# Design Project: Phase Three

Jarod Tang

November 2, 2026

## Contents

**1 Introduction** **2**

**2 Preliminaries** **2**
2.1 State-Space Model . . . . . 2
2.2 Reduced-Order Model . . . . . 3
2.3 Performance Requirements . . . . . 3

**3 Task A: Linear Quadratic Regulator (LQR) Control** **3**
3.1 A1. Symmetric Root Locus (SRL) Controller Design . . . . . 4
3.2 A2. Riccati Equation Controller Design . . . . . 5

**4 Task B: Linear Quadratic Gaussian (LQG) Control** **6**
4.1 Sensor Selection . . . . . 6
4.2 Implementation of a LQG Controller . . . . . 6

**5 Task C: Sinusoidal Disturbance Rejection** **8**

**6 Task D: Model Predictive Control (MPC)** **9**

**7 Conclusion** **13**

# 1 Introduction

The objective of Phase 3 is to design optimal control methods for the single-link flexible-joint robot. Firstly, an LQR controller is designed using Symmetric Root Locus (SRL) and the Riccati equation approach. Next, using one or more of the available sensors, an LQG controller is designed which incorporates sinusoidal input disturbance whilst also satisfying the required specifications. Lastly, a model predictive controller (MPC) is implemented.

# 2 Preliminaries

## 2.1 State-Space Model

From Phase One, the state space is depicted by Equation 1, where $\theta_m$, $\theta_l$, $\dot{\theta}_m$, $\dot{\theta}_m$ and $I_a$ is the angular position of the motor and load with respect to the fixed base, angular speed of the motor and load and the armature current respectively.

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix} = \begin{bmatrix} \theta_m & \theta_l & \dot{\theta}_m & \dot{\theta}_l & I_a \end{bmatrix} \tag{1}$$

The plant and motor model provided by the Design Project specifications is as follows and depicted by Equations 2 and 3 respectively. Note that $K_s$ is the spring constant and the output torque ($T_m$) is given by Equation 4. The selection of the $b$ value remains trivial, as it does not carry over into future calculations, whilst $K_s$ was set to 1.8.

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{K_s}{J_m} & \frac{K_s}{J_m} & 0 & 0 \\ \frac{K_s}{J_l} & -\frac{K_s}{J_l} & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{-b \cdot \text{sgn}(x_3) x_3^2}{J_m} \\ \frac{mgl \sin(x_2)}{J_l} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \frac{1}{J_m} \\ 0 \end{bmatrix} T_m \tag{2}$$

$$\dot{x}_5 = -\frac{R_a}{L_a} x_5 - \frac{K_m}{L_a} x_3 + \frac{1}{L_a} u \tag{3}$$

$$T_m = K_m I_a = K_m x_5 \tag{4}$$

The linearisation point and selected motor will remain the same as Phase One and Two, being 30° and Motor 2 respectively. The equilibrium values are as follows:

$$\begin{bmatrix} x_1^* & x_2^* & x_3^* & x_4^* & x_5^* \end{bmatrix} = \begin{bmatrix} 31.56 & 30 & 0 & 0 & 1.8167 \end{bmatrix} \tag{5}$$

The incremental variables were set to the following:

$$\delta_x = \begin{bmatrix} \delta_{x_1^*} & \delta_{x_2^*} & \delta_{x_3^*} & \delta_{x_4^*} & \delta_{x_5^*} \end{bmatrix}^T = \begin{bmatrix} x_1 - x_1^* & x_2 - x_2^* & x_3 - x_3^* & x_4 - x_4^* & x_5 - x_5^* \end{bmatrix}^T \tag{6}$$

$$\delta_u = u - u^* \tag{7}$$

The linear system model of the plant and motor combination is depicted by Equations 8 and 9.

$$\dot{\delta_x} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ -\frac{K_s}{J_m} & \frac{K_s}{J_m} & -\frac{2b|x_3 j|}{J_m} & 0 & \frac{K_m}{J_m} \\ \frac{K_s}{J_l} & -\frac{K_s}{J_l} - \frac{mgl\cos(x_2 j)}{J_l} & 0 & 0 & 0 \\ 0 & 0 & -\frac{K_m}{L_a} & 0 & -\frac{R_a}{L_a} \end{bmatrix} \delta_x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \frac{1}{L_a} \end{bmatrix} \delta_u \tag{8}$$

$$y(t) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \delta_x \tag{9}$$

### 2.2 Reduced-Order Model

As per Phase Two, to reduce the complexity of the controller tuning process, the motor dynamics were omitted. After linearisation, the reduced-order model (ROM) can then be derived from this new expression and is depicted by Equation 10.

$$\dot{\delta}_{x_{red}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{K_s}{J_m} & \frac{K_s}{J_m} & -\frac{2b|x_3^*|}{J_m} - \frac{K_m^2}{J_m R_a} & 0 \\ \frac{K_s}{J_l} & -\frac{K_s}{J_l} - \frac{mgl\cos(x_2 j)}{J_l} & 0 & 0 \end{bmatrix} \delta_{x_{red}} + \begin{bmatrix} 0 \\ 0 \\ \frac{K_m}{J_m R_a} \\ 0 \end{bmatrix} \delta_u \tag{10}$$

The new incremental variables are as follows:

$$\delta_{x_{red}} = \begin{bmatrix} \delta_{x_1^*} & \delta_{x_2^*} & \delta_{x_3^*} & \delta_{x_4^*} \end{bmatrix}^T = \begin{bmatrix} x_1 - x_1^* & x_2 - x_2^* & x_3 - x_3^* & x_4 - x_4^* \end{bmatrix}^T$$

Substituting the same linearisation point used in Phase One, the matrices $A$, $B$ and $C$ are as follows:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -240000 & 240000 & -48.6 & 0 \\ 1800 & -1885 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ 1800 \\ 0 \end{bmatrix} \quad C = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}$$

Subsequently, the final reduced state-space representation is given by $\dot{\delta}_x = A\,\delta_{x_{\text{red}}} + B\,\delta_u$ and $y(t) = C\,\delta_{x_{\text{red}}}$.

### 2.3 Performance Requirements

As per Phase One, the control system was tasked with moving the arm repeatedly between 15° and 45° angular positions after 3 second intervals, where the arm's angle $\theta_l$ must track a square wave with a 15° amplitude, 30° offset, and a 6-second period. The performance requirements are as follows:

- Steady-state error of at most 2 degrees, excluding error induced by changes in the direction of the reference signal.
- Maximum magnitude of 5 degrees for any overshoot/undershoot of $\theta_l$ when the reference signal changes direction.
- A "0.5 degree" settling time of no more than 2 seconds.
- Maximum controller sampling frequency of 50 Hz.

Converting these performance requirements into the frequency domain yields the following: the maximum overshoot/undershoot percentage $M_p = 16.67\%$, damping ratio $\zeta = 0.4954$, natural frequency $w_n = 9.5797$, and crossover frequency $w_c = 9.1728$. As per Phase 2, the lower bound on the dominant poles was calculated to be $s_{1,2} = -4.746 \pm 8.321\,i$.

## 3 Task A: Linear Quadratic Regulator (LQR) Control

For this Section and beyond, Motor 2 will be used. Additionally, for Section 3, ideal sensor measurements are assumed. The goal of LQR control is to design an optimal state feedback controller such that it minimises a quadratic cost function, balancing system performance (state

deviations) and control effort. The cost function is as follows, where $Q$ is a matrix that penalises state deviations, and $R$ is a matrix that penalises control effort:

$$J = \int_0^\infty \left(x^T(t)Qx(t) + u^T(t)Ru(t)\right) dt \tag{11}$$

The following are assumed:

- $Q, R$ are symmetric matrices, $Q$ is positive semi-definite, and $R$ is positive definite
- $(A, B)$ is stabilisable
- $\left(A, Q^{\frac{1}{2}}\right)$ is detectable

The stabilising control law is $u = -Kx(t)$ where $K = R^{-1}B^TP$ and $P$ is the unique symmetric positive definite solution to the following algebraic Riccati equation.

$$A^TP + PA - PBR^{-1}B^TP + Q = 0 \tag{12}$$

By solving the Riccati equation, the poles are placed in locations that balance the trade-off between minimising the selected state deviations ($Q$) and control effort ($R$). The implications of tuning $Q$ and $R$ are as follows:

- If $||Q||$ is increased, then transient behaviour is penalised.
- If $||R||$ is increased, then control effort is penalised.

Note that only the relative magnitudes between $Q$ and $R$ matter. However, while the LQR method guarantees optimality by minimising the quadratic cost function, it doesn't explicitly take into account specific time-domain performance requirements such as settling time, overshoot, or damping ratio when tuning $Q$ and $R$.

### 3.1 A1. Symmetric Root Locus (SRL) Controller Design

To address the aforementioned limitation, the SRL approach is employed. SRL allows for a more intuitive, performance-based design by enabling the visual selection pole locations based on performance specifications, such as settling time and overshoot, directly from the root locus plot. Moreover, this combines the optimal control approach with "dominant poles" approach that captures time-domain specifications. The characteristic equation for SRL is:

$$1 + \rho^{-1}G_\rho(-s)G_\rho(s) = 0, \quad \rho > 0 \tag{13}$$

where $G(s)$ is the transfer function from the control input to the performance output, and $\rho$ is a tuning parameter that adjusts the balance between performance and control effort. An output matrix of $C_\rho = \begin{bmatrix}0 & 3 & 0 & 0 & 10\end{bmatrix}$ was selected, penalising deviations in arm angular position and integral error. Using the MATLAB routine *rlocus*, the SRL plot is produced and depicted by Figure 1, where the purple squares are the locations of the desired poles found in Phase Two. Poles should be selected that align with the desired performance specifications, which can then inform the structure of the $Q$ matrix. For example, if the desire is for a fast response with minimal overshoot, then poles can be placed further in the left-half plane (LHP), which corresponds to higher penalisation of state deviations in the $Q$ matrix. Conversely, placing poles closer to the imaginary axis would penalise control effort, reflected by a smaller weight on the $Q$ matrix and a higher weight on the $R$ matrix. In light of this, to retain the performance characteristics from Phase Two, the previously derived dominant poles are used as a guideline for balancing such trade-offs, as these were calculated according to the performance specifications

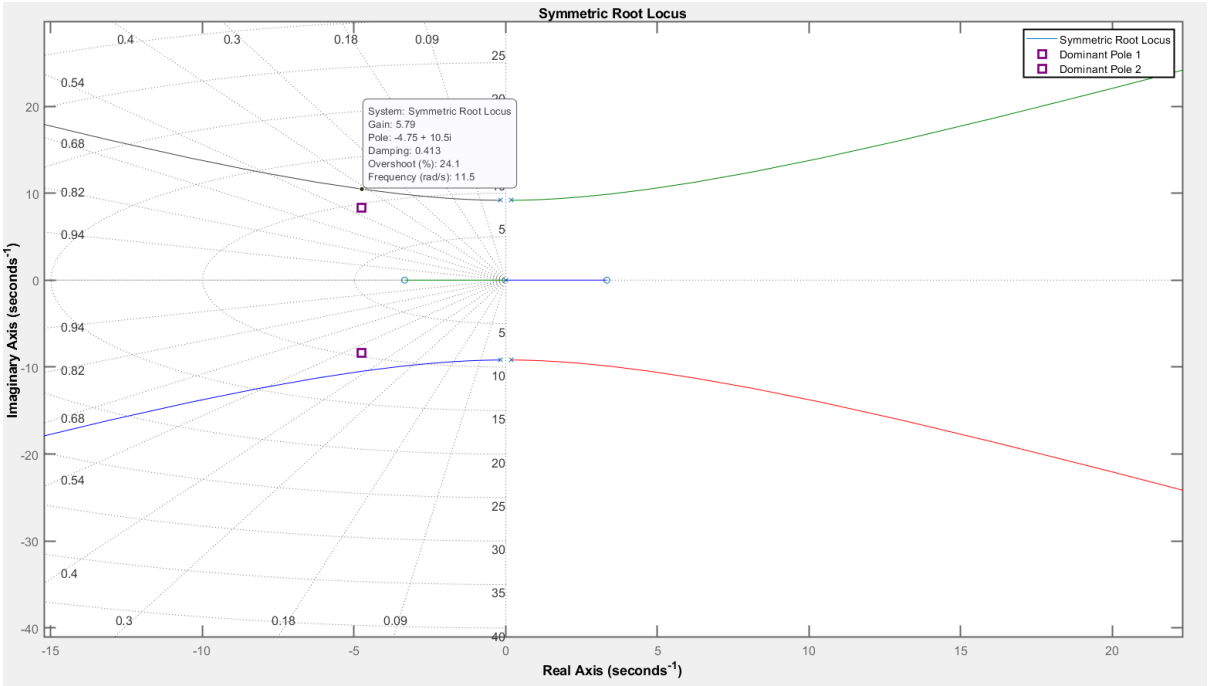



Figure 1: Selecting $\rho$ From SRL Plot

and an appropriate gain value of $\rho = 5.79$ with corresponding poles of $s = -4.75 \pm 10.5\,i$ was selected. $Q$ and $R$ can be found using the following relationships: $Q = \rho C_\rho^T C_\rho$, $R = 1$. This yields the matrices depicted below, which can be substituted into the MATLAB routine *lqr* to obtain the optimal gains $K$. The performance of the subsequent LQR controller on the linear model can be found in Section 3.2 depicted by Figure 2, where both the continuous and emulated controllers satisfy the performance requirements.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 52.11 & 0 & 0 & 173.7 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 173.7 & 0 & 0 & 579 \end{bmatrix}, \quad R = 1$$

$$K = \begin{bmatrix} 0.346 & 4.841 & 0.0064 & 0.849 \end{bmatrix}, \quad K_I = 21.984$$

### 3.2 A2. Riccati Equation Controller Design

Whilst the current iteration of $Q$ and $R$ achieve the performance specifications, these matrices can be further tuned to improve tracking. Adjusting the $\rho$ parameter offers a way to balance control effort and state deviation minimization: increasing $\rho$ amplifies the entries in $Q$, emphasizing state deviation reduction, while decreasing $\rho$ makes control input more costly, reducing control effort. In Figure 2, different values of $\rho = [0.1, 100, 100000]$ were tested to observe their impact. With $\rho = 100000$, the controller tracks the reference but shows excessive oscillations and has a steady-state error of over 2 degrees. Additionally, the actuator experiences saturation, an overall undesirable combination for practical implementation. Conversely, $\rho = 0.1$ minimises control effort but lacks sufficient precision in tracking, only loosely following the reference signal. At $\rho = 100$, the controller achieves a desirable balance, yielding a quicker response compared to the controllers developed in Section 3.1 whilst still offering accurate tracking without oscillation or saturation issues. The final tuned values for $Q$, $R$, and the resulting gains $K$ for $\rho = 100$ are as follows:

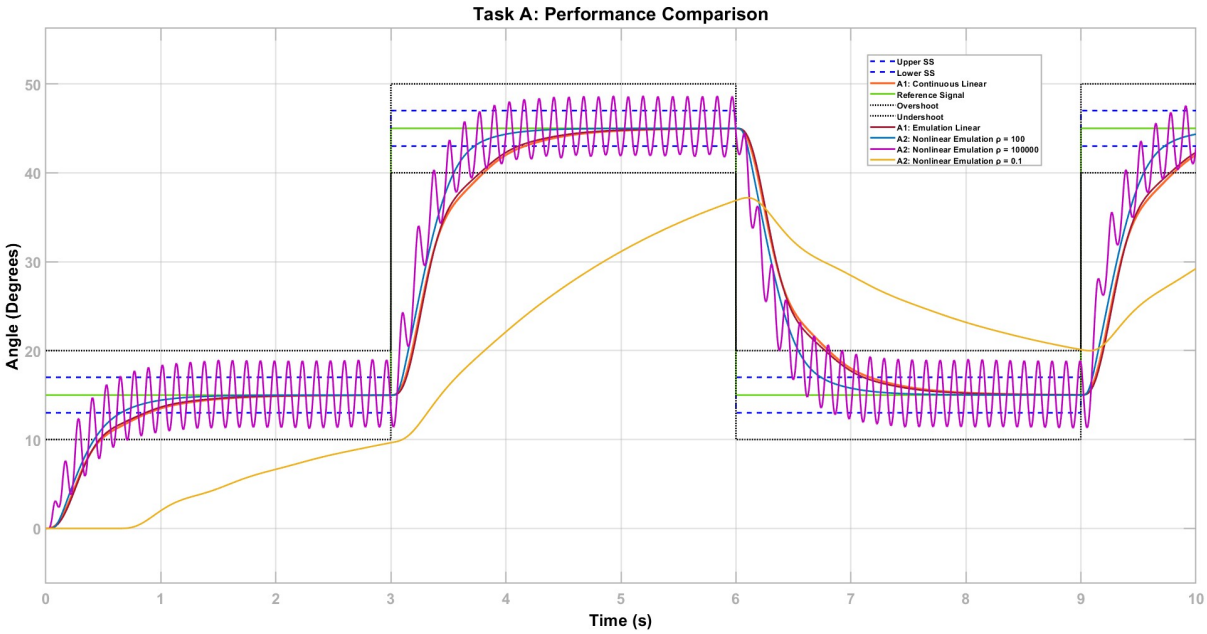



Figure 2: Task A: Performance Comparison for Different $\rho$

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 900 & 0 & 0 & 3000 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 3000 & 0 & 0 & 10000 \end{bmatrix}, \quad R = 1$$

$$K = \begin{bmatrix} 0.9888 & 29.6798 & 0.0158 & 2.1053 \end{bmatrix}, \quad K_I = 100.0000$$

# 4 Task B: Linear Quadratic Gaussian (LQG) Control

## 4.1 Sensor Selection

In Phase Two of the Design Project, initial tests comparing sensor measurements to "ideal" values demonstrated that the encoder offered the most precise estimation of the motor and link positions ($\theta_m$ and $\theta_l$), while both the gyroscope tachometer helped improve the quality of the estimated angular velocities ($\dot{\theta_m}$ and $\dot{\theta_l}$). For those reasons, the *CUI Devices (AMT203-V)* encoder, the *Invensense (ICM-20948)* gyroscope and the *CUI Devices (AMT103-2048-N4000-S)* tachometer were selected, which totals to $130 AUD and falls within the $150 AUD sensor budget allocated. It is worth noting that the gyroscope produces drift over time due to integration and will introduce errors in sensor readings. Therefore, a high-pass filter can be added after the gyroscope. Firstly, using the MATLAB routine *fft*, the Discrete Fourier Transform (DFT) of the gyroscope signal is converted from the time domain to the frequency domain to reveal the signal's frequency components and is depicted below by Figure 3. The plot shows a significant spike about 0Hz, likely reflecting the DC gain in the gyroscope signal. To mitigate these effects, a cutoff frequency of $\omega_c = 0.1Hz$ was selected. A first-order continuous-time high pass filter of the form $H(s) = \frac{s}{s+\omega_c}$ is then used.

## 4.2 Implementation of a LQG Controller

The purpose of Linear Quadratic Gaussian (LQG) control is to combine optimal state feedback (LQR) with a Kalman filter for state estimation, enabling robust control of systems with noise

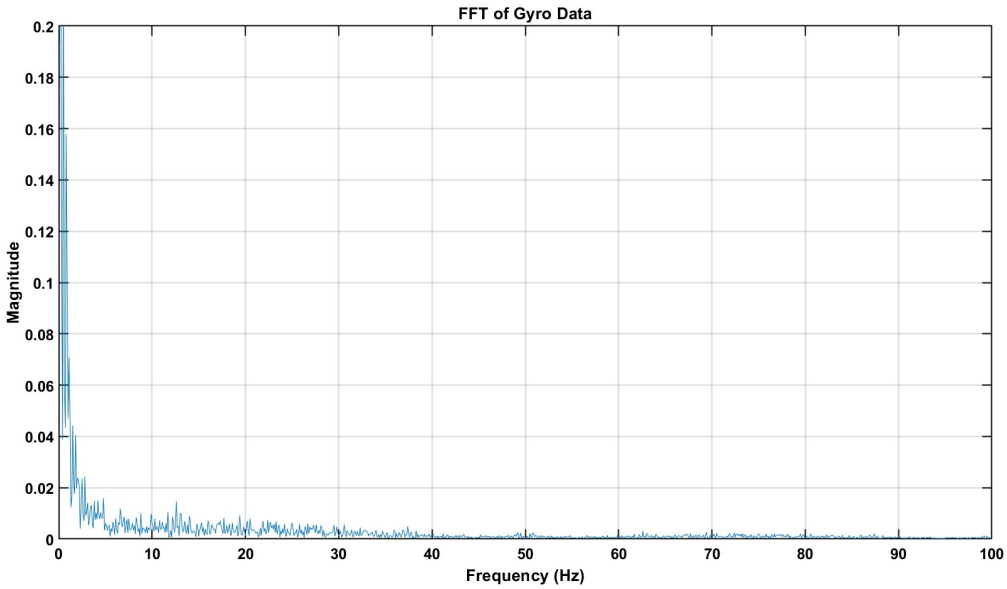



Figure 3: Fast Fourier Transform of Gyroscope Signal

and incomplete state measurements. Consider a continuous-time system with the following state-space representation

$$\dot{x}(t) = Ax(t) + Bu(t) + w(t), \quad y(t) = Cx(t) + v(t)$$

where $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the input, $y(t) \in \mathbb{R}^p$ is the output, $w(t) \in \mathbb{R}^n$ is the process noise, and $v(t) \in \mathbb{R}^p$ is the measurement noise. Assuming $w(t), v(t)$ are uncorrelated white noise processes with the following autocovariance functions:

$$E[w(t)w^T(\tau)] = W\delta(t-\tau), \quad E[v(t)v^T(\tau)] = V\delta(t-\tau)$$

where $W$ is an $n \times n$ matrix and $V$ is a $p \times p$ matrix. Moreover, suppose that a state estimate $\hat{x}(t)$ is produced using the closed-loop observer dynamics

$$\dot{\hat{x}}(t) = A\hat{x}(t) + Bu(t) + L\left(y_m(t) - C_m\hat{x}(t)\right)$$

such that the steady-state error covariance

$$P = \lim_{t\to\infty} E\left[\left(x(t) - \hat{x}(t)\right)\left(x(t) - \hat{x}(t)\right)^T\right]$$

is minimised. The optimal observer gains are then given by

$$L = PC_m^T V^{-1}$$

where:

$$C_m = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$P$ is the unique symmetric positive definite solution to the following algebraic Riccati equation:

$$AP + PA^T - PC_m^T V^{-1} C_m P + W = 0$$

The noise covariance matrices $W$ and $V$ represent the process noise w(t) and the measurement noise v(t) respectively and describe the uncertainty in the system model (process noise) and

the uncertainty in the measurements (measurement noise). However, since in practice, the exact stochastic properties of these noise terms are often unknown, they are treated as tuning parameters during the observer design. Analogous to LQR controller design, increasing $||W||$ leads to higher observer gains due to lower confidence in the system model and increasing $||V||$ leads to lower observer gains due to higher confidence in the system model. Moreover, via the separation principle, the LQR controller and Kalman filter can be designed independently. In the implementation of a Kalman filter, either a full-order or reduced-order Kalman filter can be used. Reduced-order Kalman filters estimate only the unmeasured key states, making them more efficient and faster to compute compared to full-order filters, which estimate all states. This reduced complexity results in quicker convergence and lower sensitivity to noise, focusing on essential variables for control. In contrast, full-order filters are more computationally demanding and often unnecessary when not all states need to be estimated. Reduced-order filters are thus preferred in practical applications where fewer measurements are required, offering a more efficient and flexible solution. Therefore a reduced-order Kalman will be used in the design of a LQG controller. Consider the ROM from in Section 2.2, the states of the model can be partitioned into measured $x_1$ and estimated $x_2$.

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{12}\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix} + \begin{bmatrix}B_1\\ B_2\end{bmatrix}u(t), \quad y(t) = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}x_1(t)\\ x_2(t)\end{bmatrix}$$

For $\hat{x_2} = z + J_r x_1$, the observer error as $\tilde{x_2} = z + J_r - \hat{x_2}$. The observer dynamics are $\dot{z} = J_r(-A_{11}x_1 - B_1u) + A_{21}x_1 + B_2u + (A_{22} - J_rA_{12})(z + J_rx_1)$. The error dynamics are expressed as $\dot{\tilde{x_2}} = (A_{22} - J_rA_{12})\tilde{x_2}$. $J_r$ is then designed by firstly selecting the following matrices for $W$ and $V$.

$$W = 1000 \times \begin{bmatrix}1\end{bmatrix} \quad V = 10^4 \times \begin{bmatrix}1 & 0 & 0\\ 0 & 150 & 0\\ 0 & 0 & 600\end{bmatrix}$$

Using these values, the observer gains can be calculated via the duality principle, where instead of solving the Kalman filter gains directly, the optimal gains can be found using an LQR approach on the dual system. This can be done by computing the LQR gain using the transposed system matrices via the MATLAB routine *lqr* and the optimal gains $J_r = \begin{bmatrix}0 & 0.0258 & -5.0697 \times 10^{-5}\end{bmatrix}$ are calculated as such. Figure 4 depicts the performance of the designed controllers. Both the continuous and emulated controllers satisfy the performance requirements, with little difference between either controller. Slight oscillations are observed around the steady-state position after both the positive and negative steps in the reference signal. This oscillation could be attributed to either sensor noise or unmodeled dynamics in the system.

# 5 Task C: Sinusoidal Disturbance Rejection

In this section, the internal model principle (IMP) will be implemented to counteract a sinusoidal input disturbance $d(t)$ of the form $d(t) = 0.3\sin(4\pi t)\,\mathrm{V}$ . It is assumed that this disturbance can be modelled by the following state-space model:

$$\dot{x}_d(t) = A_d x_d(t), \quad d(t) = C_d x_d(t)$$

where $A_d = \begin{bmatrix}0 & 1\\ -\omega^2 & 0\end{bmatrix}$ and $C_d = \begin{bmatrix}1 & 0\end{bmatrix}$ is the state-space model of the disturbance. Such a disturbance can then be negated by estimating it according to the above model and incorporating this estimate in the following control law: $u = -K\hat{x} + \hat{r} - \hat{d} = -K\hat{x} - C_d\hat{x}_d + \hat{r}$. The augmented system dynamics is then described by the following expression:

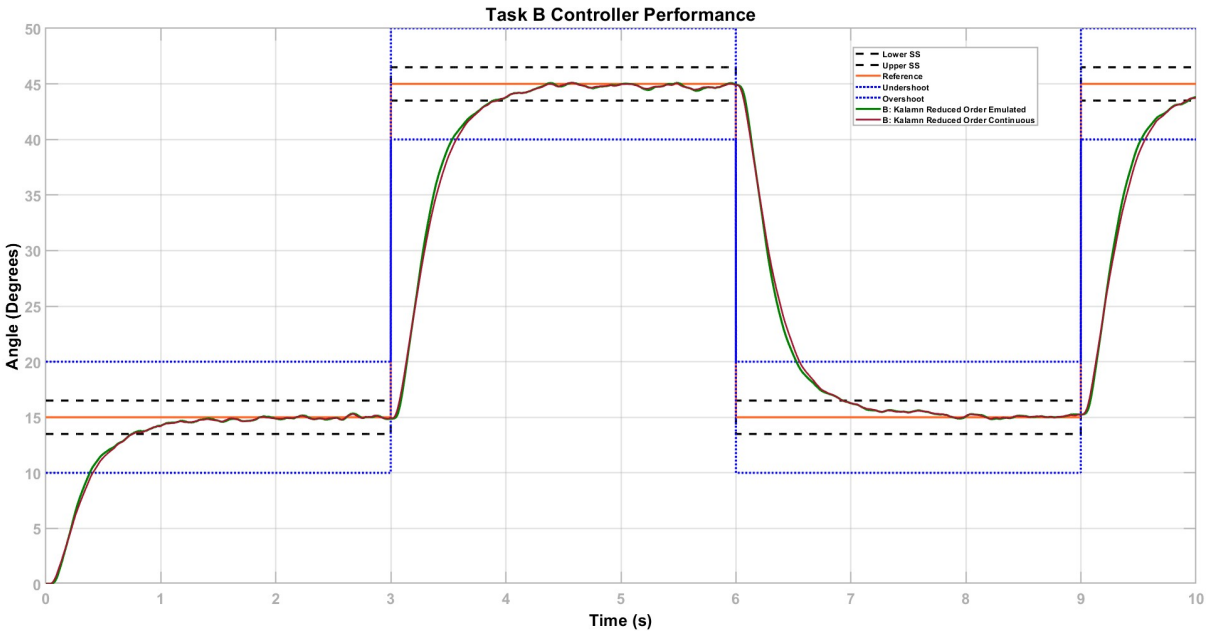



Figure 4: Task B Performance Comparison

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_d(t) \end{bmatrix} = \begin{bmatrix} A & BC_d \\ 0 & A_d \end{bmatrix} \begin{bmatrix} x(t) \\ x_d(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t), \quad y(t) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_d(t) \end{bmatrix}$$

where $A_{\text{aug}} = \begin{bmatrix} A & BC_d \\ 0 & A_d \end{bmatrix}$, $B_{\text{aug}} = \begin{bmatrix} B \\ 0 \end{bmatrix}$, $C_{\text{aug}} = \begin{bmatrix} C & 0 \end{bmatrix}$. An observer for the system states and disturbances can be designed according to the following expression:

$$\begin{bmatrix} \hat{x}(t) \\ \hat{x}_d(t) \end{bmatrix} = A_{\text{aug}} \begin{bmatrix} \hat{x}(t) \\ \hat{x}_d(t) \end{bmatrix} + B_{\text{aug}} u(t) + J \left( y(t) - C_{\text{aug}} \begin{bmatrix} \hat{x}(t) \\ \hat{x}_d(t) \end{bmatrix} \right)$$

Expanding upon the Kalman filter developed in Section 4, $W$, $V$ and $J_r$ are as follows:

$$W = 1000 \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 50 \end{bmatrix} \quad V = 10^4 \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 600 \end{bmatrix}, \quad J_r = \begin{bmatrix} 0 & 0.0257 & -0.0011 \\ 0 & 0.0273 & 0.1401 \\ 0 & 1.8179 & -0.0403 \end{bmatrix}$$

Observing Figure 5, the disturbance estimator mostly tracks the sinusoidal disturbance, with minor delays and deviations during transients, likely due to limited past data and slight sensor noise affecting rapid response adjustments. Comparing controller performance with and without the Internal Model Principle (IMP) highlights the robustness improvement with IMP, which reduces oscillations and smooths response by efficiently handling steady-state disturbances whilst meeting the performance specifications. Without IMP, the controller, whilst tracking the square wave, fails to meet the performance requirements and exhibits higher oscillatory behaviour and slightly poorer steady-state accuracy, as it struggles with persistent disturbances. Incorporating IMP thus enables the controller to predict and reject periodic disturbances more effectively, leading to improved tracking and stability.

# 6 Task D: Model Predictive Control (MPC)

MPC is a control strategy that optimises future control actions according to a system model to meet performance specifications with respect to constraints. At each time step, MPC solves

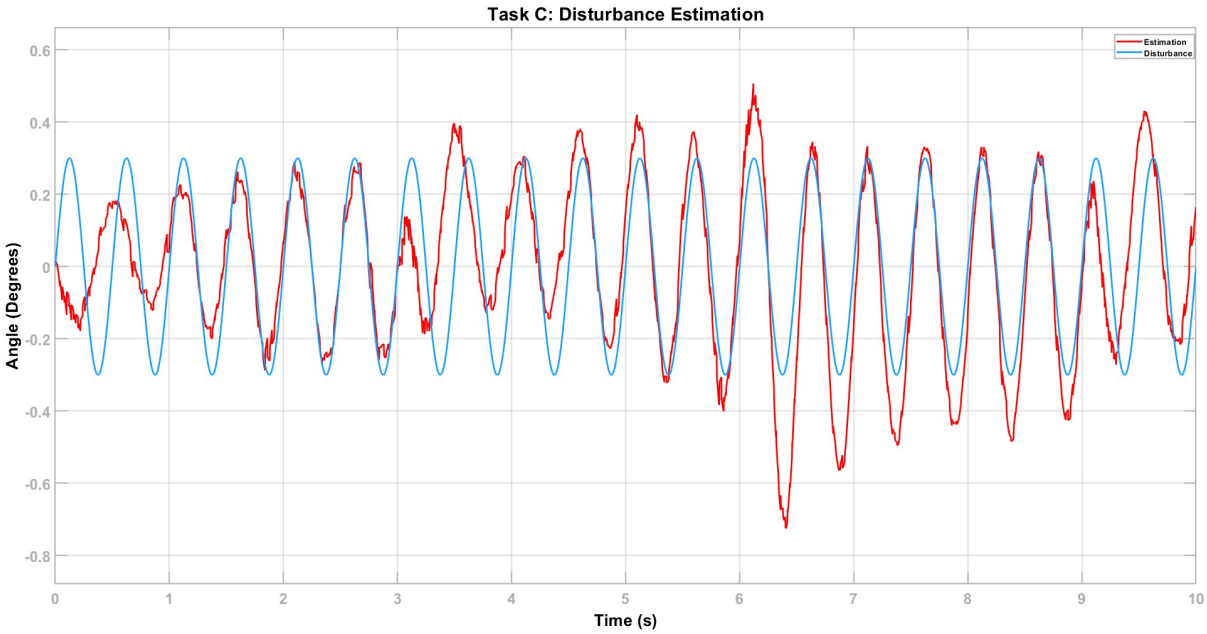



Figure 5: Disturbance Estimation

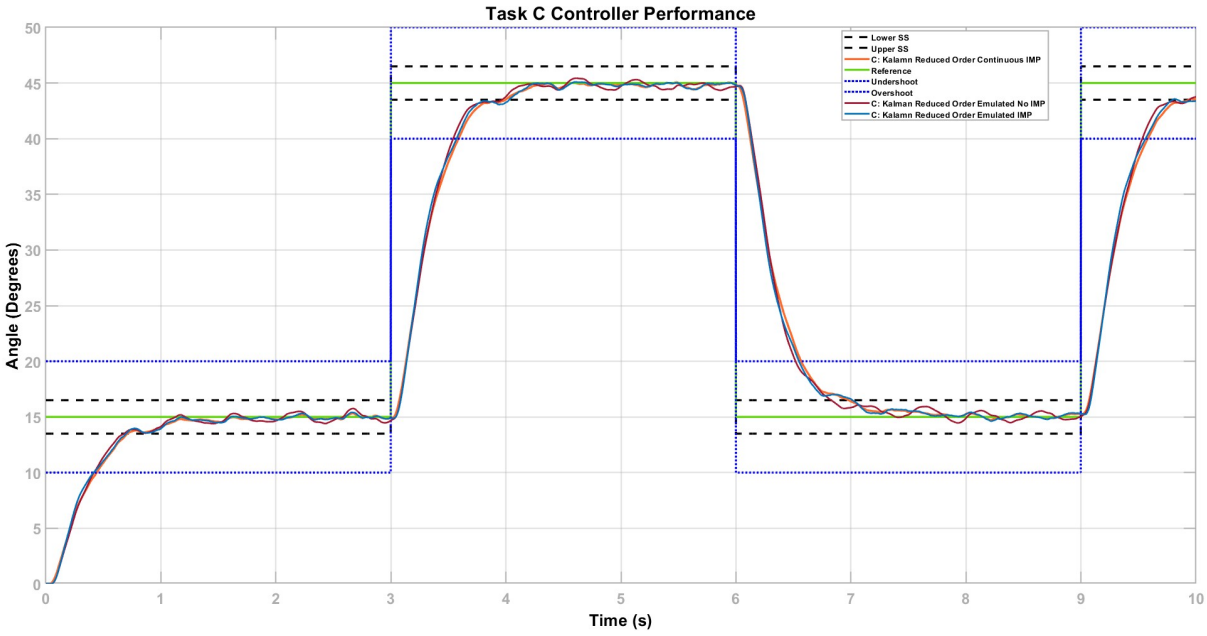



Figure 6: Task C Controller Performance

an optimisation problem over a prediction horizon and applies only the first control action. This process repeats at each step with updated state measurements, allowing MPC to adapt responsively to changes and disturbances. For this task, two types of MPC formulation can be considered: 1) Regularisation formulation; 2) Tracking formulation. Both formulations should be able to generate suitable control sequences, provided that adequate weighting matrices and constraints are selected. Ultimately, constrained linear MPC was chosen for this task, as it allows for more granular control over the weighting matrices, enabling each element within the matrices to be adjusted individually rather than with a singular value. The optimisation problem statement for regularisation, solved at each time step is as follows:

$$\min_{u_k, x_k, y_k} \; x_N^{'} P x_N + \sum_{k=0}^{N-1} \left( x_k^{'} Q x_k + u_k^{'} R u_k \right) \tag{14}$$

where $x_k$ is the predicted state at time $k$, $u_k$ is the control input at time $k$, and $P$ is the terminal cost matrix for the final state $x_N$ at the end of the prediction horizon. The objective function aims to minimize the cumulative cost over the prediction horizon $N$, which includes penalties for both state deviations (regulated by the weight matrix $Q$) and input efforts (regulated by the weight matrix $R$). The terminal cost matrix $P$ penalizes the final state $x_N$, ensuring the system approaches stability by the end of the horizon. The optimisation problem is subject to several constraints:

$$x_0 = \hat{x}(t) \tag{15}$$

Equation 15 represents the initial condition, where the initial state $x_0$ is set to the current estimated state $\hat{x}(t)$.

$$x_{k+1} = A x_k + B u_k, \quad k = 0, \ldots, N-1 \tag{16}$$

Equation 16 defines the system dynamics, where the predicted future states $x_{k+1}$ evolve according to the linear dynamics matrices $A$ and $B$.

$$y_k = C x_k, \quad k = 0, \ldots, N \tag{17}$$

The output equation establishes the relationship between the state $x_k$ and the output $y_k$, defined by the matrix $C$, allowing the system to predict future outputs.

$$u_{\min} \leq u_k \leq u_{\max}, \quad k = 0, \ldots, N-1 \tag{18}$$

Equation 18 imposes constraints on the control input $u_k$, ensuring it remains within the operating range of Motor 2 $u_{\min} = -12V$ and $u_{\max} = 12V$.

$$y_{\min} \leq y_k \leq y_{\max}, \quad k = 0, \ldots, N \tag{19}$$

Finally, Equation 19 enforces output constraints, ensuring the predicted output $y_k$ remains within the specified bounds $y_{\min}$ and $y_{\max}$ throughout the horizon. These are set to $\pm 15^{\circ}$. Through iteration, the weight matrices $Q$ and $R$ are set to the following values:

$$Q = 10 \times \begin{bmatrix} 10^{-16} & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 10^{-16} & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 100 \end{bmatrix}, \quad R = 10^{-16}$$

The $W$, $V$ and $J_r$ matrices are the same as the ones derived in Section 5. In the implementation of the emulated MPC controller, a soft state constraint was added to require zero overshoot on

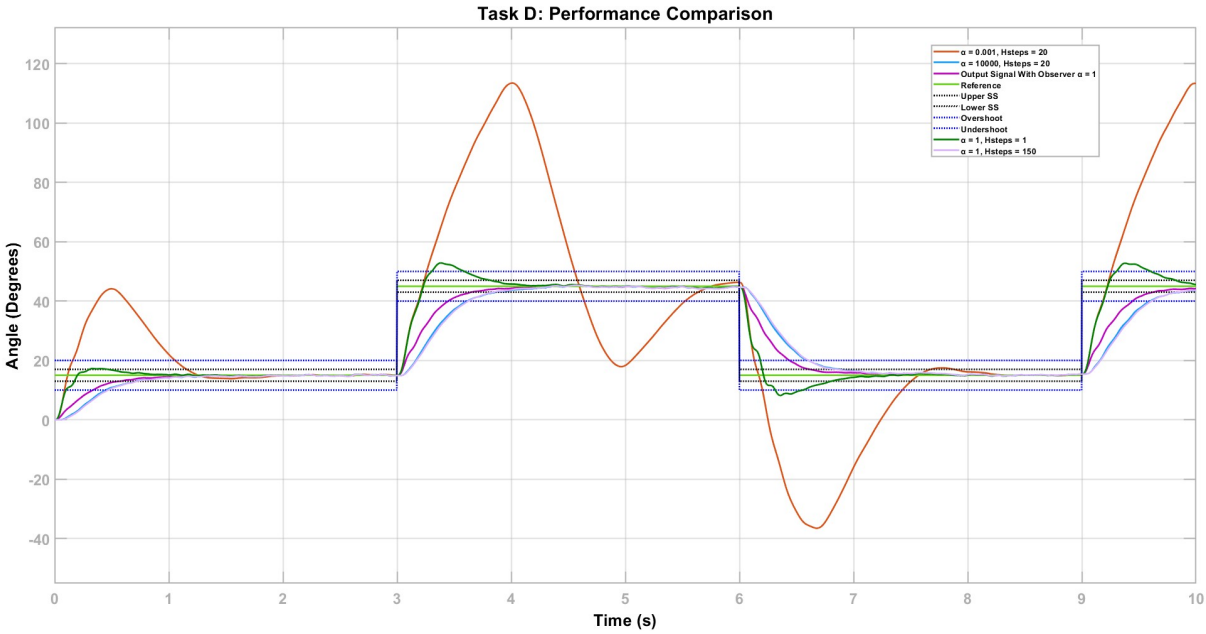



Figure 7: Task D: Controller Performance

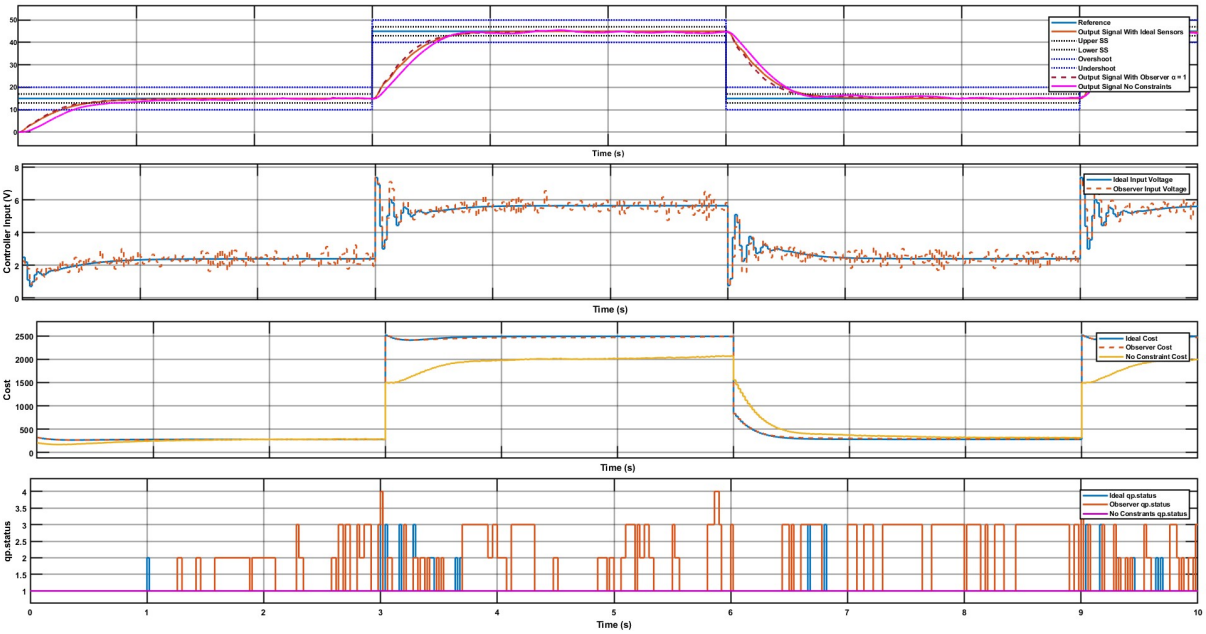



Figure 8: Task D: Controller Implementation

the angle of link $\theta_l$ and its subsequent performance will now be discussed. Note that a hard constraint was not used here since the response would become oscillatory once the reference signal was reached and the actuator would saturate. Observing Figure 7, the effects of changing the prediction horizon ($H_{Step}$) and the weighting matrices $Q$ and $R$ on the closed-loop performance are shown. Comparing controllers with $\alpha = 1$ and with differing $H_{Steps}$, all three controllers seem to track the reference signal fairly well. Moreover, controller performance barely differs for $H_{Step} = 20$ and $H_{Step} = 150$, however, while a longer prediction horizon generally improves the controller's ability to anticipate future errors and disturbances, leading to better tracking accuracy, overly long horizons may cause computational burdens, making real-time control challenging. A shorter horizon of $H_{Step} = 1$, whilst able to track the reference after taking some time to settle, demonstrates some overshoot and therefore violates the performance requirements. Since a $H_{Step} = 20$ retains the performance of a longer prediction horizon in this task, whilst requiring less computational effort, this was the chosen prediction horizon.

To analyse the effects of changing the weighting matrices $Q$ and $R$, values of $\alpha = \begin{bmatrix} 0.001 & 1 & 10000 \end{bmatrix}$ are multiplied with $Q$ and were trialled. Setting $\alpha = 0.001$ causes the controller to be incredibly responsive, with overly large peaks and troughs that violate the overshoot requirement. Setting $\alpha = 10000$ does not have a substantial impact on the response, with it being marginally slower than the original response indicated by $\alpha = 1$. This could be because the original relationship between $Q$ and $R$ already heavily penalised $Q$, meaning that this larger scalar would only have a limited impact on its pre-existing relationship.

Observing Figure 8 and specifically, the plot illustrates the number of quadratic programming (QP) solver iterations required to compute a suitable control sequence. In the constrained setup, the QP solver occasionally required multiple iterations to find feasible solutions that meet both angle and input constraints, indicating a higher computational load to handle the additional requirements. In contrast, the unconstrained setup consistently displayed an exit flag status of 1, implying that it quickly arrived at a solution without the need for further adjustments.

The impact of these behaviours on overall execution times is noticeable, with the constrained and unconstrained Simulink blocks completed in 11.266 seconds and 9.751 seconds respectively. This difference underscores how constraints in MPC introduce complexity, as the solver must adjust the control inputs to remain within the permissible bounds, whereas the unconstrained MPC in the absence of constraints, is less computationally demanding. However, both controllers successfully met the performance specifications, demonstrating similar accuracy in tracking the reference signal. This similarity in performance between the constrained and unconstrained controllers suggests that the model dynamics are well-calibrated, allowing for effective tracking even without explicitly enforcing constraints. The constrained controller, while more computationally intensive, may provide additional robustness under scenarios with tighter performance requirements or in cases where operating limits are essential for safety or system longevity. In this task, however, both approaches proved adequate, implying that the unconstrained MPC could serve as a more computationally efficient solution without compromising on control accuracy for this particular application.

# 7 Conclusion

Phase 3 of the Design Project was centred around the design of optimal controllers and demonstrated the efficacy of the LQR, LQG, and MPC control strategies. Overall, the performance of each of the controllers was acceptable, meeting all the performance requirements.