NTNU
Norwegian University of
Science and Technology

Planning and Control of Locomotion for a Quadruped

Studying the Curvet Gait

Stian Lode

Master of Science in Engineering Cybernetics
Submission date: June 2009
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

This project is focused on two problems: developing fundaments for efficient gait generation for quadruped robots and developing methods that allow orbital stabilization of newly shaped motions.

The results are to be exemplified and illustrated on the dynamical model of SemiQuad, which is a quadrupedal walker prototype recently developed and built at Ecole de Nantes in France

Assignment given: 12. January 2009
Supervisor: Anton Shiriaev, ITK

Abstract

In many ways, the simple act of walking is one of the most complex modes of locomotion there is. For control-system scientists the periodic hybrid dynamical nature of walking systems presents a number of unique challenges, many of which still lack satisfying solutions. This thesis applies fairly recent concepts of motion generation and control to generate steps and gaits for such a walking robotic system. The robot, SemiQuad, developed and built at École de Nantes in France, is a five degree of freedom, underactuated periodic hybrid dynamical system.

This text presents a generic method of reparametrizing a given smooth motion by the use of virtual holonomic constraints, and comments on the conditions required for the method to succeed. It is then shown how virtual holonomic constraints can be generated from scratch, and certain properties of holonomically constrained systems are investigated. From the generated constraints and associated motion, a controller based on the principle of transverse linearization is created, and closed loop characteristics of the system are observed.

Preface

This masters thesis is written to document the work and research done throughout the 10th and final term of the Master of Engineering Cybernetics degree at the Norwegian University of Science and Technology (NTNU). A personal interest in biologically inspired robotics was the motivation behind choosing legged locomotion as my object of research; however, the prospect of being allowed to contribute to such a bustling field of research obviously had its merits.

I would like to thank my supervisor at NTNU, Anton Shiriaev, not only for generously sharing his ideas and insights, but also for his ability to provide inspiration at times when such is scarce.

Stian Lode
Trondheim, June 15, 2009

Contents

List of Figures

Introduction

Motivation

The scientific endeavor to achieve legged locomotion is a work in progress. Useful theoretical notions such as passive dynamics and zero moment point have enabled us to create walking robots; however, state-of-the-art walking robots of today still leave a lot to be desired when compared to their biological counterparts. Robustness and efficiency are key elements on which to work on to bridge the gap.

Walking dynamical systems present unique challenges for control-system scientists for a number of reasons. First of all, a walking dynamical system follows a periodic motion, and stabilizing this type of motion is quite different from, and arguably more difficult than, stabilizing an equilibrium. Second, a proper walking system will often be desired to have intervals of the gait period where the robot is locally unstable, prompting the definition of new concepts of stability. Finally, a walking system is a natural hybrid system, having intervals of continuous dynamics, and intervals of discrete impacts. There seems to be a lack of generic mathematical principles for systems like this, and solutions are often of an ad hoc nature.

However, recently, extensive research has been done presenting new tools designed to cope with systems of this type. Papers on virtual holonomic constraints [3, 6, 12, 14] have introduced new concepts and methods for design, analysis and controller synthesis for underactuated, periodic hybrid dynamical systems.

However, in papers where virtual holonomic constraints are used alongside controllers based upon transverse linearization examples are often shown for simple systems, and this thesis therefore contributes by applying this theory of motion generating and orbital stabilization to a larger, five degree of freedom system, SemiQuad.

Background

Control-system scientists are, when all is said and done, concerned with stability. The scientific field of legged locomotion is graced with several such concepts, from the overly restrictive to the not so restrictive where the only real criterion of stability is to "not fall down"[9]. The oldest paradigm of walking stability is that of *static stability*. Static stability means that the center of mass should be kept within the support polygon spanned by the legs. This is a very restrictive stability criteria, and drastically limits the speed attainable. A huge leap forward was therefore made when Vukobratović in the early 70's introduced the concept of zero moment point. As the name implies, the zero moment point (ZMP) criterion demands that the robots center of pressure at all times lies beneath the stance leg. That is, beneath the stance leg there should be a point about which there is no moment. The ZMP-criterion is far less restrictive than static stability as it allows the center of mass to move outside the support polygon, but it still leads to awkward looking motion of speed and efficiency far inferior to that of human motion. [9]

This leads us to the seminal paper of McGeer [10], published in 1990, which introduces the concept of passive dynamic walking. Stability in the context of passive dynamic walking is seen in light of the stability of the limit cycle associated with the motion. This approach sparked a renewed interest in the field of legged locomotion[7, 16, 17], and has been shown to generate fluid gaits with impressive results in terms of efficiency and passive stability.

In 2008 Freidovich, Mettin, Shiriaev and Spong [6] showed how to apply the theory of virtual holonomic constraints to analyze the passive gait of the compass-gait biped. The concepts of virtual constraints and transverse linearized controllers are fairly recent, and are as tools very well suited for periodic, hybrid dynamical systems [3, 6, 12, 14]. Applying this theory to the legged robot, SemiQuad, therefore seemed like a natural contribution to the field of legged locomotion.

Project scope

This thesis will apply several tools related to the concept of virtual holonomic constraints to generate motion for a robot SemiQuad, developed at École de Nantes in France. First, an effort will be made to convert the original motion devised at École de Nantes to a parameterization based on virtual holonomic constraints. Then motion will be generated from scratch in a general way that allows for simple optimization. Then, a controller will be constructed using a control methodology called transversal linearization. Due to particular properties of virtually constrained systems this type of controller has been showed to work well for such systems.

Paper layout

Section 1 covers basic material related to dynamical systems and virtual holonomic constraints. Section 2 then proceeds with a short review of the SemiQuad system, both of the original dynamics and the new dynamics. Section 3 presents a generic method of reparameterizing the motion of a dynamical system to virtual holonomic constraints. Section 4 covers the basic methodology of generating virtual holonomic constraints from scratch, and an example gait for SemiQuad is generated. Section 5 briefly restates on the theory of transversal linearization of dynamical systems, and outlines the design of a controller designed to ensure orbital stability of the motions designed previously. Section 6 briefly looks into the closed-loop behavior of the controlled system. Section 7 contains a brief discussion of the research done throughout this thesis. Section 8 ends the thesis with a brief conclusion and a selection of prospects for future work.

1 Preliminaries

The theory covered within this thesis requires a certain theoretical foundation. This section starts with a brief restatement of the Euler-Lagrange Formalism and how it applies to dynamical systems. From there, as an extension of the Euler-Lagrange theory, the subject of virtual holonomic constraints will be explained throughout the last part of this section.

1.1 Euler-Lagrange Formalism

The Euler-Lagrange Formalism provides a structured and elegant way of modeling mechanical systems.

1.1.1 Dynamics of rigid bodies

The dynamics of robotic systems like the one described in this text are commonly classified by the term *rigid body dynamics*. A rigid body is a non-deformable object with an associated mass, a center of gravity, a moment of inertia, and both translational and rotational degrees of freedom. A robotic system will usually be composed of several rigid bodies – *links* – interconnected by either revolving or prismatic joints. Revolving joints enable the links to rotate relative to each other, and prismatic joints allow for links to extend and retract. Each joint adds a degree of freedom (DOF) to the system, and each joint is associated with a *generalized coordinate* describing the state of the joint. For each of the actuated joints there will also be a *generalized force* or *generalized torque*. The generalized coordinates describe how the links relate to each other, and how the links relate to the surrounding environment. [11, 15]

1.1.2 The Euler-Lagrange equations

The purpose of using the Euler-Lagrange Formalism is to obtain equations which describe the relation between the *generalized coordinates* and the *generalized forces* of the rigid body system. These equations will be second order differential equations, one for each of the generalized coordinates. To develop dynamical equations through the use of the Euler-Lagrange Formalism one starts by formulating expressions of the systems' total kinetic and potential energy in terms of generalized coordinates. For systems of more than one degree of freedom, these quantities are often written on the matrix form

$$\mathcal{K}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T M(\mathbf{q})\dot{\mathbf{q}} \tag{1}$$

and

$$\mathcal{V}(\mathbf{q}) = \sum_i m_i \mathbf{g}^T r_{ci} \tag{2}$$

where $M(\mathbf{q})$ is a matrix composed of the masses and inertias of the links, $\mathbf{q}$ is a vector of the generalized coordinates, m_i is the mass of each link, $\mathbf{g}$ is a vector specifying the direction and magnitude of gravity in the inertial frame of reference, and r_ci is the position of the center of mass of link i.

From these quantities we define the *Lagrangian* as $\mathcal{L} = \mathcal{K} - \mathcal{T}$, and the dynamical equations are found by solving [15, 11]

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

Performing the necessary calculus upon the above expression for each of the generalized coordinates yields a system of differential equations often presented on the form

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) = B(\mathbf{q})\tau \tag{4}$$

where $\mathbf{q}$ and $M(\mathbf{q})$ are defined as before, $C(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis matrix and $G(\mathbf{q})$ is a matrix of gravitational forces. $B(\mathbf{q})\tau$ is the generalized

force being applied to each of the actuated degrees of freedom. If there are unactuated degrees of freedom, as with the SemiQuad, the system is labeled *non-holonomic*. Otherwise, the system is labeled *holonomic*. A prime example of a holonomic system is the human arm. In fact, in terms of degrees of freedom the arm is *redundant*, or *overactuated*, having more degrees of freedom than strictly necessary to maneuver the hand within its workspace[1]. For further details on the Euler-Lagrange Formalism and rigid body dynamics the reader is referred to [15] and [11].

1.1.3 Computational simplifications

While calculating (3) will yield (4), several tricks can be used to lessen the computational burden. First off all, from the expression for kinetic energy (1) it can be seen that $M(\mathbf{q})$ can be extracted by computing the Hessian of $\mathcal{K}(\mathbf{q},\, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^T M(\mathbf{q})\dot{\mathbf{q}}$ with respect to the vector $\dot{\mathbf{q}}$. This procedure is implemented by computing the kinetic energy Jacobian twice, transposing the intermediary result between the two Jacobians.

$$M(\mathbf{q}) = \begin{bmatrix} \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_1 \partial \dot{\mathbf{q}}_1} & \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_1 \partial \dot{\mathbf{q}}_2} & \cdots & \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_1 \partial \dot{\mathbf{q}}_n} \\ \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_2 \partial \dot{\mathbf{q}}_1} & \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_2 \partial \dot{\mathbf{q}}_2} & \cdots & \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_1 \partial \dot{\mathbf{q}}_n} \\ & & \vdots & \\ \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_n \partial \dot{\mathbf{q}}_1} & \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_n \partial \dot{\mathbf{q}}_2} & \cdots & \frac{\partial^2 \mathcal{K}}{\partial \dot{\mathbf{q}}_n \partial \dot{\mathbf{q}}_n} \end{bmatrix} \tag{5}$$

With $M(\mathbf{q})$ extracted, the Coriolis matrix $C(\mathbf{q},\, \dot{\mathbf{q}})$ can be obtained by calculating the *Christoffel Symbols* on the form

$$C_{k,\,j} = \frac{1}{2}\left(\frac{\partial}{\partial q_i} M_{k,\,j} + \frac{\partial}{\partial q_j} M_{k,\,i} - \frac{\partial}{\partial q_k} M_{i,\,j}\right)\dot{q}_i \tag{6}$$

[1]Workspace: The workspace of a manipulator is the total volume spanned by all possible joint configurations.

for all elements $C_{j,k}$ for $j, k = 1, \ldots, n$ of $C(\mathbf{q}, \dot{\mathbf{q}})$. Finally, $G(\mathbf{q})$ can be computed as the Jacobian of the potential energy $\mathcal{V}$ with respect to $\mathbf{q}$

$$G(q) = \begin{bmatrix} \frac{\partial \mathcal{V}}{\partial q_1} \\ \frac{\partial \mathcal{V}}{\partial q_2} \\ \vdots \\ \frac{\partial \mathcal{V}}{\partial q_n} \end{bmatrix}. \tag{7}$$

These steps are easily implemented in a symbolic math environment like Maple or in the symbolic toolbox of Matlab, and results in a set of differential equations on matrix form. Formulating the equations in such a way makes the system easier to handle and analyze.

1.1.4 Contact forces

To account for external contact forces, the system needs to be augmented. In this text, which ultimately is about walking robots, contact forces arise where and when the feet touch the floor. Monitoring these forces is relevant, as it will tell us about when the feet lift off the floor, and whether or not the feet could end up sliding because of large horizontal forces. It is important to ensure that the ratio between the normal and horizontal forces are less than some assumed friction coefficient. This range of valid ratios is commonly labeled the *friction cone*.

The contact forces are initially formulated [11, Chapter 6] as constraints on the form $R(\mathbf{q}) = k$ where $R(\mathbf{q})$ are the forward kinematics of the contact point, and k is the point where the contact point is to remain stationary. These constraints are then differentiated into Pfaffian constraints on the form $A(\mathbf{q})\dot{\mathbf{q}} = 0$ where $A(\mathbf{q})$ now is the Jacobian of the point in touch with the environment. To solve (4) with these constraints, the constraints are differentiated again into $A(\mathbf{q})\ddot{\mathbf{q}} + A'(\mathbf{q})\dot{\mathbf{q}}^2 = 0$, and then the augmented

dynamics are rewritten as

$$\begin{bmatrix} M & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda \end{bmatrix} = - \begin{bmatrix} C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) - \tau \\ A'\dot{\mathbf{q}}^2 \end{bmatrix} \tag{8}$$

which when solved numerically will yields not only the constrained dynamics, but also the forces λ required to satisfy the constraints.

1.2 Virtual holonomic constraints

Virtual holonomic constraints is a powerful albeit fairly simple concept. It paves the way for several mathematical tools which can simplify the design and control of robotic systems. Holonomic constraints are geometric relations on the form $R(q) = 0$ which in some way limit the positional freedom of the system. They are as such are a natural part of most robotic systems. As an example, a joint fixed at a certain angle or that through some mechanical means is limited to some sort of predefined path is subjected to holonomic constraints. Now, if the joint is actuated and the constraint is maintained through actuator control, the joint is *virtually* constrained.

The method of using virtual holonomic constraints is at its core a device for handling underactuated system, often where oscillatory motion is desired. In the simplest case there is a single degree of underactuation. After applying constraints to such a system its behavior will be uniquely defined by it's initial condition, and a single autonomous differential equation will suffice to describe the entire evolution of the system. Virtual holonomic constraints are defined as functions of a common scalar variable, by convention named θ. This variable serves as a measure of progress throughout the motion trajectory, and it will for certain purposes be desired that θ is a monotonic function of time. In this case, an obvious and ever present option of θ is the distance traveled along the orbit of the motion defined by

$$\mathcal{O}(q_\star) = \{[q, \dot{q}] \in \mathbb{R}^n \times \mathbb{R}^n : q = q_\star(t), \dot{q} = \dot{q}_\star(t), t \in [0, T_h]\}\,, \tag{9}$$

but for many systems simpler choices are available. The constraints themselves are then formulated as smooth functions

$$\theta = \theta_\star(t), \quad q_1 = \varphi_1(\theta), \quad q_2 = \varphi_2(\theta) \ \ldots \ q_n = \varphi_n(\theta)\,, \tag{10}$$

with derivatives

$$\dot{\theta} = \dot{\theta}_\star(t), \quad \dot{q}_1 = \varphi_1'(\theta)\dot{\theta}, \quad \dot{q}_2 = \varphi_2'(\theta)\dot{\theta} \ \ldots \ \dot{q}_n = \varphi_n'(\theta)\dot{\theta}\,. \tag{11}$$

As stated, there is one constraint for each actuated degree of freedom. After substituting the constraints and their derivatives into the differential equations, a system with one degree of underactuation can be written as

$$\alpha_1(\theta)\ddot{\theta} + \beta_1(\theta)\dot{\theta}^2 + \gamma_1(\theta) = 0 \tag{12}$$

$$\alpha_i(\theta)\ddot{\theta} + \beta_i(\theta)\dot{\theta}^2 + \gamma_i(\theta) = \tau_i, \quad i = 2 \ldots n\,. \tag{13}$$

The first equation (12), often labeled the α-β-γ-equation , or the *reduced dynamics*, will, when integrated, describe the full evolution of the scalar quantities θ, $\dot{\theta}$ and $\ddot{\theta}$, assuming that all constraints are held *invariant*. That is, the equation is valid as long as there exist a controller which makes the system respect the constraints applied. During design the constraints are assumed to be held perfectly invariant, but for the actual dynamic system this is obviously not the case. The autonomous α-β-γ-equation (12) is used for planning motion, and by assuming the constraints to be invariant, a desired motion trajectory can be generated.

By substituting the motion, $[\theta,\, \dot{\theta},\, \ddot{\theta}]^T$ found during integration of the reduced dynamics (12), into the actuated equations (13), the nominal force vector τ required to achieve the motion is obtained. This is a highly useful property, as it allows force and torque to be taken into consideration when designing motion, ensuring that the required forces are, at the very least, feasible. Extrapolating, this property might also simplify the search for force-optimal trajectories.

During a phase where constraints are to be generated it can for some

purposes be sensible to insert constraints on the form

$$\theta = \theta_\star(t), \quad q_1 = \varphi_1(P,\theta), \quad q_2 = \varphi_2(P,\theta) \ \ldots \ q_n = \varphi_n(P,\theta)\,. \tag{14}$$

This means that the constraints can be swapped effortlessly, and that numerical methods can easily be implemented to take advantage of the reduced dynamics. The α-β-γ-equation is then written as

$$\alpha_1(\theta,P)\ddot{\theta} + \beta_1(\theta,P)\dot{\theta}^2 + \gamma_1(\theta,P) = 0 \tag{15}$$

where P is a vector of parameters like, for instance, coefficients of polynomial constraints of a given order. Such a rewrite is of high practical value.

As stated, the fact that virtual holonomic constraints deal with purely geometrical constraints applied to underactuated systems means that the method lends itself very well to generating oscillating motion. Oscillatory motion will be shown to play a central role in generating gaits for Semi-Quad.

1.2.1 Properties of the reduced dynamics

The behavior of the virtually constrained system (12) is very transparent compared to the original system. Below, a few key properties central to the later controller synthesis are restated.

Lemma 1 *[12, Lemma 1] Along any solution* $[q(t),\ \dot{q}(t)]$ *of system the following identity holds*

$$\frac{d^2}{dt^2}q(t) = \frac{d}{dq}\left(\frac{1}{2}\dot{q}^2(t)\right) \tag{16}$$

Proof *Applying the chain-rule to* $\frac{d^2}{dt^2}q(t)$ *leads to the line of equalities*

$$\begin{aligned}\frac{d^2}{dt^2}q(t) = \frac{d}{dt}(\dot{q}(t)) = \frac{d}{dq}(\dot{q}(t))\frac{d}{dt}(q(t)) \\ = \dot{q}(t)\frac{d}{dq}(\dot{q}(t)) = \frac{d}{dq}\left(\frac{1}{2}\dot{q}^2(t)\right)\end{aligned} \tag{17}$$

where the equality of the first and last expression gives the identity (16). ∎

The identity given in lemma 1 allows us to redefine our system, from a second order differential equation like (12) to a first order differential equation on the form

$$\frac{1}{2}\alpha(\theta)\frac{d}{d\theta}(\dot{\theta}^2) + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0. \tag{18}$$

A system such as the above is integrable for any choice of $\alpha(\theta)$, $\beta(\theta)$ and $\gamma(\theta)$ assuming that $\alpha(\theta) \neq 0$ for the relevant interval of θ. Lemma 1 is necessary for the next result, theorem 1.

The next result gives us a general integral of motion[2] for the system.

Theorem 1 *[12, Theorem 1] Along any solution $\theta(t)$ of the nonlinear system*

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = 0 \tag{19}$$

the integral function

$$\begin{aligned} I(\theta, \dot{\theta}, \theta(0), \dot{\theta}(0)) = \dot{\theta}^2 - e^{-\int_{\theta(0)}^{\theta} \frac{2\beta(\tau)}{\alpha(\tau)}\,\mathrm{d}\tau}\dot{\theta}^2(0) \\ + \int_{\theta(0)}^{\theta} e^{\int_{s}^{\theta} \frac{2\beta(\tau)}{\alpha(\tau)}\,\mathrm{d}\tau}\frac{2\gamma(s)}{\alpha(s)}\,\mathrm{d}s \end{aligned} \tag{20}$$

preserves its zero value $I(\theta, \dot{\theta}, \theta(0), \dot{\theta}(0)) \equiv 0$ for all $t \geq 0$ for which the solution $\theta(t)$ is defined.

Proof *By using the identity* (16) *found in lemma 1 and performing the substitution $\dot{\theta}^2(t) = Y(q(t))$, we're left with a linear, first-order differential equation on the form*

$$\frac{d}{d\theta}Y + a(\theta)Y = b(\theta) \tag{21}$$

where $a(q) = \frac{2\beta(\theta)}{\alpha(\theta)}$ and $b(q) = -\frac{2\gamma(\theta)}{\alpha(\theta)}$. First order differential equations like

[2] A general integral of motion of a dynamical system is a function of the systems state and initial conditions which remains constant along the solutions of the system[12].

this have a general solution on the form

$$Y = e^{\int_{\theta(0)}^{\theta} a(\tau)\,\mathrm{d}\tau}\left[Y(0) + \int_{\theta(0)}^{\theta} e^{-\int_{\theta(0)}^{s} a(\tau)\,\mathrm{d}\tau} b(s)\,\mathrm{d}s\right]. \tag{22}$$

By moving the right-hand side of (22) *over to the left side, and by undoing the previous substitutions we have proven* (20). ∎

Remark *The source [12] presents a more general result, for Bernoulli equations, but the proof is essentially the same.*

Theorem 1 provides a measure of to what degree the system is following its predefined trajectory [12] and therefore also to what degree the constraints are being respected. Failure of keeping the constraints invariant will result in the general integral of the system deviating from zero, and it will be shown to constitute an essential part of the controller design.

2 SemiQuad

At the core of this thesis is a robot named SemiQuad. SemiQuad has been designed and constructed at École de Nantes in France. While the name might imply quadrupedal motion, the robot only has two legs. However, as the motion is limited to the sagittal plane[3], the robot is designed for motions resembling that of a quadruped. Specifically, SemiQuad walks using what is know as a *curvet gait*. SemiQuad is a seven DOF robot with four joint variables q_1, q_2, q_3, q_4, and three variables α, x, y defining the absolute orientation of the robot relative to the environment. α is the absolute angle of the torso, relative to the horizontal axis, and x, y represents the absolute position of the torso center.

2.1 Assumptions related to the gait of SemiQuad

SemiQuad is designed for a particular type of gait defined by the following points:

There is no flight phase during the gait cycle. The robot is always in contact with the ground surface, and the motion therefore consists of two distinct phase types: an underactuated single support phase and an overactuated double support phase.

During the double support phase both feet are in contact with the surface. The robot is at this point overactuated, and may freely alter its configuration as long as the contact between the legs and the surface is respected. That is, as long as the robot is in double support phase, the distance between the legs remains constant. As virtual holonomic constraints relate to underactuated systems, this part of the motion will not be modeled.

[3]Sagittal plane: For bilaterally symmetric objects, the sagittal plane is an imaginary, two-dimensional plane which separates the left part of the body from the right. Alternatively, the sagittal plane can be said to be the plane on which the silhouette of the robot will fall when viewed from the side.

The single support phase is characterized by the robot only having one leg in contact with the surface. The robot is underactuated due to the lack of actuation of the first "joint", between the ground and the link in contact with the ground. The leg in contact with the floor, the *stance leg*, is assumed to neither slide nor lift. The other leg, the *swing leg*, moves freely. This is the phase for which motion will be generated. **Due to not modeling the double support phase, the single support phase of gaits are assumed to begin with zero initial velocity**.

Impacts are, due to the impact model chosen, assumed to instantly damp the motion of SemiQuad upon impact, resulting in no bounce or slipping. Post impact the robot enters double support phase, and any effect the impact could have had on the robot configuration can be counteracted during this phase.

The gait cycle step is considered to begin in double support phase. Then the front leg is elevated, and the robot enters a single support phase. While in single support phase, the front leg is extended forward some distance before it again touches the ground. At this point the robot again enters double support phase, and may freely alter its configuration. The robot then completes the gait cycle by raising its hind leg and dragging it forward, entering a configuration where the distance between the two legs is equal to that of the initial configuration.

Forward gait direction is considered to be towards the right in the sagittal plane.

2.2 The original model

The dynamic equations of the system have been found through the use of the Euler-Lagrange Formalism, and have a structure similar to (4). $M(\mathbf{q})$ and $C(\mathbf{q}, \dot{\mathbf{q}})$ were 7×7 matrices, $G(\mathbf{q})$ was a 7×1 matrix and the vector of generalized force τ had one entry for each of the four actuated joints.

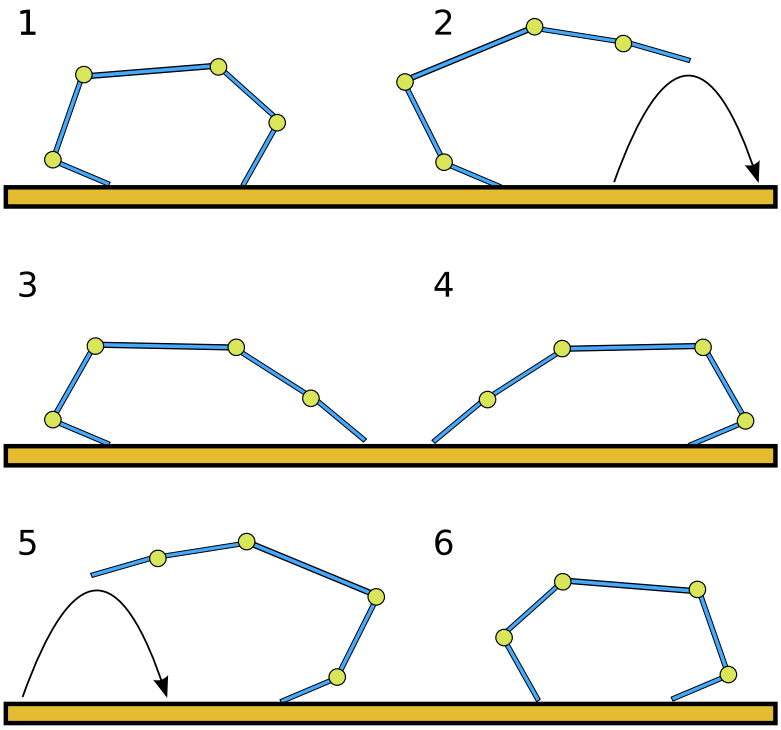


Figure 1: The entire *curvet gait* cycle. First step is an extension of the front leg. Second step is a retraction of the hind leg.

To account for the feet being in contact with the ground, the dynamics were augmented as shown in section 1.1.4, leading to the model being formulated as

$$\begin{bmatrix} M(\mathbf{q}) & A_1^T & A_1^T \\ A_1 & 0_{2\times 2} & 0_{2\times 2} \\ A_2 & 0_{2\times 2} & 0_{2\times 2} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = - \begin{bmatrix} C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) - \tau \\ A_1'\dot{\mathbf{q}}^2 \\ A_2'\dot{\mathbf{q}}^2 \end{bmatrix}. \tag{23}$$

During single support phase, the constaint corresponding to the swing leg was switched off, making the dynamics usable both during single and double support phases. Switching a constraint off meant reformulating the

dynamics as

$$\begin{bmatrix} M(\mathbf{q}) & A_1^T & A_1^T \\ A_1 & 0_{2\times 2} & 0_{2\times 2} \\ 0_{2\times 7} & 0_{2\times 2} & \mathrm{I}_{2\times 2} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = - \begin{bmatrix} C(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + G(\mathbf{q}) - \tau \\ A_1'\dot{\mathbf{q}}^2 \\ A_2'\dot{\mathbf{q}}^2 \end{bmatrix}, \qquad (24)$$

for the case where the leg corresponding to the second constraint was free. The dynamics would even be valid for flight phases. However, for applying virtual holonomic constraints we wanted a model of a more straightforward structure.

2.3 The simplified model

This original formulation was very flexible; however, for our purposes – generating motion for the single support phase only – a simpler model with fewer degrees of freedom was desired. According to the assumptions listed in section 2.1, the stance leg neither leaves the ground nor slides along the surface. Assuming this to be true, this simplification meant that the five-link robot could be represented by five degree of freedom dynamics, with a single unactuated joint. Note that when designing steps later on, confirming that the ratio between the vertical and horizontal force between the stance leg and the ground is within the friction cone is prudent.

2.3.1 The single support phase

During the single support phase the robot can be modeled as a five-link robot with revolving joints. The stance leg is connected to the ground via an unactuated revolving joint. The generalized coordinates are chosen in such a way that nominal joint values will be on the interval $q_i \in (0, \pi)$ during the first period of the gait.

To develop this model, all joint angles had to be defined in absolute coordinates. For complex systems in three dimensional space it would be

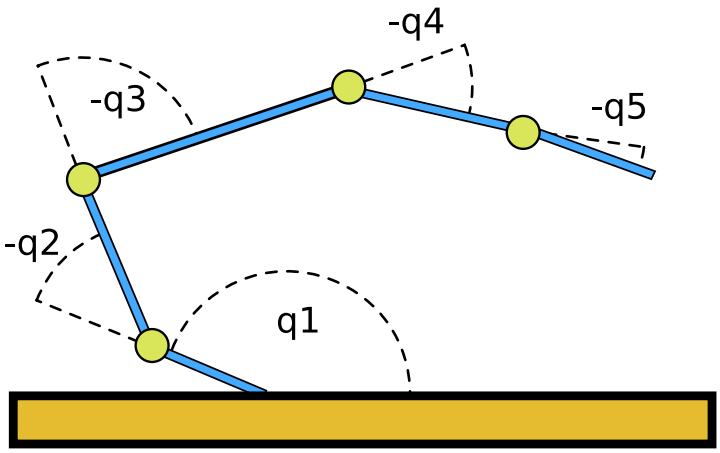


Figure 2: The new five-link SemiQuad model. The model represents the robot dynamics during single support phase, where motion is restricted to the sagittal plane.

natural to introduce Denavit-Hartenberg Parameters [15] and homogeneous transformation matrices at this point. However, as this system is two-dimensional and of a fairly straightforward structure, a much simpler approach did equally well. Based on the choice of joint variables the absolute coordinates of the system were defined by

$$\begin{aligned} a_1 &= q_1, \\ a_2 &= a_1 - q_2, \\ a_3 &= a_2 - q_3, \\ a_4 &= a_3 - q_4, \\ a_5 &= a_4 - q_5. \end{aligned} \tag{25}$$

From this, the position of mass centers for all links was computed to be

$$
\begin{aligned}
p_1 &= [(l_1 - s_1)\cos(a_1), (l_1 - s_1)\sin(a_1)] \\
p_2 &= [l_1\cos(a_1) + (l_2 - s_2)\cos(a_2), \\
&\quad\ l_1\sin(a_1) + (l_2 - s_2)\sin(a_2)] \\
p_3 &= [l_1\cos(a_1) + l_2\cos(a_2) + s_3\cos(a_3), \\
&\quad\ l_1\sin(a_1) + l_2\sin(a_2) + s_3\sin(a_3)] \\
p_4 &= [l_1\cos(a_1) + l_2\cos(a_2) + l_3\cos(a_3) + s_4\cos(a_4), \\
&\quad\ l_1\sin(a_1) + l_2\sin(a_2) + l_3\sin(a_3) + s_4\sin(a_4)] \\
p_5 &= [l_1\cos(a_1) + l_2\cos(a_2) + l_3\cos(a_3) + l_4\cos(a_4) + s_5\cos(a_5), \\
&\quad\ l_1\sin(a_1) + l_2\sin(a_2) + l_3\sin(a_3) + l_4\sin(a_4) + s_5\sin(a_5)]
\end{aligned} \tag{26}
$$

where l_i denotes the lengths of the individual links and s_i denotes the distance between a link's first joint and its center of mass. From these vectors, the kinetic energy was calculated as

$$
\mathcal{K}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\sum_{i=1}^{5}\left(m_i\dot{p}_i \cdot \dot{p}_i\right) + \frac{1}{2}\sum_{i=1}^{5}\left(\mathrm{Iy}_i\dot{a}_i \cdot \dot{a}_i\right) + \frac{1}{2}\sum_{i=2}^{5}\left(\mathrm{Ia}_{i-1}\dot{q}_i^2 N\right) \tag{27}
$$

where m_i is the link mass, Iy is the link inertia, and where Ia and N is the motor inertia and the gearing ratio, respectively. Finally, the potential energy was calculated as

$$
\mathcal{V}(\mathbf{q}) = \sum_{i=1}^{5}\left(m_i[0,\ g] \cdot p_i\right). \tag{28}
$$

From the kinetic and potential energies the Lagrangian was defined, and the model computed according to the method listed in section 1.1.3. For the complete Maple code used, see appendix A. The single support phase dynamics could then be defined by equations on the form (4).

2.3.2 The double support phase

As an exercise, the simplified dynamics were initially augmented with contact constraints for the front leg to cope with the double support phase. However, as the system is overactuated during double support phase, virtual holonomic constraints are not really applicable. Adding the contact constraint to the autonomous constrained system is almost equivalent to constraining a one-link robot – effectively hindering the motion altogether. Of course, there does exist virtual holonomic constraints which will allow motion in spite of the front leg being constrained, for instance, moving the robot in a parallelogram-fashion obviously works. However, for more useful motions adhering to this extra constraint adds severe complexity. For this reason the double support phase was omitted, meaning that the generated motion should be assumed to begin the moment either one of the legs left the floor, in contrast to the alternative where the double support phase kick-off is included as part of the motion. In other words, the motion generated later will begin with zero velocity.

2.3.3 Transformation of the generalized coordinates

There is a simple, algebraic transformation (29) between the old generalized coordinates and the new. This allows us to completely map the relevant single support phase motion generated for the old 7-DOF model to the new 5-DOF model.

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix} = \begin{bmatrix} \alpha + \hat{q}_1 + \hat{q}_2 \\ \hat{q}_1 - \pi \\ \hat{q}_2 - \pi \\ \hat{q}_3 - \pi \\ \hat{q}_4 - \pi \end{bmatrix} \tag{29}$$

The vector $[\hat{\mathbf{q}}, \alpha]$ consists of the old coordinates and the vector $\mathbf{q}$ corresponds to the new choice of coordinates.

3 A Method for Reparameterizing an Existing Motion in Terms of Virtual Holonomic Constraints

In this section, an attempt was made to closely imitate the original motion of SemiQuad as designed by the researchers at École de Nantes. The method used is generic and should be capable of reparameterizing arbitrary motions. The motion to be parameterized consisted of a single gait cycle, and from this cycle only the underactuated single support phases were relevant in the context of virtual holonomic constraints. Reparameterizing the motion of SemiQuad was a two step procedure. First, a monotonically increasing function $\theta = F(\mathbf{q})$ had to be found. Second, some sort of smooth mapping $\varphi_i(\theta)$ between each of the joint variables q_i and the θ found in the first step had to be found. This mapping should be accurate. Finally, as the constraints was substituted back into the dynamics, the behavior of the reduced dynamics should be as close to that of the original motion as possible.

3.1 The original motion

The original motion was generated by a series of polynomial references. At any time each of the actuated joints was guided by one such reference, and several sets of references then lead the robot through various phases of the gait cycle. PD-controllers were used to keep the joints variables on reference. The motion, in vectors of position, velocity and acceleration, was recorded during the gait cycle, and the intervals of single support were extracted – from the instant either leg cleared the ground to the moment it touched back down. The samples were given in terms of the original seven degree of freedom system, and had to be converted to the new five degree of freedom system using the mapping given previously (29).

3.2 Finding a measure of progress

As noted above, $\theta = F(\mathbf{q})$ was required to be monotonically increasing. When generating motion from scratch, this is not necessarily the case; however, for the purpose of imitating an existing motion, it is required. Why this is the case will be explained shortly. For some systems, there might be an obvious choice of θ. It might, for instance, be the distance traveled along a trajectory or simply the monotonic evolution of an angle of a joint during the motion.

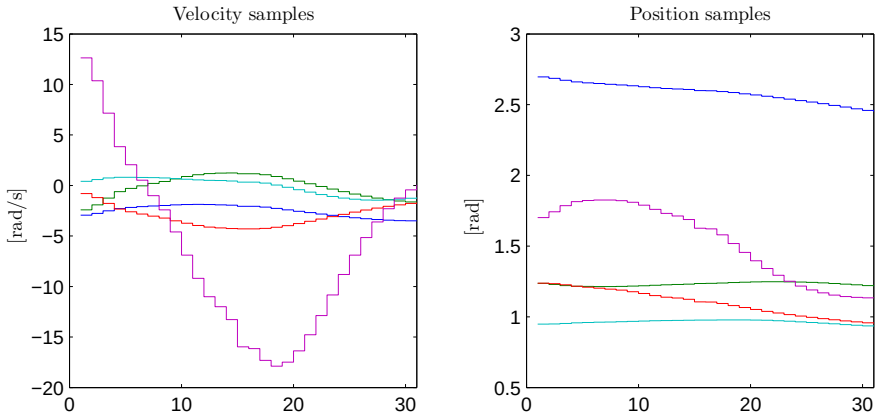


Figure 3: Discrete samples of the original motion during an interval of the single support phase.

For the motion recorded earlier, none of the states $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5]$ were monotonically increasing during the relevant interval of time, and therefore neither of them could be used alone. Therefore, a generic method of generating θ was used for this system. It is clear that while neither of the states are monotonically increasing it should be possible to formulate some sort of function $F(\mathbf{q})$ of the states that would be. Finding a monotonically increasing $F(\mathbf{q})$ is a matter of finding a sign-definite $F'(\mathbf{q})\dot{\mathbf{q}}$.

I will illustrate the method used: First, a function $\theta = F(\mathbf{q}, \alpha)$ was defined,

in this case a function on the form

$$F(\mathbf{q},\alpha)=\sum_{i=1}^{5}\left(\alpha_{i,1}q_i+\alpha_{i,2}\cos(q_i)+\alpha_{i,3}\cos(\frac{q_i}{2})+\alpha_{i,4}\cos(\frac{q_i}{4})\right) \tag{30}$$

where α is a set of unresolved parameters. Simpler functions with fewer terms were tried initially, but it seemed like a certain amount of independent terms had to be added to $F(\mathbf{q},\alpha)$ for the procedure to work. The idea then is simple: First, sample the original state trajectory at fixed intervals as shown in figure 3. The sampling needs to be dense enough to not lose any important characteristics of the original data, and sparse enough to allow for reasonably quick computation.

Then, an object function was generated by differentiating $F(q,\alpha)$ with respect to time

$$\begin{aligned}O(\mathbf{q},\dot{\mathbf{q}},\alpha)&=\frac{d}{d\mathbf{q}}\left(F(\mathbf{q},\alpha)\right)\dot{\mathbf{q}}\\&=\sum_{i=1}^{5}\left(\alpha_{i,1}\dot{q}_i-\alpha_{i,2}\sin(q_i)\dot{q}_i-\alpha_{i,3}\sin(\frac{q_i}{2})\frac{\dot{q}_i}{2}-\alpha_{i,4}\sin(\frac{q_i}{4})\frac{\dot{q}_i}{4}\right)\end{aligned} \tag{31}$$

The trick is then to find a set of $\alpha_{i,j}$ which makes (31) sign-definite for each of the points sampled from the original state trajectory. That is, for all samples $O(\mathbf{q}_i,\dot{\mathbf{q}}_i,\alpha)$ for $i=1,\ldots m$ should be either strictly positive, or strictly negative. In addition, the absolute value of this function should ideally be as large as possible at each sample, as the resulting mapping of θ to $\mathbf{q}$ then will be of higher resolution.

To clarify the issue of monotonicity and resolution: Should (31) become zero at any point during the motion, $\theta=F(q,\alpha)$ will become constant for the corresponding interval. This means that the subsequent mapping of θ and $\mathbf{q}$, dealt with in the next section, will not be unique, as multiple consecutive values of $\mathbf{q}$ will share a common θ. Conversely, a θ spanning over a large range will provide a better mapping, less sensitive to noise and

numerical degradation. It is clear from the structure of (31) that this occurs whenever $|\,\dot{\mathbf{q}}\,| = 0$. This happens at several points during the single support phases of the original motion. The problem was solved by separating the motion into separate intervals at these points, ending up with intervals of motion without zero-crossings.

To find the coefficients α, a Matlab routine was implemented which calculated the object function (31) for the vectors of motion previously recorded. The routine returned the minimum value of the resulting vector, as a minimum value larger than zero would mean that the function was sign-definite. The routine was optimized[4] with respect to the vector α, returning the coefficients which maximized the minimum value of (31).

By plugging the optimal coefficients back into (30) a monotonically increasing measure of progress for the motion had been found as a function of the states alone. One such measure can be seen in figure 4 where θ is displayed along with the corresponding sign-definite $\dot{\theta}$.

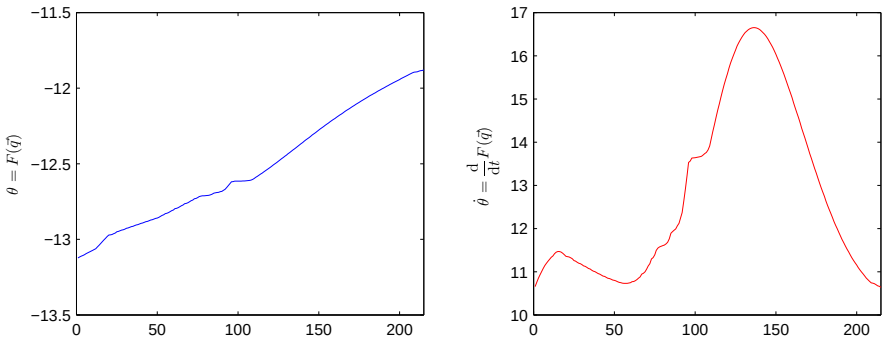


Figure 4: A monotonic $\theta = F(\mathbf{q})$ found for some interval of the step motion.

3.3 Finding constraints

Having obtained a monotonically increasing θ, creating the mapping of $q_i = \varphi_i(\theta)$ was a matter of making an approximation of the states q versus

[4]Optimizing routine: A genetic algorithm [1] was employed for the optimization, due to irregularity of the object function $\frac{d}{d\mathbf{q}_n}\left(F(\mathbf{q}_n, \alpha)\right)\dot{\mathbf{q}}_n$.

$\theta = F(\mathbf{q})$. To ensure that the fit was good, the first and second derivative of $\varphi_i(\theta)$ was included in the approximation as well. The constraints $\varphi(\theta)$ were chosen to be polynomials of fixed order, and the derivatives of these functions were thus easily computed as

$$\begin{aligned} q_i =& \varphi_i(\theta) = b_n\theta^n + b_{n-1}\theta^{n-1} + \ldots + b_0 \\ \dot{q}_i =& \varphi_i^{'}(\theta)\dot{\theta} = nb_n\theta^{n-1}\dot{\theta} + (n-1)b_{n-1}\theta^{n-2} + \ldots + b_1\dot{\theta} \\ \ddot{q}_i =& \varphi_i^{'}(\theta)\ddot{\theta} + \varphi_i^{''}(\theta)\dot{\theta}^2 \,. \end{aligned} \tag{32}$$

The fitting was formulated as a least-squares-problem on the form

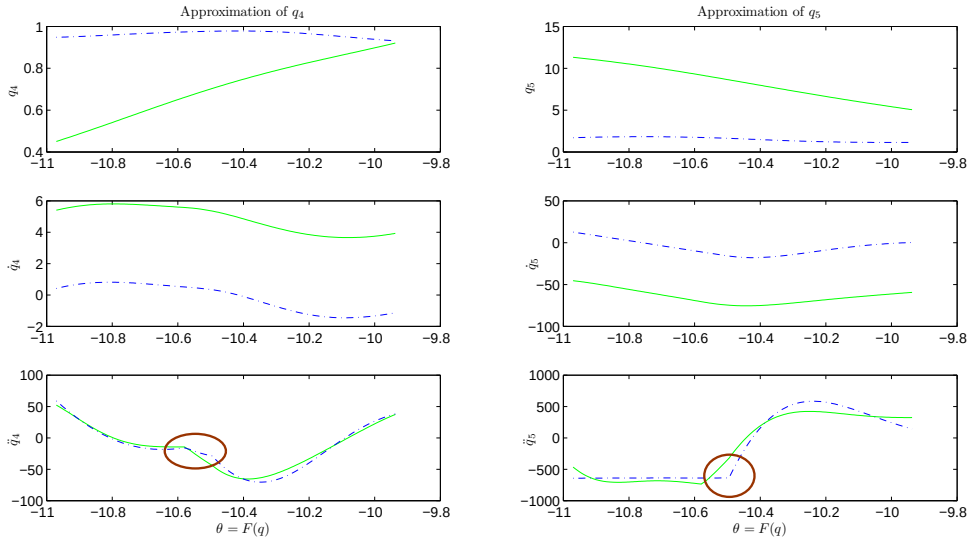
$$Ax = q \tag{33}$$

where A was populated by the variables of θ_i, $\dot{\theta}_i$, $\ddot{\theta}_i$ of the constraints (32), while x was a vector of the corresponding coefficents $b_0, \ldots, b_n$. The polynomial variables θ_i, $\dot{\theta}_i$ and $\ddot{\theta}_i$ were easily calculated by differentiating the function $\theta_i = F(\mathbf{q_i})$ found in the previous section. Finally, the vector q consisted of samples from the original motion of the joint variable for which the approximation was done.

$$\begin{bmatrix} 1 & \theta_1 & \theta_1^2 & \ldots & \theta_1^n \\ \vdots & & \vdots & & \vdots \\ 1 & \theta_m & \theta_m^2 & & \theta_m^n \\ 0 & \dot{\theta}_1 & 2\theta_1\dot{\theta}_1 & \ldots & n\theta_1^{n-1}\dot{\theta}_1 \\ \vdots & & \vdots & & \vdots \\ 0 & \dot{\theta}_m & 2\theta_m\dot{\theta}_m & & n\theta_m^{n-1}\dot{\theta}_m \\ 0 & \ddot{\theta}_1 & 2\theta_1\ddot{\theta}_1 + 2\dot{\theta}_1^2 & & n\theta_1^{n-1}\ddot{\theta}_1 + (n-1)n\theta_1^{n-2}\dot{\theta}_1^2 \\ \vdots & & \vdots & & \vdots \\ 0 & \ddot{\theta}_m & 2\theta_m\ddot{\theta}_m + 2\dot{\theta}_m^2 & \ldots & n\theta_m^{n-1}\ddot{\theta}_m + (n-1)n\theta_m^{n-2}\dot{\theta}_m^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_m \\ \dot{q}_1 \\ \vdots \\ \dot{q}_m \\ \ddot{q}_1 \\ \vdots \\ \ddot{q}_m \end{bmatrix} \tag{34}$$

The least squares fitting outlined above was run five times, once for each of

the joint variables, leading to what should have been a very good approximation of the original motion. From the plots in figure 5 it is clear that the polynomial fittings were not sufficiently accurate, and this is assumed to be due to a lack of smoothness of the original motion. As can be seen from the stippled blue line in figure 5 the accelerations of the original motion had certain points of very abrupt changes, and the procedure outlined in this section relies on the motion to be imitated having some level of smoothness.



(a) Approx. of $q_1(\theta)$ (b) Approx. of $q_2(\theta)$

Figure 5: Plots of 15th order polynomial approximation for joint variables q_4 and q_5 during a interval of the single support phase. Each subplot consists of three curves of, from top to bottom, position, velocity and acceleration. The blue, stippled line is the original motion, while the green solid line is the approximation. Note the jump of acceleration circled.

3.4 Discussion and remarks

As indicated there were pitfalls with imitating the original motion. Drawing conclusions from this section the following issues should be pointed out:

1. The original trajectory had several points at which all joint velocities reached zero more or less simultaneously. This lead to θ becoming constant and thus to poor mapping of constraints $\varphi_i(\theta)$. This problem was partially solved by splitting the trajectory into several sub-intervals, each without velocity zero-crossings. Still, the fact that all velocities converged towards zero at the ends of these sub-intervals meant that $F(q, \alpha)$ ended up with less than ideal performance.

2. The biggest obstacle with the reparameterizing approach, and the factor which ultimately decided against continuing trying to imitate the original motion, arose due to what was perceived to be unfortunate properties of the original motion. As noted earlier, the original motion trajectory parameterization consisted of several sets of polynomials. It looked as if the changing of one set of reference trajectories to another created instant jumps in reference. This caused very abrupt fluctuations in torque and acceleration. Note the jumps of acceleration in figure 5.

   The fact that the original motion made discrete jumps in acceleration made it very difficult to approximate the motion with smooth constraints. As the polynomial approximation took not only $\mathbf{q}$ and $\dot{\mathbf{q}}$ into consideration, but $\ddot{\mathbf{q}}$ as well, the resulting constraints were inaccurate. This problem resurfaced when the constraints were substituted into the α-β-γ-equation (12) later on, which then exhibited a behavior very unlike the original motion. A possible solution might have been to not include the accelerations in the approximation and rather create a new, smooth acceleration profile based on the position and velocity data alone.

The method used to parameterize the original motion using virtual holonomic constraints here is very generic, and initially seemed to be very promising. A numerical optimization method was used to find a suitable measure of progress, and a least-squares-approximation made to generate the constraints. The positive thing about this procedure is that it's completely generic. The negative thing is that the generated θ and $\varphi_i(\theta)$

will tend to be large and numerically cumbersome, and that undesirable properties of the original motion will at best be reflected in the results, and may in worst case distort the results altogether.

4 Generating New Motion via Virtual Holonomic Constraints

While imitating motion via the use of virtual constraints and the method outlined above is certainly possible, generating motion from scratch offers much more freedom and flexibility, and where constraints created for imitated motion will tend to be complicated, constraints generated from scratch will often be both elegant and simple. Additionally, being liberated from having to stay faithful to the original motion, entirely new types of steps could be created. A variety of steps were tried and tested, however, all steps found could be divided into two general categories: leap steps and balanced steps.

During a leap step, there will be a point during the step after which the robot will be unable to abort the motion. The robot will have to commit to the step. The impact will typically be rough, as during the last phase of the step the center of mass of the robot will often be more or less free-falling towards the ground. It is quite easy to optimize these steps for walking up slopes and stairs, and clever choices of optimization criteria can limit the impact velocity as well, reducing the force of the impact. A leap step could be very long, and might lead to a post impact robot configuration where the robot will be unable to move its center of gravity outside the statically stable area. This means that the robot will not be able to complete the gait with a follow-up step starting with zero velocity – one of our assumptions when generating these virtual constraints (see sec 2.3.2). For this reason, parameterizing a complete gait in this manner wasn't done, only front leg steps were created.

A balanced step, on the other hand, is characterized by the property that that at any point during the motion, even after the tip of the leg has impacted with the floor, the step can be aborted. This means that all the constraints of which the step is made are periodic motions by themselves. This allows for the creation of steps with far greater control and of poten-

tially zero impact velocity; however, the balanced step is more restricted than the leap step, and, in my experience, more difficult to generate.

The main restriction with balanced steps arises from having to move the center of mass outside the polygon spanned by its legs to create torque around the stance leg. This then means that the possible distance between the legs is limited.

While a large variety of leap steps were constructed, for a variety of optimization criteria, the search for a gait focused on balanced steps. As constructing a balanced step is searching for constraints for which the reduced dynamics have stable periodic motion, it is particularly relevant in the context of virtual holonomic constraints.

There is also a choice of how many sets of constraints one wishes to use when constructing the motion. Using more sets of constraints might allow the use of simpler constraints, and vice versa. The motion generated within this section was designed to use two constraints per step. This means that at a point during each of the steps, the sets of constraints are swapped. These points are called “switching surfaces”.

Using two sets of constraints allows us to use simpler constraints. As generating constraints might in some cases be difficult, this was important. Next, using two periodic sets of constraints allowed us to illuminate a special property of periodic motions of the reduced dynamics. This property, which will be elaborated later, in section 4.3, indicates that certain motions will in fact require two or more sets of constraints to achieve the desired motion.

4.1 Creating a measure of progress

As stated in the previous section, generating virtual holonomic constraints for a dynamic system takes place in two steps, and as before a $\theta = F(\mathbf{q})$ has to be found. **However, this function needed not necessarily be monotonically increasing, as is required when imitating motion.** In

fact, certain choices of constraints will yield stable oscillations with correspondingly oscillating θ. This behavior can be of great use when generating motion, and in particular when generating motion such as the balanced steps discussed previously.

There is freedom in the choice of θ. There exists numerous metrics of progress for a robot like SemiQuad, several of which could yield satisfying motion. For instance, choosing the position of the robot's center of mass, or the vertical position of the freely swinging leg could work. However, for this thesis the choice of θ should preferably be one of computational simplicity and of predictable behavior. For the motion generated for SemiQuad in this text, the choice fell on the unactuated joint of the robot's first link. Choosing

$$\theta = q_1$$

will later be shown to allow sensible motion using very simple constraints.

4.2 Creating new constraints

Two sets of constraints per step were to be found, both for the leap steps and balanced steps. The first set to elevate the swing leg from the ground, and the next to extend or retract the swing leg, finishing with an impact to the floor. The constraints were chosen to be linear and cubic functions of θ, and each set provided one constraint for each actuated joint. Analogue to what's shown in the introductory section on virtual holonomic constraints 1.2, constraints on the form

$$\varphi_i(\theta) = P_{i1}\theta^n + P_{i2}\theta^{n-1} + \ldots + P_{in}$$

were inserted into the α-β-γ-equation (12), creating reduced dynamics on the form of (15). By inserting the chosen constraint coefficients and integrating, the evolution of the reduced dynamics was obtained for the corresponding set of constraints. By then substituting θ into the constraints and

solving for the joint variables, the full motion characteristics of SemiQuad could be visually observed and qualitatively analyzed.

When finding constraints for this type of systems, there are a few properties of the reduced dynamics which can be of great use. First of all, for a set of constraints an equilibrium is found whenever

$$\gamma(\theta_0) = 0\,.$$

In short, this means that the center of mass of the robot at this point generates no torque around the stance leg. This property easily proved by inserting $\theta, \dot{\theta} = 0$ into the α-β-γ-equation (12). It also means that if $\gamma(\theta)$ becomes 0 at some point during the motion, the motion might exhibit useful, periodic behavior. Whether or not the oscillation is stable will obviously depend on how the center of mass of the robot moves on either side of the equilibrium. The stability of the equilibrium can be determined by

$$\frac{d}{d\theta}\frac{\gamma(\theta)}{\alpha(\theta)}\bigg|_{\theta=\theta_0} > 0, \tag{35}$$

where θ_0 is equilibrium in question [3]. If (35) is positive, then the motion of the reduced dynamics is captured in a vector field loop, and will perpetuate this motion until the vector field is altered.

4.2.1 Initial conditions

The initial configuration of the robot was chosen in such a way that the center of mass generated counter-clockwise torque around the stance leg from the very start. A stable oscillation of the first set of constraints C_1 was created by moving the center of mass forward, halting the counter-clockwise motion as the front leg reached its elevated apex. The reduced dynamics would then retract its motion in reverse direction, completing the cycle. Various versions of this elevating constraint constituted the basis of both balanced and leap steps.

4.2.2 Leap step optimization

First, a framework was created in Matlab to facilitate the optimization of a single leap step. All steps started with the same initial set of constraints, C_1, which simply elevated the front leg. The conditions for swapping the constraints and the parameterization of the following set of constraints were left variable. This way an solution could be found, with respect to a very limited number of parameters. A sanity check was run at the end of each optimization iteration to ensure that the robot configuration was feasible. That is, joints should stay within a certain interval during motion, joint velocities should be within certain limits and the robot should at all times stay above the surface. A number of optimization criteria were tested, and based on a calculated sum of one or more criteria some metric of optimality was attained.

Below is a brief outline of the routine used to optimize the various steps. Optimization criteria tested includes: lowest maximum sum of joint velocities during trajectory; lowest vertical impact speed of swing leg on impact; and highest reach of swing leg during trajectory.

Procedure `Outline of optimization flow`

IN is a vector of parameters generated by the optimizing routine
OUT is a scalar score indicating proximity to optimum

repeat
  $k \longleftarrow$ IN
  $P =$ CreateConstraints(k)
  $t, x =$ IntegrateReducedDynamics(P)
  $Y =$ = CalculateScore(t, x)
  OUT $= Y$.
until *until minimum value is reached*

From the elevated position a step could also be defined by the initial position of the robot, and a desired end configuration. The then unresolved parameters of the trajectory were then available for optimization. By using this method step length was defined explicitly. By using cubic or constraints

of higher order, more parameters were available for optimization, and this way interesting and useful step trajectories could be generated.

4.2.3 Balanced step gait

A periodic gait was sought as a basis for the later development of the transverse linear controller. Balanced steps were chosen for this purpose due to their simple structure. To further simplify the motion design, the constraints were chosen in such a way that the same constraints could be applied for both steps of the gait, although, in reversed order. That is, for the first, front leg step, one constraint C_1 elevated the swing leg. As before, this set of constraints was designed to be stably oscillating, and could perpetuate the motion indefinitely, back and forth, up and down. To advance the motion, a switching surface was defined at the point where the swing leg reached its apex, and at this point the constraints, C_1 and C_2, were swapped. The switching surface could, however, been defined at any point along the C_1 trajectory. The second set of constraints C_2 was stably oscillating as well, however, now extending the swing leg during the declining motion, as seen in figure 6. Together, these constraints could be used for both extending the front leg and retracting the hind leg, by merely reflecting the constraints about the vertical axis.

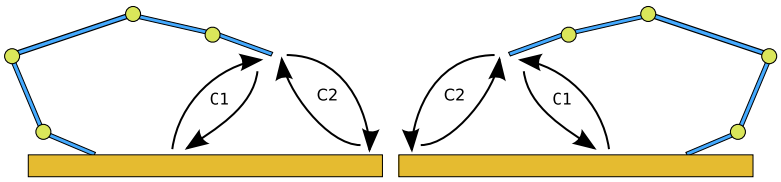


Figure 6: Figure showing how a gait can be designed to take advantage of symmetric properties of the robot. Note that the switching surface where the constraints are swapped could be placed at any point, and not necessarily at $\dot{\theta} = 0$.

Still, finding these constraints was an exercise in gentle tuning. Numerical methods may perform well for designing certain some types of steps as

noted in the previously, but for balanced steps it might be hard to define suitable criteria for optimization. Still, the simplicity of the constraints found made tuning a worthwhile effort. The linear constraints were defined by the initial conditions at the beginning of the motion, and the desired configuration at the point midway between the two extremes. This way, for given initial conditions, the five states of the trajectory midpoint defined the motion alone. The constraints were on the form

$$\varphi_i(\theta) = K(\theta - \theta_p) + \varphi_{p,i} \tag{36}$$

and could be easily computed on the basis of the defined states as shown below. The computed constraints were inserted into the reduced dynam-

---

Procedure `Computing the linear constraint coefficients`

Input: Initial configuration $\theta_0, \varphi_{0,i}$
Input: Midpoint configuration $\theta_p, \varphi_{p,i}$
Output: Coefficients PQ

for $i = 0 \dots 4$ **do**
    $K_i = (\varphi_{0,i} - \varphi_{p,i})/(\theta_0 - \theta_p)$
    $PQ_i = [K_i\,, \quad \varphi_{p,i} - K_i\theta_p]$
end

---

ics, which upon integration would reveal whether the motion was periodic or not. The first constraints, C_1 was defined by the following initial and midpoint configuration

$$\begin{array}{llllll} \theta_0 = 2.895, & \varphi_0 = [& 0.53, & 2.243, & 0.57, & 0.571 \quad] \\ \theta_p = 2.86, & \varphi_p = [& 0.3, & 2.7, & 0.9, & 0.9 \quad] \end{array} \tag{37}$$

and for this set of constraints the reduced dynamics produced a stable periodic motion with period $T = 2.0668$ seconds. As noted above, this set of constraints lifted the leg from the ground. Characteristics of this periodic motion can be seen in figure 7 where it can be seen that $\gamma(\theta)$ is stably oscillating. In the middle plot the positive vertical force means that the stance leg is firmly in contact with the ground at all times. Further, the

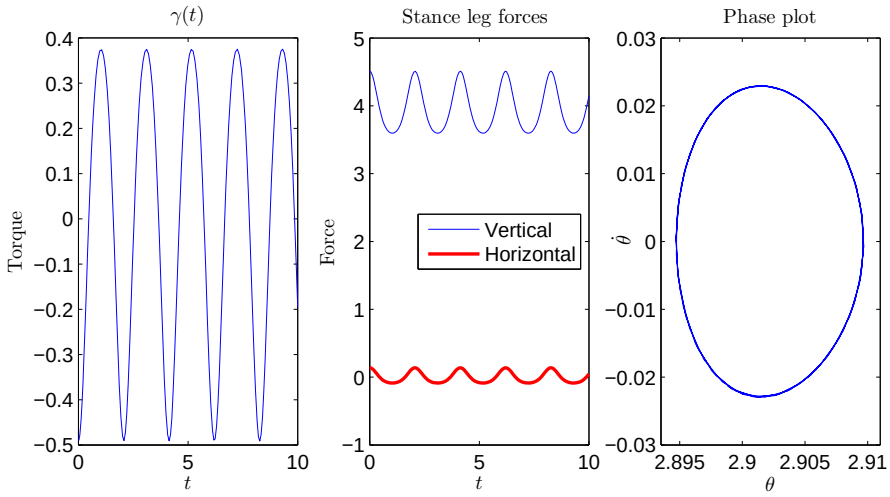


Figure 7: Plot showing characteristics of the periodic motion corresponding to the constraints (37) C_1.

ratio of the vertical and horizontal force is large enough to assume that the stance leg will not slide on most surfaces. Observe in the rightmost phase plot of the reduced dynamics that the θ_p chosen as algebraic midpoint for the constraints will not in general be the actual midpoint of the dynamical system.

Similar characteristics can be seen in figure 8 for the second set of constraints C_2. These constraints were defined by

$$\begin{array}{llll} \theta_0 = 2.79, & \varphi_0 = [& 0.29, \quad 2.65, \quad 0.09, \quad 0.09 &] \\ \theta_p = 2.9, & \varphi_p = [& 0.6, \quad 2.1, \quad 0.4, \quad 0.4 &] \end{array}. \tag{38}$$

From these constraints the chosen gait was formed. It was a slow gait, with a period of 4.78 seconds, not counting the intermediary double support phase. The reduced dynamics of the systems subjected to the two constraints can be seen in figure 9 where each system runs for half a period in a clockwise direction, for a range of initial conditions. The switching surface was defined to be at $\dot{\theta} = 0$. Specifically, the plot in figure 9 shows the motion of the reduced dynamics during a forward step.

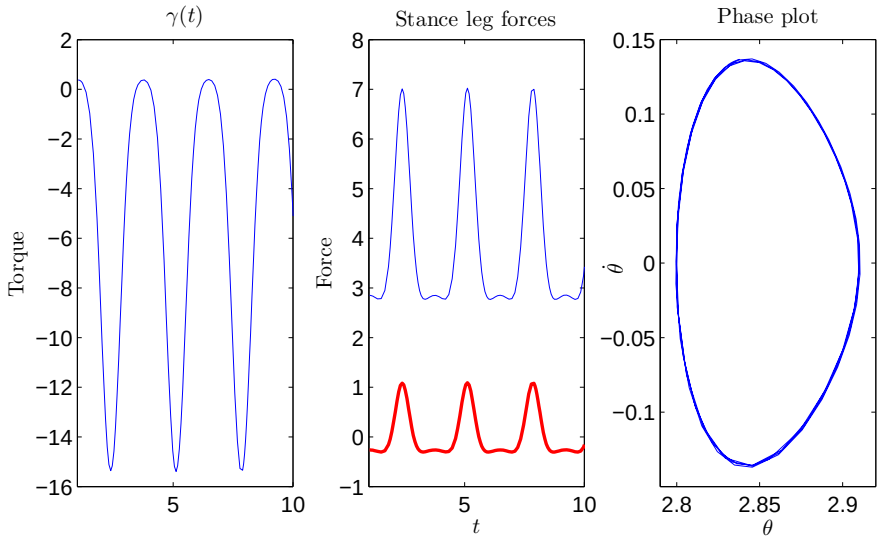


Figure 8: Plot showing characteristics of the periodic motion corresponding to the constraints (38) C_2. As in 7 the thick red line of the middle plot is the horizontal force vector on the stance leg.

The gait is characterized by ending each step with close to zero vertical velocity. Further, when extending the front leg, gravitational force from the robot's center of mass isn't put on the swing leg until after the swing leg is in contact with the surface. Consider the case of a person walking on slippery stones in a river. In that case most people might instinctively perform some form of probing action with the swing leg while putting no real weight upon it, ensuring that they have a firm foothold before transferring weight from the stance leg to the swing leg. The extending leg is then balanced by moving the arms and torso in the other direction. The gait created here follows the same principles.

In figure 10 an animation is shown detailing robot configuration and reduced dynamics phase plots for both steps. By correct controller synthesis, similar motion should be attained for the unconstrained SemiQuad dynamics.

It is clear that just about any type of step might be created using virtual

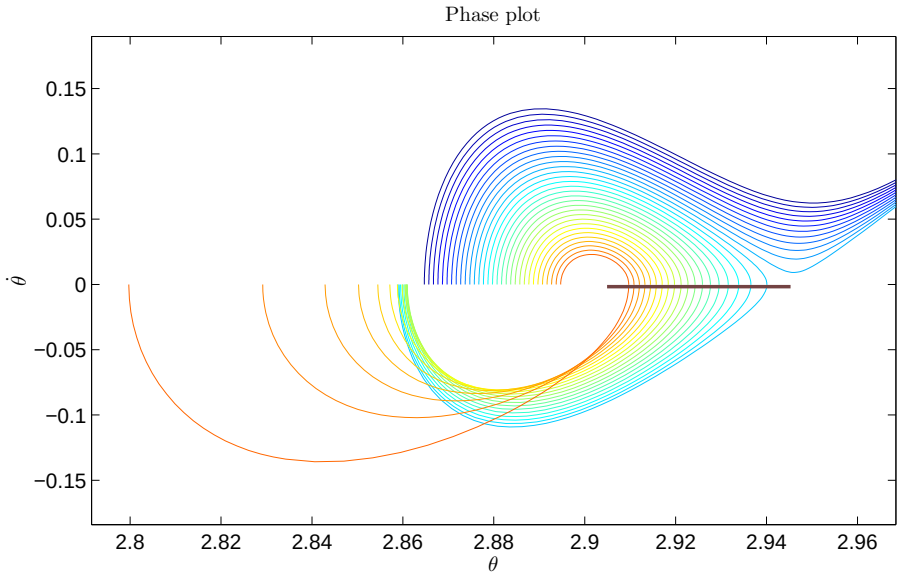


Figure 9: Phaseplots of the constrained system subjected to two sets of linear constraints for a variety of initial conditions. Note that both constraints are periodic with a clockwise motion, and that the constraints are swapped at $\dot{\theta} = 0$, after each of the "constraints" have run for half a period. This switching surface is represented by the thick line.

holonomic constraints, and the constraints could easily be designed to perform perfectly well by any chosen criteria through optimization. The above constraints were chosen for the following reasons: the motion should preferably constitute a periodic gait; the constraints should be simple; and there should be multiple sets of constraints per step.

4.3 Properties of a stable periodic motion

For a given set of constraints it is clear that the vector field of $[\,\theta,\,\dot{\theta}\,]$ created by the reduced dynamics will display properties of symmetry around the θ-axis. Assuming the reduced dynamics start with no initial velocity, they will always follow such a closed loop however large it may be, and any motion of the full constrained dynamical system will therefore in principle

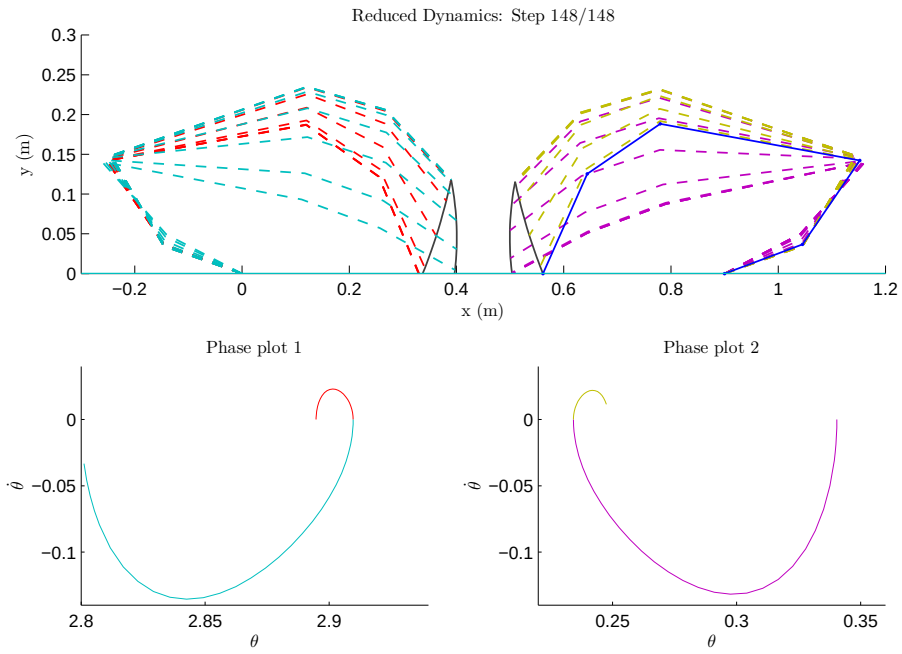


Figure 10: Animation and phaseplots detailing behavior of reduced dynamics when all constraints are held perfectly invariant.

always be a rocking motion, back and forward, following the exact same route both ways.

This leads to the conclusion that to generate certain types of motion, where the resulting dynamical system is desired to move in a perpetual circular fashion rather than a rocking fashion, more than two constraints will always be needed.

To illustrate, consider the case of a double pendulum where the first link serves as $\theta(t)$ and where the second link is constrained by some smooth 2π-periodic function $\varphi(\theta(t))$. Assuming this system to start with zero initial velocity and with both links pointing upwards, slightly offset from the vertical axis, it is clear that the motion of the constrained double pendulum will never attain perpetual circular rotation, but will rather rock back and forth with the stable equilibrium at the center of the trajectory. The constraint chosen could conceivably be n-periodic, but this would simply mean that

the system would able to go a certain number of turns before stopping and returning back along the same trajectory. However, by switching constraints during the motion, a perpetual circular motion can be achieved, as by switching constraints an abrupt change of torque is applied on the constrained joint, increasing the energy of the system. This shows that there must be some sort of **asymmetry** of the constraints for an oscillating θ to achieve perpetual circular motion.

Consequently, a gymnast swinging from a horizontal bar will have to apply an abrupt change of torque during the swing to perpetuate the motion.

5 Control by Transversal Linearization

Controlling systems like the SemiQuad presents a wide array of challenges. First of all, by being able to walk the system is desired to exhibit some sort of periodic behavior – the gait. The gait consists of intervals of continuous dynamics and discrete jumps, making the SemiQuad system hybrid. Due to lack of ankle-joint actuation, the system is underactuated as well, of degree one.

The traditional and most common way of analyzing periodic systems focuses on the Poincaré first-return map. Such first-return maps are defined on a hypersurface transversal to the state-space trajectory at a point along the cycle. Through numerical search one can then determine characteristics of the periodic motion by how the first-return map behaves, transversally to the motion. This data might provide valuable numerical information from which one can extract properties of stability, sensitivity and area-of-attraction etc. However, as creating these maps usually relies on numerically integrating the dynamics along the cycle for a large number of initial conditions, the method will often be time-consuming, and ill-suited for real-time computation [3].

Transversal linearization in the context of this thesis, takes advantage of certain properties of virtually constrained systems. First and foremost, it can be shown that a virtually constrained system on the form of (12) always will be integrable [12, 3]. By introducing transversal error dynamics to the virtually constrained system, analytical solutions of the Poincaré maps become available in the vicinity of the periodic solution. Such Poincaré maps are referred to as *moving Poincaré sections*.

5.1 The continuous-in-time dynamics

The essence of transversal linearization is to replace the state vector $[q, \dot{q}]$ by a new set of generalized coordinates, $[\psi(t), x_\perp(t)]$, where $x_\perp(t)$ is a $2n -$

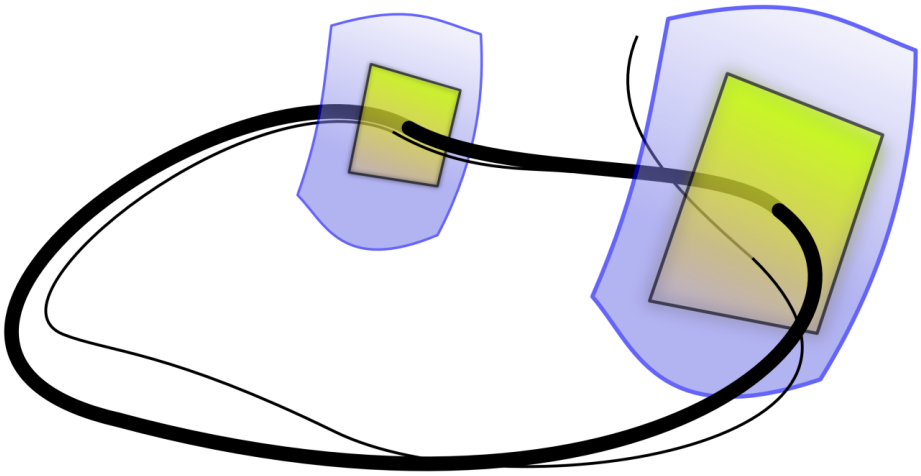


Figure 11: The Poincaré sections transversal to the motion trajectory, and the trajectory converging towards the stable periodic motion.

1-dimensional vector transversal to the motion trajectory, and the scalar $\psi(t)$ defines unique position along the motion. This means that at each point along the trajectory, a $2n-1$-dimensional, C^1-smooth hypersurface, or Poincaré section, $S(t)$ is spanned by the transverse coordinates, and on each such surface the transverse coordinates $x_\perp(t)$ describe the deviation of the system from a desired trajectory $[\theta, \varphi_i(\theta)]$. The dynamics of how these error coordinates behave in a vicinity of the desired trajectory is then formulated as an auxiliary linear system which upon stabilization ensures that the full dynamic system as a whole keep to its periodic motion.

The procedure described within this section is documented partially or in full in [12, 3, 13, 14].

The transverse coordinates are defined as

$$y_1 = q_1 - \varphi_1(\theta), \quad \ldots, \quad y_n = q_n - \varphi_n(\theta)\,. \tag{39}$$

It is obvious that both the above coordinates and their time derivatives

$$\dot{y}_1 = q_1 - \varphi_1'(\theta)\dot{\theta}, \quad \ldots, \quad \dot{y}_n = q_n - \varphi_n'(\theta)\dot{\theta}\,, \tag{40}$$

will be equal to zero along the orbit. As the moving Poincaré section is defined by $2n-1$ coordinates, it is also clear that these $2n$ coordinates along with θ will be excessive, and that one of the deviations y_n therefore can be written as a function of the other $y = (y_1, \ldots, y_{n-1})^T$ and θ

$$y_n = \varphi_n(\theta) + h(q_1, \ldots, q_{n-1}, \theta)\,. \tag{41}$$

To rewrite the original dynamics in terms of the new generalized coordinates, the both generalized velocities and accelerations must be rewritten as functions of these new coordinates

$$\dot{q} = L(\theta, y)\begin{bmatrix}\dot{y}\\ \dot{\theta}\end{bmatrix}, \quad \ddot{q} = L(\theta, y)\begin{bmatrix}\ddot{y}\\ \ddot{\theta}\end{bmatrix} + \dot{L}(\theta, y)\begin{bmatrix}\dot{y}\\ \dot{\theta}\end{bmatrix} \tag{42}$$

where the $n \times n$ matrix L is

$$L(\theta, y) = \begin{bmatrix}1_{(n-1)\times(n-1)},\ 0_{(n-1)\times 1}\\ \nabla h(\cdot)\end{bmatrix} + \begin{bmatrix}0_{n\times(n-1)},\ \Phi'(\theta)\end{bmatrix} \tag{43}$$

where $\nabla h(\cdot) = \left[\frac{\partial h}{\partial y_1}, \ldots, \frac{\partial h}{\partial y_{n-1}}, \frac{\partial h}{\partial \theta}\right]$ and $\Phi'(\theta) = \left[\frac{d}{d\theta}\varphi_1(\theta), \ldots, \frac{d}{d\theta}\varphi_n(\theta)\right]^T$.

For SemiQuad, the original set of virtual constraints dictated that this new set of coordinates were to be formulated as

$$\begin{bmatrix}q_1\\ q_2\\ q_3\\ q_4\\ q_5\end{bmatrix} = \begin{bmatrix}\theta\\ y_2 + \varphi_2(\theta)\\ y_3 + \varphi_3(\theta)\\ y_4 + \varphi_4(\theta)\\ y_5 + \varphi_5(\theta)\end{bmatrix}, \tag{44}$$

with corresponding derivatives. All $\varphi_i(\theta)$ were linear constraints. It is seen that the first generalized coordinate q_1 takes the role of the $y_n = \varphi_n(\theta) + h(q_1, \ldots, q_{n-1}, \theta)$, albeit with the simplest structure possible. By rewriting the coordinates in such a fashion a few fundamental elements of the transverse linearized controller can take form, starting with the

feedback transformation.

5.1.1 The feedback transform

The feedback transform provides a relation between the actuation, v, of the transverse dynamics and the actuation for the original system, u. By solving the Euler-Lagrange system (4) with respect to $\ddot{q}$

$$\ddot{q} = M(q)^{-1}\left(-C(q,\dot{q}) - G(q) + Bu\right) , \tag{45}$$

and inserting the resulting equation into the expression of acceleration in the new coordinates (42) defined above one obtains a transformation

$$L(\theta, y)\begin{bmatrix}\ddot{y}\\ \ddot{\theta}\end{bmatrix} = M(q)^{-1}\left(-C(q,\dot{q}) - G(q) + Bu\right) - \dot{L}(\theta, y)\begin{bmatrix}\dot{y}\\ \dot{\theta}\end{bmatrix} . \tag{46}$$

Thus, by moving solving the resulting equation with respect to $\ddot{y}$, the feedback transform takes the form

$$\begin{aligned}\ddot{y} = &\underbrace{EL^{-1}(\theta, y)M(q)^{-1}B}_{N(\theta,y)}u \\ &+ \underbrace{EL^{-1}(\theta, y)\left[-M(q)^{-1}(C(q,\dot{q}) + G(q)) - \dot{L}(\theta, y)[\dot{y},\, \dot{\theta}]^T\right]}_{R(\theta,\dot{\theta},y,\dot{y})}\Bigg|_{\substack{q_i = y_i + \varphi_i(\theta),\, i=1\ldots n-1 \\ q_n = \varphi_n(\theta) + h(\cdot) \\ \dot{q} = L(\theta,y)\left[\dot{y}^T,\, \dot{\theta}\right]^T}}\end{aligned} \tag{47}$$

where the matrix $E = [1_{(n-1)\times(n-1)},\, 0_{(n-1)\times 1}]$ is added to both sides to extract $\ddot{y}$ from $[\ddot{y},\, \ddot{\theta}]^T$. By defining $\ddot{y} = v$, the transformation

$$v = R(\theta, \dot{\theta}, y, \dot{y}) + N(\theta, y)u \tag{48}$$

is complete. As long as $L(\theta, y)$ is non-singular on the trajectory, the matrix $N(\theta, y)$ is invertible, and the transformation valid both ways. This transform was readily computed for the SemiQuad system; however, care had to be taken to adapt the structure of the matrices to our choice of

constraints. The alterations were mathematically trivial, though. For the feedback transform going from v to u, the equation was formed as a function of q rather than y and θ thus looking like

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

5.1.2 The transverse dynamics

The transverse dynamics describe the dynamics of the system (4) deviation from the desired trajectory defined by (12). By substituting the new error coordinates into (4), in the same fashion as done when substituting in the virtual holonomic constraints in section 1.2, an equation much like the α-β-γ-equation is obtained.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu \Big|_{\substack{q_i = y_i + \varphi_i(\theta),\, i=1\ldots n-1 \\ q_n = \varphi_n(\theta) + h(\cdot) \\ \dot{q} = L(\theta,y)\left[\dot{y}^T, \dot{\theta}\right]^T \\ \ddot{q} = L(\theta,y)\left[\ddot{y}^T, \ddot{\theta}\right]^T + \dot{L}(\theta,y)\left[\dot{y}^T, \dot{\theta}\right]^T}} \tag{49}$$

The resulting unactuated equation can be written as

$$\bar{\alpha}(y,\theta)(\ddot{y} + \varphi^{'}(\theta)\ddot{\theta} + \varphi^{''}(\theta)\dot{\theta}^2) + \bar{\beta}(y,\dot{y},\theta,\dot{\theta})(\dot{y} + \varphi^{'}(\theta)\dot{\theta}) + \bar{\gamma}(y,\theta) = 0 \tag{50}$$

and it is clear that this equation will be equal to zero on the orbit. For $y,\, \dot{y},\, \ddot{y} = 0$ this equation becomes the α-β-γ-equation (12). However, as (50) too equals zero on the orbit the reduced dynamics can be written as

$$\begin{aligned} \alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) =& \Big[\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) \\ - \bar{\alpha}(\cdot)(\ddot{y} + \varphi^{'}(\theta)\ddot{\theta} + \varphi^{''}(\theta)\dot{\theta}^2) &+ \bar{\beta}(\cdot)(\dot{y} + \varphi^{'}(\theta)\dot{\theta}) + \bar{\gamma}(\cdot)\Big] \\ =& g(y,\dot{y},\ddot{y},\theta,\dot{\theta},\ddot{\theta}) \end{aligned} \tag{51}$$

which is a valid representation as well. A useful property, taken from Hadamard's lemma for functions of multiple variables, states that

Lemma 2 *[5, Lemma 4.28] Suppose that $f(x, y)$ is smooth in a neighborhood*

of $(0,0)$*. Then suppose that* $f(x,0) = 0$ *for all x sufficiently close to* 0*. Then* $f(x,y) = yf_1(x,y)$ *for a smooth function* f_1 *defined in a neighborhood of* $(0,0)$*.*

Proof *This statement is easy to explain. By the following identities*

$$\int_0^1 \frac{\partial}{\partial t} f(x,ty)\,\mathrm{d}t = [f(x,ty)]_{t=0}^{t=1} = f(x,y) - f(x,0) = f(x,y) \tag{52}$$

it is seen that this equation is valid as long as $f(x,0) = 0$ *holds, and that* $f(x,y)$ *can thus, by Hadamard's lemma, be written as*

$$f(x,y) = \int_0^1 \frac{\partial}{\partial t} f(x,ty)\,\mathrm{d}t = y\int_0^1 \frac{\partial}{\partial y} f(x,ty)\,\mathrm{d}t = yf_1(x,y) \tag{53}$$

completing the proof. ∎

By this logic it should be clear that (51) can be written as

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = g_v(\cdot)\ddot{y} + g_{\dot{y}}(\cdot)\dot{y} + g_y(\cdot)y \tag{54}$$

where

$$g_v(\cdot) = g_{\ddot{y}}(\cdot)/\ddot{y}, \quad g_{\dot{y}}(\cdot) = g_{\dot{y}}(\cdot)/\dot{y}, \quad g_y(\cdot) = g_y(\cdot)/y\,. \tag{55}$$

As $y, \dot{y}, \ddot{y}$ all are equal to zero on the orbit, the limit of each of the functions is found by the use of l'Hôpital's rule. l'Hôpital's rule states that fractions where the limits of both nominator and denominator tends towards zero, can be found by finding the limits of the derivative of the nominator and denominator. Therefore, the coefficients $g_v(\cdot)$, $g_{\dot{y}}(\cdot)$, $g_y(\cdot)$ are in fact found as $\frac{\partial \bar{g}_{\ddot{y}}(\cdot)}{\partial \ddot{y}}$, $\frac{\partial \bar{g}_{\dot{y}}(\cdot)}{\partial \dot{y}}$, $\frac{\partial \bar{g}_y(\cdot)}{\partial y}$, respectively, which of course means that the rewrite from (51) to (54) could be considered a linearization of sorts.

5.1.3 The transverse linearization

In some vicinity of the orbit, the mechanical system (4) can now be rewritten as (48) and (54). However, to make the transverse coordinates $x_\perp$

complete, yet another dimension must be added, as $[y,\, \dot{y}]$ merely add up to $2(n-1)$.

It turns out that the scalar general integral from theorem 1 can provide this last dimension of our $2n-1$-dimensional Poincaré section. The time derivative of the general integral (20) along the solution

$$\alpha(\theta)\ddot{\theta} + \beta(\theta)\dot{\theta}^2 + \gamma(\theta) = W \tag{56}$$

can be shown to be on the form [3]

$$\dot{I}(\theta(t), \dot{\theta}(t), \theta_0, \dot{\theta}_0) = \frac{2\dot{\theta}(t)}{\alpha(\theta(t))}\{W - \beta(\theta(t)) \cdot I(\theta(t), \dot{\theta}(t), \theta_0, \dot{\theta}_0)\}\,, \tag{57}$$

where W then is the α-β-γ-equation (12). The use of this integral requires that $\alpha(\theta_\star(t)) \neq 0$ during the motion. If this isn't the case, then other identities can be used, like, for instance, $\alpha(\theta_\star(t)) \equiv 0$ during the relevant intervals of time. The error variables y and $\dot{y}$ does of course represent how much each of the joints deviate from being held invariant. The general integral, on the other hand, is a measure of how far the system is from following its desired path.

By substituting (54) into (57) one can compute the linearization of the dynamics in transverse coordinates $x_\perp$. The complete transverse coordinates now takes the form

$$x_\perp = [I(\theta(t), \dot{\theta}(t), \theta_0, \dot{\theta}_0),\, y_1,\, \ldots, y_{n-1},\, \dot{y}_1,\, \ldots, \dot{y}_{n-1}]^T\,, \tag{58}$$

and the linearized system is written

$$\dot{x}_\perp = \begin{bmatrix} a_{11}(t) & a_{12}(t) & a_{13}(t) \\ 0_{(n-1)\times 1} & 0_{(n-1)\times(n-1)} & 1_{(n-1)\times(n-1)} \\ 0_{(n-1)\times 1} & 0_{(n-1)\times(n-1)} & 0_{(n-1)\times(n-1)} \end{bmatrix} x_\perp + \begin{bmatrix} b_1(t) \\ 0_{(n-1)\times(n-1)} \\ 1_{(n-1)\times(n-1)} \end{bmatrix} v \tag{59}$$

where the coefficients of the matrix are given by

$$\begin{aligned} b_1(t) =& \dot{\theta}_\star(t) \frac{2g_v(\theta_\star(t), \dot{\theta}_\star(t), 0, 0)}{\alpha(\theta_\star(t))}\,, \\ a_{11}(t) =& -\dot{\theta}_\star(t) \frac{2\beta(\theta_\star(t))}{\alpha(\theta_\star(t))}\,, \\ a_{12}(t) =& \dot{\theta}_\star(t) \frac{2g_y(\theta_\star(t), \dot{\theta}_\star(t), \ddot{\theta}_\star(t), 0, 0)}{\alpha(\theta_\star(t))}\,, \\ a_{13}(t) =& \dot{\theta}_\star(t) \frac{2g_{\dot{y}}(\theta_\star(t), \dot{\theta}_\star(t), \ddot{\theta}_\star(t), 0, 0)}{\alpha(\theta_\star(t))}\,. \end{aligned} \tag{60}$$

5.2 The controller design

The basic, underlying idea of implementing this auxiliary system is that *any feedback control which manages to drive this linear system towards zero, will also keep the keep the constraints invariant.* A natural proposition for a linear controller comes on the form

$$v(t) = -K(t)x_\perp \tag{61}$$

where $K(t)$ is a matrix chosen to stabilize the linear system

$$\dot{x}_\perp = A(t)x_\perp(t) + B(t)v(t)\,. \tag{62}$$

by forcing the system

$$\dot{x}_\perp = [A(t) - B(t)K(t)]x_\perp(t) \tag{63}$$

to have distinct eigenvalues with negative real parts at all points during the periodic motion. As $A(t)$ and $B(t)$ are T-periodic, it is natural to assume that $K(t)$ is T-periodic too, although one could perhaps consider a stationary feedback gain matrix as well, as long as the closed-loop system is stable for the entire period. A periodic $K(t)$ was used for the orbital stabilization of the controller developed for SemiQuad in this thesis.

There are several ways of constructing a periodic $K(t)$. In literature on virtual holonomic constraints control based on transverse linearization, a common method is to let K be found by solving the *periodic Riccati differential equation* (PRDE) for the relevant period of time, from $(0, T_h)$. Using this controller, both control input and error deviation can be given arbitrary weight, and $K(\cdot)$ will be optimal with respect to this weighting. The Riccati equation differential equation

$$-\dot{P}(t) = P(t)A(t) + A^T(t)P(t) - P(t)B(t)R_c^{-1}(t)B^T(t)P(t) + Q(t) \quad (64)$$

is the solution of the optimization problem related to finding the control input $v(t)$ of the linear system (62) which minimizes the following quadratic performance index

$$J = x^T(t_f)W_c x(t_f) + \int_{t_0}^{t_f} x^T Q_c x + v^T R_c v \, \mathrm{d}t \quad (65)$$

where Q_c, $W_c \in \mathbb{R}^{n\times n}$ and $R_c \in \mathbb{R}^{m\times m}$ are smooth functions of time, and where $R_c > 0$, $W_c > 0$ and $Q_c \geq 0$. The product $x^T Q_c x$ is a measure of to what degree deviations from $x = 0$ should be allowed during the horizon interval (t_0, t_f), balanced by the product $v^T R_c v$ which punishes excessive use of actuation. The matrix W_c penalizes the controller for not reaching the desired value within the terminal time, t_f. The matrix Riccati equation plays a significant role within the theory of linear control and optimization, not only for creating feedback gain matrices, but also other tasks similar in structure, like, for instance, for generating optimal (with respect to noise filtering properties) state estimators [4]. For time-invariant linear systems the optimal feedback control is given by the simpler stationary, or algebraic Riccati equation (ARE) obtained by equating $\dot{P} = 0$. This equation corresponds to the infinite horizon quadratic cost function

$$J = \int_0^\infty x^T Q_c x + v^T R_c v \, \mathrm{d}t \,, \quad (66)$$

and the resulting controller design is often referred to as a linear quadratic

regulator-design (LQR).

Below are some of the properties and results related to the solution of the periodic Riccati differential equation briefly restated. For details see [8, 2]. The Riccati differential equation can be rewritten as two linear differential equations structured as a Hamiltonian. For our system that means that (64) can be written like

$$\frac{d}{dt}\begin{bmatrix} X \\ Y \end{bmatrix} = \underbrace{\begin{bmatrix} A & -BR_c^{-1}B^T \\ -Q & -A^T \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{67}$$

where $P = XY^{-1}$. Formulating the Riccati differential equation in this way allows us to define the *transition matrix*, or *fundamental matrix* as

$$\frac{\partial}{\partial t}\Phi_H(t, t_0) = H\Phi_H(t, t_0), \quad \Phi(t_0, t_0) = I_{2n} \tag{68}$$

which is the Hamiltonian(67) integrated with initial conditions set to identity matrix I_{2n}. That is, the transition matrix is the Hamiltonian subjected to the unit input, and the results tell us something about the eigenvalues of the system. For a time invariant linear system the transition matrix is

$$\Phi(t, t_0) = e^{A(t-t_0)}X(t_0), \quad X(t_0) = I \tag{69}$$

where I is the identity. With a time variant system the solution will be different but conceptually comparable. In particular, the eigenvalues of the transition matrix after one period $\Phi(t_o + T, t_0)$ are interesting, or alternatively, the eigenvalues at the next switching surface. The transition matrix at this point is called the *monodromy* matrix

$$\Psi_H(t_0) = \Phi_H(t_0 + T, t_0)\,, \tag{70}$$

and the eigenvalues at this point are termed *characteristic multipliers* or *Floquet multipliers*. These characteristic multipliers provide a measure of how the solutions of the system (67) will behave over periods. In fact,

for certain cases, the relation $x(t_0 + T) = \mu_j x(t_0)$ will hold, where μ_i is a characteristic multiplier, indicating that to ensure stability of the periodic system the characteristic multipliers should stay within the unit circle [2, chapter 5].

The controller constructed herein is based on matlab code associated with [13], generously donated to further the SemiQuad controller design. Specifically, the controller design is based on numerically solving a periodic Riccati differential equation (PRDE) for the relevant period of time, producing a series of feedback matrices which are then applied in sequence as time goes by, at each instant allowing optimal control. The original code was edited to run faster, reducing the calculation time from ~ 80 seconds to ~ 3 seconds. The algorithm is documented in [8, Algorithm 1], where it is labeled "The One-shot Generator Method" and the general procedure is briefly outlined below.

1. Compute the monodromy matrix $\Psi(t_0) = \Phi(t_0 + T, t_0)$ by finding the transition matrix for one period.

2. Compute the ordered real Schur form of the monodromy matrix $\Psi(t_0)$ which then decomposes $\Psi(t_0)$ into

$$\begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}^T \Psi(t_0) \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}, \tag{71}$$

   where $S_{11} \in \mathbb{R}^{n \times n}$ is upper quasi-triangular with n eigenvalues within the unit circle, and where $S_{22} \in \mathbb{R}^{n \times n}$ is upper quasi-triangular with n eigenvalues outside the unit circle. The columns of U_{11} and U_{21} then constitute the stable subspace of $\Phi(t_0)$.

3. Integrate the Hamiltonian once more over one period, but this time with initial condition $[X_0\, Y_0]^T = [U_{11}\, U_{21}]^T$

4. Group the solutions found into blocks of $[X(t)\, Y(t)]$ and compute the proper solution of the Riccati differential equation by $P(t) = X(t)Y^{-1}(t)$.

5.3 Controller implementation

The algorithm was implemented, and the period for each of the periodic phases of the motion was divided into N segments, each segment with an associated feedback gain matrix P_i which then would ensure that the auxiliary linear system was driven to zero, and that the motion of the SemiQuad dynamics would become orbitally stable.

The closed-loop system was implemented in Matlab, and structurally consisted of a series of steps where first the error coordinates

$$x(t)_\perp = [I(\theta_\star(t), \dot{\theta}_\star(t), \theta(0), \dot{\theta}(0)), y_1(t), \ldots, y_{n-1}(t), \dot{y}_1(t), \ldots, \dot{y}_{n-1}(t)]^T \tag{72}$$

were calculated. A Matlab routine for definite integration[5] of the general integral (20) from $\theta_\star(0)$ to $\theta(t)$. As previously stated, $I(\cdot)$ is a metric of much the dynamics deviate from the desired motion, transversally to the periodic orbit. The vectors y and $\dot{y}$ were calculated according to (39), and serve as a measure of how far the joint variables are from satisfying their individual constraints.

From the previously generated feedback matrices a linear interpolation of two matrices P_i, P_{i+1} was made with respect to the integration timer t and the discrete segments of the period. Using the interpolated feedback gain matrix $\tilde{P}$ and the error coordinates, the optimal input $v(t)$ of the transverse dynamics (62) was calculated using

$$v(\cdot) = -R_c^{-1} B^T \tilde{P} x_\perp \tag{73}$$

This input was then transformed to the input $u(t)$ of the original SemiQuad dynamics (4) using (48), and finally, the SemiQuad dynamics were solved with respect to $\ddot{q}$ and integrated using the general purpose Matlab integra-

[5]The Matlab routine `quad()` was used to integrate the general integral $I(\cdot)$ (20) over the interval from $\theta(0)$ to $\theta(t)$. `quad()` numerically evaluates integrals using an adaptive Simpson quadrature.

tion schemes `ode45` and `ode113`[6]. In algorithmic form the structure of the controller looks like

---

Procedure `Closed-loop system`

Input: Constraints PQ, initial configuration q_0.
$\theta_d(i) =$ IntegrateVirtualSystem$(q_0, PQ), i = 1 \dots N$
$R(i) =$SolveRiccati$(q_0, PQ), i = 1 \dots N$

while *exit event not reached* **do**
$q, t \longleftarrow$ from ODE45
$\tilde{\theta}_d, \tilde{R} =$ Interpolate(θ_d, R, t)
$y, \dot{y} =$ CalculateErrorCoordinates(q)
$I =$ CalculateGeneralIntegral$(\tilde{\theta}_d, q)$
$v = -R_c^{-1} B^T \tilde{P} [\, I,\, y,\, \dot{y}\,]^T$
$u = N^{-1}(q)(v - R(q, \dot{q}))$
$\ddot{q} = M^{-1}(-C\dot{q} - G + Bu)$
to ODE45 $\longleftarrow \dot{q}, \ddot{q}$
end

---

[6] `ode113` is better suited for stiff systems, and proved to be significantly faster than `ode45` for integrating the closed-loop system. Suggested in [8].

6 Closed-loop Properties

The closed-loop dynamics were run for the constraints corresponding to the gait found in section 4.2.3. Surprisingly, with very little consideration put into the choice of weighting matrices R_c and Q_c, it was interesting to observe that the feedback control still enabled SemiQuad to stay true to the trajectory defined by its virtual holonomic constraints.

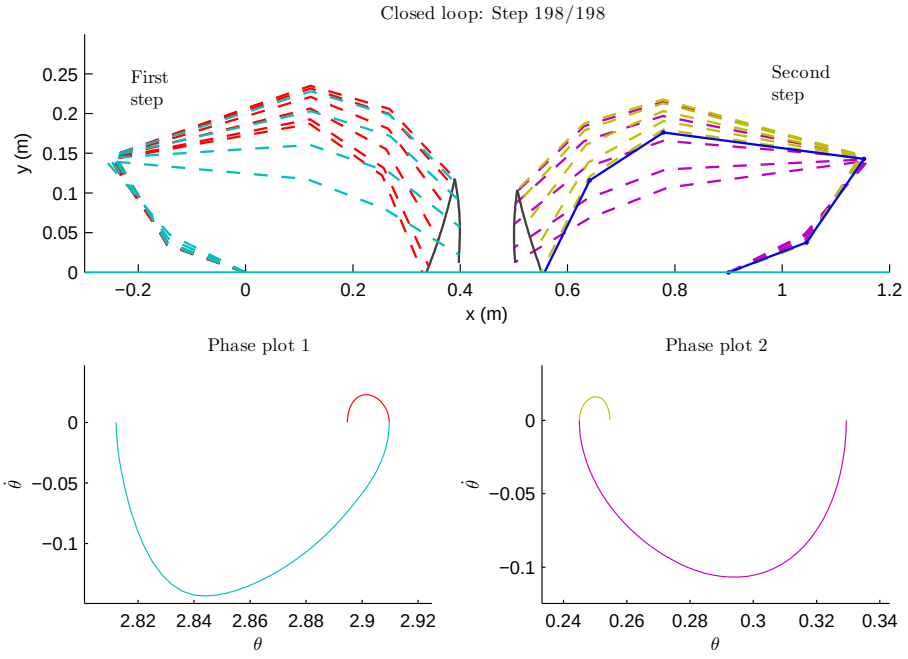


Figure 12: Animation and phaseplots detailing qualitative properties of the closed-loop dynamics. Note that the chronological sequence is from left to right, and that the spacing between the two SemiQuad animations is there just to separate the steps.

If comparing figure 12 with the corresponding animation of the reduced dynamics shown in figure 10, it is clear that from a qualitative point of view the controller seems to work well. These cases differ by the fact that while the reduced dynamics follows an ideal trajectory by *assuming* the constraints to be held invariant, the closed-loop dynamics obviously has to

apply actuation force to keep the constraints invariant and the motion on the ideal trajectory.

Looking at closed-loop periodic system (63) there are a few key points of interest: convergence rate; area of attraction; and resistance to disturbances. It is clear from figure 13 that for even for slight disturbances the dynamics might need several periods to fully converge to the periodic trajectory. R_c had to be weighted quite heavily in favor of the general integral $I(\cdot)$ to achieve fast convergence, however, too much weighting of this integral led to a deterioration of performance. The magnitude of $I(\cdot)$ is directly related to the speed and acceleration of $\theta(t)$, leading to the conclusion that constraints should have been chosen with this in mind. The θ of the first set of constraints has a nominal range from 2.895 to 2.91 radians, which is quite a short interval. More robust constraints would have been chosen to generate a larger range of θ, or, perhaps, with a different choice of θ altogether. This applies in particular for the case where the controller should be implemented physically.

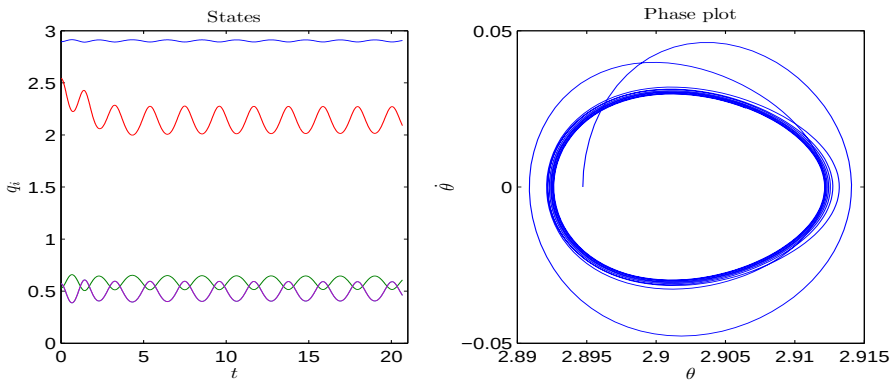


Figure 13: Plots showing the evolution of the dynamics starting with a slightly offset initial configuration for the first set of constraints.

However, in the simulation environment of Matlab, the controller seems to have created a fairly large area-of-attraction. With the exception of disturbances to the first link, $q_1 = \theta$, SemiQuad has an impressive capability to catch up with the periodic motion. As an example, in plot 15 the

controller is shown to be able to converge from an initial offset of one radian on q_2, reaching an orbit very close to the desired one within $3-4$ periods.

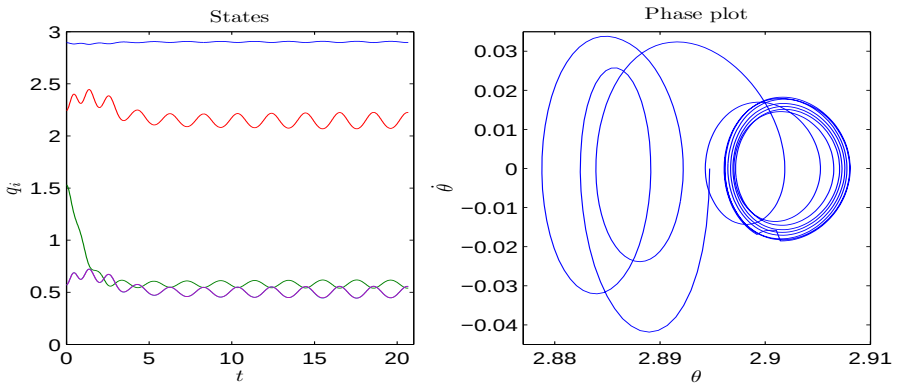


Figure 14: Plots showing the evolution of the dynamics starting with a substantial offset of initial configuration for the first set of constraints. Note how the periodic motion contracts towards the desired orbit.

So how then, exactly, does this controller differ from a conventional PD-controller? Well, it is clear that keeping the constraints invariant could easily be achieved by the use of a PD-controller, and *a* periodic motion could quite easily be maintained. What the PD-controller lacks is a notion of a *desired* periodic motion. The controller design based upon transverse linearization will have such a notion, and if the system should deviate from the desired orbit due to disturbances, the controller will try to catch up with the desired orbit, even though that action might render the constraints invariant for a while.

Finally, it should be pointed out that this controller, in this case, doesn't by itself attain orbital stability of the *gait*. Rather, the controller ensures that the periodic motions of which the gait is constructed attain orbital stability. To achieve orbital stability of the gait, as one periodic motion, the switching surfaces – the discrete points between the intervals of continuous dynamics – would have to be included into the design. This is the hybrid part of the design.

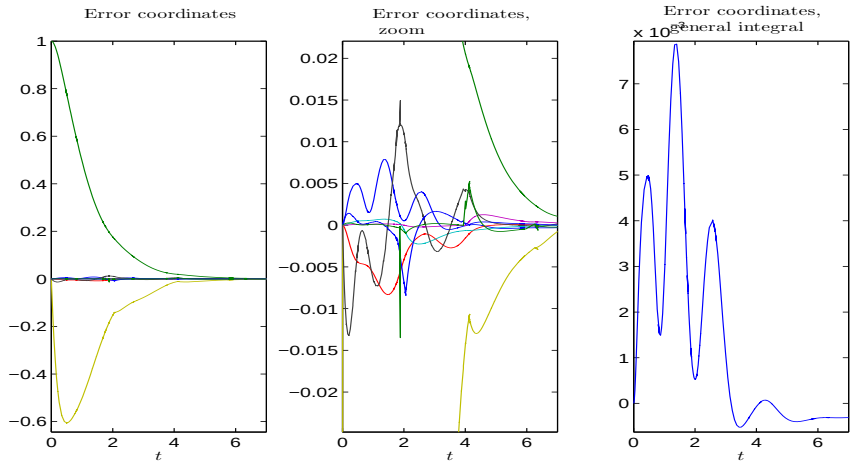


Figure 15: Plots showing the evolution of the error coordinates over time when the system is subjected to a large initial deviation from the ideal trajectory. From left to right the plot show: convergence of all error dynamics; close-up of error dynamics; and the deviation of the general integral.

7 Discussion

This thesis applies new methods of motion generation to a five degree of freedom system, SemiQuad. SemiQuad is modeled using the Euler-Lagrange Formalism, and is conceptually regarded as a five link kinematic chain where one end is connected to the ground by a unactuated revolving joint. The motion of the original model of SemiQuad was reparameterized in terms of virtual holonomic constraints using a generic method of constraint generation. However, due to properties of the original motion, the reparameterizing failed, and a gait was designed from new constraints instead. After creating the gait a controller based upon transverse linearization was implemented, and closed-loop behavior observed.

7.1 Reparamatrizing motion

This paper proposes a generic way of reparameterizing smooth motions of dynamical systems in terms of virtual holonomic constraints. The method ran into problems due to non-smoothness of the original motion, and was not used for the final motion generated for Semiquad. In hindsight, the problems which arose during the reparameterizing of motion might actually have been resolved. As noted in section 3.1, the non-smoothness of the original acceleration made the motion approximation very difficult, while the vectors of the original velocities and positions were quite smooth. There is a good chance that if the acceleration had been left out of the approximation of the constraints, a new smooth acceleration profile could have been designed, leaving the motion in terms of position and velocity more or less intact, but with new, smooth acceleration and a smooth nominal torque during the interval. This motion might then have been even better than the original one. Later tests indicated that the approximation sans acceleration was very good, although the reduced dynamics corresponding to these constraints were never tested.

Another interesting point about the method discussed above. While the large and cumbersome $\theta = F(\mathbf{q})$ found served as a monotonically increasing measure of progress when mapping constraints to the motion, if should be noted that **once the constraints are found, the analytical expression of $\theta = F(\mathbf{q})$ is not needed anymore**. Upon finding the constraints, the variable θ will be implicitly given by the corresponding reduced dynamics. An initial condition θ_0 will of course be needed to run the reduced dynamics, but as a rule, the function $F(\mathbf{q})$ can be disregarded after the constraints have been generated.

7.2 Generating motion

Perhaps just as important than tools for reparameterizing motion, are methods for generating new motion. Generating motion via both optimization and tuning was surprisingly easy once the structure of the constraints had been decided. This thesis uses polynomial constraints for convenience, and then in particular linear constraints, but there are other options, and any constraint is valid as long as it can provide a smooth trajectory for the relevant motion. By expressing constraints in a parametric fashion and by formulating criteria of optimality a wide range of steps can be generated. The gait which was generated for SemiQuad was created by tuning and displayed interesting properties of balance and periodicity. Had these constraints been made now, with the experience accumulated throughout the controller synthesis in mind, the gait would probably have looked a bit differently. The synthesis of the controller shed new light on why the constraints should be chosen to be robust and why θ preferably should span over a greater range of values. A more vigorous gait might also better show off the qualitative properties of the controller.

Emphasis should be put on the fact that the two methods of generating constraints shown in this thesis are quite different. In the section on reparametrizing, constraints are found from the motion, while in the section on generating constraints from scratch, motion is found from the constraints.

7.3 Control by transverse linearization

The transverse linearization is a novel solution to the challenging task of achieving orbital stability of a dynamical system. Being able to analytically create a linear auxiliary system describing how the dynamics behave transversally to the desired path of a periodic motion is a very useful thing indeed. Conceptually equivalent to the problem of linearizing a nonlinear system in the vicinity of an equilibrium to achieve stability, however, with time-varying, nonlinear coefficients requiring feedback designed to cope with periodic systems. Getting an intuitive feel for the mechanics of this controller is easy. However, solving the periodic Riccati differential equation presents a challenge, and mathematical software suites like Matlab lack routines to handle such systems in a straight-forward way.

It should be noted that there are alternatives to solving the periodic Riccati differential equation to stabilize the linear auxiliary system. Any feedback matrix K that stabilizes the transverse dynamics will work, and good convergence could be achieved using, for instance, pole placement.

8 Conclusion

Generating motion for walking systems is a complex task, not just because of the challenges posed by systems of this type, but because of a lack of general mathematical tools and principles capable of fully describing them. This thesis shows how the theory of virtual holonomic constraints can be applied to a certain class of systems to bridge this gap, and further shows how motion synthesis can be performed using simple geometrical relations between the generalized coordinates of these systems.

The fact that this arbitrarily complex motion can be imposed on the dynamical system through control of a comparably simple transverse linearized auxiliary system is surprising, and that this linear system can be developed in an analytical fashion is even more so. The union of these two concepts constitutes an important step towards achieving fluid, efficient and natural legged locomotion.

The need for robots to take on dangerous, hard and complex tasks is obvious, and for many such tasks, robotic replacements have already been found, resulting in safer and more efficient working environments. However, some tasks depend on the inherent human mobility and flexibility, and as these are skills yet to be properly mastered in the robotic world, human fire-fighters still have to physically enter flaming buildings, and police officers still have to enter hostile situations. Through research on biologically inspired modes of locomotion these tasks might one day be left for robots.

8.1 Future work

This thesis touches upon several concepts and ideas which could benefit from further research. Amongst these are

- First of all, incorporating the switching surfaces as a part of the motion would complete the theory presented within this thesis. By join-

ing the separate intervals of continuous dynamics by linearized switching surfaces orbital stability of the entire hybrid dynamical gait could be achieved. I sincerely believe that such a result would be of interest to the research community. Implementing this controller on the walking robot, SemiQuad, would also be of great academic interest.

- Generating and controlling other types of gait for SemiQuad. Generating slope or stair-climbing gaits might be possible, and would certainly be useful for many of the tasks one might consider using legged robots for. Faster motion suited for running and jumping would also be possible, however, this type of motion would present challenges not dealt with in this thesis.
- Applying the generic method of reparametrizing to a different system with smoother motion could bring the method some vindication. Ideally, a suite of tools could be implemented that did the the conversion automatically, from generating θ to controlling the system dynamics.
- In-depth theory of solving Riccati differential equations is of course outside the scope of this thesis. Still, doing further research into theory surrounding these equations would be of great personal interest for me.
- Finally, running a comparative study, generating motion and controlling SemiQuad by other available methods could shed some light on the pros and cons of each method.

References

[1] *Genetic Optimization Algorithm for Matlab*. May 2009. `http://www.geatbx.com/links/genetic_maximization_matlab_m_gordy.html` accessed June 6, 2009.

[2] Hisham Abou-Kandil, Gerhard Freiling, Vlad Ionescu, and Gerhard Jank. *Matrix Riccati Equations in Control and Systems Theory*. Birkhauser, July 2003.

[3] A.S, L.B, and I.R. Can we make a robot ballerina perform a pirouette? orbital stabilization of periodic motions of underactuated mechanical systems. *Annual Reviews in Control*, 32(2):200–211, December 2008.

[4] Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering, 3rd Edition*. Wiley, 3 edition, November 1996.

[5] J. W. Bruce and P. J. Giblin. *Curves and Singularities: A Geometrical Introduction to Singularity Theory*. Cambridge University Press, 2 edition, November 1992.

[6] Leonid B. Freidovich, Uwe Mettin, Anton S. Shiriaev, and Mark W. Spong. A Passive 2DOF Walker: Hunting for Gaits Using Virtual Holonomic Constraints. Technical report, Umeå University, 2008.

[7] Ambarish Goswami, Benoit Thuilot, and Bernard Espiau. Compass-like biped robot part i : Stability and bifurcation of passive gaits. http://hal.inria.fr/inria-00073701/en/, 1996.

[8] Sergei Gusev, Stefan Johansson, Bo Kågström, Anton Shiriaev, and Andras Varga. A numerical evaluation of solvers for the periodic riccati differential equation. Technical Report 09.03, Umeå University, Departement of Computing Science, 2009.

[9] D.G.E. Hobbelen and M. Wisse. Controlling the walking speed in limit cycle walking. *The International Journal of Robotics Research*, 27(9):989–1005, September 2008.

[10] Tad McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–82, April 1990.

[11] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1 edition, March 1994.

[12] A. Shiriaev, A. Robertsson, J. Perram, and A. Sandberg. Periodic motion planning for virtually constrained euler–lagrange systems. *Systems & Control Letters*, 55(11):900–907, November 2006.

[13] Anton S. Shiriaev and Leonid B. Freidovich. Transverse linearization for impulsive mechanical system with one passive link. April 2009.

[14] Anton S. Shiriaev, Leonid B. Freidovich, and Sergei V. Gusev. Computing a transverse linearization for mechanical systems with two or more degrees of freedom. October 2008.

[15] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, November 2005.

[16] M.W. Spong and F. Bullo. Controlled symmetries and passive walking. *Automatic Control, IEEE Transactions on*, 50(7):1025–1031, 2005.

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

Appendix

A Maple Code

A.1 SemiQuad dynamics

```
> restart;
> with(LinearAlgebra);
> with(CodeGeneration);
> with(VectorCalculus);
> with(ListTools, Reverse);
> alias(gamma=gamma);
> sqrdiff:=(p)→(diff(p[1], t))^2 + (diff(p[2], t))^2;
> R:=(theta)→Matrix(2, 2, [cos(theta), - sin(theta),
sin(theta), cos(theta)]);
> ELJacobian:=(F::list, V::list)→Matrix([seq([seq(diff(f, v),
v=V)], f=F)]);
> Polygen:=(L::list, X)→sort(add(L[i]*X^(i - 1),
i=1..nops(L)));
> SIZE:=5;
> pa[1]:=q[1];
> pa[2]:=pa[1] - q[2];
> pa[3]:=pa[2] - q[3];
> pa[4]:=pa[3] - q[4];
> pa[5]:=pa[4] - q[5];
> p[1]:=<(l[1] - s[1])*cos(pa[1]), (l[1] - s[1])*sin(pa[1])>;
> p[2]:=<l[1]*cos(pa[1]) + (l[2] - s[2])*cos(pa[2]),
(l[1]*sin(pa[1]) + (l[2] - s[2])*sin(pa[2])>;
> p[3]:=<l[1]*cos(pa[1]) + l[2]*cos(pa[2]) + s[3]*cos(pa[3]),
l[1]*sin(pa[1]) + l[2]*sin(pa[2]) + s[3]*sin(pa[3])>;
```

```
> p[4]:=<l[1]*cos(pa[1]) + l[2]*cos(pa[2]) + l[3]*cos(pa[3])
+ s[4]*cos(pa[4]), l[1]*sin(pa[1]) + l[2]*sin(pa[2]) +
l[3]*sin(pa[3]) + s[4]*sin(pa[4])>;
> p[5]:=<l[1]*cos(pa[1]) + l[2]*cos(pa[2]) + l[3]*cos(pa[3])
+ l[4]*cos(pa[4]) + s[5]*cos(pa[5]), l[1]*sin(pa[1]) +
l[2]*sin(pa[2]) + l[3]*sin(pa[3]) + l[4]*sin(pa[4]) +
s[5]*sin(pa[5])>;
> for j to SIZE do va[j]:=evalm(ELJacobian([pa[j]], [seq(q[i],
i=1..5)])&*Matrix(5, 1, [seq(dq[i], i=1..5)]))end do;
> for j to SIZE do vc[j]:=convert(map(simplify, map(combine,
evalm((ELJacobian([p[j]], [seq(q[i], i=1..5)])&*[seq(dq[i],
i=1..5)])), trig), size), Vector);end do;
> for i to SIZE do K[i]:=1/2*(m[i]*map(simplify,
map(combine, evalm(vc[i]&*Transpose(vc[i])), trig), size)) +
Iy[i]*evalm(va[i]*va[i])[1, 1]);end do;
> for i to 4 do Km[i]:=1/2*Ia[i]*dq[i + 1]*dq[i + 1]*N;enddo;
> for i to SIZE do P[i]:=p[i][2]*m[i]*g;enddo;
> Ks:=sum(K[a], a=1..5) + sum(Km[a], a=1..4);
> Ps:=sum(P[a], a=1..5);
> De:=ELJacobian([Ks], [seq(dq[i], i=1..SIZE)]);
> De:=map(combine, map(simplify, Transpose(ELJacobian(convert(De,
list), [seq(dq[i], i=1..SIZE)])), size), trig);
> Ge:=map(combine, map(simplify, expand(ELJacobian([Ps],
[seq(q[i], i=1..SIZE)]))), size, trig);
> Ce:=Matrix(SIZE);
> for k to SIZE do
for j to SIZE do
for i to SIZE do
Ce[k, j]:=Ce[k, j] + 1/2*(diff(De[k, j], q[i]) + diff(De[k, i],
q[j]) + diff(De[i, j], q[k])*dq[i]);
end do
end do
end do
> Ce:=map(simplify, map(combine, Ce, trig), size);
> De:=map(simplify, map(combine, De, trig), size);
```

```
> Ge:=Transpose(map(simplify, map(combine, Ge, trig), size));
```

Matlab export

```
> Matlab(De, resultname="D");
```

$$D = [\ldots]$$

```
> Matlab(Ce, resultname="C");
```

$$C = [\ldots]$$

```
> Matlab(Ge, resultname="G");
```

$$G = [\ldots]$$

A.2 Virtual Holonomic Constraints

```
> SQq[1]:=[1, 0];
> var:=theta(t);
> SQq[2]:=[PQ(1, 1), PQ(1, 2)];
> SQq[3]:=[PQ(2, 1), PQ(2, 2)];
> SQq[4]:=[PQ(3, 1), PQ(3, 2)];
> SQq[5]:=[PQ(4, 1), PQ(4, 2)];
> phi[1]:=Polygen(Reverse(SQq[1]), var);
> phi[2]:=Polygen(Reverse(SQq[2]), var);
> phi[3]:=Polygen(Reverse(SQq[3]), var);
> phi[4]:=Polygen(Reverse(SQq[4]), var);
> phi[5]:=Polygen(Reverse(SQq[5]), var);
> DYNSYSTEM:=evalm(De&*Transpose(<seq(ddq[i], i=1..5)>) +
Ce&*Transpose(<seq(dq[i], i=1..5)>) + Ge);
> DYN2:=DYNSYSTEM;
> for i to SIZE do
DYN2:=subs({ddq[i]=diff(phi[i], t, t), dq[i]=diff(phi[i], t),
q[i]=phi[i]}, evalm(DYN2));
end do;
> DYN2:=collect(evalm(DYN2), [diff(theta(t), t, t),
(diff(theta(t), t))^2]);
> DYN3:=DYN2;
> DYN3:=subs({diff(theta(t), t, t)=ddth, theta(t)=th,
diff(theta(t), t)=dth}, evalm(DYN3));
> DYNMATLAB:=collect(evalm(DYN3), [ddth, dth^2]);
```

Matlab export

```
> alpha:=coeff(DYNMATLAB[1, 1], ddth);
> Matlab(alpha, resultname="alpha");
```

$$alpha = [\ldots]$$

```
> beta:=coeff(DYNMATLAB[1, 1], dth^2);
> Matlab(beta, resultname="beta");
```

$$beta = [\ldots]$$

```
> gamma:=subs(ddth=0, dth=0, DYNMATLAB[1, 1]);
> Matlab(gamma, resultname="gamma");
```

$$gamma = [\ldots]$$

```
> with(linalg, inverse)
```

A.3 Feedback transformation

```
> M1:=Matrix(De); - 1;C1:=Matrix(Ce); - 1;G1:=Matrix(Ge);
> SubToError1:=q[1]=phi[1], dq[1]=diff(phi[1], t),
ddq[1]=diff(phi[1], t, t);
> M2:=subs(SubToError1, evalm(M1));
> C2:=subs(SubToError1, evalm(C1*Matrix(5, 1, [seq(dq[i],
i=1..5)]));
> G2:=subs(SubToError1, evalm(G1));
> for i from 2 to SIZE do
M2:=subs(dq[i]=(dy[i], diff(phi[i], t)), q[i]=(y[i], phi[i]),
ddq[i]=(ddy[i], diff(phi[i], t, t)), evalm(M2));
C2:=subs(dq[i]=(dy[i], diff(phi[i], t)), q[i]=(y[i], phi[i]),
ddq[i]=(ddy[i], diff(phi[i], t, t)), evalm(C2));
G2:=subs(dq[i]=(dy[i], diff(phi[i], t)), q[i]=(y[i], phi[i]),
ddq[i]=(ddy[i], diff(phi[i], t, t)), evalm(G2))
end do;
> M2:=Matrix(M2); - 1;C2:=Matrix(C2); - 1;G2:=Matrix(G2);
> phidot:=Matrix(5, 1, [seq(simplify((diff(phi[i], t),
1/(diff(theta(t), t)))), i=1..5)]);
> L:=Matrix(5, 5);
> for i to 5 do
L[i, 1]:=phidot[i, 1]
end do;
> for i to 4 do
L[i+1, i+1]:=1
end do;
> invL:=Matrix(inverse(L));
> E:=Matrix(4, 5, shape=identity);
> B:=Matrix(5, 4, [[0, 0, 0, 0], [1, 0, 0, 0], [0, 1, 0, 0],
[0, 0, 1, 0], [0, 0, 0, 1]]);
> Ldot:=Matrix(subs(, evalm(Matrix(map(diff, L, t))&*Matrix(5,
1, [seq(diff(y[i](t), t), i=1..4), diff(theta(t), t)]));
```

Matlab export

```
> MATSUB:={diff(theta(t), t, t)=ddth, theta(t)=th,
diff(theta(t), t)=dth};
> MatlabM2:=subs(MATSUB, M2);
> Matlab(MatlabM2, resultname="D");
```

$$D = [\ldots]$$

```
> MatlabC2:=subs(MATSUB, C2);
> Matlab(MatlabC2, resultname="C");
```

$$C = [\ldots]$$

```
> MatlabG2:=subs(MATSUB, G2);
> Matlab(MatlabG2, resultname="G");
```

$$G = [\ldots]$$

```
> MatlabinvL:=invL;
```

A.4 Transversal linearization

```
> DYNSYSTEMerror := DYNSYSTEM[1, 1];
> DYNSYSTEMabg := DYNSYSTEM[1, 1];
> SubToTheta := {q[1] = phi[1], dq[1] = diff(phi[1], t), ddq[1]
= diff(phi[1], t, t)};
> DYNSYSTEMabg := subs(SubToTheta, evalm(DYNSYSTEMabg));
> DYNSYSTEMerror := subs(SubToTheta, evalm(DYNSYSTEMerror));
> for i from 2 to SIZE do
DYNSYSTEMerror := subs({q[i] = y[i] + phi[i], dq[i] = dy[i]
+ diff(phi[i], t), ddq[i] = ddy[i] + diff(phi[i], t, t)},
evalm(DYNSYSTEMerror));
DYNSYSTEMabg := subs({q[i] = phi[i], dq[i] = diff(phi[i], t),
ddq[i] = diff(phi[i], t, t)}, evalm(DYNSYSTEMabg))
end do;
> TRANVERSDYN := DYNSYSTEMabg - VectorCalculus;
> TRANVERSDYN := simplify(TRANVERSDYN, size);
> for i from 2 to 5 do
Gy[i-1] := subs(y[i] = 0, diff(TRANVERSDYN, y[i]))
end do;
> Gy := Vector(4, Gy);
> for i from 2 to 5 do
Gdy[i-1] := subs(dy[i] = 0, diff(TRANVERSDYN, dy[i]))
end do;
> Gdy := Vector(4, Gdy);
> for i from 2 to 5 do
Gv[i-1] := subs(ddy[i] = 0, diff(TRANVERSDYN, ddy[i]))
end do;
> Gv := Vector(4, Gv);
```

Matlab export

```
> MATSUB := {diff(theta(t), t, t) = ddth, theta(t) = th,
diff(theta(t), t) = dth, seq(y[i] = 0, i = 2 ..  5), seq(dy[i]
= 0, i = 2 ..  5), seq(ddy[i] = 0, i = 2 ..  5)};
```

```
> Gy := subs(MATSUB, Gy);
> Matlab(Gy, resultname="Gy");
```

$$Gy = [\ldots]$$

```
> Gdy := subs(MATSUB, Gdy);
> Matlab(Gdy, resultname="Gdy");
```

$$Gdy = [\ldots]$$

```
> Gv := subs(MATSUB, Gv);
> Matlab(Gv, resultname="Gv");
```

$$Gv = [\ldots]$$