 0\}\end{aligned} \tag{3.16}$$

$$\begin{aligned}\vec{m}_2 = [&l_1 \cos(\theta) + l_2 \cos(\theta + q_2) + l_3 \cos(\theta + q_2 + q_3) \\ &+ l_4 \cos(\theta + q_2 + q_3 + q_4) + l_5 \cos(\theta + q_2 + q_3 + q_4 + q_5), \\ &l_2 \cos(\theta + q_2) + l_3 \cos(\theta + q_2 + q_3) \\ &+ l_4 \cos(\theta + q_2 + q_3 + q_4) + l_5 \cos(\theta + q_2 + q_3 + q_4 + q_5), \\ &l_3 \cos(\theta + q_2 + q_3) + l_4 \cos(\theta + q_2 + q_3 + q_4) \\ &+ l_5 \cos(\theta + q_2 + q_3 + q_4 + q_5), \\ &l_4 \cos(\theta + q_2 + q_3 + q_4) + l_5 \cos(\theta + q_2 + q_3 + q_4 + q_5), \\ &l_5 \cos(\theta + q_2 + q_3 + q_4 + q_5), 0, 0, 0, 0, 0]\end{aligned} \tag{3.17}$$

It should be noted that $\vec{m}_3 = \vec{m}_1$, and $\vec{m}_4 = \vec{m}_2$.

Step number two in the process of making the linearized system was to linearize the system along the optimal solution, $[q_*, \dot{q}_*]$. This required rewriting the $\alpha - \beta - \gamma$-equations as a combination of the functions (2.30)-(2.32) and the transverse coordinates that were described in section 2.6.2. The transverse coordinates were given as $X_\perp$ in equation (2.33). This made it possible to define the system (2.36) and (2.38), which is the linearized system. This linearized system is a valid representation of the nonlinear system (2.3) in the vicinity of the optimal trajectory. The relations described in equations (2.26)-(2.26) were inserted into the dynamic equations (2.3), which yielded the g-functions (2.30)-(2.32). The transverse system was then created, as described in section 2.6.2.

Step three of the method was to merge the continuous and discrete parts of the linearization together. To do this, the coordinates of the solution of the linearized system had to be mapped between the step phases. As described in section 2.6.2, this would entail transforming the solution of the linearized system, a point on the plane $TS(t_-)$, to a point on the plane $TS(t_+)$.

The transformation into the original coordinates from the transverse coordinates was done as described in equation (2.43), which yielded the coordinates $[q_i^-, \dot{q}_i^-]$. With these transformed coordinates, it should be noted that they are a solution to the linear system in the tangent plane $TS(T_-)$. These coordinates do not necessarily correspond to a solution for the nonlinear system, which is why the projection shown in figure 2.9 was necessary. The projection gave the solution on the tangent plane to the switching surface, $T\Gamma$, of the nonlinear system.

The projection was calculated with the set of points P_i of the linear solution given, along with the vector $\vec{n}^{(-)}$. The equation to be solved was found to be

$$\begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_{10} \end{bmatrix} + k_1 \begin{bmatrix} \vec{n}_1^{(-)} \\ \vec{n}_2^{(-)} \\ \vdots \\ \vec{n}_{10}^{(-)} \end{bmatrix} = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{10} \end{bmatrix} \tag{3.18}$$

Here, the vector of points V denoted the points on the tangent plane to the switching surface Γ, namely $T\Gamma$. The solution was projected along the vector $\vec{n}^{(-)}$, as shown in figure 2.9. To calculate the k to solve the equations, the equation was formulated as follows, with the vector P denoting the solution as first calculated by the linearization.

$$(P + k_1\vec{n}^{(-)}) \perp \vec{m} \tag{3.19}$$

$$(P + k_1\vec{n}^{(-)}) \cdot \vec{m} = 0 \tag{3.20}$$

$$k_1\vec{n}^{(-)} \cdot \vec{m} + P \cdot \vec{m} = 0 \tag{3.21}$$

$$k_1 = -\frac{P \cdot \vec{m}}{\vec{n}^{(-)} \cdot \vec{m}} \tag{3.22}$$

This equated to the coordinates of the projection equaling as follows.

$$V = \left(-\frac{P \cdot \vec{m}}{\vec{n}^{(-)} \cdot \vec{m}} \right) \vec{n}^{(-)} + P \tag{3.23}$$

The coordinates V represented the solution on the tangent plane to switching surface, $T\Gamma$. To start the next phase of the gait, these coordinates needed to be represented on the tangent plane $TS(T_+)$. To project these coordinates into the tangent plane $TS(T_+)$, the projection operation had to be repeated, with the formula being slightly different. It was calculated as follows

$$\begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{10} \end{bmatrix} + k_2 \begin{bmatrix} \vec{n}_1^{(+)} \\ \vec{n}_2^{(+)} \\ \vdots \\ \vec{n}_{10}^{(+)} \end{bmatrix} = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_{10} \end{bmatrix} \tag{3.24}$$

Here, the points V_i represented the points found in the previous projection, and the points W_i represented the resulting points on the tangent plane $TS(T_+)$. The equation for calculating k became as in the following equations.

$$(V + k_2\vec{n}^{(+)}) \perp \vec{n}^{(+)} \tag{3.25}$$

$$(V + k_2\vec{n}^{(+)}) \cdot \vec{n}^{(+)} = 0 \tag{3.26}$$

$$k_2\vec{n}^{(+)} \cdot \vec{n}^{(+)} + V \cdot \vec{n}^{(+)} = 0 \tag{3.27}$$

$$k_2 = -\frac{V \cdot \vec{n}^{(+)}}{\|\vec{n}^{(+)}\|^2} \tag{3.28}$$

This equated to the coordinates of the projection equaling

$$W = \left(-\frac{V \cdot \vec{n}^{(+)}}{\|\vec{n}^{(+)}\|^2}\right) \vec{n}^{(+)} + V \tag{3.29}$$

As the points W_i represented a point on the tangent plane $TS(T_+)$ in original coordinates, they needed to be rewritten in transverse coordinates. This was done as was shown in equation (2.40). The results of these operations could be given as a mapping directly from transverse coordinates at time t^- to the transverse coordinates at time t^+, as shown in equation (2.44). This mapping had to be made for each switching surface. This step completed the three steps that were needed to make the linearized system.

The linearized system could now be compared to the nonlinear system. This was done to control if the behaviour of the linear system really did represent the nonlinear system in the vicinity of the optimal trajectory. If it id, a controller could be designed for the linearized system, and be implemented on the nonlinear system. A comparison of the response of the linear system and the nonlinear system was therefore made. Because of how the system matrices were designed for the linear system, it was natural to look at the integral value to compare the responses. This is shown in figure 3.16.

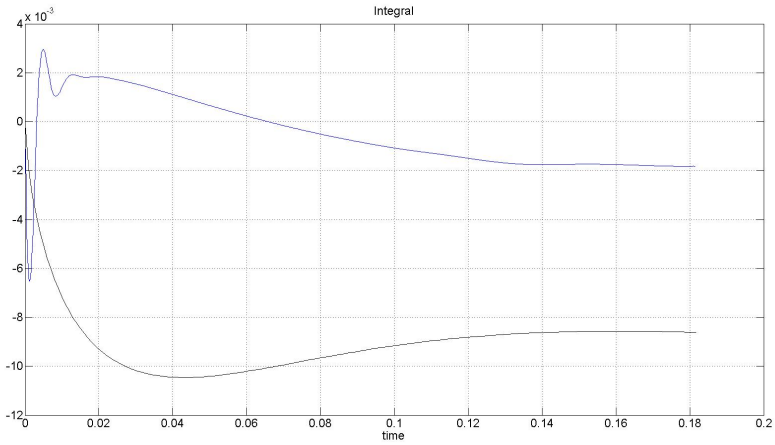


Figure 3.16: *Comparison of the response of the linear system shown in black, and the nonlinear system shown in blue.*

As can be seen from figure 3.16, the responses were not similar. A reason for this could be that the linear system was not simulated with the dynamics of the DC-motors that were implemented for the nonlinear system. This means that the response of the linear system would not equal that of the nonlinear system. Consequently, the actuation on the linear system would not achieve stabilization of the nonlinear system.

A controller was made available by Prof. Sergei Gusev, which was to stabilize the linear system with the intent to test it on the nonlinear system. To implement this time-varying controller, it was made as a discrete controller which changed every 0.001 second. This was because it was not designed to be a continuous function, but it was calculated from the linearization of the optimal solution at given time instants. Its results on controlling the linearized system is shown in figure 3.17.

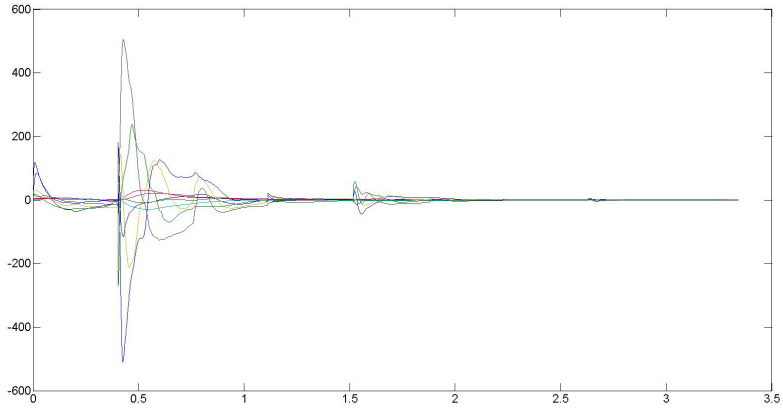


Figure 3.17: *Result of the controller on the linearized system*

As can be seen from figure 3.17, the controller managed to control the linearized system. The controller was then implemented on the nonlinear system. The response was surprisingly bad, as the robot failed to even correctly simulate the first phase of the gait. It should be noted that stabilizing the linear system was very challenging, due to the very large jumps in the states excited by the impacts with the floor. Bearing this in mind, the results indicated that a motion like this was not realistic for forward locomotion of this robot.

The designed gait motion is not a conventional motion for this kind of biped. A conventional gait would be having a prolonged double support phase after impact, which would include reconfiguring the robot before lifting up a leg for the next single support phase. This would have been much simpler, as the impacts with the floor would not excite a large and unstable phase switch. The angular velocities would be much lower, which would make a more stable motion. The gait studied in this thesis used the forces upon the impact of the robot to lift the stance leg off the floor. This was a very complicated type of gait, and is usually used on much simpler robots, such as the biped studied by Goswami et al.[10], which uses the compass gait. The problem with this did not lie in the existence of the gait, as it could very well exist in theory, but in the implementation on a real robot. The problems are related to the angular velocities required to attain the calculated forces at impact. They were very high, and by using them on mechanical equipment could easily damage the robot and its instruments. It could be possible to design a gait of this type with lower velocities, but this would affect the possible step lengths in a negative way, as they would shorten by a great amount.

3.6 New Motion

Because of the results from the previously designed gait, it was attempted to design a new gait. This new gait was of a different character, as it was designed with a prolonged double support phase, allowing the robot to reconfigure itself on the ground, before starting the next single support phase. To initialize the gait, it was made such that the configuration itself would create torque around the stance foot, lifting the leg up. This removed the necessity for the robot to make a jumping motion to lift the swing leg off the ground. This resulted in a quirky configuration, as shown in figure 3.18.

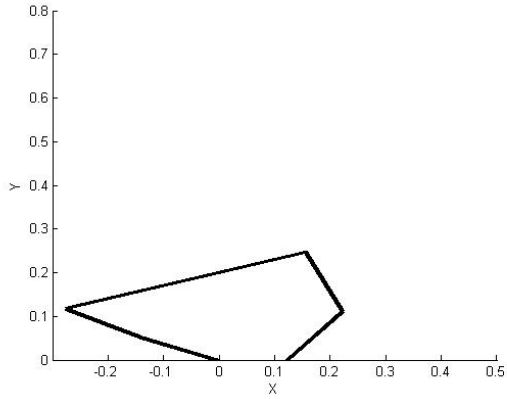


Figure 3.18: *Starting configuration of the new designed gait*

This was a sharp contrast to the symmetric configuration of the original motion described in section 3.1.2. As this new gait allowed the robot to reconfigure itself on the ground, this would also reset any offset the uncontrolled variable could have attained during the single support phase. Consequently, a simple constant gain controller could be used. This way, as long as the motors followed their reference values, any drifting of the uncontrolled variable would be reset every half step, allowing a simpler controller than for the previous gait. The switching surfaces that were used were the same as in the previous gait, and are given in equation (3.4). The virtual constraints for this gait were made to be linear functions to make the motion as simple as possible. The constraint coefficients are given in appendix B. The resulting step was $1.87cm$ long, which was not very long considering the size of the robot. The gait is shown in figure 3.19, where six snapshots were taken of the evolution of the motion.

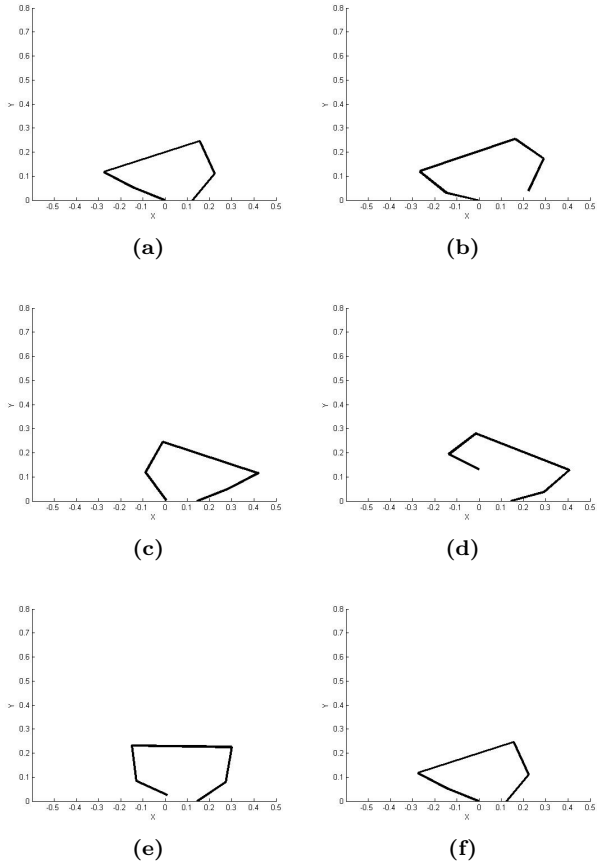


Figure 3.19: *Example of movement for the SemiQuad(from left to right) for the new gait*

A PID-controller was designed, which gave good results for this gait, as shown in figure 3.20. Figure 3.20 shows a phase plot of the uncontrolled variable.

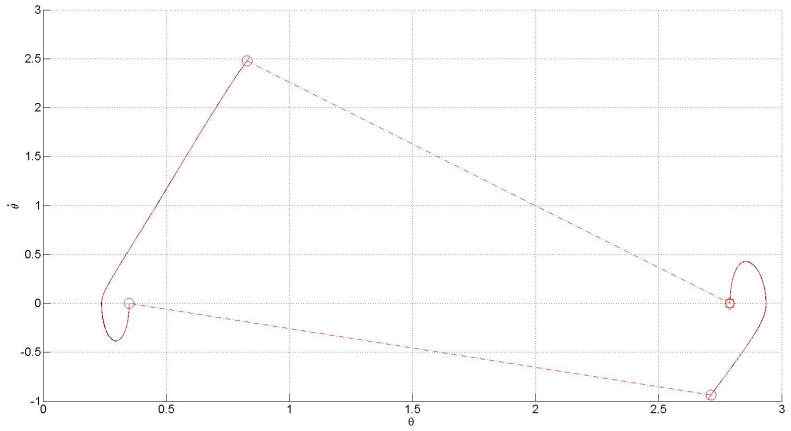


Figure 3.20: *Phase plot for the gait with a PID-controller is shown in blue, starting on the red square on the right side of the plot*

As can be seen from figure 3.20, the blue line is barely visible as it overlaps with the optimal trajectory. The PID-controller managed to follow the references well, partly because of their linear property. This made the robot perform very close to the optimal trajectory, with any deviation in the uncontrolled state resetting at every impact of the swing leg with the floor. This allowed the robot to make continuous steps. Because of the uncontrolled variable being reset onto the optimal value at every half step, any error was eliminated during the step. Therefore, even though the controller not necessarily was orbitally stabilizing, since the nature of the motion corrected the errors that occurred, it can be argued that it was an orbitally stable motion. This was made possible due to the simplicity of the gait, which essentially was the joining of two distinct half steps, in place of one continuous motion.

Chapter 4

Conclusion

The goal of this thesis was to design a stabilizing controller for the previously designed gaits for the SemiQuad robot. To do this, an analysis was performed on the designed gaits. This was used to determine whether or not the gaits were feasible. The method of transverse linearization was then used, to analyse the orbital stability properties of the system. A controller was then devised to stabilize the time-varying linearized system, and subsequently used to control the nonlinear system. A new gait was then designed, and a constant gain controller was made for this new gait.

It was seen from the results that the gaits that were designed with third order virtual constraints were not feasible, since the stance foot lifted off the ground at a point along the step sequence. The gaits with second order constraints were found to be feasible. The continued calculation was done for a gait of step length 4.55cm, with second degree virtual constraints. The transverse linearization was found for this gait, but the response on the linear system did not correspond well with the behaviour of the nonlinear system. The controller that stabilized the linearized system did not stabilize the nonlinear system. It was also found that the type of gait that was designed was not realizable, due to the complexity of the motion. Therefore, a new and much simpler type of gait was designed. This new motion was found to be orbitally stabilizable with a constant gain controller, due to the simplicity of the motion.

Chapter 5

Further Work

This project was focused on designing a stabilizing controller, using the transverse linearization. Since it was found that the predefined motions were not easily stabilizable, there should be designed a new type of motion with much simpler mechanics. This would include prolonged double support phases, and also starting the upwards motion of the swing leg with a jumping motion. With a simple and realizable gait, a controller could be designed and tested on the robot itself, and not only in simulation.

It could be interesting to see if it was possible to realize gait over inclining and declining terrain. It would also be interesting to see if any running gaits can be made, including phases of flight, where there are no legs in contact with the ground. This could yield much higher walking speed.

References

[1] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E. R. Westervelt, C. Canudas-de Wit, and J. Grizzle, "Rabbit - a testbed for advanced control theory," *IEEE Control Systems Magazine*, vol. October, pp. 57–79, 2003.

[2] Y. Aoustin, G. Garcia, and P. Lemoine, *Estimation of the Absolute Orientation of a Five-link Walking Robot with Passive Feet*, 2007, ch. 3, pp. 31–44.

[3] A. Muraro, C. Chevallereau, and Y. Aoustin, "Optimal trajectories for a quadruped robot with trot, amble and curvet gaits for two energetic criteria," *Multibody System Dynamics*, vol. 9, pp. 39–62, 2003.

[4] M. Raibert, M. Chepponis, and H. B. Brown, "Running on four legs as though they were one," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 2, pp. 70 – 82, June 1986.

[5] H. Dickinson, "How animals move: An integrative view," *Science Magazine*, vol. 288, pp. 100–106, 2000.

[6] D. G. E. Hobbelen and M. Wisse, *Humanoid Robots: Human-like Machines*, 2007, ch. Limit Cycle Walking, pp. 277–294.

[7] A. Kuo, "Choosing your steps carefully: Trade-offs between economy and versatility in dynamic walking bipedal robots," *IEEE Robotics and Automation Magazine*, vol. 14, no. 2, pp. 18–29, 2007.

[8] M. Vukobratović, A. A. Frank, and D. Juricic, "On the stability of biped locomotion," *IEEE Transactions on Biomedical Engineering*, vol. BME-17, no. 1, pp. 25 –36, jan. 1970.

[9] Y. Hurmuzlu and G. D. Moskowitz, "Role of impact in the stability of bipedal locomotion," *International Journal of Dynamics and Stability of Systems*, vol. Vol. 1, Iss. 3, pp. 217–234, 1986.

[10] A. Goswami, B. Espiau, and A. Keramane, "Limit cycles in a passive compass gait biped and passivity-mimicking control laws," *Autonomous Robots*, vol. vol. 4, issue 3, pp. 273–286, 1997.

[11] C. Chevallereau, E. R. Westervelt, and J. W. Grizzle, "Asymptotically stable running for a five-link, four-actuator, planar bipedal robot," *International Journal of Robotics Research*, vol. 24, pp. 431–464, 2005.

[12] Y. Aoustin, C. Chevallereau, and A. Formal'sky, "Numerical and experimental study of the virtual quadrupedal walking robot-semiquad," *Multibody System Dynamics*, vol. vol.16, issue 1, pp. 1–20, 2006.

[13] Y. Aoustin, S. Bellavoir, G. Branchu, C. Chevallereau, A. Formal'sky, P. Lemoine, and P. Molina, "A semi quadruped walking robot - first experimental results," in *6th International Conference on Climbing and Walking Robots CLAWAR'03, Catania : Italy*, 2003.

[14] J. Grizzle, G. Abba, and F. Plestan, "Asymptotically stable walking for biped robots: analysis via systems with impulse effects," *Automatic Control, IEEE Transactions on Automatic Control*, vol. 46, no. 1, pp. 51 –64, jan 2001.

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

[16] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[17] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. John Wiley & Sons, 2006.

[18] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. Taylor & Francis Group, LLC, 2007.

[19] A. Shiriaev, L. Freidovich, A. Robertsson, R. Johansson, and A. Sandberg, "Virtual-holonomic-constraints-based design of stable oscillations of furuta pendulum: Theory and experiments," *IEEE Transactions on Robotics*, vol. vol. 23, no. 4, no. 4, pp. 827 –832, aug. 2007.

[20] A. Shiriaev, J. Perram, and C. Canudas-de Wit, "Constructive tool for orbital stabilization of underactuated nonlinear systems: Virtual constraints approach," *Automatic Control, IEEE Transactions on Automatic Control*, vol. 50, no. 8, pp. 1164 – 1176, aug. 2005.

[21] A. Shiriaev, J. Perram, A. Robertsson, and A. Sandberg, "Explicit formulas for general integrals of motion for a class of mechanical systems subject to virtual constraints," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on Decision and Control*, vol. 2, dec. 2004, pp. 1158 – 1163 Vol.2.

[22] G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*. Elsevier Science & Technology Books, 1997.

[23] H. Khalil, *Nonlinear Systems*. Prentice Hall, 2002.

[24] 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," *Annual Reviews in Control*, vol. vol 32, no 2, pp. 200–211, 2008.

[25] V. I. Arnol'd, *Ordinary Differential Equations*. Spreinger-Verlag, 1992.

[26] A. Shiriaev and L. Freidovich, "Transverse linearization for impulsive mechanical systems with one passive link," *IEEE Transactions on Automatic Control*, vol. vol. 54, no. 12, pp. 2882–2888, 2009.

[27] M. Spong, "Partial feedback linearization of underactuated mechanical systems," in *Proceedings of Inernational Conference on Intelligent Robots and Systems. Munich, Germany*, 2004.

[28] A. Shiriaev, L. Freidovich, and S. V. Gusev, "Transverse linearization for controlled mechanical systems with severeal passive degrees of freedom," *IEEE Transactions on Automatic Control*, vol. vol. 55, no. 4, pp. 893 – 906, 2010.

[29] L. Freidovich, A. Shiriaev, and I. R. Manchester, "Stability analysis and control design for an underactuated walking robot via computation of a transverse linearization," in *Proceedings of the 17th World Congress, Seoul, Korea 2008 The International Federation of Automatic Control*, 2008.

[30] G. Leonov, "Generalization of the andronov-vitt theorem," *Regular and Chaotic Dynamics*, vol. vol.11, no.2, pp. 281–289, 2006.

[31] S. Pchelkin, A. Shiriaev, L. Freidovich, U. Mettin, S. Gusev, and W. Kwon, "Natural sit-down and chair-rise motions for a humanoid robot," in *49th IEEE Conference on Decision and Control (CDC), 2010*, dec. 2010, pp. 1136 –1141.

[32] O. Egeland and J. Gravdahl, *Modeling and Simulation for Automatic Control*. Marine Cybernetics AS, 2002.

[33] P. Tipler and G. Mosca, *Physics for Scientists and Engineers*. W. H. Freeman and Company, 2008.

Appendix A

Controller Parameters

This section presents the controller parameters for the simulations presented in the thesis. They are here presented transposed, due to space.

Controller presented in figure 3.7

$$K = \begin{bmatrix} -2.4222 & -1.2037 & 1.0367 & 0.6924 \\ -1.1914 & -6.3263 & 12.7070 & 3.8830 \\ -13.2223 & 14.1460 & 9.6868 & 9.2950 \\ -2.9952 & 0.9077 & 10.8703 & 1.2295 \\ -4.8447 & 6.6042 & -0.2010 & 27.2250 \\ 5.4305 & -1.7999 & 2.6835 & 0.8367 \\ -3.0376 & 8.8886 & 1.9118 & 1.7119 \\ 0.1891 & 0.4564 & 7.3649 & 0.0338 \\ -0.8774 & 1.1171 & -0.0214 & 11.0151 \end{bmatrix}^T \tag{A.1}$$

Controller presented in figure 3.8 and after, in section 3.4

$$K = \begin{bmatrix} 0.0097 & 0.0019 & -0.0068 & -0.0080 \\ 529.7798 & -33.0216 & -82.3883 & -80.7595 \\ -19.1812 & 536.9022 & -96.7908 & -38.5045 \\ 27.4238 & -42.3056 & 560.2238 & -28.2862 \\ -13.4905 & 19.8392 & -25.2540 & 536.3877 \\ 47.1890 & -2.0508 & -4.1224 & -4.0300 \\ -0.2638 & 46.2430 & -4.4296 & -2.1342 \\ 1.1388 & -1.7145 & 47.4427 & -1.2341 \\ -0.5069 & 0.7655 & -1.0760 & 46.4661 \end{bmatrix}^T \tag{A.2}$$

PID-controller presented in section 3.6

$$P = 800 \qquad I = 0.1 \qquad D = 10 \tag{A.3}$$

Appendix B

Path Coefficients

This section presents the path coefficients of the virtual constraints of the gait designed in section 3.6. Since the constraints are linear, the points define the value of the constraint at the given boundary points of theta, with the constraint being a linear function in between the two points.

Phase 1

$$\theta_{begin} = 2.7900 \tag{B.1}$$
$$\theta_{end} = 2.9000 \tag{B.2}$$

$$\begin{bmatrix} \phi_2(\theta_{begin}) & \phi_2(\theta_{end}) \\ \phi_3(\theta_{begin}) & \phi_3(\theta_{end}) \\ \phi_4(\theta_{begin}) & \phi_4(\theta_{end}) \\ \phi_5(\theta_{begin}) & \phi_5(\theta_{end}) \end{bmatrix} = \begin{bmatrix} -0.100 & -0.350 \\ -2.400 & -2.250 \\ -1.400 & -1.000 \\ -1.194 & -1.400 \end{bmatrix} \tag{B.3}$$

Phase 2

$$\theta_{begin} = 2.9369 \tag{B.4}$$
$$\theta_{end} = 2.7000 \tag{B.5}$$

$$\begin{bmatrix} \phi_2(\theta_{begin}) & \phi_2(\theta_{end}) \\ \phi_3(\theta_{begin}) & \phi_3(\theta_{end}) \\ \phi_4(\theta_{begin}) & \phi_4(\theta_{end}) \\ \phi_5(\theta_{begin}) & \phi_5(\theta_{end}) \end{bmatrix} = \begin{bmatrix} -0.4338 & -0.4000 \\ -2.1997 & -2.2000 \\ -0.8659 & -1.3000 \\ -1.4691 & -1.3705 \end{bmatrix} \tag{B.6}$$

Phase 3

$$\theta_{begin} = 0.3500 \tag{B.7}$$
$$\theta_{end} = 0.2500 \tag{B.8}$$

$$\begin{bmatrix} \phi_2(\theta_{begin}) & \phi_2(\theta_{end}) \\ \phi_3(\theta_{begin}) & \phi_3(\theta_{end}) \\ \phi_4(\theta_{begin}) & \phi_4(\theta_{end}) \\ \phi_5(\theta_{begin}) & \phi_5(\theta_{end}) \end{bmatrix} = \begin{bmatrix} 0.100 & 0.400 \\ 2.400 & 2.150 \\ 1.300 & 1.000 \\ 1.245 & 2.000 \end{bmatrix} \tag{B.9}$$

Phase 4

$$\theta_{begin} = 0.2374 \tag{B.10}$$
$$\theta_{end} = 0.8376 \tag{B.11}$$

$$\begin{bmatrix} \phi_2(\theta_{begin}) & \phi_2(\theta_{end}) \\ \phi_3(\theta_{begin}) & \phi_3(\theta_{end}) \\ \phi_4(\theta_{begin}) & \phi_4(\theta_{end}) \\ \phi_5(\theta_{begin}) & \phi_5(\theta_{end}) \end{bmatrix} = \begin{bmatrix} 0.4378 & 1.1940 \\ 2.1185 & 1.4000 \\ 0.9622 & 2.4000 \\ 2.0951 & 0.1000 \end{bmatrix} \tag{B.12}$$