Speed Control of a PMDC Motor

Using a DC-to-DC Buck Converter

Devrat Singh, Fikrican Özgür, Matei Căciuleanu
B.Sc. in Electronics and Computer Engineering
Group ED4-1-F19

Fourth Semester Project

This document was typeset using Overleaf, an online LaTeX editor. The control systems were analyzed using MATLAB R2019a. The simulations were performed in SIMULINK 9.3 and the graphs were made in MATLAB as well. The other figures were created in Inkscape and draw.io.

Faculty of Engineering and Science
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Speed Control of a PMDC Motor

Theme:
Control Engineering

Project Period:
Spring Semester 2019

Project Group:
ED4-1-F19

Participant(s):
Devrat Singh
Fikrican Özgür
Matei Căciuleanu

Supervisor(s):
Amin Hajizadeh

Copies: 1

Page Count: 85

Date of Completion:
June 3, 2019

Abstract:

The present report explores the matter of regulating the angular speed of a PMDC motor interfaced by a step-down power converter employing fundamental techniques of classical and modern control theory. Suitable models retaining only the relevant system dynamics are constructed and validated following the experimental motor parameter identification and the design of the switching voltage regulator. Four feedback control methods are put forward: PID for speed-only control, cascade PID for speed and current control, pole placement and LQR. With the aid of MATLAB and SIMULINK, their performance is assessed within the context of stability and reference tracking. The controllers are successfully implemented using an MCU, allowing the individual designs to be tested on the real setup. A qualitative analysis is carried out through a testing session that evinces the superiority of modern control design.

Table of Contents

List of Figures

List of Tables

Preface

The project entitled "Control Engineering" was written by three students from the Electronics and Computer Engineering Bachelor's program at Aalborg University Esbjerg, for the P4 project in the fourth semester. Hereafter, every mention of "we" refers to the three co-authors listed below.

The completion of this project would not have been possible without the contribution of our supervisor, Amin Hajizadeh, who directed his efforts towards guiding us to the right path. We would like to express our gratitude for his support and relevant advice.

Aalborg University, June 3, 2019

Devrat Singh
<dsingh17@student.aau.dk>

Fikrican Özgür
<fyozgy17@student.aau.dk>

Matei Căciuleanu
<ccaciu17@student.aau.dk>

Chapter 1

Introduction

"An electrical drive is defined as a form of machine equipment designed to convert electrical energy into mechanical energy and provide electrical control of the process" [1]. Electric motors are conventional components of electric drive systems and as a means of controlling them these drives are incorporated into numerous applications. With their use ranging from everyday household appliances to a very large portion of industrial processes, such as paper mills, printing presses, mining or propulsion systems for underwater vehicles. It is no exaggeration to say that electric drive applications are as broad and vast as industrialization itself [2].

The functionality of an electric drive system, some of them being speed, torque or position control, is dependent on the application. Among all functionalities, speed control of electric motors is one of the most sought-after aspects of electric drives (after reliability and efficiency) [3]. The system controlling the speed, usually referred to as Adjustable Speed Drive (ASD) or Variable Speed Drive (VSD), allows the adjustment of the motor speed to any desired value within its constraints, contrary to just zero or maximum speed. It can also provide control such that the motor speed matches to the load requirements. Besides adjusting the speed of an electric motor, a VSD can also regulate the control parameters in order to achieve a constant speed level, in the case of a varying load [4]. This last prospect serves as a leading requirement for this report and will be thoroughly discussed in the upcoming sections. One example of the feature of speed control can be seen in electric cars, where the drive system is utilized to maintain a constant speed level when the load on the car varies as the travelling slope changes (see Figure 1.1). Other applications of constant speed with variable load are conveyor belts, elevators, and electric traction systems [1].

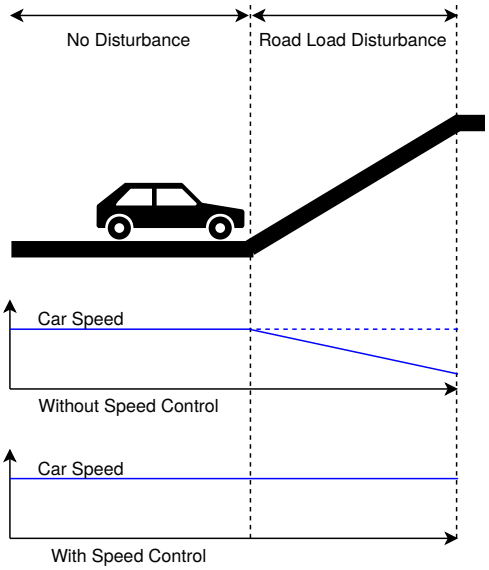

Figure 1.1: Car Speed Variation with Variable Load

An electrical drive, and in our case, an ASD, is implemented through the interworking of the following components: power source, power modulator, control unit, sensing unit and motor [1]. Each of these are crucial elements of the speed control system that is to be developed and implemented in this project and thus they will be discussed un detail in the succeeding sections.

Chapter 2

Problem Analysis

2.1 Applications of DC Motors

Whether we are talking about household appliances (refrigerators, washing machines, vacuum cleaners), electric automobile propulsion systems or industrial applications (machine tools, pumps, cranes) [5], [6], the ubiquity of electric motors is an undeniable fact. It is because of this vast range of uses why several variations of such devices have been developed. A common feature of all motors[1] consists of the use of electricity to produce rotational mechanical energy. Additionally, every motor has a stationary part, a stator, and a rotating component, referred to as rotor [5], [6], [7]. DC motors, in particular, are electrical machines whose input is in the form of constant voltage and current. They are rather complex in terms of construction due to the need of brushes and a commutator ring to facilitate the energy transfer between the stator - either a permanent magnet (PM) or an electromagnet - and the armature windings of the rotor [5], [7]. In spite of that, conventional DC motors[2] exhibit certain traits that place them in pole position for specific applications.

The first advantage would be precisely the DC nature of the motor, which makes it suitable for situations in which DC power systems are readily available. This is the case in automotive, where the the battery of the vehicle supplies the starter motor and the motors that control the opening of the power windows and move the windscreen wipers [5], [7]. Another area in which DC machines excel is represented by speed control, because the angular velocity can be straightforwardly adjusted by adequately varying the voltage at the armature using unsophisticated hardware [5], [7], [8]. According to Leonhard [9] 1996, DC motors "have been dominating the field of adjustable speed drives for over a century; they are still the most common choice if a controlled electrical drive operating over a wide speed range is specified". This assertion is substantiated in a 2014 book by Rashid [10], who avers that "dc drives are currently used in many industries. It might be a few decades before the dc drives are completely replaced by ac drives".

The list of applications pertaining to these motors is quite long. They are frequently utilized in servomechanisms [11], rolling mills [12], reeling [13], crane

[1] Throughout this report, the short form "motor" will be used with the meaning of "electric motor"

[2] The terms "DC motor" and "PMDC motor" will be used interchangeably

hoists (Figure 2.1(a)) propulsion of Automated Guided Vehicles [14] (Figure 2.1(b)) or industrial tasks requiring traction [11] (high-torque series DC motors may be found in locomotives or lifts [7]). Other niche uses include: controlling telescope movement [15] and panning cameras for robots utilized in telemedicine [16].

(a) Hoist (Source: [17]) **(b)** Automated Guided Vehicle (Source: Adapted from [18])

Figure 2.1: Examples of DC Motor Applications

Knowing the importance of DC machines for all the preceding applications, the rationale for investigating their potential control techniques should be apparent to the reader at this point.

2.2 Power Electronics

Power electronics is an intermediate state where one form of electrical energy (usually from power supply) is converted to another form by means of changing voltage, current and frequency. Such electric circuits are referred to as converters [19]. As mentioned before, power source and power modulator are two crucial elements of a speed control system. It is our job to design relevant power electronics, the converter, to provide necessary electric power to the DC motor. Depending on the power supply and load requirements (AC or DC), different types of converters may be more suitable for the application in question, e.g.: rectifier, inverter, AC-to-AC converter and DC-to-DC converter [20].

In this project, we developed a generic DC motor speed controller for applications that are powered with a DC supply. We did not focus our attention on a specific application, but rather developed a solution that may be applicable for various purposes where a similar setup exists (a DC supply powering a DC motor through a converter). Hence, the choice of a DC-to-DC converter for the project becomes clear immediately. Such power converters may be designed to step-up or step-down the supply voltage, referred to as boost and buck converter respectively. These two types of switching regulators have no superiority over each other and selecting one only depends on the application requirements. A buck converter was implemented in the project as there was a need to lower the voltage to a level necessary to provide constant angular speed under changing conditions. Since the project involves a constant DC power supply (not a battery), the aspect of utilizing regenerative breaking was not considered, ergo there was no need to implement a bidirectional converter.

In fact, there are various means of lowering the supply voltage. For instance, one can also lower DC voltage using simpler circuits such as voltage divider, linear voltage regulator or as simple as a transistor that connects the source and the motor according to a PWM (pulse width modulation) signal. However, it is also important to consider aspects such as efficiency and flexibility in conversion of electrical energy. An efficient power electronics design requires less usage of components with high dissipative losses, such as resistors. Therefore, only components with minimum losses, fundamentally reactive components such as inductor, capacitor, transformer and switches are considered for an efficient converter design [19]. Consequently, a voltage divider is not a good choice in terms of efficiency as it incorporates only resistive elements. Regarding linear voltage regulators, the fact that they always output a constant voltage value makes them less flexible to use when the required voltage for the load varies in time.

Fully controlled power switches, such as BJT (bipolar junction transistor), MOSFET (metal-oxide-semiconductor field-effect transistor), IGBT (insulated gate bipolar transistor) and GTO (gate turn-off thyristor) are commonly used in power electronics as they allow the control of current and voltage. Therefore the aforementioned solution to lower down the voltage using a PWM-driven switch is a possible and valid choice. In fact, a buck converter is, in its essence, a PWM-driven switch with LC filtering at the output. The filtering softens the power delivered to the motor as the capacitor and the inductor prevent sudden changes in voltage and current, respectively [21], thus providing a smoother drive. With a buck converter without LC filtering, that is the beforementioned PWM-driven switch solution, the motor will be driven with a fast varying signal across its terminals with sharp edges. Driving a motor with such a signal may alter the motor behaviour due to radiated electromagnetic interference (EMI). Therefore, a buck converter is usually preferred over a PWM-driven switch to lower down battery voltage and deliver necessary power to the motor in a reliable manner [22]. For this reason, a buck converter seems appropriate to be implemented in the solution to lower the power source voltage.

2.3 Control and Modeling of Systems

Control theory consists of two main branches: linear and non-linear. Linear control theory is applicable to systems that obey homogeneity and superposition principles, which signifies linear input-output relationship. Such systems are governed by linear differential equations. In addition to this, linear time-invariant systems (LTI), a subclass of linear systems, also exhibit unchanging system behaviour over time. However, no real-world system is linear. They are often governed by non-linear differential equations. And few mathematical techniques (limit cycle theory, Poincaré maps, Lyapunov stability theorem and so on) developed for non-linear systems are more difficult to implement and less general. On the other hand, there exist more powerful and general mathematical techniques (Laplace transform, Fourier transform, Z-transform, Bode plot, root locus, Nyquist stability criterion and so on) for linear systems. The fact that a wide range of real-world physical systems can be approximated very accurately by an LTI model and we can utilise

linear mathematical techniques to solve them motivates us to focus on linear control theory [23]. Moreover, our semester course on control theory is only focused on linear systems. Thus, the project will center upon linear control techniques and linear or linearized systems.

The PID (proportional–integral–derivative) controller is the most commonly used feedback control architecture, approximately 95% of all controllers according to [24], since it is intuitive and easy to implement. Even though more advanced controller techniques may increase system performance, they are not often preferred due to the extra effort and expense, as long as PID controller provides adequate results [25], [26]. PID controllers are also well suited for single-input single-output (SISO) linear systems as in our project's case. There also exist other control techniques for more advanced systems such as non-linear multiple-input multiple-output (MIMO) systems. For instance, LQR (Linear Quadratic Regulator) and MPC (Model Predictive Control) are two optimal control techniques that optimize a certain cost index and, thus, minimize a cost function (e.g.: time, fuel, speed). SMC (Sliding Mode Control) is a robust control technique used to sustain robustness and/or stability of the system in the presence of slightly faulty system model [27], [23]. However, as this list may continue to grow with examples of many great control techniques, developing a PID controller will be well-suited for our SISO linear system as a starting point. Modern control techniques (pole placement and LQR) will also be implemented and compared.

Regardless of the chosen control technique, developing a controller for a system first requires the control engineer to model the system. The quote from [28] simply explains the necessity for system models: "A model of a system is a tool we use to answer questions about the system without having to do an experiment". They relieve control engineers from conducting costly and even maybe hazardous experiments, hence aiding in controller development. As a system may be modeled based on physical relations and mathematical equations governing them, there also exist another methodology called "black-box modeling" that examines the input-output relationship of the system. Another technique, named "grey-box modeling", utilizes both physical and black-box modeling techniques, as not all physical relations may be simply represented using first principles. Regardless of the opted modeling technique, the control engineer needs to carefully decide which details to include in their model as there is no need to include, if possible, every small detail of the system. Deriving a simple but valid model of a system is the essence of this process. Validation of the model with collected real data is a latter significant step that leads to a more reliable ultimate model on which a controller can be implemented. In addition, being aware of model's domain of validity and adhering to valid system parameters, that is using the model within the validated domain, is crucial for simulation correctness [28].

2.4 Analog and Digital Control

According to some established instances of the control theory [29], [30], [31] and power electronics literature [32], since the emergence of digital computers, the general tendency has been towards the use of controllers relying on such devices.

This phenomenon stems from the advantages digital controllers bring in terms of cost, size, noise immunity and flexibility [31].

A comprehensive comparison between analog and digital control of power electronics systems is carried out in [33], based on the research and results described in multiple scientific papers related to this field. Here, several assets of digital controllers are identified. First of all, they are reprogrammable and hence modifying the controller becomes essentially a matter of software, with no hardware alteration being required. Furthermore, it is stated that digital components are more durable and their performance is less likely to be affected by noise. Nonetheless, the finite word length of the utilized processor, as well as the limits in the ADC/DAC resolution, make digital control imperfect and therefore inferior to the analog counterpart, as far as precision is concerned. These shortcomings are arguably tolerable and the consensus was identified to be favorable for the digital controllers.

An assessment of the performance of the two types of controllers is also treated in [34], where the authors use SIMULINK to model a buck-boost converter and then evaluate its behaviour for both analog- and digital-controller scenarios. The results show a considerable decrease in rise time, settling time and maximum overshoot, respectively, for the responses of the system based on the digital controller.

These references corroborate the superiority of digital control and consequently the approach will be also taken in the present project. To implement such a controller however, specific hardware is necessary and several options are available on the market. The most prominent pieces of equipment are: microcontroller (MCU), Field Programmable Gate Array (FPGA), Digital Signal Processor (DSP) [35] and Programmable Logic Controller (PLC) [36] Bahamas. Since it is clearly asserted in [30] that "Most digital controllers used today are built around a microcontroller", and given the relevant background we have regarding these devices, the choice of selecting an MCU for the development of the digital control system was significantly expedited. Our pick was Arduino Mega, because of its user-friendliness and prevalence in electronics projects.

2.5 Sensors

Speed Sensing

For a variable speed drive, the output of interest is the rotational speed of an electromechanical device. Consequently, the sensor which completes the closed loop, is responsible for measuring the angular speed of the motor and producing an output that is subsequently compared to a reference speed. As we know, this comparison yields an error which is consequential to the implemented control method. Accordingly, if the generated error signal is itself wrong due to inaccurate measurement from the sensor, the controller can do very little to achieve the desired value. Therefore, the selection of a suitable sensor is an unarguable aspect of the project [10], [37], [38].

The sensors for speed measurement come in various types but the most commonly employed are: tachogenerators, resolvers, Hall effect sensors, and encoders

[39], [10]. Out of these options, encoders and digital Hall effect sensors present themselves as the most suitable choices for the speed control system to be implemented. They are inherently digital, easily applicable, economically viable and give a good resolution [39]. The DC motor used in the project [40] came together with an in-built speed sensor whose output signal is incremental and whose sensing technology is magnetic (Hall effect) [41]. Henceforth, using this digital Hall effect sensor was the best option.

Current Sensing

The components used in speed control drives naturally come with limitations to currents they can handle. In most practical applications, a very fast response to a sudden change in speed would require large peak currents [38]. Furthermore, the stalling of a motor due to the load would also result in excessive current [37]. This current, if higher than the rated value, can harm the motor, as well as the converter. Accordingly, the problem can be solved through current sensing and limiting, the latter being achieved by restraining the desired current value to the maximum rated current for the motor [37], [38]. Another facet of current sensing is that it serves as a state in the modern control approach and is necessary for full-state feedback. The standard current sensors used in electric drives are Hall-effect-based [10], [42].

2.6 Problem Formulation

The preceding pages were meant to provide the reader with the general understanding of the project, such that a detailed discussion of the methods and materials can be easily followed. The Problem Analysis chapter discussed the significance of electrical speed drives, DC motors, digital controllers and sensors, as well as the need of using modeling, control theory and power electronics to obtain efficient and reliable systems. As a result, we may define the primary goal of the project in the from of the following problem statement:

Development of a digitally-controlled electrical speed drive based on a DC-to-DC converter for maintaining the speed of a DC motor at a constant level under varying load conditions.

2.7 System Overview

In the endeavour to solve the aforementioned problem, we will employ a permanent magnet DC motor powered by a constant DC supply. The voltage of the supply will be adjusted to the motor requirements (from 0 to 12V) by interfacing the actuator with a buck converter. The duty cycle of the control signal applied to the switch of the regulator will be produced using Arduino Mega. The employed digital control schemes will be PID, pole placement and LQR. The controlled variables will be the angular speed and, if required by the control technique, the motor armature current, with the corresponding sensors being of course utilized for

closed-loop control. The modeling and simulations will be conducted using MATLAB and SIMULINK. For convenience, Table 2.1 summarizes all the aforementioned features of the electrical speed drive to be developed.

Table 2.1: Overview of the Features of the Developed Speed Drive

Control Structure	Closed-Loop Structure	**Control Methods**	PID, LQR, Pole Placement
Control System Type	SISO Dynamic LTI	**Controlled Variables**	Angular Speed (and Current)
Control Type	Digital	**Microcontroller**	Arduino Mega
Programming Languages	MATLAB C/C++	**Power Electronics**	Buck Converter
Motor	PM Brushed DC	**Power Supply**	Constant DC
Software	MATLAB SIMULINK	**Sensors**	Speed Current

Project Objectives

As the Problem Analysis draws to a close, we would like to set some objectives that the solution needs to meet to be considered successful. Tables 2.2 and 2.3 list the requirements for the converter design and the project objectives. Once the solution is developed, it will be evaluated by referring to these lists.

Table 2.2: Buck Converter Requirements

Efficiency	$> 90\%$	**Output Voltage Ripple**	$< 0.01\%$
Input Voltage	$12V$	**Frequency**	$20kHz$
Output Voltage	$0 - 12V$	**Operation Mode**	Continuous Current

Table 2.3: System Requirements

Project Objectives
Design a buck converter in continuous current mode to provide the necessary voltage and current
Create models for the motor and converter and verify them by experiment
Achieve satisfactory time domain specifications
Employ established control tools to analyse system performance (Step response, Root locus and so on)
Convert the developed continuous-time controllers into digital controllers
Compare the system responses under various control methods
Prevent excessive voltage and current to be applied to the motor
Utilize both classical and modern control techniques

General Control Loop

A rudimentary illustration of the closed-loop structure that will be utilized for all the control methods is shown in Figure 2.2. It represents a culmination of the analysis done in the present chapter, consisting of the interconnection between all the system components. It also summarizes the interdependent relationship between the power electronics and control aspects of speed control.

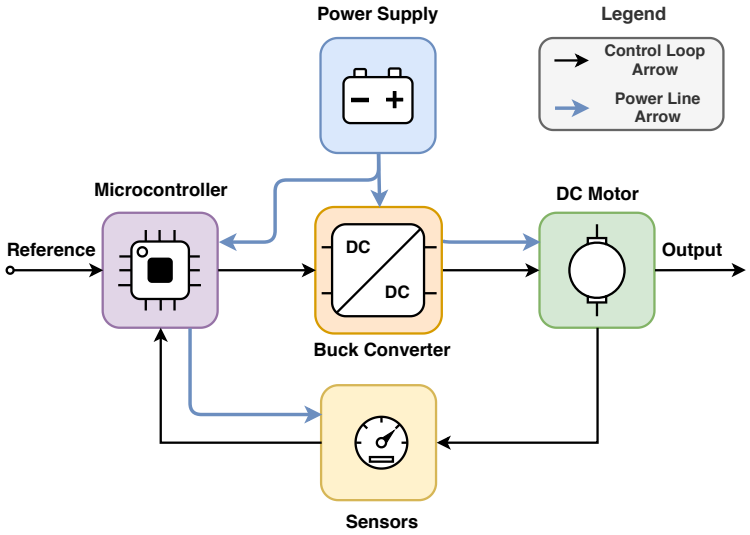

Figure 2.2: General Control Loop of the Electrical Speed Drive Along with System Components

Project Flowchart

The raison d'être of the flowchart shown in Figure 2.3 is to encapsulate the implementation stages and the process of the speed control design.

2.8 Project Delimitation

Direction Control

The diversity of power converters that can be used to achieve speed control of DC motors is rather high, with configurations allowing rotational motion in both clockwise and counterclockwise directions being quite common [32], [43]. A classical buck converter is an exponent of single-quadrant converters and hence, the voltage at the motor terminals has a fixed polarity. Notwithstanding more intricate circuits, such as an H-bridge buck-boost converter [44] or a DC/DC buck-boost converter-inverter [45], could be used for direction control, the objectives of the project are actually independent of this feature. While the ability to rotate in both directions is clearly an advantage, speed control can be successfully achieved without it. In view of that, the traditional buck switching regulator was chosen for this project and direction control was deliberately excluded from the implementation part.

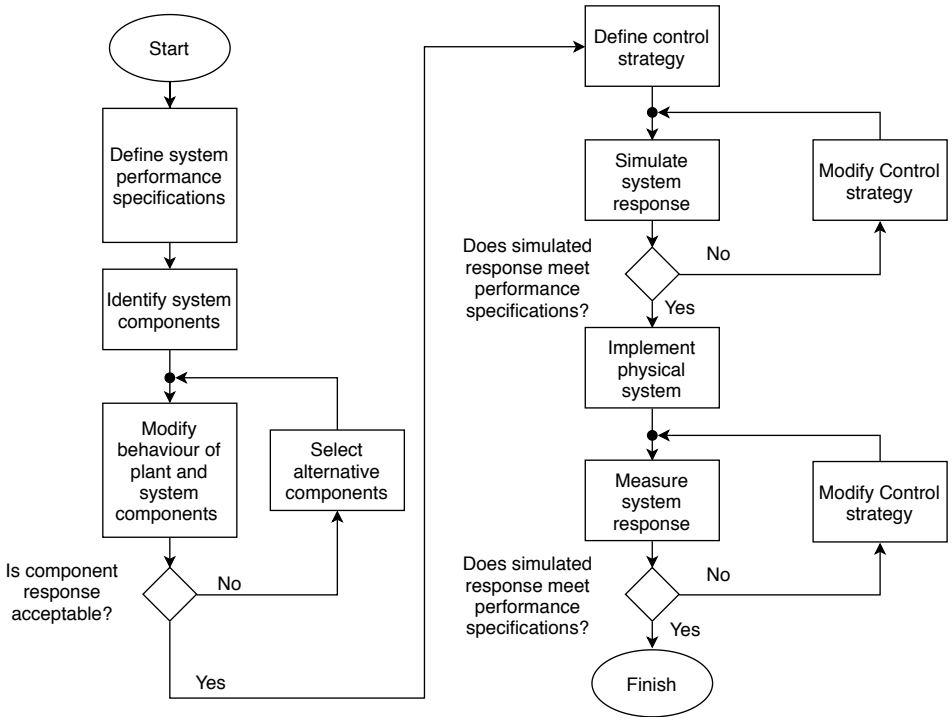

Figure 2.3: Flowchart Describing Project Stages

Single Motor

Not only is the application domain of speed control of DC motors vast but also it comes with different requirements [46]. An analogous requirement that pertains to this project is the number of motors to be implemented in the solution. For a hoist design, a single motor is ample, whereas an automated guided vehicle entails at least two but possibly four motors for finer locomotion [47], [48]. While control of the motor being a sine qua non, the number of motors is a design decision opted by engineers in accordance with the application in question. In the light of the fact that we are not directing our attention towards any specific application domain of speed control of DC motors, implementing more than one motor becomes superfluous. For this reason, a single motor is seen adequate to develop the intended project goal of speed control via buck converter.

Chapter 3

Methods and Materials

3.1 Current and Speed Sensors

The utilised current sensor, ACS723 from Sparkfun [49], works based on the Hall-effect principle and outputs an analogue voltage [50]. Its output is directly proportional to the magnetic field generated by the current with a sensitivity of $400mV/A$ that we are interested in measuring [51].

The speed sensor output is a square wave signal with r number of high pulses per rotation. Figure 3.1 illustrates how the output square wave is produced. As the magnetized wheel attached to end of the motor shaft rotates, the magnetic field subjected to the Hall-effect sensor is altered [52], [53]. As a consequence, the output is continuously switched between high and low as long as the motor spins.

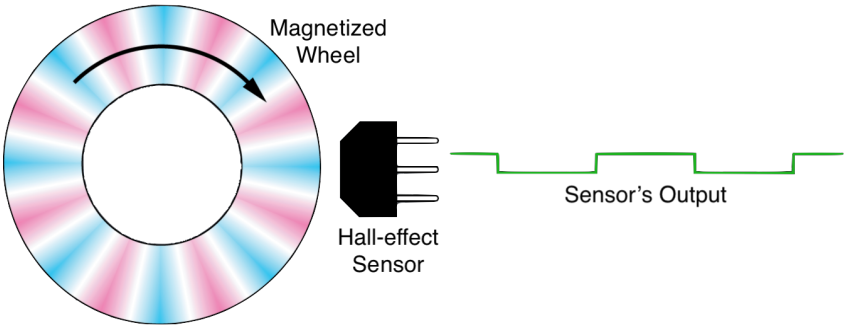

Figure 3.1: Rotating Magnetized Wheel Affecting Hall-effect Sensor Output [54]

The speed of the motor can be calculated with the following simple relation

$$\omega_m = \frac{60p}{r} \tag{3.1}$$

where ω_m is the motor speed in rpm, p is the number of pulses per second and r is the number of pulses per rotation.

3.2 Permanent Magnet DC Motor

Figure 3.2 shows the equivalent circuit for a DC motor, depicting both its electrical and mechanical sides.

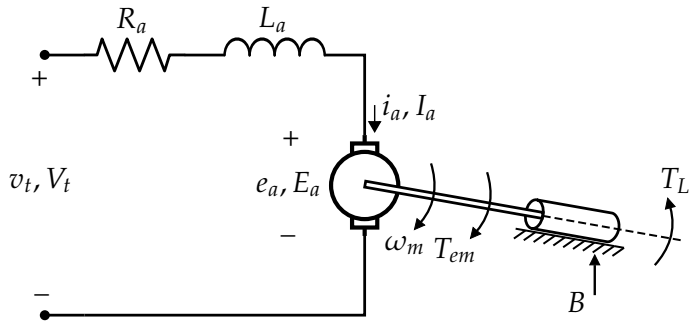

Figure 3.2: Equivalent Circuit of Permanent Magnet DC Motor

Electrical Side

An armature current i_a flows in the circuit and produces the electromagnetic torque T_{em}, imperative to the rotation of the mechanical load at an angular speed of ω_m. Consequently, a voltage, opposite to the applied voltage, called the back-emf is induced across the armature terminals in proportion to the speed. Equation 3.2 delineates the relation between the back-emf (e_a) and the motor angular speed ω_m [38], [55].

$$e_a = k_E \omega_m \tag{3.2}$$

Furthermore, in DC motors the electromagnetic torque is produced by the interaction of the field flux and the armature current, however for a PMDC motor the field flux is constant and the resulting correspondence is:

$$T_{em} = k_T i_a \tag{3.3}$$

The torque constant k_T and the voltage constant k_E are numerically equal for a DC machine, provided that the units are expressed in SI [55]. In Figure 3.2, for the electrical side, Kirchoff's voltage law yields the following equation

$$v_t = e_a + R_a i_a + L_a \frac{di_a}{dt} \tag{3.4}$$

where R_a and L_a are the armature resistance and inductance, respectively. The voltage for the inductor term is proportional to the rate of change of current and thus, under steady state, this voltage drop is zero and the equation develops into Equation 3.5

$$I_a = \frac{V_t - E_a}{R_a} \tag{3.5}$$

Using Equation 3.2, 3.3, and 3.5, the relation between the steady-state speed ω_m and the electromagnetic torque T_{em} for a given applied voltage V_t can be derived as Equation 3.6 [38]

$$\omega_m = \frac{1}{k_E}\left(V_t - \frac{R_a}{k_T}T_{em}\right) \tag{3.6}$$

The speed-torque characteristic plot from Equation 3.6 is shown in Figure 3.3, where the applied voltage $V_{t_5} > V_{t_4} > V_{t_3} > V_{t_2} > V_{t_1}$. It is apparent from the figure that, for a fixed terminal voltage, when the torque demand increases, the speed is reduced.

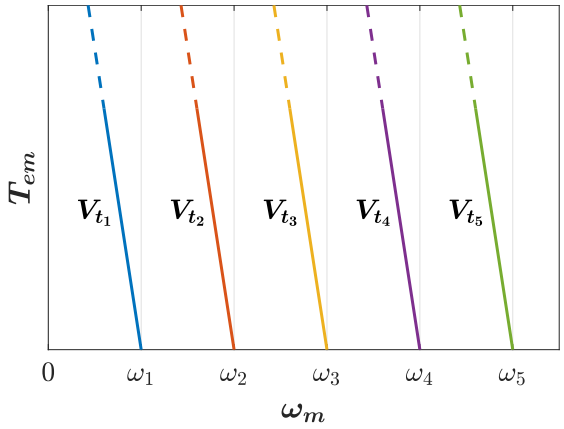

Figure 3.3: Speed-Torque Characteristics of a DC Motor

Mechanical Side

For the mechanical side of Figure 3.2, the interaction between T_{em} and load torque determines how the motor speed builds up

$$T_{em} = J\frac{d\omega_m}{dt} + B\omega_m + T_{wL}(t) \tag{3.7}$$

such that, J and B are the total equivalent inertia and damping (viscous friction), respectively, for the motor-load combination and T_{wl} is the equivalent working torque of the load [38].

3.3 Buck Converter

The circuit in Figure 3.4(a) depicts the classic topology of the buck regulator with a resisitve load. This arrangement produces an output voltage that is lower than and has the same polarity as the input voltage. The analysis of the ideal converter assumes periodic operation, perfect power transfer, steady-state conditions, zero average capacitor current and continuous inductor current (i.e., the current i_L does not reach zero during one period) [32], [43].

The waveforms associated with the signals that characterize the buck converter can be seen in Figure 3.5.

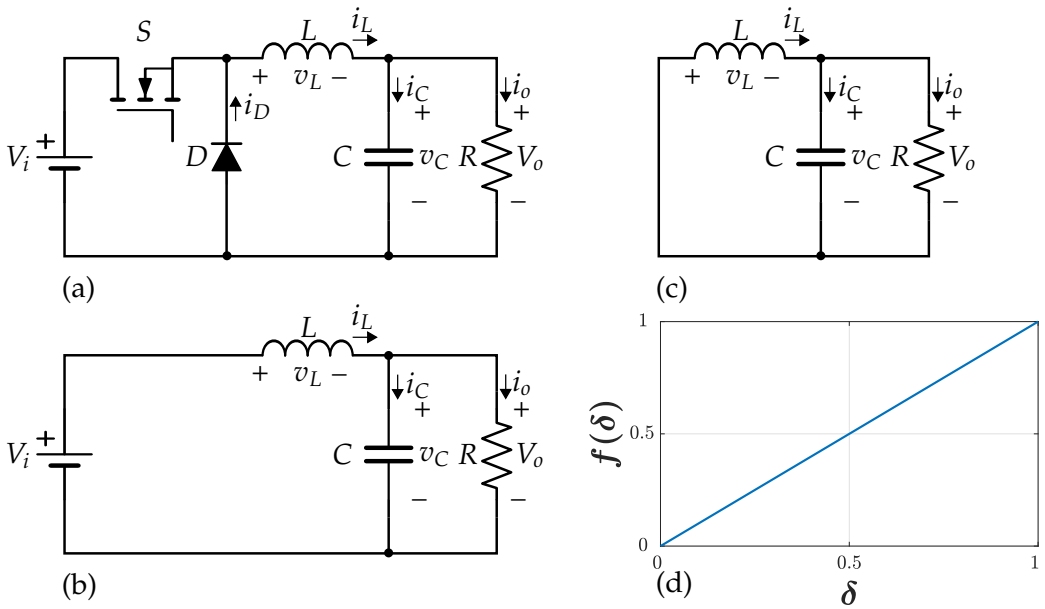

Figure 3.4: (a) DC-DC Buck Converter Circuit; (b) Circuit During On Time; (c) Circuit During Off Time; (d) Voltage Conversion Curve

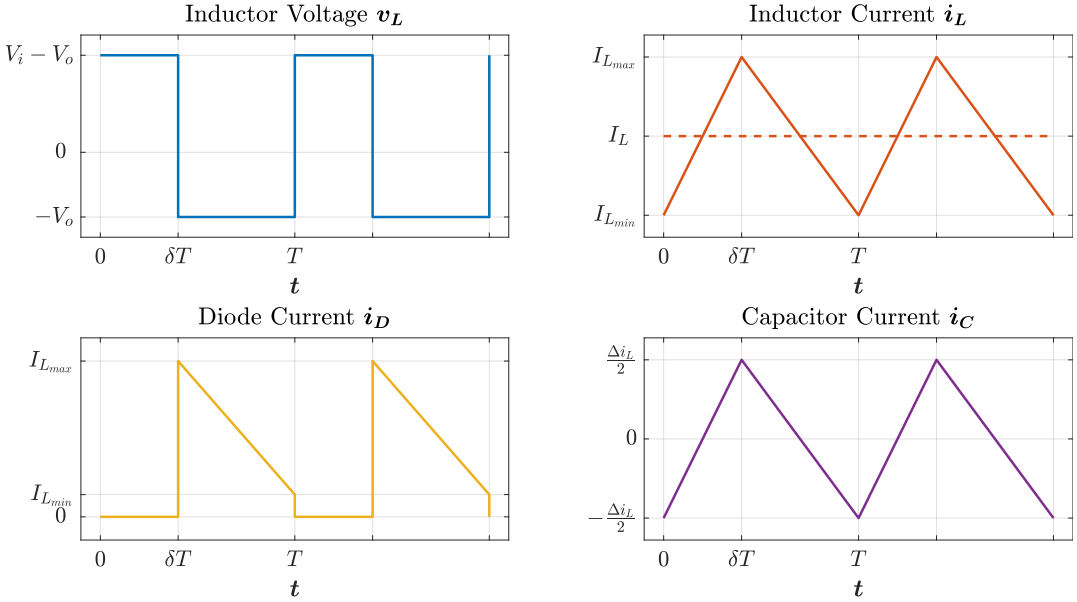

Figure 3.5: Buck Converter Waveforms

The behavior of the converter is investigated by applying Kirchhoff's Laws in the given circuit, while distinguishing between the two states of the controlled switch S, represented in Figure 3.4(a) as an enhancement-type N-channel MOSFET, switched with a frequency f. The transistor is considered on (switch closed) for a time δT, the circuit becoming the one shown in Figure 3.4(b), and off (switch open) for $(1-\delta)T$, as seen in Figure 3.4(c). We have thus defined δ as the duty cycle and T as the switching period [32], [43].

On the basis of [32] and [43], we can easily analyze the buck converter cir-

cuit. During the on and off time, respectively, KVL yields the following relations, assuming the inductor current increases and decreases in a linear fashion

$$\Delta i_{L_{closed}} = \frac{(V_i - V_o)\delta T}{L} \tag{3.8}$$

$$\Delta i_{L_{open}} = -\frac{V_o(1-\delta)T}{L} \tag{3.9}$$

Steady-state conditions impose

$$\Delta i_{L_{closed}} + \Delta i_{L_{open}} = 0 \tag{3.10}$$

resulting in

$$V_o = V_i\delta \tag{3.11}$$

Equation 3.11 shows that the ratio between the output and input voltages can be expressed as a function of the duty cycle, $f(\delta)$, whose corresponding curve is plotted in Figure 3.4(d). It is apparent that the higher δ is, the smaller the gap between input and output.

Going further with the analysis, we note that the average inductor current equals the average output current

$$I_L = I_o = \frac{V_o}{R} \tag{3.12}$$

This relation and Equation 3.9 can be subsequently used to find the maximum and minimum inductor current

$$I_{L_{max/min}} = I_L \pm \frac{\Delta i_L}{2} = \frac{V_o}{R} \pm \frac{V_o(1-\delta)T}{2L} \tag{3.13}$$

The boundary between continuous and discontinuous operation is the condition $I_{L_{min}} = 0$, which means that the minimum inductance can be found to be

$$L_{min} = \frac{(1-\delta)R}{2f} \tag{3.14}$$

Selecting a suitable capacitance can be done by defining a maximum peak-to-peak output voltage ripple and introducing it in the equation that describes the variation in the capacitor charge during the time the capacitor current is positive. We can identify this time as being a half of one period by looking at the capacitor current graph in Figure 3.5. The corresponding formula is

$$\Delta V_o = \frac{T\Delta i_L}{8C} \tag{3.15}$$

Substituting Δi_L with the right-hand side of Equation 3.9 and then dividing by V_o yields

$$C_{min} = \frac{1-\delta}{8L\frac{\Delta V_o}{V_o}f^2} \tag{3.16}$$

In the preceding equations, the switching frequency f is to be chosen by the designer, taking into consideration that a low frequency increases the value of the minimum inductance and capacitance needed to keep the performance of the converter at the desired level. On the other hand, high-speed switching auguments losses. A compromise between f, L, C and losses, if necessary, should be made [32].

3.4 Proportional–Integral–Derivative (PID)

This section explores the basic properties for the class of controllers referred by the generic name "PID controllers". Equation 3.17 shows the general equation for a PID controller in time domain, where e represents the error and u is the control signal, which is equal to the sum of the proportional term (P), integral term (I) and the derivative term (D). The controller parameters k_p, k_i, k_d are the proportional gain, integral gain and the derivative gain respectively. Taking the Laplace of Equation 3.17 emanates the transfer function for the PID controller and is given by equation 3.18. A block diagram of closed loop systems with PID controller is shown in Figure 3.6 and the ensuing subsections will present the effect of P, I and D terms on the system response [30], [56].

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de(t)}{dt} \tag{3.17}$$

$$D_c(s) = k_p + \frac{k_i}{s} + k_d s \tag{3.18}$$

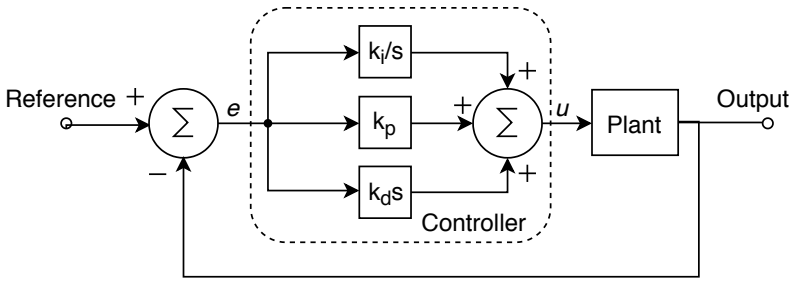

Figure 3.6: PID Controller

Proportional Control (P)

In the case of proportional control, the feedback control signal is directly proportional to the system error and thus the relation between them is instantaneous. The proportional gain affects the steady-state error of the system such that the increase in k_p decreases the steady-state error along with the increase in the system response speed. However, this increase also leads to a less overall damping and thus a more oscillatory response. A system implemented with just a P controller can have non-zero steady-state offset which cannot be eliminated no matter the gain k_p, thus, to remove this offset the integral term is added to the controller , [30], [56].

Proportional Plus Integral Control (PI)

The control signal due to I is proportional to the integral of the system error, therefore implying that the control signal produced by the same term is, at any instant of time, a summation of previous errors. The integral term servers the purpose of eliminating the steady-state error for any value of k_i, the integral gain. However, the increase in k_i increases the settling time and the overshoot of the response , [30], [56].

Proportional–Integral–Derivative Control (PID)

The oscillatory response introduced due to the integral term can be removed by adding the derivative term to the controller. For derivative, the control signal is proportional to the rate of change of the system error, thus giving a slope, resulting in an anticipatory action based on the error trends. The addition of the derivative term increases the stability, speeds up the transient response and reduces the overshoot , [30], [56].

Additional Considerations

It is clear from the previous paragraphs that the control parameters k_p, k_i and k_d hold a major significance in pursuit of the desired system response. Therefore, for the tuning of these parameters, techniques, such as manual tuning, root locus tuning or Ziegler-Nichols tuning, prove as assisting tools. Another consideration to take into account is that the ideal PID loop discussed above may need the following modification for practical implementation. First, the high frequency noise signals from the measurement output of a sensor can produce large variation in the control signal. This is due to the drawback of derivative action, since it produces a high gain for high frequency signals. To avoid this, a low-pass filter can be implemented in series with the D term. Second, the PID controller is subject to integral windup, as a result of saturation in the system output and can cause large overshoot. Therefore, an anti-windup technique (such as clamping or back calculation) needs to implemented as a precautionary measure along with the integral part of the controller [56].

3.5 Pole Placement

Pole placement is a modern control technique usually developed in state-space. Figure 3.7 illustrates the general structure of pole placement control loop. Compared to classical control design, it incorporates full state feedback, demanding the system to be observable. Full-state feedback offers complete control over the systems dynamics as the designer is fully knowledgeable about them [57]. If the system is also state controllable, the desired closed-loop response can be achieved by placing closed-loop poles at appropriate locations via a suitable state feedback gain matrix, $\boldsymbol{K_c}$. Choosing the desired pole locations is done by considering time and/or frequency-domain requirements such as damping ratio, settling time, natural frequency and so on [58].

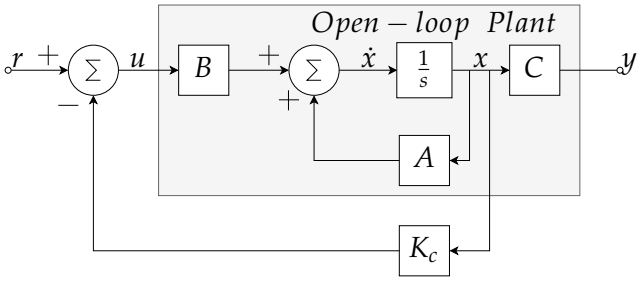

Figure 3.7: Pole Placement Control Loop Structure

As it is also seen in Figure 3.7, the following linear time-invariant state-space representation is considered for any modern control theory application in this report.

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}u \\ y &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{3.19}$$

Before explaining how pole placement technique works, two important key concepts are needed to be covered. The first key concept, controllability, states that the system can reach any linear combination of its states within its state-space in a finite amount of time [59]. A system is controllable if the controllability matrix, $\boldsymbol{M_C}$, has full row rank. The controllability matrix is computed as

$$\boldsymbol{M_C} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A^2B} & \cdots & \boldsymbol{A^{n-1}B} \end{bmatrix} \tag{3.20}$$

Observability indicates that all states can be known from the output of the system. A system is observable if the observability matrix, $\boldsymbol{M_O}$, has full column rank. The observability matrix is computed as

$$\boldsymbol{M_O} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{C^2A} \\ \vdots \\ \boldsymbol{C^{n-1}A} \end{bmatrix} \tag{3.21}$$

The following steps are taken to determine the effect of feedback gain matrix $\boldsymbol{K_c}$ over the location of closed-loop poles [60]. First, the input to the system u is calculated from Figure 3.7:

$$u = r - \boldsymbol{K_c}\boldsymbol{x} \tag{3.22}$$

When the loop is closed and Equation 3.22 is substituted into Equation 3.19 as shown in Equation 3.23, a new closed loop state-space representation, given in Equation 3.24, is obtained [60].

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}(r - \boldsymbol{K_c}\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}r - \boldsymbol{B}\boldsymbol{K_c}\boldsymbol{x} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K_c})\boldsymbol{x} + \boldsymbol{B}r \tag{3.23}$$

$$\begin{aligned} \dot{\boldsymbol{x}} &= (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K_c})\boldsymbol{x} + \boldsymbol{B}r \\ y &= \boldsymbol{C}\boldsymbol{x} \end{aligned} \tag{3.24}$$

The roots of the closed-loop system can be directly calculated from the characteristic equation:

$$\left[s\boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{K_c})\right] = 0 \tag{3.25}$$

By choosing appropriate entries for $\boldsymbol{K_c}$, poles can be moved to desired location on the pole-zero map. However, this control structure will not be enough to provide a zero steady-state error per se, as it is analogous to a proportional controller. One way of eliminating steady-state error is adding integral control with its corresponding gain, $\boldsymbol{K_i}$. Figure 3.8 illustrates the general structure of the pole placement control loop with integral action in place [61].

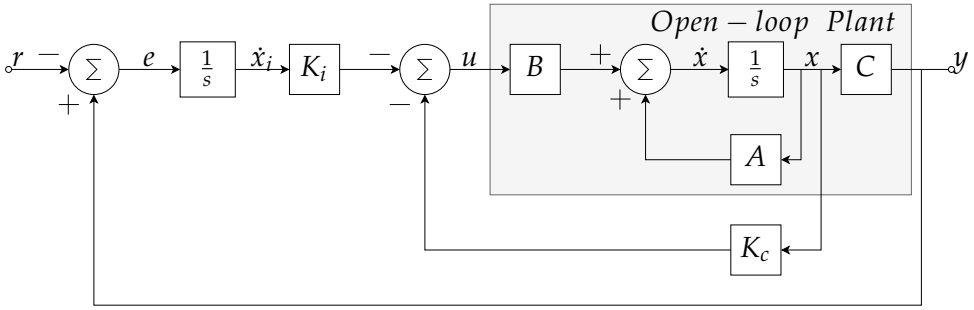

Figure 3.8: Pole Placement Control Loop Structure With Integral Action

Increasing the system type by one with an integrator requires the inclusion of a new state in the model. This new state is labelled x_i and it is defined as follows:

$$\begin{aligned} x_i &= \int e dt = \int (y - r) dt = \int (\boldsymbol{C}\boldsymbol{x} - r) dt \\ \dot{x}_i &= \boldsymbol{C}\boldsymbol{x} - r \end{aligned} \tag{3.26}$$

Subsequently, the new state-space representation becomes:

$$\begin{aligned} \begin{bmatrix} \dot{\boldsymbol{x}} \\ \dot{x}_i \end{bmatrix} &= \begin{bmatrix} \boldsymbol{A} & 0 \\ \boldsymbol{C} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ x_i \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ -1 \end{bmatrix} r \\ y &= \begin{bmatrix} \boldsymbol{C} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ x_i \end{bmatrix} \end{aligned} \tag{3.27}$$

Equation 3.27 represents the open-loop system. The new $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ matrices, the matrix multiplied with r and the new state vector will be referred to as $\boldsymbol{A_a}$, $\boldsymbol{B_a}$, $\boldsymbol{C_a}$, $\boldsymbol{B_r}$ and $\boldsymbol{x_a}$ respectively, giving the new open-loop state-space representation:

$$\begin{aligned} \dot{\boldsymbol{x}}_{\boldsymbol{a}} &= \boldsymbol{A_a}\boldsymbol{x_a} + \boldsymbol{B_a}u + \boldsymbol{B_r}r \\ y &= \boldsymbol{C_a}\boldsymbol{x_a} \end{aligned} \tag{3.28}$$

The closed-loop state-space representation is calculated the same way as before, shown in Equation 3.29. First, u is derived based on Figure 3.8 and substituted in Equation 3.28.

$$u = -\boldsymbol{K_c}\boldsymbol{x} - \boldsymbol{K_i}\boldsymbol{x_i} = -\boldsymbol{K_a}\boldsymbol{x_a} \quad \text{where} \quad \boldsymbol{K_a} = \begin{bmatrix}\boldsymbol{K_c} & \boldsymbol{K_i}\end{bmatrix}$$
$$\dot{\boldsymbol{x}}_{\boldsymbol{a}} = \boldsymbol{A_a}\boldsymbol{x_a} + \boldsymbol{B_a}u + \boldsymbol{B_r}r = \boldsymbol{A_a}\boldsymbol{x_a} - \boldsymbol{B_a}\boldsymbol{K_a}\boldsymbol{x_a} + \boldsymbol{B_r}r = (\boldsymbol{A_a} - \boldsymbol{B_a}\boldsymbol{K_a})\boldsymbol{x_a} + \boldsymbol{B_r}r \quad (3.29)$$

According to the above calculations, the closed-loop state-space representation takes the form:

$$\begin{aligned}\dot{\boldsymbol{x}}_{\boldsymbol{a}} &= (\boldsymbol{A_a} - \boldsymbol{B_a}\boldsymbol{K_a})\boldsymbol{x_a} + \boldsymbol{B_r}r \\ y &= \boldsymbol{C_a}\boldsymbol{x_a}\end{aligned} \quad (3.30)$$

The roots of the closed-loop system can be directly calculated from the characteristic equation:

$$\left[s\boldsymbol{I} - (\boldsymbol{A_a} - \boldsymbol{B_a}\boldsymbol{K_a})\right] = 0 \quad (3.31)$$

By choosing appropriate gain values for $\boldsymbol{K_a}$, poles can be moved to desired location on the pole-zero map. With integral control added, steady-state error is also eliminated.

3.6 Linear Quadratic Regulator (LQR)

LQR is an optimal modern control technique based on state-space representation that is very similar to pole placement. It also includes full-state feedback and incorporates the same control loop structure. Both methods differ from each other in the way that feedback gain matrix, $\boldsymbol{K_a}$, is calculated. Compared to pole placement where $\boldsymbol{K_a}$ is derived based on desired pole locations, LQR implements a cost function that computes the optimal $\boldsymbol{K_a}$ values based on performance and effort [59].

$$J(\boldsymbol{x}, \boldsymbol{u}) = \int_0^\infty (\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u})dt \quad (3.32)$$

The version of cost function that will be used for the project is given in Equation 3.32. The $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices are respectively used to adjust performance and actuator effort. Performance is evaluated based on state vector and the $\boldsymbol{Q}$ matrix is used to penalize bad performance. Furthermore, LQR is unconstrained. It assumes $\boldsymbol{u}$ can have any value from $-\infty$ to $+\infty$. Therefore, u it is also introduced into the cost function and constrained. Actuator effort is evaluated based on $\boldsymbol{u}$ and the $\boldsymbol{R}$ matrix is used to penalize actuator effort [62].

The values are squared to compensate for negative values while calculating the cost. This manipulation turns the cost function $J(\boldsymbol{x}, \boldsymbol{u})$ into a quadratic function which is convex and has a definite minimum value. This will remain true even when it is exposed to linear dynamics of any system. Taking the integral of this function over infinite horizon gives the area under the curve which is a measure of how quickly it is converging to the desired state-space. Considering infinite horizon is important to guarantee stability in the long run [59], [62].

The $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices are usually diagonal, with each entry corresponding to a specific state/input. Higher $\boldsymbol{Q}$ matrix entries indicate lower error for their corresponding state. Similarly, higher $\boldsymbol{R}$ matrix entries mean penalizing corresponding actuator more and possibly slowing down the system.

By tuning the entries of $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices, desired performance and cost values can be obtained. For instance, if we are aiming for a very little error for a specific state, we can achieve it by increasing its corresponding entry in the $\boldsymbol{Q}$ matrix. Likewise, if one actuator consumes more energy, it can be penalized via the $\boldsymbol{R}$ matrix.

Bryson's Rule

Even though the design of $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices is usually a trial and error procedure without much literature available on, Bryson's rule provide the designer with initial values. According to the mentioned rule, the initial diagonal values of $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen as follows:

$$\begin{aligned} \boldsymbol{Q}[\boldsymbol{i},\boldsymbol{i}] &= 1/\text{maximum acceptable value of } x_i^2 \\ \boldsymbol{R}[\boldsymbol{i},\boldsymbol{i}] &= 1/\text{maximum acceptable value of } u_i^2 \end{aligned} \tag{3.33}$$

The matrices are later altered to fine-tune the performance and actuator effort in an acceptable manner.

Chapter 4

Modeling

4.1 PMDC Motor Modeling

In order to maintain the speed of the DC motor constant, a controller needs to be designed and to do so the transfer function of the motor is critical. It labels the dynamic input-output relation for the DC motor and is later combined with the rest of the drive's transfer functions. Therefore, the present section concenters on determining the transfer function for a PMDC motor.

In practicality, a motor is limited by its rated current and voltage, ergo its linear model, i.e.: the transfer function is only valid for small changes within the region where the motor current is not bounded. Accordingly, for the analysis, only the small signal dynamics of the motor are concerned around their steady state values, thus the equations 3.2, 3.3, 3.4, and 3.7 from Section 3.2 are written in terms of small deviations [38].

$$\Delta e_a = k_E \Delta \omega_m \tag{4.1}$$

$$\Delta T_{em} = k_T \Delta i_a \tag{4.2}$$

$$\Delta v_t = \Delta e_a + R_a \Delta i_a + L_a \frac{d\Delta i_a}{dt} \tag{4.3}$$

$$\Delta T_{em} = J \frac{d\Delta \omega_m}{dt} + B\Delta \omega_m + \Delta T_{WL} \tag{4.4}$$

Taking the Laplace transform of the equations above, we acquire the resulting equations in s-domain.

$$E_a(s) = k_E \Omega_m(s) \tag{4.5}$$

$$T_{em}(s) = k_T I_a(s) \tag{4.6}$$

$$V_t(s) = E_a(s) + (L_a + sR_a) I_a(s) \tag{4.7}$$

$$T_{em}(s) = (B + sJ)\Omega_m(s) + T_{WL}(s) \tag{4.8}$$

These Laplace equations, when solving for Ω_m, yield a steady-state relationship between change in motor speed to a step change in the motor inputs which are the voltage and load torque. Figure 4.1 also represents the aforementioned relationship in block diagram format [38].

$$\Omega_m(s) = \frac{k_T}{k_T k_E + (R_a + sL_a)(B + sJ)} V_t(s) - \frac{(R_a + sL_a)}{k_T k_E + (R_a + sL_a)(B + sJ)} T_{WL}(s) \tag{4.9}$$

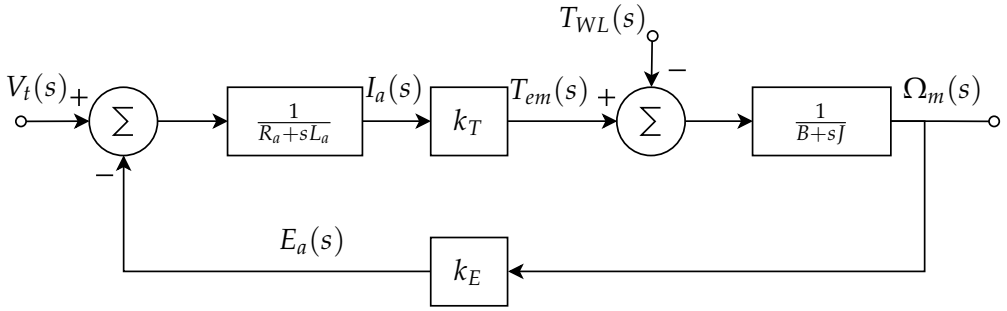

Figure 4.1: Block Diagram of DC Motor

If only one input is applied at a time, Equation 4.9 develops into two transfer functions. We are only interested in the relationship between voltage and angular speed, given by Equation 4.10 [38].

$$\frac{\Omega_m(s)}{V_t(s)} = \frac{k_T}{JL_a s^2 + (BL_a + JR_a)s + k_T k_E + BR_a}\bigg|_{T_{WL}(s)=0} \tag{4.10}$$

Ignoring the load torque disturbance and performing some block diagram manipulations, the structure in Figure 4.1 becomes the one shown in Figure 4.2. This structure will further be utilized for the development of speed and current cascade PID control.

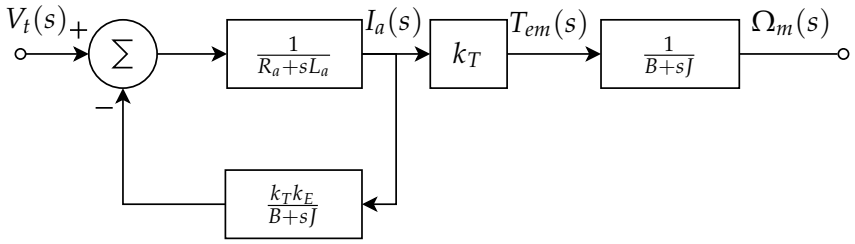

Figure 4.2: Modified Block Diagram of DC Motor

4.2 PMDC Motor Parameter Identification

Given the fact that the datasheets of the DC motor and speed sensor utilized in this project did not offer sufficient details regarding the characteristics of the actuator and the sensor, a series of experiments was conducted in order to obtain the necessary constant parameters for modeling. The steps that were taken in finding these constants, along with the experimental results, are delineated in this section.

Preamble

The experiments have been initially performed by running the motor under no load at different voltages, starting from $2V$ and reaching $12V$ ($12V$ is the rated voltage for the utilized DC machine [40]), using $1V$ increments. When the motor viscous friction parameter B was identified, a substantial variation has been noticed and thus, taking an average of the obtained results and using it in a model would cause important discrepancies between simulation and reality. This observation is reflected by Figure 4.3, in which the dashed line represents the average viscous friction coefficient.

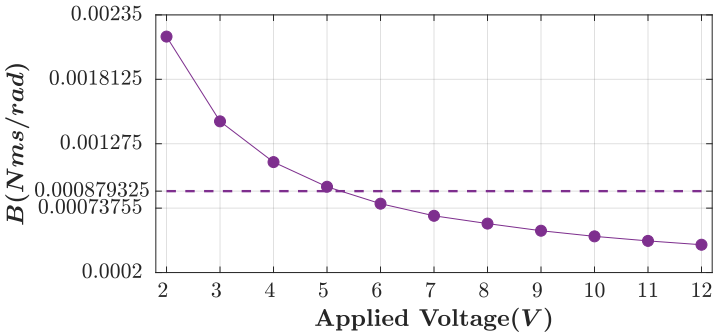

Figure 4.3: Motor Viscous Friction Constant Experimental Results

From the plot in Figure 4.3 it is apparent that eliminating the data points corresponding to low voltages $(2-4V)$ would produce an average that better describes the actual behaviour of the machine. In view of this, for parameter identification purposes, only the voltage interval $[5, 12V]$ has been utilized.

Speed Sensor Resolution

The resolution of the Hall-effect sensor is represented by r, the number of pulses produced per rotation. Looking at Equation 3.1, it is obvious that this resolution can be found by measuring the motor angular speed ω_m and the frequency (number of pulses per second) p of the square wave generated by the sensor. The angular speed was measured using a handheld digital tachometer. The sensor output was monitored with an oscilloscope that provided the frequency measurement. Additionally, the digital signal coming from the sensor was fed to Arduino and a simple program was written to achieve an alternative frequency monitoring technique, thus validating the oscilloscope readings. The sequential measurements are plotted and shown in Figure 4.4.

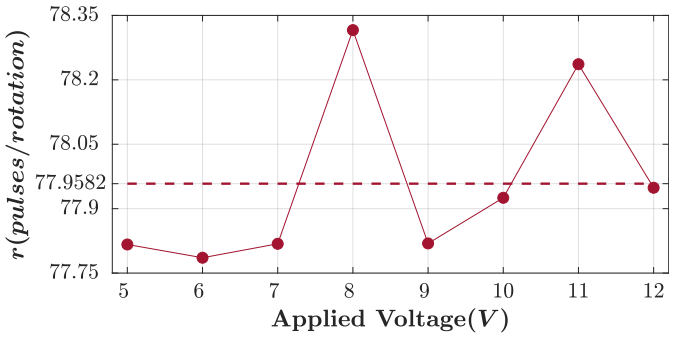

Figure 4.4: Speed Sensor Resolution Experimental Results

The dashed line shows the average value of the measurements, which will be hereafter used in motor speed calculations. We thus have

$$r = 77.9582 pulses/rotation$$

Armature Resistance and Inductance

The physical structure of a PMDC motor includes a commutator ring which only allows one rotor winding to be connected to the motor terminals for a given rotor position. The resistance, R_a, and the inductance, L_a, of the motor armature, have been approximated by taking measurements with a multimeter for 10 distinct rotor positions and then averaging the obtained values. The results are illustrated in Figures 4.5 and 4.6.

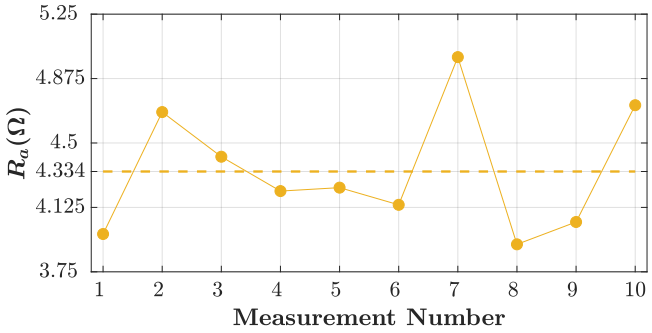

Figure 4.5: Armature Resistance Experimental Results

The average armature resistance and armature inductance are represented by the dashed lines in the figures above. The corresponding values are:

$$R_a = 4.334\Omega$$

$$L_a = 3.334mH$$

Back-emf and Torque Constants

Two different experiments are conducted to calculate the back-emf constant k_E of the DC motor. Since the torque constant k_T is equal to k_E, no additional tests are

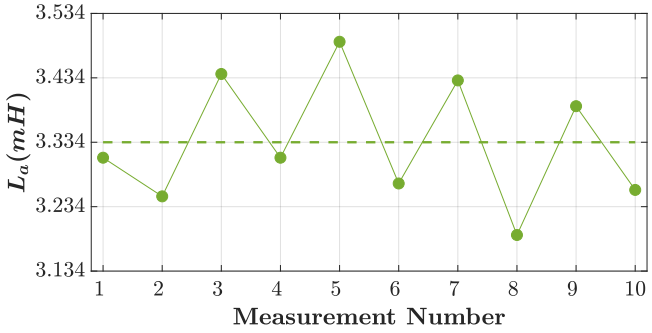

Figure 4.6: Armature Inductance Experimental Results

needed for its identification. The specific procedures followed for each experiment are described below.

In the first experiment, each steady voltage input to the motor resulted, after a period of transient, in an unchanging speed ω_m and current I_a, which were captured using a digital tachometer and a multimeter, respectively. The equation that gives k_E is derived as follows. Combining Equations 3.2 and 3.5, we obtain:

$$I_a = \frac{V_t - k_E w_m}{R_a} \tag{4.11}$$

Solving for k_E yields:

$$k_E = \frac{V_t - R_a I_a}{\omega_m} \tag{4.12}$$

By inserting the measured values of I_a and ω_m into Equation 4.12, together with the corresponding voltage and the already-computed R_a, we can calculate k_E. Figure 4.7 plots the determined k_E values for the applied set of voltages as well as the average of these values.

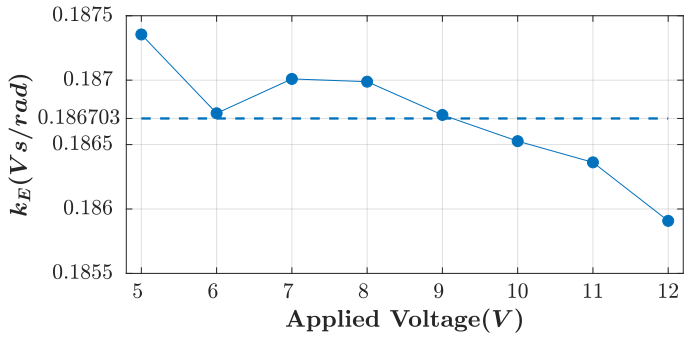

Figure 4.7: Back-emf Constant Experimental Results (1)

Figure 4.7 shows consistent experimental results. The average back-emf constant is represented by the dashed line in the figure. Its corresponding values is:

$$k_{E_1} = 0.1867 Vs/rad$$

In the second experiment, the motor shaft is rotated using a second DC motor by means of a 3D-printed shaft coupling. This time, the set of constant voltages

has been applied to the additional machine. The speed of the DC motor on which the identification is performed, ω_m, was measured with a digital tachometer. The steady-state voltage that appeared at the motor terminals due to the rotor conductors moving in a magnetic field (the back-emf), E_a, was captured with a multimeter. Equation 3.2 has been used to calculate the k_E for each back-emf-speed pair. The individual results and their mean value are illustrated in Figure 4.8.

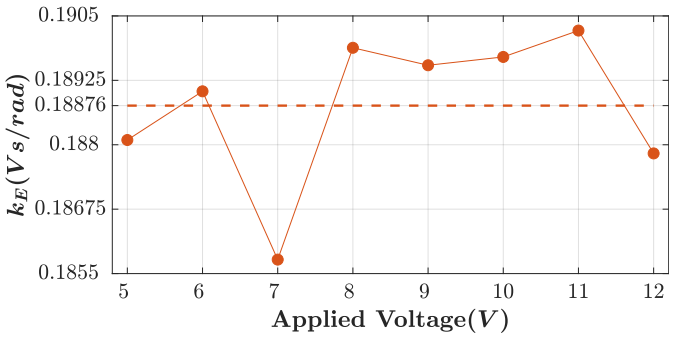

Figure 4.8: Back-emf Constant Experimental Results (2)

In this case, the average back-emf constant k_E is

$$k_{E_2} = 0.1888 Vs/rad$$

Comparing the results of the two experiments, it is seen that the average k_E values differ by $0.0021 \frac{Vs}{rad}$. Hence, the experiments are considered to be successful and the k_E constant to be used in the modeling and simulation of the motor is estimated by taking the mean of the average results of the two experiments.

$$k_E = \frac{k_{E_1} + k_{E_2}}{2} = \frac{0.1867 + 0.1888}{2} = 0.1877 Vs/rad$$

Consequently, k_T is also determined as being

$$k_T = k_E = 0.1877 Nm/A$$

Viscous Friction Constant

When the motor reaches constant speed under no load, Equation 3.7 is still going to hold true and it may be solved for B.

$$B = \frac{T_{em}}{\omega_m} \tag{4.13}$$

Furthermore, using Equation 3.3 we have

$$B = \frac{k_T I_a}{\omega_m} \tag{4.14}$$

The set of constant voltages has been applied to the motor, while measuring the steady-state current I_a and the speed ω_m, as in the previous experiments, for each voltage value. By inserting these values into Equation 4.14 together with the computed k_T, different values of B were found, as shown in Figure 4.9.

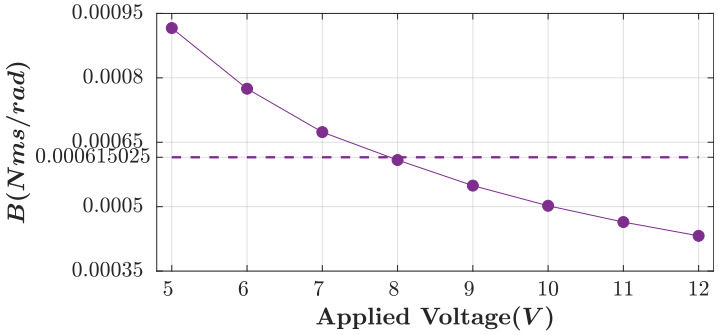

Figure 4.9: Motor Viscous Friction Constant Experimental Results

The average motor viscous friction constant B is represented by the dashed line in the figure. Its corresponding value is:

$$B = 6.1502 \cdot 10^{-4} Nms/rad$$

Rotor Inertia

The rotor inertia J can be calculated from the voltage-to-speed DC motor transfer function described by Equation 4.10. Considering a step input voltage in the form $V_t(s) = \frac{V_t}{s}$, the expression for the angular speed $\Omega(s)$ is

$$\Omega_m(s) = \frac{V_t k_T}{JL_a s^3 + (BL_a + JR_a)s^2 + (BR_a + k_E k_T)s} \tag{4.15}$$

Taking the inverse Laplace transform on both sides of Equation 4.15 yields an expression for the instantaneous angular speed in time domain. Knowing the speed of the motor and the corresponding time at a specific point during the transient period, one can find J with the aid of MATLAB.

The experiment for finding the motor inertia is based on the method described above. All the constant parameters in Equation 4.15, apart form J_m, of course, have been previously identified. To measure the speed during the transient period, the speed sensor signal is read by Arduino and the speed is deduced using the known sensor resolution. This is performed for the integer voltage values in the chosen interval. The experimental results are shown in Figure 4.10.

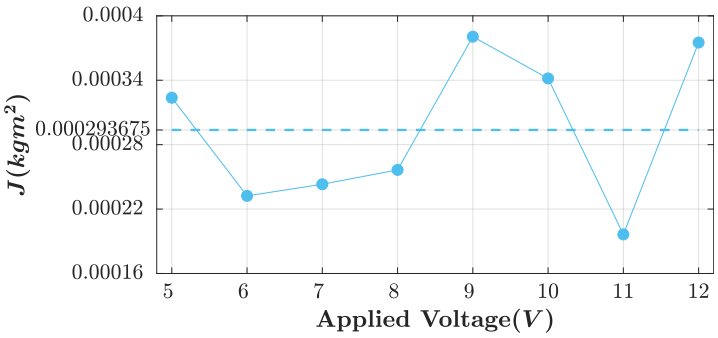

Figure 4.10: Motor Inertia Experimental Results

The mean inertia is marked by the dashed line, which indicates:

$$J = 2.9367 \cdot 10^{-4} kgm^2$$

The sole objective of the experiments in this section was, in the end, to model our PMDC motor. The transfer function giving the mathematical LTI model of the motor for voltage input and speed output is:

$$\frac{\Omega_m(s)}{V_t(s)} = \frac{1.917 \cdot 10^5}{(s+1271.6)(s+30.4)}\bigg|_{T_{WL}(s)=0} \tag{4.16}$$

4.3 Buck Converter Design

The buck power converter was essentially designed employing the Equation 3.14 and 3.16 discussed earlier in Section 3.3, since these relations lead to the inductance L and capacitance C. The former parameter dictates whether the converter operates in the continuous current region, whereas the latter decisively influences the output voltage ripple. Even though the equations hold solely for resistive loads and a DC motor stands as a classic example of an inductive load, one may, for design purposes only, disregard the effect of motor inductance on the basis that the output voltage (which coincides with the motor input voltage) is of a pure DC nature. The constant resistance-voltage source combination can then be replaced, as far as the converter is concerned, with a variable resistance, dependent on the motor winding current and therefore on the torque requirements of the mechanical load attached to the motor shaft. The design approach taken in this project consists of the fact that a robust switching regulator interfacing the DC machine should be able to maintain continuous current and satisfy the desired ripple constraint for the two "worst-case scenarios": minimum ad maximum output resistance. This slant on converter design was observed in [63].

We begin the design procedure by defining an operation boundary for the motor input voltage. Following the observations made in the beginning of Section 4.2, the interval $[5, 12V]$ was chosen. To find the minimum and maximum converter output resistance, the simple formulas

$$R_{min} = \frac{V_{o_{min}}}{I_{a_{max}}} \tag{4.17}$$

$$R_{max} = \frac{V_{o_{max}}}{I_{a_{min}}} \tag{4.18}$$

were utilized, where the minimum and maximum output current were considered to be the no-load and stall currents specified by the manufacturer as having the magnitudes of $0.18A$ and $2.5A$, respectively [40]. When inserting the numerical values into the equations above, the results are

$$R_{min} = \frac{5}{2.5} = 2\Omega \qquad R_{max} = \frac{12}{0.18} = 66.67\Omega$$

Using the extremities of the defined output voltage range, the corresponding duty cycle limits were found by isolating δ in Equation 3.11, as follows

$$\delta_{min} = \frac{V_{o_{min}}}{V_i} \tag{4.19}$$

$$\delta_{max} = \frac{V_{o_{max}}}{V_i} \tag{4.20}$$

and, after plugging the numbers, we obtain

$$\delta_{min} = \frac{5}{12} = 0.4167 = 41.67\% \qquad \delta_{max} = \frac{12}{12} = 1 = 100\%$$

These parameters were subsequently used to find a suitable inductor and capacitor that would guarantee the demanded power converter behaviour under any circumstances that fall within the defined operating range. By inspection of Equations 3.14 and 3.16, we notice that that the value of the minimum inductance is directly proportional to the output resistance. For this reason, when selecting an inductor, the maximum resistance R_{max} was taken into consideration. The relation between capacitance and resistance was not considered since C resulted from the computed inductance. In a relatively similar fashion we can examine the dependence of L and C on the duty cycle. Both values increase as δ decreases. Mathematically, these observations can be summarized by writing

$$L_{min} = \frac{(1-\delta_{min})R_{max}}{2f} \qquad (4.21) \qquad C_{min} = \frac{1-\delta_{min}}{8L\frac{\Delta V_o}{V_o}f^2} \qquad (4.22)$$

For a switching frequency f of $20kHz$ and a target output voltage ripple $\frac{\Delta V_o}{V_o}$ of 0.01%, substituting numerical values for the known parameters once again results in

$$L_{min} = \frac{0.5833 \cdot 66.67}{40000} = 9.72 \cdot 10^{-4} H = 0.972mH$$

$$C_{min} = \frac{0.5833 \cdot 100}{8 \cdot 9.72 \cdot 10^{-4} \cdot 0.01 \cdot 20000^2} = 1.87 \cdot 10^{-3} F = 1.87mF$$

These are the absolute minimum values for which the power converter would behave as desired. A general design rule is to take a safety margin when it comes to the inductor value [43]. Increasing L leads to a decrease in C, as it can be observed from Equation 3.16. Notwithstanding, a high capacitance does not negatively affect the circuit performance. Suitable components were directly available from the university electronics laboratory. The chosen values were the following:

$$L = 3.65mH \qquad C = 2.2mF$$

In terms of switch selection, the main requirements were satisfactory speed and current ratings as well as low power losses. For the controlled switch, the IRFZ44VZ N-channel enhancement mode power MOSFET was chosen. According to the datasheet, the part it is well-suited for applications in which reliability and efficiency are pivotal, as it provides fast switching and ultra-low on-resistance. The maximum values of the most relevant transistor characteristics are: gate threshold voltage V_{GS} of $4V$, drain-to-source on-resistance $R_{DS_{ON}}$ of $12m\Omega$ and continuous drain current I_D of $57A$ [64]. The selected uncontrolled switch, i.e.: the diode, was the power rectifier MUR805G, since its datasheet specifies the possibility of utilization in switching power supplies. Some of the main features of the diode include: forward voltage V_F of $0.975V$, average rectified forward current I_F 0f $8A$, peak reverse voltage V_R of $50V$ and reverse recovery time t_{rr} of $35ns$ [65]. Heat sinks were added to the two switches to prevent overheating and thus malfunctioning, although no detailed analysis was performed regarding this aspect.

4.3.1 Buck Converter Simulation and Testing

With the establishment of components, the next step towards concluding the design procedure is to corroborate the operating idiosyncrasies of the power converter, ensuring their compliance with the project requirements. The verification is pursued through circuit simulation in SIMULINK and subsequently through the real converter circuit. The simulation circuit includes the PMDC motor as load and along with the conventional components also considers the electrical characteristics such as on-resistance, to obtain results that are comparable to the actual circuit. The SIMULINK buck converter circuit is shown in Figure A.1 in Appendix A.

First, the circuit is simulated with the maximum duty cycle to examine the converter output voltage, which stabilizes to a value of $11.86V$ as compared to the expected $12V$. This distinction is an attribute of the power losses in the non-ideal components. Likewise, for the real circuit the output is $11.74V$. Moreover, the condition with the foremost importance is operation in continuous-conduction mode (CCM) during steady state. The lowest inductor current is at the minimum permissible duty cycle. Therefore, to warrant CCM operation in the desired duty cycle range (41.67%-100%), the steady state inductor current at minimum duty cycle should always remain positive. Accordingly, the simulated inductor current waveform for 41.67% duty cycle is plotted in Figure 4.11. The figure also shows a zoomed-in version of the waveform. Although the inductor current reaches negative magnitudes during the transient, the CCM only concerns the steady state. Subsequently, during steady state, the smallest inductor current is $50mA$, proving the analysis for continuous-conduction mode, correct. It is further mandated, for minimum duty cycle, by the inductor current of the real buck converter circuit. The oscilloscope plot of current in regard to the actual circuit is illustrated in Figure 4.12. The current in this case has a margin of $75mA$ between continuous and discontinuous conduction, thus still preserving the CCM operation. Furthermore, these results ratify the procedure of replacing the constant resistance-voltage source combination with a variable resistance considering the design process.

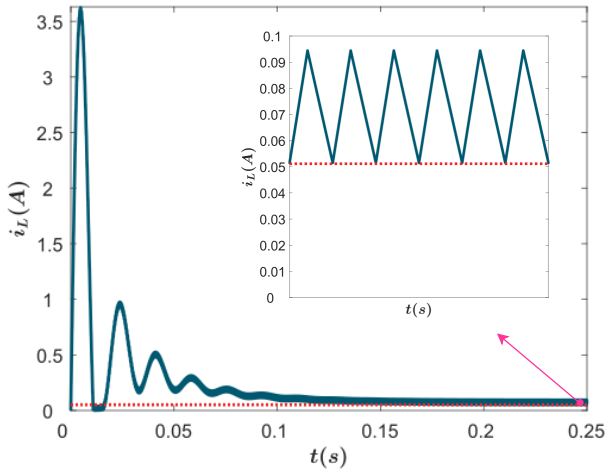

Figure 4.11: Inductor Current for Minimum Duty Cycle

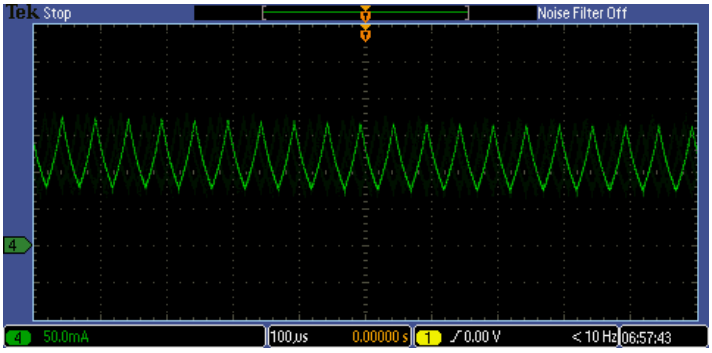

Figure 4.12: Oscilloscope Wave Forms Illustrating the Inductor Current of Buck Converter

The scrutinization of duty cycle to output voltage behavior for the actual buck converter circuit is conducted through comparison with the linear relation in Equation3.11. In Figure 4.13, the data from the actual circuit (designated by the blue line) relatively emulates the ideal relation (designated by the red line). The perturbations in the blue plot ascribe to the ineludible non-ideal conditions of the real setup. Notwithstanding, the behavior of the actual circuit can still be closely approximated with a line of gradient 12. Shifting the focus to the efficiency of the actual circuit, predictably, the efficiency of the converter is directly proportional to the duty cycle. The highest efficiency of 96.90% is observed at 100% duty ratio but the average efficiency descends to 80.92%. With this section, we conclude the design procedure, validated by the evaluation which infers the satisfactory compliance of the buck converter characteristics with the project requirements.

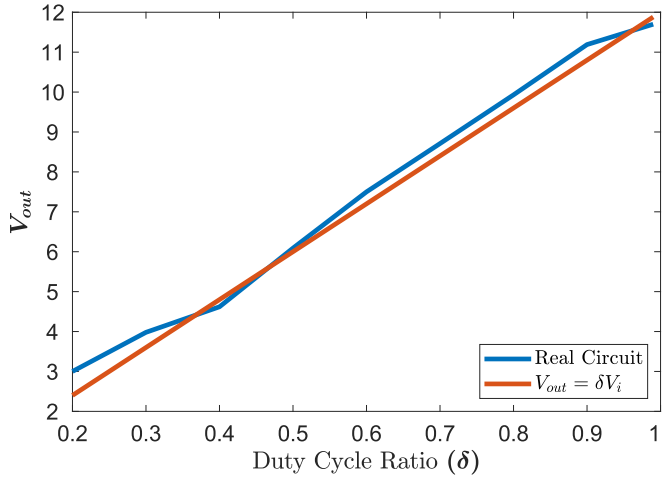

Figure 4.13: Output Voltage Vs Duty Cycle for Actual Converter Circuit and Equation 3.11

4.4 Buck Converter Modelling

The last section (Section 4.3) concluded the design procedure for the buck converter, finalizing the electrical component values. As a natural succession to it, the present section focuses on modelling of the converter, exploring the modelling

procedure which ultimately aims towards obtaining the converter's averaged state-space model.

The operation of a buck converter can be divided into two distinct circuit states. One, where the switch is "ON" and the other corresponding to switch "OFF" condition. A third circuit state also exists during discontinuous mode but the converter in question here is designed to operate solely in continuous-conduction mode, thus only the first two circuit states are considered for the analysis [38].

For both operating modes, differential equations with state variables are extracted. Generally, a system's state variables are associated with the storage of energy, thus for a buck converter they are the inductor current (i_L) and the capacitor voltage (v_t). Furthermore, the load for the converter is a PMDC motor with the desired output being the angular velocity. This leads to the remaining state variables for the combined buck converter and motor load configuration (shown in Figure 4.14), which are armature current (i_a) and the angular velocity (ω_m) [66].With the state variables known, the differential equations can be identified as follows:

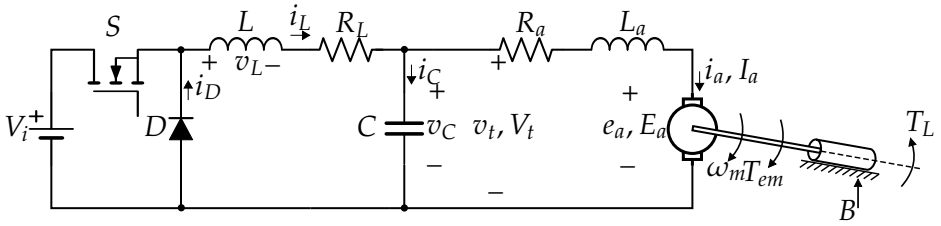

Figure 4.14: Buck Converter With PMDC Motor Load

Switch "ON" Mode

Based on the equivalent circuit (in Figure 4.15), we obtain the following equations for inductor current and capacitor voltage using KVL and KCL [67].

$$\frac{di_L}{dt} = \frac{1}{L}(V_i - i_L R_L - v_t) \tag{4.23}$$

$$\frac{dv_t}{dt} = \frac{1}{C}(i_L - i_a) \tag{4.24}$$

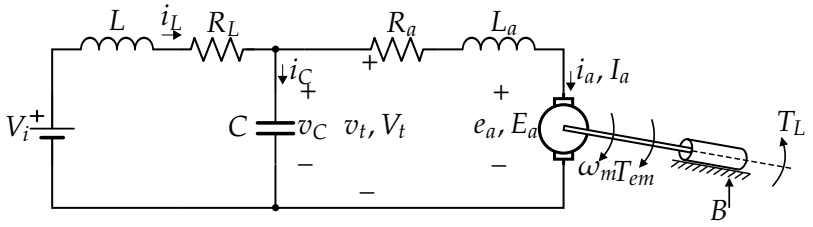

Figure 4.15: Buck Converter With Motor Load (On Mode)

Switch "OFF" Mode

Following the same procedure for the equivalent circuit shown in Figure 4.16, yields the equations shown below.

$$\frac{di_L}{dt} = \frac{-v_t - i_L R_L}{L} \tag{4.25}$$

$$\frac{dv_t}{dt} = \frac{1}{C}(i_L - i_a) \tag{4.26}$$

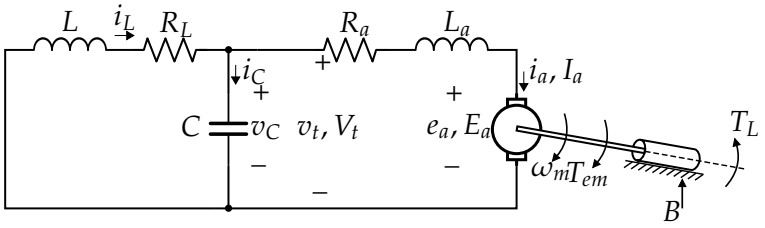

Figure 4.16: Buck Converter With Motor Load (Off Mode)

State Space Averaging

The DC motor Equations 3.4 and 3.7 ($T_{WL} = 0$) (from Section 3.2) for state variables ω_m and i_a, respectively, are also applicable for both modes. As the converter has two different modes of operation, we will apply state space averaging, a technique which can describe the input-output relation of a switching converter with different modes of operation [32]. Therefore, the final differential equations for constructing the state-space model are obtained by averaging, i.e.: by adding the terms in one mode to their counterparts in another, this is shown in Equations 4.27 and 4.28.

$$\frac{di_L}{dt} = \frac{1}{L}(V_i - i_L R_L - v_t)\delta + \frac{(-v_t - i_L R_L)}{L}(1 - \delta) \tag{4.27}$$

$$\frac{dv_t}{dt} = \frac{1}{C}(i_L - i_a)\delta + \frac{1}{C}(i_L - i_a)(1 - \delta) \tag{4.28}$$

The same averaging is also done for the differential equations obtained from the DC motor but they remain the same. In the end, the final averaged equations for state variables i_L, v_t, i_a and ω_m are shown below.

$$\frac{di_L}{dt} = \frac{1}{L}(V_i\delta - i_L R_L - v_t) \tag{4.29}$$

$$\frac{dv_t}{dt} = \frac{1}{C}(i_L - i_a) \tag{4.30}$$

$$\frac{di_a}{dt} = \frac{1}{L_a}(v_t - R_a i_a - k_E \omega m) \tag{4.31}$$

$$\frac{d\omega_m}{dt} = \frac{1}{J}(k_e i_a - B\omega_m) \tag{4.32}$$

There are four differential equations for four state variables. The controlled input for the system is the duty ratio of the active switch and the output of interest is the angular velocity. Therefore, we can now create the state-space model for the buck converter using the general state-variable form and the resulting model is represented by the Equations 4.33.

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix} i_L \\ v_t \\ i_a \\ \omega_m \end{bmatrix} &= \begin{bmatrix} -\frac{R_L}{L} & -\frac{1}{L} & 0 & 0 \\ \frac{1}{C} & 0 & -\frac{1}{C} & 0 \\ 0 & \frac{1}{L_a} & -\frac{R_a}{L_a} & -\frac{k_E}{L_a} \\ 0 & 0 & \frac{k_E}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} i_L \\ v_t \\ i_a \\ \omega_m \end{bmatrix} + \begin{bmatrix} \frac{V_i}{L} \\ 0 \\ 0 \\ 0 \end{bmatrix} \delta \\ y &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i_L \\ v_t \\ i_a \\ \omega_m \end{bmatrix} \end{aligned} \tag{4.33}$$

Additionally, it should be noted that the output matrix in Equation 4.33 can be changed to obtain a different output other than the rotational speed (e.g.: terminal voltage). Furthermore, the terms appearing in the $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ matrices have been summarized in the Table 4.1.

Table 4.1: Parameter Table

Parameter	Value
R_a	4.334Ω
L_a	$3.334mH$
k_T	$0.1877Nm/A$
k_E	$0.1877Nm/A$
B	$6.1502 \cdot 10^{-4}Nms/rad$
J	$2.9367 \cdot 10^{-4}kgm^2$
V_i	$12V$
L	$3.65mH$
C	$2.2mF$
R_L	0.145Ω

4.5 Filtering

The closed loop control scheme depends heavily on the sampled sensor data, which is a crucial contributor to the error calculation. Generally, in real systems, the relevant sensor output is conjoined with random unwanted noise. As such, if the sensor noise is not removed, it causes a conflict for the controller in directing its efforts towards error reduction as a response to plant disturbances or towards reducing the error caused by the sensor noise [30]. The rapid fluctuations in the sensor sample values are mainly associated with the high frequency noise components while the low-frequency components of the signal contain the pertinent measured data. Consequently, a low pass filter would adequately serve the purpose of filtering the sensor noise. The transfer function for a first order low pass

filter is given by Equation 4.34 where τ is the time constant which is the inverse of the cutoff frequency.

$$\frac{1}{\tau s + 1} \tag{4.34}$$

The measurement units for the system involves a current sensor and a speed sensor. Accordingly, a single pole low-pass filter is applied and the filter transfer function for the current and speed sensor is given by Equation 4.35 and 4.36 respectively. The cutoff frequency in the transfer functions of Equations 4.35 and 4.36 is obtained through manual tuning of time constant, with the selection of suitable filter for each sensor being the objective. These filters are later used in the forthcoming sections while discussing the further analysis.

Current Sensor Filter:

$$\frac{1}{0.007s + 1} \tag{4.35}$$

Speed Sensor Filter:

$$\frac{1}{0.09s + 1} \tag{4.36}$$

Additionally, the plots for the filtered and unfiltered current sensor data as well as the speed sensor data are shown in Figure 4.17.

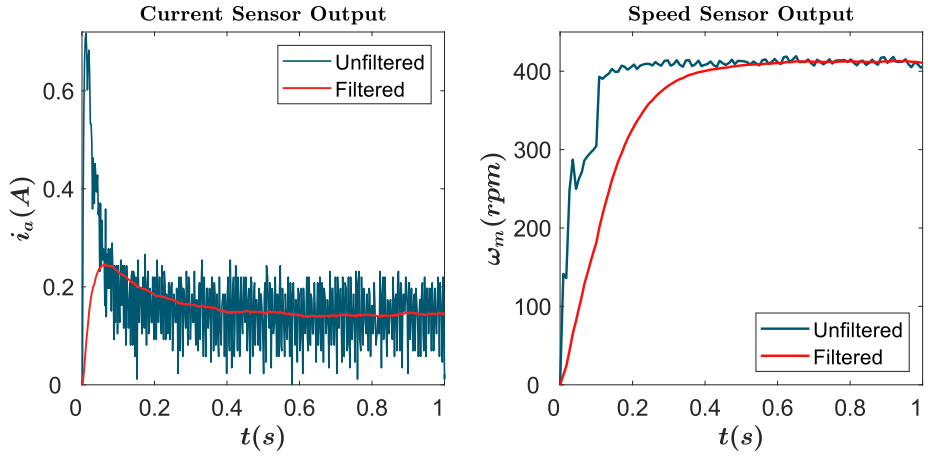

Figure 4.17: Current and Speed Sensor Filtered and Unfiltered Output

4.6 Model Validation

Although, we have obtained the motor model in Section 4.2, for it to be useful we need to ensure that the results and predictions inferred from it are reliable [28]. This can be done by validating the model and as such the following section will concern towards model validation.

As stated in [28], model validation is done by comparing the model's behavior with the system's and evaluating the difference. As a result, the motor model is verified by comparing the step responses of the obtained transfer function to that of the actual motor. To simulate the step response, we employ SIMULINK, where

we implement the transfer function, giving it a step input voltage and plotting the resulting step response that describes the rotational speed dynamics of the motor. Likewise, the real motor receives the step input directly from a DC power supply and the resulting speed data is captured using Arduino through the speed sensor. To acquire a scrupulous validation, responses due to the step inputs ranging from $5 - 12V$ are compared, out of which the simulation and the real response for $5V$ and $12V$ is plotted and illustrated in the Figure 4.19 and 4.18, respectively.

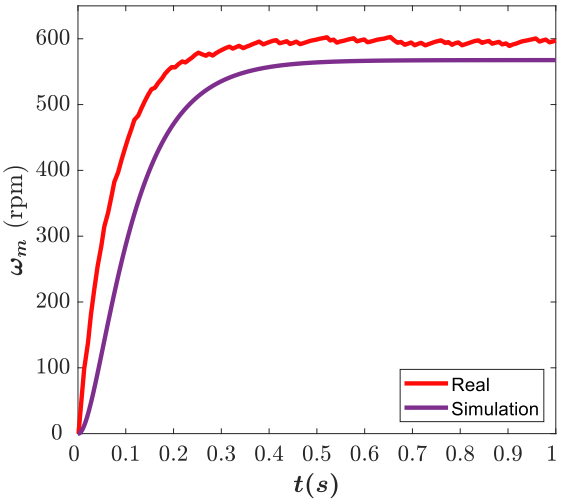

Figure 4.18: Simulated Motor Model vs Actual Motor Speed Response for $12V$

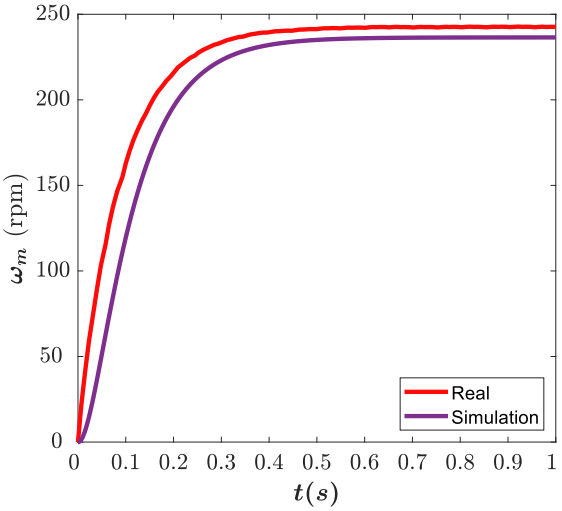

Figure 4.19: Simulated Motor Model vs Actual Motor Speed Response for $5V$

From Figures 4.19 and 4.18, the slight differences between the simulation and real step response is apparent, nonetheless it does not infer the invalidity of the model. The difference in the responses can be traced back to the motor parameters which are results of taking average of values at different input voltages. Considering the steady state difference between the compared entities in Figure 4.18, it is

a causal effect of the friction term (B). For instance, the graph in Figure 4.9 has a clear depiction of the $12V$ fiction value being lower than the mean friction value, which is used in the present transfer function (Equation 4.16). Thus, the transfer function with a averaged B term will have a lesser steady state value in contrast to the transfer function with a B term at $12V$, whose steady state will be much closer to that of the real response. Correspondingly, the difference in the transients, can be explained by the inertia term (J). Likewise, the prior reasonings also pertain to the $5V$ responses shown in Figure 4.19.

Notwithstanding, it can be reasoned that the model could be improved by adjusting the parameter values by suiting them to obtain a best fit between the simulation and real response. However, the prospect of this method is hindered by the circumstance that no one particular set of parameters would yield an ideal response for all the inputs in the operating range. In spite of that, limiting the operating range can likely result in a considerably well-suited model to the motor, yet in this project's case, any further limitation in the operating range may negatively affect the performance of the soon to be discussed closed-loop control methods. Therefore, the present model is evaluated befitting, serving the required purpose of adequately replicating the transients and steady-state behavior of the real motor.

4.7 Buck Converter Static and Dynamic Model Comparison

We have so far discussed the response of the DC motor in simulation and when directly connected to the power supply. However, in the control scheme the buck converter is placed as an intermediate component between the supply and the motor. The same configuration was also utilized while modelling the buck converter in Section 4.4. Therefore, the final combined converter and motor model which encompasses the steady-state and dynamic behaviors of the said components is acquired by substituting the parameter values given in Table 4.1 into the state-space model of Equation 4.33.

The acquired state space model considers the input to be the duty cycle and the rotational speed of the motor as the output. To analyze the model, we provide it with a unit step input (i.e.: 100%duty cycle) which produces the step response shown in Figure 4.20, indicated by the red line. The respective step response, corresponds to a rise time of $71ms$ and settling time of $131ms$. Likewise, the terminal voltage of the motor is a crucial state, whose step response for 100% duty cycle is illustrated in Figure 4.21. This response identifies the dynamics of the terminal voltage applied to the motor through a buck converter, which settles in $53ms$ to a steady state voltage, i.e.: $12V$ for maximum duty cycle. The oscillatory nature of the transient in this response can be attributed to the model's complex poles and the influence of such oscillations is also reflected in the dynamics of the speed response, due to the dependence of rotational speed on terminal voltage applied.

Hitherto, the analysis and the responses in this section have been a product of the dynamic model of buck converter and motor. However, an alternate approach to the modelling of a power electronics converter is through a static model. The buck converter output voltage follows a linear relation with the duty cycle ratio (Equation 4.37). This derives the static model for the respective converter with the

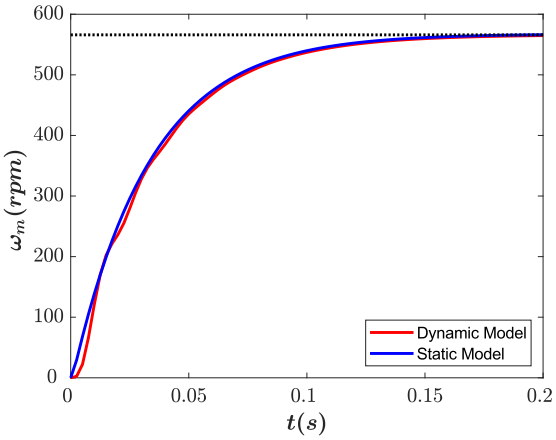

Figure 4.20: Simulated Speed Response with Static and Dynamic Model

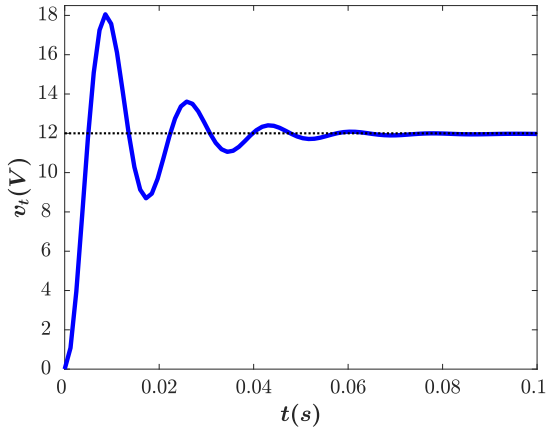

Figure 4.21: Terminal Voltage Response to 100% Duty Cycle for Dynamic Model

input being the duty cycle ratio and the output as the terminal voltage for motor.

$$v_t(t) = \delta(t)V_i \tag{4.37}$$

Moreover, in consideration of controller design and simulations, the converter's static model is placed just before the dynamic motor model, with its voltage output serving as input to the motor. The rotational speed response, now for the static converter model is shown in Figure 4.20, designated by the blue line. The response characterizes into a rise time of $72ms$ and settling time of $129ms$. Additionally, Figure 4.20 also compares the speed dynamics for both static and dynamic configurations of the buck converter. First, it can be noted that the dynamic model closely matches its counterpart, which serves as a validation for the acquired state space model. This is because of the static model configuration response being equivalent to the dynamic motor model response, which was verified in the last section. Furthermore, the characteristic differences between the two responses is $1ms$ considering rise time and $2ms$ for settling time, with a negligible steady state contrast. The difference infers a slightly faster response for the static model. Nonetheless,

the inconsequent small differences between the rise time and settling time deduce that replacing the dynamic model of the converter with the static one does not have any considerable adverse effects on the credibility of the model. The above argument is also supported by the dominant poles in the two cases, as they are in close proximity, separated only by a frequency difference of $0.031Hz$.

To make sure, the conclusion, i.e. the dynamic model can be replaced by the static model, is preserved in the real motor and buck converter circuit. An experiment was conducted to record the speed responses with and without the buck converter as an intermediate component between the power supply and the motor. The obtained results are shown in Figure 4.22. Similar to the simulation plot, the responses differ. When the converter was included the system was slower than without the converter. Likewise, the speed in the steady state for both conditions is approximately the same. In the end, the differences are small enough to be ignored and as a result, the static model of the buck converter in relation with the dynamic model of the motor will be utilized for any further procedures.

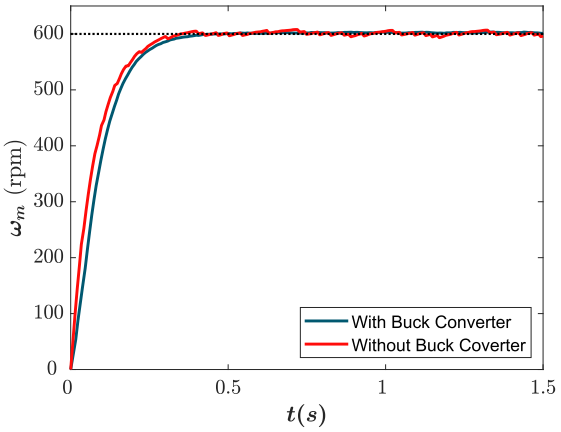

Figure 4.22: Speed response for Real Motor and Buck Converter Circuit

Chapter 5

Controller Design and Implementation

5.1 Preamble

The present chapter delineates the design and implementation of different controllers for achieving the desired DC motor speed control. Both classical and modern techniques were 5 utilized in this endeavor. The control strategy initially adopted was PID, previously discussed in Section 3.4. In this case, two different popular approaches were considered: (1) a single PID controller to regulate the angular speed and (2) a cascade PID configuration for controlling both the current and the speed. Additionally, the design and implementation of pole placement and LQR, introduced in Sections 3.5 and 3.6, respectively, were investigated. In all situations a reference of $400rpm$ was considered for the closed-loop system.

5.2 PID: Speed Control

The general structure of the PID speed control loop is shown in Figure 5.1.

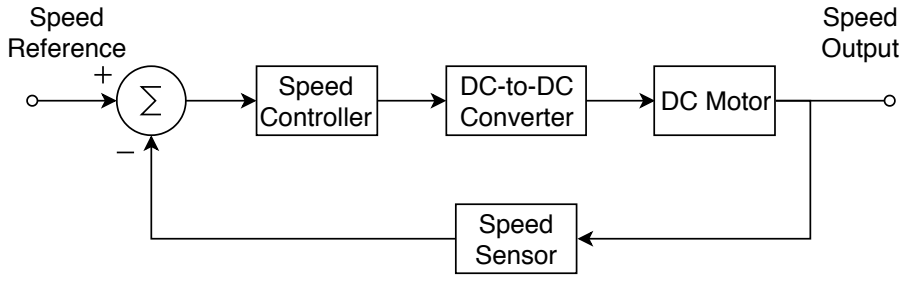

Figure 5.1: Control Loop for Speed Control

Eliminating the torque disturbances and replacing each block in Figure 5.1 with the equivalent transfer functions, we obtain the structure shown in Figure 5.2. Based on the discoveries presented in Chapter 4, one can identify the transfer functions shown in the block diagram.

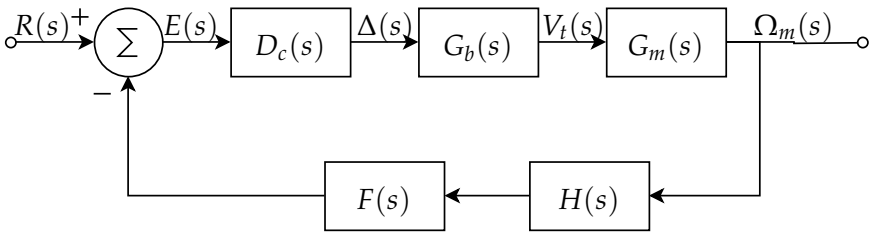

Figure 5.2: Control Loop for Speed Control Showing Transfer Functions

Combining $G_b(s)$ (buck converter) and $G_m(s)$ (motor) we find the total plant relationship $G(s)$ between duty cycle $\Delta(s)$ and angular speed $\Omega_m(s)$.

$$G(s) = \frac{2.3 \cdot 10^6}{(s + 1271.6)(s + 30.4)}$$

The function $H(s)$ is dependent on the time delay of the encoder t_d, experimentally found to be $8ms$. Due to the fact that time delay is expressed as an exponential in the Laplace domain, the two-term Padé approximation can be utilized to obtain a first-order rational transfer function [31]. In view of this we have

$$H(s) = e^{-t_d s} = \frac{e^{-t_d s/2}}{e^{t_d s/2}} \approx \frac{1 - t_d s/2}{1 + t_d s/2} = \frac{1 - 0.004s}{1 + 0.004s}$$

The second function present on the feedback path, $F(s)$, is the transfer function of the first-order low-pass filter with time constant $\tau_f = 0.09$, which is the same as the one in Equation 4.36.

PID Tuning

The employed PID design procedure is the one suggested in [59] and it is accomplished via computer-aided root locus. This method was chosen for it was deemed to be more analytical than the manual tuning approach. Before showing the steps taken for controller development, a certain PID variant needed to be chosen. Judging by the specialized literature, the most common classical controller for DC motors is the PI configuration [55, 68, 69], giving

$$D_c(s) = \frac{\Delta(s)}{E(s)} = k_p + \frac{k_i}{s} \tag{5.1}$$

which may be rewritten in the more convenient form for root locus design

$$D_c(s) = \frac{k(s + a)}{s} \tag{5.2}$$

Without further ado, the root locus design consisted of the following three stages:

1. Use the root locus method for proportional-only control, that is, while k varies and a is set to 0, and select a gain that creates a trade-off between stability and steady-state performance.

2. Fix k to the value identified in Stage 1 and obtain the root locus as the parameter a varies, thus introducing the integral action. Find the value of a which maximizes the relative stability of the closed-loop system.

3. Fix a to the value identified in Stage 2 and obtain once again the root locus as the parameter k changes. If possible, choose a k that improves the transient performance. Alternatively, utilize the value of k found in Stage 1.

We begin by noting that for the system shown in Figure 5.1, the closed-loop transfer function is

$$T_{cl}(s) = \frac{\Omega_m(s)}{R(s)} = \frac{D_c(s)G(s)}{1 + D_c(s)G(s)H(s)F(s)} \tag{5.3}$$

with the characteristic equation

$$1 + D_c(s)G(s)H(s)F(s) = 1 + D_c(s)L(s) = 0 \tag{5.4}$$

where $L(s)$ is the loop transfer function without the controller. Multiplying the known transfer functions yields

$$L(s) = \frac{-2.556 \cdot 10^7 (s - 250)}{(s + 1271.6)(s + 250)(s + 30.4)(s + 11.1)}$$

Considering proportional compensation ($D_c(s) = k$), Equation 5.4 becomes

$$1 + kL(s) = 0 \tag{5.5}$$

The root locus plot corresponding to Equation 5.5 is shown in Figure 5.3.

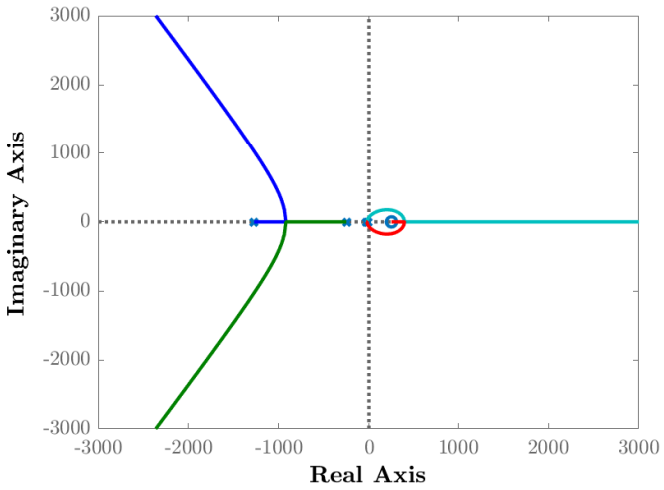

Figure 5.3: Root Locus for $L(s)$

Since the effect of changing the gain k over the dominant closed-loop poles is not apparent from this illustration, an additional figure (Figure 5.4) shows the same plot but this time the focus is on the trajectory of the two poles that are closer to the jw-axis.

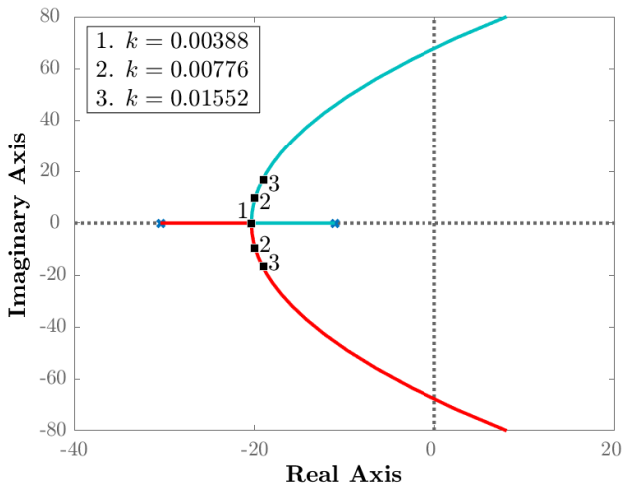

Figure 5.4: Closer View Upon the Root Loci for the Dominant Closed-Loop Poles (Stage 1)

It is fairly conspicuous that choosing a k that brings the two poles at the loci breakout point (-20.4) provides the best relative stability. This corresponds to $k = 0.00388$. One can check the transient response of the proportionally-controlled closed-loop system for values bigger than 0.00388 and opt for the gain that produces the trade-off mentioned in Stage 1. Figure 5.5 shows three responses corresponding to the values of the break-out gain, and the same gain increased twofold and fourfold, respectively.

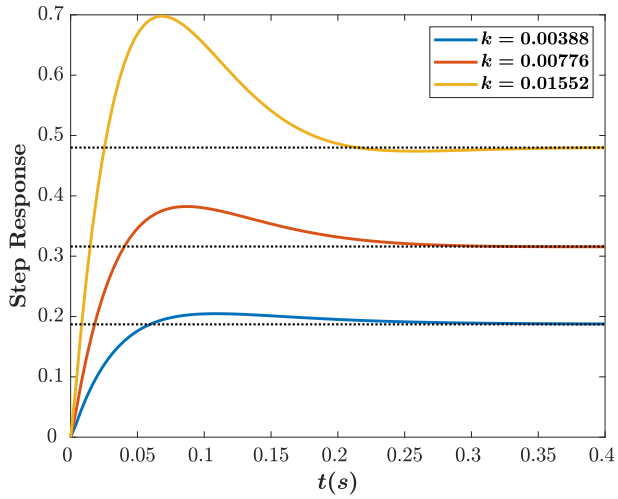

Figure 5.5: Transient Response of $T_{cl}(s)$ with Proportional Control for Different Values of k

Based on the graphs in Figure 5.5, $k = 0.00776$ represents the best option out of the investigated gain set, judging from a stability and steady-state perspective. Evidently, a k different than, yet close to, 0.00776 may actually offer the best performance, but Stage 1 will be ended here.

The second step of root locus design assumes a different loop transfer function, as k is fixed and a is varied. In this case, the characteristic equation of $T_{cl}(s)$ is

obtained by substituting Equation 5.2 into Equation 5.4 yielding

$$1+\frac{k(s+a)}{s}L(s)=1+kL(s)+\frac{ka}{s}L(s)=1+a\frac{kL(s)}{s(1+kL(s))}=1+aL'(s)=0 \quad (5.6)$$

where $L_1(s)$ can be calculated to be

$$L_1(s)=\frac{-1.983\cdot 10^5(s-250)}{s(s+1271.4)(s+251.8)(s-(-20+j9.6))(s-(-20.9-j9.6))}$$

Note that $L_1(s)$ is one order higher than $L(s)$. This is due to the pole added at origin by the integral part of $D_c(s)$. The root locus plot corresponding to Equation 5.6 is the one in Figure 5.6 and a closer view upon the dominant closed-loop poles is shown in Figure 5.7. Maximizing relative stability implies having the two poles at the break-out point, that is, they should coincide. The gain a that produces this effect is 11.4, and the poles move to -8.95.

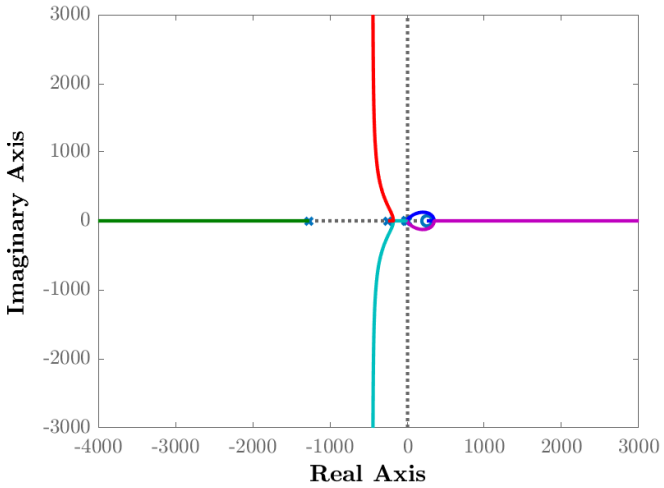

Figure 5.6: Root Locus for $L_1(s)$

The last step of the design procedure recommends yet another root locus analysis using the newly-found constant a and a variable k. We substitute $D_c(s)$ in Equation 5.4 with Equation 5.2 and obtain the characteristic equation of $T_{cl}(s)$, which, arranged in root locus form, is

$$1+\frac{k(s+a)}{s}L(s)=1+k\frac{(s+a)L(s)}{s}=1+kL_2(s)=0 \quad (5.7)$$

where

$$L_2(s)=\frac{-2.556\cdot 10^7(s-250)(s+11.4)}{s(s+1271.6)(s+250)(s+30.4)(s+11.1)}$$

Again, one can observe the presence of the integral term by looking at the denominator of $L_2(s)$. The entire root locus plot can be found in Figure 5.8 and

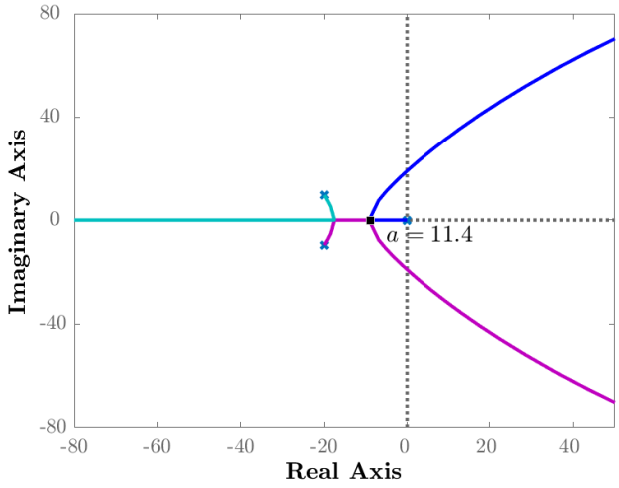

Figure 5.7: Closer View Upon the Root Loci for the Dominant Closed-Loop Poles (Stage 2)

the trajectory of the three dominant closed-loop poles is also shown in Figure 5.9. At the break-out point situated at -8.98 one finds for k the same value as the one chosen in Stage 1, namely 0.00776.

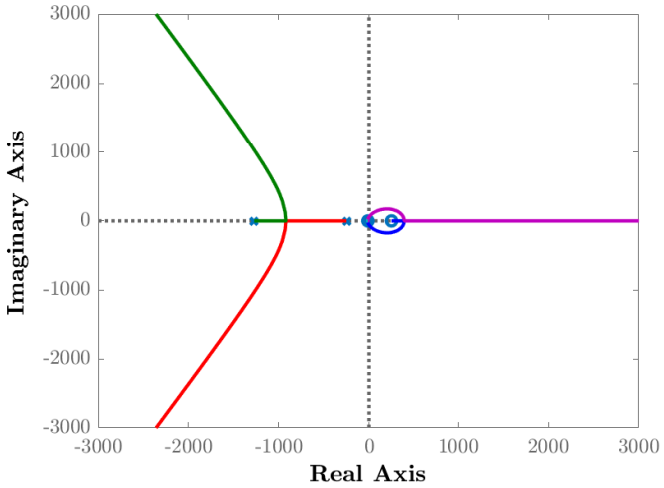

Figure 5.8: Root Locus for $L_2(s)$

We can investigate how a slight increase in k affects the transient performance of the closed-loop step response for a fixed a. For that purpose, we look at the output with proportional gains 0.00776, $1.25 \cdot 0.00776 = 0.0097$ and $1.5 \cdot 0.00776 = 0.01164$. The three resulting wave forms are plotted together in Figure 5.10. Additionally, Table 5.1 lists the values of the relevant performance parameters.

From both the figure and the table, it seems that the best choice is $k = 0.0097$, as for this gain the response rises and settles quickly with almost no overshoot. Using the previously found $a = 11.4$, we may proceed to calculate the coefficients of the PI controller.

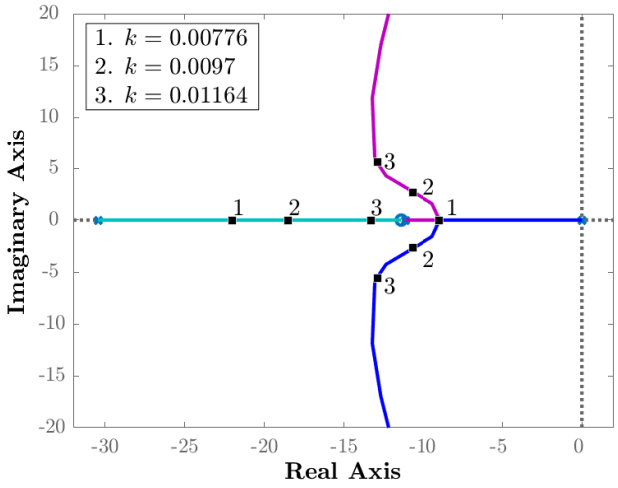

Figure 5.9: Closer View Upon the Root Loci for the Dominant Closed-Loop Poles (Stage 3)

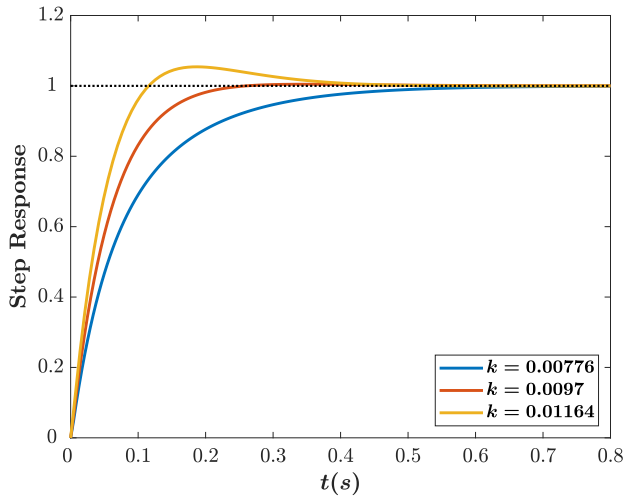

Figure 5.10: Step Response of $T_{cl}(s)$ with Proportional-Integral Control for Different Values of k

Table 5.1: Transient-Response Specifications of the Closed-Loop System for Different Values of k

k	$t_r[s]$	$t_s[s]$	$M_p[\%]$
0.00776	0.216	0.42	0
0.0097	0.12	0.197	0.461
0.01164	0.0775	0.326	5.42

$$k_p = k = 0.0097 \qquad k_i = k \cdot a = 0.1106$$

Thus, the controller is described by

$$D_c(s) = 0.0097 + \frac{0.1106}{s}$$

We now substitute the expressions of all the transmittances into $D_c(s)L(s)$ to

obtain the final loop transfer function $L_g(s)$, with the developed controller

$$L_g(s) = \frac{-2.479 \cdot 10^5 (s-250)(s+11.4)}{s(s+1271.6)(s+250)(s+30.4)(s+11.1)}$$

Frequency Domain Analysis

The characteristics of the closed-loop system can be further explored using the tools available in frequency domain, i.e.: Bode and Nyquist plots, since they provide an answer to the question regarding the relative stability, in other words, how stable the closed-loop system is [30, 59].

First, note that Figure 5.11 shows the Nyquist plot of the just-definded loop transfer function $L_g(s)$. The right-hand side of the illustration provides a closer view over the region where the parametric curve crosses the real axis of the $L_g(s)$-plane.

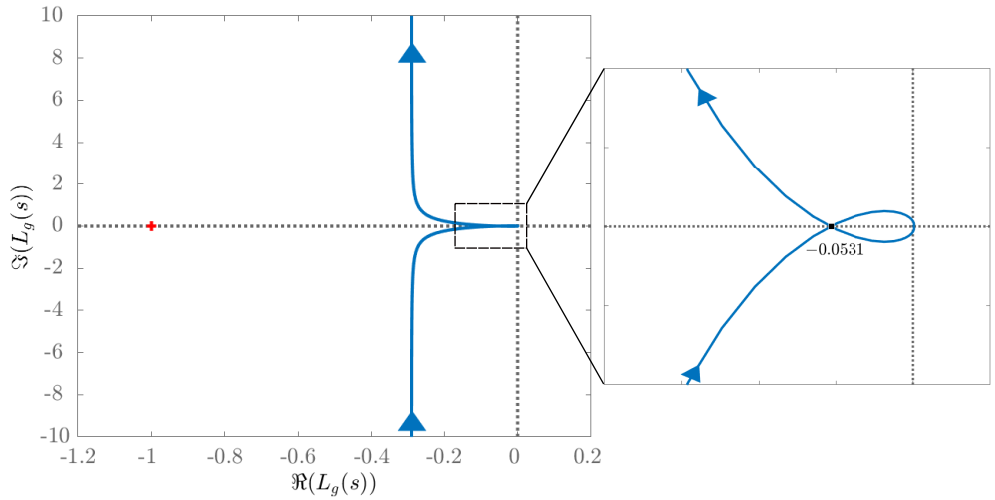

Figure 5.11: Nyquist Plot of $L_g(s)$

An initial observation worth-mentioning is that the plot does not encircle the point $(-1, 0 \cdot j)$. Moreover, $L_g(s)$ has no RHP poles and hence the same can be said about $1 + L_g(s)$, the denominator of $T_cl(s)$. Using the Nyquist stability criterion and the associated notation we have

$$N = Z - P \tag{5.8}$$

which means that, since $N = 0 = -P$, the number of closed-loop poles (Z) for the system is also zero, guaranteeing stability.

Having an answer to the stability question, we can examine the stability margins. From Figure 5.11 we observe that the magnitude of the loop gain $|L_g(s)|$ is 0.0531 when its phase $\angle L_g(s)$ is $-180°$. This happens at the phase crossover frequency ω_{PC} of $56.5 rad/s$. The gain margin in dB GM_{dB} can be calculated to be

$$GM_{dB} = 20 \cdot log\left(\frac{1}{|L_g(j\omega_{PC})|}\right) = 25.4dB$$

Similarly, from the same Figure 5.11, it can be estimated that when $|L_g(s)|$ equals 1, $\angle L_g(s)$ is -105.7. The corresponding gain crossover frequency ω_{GC} is $6.39rad/s$, giving a phase margin PM of

$$PM = 180^\circ + \angle L_g(j\omega_{GC}) = 74.3^\circ$$

The same information about relative stability is encoded in the Bode plot of the loop transfer function $L_g(s)$, shown in Figure 5.12.

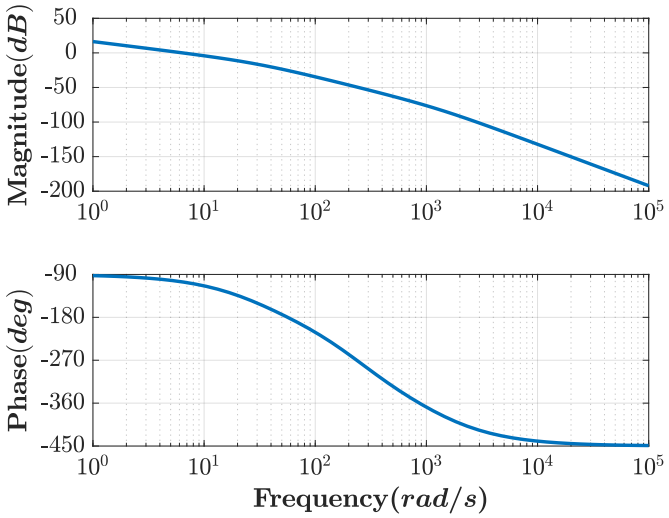

Figure 5.12: Bode Plot of $L_g(s)$

One can easily identify, directly from the plot, ω_{PC} and ω_{GC} by searching for the frequency at which the phase plot crosses -180° and the magnitude plot crosses $0dB$, respectively. The gain at ω_{PC} is linked with the GM_{dB}, whereas the phase at ω_{GC} defines the PM. A fast sanity check can be performed to see that the values found from the Bode plot are the same as the ones extracted from the Nyquist plot.

To form the closed-loop transfer function, we substitute each term in Equation 5.3 with its numerical equivalent, an operation which yields

$$T_{cl}(s) = \frac{2.231 \cdot 10^4 (s+250)(s+11.11)(s+11.4)}{(s+1271.4)(s+252.2)(s+18.5)(s-(-10.5+j2.8))(s-(-10.5-j2.8))}$$

The model of the PMDC motor interfaced by a buck converter and controlled by a PI compensator is a minimum-phase fifth order system with three zeros.

It should be stressed that the preceding design procedure is valid under the assumption that the controlled motor is an ideal system, whereas in reality the working of any electrical machine is limited by its physical characteristics (e.g.: there is an upper bound for the voltage V_t that can be applied at the terminals). A similar argument holds with respect to the MOSFET that dictates the operation

of the power converter: the gate can be in a state between fully open ($\delta = 0\%$) and fully closed ($\delta = 100\%$). To simulate this nonlinear phenomenon, a saturation block is utilized in the SIMULINK model, and the controller is further tuned, if necessary, to obtain a satisfactory performance. For this reason, differences between the responses obtained in MATLAB and SIMULINK are to be expected and must be tolerated.

Implementation

Notwithstanding the digital implementation, as it was already noticed, the design was entirely based on continuous-time techniques. This is in fact valid for all the controllers developed as part of this project. To emulate the continuous-time PID behaviour, the following approximations have been performed through discretization:

$$k_p e(t_n) + k_i \int_0^{t_n} e(\tau)d\tau + k_d \frac{de(t_n)}{dt} = k_p e(t_n) + k_i \sum_{j=1}^{n} e(t_j)\Delta t + k_d \frac{e(t_n) - e(t_{n-1})}{\Delta t} \tag{5.9}$$

where Δt is the fixed sampling period T_s associated with a specific control loop. For the PI compensator and the speed loop shown in Figure 5.2, the sampling frequency (in rad/s) ω_s was selected based on the closed-loop system bandwidth ω_{BW}. Choosing ω_s to be $40\omega_{BW}$ is a popular guideline for sampling, a "rule of thumb" for practical applications, although a theoretical value of $2\omega_{BW}$ would suffice, according to the Nyquist-Shannon sampling theorem [30]. The bandwidth of $T_{cl}(s)$ is found from the Bode magnitude plot in Figure 5.13: it is the frequency at which $|T_{cl}(j\omega)|$ drops to $-3dB$. The exact value is $17.81 rad/s$, which means

$$T_s = \frac{2\pi}{\omega_s} = \frac{\pi}{20\omega_{BW}} = \frac{\pi}{20 \cdot 17.81} = 0.0088s$$

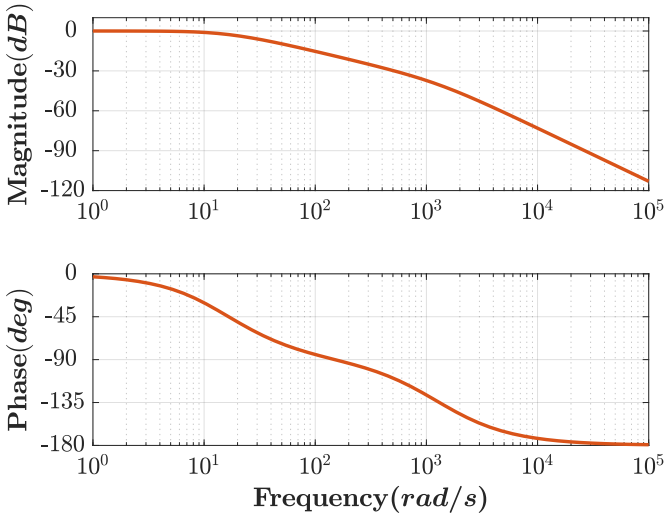

Figure 5.13: Bode Plot of $T_{cl}(s)$

In order to implement the duty cycle saturation, the output of the PI controller is compared to the duty cycle limits at each time step t_n and simply restricted to increase or decrease past the bounds. For the integrator antiwindup (first mentioned in Section 3.4), a decision is made to interrupt the summing action of the I term depending on the sign of the error $e(t_n)$ and whether the controller is in saturation or not.

The employed discrete equivalent of the filter described by $F(s)$ is the Infinite Impulse Response (IIR) single-pole low-pass filter whose recursion equation is [70, 71]

$$y[n] = y[n-1] + \alpha(y[n-1] - x[n]) \tag{5.10}$$

where $x[n]$ designates the nth sample of the filter input and $y[n]$ represents the nth sample of the filtered signal. The coefficient $\alpha \in [0, 1]$ is given by [70, 71]

$$\alpha = 1 - e^{-T_s/\tau_f} \approx T_s/\tau_f \tag{5.11}$$

Results

A comparison between the response of the SIMULINK model (saturation and antiwindup included) and the real system to a step speed reference of $400rpm$ is depicted in Figure 5.14.

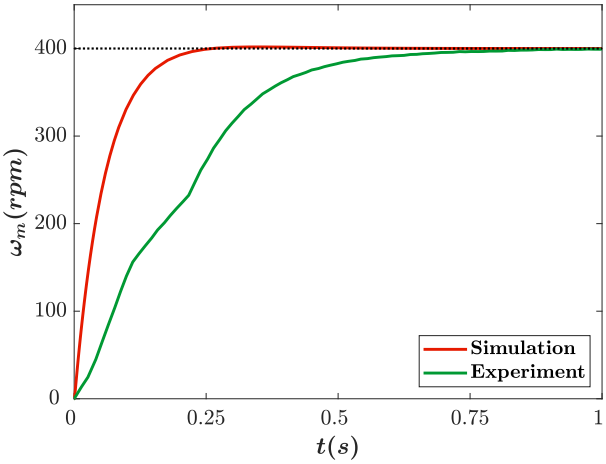

Figure 5.14: Simulation and Experimental Results for the PID Speed Controller

The time-domain specifications for the real response are listed in Table 5.2.

Table 5.2: Transient-Response Specifications of the Finalized Pole Placement Design

$t_r[s]$	$t_s[s]$	$M_p[\%]$
0.357	0.625	0

Although according to Figure 5.14 the motor eventually reaches the required speed in both cases, it becomes immediately clear that the transient characteristics are altered in practice. The spring of this mismatch is actually manifold. Firstly, the controller was designed in continuous time and implemented on a discrete

machine. Hence, discrepancies are unavoidable. Additionally, the phase shift can be explained by the potentially excessive filtering required to eliminate the speed sensor noise. The responses differ in rise time and settling time, but neither of them exhibits significant overshoot. All in all, the result of the implementation of the PID for speed control is considered satisfactory and successful. The SIMULINK model can be found in Appendix A, Figure A.2 and the full Arduino program is given in Appendix B.1.

5.3 PID: Speed and Current Cascade Control

From a general control standpoint, the cascade architecture comprises an inner (secondary) loop and an outer (primary) loop and offers better disturbance rejection [72]. Cascade PID control of converter-fed DC motors with an inner current loop and an outer speed loop is actually a prevalent approach in the electric drives literature and industry, as it is argued to offer flexibility, better transient performance and improved rejection of torque disturbances, while limiting the current [32, 55, 68]. This gives the rationale of attempting to implement such a control method in the present project. Figure 5.15 shows the employed structure of cascade architecture, while Figure 5.16 illustrates the block diagram when each subsystem is replaced with the corresponding transfer functions.

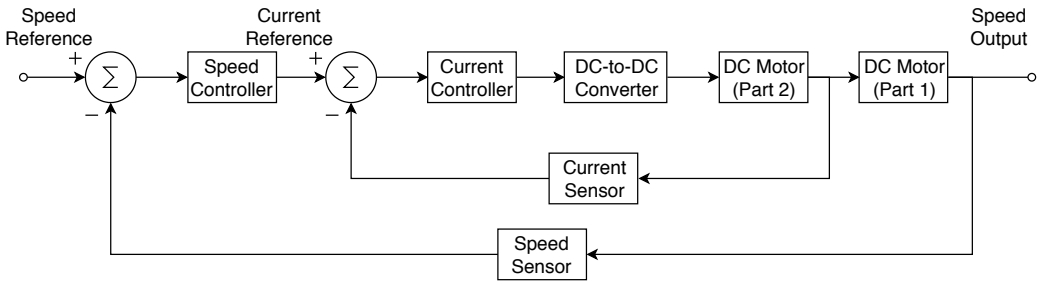

Figure 5.15: Control Loop for Current and Speed Cascade Control

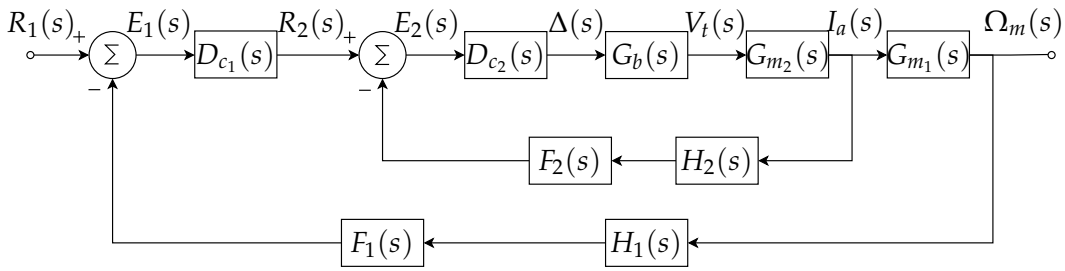

Figure 5.16: Control Loop for Current and Speed Cascade Control Showing Transfer Functions

Notice that the configuration is rather complex when compared to the single-PID topology of Figure 5.1. This time, the speed controller $D_{c_1}(s)$ outputs the current reference $R_2(s)$ for the inner loop controlled by $D_{c_2}(s)$. Also, the motor transmittance $G_m(s)$ is split into $G_{m_1}(s)$ and $G_{m_2}(s)$, as shown in Figure 4.2, since the armature current $I_a(s)$ is fed back. A current sensor has been added to the system as well, and its representative transfer functions are the delay $H_2(s)$ ($t_{d_2} =$

$0.2ms$) and the low-pass filter $F_2(s)$ ($\tau_{f_1} = 0.007$). The transmittances $H_1(s)$ and $F_1(s)$ are the well-known functions associated with the speed sensor.

Preliminary Considerations

In order to maintain the desired behaviour of the control system when the cascade controller configuration is utilized, the outer (speed) loop should have a bandwidth $\omega_{BW_{speed}}$ significantly lower than the one of the inner (current) loop $\omega_{BW_{current}}$ [73]. Because of that, the two loops require different update rates [74]. The maximum frequency at which the fastest loop can run is limited by the time the MCU requires to complete the necessary tasks associated with the entire control system: acquiring sensor data and calculating the controller output. Thus, the minimum sampling time for the current loop is given by

$$T_{s_{current}} = T_{ex_{speed}} + T_{ex_{current}} \tag{5.12}$$

in which $T_{ex_{speed}}$ and $T_{ex_{current}}$ are the execution times for the outer and inner loops, respectively, and $T_{s_{current}}$ is the sampling time for the inner loop. Since it was decided to run the speed loop five times less often than the current one, the sampling time for the outer loop $T_{s_{speed}}$ is simply

$$T_{s_{speed}} = 5T_{s_{current}} \tag{5.13}$$

An illustration of the preceding explanation is shown in Figure 5.17.

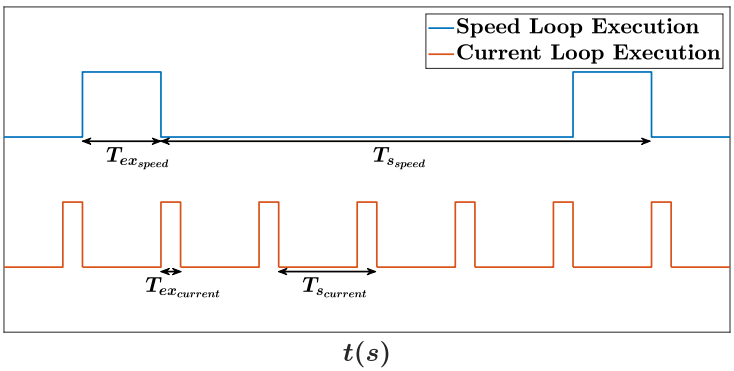

Figure 5.17: Timing Diagram for Cascade Control Execution

The intervals $T_{ex_{speed}}$ and $T_{ex_{current}}$ were accurately measured in software using dedicated Arduino functions that indicated a $T_{ex_{speed}}$ of $8ms$ and a value of $0.42ms$ for $T_{ex_{current}}$. From Equations 5.12 and 5.13, we have

$$T_{s_{current}} = 8.42ms \qquad\qquad T_{s_{speed}} = 42.1ms$$

Now, using the relationship between sampling time and bandwidth, one can find the maximum bandwidths $\omega_{BWmax_{current}}$ and $\omega_{BWmax_{speed}}$ of the two closed-loop systems for which the MCU is able to sample at the required rate, i.e.: 40 times faster than the individual bandwidths, as follows

$$\omega_{BWmax_{current}} = \frac{\omega_{s_{current}}}{40} = \frac{\pi}{20T_{s_{current}}} = \frac{1000\pi}{20 \cdot 8.42} = 18.65rad/s$$

$$\omega_{BWmax_{speed}} = \frac{\omega_{s_{speed}}}{40} = \frac{\pi}{20T_{s_{speed}}} = \frac{\omega_{BWmax_{current}}}{5} = 3.73rad/s$$

Note that bandwidths lower than the calculated ones are of course acceptable as long as the ratio between them (5) is maintained, since that would only increase the the sampling-to-bandwidth quotient.

PID Tuning

Quoting [68], "Speed, torque, current and position of PMDC motor are generally controlled by cascade connected controllers. In these applications, PI controllers are preferred instead of PID controllers". In view of that, the PI configuration was selected once again, as in the single-loop case. Notwithstanding the root locus method for controller tuning presented in Section 5.2 can be employed for the cascade configuration, the bandwidth limitations pushed towards a heuristic approach for finding the two sets of PI coefficients. Presenting the entire tuning process would be redundant and hence only the final parameters that fulfilled the bandwidth criteria and gave a satisfactory transient performance as well are shown here. A relevant observation is that the controllers were tuned sequentially, from inner to outer, as indicated in [55, 68, 73]: the secondary loop was tuned first and the resulting closed-loop transfer function was utilized as a substitute for the inner loop when the primary controller was tuned.

In the case of the current loop, the results were

$$k_p = 0.2 \qquad k_i = 25$$

which caused a bandwidth $\omega_{BW_{current}}$ of $16.56rad/s$, lower than $\omega_{BWmax_{current}}$.

For the speed loop, the manual tuning eventually led to

$$k_p = 0.004 \qquad k_i = 0.01$$

and the resulting bandwidth of $3.4rad/s$, lower than $\omega_{BWmax_{speed}}$ and approximately five times smaller than $\omega_{BW_{current}}$. The step responses for the two closed-loops are shown in Figure 5.18.

Implementation

Implementing the cascaded controllers for speed and current in the MCU was done in a similar fashion to the case of a single control loop for speed. The simulation block diagram can be found in Appendix A, Figure A.3 and the code is given in Appendix B.2. The essence of this particular program consists of a nested software loop, allowing the current control loop to run 5 times faster than its speed counterpart. Figure 5.19 proves that the program is able to run the loops in the desired manner.

Results

The simulation and the experimental responses of the cascade speed and current control loops are plotted in Figure 5.20. The observed simulation response demon-

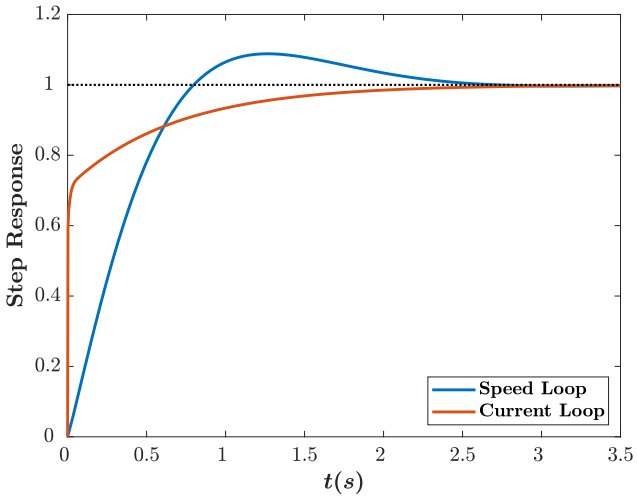

Figure 5.18: Step Responses of the Primary and Secondary Loops

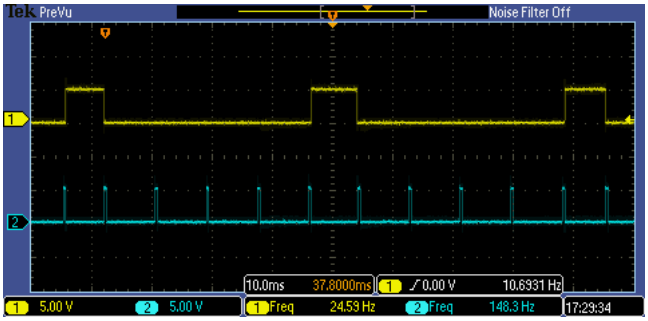

Figure 5.19: Oscilloscope Wave Forms Illustrating the Frequency of the Control Loops

strates certain time-domain features which we consider decent since it could not be much improved due to bandwidth limitations. With this, the cascade PID controller design is finalized. On the other hand, the experimental data exhibits a different behaviour compared to the obtained step response in all time-domain aspects. In addition, the reference is not followed in a desired manner and there exist periodic dips and bumps. Because of that, the time-domain specifications of the practical results are not computable.

As there could be many reasons causing the malfunction, we think the following two are the most significant. First, the implementation of cascade loops, where the reference for the inner loop is set based on the readings of a sensor makes the system more prone to noise. Furthermore, the presence of two sensors (speed and current) most likely increases noise dominance over the actual signal. As filtering can only be used to remove the noise partially, it is not attainable to have noise-free signals with no delay.

Another reason for the misbehaviour of the cascaded loops might be related to the manner the loops are executed. We have chosen to implement a dual-rate cascade loop structure as a consequence of our findings and discussion with control theory professionals. However, it is also a common practice to execute the loops at the same rate. On the other hand, similar results have been obtained through

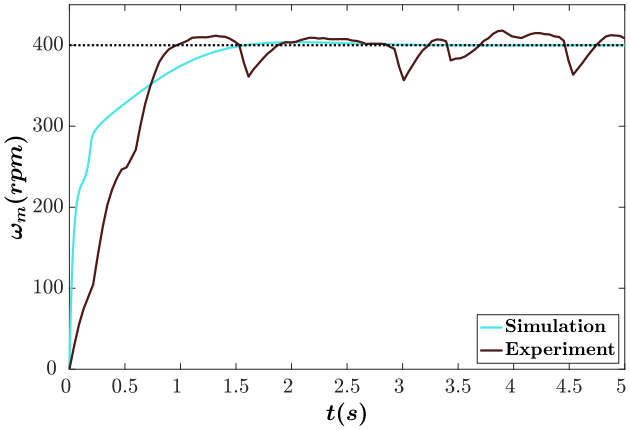

Figure 5.20: Simulation and Experimental Results for the Cascade PID Structure

experiments when the controller is tested for both fashions of loop execution. As a result, the response could not be improved further in spite of our continuous effort.

Regarding the measured current, the system response in practice was evidently inadequate pointing towards implementation faults,ergo the graph does not convey any useful information. Consequently, the said plot is not presented.

5.4 Pole Placement

The state-space representation of the system needs to be obtained before we can proceed to controller design. Based on Equations 3.2, 3.3, 3.4, 3.7 and 3.26, the following state-space representation including the integral state, x_i, is derived.

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix}\omega_m \\ i_a \\ x_i\end{bmatrix} &= \begin{bmatrix}\frac{-b}{J} & \frac{K_T}{J} & 0 \\ \frac{-K_T}{L_a} & \frac{-R_a}{L_a} & 0 \\ 1 & 0 & 0\end{bmatrix}\begin{bmatrix}\omega_m \\ i_a \\ x_i\end{bmatrix} + \begin{bmatrix}0 \\ \frac{V_t}{L_a} \\ 0\end{bmatrix}\delta + \begin{bmatrix}0 \\ 0 \\ -1\end{bmatrix} r \\ y &= \begin{bmatrix}1 & 0 & 0\end{bmatrix}\begin{bmatrix}\omega_m \\ i_a \\ x_i\end{bmatrix} \end{aligned} \tag{5.14}$$

We use MATLAB and prove that the system is controllable and observable. Since our system incorporates a speed and a current sensor, there is no need to develop an observer for the purpose of full-state feedback. We continue with deciding acceptable time domain properties for the closed-loop system as follows:

Table 5.3: Desired Time Domain Requirements for the Closed-loop System

$M_p[\%]$	< 1	$t_s[s]$	< 0.1

Roots for a second order system are formulated as in Equation 5.15.

$$s = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} \tag{5.15}$$

Relevant ω_n and ζ values for the predetermined time domain properties are calculated as shown below:

$$M_p = e^{-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}} = 0.01 \qquad \Longrightarrow \zeta = 0.82$$
$$t_s = \frac{3.2}{\zeta\omega_n} = \frac{3.2}{0.82 \cdot \omega_n} = 0.1 \implies \omega_n = 39.02$$

The computed ω_n and ζ values indicate the following pole locations: $s_{1,2} = -32 \pm j22$. Since the system consists of three states, it requires three roots to be placed on the pole-zero map. The third pole is placed fifty times further to the left compared to the other two poles so that its response is significantly faster. By having two dominant poles, the system behaves as a second order system and exhibits the desired time domain properties.

$$s_3 = 50 \cdot (-32) = -1600 \tag{5.16}$$

Figure 5.21 shows the step response of the system developed with the pole placement technique. It is seen that the rise-time and the overshoot requirements are met.

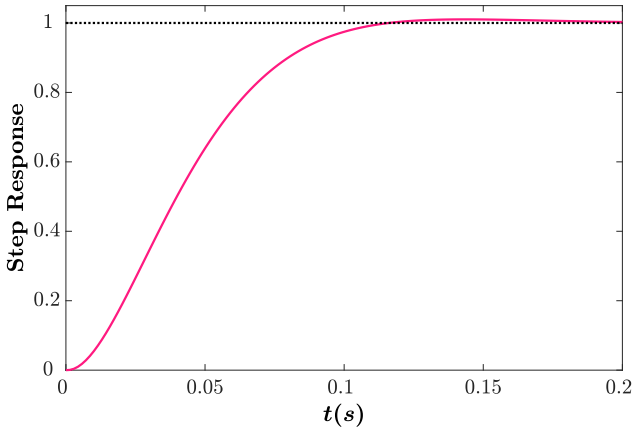

Figure 5.21: Step Response of the Closed-Loop System With Pole Placement

Consequently, $\boldsymbol{K_C}$ is computed according to the determined pole locations as being $[0.02800.10061.0489]$.

Results

Simulation of the system with the mentioned $\boldsymbol{K_C}$ gives a stable and satisfactory response that obeys the before-mentioned time domain features. However, the implementation of it in the setup yields an oscillatory response, yet a stable one. Nevertheless, this response is not considered to be satisfactory as its t_s is as long as $4.002s$, very slow to be regarded as adequate. In order to eliminate the oscillation and decrease t_s, we intuitively reduce $\boldsymbol{K_C}[3]$, the integral coefficient. After an iterative process of altering $\boldsymbol{K_C}[3]$ and evaluating the response, it is found that an acceptable closed-loop response is achieved when preliminary $\boldsymbol{K_C}[3]$ is divided

by 3.2. Figure 5.22 shows the aforementioned three closed-loop responses: respectively simulation response and experimental response before and after $K_C[3]$ is fine-tuned.

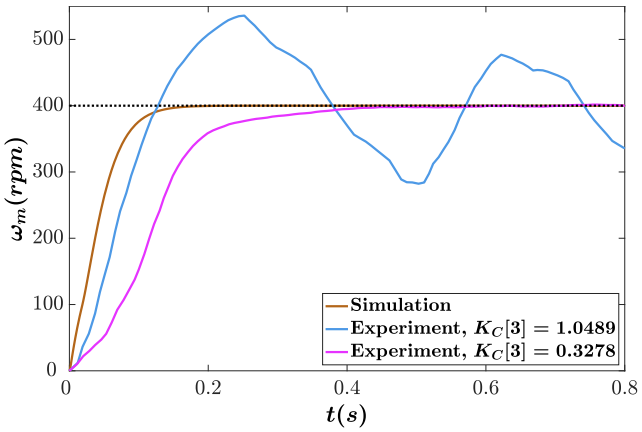

Figure 5.22: Simulation and Experimental Results for Pole Placement (Speed)

The results are assessed satisfactory as the motor is able to reach the reference in 0.37s without any overshoot. The system is also stable and has zero steady-state error. This concludes the design of the controller with pole placement. Table 5.4 lists the transient-response specifications of the finalized pole placement design.

Table 5.4: Transient-Response Specifications of the Finalized Pole Placement Design

$t_r[s]$	$t_s[s]$	$M_p[\%]$
0.167	0.370	0

Even though the finalized design performance has achieved desired M_p for the system, t_r has been slowed down by 0.67s. Notwithstanding, the response is viewed as successful and the controller design is concluded. Furthermore, we would like to discuss the current plots of the experimental data and the simulation for the finalized controller, illustrated in Figure 5.23.

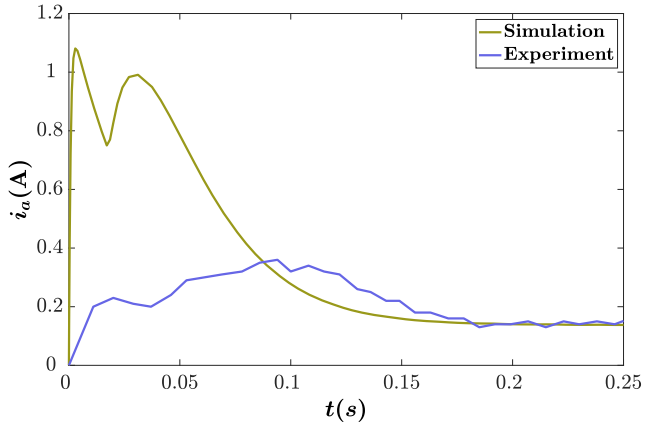

Figure 5.23: Simulation and Experimental Results for Pole Placement (Current)

It is seen that, within the first 0.08s, the setup seems to be drawing less current,

$0.35A$ at maximum, compared to the simulation which has a peak of $1.08A$. The reason is apparently the low-pass filter applied to the current measurements. As the current is increased from zero to $1A$ or so, within almost no time when the setup is first powered, the filter attenuates most of this rise and also introduces a phase delay. This is due to the cut-off frequency of the filter. Increasing the cut-off frequency would result in less attenuation, but also mean more noisy readings. As it is discussed in Section 4.5, a trade-off between noise reduction and signal fidelity has been made and the sensor cut-off frequency is determined. Therefore, we can do nothing but accept the captured sensor measurements. On the other hand, the steady-state value of the filtered signal is judged to be consistent with the simulation result. The signal is still noisy but it is as reasonable as it can be with the utilised current sensor and filter.

The Arduino code for pole placement design can be found in Appendix B.3 and the SIMULINK model is given in Appendix A, Figure A.4.

5.5 LQR

The same state-space representation as previous is considered for LQR. Thus, the system is both controllable and observable. We start with constructing $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices based on Bryson's rule as discussed in Section 3.6 considering the maximum state/input values given in Table 5.5, and obtain the matrices shown in Equation 5.17.

Table 5.5: Maximum Acceptable State/Input Values

$\omega_m[rad/s]$	60	$i_a[A]$	2	$\delta[\%]$	100

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{3600} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R} = 1 \tag{5.17}$$

As we do not have a maximum value for the integral state x_i, we initially give a weight of 1 to its corresponding $\boldsymbol{Q}$ matrix entry, $\boldsymbol{Q}[3,3] = 1$. In order to fine-tune it, various values have been tried based on our understanding of the system. It is found that the closed loop simulation response is satisfactory in terms of rise and settling time when $Q[3,3] = 2$. Figure 5.24 illustrates the corresponding closed-loop simulation responses for a reference of $400rpm$ when $\boldsymbol{Q[3,3]} = 1$ and $\boldsymbol{Q[3,3]} = 2$.

Table 5.6 lists the performance specifications of the two responses shown in Figure 5.24.

Table 5.6: Time Domain Specifications of the Closed-Loop Simulation for Different $Q[3,3]$ Values

$\boldsymbol{Q}[3,3]$	$t_r[s]$	$t_s[s]$	$M_p[\%]$
1	0.0963	0.1446	0
2	0.0773	0.1178	0

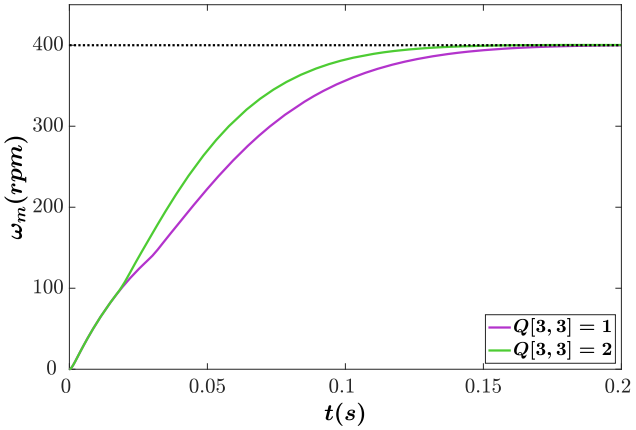

Figure 5.24: Simulation Responses of the LQR Controller for Two Different $Q[3,3]$ Values

Since the response for $\boldsymbol{Q}[3,3] = 2$ yields approximately 20% faster t_r and 19% faster t_s, the $\boldsymbol{Q}$ matrix is updated accordingly as shown in Equation 5.18.

$$\boldsymbol{Q} = \begin{bmatrix} \frac{1}{3600} & 0 & 0 \\ 0 & \frac{1}{4} & 0 \\ 0 & 0 & 2 \end{bmatrix} \quad \text{and} \quad \boldsymbol{R} = 1 \tag{5.18}$$

Consequently, we conclude the fine-tuning of the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices based on the closed-loop simulation response.

Results

For the chosen matrix entries, we obtain $\boldsymbol{K_C}$ = [0.0401 0.2671 1.4142]. The subsequent step is to run the setup with these gain values and observe the response. The experimental data shows that the response is marginally stable for the calculated $\boldsymbol{K_C}$. This is due to the fundamental difference between continuous and digital control design implementation. Due to our limited knowledge of digital control theory, we are compelled to manual-tune $\boldsymbol{K_C}$ matrix and better the system stability and performance. Intuitively, we evaluate the response by decreasing $\boldsymbol{K_C}[3]$, the integral coefficient, to various values in order to eliminate oscillatory behaviour of the system. A satisfactory result is achieved when $\boldsymbol{K_C}[3]$ is divided by 4. Figure 5.25 demonstrates the closed-loop response of the setup for both the preliminary $\boldsymbol{K_C}[3]$ and its updated value.

This finalizes the design of the LQR controller. Table 5.7 lists the transient-response specifications for the finalized design.

Table 5.7: Transient-Response Specifications of the Experimental Data

$t_r[s]$	$t_s[s]$	$M_p[\%]$
0.248	0.495	0

The results are regarded as satisfactory since the motor is able to reach the reference in less than half a second, 0.495s, without any overshoot. It also has t_r as short as 0.248s. Additionally, we would like to compare the current readings

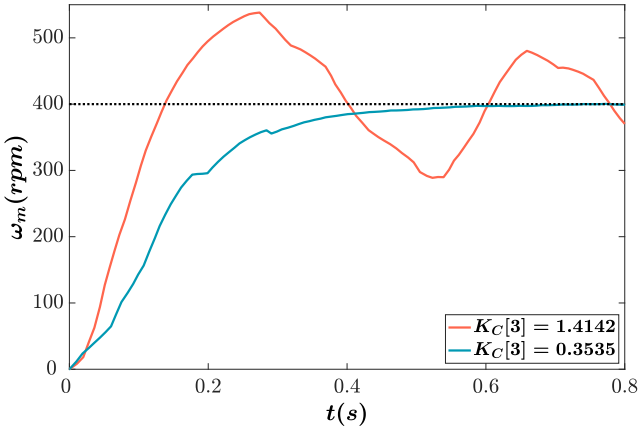

Figure 5.25: Simulation and Experimental Results for LQR (Speed)

from the system with the simulation. Figure 5.26 displays the simulation and the experimental i_a together.

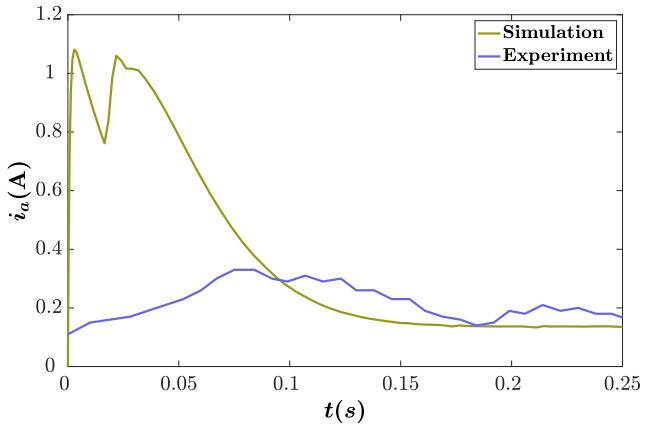

Figure 5.26: Simulation and Experimental Results for LQR (Current)

It is seen that the same behaviour of the measured current signal, attenuated and phase-delayed, as in the case of pole placement, lingers. Based on the same reasoning given in Section 5.4, the response is evaluated to be consistent with the simulation.

The Arduino code and the simulation model for the LQR controller design can be found in Appendix B.3 and Appendix A, Figure A.4, respectively.

Chapter 6

Testing

The controller design sections culminated to the finalized controllers for PID speed control, PID speed and current cascade control, Pole placement and LQR, along with exploring the set point tracking capabilities of the respective controllers. Moreover, the aspect still to be probed for the said controllers is disturbance rejection. It was established as the primary motive, in the problem analysis segment, for the controller to reify. In acknowledgement, the present section delves into the delineation of the procedure and results of the testing session, conducted to evaluate the disturbance rejection potentiality of the implemented controllers.

For the purpose of testing, each controller was effectuated through Arduino Mega to track the set point speed of $400rpm$ for the PMDC motor, at no load. Once the speed settled into the steady state, the motor was subjected to a step input in load torque, marking the first disturbance. As the motor reattains the reference speed, the previously input load torque is removed, exposing the motor to the second torque disturbance during the process. The same approach is applied for all four controllers. The graph presenting each control method's response during steady state is shown in Figure 6.1, also noting that the black doted line in the responses label the reference speed. Furthermore, every plot in Figure 6.1 has distinct axes to best show the collected data. On the application of the load, predictably, the speed undershoots and then, through error correction efforts of the controller, settles back to the set point. Likewise, when the load is removed, the speed overshoots, after which it again settles to the reference. The settling time after disturbance ([1]) and the percentage of overshoot and undershoot are determined as the judging parameters to a controllers' disturbance rejection proficiency. The Table 6.1 enlists these attributes for each controller.

Referring to the Table 6.1 and Figure 6.1, albeit all four controllers were successful in disturbance rejection and tracking the reference speed, the PID speed and current cascade control's response to disturbance proves to be the most inferior. Presumably, due to the rationale provided in Section 5.3. Notwithstanding, considering overshoot and undershoot, the remaining three tested controllers display identical reactions, with the LQR response having the minimum peak heights caused due to disturbances. In terms of the other attribute, i.e.: the recovery time,

[1]Let us call this "settling time after disturbance" "recovery time"

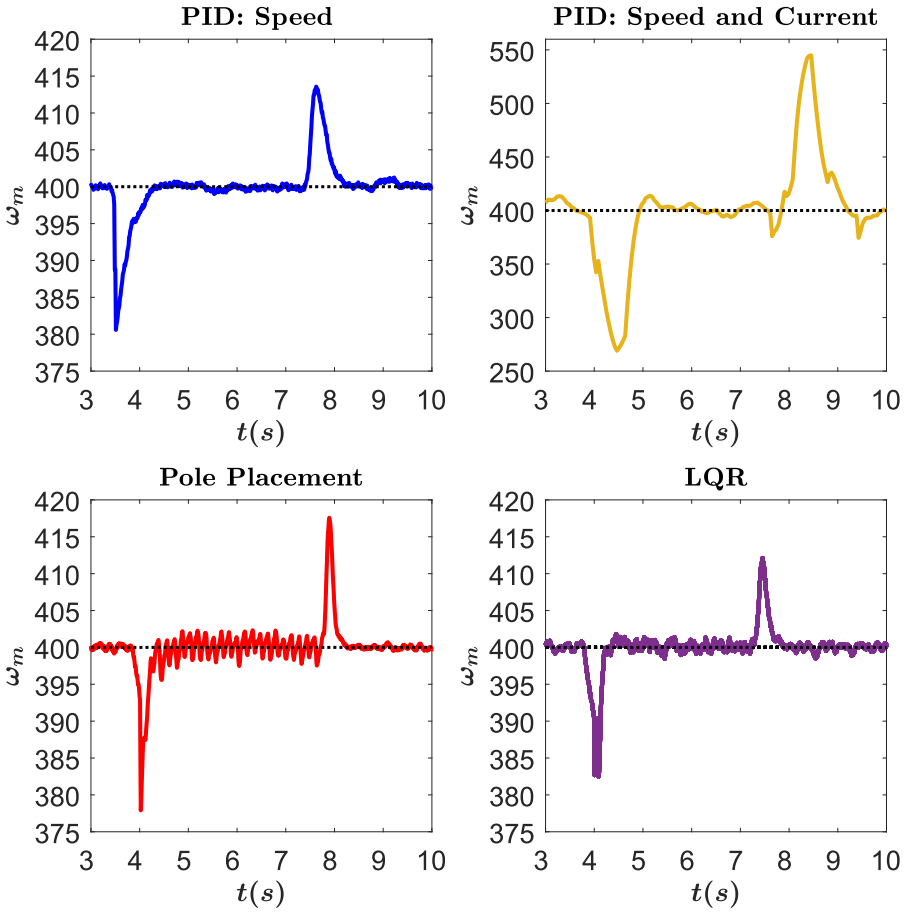

Figure 6.1: Responses to Disturbances for Different Controllers

Table 6.1: Testing Results

Controller	Load Applied		Load Removed	
	Undershoot	**Recovery Time**	**Overshoot**	**Recovery Time**
PID Speed	4.85%	0.93s	3.39%	0.83s
PID Cascade	32.75%	1.31s	36.23%	1.39s
Pole Placement	5.52%	0.50s	4.39%	0.51s
LQR	4.37%	0.43s	3.03%	0.66s

when the load is applied, the speed PID is noticeably two times slower than the fastest recovery time of 0.43s for LQR. Comparatively, there is only a minuscule difference between pole placement and LQR. Correspondingly, when the load is removed, the pole placement response settles the fastest to the set point. However, the difference between pole placement and LQR recovery time in this case, is just 0.15s. In conclusion, PID speed, pole placement, and LQR controllers show marginally contrasting responses to disturbances, but considering this marginality, the modern control technique LQR, quantifies as the most proficient when concerning disturbance rejection.

Chapter 7

Discussion

In the endeavor of controlling the angular speed of a buck-converter-fed PMDC motor, the following actions have been taken: a step-down switching regulator has been built, several experiments for determining the motor parameters have been carried out, a model of the system was created and analyzed and four controllers have been designed, implemented and tested. Unsurprisingly, the end result is far from a perfect one, as this project merely constitutes the first contact of the authors with the vast and intricate fields of control theory and power electronics. The following paragraphs are meant to elucidate the deficiencies of the developed system with respect to all its aspects, as well as to present possible solutions for overcoming these inadequacies.

The first discussed aspect of the project is the modeling of the buck converter, whose development was delineated in Section 4.4 and whose mathematical representation was eventually reduced to a constant gain. Despite the similarities found by pole and response comparison, it can still be argued that the static and dynamic models affect the transient response of the plant. Accordingly, capturing the converter dynamics would result in a more accurate model that would more closely describe the real system behavior. Continuing on the same topic, the modeling of the DC motor could have been improved as well, as the plots of Section 4.6 indicate. The parameters affecting the transient and steady-state performance of the model step response (mainly inertia and friction) should have been adjusted to compensate for the discrepancies between the theoretical representation and the actual motor. However, this is only possible in the context of a specific application, for which an operating point can be defined. Thus, it may be beneficial to direct the focus of the project towards a certain, more concentrated, practical functionality.

Another very important subject that requires clarifications is the controller design part. As emphasised many times throughout Chapter 5, all the compensators were created based on continuous time mathematical techniques (e.g.: the Laplace transform) even though they were implemented on a digital machine. This represented one of the main springs of discrepancy between the simulations and the experimental results. Digital control (e.g.: Z-transform) should be utilized in the design process in order to obtain closely-related responses in theory and practice. Moreover, the aforementioned mismatch also stems from the nonlinearities (i.e.:

saturation) ignored at the stage of choosing the controller gains using LTI system procedures (e.g.: root locus). Thus, nonlinear control is another aspect to be considered in the future to realize better compensation. As a remark for the controller design in the LTI case, nonlinearities and discretization issues left aside, perhaps delving deeper into frequency domain design would have been beneficial from a learning point of view.

It is worth discussing further about the controller performance. Both Chapters 5 and 6 demonstrated the inferiority of the cascade PID configuration. While we are not, by any means, claiming the ineffectiveness of the said approach in a general sense, for this project only, the other controllers are simply preferable. At the same time, we are well aware of the fact that the fashion in which the cascade control was implemented is probably not the correct one. The idea of using two different update rates for the primary and secondary loop was taken from [74] and was supported by discussions with teachers at the university, but the conducted experiments point towards design or implementation flaws. As an example, cascade control is supposed to improve disturbance rejection and yet this does not seem to be the case for the developed system. Solutions for these problems are yet to be found and would have probably been further investigated if it had not been for the time constraints.

In terms of the utilized hardware, it can be stated without the shadow of a doubt that a more thoughtful analysis is required at the stage of choosing sensors. An important lesson is that working with low-quality apparatus entails several issues when it comes to data acquisition and controller performance. For example, a speed sensor with a better resolution and detailed documentation would have greatly benefited the entire system. A similar argument is valid for the PMDC motor representing the core of this project. Utilizing a high-end device with an available datasheet specifying its characteristic constants is a more sensible decision than trying to identify (possibly erroneously) the parameters by experiment. Furthermore, working with a more robust machine, less prone to wearing and parameter fluctuations, is another hardware aspect worth-considering in the future.

Last, but not least, the implementation process was greatly hindered by our inability to establish a functional, not to mention reliable, communication between Simulink and Arduino. Different support packages have been tested to no avail, even though the Simulink documentation clearly suggests that computer-MCU serial data exchange is possible and widely used. Deploying the Simulink model to the external hardware platform was intended as well, though it really was just another fruitless attempt, as the microcontroller quickly ran out of memory, making the idea of uploading the model to the Arduino unfeasible. These obstacles led to the programming of the MCU by hand and significant effort was necessary to create four functioning pieces of software that could emulate the features of Simulink.

Chapter 8

Conclusion

The report elucidated the design of a DC motor speed controller for various control techniques with the intent of disturbance rejection. The success of the project will be evaluated by revisiting the set project objectives in Table 2.3. Based on the testing session, the description of the fulfilled project goals is the follwing.

A buck converter has been designed via a systematic approach that comprised theoretical circuit calculations. A substantial effort was directed towards ensuring CCM condition. The DC motor parameters for the selected range of operation were identified as a consequence of a series of experiments. The motor and the converter were modelled in Simulink using the determined parameters and have been subjected to several experiments to be verified. Two classical and two modern controllers were developed with the purpose of evaluating benefits and shortcomings of each. The controller design process consisted of the necessary measures to procure first stability, then time-domain specifications. The finalized controllers were applied in Arduino in line with their corresponding digital implementation.

The results collected in the testing period were analysed and a final conclusion was attained. The appearance of slightly different transient response to disturbance served as a basis for controller comparison. Three out of four controllers satisfactorily sustained the reference speed under load and no-load conditions. LQR is regarded as the best among them. The speed-only PID and pole placement techniques gave good results considering their short rise time and small overshoot, whereas the cascade PID loop had certain drawback in terms of following the reference.

In the light of the statements above, it is inferred that the developed prototype has been a product of constant meticulous effort. The work proved itself to be a sufficient demonstration for a speed controller. Inevitably, further refinement is entailed. Nonetheless, the present state is regarded complete and satisfactory.

Bibliography

[1] *Electrical Drives*. URL: https://www.ikbooks.com/home/samplechapter?filename=32_Sample_Chapter.pdf.

[2] *ELECTRIC DRIVES - N. K. DW, P. K. SEN - Google Books*. URL: https://books.google.dk/books/about/ELECTRIC_DRIVES.html?id=YikAs8Bp0yYC&printsec=frontcover&source=kp_read_button&redir_esc=y#v=onepage&q&f=true.

[3] Teresa Orlowska-Kowalska and Mateusz Dybkowski. "Industrial Drive Systems . Current State and Development Trends". In: *Power Electronics and Drives* 1.1 (2016), pp. 5–25. DOI: 10.5277/PED160101.

[4] R Saidur et al. *Applications of variable speed drive (VSD) in electrical motors energy savings*. Tech. rep. 2011, pp. 543–550. DOI: 10.1016/j.rser.2011.08.020. URL: http://kchbi.chtf.stuba.sk/upload_new/file/Miro/Procproblemyodovzdanezadania/Bajza/saidur2012.pdf.

[5] Allan R. Hambley. *Electrical Engineering Principles and Applications*. 6th ed. Pearson, 2014. ISBN: 978-0-273-79325-0.

[6] Giorgio Rizzoni. *Principles and Applicaions of Electrical Engineering*. 3rd ed. London, UK: McGraw-Hill, 2001. ISBN: 0072482885.

[7] Stephen J. Chapman. *Electric Machinery Fundamnetals*. 4th ed. McGraw-Hill, 2005. ISBN: 0-07-246523-9.

[8] Stefán Baldursson. "BLDC motor modelling and control – a Matlab/Simulink implementation". PhD thesis. Chalmers University of Technology, 2005. URL: http://webfiles.portal.chalmers.se/et/MSc/BaldurssonStefanMSc.pdf.

[9] Werner Leonhard. *Control of Electrical Drives*. 2nd ed. Springer, 1996. ISBN: 978-3-642-97648-3.

[10] Muhammad H. Rashid. *Power Electronics Handbook*. 1st ed. Academic Press, 2001, pp. 211–224. ISBN: 0-12-581650-2.

[11] V. M. Hernández-Guzmán, R. Silva-Ortigoza, and D. Muñoz-Carrillo. "Velocity Control of a Brushed DC-Motor Driven by a DC to DC Buck Power Converter". In: *International Journal of Innovative Computing, Information & Control: IJICIC* 11.2 (2015), pp. 509–521.

[12] T. K. Roy et al. "Adaptive Controller Design for Speed Control of DC Motors Driven by a DC-DC Buck Converter". In: *International Conference on Electrical, Computer and Communication Engineering (ECCE)*. Cox's Baza, Bangldesh, 2017. DOI: 10.1109/ECACE.2017.7912888.

[13] Ohio Electric Motors. *DC Motors Used in Electric Hoists, Reels and Winches - Ohio Electric Motors*. 2015. URL: http://www.ohioelectricmotors.com/2015/07/dc-motors-used-in-electric-hoists-reels-and-winches/.

[14] ElectroCraft. *Motors and Drives for Automated Guided Vehicles (AGV) - ElectroCraft*. 2019. URL: https://www.electrocraft.com/motors-for/mobile-platform-traction-systems/automated-guided-vehicles/.

[15] MICROMO. *Telescope Motion Control: DC Motor Applications | MICROMO*. 2019. URL: https://www.micromo.com/applications/optics-photonic-applications/telescope-motion-control.

[16] MICROMO. *Telemedicine Robots: DC Motor Applications | MICROMO*. 2019. URL: https://www.micromo.com/applications/medical-lab-automation-equipment/telemedicine-robots.

[17] Streetcrane. *Wire Rope Hoist, Creative Commons Attribution-Share Alike 4.0 International. Full Terms at: https://bit. ly/1SrbRBk*. 2015. URL: https://commons.wikimedia.org/wiki/File:Street_ZX_Wire_Rope_Hoist.jpg.

[18] MrGRA. *Automated Guided Vehicles charging in the hospitals basement level, Creative Commons Attribution-Share Alike 4.0 International. Full Terms at: https://bit.ly/1SrbRBk*. 2015. URL: https://en.m.wikipedia.org/wiki/File:Automated_Guided_Vehicles.jpg.

[19] Joachim Böcker. *Power Electronics*. Tech. rep. Paderborn University, 2017, p. 174. URL: https://ei.uni-paderborn.de/fileadmin/elektrotechnik/fg/lea/Lehre/LE/Dokumente/Skript_EN.pdf.

[20] Y Baghzouz. *Power Electronics Introduction*. Tech. rep. URL: http://www.egr.unlv.edu/~eebag/EE-442-642IntroductionF14.pdf.

[21] *LC Selection Guide for the DC-DC Synchronous Buck Converter*. Tech. rep. URL: http://onsemi.com.

[22] Timothy Hegarty. "Reduce buck-converter EMI and voltage stress by minimizing inductive parasitics". In: *Analog Applications Journal* (). URL: http://www.ti.com/lit/an/slyt682/slyt682.pdf.

[23] *Control theory*. Tech. rep. URL: http://www.basicknowledge101.com/pdf/control/Controltheory.pdf.

[24] Udani Mohit, Patil Nilesh, and D. R. Mehta. "Speed Control of DC Motor using Digital Control System". In: *International Journal of Engineering Research & Technology (IJERT)* 3.2 (2014), p. 3.

[25] Vance Vandoren. *To PID or not to PID - Control Engineering*. 2017. URL: https://www.controleng.com/articles/to-pid-or-not-to-pid/.

[26] David Greenfield. *When is PID Not the Answer?* 2010. URL: https://www.controleng.com/articles/when-is-pid-not-the-answer/.

[27] Greg McMillan. *When do I Use MPC instead of PID for Advanced Regulatory Control - Tips?* 2013. URL: https://www.controlglobal.com/blogs/controltalkblog/when-do-i-use-mpc-instead-of-pid-for-advanced-regulatory-control-tips/.

[28] Lennart Ljung and Torkel Glad. *Modeling of Dynamic Systems*. Prentice Hall, 1994. ISBN: 0-13-597097-0.

[29] Katsuhiko Ogata. *Discret-Time Control Systems*. Prentice Hall, 1995. ISBN: 0-13-034281-5.

[30] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback Control of Dynamic Systems*. 6th ed. Pearson, 2010. ISBN: 978-0-13-601969-5.

[31] Farid Golnaraghi and Benjamin C. Kuo. *Auomatic Control Systems*. 9th ed. John Wiley and Sons, Inc., 2010. ISBN: 978-0470-04896-2.

[32] Muhammad H. Rashid, Narendra Kumar, and Ashish R. Kulkarni. *Power Electronics*. 4th ed. Pearson, 2014. ISBN: 978-0-13-312590-0.

[33] P. Murphy et al. "Study of Digital Vs Analog Control". In: *Power Electronics Seminar Proceedings (CPES Center for Power Electronics Systems)*. 2002, pp. 203–206.

[34] A. Benlafkih, S. Krit, and M. C. Elidrissi. "A Comparative study of Analog and digital Controller On DC/DC Buck-Boost Converter Four Switch for Mobile Device Applications". In: *IJCSI International Journal of Computer Science Issues* 10.1 (2013), pp. 442–447.

[35] Eric Monmasson et al. "FPGAs in Industrial Control Applications". In: *IEEE Transactions on Industrial Informatics* 7.2 (2011), pp. 224–243. DOI: 10.1109/TII.2011.2123908.

[36] Rahul Pawar and N. R. Bhasme. "Application of PLC's for Automation of Processes in Industries". In: *International Journal of Engineering Research and Applications* 6.6 (2016), pp. 53–59.

[37] Austin Hughes. *Electric Motors and Drives*. 3rd ed. Elsevier, 2006. ISBN: 978-0-7506-4718-2.

[38] Ned Mohan, Tore M. Underland, and William P. Robbins. *Power Electronics: Converters, Applications and Design*. 2nd ed. John Wiley and Sons, Inc., 1995. ISBN: 0-471-58408-8.

[39] *An Engineering Guide to Position and Speed Feedback Devices for variable speed drives and servos*. Tech. rep. URL: www.controltechniques.com.

[40] *Lynxmotion 12V 450 rpm 10.18oz-in 1:5.2 Brushed DC Gear Motor w/ Encoder - RobotShop*. URL: https://www.robotshop.com/en/lynxmotion-12v-450-rpm-1018oz-in-152-brushed-dc-gear-motor-w--encoder.html.

[41] *Magnetic Angular Position Sensors Product & Applications Brochure phase detection SPINAXIS TM A REVOLUTION IN ANGLE SENSING*. Tech. rep. URL: https://media.monolithicpower.com/cms_document/product_literature/Magnetic_Angular_Position_Sensors_Brochure_Q4_2016_MPS.pdf.

[42] Joško Deur and Danijel Pavković. "Fundamentals of electrical drive controls". In: (2012).

[43] Daniel W. Hart. *Power Electronics*. Valparaiso, IN: McGraw-Hill, 2011. ISBN: 978-0-07-338067-4.
[44] Keiko Muro et al. "H-Bridge Buck-Boost Converter with Dual Feedforward Control". In: *Power Electronics and Drive Systems, 2009. PEDS 2009*. 2009, pp. 1002–1007. DOI: 10.1109/PEDS.2009.5385893.
[45] E. Hernández-Márquez et al. "A DC/DC Buck-Boost Converter-Inverter-DC Motor System: Sensorless Passivity-Based Control". In: *IEEE Access* 6.1 (2018). DOI: 10.1109/ACCESS.2018.2846614.
[46] *Robot Drive Systems Key Concepts to help you choose Motors, Wheels and Gearboxes*. Tech. rep. URL: https://curriculum.vexrobotics.com/curriculum.html.
[47] *Drive Essentials*. Tech. rep. URL: http://cyborgeagles.weebly.com/uploads/2/1/8/1/21816346/robot__drive_systems.pdf.
[48] *Motor Control Applications - Silicon Labs*. URL: https://www.silabs.com/products/mcu/motor-control.
[49] *SparkFun Current Sensor Breakout-ACS723*. Tech. rep. URL: https://www.sparkfun.com/products/13679.
[50] *2.1.3 The Hall Effect*. Tech. rep. URL: https://www.tf.uni-kiel.de/matwis/amat/mw2_ge/kap_2/backbone/r2_1_3.pdf.
[51] Honeywell. *Hall Effect Sensing and Application*. Tech. rep. URL: https://www.adrirobot.it/sensori/sensore_magnetico/Esempi_applicativi_sensore-magnetico.pdf.
[52] *Encoders & Resolvers*. Tech. rep. URL: www.dynapar.com.
[53] *AN1018: Using the Si72xx Hall-effect Magnetic Position Sensors*. Tech. rep. URL: https://www.silabs.com/documents/public/application-notes/an1018-si72xx-sensors.pdf.
[54] Platt Charles and Jansson Fredrik. *Encyclopedia of Electronic Components*. First. San Francisco, 2016, p. 228. ISBN: 978-1-4493-3431-4.
[55] N. Mohan. *Electric Drives: An Integrative Approach*. 2003. ISBN: 0-9715292-1-3.
[56] Karl Johan Åström and Richard M. Murray. *Feedback Systems*. Electronic. Princeton University Press, 2012. ISBN: 978-0-691-13576-2. URL: http://press.princeton.edu/titles/8701.html..
[57] O Sename. *Pole placement control O.Sename State feedback control Pole placement control: a state space approach Specifications Observer Observer-based control Integral Control Some important features Pole placement control: state space and polynomial approaches Lec*. Tech. rep. 2017. URL: www.gipsa-lab.fr/âĹijo.sename.
[58] *Controller Design by Pole placement*. Tech. rep. URL: https://fac.ksu.edu.sa/sites/default/files/control_design_by_pole_placement_0.pdf.
[59] Raymond T. Stefani et al. *Design of Feedback Control Systems*. 4th. Oxford University Press, 2002.
[60] *Control Tutorials for MATLAB and Simulink - Motor Position: State-Space Methods for Controller Design*. URL: http://ctms.engin.umich.edu/CTMS/index.php?example=MotorPosition&section=ControlStateSpace.
[61] Alberto Bemporad. *Lecture: Integral action in state feedback control Integral action in state feedback control*. Tech. rep. 2010. URL: http://cse.lab.imtlucca.it/~bemporad/teaching/ac/pdf/08-integral-action.pdf.
[62] Gene F. Franklin, J. David Powell, and Michael L. Workman. *Digital Control of Dynamic Systems*. 3rd ed. Addison Wesley Longman, 1998. ISBN: 0-201-33153-5.
[63] Liridon Xheladini, Abdullah Polat, and Lale T. Ergene. "Design of High Frequency Buck Converter for DC Motor Control". In: *ACEMP - OPTIM - ELECTROMOTION JOINT CONFERENCE*. 2015, pp. 768–773. DOI: 10.1109.
[64] International Rectifier. *AUTOMOTIVE MOSFET (IRFZ44VZ)*. URL: https://alltransistors.com/pdfview.php?doc=irfz44vzl_irfz44vzpbf_irfz44vzspbf.pdf&dire=_update.
[65] ON Semiconductor. *Switch-mode Power Rectifiers (MUR805G) - Datasheet*. URL: https://www.onsemi.com/pub/Collateral/MUR820-D.PDF.
[66] Robert W. Erickson and Dragan Maksimovic. *Fundamentals of Power Electronics- Seconfd Edition*. Second. 2000, pp. 213–222. ISBN: 0471017507. DOI: 10.1017/CBO9781107415324.004.
[67] Kotb Basem Tawfiq and Elwy El-kholy. *Speed Control of DC Motor Using DC-DC Buck Converter*. Tech. rep. March 2017. 2019.
[68] Taha Nurettin Gücin et al. "No Title". In: *2015 9th International Conference on Electrical and Electronics Engineering (ELECO)*. IEEE, 2015. DOI: 10.1109/ELECO.2015.7394556.

[69] R. M. T. Raja Ismail, M. A. Ahmad, and M. S. Ramli. "Speed Control of Buck-converter Driven Dc Motor Using LQR and PI: A Comparative Assessment". In: *2009 International Conference on Information Management and Engineering*. IEEE, 2009.

[70] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2nd. California Technical Publishing, 1999. ISBN: 0-9660176-6-8.

[71] J. Sachs. *Ten Little Algorithms, Part 2: The Single-Pole Low-Pass Filter*. 2015. URL: `https://www.embeddedrelated.com/showarticle/779.php`.

[72] D. Cooper, R. Rice, and J. Arbogast. *Tutorial: Cascade vs. Feed Forward for Improved Disturbance Rejection*. Tech. rep. 2004.

[73] C Maffezzoni, N Schiavoni, and G. Ferretti. "Robust design of cascade control". In: *IEEE Control Systems Magazine* 10.1 (1990), pp. 21–25. DOI: `10.1109/37.50665`.

[74] Sho Ito et al. "Two-loop Design for Dual-rate Cascade System". In: *Preprints of the 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control*, 2018, pp. 581–585.

Appendix A

Simulink Models

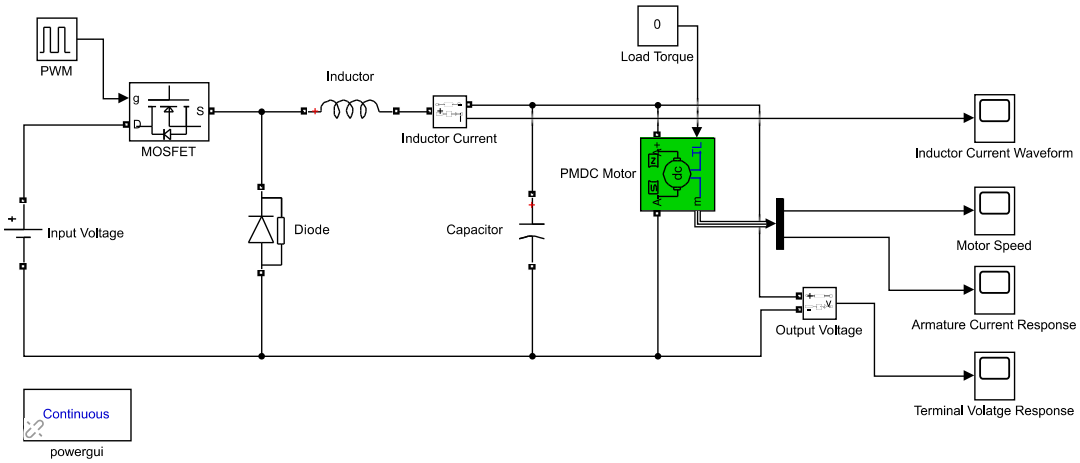

Figure A.1: Buck Converter SIMULINK Model

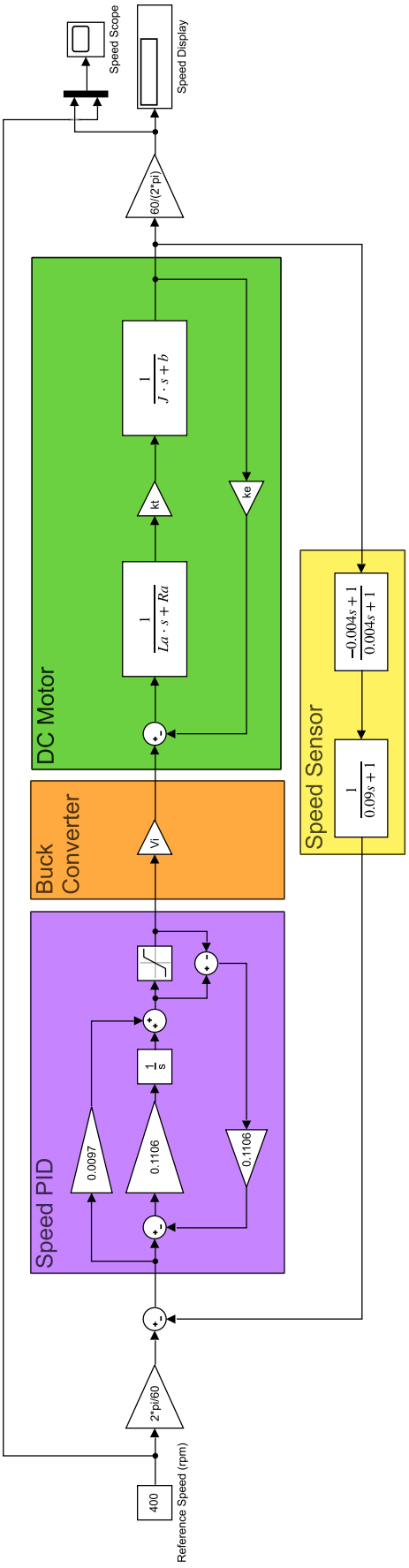

Figure A.2: Speed PID Simulink Model

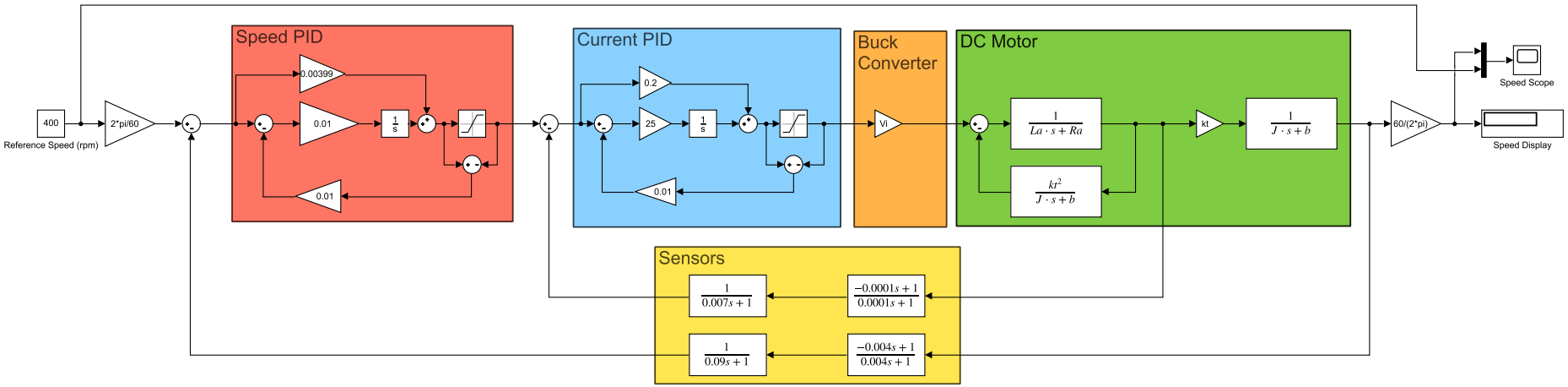

Figure A.3: Speed and Current Cascade PID Simulink Model

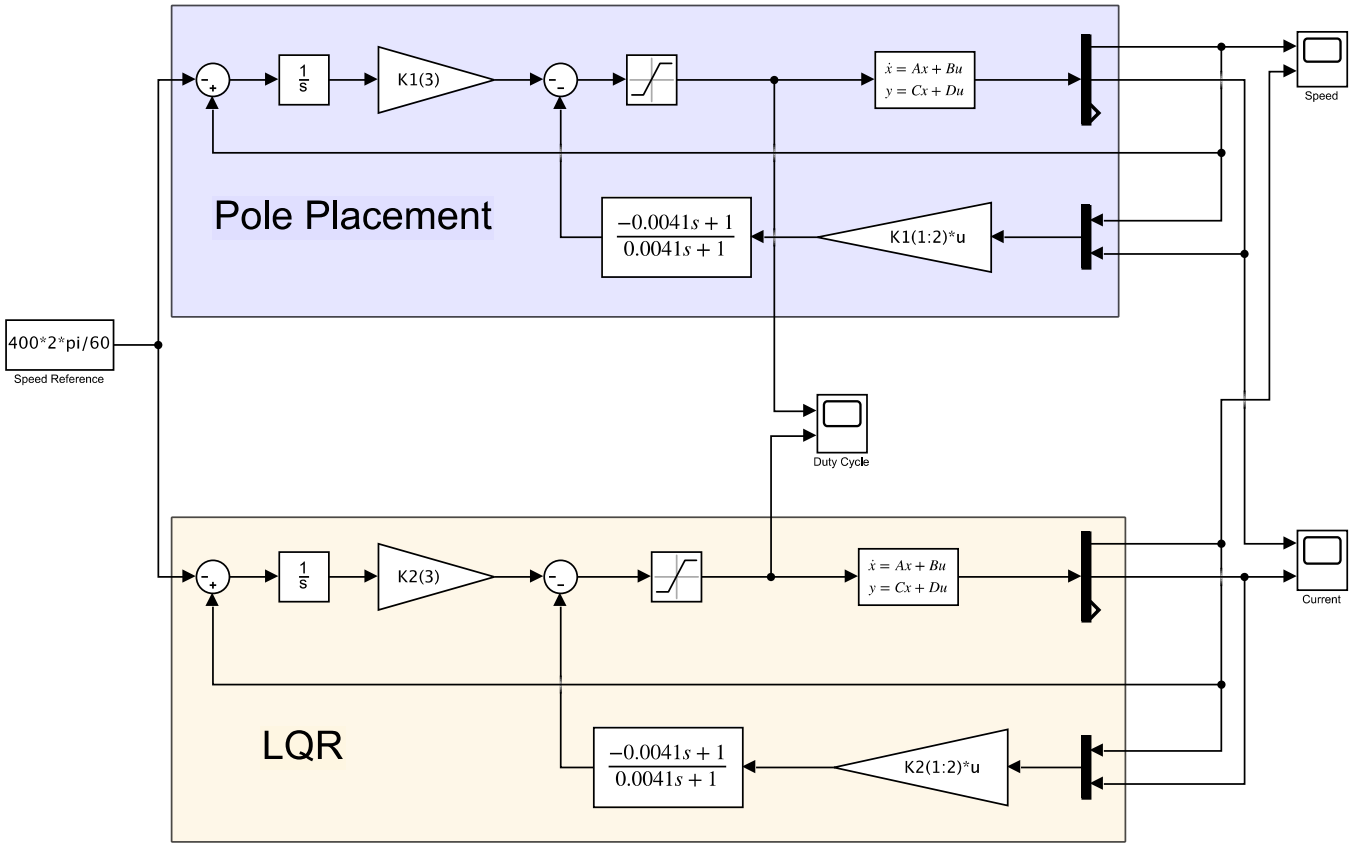

Figure A.4: Pole Placement and LQR Simulink Model

Appendix B

Arduino Code

B.1 PID: Speed Control

```
1
2  /* Libraries */
3
4  #include <TimerOne.h>
5
6  /* Constants and I/O Pins */
7
8  const int encoder_pin = 8;
9  const int current_pin = A5;
10 const int pwm_pin = 11;
11 const double encres = 77.9582;
12 const double dmin = 0.4167;
13 const double dmax = 0.99;
14 const double rpm = 60/(2*M_PI);
15 const double speedreference = 400/rpm;
16 const double kps = 0.0097;
17 const double kis = 0.1106;
18 const double kds = 0;
19 const double tau = 0.09;
20 long int starttime = 0;
21
22 /* Functions */
23
24 void runSpeedLoop();
25 void getSpeed();
26 void PWM();
27
28 /* Variables */
29
30 double w = 0;
31 double current = 0;
32 double hallvoltage = 0;
33 double duty = 0;
34 int firstflag = 1;
35
36 class PID {
37 private:
```

```
38   double kp, ki, kd, integral, derivative;
39   double error, lasterror, setpoint, pout, iout, dout, pidout, sensorval,
       stopintegral;
40   double elapsedtime, filtout;
41 public:
42   PID(double k_p, double k_i, double k_d, double set) {
43     kp = k_p;
44     ki = k_i;
45     kd = k_d;
46     setpoint = set;
47     error = 0;
48     pidout = -1;  //put this such that we do not stop integrating at the
       first iteration
49     stopintegral = 0;
50     filtout = 0;
51     elapsedtime = 0.0088;//0.0096;
52   }
53   void getSensorValue();
54   void updateErrorTime();
55   void calculatePIDout();
56   void saturation();
57   void antiWindup();
58   void lowPassFilter();
59   void toMatlab();
60 };
61
62 PID speedloop(kps, kis, kds, speedreference);
63
64 void setup() {
65   pinMode(encoder_pin, INPUT);
66   pinMode(current_pin, INPUT);
67   Serial.begin(115200);
68   Serial.println("**********************");
69   Timer1.initialize(50);
70 }
71
72 void loop() {
73   //double t0 = micros();
74   runSpeedLoop();
75   //Serial.println(micros()-t0);
76 }
77
78 void runSpeedLoop() {
79   speedloop.getSensorValue();
80   speedloop.updateErrorTime();
81   speedloop.antiWindup();
82   speedloop.calculatePIDout();
83   speedloop.saturation();
84   PWM();
85   speedloop.toMatlab(); //pidout = 0 will not be shown!!!
86 }
87
88 void PID :: getSensorValue() {
89   getSpeed();
90   sensorval = w;
91 }
92
```

```
93 void PID :: updateErrorTime() {
94   lowPassFilter();
95   sensorval = filtout;
96   lasterror = error;
97   error = setpoint - sensorval;
98 }
99
100 void PID :: lowPassFilter() {
101   double alpha = elapsedtime/tau;
102   filtout += alpha * (sensorval - filtout);
103 }
104
105 void PID :: antiWindup() {
106   if ((pidout == dmax && error > 0) || (pidout == dmin && error < 0)) {
107     stopintegral = 1;
108   } else {
109     stopintegral = 0;
110   }
111 }
112
113 void PID :: calculatePIDout() {
114   pout = kp * error;
115   if (!stopintegral) {
116     integral += error * elapsedtime;
117     iout = ki * integral;
118   }
119   derivative = (error - lasterror)/elapsedtime;
120   dout = kd * derivative;
121   pidout = pout + iout + dout;
122 }
123
124 void PID :: saturation() {
125   if (pidout > dmax) {
126     pidout = dmax;
127   } else if (pidout < dmin) {
128     pidout = dmin;
129   }
130   duty = pidout;
131 }
132
133 void PID :: toMatlab() {
134   if(firstflag){
135     starttime = millis();
136     firstflag = 0;
137   }
138   Serial.print(w*rpm,4);
139   Serial.print(", ");
140   Serial.print(filtout*rpm,4);
141   Serial.print(", ");
142   Serial.print(pout,7);
143   Serial.print(", ");
144   Serial.print(iout,4);
145   Serial.print(", ");
146   Serial.print(pidout,4);
147   Serial.print(", ");
148   Serial.print(error,4);
149   Serial.print(", ");
```

```
150   Serial.print((millis()-starttime)/1000.,4);
151   Serial.print(", ");
152   Serial.print(filtcurrent);
153   Serial.println(";");
154 }
155
156 void getSpeed() {
157   if (firstflag) {
158     w = 0;
159     return;
160   }
161   double f;
162   double period;
163   int i = 0;
164   double t0, t1;
165   int read_pin = digitalRead(encoder_pin);
166   while(read_pin == digitalRead(encoder_pin)) {}
167   t0 = micros();
168   for (;;) {
169     i++;
170     read_pin = digitalRead(encoder_pin);
171     while (read_pin == digitalRead(encoder_pin)) {}
172     t1 = micros();
173     if(t1 - t0 >= 5000){
174       break;
175     }
176   }
177   period = (t1-t0)/(i/2);
178   f = 1000000/period;
179  w = (2*M_PI*f/encres);
180 }
181
182 void PWM() {
183   Timer1.pwm(pwm_pin, duty*1023);
184 }
```

B.2 PID: Speed and Current Cascade Control

```
1
2 /* Libraries */
3
4 #include <TimerOne.h>
5
6 /* Constants and I/O Pins */
7 const int oscilospeed = 30;
8 const int oscilocurrent = 31;
9 const int encoder_pin = 8;
10 const int pwm_pin = 11;
11 const int current_pin = A4;
12 const double encres = 77.9582;
13 const double dmin = 0.4167;
14 const double dmax = 0.99;
15 const double imax = 2;
16 const double imin = 0;
17 const double speedreference = 400*2*M_PI/60;
18 const double kps = 0.004;
```

```
19 const double kis = 0.01;
20 const double kds = 0;
21 const double kpc = 0.2;
22 const double kic = 25;
23 const double kdc = 0;
24 const double tauc = 0.007;
25 const double taus = 0.09;
26 long int starttime = 0;
27
28 /* Functions */
29
30 void runSpeedLoops();
31 void getSpeed();
32 void getCurrent();
33 void PWM();
34 void runSpeed();
35 void runCurrent();
36
37 /* Variables */
38
39 double w = 0;
40 double current = 0;
41 double duty = 0;
42 double currentreference = 0;
43 double hallvoltage = 0;
44 int firstflag = 1;
45
46 class PID {
47 private:
48   double kp, ki, kd, ka, integral, derivative;
49   double error, lasterror, integralerror, setpoint, pout, iout, dout,
       pidout, sensorval, stopintegral;
50   double elapsedtime, filtout;
51   String loopname;
52 public:
53   PID(double k_p, double k_i, double k_d, double set, double elapsed_time,
        String loop_name) {
54     kp = k_p;
55     ki = k_i;
56     kd = k_d;
57     setpoint = set;
58     loopname = loop_name;
59     error = 0;
60     pidout = -1;
61     stopintegral = 0;
62     filtout = 0;
63     elapsedtime = elapsed_time;
64   }
65   void getSensorValue();
66   void updateErrorTime();
67   void calculatePIDout();
68   void saturation(double, double);
69   void antiWindup(double, double);
70   void lowPassFilter(double);
71   void toMatlab();
72 };
73
```

```
74 PID speedloop(kps, kis, kds, speedreference, 0.043, "speed");
75 PID currentloop(kpc, kic, kdc, currentreference, 0.00842, "current");
76
77 void setup() {
78   pinMode(encoder_pin, INPUT);
79   pinMode(current_pin, INPUT);
80   pinMode(oscilospeed,OUTPUT);
81   pinMode(oscilocurrent,OUTPUT);
82   Serial.begin(115200);
83   Serial.println("**********************");
84   Timer1.initialize(50);
85 }
86
87 void loop() {
88   runSpeed();
89   for(int i = 0; i < 5; i++) {
90     runCurrent();
91     if (i < 4) delayMicroseconds(8000);
92   }
93   currentloop.toMatlab();
94 }
95
96 void runSpeed(){
97   digitalWrite(oscilospeed,HIGH);
98   speedloop.getSensorValue();
99   speedloop.updateErrorTime();
100  speedloop.antiWindup(imin, imax);
101  speedloop.calculatePIDout();
102  speedloop.saturation(imin, imax);
103  speedloop.toMatlab();
104  digitalWrite(oscilospeed,LOW);
105 }
106
107 void runCurrent(){
108  digitalWrite(oscilocurrent,HIGH);
109  currentloop.getSensorValue();
110  currentloop.updateErrorTime();
111  currentloop.antiWindup(dmin, dmax);
112  currentloop.calculatePIDout();
113  currentloop.saturation(dmin, dmax);
114  PWM();
115  digitalWrite(oscilocurrent,LOW);
116 }
117
118 void PID :: getSensorValue() {
119   if (loopname == "speed") {
120     getSpeed();
121     sensorval = w;
122   } else if (loopname == "current") {
123     getCurrent();
124     sensorval = current;
125   }
126 }
127
128 void PID :: updateErrorTime() {
129   lasterror = error;
130   if (loopname == "speed") {
```

```
131     setpoint = speedreference;
132     lowPassFilter(taus);
133   } else if (loopname == "current") {
134     setpoint = currentreference;
135     lowPassFilter(tauc);
136   }
137   sensorval = filtout;
138   error = setpoint - sensorval;
139 }
140
141 void PID :: lowPassFilter(double tau) {
142   double alpha = elapsedtime/tau;
143   filtout += alpha * (sensorval - filtout);
144 }
145
146 void PID :: antiWindup(double downlim, double uplim) {
147   if ((pidout == uplim && error > 0) || (pidout == downlim && error < 0))
        {
148     stopintegral = 1;
149   } else {
150     stopintegral = 0;
151   }
152 }
153
154 void PID :: calculatePIDout() {
155   pout = kp * error;
156   if (!stopintegral) {
157     integral += error * elapsedtime;
158     iout = ki * integral;
159   }
160   derivative = (error - lasterror)/elapsedtime;
161   dout = kd * derivative;
162   pidout = pout + iout + dout;
163 }
164
165 void PID :: saturation(double downlim, double uplim) {
166   if (pidout > uplim) {
167     pidout = uplim;
168   } else if (pidout < downlim) {
169     pidout = downlim;
170   }
171   if (loopname == "speed") {
172     currentreference = pidout;
173   } else if (loopname == "current") {
174     duty = pidout;
175   }
176 }
177
178 void PID :: toMatlab() {
179   if(firstflag) {
180     starttime = millis();
181     firstflag = 0;
182   }
183   if (loopname == "speed") {
184     Serial.print(filtout * 60/(2*M_PI));
185     Serial.print(", ");
186   } else {
```

```
187     Serial.print(currentreference);
188     Serial.print(", ");
189     Serial.print(filtout);
190     Serial.print(", ");
191     Serial.print(millis());
192     Serial.println(";");
193   }
194 }
195
196 void getSpeed() {
197   if (firstflag) {
198     w = 0;
199     return;
200   }
201   double f;
202   double period;
203     int i = 0;
204   double t0, t1;
205   int read_pin = digitalRead(encoder_pin);
206   while(read_pin == digitalRead(encoder_pin)) {}
207   t0=micros();
208   for (;;) {
209     i++;
210     read_pin = digitalRead(encoder_pin);
211     while (read_pin == digitalRead(encoder_pin)) {}
212     t1 = micros();
213     if (t1 - t0 >= 5000) {
214        break;
215     }
216   }
217   period = (t1-t0)/(i/2);
218   f = 1000000/period;
219  w = (2*M_PI*f/encres);
220 }
221
222 void getCurrent() {
223   double tc0=micros();
224   hallvoltage += analogRead(current_pin);
225   current = ((hallvoltage - 508)/86.4);
226   hallvoltage = 0;
227   double tc1=micros()-tc0;
228 }
229
230 void PWM() {
231   Timer1.pwm(pwm_pin, duty*1023);
232 }
```

B.3 Pole Placement and LQR

```
1 /* Libraries */
2 #include <TimerOne.h>
3
4 /* Constants and I/O Pins */
5
6 const int encoder_pin = 8;
7 const int current_pin = A4;
```

```
8  const int pwm_pin = 11;
9  const int avgSamples = 1;
10 const float encres = 77.9582;
11 const float dmin = 0.4167;
12 const float dmax = 0.99;
13 const double speedreference = 400*2*M_PI/60;
14 const double K[3] = {0.0401, 0.2671, 1.4142/4};
15 // Pole placement gains: {0.0280, 0.1006, 1.0489/3.2}
16 const int Nbar = 0.0098;
17 const double cdelay = 0.0002;
18 const double sdelay = 0.008;
19 const double tauc = 0.007;
20 const double taus = 0.09;
21
22 /* Functions */
23
24 void getSpeed();
25 void getCurrent();
26 void PWM();
27 void filter_w(String);
28 void filter_c(String);
29 void printMeasurements();
30
31 /* Variables */
32
33 double w = 0;
34 double current = 0;
35 double duty = 0;
36 double hallvoltage = 0;
37 int firstflag = 1;
38 double filt_w = 0;
39 double filt_c = 0;
40 double integral = 0;
41
42 void setup() {
43   pinMode(encoder_pin, INPUT);
44   pinMode(current_pin, INPUT);
45   Serial.begin(115200);
46   Serial.println("**********************");
47   Timer1.initialize(50);
48 }
49
50 void loop() {
51   getSpeed();
52   getCurrent();
53   filter("speed");
54   filter("current");
55   printMeasurements();
56   integral += K[2] * (filt_w - speedreference) * (cdelay + sdelay);
57   duty = -integral - (filt_w * K[0]) - (filt_c * K[1]);
58   //Serial.println(duty);
59   if(duty > dmax){duty = dmax;}
60   else if(duty < dmin){duty = dmin;}
61  PWM();
62   //Serial.println(current);
63 }
64
```

```
65 void getSpeed() {
66   if (firstflag) {
67     w = 0;
68     firstflag = 0;
69     return;
70   }
71   double f;
72   double period;
73   int i = 0;
74   double t0, t1;
75   int read_pin = digitalRead(encoder_pin);
76   while(read_pin == digitalRead(encoder_pin)) {}
77   t0=micros();
78   for (;;) {
79     i++;
80     read_pin = digitalRead(encoder_pin);
81     while (read_pin == digitalRead(encoder_pin)) {}
82     t1 = micros();
83     if (t1 - t0 >= 5000) {
84        break;
85     }
86   }
87   period = (t1-t0)/(i/2);
88   f = 1000000/period;
89  w = (2*M_PI*f/encres);
90 }
91
92 void getCurrent() {
93   hallvoltage += analogRead(current_pin);
94   current = ((hallvoltage - 508)/86.4);
95   hallvoltage = 0;
96 }
97
98 void PWM() {
99   Timer1.pwm(pwm_pin, duty*1023);
100 }
101
102 void filter(String loopname) {
103   if(loopname == "speed"){
104     double alpha = (sdelay+cdelay)/taus;
105     filt_w += alpha * (w - filt_w);
106   }else{
107     double alpha = (sdelay+cdelay)/(tauc*5);
108     filt_c += alpha * (current - filt_c);
109   }
110 }
111
112 void printMeasurements(){
113   Serial.print(filt_w*60/(2*M_PI));
114   Serial.print(" ");
115   Serial.print(filt_c);
116   Serial.print(" ");
117   Serial.print(millis());
118   Serial.print(";");
119   Serial.println();
120 }
```