**NTNU – Trondheim**
Norwegian University of
Science and Technology

# Creation and Control of Gaits for the SemiQuad Robot

Harald Hareide

Master of Science in Cybernetics and Robotics
Submission date: May 2014
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

**Problem Description**

The object of this thesis is divided in two distinct parts. The first section focuses on the creation of orbitally stable periodic gaits for the SemiQuad robot, using the concept of limit cycle walking. The second part focuses on the implementation of the gaits through the creation of an exponentially orbitally stabilizing controller capable of ensuring the SemiQuad completes the predesigned motion.

Assignment given: 07. January 2014
Supervisor: Prof. Anton Shiriaev, ITK

## Preface

This thesis is written to document the work and research carried out during the Spring Semester of 2014, in the final term of the Cybernetic studies at the Norwegian University of Science and Technology [NTNU]. I chose to work on the subject of legged robotics as I consider this to be an exciting field with a large potential for rapid scientific advances.

I would first and foremost like to thank my supervisor, Prof. Anton Shiriaev, for having given me the chance to contribute to this field of research, and for all the help and guidance I received while working on this topic. I would furthermore like to extend my thanks to Dr. Leonid Paramonov for his help and explanations of various subjects throughout my work. Finally, I would like to extend my thanks to Prof. Sergei V. Gusev, for having created the final controller used in this thesis.

*Harald Hareide*
May 30, 2014

**Abstract**

The object of this thesis was divided in two distinct parts, the first being a study into discovering functional gaits for the SemiQuad Robot, whereas the second part involved implementing the predesigned gaits with the aid of a suitable control system. The stability paradigm employed was that of Limit Cycle Walking, and the gaits were thus created as a series of exact repetitions of a closed trajectory in state space. To simplify the dynamics of the system, the notion of Virtual Holonomic Constraints were employed. All joint values were defined by a single parameter $\theta$, reducing the dynamic model from a 10th order system to that of a 2nd order system, while the the complete motion of the SemiQuad was described by the state space evolution of $[\theta, \dot{\theta}]$. Through the use of Bèzier polynomials, appropriate parametrizations for the virtual constraints were found through customized optimization algorithms. Various cost functions were designed, primarily ensuring a cyclic and therefore functional gait, but also terms increasing the energy efficiency or the velocity of the SemiQuad was developed. The discovered gaits were used as reference trajectories for the full simulation of the 10-dimensional system. To suitably describe the deviation from the reference trajectories, the notion of transverse coordinates were introduced. This reduced the description of the system to a set of coordinates transverse to the reference trajectory, that could then be linearized to allow the use of linear control techniques. Various control methods were tested before a functioning controller was found. The final controller was developed through the study of SDP based approximations of stabilizable solutions of the periodic Riccati differential equations.

Several gaits were created, achieving the first goal of the thesis. It was further shown that increasing the order of the parametrizations led to better results, both in terms of energy efficiency and higher velocities. The system was in addition able to recreate the gaits with simulations using the full 10-dimensional system as long as it was simulated with correct initial values. The gaits were shown to be unstable without any control action implemented, with any type of error eventually leading to a failure in performing the gait. With control implemented however, the gaits were stabilized provided the errors were kept within a reasonable boundary, thus demonstrating the possibility of creating orbitally exponentially stable gaits for the SemiQuad system.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

In this paper we will examine the creation of gaits for the SemiQuad robot[1] created at Nantes in France. The robot itself is technically a biped, though it will use a gait type known as Curvet[6], which in turn could be extended to be used for quadruped motion. The choice of this topic was inspired by a strong interest in legged robotics and the study of natural locomotion.

## 1.1 History

The field of robotics has in the span of decades seen a tremendous growth, with new discoveries and rapid innovation arriving continuously. In some fields we have gone further than anyone dared to hope. Industrial robots are producing products with an unrivalled efficiency, while in other areas robots can be used to perform dangerous tasks, which previously demanded human labour. Yet, the field of legged locomotion is still in the research phase, as most legged robots built to date are unable to act independently and function well enough to be of commercial interest. There are many reasons for this. First of all, the simple act of walking is not a simple motion at all when transferred to mechanical structures such as robots. For industrial robotics, the control schemes created for other processes could be transferred without much alterations, and complex operations were reduced to creating joint movement in state space accurate enough to allow for a tool center to do the intended work. In addition, as the robots are to function in a controlled environment, they can be preprogrammed exactly, without the need for any adaptations. For legged robotics however, whole sections of stability and control had to be created for stable movement to be possible. The problem is severely expanded if the robot is to move in unknown terrain, which puts a much stronger reliance on controllers and sensors to achieve a successful walking motion[12]. As a result of this, there exists no "one way" to create robotic walking gaits, but rather several paradigms with different advantages and disadvantages. Looking to nature, we must also acknowledge that even us humans spend the better part of our two first years standing up and falling until we ourself get the hang of it. Even among animals there is a distinct "learning curve" while they are young before they can move as efficiently as adult specimens. Yet, when the movement is perfected, it can be seen as the end product of an evolutionary process, capable of moving in an optimal way with respect to speed and energy efficiency[4]. Clearly it would be wise to copy such movement patterns, and try to implement them in our Robots. This is however not a straightforward task, as robots obviously lack the extensive control that humans and animals have. In addition, humans and animals have a complex muscular system, which for nearly all movement is overactuated by several degrees. Such complexity is not yet available through the use of actuators and other mechanical aids alone, meaning natural motion cannot be copied exactly.
The field of legged robotics is nevertheless a booming research area, with many various types of robots appearing on the scene. There is a strong wish both commercially and sociologically for the study and creation of such robots, as the uses for functioning models would be endless. The most popular robot to date is probably Honda's ASIMO,

a bipedal robot that is constantly being upgraded and can now achieve an impressive range of tasks. Boston Dynamics have also come into the spotlight with several impressive Quadrupeds such as BigDog, RoboMule, Cheetah and the newly arrived WildCat. They exhibit impressive stability while carrying out versatile tasks, the first two moving through uneven terrain while carrying heavy loads and the latter two moving at very high speeds. Results are coming fast, but these versions still represent the very cutting edge, and the price is most likely reflected in this. There is still a long way to go until legged robots can exist and be helpful in our everyday lives.

## 1.2 Project Scope

A mathematical model for the robot will be created through the use of Euler-Lagrange. Then through the use of virtual Holonomic Constraints the motion of the system will be described by a single variable, rendering the degrees of freedom equal to one and allowing the full dynamics of the system to be described by the reduced dynamics. The complete walking motion of the semiQuad will be developed, which will consist of movement phases calculated from the reduced dynamics of the system and impact phases signifying the impact of the legs to the ground. Several types of gaits will be created by optimizing the virtual constraints with respect to criteria concerning energy efficiency and speed. The robustness of the gaits will then be tested by simulating them using the full dynamics with varying degrees of errors. To implement the gaits the notion of transverse linearization will be used, reducing the system coordinates to those transverse to the solution. A control scheme will be created with the goal of eliminating errors in the system and thus ensuring a robust gait with exponential orbital stability.
The first goal is to expand on the optimization of previous work, and add models of much higher order than what has been previously created. The idea is further to show that with the increased freedom found in higher order models, the optimization routines will be free to discovered better solutions. The final goal is to create a controller able to ensure completion of the gait within a suitable range of errors, thus demonstrating that the proposed motion could be implemented on a real system.

## 1.3 Layout

This report is built up in the following way. Chapter two will study the design of the SemiQuad, the theory of gaits and the stability paradigm known as limit cycle walking. Chapter three will look into the mathematical modelling of the SemiQuad, by calculating the energy relations using Euler-Lagrange. Chapter four looks at the theory behind virtual holonomic constraints and the creation of a Hybrid System model. Chapter five will introduce the optimization routines being used and the theory behind their selection. Chapter six will implement the simulation of the gaits using the reduced dynamics and compare the results between varying parametrization orders and step lengths, thus marking the conclusion of the first part of this thesis. Chapter seven will demonstrate the simulation of a gait using the full dynamics of the system, while

also exploring the error tolerance of the gait. Chapter eight will describe the theory behind transverse linearization, as well as the fundamental building blocks for describing deviations from a predefined gait. Chapter nine will explore the theory behind control of the SemiQuad as well as introduce the control scheme implemented in this thesis. Chapter ten will explore the functionality of the proposed controller for a large range of errors and simulation conditions. Finally, chapter eleven and twelve will respectively contain the conclusion to this thesis and a brief study into possible future directions for the study of the SemiQuad system.

# 2 Theory

The robot that will be examined in this thesis is the 5-DOF SemiQuad. This section will briefly explain the design of the robot, and the evolution of a whole gait using the notion of limit cycle walking. For a full study of the robot however, the reader is suggested to examine [1].

## 2.1 The SemiQuad

The SemiQuad is composed of a platform and two identical double-link legs with knees, for a total of 5 rigid links. The feet are passive while the joints consist of 4 electrical DC motors with gearbox reducers. The robot will perform the Curvèt gait as described in [6], which is a simple type of movement similar to that of a caterpillar. The simple nature of the SemiQuad, in combination with the simplicity of the gait allows us to extend the motion to a quadruped robot, limited to motion in the 2D-Space (the sagittal plane), with each front leg (respectively each hind leg) moving together as a pair. The problem can then be said to be the discovery of gaits for a hypothetical Quadruped robot, as the SemiQuad can be considered as an implementation of a virtual quadruped, hence its name[1]. The SemiQuad is pictured in fig(1) while the dimensions and physical constraints of the SemiQuad is available in table 1.

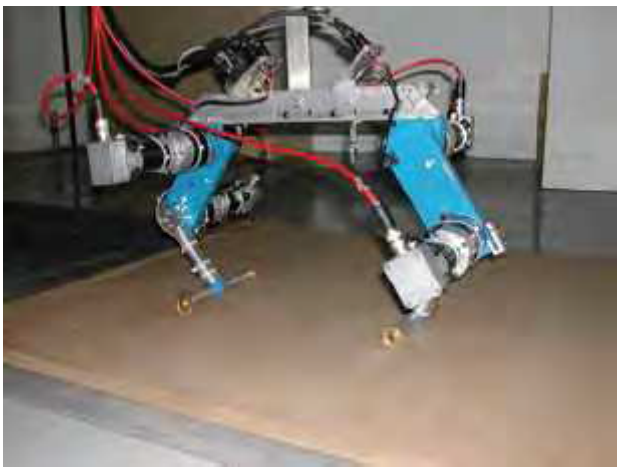


Figure 1: Photography of the SemiQuad Robot.

## 2.2 Taking a Step

The goal of the report is as mentioned to create gaits that the SemiQuad will be able to perform. As such, the movement itself can be separated into two phases that will

| Link | Mass | length | Center of Mass Location | Moment of inertia (around COM) |
|---|---|---|---|---|
| link 1 | $m_1 = 0.40$ | $l_1 = 0.15$ | $s_1 = 0.083$ | $I_1 = 0.0034$ |
| link 2 | $m_2 = 3.21$ | $l_2 = 0.15$ | $s_2 = 0.139$ | $I_2 = 0.0043$ |
| link 3 | $m_3 = 6.618$ | $l_3 = 0.375$ | $s_3 = 0.1875$ | $I_3 = 0.3157$ |
| link 4 | $m_4 = 3.21$ | $l_4 = 0.15$ | $s_4 = 0.139$ | $I_4 = 0.0043$ |
| link 5 | $m_5 = 0.40$ | $l_5 = 0.15$ | $s_5 = 0.083$ | $I_5 = 0.0034$ |

Table 1: Physical parameters of the SemiQuad

be referenced throughout the paper. The moment where both feet are touching the ground will be known as the Double Support phase[1], which is an overactuated, stable position. By overactuation it is meant that the DOF(degrees of freedom) $n$ are less than the number of actuators $d$. When the robot starts moving, it will systematically lift one of its legs, starting with the front leg. This phase is known as the Single Support Phase[1], which can be seen as an underactuated, unstable position. We will further simplify this phase by adding that the stance leg (leg not currently lifted) will remain in perfect contact with the ground throughout the movement. This allows us to look upon the single support phase as a 5-DOF robotic manipulator, fixed to a point at its base. The credibility of this is naturally based on the friction to the surface and thus the angle between the stance leg and the ground should be kept above a minimum value to ensure that slipping would be unlikely. As it has 5 DOF, it can be said to have an underactuation of $n - d = 1$. Using the same requirement for the double support phase, we have that both legs must be kept in contact with the ground, reducing the DOF to 3 and thereby giving us an over-actuated system as $d - n = 1$.
A full step then consists of the lifting of the front leg, an extension and return to double support phase, then lifting of the hind leg, a contraction and return to double support phase, finishing the gait and returning the SemiQuad to its original configuration. The evolution of a whole step is presented in fig 2.
One question remains however. The SemiQuad was introduced as an underactuated system with both feet being passive, so how is the motion around the ankle generated? The answer is gravity, and how the remaining actuated joints can use gravity to their benefit. In fact, the actuators directly act on the shape or posture of the robot, thereby changing the position of the center of mass, and thus, the moment arm through which the gravity acts on the robot[5]. Motion is thereby generated in the ankle indirectly by the torque generated by the motors in the joints, an important notion for the later creation of gaits.
Further, while performing a gait, the double support phase will be seen as an instantaneous phase in which the leg currently being lifted makes an impact with the floor, resulting in the former stance leg being lifted into the air. This is done to increase the efficiency of the movement, while also avoiding the requirement of ensuring zero angular velocity before hitting the floor. The total gait can therefore be seen as overlapping sections of Single Support Phases, with the legs changing between being stance or swing

leg, separated by instantaneous double support phases. Our robot can therefore be said to constitute a hybrid system, a notion that will be developed further in the following chapters.
Below is a summary of the simplifications and speculations that has been considered before starting the work on creating gaits for the SemiQuad:

1. The gait consists only of the phases mentioned thus far, namely the single support phase, and the double support phase. Therefore there exists no flight phase, in which no part of the robot is in contact with the ground.

2. The impact moment as the swing leg hits the ground is considered to be instantaneous, and the impact itself is considered to be inelastic. The configuration of the robot is therefore not expected to change, and the impact will therefore only influence the angular velocities of the joints.

3. When the robot is in the Single Support phase the stance leg will not slip, be lifted or in any way be moved from its position. The system can therefore be modelled as a robotic manipulator fastened to the ground. The design of the robot must naturally ensure that such an assumption is reasonable.

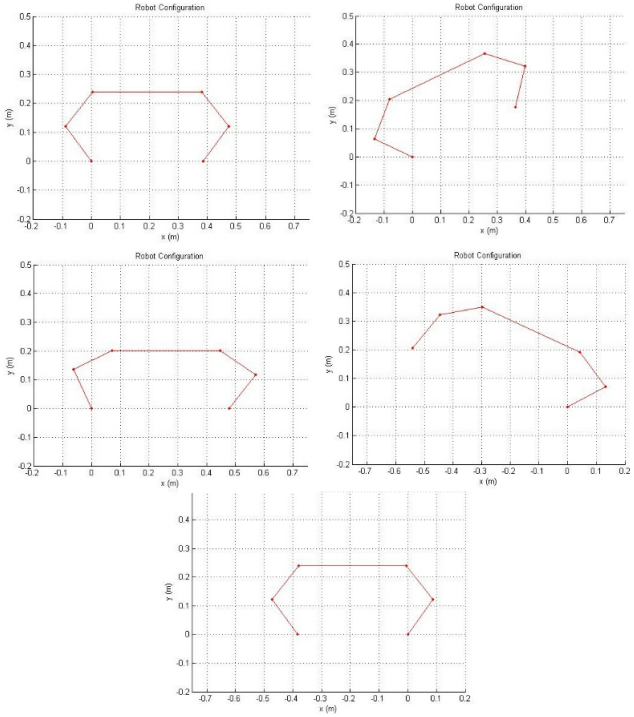



Figure 2: An example of a full gait.

## 2.3 Stability and Limit Cycle Walking

Closely related to the subject of walking is naturally the subject of stability. Walking is after all a naturally unstable motion, and therefore many techniques for finding gaits often started as problems aiming to ensure stability during movement. However, it can be shown that rigorously following various stabilizing criteria often lead to unnatural and inefficient solutions. In fact, the more restrictive the constraints are, the less freedom is left for optimizing performance[2], thereby making it more difficult to create efficient gaits. One of the oldest types of stability for walking robots is that of Static Stability[2]. As the name implies, the method entails keeping the Center of Mass within the support polygon formed by the feet[2]. Thus stability is ensured, but one has very little freedom for creating gaits, which again tends to only allow very slow movement. A more recent paradigm is that of Zero Moment Point(ZMP) introduced by Miomir Vukobratovic. The ZMP refers to the location within the base of support about which the ground contact forces exert no moment in the lateral and foraft directions[3]. Thus for the ZMP paradigm the stability is ensured when the stance foot remains in flat contact with the floor at all times[2], and when the ZMP exists within the support polygon spanned by the two feet[12]. This leads to a stability paradigm that is less strict than that of static stability, which leads to more types of gaits becoming available. It is still too strict however to lead to the natural fluency of human or animal motion and will therefore not be able to reach the same level of efficiency. To create more natural gaits, it becomes clear that the walking motion must be even less constrained, which bring us to the next concept: Limit Cycle Walking.

As mentioned, most paradigms for creating walking motion in legged robotics was based on creating a method to stabilize the robot, with the walking of the robot being nearly just an added bonus. It was much because of this that the paper written by Tad McGeer known as Passive Dynamic Walking[13] received such a great response when it was released. Through the use of passive walkers who could move down gentle slopes without any actuation, he showed that it is possible to create stable walking, without restraining the robot with arduous stability requirements. He further showed that it was easy to shift this aspect over to robots with actuation, as in fact adding power to a passive walker involves a comparably minor modification[13]. Since then much study have gone into exploring the notion of dynamic walking. At its core is a desire to impose fewer artificial constraints and thus allow for more freedom in finding efficient, natural, fast and robust walking motions[2]. This led to the development of the term "Limit Cycle Walking", which can be defined as a nominally periodic sequence of steps that is stable as a whole but not locally stable at every instant in time[2]. The walking motion is then a series of exact repetitions of a closed trajectory in state space[2]. Following this line of thinking we are free to create any number of specific gaits for the SemiQuad, by ensuring that the gait constitutes a limit cycle. This means having the same configuration at the end of the motion as at the start of the motion, so the motion itself constitutes a periodic sequence. This can be implemented as a part of the requirements of the optimization that will be performed. As mentioned in the previous section, we know that the gait of the SemiQuad will be separated into interchanging phases of single support and double

support. A full gait will have been completed when the last leg to have been lifted has been settled on the ground. In this moment the final values for the configuration of the robot must be identical to the starting configuration. Primarily this entails that the SemiQuad finishes its motion in the same configuration in which it started. Additionally since the plan is to take advantage of the impact to generate angular velocities, the SemiQuad will also have an initial angular velocity. Thus the angular velocity at the end of the motion must be such that it is equal to the initial angular velocity after the impact has taken place.

# 3 Mathematical Model

In this chapter the mathematical model for the SemiQuad will be developed. The first part will contain a description of the kinematics of the robot. The second part will calculate the dynamics of the robot using the method of Euler-Lagrange. In the last section an impact model will be created so as to describe the instantaneous double support phase introduced in section 2.2.

## 3.1 Kinematic Model

As explained in section 2.1, the SemiQuad consists of 5 rigid links, with the legs of the robot being identical in size and mass. To be able to describe the motion of the robot it is necessary to describe the change in each degree of freedom with a single parameter. When it comes to the design of the robot gait we will mainly be concerned with the single support phase, as this constitutes the movement. The double support phase is as mentioned instantaneous and it will be handled in chapter 3.3.
In the single support phase the robot has 5-DOF, and with the simplifications added in 2.2 we can treat the robot as a standard robotic manipulator. The robot is restricted to move in the sagittal plane, and its movement is then fully described by the angle of each joint in the same way as a 5-R robotic manipulator (despite the ankle being passive). The choice for the angles is then arbitrary as long as the movement of each joint is suitably described. In addition, as the gait is divided in two sections of single support, it is natural to choose two sets of angles to describe the robot. The configuration for the first part of the movement, that is when the front leg is raised, is modelled as in figure 3, while the configuration for the second part of the gait, when the hind leg is raised, is modelled as in figure 4.

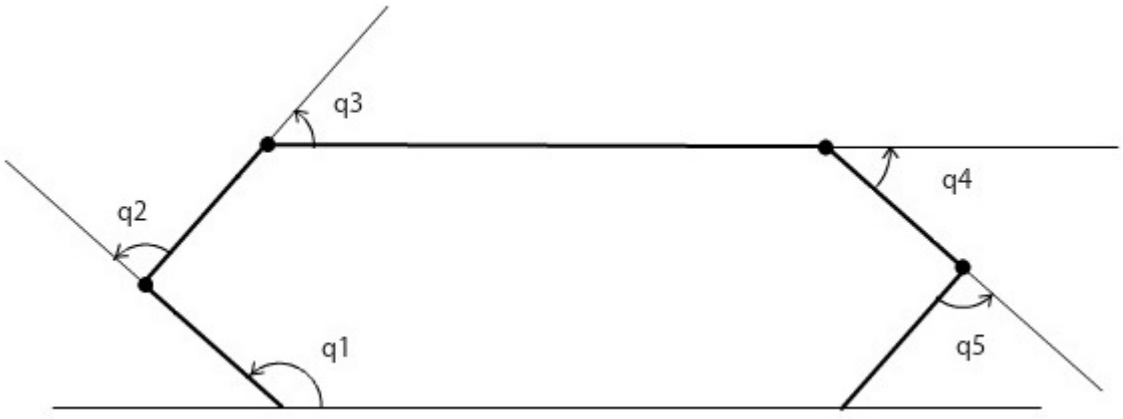



Figure 3: Configuration of the Semiquad for the first half of the movment

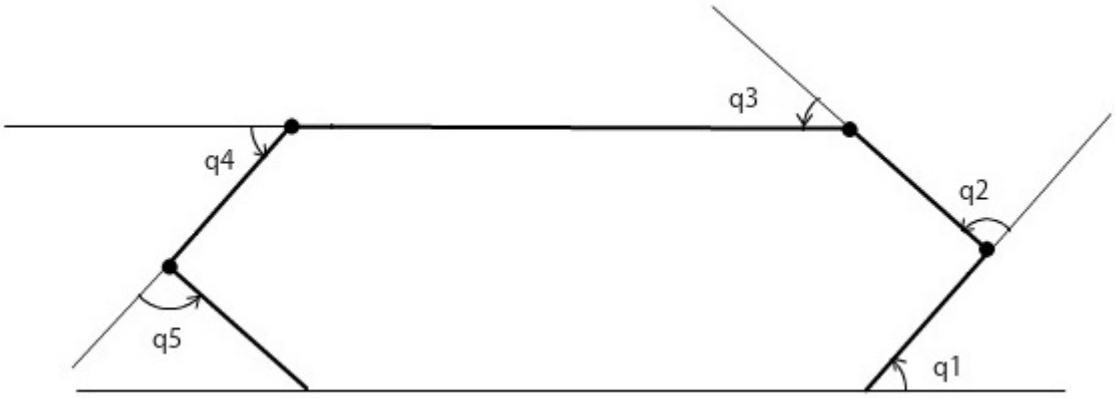



Figure 4: Configuration of the Semiquad for the second half of the movment

In figure 3 it can be seen that the angles $q_2$ to $q_5$ will be negative, whereas in figure 4 we see that all angles will have positive values. These configurations are identical to those used in [16].
The forward kinematics for the SemiQuad can be calculated from the evolution of the centers of mass of each of the five rigid links. As the variables $q_1$-$q_5$ describe each degree of freedom, they will function as the generalized coordinates for the system. Using the known parameters shown in table 1, the forward kinematic functions can then be described as follows

$$
\begin{aligned}
P_1 =&[(l_1 - s_1)cos(q_1),\\
&(l_1 - s_1)sin(q_1)]\\
P_2 =&[l_1cos(q_1) + (l_2 - s_2)cos(q_1 + q_2),\\
&l_1sin(q_1) + (l_2 - s_2)sin(q_1 + q_2)]\\
P_3 =&[l_1cos(q_1) + l_2cos(q_1 + q_2) + (l_3 - s_3)cos(q_1 + q_2 + q_3)\\
&l_1sin(q_1) + l_2sin(q_1 + q_2) + (l_3 - s_3)sin(q_1 + q_2 + q_3)]\\
P_4 =&[l_1cos(q_1) + l_2cos(q_1 + q_2) + l_3cos(q_1 + q_2 + q_3)) + s_4cos(q_1 + q_2 + q_3 + q_4),\\
&l_1sin(q_1) + l_2sin(q_1 + q_2) + l_3sin(q_1 + q_2 + q_3) + s_4cos(q_1 + q_2 + q_3 + q_4)]\\
P_5 =&[l_1cos(q_1) + l_2cos(q_1 + q_2) + l_3cos(q_1 + q_2 + q_3)) + l_4cos(q_1 + q_2 + q_3 + q_4)\\
&+ s_5cos(q_1 + q_2 + q_3 + q_4 + q_5),\\
&l_1sin(q_1) + l_2sin(q_1 + q_2) + l_3sin(q_1 + q_2 + q_3) + s_4sin(q_1 + q_2 + q_3 + q_4)\\
&+ s_5sin(q_1 + q_2 + q_3 + q_4 + q_5)]
\end{aligned}
\quad , \quad (1)
$$

where $P_i$ is the mass center of link $i$ in the 2-dimensional coordinate frame (x,y), starting with the position of the stance leg as origo. Since the legs of the robot are identical, and since the COM of the body section lies exactly in the center of the body,

these functions will be valid for both configurations of the robot.

## 3.2 Dynamic Model

With the forward kinematic equations calculated as in (1), the dynamics of the SemiQuad can be found. The dynamic model constitutes the full description of each link of the system and will act as the physical description of the robot for later simulations.

### 3.2.1 Euler-Lagrange

The dynamics will be calculated using the notion of Euler-Lagrange. First the Lagrangian is calculated, which is given as

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{P}(q) \tag{2}$$

where $q$ and $\dot{q}$ are the generalized coordinates and their respective velocities as explained in the previous section. The lagrangian is thus the difference between the sum of total kinetic energy $\mathcal{K}$ and the sum of total potential energy $\mathcal{P}$ calculated from each of the links. Once the energy function is known, a second-order dynamical model immediately follows[5] from Lagrange's equation given as

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i, \qquad i = 1, .., n \tag{3}$$

which results in $i = 1, 2 \ldots 5$ equations, one for each degree of freedom in the system. Each equation is equal to a variable $\tau_i$ which is the sum of generalized forces and torques acting on joint $i$. These equations are known as the equations of motion, and with these the system model can be fully described. It is simpler to write (3) in compacted form

$$\mathcal{M}(q)\ddot{q} + \mathcal{C}(q, \dot{q})\dot{q} + \mathcal{G}(q) = \Gamma \tag{4}$$

where $\mathcal{M}(q)$ is the matrix of inertia, $\mathcal{C}(q, \dot{q})$ contains the centrifugal and coriolis accelerations, $\mathcal{G}(q)$ is the gravity vector and $\Gamma$ is the vector of generalized forces and torques. $\mathcal{M}(q)$ and $\mathcal{C}(q, \dot{q})$ are (5x5) matrices while $\mathcal{G}(q)$ and $\Gamma$ are (5x1) vectors. Then, choosing a state vector as

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{5}$$

which for 5-DOF has dimension 10, the model can be written in state space form as

$$\dot{x} = \begin{bmatrix} \dot{q} \\ \mathcal{M}^{-1}(q)\left[-\mathcal{C}(q, \dot{q})\dot{q} - \mathcal{G}(q)\right] \end{bmatrix} \tag{6}$$

### 3.2.2 Energy Calculations

To calculate the Lagrangian, the total energy of the system must be found. The potential energy $\mathcal{P}$ is simply the total sum of the potential energy from each link, calculated from the y coordinates found in (1). These coordinates are given as the vertical position of the center of mass with respect to the ground.

$$\mathcal{P} = \sum_{i=1}^{N} g y_i m_i \tag{7}$$

Likewise, the total kinetic energy is found by looking at the contribution of kinetic energy from each link, the contribution from a single link being described by

$$\mathcal{K}_i(q, \dot{q}) = \frac{1}{2} v_{c_i}^T(q, \dot{q}) m_i v_{c_i}(q, \dot{q}) + \frac{1}{2} w_i^T(q, \dot{q}) \mathcal{I}_i w_i(q, \dot{q}) \tag{8}$$

Here $v_{c_i}$ is the linear velocity of the COM of the i-th body. Likewise $w_i$ is the angular velocity of the i-th body, whereas $\mathcal{I}_i$ is the inertia matrix with respect to the center of mass. The total kinetic energy is thus a sum of kinetic energy from translation and kinetic energy due to rotation. The expression for total kinetic energy is then given as

$$\frac{1}{2} \dot{q}^T \sum_i \left[ \mathcal{J}_L^{(i)}(q)^T m_i \mathcal{J}_L^{(i)}(q) + \mathcal{J}_A^{(i)}(q)^T \mathcal{I}_i \mathcal{J}_A^{(i)}(q) \right] \dot{q} \tag{9}$$

$$\frac{1}{2} \dot{q}^T M(q) \dot{q} \tag{10}$$

The symbols $\mathcal{J}_L^{(i)}$ and $\mathcal{J}_A^{(i)}$ are the linear velocity Jacobian and the angular velocity Jacobian of the i-th link respectively. This expression can be further simplified[16] by noticing that the expression for the rotational kinetic energy (the second part of (9)) can be reduced to that of (11), which summed up is equal to (12)

$$K_{r,i} = I_i w_i^2 \tag{11}$$

$$K_r = I_1 w_1^2 + I_2 w_2^2 + I_3 w_3^2 + I_4 w_4^2 + I_5 w_5^2 = \frac{1}{2} \dot{q}^T \mathcal{I} \dot{q} \tag{12}$$

and the kinetic energy equation is then reduced to that of

$$\frac{1}{2} \dot{q}^T \sum_i^n \left[ \mathcal{J}_L^{(i)}(q)^T m_i \mathcal{J}_L^{(i)} + \mathcal{I} \right] \tag{13}$$

and thus the Lagrangian can be calculated. With the Lagrangian, the equations of motion are readily available by calculating (3).

### 3.2.3 Alternative Method

Instead of calculating the kinetic and potential energy and then equating the Lagrangian, it is also possible to arrive at the equations of motion solving the matrices from equation (4), namely ($\mathcal{M}$, $\mathcal{C}$, $\mathcal{G}$), directly using the following equation

$$\sum_j m_{ij}(q)\ddot{q} + \sum_i \sum_j c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k \tag{14}$$

for each equation of motion $k$ of the dynamics. Here $m_{ij}$ is given by the row $i$ and column $j$ of matrix $\mathcal{M}(q)$, whereas $c_{ijk}$ are the Christoffel symbols of the first type. Having already computed $\mathcal{M}(q)$ from (10)-(13), these can be found as

$$c_{ijk} = \frac{1}{2}\left(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k}\right) = c_{jik} \tag{15}$$

which has the symmetry as shown. The matrix $\mathcal{C}(q, \dot{q})$ can then be calculated with each entry given as

$$\mathcal{C}(k, j) = \sum_{i=1}^{5} c_{ijk}\dot{q}_i \quad k = 1, 2..5,\ j = 1, 2..5 \tag{16}$$

Then with both $\mathcal{M}(q)$ and $\mathcal{C}(q, \dot{q})$ calculated, only the vector $\mathcal{G}(q)$ remains, which can be calculated from the potential energy (7) as

$$g_k = \frac{\partial \mathcal{P}}{\partial q_k} \tag{17}$$

and thus equation (4) can be written directly.

### 3.2.4 The Final Model

Regardless of which method is used, the equations of motion for the SemiQuad will take the following form

$$\mathcal{M}(q)\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \vdots \\ \ddot{q}_5 \end{bmatrix} + \mathcal{C}(q, \dot{q})\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_5 \end{bmatrix} + G(q) = \begin{bmatrix} 0 \\ \tau_2 \\ \vdots \\ \tau_5 \end{bmatrix} \tag{18}$$

we see that the first equation of motion equates to zero, which is due to the fact that the stance leg of the robot is unactuated. This equation, (18.1), is therefore referred to as the zero dynamics of the system, an important feature that will be expanded upon in the following chapters.
The dynamics were calculated using the MuPAD feature of Matlab Symbolic Toolbox and the procedure can be viewed in Appendix C. Both methods introduced in this chapter was used, mainly as a way to check for errors as both methods should lead to the same set of equations.

### 3.3 Impact Model

As explained in section 2.2, the double support phase will be modelled as an instantaneous and inelastic impact of the current swing leg to the ground. As further explained, it would for the sake of efficiency be wise to use this impact in the direct motion of the robot instead of reducing it by enforcing that angular velocity must reach zero before setting the leg down. Hence, an impact will occur when the swing leg is placed on the ground. With the restrictions also given in 2.2 it is known that the impact will not influence the configuration of the robot, in fact the impulsive forces due to the impact may result in an instantaneous change in the velocities, but there is no instantaneous change in the positions[5]. The problem is thus reduced to describing the change that occurs in the velocities as a result of the impact and apply this to an update law that can be used to calculate the new velocities $\dot{q}^+$ from the old velocities $\dot{q}^-$. For the sake of consistency, the impact map model will be applied before the swap of generalized coordinates as explained in chapter 3.1. This swap will however occur straight after the impact takes place as the previous stance leg is assumed to lift from the ground without interaction[15], and thus a new phase of single support is initiated.
Now to fully describe the impact model an unpinned (N+2)-DOF model of the robot is needed[15]. This is done by adding to the already completed model for the single support phase an extra set of two variables. Setting $q_s = q_1, q_2, \dots q_5$ as the first five generalized coordinates, we can add two additional coordinates $P_e = (p_e^h, p_e^v)$ which are the Cartesian coordinates of some fixed point on the robot[15]. For simplicity the point at the end of the swing leg is chosen given by modifying the last expressing in (1).

$$\begin{aligned} P_e =&[l_1 * cos(q_1) + l_2 cos(q_1 + q_2) + l_3 cos(q_1 + q_2 + q_3)) + l_4 cos(q_1 + q_2 + q_3 + q_4) \\ &+ l_5 cos(q_1 + q_2 + q_3 + q_4 + q_5), \\ &l_1 sin(q_1) + l_2 sin(q_1 + q_2) + l_3 sin(q_1 + q_2 + q_3) + s_4 sin(q_1 + q_2 + q_3 + q_4) \\ &+ l_5 sin(q_1 + q_2 + q_3 + q_4 + q_5)] \end{aligned} \tag{19}$$

The total set of generalized coordinates is thus $q_e = (q_s, P_e)$, and with the method of Euler-Lagrange the system model becomes[15]

$$\mathcal{M}_e(q)\ddot{q}_e + \mathcal{C}_e(q_e, \dot{q}_e)\dot{q}_e + \mathcal{G}_e(q)_e = \mathcal{B}_e(q_e)u + \delta F_{ext} \tag{20}$$

where $\delta F_{ext}$ represents the vector of external forces acting on the robot due to the contact between the swing leg and the ground. The impulsive instantaneous nature of the impact is reflected in the $\delta$ notation. When integrating (20) over a finite duration of the impact one obtains[15]

$$\mathcal{M}_e(q_e^+)\dot{q_e}^+ - \mathcal{M}_e(q_e^-)\dot{q_e}^- = F_{ext} \tag{21}$$

where $q_e^-$ and $q_e^+$ are as previously described the angular velocities before and after the impact respectively and

$$F_{ext} := \int_{t^-}^{t^+} \delta F_{ext}(\tau) d\tau \tag{22}$$

is the resultant after integrating the impulse forces. Equation (21) thus expresses the conservation of momentum of the impact.
The values of the derivatives of the generalized coordinates from the single support phase before the impact ($\dot{q}_1^-$ to $\dot{q}_5^-$) are calculated from the single support model and are available at the moment of impact. Further, the Cartesian coordinates $P_e$ can be calculated from the known generalized coordinates from the single support phase such that all coordinates before the impact can be described as[15]

$$q_e^- = \begin{bmatrix} q_s^- \\ \Upsilon_e(q_s^-) \end{bmatrix} \tag{23}$$

and the derivatives of the added generalized coordinates are readily available as

$$\dot{q}_e^- = \begin{bmatrix} I_{NxN} \\ \frac{\partial}{\partial P_e}\Upsilon_e(P_e^-) \end{bmatrix} \dot{q_s}^- \tag{24}$$

From (21) seven equations can be derived, that is $(N+2)$. From the assumption that the configuration of the robot does not change during the impact, we have that $q_e^- = q_e^+$, and thus the problem has a total of 9 unknowns, that is $(N+4)$. They are the generalized velocities after the impact $\dot{q}_e^+$, in addition to the unknown forces. Using the principle of virtual work, these can be represented as

$$F_{ext} = \frac{\partial}{\partial q_e}(P_e)\mathcal{F}_2, \tag{25}$$

where $F_2 = \left[F_2^T, F_2^N\right]$ is the vector of forces acting at the end of the swing leg[15]. Thus, $F_2^T$ and $F_2^N$ are the two remaining unknowns. As there are two more unknowns than there are equations, additional equations must be added. Using the no-slip requirement added in chapter 2.2 these are found as

$$\frac{\partial}{\partial q_e}(P_e)\dot{q}_e^+ = 0 \tag{26}$$

which states that as the swing leg hits the ground, it will not slip from its position or rebound up again from the floor, a requirement that is necessary as the swing leg will in turn become the new stance leg.
The previous equations can be combined as

$$\begin{bmatrix} \mathcal{M}_e(q_e^-) & -\frac{\partial}{\partial q_e}(P_e) \\ \frac{\partial}{\partial q_e}(P_e) & 0_{2x2} \end{bmatrix} \begin{bmatrix} \dot{q_e}^+ \\ \mathcal{F}_2 \end{bmatrix} = \begin{bmatrix} \mathcal{M}_e(q_e^-)\dot{q}_e^- \\ 0_{2x1} \end{bmatrix} \tag{27}$$

and solving this results in [15].

$$\begin{bmatrix} \dot{q}_e^+ \\ \mathcal{F}_2 \end{bmatrix} = \begin{bmatrix} \Delta_{\dot{q}_e^-}(q_s^-) \\ \Delta_{\mathbb{F}_2}(q_s^-) \end{bmatrix} \dot{q}_s^- \tag{28}$$

where $\Delta_{\dot{q}_e^-}(q_s^-)$ and $\Delta_{\mathbb{F}_2}(q_s^-)$ contains the the update laws of the impact for the generalized velocities and impact forces respectively. They are given as

$$\Delta_{\mathcal{F}_2}(q_s^-) = -\left(\frac{\partial}{\partial q_e}(P_e)\mathcal{M}_e^{-1}\frac{\partial}{\partial q_e}(P_e)'\right)^{-1}\frac{\partial}{\partial q_e}(P_e)\begin{bmatrix} I_{NxN} \\ \frac{\partial}{\partial q_s}\Upsilon_e \end{bmatrix} \tag{29}$$

and

$$\Delta_{q_e^-}(q_s^-) = \mathcal{M}_e^{-1}\frac{\partial}{\partial q_e}(P_e)'\Delta_{\mathbb{F}_2}(q_s^-) + \begin{bmatrix} I_{NxN} \\ \frac{\partial}{\partial q_s}\Upsilon_e \end{bmatrix} \tag{30}$$

Thus, to reinitialize the SemiQuad, the first $N$ ($N = 5$) variables of $\dot{q}_e^+$ are used. Now as mentioned, this will be proceeded immediately by the switch of variables to begin the next phase of Single Support as will be explained in the next chapter. The impact model was calculated together with the mathematical model and it is also given in Appendix C.

# 4 Virtual Holonomic Constraints & The Hybrid System Model

With the mathematical model implemented and the dynamics of the system found, the work to discover functioning gaits of the SemiQuad can begin. The problem at hand is that of describing the walking pattern of the SemiQuad as periodic orbits traced out in the state space of our robot model[15]. However, as stated in section 3.2, our system on state space form will be of dimension 10, which makes the creation of a periodic motion computationally complex. Further, since the system exhibits one degree of underactuation, the passive degree of motion, in our case the ankle, must be controlled indirectly through the actuated degrees of freedom[15], thus complicating the movement. Even further, the walking pattern of the SemiQuad will contain instantaneous impacts with the floor as calculated in section 3.3. The solution in state space will therefore exhibit discontinuous leaps in the velocities of the generalized coordinates which means the dynamic model must be extended to a hybrid model. All these requirements must be handled to achieve functioning periodic gaits and they will therefore be treated in this chapter.

## 4.1 Virtual Holonomic Constraints

The first difficulty described above was the problem of discovering functional gaits when the dynamic model is 10-dimensional. This is indeed a difficult task, as the computational complexity becomes quite high. There are however methods which can simplify the system model while still being able to find possible solutions. One idea is to reduce the solution space of the system to something that will let us easier find functioning gaits by creating restraints between the generalized coordinates. As the SemiQuad is pre-made, and since altering the physical configuration of the robot is unwanted, mechanical constraints will not be implemented. To achieve the reduction required we will instead use the Virtual Holonomic Constraint Approach.
The core idea of the VHC approach is simply as the name suggests to add virtual constraints to the dynamic system. The aim is to enforce a convergence to a particular geometric curve in the state space by means of control action[14]. It can further be shown that the number of differential equations of an Euler-Lagrange system gets reduced by introducing a *Holonomic Constraint*, which is a geometric restriction of the generalized coordinates to a particular curve in the configuration space[14]. The virtual constraint is then implemented by proper feedback control, instead of having to be created mechanically. The advantage of this is naturally that the virtual constraints can be changed on the fly by simply altering the simulation configuration. Changing a mechanical constraint on the other hand, means specifically changing the physical configuration of the robot, as in moving or adding parts, which is laborious and could be error prone. A negative side of VHC approach is that since it enforces the constraints via the actuators of the robot, it must necessarily spend energy to achieve the feedback control, which is unwanted if a mechanical constraint, which one could expect to not be

removed in due time, could have done the work passively. The latter is however not a concern here.

### 4.1.1 Creating the Constraints

Using the same method as in[14][8] the geometric relations between the generalized coordinates are introduces as follows

$$q_1 = \phi_1(\theta),\ q_2 = \phi_2(\theta),\ \dots\ q_5 = \phi_5(\theta) \quad \theta = \theta_\star(t),\ t \in [0, T] \tag{31}$$

all generalized coordinates are thus functions of a single scalar parameter $\theta \in [\theta_s, \theta_e]$, which can be seen as a trajectory generator for parametrising the time evolution. There are a myriad of ways that $\theta$ can be chosen that will lead to possible parametrizations, but for this system it has been decided to choose one the the generalized coordinates to represent $\theta$, namely the passive ankle angle $q_1$. All variables can then be rewritten as

$$q = \begin{bmatrix} \theta \\ \phi_2(\theta) \\ \phi_3(\theta) \\ \phi_4(\theta) \\ \phi_5(\theta) \end{bmatrix},\ \dot{q} = \begin{bmatrix} \dot{\theta} \\ \phi_2^{'}(\theta)\dot{\theta} \\ \phi_3^{'}(\theta)\dot{\theta} \\ \phi_4^{'}(\theta)\dot{\theta} \\ \phi_5^{'}(\theta)\dot{\theta} \end{bmatrix},\ \ddot{q} = \begin{bmatrix} \ddot{\theta} \\ \phi_2^{''}(\theta)\dot{\theta}^2 + \phi_2^{'}(\theta)\ddot{\theta} \\ \phi_3^{''}(\theta)\dot{\theta}^2 + \phi_3^{'}(\theta)\ddot{\theta} \\ \phi_4^{''}(\theta)\dot{\theta}^2 + \phi_4^{'}(\theta)\ddot{\theta} \\ \phi_5^{''}(\theta)\dot{\theta}^2 + \phi_5^{'}(\theta)\ddot{\theta} \end{bmatrix} \tag{32}$$

this choice is arbitrary, but a valid one that will function well for the system. To clarify the notion of virtual holonomic constraints further, a draft from Definition 3.1 from [14] is added:
*The restriction* $q = \Phi(\theta)$ is known as:

- *holonomic (geometric) constraint if it represents a restriction on generalized coordinates physically imposed on the system;*
- *virtual holonomic (geometric) constraint if the relation is preserved by some control action along solutions of the closed-loop system, provided that initial conditions are chosen to satisfy the constraint:* $q(0) = \Phi(\theta(0))$ *and* $\dot{q(0)} = \Phi^{'}(\theta(0))\dot{\theta(0)}$

*Holonomic constraints represents a restriction in the configuration space of the mechanical system to one specific curve.*

With the representation as in (32), the system model is then reduced to a two-dimensional model, given by $[\theta, \dot{\theta}]$, and a valid solution is then given by a periodic orbit in the phase plane of $[\theta, \dot{\theta}]$.
There is still the issue of underactuation to attend to however, and usually the motion planning for underactuated systems is much more complicated compared to fully actuated ones[14]. It can be shown however, that with an underactuation of 1 (or more), the dynamics of the system can be simplified and suitably described by the reduced dynamics of the system, which will be shown in the next section.

### 4.1.2 Reduced Dynamics

Since the SemiQuad has one passive joint to four actuated joints it has an underactuation degree of 1. The motion of the passive ankle must as previously described be indirectly controlled by the actuated joints, by indirectly shifting the center of mass to produce a torque on the ankle. To solve this, the reduced dynamic system is introduced. By Lemma 3.1 from [14] it is shown that for a degree of underactuation (n-m) where n is the number of generalized coordinates and m is independent control inputs (motors), the system can be described fully by solving the (m-n) second order differential equations given as

$$\alpha_i(\theta)\ddot{\theta} + \beta_i(\theta)\dot{\theta}^2 + \gamma_i(\theta) = 0 \tag{33}$$

as our system has an underactuation of one, this is simply to solve the previously defined zero dynamics (i.e the equation from the dynamic model that equated to zero). It is given as

$$\alpha_1(\theta)\ddot{\theta} + \beta_1(\theta)\dot{\theta}^2 + \gamma_1(\theta) = 0 \tag{34}$$

which is found by inserting the the parametrizations of the generalized coordinates from (32) into the zero dynamics equation (18.1). This is summarized in the following proposition[8]:

*Proposition 2: Suppose there exists a control law $u^*$ for the Lagrangian Equations that makes the relations (31) invariant along solutions of the closed loop system. Then, $\theta$ is a solution of the system (33) where $\alpha_i(\theta)$, $\beta_i(\theta)$ and $\gamma_i(\theta)$ are scalar variables.*

Thus (34), which shall be defined as the reduced dynamics of the system, define feasible motions of the system with perfect synchronization of the generalized coordinates given by the virtual holonomic constraints[14]. The entire motion of the SemiQuad is then available by simply solving one second-order equation of motion, reducing the complexity of the problem immensely. The additional equations of (18) still play an important part naturally, and just like for the zero dynamics equation, the parametrizations can be inserted as follows

$$\begin{aligned}
\alpha_2(\theta)\ddot{\theta} + \beta_2(\theta)\dot{\theta}^2 + \gamma_2(\theta) &= \tau_1 \\
\alpha_3(\theta)\ddot{\theta} + \beta_3(\theta)\dot{\theta}^2 + \gamma_3(\theta) &= \tau_2 \\
\alpha_4(\theta)\ddot{\theta} + \beta_4(\theta)\dot{\theta}^2 + \gamma_4(\theta) &= \tau_3 \\
\alpha_5(\theta)\ddot{\theta} + \beta_5(\theta)\dot{\theta}^2 + \gamma_5(\theta) &= \tau_4
\end{aligned} \tag{35}$$

after a proposed motion is found by solving the reduced dynamics, the torques $\tau_1 - \tau_4$ can be calculated by inserting the values of $\theta$, $\dot{\theta}$ and $\ddot{\theta}$ directly into (35). This will be important for implementing the simulation for several reasons. First of all as the actuators are limited in power, and a solution must obviously not have torque values exceeding this limit. Further, as will be explained in chapter 5, the goal in the end is to not just find functioning gaits, but also to find gaits that exhibit beneficial characteristics such as low energy usage. Trying to minimize the torque used for completing a gait is naturally one such way of ensuring this. Equation (34) together with (35) are typically referred to as the $\alpha$-$\beta$-$\gamma$-equations.

### 4.1.3 Existence of a Solution

From proposition 2 stated in the previous section we know that as long as a control law keeps the virtual constraints active, then $\theta$ will be a solution to (33). We next look at the integral of the reduced dynamics given by [11].

*Theorem 1: Given an initial condition* $[q(0), \dot{q(0)}]$*, if the solution*

$$\Big[q(t), \dot{q(t)}\Big] = \Big[q(t, q(0)), \dot{q}(t, \dot{q(0)})\Big] \tag{36}$$

*of system*

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0 \tag{37}$$

*exists for these initial conditions, then the function*

$$\begin{aligned} I_0(q, \dot{q}, q(0), \dot{q(0)}) = \dot{q}^2 - exp\Big\{ -2\int_{q(0)}^{q} \tfrac{\beta_1(\tau)}{\alpha(\tau)} d\tau \Big\} \dot{q(0)}^2 + \\ + exp\Big\{ -2\int_{q(0)}^{q} \tfrac{\beta_1(\tau)}{\alpha(\tau)} d\tau \Big\} \int_{q(0)}^{q} exp\Big\{ 2\int_{q(0)}^{s} \tfrac{\beta_1(\tau)}{\alpha(\tau)} d\tau \Big\} \tfrac{2\gamma(s)}{\alpha(s)} ds \end{aligned} \tag{38}$$

*preserves its zero value along this solution. The last property holds irrespective of the boundedness of* $[q(t), \dot{q}(t)]$*.*

thus, provided the chosen initial conditions give a solution, the system will always be integrable. That the function keeps its zero value is then sign that the constraints (32) are being respected, and that they therefore are valid for the solution.

## 4.2 The Hybrid Dynamics

As the walking gait of the SemiQuad exhibits instantaneous impacts with the ground, the dynamic model needs to be extended. The update laws for the impact itself was established in chapter 3.3, and the task is thus to combine the swing phase model with that of the impact model to form a system with impulse effects[15].

### 4.2.1 The Hybrid Model

An autonomous system with impulse effects consists of an ordinary differential equation defined on a state space $\mathcal{X}$, a hyper surface $\mathcal{S}$ at which solutions of the differential equations undergo a discrete transition that is modelled as an instantaneous reinitialization of the differential equation and finally a rule $\Delta : \mathcal{S} \to \mathcal{X}$ that specifies the new initial conditions as a function of the point at which the solution impacts $\mathcal{S}$[15]. For the SemiQuad system the differential equation is as defined the dynamic model (18), the state space is the generalized coordinates $\mathcal{X} = [q, \dot{q}]$, the hyper surface is the configurations for which the swing leg makes an impact with the ground which can be defined as the

*impact surface*, while $\Delta$ is the update law for the impact calculated in chapter 3.3, $\Delta_{q_s}$. The dynamic model and the impact model can then be combined as

$$\sum : \begin{cases} \dot{x} = & f(x) + g(x)u & x^- \notin S, \\ x^+ = & \Delta(x^-) & x^- \in S \end{cases} \tag{39}$$

where

$$\mathcal{S} = \left\{ (q, \dot{q} | P_e^v(q) = 0 \right\} \tag{40}$$

However, this hybrid model can be simplified further after having implemented the virtual holonomic constraints in the previous section. There we reduced the state space to that of $\mathcal{Z} = [\theta, \dot{\theta}]$, the dynamic model is fully explained by (33) and the new impact surface can be seen as the intersection of the entire impact surface $\mathcal{S}$ with the new state space $\mathcal{Z}$, that is $\mathcal{S} \cap \mathcal{Z}$[5]. The new model can thus be stated as:

$$\sum : \begin{cases} \dot{z} = & f_{zero}(z) & z^- \notin \mathcal{S} \cap \mathcal{Z}, \\ z^+ = & \Delta_{zero}(z^-) & z^- \in \mathcal{S} \cap \mathcal{Z} \end{cases} \tag{41}$$

with

$$\mathcal{S} \cap \mathcal{Z} = \left\{ (\theta, \dot{\theta} | p_{sw}^v(\theta) = 0 \right\} \tag{42}$$

where $p_{sw}^v(\theta)$ is defined as the vertical position of the swing leg with respect to $\theta$ which can be derived from (19) by inserting for the parametrizations defined in the last section. This model is known as the Hybrid Zero Dynamics[5]. The dynamics are thus described solely by the evolution of the single support phase until the state attains the set $\mathcal{S} \cap \mathcal{Z}$[5]. At this instance, the impact will result in an instantaneous switch of velocity components, and then the single support phase begins anew with the previous stance leg acting as the swing leg until the state again attains the set $\mathcal{S} \cap \mathcal{Z}$.

### 4.2.2 Other Discontinuities

The impact is not the only discontinuity in the gait however. As stated in chapter 2.2 the generalized coordinates will also be switched after impact so that the ankle angle, now $\theta$, can still describe the relation between the stance leg and the ground. Swapping the value of $\theta$ naturally introduces a discontinuous shift in our phase plot, and this change must therefore also be included in the hybrid model. Labelling the update law as $\mathcal{F}_{swap}$, the hybrid model can be augmented as follows.

$$z^+ = \mathcal{F}_{swap}(\Delta_{zero}(z^-)) \quad z^- \in \mathcal{S} \cap \mathcal{Z} \tag{43}$$

which states that immediately after the new velocities are calculated from the impact model, the coordinate shift takes place. $\mathcal{F}_{swap}$ will have a different form depending on which shift takes place.

The full model for the SemiQuad is still not fully described however. When looking into possible ways for the SemiQuad to move when the ankle is passive, it becomes clear that a motion must begin by increasing the value of the angle $q_1 = \theta$ so that the front leg

may be lifted into the air, see fig(2). To place the front leg at a point further from the starting point, the value of $\theta$ must naturally decrease. Thus, during the movement from the starting configuration (or from an impact) to the next impact phase, $\theta$ will exhibit the same value for different points in time. This again breaks the condition which states that $\theta$ should uniquely describe the evolution of the movement. This is not due to $q_1 = \theta$ being a bad choice as a reference value, as this phenomena will most likely be present in the other joints as well. This can be solved easily however, as the movement phase can be further divided into two parts, transforming the system from a two-part system to a four-part system. As $\theta$ must obviously stop before moving in the opposite direction, a natural choice is to introduce another switching criteria when $\dot{\theta} = 0$. The Hybrid model is thus further expanded.

$$\sum : \begin{cases} z = & f_{zero}(z) & \dot{\theta} \neq 0 \\ z^+ = & \mathcal{F}_{shift}(z) & \dot{\theta} = 0 \\ z = & f_{zero}(z) & z^- \notin \mathcal{S} \cap \mathcal{Z} \\ z^+ = & \mathcal{F}_{swap}(\Delta_{zero}(z^-)) & z^- \in \mathcal{S} \cap \mathcal{Z} \end{cases} \tag{44}$$

where $\mathcal{F}_{shift}$ is the update law from the section before $\dot{\theta}$ reaches zero to the start of the next section. The Hybrid Model (44) can also be seen as a chronological guide to the movement of the SemiQuad. The first movement is that of lifting the leg, and thus any impact in this phase can be considered impossible. Thus, the model solely focuses on detecting the moment $\dot{\theta}$ becomes negative as this signifies that the first switch must occur. The next session however will end upon the moment the swing leg makes an impact with the ground, and thus the model focuses solely on detecting this. As one reaches the end of (44), the SemiQuad will have completed half the gait, and the process begins anew from the top, with the new configuration of the generalized coordinates.

### 4.2.3 Creating the Update Laws

With the system model fully explained, the dimension of the problem reduced as in 4.1 and the hybrid relations and discontinuities explained, the work on finding a complete set of update laws for the SemiQuad can begin. As briefly mentioned in 4.2.2, the gait will consist of four distinct phases, two phases of lifting the leg and two phases of placing the leg down. Yet undefined from these sections are the numerical values of $\mathcal{F}_{shift}(z)$ and $\mathcal{F}_{swap}(z)$. Now these could be defined as an update of $\theta$ alone and then (32) could be used to calculate the other joint values. For the sake of readability however, these update laws will describe the change for all generalized coordinates. The update laws thus have the following form.

$$\mathcal{F}_{pos}(q^-) = q^+, \quad \mathcal{F}_{vel}(\dot{q}^-) = \dot{q}^+ \tag{45}$$

where the update laws are either of type *Swap*, or type *Shift* as specified previously. Likewise $\mathcal{F}_{pos}$ defines the update of the positions of the generalized coordinates while $\mathcal{F}_{vel}$ specifies the update in velocities. We will first look at the re-initiation after the leg has been raised up to its highest point described by $\mathcal{F}_{shift}$. As the gait consists of four

phases, the update laws can be defined by the transition from one phase to the other. $\mathcal{F}_{shift}$ then constitutes the shift from phase 1 to 2 and from phase 3 to 4 and can be rewritten as $\mathcal{F}^{1\_2}$ and $\mathcal{F}^{3\_4}$. As each shift is considered instantaneous these update are trivially defined as

$$\mathcal{F}^{1\_2}_{pos} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} q^- \quad \mathcal{F}^{1\_2}_{vel} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \dot{q}^- \tag{46}$$

or simply put just the $5x5$ identity matrix, which also holds true for $\mathcal{F}^{3\_4}$. Naturally when the robot is not subjected to any influence from any interaction (with environment etc) the configuration of the start of one step must be exactly that of the ending configuration for the previous step.
$\mathcal{F}_{swap}$ can be calculated in the same way, constituting the remaining step transitions, $\mathcal{F}^{2\_3}$ and $\mathcal{F}^{4\_1}$. Using the kinematic model from section 3.1, and especially looking at the change from figure (3) to figure (4) and vice-versa, these update laws can be computed. With respect to the update of positions, a way of describing a shift from the new coordinates to the old coordinates immediately becomes clear for coordinates $q_2 - q_5$ of the new coordinates. As an example one can examine the new coordinate $q_2^+$ and compare it to old coordinate $q_5^-$. It is easy to see that the relation between the two can be stated as $q_2^+ = -q_5^-$. A quick examination of the same joint reveals that this also holds true for the velocities, that is $\dot{q}_2^+ = -\dot{q}_5^-$. This holds true for all new generalized coordinates, excluding $q_1^+$, for either swap. As the new stance leg will be at the very opposite "side" of the robot from the previous stance leg, the new ankle angle $q_1^+$ can be found by summing together all the old generalized coordinates, then adding or subtracting $\pi$[16]. The new generalized velocity $\dot{q}_1^+$ can be found by adding all the old generalized velocities. The final result is then

$$\mathcal{F}^{2\_3}_{pos} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} q^- + \begin{bmatrix} \pi \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathcal{F}^{2\_3}_{vel} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \dot{q}^- \tag{47}$$

and

$$\mathcal{F}^{4\_1}_{pos} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} q^- + \begin{bmatrix} -\pi \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathcal{F}^{4\_1}_{vel} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix} \dot{q}^- \tag{48}$$

this concludes the update laws. In fig(5), the full evolution of a gait is shown with the

update laws included.

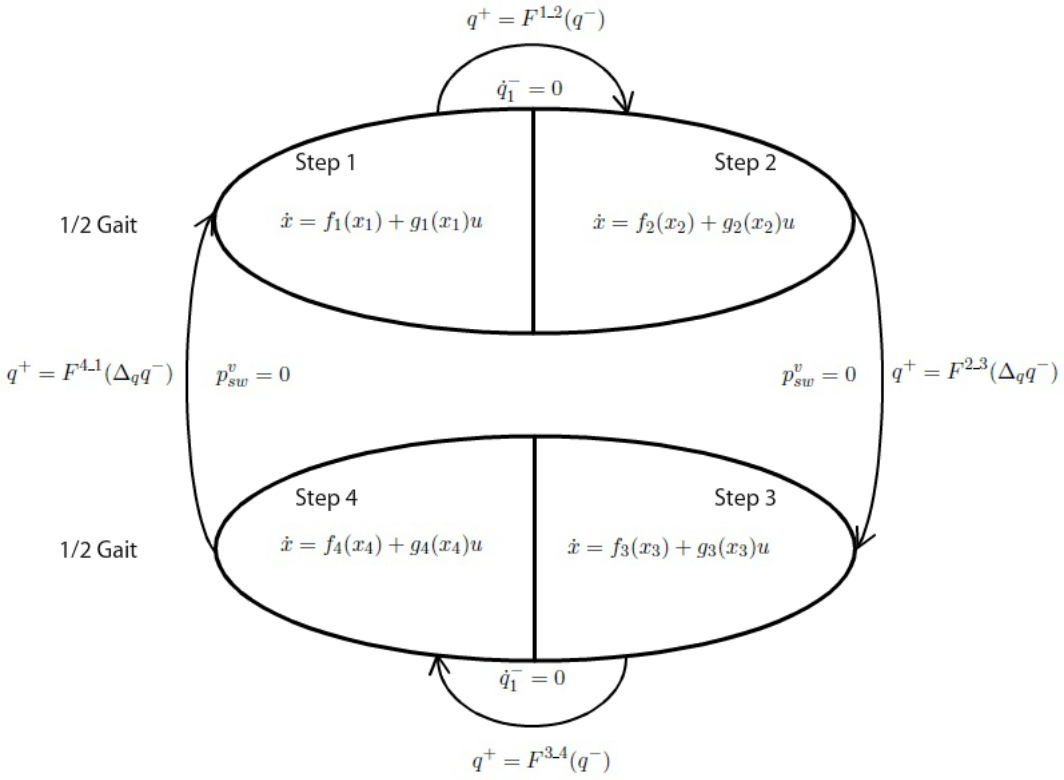



Figure 5: The full evolution of the Gait

# 5 Designing the Gait

## 5.1 Choosing the Parametrization

As have been established in chapter 4, the entire motion of the joints have been synchronized to the motion of the ankle-angle, which again is calculated from the reduced dynamics (34). To solve this equation however, the parametrization of the joints must be inserted, and the problem is thus reduced to that of selecting suitable coefficients. The parametrizations of the actuated joints must be chosen such that the solutions of the reduced dynamics constitutes a movement of the robot which in turn constitutes a valid gait. We can begin by rewriting equation (31) as

$$q_1 = \theta,\ q_2 = \phi_2(\theta, \mathcal{P}_1),\ q_3 = \phi_3(\theta, \mathcal{P}_2),\ q_4 = \phi_4(\theta, \mathcal{P}_3),\ q_5 = \phi_5(\theta, \mathcal{P}_4) \tag{49}$$

where the choice for $q_1$ representing the evolution of $\theta$ is directly inserted, in addition to the parameter matrix $\mathcal{P}$ given as

$$\mathcal{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ p_{31} & p_{32} & \cdots & p_{3n} \\ p_{41} & p_{42} & \cdots & p_{4n} \end{bmatrix} \tag{50}$$

where $\mathcal{P}_m$ is row $m$ of $\mathcal{P}$, which contains the parameters for joint $m+1$. Each joint angle is then represented as a polynomial of order $n-1$ given as

$$q_{m+1} = p_{m1}\theta^{n-1} + p_{m2}\theta^{n-2} + \cdots + p_{mn}, \quad m = 1, 2, 3, 4 \tag{51}$$

thus the parametrizations can be described and the reduced dynamics can be created prior to the exact values being known. Another positive effect of describing the parametrizations as polynomials is that it introduces efficient ways to calculate virtual constraints based on physical criteria. Here the coefficients will be calculated using Bèzier polynomials. One reason for this is that they make it very easy to satisfy the invariance condition, so that the hybrid zero dynamics are guaranteed to exist[5]. The Bèzier polynomial over an arbitrary interval can be states as

$$\mathcal{P}_{[\theta_0,\theta_e]}(\theta) = \sum_{i=0}^{n} \binom{n}{i} \left(\frac{\theta_e - \theta}{\theta_e - \theta_0}\right)^{n-1} \left(\frac{\theta - \theta_0}{\theta_e - \theta_0}\right)^{i} P_i \tag{52}$$

where

$$\binom{n}{i} = \frac{n!}{i!(n-i)!} \tag{53}$$

$\theta_e$ is the ending configuration of $\theta$ which is the desired value at the end of the motion, $\theta_0$ is the initial condition of $\theta$ before the motion starts, while $P_0, P_1, \ldots, P_{n-1}$ are concrete positional values of the the joint angles.

The use of Bèzier curves thus allows the parametrizations to be designed based on the desired values of $\theta$, in addition to providing suitable initial and ending configurations

of the joint values. This works since it is guaranteed that the curves moves from the position $P_0$ to the position $P_{n-1}$. The intermediary points aren't necessarily visited, but they act as guidance flags controlling the curvature and creating smooth motion. The number of intermediary points decides the order of the parametrizations. The more intermediary points, the more freedom of movement will the curve exhibit. On the one hand, it is preferable to increase the order of parametrization to allow for more freedom of choice. On the other hand this increases the number of parameters and thus the computational complexity. In this paper parametrizations have been developed for systems of the 2nd, 3rd, 4th and 5th order.

### 5.2 Optimization Routines

To begin the optimization process, we must first define how a legal gait can be defined as a numerical value. As mentioned in section 2.3, it is desired to configure the walking pattern of the SemiQuad as a Limit Cycle. This is achieved by configuring the movement of the SemiQuad as exact repetitions of a closed trajectory[2][5], which means the movement is periodic. Now as the SemiQuad is a hybrid system, the whole movement will be calculated by solving the reduced dynamics for four distinct starting configurations (step 1 - step 4) while applying the update laws and impact model as specified between each step. A periodic gait in this regard is then achieved if the robot has the exact same configuration after the impact at step 4 as it had before starting at step 1. The same configuration entails having the exact same values for the generalized coordinates and the generalized velocities. If this is the case, then naturally the solution of the next four steps will yield the exact same movement as the first four steps and the movement will repeat itself. To ensure this a cost function can be created as

$$\mathcal{J}_1(\theta, \dot{\theta}) = \left\| \theta_1^- - \mathcal{F}_{pos}^{4\_1}(\theta_4^+) \right\|_2 \tag{54}$$

where $\theta_1^-$ is the starting value for $\theta$ at step 1 and $\theta_4^+$ is the final value for $\theta$ at the end of step 4. By the nature of the virtual holonomic constraints, ensuring that $\theta$ will have the same value after a successful gait is the same as ensuring that all joints have the same value after a successful gait. Further to ensure that the velocities will be the same we include

$$\mathcal{J}_2(\theta, \dot{\theta}) = \left\| \dot{\theta}_1^- - \mathcal{F}_{vel}^{4\_1}(\Delta_q \dot{\theta}_4^+) \right\|_2 \tag{55}$$

in total this becomes

$$\mathcal{J}(\theta, \dot{\theta}) = \delta \left\| \theta_1^- - \mathcal{F}_{pos}^{4\_1}(\theta_4^+) \right\|_2 + \gamma \left\| \dot{\theta}_1^- - \mathcal{F}_{vel}^{4\_1}(\Delta_q \dot{\theta}_4^+) \right\|_2 \tag{56}$$

where $\delta$ and $\gamma$ are weighting constants that can assign a specific preference to either of the two expressions. With this cost function, an optimization routine can be used with the goal being to minimize $\mathcal{J}$, which again will force the ending configuration to be equal to the initial configuration.
Ensuring periodicity is not the only requirement for having a successful gait however,

there are naturally also several physical constraints that must be taken into consideration. These must be accounted for when solving the reduced dynamics, so that the SemiQuad will be capable to perform the motion. A short list of the most dominant restraints are added below

- Limitations in the actuators
- Staying above the floor at all times
- The gait involves a forward movement, the legs being in front of their earlier position.
- The ankle-ankle stays above or below a certain threshold with the ground.

the first requirement is perhaps the most dominant one. When the initial conditions for the SemiQuad makes it impossible to move to the desired configuration, the actuators will often exhibit very high values which they naturally will not (physically) be able to achieve. For this system this limit is given by

$$\tau_{max} \leq 40 \tag{57}$$

for each of the four actuators.
The second requirement states that the robot can't move through the floor, which might seem as a trivial constraint, but which still needs an error warning when it does occur. The stance leg is considered fixed, thus this is mainly a concern for the end of the swing leg. The most critical moments are at the very start of the step when the swing leg recently was the stance leg, and at the very end of the step when the swing leg is placed on the ground. A requirement can thus be made that

$$y_{sw} \geq -\epsilon \tag{58}$$

which states that the horizontal position of the tip of the swing leg must be larger than an arbitrarily small value $\epsilon$.
The ankle-ankle restraint is taken care of by adding the restraint that

$$\theta_{min} \leq \theta \leq \theta_{max} \tag{59}$$

where $\theta_{min}$ is a value appropriately larger than 0 and $\theta_{max}$ is a value appropriately smaller than $\pi$.
The last constraint, which is that a gait must exhibit forward movement, does not need any consideration as the forward movement will be a part of the design process for the gait (it is chosen from the start), as will be explained in the next chapter.

## 5.3 Optimization Criteria

In addition to finding a gait that is periodic and which does not violate any physical constraints, it is usually desired to optimize the gait further with respect to certain

functional requirements. This section will do precisely that, where specific cost functions will be created to be included in the total optimization routine presented in (56). These functional requirements can be divided in two sub categories. Energy optimization and Speed Optimization.

### 5.3.1 Energy Usage

Even though the SemiQuad may find a configuration that allows it to take a step of a certain length, that configuration will most certainly not be the only one that exists, nor is it necessarily as energy efficient as other options. What is meant by energy efficiency is to complete a specified movement using as little energy as possible. For the SemiQuad the energy used is due to the work done by the four actuators.
The method proposed is that of specific cost of transport[2]. It is defined as

$$c_t = \frac{\mathcal{E}}{md} \tag{60}$$

where $m$ is the mass of the SemiQuad, $d$ is the length of the step and $\mathcal{E}$ is the total energy used. The cost function can then be updated with the following expression

$$\mathcal{J}_3 = \frac{1}{md} \int_{t=0}^{T} \mathcal{E}(t) dt \tag{61}$$

in which the integral sums the energy usage up over the entire span of time.

### 5.3.2 Optimization of Velocity

Another relevant criteria is that of maximizing the velocity of the SemiQuad. For our system this primarily means either maximizing forward velocity $v$ or minimizing the total time used $T_{tot}$. To then be able to compare the results with that of other robots the Froude number can be used, which is given as

$$F_r = \frac{v}{\sqrt{gl}} \tag{62}$$

in which $v$ is the forward velocity, $g$ is the gravity constant, while $l$ is leg length.
Naturally, it is also possible to combine elements of low energy usage while also trying to maximize velocity, where both of the previous mentioned cost functions can be used together. To prioritize one over the other, weight parameters can once again be introduced.

# 6 Simulating the 2nd-order System

## 6.1 Selecting Optimization variables

The final part of the creation of a functioning gait is to use all tools that has been discovered so far together to achieve the best possible solutions. To find an optimal solution, various cost functions were introduced constituting both necessary criteria for a gait to function, and desirable criteria for a gait to exhibit. Yet if a gait is unsatisfactory, or if an attempted solution does not work, the optimization routine must edit the optimization variables. These variables are first of all the desired values of the joint variables as introduced in the previous chapter. Through the use of Bèzier polynomials, they constitute the parametrization of the joint parameters which again yields the reduced dynamics. They are stated as

$$P = \begin{bmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{2n} \\ P_{31} & P_{32} & \dots & P_{3n} \\ P_{41} & P_{42} & \dots & P_{4n} \end{bmatrix} \tag{63}$$

in which $P_m$ constitutes row $m$ of $P$ which yields the desired evolution of joint $m+1$. $P_{m1}$ then constitutes the starting configuration, $P_{mn}$ the final configuration, with points $P_{m2}$ to $P_{m(n-1)}$ being the intermediary points defining the evolution between the two. To avoid confusion please note that $P \neq \mathcal{P}$. As the system is divided in four steps, and the reduced dynamics must be calculated four times, it is also necessary to define four such matrices $P^1 - P^4$ for any possible solutions. As will be made clear, not all entries in these matrices will be customizable.
In addition to the position matrices, the SemiQuad is designed to start the gait with an initial angular velocity. The optimal starting velocity is naturally unknown and must therefore enter into the optimization routine. Further, for the steps 1 and 3 of the gait, the decision to switch steps the moment $\dot{\theta} = 0$ makes deciding a specific ending value for the ankle angle $\theta_e$ difficult. Rigorously demanding a specific value is also not necessary and can in fact be seen as a constriction to the freedom of finding suitable solutions. Therefore, the ending values used in step 1 and 3, namely $\theta_e^1$ and $\theta_e^3$, will also enter into the optimization routine. As the reduced dynamics in step 1 and 3 will be calculated after their selection, they will most likely not be the real ending values, but rather act as variables that shape the evolution of the motion.
The simulation has been carried out with different orders of parametrizations. First the second order model will be explored.

### 6.1.1 2nd Order Optimization

When the parametrization is of 2nd order, both parameter matrix $\mathcal{P}$ and position matrix $P$ will be $4x3$ matrices. Each joint will then have only one intermediary point in the position matrix. The optimizable parameters are as mentioned the initial angular velocity of the ankle, $\dot{\theta}$, the ending values $\theta_e^1$ and $\theta_e^3$, and some columns of the four position

matrices. It is desired that the SemiQuad starts in a suitable configuration, thus the first column of $P_{c1}^1$ is a design choice. Given that the gait should be a periodic motion this further means that the last column of matrix four, $P_{c3}^4$, is decided indirectly from the choice of the starting configuration. In fact, this is true for all switching phases since the positions of the joints cannot change instantaneously. This can be stated as

$$P_{c1}^{n+1} = \mathcal{F}^{n\_n+1} q_n^-, \quad n = 1, 2, 3 \tag{64}$$

$$P_{c1}^1 = \mathcal{F}^{4\_1} P_{c4}^4 \tag{65}$$

Notice that this is true despite the optimizable variables $\theta_e^1$ and $\theta_e^3$ not representing the exact ending values, as the starting column of the position matrices will nevertheless be calculated from the true ending parameters of the previous step. Thus, it is $P_{c1}^n$ of the position matrices that will represent the true configuration of the SemiQuad at the moment between the steps.
The length of the step will also be treated as a design choice, thus the last column of matrix 2, $P_{c3}^2$, will be chosen as well, as this column signifies the configuration of the SemiQuad at the moment of impact. In theory, within a certain limit most step lengths should be possible to achieve. In this report however, the previously discovered step lengths from [16] have been implemented for further optimization.
It is also required that the full evolution of the joint positions should be $\mathcal{C}2$ continuous. This makes for smoother motion of the joints and thus less strain on the actuators. This is easily obtained using Bèzier curves as the individual segments can be pieced together. They are already $\mathcal{C}^0$ continuous since it is required that the ending position of one segment will be the starting position for the next segment. Further they are said to be $\mathcal{C}^2$ continuous if for two polynomials, say in this case $\mathcal{P}_1$ and $\mathcal{P}_2$ we have that

$$\mathcal{P}_1(\theta_e^1) = \mathcal{P}_2(\theta_0^2),\ \mathcal{P}_1^{'}(\theta_e^1) = \mathcal{P}_2^{'}(\theta_0^2),\ \mathcal{P}_1^{''}(\theta_e^1) = \mathcal{P}_2^{''}(\theta_0^2), \tag{66}$$

the calculation will not be carried out in detail here, but the result is nevertheless that the first intermediary point $P_{m2}$ will be a function of the other positions. Thus the second column from position matrices $P_2$, $P_3$ and $P_4$ , will not contain optimization variables.
In total the columns that are editable are

$$P_{c_2}^1,\ P_{c_3}^1,\ P_{c_3}^3 \tag{67}$$

for a total of $3x4 = 12$ parameters, bringing the total number of optimization variables to 15.

### 6.1.2 3rd Order Optimization

Increasing the order of the parametrization to the third order increases the size of the parametrization matrix $\mathcal{P}$ and the position matrix $P$ to $4x4$. As each position matrix now has an extra column, the evolution of each joint will have a second intermediary point, thus increasing the possible motions. These additional columns are also unconstrained, and the number of optimizable parameters increases to $15 + 4 * 4 = 31$, significantly higher than with the second order model.

### 6.1.3 Higher Order Parametrizations

When implementing higher order models, the change of increasing the order by one degree will constitute the same change as between the second and third order models. The position matrices will each have an extra column, and thus an extra intermediary position value, while the number of optimizable variables increases by 16. Here systems have been implemented for the fourth and fifth order, with the number of optimizable variables being equal to 47 and 63 respectively.

## 6.2 Simulation Results

Gaits where found for three distinct step lengths (short-medium-long) for each of the system models. They were first created for the second degree model, and then in turn the results were used to create models of higher and higher order. The energy optimization routine was used by default as it was shown to aid in the discovery of new gaits. This was due to the added cost function term which desired that torques should be kept as low as possible, which in turn ensured legal gaits could be found. This section will thus look into the functioning gaits found for the various model orders, and then attempt to optimize a number of these with respect to velocity. An extensive list of results found are available in Appendix A.

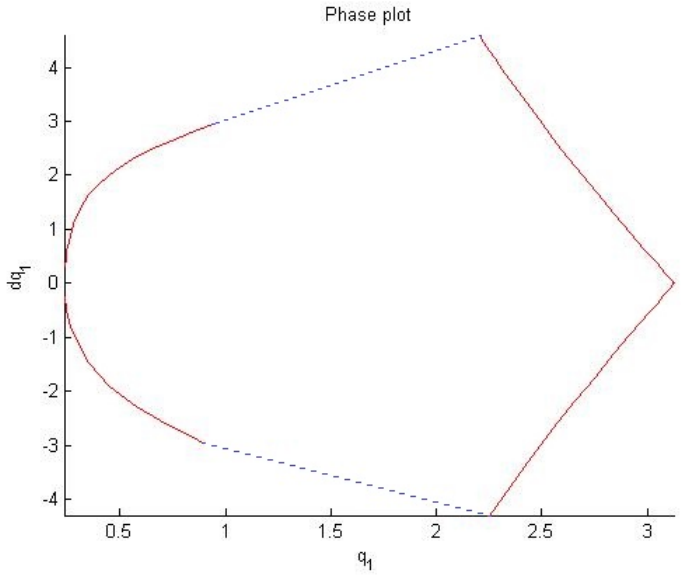



Figure 6: State Space evolution for the second order model for a short step length

### 6.2.1 Second Degree Model

As the second degree model was the first to be implemented, much time had to be used to ensure the optimization routine started in a configuration that wasn't too far away

from a possible solution. This consisted in finding parameters for the optimizable parts of position matrices $P_1$ and $P_3$, an initial angular velocity $\dot{\theta}$ in addition to ending values $\theta_e^1$ and $\theta_e^3$, which were in relative proximity to what a functioning gait would exhibit. Naturally this process involved some trial, but the parameters that in the end was given to the optimization routine ensured that a functioning gait could be found much quicker. The gaits found seemed to perform quite well, despite the lack of options for the motion of the joints. In figure (6) an evolution of $\theta$ and $\dot{\theta}$ can be seen for the gait for the short step length, with energy optimization implemented. The evolution can be seen to begin

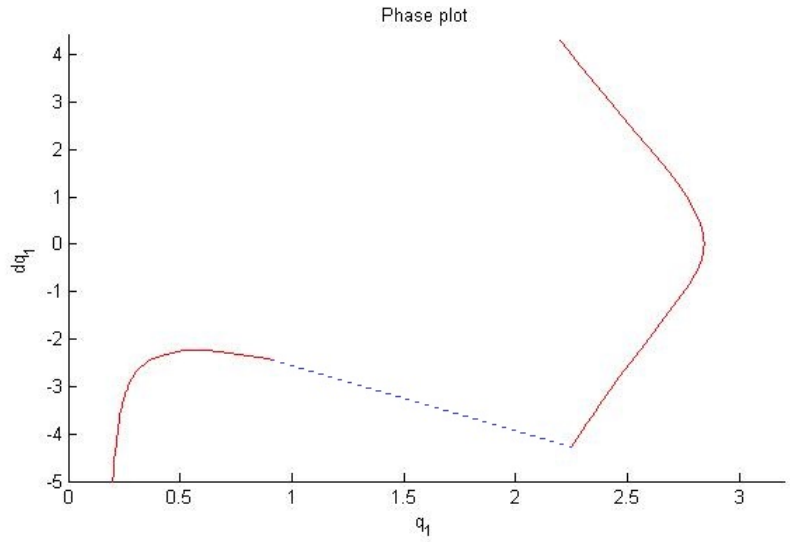



Figure 7: State Space evolution for the second order model which failed

in the top right corner and move clockwise. The dotted lines constitute the impacts in which the values of $\theta$ are changed as a result of the switch. To compare figure (6) up against something, figure (7) shows an evolution where the initial angular velocity was changed from the optimal value, which resulted in the SemiQuad failing to perform the gait. From the plot one can see that the SemiQuad is unable to recover after the first impact, leading the angular velocity to decrease rapidly.

### 6.2.2 Third Degree Model

With the second degree models already implemented, the task of finding functioning gaits for the third degree models became much easier. For each step length, the parameters from the second degree model was used as guidance for creating the new models, which led to new gaits being found quite quickly. The added freedom of motion was quite noticeable, though performance in terms of energy efficiency failed to surpass that of the second degree models initially. This led to more manual work having to be performed directly on the optimizable parameters, before restarting the optimization routine. Eventually new results were found that outperformed the previous second or-

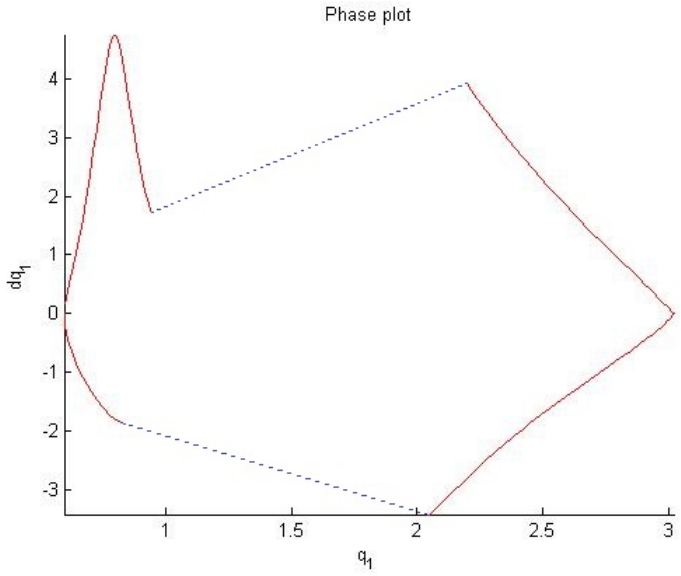



Figure 8: State Space evolution for the third order model for a medium step length

der solutions considerably. In figure (8) the energy optimized gait with a medium step length is shown.

### 6.2.3 Higher Order Models

This process repeated itself for the fourth and fifth order models, showing that it is possible to create functional polynomials with a relatively high order, by first initiating the problem using low order polynomials. The energy efficiency was found to be slightly below that of the third order model, but this is likely due to the fact that more effort was put into finding the third order gaits. In addition, it is also natural to assume that when the model is increased above a certain threshold, the performance cannot be expected to increase noticeably. The difference between the second and third order models was in comparison massive, despite the difference in optimization variables being the same. This was mainly due to the limitation of the second order model, which had no optimization variables in position matrices 2 and 4. The velocity optimization was also introduced, and the results were similar to that of energy optimization with the 3rd order optimzation outperforming the 2nd order optimization.
The complete results are available in Appendix A, where the cost function value for energy, $f$, as well as the total time of the gait, $T_{tot}$, is shown for each gait together with the optimal position matrices.

# 7 Implementing the Gait

The first part of this thesis has shown that the discovery of gaits can be simplified by the introduction of Virtual Holonomic Constraints, allowing the entire motion of the robot to be described using only a single parameter $\theta$. It has further been shown that the gait can be reproduced provided the configuration of the Semiquad after the gait is exactly the same as it was before the gait started. This allowed the creation of numerous gaits to be simulated using the 2-dimensional system spanned by $\theta$ and $\dot{\theta}$, while ensuring that the Semiquad ended in the suitable configuration. Thus, it was shown that these types of gaits are certainly possible, though it remains to be seen if they can be made functional in a practical sense.

## 7.1 Describing the Error

As the simulations were done using the 2-dimensional system, it is clear that the legality of the Virtual Holonomic Constraints was simply implied from the start. Further, as no errors were introduced, the gaits would by their creation always end up in the predefined suitable configurations. This ideal situation is not particularly realistic however as small errors will almost certainly be present. Indeed, the very concept of virtual Holonomic Constraints are based upon having available actuation to ensure the constraints are always kept true. The process of measuring and then counteracting deviations from the predesigned gait must then naturally be a part of any implementation. These deviations can be summed up as

$$\begin{aligned} &q_i \neq \Phi_i(\theta) && i = 2..5 \\ &\dot{q}_i \neq \dot{\Phi}_i(\theta) && i = 2..5 \\ &I\left(\theta(0)^*, \dot{\theta}(0)^*, \theta(t), \dot{\theta}(t)\right) \neq 0 \end{aligned} \tag{68}$$

where $I$ is taken from (38). If any of these are breached, the robot will no longer be following the predesigned trajectory and as a result the Semiquad might fail to complete the gait. This will be shown in the next section.

## 7.2 Simulating with Errors

Before the work of creating a robust implementation of the gaits can begin, it is interesting to study the exact effect errors may have on the gait. Given that the gaits rely on having a correct initial configuration, the introduction of errors will naturally make the gaits fail at some point. However for control purposes, it is interesting to note where such failures manifest themselves. Therefore, instead of the 2-dimensional simulations that was carried out in the previous chapter, the full 10-dimensional system will now be simulated given by $[q, \dot{q}] \in \mathbb{R}^{10}$. This further means the simulation will be run with the full dynamic system given by (18), and no longer solely be updated by the zero dynamics. The gait that will be explored is a solution of a third order parametrization, optimized both with respect to velocity and energy for the short step length.

### 7.2.1 Increasing Deviations of $\dot{\theta}(0)$

The first error scenario to be explored was deviations in the initial optimal velocity of $\dot{\theta}$. One of the main optimization criteria from the previous chapter was after all to ensure the angular velocity after a successful gait would be equal to the initial angular velocity. However, with errors in the system, and given that there is an impact with the ground right before a new gait cycle begins, the initial angular velocity might exhibit large discrepancies. Simulations were therefore carried out with increasing deviations in the initial configuration to mimic a scenario where the SemiQuad has completed a gait, but encountered errors along the way.
For the first simulation, no specific errors were introduced and the SemiQuad was started with the optimal initial coordinates, that is

$$\theta(0) = \theta(0)^*, q_i(0) = q_i(0)^*,\ \dot{\theta}(0) = \dot{\theta}(0)^*,\ \dot{q}_i(0) = \dot{q}_i(0)^*,\quad i = 2..5 \tag{69}$$

As a result, the SemiQuad was able to complete the gait correctly, showing that the 10-dimensional system matches the 2-dimensional system, provided the nominal trajectory is followed. This can be seen in fig(9), where the phase portraits of all simulations are shown. An error was then introduced by adding a small deviation $\delta$, so that $\dot{\theta}(0) = \dot{\theta}(0)^* + \delta$. The deviation was initially set to $\delta_1 = 0.02$, meaning a slight increase in initial angular velocity. Though the increase in velocity caused the motion of the robot to diverge from the nominal trajectory, the SemiQuad was still able to complete the gait. The final configuration however, which is the starting configuration for the next gait, had shifted even further from the optimal configuration, meaning the SemiQuad would most likely fail to complete a full gait sooner or later. The deviation was then increased

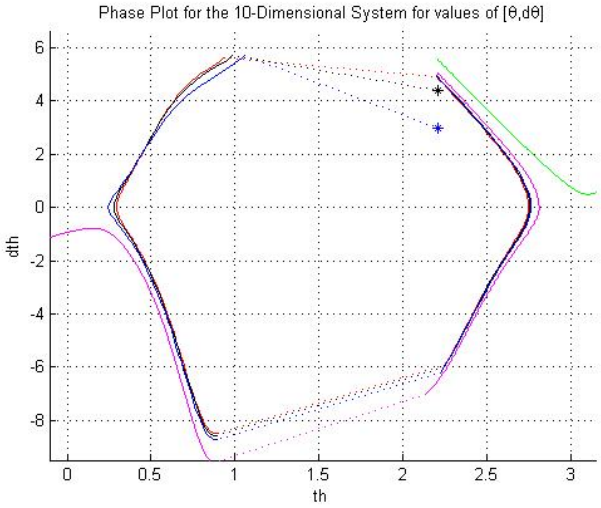



Figure 9: The Phase Portraits $\left[\theta, \dot{\theta}\right]$ for several simulations of the SemiQuad system. The whole lines are the evolution of the coordinates, while the stippled lines are the instantaneous impact phases.

even further to $\delta_2 = 0.05$. Though the SemiQuad was still able to complete the gait, the final configuration, as shown in fig (10), was now very far from the initial configuration. The robot is no longer in a symmetric position, and has in addition also taken a longer step than intended. As a matter of fact, for both $\delta_1$ and $\delta_2$, the SemiQuad was unable to complete a second gait, meaning the errors were sufficient to cause a failure.
As a final test, two large errors were introduced, $\delta_3 = 0.2$ and $\delta_4 = 0.7$. As expected, for errors of this magnitude the SemiQuad was unable to complete the gait. For $\delta_3 = 0.2$ the motion of the SemiQuad can be seen to deviate heavily in step 3, as instead of coming to a stop it instead tilts over. This is due to the large increase in initial angular velocity making the SemiQuad rise much higher than intended during the first step, resulting in a much larger impact, and consequently a too high initial angular velocity in step 3. For the error of $\delta_4 = 0.7$, the SemiQuad fails already in step 1, as the added initial angular velocity is of sufficient magnitude to make the robot to tip backwards instead of stopping.

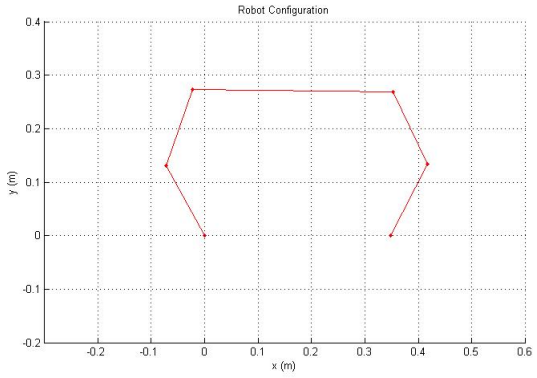



Figure 10: The Configuration of the SemiQuad after a gait with initial deviation $\dot{\theta}(0)^* + 0.05$.

### 7.2.2 Violating the Virtual Holonomic Constraints

The previous section demonstrated the loss of functionality the SemiQuad experiences when introduced to deviations of the passive ankle-angle angular velocity. This section will instead focus on the new degrees of freedom found in the 10-dimensional system which stems from breaching the virtual holonomic constraints. The precalculated nominal trajectory is after all reliant on these constraints holding true at all times, which will not be the case when there are errors in the system. The simulations were carried out by again introducing error parameters $\delta_1$ and $\delta_2$. For the first simulation, the initial generalized coordinates $q_2(0)$ and $q_5(0)$ were each given an error parameter such that

$$q_2((0) = q_2(0)^* + \delta_1 \quad q_5((0) = q_5(0)^* + \delta_2 \tag{70}$$

with the errors chosen so the SemiQuad would still start its gait standing with both feet on the ground. For the first simulation the error parameters were set to $\delta_1 = 0.01$

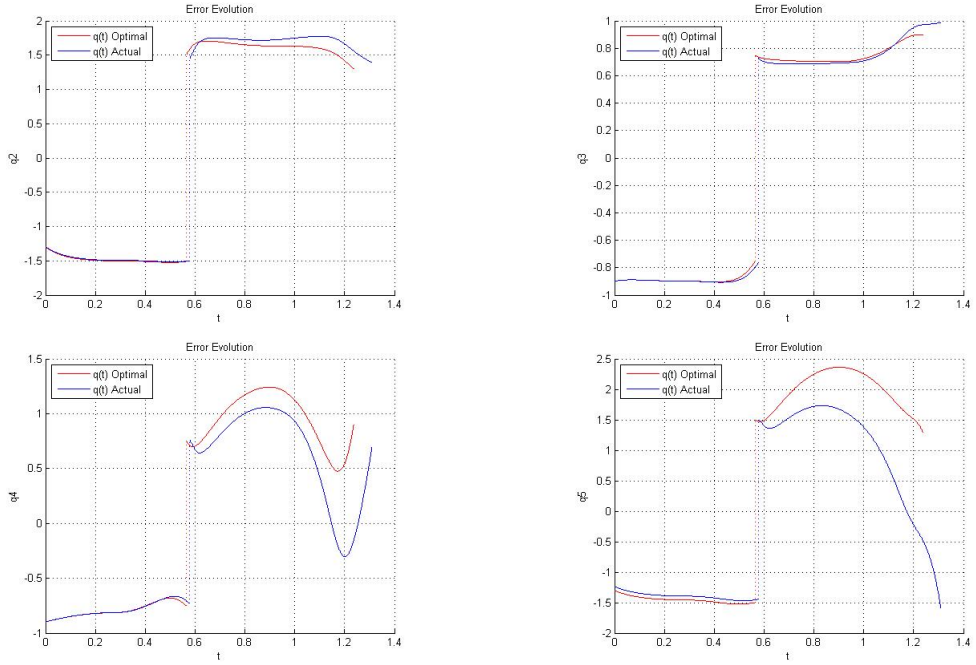



Figure 11: The Evolution of the Generalized Coordinates $q_i$. $i = 2..5$ given an initial error in $q_2$ and $q_5$.

and $\delta_2 = 0.055$. This resulted in the SemiQuad being tilted backwards from start. The evolution of the four generalized coordinates are shown in fig(11) together with the nominal values. It can be seen that the magnitude of the errors for $q_2$ and $q_5$ stay roughly the same throughout the first two steps. The two remaining generalized coordinates $q_3$ and $q_4$ will for the same period follow their respective nominal values closely. However, the fault in the system becomes apparent at the impact, as the faulty system makes an impact with the ground much later. As shown in the phase portrait given by fig(12), this introduces additional errors in $[\theta, \dot{\theta}]$, as their values go past those that would have given an optimal impact. This further entails step 3 beginning with much larger errors, and the SemiQuad eventually fails to complete the gait. The simulation was then repeated for new larger values of $\delta$, with $\delta_1 = 0.02$ and $\delta_2 = 0.11$. The results were much the same, though the effect of the impact became more visible. This shows that even in the presence of errors, most coordinates may follow their nominal values closely during the motion phase of step 1 and step 2. However, at the moment of impact the faults in the system will become apparent, with errors also manifesting themselves in the coordinates that followed their nominal values before the impact. This obviously has an affect on the functionality of the SemiQuad as a single error might give rise to several others.

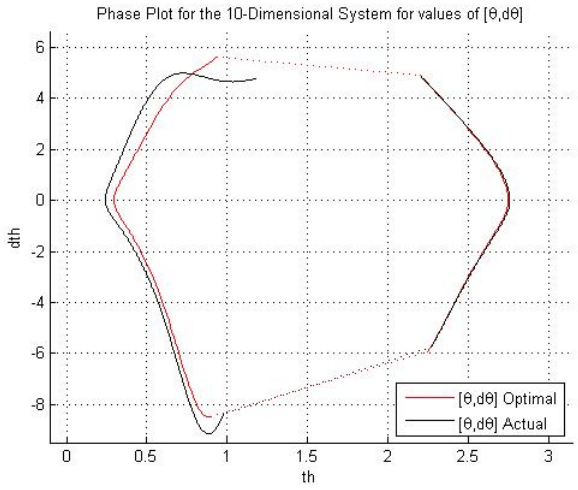



Figure 12: The Phase Portrait $[\theta, \dot{\theta}]$ given an initial error in $q_2$ and $q_5$.

### 7.3 Analysis of the Errors

Simulating the SemiQuad with varying degrees of errors has shown the true difficulty of implementing the gaits found in the previous chapter. Consequently, two important revelations must be addressed before any implementation can be attempted. The first is that errors must always be expected and properly accounted for. Indeed, many of the simulations carried out in the previous section were done with errors of a quite a small magnitude, though despite their size their affect on the system was still shown to be severe. This is due to the second revelation found, which was that any error in the system, large or small, would eventually cause the system to fail. As seen for the first simulations in section 7.2.1, the SemiQuad was capable of completing the gait, though its final configuration could be said to be worse than its initial one. This seemed to be true for all simulations, that any error in the system were expected to grow throughout the gait. Having errors grow for each step taken will naturally make any implementation fail, as the goal is after all to achieve a continuous periodic walking motion, with several full steps being taken. This can be seen as the main issue with designing gaits based on the paradigm of dynamic walking. After all, the walking motion is as mentioned in chapter 2, designed using the notion of cyclic stability, meaning the motion is said to be stable as a whole, but not necessarily stable at exact moments in time. As a result of this, there is no region of attraction to speak of, meaning no key points the values of the Semiquad will be attracted to provided they are close enough. Thus, if the Semiquad is no longer following the trajectory, which is the case in (68), its gait can no longer be said to be orbitally stable.
The different form of stability makes the process of developing suitable descriptions of the errors and consequently the development of a suitable controller more complicated. The controller must after all be able to ensure the gait achieves orbital stability also in the presence of errors. The creation of said controller will therefore be discussed in chapter

9. However, as it is clear that any failure in carrying out the gait will happen due to a failure of following the predesigned trajectory, the deviation from the trajectory must be adequately described. This description will therefore be carried out in the following chapter using the notion of Transverse Linearization.

# 8 Transverse Linearization

One of the main challenges in the actual implementation of a gait is to suitably describe how the motion of the Semiquad may deviate from the the predesigned trajectory. The description should not only provide a reliant and easily understandable measurement of any error, but should also allow for the construction of control directly utilizing these measurements. However, given the nonlinear nature of the system, the implementation of direct control is not a trivial task. It is for this reason that linearization of nonlinear systems is highly desirable as it allows access to a myriad of already well established control schemes built for linear systems. Given the specific nature of our system however, conventional methods of linearization such as the linearization of the full-state dynamics will not suffice. The reason for this is that such a linearization will not be suitable for examining local properties, and additionally it will not be asymptotically stable around a periodic trajectory[17]. It therefore becomes clear that any linearization must instead focus solely on the local behaviour around the solution. Then in vicinity of the orbit, our 2n dimensional nonlinear system can with a suitable change of coordinates be decomposed into two subsystems[21]: A scalar system representing the dynamics along the trajectory, and a $(2n-1)$-dimensional system representing the dynamics transverse to the trajectory. The scalar variable of the first system can be regarded as representing the position along the trajectory. However, it can be shown that within close proximity to the solution trajectory, the local properties of the system dynamics will actually be independent of this variable, and it can therefore be omitted[17]. This leaves only the second subsystem, and thus the process of transverse linearization is then the linearization of the coordinates given by this subsystem, which creates a $(2n-1)$-dimensional linear system. These $(2n-1)$ coordinates, which further will be known as the transverse coordinates, then represents the dynamics *transverse* to the trajectory of the solution, thus automatically defining a moving Poincarè section, a notion that will be expanded upon in the following section.

## 8.1 Poincarè Return Maps and Moving Poincarè Sections

One of the classical techniques for determining the existence and stability of periodic orbits for nonlinear systems is the method of Poincarè return maps. This method transforms the problem of finding periodic orbits into one of finding fixed points of a map, or more directly how to discover equilibrium points of a certain discrete-time nonlinear system[15]. An example of a Poincarè return map is given in fig(13). The surface $S$ is transversal to the flow of the periodic orbit of the dynamic system $x^*$. The poincarè map is defined by the first hit rule[18], meaning it maps the points of the solution of the nonlinear system belonging to $S$ into the points where they hit $S$ again. Thus if the Poincarè map is contracting, meaning the hit after following the trajectory is closer in distance than the previous period, the orbit is said to be asymptotically stable. Likewise, exponential orbital stability can be ensured using linearization of the poincarè map, namely $dP : TS \to TS$, where $TS$ is the tangent space of $S$ at the intersection with the trajectory.

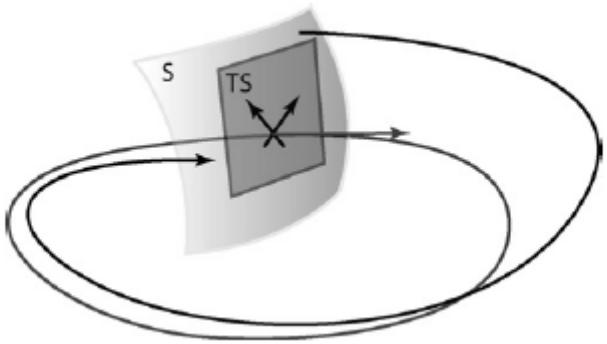



Figure 13: A Poincarè return map for a given cyclic trajectory

There are several drawbacks concerning this method however. First and foremost, the return map cannot be solved directly as it requires the closed-form solution of a non-linear ordinary differential equation. This requires numerical methods to be employed to discover suitable solutions, making the method quite computationally intensive. Further, [15] argues that the computations are not insightful, meaning there is not much causality between the existence or stability properties of a solution, and the actual dynamics of the system. It is important to note that difficult does not necessarily mean impossible as there have been numerous successful implementations of this method, for example in [5].

This difficulty is however naturally a motivation for exploring alternative methods, such as the concept of poincarè return sections. It is built upon introducing not one surface transverse to the trajectory but a family of transverse surfaces $S(t)_{t\in[0,T]}$, parameterized by the points on the cyclic trajectory[18], see fig(14). It is this method

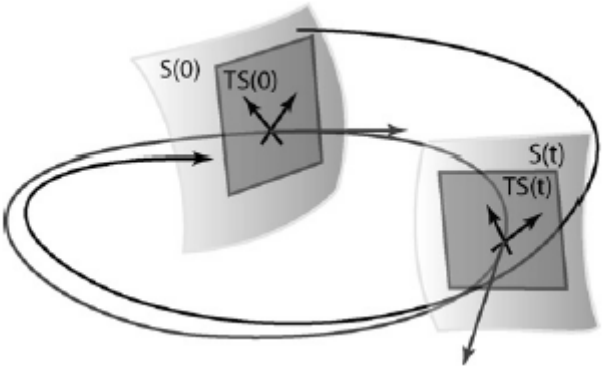



Figure 14: A Poincarè return section for a given cyclic trajectory

that is the basis for transverse linearization. The coordinates described in the previous section automatically creates the moving Poincarè section, thus offering a decription of

the deviation from a point on the orbit $x^*$ as a distance along the surface $S(\cdot)$, with $x^*$ being the intersection of $S$ and the trajectory.

## 8.2 The 3-Step Method of Transverse Linearization

The previous sections defined the general idea of transverse linearization and briefly touched upon the mathematical background. This section will expand upon this and fully implement the method of transverse linearization for the Semiquad system. The first observation however is that a single linearization of a dynamic orbit will not be enough. That is because the total gait of the Semiquad is a hybrid dynamic system consisting of 4 interchanging steps with both switching phases and impact phases. These sections must also be included in any general linearization. Indeed, the total process can be summed up in three steps[19]:

1. Linearize the Update Laws, including the shift in parameters $\mathcal{F}_{shift}(z)$ and the impact update phase $\mathcal{F}_{swap}(\Delta_{zero}(z^-))$, at the switching surfaces $\Gamma_-$ and $\Gamma_+$.

2. Linearize the transverse dynamics of the complete dynamic system along the continuous-in-time orbit of $q_*(t)$.

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

Thus, each individual segment of motion and each individual discrete segment will be linearized independently, and then merged together. This is carried out step-by-step in the following sections.

### 8.2.1 Linearizing the Update Laws

Since the gait exhibits discontinuous sections in between the interchanging steps of solving the dynamic system, these must naturally also be included in the linearization of the system. Now the linearization of these sections will be given as the jacobian of the switching conditions calculated at the moment of impact[19]. For the SemiQuad system that means at the shift after step 1 and 3 and at the impact at the end of step 2 and 4. This is calculates as

$$dF = \left.\frac{\partial F}{\partial[q,\dot{q}]}\right|_{\substack{q=q_*(T_-)\\ \dot{q}=\dot{q}_*(T)}} : T\Gamma_-|_{\substack{q=q_*(T_-)\\ \dot{q}=\dot{q}_*(T_-)}} \rightarrow T\Gamma_+|_{\substack{q=q_*(t_0)\\ \dot{q}=\dot{q}_*(t_0)}} \tag{71}$$

with $T\Gamma_-$ and $T\Gamma_+$ being the tangent planes to their respective switching surfaces. The specific implementation depends on which switching phase the trajectory is reaching. For the first switch at the end of step 1, and likewise for step 3, the switching surface is given by the moment when the angular velocity of $\theta$ reaches zero. That is

$$\begin{aligned} {\Gamma^1}_- &\in \left\{\dot{\theta}=0\right\} \\ {\Gamma^3}_- &\in \left\{\dot{\theta}=0\right\} \end{aligned} \tag{72}$$

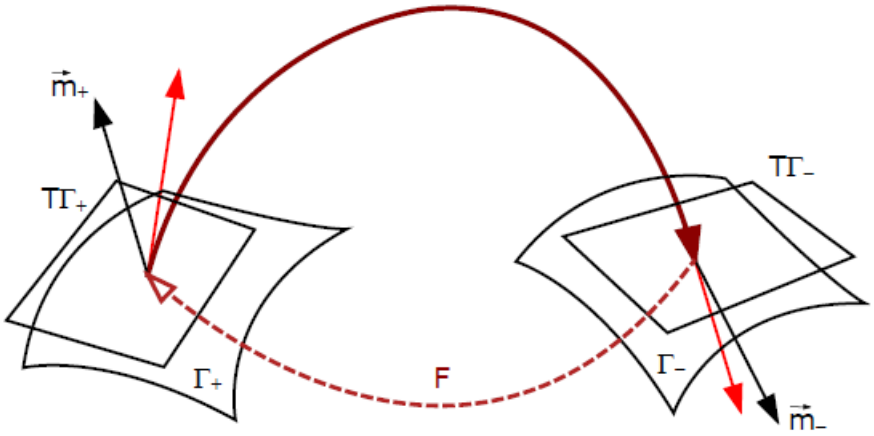



Figure 15: Tangent planes $T\Gamma_-$ and $T\Gamma_+$ to the switching surfaces $\Gamma_-$ and $\Gamma_+$ at the points of intersection with the periodic orbit $[q_*, \dot{q}_*]$[14].

The linearization is then given by calculating the jacobian of the switching surface. This yields the vector $\overrightarrow{m_i}$, the normal to the tangent plane of the switching surface, $T\Gamma^i$, at the point of intersection between the switching surface and the trajectory, see fig(15). Given the simplicity of the switching surface, the vectors can easily be calculated as

$$\overrightarrow{m_i} = [0, 0, 0, 0, 0, 1, 0, 0, 0, 0] \quad i = [1, 3] \tag{73}$$

The remaining switching surfaces, that is at the end of step 2 and 4, are a bit more complicated. As explained before, the switching occurs when the swing leg hits the ground, which is given by (19). The switching condition is then given as

$$P^y_{end} = 0 \tag{74}$$

The vectors $\overrightarrow{m_i}$ for step 2 and 4 are then calculated as before by taking the jacobian of the switching surfaces

$$\overrightarrow{m_i} = [m_\theta, m_{q_2}, m_{q_3}, m_{q_4}, m_{q_5}, 0, 0, 0, 0, 0] \quad m_\theta = \frac{\partial P^y_{end}}{\partial \theta}, \; m_{q_i} = \frac{\partial P^y_{end}}{\partial q_i} \tag{75}$$

with

$$\begin{aligned} m_\theta = \; & l_1 cos(\theta) + l_2 cos(\theta + q_2) + l_3 cos(\theta + q_2 + q_3)) + \dots \\ & l_4 cos(\theta + q_2 + q_3 + q_4) + l_5 cos(\theta + q_2 + q_3 + q_4 + q_5) \\ m_{q_2} = \; & l_2 cos(\theta + q_2) + l_3 cos(\theta + q_2 + q_3)) + \dots \\ & l_4 cos(\theta + q_2 + q_3 + q_4) + l_5 cos(\theta + q_2 + q_3 + q_4 + q_5) \\ m_{q_3} = \; & l_3 cos(\theta + q_2 + q_3)) + \dots \\ & l_4 cos(\theta + q_2 + q_3 + q_4) + l_5 cos(\theta + q_2 + q_3 + q_4 + q_5) \\ m_{q_4} = \; & l_4 cos(\theta + q_2 + q_3 + q_4) + l_5 cos(\theta + q_2 + q_3 + q_4 + q_5) \\ m_{q_5} = \; & l_5 cos(\theta + q_2 + q_3 + q_4 + q_5) \end{aligned} \tag{76}$$

thus all 4 vectors have been calculated. Their use will become clear in section 8.2.3.

### 8.2.2 Linearization of the Transverse Dynamics

With the intermediary impact surfaces handled in the previous section, the task at hand is the linearization of the motion phases of the SemiQuad. As mentioned, this is given as the linearization of the $(2n-1)$ new coordinates that exist transverse to the solution at a given point on the predesigned trajectory. As these coordinates lie transverse to the solution, they will for each instant of time $t$ create a separate poincarè section $S(t)$. The evolution of these coordinates then naturally creates a family of transverse surfaces, creating the moving Poincarè section. The linearization of the transverse coordinates will then be defined on the tangent planes to these surfaces $TS(t)$, as shown in fig(16). However, before a linearization can take place, the coordinates must first be chosen. Thus, the first task is naturally the discovery of suitable coordinates that provide a good representation of the nonlinear system in a vicinity of the solution.

**Choosing the Error Coordinates**

The choice of new coordinates is primarily linked to allowing a suitable description of potential errors in following the predesigned optimal gait. It is then natural to start the search by looking at the original motion of each step described by the evolution of the generalized coordinates and their respective velocities, that is

$$q_i = q_i^*(t), \quad \dot{q_i} = \dot{q_i^*}(t), \quad i = 1..5, \quad t \in [0, T] \tag{77}$$

where the motion of the SemiQuad will be erroneous when (77) no longer holds true. The description of the motion was however greatly simplified by the introduction of Virtual Holonomic Constraints, with the key point in the implementation of said constraints being that they must be held true throughout the gait. A possible representation of an error will then be the degree to which its corresponding virtual holonomic constraint is being breached. This allows the deviation of any generalized coordinate to be described by a new set of variables $y$ as

$$y_i = q_i - \phi_i(\theta), \quad i = 1..5 \tag{78}$$

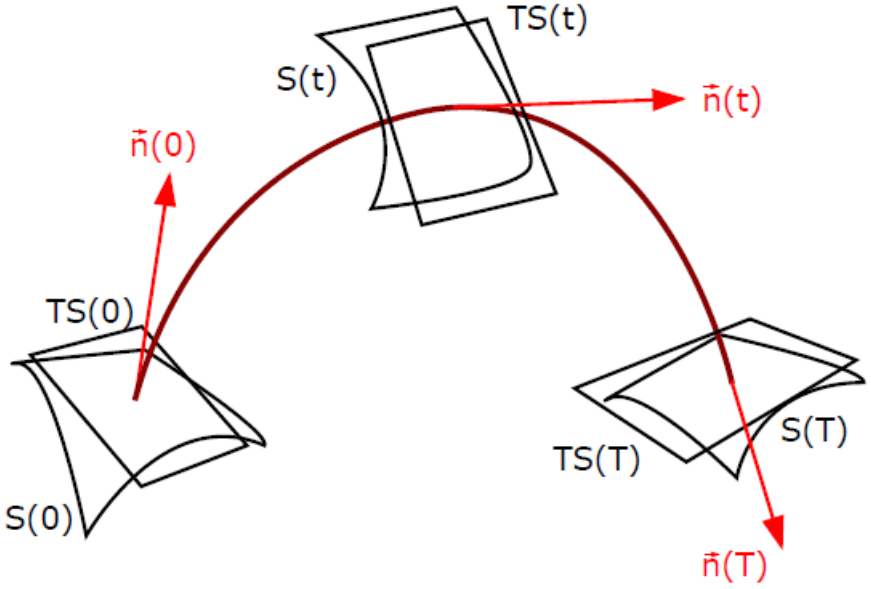



Figure 16: Moving Poincaré section $\{S(t)_{t\in[0,T]}\}$ transversal to the continuous-time part of the periodic orbit $[q_*, \dot{q}_*]$. The linearization of the transverse dynamics is valid on the tangent planes $TS(t)$ to the Poincarè sections $S(t)$[14].

yielding the $n$ error variables $y$. However, as we are using $n+1$ coordinates to describe the $n$ actual generalized coordinates of the SemiQuad, the description (78) can be said to be excessive. This allows a given coordinate $y_i$ to instead be described as a function of the other coordinates[18]. The choice of this coordinate is arbitrary, but for simplicity $y_1$ has been chosen for this system. The final independent coordinates are then

$$y = (y_2, ..y_n) \in \mathbb{R}^{n-1}, \quad \theta \in \mathbb{R} \tag{79}$$

In addition comes the description of $y_1$, which is to be given as a combination of the above coordinates. This can be written as

$$q_1 = \phi(\theta) + h(\theta, y) \tag{80}$$

where $h(\theta, y)$ is a scalar smooth function. For the SemiQuad system however, the introduction of the virtual holonomic constraints states that $q_1 = \theta$, simplifying the above expression greatly

$$q_1 = \phi_1(\theta) + h(\theta, y) \rightarrow q_1 = \theta, \tag{81}$$

given $\phi_1(\theta) = \theta$, the function $h(\theta, y)$ has the simplest form possible, namely zero. With the generalized coordinates defined, the generalized velocities and accelerations can be readily derived introducing

$$\dot{q} = L(\theta, y) \begin{bmatrix} \dot{y} \\ \dot{\theta} \end{bmatrix}, \quad \ddot{q} = L(\theta, y) \begin{bmatrix} \ddot{y} \\ \ddot{\theta} \end{bmatrix} + \dot{L}(\theta, y) \begin{bmatrix} \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{82}$$

with $L$ being a nonsingular matrix given as[14]

$$L(\theta, y) = \begin{bmatrix} \dfrac{\partial h(\cdot)}{\partial y} & \dfrac{\partial h(\cdot)}{\partial \theta} \\ \mathbf{I}_{(n-1)} & \mathbf{0}_{(n-1)x1} \end{bmatrix} + \begin{bmatrix} & \phi_1^{'} \\ \mathbf{0}_{nx(n-1)} & \vdots \\ & \phi_n^{'} \end{bmatrix} \tag{83}$$

Note that the first row now contains the gradient of $h$, that is $\Delta h(\theta, y)$. This is due to the fact that $q_1$ was chosen to be described by the other coordinates and not $q_n$ as suggested in the literature. For the SemiQuad system, $L(\theta, y)$ and consequently $\dot{L}(\theta, y)$ becomes

$$L(\theta, y) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & \phi_2^{'}(\theta) \\ 0 & 1 & 0 & 0 & \phi_3^{'}(\theta) \\ 0 & 0 & 1 & 0 & \phi_4^{'}(\theta) \\ 0 & 0 & 0 & 1 & \phi_5^{'}(\theta) \end{bmatrix}, \quad \dot{L}(\theta, y) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \phi_2^{''}(\theta)\dot{\theta} \\ 0 & 0 & 0 & 0 & \phi_3^{''}(\theta)\dot{\theta} \\ 0 & 0 & 0 & 0 & \phi_4^{''}(\theta)\dot{\theta} \\ 0 & 0 & 0 & 0 & \phi_5^{''}(\theta)\dot{\theta} \end{bmatrix} \tag{84}$$

It can be seen that this makes the description of the ankle-angle $q_1$ the same as before with $\dot{q}_1 = \dot{\theta}$ and $\ddot{q}_1 = \ddot{\theta}$ while the additional coordinates are changed into

$$\begin{aligned} q_i &= y_i + \phi_i(\theta) \\ \dot{q}_i &= \dot{y}_i + \phi_i^{'}(\theta)\dot{\theta} \\ \ddot{q}_i &= \ddot{y}_i + \phi_i^{''}(\theta)\dot{\theta}^2 + \phi_i^{'}(\theta)\ddot{\theta} \end{aligned}_{\ i=2\ldots5} \tag{85}$$

Now solving the first two equations in the above expression for $y$ and $\dot{y}$ yields the remaining $(2n-2)$ error coordinates. Each represents the degree to which a generalized coordinate or velocity fails to satisfy its given virtual holonomic constraint. Simply put if $y_i \neq 0$, there is an error with generalized coordinate $i$. Likewise if $\dot{y}_i \neq 0$, there is an error with generalized velocity $i$. The requirement for a suitable error description is then fulfilled and the error coordinates $y$ and $\dot{y}$ can be taken as the first $(2n-2)$ transverse coordinates.

**The y-dynamics**

With the new generalized coordinates expressed by (85), the associated dynamics must be expressed with respect to the original system. The dynamics of the $y$-variables can be expressed as[17]

$$\ddot{y} = \mathcal{R}(\theta, \dot{\theta}, y, \dot{y}) + \mathcal{N}(\theta, y)u \tag{86}$$

which represents the synchronization error to the specified virtual holonomic constraints[14]. This expression is calculated from the original dynamic equation given by

$$\ddot{q} = \mathcal{M}(q)^{-1}\left[-\mathcal{C}(q, \dot{q})\dot{q} - \mathcal{G}(q) + \mathcal{B}u\right] \tag{87}$$

by first inserting the relations provided in (82) and rearranging the terms to get

$$L(\theta, y)\begin{bmatrix} \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \mathcal{M}^{-1}(q)\left[-\mathcal{C}(q, \dot{q})\dot{q} - \mathcal{G}(q) + \mathcal{B}u\right] - \dot{L}(\theta, y)\begin{bmatrix} \dot{y} \\ \dot{\theta} \end{bmatrix} \tag{88}$$

When further solving for $\ddot{y}$ and with (85) holding true, (86) is obtained with

$$\begin{aligned}\mathcal{R}(\theta,\dot{\theta},y,\dot{y}) &= QL^{-1}(\theta,y)\left[-\mathcal{M}^{-1}(q)\left(\mathcal{C}(q,\dot{q})\dot{q}+\mathcal{G}(q)\right)-\dot{L}(\theta,y)\begin{bmatrix}\dot{y}\\ \dot{\theta}\end{bmatrix}\right]\\ \mathcal{N}(\theta,y) &= QL^{-1}(\theta,y)\mathcal{M}^{-1}(q)\mathcal{B}\end{aligned} \tag{89}$$

where $Q = [\mathbf{1}_{(n-1)\times(n-1)}, \mathbf{0}_{(n-1)\times 1}]$ is a matrix that extracts $\ddot{y}$ from $[\ddot{y}, \ddot{\theta}]^T$. With the dynamics of the $y$-variables calculated, the goal is to establish a connection with the original system. First a control transform from $u$ to a variable $v$ can be given as[17]

$$u = v + U(\theta,\dot{\theta},y,\dot{y}), \quad u_*(t) = U(\theta_*(t),\dot{\theta}_*(t),\mathbf{0},\mathbf{0}) \tag{90}$$

with $u_*(t)$ being the ideal actuation when no errors exist, that is when $y = \dot{y} = 0$. The new variable $v$ can then be thought of as a representation of the error with respect to actuation, that is $v = 0$ when no errors exist. Choosing $\ddot{y} = v$, (86) becomes

$$\ddot{v} = \mathcal{R}(\theta,\dot{\theta},y,\dot{y}) + \mathcal{N}(\theta,y)u \tag{91}$$

which when solved for $u$ yields an expression of actual actuation derived from the $y$-variables

$$u = \mathcal{N}^{-1}(\theta,y)\left[v - \mathcal{R}(\theta,\dot{\theta},y,\dot{y})\right] \tag{92}$$

This provides a link between the actuation of the linear system $v$ to the nonlinear system $u$, something which will become tremendously important for the implementation of a control system in the next chapter.

**The Complete Error Dynamics**

The previous section described the y-dynamics, which represents only the part of the dynamics of the mechanical system given by the new coordinates in (79). This must be complemented by a scalar second order differential equation for the $\theta$-variable[17]. This equation can be constructed using the zero-dynamics equation given by (34), given that $\ddot{\theta}$ is separated from zero on the orbit, that is $\alpha_i(\theta_*(t)) \neq 0$. The $\theta$-dynamics can then be rewritten as

$$\alpha_i(\theta)\ddot{\theta} + \beta_i(\theta)\dot{\theta}^2 + \gamma(\theta) = g_i(\theta,\dot{\theta},\ddot{\theta},y,\dot{y},v) \tag{93}$$

where $\ddot{y}$ has been substituted by $v$. Note that if the following is true

$$y_2 = y_{2*}(t) = 0, \ \ldots y_5 = y_{5*}(t) = 0, \quad \theta = \theta_*(t), \quad \dot{\theta} = \dot{\theta}^*(t) \tag{94}$$

then (93) will have the same form as the zero-dynamics as the smooth function $g_i(\cdot)$ is zero along the optimal trajectory. This notion is quite important as it allows $g_i(\cdot)$ to be decomposed into the following relation[17]

$$g_i(\cdot) = g_y^i(\cdot)y + g_{\dot{y}}^i(\cdot)\dot{y} + g_v^i(\cdot)v \tag{95}$$

known as the $g$-functions. This relation holds true provided the solution is close to the nominal trajectory. The $g$-functions are calculated as follows

$$\begin{aligned} g_y &= \frac{\partial g_i(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y}, v)}{\partial y} \\ g_{\dot{y}} &= \frac{\partial g_i(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y}, v)}{\partial \dot{y}} \quad , \quad y = 0,\ \dot{y} = 0,\ \ddot{y} = 0 \\ g_v &= \frac{\partial g_i(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y}, v)}{\partial v} \end{aligned} \tag{96}$$

and the complete dynamics can be written as

$$\ddot{y} = v \tag{97}$$

$$\alpha_i(\theta)\ddot{\theta} + \beta_i(\theta)\dot{\theta}^2 + \gamma(\theta) = g_y^i(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y})y + g_{\dot{y}}^i(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y})\dot{y} + g_v^i(\theta, \dot{\theta}, \ddot{\theta}, y, \dot{y})v \tag{98}$$

This process is summed up in the following theorem[17]:

***Theorem 1:*** *Let $q = q_*(t)$ be the solution of (87) with $u = u_*(t)$ both defined and $C^1$-smooth on $[0, T]$. Suppose $\phi_1(\cdot), \ldots, \phi_5(\cdot)$ are $C^2$-smooth functions representing an alternative parametrization (32) of this motion such that the smooth $n \times n$ matrix function $L(\cdot)$ with coefficients defined by (83) is invertible in a vicinity of the orbit. Then the quantities $[\theta, y_2, \ldots, y_5]$ defined by (79) and (81) are the generalized coordinates for the mechanical system. Moreover, the dynamics (87) can be locally rewritten as (97)-(98).*

This states that the description of the system using the new coordinates will be equivalent to the actual system as long as it is in the vicinity of the desired trajectory[17]. Thus a linear system can be created, which should in proximity of the optimal solution be able to suitable describe the system.

**Creating the Linear System**

With the $y$-coordinates numbering eight, one additional coordinate remains to be found to have the $(2n-1)$ necessary coordinates. As the $y$-coordinates each describe the error of a specific coordinate $q$ or the belonging velocity $\dot{q}$, the last variable must naturally give a description of the possible deviation of the value of $\theta$. Looking at the system description (98), it can be seen that it is not resolved with respect to $\ddot{\theta}$, with $\ddot{\theta}$ appearing on both sides of the equality. The reason for doing this is that (98) is integrable[18], and thus a reasonable choice for the last coordinate is the solution of the integral $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$, given by (38). Then the complete set of $(2n-1)$ transverse coordinates are given by

$$X_\perp^{(i)} = \begin{bmatrix} I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0) \\ y_2 \\ \vdots \\ y_5 \\ \dot{y}_2 \\ \vdots \\ \dot{y}_5 \end{bmatrix} \tag{99}$$

which will suitably describe the new system in a vicinity of the solution (94)[17]. With the transverse coordinates $X_\perp$ found, a linearization of the dynamics can be created, which should be valid in the vicinity of the periodic motion[18]. This is given as

$$\dot{X}_\perp(t) = \mathcal{A}(t)X_\perp(t) + \mathcal{B}(t)v(t) \tag{100}$$

where $\mathcal{A}$ and $\mathcal{B}$ are given as[18]

$$\begin{aligned} \mathcal{A}(t) &= \begin{bmatrix} a_{11}(t) & a_{12}(t) & a_{13}(t) \\ \mathbf{0}_{(n-1)\times 1} & \mathbf{0}_{(n-1)\times(n-1)} & \mathbf{I}_{(n-1)\times(n-1)} \\ \mathbf{0}_{(n-1)\times 1} & \mathbf{0}_{(n-1)\times(n-1)} & \mathbf{0}_{(n-1)\times(n-1)} \end{bmatrix} \\ \mathcal{B}(t) &= \begin{bmatrix} b_1(t) \\ \mathbf{0}_{(n-1)\times(n-1)} \\ \mathbf{I}_{(n-1)\times(n-1)} \end{bmatrix} \end{aligned} \tag{101}$$

with coefficients

$$\begin{aligned} b_1(t) &= 2\dot{\theta}_* \frac{g_v(\theta_*(t), \dot{\theta}_*(t), 0, 0)}{\alpha(\theta_*(t)} \\ a_{11}(t) &= 2\dot{\theta}_* \frac{g_y(\theta_*(t), \dot{\theta}_*(t), \ddot{\theta}_*(t), 0, 0)}{\alpha(\theta_*(t)} \\ a_{12}(t) &= 2\dot{\theta}_* \frac{g_{\dot{y}}(\theta_*(t), \dot{\theta}_*(t), \ddot{\theta}_*(t), 0, 0)}{\alpha(\theta_*(t)} \\ a_{13}(t) &= 2\dot{\theta}_* \frac{g_v(\theta_*(t), \dot{\theta}_*(t), \ddot{\theta}_*(t), 0, 0)}{\alpha(\theta_*(t)} \end{aligned} \tag{102}$$

which allows the linear dynamics to be calculated directly. The evolution of this system can be seen as a linear approximation of the evolution of the errors, described by the transverse coordinates. For any description to be adequate, the evolution of said dynamics should resemble the real error evolution when simulating the real 10-dimensional nonlinear system. This will be studied in section 8.3.

### 8.2.3 Merging the Discrete and Continuous Sections

Combining the linearization of the update law given by (71) with the transverse linearization given in the previous section (100)-(102), is not a straightforward task. The linear mapping acts between the hyperplanes $T\Gamma_-$ and $T\Gamma_+$, calculated by the update laws, while the linear differential equation (100) maps a vector on $TS(0)$ to a vector on $TS(T_h)$[19]. The problem of combining these sections is a result of the hyperplanes $T\Gamma_-$ and $TS(T_h)$, and likewise $T\Gamma_+$ and $TS(0)$, not coinciding with each other. Therefore, upon reaching the end of the trajectory as given by the solution of the linear differential equation at $T_h$, the update law cannot be executed as the solution does not lie on the

plane given by $T\Gamma_-$. There are naturally two solutions to this problem. The first, and perhaps most laborious, is to go back to the creation process of these planes and ensure they indeed are equal, that is $T\Gamma_- = TS(T_h)$ and $T\Gamma_+ = TS(0)$. This is not easy to do however, as an alternative to (100)-(102) must then be computed. A more suitable solution is to change the total update law $dF(\cdot)$ so that it indeed acts from $TS(T_h)$ onto $TS(0)$, with the choice of $S(t)_{t\in[0,Th]}$ as in the previous section. It is defining this extended update law which is the motivation for this section. To sum up the scenario the following is provided:

1. The hyperplanes $TS(0)$, $TS(T_h)$, the normal vectors $\overrightarrow{n}(0)$, $\overrightarrow{n}(T_h)$, defined by a moving poincarè section $\{S(t)\}_{t\in[0,T_h]}$, associated with the continuous-in-time sub-arc of $q_*(t)$.

2. The tangent planes $T\Gamma_-$ and $T\Gamma_+$ to the switching surfaces, that are transversal to the hybrid-periodic motion defined at the end-points of the continuous-in-time sub-arc
$$[\dot{q}_*(0_+); \ddot{q}_*(0_+)] \notin T\Gamma_+ \quad [\dot{q}_*(T_{h-}); \ddot{q}_*(T_{h-})] \notin T\Gamma_- \tag{103}$$

3. The linear mapping $dF : T\Gamma_- \to T\Gamma_+$ defined by (71)

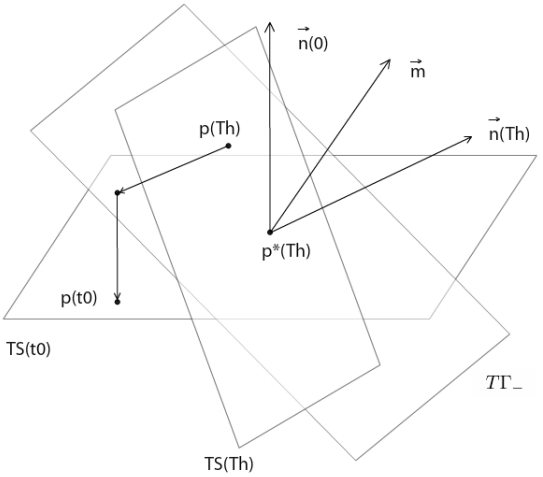



Figure 17: Point $p_*(T_h)$ shows the ideal final configuration of the coordinates, common for $S(T_h)$ and $\Gamma_-$. With the transverse linearization however, the ending configuration might equal another point, that is $p(T_h) \notin T\Gamma_-$, and thus a shift must take place. In addition a second shift must be implemented to bring the coordinates from $T\Gamma_+$ over to a suitable starting configuration on $TS(t_0)$.

The problem, as illustrated in figure (17), can thus be described as finding the new update laws to transform a given solution in transverse coordinates from first the plane

spanned by $TS_i(T_h)$ for a current step $i$, to a suitable starting configuration on plane $TS_{i+1}(t_0)$ for the next step. This will be done step-by-step in the next section.

**Creating the Projections**

As the transverse linearization arrives at the end of a step, the solution will be given by the transverse coordinates on the plane $TS(T_h)$. Now as the impact laws are defined for the original system, the transverse coordinates must first be related to the corresponding coordinates in the original system $\left[\Delta q(T_{h^-}), \dot{q}(T_{h^-})\right] \in \mathbb{R}^{2n}$[21]. This can be done by noting the relation between the linear parts of the increments of the transverse coordinates, and the linear parts of the generalized coordinates given by

$$\begin{bmatrix} \Delta I \\ \Delta y \\ \Delta \dot{y} \end{bmatrix} = \mathcal{L}(t) \begin{bmatrix} \Delta q \\ \Delta \dot{q} \end{bmatrix} = \mathcal{L}(t)\Delta x \tag{104}$$

where $\mathcal{L}(t)$ is given by[21]

$$\mathcal{L}(t) = \begin{bmatrix} -2\ddot{\theta}_*(t) & \mathbf{0}_{1\times(n-1)} & 2\dot{\theta}_*(t) & \mathbf{0}_{1\times(n-1)} \\ -\Phi^{'}(\theta_*(t)) & \mathbf{I}_{(n-1)\times(n-1)} & \mathbf{0}_{(n-1)\times 1} & \mathbf{0}_{(n-1)\times(n-1)} \\ -\Phi^{''}(\theta_*(t))\dot{\theta}_*(t) & \mathbf{0}_{(n-1)\times 1} & -\Phi^{'}(\theta_*(t)) & \mathbf{I}_{(n-1)\times(n-1)} \end{bmatrix} \tag{105}$$

with

$$\Phi(\theta(t)) = [\phi_2(\theta(t)), \phi_3(\theta(t)), \phi_4(\theta(t)), \phi_5(\theta(t))]^T \tag{106}$$

To create the inverse relation, the matrix $\mathcal{L}(t)$ must be extended from a $9\times 10$ matrix into a square matrix. This can be done by introducing the vector

$$n(t) = \begin{bmatrix} \dot{q}_*(t) \\ \ddot{q}_*(t) \end{bmatrix} \tag{107}$$

as the final row, thus creating a $10\times 10$ matrix. The vector of transverse coordinates $x_\perp$ can then be extended to a $10\times 1$ vector by adding a zero at the end, that is

$$\zeta = \begin{bmatrix} x_\perp \\ 0 \end{bmatrix} \tag{108}$$

Now as $n(t)$ is orthogonal to the surface $TS(t)$, it is naturally orthogonal to any vector in $TS(t)$. This creates the relation that $n(t)\times\Delta x = 0$, which is always true as $\Delta x \in TS(T_h)$. The extended form of both matrix and vector are therefore in fact equivalent to the original forms, and the complete inverse relation can be written as

$$\Delta x = \begin{bmatrix} \Delta q \\ \Delta \dot{q} \end{bmatrix} = \begin{bmatrix} \mathcal{L}(t) \\ n(t) \end{bmatrix}^{-1} \begin{bmatrix} \zeta \\ 0 \end{bmatrix} \tag{109}$$

Using (109), the linear parts of the generalized coordinates can be obtained at the end of a given step $\Delta x(T_h)$. Then the projection of $TS(T_h)$ onto $T\Gamma_-$ can be calculated. The first step is to define a set of new coordinates $\delta x(T_h) \in T\Gamma_-$, which are given as

$$\delta x(T_h) = \Delta x(T_h) + f(x_*(T_h), u_*(T_h))\tau_- \tag{110}$$

where $f(x_*(T_h), u_*(T_h)) = n(T_h)$, or simply the vector orthogonal to the moving poincarè surface of the coordinates. Thus the projection starts at $\Delta x(T_h)$ and moves along the the orthogonal vector $n(T_h)$. The plane of the switching surface, $T\Gamma_-$ has the orthogonal vector given by $m(T_h)$, calculated as in 8.2.1. Given that $m(T_h)$ is orthogonal to the plane, the following naturally also holds true

$$m(T_h) \times \delta x(T_h) = 0 \tag{111}$$

meaning

$$m(T_h) \times (\Delta x + n(T_h)\tau_-) = 0 \tag{112}$$

the expression above can thus be solved for $\tau_-$, yielding the remaining expression of the update law allowing the new coordinates to be calculated. The whole process can be summed up as

$$\delta x(T_h) = \Delta x(T_h) - \left( \frac{\Delta x(T_h) \times m(T_h)}{n(T_h) \times m(T_h)} \right) n(T_h) \tag{113}$$

As $\delta x(T_h) \in T\Gamma_-$, the update laws can be applied as normal. Then, as mentioned in the previous section, a second projection must also take place. The current coordinates, given by (113), must further be projected onto the tangent to the poincarè surface of the next step, that is $TS(0)$. This can be done by modifying the expression already given for the first projection. First of all $TS(t_0)$ has a similar orthogonal vector as $TS(T_h)$ given by

$$n(t_0) = \begin{bmatrix} \dot{q_*}(t_0) \\ \ddot{q_*}(t_0) \end{bmatrix} \tag{114}$$

calculated at the start of the next step. For the second projection, this vector can be swapped with vector $m(T_h)$ from the previous projection, creating the modified form of (111)

$$n(t_0) \times \delta x(t_0) \tag{115}$$

where $\delta x(t_0) \in TS(t_0)$ is the new expression to be found. Following the same calculation as (112) yields the expression for $\tau_+$ and the new coordinates can thus be calculated as

$$\delta x(t_0) = \delta x(T_h) - \left( \frac{\delta x(T_h) n(t_0)}{||n(t_0)||^2} \right) n(t_0) \tag{116}$$

To initiate the next phase the starting configuration for the linear system must be given in transverse coordinates. This is done as in (104) with

$$\begin{bmatrix} \Delta I(t_0) \\ \Delta y(t_0) \\ \Delta \dot{y}(t_0) \end{bmatrix} = \mathcal{L}(t_0)\delta x(t_0) \tag{117}$$

and the next step of the gait can be initiated. These coordinates will then be used as the initial configuration for the next step of simulation.

## 8.3 Error Simulations

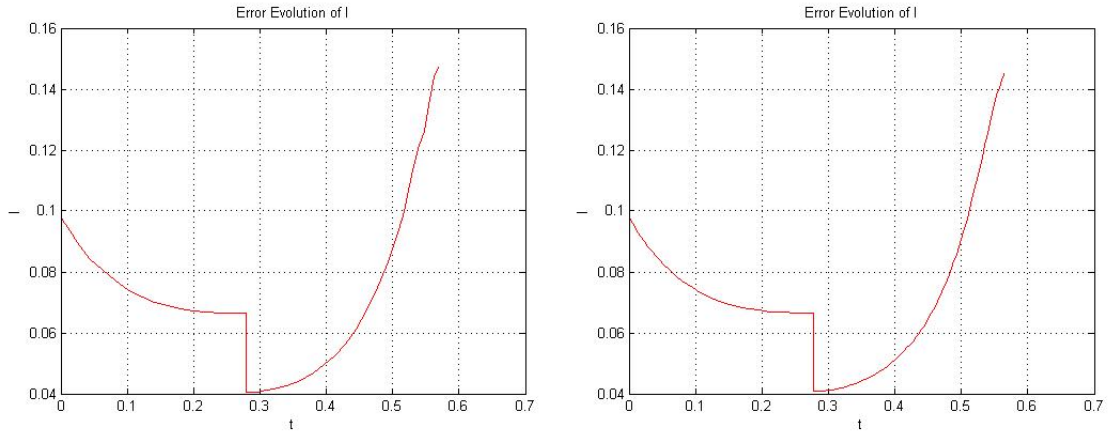
As mentioned in the end of section 8.2.2, the functionality of the transverse linearization method is based upon how well the linear dynamic system, given by (100)-(102), is able to represent the actual error dynamics of the full nonlinear system. The transverse linearization was after all implemented to simplify the creation of a control system, but for any controller to be successful, the linear dynamics must be able to adequately describe the nonlinear dynamics provided the solution is close to the nominal trajectory. To ensure this is the case, both the nonlinear and linear systems will be simulated with an initial error, and the transverse coordinates for each system will be compared. For the linear system the transverse coordinates will be updated as in section 8.2.2, with the update at each switch being given by section 8.2.3. Additionally, the parameter $v \in \mathbb{R}^{(n-1)}$ in (100) will be set to zero throughout the simulation as only the natural response of the linear system is desired. The real errors can likewise be calculated using (38) and (85) at each iteration of the integration of the 10-dimensional system, thereby calculating the errors directly. Several tests were carried out, initially for a system without errors.

### 8.3.1 Simulating without Errors

Without an initial error in the system, and without adding any errors during the motion, the system should function as normal and consequently the transverse coordinates should remain zero during the gait. The calculation for the linear system became quite simplistic for this scenario as (100) would forever remain zero throughout the motion of the first step. Likewise, the update at the first switch reinitialized the coordinates with all zero values for the second step, meaning the coordinates remained zero throughout the entire motion as expected. The same was experienced for the nonlinear system, though small errors due to integration tolerances of the simulation were visible. However, as these tolerances were kept very small, below $10^{-6}$, they would not influence the system in any way. Thus for both the linear and nonlinear error evolutions, the transverse coordinates will remain zero throughout an evolution of a functional gait.

### 8.3.2 Simulating with Errors

To simulate the system with errors, the initial velocity of the ankle-angle was increased by an incremental value, that is $\dot{\theta}_0 = \dot{\theta}_0^* + \delta_0$, with $\delta_0$ being of a sufficient size to induce errors without completely disrupting the gait. In fig(18) the evolution of the transverse coordinate $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$ is shown for both the nonlinear and linear systems for the first half of the gait. It can be seen that the incremental increase of $\dot{\theta}_0$ led to an initial error value of $I_0 \approx 0.1$. Throughout the first step, the two graphs are very similar with the linear system following closely to the real dynamics. As step 1 comes to an end, the linear system is updated with new coordinates as in 8.2.3. For the nonlinear system however, no update laws take place, but the error coordinates are instead simply calculated as normal at the start of the second step. Despite this, the change in coordinates is nearly



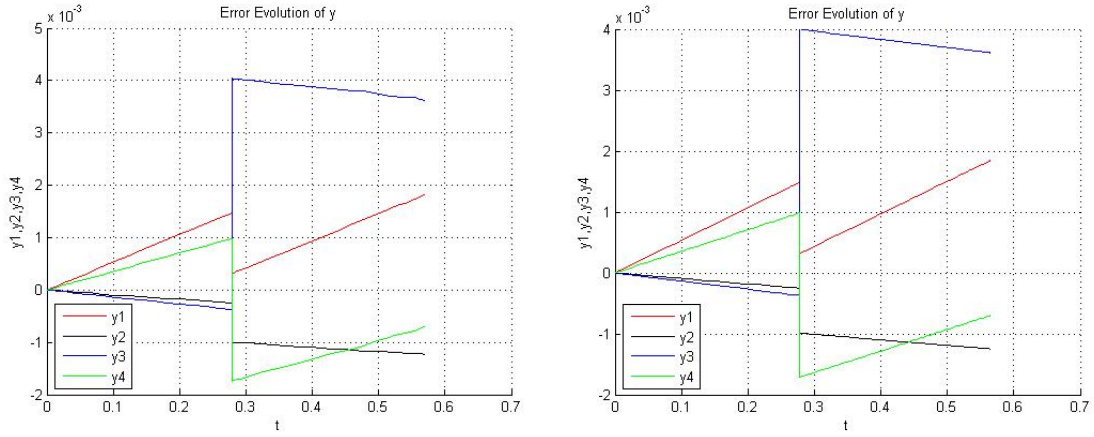

(a) Evolution of the nonlinear System

(b) Evolution of the linear system

Figure 18: The evolution of the transverse variable $I$ for the first two steps of the gait.

identical for the two systems, showing that the update laws for the linear system manage to correctly represent the change in the nonlinear system. One difference in the two systems is however the moment the switching occurs. The linear system was simulated using the nominal trajectory as a reference, and will thus perform the switch at the precalculated optimal time $t^1_{end} = 0.2772s$. The nonlinear system however, will perform the switch the moment the switching surface is reached, though with an increase in $\dot{\theta}_0$, the time it will take for $\dot{\theta}(t)$ to reach zero will be longer and thus the switching can be seen to occur at $t^1_{end} = 0.2788s$. Additionally, due to the now higher altitude of the SemiQuad, the impact with the ground in step 2 will also happen at a later time, which again increases the deviation. Now though these differences are quite small, it is still worth keeping in mind when later implementing a controller, especially considering the introduction of larger errors would further expand this gap. In any way, the coordinates will begin step 2 with the same values, and also throughout this step it is clear that the linear system follows the nonlinear one very closely. In fact, despite the slight deviation as mentioned, the two graphs are nearly identical, showing that the linear system is capable of representing the nonlinear system for the evolution of the first transverse coordinate.

Likewise in fig(19) the evolution of the transverse coordinates $y \in \mathbb{R}^4$ is shown. As the error was present in the angular velocity $\dot{\theta}_0$, the generalized coordinates $\{q_i,\ i = 2..5\}$ will start the gait in the optimal configuration. Their transverse coordinates will then initially be zero, but the error will cause them to increase throughout the step. The evolution of step 1 is almost identical for each coordinate, further proving the similarity of the two systems. Despite the time deviation also being present here, the two system will nevertheless start step 2 with the exact same values, showing once again that the switching is identical. The evolution of step 2 is then again identical for the two systems, showing that the $y$-coordinates stay close throughout the entire simulation.

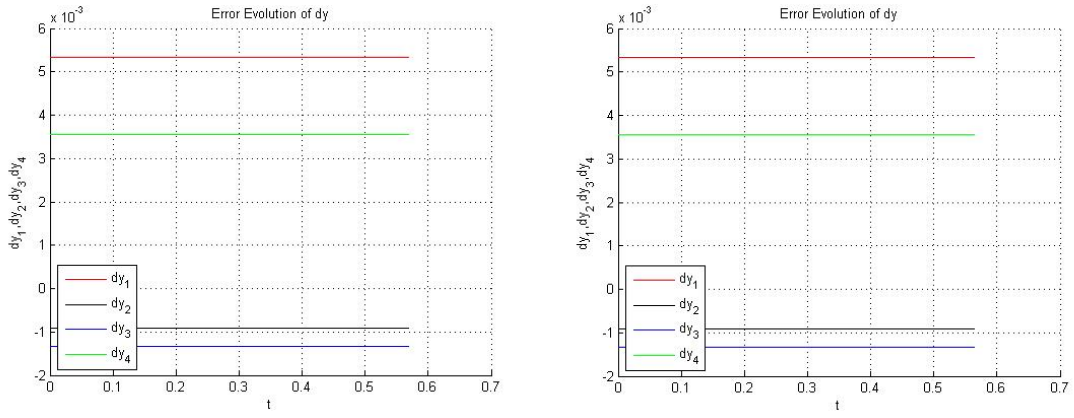



(a) Evolution of the nonlinear System (b) Evolution of the linear system

Figure 19: The evolution of the transverse variables $y$ for the first two steps of the gait

In fig(20) the final four transverse coordinates $\dot{y}$ are shown. It can be seen that the deviation in the initial angular velocity causes an initial deviation, that stays the same for both simulations throughout the gait. For the linear system this is trivial as it is the actuation $v \in \mathbb{R}^{(n-1)}$ that changes the transverse coordinates $\dot{y}_i$. This is however also present in the nonlinear system, making the two graphs more or less identical.

To sum up, the responses from the two system are exceedingly similar, which so far

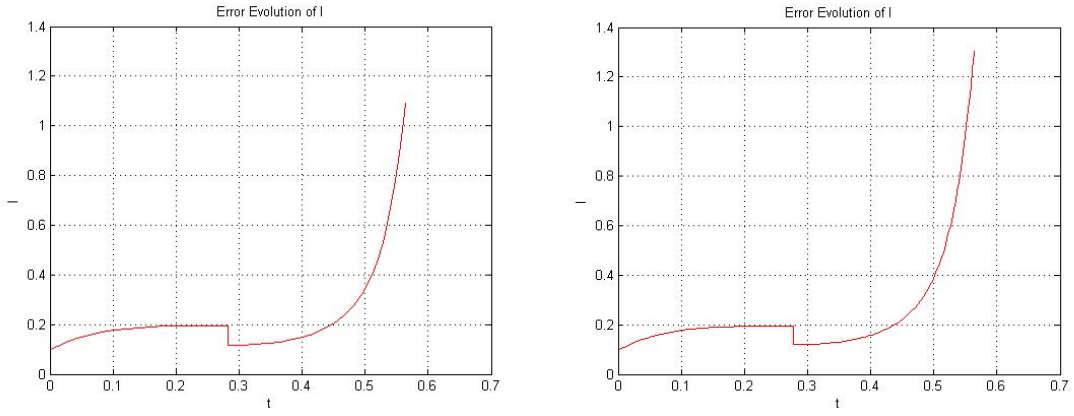



(a) Evolution of the nonlinear System (b) Evolution of the linear system

Figure 20: The evolution of the transverse variables $\dot{y}$ for the first two steps of the gait

shows that the linear system is representative for the nonlinear system at least for the natural dynamics. To also verify that this is the case when added actuation is present, $v(t)$ was set as predefined constant vector. For the linear system the vector $v$ was set directly and then integrated into the linear system, which also meant the matrix $B$ from

(100) needed to be calculated. For the nonlinear system, v was first set directly, and then the actual actuation $u \in \mathbb{R}^4$ was calculated using (92). In fig(21) the result of the simulation is shown once again for the transverse coordinate $I(\theta, \dot{\theta}, \theta_0, \dot{\theta}_0)$. The evolution of the two systems are once again nearly identical, showing that the two systems resemble each other also with added actuation. This shows that the linear system should be representative for the nonlinear system and thus, a controller made for the linear system should be able to function also for the nonlinear system.




(a) Evolution of the nonlinear System

(b) Evolution of the linear system

Figure 21: The evolution of the transverse variables $I \in \mathbb{R}$ for the first two steps of the gait with added actuation $v$.

# 9 Implementation of Control

In chapter 7 the SemiQuad system was simulated with erroneous initial configurations, demonstrating the robot's high susceptibility to errors. It was further shown that for only a slight deviation in the initial conditions, the SemiQuad would eventually fail to complete a full gait. Indeed it became clear that the introduction of any deviation would over time introduce larger errors, meaning a failure at some point was inevitable. The gait is therefore not exponentially stable, as any deviation will lead the SemiQuad away from the optimal and stable trajectory. This further limits the system from participating in any practical applications, as errors of at least a certain minimum size must be expected. This is one of the difficulties regarding orbital stability, as the walking motion is actually unstable, as explained in chapter 2 which states the gait is designed to be stable as a whole, though not locally stable at every instance in time. Having the SemiQuad rigorously following a predefined trajectory is easy in simulations, with correct initial conditions ensuring a correct gait at each cycle. A real life application however, could instead be compared to balancing on a tightrope, with any deviation potentially leading to a fall. It becomes obvious that a controller must be created, which, within a reasonable margin of error, would be able to bring the SemiQuad back to the optimal trajectory, and thus ensure exponential stability.

## 9.1 The Feedback Law

The creation of the controller is not a straightforward task, especially considering the nonlinearity of the system. However, with the introduction of the transverse linearization as in the previous section, the controller can instead be created for a linear system, which simplifies the problem by allowing the use of well researched linear control techniques. Thus, given the linear system (100) from the previous section, a $C^1$ smooth $T$-periodic matrix gain $K(\tau)$ can be introduced by the following relation[17][18][19]

$$v(t) = K(\tau)x_{\perp}(t), \quad K(\cdot) \in C^1\left[0, T_h\right] \tag{118}$$

where $v(t)$ is given by (100) and $x_{\perp}(t)$ are the transverse coordinates for the given time $t$. The matrix $K(\tau)$ can further be chosen so that it stabilizes the origin of the linear control system (100)-(102)[17]. However as established in the previous chapter, this is equivalent as saying there exists a state feedback controller given as[18]

$$v = f(\theta, \dot{\theta}, y, \dot{y}) \tag{119}$$

which makes the motion given by (94) orbitally exponentially stable in the new system (97)-(98). This is a very important feature as it states that a chosen functioning controller for the linear system, is in fact equivalent to a controller for the nonlinear system, provided the solution is close to the nominal trajectory. It is this feature that is the motivation behind the method of transverse linearization. Indeed if (119) constitutes a valid controller for the new system, then the relation (92) from section 8.2.2 will yield a valid controller $u(t)$ for the original system. This then simplifies the otherwise laborious

task of discovering a controller for the nonlinear system directly as the problem is reduced to discovering a suitable choice for $K(\tau) \in C^1[0, T_h]$.

## 9.2 Choosing the Linear Controller

Despite the fact that the controller $K(\cdot)$ can be created for a linear system does simplify the creation, it is still far from a trivial task to control a system such as the SemiQuad which has four interchanging steps divided by immediate jumps in all coordinates. The creation process will therefore be thoroughly studied in this section, in which first simplified control techniques will be proved to be insufficient before more advanced control techniques will be introduced. As a reference for the functionality of the controllers, the system will first be simulated without control. The results are shown below in fig(22), which shows the evolution of the transverse coordinates for all four steps of the gait with an initial error of $\dot{\theta}(0) = \dot{\theta}^*(0) + 0.05$.

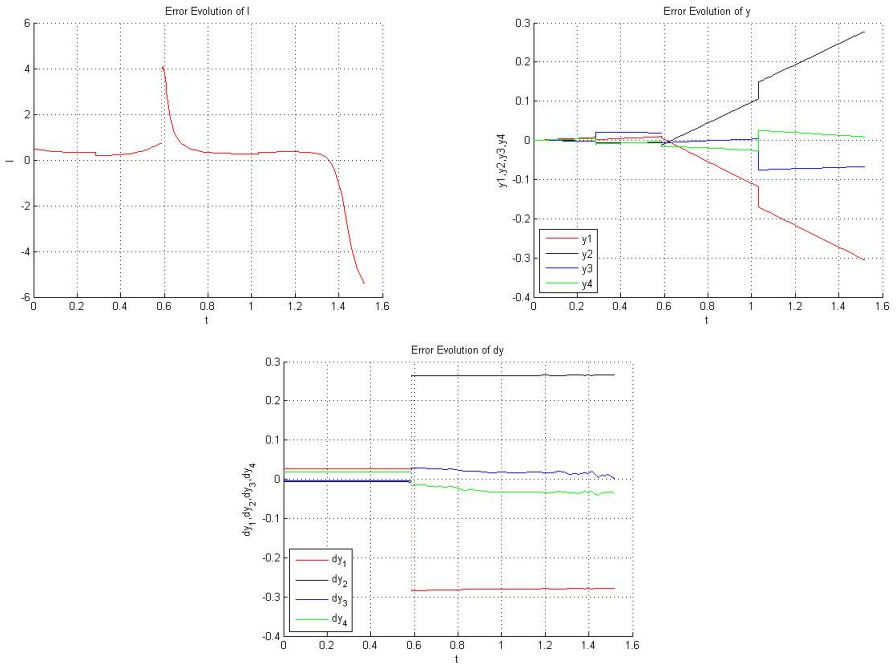



Figure 22: The Evolution of the Transverse Coordinates $[I, y, \dot{y}]$, given an initial error

It can be seen that the error grows consistently throughout the gait, with the impact in particular causing a sudden increase. As was shown in chapter 7, the SemiQuad would not be able to complete a second gait with this ending configuration. The goal of the controllers given in the following sections would then naturally be to improve upon this result, and preferably discover an ending configuration that is better than the initial configuration for all transverse coordinates.

### 9.2.1 Constant Gain matrix

As an introduction to the problem of designing a controller $K(\cdot)$, the implementation of a constant gain matrix will be attempted, that is

$$K(t_1) = K(t_2) = K, \quad t = t_0..T_h \tag{120}$$

where $T_h$ is the period of the entire gait. Now, although there is reason to believe that such a solution will be too simplistic to function properly, it nevertheless provides a suitable starting point for exploration of control. The first question that arises is naturally: For what choice of $K(\cdot)$ will the linear system, and subsequently the nonlinear system, be stabilized. With a constant gain matrix the linear system (100) becomes

$$\dot{X}_\perp(t) = (\mathcal{A}(t) - \mathcal{B}(t)K)\, X_\perp(t) \tag{121}$$

where relation (118) has been used. To find a suitable value for $K$ the matrices $\mathcal{A}(0)$ and $\mathcal{B}(0)$ are evaluated. Then the poles of the system

$$\dot{X}_\perp(t) = A_K X_\perp(t) \tag{122}$$

with

$$A_K = \mathcal{A}(0) - \mathcal{B}(0)K \tag{123}$$

can be placed in a suitable location by using matlab function *place*. Finding a functional $K$ then becomes a problem of trial and error. In fig(23) the results are shown for a poleplacement between $-6$ and $-10$, with imaginary pairs of roughly the same size. It can be seen that the immediate response is better than without control, with especially the response for the $y$ and $\dot{y}$ values being considerably lower at the end of the gait. However, though the same is somewhat true for the first transverse coordinate $I$ as well, the final value is still worse than the initial value, meaning the controller fails to bring all coordinates towards zero. This means the controller is somewhat capable of enforcing the virtual holonomic constraint, but that it is not capable of reducing the error of the passive-ankle-angle. Additionally, though the response is better than with no control, all errors can still be considered to be too large at the end of the gait. Therefore, when the same controller is simulated with a higher initial error, the SemiQuad is still unable to complete the gait, showing that the simple controller is not sufficient for the task.

### 9.2.2 The Linear Quadratic Regulator Approach

As the constant gain feedback controller was unable to ensure stability, another method with a non constant $K$ was attempted. The model was a simple version of the LQR approach, a well established tool for controlling linear systems. It consists in first defining a set of matrices[11]

$$G(t) \in \mathbb{R}^{9\times 9},\ g(t) \in \mathbb{R}^{9\times 4},\ \Gamma(t) \in \mathbb{R}^{4\times 4} \tag{124}$$

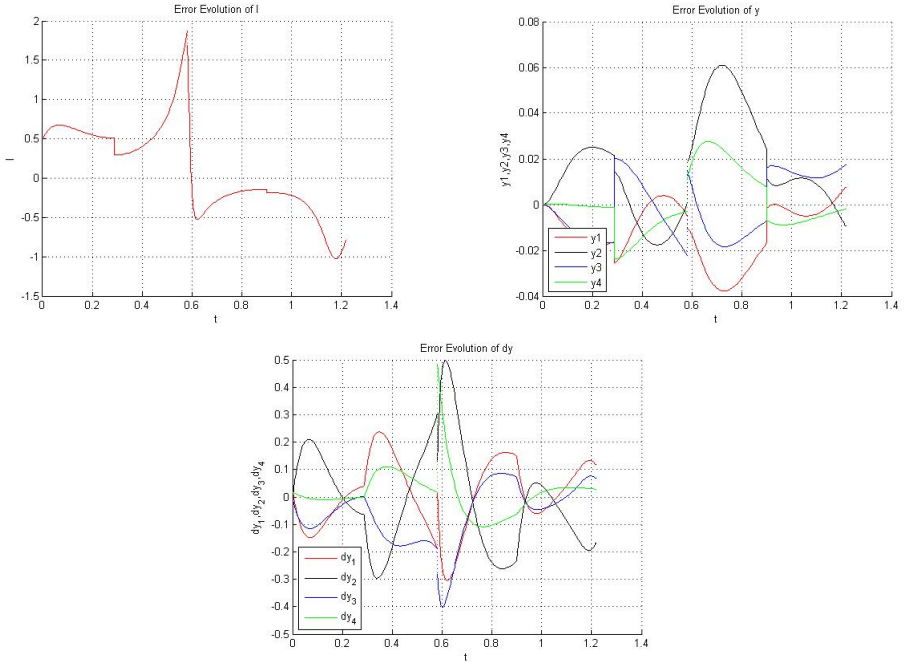



Figure 23: The Evolution of the Transverse Coordinates $[I, y, \dot{y}]$ using a constant gain controller.

with

$$G(t) = G(t)^T, \quad \Gamma(\cdot) > 0, \quad t \in [0, T_h] \tag{125}$$

which will function as weighting coefficients for a later optimization. The method is based on the solution of the Riccati equation $P(\cdot)$ which is given as

$$\begin{aligned} &\tfrac{d}{dt}P(t) + A(t)^T P(t) + A(t)P(t) + G(t) = \dots \\ &\cdots = [P(t)B(t) + g(t)]\,\Gamma^{-1}\,[P(t)B(t) + g(t)]^T \end{aligned} \tag{126}$$

where $A(t)$ and $B(t)$ are given by the linear system equation (100). Provided this equation is well defined for the interval $[0, T_h]$, the LQR controller is given by

$$v(t) = K(\tau)x_\perp(t) = -\Gamma(t)^{-1}\left[B(t)^T P(t) + g(t)^T\right] x_\perp(t) \tag{127}$$

which can be directly inserted in the simulation, provided $P(t)$ is known, to construct a periodic matrix $K(t) = K(t+T_h)$. This feedback law works by minimizing the following performance index

$$\int_0^{T_h} \begin{bmatrix} x_\perp(t) \\ v(t) \end{bmatrix}^T \begin{bmatrix} G(t) & g(t) \\ g(t)^T & \Gamma(t) \end{bmatrix} \begin{bmatrix} x_\perp(t) \\ v(t) \end{bmatrix} dt \tag{128}$$

along the solutions of the closed loop system. The choice of an LQR-feedback gain $K(\cdot)$ is particularly attractive, as it can yield a whole family of controllers (119) by tuning the matrix coefficients (124)[20]. The LQR attempted was as stated of a simple

nature, and thus the matlab function *care* was used to find a suitable matrix $K$ at each iteration of the simulation.

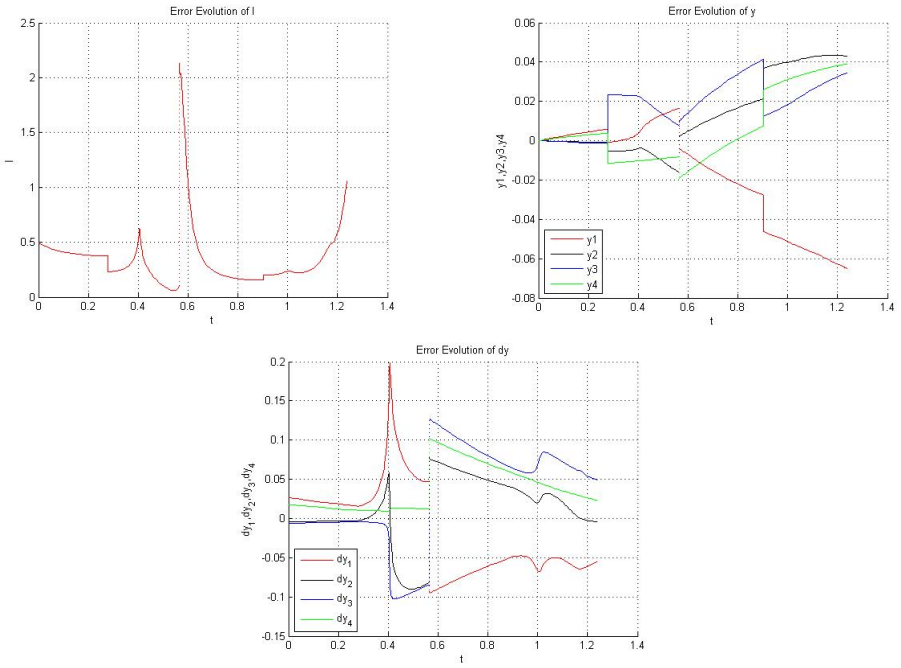



Figure 24: The Evolution of the Transverse Coordinates $[I, y, \dot{y}]$ with an LQR controller.

The results are shown in fig(24) for the same error as in the previous section. The results are similar to the constant gain controller, with the results being an improvement over simulating with no control. However, as with the constant gain controller, the results are not sufficiently satisfactory to ensure several functioning gaits. Though the LQR design has potential, a simple implementation such as this is not enough to control a hybrid system such as the movement of the SemiQuad. The problem lies in the fact that the controller only aims to control the separate movement phases of the SemiQuad, giving no regard to the discrete phases in between. Therefore, even with a systematic search for a functional set of matrices, there is no real guarantee that the controller will function with the discrete parts of the gait. It becomes clear that a more advanced controller is required, which will not only attempt to control the individual movement phases, but rather control the entirety of the gait as a whole.

## 9.3 SDP-Based Approximation of Stabilizing Solutions for PRDE

The creation of a controller able to stabilize the gait of the SemiQuad has been proven in the previous section to be a daunting task. It is for this reason I extend great thanks to professor *Sergei V. Gusev,* who created the final controller for this thesis. The theory behind the controller is available in [22], though the main points have been added in

section 9.3.1. The preliminary implementation and testing of the controller is available in section 9.3.2.

### 9.3.1 Constructing the Controller

The controller is based on finding stabilizing solutions of the periodic Riccati differential equations (PRDE), a problem that has been given much focus throughout the years. The differential matrix Riccati equation is given as[22]

$$\mathcal{R}(H,t) = 0, \quad \forall t \geq 0 \tag{129}$$

where

$$\mathcal{R}(H,t) = \frac{d}{dt}H(t) + A^T(t)H(t) + H(t)A(t) - H(t)B(t)R(t)^{-1}B^T(t)H(t) + Q(t) \tag{130}$$

The matrices $A(t)$ and $B(t)$ are as previously given by the linear system (100) while $Q(t) \in \mathbb{R}^{9\times 9}$ and $R(t) \in \mathbb{R}^{4\times 4}$ are positive definite matrices. In addition, all matrices are considered as continuous and T-periodic. Then, given a specific solution $H = H^+$ of (129), if the resulting linear system (121) with

$$K = K(t) = -R^{-1}(t)B^T(t)H^+(t) \tag{131}$$

is $L_2$-integrable on $[0, +\inf)$[22], $H^+$ will be known as a stabilizing solution of the PRDE (129). The problem at hand is thus that of finding $H$ so the resulting system becomes stable. This is summarized in the following proposition[22]

***Proposition 1:*** *Suppose that A, B, Q and R are continuous T-periodic matrix-functions, the pair $(A, B)$ stabilizable and $Q(t) > 0$, $R(t) > 0$, $\forall t \in [0, T_h]$. Then, there exists a T-periodic stabilizing solution $H^+$ of (129). Moreover, any T-periodic solution H of the Riccati inequality*

$$\mathcal{R}(H,t) \geq 0, \quad \forall t \in [0, T_h], \tag{132}$$

*satisfies the inequality*

$$H(t) \leq H^+(t) \quad \forall t \in [0, T_h] \tag{133}$$

*i.e. $H^+(t) - H(t)$ is a positive semidefinite matrix $\forall t \in [0, T_h]$.*

***Remark 1:*** *It immediately follows from Proposition 1 that the stabilizing solution is unique.*

The proposition thus underlines the requirements of the linear system for a solution $H^+$ to exist, a solution which is further shown to be unique. The inequality (132) can further be decomposed into a linear matrix inequality via the Schur complement[22]

$$\mathcal{L}(H,t) \geq 0, \quad \forall t \in [0, T_h] \tag{134}$$

where

$$\mathcal{L}(H,t) = \begin{bmatrix} \frac{d}{dt}H(t) + A^T(t)H(t) + H(t)A(t) - H(t) + Q(t) & H(t)B(t) \\ B^T(t)H(t) & R(t) \end{bmatrix} \tag{135}$$

The final task of finding the optimal and unique solution $H^+$ of (129), referred to as problem $\mathcal{P}$, can then be stated as the following optimization problem[22]

$$\mathcal{F}(H) = \int_0^{T_h} tr\left(H(T)\right) dt \tag{136}$$

with a single unique solution existing for $H^+$. The above problem cannot be calculated directly however as it is infinite dimensional and a solution can therefore not be found. The problem must therefore be simplified, which is possible to achieve by searching for an approximation of $H^+$ instead.
To achieve an approximation of the stabilizing solution $H^+$, the matrix $H$ can first be rewritten as matrix polynomial of order $2M$, that is

$$H = \sum_{-M}^{M} e^{\mathbf{i}\frac{2\pi kt}{T}} F_j(H) \tag{137}$$

with $F_j(H)$ being symmetric complex matrices of the same dimension. Then the vector space of all polynomials of degree at most $M$ are set as $\mathcal{H}_M$. Further, inequality (134) can be rewritten as[22]

$$\mathcal{L}(H,t_j) \geq 0, \quad t_j = (j-1)\frac{T}{N}, \; j = 1..N \tag{138}$$

where as before $T_h$ is the total time of the period and $N$ is the number of samples. To ensure the boundedness on the polynomial $H$, a second set of equations can be introduced[22]

$$-dI_n \leq H(t_j) \leq dI_n, \quad j = 1..N \tag{139}$$

where $d > 0$ is a constant of sufficient size. With this in place the optimization problem $\mathcal{P}$ can be reformulated as $\mathcal{P}_{M,N}$, which for any $N$ and $M$ constitute a finite dimensional SDP problem. The optimization is then given by (136) as before, though it is now limited to only include the matrices $H \in \mathcal{H}_M$ which also satisfy (138) and (139), thus reducing the size of the problem. The solution of this optimization is then given as $H^+_{M,N}$, and though $H^+_{M,N} \neq H^+$, it is still a valid approximation provided the ratio $\frac{N}{M}$ is sufficiently large. This is summarized as[22]

$$||H^+ - H^+_{M,N}||_C \to 0, \quad M \to \infty, \quad \frac{N}{M^3} \to \infty \tag{140}$$

Thus, the error of approximation tends towards zero for suitably high values of $N$ and $M$, making the optimization based on the SDP approach valid.

### 9.3.2 Implementing the SDP-based PRDE controller

The controller described in the previous section was created for a sampled set of matrices given as

$$A_i(1..n),\ B_i(1..n),\ T_i(1..n),\ L_i, \quad i = 1..4 \tag{141}$$

where $A_i$ and $B_i$ are the matrices $A(t)$ and $B(t)$ for step $i$, sampled at time instants $t_j \in T_i$, while the matrix $L_i$ is the update matrix for the transverse coordinates from step $i$ to step $i+1$. The sample size is given by $n$ and for the SemiQuad system it was set to $n = 101$. From the sampled linear system a controller $K(t_j)_i$ was created for each step of the gait. Then for the simulation of the actual nonlinear system, the precalculated $K$-matrices could be extracted and the system could be simulated with control as in section 9.2. As the matrices were sampled at a given set of time instances $t_j$, the actual $K$ matrix would for each iteration of the simulation be made up of the $K(t_j)$-matrices closest to the actual time $t$. The results are shown in fig(25) for the same errors as in the previous simulations.

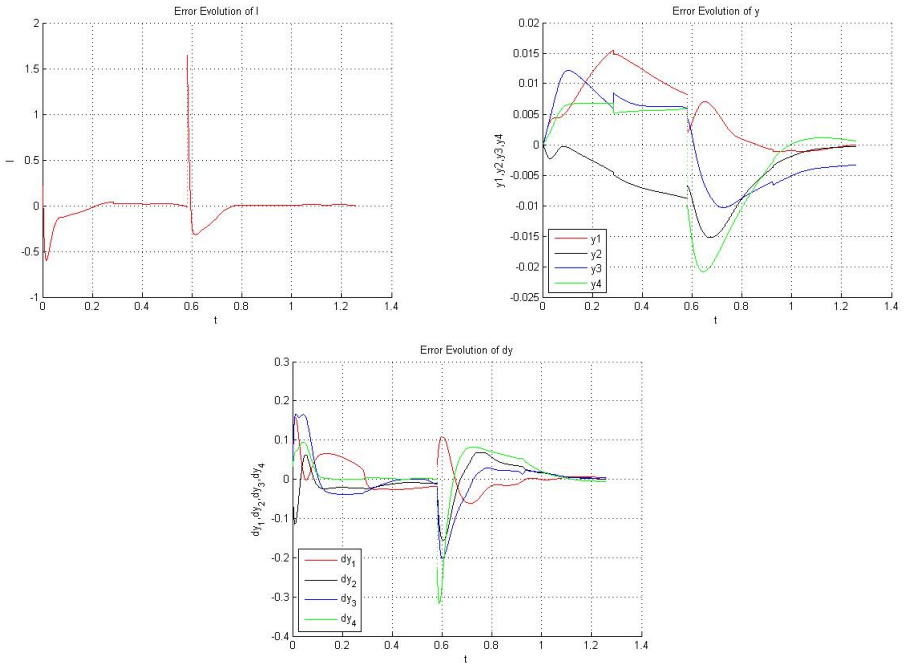



Figure 25: The Evolution of the Transverse Coordinates $[I, y, \dot{y}]$ using SDP-based control.

The results are a drastic improvement over both the natural evolution of the system and the other controllers, with all values of the transverse coordinates being close to zero at the end of the gait. Additionally, when continuing the simulation beyond the first completed gait, the SemiQuad is not only able to complete a second full step, but the ending configuration after the second gait is even better than it was after the first.

Thus, after the preliminary tests, it seems the controller is able to ensure a stable gait for the SemiQuad, at least for errors of this magnitude. A full study of the system with this controller implemented must however be carried out. This will be handled in the next and final chapter.

# 10 Simulating With Control

With the notion of transverse linearization explored as in chapter 8 and control implemented as in chapter 9, the work of testing the robustness of the gait can begin. The final goal will be to show that the gait is both versatile with respect to speed and energy efficiency, while also being adequately robust with respect to errors. Given the already functional gait, it is of interest to explore for what region of errors the SemiQuad will still be able to perform the gait without failing. Hopefully this region of error will be large enough to allow an implementation of the gait to function in normal operating conditions. Before that however, a complete study of the implementation of the controller and the integration with the 10-dimensional simulation is added.

## 10.1 The Controller Algorithm

The controller that is to be implemented is that of the SDP based approximation of solutions for the PRDE, as discussed in section 9.3. The system will be simulated for all generalized coordinates and velocities, that is

$$x = \left[\theta, q_2, \ldots, q_5, \dot{\theta}, \dot{q}_2, \ldots, \dot{q}_5\right]. \tag{142}$$

The simulation will then carried out using the matlab integration function *ODE45*. At the start of each simulation all 10 coordinates are extracted. As the error was to be described by the use of transverse coordinates, these would first have to be calculated directly.
The first transverse coordinate, $I(\theta_0^*, \dot{\theta}_0^*, \theta, \dot{\theta})$, are calculated as in (38). This integral gives a measure of how well the nominal trajectory given by the solution of the $\alpha - \beta - \gamma$ equation is being followed. In this sense the values of $[\theta_0^*, \dot{\theta}_0^*]$ are here the optimal starting values for the current step of the simulation, while $[\theta, \dot{\theta}]$ are the real values taken from the coordinates in the simulation, that is $\theta = x(1),\ \dot{\theta} = x(6)$. Errors will then occur if the real values differ from the nominal values, such as when $\dot{\theta}_0 \neq \dot{\theta}_0^*$, as was illustrated in section 8.3.
The remaining transverse coordinates are calculated by the modified equation (85), given as

$$y_{i-1} = q_i - \phi_i(\theta), \quad \dot{y}_{i-1} = \dot{q}_i - \phi_i'(\theta)\dot{\theta}, \quad i = 2..5 \tag{143}$$

As mentioned, these $(2n-2)$ error variables gives a direct measurement to how well the virtual holonomic constraints are being followed. Note however, that abiding by the virtual constraints does not automatically mean the actual generalized coordinates and velocities are correct. Indeed, an erroneous value in $\dot{\theta}$, will result in increased values of $\dot{y}_i,\ i = 2..5$, despite the values of $\dot{q}_i,\ i = 2..5$ being correct.
With the transverse coordinates calculated, a functional measure for the degree of error has been found. Further, it was shown in chapter 8 that the evolution of these coordinates could without loss of generality be linearized to form the system given by 100. This again gives rise to the possibility of introducing a linear controller given by 118, which should be able to control the linear system, and thus also the nonlinear system. The gain matrix

$K$ was in fact calculated prior to the simulation for a number of time steps. Given a time $t$, an appropriate matrix $K_t$ could then be interpolated from the set of matrices $K(j)_i,\ j = 1..n$ during step $i$ for a given resolution $n$. With $v \in \mathbb{R}^4$ known, the relation given by (92) could be used to calculate the actual actuation $u \in \mathbb{R}^4$ for the nonlinear system. With the new actuation the nonlinear system can be updated by calculating the angular acceleration as in (87), which can then together with $\dot{q}$ be given back to the *ODE45* function. The starting coordinates for the next step are then calculated as normal depending on the type of switch that's occurring. The process is summed up in the following pseudo code:

$K_i =$ **calculateSDPapprox(A,B,T,L)**
**for step** $= 1:4$ **do**
$[q_0, \dot{q}_0] \rightarrow$ **Integrate10DimSystem**
$q,\ \dot{q},\ t \leftarrow ODE45$
$x_\perp(1) = I(\theta_0^*, \dot{\theta}_0^*, \theta, \dot{\theta})$
$x_\perp(2:5) = q_i - \phi_i(\theta)$
$x_\perp(6:9) = \dot{q}_i - \phi_i^{'}(\theta)\dot{\theta}$
$K = interPolateK(K_i, t)$
$v = Kx_\perp$
$u = \mathcal{N}^{-1}(\theta, y)\left[v - \mathcal{R}(\theta, \dot{\theta}, y, \dot{y})\right]$
$\ddot{q} = \mathcal{M}^{-1}(q)\left[-\mathcal{C}(q)\dot{q} - \mathcal{G}(q) + \mathcal{B}u\right]$
$\dot{q}, \ddot{q} \rightarrow ODE45$

$[q_0, \dot{q}_0] \leftarrow$ **updateCoordinates**

note that the controller was made specifically for the response of the gait that was studied in the previous chapters, and it is therefore doubtful that it will work well for other gaits. Further, the matrices of the linear system, as well as the matrices $K_i$ of the controller are constant matrices. It is this that allows the creation of a controller prior to the simulations being run. This in turn however, means the force of the actuation control from one simulation to the next is entirely dependent on the size of the transverse coordinates.

## 10.2 The Error Tolerance of the System

In section 9.3.2, the final controller was tested for the first time with a suitable erroneous initial configuration. The results, as shown in fig(25), demonstrated that the SemiQuad was able to complete a series of steps despite the error, something that had been shown to be impossible without control. The question that naturally arises is then for what degree of errors will the SemiQuad still be able to accomplish several successful gaits, or consequently, at what point will the errors be too large for the SemiQuad to function. Therefore, this section will attempt to discover a suitable band on the errors by first simulating the system with increasing deviations in the initial configurations of single

variables. Then combinations of errors will be simulated to assess if the controller is capable of stabilizing gaits with more than one major deviation. An exact bound will most likely not be found, as that would require running the simulation for a near endless amount of error situations, a simulation which for a healthy resolution could take a very long time to calculate. However, an estimate on the size of the various errors should suffice to gauge the functionality of the controller for the SemiQuad system.

### 10.2.1 Increasing Deviations of $\dot{\theta}_0$

The first transverse coordinate was, as shown in the previous chapter, one of the hardest coefficients to bring to zero throughout a gait. This is due to the fact that the value of this error coefficient stems from the value of both $\theta$ and $\dot{\theta}$, which are the generalized coordinate and velocity of the passive ankle-angle of the system. As the ankle is passive, it does not possess a subsequent motor to immediately counter any deviation, but rather the system as a whole must move to induce change in the ankle through changing the center of gravity, see chapter 2. To evaluate the controller, the system was again simulated with increasing deviations of $\dot{\theta}_0$ as in section 7.2.1. In fig(26), the response of simulating with the same magnitude of errors is shown with control implemented. The

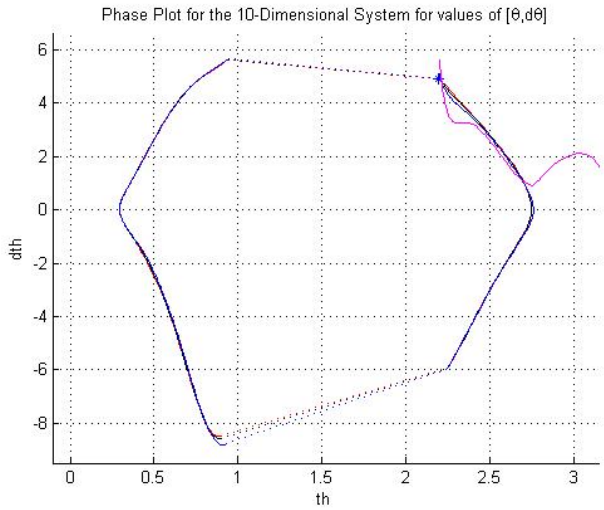


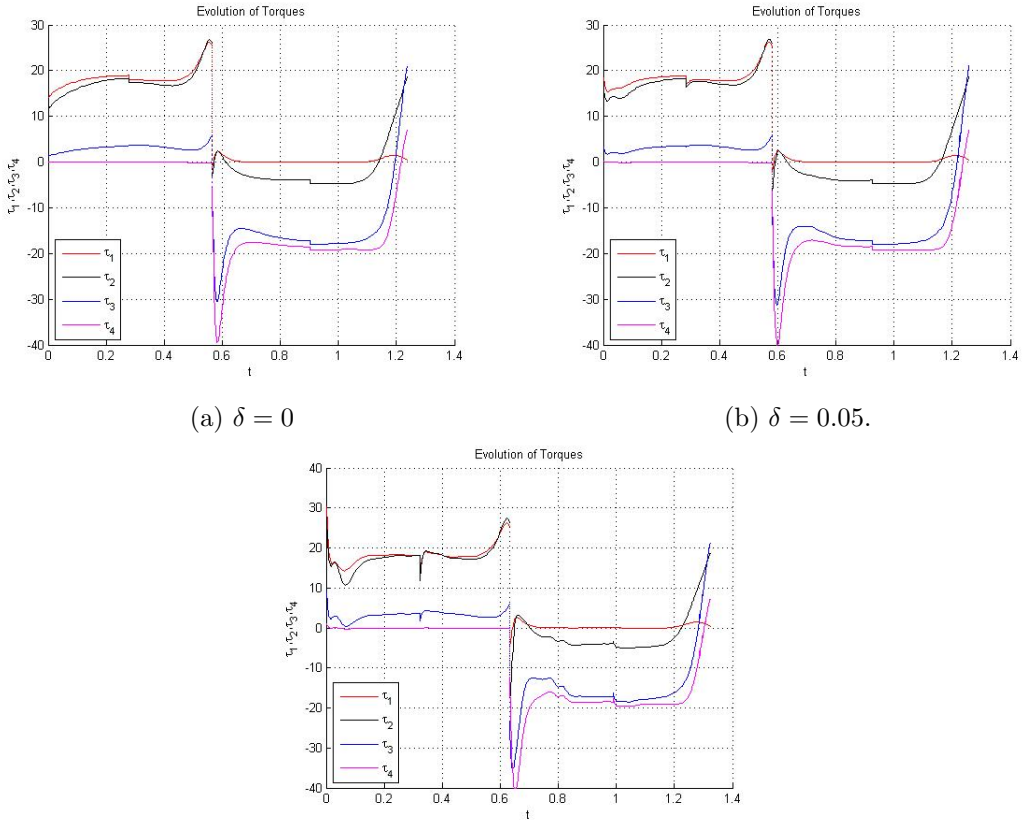

Figure 26: The Phase Portraits $\left[\theta, \dot{\theta}\right]$ for several simulations of the SemiQuad system with control implemented. The whole lines are the evolution of the coordinates, while the stippled lines are the instantaneous impact phases.

phase portraits show the evolution of all simulations with a nominal gait included as a reference. It can be seen that all but one simulation manage to complete the full gait, with the exception being the largest error of $\dot{\theta}_0 = \dot{\theta}_0^* + 0.7$. The error proves too great and the SemiQuad still falls backwards due to the velocity of the ankle-angle not reaching zero in time. However, the response provides a stark contrast to the simulation without control as shown in fig(9), with even the erroneous trajectory following the nominal

trajectory for a short stretch of time. Indeed, the response of the working simulations are not only shown to complete the gait, but their final configuration is very close to the optimal ending values $[\theta_e^*, \dot{\theta}_e^*]$, as seen in the second impact. This is particularly remarkable for the second highest error of $\dot{\theta}_0 = \dot{\theta}_0^* + 0.2$, as this was earlier shown to not be able to complete step 3. A quick estimate then places the highest error $\delta$ possible as a value between the two highest errors, that is

$$0.2 < \delta_{max}^{+} < 0.7 \tag{144}$$

However, only looking at the capabilities of the controller is not enough as also the physical limitations of the SemiQuad must be taken into consideration. Therefore, in fig(27) the evolution of the motor torques are shown for three simulations. In the second




(a) $\delta = 0$

(b) $\delta = 0.05$.

(c) $\delta = 0.2$

Figure 27: The Evolution of Torques $\tau_i$. $i = 1..4$ given an initial error in $\dot{\theta}_0$

plot, where the error is $\dot{\theta}_0 = \dot{\theta}_0^* + 0.05$, it can be seen that the response changes slightly with control implemented. The response still stays within the actuation limit however, which is $|\tau_{max}| = 40$, meaning it is a legal gait. The same can not be said for the third plot where the error is $\delta = 0.2$. Though the system for the most part stays within the

bounds of the actuators, there is one slight moment where it goes under $-40$ at the start of the third step, meaning the system is not realizable. This is an important result, as it shows that the limitation in actuation will prevent the controller from functioning as intended long before the controller fails due to the errors being too large. As the problem seems to only occur at a single point during the step, the simulation is again carried out, but now with a saturation limit in the actuation implemented. Unfortunately, with the saturation in place, the simulation fails to complete the gait, as the saturated torques prove insufficient to control the system. This in turn means the estimate of the highest error the system is able to control must be decreased, that is

$$0.05 < \delta^{+}_{max} < 0.2 \tag{145}$$

which should yield a sufficient image for the capabilities of the controller in handling high values of $\dot{\theta}_0$. The same simulations could then be done with a negative error instead. After simulating several times, an error estimate was found as

$$\delta^{-}_{max} \approx -0.2 \tag{146}$$

with the evolution of torques being shown in fig(28). It can be seen that the torques do not pose a similar problem for this instance, as the values of the torques are actually smaller in the critical instants. The study of torques have so far proven to be of quite some importance, as it becomes clear that the functionality of the controller is directly linked to having sufficient torque available to correct any error. In this case the nominal

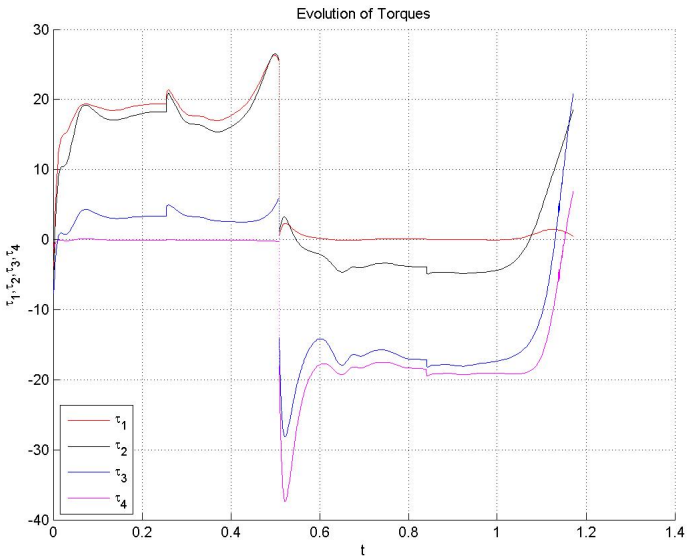



Figure 28: The Evolution of Torques $\tau_i$. $i = 1..4$ given an initial error $\delta = -0.2$.

trajectory seems to performing quite well, as its torque values usually stay far below the

saturation limits. This is in turn due to the fact that the gait was created to be energy economic, meaning lower torques gave a better result in the optimization. Thus, keeping this in mind, new gaits could be created which would not only aim to lower energy usage but also aim to avoid spikes in the torque values. This should in turn lead to a more robust gait capable of handling greater errors.

### 10.2.2 Increasing Deviations of Initial Body Configuration

The previous section showed the direct response of the controller for several erroneous starting configuration with $\dot{\theta}_0 \neq \dot{\theta}_0^*$. The controller will now instead be tested for erroneous starting configurations of the generalized coordinates, that is

$$\theta(0) \neq \theta^*(0), \quad q_i(0) \neq q_i^*(0), \quad i = 2..5 \tag{147}$$

This entails the SemiQuad starting its gait with its body in a shifted position, something which was seen to cause the system to fail during the gait in section 7.2.2. The system was initially simulated for the same errors that were tested there. In fig(29) the response

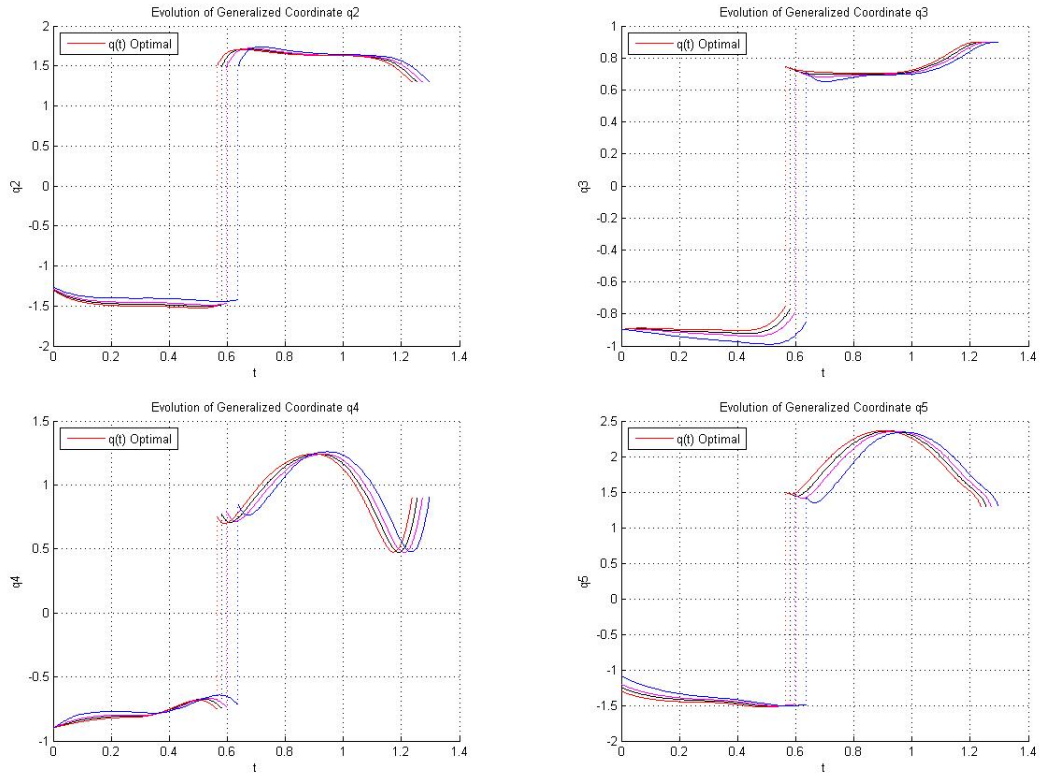



Figure 29: Several Evolution of the Generalized Coordinates $q_i$. $i = 2..5$ given an initial error in $q_2$ and $q_5$ simulated with control.

of several simulations are shown, with increasing errors in $q_2$ and $q_5$, with the nominal

values included for reference and the error values given as

$$q_2(0) = q_2^*(0) + \delta_1, \quad q_5(0) = q_5^*(0) + \delta_2, \quad \delta_1 = \{0.01, 0.02, 0.04\}, \quad \delta_2 = \{0.055, 0.11, 0.22\}$$

It can be seen that for all errors, the controller manages to not only complete the gait, but also end up in a better final configuration. In fact, for the last two simulations, the controller proposed higher torques than the limit, but it was still able to ensure functioning gaits for both instances. The noticeable difference is the time instants of the switches, with higher errors leading to delayed switching. This is due to the body configuration having been moved backwards, thus making it easier for the SemiQuad to lean backwards at the start of the movement. However, as this is not accounted for, the body will move further back than intended during the motion, meaning the time of impact will occur later. Despite this, the effectiveness of the controller is quite visible, as even though the time instants vary, the arc each coordinate follows is almost identical, meaning the nominal trajectory is being followed. The same results are shown for the remaining generalized coordinates with a larger set of errors

$$\theta(0) = \theta^*(0) + 0.2, \quad q_3(0) = q_3^*(0) - 0.1980, \quad q_4(0) = q_4^*(0) - 0.16$$

Also here the controller manage to stabilize the gait, though saturation does occur in multiple locations. In fig(30) the evolution of $[\theta, \dot{\theta}]$ is shown, together with their respective nominal values. Despite the large initial offset, the system can be seen to quickly reach the nominal trajectory, which it then follows closely throughout the gait before finishing in a near optimal final configuration. Now, due to saturation occurring,

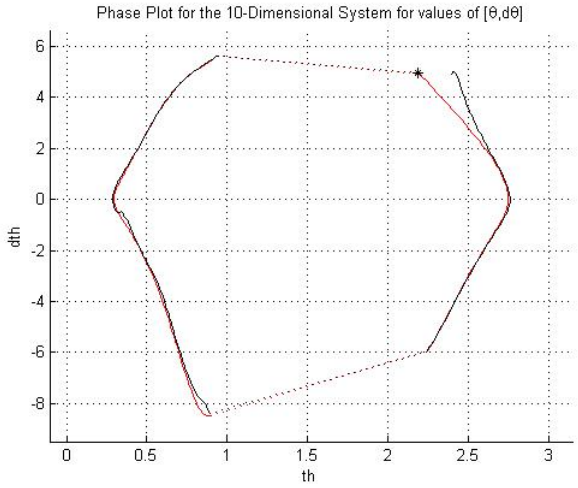



Figure 30: The Phase Potrait of $[\theta, \dot{\theta}]$ given an erroneous initial configuration.

there is reason to believe the gaits will eventually fail if the calculated torques grows much larger than their saturated values. Despite this, the SemiQuad seems quite capable of starting the gait in various erroneous configurations, though that might be due to the fact

that all errors tested so far have been of the same type. Indeed, the errors have caused the initial configuration of the SemiQuad to be tilted backwards, meaning the robot has been tilted in the intended direction of motion. Therefore, the first configuration was simulated once more, but this time with reversed errors, that is

$$q_2(0) = q_2^*(0) - 0.04, \quad q_5(0) = q_5^*(0) - 0.22$$

with the new body configuration as shown in fig(31). Despite the errors being of the same magnitude, the SemiQuad now fails to complete the gait. This shows that a front tilted configuration can be considered a more severe error than a backwards tilt, despite the error values, and thereby the transverse coordinates, being of the same magnitude. This is due to the extra work demanded of the SemiQuad to lift its front leg when the center of mass has been shifted in the opposite direction. This becomes clear when looking at fig(30). Having an initial error of $\theta_0 = 2.4$ does not hinder the SemiQuad too much as it was intended to reach this value during step 1, which then makes this error a more trivial case than say $\theta_0 = 2.0$.

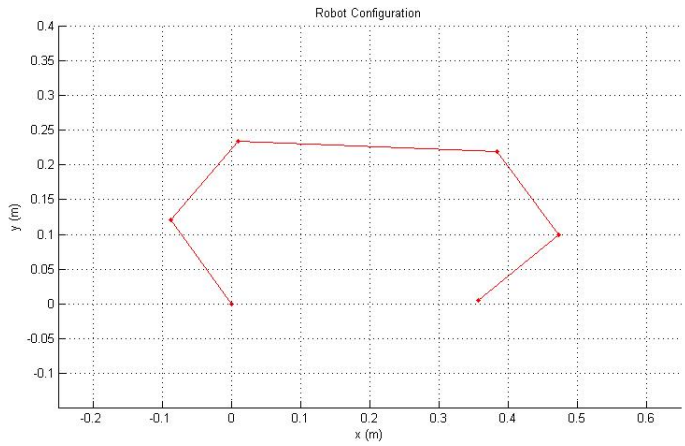



Figure 31: The configuration of the SemiQuad, given an initial error in $q_2$ and $q_5$.

In any way, the SemiQuad seems adequately robust to handle errors in its configuration, as long as they don't becomes too large. This should be a reasonable assumption given normal operating conditions, as large changes in configuration would most likely be the result of external forces on the system, or other large errors in a previous gait yielding a highly erroneous starting configuration. Therefore, it is reasonable to assume the SemiQuad should remain operational within normal operating conditions.

### 10.2.3 Increasing Deviations of the Angular Velocities

The only initial errors yet to be tested, are the deviations in the angular velocities of the generalized coordinates. It was stated in section 10.2.1 that the value of $\dot{\theta}$ was especially susceptible to errors due to the impacts with the ground and the resulting instantaneous

changes. This holds true also for the values of $\dot{q}_i,\ i = 2..5$, especially when considering they change more rapidly than $\dot{\theta}$ at the moment of impact. As an initial test, the system is as in the previous section simulated for a set of erroneous initial values given as

$$\dot{q}_2(0) = \dot{q}_2^*(0) + \delta_1, \quad \delta_1 = \{-0.05, -0.1, -0.15\}$$

$$\dot{q}_3(0) = \dot{q}_3^*(0) + \delta_2, \quad \delta_2 = \{-0.05, -0.1, -0.15\}$$

The evolution of all generalized velocities are shown in fig(32). Though the errors causes an initial offset, all trajectories appear to have very similar arcs. The response

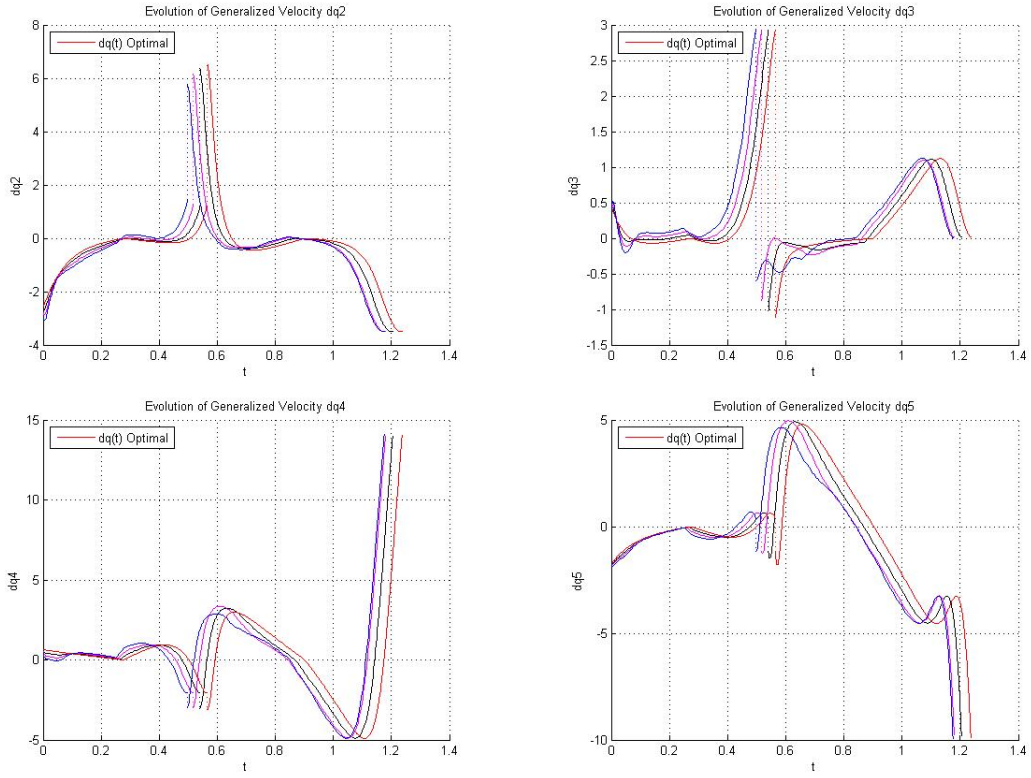



Figure 32: Several Evolution of the Generalized Velocites $\dot{q}_i$. $i = 2..5$ given an initial error in $\dot{q}_2$ and $\dot{q}_3$ simulated with control.

is in fact quite similar to those seen in fig(29), with the main difference between the simulations again being the time instants where switching occurs. As the given error works against the backwards motion, the erroneous simulations can be seen to reach the switching instances earlier than the optimal time. The values of the generalized velocities are still nearly the same for similar moments during their respective gaits however, further proving the functionality of the controller. When simulating with higher initial errors than those shown the gait would eventually malfunction, showing a similar error tolerance as for the other coordinates. It can be mentioned that this error tolerance does increase for the generalized velocities further away from stance leg, that is $\dot{q}_4$ and

$\dot{q}_5$, which where seen to work for error values of $\delta > 0.5$. This is due to the change in velocities in this region inducing a lower change on the velocity of the ankle angle due a smaller change in the movement of the center of mass. For simulations this can still be considered quite important however, as an error in $\dot{q}_2$ will induce a larger change in the system than an error in $\dot{q}_5$, despite their respective transverse coordinates having the same value. It might then be considered to include this knowledge in the control of the SemiQuad, with the transverse coordinates of coefficients with a low error tolerance being weighted heavier than those with high error tolerance.

### 10.2.4 Increasing Deviations in All Coordinates

The final entry in this section will be a short study into highly erroneous starting configurations, with most of the coordinates deviating in some way. Unfortunately for the SemiQuad system, a scenario such as this would probably be more likely than having individual errors, as an error in the system usually means several coordinates will be affected. The simulation was carried out by giving each coordinate its own error $\delta_i$, with the values of the errors initially set as

$$\delta_{tot} = \{0.1,\ -0.1,\ 0.05,\ -0.12,\ 0.105,\ 0.08,\ 0.05,\ 0.05,\ -0.2,\ -0.2\}$$

where $\delta_1$ is the deviation in the optimal value of $\theta$ and $\delta_6$ the deviation in the optimal value of $\dot{\theta}$ etc. In fig(33) the phase portrait of $[\theta, \dot{\theta}]$ is again shown with the nominal trajectory as a reference.

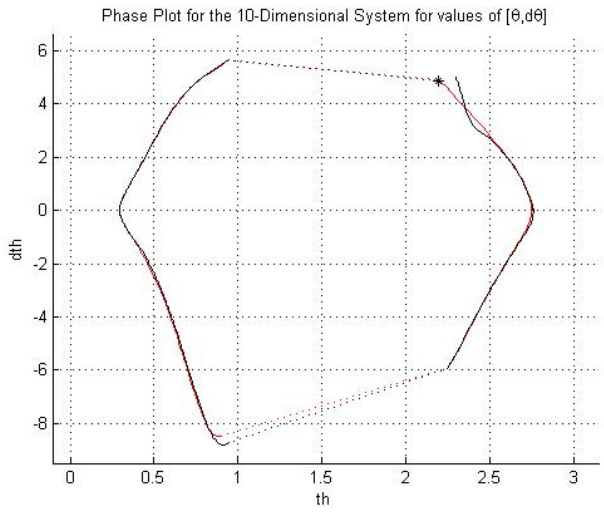



Figure 33: The Phase Portraits $\left[\theta, \dot{\theta}\right]$ for a trajectory with several initial errors. The whole lines are the evolution of the coordinates, while the stippled lines are the instantaneous impact phases.

It can be seen that despite the number of errors the controller is quick to reach the nominal trajectory. However, it can further be seen that the remaining errors in the

system causes an additional deviation from the trajectory during the first impact. The system recovers however, and in the end reaches a near optimal final configuration. As the controller now had several errors to contend with it is interesting to view the individual torque values. This is shown in fig(34), which shows the the evolution of the torques with the nominal values included as a reference. The effect of the controller

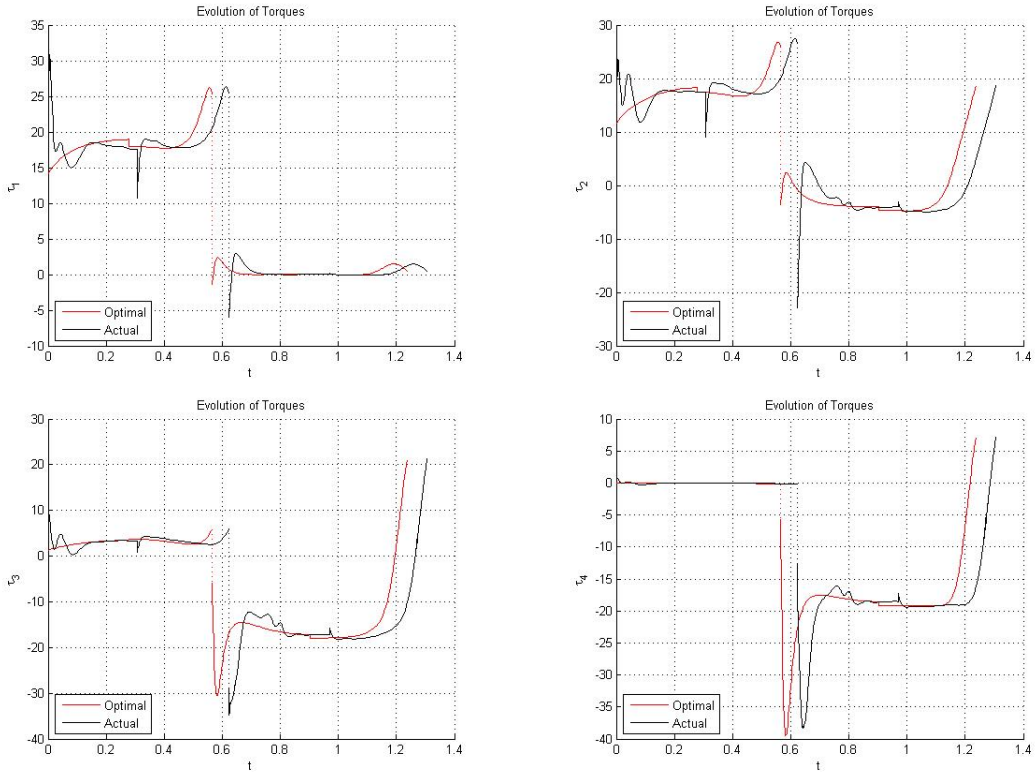



Figure 34: Evolutions of the torque values $\tau_i, i = 1..4$, given initial errors in all coordinates simulated with control.

is quite visible, with the values switching back and forth on both sides of the nominal trajectories. Further, the difference in switching time is also quite visible, though the responses can again be seen to follow the same general arcs. Increasing the errors would, as in the other instances, eventually lead to the failure of the gait.
In summary, the SemiQuad has proven capable of handling most instances of errors, with a relatively high error tolerance. In addition, for the instances where the controller is working, the final configurations can be seen to be very close to the optimal final configurations, meaning the next iteration of the gait would yield even better results. This in turn means the SemiQuad is capable of performing several gaits, with the controller at all times maintaining a close proximity to the nominal trajectory, effectively creating an exponentially stable periodic walking motion. Additionally it must be added, that this was indeed the first gait to receive a functioning controller, meaning it was created

before the capabilities of the controller was taken into consideration. After all, it is obvious that there exists gaits that would prove to have a higher error tolerance, and the creation of these will be simplified now that the functionality of the controller has been tested for a prototype.

## 10.3 Simulating the SemiQuad with Varying Floor Height

The previous section showed the response of the SemiQuad system given a set of errors in its initial configuration. This was done to give an overview of the functionality of the SemiQuad under normal operating conditions. The idea of this section is instead to see how the SemiQuad would perform given a more natural setting, with a fitting example being a room with varying floor height. Naturally, the motion of the robot is made with the assumption that the movement will happen on a perfect even floor, with no height variation whatsoever. However, even when moving indoors it is possible for a robot to experience difference in terrain, and any implementation of robustness should be able to counter the effect of this. Instead of simulating the system with a given set of initial errors, the system will instead be simulated starting with an optimal configuration. Then during the movement, the section of the floor where the swing foot will land will either be raised up or lowered, thus abruptly causing the system to deviate from the nominal trajectory. This will in addition coincide with the impact, meaning it occurs at a critical moment during the step. In the first simulation, the height of the floor was lowered with 5 cm, yielding the configuration after step 2 as shown in fig(35).

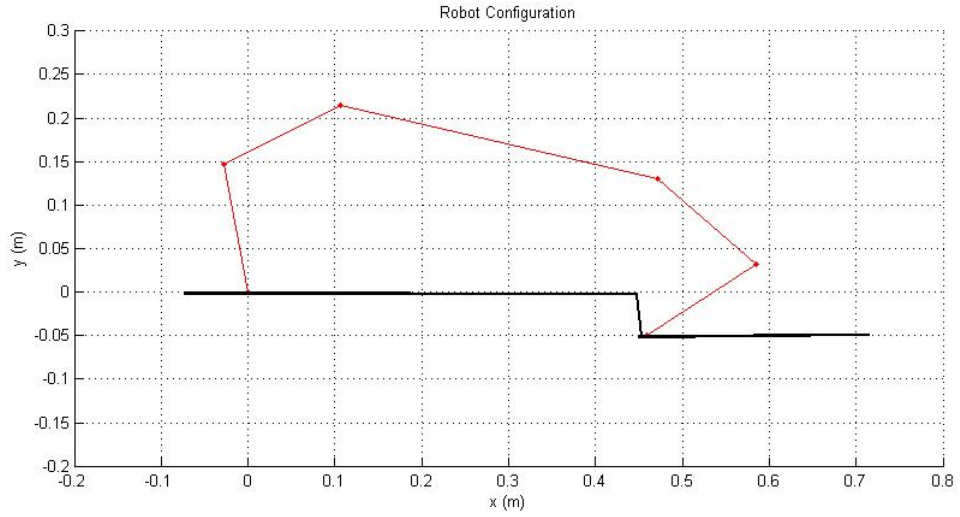



Figure 35: Configuration of the SemiQuad with the height of the floor lowered 5cm where the swing leg makes an impact.

Unfortunately, in this instance the pitfall proves too great, as the initial value for the

transverse coordinates for step 3 are far too high for the saturated controller to handle. In fact, the suggested torque values from the controller are more than twice that of the limit, thus with saturated torques the system is not able to recover in time. A more conservative value of 1 cm is then attempted, but again the system is unable to complete the gait. As the SemiQuad seemed incapable of handling this situation, the floor was instead raised by an increment of 5 mm. Theoretically this should be simpler for the SemiQuad as the generalized velocities will be smaller when the gait is stopped prematurely, compared to when the robot continue falling after the impact with the ground should have occurred. In this instance the SemiQuad is able to finish the gait, but given the minimal value of 5 mm, this can hardly be called a success. The explanation for the incapability in handling this situation however, becomes somewhat clear after studying the response of the one successful attempt. In fig(36) the phase portrait of $[\theta, \dot{\theta}]$ is shown. It can be seen that though the response is accurately following the nominal trajectory

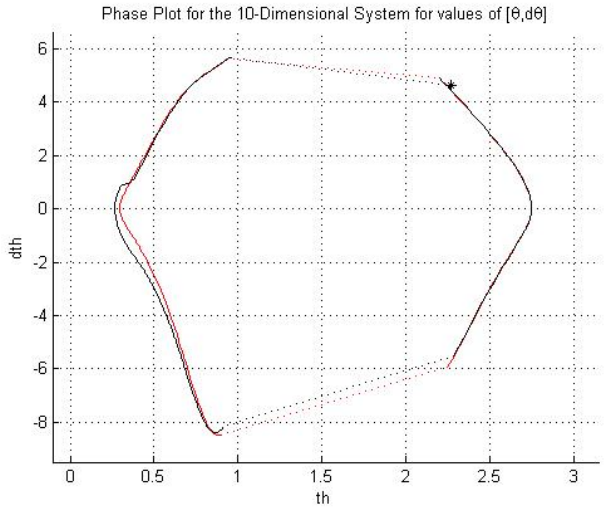



Figure 36: Phase portrait of $[\theta, \dot{\theta}]$ with the floor raised 5 mm

during step 1 and most of step 2, the premature impact with the ground causes a major deviation at the start of step 3. The system manages to recover, and a final configuration is found, though some distance away from the optimal values of $[\theta_e^*, \dot{\theta}_e^*]$. That the system does not end up in an optimal configuration however, is unfortunately due to the fact that an optimal configuration cannot be found with the two stance legs being at different heights. The entire gait and its trajectory is after all precalculated, with the values of the parametrization of the generalized coordinates and velocities already having been found. Indeed, if the final configuration has an optimal value for the first generalized coordinate given by $\theta_e^* = \theta_0^* = 2.2$, then at least one of the remaining generalized coordinates must deviate from their optimal value. If not, then the two stance legs would be at the same height, meaning the leg is either inside the object it should have stepped on, or it is freely floating in the air. This might be the greatest challenge

of running the system as it is now, as it is actually impossible to remove all errors by the end of the gait. This is shown in fig(37), which gives the evolution of all transverse coordinates for the four steps. As expected, the values stay approximately zero while

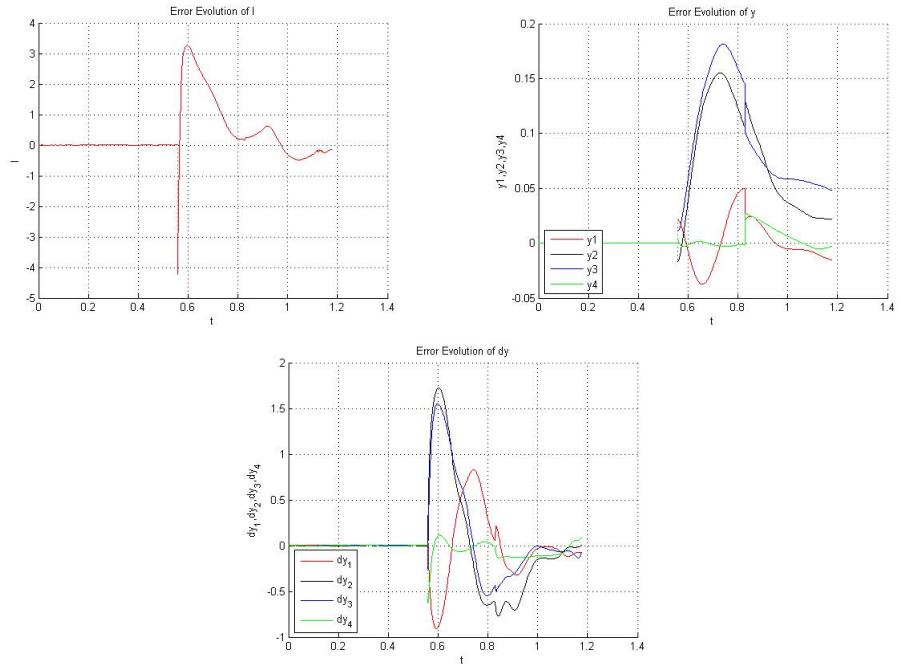



Figure 37: The Evolution of the Transverse Coordinates $[I, y, \dot{y}]$, given a variation in floor height.

following the gait in the first two steps. As the system reaches the premature impact however, the values of the transverse coordinates $I$ and $\dot{y}$ instantaneously grows to very large values, effectively signifying large errors in the system. This in turn yields a massive response from the controller, with torque values deviating heavily from those of the nominal trajectory as shown in figure(38). In particular, the initial spikes of $\tau_3$ and $\tau_4$ can be noticed right after impact, with the value of $\tau_3$ actually being saturated. This in turn means that for higher errors, which would be caused be increasing the floor height, these torque values would be even higher. This effectively means the controller will have saturated torque values for most floor deviations higher than 5 mm, which would most likely cause these simulations to fail. Running the simulation once more with the floor raised 1 cm confirms this, as the system is again incapable of completing the gait.
Though the system was capable of reducing its initial error configuration under normal operating conditions, the same can not be said for the introduction of errors through varying floor height. The main issue is the magnitude of the errors when introduced to only small changes in the height of the floor, again causing large deviations in all coordinates of the SemiQuad system. Additionally, the capabilities of the controller has been well demonstrated in this chapter, especially its capabilities of yielding near opti-

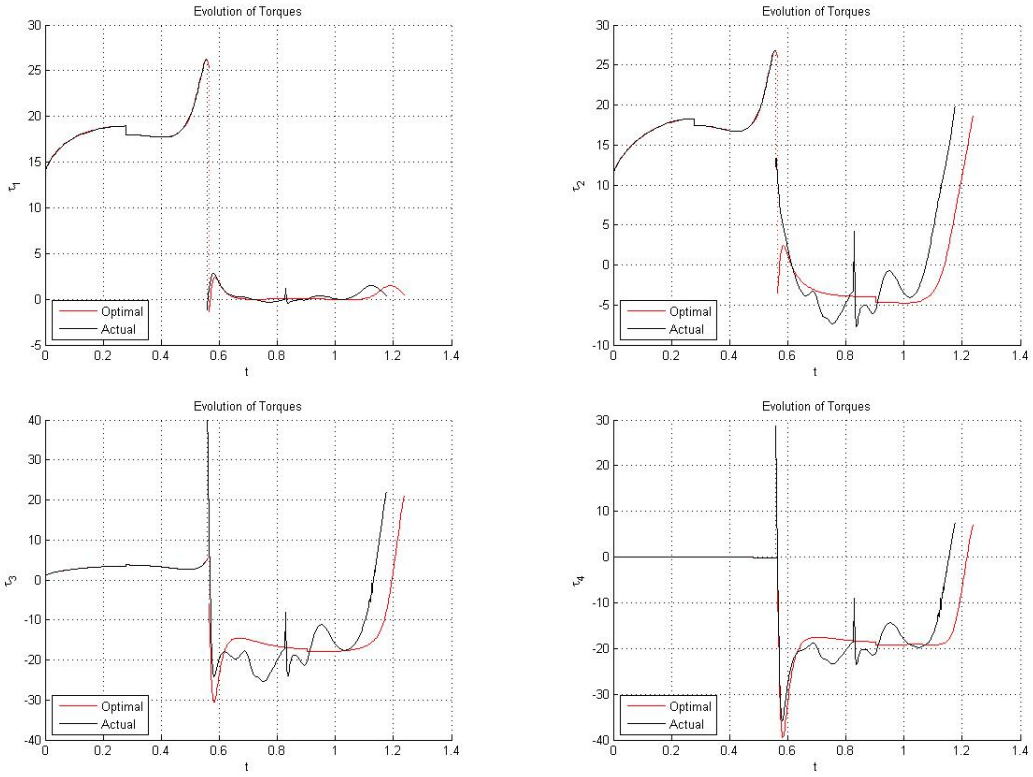



Figure 38: Evolutions of the torque values $\tau_i, i = 1..4$ with a varying floor height.

mal ending configurations. However, the results were most often either of an optimal nature, or a failure, with no results in between showing functional gaits but with bad ending configurations. This leads to the assumption that the controller might not be adequate for a system as shown in this section, where the complete removal of errors is in fact impossible. In fact, when simulating the same scenarios without the controller, the SemiQuad would occasionally manage to complete the gait, though in ending configurations that were far from optimal. Thus, it might be possible that the controller attempting to eliminate all errors from the system, actually causes the SemiQuad to fail instead. However, as mentioned in section 10.2.4, this is the first gait tested with control, and also the first controller to be tested with varying floor height, so it is too early to give any final conclusions on the exact capabilities of the controller at this point.

# 11 Conclusion

The work carried out in this thesis has shown that functional gaits can be created for the SemiQuad by describing the movement as a limit cycle. This simplified the design of the walking motion considerably, by reducing the stability criteria to that of ensuring the motion is a series of exact repetitions of a closed trajectory in state space. For the SemiQuad this was reduced to ensuring that each cycle started with the exact same configuration. It was further shown that through the use of virtual holonomic constraints, the dynamics of the SemiQuad could be fully described by a single variable $\theta$, simplifying both the dynamic model and the computational difficulty of finding evolutions of the robot. Indeed the 10th order dynamic system was readily reduced to that of a 2nd order dynamic system, allowing the entire movement to be described by the state space evolution of $[\theta, \dot{\theta}]$. All joints were thus synchronized with respect to the ankle angle of the stance leg and the reduced dynamics of the system could completely describe the motion of the SemiQuad. The complete gait was further modelled as a hybrid system, as the impacts with the ground introduced instantaneous changes in the joint velocities. An impact model was created, mainly since it was desired to take advantage of the impact itself to create a faster and more efficient motion. As the SemiQuad was not constrained to come to a complete stop before each impact, it had the advantage of beginning the next section of the step with an angular velocity in the direction of intended motion.
Through the optimization of specific cost functions, which ensured periodicity of the gait and sought to enhance specific criteria such as energy efficiency or increased velocity, many types of gaits were found. It was shown that when the order of the parametrization was increased, the SemiQuad was allowed more freedom in its choice of movement, leading to better results. With this increase however, came also the computational complexity of having more optimization variables, meaning more time was needed to find proper solutions.
With the previous 2-dimensional simulations acting as reference trajectories, the system was expanded to its original 10-dimensional size. Then, the SemiQuad was shown to be capable of following the nominal trajectories, provided the initial configuration was correct. However, this was no longer the case when errors were introduced into the system, as the SemiQuad would then deviate from the intended path, which would result in erroneous final configurations or a failure in completing the gait altogether. In fact, it was further shown that the introduction of any error would eventually cause the SemiQuad to fail, as the errors would grow larger over time, thus preventing the SemiQuad from completing several steps in a row.
As a way to describe the deviation from the nominal trajectory, the transverse coordinates where introduced. These $x_\perp \in \mathbb{R}^9$ coordinates defined a moving poincarè section, describing the deviation from the nominal trajectory $x^*$ at every moment of time $t$. These coordinates were then linearized and a linear system was created, a system which was shown to accurately represent the nonlinear system in a vicinity of the nominal trajectory. This meant that a controller made for the linear system could be used on the nonlinear system, thus simplifying the creation of control. The implementation of con-

trol still proved to be quite challenging however, with simple designs being incapable of stabilizing the gait. The final controller used was that of an SDP-based approximation of stabilizing solutions for periodic Riccati differential equations. With this controller implemented the gait was stabilized, as long as the errors didn't exceed a certain threshold. Failure to correct an error however, was most often caused by the controller demanding more torque than the system could give. The torque values were then set as high as they could, but this would often prove insufficient to eliminate the errors. The instants where the natural response of the gait had very high torques became especially susceptible to failure, as the controller would have little room to change the torques without reaching the saturation limit. However, the error threshold was still of sufficient size, which proved the capability of designing exponentially stable periodic gaits for the SemiQuad system. It is still worth keeping in mind that a gait using even less torque for the natural movement, would most likely have a higher resistance to errors.
Finally, the SemiQuad was tested with changing floor height, to try to induce errors from interaction with the environment. This entailed having the swing foot hit ground either before or after what was intended, naturally creating errors, while also shifting the configuration of the robot. Unfortunately, the SemiQuad had much difficulty in completing the gait for errors of this nature, with the system only finishing the gait for slight deviations. It was shown that errors of this type were especially difficult for the controller, as the removal of errors completely from the system became impossible.
In summary, the combination of limit cycle modelling with optimized virtual constraints can be considered to be an efficient way of discovering natural gaits. Not only is the process straightforward, but control can also be suitably implemented. Though the SemiQuad will experience failures for errors above a certain threshold, the gait is still exponentially stable within its working condition. Further, as control was only implemented for a single gait, this can be considered a prototype. Therefore, there is reason to believe future gaits created based on this prototype will achieve better results while having a higher resistance to errors.

# 12 Future Work

The first part of this thesis consisted in creating various gaits for the SemiQuad robot, and though many walking motions were eventually discovered, I feel the method of finding them has room for improvement. First of all, the optimization routine was too simplistic and consisted mainly in maintaining a periodic solution while limiting the energy usage. This was later expanded upon to take into consideration specific errors and optimize specific parts of the gait. However, to ensure that results could be found within a reasonable amount of time, much work had to be done manually on the variables to ensure good starting conditions for the optimization routine, which again required a great deal of natural knowledge about the gait. As I have worked and created this thesis, this task was adequate for me, but it would have been difficult for someone new with the topic to quickly realize how to edit these variables correctly. One feature that could be implemented in the future is an expansion on the optimization routine that includes specific knowledge of the gait of the SemiQuad. There are many pit falls when starting to design a gait, which if they are followed will never yield proper results. One example is the evolution of the first step. If the SemiQuad is unable to move far enough back, thus raising its body high enough before the velocity of the ankle angle reaches zero, it will never be able to reach the step length required in step 2 without exhausting the maximum torque. Further, if the velocity of the ankle angle after the impact is too high, and step 3 fails as a result of this, the solution is not to edit step 3 itself, but rather edit step 1 and 2 to ensure an initial velocity within reasonable bounds. These are just two examples of insightful knowledge of the gait of the SemiQuad which could be useful if implemented correctly in an optimization routine. The idea would then be to create an optimization algorithm which could work much more independently and efficiently at finding gaits.

The second part of this thesis consisted in implementing the discovered gaits with a control system which could ensure orbital exponential stability. The controller itself was shown to be capable of handling errors of quite a large magnitude, though the result on the system became somewhat restrained due to the limitation in actuation. This was shown to mainly be caused by the instants where the natural gait had high torques, meaning the controller had little or no room to move. Now the success of the gait can in the first place be contributed to the energy optimization which for the most part had limited the usage of high torques. One way of ensuring gaits with even higher robustness would then be to edit the optimization of the gait to not only ensure low energy usage, but also try to limit high spikes in the torque values. Further, as the controller was only created for a single gait, the amount of knowledge one could gather about the various causes for failure was limited. The logical next step is then to create controllers for several other gaits and compare the functionality. When a suitable amount of data has been attained, this could be inserted into the aforementioned optimization algorithm to specifically create a gait which would work well together with a controller, thus achieving a higher robustness towards errors.

Lastly, the system was not able to solve errors of a high magnitude, nor to correctly

simulate gaits with varying floor height, as this would cause the controller to issue very high torque signals which, with saturation, would fail to control the gait. This again means the system is capable of operating under normal conditions with errors of a certain size, but that it is not capable of surviving a shock to the system, a shock being a sudden error of a large magnitude. However, during the first part of this thesis, numerous types of gaits were found. These differed much in terms of body configuration and greatly in terms of general velocities, with $\dot{\theta}_0$ alone varying much from gait to gait. The difference in the gaits were much larger than the highest tolerance of errors from the controller, which brings me to my point. Say a jolt in the system lowers the value of $\dot{\theta}_0$ greatly, the actuation required to return to the nominal trajectory would most likely be higher than the saturation limit. In addition to the saturated torques, there is also the issue of region of functionality for the controller, as it is only designed to work in proximity to the nominal gait. Either of these will most likely result in the gait failing. However, say the system suffers a jolt and the new $\dot{\theta}$ is in fact close to a secondary gait, the SemiQuad might avoid falling if the controller is shifted to follow this secondary nominal trajectory instead. For proper functionality several walking techniques must be precalculated, and control must in turn be found for each of them. Though for our system, the solutions and control are all precalculated, and thus several solutions with control could be stored at once. What trajectory to follow would then be the trajectory with the lowest transverse coordinates, with lowest referring to some predefined weighting set by the user. The dominant issues are naturally the high number of gaits that need to be found to ensure the system should in most cases be close to one of them, in addition to the computational complexity of calculating the transverse coordinates several times. Though, implemented properly, it could still have a good effect on the system, acting as a fail safe against larger errors.

## References


[1] Yannick Aoustin, Gaëtan Garcia, Philippe Lemoine: "*Estimation of the absolute Orientation of a Five-link Walking Robot with Passive Feet*", 2007, Institut de Recherche en Communications et Cybernétique de Nantes (IRCCyN), ch 3, pp. 31-44

[2] Daan G.E Hobbelen, Martijn Wisse: "*Humanoid Robots: Human-like Machines*", 2007, Delft University of Technology. ch. 14: Limit Cycle Walking, pp. 277-294

[3] A.D. Kuo: "*Choosing your steps carefully: Trade-offs Between Economy and versatility in Dynamic walking Bipedal Robots*", June 2007, IEEE Robotics & Automation Magazine, pp. 18-29

[4] Micheal H. Dickinson, et al: "*How Animals Move: An Integrative View*", 2000, Science Magazine vol. 288, pp. 100-106

[5] Chevallereau, Abba, Aoustin, Plestan, Westervelt, Canudas-de-vit, Grizzle: "*RABBIT: A testbed for Advanced Control Theory*, October 2003, IEEE Control Systems Magazine, pp. 57-79

[6] A. Muraro, C. Chevallereau, Y. Aoustin: "*Optimal trajectories for a Quadruped Robot with Trot, Amble and Curvet gaits for Two Energetic Criteria*, 2003, Multibody System Dynamics 9, 99. 39-62

[7] Shiriaev, Freidovich, Robertsson, Johansson, Sandberg: "*Virtual-Holonomic-Constraints-Based Design of Stable Oscillations of Furuta Pendulum: Theory and Experiments*, 2007, IEEE Transactions on Robotics, vol 23, NO. 4, pp. 827-832

[8] Shiriaev, Perram, Canudas-de-Wit: "*Constructive Tool for Orbital Stabilization of Underactuated Nonlinear Systems: Virtual Constrains Approach*, 2005, Transactions on automatic control,Vol. 50, NO. 8, pp. 1164-1176

[9] Shiriaev, Robersson, Perram, Sandberg: "*Periodic motion planning for virtually constrained Euler–Lagrange systems*, 2006, System and Control Letters 55, pp. 900-907

[10] Shiriaev, Robersson, Perram, Sandberg: "*Periodic Motion Planning for Virtually Constrained (Hybrid) Mechanical Systems*", 2005, 44th IEEE Conference on Decision and Control, and the Europeans Control conference, Seville Spain, pp. 4035-4040

[11] Shiriaev, Perram, Robertsson, Sandberg: "*Formulas for Gneeral Integrals of Motion for a class of Mechanical Systems Subject to Virtual Constraints*", 2004, 43rd IEEE Conference on Decision and Control, Dec. 14-17, pp. 1159-1163

[12] J.Y. Kim, I.W. Park, J.H. Oh: "*Walking Control Algorithms of Biped Humanoid Robot on Uneven and Inclined Floor*, 2006, J Intell Robot Syst(2007) 48, pp. 457-484

[13] Tad McGeer: "*Passive Dynamic Walking*, 1990, The International Journal of Robotics Research, 9(2), pp. 62-88

[14] Uwe Mettin: "*Principles for Planning and Analyzing Motions of Underactuated Mechanical Systems and Redundant Manipulators*, 2009, Umeå University, Doctoral Thesis

[15] E.R. Westervelt, J.W. Grizzle, C. Chevallereau, J.H. Choi, B. Morris: "*Feedback control of Dynamic bipedal Robot Locomotion*

[16] Henrik Røst Breivik: "*Modelling and gait Creation for the SemiQuad Robot*, 2012, NTNU, project work

[17] Shiriaev, Freidovich, Gusev: "*Transverse Linearization for Controlled Mechanical Systems With Several Passive Degrees of Freedom*", IEEE transactions on automatic control, Vol 55, NO. 4, April 2010

[18] Shiriaev, Freidovich, Manchester: "*Can we make a robot ballerina perform a pirouette? Orbital stabilization of periodic motions of underactuated mechanical systems*", Annual Reviews in Control 32 (2008) 200-211

[19] Shiriaev, Freidovich: "*Transverse Linearization for Impulsive Mechanical Systems With One Passive Link*", IEEE Transactions on Automatic Control, Vol. 54, NO. 12, December 2009

[20] Pchelkin, Shiriaev, Freidovich, Mettin, Gusev, Kwon: "*Natural Sit-Down and Chair-Rise Motions for a Humanoid Robot*", 49th IEEE Conference on Decision and Control, December 15-17, 2010

[21] Freidovich, Shiriaev, Manchester: "*Stability Analysis and Control Design for an Underactuated Walking Robot via Computation of a Transverse Linearization*", 17th World Congress, The International Federation of Automatic Control, July 6-11, 2008

[22] Gusev, Shiriaev, Freidovich: "*SDP-Based Approximation of Stabilizing Solutions for Periodic Riccati Differential Equations*", International Journal of Control

# Appendix A

This section contains the results of the simulations performed in chapter 6. The position matrices are as shown, included with the other optimizable parameters below. $f$ is the value of the energy cost function, while $T_{tot}$ is the total time of the gait.

## 2nd Order Parametrizations

### Energy Optimized Solutions

#### Step Length Small

$$P_1 = \begin{bmatrix} -1.3000 & -1.4737 & -1.4992 \\ -0.9000 & -0.7501 & -0.7406 \\ -0.9000 & -0.8084 & -0.7170 \\ -1.3000 & -1.5259 & -1.5150 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.4808 & -1.6567 & -1.5000 \\ -0.7749 & -0.6153 & -0.7500 \\ -0.6139 & -0.6633 & -0.7500 \\ -1.4270 & -1.7016 & -1.5000 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.5000 & 1.4589 & 1.5581 \\ 0.7500 & 0.7382 & 0.6519 \\ 0.7500 & 0.9522 & 0.9163 \\ 1.5000 & 1.1896 & 1.3233 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.7634 & 1.2443 & 1.3000 \\ 0.4962 & 0.8933 & 0.9000 \\ 0.7516 & 1.1371 & 0.9000 \\ 1.7233 & 0.9296 & 1.3000 \end{bmatrix}$$

$$f = 12.9341,\ \dot{\theta}_0 = 4.6011,\ \theta_e^1 = 2.7912,\ \theta_e^3 = 0.5042,\ T_{tot} = 3.4559$$

#### Step Length Medium

$$P_1 = \begin{bmatrix} -1.3000 & -1.4626 & -1.5404 \\ -0.9000 & -0.7463 & -0.7000 \\ -0.9000 & -0.8611 & -0.7725 \\ -1.3000 & -1.5637 & -1.5195 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.6110 & -1.7521 & -1.5500 \\ -0.6965 & -0.3042 & -0.5000 \\ -0.5926 & -0.6639 & -0.7000 \\ -1.2379 & -1.9917 & -1.6340 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6340 & 1.4697 & 1.5468 \\ 0.7000 & 0.6402 & 0.5478 \\ 0.5000 & 0.7121 & 0.7050 \\ 1.5500 & 1.3466 & 1.5181 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.6796 & 1.0209 & 1.3000 \\ 0.4497 & 0.8355 & 0.9000 \\ 0.6459 & 1.2318 & 0.9000 \\ 1.7749 & 0.9311 & 1.3000 \end{bmatrix}$$

$$f = 13.1827,\ \dot{\theta}_0 = 4.49307,\ \theta_e^1 = 2.7415,\ \theta_e^3 = 0.4430,\ T_{tot} = 2.8400$$

#### Step Length Long

$$P_1 = \begin{bmatrix} -1.3000 & -1.5051 & -1.6737 \\ -0.9000 & -0.6794 & -0.5173 \\ -0.9000 & -0.8189 & -0.4830 \\ -1.3000 & -1.5888 & -1.9126 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.8191 & -1.8115 & -1.5500 \\ -0.3822 & -0.1634 & -0.4500 \\ -0.1258 & -0.5711 & -0.6000 \\ -2.2136 & -1.9923 & -1.6480 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6480 & 1.5638 & 1.7686 \\ 0.6000 & 0.5988 & 0.3129 \\ 0.4500 & 0.4143 & 0.5120 \\ 1.5500 & 1.4975 & 1.7099 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.6608 & 1.1401 & 1.3000 \\ 0.4780 & 0.9585 & 0.9000 \\ 0.4600 & 0.8299 & 0.9000 \\ 1.5938 & 1.1824 & 1.3000 \end{bmatrix}$$

$$f = 13.7255,\ \dot{\theta}_0 = 4.5647,\ \theta_e^1 = 2.8094,\ \theta_e^3 = 0.4810,\ T_{tot} = 1.8412$$

**Velocity Optimized Solutions**

**Step Length Small**

$$P_1 = \begin{bmatrix} -1.3000 & -1.5297 & -1.5913 \\ -0.9000 & -0.7520 & -0.7631 \\ -0.9000 & -0.7749 & -0.6341 \\ -1.3000 & -1.4807 & -1.3380 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.5707 & -1.6870 & -1.5000 \\ -0.7631 & -0.6346 & -0.7500 \\ -0.6732 & -0.6374 & -0.7500 \\ -1.3717 & -1.6281 & -1.5000 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.5000 & 1.4835 & 1.7098 \\ 0.7500 & 0.8648 & 0.7041 \\ 0.7500 & 0.6830 & 0.5594 \\ 1.5000 & 1.4117 & 1.6323 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.6377 & 1.2244 & 1.3000 \\ 0.7521 & 1.1004 & 0.9000 \\ 0.6011 & 0.8330 & 0.9000 \\ 1.5642 & 1.1236 & 1.3000 \end{bmatrix}$$

$$f = 39.5784,\ \dot{\theta}_0 = 4.6969,\ \theta_e^1 = 2.7165,\ \theta_e^3 = 0.5777,\ T_{tot} = 0.7433$$

**Step Length Medium**

$$P_1 = \begin{bmatrix} -1.3000 & -1.4594 & -1.5172 \\ -0.9000 & -0.7612 & -0.7227 \\ -0.9000 & -0.8092 & -0.8110 \\ -1.3000 & -1.5302 & -1.5450 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.5500 & -1.7718 & -1.5500 \\ -0.7120 & -0.3036 & -0.5000 \\ -0.8445 & -0.5643 & -0.7000 \\ -1.4893 & -1.9727 & -1.6340 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6340 & 1.2416 & 1.5129 \\ 0.7000 & 1.0041 & 0.6946 \\ 0.5000 & 1.0321 & 0.8189 \\ 1.5500 & 0.8362 & 1.2979 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.8947 & 0.5873 & 1.3000 \\ 0.2887 & 1.4849 & 0.9000 \\ 0.4730 & 1.8157 & 0.9000 \\ 1.9569 & 0.0140 & 1.3000 \end{bmatrix}$$

$$f = 37.0619,\ \dot{\theta}_0 = 4.5397,\ \theta_e^1 = 2.6306,\ \theta_e^3 = 0.4539,\ T_{tot} = 1.5967$$

**Step Length Long**

$$P_1 = \begin{bmatrix} -1.3000 & -1.5043 & -1.7065 \\ -0.9000 & -0.6802 & -0.5167 \\ -0.9000 & -0.8194 & -0.4858 \\ -1.3000 & -1.5895 & -1.8152 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.9475 & -1.7969 & -1.5500 \\ -0.3413 & -0.1709 & -0.4500 \\ 0.0037 & -0.5657 & -0.6000 \\ -2.0622 & -2.0130 & -1.6480 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6480 & 1.7146 & 1.8232 \\ 0.6000 & 0.4870 & 0.3143 \\ 0.4500 & 0.4480 & 0.5133 \\ 1.5500 & 1.5257 & 1.7118 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.7156 & 1.3701 & 1.3000 \\ 0.4877 & 0.7772 & 0.9000 \\ 0.4615 & 0.8752 & 0.9000 \\ 1.5700 & 1.1997 & 1.3000 \end{bmatrix}$$

$$f = 16.3229,\ \dot{\theta}_0 = 4.5551\ \theta_e^1 = 2.8078,\ \theta_e^3 = 0.4831,\ T_{tot} = 1.6648$$

### 3rd Order Parametrizations

#### Energy Optimized Solutions

#### Step Length Small

$$P_1 = \begin{bmatrix} -1.3000 & -1.4505 & -1.4724 & -1.1273 \\ -0.9000 & -0.7070 & -0.8008 & -0.4923 \\ -0.9000 & -0.8764 & -1.0561 & -0.6789 \\ -1.3000 & -1.2055 & -1.8755 & -1.3719 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.3965 & -1.4819 & -1.4857 & -1.5000 \\ -0.7814 & -0.8461 & -0.7854 & -0.7500 \\ -0.9144 & -0.9776 & -0.8474 & -0.7500 \\ -1.3777 & -1.9784 & -1.4333 & -1.5000 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.5000 & 1.5039 & 1.4698 & 1.5070 \\ 0.7500 & 0.7500 & 0.7549 & 0.5748 \\ 0.7500 & 0.7567 & 0.8272 & 1.4828 \\ 1.5000 & 1.1695 & 0.1306 & 0.8124 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.4943 & 0.9977 & 1.0502 & 1.3000 \\ 0.7425 & 1.2536 & 1.2140 & 0.9000 \\ 0.8096 & 1.2423 & 1.2679 & 0.9000 \\ 0.9673 & -0.7586 & 0.8012 & 1.3000 \end{bmatrix}$$

$$f = 3.0896,\ \dot{\theta}_0 = 3.0896,\ \theta_e^1 = 4.4357,\ \theta_e^3 = 0.0365,\ T_{tot} = 1.5978$$

#### Step Length Medium

$$P_1 = \begin{bmatrix} -1.3000 & -1.4064 & -1.3129 & -1.3777 \\ -0.9000 & -0.8039 & -0.8052 & -0.7995 \\ -0.9000 & -0.8821 & -1.2952 & -0.8527 \\ -1.3000 & -1.4702 & -1.7336 & -1.3636 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.3560 & -1.3418 & -1.4865 & -1.5500 \\ -0.8025 & -0.5196 & -0.5166 & -0.5000 \\ -1.0160 & -1.4591 & -0.7861 & -0.7000 \\ -1.5061 & -2.1929 & -1.7448 & -1.6340 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6340 & 1.6425 & 1.6364 & 1.2991 \\ 0.7000 & 0.5683 & 0.5856 & 0.6164 \\ 0.5000 & 0.5064 & 0.2997 & 0.9115 \\ 1.5500 & 1.7787 & 1.7010 & 1.3253 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.6040 & 1.6340 & 1.3029 & 1.3000 \\ 0.5991 & 0.7965 & 0.7820 & 0.9000 \\ 0.4747 & -0.2526 & 0.9273 & 0.9000 \\ 1.6707 & 1.4220 & 1.2417 & 1.3000 \end{bmatrix}$$

$$f = 5.5720,\ \dot{\theta}_0 = 3.9397,\ \theta_e^1 = 3.2028,\ \theta_e^3 = 0.3539,\ T_{tot} = 1.6017$$

#### Step Length Long

$$P_1 = \begin{bmatrix} -1.3000 & -1.4957 & -1.6721 & -1.6848 \\ -0.9000 & -0.6761 & -0.5077 & -0.5182 \\ -0.9000 & -0.8544 & -0.4909 & -0.4691 \\ -1.3000 & -1.5815 & -1.9743 & -1.9864 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.6418 & -1.8100 & -1.5523 & -1.5500 \\ -0.5877 & -0.2017 & -0.4539 & -0.4500 \\ -0.5753 & -0.1786 & -0.7110 & -0.6000 \\ -1.8645 & -2.3000 & -1.7217 & -1.6480 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6480 & 1.5512 & 1.4000 & 1.6699 \\ 0.6000 & 0.5602 & 0.7164 & 0.5578 \\ 0.4500 & 0.4499 & 0.9208 & 0.5091 \\ 1.5500 & 1.4630 & 1.2181 & 1.0160 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.6540 & 1.0854 & 1.2916 & 1.3000 \\ 0.5670 & 1.0935 & 0.8922 & 0.9000 \\ 0.5331 & 1.2983 & 0.6981 & 0.9000 \\ 1.0283 & 1.1337 & 1.4049 & 1.3000 \end{bmatrix}$$

$$f = 13.432,\ \dot{\theta}_0 = 4.5729\ \theta_e^1 = 2.8985,\ \theta_e^3 = 0.3512,\ T_{tot} = 2.3423$$

**Velocity Optimized Solutions**

**Step Length Small**

$$P_1 = \begin{bmatrix} -1.3000 & -1.4929 & -1.5090 & -1.5072 \\ -0.9000 & -0.7460 & -0.7533 & -0.8920 \\ -0.9000 & -0.7895 & -0.7409 & -0.7036 \\ -1.3000 & -1.4889 & -1.4900 & -1.4818 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.4996 & -1.5303 & -1.5195 & -1.5000 \\ -0.8032 & -0.7743 & -0.7495 & -0.7500 \\ -0.7419 & -0.7036 & -0.7421 & -0.7500 \\ -1.4818 & -1.4846 & -1.4850 & -1.5000 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.5000 & 1.5515 & 1.6445 & 1.5545 \\ 0.7500 & 0.8100 & 0.6706 & 0.6780 \\ 0.7500 & 0.7449 & 0.8937 & 0.9020 \\ 1.5000 & 1.4568 & 1.3427 & 1.2895 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.5410 & 1.5827 & 1.3409 & 1.3000 \\ 0.7636 & 0.6070 & 0.8930 & 0.9000 \\ 0.7687 & 1.1773 & 0.8816 & 0.9000 \\ 1.4587 & 1.0757 & 1.2376 & 1.3000 \end{bmatrix}$$

$$f = 104.6375,\ \dot{\theta}_0 = 4.5699,\ \theta_e^1 = 2.6137,\ \theta_e^3 = 0.3960,\ T_{tot} = 0.5203$$

**Step Length Medium**

$$P_1 = \begin{bmatrix} -1.3000 & -1.4658 & -1.5050 & -1.5189 \\ -0.9000 & -0.7633 & -0.7276 & -0.7286 \\ -0.9000 & -0.8084 & -0.8542 & -0.7465 \\ -1.3000 & -1.5177 & -1.5678 & -1.5316 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.5066 & -1.6393 & -1.5653 & -1.5500 \\ -0.7329 & -0.4232 & -0.4997 & -0.5000 \\ -0.7951 & -0.8852 & -0.7164 & -0.7000 \\ -1.5423 & -1.8289 & -1.6986 & -1.6340 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6340 & 1.6427 & 1.5712 & 1.5028 \\ 0.7000 & 0.5325 & 0.6497 & 0.7102 \\ 0.5000 & 1.0370 & 0.8222 & 0.8511 \\ 1.5500 & 1.3052 & 1.2840 & 1.3113 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.6238 & 1.1300 & 1.3054 & 1.3000 \\ 0.6183 & 1.3513 & 0.9035 & 0.9000 \\ 0.7990 & -0.3475 & 0.9034 & 0.9000 \\ 1.3911 & 1.1052 & 1.1916 & 1.3000 \end{bmatrix}$$

$$f = 61.1766,\ \dot{\theta}_0 = 4.5340,\ \theta_e^1 = 2.6101,\ \theta_e^3 = 0.4531,\ T_{tot} = 0.7497$$

**Step Length Long**

$$P_1 = \begin{bmatrix} -1.3000 & -1.5180 & -1.6864 & -1.6703 \\ -0.9000 & -0.6846 & -0.5223 & -0.5419 \\ -0.9000 & -0.8266 & -0.4898 & -0.4864 \\ -1.3000 & -1.6074 & -1.9046 & -1.7620 \end{bmatrix}, \quad P_2 = \begin{bmatrix} -1.6699 & -1.8730 & -1.5637 & -1.5500 \\ -0.5412 & -0.1525 & -0.4523 & -0.4500 \\ -0.4963 & -0.0222 & -0.6247 & -0.6000 \\ -1.7916 & -2.2951 & -1.6973 & -1.6480 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 1.6480 & 1.7426 & 1.4669 & 1.6674 \\ 0.6000 & 0.4259 & 0.7239 & 0.5176 \\ 0.4500 & 0.3733 & 1.0113 & 0.5216 \\ 1.5500 & 1.6635 & 1.1757 & 1.1055 \end{bmatrix}, \quad P_4 = \begin{bmatrix} 1.5930 & 0.8783 & 1.2511 & 1.3000 \\ 0.5931 & 1.3147 & 0.9130 & 0.9000 \\ 0.7065 & 1.5838 & 0.7171 & 0.9000 \\ 1.2219 & 0.8334 & 1.3451 & 1.3000 \end{bmatrix}$$

$$f = 35.1338,\ \dot{\theta}_0 = 4.7471\ \theta_e^1 = 2.8532,\ \theta_e^3 = 0.3607,\ T_{tot} = 1.3320$$

## 4th Order Parametrizations

### Energy Optimized Solutions

### Step Length Small

$$P_1 = \begin{bmatrix} -1.3000 & -1.4461 & -1.4973 & -1.5108 & -1.5065 \\ -0.9000 & -0.7525 & -0.7495 & -0.8416 & -0.5838 \\ -0.9000 & -0.9458 & -0.7606 & -0.7901 & -0.7040 \\ -1.3000 & -1.4837 & -1.5879 & -1.6539 & -1.4674 \end{bmatrix}, \quad f = 12.2441,$$

$$P_2 = \begin{bmatrix} -1.5070 & -1.6999 & -1.6881 & -1.6566 & -1.5000 \\ -0.6735 & -0.6658 & -0.5984 & -0.6468 & -0.7500 \\ -0.7349 & -0.6746 & -0.6533 & -0.6600 & -0.7500 \\ -1.5318 & -1.7301 & -1.6819 & -1.6879 & -1.5000 \end{bmatrix}, \quad \dot{\theta}_0 = 4.3759,$$

$$P_3 = \begin{bmatrix} 1.5000 & 1.2581 & 1.4186 & 1.4289 & 1.6408 \\ 0.7500 & 0.7976 & 0.7042 & 0.7392 & 0.6579 \\ 0.7500 & 1.1139 & 1.0166 & 1.0003 & 0.7712 \\ 1.5000 & 1.0212 & 1.2334 & 1.1845 & 1.3246 \end{bmatrix}, \quad \theta_e^1 = 3.1550,\ \theta_e^3 = 0.3210,$$

$$P_4 = \begin{bmatrix} 1.6243 & 1.2824 & 1.2907 & 1.2928 & 1.3000 \\ 0.6641 & 1.0559 & 1.0065 & 0.8545 & 0.9000 \\ 0.7890 & 1.1566 & 1.1536 & 1.1764 & 0.9000 \\ 1.3139 & 0.8302 & 0.9018 & 0.9706 & 1.3000 \end{bmatrix}, \quad T_{tot} = 2.3368$$

### Step Length Medium

$$P_1 = \begin{bmatrix} -1.3000 & -1.3837 & -1.2590 & -1.4084 & -1.3296 \\ -0.9000 & -0.8203 & -0.7925 & -0.8053 & -0.8216 \\ -0.9000 & -0.8841 & -1.2738 & -0.8614 & -0.8662 \\ -1.3000 & -1.4804 & -1.5394 & -1.3211 & -1.3992 \end{bmatrix}, \quad f = 7.0610,$$

$$P_2 = \begin{bmatrix} -1.3465 & -1.5845 & -1.3571 & -1.5430 & -1.5500 \\ -0.8076 & -0.5452 & -0.5307 & -0.5134 & -0.5000 \\ -0.9776 & -0.8227 & -1.4036 & -0.7971 & -0.7000 \\ -1.4115 & -1.8594 & -2.1836 & -1.6891 & -1.6340 \end{bmatrix}, \quad \dot{\theta}_0 = 3.8865,$$

$$P_3 = \begin{bmatrix} 1.6340 & 1.5794 & 1.6304 & 1.6614 & 1.3529 \\ 0.7000 & 0.5442 & 0.5503 & 0.5758 & 0.6236 \\ 0.5000 & 0.6250 & 0.5227 & 0.3014 & 0.9199 \\ 1.5500 & 1.9087 & 1.8046 & 1.6525 & 1.3540 \end{bmatrix}, \quad \theta_e^1 = 3.8865,\ \theta_e^3 = 0.3661,$$

$$P_4 = \begin{bmatrix} 1.5869 & 1.7244 & 1.4472 & 1.2965 & 1.3000 \\ 0.5749 & 0.8284 & 0.8063 & 0.7778 & 0.9000 \\ 0.5198 & -1.3198 & -0.4223 & 0.9066 & 0.9000 \\ 1.6728 & 1.2870 & 1.4099 & 1.3022 & 1.3000 \end{bmatrix}, \quad T_{tot} = 2.8252$$

**Velocity Optimized Solutions**

**Step Length Small**

$$P_1 = \begin{bmatrix} -1.3000 & -1.4685 & -1.4765 & -1.5312 & -1.5240 \\ -0.9000 & -0.7470 & -0.7529 & -0.7580 & -0.7597 \\ -0.9000 & -0.8438 & -0.8101 & -0.8147 & -0.7014 \\ -1.3000 & -1.5193 & -1.5329 & -1.4863 & -1.5066 \end{bmatrix}, \quad f = 195.4920,$$

$$P_2 = \begin{bmatrix} -1.5218 & -1.6958 & -1.6510 & -1.6498 & -1.5000 \\ -0.7584 & -0.6076 & -0.6033 & -0.5628 & -0.7500 \\ -0.7530 & -0.6110 & -0.6163 & -0.6659 & -0.7500 \\ -1.5017 & -1.6500 & -1.6871 & -1.6847 & -1.5000 \end{bmatrix}, \quad \dot{\theta}_0 = 4.7320,$$

$$P_3 = \begin{bmatrix} 1.5000 & 1.0741 & 1.4744 & 1.4667 & 1.5148 \\ 0.7500 & 1.3027 & 0.7411 & 0.7323 & 0.6463 \\ 0.7500 & 1.1013 & 0.9487 & 1.0922 & 0.9214 \\ 1.5000 & 0.6860 & 1.1779 & 1.2087 & 1.3368 \end{bmatrix}, \quad \theta_e^1 = 2.8566,\ \theta_e^3 = 0.3108,$$

$$P_4 = \begin{bmatrix} 1.3142 & 1.2639 & 1.3016 & 1.2357 & 1.3000 \\ 0.9800 & 0.9726 & 0.9467 & 0.9796 & 0.9000 \\ 0.9521 & 1.4974 & 1.1482 & 1.1342 & 0.9000 \\ 1.0807 & 0.9264 & 0.9321 & 0.9777 & 1.3000 \end{bmatrix}, \quad T_{tot} = 0.9576$$

**Step Length Medium**

$$P_1 = \begin{bmatrix} -1.3000 & -1.4993 & -1.5123 & -1.6622 & -1.6777 \\ -0.9000 & -0.7017 & -0.6356 & -0.4573 & -0.4542 \\ -0.9000 & -0.8302 & -0.8605 & -0.2891 & -0.2947 \\ -1.3000 & -1.5897 & -1.6582 & -1.7919 & -1.7278 \end{bmatrix}, \quad f = 73.0497,$$

$$P_2 = \begin{bmatrix} -1.3560 & -2.1436 & -1.8870 & -1.5466 & -1.5500 \\ -0.8025 & -0.0315 & -0.3433 & -0.4916 & -0.5000 \\ -1.0160 & 0.4504 & -0.5556 & -0.6951 & -0.7000 \\ -1.5061 & -1.9437 & -1.6812 & -1.6969 & -1.6340 \end{bmatrix}, \quad \dot{\theta}_0 = 4.5233,$$

$$P_3 = \begin{bmatrix} 1.6340 & 1.7231 & 1.5890 & 1.7650 & 1.6053 \\ 0.7000 & 0.5578 & 0.6063 & 0.3471 & 0.3294 \\ 0.5000 & 0.4283 & 0.4769 & 0.5354 & 0.5302 \\ 1.5500 & 1.5126 & 1.5817 & 1.7608 & 1.7153 \end{bmatrix}, \quad \theta_e^1 = 2.9747,\ \theta_e^3 = 0.4567,$$

$$P_4 = \begin{bmatrix} 1.6040 & 1.9633 & 1.2940 & 1.2734 & 1.3000 \\ 0.5991 & 0.1190 & 0.8219 & 0.8927 & 0.9000 \\ 0.4747 & 1.0924 & 0.9235 & 0.8592 & 0.9000 \\ 1.6707 & 1.7201 & 1.1698 & 1.2732 & 1.3000 \end{bmatrix}, \quad T_{tot} = 1.1159$$

## 5th Order Parametrizations

### Step Length Long

#### Energy Optimized Solution

$$P_1 = \begin{bmatrix} -1.3000 & -1.3999 & -1.4966 & -1.5463 & -1.6095 & -1.6478 \\ -0.9000 & -0.7961 & -0.6871 & -0.5861 & -0.5325 & -0.5135 \\ -0.9000 & -0.8503 & -0.8107 & -0.8176 & -0.8109 & -0.4827 \\ -1.3000 & -1.4801 & -1.5928 & -1.5728 & -1.5625 & -1.9191 \end{bmatrix}, \quad f = 13.9812,$$

$$P_2 = \begin{bmatrix} -1.8191 & -1.8115 & -1.8410 & -1.7604 & -1.6652 & -1.5500 \\ -0.3822 & -0.1634 & -0.1629 & -0.2046 & -0.2594 & -0.4500 \\ -0.1258 & -0.5711 & -0.4619 & -0.3399 & -0.4698 & -0.6000 \\ -2.2136 & -1.9923 & -2.0216 & -2.0957 & -2.0143 & -1.6480 \end{bmatrix}, \quad \dot{\theta}_0 = 4.5251,$$

$$P_3 = \begin{bmatrix} 1.6480 & 1.5638 & 1.4993 & 1.5706 & 1.6011 & 1.6943 \\ 0.6000 & 0.5988 & 0.3950 & 0.5676 & 0.5583 & 0.2453 \\ 0.4500 & 0.4143 & 0.2978 & 0.4134 & 0.4114 & 0.3104 \\ 1.5500 & 1.4975 & 1.3614 & 1.4747 & 1.6055 & 1.7110 \end{bmatrix}, \quad \theta_e^1 = 2.9833,\ \theta_e^3 = 0.4179,$$

$$P_4 = \begin{bmatrix} 1.6608 & 1.1401 & 1.1542 & 1.1729 & 1.1325 & 1.3000 \\ 0.4780 & 0.9585 & 0.9440 & 0.9681 & 0.9520 & 0.9000 \\ 0.4600 & 0.8299 & 0.8300 & 0.8159 & 0.8070 & 0.9000 \\ 1.5938 & 1.1824 & 1.2012 & 1.2412 & 1.3019 & 1.3000 \end{bmatrix}, \quad T_{tot} = 2.7749$$

#### Velocity Optimized Solution

$$P_1 = \begin{bmatrix} -1.3000 & -1.4013 & -1.4583 & -1.5086 & -1.6589 & -1.6596 \\ -0.9000 & -0.8002 & -0.7003 & -0.5807 & -0.5206 & -0.5183 \\ -0.9000 & -0.8200 & -0.8208 & -0.8208 & -0.8201 & -0.4646 \\ -1.3000 & -1.5925 & -1.5938 & -1.5921 & -1.5857 & -1.9162 \end{bmatrix}, \quad f = 87.0384,$$

$$P_2 = \begin{bmatrix} -1.8191 & -1.8115 & -1.8154 & -1.8106 & -1.6140 & -1.5500 \\ -0.3822 & -0.1634 & -0.1637 & -0.2038 & -0.2538 & -0.4500 \\ -0.1258 & -0.5711 & -0.4721 & -0.3418 & -0.4722 & -0.6000 \\ -2.2136 & -1.9923 & -1.9962 & -1.9658 & -1.9363 & -1.6480 \end{bmatrix}, \quad \dot{\theta}_0 = 4.5500,$$

$$P_3 = \begin{bmatrix} 1.6480 & 1.5638 & 1.5096 & 1.5635 & 1.6024 & 1.7115 \\ 0.6000 & 0.5988 & 0.3903 & 0.5900 & 0.5595 & 0.2517 \\ 0.4500 & 0.4143 & 0.3021 & 0.4150 & 0.4148 & 0.3125 \\ 1.5500 & 1.4975 & 1.4068 & 1.4999 & 1.6002 & 1.7124 \end{bmatrix}, \quad \theta_e^1 = 2.8561,\ \theta_e^3 = 0.4003,$$

$$P_4 = \begin{bmatrix} 1.6608 & 1.1401 & 1.1425 & 1.1425 & 1.1424 & 1.3000 \\ 0.4780 & 0.9585 & 0.9585 & 0.9605 & 0.9604 & 0.9000 \\ 0.4600 & 0.8299 & 0.8308 & 0.8313 & 0.8317 & 0.9000 \\ 1.5938 & 1.1824 & 1.1844 & 1.2523 & 1.2681 & 1.3000 \end{bmatrix}, \quad T_{tot} = 1.5174$$

# APPENDIX B

## Bezier Curves

### Bezier function for Arbitrary Parameter Intervals

```
f := simplify((n!/(i!*(n-i)!))*(((t_1-t)/(t_1-t_0))^(n-i))
    *(((t-t_0)/(t_1-t_0))^i)*P[i+1]);
```

$$\frac{n!\left(-\frac{t-t_0}{t_0-t_1}\right)^i\left(\frac{t-t_1}{t_0-t_1}\right)^{n-i}P_{i+1}}{i!\,(n-i)!}$$

```
Pos := sum(f,i=0..n);
```

$$n!\left(\sum_{i=0}^{n}\frac{(-1)^i\left(\frac{t-t_1}{t_0-t_1}\right)^{n-i}P_{i+1}\,(t-t_0)^i}{i!\,(n-i)!\,(t_0-t_1)^i}\right)$$

### 2-Dimensional Bezier Curves

```
Bez_2_dim := (simplify(subs(Pos,n=2))):
```

```
Bez_2_temp := coeff(Bez_2_dim,t):
Bez_2 := matrix([Bez_2_temp[1],Bez_2_temp[2],Bez_2_temp[3]]);
```

$$\begin{pmatrix} \frac{P_1}{(t_0-t_1)^2}-\frac{2P_2}{(t_0-t_1)^2}+\frac{P_3}{(t_0-t_1)^2} \\ \frac{2(t_0+t_1)P_2}{(t_0-t_1)^2}-\frac{2t_1P_1}{(t_0-t_1)^2}-\frac{2t_0P_3}{(t_0-t_1)^2} \\ \frac{{t_1}^2P_1}{(t_0-t_1)^2}+\frac{{t_0}^2P_3}{(t_0-t_1)^2}-\frac{2t_0t_1P_2}{(t_0-t_1)^2} \end{pmatrix}$$

```
d_bez_2_dim := diff(Bez_2_dim,t):
dd_bez_2_dim := diff(d_bez_2_dim,t):
dbez_2 := subs(d_bez_2_dim,t=t_1);
ddbez_2 := subs(dd_bez_2_dim,t=t_1);
```

$$\frac{2P_2}{t_0-t_1}-\frac{(2t_0-2t_1)P_3}{(t_0-t_1)^2}$$

$$\frac{2P_1}{(t_0-t_1)^2}-\frac{4P_2}{(t_0-t_1)^2}+\frac{2P_3}{(t_0-t_1)^2}$$

### 3-Dimensional Bezier Curves

```
Bez_3_dim := (simplify(subs(Pos,n=3))):
```

```
Bez_3_temp := coeff(Bez_3_dim,t):
```

```
Bez_3 := matrix([Bez_3_temp[1],Bez_3_temp[2],Bez_3_temp[3],
    Bez_3_temp[4]]);
```

$$\begin{pmatrix} \frac{P_1}{(t_0-t_1)^3} - \frac{3P_2}{(t_0-t_1)^3} + \frac{3P_3}{(t_0-t_1)^3} - \frac{P_4}{(t_0-t_1)^3} \\ \frac{3t_0P_4}{(t_0-t_1)^3} - \frac{3t_1P_1}{(t_0-t_1)^3} + \frac{3(t_0+2t_1)P_2}{(t_0-t_1)^3} - \frac{3(2t_0+t_1)P_3}{(t_0-t_1)^3} \\ \frac{(t_0^2+2t_1t_0)P_3 3}{(t_0-t_1)^3} - \frac{(t_1^2+2t_0t_1)P_2 3}{(t_0-t_1)^3} + \frac{3t_1^2P_1}{(t_0-t_1)^3} - \frac{3t_0^2P_4}{(t_0-t_1)^3} \\ \frac{t_0^3P_4}{(t_0-t_1)^3} - \frac{t_1^3P_1}{(t_0-t_1)^3} + \frac{3t_0t_1^2P_2}{(t_0-t_1)^3} - \frac{3t_0^2t_1P_3}{(t_0-t_1)^3} \end{pmatrix}$$

```
dBez_3_dim := diff(Bez_3_dim,t):
ddBez_3_dim := diff(dBez_3_dim,t):
dBez_3  := subs(dBez_3_dim,t=t_1);
ddBez_3 := subs(ddBez_3_dim,t=t_1);
```

$$\frac{3P_3}{t_0-t_1} - \frac{3P_4}{t_0-t_1}$$

$$\frac{6P_2}{(t_0-t_1)^2} - \frac{6(2t_0-2t_1)P_3}{(t_0-t_1)^3} + \frac{3(2t_0-2t_1)P_4}{(t_0-t_1)^3}$$

### 4-Dimensional Bezier Curves

```
Bez_4_dim := (simplify(subs(Pos,n=4))):
```

```
Bez_4_temp := coeff(Bez_4_dim,t):
Bez_4 := matrix([Bez_4_temp[1],Bez_4_temp[2],Bez_4_temp[3],
    Bez_4_temp[4],Bez_4_temp[5]]);
```

$$\begin{pmatrix} \frac{P_1}{(t_0-t_1)^4} - \frac{4P_2}{(t_0-t_1)^4} + \frac{6P_3}{(t_0-t_1)^4} - \frac{4P_4}{(t_0-t_1)^4} + \frac{P_5}{(t_0-t_1)^4} \\ \frac{4(t_0+3t_1)P_2}{(t_0-t_1)^4} - \frac{4t_1P_1}{(t_0-t_1)^4} - \frac{4t_0P_5}{(t_0-t_1)^4} - \frac{6(2t_0+2t_1)P_3}{(t_0-t_1)^4} + \frac{4(3t_0+t_1)P_4}{(t_0-t_1)^4} \\ \frac{P_3(t_0^2+4t_0t_1+t_1^2)6}{(t_0-t_1)^4} + \frac{6t_1^2P_1}{(t_0-t_1)^4} + \frac{6t_0^2P_5}{(t_0-t_1)^4} - \frac{P_2(3t_1^2+3t_0t_1)4}{(t_0-t_1)^4} - \frac{P_4(3t_0^2+3t_1t_0)4}{(t_0-t_1)^4} \\ -\frac{(2t_0^2t_1+2t_0t_1^2)P_3 6}{(t_0-t_1)^4} - \frac{4t_1^3P_1}{(t_0-t_1)^4} - \frac{4t_0^3P_5}{(t_0-t_1)^4} + \frac{P_2(t_1^3+3t_0t_1^2)4}{(t_0-t_1)^4} + \frac{P_4(t_0^3+3t_1t_0^2)4}{(t_0-t_1)^4} \\ \frac{t_1^4P_1}{(t_0-t_1)^4} + \frac{t_0^4P_5}{(t_0-t_1)^4} - \frac{4t_0t_1^3P_2}{(t_0-t_1)^4} - \frac{4t_0^3t_1P_4}{(t_0-t_1)^4} + \frac{6t_0^2t_1^2P_3}{(t_0-t_1)^4} \end{pmatrix}$$

```
dBez_4_dim := diff(Bez_4_dim,t):
ddBez_4_dim := diff(dBez_4_dim,t):
dBez_4  := subs(dBez_4_dim,t=t_1);
ddBez_4 := subs(ddBez_4_dim,t=t_1);
```

$$\frac{4P_4}{t_0-t_1} - \frac{4P_5}{t_0-t_1}$$

$$\frac{12P_3}{(t_0-t_1)^2} - \frac{24P_4}{(t_0-t_1)^2} + \frac{12P_5}{(t_0-t_1)^2}$$

## 5-Dimensional Bezier Curves

```
Bez_5_dim := (simplify(subs(Pos,n=5))):
```

```
Bez_5_temp := coeff(Bez_5_dim,t):
Bez_5 := matrix([Bez_5_temp[1],Bez_5_temp[2],Bez_5_temp[3],
    Bez_5_temp[4],Bez_5_temp[5],Bez_5_temp[6]]);
```

$$\left[\left[\frac{P_1}{(t_0-t_1)^5}-\frac{5P_2}{(t_0-t_1)^5}+\frac{10P_3}{(t_0-t_1)^5}-\frac{10P_4}{(t_0-t_1)^5}+\frac{5P_5}{(t_0-t_1)^5}-\frac{P_6}{(t_0-t_1)^5}\right],\right.$$

$$\left[\frac{10(3t_0+2t_1)P_4}{(t_0-t_1)^5}-\frac{10(2t_0+3t_1)P_3}{(t_0-t_1)^5}-\frac{5t_1P_1}{(t_0-t_1)^5}+\frac{5t_0P_6}{(t_0-t_1)^5}+\frac{5(t_0+4t_1)P_2}{(t_0-t_1)^5}-\frac{5(4t_0+t_1)P_5}{(t_0-t_1)^5}\right],$$

$$\left[\frac{P_3(t_0^2+6t_0t_1+3t_1^2)10}{(t_0-t_1)^5}-\frac{P_4(3t_0^2+6t_0t_1+t_1^2)10}{(t_0-t_1)^5}+\frac{10t_1^2P_1}{(t_0-t_1)^5}-\frac{10t_0^2P_6}{(t_0-t_1)^5}-\frac{P_2(6t_1^2+4t_0t_1)5}{(t_0-t_1)^5}\right.$$

$$\left.+\frac{P_5(6t_0^2+4t_1t_0)5}{(t_0-t_1)^5}\right],$$

$$\left[\frac{(4t_1^3+\sigma_2)P_2 5}{(t_0-t_1)^5}-\frac{(4t_0^3+6t_1t_0^2)P_5 5}{(t_0-t_1)^5}-\frac{P_3(3t_0^2t_1+\sigma_2+t_1^3)10}{(t_0-t_1)^5}+\frac{P_4(t_0^3+6t_0^2t_1+3t_0t_1^2)10}{(t_0-t_1)^5}-\frac{10t_1^3P_1}{(t_0-t_1)^5}\right.$$

$$\left.+\frac{10t_0^3P_6}{(t_0-t_1)^5}\right],$$

$$\left[\frac{10(\sigma_1+2t_0t_1^3)P_3}{(t_0-t_1)^5}-\frac{(2t_0^3t_1+\sigma_1)P_4 10}{(t_0-t_1)^5}+\frac{5t_1^4P_1}{(t_0-t_1)^5}-\frac{5t_0^4P_6}{(t_0-t_1)^5}-\frac{P_2(t_1^4+4t_0t_1^3)5}{(t_0-t_1)^5}+\frac{P_5(t_0^4+4t_1t_0^3)5}{(t_0-t_1)^5}\right],$$

$$\left.\left[\frac{t_0^5P_6}{(t_0-t_1)^5}-\frac{t_1^5P_1}{(t_0-t_1)^5}+\frac{5t_0t_1^4P_2}{(t_0-t_1)^5}-\frac{5t_0^4t_1P_5}{(t_0-t_1)^5}-\frac{10t_0^2t_1^3P_3}{(t_0-t_1)^5}+\frac{10t_0^3t_1^2P_4}{(t_0-t_1)^5}\right]\right]$$

where

$$\sigma_1 = 3t_0^2t_1^2$$

$$\sigma_2 = 6t_0t_1^2$$

```
dBez_5_dim := diff(Bez_5_dim,t):
ddBez_5_dim := diff(dBez_5_dim,t):
dBez_5  := subs(dBez_5_dim,t=t_1);
ddBez_5 := subs(ddBez_5_dim,t=t_1);
```

$$\frac{5P_5}{t_0-t_1}-\frac{5P_6}{t_0-t_1}$$

$$\frac{20P_4}{(t_0-t_1)^2}-\frac{40P_5}{(t_0-t_1)^2}+\frac{20P_6}{(t_0-t_1)^2}$$

# APPENDIX C

## Dynamic system of a 5-DoF SemiQuad Robot

### Kinematic Model

```
Pref::abbreviateOutput(FALSE):
```

**System input, angles q1->q5**

```
qt[1] := q_1(t):
qt[2] := qt[1]+q_2(t):
qt[3] := qt[2]+q_3(t):
qt[4] := qt[3]+q_4(t):
qt[5] := qt[4]+q_5(t):
```

**Forward kinematics of the System**

```
P_C_M_x_1 := t->(l[1]-s[1])*cos(qt[1])+xe:
P_C_M_y_1 := t->(l[1]-s[1])*sin(qt[1])+ye:
P_C_M_x_2 := t->l[1]*cos(qt[1])+(l[2]-s[2])*cos(qt[2])+xe:
P_C_M_y_2 := t->l[1]*sin(qt[1])+(l[2]-s[2])*sin(qt[2])+ye:
P_C_M_x_3 := t->l[1]*cos(qt[1])+l[2]*cos(qt[2])
    +(l[3]-s[3])*cos(qt[3])+xe:
P_C_M_y_3 := t->l[1]*sin(qt[1])+l[2]*sin(qt[2])
    +(l[3]-s[3])*sin(qt[3])+ye:
P_C_M_x_4 := t->l[1]*cos(qt[1])+l[2]*cos(qt[2])
    +l[3]*cos(qt[3])+s[4]*cos(qt[4])+xe:
P_C_M_y_4 := t->l[1]*sin(qt[1])+l[2]*sin(qt[2])
    +l[3]*sin(qt[3])+s[4]*sin(qt[4])+ye:
P_C_M_x_5 := t->l[1]*cos(qt[1])+l[2]*cos(qt[2])
    +l[3]*cos(qt[3])+l[4]*cos(qt[4])+s[5]*cos(qt[5])+xe:
P_C_M_y_5 := t->l[1]*sin(qt[1])+l[2]*sin(qt[2])
    +l[3]*sin(qt[3])+l[4]*sin(qt[4])+s[5]*sin(qt[5])+ye:
P_C_M_x_e := t->l[1]*cos(qt[1])+l[2]*cos(qt[2])
    +l[3]*cos(qt[3])+l[4]*cos(qt[4])+l[5]*cos(qt[5])+xe:
P_C_M_y_e := t->l[1]*sin(qt[1])+l[2]*sin(qt[2])
    +l[3]*sin(qt[3])+l[4]*sin(qt[4])+l[5]*sin(qt[5])+ye:
```

```
P_C_M := matrix(
    [[P_C_M_x_1 ,P_C_M_x_2,P_C_M_x_3,P_C_M_x_4,P_C_M_x_5,P_C_M_x_e],
    [P_C_M_y_1,P_C_M_y_2,P_C_M_y_3,P_C_M_y_4,P_C_M_y_5,P_C_M_y_e]]
):
```

### Dynamic Model

**I have chosen to implement the dynamic model in two arbitrary ways, to act as a fail-safe against potential errors.**

## Method 1: Energy Calculations\Euler-Lagrange

### Kinematic Energy:

### Velocity of COM:

```
V_C_M[1,1] := P_C_M[1,1]'(t):
V_C_M[2,1] := P_C_M[2,1]'(t):
V_C_M[1,2] := P_C_M[1,2]'(t):
V_C_M[2,2] := P_C_M[2,2]'(t):
V_C_M[1,3] := P_C_M[1,3]'(t):
V_C_M[2,3] := P_C_M[2,3]'(t):
V_C_M[1,4] := P_C_M[1,4]'(t):
V_C_M[2,4] := P_C_M[2,4]'(t):
V_C_M[1,5] := P_C_M[1,5]'(t):
V_C_M[2,5] := P_C_M[2,5]'(t):
```

### Kinematic Energy of Joints:

```
Dqt[1] := q_1'(t):
Dqt[2] := Dqt[1]+q_2'(t):
Dqt[3] := Dqt[2]+q_3'(t):
Dqt[4] := Dqt[3]+q_4'(t):
Dqt[5] := Dqt[4]+q_5'(t):
```

```
K_r[1] := 0.5*m[1]*(V_C_M[1,1]^2 + V_C_M[2,1]^2) + 0.5*I_r[1]*Dqt[1]^2:
K_r[2] := 0.5*m[2]*(V_C_M[1,2]^2 + V_C_M[2,2]^2) + 0.5*I_r[2]*Dqt[2]^2:
K_r[3] := 0.5*m[3]*(V_C_M[1,3]^2 + V_C_M[2,3]^2) + 0.5*I_r[3]*Dqt[3]^2:
K_r[4] := 0.5*m[4]*(V_C_M[1,4]^2 + V_C_M[2,4]^2) + 0.5*I_r[4]*Dqt[4]^2:
K_r[5] := 0.5*m[5]*(V_C_M[1,5]^2 + V_C_M[2,5]^2) + 0.5*I_r[5]*Dqt[5]^2:
```

```
K_r_tot := K_r[1]+K_r[2]+K_r[3]+K_r[4]+K_r[5]:
```

### Kinematic Energy of Motors

```
K_m[1] := 0.5*N*I_m[1]*(q_2'(t))^2:
K_m[2] := 0.5*N*I_m[2]*(q_3'(t))^2:
K_m[3] := 0.5*N*I_m[3]*(q_4'(t))^2:
K_m[4] := 0.5*N*I_m[4]*(q_5'(t))^2:
```

```
K_m_tot := K_m[1]+K_m[2]+K_m[3]+K_m[4]:
```

### Total Kinematic Energy

```
K_tot :=  K_r_tot + K_m_tot:
```

### Potential Energy

```
P[1] := g*m[1]*P_C_M[2,1](t):
P[2] := g*m[2]*P_C_M[2,2](t):
P[3] := g*m[3]*P_C_M[2,3](t):
P[4] := g*m[4]*P_C_M[2,4](t):
P[5] := g*m[5]*P_C_M[2,5](t):
```

```
P_tot := P[1]+P[2]+P[3]+P[4]+P[5]:
```

### Lagrangian.

```
L := t-> K_tot - P_tot:
```

```
F_1_1 := t->subs(diff(subs(L(t),q_1'(t)=Temp),Temp),Temp=q_1'(t)):
F_1_2 := t-> F_1_1'(t):
F_1_3 := t-> subs(diff(subs(L(t),q_1(t)=Temp),Temp),Temp=q_1(t)):
F_2_1 := t->subs(diff(subs(L(t),q_2'(t)=Temp),Temp),Temp=q_2'(t)):
F_2_2 := t-> F_2_1'(t):
F_2_3 := t-> subs(diff(subs(L(t),q_2(t)=Temp),Temp),Temp=q_2(t)):
F_3_1 := t->subs(diff(subs(L(t),q_3'(t)=Temp),Temp),Temp=q_3'(t)):
F_3_2 := t-> F_3_1'(t):
F_3_3 := t-> subs(diff(subs(L(t),q_3(t)=Temp),Temp),Temp=q_3(t)):
F_4_1 := t->subs(diff(subs(L(t),q_4'(t)=Temp),Temp),Temp=q_4'(t)):
F_4_2 := t-> F_4_1'(t):
F_4_3 := t-> subs(diff(subs(L(t),q_4(t)=Temp),Temp),Temp=q_4(t)):
F_5_1 := t->subs(diff(subs(L(t),q_5'(t)=Temp),Temp),Temp=q_5'(t)):
F_5_2 := t-> F_5_1'(t):
F_5_3 := t-> subs(diff(subs(L(t),q_5(t)=Temp),Temp),Temp=q_5(t)):
```

### Dynamic Equations:

```
DY1_1 := t-> F_1_2(t)-F_1_3(t):
DY2_1 := t-> F_2_2(t)-F_2_3(t):
DY3_1 := t-> F_3_2(t)-F_3_3(t):
DY4_1 := t-> F_4_2(t)-F_4_3(t):
DY5_1 := t-> F_5_2(t)-F_5_3(t):
```

## Method 2: Matrix Calculations

### M matrix

#### Inertia Matrix

```
IMat := matrix(
[[I_r[1]+I_r[2]+I_r[3]+I_r[4]+I_r[5],I_r[2]+I_r[3]+I_r[4]+I_r[5],
I_r[3]+I_r[4]+I_r[5],I_r[4]+I_r[5],I_r[5]],
[I_r[2]+I_r[3]+I_r[4]+I_r[5],N*I_m[1]+I_r[2]+I_r[3]+I_r[4]+I_r[5],
I_r[3]+I_r[4]+I_r[5],I_r[4]+I_r[5],I_r[5]],[I_r[3]+I_r[4]+I_r[5],
I_r[3]+I_r[4]+I_r[5],I_m[2]*N+I_r[3]+I_r[4]+I_r[5],I_r[4]+I_r[5],
I_r[5]],[I_r[4]+I_r[5],I_r[4]+I_r[5],I_r[4]+I_r[5],
I_m[3]*N+I_r[4]+I_r[5],I_r[5]],[I_r[5],I_r[5],I_r[5],I_r[5],
I_m[4]*N+I_r[5]]]):
```

### Calculating the Jacobians

```
Dim := 5:
for i from 1 to Dim do
Jvc[i] := matrix(
[[subs(diff(subs(P_C_M[1,i](t),q_1(t)=temp),temp),temp=q_1(t)),
subs(diff(subs(P_C_M[1,i](t),q_2(t)=temp),temp),temp=q_2(t)),
subs(diff(subs(P_C_M[1,i](t),q_3(t)=temp),temp),temp=q_3(t)),
subs(diff(subs(P_C_M[1,i](t),q_4(t)=temp),temp),temp=q_4(t)),
subs(diff(subs(P_C_M[1,i](t),q_5(t)=temp),temp),temp=q_5(t))],
[subs(diff(subs(P_C_M[2,i](t),q_1(t)=temp),temp),temp=q_1(t)),
subs(diff(subs(P_C_M[2,i](t),q_2(t)=temp),temp),temp=q_2(t)),
subs(diff(subs(P_C_M[2,i](t),q_3(t)=temp),temp),temp=q_3(t)),
subs(diff(subs(P_C_M[2,i](t),q_4(t)=temp),temp),temp=q_4(t)),
subs(diff(subs(P_C_M[2,i](t),q_5(t)=temp),temp),temp=q_5(t))]]);
```

```
end_for:
delete i
```

```
M := m[1]*transpose(Jvc[1])*Jvc[1]+m[2]*transpose(Jvc[2])*Jvc[2]
    +m[3]*transpose(Jvc[3])*Jvc[3]+m[4]*transpose(Jvc[4])*Jvc[4]
    +m[5]*transpose(Jvc[5])*Jvc[5] + IMat:
```

### C matrix

```
q := matrix([q_1(t),q_2(t),q_3(t),q_4(t),q_5(t),xe(t),ye(t)]):
dq := matrix([q_1'(t),q_2'(t),q_3'(t),q_4'(t),q_5'(t),xe'(t),ye'(t)]):
ddq:= matrix(
[q_1''(t),q_2''(t),q_3''(t),q_4''(t),q_5''(t),xe''(t),ye''(t)]):
```

```
C := matrix(Dim,Dim):
for k from 1 to Dim do
         for j from 1 to Dim do
              for i from 1 to Dim do
                  C[k,j] := C[k,j] +
                       (0.5*(subs(diff(subs(M[k,j],q[i]=temp),temp),
                       temp=q[i])+subs(diff(subs(M[k,i],q[j]=temp),
                       temp),temp=q[j])-subs(diff(subs(M[i,j],
                       q[k]=temp),temp),temp=q[k]))*dq[i]);
              end_for
         end_for
end_for:
delete i,k,j
```

### G matrix

```
G := matrix(5,1):
for i from 1 to Dim do
         G[i] := subs(diff(subs(P_tot,q[i]=temp),temp),temp=q[i])
end_for:
```

```
DYN_SYST_2 := M*ddq[1..5] + C*dq[1..5] + G:
```

```
DY1_2 := DYN_SYST_2[1]:
```

## Reduced Dynamic System

**The reduces system variables:**

```
th := Symbol::theta:
q1 := t->th(t):
q2 := t->Symbol::subScript(Symbol::phi,1)(th(t)):
q3 := t->Symbol::subScript(Symbol::phi,2)(th(t)):
q4 := t->Symbol::subScript(Symbol::phi,3)(th(t)):
q5 := t->Symbol::subScript(Symbol::phi,4)(th(t)):
dq1 := t->q1'(t):
dq2 := t->q2'(t):
dq3 := t->q3'(t):
dq4 := t->q4'(t):
dq5 := t->q5'(t):
ddq1:= t->q1''(t):
ddq2:= t->q2''(t):
```

```
ddq3:= t->q3''(t):
ddq4:= t->q4''(t):
ddq5:= t->q5''(t):
phi_1   := Symbol::subScript(Symbol::phi,1)(th):
phi_2   := Symbol::subScript(Symbol::phi,2)(th):
phi_3   := Symbol::subScript(Symbol::phi,3)(th):
phi_4   := Symbol::subScript(Symbol::phi,4)(th):
dphi_1  := Symbol::subScript(Symbol::phi,1)'(th):
dphi_2  := Symbol::subScript(Symbol::phi,2)'(th):
dphi_3  := Symbol::subScript(Symbol::phi,3)'(th):
dphi_4  := Symbol::subScript(Symbol::phi,4)'(th):
ddphi_1 := Symbol::subScript(Symbol::phi,1)''(th):
ddphi_2 := Symbol::subScript(Symbol::phi,2)''(th):
ddphi_3 := Symbol::subScript(Symbol::phi,3)''(th):
ddphi_4 := Symbol::subScript(Symbol::phi,4)''(th):
```

**Inserted into the first equations:**

```
EQ1_temp_1 := t->subs
(DY1_1(t),q_1(t)=q1(t),q_2(t)=q2(t),q_3(t)=q3(t),q_4(t)=q4(t),
q_5(t)=q5(t),q_1'(t)=dq1(t),q_2'(t)=dq2(t),q_3'(t)=dq3(t),
q_4'(t)=dq4(t),q_5'(t)=dq5(t),q_1''(t)=ddq1(t),q_2''(t)=ddq2(t),
q_3''(t)=ddq3(t),q_4''(t)=ddq4(t),q_5''(t)=ddq5(t)):
```

```
EQ1_temp_2 := t->subs(DY1_2,q_1(t)=q1(t),q_2(t)=q2(t),q_3(t)=q3(t),
q_4(t)=q4(t),q_5(t)=q5(t),q_1'(t)=dq1(t),q_2'(t)=dq2(t),
q_3'(t)=dq3(t),q_4'(t)=dq4(t),q_5'(t)=dq5(t),q_1''(t)=ddq1(t),
q_2''(t)=ddq2(t),q_3''(t)=ddq3(t),q_4''(t)=ddq4(t),q_5''(t)=ddq5(t)):
```

## Alfa-Beta-Gamma Equation

**Inserting for Phi - 2-dimensional case:**

```
Phi_1   := PQ[1,1]*th^2 + PQ[1,2]*th + PQ[1,3]:
dPhi_1  := 2*PQ[1,1]*th + PQ[1,2]:
ddPhi_1 := 2*PQ[1,1]:
Phi_2   := PQ[2,1]*th^2 + PQ[2,2]*th + PQ[2,3]:
dPhi_2  := 2*PQ[2,1]*th + PQ[2,2]:
ddPhi_2 := 2*PQ[2,1]:
Phi_3   := PQ[3,1]*th^2 + PQ[3,2]*th + PQ[3,3]:
dPhi_3 := 2*PQ[3,1]*th + PQ[3,2]:
ddPhi_3 := 2*PQ[3,1]:
Phi_4   := PQ[4,1]*th^2 + PQ[4,2]*th + PQ[4,3]:
dPhi_4  := 2*PQ[4,1]*th + PQ[4,2]:
ddPhi_4 := 2*PQ[4,1]:
```

```
EQ1_1 := subs(EQ1_temp_1(t),q1(t)=th,q1'(t)=th',q1''(t)=th'',
    phi_1=Phi_1,phi_2=Phi_2,phi_3=Phi_3,phi_4=Phi_4,dphi_1=dPhi_1,
    dphi_2=dPhi_2,dphi_3=dPhi_3,dphi_4=dPhi_4,ddphi_1=ddPhi_1,
    ddphi_2=ddPhi_2,ddphi_3=ddPhi_3,ddphi_4=ddPhi_4):
```

```
EQ1_2 := subs(EQ1_temp_2(t),q1(t)=th,q1'(t)=th',q1''(t)=th'',
    phi_1=Phi_1,phi_2=Phi_2,phi_3=Phi_3,phi_4=Phi_4,dphi_1=dPhi_1,
    dphi_2=dPhi_2,dphi_3=dPhi_3,dphi_4=dPhi_4,ddphi_1=ddPhi_1,
    ddphi_2=ddPhi_2,ddphi_3=ddPhi_3,ddphi_4=ddPhi_4):
```

```
Beta_1 := coeff(EQ1_1,th',2):
alfa_1 := coeff(EQ1_1,th'',1):
Gamma_1 := subs(EQ1_1,th''=0,th'=0):

Beta_2 := coeff(EQ1_2,th',2):
alfa_2 := coeff(EQ1_2,th'',1):
Gamma_2 := subs(EQ1_2,th''=0,th'=0):
```

**Changing the variable to an allowed matlab variable**

```
alfa_out_1 := subs(alfa_1,th=the):
Beta_out_1 := subs(Beta_1,th=the):
Gamma_out_1 := subs(Gamma_1,th=the):

alfa_out_2 := subs(alfa_2,th=the):
Beta_out_2 := subs(Beta_2,th=the):
Gamma_out_2 := subs(Gamma_2,th=the):
```

## Torque

**The remainding 4 equations can be seen as a vector tau with the 4 torques in the actuated joints.**

```
Rem_Dyn_Syst_temp1 := matrix(
[[DY2_1(t)],[DY3_1(t)],[DY4_1(t)],[DY5_1(t)]]):
```

```
Rem_Dyn_Syst_temp2 := subs(
    Rem_Dyn_Syst_temp1,q_1(t)=q1(t),q_2(t)=q2(t),q_3(t)=q3(t),
    q_4(t)=q4(t),q_5(t)=q5(t),q_1'(t)=dq1(t),q_2'(t)=dq2(t),
    q_3'(t)=dq3(t),q_4'(t)=dq4(t),q_5'(t)=dq5(t),q_1''(t)=ddq1(t),
    q_2''(t)=ddq2(t),q_3''(t)=ddq3(t),q_4''(t)=ddq4(t),
    q_5''(t)=ddq5(t)):
```

```
Rem_Dyn_Syst := subs(Rem_Dyn_Syst_temp2,q1(t)=th,q1'(t)=th',
    q1''(t)=th'',phi_1=Phi_1,phi_2=Phi_2,phi_3=Phi_3,phi_4=Phi_4,
    dphi_1=dPhi_1,dphi_2=dPhi_2,dphi_3=dPhi_3,dphi_4=dPhi_4,
    ddphi_1=ddPhi_1,ddphi_2=ddPhi_2,ddphi_3=ddPhi_3,
    ddphi_4=ddPhi_4):
```

```
Rem_Dyn_Syst_out := subs(Rem_Dyn_Syst,th''=ddthe,th'=dthe,th=the):
```

## Impact Model

**Extended M matrix**

```
for i from 1 to 5 do
         V_C_M_Ext[1,i] := V_C_M[1,i] + xe'(t);
         V_C_M_Ext[2,i] := V_C_M[2,i] + ye'(t);
         K_r_Ext[i] := 0.5*m[i]*(V_C_M_Ext[1,i]^2
                       + V_C_M_Ext[2,i]^2) + 0.5*I_r[i]*Dqt[i]^2;
end_for:
delete i;
```

```
K_tot_Ext := K_m_tot+K_r_Ext[1]+K_r_Ext[2]
```

```
            +K_r_Ext[3]+K_r_Ext[4]+K_r_Ext[5]:
P_tot_Ext := subs(P_tot,ye = ye(t)):
L_Ext := K_tot_Ext - P_tot_Ext:
for i from 1 to 7 do
    F_Ext_temp_1 := t->subs(
        diff(subs(L_Ext,dq[i]=Temp),Temp),Temp=dq[i]);
        F_Ext_temp_2 := F_Ext_temp_1'(t);
        F_Ext_temp_3 := subs(diff(subs(L_Ext,q[i]=Temp),Temp),Temp=q[i])
        Dyn_Syst_Ext[i] := F_Ext_temp_2 - F_Ext_temp_3;
end_for:
delete i

Dyn_Syst_Ext_temp := matrix(
    [Dyn_Syst_Ext[1],Dyn_Syst_Ext[2],Dyn_Syst_Ext[3],Dyn_Syst_Ext[4],
    Dyn_Syst_Ext[5],Dyn_Syst_Ext[6],Dyn_Syst_Ext[7]]):
Dyn_Syst_Ext := subs(Dyn_Syst_Ext_temp,ddq[1]=d2q[1],ddq[2]=d2q[2],
    ddq[3]=d2q[3],ddq[4]=d2q[4],ddq[5]=d2q[5],ddq[6]=d2q[6],ddq[7]=d2q[7

M_Ext := matrix(7,7):
for i from 1 to 7 do
         for j from 1 to 7 do
                  M_Ext[i,j] := diff(Dyn_Syst_Ext[i],d2q[j]);
         end_for:
end_for:
delete i,j

M_Ext_Out_Temp := subs(M_Ext,q_2(t)=Phi_1,q_3(t)=Phi_2,
                  q_4(t)=Phi_3,q_5(t)=Phi_4):
M_Ext_Out := subs(M_Ext_Out_Temp,q_1(t)=the):

M_1_1 := M_Ext[1..5,1..5]:
M_1_2 := M_Ext[1..5,6..7]:
M_2_2 := M_Ext[6..7,6..7]:
```

**Jacobian of End point**

```
Jvc_end := subs(Jvc[5],s[5]=l[5]):
```

**Matrices U1 and U1 defines the Update law**

```
U_1 := M_1_1 - M_1_2*Jvc_end -
       transpose(Jvc_end)*(transpose(M_1_2)-M_2_2*Jvc_end):
U_2 := M_1_1 - transpose(Jvc_end)*transpose(M_1_2):

U_1_temp := subs(U_1,q_1(t)=th,q_2(t)=Phi_1,q_3(t)=Phi_2,
            q_4(t)=Phi_3,q_5(t)=Phi_4):
U_2_temp := subs(U_2,q_1(t)=th,q_2(t)=Phi_1,q_3(t)=Phi_2,
             q_4(t)=Phi_3,q_5(t)=Phi_4):

U_1_Out := subs(U_1_temp,th=the):
U_2_Out := subs(U_2_temp,th=the):
```