CHALMERS

Methods for Servo Position Control

Comparison and evaluation of robust solutions with different control implementations

Bernardo T. Barata & Klas Lundgren

Department of Electrical Engineering
Chalmers University of Technology
Göteborg, Sweden 2019

Methods for Servo Control
Comparison and evaluation of robust solutions with different control implementations
KLAS LUNDGREN
BERNARDO BARATA

Supervisors: Daniel Chädström, Jakob Fjellström, Joel Strand, Aros Electronics AB
Examiner: Nikolce Murgovski, Department of Electrical Engineering

Master's Thesis 2019:NN
Department of Electrical Engineering
Division of Division name
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2019

Abstract

Proportional-Integral-Derivative (PID) control is the most abundant type of control used in Industry, yet control theory has progressed with different control options that are not necessarily explored in practical contexts. This project aims to show how other controllers fare in contrast to the benchmark. The mechanical section of the Permanent Magnet Synchronous Motor (PMSM) was the considered system plant. Four controllers were designed and put through multiple tests of disturbance and robustness and reference tracking. Reference tracking included both a step input, for fixed sudden changes, and a sine wave input, for constantly varying setpoints. Robustness and disturbance tests included an added step load torque, white noise on sensor readings, increased delays, varying inertia, cogging torque and stick-slip friction. The selected controllers were the Linear Quadratic Integral (LQI) controller, the Implicit Model Predictive Controller (MPC), the Explicit Model Predictive Controller (EMPC) and Cascade Proportional-Integral Controller (Cascade PI). These controllers were analyzed for certain key parameters, including how much total error they produced and how fast they responded. All simulations were run on Matlab and Simulink and the controllers were designed to be exported into code that could be used in embedded products for future applications. It was noted that every controller had their own strengths and weaknesses but for the performance standards that were requested, the Explicit MPC had the best results. Implicit MPC, however, is the recommended option as a general rule. This study has the potential to help improve the control that already exists for the PMSM motor or, at the very least, increase awareness to explore other options for control.

Keywords: PMSM, MPC, EMPC, LQI, Motor, Cascade PI

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

In many industrial products there is some degree of control that is required in order to achieve stable and precise system functionality. Despite advancements in Control theory throughout the decades, the most widespread controller in industry today is still the Proportional-Integral-Derivative (PID) controller [1], with much of its tuning being derived from trial-and-error methods focused on the specific application outcomes of the product.

PID controllers are conceptually simpler to adapt to problems under fixed conditions and can provide satisfying results when correctly tuned [1]. On the other hand, they supposedly lack robustness in comparison to other control methods within scenarios that include unexpected variations or certain non-linearities [2]. It would, therefore, be interesting to compare different control structures to a tuned PID controller and analyze their performance for a specific system.

In this comparative study, the end application is the Permanent Magnet Synchronous Motor (PMSM) developed at Aros Electronics. The main intention is to control the position of the motor to accurately follow a desired trajectory. The system as a whole can be decomposed into two main control sections - an electric current control section and a torque control section.

The thesis work was initially focused on literature research, wherein most of the efforts were placed on: Understanding how permanent magnet motors work, which controllers would be worth investigating and implementing, understanding the theory behind all the desired controllers, how to actually set up the controllers to function, how broad the thesis would be in terms of whether it would cover the full 3-phase alternating current Field Oriented Control (including current control of the motor) or if it would be specialized on the mechanical DC control portion of the motor, how to convert all the code used in simulation to code that would be usable in actual motor systems, which constraints would have to be considered to closely replicate the motor's functionality and how to assess each controller's performance and compare them with each other via objective parameters.

This study was ultimately decided to be focused on the torque control. The chosen controllers to be analyzed are: a Linear-Quadratic-Integral (LQI), an implicit (online) Model Predictive Controller (MPC), an explicit (offline) MPC controller and, as stated previously, a PID controller.

1.2 Research Questions / Objectives

The primary objective is to assess how each controller performs under different input signals as well as with varying disturbances and system non-linearities.

This assessment involves specific criteria that each controller is judged on. These criteria are: Maximum Torque, Maximum Speed, Torque Root-Mean-Square, Total Error, Maximum Error, Rise Time and Settling Time. The two inputs that are used are the step and sine wave. The former is used to observe how fast a controller tracks and stabilizes to a given static reference value. The latter is used to observe how fast and accurately a controller can track a constantly changing reference value. Both are important to characterize a controller's behavior.

The non-linearities in the system that are considered are: cogging Torque and stick-slip friction. These are added later in the simulations to understand how they affect the performances.

The questions in this study are, therefore:

- Does the PID controller outperform the other controllers in general?
- Which controller is overall the most appropriate for this application?
- What are the best and worst features of each controller for this application?

1.3 Scope

This thesis is limited to the torque control of the PMSM motor. It will not cover the Current control, which is assumed to give a fast and stable response.The controller selection will be restricted to LQI, PID, MPC and EMPC.

1.4 Thesis Outline

This thesis is split into two big sections - Modelling and Control - followed after by Results and Discussion.

The Modelling section describes the PMSM motor system as a whole unit, explaining the core parts and their respective functions. The description places emphasis on the electrical sector to cover the current control and the mechanical sector

to delve into torque control. The Control section gives an overview of the different employed control schemes, how they are actually implemented in the system and also covers other technicalities involved in the schematics.

Lastly, the simulation results are presented and discussed.

2 Theory

2.1 PMSM Motor

A Permanent Magnet Synchronous Motor is a motor that contains permanent magnets that generate constant motor flux attached to the rotor part and whose torque motion is produced by supplying alternating current to different parts of the stator winding so as to produce a rotating magnetic field [3].

The process of supplying current to separate parts of the stator in sequence is called commutation. The rotor part supplies a constant magnetic field while the stator supplies a rotating magnetic field, which induces the rotor to rotate alongside it [4].

As the rotor starts rotating, its poles lock in synchronously to the rotating magnetic field once it reaches synchronous motor speed ω_s, defined (in angular velocity) as

$$\omega_s = 2\pi \frac{f}{pp} \tag{1}$$

where f is the AC current frequency and pp is the number of pole pairs of the rotor. From that point on, the synchronous speed of the rotor depends on the supplied AC frequency [4].

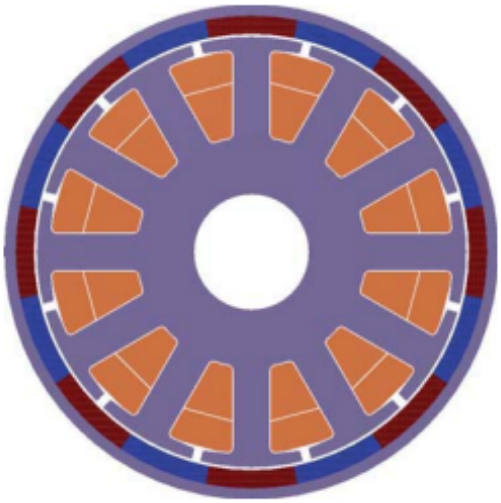


Figure 1: Cross-section of a permanent magnet rotor with an outer rotor [4].

2.2 Motor Model

The working schematic for the PMSM motor used at Aros Electronics is shown in Figure 2, which applies to the generality of PMSM motors simulation-wise. This is not the environment used in the thesis but rather the whole setting in which it is integrated, giving an overview of the entire system.

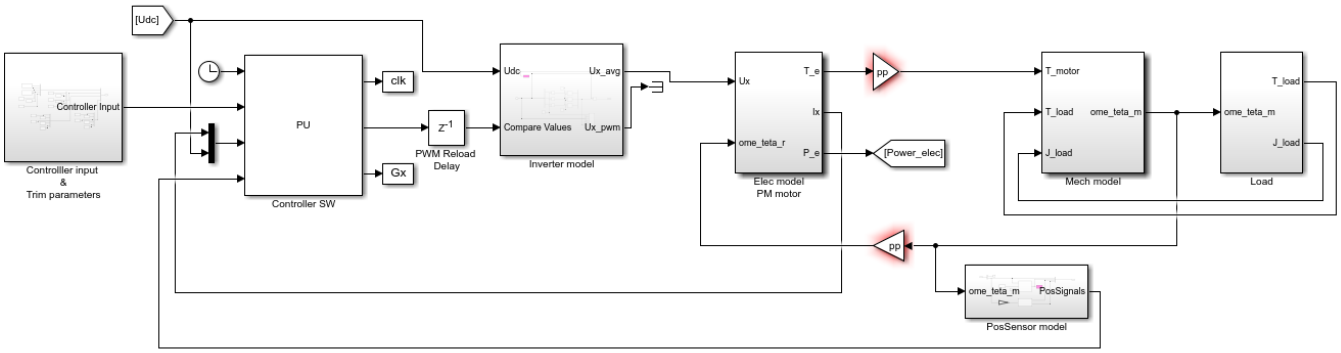


Figure 2: Overall schematic of the Aros PMSM motor.

The subsystems that stand out from the schematic are: the controllers, the inverter model, the electrical model of the motor, the mechanical model with respective load response and lastly the sensor model.

The motor itself works on 3-phase Alternating Current (AC) but since it is far more practical to control states via Direct Current (DC) values, then an inverter is applied to handle the conversion between DC reference values to actual working AC signals for the motor. Sensors are then used to interpret motor state values into DC form.

2.2.1 Inverter Model

The inverter in consideration is a three-phase Pulse-Width-Modulation (PWM) Inverter, its purpose is to convert a DC reference signal into a functional AC signal for the remainder of the model.

Pulse-Width-Modulation is a technique that utilizes on-off switches with varying duty cycles to represent more complex alternating signals [5].

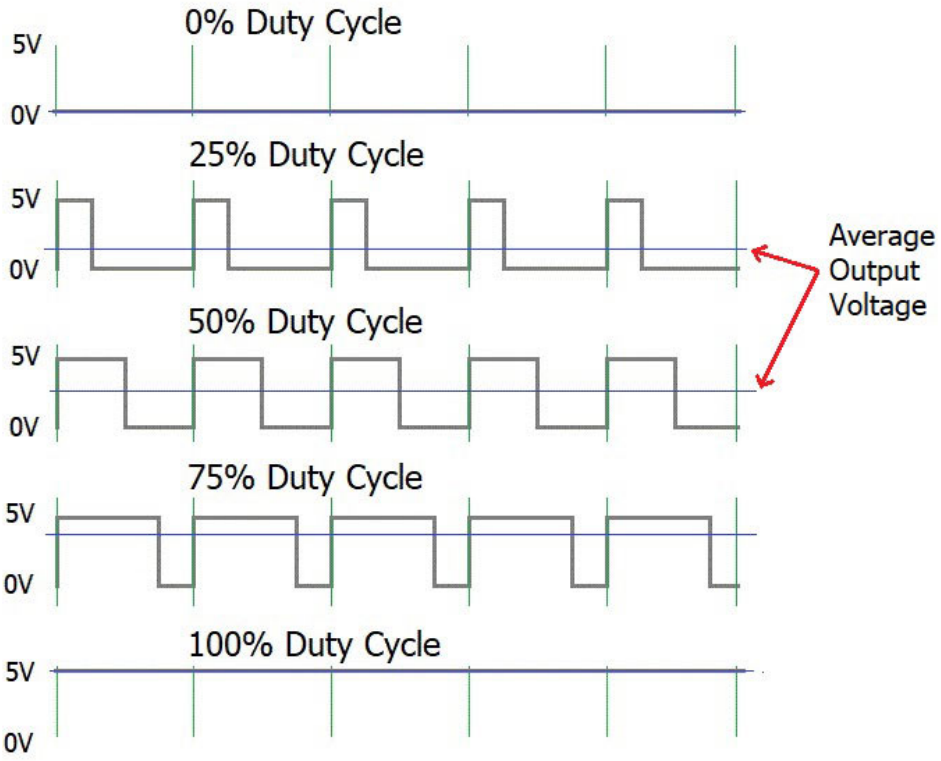


Figure 3: PWM example [6].

As shown in Figure 3, the duty cycle of the pulsing signal gives information about the original signal over the pulsing signal's period. The higher the frequency of the pulsing signal, the higher the accuracy in representing the original signal. These Duty Cycle values are obtained by comparing the original signal with a triangular carrier wave, if the signal has a higher value than the wave - it outputs a HIGH value, otherwise it outputs a LOW value [7]. Since this is a three-phase PWM inverter, then it is necessary to have enough switches to handle outputting a pulsing signal for each phase, which would be separated by 120°.

Lastly the PWM three-phased signal is passed through a Low-pass filter to yield an approximation of an AC signal based on the referenced DC signal.

2.2.2 Sensor Model

The sensor is essentially an encoder which is used to derive the angular position from the motor's behavior. Its behavior was observed directly from one of Aros's sensor simulated models

The way it works is fairly simple. By following the equation,

$$\theta = \theta_m - \lfloor \frac{\theta_m}{2\pi} \rfloor \cdot 2\pi \tag{2}$$

it starts by obtaining the angle θ in between a full cycle interval of $[0, ..., 2\pi]$ from the current mechanical angle θ_m, by removing the amount of redundant rotations it has performed already

Then, by knowing the encoder's Cycles Per Revolution value (CPR), the amount of radians per encoder count is calculated

$$Enc_\theta = \frac{2\pi}{4 \cdot Enc_{CPR}} \tag{3}$$

With these two variables, it is possible to obtain a rough idea of which index of the encoder's count range the position lies in

$$Index = \lfloor \frac{\theta}{Enc_\theta} \rfloor \tag{4}$$

Lastly, it is important to figure out the actual edge that separates two indexes in the encoder. To this effect, the ideal edge is derived and then the encoder error is accounted for also.

$$\theta_{Ideal-edge} = Index \cdot Enc_\theta + \frac{Enc_\theta}{2} \tag{5}$$

$$\theta_{Real-edge} = \theta_{Ideal-edge} + Enc_{error} \tag{6}$$

So if θ is below this real edge value, then the encoder value is the previously calculated index, if it is above it then the encoder value is the next index. The position encoder value is then fed back into the controller.

2.3 Electrical Section

The electrical part of the model involves controlling the flowing current in order to produce a torque reference for the mechanical part. However, since the reference input voltage is a three-phased AC signal, it would be desirable to transform this signal into a reference frame that resembles DC signals so as to facilitate the control process and to compute the torque reference [8].

To this effect, Clarke and Park transformations are applied on the input voltage U_{abc} to obtain U_{dq0}, the input voltage in the Direct-Quadrature frame. The Clarke transform starts by changing the frame from a fixed stator three-phased (abc) reference to a fixed stator orthogonal two-phased ($\alpha\beta$) reference [9].

$$\begin{bmatrix} U_\alpha \\ U_\beta \\ U_0 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{-1}{3} & \frac{-1}{3} \\ 0 & \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{3}} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} U_a \\ U_b \\ U_c \end{bmatrix} \tag{7}$$

The Park transform then changes the fixed stator two-phased reference to a rotating two-phased reference (dq) by using the instantaneous flux angle θ from the motor model [10].

$$\begin{bmatrix} U_d \\ U_q \\ U_0 \end{bmatrix} = \begin{bmatrix} cos(\theta t) & sin(\theta t) & 0 \\ -sin(\theta t) & cos(\theta t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U_\alpha \\ U_\beta \\ U_0 \end{bmatrix} \tag{8}$$

After this conversion, the direct-quadrature values of the current are obtained through the differential equations:

$$\dot{i_d} = \frac{U_d - R_s \cdot i_d + \omega_m \cdot i_q \cdot L_q}{L_d} \tag{9}$$

$$\dot{i_q} = \frac{U_q - R_s \cdot i_q - \omega_m (i_d \cdot L_d + \Psi_{PM})}{L_q} \tag{10}$$

where L_d and L_q are the direct and quadrature axis synchronous inductances, Ψ_{PM} is the flux produced by the permanent magnets, R_s is the stator resistance and ω_m is the angular velocity obtained from the motor. These currents can then be controlled by the controller [11].

Figure 4 shows the difference in reference frames after the transformations

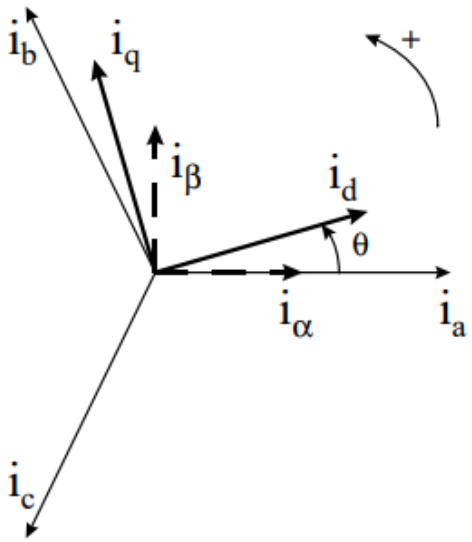


Figure 4: abc, $\alpha\beta$ and dq frames [12]

To compute the torque reference (electromagnetic torque) from these values the following equation is used

$$T_e = \frac{3pp}{2} \cdot (i_q i_d (L_d - L_q) + \Psi_{PM} i_q) \tag{11}$$

where pp is the amount of pole pairs considered in the PMSM motor [13].

2.4 Mechanical Section

The mechanical dynamics are simply expressed straight from Newton's rotational laws as

$$J\frac{d\omega_m}{dt} = T_e - T_L - b\omega_m \tag{12}$$

where J is the inertia, ω_m is the angular velocity measured from the motor, T_L is a constant load torque and b is the mechanical damping constant [14].

This is the focus of the thesis for which the controllers are designed - given a position reference, the mechanical section of the motor is controlled to yield fast and stable convergence to the desired setpoint.

2.5 State-space Derivation

As mentioned before, the control is focused on the mechanical section of the motor - to this effect the electrical section is simplified through the use of a first-order transfer function that approximates the passage of reference torque to actual mechanical Torque.

The equation in the mechanical section of the report can be altered so that it becomes:

$$J\frac{d\omega_m}{dt} = T_m - T_L - b\omega_m \tag{13}$$

in which the load torque T_L is considered to be an input coming from the load. By considering the aforementioned transfer function as

$$G(s) = \frac{1}{\tau_T s + 1} \tag{14}$$

where τ_T is a time constant given by Aros Electronics that represents the observed

delay while assuming close to ideal current control, then

$$\tau_T \frac{dT_m}{dt} = -T_m + T_{ref} \tag{15}$$

in which T_m is the mechanical torque and T_{ref} is the input torque reference.

The two main states to be controlled are then the angular velocity ω_m and the mechanical torque T_m, while the last state is the desired output variable - angular position θ_m.

$$\frac{d\theta_m}{dt} = \omega_m \tag{16}$$

Given these differential equations and by defining the state vector as $x = [\theta_m, \omega_m, T_m]^T$ and the input vector as $u = [T_{ref}, T_L]^T$, the state-space is immediately derived as

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{1}{J} \\ 0 & 0 & -\frac{1}{\tau_T} \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & \frac{1}{J} \\ \frac{1}{\tau_T} & 0 \end{bmatrix} u \tag{17}$$

As for the output, it depends on the controller that is used since some controllers require all state information but generally speaking the desired output is only the angular position θ_m, therefore

$$y = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \end{bmatrix} u \tag{18}$$

2.6 Discretization

Since the controller, in reality, handles digital values read from the sensors and not actual continuous values of the states, then it is necessary to convert the state-space from continuous time to discrete time for a more accurate representation. From the continuous state-space representation:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{19}$$

By multiplying with the factor e^{-At}, rearranging the equation and substituting $\frac{d}{dt}e^{-At}x(t) = e^{-At}\dot{x}(t) - e^{-A}Ax(t)$, the eqatuion becomes:

$$\frac{d}{dt}e^{-At}x(t) = e^{-At}Bu(t) \tag{20}$$

Finally by integrating both sides and considering a Zero-order-hold (input maintained constant between sampling instants) one obtains the solution:

$$x(t_n) = e^{A(t_n - t_{n-1})}x(t_{n-1}) + \int_{t_{n-1}}^{t_n} e^{A(t_n - t)}B \, dt \cdot u_{n-1} \tag{21}$$

Since this equation is dependent on $t_n - t_{n-1}$, or, in other words, the sampling time T_s of the continuous values, then its choice heavily affects the outputs [15].

For the controllers in this thesis the choice of controller sampling time was $T_c = 0.001s$. Given that the state-space constants that were considered before were valued as $\tau_T = 10^{-3}$, $J = 3.5 \cdot 10^{-5}$ and $b = 1e \cdot 10^{-4}$, then the discretized state-space is

$$x(k+1) = \begin{bmatrix} 1 & 10^{-3} & 1.3821 \cdot 0.0105 \\ 0 & 1 & 18.03 \\ 0 & 0 & 0.3679 \end{bmatrix} x(k) + \begin{bmatrix} 0.0038 & 0.0143 \\ 10.5 & 28.53 \\ 0.6321 & 0 \end{bmatrix} u(k) \tag{22}$$

However, the actual sensor sampling time - so the sample time from the continuous plant T_p is faster than that of the controllers. The sensor sampling time is considered to be $T_p = 0.0001s$

2.7 Constraints

Since this is a real physical system, constraints on the states and inputs have to be considered during the controller design given that it is not an ideal motor.

The primary constraint is that the torque reference must be within the range $[-1, 1]\ N \cdot m$ as this is the maximum allowed input torque for the motor. It is defined as a hard constraint, so exceeding it is strictly not possible. The other constraint is on the maximum angular velocity ω_m, which should be within the bounds $[-2\pi \cdot 50, 2\pi \cdot 50]\ rad.s^{-1}$. This is defined as a soft constraint, as it can technically be exceeded despite not being favorable and potentially causing mechanical problems. Another factor that can be seen as a constraint is the inherent computational delays that have to be considered. Delays from the sampling process are inserted in the schematics.

2.8 Nonlinearities

As is common in most real systems, there are often non-linear factors that have to be accounted for during run-time. In this thesis, two experimentally observable non-linearities are explored in the simulations: stick-slip Friction and cogging torque.

When the stator and rotor pole teeth of a Permanent Magnet motor are aligned in a certain position and there is no actuator involved, there exists some break-away force to go from that aligned position to the next aligned position. That force is the cogging torque [16].

Once the motor is actuated, especially at low speeds, the cogging torque can be somewhat impactful, causing a ripple effect on the output position - inducing

undesired oscillatory behaviour [16]. The cogging torque is designed to be

$$\tau_{cog} = 0.1\ cos(6 \cdot pp \cdot \theta) = 0.1\ cos(18 \cdot \theta)\ [N \cdot m] \tag{23}$$

where in this case the pole pairs (pp) of the PMSM are 3.

The stick-slip friction is basically representative of a high inertial friction at the start of the motor's movement which then immediately declines past a certain rotational speed ω_m. It creates a jerk-like effect when transitioning from static behavior to dynamic [17]. For the simulations it is designed to produce a Friction Torque $b\omega_m$ which is as shown in Figure 5, for an example of a time-proportionally increasing ω_m from $[-5, 5]\ rad \cdot s^{-1}$ (such that it follows a $\omega_m = t - 5$ function), over a span of 10s simulation time,

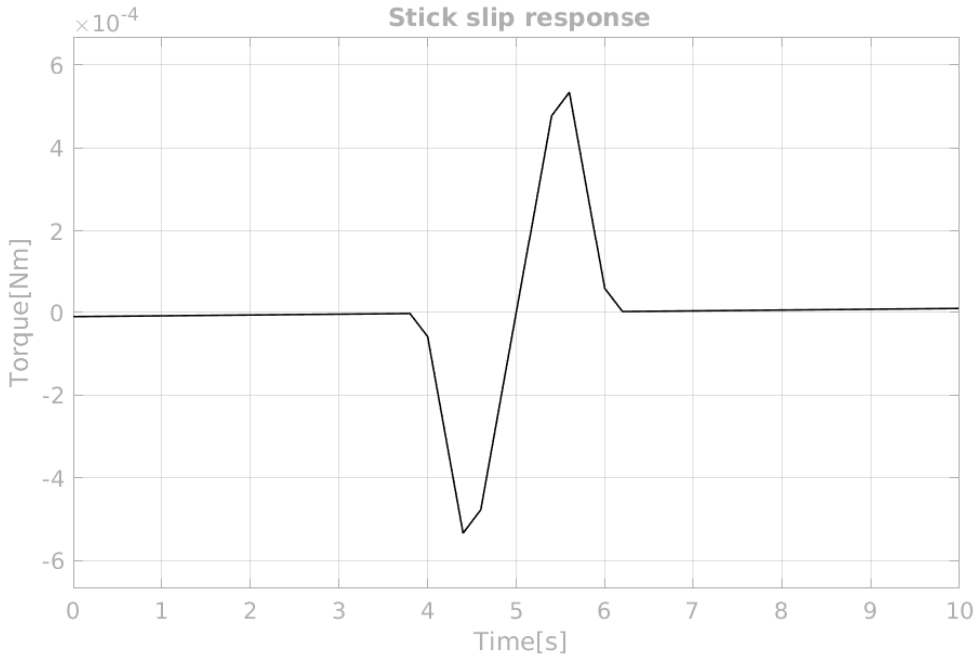


Figure 5: Friction torque response.

2.9 Proportional-Integral-Derivative Controller

The PID controller is one of the most straightforward control schemes to incorporate into a system. As the name suggests, it contains a proportional, an integral and a derivative component - each with their own functionality.

In a controlled system, the process variable to be controlled is fed back to the Plant and is compared to its desired set point, resulting in an error signal. As in

Fig. 6, the PID components act on both the process variable and the resulting error.

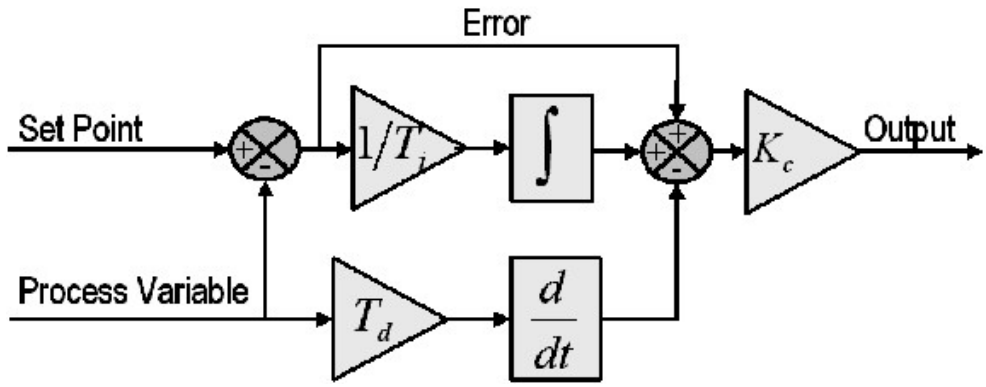


Figure 6: Control setup for PID [18].

The proportional component is characterized by the proportional Gain K_c which produces a proportional response relative to the amount of existing error. The more one increases K_C the higher the response becomes, resulting in a faster control system response to a certain set point. If the gain is increased too much, however, then even for slight errors the response can become too significant - resulting in oscillatory behavior as the system's Steady-state never closely stabilizes towards the set point. Further increasing K_c will render the system unstable [18].

The integral component is characterized by the integral time Parameter T_i followed by an integrator. Its purpose is to sum the error over time such as to induce an increasing response unless the set point is tracked accurately - which creates incentive for the Steady-state error to drive towards zero. If the integral component's gain is too high, depending on the system's physical limitations, then an integral windup can occur - wherein the integral part's demands on the system are too high as the error accumulated becomes too dense resulting in saturated feedback. In turn this can lead to high overshoots in the response [18].

The derivative component is characterized by the derivative time parameter T_d followed by a derivative block, which acts directly on the process variable. The derivative response is then dependent on the derivative of the variable, so the rate of change. The higher the T_d the more the system responds to variations - which, in turn, suggests that to achieve more stable responses low values of T_d are more favorable. In this thesis the derivative component is set to 0, as it did not appear to be necessary for the application in question - the controller is then only PI-based [18].

2.10 Linear-Quadratic-Integral Controller

The LQI control algorithm aims to optimally determine the control signal that minimizes a defined cost function while following any existent constraints in the system. As the name suggests, it is intended to be used on a linear system and its cost function is of quadratic nature [19]. It also includes an extra integration state which serves to reduce the steady-state error to zero when tracking a reference.

This cost function J, for an infinite-time horizon and a discrete state-space, can be generally expressed as:

$$J = \sum_{n=0}^{\infty} x^T Q x + u^T R u \tag{24}$$

in which x corresponds to the states of the state-space with the addition of an integral state x_i - which is the discrete integration of the error between the reference and the desired state to be tracked $x_{ref} - x$, u corresponds to the input and Q and R are their respective diagonal weight matrices. Q and R are selected based on how much one intends to penalize deviations on the states or the plant input [19].

Once the cost function is defined, it is then necessary to solve it as an optimal minimization problem, such that:

$$\begin{aligned} min\ _u\ J &= \sum_{n=0}^{\infty} x^T Q x + u^T R u \\ s.t.\ x(k+1) &= Ax(k) + Bu; \\ x(0) &= x_0 \end{aligned} \tag{25}$$

where A and B are the two matrices that define the state-space and x_0 is the initial state.

There are several methods to obtain a solution to this minimization problem, one of them is via the Discrete Algebraic Ricatti Equation (DARE) [20]. Since the cost function is of infinite-time horizon, the optimal control solution can be defined as

$$u = -(R + B^T P B)^{-1} B^T P A x = -K_d x \tag{26}$$

P becomes constant and can be derived from the DARE equation iteratively:

$$P = A^T P A - (A^T P B)(R + B^T P B)^{-1}(B^T P A) + Q \tag{27}$$

Since the end goal is to track a position reference for the motor then, in fact, the optimal control solution is applied such that

$$u = K_d \begin{bmatrix} x \\ x_i \end{bmatrix} \tag{28}$$

where x_i is the integration state, so the integral of the error $x_{ref} - x_p$, x_{ref} is the reference input for the position state and x_p would be the position state (taken from the general x state vector) [21].

2.11 Implicit Model Predictive Controller

The general working principle of model predictive controllers is to generate predicted outputs $\hat{y}(t+k|t)$ from past inputs and outputs via the system's model and compute predicted future inputs $u(t+k|t)$ by optimizing a cost function so as to reduce the predicted error as much as possible. The first predicted input from the most optimal sequence of predicted inputs is then utilized as the next actual control input in the system model.

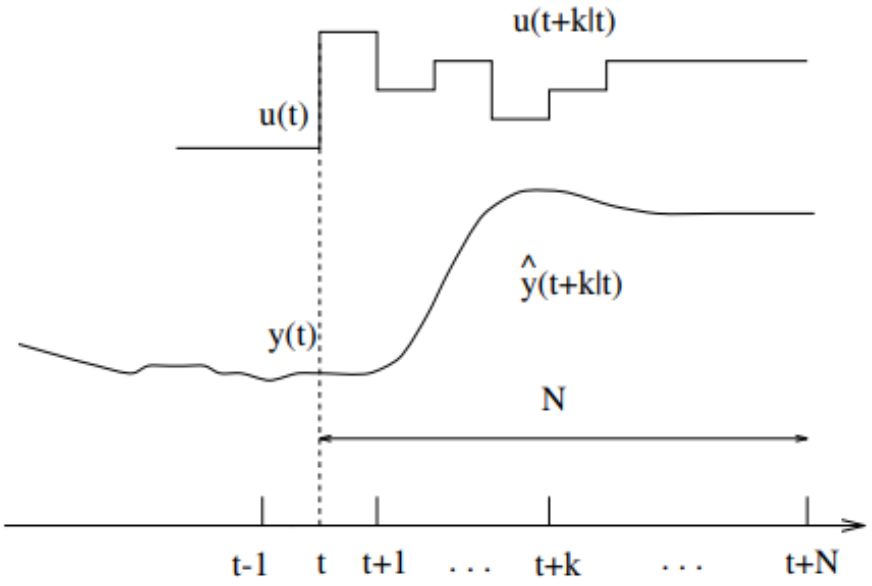


Figure 7: MPC prediction overview [22].

As seen in Fig. 7, the prediction values for the outputs are generated over a span of N time steps, known as the Prediction Horizon. The control input values can be computed within this scope but they have their own respective range of M values, known as the Control Horizon. Typically $M \leq N$, in the case where $M < N$ the computed control inputs are assumed to be constant after the Mth-time step.

For the Implicit MPC controller, these calculations are done online - meaning that the sequence of predicted outputs and control inputs is calculated on the fly,

which can potentially be quite computationally demanding since they are performed for every time step [22]. The first predicted output is calculated based on the system's linearized model

$$\hat{x}(t+1|t) = Ax(t) + B\hat{u}(t|t) \tag{29}$$

$$\hat{y}(t+1|t) = C\hat{x}(t+1|t) \tag{30}$$

In the following predicted outputs there is a dependency on the previously computed predictions, and this goes on and on over the span of the prediction horizon

$$\hat{x}(t+2|t) = A\hat{x}(t+1|t) + B\hat{u}(t+1|t) \tag{31}$$

$$\hat{y}(t+2|t) = C\hat{x}(t+2|t) \tag{32}$$

To obtain the optimal control inputs there is a quadratic cost function that is aimed to be minimized - for the unconstrained case it typically looks as follows:

$$V = \sum_{n=1}^{N} \|\hat{y}(t+i|t) - r(t+i|t)\|^2_{Q(i)} + \sum_{n=0}^{M} \|\delta\hat{u}(t+i|t)\|^2_{R(i)} \tag{33}$$

where N is the prediction horizon, M is the control horizon, $r(k+i|t)$ is the predicted reference value, $Q(i)$ and $R(i)$ are the respective weights and $\delta\hat{u}(t+i|t) = \hat{u}(t+i|t) - \hat{u}(t)$ are the control input increments [23].

The equations can then be inserted into the cost function and rearranged to obtain an expression which is dependent only on the known information and the control input increments, which can then be altogether minimized to obtain the optimal increments and therefore acquire the optimal sequence of control inputs. Constraints on the states and control inputs can also be rearranged into a system of equations that is dependent only on the control input increments and can therefore be added as well into the cost function.

2.12 Explicit Model Predictive Controller

The theory behind the Explicit version of MPC is the same at its core as the Implicit version. The point where it diverges from the Implicit controller lies on its major advantage regarding computational complexity, all the calculations are performed beforehand and, in turn, memory capacity becomes the primary setback.

To be able to achieve this, the cost function in the MPC controller can be converted into a Multiparametric Quadratic Programming (mp-QP) problem. In this form it incorporates all of the constraints and, when solved, outputs a span of different regions that encompass the optimal inputs for different combinations of state values (the entire state space). With this collection of critical regions it then becomes straightforward during run time to assess which is the optimal input, all it

takes is feeding in the current state information and retrieving the control input that is mapped to those states [24].

The cost function and all the constraints can be converted into an mp-QP problem as such:

$$\begin{aligned} min\ _U\ \frac{1}{2}U'HU + x'(t)FU + \frac{1}{2}x'(t)Yx(t) \\ s.t.\ GU \leq W + Sx(t) \end{aligned} \tag{34}$$

where $U = [u(t), u(t+1), ..., u(t+N-1)]'$ is the array of all the necessary control inputs for computing the predicted outputs and also the variable to optimize. H, F, G, W and S are matrices formed by reorganizing the cost function and inequalities into this expression, such that $H = H' \succ 0$ is satisfied. N is the prediction horizon taken into account for the calculations. The term $\frac{1}{2}x'(t)Yx(t)$ can actually be disregarded since it has no impact on the optimization [25].

The algorithms that actually solve the mp-QP problem are based on the Karush-Kuhn-Tucker conditions (KKT) [24]:

$$\begin{aligned} HU + Fx(t) + G'\lambda = 0 \\ \lambda_i(G^iU - W^i - S^ix(t)) = 0 \\ GU \leq W + Sx(t) \\ \lambda \geq 0 \end{aligned} \tag{35}$$

in which $\lambda \in \mathbb{R}^q, \forall i = 1, ..., q$ are the Lagrange multipliers.

To compute the critical regions, an initial state vector x_0 is selected and the quadratic program is solved. Then, for the optimal solution $U(x_0)$ the active and inactive constraints are verified. For active constraints the following subset is formed

$$\tilde{G}z(x) = \tilde{S}x(t) + \tilde{W} \tag{36}$$

and for inactive the following subset is formed

$$\hat{G}z(x) \leq \hat{S}x(t) + \hat{W} \tag{37}$$

In parallel, two subvectors are formed for the Lagrange multipliers $\tilde{\lambda(x)} \geq 0$ and $\hat{\lambda(x)} = 0$ [24]. From these and with some substitutions via the KKT conditions, an expression for $z(x)$ and $\tilde{\lambda(x)}$ can be derived and by imposing constraints the polyhedral set (critical region) CR_0 can be formed

$$CR_0 = \{x \in \mathbb{R}^n : \tilde{\lambda}(x) \geq 0, \hat{G}z(x) \leq \hat{S}x(t) + \hat{W}\} \tag{38}$$

By applying different constraints more critical regions are formed for different groups of states. Therefore, a map of states to optimal control inputs is eventually created [24].

3 Methods

3.1 Materials used

To perform all the simulations and to obtain all the results for the study, it was necessary to use two laptops with Matlab and Simulink installed. Schematics from Aros Electronics aided in understanding how to proceed with the control implementations. The Multi-parametric Toolbox (MPT) and Parametric Optimization Toolbox (POP) were initially installed to compute the Explicit Model Predictive Controllers but were later discarded in favor of the Matlab's Model Predictive toolbox. Chalmers Library as well as article searching on Google provided all the required literature resources.

All simulations and control testing were performed through Matlab and Simulink with help from the supervisors involved in this study. Weekly meetings with both Aros and Chalmers supervisors were scheduled to keep track of what had been done as well as to figure out the following steps in line. Most of the consumed study time was put into several attempts at manually tuning as well as searching for analytical ways to tune and optimize the controllers.

3.2 Schematics

These are the schematics that were used to run all the simulations. In Figure 8, the general setup of the motor control is given. The two distinct reference inputs are seen on the left-hand side feeding into the controller subsystem, which delivers its output to the plant subsystem. The output of the plant subsystem is extracted from a sensor, discretizing it, and subsequently fed into the measurement subsystem, in which the sensor noise is added to the signal. Its output once again is taken back to the controller subsystem. On the right-hand side the state and references are saved as external variables and plotted together.

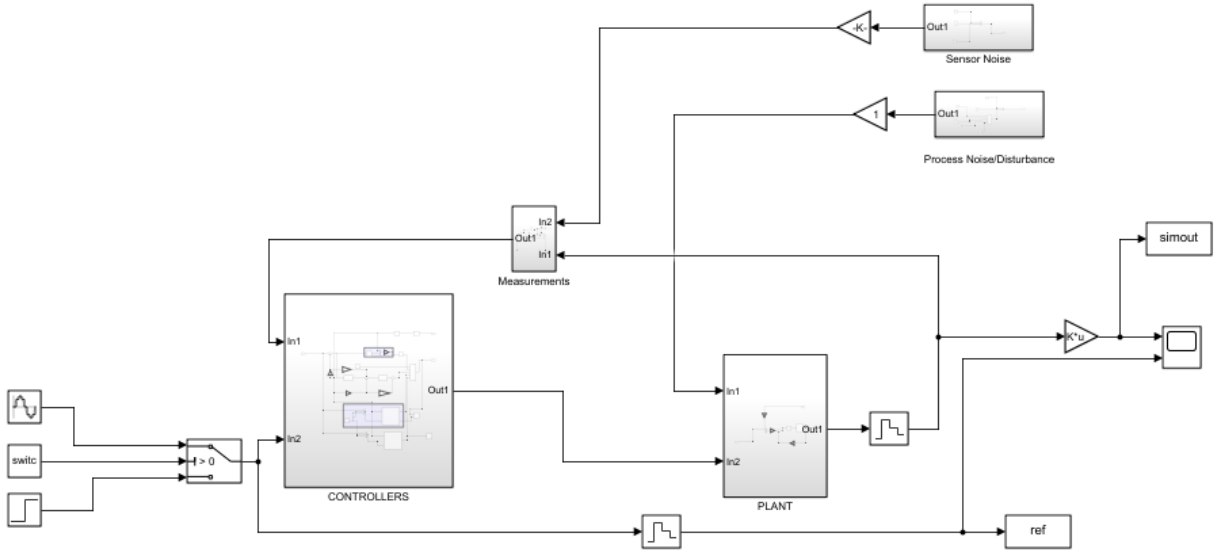


Figure 8: Setup Motor Control.

In Figure 9, the schematic for the plant nonlinearities are shown - these are included in the Plant subsystem. The cogging torque and the friction torque are added directly to the torque state so as to replicate the effects during run-time.

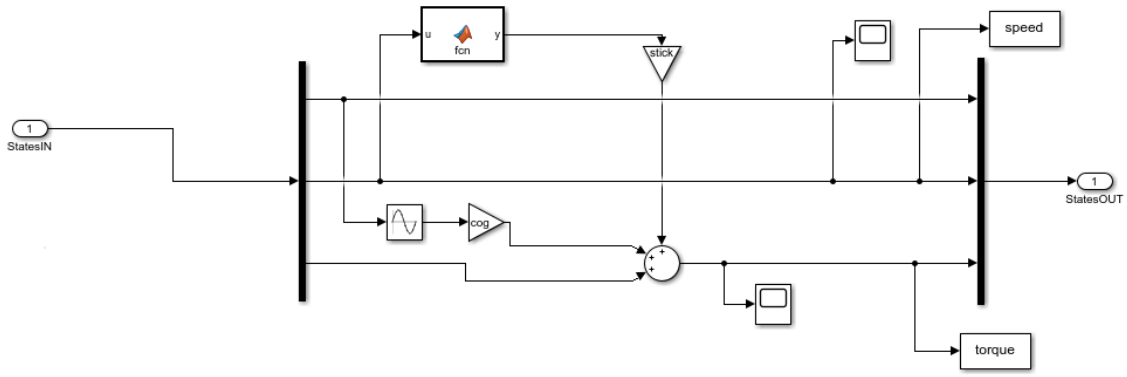


Figure 9: Nonlinearities that were considered for the simulations.

In Figure 10, the controller subsystem is shown, in which all the 4 different controllers are run in parallel and the output is toggled according to the desired output. The states that are inputs to the subsystem have an added delay to replicate delays in sensor reading.

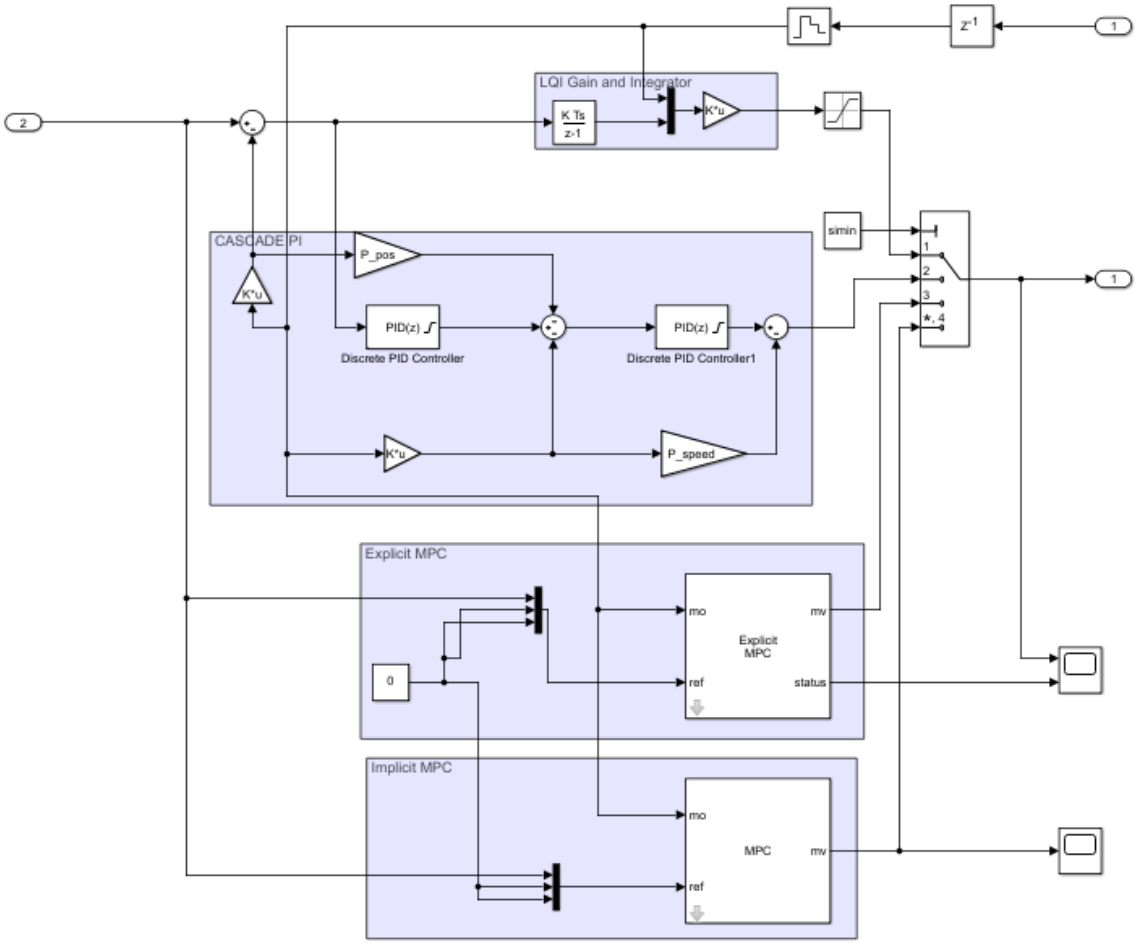


Figure 10: Controller section.

3.3 Proportional-Integral-Derivative Controller implementation

The choice of design for this thesis was to arrange and tune two PI controllers in a cascade structure as in Figure 11

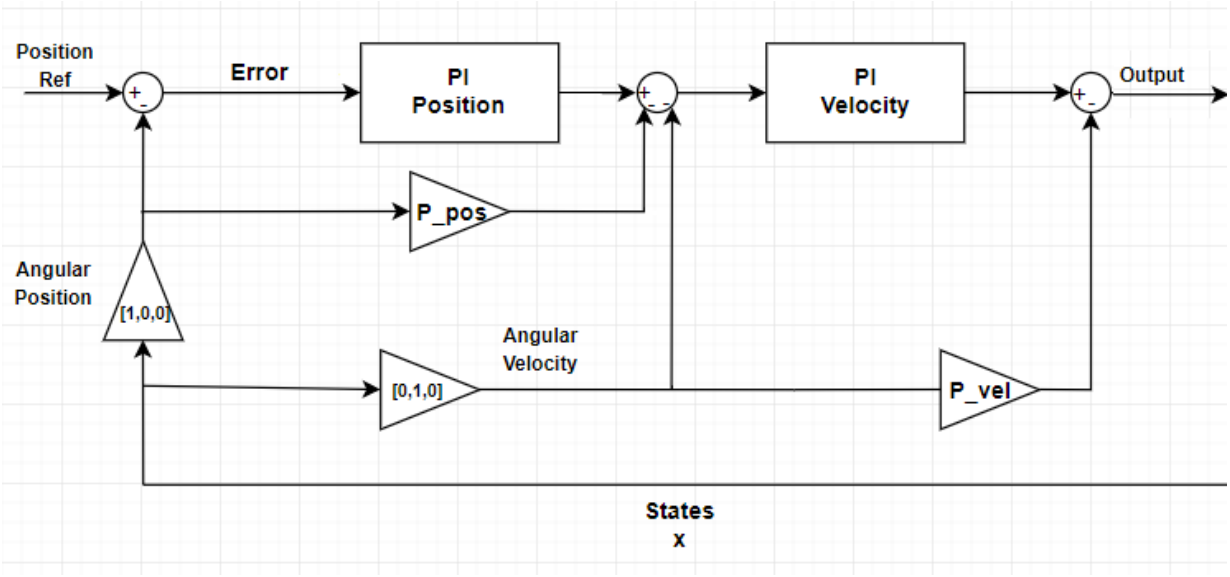


Figure 11: Cascade PI structure used in the simulations

The reasoning behind this is that typically a cascade organization of controllers is favorable for when there is a state that has a relatively slow control loop and another with a relatively fast control loop [26] - such is the case with the angular position θ_m and angular velocity ω_m, respectively.

Despite the drawback of added complexity with having to tune two PI controllers, the cascaded nature of the controller helps reduce oscillatory behavior, non-linear factors and disturbance that come into play [26].

Another design aspect that is introduced in Figure 11 is a feed-forward supplement, where the input for the second PID is the sum of the output of the first PID with the angular position measurement scaled by the position proportional component as well as with the angular velocity measurement - and the output of the second PID is summed with the angular velocity scaled with the velocity proportional component. This aims to further reduce disturbances in the states by feeding forward information about the disturbed state to the controllers [27].

Tuning was performed with the help of [28]. The position control PI was defined with gains $K_{p\theta} = \alpha_\theta$ and $K_{i\theta} = \alpha_\theta^2$ while the PI relative to the speed control had as gains $K_{p\omega} = J\alpha_\omega$ and $K_{i\omega} = J\alpha_\omega^2$ - with this relationship between the gains and some trial-and-error tuning it was possible to obtain a stable adequate response for $\alpha_\theta = 50$ and $\alpha_\omega = 200$.

3.4 Linear-Quadratic-Integral Controller implementation

The gain K_d for the LQI is directly calculated through the Matlab's Linear-Quadratic-Integral control command

$$K_d = lqi(SYS, Q, R) \tag{39}$$

which handles all the steps mentioned before. It is then just necessary to add a discrete integration block of gain 1 along with the controller sampling time T_c to the error between the position state and its reference and feed it alongside the other states to the gain block as in Figure 12.

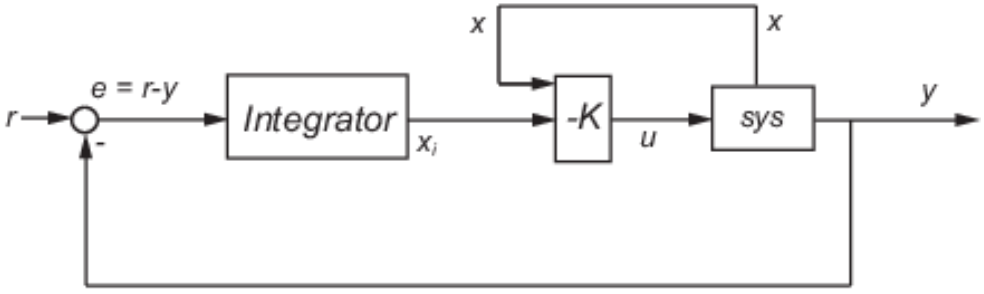


Figure 12: Control setup for LQI [21].

Tuning was performed such that there would be a relatively much higher penalty on the position θ_m state than the others, the velocity and torque were kept with very low penalty as accurate tracking was prioritized. The integration error state was given a very high penalty such as to minimize deviations in steady-state as much as possible. The plant input weight was chosen to be relatively high as well to avoid big variations in the system. The chosen weights were $Q = [1, 0.00001, 0, 10000]$ and $R = 1$.

3.5 Implicit Model Predictive Controller implementation

To create and simulate with an MPC controller both Simulink and the Matlab Model Predictive Control Toolbox were used. The reason for this is that all the code used is easily converted into C-code which can then be used for practical applications.

It is possible to create a controller via the command line and use it directly on the MPC toolbox's Simulink block. However, Matlab's MPC toolbox includes an additional software called MPC Designer, which is user-friendly and very powerful in helping construct the controller alongside constraints and added disturbances as

it provides instant visual feedback of the controller's response to inputs according to the designed controller parameters.

By correctly inserting the desired sample time, control and prediction horizons, constraints and weights the controller can be tuned according to the system's needs. Once that is accomplished, one can simply export the controller and use the MPC Simulink block to perform simulations.

The chosen value for the prediction and control Horizon for the simulations was 2, the reason being that since Implicit MPC can be a reasonably computational-heavy algorithm and the motors might have limited processing power, then it would be favorable to obtain satisfying results with the least amount of computational power necessary to avoid any big performance issues when transferring the controller to a real motor.

The chosen weights for the plant outputs (so in this case all the states $x(t)$) were $Q = [10.8, 0.027, 0]$. Since $x = [\theta_m, \omega_m, \tau_m]^T$, then this implies a high penalty on position deviation from its respective reference and low penalty on the velocity and torque. This gives higher priority to position tracking rather than conserving torque within its constraints. As for the plant input's weight (the manipulated variable, or the output of the controller) and input rate of change weight, they were chosen to have a value of 0.1 for both.

3.6 Explicit Model Predictive Controller implementation

One of the most straightforward ways to design an Explicit MPC controller is to first design an Implicit one and then convert it via Matlab's MPC toolbox functions.

One can use MPC Designer once more to obtain the desired system response and then export the controller and convert it. The prediction horizon in this case was chosen to be 10 while the control horizon was kept at 4. The reason behind this is that since an Explicit controller is pre-computed, then it is of interest to expand the amount of predicted outputs in order to reduce the error - this does, however, imply a bigger sized controller and even the pre-computed time can be significantly high if the horizon is too vast. The control horizon was left at 4 since typically it's the first control inputs that have the highest impact on the response, the others can at times be just a computational burden than actually being worth optimizing. The chosen weights considering these horizons were: $Q = [6.767, 0.027, 0]$

With the exported implicit controller, it is then necessary to generate a range of parameter bounds for the controller conversion. This is achieved by using the

command:

$$Range = generateExplicitRange(MPCobj) \tag{40}$$

Since the parameters are mostly already constrained, the range values were set to be rather high - the states x have bounds of $[-1000, 1000]$, the reference has bounds of $[-100, 100]$ and the manipulated variables have as bounds $[-10, 10]$. This was to ensure that the explicit controller worked as it should, since by restricting the bounds close to constraint values caused unstable behavior. The explicit controller is obtained by using the command:

$$EMPCobj = generateExplicitMPC(MPCobj, range) \tag{41}$$

3.7 Simulation Execution

The controllers are tested for multiple different scenarios to assess how well they perform under different stimuli and how comparatively robust they are. It's important to mention that the controllers were designed to deliver no overshoot and as fast settling to the reference (while minimizing the steady-state error) as much as possible, these were the highest prioritized features. The two main situations that are analyzed are reference tracking and disturbance rejection.

Reference tracking involves receiving an input trajectory that the desired plant output variable (in this case the angular position θ_m) should approximate as much as possible. The two types of inputs that are considered are the step function and the sine wave.

The step function is useful to clearly evaluate how fast the controller adapts to a sudden change as well as how fluidly it stabilizes to that new fixed value. On the other hand the sine wave is interesting to use as an input due to its oscillatory nature, since in some motor applications the reference values are constantly varying - it is then practical to observe how closely the controller follows the variations. The time constant τ_c is also measured to give an idea of the system bandwidth for that controller, or, in other words, to get an idea of at beyond which frequency does the system start to decay if the input's frequency changes. The bandwidth is then calculated by calculating the cutoff frequency f_{3db}, the frequency at which the magnitude of the steady-state response drops by half or $-3db$, it is then $f_{3db} = \frac{1}{2\pi\tau_c}$

The variables that are considered for the reference tracking are: rise time (step) - how long it takes for the controller value to go from 10% to 90% of the reference value, settling time (step) - the time it takes for the controller to stabilize to values such that $|\theta_m - \theta_{ref}| < 0.0005$, max torque - highest amount of used torque recorded during execution, max error - highest amount of instant error recorded , total error -

accumulated value of error over the entire execution, max speed - maximum amount of angular velocity ω_m during execution and torque RMS - Root Mean Square value of the torque for power expenditure purposes.

Regarding disturbance rejection, the controllers are tested for both sensor noise rejection as well as load torque rejection. The former concerns noise that is presented as the output is extracted from the sensor, so it's considered to be output noise. The latter concerns a step Load torque which is added directly to the plant as an external disturbance,

The variables that are tested for the disturbance rejection are: settling time (load torque), max torque, max speed, total error, max error and torque RMS. These two scenarios are then repeated with the insertion of the cogging Torque and stick friction non-linearities to analyze how they impact the controllers' performances.

Robust testing is also performed to evaluate how the controllers perform under different values of inertia J and different delay values when reading from the sensors.

4 Results

4.1 Controller Setup

For all the following simulations the controller setup parameters are listed below.

The proportional and integral gains used for these simulations are as follows:

$$P_\theta = 50$$

$$I_\theta = 2500$$

$$P_\omega = 0.007$$

$$I_\omega = 1.4$$

The weights used for the LQI controller, which are - by order - respective to the angular position θ_m, the angular velocity ω_m, the torque τ_m, the integral state of the position error and the output of the controller (input to the plant) u are as follows:

$$Q_1 = 1$$

$$Q_2 = 0.00001$$

$$Q_3 = 0$$

$$Q_4 = 10000$$

$$R = 1$$

The Implicit MPC's prediction and control horizon were set to 2 for the simulations. The weights for the parameters - ordered as: Manipulated Variable u (output of controller), Rate of change of u, angular position, angular velocity and torque - are defined as:

$$MV = 0.00135$$

$$Rate = 0.739$$

$$Q_1 = 10.8$$

$$Q_2 = 0.027$$

$$Q_3 = 0$$

The Explicit MPC's prediction horizon was set to 10 and the control horizon was set to 4 for the simulations. The resulting size of the controller was 1.6 MB. The weights for the parameters - ordered as: Manipulated Variable u (output of controller), rate of change of u, angular position, angular velocity and torque - are defined as:

$$MV = 0.135$$

$$Rate = 0.739$$

$$Q_1 = 6.767$$

$$Q_2 = 0.027$$

$$Q_3 = 0$$

4.2 Reference Tracking

The tracking of a reference regarding angular position θ_m for a step input of amplitude 2 radians ($rads$) and a sine wave input of amplitude 1 rad is shown below for all the controllers. The time constants τ_c were measured for each response and they are $\tau_{c-PI} = 0.01783$, $\tau_{c-LQI} = 0.02124$, $\tau_{c-MPC} = 0.01246$ and $\tau_{c-EMPC} = 0.01075$ for the Cascade PI, LQI, Implicit MPC and Explicit MPC, respectively. Therefore, their respective system's bandwidths are $f_{c-PI} = 8.926Hz$, $f_{c-LQI} = 7.493Hz$, $f_{c-MPC} = 12.77Hz$ and $f_{c-EMPC} = 14.8Hz$.

For Cascade PI, the step input plots for the angular position θ_m response, mechanical torque τ_m and angular velocity ω_m are shown in Figures 13, 14a and 14b. The sine wave input plot for the angular position θ_m response is given in Figure 15.

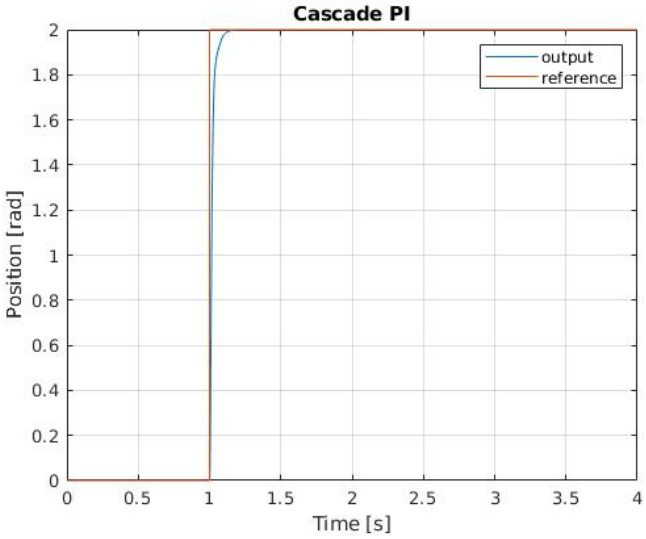
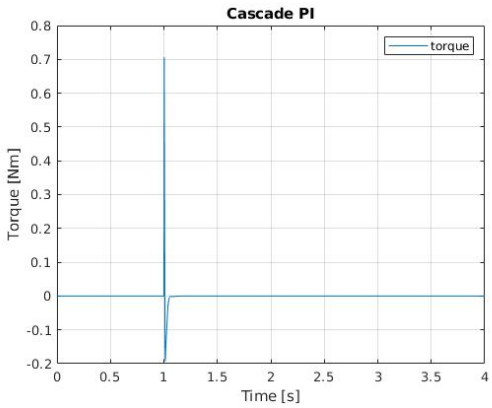
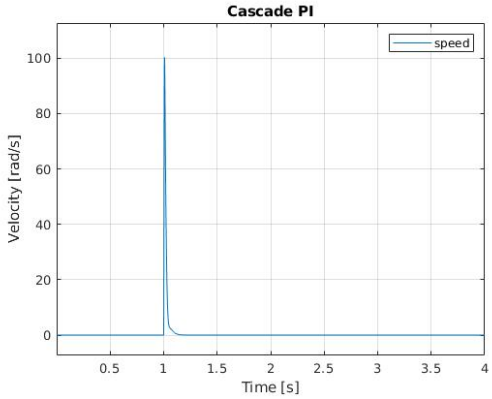


Figure 13: Step input reference tracking θ_m for Cascade PI.



(a) Step Reference tracking τ_m.



(b) Step Reference tracking ω_m.

Figure 14: Velocity and Torque responses.

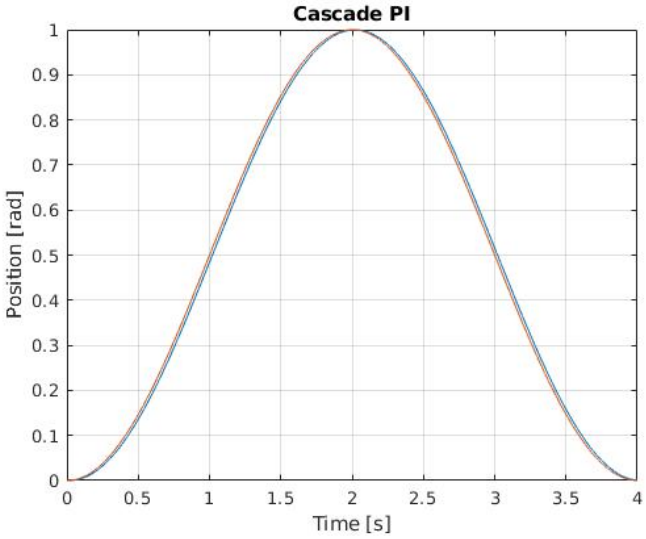


Figure 15: Sine wave input reference tracking θ_m for Cascade PI - one period.

For LQI, the step input plots for the angular position θ_m response, mechanical torque τ_m and angular velocity ω_m are shown in Figures 16, 17a and 17b. The sine wave input plot for the angular position θ_m response is given in Figure 18.

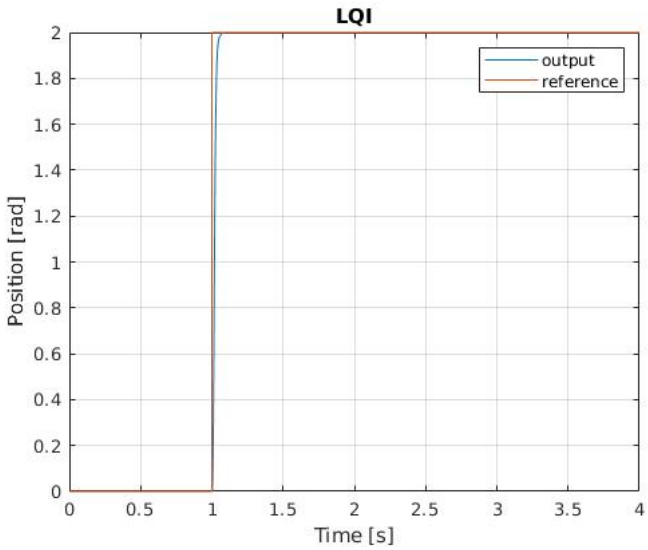
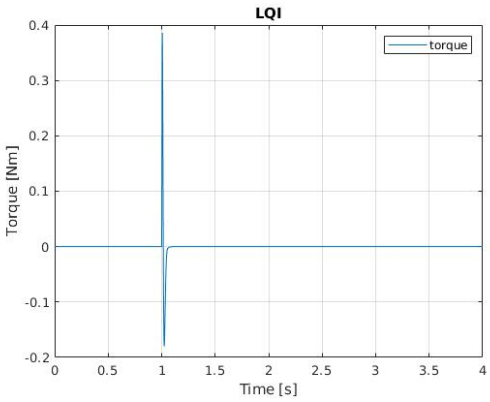
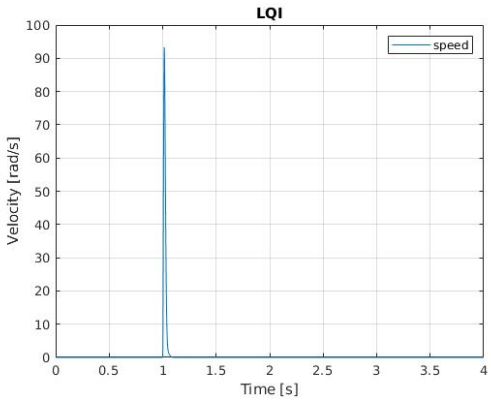


Figure 16: Step input reference tracking θ_m for LQI.



(a) Step Reference tracking τ_m.



(b) Step Reference tracking ω_m.

Figure 17: Velocity and Torque responses.

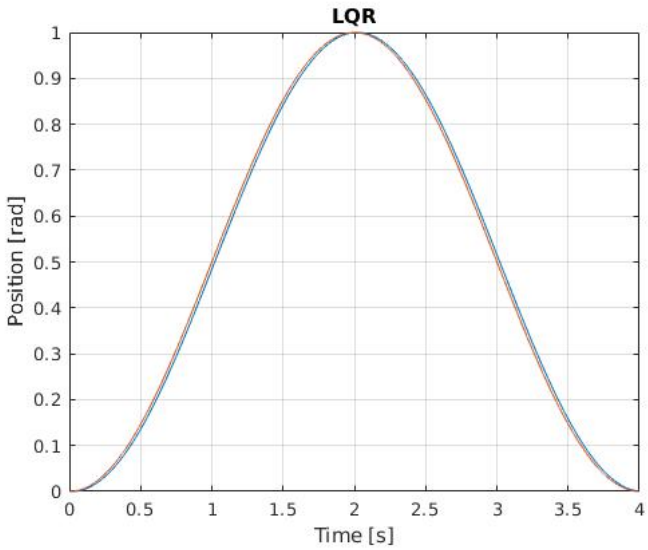


Figure 18: Sine wave input reference tracking θ_m for LQI.

For Implicit MPC, the step input plots for the angular position θ_m response, mechanical torque τ_m and angular velocity ω_m are shown in Figures 19, 20a and 20b. The sine wave input plot for the angular position θ_m response is given in Figure 21.

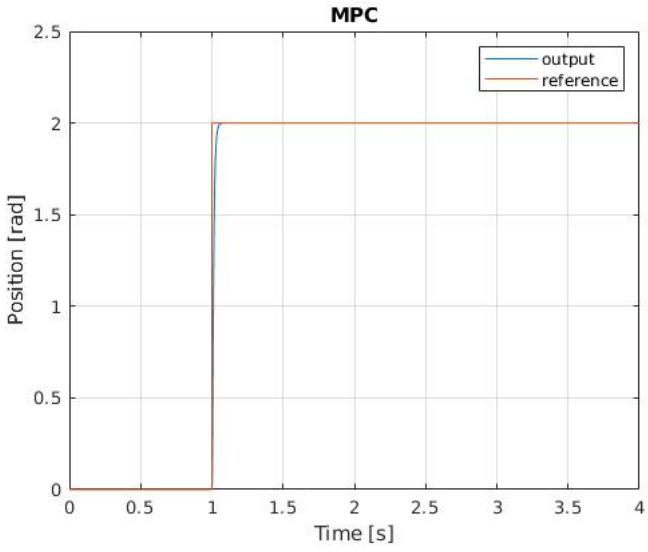
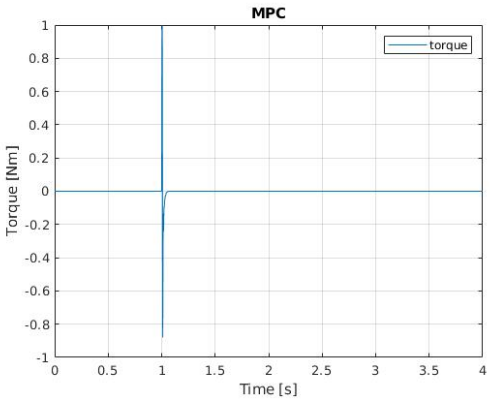
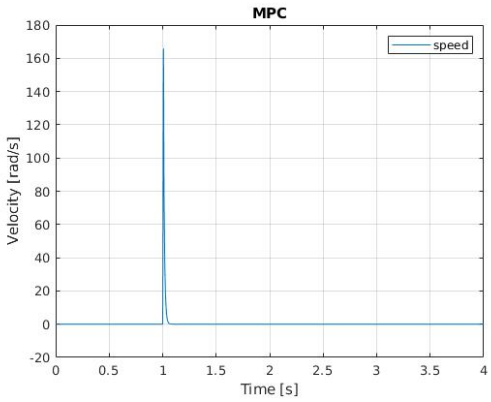


Figure 19: Step input reference tracking θ_m for Implicit MPC.



(a) Step input reference tracking τ_m.



(b) Step input reference tracking ω_m.

Figure 20: Velocity and Torque responses.

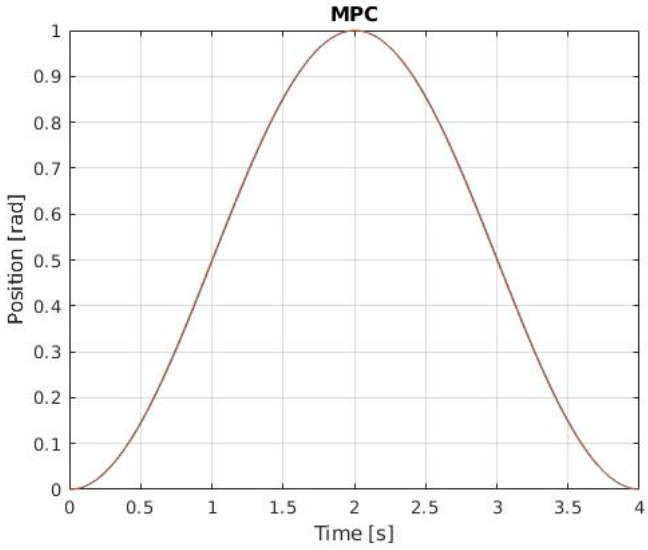


Figure 21: Sine wave input reference tracking θ_m for Implicit MPC.

For Explicit MPC, the step input plots for the angular position θ_m response, mechanical torque τ_m and angular velocity ω_m are shown in Figures 22, 23a and 23b. The sine wave input plot for the angular position θ_m response is given in Figure 24.

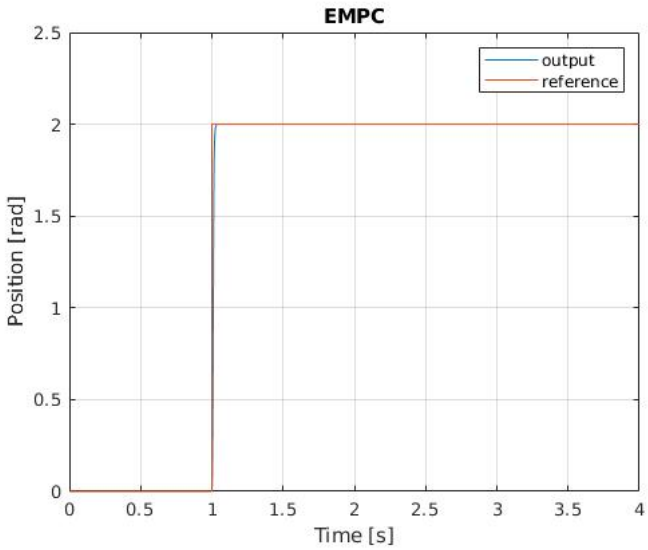
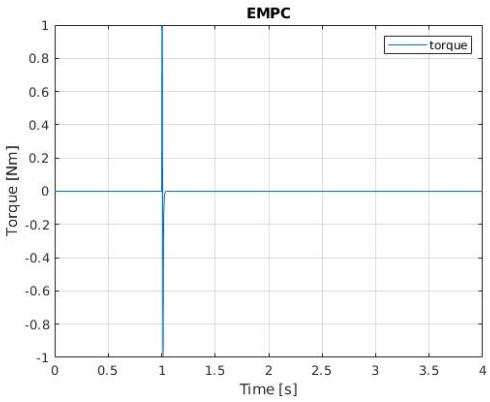
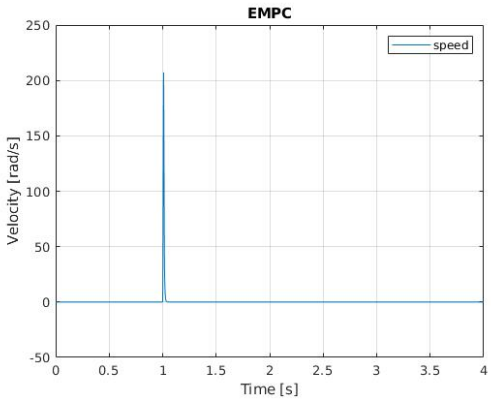


Figure 22: Step input reference tracking θ_m for Explicit MPC.



(a) Step input reference tracking τ_m.



(b) Step input reference tracking ω_m.

Figure 23: Velocity and Torque responses.

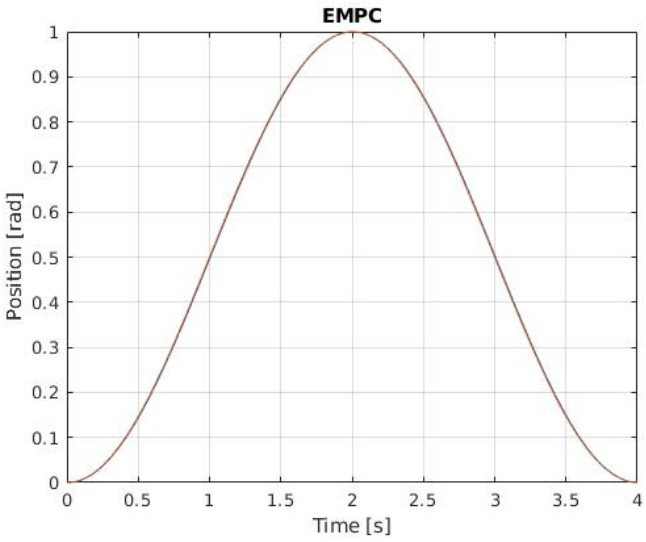


Figure 24: Sine wave input reference tracking θ_m for Explicit MPC.

4.3 Step Load Torque Disturbance Rejection

The rejection response for an added load torque, represented by a step function of amplitude 0.5 $N \cdot m$ while keeping all state references to zero is presented in the following plots.

For Cascade PI, the angular position response is shown in Figure 25. The respective torque and velocity responses are shown in Figures 26a and 26b.

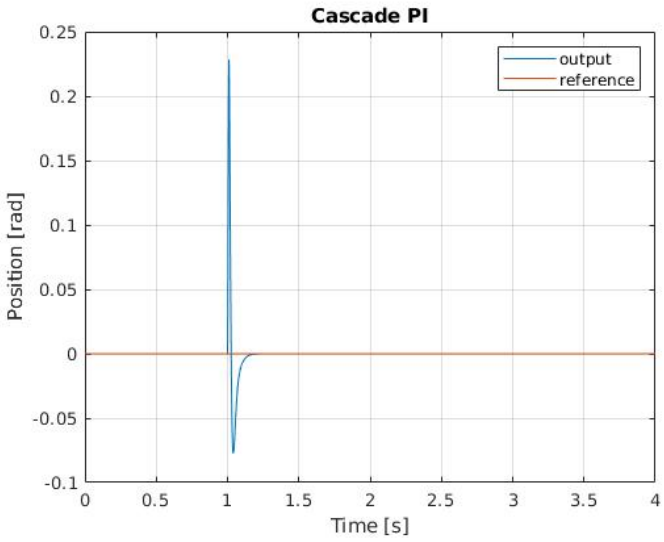
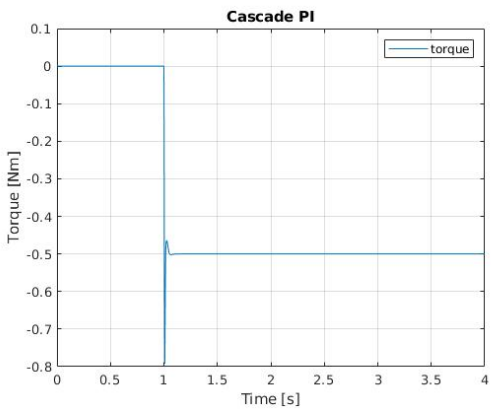
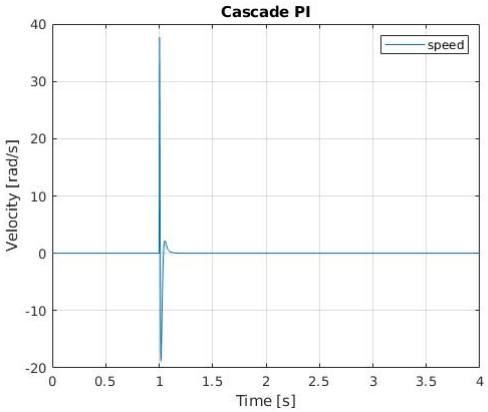


Figure 25: Step Load Torque Disturbance θ_m for Cascade PI.



(a) Step Load Torque Disturbance τ_m.



(b) Step Load Torque Disturbance ω_m.

Figure 26: Velocity and Torque responses.

For Cascade PI, the angular position response is shown in Figure 27. The respective torque and velocity responses are shown in Figures 28a and 28b.

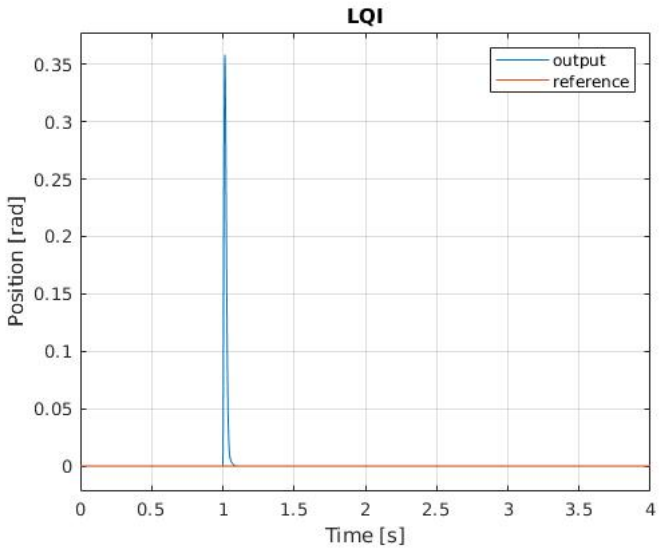
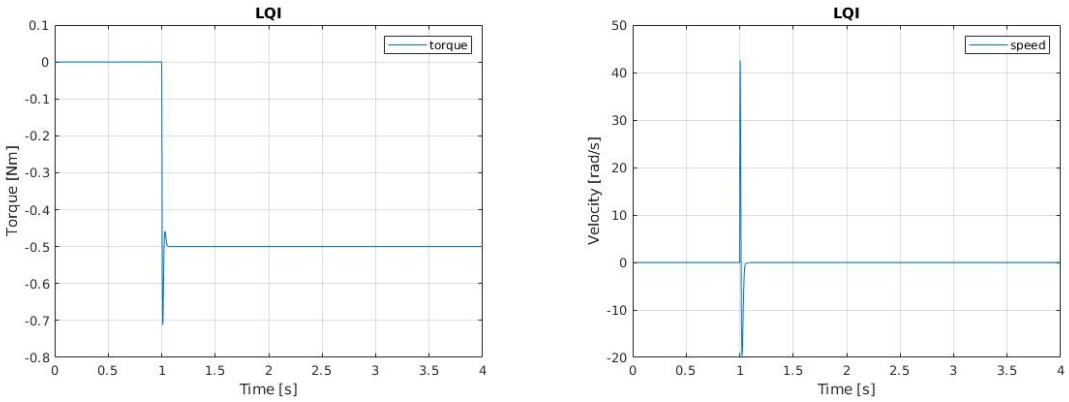


Figure 27: Step Load Torque Disturbance θ_m for LQI.



(a) Step Load Torque Disturbance τ_m. (b) Step Load Torque Disturbance ω_m.

Figure 28: Velocity and Torque responses.

For Implicit MPC, the angular position response is shown in Figure 29. The respective torque and velocity responses are shown in Figures 30a and 30b.

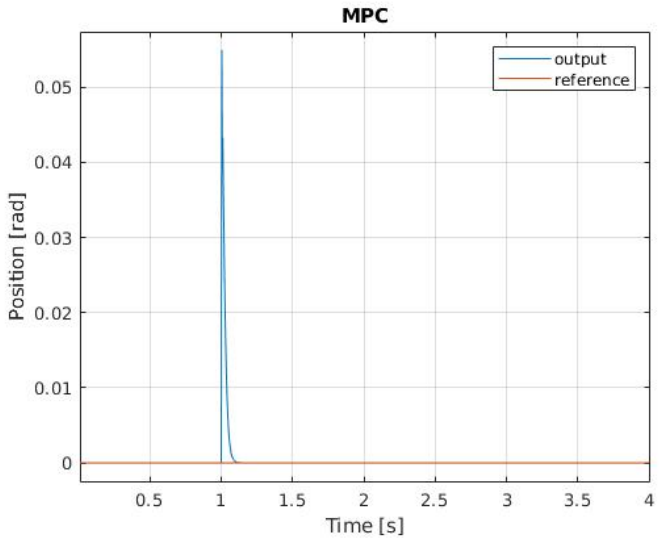
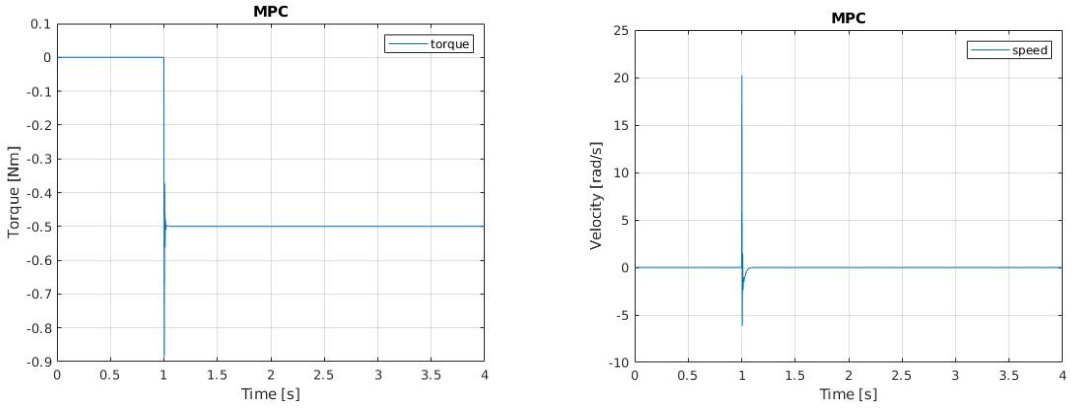


Figure 29: Step Load Torque Disturbance θ_m for Implicit MPC.



(a) Step Load Torque Disturbance τ_m. (b) Step Load Torque Disturbance ω_m.

Figure 30: Velocity and Torque responses.

For Explicit MPC, the angular position response is shown in Figure 31. The respective torque and velocity responses are shown in Figures 32a and 32b.

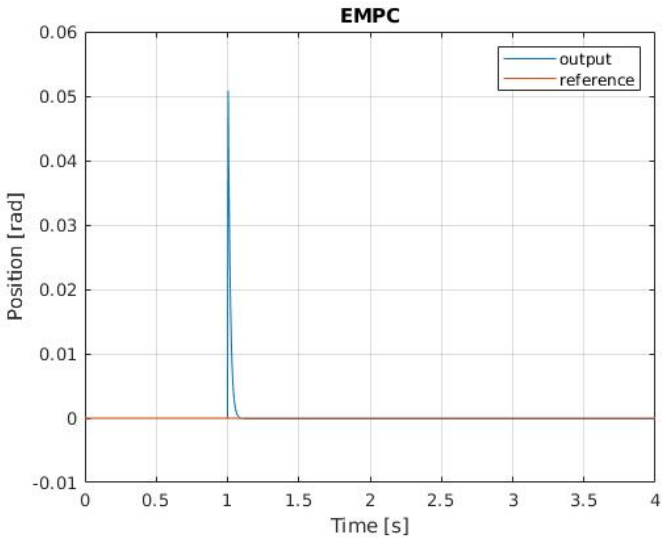
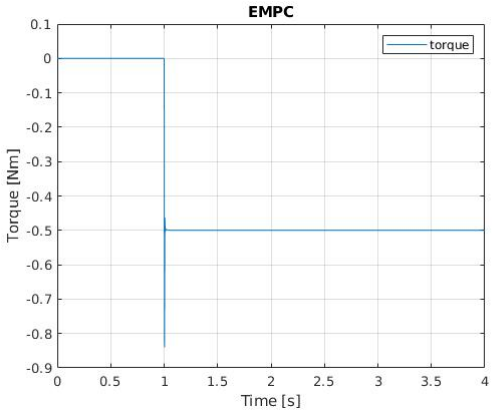
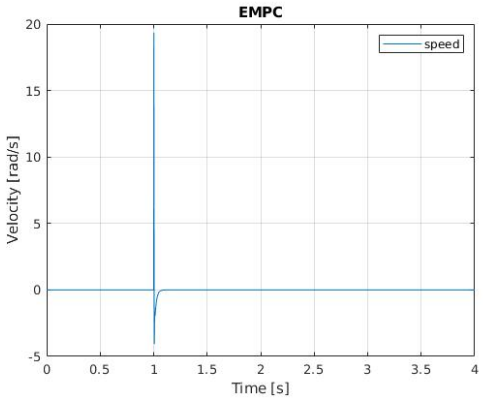


Figure 31: Step Load Torque Disturbance θ_m for Explicit MPC.



(a) Step Load Torque Disturbance τ_m.



(b) Step Load Torque Disturbanceω_m.

Figure 32: Velocity and Torque responses.

4.4 Reference Tracking and Disturbance Rejection simulation result values

The tested parameters for each controller that were obtained from the simulation results are found in the tables below. Max error is ignored for the step response as, since there is no overshoot, the maximum error would be at the instant when the step switches from 0 to 1. Max torque, max speed, torque RMS, settling time and

rise time are neglected for the Sine case since they're either not applicable or the variations in torque and speed are too small to be of interest (due to the gradually varying nature of the input).

For the Cascade PI, the results are shown in Table 1:

Cascade PI Results							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	0.707	99.703	389	0.0239	0.206	0.03
Sine	0.0156	X	X	388.75	X	X	X
Load Dist. Rej.	0.227	0.788	37.483	61.907	0.434	0.178	X

Table 1: Cascade PI simulation results.

For LQI, the results are shown in Table 2:

LQI Results							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	0.387	92.79	394.17	0.0184	0.09	0.024
Sine	0.0158	X	X	394	X	X	X
Load Dist. Rej.	0.357	0.707	42.24	76.71	0.434	0.081	X

Table 2: LQI simulation results.

For Implicit MPC, the results are shown in Table 3:

MPC Results							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	0.997	165.04	259.86	0.0432	0.078	0.02
Sine	0.0084	X	X	205	X	X	X
Load Dist. Rej.	0.0547	0.882	20.149	12.478	0.4352	0.083	X

Table 3: MPC simulation results.

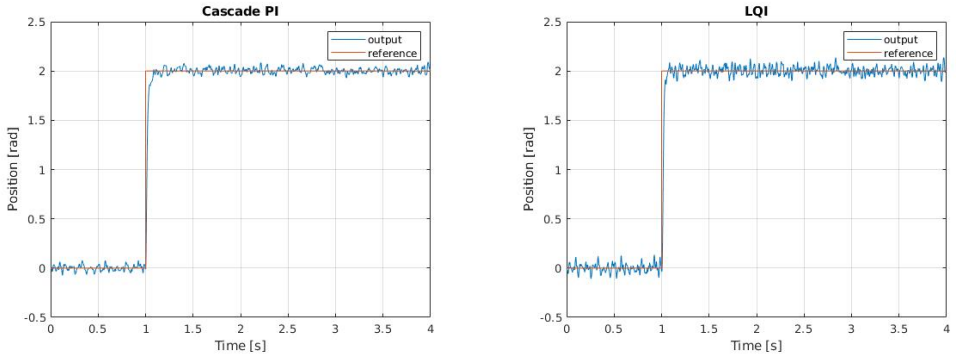
For Explicit MPC, the results are shown in Table 4:

Explicit MPC Results							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	0.999	206.97	200.9	0.0541	0.039	0.011
Sine	0.005	X	X	119	X	X	X
Load Dist. Rej.	0.0506	0.839	19.299	11.933	0.435	0.068	X

Table 4: EMPC simulation results.

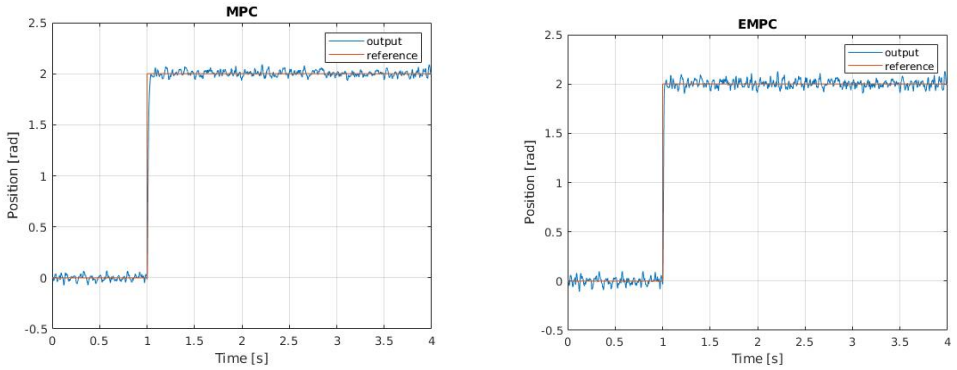
4.5 Sensor Noise Disturbance Rejection

White noise was simulated by using Simulink's Band-Limited White Noise block, characterized by a noise power of 0.0001, added to all the state readings. This was added during reference tracking to observe how the controllers adapt to change despite the sensor noise, the angular position responses are shown in Figures 33 and 34.



(a) Noise PI. (b) Noise LQI.

Figure 33: Simulation results for PI and LQI.



(a) Noise MPC. (b) Noise Explicit MPC.

Figure 34: Simulation of results for MPC and EMPC.

Noise disturbance				
Measurement	PI	LQI	MPC	EMPC
Max Error	2.0235	2.1748	2.1295	2.0111
Total Error	2496.8	6097.1	3175.5	1491.3

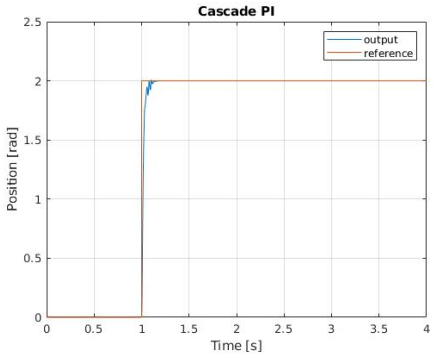
Table 5: Noise Disturbance simulation results.

4.6 Nonlinearities

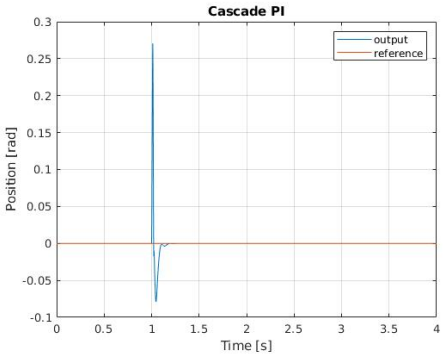
4.6.1 Cogging Torque and Stick-slip Friction

The same scenarios of 2 $rads$ step reference tracking and 0.5 $N \cdot m$ step load torque rejection are repeated but with the addition of the cogging torque and stick-slip friction.

For Cascade PI, the angular position response plots are shown in Figure 35 and the values are presented in Table 6. Reference tracking maintains a similar behaviour when compared to its linear counterpart, however, disturbance rejection causes oscillations as the controller attempts to drive the signal back to the reference.



(a) Cascade PI Reference Tracking with nonlinearities.



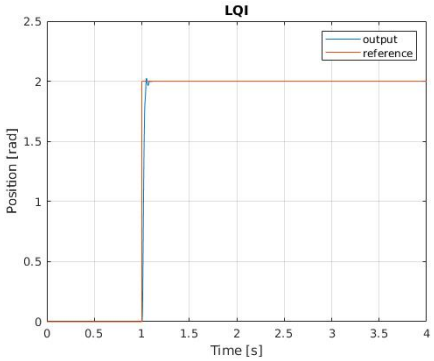
(b) Cascade PI Disturbance Rejection with nonlinearities.

Figure 35: Simulation results for PI with nonlinearities.

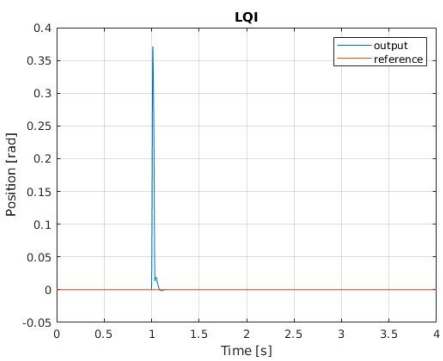
Cascade PI with Nonlinearities							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	0.940	112.21	389.6	0.0384	0.19	0.034
Load Dist. Rej.	0.271	0.917	46.982	61.903	0.435	0.176	X

Table 6: Cascade PI simulation results w/ nonlinearities.

For LQI, the angular position response plots are shown in Figure 36a nd the values are presented in Table 7. Similarly to Cascade PI, its reference tracking performance is comparatively close to the linear one, however, disturbance rejection causes oscillations as the controller attempts to drive the signal back to the reference.



(a) LQI Reference Tracking with nonlinearities.



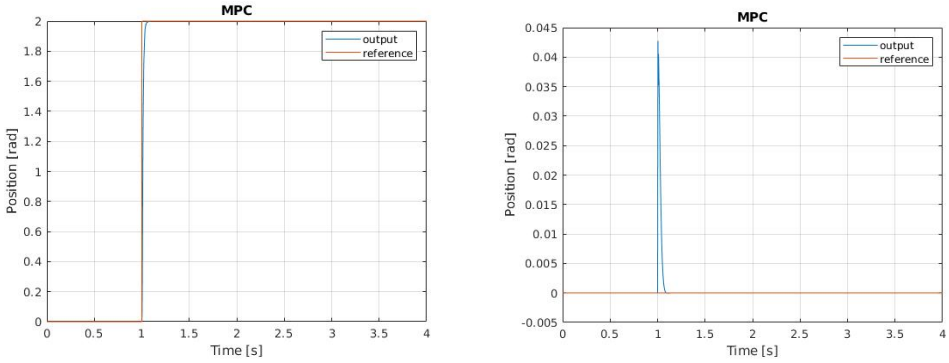
(b) LQI Disturbance Rejection with nonlinearities.

Figure 36: Simulation results for LQI with nonlinearities.

LQI with Nonlinearities							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	0.761	103.38	397.52	0.0325	0.114	0.022
Load Dist. Rej.	0.371	1.0571	49.795	77.806	0.435	0.125	X

Table 7: LQI simulation results w/ nonlinearities.

For Implicit MPC, the angular position response plots are shown in Figure 37a nd the values are presented in Table 8.



(a) MPC Reference Tracking with nonlinearities.

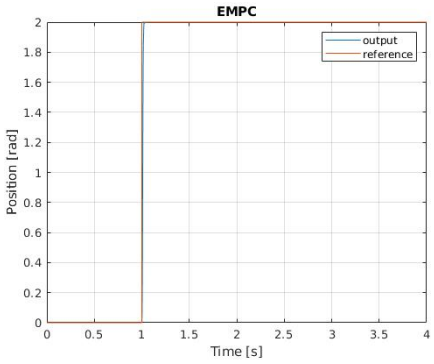
(b) MPC Disturbance Rejection with nonlinearities.

Figure 37: Simulation results for MPC with nonlinearities.

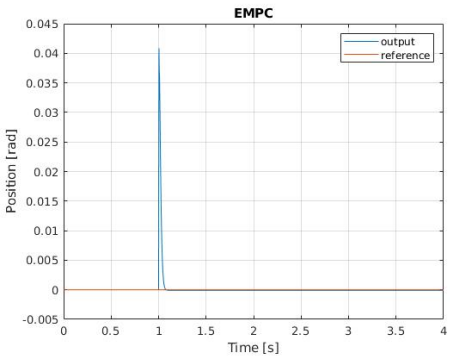
MPC with Nonlinearities							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	1.492	168.69	260.77	0.0526	0.098	0.021
Load Dist. Rej.	0.0428	0.899	18.005	12.093	0.436	0.074	X

Table 8: MPC simulation results w/ nonlinearities.

For Explicit MPC, the angular position response plots are shown in Figure 38a nd the values are presented in Table 9.



(a) EMPC Reference Tracking with nonlinearities.



(b) EMPC Disturbance Rejection with nonlinearities.

Figure 38: Simulation results for EMPC with nonlinearities.

EMPC with Nonlinearities							
TEST CASE	Max Error	Max Torque	Max Speed	Total Error	Torque RMS	Settling Time	Rise Time
Step	X	1.492	207.45	199.95	0.0588	0.067	0.011
Load Dist. Rej.	0.0408	0.867	17.492	11.923	0.436	0.061	X

Table 9: EMPC simulation results w/ nonlinearities.

4.7 Robust Testing

4.7.1 Inertia J

For an inertia J value ranging from $[0.5, 7.5] \cdot 10^{-5}\ kgm^2$, the controllers were simulated with the same 2 $rads$ step reference tracking and their total errors, rise time and settling time were compared with each other as the inertia was increased. Figures 39 and 42 show to what values the different controllers tend towards with this variation.

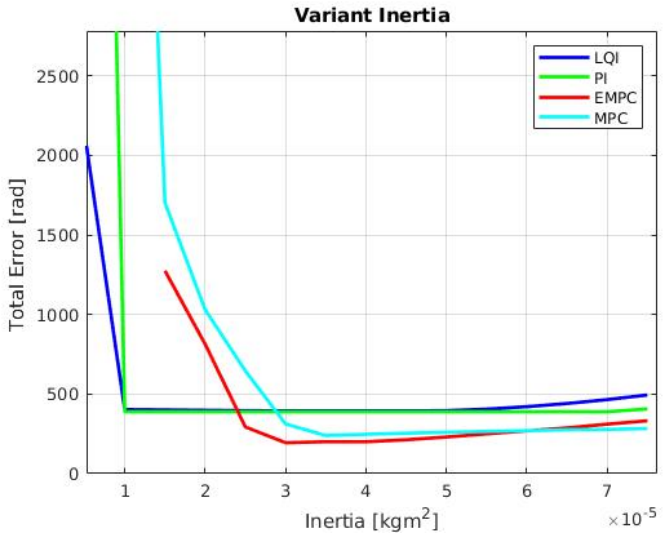
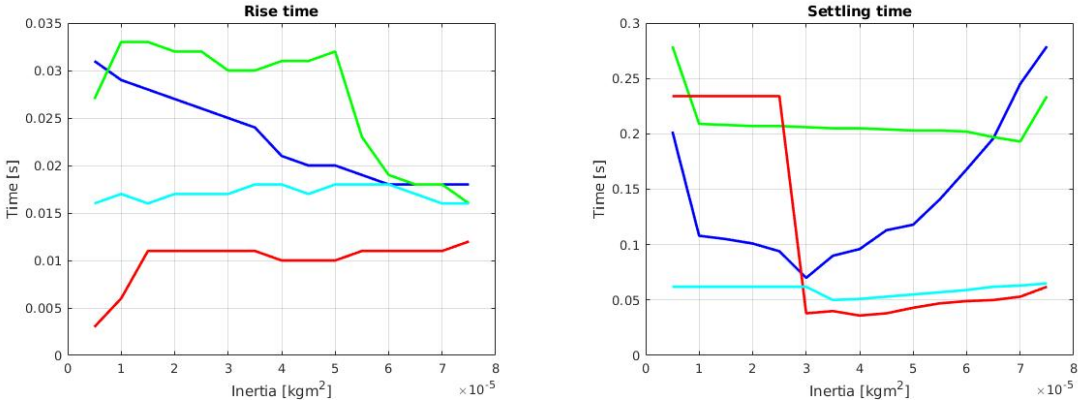


Figure 39: Total Error for different values of inertia.



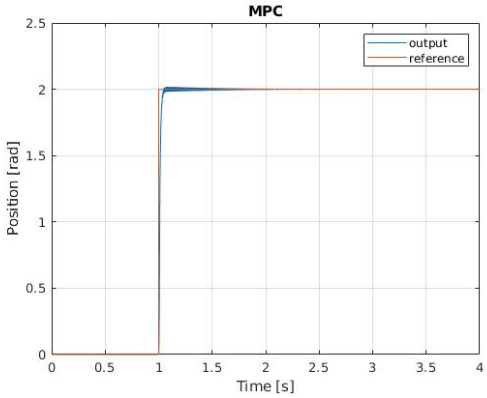
(a) Rise Time for different values of inertia.

(b) Settling Time for different values of inertia.

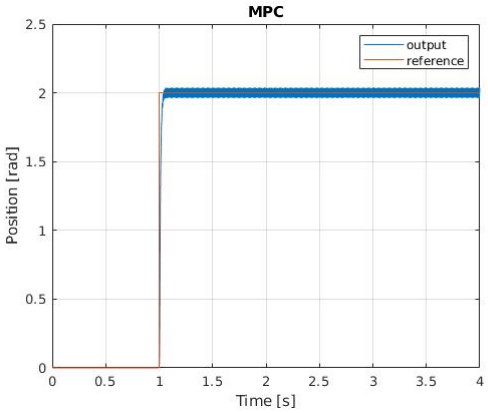
Figure 40: Simulation results for different values of inertia.

4.7.2 Delays

For delay values ranging from $[0, 1]$ ms, the controllers maintained functionality that was approximately very similar to their normal behaviour. The only two cases that differed were Implicit and Explicit MPC. Implicit MPC showed stable behaviour up until a delay of $0.7ms$, while Explicit MPC maintained stability up until a delay of $1ms$. Since the sensor's sampling time of the plant process is $T_p = 0.1ms$, then this corresponds to 7 and 10 samples of delay respectively.

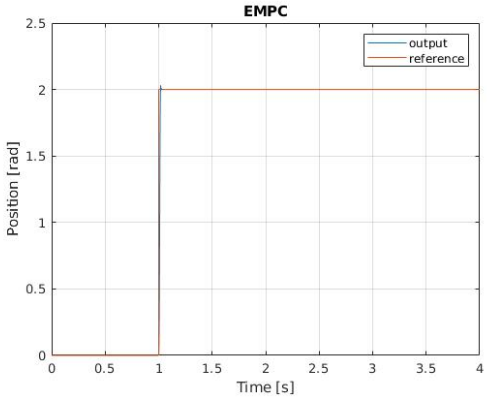


(a) Reference Tracking response for a delay of 0.5 ms.

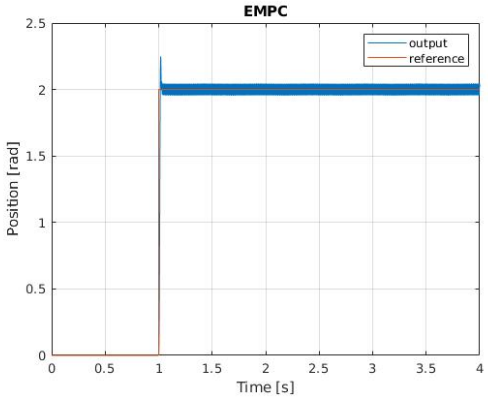


(b) Reference Tracking response for a delay of 1 ms.

Figure 41: Simulation results for different values of delay Implicit MPC.



(a) Reference Tracking response for a delay of 0.5 ms.



(b) Reference Tracking response for a delay of 0.9 ms.

Figure 42: Simulation results for different values of delay for Explicit MPC.

5 Discussion

Starting off by checking the case of no added non-linearities, one can make some observations about the differences in performance between the controllers that stand out the most.

Firstly, when it comes to total error, the Explicit MPC controller is dominant as on average it outperformed the others in the reference tracking and disturbance rejection scenarios. Implicit MPC was second best, followed by Cascade PI and LQI. It was expected that the model predictive controllers would be better at minimizing the amount of error due to the fact that they do not utilize just a simple cost function or a set of gains to correct deviations, the algorithms behind them are more complex - taking more parameters into account - and they compute the most optimal path for predicted possible trajectories. Despite that LQI is faster at converging to the steady-state than Cascade PI, the latter is faster at reacting to the change in the reference initially, which leads to less total error by a small margin. This is due to the fact that if both controllers start with states that are initially zero, Cascade PI has a proportional component that will act directly on the change of input (error) thus giving a response with practically no delay, on the other hand LQI depends on the integrator to output a different value so it is delayed by a controller sample time T_c at the start. Interestingly, besides LQI having a lower settling time than Cascade PI it actually has a higher time constant τ_c, this can be attributed to its very high weight on the integration state which helps it reach the setpoint earlier than Cascade PI.

In contrast to this observation, LQI had the lowest values of torque RMS and its max torque was lower than the others. It is therefore less taxing power-wise. This is due to the tuning being limited to avoid overshoot as much as possible and perhaps an alternative structure for the LQI would have been more appropriate (e.g. Cascaded), as the weights that were used were the ones that were seemingly best despite its relative slowness as to what it could be. In other words, attempts at increasing its speed, which would imply more torque usage, led to breaches in terms of overshoot. As for settling time, Explicit MPC leads in performance followed closely by Implicit MPC while Cascade PI and LQI lag a bit behind - the reasoning supporting it is the same as in the total error comparison.

Regarding sensor noise disturbance, Explicit MPC ranks once more at the top, showing the lowest amount of total error followed by Cascade PI, Implicit MPC and LQI. The variations in handling the noise were not as significant to analyze as in the other scenarios but it's interesting to see that Cascade PI is one of the controllers that handles the noise the best. This is likely due to the feedforward mechanism that was implemented, which aims to lessen the impact of noise overall.

In the added nonlinearities scenario, Explicit MPC once again ranks first, followed by Implicit MPC, Cascade PI and LQI. The controllers all function well regardless of the nonlinearities, with some observable slight changes in performance. The biggest highlight of this scenario is the torque usage given the nonlinearities. All controllers had an increase in torque RMS, which is natural since there are added forces that have to be compensated for to meet the setpoint. Curiously, the max torque on the model predictive controllers (which is taken from the plant via sensors) exceeded in value when compared to the constraint limits of the input torque to the plant, which are in effect and in fact do restrict the controllers' outputs. This is likely because of the magnets' pulling effect from the cogging torque.

Finally, regarding robust testing relative to varying inertia J, LQI is the most robust followed closely by Cascade PI. Implicit MPC shows a lot of error for the lowest values of inertia and Explicit MPC does not even work for a certain interval of initial inertia values even though it shows the lowest registered total error within its functional range - this is most definitely due to its pre-computed nature which assumes that a fixed state-space will be kept on run-time. When it comes to delays, Implicit MPC was the least robust followed then by Explicit MPC - this would lead one to believe that their predictive nature causes them to be very reliant on accurate responses time-wise.

The controllers each have their own pros and cons, there is not one that can be considered the outright best. Cascade PI is easy to implement, can provide fairly good results and is quite robust as far as delays and changing inertia is concerned. LQI is also fairly easy to implement and is also at the top of robustness, it excelled in torque usage but suffered somewhat in terms of actual performance relative to the others. Explicit MPC had the best overall performance, followed closely by Implicit MPC - however, despite being the best at handling Step Torque Load disturbance they both lacked in robustness when compared to the other two controllers.

It is worth mentioning that the weights on the plant input or controller output of the model predictive controllers is highly correlated to how much delay they can handle versus how well they perform with disturbances. The higher one increases the weight, the worse the performance is with added disturbances and vice-versa with the delays. It was opted to favor better performance with added disturbances.

Another important comment is to mention how increasing the prediction and control horizons on the Implicit MPC could further improve its performance, these were kept low due to the unknown amount of available computational power. LQI could have also potentially improved in performance with more exploration in the tuning.

6 Conclusion

This thesis was meant to cover the comparison between different controllers for the same plant and to determine which of them had the most appealing features for the applications that Aros Electronics has in store for the PMSM motor. As stated previously, it was desirable to design controllers which provided responses with no overshoot, negligible steady-state error and as rapid settling to the set-point as possible.

From the simulations, one can draw some important findings. Model predictive controllers appear to have a significantly impressive performance for the system at hand. The Explicit MPC has the objectively best results but it's interesting to note that the Implicit MPC was not far behind despite a cut on its prediction and control horizons. Although they had worse results, Cascade PI and LQI are still very reliable options and have their own strengths.

If there is enough available computational power to integrate an Implicit MPC in a motorized system, then it appears to be the most appealing option. It would not require additional memory as its Explicit counterpart and, if there are enough computational resources, the performance can still be further increased beyond what was observed. If computational power is a bottleneck resource and memory is not a problem, then Explicit MPC is favored.

This thesis was limited to and focused on the mechanical section of the motor, to concentrate on achieving the best results possible while under the assumption that the current control of the electrical section is satisfyingly functional. It would be interesting to observe the controllers' behaviors on the actual real motors and compare them to the simulated environments and see if the observations made would still hold. It would also be worth looking into modelling the full motor system with included current control despite the added complexity and nonlinear aspects - possibly a topic to dive into in future projects.

Bibliography

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

[2] B.Pradeepa. R.Kiruthiga. P.B.Nevetha. H.Kala. S.Abirami. P.Sujithra. "Performance Comparison of Different Controllers for Flow Process". In: *International Journal of Computer Applications* 90 (). DOI: https://pdfs.semanticscholar.org/8fcc/0f4a72b3664504e4c57fa1eb1489dfb64624.pdf.

[3] Gordon R. Slemon. "Electric motor". In: *Encyclopaedia Brittanica* (2018). URL: https://www.britannica.com/technology/electric-motor/Synchronous-motors.

[4] Hampus Isaksson. Patrik Önnheim. "High Precision Positioning and Very Low Velocity Control of a Permanent Magnet Synchronous Motor". MA thesis. Department of Automatic Control, Lund University, 2015.

[5] Janet Heath. *PWM: Pulse Width Modulation: What is it and how does it work?* Apr. 2017. URL: https://www.analogictips.com/pulse-width-modulation-pwm/.

[6] Aswinth Raj. *What is PWM: Pulse Width Modulation*. Sept. 2018. URL: https://circuitdigest.com/tutorial/what-is-pwm-pulse-width-modulation.

[7] Nazmul Islam Raju. Md. Shahinur Islam. Ahmed Ahsan Uddin. "Sinusoidal PWM Signal Generation Technique for Three Phase Voltage Source Inverter with Analog Circuit Simulation of PWM Inverter for Standalone Load Microgrid System ". In: *International Journal of Renewable Energy Research* 3 ().

[8] Mathworks. *Clarke and Park Transforms*. URL: https://se.mathworks.com/solutions/power-electronics-control/clarke-and-park-transforms.html.

[9] Mathworks. *Clarke Transform*. URL: https://se.mathworks.com/help/physmod/sps/ref/clarketransform.html.

[10] Mathworks. *Park Transform*. URL: https://se.mathworks.com/help/physmod/sps/ref/parktransform.html.

[11] Mukesh Kumar. Bhim Singh. B.P.Singh. "Modeling and Simulation of Permanent Magnet Brushless Motor Drives using Simulink". In: (Dec. 2002).

[12] Texas Instruments. *Clarke Park Transforms on the TMS320C2xx*. Tech. rep. Texas Instruments, 1997.

[13] Marek Stulrajter. Valeria Hrabovcova. Marek Franko. "Permanent Magnets Synchronous Motor Control Theory". In: *Journal of Electrical Engineering* 58 (), pp. 79–84.

[14] Emil Klintberg. “Comparison of Control Approaches for Permanent Magnet Motors”. MA thesis. Department of Energy and Environment, Chalmers University, 2013.

[15] Christopher J. Damaren. *AER1503H Spacecraft Dynamics and Control II, University of Toronto, Positive Real Design*. URL: http://arrow.utias.utoronto.ca/~damaren/aer506h.html#plan.

[16] Precision Microdrives. *Cogging Torque In Permanent Magnet Motors*. URL: https://www.precisionmicrodrives.com/content/cogging-torque-in-permanent-magnet-motors/.

[17] Danielle Collins. *What is stick-slip?* Dec. 2016. URL: https://www.linearmotiontips.com/faq-what-is-stick-slip/.

[18] National Instruments. *PID Theory Explained*. May 2019. URL: http://www.ni.com/sv-se/innovations/white-papers/06/pid-theory-explained.html.

[19] Francesco Sabatino. “Quadrotor control: modeling, nonlinear control design, and simulation”. MA thesis. Department of Electrical Engineering, KTH University, 2015.

[20] Anton A. Stoorvogel. Ali Saberi. “The Discrete Algebraic Riccati Equation and Linear Matrix Inequality”. MA thesis. 1998.

[21] MathWorks. *Linear-Quadratic Integral Control*. URL: https://se.mathworks.com/help/control/ref/lqi.html.

[22] Eduardo Camacho. Carlos Bordons. *Model Predictive Control*. Springer, 2012.

[23] P.E. Orukpe. “Model Predictive Control Fundamentals”. In: *Nigerian Journal of Technology* 31 (2012), pp. 139–148.

[24] Alberto Bemporad. “Explicit Model-Predictive Control”. In: *Encyclopedia of Systems and Control* (2014).

[25] Junho Lee. Hyuk-Jun Chang. “Analysis of explicit model predictive control for path-following control”. In: (2018). DOI: https://doi.org/10.1371/journal.pone.0194110.

[26] Jacques Smuts. *A Tutorial on Cascade Control*. Mar. 2015. URL: http://blog.opticontrols.com/archives/105.

[27] Dimitry Gorinevsky. *EE392m: Control Engineering Methods for Industry, Stanford University - Lecture 5: Feedforward*. 2002. URL: https://web.stanford.edu/class/archive/ee/ee392m/ee392m.1034/Lecture5_Feedfrwrd.pdf.

[28] Lennart Harnefors. *Control of Variable-Speed Drives*. Västerås : Applied Signal Processing and Control, Department of Electronics, Mälardalen University, 2002.