NTNU
Norwegian University of
Science and Technology

Case study research: the Butterfly Robot

Oskar Rømyhr Lund

Master of Science in Cybernetics and Robotics
Submission date: January 2018
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Case study research: the ”Butterfly” robot

Oskar Rømyr Lund

Department of Engineering Cybernetics, NTNU

Project description:
Non-prehensile manipulation refers to the art of manipulating an object without grasping it. Instead, the object may be pushed, thrown, balanced and so on. The concept of non-prehensile manipulation is highly complicated and needs a ton of research, which is why this thesis investigates the example of the ”Butterfly” robot. The ”Butterfly” robot does not depend on prehensile manipulation, but needs to have its motions planned and stabilized. Deriving an accurate model for the system is crucial because the theoretical system should behave as closely as possible to the real system. The better the theoretical model is, the more likely the system is to follow a planned trajectory, and the dependency on a controller diminishes.

It is of interest to have a thorough investigation of the ”Butterfly” robot, as this system is highly relevant for the inevitable progress in non-prehensile manipulation. The project should start all the way from scratch by deriving a model for the system, before a plan on how to generate feasible motions for the system should be presented.

The following items should be considered:

1. Choose a coordinate system that represents the behaviour of the ”Butterfly” robot in an efficient manner, and develop a kinematic model of the system.
2. Conduct research surrounding the commonly used assumption in the system dynamics and determine potential consequences.
3. Investigate motion planning for the underactuated system of the ”Butterfly” robot.
4. Implement the system in a numerical computing program and simulate the findings of the paper.

Supervisor: *Anton Shiriaev*

Co-supervisor: *Christian Fredrik Sætre*

January, 2018

Abstract

Robots that can only manipulate objects by grasping are very restricted and limited in their actions. Nevertheless, manipulation by grasping remains a common feature for robots, as it is the easiest way to always maintain control of the states in a system. By developing non-prehensile manipulation for robots the possibility of robots who can use human objects, with the same functionality as humans, is feasible and expected. To gain knowledge of non-prehensile manipulation, a benchmark example known as the ”Butterfly” robot is studied. The benchmark example from 1998 was meant to propose the challenge of developing a systematic technique for non-prehensile manipulation of a rolling motion.

This thesis considers an underactuated dynamic model, which is derived with all the surrounding theory carefully explained. Furthermore, a common assumption applied to the system is investigated to possibly reveal inaccuracies caused by the assumption itself. Virtual-holonomic-constraints-based motion planning is then applied to the system to find feasible trajectories for the ”Butterfly” robot.

Sammendrag

Roboter som bare kan manipulere objekter ved å gripe er svært begrenset i sine handlinger. Likevel er manipulering ved griping et svært vanlig trekk for roboter. Dette er fordi det er den enkleste måten å alltid opprettholde kontrollen av statene i systemet. Ved å videreutvikle ikke-gripende manipulasjon for roboter er muligheten for roboter som kan bruke menneskelige gjenstander, med samme funksjonalitet som mennesker, mulig og forventet. For å øke kunnskapen om ikke-gripende manipulasjon studeres et referanseeksempel kalt ”Sommerfugl”-roboten. Referanseeksemplet fra 1998 var ment som et forslag for å utvikle en systematisk teknikk for ikke-gripende manipulering av en rullende bevegelse.

Denne avhandlingen vurderer en under-aktuert dynamisk modell, som er utledet med all omliggende teori nøye forklart. Videre undersøkes en hyppig brukt antagelse for å muligens avsløre unøyaktigheter forårsaket av selve antagelsen. Virtual-holonomic-constraints-basert bevegelsesplanlegging blir deretter utført for å finne fysisk oppnåelige baner for ”Sommerfugl”-roboten.

Preface

This paper was developed and submitted during Fall 2017 as the final requirement for a Master of Science in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The paper is my thesis and documents my work done on non-prehensile manipulation of a particular robot called the ”Butterfly” robot.

Acknowledgements

I would like to thank my supervisor, Anton Shiriaev, for guidance and help with understanding the system at hand and for encouraging me to define and solve an extremely interesting and fun problem. Thanks are also due to PhD candidate Christian Fredrik Sætre for always responding and patently helping me when I stumbled over some of the more complicated topics.

Basis for thesis

The desired goal for this project was to expand the knowledge surrounding the ”Butterfly” robot and its possibilities. My supervisor, Anton Shiriaev, is one of the world’s leading experts on the ”Butterfly” robot, and it was he that requested more research about the system. There exists a common assumption that vastly simplifies the representation of the dynamics of the system. The downside of this assumption is that it distances the theoretical dynamics from the actual dynamics, which again might lead to certain consequences for behaviour of the system. Another topic that is in need of more research is the process of finding different feasible trajectories, which is a complicated matter.

Naturally, Shiriaev has been my main source throughout the semester, along with one of his published papers about the ”Bytterfly” robot, [1]. Maksim Surov is a co-author of the said paper, and with his team he has successfully created an original, functioning ”Butterfly” robot to perform experiments with. The robotics lab at NTNU contains two of these robots, which I had at my disposal. Unfortunately, I did not get to use them due to time

constraints stemming from the vast amounts of theoretical work needed to be done first. The main contribution from this thesis is the detailed and extensive theoretical work done. The theoretical work in this thesis was derived with pen and paper, while the experimental work was solely performed in MATLAB in the form of simulations.

Basic information about the ”Butterfly” robot was discussed with my supervisor. This included a description of how the system was composed, how the system behaved, and why the system needed further investigation and research. Also, big concepts like; choice of coordinate system, previously used assumptions, and finding feasible trajectories were heavily discussed throughout the year. When more concrete problems occurred, e.g., mathematical errors, distorted simulations, or just general information about how the thesis should be structured, PhD candidate Christian Sætre has helped me. Lastly, since Surov is responsible for the experimental setup of the ”Butterfly” robot, he has been consulted about the robot's physical design and layout.

Table of Contents

List of Tables

List of Figures

List of Symbols

The International System of Units (SI) is used for all the variables/symbols. ”The body fixed frame” or ”the body frame” refers to the body fixed frame of the figure eight shape, not the fixed frame of the ball. The ”intersection point” refers to the contact point between the ball and the frame. These default notations are used unless otherwise specified.

Symbols	**Units**	**Description**
x_f	-	x position of the frame’s center
y_f	-	y position of the frame’s center
$\theta_f = \theta$	rad	Rotation from inertial to fixed body frame
$x_b = x$	-	x position of the ball’s center
$y_b = y$	-	y position of the frame’s center
θ_b	rad	Rotation from inertial to fixed body frame of ball
$\vec{e}_i^i$	-	Directional unit vector
R	m	Effective radius of ball
R_f	m	Radius of the frame
R_b	m	Radius of the ball
r_f	m	Half the distance between the two plates of figure eight
w	m	The ball’s distance from the ideal curve
ψ	rad	Rotation from body frame to the ball’s body frame
φ	rad	Rotation from the body frame to the ball’s position
ϕ	rad	Rotation from the body frame to the intersection point
α	rad	Angle between the tangent at point (x’,y’) and x’-axis
$\hat{k}$	-	Unit vector in z-direction

$\vec{\delta} = \vec{\delta}(\phi)$	-	Vector from the origin to the intersection point
$\vec{\rho} = \vec{\rho}(\varphi)$	-	Vector from the origin to the center of the ball
$s = s(\varphi)$	-	Natural parameter of the curve formed by $\vec{\rho}$
$s_f = s_f(\phi)$	-	Natural parameter of the curve formed by $\vec{\delta}$
$\vec{\tau} := \frac{d\vec{\rho}}{ds}$	-	Unit tangent vector for ball
$\vec{\tau_f} := \frac{d\vec{\delta}}{ds_f}$	-	Unit tangent vector for frame
$\vec{n} := \hat{k} \times \vec{\tau}$	-	Unit normal vector for ball
$\vec{n_f} := \hat{k} \times \vec{\tau_f}$	-	Unit normal vector for frame
$\vec{\kappa} := \frac{d^2\vec{\rho}}{ds^2}$	-	Curvature vector for the ball's path
$\vec{\kappa_f} := \frac{d^2\vec{\delta}}{ds_f^2}$	-	Curvature vector for the frame
$\vec{r}_f$	m	Position of frame
$\vec{v}_f$	m s^{-1}	Velocity of frame
$\vec{\omega}_f$	rad s^{-1}	Angular velocity of frame
$\vec{r}_b$	m	Position of ball
$\vec{v}_b$	m s^{-1}	Velocity of ball
$\vec{\hat{\omega}}_b$	rad s^{-1}	Angular velocity of the ball about its own rotation
$\vec{\omega}_b$	rad s^{-1}	Angular velocity of the ball
x'	-	x-coordinate relative to the body frame
y'	-	y-coordinate relative to the body frame
$\bar{x}$	-	x-coordinate relative to the inertial frame
O	-	Origin of the inertial and body fixed frame
$\Pi(\theta)$	-	Three-dimensional rotation matrix
$\mathcal{K}_{f,\omega}$	J	Rotational kinetic energy of frame
$\mathcal{K}_{b,\omega}$	J	Rotational kinetic energy of ball
$\mathcal{K}_{b,v}$	J	Translational kinetic energy of ball
$\mathcal{K}$	J	Total kinetic energy
$\mathcal{P}$	J	Total potential energy
$\vec{g}$	m s^{-2}	Gravitational acceleration
m	kg	Mass of ball
J_f	kg m^2	Mass moment of inertia of frame
J_b	kg m^2	Mass moment of inertia of ball

u	kg m s^{-2}	Actuator of the frame
q	-	Vector of generalized coordinates
$M(q)$	-	The inertia matrix
$C(q, \dot{q})$	-	The Corilois and centrifugal matrix
$G(q)$	-	Gravity vector
$B(q)$	-	Coupling matrix
$\vec{F}_b$	N	Forces acting on the ball
F_n	N	Reaction force in the normal direction at intersection point
F_s	N	Friction force at the intersection point
μ	-	Friction coefficient of intersection point
$\vec{\xi}$	-	Vector representing position on butterfly shape
Υ	-	Matrix for differentiation of ξ
$\Theta(\varphi)$	-	Virtual holonomic constraint

Chapter 1

Introduction

1.1 Background and Motivation

The study of non-prehensile manipulation is highly relevant as robotic technology is getting more and more advanced. Trying to manipulate objects with more degrees of freedom than what can be directly actuated makes non-prehensile manipulation challenging in both theoretical and practical work. The goal is to make robots capable of manipulating human objects with the same functionality as humans themselves. Humans can control nonlinear systems and adapt to environments very well, while robots struggle because of the high complexity. Working in the robots advantage is their high bandwidth and quick sensors, which no human reaction can compete with. By developing robots to perform more complex actions, like walking on two legs [2][3], managing independent objects [4][5] or any other non-prehensile motion, they will eventually become a valued asset to everyday life. Some potential benefits to mastering non-prehensile manipulation could be new robot primitives, simpler manipulators, flexibility, increased workspace size and increased workspace dimensionality. For a more detailed explanation of these potential benefits, see [6], while an introduction to non-prehensile manipulation and the ”Butterfly” robot can be viewed in[1].

[1] https://www.youtube.com/watch?v=V30e77x8BQA

In cases of underactuated robots where the control system cannot directly influence all the degrees of freedom, manipulation becomes difficult [7]. The system has to have dynamics that represent both the robot and the object that is being manipulated. A benchmark example of one of these systems was published in 1998 by Lynch et al [8]. This benchmark example is called the ”Butterfly” robot and involves continuously manipulating a ball rolling around what is a butterfly-shaped frame. Two identical figure eight shaped plates are rigidly placed parallel to each other with a gap smaller than the diameter of the ball between them, as seen in Figure 1.1. An actuator is attached to the plates which controls the movement of the frame and thereby indirectly controls the ball’s motion. The ball is driven by the force of gravity and is not attached to the frame, which means that it could depart from the frame if not properly controlled.

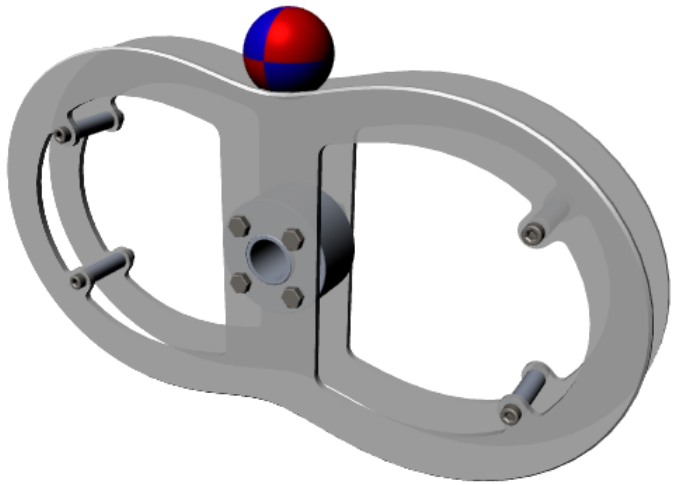


Figure 1.1: Schematic view of the ”Butterfly” robot

This thesis has based its methodologies of the work of Surov and Shiriaev et al [1]. The work done in [1] was published 17 years after the benchmark example was introduced, and it displayed their ability to accurately model the system dynamics of the ”Butterfly” robot and to plan feasible continuous one-directional motions for the ball on the frame. The published paper omitted the details of their approach, but proved that there existed a

functioning solution to the system. Surov and his coworkers are the leading researchers on the "Butterfly" robot and have built a successful physical system where they perform experiments[2]. Other papers about the "Butterfly" robot have not achieved as accurate and favorable results, e.g. [9] due too oversimplifications or [10] due to analytical errors.

Using continuous motion to generate stability is very useful and could be applied to more than just the "Butterfly" robot. Bipedal robots is an example of systems that could benefit from further research in this field. Continuing the work of [1] is highly relevant for not only this specific system, but for non-prehensile manipulation in general. This thesis focuses on an analytical analysis of the "Butterfly" robot and how feasible motions for the robot can be found.

1.2 Outline of the Thesis

This thesis is organized as follows; the choice of generalized coordinates and the dynamics of the system are given in Chapter 2. The process of finding parameters for the system, with and without a common assumption, are located in Chapter 3. An introduction to virtual holonomic constraints, a guide to motion planning and results are provided in 4. The results are discussed in Chapter 5, along with a discussion about the leap from theoretical work to the physical robot. Lastly, a conclusion and future work is presented in Chapter 6

Note: The notifications of a vector $\vec{a}$ & **a** are equivalent and interchangeable with each other. $\vec{a}$ is used throughout the paper, while **a** is used in the Appendix. This is because there are clusters of calculations which look cleaner without the arrow above the variable. Another important distinction is the difference between $\dot{a}$ & a'. While $\dot{a}$ is always differentiating with respect to time, a' is differentiating with respect to the variable the function is given by, e.g. $a'(x) = da/dx$ and $a'(y) = da/dy$. Both of these notes are worth mentioning even though they are well known mathematical facts.

[2] https://www.youtube.com/watch?v=kyvW5sOcZHU

Chapter 2

Dynamics of ”Butterfly” Robot

The first step in any control problem is to derive the system dynamics. This allows for a greater understanding of the system and will be a natural first step in creating a model that represents the physical behaviour of the ”Butterfly” robot. This chapter uses a Langrangian approach to step by step derive the equations of motion for the system.

2.1 Derive system dynamics

Before deriving the system dynamics there are certain assumptions about the ”Butterfly” robot that needs to be specified. First and foremost, the frame and ball are solid, rigid bodies which will not deform, meaning that there will not be line contact between the ball and frame. Secondly, both the frame and ball are assumed to be smooth objects without any imperfections to disturb motion. Thirdly, they are also assumed to have a uniform distribution of mass, which means that they both have a center of mass in their geometric center. The final assumption made is that the frame is constructed by two rigid figure eight shaped plates, as depicted in Figure 1.1, which are identical and perfectly aligned with one another, thereby ensuring the ball a 2-D rolling surface.

2.1.1 Reducing Degrees of Freedom

The assumptions mentioned gives the opportunity to represent the "Butterfly" robot as a two-dimensional system, as can be seen in Figure 2.1. The system is here divided into three separate frames; the reference frame (span of $\vec{e}_1^{\,0}$ & $\vec{e}_2^{\,0}$), the body fixed frame for the figure eight shape (span of $\vec{e}_1^{\,1}$ & $\vec{e}_2^{\,1}$), and the body fixed frame for the ball (span of $\vec{e}_1^{\,2}$ & $\vec{e}_2^{\,2}$). Note that the figure eight shaped plates that make up the frame of the robot is also referred to as the frame throughout the paper. This can be confusing, so it is important to look at the context in which it is used.

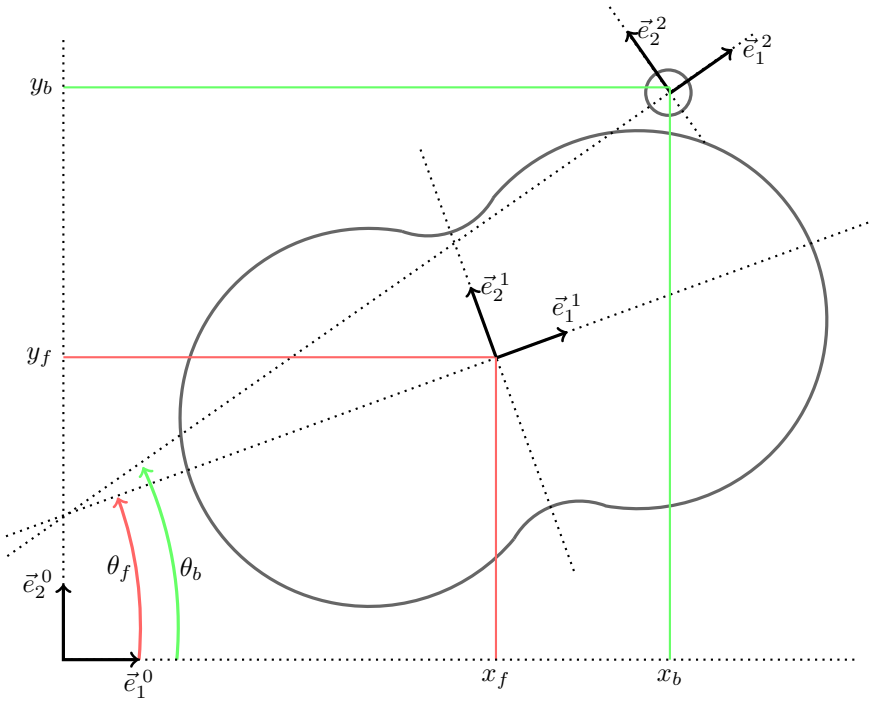


Figure 2.1: Six degrees of freedom in Cartesian coordinates

There are three degrees of freedom per rigid body (frame and ball), making it six degrees of freedom for the whole system. Horizontal displacement, vertical displacement and the body fixed frame's rotation compared to the reference frame is respectively represented by (x_i, y_i, θ_i), where i is a variable of frame or ball. Figure 2.1 clearly illustrates the corre-

sponding degrees of freedom and how they describe the robots behaviour.

An actuator is driving a rotating axing that is attached through the center of mass of the robot frame (also the geometric center). This is the only motor, and thereby the only external control the ”Butterfly” robot contains. By moving the inertial reference frame's origin to the body fixed frame centerpoint, the displacement of the robot frame will be constant ($x_f = y_f = 0$). Figure 2.2 is a closeup of the top right corner of the system, where the new position of the reference system is illustrated. This imposed constraint reduces the degrees of freedom from six to four; $(x_f, x_b, y_f, y_b, \theta_f, \theta_b) \rightarrow (x, y, \theta_f, \theta_b)$.

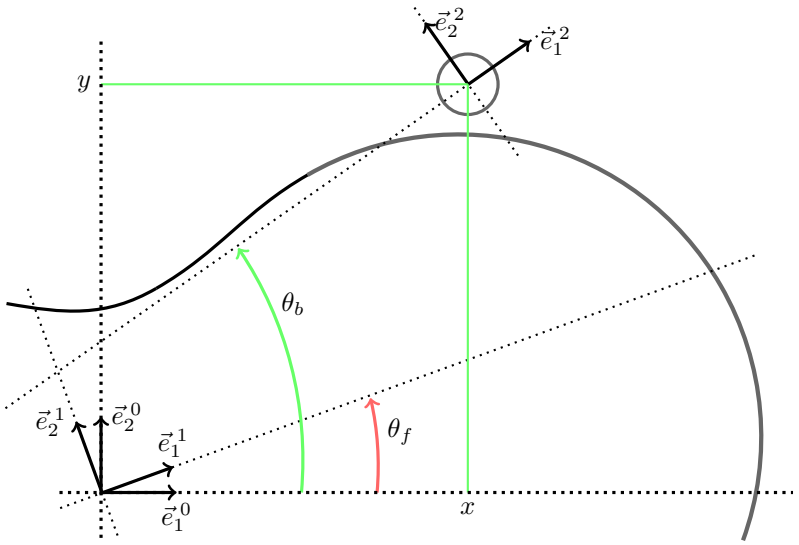


Figure 2.2: Four degrees of freedom in cartesian coordinates

2.1.2 Changing Coordinate System

Cartesian coordinates poorly describes how the two objects that make up the ”Butterfly” robot are linked together. Both rigid bodies are independently represented, which can be impractical when analyzing how the two objects work together, e.g. if the ball is even in contact with the frame. There are of course many different ways to represent the system, and many ways one could change the coordinate system. A practical coordinate system

is the priority, and therefore a polar coordinate system is introduced. The new coordinate system, and the manner in which it is implemented, was first introduced by Surov and Shiriaev at el [1].

Assuming that there will be no line contact, as mentioned before, leaves two other possibilities;

1. The ball has no contact with the frame, resulting in two independent systems where there is no way to control the motion of the ball.

2. The ball is in point contact with the frame, resulting in a ideal situation where the motion of the ball can be affected by the actuated frame. The ball rests between two plates, technically creating two points of contact, but an undramatic simplification is here applied to create just one point of contact. This simplification will not have any significant effect on the derived mechanics. Note that the mentioned setup will make it so the radius of the ball R_b is not the distance from the center of the ball to the point contact on the frame. This distance is called the effective radius R and is found by applying Phytagoras theorem to the triangle depicted in Figure 2.3,

$$R = \sqrt{R_b^2 - r_f^2}, \tag{2.1}$$

where r_f is half the distance between the two plates.

A variable w is introduced as a degree of freedom to decide which of the two circumstances mentioned above the system resides in. Together with another new degree of freedom s, they represent the position of the ball in reference to the body frame of the figure eight shape. If desired behaviour with point contact between the ball and frame is achieved, a smooth curve with an offset of the effective radius R is created by tracing the center of mass for the ball along the curvature traveled, as Figure 2.4 shows.

$\vec{\rho}$ denotes the vector from origin to the point closest to the ball's center on this curvature, while s denotes the curve from the initial starting position of the body frame of the

figure eight shape to the position of $\vec{\rho}$, depicted by the thick, black arc length in Figure 2.4. Every location of $\vec{\rho}$ can therefore be represented by the arc length s, resulting in a re-parametrization of $\vec{\rho} = \vec{\rho}(s)$.

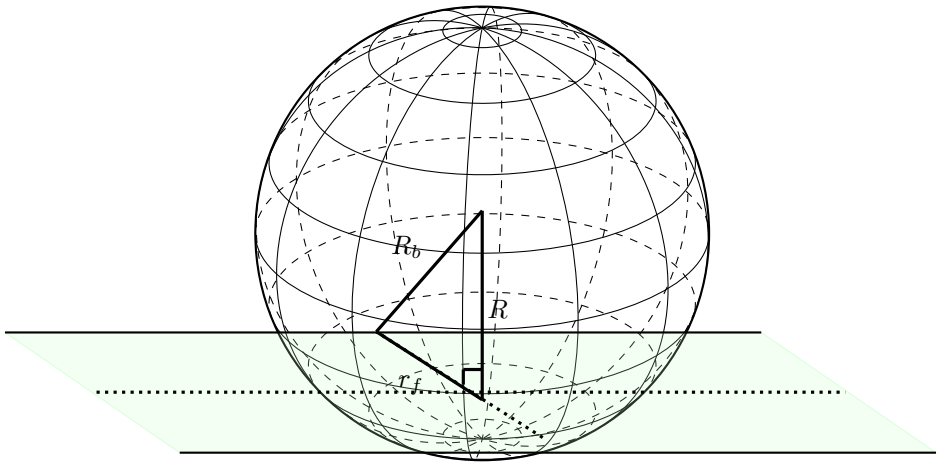


Figure 2.3: Pythagoras triangle in the intersection between frame and ball

By implementing the Frenet frame (see Appendix A) to the curvature, the so-called tangent $\vec{\tau}$, normal $\vec{n}$, and curvature $\vec{\kappa}$ unit vectors can be introduced. The tangent and normal vectors are perpendicular unit vectors that form a basis at any point on the curve as shown in Figure 2.4. Their definitions are as follows; $\vec{\tau} := \frac{d\vec{\rho}}{ds}$, $\vec{\kappa} := \frac{d\vec{\tau}}{ds}$ and $\vec{n} := \hat{k} \times \vec{\tau}$, with the unit vector $\hat{k} = [0, 0, 1]^T$.

Now s represents the traveled distance of the ball's center along the offset curve, and w represents the ball's distance from the ideal curve. By using these two degrees of freedom, the position of the ball with respect to the body fixed frame of the figure eight shape can be expressed as

$$r_b = \vec{\rho} + w\vec{n}, \tag{2.2}$$

where $w = 0$ is point contact and $w > 0$ is when the ball has departed from the frame.

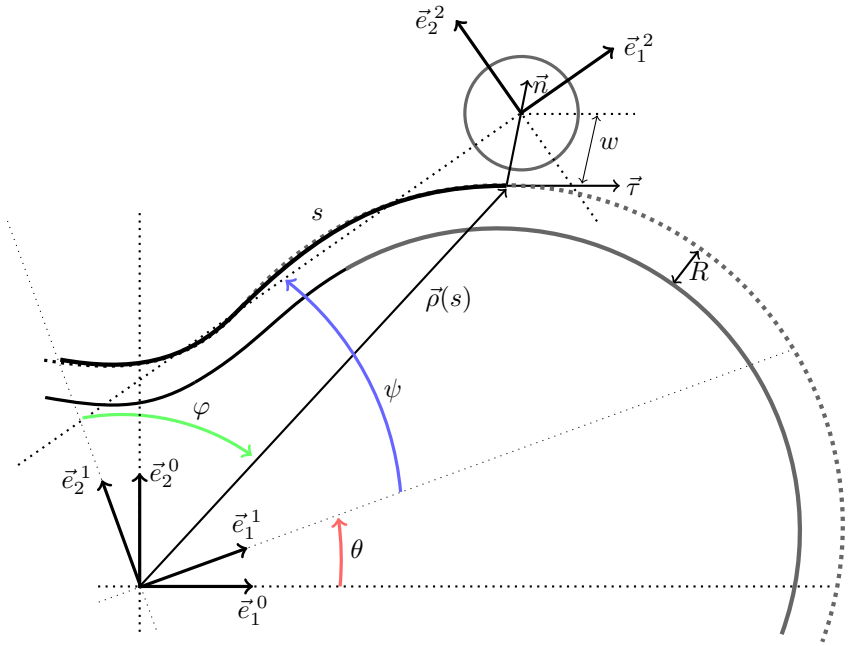


Figure 2.4: Four degrees of freedom in polar coordinates

To tansform the orientation of the ball to the body fixed frame of the figure eight shape, a third degree of freedom ψ is introduced. This new variable is the rotation between the robot's two body fixed frames (the figure eight shape and the ball). A fourth and last degree of freedom θ is introduced to represent the rotation between the inertial reference frame and the body fixed frame of the figure eight shape. The configuration of the overall system can thereby be expressed by a new set of variables with respect to the inertial reference frame. The change of coordinates $(x, y, \theta_f, \theta_b) \rightarrow (s, w, \psi, \theta)$ has yielded a more convenient system when working on the control design of the "Butterfly" robot. This new and practical set of generalized coordinates yields immediate information about the location of the ball with respect to the frame and is useful for control purposes.

2.2 Dynamic Equations for the System

The dynamic equations, or ”the equations of motion”, for a physical system (e.g. the ”Butterfly” robot) is a mathematical description of the system's behaviour by the use of dynamic variables. The dynamics of a system are general and can be derived by various approaches, most commonly by either Newton's second law or Euler–Lagrange's equations. In this thesis an Euler-Langrangian approach will be used to derive the equations of motion. For a more in-depth description of the concept of equations of motion, see [11]. To get a thorough understanding of a Langrangian approach and the benefits it can bring, see [12] to follow a simpler, yet similar control problem called ”Cart-Pendulum.”

2.2.1 System Constraints

In order to reduce the complexity of the system, two assumptions are made. These assumptions' purpose is to reduce the degrees of freedom so that the system is easier to work with. The first assumption made is:

- At any time moment, the ball and frame have one point of contact.

Since the ball and frame have one point of contact at all time without any deformation, $w = 0$, making this free variable redundant. This has removed the worry of mathematically expressing the instant when the ball elevates from the frame, leaving no way to control the ball. The second assumption made is:

- The ball rolls without slipping.

The ball's center will now always move along the offset curve without the intersection point ever sliding. Both assumptions contribute to constraining the system, reducing the degrees of freedom from four to two $(s, w, \psi, \theta) \rightarrow (\theta, \varphi)$. The two variables can sufficiently describe the configuration of the system, as 2.5 displays. θ is now an angle representing the orientation of the figure eight frame relative to the inertial coordinate system $O\bar{\mathrm{x}}\bar{\mathrm{y}}$, while φ is an angle representing the orientation of the ball's center relative to the body fixed coordinate system of the figure eight shape $O\mathrm{x}'\mathrm{y}'$.

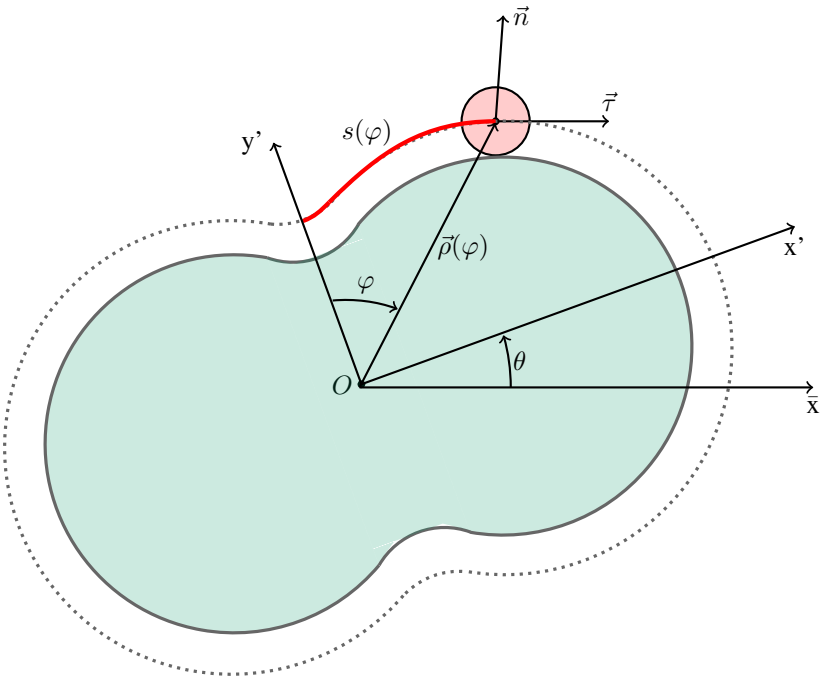


Figure 2.5: General overview of polar coordinate system

2.2.2 Kinematics

The kinematic equations of the system are derived to represent the motion of the two objects in the "Butterfly" robot. The position $\vec{r}_i$, the velocity $\vec{v}_i$ and the angular velocity $\vec{\omega}_i$ of both the frame and ball are examined.

Frame

The center of mass of the figure eight frame is located at the origin of rotation, which is also the origin of both the inertial frame of reference and the body fixed frame. This fact results in

$$\vec{r}_f = \vec{v}_f = [0, 0, 0]^T, \tag{2.3}$$

$$\vec{\omega}_f = \dot{\theta}\hat{k}. \tag{2.4}$$

Ball

Combining (2.2) with the constraint of one contact point results in $\vec{\rho}$ representing the center of the ball measured from the origin of the x'-y' coordinate system, seen in Figure 2.5. The position in the inertial frame is therefore given by

$$\vec{r}_b = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \vec{\rho} = \Pi(\theta)\vec{\rho}, \tag{2.5}$$

where $\Pi(\theta)$ is a three-dimensional rotation matrix about the z-axis, translating the expression to the inertial frame. See [13] for more information about the rotation matrix Π. For convenience, Π is used instead of $\Pi(\theta)$ in certain expressions to increase readability.

The velocity of the ball is found by differentiating the position with respect to time. Arc length is defined as $s := \int_0^{\varphi} \parallel \frac{d\vec{\rho}}{d\varphi} \parallel d\varphi$, which is dependant on φ. This is used in the following derivation of the ball's velocity,

$$\vec{v}_b = \frac{d\vec{r}_b}{dt} = \dot{\Pi}\vec{\rho} + \Pi\frac{d\vec{\rho}}{dt} = \omega_f \times (\Pi\vec{\rho}) + \Pi\frac{d\vec{\rho}}{ds}\frac{ds}{d\varphi}\frac{d\varphi}{dt} = \dot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \Pi\vec{\tau}s'\dot{\varphi}. \tag{2.6}$$

Using the constraint mentioned where the ball is assumed to not slip while rolling, the angular velocity of the ball about its own center rotation is expressed as

$$\vec{\omega}_{b_{center}} = -\frac{1}{R}\dot{s}\hat{k} = -\frac{1}{R}s'\dot{\varphi}\hat{k}. \tag{2.7}$$

The overall angular velocity is therefore given by

$$\vec{\omega}_b = \vec{\omega}_f + \vec{\omega}_{b_{center}} = \dot{\theta}\hat{k} - \frac{1}{R}s'\dot{\varphi}\hat{k} = (\dot{\theta} - \frac{1}{R}s'\dot{\varphi})\hat{k}. \tag{2.8}$$

2.2.3 Energy

To find the equations of motion, the system's kinetic and potential energy must be analyzed. The kinematics from the previous subsection are inserted into the definitions of the energies defined here. The frame and ball differs in contribution of both potential and

kinetic energy, and the energies are therefore studied separately.

Kinetic Energy

The kinetic energy of the system is defined as

$$\mathcal{K} = \mathcal{K}_f + \mathcal{K}_b, \tag{2.9}$$

where f represents *frame* and b represents *ball*.

$$\mathcal{K}_{f/b} = \frac{1}{2}\sum_{i=1}^{n}\left(m_i\vec{v}_i \cdot \vec{v}_i + I_i\vec{\omega}_i \cdot \vec{\omega}_i\right), \tag{2.10}$$

with the ball's mass m, translational velocity $\vec{v}$, moment of inertia I, angular velocity $\vec{\omega}$.

Frame

$$\mathcal{K}_f = \frac{1}{2}m\vec{v}_f \cdot \vec{v}_f + \frac{1}{2}J_f\vec{\omega}_f \cdot \vec{\omega}_f.$$

There is rotational energy from the frame, derived in (2.4), as the actuator can move it around, but there is no linear energy (2.3). This gives the following kinetic energy of the frame,

$$\mathcal{K}_f = \frac{1}{2}J_f\dot{\theta}^2. \tag{2.11}$$

Ball

$$\mathcal{K}_b = \frac{1}{2}m\vec{v}_b \cdot \vec{v}_b + \frac{1}{2}J_b\vec{\omega}_b \cdot \vec{\omega}_b.$$

The ball has both translational and rotational energy, see (2.6) and (2.8). This gives the

following kinetic energy of the ball,

$$\begin{aligned}
\mathcal{K}_b &= \frac{1}{2}m\Big(\big(\dot\theta\hat k\times(\Pi\vec\rho)+\Pi\vec\tau s'\dot\varphi\big)\cdot\big(\dot\theta\hat k\times(\Pi\vec\rho)+\Pi\vec\tau s'\dot\varphi\big)\Big)\\
&+\frac{1}{2}J_b\Big(\big(\dot\theta-\frac{1}{R}s'\dot\varphi\big)\hat k\cdot\big(\dot\theta-\frac{1}{R}s'\dot\varphi\big)\hat k\Big)\\
&=\frac{1}{2}m\Big(\dot\theta^2\underbrace{\big(\hat k\times(\Pi\vec\rho)\big)\cdot\big(\hat k\times(\Pi\vec\rho)\big)}_{||\vec\rho||^2}+2s'\dot\theta\dot\varphi\underbrace{\big(\Pi\vec\tau\big)\cdot\big(\hat k\times(\Pi\vec\rho)\big)}_{\hat k\cdot(\vec\rho\times\vec\tau)}+s'^2\dot\varphi^2\underbrace{\vec\tau^T\Pi^T\Pi\vec\tau}_{1}\Big)\\
&+\frac{1}{2}J_b\Big(\dot\theta^2-\frac{2}{R}s'\dot\theta\dot\varphi+\frac{1}{R^2}s'^2\dot\varphi^2\Big)\\
&=\frac{1}{2}m\Big(\dot\theta^2||\vec\rho||^2+2s'\dot\theta\dot\varphi\big(\hat k\cdot(\vec\rho\times\vec\tau)\big)+s'^2\dot\varphi^2\Big)\\
&+\frac{1}{2}J_b\Big(\dot\theta^2-\frac{2}{R}s'\dot\theta\dot\varphi+\frac{1}{R^2}s'^2\dot\varphi^2\Big).
\end{aligned}\tag{2.12}$$

Total Kinetic Energy

With both the separate kinetic energies calculated, the total kinetic energy of the system equals

$$\begin{aligned}
\mathcal{K}&=\mathcal{K}_f+\mathcal{K}_b\\
&=\frac{1}{2}\big(J_f+J_b+m||\vec\rho||^2\big)\dot\theta^2+\big(m\hat k\cdot(\vec\rho\times\vec\tau)-\frac{J_b}{R}\big)s'\dot\theta\dot\varphi+\frac{1}{2}\big(m+\frac{J_b}{R^2}\big)s'^2\dot\varphi^2.
\end{aligned}\tag{2.13}$$

Potential Energy

The potential energy of the system is defined as

$$\mathcal{P}=\mathcal{P}_f-\mathcal{P}_b,\tag{2.14}$$

where f represents *frame* and b represents *ball*.

$$\mathcal{P}_{f/b}=\frac{1}{2}\sum_{i=1}^{n}\big(m_i\vec g\cdot\vec r_i\big),\tag{2.15}$$

with the gravitational acceleration $\vec g=[0,g,0]^T$ and the position of particular body $\vec r$.

Frame

$$\mathcal{P}_f = m_f \vec{g} \cdot \vec{r}_f.$$

There is no gravitational force acting on the frame, and there is therefore no potential energy of the frame. Another way to see this is that the frame is centered at the origin and the position is therefore zero (2.3),

$$\mathcal{P}_f = 0. \tag{2.16}$$

Ball

$$\mathcal{P}_b = m\vec{g} \cdot \vec{r}_b.$$

The ball's position is affected by the gravitational force as it rolls around the frame and is defined in (2.5).

$$\mathcal{P}_b = m\vec{g} \cdot (\Pi\vec{\rho}). \tag{2.17}$$

Total Potential Energy

The gravitational force acting on the ball is the only potential energy in the system and the total potential energy is therefore expressed as

$$\begin{aligned} \mathcal{P} &= \mathcal{P}_f + \mathcal{P}_b \\ &= m\vec{g} \cdot (\Pi\vec{\rho}). \end{aligned} \tag{2.18}$$

2.2.4 Equations of Motion

The assumptions from Subsection 2.2.1 makes it possible to model the system with the generalized coordinates $q = [\theta, \varphi]^T$. This reduction of the degrees of freedom will simplify the equations of motion (EoM) substantially. To find the EoM, the energy derived in

the previous section will be used. Firstly, the Langrangian is defined as

$$\begin{aligned}\mathcal{L} &= \mathcal{K} - \mathcal{P} \\ &= \frac{1}{2}\big(J_f + J_b + m||\vec{\rho}||^2\big)\dot{\theta}^2 + \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\big)s'\dot{\theta}\dot{\varphi} \\ &+ \frac{1}{2}\big(m + \frac{J_b}{R^2}\big)s'^2\dot{\varphi}^2 - m\vec{g}\cdot(\Pi\vec{\rho}).\end{aligned} \tag{2.19}$$

The Euler-Langrange equation uses the Langrangian to derive the EoM. The Euler-Langrange equations for this system, defined in (B.7), are presented as

$$\frac{d}{dt}\left[\frac{d\mathcal{L}}{d\dot{\theta}}\right] - \frac{d\mathcal{L}}{d\theta} = u, \tag{2.20a}$$

$$\frac{d}{dt}\left[\frac{d\mathcal{L}}{d\dot{\varphi}}\right] - \frac{d\mathcal{L}}{d\varphi} = 0. \tag{2.20b}$$

The expressions of (2.20) can then be reformulated to represent the equations of motion seen below.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)u, \tag{2.21}$$

where u is the actuator driving the rotating figure eight frame and thereby affecting θ, and $B(q) = [1,0]^T$ is a coupling matrix function defined by the location of the controlled generalized forces affecting the system. To get a deeper understanding of how this previous and following process is done, see Appendix B.

The three algorithms presented in (2.22) is a workaround of the path from the Euler-Langrange equation presented in (2.20) to the full EoM presented in (2.21). The three algorithms gives a shortcut to achieving the inertia matrix $M(q)$, the Coriolis and centrifugal matrix $C(q,\dot{q})$ and the gravity matrix $G(q)$ used in the EoM.

$$m_{ij} := \frac{\partial}{\partial\dot{q}_i}\left(\frac{\partial\mathcal{K}}{\partial\dot{q}_j}\right), \tag{2.22a}$$

$$c_{jk} := \sum_{i=1}^{2} c_{ijk}(q)\dot{q}_i, \quad c_{ijk}(q) = \frac{1}{2}\left[\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k}\right], \tag{2.22b}$$

$$g_i := \frac{\partial\mathcal{P}}{\partial q_i}. \tag{2.22c}$$

The Inertia Matrix: $M(q)$

Using the algorithm in (2.22a) gives the following expressions,

$$
\begin{aligned}
m_{11} &= \frac{\partial}{\partial\dot{\theta}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\theta}}\right) = J_f + J_b + m||\vec{\rho}||^2, \\
m_{12} &= \frac{\partial}{\partial\dot{\theta}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\varphi}}\right) = \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\big)s', \\
m_{21} &= \frac{\partial}{\partial\dot{\varphi}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\theta}}\right) = m_{12}, \\
m_{22} &= \frac{\partial}{\partial\dot{\varphi}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\varphi}}\right) = \big(m + \frac{J_b}{R^2}\big)s'^2,
\end{aligned}
$$

which again leads to the inertia matrix,

$$
M(q) = \begin{bmatrix} J_f + J_b + m||\vec{\rho}||^2 & \Big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\Big)s' \\ \Big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\Big)s' & \Big(m + \frac{J_b}{R^2}\Big)s'^2 \end{bmatrix}. \tag{2.23}
$$

The Coriolis and Centrifugal Matrix: $C(q,\dot{q})$

In order to use the algorithm in (2.22b), certain sub-expressions are necessary.

$$
\begin{aligned}
\frac{\partial m_{ij}}{\partial\theta} &= 0, \quad \forall\, i,j = 1,2 \\
\frac{\partial m_{11}}{\partial\varphi} &= 2ms'\vec{\rho}\cdot\vec{\tau}, \\
\frac{\partial m_{12}}{\partial\varphi} &= \frac{\partial m_{21}}{\partial\varphi} = \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\big)s'' + \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\kappa})\big)s'^2, \\
\frac{\partial m_{22}}{\partial\varphi} &= 2\big(m + \frac{J_b}{R^2}\big)s's'',
\end{aligned}
$$

$$
\begin{aligned}
c_{111} &= \frac{1}{2}\left[\frac{\partial m_{11}}{\partial\theta}\right] = 0, \\
c_{112} &= \frac{1}{2}\left[2\frac{\partial m_{21}}{\partial\theta} - \frac{\partial m_{11}}{\partial\varphi}\right] = -ms'\vec{\rho}\cdot\vec{\tau}, \\
c_{121} &= \frac{1}{2}\left[\frac{\partial m_{11}}{\partial\varphi}\right] = ms'\vec{\rho}\cdot\vec{\tau}, \\
c_{122} &= \frac{1}{2}\left[\frac{\partial m_{22}}{\partial\theta}\right] = 0,
\end{aligned}
$$

$$
\begin{aligned}
c_{211} &= \frac{1}{2}\left[\frac{\partial m_{11}}{\partial \varphi}\right] = ms'\vec{\rho}\cdot\vec{\tau},\\
c_{212} &= \frac{1}{2}\left[\frac{\partial m_{22}}{\partial \theta}\right] = 0,\\
c_{221} &= \frac{1}{2}\left[2\frac{\partial m_{12}}{\partial \varphi} - \frac{\partial m_{22}}{\partial \theta}\right] = \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\big)s'' + \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\kappa})\big)s'^2,\\
c_{222} &= \frac{1}{2}\left[\frac{\partial m_{22}}{\partial \varphi}\right] = \big(m + \frac{J_b}{R^2}\big)s's''.
\end{aligned}
$$

These calculations, together with the algorithm, gives the following,

$$
\begin{aligned}
c_{11} &= c_{111}\dot{\theta} + c_{211}\dot{\varphi} = ms'\vec{\rho}\cdot\vec{\tau}\dot{\varphi},\\
c_{12} &= c_{121}\dot{\theta} + c_{221}\dot{\varphi} = ms'\vec{\rho}\cdot\vec{\tau}\dot{\theta} + \left[\big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\big)s'' + \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\kappa})\big)s'^2\right]\dot{\varphi},\\
c_{21} &= c_{112}\dot{\theta} + c_{212}\dot{\varphi} = -ms'\vec{\rho}\cdot\vec{\tau}\dot{\theta},\\
c_{22} &= c_{122}\dot{\theta} + c_{222}\dot{\varphi} = \big(m + \frac{J_b}{R^2}\big)s's''\dot{\varphi},
\end{aligned}
$$

which again leads to the following Coriolis and centrifugal matrix,

$$
C(q,\dot{q}) = \begin{bmatrix} ms'\vec{\rho}\cdot\vec{\tau}\dot{\varphi} & ms'\vec{\rho}\cdot\vec{\tau}\dot{\theta} + \left[\big(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\big)s'' + \big(m\hat{k}\cdot(\vec{\rho}\times\vec{\kappa})\big)s'^2\right]\dot{\varphi} \\ -ms'\vec{\rho}\cdot\vec{\tau}\dot{\theta} & \Big(m + \frac{J_b}{R^2}\Big)s's''\dot{\varphi} \end{bmatrix}. \tag{2.24}
$$

The Gravity Vector: $G(q)$

Using (2.22c) gives the following gravity vector elements,

$$
\begin{aligned}
g_1 &= \frac{\partial \mathcal{P}}{\partial \theta} = m\vec{g}\cdot(\Pi'\vec{\rho}),\\
g_2 &= \frac{\partial \mathcal{P}}{\partial \varphi} = m\vec{g}\cdot(\Pi\vec{\tau}s'),
\end{aligned}
$$

which leads to the gravity vector,

$$
G(q) = \begin{bmatrix} m\vec{g}\cdot(\Pi'\vec{\rho}) \\ m\vec{g}\cdot(\Pi\vec{\tau}s') \end{bmatrix}. \tag{2.25}
$$

where $\Pi' = d\Pi/d\theta$.

2.3 Constrained Dynamics

It is important to note that even though it is assumed that the ball never leaves the frame and that the ball never slips while rolling, that there are no guarantees that these assumptions will not be violated. By introducing the reaction force in the normal direction between the ball and the frame F_n and the friction force between the ball and the frame F_s, the assumptions can be represented by the two said forces. For the ball to never leave the frame,

$$F_n > 0, \tag{2.26}$$

and for the ball not to slip,

$$F_s \leq \mu F_n, \tag{2.27}$$

where μ is the friction coefficient between ball and frame. Conditions on F_n and F_s are mathematically calculated to display the restrictions needed on the forces to make sure the assumptions are not broken. By choosing the material of the frame and ball, the friction coefficient can be manually determined. A sufficiently large μ is chosen to always satisfy the condition (2.27).

To find a restriction that satisfies condition (2.26), and thereby automatically satisfies (2.27), it is necessary to derive the reaction forces. This can be achieved by applying Newton's second law to the ball,

$$\begin{aligned} \sum \vec{F} &= m\vec{a}, \\ \sum \vec{F_b} &= \Pi\vec{F_n} + \Pi\vec{F_s} - m\vec{g} \\ &= F_n\Pi\vec{n} + F_s\Pi\vec{\tau} - m\vec{g} = m\frac{d\vec{v}_b}{dt}. \end{aligned} \tag{2.28}$$

To be able to derive F_n and F_s, the acceleration $\vec{a}_b = d\vec{v}_b/dt$ needs to be found. The

velocity of the ball $\vec{v}_b = \dot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \Pi\vec{\tau}s'\dot{\varphi}$, derived in (2.6), is differentiated with respect to time.

$$\begin{aligned}
\frac{d\vec{v_b}}{dt} &= \ddot{\theta}\big(\hat{k} \times (\Pi\vec{\rho})\big) + \dot{\theta}\big(\underbrace{0 \times (\Pi\vec{\rho})}_{=0} + \hat{k} \times \frac{d(\Pi\vec{\rho})}{dt}\big) + \frac{d}{dt}\big(\Pi\vec{\tau}s'\dot{\varphi}\big) \\
&= \ddot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \dot{\theta}\big(\underbrace{\hat{k} \times \dot{\Pi}\vec{\rho}}_{=0} + \hat{k} \times \Pi\frac{d\vec{\rho}}{dt}\big) + \dot{\Pi}\vec{\tau}s'\dot{\varphi} + \Pi\frac{d\vec{\tau}}{dt}s'\dot{\varphi} + \Pi\vec{\tau}\frac{ds'}{dt}\dot{\varphi} + \Pi\vec{\tau}s'\ddot{\varphi} \\
&= \ddot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \dot{\theta}\hat{k} \times \Pi\big(\frac{d\vec{\rho}}{ds}\frac{ds}{d\varphi}\frac{d\varphi}{dt}\big) + \big(\underbrace{\dot{\theta}\hat{k} \times (\Pi\vec{\tau})}_{\dot{\theta}\Pi\vec{n}}\big)s'\dot{\varphi} + \Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \Pi\vec{\tau}s''\dot{\varphi}^2 + \Pi\vec{\tau}s'\ddot{\varphi} \\
&= \ddot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \underbrace{\dot{\theta}\hat{k} \times \Pi\vec{\tau}s'\dot{\varphi}}_{\dot{\theta}\Pi\vec{n}s'\dot{\varphi}} + \dot{\theta}\Pi\vec{n}s'\dot{\varphi} + \Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \Pi\vec{\tau}s''\dot{\varphi}^2 + \Pi\vec{\tau}s'\ddot{\varphi} \\
&= 2\Pi\vec{n}s'\dot{\theta}\dot{\varphi} + \Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \Pi\vec{\tau}s''\dot{\varphi}^2 + \Pi\vec{\tau}s'\ddot{\varphi} + \ddot{\theta}\hat{k} \times (\Pi\vec{\rho})
\end{aligned}$$

Find F_n

Left multiply (2.28) with $(\Pi\vec{n})^T$,

$$F_n\underbrace{(\Pi\vec{n})^T\Pi\vec{n}}_{=1} + F_s\underbrace{(\Pi\vec{n})^T\Pi\vec{\tau}}_{=0} - (\Pi\vec{n})^T m\vec{g} = (\Pi\vec{n})^T m\frac{d\vec{v}_b}{dt},$$

$$\begin{aligned}
F_n &= m(\Pi\vec{n})^T\left[\frac{d\vec{v}_b}{dt} + \vec{g}\right] \\
&= m(\Pi\vec{n})^T\left[2\Pi\vec{n}s'\dot{\theta}\dot{\varphi} + \Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \Pi\vec{\tau}s''\dot{\varphi}^2 + \Pi\vec{\tau}s'\ddot{\varphi} + \ddot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \vec{g}\right] \quad (2.29) \\
&= m(\Pi\vec{n})^T\left[2\Pi\vec{n}s'\dot{\theta}\dot{\varphi} + \Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \ddot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \vec{g}\right],
\end{aligned}$$

with $(\Pi\vec{n})^T(\Pi\vec{\tau}) = 0$.

Find F_s

Left multiply (2.28) with $(\Pi\vec{\tau})^T$,

$$F_n\underbrace{(\Pi\vec{\tau})^T\Pi\vec{n}}_{=0} + F_s\underbrace{(\Pi\vec{\tau})^T\Pi\vec{\tau}}_{=1} - (\Pi\vec{\tau})^T m\vec{g} = (\Pi\vec{\tau})^T m\frac{d\vec{v}_b}{dt},$$

$$
\begin{aligned}
F_s &= m(\Pi\vec{\tau})^T\left[\frac{d\vec{v}_b}{dt} + \vec{g}\right] \\
&= m(\Pi\vec{\tau})^T\left[2\Pi\vec{n}s'\dot{\theta}\dot{\varphi} + \Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \Pi\vec{\tau}s''\dot{\varphi}^2 + \Pi\vec{\tau}s'\ddot{\varphi} + \ddot{\theta}\hat{k}\times(\Pi\vec{\rho}) + \vec{g}\right] \\
&= m(\Pi\vec{\tau})^T\left[\Pi\vec{\kappa}s'^2\dot{\varphi}^2 + \Pi\vec{\tau}s''\dot{\varphi}^2 + \Pi\vec{\tau}s'\ddot{\varphi} + \ddot{\theta}\hat{k}\times(\Pi\vec{\rho}) + \vec{g}\right],
\end{aligned}
\tag{2.30}
$$

with $(\Pi\vec{\tau})^T(\Pi\vec{n}) = 0$.

$F_n > 0$ will make sure there is no elevation between the ball and the frame and $F_s \leq \mu F_n$ will make sure the ball rolls with no slip. If these constraints are broken, the system becomes invalid and unstable.

2.4 Model Evaluation

An evaluation of the system dynamics derived is done to make sure the results are credible, reliable and transferable to a physical system. First the validation of the model dynamics are discussed, before looking into the validation of the associated constraints.

2.4.1 Validation of System Dynamics

To make sure the derived model of the "Butterfly" robot is an accurate representation of the actual system, the model is evaluated. The system dynamics in (2.23)-(2.25) are derived as general dynamics, and should be valid for any frame used, e.g. butterfly, ellipse, circle, etc. To validate the derived dynamics, they will be compared to a new set of dynamics, which are derived specifically for a circle. By comparing these circle-specific dynamics with the general dynamics where the shape is chosen to be a circle, the general dynamics will either be confirmed as correct or they will be exposed as incorrect.

Deriving the specific dynamics of a circle frame balancing a ball on top of itself is substantially simpler than for a butterfly frame. The simplicity of the dynamics for a circle reduces

the chance of mathematical errors in the calculations, making it a well suited comparison-partner to validate the general dynamics. The derivation of the kinematics, energies and EoMs will not be as elaborate as for the general dynamics, as this thesis have already shown the step-by-step process in Section 2.2.

The frame is chosen as $\delta(\phi) = R_f$, where R_f is the radius of the circle. Note that since the frame has the shape of a circle, the two angles ϕ and φ are always identical. The distance from the origin to the center of the ball is $R_\delta = R_f + R$ all around the frame. The position of the ball is defined as $\vec{\rho} = R_\delta \vec{\xi}(\phi)$, with $\vec{\xi}(\phi) = \vec{\xi}(\varphi)$ representing the position on the frame.

Kinetic Energy

Frame

$$\mathcal{K}_f = \frac{1}{2} m \vec{v}_f \cdot \vec{v}_f + \frac{1}{2} J_f \vec{\omega}_f \cdot \vec{\omega}_f,$$

$$\vec{v}_f = 0 \text{ (no linear energy)},$$

$$\vec{\omega}_f = \dot{\theta}\hat{k} \text{ (in z-direction)},$$

$$\mathcal{K}_f = \frac{1}{2} J_f \dot{\theta}^2. \tag{2.31}$$

Ball

$$\mathcal{K}_b = \frac{1}{2} m \vec{v}_b \cdot \vec{v}_b + \frac{1}{2} J_b \vec{\omega}_b \cdot \vec{\omega}_b,$$

$$\vec{r}_b = \Pi R_\delta \vec{\xi},$$

$$\vec{v}_b = \frac{d\vec{r}_b}{dt} = \dot{\Pi} R_\delta \vec{\xi} + \Pi R_\delta \frac{d\vec{\xi}}{dt} = \omega_f \times (\Pi R_\delta \vec{\xi}) + \Pi R_\delta \vec{\xi}' \dot{\varphi},$$

$$\vec{\omega}_b = \vec{\omega}_f + \vec{\omega}_{b_{center}} = \dot{\theta}\hat{k} - \frac{1}{R} s' \dot{\varphi} \hat{k} = (\dot{\theta} - \frac{1}{R} s' \dot{\varphi}) \hat{k},$$

$$\begin{aligned}
\mathcal{K}_b &= \frac{1}{2}m\Big(\big(\dot\theta\hat k\times(\Pi R_\delta\vec\xi)+\Pi R_\delta\vec{\xi'}\dot\varphi\big)\cdot\big(\dot\theta\hat k\times(\Pi R_\delta\vec\xi)+\Pi R_\delta\vec{\xi'}\dot\varphi\big)\Big)\\
&+\frac{1}{2}J_b\Big(\big(\dot\theta-\frac{1}{R}s'\dot\varphi\big)\hat k\cdot\big(\dot\theta-\frac{1}{R}s'\dot\varphi\big)\hat k\Big)\\
&=\frac{1}{2}m\Big(\dot\theta^2\underbrace{\big(\hat k\times(\Pi R_\delta\vec\xi)\big)\cdot\big(\hat k\times(\Pi R_\delta\vec\xi)\big)}_{||R_\delta\vec\xi||^2=R_\delta^2}+2s'\dot\theta\dot\varphi\underbrace{\big(\Pi R_\delta\vec{\xi'}\big)\cdot\big(\hat k\times(\Pi R_\delta\vec\xi)\big)}_{\hat k\cdot(R_\delta\vec\xi\times R_\delta\vec{\xi'})=-R_\delta^2}\\
&+R_\delta^2\dot\varphi^2\underbrace{\vec{\xi'}^T\Pi^T\Pi\vec\xi}_{1}\Big)+\frac{1}{2}J_b\Big(\dot\theta^2-\frac{2}{R}\underbrace{s'}_{R_\delta}\dot\theta\dot\varphi+\frac{1}{R^2}\underbrace{s'^2}_{R_\delta^2}\dot\varphi^2\Big)\\
&=\frac{1}{2}m\Big(\dot\theta^2R_\delta^2-2R_\delta^2s'\dot\theta\dot\varphi+R_\delta^2\dot\varphi^2\Big)+\frac{1}{2}J_b\Big(\dot\theta^2-\frac{2R_\delta}{R}\dot\theta\dot\varphi+\frac{R_\delta^2}{R^2}\dot\varphi^2\Big).
\end{aligned}\tag{2.32}$$

Total Kinetic Energy

$$\begin{aligned}
\mathcal{K}&=\mathcal{K}_f+\mathcal{K}_b\\
&=\frac{1}{2}\big(J_f+J_b+mR_\delta^2\big)\dot\theta^2-R_\delta\big(mR_\delta+\frac{J_b}{R}\big)\dot\theta\dot\varphi+\frac{R_\delta^2}{2}\big(m+\frac{J_b}{R^2}\big)\dot\varphi^2.
\end{aligned}\tag{2.33}$$

Potential Energy

Frame

$$\begin{aligned}
\mathcal{P}_f&=m_f\vec g\cdot\vec r_f,\\
\vec r_f&=0,\\
\mathcal{P}_f&=0.
\end{aligned}\tag{2.34}$$

Ball

$$\begin{aligned}
\mathcal{P}_b&=m\vec g\cdot\vec r_b,\\
\vec r_b&=\Pi R_\delta\vec\xi,\\
\mathcal{P}_b&=m\vec g\cdot(\Pi R_\delta\vec\xi).
\end{aligned}\tag{2.35}$$

Total Potential Energy

$$\begin{aligned}\mathcal{P} &= \mathcal{P}_f + \mathcal{P}_b \\ &= m\vec{g} \cdot (\Pi R_\delta \vec{\xi}).\end{aligned} \tag{2.36}$$

Equation of Motion: Finding $M(q), C(q,\dot{q}),\ \&\ G(q)$ for Circle

Once again, the algorithms in (2.22) are used to derive the matrices of the EoM. The inertia matrix is found with (2.22a),

$$\begin{aligned}
m_{11} &= \frac{\partial}{\partial\dot{\theta}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\theta}}\right) = J_f + J_b + mR_\delta^2, \\
m_{12} &= \frac{\partial}{\partial\dot{\theta}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\varphi}}\right) = -R_\delta\big(mR_\delta + \frac{J_b}{R}\big), \\
m_{21} &= \frac{\partial}{\partial\dot{\varphi}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\theta}}\right) = m_{12}, \\
m_{22} &= \frac{\partial}{\partial\dot{\varphi}}\left(\frac{\partial\mathcal{K}}{\partial\dot{\varphi}}\right) = R_\delta^2\big(m + \frac{J_b}{R^2}\big),
\end{aligned}$$

$$M(q) = \begin{bmatrix} J_f + J_b + mR_\delta^2 & -R_\delta\big(mR_\delta + \frac{J_b}{R}\big) \\ -R_\delta\big(mR_\delta + \frac{J_b}{R}\big) & R_\delta^2\big(m + \frac{J_b}{R^2}\big) \end{bmatrix}. \tag{2.37}$$

Since the inertia matrix $M(q)$ is constant, all the elements of the Coriolis and centrifugal matrix $C(q,\dot{q})$ become zero, as (2.22b) shows,

$$\begin{aligned}
\frac{\partial m_{ij}}{\partial\theta} = \frac{\partial m_{ij}}{\partial\varphi} = 0, \\
c_{11} = c_{12} = c_{21} = c_{22} = 0, \\
G(q,\dot{q}) = 0.
\end{aligned} \tag{2.38}$$

The gravity vector function is found with (2.22c),

$$\begin{aligned}
g_1 &= \frac{\partial\mathcal{P}}{\partial\theta} = m\vec{g} \cdot (\Pi' R_\delta \vec{\xi}), \\
g_2 &= \frac{\partial\mathcal{P}}{\partial\varphi} = m\vec{g} \cdot (\Pi R_\delta \vec{\xi'}),
\end{aligned}$$

$$G(q) = \begin{bmatrix} m\vec{g} \cdot (\Pi' R_\delta \vec{\xi}) \\ m\vec{g} \cdot (\Pi R_\delta \vec{\xi}') \end{bmatrix}, \tag{2.39}$$

where $\xi' = d\xi/d\phi = d\xi/d\varphi$.

Comparison

The new-found dynamics and the general dynamics are compared to see if they produce identical system behavior. They are also compared with what one would expect in reality. The two scenarios will be simulated with a ball rolling on their frame. The initial conditions used for the comparison are $(\theta_0, \varphi_0, \dot{\theta}_0, \dot{\varphi}_0) = (0, \pi/2, 0, 0)$, shown in Figure 2.6. Since $\theta_0 = 0$, the inertial frame is identical to the body frame of the circle. This means that the inertial frame is hidden underneath the body frame of the circle. Note that there is no actuation $u = 0$ in either of the two simulations. The simulation parameters are shown in Table 4.1, and the corresponding results are shown in Figure 2.7 and Figure 2.8.

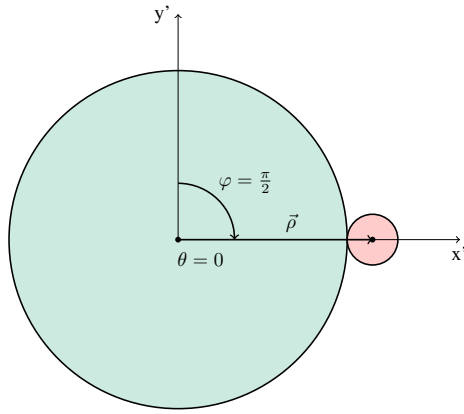
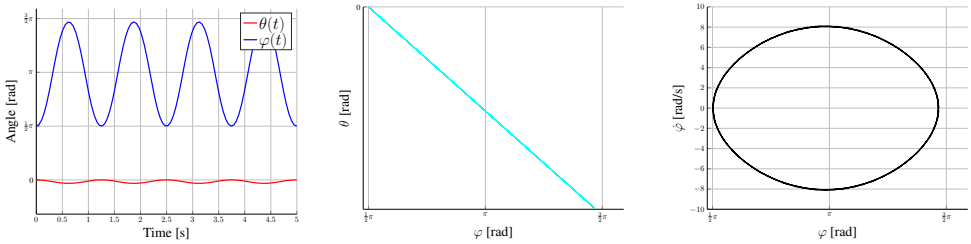


Figure 2.6: Schematic view of initial conditions

The simulations clearly show that the direct dynamics of a circle and the general dynamics produce the exact same results (Figure 2.7 VS. Figure 2.8). This substantially enforces the belief of correct system dynamics, but there is still a small chance of errors in both derived systems. The next step is to compare the results to what one would expect in reality. This

will exclude the possibility of two incorrect systems.

Since the simulations are modelled without any damping, the ball should theoretically make infinite oscillations on the frame. Figure 2.7c shows the phase plot of the system, and it clearly illustrates oscillating behaviour with both the angle φ and the angular velocity $\dot{\varphi}$ rising and sinking in a circular fashion. More information about phase trajectories will occur later in the thesis, when motion planning is being discussed. There is no actuation on the system and the initial conditions places the ball at the midpoint section of the circle. This leads to the expected behaviour of oscillations between $\varphi = \frac{\pi}{2}$ rad and $\varphi = \frac{3\pi}{2}$ rad, which is exactly what Figure 2.7a depicts. Figure 2.7b gives a description of the relationship between the two angles φ (representing the ball's center relative to the body fixed coordinate system) and θ (representing the body frame relative to the inertial frame). This relationship shows that a big increase in φ will lead to a fractional decrease in θ, as the two angles are defined in opposite directions. This makes sense with what is expected in reality as the ball slightly rotates the frame in the same direction as the ball travels past the bottom point of the circle, slightly shifting the weight of the system from side to side. This behavior between the two angles can be seen in Figure 2.7a. confirming the expected behaviour.

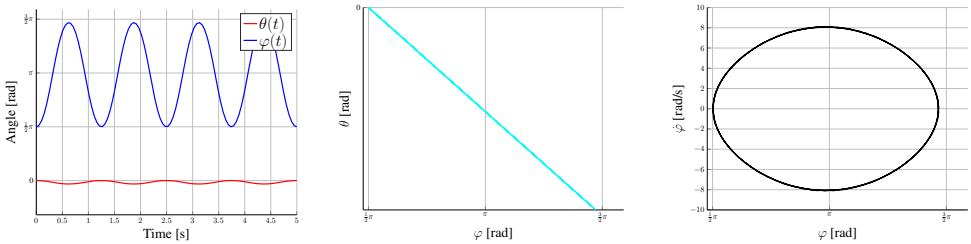


(a) Behaviour of the degrees of freedom

(b) Relationship between general coordinates

(c) Phase trajectory

Figure 2.7: Simulation results of the circle dynamics



(a) Behaviour of the degrees of freedom

(b) Relationship between general coordinates

(c) Phase trajectory

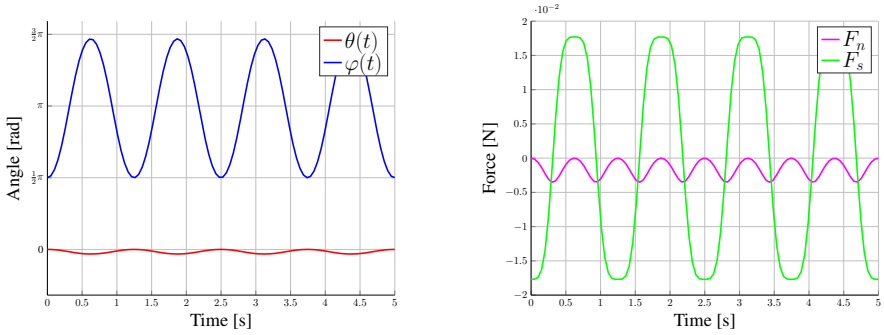
Figure 2.8: Simulation results of the general dynamics

Since the modelled system dynamics are identical and in accordance with the expected behaviour of reality, it can be concluded that the model of (2.21) is reliable. Another form of verification of the authenticity of the derived model can be found in [1], where the dynamics mirror the ones presented in (2.21)-(2.25).

2.4.2 Validations of Constraints

The constraints of (2.26) and (2.27), derived from the assumptions that the ball never slips or leaves the frame, are evaluated here. The evaluation will use the same example as for the validation of the dynamics; ball rolling on circle frame. The goal here is not to satisfy the constraints, but to make sure that the forces are acting in accordance with reality, thereby confirming the authenticity of the expressions in (2.26) and (2.27).

The previously found angular behaviour of the ball and frame are repeated in Figure 2.9a, while the corresponding forces are showed in Figure 2.9b. Since the ball is oscillating on the bottom half of a circular frame, the normal forces are expected to be less than zero, meaning that the ball in reality would depart from the frame. This behaviour can be seen by the normal force in the simulation. At the initial position $\varphi = \frac{\pi}{2}$ rad, the normal force is defined as $F_n = 0$ N. F_n then decreases as the ball 'rolls' down the frame, until $\varphi = \pi$ rad (very bottom of the frame), where the normal force starts to increase and the ball starts to roll upwards. The ball then 'rolls' up the frame and the normal force goes all the way back up to $F_n = 0$ N as $\varphi = \frac{3\pi}{2}$ rad. This gives the normal force in Figure 2.9b a periodic



(a) Behaviour of the degrees of freedom

(b) Behaviour of the normal and friction forces

Figure 2.9: Simulation results of the forces

motion with twice the period of the angle φ in Figure 2.9a. The no-slip constraint (2.27) is heavily dependent on the normal force F_n and the friction coefficient μ. The behaviour of the slip force F_s is therefore deemed less important to investigate. However, there is one crucial behavior which is defined by the slip force; since the initial slip force is less than the gravitational force, the ball accelerates and starts its rolling motion.

Since the forces of the system are in accordance with the expected behaviour of reality, it can be concluded that the normal force of (2.29), the slip force of (2.30), and the corresponding constraints of (2.26) and (2.27) are reliable.

Chapter 3

Finding Parameters for M, C, G

The next step after finding the system dynamics is to actually find the parameters used to describe the system. The representations seen in equations (2.23), (2.24) and (2.25) are all dependent on parameters which can be difficult to derive. Because of the difficulty in deriving accurate parameters, an assumption can be implemented, massively decreasing the complexity of the math. This chapter will derive the parameters for both cases, with and without the assumption. The two will then be compared to see if there is in fact any substantial difference. The problem behind this assumption and the assumption itself is first introduced, before a step by step method derives the two solutions.

3.1 Description of Problem

The general coordinates $q = [\theta, \varphi]^T$ are two free variables which are meant to describe the system, i.e. to represent the position of both the frame and the ball, and how they are positioned relative to each other. There is, however, no analytical expression present for the angle φ. There is therefore no analytical expressions for $s(\varphi)$ and $\vec{\rho}(\varphi)$, see Figure 3.1, since they are dependent on the variable φ. The previous solution to this issue has been to use a look-up-table containing numerical values, where a certain value binds the ball to a specific position on the frame. This look-up-table represents an approximation enforced on the system, which results in a less accurate physical system. Before the search for an

answer to this problem begins, it is important to figure out why a solution is needed.

3.1.1 The Problem of Expressing the Ball's Center

There is no way to always represent the center of the ball analytically with the angle φ as the function variable for all cases. There is, however, a way of representing the center of the ball if the ball is firmly planted on the robot frame. In other words, if the ball is following the curve of the butterfly shape with an offset of R, creating the arc length s. This fact can be used to find the center of the ball. By knowing the shape of the robot it is possible to find a vector $\vec{\delta}$ from the origin to the intersection point between the frame and ball. This new vector is dependent on an angle ϕ, which is the angle between the body frame of the robot and the vector $\vec{\delta}$, as seen in Figure 3.1. The vector is given as

$$\vec{\delta}(\phi) = \delta(\phi) \begin{bmatrix} \sin\phi \\ \cos\phi \\ 0 \end{bmatrix} = \delta(\phi)\vec{\xi}, \tag{3.1}$$

with δ usually given as $\delta(\phi) = a - b\cos(2\phi)$, since this function achieves the butterfly shape of the robot seen in figures throughout this paper. By changing the variable δ, the shape of the frame is changed (e.g. circle or ellipse). Note that the representation for zero degrees is straight up and not to the right, unlike the normal angle-representation of a unit circle. The x-position therefore gets represented by sinus and the y-position gets represented by cosine. Also note $\vec{\xi} = [\sin(\phi), \cos(\phi), 0]^T$, as this vector was used in the previous derivation of the circle dynamics, and will continue to be an essential tool in the derivation of the parameters describing the dynamics.

Using the normal vector $\vec{n}$ of the intersection point between the frame and ball, together with the effective radius of the ball R, the distance from the origin to the center of the ball can be expressed by

$$\vec{\rho}\big(\varphi(\phi)\big) = \vec{\delta}(\phi) + R\vec{n}(\phi), \tag{3.2}$$

which can be visually verified by Figure 3.1. This method only works when the ball is in

contact with the frame. The instant the ball looses contact with the frame, the representation becomes invalid due to the fact that the function assumes the distance from the frame to the ball is the effective radius R at all times.

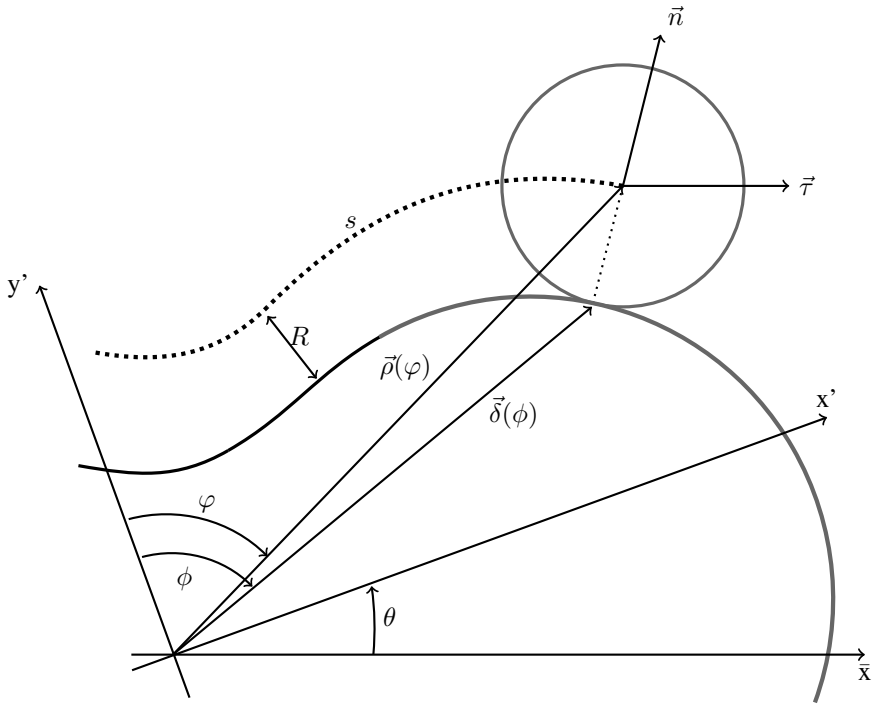


Figure 3.1: Closeup and notation of system

3.1.2 System With and Without the Assumption

The previous subsection clearly showed that there is a distinction between the angle representing the center of the ball and the angle representing the intersection point. Nevertheless, a common assumption for the "Butterfly" robot is to assume these two angles are identical, namely that $\varphi = \phi$. This is the commonly used assumption mentioned in the project desciption. Figure 3.2 shows an example where the frame of the robot is very large compared to the size of the ball. The two vectors $\vec{\rho}$ and $\vec{\delta}$ are almost identical, which makes the assumption $\varphi = \phi$ close to valid. Figure 3.3 shows an example where the robot frame

is the same size as before, but the ball has increased considerable in size. The two vectors $\vec{\rho}$ and $\vec{\delta}$ are now far from identical, making the assumption $\varphi = \phi$ invalid.

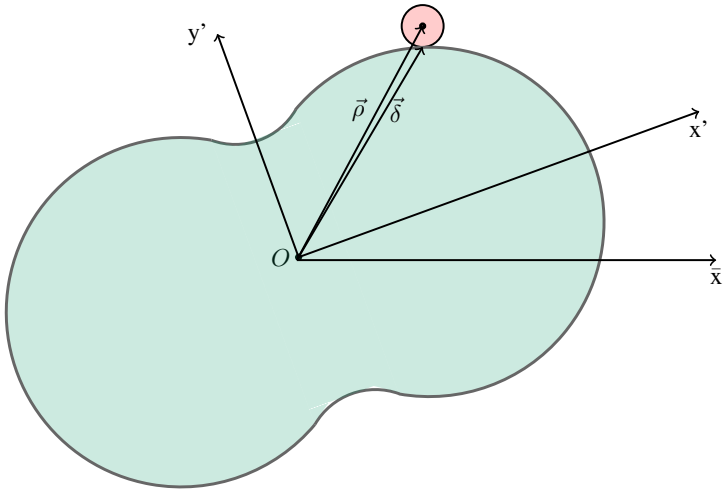


Figure 3.2: Illustration of when simplification is **close to** valid

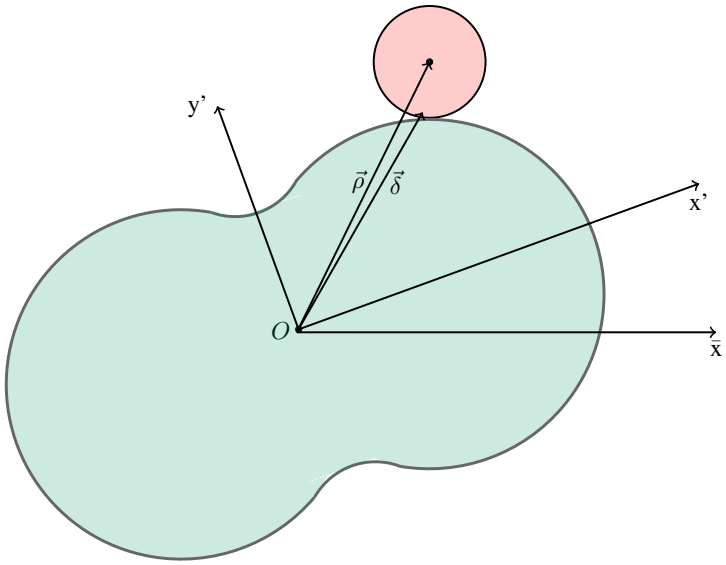


Figure 3.3: Illustration of when simplification is **far from** valid

3.1.3 Motivation to Find a Solution

There is always a desire to mathematically express the system in the most exact way possible. Unfortunately, this cannot always be achieved due to over-complexity, or due to the amount of resources needed to achieve it. If the assumption gives negligible differences in the results when compared to the exact system, the assumption can be considered acceptable. Analytically achieving an exact mathematically representation of the "Butterfly" robot has been attempted in [10], where the conclusion confirmed the high complexity of the system and suggested the use of easier shapes (e.g. circle) before looking at more advanced shapes (e.g. butterfly). Even with the knowledge that this is a highly complex system, an attempt will be made to achieve an analytical representation without the assumption $\varphi = \phi$. This will lead to a more accurate system compared to the look-up-table method previously used by some researchers.

The reasoning behind this improved accuracy lies within the inertia matrix $M(q)$ and the Coriolis and centrifugal matrix $C(q, \dot{q})$, located in (2.23) and (2.24). The assumption $\varphi = \phi$ gives a new vector describing the center of the ball. The difference from the one defined in (3.2) can be seen here,

$$\vec{\rho}_{\varphi=\phi} = \vec{\delta} + R\vec{\xi}, \tag{3.3a}$$

$$\vec{\rho}_{\varphi\neq\phi} = \vec{\delta} + R\vec{n}. \tag{3.3b}$$

$\vec{\rho}$ is a frequently used vector in describing the dynamics of the system through the matrices $M(q)$ and $C(q, \dot{q})$. This means that the small difference in representing $\vec{\rho}$ will affect the system dynamics. The inertia matrix $M(q)$ uses the vector $\vec{\rho}$, as well as a differentiation of the vector. Differentiating the small deviance between $\vec{\rho}_{\varphi=\phi}$ and $\vec{\rho}_{\varphi\neq\phi}$ will lead to an even bigger error, adding to the already mentioned difference between the two vectors. The Coriolis and centrifugal matrix $C(q, \dot{q})$ also performs this differentiation, not only once, but twice. This will again lead to an even bigger error.

With the knowledge concerning the size difference between the frame and the ball gained in the previous subsection, a suggestion could be to always use a big frame and a small ball. But importantly, this analytical problem is not reduced to just the ”Butterfly” robot. This issue affects all kinds of shapes and solving it could help other problems using non-prehensile manipulation. There is therefore a big desire to find a general analytical expression which will work for every shape. In this thesis there are three shapes that are used for verification of the expression derived: circle, ellipse and butterfly. Making one expression that works for all three shapes is considered a valid solution and a success. There might be an infinite number of shapes to validate, but there is neither room nor time to try all of them in this paper. Now that the problem itself is established, a step by step review will be presented, showing how the required parameters for the dynamics are derived.

3.2 Curvature

To understand how to derive the parameters needed, a deep understanding of curvature and offset curves is required. The parameters heavily depend on the derived expressions for $\vec{\tau}, \vec{n}$ and $\vec{\kappa}$. Figure 3.6 depicts the curvature vectors. Note the distinction of vectors with and without the subscription f. The vectors with the letter f refers to the vectors of the frame, which are separate from the vectors needed to calculate the parameters. This is a very important distinction, as s and s_f do not have the same curvature at every point. When the ball is in point contact with the frame, as is assumed in this paper, the complexity is somewhat reduced. This will be explained when deriving the curvature of the system.

3.2.1 Overview: Wanted Parameters

To get a better understanding of what is known and what is unknown, Table 3.1 is created. This gives an overview of the wanted parameters, needed by the equations of motion in (2.23)-(2.25).

Table 3.1: Parameters needed for M, C, G

Known	m	J_f	J_b	R
	$\hat{k}$	$\vec{g}$	$\Pi(\theta)$	$\Pi'(\theta)$
Unknown	$\vec{\rho}(\varphi)$	$\vec{\tau}(\varphi)$	$\vec{\kappa}(\varphi)$	$s'(\varphi)$
	$s''(\varphi)$			

The known parameters are either constants dependent on the system dimensions used (m, J_f, J_b, R), constant vectors ($\hat{k}, \vec{g}$) or rotational matrices (Π, Π'). The structure of the rotational matrices themselves are known, even though the parameter they depend on, θ, is unknown for now. More about the general coordinates θ and φ in the next chapter, under Section 4.1.

The shape of the butterfly frame is known, while the shape of the offset shape given by the ball's center is unknown. To find the unknown parameters describing this offset curve, seen in Table 3.1, a translation from the known shape to the unknown shape must take place. The vector describing the known butterfly shape $\vec{\delta}(\phi)$ is defined in (3.1) and repeated here, along with the definition of $\vec{\xi}(\phi)$. To increase simplicity for further calculations, the following is defined for $\vec{\xi}(\phi)$ and $\vec{\delta}(\phi)$,

$$\vec{\xi} = \begin{bmatrix} \sin\phi \\ \cos\phi \\ 0 \end{bmatrix}, \tag{3.4a}$$

$$\vec{\xi}' = \begin{bmatrix} \cos\phi \\ -\sin\phi \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \vec{\xi} = \Upsilon\vec{\xi}, \tag{3.4b}$$

$$\vec{\xi}'' = \begin{bmatrix} -\sin\phi \\ -\cos\phi \\ 0 \end{bmatrix} = -\vec{\xi}, \tag{3.4c}$$

$$\vec{\delta} = \delta\vec{\xi}, \tag{3.5a}$$

$$\vec{\delta}' = \delta'\vec{\xi} + \delta\vec{\xi}' = (\delta' I + \delta\Upsilon)\vec{\xi}, \tag{3.5b}$$

$$\vec{\delta}'' = \delta''\vec{\xi} + 2\delta'\Upsilon\vec{\xi} - \delta\vec{\xi} = (\delta'' I + 2\delta'\Upsilon - \delta I)\vec{\xi}. \tag{3.5c}$$

3.2.2 Introducing New Angle α

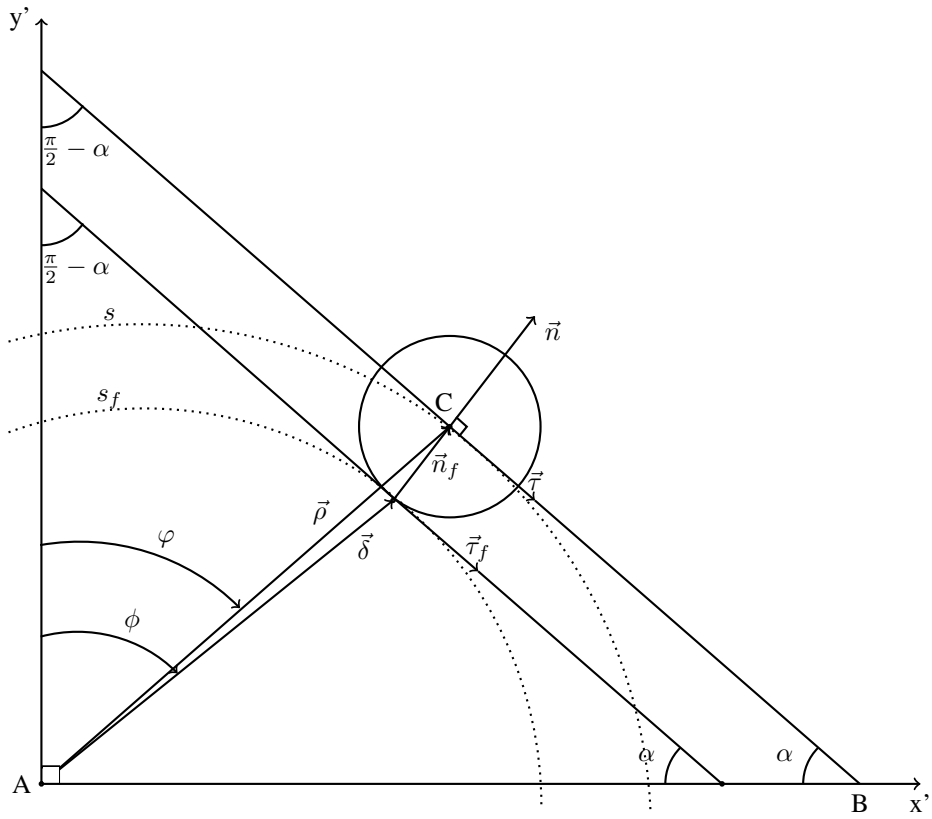


Figure 3.4: Closeup of the top-right corner of the butterfly: Introducing α

Figure 3.4 is a tight closeup of the top-right corner of the "Butterfly" robot, where the robot frame and the arc length describing the ball's path are displayed by the two dotted lines s_f and s. A new angle α is introduced to help in the process of analytically expressing certain expressions. The new variable α is the angle between the tangent at a point and the

positive direction of the x'-axis. The definition given to α, described here, is valid for all shapes and curves. Figure 3.4 gives a visual understanding of the information explained. Three intersection points (A,B & C) are included in the figure to easily isolate the triangle created by these notations and to increase the understandability in the following mathematical processes dependent on this isolated triangle.

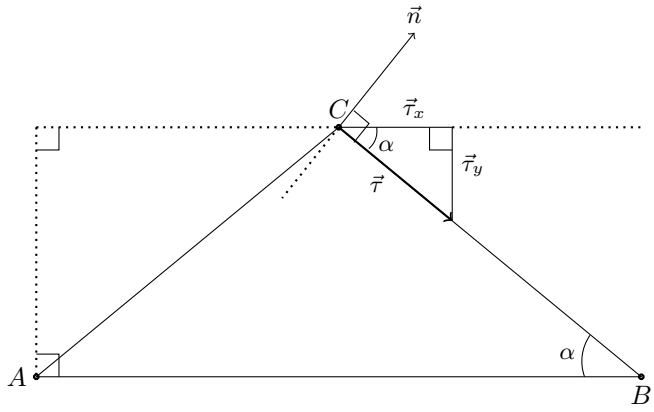


Figure 3.5: Closeup of subpart ABD

The subpart ABC of Figure 3.4 is presented in Figure 3.5. Virtual lines, represented by dotted lines in Figure 3.5, are inserted into the figure to create two parallel lines with another line crossing them both, called a transversal. The transversal makes it possible to express the angle α at the intersection point C, in the little virtual triangle created from the tangent vector $\vec{\tau}$ and the normal vector $\vec{n}$. The tangent and normal vectors are dependent on the angle ϕ, so the expression for α also becomes dependent on ϕ. An expression for α becomes

$$|\vec{\tau}_x| = \tau_x, \quad |\vec{\tau}_y| = \tau_y,$$

$$\alpha(\phi) = \arctan\left(\frac{-\tau_y(\phi)}{\tau_x(\phi)}\right),$$

$$\tau_x = \frac{\vec{\delta}'_x(\phi)}{||\vec{\delta}'(\phi)||}, \quad \tau_y = \frac{\vec{\delta}'_y(\phi)}{||\vec{\delta}'(\phi)||},$$

$$\alpha(\phi) = \arctan\left(\frac{-\vec{\delta}'_y(\phi)}{\vec{\delta}'_x(\phi)}\right),$$

with $\vec{\delta}'_x$ and $\vec{\delta}'_y$ given by (3.5b), resulting in

$$\alpha(\phi) = \arctan\left(\frac{\delta(\phi)\sin(\phi) - \delta'(\phi)\cos(\phi)}{\delta(\phi)\cos(\phi) + \delta'(\phi)\sin(\phi)}\right). \tag{3.6}$$

3.2.3 Finding $\vec{\tau}, \vec{n}, \vec{\kappa}$

.

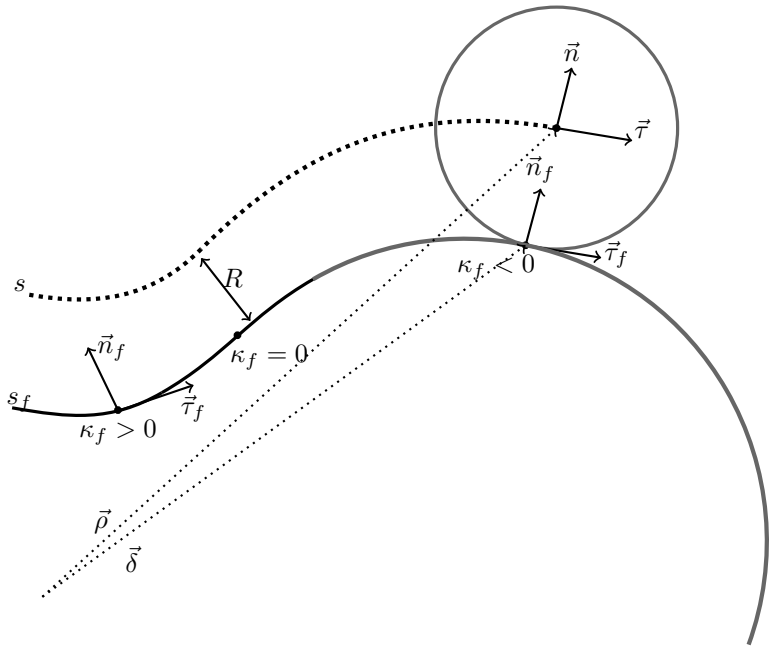


Figure 3.6: Closeup with curvature notifications included

Having defined expressions to represent $\vec{\delta}, \vec{\delta}', \vec{\delta}''$ and α makes it possible to find the vectors for the tangent, the normal and the curvature. The fact that the ball and frame has one point of contact results in identical tangent and normal vectors for the frame and ball, see Figure 3.6. By using the definition of (A.1) the tangent vector becomes

$$\vec{\tau} = \frac{d\vec{\rho}}{ds}, \quad \vec{\tau}_f = \frac{d\vec{\delta}}{ds_f}, \quad \vec{\tau} = \vec{\tau}_f,$$

$$\vec{\tau} = \vec{\tau}_f = \frac{\vec{\delta}'}{||\vec{\delta}'||}. \tag{3.7}$$

The normal vector is defined in (A.2) as

$$\vec{n} = \hat{k} \times \vec{\tau}, \quad \vec{n}_f = \hat{k} \times \vec{\tau}_f, \quad \vec{n} = \vec{n}_f,$$

but a different representation of $\vec{n}$ is used due to that representation's simplicity when differentiating the term. The terms $\vec{n}, \vec{n}', \vec{n}''$ are all used at a later stage in the thesis and a simplistic representation is therefore wanted. All three expressions can easily be expressed by the angle α. A closeup of the representation of α is shown in Figure 3.7, where the link between the normal vector and α can be studied.

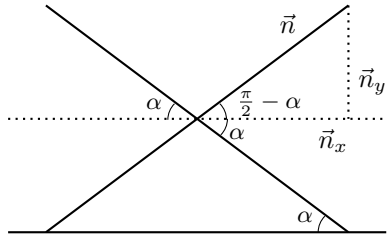


Figure 3.7: Closeup of α angle

$$|\vec{n}_x| = n_x, \quad |\vec{n}_y| = n_y, \quad |\vec{n}| = 1,$$

$$n_x = |\vec{n}| \cos(\frac{\pi}{2} - \alpha) = \cos(\frac{\pi}{2}) \cos(\alpha) + \sin(\frac{\pi}{2}) \sin(\alpha) = \sin(\alpha),$$

$$n_y = |\vec{n}| \sin(\frac{\pi}{2} - \alpha) = \sin(\frac{\pi}{2}) \cos(\alpha) + \cos(\frac{\pi}{2}) \sin(\alpha) = \cos(\alpha),$$

$$\begin{aligned} n_x(\phi) &= \sin\big(\alpha(\phi)\big), \\ n_x'(\phi) &= \cos\big(\alpha(\phi)\big)\alpha'(\phi), \\ n_x''(\phi) &= -\sin\big(\alpha(\phi)\big)\alpha'(\phi)^2 + \cos\big(\alpha(\phi)\big)\alpha''(\phi), \end{aligned}$$

$$
\begin{aligned}
n_y(\phi) &= \cos\big(\alpha(\phi)\big), \\
n_y'(\phi) &= -\sin\big(\alpha(\phi)\big)\alpha'(\phi), \\
n_y''(\phi) &= -\cos\big(\alpha(\phi)\big)\alpha'(\phi)^2 - \sin\big(\alpha(\phi)\big)\alpha''(\phi),
\end{aligned}
$$

$$
\vec{n} = \begin{bmatrix} \sin(\alpha) \\ \cos(\alpha) \\ 0 \end{bmatrix}, \tag{3.8a}
$$

$$
\vec{n}' = \begin{bmatrix} \cos(\alpha) \\ -\sin(\alpha) \\ 0 \end{bmatrix} \alpha', \tag{3.8b}
$$

$$
\vec{n}'' = \begin{bmatrix} -\sin(\alpha) \\ -\cos(\alpha) \\ 0 \end{bmatrix} \alpha'^2 + \begin{bmatrix} \cos(\alpha) \\ -\sin(\alpha) \\ 0 \end{bmatrix} \alpha''. \tag{3.8c}
$$

The process of deriving the curvature vector $\vec{\kappa}$ of the ball's path is a little more complex. The idea is to use the curvature vector of the frame $\vec{\kappa}_f$ and somehow find the offset curvature vector $\vec{\kappa}$ by using this achievable $\vec{\kappa}_f$. The curvature constant κ_f is signed, meaning that the curvature can change from positive to negative, and from negative to positive. When the tangent vector rotates counterclockwise the following applies: $\kappa_f > 0$, and when the tangent vector rotates clockwise: $\kappa_f < 0$. The point where the curve changes from concave to convex is called an inflection point, $\kappa_f = 0$. These features can be seen in Figure 3.6. The curvature describes the amount of change the tangent vectors go through along the curve, while the tangent vectors describes the change of the curve itself. The definition of curvature is given in Appendix A as

$$
\vec{\kappa} = \frac{d\vec{\tau}}{ds}, \quad \vec{\kappa_f} = \frac{d\vec{\tau}_f}{ds_f}, \quad \vec{\kappa} \neq \vec{\kappa}_f.
$$

The expression for curvature is taken from [14], where more information about the derivation of curvature can be explored. The curvature of the frame can be defined as

$$\kappa_f = \frac{||\vec{\delta}' \times \vec{\delta}''||}{||\vec{\delta}'||^3}.$$

The product of the curvature and the normal tangent will represent how the curve behaves and how it bends.

The challenge of finding the translation from κ_f into κ starts by looking at $\vec{\rho}$ in relativity to the arc length of the frame s_f. The next step is to use the definition of tangent vector and the Frenet formulas defined in (A.3),

$$\frac{d\vec{\rho}}{ds_f} = \frac{d\vec{\delta}}{ds_f} + R\frac{d\vec{n}_f}{ds_f} = \vec{\tau}_f - R\kappa_f\vec{\tau}_f = (1 - R\kappa_f)\vec{\tau_f}.$$

Thus the arc lengths relative to each other becomes

$$\frac{ds}{ds_f} = 1 - R\kappa_f,$$

which gives the following definition for $\vec{\kappa}$,

$$\vec{\kappa} = \frac{d^2\vec{\rho}}{ds^2} = \frac{d\vec{\tau}}{ds} = \frac{d\vec{\tau_f}}{ds} = \frac{ds_f}{ds}\frac{d\vec{\tau_f}}{ds_f} = \frac{\kappa_f\vec{n}_f}{1 - R\kappa_f} = \kappa\vec{n}.$$

The curvature of the offset curve is then

$$\vec{\kappa} = \kappa\vec{n}, \quad \kappa = \frac{\kappa_f}{1 - R\kappa_f}. \tag{3.9}$$

As long as the denominator of the offset curve is positive, meaning $R < \frac{1}{\kappa_f}$, the offset curve is smooth. If $R = \frac{1}{\kappa_f}$, a cusp appears. This cusp represents a singularity in the offset curve. This will also occur for $R > \frac{1}{\kappa_f}$.

3.2.4 Finding s' & s''

The parameters found in the previous subsection stay the same for both systems, with and without the assumption $\phi = \varphi$. The arc length s and its derivatives s', s'', however, do not stay the same for the two approaches. Derivation of s' and s'' will first be done for the system with no assumption, before turning to the system including the assumption.

The arc length s and its derivatives are redefined here for convenience,

$$s(\varphi) = \int_0^{\varphi} \left|\left| \frac{d\vec{\rho}}{d\varphi} \right|\right| d\varphi, \quad s' = \frac{ds(\varphi)}{d\varphi}, \quad s'' = \frac{d^2 s(\varphi)}{d\varphi^2}. \tag{3.10}$$

Without assumption $\phi \neq \varphi$

The center of the ball is now represented by $\vec{\rho}_{\varphi \neq \phi} = \vec{\delta} + R\vec{n}$ from (3.3b). A relationship between ϕ and φ needs to be found. The relationship chosen in this thesis is $\varphi = g(\phi)$, which will be explained and discussed in the next session. For now it is assumed that $g(\phi)$, $g'(\phi)$ and $g''(\phi)$ are feasible and valid. Using these expressions, together with the previously found (3.5) and (3.8), the parameters s' and s'' are derived,

$$s' = \left|\left| \frac{d\vec{\rho}}{d\phi}\frac{d\phi}{d\varphi} \right|\right| = \left|\left| \frac{d\vec{\rho}}{d\phi} \right|\right| \frac{d\phi}{d\varphi} = \left|\left| \frac{d\vec{\delta}}{d\phi} + R\frac{d\vec{n}}{d\phi} \right|\right| \left[\frac{d\varphi}{d\phi}\right]^{-1} = \left|\left|\vec{\delta}' + R\vec{n}'\right|\right| \frac{1}{g'(\phi)}. \tag{3.11}$$

Note that $\left|\frac{d\phi}{d\varphi}\right| > 0$ due to the definition of the two angles ϕ and φ, and it can therefore be extracted from the magnitude. The product rule is used when differentiating s' to achieve s''. The expression in (3.11) is defined as

$$p(\cdot) = s',$$

where p is a temporary function used for calculation purposes only. The same goes for the function h, which is defined as

$$h(\cdot) = \vec{\delta}' + R\vec{n}',$$

$$h'(\cdot) = \vec{\delta}'' + R\vec{n}''.$$

These definitions make the following calculations become more transparent. The function p is differentiated with respect to ϕ, which is used in the upcoming differentiation with respect to φ.

$$\frac{dp}{d\phi} = \frac{d}{d\phi}||h||\frac{1}{g'(\phi)} - ||h||\frac{g''(\phi)}{g'(\phi)^2},$$

where differentiating the magnitude of a vector is defined as

$$\frac{d}{d\phi}||h|| = \frac{h \cdot h'}{||h||}.$$

Combining all this information leads to the wanted expression,

$$\frac{dp}{d\varphi} = \frac{dp}{d\phi}\frac{d\phi}{d\varphi} = \left(\frac{d}{d\phi}||h||\frac{1}{g'(\phi)} - ||h||\frac{g''(\phi)}{g'(\phi)^2}\right)\left[\frac{d\varphi}{d\phi}\right]^{-1}.$$

Inserting the full expressions of h and h' gives the final expression,

$$\begin{aligned} s'' = \frac{dp}{d\varphi} &= \frac{h \cdot h'}{||h||}\frac{1}{g'(\phi)^2} - ||h||\frac{g''(\phi)}{g'(\phi)^3} \\ &= \frac{\left(\vec{\delta}' + R\vec{n}'\right) \cdot \left(\vec{\delta}'' + R\vec{n}''\right)}{||\vec{\delta}' + R\vec{n}'||}\frac{1}{g'(\phi)^2} - ||\vec{\delta}' + R\vec{n}'||\frac{g''(\phi)}{g'(\phi)^3}. \end{aligned} \tag{3.12}$$

With assumption $\phi = \varphi$

The center of the ball is now represented by $\vec{\rho}_{\varphi=\phi} = \vec{\delta} + R\vec{\xi}$ from (3.3a). A helpful feature of the assumption is that since $\phi = \varphi \Rightarrow \frac{d\phi}{d\varphi} = 1 \Rightarrow g'(\phi) = 1$ and $g''(\phi) = 0$. Also note that since $\phi = \varphi \Rightarrow \vec{n} = \vec{\xi}$. Inserting these features into (3.11) and (3.12) gives the expressions,

$$s' = \left|\left|\vec{\delta}' + R\vec{\xi}'\right|\right|, \tag{3.13}$$

$$s'' = \frac{\left(\vec{\delta}' + R\vec{\xi}'\right) \cdot \left(\vec{\delta}'' + R\vec{\xi}''\right)}{||\vec{\delta}' + R\vec{\xi}'||}. \tag{3.14}$$

3.2.5 Representation of ϕ

All the work done in the previous subsection is summed up in Table 3.2. The parameters are all dependent on ϕ, which is an angle without any form of representation at the moment. Finding a link between φ and ϕ is therefore crucial to solving the expressions in Table 3.2. The most straightforward approach would be to find $\phi = f(\varphi)$ since φ is one of the generalized coordinates, while ϕ is not. The search for this function has been pursued in [10], where a direct inverse relation from $\varphi = g(\phi) \rightarrow \phi = g(\varphi)^{-1} = f(\varphi)$ was attempted. The conclusion from [10] suggested that this relationship might not exist.

Table 3.2: Overview of expressions for parameters

	With assumption ($\phi = \varphi$)	Without assumption ($\phi \neq \varphi$)
$\vec{\boldsymbol{\rho}}(\boldsymbol{\phi}) =$	$\vec{\delta} + R\vec{\xi}$	$\vec{\delta} + R\vec{n}$
$\vec{\boldsymbol{\tau}}(\boldsymbol{\phi}) =$	$\dfrac{\vec{\delta}'}{\|\|\vec{\delta}'\|\|}$	$\dfrac{\vec{\delta}'}{\|\|\vec{\delta}'\|\|}$
$\vec{\boldsymbol{\kappa}}(\boldsymbol{\phi}) =$	$\dfrac{\|\|\vec{\delta}' \times \vec{\delta}''\|\|}{\|\|\vec{\delta}'\|\|^3 - R\|\|\vec{\delta}' \times \vec{\delta}''\|\|}\vec{n}$	$\dfrac{\|\|\vec{\delta}' \times \vec{\delta}''\|\|}{\|\|\vec{\delta}'\|\|^3 - R\|\|\vec{\delta}' \times \vec{\delta}''\|\|}\vec{n}$
$\boldsymbol{s'}(\boldsymbol{\phi}) =$	$\|\|\vec{\delta}' + R\vec{\xi}'\|\|$	$\|\|\vec{\delta}' + R\vec{n}'\|\|\dfrac{1}{g'}$
$\boldsymbol{s''}(\boldsymbol{\phi}) =$	$\dfrac{\left(\vec{\delta}' + R\vec{\xi}'\right) \cdot \left(\vec{\delta}'' + R\vec{\xi}''\right)}{\|\|\vec{\delta}' + R\vec{\xi}'\|\|}$	$\dfrac{\left(\vec{\delta}' + R\vec{n}'\right) \cdot \left(\vec{\delta}'' + R\vec{n}''\right)}{\|\|\vec{\delta}' + R\vec{n}'\|\|}\dfrac{1}{g'^2}$ $-\|\|\vec{\delta}' + R\vec{n}'\|\|\dfrac{g''}{g'^3}$

This thesis will explore a different expression for the relationship, and it will also avoid the necessity of finding a direct expression for the relationship. The fact that the relationship

$\phi = f(\varphi)$ is a property of curvature makes it very complex and incredible hard to solve, even with help from mathematical tools like Maple and MATLAB. The idea here is to find $\varphi = g(\phi)$ and use this expression to derive values for the parameters, as Table 3.2 shows. The need for the expression of $\phi = f(\varphi)$ therefore diminishes, and the new challenge becomes to find an expression for $\varphi = g(\phi)$ instead.

After finding $\varphi = g(\phi)$, a spline is fitted to the function using MATLAB. This spline is then a close-to-exact representation of $\phi = f(\varphi)$. This is a minor simplification which gives a very good representation of ϕ. The simplification is far less dramatic than the assumption of $\phi = \varphi$, and will have a next-to-nothing effect on the results. This new found representation of ϕ will be used for simulating purposes. The math of the parameters will still be accurate.

3.3 Finding $\varphi = g(\phi)$

The expression of g is an important discovery and will be used throughout the thesis. Recently, s' and s'' were defined as dependent on g, and it is therefore important to derive an expression for $\varphi = \varphi(\phi) = g(\phi)$. An analytical process will be used, along with visual aids from geometric figures to find the correct expression.

3.3.1 Derive an Expression for φ

Figure 3.8 is a tight closeup of the top-right corner of the ”Butterfly” robot, where the robot frame and the arc length describing the ball's path are displayed by the two dotted lines s_f and s. The center of the ball is located at (x',y'), which is a representation relative to the robot frame's body frame. The following method is dedicated to finding several ways to represent the point (x',y'), before the representations are manipulated to find an expression for angle φ. It is worth mentioning again that upwards, along the y'-axis, is considered zero degrees, while to the right, along the x'-axis, is considered 90 degrees. Angles φ and ϕ show this phenomenon in Figure 3.8, where they start from the positive

y'-axis and move towards the positive x'-axis to increase the angle.

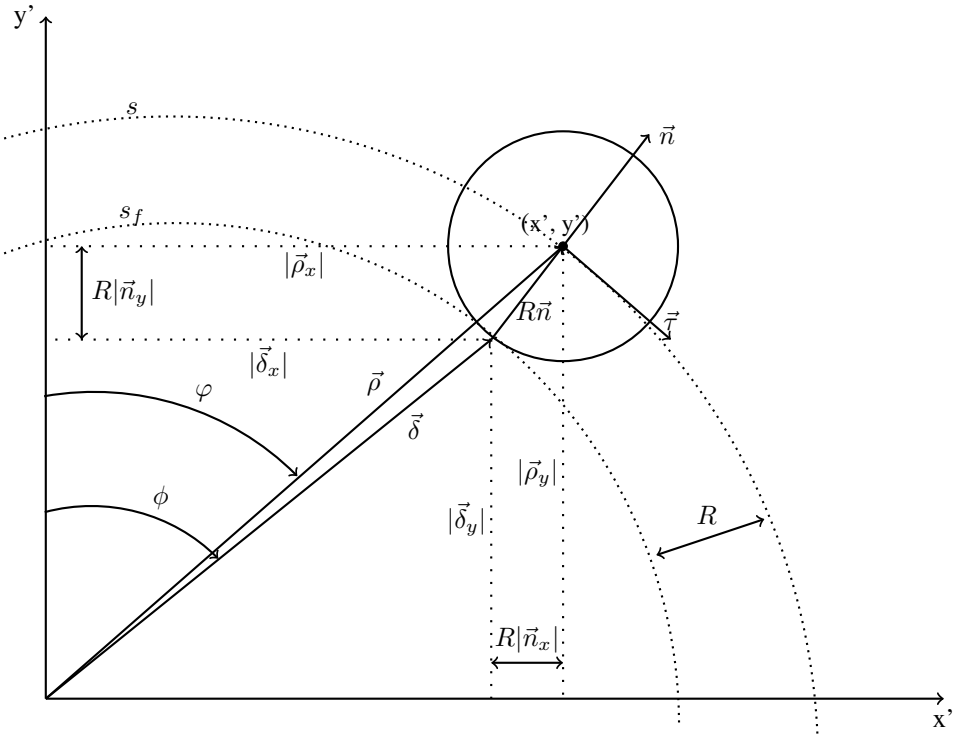


Figure 3.8: Closeup of the top-right corner of the butterfly: Finding φ

Using a mathematical approach, disregarding if an angle is in fact expressible or not, makes it possible to represent the vector $\vec{\rho}$ stretching from the origin to the center of the ball in two different ways. The first one (3.15a) is a composition of two vectors explained in (3.2), while the other one (3.15b) goes directly from the origin to the center of the ball.

$$\vec{\rho} = \vec{\delta}(\phi) + R\vec{n}(\phi), \tag{3.15a}$$

$$\vec{\rho} = \vec{\rho}(\varphi). \tag{3.15b}$$

The components describing the center of the ball (x',y') are derived by extracting the x and y value of $|\vec{\rho}|$. The equations in (3.16) used the expression in (3.15a), while the equations in (3.17) used the expression in (3.15b). Figure 3.8 clearly shows all the notations and what they represent in the following equations.

$$\text{x'} = |\vec{\delta}_x| + R|\vec{n}_x| = |\vec{\delta}|\sin(\phi) + R|\vec{n}_x|, \tag{3.16a}$$

$$\text{y'} = |\vec{\delta}_y| + R|\vec{n}_y| = |\vec{\delta}|\cos(\phi) + R|\vec{n}_y|. \tag{3.16b}$$

$$\text{x'} = |\vec{\rho}_x| = |\vec{\rho}|\sin(\varphi), \tag{3.17a}$$

$$\text{y'} = |\vec{\rho}_y| = |\vec{\rho}|\cos(\varphi). \tag{3.17b}$$

By looking at the two equations in (3.17) and dividing the first with the second, together with the action of isolating the angle φ, an expression for φ is found. Inserting the definitions of x' and y' from (3.16) into the newly created expression for φ, introduces an even better expression for $\varphi = \varphi(\vec{\delta}, \vec{n}, \phi)$, where $\vec{\delta} = \vec{\delta}(\phi)$ and $\vec{n} = \vec{n}\big(\alpha(\phi)\big)$,

$$\begin{aligned}
\varphi &= \arctan\left(\frac{\text{x'}}{\text{y'}}\right) \\
&= \arctan\left(\frac{|\vec{\delta}|\sin(\phi) + R|\vec{n}_x|}{|\vec{\delta}|\cos(\phi) + R|\vec{n}_y|}\right) \\
&= \arctan\left(\frac{|\vec{\delta}|\sin(\phi) + R\sin(\alpha)}{|\vec{\delta}|\cos(\phi) + R\cos(\alpha)}\right).
\end{aligned}$$

Using the fact that

$$\begin{aligned}
|\vec{\delta}(\phi)| &= \sqrt{\big(\delta(\phi)\cos(\phi)\big)^2 + \big(\delta(\phi)\sin(\phi)\big)^2} \\
&= \sqrt{\delta(\phi)^2\big(\cos^2(\phi) + \sin^2(\phi)\big)} = \delta(\phi),
\end{aligned}$$

the final expression becomes

$$\varphi = g(\phi) = \arctan\left(\frac{\delta\sin(\phi) + R\sin(\alpha)}{\delta\cos(\phi) + R\cos(\alpha)}\right). \tag{3.18}$$

3.3.2 Validation of the Expression

A system with a ball balancing on top of a frame will be implemented into MATLAB and simulations of the expression $\varphi = g(\phi)$ will be simulated for two different sizes of the ball, represented by R. The first simulation will be performed with a small-sized radius of the ball, while the second simulation performed will have a large-sized radius where the differences become clearer. In addition to this, all simulations will be done for three different shapes; **Circle** (simple shape), **Ellipse** (more advanced shape) and **Butterfly** (complicated shape). This is to ensure that the expression of $\varphi = g(\phi)$ is a general expression, which is applicable to all systems, not just the "Butterfly" robot. This goes hand in hand with the general dynamics that were derived earlier. Making all the work in this thesis general will make sure the work is applicable to other systems dealing with non-prehensile manipulation.

Before the simulations are performed, a theoretical analysis is done of the expression. It is always smart to have a certain idea of how the simulations should behave. Inserting the two instances of the ball (small and large) into the expression $g(\phi)$ leads to the following

$$R \to 0 : \varphi = \arctan\left(\frac{\delta\sin(\phi)}{\delta\cos(\phi)}\right) = \phi, \tag{3.19a}$$

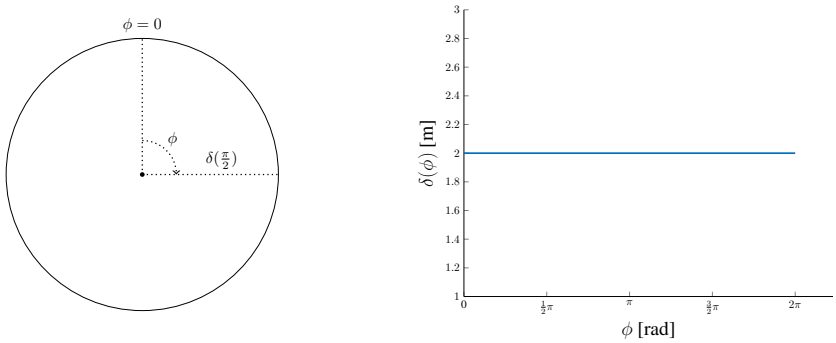
$$R \to \infty : \varphi = \arctan\left(\frac{R\sin(\alpha)}{R\cos(\alpha)}\right) = \alpha. \tag{3.19b}$$

When the ball is very small it makes perfect sense that the two angles become close to identical, as the center of the ball and the intersection point between the frame and ball become located very close to one another (previously showed in Figure 3.2). When the ball is very large, the vector from the origin of the frame to the center of the ball becomes perpendicular with the tangent vector of the center of the ball. This means that the angle φ becomes equivalent to the tangential angle α. Studying Figure 3.4 can give some visual

aids when trying to picture this geometric relationship. Now that certain expectations are set, it is time to look at the results from the simulations.

Circle

The circular shape is created by $\delta(\phi)$ and is shown in Figure 3.9a. The dotted lines are included to show where the zero-angle starts, and to show which way is defined as positive. There is a constant distance from the origin of the circle to the edge of the circle, seen in Figure 3.9b.



(a) Illustration of a circular shape

(b) Relationship between angle and distance to frame relative to center

Figure 3.9: Link between circular shape and $\delta(\phi)$

Assuming there is a ball being balanced on top of a circular frame, the relationships between the two angles of relevance become quite evident. The curvature of the circle is always positive, and the two angles become identical, as Figure 3.10 shows. Intuitively, this means that there will always be a linear relationship between the two angles, no matter what the radius of the ball is. Figure 3.11 shows the relationship for a ball with an effective radius of 0.01 meters (Figure 3.11a), as well as for a ball with an effective radius of 10 meters (Figure 3.11b). The expression $\varphi = g(\phi)$ resulted in the correct and expected behaviour, regardless of the size of the ball.

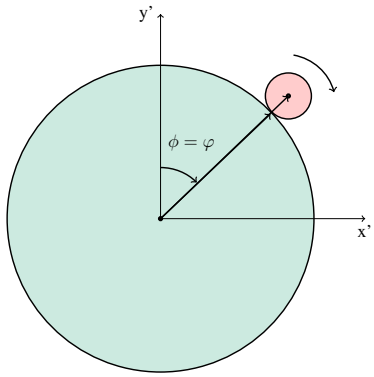
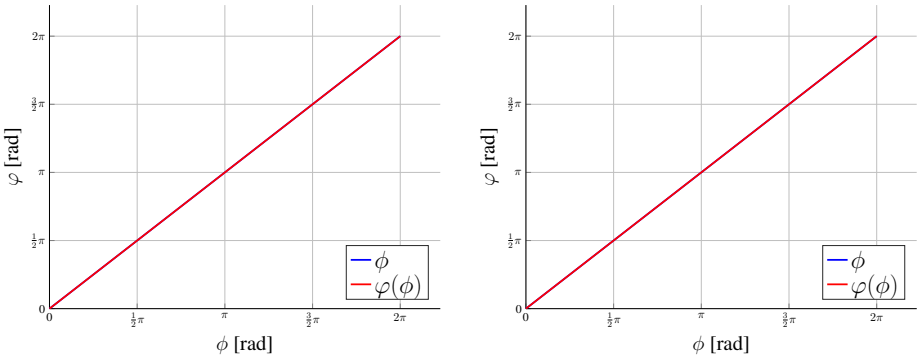


Figure 3.10: Circular frame balancing a ball on top



(a) R = 0.01 meter

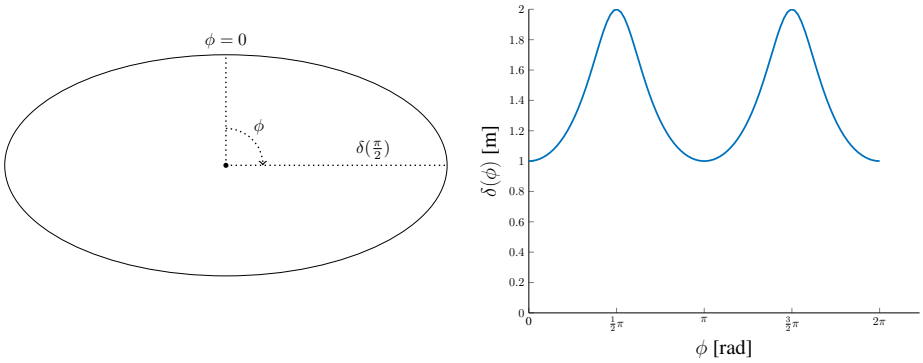
(b) R = 10 meter

Figure 3.11: Behaviour of φ for a full rotation around a circle

Ellipse

The next shape created by $\delta(\phi)$ is an elliptical shape showed in Figure 3.12a. The distance from the origin of the ellipse to the edge of the ellipse varies with respect to the angle ϕ, see Figure 3.12b. The transition from the smallest to the largest radius is very smooth, while the transition from the long side to the short side is quite sharp, as both the figures show. Once again, the dotted lines are included to show where the zero-angle starts, and

to show which direction is defined as positive.



(a) Illustration of an elliptical shape

(b) Relationship between angle and distance to frame relative to center

Figure 3.12: Link between elliptical shape and $\delta(\phi)$

Similar to the previous example of the circular frame, the elliptical shape always has a positive curvature. Dissimilar to the circle example, the ellipse does not result in the two angles ϕ and φ being identical, and they do not have a constant relationship. Figure 3.13 clearly illustrates this statement. There are only four instances around the periodic motion of the ellipse where the two angles are identical, $\phi = \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi$ rad. Expressed generally, regardless of how many rotations are completed, the expression becomes $\phi = \frac{\pi}{2}k,\ k \in \mathbb{Z}$. This results in a behaviour where $\phi \neq \varphi$ in an alternate positive and negative manner as the ball rolls around the frame, except for when the expression $\phi = \frac{\pi}{2}k,\ k \in \mathbb{Z}$ is true. Figure 3.14 illustrates this behaviour, with Figure 3.14a having far less dramatic differences than Figure 3.14b. This is naturally because a large ball results in a bigger gap between the two angles ϕ and φ.

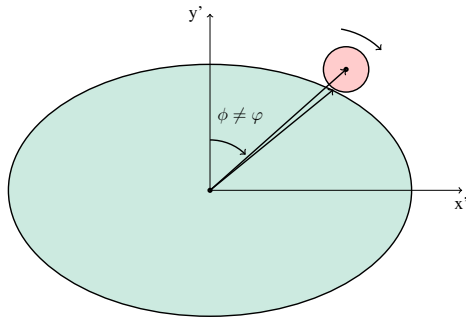
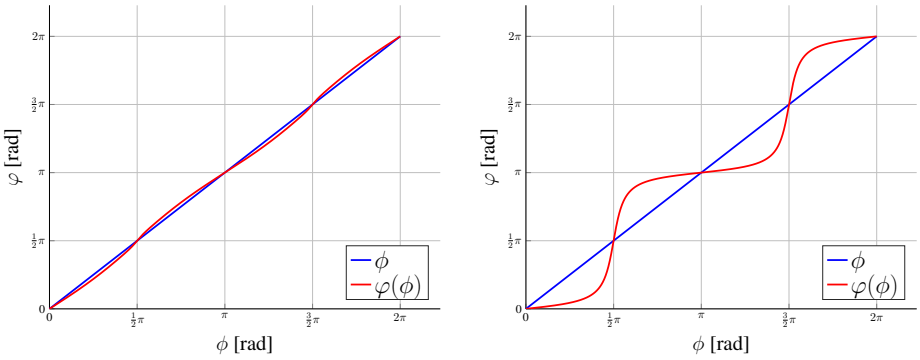


Figure 3.13: Elliptical frame balancing a ball on top



(a) R = 0.01 meter

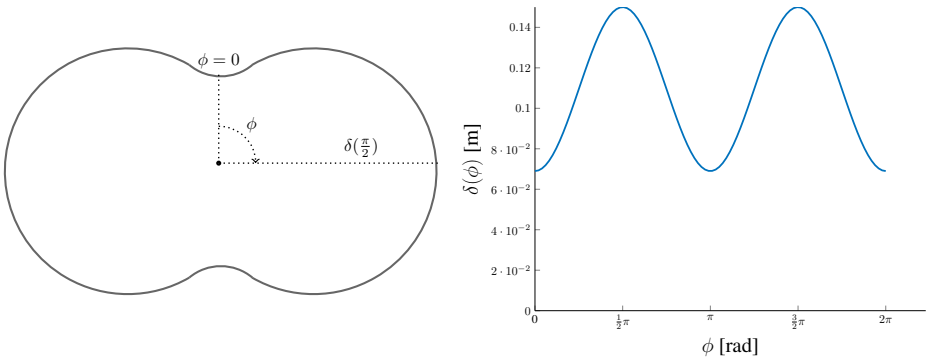
(b) R = 10 meter

Figure 3.14: Behaviour of φ for a full rotation around an ellipse

Butterfly

The last shape considered as $\delta(\phi)$ is the butterfly shape showed in Figure 3.15a. Just like the elliptic shape, the distance from the origin of the butterfly to the edge of the butterfly varies with respect to the angle ϕ, see Figure 3.15b. While the elliptic shape has a smooth transition on the short side and a sharp transition on the long side, the butterfly is the polar opposite, with a sharp transition on the short side and a smooth transition on the long side. This results in a longer duration of a large radius compared to the ellipse. The dotted lines are once again included to show where the zero-angle starts, and to show which direction

is defined as positive.



(a) Illustration of the butterfly shape

(b) Relationship between angle and distance to frame relative to center

Figure 3.15: Link between butterfly shape and $\delta(\phi)$

When it comes to the butterfly shape, the two angles ϕ and φ do not have a constant relationship, as Figure 3.16 shows. While the circle and the ellipse both had positive curvature throughout their shape, the butterfly shape has both positive and negative curvature. This is why the signed curvature had to be calculated in the previous section, as opposed to just the curvature. There are four instances where the two angles ϕ and φ are identical, and these can be represented with the same general expression as for the elliptical shape; $\phi = \frac{\pi}{4}k,\ k \in \mathbb{Z}$. The same simulations as for the two previous shapes are produced for the butterfly, see Figure 3.17. When R is small there should be a small difference between the two angles ϕ and φ, while a large R should produce a large difference, except for the four points $\phi = \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi$. $\varphi = g(\phi)$ fits the description perfectly, as Figure 3.17a and Figure 3.17b shows.

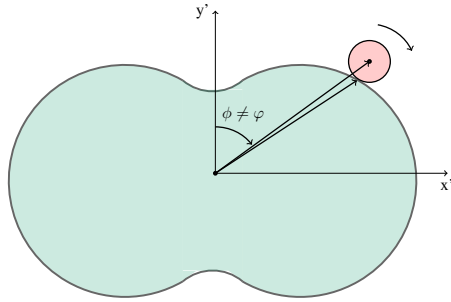
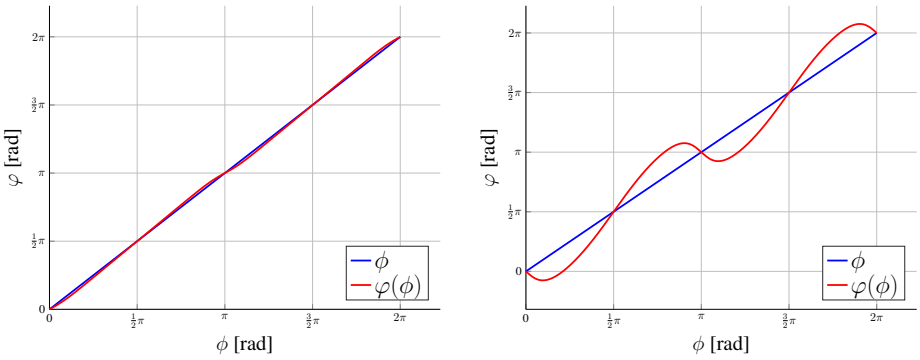


Figure 3.16: Butterfly frame balancing a ball on top



(a) R = 0.01 meter

(b) R = 10 meter

Figure 3.17: Behaviour of φ for a full rotation around the butterfly frame

The simulations in Figure 3.14 and Figure 3.17 clearly show that the linear relationship of $\phi = \varphi$ is not valid for all frame shapes. In fact, a perfect circle is the only shape where the assumption is valid. The difference between the angles is also proven to be substantial (especially if R is large). This leads to the following conclusion; *the assumption of $\phi = \varphi$ is not valid and should not be applied to the system*. The assumption, and the corresponding mathematics, will therefore not be used. Instead, the expression $\varphi = g(\phi)$ will be used in the rest of the thesis. The expression is repeated here for convenience:

$$\varphi = g(\phi) = \arctan\left(\frac{\delta \sin(\phi) + R\sin(\alpha)}{\delta \cos(\phi) + R\cos(\alpha)}\right). \tag{3.20}$$

Chapter 4

Motion Planning

With the dynamics and its parameters derived, the next step is to find feasible continuous rotations for the system. There has to be determined synchronization functions between the generalized coordinates, such that desired trajectories can be analytically detected [1]. The concept is explored in this chapter.

4.1 Virtual Holonomic Constraints

Virtual Holonomic Constraints (VHCs) is as the name suggests a virtual constraint on the system. This means that the said constraint does not physically exist, but it can be imposed onto the system to create specific motions or trajectories. Before the search for feasible trajectories begin, an analysis of VHCs, and why it is needed in this particular case, is completed.

4.1.1 Underactuated System

The "Butterfly" robot's system (2.21) is descried by the two independent generalized coordinates $q = [\theta(t), \varphi(t)]^T$. A general view on the system will give $q \in \mathbb{R}^n$ as the vector of generalized coordinates and $u \in \mathbb{R}^m$ as the external forces/toques acting on the system. When $m < n$ the robot is said to be underactuated with a degree of $n - m$. Since there

are more generalized coordinates ($n = 2$) than there are control inputs ($m = 1$), this paper has a underactuated system with degree $n - m = 1$. It can also be physically determined by comparing the number of objects needing to be controlled (frame and ball) versus the number of actuators available (rotation of frame). Both methods result in the need for a constraint to analytically solve the control problem. For a detailed discussion on control problems in underactuated systems, see [7], while an investigation into virtual holonomic constraints for Euler-Lagrange systems with n degrees of freedom and $n - 1$ controls can be found in [15].

The underactuated system needs a VHC to be able to achieve wanted control over the systems behaviour. Since the ball is not physically attached to the frame, and there is no way to directly actuate the ball, the task of controlling the ball becomes complicated. The responsibility of controlling the ball is placed on the frame since it can receive actuation. The challenge is to control the movement of the frame in such a way so that the ball is acting as desired.

4.1.2 VHCs General Form

By imposing a virtual holonomic constraint onto the system, with the job of manipulating θ, the system's behaviour can be decided by φ alone. A VHC on the following form will give a desired relationship between the generalized coordinates,

$$\theta = \Theta(\varphi), \tag{4.1a}$$

$$\dot{\theta} = \Theta'(\varphi)\dot{\varphi}, \tag{4.1b}$$

$$\ddot{\theta} = \Theta''(\varphi)\dot{\varphi}^2 + \Theta'(\varphi)\ddot{\varphi}, \tag{4.1c}$$

with $\Theta'(\varphi) = \frac{d\Theta}{d\varphi}$ and $\Theta''(\varphi) = \frac{d^2\Theta}{d\varphi^2}$. Inserting (4.1) into the generalized coordinates

$q = [\theta, \varphi]^T$ gives the following new generalized coordinates,

$$q = \begin{bmatrix} \Theta(\varphi) \\ \varphi \end{bmatrix}, \tag{4.2a}$$

$$\dot{q} = \begin{bmatrix} \Theta'(\varphi) \\ 1 \end{bmatrix} \dot{\varphi}, \tag{4.2b}$$

$$\ddot{q} = \begin{bmatrix} \Theta''(\varphi) \\ 0 \end{bmatrix} \dot{\varphi}^2 + \begin{bmatrix} \Theta'(\varphi) \\ 1 \end{bmatrix} \ddot{\varphi}. \tag{4.2c}$$

The system dynamics are re-presented here for convenience, as they will be useful to look at for the upcoming calculations,

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = B(q)u. \tag{4.3}$$

To be able to impose the VHCs onto the system, a control input is needed. This will of course be applied through the actuation of the frame, which is represented by the first equation in (4.3). This can be written as

$$\begin{aligned} u &= [M_{11} \;\; M_{12}] \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + [C_{11} \;\; C_{12}] \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + G_1 \\ &= [M_{11} \;\; M_{12}] \begin{bmatrix} \Theta''\dot{\varphi}^2 + \Theta'\ddot{\varphi} \\ \ddot{\varphi} \end{bmatrix} + [C_{11} \;\; C_{12}] \begin{bmatrix} \Theta'\dot{\varphi} \\ \dot{\varphi} \end{bmatrix} + G_1. \end{aligned} \tag{4.4}$$

The obtained control variable u in (4.4) is the only form of control there is over the system's subsequent motion. [1] dives deeper into the mentioned u and provides a more generic form of the control input, in addition to proofs of theorems used in the derivation.

Before deciding what the function $\theta = \Theta(\varphi)$ should be to produce the desired behaviour, the reduced dynamics are derived. These are essential in the upcoming process of searching for feasible and desired motions.

4.1.3 Reduced Dynamics

The reduced dynamics is a general form of representing an underactuated Euler-Lagrange system with a virtual constraint imposed on it. To further simplify the derivation of the reduced dynamics, another set of variables are introduced. These are defined as

$$\Phi(\varphi) = \begin{bmatrix} \Theta(\varphi) \\ \varphi \end{bmatrix}, \tag{4.5a}$$

$$\Phi'(\varphi) = \begin{bmatrix} \Theta'(\varphi) \\ 1 \end{bmatrix}, \tag{4.5b}$$

$$\Phi''(\varphi) = \begin{bmatrix} \Theta''(\varphi) \\ 0 \end{bmatrix}. \tag{4.5c}$$

Just as in Subsection 4.1.1, a general approach is taken when deriving the reduced dynamics. The Euler-Langrange system of (4.3) are considered to have generalized coordinates $q \in \mathbb{R}^n$, actuator input $u \in \mathbb{R}^m$ and $m < n$. Once again, the system is assumed to have one less actuator than degree of freedom, making it relatable to the example of the "Butterfly" robot with underactuation of degree $n - m = 1$.

Assuming there exists a trajectory $q_\star(t)$ parametrized by some scalar variable $\varphi \in \mathbb{R}$ with the relation,

$$q_\star(t) = \Phi(\varphi_\star) = [\phi_1(\varphi_\star), ..., \phi_n(\varphi_\star)]^T, \tag{4.6}$$

for a $\mathcal{C}^2$-smooth vector function $\Phi(\varphi)$. This results in the given positions, velocities and accelerations,

$$q = \Phi(\varphi), \tag{4.7a}$$

$$\dot{q} = \Phi'(\varphi)\dot{\varphi}, \tag{4.7b}$$

$$\ddot{q} = \Phi'(\varphi)\ddot{\varphi} + \Phi''(\varphi)\dot{\varphi}^2. \tag{4.7c}$$

To derive the reduced dynamics of the system, both sides of (4.3) are multiplied by a left-annihilator of $B(q)$. The annihilator $B^{\perp}(q)$ is created to neutralize the coupling matrix B, and therefore has the rank of $n - m$ with specific values such that $B^{\perp}(q)B(q)u = 0$ for all $q \in \mathbb{R}^n$. The model then becomes

$$B^{\perp}(q)M(q)\ddot{q} + B^{\perp}(q)C(q,\dot{q})\dot{q} + B^{\perp}(q)G(q) = B^{\perp}(q)B(q)u = 0. \tag{4.8}$$

Inserting (4.7) into (4.8) and rearranging the order of the variables leads to the reduced dynamics, famously known as the alpha-beta-gamma equation in the robotics community at NTNU,

$$\alpha(\varphi)\ddot{\varphi} + \beta(\varphi)\dot{\varphi}^2 + \gamma(\varphi) = 0. \tag{4.9}$$

This second order equation holds for all solutions which have the form and demands of (4.6), and the elements used in the equation are defined by

$$\alpha(\varphi) = B^{\perp}(\Phi)M(\Phi)\Phi', \tag{4.10}$$

$$\beta(\varphi) = B^{\perp}\Big[C(\Phi,\Phi')\Phi' + M(\Phi)\Phi''\Big], \tag{4.11}$$

$$\gamma(\varphi) = B^{\perp}G(\Phi), \tag{4.12}$$

where $\Phi = \Phi(\varphi)$, as defined in (4.5). The expressions take the form of (4.13) when inserting the previously derived parameters of the "Butterfly" robot. Note that α, β, and γ are dependent on the VHC.

$$\alpha(\varphi) = s'\left(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\right)\Theta' + s'^2\left(m + \frac{J_b}{R^2}\right), \tag{4.13a}$$

$$\beta(\varphi) = s'\left(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\right)\Theta'' - ms'\vec{\tau}\cdot\vec{\rho}\Theta'^2 + s's''\left(m + \frac{J_b}{R^2}\right), \tag{4.13b}$$

$$\gamma(\varphi) = ms'\vec{g}\cdot(\Pi\vec{\tau}). \tag{4.13c}$$

4.2 Finding Trajectories

Finding predetermined trajectories for the ball and frame is essential when trying to get the system to work, and when trying to get the system to behave in a specific manner. The desired motion and the following derivation of feasible trajectories is a complicated matter, which will be carefully explained. Simple examples are used to explain certain concepts throughout this section.

4.2.1 Desired Motion

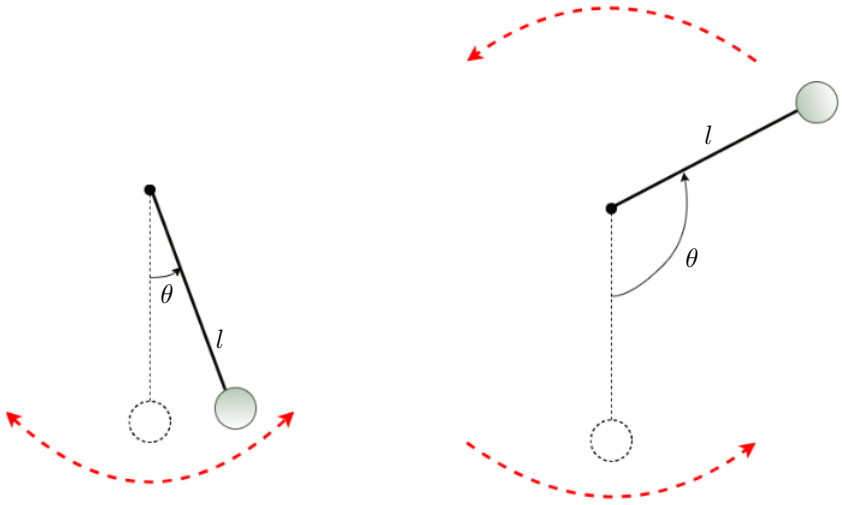
The VHC is an important part of planning motions and to search for feasible trajectories. In theory a VHC can be anything, but the desire for a specific behaviour restricts the possibilities. Wanting a specific behaviour from the system forces the need for a certain motion, which can only be achieved by applying the correct VHC. There are therefore many factors to consider when designing a VHC, represented as conditions or/and restrictions. The choice of the initial conditions are also essential when shaping the desired trajectory. The system's motion can be substantially changed depending on the initial conditions. The simple example of the pendulum is introduced to clearly illustrate this statement. The pendulum system is given as

$$\ddot{\theta} + \frac{g}{l}\sin\theta = 0.$$

The system is imagined without any form of friction, giving it only two possible motions. The first motion comes from a soft initial push where the pendulum oscillates back and fourth, illustrated in Figure 4.1a. The second motion comes from a powerful push where the pendulum swings all the way around in a one-directional periodic motion, illustrated in Figure 4.1b.

The pendulum's phase portrait is shown in Figure 4.2, where both the motions are clearly depicted. The center points of $\bar{\varphi} = 0 + 2n\pi$ with $n = 1, 2, 3, ..., k$ for $k \in \mathbb{Z}$, are represented by a periodic position and a periodic velocity for the angle θ, which fits perfectly with the behaviour of Figure 4.1a. The saddle points of $\bar{\varphi} = 0 + m\pi$ with

$m = 1, 3, 5, ..., 2k + 1$ for $k \in \mathbb{Z}$, are represented by an increasing position and a periodic velocity for the angle θ, which fits perfectly with the behaviour of Figure 4.1b. The transition between the two solutions is called the separatrix and it separates the two set of solutions.



(a) Slight push → Oscillating pendulum (b) Powerful push → Rotating pendulum

Figure 4.1: The pendulum's different behaviour dependent on the initial physical push

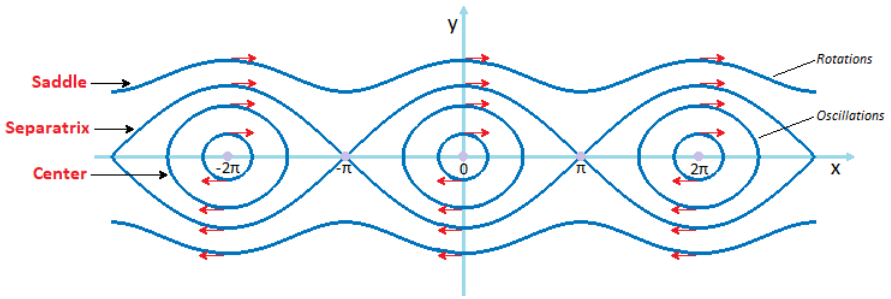


Figure 4.2: Phase portrait illustrating system behaviour of pendulum

The initial conditions are clearly very important for the outcome of the pendulum's motion. The same goes for the motion of the ball rolling on the butterfly frame. The "Butterfly" robot is naturally way more complicated than the pendulum, as the ball is rolling on a moving surface, constantly changing the ball's motion. The first thing that needs to be done is to decide on a desired motion. Should the ball oscillate back and fourth in the valley of the figure eight frame, should the ball rotate around the frame in a one-directional motion or should the ball be balanced on the side-plateau of the figure-eight shape? The possibilities are endless.

In this thesis, the chosen desired motion is a continuous one-directional rotation of the ball around the frame. It should be a periodic motion for the ball, such that after one full rotation around the frame, the ball is back to where it started. This means that the angular position of the ball φ is continuously increasing as the ball completes laps around the frame, while the angular velocity of the ball is periodic and has the same sign for every round. This behaviour matches that of a saddle, see Figure 4.2. The same behaviour is required of the butterfly frame itself and its rotation θ. After one full rotation, the position of the frame should have increased with 2π while the velocity should have been reset to the same value it started with. However, the periodic motions for θ and φ are not actually given by a full rotation of 2π rad. Since the butterfly shape has axis symmetry, the periodic motion is only for half a rotation of π rad. This means that the representation of the location of the butterfly frame relative to the inertial frame becomes

$$\begin{aligned} \theta(t + kT) &= \theta(t) + \pi k, \\ \dot{\theta}(t + kT) &= \dot{\theta}(t), \end{aligned} \tag{4.14}$$

while the location of the ball's center relative to the butterfly frame becomes

$$\begin{aligned} \varphi(t + kT) &= \varphi(t) + \pi k, \\ \dot{\varphi}(t + kT) &= \dot{\varphi}(t), \end{aligned} \tag{4.15}$$

with $T > 0$ as the period and $k \in \mathbb{Z}$.

The expressions in (4.14) and (4.15) imposes some conditions on $\Theta(\varphi)$, $\Theta'(\varphi)$ and $\Theta''(\varphi)$ because of the relationship established in (4.1a). The virtual constraint $\Theta(\varphi)$ should be twice differentiable and continuously increasing in a periodic fashion of π rad, while $\Theta'(\varphi)$ and $\Theta''(\varphi)$ should be periodic with π rad.

4.2.2 Guide to Valid Solution

Now that the desired motion has been picked, it is time to look at how a solution can be found. The following process is a step-by-step guide on how to find feasible solutions for the "Butterfly" robot, in addition to putting certain conditions on the VHC. The guide will show a general approach to finding a solution, meaning that the choice of a VHC will be unspecified. A suitable VHC will later be derived with this guide as its ground base. The guide contains four steps, and they are presented as follows,

1) Check if system has asymptote.
2) Find equilibrium points.
3) Determine type of equilibrium points.
4) Check if solution is bounded.

1 Check if system has asymptote

System is re-defined for convenience

$$\alpha(\varphi)\ddot{\varphi} + \beta(\varphi)\dot{\varphi}^2 + \gamma(\varphi) = 0. \tag{4.16}$$

The system is rearranged to evaluate the angular acceleration of the ball,

$$\ddot{\varphi} = -\frac{\gamma(\varphi)}{\alpha(\varphi)} - \frac{\beta(\varphi)}{\alpha(\varphi)}\dot{\varphi}^2, \tag{4.17}$$

where it can be seen that $\alpha(\varphi) = 0 \rightarrow \ddot{\varphi} = \pm\infty \rightarrow \dot{\varphi} = \pm\infty \rightarrow \varphi = \pm\infty$. Having this asymptote in the solution is unacceptable and must be avoided. A condition is placed upon the VHC to prevent the solution from being invalid due to an asymptote. The condition is

derived from the previously found representation of $\alpha(\varphi)$ in (4.13a),

$$\alpha(\varphi) = s'\left(m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}\right)\Theta'(\varphi) + s'^2\left(m + \frac{J_b}{R^2}\right).$$

This leads to the condition

$$\Theta'(\varphi) \neq \frac{-s'\left(m + \frac{J_b}{R^2}\right)}{m\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) - \frac{J_b}{R}}. \tag{4.18}$$

To be absolutely certain that a specific solution is valid, an evaluation of both the terms in (4.17) is reasonable. The terms are expressed as

$$\frac{\gamma(\varphi_a)}{\alpha(\varphi_a)} = \pm\infty, \quad \frac{\beta(\varphi_a)}{\alpha(\varphi_a)} = \pm\infty, \tag{4.19}$$

with φ_a as a point leading to an asymptote. If one of the expressions in (4.19) exists in the solution, the system is not valid and the trajectory is not acceptable. The safest thing is to just create a system that never has an asymptote and where $\alpha(\varphi)$, $\beta(\varphi)$ and $\gamma(\varphi)$ are finite for all values of φ.

2 Find equilibrium points

$$\text{Equilibrium point: } \dot{\varphi} = 0 \rightarrow \ddot{\varphi} = 0.$$

Inserting this into (4.17) gives the following

$$0 = \frac{-\gamma(\varphi)}{\alpha(\varphi)}, \quad \alpha(\varphi) \neq 0. \tag{4.20}$$

The expression of $\gamma(\varphi)$, defined in (4.13c) and reprinted below, needs to be equal to zero. Studying $\gamma(\varphi)$ reveals that the only way the expression can be equal to zero is when $\vec{g}\cdot\Big(\Pi\big(\Theta(\bar{\varphi})\big)\vec{\tau}(\bar{\varphi})\Big) = 0$, since $s' \neq 0$,

$$\gamma(\varphi) = ms'\vec{g}\cdot(\Pi\vec{\tau}).$$

The choice of the VHC, aka $\Theta(\varphi)$, is crucial to accomplishing this.

$$\begin{aligned}\gamma(\bar{\varphi}) &= ms'(\bar{\varphi})\underbrace{\vec{g}\cdot\big(\Pi\big(\Theta(\bar{\varphi})\big)\vec{\tau}(\bar{\varphi})\big)}_{=0} \\ &= ms'(\bar{\varphi})g\underbrace{\Big(\tau_x(\bar{\varphi})\sin\big(\Theta(\bar{\varphi})\big)+\tau_y(\bar{\varphi})\cos\big(\Theta(\bar{\varphi})\big)\Big)}_{=0}=0.\end{aligned} \tag{4.21}$$

Note that since $\vec{g} = [0, g, 0]^T$ the dot product of the two vectors in (4.21) is the second element of the vector $\Pi\vec{\tau}$ multiplied with g.

There are certain requirements set upon $\Theta(\varphi)$ to be able to satisfy (4.21). The expressions in (4.22) presents these requirements,

$$\gamma\Big(0+n\pi\Big) = ms'g\Big(\underbrace{\tau_x(0+n\pi)}_{\neq 0}\underbrace{\sin\big(\Theta(0+n\pi)\big)}_{=0}+\underbrace{\tau_y(0+n\pi)}_{=0}\underbrace{\cos\big(\Theta(0+n\pi)\big)}_{\neq 0}\Big), \tag{4.22a}$$

$$\gamma\Big(0+\frac{m\pi}{2}\Big) = ms'g\Big(\underbrace{\tau_x(0+\frac{m\pi}{2})}_{=0}\underbrace{\sin\big(\Theta(0+\frac{m\pi}{2})\big)}_{\neq 0}+\underbrace{\tau_y(0+\frac{m\pi}{2})}_{\neq 0}\underbrace{\cos\big(\Theta(0+\frac{m\pi}{2})\big)}_{=0}\Big), \tag{4.22b}$$

where $n = 1, 2, 3, ..., k$ and $m = 1, 3, 5, ..., 2k + 1$ with $k \in \mathbb{Z}$.

This can be summarized as the following condition,

$$\gamma(\bar{\varphi}) = 0, \quad \bar{\varphi} = 0 + \frac{\pi}{2}k, \quad k \in \mathbb{Z}$$

It is now shown how $\{\bar{\varphi}\}$ is separate points. The next step is to classify these points.

3 Determine type of equilibrium points

$$\begin{aligned}x_1 &= \varphi, \quad \dot{x}_1 = \dot{\varphi}, \\ x_2 &= \dot{\varphi}, \quad \dot{x}_2 = \ddot{\varphi},\end{aligned}$$

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{-\beta(x_1)x_2^2 - \gamma(x_1)}{\alpha(x_1)} \end{bmatrix},$$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}_{x_1 = \bar{\varphi}} = \begin{bmatrix} 0 & 1 \\ -\frac{\gamma'(\bar{\varphi})}{\alpha(\bar{\varphi})} & 0 \end{bmatrix},$$

$$|\lambda I - A| = \lambda^2 + \frac{\gamma'(\bar{\varphi})}{\alpha(\bar{\varphi})} = 0,$$

$$\lambda = \pm\sqrt{-\frac{\gamma'(\bar{\varphi})}{\alpha(\bar{\varphi})}},$$

- $\frac{\gamma'(\bar{\varphi})}{\alpha(\bar{\varphi})} > 0 \Rightarrow \lambda = \pm ia \Rightarrow$ center point.

- $\frac{\gamma'(\bar{\varphi})}{\alpha(\bar{\varphi})} < 0 \Rightarrow \lambda = \pm a \Rightarrow$ saddle point.

To assign the separate points $\bar{\varphi}$ to either of the two classification points, the sign of $\gamma'(\bar{\varphi})$ and $\alpha(\bar{\varphi})$ needs to be examined. By choosing a VHC so that $\alpha(\bar{\varphi}) > 0$, $\forall\, \bar{\varphi} = 0 + \frac{\pi}{2}k,\ k \in \mathbb{Z}$, the sign of $\gamma'(\bar{\varphi})$ determines if the point is a center or a saddle. The condition of (4.18) is extended to

$$\Theta'(\bar{\varphi}) > \frac{-s'\big(m + \frac{J_b}{R^2}\big)}{m\hat{k} \cdot (\vec{\rho} \times \vec{\tau}) - \frac{J_b}{R}}. \tag{4.23}$$

The expression $\gamma'(\varphi)$ needs to be derived before the expression's sign can be evaluated.

$$\frac{d}{d\varphi}\gamma(\varphi) = ms''(\varphi)\Big[\vec{g} \cdot \Big(\Pi\big(\Theta(\varphi)\big)\vec{\tau}(\varphi)\Big)\Big] + ms' \underbrace{\frac{d}{d\varphi}\Big[\vec{g} \cdot \Big(\Pi\big(\Theta(\varphi)\big)\vec{\tau}(\varphi)\Big)\Big]}_{\text{sub expr 1}},$$

$$\underbrace{\frac{d}{d\varphi}\Big[\vec{g} \cdot \Big(\Pi\big(\Theta(\varphi)\big)\vec{\tau}(\varphi)\Big)\Big]}_{\text{sub expr 1}} = \underbrace{\frac{d}{d\varphi}\vec{g}}_{=0} \cdot \Big(\Pi\big(\Theta(\varphi)\big)\vec{\tau}(\varphi)\Big) + \vec{g} \cdot \underbrace{\frac{d}{d\varphi}\Big(\Pi\big(\Theta(\varphi)\big)\vec{\tau}(\varphi)\Big)}_{\text{sub expr 2}},$$

$$\underbrace{\frac{d}{d\varphi}\Big(\Pi\big(\Theta(\varphi)\big)\vec{\tau}(\varphi)\Big)}_{\text{sub expr 2}} = \Pi'\big(\Theta(\varphi)\big)\Theta'(\varphi)\vec{\tau}(\varphi) + \big(\Theta(\varphi)\big)\underbrace{\frac{d}{d\varphi}\vec{\tau}(\varphi)}_{\text{sub expr 3}},$$

$$\underbrace{\frac{d}{d\varphi}\vec{\tau}(\varphi)}_{\text{sub expr 3}} = \frac{d}{d\varphi}\left(\frac{\vec{\delta}'(\varphi)}{||\vec{\delta}'(\varphi)||}\right) = \frac{\vec{\delta}''(\varphi)}{||\vec{\delta}'(\varphi)||} - \frac{\vec{\delta}'(\varphi)\big(\vec{\delta}'(\varphi) \cdot \vec{\delta}''(\varphi)\big)}{||\vec{\delta}'(\varphi)||^3}.$$

Putting all of this together leads to the expression

$$\gamma'(\varphi) = ms''\Big[\vec{g}\cdot(\Pi\vec{\tau})\Big] + ms'\Big[\vec{g}\cdot\Big(\Pi'\Theta'\vec{\tau} + \Pi\Big(\frac{\vec{\delta}''}{||\vec{\delta}'||} - \frac{\vec{\delta}'(\vec{\delta}'\cdot\vec{\delta}'')}{||\vec{\delta}'||^3}\Big)\Big)\Big].$$

The next step is to study the expression at the equilibrium points $\bar{\varphi}$. It is already derived that $\vec{g}\cdot(\Pi\vec{\tau}) = 0,\ \forall\ \bar{\varphi}$, which gives the following,

$$\gamma'(\bar{\varphi}) = ms'\Big[\vec{g}\cdot\Big(\Pi'\Theta'\vec{\tau} + \Pi\Big(\frac{\vec{\delta}''}{||\vec{\delta}'||} - \frac{\vec{\delta}'(\vec{\delta}'\cdot\vec{\delta}'')}{||\vec{\delta}'||^3}\Big)\Big)\Big].$$

Every individual part of the expression needs to be studied to figure out the combined output sign. The dot product $(\vec{\delta}'\cdot\vec{\delta}'') = (\vec{\delta}'_x\vec{\delta}''_x + \vec{\delta}'_y\vec{\delta}''_y)$ can be massively simplified. This is because $\vec{\delta}'_y = \vec{\delta}''_x = 0\ \forall\ \bar{\varphi} = 0 + n\pi,\ n = 1,2,3,...,k$ and $\vec{\delta}'_x = \vec{\delta}''_y = 0\ \forall\ \bar{\varphi} = 0 + \frac{m\pi}{2},\ m = 1,3,5,...,2k+1$ for $k \in \mathbb{Z}$. The expression can then be written as

$$\begin{aligned}\gamma'(\bar{\varphi}) &= ms'\Big[\vec{g}\cdot\Big(\Pi'\vec{\tau}\Theta' + \Pi\frac{\vec{\delta}''}{||\vec{\delta}'||}\Big)\Big] \\ &= \underbrace{ms'g\frac{1}{||\vec{\delta}'||}}_{>0}\underbrace{\Big(\big(\vec{\delta}'_x\cos(\Theta) - \vec{\delta}'_y\sin(\Theta)\big)\Theta' + \vec{\delta}''_x\sin(\Theta) + \vec{\delta}''_y\cos(\Theta)\Big)}_{=H(\varphi)}.\end{aligned}$$

The sign of $H(\varphi)$ will determine the sign of $\gamma'(\varphi)$.

$$H\Big(0+n\pi\Big) = \Theta'\Big(0+n\pi\Big)(a-b) + 5b - a > 0, \tag{4.24a}$$

$$H\Big(0+\frac{m\pi}{2}\Big) = \Theta'\Big(0+\frac{m\pi}{2}\Big)(a+b) - 5b - a < 0, \tag{4.24b}$$

where a and b are constants from the definition of the butterfly frame $\delta = a - b\cos(2\phi)$. This leads to the following conditions: $\Theta'\Big(0+n\pi\Big) > \frac{a-5b}{a-b}$ and $\Theta'\Big(0+\frac{m\pi}{2}\Big) < \frac{a+5b}{a+b}$. This can be summarized as the following

$$\begin{aligned}\gamma'(\bar{\varphi}) > 0 \Rightarrow&\ \text{center point}\ \Rightarrow \bar{\varphi} = 0 + n\pi, \\ \gamma'(\bar{\varphi}) < 0 \Rightarrow&\ \text{saddle point}\ \Rightarrow \bar{\varphi} = 0 + \frac{m\pi}{2},\end{aligned}$$

where $n = 1,2,3,...,k$ and $m = 1,3,5,...,2k+1$ with $k \in \mathbb{Z}$.

4 Check if solution is bounded

$q(t) = q(t, q_0, \dot{q}_0)$ is used to find a trajectory, where q is the general coordinate describing the system. By using this solution, together with the law of conservation of energy, it is possible to figure out if the system is bounded. The total energy in a system $E\big(q(t)\big)$ does not change along a solution, given as $\frac{d}{dt}E(q, \dot{q}) = \dot{E}(q, \dot{q}) = 0$. This can be written as the following representation,

$$\big[E\big(q(t), \dot{q}(t)\big) - E\big(q(0), \dot{q}(0)\big)\big] \equiv 0. \tag{4.25}$$

A couple of simple examples are used to show how the conservation of energy (4.25) can be used to decide if a system is bounded. The first example used is that of an inverted pendulum,

$$\ddot{\theta} - \sin\theta = 0.$$

The total energy of this system is given by

$$E(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^2 + \cos\theta,$$

while the total amount of energy from the initial conditions is given by

$$E_0(\theta_0, \dot{\theta_0}) = \frac{1}{2}\dot{\theta_0}^2 + \cos\theta_0.$$

The conservation of energy becomes

$$E - E_0 = \frac{1}{2}(\dot{\theta}^2 - \underbrace{\dot{\theta}_0^2}_{\text{constant}}) + \underbrace{\cos\theta - \cos\theta_0}_{\text{bounded}} \equiv 0.$$

The last term is bounded because the cosine function is bounded, while the initial angular velocity of the pendulum is bounded because it is a constant. The angular velocity of the system therefore has to be bounded to make the total amount of energy in the system unchanged. In other words, the system is bounded.

The next simple example used is that of a hyperbolic pendulum,

$$\ddot{\theta} - \theta = 0.$$

The total energy of this system is given by

$$E(\theta, \dot{\theta}) = \frac{1}{2}\dot{\theta}^2 - \frac{1}{2}\theta^2,$$

while the total amount of energy from the initial conditions is given by

$$E_0(\theta_0, \dot{\theta_0}) = \frac{1}{2}\dot{\theta_0}^2 - \frac{1}{2}\theta_0^2.$$

The conservation of energy becomes

$$E - E_0 = \frac{1}{2}(\dot{\theta}^2 - \underbrace{\dot{\theta}_0^2}_{\text{constant}}) - \frac{1}{2}(\theta^2 - \underbrace{\theta_0^2}_{\text{constant}}) \equiv 0.$$

The initial angular position and velocity are bounded as they are constants. The same can not be said for the position and velocity in general. If the angular position happened to increase forever ($\theta \to \infty$), the angular velocity would have to do the same ($\dot{\theta} \to \infty$). This is because the two have to cancel each other out to keep the total energy in the system unchanged. There is no bounds on θ and $\dot{\theta}$, and the system is therefore unbounded.

Now that the concept of bounded and unbounded systems are explained, the next step would be to apply this to the ”Butterfly” robot. Firstly, pick a solution which gives a certain set of initial conditions and insert this into the law of conservation together with the total energy of the system. Then analyze the equation and determine if the system is bounded or not.

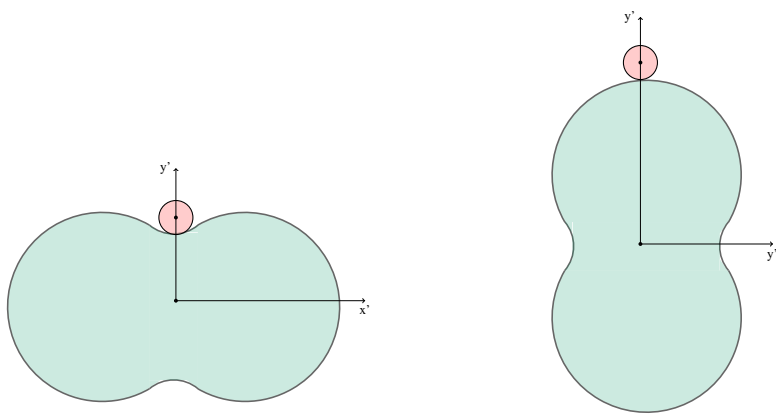
4.3 Deriving a Solution

Now that the desired motion has been decided, and the verification process for that said motion has been explained, it is time to put the two together to come up with a possible

solution. The solution's potential in reality will also be considered, i.e. making sure the normal force is greater than zero so the ball does not leave the frame. The process of implementing the system and the solution into a numerical computer program to produce simulations is also explained.

4.3.1 Deriving a VHC

The initial design of a virtual holonomic constraint is decided based on intuition. Before being able to derive an appropriate VHC, vast knowledge about the system must be acquired. Knowing the system leads to a design where the VHC maps the points $\varphi = 0 + \frac{\pi}{2}k,\ k \in \mathbb{Z}$. This is because the two angles θ and φ are identical at these locations, making the ball balance directly at the very top of the shape.



(a) Ball resting in valley of butterfly shape

(b) Ball balancing on side-plateau

Figure 4.3: Positions mapped for a full rotation

At the point $(\theta, \varphi) = (0, 0)$ the ball is resting in the valley of the butterfly frame, see Figure 4.3a, while at the point $(\theta, \varphi) = (\frac{\pi}{2}, \frac{\pi}{2})$, the ball is balancing at the top of the side-plateau of the butterfly frame, see Figure 4.3b. The next point $(\theta, \varphi) = (\pi, \pi)$, is identical to $(\theta, \varphi) = (0, 0)$. The system has then completed one rotation, and the cycle can repeat itself with $\theta = \Theta(\varphi)$ and $\varphi = 0 + \frac{\pi}{2}k,\ k \in \mathbb{Z}$.

In addition to mapping the points mentioned above, the VHC needs to satisfy the conditions provided by the one-directional desired motion, see (4.14) and (4.15). This includes $\Theta(\varphi)$ being twice differentiable and continuously increasing. There are endless functions that can satisfy being twice differentiable and continuously increasing, but the choices are substantially reduced when remembering that $\Theta'(\varphi)$ and $\Theta''(\varphi)$ must be periodic with π rad. A natural choice when requiring a periodic motion is to use a cosine or sine function. This needs to fit together with the mapping of the points and the need for continuous enlargement of the VHC. This resulted in a choice where $\Theta(\varphi)$ uses φ as a parameter, which turns into a constant when the expression is differentiated. The chosen VHC is therefore

$$\theta = \Theta(\varphi) = \varphi - c\sin(2\varphi), \tag{4.26}$$

which satisfies all the mentioned conditions. The derivatives of (4.26) are periodic with π rad, which can be seen from their expressions,

$$\begin{aligned} \Theta'(\varphi) &= 1 - 2c\cos(2\varphi), \\ \Theta''(\varphi) &= 4c\sin(2\varphi). \end{aligned} \tag{4.27}$$

The next step is checking how and if (4.26) leads to a valid and feasible solution. This is done by running it through the "Guide to Valid Solution" in Section 4.2.2.

1

The constant c, seen in (4.26), is included in the VHC to help in the process of excluding any asymptotes from the system. The parameter will be shaped to guarantee that $\alpha(\varphi) > 0$, making sure the acceleration $\ddot{\varphi}$ does not approach infinity. This is done by demanding $\alpha(\varphi) > 0$ for the equilibrium points $\varphi = 0$ and $\varphi = \frac{\pi}{2}$. Note that any pair of consecutive equilibrium points could be used, as the system is periodic with π rad, e.q. ($\varphi = \pi$ and $\varphi = \frac{3\pi}{2}$). Figure 4.4 shows the boundaries set by the two equilibrium points.

The constant is chosen from the green-colored area of Figure 4.4:

$$c = 0.49,$$

and will be used in all the upcoming simulations. The chosen parameter c gives a system where $\alpha > 0$, displayed in Figure 4.5.

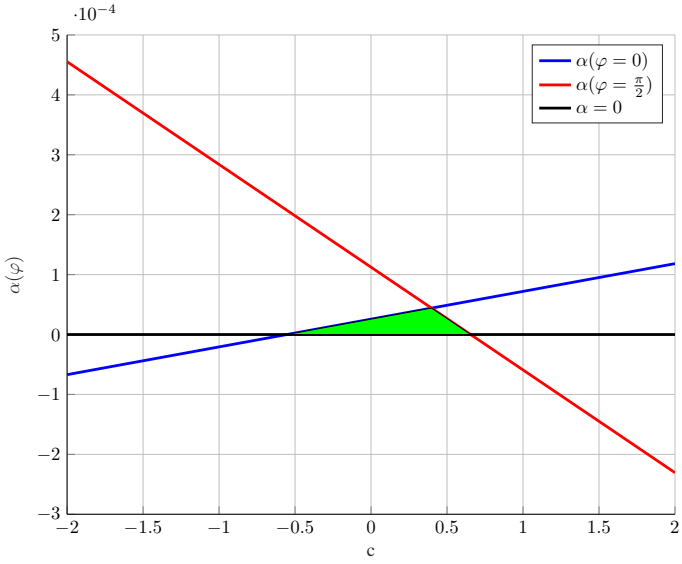


Figure 4.4: Determining constant for chosen VHC

The behaviour of the condition in (4.18) is plotted in Figure 4.6 together with the actual behaviour of the chosen VHC. If the function of the chosen $\Theta'(\varphi)_{chosen}$ had crossed the function of the invalid $\Theta'(\varphi)_{invalid}$ at any point, the condition of (4.18) would not have been satisfied. The fact that the two do not touch each other at any point in the simulation with $\varphi \in [0, 2\pi]$, and the fact that the motion is periodic makes it so the condition is always satisfied.

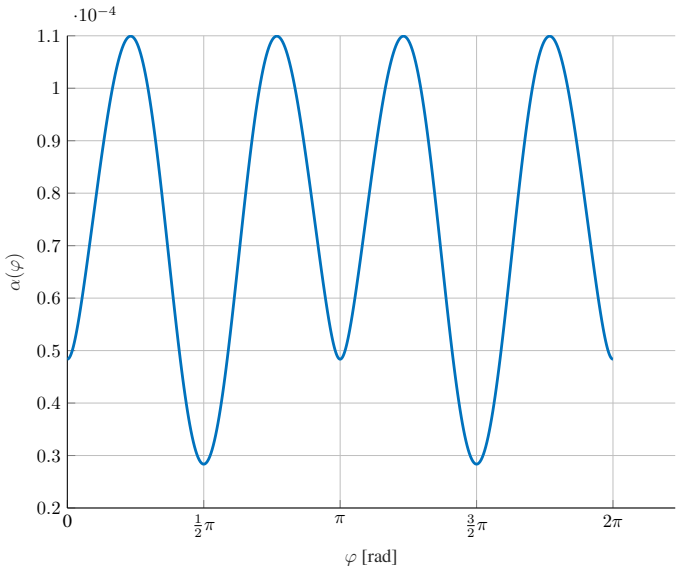


Figure 4.5: π periodic α-function always greater than zero

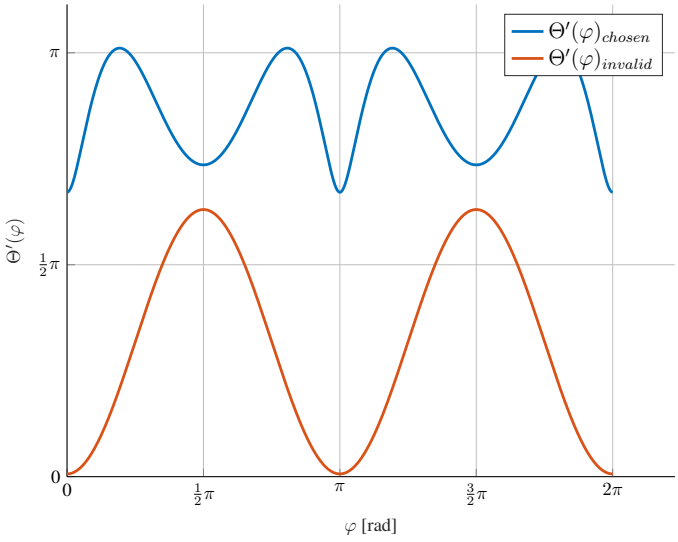


Figure 4.6: Examening condition on VHC

Figure 4.7 shows the behavior of both expressions in (4.17), and it can be seen that they are both finite. This means that φ_a in (4.19) does not exist, which confirms that there are no asymptotes in the system, and it guarantees that neither $\beta(\varphi)$ or $\gamma(\varphi)$ goes to infinity.

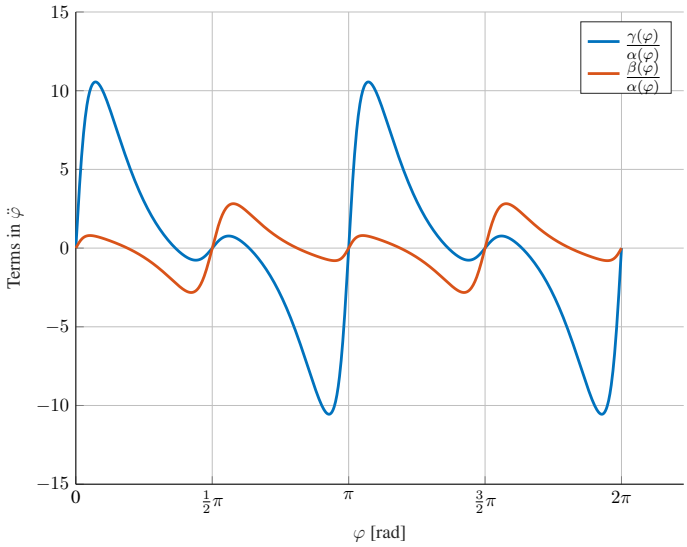


Figure 4.7: Checking system for asymptote

2

Certain conditions were put on $\Theta(\varphi)$ to find the equilibrium points of the system. The conditions are summarized as

$$\begin{aligned} \sin\big(\Theta(0+n\pi)\big) = 0, \quad \cos\big(\Theta(0+n\pi)\big) \neq 0, \\ \cos\big(\Theta(0+\frac{m\pi}{2})\big) = 0, \quad \sin\big(\Theta(0+\frac{m\pi}{2})\big) \neq 0, \end{aligned} \tag{4.28}$$

with $n = 1, 2, 3, ..., k$ and $m = 1, 3, 5, ..., 2k+1$ for $k \in \mathbb{Z}$.

The content of the sine and cosine functions used in (4.28) are analyzed using $\Theta(\varphi) = \varphi - c\sin(2\varphi)$.

$$\begin{aligned} \Theta(0+n\pi) &= n\pi - c\sin(2n\pi) = n\pi, \\ \Theta\big(0+\frac{m\pi}{2}\big) &= \frac{m\pi}{2} - c\sin(m\pi) = \frac{m\pi}{2}. \end{aligned}$$

Putting this back into (4.28) gives the following

$$\sin(n\pi) = 0, \quad \cos(n\pi) = \pm 1,$$
$$\cos\left(\frac{m\pi}{2}\right) = 0, \quad \sin\left(\frac{m\pi}{2}\right) = \pm 1,$$

which completely satisfies the conditions as $n \in \mathbb{Z}$ and m are all the odd numbers $\in \mathbb{Z}$.

All the conditions in (4.28) were satisfied by the chosen VHC and the resulting equilibrium points can be seen in Figure 4.8. Since $\alpha(\varphi) \neq 0$, the equilibrium points are defined when $\gamma(\varphi) = 0$.

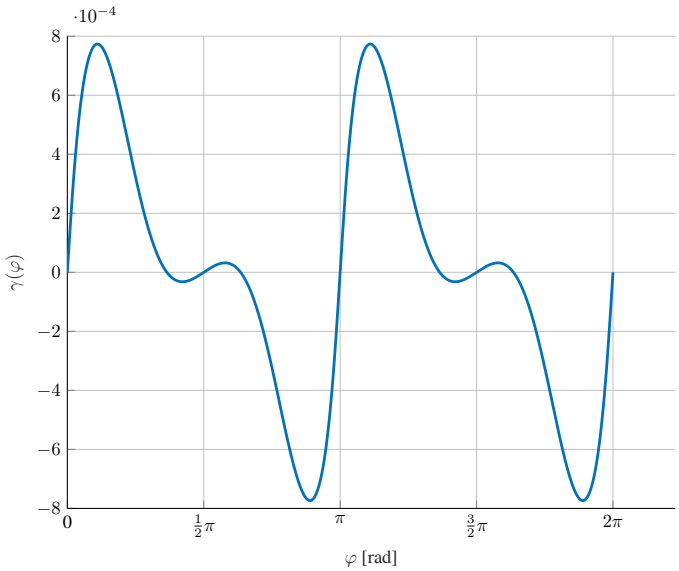


Figure 4.8: Equilibrium points

3

The condition in (4.23) is automatically satisfied as it was previously found that $\alpha(\varphi) > 0$ (see Figure 4.5). This means that the sign of $\gamma'(\varphi)$ determines the sign of $\frac{\gamma'(\varphi)}{\alpha(\varphi)}$, which again will determine if the point is a saddle or a center. Determining the sign of $\gamma'(\varphi)$ is dependent on $\Theta'(\varphi)$,

$$\begin{aligned} \Theta'\big(0+n\pi\big) &> \frac{a-5b}{a-b}, \\ \Theta'\big(0+\frac{m\pi}{2}\big) &< \frac{a+5b}{a+b}, \end{aligned} \tag{4.29}$$

with $n = 1, 2, 3, ..., k$ and $m = 1, 3, 5, ..., 2k+1$ for $k \in \mathbb{Z}$, and where

$$\begin{aligned} \Theta'\big(0+n\pi\big) &= 1 - 2c\cos(2n\pi) = 1 - 2c, \\ \Theta'\big(0+\frac{m\pi}{2}\big) &= 1 - 2c\cos(m\pi) = 1 + 2c. \end{aligned}$$

By using the parameter values a and b presented in Table 4.1, along with the previously shaped c, the conditions in (4.29) are both satisfied,

$$\begin{aligned} 1 - 2c &> \frac{a-5b}{a-b}, \\ 1 + 2c &< \frac{a+5b}{a+b}. \end{aligned}$$

This results in the following

$$\begin{aligned} \gamma'(\bar{\varphi}) > 0 \Rightarrow \frac{\gamma'(\varphi)}{\alpha(\varphi)} > 0 \Rightarrow \text{ center point } \Rightarrow \bar{\varphi} = 0 + n\pi, \\ \gamma'(\bar{\varphi}) < 0 \Rightarrow \frac{\gamma'(\varphi)}{\alpha(\varphi)} < 0 \Rightarrow \text{ saddle point } \Rightarrow \bar{\varphi} = 0 + \frac{m\pi}{2}. \end{aligned}$$

Figure 4.9 confirms that $\frac{\gamma'(\varphi)}{\alpha(\varphi)}$ is positive when $\varphi = n\pi$, for all $n \in \mathbb{Z}$, and that $\frac{\gamma'(\varphi)}{\alpha(\varphi)}$ is negative when $\varphi = \frac{m\pi}{2}$, for all odd numbers $m \in \mathbb{Z}$.

Simulations with the chosen VHC is performed in Figure 4.10. This simulation clearly confirms the placements of the center and saddle points. The simulation is in accordance with the derived theory. The equilibrium points of the system have now been identified, and a solution can be found by choosing a certain set of initial conditions.

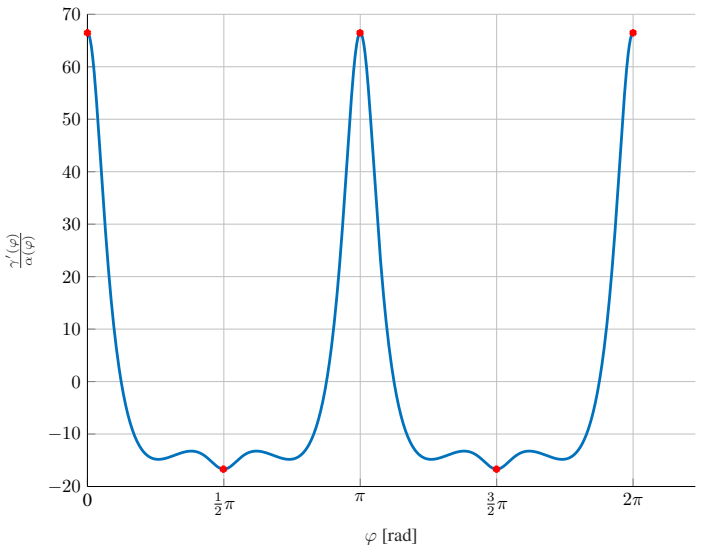


Figure 4.9: Classifying equilibrium points based on sign of $\frac{\gamma'(\varphi)}{\alpha(\varphi)}$

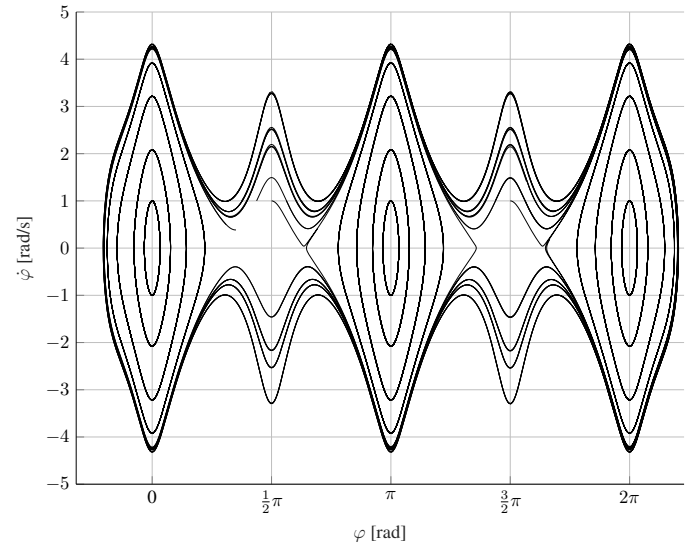


Figure 4.10: Phase trajectory

4

Unfortunately, finding an analytical representation of the total energy in the system $E(\varphi, \dot{\varphi})$ is very difficult. In complex cases like the "Butterfly" system, a more general approach must be taken. The expression of (4.25) is transformed into the more generic integral of motion, seen in (4.30). Every system containing virtual constrains can be written in the form of (4.9), and it is therefore a natural starting point for the derivation of the integral of motion. This derivation process can be seen in the Appendix of [16]. A proof of the integral of motion can be viewed in [17]. It is given as

$$I_0(\varphi, \dot{\varphi}, \varphi_0, \dot{\varphi}_0) = \dot{\varphi}^2 - exp\bigg(-2\int_{\varphi_0}^{\varphi} \frac{\beta(\tau)}{\alpha(\tau)} d\tau\bigg)\bigg[\dot{\varphi}_0^2 - 2\int_{\varphi_0}^{\varphi} \frac{\gamma(s)}{\alpha(s)} exp\bigg(2\int_{\varphi_0}^{s} \frac{\beta(\tau)}{\alpha(\tau)} d\tau\bigg) ds\bigg], \tag{4.30}$$

where α, β and γ are the periodic functions in the reduced dynamics. The function I_0 in (4.30) must preserve a zero along the solution, which can be used in the same way the conservation of energy was used to identify if the system is bounded.

Summary

The chosen vitual holonomic constraint of $\theta = \Theta(\varphi) = \varphi - c\sin(2\varphi)$ has satisfied all the previously found conditions for $\Theta(\varphi)$, $\Theta'(\varphi)$, $\Theta''(\varphi)$. Figure 4.11 shows how $\Theta(\varphi)$ is twice differentiable and continuously increasing in a periodic fashion of π rad, while $\Theta'(\varphi)$ and $\Theta''(\varphi)$ are periodic with π rad. The VHC can therefore be used to find a feasible trajectory, which will be a part of the upcoming subsection.

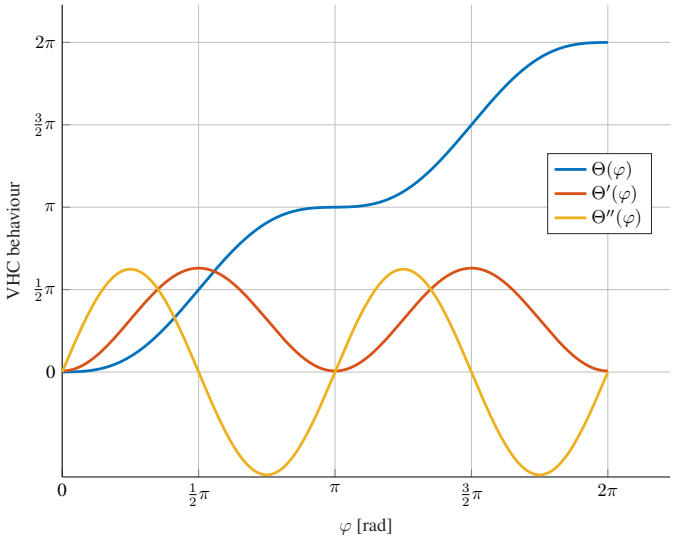


Figure 4.11: Periodic motion of VHC and its derivatives

4.3.2 Implementation

A feasible solution is found by experimentally simulating the behaviour of the full system for different sets of initial conditions. To be able to achieve these simulations, the full system must be implemented into a numerical computing software, which in this thesis is chosen to be MATLAB. The process of transforming the mathematical expressions of physical dynamics into code is discussed in the upcoming subsection. The choices surrounding implementation is explained, in addition to the choice of a specific set of initial conditions.

The main file sets a certain set of initial conditions and inputs these into a function located in a class named *ButterflyRobot*, which incidentally is the only class of the system. The class has a structured layout and runs simulations with the initial conditions received by the main, and returns the behaviour of the general coordinates in the system. The setup and content of the class is demonstrated in the compacted chunk of code shown below.

```
classdef ButterflyRobot
   % Dynamics of Butterfly robot

   properties

   methods
       %% Constructor
       function obj = ButterflyRobot()

       %% Alpha,Beta,Gamma
       function [t,x] = simABG(obj,t0,tEnd,x0)
       function dx = ABG_EOM(obj,x)

       %% Butterfly Robot
       function [t,x,Fn,Fs] = simBR(obj,t0,tEnd,x0)
       function dx = Butterfly_EOM(obj,x)
       function u = getActuatorInput(obj,x)
       function [Fn,Fs] = getForces(obj,x)

       %% Shared Functions
       function [M,C,G] = getMCG(obj,x)
       function [rho,tau,n,kappa,Ds,DDs,Pi,DPi] = getVariables(obj,x)
       function [deltaVec,DdeltaVec,DDdeltaVec] = getDeltaVecs(obj,phi)
       function [Alpha,Beta,Gamma] = getABG(obj,varphi)
       function [Phi,DPhi,DDPhi] = getPhis(obj,varphi)
       function [Theta,DTheta,DDTheta] = getVHC(obj,varphi)

       %% Symbolic Functions
       function [delta,Ddelta,DDdelta,DDDdelta] = getDELTAs(obj,phi)
       function [alpha,Dalpha,DDalpha] = getALPHAs(obj,phi)
       function [g,Dg,DDg] = getGs(obj,phi)
```

The '**Constructor**' section fits a spline to the function $\varphi = g(\phi)$, as previously mentioned at the end of Subsection 3.2.5. This makes it possible to retrieve a value of ϕ based on a value of φ throughout the class functions.

The '**Alpha, Beta, Gamma**' and '**Butterfly Robot**' sections runs two different versions of the system. This choice is made in the main file, not in the class. The function *simABG* uses the initial conditions $(\varphi_0, \dot{\varphi}_0)$ and outputs the behaviour of $(\varphi, \dot{\varphi})$ based on the $\alpha(\varphi)$, $\beta(\varphi)$ and $\gamma(\varphi)$ derived from the chosen VHC. This simulation was used to find the nominal phase trajectory. The function *simBR* uses the initial conditions $(\theta_0, \varphi_0, \dot{\theta}_0, \dot{\varphi}_0)$ in an open-loop simulation of the full system and outputs the behaviour of all the generalized coordinates $(\theta, \varphi, \dot{\theta}, \dot{\varphi})$. Note that the VHC is implemented through the actuation u, see (4.4).

Both the versions of the system uses a Ordinary Differential Equations Solver (ODE Solver) to help solve an initial value problem. This means that the ODE is solved by starting from an initial state (given by the initial conditions) and stepping through the problem iteratively, with each step of the solver applying a particular algorithm to the results of the previous step. When the iterative process is over, the ODE solver returns a vector or matrix containing a solution at each step, in addition to a vector of time steps. The default solver choice is *ode45*, but unfortunately it was too inefficient as the step size of the solver was forced down to an unreasonably small level, meaning that the system was too stiff for the explicit solver. The solver *ode23* was selected with fixed-step due to its efficiency on systems with moderate stiffness, and because it provided the accuracy needed to produce smooth graphs. The absolute tolerance of *ode23* was set to the default value of 1e-6 to prevent large absolute errors at any step in the simulation. The relative tolerance of *ode23* was set to 1e-5 so that virtually no error tolerance relative to the state vector was allowed at each simulation step.

Section '**Shared Functions**' includes all the system dynamics and is used by both versions of the system. The functions under the '**Symbolic Functions**' section were externally gen-

erated with the Symbolic Math Toolbox version 7.2. Complex mathematical expressions were implemented into MATLAB as symbolic functions. These symbolic functions were differentiated and automatically made into new, independent MATLAB functions, which again were transferred into the system's code.

Now that the setup of the code is explained, the next part is to justify the choice of parameter values used in the simulations. All the parameters, with the exception of R_f and c were determined by Surov and Shiriaev at el. Some were taken from [1], and some were collected from conversations with Shiriaev. As an expert on the "Butterfly" robot and a supervisor for this thesis, Shiriaev's previously found values were determined to be the best possible fit for this system. The parameter c was derived in Subsection 4.3.1, while R_f was estimated by scaling it to the rest of the components in the system. All the parameters used are given in Table 4.1, and implemented as properties in the class *ButterflyRobot*. See **List of Symbols** for a description of all the variable names.

Table 4.1: Model parameters for simulation

Parameters	Value	Unit
m	$3.0 \cdot 10^{-3}$	m
J_f	$1.581 \cdot 10^{-3}$	kg m^2
J_b	$5.48 \cdot 10^{-7}$	kg m^2
g	9.81	m s^{-2}
R_b	$16.55 \cdot 10^{-3}$	m
r_f	$12.5 \cdot 10^{-3}$	m
R_f	0.1	m
a	0.1095	-
b	0.0405	-
c	0.49	-

The small excerpt of code previously shown can be seen in its entirety in Appendix C, along with all the other scripts used. These other scripts do not simulate the full system,

but rather help explain sub-parts of the system sprinkled throughout the thesis. An explanation of all the scripts is included in Appendix C.

Now that the system is implemented into MATLAB, the next step is to simulate a feasible solution. This is done by choosing a set of initial conditions that produces the right behaviour from the system. It is known from Subsection 4.2.1 that the correct behaviour is a saddle, not a center. This means that the angular velocity of φ must be above a certain point so it escapes the bounds of the centers, as Figure 4.10 illustrates. Knowing this, plus the fact that the ball should start in the valley of the butterfly shape (so it does not slide off the frame), the initial values are chosen as $(\varphi_0, \dot{\varphi}_0) = (0, 4.3)$. The initial value $\dot{\varphi}_0$ was found experimentally with the trial and error method seeing where the first saddle entered the system. The chosen initial conditions generates the value of the two remaining initial conditions $(\theta_0, \dot{\theta}_0)$ through the VHC,

$$\theta_0 = \Theta(\varphi_0) = \varphi_0 - c\sin(2\varphi_0) = 0, \tag{4.31a}$$

$$\dot{\theta}_0 = \Theta'(\varphi_0)\dot{\varphi}_0 = [1 - 2c\cos(2\varphi_0)]\dot{\varphi}_0 = [1 - 2c]\dot{\varphi}_0 = 0.086. \tag{4.31b}$$

In summary, the initial conditions are $(\theta_0, \varphi_0, \dot{\theta}_0, \dot{\varphi}_0) = (0, 0, 0.086, 4.3)$. Figure 4.12 depicts what these initial conditions look like on the "Butterfly" robot. Note that since $\theta_0 = 0$, the inertial frame is identical to the body frame of the butterfly. This means that the inertial frame is hidden underneath the body frame of the butterfly.

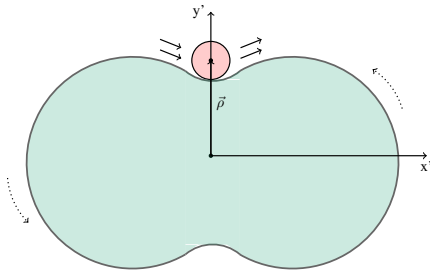


Figure 4.12: Schematic view of initial conditions for simulations

4.3.3 Simulation Results

The following simulations are the results from using the initial conditions $(\theta_0, \varphi_0, \dot{\theta}_0, \dot{\varphi}_0) = (0, 0, 0.086, 4.3)$. The results represent a solution for the system and this solution is represented as $q_\star(t) = (\theta_\star(t), \varphi_\star(t))^T$ and $\dot{q}_\star(t) = (\dot{\theta}_\star(t), \dot{\varphi}_\star(t))^T$ for $t \in \mathbb{R}$. The results following the heading; **Nominal solution**, are simulated with `simABG` and represent the desired motion, while the results following the heading; **Simulated solution**, are simulated with `simBR` and represent the actual motion of the system. The figures following the heading; **Comparing solutions**, clearly illustrates similarities and dissimilarities between the nominal and experimental simulations.

Nominal solution:

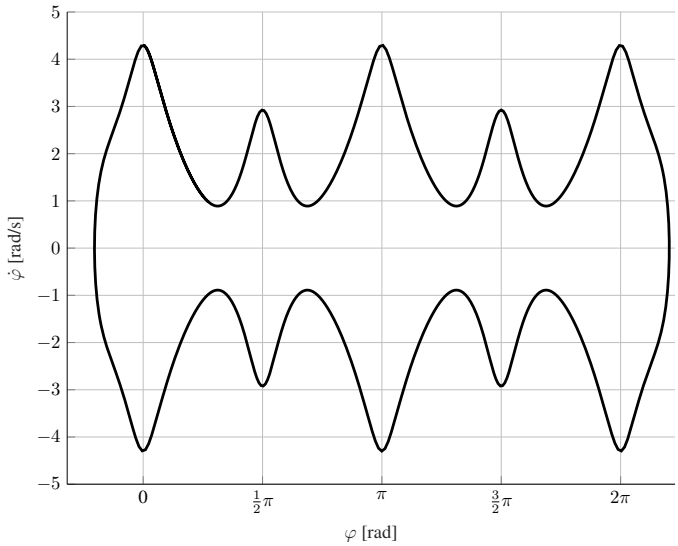


Figure 4.13: Complete phase trajectory for designed solution $\varphi_\star$

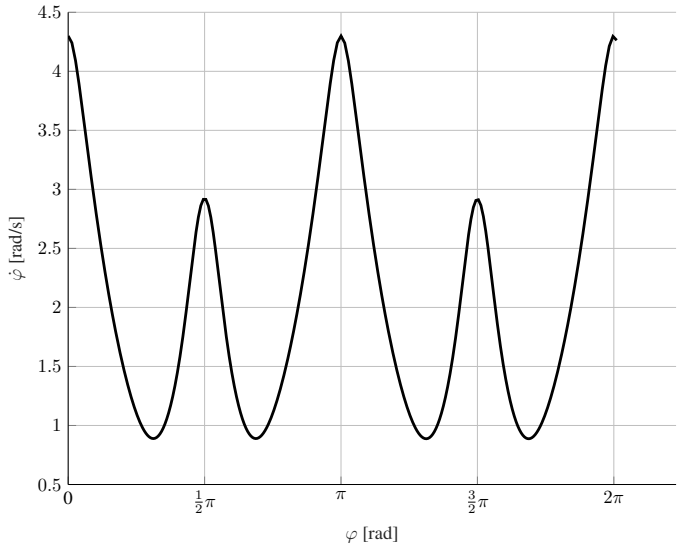


Figure 4.14: Selected phase trajectory $\varphi_\star$ (cut from of Figure 4.13)

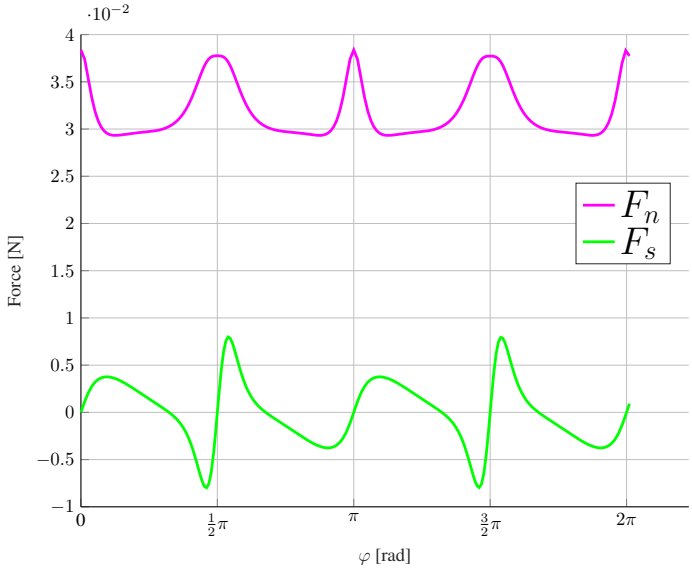


Figure 4.15: Constraint forces parameterized by φ

Simulated solution:

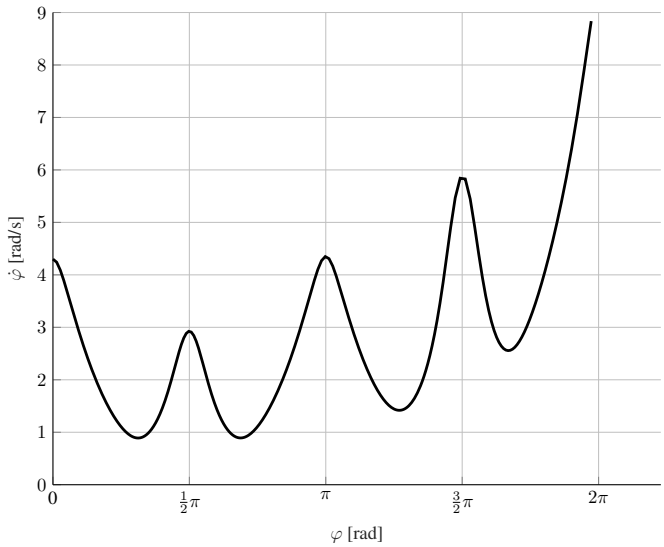


Figure 4.16: Actual phase trajectory for solution $\varphi_\star$

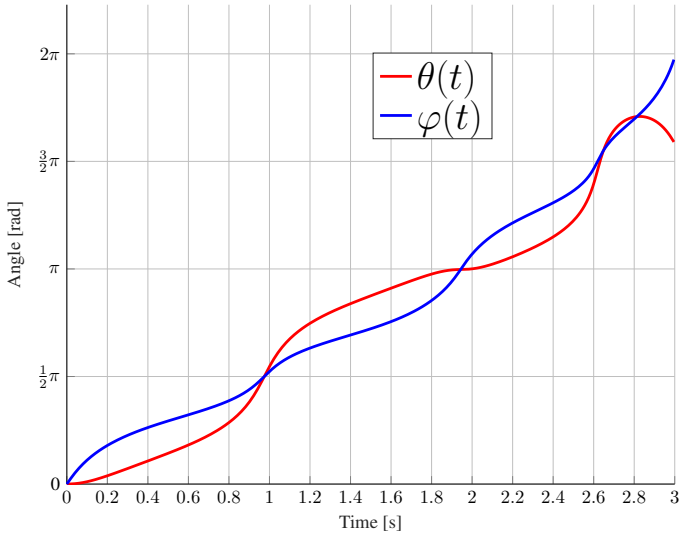


Figure 4.17: Angular position of ball and frame for solution $q_\star$

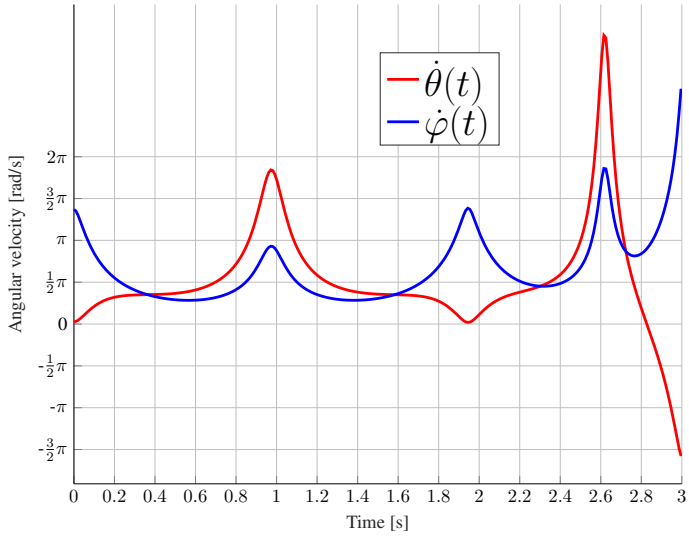


Figure 4.18: Angular velocity of ball and frame for solution $\dot{q}_\star$

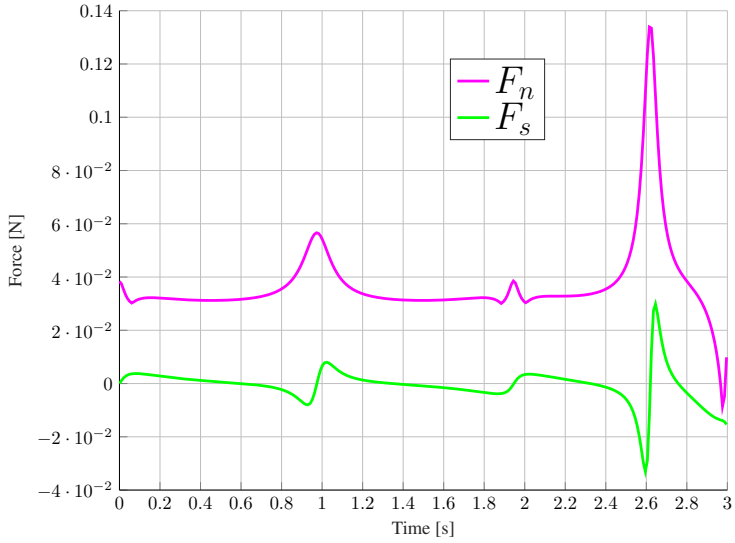


Figure 4.19: Constraint forces

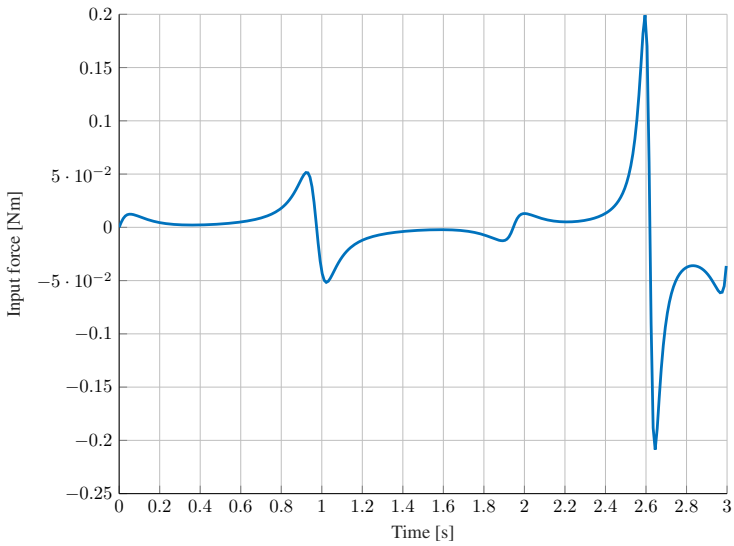
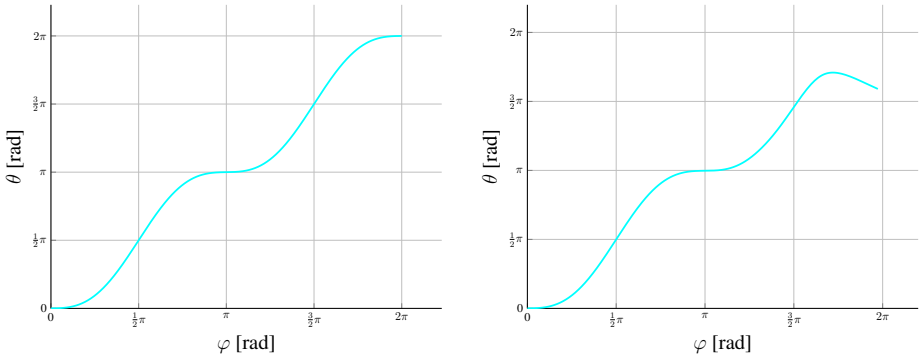


Figure 4.20: Actuation force

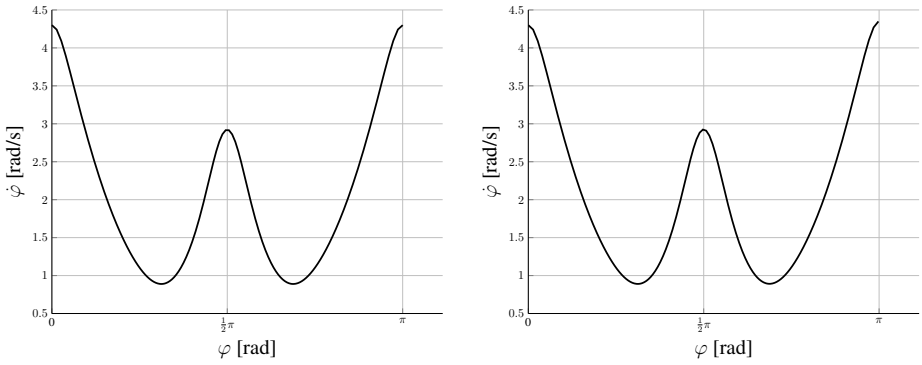
Comparing solutions



(a) Nominal relationship

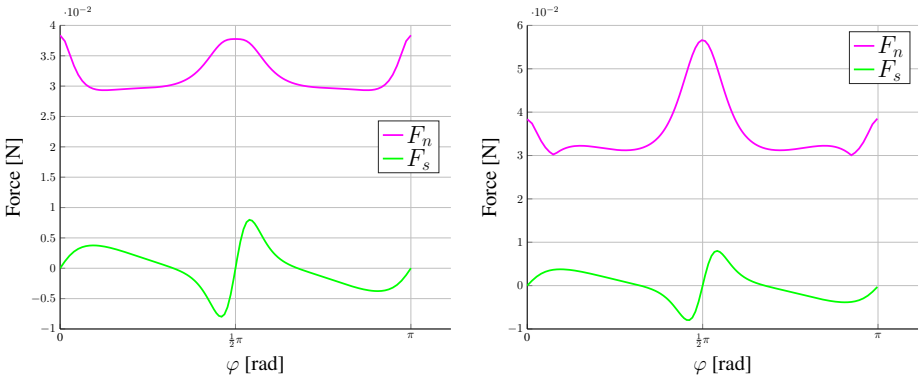
(b) Simulated relationship

Figure 4.21: Relationship between generalized coordinates given by $\Theta(\varphi)$



(a) Nominal trajectory (b) Simulated trajectory

Figure 4.22: Phase trajectory



(a) Nominal forces (b) Simulated forces

Figure 4.23: Constraint forces

Chapter 5

Discussion

In the previous chapter, a solution was derived for the ”Butterfly” robot. This upcoming chapter will discuss the achieved results and compare the theoretical work done in this thesis with the actual physical system.

5.1 Setup of physical system

Before analyzing the solution, some information about the physical system is presented. Knowledge of how the experimental system works will provide further understanding of the reasoning behind the derived solution. The experimental setup can be seen in Figure 5.1.

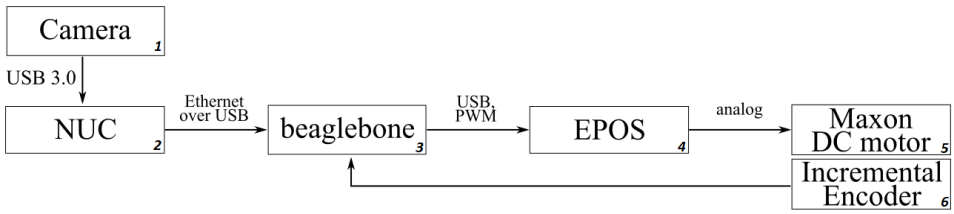


Figure 5.1: Experimental setup

The hardware equipment used in the experimental setup is given below, with the corresponding model name included:

1. **Camera** → acA1300-200uc Basler ace
2. **NUC** → INTEL KIT NUC5i5RYH
3. **BeagleBone** → Green (BBG)
4. **EPOS** → ESCON Motor Controller 70/10
5. **Maxon DC motor** → 370356 RE50
6. **Incremental Encoder** → SCANCON 8192

The camera constantly takes pictures and provides information about the location of the ball and frame. This information arrives to the NUC, which has to convert the information from pictures to a systematic, symmetric system that represents the ball and frame. The beaglebone collects the data from the NUC, along with speed and distance feedback from the encoder, and uses all of this to calculate velocity and center of symmetry for the ball. Based on the systems reaction to the newly calculated position of the ball, the motor controller regulates the speed and direction of the electric motor by manipulating the voltage that is applied to it. The DC motor drives the robot torque of the frame, before the whole process repeats itself.

There are certain limitations connected to the hardware used. The sampling frequency of the camera needs to be high enough to pick up the motion of the ball. Surov and his team experimentally tested different cameras in the experimental setup and concluded that the system needed a camera with a capture rate of about 120 Hz. Another limitation on the hardware is the speed of the image processing required. The system needs to extract information about the velocity and location of the ball really fast, which can be hard because of the intense calculations that are constantly going on. Two different frames captured by the camera go through the NUC and are compared to find out how much the ball's position has changed. The complexity increases drastically when remembering that the surface the ball rolls on (butterfly shape) also moves. And this whole process, from the camera taking

a picture to the fully detected location of the center of symmetry of the ball must take less than 2 ms. Considering these limitations on the physical system can be useful when deriving a solution, as will be explained in the next section.

5.2 Evaluate Solution

Theoretically, there are an infinite number of possibilities when choosing the initial conditions. A certain set of initial conditions will produce a specific solution for the system, but that does not guarantee the solution to be feasible in reality. The experimental system consist of real components that have limitations, just as the information in Section 5.1 explained. One limitation is the sampling frequency of the camera, as it measures the system's configurations. It is therefore crucial to consider the ball's angular velocity when searching for a solution. Minimizing the angular velocity of the ball will reduce the need for an expensive camera with high sampling frequency, and thereby contribute to a feasible solution obtained from an affordable experimental system.

Even though low angular velocity of the ball is desired, it is important that the velocity of the motion is not too low, causing the trajectory to enter a center. The motion must be of a saddle, which means that there are restrictions on how low the velocity can get. This can be visualized in Figure 4.10 as centers, saddles and separatrix are all present. A nominal solution is chosen by experimentally simulating the system with various initial conditions for the ball's velocity. The chosen trajectory $\varphi_\star$ is depicted in Figure 4.13, which is a solution extracted from Figure 4.10 together with the correlated initial conditions. This trajectory illustrates a saddle with almost the lowest possible angular velocity of the ball. The reason the absolute minimum velocity is not chosen is because some leeway is desired in case small deviations from the nominal trajectory occur, which could cause the behaviour to change from saddle to center. Figure 4.14 shows the nominal trajectory of continuous one-directional rotation of the ball around the frame. The maximum angular velocity ends up as

$$\dot{\varphi}_{max} = 4.3125 \text{ rad/s},$$

which is well within the sampling frequency of the camera in the physical setup.

Figure 4.15 shows the nominal constraint forces for the solution $\varphi_\star$. The normal force is always greater than zero, making sure the ball is always in contact with the frame. The behavior of the friction force sets a limit for the friction coefficient. This limit is set through the constraint (2.27) and is given as $\mu \geq 0.22$.

The actual behaviour of the system is obtained by simulating the full system in open-loop, resulting in Figures 4.16 - 4.20. Figure 4.16 shows the phase trajectory the system follows, which has similar features to the nominal trajectory, except for at the end where it deviates from the desired path. To further study this deviation from the nominal trajectory the position and angular speed of the ball and frame are plotted versus time, seen in Figure 4.17 and Figure 4.18. The two rigid bodies behave in the desired manner with the ball balancing on the frame in a one-directional motion for the first 2.6 seconds, before the system becomes unstable. This unstable behaviour is mirrored in the forces in Figure 4.19 and in the actuation input in Figure 4.20. Both the normal force F_n and the actuation u spikes at around 2.6 seconds and from there the normal force travels below zero, meaning that the ball leaves the frame. The actuation spikes because the system has slowly deviated from the nominal path and it wants to correct the error by applying more force. Instead of fixing the system, the sudden increase in torque makes the frame rotate really fast and the ball goes flying off the frame. Looking at the actuation input in Figure 4.20 before the system becomes unstable reveals a maximum input force of about 0.05 Nm, which can easily be achieved by most standard motors.

The system loses its stability early in the simulation as there does not exists an asymptotic stable motion for mechanical systems without using feedback. Since the system is run in open-loop, the system eventually becomes unstable, meaning that a stabilizing feedback controller is needed to correct deviations from the nominal trajectory. [1] introduces transverse-linearization-based orbital stabilization, which ensures robust orbital stabilization of the nominal trajectory. A natural continuation of this paper would be to

further derive an appropriate feedback controller based on orbital stabilization. This is done through linearization of the transverse dynamics around nominal points and can be further researched in [1], [18] and [19].

Now that the eventual instability of the system is explained, the focus lands on the first few seconds of the simulations. This small time period contains valuable information about the feasibility of the planned motion. These first few seconds are equivalent to one full rotation of the frame (2π rad), which again is equal to two periods for the frame since it has axis symmetry with π rad. Figure 4.21 shows the virtual holonomic constraint of both the nominal system and the open-loop system. The two are similar until the known deviations occur towards the end of the full rotation. Looking at just half of a full rotation, namely one period of π rad, shows that the two are in fact identical. Comparisons of the phase trajectory for the nominal and the open-loop system also show identical behaviour, see Figure 4.22. Lastly, the constraint forces of the nominal and the open-loop system are compared for the first period. The two friction forces are the same, while the open-loop normal force has a slightly higher peak value compared to the nominal normal force, as can be seen in Figure 4.23. This difference is negligible as the main concern is that the behaviour between the two are very similar, and both satisfy the constraints of (2.26) and (2.27).

5.3 Theoretical System Vs. Physical System

A feasible trajectory yielding a desired periodic motion has been found through theoretical work in this paper. The theoretical work is based on the concept of structural perturbations, which assumes that everything in the system (e.g. forces, degrees of freedom, etc.) can be predicted or modelled. In reality the system actually consists of some non-structural perturbation, which can reveal discrepancies between the theoretical and the physical system. Some examples that illustrate the mismatch between the theoretical and physical system are given,

- Possible misalignment between the two plates that make up the butterfly frame will result in a new degree of freedom because the ball can now move in one extra direction.
- The ball or the surface of the butterfly frame might not be completely smooth, making it virtually impossible to correctly model the friction in the system.
- The two rigid bodies (ball and frame) could potentially not have a uniform distribution of mass, changing the location of their center of mass.
- Unable to model the forces that generate the behaviour of an area contact, not a point contact that has been assumed in the paper.
- The ball is not guaranteed to roll without slipping, which leads to behaviour that is difficult to model.
- Forces that are dependent on velocity has a high error far away from equilibrium, which can accumulate into huge amounts of energy, even from small forces.

The theoretical model of the system is not a perfect representation of the physical system, and these differences might lead to complications when transferring the theoretical work to a physical system. On the other hand, [1] has proven that it is possible to overcome these difficulties with a well designed stabilizing controller. It is, however, important to be aware of the trouble associated with modeling the ”Butterfly” robot and other underactuated mechanical systems.

Chapter 6

Conclusion and Recommendations for Further Work

6.1 Conclusion

This paper presents valuable, new information on how to derive accurate system dynamics, and how to plan feasible trajectories for an underactuated mechanical system. The paper has used the benchmark example of the ”Butterfly” robot to illustrate the process, in addition to carefully explaining everything associated with said process. The ”Butterfly” robot continuously rolls a ball on top of a frame which is shaped like a butterfly and driven by a DC-motor.

The dynamics are derived with a common assumption omitted, making them more accurate than previously found dynamics. The second contribution of the paper is the detailed guide on how to plan feasible trajectories. The theory suggest that the found trajectory planning method delivers feasible trajectories, but original experiments are needed to validate that the transition from theory to reality is possible without fatal consequences for the system. However, since [1] has managed to produce physical experiments, there is no logical reason why the work done in this paper should not be able to. With this in mind,

the thesis will serve as an excellent source of information for further research on the ”Butterfly” robot.

6.2 Recommendations for Further Work

The concepts presented in this thesis has taken an enormous amount of time to fully understand. The intricate details of the system revealed some extremely complicated dynamics, and the underactuation of the mechanical system lead to a complex process of finding feasible trajectories. The paper presents these advanced concepts in a detailed manner in the hope that it will serve as a good base of knowledge for non-prehensile systems and that it will inspire further work in the field. Extending the work of the ”Butterfly” robot would be a natural continuation to this paper.

The first thing that needs to be done is to create a feedback controller that can stabilize the system. A feedback controller which ensures orbital stabilization of the nominal trajectory will be well-fitting. Finding this feedback controller must be achieved before implementing the model onto the physical system, which is a natural second step. Eventually, the derived dynamics has to be tested on the physical system to validate the theoretical contributions.

When the work above is finished, new feasible motions should be explored. New motions could be produced by actuating the frame in two directions, as opposed to just one direction. Another possibility is to apply the framework derived in this paper on a more complex shape of the frame (i.e. more complex than butterfly shape). A big step forward in the development of non-prehensile manipulation would be to add a third dimension. This would exponentially increase the complexity of the system. Imagine balancing a ball on top of another way bigger ball. This exact scenario might be too complicated at the moment, but an alternative approach would be to slightly move the two plates making-up the butterfly frame, so that they no longer are perfectly aligned. This will add a third dimension to the dynamics, which should be enough of a challenge.

Bibliography

[1] M. Surov, A.S. Shiriaev, L.B. Freidovich, S.V. Gusev and L. Paramonov. 'Case Study in Non-prehensile Manipulation: Planning and Orbital Stabilization of One-directional Rollings for the "Butterfly" Robot', *IEEE International Conference on Robotics and Automation*, 1484-1489, 2015.

[2] D.G.E Hobbelen and M. Wisse. 'Swing-Leg Retraction for Limit Cycle Walkers Improves Disturbance Rejection', *IEEE Transactions on Robotics*, 24(2): 377-389, 2008.

[3] C. Chevallereau, G. Abba, Y. Aoustin, F. Plestan, E.R. Westervelt, C. Canudas-de-Wit, and J.W. Grizzle. 'RABBIT: a Testbed for Advanced Control Theory', *IEEE Control Systems*, 23(5): 57-79, 2003.

[4] P. Lertkultanon and Q.C. Pham. 'Dynamic non-prehensile object transportation', *Conference on Control Automation Robotics & Vision (ICARCV 2014)*, 1392-1397, 2014.

[5] J.C. Ryu, F. Ruggiero and K.M. Lynch. 'Control of Nonprehensile Rolling Manipulation: Balancing a Disk on a Disk', *IEEE International Conference on Robotics and Automation*, 3232-3237, 2012.

[6] K.M Lynch and M.T. Mason. 'Dynamic Nonprehensile Manipulation: Controllability, Planning and Experiments', *International Journal of Robotics Research*, 18(1): 64-92, 1999.

[7] A.D. Luca, S. Iannitti, R. Mattone and G. Oriolo. 'Control Problems in Underactuated Manipulators', *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 855-861, 2001.

[8] K.M. Lynch, N. Shiroma, H. Arai and K. Tanie. 'The Roles of Shape and Motion in Dynamic Manipulation: The Butterfly Example', *IEEE International Conference on Robotics and Automation*, 1998.

[9] M. Cefalo, L. Lanari and G. Oriolo. 'Energy-Based Control of the Butterfly Robot', *IFAC Proceedings Volumes*, 39(15): 1-6, 2006.

[10] H.L. Chen. 'Alternative Approach in Modeling the Dynamics of the Butterfly Robot', Internship Report, 2016.

[11] M.W. Spong, S. Hutchinson and M. Vidyasagar. 'Robot Modeling and Control', Chapter 7.1-7.3: 239-257, John Wiley & Sons, Inc, 2006.

[12] A.S. Shiriaev, L.B. Freidovich and M.W. Spong. 'A Remark on Controlled Lagrangian Approach', *European Journal of Control*, 19(6): 438-444, 2013.

[13] M.W. Spong, S. Hutchinson and M. Vidyasagar. 'Robot Modeling and Control', Chapter 2.2: 38-65, John Wiley & Sons, Inc, 2006.

[14] R.A. Admas and C. Essex. 'Calculus, a Complete Course', Chapter 11.4: 642-649, Pearson 7^{th} edition, 2010.

[15] M. Maggiore and L. Consolini. 'Virtual Holonomic Constraints for Euler–Lagrange Systems', *IEEE Transaction on Automatic Control*, 58(4): 1001-1008, 2013.

[16] Leonid B. Freidovich, Uwe Mettin and Anton S. Shiriaev. 'A Passive 2-DOF Walker: Hunting for Gaits Using Virtual Holonomic Constraints', *IEEE Transactions on Robotics*, 25(5): 1202 - 1208, 2009.

[17] A.S. Shiriaev, A. Robertsson, J. Perram and A. Sandberg. 'Periodic Motion Planning for Virtually Constrained Euler–Lagrange Systems', *Systems & Control Letters*, 55(11): 900-907, 2006.

[18] A.S. Shiriaev, J.W. Perram and C. Canudas-de-Wit. 'Constructive Tool for Orbital Stabilization of Underactuated Nonlinear Systems: Virtual Constraints Approach', *IEEE Transactions on Automatic Control*, 50(8): 1164-1176, 2005.

[19] A.S. Shiriaev, L.B. Freidovich and S.V. Gusev. 'Transverse Linearization for Controlled Mechanical Systems With Several Passive Degrees of Freedom', *IEEE Transaction on Automatic Control*, 54(4): 893-906, 2010.

Appendix A

Frenet Frame

A smooth curve free of any points of inflection $\mathcal{C}$ (seen in Figure A.1) has several scalars and vectors describing its behaviour, which can help interpret the curves motion.

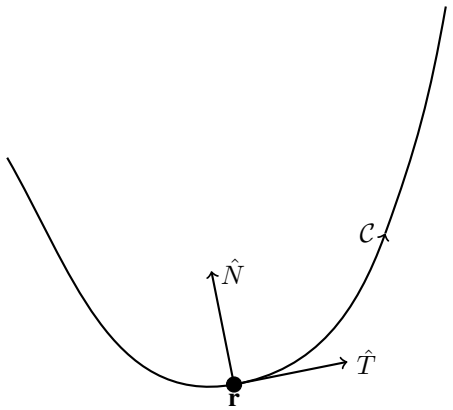


Figure A.1: The unit tangent and principal normal vectors for a curve

Let $\mathbf{r} = \mathbf{r}(s)$ be a parametrized space curve in terms of arc length. This means that $\mathbf{r}$ takes values in a three-dimensional Euclidean space represented by the arc length s, and can be interpreted as the trajectory of a particle moving along a curve $\mathcal{C}$. Arc-length

parametrization traced at unit speed gives a definition of the tangent vector

$$\hat{\mathbf{T}}(s) = \frac{d\mathbf{r}}{ds}. \tag{A.1}$$

By looking at how the curve $\mathcal{C}$ deviates away from the tangent line in $\mathbf{r}$, the rate at which the curve is turning can be measured. The curvature of $\mathcal{C}$ at point $\mathbf{r}(s)$ is defined as the length of $d\hat{\mathbf{T}}/ds$. If the curvature is never zero (meaning no straight lines), a unit principle vector can be defined as

$$\hat{\mathbf{N}}(s) = \frac{d\hat{\mathbf{T}}}{ds} \Big/ \left| \frac{d\hat{\mathbf{T}}}{ds} \right|. \tag{A.2}$$

The unit normal is perpendicular to $\mathcal{C}$ at $\mathbf{r}$ and points in the direction that the tangent vector, and therefore the curve, is turning. The setup of the curves and vectors is clearly shown in Figure A.1. The torsion of the curve, namely the rate at which the curve is twisting at any point, can be found by the help of the normal vector.

These two unit vectors, together with the unit binormal vector $\hat{\mathbf{B}} = \hat{\mathbf{T}} \times \hat{\mathbf{N}}$ (not included in this paper), form a basis at every point on a curve and is called the Frenet frame. For more detailed information about the Frenet frame, see [14].

In this paper, the trajectory of the ball has the same features as the parametrized space curve mentioned above, where the motion is happening along a curve. This allows and encourages the use of the Frenet frame to represent the motion and position of the ball.

The Formulas of Frenet

$$\frac{d\hat{\mathbf{T}}}{ds} = k\hat{\mathbf{N}}, \quad \frac{d\hat{\mathbf{N}}}{ds} = -k\hat{\mathbf{T}}, \tag{A.3}$$

where $k = |\frac{d\hat{\mathbf{T}}}{ds}|$ is the curvature at a given point.

Appendix B

Euler-Lagrange

A system with generalized coordinates $\mathbf{q} \in \mathbb{R}^n$ consisting of n rigid bodies will have its kinetic energy $\mathcal{K}$ defined as

$$\mathcal{K} = \frac{1}{2} \sum_{i=1}^{n} \left(m_i \mathbf{v}_i^T \mathbf{v}_i + I_i \boldsymbol{\omega}_i^T \boldsymbol{\omega}_i \right), \tag{B.1}$$

with $\mathbf{v}_i$ as translational velocity, $\boldsymbol{\omega}_i$ as angular velocity, m_i as mass and I_i as the moment of inertia. The potential energy $\mathcal{P}$ is defined as

$$\mathcal{P} = \sum_{i=1}^{n} m_i \mathbf{g}^T \mathbf{r}_i, \tag{B.2}$$

with $\mathbf{g}$ as the gravitational acceleration and $\mathbf{r}_i$ denoting the position of the particular body. Both the kinetic and potential energy can be represented by the general coordinate $\mathbf{q}$ by deriving the kinematics that connects the system variables to the vectors describing position and velocity and inserting them into the definitions. The Lagrangian of the system can then be written as

$$\mathcal{L}(q, \dot{q}) = \mathcal{K}(q, \dot{q}) - \mathcal{P}(q) = \frac{1}{2} \dot{q}^T M(q) \dot{q} - \mathcal{P}(q), \tag{B.3}$$

where $M(q)$ is an $n \times n$ symmetric, positive semi-definite inertia matrix whose elements are given by the relation

$$m_{i,j}(q) = \frac{\partial}{\partial \dot{q}_i}\left(\frac{\partial \mathcal{K}}{\partial \dot{q}_j}\right), \quad i,j = 1,2,...,n. \tag{B.4}$$

The $n \times n$ matrix $C(q,\dot{q})$ contains the centrifugal and Coriolis terms, where the $(k,j)^{th}$ element is defined as

$$c_{kj} = \sum_{i=1}^{n} \frac{1}{2}\left\{\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k}\right\}\dot{q}_i = \sum_{i=1}^{n} c_{ijk}(q)\dot{q}_i. \tag{B.5}$$

and the terms $c_{ijk}(q)$ are known as Christoffel symbols. The $n \times 1$ vector G(q) is known as the gravity vector with elements defined as

$$g_i = \frac{\partial \mathcal{P}}{\partial q_i} \tag{B.6}$$

The three definitions of M, C and G presented above are used in the transformation from the Euler-Lagrange equation to the equations of motion. By using the Langrangian in (B.3), the Euler-Lagrange equation can be found and is defined as

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = \tau, \tag{B.7}$$

which can be written as

$$\sum_{j=1}^{n} m_{kj}(q)\ddot{q}_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}(q)\dot{q}_i\dot{q}_j + g_k(q) = \tau_k, \quad k = 1,...,n. \tag{B.8}$$

Proof of both these claims are omitted due to the large space required to explain it properly, but can be studied in [11]. Equation (B.8) is commonly written in matrix form as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau, \tag{B.9}$$

characterized as the equations of motion of the system. For more information about the Euler-Lagrange equations or the equations of motion, see [11].

Appendix C

Code

MATLAB R2017a code

There is a quick and simple explanation of every .m-file before the code is presented, in addition to a little description within the scripts themselves. Note that many of the scripts have multiple choices for certain variables so that the code can be used for multiple instances (e.g. circle, ellipse, butterfly) and still stay compact. Multiple choices are of course clearly marked for the users benefit. The comments in the code should be enough to fully understand all the choices the user has.

All the code used in the thesis is supplied in a zip-folder along with the paper. The code included in this appendix is a compacted version of the full code presented in the zip-folder. All the plotting is excluded from the code enclosed in the paper due to the substantial amount of space that this consumes. Also, the functions created from the symbolic scripts are used in every single file. It seems unnecessary to show the same chunk of code for every script, and the symbolic script functions are therefore only presented in its entirety in the main script. In the rest of the scripts the function headline is presented without the associated code.

runButterflyRobot.m

This is the main script that can run the two most important features; the function that tests the feasibility of new velocity profiles or the function that drives the full system in open-loop. They are run through the *ButterflyRobot* class.

```
%% Author: Oskar Lund, 11.10.17

%% Description
% This script simulates the overall behaviour of the
% butterfly robot in open loop, in addition to new,
% potential trajectories

%% Define the object BR of class ButterflyRobot
BR = ButterflyRobot;

%% Choose which program to run: ABG = 0, BR = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
program = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (program == 0)
        %% Simulate alpha, beta, gamma

    t0_   = 0;  % Intial time
    tEnd_ = 10;  % Final time
    x0_   = [0; 4.3]; % Initial conditions ([varphi0;Dvarphi0])
    [t_,x_] = BR.simABG(t0_,tEnd_,x0_);

elseif (program == 1)
        %% Simulate the full butterfly system

    t0   = 0;  % Intial time
    tEnd = 5;  % Final time
    x0   = [0; 0; 0.086; 4.3]; % Initial conditions ([theta0;varphi0;
    Dtheta0;Dvarphi0])
    [t,x,Fn,Fs,u] = BR.simBR(t0,tEnd,x0);

end
```

ButterflyRobot.m

This script contains the *ButterflyRobot* class, which is used by the main file. All the relevant information derived in the paper is located here (e.g. dynamics and VHC).

```
classdef ButterflyRobot
   % Dynamics of Butterfly robot

   properties
       m        = 3e-3;          % Mass of the ball [kg]
       Jf       = 1.581e-3;      % Moment of inertia of the frame [kg*m^2]
       Jb       = 5.48e-7;       % Moment of inertia of the ball [kg*m^2]
       g        = [0;9.81;0];    % Gravitational ecceleration [m*s^2]
       R_b      = 16.55e-3;      % Radius of ball [m]
       r_f      = 12.5e-3;       % Distance between the plates [m]
       Rf       = 0.1;           % Radius of frame in meters [m]
       R;                        % Effective radius of the ball [m]
       k_hat    = [0;0;1];       % Z-directional vector
       B        = [1;0];         % Coupling matrix
       B_an     = [0 1];         % Annihilator matrix
       a        = 0.1095;        % Constant for creating butterfly-frame
       b        = 0.0405;        % Scalar for creating butterfly-frame
       c        = 0.49;          % Scalar for VHC
       shape    = 'butterfly';   % Choose shape of frame
       phiApprox;                % Approximation of phi
       I        = eye(3);        % 3x3 Identity matrix
       Q        = [0 1 0;        % Matrix for z'=Qz
                  -1 0 0;
                   0 0 0];
   end

   methods
       %% Constructor
       function obj = ButterflyRobot()
           obj.R = sqrt(obj.R_b^2 - obj.r_f^2);
           N = 100;
           phi = linspace(0,2*pi,N);
           g = zeros(1,N);
           for i=1:N
```

```
35                  g(i) = getGs(obj,phi(i));
36              end
37              vphi = mod(g,2*pi);
38              varphi = unwrap(vphi);
39              obj.phiApprox = spline(varphi,phi);
40          end
41
42          %% Alpha, Beta, Gamma
43          function [t,x] = simABG(obj,t0,tEnd,x0)
44              options = odeset('RelTol', 1e-5, 'AbsTol', 1e-6);
45              tspan = linspace(t0,tEnd,500);
46              [t,x] = ode23(@(t,y)ABG_EOM(obj,y),tspan,x0,options);
47          end
48          function dx = ABG_EOM(obj,x)
49              vphi = x(1);
50              Dvphi = x(2);
51              [alpha,beta,gamma] = getABG(obj,vphi);
52              DDvphi = -(beta*Dvphi^2 + gamma)/alpha;
53              dx = [Dvphi;DDvphi];
54          end
55
56          %% Butterfly Robot
57          function [t,x,Fn,Fs,u] = simBR(obj,t0,tEnd,x0)
58              options = odeset('RelTol', 1e-5, 'AbsTol', 1e-6);
59              tSteps = linspace(t0,tEnd,500);
60              [t,x] = ode23(@(t,y)Butterfly_EOM(obj,y),tSteps,x0,options);
61              n = length(t);
62              u = zeros(n,1);
63              for i=1:n
64                  u(i) = getActuatorInput(obj,x(i,:)');
65              end
66              Fn = zeros(n,1);
67              Fs = zeros(n,1);
68              for i=1:n
69                  [Fn(i),Fs(i)] = getForces(obj,x(i,:)');
70              end
71          end
72          function dx = Butterfly_EOM(obj,x)
```

```
            [M,C,G] = getMCG(obj,x);
            u = getActuatorInput(obj,x);
            Dq = x(3:4);
            DDq = M\(-C*Dq - G + obj.B*u);
            dx = [Dq;DDq];
        end
        function u = getActuatorInput(obj,x)
            % Generalized coordinates
            theta = x(1); Dtheta = x(3);
            varphi = x(2); Dvarphi = x(4);

            % VHC
            Theta = varphi - obj.c*sin(2*varphi);
            DTheta = 1 - 2*obj.c*cos(2*varphi);
            DDTheta = 4*obj.c*sin(2*varphi);

            % Dynamics
            [M,C,G] = getMCG(obj,x);

            % Reduced dynamics
            [alpha,beta,gamma] = getABG(obj,varphi);

            % Acceleration
            DDvarphi = -(beta/alpha)*Dvarphi^2 - (gamma/alpha);

            % Actuation
            u = M(1,:)*[DDTheta*Dvarphi^2 + DTheta*DDvarphi; DDvarphi] ...
                + C(1,:)*[DTheta*Dvarphi; Dvarphi] + G(1);
        end
        function [Fn,Fs] = getForces(obj,x)
            Dx = Butterfly_EOM(obj,x);
            Dtheta = Dx(1); DDtheta = Dx(3);
            Dvarphi = Dx(2); DDvarphi = Dx(4);

            x_mod = [x(1); x(2); 0; 0];
            [~,rhoVec,tau,n,kappa,Ds,DDs,Pi,~] = getVariables(obj,x_mod);

            % Normal force
```

```
111             Fn = obj.m*(Pi*n)'*(2*Pi*n*Ds*Dtheta*Dvarphi...
112                 + Pi*kappa*Ds^(2)*Dvarphi^(2)...
113                 + cross(DDtheta*obj.k_hat,Pi*rhoVec) + obj.g);
114             % Friction force
115             Fs = obj.m*(Pi*tau)'*(Pi*kappa*Ds^(2)*Dvarphi^(2)...
116                 + Pi*tau*DDs*Dvarphi^(2) + Pi*tau*Ds*DDvarphi...
117                 + cross(DDtheta*obj.k_hat,Pi*rhoVec) + obj.g);
118         end
119
120         %% Shared Functions
121         function [M,C,G] = getMCG(obj,x)
122             % Parameters
123             q1=x(1); q2=x(2); Dq1=x(3); Dq2=x(4);
124             m=obj.m; Jf=obj.Jf; Jb=obj.Jb; g=obj.g; R=obj.R;
125             k_hat=obj.k_hat; B=obj.B; a=obj.a; b=obj.b;
126             [rho,rhoVec,tau,~,kappa,Ds,DDs,Pi,DPi] = getVariables(obj,x);
127
128             % Make M
129             m11 = Jf + Jb + m*rho^(2);
130             m12 = (dot(m*k_hat,cross(rhoVec,tau)) - (Jb/R))*Ds;
131             m21 = m12;
132             m22 = (m + (Jb/R^(2)))*Ds^(2);
133             M   = [m11,m12;m21,m22];
134
135             % Make C
136             c11 = dot(m*Ds*rhoVec,tau)*Dq2;
137             c12 = dot(m*Ds*rhoVec,tau)*Dq1 + ((dot(m*k_hat,cross(rhoVec,
    tau)) -...
138                 (Jb/R))*DDs + dot(m*k_hat,cross(rhoVec,kappa))*Ds^(2))*Dq2
    ;
139             c21 = -dot(m*Ds*rhoVec,tau)*Dq1;
140             c22 = (m + (Jb/R^(2)))*Ds*DDs*Dq2;
141             C   = [c11,c12;c21,c22];
142
143             % Make G
144             g1 = dot(m*g,DPi*rhoVec);
145             g2 = dot(m*g,Pi*tau*Ds);
146             G  = [g1;g2];
```

```
147         end
148         function [rho,rhoVec,tau,n,kappa,Ds,DDs,Pi,DPi] = getVariables(obj
    ,x)
149             % General coordinates and angles
150             theta = x(1);
151             varphi = x(2);
152             phi = ppval(obj.phiApprox, varphi);
153             [alpha,Dalpha,DDalpha] = getALPHAs(obj,phi);
154 
155             % Vectors needed for calculation
156             [deltaVec,DdeltaVec,DDdeltaVec] = getDeltaVecs(obj,phi);
157 
158             % Tangens vectors
159             tau = DdeltaVec/norm(DdeltaVec);
160 
161             % Normal vectors
162             n = [sin(alpha); cos(alpha); 0];
163             Dn = [cos(alpha)*Dalpha; -sin(alpha)*Dalpha; 0];
164             DDn = [-sin(alpha)*Dalpha^2 + cos(alpha)*DDalpha; -cos(alpha)*
    Dalpha^2 - sin(alpha)*DDalpha; 0];
165 
166             % Curvature vector
167             kappa_f = norm(cross(DdeltaVec,DDdeltaVec))/((norm(DdeltaVec))
    ^3);
168             kappa_const = kappa_f/(1 - obj.R*kappa_f);
169             kappa = kappa_const*n;
170 
171             % Vector from origo to intersectionpoint: ball/frame
172             rhoVec = deltaVec + obj.R*n;
173             rho = norm(rhoVec);
174 
175             % Arc length, differentiated once and twice
176             [~,Dg,DDg] = getGs(obj,phi);
177             h = DdeltaVec + obj.R*Dn;
178             Dh = DDdeltaVec + obj.R*DDn;
179             Ds = (norm(h))*(1/Dg);
180             DDs = (((dot(h,Dh)/norm(h))*(1/Dg)) - ((norm(h))*(DDg/Dg^2)))
    *(1/Dg);
```

```
181
182         % Rotation matrix(RM) and differentiated RM
183         Pi = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0
    1];
184         DPi = [-sin(theta) -cos(theta) 0; cos(theta) -sin(theta) 0; 0
    0 0];
185     end
186     function [deltaVec,DdeltaVec,DDdeltaVec] = getDeltaVecs(obj,phi)
187         [delta,Ddelta,DDdelta] = getDELTAs(obj,phi);
188         z = [sin(phi); cos(phi); 0];
189         deltaVec = delta*z;
190         DdeltaVec = (Ddelta*obj.I + delta*obj.Q)*z;
191         DDdeltaVec = (DDdelta*obj.I + 2*Ddelta*obj.Q - delta*obj.I)*z;
192     end
193     function [alpha,beta,gamma] = getABG(obj,varphi)
194         [Phi,DPhi,DDPhi] = getPhis(obj,varphi);
195         Dq = [Phi;DPhi];
196         [M,C,G] = getMCG(obj,Dq);
197         alpha = obj.B_an*M*DPhi;
198         beta = obj.B_an*(C*DPhi + M*DDPhi);
199         gamma = obj.B_an*G;
200     end
201     function [Phi,DPhi,DDPhi] = getPhis(obj,varphi)
202         [Theta,DTheta,DDTheta] = getVHC(obj,varphi);
203         Phi = [Theta; varphi];
204         DPhi = [DTheta; 1];
205         DDPhi = [DDTheta; 0];
206     end
207     function [Theta,DTheta,DDTheta] = getVHC(obj,varphi)
208        Theta = varphi - obj.c*sin(2*varphi);
209        DTheta = 1 - 2*obj.c*cos(2*varphi);
210        DDTheta = 4*obj.c*sin(2*varphi);
211     end
212
213     %% Symbolic Functions
214     function [delta,Ddelta,DDdelta,DDDdelta] = getDELTAs(obj,phi)
215         a = obj.a; b = obj.b;
216         t2 = phi.*2.0;
```

```
217             t3 = cos(t2);
218             delta = a-b.*t3;
219             t4 = sin(t2);
220             Ddelta = b.*t4.*2.0;
221             DDdelta = b.*t3.*4.0;
222             DDDdelta = b.*t4.*-8.0;
223             if strcmpi(obj.shape,'circle')
224                 delta = obj.Rf;
225                 Ddelta = 0;
226                 DDdelta = 0;
227                 DDDdelta = 0;
228             end
229         end
230         function [alpha,Dalpha,DDalpha] = getALPHAs(obj,phi)
231             [delta,Ddelta,DDdelta,DDDdelta] = getDELTAs(obj,phi);
232             t2 = delta;
233             t3 = sin(phi);
234             t4 = Ddelta;
235             t5 = cos(phi);
236             alpha = atan((t2.*t3-t4.*t5)./(t2.*t5+t3.*t4));
237             if (phi > pi/2 && phi < 3*pi/2)
238                 alpha = alpha + pi;
239             end
240             if (phi > 3*pi/2)
241                 alpha = alpha + 2*pi;
242             end
243             t6 = t4.^2;
244             t7 = t2.^2;
245             t8 = t6+t7;
246             t9 = DDdelta;
247             Dalpha = (t6.*2.0+t7-t2.*t9)./t8;
248             t10 = DDDdelta;
249             DDalpha = -1.0./t8.^2.*(t2.*t4.*t9.^2.*-2.0+t2.*t4.*t6.*2.0+t2
    .*t6.*t10+t2.*t7.*t10+t4.*t6.*t9-t4.*t7.*t9.*3.0);
250         end
251         function [g,Dg,DDg] = getGs(obj,phi)
252             R = obj.R;
253             [delta,Ddelta,DDdelta,~] = getDELTAs(obj,phi);
```

```
254             [alpha,Dalpha,DDalpha] = getALPHAs(obj,phi);
255             t2 = delta;
256             t3 = alpha;
257             g = atan2((R.*sin(t3)+t2.*sin(phi)),(R.*cos(t3)+t2.*cos(phi)))
    ;
258             if (nargout > 1)
259                 t4 = t2.^2;
260                 t5 = R.^2;
261                 t6 = phi-t3;
262                 t7 = cos(t6);
263                 t8 = Dalpha;
264                 t9 = R.*t2.*t7.*2.0;
265                 t10 = t4+t5+t9;
266                 t11 = 1.0./t10;
267                 t12 = Ddelta;
268                 t13 = sin(t6);
269                 t14 = DDalpha;
270                 t15 = t2.*t12.*2.0;
271                 t16 = R.*t7.*t12.*2.0;
272                 t17 = t5.*t8;
273                 t18 = R.*t12.*t13;
274                 t19 = R.*t2.*t7;
275                 t20 = R.*t2.*t7.*t8;
276                 t21 = t4+t17+t18+t19+t20;
277                 Dg = t11.*t21;
278                 DDg = t11.*(t15+t16+t5.*t14-R.*t2.*t13+R.*t13.*DDdelta+R.*
    t2.*t7.*t14+R.*t2.*t8.^2.*t13)-1.0./t10.^2.*t21.*(t15+t16+R.*t2.*t13
    .*(t8-1.0).*2.0);
279             end
280         end
281
282     end
283 end
```

generateDelta.mlx

This is a symbolic script that is used to calculate complicated mathematical expressions and turning the answers into functions that take desired input. The following is created to find $\delta(\phi)$ and its derivatives.

```
%% Create function: getPHIs(obj,phi)
%    This function was generated by the Symbolic Math Toolbox version 7.2.
%    02-Nov-2017 16:51:23

syms phi, syms R b c

delta = b - c*cos(2*phi);
Ddelta = simplify(diff(delta,phi));
DDdelta = simplify(diff(delta,phi,2));
DDDdelta = simplify(diff(delta,phi,3));

getDELTAs = matlabFunction(delta,Ddelta,DDdelta,DDDdelta,...
    'File','getDELTAs','Outputs',{'delta','Ddelta','DDdelta','DDDdelta'});
```

generateAlpha.mlx

This is a symbolic script that is used to calculate complicated mathematical expressions and turning the answers into functions that take desired input. The following is created to find $\alpha(\phi)$ and its derivatives.

```
%% Create function: getALPHAs(obj,phi)
%    This function was generated by the Symbolic Math Toolbox version 7.2.
%    02-Nov-2017 17:21:35

syms phi
syms delta(phi) Ddelta(phi) DDdelta(phi) DDDdelta(phi)

funalpha = atan((delta*sin(phi) - Ddelta*cos(phi))/(delta*cos(phi) +
    Ddelta*sin(phi)));
funDalpha = simplify(diff(funalpha,phi));
funDDalpha = simplify(diff(funalpha,phi,2));

alpha = simplify(funalpha,'Steps',100);
Dalpha = subs(funDalpha, [diff(delta), diff(Ddelta)], [Ddelta, DDdelta]);
DDalpha = subs(funDDalpha, [diff(delta), diff(delta,2), diff(Ddelta), diff
    (Ddelta,2), diff(DDdelta)], ...
    [Ddelta, DDdelta, DDdelta, DDDdelta, DDDdelta]);

getALPHAs = matlabFunction(alpha,Dalpha,DDalpha,...
    'File','getALPHAs','Outputs',{'alpha','Dalpha','DDalpha'});
```

generateExpr.mlx

This is a symbolic script that is used to calculate complicated mathematical expressions and turning the answers into functions that take desired input. The following is created to find $g(\phi)$ and its derivatives.

```
%% Create function: getGs(obj,phi)
%    This function was generated by the Symbolic Math Toolbox version 7.2.
%    02-Nov-2017 17:52:04

syms phi, syms R
syms delta(phi) Ddelta(phi) DDdelta(phi)
syms alpha(phi) Dalpha(phi) DDalpha(phi)

fung = atan((delta*sin(phi) + R*sin(alpha))/(delta*cos(phi) + R*cos(alpha)
    ));
funDg = simplify(diff(fung,phi),'Steps',100);
funDDg = simplify(diff(funDg,phi),'Steps',100);

g = simplify(fung,'Steps',100);
Dg = subs(funDg, [diff(delta), diff(alpha)], [Ddelta, Dalpha]);
DDg = subs(funDDg, [diff(delta), diff(delta,2), diff(alpha), diff(alpha,2)
    ], ...
    [Ddelta, DDdelta, Dalpha, DDalpha]);

getGs = matlabFunction(g,Dg,DDg,...
    'File','getGs','Outputs',{'g','Dg','DDg'});
```

runCircleRobot.m

This script is identical to the main file (runButterflyRobot.m), except for that it runs the functions through a different class, namely *CircleRobot* class.

```
%% Author: Oskar Lund, 29.10.17

%% Description
% This script simulates the overall behaviour of the
% butterfly robot, but with the butterfly shape
% switched out with a circular shape

%% Define the object BR of class ButterflyRobot
CR = CircleRobot;

%% Choose which program to run: ABG = 0, BR = 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
program = 1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if (program == 0)
        %% Simulate alpha, beta, gamma

    t0_   = 0;  % Intial time
    tEnd_ = 10;  % Final time
    x0_   = [0; 1]; % Initial conditions ([varphi0;Dvarphi0])
    [t_,x_] = CR.simABG(t0_,tEnd_,x0_);

elseif (program == 1)
        %% Simulate the full circle system

    t0    = 0;  % Intial time
    tEnd = 5;  % Final time
    x0    = [0; pi/2; 0; 0]; % Initial conditions ([theta0;varphi0;Dtheta0;
    Dvarphi0])
    [t,x] = CR.simCR(t0,tEnd,x0);

end
```

CircleRobot.mlx

This script is identical to the class used by the main file (ButterflyRobot.m), except for that the general dynamics have been replaced with the specific dynamics of a circle.

```
classdef CircleRobot
    % Dynamics of Circle robot

    properties
        m       = 3e-3;         % Mass of the ball [kg]
        Jf      = 1.581e-3;     % Moment of inertia of the frame [kg*m^2]
        Jb      = 5.48e-7;      % Moment of inertia of the ball [kg*m^2]
        g       = [0;9.81;0];   % Gravitational ecceleration [m*s^2]
        R_b     = 16.55e-3;     % Radius of ball [m]
        r_f     = 12.5e-3;      % Distance between the plates [m]
        Rf      = 0.1;          % Radius of frame in meters [m]
        R;                      % Effective radius of the ball [m]
        Rd;                     % Radius from origo to ball's center [m]
        c       = 0.49;         % Scalar for VHC
        B_an    = [0 1];        % Annihilator matrix
        phiApprox;              % Approximation of phi
    end

    methods
        %% Constructor
        function obj = CircleRobot()
            obj.R = sqrt(obj.R_b^2 - obj.r_f^2);
            obj.Rd = obj.R + obj.Rf;
            N = 100;
            phi = linspace(0,2*pi,N);
            g = zeros(1,N);
            for i=1:N
                [g(i),~,~] = getGs(obj,phi(i));
            end
            varphi = mod(g,2*pi);
            obj.phiApprox = spline(varphi,phi);
        end

        %% Alpha, Beta, Gamma
```

```
35          function [t,x] = simABG(obj,t0,tEnd,x0)
36              options = odeset('RelTol', 1e-5, 'AbsTol', 1e-6);
37              tspan = linspace(t0,tEnd,1000);
38              [t,x] = ode23(@(t,y)ABG_EOM(obj,y),tspan,x0,options);
39          end
40          function dx = ABG_EOM(obj,x)
41              vphi = x(1);
42              Dvphi = x(2);
43              [alpha,beta,gamma] = getABG(obj,vphi);
44              DDvphi = -(beta*Dvphi^2 + gamma)/alpha;
45              dx = [Dvphi;DDvphi];
46          end
47          function [alpha,beta,gamma] = getABG(obj,varphi)
48              [Phi,DPhi,DDPhi] = getPhis(obj,varphi);
49              Dq = [Phi;DPhi];
50              [M,C,G] = getMCG(obj,Dq);
51              alpha = obj.B_an*M*DPhi;
52              beta = obj.B_an*(C*DPhi + M*DDPhi);
53              gamma = obj.B_an*G;
54          end
55          function [Phi,DPhi,DDPhi] = getPhis(obj,varphi)
56              [Theta,DTheta,DDTheta] = getVHC(obj,varphi);
57              Phi = [Theta; varphi];
58              DPhi = [DTheta; 1];
59              DDPhi = [DDTheta; 0];
60          end
61          function [Theta,DTheta,DDTheta] = getVHC(obj,varphi)
62             Theta = varphi - obj.c*sin(2*varphi);
63             DTheta = 1 - 2*obj.c*cos(2*varphi);
64             DDTheta = 4*obj.c*sin(2*varphi);
65          end
66
67          %% Circle Robot
68          function [t,x] = simCR(obj,t0,tEnd,x0)
69              options = odeset('RelTol', 1e-5, 'AbsTol', 1e-6);
70              tSteps = linspace(t0,tEnd,100);
71              [t,x] = ode23(@(t,y)Butterfly_EOM(obj,y),tSteps,x0,options);
72          end
```

```
73          function dx = Butterfly_EOM(obj,x)
74              Dq = x(3:4);
75              [M,C,G] = getMCG(obj,x);
76              DDq = M\(-C*Dq - G);
77              dx = [Dq;DDq];
78          end
79
80          %% Shared Functions
81          function [M,C,G] = getMCG(obj,x)
82              % Parameters
83              q1=x(1); q2=x(2); Dq1=x(3); Dq2=x(4);
84              Jf = obj.Jf; Jb = obj.Jb;
85              m = obj.m; g = obj.g;
86              R = obj.R; Rd = obj.Rd;
87              [z,Dz,Pi,DPi] = getVariables(obj,q1,q2);
88
89              % Make M
90              m11 = Jf + Jb + m*Rd^(2);
91              m12 = -(m*Rd^(2) + (Jb/R)*Rd);
92              m21 = m12;
93              m22 = (m + (Jb/R^(2)))*Rd^2;
94              M   = [m11,m12;m21,m22];
95
96              % Make C
97              c11 = 0;
98              c12 = 0;
99              c21 = 0;
100             c22 = 0;
101             C   = [c11,c12;c21,c22];
102
103             % Make G
104             g1 = dot(m*g,DPi*Rd*z);
105             g2 = dot(m*g,Pi*Rd*Dz);
106             G  = [g1;g2];
107         end
108         function [z,Dz,Pi,DPi] = getVariables(obj,theta,varphi)
109             phi = ppval(obj.phiApprox, varphi);
```

```
110             Pi = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0
    1];
111             DPi = [-sin(theta) -cos(theta) 0; cos(theta) -sin(theta) 0; 0
    0 0];
112             z = [sin(phi); cos(phi); 0];
113             Dz = [cos(phi); -sin(phi); 0];
114         end
115
116         %% Symbolic Functions
117         function [delta,Ddelta,DDdelta,DDDdelta] = getDELTAs(obj,phi)...
118         function [alpha,Dalpha,DDalpha] = getALPHAs(obj,phi)...
119         function [g,Dg,DDg] = getGs(obj,phi)...
120     end
121 end
```

ValidationExpression.m

This is a script that portraits the difference the assumption $\phi = \varphi$ makes by simulating $\varphi = g(\phi)$ for various instances.

```
%% Author: Oskar Lund, 29.11.17

%% Description
% This script creates plots of the derived general function g(phi).
% There are three shapes (circle, ellipse and butterfly) that are
% simulated to confirm the authenticity of the expression.

%% Parameters
sizeR    = 'small';      % choices: small-> R = 0.01m, big-> R = 10m
shape    = 'butterfly';  % choices: circle, ellipse or butterfly

%% Define angles and variables
N = 1000;
phi = linspace(0,2*pi,N);
fung = zeros(1,N);
vphi = zeros(1,N);

for i=1:N
    [fung(i),~,~] = getGs(phi(i),sizeR,shape);
    vphi(i) = mod(fung(i),2*pi);
end
varphi = unwrap(vphi);

%% External functions from BR script
function [delta,Ddelta,DDdelta,DDDdelta] = getDELTAs(phi,shape)...
function [alpha,Dalpha,DDalpha] = getALPHAs(phi,shape)...
function [g,Dg,DDg] = getGs(phi,sizeR,shape)...
```

DeriveVHC.m

This script simulates all the sub-parts contributing to the derivation of a valid virtual holonomic constraint.

```
%% Author: Oskar Lund, 03.11.17

%% Description
% This script looks at the reduced dynamics, also called:
% alpha,beta,gamma and simulates phase trajectories and other
% useful information.

%% Parameters
m        = 3e-3;          % Mass of the ball [kg]
Jf       = 1.581e-3;      % Moment of inertia of the frame [kg*m^2]
Jb       = 5.48e-7;       % Moment of inertia of the ball [kg*m^2]
g        = [0;9.81;0];    % Gravitational ecceleration [m*s^2]
R_b      = 16.55e-3;      % Radius of ball [m]
r_f      = 12.5e-3;       % Distance between the plates [m]
R = sqrt(R_b^2 - r_f^2);% Effective radius of the ball [m]
k_hat    = [0;0;1];       % Z-directional vector
B        = [1;0];         % Coupling matrix
B_an     = [0 1];         % Annihilator matrix
c        = 0.49;          % Scalar for VHC

%% Define angles and variables
N = 1000;
phi = linspace(0,2*pi,N);
fung = zeros(1,N);
varphi = zeros(1,N);
VHC = zeros(1,N);
VHC_check = zeros(1,N);
VHC_DTheta = zeros(1,N);
VHC_DDTheta = zeros(1,N);
Alpha = zeros(1,N);
Beta = zeros(1,N);
Gamma = zeros(1,N);
DGamma = zeros(1,N);

```

```
for i=1:N
    %% Define varphi
    [fung(i),Dg,DDg] = getGs(phi(i));
    varphi(i) = mod(fung(i),2*pi);

    %% Define variables
    [alpha,Dalpha,DDalpha] = getALPHAs(phi(i));
    [deltaVec,DdeltaVec,DDdeltaVec] = getDeltaVecs(phi(i));
    tau = DdeltaVec/norm(DdeltaVec);
    n = [sin(alpha); cos(alpha); 0];
    Dn = [cos(alpha)*Dalpha; -sin(alpha)*Dalpha; 0];
    DDn = [-sin(alpha)*Dalpha^2 + cos(alpha)*DDalpha; -cos(alpha)*Dalpha^2
     - sin(alpha)*DDalpha; 0];
    rho = deltaVec + R*n;
    h = DdeltaVec + R*Dn;
    Dh = DDdeltaVec + R*DDn;
    Ds = (norm(h))*(1/Dg);
    DDs = (((dot(h,Dh)/norm(h))*(1/Dg)) - ((norm(h))*(DDg/Dg^2)))*(1/Dg);

    %% Define VHC
    Theta = varphi(i) - c*sin(2*varphi(i));
    DTheta = 1 - 2*c*cos(2*varphi(i));
    DDTheta = 4*c*sin(2*varphi(i));
    VHC(i) = Theta;
    VHC_check(i) = (-Ds*(m + Jb/R^2))/(dot(m*k_hat,cross(rho,tau)) - Jb/R)
    ;
    VHC_DTheta(i) = DTheta;
    VHC_DDTheta(i) = DDTheta;

    %% Rotation matrix
    Pi = [cos(Theta) -sin(Theta) 0; sin(Theta) cos(Theta) 0; 0 0 1];
    DPi = [-sin(Theta) -cos(Theta) 0; cos(Theta) -sin(Theta) 0; 0 0 0];

    %% Define alpha, beta, gamma
    Alpha(i) = Ds*(dot(m*k_hat,cross(rho,tau)) - Jb/R)*DTheta + Ds^(2)*(m
    + Jb/R^2);
    Beta(i) = Ds*(dot(m*k_hat,cross(rho,tau)) - Jb/R)*DDTheta - dot(m*Ds'*
    tau,rho*DTheta.^(2)) + (m + Jb/R^2)*Ds*DDs;
```

```
    Gamma(i) = dot(m*Ds*g,Pi*tau);
    DGamma(i) = m*DDs*dot(g,Pi*tau) + m*Ds*dot(g,(DPi*DTheta*tau + Pi*((
    DDdeltaVec/norm(DdeltaVec))-((DdeltaVec*dot(DdeltaVec,DDdeltaVec))/(
    norm(DDdeltaVec)^3)))));
end

%% Studying equilibrium points and checking for asymptotes
asymCheckOne = Gamma./Alpha;
asymCheckTwo = Beta./Alpha;

[deltaVecEq1,DdeltaVecEq1,DDdeltaVecEq1] = getDeltaVecs(0);
[deltaVecEq2,DdeltaVecEq2,DDdeltaVecEq2] = getDeltaVecs(pi/2);
[alphaEq1,DalphaEq1,~] = getALPHAs(0);
[alphaEq2,DalphaEq2,~] = getALPHAs(pi/2);
[~,DgEq1,~] = getGs(0);
[~,DgEq2,~] = getGs(pi/2);

rhoEqOne = deltaVecEq1 + R*[sin(alphaEq1); cos(alphaEq1); 0];
rhoEqTwo = deltaVecEq2 + R*[sin(alphaEq2); cos(alphaEq2); 0];
tauEqOne = DdeltaVecEq1/norm(DdeltaVecEq1);
tauEqTwo = DdeltaVecEq2/norm(DdeltaVecEq2);
DsEqOne = (norm(DdeltaVecEq1 + R*[cos(alphaEq1)*DalphaEq1; -sin(alphaEq1)*
    DalphaEq1; 0]))*(1/DgEq1);
DsEqTwo = (norm(DdeltaVecEq2 + R*[cos(alphaEq2)*DalphaEq2; -sin(alphaEq2)*
    DalphaEq2; 0]))*(1/DgEq2);

aOptions = (-2:2);
for i = aOptions
    DThetaEqOne = 1 - 2*i*cos(2*0);
    DThetaEqTwo = 1 - 2*i*cos(2*(pi/2));
    AlphaEqPointOne(i+3) = DsEqOne.*(dot(m*k_hat,cross(rhoEqOne,tauEqOne))
     - Jb/R).*DThetaEqOne + DsEqOne.^(2).*(m + Jb/R^2);
    AlphaEqPointTwo(i+3) = DsEqTwo.*(dot(m*k_hat,cross(rhoEqTwo,tauEqTwo))
     - Jb/R).*DThetaEqTwo + DsEqTwo.^(2).*(m + Jb/R^2);
end

AlphaNull = zeros(length(aOptions));
P1 = InterX([aOptions;AlphaEqPointOne],[aOptions;AlphaEqPointTwo]);
```

```
P2 = InterX([aOptions;AlphaNull],[aOptions;AlphaEqPointOne]);
P3 = InterX([aOptions;AlphaNull],[aOptions;AlphaEqPointTwo]);
Px = [P1(1); P2(1); P3(1)];
Py = [P1(2); P2(2); P3(2)];

lambdaSignDecider = DGamma./Alpha;

function [deltaVec,DdeltaVec,DDdeltaVec] = getDeltaVecs(phi)
    [delta,Ddelta,DDdelta,~] = getDELTAs(phi);
    z = [sin(phi); cos(phi); 0];
    Q = [0 1 0; -1 0 0; 0 0 0];
    I = eye(3);
    deltaVec = delta*z;
    DdeltaVec = (Ddelta*I + delta*Q)*z;
    DDdeltaVec = (DDdelta*I + 2*Ddelta*Q - delta*I)*z;
end

%% External functions from BR script
function [delta,Ddelta,DDdelta,DDDdelta] = getDELTAs(phi)...
function [alpha,Dalpha,DDalpha] = getALPHAs(phi)...
function [g,Dg,DDg] = getGs(phi)...
```