POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria dell'Automazione
Dipartimento di Elettronica e Informazione

ANALYSIS AND CONTROL OF A MOBILE INVERTED PENDULUM IN A SIMULATED ENVIRNONMENT

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Supervisor: Prof. Andrea Bonarini

Master thesis by:
Pablo Lara Peinado, matricola 877233

Academic year 2017 – 2018

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis supervisor Prof. Andrea Bonarini for his continuous guidance, support and enthusiasm.

I am also deeply thankful to all my friends who have helped me these years in Milano, whose support has been fundamental all this time, and those who despite of being far have also contributed.

Last but not least, I would like to express my very enormous gratitude to the key people who made all of this even possible: my parents, Antonio and Antonia; my brother Javier and my girlfriend Amparo. Thank you for the unconditional support, advices and the continuous encouragement to never give up and to try to do my best.

ABSTRACT

This thesis, is focused on analysing and modelling a mobile inverted pendulum, based on a real robot developed in AI&R Lab of the Politecnico di Milano.

In addition, the control of the system is designed in order to maintain the robot stable and vertical during its movement and allow it to be resistant to unexpected perturbances. Different approaches and techniques have been followed with the objective to find the most optimal and efficient controller.

Each controller has been tested in a simulation environment of Matlab/Simulink, providing reliable results that allow the comparison and the efficiency of each method.

.

CONTENT INDEX

FIGURES INDEX

TABLE INDEX

CHAPTER 1: INTRODUCTION

Nowadays, human cooperation with robots in a domestic environment is an increasing research field in the last years, with several companies and universities continuously studying and investigating in this area of knowledge, allowing robotics to become one of the most important sector and with one of the brightest futures.

Due to its structural simplicity, mobility, low economical cost and energy efficiency, self-balancing robots are a great solution for domestic environment applications, as well as human interaction games.

The objective of the thesis is to model and study a balancing robot, designing controllers that will guarantee the stability of the vehicle, as well as the capacity to follow a desired trajectory.

The presented work includes the whole development of the control system of the vehicle, from the mathematical modelling of the system, the analysis and study of the different control methods that could be applied, and to the testing and simulation of each approach.

1.1. Structure of the thesis

In chapter 2, a synthesis of the consulted literature is listed, including past solutions to the problem and articles regarding the addressed topic.

In chapter 3, the theoretical study is described, with a detailed summary of the robot and the computation of the mathematical model of the system.

In chapter 4, the different analysed techniques are presented, with three different classic approaches.

In chapter 5, each method is implemented and tested in simulation, allowing the result discussion and comparison between techniques.

In chapter 6, the conclusions are stated, and the possible future work is discussed in order to improve and complete the current study.

CHAPTER 2:
STATE OF ART

During this century, robotics has suffered an exponentially growth, being present in a wide range of fields: home automation, industrial processes, space exploration, military, automated vehicles, etc... It is a field in constant change, with more discoveries and improvements being made each year.

One of the many advantages of robots is that they can substitute human labour, realizing activities that could be dangerous for humans, and do it in a more efficient and automated manner.

Between all the numerous work sectors of robotics, mobile robots adapted to operate in domestic environments is a prominent discipline. These robots must be able to interact with humans and cooperate with them.

Since 1969 there have been developing robots that could interact with their environment. This is the case of *Shakey*, a mobile robot system created by the Artificial Intelligence Centre, equipped with a TV camera and sensors, allowing him to perceive and model its environment [1].

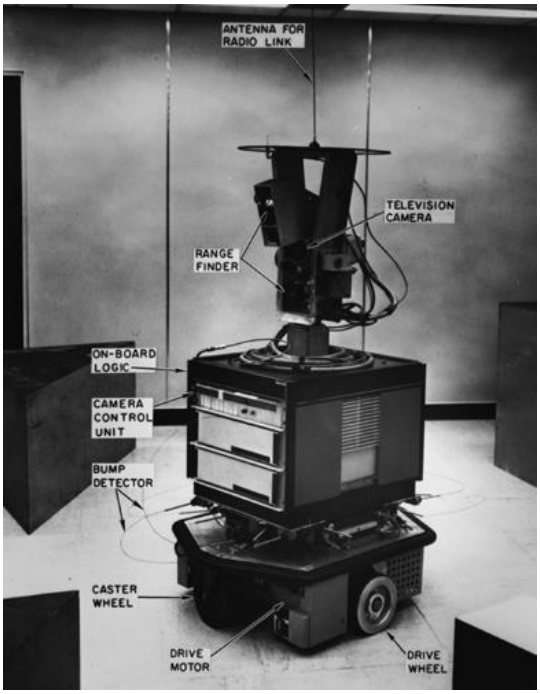

Figure 1. Shakey - Mobile robot created in 1969

Although there is much work that can be considered HRI (human-robot interaction), the multidisciplinary field started to emerge in the mid-1990s and early years of 2000 [2].

From that point, different robot with diverse structures have been studied to create robots that could operate in domestic environments, from simple robots with specific task, such as cleaning the floor (Figure 2) or more sophisticated like PR2 robot (Figure 3), which can grab a beer in the fridge and bring it to a certain location [3].

Figure 2. iRobot Roomba 560 Vacuum Cleaner

Figure 3. PR2, Robot for Research and Innovation

Typically, it has been tried to make domestic robots resemble humans, developing bipedal humanoids, such as Ubtech Robotics' Walker, who offers a butler service at home (Figure 4). The next step in this direction are the anthropomorphic robots, which try to emulate human movement during the walk. However, these types of robots can have complex mechanics and be difficult to control.

Figure 4. Ubtech Robotics' Walker, a bipedal robot

An alternative architecture has been studied to work at home: the self-balancing or mobile inverted pendulum, with to coaxial wheels that maintain the structure in equilibrium. This solution was implemented and commercialized by Segway. Other examples are QA, a telepresence robot created by Anybot (Figure 5).

Figure 5. Anybot QA, balance robot on two wheels

Numerous robots have been built with academic purposes. Classical control methods have been selected to control this type of vehicles, such as PID controllers [4] [5].This method usually worked quite fine with small robots or with low mass that allows an easy design and tuning for the controller.

Another approach to control these systems was using the optimal control theory, a LQR controller [6] [7]. With this method, a cost function is minimized, while the controller receives all the states of the system.

Additionally, more original techniques have been proposed, such as fuzzy logic PID controllers [8]

CHAPTER 3: THEORETICAL STUDY OF THE PROBLEM

In this chapter, the real robot in which the model is based and from where the values of the parameters are obtained is explained and detailed.

Besides, the mathematical equations that describe the behaviour of the robot are obtained, with the purpose of obtaining a model that can be used in a simulation environment to compute a controller for the system.

Lastly, the parameters values used, and the methods applied to estimate and calculate them are stated.

3.1. Physical description of the robot

All the simulations and results obtained have been realized with the objective of controlling a real robot. This robot was, originally, a mobile inverted pendulum constructed by Martino Migliavacca [9].

As it is shown in Figure 6, it consists of two wheels, a wooden board that functions as base for the controller and motors and a metallic frame made of aluminium.

Figure 6. Previous hardware of the robot

Later, this hardware was physically modified. The idea was to create a mobile robot, named *Basketbot*, that emulated a basketball hoop, with the intention of programming different games that could be played with it. For this reason, a Kinect was thought to be implemented in order to have a large range of games that could be developed, with the interaction of the user.

To do so, the metallic frame was modified and replace with a single metallic bar, where a plastic board was connected, emulating a basketball one. The hoop was also made of aluminium and was welded to the frame.

The final result is it shown in Figure 7 and Figure 8, where the frontal and lateral view of the robot can be seen.

It is important to remark that the robot is equipped with to sensors: an accelerometer and a gyroscope.

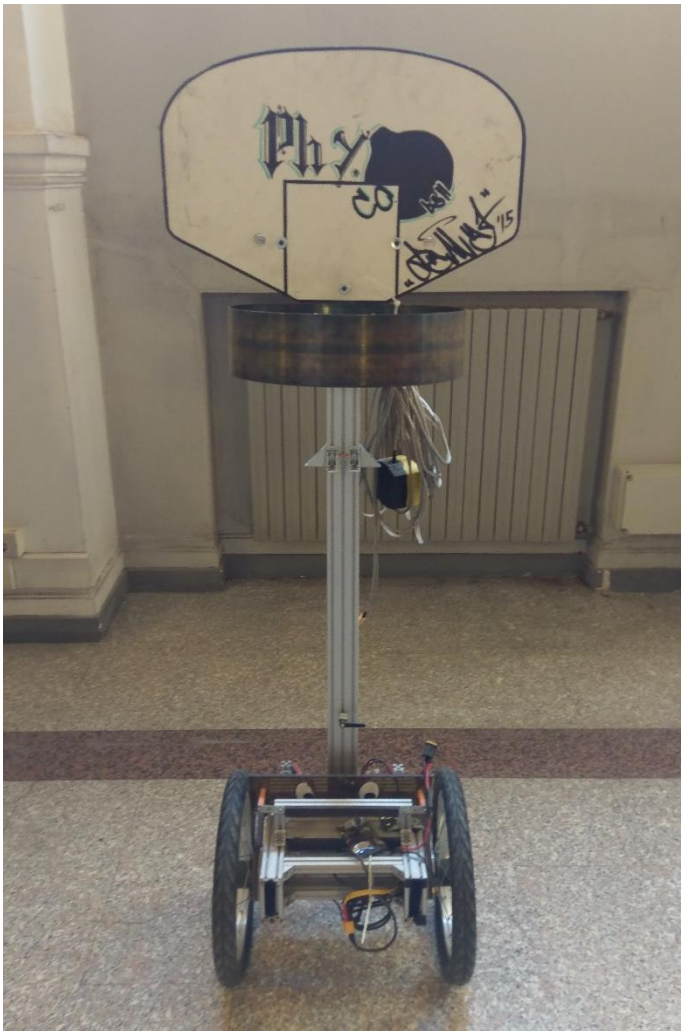

Figure 7. Frontal view of the robot

Figure 8. Lateral view of the robot

Additionally, in Figure 9 it can be seen the wheels and the platform where the controller and the motors were placed. The idea was that the Kinect was also stablished here.

Figure 9. Wheels and base of the robot

3.2. Mathematical model of the system

As it was stated, the robot consists in an inverted pendulum on two wheels. Therefore, the system will be modelled by three subsystems: the motors, the wheels and the inverted pendulum.

In the Table 1 all the variables that appear in the equations are described.

$\mathbf{M_w}$	*kg*	Mass of the wheels
$\mathbf{M_P}$	*kg*	Mass of the body (pendulum)
$\mathbf{I_w}$	$kg*m^2$	Inertia of the wheel
$\mathbf{I_P}$	$kg*m^2$	Inertia of the body
r	*m*	Radius of the wheel
l	*m*	Length to the body centre of mass
g	m/s^2	Acceleration of gravity
θ	*rad*	Tilt angle: Angle between the body and the vertical line
φ	*rad*	Angle between the vertical line and the wheel
$\mathbf{K_t}$	*Nm/A*	Motor torque constant
R	*Ω*	Terminal resistance
$\mathbf{K_e}$	*Vs/rad*	Motor back EMF constant
$\mathbf{T_M}$	*Nm/A*	Motor torque
$\mathbf{V_e}$	*V*	Back electromotive force
$\mathbf{V_e}$	*V*	Applied voltage
i	*A*	Current through the motor coil
L	*H*	Motor inductance
$\mathbf{F_H}$	*N*	Horizontal contact force between the pendulum and the wheel
$\mathbf{F_V}$	*N*	Vertical contact force between the pendulum and the wheel
$\mathbf{F_F}$	*N*	Friction force on the wheel
$\mathbf{F_N}$	*N*	Normal force on the wheel

Table 1. Description of the variables

Firstly, the motors will be modelled. It is assumed that the motors are ideal. Therefore, the torque produced by each motor is:

$$T_M = K_M * i \quad (1)$$

The back electromotive force is computed as:

$$V_e = K_e * \dot{\varphi} \quad (2)$$

The voltages that appear in the circuit are due to back EMF, resistance and inductance of the coil. According to the Kirchhoff's second law, the voltages around any closed loop in a circuit is zero:

$$V_a - R * i - L\frac{di}{dt} - V_e = 0 \quad (3)$$

To simplify the equation, the voltage due to the inductance is neglected. Therefore, equation can be computed as:

$$i = \frac{V_a}{R} - \frac{K_e * \dot{\varphi}}{R} \quad (4)$$

Applying Newton's second law, the relation between the motor torque and the angular acceleration of the wheels can be obtained. It is assumed that the torque due to the friction of the shaft is neglected.

$$I_W * \ddot{\varphi} = T_M - T_a \quad (5)$$

Substituting equation (1) and (4) into (5) and reorganizing the terms, the equation of the torque can be computed:

$$T_a = \frac{K_M}{R} V_a - \frac{K_e * K_M}{R} \dot{\varphi} \quad (6)$$

Since two DC motors are used, so the final equation modelling the motor is:

$$T_a = \frac{2K_M}{R} V_a - \frac{2K_e * K_M}{R} \dot{\varphi} \quad (7)$$

Secondly, the model of the inverted pendulum is studied. Figure 10 shows all the forces, torques and variables that appear on a pendulum. Rotational friction and wind resistance forces have been neglected.

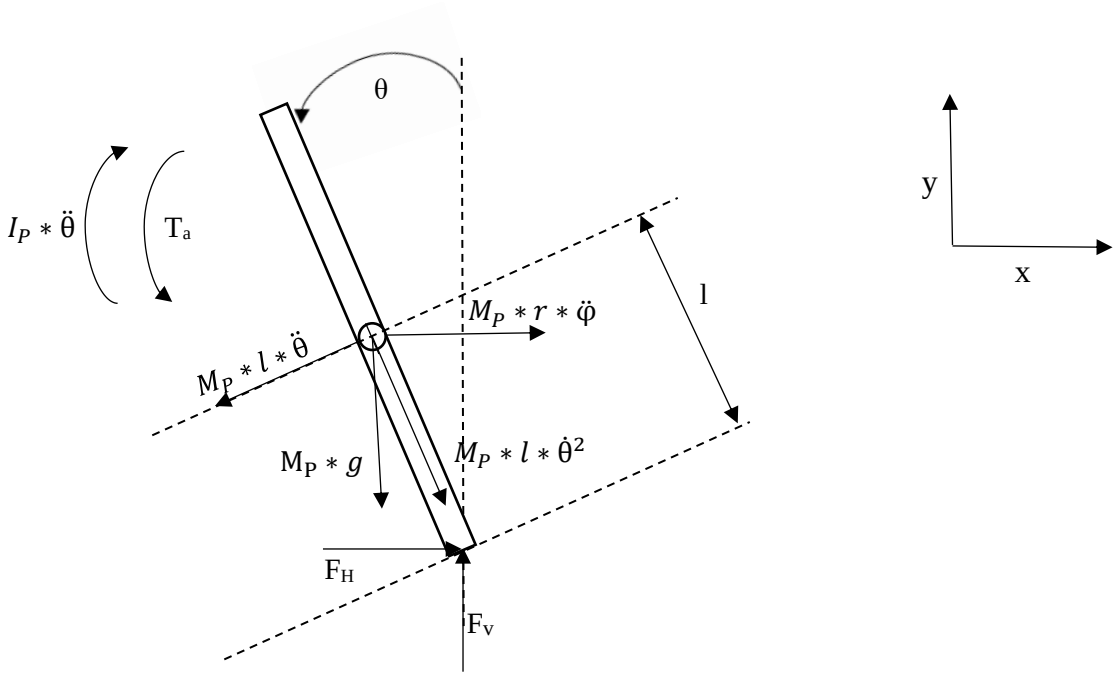

Figure 10. Variables, forces and torques on the pendulum

In the x direction, the forces acting on the pendulum are:

$$M_P * \ddot{x} = F_H - M_P * l * \dot{\theta}^2 - M_P * l * \ddot{\theta} * cos\theta \quad (8)$$

The equation regarding the acting forces in the plane perpendicular to the pendulum is:

$$M_P * \ddot{x} * cos\theta = F_V * sin\theta + F_H * cos\theta - M_P * g * sin\theta - M_P * l * \ddot{\theta} \quad (9)$$

Regarding the momentums, the sum of each around the centre of gravity is:

$$I_P * \ddot{\theta} = -T_M - F_V * sin\theta * l - F_H * cos\theta * l \quad (10)$$

The relationship between the linear acceleration of the pendulum and the angular acceleration of the wheels can be computed as:

$$\ddot{x} = r * \ddot{\varphi} \quad (11)$$

Combining the equations (9) to (11) it is possible to obtain the equation of the pendulum angular acceleration:

$$\ddot{\theta} = \frac{M_P * g * l * R * sin\theta - M_P * l * R * r * cos\theta * \ddot{\varphi} + 2K_M * K_e * \dot{\varphi} - 2K_M * V_a}{R * (M_P * l^2 + I_P)} \quad (12)$$

Then, the variables and forces acting on the wheels are studied. They are shown in the Figure 11. It is assumed that the wheels will always be in contact with the ground and that there is no slip. Besides, cornering forces will be neglected.

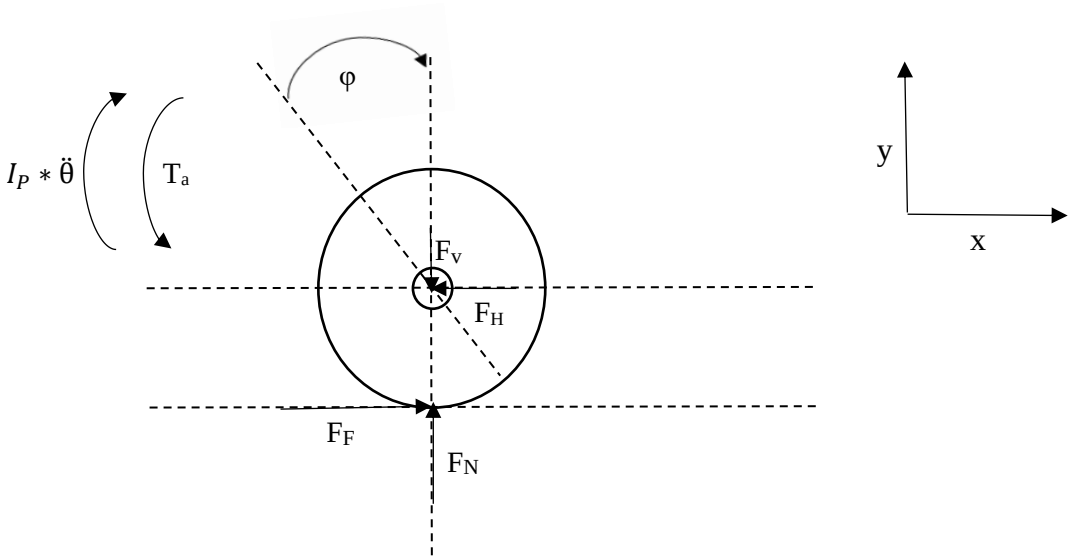

Figure 11. Variables, forces and torques on the wheel

The equilibrium of moments respect from the centre of the wheel is:

$$I_W * \ddot{\varphi} = T_M - F_F * r \quad (13)$$

Computing the sum of forces acting in the x direction:

$$M_W * \ddot{x} = F_F - F_H \quad (14)$$

The expression of the wheels angular acceleration can be obtained from equations (8), (11), (13) and (14):

$$\ddot{\varphi} = \frac{M_P*r*l*R*\theta^2*sin\theta - M_P*l*R*r*cos\theta*\ddot{\theta} + 2K_M*K_e*\dot{\varphi} - 2K_M*V_a}{R*(M_P*r^2 + I_P + M_W*r^2)} \quad (15)$$

By the combination of the equations (11) and (15), two new expressions of $\ddot{\varphi}$ and $\ddot{\theta}$ can be computed:

$$\ddot{\theta} = \frac{b*c*g*sin\theta + c*r*cos\theta*(2K_M*(K_e*\varphi - V_a) - c*r*sin\theta*\theta^2) + 2b*K_M*(K_e*\varphi - V_a)}{a*b - c^2*r*cos\theta} \quad (16)$$

$$\ddot{\varphi} = \frac{c*r*sin\theta*(a*\theta^2 - c*g*cos\theta) + 2c*r*cos\theta*K_M*(V_a - K_e*\varphi) + 2a*K_M*(V_a - K_e*\varphi)}{a*b - c^2*r*cos\theta} \quad (17)$$

Where:

$$a = R * (M_P * l^2 + I_P) \quad (18)$$

$$b = R * (M_P * r^2 + I_P + M_w * r^2) \quad (19)$$

$$c = M_P * l * R \quad (20)$$

The expressions (16) and (17) are the fundamental equations of the model of the system. In order to be able to design a proper controller, they must be linearized with respect to

the equilibrium point. In this case, the equilibrium point is $\theta, \varphi, \dot{\theta}, \dot{\varphi} = 0$. Therefore, the linearized system can be computed as a state space:

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{\varphi} \\ \ddot{\varphi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{b*c*g-c^2*r^2}{a*b-c^2*r^2} & 0 & 0 & \frac{2K_M*K_e*(c*r+b)}{a*b-c^2*r^2} \\ 0 & 0 & 0 & 1 \\ \frac{a*c*r-c^2*r*g}{a*b-c^2*r^2} & 0 & 0 & \frac{2K_M*K_e*(-c*r-a)}{a*b-c^2*r^2} \end{bmatrix} \begin{bmatrix} \theta \\ \varphi \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{2K_M*K_e*(-c*r-b)}{a*b-c^2*r^2} \\ 0 \\ \frac{2K_M*K_e*(c*r+a)}{a*b-c^2*r^2} \end{bmatrix} V_a \qquad (21)$$

3.3. Parameters measurement and estimation

For simulating and designing the controller, it is necessary to have the values for the parameters that are used in the model. For this instance, the values of the parameters of the *Basketbot* were measured or estimated.

The model required a series of physical parameters of the pendulum and the wheels, such as the masses, the moments of inertia and the distances to the centre of mass.

The masses where measured with a digital scale. The wheels mass measurement realized in [10] was used, in order not to separate the body of the robot from the wheels, so the whole robot was weighted, and then the wheel mass was subtracted to obtain the pendulum mass.

The centre of mass of the wheels is assumed to be in the centre of the wheel circumference. To calculate the centre of mass of the pendulum, it has been divided into subsystems with simple geometrical forms. The reason of this division is because the centre of mass of a homogeneous object with constant thickness is the barycentre of the object. The centre of mass of the base, metallic frame, hoop and basket board will be computed separately.

Since only the vertical distance to the centre of gravity is needed in the model, only y_G will be estimated. Thus, the centre of mass of the pendulum will be computed as:

$$y_G = \sum_{k=1}^{n} \frac{m_k * y_{gk}}{M} \qquad (22)$$

To compute the moment of inertia of the wheels, it is assumed that most of its mass is concentrated in the rim of the circumference. Therefore, it can be modelled as a hoop, which moment of inertia is:

$$I = M_w * r^2 \qquad (23)$$

Regarding the moment of inertia of the pendulum, the division into subsystems previously stated will be used. The moment of inertia of a rectangular body with constant thickness, with respect to its centre of mass (or barycentre) can be estimated as:

$$J_G = \frac{M}{12} * (b^2 + l^2) \quad (24)$$

Where b is the width of the rectangle and l is the height.

Then, according to the parallel axis theorem, it is possible to determine the moment of inertia of a rigid body about any axis, knowing the moment of inertia about the axis of the object's centre of gravity:

$$J_A = J_G + M * \overline{OA}^2 \quad (25)$$

Therefore, the moment of inertia of the pendulum can be computed using the moments of inertia of each body. The results can be seen in the Table 2.

Parameter	Value
M_w	3.4 kg (both wheels)
M_P	21.6 kg
I_w	0.068 kg*m^2
I_P	10 kg*m^2
r	0.2 m
l	0.4 m

Table 2. Value of the parameters of the pendulum and the wheels

Finally, the parameters of the motors are needed. The robot is equipped with two electric motors, produced by Maxon Motor, of 150W.

In the Figure 12. Datasheet of the Maxon motors used the datasheet of the motors is shown. From this information, the value of the parameters needed for the model are seen in Table 3.

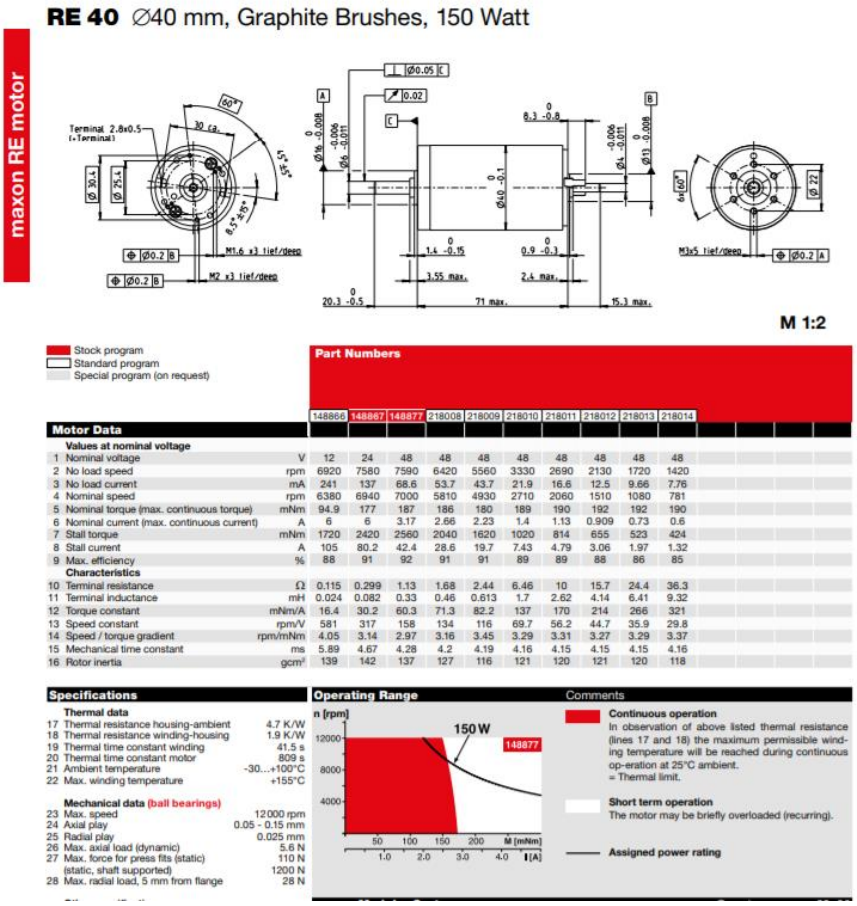

Figure 12. Datasheet of the Maxon motors used [11]

Parameter	Value
K_t	30.2 mNm/A
R	0.316 Ω
K_e	3.15 mVs/rad

Table 3. Values of the parameters of the motor

CHAPTER 4: CONTROL

There are different techniques to control these types of systems. In this chapter, two classic controllers will be studied. Firstly, the PID (proportional–integral–derivative controller) controllers will be described. Later, the LQR (Linear-quadratic regulator) and optimal control analysed. Finally, the pole placement control design will be examined.

4.1. PID controllers

There PID controller is very useful and can solve a wide range of control problems. More than 95% of all industrial control problems are solved by PID control [12].

In this type of control, the objective is to minimize the error, e, which is the difference between the reference, r, and the output signal of the system, *y*. The control signal, *u*, is computed by the formula:

$$u = K_P * e(t) + K_i * \int_0^t e(\tau)d\tau + K_D \frac{d}{dt} e(t) \quad (26)$$

The parameters K_p, K_i and K_d are, respectively, the coefficients of the proportional, integral and derivative action. These are the tuning parameters that must be designed in order to control the system.

Expression (26) can also be defined in the Laplace domain:

$$L(s) = K_P + \frac{K_i}{S} + K_D * s \quad (27)$$

Where *s* is the complex frequency.

However, the most common way in which the PID controllers are defined is in what is called the standard form:

$$u = K_P * (e(t) + \frac{1}{T_i} * \int_0^t e(\tau)d\tau + T_D \frac{d}{dt} e(t) \quad (28)$$

In this case, T_i is the integral time and T_d is the derivative time. These parameters are related to the previous gain coefficients by:

$$K_i = \frac{K_P}{T_i} \qquad K_d = K_P * T_D \quad (29)$$

4.1.1. PID tuning

In order to use a PID controller, the parameters K_p, K_i and K_d have to be properly designed. Each of them has its importance and characteristics.

The increment of the proportional action reduces the asymptotic error. However, high values of K_p can create instabilities, because it typically generates oscillations in the transient time. This behaviour can be seen in the Figure 13, where the process transfer function is $G(s)\frac{1}{(s+1)^3}$ and the reference is 1. Different values of the controller gain are shown.

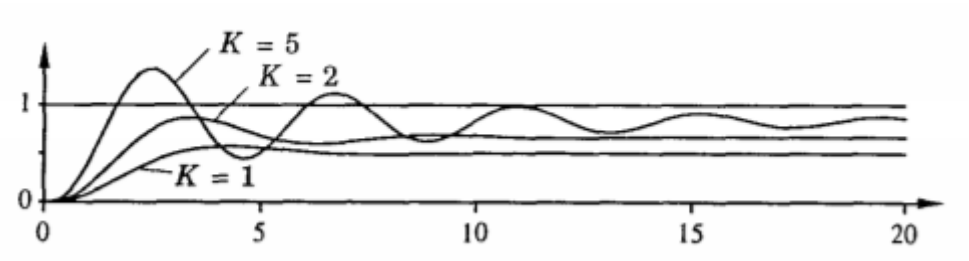

Figure 13. Simulation of a closed-loop system with proportional control [13]

The proportional control usually generates a control error in the steady state. However, with the integral action the steady state error will always be zero. Higher values of Ki will produce faster responses of the system, but can create oscillations, as it is shown in the Figure 14.

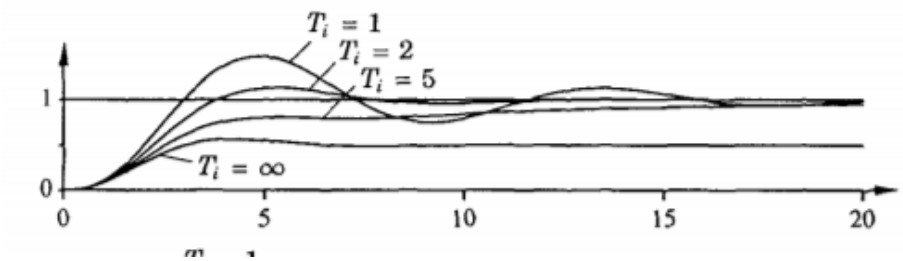

Figure 14. Simulation of a closed-loop system with integral control

The derivative term is proportional to the derivative of the control error and allow a prediction of the future error. The effects of the derivative control can be seen in Figure 15.

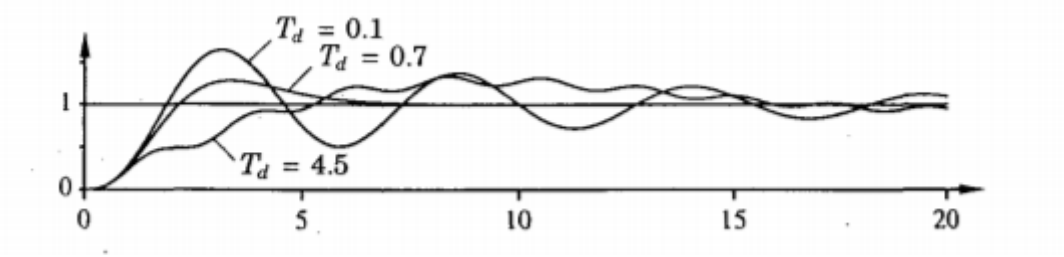

Figure 15. Simulation of a closed-loop system with derivative control

The most used method to tune the PID parameters is the Ziegler-Nichols method. This technique consists of these simple steps:

- Connect a proportional controller to the system and study the closed-loop response
- Start with a low value of K_p
- Increase the proportional gain until the response gets to a steady-state oscillation
- Save this gain as the value of the critical gain, K_{cr}
- Measure the oscillation period, P_{cr} [14]
- Compute the PID parameters following the Table 4:

Type of controller	K_P	T_i	T_D
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$0.8P_{cr}$	0
PID	$0.6\ K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Table 4. Ziegler-Nichols method

4.2. LQR control

The LQR controller is part of the control algorithm in the field of optimal control. The objective of this control method is to find a control law such that a certain optimality criterion is achieved.

If a continuous linear system is given, such as:

$$\dot{x} = A * x + B * u \quad (30)$$

With a quadratic cost function defined as:

$$J = \frac{1}{2}\int_0^T (x^T * Q * x + u^T * R * u) * dt + \frac{1}{2}x^T(T) * P_1 * x(T) \quad (31)$$

Where $Q \geq 0$, $R > 0$, $P_1 \geq 0$ are symmetric, positive (or semi-positive) definite weighting matrices. Matrix Q weights the states, meanwhile matrix R is a weight of the control input.

The control law that minimizes the cost function is a feedback control:

$$u(t) = -K * x(t) \quad (32)$$

Where the gain K is given by:

$$K = R^{-1} * B^T * P \quad (33)$$

And P is the unique positive definitive solution of the Ricatti ODE (ordinary differential equation):

$$A^T * P + P * A - P * B * R^{-1} * B^T * P + Q = 0 \quad (34)$$

The LQR algorithm is, at its core, just an automated way of finding an appropriate state [15].

CHAPTER 5: IMPLEMENTATION AND SIMULATION RESULTS

In this chapter the different proposed control techniques are implemented in Matlab/Simulink. Three different control methods are explained: Pole placement, PID and LQR control.

The results of the simulation of each controller will be discussed in detail and the differences and advantages between them are stated.

5.1. Model implementation and validation

In the equation 21, the mathematical description of the system represents the robot dynamics with four states. The third state, $\dot{\varphi}$, in the state space representation is not used to describe any of the other states. For simplicity and faster calculations this state can therefore be ignored.

A first verification to the model is done by applying a constant of 1V as an input of the open-loop system. Because the system is inherently unstable, the tilt angle should rise unboundedly. This results in the robot falling over. This behaviour can be seen in the Figure 16.

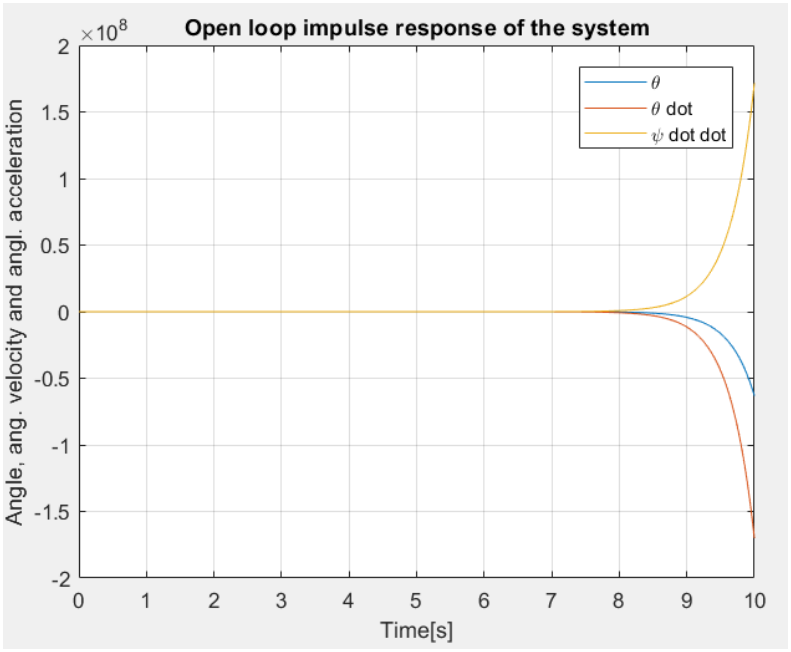

Figure 16. Step response of the open-loop system

The same result must occur if the system is excited by an impulse response, emulating a nudge, as it is shown in the Figure 17.

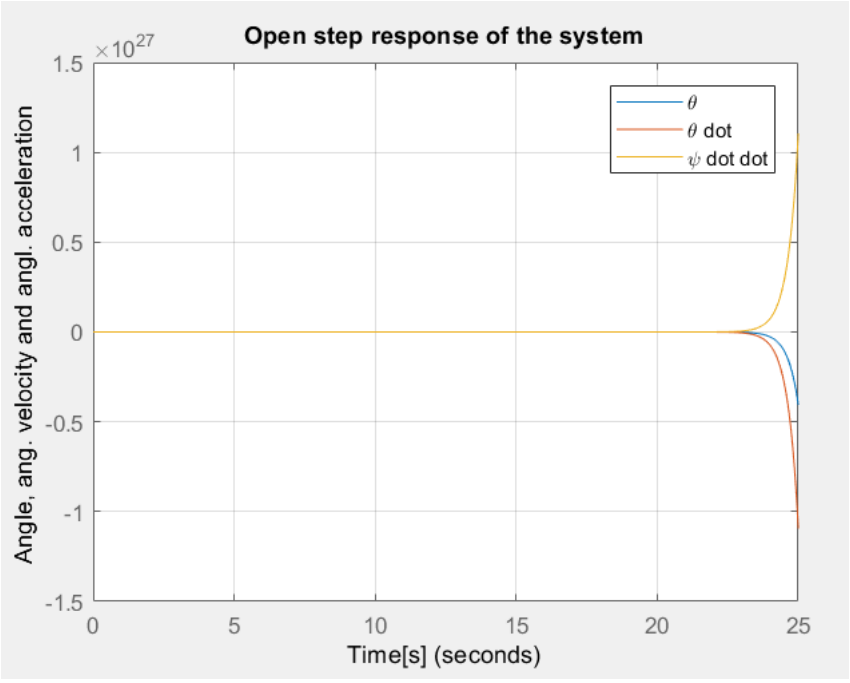

Figure 17. Impulse response of the open-loop system

To analyse the instability of the system, the poles of the system can be computed and drawn in the plane, shown in the Figure 18. The poles of the system are located at -2.69,

-0.0004 and 2.69. Having negative poles result in instabilities. Ideally, all poles should be located in the left-hand plane.

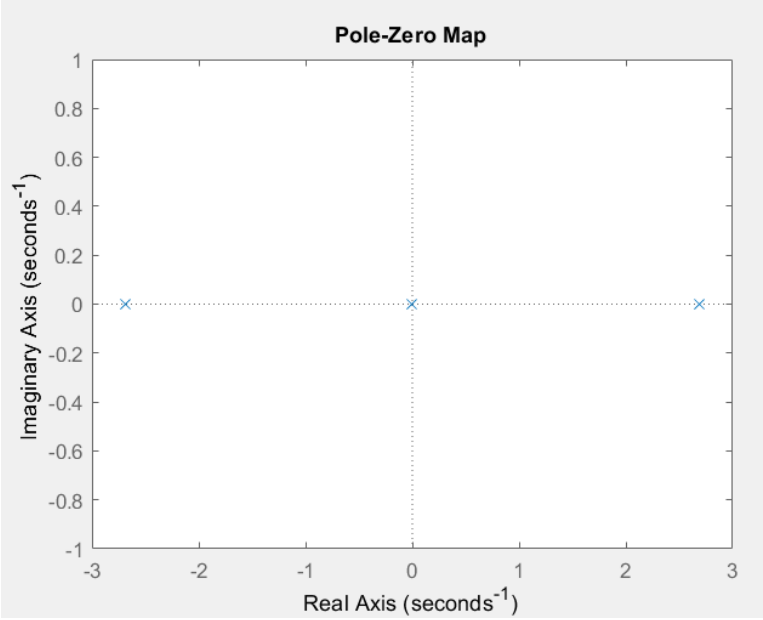

Figure 18. Pole/zero map of the system

5.2. Pole placement

In order to implement the pole placement technique, the system must be controllable. To do so, the controllability matrix must be computed:

$$C = [B \quad AB \quad A^2B] = \begin{bmatrix} 0 & -0.363 & 0 \\ -0.363 & 0 & -0.263 \\ 0.172 & 0 & 0.265 \end{bmatrix} \tag{35}$$

The rank of the controllability matrix is 3, the same as the size of the matrix, so the system is controllable.

With the Matlab command *place*, it can be easily assigned new poles to the closed-loop transfer function. It is recommended not choose the closed-loop poles very negative, because the system will be fast reacting; and not too far away from the open loop poles, because it will result in a high effort control [16].

Therefore, the new poles of the closed loop system were selected at -5.38, -2.69-1.3i, -2.69+1.3i, as it is shown in Figure 19.

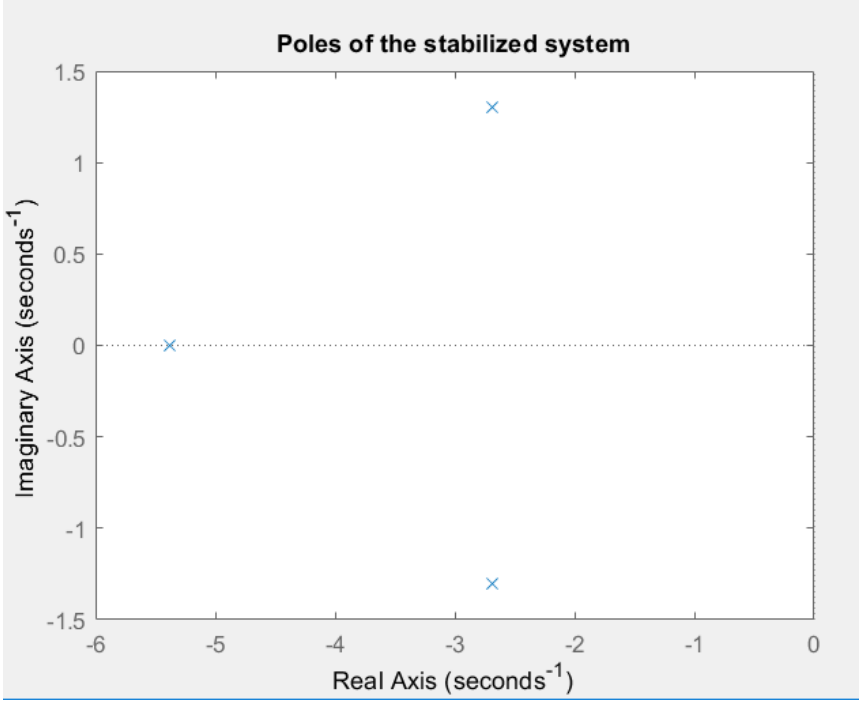

Figure 19. Poles of the stabilized system with pole placement control

To check if the system is stable, a perturbation is implemented at 1 second. It can be seen in Figure 20 that the responses of the angle and angular velocity of the pendulum; and the wheel angular acceleration tends to zero, assuring that the robot remains vertical.

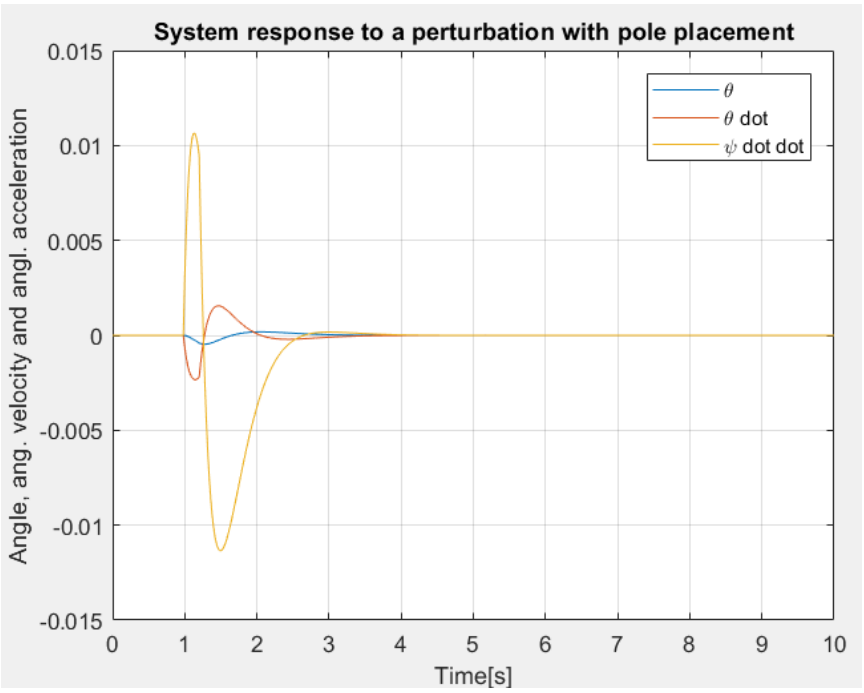

Figure 20. System response to a perturbation with pole placement control

5.2. PID control

As it was stated before, since there are 3 inputs in the system, but only 1 output, with a PID controller only one variable can be controlled. The pendulum angle, θ, was selected to be the control variable. Therefore, the design of the controllers will have the objective to minimize the error in the tilt angle.

5.2.1. Ziegler-Nichols method

In order to tune the PID by the Ziegler-Nichols method, the response of the system with a proportional controller must reach a steady state oscillation, as it is shown in Figure 21. Then, the period of the oscillation is measured. Thus, the critical gain and the critical period are:

$$K_{cr} = 920.75 \qquad P_{cr} = 2.973 \tag{36}$$

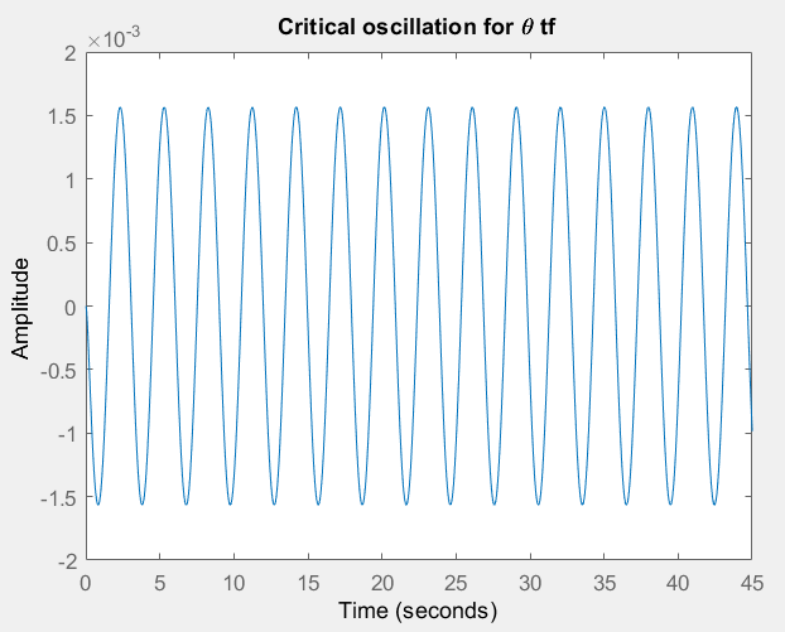

Figure 21. Critical oscillation of the closed loop system with a proportional controller

According to the Ziegler-Nichols table, the tuned PID parameters are:

$$K_P = 552.45 \qquad T_i = 1.49 \qquad T_d = 0.37 \tag{37}$$

Applying a disturbance at 1 second, the response of the system with this controller can be checked. In the Figure 22, all 3 signals tend to zero after oscillation during a short period of time because of the perturbation. Thus, the robot will remain vertical and will not fall.

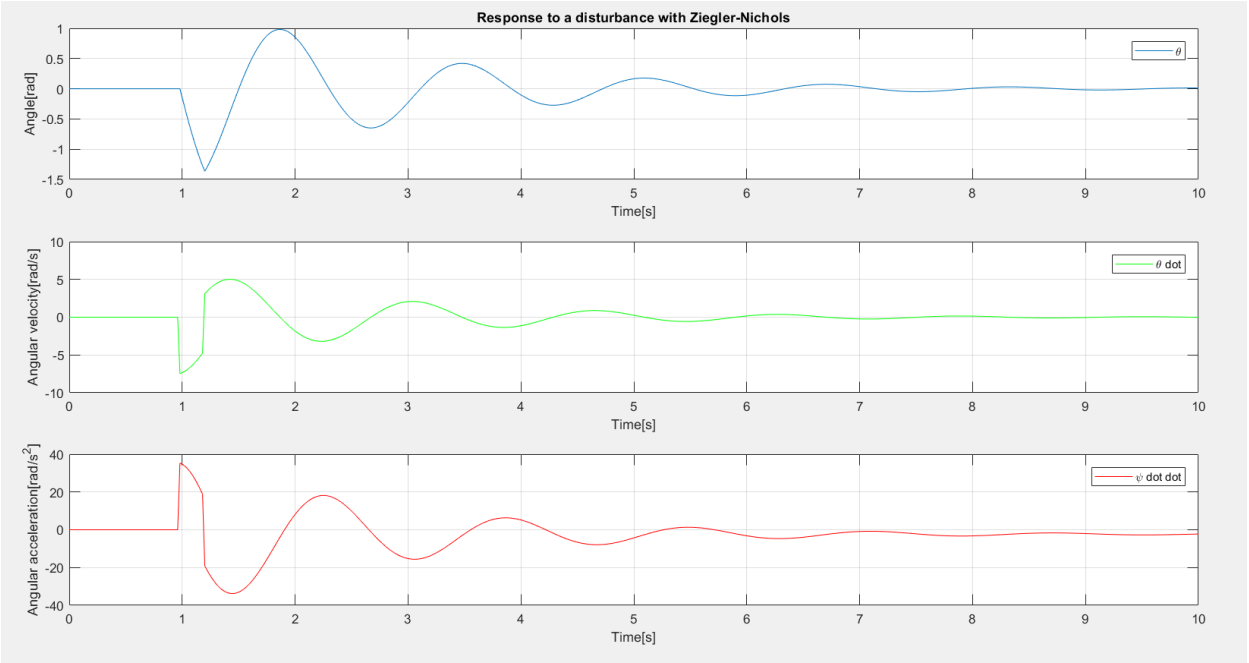

Figure 22. System response to a disturbance with PID control (Ziegler-Nichols)

If a unitary step is applied, the tilt angle is maintained stable, but does not reach the reference, as it is shown in Figure 23. This means that the controller does not guarantee zero error, and it can be improved.

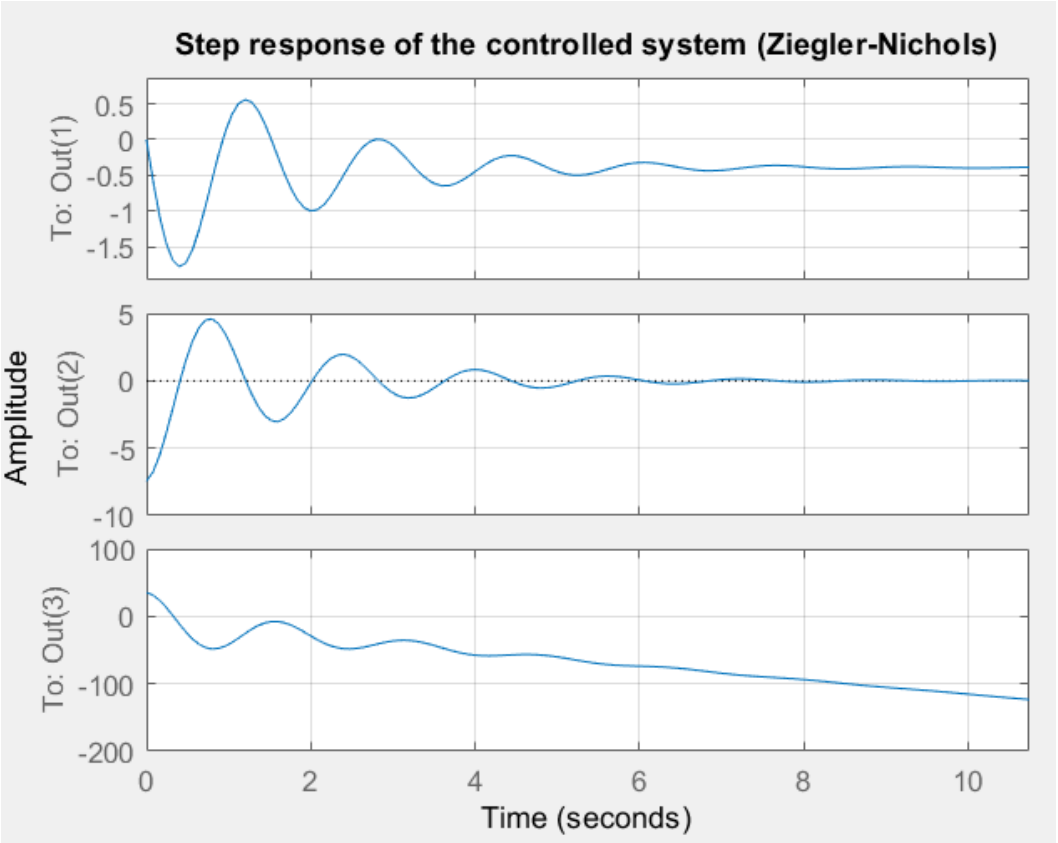

Figure 23. Step response of the system controlled by a PID (Ziegler-Nichols)

5.2.2. Manual tuning

The Ziegler-Nichols is a good method to design an initial controller but can be optimised. Therefore, another PID controller was tuned using the Matlab *rtool*, which allows to manually choose the parameters of the PID in order to have a successful response. The PID selected was:

$$K_P = -1.77 * 10^4 \qquad K_i = -9.52 * 10^4 \qquad K_d = -770 \qquad (38)$$

Realizing the same test as the previous controller, it can be seen in Figure 24 that, after a nudge, the robot recovers its vertical position, in a much faster way than the other PID controller.

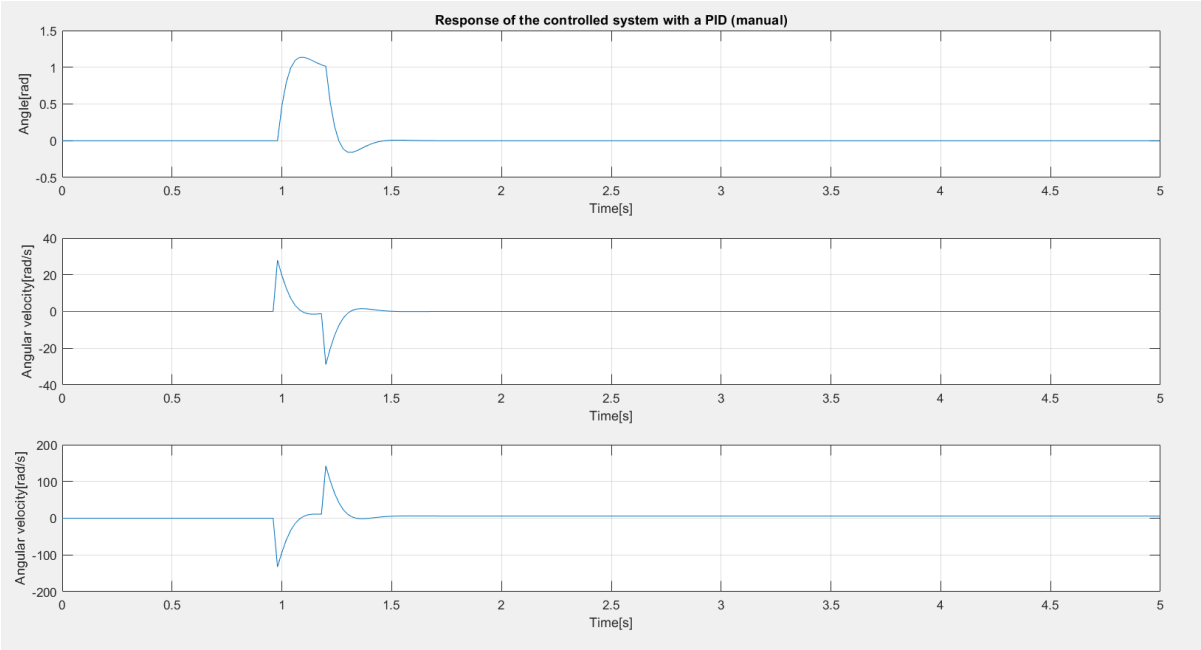

Figure 24. System response to a disturbance with PID control (manual)

Regarding the step response it is shown in Figure 25 that the tilt angle reaches the desired reference in a short period of time. However, the angular acceleration of the wheel continuously increases. This behaviour is most likely caused by the integral action of the controller, because it has accumulated the positive angular error and is still causing an output signal even if the angle error is very small.

As a conclusion, the second PID designed reacts in a much faster and less oscillatory manner than the first controller, and reaches the reference given, contrary to the designed with the Ziegler-Nichols method. Therefore, it provides a better result.

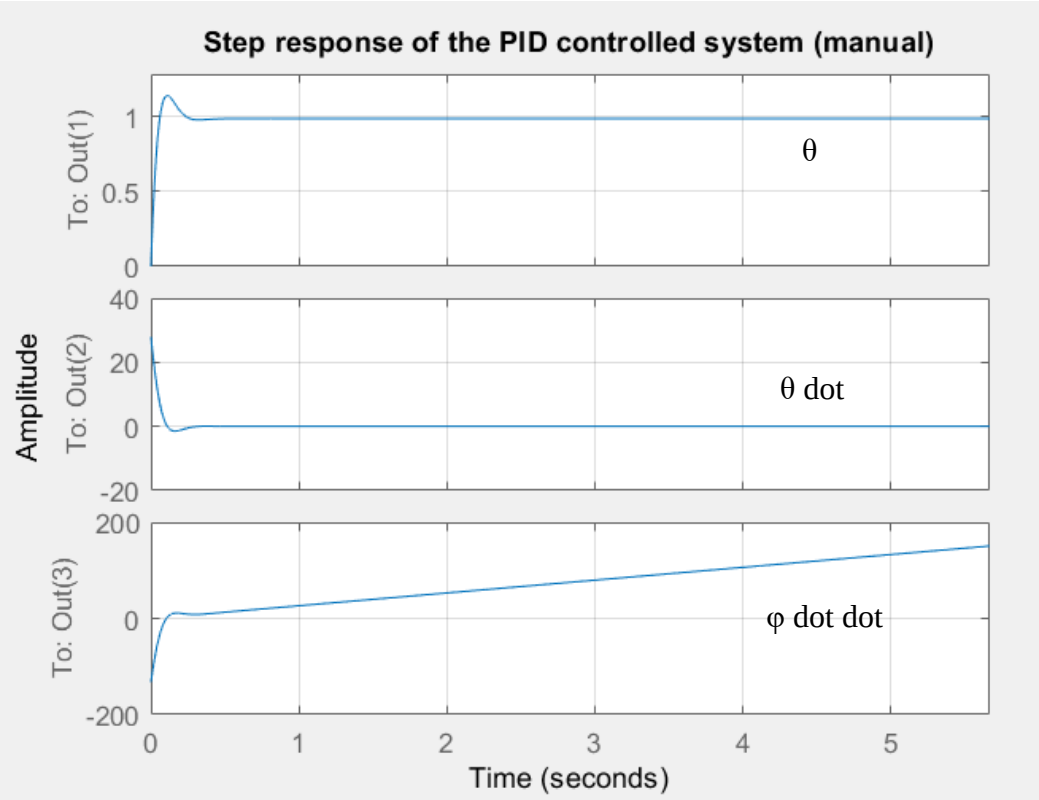

Figure 25. Step response of the system controlled by a PID (manual)

5.3. LQR control

Contrary to PID control, LQR control will allow to act on the response of the 3 signals. The first step is to check if the system is controllable, which has previously done in the pole placement section.

Next, the weight matrices Q and R have to be estimated. The simplest case is:

$$Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad R = 1 \qquad (39)$$

The command *lqr* will provide with the appropriate feedback gain K:

$$K = \begin{bmatrix} -419.01 \\ -156.78 \\ -1 \end{bmatrix} \qquad (40)$$

It can be checked that all the poles of the closed loop system are in the negative plane, as it is shown in Figure 26:

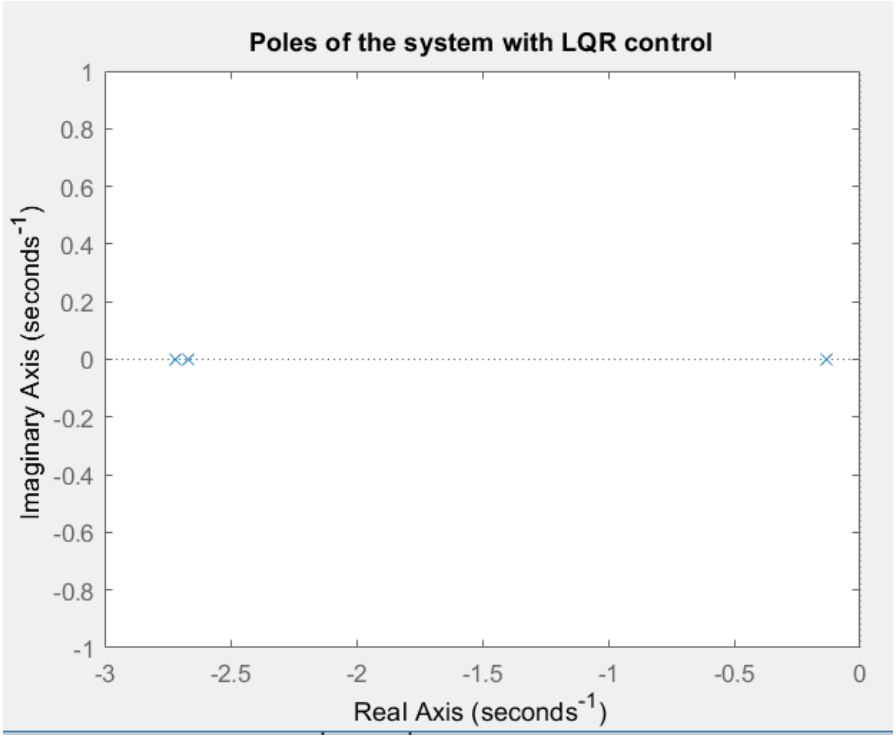

Figure 26. Poles of the closed loop system with LQR control

Moreover, the response to a perturbation can be studied, as seen in.the Figure 27. The 3 control variables go to zero after certain time, meaning that the closed loop system is stable, and the robot conserves the verticality.

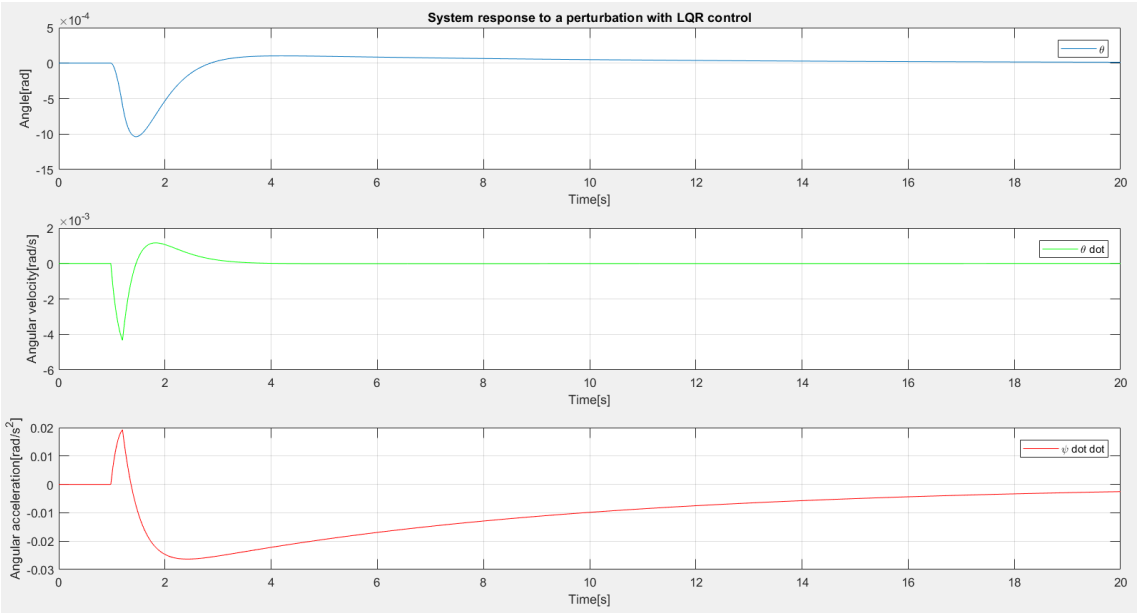

Figure 27. System response to a disturbance with LQR control

The estimation of the weight matrices can be improved. Considering that the most critical control variable is the pendulum angle, a higher value of the term (1,1) can be computed to reflect its importance:

$$Q = \begin{bmatrix} 1000 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 100 \end{bmatrix} \qquad R = 1 \qquad (41)$$

The command *lqr* will provide with the appropriate feedback gain K:

$$K = \begin{bmatrix} -621.92 \\ -238.49 \\ -10 \end{bmatrix} \qquad (42)$$

Again, the poles of the closed loop system are in the negative plane, as it is shown in Figure 28:

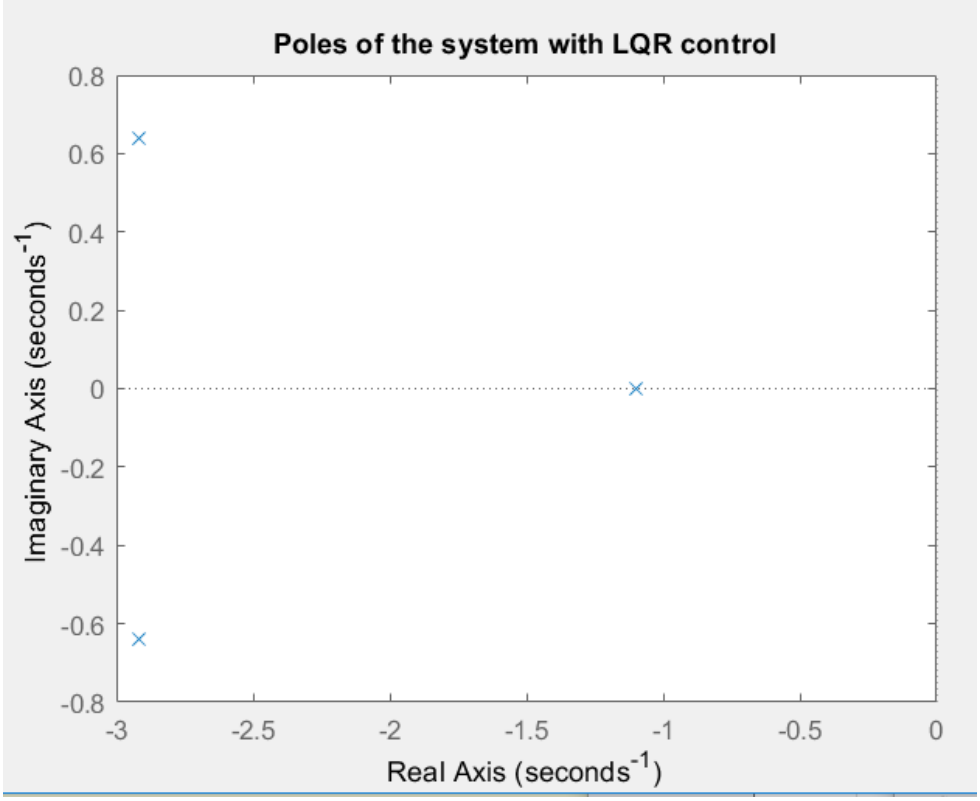

Figure 28. Poles of the closed loop system with LQR control (second design)

Analysing the system response to a disturbance, as it is shown in Figure 29, it is faster and with a smaller oscillation than in the previous case.

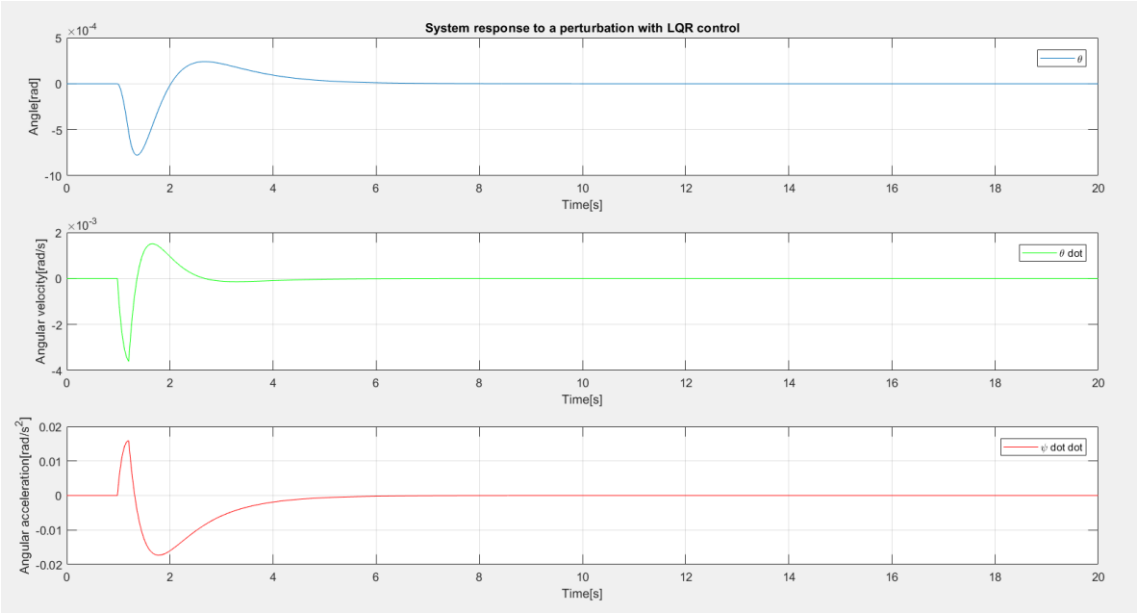

Figure 29. System response to a disturbance with LQR control (second design)

Besides, if a step input is given, it can be checked in Figure 30, that the system does not follow it, and that the wheel angular acceleration remains with a constant negative value.

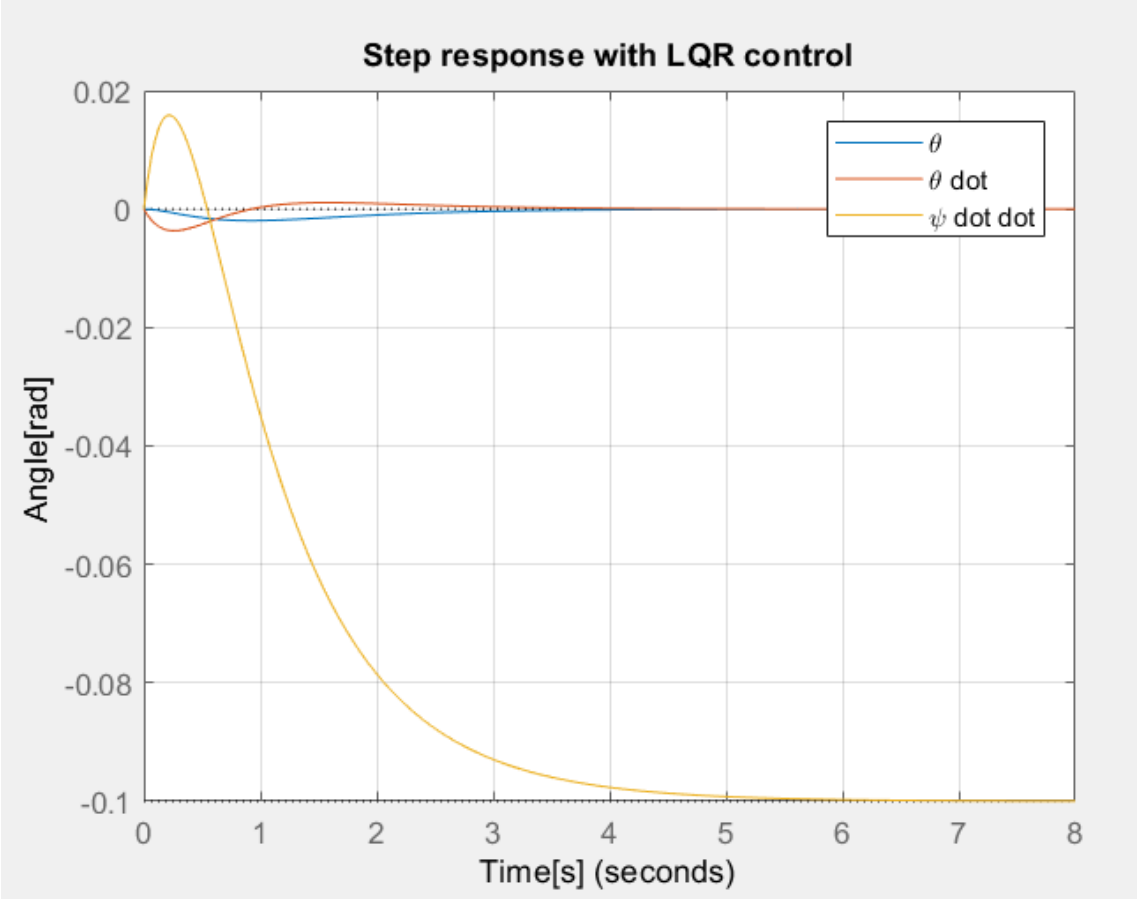

Figure 30. Step response of the system with LQR control

5.4. Results discussion

Analysing the results obtained, it can be stated that, while the PID controllers allow the system to follow a reference in the pendulum angle and are quite robust to perturbances, they have an important drawback: it is not possible to control the wheel angular acceleration, so the vehicle will not be able to stop, without implementing some kind of solution.

On the contrary, pole placement control and LQR control are really robust to unexpected disturbances and can control each variable, but the controlled system is not able to follow the desired reference.

The best solution could be a compromise between the two approaches, having a PID controller for the pendulum angle, while implementing a LQR/pole placement control for the wheel angular acceleration.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

This thesis focused on the analysis and modelling of a self-balancing robot, with the objective of designing and studying different control alternatives to be able to control the system.

The mathematical model described, even though that it is a simplification, reflects and emulates the real behaviour of the system.

The proposed methods achieve a successful control of the system, maintaining the robot stable in its vertical position. Additionally, the proposed PID controller allows the system to follow a reference in the pendulum angle.

However, there is room for improvement in the controller. The PID follow with accuracy the reference and reacts to the uncertainties, but it is not able to control the wheel angular acceleration. On the other hand, the LQR control can control each of the three control variables but cannot reach the desired reference.

In order to improve the results obtained, compromise between the two control methods should be implemented, having a PID controller for the pendulum angle, while implementing a LQR/pole placement control for the wheel angular acceleration.

Besides, a Kalman filter could be developed to estimate the sensors measurement, because in reality the signal provided by the sensor contains noise and it is not immediate.

It is important to remark that this work has been realized in a simulation environment, due to the unavailability of physical controllers to be placed on the robot. Therefore, the next step should be to implement and validate the results achieved in the real system.

Besides, once the control is implemented in the robot, several games and programmes can be developed for it, taking advantage of the basketball hoop that is welded and the Kinect device that could be placed in the base.

BIBLIOGRAPHY

[1] J. N. Nilsson,, "Shakey the robot," 1987.

[2] A. M. Goodrich and C. A. Schultz, "Human-Robot Interaction: A Survey," 2007.

[3] T. Knell, "Domestic robots: a case study on security in ubiquitous computing," 2014.

[4] S. Balasubramanian and M. N. Lathiff, "Self balancing robot," 2011.

[5] G. M. T. Nguyen and N. Duong, "A PID backstepping controller for two-wheeled self-balancing robot," 2010.

[6] H. Hellman and H. Sunnerman, "Two-Wheeled Self-Balancing Robot - Design and control based on the concept of an inverted pendulum," 2015.

[7] Y. Ding, J. Gafford and M. Kunio, "Modeling, Simulation and Fabrication of a balancing robot," 2012.

[8] S. Paliwal, V. Chopra and S. Singla, "Stabilization of mobile inverted pendulum using fuzzy PID controllers".

[9] M. Migliavacca,, "Progettazione e realizzazione di una base robotica bilanciante su ruote," 2009.

[10] A. Galbiati, "Progettazione e realizzazione del controllodi una base robotica bilanciante su ruote".

[11] Maxon motor, [Online]. Available: https://www.maxonmotor.com/medias/sys_master/root/8830469799966/2018EN-132.pdf.

[12] K. J. Aström and M. R. Murray, An Introduction for Scientists and Engineers, Princeton University Press, 2009.

[13] K. Astöm and T. Häglund, PID controllers: Theory, Design and Tuning, 1988.

[14] F. Haugen, "Ziegler-Nichols' Closed-Loop Method," 2010.

[15] N. Akmal and B. Alias, "Linear Quadratic Regulator (LQR) Controller Design For Inverted Pendulum," 2013.

[16] R. Padhi, "Pole Placement Control Design," [Online]. Available: https://nptel.ac.in/courses/101108047/module9/Lecture%2021.pdf. [Accessed 05 December 2018].

APPENDIX

Matlab simulation file

```
%% Basketbot parameters

Mw = 1.7*2;          % Mass of the wheels [Kg]
Mp = 25-Mw;          % Mass of the body [Kg]
Ip = 10;             % Inertia of the body [Kg*m^2]
Iw = 0.034*2;        % Inertia of the wheel [Kg*m^2]
l = 0.4;             % Length to the body's centre of mass [m]
r = 0.2;             % Radius of wheel [m]
g = 9.81;            % Gravity [m/s^2]


%% Motor parameters

Kt = 30.2e-3;                % Motor torque constant [Nm/A]
R = 0.316;                   % Terminal Resistance [Ohm]
Ke = 1/317;                  % Back EMF constant [Vs/rad]


%% Computation of the system

% Denominatior for A and B matrices
alpha = R*(Mp*l^2+Ip);
beta = R*(Iw+Mp*r^2+Mw*l^2);
gamma = Mp*l*R;

% System matrices
A = [0 1 0; (beta*gamma*g-r^2*gamma^2)/(alpha*beta-gamma^2*r^2)
0 2*Kt*Ke*(gamma*r+beta)/(alpha*beta-gamma^2*r^2);
(gamma*r*alpha-gamma^2*r*g)/(alpha*beta-gamma^2*r^2) 0
2*Kt*Ke*(-gamma*r-alpha)/(alpha*beta-gamma^2*r^2)];
B = [0 2*Kt*(-gamma*r-beta)/(alpha*beta-gamma^2*r^2)
2*Kt*(gamma*r+alpha)/(alpha*beta-gamma^2*r^2)]';
C = [1 0 0; 0 1 0; 0 0 1];
D = [0 0 0]';

% Transfer function of the state space model
[num,den] = ss2tf(A,B,C,D);
SegwaySys = ss(A, B, C, D);
Gs = tf(SegwaySys);


%% System response

% Time  and input vector
T = 0:0.02:10;
U = zeros(size(T));
U(1) = 1;        % Input voltage

% Representation of the system impulse response
figure()
```

```
[Y,X] = lsim(SegwaySys,U,T);
plot(T,Y);
title('Open loop impulse response of the system')
ylabel('Angle[rad]')
xlabel('Time[s]')
legend('\theta','\theta dot','\psi dot')
grid on

% Representation of the system step response
figure()
step(Gs(1))
hold on
step(Gs(2))
hold on
step(Gs(3))
title('Open step response of the system')
ylabel('Angle[rad]')
xlabel('Time[s]')
legend('\theta','\theta dot','\psi')
grid on

figure()
bode(Gs(1))
grid on
title('Bode diagram of \theta tf')
figure()
bode(Gs(2))
title('Bode diagram of \theta dot tf')
grid on
figure()
bode(Gs(3))
title('Bode diagram of \psi dot tf')
grid on


%% Poles/zeros analysis

% Poles and zeros of the voltage-tilt angle transfer function
figure()
pzmap(Gs(1))
title('\theta tf: Poles and zeros')

% Poles and zeros of the voltage-angular velocity transfer
function
figure()
pzmap(Gs(2))
title('\theta dot tf: Poles and zeros')

% Poles and zeros of the voltage-wheel velocity angle transfer
function
figure()
pzmap(Gs(3))
title('\psi dot tf: Poles and zeros')

%% Pole placement

% New poles assignation (in the left-hand plane)
```

```
poles = pole(Gs(1));
negPole = poles(poles<0);
im = negPole(1)*tan(acos(0.9));
newPoles = [negPole(1)*2, (negPole(1) + im*1i),(negPole(1) -
im*1i)];

% Feedback matrix
K = place(SegwaySys.A, SegwaySys.B, newPoles);

% Close loop of the stabilized system and poles/zeros
calculation
StabilizedSys = feedback(SegwaySys, K);
figure()
pzmap(StabilizedSys);
title('Poles of the stabilized system')

% Bode plot
figure()
bode(StabilizedSys);
title('Bode stabilized system')

% Transfer function of the stabilized system
Gv = tf(StabilizedSys);

% Stabilized system response
figure()
step(Gv(1))
hold on
step(Gv(2))
hold on
step(Gv(3))
title('Step response to a perturbation with pole placement')
ylabel('Angle[rad]')
xlabel('Time[s]')
legend('\theta','\theta dot','\psi dot')
grid on

% Stabilized system response
figure()
U2 = zeros(size(T));
U2(50:60) = 1;    % Disturbance force
[Y2,X2] = lsim(StabilizedSys,U2,T);
plot(T,Y2);
title('System response to a perturbation with pole placement')
ylabel('Angle[rad]')
xlabel('Time[s]')
legend('\theta','\theta dot','\psi dot')
grid on


%% Matlab tuning PID Controller

% PID tuning
C1 = pidtune(Gv(1),'PID');
% C2 = pidtune(Gv(2),'PID');
% C3 = pidtune(Gv(3),'PID');
```

```
% Close loop connection
Gv2 = feedback(C1*Gv,eye(1,3));
% Gv2(2,1) = feedback(C2*Gv(2),1);
% Gv2(3,1) = feedback(C3*Gv(3),1);

% Response of the controlled system
figure()
step(Gv2)
grid
title('Step response of the PID controlled system (manual)')

% Stabilized system response
figure()
T2 = 0:0.02:5;
U3 = zeros(size(T2));
U3(50:60) = 1;    % Disturbance force
[Y3,X3] = lsim(Gv2,U3,T2);
hold on
% plot(T,Y2);
% title('Open loop impulse response of the controlled system')
% ylabel('Angle[rad]')
% xlabel('Time[s]')
% legend('\theta','\theta dot','\psi dot')
% grid on
subplot(3,1,1)
plot(T2,Y3(:,1));
ylabel('Angle[rad]')
xlabel('Time[s]')
title('Response of the controlled system with a PID (manual)')
grid on
subplot(3,1,2)
plot(T2,Y3(:,2));
ylabel('Angular velocity[rad/s]')
xlabel('Time[s]')
grid on
subplot(3,1,3)
plot(T2,Y3(:,3));
ylabel('Angular velocity[rad/s]')
xlabel('Time[s]')
grid on


%% Ziegler-Nichols Method for PID controller

% Critical gain and frequency
Kc1 = 920.74734;
Pc1 = 2.9730;

% Critical oscillation plot
Osc1 = feedback(Gv(1),Kc1);
figure()
step(Osc1)
title('Critical oscillation for \theta tf')

% Ziegler-Nichols parameters for the PID
Kp1 = 0.6*Kc1;
```

```
Ti1 = 0.5*Pc1;
Td1 = 0.125*Pc1;
ZN1 = pid(Kp1,Kp1/Ti1,Kp1*Td1);

% Response of the system
Gv3 = feedback(ZN1*Gv,eye(1,3));

% Response of the controlled system (Ziegler-Nichols)
figure()
step(Gv3)
grid
title('Step response of the controlled system (Ziegler-
Nichols)')

figure()
[Y4,X4] = lsim(Gv3,U2,T);
hold on
subplot(3,1,1)
plot(T,Y4(:,1));
ylabel('Angle[rad]')
xlabel('Time[s]')
title('Response to a disturbance with Ziegler-Nichols')
grid on
subplot(3,1,2)
plot(T,Y4(:,2));
ylabel('Angular velocity[rad/s]')
xlabel('Time[s]')
grid on
subplot(3,1,3)
plot(T,Y4(:,3));
ylabel('Angular velocity[rad/s]')
xlabel('Time[s]')
grid on


%% LQR control

% Check if the system is controllable
C = ctrb(SegwaySys);                    % Controllability matrix
[B AB A^2*B...]
controllability = rank(C);
if controllability==size(C)
    disp('System is controllable')
else
    disp('System is not controllable')
end

% Weight matrices (simplest case)
% Q = C'*C;
% R = 1;

% Weight matrices (another simple case)
% Q = eye(3,3);
% R = 1;

% Weight matrices
Q = [10000 0 0; 0 1 0; 0 0 100];
```

```
R = 1;

% Optimal gain matrix
K_lqr = lqr (SegwaySys,Q,R);

% Close loop of the stabilized system and poles/zeros
calculation
StabilizedSys2 = feedback(SegwaySys, K_lqr);
figure()
pzmap(StabilizedSys2);
title('Poles of the system with LQR control')

% Stabilized system response
T3 = 0:0.02:20;
U3 = zeros(size(T3));
U3(50:60) = 1;   % Disturbance force
figure()
[Y5,X5] = lsim(StabilizedSys2,U3,T3);
hold on
subplot(3,1,1)
plot(T3,Y5(:,1));
ylabel('Angle[rad]')
xlabel('Time[s]')
title('System response to a perturbation with LQR control')
grid on
subplot(3,1,2)
plot(T3,Y5(:,2));
ylabel('Angular velocity[rad/s]')
xlabel('Time[s]')
grid on
subplot(3,1,3)
plot(T3,Y5(:,3));
ylabel('Angular velocity[rad/s]')
xlabel('Time[s]')
grid on

% Transfer function of the stabilized system
Gv4 = tf(StabilizedSys2);


% Stabilized system response
figure()
step(Gv4(1))
hold on
step(Gv4(2))
hold on
step(Gv4(3))
title('Step response with LQR control')
ylabel('Angle[rad]')
xlabel('Time[s]')
legend('\theta','\theta dot','\psi dot')
grid on
```