UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

Modelling and control of a self-balancing motorcycle

Diploma Thesis

Nikolaos Marios Patsouras

Supervisor: Dr. Konstantinos Ampountolas

September 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

Modelling and control of a self-balancing motorcycle

Diploma Thesis

Nikolaos Marios Patsouras

Supervisor: Dr. Konstantinos Ampountolas

September 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Μοντελοποίηση και έλεγχος ποδηλάτου ισορροπίας

Διπλωματική Εργασία

Νικόλαος Μάριος Πατσούρας

Επιβλέπων/πουσα: Δρ. Κωνσταντίνος Αμπουντώλας

Σεπτέμβριος 2022

Approved by the Examination Committee:

Supervisor **Dr. Konstantinos Ampountolas**
Associate professor, Department of Mechanical Engineering, University of Thessaly

Member **Dr. Dimitrios Pandelis**
Professor, Department of Mechanical Engineering, University of Thessaly

Member **Dr. Spyros Karamanos**
Professor, Department of Mechanical Engineering, University of Thessaly

Acknowledgements

I would like to express the deepest appreciation to my supervisor, Dr. Konstantinos Ampountolas, who made this work possible and kept me focused on my project. His guidance and advice carried me through all the stages of writing my diploma thesis. His unique way of teaching the course of Modern Control Theory, became one of the main reasons I chose the field of automatic control for my thesis.

Moreover, I would like to to thank the examination committee members, Dr. Dimitrios Pandelis and Dr. Spyros Karamanos, for reading my project and giving me insightful comments and suggestions in order to improve it. My sincere thanks also goes to all my professors from the department of Mechanical Engineering of University of Thessaly, for passing on their knowledge and way of thinking.

Finally, I would like to give special thanks to my family and friends who supported me when undertaking my research and writing my project. Their contribution may be invisible at first glance, but I swear that it means the world to me.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I also declare that the results of the work have not been used to obtain another degree. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Nikolaos Marios Patsouras

Diploma Thesis

Modelling and control of a self-balancing motorcycle

Nikolaos Marios Patsouras

Abstract

The self-balancing motorcycle of Arduino Engineering Kit Rev 2 is based on the inverted pendulum problem, with a spinning wheel attached to it. The purpose of this thesis is to model the motorcycle and design different types of controllers and observers that manage to stabilise the continuous time system. First, the equations of motion, derived from the dynamics of the motorcycle, will be linearized and expressed in state space form. After that, the controllers designs begin with the output feedback control. PD and PID controllers will be designed with different methods. The PD and PID will also be transferred from the continuous to the discrete time and the system's response will be evaluated. Next, from the modern control theory, the state feedback control is applied. The full-state feedback controllers will be designed with two methods: the Pole placement and the Linear-Quadratic-Control. These controllers will be combined with a Luenberger observer, that estimates the states, in order to create a compensation system. Nevertheless, the real-life robot accepts disturbances from the environment and to take them into consideration, an optimal control law must be designed. This control is called Linear-Quadratic-Gaussian (LQG), and is obtained by combining the concepts of LQR and Linear-Quadratic-Estimator (Kalman filter). By designing all these control systems, one can discretize them after and decide which one is better for the real-life robot.

Keywords:

Self-balancing motorcycle, Arduino,Arduino Engineering Kit Rev 2, Output-feedback-control, State-feedback-control, PD, PID, Pole placement, Linear-Quadratic-Regulator, LQR, Luenberger observer, Linear-Quadratic-Estimator, LQE, Kalman filter, Linear-Quadratic-Gaussian, LQG, continuous time, discrete time, inverted pendulum, stability.

Διπλωματική Εργασία

Μοντελοποίηση και έλεγχος ποδηλάτου ισορροπίας

Νικόλαος Μάριος Πατσούρας

Περίληψη

Το ποδήλατο ισορροπίας του Arduino Engineering Kit Rev 2, βασίζεται πάνω στην αρχή λειτουργίας του ανάστροφου εκκρεμούς που φέρει μία περιστρεφόμενη ρόδα στο ένα άκρο του. Ο σκόπος της παρούσας διπλωματικής είναι η μοντελοποίηση του ποδηλάτου και ο σχεδιασμός διαφόρων τύπων ελεγκτών και παρατηρητών που σταθεροποιούν το σύστημα στο συνεχή χρόνο. Αρχικά, οι εξισώσεις κίνησης, που προκύπτουν από την δυναμική του ποδηλάτου, γραμμικοποιούνται και εκφράζονται στον χώρο των καταστάσεων. Έπειτα, ο σχεδιασμός των ελεγκτών αρχίζει με τον έλεγχο ανάδρασης εξόδου. PD και PID ελεγχτές θα σχεδιαστούν με διάφορες μεθόδους. Μάλιστα, οι δύο αυτοί ελεγκτές θα μεταφερθούν, από τον συνεχή χρόνο στον διακρίτο, και θα αξιολογηθεί η απόκριση του σύστηματος. Στη συνέχεια, από την μοντέρνα θεωρία ελέγχου, θα εφαρμοστεί ο έλεγχος ανάδρασης κατάστασης. Ο ελεγκτής ανάδρασης πλήρους βαθμού θα σχεδιαστεί με δύο μεθόδους: την αυθαίρετη τοποθέτηση πόλων (Pole placement) και τον γραμμικό-τετραγωνικό-έλεγχο (LQR). Αυτοί οι ελεγκτές θα συνδυαστούν με έναν πλήρους-τάξης εκτιμητή κατάστασης Luenberger, προκειμένου να δημιουργηθεί ένα σύστημα αντιστάθμισης. Ωστόσο, το πραγματικό ρομπότ δέχεται διαταραχές από το περιβάλλον και για να ληφθούν υπόψιν, ένας βέλτιστος κανόνας ελέγχου πρέπει να σχεδιαστεί. Ο έλεγχος αυτός ονομάζεται γραμμικός-τετραγωνικός-γκαουσιανός (LQG) και προκύπτει από τον συνδυασμό του γραμμικού-τετραγωνικού ελεγκτή (LQR) με τον γραμμικό-τετραγωνικό-εκτιμητή (LQE), γνωστό και ως φίλτρο Kalman. Με τον σχεδιασμό όλων αυτών των συστημάτων ελέγχου και την διακριτοποίηση τους, μπορεί να αποφασιστεί ποιό είναι το κατάλληλο για να σταθεροποιήσει το αληθινό ρομπότ.

Λέξεις-κλειδιά:

Ποδήλατο ισορροπίας, Arduino Engineering Kit Rev 2, Ανάστροφο εκκρεμές, μοντελοποίηση, Ελεγκτές, Παρατηρητές, Συνεχής χρόνος, Διακριτός χρόνος, Εξισώσεις κίνησης, Χώρος καταστάσεων, Έλεγχος ανάδρασης εξόδου, PD, PID, Μοντέρνα θεωρία ελέγχου, Έλεγχος

ανάδρασης κατάστασης, Τοποθέτηση πόλων, Γραμμικόςτετραγωνικός-έλεγχος (LQR), Εκτιμητής κατάστασης Luenberger, Σύστημα αντιστάθμισης, Γραμμικός-Τετραγωνικός-Γκαουσιανός (LQG), Γραμμικός-Τετραγωνικός-Εκτιμητής (LQE), φίλτρο Kalman.

Table of contents

List of figures

List of tables

Abbreviations

BC	Before Christ
DC	Direct current
EKF	Extended Kalman Filter
EMF	Electromotive Force
IMU	Inertial Measurement Unit
KF	Kalman Filter
LHP	Left Half Plane
LiPo	Lithium Polymer
LQE	Linear-Quadratic-Estimator
LQG	Linear-Quadratic-Gaussian
LQR	Linear-Quadratic-Regulator
LTI	Linear-Time-Invariant
MIMO	Multiple-Input-Multiple-Output
MPC	Model Predictive Control
PF	Particle Filter
PD	Proportional Derivative
PID	Proportional Integral Derivative
RHP	Right Half Plane
SISO	Single-Input-Single-Output
TF	Transfer Function
UKF	Unscented Kalman Filter
ZOH	Zero-Order-Hold

Chapter 1

Introduction

1.1 Control theory

Control theory is a branch of mathematics and engineering that has do with the behavior of *dynamical systems* and how to control them. In other words, to make any system to generate something that is desired. What is generated it is called the *output*, the system is called the *plant*, and the way the system is changing in order to generate this desired output, is called the *control input*. Control theory deals with linear or non-linear dynamical systems in continuous or discrete time. In this thesis, the majority of the analysis will be done for a linear dynamical system in continuous time.

More specifically, control theory uses mathematical models to design controllers that will guarantee a standard behavior of the system, based on the desired or the optimal one. This behavior is many-sided and its main aspects that are evaluated are: the *stability*, the *steady-state error*, the specs of the *time response* (*overshoot*,*peak time*, *rise time* and *settling time*) and the *robustness*.

Control theory can be divide in two big categories: the *Classical control theory* and the *Modern control theory*. Classical control theory deals meanly with linear time-invariant systems that accept one input and produce one output (SISO: Single-Input-Single-Output), and that are described by differential equations in the *time domain*. A very useful tool, called the *Laplace transform*, transforms these differential equations from the time domain to the *frequency domain*, while the *transfer function* relates the Laplace transform of the input and the output. The differential equations in the frequency domain are much simpler to solve but, can only describe linear systems. In classical control theory, the sensors measure the output

which is then compared with a desired control signal, called the *reference*. The difference between these two values, called the *error*, is then applied to the input of the system as *feedback*, aiming to minimize the distance between the output and the reference. [4] [5]

The other branch of control theory is the Modern control theory. In opposition to the classical control theory, modern control theory deals mainly (can be used also for SISO) with multiple-input-multiple-output systems (MIMO) and uses a time-domain state space representation, which models a system as a set of state, output and input variables related by first-order differential equations. The state-space representation is in matrix form and offers an easy and practical way to model a MIMO system. Moreover, it opens the path for modelling nonlinear system besides linear ones and with some advanced techniques it allows the design of *optimal linear controllers*. [4]

Control systems of various types can be dated back to the ancient years. Romans used state-of-the-art systems of regulating valves in their aqueducts, to keep the water level constant. Even before that, in $2000B.C.$ in ancient Mesopotamia, there are some records of controlling the irrigation system. The first modern example of control theory, is the study of oscillation of the pendulum from *Ch.Huygens* and *R.Hooke* on the 17^{th} Century. Their animus was to accomplish a precise measurement of time and location, much crucial in navigation applications. This work was later adapted to control the wings of windmills via a system of balls rotating around an axis. The findings of this study, came to be very pivotal for the invention of the steam engine by *J. Watt*, as they contributed to build a mechanism that control the valves of the engine. However, it was not until 1868 that a definitive mathematical description of a system was given by *J.C. Maxwell*. That system was no other than Watt's steam engine. From that time, control theory gained a lot of publicity and on the 20^{th} century, a big percentage of engineering companies used semi-automatic or automatic control techniques. [6]

Today, automatic control is an indispensable part of industry. It can be found in vehicles, robots, medical equipment, industrial and agricultural processes, telecommunication systems, biotechnology and elsewhere. People also interact in everyday life with systems that use automatic control. Some common examples are the cruise control in cars, the temperature control in boilers and the air conditioning systems.

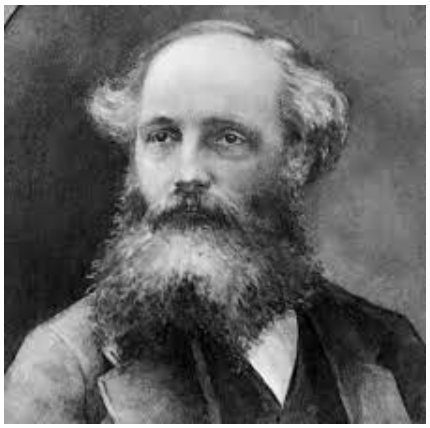

Figure 1.1: J.C Maxwell (1831-1879) [1]

1.2 The self-balancing motorcycle

Self-balancing inventions are becoming more and more famous and keep improving day by day. One branch of self-balancing machines are the self-balancing motorcycles and bicycles. These machines aim to make a two-wheeler balance -and even ride- on its own. The very first self-balancing two-wheeled for sale, was a kids bicycle called the *Jyrobike*, presented in 2014. This bicycle used a special front wheel that contained a rotating flywheel, driven by battery, to make balancing easier. The rotating flywheel spun way faster than the front wheel, acting like a gyroscope, creating a force field (like gravity) that holds the bike upright [7]. Regrading motorcycles, HONDA was the first company to lead the way towards the manufacturing of motorcycles that never fall over, with the presentation of *Honda Riding Assist* in Las Vegas Consumer Electronics Show back in 2017. The system that used was simple yet brilliant: when engaged, the system increased the fork angle, disconnecting the front forks from the handlebars. The system then makes minor steering to balance the motorcycle, without the use of heavy gyroscopes or other shifting devices. Today HONDA looks closer than ever in launching its first motorcycle with the ability to stay upright on its own and ride itself at low speeds, available for mass production. Other big companies also try to produce self-balancing mechanisms for bikes [8], [9]. Harley Davidson and BMW with its *Vision Next 100* model are some of them. As of today, the only company that accepts orders but have not yet deliver any self-balancing motorcycle, is Davinci motors with its *DC 100* model. [10]

But what is the main reason motor companies try to produce self-balancing motorcycles? a motorcycle that stands on its own enhances safety for the driver, as even the most experienced drivers face difficulties when it comes to balancing a heavy motorcycle. Especially

when stopping at the traffic lights or slowing down, balancing is an important concern. Also, it smooths the path for beginners, as they can focus on learning how to ride without worrying about falling down. [9], [11]

Figure 1.2: HONDA's riding assist prototype: World Premiere in Las Vegas Consumer Electronics Show [2]

1.3 The Arduino self-balancing motorcycle

The motorcycle project of *Arduino Engineering Kit Rev2* consists of a two-wheeled robot that can balance and move using a rotating disc, which is called *inertia wheel*, to compensate if the motorcycle loses balance. The motorcycle is controlled by an *Arduino Nano 33 IoT*, the *Arduino Nano Motor Carrier*, a DC motor to move the back wheel, DC motor with Encoder to control the inertia wheel and a standard servo motor to steer the motorcycle's handle. [3]

1.3.1 Arduino Board

A board is a device that holds together crucial components such as different processors, memory, and connectors for input and output devices. The base of a board consists of a very firm sheet of non-conductive material, typically some sort of rigid plastic. Thin layers of copper or aluminum foil, referred to as traces, are printed onto this sheet. These traces are

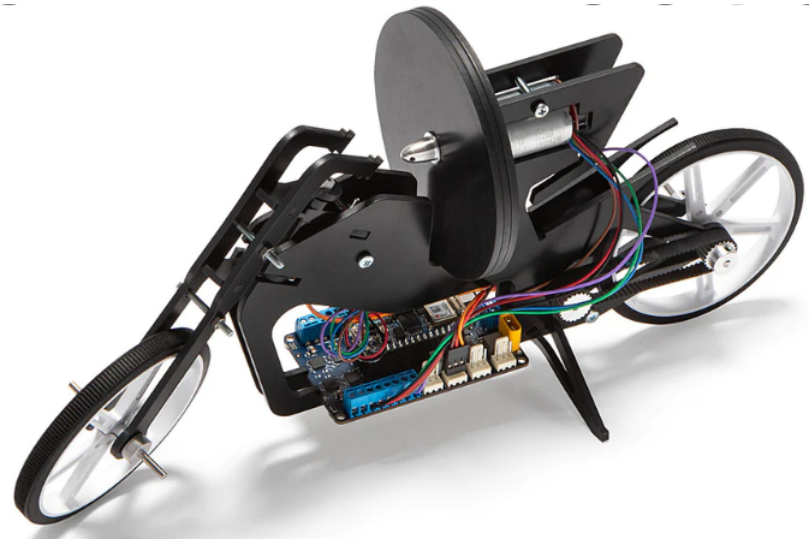

Figure 1.3: The self-balancing motorcycle that comes with the Arduino Engineering Kit Rev2 [3]

very narrow and form the circuits between the various components. In addition to circuits, a board contains a number of sockets and slots to connect other components. The Arduino Engineering Kit Rev2 contains the Arduino Nano 33 IoT board. It is a powerful board that uses an *ARM Cortex M0+*, a high-performance, 32-bit, microprocessor that is designed to allow developers to optimize power usage for specific applications with built-in, low-power features. The board also embodies an *IMU sensor* (a 3D digital accelerometer and a 3D digital gyroscope performing at $1.25mA$), a device that can be used to measure board orientation (by checking the gravity acceleration vector orientation) or to measure shocks, vibration, acceleration, and rotation speed. Furthermore, it gives the ability to communicate via WiFi or Bluetooth, as it has a multi-radio module with an integrated antenna. [3]

The Arduino Nano 33 IoT board has 19 Digital Pins and 8 Analog pins. The analog pins can transmit or receive voltage values between 0 and 3.3 volts, relative to GND, whereas the digital pins can only transmit data and voltage values of 0 or 3.3 volts, relative to GND. When they receive data, the voltage is interpreted as HIGH or LOW, relative to some threshold between 0 and 3.3 volts. Pins marked as D14-D21 or A0 - A7 can both transmit and receive voltage values and there are also called General purpose Input/Output (GPIO) pins. The GPIO pins (D14-D21) can behave as Analog pins and Digital pins depending on the configuration. It is significant to bring up that no more than 3.3 V must be applied to the board's digital and analog pins. Connecting higher voltage signals, like the 5V commonly used with the other Arduino boards, will damage the Arduino Nano 33 IoT. In addition to the analog and digital

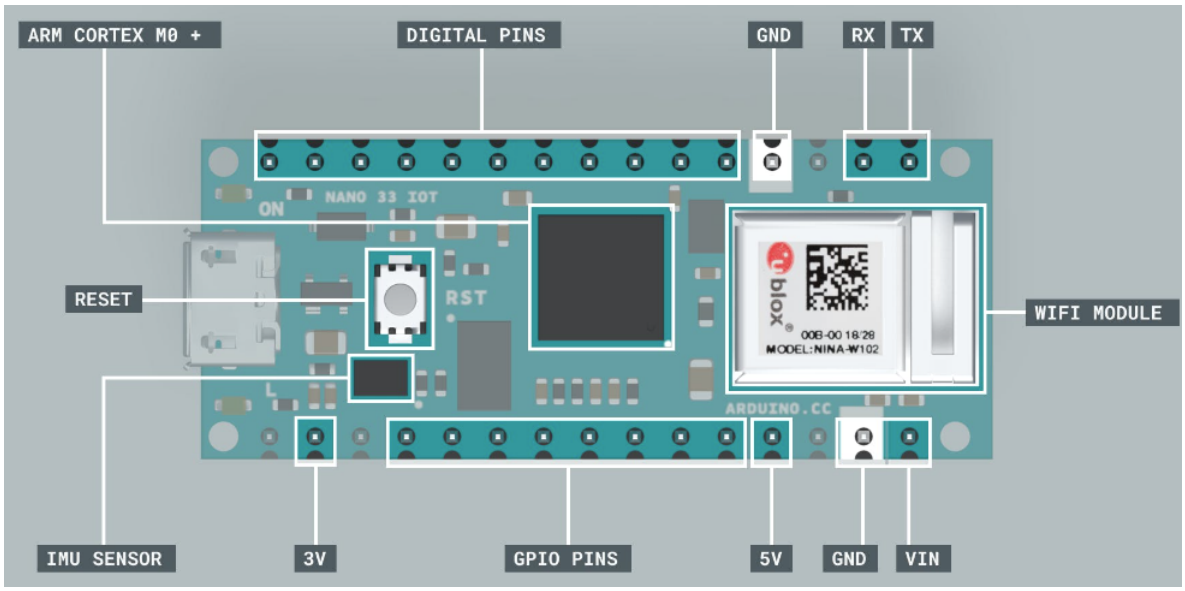

Figure 1.4: The Arduino Nano 33 IoT board [3]

pins, the board has a GND (ground), a 3V and a 5V pins that provide access to the board's ground and 5-volt reference voltage lines respectively. Furthermore, it has a Vin pin with the capability to provide power to the board from an external battery source. [3]

1.3.2 Arduino Nano Motor Carrier

Designed to facilitate motor control, the Nano Motor Carrier is a Nano add-on board directed to control servo, DC, and stepper motors. The Arduino Nano 33 IoT board is attached to the motor carrier and simplifies connecting various motors and analog sensors.

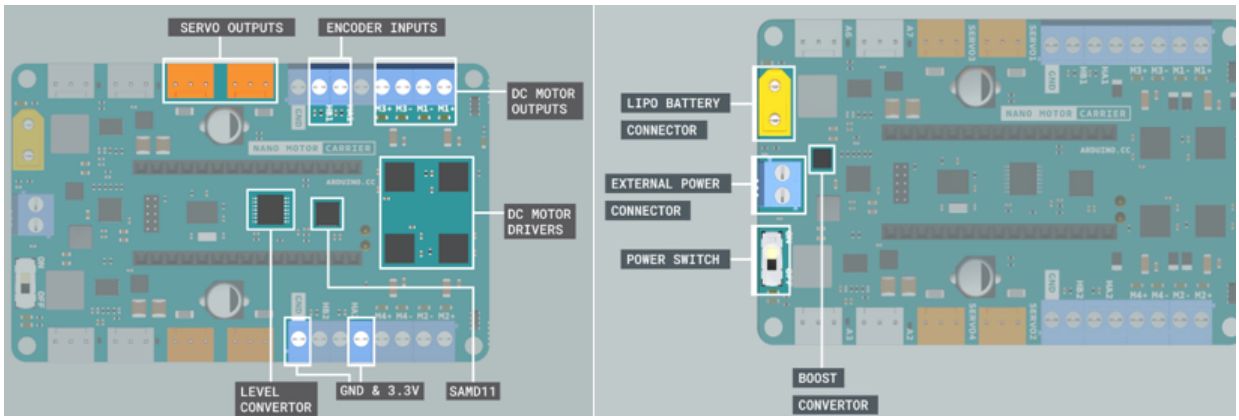

Figure 1.5: Arduino Nano Motor Carrier [3]

The Nano Motor Carrier uses a *SAMD11* microcontroller to control the servos, read values from the encoders, and read the battery voltage in an autonomous way. This microcontroller

receives commands and sends information to the Arduino Nano 33 IoT Also, it embodies 4 servo pins (orange color) that control all the communications between the servos and the board, and 4 DC motor terminal pins where a DC motor can connect to the carrier. It is possible to connect 4 DC motors at the same time but there are only 2 encoder inputs. The four black squares next to the DC terminal pins mark the 4 DC motor drivers. These drivers are responsible for high-performance DC motor control with a direct connection to the Arduino board.

In order to feed the motor drivers and power up the motors efficiently, the use of an external power source is considered necessary. The power source in this kit is a LiPo battery that is connected to the battery connector. When the connection is made it is suggested that the power switch is at the OFF position. More about the LiPo battery can be found later in this thesis. However, the LiPo battery itself is not able to provide enough voltage to drive the DC motors and that is where the boost converter comes. The boost converter lies next to the battery connector and as the name suggests, takes an input voltage and boosts or increases it, making it the perfect tool for boosting the battery's voltage. Moreover, it features the *BNO055 IMU sensor* of BOSCH, a 9-axis (acc+gyro+magnetometer) orientation sensor that plays a key role in balancing the motorcycle. [3]

1.3.3 DC motor

The self-balancing motorcycle carries two DC motors and so it is considered important to explain the technical details of the sub-components and the underlying electronic concepts used. A DC (Direct Current) motor will cause the motor shaft to rotate around its longitudinal axis when applying an electric current between its terminal pins. Thus, the DC motor is a type of actuator that transforms electrical current into rotational motion. [3]

Inside a DC motor two main parts are observed: a rotor (the shaft is part of this) that is the moving part and a stator that is the static part. The stator and the rotor use both permanent magnets and electromagnets. Depending on the type of motor, the stator can be a permanent magnet while the rotor is an electromagnet, or vice-versa. Turning on the electromagnet creates attraction and repulsion forces that make the motor spin. The DC motor spins when DC voltage is applied through its two terminal pins and so, altering the voltage level changes the motor's speed, and reversing the current's direction gives the ability to run the motor in both directions.

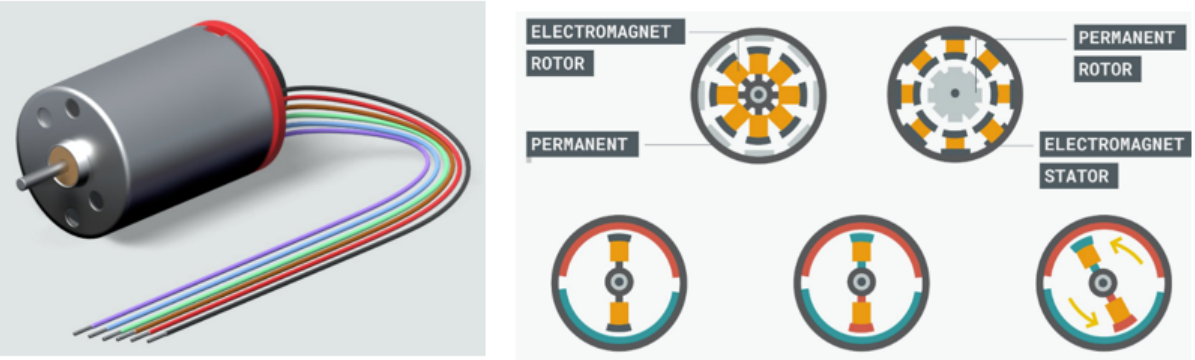

Figure 1.6: DC motor: working principal [3]

Electric motors usually need a higher voltage than a microcontroller can provide. So, in order to control a DC motor from a microcontroller without damaging it, a circuit in between is being used, called driver. This can be a transistor, a relay, or an H-Bridge. The Arduino Nano Motor Carrier uses *H-Bridges* to drive the DC motors, capable to change the speed and the direction of the motor and to provide higher power levels to drive it.

Furthermore, the DC motors of the motorcycle can calculate the motor speed via a device attached to it, called the *magnetic encoder*. An encoder sensor is a type of mechanical motion sensor that creates a digital signal from a motion. It is a device that can provide precise information on the speed, direction, and positioning of a piece of mechanical equipment. In this kit, the DC motors come with *Hall Sensor-based encoders*. The self-balancing motorcycle features one micro geared (100:1) DC motor with encoder and one bigger DC motor with encoder. The geared motor drives the motorcycle backwards and forwards by rotating the rear wheel of the motorcycle, while the other DC motor powers the inertia wheel connected to the motorcycle. The bigger DC motor is enough forceful to rotate heavy objects such as the inertia wheel. The model used in this project is the *TRK-370CA-17260-51V-EN* and has an operating voltage of 12 V, a no-load speed of 7000 rpm, a stall torque of 200 g-cm and a stall current of 1.85 A. On the other hand, the geared motor that is connected to the rear wheel is the *GM12-N20VA-08225-100-EN* and has an input voltage of 3.5 to 24 V while the maximum speed it can reach without load is 320 rpm and the stall torque is 2.2 Kg-cm. [3]

1.3.4 Servo Motor

A servo motor is a type of motor that can rotate with great precision. This motor consists of a control circuit that provides feedback on the current position of the motor shaft that allows

the servo motors to rotate with great precision. In this kit, the servo motor model is the *GS-9025MG*, which is a standard DC servo motor that can rotate between 0 and 180 degrees. As a result, it can not be used to drive a wheel, however, it can be used to move things back and forth at certain angles with high accuracy. In the self-balancing motorcycle, the servo motor is responsible for steering the front wheel. The motor has an operating voltage range of 4.8 and 6 V, and a standing torque of 2.3 Kg-m at 4.8 V. [3]

1.3.5 Inertial Measurement Unit (IMU)

The inertial measurement units are special devices that can measure a variety of factors, including speed, direction, acceleration, specific force, angular rate, and (in the presence of a magnetometer), magnetic fields surrounding the device. IMUs use a combination of accelerometers, gyroscopes and magnetometers in order to provide the right data for describing a movement, in this case, the movement of the motorcycle. Each of the sensors included in the IMU has a unique purpose. The accelerometer measures velocity and acceleration while the gyroscope detects orientation and as a result, it can measure rotation and rotation rate (angular velocity). Also, some IMUs embody a magnetometer, a device used to measure magnetic fields. Its role on IMUs is to detect the earth's magnetic field, improving the accuracy of measurements and providing a North Magnetic Pole reference point. [3]

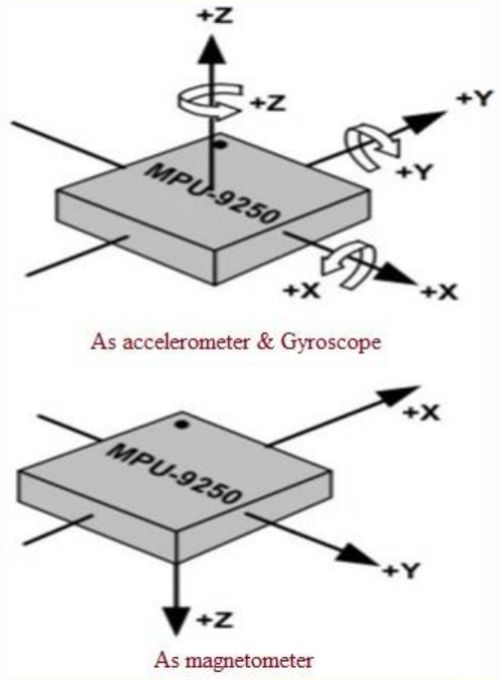

Figure 1.7: 9-axis IMU sensor

The motorcycle of this project uses the IMU BNO055 -manufactured by BOSCH-, a 9-axis orientation sensor. The IMU is used to measure the vertical position of the self-balancing motorcycle and detect when it is falling.

1.3.6 Battery

The motorcycle uses a lithium-ion battery to power its motors. These batteries (LiPo) are famous in applications where weight and shape are considered critical factors. One point worth mentioning is that Li-Ion batteries must be handled with care. Discharging the battery under the minimum safe charge can cause irreparable damage to the battery. In addition, overcharge it can cause the battery to ignite. For this reason, Li-ion batteries should always be charged with a specialized charger and should never get charged unattended. [3]

The battery that comes with the kit is Samsung's *INR18650-25R 2500mAh*. The battery has a nominal voltage of 3.6 V, and that voltage may seem low to power a $12V$ DC motor, however, the Arduino Nano Motor Carrier has a boost converter circuit that powers the motors that require a higher voltage than the battery has. The capacity and the discharge rating are equal to 2500 mAh and 20 A, respectively. Lastly, the battery weighs approximately 43.8 g and can be recharged via the Arduino Nano Motor Carrier.

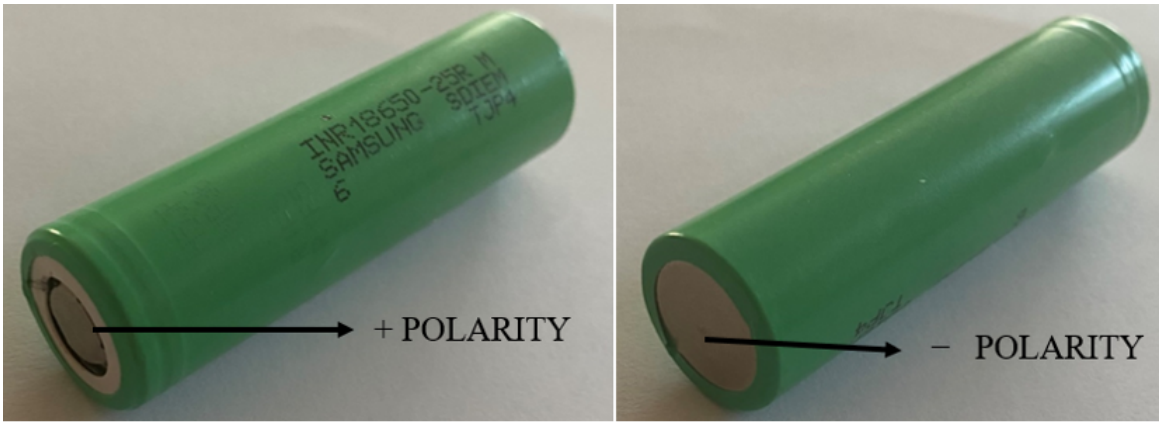

Figure 1.8: LiPo battery used by the motorcycle

1.4 Purpose

The main purpose of this thesis is creating a model that describes the dynamical system of the motorcycle and try to stabilise it. The idea is that if the model reaches stability then

the motorcycle will probably balance too. For this to happen different types of controllers and observers will be designed. The design will be based on techniques of Classical control theory as well as of Modern control theory.

The thesis is organized in 6 chapters and the current chapter is the first one.

Chapter 2 is about creating a model for the motorcycle by constructing its equations of motion, based on the dynamics of the motorcycle and the DC motor. After that, the model is linearized and the final state space model, along with its transfer function, are derived. A state space model is also presented for when the motorcycle accepts external disturbances but will not be used further.

The third chapter contains the design of output feedback control. The system is considered as SISO and PD and PID controllers are designed, using different kind of methods, in continuous and also in discrete time.

Chapter 4 deals with modern control theory. State feedback controllers are being designed, in continuous time, with pole placement and by minimising a cost criterion (Linear-Quadratic-Regulator) methods. Also, a Luenberger observer is created and is connected with the controller in order to construct a compensator system (continuous time).

Chapter 5 is about the Linear-Quadratic-Gaussian control in continuous time, which consists of the Linear-Quadratic-Regulator, already designed in chapter 4, and the Linear-Quadratic-Estimator, known as the Kalman filter. This type of control is used for system affected by noise and disturbances.

The sixth chapter contains the conclusions extrapolated from the thesis and some suggestion for further.

In the end, the bibliography that was used in order to write this thesis is listed, along with an appendix containing MATLAB codes that were developed in order to make specific calculations and create graphs.

Chapter 2

Modelling the motorcycle

2.1 Modelling

The basic goal of modelling is to build a mathematical model that will describe the dynamics of the self-balancing motorcycle. A mathematical model is defined as the number of equations representing the system's dynamics precisely or adequately. However, a mathematical model is not unique to a given system and that is because a system can be described in many varied ways, providing every time a different model. The mathematical model that will be constructed for the self-balancing motorcycle can be later used to develop output feedback, state feedback and linear quadratic control for linear or non-linear control systems. [3]

2.2 Equations of motion

2.2.1 Introduction

The first step for deriving the equations of motion that model the motorcycle is to create a schematic representation of it, which will be used to generate the free-body diagram. The problem of balancing the motorcycle can be considered as a problem of controlling an inverted pendulum and so, in this project, the schematic representation will depict an inverted pendulum, instead of a motorcycle. Truly, if someone stares at a standing motorcycle from the back, it is much alike a pendulum rod with an inertia wheel attached to it. The following assumptions, however, must be made to model the self-balancing motorcycle as an inverted

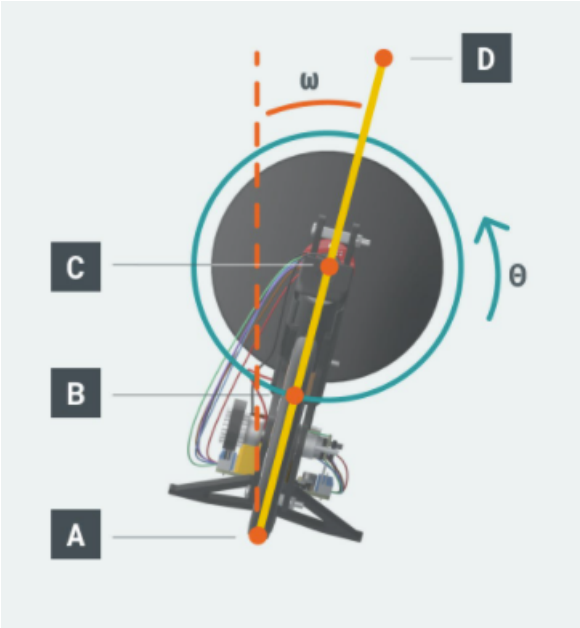

Figure 2.1: Motorcycle presented as a pendulum rod [3]

pendulum:

- The motorcycle is only free to move about the wheel-ground axis, an imaginary line connecting the points on the rear and front wheels where they touch the ground.
- The motorcycle wheels' thickness is considered insignificant.
- The friction between all the moving objects is considered negligible.

The inverted pendulum that will represent the motorcycle consists of two basic components. The first is a pendulum rod connected to the ground via a revolute joint with one degree of freedom. It is assumed that the pendulum rod is a cylinder with radius r, length l_{AD}, mass m_r and lean angle θ.θ is equal to 0 degrees when the pendulum is perfectly upright, positive when the pendulum leans clockwise, and negative when the pendulum leans counterclockwise. The second is an inertia wheel connected to the pendulum rod via another revolute joint with one degree of freedom. The inertia wheel is also modelled as a cylinder, with radius R, mass m_w, angle of rotation φ and rotational speed ω. It is important to mention that the side view of a cylinder is a rectangle and the rotation axes of two considered revolute joints are parallel to each other.

Figure 2.1 has some notable points. More specifically, **A** is the rotation axis of the revolute joint between the pendulum rod and the ground. In this context rotation, axis **A** means axis

going through **A** and perpendicular to the paper plane. Next, **B** is the center of mass of the pendulum rod. It is assumed that the pendulum rod has a uniform density, which means point **B** will be in the middle between **A** and **D**. As for **C**, it is the revolute joint's rotation axis between the pendulum rod and the inertia wheel. Also, it is assumed that the inertia wheel is mounted on the top of the pendulum and as a result, points **C** and **D** are on the same spot ($l_{AD} = l_{AC}$).

2.2.2 Dynamics of the motorcycle

The self-balancing motorcycle model is presented in Figure 2.2, in which I_{ω}^{C} is the moment of inertia of the inertia wheel and I_{r}^{B} is the moment of inertia of the of the pendulum rod.

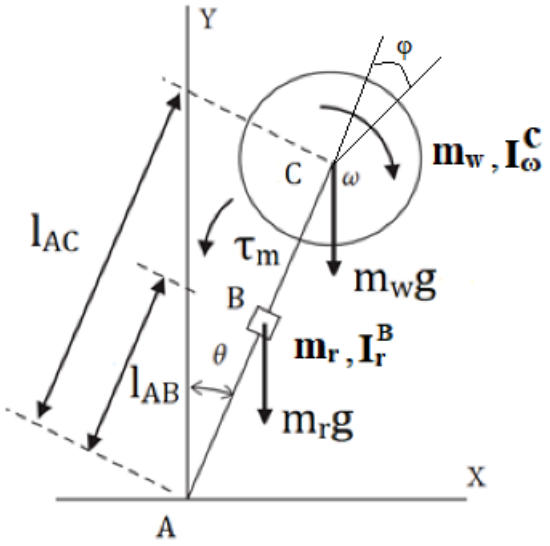

Figure 2.2: Self-balancing motorcycle model based on the principle of an inverted pendulum

As expected, the gravity force $m_r \cdot g$ is applied to the pendulum rod and the gravity force $m_w \cdot g$ is applied to the inertia wheel. The gravity force is applied to the center of mass **B** of the rod and to the point **C** of the inertia wheel, vertically to the horizontal axis. The torque of the DC motor T_m is also applied to the pendulum. This DC motor provides torque for the inertia wheel to spin, however, because the wheel is mounted on the rod, the same absolute value of torque is applied to the rod, but in different direction.

Since the system is based on the principle of an inverted pendulum with an inertia wheel attached to it, by disregarding other forces from forward movement and steering, a simplified

dynamical model of the motorcycle can be derived using the *Lagrange method*. [12]

Before starting the process, it is important to define the absolute velocities:

- Absolute velocity of **B** is $|v_B| = l_{AB}\dot{\theta}$
- Absolute velocity of **C** is $|v_C| = l_{AC}\dot{\theta}$

The Lagrange method uses the *Euler-Lagrange* equations. The total kinetic and the total potential energies are required.

$$T = \frac{1}{2}m_w|v_B|^2 + \frac{1}{2}m_r|v_C|^2 + \frac{1}{2}I_r^B\dot{\theta}^2 + \frac{1}{2}I_w^C\omega^2 + I_w^C\dot{\theta}\omega \tag{2.1}$$

$$\Rightarrow T = \frac{1}{2}(m_r l_{AB}^2 + m_w l_{AC}^2 + I_r^B + I_w^C)\dot{\theta}^2 + \frac{1}{2}I_w^C\omega^2 + I_w^C\dot{\theta}\omega \tag{2.2}$$

$$V = g \cdot \cos\theta \cdot (m_r l_{AB} + m_w l_{AC}) \tag{2.3}$$

Where T is the total kinetic energy of the system and V is the total potential energy of the system. Euler-Lagrange equations can be derived by using:

$$\frac{d}{dt}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = Q_i \tag{2.4}$$

where q_i are the generalized coordinates, $Q_i, i = 1, 2, 3, ..$ denotes the external forces corresponding to q_i, and L is the *Lagrangian operator* representing the difference between the total kinetic and total potential energy, given by the following equation:

$$L[\theta, \omega, \dot{\theta}] = T - V, \tag{2.5}$$

With $q_i = \theta$, equation (2.4) gives:

$$(m_r l_{AB}^2 + m_w l_{AC}^2 + I_r^B + I_w^C)\ddot{\theta} + I_w^C\dot{\omega} - g\sin\theta(m_r l_{AB} + m_w l_{AC}) = 0 \tag{2.6}$$

With $q_i = \varphi$, equation (2.4) gives:

$$I_w^C\dot{\omega} + I_w^C\ddot{\theta} = T_m \tag{2.7}$$

2.2.3 Dynamics of the DC motor

The dynamics of the DC motor play a vital role in forming the kinematic equations of the system. The self-balancing motorcycle uses a DC motor with magnet and encoder to spin the inertia wheel. A free body diagram of a DC motor is presented in figure 2.3. [13]

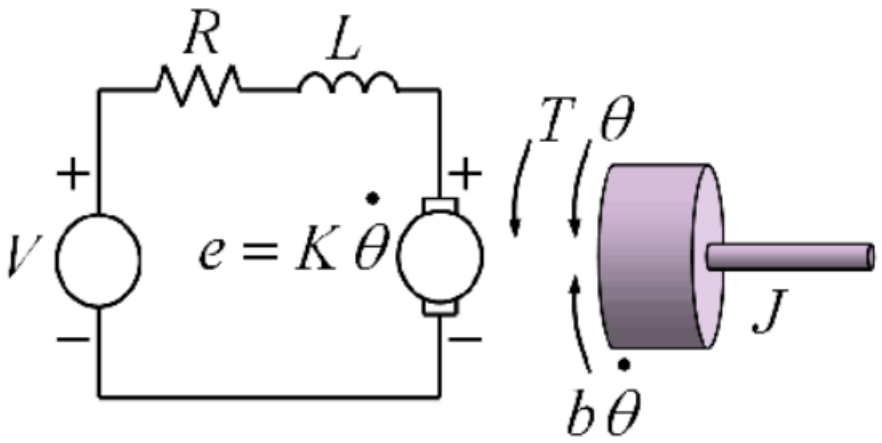

Figure 2.3: Free body diagram of DC motor

Kirchhoff's voltage law states that directed sum of the voltages around any closed loop is equal to zero. Applying this to the circuit of figure 2.3 gives:

$$L_m \frac{d_i}{d_t} + R_m i + e - V_m = 0 \tag{2.8}$$

where V_m is the motor supply voltage, e is the counter-electromotive force or back EMF, L_m is the armature coil inductance, R_m is the armature coil resistance and i is the armature current. Also, the back EMF can be substituted by the following equation:

$$e = K_e \omega_m = K_e \omega \tag{2.9}$$

where K_e is the motor's back EMF constant and ω_m the angular speed of the motor. It is known that $\omega_m = a\omega$, where a is the gear ratio. However, the DC motor of the motorcycle has $a = 1:1$ and so $\omega_m = \omega$. Next, considering that the motor's torque is proportional to i:

$$T_m = K_t i \tag{2.10}$$

where K_t is the torque constant of the motor.

The term of inductance is considered negligible ($L_m = 0$) since, the motor inductance is much less than the motor resistance ($L_m \ll R_m$). So, from equations (2.8), (2.9) and (2.10):

$$T_m = \frac{K_t}{R_m}(V_m - K_e \omega) \tag{2.11}$$

Implementing (2.11) to (2.7):

$$\boxed{I_w^C \dot{\omega} + I_w^C \ddot{\theta} = T_m = \frac{K_t}{R_m}(V_m - K_e \omega)} \tag{2.12}$$

Equations (2.6) and (2.12) are the kinematic equations of the system. Obviously, they are nonlinear as (2.6) contains the $\sin\theta$ term.

2.3 Linearization of the model

The differential equation (2.13) is nonlinear and so, finding its solution is a difficult task and few times impossible. So, when the model is nonlinear and complicated, it is often advantageous to find approximations of it. Under specific conditions, the nonlinear system can be replaced with an approximate linear one. The linearization takes place around the equilibrium points and that is because many systems settle into an equilibrium state after some time, providing in that way information about the long-term behavior of the system. Such information can be whether or not the point is stable or unstable, as well as how the system approaches or moves away from the equilibrium point. Linearizing about non-equilibrium points does not give anything useful.

The term that makes the equation (2.13) nonlinear is the $\sin\theta$ term. The linearization will be done with the *Taylor series* expansion of functions. Considering a function $f(\theta) = \sin(\theta)$ of a single variable θ, and supposing that $\bar{\theta}$ is a point that $f(\bar{\theta}) = 0$ (equilibrium point). The Taylor series expansion of $f(\theta)$ around the equilibrium point $\bar{\theta} = 0$ is given by:

$$f(\theta) = \sum_{n=0}^{\infty} \frac{f^{(n)}(\bar{\theta})}{n!} + (\theta - \bar{\theta})^n = f(\bar{\theta}) + f'(\bar{\theta})(\theta - \bar{\theta}) + \frac{f''(\bar{\theta})}{2!}(\theta - \bar{\theta})^2 + \frac{f'''(\bar{\theta})}{3!}(\theta - \bar{\theta})^3 + ...$$

$$f(\theta) = \sin(\theta) = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + ...$$

Ignoring the higher order terms (because angle θ is considered very small, i.e. $\theta \approx 0$)

$$f(\theta) = \sin(\theta) \approx \theta$$

So, equation (2.6) with Taylor's linearization becomes:

$$\boxed{(m_r l_{AB}^2 + m_w l_{AC}^2 + I_r^B + I_w^C)\ddot{\theta} + I_w^C \dot{\omega} - g\theta(m_r l_{AB} + m_w l_{AC}) = 0} \tag{2.13}$$

Which is now a linear differential equation. So the final kinematic equations are (2.12) and (2.13).

2.4 State space model

There are plenty of ways to describe a system of linear differential equations and one of them is the state space representation. Basically, it is a mathematical model that describes a physical system as a set of input, output and state variables related by first-order differential equation. The continuous-time form of a state space model of a *Linear Time-Invariant* (LTI) system can be represented as follows: [14]

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(0) = \mathbf{x}_0, t \geq 0$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad t \geq 0$$

where:

- $\mathbf{x} \in \Re^n$ is the state vector
- $\mathbf{u} \in \Re^m$ is the input/control vector
- $\mathbf{y} \in \Re^p$ is the output vector
- $\mathbf{A} \in \Re^{n\,x\,n}$ is the state matrix
- $\mathbf{B} \in \Re^{n\,x\,m}$ is the control matrix
- $\mathbf{C} \in \Re^{p\,x\,n}$ is the output state matrix
- $\mathbf{D} \in \Re^{p\,x\,m}$ is the output control matrix

The state vector $\mathbf{x}(t)$ is a set of state variable that actuate as an internal description of the system and can represent the entire state of the system at any given time. The dimension n of the state vector $\mathbf{x}$ is called the *order of the system*. The input vector $\mathbf{u}(t)$ represents the variables that can be controller and the output vector $\mathbf{y}(t)$ the variables that are measured depending on the needs of the problem and the capabilities of the sensors. The matrices $\mathbf{A}$ and $\mathbf{B}$ are the properties of the system and are determined by the system structure and elements (physics). The output equation matrices $\mathbf{C}$ and $\mathbf{D}$ are determined by the particular choice of output variables.

For the self-balancing motorcycle, the state variables $\mathbf{x}$ are the pendulum rod lean angle θ, its first time derivative $\dot{\theta}$ that represents the angular velocity of the rod and the rotational speed ω of the inertia wheel.

Setting:

$\mathbf{x} = \begin{bmatrix} x_1 = \theta \\ x_2 = \dot{\theta} \\ x_3 = \omega \end{bmatrix}$ as state variables, $\mathbf{y} = \theta$ as output variable and $\mathbf{u} = V_m$ as control variable,

and for convenience, $A_1 = (m_r l_{AB}^2 + m_w l_{AC}^2 + I_r^B + I_w^C)$ and $B_1 = (m_r l_{AB} + m_w l_{AC})$

Next, setting:

$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 = \dot{\theta} \\ \dot{x}_2 = \ddot{\theta} \\ \dot{x}_3 = \dot{\omega} \end{bmatrix}$ and implementing them into equations (2.12) and (2.13), produces:

$$\begin{aligned} \dot{x}_1 &= \dot{\theta} = x_2 \\ \dot{x}_2 &= \ddot{\theta} = \frac{gB_1}{A_1 - I_w^C} x_1 + \frac{K_t K_e}{R_m(A_1 - I_w^C)} x_3 - \frac{K_t}{R_m(A_1 - I_w^C)} u \\ \dot{x}_3 &= \dot{\omega} = -\frac{gB_1 x}{A_1 - I_w^C} x_1 - \frac{A_1 K_t K_e}{R_m I_w^C (A_1 - I_w^C)} x_3 + \frac{A_1 K_t}{R_m I_w^C (A_1 - I_w^C)} u \end{aligned}$$

So, from the above equations it is quite simple forming matrices **A** and **B**:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 \\ \frac{gB_1}{A_1 - I_w^C} & 0 & \frac{K_t K_e}{R_m(A_1 - I_w^C)} \\ -\frac{gB_1}{A_1 - I_w^C} & 0 & -\frac{A_1 K_t K_e}{R_m I_w^C (A_1 - I_w^C)} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ -\frac{K_t}{R_m(A_1 - I_w^C)} \\ \frac{A_1 K_t}{R_m I_w^C (A_1 - I_w^C)} \end{bmatrix}$$

and given that the output is only the angle θ and the system does not face yet any disturbances, matrices **C** and **D** are:

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

So, implementing the values of the parameters from table 2.1 the complete state space model is produced:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 96.52 & 0 & 0.0194 \\ -96.52 & 0 & -0.667 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ -1.0235 \\ 34.7743 \end{bmatrix} V_m \tag{2.14}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} V_m. \tag{2.15}$$

Parameters	Value	Unit
I_r^B	$4.46 \cdot 10^{-4}$	$kg \cdot m^2$
l_{AB}	0.065	m
I_w^C	$8.68 \cdot 10^{-5}$	$kg \cdot m^2$
l_{AC}	0.13	m
m_r	0.2948	kg
m_w	0.0695	kg
K_e	0.019	$V \cdot s/rad$
K_t	0.019	$N \cdot m/A$
R_m	6.48	Ω
g	9.81	m/s^2

Table 2.1: Parameters of the self-balancing motorcycle

2.5 Transfer function of the linear system

One of the most convenient and widely used ways of representing a system is by its *transfer function* (TF). For a continuous-time system with input signal $x(t)$ and output signal $y(t)$, the transfer function $G(s)$ is the ratio of the Laplace transform of the output, $Y(s)$, to the Laplace transform of the input, $U(s)$: [15]

$$G(s) = \frac{Y(s)}{U(s)} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}$$

The transfer function can provide many useful information for the system such as the poles and zeros, that define whether or not a system is stable. Creating the transfer function from the state space model becomes a simple task with the use of MATLAB. As seen in MATLAB CODE the transfer function is produced by calling commands [num,den]=ss2tf(A,B,C,D) and tf(num,dem):

$$G(s) = \frac{-1.024s - 0.008}{s^3 + 0.667s^2 - 96.52s - 62.51} \tag{2.16}$$

Also, by using the command zpk(tf(num,dem)) , the zero-pole-gain model is created:

$$G(s) = \frac{-1.0235(s + 0.0078)}{(s - 9.816)(s + 9.835)(s + 0.6475)} \tag{2.17}$$

From the transfer function in (2.17), it is obvious that the poles of the systems are $p_1 = 9.816, p_2 = -9.835$ and $p_3 = -0.6475$, and the zero of the system is $z_1 = -0.0078$. As expected, the system is unstable as long as it has one pole (p_1) on the right-hand complex plane (RHP). The poles and zeros of a transfer function sometimes have complex parts. So, they can be represented graphically by plotting their locations on the complex s-plane, whose axes represent the real and imaginary parts of the complex variable s. These plots are called pole-zero maps and the zero location is marked by a circle ($\circ$), whether the pole's is marked by a cross (x). [16]

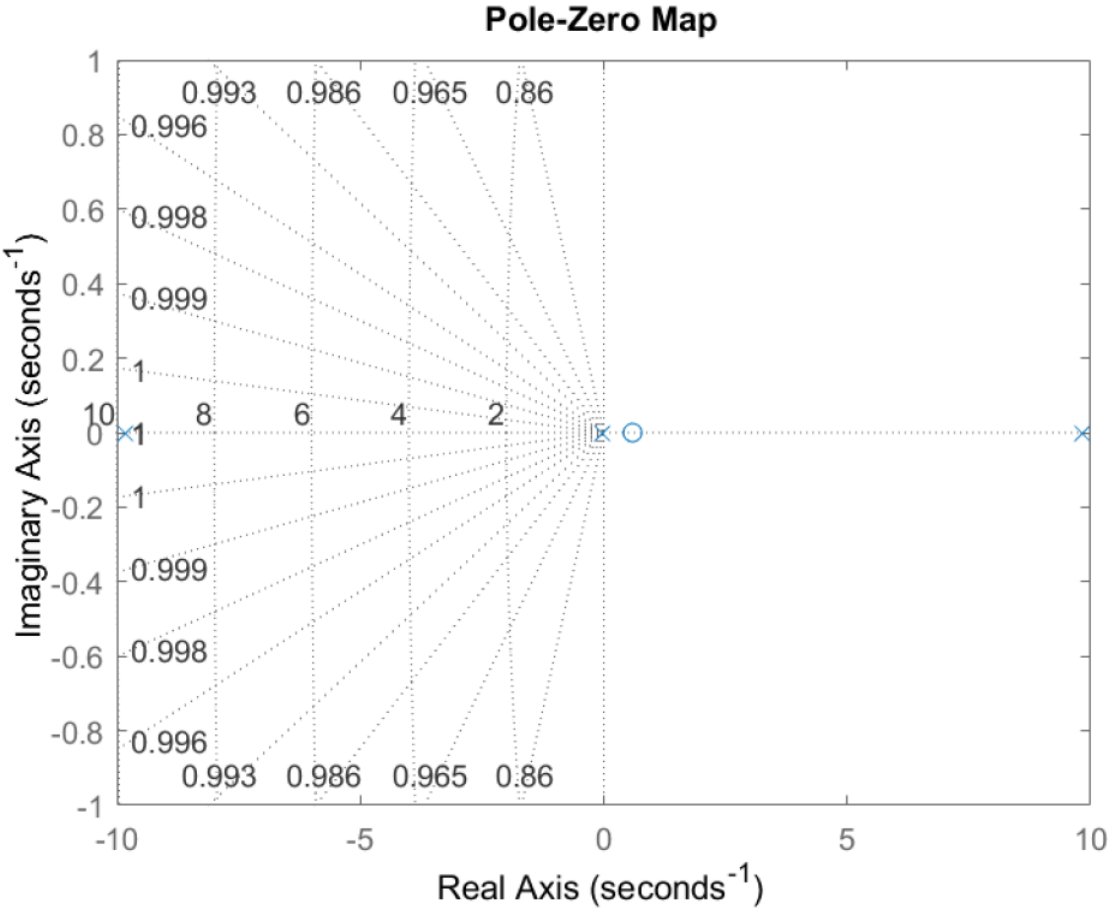

Figure 2.4: Pole-zero map of the system

Another important graph is the root locus. The purpose of the root locus method is to evaluate the variation of the poles of the closed loop transfer function with respect to some system parameter, usually the proportional control gain K of the system. The main advantage of root locus is to check the system behaviour by adjusting the value of gain K . The adjustment in the value of K will trace the roots accordingly on the RHP and will give the conditions for the system to be overdamped, underdamped, critically damped, overdamped

or undamped.

From equation (2.17) the control gain K is observed to be negative. This means that to plot the root locus on MATLAB, one has to call the command rlocus(-tf(num,dem)) . Without the minus sign, the command will produce the complementary root locus.

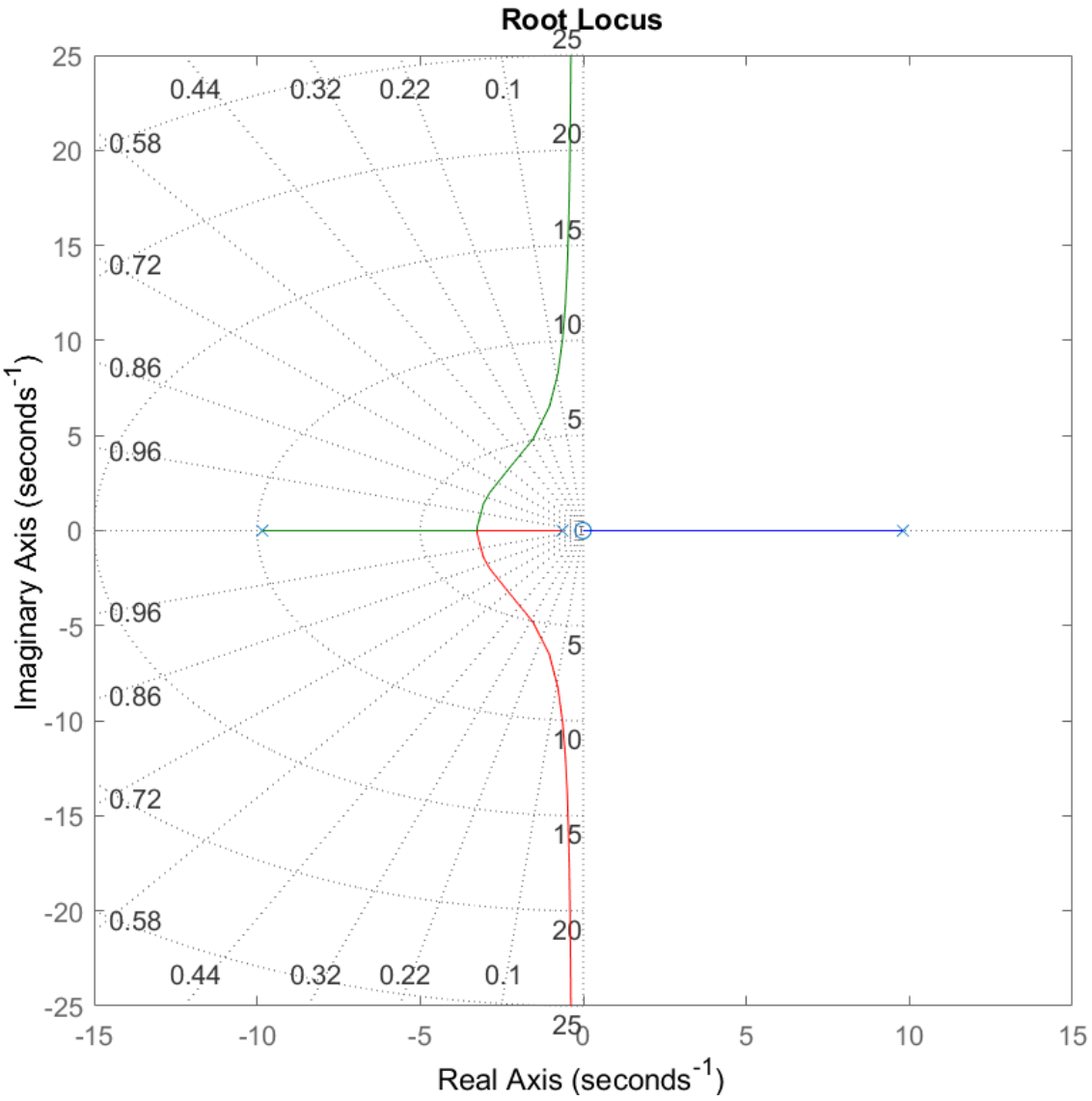

Figure 2.5: Root locus of the system

2.6 Disturbance modelling

In the real world, the presence of disturbances is a very common issue. Almost every system in nature is disturbed by external factors and the self-balancing motorcycle is not an exception. Wind forces or other external forces (push with hand) are the most usual distur-

bances that the robot faces. By modelling the system under disturbances, the model describes the behavior of the real system even better. The symbol d represents an external force that is applied to rod's mass center and will actuate as the disturbance.

With the presence of d, the only thing that changes is that in equation (2.6), the external forces Q_i are no more zero but equal to d, while (2.12) remains the same. So, the kinematic equation (2.13) becomes:

$$\boxed{(m_r l_{AB}^2 + m_w l_{AC}^2 + I_r^B + I_w^C)\ddot{\theta} + I_w^C\dot{\omega} - g\theta(m_r l_{AB} + m_w l_{AC}) = d} \tag{2.18}$$

The next step is the description of equations (2.12) and (2.18) in the state space. The state variables $\mathbf{x}$, the output variable $\mathbf{y}$, and the output vector $\mathbf{u}$ stays the same. However, a disturbance vector $\mathbf{d}$, equal to the disturbance force d, is added to the state system and so:

$$\mathbf{x} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix}, \qquad \mathbf{y} = \theta, \qquad \mathbf{u} = V_m, \qquad \mathbf{d} = d$$

Next, setting:

$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x}_1 = \dot{\theta} \\ \dot{x}_2 = \ddot{\theta} \\ \dot{x}_3 = \dot{\omega} \end{bmatrix}$, $\mathbf{u} = u_1 = V_m$, and implementing them into equations(2.12)and(2.18), while

$A_1 = (m_r l_{AB}^2 + m_w l_{AC}^2 + I_r^B + I_w^C)$ and $B_1 = (m_r l_{AB} + m_w l_{AC})$, produces:

$$\begin{aligned}
\dot{x}_1 &= \dot{\theta} = x_2 \\
\dot{x}_2 &= \ddot{\theta} = \frac{gB_1}{A_1 - I_w^C}x_1 + \frac{K_t K_e}{R_m(A_1 - I_w^C)}x_3 - \frac{K_t}{R_m(A_1 - I_w^C)}u_1 + \frac{1}{A_1 - I_w^C}d \\
\dot{x}_3 &= \dot{\omega} = -\frac{gB_1 x}{A_1 - I_w^C}x_1 - \frac{A_1 K_t K_e}{R_m I_w^C(A_1 - I_w^C)}x_3 + \frac{A_1 K_t}{R_m I_w^C(A_1 - I_w^C)}u_1 - \frac{1}{A_1 - I_w^C}d
\end{aligned}$$

So, from the above equations it is quite simple understanding that matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$ remain the same as (2.14) and (2.15). However this time, a new matrix $\mathbf{F}$ is added to describe the effect of the disturbance force d:

$$\mathbf{F} = \begin{bmatrix} 0 \\ \dfrac{1}{A_1 - I_w^C} \\ -\dfrac{1}{A_1 - I_w^C} \end{bmatrix} \tag{2.19}$$

Replacing the parameters with their values from table (2.1), the state space model of the system under disturbances is derived:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 \\ 96.52 & 0 & 0.0194 \\ -96.52 & 0 & -0.667 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ -1.0235 \\ 34.7743 \end{bmatrix} V_m + \begin{bmatrix} 0 \\ 349 \\ -349 \end{bmatrix} d \tag{2.20}$$

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} V_m. \tag{2.21}$$

Chapter 3

Output Feedback control

In the previous chapter, the state space model that describes the dynamics of the motorcycle was created. This model will be used in the current chapter to create Proportional-Integral-Derivative (PID) feedback controller. This controller will be examined for the stability that provides to the system, with or without the existence of disturbances. Moreover, because the self-balancing robot is a digital system, the controller will not only be examined in continuous time, but also in discrete time.

3.1 Proportional-Integral-Derivative Control

The PID controller, just like other controllers (P, PD, PI), attempts to correct the error between a measured process variable and a desired setpoint by calculating the difference and then performing a corrective action to adjust the process accordingly. A PID controller controls a process through three parameters: Proportional (P), Integral (I), and Derivative (D). These parameters can be weighted, or tuned, to adjust their effect on the process. Due to robust performance and functional simplicity, PID controllers have been used in industrial applications where a more precise control is essential. As a result, PID controller is the most frequent controller used in industrial automation and applications, meaning that more than 95% of the industrial controllers are of this type.

3.1.1 Basic principles of a PID controller

PID-control correlates the controller output to the error, integral of the error, and derivative of the error. The behavior of a PID controller is mathematically illustrated in equations

(3.1) and (3.2):

$$u(t) = K_p e(t) + K_i \int_0^T e(t)d(t) + K_d \frac{de(t)}{dt}, \tag{3.1}$$

or in the s-domain (frequency domain):

$$C(s) = K_p + K_i \frac{1}{s} + K_d s \tag{3.2}$$

where:

- $u(t)$: controller output
- $\mathrm{e}(t) = \mathrm{r}(t) - \mathrm{y}(t)$: error
- K_p : gain of the proportional P term
- K_i : gain of the integral I term
- K_d : gain of the derivative D term

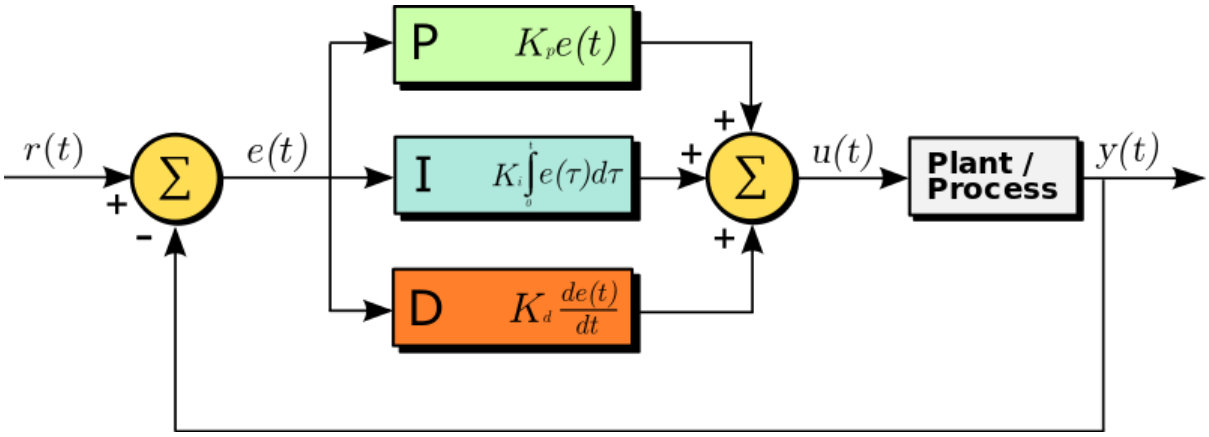

Figure 3.1: Schematic diagram of a PID controller

So, a PID controller continuously calculates an error value $e(t)$ as the difference between a desired setpoint $r(t)$ and a measured variable $y(t)$ and applies a correction based on the gains of the proportional, integral, and derivative terms. The P, I, and D term each has a very specific role in the overall control: [17]

- The P term is proportional to error value $e(t)$ and as a result, the proportional gain K_p is multiplied with the error. A high proportional gain results in a substantial change in the output for a given change in the error and a faster response. However, if the

proportional gain is too high, the system can face oscillations and become unstable and have a larger overshoot. In contrast, a slight gain results in a small output response to a significant input error and a less responsive controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances. Also, bad tuning of the P term may produce a deviation from the set point. This deviation is known as the offset and is usually not desired in a process. The existence of an offset implies that the system could not be maintained at the desired set point at a steady state. The offset can be minimized by combining P-control with another form of control, such as I or D control.

- The integral term I in a PID controller sums the instantaneous error over time (integral of the error) and gives the accumulated offset (steady-state error) that should have been corrected previously. The accumulated error is then multiplied by the integral gain K_i and added to the controller output. I term is often used because it can remove any deviations that may exist and so eliminates the steady-state error. Although integral action can eliminate the steady state error, it can strongly contribute to controller output overshoot and possible instability.

- The derivative term D in a PID controller is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The larger the rate of the change in error, the more pronounced the controller response will be. D-control anticipates the process conditions by analyzing the change in error. It functions to minimize the change of error, thus keeping the system at a consistent setting. D controllers' primary benefit is reducing overshoot and improving settling time. However, sometimes it may slow the response or worsen the steady-state error.

So, the most crucial part is finding the correct values of the gains to achieve the optimum results. This task is often tricky, given that the effects of the gains usually counteract each other and here is where the so-called tuning takes place. Tuning is the adjustment of its control parameters (proportional, integral, and derivative gains) to the right values for the desired control response. There are several methods for tuning a PID loop. One of the most famous and easiest is the trial-and-error method. It steps through the parameters from proportional to integral to derivative. Usually, the user starts from an existing set of parameters from which he performs small tweaks to improve the response. When altering the values, it is necessary to

Increased Parameter	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
K_p	Decrease	Increase	Small Increase	Decrease	Decrease
K_i	Small Decrease	Increase	Increase	Large decrease (close to zero)	Decrease
K_d	Small Decrease	Decrease	Decrease	Small change	Improve

Table 3.1: Effect of K_p, K_i and K_d on the response of the system

remember that The P-action is introduced to increase the speed of the response, the I-action is introduced to obtain a desired steady-state response, and the D-action is introduced for damping purposes. Although, the trial-and-error method is easy, quick and does not require high process knowledge, it is time-consuming and does not guarantee a robust and stable response. Another well-known tuning method used in the industry is *Ziegler-Nichols*. This method starts by zeroing the integral and differential gains and then raising the proportional gain until the system is unstable. The value of K_p at the point of instability is called K_u and the frequency of oscillation is f_0. The method then backs off the proportional gain a predetermined amount and sets the integral and differential gains as a function of f_0. The P, I, and D gains are set according to a specific table. Ziegler-Nichols method is relatively simple, intuitive and provides satisfying results for simple loops. However, it results in an oscillatory closed loop response and has high proportional gains, low integral action with too low damping of the closed-loop system and too low robustness against changes in the process dynamics, including non-linearities. In this thesis, the tuning for the PID controller will be done via MATLAB with built-in functions that can automatically tune the gains of a PID controller to achieve a balance between performance and robustness and with the Ziegler-Nichols method. These methods will be then compared in order to find which one provides better results.

3.1.2 Ziegler-Nichols method for PD & PID design

A popular method for tuning PID controllers is the Ziegler–Nichols method. . This method proposes rules for determining values of proportional gain K_p, integral time T_i, and derivative

Control Type	K_p	T_i	T_d	K_i	K_d
P	$0.5K_u$	–	–	–	–
PI	$0.45K_u$	$P_u/1.2$	–	K_p/T_i	–
PD	$0.8K_u$	–	$P_u/8$	–	K_pT_d
PID	$0.6K_u$	$0.5P_u$	$P_u/8$	K_p/T_i	K_pT_d

Table 3.2: Ziegler Nichols formula

time T_d based on the transient response characteristic of a given plant. To determine these parameters, two methods were presented by Ziegler and Nichols: step response method and self-oscillation method. In this thesis, the self-oscillation method will be used. This method starts by zeroing the integral and differential gains and then raising the proportional gain until the closed-loop system starts oscillating sinusoidally. The minimum amount of controller gain necessary to sustain sinusoidal oscillations is called ultimate gain K_u , while the time (period) between successive oscillation peaks is called the ultimate period P_u of the process. These values will be used to calculate the gain values of the controller, according to table (3.2). [18] [19] [20]

3.1.2.1 PID with Ziegler Nichols

It was calculated that $K_u = -7760$ and $P_u = 0.07$ sec. So, from table (3.2) the gain values are: $K_p = -4657$, $K_i = -130000$ and $K_d = -41.3$. The impulse response of the system under this PID controller is presented in figure (3.2).

As observed, the system reaches stability, however, its behavior is far from ideal. The *peak amplitude* or *overshoot* is 0.0107 rad ($= 0.6°$) and the time to reach it (*peak time*) is 0.06 sec. Also, the transient time, which is the *settling time* t_s for $\pm 2\%$ is 0.695 sec. From this angle, the behavior is very good, as the robot does not have a big maximum lean angle and it reaches stability quickly. It is in the oscillations of the system the problem lies. These oscillations increase the settling time and may cause instability to the system. The self-balancing motorcycle is very vulnerable to oscillations, even in small ones, and that is the reason this PID controller is not efficient.

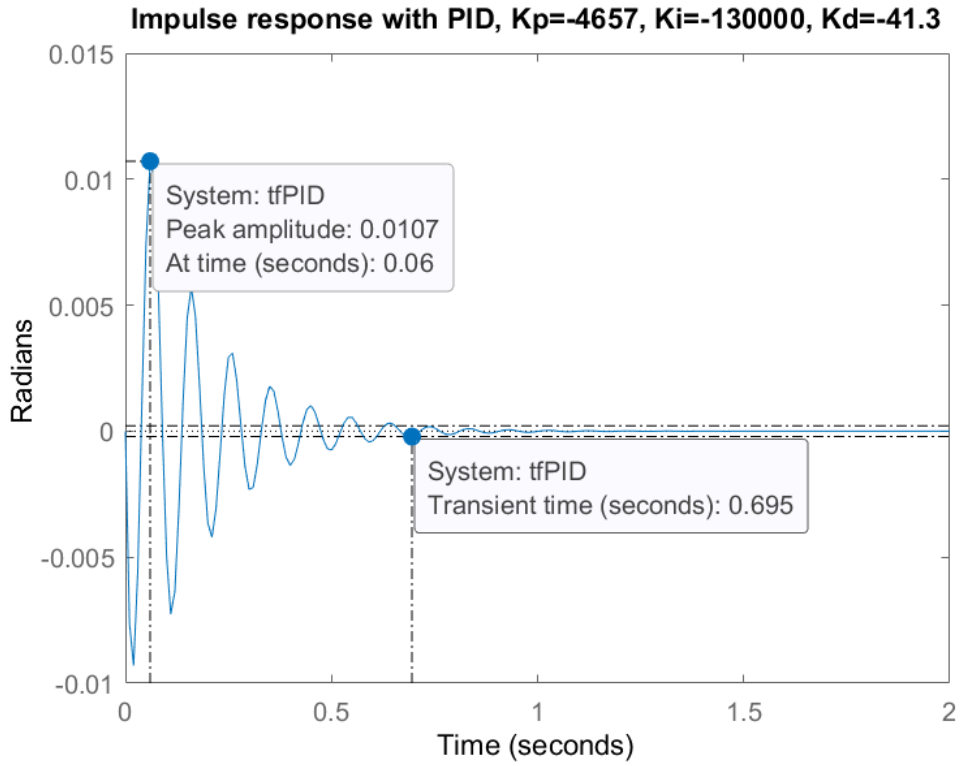

Figure 3.2: Impulse response of the system under PID control, designed with Ziegler Nichols method

3.1.2.2 PD with Ziegler Nichols

I derivative may set the steady state error close to zero, however, it produces a lot of oscillations. In this chapter, the effect of a PD controller will be examined. From table (3.2), $K_p = -6209$ and $K_i = -55.1$. The impulse response of the system under this PD controller is presented in figure (3.3).

The system reaches stability but this time with better overshoot and settling time than under the PID of sub-chapter (3.1.2.1). Overshoot is now -0.0078 rad, rise and settling time are reduced to 0.02 sec and 0.152 sec, respectively. The closed-loop poles are: $p_1 = -28.5 + 73i$, $p_2 = -28.5 - 73i$, and $p_3 = 0$. Moreover, oscillations have been reduced but are still present. As mentioned earlier, the robot is -by nature- very vulnerable to oscillations and it would be for the best to be eliminated. To face this problem, K_p is set equal to -2209 and K_i to -70. These gains may slightly augment the overshoot to -0.00938 rad and the settling time to 0.0178 sec, but on the other hand they handle very well the system's oscillations, as observed in figure (3.4). Considering the raises of overshoot and settling time negligible, the

second PD offers better response. The closed loop poles of the system under the second PD are: $p_1 = -36 + 29.2i$, $p_2 = -36 - 29.2i$, and $p_3 = 0$.

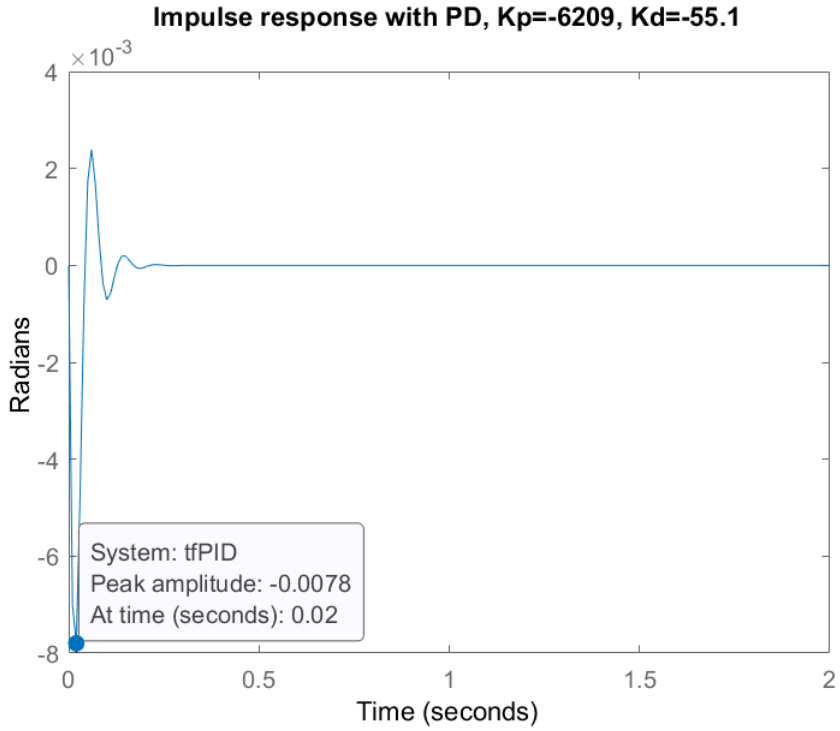

Figure 3.3: Impulse response of the system under PD control, designed with Ziegler Nichols method

3.1.3 PID design with MATLAB

MATLAB provides tools for automatically choosing optimal PID gains which makes the trial and error process less effective and more time consuming. However, the gain values that MATLAB provides are not optimal and so small alternations may take place to achieve better results, depending every time on the needs of the problem. The effect of gain values shown in table (3.1) should be advised before a gain value is changed. The tuning algorithm can be accessed directly through the pidtune command. With the use of this command, the gains of the PID are: $K_p = -256$, $K_i = -733$ and $K_d = -22.4$.

The crossover frequency is 20 rad/sec and the phase margin (PM) is $60°$. Also, as presented in figure (3.5), the settling time t_s is 1.26 sec. Next, the peak amplitude is -0.0304 rad $(= -1.74°)$ and the peak time is 0.08 sec. The response of the system is characterized as decent, because the peak amplitude is rather small and the settling time acceptable. However,

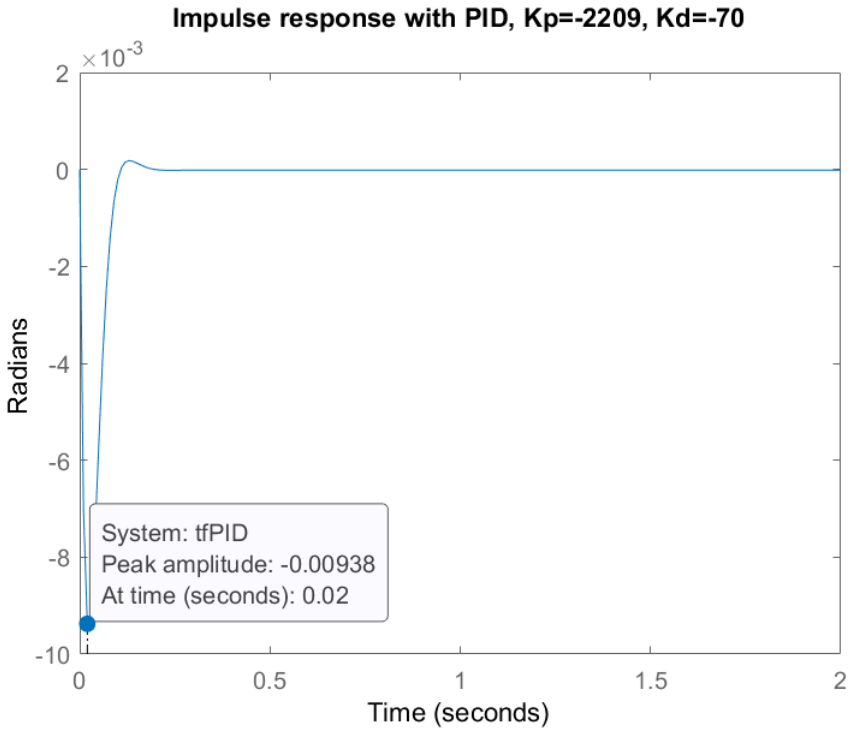

Figure 3.4: Impulse response of the system under PD control, with other gain values to reduce oscillations

MATLAB's PID tuner does not guarantee optimal behavior. As a result, if better response of the system is needed, other gain values (close to those provided by MATLAB) must be searched. For this, the trial and error method will be used, in which the gain values undergo small changes to improve the response. In the self-balancing motorcycle, the more time the motorcycle searches for its stability point, the more likely is to fall. So, it is pivotal to try to minimize the settling time, without affecting the peak amplitude. It was observed that the high K_i value was increasing the settling time. After decreasing the value of K_i and trying a series of combinations for the other gain values, having always in mind table (3.1), the settling time was reduced to $0,4$ sec, while the peak amplitude faced also a minor decrease, as it was settled to -0.0297 rad at peak time 0.08 sec. The decrease of the settling time was significant, as it was almost one second (68% decrease). Another important outcome was that the response did not faced oscillations.The gain values that managed this behavior are: $K_p = -280$, $K_i = -60$ and $K_d = -23$. The poles of the closed loop system are: $p_1 = -12 + 6.6i$, $p_2 = -12 - 6.6i$, $p_3 = -0.0029 + 0.05i$, and $p_4 = -0.0029 - 0.05i$.

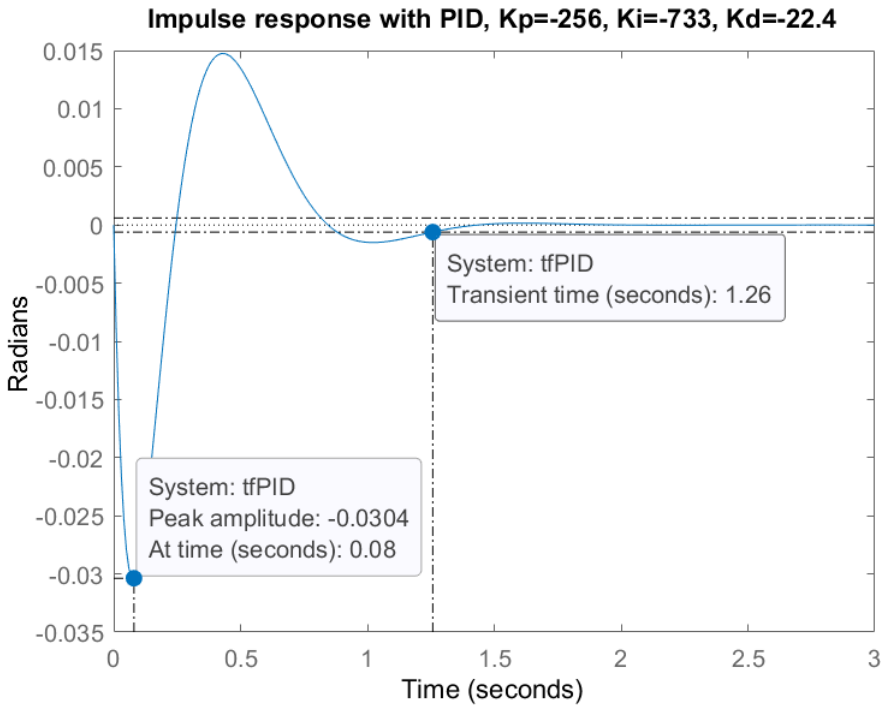

Figure 3.5: Impulse response of the system under PID control, designed with MATLAB's pidtune

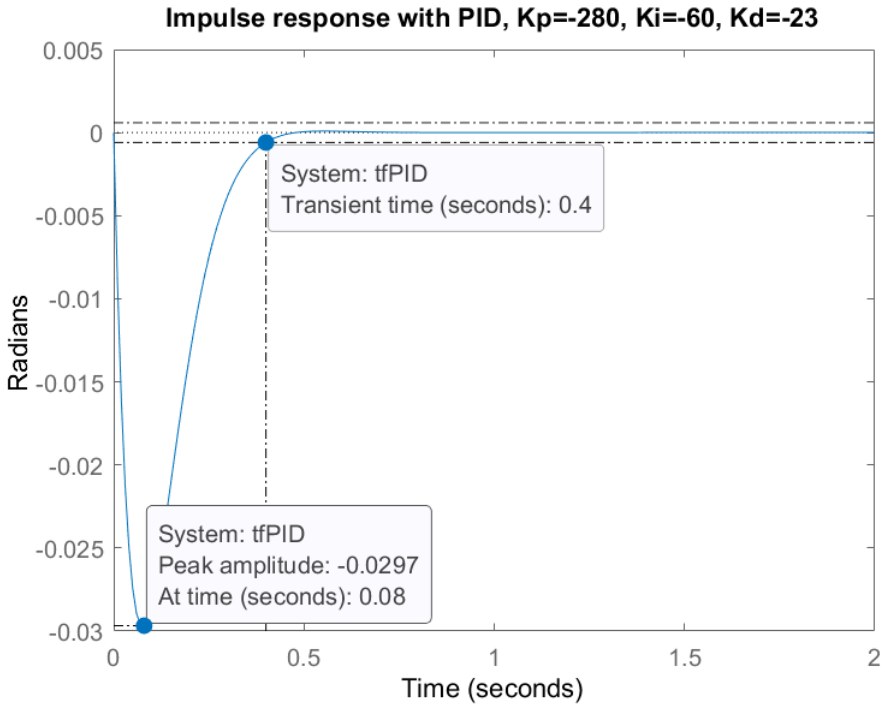

Figure 3.6: Impulse response of the system under PID control, with trial and error method

3.1.4 PD Vs PID

One of the three poles of the initial system (2.16) is close to zero (p_3). What the PID controller does, is adding another pole close to zero to achieve better response. From the theory, two poles exactly at zero make the system go unstable. To prevent this from happening, the two poles lie very near but not exactly at zero. This is the case with the PID of figure (3.6) and as observed the impulse response is stable. However, having two poles two zero makes the system even more vulnerable to instabilities. So, the PID controller in this problem is legitimate but impractical and should not be used to control the robot.

On the other hand, the PD controller does not add poles to the system. With the Ziegler-Nichols method for PD design, two out of three closed-loop poles have a big imaginary part ($+73i$ and $-73i$) and that is the reason the impulse response (3.3) presents some oscillations. By changing the gain values of PD controller, the reduction of the imaginary parts ($+29.2i$ and $-29.2i$) lowers the oscillations (3.4).

So after all, for this problem the PD controller is preferred over PID, as it manages very good settling times and overshoots and at the same time reduces oscillations without being vulnerable to instabilities.

3.2 PD-PID design in discrete time

The PD and PID controllers that were designed, functioned in the continuous time. However, the real motorcycle, in order to work, uses an Arduino microprocessor. The Arduino is a digital controller whose inputs and outputs are defined at discrete time instances. So, to implement the PD, designed on the previous chapter, on the real robot, it is necessary that the closed-loop control function, under the PD, undergoes discretization. For the discretization, the *Zero-order hold* (ZOH) method will be used, a method which is widely used to reconstruct the signals in real time with the help of the *analog-to-digital converter* (DAC) of the Arduino. In the zero-order hold reconstruction method, the continuous signal is reconstructed from its samples by holding the given sample for an interval until the next sample is received. Therefore, the zero-order hold generates the step approximations. [21] [22]

The zero-hold order is one of the methods of *Data hold* processes. Data hold is a process of generating a continuous-time signal $h(t)$ from a discrete-time sequence $x(kT)$. The signal $h(t)$ during the time interval $kT \leq t \leq (k+1)T$ may be approximated by a polynomial in

τ as:

$$h(kT+\tau) = a_n\tau_n + a_{n-1}\tau_{n-1} + \ + a_1\tau + a \tag{3.3}$$

where $0 \leq \tau \leq T$ and $h(kT) = x(kT)$ and so:

$$h(kT+\tau) = a_n\tau_n + a_{n-1}\tau_{n-1} + \ + a_1\tau + x(kT) \tag{3.4}$$

If the data hold circuit is an n^{th}-order polynomial, it is called an n^{th}-order hold. It uses the past $n+1$ discrete data $x((k-n)T), x((k-n+1)T), , x(kT)$ to generate $h(kT+\tau)$. In this case, the zero-order hold means that $n = 0$ and so the equation (3.4) becomes:

$$h(kT+\tau) = x(kT), \quad 0 \leq \tau \leq T, \quad k = 0, 1, 2, ... \tag{3.5}$$

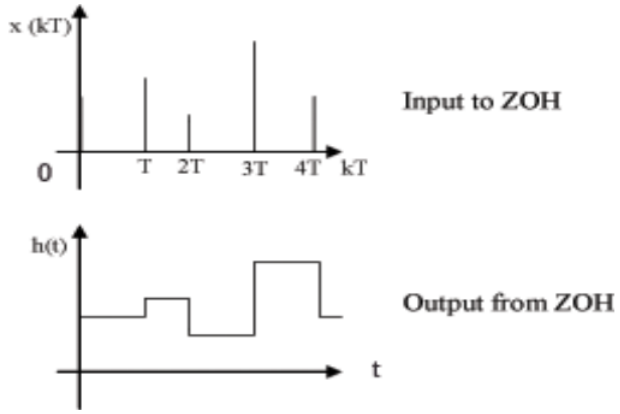

Figure 3.7: The process of reconstruction by zero-order hold

Moreover, the sampling period T should be specified. If a big sampling period is selected, it will possibly cause instability to the system and so, it is important that the right value is chosen. The sampling time for this problem is 0.01 sec.

3.2.1 PD in discrete time

The discretization will happen in MATLAB environment with the use of the c2d(tf,Ts,'ZOH') command. The discrete-time transfer function under the PD with $K_p = -2209$ and $K_d = -70$ becomes:

$$H(z) = \frac{-4.009\ 10^{-5}z^2 + 8.608\ 10^{-6}z + 3.148\ 10^{-5}}{z^3 - 2.334z^2 + 1.819z - 0.4852} \tag{3.6}$$

In discrete-time, an LTI system is considered asymptotically stable when all the poles of $H(z)$ lie strictly inside the unit circle in the z-plane. In other words, all poles must have a

magnitude strictly smaller than one the. If one pole lies exactly in the unit circle the system is considered marginally stable. The zeros and poles of the discrete transfer function are presented in figure (3.8). As observed, all poles lie inside the unit circle, except on that lies exactly at the unit circle and so the discrete-time system is marginally stable. The impulse response under PD controller in discrete time is presented in figure (3.9) and it is compared with that in continuous time. It is clear that both systems reach stability and their settling times and peak amplitudes are almost equal to each other.

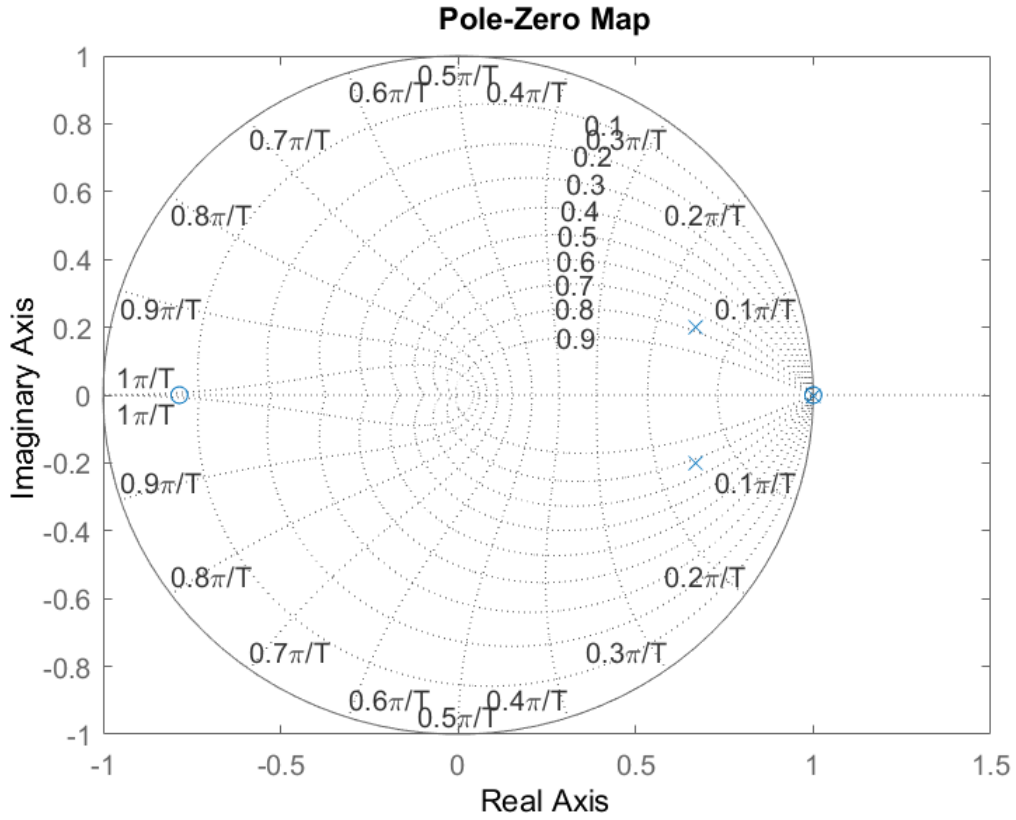

Figure 3.8: Pole-zero map of the discrete-time system under PD, $K_p = -2209$,$K_d = -70$

3.2.2 PID in discrete time

The discrete-time transfer function under the PID with $K_p = -280$, $K_d = -23$ and $K_i = -60$ becomes:

$$H(z) = \frac{-4.721\ 10^{-5}z^3 + 5.087\ 10^{-5}z^2 + 3.989\ 10^{-5}z - 4.355\ 10^{-5}}{z^4 - 3.768z^3 + 5.321z^2 - 3.338z + 0.785} \tag{3.7}$$

The zeros and poles of the discrete transfer function are presented in figure (3.10). Two of the poles of the system lie exactly at the unit circle. In this case, the system loses its stability.

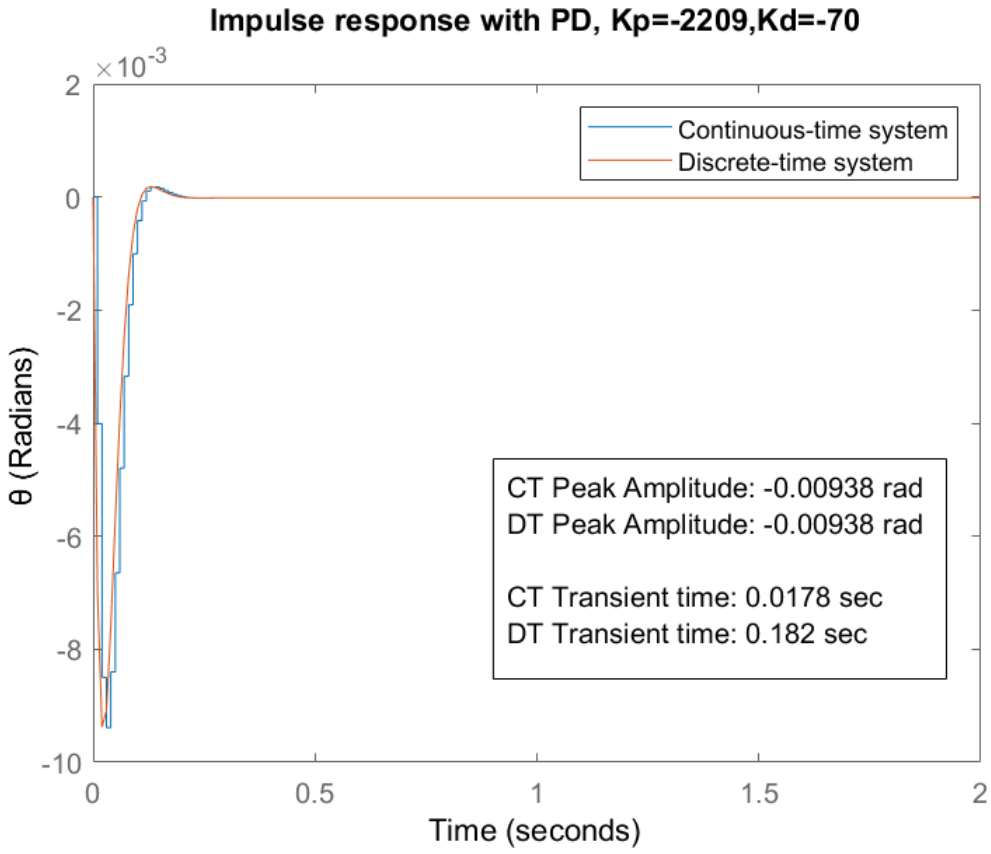

Figure 3.9: Impulse response of the system under PD controller in continuous and discrete time, $K_p = -2209, K_d = -70$

This can be also confirmed by the impulse response of the system under this PID, presented in figure (3.11). So, the PID in continuous time makes the system stable, but in the discrete time it can not manage to do the same. This is a very good example why the system must always be examined how is acting on the discrete time (where the real robot works), and not only on the continuous time.

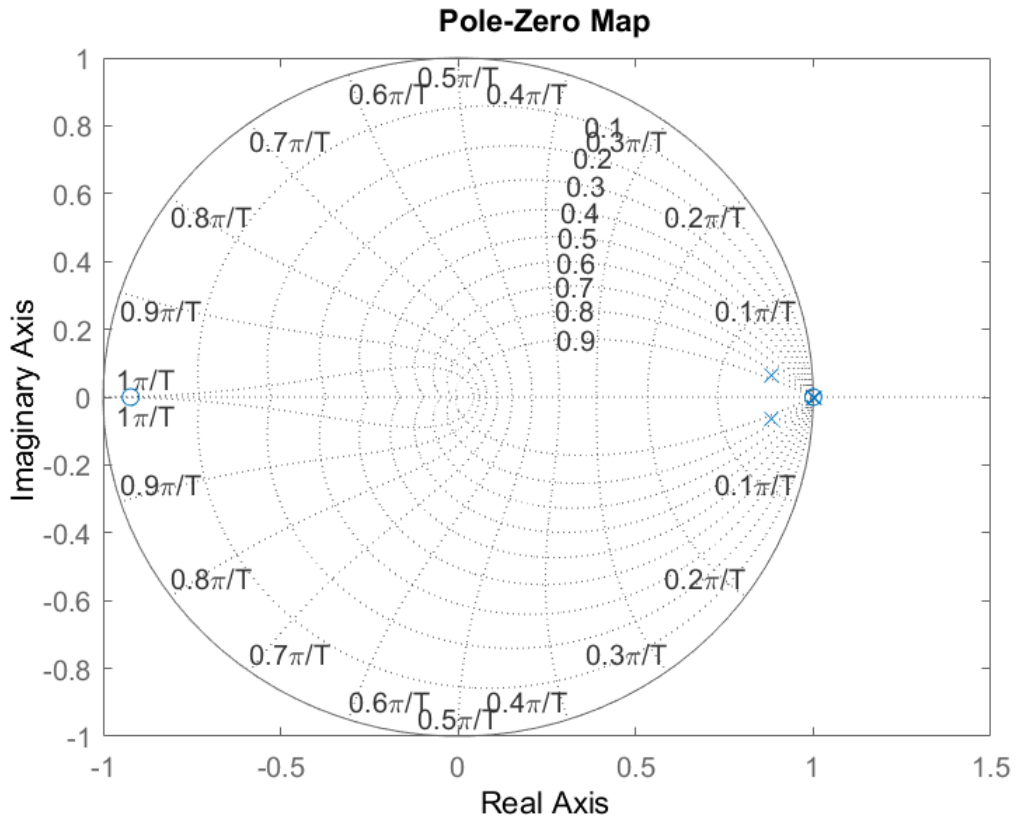

Figure 3.10: Pole-zero map of the discrete-time system under PID

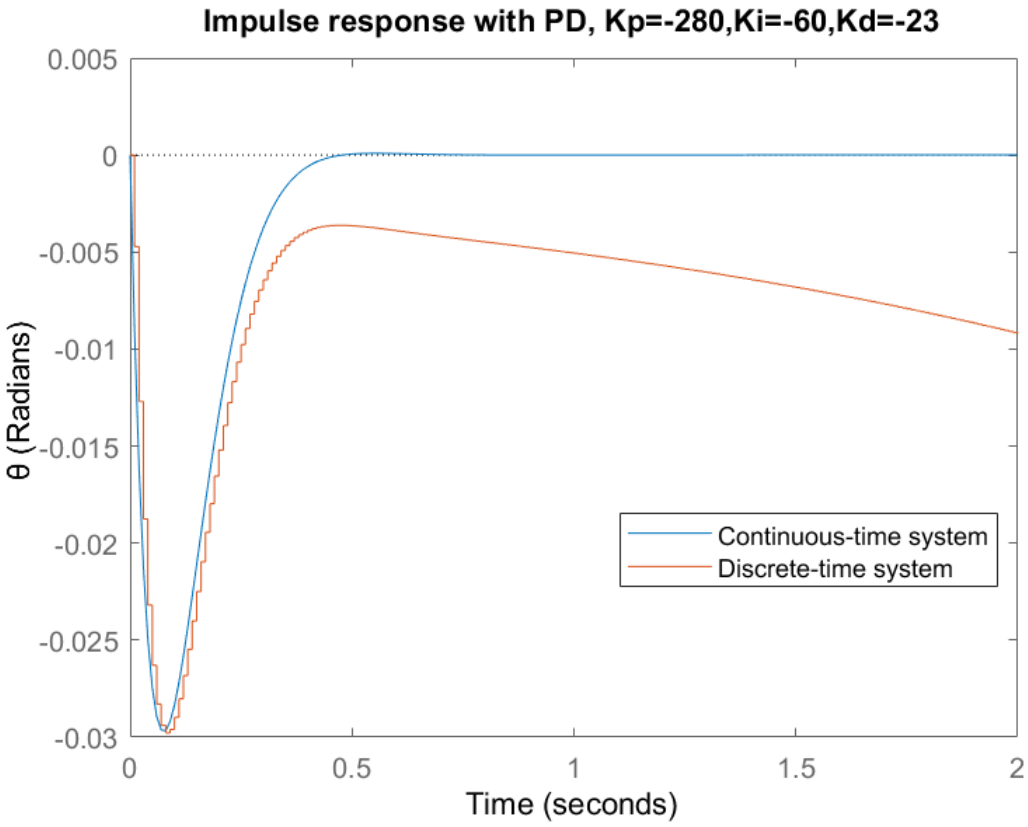

Figure 3.11: Impulse response of the system under PID controller in continuous and discrete time, $K_p = -280$,$K_d = -23$ and $K_i = -60$

Chapter 4

State feedback control

4.1 Introduction

In this chapter, the idea of designing a controller that feedbacks the state variables of the system will be discussed. First, assuming that all the state variables are known, the *state feedback control* is designed. However, most of the times, it is not possible to measure all the states and so designing one is impractical. In that case, a *state observer* is designed. Its purpose is to estimate all the states that can not be measured in the first place. There are two types of state observers: the full-order and the reduced-order. In the last step, the observer is combined with the state feedback controller and give the compensator.

Before the design, some standards will be set in order to evaluate if the results are reasonable or not:

- Settling time $t_s < 5$ for all state variables.
- Maximum lean angle $\theta < 6^\circ$ or 0.1 rad.
- Steady state error $e_{ss} < \pm 2\%$ for all state variables.

4.2 Controllability and Observability

The state feedback control aims to change the location of the poles and place them somewhere that assures stability and non-oscillatory response. However, there are some prerequisites in order to place the poles arbitrary while knowing all the state variables. Controllability

and observability represent two major concepts of modern control system theory. These concepts were introduced by *R. Kalman* in 1960. Controllability is one of the most critical aspects of process systems' operability and that is because when a system is considered state controllable, the poles or eigenvalues can be moved or placed in arbitrary places in the s-plane by using state feedback control. Observability is another important aspect of a system. Many systems do not have a sensor for measuring every state, but if a system is considered observable, then the states can be estimated. Considering a linear time-invariant state model in continuous time: [23] [24]

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(0) = \mathbf{x}_0, t \geq 0$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \quad t \geq 0$$

The controllability of a system is defined as follows:

A system is *completely state controllable* (or just controllable) if the initial state $\mathbf{x}(t_0)$ can be driven to any other state $\mathbf{x}(t)$ in finite time using the control input $\mathbf{u}(t)$.

This simply means that when a system is controllable, the state can be adjusted through an external input signal to guide it to any desired state. A necessary and sufficient condition for controllability of a LTI continuous- or discrete-time system (i.e., with **A**, **B** time-invariant) is that the controllability matrix $\boldsymbol{C}$ has rank n:

$$rank(\boldsymbol{C}) = rank[\mathbf{B} \;\; \mathbf{AB} \;\; \mathbf{A^2B} \;\; ... \;\; \mathbf{A}^{n-1} \;\; \mathbf{B}] = n \tag{4.1}$$

The controllability matrix of the robot's system, presented in equations (2.14) and (2.15), can be derived from MATLAB using the command ctrb(A,B) :

$$\boldsymbol{C} = \begin{bmatrix} 0 & -1.0235 & 0.6746 \\ -1.0235 & 0.6746 & -99.2423 \\ 34.7743 & -23.1945 & 114.2630 \end{bmatrix} \tag{4.2}$$

It is calculated that $rank(\boldsymbol{C}) = 3$ and so the system is completely state controllable.

The observability of a system is defined as follows: A system is *completely state observable* (or just observable) if the initial state $\mathbf{x}(t_0)$ can be determined from the system output $y(t)$ and the control input $u(t)$ in finite time.

So, observability is defined as how well the internal states of a system can be inferred from knowledge of that system's external outputs. A necessary and sufficient condition for

observability of a LTI continuous- or discrete-time system (i.e., with **A**, **C** time-invariant) is that the controllability matrix $\boldsymbol{O}$ has rank n:

$$rank(\boldsymbol{O}) = rank[\mathbf{C^T} \;\; \mathbf{A^T C^T} \;\; (\mathbf{A^T})^2\mathbf{C^T} \;\; ... \;\; (\mathbf{A^T})^{n-1} \;\; \mathbf{C^T}] = n \tag{4.3}$$

The observability matrix of the robot's system, presented in equations (2.14) and (2.15), can be derived from MATLAB using the command obsv(A,C) :

$$\boldsymbol{O} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 96.52 & 0 & 0.0194 \end{bmatrix} \tag{4.4}$$

It is calculated that $rank(\boldsymbol{O}) = 3$ and so the system is completely state observable.

4.3 Full-state feedback control

In chapter 3, the basic idea was feedbacking the system's output to control the system. In full-state feedback control, the idea is to feedback the state vector to compute the control action for specified system dynamics. Figure (4.1) shows a diagram of a typical control system using full-state feedback. The full system consists of the process dynamics, the controller elements, K and k_r, the reference input, r, and processes disturbances, d. The goal of the feedback controller is to regulate the output of the system, y, such that it tracks the reference input in the presence of disturbances and uncertainty in the process dynamics. [25]

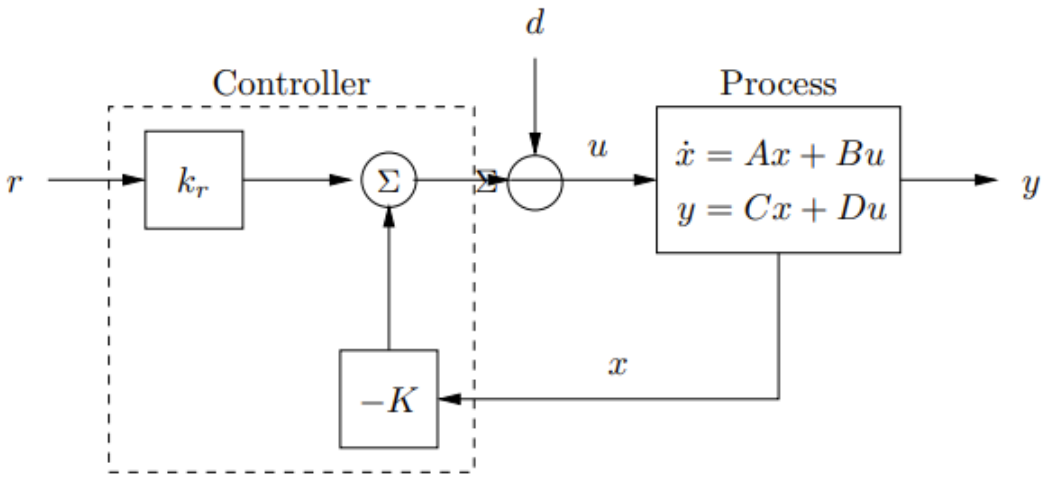

Figure 4.1: Full-state feedback block diagram

For the full-state feedback control, the state model remains the same as in (2.14) but, the output model changes:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(0) = \mathbf{x}_0, \quad t \geq 0 \tag{4.5}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \Rightarrow \mathbf{y} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ \omega \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} V_m. \tag{4.6}$$

Assuming that all components of the state vector $\mathbf{x}(t)$ are measured and ignoring the disturbance signal d, the state-feedback control can be described by the following equation:

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) + \mathbf{k}_r\mathbf{r}(t) \tag{4.7}$$

Where $\mathbf{K} \in \Re^{1xn}$ is the *control matrix* or *state-feedback gain matrix* of the controller and $r(t)$ is the reference value. In this problem, it is considered that the reference value is equal to zero for every $t \geq t_0$. So, equation (4.7) becomes:

$$\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t) \tag{4.8}$$

The state equation of the closed-loop system obtained when the feedback is applied to the system is given by implementing equation (4.8) in:

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}(t) \tag{4.9}$$

As a result, the poles of the closed loop system, with the full state feedback applied, can be found by the eigenvalues of matrix $\mathbf{A} - \mathbf{B}\mathbf{K}$, for known $\mathbf{K}$. The most common ways to calculate matrix $\mathbf{K}$ are: the *Pole placement method* and the *Linear Quadratic Regulator* (LQR).

4.4 Pole placement method

Pole placement method is one of the classic control theories and has an advantage in system control for desired performance. Theoretically pole placement is setting the desired pole location and then moving the poles of the system to that desired pole location in pursuance of getting the desired system response. Mathematically pole placement aims to find a matrix $\mathbf{K}$, for which the eigenvalues of matrix $\mathbf{A} - \mathbf{B}\mathbf{K}$ match the desired poles. However, to

place the poles of the system arbitrary, a necessary and sufficient condition is that the system is completely state controllable. If suppose the system is not completely state controllable, then there are eigenvalues of matrix $\mathbf{A} - \mathbf{BK}$ that can not be controlled by state feedback. Pole placement method becomes simpler with MATLAB command place(A, B, poles), which computes a state-feedback gain matrix $\mathbf{K}$, given the desired closed-loop poles. So, the critical part is selecting the right location to place the poles. Firstly, all the poles should land on the left half of s-plane in order to achieve stability of the continuous-time system. Next, the dominant closed-loop poles (the poles which is near to the imaginary axis) should not be placed too far away from the imaginary axis, as the response obtained will be quicker but at the cost of the system becoming nonlinear, and not too close to it, as the system could turn unstable. Also, in the case of imaginary poles, the bigger the imaginary part of the pole the more oscillations the system faces, so low imaginary parts are preferred. [26]

As presented in chapter (2.5), the poles of the initial system with transfer function (2.16) are $p_1 = 9.816, p_2 = -9.835$ and $p_3 = -0.6475$. The main idea is to move the pole that causes instability (p_1) from the right-hand plane to left-half plane. After that, a number of cases will be examined. Table () contains the the locations of the poles for each case along with their gains. For example in case 1, with poles defined as poles=[-9.835,-9.816,-0.6475] , MATLAB's place command calculates the following gain matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} -200.9 & -20.46 & -0.037 \end{bmatrix}$$

Information about the impulse response of the angle θ and the angular velocity $\dot{\theta}$, for each of the 5 cases, is presented at tables (4.2) and (4.3), respectively. Before starting to further analyzing every case, it is important to mention that in all cases, θ and $\dot{\theta}$ reach stability at zero. In case 1, the positive pole of the initial system has been transferred to the exact same location of the left-half plane ($p_1' = -p_1$). The settling time of θ is very big and may cause instability, and the same goes with the negative max angle. To reduce the settling time and the overshoot of θ the idea was to move the pole close to zero (p_3) more to the left. In case 2, it is moved to -20 and indeed both of these parameters are reduced considerably. To reduce them even more, in case 3 this pole is moved even further to the left (-100) and produces a very small negative max angle and little overshoot.

Case 3 provides very promising results. However, there is still room for improvement as for the settling time. In case 4, all poles are moved further to the left and it is managed to reduce the settling time by 50% from case 3. In case 5, a different approach is tried. The pole

	Case 1		**Case 2**		**Case 3**		**Case 4**		**Case 5**	
	Poles	Gains	Poles	Gains	Poles	Gains	Poles	Gains	Poles	Gains
p_2	-9.835	-200.9	-9.835	-572.1	-9.835	-2106	-60	-3607	-60	-2501
p_1	-9.816	-20.46	-9.816	-58.82	-9.816	-217.4	-30	-486	-40	-114.3
p_3	-0.6475	-0.037	-20	-0.61	-100	-3	-20	-11.1	-0.6475	-0.5

Table 4.1: Pole placement cases and gains of matrix $\mathbf{K}$

near zero remains stable and the other two poles are moved further to the left, reducing the upper oscillation (less positive max angle from all cases) but not managing to lower even more the settling time of case 4.

The priorities of every problem change depending what needs to be done. So, in order to decide which of the cases presented is the more appropriate, the importance of its spec must be classified. Due to the vulnerability of the robot to instability, the two most important specs are decided to be the settling time and the maximum angle. It is desired to reach stability quickly and without facing big angle displacements. After these, the oscillations of the system need to be checked. Having all that in mind, the best of these cases is case 4, as it manages the lowest settling time and the lowest angle overshoot. Also, the only oscillation that θ response faces is considered negligible and the peak time is the fastest of five (along with case 5). As for the angular velocity $\dot{\theta}$, it also manages the lowest settling time.

Angle θ	Case 1	Case 2	Case 3	Case 4	Case 5
Negative max angle (rad)	-0.0366	-0.018	-0.00681	-0.00645	-0.00751
Negative max angle (degrees)	-2	-1	-0.39	-0.36	-0.43
Positive max angle (rad)	0.0046	0.006	0.0013	0.002	0.00023
Positive max angle (degrees)	0.26	0.34	0.074	0.11	0.013
Peak time (sec)	0.1	0.05	0.02	0.02	0.02
Stability at:	0	0	0	0	0
Settling time (sec)	3.65	0.744	0.612	0.294	0.963

Table 4.2: Impulse response information of angle θ: Pole placement

Angular velocity $\dot{\theta}$	Case 1	Case 2	Case 3	Case 4	Case 5
Maximum velocity (rad/sec)	-1.02	-1.02	-1.02	-1.02	-1.02
Peak time (sec)	≈ 0	≈ 0	≈ 0	≈ 0	≈ 0
Stability at:	0	0	0	0	0
Settling time (sec)	0.539	0.245	0.14	0.0917	0.111

Table 4.3: Impulse response information of angular velocity $\dot{\theta}$: Pole placement

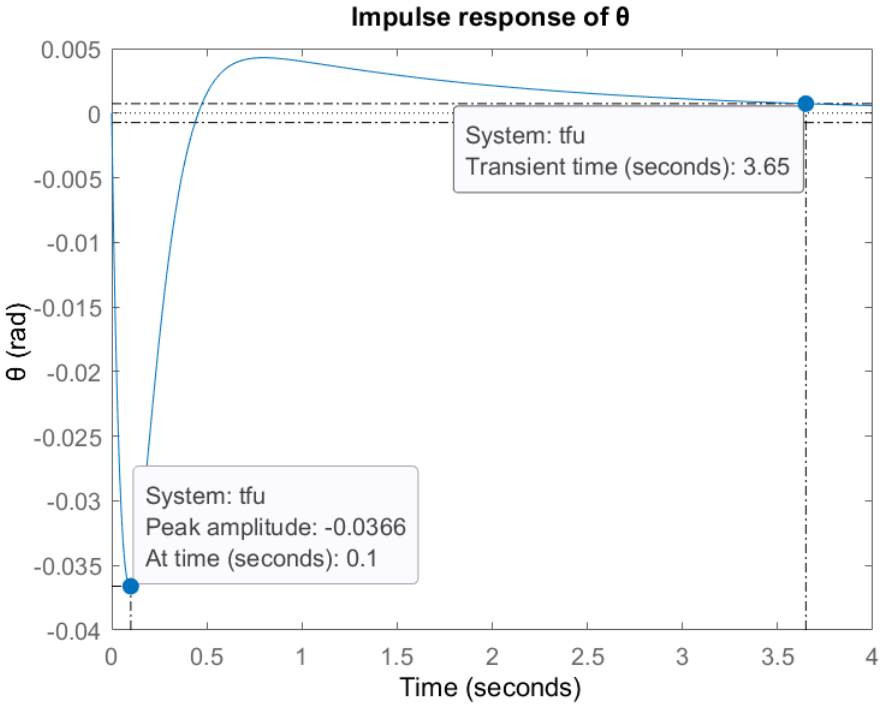

Figure 4.2: Impulse response of θ: Case 1 of Pole placement

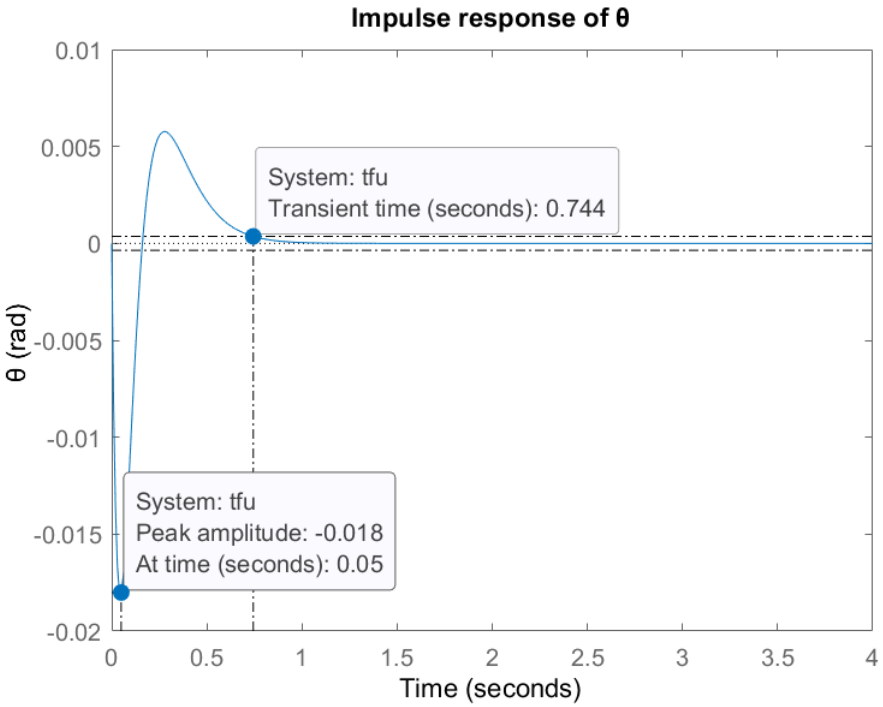

Figure 4.3: Impulse response of θ: Case 2 of Pole placement

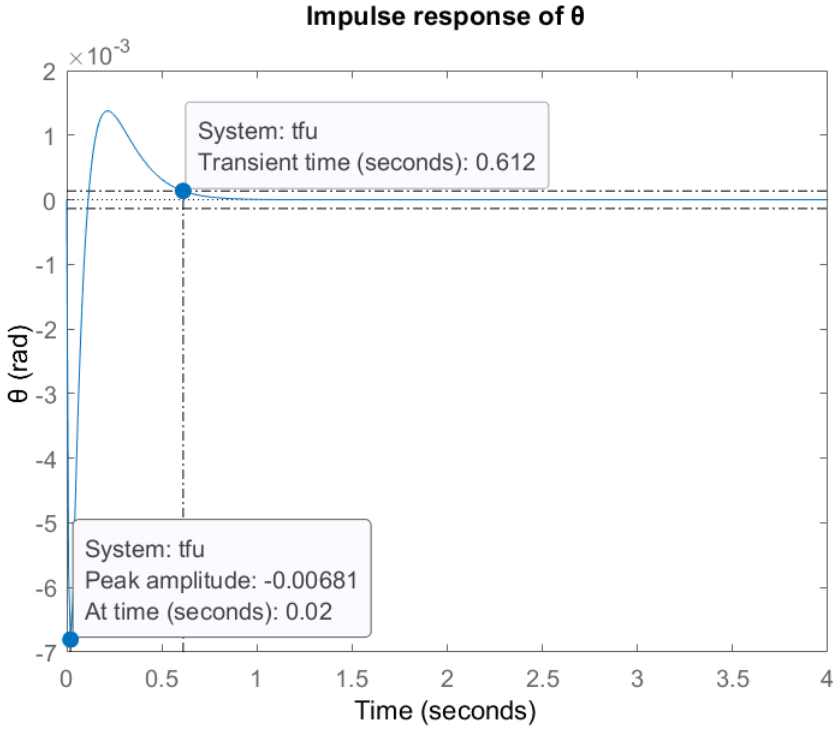

Figure 4.4: Impulse response of θ: Case 3 of Pole placement

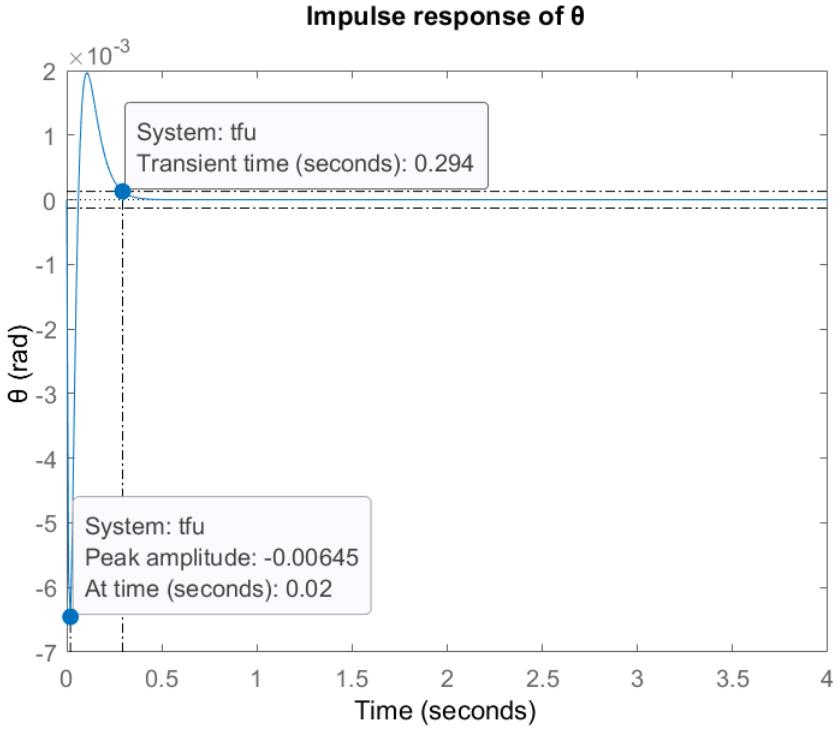

Figure 4.5: Impulse response of θ: Case 4 of Pole placement

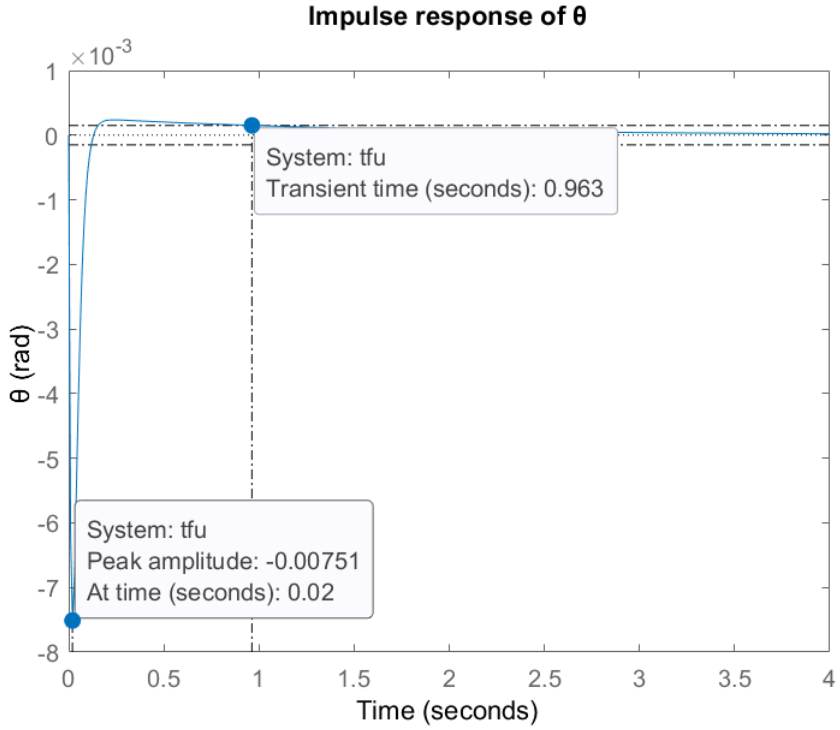

Figure 4.6: Impulse response of θ: Case 5 of Pole placement

4.5 Linear Quadratic Regulator (LQR)

Creating a full state feedback controller with the pole placement method has a major drawback: finding a right location for the closed-loop poles on the complex plain is quite difficult and requires a high level of skills in establishing a relationship between poles and dynamic performances of the closed-loop control system. A solution to the above problem is provided by the *Linear Quadratic Regulator* (LQR), sometimes referred to as *Linear Quadratic Control* (LQC) or *Optimal control*. So, LQR is the optimal theory of pole placement method as it defines the optimal pole location based on a quadratic cost function. [24]

The resulting control law of LQC is optimal, but that does not mean that it is the best that exists. It only means that is the one that minimizes the cost function and if the cost function is altered then the optimal controllers will also change. Moreover, LQC guarantees robustness, stability and works for linear MIMO systems as well for SISO systems. [27]

Considering a linear time-variant system in state-space form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t), \quad \mathbf{x}(0) = \mathbf{x}_0, t \geq 0 \tag{4.10}$$

where, $\mathbf{x}(t) \in \Re^n$ is the state vector, $\mathbf{u}(t) \in \Re^m$ is the control vector, $\mathbf{A}(t) \in \Re^{nxn}$ is the state matrix and $\mathbf{B}(t) \in \Re^{mxm}$ the control matrix. The quadratic cost function that needs to

be minimized:

$$J = \frac{1}{2}\|\mathbf{x}(T)\|_S^2 + \frac{1}{2}\int_0^T \left\{\|\mathbf{x}(T)\|_{Q(t)}^2 + \|\mathbf{u}(T)\|_{R(t)}^2\right\} dt \tag{4.11}$$

where, the final time T is given, $\mathbf{S} \succeq \mathbf{0}$,$\mathbf{Q}(t) \succeq \mathbf{0}$ are symmetric,*positive semi-definite* weighting matrices and $\mathbf{R}(t) \succ \mathbf{0}$ is a symmetric, *positive definite* weighting matrix.$\mathbf{Q}(t) \in \Re^{nxn}$ and $\mathbf{R}(t) \in \Re^{mxm}$ matrices are also time variant but for brevity purposes, the time constant t will not always appear. Also, it is important to mention that the quadratic form is: $T(\mathrm{x}) = \|\mathbf{x}\|_S^2 = \mathbf{x}^T\mathbf{Q}\mathbf{x}$. For an LQC the following assumptions are necessary: [28]

- All the states $\mathbf{x}(t)$ of the system are available for feedback.

- The system (**A**,**B**) is *stabilizable*. This means that all unstable modes are controllable,or in other words, all uncontrollable modes are stable. Stabilizability is a slightly weaker notion than controllability and so, complete controllability implies stabilizability.

- The pair $(\tilde{\mathbf{A}}, \tilde{\mathbf{Q}}) = \left(\mathbf{A} - \mathbf{B}\mathbf{R}^{-1}\mathbf{S}^\top, \mathbf{Q} - \mathbf{S}\mathbf{R}^{-1}\mathbf{S}^\top\right)$ is *detectable*. This means that all unstable modes are observable,or in other words, all unobservable modes are stable. Detectability is a slightly weaker notion than observability and so, observability implies detectability.

Under these assumptions,the goal is to find control law $\mathbf{u}(t) = -\mathbf{K}(t)\mathbf{x}(t)$, where $\mathbf{K}(t) \in \Re^{mxn}$ is the gain matrix, to transfer $\mathbf{x}_0$ to the origin, such that the criterion is minimized. This transfer, however, takes into consideration the cost of control of the second term in equation. Also, the final state $\mathbf{x}(T)$ is free, though the deviations from the origin (**0**) are penalized with weighting matrix $\mathbf{S}$. Matrices $\mathbf{Q}$ and $\mathbf{R}$ are also called design parameters and penalize the states **x**(t) and the control $\mathbf{u}(t)$ respectively, and most of the times are diagonal. Matrix $\mathbf{Q}$ expresses the penalty for the speed at which each of the parameters in the state vector should be reached. For instance, if matrix gives a bigger penalty for one state parameter and smaller for another, it punishes the first state parameter for reaching slower than the second. Matrix $\mathbf{R}$ specifies the penalty for using up control effort $\mathbf{u}(t)$. So, as bigger as the parameters of $\mathbf{R}$ gets, the more expensive the application of the control $\mathbf{u}(t)$ becomes. The most common form of these two matrices is presented below. The diagonal parameters of $\mathbf{Q}$ represent the

weights of every state variable and those of $\mathbf{R}$ the weights of every control variable. [24] [29]

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & \cdots & 0 \\ 0 & q_{22} & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & q_{nn} \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} r_{11} & 0 & \cdots & 0 \\ 0 & r_{22} & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & \cdots & \cdots & r_{mm} \end{bmatrix} \tag{4.12}$$

So, the optimal control law is given from equation (4.13) , where $\mathbf{K}(t)$ is derived from equation (4.14):

$$\mathbf{u}(t) = -\mathbf{K}(t)\mathbf{x}(t) \tag{4.13}$$

$$\mathbf{K}(t) = \mathbf{R}(t)^{-1}\mathbf{B}(t)^{\mathrm{T}}\mathbf{P}(t) \tag{4.14}$$

where, $\mathbf{B}(t)$ is the control matrix and $\mathbf{P}(t) \in \Re^{nxn}$ is the *Riccati matrix*, a symmetric and positive definite matrix that is the solution to the following nonlinear differential equation of matrices, known as the *Riccati equation*:

$$\dot{\mathbf{P}}(t) = -\mathbf{P}(t)\mathbf{A}(t) - \mathbf{A}(t)^{T}\mathbf{P}(t) + \mathbf{P}(t)\mathbf{B}(t)\mathbf{R}(t)^{-1}\mathbf{B}(t)^{\mathrm{T}}\mathbf{P}(t) - \mathbf{Q}(t) \tag{4.15}$$

that has boundary condition:

$$\mathbf{P}(t) = \mathbf{S} \tag{4.16}$$

If $\mathbf{P}(t)$ is obtained for all t, then the state feedback LQR becomes:

$$\mathbf{u}(t) = -\mathbf{R}(t)^{-1}\mathbf{B}(t)^{\mathrm{T}}\mathbf{P}(t)\mathbf{x}(t) = -\mathbf{K}(t)\mathbf{x}(t) \tag{4.17}$$

As observed, both $\mathbf{P}(t)$ and $\mathbf{K}(t)$ depend only on the system's matrices $\mathbf{A}$, $\mathbf{B}$, on the design matrices $\mathbf{Q}$, $\mathbf{R}$,and $\mathbf{S}$, and time T, and not on the state vector $\mathbf{x}(t)$. This means that the calculation of $\mathbf{P}(t)$ and $\mathbf{K}(t)$ through the equations (4.15) and (4.14), can be done off-line (in the lab) while developing the control law, while $\mathbf{u}(t)$ can be calculated on-line if the state vector $\mathbf{x}(t)$ is available for all t from the equation (4.17).

Most practical applications do not require a time-variant solution and so, a time-invariant is preferred. To obtain such a solution the following assumption must be made:

- The problem's matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$ and $\mathbf{R}$ must be time-invariant.
- The time horizon is infinite, i.e. $T \to \infty$.
- The system [$\mathbf{A}$,$\mathbf{B}$] is controllable. (4.2)

- The system [$\mathbf{A}$,$\mathbf{C}$] is observable (4.4), where $\mathbf{C}$ is any matrix such that $\mathbf{C}^T\mathbf{C} = \mathbf{Q}$.

Under the above assumptions, it is proven that the integration of the Riccati matrix $\mathbf{P}(t)$ starting from any terminal condition $\mathbf{P}(t) = \mathbf{S} \succeq \mathbf{0}$, converges towards a unique stationary value $\bar{\mathbf{P}}$. So, the calculation of matrix $\mathbf{P}(t)$ can be done through equation (4.15), while substituting $\dot{\mathbf{P}}(t) = 0$. The new equation that calculates the Riccati matrix is called *algebraic Riccati equation* and is presented below:

$$-\mathbf{PA} - \mathbf{A}^T\mathbf{P} + \mathbf{PBR}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P} - \mathbf{Q} = \mathbf{0} \tag{4.18}$$

The Riccati matrix $\mathbf{P}$ of equation (4.18) guarantees that the system will have stability and gives the ability to calculate gain matrix $\mathbf{K}$, which under the new circumstances is considered time-invariable.

$$\mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{4.19}$$

Obtaining $\mathbf{K}$ provides the invariant control law of the system:

$$\mathbf{u}(t) = -\mathbf{R}^{-1}\mathbf{B}^T\bar{\mathbf{P}}\mathbf{x}(t) = -\mathbf{K}\mathbf{x}(t) \tag{4.20}$$

The above problems are called *infinite horizon problems*. The self-balancing motorcycle problem can be included in this category. Matrix $\mathbf{Q}$ will be a 3 x 3 matrix and $\mathbf{R}$ will be scalar and equal to $r > 0$. However, finding the components of these two matrices in LQR is a difficult stage in control design as there is no general methodology. First of all, the form of matrix $\mathbf{Q} = \mathbf{C}^T\mathbf{C}$ will be:

$$\mathbf{Q} = \mathbf{C}^T\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.21}$$

and all the components of $\mathbf{Q}$ that are equal to one will be replaced with non-zero variables. In this problem, matrix $\mathbf{Q}$ has ones at the positions (1,1), (2,2) and (3,3). These values will be replaced with the variables q_{11}, q_{22} and q_{33} that represent the weights of the motorcycle's lean angle θ, the angular velocity of the rod $\dot{\theta}$ and the inertia's wheels velocity ω, respectively. So, eventually the matrices $\mathbf{Q}$ and $\mathbf{R}$ will be as follows:

$$\mathbf{Q} = \begin{bmatrix} q_{11} & 0 & 0 \\ 0 & q_{22} & 0 \\ 0 & 0 & q_{33} \end{bmatrix}, \quad \mathbf{R} = r \tag{4.22}$$

But what will the values of q_{11}, q_{22} and q_{33} be? In general, there is no specific rule to follow. A simple way is holding stable matrix $\mathbf{Q}$ in (4.21) and changing the value of $\mathbf{R} = r\mathbf{I}, r >)$, reducing it per ten, starting from one (e.g. from 1 to 0.1, from 0.1 to 0.01 etc.). Also, another simple and reasonable choice for the matrices $\mathbf{Q}$ and $\mathbf{R}$ is given by Bryson's rule. According to it, the diagonal components of these matrices are equal to the inverse of the absolute maximum acceptable values of the state variables and the control variables respectively, to the power of two. In mathematical terms the above interpret as:

$$q_{ii} = \frac{1}{|\mathrm{x}_{i,\max}|^2}, \quad i = 1, 2, \ldots, n.$$
$$r_{jj} = \frac{1}{|\mathrm{u}_{i,\max}|^2}, \quad j = 1, 2, \ldots, m.$$

With the first method, where r is reduced, considering $\mathbf{Q}$ of (4.21) and $r = 0.01$. MATLAB's command lqr(A,B,Q,R) , where matrices $\mathbf{A}$ and $\mathbf{B}$ are from (4.5), computes gain matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} -6758.2 & -698 & -10 \end{bmatrix}, \tag{4.23}$$

with the eigenvalues of $\mathbf{A} - \mathbf{BK}$ to be the closed-loop poles: $p_1 = -347$, $p_2 = -9.67 + 0.14i$ and $p_3 = -9.67 - 0.14i$. The impulse response of θ for this case, is presented in (4.7).

With Bryson's rule, the maximum acceptable values of the state and control variables must be defined. First, the angle of the rod θ is preferred not to exceed 6° or 0.1 rad and so $\theta_{max} = 0.1$rad:

$$q_{11} = \frac{1}{|\theta_{max}|^2} = 100 \tag{4.24}$$

Next, the maximum value of the angular velocity $\dot{\theta}$ is set to 2 rad/sec and the maximum value of rotational speed ω to 300 rad/sec:

$$q_{22} = \frac{1}{\left|\dot{\theta}_{max}\right|^2} = 0.25 \tag{4.25}$$

$$q_{33} = \frac{1}{|\omega_{max}|^2} = 10^{-5} \tag{4.26}$$

Regarding the maximum Volts that the DC motor can produce, a value of 12V is given from the manufacturer:

$$r = \frac{1}{|V_{mmax}|^2} = 0.0069 \tag{4.27}$$

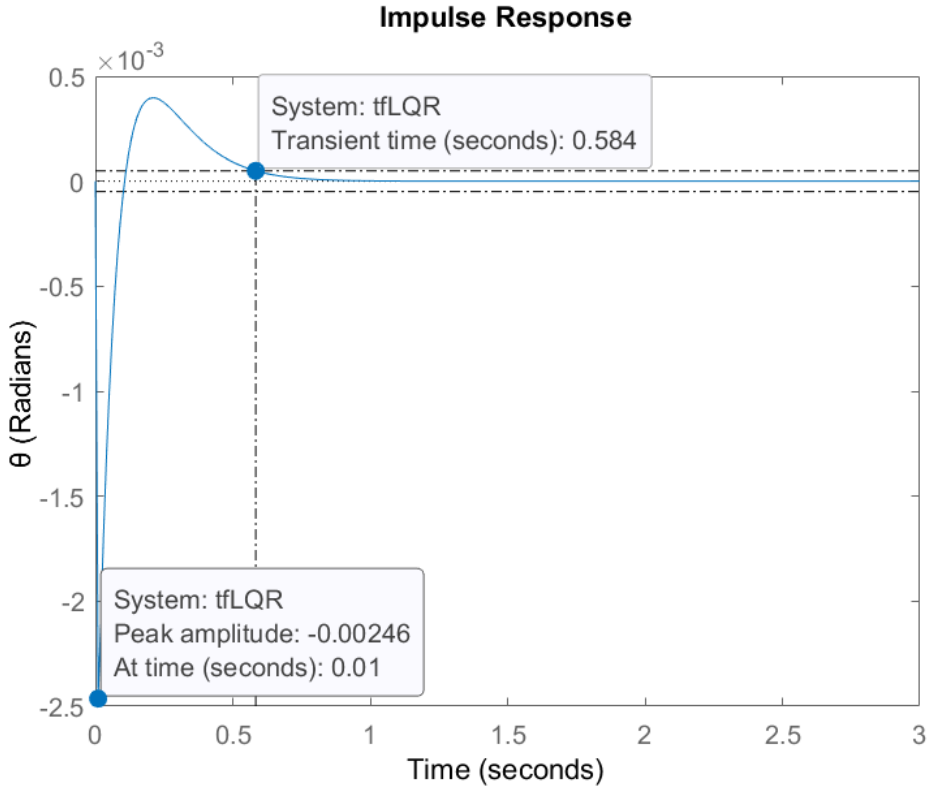

Figure 4.7: Impulse response of θ, with LQR control and reducing r method ($r = 0.01$)

Replacing these variables, matrices $\mathbf{Q}$ and $\mathbf{R}$ become:

$$\mathbf{Q} = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 10^{-5} \end{bmatrix}, \quad \mathbf{R} = 0.0069 \tag{4.28}$$

MATLAB's command lqr(A,B,Q,R) , computes gain matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{bmatrix} -267.93 & -25.13 & -0.0612 \end{bmatrix}, \tag{4.29}$$

with the eigenvalues of $\mathbf{A} - \mathbf{BK}$ to be the closed-loop poles: $p_1 = -11 + 4.53i$, $p_2 = -11 - 4.53i$ and $p_3 = -0.88$. The impulse response of θ for this case, is presented in (4.8).

To get an even better response, r is reduced to 0.000069, while matrix $\mathbf{Q}$ remains the same. The gain matrix $\mathbf{K}$ is:

$$\mathbf{K} = \begin{bmatrix} -1391.8 & -93.5 & -0.4 \end{bmatrix}, \tag{4.30}$$

with the eigenvalues of $\mathbf{A} - \mathbf{BK}$ to be the closed-loop poles: $p_1 = -61.2$, $p_2 = -20.3$ and $p_3 = -0.99$. The impulse response of θ for this case, is presented in (4.9).

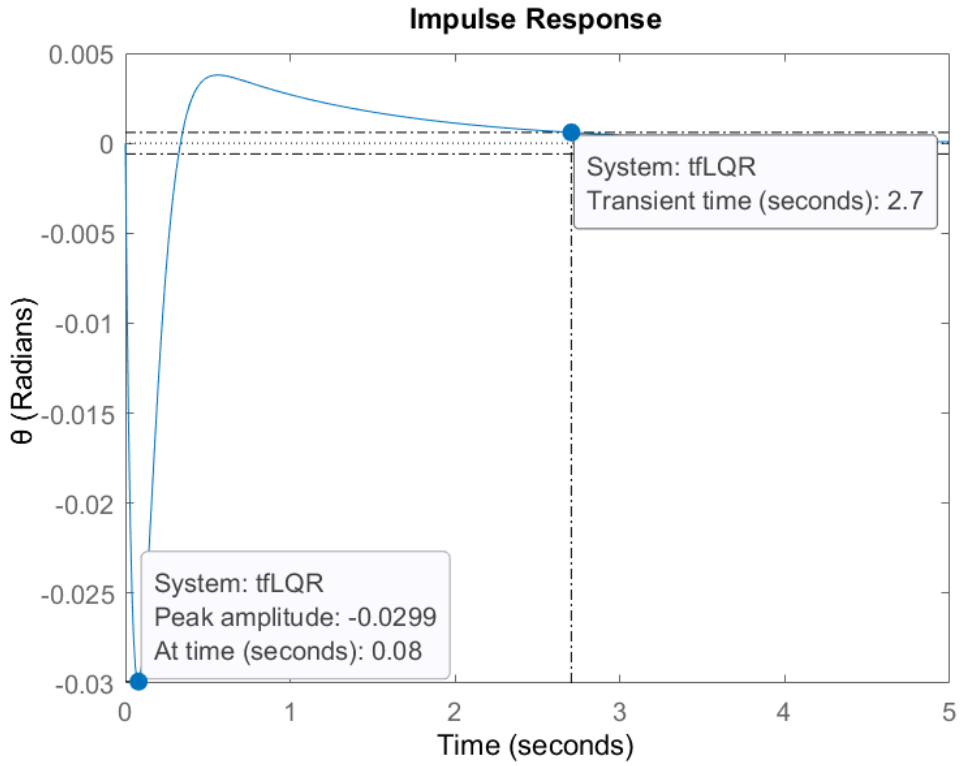

Figure 4.8: Impulse response of θ, with LQR control and Bryson's method ($r = 0.0069$)

Indeed, reducing r in Bryson's method does offer better settling time and peak amplitude. However, it is not advisable to reduce it even more because that will mean that the control would be very free, causing problems to the system. So, the best case for LQR control is the first one, where $\mathbf{Q}$ is the identity matrix and $r = 0.01$, as it provides the best behavior. In the first case, it is worth mentioning that the poles are far from zero, in contrast to the other two cases where one of the poles is almost zero. So, it is fair to say that in the self-balancing motorcycle's model, a pole close to zero does more harm than good.

4.6 Luenberger state observer

In order to apply the full state feedback control, that was mentioned in the previous chapters, to a system, all of its state space variables must be available at all times. Moreover, in some control system applications, information about the system's state space variables is always needed. However, most of the times these variables are not available for measurement, or it is impractical to measure all of them, or the cost to measure them is very big. In these instances, the unavailable variables are being estimated using a *state observer* or *state estima-*

Angle θ	Q identity r=0.01 Case 1	Bryson method r=0.0069 Case 2	Bryson method r=0.000069 Case 3
Negative max angle (rad)	-0.00246	-0.0299	-0.00939
Negative max angle (degrees)	-0.14	-1.65	-0.53
Positive max angle (rad)	0.0004	0.0035	0.0005
Positive max angle (degrees)	0.022	0.2	0.028
Peak time (sec)	0.01	0.08	0.03
Stability at:	0	0	0
Settling time (sec)	0.584	2.7	1.54

Table 4.4: Impulse response information of angle θ: LQR control

Angle θ	Q identity r=0.01 Case 1	Bryson method r=0.0069 Case 2	Bryson method r=0.000069 Case 3
Maximum velocity (rad/sec)	-1.02	-1.02	-1.02
Peak time (sec)	≈ 0	≈ 0	≈ 0
Stability at:	0	0	0
Settling time (sec)	0.069	0.412	0.158

Table 4.5: Impulse response information of angular velocity $\dot{\theta}$: LQR control

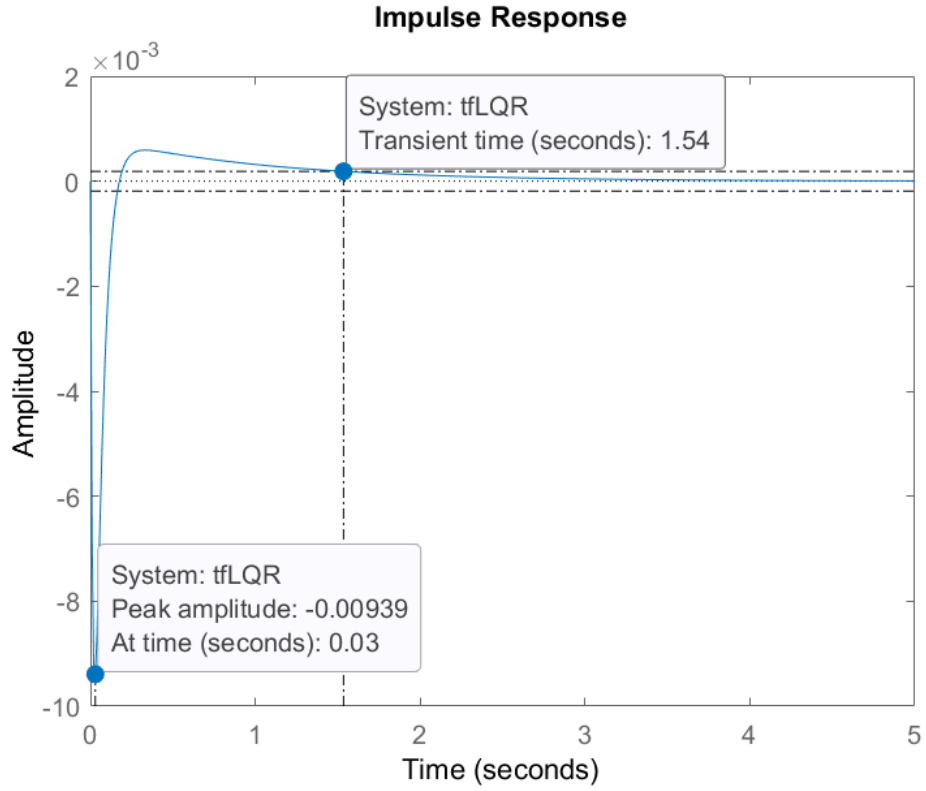

Figure 4.9: Impulse response of θ, with LQR control and Bryson's method ($r = 0.000069$)

tor, that is also known as the *Luenberger state observer*. The observer provides an estimate of the internal state of a given system, from measurements of the input and output of the system. If the role of an observer is to provide estimates of all state variables, then it is referred to as a *full-order state observer*. Sometimes it is possible to obtain satisfactory measurement of some, but not all, states. In this situation, a *reduced-order state observer* may be used to estimate only the non-accessible states. [30] [31]

4.6.1 Full-order state observer

A full-order state observer estimates all the state variables of a system, even if they are not measurable. So, a full-order observer design produces an observer that has the same dimension as the original system. For the initial system of the self-balancing robot that is presented in equations (2.14) and (2.15), the full-order state observer can be described by the following equations: [31]

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{L}[\boldsymbol{y}(t) - \mathbf{C}\hat{\mathbf{x}}(t)] \tag{4.31}$$

$$\hat{\mathbf{y}}(t) = \mathbf{C}\hat{\mathbf{x}}(t), \tag{4.32}$$

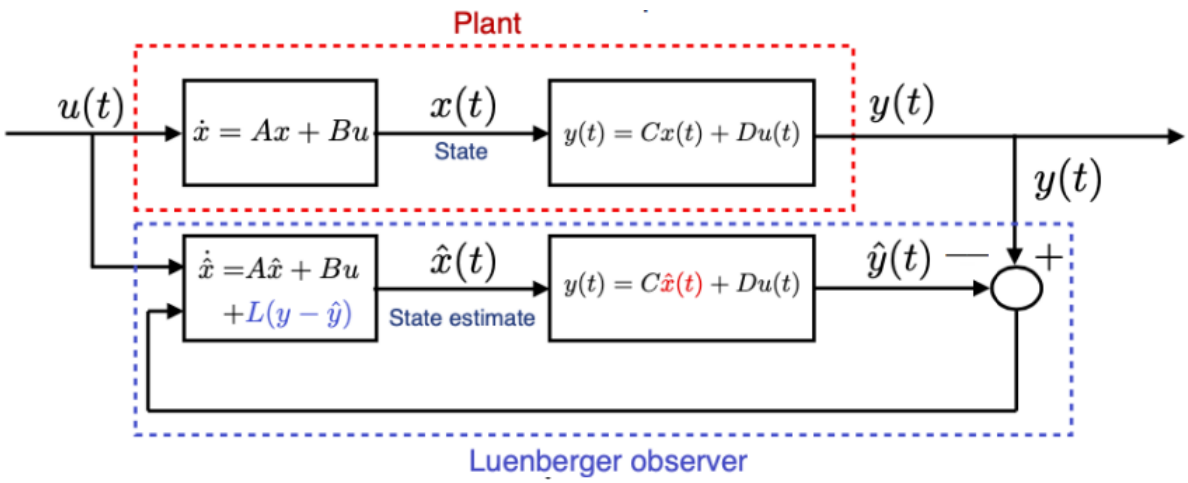

Figure 4.10: Luenberger observer block diagram

where, $\hat{\mathbf{x}} \in \Re^{nx1}$ is the estimate of the real state vector $\mathbf{x}$, $\hat{\mathbf{y}}$ the measured output, and $\mathbf{L} \in \Re^{mxn}$ is the gain matrix of the observer. The two outputs $\mathbf{y}(t)$ and $\hat{\mathbf{y}}(t)$ will be different since in the first case the system initial condition is considered unknown, and in the second case it has been chosen arbitrarily. So, the difference between these two outputs will generate an error signal $\mathbf{e}(t)$:

$$\boldsymbol{y}(t) - \hat{\mathbf{y}}(t) = \mathbf{C}\mathbf{x}(t) - \mathbf{C}\hat{\mathbf{x}}(t) = \mathrm{Ce}(t) \tag{4.33}$$

where,

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{4.34}$$

The goal of the observer is to produce an estimate $\hat{\mathbf{x}}$, that will try to reach the real state $\mathbf{x}$. In other words, to try and reduce as much as possible, hopefully to zero (at least at steady-state), the error signal $\mathbf{e}(t)$. It is easy and simple to derive an expression for dynamics of the observation error as:

$$\dot{\mathbf{e}}(t) = \dot{\mathbf{x}}(t) - \dot{\hat{\mathbf{x}}}(t) = (\mathbf{A} - \mathbf{LC})\mathbf{e}(t) \tag{4.35}$$

If the gain matrix $\mathbf{L}$ is chosen so that the eigenvalues of $\mathbf{A} - \mathbf{LC}$ matrix are strictly in the left-half of the complex plane, then the error equation is asymptotically stable, and the estimation error will decay to zero over time. In order to place the eigenvalues of $\mathbf{A} - \mathbf{LC}$ matrix at arbitrary locations, the system $(\mathbf{A}, \mathbf{C})$ needs to be observable. More precisely, by taking the transpose of the estimation error feedback matrix, i.e. $\mathbf{A}^T - \mathbf{C}^T\mathbf{K}^T$, it is noticed that if the pair $(\mathbf{A}^T, \mathbf{C}^T)$ is controllable, then the poles can be placed in arbitrarily asymptotically stable positions. Consequently, the controllability of the pair $(\mathbf{A}^T, \mathbf{C}^T)$ is equal to the observability of the pair $(\mathbf{A}, \mathbf{C})$. After all, the design of a full-order state observer is based on

finding matrix $\mathbf{L}$, a process that becomes really simple and fast with MATLAB's command place(A', C', poles)', where poles are the poles of the observer that can be placed arbitrarily. It should be noted that, if place was called without the transpose sign (') at the end, then matrix $\mathbf{L}$ would have dimensions nxm, instead of mxn, making the multiplication of matrix $\mathbf{L}$ and $\mathbf{C}$ impossible. The command place(...) is the exact same that was used in the pole placement method.

4.7 State feedback compensator

The state feedback compensator is obtained by connecting the full-state feedback law to the state observer. The design begins with a full state feedback control law $\mathbf{u}(t) = -\mathbf{K}\mathbf{x}(t)$ and continues with a state observer capable of providing an estimation $\hat{\mathbf{x}}$ of the state vector. To connect the control with the observer, the estimation $\hat{\mathbf{x}}$ is applied to the control law instead of the real state vector $\mathbf{x}$. So, the control turns into: [32]

$$\mathbf{u}(t) = -\mathbf{K}\hat{\mathbf{x}}(t) \tag{4.36}$$

However, it is important to guarantee that this alternation of the control will not affect the stability of the closed-loop system. Equation (4.36) is being applied to equation (4.31), producing the compensator system:

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{B}\mathbf{K} - \mathbf{L}\mathbf{C})\hat{\mathbf{x}}(t) + \mathbf{L}y(t). \tag{4.37}$$

From equation (4.34), $\hat{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{e}(t)$. Implementing this relation and equation (4.36) into the initial state space model:

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{B}\mathbf{K})\mathbf{x}(t) + \mathbf{B}\mathbf{K}\mathbf{e}(t) \tag{4.38}$$

Rewriting equation (4.38) in matrix form gives:

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{e}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A} - \mathbf{B}\mathbf{K} & \mathbf{B}\mathbf{K} \\ 0 & \mathbf{A} - \mathbf{L}\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{e}(t) \end{bmatrix} = \mathbf{A}_{cl} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{e}(t) \end{bmatrix} \tag{4.39}$$

As noticed, matrix $\mathbf{A}_{cl}$ is upper triangular, which means that the system's eigenvalues depend only on the diagonal components. As a result, in order to reach stability for the closed loop compensator system, the eigenvalues of matrices $\mathbf{A} - \mathbf{B}\mathbf{K}$ and $\mathbf{A} - \mathbf{L}\mathbf{C}$ must be placed on the left half side of the complex plane. However, from the design of the controller, matrix

$\mathbf{A} - \mathbf{BK}$ has already guaranteed stability (eigenvalues are placed on the left half side of the complex plane) and from the design of the estimator, matrix $\mathbf{A} - \mathbf{LC}$ has also already guaranteed stability. Consequently, to ensure stability for the closed-loop compensator system, it is enough just to ensure the stability of the systems of the controller and the observer, separately. This is called the *separation principle* and states that under some assumption ($\mathbf{A} - \mathbf{BK}$ and $\mathbf{A} - \mathbf{LC}$ have eigenvalues on the left half side), the design of a compensator can be separated into a state feedback control and an observer design. In fact, the pole locations for the controller and the observer will also be the closed-loop pole locations. [33]

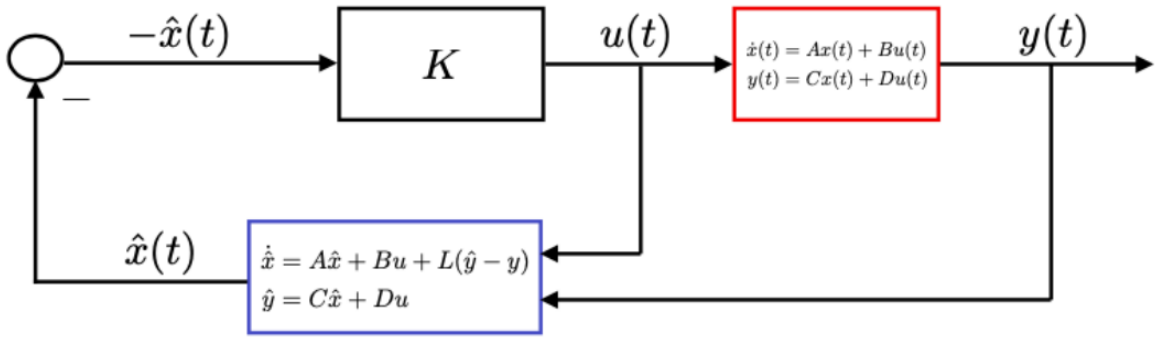

Figure 4.11: State feedback compensator block diagram

Following the are some design guidelines for a state feedback compensator. Firstly, the controller is being designed and a $\mathbf{K}$ matrix is calculated to place the poles of $\mathbf{A} - \mathbf{BK}$ wherever it is desired on the LHP. Next, the same process is being followed for the estimator, except matrices are now $\mathbf{L}$ and $\mathbf{A} - \mathbf{LC}$. An efficient and common option is to design an observer that is 5 to 10 times faster than the controller. This can happen by placing the poles of the estimator 5 to 10 times more left to the LHP than those of the controller. After finding matrices $\mathbf{K}$ and $\mathbf{L}$ the dynamics of the compensator are given by:

$$\mathbf{A}_c \triangleq \mathbf{A} - \mathbf{BK} - \mathbf{LC}, \quad \mathbf{B}_c \triangleq \mathbf{L}, \quad \mathbf{C}_c \triangleq \mathbf{K} \tag{4.40}$$

It must be said that a fast observer (i.e. poles far in LHP) typically implies large error magnitudes in the transient and that high observer gains magnify measurement noise (excessively aggressive Luenberger observer is noisy). On the other hand, if an observer is slow, then the controller becomes less “strong” against disturbances. Also, the observer should be slow when the sensors face a lot of noise. The majority of times, an estimation, even for the sensor-measured variables, takes place for the purpose of reducing the noise coming from the sensors. So, the full-order state observers are more commonly used instead of the reduced-

order state observers. In case the model is not perfect or corrupted by noise, then another optimal observer, called Kalman filter, is used. Kalman filter will be later described in detail.

4.7.1 Pole placement and Luenberger observer

In this section, the full-state feedback control law designed with the pole placement method will be combined with a Luenberger observer. The cases of the pole placement method that will be examined, are cases 1, 3 and 4. For the Luenberger observer there will be 3 distinct cases, as the poles of each case will be moved five, eight and ten times to left. Meaning that the observer will be five, eight and ten times faster than the controller, respectively.

Regarding case 4, the poles of the controller are: $p_1 = -30$, $p_2 = -60$ and $p_3 = -20$. The locations of the observer's poles are presented in table (4.6), along with the gains of matrix $\mathbf{L}$. The impulse response's information of the closed-loop compensator system for θ is presented in table (4.7). As observed, the closed-loop system does stabilise, however, the response is unpractical for the robot's problem. The maximum acceptable overshoot was set to 6°, when the angle overshoot is way more than this at all observer cases for case 4 of pole placement. Moreover, there is no point of making the observer more fast, because table (4.7) clearly shows that the faster the observer the larger the maximum angle. Even making the observer 2 times faster does not provide better results (max positive angle: 105°) and as a result this compensator design is considered impractical and must be avoided.

As for case 1, the poles of the controller are: $p_1 = -9.816$, $p_2 = -9.835$ and $p_3 = -0.6475$. The locations of the observer's poles are presented in table (4.8), along with the gains of matrix $\mathbf{L}$, and the impulse response's information of the closed-loop compensator system for θ in table (4.9). It is observed that when the observer is 10 faster that the controller, the minimum negative overshoot is achieved, while the settling times remains almost the same, making it the best of these 3 compensators examined. Impulses responses of θ for different Luenberger observers can been seen at figure (4.12).

Regarding the last case examined (case 3 of pole placement), the poles of the controller are: $p_1 = -9.816$, $p_2 = -9.835$ and $p_3 = -100$. The locations of the observer's poles are presented in table (4.10), along with the gains of matrix $\mathbf{L}$. The impulse response's information of the closed-loop compensator system for θ is presented in table (4.11). As observed, both positive and negative maximum angles are bigger than the angle limit that was set (6°)

	Observer 5 times faster		**Observer 8 times faster**		**Observer 10 times faster**	
	Poles	Gains	Poles	Gains	Poles	Gains
p_2	-300	549	-480	879.3	-600	1099
p_1	-150	89730	-240	229910	-300	359360
p_3	-100	228880000	-160	942200000	-200	1843300000

Table 4.6: Luenberger observer poles and gains of matrix **L**, for case 4 of pole location.

in all cases. Just like in the first compensator designed (for case 4 of pole placement), the observer is set 2 times faster than the controller. Again the maximum angles obtained are not acceptable ($-8.7°$ and $21.8°$). As a last option, the observer was made equal fast as the controller. The results was the best achieved for case 3 of pole placement, but still not good enough (max angles: $-5.16°$ and $12°$).

So what is the final verdict? It is taken for granted that the compensators designed for pole placement methods 3 and 4, can not manage to produce promising results and all of them will make the robot fall, regardless how fast the observer will be over the controller. In these two cases, the faster the observer is, compared to the controller, the bigger the angle overshoot of the response and so there is no point trying to design faster of observers than the ones already designed in tables (4.7) and (4.11). On the other hand, the compensator designed for case 1 of pole placement gives better results. More specifically, among the observers examined, the observer that is 10 times faster than the controller achieves the smallest angle overshoot, with small difference in settling time. So, from all cases presented, the best is considered the compensator of case 1 of pole location, with a Luenberger observer that is 10 times faster than the controller.

Angle θ	Observer 5 times faster faster than controller	Observer 8 times faster faster than controller	Observer 10 times faster faster than controller
Neg. max angle (rad)	-1.06	-1.11	-1.12
Neg. max angle (degrees)	-60	-63	-64
Pos. max angle (rad)	2.95	3.37	3.49
Pos. max angle (degrees)	160	190	199
Peak time (sec)	0.04	0.03	0.03
Stability at:	0	0	0
Settling time (sec)	0.323	0.313	0.308

Table 4.7: Impulse response information of closed-loop system for angle θ: Pole placement case 4 with different Luenberger observers.

	Observer 5 times faster		**Observer 8 times faster**		**Observer 10 times faster**	
	Poles	Gains	Poles	Gains	Poles	Gains
p_2	-49.17	100.8	-78.68	161.7	-98.35	202.3
p_1	-49.08	2759	-78.52	6980	-98.16	10888
p_3	-3.23	310130	-5.18	1412800	-6.47	2851000

Table 4.8: Luenberger observer poles and gains of matrix **L**, for case 1 of pole location.

Angle θ	Observer 5 times faster faster than controller	Observer 8 times faster faster than controller	Observer 10 times faster faster than controller
Neg. max angle (rad)	-0.054	-0.0409	-0.0356
Neg. max angle (degrees)	-3	-2.34	-2.03
Pos. max angle (rad)	0.0335	0.0334	0.0337
Pos. max angle (degrees)	1.91	1.91	1.93
Peak time (sec)	0.09	0.07	0.06
Stability at:	0	0	0
Settling time (sec)	3.94	3.99	4.09

Table 4.9: Impulse response information for closed-loop system of angle θ: Pole placement case 1 with different Luenberger observers.

	Observer 5 times faster		**Observer 8 times faster**		**Observer 10 times faster**	
	Poles	Gains	Poles	Gains	Poles	Gains
p_2	-49.17	597.5	-78.68	956.5	-98.35	1195.8
p_1	-49.08	51236	-78.52	131400	-98.16	205460
p_3	-500	60439000	-800	250250000	-1000	490570000

Table 4.10: Luenberger observer poles and gains of matrix **L**, for case 3 of pole location.

4.7.2 LQR and Luenberger observer

In this section, the full-state feedback control law designed with the LQR will be combined with a Luenberger observer. For case 1 and 3 of the LQR control, observers with different pole locations will be designed. The observers will be five, eight and ten times faster than the controller, meaning that its poles will be five, eight and ten times more to the left from the controller's poles, respectively.

In case 1 of the LQR control, the controller's pole are: $p_1 = -347$, $p_2 = -9.67 + 0.14i$ and $p_3 = -9.67 - 0.14i$. The locations of the observer's poles are presented in table (4.12), along with the gains of matrix **L**. The impulse response's information of the closed-loop compensator system for θ is presented in table (4.13). The same situation as in the compensator

Angle θ	Observer 5 times faster faster than controller	Observer 8 times faster faster than controller	Observer 10 times faster faster than controller
Neg. max angle (rad)	-0.203	-0.208	-0.211
Neg. max angle (degrees)	-11.6	-11.9	-12
Pos. max angle (rad)	0.662	0.801	0.853
Pos. max angle (degrees)	37.9	45.3	48.8
Peak time (sec)	0.07	0.05	0.05
Stability at:	0	0	0
Settling time (sec)	0.718	0.676	0.662

Table 4.11: Impulse response information for closed-loop system of angle θ: Pole placement case 3 with different Luenberger observers.

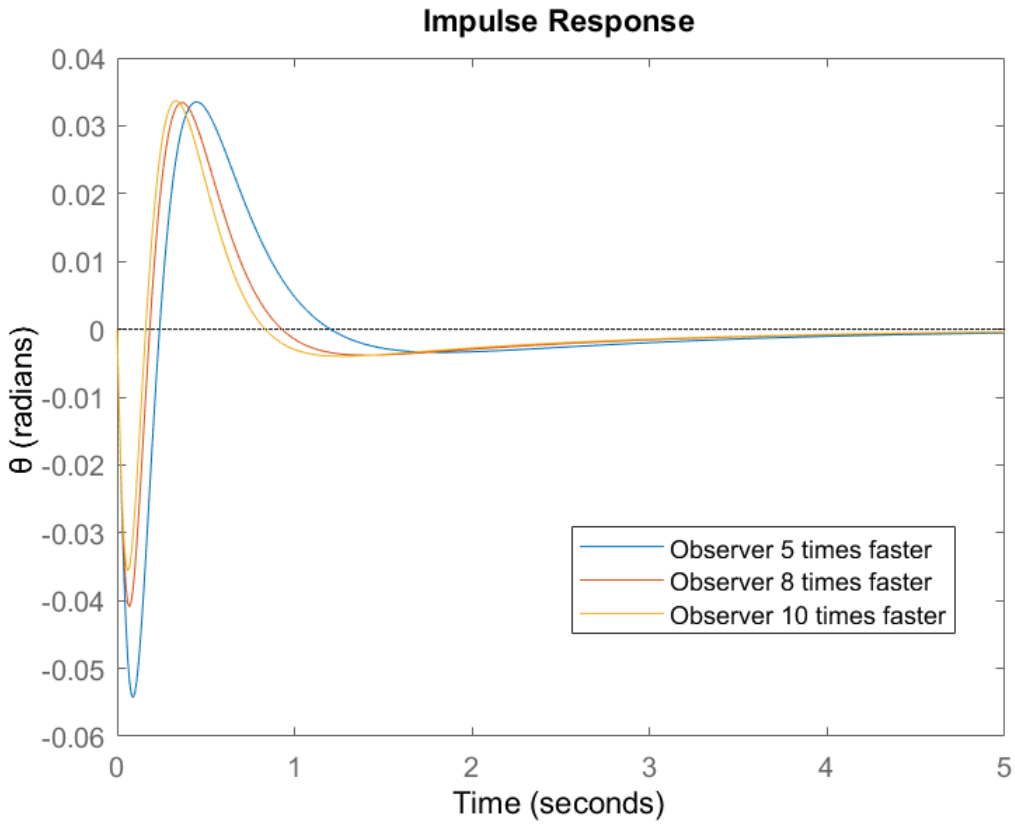

Figure 4.12: Impulse response of θ: Pole placement case 1 and different Luenberger observers.

	Observer 5 times faster		Observer 8 times faster		Observer 10 times faster	
	Poles	Gains	Poles	Gains	Poles	Gains
p_1	-1735	1831	-2776	2930	-3470	3662
p_2	-48.35+0.7i	168990	-77.36+1.12i	433630	-96.7+1.4i	678100
p_3	-48.35-0.7i	203310000	-77.36-1.12i	841620000	-96.7-1.4i	1649600000

Table 4.12: Luenberger observer poles and gains of matrix **L**, for case 1 of LQR.

Angle θ	**Observer 5 times faster faster than controller**	**Observer 8 times faster faster than controller**	**Observer 10 times faster faster than controller**
Neg. max angle (rad)	-0.2	-0.2	-0.2
Neg. max angle (degrees)	-11	-11	-11
Pos. max angle (rad)	0.66	0.82	0.89
Pos. max angle (degrees)	37.8	46.9	50.9
Peak time (sec)	0.06	0.04	0.04
Stability at:	0	0	0
Settling time (sec)	0.715	0.67	0.654

Table 4.13: Impulse response information of angle θ: LQR case 1 with different Luenberger observers.

with pole placement cases 3 and 4 occurs. In all the observers examined, the maximum angle is bigger than the upper limit of 6° that was set in order to prevent instability of the robot. Also, it should be noted that there is no point of making the observer even faster, because table (4.13) clearly shows that the faster the observer the larger the maximum angle. It was also tried to make the observer 2 times faster than the controller, however, not even that could prevent the system from reducing the angle overshoot below the acceptable limit (max angles: 21.4° and -8.5°). So, a compensator with this LQR controller is dangerous and must not be used.

Regarding case 3 of the LQR control, the controller's poles are: $p_1 = -61.2$, $p_2 = -20.3$ and $p_3 = -0.99$. The locations of the observer's poles are presented in table (4.14), along with the gains of matrix **L**. The impulse response's information of the closed-loop compensator

	Observer 5 times faster		Observer 8 times faster		Observer 10 times faster	
	Poles	Gains	Poles	Gains	Poles	Gains
p_1	-306	411	-489.6	659	-612	824
p_2	-101	32743	-162.4	84332	-203	131850
p_3	-4.95	6763300	-7.92	29564000	-9.9	58869000

Table 4.14: Luenberger observer poles and gains of matrix **L**, for case 3 of LQR.

Angle θ	**Observer 5 times faster than controller**	**Observer 8 times faster than controller**	**Observer 10 times faster than controller**
Neg. max angle (rad)	-0.011	-0.0069	-0.0059
Neg. max angle (degrees)	-0.6	-0.39	-0.33
Pos. max angle (rad)	0.048	0.065	0.075
Pos. max angle (degrees)	2.75	3.7	4.2
Peak time (sec)	0.13	0.11	0.1
Stability at:	0	0	0
Settling time (sec)	3.16	2.74	2.57

Table 4.15: Impulse response information of angle θ: LQR case 3 with different Luenberger observers.

system for θ is presented in table (4.15). As observed, all cases provide good results, with reasonable angle overshoot and settling time. The best of 3 is considered that with the 5 times faster observer, because it achieves the lowest angle overshoot and a reasonable settling time. Impulses responses of θ for case 3 of LQR and different Luenberger observers can been seen at figure (4.13).

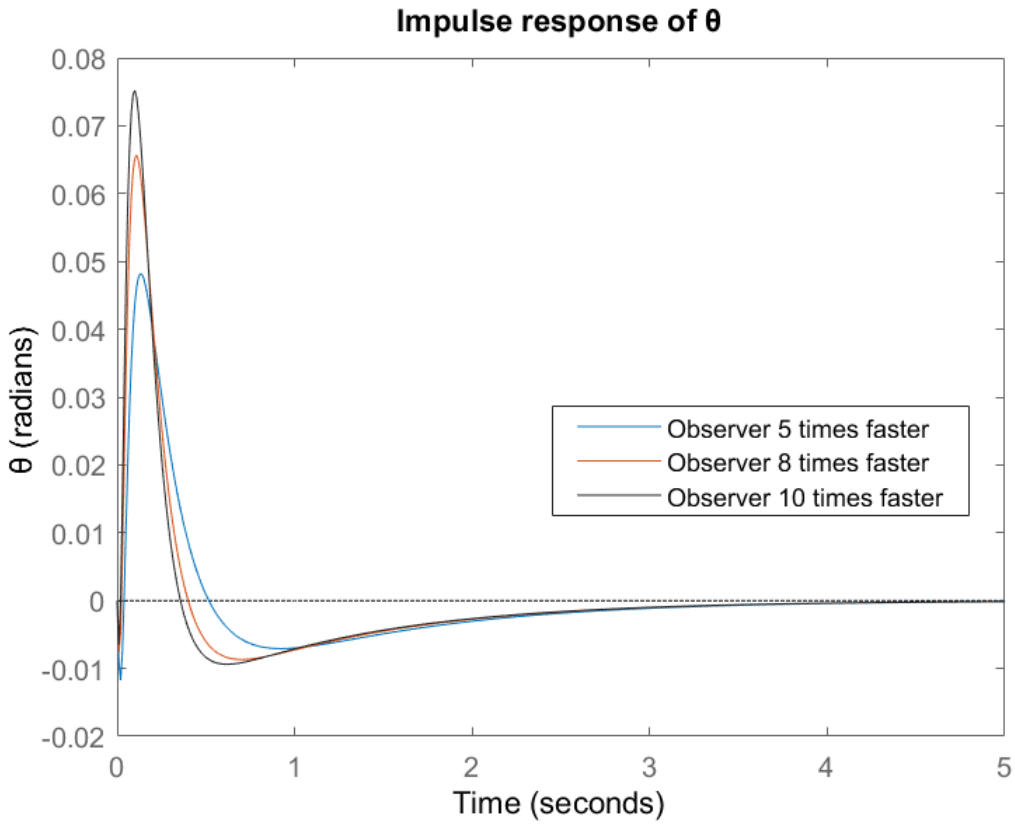

Figure 4.13: Impulse response of θ: LQR case 3 and different Luenberger observers.

Chapter 5

Linear-Quadratic-Gaussian control

5.1 Introduction

The *Linear-Quadratic-Gaussian* (LQG) control problem is one of the most important optimal control problems and has plenty of applications in the modern world. It combines both concepts of Linear-Quadratic-Regulators (LQR) for full state feedback and *Linear-Quadratic-Estimator* (LQE), which is also known as the *Kalman Filter*, for state estimation. The goal is to determine an output feedback law that is optimal for minimizing the value of a quadratic cost criterion. However, the measurements of the outputs can now be disturbed by Gaussian noise, and here lies the difference in the LQR, as this control problem takes into account process disturbances and measurement noise. Gaussian noise, name after *Carl Friedrich Gauss*, denotes a signal of noise that has a *probability density function* equal to that of the *normal distribution* (also known as *Gaussian distribution*) [34]

One of the most fundamental principles of feedback theory is that the problems of optimal control and state estimation can be decoupled in certain cases. This is known as the separation principle and gives the ability to independently design the control problems. So, the first step is the design of an LQR controller, as shown in chapter 4.5, and the second is the design of a Kalman filter that will estimate the states. The last step is to combine these two and create an LQG regulator. The LQG that is designed guarantees the stability of the closed-loop system. However, although LQR and Kalman filter on their own provide solid robustness guarantees, their union (LQG) does not. So, LQG optimality does not automatically ensure good robust properties and the robust stability of the closed-loop system must be checked separately after the design of the LQG controller. This happens because the controllers designed are very

sensitive to errors. To promote robustness, some of the tools of robust control, such as the *loop transfer recovery* (LQT), the *quantitative feedback theory* (QFT) or other, must be used. [35] [36] [37]

5.2 Kalman filter

Very common in statistics and control systems, Kalman filter, also known as Linear-Quadratic-Estimation (LQE), is an algorithm that provides a recursive computational methodology for estimating the parameters of a system, from measurements that are noisy or inaccurate, while providing an estimate of the uncertainty of the estimates. The real power of the Kalman filter is not smoothing measurements, but the ability to estimate system parameters that can not be measured or observed with accuracy. This algorithm is used in a series of applications such as object tracking, body weigh estimate on digital scale and guidance, navigation and control of vehicles, particularly aircraft and spacecraft. The filter is named after *Rudolf E. Kalman*, who was one of the primary developers of its theory. [38] [39]

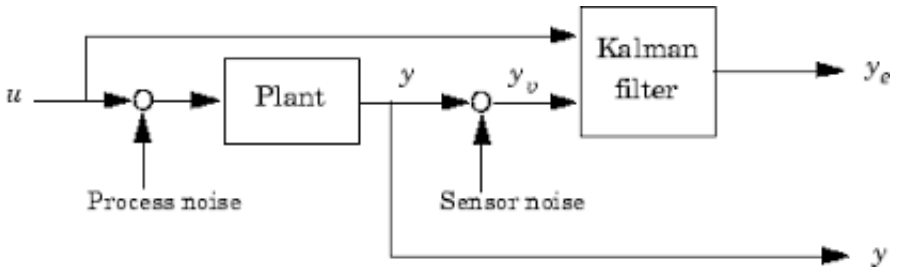

Figure 5.1: Kalman filter block diagram

Considering a linear, time-invariant and continuous time system with the presence of multiple sources of noise:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) + \mathbf{v}(t) \end{aligned} \tag{5.1}$$

where $\mathbf{w}$ represents the *process noise*, modeling uncertainty in the model (e.g., wind disturbance), while $\mathbf{v}$ represents the *sensing noise*, modeling uncertainty in the measurement (e.g., poor rate gyro). It is assumed that both $\mathbf{w}$ and $\mathbf{v}$ have zero mean, such that $E\{\mathbf{w}(t)\} = E\{\mathbf{v}(\mathbf{t})\} = 0$. The noises are modeled as uncorrelated, Gaussian white random

noises, meaning there is no correlation between the noise at one time instant and another:

$$E\left\{\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(\tau)\right\} = \mathbf{Q}\cdot\delta(t-\tau), \quad E\left\{\mathbf{v}(t)\mathbf{v}^{\mathrm{T}}(\tau)\right\} = \mathbf{R}\cdot\delta(t-\tau), \quad E\left\{\mathbf{w}(t)\mathbf{v}^{\mathrm{T}}(\tau)\right\} = \mathbf{0} \tag{5.2}$$

where $E()$ denotes expectation and $\delta(t-\tau)$ is the *Dirac delta function* or impulse function:

$$\delta(t-\tau) = \begin{cases} \infty, & t = \tau \\ 0, & t \neq \tau \end{cases}$$

These noises can be written as $\mathbf{w}(t) \sim N(\mathbf{0}, \mathbf{Q})$ and $\mathbf{v}(t) \sim N(\mathbf{0}, \mathbf{R})$, where the notation $N(a, \mathbf{P})$ indicates a Gaussian noise with mean value a and covariance matrix $\mathbf{P}$. The covariance represents the ”spread” of the distribution – a large covariance corresponds to a broad distribution, representing a larger uncertainty. On the other hand, a small covariance corresponds to a tight distribution around the mean value. Matrices $\mathbf{Q} \in \Re^{nxn}$ and $\mathbf{R} \in \Re^{pxp}$ are usually diagonal, and their components define the disturbance of the system and the noise of the output measurements, respectively. [40]

As mentioned earlier, the terms $\mathbf{w}(t)$ and $\mathbf{v}(t)$ have physical meaning. More specifically, if the process noise $\mathbf{w}$ is big, this means that the model is not good and should not be trusted. On the other hand, if they are small, the model is trustworthy and if they do not exist at all, the model is considered perfect. As for the sensor noise $\mathbf{v}$, if it is big, the measurements are not accurate and should not be taken into consideration, while if it small, the measurements can be used as they are good. In case there are no sensor noise at all, it is impossible to do estimations and so, the Kalman filter can not be used. [41]

Kalman filter estimator takes the form:

$$\begin{aligned} \dot{\hat{\mathbf{x}}}(t) &= \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{L}(t)[\boldsymbol{y}(t) - \mathbf{C}\hat{\mathbf{x}}(t)] \\ \hat{\mathbf{y}}(t) &= \mathbf{C}\hat{\mathbf{x}}(t) \end{aligned} \tag{5.3}$$

the estimation error is $\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$ and forming its derivative $\dot{\mathbf{e}}(t) = \dot{\mathbf{x}}(t) - \dot{\hat{\mathbf{x}}}(t)$ gives:

$$\begin{aligned} \dot{\mathbf{e}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}(t) - [\mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{L}(t)[\boldsymbol{y}(t) - \mathbf{C}\hat{\mathbf{x}}(t)] \\ &= \mathbf{A}\mathbf{e}(t) + \mathbf{w}(t) - \mathbf{L}(t)[\mathbf{C}\mathbf{x}(t) + \mathbf{v}(t) - \mathbf{C}\hat{\mathbf{x}}(t)] \\ &= (\mathbf{A} - \mathbf{L}\mathbf{C})\mathbf{e}(t) + \mathbf{w}(t) - \mathbf{L}\mathbf{v}(t) \end{aligned} \tag{5.4}$$

Equation (5.4) depicts the conflict in the estimator design process. The estimator must find the balance between the speed of the estimator decay rate, controlled by the eigenvalues

of $\mathbf{A} - \mathbf{LC}$ matrix, and the impact of the sensing noise $\mathbf{v}$ through the gain $\mathbf{L}$. A quick state reconstruction requires a fast decay rate, which implies a large gain matrix $\mathbf{L}$. However, this situation usually enlarges the effect of $\mathbf{v}$ on the estimate process. To find an optimal balance between these two problems, the Kalman filter is presented. The optimal estimator of the Kalman filter must minimize the estimation error $\mathbf{e}(t)$ over time. This will happen by designing an optimal estimator that minimizes the trace of the mean square value of the estimation error:

$$\min J = \text{trace } (\mathbf{P}(t))$$

where:

$$\mathbf{P}(t) = E\left\{[\mathbf{x}(t) - \hat{\mathbf{x}}(t)][\mathbf{x}(t) - \hat{\mathbf{x}}(t)]^{\mathrm{T}}\right\} > \mathbf{0}, \quad t > 0 \tag{5.5}$$

Matrix $\mathbf{P}(t)$ is a squared and positive definite matrix that is the solution to Riccati equation:

$$\dot{\mathbf{P}}(t) = \mathbf{AP}(t) + \mathbf{P}(t)\mathbf{A}^T - \mathbf{P}(t)\mathbf{C}^{\mathbf{T}}\mathbf{R}^{-1}\mathbf{CP}(t) + \mathbf{Q} \tag{5.6}$$

Also, given the uncertainty of the system, it is impossible to know the initial condition $\mathbf{x}(0) = \mathbf{x}_0$. For this reason, the initial condition $\mathbf{x}_0$ is considered a random variable with Gaussian distribution, with known mean value $\bar{\mathbf{x}}_0$ and known covariance $\mathbf{P}_0$:

$$E\{\mathbf{x}(0)\} = \overline{\mathbf{x}}_0, \quad E\left\{[\mathbf{x}(0) - \overline{\mathbf{x}}_0]\,[\mathbf{x}(0) - \overline{\mathbf{x}}_0]^{\mathrm{T}}\right\} = \mathbf{P}_0 \tag{5.7}$$

The optimal gain matrix $\mathbf{L}(t)$ of the Kalman filter is:

$$\mathbf{L}(t) = \mathbf{P}(t)\mathbf{C}^{\top}\mathbf{R}^{-1} \tag{5.8}$$

If $\mathbf{A}$, $\mathbf{B}$, $\mathbf{Q}$ and $\mathbf{R}$ are constant, which means that the system model is time-invariant, the time horizon infinite and the noise statistics constant, then under the assumption that the system is $(\mathbf{A}, \mathbf{B})$ is completely controllable and $(\mathbf{A}, \mathbf{C})$ completely observable, the algebraic Riccati equation is derived:

$$\mathbf{AP} + \mathbf{PA}^T - \mathbf{PC}^{\mathrm{T}}\mathbf{R}^{-1}\mathbf{CP} + \mathbf{Q} = \mathbf{0} \tag{5.9}$$

where $\mathbf{P}(t) = \bar{\mathbf{P}} = \mathbf{P}$. From that, it is concluded that matrix $\mathbf{L}$ is time-invariant:

$$\mathbf{L} = \mathbf{PC}^{\top}\mathbf{R}^{-1} \tag{5.10}$$

The estimator gain matrix $\mathbf{L}$ is basically a feedback on the "innovation" $\mathbf{y}(t) - \hat{\mathbf{y}}(t)$. If the uncertainty about the states is high, then $\mathbf{P}$ is large, and the innovation is weighted

LQR	LQE
$\mathbf{A}$	$\mathbf{A}^{\mathrm{T}}$
$\mathbf{B}$	$\mathbf{C}^{\mathrm{T}}$
$\mathbf{Q}_{\mathrm{lqr}}$	$\mathbf{Q}_{\mathrm{lqe}}$
$\mathbf{R}_{\mathrm{lqr}}$	$\mathbf{R}_{\mathrm{lqe}}$
$\mathbf{P}$	$\mathbf{P}$
$\mathbf{K}$	$\mathbf{L}=\mathbf{K}^{\mathrm{T}}$

	LQR	**LQE** (Kalman filter)
Riccati Equation	$\mathbf{A}^{T}\mathbf{P}+\mathbf{P}\mathbf{A}+\mathbf{P}\mathbf{B}\mathbf{R}_{\mathrm{lqr}}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}+\mathbf{Q}_{\mathrm{lqr}}=\mathbf{0}$	$\mathbf{A}\mathbf{P}+\mathbf{P}\mathbf{A}^{T}-\mathbf{P}\mathbf{C}^{\mathbf{T}}\mathbf{R}_{\mathrm{lqe}}^{-1}\mathbf{C}\mathbf{P}+\mathbf{Q}_{\mathrm{lqe}}=\mathbf{0}$
Gain matrix	$\mathbf{K}=\mathbf{R}_{\mathrm{lqr}}^{-1}\mathbf{B}^{\mathrm{T}}\mathbf{P}$	$\mathbf{L}=\mathbf{K}^{\mathrm{T}}=\mathbf{P}\mathbf{C}^{\mathrm{T}}\mathbf{R}_{\mathrm{lqe}}^{-1}$

Table 5.1: Duality among LQR control and Kalman filter (LQE)

more heavily by increasing $\mathbf{L}$. On the other side, if the measurements are highly accurate (components of $\mathbf{Q}_{lqe}$ are low), the measurements are also heavily weighted. So, the Kalman filter estimator take its final form:

$$\begin{aligned}\dot{\hat{\mathbf{x}}}(t)&=\mathbf{A}\hat{\mathbf{x}}(t)+\mathbf{B}\mathbf{u}(t)+\mathbf{L}[\boldsymbol{y}(t)-\mathbf{C}\hat{\mathbf{x}}(t)]\\ \hat{\mathbf{y}}(t)&=\mathbf{C}\hat{\mathbf{x}}(t)\end{aligned}\tag{5.11}$$

As it easily observed from these equations and the equations of chapter 4.5, between the LQR and LQE exists duality. This duality is presented in tables (5.1).

But how is $\mathbf{Q}_{lqe}$ and $\mathbf{R}_{lqe}$ selected? Ultimately, it comes down to which is more trustful, the model or the measurements. Obviously, just like in the LQR there is no point on scaling up both, all the importance is in the relative values of the two matrices. To the one hand, if the system is good but measured by poor sensors, $\mathbf{R}_{lqe}$ will probably be larger than $\mathbf{Q}_{lqe}$. On the other hand, if the dynamics are poorly modeled but well measured, $\mathbf{Q}_{lqe}$ will probably be larger than $\mathbf{R}_{lqe}$. The observer gain matrix $\mathbf{L}$ of the Kalman filter can be calculated via the command (lqr(A', C', Q_{lqe}, R_{lqe}))' in MATLAB.

5.3 LQG regulator

5.3.1 Introduction

An LQG regulator has a very specific role, and that is to connect the LQR controller with the Kalman filter (LQE). The design of an LQG regulator is similar with the design of a compensator presented in chapter 4.7. A block diagram that describes the LQG problem is shown in figure (5.2). [42]

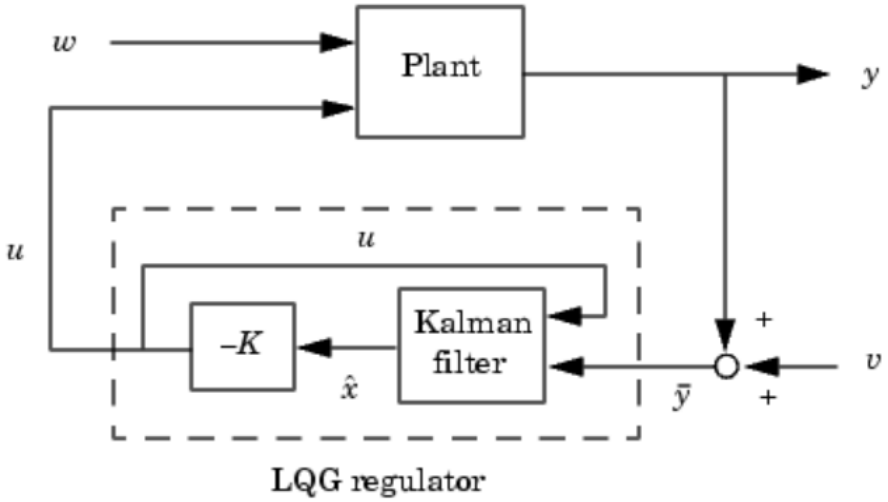

Figure 5.2: LQG problem block diagram

Using the LQR control law of equation (4.20) and replacing the state variable $\mathbf{x}$ with the estimation $\hat{\mathbf{x}}$ of the Kalman filter, the equation that relates the control with the estimation is derived:

$$\mathbf{u}(t) = -\mathbf{K}\hat{\mathbf{x}}(t) \tag{5.12}$$

The estimation error of the Kalman filter is:

$$\mathbf{e}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t) \tag{5.13}$$

Finding the derivative of the error and replacing equations (5.1) and (5.11) gives:

$$\dot{\mathbf{e}}(t) = \dot{\mathbf{x}}(t) - \dot{\hat{\mathbf{x}}}(t) = (\mathbf{A} - \mathbf{LC})\mathbf{e}(t) + \mathbf{w} - \mathbf{Lv} \tag{5.14}$$

Next, plugging the control law of equation (5.12) into the state space model with noise of equation (5.1), having in mind that $\hat{\mathbf{x}}(t) = \mathbf{x}(t) - \mathbf{e}(t)$, gives:

$$\dot{\mathbf{x}}(t) = (\mathbf{A} - \mathbf{BK})\mathbf{x}(t) + \mathbf{BK}\mathbf{e}(t) + \mathbf{w} \tag{5.15}$$

Rewriting equations (5.14) and (5.15) in matrix form:

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{e}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{A}-\mathbf{BK} & \mathbf{BK} \\ 0 & \mathbf{A}-\mathbf{LC} \end{bmatrix} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{e}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{I} & 0 \\ \mathbf{I} & -\mathbf{L} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} \Rightarrow$$
$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{e}}(t) \end{bmatrix} = \mathbf{A}_{cl} \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{e}(t) \end{bmatrix} + \mathbf{B}_{cl} \begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix}. \tag{5.16}$$

Just like what happened in chapter 4.7, matrix $\mathbf{A}_{cl}$ is upper triangular, which means that the system's eigenvalues depend only on the diagonal components. As a result, LQG regulator guarantees stability for the closed-loop system as long as the eigenvalues of both $\mathbf{A}-\mathbf{BK}$ and $\mathbf{A}-\mathbf{LC}$ matrices lie on the left half side of the s-plane. However, both these matrices have guaranteed to provide stability, independently from one another, during the design of the controller (LQR) and the observer (LQE), respectively. So, it is proven that the *separation principle* applies also to the LQG problem.

5.3.2 LQG regulator design

In this section, the LQG control will be applied to the self-balancing motorcycle. As mentioned earlier, the LQG consists of an LQR and a Kalman filter. The first has already been designed to guarantee system's stability in chapter 4.5. So, what is up next is the design of a Kalman filter.

The tricky part during the design of a Kalman filter is the choice of matrices $\mathbf{Q}_{lqe}$ and $\mathbf{R}_{lqe}$. Before starting, it is important to mention that the model used, has been linearized. This interprets as that the model is not trustworthy and so, taking good measurements from the sensors becomes crucial. As a result, the disturbance applied to the system should be higher than the noise of the output measurements. In other words, the components of matrix $\mathbf{Q}_{lqe}$ will be higher than those of $\mathbf{R}_{lqe}$.

For the LQR controller, case 3 will be examined. The control gain has already been calculated in (4.30) as:

$$\mathbf{K} = \begin{bmatrix} -1391.8 & -93.5 & -0.4 \end{bmatrix}$$

For the Kalman Filter, two different cases of covariance matrices will be examined. The first is:

$$\mathbf{Q}_{lqe} = \begin{bmatrix} 10^{-5} & 0 & 0 \\ 0 & 10^{-4} & 0 \\ 0 & 0 & 10^{-4} \end{bmatrix}, \quad \mathbf{R}_{lqe} = 10^{-7} \tag{5.17}$$

with Kalman gain matrix:

$$\mathbf{L} = \begin{bmatrix} 22.25 & 197.73 & -175.61 \end{bmatrix} \tag{5.18}$$

The response of the system in (5.16), with the following initial condition, is presented in figures (5.3) and (5.4).

$$\mathbf{x}(0) = \begin{bmatrix} 0.05 \\ 0.1 \\ 0 \end{bmatrix}, \quad \mathbf{e}(0) = \begin{bmatrix} 0.1 \\ 0.1 \\ 0.1 \end{bmatrix} \tag{5.19}$$

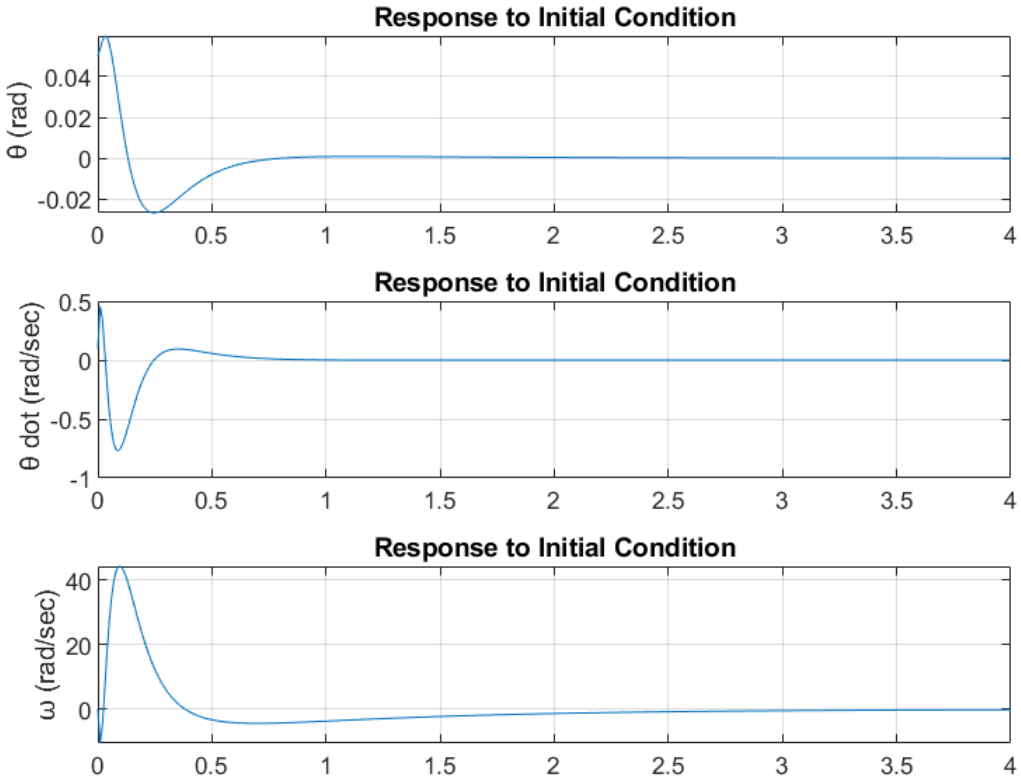

Figure 5.3: Response curves of states for initial conditions: LQG system case 1

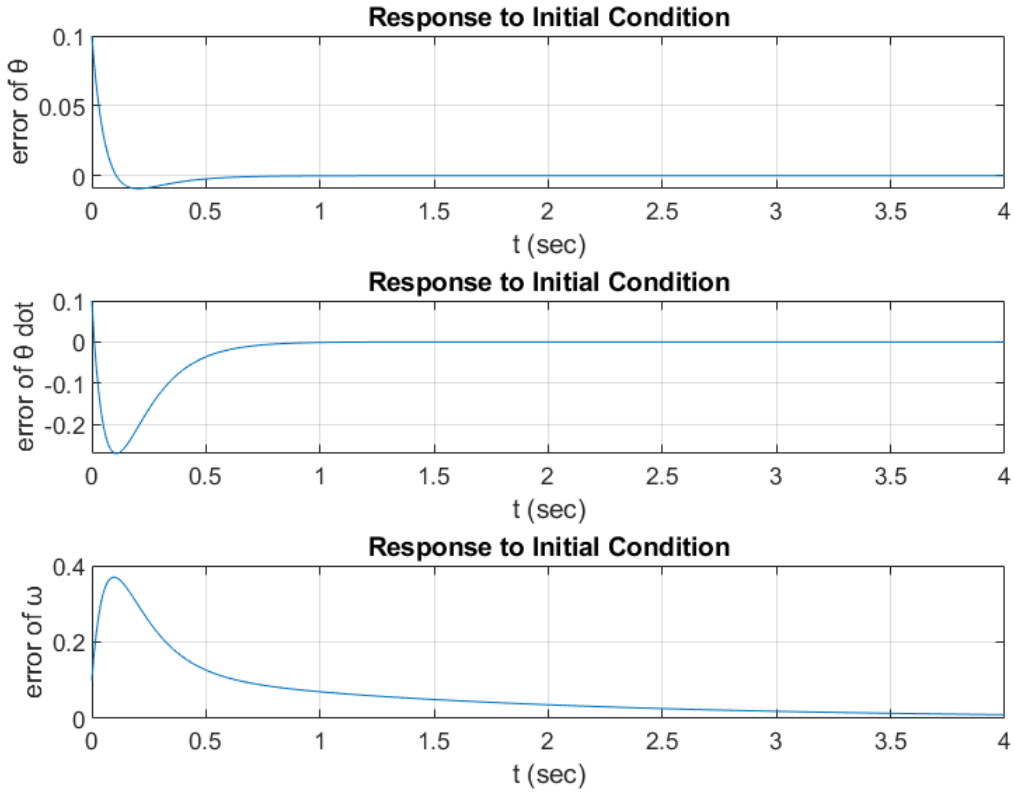

Figure 5.4: Response curves of errors for initial conditions: LQG system case 1

In the second case for LQG, it will be assumed that the model can not be trusted even more and the measurements are better than in case 1. So, the components of $\mathbf{Q}_{lqe}$ will get bigger and those of $\mathbf{R}_{lqe}$, smaller:

$$\mathbf{Q}_{lqe} = \begin{bmatrix} 10^{-2} & 0 & 0 \\ 0 & 10^{-2} & 0 \\ 0 & 0 & 10^{-2} \end{bmatrix}, \quad \mathbf{R}_{lqe} = 10^{-9} \tag{5.20}$$

with Kalman gain matrix:

$$\mathbf{L} = \begin{bmatrix} 3163.3 & 3260.8 & -70.7 \end{bmatrix} \tag{5.21}$$

The response of the system (5.16) with the same initial conditions of (5.19) is presented in figures (5.5) and (5.6).

As observed in tables (5.2) and (5.3), making the model less trustful (by increasing $\mathbf{Q}_{lqe}$) and the measurements more reliable (by decreasing $\mathbf{R}_{lqe}$), the response of the system was better. The overshoots of all state variables were reduced, along with the settling times (except

Specs	Angle θ in deg	Angular velocity $\dot{\theta}$	Rotation speed ω
Negative Overshoot	-1.48	-0.76	-10
Positive Overshoot	3.38	0.45	44
Peak time (sec)	0.03	0.09	0.09
Stability at:	0	0	0
Settling time (sec)	0.61	0.47	1.54
Error goes to 0	yes	yes	yes

Table 5.2: Response information of state variables: LQG case 1.

ω that had a small increase of settling time). Also, in case 2 of LQG the errors go faster to zero than those of case 1. So, it can be said that due to the linearization of the system, the model should not be trusted and on the opposite, the measurements of the senors must be taken more into consideration.

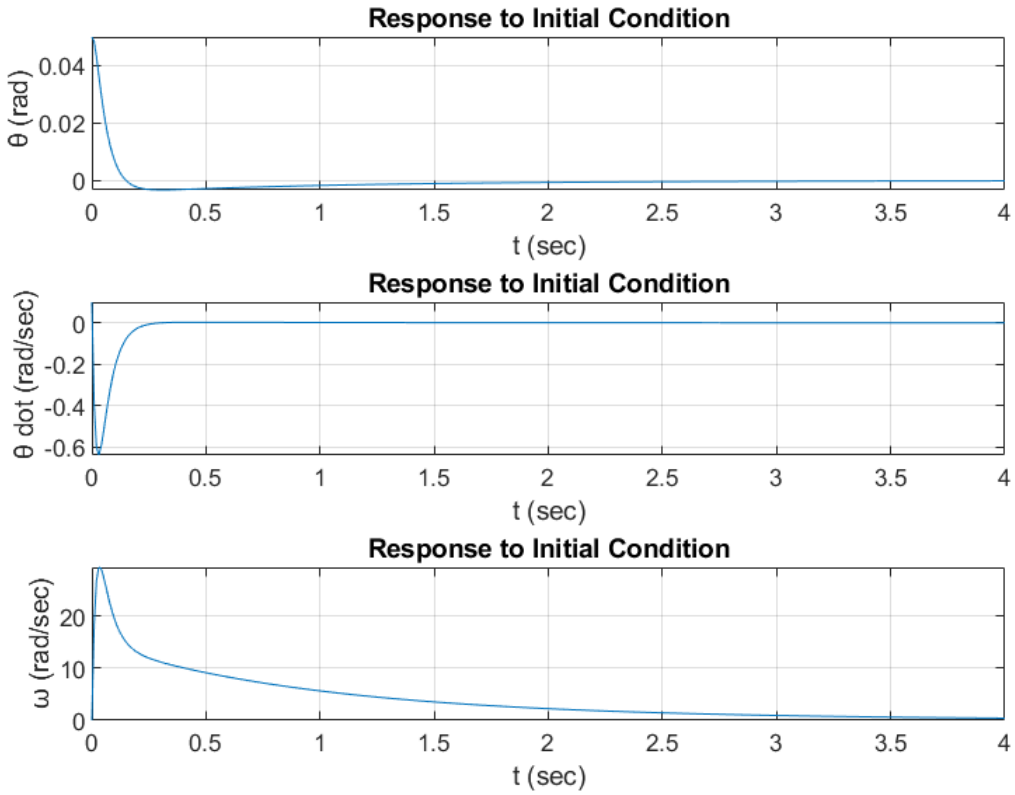

Figure 5.5: Response curves of states for initial conditions: LQG system case 2

Specs	Angle θ in deg	Angular velocity $\dot{\theta}$	Rotation speed ω
Negative Overshoot	-0.1	-0.63	0
Positive Overshoot	2.86	0.1	29.4
Peak time (sec)	0	0.03	0.1
Stability at:	0	0	0
Settling time (sec)	0.19	0.19	1.68
Error goes to 0	yes	yes	yes

Table 5.3: Response information of state variables: LQG case 2.

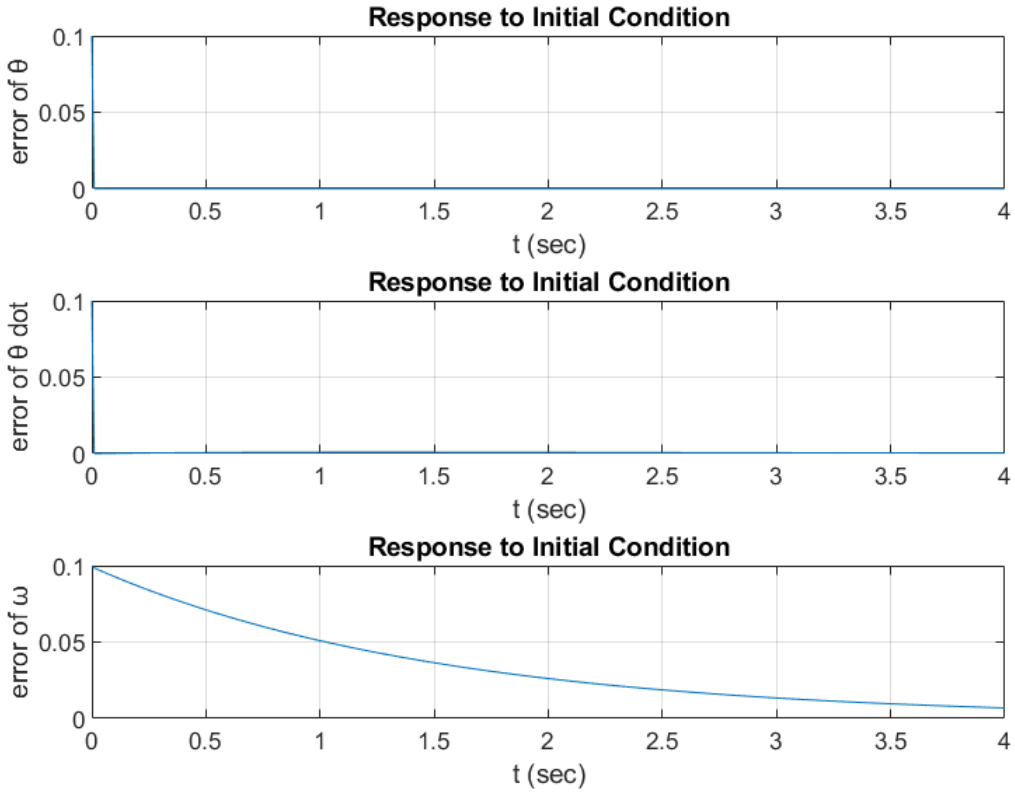

Figure 5.6: Response curves of errors for initial conditions: LQG system case 2

Chapter 6

Conclusion

6.1 Conclusion and discussion

In this thesis, the problem of a self-balancing motorcycle , from Arduino Engineering Kit Rev 2, mainly in continuous time, was presented. Achieving stability was a complex task, even after the system was linearized. Different types of controllers and observers were designed in order to stabilise the system. The most important desire was to make the system stabilise fast and without having big amplitudes.

After the dynamical system of the motorcycle was described by the equations of motion and after these equation were linearized, the state space model was derived. Based on this, the first controllers that were designed were from output feedback theory, which is part of the Classical control theory. The PD and PID controllers were designed with two methods: Ziegler-Nichols and MATLAB's pidtune command. Amongst them, the first one provided better results. However, in the attempt to produce even better results, the trial and error method was also used and managed to guarantee better responses. Also, it was observed that the PID offered to the system another pole close to zero, making it more vulnerbale to instability. This was the main reason the PD was preffered over the PID. Next, one PID and one PD controller were transferred to the discrete time. In the case of the PD, the discretization was successful, however, the discrete-time PID could not reach stability. If a controller works fine in the continuous time, it does not necessarily mean that it will also work fine in the discrete time, where the robot functions. So, applying the work done in continuous time to the robot, always needs discretization first. Alternatively, strange results may occur.

In the next chapter, state feedback control, from modern control theory, was applied to the

system. After the controllability and the observability of the system were checked, the full-state feedback control were designed, using two methods: Pole placement and LQR. In pole placement, the poles of the system were moved arbitrarily in order to achieve better response. From the cases examined, case 4 provided the more promising results ($p_1 = -30$,$p_2 = -60$,$p_3 = -20$). After that an LQR controller was designed. LQR tries to minimize a cost criterion and the resulting control law is considered optimal. However, if the cost criterion changes, the LQG will also change and so optimal does not mean the best that exists. Finding matrices $\mathbf{Q}$ and $\mathbf{R}$ in LQG was done with two methods: leaving $\mathbf{Q}$ as identity matrix while reducing $\mathbf{R}$ and Bryson's rule. The first one gave better results for $\mathbf{Q}$ identity matrix and $\mathbf{R} = 0.01$. However, in full-state feedback control, all states were considered known and most of the times that is not the case. To estimate the states that can not be measured a full-order Luenberger observer needed to be designed. Following that, a system that connects the observer with the controller, called compensator, was designed. Two types of compensators were designed: one with pole placement controller and Luenberger observer and one with LQR controller and also Luenberger observer. In order to obtain a good response, different Luenberger observers were tried ($5, 8$ and 10 times faster that the controller).

In the final chapter, the LQG control was presented and applied to the system. In LQG, the measurements of the outputs can be disturbed by disturbances and noises, making it the system acting more realistic. The LQG combines the concepts of LQR control and LQE, known as the Kalman filter. Due to the fact that an LQR had already been designed, a Kalman filter was also designed and connected with it to create an LQG regulator. The results of the LQG showed that, due to linearization that has undergo, the model must not be trusted much and on the contrary, the measurements of the sensors must be taken more into consideration.

In a nutshell, the self-balancing motorcycle in continuous time can be stabilized with many methods. It is up to the user to decide which one gives better results, based on the needs of every situation. In this thesis, when deciding which case is better over another, a lot of importance was given to the maximum angle θ and how fast the system reaches stability. One thing is for sure, all this work done in continuous time, needs to be transferred to the discrete time before implemented to the real-life robot. Otherwise, the real robot will propably fail to stabilize.

6.2 Future work

As mentioned earlier, one direct future work is to discretize all the controllers and observers designed in this thesis, implement them to the real robot and observe the robot's behavior.

Also, balancing with straight motion could be the next logical extension of the controllers developed. Just like the inertia wheel wheel, the rear wheel motor is driven by a DC motor and is equipped with another encoder. Moreover, the robot carries a servo motor and so balancing with steering could also be tried.

Last but not least, it is obvious that the self-balancing motorcycle has robustness problems. So, robust control could improve the situation, as the robust methods try to manage robust performance and/or stability in the presence of bounded modelling errors. Furthermore, the initial problem is nonlinear and so other types of filters that deal with nonlinear systems could be used. Some of them are: the Extended Kalman filter (EKF), the Unscented Kalman filter (UKF) and the Particle Filter (PF) of Model Predictive Control (MPC).

Bibliography

[1] Maxwell. https://www.biography.com/scientist/james-c-maxwell.

[2] Honda riding assist motorcycle. https://www.youtube.com/watch?v=UZMlu7_4Ub0.

[3] Arduino engineering kit rev 2. https://engineeringkit.arduino.cc/.

[4] Robin H. Pearce. The theory of control : A brief overview. 2012.

[5] Zhong Wanxie. *Duality system in applied mechanics and optimal control*. Boston : Kluwer Academic Publishers, 2004.

[6] E. Fernandez-Car and Enrique Zuazua. Control theory: History, mathematical achievements and perspectives. 01 2003.

[7] Jyrobike: First auto-balancing bicycle unveiled. https://www.bbc.com/news/av/technology-27685867.

[8] Visor down. https://www.visordown.com/news/industry/honda-moving-forward-self-balance-steering-assist-innovations.

[9] Car and bike. https://www.carandbike.com/news/how-do-self-balancing-motorcycles-work-2887409.

[10] Da vinci dc100, a self-balancing electric motorcycle. https://newatlas.com/motorcycles/da-vinci-dc100-self-balancing-electric-motorcycle/.

[11] Vision next 100 by bmw. https://www.bmw-motorrad.com/en/experience/stories/brand/vision-next-100.html.

[12] Ngoc Vu and Hong Nguyen. Balancing control of two-wheel bicycle problems. *Mathematical Problems in Engineering*, 2020:1–12, 07 2020.

[13] Kiattisin Kanjanawanishkul. Lqr and mpc controller design and comparison for a stationary self-balancing bicycle robot with a reaction wheel. *Kybernetika*, 51:173–191, 2015.

[14] Brogan WL. *Modern Control Theory (1st edition)*. Quantum Publishers, 1974.

[15] Alexander Stenger Bernd Girod, Rudolf Rabenstein. *Signals and systems, 2nd edition*. Wiley, 2001.

[16] Unknown. Analysis and design of feedback control systems. Technical report, MASSACHUSETTS INSTITUTE OF TECHNOLOGY DEPARTMENT OF MECHANICAL ENGINEERING.

[17] Abbas Emami-Naeini Gene F. Franklin, J. David Powell. *Feedback control of dynamic systems, 7th edition*. Pearson, 2015.

[18] Susmita Das, Ayan Chakraborty, Jayanta Ray, Soumyendu Bhattacharjee, and Biswarup Neogi. Study on different tuning approach with incorporation of simulation aspect for z-n (ziegler-nichols) rules. *International Journal of Scientific and Research Publications*, 2:2250–3153, 08 2012.

[19] Vishakha Vijay Patel. Ziegler–nichols tuning method, understanding the pid controller. 2020.

[20] Brian R Copeland. The design of pid controllers using ziegler nichols tuning. 2008.

[21] Amer R. Zerek Ibrahim A . El-sharif, Fathi O. Hareb. Design of discrete-time pid controller. 2014.

[22] R. Toth, F. Felici, P.S.C. Heuberger, and P.M.J. Van den Hof. Crucial aspects of zero-order hold lpv state-space system discretization. *IFAC Proceedings Volumes*, 41(2):4952–4957, 2008. 17th IFAC World Congress.

[23] BISWA NATH DATTA. Chapter 6 - controllability, observability, and distance to uncontrollability. In BISWA NATH DATTA, editor, *Numerical Methods for Linear Control Systems*, pages 159–199. Academic Press, San Diego, 2004.

[24] Ogata Katsuhiko. *Modern Control Engineering. 5th Edition*. Pearson, 2010.

[25] Eduardo Sontag. *Mathematical Control Theory: Deterministic Finite Dimensional Systems. Second Edition*. Συμμετρία, 2014.

[26] Miss. T. V. Gudadhe and Dr. Anvita Telang. Pole placement technique for effective design and analysis of state feedback controller. 2021.

[27] Russ Tedrake. *Underactuated Robotics*. 2022.

[28] Gioele Zardini. Lecture 9: Linear quadratic regulator, 2018.

[29] Αναστάσιος Πουλιέζος. *Σύγχρονη Θεωρία Ελέγχου*. Σύνδεσμος Ελληνικών Ακαδημαϊκών Βιβλιοθηκών, 2015.

[30] David Luenberger. An introduction to observers. *Automatic Control, IEEE Transactions on*, 16:596 – 602, 01 1972.

[31] César Pichardo-Almarza, Ahmed Rahmani, Geneviève Dauphin-Tanguy, and Marisol Delgado. Luenberger observers for linear time-invariant systems modelled by bond graphs. *Mathematical and Computer Modelling of Dynamical Systems*, 12:219 – 234, 2006.

[32] J. B. PEARSON. Compensator design for dynamic optimization. *International Journal of Control*, 9(4):473–482, 1969.

[33] Dongxiao Wu and Jun Wu. Separation principle for networked control systems with multiple-packet transmission. *Control Theory and Applications, IET*, 5:507 – 513, 03 2011.

[34] Tryphon T. Georgiou and Anders Lindquist. The separation principle in stochastic control, redux. *IEEE Transactions on Automatic Control*, 58(10):2481–2494, 2013.

[35] Gerard van Willigenburg and W.L. Koning. Optimal reduced-order compensators for time-varying discrete-time systems with deterministic and white parameters. *Automatica 35 (1999)*, 01 1999.

[36] Abdulrahman Kalbat. Linear quadratic gaussian (lqg) control of wind turbines. In *2013 3rd International Conference on Electric Power and Energy Conversion Systems*, pages 1–5, 2013.

[37] Karl Johan Åström. *Introduction to stochastic control theory*, volume 70 of *Mathematics in science and engineering*. Academic Press, United States, 1970.

[38] N. Shimkin. Estimation and identification in dynamical systems, Fall 2009.

[39] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, USA, 1995.

[40] Brandon Luders. Linear quadratic estimator (lqe), lecture, November 2010.

[41] *Random Processes and Stochastic Systems*, chapter 3, pages 67–129. John Wiley and Sons, Ltd, 2008.

[42] J. Doyle. Guaranteed margins for lqg regulators. *IEEE Transactions on Automatic Control*, 23(4):756–757, 1978.

APPENDICES

Appendix

MATLAB Codes

.1 Robot Parameters

```
g = 9.80665;          % gravity constant
m_r = 0.2948;         % mass of the rod
m_w = 0.0695;         % mass of the inertia wheel
R = 0.05;             % radius of the inertia wheel
r = 0.02;             % cross section radius of the rod
l = 0.13;             % corresponding lengths
l_AD = l;
l_AC = l;             % assume wheel is mounted on the top of the pendulu
l_AB = l/2;
I_w_C = 0.5*m_w*R^2; % corresponding inertias
I_w_A = I_w_C + m_w*l_AC^2;
I_r_B = (1/12)*m_r*(3*r^2+l_AD^2);
I_r_A = I_r_B + m_r*l_AB^2;
K=0.019;
R=6.48;
```

.2 State Space Matrices

```
A=[0  1  0
```

```
    96.524 0 0.0194
    -96.524 0 -0.667];

B=[0;-1.0235;34.7743];

C=[1,0,0];

D=[0];
```

.3 Transfer Function

```
[a,b] = ss2tf(A,B,C,D);
tfu = tf(a,b); % system transfer function
z=zero(tfu); %zeros of system
p=pole(tfu); %poles of system
figure
pzplot(tfu); %pole zero map
rlocus(-tfu) % root locus figure
```

.4 PID and PD design

```
Ts = 0.01;
t=0:Ts:3;
Kp =-280; %proportional gain
Ki = -60;  % integral gain
Kd = -23; %derivative gain
Cpi = pid(Kp,Ki,Kd) % PID transfer function
tfPID=feedback(tfu,Cpi); %closed loop system transfer function
impulse(tfPID,t) % Impulse response
title(['Impulse response with PID, Kp=-2209, Kd=-70'])
ylabel('Radians')
```

```
% Ziegler Nichols
%----------------ZIEGLER NICHOLS---------------
[b,a] = ss2tf(A,B,C,D);
tfu = tf(b,a);
rlocus(-tfu)
[Ku]=-rlocfind(-tfu,0);
Kp=Ku;
Ki=0;
Kd=0;
Cpi = pid(Kp,Ki,Kd)
tfPID=feedback(tfu,Cpi);
impulse(tfPID,t)
[~,locs]=findpeaks(impulse(tfPID,t));
Pu=mean(diff(locs)*0.01);
%----------------PID-----------------------------------------
Kp=0.6*Ku;
Td=Pu/8;
Ti=Pu/2;
Kd=Kp*Td;
Ki=2*Kp/Pu;
Cpi = pid(Kp,Ki,Kd);
tfPID=feedback(tfu,Cpi);
impulse(tfPID,t)
title(['Impulse response with PID, Kp=-4657, Ki=-130000, Kd=-41.3'])
ylabel('Radians')
%----------------PD------------------------------
Kp=0.8*Ku;
Td=Pu/8;
Kd=Kp*Td;
Ki=0;
Cpi = pid(Kp,Ki,Kd);
```

```
tfPID=feedback(tfu,Cpi);
impulse(tfPID,t)
title(['Impulse response with PD, Kp=-6209, Kd=-55.1'])
ylabel('Radians')

% Zero Order Hold
Ts=0.01; % sampling time
tfdis=c2d(tfPID,Ts,'ZOH') % From continuous time to discrete
pzmap(tfdis) % Pole zero map of discrete transfer function
```

.5 State Feedback Control

```
%controllability matrix
C=ctrb(A,B)
%Observability matrix
O=obsv(A,C)

% Pole placement example
clear
poles=[-347.9 -9.68+0.14i -9.68-0.14i]
KK=place(A,B,poles);
Al=A-B*KK;
eig(Al);
[b,a] = ss2tf(Al,B,C,D);
tfu = tf(b,a);
impulse(tfu,t) % response for θ
title(['Impulse response of θ'])

% LQR example
clear
Q=[1 0 0
   0 1 0
```

```
     0  0  1];
R=[0.01];
KK=lqr(A,B,Q,R);
Al=A-B*KK;
eig(Al);
[b,a] = ss2tf(Al,B,C,D);
tfLQR = tf(b,a);
t=0:Ts:5;
impulse(tfLQR,t)

%Luenberger Observer example
clear
poles=[-1735  -48.35-0.7i  -48.35+0.7i]
LL=place(A',C',poles)';
Al=A-LL*C;
eig(Al);

%Compensator example
clear
poles=[-347.9  -9.68+0.14i  -9.68-0.14i];
KK=place(A,B,poles);
Al=A-B*KK;
eig(Al);

poles=[-694  -19.34+0.28i  -19.34-0.28i];
LL=place(A',C',poles)';
Al=A-LL*C;
eig(Al);

Ac=A-B*KK-LL*C; %compensator dynamics
Bc=LL;
Cc=KK;
```

```
[f,g] = ss2tf(Ac,Bc,Cc,D);
tfg = tf(f,g);
[a,b] = ss2tf(A,B,C,D); %of the initial system
tfu = tf(a,b);
G=feedback(tfu,tfg);
title(['Impulse response of θ'])
impulse(G,t);
```

.6 LQG regulator

```
Qn = [0.01 0 0; 0 0.01 0; 0 0 0.01];
Rn = 0.000000001;
Q=[1 0 0
    0 1 0
    0 0 1];
R=[0.01];
LLL=(lqr(A',C',Qn,Rn))'; % Kalman gain

sys = ss([A-B*KK B*KK; zeros(3,3) A-LLL*C],eye(6),eye(6),eye(6));
t = 0:0.01:4;
z = initial(sys,[0.05;0.1;0;0.1;0.1;0.1],t);
x1 = [1 0 0 0 0 0]*z';
x2 = [0 1 0 0 0 0]*z';
x3 = [0 0 1 0 0 0]*z';
e1 = [0 0 0 1 0 0]*z';
e2 = [0 0 0 0 1 0]*z';
e3 = [0 0 0 0 0 1]*z';
subplot(3,1,1); plot(t,x1),grid
title('Response to Initial Condition')
xlabel('t (sec)'),ylabel(' θ (rad)')
```

```
subplot(3,1,2); plot(t,x2),grid
title('Response to Initial Condition')
xlabel('t (sec)'),ylabel(' θ dot (rad/sec)')

subplot(3,1,3); plot(t,x3 ),grid
title('Response to Initial Condition')
xlabel('t (sec)'),ylabelω(' (rad/sec)')

subplot(3,1,1); plot(t,e1),grid
title('Response to Initial Condition')
xlabel('t (sec)'), ylabel('error of θ')

subplot(3,1,2); plot(t,e2),grid
title('Response to Initial Condition')
xlabel('t (sec)'), ylabel('error of θ dot')

subplot(3,1,3); plot(t,e3 ),grid
title('Response to Initial Condition')
xlabel('t (sec)'), ylabel('error of ω')
```