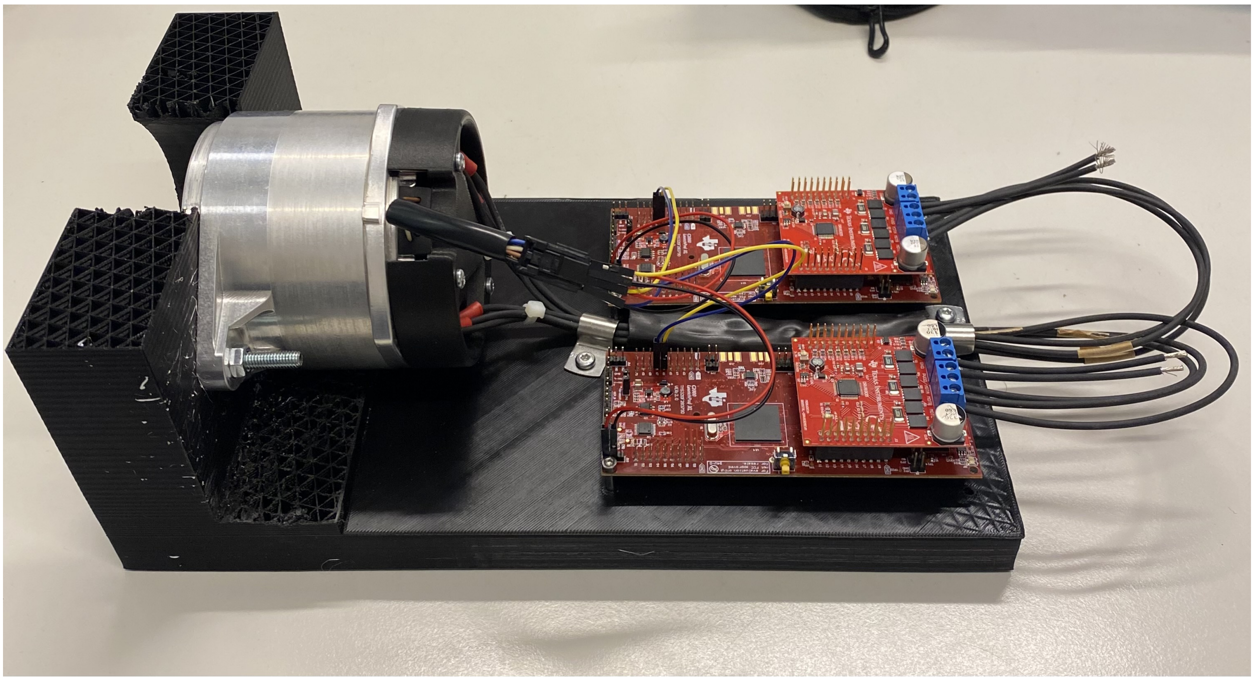
CHALMERS
UNIVERSITY OF TECHNOLOGY



Modelling and Control of a Dual Three-Phase Permanent Magnet Synchronous Motor

Master Thesis in System, Control and Mechatronics

ALBIN EMVIN
BENJAMIN FORSBÄCK

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

DEGREE PROJECT REPORT 2024

Modelling and Control of a Dual Three-Phase Permanent Magnet Synchronous Motor

Albin Emvin
Benjamin Forsbäck

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024

Modelling and Control of a Dual Three-Phase Permanent Magnet Synchronous Motor

Albin Emvin
Benjamin Forsbäck

Supervisors: Stefan Andersson (Autoliv), Martin Johansson (Autoliv)
Examiner: Prof. Jonas Fredriksson, Department of Electrical Engineering

Degree project report 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: A picture of the hardware rig that was constructed during the project. It includes the PMSM, microcontrollers and inverters

Typeset in LaTeX
Gothenburg, Sweden 2024

Abstract

This project aimed to study different torque control techniques for a multi-phase PMSM with an emphasis on minimising the torque ripple. Where torque ripple refers to noise in the torque, mainly due to inverter steps and sensor noise. Multiple Field Oriented Control (FOC) approaches, including Linear Quadratic Regulator (LQR), Gain Scheduled Controller (GSC), and Model Predictive Control (MPC), were tested in a simulation environment created in Simulink. The simulations showed similar performance for all controllers with the MPC having a slightly better performance. Different filter setups were also tested for their effect on the torque ripple. As the sensors equipped to measure angle and current had inherent noise, given sufficient computational power model-based filtering might improve the estimation of these variables. When tested in simulation, the angle filter and the current filter showed a large reduction of torque ripple, while filtering the angular velocity did not decrease the torque ripple by any significant margin. Some key factors contributing to the torque ripple were identified, e.g.: motor inductance, switching frequency, sensor accuracy, and possible improvements to these factors were discussed thoroughly in the concluding part of this thesis. Finally, one of the developed controllers was tested on the real motor, the behaviour was similar to the simulations, with very small torque ripple and responsive torque tracking.

Keywords: Six-Phase Permanent Magnet Synchronous Motor, Dual Three-Phase Motor, Torque Control, Torque ripple minimisation, Simulation and Hardware, Field Oriented Control, Linear Quadratic Regulator, Gain Scheduled Controller, Model Predictive Control, Kalman Filter.

Acknowledgements

We would like to thank Autoliv for the opportunity to do this project, and especially Stefan Andersson and Martin Johansson for supervising us. Furthermore, we would like to thank Prof. Jonas Fredriksson for the guidance during the thesis project. His insights has been invaluable for its success. We also would like to thank Stefan Lundberg for giving us feedback on the modelling of the PMSM.

Albin Emvin, Gothenburg, June 2024
Benjamin Forsbäck, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AC	Alternating Current
AC-MPC	Additive Complex-Model Predictive Controller
AS-MPC	Additive Simple-Model Predictive Controller
C-GSC	Complex-Gain Scheduled Controller
CKF	Cubature Kalman Filter
C-LQR	Complex-Linear Quadratic Controller
DC	Direct Current
EKF	Extended Kalman Filter
FOC	Field-Oriented Control
FS-MPC	Finite Set-Model Predictive Control
IPMSM	Interior Permanent Magnet Synchronous Motor
LPV	Linear Parametically Varying
LTI	Linear Time Invariant
LQR	Linear Quadratic Regulator
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MPC	Model Predictive Control
MTP	Multi Three Phase
PID	Proportional Integral Derivative
PMSM	Permanent Magnet Synchronous Motor
PWM	Pulse Width Modulation
GSC	Gain Scheduled Controller
S-GSC	Simple-Gain Scheduled Controller
S-LQR	Simple-Linear Quadratic Controller
SPMSM	Surface-Mounted Permanent Magnet Synchronous Motor
SPWM	Sinusoidal Pulse Width Modulation
SS-MPC	Separate Simple-Model Predicive Controller
SVPWM	Space Vector Pulse Width Modulation
VSI	Voltage Source Inverter

Contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Recently, multi-phase PMSMs have received more attention due to their higher torque and power density compared to their three-phase counterparts, as well as their redundancy aspects. Redundancy is vital for safety-critical products as in the case of a failure, the system can still operate and ensure the safety of the user. Thus, this project will create, evaluate, and compare control algorithms for multi-phase PMSM to investigate the capabilities that these types of motors could bring for the future. This thesis aims to study a dual three-phase (six-phase) PMSM with zero-degree phase shift.

The thesis will control the six-phase PMSM to a given torque reference. When creating a controller, it is important to evaluate its response time but also its ability to track the reference. However, there is a large difference between controlling torque and controlling speed or position. Such controllers are, in some sense, easier to develop, as integral action on the axle dynamics makes the desired dynamics smooth. This is not the case for torque controllers, as the developed torque is directly proportional to the motor's current, and therefore it is more susceptible to noise. Thereby, one of the major factors in evaluating a torque controller is to reduce the high-frequency noise, also known as torque ripple.

The motor will be used in a dynamic context. Therefore, feedback control will be used to control the motor. Specifically, different Field Oriented Control (FOC) approaches will be investigated. FOC works by controlling the orientation of the magnetic field generated by the windings in the stator. The magnetic field is dependent on the current flow in each winding, i.e., it is possible to control the field by controlling the phase currents. As there are 3 phases (3 dimensions) in a 2D space, it is also possible to use the Park transform to convert the 3 phases to two (direct and quadrature). Then, different types of ordinary feedback controllers can be inserted, such as Linear Quadratic Regulators (LQRs), Gain Scheduled Controllers (GSCs), and Model Predictive Controllers (MPCs), to control the direct and quadrature currents. These controllers generate a desired input voltage, which is sent to an inverter, where the appropriate phase voltages are created. This is the complete feedback loop.

The most used controller structure embedded in the FOC feedback loop is a simple PID controller, e.g. [1][2][3]. The PID is frequently used due to the simplicity of implementation and the non-linearities in the PMSM model. This has caused the PID controller to be a standard for motor control, research has been done on how to make them better and more reliable. With added cushioning, for example, with fuzzy controllers like in [1], or multiple degrees of freedom like in [2], PID controllers can be implemented with real efficiency.

Recent developments within PMSM control have generated a lot of new types of controllers, which linearise in specific regions and create a set of controllers. The active controller changes depending on the current state space vector [4]. GSCs work similarly, however a state in the state-space is parameterised, making the state-space linear but also dependent on the parameter. From this, a controller can be calculated dependent on the parameterised state [5]. Another popular approach is MPC which instead calculates the optimal control input given a certain receding horizon, the added benefit of MPC is that constraints on the input voltage to the motor can be explicitly stated to the optimiser [6]. Another important aspect in some applications is robustness, to perform well in the presence of some disturbance, this gives rise to various robust controllers [7]. Finally, the redundant aspects of multi-phase machines facilitate the possibility of operating smoothly even when some phases experience failure. Detecting the failure and updating the controller fast is vital, and special fault-tolerant controllers have been developed to handle this situation [8]. This project aims to implement some of these more sophisticated control strategies and create necessary filtering to minimise the apparent torque ripple in the produced torque.

1.2 Purpose and Goal

The purpose of this thesis is to evaluate torque tracking controllers for a six-phase PMSM. To achieve this purpose the goal for this thesis is to first create a realistic simulation environment. The simulation environment will be constructed using Simulink. Secondly, different controllers will be developed, tuned, tested, and compared to each other in the simulation environment. Thirdly, filters for decreasing sensor noise will be developed, integrated, and investigated for their performance benefits in the control system. The control systems have to be both fast but also be able to compensate for the torque ripple. The torque controllers will then be compared and analysed on their relative performance, advantages and disadvantages. Finally, one of the controllers will be implemented on real hardware and tested for deviations from the simulated behaviour.

1.3 Research Questions

The research questions which the project aims to answer are:

1. Which control strategy gives the best torque tracking performance according to the step response, $\mathcal{H}_2$ and $\mathcal{H}_\infty$ norm?

2. What are the main hardware and software factors contributing to torque ripple, and what techniques exist to minimise these factors?

1.4 Limitations

The Project will be limited to:

- Modelling and control of a zero-degree offset six-phase PMSM.
- The given sensor setup is an angle sensor for the rotor and phase current sensors.
- The project will be created in Matlab R2018b.

1.5 Thesis outline

1. *Introduction:* A preamble stating the purpose, goal, and scope of the thesis project.

2. *Theory:* Giving a short overview of the different methods and techniques used in the project.

3. *Method:* A summary of what has been developed in the project. Derivation of models, controllers, filters, PWM techniques and their implementation into Simulink are discussed.

4. *Results:* Different performance measures on the final systems are stated and shortly discussed.

5. *Discussion:* A deeper discussion on the results.

6. *Conclusion:* The main insights from the project are stated, and suggestions for further work are given.

2

Theory

To study motor control methods some prerequisites on motor geometry, PWM strategies and control needs to be known. This chapter aims to give a short introduction to some of the core concepts used in this project.

2.1 Six-phase PMSM

The motor studied in this project is a multi-phase PMSM with six phases. As for all electrical motors, the PMSM consists of two parts: the rotor and stator. The rotor is the rotating axis in the motor with permanent magnets attached to the surface. The stator is the static shell around the rotor where the phase windings are wound. A PMSM works on the principle of induction. The phase currents induce a magnetic field in the air gap between the rotor and stator, which will cause the permanent magnet in the rotor to align itself with this induced magnetic field [9]. The field produced by one phase is determined by the current and the inductance of the phase winding.

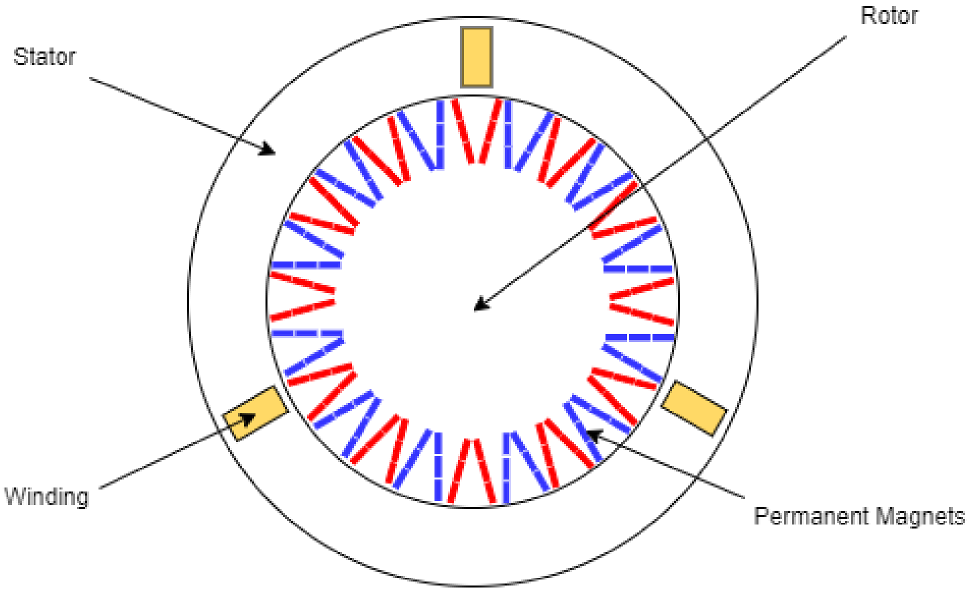


Figure 2.1: A simplified sketch of a PMSM.

2.1.1 Rotor Configuration

Circular rotor configurations can be categorised into two distinct categories: Surface-mounted Permanent Magnet Sychronous Motor (SPMSM) and Interior Permanent Magnet Sychronous Motor (IPMSM). The difference is the placement of the rotor magnets, which are adhered symmetrically to the surface in an SPMSM while they are built into the core of an IPMSM.

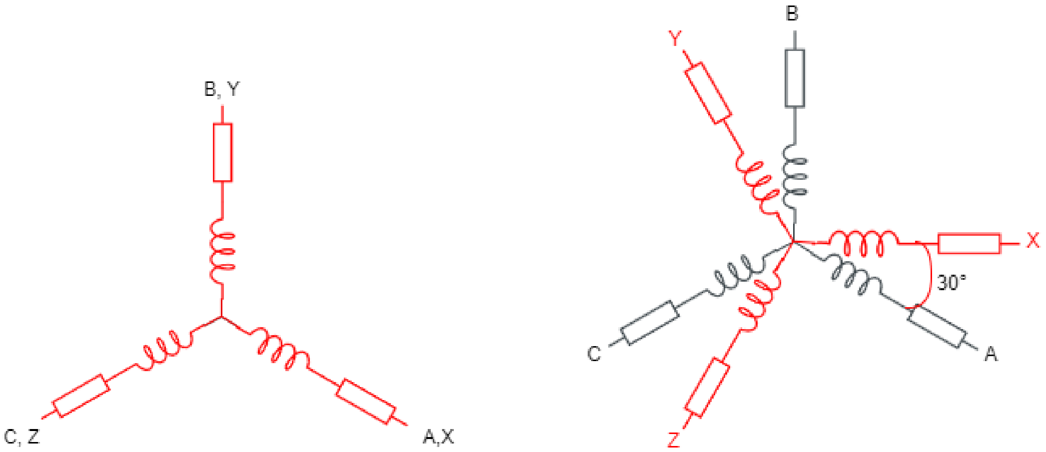
Generally, SPMSMs are made symmetrical, giving equal inductance over the whole surface. This makes operation smooth as the torque ripple is kept low. However, as the magnets are only adhered with glue to the rotor, this configuration is not suitable for high-speed operation when the centrifugal forces overcome the adhesive, resulting in dislodged magnets.

Conversely, IPMSMs offer safer high-speed operation. However, as the magnets are mounted inside the rotor, a symmetrical placement is hard to realise. This makes the magnetic field asymmetrical and can cause high torque ripple, resulting in worse torque tracking performance [10]. The motor considered in this project is an SPMSM.

2.1.2 Configurations of Phases

Multi-phase PMSMs with a number of phases divisible by 3 can be classified as multi-three-phase systems (MTP). MTP motors can be modelled as independent three-phase systems with some mutual inductance between the different three-phase systems. This gives the added benefit of redundancy, as the systems can be controlled separately.

The most common configuration for six-phase MTP is 0° and 30° between the two three-phase systems [11]. There are benefits for both types, which can be seen in the Table (2.1).



(a) 0 °phase shift **(b)** 30 °phase shift

Figure 2.2: The two most common dual three-phase layouts.

0°	30°
More freedom in winding layouts	Higher torque density Lower torque ripple under healthy conditons Cancellation of sixth harmonic

Table 2.1: Benefits with the different angle displacements, [11] and [12].

Furthermore, it is commonly stated that when using two separate controllers while having a phase shift might lead to instability issues. This can be solved by having a lower mutual coupling between the two sets of three-phase windings[12]. Mutual coupling is the electromagnetic field between the two sets of three-phase windings, and it is directly dependent on the physical characteristics of the motor.

2.1.3 Fault Compensation Techniques

Fault compensation techniques within multi-phase PMSM can be divided into two categories: fault-tolerant and redundant PMSM. Fault-tolerant PMSM will continue working with all operational phases even though one phase has failed. Conversely, the redundant PMSM works by having multiple sets of three-phase systems. Then, if a winding gets damaged, it will not use any of the phases in the faulty three-phase set. The performances of both methods are very similar, and both are used regularly [13].

2.1.4 Clarke and Park Transformations

For a three-phase system, the motor contains three currents, which are controlled with the three-phase voltages. However, as the rotational area is two-dimensional, a more intuitive understanding of PMSM can be gained by converting the phasors to the two-dimensional space also known as the dq - reference frame. In the dq-axes, currents and voltages have a more direct coupling to the behaviour of the motor than in the three-phase currents. To derive this transformation, we study the three-phase symmetrical Y-connected motor in Figure 2.3a with phase currents i_a, i_b and i_c.

Firstly, the three-phase current vectors need to be decomposed into orthogonal vectors. This subspace is called the $\alpha\beta$ - reference frame and the transformation is called the Clarke transformation. By transforming these currents into joint orthogonal vectors in the $\alpha\beta$ - plane, the sum of the currents and, hence, the induced magnetic field can be studied with more ease. The transformation can be found with simple trigonometry from Figure 2.3a:

$$\begin{bmatrix} i_\alpha \\ i_\beta \end{bmatrix} = \begin{bmatrix} cos(0) & cos(2\pi/3) & cos(4\pi/3) \\ sin(0) & sin(2\pi/3) & sin(4\pi/3) \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix}. \tag{2.1}$$

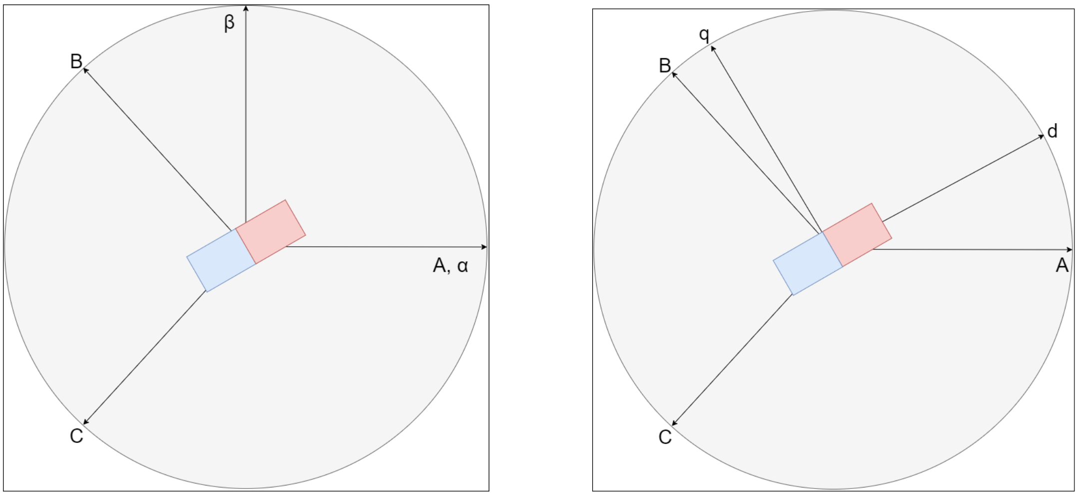
With the Clarke transformation the joint action of the phases can more readily be studied. However, in an SPMSM, the torque produced is proportional to the current perpendicular to the rotor position. To further simplify the complex workings of the motor, the currents can thus be projected onto a rotating reference frame with axes parallel and perpendicular to the rotor's position; these are called direct and quadrature axes, respectively (dq-axes). The transformation called the Park transformation, can be easily obtained from the $\alpha\beta$-space by a rotation matrix with the rotor's position. An illustration of the transformation is given in Fig. 2.3b. By adding the fact that the three-phase currents will sum to zero due to Kirchhoff's current law, and a coefficient to make the scaling factor between the phasors equal to 1, the transformation can be made invertible. The full transformation between abc and dq can be obtained as

$$T(\theta) = \frac{2}{3} \begin{bmatrix} cos(\theta) & cos(\theta - \frac{2\pi}{3}) & cos(\theta + \frac{2\pi}{3}) \\ -sin(\theta) & -sin(\theta - \frac{2\pi}{3}) & -sin(\theta + \frac{2\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.2}$$

$$\text{such that } i_{dq0} = T(\theta) i_{abc}, \tag{2.3}$$

where θ is the angle between the rotor and phase a. The inverse of T is also obtainable and can be written as

$$T^{-1}(\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 1 \\ cos(\theta - \frac{2\pi}{3}) & -sin(\theta - \frac{2\pi}{3}) & 1 \\ cos(\theta + \frac{2\pi}{3}) & -sin(\theta + \frac{2\pi}{3}) & 1 \end{bmatrix}. \tag{2.4}$$



(a) Clarke transformation **(b)** Park transformation

Figure 2.3: Space vector transformations.

2.2 Typical Control Methods

Electrical motors can be controlled in multiple ways. By creating a feedback loop from an angle and current sensor, the speed, position, torque and current in the motor can be controlled to some specific value or trajectory. Typically, electric motor control algorithms can be classified into two different groups, scalar control and vector control [14].

2.2.1 V/f - Control (Scalar Control)

Scalar control is a simple solution to the control problem which aims to control an average performance measure of the motor such as speed or torque. A well-studied technique of scalar motor control is V/f control. The ratio of phase voltage and the stator frequency is proportional to the developed torque in the motor, thus by controlling this ratio, a steady-state behaviour can be achieved. This technique works well for applications which do not require accurate dynamics and only operate in steady-state, however if transient behaviour is important V/f control cannot be used [10].

2.2.2 Field Oriented Control

Field-oriented control is a vector control method. The method utilises a transformation of phasors (e.g., currents and voltages) to the direct and quadrature axis (dq-axis) of the rotor. In dq variables, the following non-linear state-space model can be given for the motor dynamics, according to [15]:

$$\frac{di_d}{dt} = \frac{u_d}{L_d} - \frac{R_s}{L_d} i_d + \frac{n_p \omega L_q}{L_d} i_q, \tag{2.5}$$

$$\frac{di_q}{dt} = \frac{u_q}{L_q} - \frac{R_s}{L_q} i_q - \frac{n_p \omega L_d}{L_q} i_d - \frac{m}{L_q} \omega, \tag{2.6}$$

where u_d, u_q, i_d, i_q are voltages and currents along the d-and q-axis respectively. L_d, L_q inductances for the d and q equivalent circuit with R_s resistance, n_p the number of pole pairs in the rotor, ψ_m the permanent magnet field flux and finally ω is the rotor's angular velocity. The torque produced from a PMSM in dq variables can be derived from the total power input to the system

$$P_{in} = u_{abc}^T i_{abc} = \frac{3}{2}(u_d i_d + u_q i_q). \tag{2.7}$$

Transforming it to dq and substituting the state-space equations (2.5) and (2.6) gives

$$P_{in} = \frac{3}{2}(R_s(i_d^2 + i_q^2) + (L_d \frac{di_d}{dt} i_d + L_q \frac{di_q}{dt} i_q) + n_p \omega (L_d - L_q) i_d i_q + n_p \omega \psi_m i_q). \tag{2.8}$$

The first term of (2.8) refers to resistive losses in the machine, the second to losses in the production of the magnetic torque, and lastly, the last two terms correspond to the produced torque. Dividing these terms with the angular velocity will solve for the produced electromagnetic torque T_e,

$$T_e = \frac{3n_p}{2}(\psi_m i_q + (L_d - L_q) i_d i_q). \tag{2.9}$$

For an SPMSM, [10], $L_d = L_q$ and the expression simplifies to

$$T_e = \frac{3n_p}{2} \psi_m i_q. \tag{2.10}$$

Hence, to control the torque, speed, or position of the motor, only the quadrature current should be non-zero as current in the direct axis does not contribute to the produced torque and can be seen as a loss. The goal of FOC is then to control the d and q axis currents to obtain the desired dynamics. For an SPMSM, this means that the inner control loop for the currents has the following two objectives:

1. Keep the current vector parallel to the q-axis
2. Make the q-axis component of the current vector of the desired magnitude.

An example would be a torque controller, which has some reference torque T_r. Through (2.10) the reference q current is $i_{qr} = \frac{2}{3n_p \psi_m} T_r$ and the reference d current is 0 as this current does not produce any torque. The advantage of FOC is the decoupling of the currents, which makes the controller much easier to implement. The closed-loop can be seen in Figure 2.4, where a simple overview of the system can be seen.

2.2.3 PID

A PID controller works on the basis of a control error e. The error can be found as $e = r - y$ where r is a reference value that the system should operate at and y is a measurement taken on the same quantity. A P-controller can be devised by assigning a proportional gain K_p, making the input signal to the system $u = K_p e$.

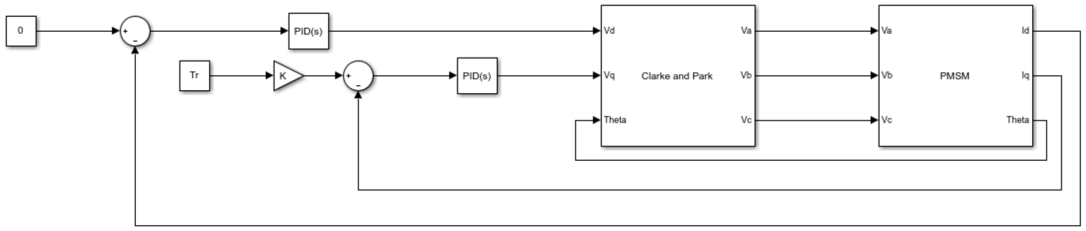


Figure 2.4: Example of closed-loop for torque control.

By increasing this gain, the response of the system will be faster. However, it could also become unstable depending on the system.

To suppress steady-state errors in the presence of some disturbance a integral gain K_i and a term can be added such that $u = K_p e + K_i \int e$. This makes a PI controller. By adding yet another term for the derivative of the control error with gain K_d, a PID controller can be formed $u = K_p e + K_i \int e + K_d \frac{d}{dt} e$.

2.2.4 Linear Quadratic Regulator

A Linear Quadratic regulator (LQR) is a model-based controller [16]. It requires a system with linear dynamics in the form

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx + Du, \end{aligned}$$

where A,B,C, and D are matrices describing the rate of change of the system states x and output y as a function of x and the input u. The goal of LQR is to minimise the cost function

$$J = \int_0^t x^T Q x + u^T R u \; dt$$

with weighting matrices Q and R. By tuning these matrices, more importance can be given to minimise state, input usage, or a combination of states or inputs. If (A, B) is stabilisable, and all states are observable, there exists a state feedback law $u = -Kx$ that minimises the cost function J. The feedback gain K can be found using the Riccati equation

$$AP + PA^T + Q + PBR^{-1}B^T P^T = 0. \tag{2.11}$$

Solving the Riccati equation for P, the controller gain is given as $K = R^{-1}B^T P$.

2.2.5 LPV Models and Scheduled Controllers

Linear Parametrically Varying (LPV) models are an extension of Linear Time Invariant (LTI) models, where for a given parameter p, the LPV model evaluates to an LTI model,

$$\begin{aligned} \dot{x} &= A(p)x + B(p)u, \\ y &= C(p)x + D(p)u. \end{aligned} \tag{2.12}$$

Therefore, such a model can be used to handle some non-linearities in the system by reassigning some state to be a parameter. Through the same analysis that leads to LQR for LTI systems, $K(p)$, can be created that takes the scheduling parameter p as input, i.e., it schedules the controller based on the parameter p, a so-called GSC.

2.2.6 Model Predictive Control

Model Predictive Control or MPC for short, is also a model-based controller, [17]. It uses the model to simulate the system a set number of samples into the future, and evaluates what inputs minimises a given cost function. This results in an optimisation problem, finding the inputs which result in the lowest cost given some current state and the state-space model.

As the motor will be controlled by an inverter, only a finite number of phase voltage combinations can be achieved for a single sample time. This means that it is possible to implement a Finite Set-Model Predictive Control (FS-MPC) algorithm. It tests all possible input combinations in a predictive model and compares them with a given cost function. Similar methods have been tested in [17],[18] and [19].

The benefit of MPC compared to LQR is faster performance, as the controller can plan for eventual changes in advance. MPC also offers the ability to explicitly set constraints on inputs and states to the optimiser. Conversely, LQR offers no guarantees on constraints and can only be tuned to be within some bounds for a predefined trajectory.

However, predicting far into the future is computationally heavy and problems can arise with delayed control signals. That is, the controller takes too much time to find the solution. Using MPCs with a large horizon also creates a new problem, finding a suitable reference signal for values far into the future. This can be an impossible challenge for some systems as some reference signals cannot be modelled and require information not obtained yet.

2.3 Observers

The ability to measure all states of a given system is not usual, and to apply a state feedback controller, the full state vector must be available. The problem can then be formulated as: *Given the input to the system, the previous state, the state-space model and some measured signal, what is the current state?.* For linear systems, the Kalman filter offers the estimate that minimises the given variation of the estimation error.

2.3.1 Kalman filter

Kalman filter is a type of linear filtering, see e.g., [20] and [21]. The Kalman filter assumes:

- A linear discrete state space model.

- That the noise is zero mean Gaussian distributed.

It works by first defining the discrete state space model representing the system

$$\begin{aligned} x_k &= Ax_{k-1} + u_{k-1} + q_k, \\ y_k &= Hx_k + r_k, \end{aligned} \tag{2.13}$$

where x_k is the state, y_k is the measurement, u_k is the input, A is the transition matrix, H is the measurement model, $q_k \sim N(0, Q)$ is the process noise and $r_k \sim N(0, R)$ is the measurement noise. Kalman filters are an iterative process, where in each time step a prediction and update is done. The iterative process can be seen in Figure 2.5. The prediction step uses the previous state estimation to make a prediction of the current state using the process model. The state estimation is assumed to be normally distributed, and as the process model is linear, the prediction is also Gaussian. The prediction forms the prior to the current estimate. Based on the prior, a likelihood of making the measurement can be found. By combining the information of the prior and likelihood, a new estimate called the posterior is achieved through the relationship

$$\textit{posterior} \propto \textit{prior} \times \textit{likelihood}.$$

This posterior is then fed to the prediction step for the next time sample.

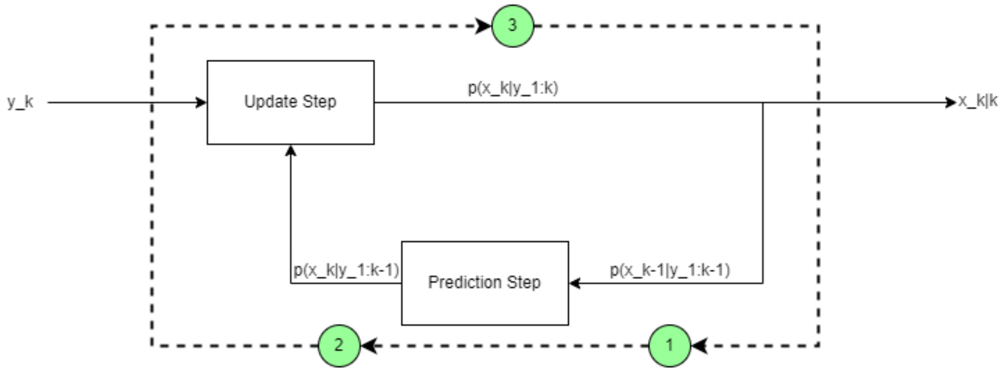


Figure 2.5: Kalman filters iterative process.

2.3.2 Extended Kalman filter

The Extended Kalman filter (EKF) works in a similar manner to the regular Kalman filter. Except that it handles nonlinear models. The Extended Kalman filter assumes:

- That the noise is zero mean Gaussian distributed.

It works in a similar way as the Kalman filter by first defining the model

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}) + q_k, \\ y_k &= h(x_k) + r_k, \end{aligned} \tag{2.14}$$

where x_k is the state, y_k is the measurement, u_k is the input, $f(x, u)$ is the non-linear process model, $h(x)$ is the non-linear measurement model, $q_k \sim N(0, Q)$ is the process noise and $r_k \sim N(0, R)$ is the measurement noise. As the model is not linear, the EKF solves this by linearising in each step and based on the linearised model, creates a new A/H for that step [21].

2.3.3 Cubature Kalman filter

The Cubature Kalman filter (CKF) is also an extension of the Kalman filter. It is computationally heavier than EKF and can, therefore, be unsuitable to implement in time-sensitive use cases. However, it handles highly non-linear systems better than the EKF which can become unstable in some parts of the state-space. The CKF assumes that:

- The predictive density of the joint state-measurement random variable is Gaussian.

The model is defined the same as in the EKF, (2.14). The CKF works by, in each step, sampling points surrounding the current state (sigma points). Then, it uses sampling variance, sampling covariance and mean to calculate the prediction and update step [21].

2.4 Inverter

To control AC powered motors during variable speed operation a DC-AC link, which transforms the battery voltage to the desired AC waveforms is needed. This is done by controlling an inverter circuit such as a Voltage Source Inverter (VSI) [22].

The most common device to achieve this is the MOSFET as it can achieve a very high switching frequency. Commonly a VSI consists of multiple MOSFETs connected like in Figure 2.6. By controlling the gating signals $G1 - G6$, the output phases can take a value between $-V_{DC}/2$ and $V_{DC}/2$, e.g., by keeping $G1$ closed and $G2$ open phase A will become $V_{DC}/2$. Each parallel system in Figure 2.6 controls a phase and is also known as a inverter leg. The inverter shown in the figure is the most common type of VSI, and it is called the two-level VSI due to the fact that each inverter leg has two possible voltages. Other more complicated inverter structures exist, such as three or four-level inverters [10].

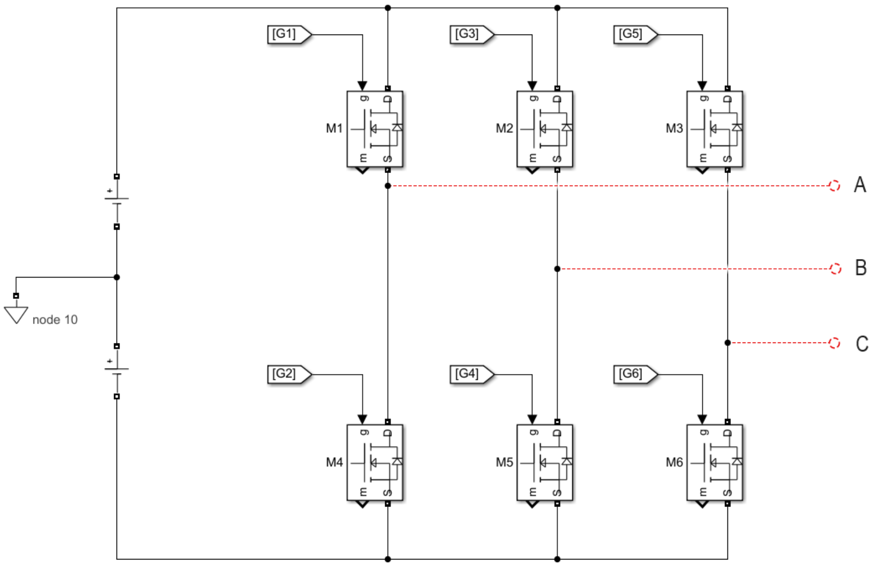


Figure 2.6: Three - Phase VSI.

2.5 PWM - Pulse Width Modulation

PWM is the technique used to convert a desired input voltage phasor from the controller to the actual gating signals in the inverter. This is done by finding an appropriate translation in terms of on and off switches for the inverter legs during a short time period. There are many different PWM methods, for example, SVPWM and SPWM [23]. The only PWM method evaluated in this project is SVPWM.

2.5.1 Space Vector Pulse Width Modulation

Space Vector Pulse Width Modulation (SVPWM) is a type of PWM that divides the $\alpha\beta$-plane into sectors, and then, based on the sector, it calculates the time (width) of each PWM signal. This method is commonly used for torque control. The SVPWM method can be summarised in 3 steps.

1. Calculate sector
2. Calculate the switch time
3. Generate PWM signals from switch time

The rotational area of the motor can be subdivided into 6 equally sized sectors, where each dividing vector is a Voltage vector for a MOSFET combination in the inverter, as in Figure 2.7.

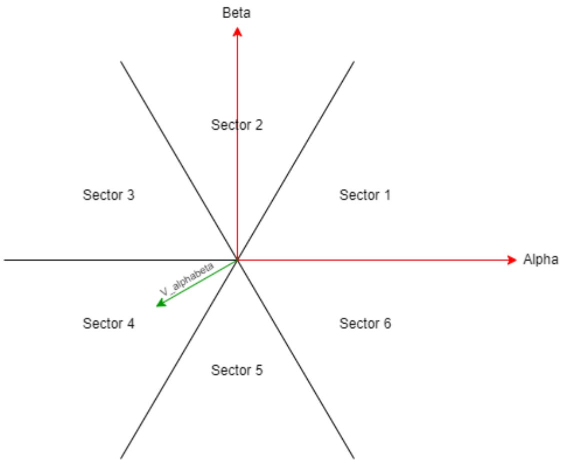


Figure 2.7: Sectors partition in SVPWM 2-level.

The next step is to calculate the switching times for each MOSFET. The switching time, or the duty time, is the duration that each switch should be on to recreate V_{ref}. The problem is further simplified as there are only 6 predefined voltage vectors V1-V6. Using the projection of V_{ref} onto the voltage vectors, the duration spent on each one can be deduced. An example of this is shown in Figure 2.8.

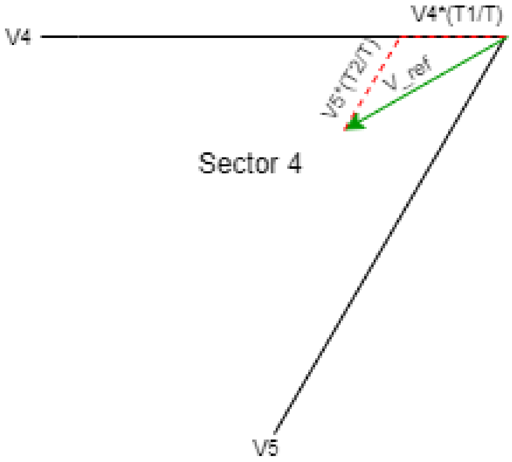


Figure 2.8: SVPWM time calculation.

The final transformation from the desired voltage to the gating signals is to convert the duration on each predefined voltage vector to binary values for each MOSFET. To create these values, any of the triangle waveforms in Figure 2.9 can be used. These waveforms, known as left-aligned, right-aligned, centre-aligned, and right-left-aligned, can be compared to the value $P = PWM\_Period - Duty\_Time$ like in Figure 2.10. If the current value of the waveform is larger than P, the resulting binary value will be 1, and the switch should be closed. Otherwise, the value will be 0, and the switch should be open.

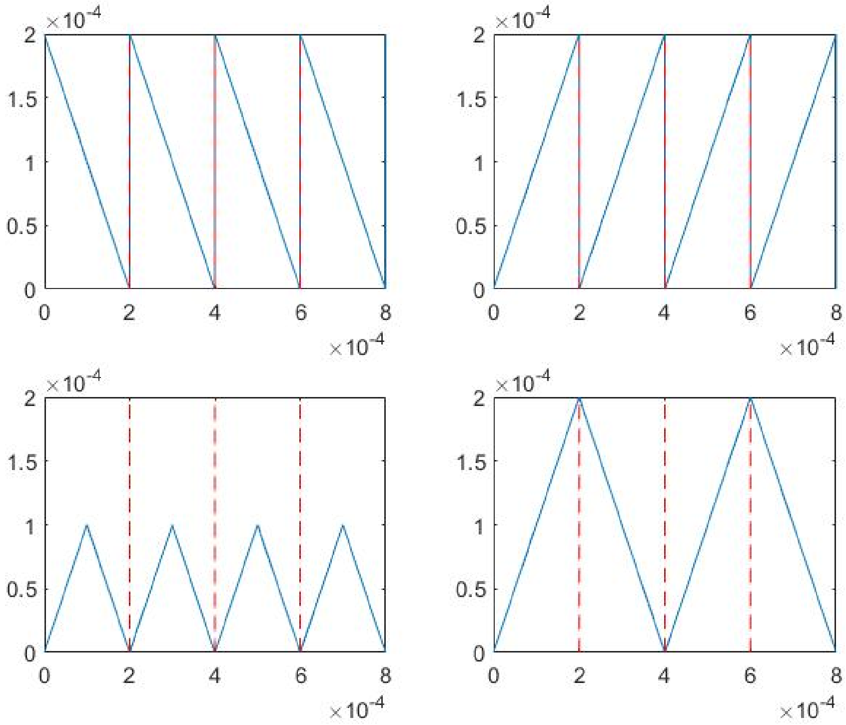


Figure 2.9: The four different triangle waveforms for SVPWM (Top left: left-aligned, Top right: right-aligned, Bottom left: centre-aligned, and Bottom right: right-left-aligned).

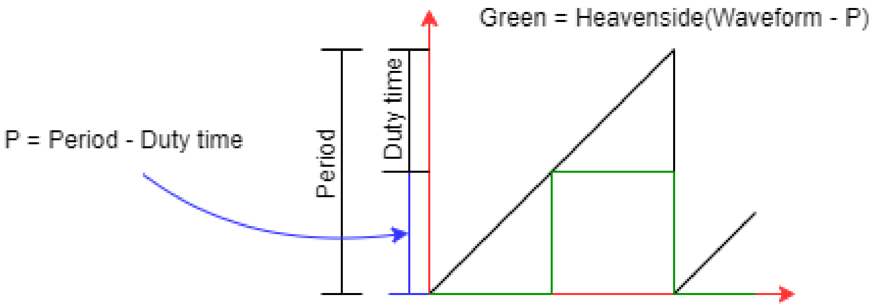


Figure 2.10: Conversion from duty time to a binary that is on for that time.

Implementing the triangle waveforms from Figure 2.9 on three different duty times results in the pulses in Figure 2.11, [24].

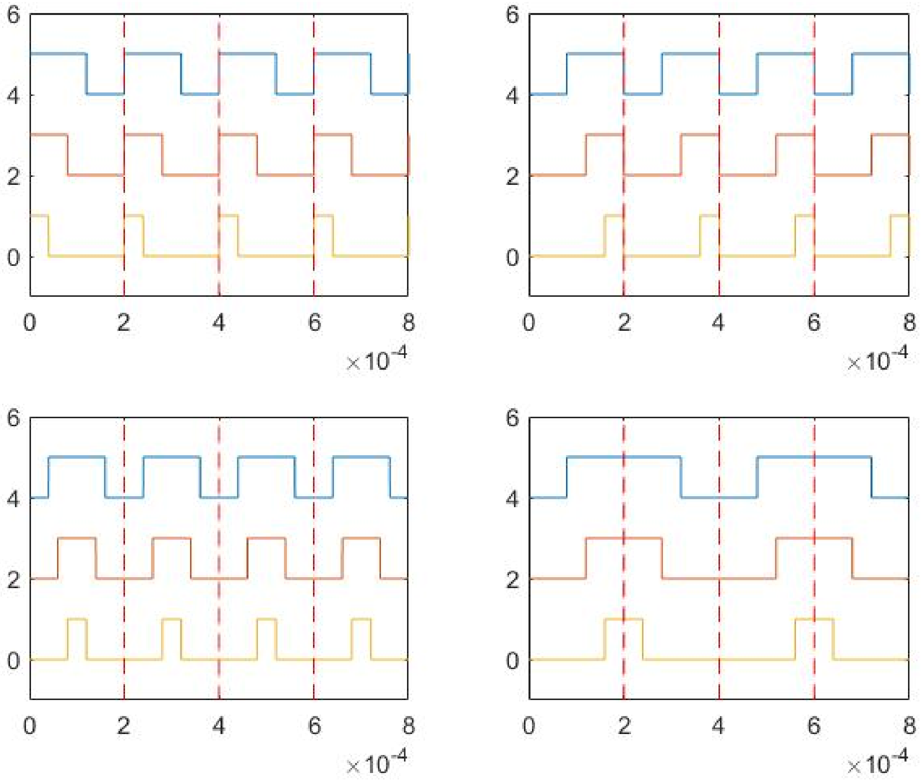


Figure 2.11: Example of the PWM signals for the four different triangle waveforms. It has the same order as in Figure 2.9.

2.6 Dynamometer

The axle of the motor will be coupled to a dynamometer (a DC motor) to make simulations more realistic and reliable. If no connection is made the PMSM motor would be able to spin freely during simulations. As the aim of the project is to create and evaluate torque tracking controllers, a freely spinning rotor could create problems, e.g., given torque, the angular velocity will increase until it reaches the endpoint of the motor load curve. This will severely affect the torque tracking performance. Thus it is clear to see that a dynamometer is justified to get reasonable performance from a torque tracking controller. As mentioned above, the dynamometer will be implemented as a DC motor with a simple LQR controller to control the axle to maintain a certain angle. The thesis will describe this motor and its controller.

3

Method

The method for completing this project can be briefly summarised as:

1. Make a state-space model of the PMSM from first principle to simulate its behaviour.
2. Create the PWM logic that determines the inverter configuration.
3. Develop different model-based control strategies for torque control of the PMSM.
4. Implement improvements for different sources of noise in the produced torque.
5. Test final solution on hardware.

This chapter will thoroughly examine how all these steps were fulfilled. Showcasing the complete methodology for the produced simulation models and the wiring of the hardware rig. In Figures 3.1-3.3, the three types of feedback loops that were developed in the project are shown, and each of the elements in these sketches will be explained in the following chapter.

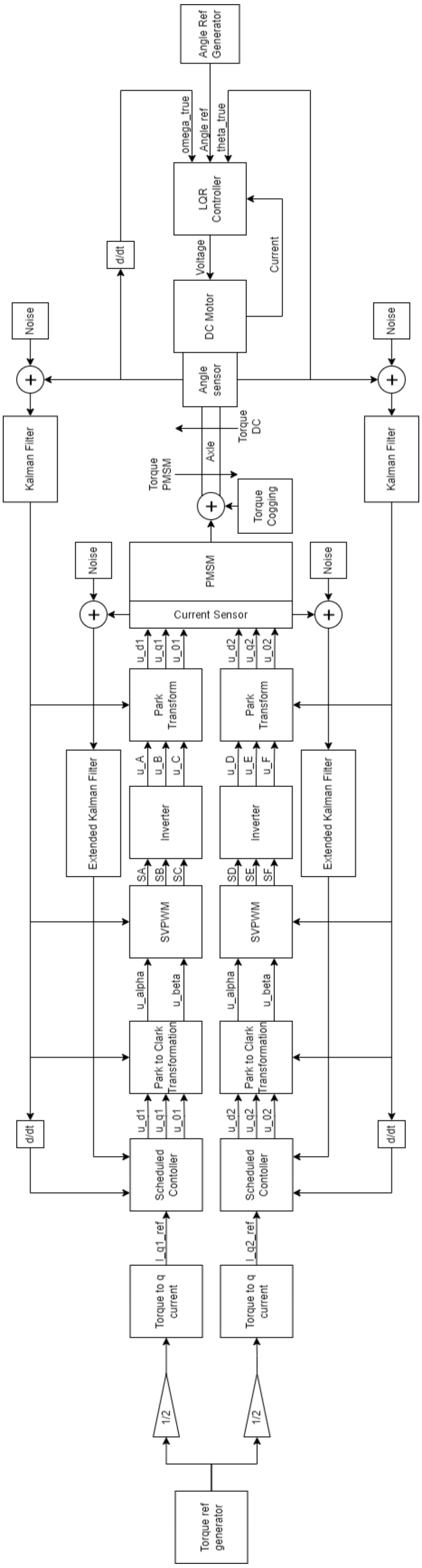


Figure 3.1: Overview of the GSC feedback loop with auxiliary filters and actuators.

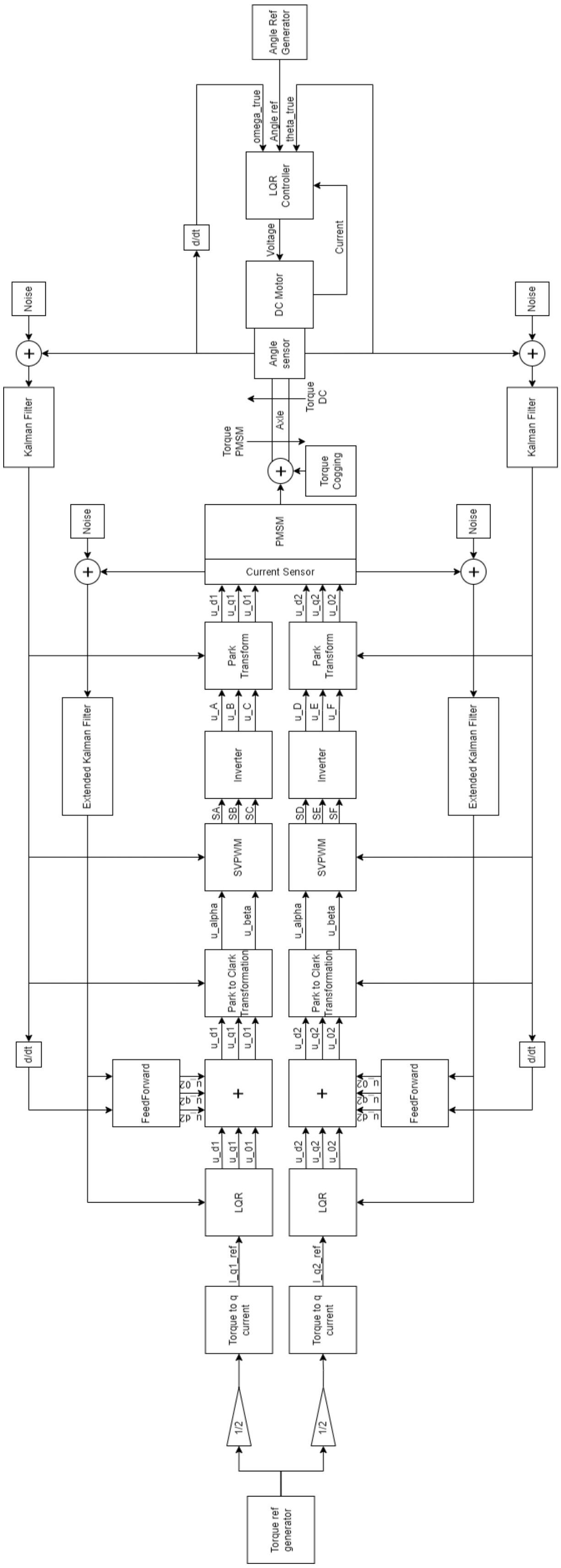


Figure 3.2: Overview of the LQR feedback loop with auxiliary filters and actuators.

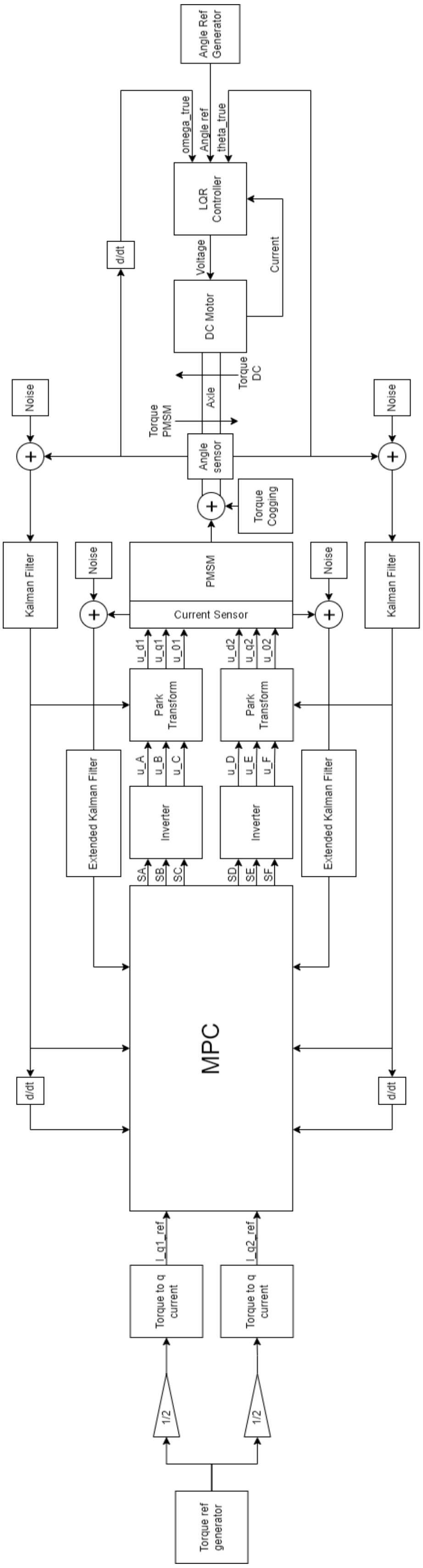


Figure 3.3: Overview of the MPC feedback loop with auxiliary filters and actuators.

3.1 Modelling of a Six-Phase PMSM

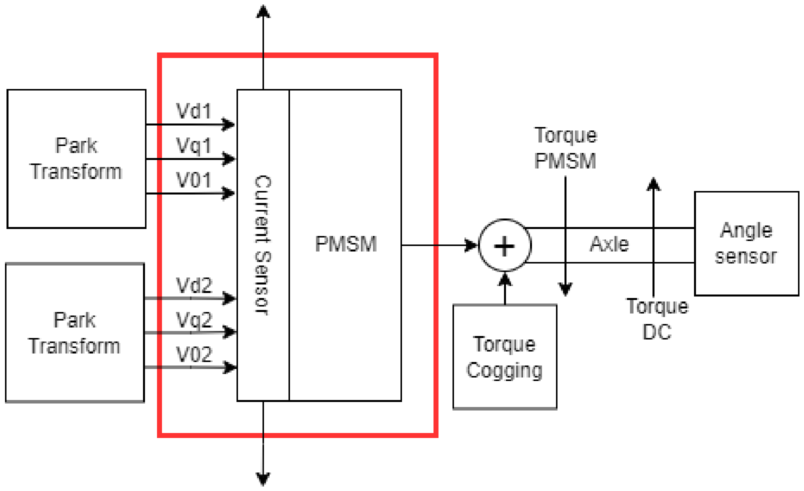


Figure 3.4: PMSM in the system.

The motor contained six phases a, b, c, d, e, f. Where a, b, c made a symmetrical three-phase system and d, e, f another three-phase system, with zero degree offset between the two systems. Under the assumption of sinusoidal back emf, an absence of Eddy currents, and no saturation of the system. The phase currents $i = \begin{bmatrix} i_a & i_b & i_c & i_d & i_e & i_f \end{bmatrix}$, the phase voltages $u = \begin{bmatrix} u_a & u_b & u_c & u_d & u_e & u_f \end{bmatrix}$ and the phase resistances R obeyed the relationship

$$u = Ri + \frac{d\psi}{dt}, \tag{3.1}$$

where ψ was the magnetic field flux of the motor. ψ was compromised of both an induced field due to currents in the stator windings and the permanent magnet in the rotor. This could then be written as

$$\psi = \begin{bmatrix} L_{aa} & L_{ab} & L_{ac} & L_{ad} & L_{ae} & L_{af} \\ L_{ba} & L_{ab} & L_{bc} & L_{bd} & L_{be} & L_{bf} \\ L_{ca} & L_{cb} & L_{cc} & L_{cd} & L_{ce} & L_{cf} \\ L_{da} & L_{db} & L_{dc} & L_{dd} & L_{de} & L_{df} \\ L_{ea} & L_{eb} & L_{ec} & L_{ed} & L_{ee} & L_{ef} \\ L_{fa} & L_{fb} & L_{fc} & L_{fd} & L_{fe} & L_{ff} \end{bmatrix} i + \begin{bmatrix} \psi_m cos(\theta_e) \\ \psi_m cos(\theta_e - \frac{2\pi}{3}) \\ \psi_m cos(\theta_e + \frac{2\pi}{3}) \\ \psi_m cos(\theta_e) \\ \psi_m cos(\theta_e - \frac{2\pi}{3}) \\ \psi_m cos(\theta_e + \frac{2\pi}{3}) \end{bmatrix}, \tag{3.2}$$

where ψ_m was the permanent magnet magnetic field flux and θ_e was the electrical angle. Assuming a symmetrical arrangement and equal phase inductances, the self-inductance could be simplified to $L_1 = L_{aa} = L_{bb} = ...$ As the systems were not magnetically isolated, phase a had a strong magnetic coupling with d and vice versa. Thus $L_2 = L_{ad} = L_{da} = L_{be} = ...$ Lastly, all other inductances, such as L_{ab}, produced a negative flux in the i_a direction when a positive phase current i_b was applied, therefore the mutual inductance $L_3 = -L_{ab} = -L_{ba} = -L_{ac} = ...$ This meant ψ could be simplified to

$$\psi = \begin{bmatrix} L_1 & -L_3 & -L_3 & L_2 & -L_3 & -L_3 \\ -L_3 & L_1 & -L_3 & -L_3 & L_2 & -L_3 \\ -L_3 & -L_3 & L_1 & -L_3 & -L_3 & L_2 \\ L_2 & -L_3 & -L_3 & L_1 & -L_3 & -L_3 \\ -L_3 & L_2 & -L_3 & -L_3 & L_1 & -L_3 \\ -L_3 & -L_3 & L_2 & -L_3 & -L_3 & L_1 \end{bmatrix} i + \begin{bmatrix} \psi_m cos(\theta_e) \\ \psi_m cos(\theta_e - \frac{2\pi}{3}) \\ \psi_m cos(\theta_e + \frac{2\pi}{3}) \\ \psi_m cos(\theta_e) \\ \psi_m cos(\theta_e - \frac{2\pi}{3}) \\ \psi_m cos(\theta_e + \frac{2\pi}{3}) \end{bmatrix}. \tag{3.3}$$

With the three-phase currents $i_a + i_b + i_c = 0$ and $i_d + i_e + i_f = 0$ summing to zero, the expression in (3.3) could be simplified further. As $L_3 i_a = -L_3 i_b - L_3 i_c$ this implied that $L_1 i_a - L_3 i_b - L_3 i_c = (L_1 + L_3) i_a$. Substituting into (3.3) gave

$$\psi = \begin{bmatrix} L_1 + L_3 & 0 & 0 & L_2 + L_3 & 0 & 0 \\ 0 & L_1 + L_3 & 0 & 0 & L_2 + L_3 & 0 \\ 0 & 0 & L_1 + L_3 & 0 & 0 & L_2 + L_3 \\ L_2 + L_3 & 0 & 0 & L_1 + L_3 & 0 & 0 \\ 0 & L_2 + L_3 & 0 & 0 & L_1 + L_3 & 0 \\ 0 & 0 & L_2 + L_3 & 0 & 0 & L_1 + L_3 \end{bmatrix} i + \begin{bmatrix} \psi_m cos(\theta_e) \\ \psi_m cos(\theta_e - \frac{2\pi}{3}) \\ \psi_m cos(\theta_e + \frac{2\pi}{3}) \\ \psi_m cos(\theta_e) \\ \psi_m cos(\theta_e - \frac{2\pi}{3}) \\ \psi_m cos(\theta_e + \frac{2\pi}{3}) \end{bmatrix}. \tag{3.4}$$

Transforming (3.1) to the $\alpha\beta$-subspace using the Clarke Transform yielded

$$u_s^s = R i_s^s + \frac{d\psi_s^s}{dt}, \tag{3.5}$$

where $u_s^s = \begin{bmatrix} u_s^{s1} & u_s^{s2} \end{bmatrix}^T$ and the u_s^{s1} and u_s^{s2} were complex valued, as were the currents. The magnetic field ψ_s^s could then be described as

$$\psi_s^s = \begin{bmatrix} L_1 + L_3 & L_2 + L_3 \\ L_2 + L_3 & L_1 + L_3 \end{bmatrix} i_s^s + \begin{bmatrix} \psi_m e^{j\theta_e} \\ \psi_m e^{j\theta_e} \end{bmatrix}$$

Now the motor equations (3.1) and (3.5) could be transformed into a rotating reference frame using the Park transformation via a rotation in the complex plane $e^{j\theta_e}$. The voltage in this plane denoted $u_s^s = u_s e^{j\theta_e}$, the current $i_s^s = i_s e^{j\theta_e}$ and the magnetic field $\psi_s^s = \psi_s e^{j\theta_e}$. This yielded the motor equations in the dq-subspace

$$u_s = R i_s + \frac{d\psi_s e^{j\theta_e}}{dt} e^{-j\theta_e} = R i_s + \frac{d\psi_s}{dt} + j\omega_e \psi_s. \tag{3.6}$$

Finally, by expanding and assigning the real part of the equation above to the d-axis, and the imaginary component to the q-axis, the state-space for the motor could be found in dq-variables.

$$\psi_s = \begin{bmatrix} L_1 + L_3 & L_2 + L_3 \\ L_2 + L_3 & L_1 + L_3 \end{bmatrix} \begin{bmatrix} i_s^1 \\ i_s^2 \end{bmatrix} + \begin{bmatrix} \psi_m \\ \psi_m \end{bmatrix} = \begin{bmatrix} L_1 + L_3 & L_2 + L_3 \\ L_2 + L_3 & L_1 + L_3 \end{bmatrix} \begin{bmatrix} i_d^1 + j i_q^1 \\ i_d^2 + j i_q^2 \end{bmatrix} + \begin{bmatrix} \psi_m \\ \psi_m \end{bmatrix}, \tag{3.7}$$

$$\begin{bmatrix} u_d^1 + j u_q^1 \\ u_d^2 + j u_q^2 \end{bmatrix} = R \begin{bmatrix} i_d^1 + j i_q^1 \\ i_d^2 + j i_q^2 \end{bmatrix} + \frac{d\psi_s}{dt} + j\omega_e \psi_s. \tag{3.8}$$

Combining these equations and separating the real and imaginary components thus yielded

$$u_d^1 = Ri_d^1 + (L_1 + L_3)\frac{di_d^1}{dt} + (L_2 + L_3)\frac{di_d^2}{dt} - \omega_e((L_1 + L_3)i_q^1 + (L_2 + L_3)i_q^2), \tag{3.9}$$

$$u_q^1 = Ri_q^1 + (L_1 + L_3)\frac{di_q^1}{dt} + (L_2 + L_3)\frac{di_q^2}{dt} + \omega_e((L_1 + L_3)i_d^1 + (L_2 + L_3)i_d^2) + \omega_e\psi_m, \tag{3.10}$$

$$u_d^2 = Ri_d^2 + (L_2 + L_3)\frac{di_d^1}{dt} + (L_1 + L_3)\frac{di_d^2}{dt} - \omega_e((L_2 + L_3)i_q^1 + (L_1 + L_3)i_q^2), \tag{3.11}$$

$$u_q^2 = Ri_q^2 + (L_2 + L_3)\frac{di_q^1}{dt} + (L_1 + L_3)\frac{di_q^2}{dt} + \omega_e((L_2 + L_3)i_d^1 + (L_1 + L_3)i_d^2) + \omega_e\psi_m. \tag{3.12}$$

Getting the above motor equations on a state-space form required expressing $\frac{di_d^2}{dt}$ in terms of $\frac{di_d^1}{dt}$ and then substituting it into the second subsystem d-current equation. This gave an expression of $\frac{di_d^1}{dt}$ in terms of the currents, voltages and the electrical angular velocity without any dependency on other current derivatives. A similar derivation for the q-currents was done. With some simplifications this gave the state-space equations below.

$$\frac{di_d^1}{dt} = \frac{1}{L_A^2 - L_B^2}\Big(L_A u_d^1 - L_B u_d^2 + L_B R i_d^2 - L_A R i_d^1\Big) + \omega_e i_q^1, \tag{3.13}$$

$$\frac{di_q^1}{dt} = \frac{1}{L_A^2 - L_B^2}\Big(L_A u_q^1 - L_B u_q^2 - L_A R i_q^1 + L_B R i_q^2\Big) - \omega_e i_d^1 - \frac{1}{L_A + L_B}\omega_e\psi_m, \tag{3.14}$$

$$\frac{di_d^2}{dt} = \frac{1}{L_A^2 - L_B^2}\Big(L_A u_d^2 - L_B u_d^1 + L_B R i_d^1 - L_A R i_d^2\Big) + \omega_e i_q^2, \tag{3.15}$$

$$\frac{di_q^2}{dt} = \frac{1}{L_A^2 - L_B^2}\Big(L_A u_q^2 - L_B u_q^1 - L_A R i_q^2 + L_B R i_q^1\Big) - \omega_e i_d^2 - \frac{1}{L_A + L_B}\omega_e\psi_m. \tag{3.16}$$

With $L_A = (L_1 + L_3)$ and $L_B = (L_2 + L_3)$

The produced torque was found by studying the power flow in the motor,

$$P_{in} = u_{abcdef}^T i_{abcdef} = \frac{3}{2}u_{dq}^T i_{dq}. \tag{3.17}$$

As mentioned in section 2.2.2, the power can be simplified to three terms: the copper losses (3.18), the production of the magnetic field (3.19), and the mechanical output power (3.20).

$$P_{copper\_loss} = \frac{3}{2}R((i_d^1)^2 + (i_d^2)^2 + (i_q^1)^2 + (i_q^2)^2), \tag{3.18}$$

$$\begin{aligned} P_{magnetic\_field\_loss} =& \frac{3}{2}L_A(\frac{di_d^1}{dt}i_d^1 + \frac{di_d^2}{dt}i_d^2 + \frac{di_q^1}{dt}i_q^1 + \frac{di_q^2}{dt}i_q^2) + \\ &+ \frac{3}{2}L_B(\frac{di_d^1}{dt}i_d^2 + \frac{di_d^2}{dt}i_d^1 + \frac{di_q^1}{dt}i_q^2 + \frac{di_q^2}{dt}i_q^1), \end{aligned} \tag{3.19}$$

$$P_{out} = \frac{3}{2}\omega_e \psi_m (i_q^1 + i_q^2), \tag{3.20}$$

$$P_{in} = P_{out} + P_{magnetic\_field\_loss} + P_{copper\_loss}. \tag{3.21}$$

By dividing the mechanical output power with the angular velocity of the shaft, the produced motor torque T_e was found,

$$T_e = \frac{3}{2} n_p \psi_m (i_q^1 + i_q^2), \tag{3.22}$$

where n_p was the number of pole pairs in the rotor.

3.1.1 Simplified Model for Control Schemes

Assuming perfect isolation between the different phases, L_2 and L_3 could be set to zero. Inserting this into (3.13)-(3.16) gave a simplified model of the system, which could easily be integrated into various control schemes. The simplified state-space model could then be written as

$$\frac{di_d^1}{dt} = \frac{1}{L_1}u_d^1 - \frac{1}{L_1}Ri_d^1 + \omega_e i_q^1, \tag{3.23}$$

$$\frac{di_q^1}{dt} = \frac{1}{L_1}u_q^1 - \frac{1}{L_1}Ri_q^1 - \omega_e i_d^1 - \frac{1}{L_1}\omega_e \psi_m, \tag{3.24}$$

$$\frac{di_d^2}{dt} = \frac{1}{L_1}u_d^2 - \frac{1}{L_1}Ri_d^2 + \omega_e i_q^2, \tag{3.25}$$

$$\frac{di_q^2}{dt} = \frac{1}{L_1}u_q^2 - \frac{1}{L_1}Ri_q^2 - \omega_e i_d^2 - \frac{1}{L_1}\omega_e \psi_m. \tag{3.26}$$

3.2 DC Motor

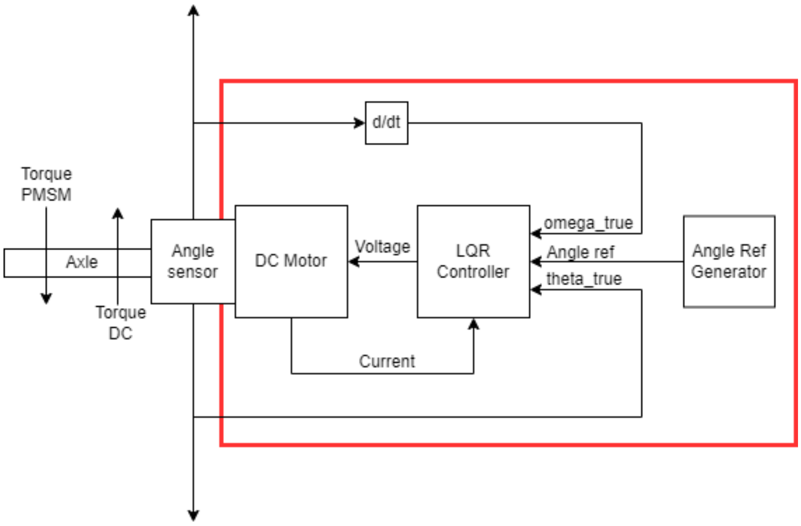


Figure 3.5: Dynamometer in the system.

Due to the reasons discussed in Section 2.6, the PMSM was connected to a dynamometer. The proposed solution was to couple the rotor shaft of the PMSM to a DC motor with an unlimited current and voltage supply. This DC motor used LQR to control the angle of the shaft according to a given reference trajectory. With this setup the produced torque in the PMSM could be decoupled from the motor shaft angular velocity and angle. From the perspective of the DCmotor, this torque was a disturbance which changed its direction and magnitude depending on the current state of the axle.

To ensure that the DC motor control system did not limit the capabilities of the PMSM system torque controller, the DC motor was modelled without any realistic power supply, i.e., it got whatever voltage it required. The motor was modelled with the axle dynamics and viewed the torque produced in the PMSM as a disturbance, this can be seen in Figure 3.6.

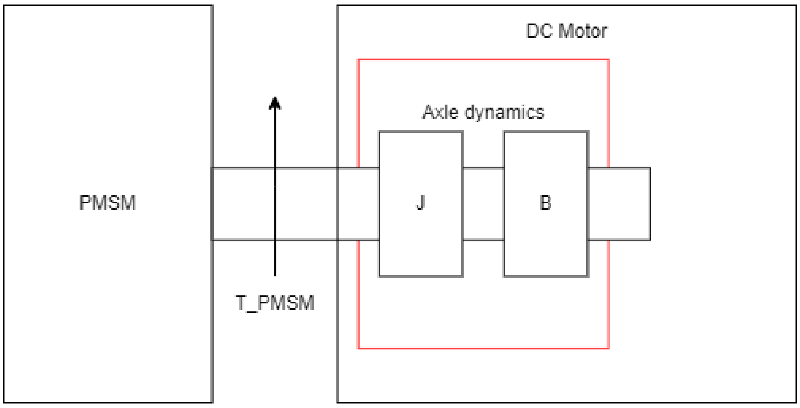


Figure 3.6: An overview of how the losses are modelled in the system.

The space vector which defined the motor and the axle was

$$x = \begin{bmatrix} \theta \\ \omega \\ i_{DC} \\ \int \theta \end{bmatrix}. \tag{3.27}$$

The axle-related equations were summarised in (3.28) - (3.30). Derivative relationships were given for the angle θ, angular velocity ω, and the integral state $\int \theta$ and torque equilibrium defined the angular acceleration $\dot{\omega}$.

$$\dot{\theta} = \omega, \tag{3.28}$$

$$\int \dot{\theta} = \theta, \tag{3.29}$$

$$\dot{\omega} = \frac{T_{DC}}{J} - \frac{T_{PMSM}}{J} - \frac{B\omega}{J} = \frac{K_m i_{DC}}{J} - \frac{T_{PMSM}}{J} - \frac{B\omega}{J}, \tag{3.30}$$

where i_{DC} was the current to the DC motor, T_{DC} was the produced torque by the DC motor, T_{PMSM} was the produced torque by the PMSM motor, K_m was the motor torque constant, J was the moment of inertia of the axle, B was the motor viscous friction constant, R was the electrical resistance in the DC motor and L was the electric inductance. Finally, the current in the motor obeyed the following differential equation

$$\dot{i}_{DC} = -\frac{i_{DC} R}{L} - \frac{K_e \omega}{L}, \tag{3.31}$$

where K_e was the electromotive force constant of the motor. The model was linear and can be transformed into state-space form:

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{B}{J} & \frac{K_m}{J} & 0 \\ 0 & \frac{K_e}{L} & \frac{R}{L} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \omega \\ i_{DC} \\ \int \theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \\ 0 \end{bmatrix} V_{DC} - \begin{bmatrix} 0 \\ \frac{T_{PMSM}}{J} \\ 0 \\ 0 \end{bmatrix}. \tag{3.32}$$

As the system was linear, it was also possible to control the DC motor using LQR. The controller was tuned with an emphasis on minimising deviations of the angle and the integral state. While disregarding the angular velocity and current usage as the power supplied was assumed to be infinite. This was done to ensure good angle tracking performance.

3.3 Modelling of SVPWM and Inverters

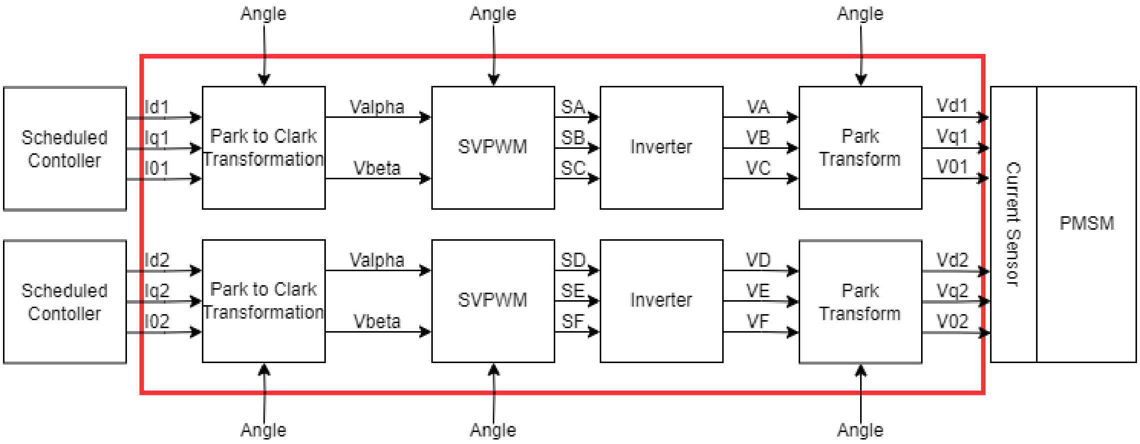


Figure 3.7: SVPWM and inverter in the system.

The following section will explain the implementation of SVPWM and inverters. The implementation can be divided into 6 parts.

1. Calculation of which sector the V_{ref} lies in (Sector calculation)
2. Calculation of the duty times for the two vectors that enclose the sector such that the V_{ref} can be recreated (Calculation of sector duty times)
3. Normalisation of duty times such that the sum of $T1 + T2$ does not exceed the PWM period (Normalisation of duty times)
4. Converts the sector duty times to phase duty times (Conversion from sector to phase duty times)
5. Calculate binary switches based on the duty times and the PWM period (Calculation of binary switches)
6. Calculate the phase voltages depending on the switch configuration (Calculation from binary switches to voltages)

3.3.1 Sector Calculation

The SVPWM was created firstly by dividing the circle into 6 different sectors. The sectors were defined as every $\frac{\pi}{3}$. Such that the first sector was defined as $0 \leq \alpha < \frac{\pi}{3}$ and the second was defined as $\frac{\pi}{3} \leq \alpha < \frac{2\pi}{3}$, where each dividing line was a possible voltage vector from a inverter switch combination. The sectors can be seen in Figure 3.8.

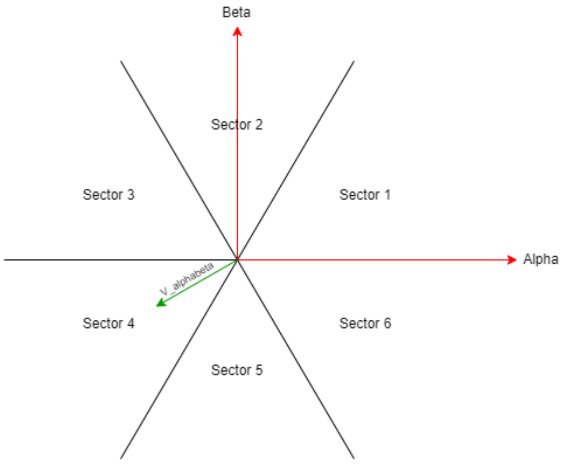


Figure 3.8: SVPWM sector partition.

Finding which sector the voltage reference lied in could be achieved by first checking for the quadrant, this could easily be read from V_α and V_β

$$B_\alpha : V_\alpha \geq 0, \tag{3.33}$$

$$B_\beta : V_\beta \geq 0. \tag{3.34}$$

Quadrant	B_α	B_β
1	1	1
2	0	1
3	0	0
4	1	0

Table 3.1: Binary combinations to determine Quadrant of V_{ref}.

Then, to distinguish which sector in the quadrant the reference vector lay, another computation was necessary. In the first quadrant, it was $\pi/3$ between the α-axis and the dividing line of the two sectors. Projecting onto the α and β axes, $cos(\frac{\pi}{3}) = \frac{1}{2}$ and $sin(\frac{\pi}{3}) = \frac{\sqrt{3}}{2}$. With this information, the sector could be found:

$$V_{ref} = \frac{V_\alpha}{cos(\frac{\pi}{3})} = 2V_\alpha, \tag{3.35}$$

$$V_{ref} = \frac{V_\beta}{sin(\frac{\pi}{3})} = \frac{2V_\beta}{\sqrt{3}}, \tag{3.36}$$

$$2V_\alpha = \frac{2V_\beta}{\sqrt{3}}, \tag{3.37}$$

$$V_\alpha = \frac{V_\beta}{\sqrt{3}}. \tag{3.38}$$

This meant that if $V_\alpha > \frac{V_\beta}{\sqrt{3}}$ the V_{ref} lied in sector 1 and otherwise in sector 2. To convert this to all the quadrants, it was possible to compare the lengths of both

$$B_{\alpha\beta} : |V_\alpha| > |\frac{V_\beta}{\sqrt{3}}|. \tag{3.39}$$

Now, it was possible to calculate all six sectors based on the 3 binary values. The combination matrix and its resulting output have been summarised in Table 3.2. The implementation into Simulink is shown in Figure 3.9.

Sector	Quadrant	B_α	B_β	$B_{\alpha\beta}$
2	1	1	1	0
2	2	0	1	0
5	3	0	0	0
5	4	1	0	0
1	1	1	1	1
3	2	0	1	1
4	3	0	0	1
6	4	1	0	1

Table 3.2: Binary combination to determine the sector of V_{ref}.

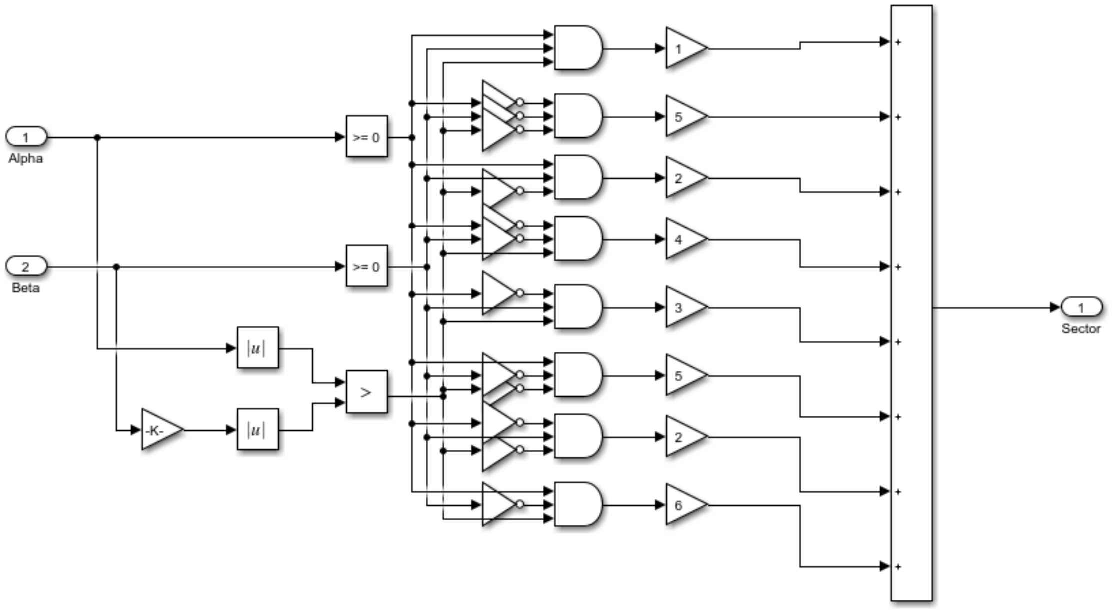


Figure 3.9: Simulink implementation that determines the sector of V_{ref}.

3.3.2 Calculation of Sector Duty Times

The switching times, also known as duty time, was the duration that each switch should be on to recreate the V_{ref} vector. All sector times were calculated in the same way, but to make it easier to follow, sector 4 has arbitrarily been chosen, in Figure 3.10 a decomposition of the basis vectors in this sector has been shown.

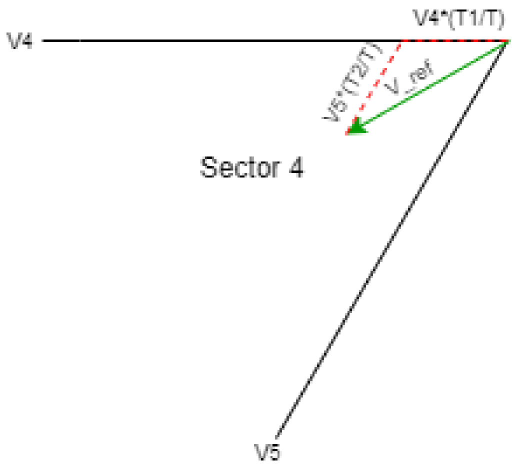


Figure 3.10: SVPWM time calculation.

The sub vector $V_{ref_{sub5}}$ pointed in the same direction as the voltage vector V5 and the angle between V4 and $V_{ref_{sub5}}$ was $\frac{\pi}{3}$, which with some trigonometry resulted in the following equations for $V_{ref_{sub4}}$

$$V_{ref_{sub4}} = |V_{ref}|cos(\theta_S) - V_{rest} = |V_{ref}|cos(\theta_S) - \frac{sin(\theta_S)|V_{ref}|}{tan(\frac{\pi}{3})} = |V_{ref}|\left(cos(\theta_S) - \frac{sin(\theta_S)}{\sqrt{3}}\right), \tag{3.40}$$

$$\text{where } \theta_S = \theta_r - \frac{(n-1)\pi}{3}, \tag{3.41}$$

where θ_S was the angle inside of a sector, θ_r was the mechanical angle, and n was the sector number. The calculation for $V_{ref_{sub5}}$ was done in a similar way,

$$V_{ref_{sub5}} = \frac{V_{opposite}}{sin(\frac{\pi}{3})} = |V_{ref}|\frac{sin(\theta_S)}{sin(\frac{\pi}{3})}. \tag{3.42}$$

Then, the duty times could be calculated as

$$V_{max} = \frac{2}{3}V_{dc}, \tag{3.43}$$

$$\frac{T1}{T}V_{max} = V_{ref_{sub4}}, \tag{3.44}$$

which yielded

$$T1 = |V_{ref}|\left(cos(\theta_S) - \frac{sin(\theta_S)}{\sqrt{3}}\right)\frac{T}{\frac{2}{3}V_{dc}}. \tag{3.45}$$

Simplifying the expression gave

$$T1 = |V_{ref}|\left(\frac{2sin(\frac{\pi}{3})cos(\theta_S) - sin(\theta_S)}{2sin(\frac{\pi}{3})}\right)\frac{T}{\frac{2}{3}V_{dc}}. \tag{3.46}$$

Utilising sum identities for the trigonometric functions simplified the expression further

$$T1 = |V_{ref}|\left(\frac{2(sin(\frac{\pi}{3} - \theta_S) - sin(-\theta_S)cos(\frac{\pi}{3})) - sin(\theta_S)}{2sin(\frac{\pi}{3})}\right)\frac{T}{\frac{2}{3}V_{dc}}, \tag{3.47}$$

$$T1 = |V_{ref}|\left(\frac{2(sin(\frac{\pi}{3} - \theta_S) + \frac{1}{2}sin(\theta_S)) - sin(\theta_S)}{2sin(\frac{\pi}{3})}\right)\frac{T}{\frac{2}{3}V_{dc}}, \tag{3.48}$$

$$T1 = |V_{ref}|\left(\frac{sin(\frac{\pi}{3} - \theta_S)}{sin(\frac{\pi}{3})}\right)\frac{T}{\frac{2}{3}V_{dc}}. \tag{3.49}$$

Then, T2 could be calculated by

$$\frac{T2}{T}V_{max} = V_{ref_{sub5}}, \tag{3.50}$$

$$T2 = |V_{ref}|\frac{sin(\theta_S)}{sin(\frac{\pi}{3})}\frac{T}{\frac{2}{3}V_{dc}}, \tag{3.51}$$

and lastly, the off-time was the remaining time of the period T

$$T0 = T - T1 - T2. \tag{3.52}$$

The implementation of the T1 and T2 calculations in Simulink can be seen in Figure 3.11

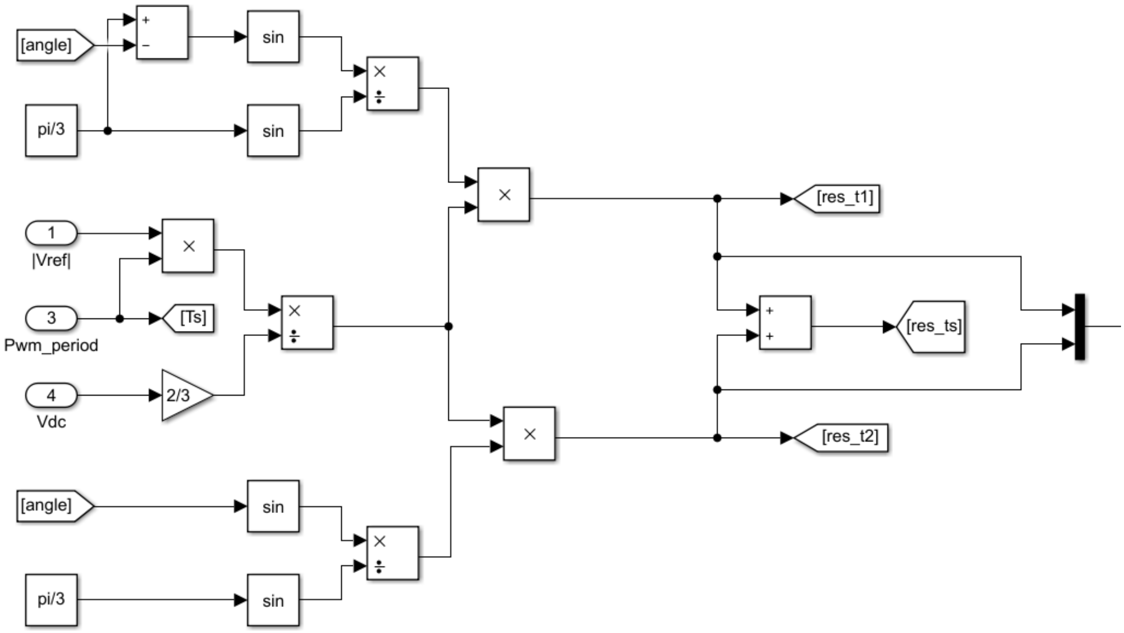


Figure 3.11: Simulink implementation of the time calculation.

3.3.3 Normalisation of Duty Times

To make sure that $T1+T2 \leq T$, a normalisation was used. This made sure that the vectors could not be longer than what was physically possible. The normalisation worked as follows:

$$T1_{norm} = \begin{cases} T1, & \text{if } T1+T2 \leq T \\ \frac{T}{T1+T2}T1, & \text{if } T1+T2 > T \end{cases}, \tag{3.53}$$

$$T2_{norm} = \begin{cases} T2, & \text{if } T1+T2 \leq T \\ \frac{T}{T1+T2}T2, & \text{if } T1+T2 > T \end{cases}. \tag{3.54}$$

The full implementation of the duty time calculations into Simulink can be seen in Figure 3.12.

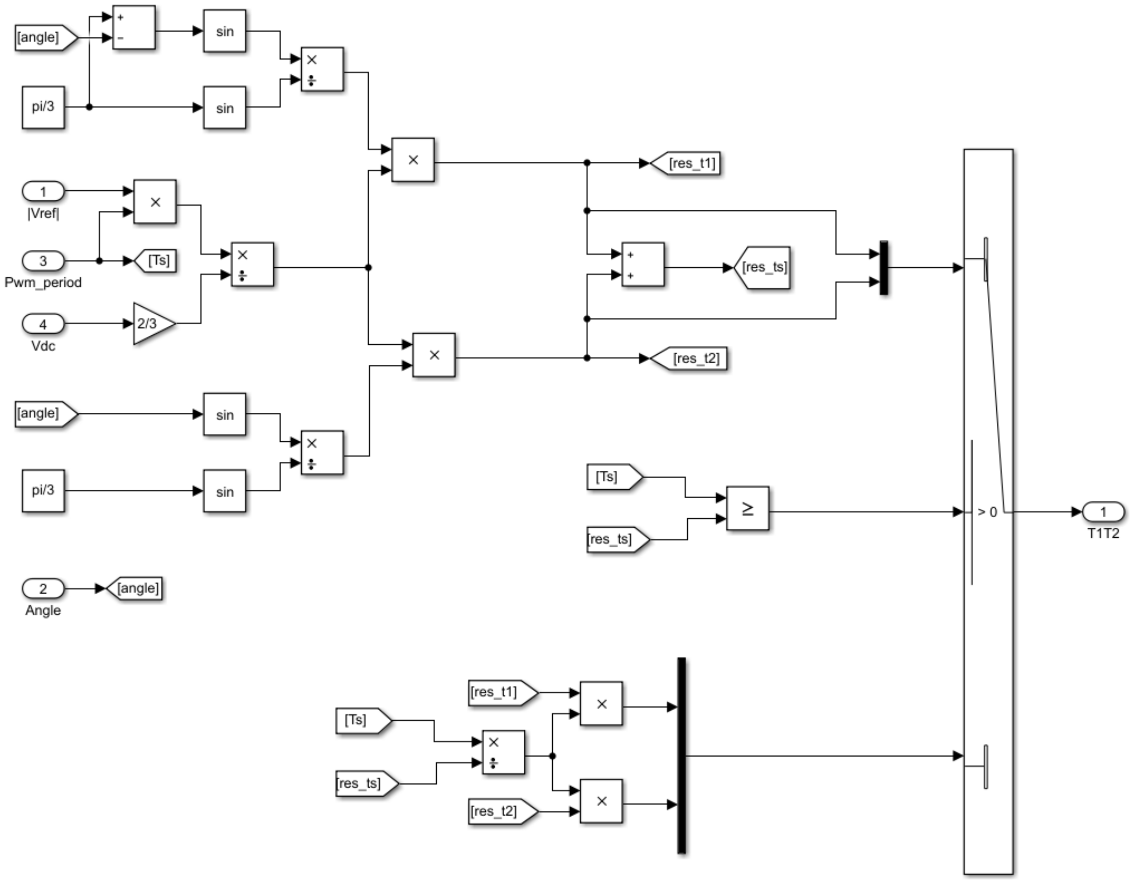


Figure 3.12: SVPWM 2-level time calculation.

3.3.4 Conversion from Sector to Phase Duty Times

The duty times were then converted to the three-phase system depending on the sector of V_{ref}. This was implemented with a table that determined the switching order, as can be seen in table 3.3. For example, in sector one, to recreate V1, the switch configuration was $[1, 0, 0]$, and for V2, it was $[1, 1, 0]$. Therefore, TA had to be on for T1 + T2 and half of the dead time, while TB should have only be on for T2 and half of the dead time. The implementation into Simulink can be seen in Figure 3.13.

Sector	TA	TB	TC
1	$T1 + T2 + \frac{T0}{2}$	$T2 + \frac{T0}{2}$	$\frac{T0}{2}$
2	$T1 + \frac{T0}{2}$	$T1 + T2 + \frac{T0}{2}$	$\frac{T0}{2}$
3	$\frac{T0}{2}$	$T1 + T2 + \frac{T0}{2}$	$T2 + \frac{T0}{2}$
4	$\frac{T0}{2}$	$T1 + \frac{T0}{2}$	$T1 + T2 + \frac{T0}{2}$
5	$T2 + \frac{T0}{2}$	$\frac{T0}{2}$	$T1 + T2 + \frac{T0}{2}$
6	$T1 + T2 + \frac{T0}{2}$	$\frac{T0}{2}$	$T1 + \frac{T0}{2}$

Table 3.3: Table of the phase duty times dependent on sector.

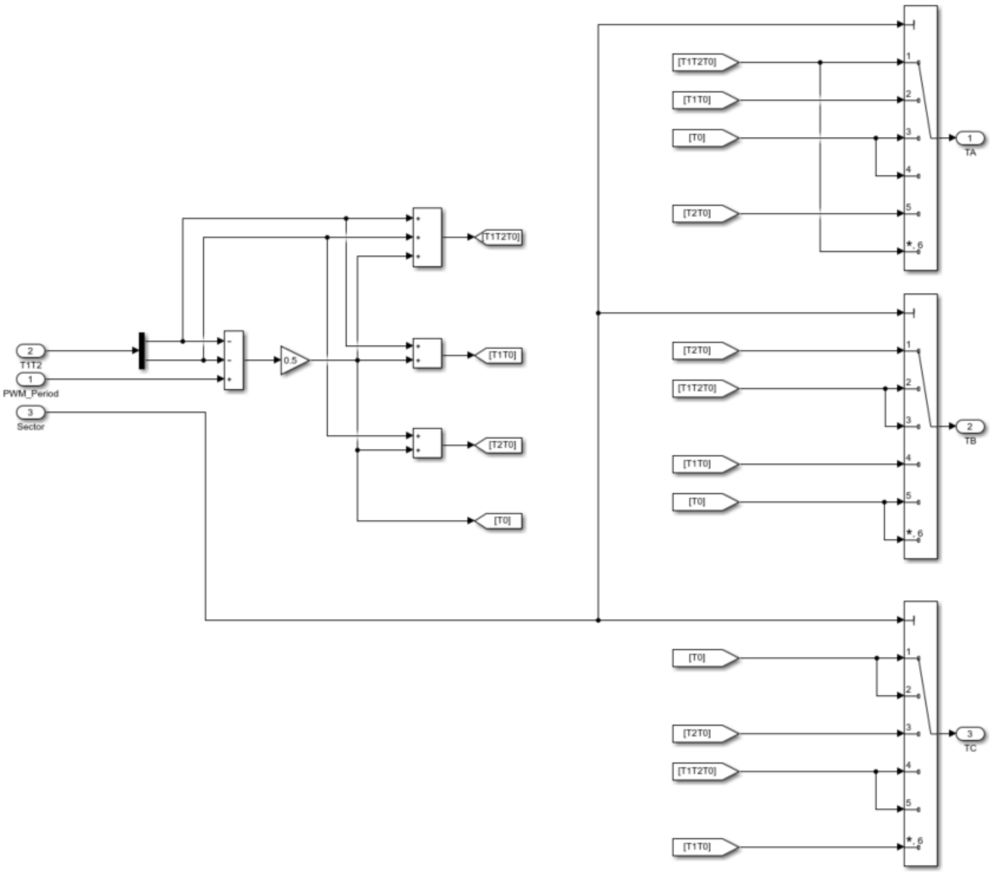


Figure 3.13: Simulink implementation of the phase duty times.

3.3.5 Calculation of Binary Switches

The next step was to convert the times TA, TB and TC to a boolean signal that was on for the duration of the given time. This conversion was implemented by choosing a triangle waveform for the boolean. As mentioned in Section 2.5.1, there are four types of triangle waveforms, left-aligned, right-aligned, centre-aligned and right-left-aligned. The project chose to use the centre-aligned triangle waveform because it did not periodically switch between the two off states. Which can be seen in Figure 3.14.

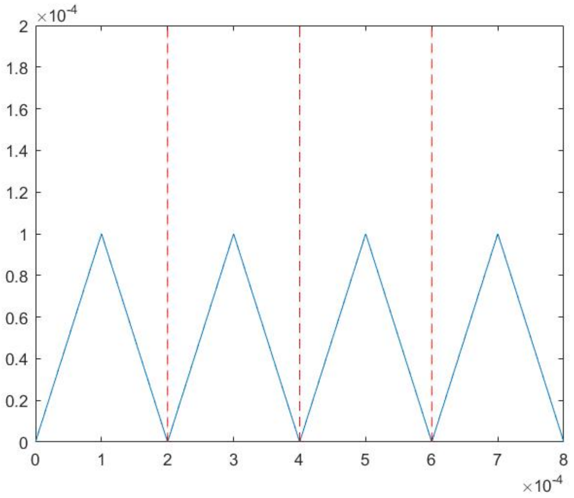


Figure 3.14: Centre-aligned triangle waveform.

As mentioned before, it should output a one during the given duty time and zero otherwise. This was solved by knowing that the waveform had the equation $y = x$ for the first half of the period and $y = T - x$ for the second half. However, the waveform could only reach half of the PWM period as the slope was equal to 1 or -1, which meant that the duty times and period had to be divided by two. Then, because $x = y$ it was possible to compare y and x. There are multiple ways to do this, but in this project, the off time was calculated and then compared to the waveform such that if the y of the waveform was greater than the off time, then it was on, and otherwise, it was off.

$$S_A : y_{waveform} \geq (T_{half} - TA_{half}), \tag{3.55}$$

$$S_B : y_{waveform} \geq (T_{half} - TB_{half}), \tag{3.56}$$

$$S_C : y_{waveform} \geq (T_{half} - TC_{half}). \tag{3.57}$$

When implementing (3.55) - (3.57) it looked similar to Figure 3.15 where the different lines were the ABC binaries shifted in y direction such that it was easier to see.

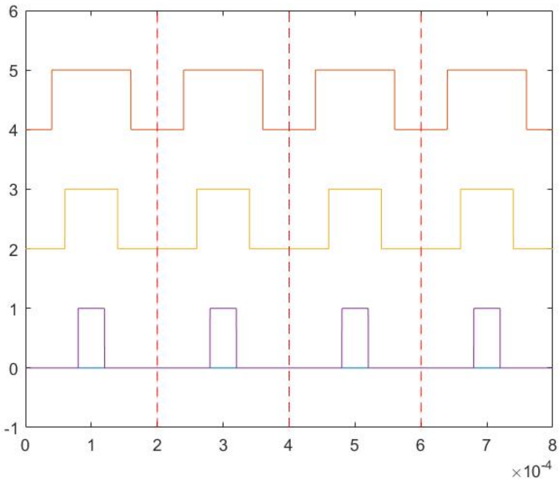


Figure 3.15: Example of PWM output given centre-aligned triangle waveform.

3.3.6 Calculation from Binary switches to Voltages (Inverter)

These signals are then sent to the inverter. The 8 different combinations generated different voltage outputs on the phases. All the combinations and the corresponding phase voltages can be seen in Table 3.4.

State	S_A	S_B	S_C	Voltage in phase A	Voltage in phase B	Voltage in phase C
V_0	0	0	0	0	0	0
V_1	1	0	0	$2\frac{V_{DC}}{3}$	$-\frac{V_{DC}}{3}$	$-\frac{V_{DC}}{3}$
V_2	1	1	0	$\frac{V_{DC}}{3}$	$\frac{V_{DC}}{3}$	$-2\frac{V_{DC}}{3}$
V_3	0	1	0	$-\frac{V_{DC}}{3}$	$2\frac{V_{DC}}{3}$	$-\frac{V_{DC}}{3}$
V_4	0	1	1	$-2\frac{V_{DC}}{3}$	$\frac{V_{DC}}{3}$	$\frac{V_{DC}}{3}$
V_5	0	0	1	$-\frac{V_{DC}}{3}$	$-\frac{V_{DC}}{3}$	$2\frac{V_{DC}}{3}$
V_6	1	0	1	$\frac{V_{DC}}{3}$	$-2\frac{V_{DC}}{3}$	$\frac{V_{DC}}{3}$
V_7	1	1	1	0	0	0

Table 3.4: Table of the output phase voltages given the binary combinations of the PWM signals.

The phase voltages could also be expressed as functions of the gating signals

$$V_A = \frac{V_{DC}}{3}(2S_A - S_B - S_C), \tag{3.58}$$

$$V_B = \frac{V_{DC}}{3}(-S_A + 2S_B - S_C), \tag{3.59}$$

$$V_C = \frac{V_{DC}}{3}(-S_A - S_B + 2S_C). \tag{3.60}$$

The full implementation of the system in Simulink from V_{ref} to V_A, V_B, V_C can be seen in Figure 3.16.

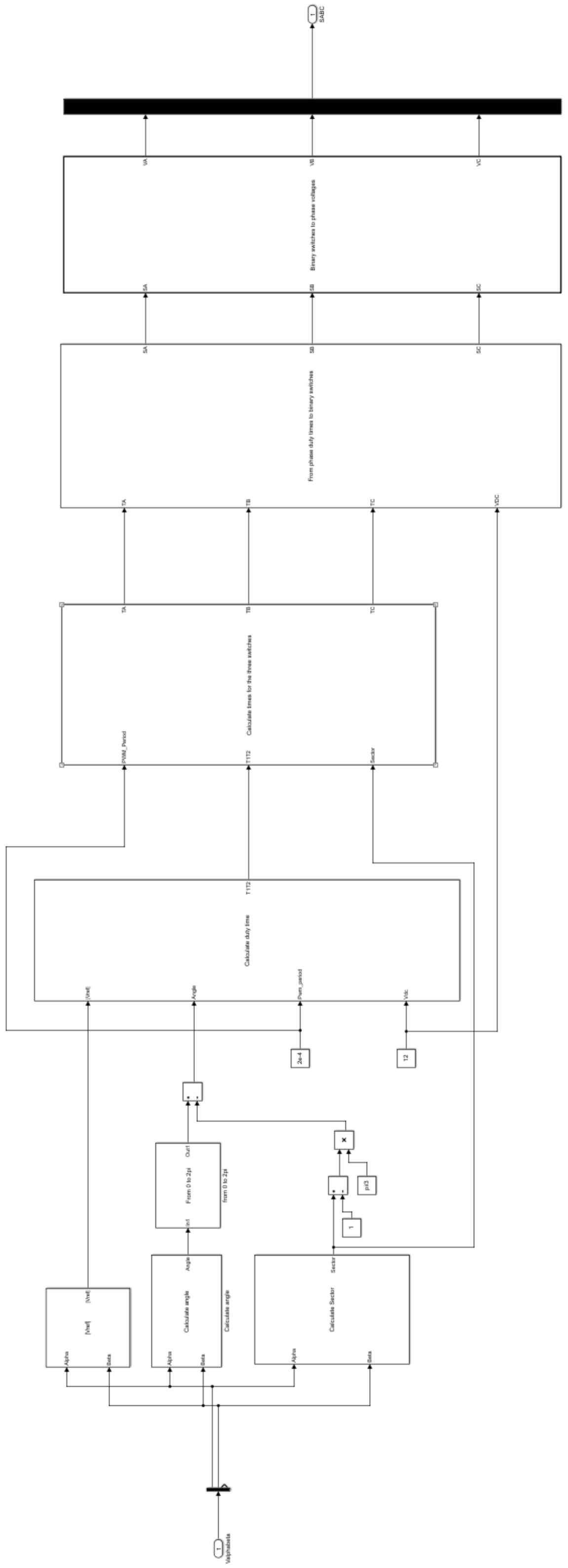


Figure 3.16: Implementation of the SVPWM and inverters.

3.4 Control of a Six-Phase PMSM

The objective of the controller is to generate a torque that resembles a given reference. Measurements of the phase currents and the rotor angle are given. Multiple controllers were developed, implemented, and tested. These were LQR, GSC, and two types of MPC, where each controller was tested with both the full state space model and the simplified model. This section will cover the derivation of all these controllers, a total of 7 different controllers.

3.4.1 LQR

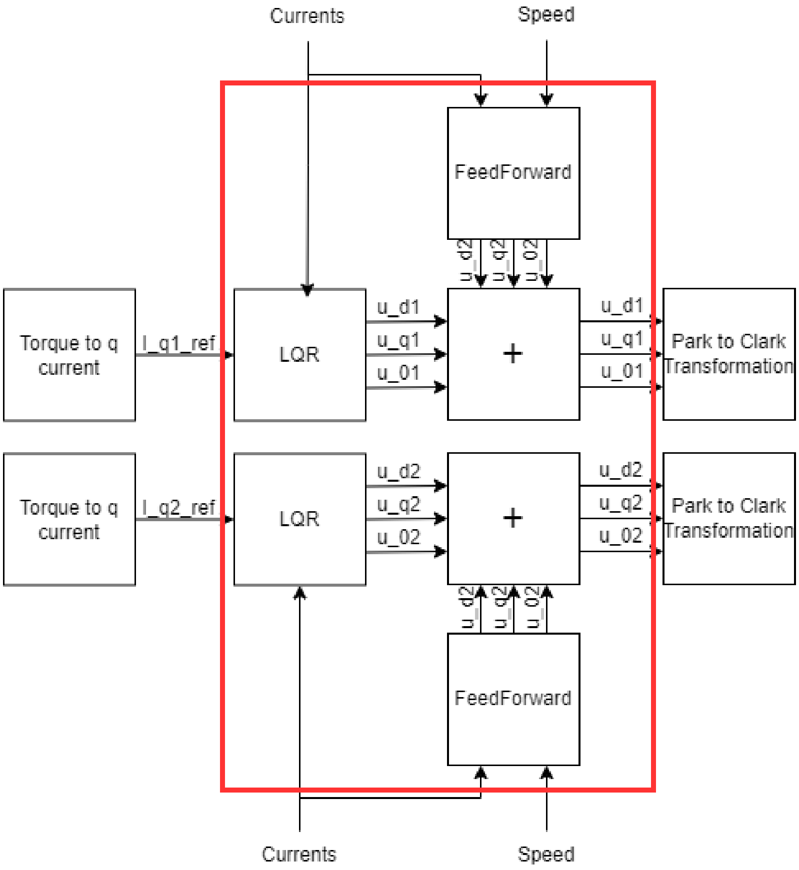


Figure 3.17: Linear quadratic control in the system.

Due to the non-linearities in the state-space models listed in Section 3.1, an LQR could not be developed. By separating the state-space equations into linear and non-linear terms a feedforward network could be used to compensate for the non-linear behaviour. The non-linear terms are the cross-couplings between the electrical velocity and the currents as well as for the back electromotive force. This left a linear state space, which was used to create an LQR feedback controller.

3.4.1.1 Simplified Model LQR

The simplified state-space, (3.23)-(3.26), consisted of a linear part and a non-linear part, which could be rewritten as

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix} i_d^1 \\ i_q^1 \\ i_d^2 \\ i_q^2 \end{bmatrix} &= \begin{bmatrix} -\frac{R}{L_1} & 0 & 0 & 0 \\ 0 & -\frac{R}{L_1} & 0 & 0 \\ 0 & 0 & -\frac{R}{L_1} & 0 \\ 0 & 0 & 0 & -\frac{R}{L_1} \end{bmatrix}\begin{bmatrix} i_d^1 \\ i_q^1 \\ i_d^2 \\ i_q^2 \end{bmatrix} + \begin{bmatrix} \frac{1}{L_1} & 0 & 0 & 0 \\ 0 & \frac{1}{L_1} & 0 & 0 \\ 0 & 0 & \frac{1}{L_1} & 0 \\ 0 & 0 & 0 & \frac{1}{L_1} \end{bmatrix}\begin{bmatrix} u_d^1 \\ u_q^1 \\ u_d^2 \\ u_q^2 \end{bmatrix} + \\ &+ \omega_e \begin{bmatrix} i_q^1 \\ -i_d^1 - \frac{1}{L_q}\psi_m \\ i_q^2 \\ -i_d^2 - \frac{1}{L_q}\psi_m \end{bmatrix} = A\begin{bmatrix} i_d^1 \\ i_q^1 \\ i_d^2 \\ i_q^2 \end{bmatrix} + B\begin{bmatrix} u_d^1 \\ u_q^1 \\ u_d^2 \\ u_q^2 \end{bmatrix} + N.\end{aligned} \tag{3.61}$$

The input could then be split into two terms, one compensating for the cross-couplings and back-emf, the other: the control input from the LQR,

$$\frac{d}{dt}\begin{bmatrix} i_d^1 \\ i_q^1 \\ i_d^2 \\ i_q^2 \end{bmatrix} = A\begin{bmatrix} i_d^1 \\ i_q^1 \\ i_d^2 \\ i_q^2 \end{bmatrix} + B\begin{bmatrix} u_{d1}^1 \\ u_{q1}^1 \\ u_{d1}^2 \\ u_{q1}^2 \end{bmatrix} + B\begin{bmatrix} u_{d2}^1 \\ u_{q2}^1 \\ u_{d2}^2 \\ u_{q2}^2 \end{bmatrix} + N \quad \text{, where } u_*^* = u_{*1}^* + u_{*2}^*. \tag{3.62}$$

The compensation in u_{*2}^* could be calculated by setting $B\begin{bmatrix} u_{d2}^1 & u_{q2}^1 & u_{d2}^2 & u_{q2}^2 \end{bmatrix}^T + N^T = 0$ which yield the expressions

$$\begin{aligned} u_{d2}^1 &= -L_1\omega_e i_q^1, \\ u_{q2}^1 &= L_1\omega_e i_d^1 + \omega_e\psi_m, \\ u_{d2}^2 &= -L_1\omega_e i_q^1, \\ u_{q2}^2 &= L_1\omega_e i_d^1 + \omega_e\psi_m. \end{aligned} \tag{3.63}$$

By making this compensation, the system could now be viewed as linear. Furthermore, integral states for all the currents were added into the process matrix $A_{control}$ and the control input matrix $B_{control}$

$$A_{control} = \begin{bmatrix} -\frac{R}{L_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{R}{L_1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{R}{L_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{R}{L_1} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad B_{control} = \begin{bmatrix} \frac{1}{L_1} & 0 & 0 & 0 \\ 0 & \frac{1}{L_1} & 0 & 0 \\ 0 & 0 & \frac{1}{L_1} & 0 \\ 0 & 0 & 0 & \frac{1}{L_1} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.64}$$

With weighting matrices Q and R, the LQR controller gain could be found for the system by solving the control algebraic Riccati equation (2.11). This controller will hence be referred to as S-LQR.

3.4.1.2 Full Model LQR

A similar LQR controller to the one above could be developed with the full state-space model stated in (3.13) - (3.16). The model was split into two parts: a linear part and a non-linear part. The control input was separated into a feedforward compensation term and a feedback term. The feedforward compensation terms u_{*2}^{*} were

$$\begin{aligned} u_{d2}^{1} &= -L_A \omega_e i_q^1 - L_B \omega_e i_q^2, \\ u_{q2}^{1} &= L_A \omega_e i_d^1 + L_B \omega_e i_d^2 + \omega_e \psi_m, \\ u_{d2}^{2} &= -L_A \omega_e i_q^2 - L_B \omega_e i_q^1, \\ u_{q2}^{2} &= L_A \omega_e i_d^2 + L_B \omega_e i_d^1 + \omega_e \psi_m. \end{aligned} \tag{3.65}$$

The linear dynamics with described by the matrices $A_{control}$ and $B_{control}$ where integral action was added to compensate for steady-state errors. The controller achieved by using the A and B matrices below will be referred to as C-LQR.

$$\begin{aligned} A_{control} &= \begin{bmatrix} -\frac{L_A R}{L_A^2 - L_B^2} & 0 & \frac{L_B R}{L_A^2 - L_B^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{L_A R}{L_A^2 - L_B^2} & 0 & \frac{L_B R}{L_A^2 - L_B^2} & 0 & 0 & 0 & 0 \\ \frac{L_B R}{L_A^2 - L_B^2} & 0 & -\frac{L_A R}{L_A^2 - L_B^2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{L_B R}{L_A^2 - L_B^2} & 0 & -\frac{L_A R}{L_A^2 - L_B^2} & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \\ B_{control} &= \begin{bmatrix} \frac{L_A}{L_A^2 - L_B^2} & 0 & -\frac{L_B}{L_A^2 - L_B^2} & 0 \\ 0 & \frac{L_A}{L_A^2 - L_B^2} & 0 & -\frac{L_B}{L_A^2 - L_B^2} \\ -\frac{L_B}{L_A^2 - L_B^2} & 0 & \frac{L_A}{L_A^2 - L_B^2} & 0 \\ 0 & -\frac{L_B}{L_A^2 - L_B^2} & 0 & \frac{L_A}{L_A^2 - L_B^2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{3.66}$$

3.4.2 GSC

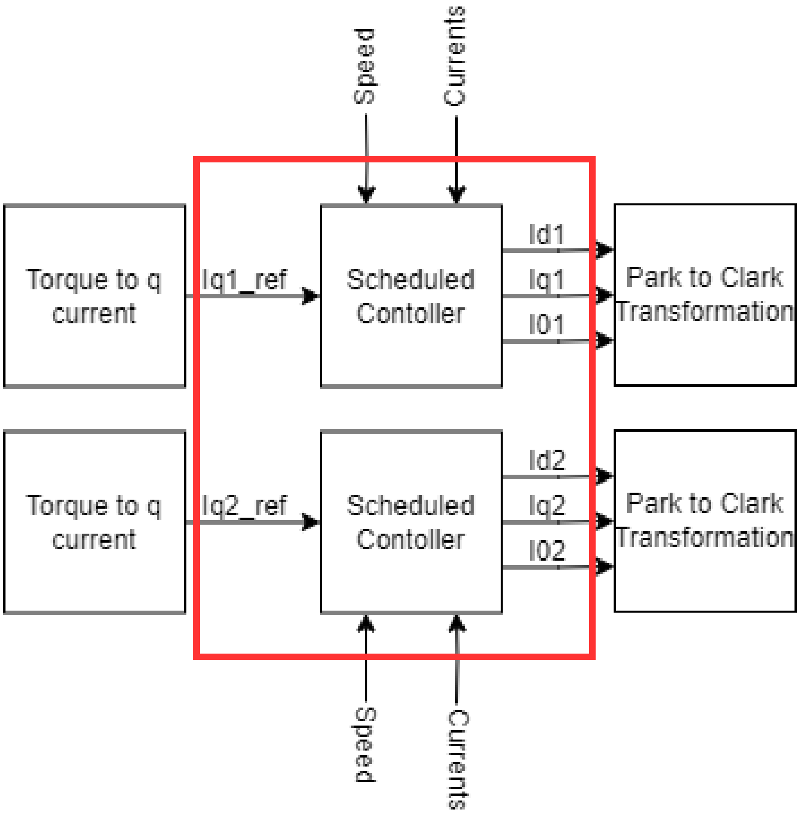


Figure 3.18: GSC in the system.

The GSC operated using a slightly different method than the feedforward compensation technique discussed in the previous section. Instead of compensating for the non-linear terms, the GSC took the angular velocity as a parameter and a state. The parameter decided the controller gain, and the state was coupled to the back-emf term in the state space.

3.4.2.1 Simplified Model GSC

The states defining the six-phase PMSM were i_d^1, i_q^2, i_d^2, i_q^2, ω_e. The state-space model shown in (3.23) - (3.26) and the torque equation in (3.22), show the relation between these variables and the input voltage. The magnetic field flux term contained a product between the angular velocity ω_e and a current for every equation, making the state-space non-linear. By assigning a parameter $p(t) = \omega_e$, the state-space model could be expressed in the form of an LPV, see (2.12), with the state vector

$x^T = \begin{bmatrix} i_d^1 & i_q^1 & i_d^2 & i_q^2 & \omega_e \end{bmatrix}$, and the motor inertia J, this model is obtained

$$\begin{gathered}
A(p) = \begin{bmatrix} -\frac{R_s}{L_1} & p & 0 & 0 & 0 \\ -p & -\frac{R_s}{L_1} & 0 & 0 & -\frac{1}{L_1}\psi_m \\ 0 & 0 & -\frac{R_s}{L_1} & p & 0 \\ 0 & 0 & -p & -\frac{R_s}{L_1} & -\frac{1}{L_1}\psi_m \\ 0 & \frac{3n_p^2\psi_m}{2J} & 0 & \frac{3n_p^2\psi_m}{2J} & 0 \end{bmatrix}, \\
B(p) = \begin{bmatrix} \frac{1}{L_1} & 0 & 0 & 0 \\ 0 & \frac{1}{L_1} & 0 & 0 \\ 0 & 0 & \frac{1}{L_1} & 0 \\ 0 & 0 & 0 & \frac{1}{L_1} \\ 0 & 0 & 0 & 0 \end{bmatrix}.
\end{gathered} \tag{3.67}$$

A measurement of the phase currents were available through the inverter, with the angle sensor these currents could be transformed to the *dq*-reference frame with a Park transformation. Thus the measurement matrix C is easily found,

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.68}$$

Given a DC source to power the motor, the angular velocity was naturally capped at some minimum and maximum value, $\omega \in \begin{bmatrix} \omega_{min} & \omega_{max} \end{bmatrix}$, with $\omega_{min} = -\omega_{max}$. The model was evaluated for different angular velocities within the parameter set, and the LQR gain for all these evaluated models was calculated. The gain was calculated for many values of p, which made it possible to see that the gain was quadratic in some terms and linear in others, as can be seen in Figure 3.19.

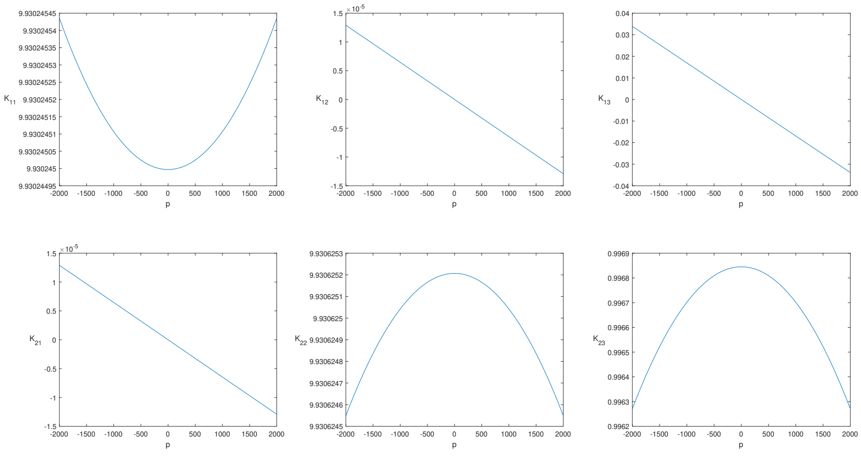


Figure 3.19: GSC gain matrix depending on the angular velocity.

As the controller gain included quadratic terms in p, an approximation was done with piece-wise linear functions evaluated at three values for p, $p = \omega_{min}$, $p = \omega_{max}$

and $p = 0$. The controller gains found at these values K_{min}, K_{max} and K_0 were then used to create the GSC of the simplified model. This controller will be referred to as S-GSC in this project. Thus if $p > 0$

$$K(p) = (K_{max} - K_0)\frac{p}{\omega_{max}} + K_0, \tag{3.69}$$

and if $p \leq 0$

$$K(p) = (K_{min} - K_0)\frac{p}{\omega_{min}} + K_0. \tag{3.70}$$

For the feedback loop, a desired torque was sent as a reference to the control loop by using (3.22), the reference could be transformed to two equal q-current references, (3.71), and the reference for d-currents was kept as a constant zero.

$$i^1_{q,ref} = i^2_{q,ref} = \frac{T_{e,ref}}{3n_p\psi_m}. \tag{3.71}$$

A control error could then be calculated between the measured and desired currents from which the S-GSC decides an appropriate dq voltage.

3.4.2.2 Full Model GSC

An identical approach to the previous section with the complete state-space model leads to the following $A(p)$ and $B(p)$ matrices

$$\begin{aligned}
A_{control} &= \begin{bmatrix} -\frac{L_AR}{L_A^2-L_B^2} & p & \frac{L_BR}{L_A^2-L_B^2} & 0 & 0 & \\ -p & -\frac{L_AR}{L_A^2-L_B^2} & 0 & \frac{L_BR}{L_A^2-L_B^2} & 0 & -\frac{\psi_m}{L_A+L_B} \\ \frac{L_BR}{L_A^2-L_B^2} & 0 & -\frac{L_AR}{L_A^2-L_B^2} & p & 0 & \\ 0 & \frac{L_BR}{L_A^2-L_B^2} & -p & -\frac{L_AR}{L_A^2-L_B^2} & -\frac{\psi_m}{L_A+L_B} & \\ 0 & \frac{3n_p^2\psi_m}{2J} & 0 & \frac{3n_p^2\psi_m}{2J} & 0 & \end{bmatrix}, \\
B_{control} &= \begin{bmatrix} \frac{L_A}{L_A^2-L_B^2} & 0 & -\frac{L_B}{L_A^2-L_B^2} & 0 \\ 0 & \frac{L_A}{L_A^2-L_B^2} & 0 & -\frac{L_B}{L_A^2-L_B^2} \\ -\frac{L_B}{L_A^2-L_B^2} & 0 & \frac{L_A}{L_A^2-L_B^2} & 0 \\ 0 & -\frac{L_B}{L_A^2-L_B^2} & 0 & \frac{L_A}{L_A^2-L_B^2} \end{bmatrix}.
\end{aligned} \tag{3.72}$$

The controller $K(p)$ was again found as piece-wise linear approximations between the points $p = -\omega_{min}$, ω_{max} and 0. This controller will be referred to as C-GSC.

3.4.3 Model Predictive Control

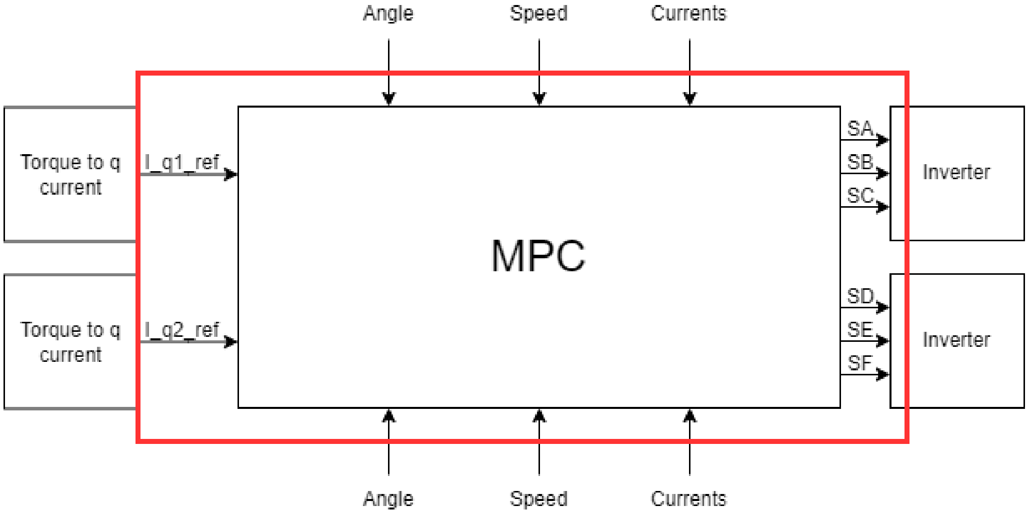


Figure 3.20: Model Predictive Control in the system.

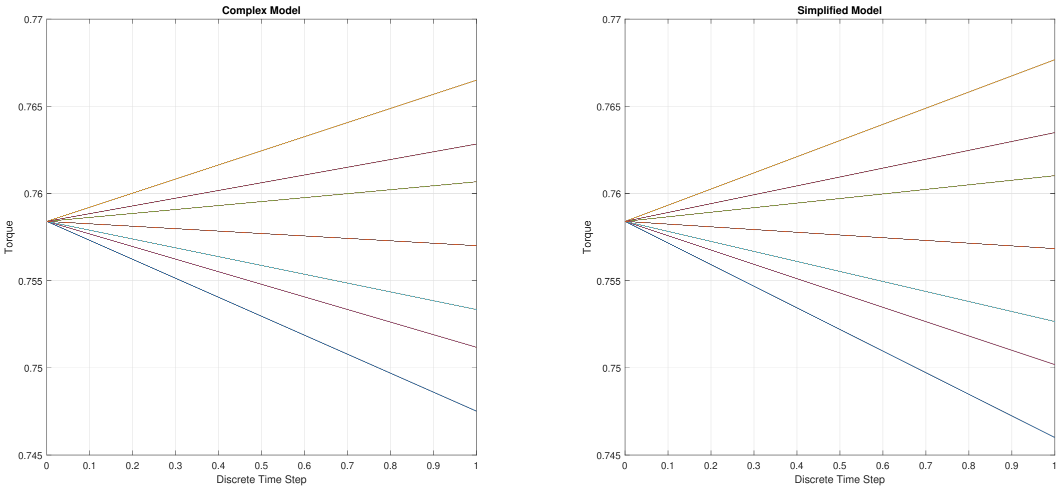
In reality the controller did not have the capability to decide any specific voltage for a given sample time. From (3.58)-(3.60), each three-phase subsystem only had 8 possible output voltages per sample. The control objective for the controller was then to minimise the next sample current from some current reference by controlling the switches in the inverter. Three types of MPC were developed:

1. Separate control of the simplified model (SS-MPC)

2. Additive control of the simplified model (AS-MPC)

3. Additive control of the complex model (AC-MPC)

This can be summarised by the selection of either separate or additive control as well as simple or complex models. However, the general principle was the same for all the MPC controllers. The 8 possible switch configurations are illustrated in Figure 3.21. Note there are only 7 different steps because switching state $[1, 1, 1]$ and $[0, 0, 0]$ generated zero voltage. It can be seen in (3.58)-(3.60).



(a) The complex models predicted the next step for each switch combination.

(b) The simplified models predicted the next step for each switch combination.

Figure 3.21: Generated torque given switch configurations for the two models.

The MPC calculated all the possible steps and then took the action that minimised the deviation from the reference current. The MPC took the following three steps:

1. Calculate all possible switch configuration voltages in a three-phase system
2. Calculate the next current either using the complex or simplified model
3. Take the action that minimises the error according to the additive or separate method

These steps were repeated every iteration. An example can be seen in Figure 3.22

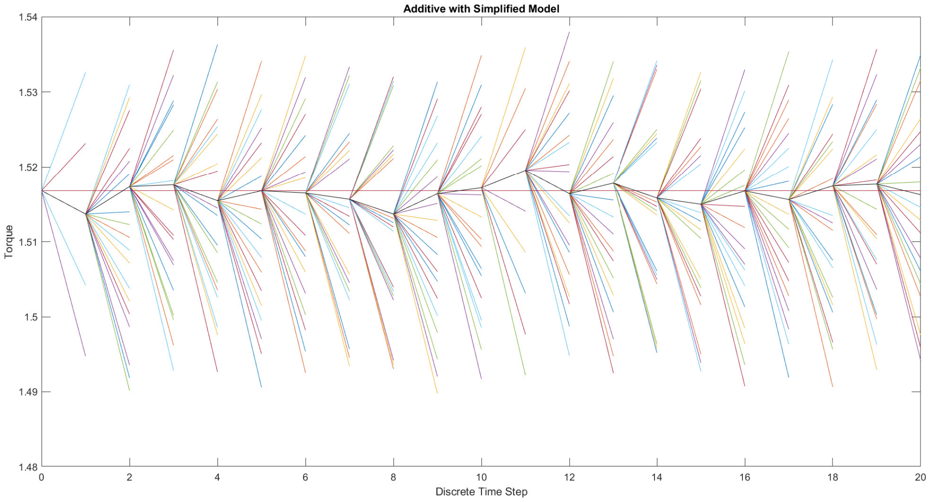


Figure 3.22: Example of the MPC working principle.

3.4.3.1 Voltage Calculations

The phase voltages had to be calculated in each iteration because they were angle-dependent. The calculation was done by first defining all the binary combinations of the switches as

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}. \tag{3.73}$$

With all combinations known, it was possible to calculate the phase voltages using (3.58)-(3.60). Then, the voltages could be transformed to the *dq*-plane using the Park transform.

Algorithm 1 Algorithm Voltage calculations

```
$S \leftarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$        ▷ All possible configurations
for i = 1:length(S) do
    Calculating the phase voltages:
    $V_{ABC} \leftarrow \begin{bmatrix} \frac{VDC}{3}(2S(1,i) - S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) + 2S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) - S(2,i) + 2S(3,i)) \end{bmatrix}$
    Transforming to dq voltages:
    $V_{dq} \leftarrow \begin{bmatrix} cos(\theta_{e1}) & cos(\theta_{e1} - \frac{2pi}{3}) & cos(\theta_{e1} + \frac{2pi}{3}) \\ -sin(\theta_{e1}) & -sin(\theta_{e1} - \frac{2pi}{3}) & -sin(\theta_{e1} + \frac{2pi}{3}) \end{bmatrix} V_{ABC}$
end for
```

3.4.3.2 Complex or Simplified Model

The complex model used (3.13)-(3.16). The equations were continuous, which meant that they had to be discretised to be able to use them in the MPC. The discretisation was done using the forward Euler method, $\frac{di_q^2}{dt}(t_k) \approx \frac{i_q^2(k+1)-i_q^2(k)}{T}$. Performing this on the four equations in the state-space yielded

$$\begin{aligned} i_d^1(k+1) = \frac{T}{L_A^2 - L_B^2}\Big(L_A u_d^1(k) - L_B u_d^2(k) + L_B R i_d^2(k) - L_A R i_d^1(k)\Big) \\ + T\omega_e(k) i_q^1(k) + i_d^1(k), \end{aligned} \tag{3.74}$$

$$\begin{aligned} i_q^1(k+1) = \frac{T}{L_A^2 - L_B^2}\Big(L_A u_q^1(k) - L_B u_q^2(k) - L_A R i_q^1(k) + L_B R i_q^2(k)\Big) \\ - T\omega_e(k) i_d^1(k) - \frac{T}{L_A + L_B}\omega_e(k)\psi_m + i_q^1(k), \end{aligned} \tag{3.75}$$

$$\begin{aligned} i_d^2(k+1) = \frac{T}{L_A^2 - L_B^2}\Big(L_A u_d^2(k) - L_B u_d^1(k) + L_B R i_d^1(k) - L_A R i_d^2(k)\Big) \\ + T\omega_e(k) i_q^2(k) + i_d^2(k), \end{aligned} \tag{3.76}$$

$$
\begin{aligned}
i_q^2(k+1) = \frac{T}{L_A^2 - L_B^2}\Big(L_A u_q^2(k) - L_B u_q^1(k) - L_A R i_q^2(k) + L_B R i_q^1(k)\Big) \\
-T\omega_e(k) i_d^2(k) - \frac{T}{A+B}\omega_e(k)\psi_m + i_q^2(k).
\end{aligned}
\tag{3.77}
$$

The simplified system used (3.23)-(3.26). These equations were also continuous, which meant that they had to be discretised. They were also discretised using the forward Euler method, which yielded

$$
i_d^1(k+1) = \frac{T}{L_1} u_d^1(k) - \frac{T}{L_1} R i_d^1(k) + T\omega_e(k) i_q^1(k) + i_d^1(k), \tag{3.78}
$$

$$
i_q^1(k+1) = \frac{T}{L_1} u_q^1(k) - \frac{T}{L_1} R i_q^1(k) - T\omega_e(k) i_d^1(k) - \frac{T}{L_1}\omega_e(k)\psi_m + i_q^1(k), \tag{3.79}
$$

$$
i_d^2(k+1) = \frac{T}{L_1} u_d^2(k) - \frac{T}{L_1} R i_d^2(k) + T\omega_e(k) i_q^2(k) + i_d^2(k), \tag{3.80}
$$

$$
i_q^2(k+1) = \frac{T}{L_1} u_q^2(k) - \frac{T}{L_1} R i_q^2(k) - T\omega_e(k) i_d^2(k) - \frac{T}{L_1}\omega_e(k)\psi_m + i_q^2(k). \tag{3.81}
$$

Both the simplified and the complex equations were then used in the MPC to predict the next step of the current given a switch configuration.

3.4.3.3 Separate or Additive Control

Separate control refers to that the two three phase systems were separated such that they are working without the knowledge of one another. This was beneficial from a redundancy standpoint because then the systems were totally separated from each other. Furthermore, it was less computationally heavy as all combinations of the systems didn't have to be calculated. The loss function of the separate system was defined as

$$
J_1 = Q_1(i_{d_1,ref} - i_{d_1}(i+1))^2 + Q_2(i_{q_1,ref} - i_{q_1}(i+1))^2, \tag{3.82}
$$

$$
J_2 = Q_1(i_{d_2,ref} - i_{d_2}(i+1))^2 + Q_2(i_{q_2,ref} - i_{q_2}(i+1))^2. \tag{3.83}
$$

The additive control used information from both the three-phase systems. With that information, it took action for both systems that resulted in the lowest total deviation from the reference. This method thus had a lower torque ripple. The additive system calculated the loss function as

$$
J = Q_0(i_{q,ref} - i_{q_1}(i+1) - i_{q_2}(i+1))^2 + \sum_{n=1}^{2} Q_n(i_{d_n,ref} - i_{d_n}(i+1))^2. \tag{3.84}
$$

3.4.3.4 Algorithms of the Three Different MPC

Algorithm 2 Algorithm for separate control of simplified model

```
$S \leftarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$    ▷ All possible configurations
for i = 1:length(S) do
    Calculating the phase voltages:
    $V_{ABC} \leftarrow \begin{bmatrix} \frac{VDC}{3}(2S(1,i) - S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) + 2S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) - S(2,i) + 2S(3,i)) \end{bmatrix}$
    Transforming to dq voltages:
    $V_{dq} \leftarrow \begin{bmatrix} cos(\theta_{e1}) & cos(\theta_{e1} - \frac{2pi}{3}) & cos(\theta_{e1} + \frac{2pi}{3}) \\ -sin(\theta_{e1}) & -sin(\theta_{e1} - \frac{2pi}{3}) & -sin(\theta_{e1} + \frac{2pi}{3}) \end{bmatrix} V_{ABC}$
    $i_d^1, i_q^1 \leftarrow$Calculate currents according to (3.78)-(3.79)
    $G(i) \leftarrow$ Compare currents according to (3.82)
end for
$Index \leftarrow argmin(G)$
$S_{out}(1:3) \leftarrow S(:, Index)$
Repeat with the other system
```

Algorithm 3 Algorithm for additive control of simplified model

```
$S \leftarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$    ▷ All possible configurations
for i = 1:length(S) do
    Calculating the phase voltages:
    $V_{ABC} \leftarrow \begin{bmatrix} \frac{VDC}{3}(2S(1,i) - S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) + 2S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) - S(2,i) + 2S(3,i)) \end{bmatrix}$
    Transforming to dq voltages:
    $V_{dq1} \leftarrow \begin{bmatrix} cos(\theta_{e1}) & cos(\theta_{e1} - \frac{2pi}{3}) & cos(\theta_{e1} + \frac{2pi}{3}) \\ -sin(\theta_{e1}) & -sin(\theta_{e1} - \frac{2pi}{3}) & -sin(\theta_{e1} + \frac{2pi}{3}) \end{bmatrix} V_{ABC}$
    $V_{dq2} \leftarrow \begin{bmatrix} cos(\theta_{e2}) & cos(\theta_{e2} - \frac{2pi}{3}) & cos(\theta_{e2} + \frac{2pi}{3}) \\ -sin(\theta_{e2}) & -sin(\theta_{e2} - \frac{2pi}{3}) & -sin(\theta_{e2} + \frac{2pi}{3}) \end{bmatrix} V_{ABC}$
    $i_d^1, i_q^1, i_d^2, i_q^2 \leftarrow$Calculate currents according to (3.78)-(3.81)
end for
$G(i,j) \leftarrow$ Loop through all combinations and compare the currents according to (3.84)
$Indexi, Indexj \leftarrow argmin(G)$
$S_{out}(1:6) \leftarrow [S(:, Indexi); S(:, Indexj)]$
```

Algorithm 4 Algorithm for additive control of complex model

```
$S \leftarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$    ▷ All possible configurations
for i = 1:length(S) do
    Calculating the phase voltages:
    $V_{ABC} \leftarrow \begin{bmatrix} \frac{VDC}{3}(2S(1,i) - S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) + 2S(2,i) - S(3,i)) \\ \frac{VDC}{3}(-S(1,i) - S(2,i) + 2S(3,i)) \end{bmatrix}$
    Transforming to dq voltages:
    $V_{dq1} \leftarrow \begin{bmatrix} cos(\theta_{e1}) & cos(\theta_{e1} - \frac{2pi}{3}) & cos(\theta_{e1} + \frac{2pi}{3}) \\ -sin(\theta_{e1}) & -sin(\theta_{e1} - \frac{2pi}{3}) & -sin(\theta_{e1} + \frac{2pi}{3}) \end{bmatrix} V_{ABC}$
    $V_{dq2} \leftarrow \begin{bmatrix} cos(\theta_{e2}) & cos(\theta_{e2} - \frac{2pi}{3}) & cos(\theta_{e2} + \frac{2pi}{3}) \\ -sin(\theta_{e2}) & -sin(\theta_{e2} - \frac{2pi}{3}) & -sin(\theta_{e2} + \frac{2pi}{3}) \end{bmatrix} V_{ABC}$
    $i_d^1, i_q^1, i_d^2, i_q^2 \leftarrow$Calculate currents according to (3.74)-(3.77)
end for
$G(i,j) \leftarrow$ Loop through all combinations and compare the currents according to (3.84)
$Indexi, Indexj \leftarrow argmin(G)$
$S_{out}(1:3) \leftarrow S(:, Indexi)$
$S_{out}(4:6) \leftarrow S(:, Indexj)$
```

3.5 Auxiliary functions and disturbances

The system that was explained above was an ideal system without noise and disturbances. Therefore, to make the system more realistic a couple of uncertainties had to be implemented.

3.5.1 Angle Sensor Noise and Filtering

In real-world scenarios, sensors have imperfections, usually in the form of noise. A Gaussian noise model was added to the actual signal to imitate this noise. To compensate for the noise, a Kalman filter with a constant velocity model was implemented. A constant velocity model assumed that the velocity was constant with some Gaussian distribution. Thereby the model was:

$$\begin{array}{c} x_k = A_{k-1}x_{k-1} + u_{k-1} + q_{k-1} \\ y_k = Hx_k + r_k \end{array}, \tag{3.85}$$

$$x = \begin{bmatrix} \theta_e \\ \omega_e \end{bmatrix}, A = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, u = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, q_{k-1} \sim (\begin{bmatrix} 0 \\ 0 \end{bmatrix}, Q), H = \begin{bmatrix} n_p & 0 \end{bmatrix}, r_k \sim (0, R),$$

where θ_e was the electrical angle, ω_e the electrical angular velocity, T the sampling time, q_{k-1} and r_k were Gaussian random variables with zero mean and covariance Q and R. The first step to implement the Kalman filter is the prediction step. The filter made a prediction of the next time step based on knowledge about earlier measurements. This could be described by the following equations:

$$\begin{aligned} \hat{x}_{k|k-1} &= A_{k-1}\hat{x}_{k-1|k-1} + u_k \\ P_{k|k-1} &= A_{k-1}P_{k-1|k-1}A_{k-1}^{\top} + Q \end{aligned} , \tag{3.86}$$

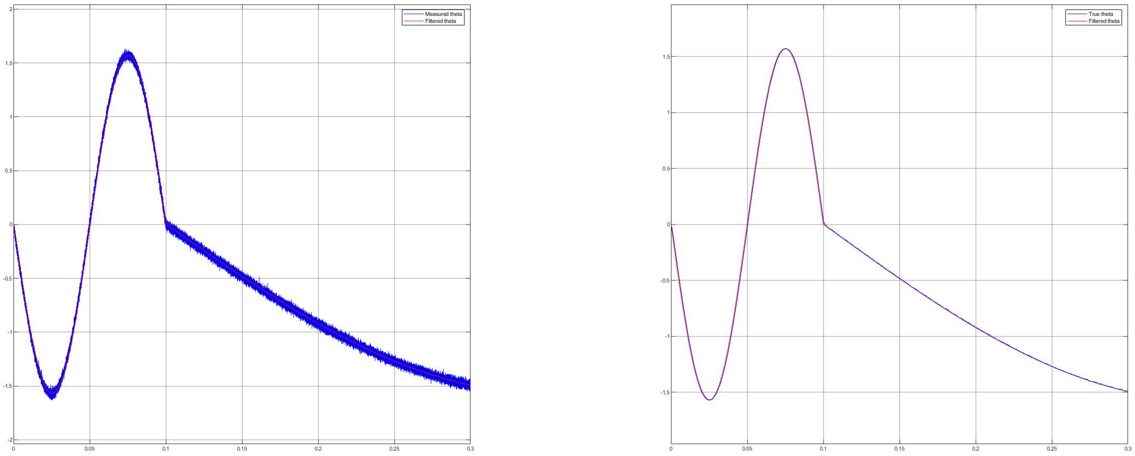
$$P_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} .$$

Then, the prediction and the measured value were compared in the update step:

$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k v_k \\ P_{k|k} &= P_{k|k-1} - K_k S_k K_k^{\top} \end{aligned} , \tag{3.87}$$

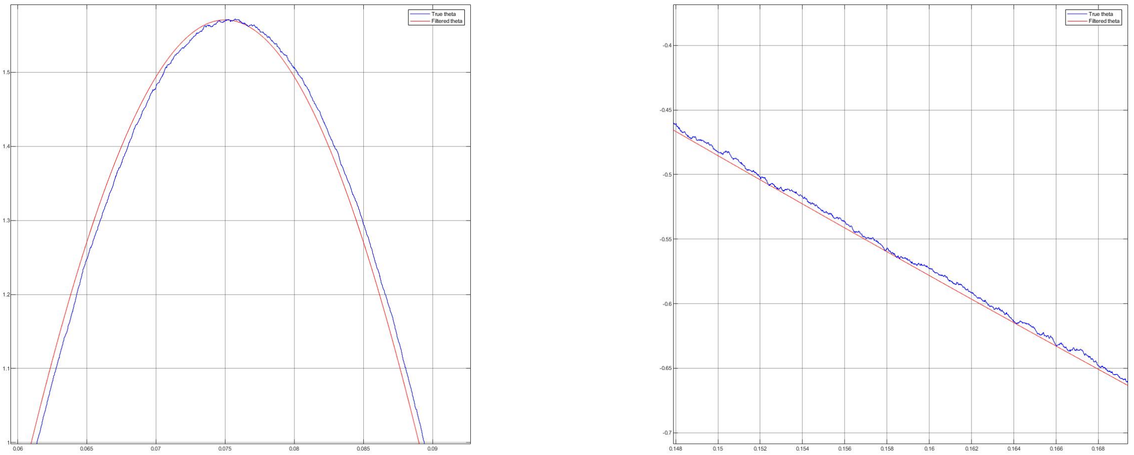
$$\begin{aligned} K_k &= P_{k|k-1}H^{\top}S_k^{-1} \\ v_k &= y_k - H\hat{x}_{k|k-1} \\ S_k &= HP_{k|k-1}H^{\top} + R \end{aligned} . \tag{3.88}$$

The results from the tuned filter can be seen in Figure 3.23.



(a) Non-filtered and filtered signal.

(b) True and filtered signal.



(c) Zoomed in on filter response at fast changing trajectory.

(d) Zoomed in on filter response at slow changing trajectory.

Figure 3.23: Performance of the Kalman angle filter.

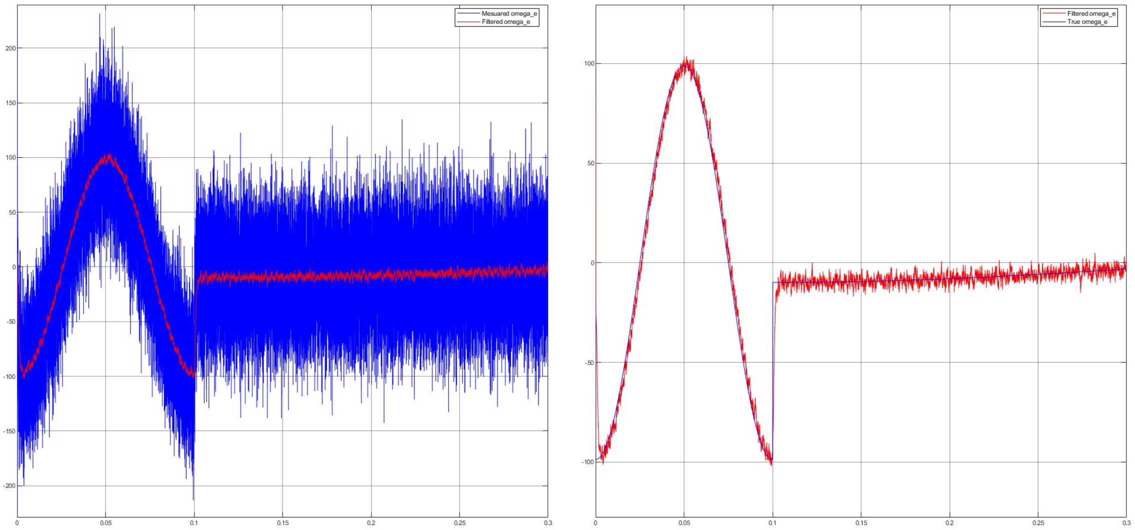
3.5.2 Velocity Filtering

The system did not include a velocity sensor. This means that the velocity had to be calculated using the angle. A discrete derivative of the angle measurement would have resulted in a lot of noise, therefore the estimated angle was used. Two methods could be used to decrease the velocity estimate noise further. Firstly, the Kalman filter for the angle could be changed such that its angle estimate was less noisy. However, this would have hampered the responsiveness of the angle estimate. The second method was to low-pass filter the velocity. This would have induced a time shift in the velocity. As the motor control was very angle dependent the latter method was picked. The filtering was done by first calculating the discrete derivative of the angle using the central difference.

$$\omega_e = \frac{\theta_e(k) - \theta_e(k-2)}{2T}. \tag{3.89}$$

Then, apply a low-pass filter on the signal.

$$G = \frac{K_1}{K_2 s + K_3}. \tag{3.90}$$



(a) Non-filtered and filtered signal. **(b)** True and filtered signal.

Figure 3.24: Performance of the low-pass filter.

3.5.3 Current Sensor Noise and Filtering

Gaussian noise was added to the true current. To be able to compensate for the noise the project looked at different models to calculate the currents. The tested models were the discretised simplified PMSM equations (3.78)-(3.81) and the discretised complex PMSM (3.74)-(3.77). The equations were nonlinear, which meant that a regular Kalman filter could not be used. Both an EKF and a CKF were developed. The reason behind trying two different filters was that an EKF is not as computationally heavy as a CKF. However a EKF generally performs worse compared to the CKF, the more non-linear the system is. Therefore, by trying both, it was possible to evaluate if the performance gain was worth the extra computation. The project also evaluated the performance gain using the two different PMSM models. The EKF and CKF models were defined as

$$\begin{aligned} x_k &= f(x_{k-1}, u_{k-1}) + q_{k-1}, \\ y_k &= h(x_k) + r_k, \end{aligned} \tag{3.91}$$

$$x = \begin{bmatrix} i_{d_1} \\ i_{q_1} \\ i_{d_2} \\ i_{q_2} \end{bmatrix}, u = \begin{bmatrix} u_{d_1} \\ u_{q_1} \\ u_{d_2} \\ u_{q_2} \\ \omega_1 \\ \omega_1 \end{bmatrix}, q_{k-1} \sim (\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, Q), h(x_k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k, r_k \sim (\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, R). \tag{3.92}$$

Lastly $f(x_{k-1}, u_{k-1})$ was the process model which were (3.78)-(3.81) or (3.74)-(3.77). The extended Kalman filter worked by linearising in each step and based of that linearisation a prediction was made. The prediction step was calculated as

$$\begin{aligned} \hat{x}_{k|k-1} &= f(\hat{x}_{k-1|k-1}, u_k), \\ P_{k|k-1} &= f'(\hat{x}_{k-1|k-1}, u_k) P_{k-1|k-1} f'(\hat{x}_{k-1|k-1}, u_k) + Q, \\ f'(\hat{x}_{k-1|k-1}, u_k) &= \nabla f = \left[\tfrac{\delta f}{\delta x}\right]. \end{aligned} \tag{3.93}$$

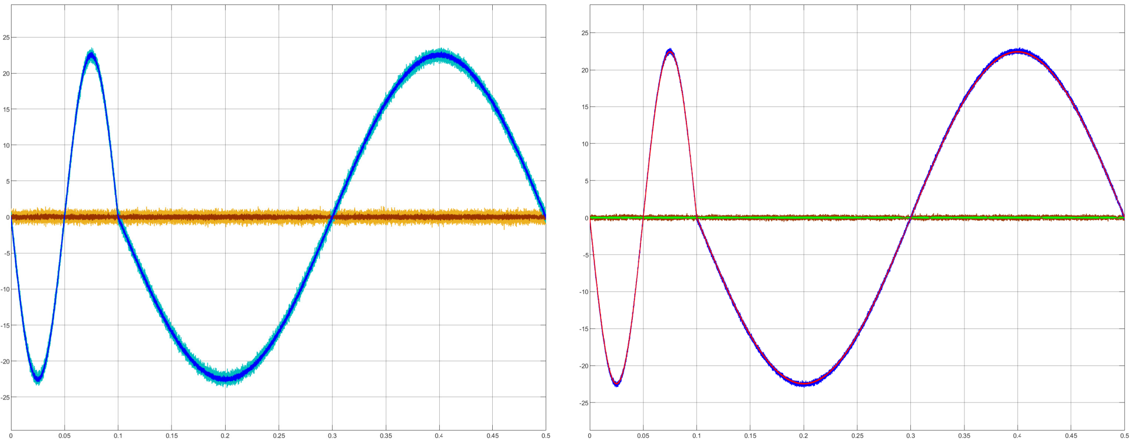
The update was implemented in the same way as in the Kalman filter, except that the measurement model was linearised in the current state. The update step was calculated as

$$\begin{aligned} K_k &= P_{k|k-1} h'(\hat{x}_{k|k-1})^\top S_k^{-1} \\ v_k &= y_k - h(\hat{x}_{k|k-1}) \\ S_k &= h'(\hat{x}_{k|k-1}) P_{k|k-1} h'(\hat{x}_{k|k-1})^\top + R \\ h'(\hat{x}_{k|k-1}) &= \nabla h = \left[\tfrac{\delta h}{\delta x}\right] \end{aligned} . \tag{3.94}$$

Then it was possible to calculate the state mean $\hat{x}_{k|k}$ (filtered x) and covariance $P_{k|k}$.

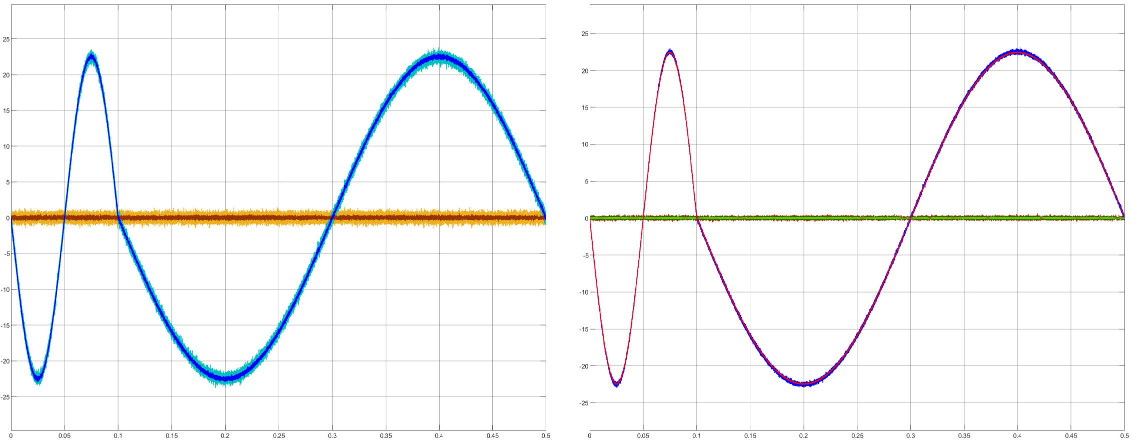
$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k v_k \\ P_{k|k} &= P_{k|k-1} - K_k S_k K_k^\top . \end{aligned} \tag{3.95}$$

The resulting filter estimate can be seen in Figure 3.25.



(a) (Simplified model) Non-filtered and filtered signal.

(b) (Simplified model) True and filtered signal.



(c) (Complex model) Non-filtered and filtered signal.

(d) (Complex model) True and filtered signal.

Figure 3.25: Performance of the EKF current filter.

The CKF could be divided into two parts a prediction step and a update step. The prediction was the first step. It was implemented by first defining the sigma points:

$$\begin{array}{c} \mathcal{X}_{k-1}^{i} = \hat{x}_{k-1|k-1} + \sqrt{n}(P_{k-1|k-1}^{\frac{1}{2}}), i = 1, 2, ..., n \\ \mathcal{X}_{k-1}^{i+n} = \hat{x}_{k-1|k-1} - \sqrt{n}(P_{k-1|k-1}^{\frac{1}{2}}), i = 1, 2, ..., n \\ \mathcal{W} = \frac{1}{2n} \end{array} . \tag{3.96}$$

Then the predicted mean $\hat{x}_{k|k-1}$ and covariance $P_{k|k-1}$ was computed as

$$\begin{array}{c} \hat{x}_{k|k-1} \approx \sum_{i=1}^{2n} f(\mathcal{X}_{k-1}^{i}, u_{k-1})\mathcal{W} \\ P_{k|k-1} \approx Q + \sum_{i=1}^{2n}(f(\mathcal{X}_{k-1}^{i}, u_{k-1}) - \hat{x}_{k|k-1})(f(\mathcal{X}_{k-1}^{i}, u_{k-1}) - \hat{x}_{k|k-1})^{\top}\mathcal{W} \end{array} . \tag{3.97}$$

Computing the update step was done by first defining new sigma points:

$$\begin{array}{c} \mathcal{X}_{k}^{i} = \hat{x}_{k|k-1} + \sqrt{n}(P_{k|k-1}^{\frac{1}{2}}), i = 1, 2, ..., n \\ \mathcal{X}_{k}^{i+n} = \hat{x}_{k|k-1} - \sqrt{n}(P_{k|k-1}^{\frac{1}{2}}), i = 1, 2, ..., n \\ \mathcal{W} = \frac{1}{2n} \end{array} . \tag{3.98}$$

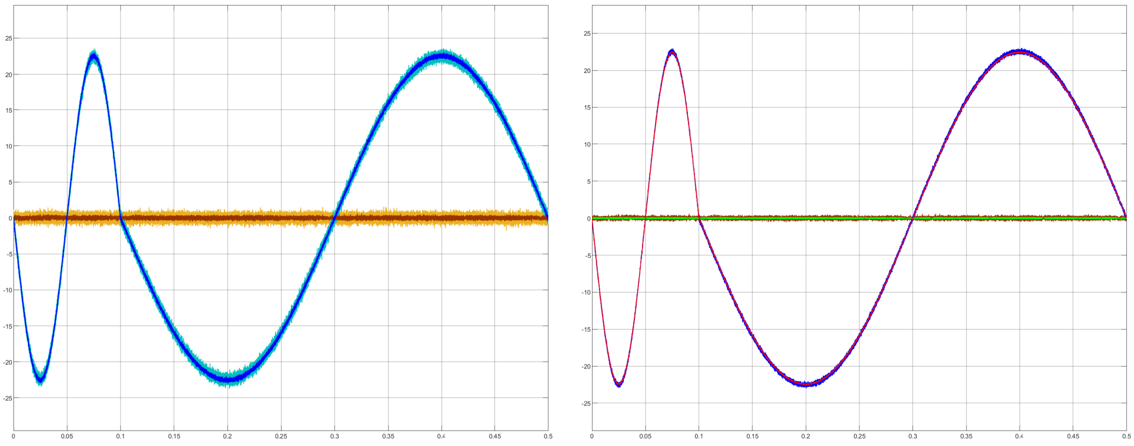
Then the predicted mean $\hat{y}_{k|k-1}$, predicted covariance of the measurement S_k and the cross-covariance of the state and measurement C_k could be found as

$$\begin{aligned} \hat{y}_{k|k-1} &= \textstyle\sum_{i=1}^{2n} h(\mathcal{X}_k^i)\mathcal{W}, \\ C_k &= \textstyle\sum_{i=1}^{2n}(\mathcal{X}_k^i - \hat{x}_{k|k-1})(h(\mathcal{X}_k^i) - \hat{y}_{k|k-1})^\top \mathcal{W}, \\ S_k &= R + \textstyle\sum_{i=1}^{2n}(h(\mathcal{X}_k^i) - \hat{y}_{k|k-1})(h(\mathcal{X}_k^i) - \hat{y}_{k|k-1})^\top \mathcal{W}. \end{aligned} \tag{3.99}$$

From these values, the state mean $\hat{x}_{k|k}$ (filtered x) and covariance $P_{k|k}$ was calculated.

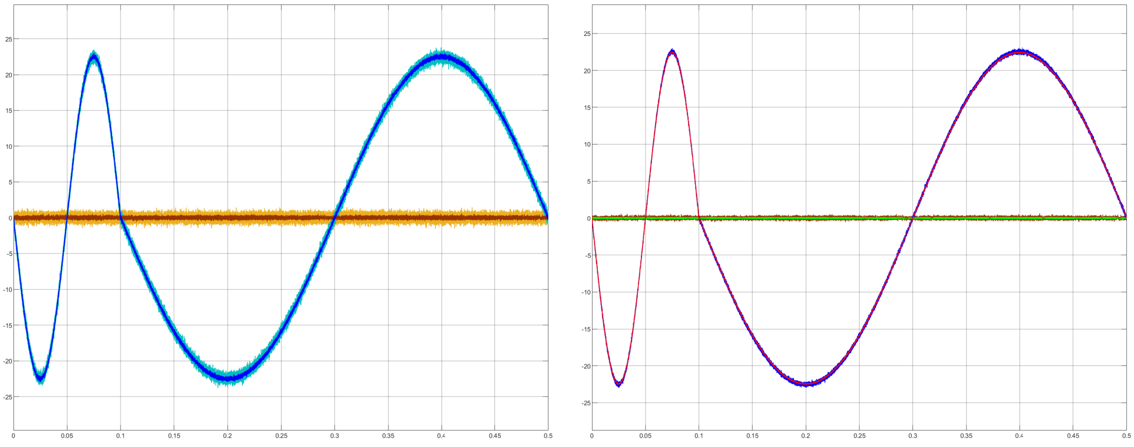
$$\begin{aligned} \hat{x}_{k|k} &= \hat{x}_{k|k-1} + C_k S_k^{-1}(y_k - \hat{y}_{k|k-1}), \\ P_{k|k} &= P_{k|k-1} - C_k S_k^{-1} S_k (C_k S_k^{-1})^\top. \end{aligned} \tag{3.100}$$

The resulting current estimates can be seen in Figure 3.26.



(a) (Simplified model) Non-filtered and filtered signal.

(b) (Simplified model) True and filtered signal.



(c) (Complex model) Non-filtered and filtered signal.

(d) (Complex model) True and filtered signal.

Figure 3.26: Performance of the CKF current filter.

Comparing Figure 3.25 and 3.26 there was no obvious difference. Thus a performance measure needed to be deviced to compare the results. The project chose to compare the filter using the H_2-norm on the i_d and i_q currents relative to the true trajectory.

$$\|i_{filtered} - i_{true}\|_{H_2} = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(i_{filtered}(k) - i_{true}(k))^2}. \tag{3.101}$$

Method	H_2-Norm i_q	H_2-Norm i_d
No filters	0.5162	0.5163
EKF Simplified model	0.2271	0.1497
EKF Complex model	0.2162	0.1515
CKF Simplified model	0.2273	0.1492
CKF Complex model	0.2172	0.1516

Table 3.5: H_2-norm of i_d and i_q for each of the filters.

As can be seen in table 3.5, there were very small differences between the EKF and the CKF. As mentioned earlier CKF is a much more computationally heavy operation than EKF. In this case, there was a minor benefit of using the CKF, but it was not enough to compensate for its computational need. Therefore the project chose the EKF. Comparing the models used in the filter there was a minor difference in performance ($\approx$ 5%). The benefit of the simplified model was that it was less computationally heavy but also that each three-phase system was independent of each other, which made it possible to create a fully redundant system. The project, therefore, chose to use the EKF with the simplified model.

3.5.4 Switch Losses

When using an inverter in the real world there are losses when switching between different inverter configurations. The losses are a consequence of avoiding short-circuiting the inverter. Short-circuiting will happen if both the upper and lower switch is on at the same time. To avoid this when switching from the upper side to the lower side, the inverter needs to have both the upper and lower switch off for a short period. This can be seen in 3.27.

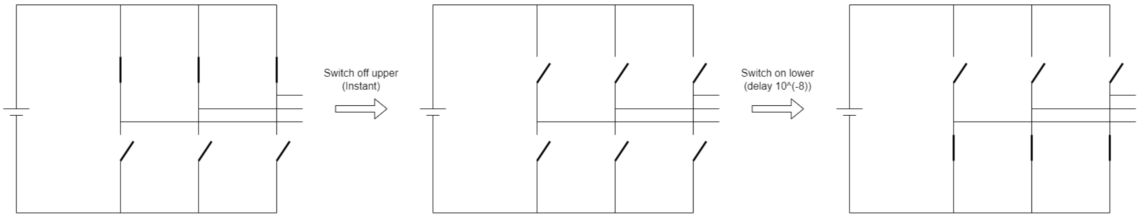


Figure 3.27: An example of where the switch losses comes from

During the time between the last state and the new state, the voltages will be the same as in the last state[10]. This behaviour could, therefore, be implemented as a constant delay in the system input. This was implemented by up-sampling the system and then delaying it by one time step, then down-sampling it again. The Simulink implementation can be seen in 3.28

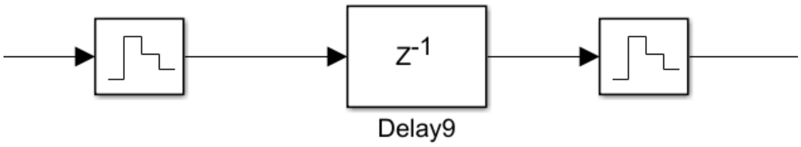


Figure 3.28: Simulink implementation of Switch losses.

As the simulation was run in 1 MHz the delay had to be 1 μs, usual switching losses are at the scale of 10 ns.

3.5.5 Cogging Torque

One of the major sources of torque ripple for the PMSM was the cogging torque which was a consequence of the rotor magnets aligning with the stator teeth. To illustrate this behaviour Figure (3.29) was generated, an example with 12 stator teeth. Four significant points of interest were shown in the figure, the stable zero which occurred when the rotor was aligned with a stator tooth, when the angle increased the force between the passed tooth and the rotor increased and resulted in a torque, opposite the way of travel. Once the rotor was some distance away from the tooth, the amplitude of the cogging torque reached its maximum and then started to decrease as the next tooth began to influence the rotor. When the rotor was centred between the two teeth the apparent cogging torque was zero but soon started to increase when it closed in on the next stator tooth. This process happened for every stator tooth, and thus, for a full revolution of the rotor, 12 periods were produced, [25].

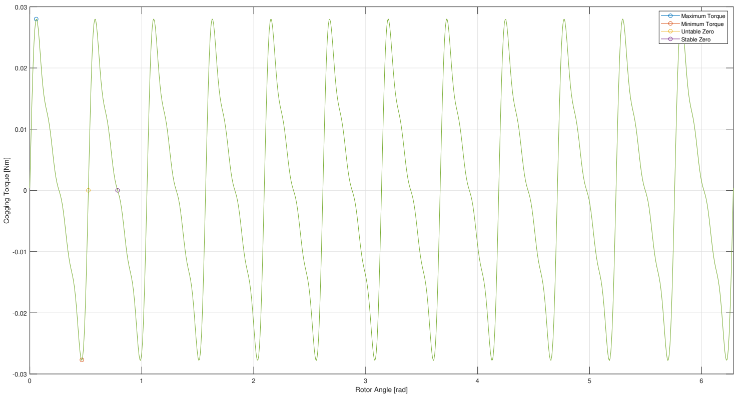


Figure 3.29: Cogging torque for a PMSM with 12 stator teeth.

Due to the periodicity of the disturbance, the cogging torque could be modelled as the sum

$$T_{cog} = \sum_{i=1}^{N} T_i sin(iM\theta_r), \tag{3.102}$$

where M is the number of stator teeth[26]. Assuming the model of the cogging torque was accurate and a responsive controller coupled to the motor, the disturbance could be cancelled. By subtracting the modelled cogging torque for the current angle from the reference torque the disturbance could almost be perfectly rejected. This results in a smoother torque tracking performance.

However, finding a perfect representation of the cogging torque is not possible in any real-world scenario, as the higher-order harmonics are not measurable due to limited measuring equipment with inherent noise. Still, compensating for lower-order terms where most of the energy content resides will have a large impact on the torque ripple, and thus, this project investigated this. To model this, the cogging torque was added to the output torque of the motor model, and two terms, the fundamental and one harmonic, were compensated for in the controller.

In a practical setup the cogging torque terms cannot be known perfectly beforehand. To find a suitable approximation two methods could be used:

1. FEM analysis of the motor.
2. Experimental analysis.

Approach number 1, used in [25], could achieve good results as long as the motor model used in the FEM analysis was accurate to its real-world counterpart. Method 2 could be realised by spinning the motor with a known speed and measuring the torque in the axle. All deviations would, in theory, be due to the cogging effect, and the aforementioned expression for the cogging torque could be interpolated on this data. However, this method will be limited by the measuring equipment used, which in reality will be noisy and thus, accurately finding higher-order harmonics will be impossible.

3.6 Implementation of Non-Essential Sensors

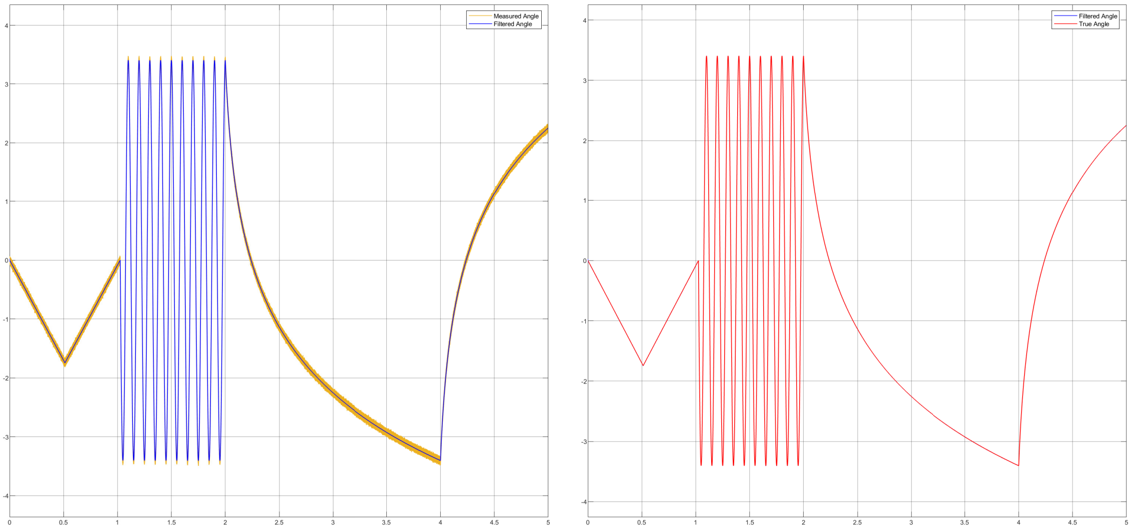
In addition to the developed controllers and filter subsystem, simulations were made with added angular velocity sensors that were not implemented on the hardware. This was done to illustrate the potential benefits of adding these sensors to the system. The angular velocity was a needed quantity in all controllers as it was used as a scheduling parameter in the GSCs, a state in the state-space equations for the various MPC controllers, and also a factor in the feedforward terms of the LQRs.

In the base system, no direct measurement of the angular velocity was made. Instead, the angle was measured and filtered with the constant velocity model as described in Section 3.5.1. The angular velocity was then estimated with the Euler method for discrete derivatives. As the angle estimate was still noisy after the filtering, this noise was amplified, giving a poor estimate of the angular velocity.

The underlying model for the filter was almost identical as in (3.85), except for the measurement model, which now has two outputs as in (3.103).

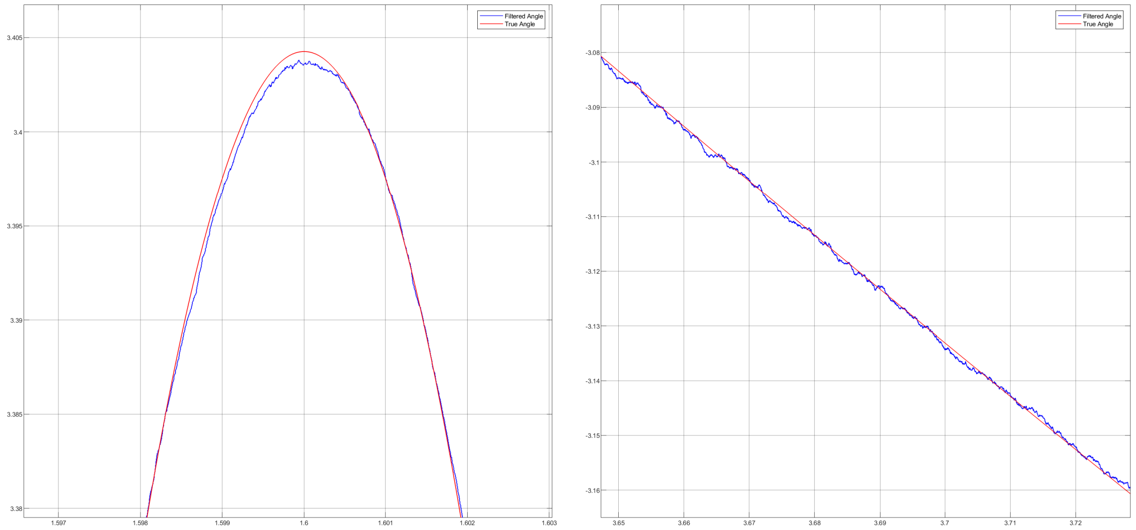
$$y_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta_e \\ \omega_e \end{bmatrix} + r_k, \quad r_k \sim \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix} \right). \tag{3.103}$$

There were large improvements in both the angle and angular velocity. This was because when predicting the next angle the velocity was used. Therefore, by having a better velocity reading, the predictions of the next step became better, which, in turn, made the filtered angle better. Before, the angular velocity in the Kalman filter was always zero because it was never updated with a sensor reading. The result for the angle and angular velocity can be seen in Figure 3.30-3.31



(a) Non-filtered and filtered signal.

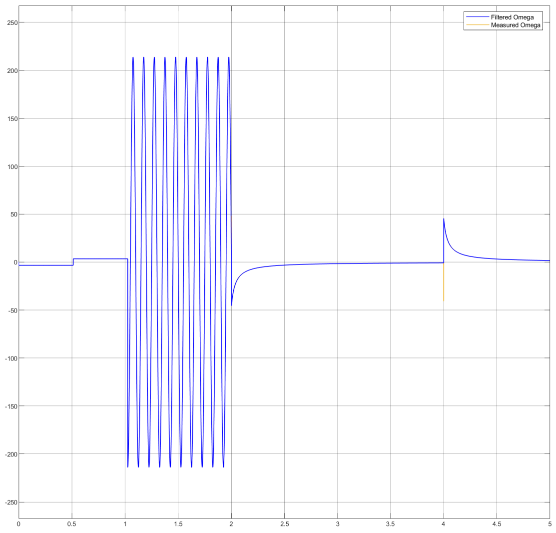
(b) True and filtered signal.



(c) Zoomed in on filter response at fast changing trajectory.

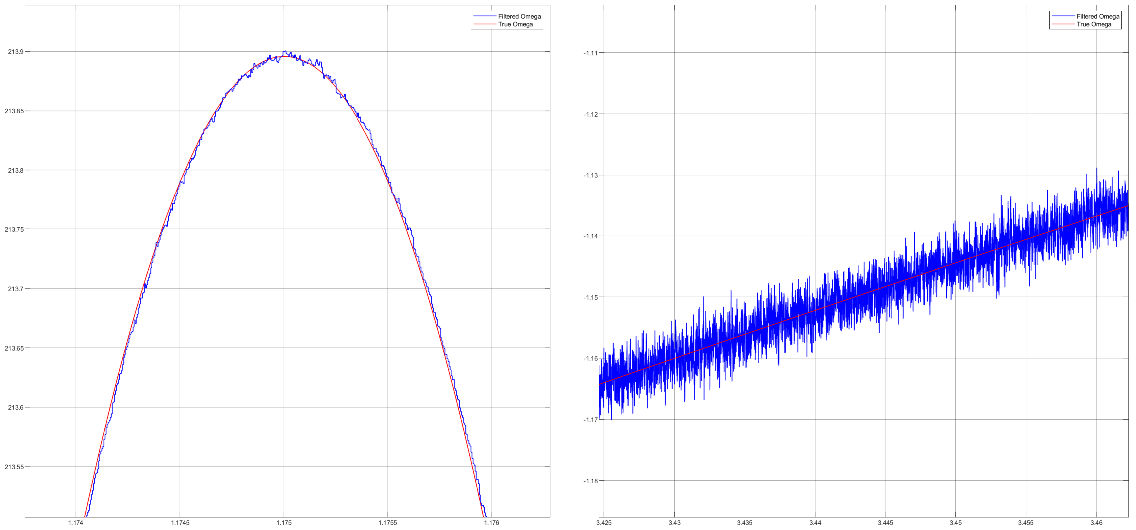
(d) Zoomed in on filter response at slow changing trajectory.

Figure 3.30: Performance of the Kalman angle filter.



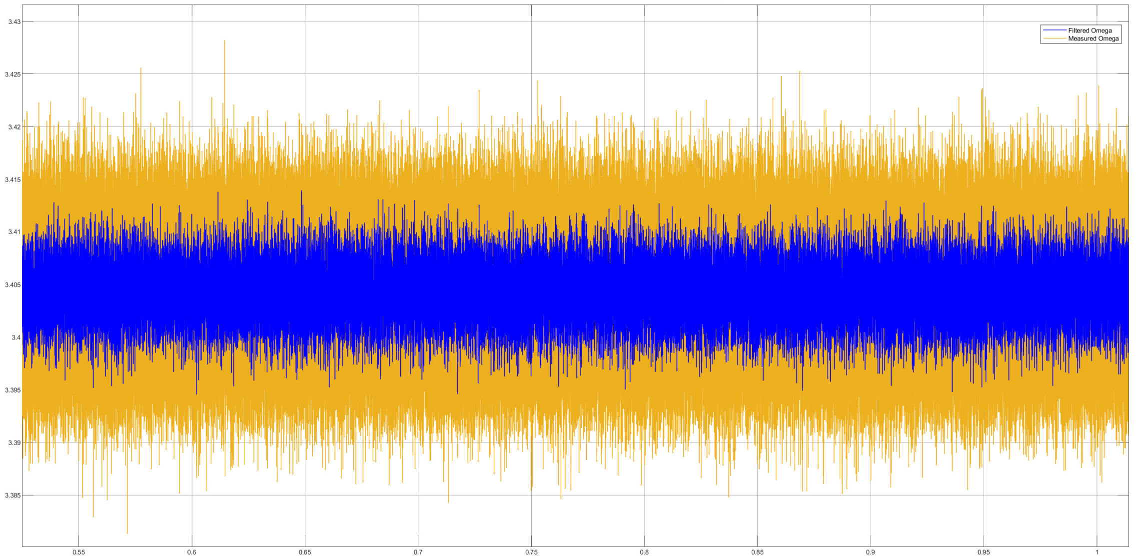
(a) Non-filtered and filtered signal.

(b) True and filtered signal.



(c) Zoomed in on filter response at fast changing trajectory.

(d) Zoomed in on filter response at slow changing trajectory.



(e) Zoomed in on filter response at slow changing trajectory compared to the measured signal.

Figure 3.31: Performance of the Kalman angular velocity filter.

3.7 Hardware Setup and Calibration

The hardware setup was compromised of the following parts:

1. *LAUNCHXL-F28379D*
2. *BOOSTXL-DRV8305EVM*
3. *DC Power Supply*
4. *Six-Phase PMSM*
5. *Sin/Cos Encoder*
6. *Low-Side Current Shunt Sensor*

The Texas Instrument board *LAUNCHXL - F28379D* and the VSI *BOOSTXL - DRV8305EVM* were used to implement the hardware solution. The *BOOSTXL - DRV8305EVM* inverter was a three-phase inverter, and thus, two were used in parallel to control the motor. For redundancy also two *LAUNCHXL - F28379D* were used, these two circuit boards were used to run the software. In Figure 3.32 a picture of the boards can be seen.

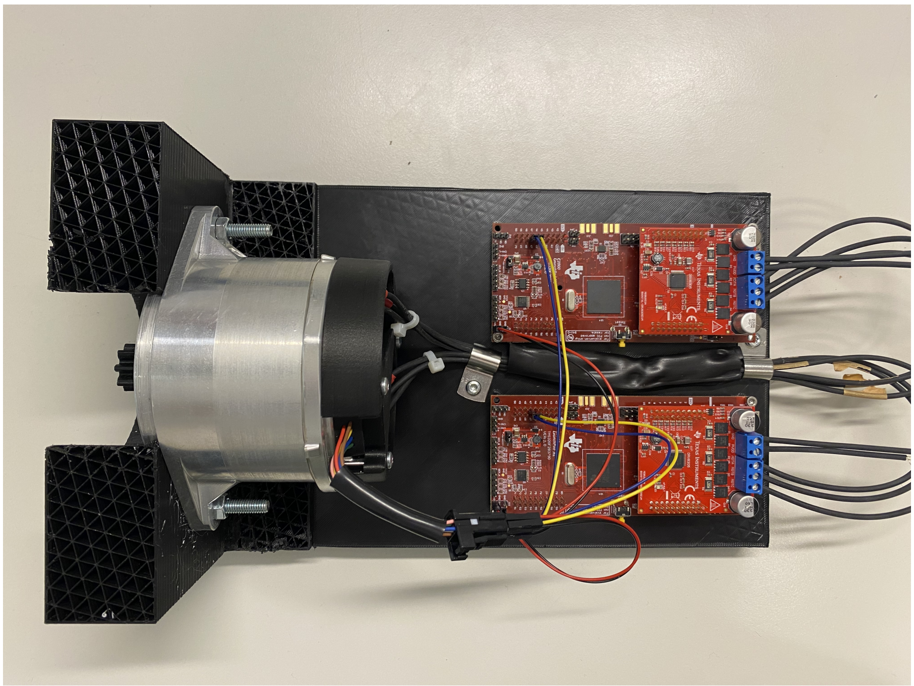


Figure 3.32: The two *LAUNCHXL-F28379D* and *BOOSTXL-DRV8305EVM* boards connected to the angle encoders and motor.

The encoder on the motor was a Sin/Cos encoder which could measure the angle with a network of magnetoresistances. The outputs from the sensors were two signals the sine and cosine of the rotor angle. To find the angle $atan2()$ was used on these two signals. The sensor was connected to the *LAUNCHXL - F28379D* where the calculations were performed.

The control algorithms require feedback of the dq - currents. To find these the current shunt sensors on the *BOOSTXL - DRV8305EVM* inverter was used. These shunt sensors provided information about the current leaving the inverter. By using the Park transform on these phase currents the correct dq currents were found and could be used for feedback.

With all components connected as above, the simple LQR strategy was implemented and tested. The simple LQR was chosen because the simulations showed that the controllers performed similarly and that it was the least computationally heavy controller.

3.7.1 Sensor Calibration

Both the Sin/Cos sensors and the current shunt sensors were analogue and needed calibration to be used. Both of them were able to read negative and positive values; therefore, they had to be centred around zero. This was achieved by controlling the motor at a constant speed and finding the mean of each sensor value, which was set to the offset of the corresponding sensor. The Sin/Cos encoder then needs to be scaled from -1 to 1, which was accomplished by dividing the sensor value by its peak-to-peak value ($\frac{2}{V_{p-p}}$). However, this could not be done for the current sensor. Instead, the project used the relationship between the torque and the q-currents. However, as the project did not include a torque sensor it was not possible to measure torque. To resolve this issue a known torque was applied to the motor, by suspending a weight from a wheel attached to the rotor like in Figure 3.33. By increasing the q-current depending on the angle the current measurements which made the motor reach a steady state could be observed. The corresponding q-current to the applied torque could be identified with (2.10) and a correct scaling factor for the current sensors was found.

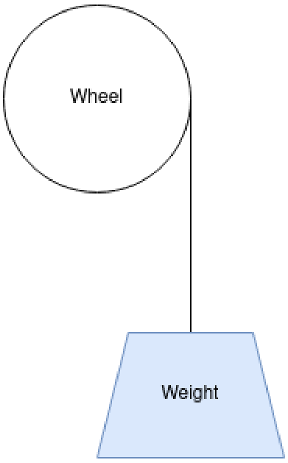


Figure 3.33: Calibration setup for the Low-Side Current Shunt Sensor.

4

Results

Various simulations were done on the system for the different motor controllers and auxiliary functions. This chapter aims at showcasing the differences of these controllers to better understand their strengths and weaknesses. Because of the large number of configurations, most figures were deemed superfluous. However, all simulated trajectories can be found in the appendix A.

4.1 Step Response

The torque controller should be responsive. A step response analysis was done to test the responsiveness of each controller. To keep the performance measurements comparable between the different controllers, the sensor noise was removed from the test, the angle was kept at zero for the entire simulation, and the torque step was from 0 Nm to 2 Nm. The simulated developed torque can be found in section A.1 and the step time in Table 4.1. From the table, it is clear that all controllers are very responsive, and ultimately, it is not the controller itself that limits the responsiveness but rather the inverter. The three different MPCs take the fastest possible route toward the set point and are only limited by the inverter and DC link. The remaining four controllers take slightly longer due to limitations in the PWM generation, but this difference is negligible for the overall responsiveness of these controllers.

Method	Step response time [μs]
AS-MPC	410
AC-MPC	410
SS-MPC	410
S-LQR	437
C-LQR	437
S-GSC	437
C-GSC	437

Table 4.1: Step responses from 0 Nm to 2 Nm with zero angular velocity and zero angle. Measured at 63% of settling value.

4.2 Torque Tracking Capabilities for a Trajectory

The torque tracking capabilities were measured using the H_2-norm (3.101) and H_∞-norm:

$$\|T_e(t) - T_{ref}(t)\|_\infty = \underset{t}{max} \quad |T_e(t) - T_{ref}(t)| \tag{4.1}$$

The H_2-norm measures the average absolute deviation while the H_∞-norm measures the max absolute deviation. To make the comparison fair, trajectories for both the torque and angle reference were picked such that the controllers are evaluated under different angular velocities, angles and torques. The trajectory can be seen in Figures 4.1 and 4.2, and were intentionally made difficult to challenge the control systems. In the following tests, the PMSM is decoupled from the axle to be able to measure the ability to track a torque reference without an outer disturbance. Thereby, the angle and angular velocity are set to the reference, without the interaction of the PMSM.

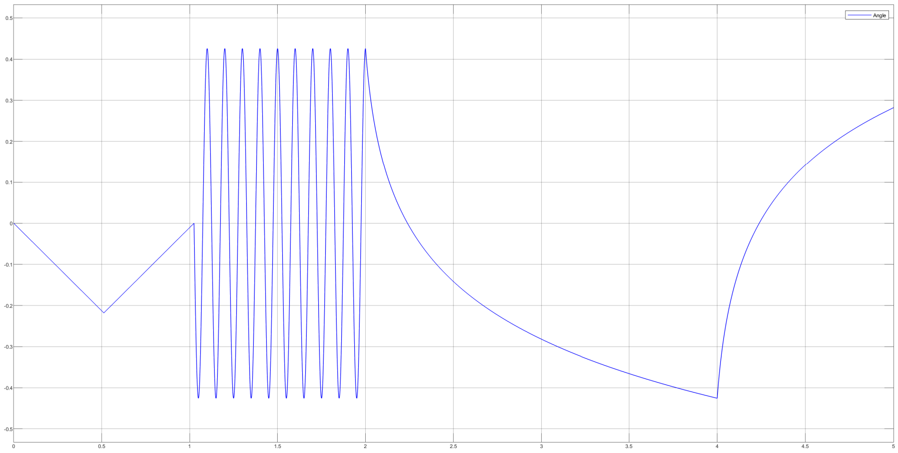


Figure 4.1: The angle reading without noise.

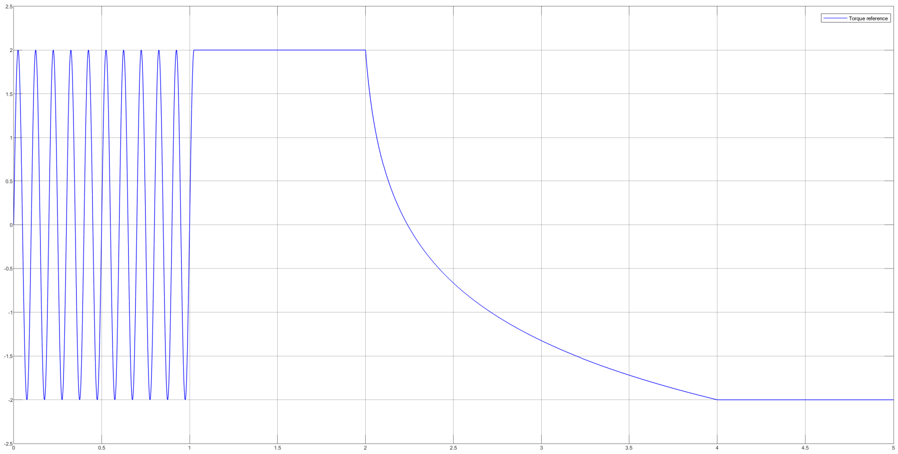


Figure 4.2: The torque reference.

4.2.1 Ideal Case

To be able to compare the different filter and control combinations it is important to know what is ultimately achievable. The results in Table 4.2 were obtained by ignoring sensor noise for the angle and current sensors.

Method	**H_2-norm**	**H_∞-norm**
MPC		
AC-MPC	0.0017	0.0059
AS-MPC	0.0016	0.0061
SS-MPC	0.0043	0.0117
LQR		
C-LQR	0.0066	0.0287
S-LQR	0.0066	0.0283
GSC		
C-GSC	0.0073	0.0298
S-GSC	0.0073	0.0298

Table 4.2: Simulation results for controllers with no sensor noise.

4.2.2 Control without Filters

To begin comparing the different developed controllers, the system was simulated once with every type of controller and no added filtering on the sensor measurements. The angle and angular velocity were assumed to be separated from the PMSM motor and completely controlled by the angle reference. This made it possible to eliminate potential resonant behaviour between the PMSM and the dynamometer and study the developed torque in isolation.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0129	0.0640
AS-MPC	0.0115	0.0575
SS-MPC	0.0113	0.0565
LQR		
C-LQR	0.0136	0.0659
S-LQR	0.0138	0.0673
GSC		
C-GSC	0.0138	0.0628
S-GSC	0.0138	0.0633

Table 4.3: Simulation results for controllers with no filtering on sensor input

The results in Table 4.3 show overall similar and comparable results for all the different controllers. The maximum deviation from the reference torque is between 56.5 to 67.3 mNm, making the torque ripple 2.8% to 3.4%. The controller with the best performance is the SS-MPC, and its transient performance can be seen in Figure 4.3.

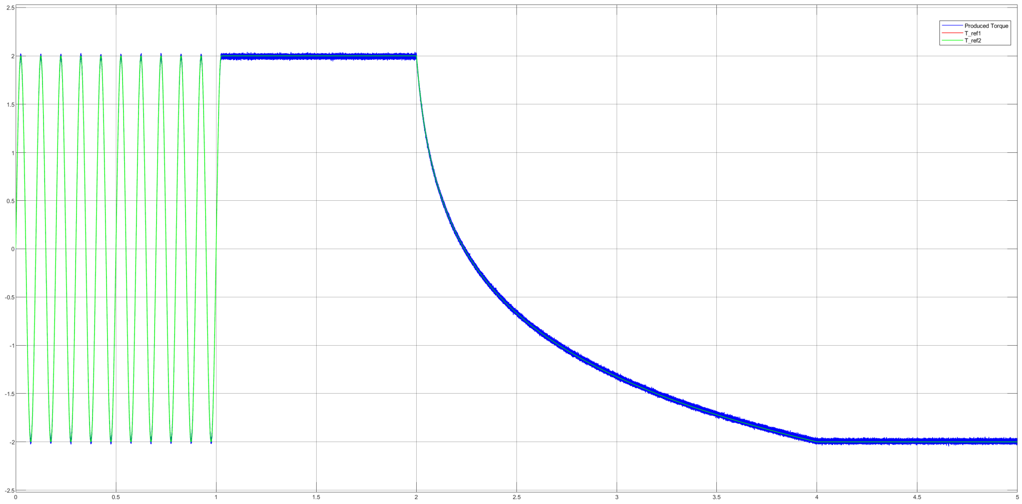


Figure 4.3: SS-MPC without filters.

4.2.3 Simulation with Angle Filter

For this simulation the angle filter from Section 3.5.1 was added to the system. The filter reduces the noise on the angle signal and, hence, also the angular velocity.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0120	0.0636
AS-MPC	0.0117	0.0646
SS-MPC	0.0113	0.0555
LQR		
C-LQR	0.0126	0.0634
S-LQR	0.0126	0.0609
GSC		
C-GSC	0.0138	0.0653
S-GSC	0.0138	0.0656

Table 4.4: Simulation results with angle filtration

Overall, the improvement compared to Table 4.3 is small. However, a small decrease can be seen in the LQR and the MPC controllers. The controller with the best performance is the SS-MPC, and its transient performance is shown in Figure 4.4.

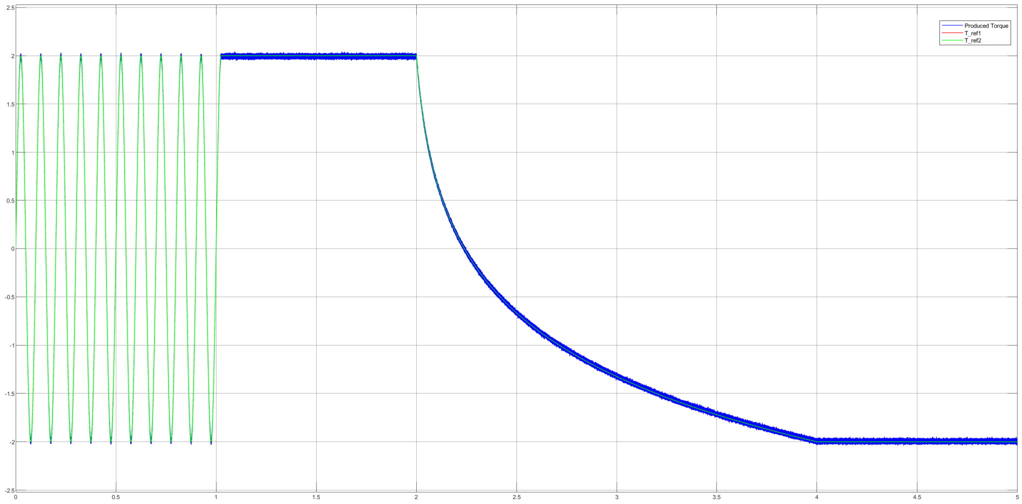


Figure 4.4: SS-MPC with only angle filter.

4.2.4 Simulation with Angle and Angular Velocity Filter

The same angle filter as above was used for this simulation but with an added low pass filter on the angular velocity, which still is very noisy.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0120	0.0652
AS-MPC	0.0117	0.0580
SS-MPC	0.0114	0.0595
LQR		
C-LQR	0.0126	0.0651
S-LQR	0.0126	0.0636
GSC		
C-GSC	0.0138	0.0650
S-GSC	0.0138	0.0622

Table 4.5: Simulation results with angle filter and low passed angular velocity

As can be clearly seen in Table 4.5, the additional low pass filter for the angular velocity does not contribute to a better performance in either H_2 or H_∞ sense for any controller compared to the results in Table 4.4. The controller with the best performance is the SS-MPC, and its transient performance can be seen in Figure 4.5.

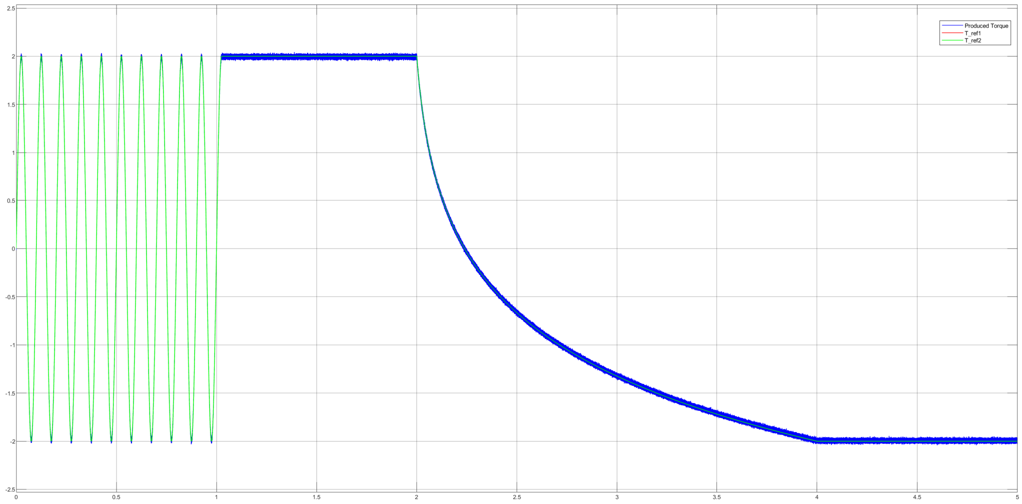


Figure 4.5: SS-MPC with angle filter and low-passed angular velocity filter.

4.2.5 Simulation with Angle, Angular Velocity and Current Filter

Now the last of the developed filters was added, the current EKF. This will reduce the apparent noise in the current estimations as can be seen in Table 3.5, which has a direct coupling to the current error and thus the control algorithm.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0066	0.0321
AS-MPC	0.0065	0.0320
SS-MPC	0.0068	0.0327
LQR		
C-LQR	0.0077	0.0398
S-LQR	0.0077	0.0443
GSC		
C-GSC	0.0082	0.0414
S-GSC	0.0082	0.0408

Table 4.6: Simulation result with angle filter, low passed angular velocity and current filter.

There are major improvements for every controller. With all filters added to the system the torque ripple has been reduced from $2.8 - 3.4\%$ to $1.6 - 2.2\%$. The largest improvement was seen with the AC-MPC which nearly halved the torque ripple with the filters compared to without any filtering. The controller with the best performance is the AS-MPC, and its transient performance is shown in Figure 4.6.

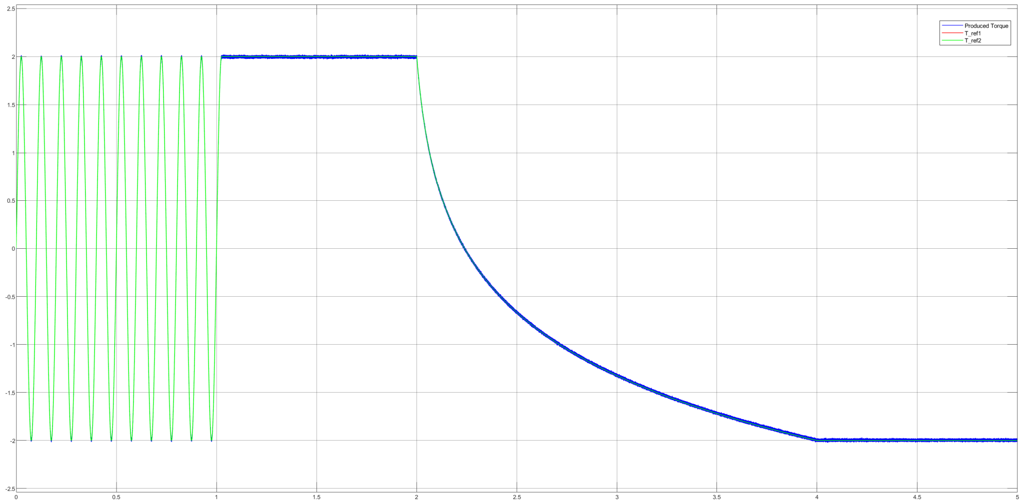


Figure 4.6: AS-MPC with angle filter, low-passed angular velocity filter and current filter.

4.2.6 Simulation with Current Filter

Due to the large improvement by adding the current filter, the filter was also tested by itself. This simulation only filters the currents and takes noisy signals for angle and angular velocity.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0099	0.0473
AS-MPC	0.0068	0.0342
SS-MPC	0.0079	0.0380
LQR		
C-LQR	0.0109	0.0566
S-LQR	0.0109	0.0548
GSC		
C-GSC	0.0082	0.0407
S-GSC	0.0082	0.0444

Table 4.7: Simulation result with only current filter

Comparing Table 4.7 and 4.6, the angle and angular velocity filters clearly contribute more to the result in Table 4.6 than is readily visible in Table 4.5. This is most likely due to the various Park transformations used in the controllers, which utilise the angle to transform the currents and the usage of the angle in the current EKF. An interesting exception to this behaviour is the GSC controller, which has no difference in performance, comparing the results in Table 4.7 and 4.6. The controller with the best performance is the AS-MPC, and its transient performance can be seen in Figure 4.8.

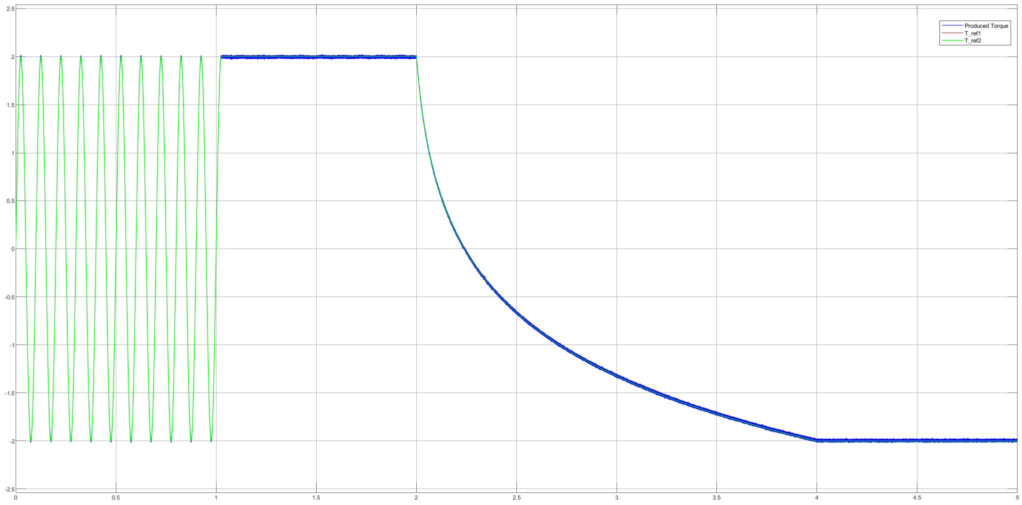


Figure 4.7: AS-MPC with only current filter

4.2.7 Simulation with Angular Velocity Sensor

To investigate what impact adding an angular velocity sensor would make on the system this simulation was made. A realistic noise was added to the true angular velocity and this signal was then filtered with the angle measurement.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0066	0.0329
AS-MPC	0.0065	0.0321
SS-MPC	0.0068	0.0330
LQR		
C-LQR	0.0077	0.0408
S-LQR	0.0077	0.0385
GSC		
C-GSC	0.0083	0.0397
S-GSC	0.0082	0.0407

Table 4.8: Simulation result with angular velocity sensor.

As can be seen by comparing the results in Table 4.6 and Table 4.8, there are no improvements from implementing an angular velocity sensor. The controller with the best performance is the AS-MPC, and its transient performance is shown in Figure 4.8.

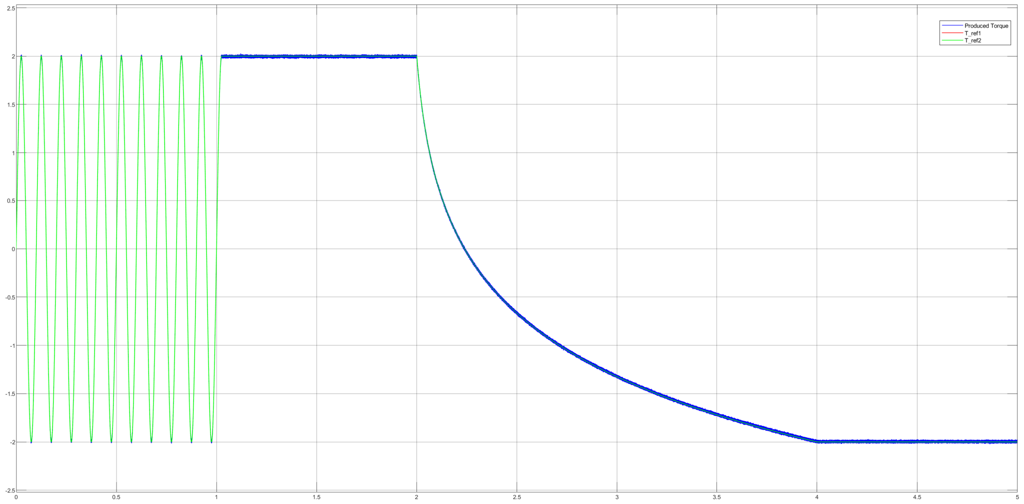


Figure 4.8: AS-MPC with angle filter, angular velocity sensor, angular velocity filter and current filter.

4.2.8 Simulation with Modelled Cogging Torque and No Compensation

The cogging model from Section 3.5.5 was used to investigate the impact on the developed torque and the possibility of compensating for the disturbance. Firstly, the controllers were simulated with angle, angular velocity, and current filters but with no compensatory term for the cogging torque.

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0175	0.0598
AS-MPC	0.0175	0.0560
SS-MPC	0.0176	0.0614
LQR		
C-LQR	0.0180	0.0624
S-LQR	0.0180	0.0627
GSC		
C-GSC	0.0181	0.0629
S-GSC	0.0182	0.0649

Table 4.9: Simulation result with cogging.

From Table 4.9, we see that the controller performance is roughly equal for all controllers. This is to be expected as the cogging torque is the largest contributing factor to deviations from the reference trajectory. For example, the trajectory of the AS-MPC can be seen below in Figure 4.9.

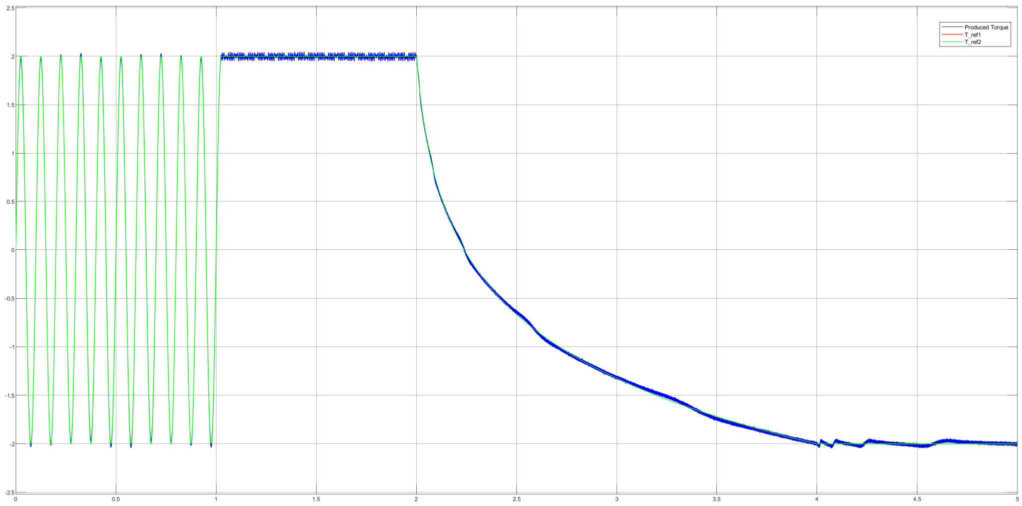


Figure 4.9: AS-MPC with modelled cogging torque and no compensation term.

4.2.9 Simulation with Modelled Cogging Torque and Added Compensation

The results below were achieved by adding the compensatory term described in Section 3.5.5. It can be seen that although not perfectly restoring the performance to the results in Table 4.6 the compensation still reduce the torque ripple by a large margin. This can be seen in Figure 4.10 where the compensatory term has been added to the controller in Figure 4.9.

Method	H_2**-Norm**	H_∞**-Norm**
MPC		
AC-MPC	0.0073	0.0505
AS-MPC	0.0072	0.0505
SS-MPC	0.0075	0.0480
LQR		
C-LQR	0.0083	0.0516
S-LQR	0.0083	0.0536
GSC		
C-GSC	0.0088	0.0553
S-GSC	0.0088	0.0535

Table 4.10: Simulation result with cogging.

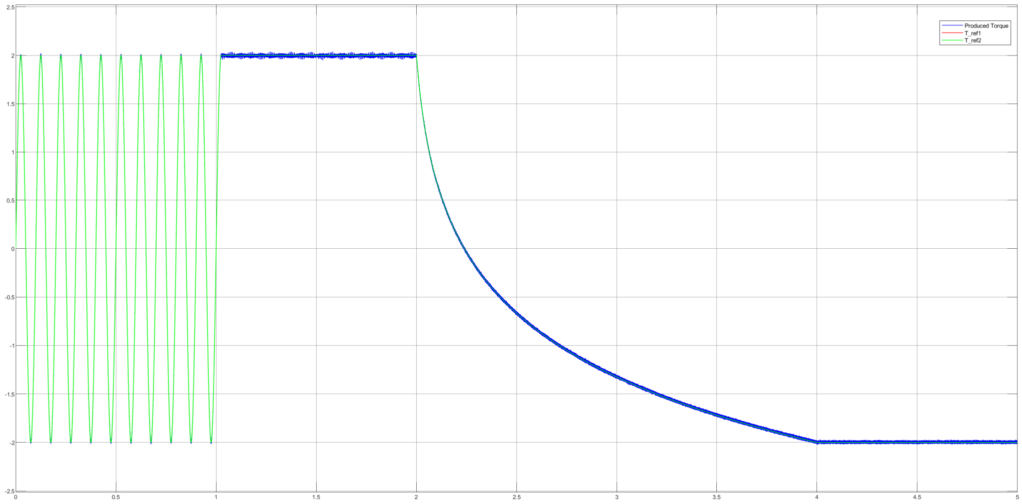


Figure 4.10: AS-MPC with modelled cogging torque and compensation in the controller.

4.3 Final system

This simulation includes an angle filter, angular velocity low-pass filter, current filter, cogging compensation, and most importantly the dynamometer. The simulations above have the physical system decoupled from the PMSM to see the performance of each control system without the interference of the dynamometer. This simulation was made to see how the control systems would react when they have an effect on the physical system. This meant that forces applied by the PMSM had an effect on the axle dynamics which the the dynamometer tries to control according to the angle trajectory. As can be understood the angle trajectory will look a bit different for all the simulations as their effect on the axle will be different. The coupling also meant that the angular velocity could not change arbitrarily fast, this makes this test easier than the rest as the requirement for responsivness in the PMSM is not instant but instead related to the tardiness of the DC motor control system. This can be seen in Table 4.11 where the performance measures are consistently better for all controllers compared to Table 4.10. The best-performing controller in this test was the AC-MPC, which can be seen in Figure 4.11

Method	H_2**-norm**	H_∞**-norm**
MPC		
AC-MPC	0.0066	0.0328
AS-MPC	0.0073	0.0373
SS-MPC	0.0074	0.0426
LQR		
C-LQR	0.0080	0.0448
S-LQR	0.0081	0.0434
GSC		
C-GSC	0.0083	0.0443
S-GSC	0.0083	0.0385

Table 4.11: Simulations of the final system.

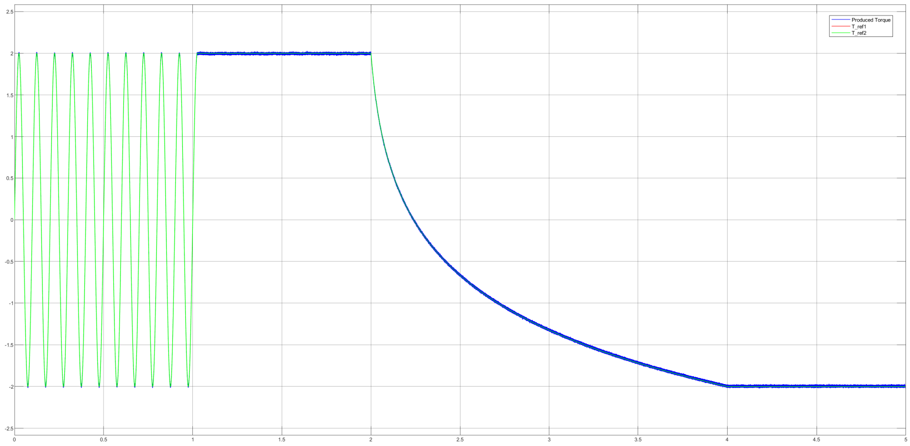


Figure 4.11: AS-MPC with modelled cogging torque and no compensation term.

4.4 Hardware Test

The same trajectory that was used in the simulations, was also used to test the control system on the hardware. The control system that was tested on hardware was the LQR. However, due to limitations in the maximum current for the sensor, it was not possible to control the system to the 2 Nm as in the simulations. The trajectory was thus scaled to be between -0.5 to 0.5 Nm. Another limitation was that it was not possible to communicate with both the microcontrollers at the same time. Therefore, the test was done on one three-phase system. Also, the axle was locked during the test due to the reasons discussed in Section 2.6. The result can be seen in Figure 4.12

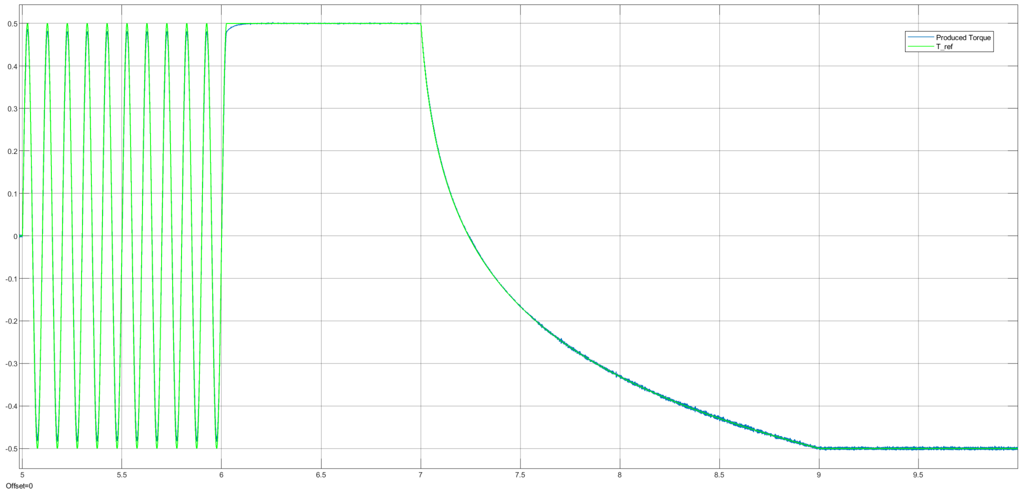


Figure 4.12: The produced torque from the real PMSM and the reference.

The control system behaved as expected and performed well on the hardware. The project also tested uploading the code to the hardware so that both microcontrollers could be run simultaneously. However, as mentioned before, this could not be plotted.

As the code is uploaded, it is difficult to synchronise the trajectory reference between the controllers. Therefore, to test the joint system, a control loop that requested more torque depending on the angle of the rotor was made. This was possible to load to both systems and could be tested without a connection to a computer and between the microcontrollers. The generated torque had a low torque ripple, and the systems worked well together.

5

Discussion

The purpose of this thesis was to create torque tracking controllers for a six-phase PMSM. From the results, it is possible to conclude that all the control systems performed well. The control systems had minimal differences in the response time; the MPCs were a bit faster than the GSCs/LQRs; this was because the GSC and LQR were limited by the SVPWM. Otherwise, they would have performed equally well, as all of the systems were limited by the number of inverter configurations. However, the control systems have some differences in how well they compensate for noisy sensor readings, and also their potential if the sensor setup was more accurate. The MPCs have the highest potential while the LQR and GSC are limited by the SVPWM. This can be seen by comparing the ideal case in Table 4.2 with the filtered case in Table 4.6. Furthermore, the results showed that the simplified models performed as well as the complex.

The project also evaluated the performance gain of different types of filters. It showed that there was a small performance gain by implementing angle filters. The small gain is due to the high precision sensor used for angle measurement, but also that the model used does not sufficiently capture the axle dynamics. Furthermore, a current filter was implemented, which resulted in large reductions in torque ripple. This was mainly due to the fact that the project used a current sensor with low precision but also that it used the motor model equations, which are very accurate. Lastly, velocity filters and sensors were tested, but did not result in any performance gain. The low-pass filter used induces a time shift, which cancels out any benefits of using such a filter in this setup. Testing with the added sensor also revealed that the velocity was not so important for reducing torque ripple, at least in the tested areas of the state space. The final system containing all filters showed that there was a minimal performance difference between the controllers as the current sensor is the limiting factor.

The hardware tests reinforced the validity of the simulations, as the ability to track the torque reference in Figure 4.12 is very similar to the simulated behaviour in Figure A.11. The differences can be explained by a higher sampling frequency in the simulation and the necessity to limit the output torque to -0.5 to 0.5 Nm due to hardware limitations.

As mentioned, the purpose of this project was to create torque controllers and minimising torque ripple is vital for the performance of such controllers. During the project, a number of factors to reduce the torque ripple even further have been identified. Excluding motor geometry, the factors identified are:

1. The inductance of the motor windings
2. The switching frequency of the inverter
3. The accuracy of the current sensor
4. Type of inverter

For an SPMSM the torque ripple is solely dependent on the q-current ripple. Thus, to minimise torque ripple, the derivative of the q-current should be small. Observing (3.14) and (3.16) a larger inductance will reduce the ripple in the q-current. But increasing the inductance also comes at the cost of a less responsive motor, with a larger power loss in the magnetic field. However, increasing the inductance might be desirable depending on the importance of low torque ripple. Increasing the inductance is essentially equivalent to giving the physical system low-pass filtering qualities.

Instead of increasing the inductance the switching frequency of the inverter can be increased. This is essentially the opposite of the method above and focuses on increasing the switching frequency so that the step size is lowered instead of lowering the derivative of the q-current. This will make the inherent inductance in the stator windings enough to filter some of the switching. Although very effective in simulation this method is ultimately limited by available hardware. The inverter used in this project has a maximum switching frequency of 200 kHz. To buy MOSFETs with even higher switching frequencies would be costly.

The two factors above relate to how fast the inverter needs to switch to reduce the torque ripple. The following factors instead relate to how the inverter switches. The first factor among these is the accuracy of the current sensor. All controllers in this project make a decision on the inverter combination given some control error in the current. This procedure needs an accurate current feedback measurement. If the switching frequency and inductance are increased, the performance might eventually stagnate due to the noise in the sensors, as can be observed from this project. The different combinations that the motor can choose have then become too small for the given sensor noise, and the algorithm begins to choose the wrong inverter combinations. To remedy this behaviour, filters such as the ones implemented in this project can be created, which introduce some computational costs. The sensor can also be improved, however yet again this is cost and hardware limited.

Finally, although they are not commercially common the torque ripple can be improved with more advanced inverters. The standard two-level inverter switches the phase voltage between $V_{DC}/2$ and $-V_{DC}/2$. Increasing the number of MOSFETs in the inverter makes switching between more distinct voltage levels possible. This makes the input to the motor smoother and can thus decrease the ripple in the currents.

5.1 Sustainability aspects

To use electric machines instead of combustion engines provides an opportunity to decrease carbon emissions. If the electricity used is sourced in a sustainable way, such as wind or solar, the remaining emissions are related to the manufacturing process. Thus, investigating and further broadening the knowledge of electric motors will help electrification. Furthermore, a better control system with minimal noise in output will decrease the system power loss while also leading to a better product.

6

Conclusion

From the gathered results it is clear to see that the simplified model controllers and the complex controllers perform similarly in all cases. Thus adding this complexity to the controller is not needed and will only limit the controller when implemented on real hardware. The complex model controllers also necessitate information about the second system and thus redundancy is lost. Therefore the added complexity and loss of redundancy of the true model makes the simplified model the best option for model-based controllers.

Comparing the controllers, the MPC approach has the potential to reach a lower torque ripple than the LQR and GSC as they are limited by the SVPWM algorithm. However, the system has a noisy current reading which in turn becomes the limiting factor of the MPC algorithm. Thereby, the differences between the three systems are small. The project, therefore, concludes that the S-LQR was the best choice given the existing sensor configuration. This solution was implemented on the hardware and performed similarly to the simulations. However, if a better sensor setup was used the SS-MPC might become viable even though the increased computational need.

6.1 Future Work

Firstly, doing a similar study but instead comparing the control systems purely on hardware might lead to interesting results. Testing the MPC strategies with current filtering is especially important to see if the small torque ripple simulated in this project can be achieved on a real motor.

Due to the redundancy qualities of dual three-phase systems, an interesting task would be to investigate torque control during failure and how the failure state and following error handling affect the torque tracking ability.

Investigate if adding an approximated cogging compensation would result in a performance benefit. As the cogging torque is a sum of sinusoidal terms, it is possible to use backpropagation to estimate the torque with the sine as the activation function.

The project found that a variant of the six-step commutation controller could work as a PWM strategy. It worked by finding the closest voltage vector to V_{ref}. Then, it predicted the next current by using the discrete PMSM equations (3.74) - (3.74) for both the zero vector and the closest voltage vector. The currents were then compared, and the one closest to the reference was taken. However, due to time limitations, this was only tested once.

A possible way to improve the angle estimate is to use sensorless control strategies. This estimate can then be combined with the angle sensor reading in a Kalman filter. This might improve the validity of the Park transforms used throughout the feedback loop and thus also improve the torque ripple effect.

Lastly, as the torque ripple is largely caused by the inverter switching its state, a comparative study using different level inverters and their impact on the torque ripple might lead to improved results.

Bibliography

[1] Wanwan Ren, You Wu, and Ronghua Du. “A Vector Control System of PMSM with the Assistance of Fuzzy PID Controller”. In: *2020 39th Chinese Control Conference (CCC)*. 2020, pp. 2205–2210. DOI: `10.23919/CCC50068.2020.9189325`.

[2] Kai Lin, Ping Lin, and Bin Yang. “Two-Degree-of-Freedom Control with Linear Extended State Observer for PMSM Drive System”. In: *2022 41st Chinese Control Conference (CCC)*. 2022, pp. 2779–2784. DOI: `10.23919/CCC55666.2022.9901541`.

[3] Wenshan Wei et al. “Fault Tolerant Control for PMSM Position Measurement Failure in CMG”. In: *2023 IEEE 18th Conference on Industrial Electronics and Applications (ICIEA)*. 2023, pp. 646–650. DOI: `10.1109/ICIEA58696.2023.10241848`.

[4] Qian Wan, Huiming Wang, and Xia Sun. “Position Tracking Control for PMSM System Based on Continuous Sliding Mode Control and Sliding Mode Observer”. In: *2022 41st Chinese Control Conference (CCC)*. 2022, pp. 2917–2922. DOI: `10.23919/CCC55666.2022.9902155`.

[5] Michał Brasel. “A gain-scheduled multivariable LQR controller for permanent magnet synchronous motor”. In: *2014 19th International Conference on Methods and Models in Automation and Robotics (MMAR)*. 2014, pp. 722–725. DOI: `10.1109/MMAR.2014.6957443`.

[6] Wang Yong-Bin et al. “A MPC Method Based on the Oval Invariant Set for PMSM Speed Control System”. In: *2016 International Symposium on Computer, Consumer and Control (IS3C)*. 2016, pp. 575–578. DOI: `10.1109/IS3C.2016.148`.

[7] Liu Sheng, Guo Xiaojie, and Zhang Lanyong. “Robust Adaptive Backstepping Sliding Mode Control for Six-Phase Permanent Magnet Synchronous Motor Using Recurrent Wavelet Fuzzy Neural Network”. In: *IEEE Access* 5 (2017), pp. 14502–14515. DOI: `10.1109/ACCESS.2017.2721459`.

[8] Hanying Gao et al. “Fault-tolerant control strategy of six-phase permanent magnet synchronous motor based on deadbeat current prediction”. In: *PLOS ONE* 18.7 (July 2023), pp. 1–29. DOI: `10.1371/journal.pone.0288728`. URL: `https://doi.org/10.1371/journal.pone.0288728`.

[9] Sebastián Rojas et al. "Torque ripple modeling of a permanent magnet synchronous motor". In: *2010 IEEE International Conference on Industrial Technology*. 2010, pp. 433–438. DOI: `10.1109/ICIT.2010.5472761`.

[10] S.-H. Kim. *Electric Motor Control: DC, AC, and BLDC Motors*. Joe Hayton, 2017.

[11] Ziqiang Zhu et al. "Advances in Dual-Three-Phase Permanent Magnet Synchronous Machines and Control Techniques". In: *Energies* 14.22 (2021). ISSN: 1996-1073. DOI: `10.3390/en14227508`. URL: `https://www.mdpi.com/1996-1073/14/22/7508`.

[12] Yashan Hu, Z. Q. Zhu, and Milijana Odavic. "Comparison of Two-Individual Current Control and Vector Space Decomposition Control for Dual Three-Phase PMSM". In: *IEEE Transactions on Industry Applications* 53.5 (2017), pp. 4483–4492. DOI: `10.1109/TIA.2017.2703682`.

[13] Xiaolin KUANG et al. "Research on a six-phase permanent magnet synchronous motor system at dual-redundant and fault tolerant modes in aviation application". In: *Chinese Journal of Aeronautics* 30.4 (2017), pp. 1548–1560. ISSN: 1000-9361. DOI: `https://doi.org/10.1016/j.cja.2017.05.001`. URL: `https://www.sciencedirect.com/science/article/pii/S1000936117301048`.

[14] K. Mohith, S. Patilkulkarni, and N. Kollaparti. "Comparative Analysis of Different Control Techniques for Six-Phase PMSM as an Application to HEV". In: *2017 IEEE Transportation Electrification Conference and Expo, Asia-Pacific (ITEC Asia-Pacific)*. 2019. DOI: `10.1007/978-981-16-0336-5_9`.

[15] Fatima AZAIZ and Ahmed Azaiz. "Design of a Robust Control System for the PMS Motor". In: (Jan. 2008).

[16] T. Glad and L. Ljung. *Control Theory - Multivariable and Nonlinear Methods*. Taylor & Francis, 2000.

[17] M.R. Arahal et al. "Multi-phase current control using finite-state model-predictive control". In: *Control Engineering Practice* 17.5 (2009), pp. 579–587. ISSN: 0967-0661. DOI: `https://doi.org/10.1016/j.conengprac.2008.10.005`. URL: `https://www.sciencedirect.com/science/article/pii/S0967066108001858`.

[18] Federico Barrero et al. "An Enhanced Predictive Current Control Method for Asymmetrical Six-Phase Motor Drives". In: *IEEE Transactions on Industrial Electronics* 58.8 (2011), pp. 3242–3252. DOI: `10.1109/TIE.2010.2089943`.

[19] Mario J. Durán et al. "Predictive current control of dual three-phase drives using restrained search techniques and multi level voltage source inverters". In: *2010 IEEE International Symposium on Industrial Electronics*. 2010, pp. 3171–3176. DOI: `10.1109/ISIE.2010.5637556`.

[20] Hamed Masnadi-Shirazi, Alireza Masnadi-Shirazi, and Mohammad-Amir Dastgheib. *A Step by Step Mathematical Derivation and Tutorial on Kalman Filters*. 2019. arXiv: `1910.03558 [stat.OT]`.

[21] Simo Särkkä. *Bayesian Filtering and Smoothing.* Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013.

[22] Radu Bojoi et al. "Dual-Three Phase Induction Machine Drives Controlmdash;A Survey". In: *IEEJ Transactions on Industry Applications* 126.4 (2006), pp. 420–429. DOI: `10.1541/ieejias.126.420`.

[23] Z. Yu, A. Mohammed, and I. Panahi. "A review of three PWM techniques". In: *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041).* Vol. 1. 1997, 257–261 vol.1. DOI: `10.1109/ACC.1997.611797`.

[24] Gabriele Grandi, Giovanni Serra, and Angelo Tani. "Space Vector Modulation of a Six-Phase VSI based on three-phase decomposition". In: *2008 International Symposium on Power Electronics, Electrical Drives, Automation and Motion.* 2008, pp. 674–679. DOI: `10.1109/SPEEDHAM.2008.4581262`.

[25] C. Studer et al. "Study of cogging torque in permanent magnet machines". In: *IAS '97. Conference Record of the 1997 IEEE Industry Applications Conference Thirty-Second IAS Annual Meeting.* Vol. 1. 1997, 42–49 vol.1. DOI: `10.1109/IAS.1997.643006`.

[26] Tudorache Tiberiu et al. "Improved Mathematical Model of PMSM Taking Into Account Cogging Torque Oscillations". In: *Advances in Electrical and Computer Engineering* 12 (Aug. 2012), pp. 59–64. DOI: `10.4316/aece.2012.03009`.

A

Appendix: Complete Simulation Results

The following appendix aims at completing the results by displaying all obtained torque trajectories. This can give some insight into how the different combinations of controllers, sensors and filters operate but was left out of the result due to brevity.

A.1 Step Responses

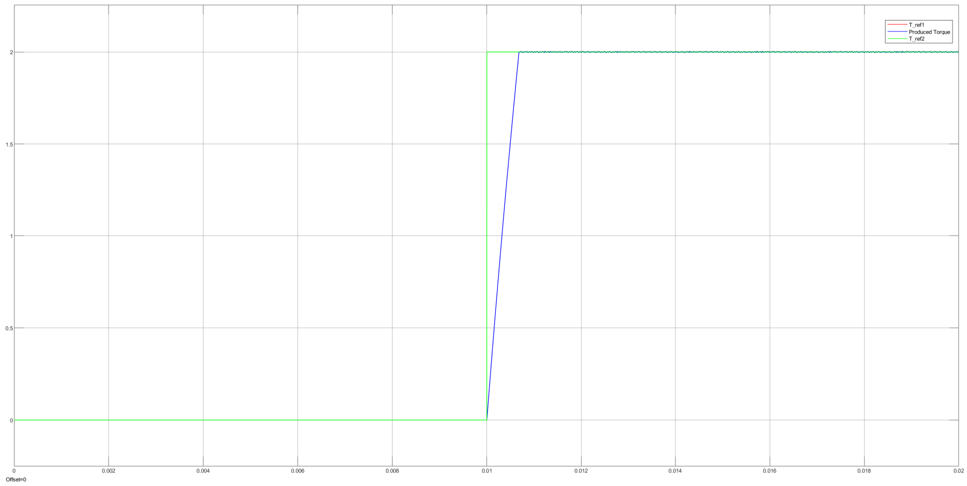


Figure A.1: AC-MPC

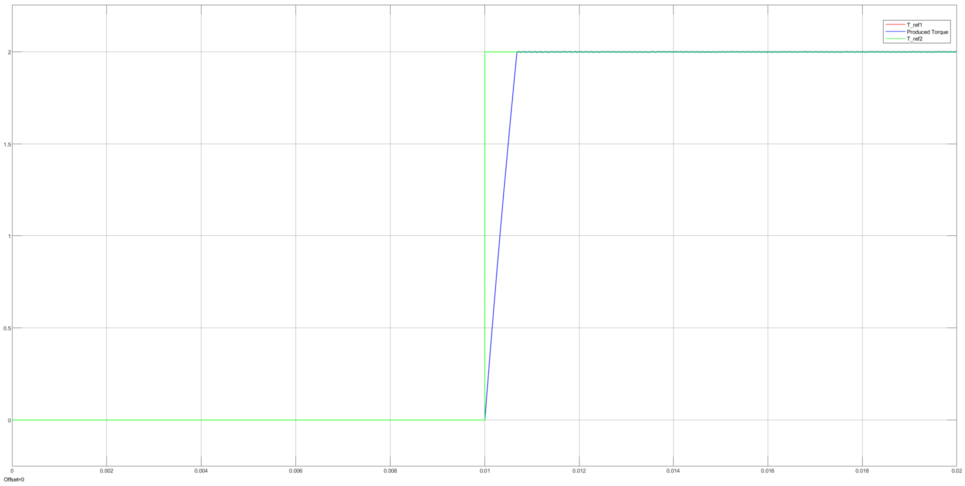


Figure A.2: AS-MPC

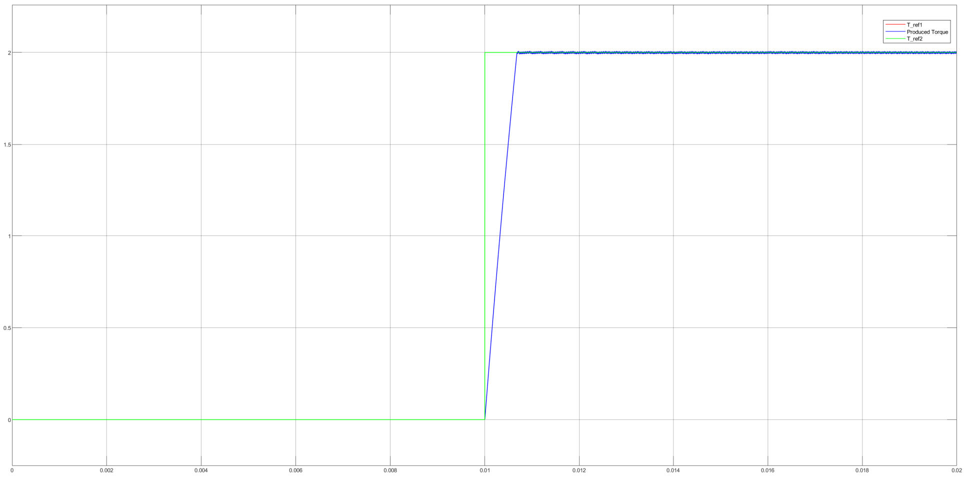


Figure A.3: SS-MPC

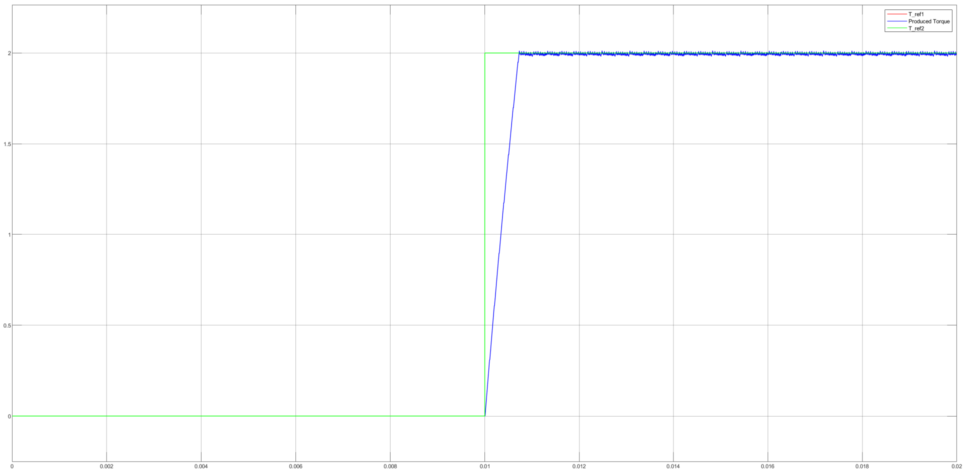


Figure A.4: S-LQR

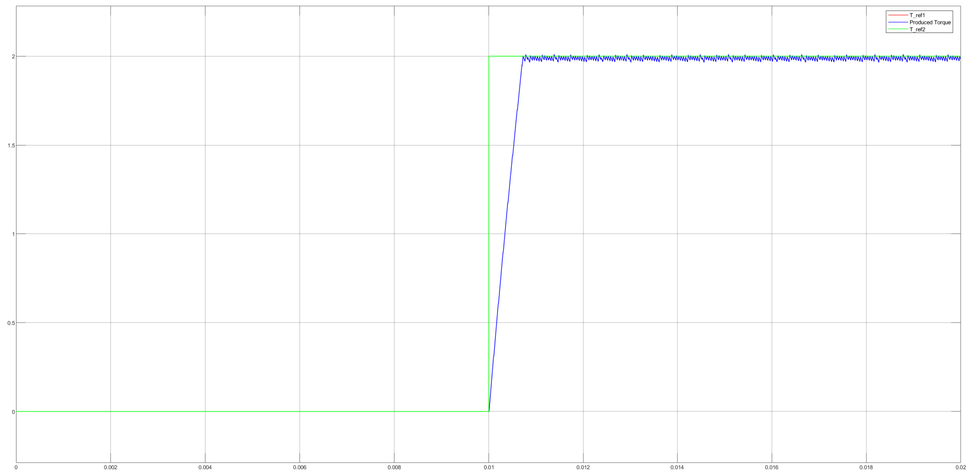


Figure A.5: C-LQR

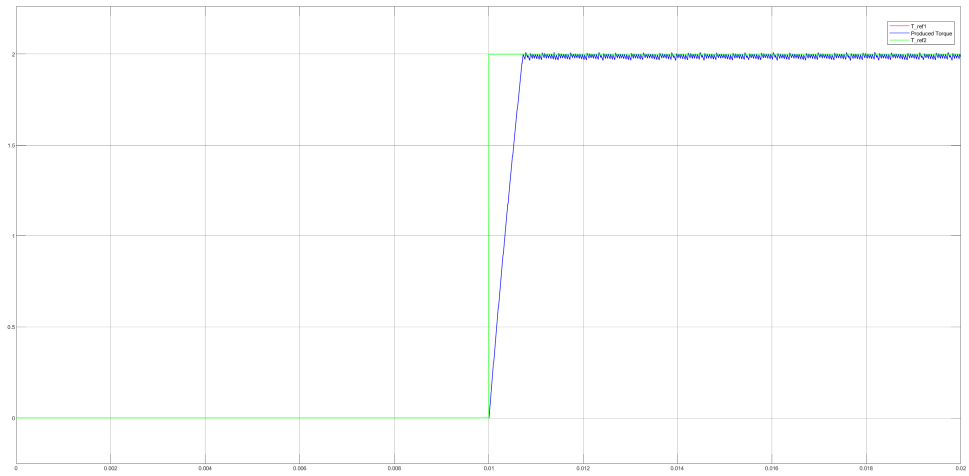


Figure A.6: S-GSC

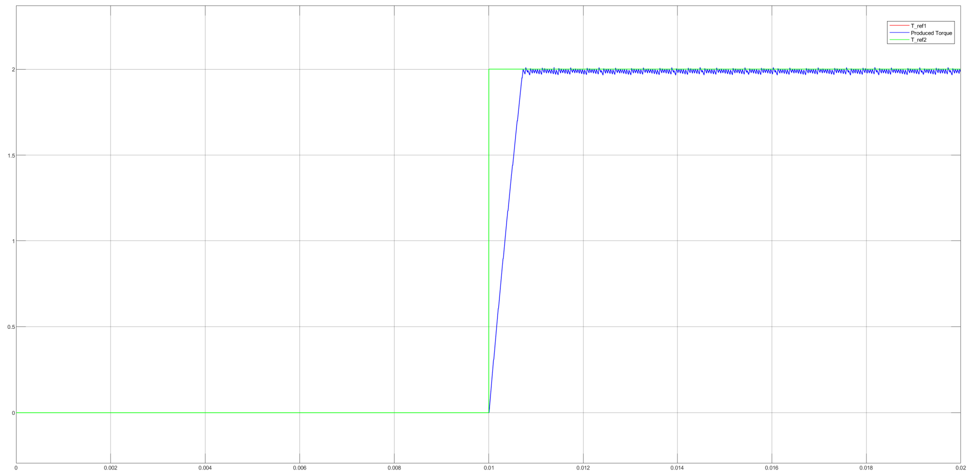


Figure A.7: C-GSC

A.2 Torque Trajectory with No Filter

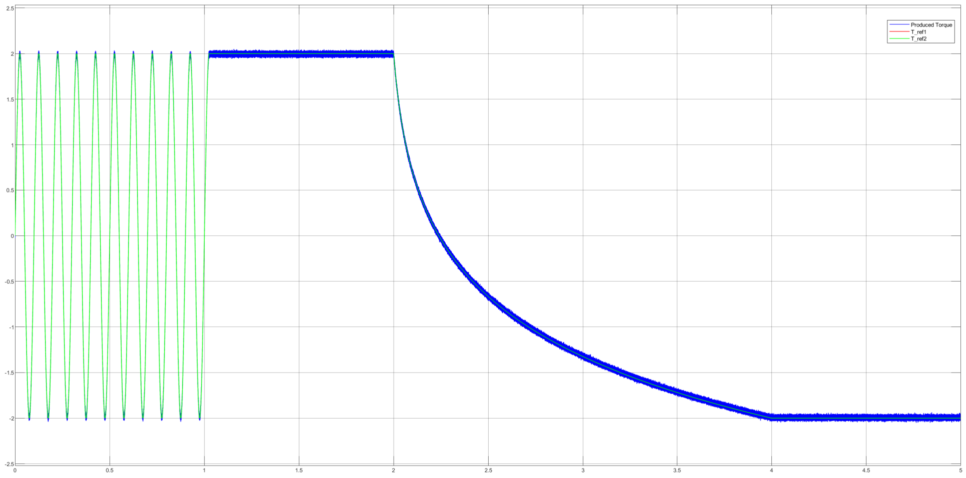


Figure A.8: AC-MPC

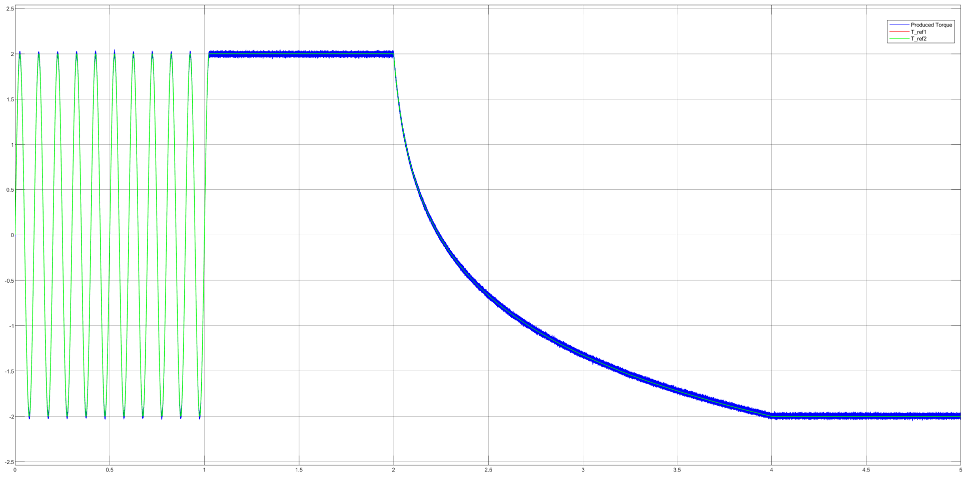


Figure A.9: AS-MPC

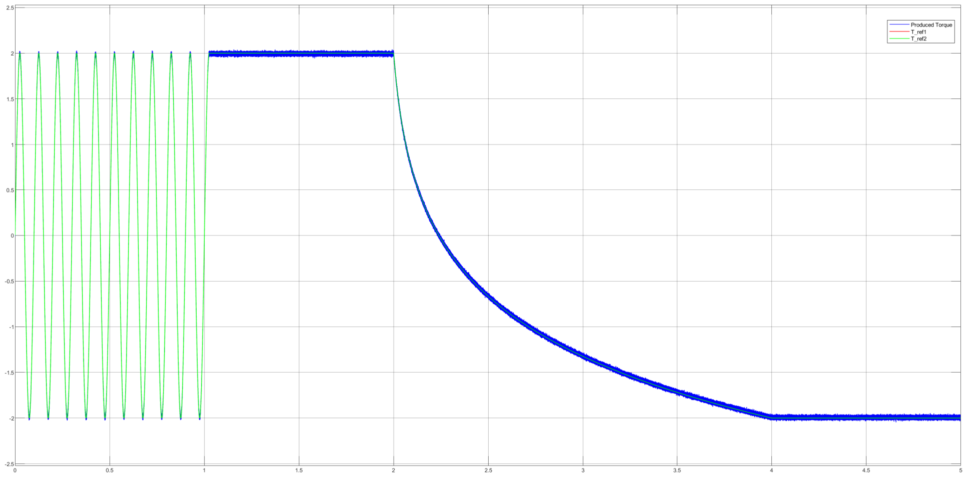


Figure A.10: SS-MPC

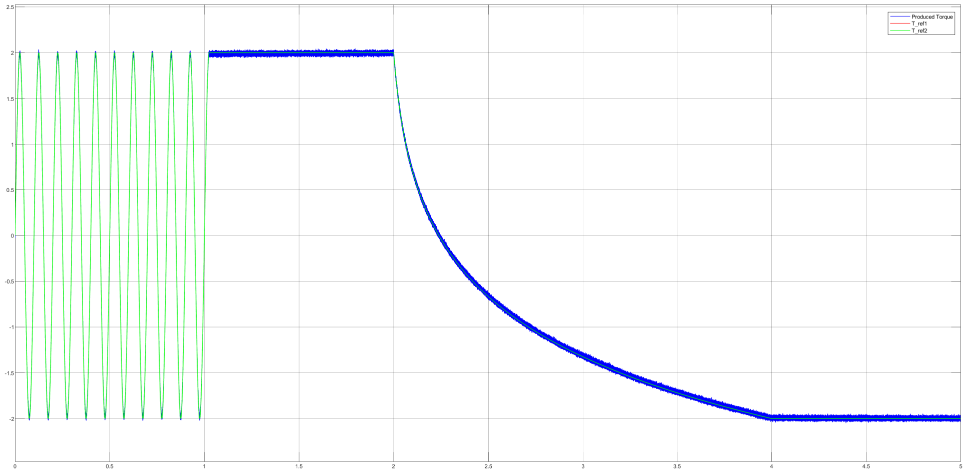


Figure A.11: S-LQR

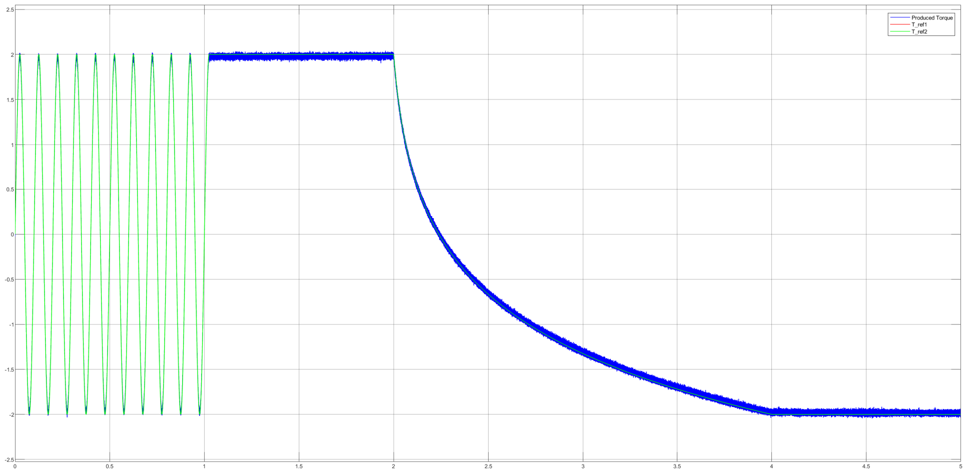


Figure A.12: C-LQR

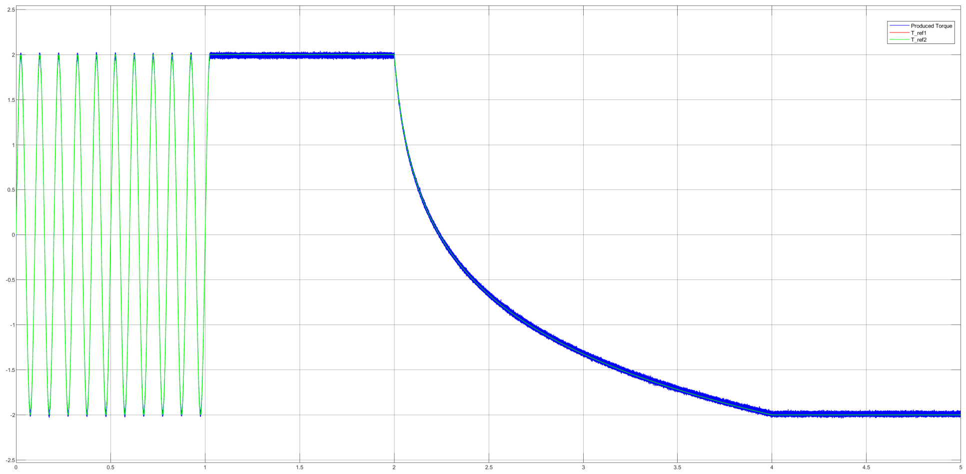


Figure A.13: S-GSC

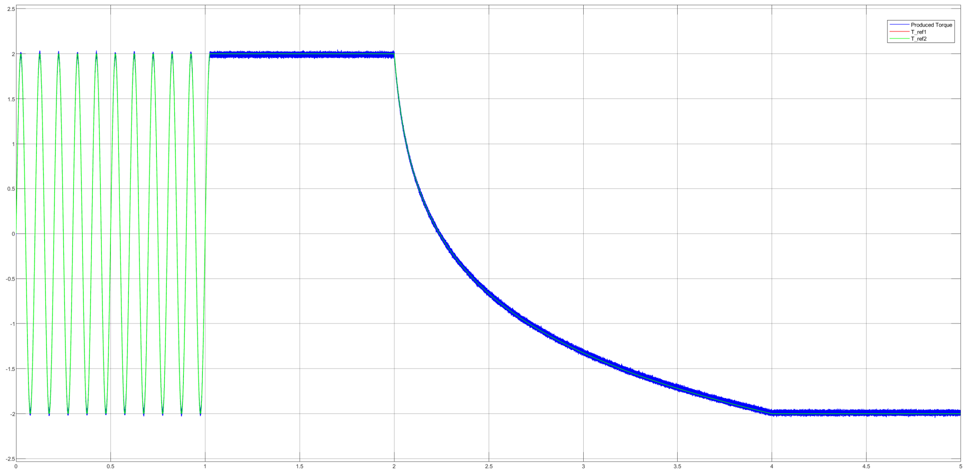


Figure A.14: C-GSC

A.3 Control with Angle Filter

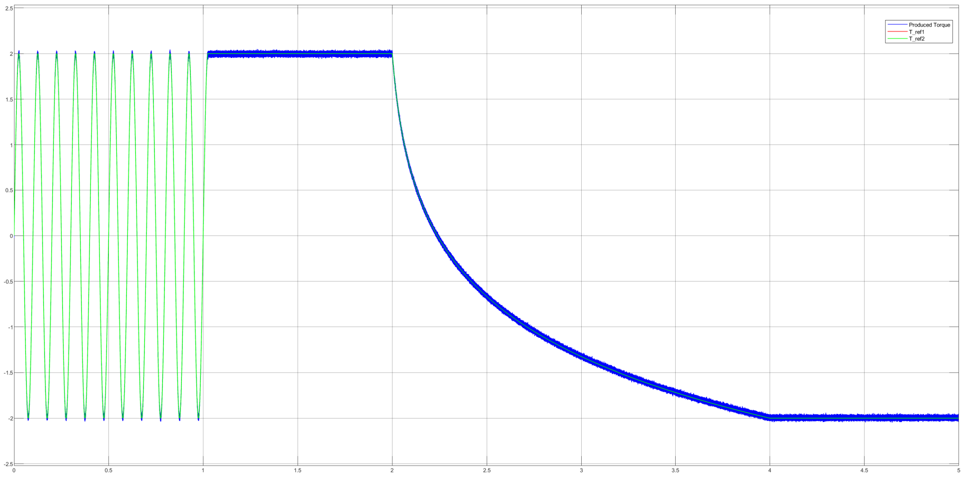


Figure A.15: AC-MPC

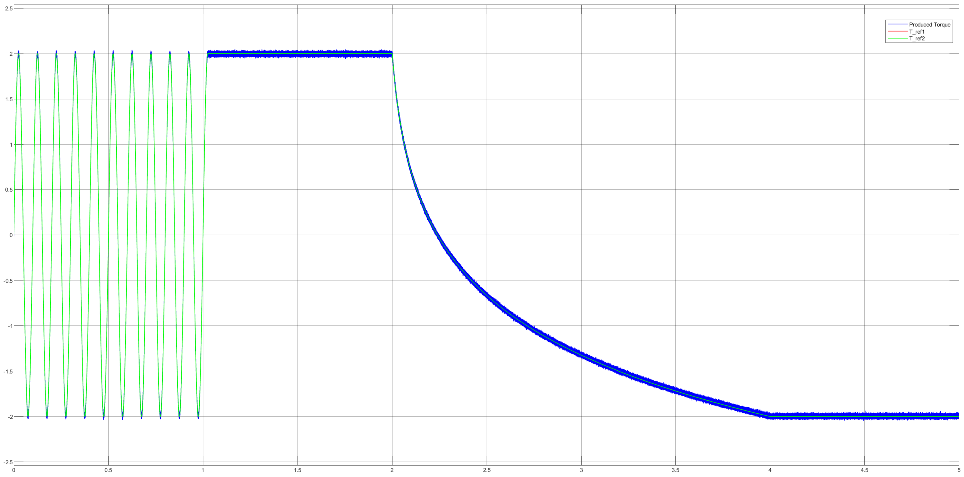


Figure A.16: AS-MPC

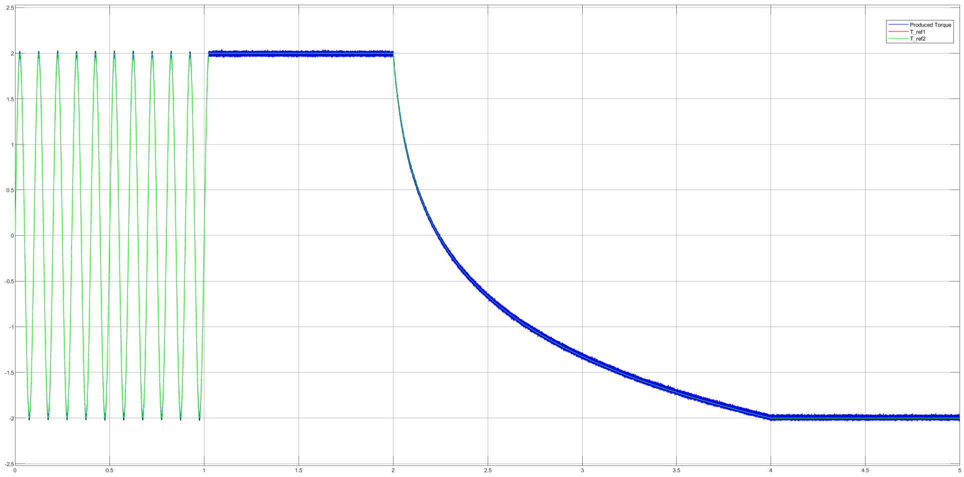


Figure A.17: SS-MPC

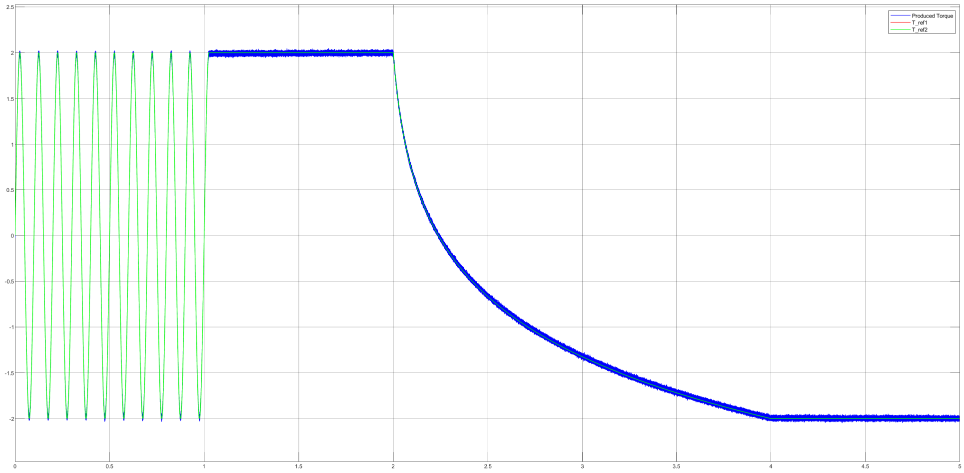


Figure A.18: S-LQR

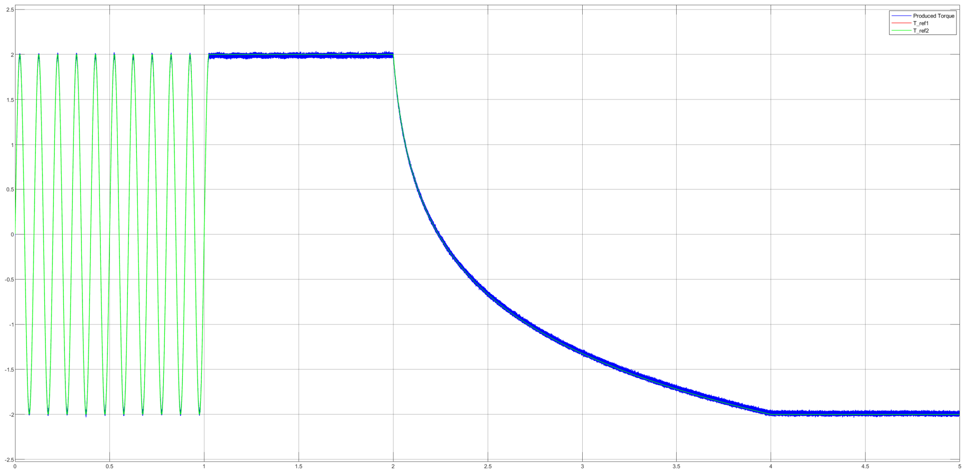


Figure A.19: C-LQR

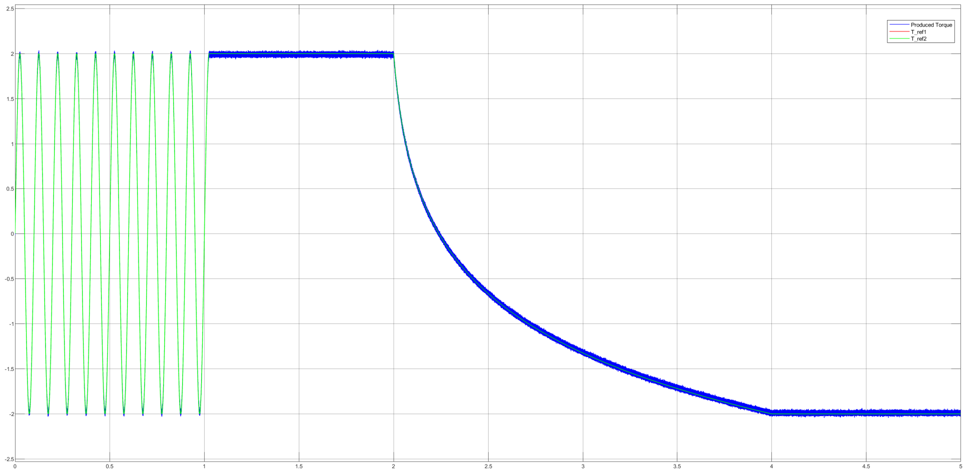


Figure A.20: S-GSC

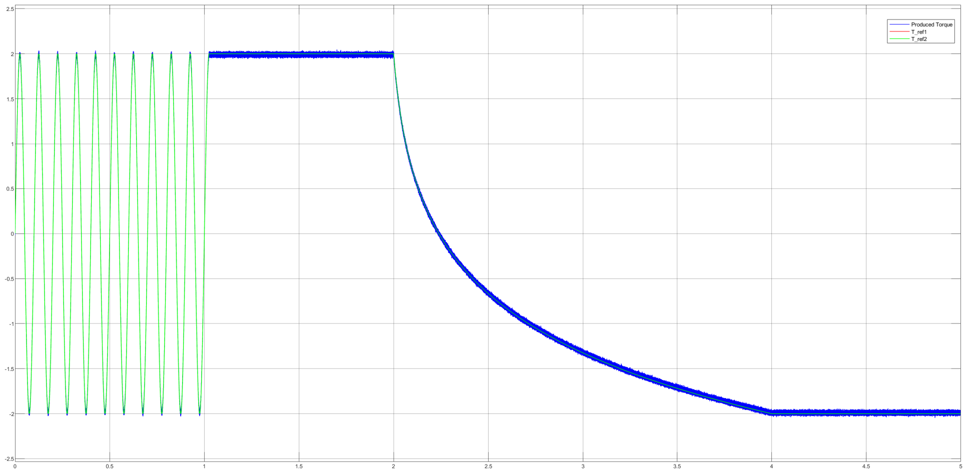


Figure A.21: C-GSC

A.4 Control with Angle and Angular Velocity Filter

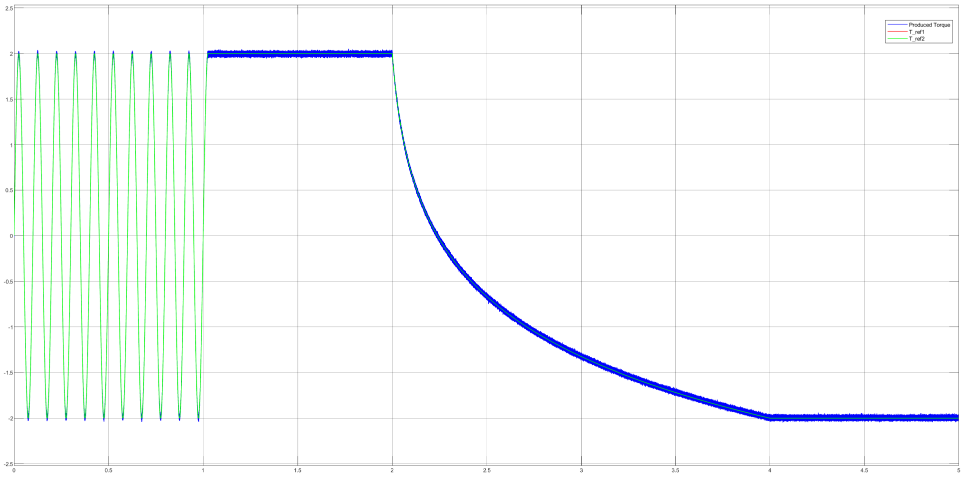


Figure A.22: AC-MPC

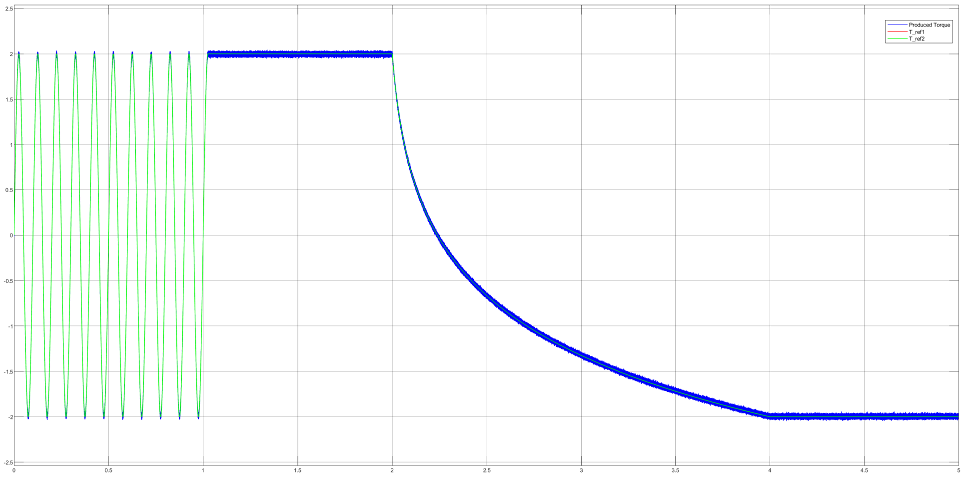


Figure A.23: AS-MPC

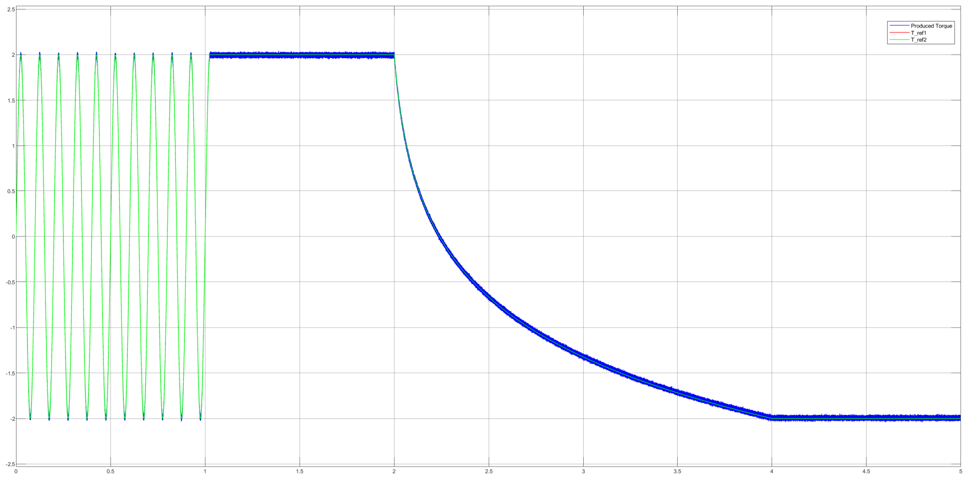


Figure A.24: SS-MPC

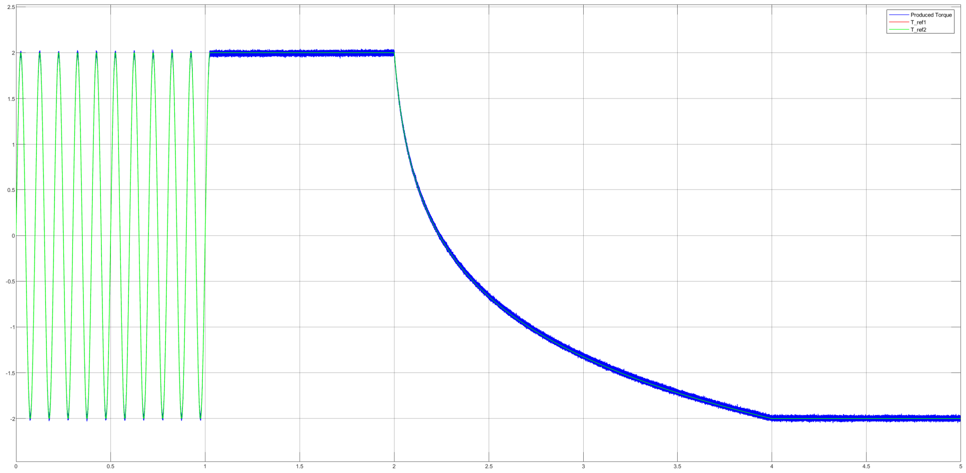


Figure A.25: S-LQR

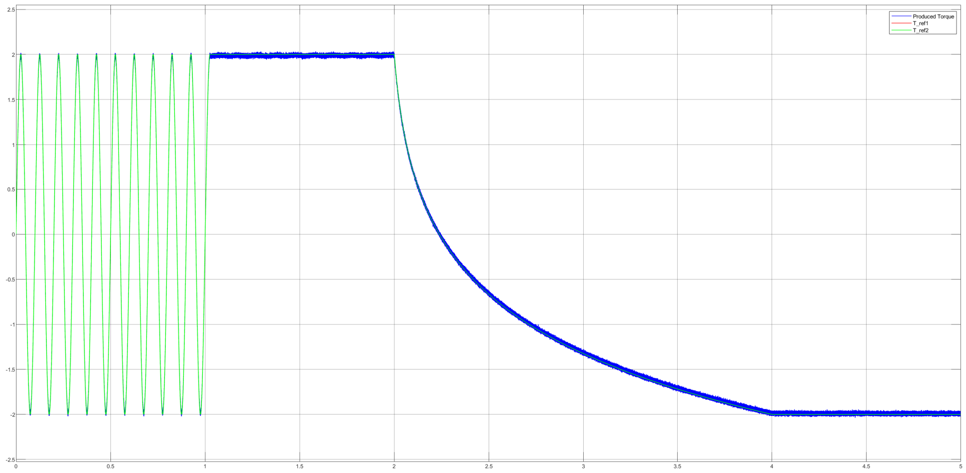


Figure A.26: C-LQR

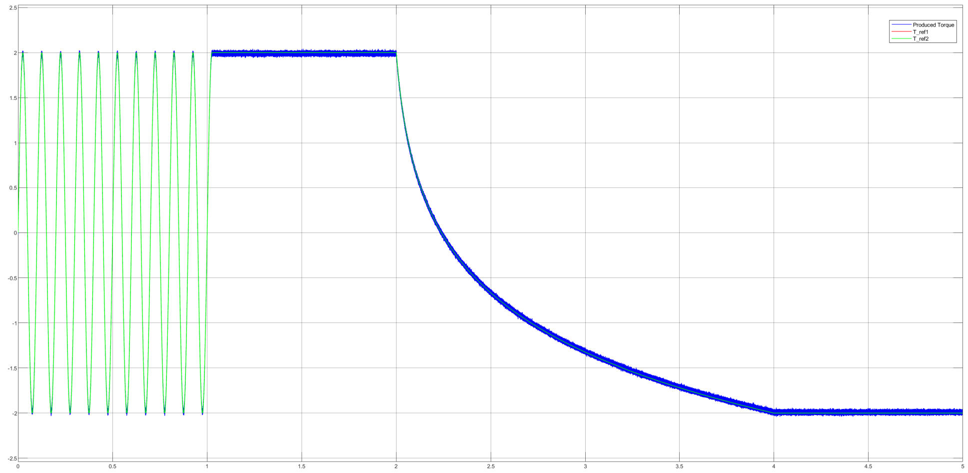


Figure A.27: S-GSC

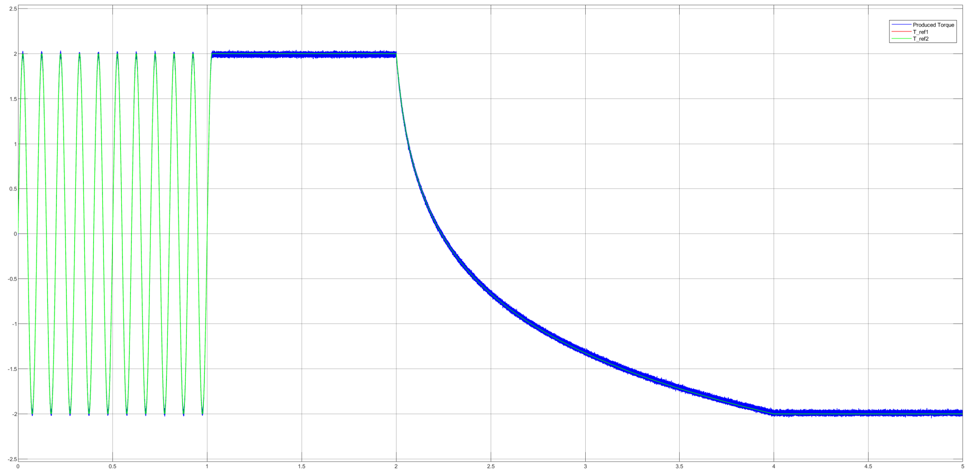


Figure A.28: C-GSC

A.5 Control with Angle, Angular Velocity and Current Filter

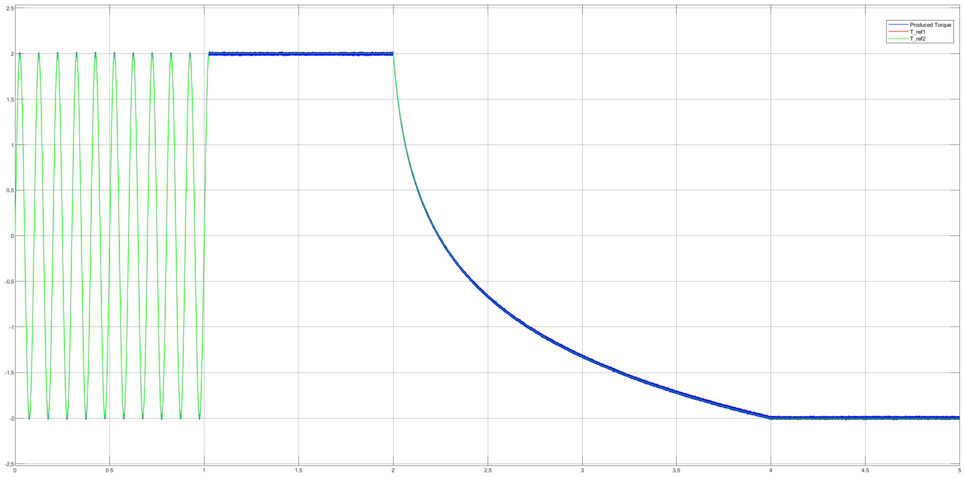


Figure A.29: AC-MPC

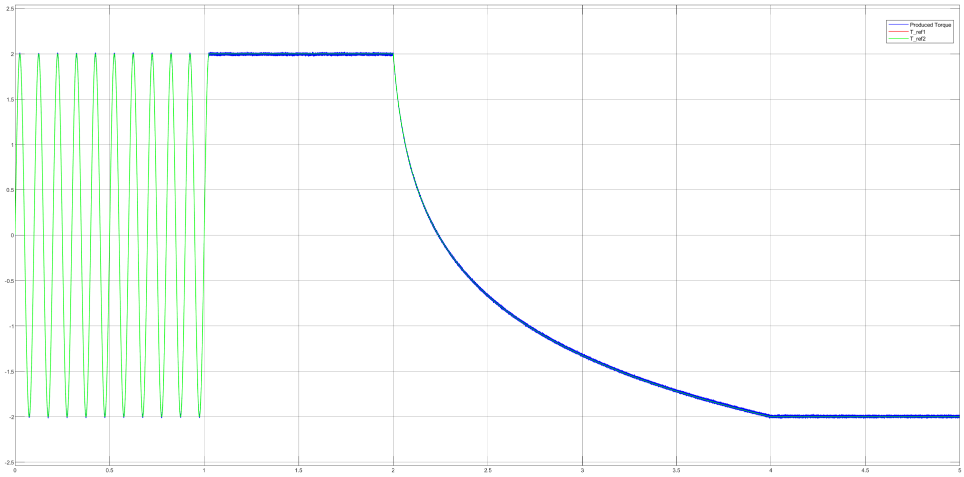


Figure A.30: AS-MPC

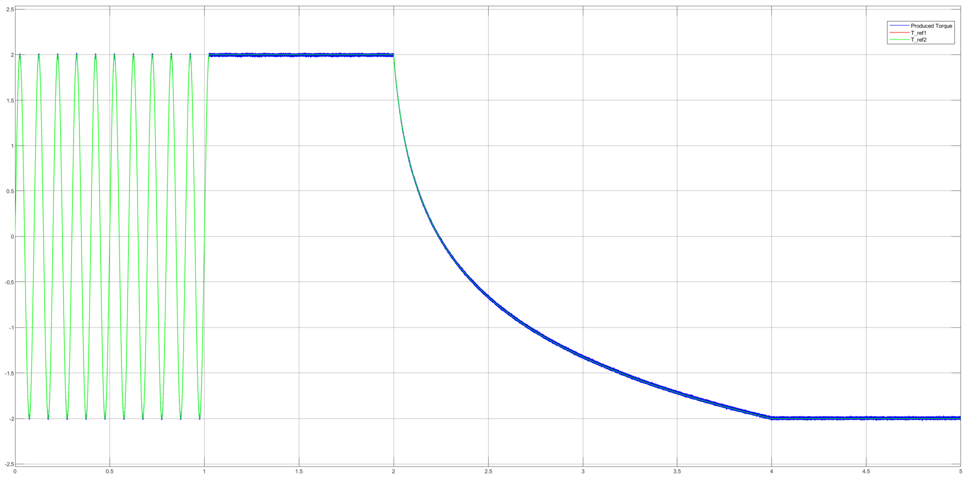


Figure A.31: SS-MPC

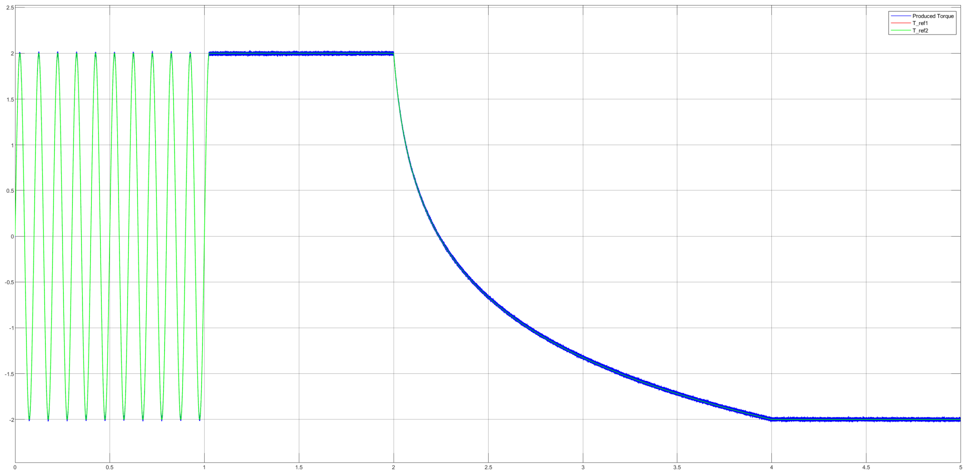


Figure A.32: S-LQR

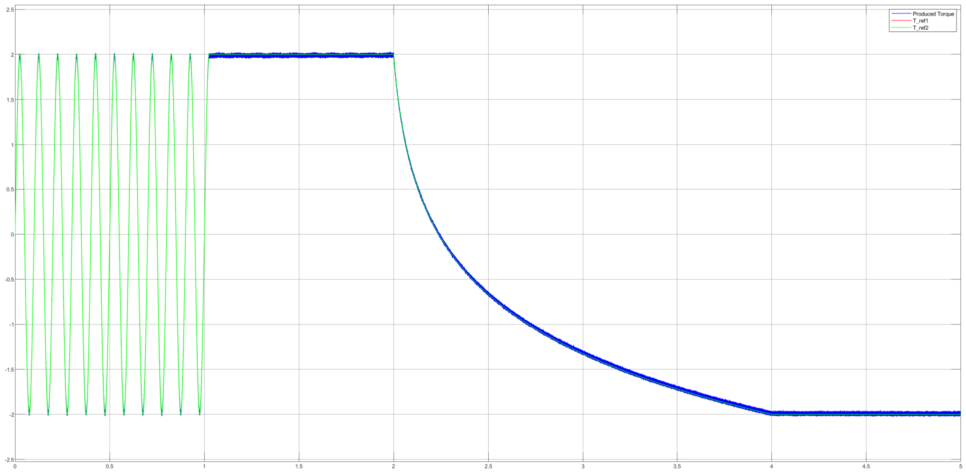


Figure A.33: C-LQR

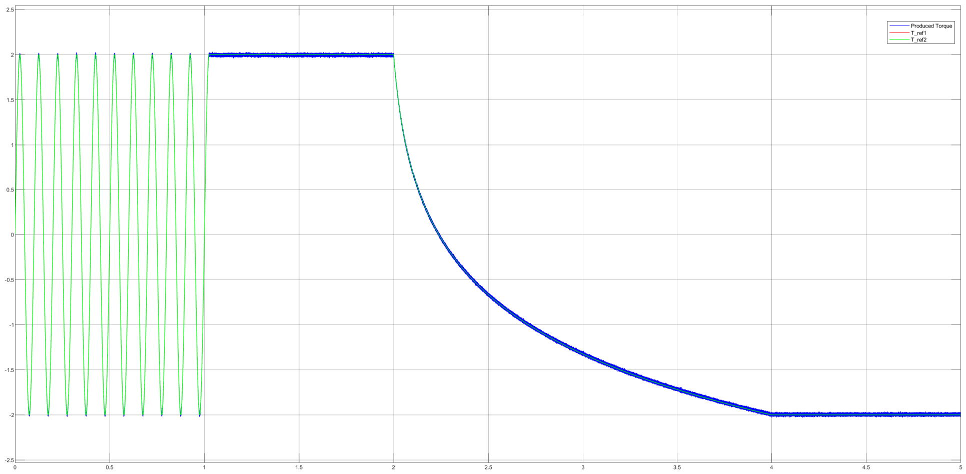


Figure A.34: S-GSC

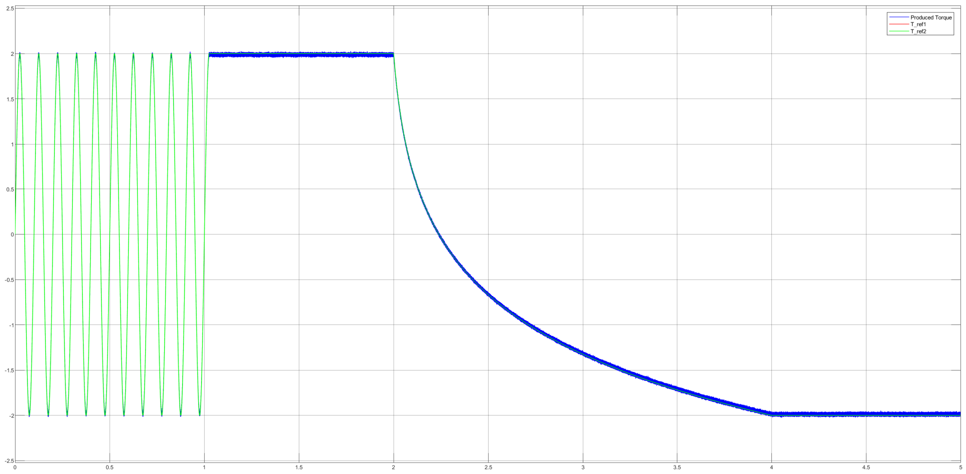


Figure A.35: C-GSC

A.6 Control with Current Filter

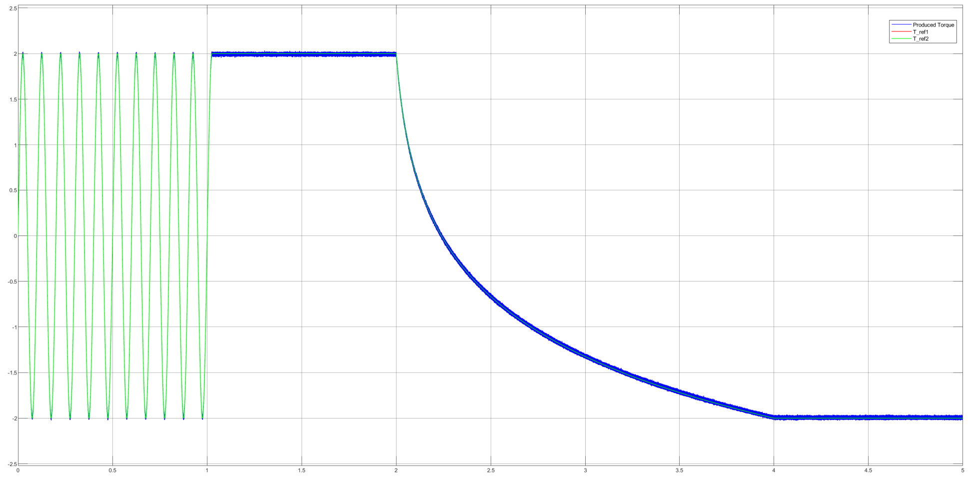


Figure A.36: AC-MPC

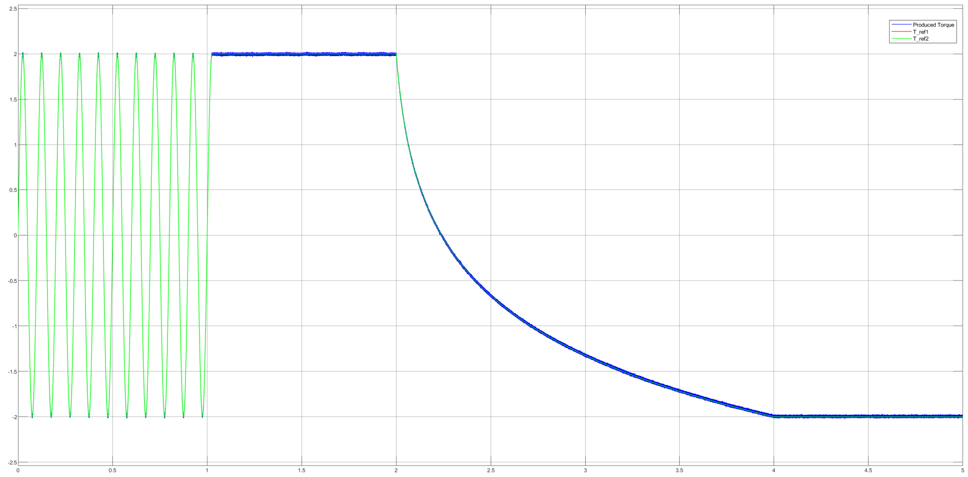


Figure A.37: AS-MPC

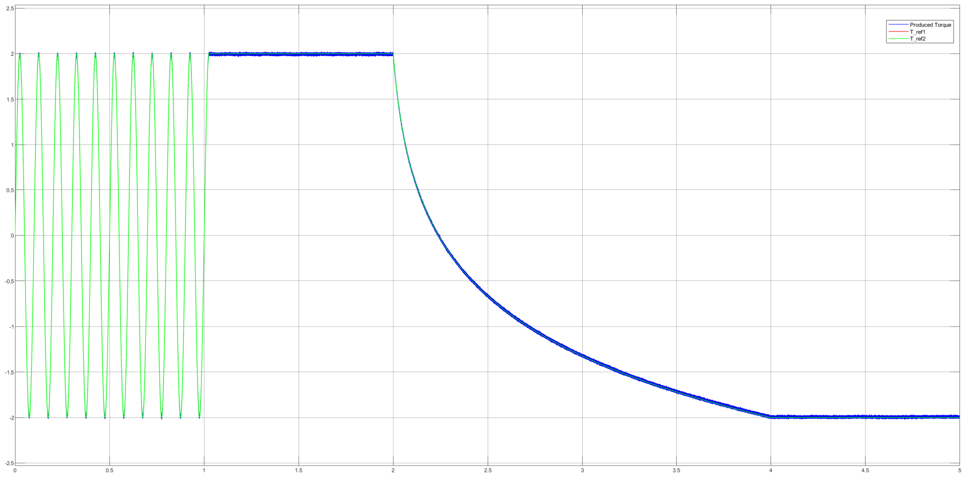


Figure A.38: SS-MPC

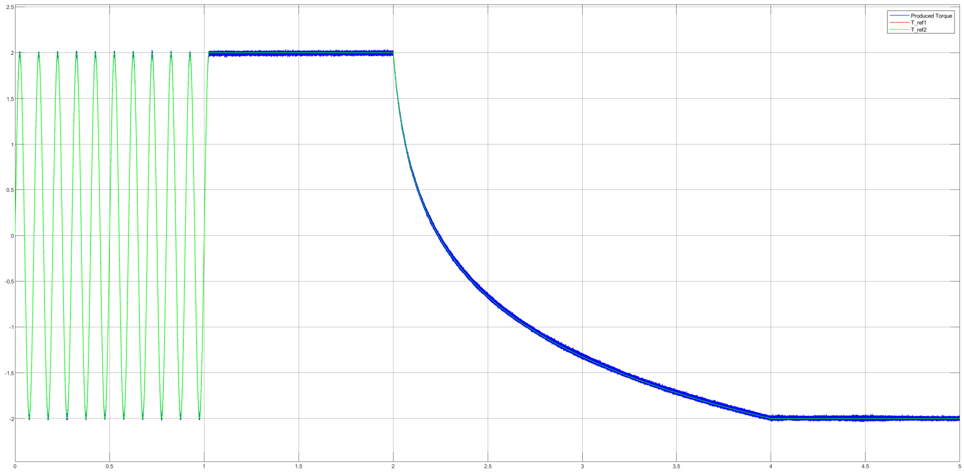


Figure A.39: S-LQR

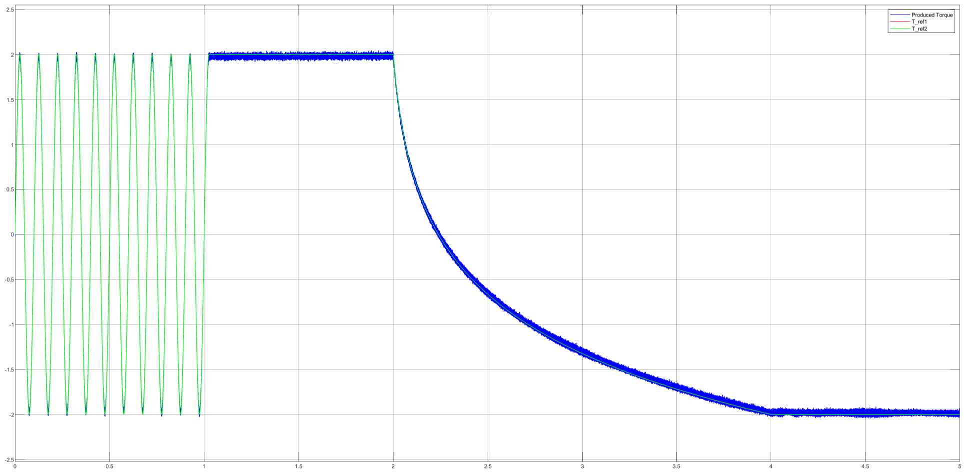


Figure A.40: C-LQR

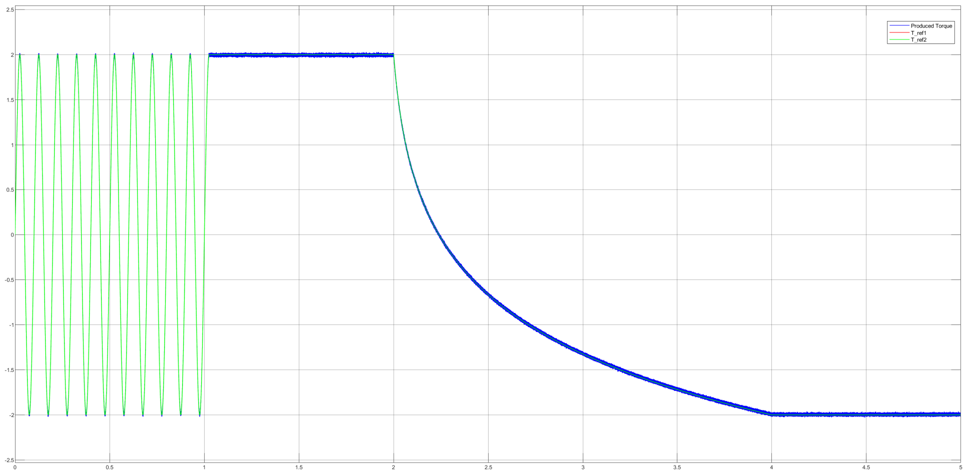


Figure A.41: S-GSC

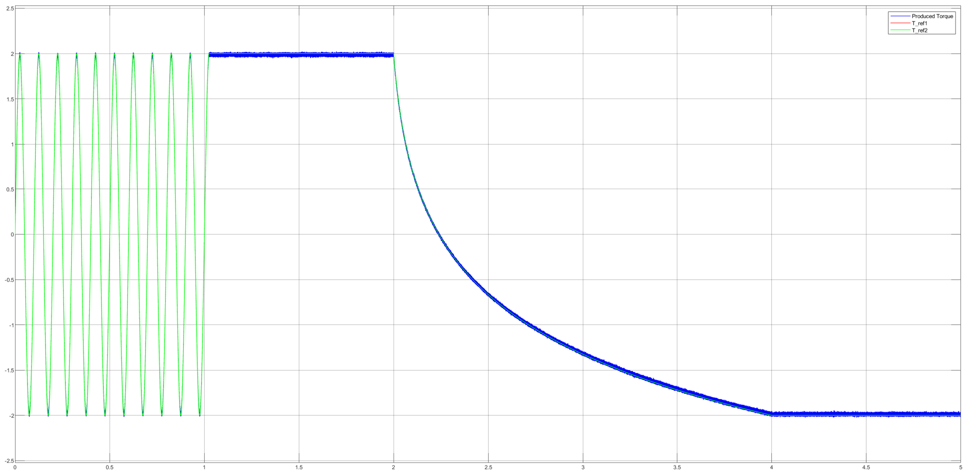


Figure A.42: C-GSC

A.7 Control with Added Angular Velocity Sensor

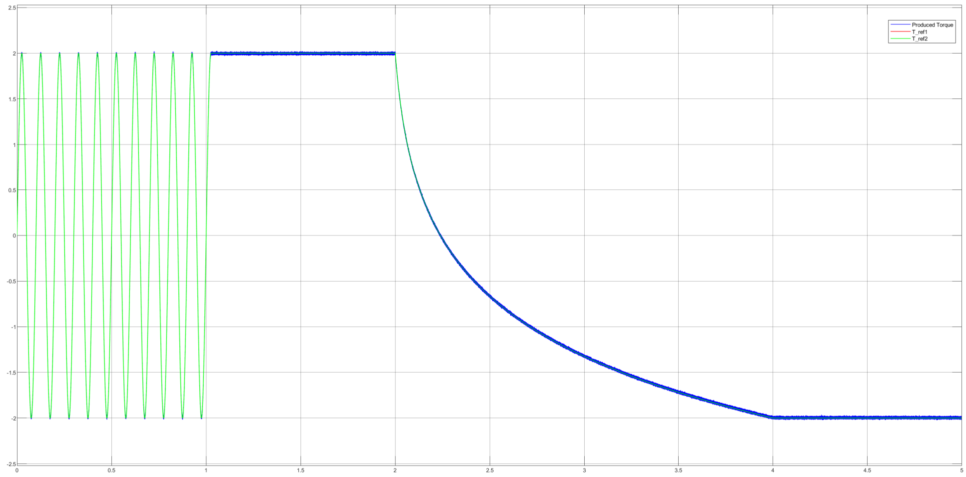


Figure A.43: AC-MPC

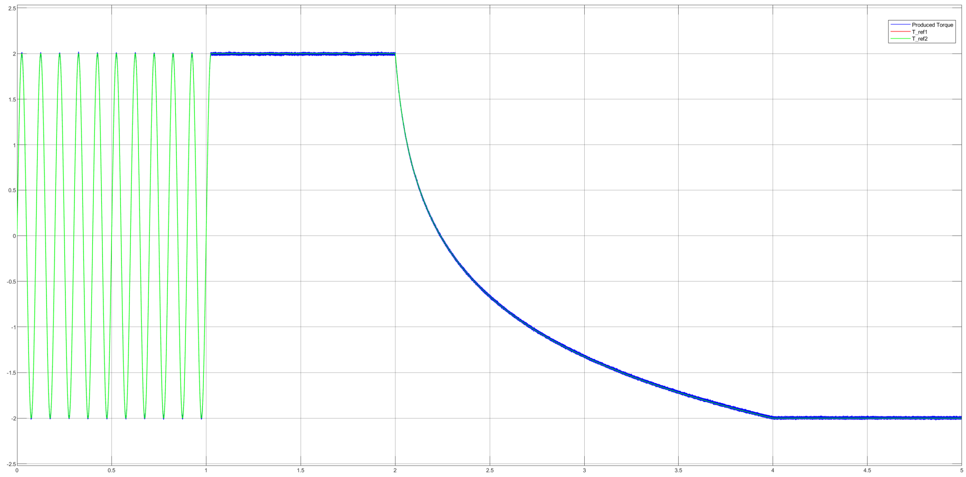


Figure A.44: AS-MPC

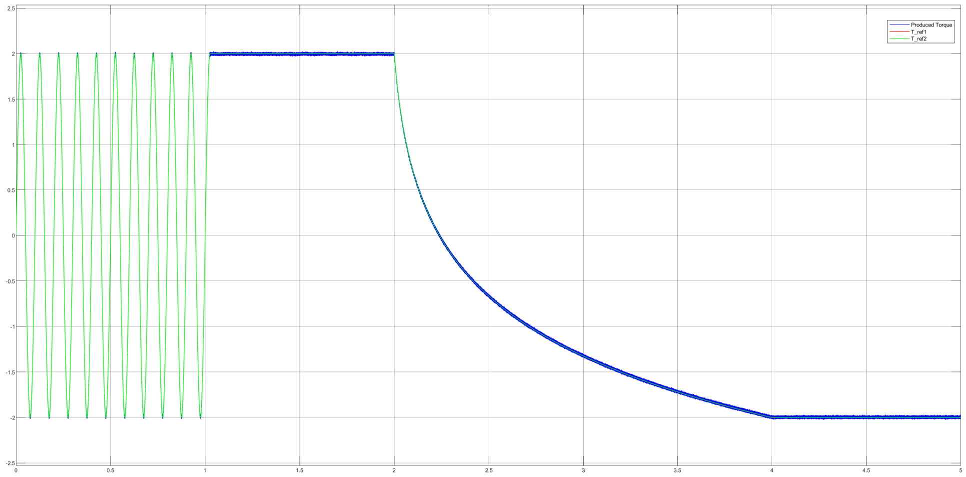


Figure A.45: SS-MPC

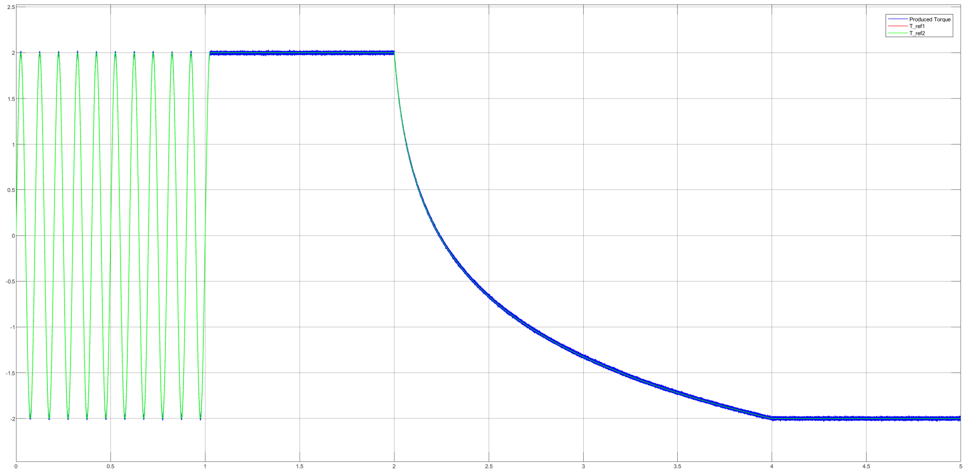


Figure A.46: S-LQR

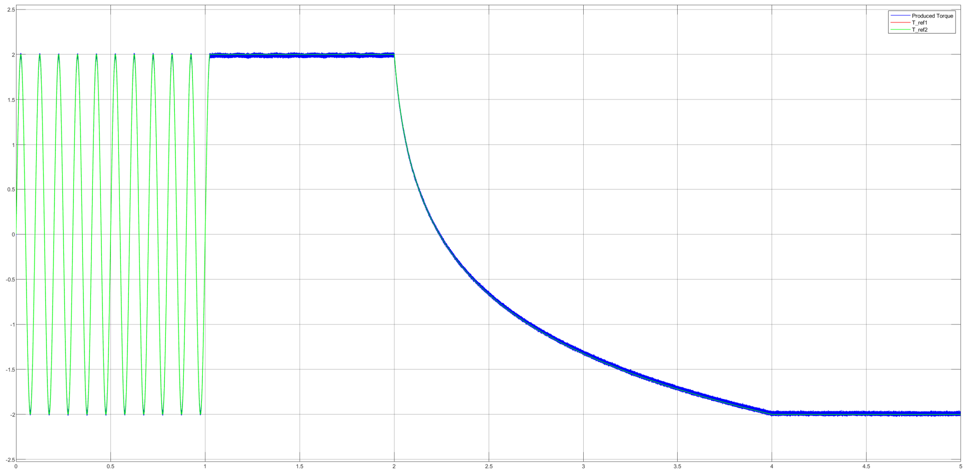


Figure A.47: C-LQR

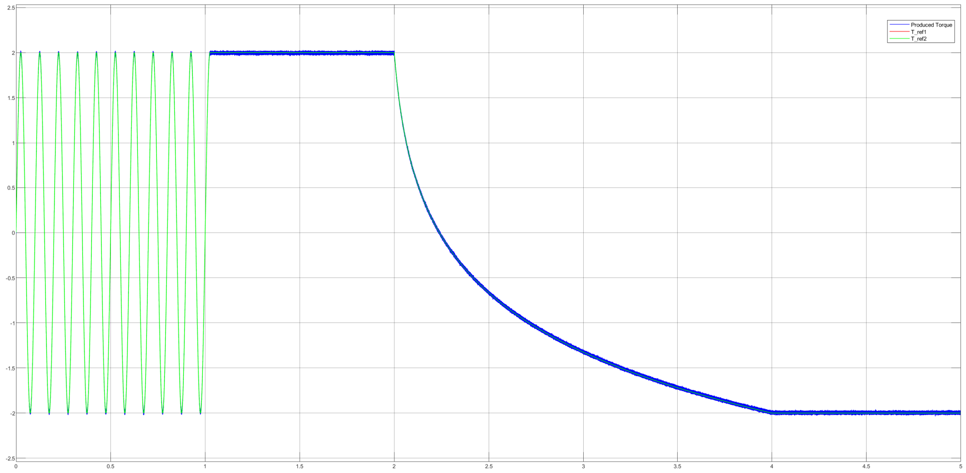


Figure A.48: S-GSC

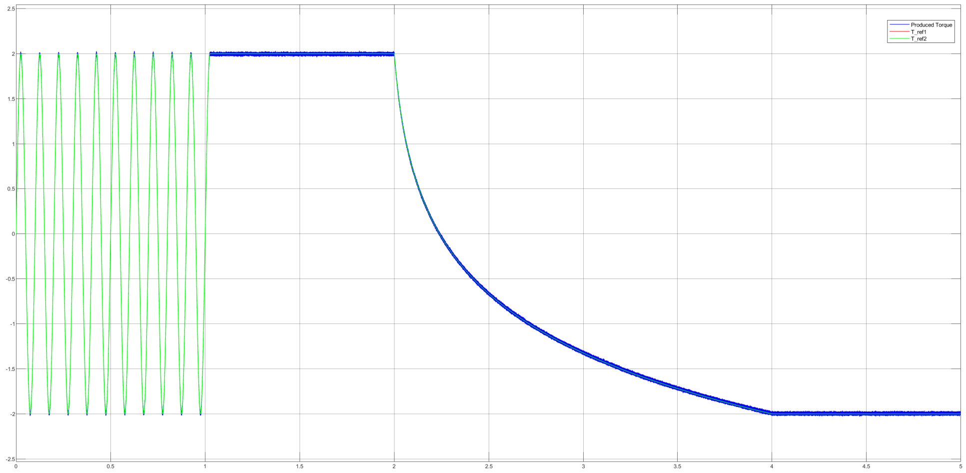


Figure A.49: C-GSC

A.8 Control with Modelled Cogging Torque and No Compensation

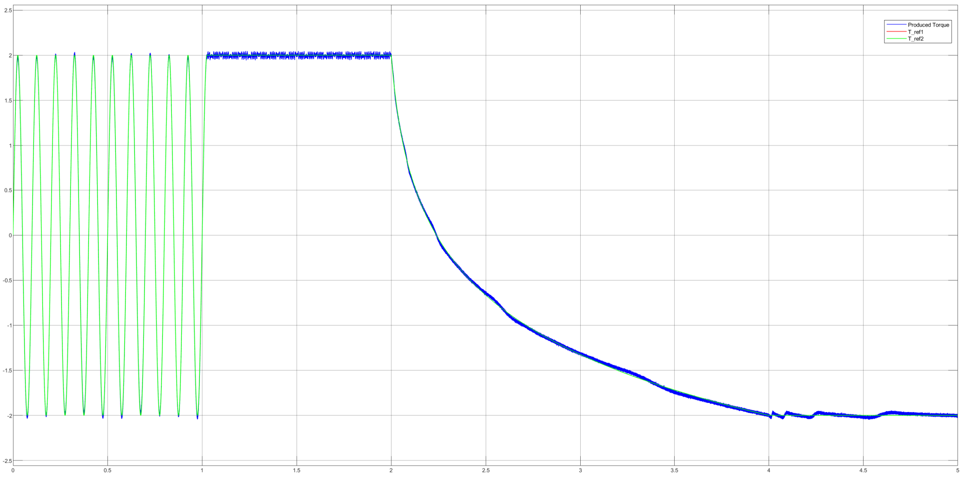


Figure A.50: AC-MPC

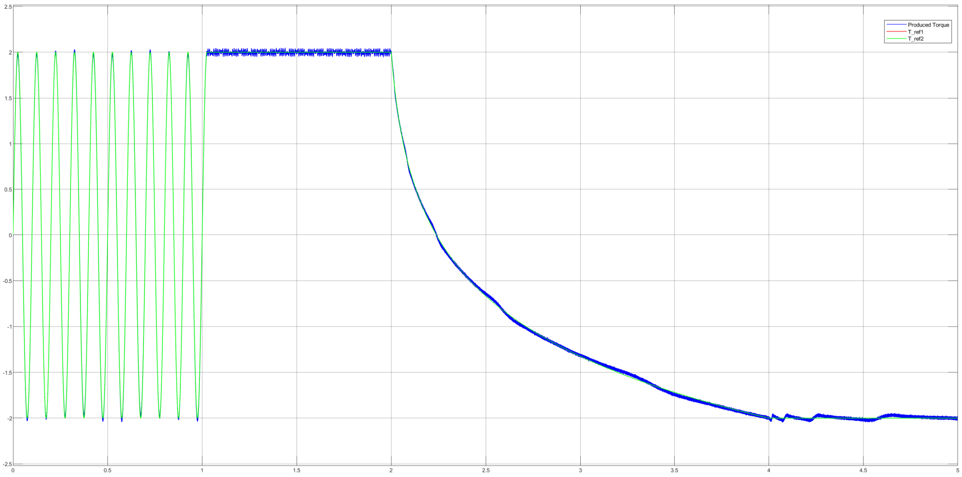


Figure A.51: AS-MPC

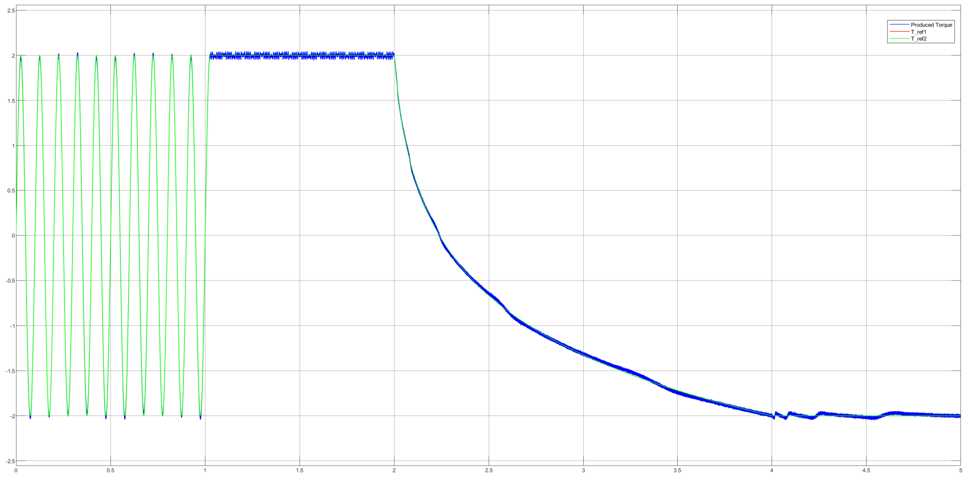


Figure A.52: SS-MPC

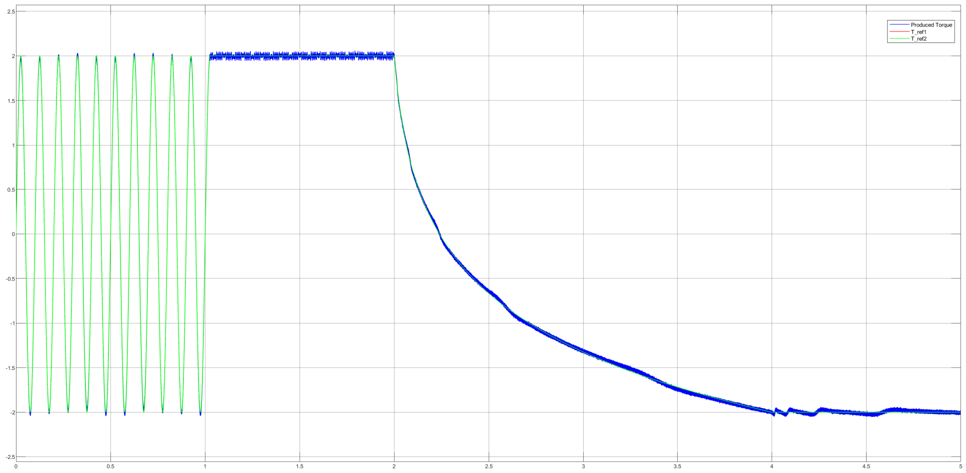


Figure A.53: S-LQR

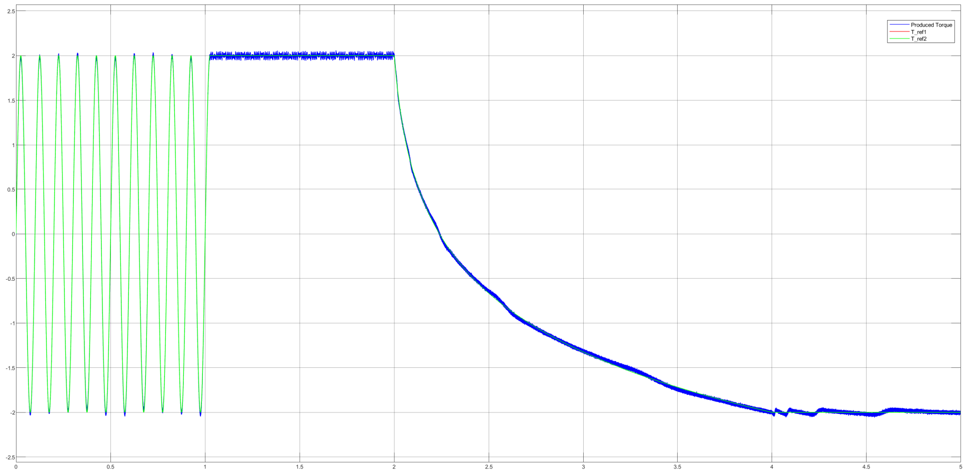


Figure A.54: C-LQR

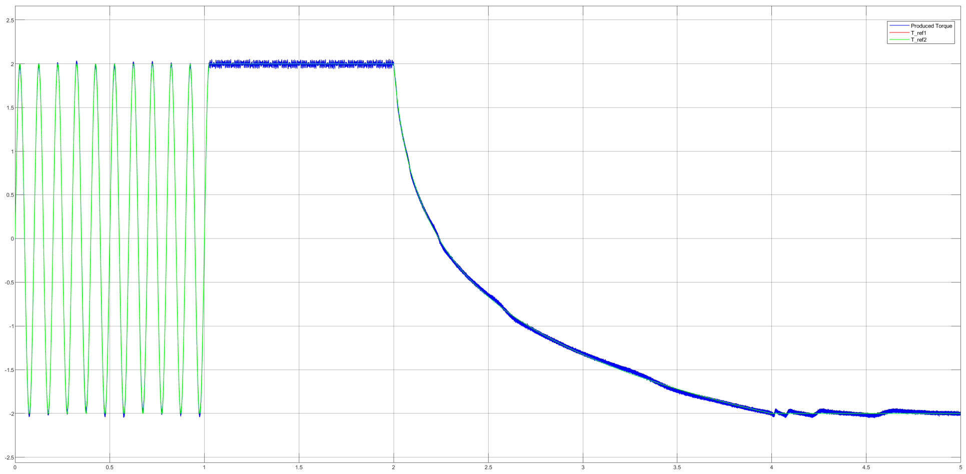


Figure A.55: S-GSC

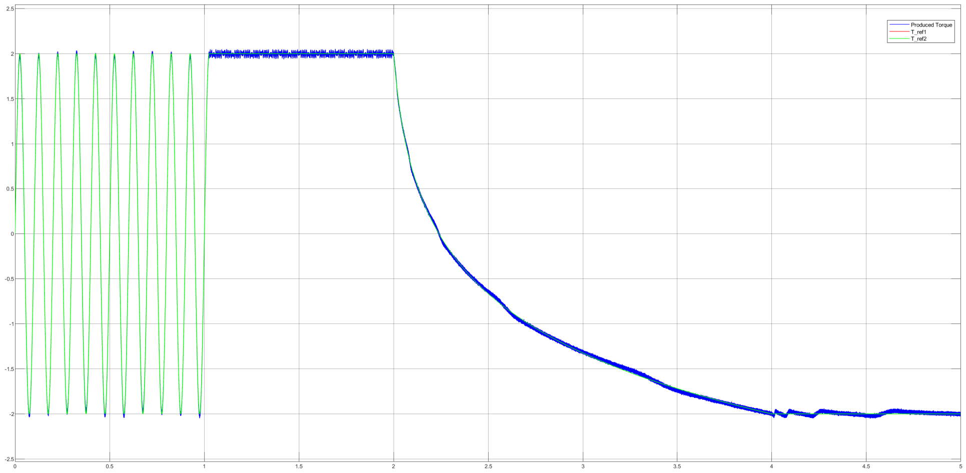


Figure A.56: C-GSC

A.9 Control with Modelled Cogging Torque and Added Compensation

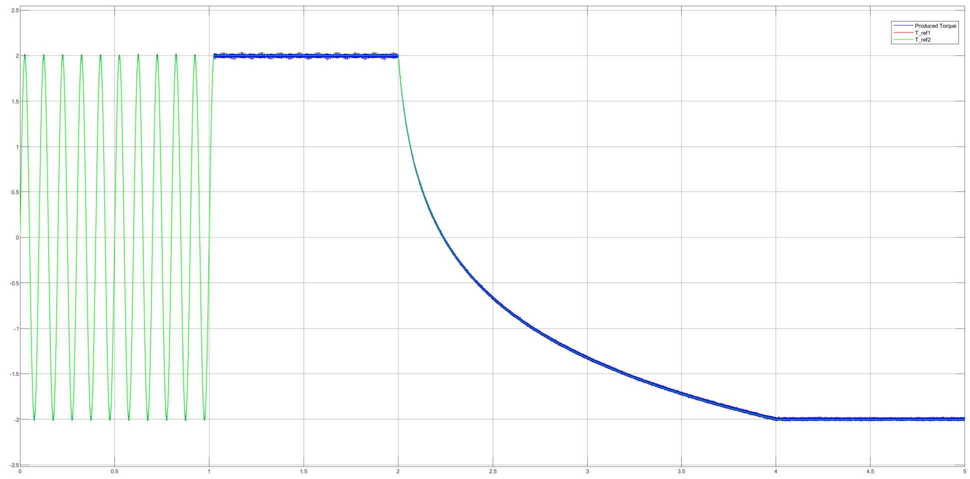


Figure A.57: AC-MPC

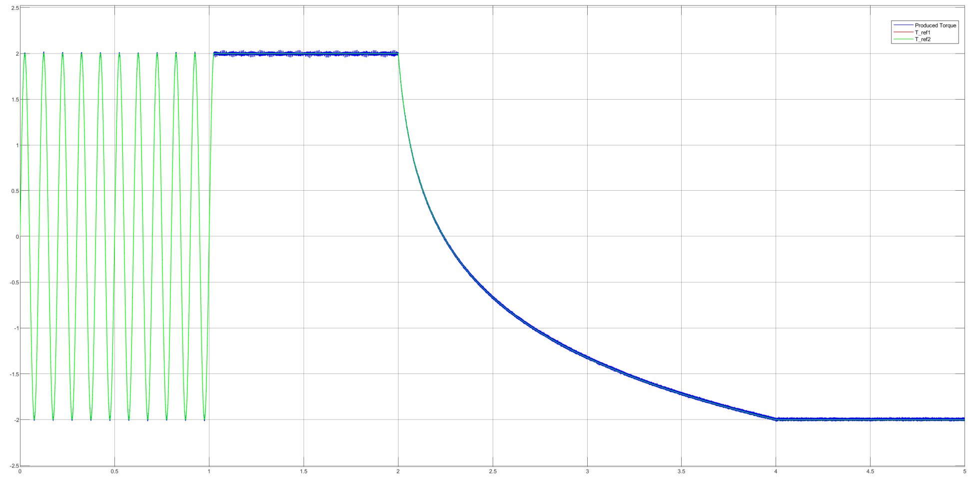


Figure A.58: AS-MPC

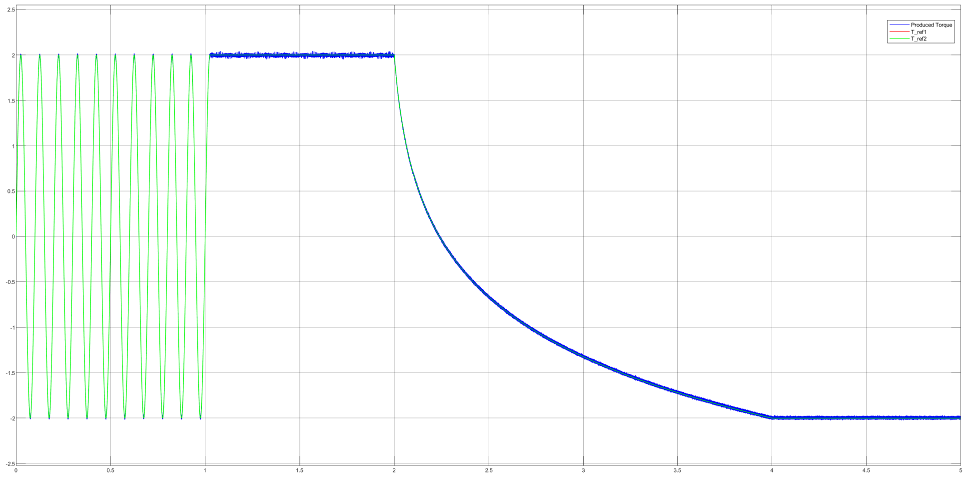


Figure A.59: SS-MPC

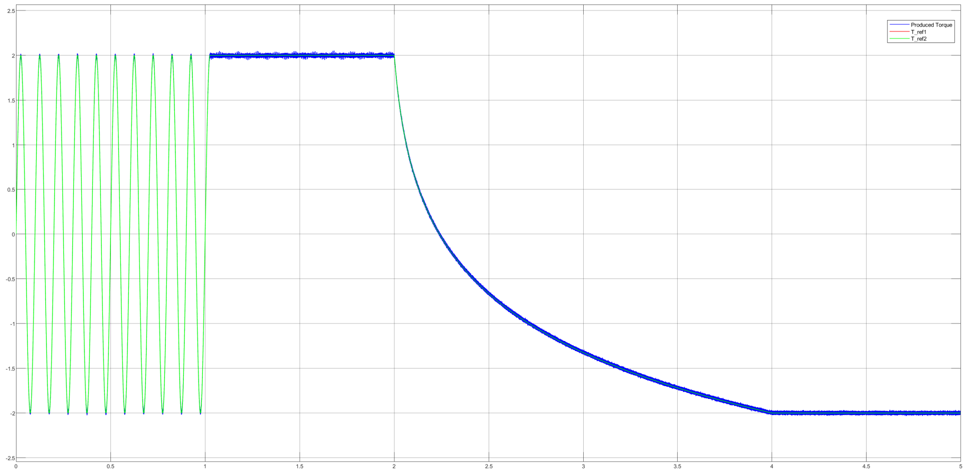


Figure A.60: S-LQR

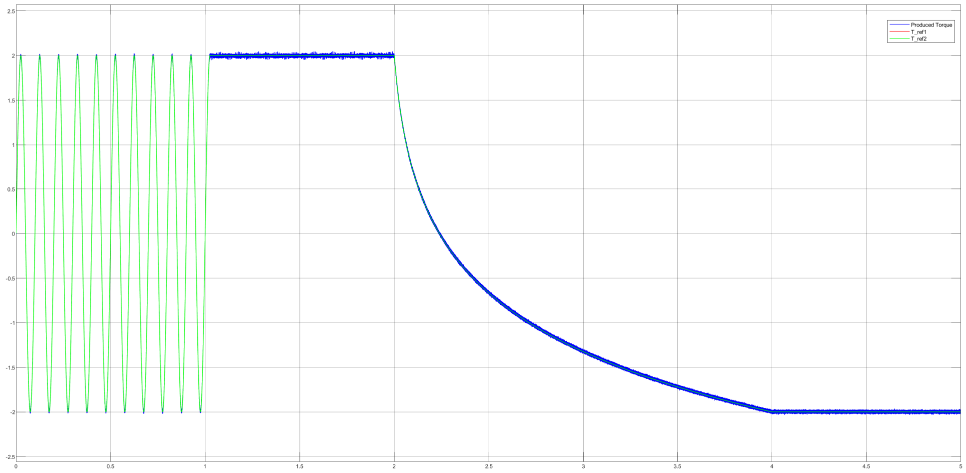


Figure A.61: C-LQR

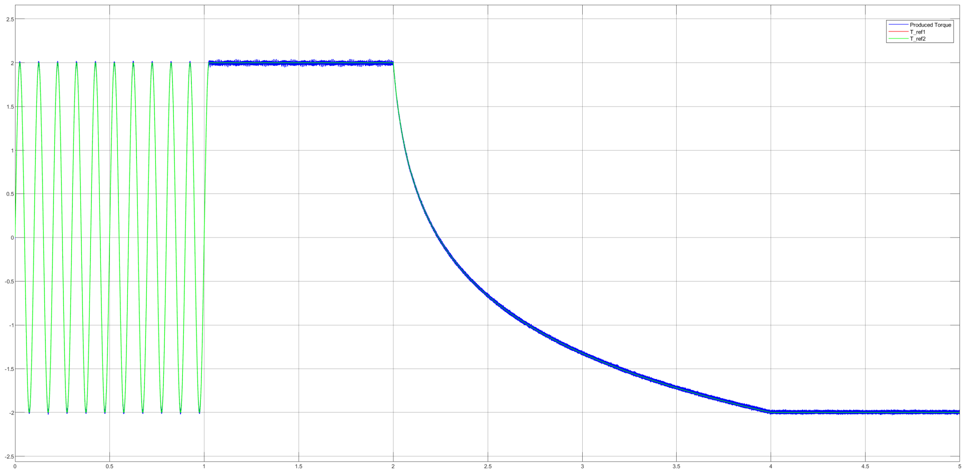


Figure A.62: S-GSC

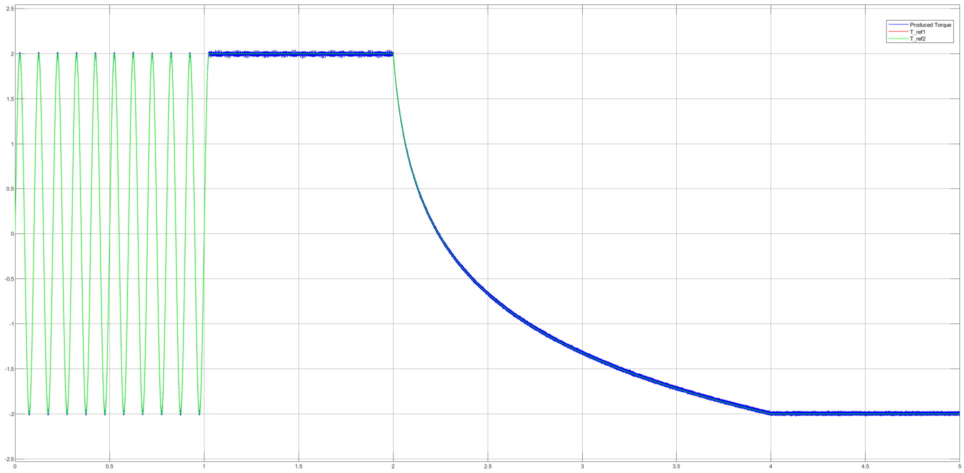


Figure A.63: C-GSC

A.10 Finalised system with realistic dynamometer

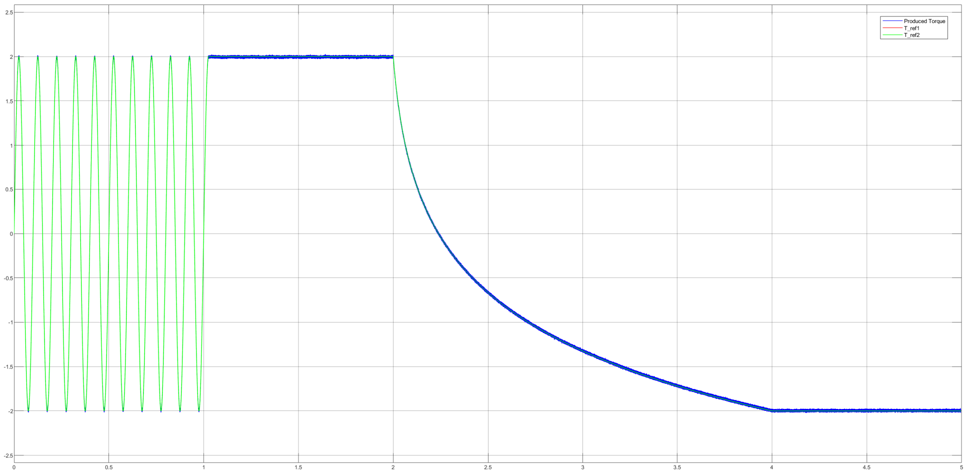


Figure A.64: AC-MPC

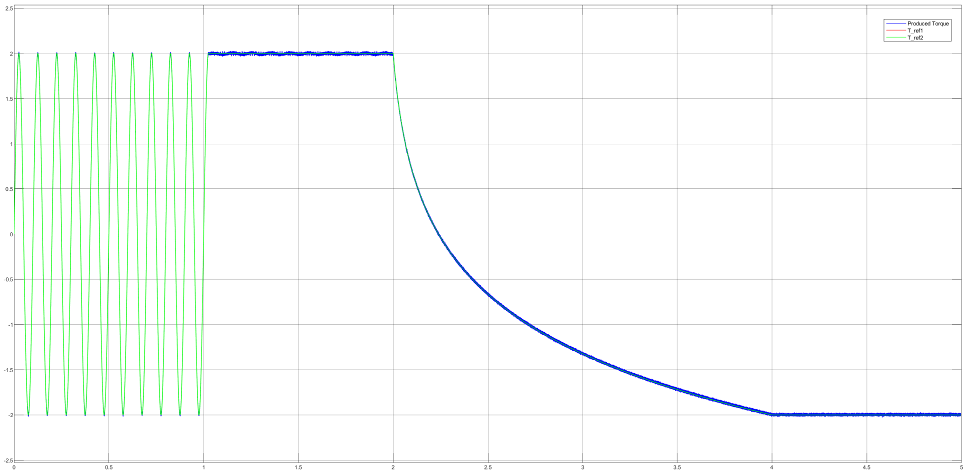


Figure A.65: AS-MPC

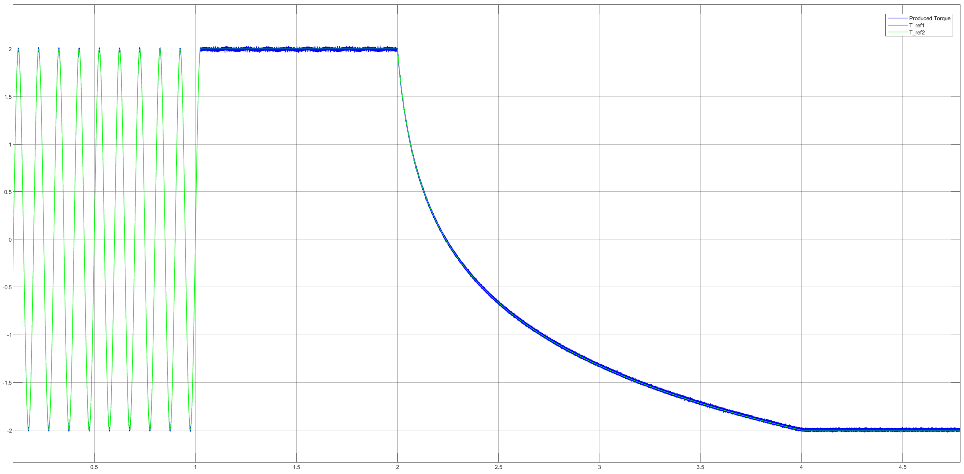


Figure A.66: SS-MPC

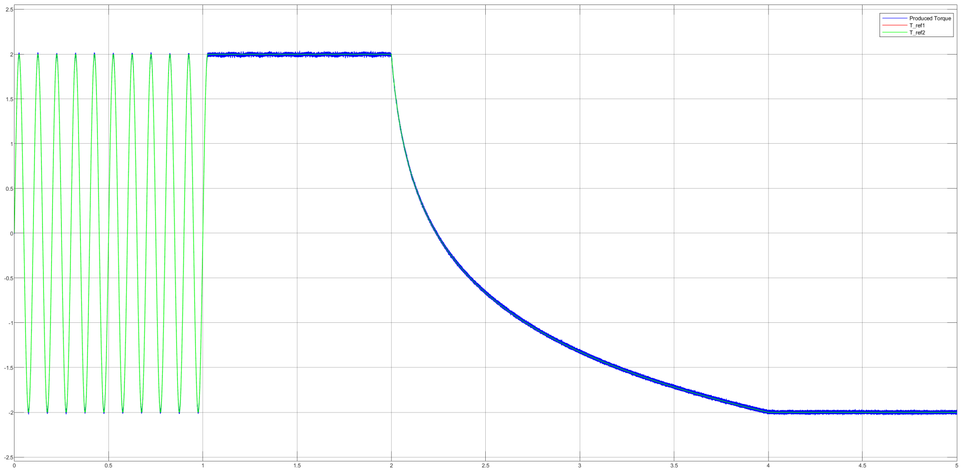


Figure A.67: S-LQR

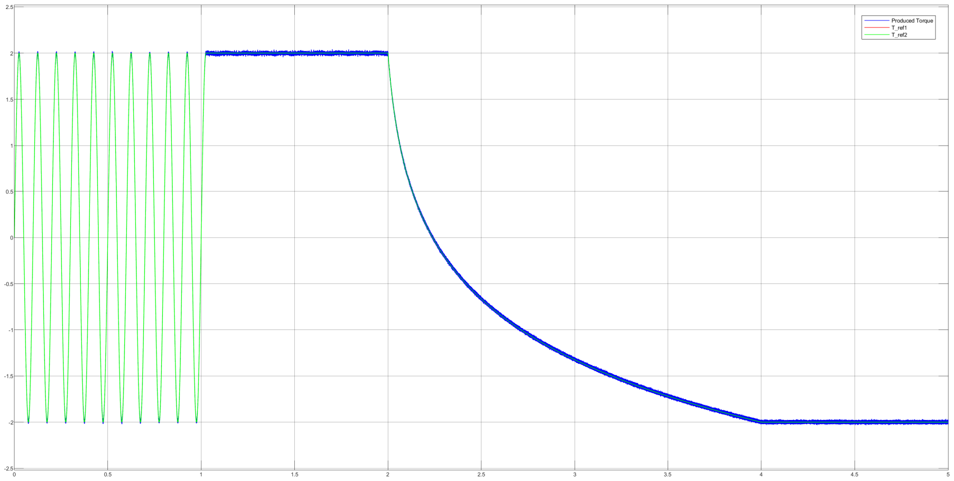


Figure A.68: C-LQR

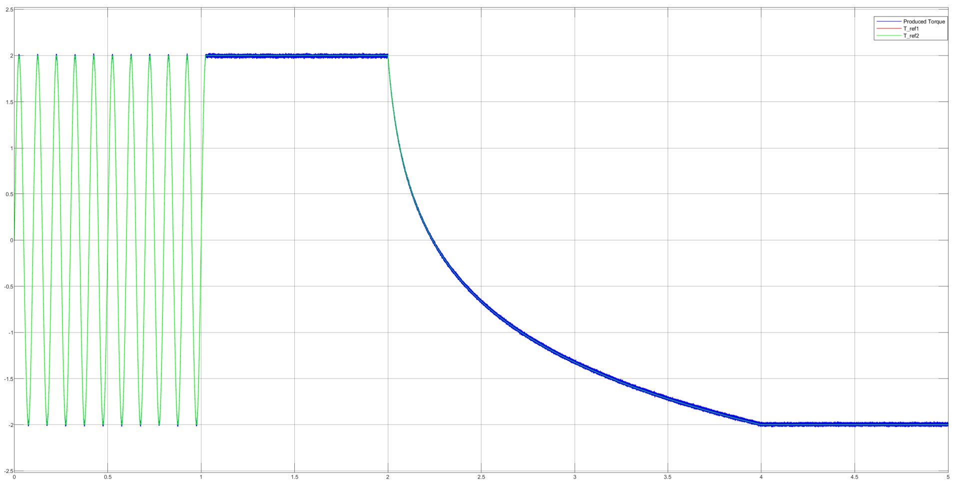


Figure A.69: S-GSC

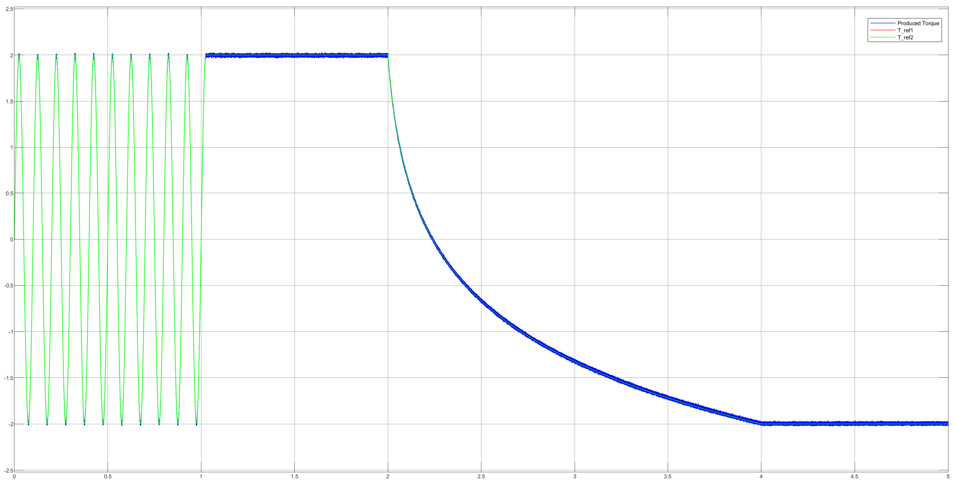


Figure A.70: C-GSC

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY