Ignacio Pons Alcalá

Modelling, motion planning and control for performing feasible rolling motion of a passive disc on a frame of the Butterfly Robot

Master's thesis in Master Thesis
Supervisor: Anton Shiriaev
August 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

MASTER’S FINAL THESIS IN INDUSTRIAL ENGINEERING

Modelling, motion planning and control for performing feasible rolling motion of a passive disc on a frame of the Butterfly Robot

AUTHOR: **Ignacio Pons Alcalá**

NTNU SUPERVISOR : **Anton Shiriaev**

DEPARTMENT OF ENGINEERING CYBERNETICS

UPV SUPERVISOR: **Raúl Simarro Fernández**

DEPARTMENT OF ENGINEERING SISTEMS AND AUTOMATIC

ACADEMIC COURSE: 2019-20

CONTRIBUTION:

The Butterfly Robot is a highly complex and complete example in a field yet very unknown, even in sciences, that are the non-prehensile underactuated systems. A system that because of its physical simplicity allows to understand the entire concept and yet requires a thorough study of alternative approaches for the different difficulties appearing. In previous thesis have been developed mainly the dynamics of the system, how it works, which simplifications have been made and how to develop trajectories. The next step and the main contribution of this thesis was to study and perform the control for the system because of the real behaviour of the system and possible external perturbations . An ambitious problem that shaped the main idea of this thesis.

At the beginning it is been necessary to understand the ideas and concepts exposed by other students and how they arrived to those conclusions including their algorithms to calculate the dynamics and the motion planning for the Butterfly Robot in MATLAB. Using those same algorithms and doing some changes, was possible to define different and more complex trajectories for the system. After that, a thorough research using some studies ,papers and reports was necessary to understand the concept of orbital stability, how could be simplified the control for the system and how to implement it in an algorithm developed in MATLAB. This is what is really challenging, and differences this thesis. The part of the MATLAB code of control has been completely developed by own contribution.

This project would have been completely impossible without the invaluable assistance of my supervisor Anton Shiriaev, an expert in this field who, in this difficult year have guided me and helped to overcome the difficulties not only of the project but also of communication. This thesis has permitted me to expand my thinking with alternative ways to approach a problem and to coordinate different aspects and fields of science. Something very difficult to evaluate.

ABSTRACT:

The master thesis presented below has as objective the analysis of the modelling and control of an underactuated non-prehensile system known as the Butterfly Robot.

The Butterfly Robot consists of a frame/disc with a particular edge shape similar to a butterfly, that is commanded directly just with a rotational movement, and a ball controlled indirectly by the effect of gravity, rolling in the top edge of the frame/disc. A clear example of an underactuated non-prehensile control.

A thorough study of the modelling of the system is done, then followed by developing different motions of feasible rolling, assuming some requirements.

Once the different motions have been settled, the control of the system for each motion is conducted, a step by step method, so that the possible discrepancies between the theorical system and the physical system can be settled.

ACKNOWLEDGEMENTS

“To the professors that transmitted me the pleasure of knowledge,

To my supervisor Anton for his dedication to me”

CONTENTS INDEX

LIST OF SYMBOLS

Symbol	Units	Description
a	-	Constant to describe the shape of the butterfly frame
$A(t)$	-	Variable matrix in the PRDE problem
A_c	-	Parameter matrix in the ARE problem
b	-	Constant to describe the shape of the butterfly frame
$B(q)$	-	Coupling matrix for the input u
$B(t) = b(t)$	-	Variable matrix in the PRDE problem
$B^{\perp}(q)$	-	Annihilator matrix for the input u
B_c	-	Parameter matrix in the ARE problem
c	-	Constant from the VHC
$c = [c_i]$	-	Maximal solution vector coefficients for SEDUMI
$C(q, \dot{q})$	-	Coriolis and centrifugal matrix
d	-	Parameter for the LMI constraints
$\overrightarrow{e^i}$	-	Reference frame
$F = [F_i]$	-	LMI constraints matrix coefficients for SEDUMI
F_n	N	Normal force
F_s	N	Friction force
$g(\phi)$	-	Coupler between ϕ and φ
$\vec{g}$	$m \ s^{-2}$	Gravity constant vector
g_y	-	Variable of the transverse dynamics
$g_{\dot{y}}$	-	Variable of the transverse dynamics
g_w	-	Variable of the transverse dynamics
$G(q)$	-	Gravitational matrix
h	-	Variable to introduce differentiation of s
$I_n = I$	-	Identity matrix size nxn
J_b	$kg \ m^2$	Mass moment of inertia of the ball
J_f	$kg \ m^2$	Mass moment of inertia of the frame
$\hat{k}$	-	Unit vector in z-direction
$K(t)$	-	Matrix gain in the feedback control law
$\mathcal{K}$	J	Kinetic energy of the system
$\mathcal{K}_b$	J	Kinetic energy of the ball
$\mathcal{K}_f$	J	Kinetic energy of the frame
L	-	Division of the discrete system
$L(\varphi)$	-	Parameter matrix to compute the actuator input
$\mathcal{L}$	J	Lagrangian symbol
m_b	kg	Mass of the ball
m_f	kg	Mass of the frame
M	-	Grade of the trigonometric polynomial approximation

Symbol	Units	Description
$M(q)$	-	Inertia matrix
$\vec{n}$	-	Unit normal vector for ball
$\vec{n}_f$	-	Unit normal vector for frame
$N(\varphi,\dot{\varphi})$	-	Parameter matrix to compute the actuator input
p	-	Total number of coefficients in the SDP problem
$P(t)$	-	T-periodic transform matrix function
$\mathcal{P}$	J	Potential energy of the system
$\mathcal{P}_b$	J	Potential energy of the ball
$\mathcal{P}_f$	J	Potential energy of the frame
q	-	Vector of generalized coordinates
$q_*(t)$	-	Vector of Generalized coordinates on solution trajectory
$Q(t)$	-	Variable matrix in the PRDE problem
Q_c	-	Parameter matrix in the ARE problem
$\overrightarrow{r_b}$	m	Vector of the position of the ball
r_f	m	Half of the distance between both frame plates
$\overrightarrow{r_f}$	m	Vector of the position of the frame
R	m	Effective radius of the ball
$R(t)=R^+(t)$	-	Solution of the PRDE
R_c^+	-	Solution of the ARE
$R_{M,L}^+(t)$	-	Solution of the SDP problem
$\bar{R}(t)$	-	Trigonometric polynomial approximation of $R(t)$
R_0	-	Matrix of the trigonometric polynomial approximation
R_b	m	Real radius of the ball
$R_{a,k}$	-	Matrix of the trigonometric polynomial approximation
$R_{b,k}$	-	Matrix of the trigonometric polynomial approximation
$\mathfrak{R}$	-	Periodic Riccati Differential Equation (PRDE)
$\mathfrak{R}_c$	-	Algorithmic Riccati Equation (ARE)
$s=s(\varphi)$	m	Arclength described by the centre of the ball
$s_f=s_f(\phi)$	m	Arclength described by the contact between ball and frame
$S(R,t)=[S_j]$	-	Linear matrix inequalities (LMI) of the SDP problem
$S(\cdot)$	-	Moving Poincaré Section
t	s	Time
t_j	s	Discretional time
T	s	Period
u	$N\ m$	Actuator input or torque
$\overrightarrow{v_b}$	$m\ s^{-1}$	Velocity vector of the ball
$\overrightarrow{v_f}$	$m\ s^{-1}$	Velocity vector of the frame
w	-	Virtual control input

Symbol	Units	Description
$\vec{w}_b$	rad s^{-1}	Angular velocity of the ball about its own rotation
$\vec{w}_{b_{center}}$	rad s^{-1}	Angular velocity of the ball about its own rotation on ideal curve
$\overrightarrow{w_f}$	rad s^{-1}	Angular velocity of the frame
x'	m	x-coordinate of the frame relative to its own reference frame
$x_b = x$	m	x-coordinate of the ball relative to the reference frame
x_f	m	x-coordinate of the frame relative to the reference frame
$x_\perp(t)$	-	Transverse coordiantes
$y = [y_i]$	-	Coefficient solution by SEDUMI of the SDP problem
y'	m	y-coordinate of the frame relative to its own reference frame
$y_b = y$	m	y-coordinate of the ball relative to the reference frame
y_f	m	y-coordinate of the frame relative to the reference frame
z	-	Third transversal coordinate
$\alpha(\phi)$	rad	Angle between the tangent of the frame and the x'-axis
$\alpha(\varphi)$	-	Parameter of the $\alpha\beta\gamma$ equation
$\beta(\varphi)$	-	Parameter of the $\alpha\beta\gamma$ equation
$\gamma(\varphi)$	-	Parameter of the $\alpha\beta\gamma$ equation
$\mathit{\Gamma}(t)$	-	Variable matrix in the PRDE problem
$\mathit{\Gamma}_c$	-	Parameter matrix in the ARE problem
$\vec{\delta} = \vec{\delta}(\phi)$	m	Position of the contact point between frame and ball
θ_b	rad	Angle of the ball in the reference frame
$\theta_f = \theta$	rad	Angle of the frame in the reference frame
$\theta_* = \Theta(\varphi_*)$	rad	Generalized coordinate in the solution trajectory
$\Theta(\varphi)$	-	VHC- Virtual holonomic constraint
κ	-	Curvature of the trajectory of the centre of the ball
$\vec{\kappa}$	-	Curvature vector of the centre of the ball
κ_f	-	Curvature of the frame
$\vec{\kappa}_f$	-	Curvature vector of the frame
$\Pi(\theta)$	-	Three dimensional rotation matrix
$\vec{\rho} = \vec{\rho}(\varphi)$	m	Position of the ball
$\vec{\tau}$	-	Unit tangent vector of the trajectory of the centre of the ball
$\vec{\tau}_f$	-	Unit tangent vector of the frame
ϕ	rad	Angle between the y'-axis and the contact point of the ball
$\Phi(\varphi)$	-	Variable to describe the reduced dynamics
$\jmath$	-	Angle between the y'-axis and the centre of the ball
φ	rad	Rotation of the ball from the body reference frame
φ_*	rad	Angular velocity of the ball
ψ	$rad\ s^{-1}$	Maximal function in the SDP problem
$\psi(t)$	-	Scalar variable to describe the position along the trajectory
ω	$rad\ s^{-1}$	Frequency

1 INTRODUCTION

1.1 PROJECT AIM

Analise how the Butterfly Robot works by the study of its modelling to create new rolling motion planning and develop the required controls so that there is not discrepancies between the real and the theorical behaviour of this motions. For that purpose it is been studied and perfected previous work of the modelling of the system

1.2 BASIS FOR THE THESIS

The final master thesis pretends to evaluate the acquired competences in the studies conducted in the Master in Technological Industrial Engineering with the specialisation in Robotics and Automatic, that are reflected in a developed way in this project, and that are essential to its conclusion.

The thesis has permitted to study the behaviour of the Butterfly Robot case study and to propose new interesting rolling motions not seen until now, with the further development of the control to achieve the same theorical and real behaviour.

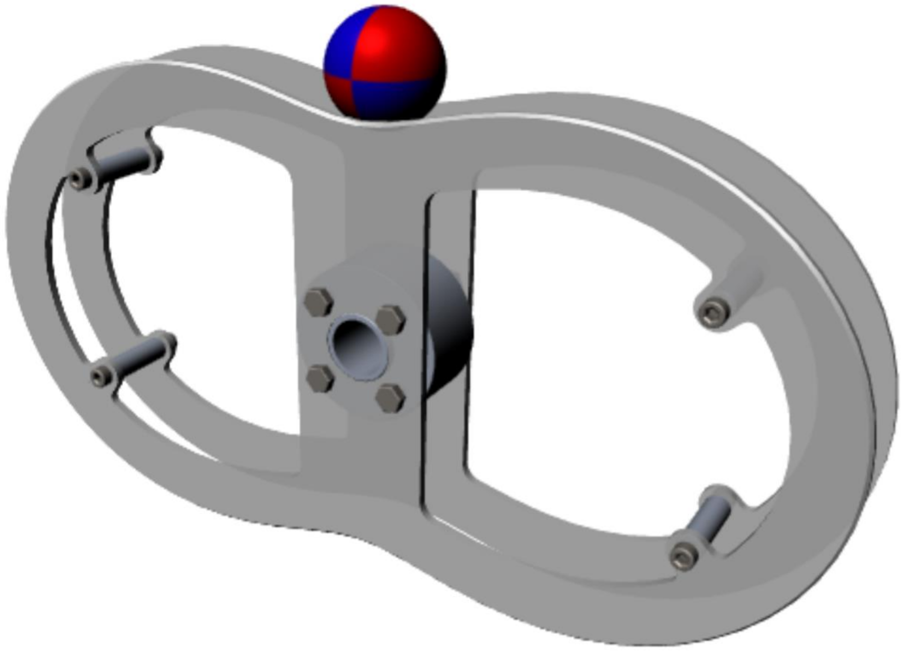
Figure 1.1 The butterfly robot in its equilibrium point – source: [1]

2 THEORETICAL FRAMEWORK

2.1 NON-PREHESILE MANIPULATION ROBOTS

Human interaction with objects is continuous and its manipulation can take place by prehensile or non-prehensile manipulation. Grasp an object is a task humans perform ordinary, a clear example of prehensile-manipulation, but, instead of that could make any other actions such as rolling, pushing, toppling or pulling, all of them examples of non-prehensile manipulations that we perform every day.

In the well-known as forth industrial revolution, robots are acquiring a growing role in the industries and somehow are replacing jobs performed by humans in the past. Robots are becoming more and more close to imitate perfectly physical human motion, however there are still considerable differences, especially when it comes to non-prehensile manipulation. That’s what restricts substantially the applicability of the robots in the real world. In order for robots to approach or overtake human manipulation skill, is important to improve non-prehensile manipulation. For that reason, the studies in this field have increased in the last years and have become in very challenging problems.

Non-prehensile manipulation implies there are more degrees of freedom than actuators that can be controlled directly, what is called un underactuated system. The reason why non-prehensile manipulation problem is so complex is because it cannot be simplified o linearized at any point, it has to be solved in the non-linear fashion.

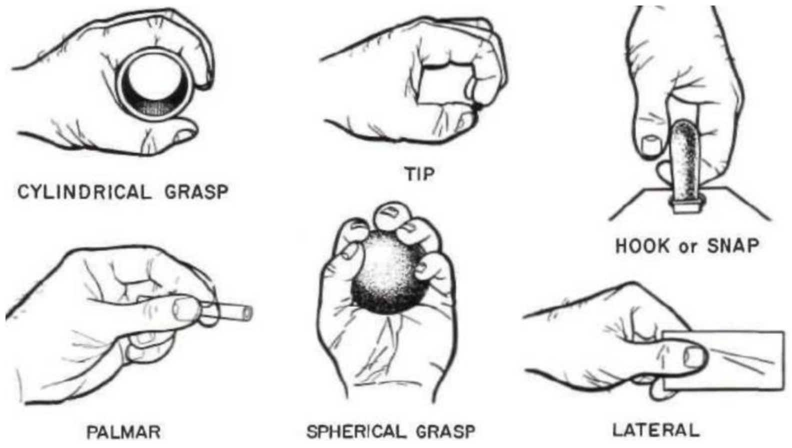

Figure 2.1 Examples prehensile – source: [10]

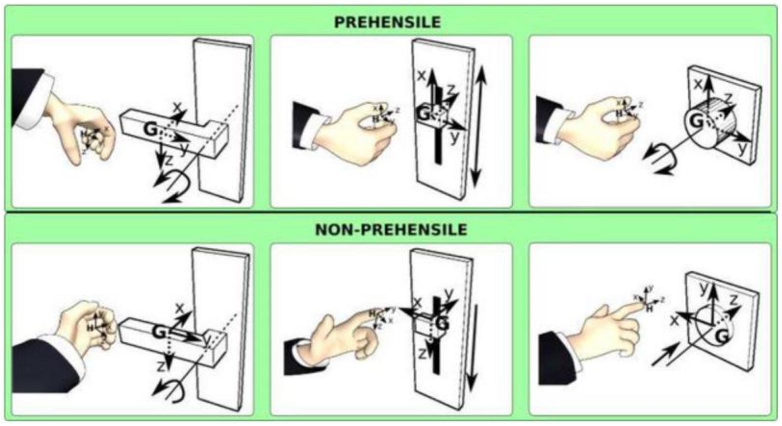

Figure 2.2 Examples prehensile and non-prehensile - source: [10]

2.2 THE BUTTERFLY ROBOT

This robot was originally conceived by Kevin Lynch at Northwestern University and is a perfect example of non-prehensile manipulation and an underactuated system. Further work and experiments developed by Maksim Surov, Anton Shiriaev and their team [1] it is been an illustrative example of progress in this field and provides the basis for this thesis including its methodology as it is been before by Lund [2] and Vogels [3]. The mechanism of the Butterfly Robot is very simple, but is the algorithm that is driving it what is really revolutionary.

The mechanism consists of two parallel identical frame with a particular edge shape similar to a butterfly, that is commanded directly just with a rotational movement (driven by an electrical motor), and a ball controlled indirectly by the effect of gravity, rolling freely in the top edge of the frame.

3 DYNAMICS OF THE BUTTERFLY ROBOT

3.1 METODOLOGY APPLIED

For the realisation of this thesis it is been followed a methodology, described in the diagram of the Figure 3.1, to organizing in a simple way its study and development. It involves the steps to follow with the most important sections.

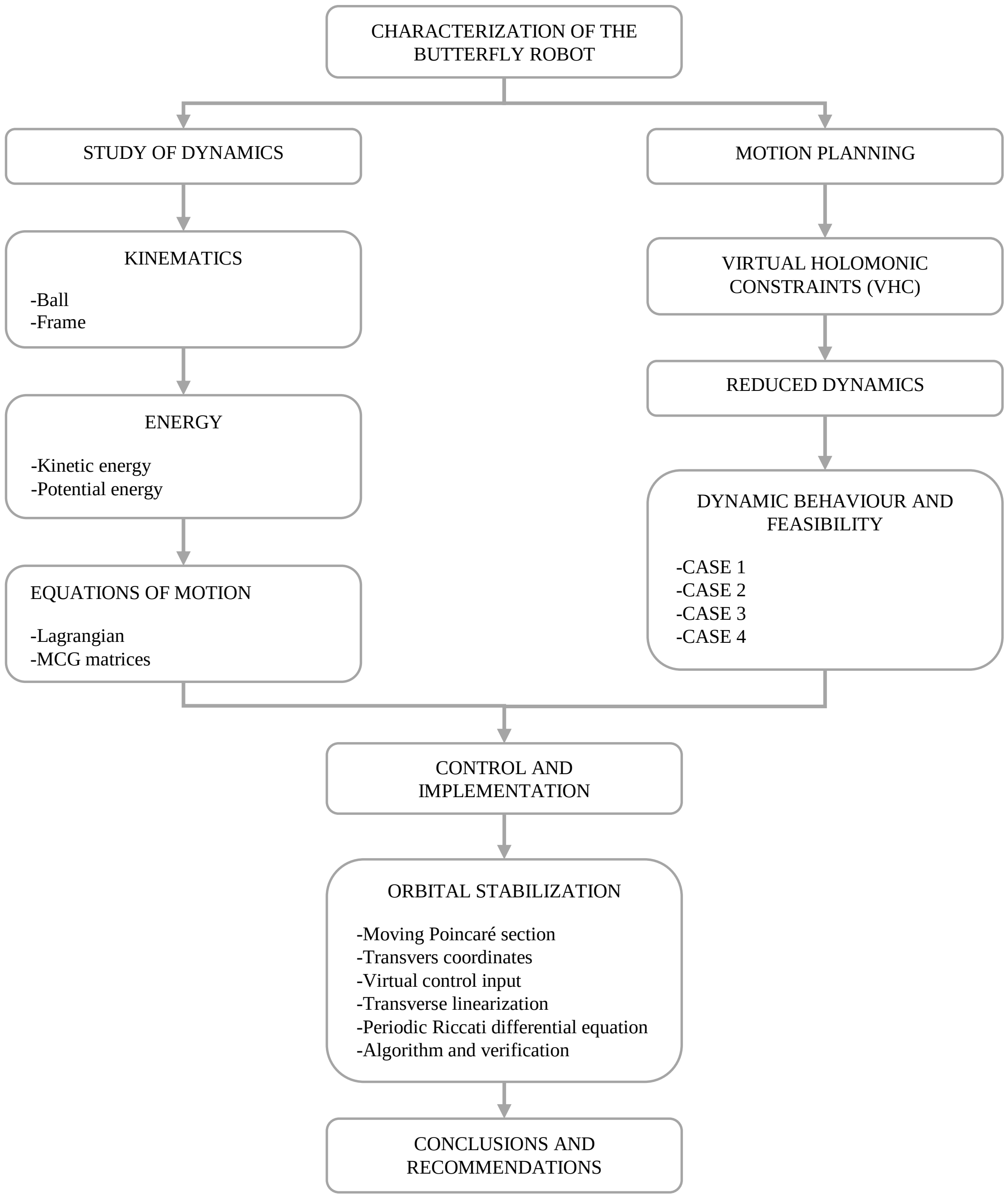

Figure 3.1 Methodology diagram – source: own elaboration.

3.2 DERIVE SYSTEM DYNAMICS

In order to develop a model based control strategy of the robot, is necessary to obtain an accurate model that represents the real physical behaviour of the Butterfly Robot. For this propose, it is been derived the system dynamics. This chapter follows the example of previous projects [2] and [3], including appropriate assumptions and simplifications.

First are exposed which are the model assumptions and the chosen coordinates as well as the reduction of degrees of freedom to represent the model. This first approach is followed by the relations of kinetic and potential energy within the elements that allow to determinate the dynamics of the system. Finally this equations of motion that represent the system are derived.

3.2.1 MODEL ASSUMPTIONS

The following assumptions take place to simplified the modelling and before any advance.

- The ball and the frame are both smooth, solid and rigid bodies with no imperfections. That means they don't deform and thereby there is just point contacts between them.
- The ball and the frame also have a uniform distribution of mass being the same the centre of mass and the geometric centre.
- The construction of the frame is such, that ensures the ball rolling motion in a 2-D surface. Consisting in two parallel identical frame with a particular edge shape similar to a butterfly.
- There is no slipping conditions between the ball and the frame. So the ball rolls through the surface of the frame without slipping.
- The ball doesn't depart from the frame so the contact between the ball and the frame is just one point every time. Even though in the study other possibilities will be consider.
- The position of the bodies of the system at the initial conditions is in one of the equilibrium points. Being the frame and the ball in the position that the Figure 1.1 describes.

3.2.2 COORDINATES AND DEGREES OF FREEDOM

To follow correctly the thesis are stablished the origin of coordinates and necessary coordinates that will guide the rest of the project. It is been assumed that the rolling motion is produced in a 2D surface, therefor the Butterfly Robot will be treated as a 2D system, like the one given in the Figure 3.2. In this figure are described the different frame reference and the degrees of freedom.

To simplify, the inertial reference frame $\overrightarrow{e^0}$, is in the centre of the frame as the no-inertial body reference frame of the frame $\overrightarrow{e^1}$. The no-inertial body reference frame of the ball $\overrightarrow{e^2}$ is in its centre of the ball.

The geometric centre of the ball and the frame determinates their position and each one have three degrees of freedom, known as $(x_f, y_f, \theta_f, x_b, y_b, \theta_b)$. However, since the reference frame is in the centre of the frame, the coordinates $x_f = y_f = 0$, and the degrees of freedom can be reduced to $(x, y, \theta_f, \theta_b)$.

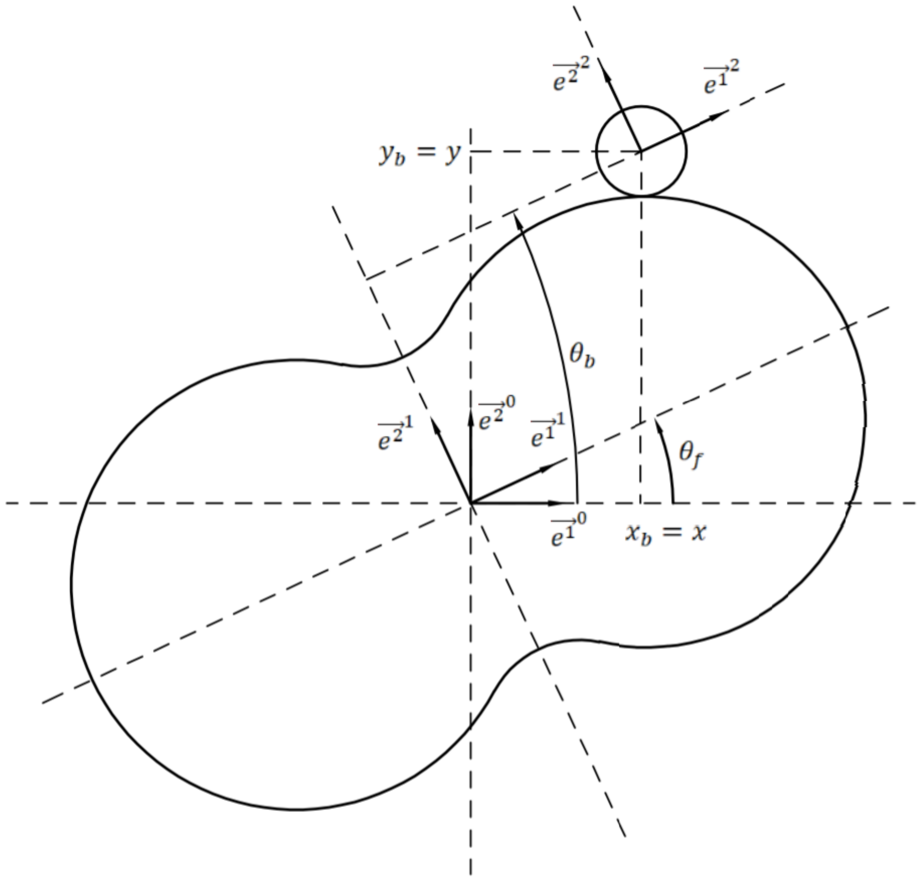

Figure 3.2 2D system, degrees of freedom – source: own elaboration.

From now on x and y are the coordinates for the position of the ball in the reference frame $\overrightarrow{e^0}$, and x' and y' the coordinates for the position of the ball in the body reference frame $\overrightarrow{e^1}$.

Due to the fact of the orbital motion that characterises the system, it is been considered more appropriate the polar coordinate system, whose equation in the Cartesian coordinate system would be much more intricate.

The distance between the ball and the ideal curve is considered to be 0. The real possibilities of interaction between the frame and the ball are two. Whether the ball is in contact with the frame or not. However, following the last of the assumptions, from now on the distance of the ball to the ideal curve is 0. Under this circumstances the ball rolls along the curve of the frame with an offset of the ball centre that is been called R see Figure 3.4.

In the figure 3.3 it can be seen the new variables to introduce the polar coordinates. The arclength of the position of the centre of the ball from the initial position along the ideal curve is described by variable s, the arclength of the position of the contact point of the ball and frame is described by variable s_f, whilst $\vec{\rho}$ is the vector from the origin of the reference frame of the frame till the centre of the ball on the ideal curve. Exists the possibility of introducing the variable $\vec{\rho}$ as a function of s given the geometrical characteristics, so that $\vec{\rho} = \vec{\rho}(s)$.

Following the previous projects [2] and [3] and from [4] is introduced the Frenet Frame that describe the kinematic properties of a particle moving along a continuous, differential curve. The Frenet formulas are used in this case to describe the position of the ball along the curve. For this matter is introduced the orthonormal base along the curve described for the ball, formed by the unit normal vector $\vec{n}$ and the tangent vector $\vec{\tau}$. The curvature vector $\vec{\kappa}$ is the third variable to define the Frenet frame. These new variables are defined as follow:

$$\vec{\tau} := \frac{d\vec{\rho}}{ds}, \qquad \vec{\kappa} := \frac{d\vec{\tau}}{ds} = \frac{d^2\vec{\rho}}{ds^2}, \qquad \vec{n} := \hat{k} \times \vec{\tau} \tag{3.1}$$

The rotation occurs around the z-direction represented by the vector unit $\hat{k} = [0 \quad 0 \quad 1]^T$.

Another way to represent the position of the ball is through the angle φ that the ball makes with respect $\overrightarrow{e_2^1}$. That allows to write $s = s(\varphi)$, and so, $\vec{\rho} = \vec{\rho}(\varphi)$. To represent the angle of the ball, instead of θ_b is introduced the variable ψ, being the rotation of the ball around its own body fixed frame. But, since the non-slip condition is been stablished, the angle ψ can be linked with the arclength s, or what is the same, with φ. See Figure 3.3.

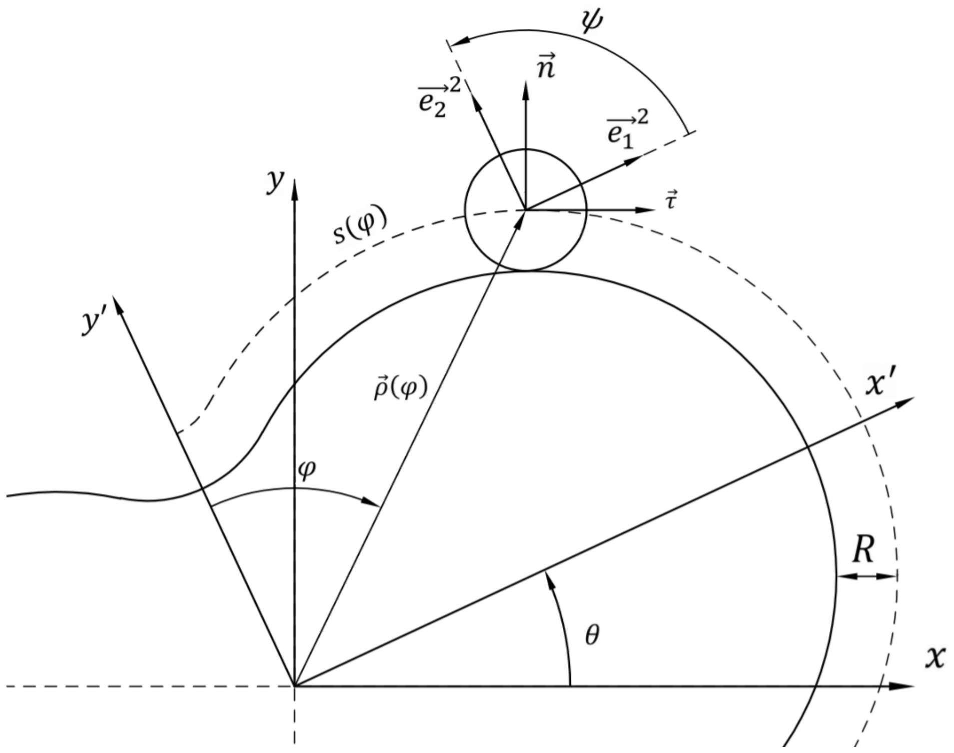

Figure 3.3 degrees of freedom in polar coordinates – source: own elaboration.

From all this new variables is presented a new set of coordinates from $(x, y, \theta_f, \theta_b)$ to (θ, φ) with $\theta_f = \theta$. The set of coordinates is being reduced to just two. This simplifies a lot the general vision of the hole process by just fixing two coordinates. These are presented as the general coordinates: $q = [\theta \quad \varphi]^T$.

The variable R, seen in the Figure 3.4, is just the projected distance between the centre of the ball and the contact of the ball with the frame, and must not be mistaken with the radius of the ball R_b. This variable R can be calculated as $R = \sqrt{{R_b}^2 - {r_f}^2}$.

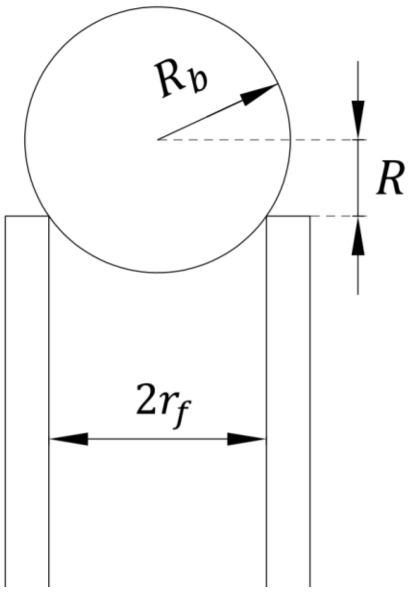

Figure 3.4 Real and effective radius – source: own elaboration.

3.3 KINEMATICS FRAME AND BALL

To represent the motion of the different bodies a kinematic study of the system is done. To obtain the equations of motion, are necessary the expressions that relate energy and kinematics.

In this section is studied the kinematics of the system considering that its trajectory is expressed in polar coordinates. The geometry of the system is been already described, as well as the position of the bodies at the initial conditions settled in the model assumptions.

From now on the variable $\vec{r}$ represents position, $\vec{v}$ represents velocity and $\vec{w}$ represents the angular velocity.

In the case of the frame considering that its centre of mass is in the origin of the reference frame of the frame $\overrightarrow{e^1}$ and the reference frame $\overrightarrow{e^0}$ it can be determined that:

$$\overrightarrow{r_f} = \overrightarrow{v_f} = [0 \quad 0 \quad 0]^T \tag{3.2}$$

$$\overrightarrow{w_f} = \dot{\theta}\hat{k} \tag{3.3}$$

The case of the ball is not as simple as in the frame and is used the vector $\vec{\rho}$ to first determinate the position of the ball with respect the reference frame of the frame $\overrightarrow{e^1}$ and then is used the cosine matrix Π to determinate the position with respect the reference frame $\overrightarrow{e^0}$.

$$\overrightarrow{r_b} = \Pi(\theta)\vec{\rho}(\varphi) \tag{3.4}$$

With:

$$\Pi(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

In order to obtain the velocity of the ball, its position is derived with respect time. As follows:

$$\overrightarrow{v_b} = \frac{d\overrightarrow{r_b}}{dt} = \dot{\Pi}\vec{\rho} + \Pi\dot{\vec{\rho}} = \overrightarrow{w_f} \times (\Pi\vec{\rho}) + \Pi\frac{d\vec{\rho}}{ds}\frac{ds}{d\varphi}\frac{d\varphi}{dt} = \dot{\theta}\hat{k} \times (\Pi\vec{\rho}) + \Pi\vec{\tau}s'\dot{\varphi} \tag{3.6}$$

With: $s = \int_0^{\varphi} \left\| \frac{d\vec{\rho}}{d\varphi} \right\| d\varphi$ and $s' = \frac{ds(\varphi)}{d\varphi}$ fully explained in equations (3.23) and (3.24).

The angular velocity of the ball is composed by 2 variables, the angular velocity of the ball around its own centre of rotation $\vec{w}_{b_{center}}$, that is exposed below, and the angular velocity of the frame $\overrightarrow{w_f}$, calculated before . Attached to the no slipping condition, the angular velocity of the ball is proportional to the trajectory of the ball along the frame s_f, and the condition can be expressed as:

$$R\psi = s_f \tag{3.7}$$

$$R\dot{\psi} = \dot{s}_f \tag{3.8}$$

The angular velocity of the ball around its own centre of rotation is:

$$\vec{w}_{b_{center}} = -\dot{\psi}\hat{k} = -\frac{1}{R}\dot{s}_f\hat{k} \tag{3.9}$$

The relation that links s_f and s is obtained through a new Frenet frame, this time at the curve of the butterfly robot frame as can be seen in Figure 3.5, denoted by:

$$\vec{\tau}_f := \frac{d\vec{\delta}}{ds_f}, \qquad \vec{\kappa}_f := \frac{d\vec{\tau}_f}{ds_f} = \frac{d^2\vec{\delta}}{ds^2{}_f}, \qquad \vec{n}_f := \hat{k} \times \vec{\tau}_f \tag{3.10}$$

Considering $\vec{\tau} = \vec{\tau}_f$, $\vec{n} = \vec{n}_f$ and $\vec{\kappa} \neq \vec{\kappa}_f$.

At this point, $\vec{\delta}$ is not been introduced yet. Basically it represents the position of the contact point of the ball and the frame with respect to the origin with $\delta = a - b\cos(2\phi)$ describing this position.

The parameter $\vec{\delta}$, depends on the angle between the y' axis of the reference frame of the frame $\overrightarrow{e^1}$, and $\vec{\delta}$. This new angle is expressed as ϕ, to express parameters, as can be seen in Figure 3.6, which results in:

$$\vec{\delta}(\phi) = \delta(\phi)\begin{bmatrix} \sin(\phi) \\ \cos(\phi) \\ 0 \end{bmatrix} \tag{3.11}$$

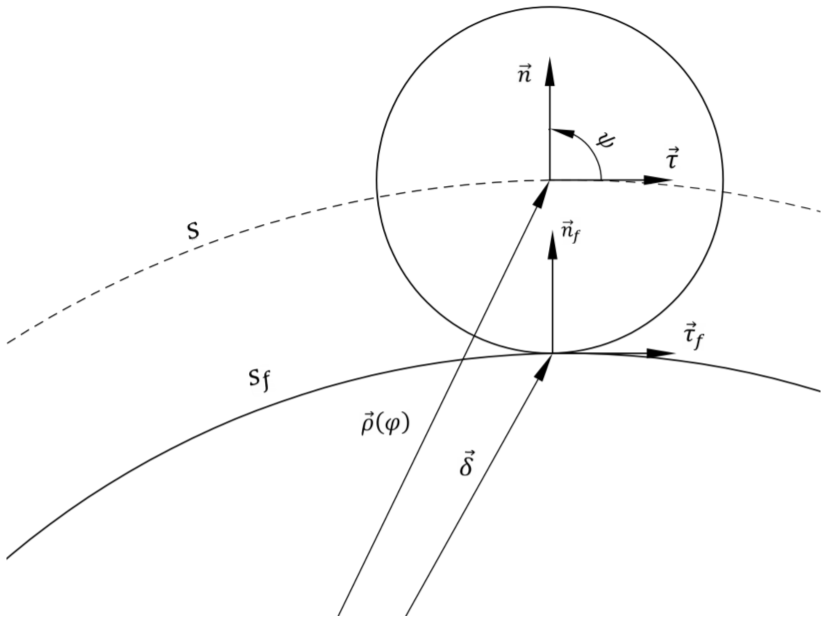

Figure 3.5 Both Frenet frame and arclength – source: own elaboration.

The unit tangential velocity along the curve, is the vector $\vec{\tau}$, and can be expressed now as:

$$\vec{\tau}(\phi) = \frac{\overrightarrow{\delta'}(\phi)}{\left\|\overrightarrow{\delta'}(\phi)\right\|}, \qquad \tau_x = \frac{\overrightarrow{\delta'}_x(\phi)}{\left\|\overrightarrow{\delta'}(\phi)\right\|}, \qquad \tau_y = \frac{\overrightarrow{\delta'}_y(\phi)}{\left\|\overrightarrow{\delta'}(\phi)\right\|} \tag{3.12}$$

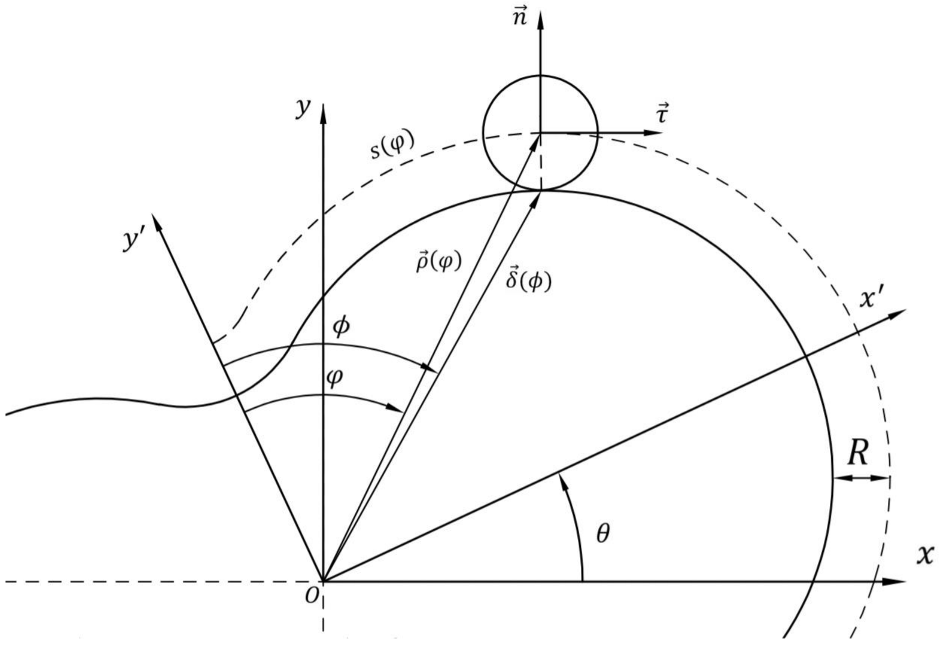

Figure 3.6 Position and angle of the contact point - source: own elaboration.

The new parameter $\alpha(\phi)$ is introduced, that represents the angle between the axis x' of the reference frame of the frame $\overrightarrow{e^1}$, and the tangent of the frame, as can be seen in Figure 3.7 and Figure 3.8. Can be described as:

$$\alpha(\phi) = \arctan\left(\frac{-\tau_y(\phi)}{\tau_x(\phi)}\right) = \arctan\left(\frac{-\vec{\delta}'_y(\phi)}{\vec{\delta}'_x(\phi)}\right) = \arctan\left(\frac{\delta(\phi)\sin(\phi) - \delta'(\phi)\cos(\phi)}{\delta(\phi)\cos(\phi) - \delta\prime(\phi)\sin(\phi)}\right) \qquad (3.13)$$

From this, as can be checked in the Figure 3.7, an expression for $\vec{n}$ can be developed, dividing it into n_x and n_y:

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

$$n_y = |\vec{n}|\sin\left(\frac{\pi}{2} - \alpha\right) = \cos(\alpha) \qquad (3.14)$$

$$\vec{n} = \begin{bmatrix} \sin(\alpha) \\ \cos(\alpha) \\ 0 \end{bmatrix} \qquad (3.15)$$

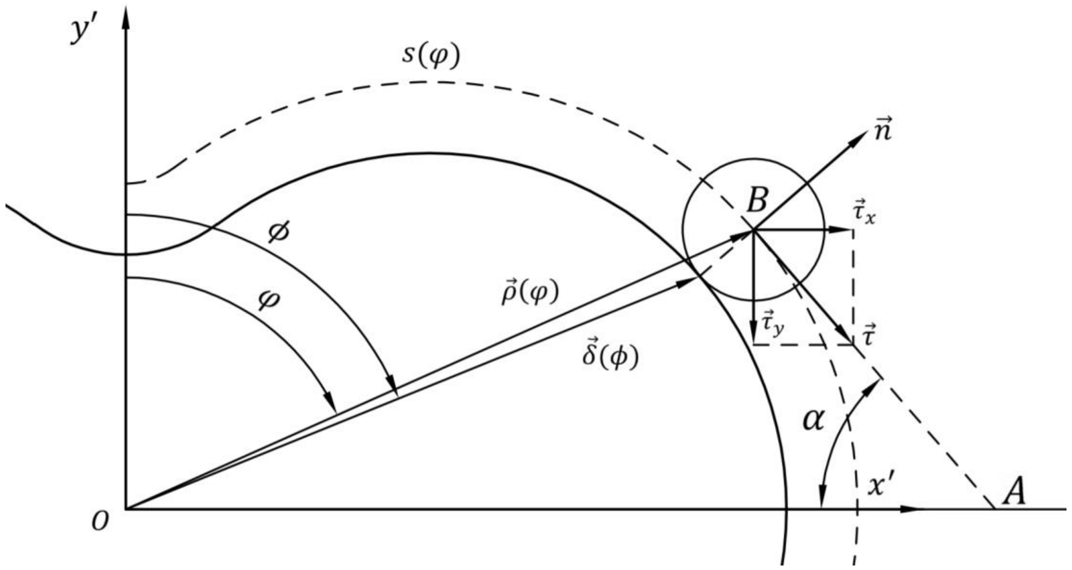

Figure 3.7 Graphical description of α ***– source: own elaboration.***

From this vectors, can be described:

$$\vec{\rho}(\phi) = \vec{\delta}(\phi) + R\vec{n}(\phi), \qquad \frac{d\vec{\rho}}{ds_f} = \frac{d\vec{\delta}}{ds_f} + R\frac{d\vec{n}}{ds_f} = \vec{\tau}_f - R\kappa_f\vec{\tau}_f = \vec{\tau}_f(1 - R\kappa_f) \qquad (3.16)$$

Considering $\frac{d\vec{n}}{ds_f} = -\kappa_f\vec{\tau}_f$ from [3].

Using the equations (3.16) and (3.10), and the derivation with respect time, leads to:

$$d\vec{\rho} = \vec{\tau}_f\left(1 - R\kappa_f\right)ds_f = \vec{\tau}_f ds \tag{3.17}$$

$$ds = \left(1 - R\kappa_f\right)ds_f \tag{3.18}$$

$$s = \left(1 - R\kappa_f\right)s_f \tag{3.19}$$

$$\dot{s} = \left(1 - R\kappa_f\right)\dot{s}_f \tag{3.20}$$

Following the example in [3] is possible to express the relation between $\vec{\kappa}$ and $\vec{\kappa}_f$ as:

$$\vec{\kappa} = \frac{d\vec{\tau}}{ds} = \frac{d\vec{\tau}_f}{ds} = \frac{ds_f}{ds}\frac{d\vec{\tau}_f}{ds_f} = \frac{\kappa_f\vec{n}_f}{1-R\kappa_f} = \kappa\vec{n} \tag{3.21}$$

Considering that this relations are true whenever ds and ds_f, are small enough. And finally is obtained the second variable, described like:

$$\vec{w}_{b_{center}} = -\frac{1}{R}\dot{s}_f\hat{k} = -\frac{1}{R\left(1-R\kappa_f\right)}\dot{s}\hat{k} = \frac{1}{R\left(R\kappa_f-1\right)}s'\dot{\varphi}\hat{k} = ps'\dot{\varphi}\hat{k} \tag{3.22}$$

With: $p = \frac{1}{R\left(R\kappa_f-1\right)}$.

At this point is convenient to express :

$$s = \int_0^{\varphi}\left\|\frac{d\vec{\rho}}{d\varphi}\right\| d\varphi \tag{3.23}$$

$$s' = \frac{ds(\varphi)}{d\varphi} = \left\|\frac{d\vec{\rho}}{d\varphi}\right\| = \left\|\frac{d\vec{\rho}}{d\phi}\frac{d\phi}{d\varphi}\right\| = \left\|\frac{d\vec{\rho}}{d\phi}\right\|\frac{d\phi}{d\varphi} = \left\|\frac{d\vec{\delta}}{d\phi} + R\frac{d\vec{n}}{d\phi}\right\|\frac{d\phi}{d\varphi} = \left\|\vec{\delta}' + R\vec{n}'\right\|\frac{1}{g\prime} \tag{3.24}$$

$$s'' = \frac{ds\prime}{d\varphi} = \frac{ds\prime}{d\phi}\frac{d\varphi}{d\phi} \tag{3.25}$$

Where g is the relation $\varphi = g(\phi)$. Following [2], the position of the centre of the ball $\vec{\rho}$ can be expressed with its coordinates:

$$\begin{aligned} x' &= \left|\vec{\delta}_x\right| + R\left|\vec{n}_x\right| = \delta\sin(\phi) + R\sin(\alpha) \\ y' &= \left|\vec{\delta}_y\right| + R\left|\vec{n}_y\right| = \delta\cos(\phi) + R\cos(\alpha) \end{aligned} \tag{3.26}$$

$$x' = \left|\vec{\rho}_x\right| = \left|\vec{\rho}\right|\sin(\varphi)$$

$$y' = |\vec{\rho}_y| = |\vec{\rho}|\cos(\varphi) \tag{3.27}$$

And as a result:

$$\varphi = \arctan\left(\frac{x'}{y'}\right) = \left(\frac{\delta\sin(\phi) + R\sin(\alpha)}{\delta\cos(\phi) + R\cos(\alpha)}\right) = g(\phi) \tag{3.28}$$

This formula can be rewritten following [4] and by introducing the variable $h = \vec{\delta}' + R\vec{n}'$ and its derivative $h' = \vec{\delta}'' + R\vec{n}''$, like this:

$$\frac{ds'}{d\phi} = \frac{d\|h\|}{d\phi}\frac{1}{g'} - \frac{\|h\|g''}{g'^2} = \frac{h\cdot h'}{\|h\|g'} - \frac{\|h\|g''}{g'^2} \tag{3.29}$$

$$s'' = \left(\frac{h\cdot h'}{\|h\|g'} - \frac{\|h\|g''}{g'^2}\right)\frac{1}{g'} = \frac{(\vec{\delta}'+R\vec{n}')\cdot(\vec{\delta}''+R\vec{n}'')}{\left\|\vec{\delta}'+R\vec{n}'\right\|g'^2} - \frac{\left\|\vec{\delta}'+R\vec{n}'\right\|g''}{g'^3} \tag{3.30}$$

The total angular velocity of the ball is expressed as:

$$\vec{w}_b = \vec{w}_f + \vec{w}_{b_{center}} = \dot{\theta}\hat{k} + ps'\dot{\varphi}\hat{k} = (\dot{\theta} + ps'\dot{\varphi}\ )\hat{k} \tag{3.31}$$

3.4 ENERGY

In this section are analysed the formulas for potential and kinetic energy of both frame and ball, necessary to develop the equations of motion. It is been followed the same assumptions than in [2] and [3].

3.4.1 KINETIC ENERGY

For the kinetic energy, from its main formula are calculated the expressions for the ball and the frame, considering that its formed by the translational and rotational motion as follows:

$$\mathcal{K} = \frac{1}{2}m\vec{v}\cdot\vec{v} + \frac{1}{2}J\vec{w}\cdot\vec{w} \tag{3.32}$$

In the following equations, m represents the mass and J the moment of inertia. From this point, can be extrapolated to the frame and the ball considering the kinematics formulas in the previous section, as follows:

$$\mathcal{K}_f = \frac{1}{2}m_f\vec{v}_f\cdot\vec{v}_f + \frac{1}{2}J_f\vec{w}_f\cdot\vec{w}_f = \frac{1}{2}J_f\dot{\theta}^2 \tag{3.33}$$

$$\mathcal{K}_b = \frac{1}{2}m_b\vec{v}_b\cdot\vec{v}_b + \frac{1}{2}J_b\vec{w}_b\cdot\vec{w}_b$$

$$= \frac{1}{2}m_b\left(\left(\dot{\theta}\hat{k}\times(\Pi\vec{\rho}) + \Pi\vec{\tau}s'\dot{\varphi}\right)\cdot\left(\dot{\theta}\hat{k}\times(\Pi\vec{\rho}) + \Pi\vec{\tau}s'\dot{\varphi}\right)\right) + \frac{1}{2}J_b\left((\dot{\theta} + ps'\dot{\varphi}\ )\hat{k}\cdot(\dot{\theta} + ps'\dot{\varphi}\ )\hat{k}\right)$$

$$= \frac{1}{2} m_b \big(\dot{\theta}^2 (\hat{k} \times (\Pi \vec{\rho})) \cdot (\hat{k} \times (\Pi \vec{\rho}) + 2 s' \dot{\theta} \dot{\varphi} (\Pi \vec{\tau}) \cdot (\hat{k} \times (\Pi \vec{\rho})) + {s'}^2 \dot{\varphi}^2 \vec{\tau}^T \Pi^T \Pi \vec{\tau}\big)$$

$$+ \frac{1}{2} J_b \big(\dot{\theta}^2 + 2 p s' \dot{\theta} \dot{\varphi} + p^2 {s'}^2 \dot{\varphi}^2\big) \tag{3.34}$$

This equation can be simplified with:

$$(\hat{k} \times (\Pi \vec{\rho}) \cdot (\hat{k} \times (\Pi \vec{\rho}) = \|\vec{\rho}\|^2$$

$$(\Pi \vec{\tau}) \cdot (\hat{k} \times (\Pi \vec{\rho})) = \hat{k} \cdot (\vec{\rho} \times \vec{\tau})$$

$$\vec{\tau}^T \Pi^T \Pi \vec{\tau} = 1 \tag{3.35}$$

As a result:

$$\mathcal{K}_b = \frac{1}{2} m_b \big(\dot{\theta}^2 \|\vec{\rho}\|^2 + 2 s' \dot{\theta} \dot{\varphi} (\hat{k} \cdot (\vec{\rho} \times \vec{\tau})) + {s'}^2 \dot{\varphi}^2\big) + \frac{1}{2} J_b \big(\dot{\theta}^2 + 2 p s' \dot{\theta} \dot{\varphi} + p^2 {s'}^2 \dot{\varphi}^2\big) \tag{3.36}$$

Being the total kinetic energy:

$$\mathcal{K} = \mathcal{K}_f + \mathcal{K}_b = \frac{1}{2} \big(m_b \|\vec{\rho}\|^2 + J_f + J_b\big) \dot{\theta}^2 + \big(m_b \hat{k} \cdot (\vec{\rho} \times \vec{\tau}) + J_b p\big) s' \dot{\theta} \dot{\varphi} + \frac{1}{2} (m_b + J_b p^2) {s'}^2 \dot{\varphi}^2 \tag{3.37}$$

3.4.2 POTENTIAL ENERGY

For the potential energy, from its main formula are calculated the expressions for the ball and the frame as well, considering $\vec{g} = (0 \quad g \quad 0)^T$ the gravitational acceleration and $\vec{r}$ the position of the body as:

$$\mathcal{P} = m \vec{g} \cdot \vec{r} \tag{3.38}$$

For the frame $\mathcal{P}_f = 0$, because the gravitational force doesn't have an effect on it. For the ball:

$$\mathcal{P}_b = m_b \vec{g} \cdot \vec{r}_b = m_b \vec{g} \cdot (\Pi \vec{\rho}) \tag{3.39}$$

Finally the total potential energy can be obtained as:

$$\mathcal{P} = \mathcal{P}_f + \mathcal{P}_b = m_b \vec{g} \cdot (\Pi \vec{\rho}) \tag{3.40}$$

2.5 EQUATIONS OF MOTION

In the first steps, is been exposed which are the reduced degrees of freedom to represent the model and the relations of kinetic and potential energy that determinates the dynamics of the system. In this section this equations of motion that represent the system are derived to find the equations of motion. For this purpose, is used the Lagrangian:

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

$$= \frac{1}{2} \big(m_b \|\vec{\rho}\|^2 + J_f + J_b\big) \dot{\theta}^2 + \big(m_b \hat{k} \cdot (\vec{\rho} \times \vec{\tau}) + J_b p\big) s' \dot{\theta} \dot{\varphi} + \frac{1}{2} (m_b + J_b p^2) {s'}^2 \dot{\varphi}^2 -$$

$$m_b \vec{g} \cdot (\Pi \vec{\rho}) \tag{3.41}$$

The Euler-Langrange equations derive the equations of motion with the Lagrangian as follows:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}}\right) - \frac{\partial \mathcal{L}}{\partial \theta} = u$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\varphi}}\right) - \frac{\partial \mathcal{L}}{\partial \varphi} = 0 \qquad (3.42)$$

The equation of motion can be represented using a transformation of the previous equations as follows:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)u \qquad (3.43)$$

In this expression is particularly important the character u that is the actuator driving the frame. $B(q) = [1 \quad 0]^T$ is the coupling matrix to stablish where the actuator drives the system, $M(q)$ is the mass matrix, $C(q,\dot{q})$ is the Coriolis and centrifugal matrix and $G(q)$ is the gravity matrix. The steps to obtain these matrix, are the same taken in previous work [2] and [3]. Each matrix has its own algorithm where:

$$m_{ij} \coloneqq \frac{\partial}{\partial \dot{\theta}}\left(\frac{\partial \mathcal{K}}{\partial \dot{q}_j}\right)$$

$$c_{jk} \coloneqq \sum_{i=1}^{2} c_{ijk}(q)\dot{q}_i$$

$$g_i \coloneqq \frac{\partial \mathcal{P}}{\partial q_i} \qquad (3.44)$$

Where $c_{ijk} \coloneqq \frac{1}{2}\left(\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k}\right)$

Mass Matrix $M(q)$

$$m_{11} \coloneqq \frac{\partial}{\partial \dot{\theta}}\left(\frac{\partial \mathcal{K}}{\partial \dot{\theta}}\right) = m_b\|\vec{\rho}\|^2 + J_f + J_b$$

$$m_{12} \coloneqq \frac{\partial}{\partial \dot{\theta}}\left(\frac{\partial \mathcal{K}}{\partial \dot{\varphi}}\right) = \left(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) + J_b p\right)s'$$

$$m_{21} \coloneqq \frac{\partial}{\partial \dot{\varphi}}\left(\frac{\partial \mathcal{K}}{\partial \dot{\theta}}\right) = m_{12}$$

$$m_{22} \coloneqq \frac{\partial}{\partial \dot{\varphi}}\left(\frac{\partial \mathcal{K}}{\varphi}\right) = (m_b + J_b p^2){s'}^2 \qquad (3.45)$$

As a result:

$$M(q) = \begin{bmatrix} m_b\|\vec{\rho}\|^2 + J_f + J_b & \left(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) + J_b p\right)s' \\ \left(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) + J_b p\right)s' & (m_b + J_b p^2){s'}^2 \end{bmatrix} \qquad (3.46)$$

Centrifugal and Coriolis matrix $C(q,\dot{q})$

$$c_{111} \coloneqq \frac{1}{2}\frac{\partial m_{11}}{\partial \theta} = 0$$

$$c_{112} \coloneqq \frac{1}{2}\left(2\frac{\partial m_{21}}{\partial \theta} - \frac{\partial m_{11}}{\partial \varphi}\right) = -m_b s' \vec{\rho} \cdot \vec{\tau}$$

$$c_{121} \coloneqq \frac{1}{2}\frac{\partial m_{11}}{\partial \varphi} = m_b s' \vec{\rho} \cdot \vec{\tau}$$

$$c_{122} \coloneqq \frac{1}{2}\frac{\partial m_{22}}{\partial \theta} = 0$$

$$c_{211} \coloneqq \frac{1}{2}\frac{\partial m_{11}}{\partial \varphi} = m_b s' \vec{\rho} \cdot \vec{\tau}$$

$$c_{212} \coloneqq \frac{1}{2}\frac{\partial m_{22}}{\partial \theta} = 0$$

$$c_{221} \coloneqq \frac{1}{2}\left(2\frac{\partial m_{12}}{\partial \varphi} - \frac{\partial m_{22}}{\partial \theta}\right) = \left(m_b \hat{k} \cdot (\vec{\rho} \times \vec{\tau}) + J_b p\right) s'' + m_b \hat{k} \cdot (\vec{\rho} \times \vec{\kappa}) {s'}^2$$

$$c_{222} \coloneqq \frac{1}{2}\frac{\partial m_{22}}{\partial \varphi} = (m_b + J_b p^2) s' s'' \tag{3.47}$$

Now using the formulas above:

$$c_{11} = c_{111}\dot{\theta} + c_{211}\dot{\varphi} = m_b s' \vec{\rho} \cdot \vec{\tau} \dot{\varphi}$$

$$c_{12} = c_{121}\dot{\theta} + c_{221}\dot{\varphi} = m_b s' \vec{\rho} \cdot \vec{\tau} \dot{\theta} + \left(\left(m_b \hat{k} \cdot (\vec{\rho} \times \vec{\tau}) + J_b p\right) s'' + m_b \hat{k} \cdot (\vec{\rho} \times \vec{\kappa}) {s'}^2\right) \dot{\varphi}$$

$$c_{21} = c_{112}\dot{\theta} + c_{212}\dot{\varphi} = -m_b s' \vec{\rho} \cdot \vec{\tau} \dot{\theta}$$

$$c_{22} = c_{122}\dot{\theta} + c_{222}\dot{\varphi} = (m_b + J_b p^2) s' s'' \dot{\varphi} \tag{3.48}$$

As a result:

$$C(q, \dot{q}) = \begin{bmatrix} m_b s' \vec{\rho} \cdot \vec{\tau} \dot{\varphi} & m_b s' \vec{\rho} \cdot \vec{\tau} \dot{\theta} + \left(\left(m_b \hat{k} \cdot (\vec{\rho} \times \vec{\tau}) + J_b p\right) s'' + m_b \hat{k} \cdot (\vec{\rho} \times \vec{\kappa}) {s'}^2\right) \dot{\varphi} \\ -m_b s' \vec{\rho} \cdot \vec{\tau} \dot{\theta} & (m_b + J_b p^2) s' s'' \dot{\varphi} \end{bmatrix} \tag{3.49}$$

Gravity matrix $G(q)$

$$g_1 = \frac{\partial \mathcal{P}}{\partial \theta} = m_b \vec{g} \cdot (\Pi' \vec{\rho})$$

$$g_2 = \frac{\partial \mathcal{P}}{\partial \varphi} = m_b \vec{g} \cdot (\Pi \vec{\tau} s') \tag{3.50}$$

As a result:

$$G(q) = \begin{bmatrix} m_b \vec{g} \cdot (\Pi' \vec{\rho}) \\ m_b \vec{g} \cdot (\Pi \vec{\tau} s') \end{bmatrix} \tag{3.51}$$

With $\Pi' = \frac{d\Pi}{d\theta}$.

4 MOTION PLANNING

4.1 INTRODUCTION

In the last chapters first have been studied the physical characteristics with some assumptions. Then the relations between kinematics and the study of the energy in the system have permitted to express the dynamics of the Butterfly robot and all the relations between parameters. The next natural step is a study of the motion planning to find feasible trajectories for the system. In this chapter, is studied the motion planning performed in [2] and [3] and new feasible trajectories are accomplished.

4.2 VIRTUAL HOLOMONIC CONSTRAINT

In the study of the system done, one of the things that characterizes and makes it challenging is the fact that is an underactuated system with to degreed of freedom $q = [\theta \quad \varphi]^T$ and just one input or actuator in the system u to control them.

To overcome this problem is used what is known as a virtual holonomic constraint (VHC). A VHC is a constraint that does not exists in reality and that links two variables, in this case, the two degrees of freedom θ and φ, so that controlling one of them, the other is controlled as well. The ball forms part of the system but a VHC is necessary in order to control it. The same procedure is conducted in [2] and [3].

In the case that this thesis fronts, the VHC is applied from φ to θ. That means that θ can be described as a function of φ and the structure that describes is as follows:

$$\begin{aligned} \theta_* &= \Theta(\varphi_*) \\ \dot{\theta}_* &= \Theta'(\varphi_*)\dot{\varphi}_* \\ \ddot{\theta}_* &= \Theta''(\varphi_*)\dot{\varphi}_*{}^2 + \Theta'(\varphi_*)\ddot{\varphi}_* \end{aligned} \tag{4.1}$$

Where $\Theta'(\varphi_*) = \frac{d\Theta(\varphi_*)}{d\varphi_*}$ and $\Theta''(\varphi) = \frac{d^2\Theta(\varphi_*)}{d\varphi_*{}^2}$. Using the VHC, is possible to define the generalized coordinates as:

$$\begin{aligned} q_*(t) &= \begin{bmatrix} \theta_* \\ \varphi_* \end{bmatrix} = \begin{bmatrix} \Theta(\varphi_*) \\ \varphi_* \end{bmatrix} = \phi(\varphi_*) \\ \dot{q}_*(t) &= \begin{bmatrix} \dot{\theta}_* \\ \dot{\varphi}_* \end{bmatrix} = \begin{bmatrix} \Theta'(\varphi_*) \\ 1 \end{bmatrix} \dot{\varphi}_* = \phi'(\varphi_*)\dot{\varphi}_* \\ \ddot{q}_*(t) &= \begin{bmatrix} \ddot{\theta}_* \\ \ddot{\varphi}_* \end{bmatrix} = \begin{bmatrix} \Theta''(\varphi_*) \\ 0 \end{bmatrix} \dot{\varphi}_*{}^2 + \begin{bmatrix} \Theta'(\varphi_*) \\ 1 \end{bmatrix} \ddot{\varphi}_* = \phi''(\varphi_*)\dot{\varphi}_*{}^2 + \phi'(\varphi_*)\ddot{\varphi}_* \end{aligned} \tag{4.2}$$

From now on, using the symbol '$*$' means that it is using the VHC, so θ is linked virtually with φ and all the variables depend on $\varphi = \varphi_*$.

The assumptions of the system taken in the previous chapter forces the need to find motions with certain characteristics, which can only be achieved by applying a correct VHC and initial conditions.

Therefore there are many factors to consider when designing a VHC, represented as requirements and restrictions.

This conditions have been studied before and thoroughly in [2] and [3]. Is not object of this thesis to insist in this matter. Therefore, from this document are gathered the requirements that the VHC has to fulfil according with how the system behaves and that can be summarized in:

- $\Theta(\varphi_*)$ must be twice differentiable
- If the desired motion is one-directional, then $\Theta(\varphi_*)$ must be continuously increasing.
- $\Theta'(\varphi_*)$ and $\Theta''(\varphi_*)$ must be π-periodic.
- The resulting reduced dynamics should be bounded.

The VHC is an important part of the motion planning to search for feasible trajectories. And the desire for a specific behaviour restricts the possibilities of finding one. For that reason in this thesis it is been chosen a simple VHC considered before in [2] and can be defined as:

$$\Theta(\varphi_*) = \varphi_* - c\sin(2\varphi_*)$$

$$\Theta'(\varphi_*) = 1 - 2c\cos(2\varphi_*)$$

$$\Theta''(\varphi_*) = 4c\sin(2\varphi_*) \tag{4.3}$$

The value of parameter c that completes the definition of the VHC, is specified in the section 4.5 and it is vitally important to develop the different desired trajectories.

4.3 REDUCED DYNAMICS

Following [3], the VHC developed is used to rewrite the equation of motion 4.4 , but just as a function of the only degree of freedom left φ_*.

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)u \tag{4.4}$$

For this propose, is used the called annihilator matrix $B^{\perp}(q)$ to cancel the effect of u as follow:

$$B^{\perp}(q)M(q)\ddot{q} + B^{\perp}(q)C(q,\dot{q})\dot{q} + B^{\perp}(q)G(q) = B^{\perp}(q)B(q)u = 0 \tag{4.5}$$

Where $B^{\perp}(q) = [0 \quad 1]^T$.

The rest of the equation is adapted from the effect of $B^{\perp}(q)$ and from the ordinary generalized coordinates $q = [\theta \quad \varphi]^T$ to the new generalized coordinates from the equations (4.2) as follows:

$$[m_{21} \quad m_{22}]\left(\phi'\ddot{\varphi}_* + \phi''\dot{\varphi}_*{}^2\right) + [c_{21} \quad c_{22}]\phi'\dot{\varphi}_* + g_2 = 0$$

$$(m_{21}\Theta' + m_{22})\ddot{\varphi}_* + m_{21}\Theta''\dot{\varphi}_*{}^2 + (c_{21}\Theta' + c_{22})\dot{\varphi}_* + g_2 =$$

$$= (m_{21}\Theta' + m_{22})\ddot{\varphi}_* + (m_{21}\Theta'' + \tfrac{c_{21}\Theta'}{\dot{\varphi}_*} + \tfrac{c_{22}}{\dot{\varphi}_*})\dot{\varphi}_*{}^2 + g_2 = 0 \tag{4.6}$$

What can be finally reduced to what is called the $\alpha\beta\gamma$ equation, depending just on φ_* and its derivatives. The equation is described as:

$$\alpha(\varphi_*)\ddot{\varphi}_* + \beta(\varphi_*)\dot{\varphi}_*{}^2 + \gamma(\varphi_*) = 0 \tag{4.7}$$

Where:

$$\alpha(\varphi_*) = \big(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) + J_b p\big)s'\Theta' + (m_b + J_b p^2){s'}^2$$
$$\beta(\varphi_*) = \big(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) + J_b p\big)s'\Theta'' - m_b s'\vec{\rho}\cdot\vec{\tau}\Theta'^2 + (m_b + J_b p^2)s's''$$
$$\gamma(\varphi_*) = m_b\vec{g}\cdot(\Pi\vec{\tau}s') \tag{4.8}$$

And the equation for the input u can be simplified:

$$u = [m_{11} \quad m_{12}]\begin{bmatrix}\ddot{\theta}_*\\ \ddot{\varphi}_*\end{bmatrix} + [c_{11} \quad c_{12}]\begin{bmatrix}\dot{\theta}_*\\ \dot{\varphi}_*\end{bmatrix} + g_1 =$$
$$= [m_{11} \quad m_{12}]\begin{bmatrix}\Theta'\ddot{\varphi}_* + \Theta''{\dot{\varphi}_*}^2\\ \ddot{\varphi}_*\end{bmatrix} + [c_{11} \quad c_{12}]\begin{bmatrix}\Theta'\dot{\varphi}_*\\ \dot{\varphi}_*\end{bmatrix} + g_1 \tag{4.9}$$

4.4 CHECKING FEASIBILITY

To check the feasibility of the motion planning related with the election of the VHC and the initial conditions, is necessary to set some relation between parameters and verify the fact that there are no slip conditions and that the ball does not depart any moment in its trajectory from the frame as has been established in the model assumptions.

From the equation (4.7), is possible to obtain the next relation:

$$\ddot{\varphi}_* = -\frac{\beta}{\alpha}{\dot{\varphi}_*}^2 - \frac{\gamma}{\alpha} \tag{4.10}$$

From this equation is necessary to prevent the that $\ddot{\varphi}_*$ tends to infinity. If $\ddot{\varphi}_* \to \infty$ is because $\alpha \to 0$. For this propose, in the thesis are chosen different VHC changing the value of c from (4.3) with a condition. What can be traduced from the equation (4.7) into the next condition:

$$\Theta'(\varphi_*) \neq -\frac{(m_b+J_b p^2)s'}{(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau})+J_b p)} \tag{4.11}$$

This way, the resulting reduced dynamics are bounded.

The next question to solve is related with the assumptions made at the beginning of the thesis, specifically with the no slip and no departure conditions. The problem with this conditions is that there is no relation between parameters that can assure its compliance. However, it can be checked by obtaining the values of 3 new variables, the normal force F_n, the friction force F_s and the friction coefficient μ between the ball and the frame. The no departure condition complies if F_n is positive and the no slip condition complies if there is an agreement between F_n, F_s and the coefficient μ. What can be traduced into:

$$F_n > 0$$
$$-\mu F_n < F_s < \mu F_n \tag{4.12}$$

Is possible to find the expressions for the forces F_n and F_s by using the degrees of freedom and using constraints. A way to get these expressions can be found following the example and complete derivation of dynamics in [3] as follows:

$$F_n = m_b\left(\hat{k}\cdot(\vec{\rho}\times\vec{n})\ddot{\theta}_* + \frac{1}{2}(\vec{\rho}\cdot\vec{n})\dot{\theta}_*^{\ 2} + 2s'\dot{\theta}_*\dot{\varphi}_* + \vec{g}\cdot(\Pi\vec{n})\right)$$

$$F_s = -\tfrac{J_b}{R}\left(ps''\dot{\varphi}_*^{\ 2} + ps'\ddot{\varphi}_* + \ddot{\theta}_*\right) \tag{4.13}$$

Finally, the last of the requirements can be checked for the VHC selected in the different trajectories. In the next section different values for c to complete the VHC are used and for each trajectory solution its feasibility is verified.

4.5 DYNAMIC BEHAVIOUR

The choice of the initial conditions φ_{0*} and $\dot{\varphi}_{0*}$ are also essential when shaping the desired trajectory. The system's motion can be substantially changed depending on the initial conditions. To get a valid solution is needed an agreement between them and the VHC.

As it is been explained, there is an entire process to find new motion planning so that we can get different behaviours. The idea in this section is to find different new motions based on the change of parameters. That is why are presented 4 different feasible trajectories for the system. The first of them has been replicated from [2] and [3]. The parameters to change are and the initial conditions φ_0 and $\dot{\varphi}_0$ and the value of c, from the VHC:

$$\Theta(\varphi_*) = \varphi_* - c\sin(2\varphi_*)$$

$$\Theta'(\varphi_*) = 1 - 2c\cos(2\varphi_*)$$

$$\Theta''(\varphi_*) = 4c\ \sin(2\varphi_*) \tag{4.14}$$

All the algorithms developed in MATLAB of this part of the thesis can be found in APPENDIX. First are given the values of the parameters to define the VHC and the initial conditions during the trajectory. Then is shown the phase portrait for the trajectory, followed by the required input, the evolution with time of the different variables and the demonstration of its feasibility. Ultimately are given some values of reference for each trajectory as the period T of the trajectory and the minimum value for μ.

4.5.1 CASE 1

In the first case, the trajectory described is a continuous rotation of the frame in a clockwise direction. For this propose is used:

$c = 0.49$ during all the trajectory.

Initial conditions: $\varphi_{0*} = 0\ \ rad$ and $\dot{\varphi}_{0*} = 4.3\ \ rad/s$

For this initial conditions and VHC, the results can be seen in Figure 4.1, 4.2, and 4.3. As can be seen in the Figure 4.3 the variable φ_* is continuously increasing with time and that results in a periodical response of $\dot{\varphi}_*$ and u.

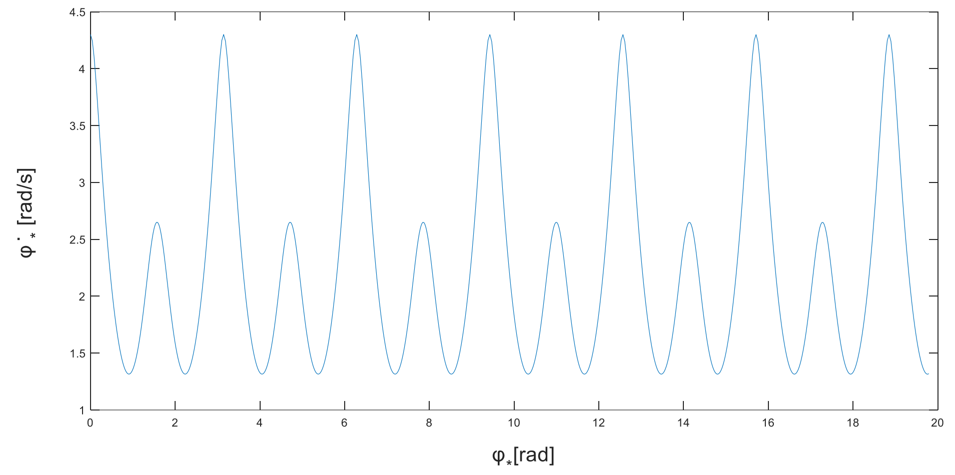

Figure 4.1 Case 1 trajectory – source: own elaboration.

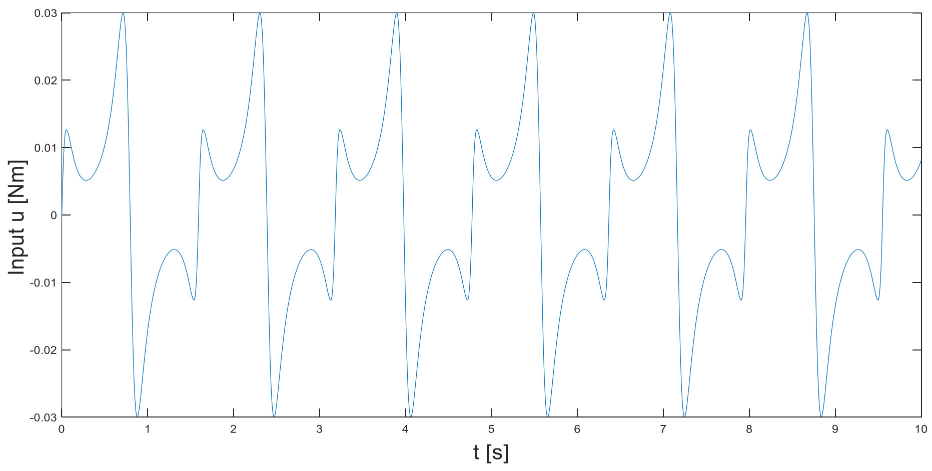

Figure 4.2 Case 1 input – source: own elaboration.

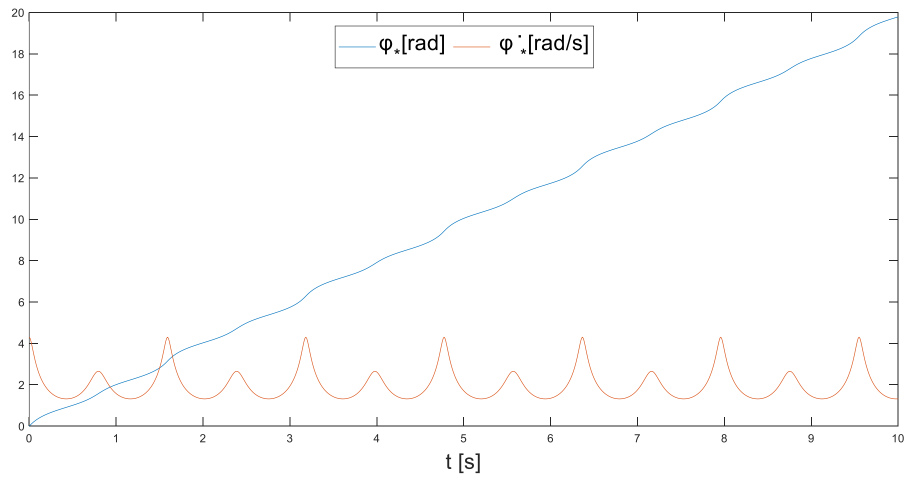

Figure 4.3 Case 1 evolution time – source: own elaboration.

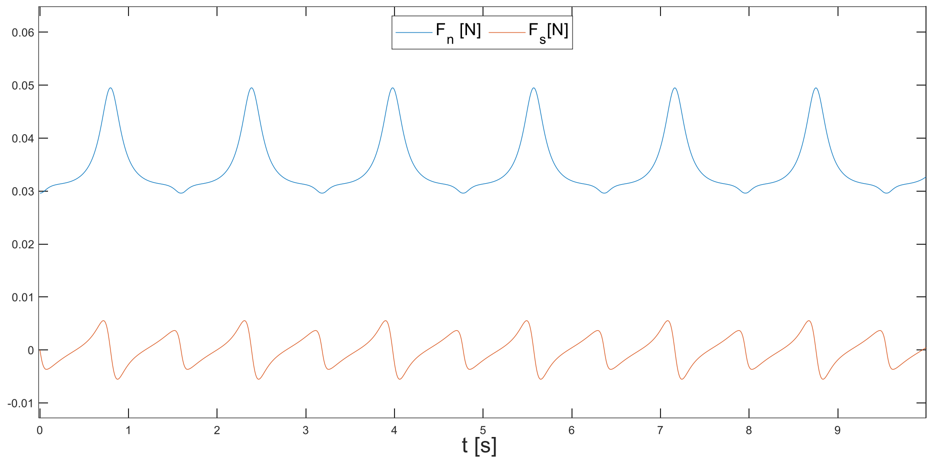

Figure 4.4 Case 1 Normal and friction force – source: own elaboration

As can be seen in Figure 4.4 the normal force F_n is positive and meets the requirements. And for this trajectory the value of μ that guarantees the no slip conditions is $\mu > 0.1252$. The period in this trajectory is $T = 3.19 seconds$.

4.5.2 CASE 2:

In the second case, the trajectory described is a continuous rotation of the frame in a clockwise direction as the case 1. The difference is that now the laps are slower. For this propose is used:

$c = 0.49$ during all the trajectory.

Initial conditions: $\varphi_{0*} = 0 \ rad$ and $\dot{\varphi}_{0*} = 4.1 \ rad/s$

For this initial conditions and VHC, the results can be seen in Figures 4.5, 4.6, and 4.7. As can be seen in the Figure 4.7 the variable φ_* is continuously increasing with time and that results in a periodical response of $\dot{\varphi}_*$ and u.

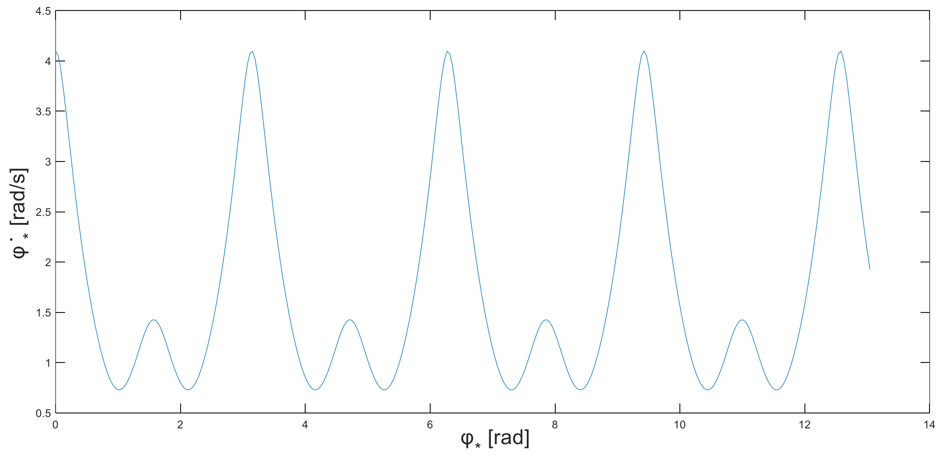

Figure 4.5 Case 2 trajectory – source: own elaboration.

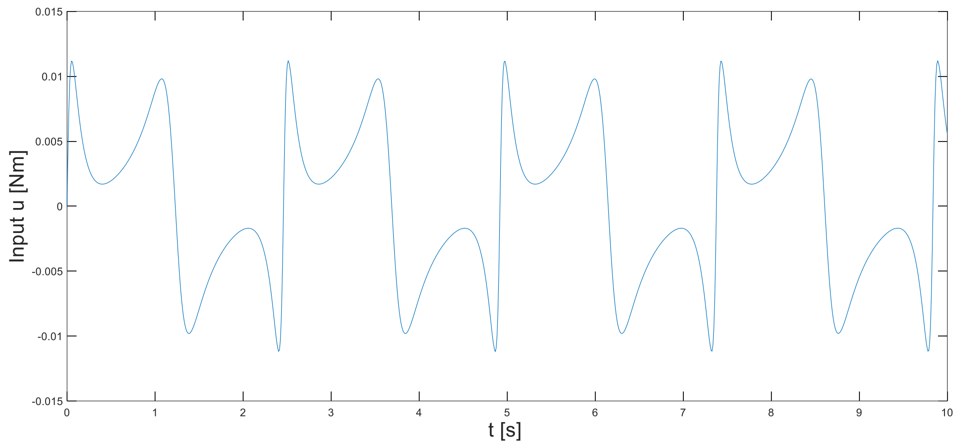

Figure 4.6 Case 2 input – source: own elaboration.

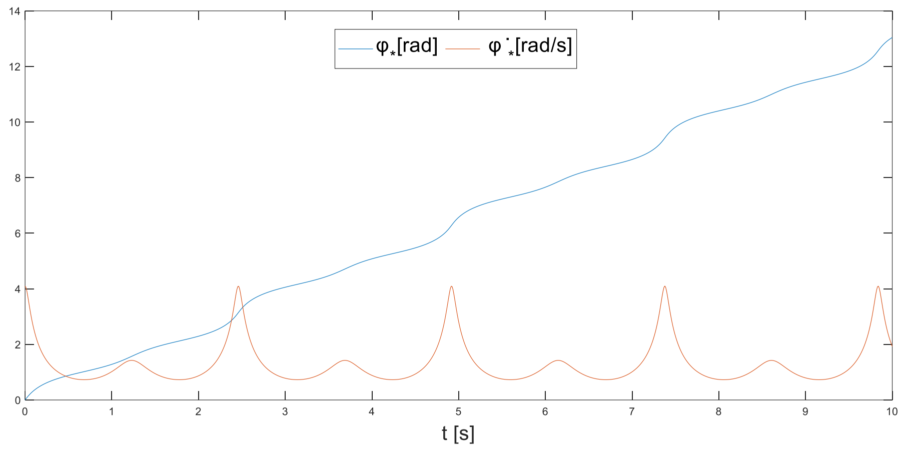

Figure 4.7 Case 2 evolution time – source: own elaboration.

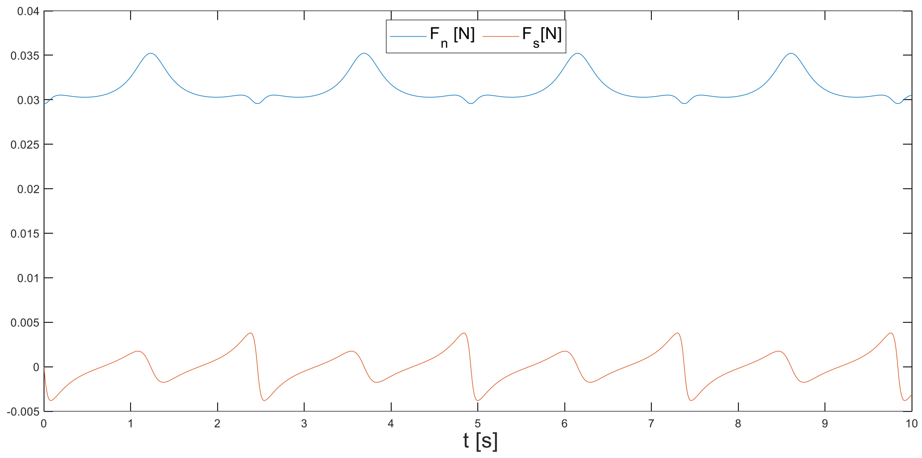

Figure 4.8 Case 2 Normal and friction force – source: own elaboration

As can be seen in Figure 4.8 the normal force F_n in this case 2 is positive and meets the requirements. And for this trajectory the value of μ that guarantees the no slip conditions is $\mu > 0.1263$. The period in this trajectory is $T = 4.8192\ seconds$.

As can be seen in Figure 4.8 the normal force F_n in this case 2 is positive and meets the requirements.

4.5.3 CASE 3:

In the third case, the trajectory described is a continuous rotation of the frame in a counter clockwise direction. For this propose is used:

$c = 0.49$ during all the trajectory.

Initial conditions: $\varphi_{0*} = 0\ rad$ and $\dot{\varphi}_{0*} = -4.3\ rad/s$

For this initial conditions and VHC, the results can be seen in Figures 4.9, 4.10, and 4.11. As can be seen in the Figure 4.11 the variable φ_* is continuously decreasing with time and that results in a periodical response of $\dot{\varphi}_*$ and u. This trajectory can be noticed that the input is right the opposite of the first case.

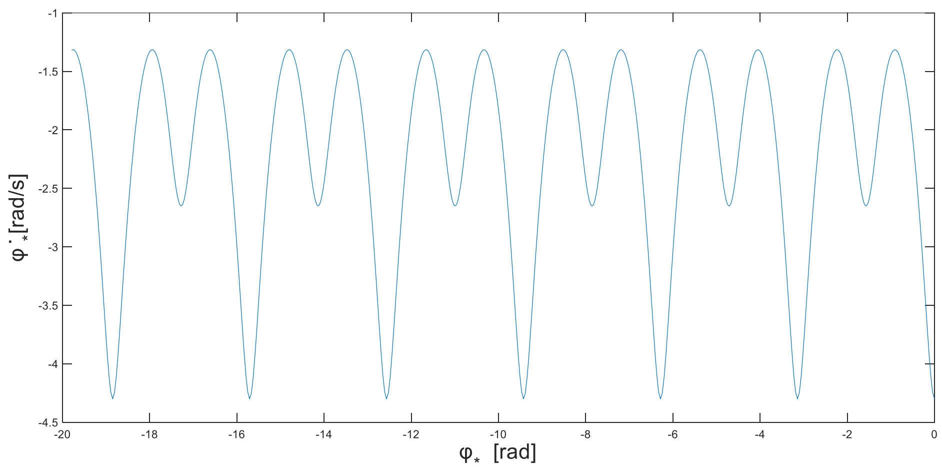

Figure 4.9 Case 3 trajectory – source: own elaboration.

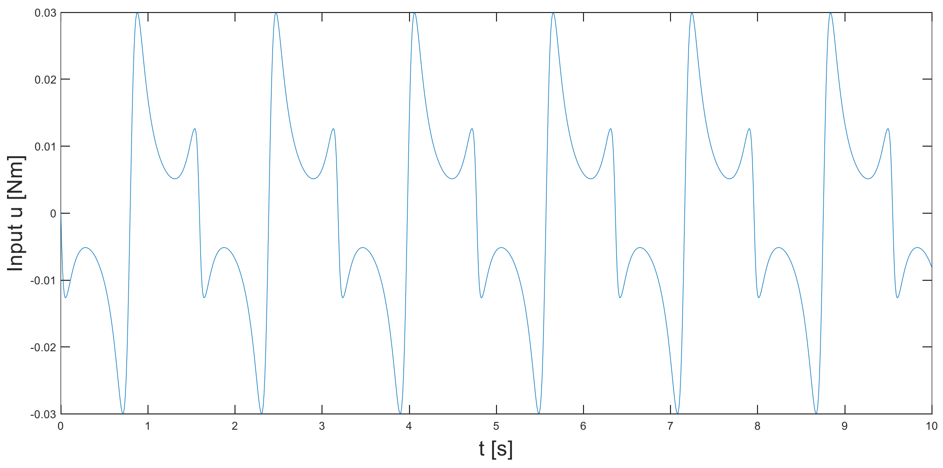

Figure 4.10 Case 3 input – source: own elaboration.

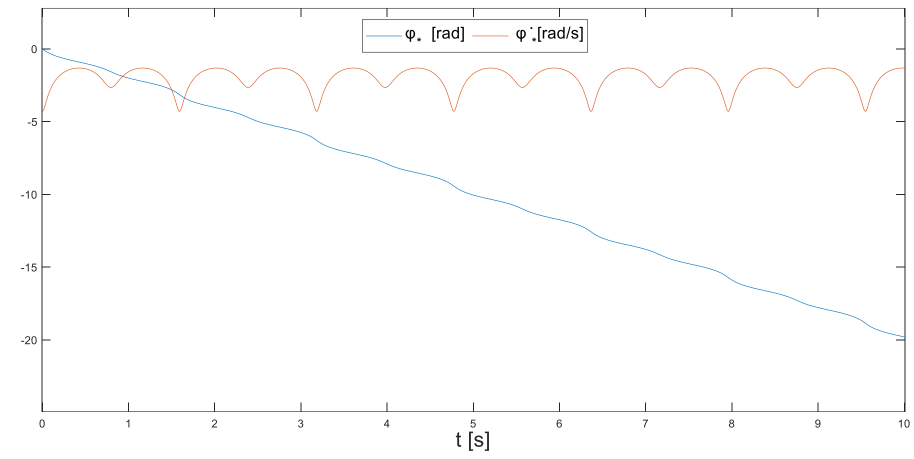

Figure 4.11 Case 3 evolution time – source: own elaboration.

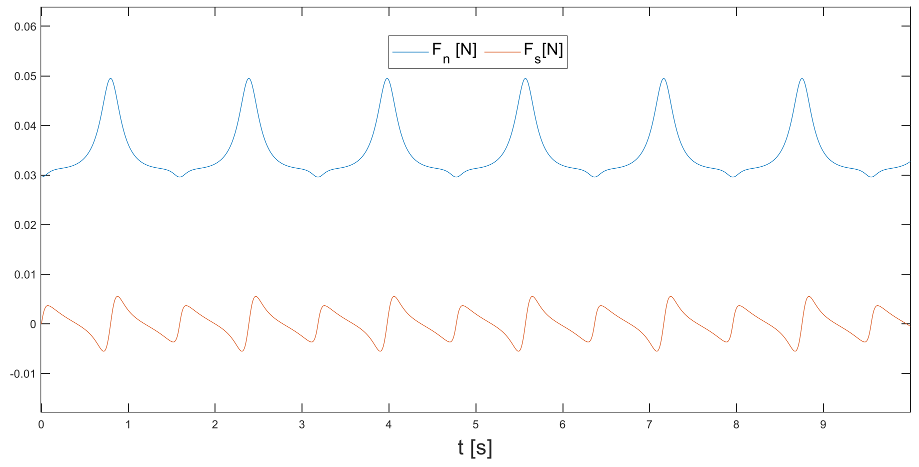

Figure 4.12 Case 3 Normal and friction force – source: own elaboration

As can be seen in Figure 4.12 the normal force F_n again is positive and meets the requirements. And for this trajectory the value of μ that guarantees the no slip conditions is $\mu > 0.1252$. The period in this trajectory is $T = 3.19$.

4.5.4 CASE 4:

In the last case, the trajectory describes an alternating rotation between clockwise direction and counter clockwise direction with a transition trajectory from one direction to the other. For this propose is used:

$c = 0.48$ during the laps, and $c = 0.45$ during the transition from one direction to the other.

Initial conditions: $\varphi_{0*} = 0 \ rad$ and $\dot{\varphi}_{0*} = 4.45 \ rad/s$

For this initial conditions and VHC, the results can be seen in Figure 4.13, 4.14 and 4.15. As can be seen in the Figure 4.15 the variable φ_* is continuously increasing and decreasing with time and that results in a periodical response of $\dot{\varphi}_*$ and u.

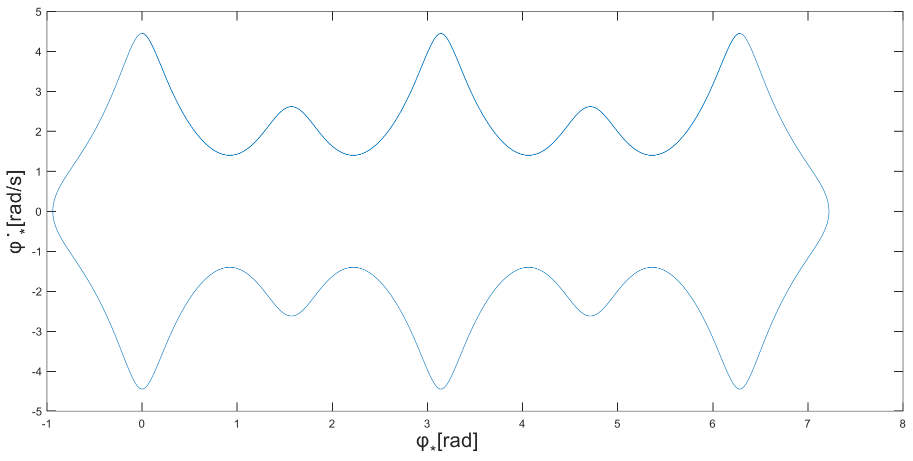

Figure 4.13 Case 4 trajectory – source: own elaboration.

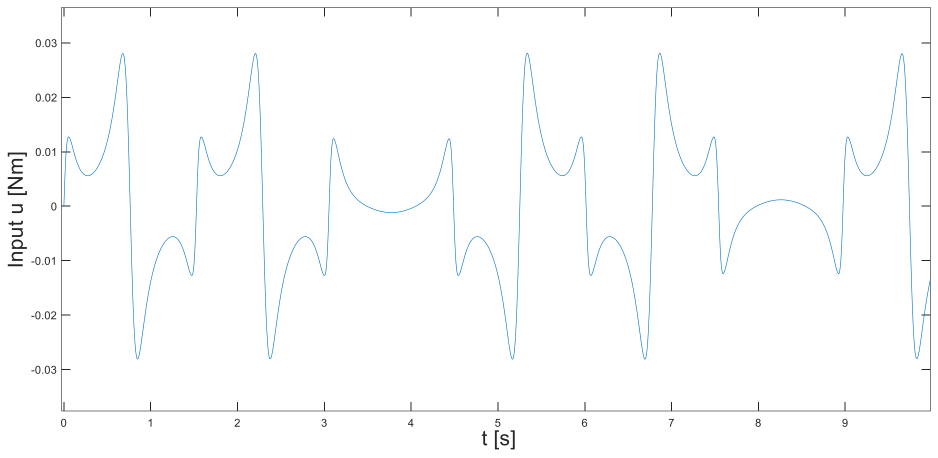

Figure 4.14 Case 4 input – source: own elaboration.

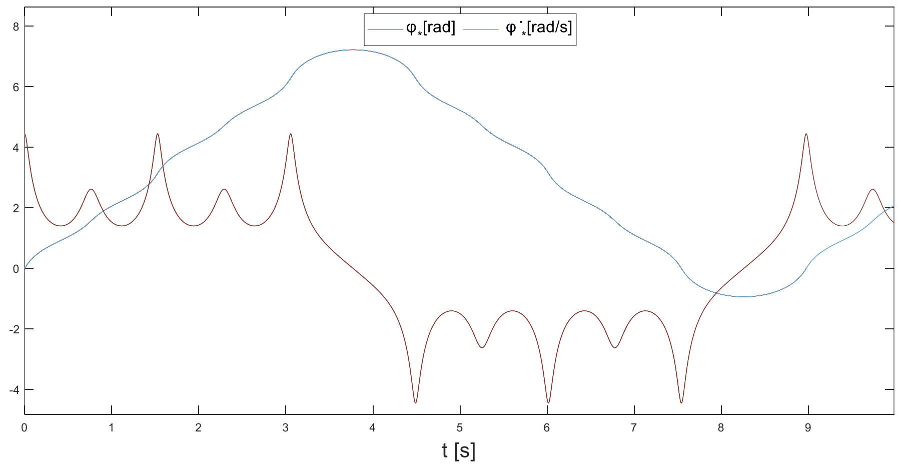

Figure 4.15 Case 4 evolution time – source: own elaboration.

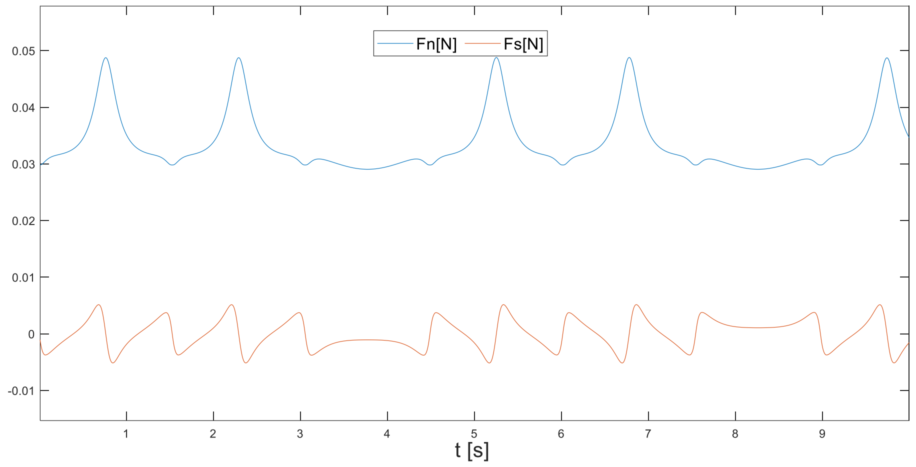

Figure 4.16 Case 4 Normal and friction force – source: own elaboration

As can be seen in Figure 4.16 the normal force F_n in the 4th case is positive and meets the requirements. And for this trajectory the value of μ that guarantees the no slip conditions is $\mu > 0.1397$. The period in this trajectory is $T = 8.97\ \ seconds$.

The friction coefficient μ for plastic materials, as is the case in the butterfly robot, is approximately 0.25-0.5. What means that during the simulations, there are no slip conditions in all the trajectories studied.

5 CONTROL AND IMPLEMENTATION

5.1 INTRODUCTION TO CONTROL AND IMPLEMENTATION

The thesis presented, has as objective the analysis of the modelling and control of the Butterfly Robot, an underactuated non-prehensile system.

At this point, a thorough study of the modelling of the system it is been done, then followed by developing different motions of feasible rolling, assuming some requirements.

Hence, the next logical step in this thesis is the development of the control of the system, a step by step method, so that the possible discrepancies between the theorical system and the physical system can be settled.

For this propose, a study and appliance of a stabilizing feedback controller is conducted. The justification of this section is based in the fact that the theorical behaviour and the real behaviour have some deviations that have to be solved. The motion planning in the previous section are unstable and the nature of this deviations are multiple, for example the assumptions at the beginning of the thesis, some simplifications of the theorical system and physical behaviour or imperfections in the shape of the two bodies that conform the motion.

Underactuated system are becoming more and more common in fields like robotics. The case of study is an underactuated system where the actuator can't be feedback linearized. This makes the control far more complex and requires an alternative way to perform it.

The orbital stabilization problem is commonly used to control complex problems where the number of actuators are less than the number of degrees of freedom as in the Butterfly robot. The problem becomes more complex when the system is more underactuated. In this particular case, the number of actuators is one, and is referred to the torque conducted in the centre of the frame. While the number of degrees of freedom has been reduced to two in the previous sections.

So in this section is presented a control based on orbital stability and transverse coordinates to solve this problem.

5.2 ORBITAL STABILIZATION

In this thesis the method used follows [1] and [5], known as the orbital stabilization based in transverse linearization. The traditional way of solving this problem is not possible and that's why is used the orbital stability that can be reduced into stability of transverse coordinates associated with the motion. For this propose is computed a transverse linearization, for a given motion.

5.3 MOVING PONCAIRÉ SECTION

To study the orbital stability is introduced the Moving Poincaré sections $S(\cdot)$. As described in [6] a Moving Poincaré section is a family of surfaces $S(\cdot)$ associated with a solution of the system $q_*(t)$. The orbit of the followed motion is sampled periodically as it intersects with the Poincaré sections as can be seen in the Figure 5.1.

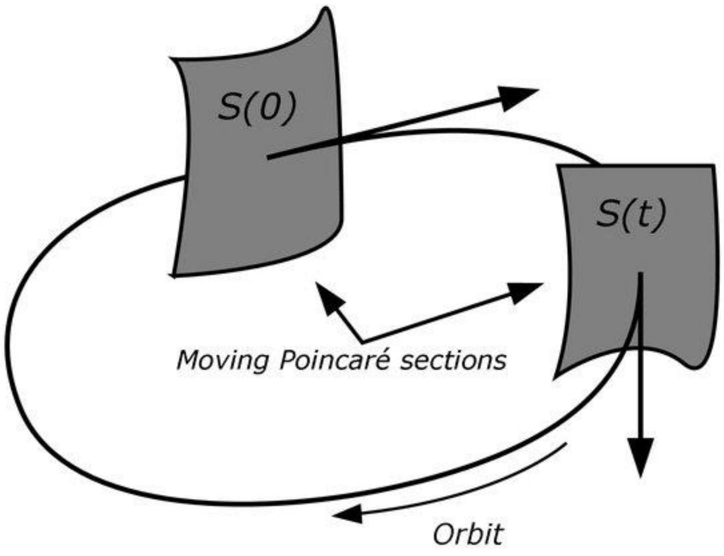

Figure 5.1 Moving Poincaré sections – source: own elaboration.

The surfaces $S(\cdot)$ locally intersects the orbit and its transversal to it. The local behaviour is studied in a proximity of the solution and therefore new variables have to be found so that the transversal linearization can be conducted.

At this point is introduced the scalar variables $\psi(t)$, describing the position along the trajectory of the motion planning, and the transversal coordinates $x_{\perp}$, that define the location on the surfaces $S(\cdot)$ of the dynamics transversal to the ideal trajectory solution $q_*(t)$.

5.4 CHANGING GENERALIZED COORDINATES TO TRANSVERSAL COORDINATES

From the Euler-Lagrange equations modelling the system was obtained its transformation to:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B(q)u \tag{5.1}$$

The generalized coordinates of the system and their derivative are:

$$\begin{aligned} q &= [\theta \quad \varphi]^T \\ \dot{q} &= [\dot{\theta} \quad \dot{\varphi}]^T \\ \ddot{q} &= [\ddot{\theta} \quad \ddot{\varphi}]^T \end{aligned} \tag{5.2}$$

Both degrees of freedom are linked by a VHC:

$$\theta_* = \Theta(\varphi_*)$$

$$\dot{\theta}_* = \Theta'(\varphi_*)\dot{\varphi}_*$$

$$\ddot{\theta}_* = \Theta''(\varphi_*)\dot{\varphi}_*{}^2 + \Theta'(\varphi_*)\ddot{\varphi}_* \tag{5.3}$$

It is been seen that this permits to rewrite the dynamics as:

$$\alpha(\varphi_*)\ddot{\varphi}_* + \beta(\varphi_*)\dot{\varphi}_*{}^2 + \gamma(\varphi_*) = 0 \tag{5.4}$$

As a result, in this format have been developed different motion planning as a solution of the system that can be described as:

$$q_*(t) = \begin{bmatrix}\theta_* \\ \varphi_*\end{bmatrix} = \begin{bmatrix}\Theta(\varphi_*) \\ \varphi_*\end{bmatrix} = \Phi(\varphi_*)$$

$$\dot{q}_*(t) = \begin{bmatrix}\dot{\theta}_* \\ \dot{\varphi}_*\end{bmatrix} = \begin{bmatrix}\Theta'(\varphi_*) \\ 1\end{bmatrix}\dot{\varphi}_* = \Phi'(\varphi_*)\dot{\varphi}_*$$

$$\ddot{q}_*(t) = \begin{bmatrix}\ddot{\theta}_* \\ \ddot{\varphi}_*\end{bmatrix} = \begin{bmatrix}\Theta''(\varphi_*) \\ 0\end{bmatrix}\dot{\varphi}_*{}^2 + \begin{bmatrix}\Theta'(\varphi_*) \\ 1\end{bmatrix}\ddot{\varphi}_* = \Phi''(\varphi_*)\dot{\varphi}_*{}^2 + \Phi'(\varphi_*)\ddot{\varphi}_* \tag{5.5}$$

The system (5.1) can be reformulated the next way as in [1], when a motion planning is defined by the control input variable:

$$u_*(\varphi, \dot{\varphi}) = \frac{\left[L^{-1}\left(N + M^{-1}CL\begin{pmatrix}0 \\ \dot{\varphi}\end{pmatrix}\right) + M^{-1}G\right]_1}{[L^{-1}M^{-1}]_{1,1}} \tag{5.6}$$

Where:

$$M = M(\Theta(\varphi), \varphi)$$

$$C = C(\Theta(\varphi), \varphi, \Theta'(\varphi)\dot{\varphi}, \dot{\varphi})$$

$$G = G(\Theta(\varphi), \varphi)$$

$$L = L(\varphi) = \begin{pmatrix}1 & \Theta'(\varphi) \\ 0 & 1\end{pmatrix}$$

$$N = N(\varphi) = \begin{pmatrix}\Theta''(\varphi)\dot{\varphi}^2 \\ 0\end{pmatrix} \tag{5.7}$$

Once the orbital solution it is been described, with it can be associated a moving Poincaré section $S(t)$. As was pointed in the previous section, a scalar variable is necessary to describe the position along the curve, it is presented as:

$$\psi(t) = \varphi(t) = \varphi_*(t) \tag{5.8}$$

The same way, to stablish the location on $S(t)$, and an alternative set of transversal coordinates linear independent are proposed:

$$x_\perp(q, \dot{q}) = \begin{bmatrix}y \\ \dot{y} \\ z\end{bmatrix} = \begin{bmatrix}\theta - \Theta(\varphi_*) \\ \dot{\theta} - \Theta'(\varphi_*)\dot{\varphi}_* \\ \dot{\varphi} - \dot{\varphi}_*(\varphi_*)\end{bmatrix} \tag{5.9}$$

Note that if the real process follows exactly the motion planning, $x_\perp(q, \dot{q}) = x_\perp(q_*, \dot{q}_*)$ then:

$$x_{\perp}(q_*,\dot{q}_*) = \begin{bmatrix} y \\ \dot{y} \\ z \end{bmatrix} = \begin{bmatrix} \theta(\varphi_*) - \theta(\varphi_*) \\ \theta'(\varphi_*)\dot{\varphi}_* - \theta'(\varphi_*)\dot{\varphi}_* \\ \dot{\varphi}_*(\varphi_*) - \dot{\varphi}_*(\varphi_*) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.10}$$

It has sense since $(q_*,\dot{q}_*)$ is the origin of the transversal coordinates of the system.

5.5 VIRTUAL CONTROL INPUT AND TRANSVERSE LINEARIATION

In this section is computed the transverse linearization, that is defined as the linearization of the dynamics of the transverse coordinates.

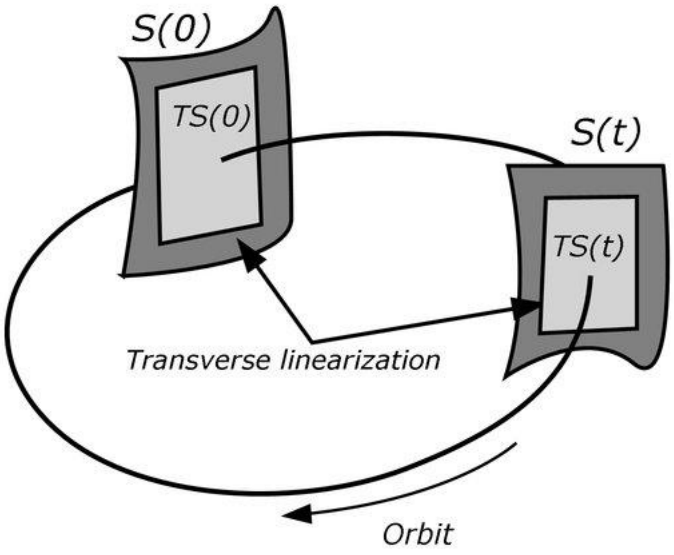

Figure 5.2 Transverse linearization – source: own elaboration.

Using the transverse coordinates $x_{\perp}$, can be computed a transvers linearization of the transverse dynamics above the system solution q_*. The dynamics of the mechanical system (5.1) is transformed into the transverse dynamics through the transverse coordinates. For this propose are derived the new set of transverse coordinates $x_{\perp}$ to obtain the transverse dynamics $\dot{x}_{\perp}$:

$$\dot{x}_{\perp}(q,\dot{q}) = \begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dot{z} \end{bmatrix} = \frac{dx_{\perp}}{dt} = \frac{dx_{\perp}}{dq}\frac{dq}{dt} + \frac{dx_{\perp}}{d\dot{q}}\frac{d\dot{q}}{dt} \tag{5.11}$$

At this point, is important to define the partial derivations:

$$\frac{dx_{\perp}}{dq} = \begin{bmatrix} \frac{dy}{dq} \\ \frac{d\dot{y}}{dq} \\ \frac{dz}{dq} \end{bmatrix} = \begin{bmatrix} \frac{dy}{d\theta} & \frac{dy}{d\varphi} \\ \frac{d\dot{y}}{d\theta} & \frac{d\dot{y}}{d\varphi} \\ \frac{dz}{d\theta} & \frac{dz}{d\varphi} \end{bmatrix} = \begin{bmatrix} 1 & -\theta' \\ 0 & -\theta''\dot{\varphi} \\ 0 & -\frac{d\dot{\varphi}_*}{d\varphi} \end{bmatrix} = \begin{bmatrix} 1 & -\theta' \\ 0 & -\theta''\left(z + \dot{\varphi}_*(\varphi_*)\right) \\ 0 & -\frac{d\dot{\varphi}_*}{d\varphi} \end{bmatrix} \tag{5.12}$$

$$\frac{dx_\perp}{d\dot{q}} = \begin{bmatrix} \frac{dy}{d\dot{q}} \\ \frac{d\dot{y}}{d\dot{q}} \\ \frac{dz}{d\dot{q}} \end{bmatrix} = \begin{bmatrix} \frac{dy}{d\dot{\theta}} & \frac{dy}{d\dot{\varphi}} \\ \frac{d\dot{y}}{d\dot{\theta}} & \frac{d\dot{y}}{d\dot{\varphi}} \\ \frac{dz}{d\dot{\theta}} & \frac{dz}{d\dot{\varphi}} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & -\Theta' \\ 0 & 1 \end{bmatrix} \tag{5.13}$$

Also are defined:

$$\frac{dq(x_\perp,\psi,u)}{dt} = \dot{q}(x_\perp,\psi,u) = \begin{bmatrix} \dot{\theta}(x_\perp,\psi,u) \\ \dot{\varphi}(x_\perp,\psi,u) \end{bmatrix} = \begin{bmatrix} \dot{y} + \Theta'\big(z + \dot{\varphi}_*(\varphi_*)\big) \\ z + \dot{\varphi}_*(\varphi_*) \end{bmatrix} \tag{5.14}$$

$$\frac{d\dot{q}(x_\perp,\psi,u)}{dt} = \ddot{q}(x_\perp,\psi,u) = \begin{bmatrix} \ddot{\theta}(x_\perp,\psi,u) \\ \ddot{\varphi}(x_\perp,\psi,u) \end{bmatrix} = M^{-1}\left(-C\dot{q}(x_\perp,\psi,u) - G + \begin{pmatrix} u \\ 0 \end{pmatrix}\right) \tag{5.15}$$

Now, the complete calculation of the transversal dynamics can be computed as:

$$\dot{x}_\perp(q,\dot{q}) = \begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} 1 & -\Theta' \\ 0 & -\Theta''\dot{\varphi} \\ 0 & -\frac{d\dot{\varphi}_*}{d\varphi} \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & -\Theta' \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \ddot{\theta} \\ \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} \dot{\theta} - \Theta'\dot{\varphi} \\ -\Theta''\dot{\varphi}^2 + \ddot{\theta} - \Theta'\ddot{\varphi} \\ -\frac{d\dot{\varphi}_*}{d\varphi}\dot{\varphi} + \ddot{\varphi} \end{bmatrix} =$$

$$= \begin{bmatrix} \dot{y} \\ -\Theta''(z+\dot{\varphi}_*)^2 + \ddot{\theta} - \Theta'\ddot{\varphi} \\ -\frac{d\dot{\varphi}_*}{d\varphi}(z+\dot{\varphi}_*) + \ddot{\varphi} \end{bmatrix} \tag{5.16}$$

Is introduced the virtual control input w, that is used as a feedback control law in the system to stabilize it, as proposed in [1]. Introducing this input, the system (5.6) would be rearranged to:

$$w = [L^{-1}M^{-1}]_{1,1}u - \left[L^{-1}\left(N + M^{-1}CL\begin{pmatrix} 0 \\ \dot{\varphi} \end{pmatrix}\right) + M^{-1}G\right]_1 \tag{5.17}$$

Introducing this new virtual control input w, into the dynamics $\dot{x}_\perp$, we obtain the following transformed system:

$$\ddot{y} = w \tag{5.18}$$

$$\dot{z} = \frac{1}{\alpha}\left(\frac{\gamma}{\dot{\varphi}_*}z - \beta\dot{\varphi}_* z - \beta z^2 + g_y y + g_{\dot{y}}\dot{y}\right) + \frac{g_w}{\alpha}w \tag{5.19}$$

Where:

$$g_y = -s' m_b g\left(\cos\left(\frac{\gamma}{2}+\theta\right), -\sin\left(\frac{\gamma}{2}+\theta\right)\right)\tau \ \ \mathrm{sinc}\frac{\gamma}{2}$$

$$g_{\dot{y}} = m_b s' \tau^T \rho(\dot{y} + 2\Theta'\dot{\varphi})$$

$$g_w = -s'\big(m_b\hat{k}\cdot(\vec{\rho}\times\vec{\tau}) + J_b p\big) \tag{5.20}$$

Then, the linearization around the solution q_* of the dynamics from above of the transverse coordinates $x_\perp$, can be described as:

$$\dot{x}_{\perp} = A(t)\ \ x_{\perp} + b(t)w \tag{5.21}$$

Where: $A(t) = A(\varphi_*) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ \frac{g_y}{\alpha} & \frac{g_{\dot{y}}}{\alpha} & \frac{\gamma - \beta\dot{\varphi}_*^2}{\alpha\dot{\varphi}_*} \end{bmatrix}$ and $b(t) = b(\varphi_*) = \begin{bmatrix} 0 \\ 1 \\ \frac{g_w}{\alpha} \end{bmatrix}$

The solution of the system (5.21) is the key to solve the problem of orbital stability for the motion planning $q_*(\varphi_*)$. Following the same procedure of [6], is possible to find a matrix gain $K(t)$, T periodic, that makes the feedback control w to stabilize the linear control system. This concept it is been developed from the minimization of a cost function. For more information about the procedure of how to arrive to this simplification, see [6]. The result would be the following:

$$w(t, x_{\perp}) = K(t)x_{\perp}(t) = -\frac{1}{\Gamma}b(t)^T R(t)x_{\perp} \tag{5.22}$$

Where the new variable $R(t)$, is a symmetric positive-semidefinite matrix 3x3 function, solution of the periodic Riccati differential equation (PRDE). The Riccati differential equation $\mathfrak{R}$ can be expressed as:

$$\mathfrak{R}(R,t) = \dot{R}(t) + A(t)^T R(t) + R(t)A(t) + Q(t) - R(t)B(t)\Gamma(t)^{-1}B(t)^T R(t) = 0 \tag{5.23}$$

With: $\Gamma > 0$ and $Q > 0$, both symmetric positive-definite matrices (SPD).

Once this equation is solved, the next step then would be to use the feedback virtual control input w, to compute the control input of the system $u(t)$ as follows:

$$u(t) = \frac{w + \left[L^{-1}\left(N + M^{-1}CL\begin{pmatrix}\dot{y} \\ \dot{\varphi}\end{pmatrix}\right) + M^{-1}G\right]_1}{[L^{-1}M^{-1}]_{1,1}} \tag{5.24}$$

With this equation, the orbital stabilisation arrives to its conclusion. In the next sections is explained how is solved the periodic Riccati differential equation and then how the control of the system is finally developed.

5.6 SOLUTION OF PERIODIC RICATTI DIFFERENTIAL EQUATION

A way to solve this equation has been studied in this section. The complication of this equation resides in the fact that there is not a reliable algorithm to solve it. For this reason, what it is been done is a simplification of the problem so that a feasible solution can be achieved. It is been followed the method and simplification of [6], [7] and [8].

The solution of this matrix is unique and can be computed in different ways. In this thesis, the PRDE problem is transformed in a semi-definite programming (SDP) problem, where a function is minimized and is subject to some linear matrix inequalities (LMI) constraints. This is possible because of the nature of the equation, where different periodic solutions $R(t)$ can be found for the inequality $\mathfrak{R}(R,t) > 0$, but only the maximation solution, $R^{+}(t)$ is the periodic stabilizing solution.

This way, the problem can be transformed into the next SDP problem:

$$\text{Minimize the function: } -\mathcal{J}(R(t))$$
$$\text{Subject to } S(R,t) \geq 0 \tag{5.25}$$

Where:

$$\mathcal{J}\big(R(t)\big) = tr\big(R(t)\big) \tag{5.26}$$

$$S(R,t) = \begin{bmatrix} \dot{R}(t) + A(t)^T R(t) + R(t)A(t) + Q(t) & R(t)B(t) \\ B(t)^T R(t) & \Gamma(t) \end{bmatrix} \geq 0 \tag{5.27}$$

The function tr means trace (sum of the diagonal coefficients of a matrix). By using the Schur complement, the problem $\Re(R,t) > 0$, its been transformed into the LMI $S(R,t) > 0$.

Still, remains the complication that this new definition of the problem, is an infinite problem and is necessary to transform it in a finite problem so that an analytical solution with existing numerical methods can be performed. For that propose, the SDP infinite problem can be reformulated as a SDP discrete finite problem as:

$$\text{Minimize the function: } -\mathcal{J}(\bar{R}(t))$$
$$\text{Subject to } S_j \geq 0 \quad for \;\; j = 1, \dots, L. \tag{5.28}$$

Where the function $\bar{R}(t)$, is a symmetric, T periodic trigonometric polynomial approximation of $R(t)$ with the period T of the system in each case. For the case of study, it is been decided to use as a , T periodic trigonometric polynomial the next:

$$\bar{R}(t) = R_0 + \textstyle\sum_{k=1}^{M}\big(\cos(k\omega t)\, R_{a,k} + \sin(k\omega t) R_{b,k}\big) \tag{5.29}$$

With $\omega = \frac{2\pi}{T}$ and a total of $2M + 1$ symmetric matrices. In this new formulation, L determinates the evenly spaced time samples that discretizes the problem. Now the problem can be defined as:

$$\mathcal{J}\big(\bar{R}(t)\big) = \textstyle\sum_{j=1}^{L} tr\big(R_0 + \sum_{k=1}^{M}\big(\cos(k\omega t_j)\, R_{a,k} + \sin(k\omega t_j) R_{b,k}\big)\big) \tag{5.30}$$

$$S_j\big(\bar{R}(t_j), t_j\big) = \begin{bmatrix} \dot{\bar{R}}(t_j) + A(t_j)^T \bar{R}(t_j) + \bar{R}(t_j)A(t_j) + Q(t_j) & \bar{R}(t_j)B(t_j) \\ B(t_j)^T \bar{R}(t_j) & \Gamma(t_j) \end{bmatrix} \geq 0 \tag{5.31}$$

Where $t_j = (j-1)\frac{T}{L}$ and $j = 1, \dots, L$.

To complete the definition of the problem, is necessary to add a set of LMI constraints that assures the boundedness on the polynomial $\bar{R}$. This new constraints are:

$$-dI_n \leq \bar{R}(t_j) \leq \; dI_n \tag{5.32}$$

With d, a constant value of choice. An infinite problem, it is been transformed in a finite problem with a finite number of matrices that can be computed. So the problem is reduced to get all the coefficients from R_0, $R_{a,k}$ and $R_{b,k}$.

5.7 FINDING THE ALGORITH

In this section, is explained how the problem is restructured to be solved through the SEDUMI tool for MATLAB. An algorithm is proposed in this thesis using the structures of SEDUMI to compute the problem of interest. After this, the feasibility and validity of this algorithm is checked to assure that the problem is solved correctly.

The SEDUMI tool for MATLAB have different ways to be used depending of the problem of interest. In this case, the structure from [9] that is going to be used and that this thesis uses to transform the problem, is the standard dual SDP problem:

$$\text{Minimize } c^T y$$

$$\text{Subject to: } F_0 + y_1 F_1 + \cdots + y_p F_p \geq 0 \tag{5.33}$$

With vector $y = [y_1 \quad y_2 \quad \cdots \quad y_p]^T$, and F_i are symmetric matrices.

It is been necessary then to adapt the SDP problem of interest to this structure. Considering that all the $2M+1$ incognito matrices are symmetric, the incognito coefficients are the diagonal and the superior triangle of each matrix, as can be seen in the figure (5.34).

$$R_0 = \begin{bmatrix} y_1 & y_2 & y_3 \\ y_2 & y_4 & y_5 \\ y_3 & y_5 & y_6 \end{bmatrix},\ R_{a,1} = \begin{bmatrix} y_7 & y_8 & y_9 \\ y_8 & y_{10} & y_{11} \\ y_9 & y_{11} & y_{12} \end{bmatrix}, \ldots,\ R_{b,M} = \begin{bmatrix} y_{p-5} & y_{p-4} & y_{p-3} \\ y_{p-4} & y_{p-2} & y_{p-1} \\ y_{p-3} & y_{p-1} & y_p \end{bmatrix} \tag{5.34}$$

Where the total number of incognito coefficients is $p = n\frac{n+1}{2}(2M+1)$, and where $n = 3$ is the size of the matrix $R\ nxn$. Then each y_i is an incognito coefficient multiplied by the vector c^T, necessary to represent the function $-\mathcal{J}(\bar{R}(t))$ to minimize, as follows:

$$-\mathcal{J}(\bar{R}(t)) = c^T y = c_1 y_1 + c_2 y_2 + \cdots + c_p y_p \tag{5.35}$$

From (5.30) is possible to compute:

$$c = [c_1, c_2, c_3, c_4 \ldots, c_7, c_8, c_9, c_{10} \ldots, c_{13}, c_{14}, c_{15}, c_{16}, \ldots, c_{p-3}, c_{p-2}, c_{p-1}, c_p] =$$
$$= \Big[L, 0,0, L, \ldots, \textstyle\sum_{j=1}^{L} \cos(\omega t_j), 0,0, \sum_{j=1}^{L} \cos(\omega t_j), \ldots, \sum_{j=1}^{L} \sin(\omega t_j), 0,0, \sum_{j=1}^{L} \sin(\omega t_j), \ldots,$$
$$\textstyle\sum_{j=1}^{L} \sin(M\omega t_j), 0,0, \sum_{j=1}^{L} \sin(M\omega t_j)\Big] \tag{5.36}$$

The case of the F_i matrices is more elaborated because the huge amount of LMI constraints that complete the SDP problem. At this point, the constraints of boundedness are divided in:

$$-dI_n \leq \bar{R}(t_j) \leq \ dI_n =$$
$$= \begin{cases} S_{L+j} = dI_n + \bar{R}(t_j) \geq 0 \\ S_{2L+j} = dI_n - \bar{R}(t_j) \geq 0 \end{cases} \tag{5.37}$$

For $j = 1, \ldots, L$.

With this division, each constraint $S_1, S_2, \dots, S_L, S_{L+1}, \dots, S_{3L}$ can be transformed into:

$$S_j = S_{j,0} + \sum_i^p S_{j,i} y_i = S_{j,0} + S_{j,1} y_1 + S_{j,2} y_2 + \cdots + S_{j,p} y_p \geq 0 \tag{5.38}$$

For $j = 1, \dots, 3L$ and $i = 1, \dots, p$. Where $[S_1, S_2, \dots, S_L]$ are matrices 4x4 as can be seen in (5.31) and $[S_{L+1}, S_{L+2}, \dots, S_{3L}]$ are 3x3 as can be seen in (5.37) for this control.

To develop the constraints, in necessary to define for $[S_1, S_2, \dots, S_{3L}]$ that can be divided into $[S_1, S_2, \dots, S_L]$, $[S_{L+1}, S_{L+2}, \dots, S_{2L}]$ and $[S_{2L+1}, S_{2L+2}, \dots, S_{3L}]$. This way their $S_{j,i}$, can be computed as:

For $[S_1, S_2, \dots, S_L]$ with $j = 1,2, \dots, L$:

$$S_{j,i} = \begin{bmatrix} \dot{\bar{R}}_i(t_j) + A(t_j)^T \bar{R}_i(t_j) + \bar{R}_i(t_j) A(t_j) & \bar{R}_i(t_j) B(t_j) \\ B(t_j)^T \bar{R}_i(t_j) & 0 \end{bmatrix} \tag{5.39}$$

$$S_{j,0} = \begin{bmatrix} Q(t_j) & 0 \\ 0 & \Gamma(t_j) \end{bmatrix} \tag{5.40}$$

For $[S_{L+1}, S_{L+2}, \dots, S_{2L}]$ and $j = L+1, L+2, \dots, 2L$:

$$S_{j,i} = \bar{R}_i(t_j) \tag{5.41}$$

$$S_{j,0} = S_{j,0L} = d[I_n] = d[I_3] \tag{5.42}$$

For $[S_{2L+1}, S_{2L+2}, \dots, S_{3L}]$ and $j = 2L+1, 2L+2, \dots, 3L$

$$S_{j,i} = -\bar{R}_i(t_j) \tag{5.43}$$

$$S_{j,0} = S_{j,0L} = d[I_n] = d[I_3] \tag{5.44}$$

Considering that:

$$\bar{R}(t_j) = \sum_{i=1}^p \bar{R}_i(t_j) y_i = \bar{R}_1(t_j) y_1 + \bar{R}_2(t_j) y_2 + \bar{R}_3(t_j) y_3 + \cdots + \bar{R}_7(t_j) y_7 + \cdots + \bar{R}_p(t_j) y_p =$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_1 + \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_2 + \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} y_3 + \cdots + \begin{bmatrix} \cos(\omega t_j) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} y_7 + \cdots +$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sin(M\omega t_j) \end{bmatrix} y_p \tag{5.45}$$

This transformation, allows to described from the standard dual SDP problem structure:

$$F_{i,j} = \begin{bmatrix} S_{j,i} & 0 & 0 \\ 0 & S_{L+j,i} & 0 \\ 0 & 0 & S_{2L+j,i} \end{bmatrix} \tag{5.46}$$

$$F_{\ i} = \begin{bmatrix} F_{i,1} & 0 & \cdots & 0 \\ 0 & F_{i,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & F_{i,L} \end{bmatrix} =$$

$$= \begin{bmatrix} \begin{bmatrix} S_{j,1} & 0 & 0 \\ 0 & S_{L+j,1} & 0 \\ 0 & 0 & S_{2L+j,1} \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \cdots & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} S_{j,2} & 0 & 0 \\ 0 & S_{L+j,2} & 0 \\ 0 & 0 & S_{2L+j,2} \end{bmatrix} & \cdots & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \vdots & \vdots & \ddots & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \begin{bmatrix} S_{j,p} & 0 & 0 \\ 0 & S_{L+j,p} & 0 \\ 0 & 0 & S_{2L+j,p} \end{bmatrix} \end{bmatrix} \tag{5.47}$$

For $j = 1, \dots, L$ and $i = 1, \dots, p$.

As can be seen, the problem of the algorithm is reduced into produce all this matrices formed by all the $S_{j,i}$.

The last element to define is F_0. This matrix, has the same size as the others and would be formed by the rest of the elements in the constraints that are not multiplied by any of the factors y_i.

$$F_{\ 0} = \begin{bmatrix} F_{0,1} & 0 & \cdots & 0 \\ 0 & F_{0,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & F_{0,L} \end{bmatrix} \tag{5.48}$$

With: $F_{0,j} = \begin{bmatrix} S_{j,0} & 0 & 0 \\ 0 & S_{j,0L} & 0 \\ 0 & 0 & S_{j,0L} \end{bmatrix}$ using (5.40)-(5.44).

This definition of all F_i, is correct since all the matrices of the diagonal have to be higher than 0, and thereby all the constraints meet the requirements if: $F_0 + y_1 F_1 + \cdots + y_p F_p \geq 0$. Finally, the equation for the constraints, can be rewritten as:

$$F_0 + y_1 F_1 + \cdots + y_p F_p =$$

$$= \begin{bmatrix} F_{0,1} & 0 & \cdots & 0 \\ 0 & F_{0,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & F_{0,L} \end{bmatrix} + \begin{bmatrix} F_{1,1} & 0 & \cdots & 0 \\ 0 & F_{1,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & F_{1,L} \end{bmatrix} y_1 + \cdots + \begin{bmatrix} F_{p,1} & 0 & \cdots & 0 \\ 0 & F_{p,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & F_{p,L} \end{bmatrix} y_p =$$

$$= \begin{bmatrix} S_1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & S_{L+1} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & S_{2L+1} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & S_L & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{L+L} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & S_{2L+L} \end{bmatrix} \geq 0 \tag{5.49}$$

The algorithm it is been developed using all the elements defined and using the correct values for d, M and L. The next step in the process would be to verify if the algorithm is acceptable to be used in the study control. All the MATLAB code performed for this algorithm, can be found at the final chapter MATLAB code.

5.8 VERIFYING THE ALGORITHM

In this section it is been tried to verify the validity of the algorithm performed, by using it in an example where the periodic stabilizing solution or maximation solution, $R^+(t)$, of a PRDE is known. That means, that the algorithm performed gets a theoretical stabilizing solution $R^+_{M,L}(t)$, and it is compared later with the real stabilising solution $R^+(t)$, to verify its validity as a problem solver for the study control. This process of verification it is been developed following [8] and using the same example to verify.

The idea is that from an algebraic Riccati equation (ARE) $\mathfrak{R}_c$, and its stabilizing solution R^+_c, using a T-periodic transform matrix function $P(t)$, is possible to compute the stabilizing solution $R^+(t)$, of the PRDE associated with $P(t)$ for later to compare with $R^+_{M,L}(t)$. The algebraic Riccati equation (ARE) $\mathfrak{R}_c$, can be solved directly with a function of MATLAB called 'are' and is defined as:

$$\mathfrak{R}_c = A_c^T R_c + R_c A_c + Q_c - R_c B_c \Gamma_c^{-1} B_c^T R_c = 0 \tag{5.50}$$

Once the stabilizing solution R^+_c of the ARE its been calculated, the next PRDE problem can be defined using $P(t)$:

$$\mathfrak{R}(R,t) = \dot{R}(t) + A(t)^T R(t) + R(t)A(t) + Q(t) - R(t)B(t)\Gamma(t)^{-1}B(t)^T R(t) = 0 \tag{5.51}$$

Where:

$$A(t) = P(t)^{-1} A_c P(t) - P(t)^{-1}\dot{P}(t)$$

$$B(t) = P(t)^{-1} B_c$$

$$Q(t) = P(t)^T Q_c P(t)$$

$$\Gamma(t) = \Gamma_c \tag{5.52}$$

And the stabilizing solution of this PRDE is defined as:

$$R^+(t) = P(t)^T R^+_c P(t) \tag{5.53}$$

In this thesis, following the example in [8], the matrices that have been used to compute the ARE problem are:

$$A_c = \begin{bmatrix} 4 & 3 \\ -4.5 & -3.5 \end{bmatrix}$$

$$B_c = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

$$Q_c = \begin{bmatrix} 10 & 6 \\ 6 & 4 \end{bmatrix}$$

$$\Gamma_c = 1 \tag{5.54}$$

And the T-periodic transform matrix function $P(t)$ is:

$$P(t) = \frac{1}{1.5+\cos(2\pi t)} \begin{bmatrix} \cos(2\pi t) & \sin(2\pi t) \\ -\sin(2\pi t) & \cos(2\pi t) \end{bmatrix} \tag{5.55}$$

As a result, the stabilizing solution for the ARE problem is:

$$R_c^+ = \begin{bmatrix} 22.4094 & 15.1092 \\ 15.1092 & 10.5412 \end{bmatrix} \tag{5.56}$$

And thereby, computing the stabilizing solution of the PRDE as (2.53), the evolution with time of its coefficients are:

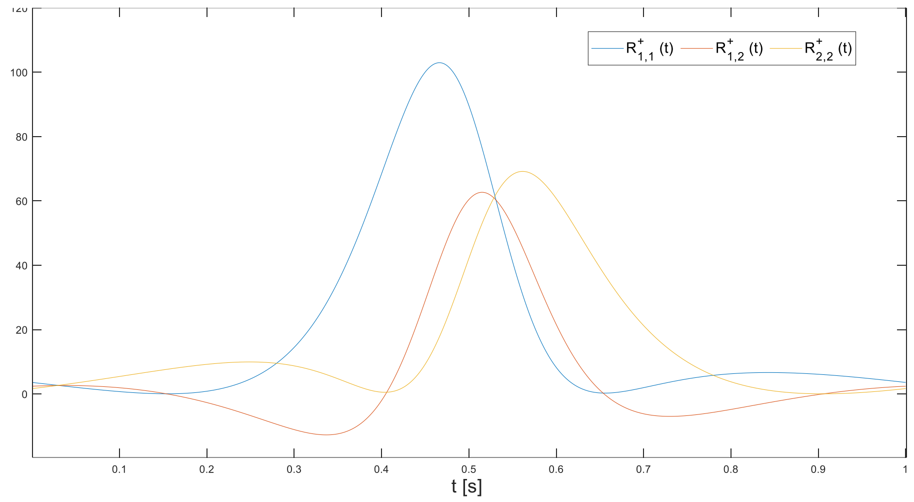

Figure 5.3 Stabilizing solution $R^+(t)$ of the PRDE – source: own elaboration.

Where:

$$R^+(t) = \begin{bmatrix} R^+_{1,1}(t) & R^+_{1,2}(t) \\ R^+_{2,1}(t) & R^+_{2,2}(t) \end{bmatrix} \tag{5.57}$$

The algorithm described in this section and developed in MATLAB under the name Check_Algorithm is used with $d = 200000$, $M = 30$ and $L = 100$. The result for this chosen values is:

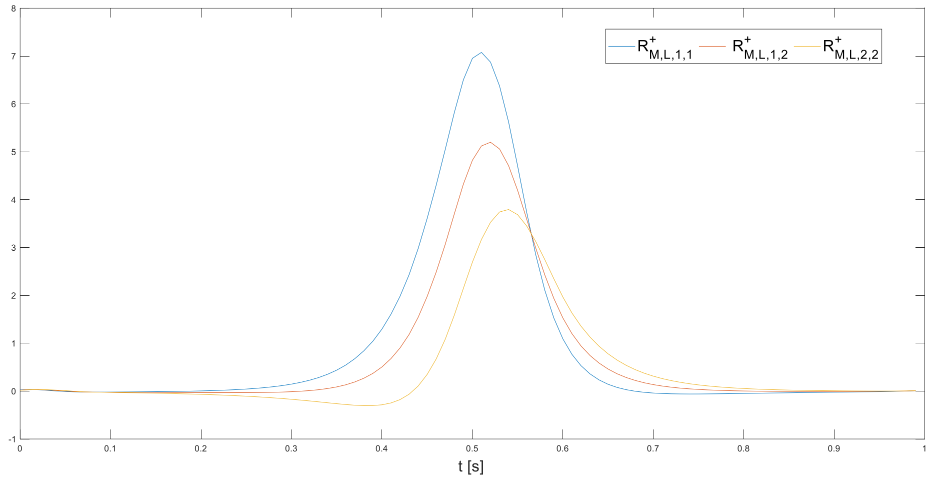

Figure 5.4 Algorithm result for the theoretical stabilizing solution $\boldsymbol{R^+_{M,L}(t)}$ ***– source: own elaboration.***

Where:

$$R^+_{M,L}(t) = \begin{bmatrix} R^+_{M,L,1,1}(t) & R^+_{M,L,1,2}(t) \\ R^+_{M,L,2,1}(t) & R^+_{M,L,2,2}(t) \end{bmatrix} \tag{5.58}$$

As can be seen, the results are not as good as expected, because the difference are quite considerable. So a change in the algorithm or a better approach would be necessary to develop successfully the control.

6 CONCLUSSION AND RECOMMENDATIONS

6. 1CONCLUSSION

The master thesis presented below has as objective the analysis of the modelling and control of an underactuated non-prehensile system known as the Butterfly robot. A robot that is commanded directly just with a rotational movement, and a ball controlled indirectly by the effect of gravity. A clear example of an underactuated non-prehensile control.

A thorough study of the modelling of the system it is been performed using previous studies, then followed by developing 4 different motions of feasible trajectories. Once the different motions have been settled, the control of the system for each motion it is been conducted, a step by step method, so that the possible differences between the theorical system and the real system could be settled. The motion planning in the previous section are unstable and the nature of this deviations are multiple.

The case of study is an underactuated system where the actuator can't be feedback linearized. This makes the control far more complex and requires an alternative way to perform it. For that propose it is been presented a control based on orbital stability and transverse coordinates to solve this problem. As a result a study and appliance of a stabilizing feedback controller it is been conducted.

The orbital stabilization problem is used to control the system of the Butterfly robot where the number of actuators is one, and is referred to the torque conducted in the centre of the frame, whilst the number of degrees of freedom has been reduced to two. To conduct the control it is been necessary to solve the PRDE, an equation where there is not a reliable algorithm to solve it. The problem it is been transformed into a simple semi-definite programming (SDP) problem, where a function is minimized and is subject to some linear matrix inequalities (LMI) constraints. .

The next step has been the explanation of how the problem is restructured to be solved through the SEDUMI tool for MATLAB. An algorithm is proposed in this thesis using the structures of SEDUMI to compute the problem of interest. After this, the feasibility and validity of this algorithm it is been checked with an example of solution known to assure that the problem is solved correctly.

The final result of the algorithm is not been as good as expected and so, it would require an even more deep study or a better approach of what it is been studied and performed would be necessary to develop successfully the control.

The part of control and the algorithm in the MATLAB code it is been elaborated by following a step by step explained process on how to transform the complex problem into something more simple.

6.2 RECOMMENDATIONS

This thesis it is been developed using the basis of other reports and going forward. For further work it would be recommended to follow this report, understand the explained concept, way of work and finally try to develop a better approach so that the control can be performed satisfactorily. The MATLAB code can be conducted in multiple ways and maybe a different form of thinking would be better to improve it.

Apart from that it would be interesting to go even forward and develop the control not just theoretically but also implementing it in the real system. In the report have been studied different cases for feasible trajectories but it is possible to create even more strangers and creative motions. Also different virtual holonomic constraints could be considered as long as it meets the requirements. Not just for the trajectories but also a different control based in orbital stability could be done. For example by a change of transverse coordinates.

From what it is been studied until this point, another assumptions could be taken so that the system is immersed in a total different environment as could be the effect of external perturbations, different shape of the objects or different friction coefficients, for example.

BIBLIOGRAPHY AND REFERENCES

[1] M. Surov, A. Shiriaev, L. Freidovich, S. Gusev, 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] O.R. Lund. 'Case study research: the Butterfly Robot', 2018.

[3] A.A.A. Vogels. 'The Butterfly Robot: Motion Planning', 2018.

[4] R.A. Adams, C. Essex. 'Calculus, a complete course', Seventh Edition, Chapter 11.4: 642-652, Pearson, 2010.

[5] A. Shiriaev, L. Freidovich, S. Gusev. 'Transverse linearization for controlled mechanical systems with several passive degrees of freedom', IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL55, NO. 4, 893-906, 2010.

[6] C.F. Sætre, 'Stable Gaits for an Underactuated Compass Biped Robot with a Torso, Trajectory Planning and Control Design using the Virtual Holonomic Constraints Approach', 2016.

[7] S. Gusev, S. Johansson, B. Kågström, A. Shiriaev, A. Varga, 'A numerical evaluation of solvers for the periodic Riccati differential equation', 302-329 2010.

[8] S. Gusev, S., A. Shiriaev, L. Freidovich, 'SDP-based approximation of stabilising solutions for periodic matrix Riccati differential equations', International Journal of Control, 1396-1405, 2016.

[9] Wu-Sheng Lu, 'Use SeDuMi to Solve LP, SDP and SCOP Problems: Remarks and Examples*', 2009.

[10] https://www.kamilgrzybek.com/design/grasp-explained/

APPENDIX

MATLAB CODE

The MATLAB code of the thesis is divided in three different parts.

The first part is dedicated to get the different motion planning with feasible trajectories and the inputs necessary for that propose. Is important to remark that the most part of this code it is been replicated from [3] with some modifications adapted to the propose of this section. This part of the code is formed by the next scripts:

get_motion123

This script is used to obtain the phase portrait for the different cases 1,2 and 3. The key resides in the different initial conditions that can adopt the script depending on the case.

get_motion4

This script is used to obtain the phase portrait for the different case 4.

get_prop

In this script are contained all the constant properties necessary to run the rest of the scripts.

get_spline

This scripts allows to write $\phi = f(\varphi)$.

get_dynamics

This script is used to compute the differential equations that permit to get φ and $\dot{\varphi}$.

get_par

This script is used to get all the parameters used as a function of φ.

get_u_Fn_Fs

This last script is used to get the actuator input and the normal and friction force.

The second part of the MATLAB code is dedicated to verify the validity of the algorithm used to control the system. It consists into try to solve the example problem, whose solution is known, with the designed algorithm and later compare both solutions graphically.

Check_Algorithm

This script is used to compare both real solution $R^{+}(t)$ and the solution obtained by the algorithm $R^{+}_{M,L}(t)$.

CheckC

This script is used to compute the vector c necessary to solve the problem with SEDUMI.

CheckR

This script is used to compute the matrix F necessary to solve the problem. It also obtains the vector y as the solution of the optimization problem that contains all its coefficients.

The third part of the MATLAB code is dedicated to implement the algorithm above to the problem that concerns in this thesis of the butterfly robot.

Control

This script receives the inputs from the real process and the algorithm solves the control problem and stablish the actuator u. This part of the script is supposed to be used once the control is implemented in the real process. It is been developed just theoretically since it is not been possible to access to the real system.

get_c

This script is similar to the script CheckC but adapted to the butterfly system.

get_R

This script is used as CheckR adapted to the butterfly system and computes all the possible solutions for the PRDE depending on φ.

```
%% Initialize
motion_case=1;
if motion_case==1
    T=3.19;
    x0 = [0;4.3];
elseif motion_case==2
    T=4.8192;
    x0 = [0;4.1];
elseif motion_case==3
    T=3.19;
    x0 = [0;-4.3];
end
get_prop;
get_spline;
options = odeset('RelTol',1e-5,'AbsTol',1e-6);
t0 = 0; tend = T; Nsim = 100;
tspan = linspace(t0,tend ,Nsim);

%% Simulation of dynamics
[t,x] = ode23(@(t,y)get_dynamics(t,y,prop),tspan ,x0,options);

varphi_nom = x(1:Nsim ,1); d_varphi_nom = x(1:Nsim ,2);
u_nom=zeros(1,Nsim); Fn_nom=zeros(1,Nsim); Fs_nom=zeros(1,Nsim);
for i=1:Nsim
    [u_nom(i),Fn_nom(i), Fs_nom(i)] = get_u_forces([varphi_nom(i); d_varphi_nom ↙
(i)],prop);
end
mu = 1/min(abs(Fn_nom./Fs_nom));

subplot (2,2,1), plot(x(1:100,1),x(1:100,2))
xlabel('varphi')
ylabel('d_varphi'),
subplot (2,2,2), plot(t,x)
xlabel('t [s]')
legend({'varphi[rad]','d_varphi [rad/s]'},'orientation','horizontal'),
subplot(2,2,3),plot(t(1:Nsim),u_nom)
xlabel('t [s]')
ylabel('Input u [Nm]');
subplot(2,2,4),plot(t(1:Nsim),Fn_nom)
hold on
plot(t(1:Nsim),Fs_nom)
hold off
xlabel('t [s]')
legend({'Fn[N]','Fs[N]'},'orientation','horizontal');
```

```
get_prop;
get_spline;

options = odeset('RelTol',1e-5,'AbsTol',1e-6);
t0 = 0; tend = 5; Nsim = 100;
tspan = linspace(t0,tend ,Nsim);
x0 = [0;4.45];
T=8.97;

prop.c=0.48;
[t,x] = ode23(@(t,y)get_dynamics(t,y,prop),tspan ,x0,options);
Nres1 =find(x(:,1) >2*pi,1);
varphi_nom1 = x(1:Nres1 ,1); d_varphi_nom1 = x(1:Nres1 ,2);
t1=t(1:Nres1);

prop.c=0.45;
x0=[x(Nres1+1,1),x(Nres1+1,2)];
tspan=linspace(t(Nres1+1),t(Nres1+1)+5,Nsim);
[t,x] = ode23(@(t,y)get_dynamics(t,y,prop),tspan ,x0,options);
Nres2 =find(x(:,1) <2*pi ,1);
varphi_nom2 = x(1:Nres2 ,1); d_varphi_nom2 = x(1:Nres2 ,2);
t2=t(1:Nres2);

prop.c=0.48;
x0=[x(Nres2+1,1),x(Nres2+1,2)];
tspan=linspace(t(Nres2+1),t(Nres2+1)+5,Nsim); %%%%%%t2 por t(Nres2+1)
[t,x] = ode23(@(t,y)get_dynamics(t,y,prop),tspan ,x0,options);
Nres3 =find(x(:,1) <0 ,1);
varphi_nom3 = x(1:Nres3 ,1); d_varphi_nom3 = x(1:Nres3 ,2);
t3=t(1:Nres3);

prop.c=0.45;
x0=[x(Nres3+1,1),x(Nres3+1,2)];
tspan=linspace(t(Nres3+1),t(Nres3+1)+5,Nsim);
[t,x] = ode23(@(t,y)sim_dynamics(t,y,prop),tspan ,x0,options);
Nres4 =find(x(:,1) >0 ,1);
varphi_nom4 = x(1:Nres4 ,1); d_varphi_nom4 = x(1:Nres4 ,2);
t4=t(1:Nres4);

prop.c=0.48;
x0=[x(Nres4+1,1),x(Nres4+1,2)];
tspan=linspace(t(Nres4+1),t(Nres4+1)+5,Nsim);
[t,x] = ode23(@(t,y)get_dynamics(t,y,prop),tspan ,x0,options);
Nres5 =find(x(:,1) >2*pi ,1);
varphi_nom5 = x(1:Nres5 ,1); d_varphi_nom5 = x(1:Nres5 ,2);
t5=t(1:Nres5);

varphi_nom=[varphi_nom1; varphi_nom2; varphi_nom3; varphi_nom4; varphi_nom5];
d_varphi_nom=[d_varphi_nom1; d_varphi_nom2; d_varphi_nom3; d_varphi_nom4;↙
d_varphi_nom5];
Nres=Nres1+Nres2+Nres3+Nres5;
t=[t1; t2; t3; t4; t5];
```

```
u_nom=zeros(1,Nres); Fn_nom=zeros(1,Nres); Fs_nom=zeros(1,Nres);
for i=1:Nres
    if i<=Nres1 || (i>Nres2 && i<=Nres3)|| (i>Nres4)
    prop.c=0.48;
    elseif (i>Nres1 && i<=Nres2) || (i>=Nres3&& i<=Nres4)
    prop.c=0.45;

    end
    [u_nom(i),Fn_nom(i), Fs_nom(i)] = get_u_forces([varphi_nom(i); d_varphi_nom↙
(i)],prop);

end

mu = 1/min(abs(Fn_nom./Fs_nom));

subplot (2,2,1), plot(varphi_nom,d_varphi_nom)
xlabel('varphi')
ylabel('d_varphi'),
subplot (2,2,2), plot(t,varphi_nom);
hold on
plot(t,d_varphi_nom);
xlabel('t [s]')
legend({'varphi [rad]','d_varphi [rad/s]'},'orientation','horizontal'),
subplot(2,2,3),plot(t(1:Nres),u_nom)
xlabel('t [s]')
ylabel('Input u [Nm]');
subplot(2,2,4),plot(t(1:Nres),Fn_nom)
hold on
plot(t(1:Nres),Fs_nom)
hold off
xlabel('t [s]')
legend({'Fn[N]','Fs[N]'},'orientation','horizontal')
```

```
%% Get all defined properties
prop.m_b = 3e-3; %mass of ball [kg]
prop.R_b = 16.55e-3; %ball radius [m]
prop.r_f = 12.5e-3; %distance between plates [m]
prop.R = sqrt(prop.R_b^2 - prop.r_f^2); %effective rolling radius [m]
prop.J_b = 5.48e-7; %moment of inertia of ball [kg*m^2]
prop.J_f = 1.581e-3; %moment of inertia of the frame [kg*m^2]
prop.grav = [0; 9.81; 0]; %gravitational acceleration [m*s^2]
prop.a = 0.1095; %bf frame
prop.b = 0.0405; %bf frame
prop.c = 0.49; %VHC parameter 0.49 ORIGINALLY
prop.k_hat = [0; 0; 1]; %z-axis (rotation)
prop.B = [1;0]; %Coupling matrix
prop.B_an = [0 1]; %Annihalitor matrix
prop.I = eye(3); %3x3 identity matrix
prop.Q = [0 1 0; -1 0 0; 0 0 0]; %Matrix used for differentiation
```

```
%% Define Spline function for phi = f(varphi)
N = 100; a_phi = linspace(0,2*pi,N);
g=zeros(1,N);
for i = 1:N
phi = a_phi(i);
delta = prop.a-prop.b*cos(2*phi);
d_delta = 2*prop.b*sin(2*phi);
alpha = atan((delta*sin(phi)-d_delta*cos(phi))/(delta*cos(phi)+d_delta*sin(phi)));
if phi > 0.5*pi && phi < 1.5*pi
    alpha = alpha+pi;
elseif phi >= 1.5*pi
    alpha = alpha+2*pi;
end
g(i) = atan2(delta*sin(phi)+prop.R*sin(alpha),delta*cos(phi)+prop.R*cos(alpha));
end
g = mod(g,2*pi);
g = unwrap(g*2)/2;
prop.phi_app = spline(g,a_phi);
```

```
function dx = sim_dynamics(t,x,prop)
varphi = x(1);
varphi= mod(varphi,2*pi);
d_varphi = x(2);
get_par

theta = Theta;
d_theta = d_Theta*d_varphi;

Pi = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
d_Pi = [-sin(theta) -cos(theta) 0; cos(theta) -sin(theta) 0; 0 0 0];

 m11 = prop.m_b*norm(rho)^2 + prop.J_f + prop.J_b;
 m12 = (dot(prop.m_b*prop.k_hat ,cross(rho,tau)) + prop.J_b*p)*d_s;
 m21 = m12;
 m22 = (prop.m_b + prop.J_b*p^2)*d_s^2;
 M = [m11 m12; m21 m22];

 c11 = dot(prop.m_b*d_s*rho,tau*d_varphi);
 c12 = dot(prop.m_b*d_s*rho,tau*d_theta)+d_varphi*((dot(prop.m_b*prop.k_hat ,cross ↙
(rho,tau))+prop.J_b*p)*dd_s + dot(prop.m_b*prop.k_hat ,cross(rho,kappa))*d_s^2);
 c21 = -dot(prop.m_b*d_s*rho,tau*d_theta);
 c22 = (prop.m_b+prop.J_b*p^2)*d_s*dd_s*d_varphi;
 C = [c11 c12; c21 c22];

 g1 = dot(prop.m_b*prop.grav ,d_Pi*rho);
 g2 = dot(prop.m_b*prop.grav ,Pi*tau*d_s);
 G = [g1;g2];
%% Get Alpha , Beta , Gamma
Alpha = m21*d_Theta + m22;
Beta1 = m21*dd_Theta;
Beta2 = c21*d_Theta + c22;
Gamma = g2;

dd_varphi = -(Beta1*d_varphi^2 + Beta2*d_varphi + Gamma)/Alpha;
dx = [d_varphi;dd_varphi];
end
```

```
%% Get all parameters
phi = ppval(prop.phi_app ,varphi);

delta = prop.a-prop.b*cos(2*phi);
d_delta = 2*prop.b*sin(2*phi);
dd_delta = 4*prop.b*cos(2*phi);
ddd_delta = -8*prop.b*sin(2*phi);

xi = [sin(phi); cos(phi);0];

delta_vec = delta*xi;
d_delta_vec = (d_delta*prop.I + delta*prop.Q)*xi;
dd_delta_vec = (dd_delta*prop.I + 2*d_delta*prop.Q - delta*prop.I)*xi;

alpha = atan( (delta*sin(phi) - d_delta*cos(phi)) / (delta*cos(phi) + d_delta*sin ↙
(phi)));
if phi > 0.5*pi && phi < 1.5*pi
    alpha = alpha+pi;
elseif phi >= 1.5*pi
    alpha = alpha+2*pi;
end

d_alpha = (2*d_delta^2 + delta^2 - dd_delta*delta)/(d_delta^2 + delta^2);
dd_alpha = (2*delta*d_delta -delta*ddd_delta+3*d_delta*dd_delta)/ ↙
(delta^2+d_delta^2) -(2*d_delta*(delta+dd_delta)*(delta^2- ↙
delta*dd_delta+2*d_delta^2))/((delta^2+d_delta^2)^2);

v1 = prop.R^2*dd_alpha+delta*(prop.R*sin(phi-alpha)+prop.R*sin(phi-alpha) ↙
*d_alpha^2+prop.R*cos(phi-alpha)*dd_alpha - 2*prop.R*sin(phi-alpha)*d_alpha)+prop. ↙
R*sin(phi-alpha)*dd_delta;
v2 = delta^2 + prop.R^2+2*prop.R*cos(phi-alpha)*delta;
v3 = (2*delta*d_delta+2*prop.R*cos(phi-alpha)*d_delta+2*prop.R*sin(phi-alpha) ↙
*delta*(d_alpha -1))*(prop.R^2*d_alpha -prop.R^2+prop.R*sin(phi-alpha) ↙
*d_delta+prop.R*cos(phi-alpha)*delta*(d_alpha -1));
v4 = v2^2;

g = atan2(delta*sin(phi)+prop.R*sin(alpha),delta*cos(phi)+prop.R*cos(alpha));
d_g = (delta^2+prop.R^2*d_alpha+prop.R*delta*cos(phi-alpha)+prop.R*d_delta*sin(phi- ↙
alpha)+prop.R*d_alpha*delta*cos(phi-alpha))/v2;
dd_g = v1/v2 - v3/v4;

n = [sin(alpha); cos(alpha);0];
d_n = [cos(alpha); -sin(alpha);0]*d_alpha;
dd_n = [-sin(alpha)*d_alpha^2+cos(alpha)*dd_alpha; -cos(alpha)*d_alpha^2-sin(alpha) ↙
*dd_alpha;0];

rho = delta_vec + prop.R*n;
h = d_delta_vec + prop.R*d_n;
d_h = dd_delta_vec + prop.R*dd_n;

tau = d_delta_vec/norm(d_delta_vec);

kappa_f = norm(cross(d_delta_vec ,dd_delta_vec))/(norm(d_delta_vec)^3);
```

```
kappa_c = kappa_f/(1-prop.R*kappa_f);
kappa = kappa_c*n;

p = 1/(prop.R*(prop.R*kappa_f -1));

d_s = norm(h)/d_g;
dd_s = dot(h,d_h)/(norm(h)*d_g^2) - norm(h)*dd_g/(d_g^3);

Theta = varphi -prop.c*sin(2*varphi); %%%%%   VHC   %%%%%
d_Theta = 1-2*prop.c*cos(2*varphi);
dd_Theta = 4*prop.c*sin(2*varphi);
```

```
function [u, Fn, Fs] = get_u_Fn_Fs(x,prop)
varphi = x(1);
varphi= mod(varphi,2*pi);
d_varphi = x(2);

get_par

theta = Theta;
d_theta = d_Theta*d_varphi;

Pi = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
d_Pi = [-sin(theta) -cos(theta) 0; cos(theta) -sin(theta) 0; 0 0 1];

%% Get MCG
m11 = prop.m_b*norm(rho)^2 + prop.J_f + prop.J_b;
m12 = (dot(prop.m_b*prop.k_hat ,cross(rho,tau)) + prop.J_b*p)*d_s;
m21 = m12;
m22 = (prop.m_b + prop.J_b*p^2)*d_s^2;
M = [m11 m12; m21 m22];

c11 = dot(prop.m_b*d_s*rho,tau*d_varphi);
c12 = dot(prop.m_b*d_s*rho,tau*d_theta)+d_varphi*((dot(prop.m_b*prop.k_hat ,cross ↙
(rho,tau))+prop.J_b*p)*dd_s + dot(prop.m_b*prop.k_hat ,cross(rho,kappa))*d_s^2);
c21 = -dot(prop.m_b*d_s*rho,tau*d_theta);
c22 = (prop.m_b+prop.J_b*p^2)*d_s*dd_s*d_varphi;
C = [c11 c12; c21 c22];

g1 = dot(prop.m_b*prop.grav ,d_Pi*rho);
g2 = dot(prop.m_b*prop.grav ,Pi*tau*d_s);
G = [g1;g2];

%% Get Alpha , Beta , Gamma
Alpha = m21*d_Theta + m22;
Beta1 = m21*dd_Theta;
Beta2 = c21*d_Theta + c22;
Gamma = g2;

dd_varphi = -(Beta1*d_varphi^2 + Beta2*d_varphi + Gamma)/Alpha;
dd_theta = dd_Theta*d_varphi^2 + d_Theta*dd_varphi;

%% get u and Fn and Fs
u = [m11 m12]*[dd_theta; dd_varphi] + [c11 c12]*[d_theta;d_varphi] + g1;
Fn = dot(prop.m_b*prop.k_hat ,cross(rho,n))*dd_theta + 0.5*dot(prop.m_b*rho,n) ↙
*d_theta^2 + 2*prop.m_b*d_s*d_theta*d_varphi+dot(prop.m_b*prop.grav ,Pi*n);
Fs = -(prop.J_b/prop.R)*(p*dd_s*d_varphi^2 + p*d_s*dd_varphi+dd_theta);
end
```

```
L=100;
M=30;
d=200000;
CheckC;
CheckR;

Ac=[4 3;-4.5 -3.5];
Bc=[1;-1];
Ghec=1;
Qc=[10 6;6 4];
%% Solving ARE problem
Ae=Ac;
Be=Bc*Ghec^(-1)*Bc.';
Ce=Qc;
Xe = are(Ae,Be,Ce);

T=1;
w=2*pi/T;
L1=L;
t_div=T/L1;
n=2;

X_11=zeros(1,L1);
X_12=zeros(1,L1);
X_22=zeros(1,L1);

t_f=zeros(1,L);
t_vec=zeros(1,L1);
for z=1:L1
    t=(z-1)*T/L1;
    t_vec(1,z)=t;
P=1/(1.5+cos(2*pi*t))*[cos(2*pi*t) sin(2*pi*t); -sin(2*pi*t) cos(2*pi*t)];
D_P=1/(1.5+cos(2*pi*t))^(2)*[(1.5+cos(2*pi*t))*(-2*pi*sin(2*pi*t))-(cos(2*pi*t))* ↙
(-2*pi*sin(2*pi*t)), (1.5+cos(2*pi*t))*(2*pi*cos(2*pi*t))-(-2*pi*sin(2*pi*t))*(sin ↙
(2*pi*t));-((1.5+cos(2*pi*t))*(2*pi*cos(2*pi*t))-(-2*pi*sin(2*pi*t))*(sin ↙
(2*pi*t))),(1.5+cos(2*pi*t))*(-2*pi*sin(2*pi*t))-(cos(2*pi*t))*(-2*pi*sin ↙
(2*pi*t))];

A=P^(-1)*Ac*P-P^(-1)*D_P;
B=P^(-1)*Bc;
Q=P.'*Qc*P;
R=Ghec;
X_sol=P.'*Xe*P;
X_11(1,z)=X_sol(1,1);
X_12(1,z)=X_sol(1,2);
X_22(1,z)=X_sol(2,2);
end

dat=y;
F_11=zeros(1,L);
F_12=zeros(1,L);
F_22=zeros(1,L);
```

```
F_Y2=zeros(2,2);
F_Y_aux3=zeros(2,2);
F_Y_aux4=zeros(2,2);

for j=1:L

    tj=(j-1)*T/L;
    t_f(1,j)=tj;
    for i=1:3
        if i==1
        Y_Y=[1 0 ;0 0];
        elseif i==2
        Y_Y=[0 1 ;1 0];
        elseif i==3
        Y_Y=[0 0;0 1];
        end
        dat_aux=dat(i,1);
        F_Y_aux4=dat_aux*Y_Y;
        F_Y2=F_Y2+F_Y_aux4;
        F_Y_aux4=0;
    end

    for k=0:(M-1)

    for i=4:9

         if i==4
        Y_Y=[cos((k+1)*w*(tj)) 0 ;0 0];
        elseif i==5
        Y_Y=[0 cos((k+1)*w*(tj));cos((k+1)*w*(tj)) 0];
        elseif i==6
        Y_Y=[0 0 ; 0 cos((k+1)*w*(tj))];
        elseif i==7
        Y_Y=[sin((k+1)*w*(tj)) 0; 0 0];
        elseif i==8
        Y_Y=[0 sin((k+1)*w*(tj));sin((k+1)*w*(tj)) 0];
        elseif i==9
        Y_Y=[0 0 ; 0 sin((k+1)*w*(tj))];
         end

        dat_aux=dat((n+1)*n*k+i,1);
        F_Y_aux3=dat_aux*Y_Y;
        F_Y2=F_Y2+F_Y_aux3;
        F_Y_aux3=0;

    end

    end

    F_11(1,j)=F_Y2(1,1);
    F_12(1,j)=F_Y2(2,1);
    F_22(1,j)=F_Y2(2,2);
    F_Y2=0;
```

```
end

subplot (2,2,1),
plot(t_vec(1,1:L1),X_11(1,1:L1))
hold on
plot(t_vec(1,1:L1),X_12(1,1:L1))
plot(t_vec(1,1:L1),X_22(1,1:L1))
legend({'[X11]',' [X12]','[X22]'},'orientation','horizontal'),
hold off

subplot (2,2,2),
plot(t_f(1,1:L),F_11(1,1:L))
hold on
plot(t_f(1,1:L),F_12(1,1:L))
plot(t_f(1,1:L),F_22(1,1:L))
legend({'[X11]',' [X12]','[X22]'},'orientation','horizontal'),
hold off
```

```
%% Creating C
n=2;
T=1;
w=2*pi/T;
coeficients=n*((n+1)/2)*(2*M+1);
c=zeros(coeficients,1);

for j=1:L
    tj=(j-1)*T/L;
    c(1)=c(1)+1;
    c(2)=0;
    c(3)=c(3)+1;
    for s=0:(M-1)
        c(3+s*6+1)=c(3+s*6+1)+cos((s+1)*w*(tj));
        c(3+s*6+2)=0;
        c(3+s*6+3)=c(3+s*6+3)+cos((s+1)*w*(tj));
        c(3+s*6+4)=c(3+s*6+4)+sin((s+1)*w*(tj));
        c(3+s*6+5)=0;
        c(3+s*6+6)=c(3+s*6+6)+sin((s+1)*w*(tj));
    end
end
c=-c;
```

```
%% OBTAIN F with aditional constrains.
n=2;
T=1;
w=2*pi/T;
coeficients=n*((n+1)/2)*(2*M+1);
At=zeros((3*n+1)*L*(3*n+1)*L,coeficients);

F=zeros((3*n+1)*L,(3*n+1)*L*coeficients);
F_0=zeros((3*n+1)*L,(3*n+1)*L);

Ac=[4 3;-4.5 -3.5];
Bc=[1;-1];
Ghec=1;
Qc=[10 6;6 4];

t_div=T/L;
t=0;

for j=0:(L-1)
t=(j)*T/L;
P=1/(1.5+cos(2*pi*t))*[cos(2*pi*t) sin(2*pi*t); -sin(2*pi*t) cos(2*pi*t)];
D_P=1/((1.5+cos(2*pi*t))^(2))*[(1.5+cos(2*pi*t))*(-2*pi*sin(2*pi*t))-(cos(2*pi*t))* ↙
(-2*pi*sin(2*pi*t)), (1.5+cos(2*pi*t))*(2*pi*cos(2*pi*t))-(-2*pi*sin(2*pi*t))*(sin ↙
(2*pi*t));-((1.5+cos(2*pi*t))*(2*pi*cos(2*pi*t))-(-2*pi*sin(2*pi*t))*(sin ↙
(2*pi*t))),(1.5+cos(2*pi*t))*(-2*pi*sin(2*pi*t))-(cos(2*pi*t))*(-2*pi*sin ↙
(2*pi*t))];

A=P^(-1)*Ac*P-P^(-1)*D_P;
A_T=A.';
B=P^(-1)*Bc;
B_T=B.';
Q=P.'*Qc*P;
Ghe=Ghec;

F_01=zeros(n,1);
F_02=zeros(1,n);
F_03=[Q F_01;F_02 Ghe];
F_04=[d*ones(n) zeros(n,n); zeros(n,n) d*ones(n)];
F_05=[F_03 zeros(n+1,2*n);zeros(2*n,n+1) F_04];
F_0((3*n+1)*j+1:(3*n+1)*j+(3*n+1),(3*n+1)*j+1:(3*n+1)*j+(3*n+1))=F_05;

    F_Y_aux1=zeros((n+1),(n+1));
    F_Y_aux2=zeros((2*n),(2*n));
    F_Y_aux=zeros((3*n+1),(3*n+1));
    F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    tj=(j)*T/L;
    for i=1:3
        if i==1
        Y_Y=[1 0 ;0 0];
        elseif i==2
        Y_Y=[0 1 ;1 0];
        elseif i==3
        Y_Y=[0 0;0 1];
        end
        F_Y_aux1=[A_T*Y_Y+Y_Y*A Y_Y*B; B_T*Y_Y 0];
        F_Y_aux2=[Y_Y zeros(n,n); zeros(n,n) -Y_Y];
        F_Y_aux=[F_Y_aux1 zeros(n+1,2*n);zeros(2*n,n+1) F_Y_aux2];
        F_Y((3*n+1)*j+1:(3*n+1)*j+(3*n+1),(3*n+1)*j+1:(3*n+1)*j+(3*n+1))= F_Y_aux;
        F(1:(3*n+1)*L,(3*n+1)*L*(i-1)+1:(3*n+1)*L*(i-1)+(3*n+1)*L)=F(1:(3*n+1)*L, ↙
(3*n+1)*L*(i-1)+1:(3*n+1)*L*(i-1)+(3*n+1)*L)+F_Y;
        F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    end
    for k=0:(M-1)
        F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    for i=4:9
         if i==4
        Y_Y=[cos((k+1)*w*(tj)) 0 ;0 0];
        D_Y=[-(k+1)*w*sin((k+1)*w*(tj)) 0;0 0];
        elseif i==5
        Y_Y=[0 cos((k+1)*w*(tj));cos((k+1)*w*(tj)) 0];
        D_Y=[0 -(k+1)*w*sin((k+1)*w*(tj));-(k+1)*w*sin((k+1)*w*(tj)) 0];
        elseif i==6
        Y_Y=[0 0 ; 0 cos((k+1)*w*(tj))];
        D_Y=[0 0 ; 0 -(k+1)*w*sin((k+1)*w*(tj))];
        elseif i==7
        Y_Y=[sin((k+1)*w*(tj)) 0; 0 0];
        D_Y=[(k+1)*w*cos((k+1)*w*(tj)) 0; 0 0];
        elseif i==8
        Y_Y=[0 sin((k+1)*w*(tj));sin((k+1)*w*(tj)) 0];
        D_Y=[0 (k+1)*w*cos((k+1)*w*(tj));(k+1)*w*cos((k+1)*w*(tj)) 0];
        elseif i==9
        Y_Y=[0 0 ; 0 sin((k+1)*w*(tj))];
        D_Y=[0 0 ; 0 (k+1)*w*cos((k+1)*w*(tj))];
         end
        F_Y_aux1=[D_Y+A_T*Y_Y+Y_Y*A Y_Y*B; B_T*Y_Y 0];
        F_Y_aux2=[Y_Y zeros(n,n); zeros(n,n) -Y_Y];
        F_Y_aux=[F_Y_aux1 zeros(n+1,2*n);zeros(2*n,n+1) F_Y_aux2];
        F_Y((3*n+1)*j+1:(3*n+1)*j+(3*n+1),(3*n+1)*j+1:(3*n+1)*j+(3*n+1))=F_Y_aux;
        F(1:(3*n+1)*L,(3*n+1)*L*(i-1)+6*(3*n+1)*L*k+1:(3*n+1)*L*(i-1)+6*(3*n+1) ↙
*L*k+(3*n+1)*L)=F(1:(3*n+1)*L,(3*n+1)*L*(i-1)+6*(3*n+1)*L*k+1:(3*n+1)*L*(i-1)+6* ↙
(3*n+1)*L*k+(3*n+1)*L)+F_Y;
        F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    end
    end
end

p=length(c);
bt=-c;
ct=vec(F_0);
for o=1:p
    At(:,o)=-vec(F(1:(3*n+1)*L,(3*n+1)*L*(o-1)+1:(3*n+1)*L*(o-1)+(3*n+1)*L));
end
K.s=size(F_0,1);
[x,y,info]=sedumi(At,bt,ct,K);
info;
```

```
%% Choose Case:1, 2, 3 or 4.
motion_case=1;
if motion_case==1 || motion_case==2 || motion_case==3
get_motion123;
elseif motion_case==4
get_motion4;
end
L_sim=100;
M_mat=40;
d=10000;
get_c;
get_R;

%% Wait for lecture of the real process:
%% varphi_meas
%% d_varphi_meas
%% theta_meas
%% d_theta_meas

Nval=find(varphi_nom >varphi_meas, 1);
p_sol=d_varphi_nom(Nval);
varphi=varphi_meas;
get_par;
Pi = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
d_Pi = [-sin(theta) -cos(theta) 0; cos(theta) -sin(theta) 0; 0 0 0];

m11 = prop.m_b*norm(rho)^2 + prop.J_f + prop.J_b;
m12 = (dot(prop.m_b*prop.k_hat ,cross(rho,tau)) + prop.J_b*p)*d_s;
m21 = m12;
m22 = (prop.m_b + prop.J_b*p^2)*d_s^2;
M = [m11 m12; m21 m22];

c11 = dot(prop.m_b*d_s*rho,tau*d_varphi);
c12 = dot(prop.m_b*d_s*rho,tau*d_theta)+d_varphi*((dot(prop.m_b*prop.k_hat ,cross ↙
(rho,tau))+prop.J_b*p)*dd_s + dot(prop.m_b*prop.k_hat ,cross(rho,kappa))*d_s^2);
c21 = -dot(prop.m_b*d_s*rho,tau*d_theta);
c22 = (prop.m_b+prop.J_b*p^2)*d_s*dd_s*d_varphi;
C = [c11 c12; c21 c22];

g1 = dot(prop.m_b*prop.grav ,d_Pi*rho);
g2 = dot(prop.m_b*prop.grav ,Pi*tau*d_s);
G = [g1;g2];

%% New Matrices L and N
L_sim=[1, d_Theta; 0, 1];
N_sim=[dd_Theta*p_sol^2; 0];

%% New Transverse coordinates
y=theta_meas-Theta;
d_y=d_theta_meas-d_Theta*p_sol;
z=d_varphi_meas-p_sol;

Zeta=[y; d_y; z];
```

```
B=b_w(1:3,Nval);
B_T=B.';
Q=eye(n);
Ghe=1;

R_solPRDE=[F_11(1,Nval) F_12(1,Nval) F_13(1,Nval);F_12(1,Nval) F_22(1,Nval) F_23(1,↙
Nval); F_13(1,Nval) F_23(1,Nval) F_33(1,Nval)];
virtual_w=-(1/Ghe)*B_T*R_solPRDE*Zeta;

Linv=inv(L_sim);
Minv=inv(M);

%% New Matrices

U1=Linv*(N_sim+Minv*C*L_sim*[d_y varphi]+Minv*G);
U2=Linv*Minv;

u=(virtual_w+U1(1,1))/U2(1,1);
```

```
L_sim=60;
M_mat=10;
d=10000;
T=3.19;
n=3;
w=2*pi/T;
coeficients=n*((n+1)/2)*(2*M_mat+1);
c=zeros(coeficients,1);

for j=1:L_sim
    tj=(j-1)*T/L_sim;
    c(1)=c(1)+1;
    c(2)=0;
    c(3)=0;
    c(4)=c(4)+1;
    c(5)=0;
    c(6)=c(6)+1;
    for s=0:(M_mat-1)
        c(6+s*12+1)=c(6+s*12+1)+cos((s+1)*w*(tj));
        c(6+s*12+2)=0;
        c(6+s*12+3)=0;
        c(6+s*12+4)=c(6+s*12+4)+cos((s+1)*w*(tj));
        c(6+s*12+5)=0;
        c(6+s*12+6)=c(6+s*12+6)+cos((s+1)*w*(tj));
        c(6+s*12+7)=c(6+s*12+7)+sin((s+1)*w*(tj));
        c(6+s*12+8)=0;
        c(6+s*12+9)=0;
        c(6+s*12+10)=c(6+s*12+10)+sin((s+1)*w*(tj));
        c(6+s*12+11)=0;
        c(6+s*12+12)=c(6+s*12+12)+sin((s+1)*w*(tj));
    end
end
c=-c;
```

```
%% OBTAIN R from F with aditional constraints.
get_c;
get_prop;
get_spline;

n=3;
w=2*pi/T;
coeficients=n*((n+1)/2)*(2*M_mat+1);

At=zeros((3*n+1)*L*(3*n+1)*L,coeficients);
F=zeros((3*n+1)*L,(3*n+1)*L*coeficients);
F_0=zeros((3*n+1)*L,(3*n+1)*L);
b_w=zeros(3,L);
for j=0:(L-1)

tj=(j)*T/L;

varphi=varphi_nom(j+1);
d_varphi=d_varphi_nom(j+1);
p_sol=d_varphi_nom(j+1);
get_par;
theta = Theta;
d_theta = d_Theta*d_varphi;
Pi = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];
d_Pi = [-sin(theta) -cos(theta) 0; cos(theta) -sin(theta) 0; 0 0 0];

 m11 = prop.m_b*norm(rho)^2 + prop.J_f + prop.J_b;
 m12 = (dot(prop.m_b*prop.k_hat ,cross(rho,tau)) + prop.J_b*p)*d_s;
 m21 = m12;
 m22 = (prop.m_b + prop.J_b*p^2)*d_s^2;
 M = [m11 m12; m21 m22];

 c11 = dot(prop.m_b*d_s*rho,tau*d_varphi);
 c12 = dot(prop.m_b*d_s*rho,tau*d_theta)+d_varphi*((dot(prop.m_b*prop.k_hat ,cross ↙
(rho,tau))+prop.J_b*p)*dd_s + dot(prop.m_b*prop.k_hat ,cross(rho,kappa))*d_s^2);
 c21 = -dot(prop.m_b*d_s*rho,tau*d_theta);
 c22 = (prop.m_b+prop.J_b*p^2)*d_s*dd_s*d_varphi;
 C = [c11 c12; c21 c22];

 g1 = dot(prop.m_b*prop.grav ,d_Pi*rho);
 g2 = dot(prop.m_b*prop.grav ,Pi*tau*d_s);
 G = [g1;g2];

Alpha = m21*d_Theta + m22;
Beta1 = m21*dd_Theta;
Beta2 = c21*d_Theta + c22;
Gamma = g2;
%% New Matrices
L=[1, d_Theta; 0, 1];
N=[dd_Theta*p_sol^2; 0];

%% Calculation of gy gd_y and gw
gw = -m12;
```

```
gdy = dot(prop.m_b*d_s*rho,(tau.')*(d_y+2*d_Theta*p_sol));
gy= dot(-d_s*prop.m_b*9.81*[cos(y/2+Theta), sin(y/2+Theta),0],tau*(sin(y/2)/↙
(y/2)));

%% Calculation of A and b.
Beta =  m21*dd_Theta-dot(prop.m_b*d_s*rho,tau)*d_Theta^2+(prop.m_b+prop.J_b*p^2) ↙
*d_s*dd_s;
A=[0, 1, 0; 0, 0, 0; gy/Alpha, gdy/Alpha, (Gamma-Beta*p_sol^2)/(Alpha*p_sol)];
b=[0; 1; gw/Alpha];
b_w(1:3,j)=b;
%% Calculate solution to Ricatti diferential equation.
A_T=A.';
B=b;
B_T=b.';
Q=eye(n);
Ghe=1;

%% Construction of F0
F_01=zeros(n,1);
F_02=zeros(1,n);
F_03=[Q F_01;F_02 Ghe];
F_04=[d*ones(n) zeros(n,n); zeros(n,n) d*ones(n)];
F_05=[F_03 zeros(n+1,2*n);zeros(2*n,n+1) F_04];
F_0((3*n+1)*j+1:(3*n+1)*j+(3*n+1),(3*n+1)*j+1:(3*n+1)*j+(3*n+1))=F_05;

%% Construction of F
    F_Y_aux1=zeros((n+1),(n+1));
    F_Y_aux2=zeros((2*n),(2*n));
    F_Y_aux=zeros((3*n+1),(3*n+1));
    F_Y=zeros((3*n+1)*L,(3*n+1)*L);

    for i=1:6
        if i==1
        Y_Y=[1 0 0;0 0 0;0 0 0];
        elseif i==2
        Y_Y=[0 1 0;1 0 0;0 0 0];
        elseif i==3
        Y_Y=[0 0 1;0 0 0;1 0 0];
        elseif i==4
        Y_Y=[0 0 0;0 1 0;0 0 0];
        elseif i==5
        Y_Y=[0 0 0;0 0 1;0 1 0];
        elseif i==6
        Y_Y=[0 0 0;0 0 0;0 0 1];
        end
        F_Y_aux1=[A_T*Y_Y+Y_Y*A Y_Y*B; B_T*Y_Y 0];
        F_Y_aux2=[Y_Y zeros(n,n); zeros(n,n) -Y_Y];
        F_Y_aux=[F_Y_aux1 zeros(n+1,2*n);zeros(2*n,n+1) F_Y_aux2];
        F_Y((3*n+1)*j+1:(3*n+1)*j+(3*n+1),(3*n+1)*j+1:(3*n+1)*j+(3*n+1))= F_Y_aux;
        F(1:(3*n+1)*L,(3*n+1)*L*(i-1)+1:(3*n+1)*L*(i-1)+(3*n+1)*L)=F(1:(3*n+1)*L, ↙
(3*n+1)*L*(i-1)+1:(3*n+1)*L*(i-1)+(3*n+1)*L)+F_Y;
        F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    end
    for k=0:(M_mat-1)
        F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    for i=7:18
        if i==7
        Y_Y=[cos((k+1)*w*(tj)) 0 0;0 0 0; 0 0 0];
        D_Y=[-(k+1)*w*sin((k+1)*w*(tj)) 0 0;0 0 0; 0 0 0];
        elseif i==8
        Y_Y=[0 cos((k+1)*w*(tj)) 0;cos((k+1)*w*(tj)) 0 0; 0 0 0];
        D_Y=[0 -(k+1)*w*sin((k+1)*w*(tj)) 0;-(k+1)*w*sin((k+1)*w*(tj)) 0 0; 0 0 0];
        elseif i==9
        Y_Y=[0 0 cos((k+1)*w*(tj));0 0 0; cos((k+1)*w*(tj)) 0 0];
        D_Y=[0 0 -(k+1)*w*sin((k+1)*w*(tj));0 0 0; -(k+1)*w*sin((k+1)*w*(tj)) 0 0];
        elseif i==10
        Y_Y=[0 0 0;0 cos((k+1)*w*(tj)) 0; 0 0 0];
        D_Y=[0 0 0;0 -(k+1)*w*sin((k+1)*w*(tj)) 0; 0 0 0];
        elseif i==11
        Y_Y=[0 0 0;0 0 cos((k+1)*w*(tj)); 0 cos((k+1)*w*(tj)) 0];
        D_Y=[0 0 0;0 0 -(k+1)*w*sin((k+1)*w*(tj)); 0 -(k+1)*w*sin((k+1)*w*(tj)) 0];
        elseif i==12
        Y_Y=[0 0 0;0 0 0; 0 0 cos((k+1)*w*(tj))];
        D_Y=[0 0 0;0 0 0; 0 0 -(k+1)*w*sin((k+1)*w*(tj))];
        elseif i==13
        Y_Y=[sin((k+1)*w*(tj)) 0 0;0 0 0; 0 0 0];
        D_Y=[(k+1)*w*cos((k+1)*w*(tj)) 0 0;0 0 0; 0 0 0];
        elseif i==14
        Y_Y=[0 sin((k+1)*w*(tj)) 0;sin((k+1)*w*(tj)) 0 0; 0 0 0];
        D_Y=[0 (k+1)*w*cos((k+1)*w*(tj)) 0;(k+1)*w*cos((k+1)*w*(tj)) 0 0; 0 0 0];
        elseif i==15
        Y_Y=[0 0 sin((k+1)*w*(tj));0 0 0; sin((k+1)*w*(tj)) 0 0];
        D_Y=[0 0 (k+1)*w*cos((k+1)*w*(tj));0 0 0; (k+1)*w*cos((k+1)*w*(tj)) 0 0];
        elseif i==16
        Y_Y=[0 0 0;0 sin((k+1)*w*(tj)) 0; 0 0 0];
         D_Y=[0 0 0;0 (k+1)*w*cos((k+1)*w*(tj)) 0; 0 0 0];
        elseif i==17
        Y_Y=[0 0 0;0 0 sin((k+1)*w*(tj)); 0 sin((k+1)*w*(tj)) 0];
        D_Y=[0 0 0;0 0 (k+1)*w*cos((k+1)*w*(tj)); 0 (k+1)*w*cos((k+1)*w*(tj)) 0];
        elseif i==18
        Y_Y=[0 0 0;0 0 0; 0 0 sin((k+1)*w*(tj))];
        D_Y=[0 0 0;0 0 0; 0 0 (k+1)*w*cos((k+1)*w*(tj))];

         end

        F_Y_aux1=[D_Y+A_T*Y_Y+Y_Y*A Y_Y*B; B_T*Y_Y 0];
        F_Y_aux2=[Y_Y zeros(n,n); zeros(n,n) -Y_Y];
        F_Y_aux=[F_Y_aux1 zeros(n+1,2*n);zeros(2*n,n+1) F_Y_aux2];
        F_Y((3*n+1)*j+1:(3*n+1)*j+(3*n+1),(3*n+1)*j+1:(3*n+1)*j+(3*n+1))=F_Y_aux;
        F(1:(3*n+1)*L,(3*n+1)*L*(i-1)+12*(3*n+1)*L*k+1:(3*n+1)*L*(i-1)+12*(3*n+1) ↙
*L*k+(3*n+1)*L)=F(1:(3*n+1)*L,(3*n+1)*L*(i-1)+12*(3*n+1)*L*k+1:(3*n+1)*L*(i-1)+12* ↙
(3*n+1)*L*k+(3*n+1)*L)+F_Y;
        F_Y=zeros((3*n+1)*L,(3*n+1)*L);
    end

    end
```

```
end

p=length(c);
bt=-c;
ct=vec(F_0);
for o=1:p
    At(:,o)=-vec(F(1:(3*n+1)*L,(3*n+1)*L*(o-1)+1:(3*n+1)*L*(o-1)+(3*n+1)*L));
end

K.s=size(F_0,1);
[x,y,info]=sedumi(At,bt,ct,K);
info;

%% Construction of R_sol of PRDE
dat=y;
F_11=zeros(1,L);
F_12=zeros(1,L);
F_13=zeros(1,L);
F_22=zeros(1,L);
F_23=zeros(1,L);
F_33=zeros(1,L);

F_Y=zeros(3,3);
F_Y_aux1=zeros(3,3);
F_Y_aux=zeros(3,3);

for j=1:L

    tj=(j-1)*T/L;
    for i=1:6
        if i==1
        Y_Y=[1 0 0;0 0 0;0 0 0];
        elseif i==2
        Y_Y=[0 1 0;1 0 0;0 0 0];
        elseif i==3
        Y_Y=[0 0 1;0 0 0;1 0 0];
        elseif i==4
        Y_Y=[0 0 0;0 1 0;0 0 0];
        elseif i==5
        Y_Y=[0 0 0;0 0 1;0 1 0];
        elseif i==6
        Y_Y=[0 0 0;0 0 0;0 0 1];
        end
        dat_aux=dat(i,1);

        F_Y_aux=dat_aux*Y_Y;
        F_Y=F_Y+F_Y_aux;

    end
    for k=0:(M_mat-1)
         F_Y_aux=0;
    for i=7:18
```

```
            if i==7
            Y_Y=[cos((k+1)*w*(tj)) 0 0;0 0 0; 0 0 0];
            elseif i==8
            Y_Y=[0 cos((k+1)*w*(tj)) 0;cos((k+1)*w*(tj)) 0 0; 0 0 0];
            elseif i==9
            Y_Y=[0 0 cos((k+1)*w*(tj));0 0 0; cos((k+1)*w*(tj)) 0 0];
            elseif i==10
            Y_Y=[0 0 0;0 cos((k+1)*w*(tj)) 0; 0 0 0];
            elseif i==11
            Y_Y=[0 0 0;0 0 cos((k+1)*w*(tj)); 0 cos((k+1)*w*(tj)) 0];
            elseif i==12
            Y_Y=[0 0 0;0 0 0; 0 0 cos((k+1)*w*(tj))];
            elseif i==13
            Y_Y=[sin((k+1)*w*(tj)) 0 0;0 0 0; 0 0 0];
            elseif i==14
            Y_Y=[0 sin((k+1)*w*(tj)) 0;sin((k+1)*w*(tj)) 0 0; 0 0 0];
            elseif i==15
            Y_Y=[0 0 sin((k+1)*w*(tj));0 0 0; sin((k+1)*w*(tj)) 0 0];
            elseif i==16
            Y_Y=[0 0 0;0 sin((k+1)*w*(tj)) 0; 0 0 0];
            elseif i==17
            Y_Y=[0 0 0;0 0 sin((k+1)*w*(tj)); 0 sin((k+1)*w*(tj)) 0];
            elseif i==18
            Y_Y=[0 0 0;0 0 0; 0 0 sin((k+1)*w*(tj))];
            end
            dat_aux=dat((n+1)*n*k+i,1);

            F_Y_aux1=dat_aux*Y_Y;
            F_Y_aux=F_Y_aux+F_Y_aux1;
            F_Y_aux1=0;

        end

        F_Y=F_Y+F_Y_aux;

        end
        F_11(1,j)=F_Y(1,1);
        F_12(1,j)=F_Y(1,2);
        F_13(1,j)=F_Y(1,3);
        F_22(1,j)=F_Y(2,2);
        F_23(1,j)=F_Y(2,3);
        F_33(1,j)=F_Y(3,3);

        F_Y=0;
    end
```