TÉCNICO LISBOA

Control of Cyber-Physical System

Lab Report 2

Design and Testing of a Controller for a Flexible Arm

Authors:

Ayesha Aslam (1109220) ayesha.aslam@tecnico.ulisboa.pt
Farooq Olanrewaju (1109173) olanrewajufarooq@yahoo.com
Md Sazidur Rahman (1109170) md.sazidur.rahman@tecnico.ulisboa.pt

Group 11

2023/2024 – 1st Semester, P1

Contents

1 Introduction **2**

2 (Preview) Model Identification **2**

3 Experimental Set-Up (MATLAB Code and Simulink) **3**

4 Design Considerations **5**
4.1 Choice of Weights in LQG Controller Design 5
4.2 Effect of Noise Covariance Matrices in LTR framework 6
4.3 Frequency and Time Responses 11
4.3.1 The Uncontrolled Closed-Loop Plant 11
4.3.2 The Controlled Plant: Proportional Controller 12
4.3.3 The Controlled Plant: Normalized Proportional Controller 14
4.3.4 The Controlled Plant: Proportional-Integral Controller 15
4.4 Effect of the Inclusion of a Pre-Filter 17

5 Performance Evaluation **20**

6 Conclusion **22**

1 Introduction

The goal of this lab is to control the plant which was identified in the previous lab. An LQR (Linear Quadratic Regulator) controller is used to control the plant. Firstly, we designed a basic controller using a block diagram in the Simulink. Then, we tuned the controller and chose the covariance matrices. We added a Pre-Filter based on the system's behaviour, which improved the system's performance.

2 (Preview) Model Identification

In the previous lab, we had performed the model identification and obtained a state-space model for the plant to be controlled.

The state space equation for our plant is:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

$$A = \begin{bmatrix} 3.9879 & -5.8186 & 3.2254 & 0.3447 & -1.0312 & 0.2918 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$C = 1.0 \times 10^{-3} \times \begin{bmatrix} -0.6242 & 0.7542 & -0.0716 & 0 & 0 & 0 \end{bmatrix}$$

$$D = 0$$

Substituting the values of these matrices in our state space equation gives us:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \end{bmatrix} = \begin{bmatrix} 3.9879 & -5.8186 & 3.2254 & 0.3447 & -1.0312 & 0.2918 \\ 1.0000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} u$$

Simplifying the state equation give us:

$$x_1(k+1) = 3.9879x_1(k) - 5.8186x_2(k) + 3.2254x_3(k) + 0.3447x_4(k) - 1.0312x_5(k) + 0.2918x_6(k)$$

$$\begin{aligned} x_2(k+1) &= x_1(k) \\ x_3(k+1) &= x_2(k) = x_1(k-1) \\ x_4(k+1) &= x_3(k) = x_1(k-2) \\ x_5(k+1) &= x_4(k) = x_1(k-3) \\ x_6(k+1) &= x_5(k) = x_1(k-4) \end{aligned}$$

Hence, we can rewrite the state equation as:

$$\begin{aligned} x_1(k+1) =& 3.9879x_1(k) - 5.8186x_1(k-1) + 3.2254x_1(k-2) + 0.3447x_1(k-3) - 1.0312x_1(k-4) \\ &+ 0.2918x_1(k-5) + u \end{aligned}$$

The output equation can be written as:

$$y(k) = 1.0 \times 10^{-3} \times \begin{bmatrix} -0.6242 & 0.7542 & -0.0716 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ x_6(k) \end{bmatrix}$$

Therefore, the output equation is:

$$y(k) = 1.0 \times 10^{-3} \times \left[-0.6242x_1(k) + 0.7542x_1(k-1) - 0.0716x_1(k-2) \right]$$

3 Experimental Set-Up (MATLAB Code and Simulink)

For the implementation of the controller for steering the flexible arm to a desired point, we created a MATLAB script (as shown in Figure 1) and a Simulink block diagram (as shown in Figure 2). The MATLAB Script is used for the importation of the identified model parameters (from the previous lab) and also for the computations of control parameters. Further information on the control parameters are provided in the appropriate sections. The Simulink block is used for the implementation of the current observer, the plant (and its model), the input pre-filter and the controllers. The experiment is setup such that the controller design and testing are done with same files by using *manual switches* to perform different investigations. Brief notes and diagrams of the Simulink subsystems are given in Figure 3.

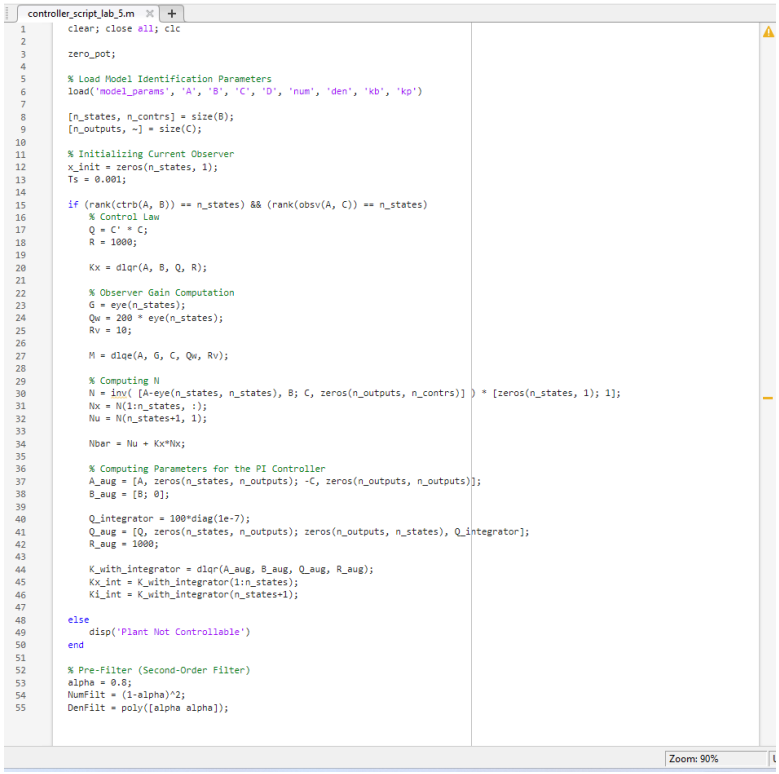


Figure 1: Controller MATLAB Script

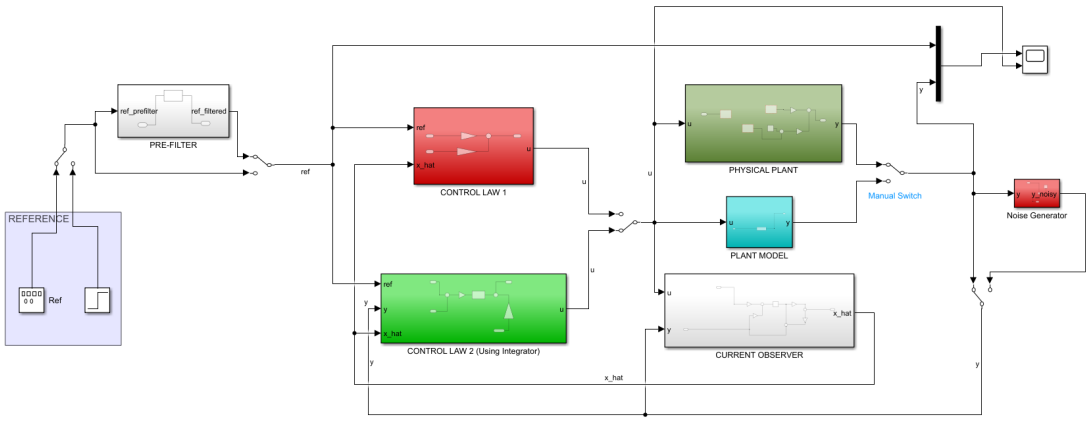
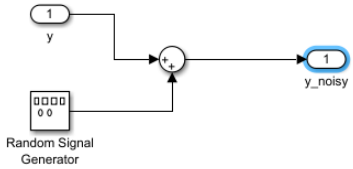
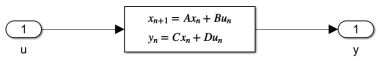


Figure 2: Controller Simulink Block Diagram

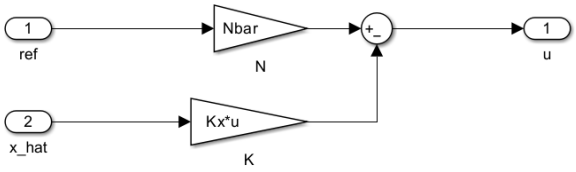
(a) A simple discrete transfer function for the implementation of the reference signal prefiltering.



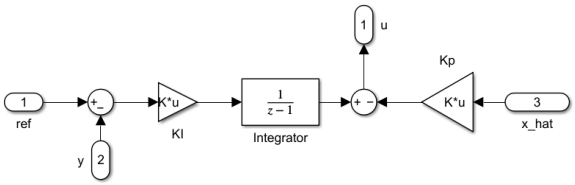
(b) Random Signal Generator to introduce noise into the sensor data during controller design.



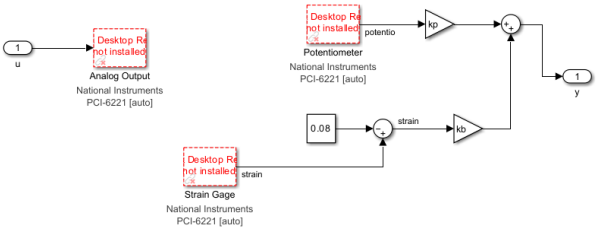
(c) A discrete state-space model of the plant which is used for controller design.



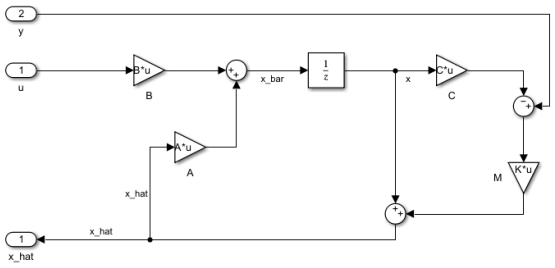
(d) A linear proportional control law with a static gain between the reference input and the plant output



(e) A linear PI control law.



(f) The interface between the controller (in Simulink) and the actual plant. The Analog Output (as seen in the diagram) is the input to the plant while the Potentiometer and the Strain Gauge are sensor measures from the plant.



(g) Current Observer Model for the Plant. The model is appropriate for both controller design and testing.

Figure 3: Diagrams of the Simulink Subsystems

4 Design Considerations

4.1 Choice of Weights in LQG Controller Design

In order to stabilize the plant or make the plant track some given trajectories, we need to design a controller. The main aim of the controller is to steer the plant to achieve the desired performance. There exist many techniques to achieve this, however, we would be utilizing the Linear Quadratic Regulator (LQR) controller as specified in the lab manual. An LQR controller is designed to regulate a linear dynamic system to achieve a balance between control effort (energy input) and system performance. The Linear part of LQR refers to the linearization of the system dynamics while the Quadratic part of LQR comes from the cost function that the

controller aims to minimize. The equation for the cost function is provided in Equation 1. In order to get the desired performance, the Q and the R values are to be tuned.

$$J = \sum_{k=1}^{\infty} \left(x(k)^T Q x(k) + u(k)^T R u(k) \right) \tag{1}$$

The cost matrices Q and R are needed to obtain the desired control for the plant. The Q matrix specifies the importance or cost associated with each state variable. A higher weight in the Q matrix means that minimizing the deviation of that state variable from its desired value is more important and R matrix specifies the cost associated with control effort or the use of control inputs. A higher weight in the R matrix means that minimizing the control effort is more important. However, before our controller can be developed, we need to check that our system is controllable. Hence, we used a MATLAB script to check that the rank of the controllability matrix gotten from the MATLAB function *ctrb*() is equal to the number of state variables we have. Since this condition is satisfied (as well as the observability - this would be discussed in the next section), we proceeded with the tuning of the LQR.

To select the right value for Q and R we started with the values given in the guide, $R = 100$ and $Q = C^T C$. And we used the MATLAB *dlqr*() command to obtain the desired gain matrix that is eventually used in the control law, $u_k = -Kx_k$[1].

$$Q = 1 \times 10^{-6} \times \begin{bmatrix} 0.3896 & -0.4707 & 0.0447 & 0 & 0 & 0 \\ -0.4707 & 0.5688 & -0.0540 & 0 & 0 & 0 \\ 0.0447 & -0.0540 & 0.0051 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After performing some tests, we observed that reducing the values of R makes the system behave erratically and noisily whereas increasing the values gives a smooth operation. Eventually, we set the value of $R = 1000$ to make the system "more stable".

4.2 Effect of Noise Covariance Matrices in LTR framework

In the design of our controller, we need the full state of system to determine the control variable according to the equation $u_k = -Kx_k$. However, we only have some values of the system output, y_k. Hence, we need develop a full-state (current) observer model that can be used to estimate the state parameters of the plant and the control law becomes $u_k = -K\hat{x}_k$. Since noise is present in our system, we have to include this in the observer model, then it becomes a Kalman Filter.

The Kalman Filter is implemented using the *dlqe*() function in MATLAB. This function requires that the gaussian noise matrix G be specified. We simply used an identity matrix to model the noise as seen in Equation 2. In the Kalman Equations, Q_w represents the covariance matrix of the process noise, where $Q_w = \epsilon(ww^T)$. This matrix describes the uncertainty associated with the system's dynamic behaviour and how it propagates from one state to the next. R_v represents the covariance matrix of the measurement noise, where $R_v = \epsilon(vv^T)$. This

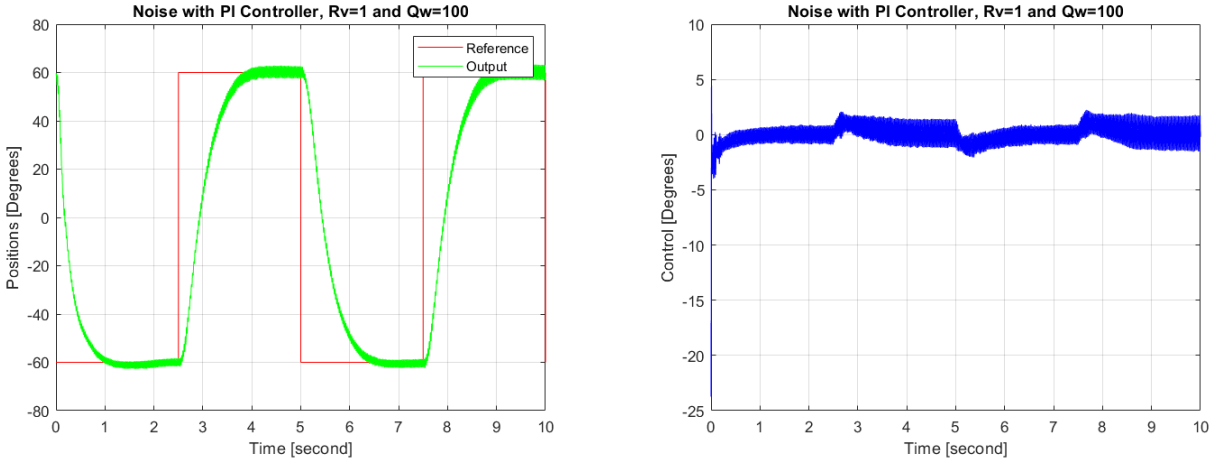
[1]The control laws eventually used on the plant is different from this. These are discussed later in the report.

matrix describes the uncertainty in the measurements taken from the system. We tuned the values of the costs (which is similar to the quadratic cost discussed in the previous section) Q_w and R_v to obtain the gain M for the state estimation.

The Kalman filter uses these covariance matrices in the prediction and update steps to balance the contributions of the predicted state and the actual measurements when estimating the true system state. The Kalman gain is a key factor in this process, and it depends on both Q_w and R_v. However, They are not estimated based on the noise covariances from the plant, but used as "tuning knobs".

To tune Q_w and R_v properly the loop transfer recovery (LTR) method was used. Initially, we set the value of $R_v = 1$ and $Q_w = 1000I$.

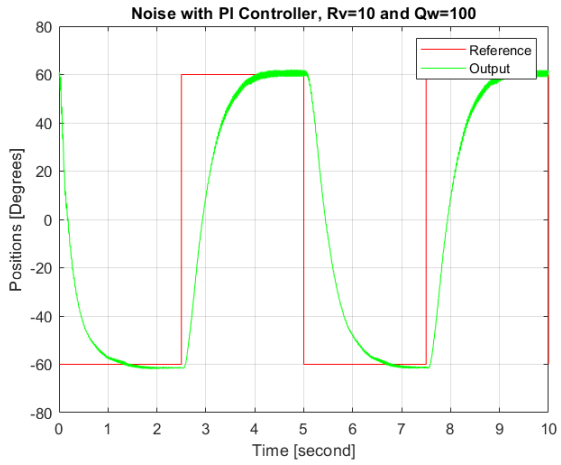
$$Q_w = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}, G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$



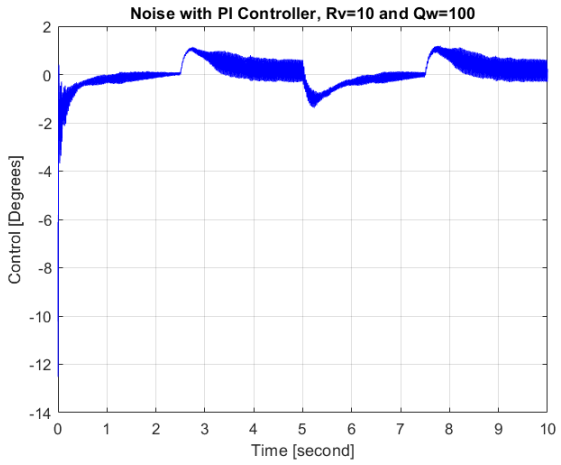
(a) Position of the flexible arm

(b) Control Output

Figure 4: Noise (QwRv) Tuning with PI controller, R1000, Qw=100 and Rv=1

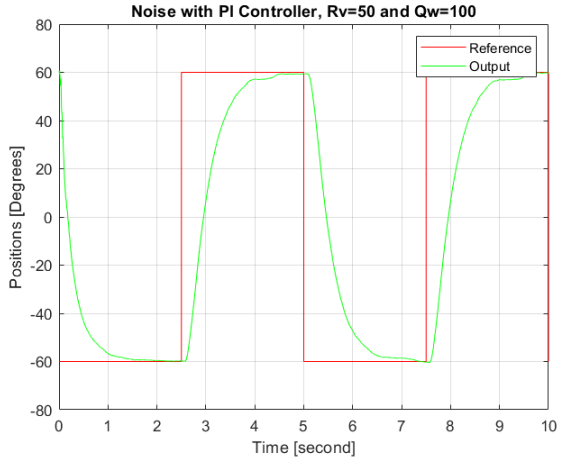


(a) Position of the flexible arm

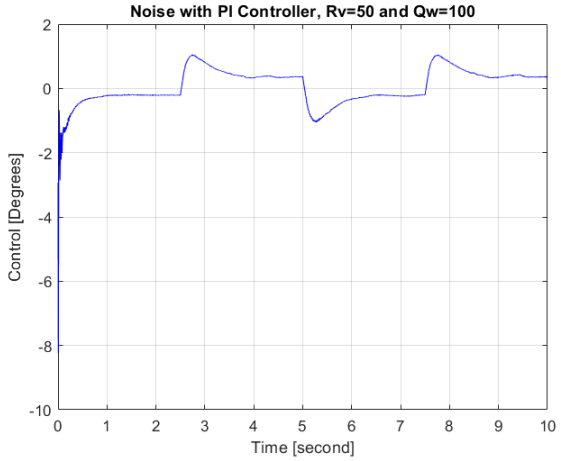


(b) Control Output

Figure 5: Noise (QwRv) Tuning with PI controller, R1000, Qw=100 and Rv=10

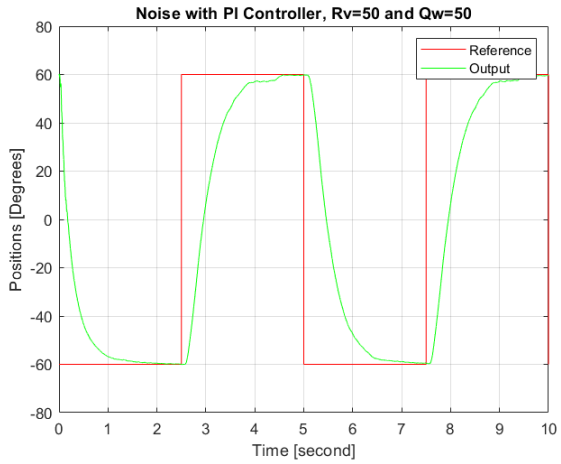


(a) Position of the flexible arm

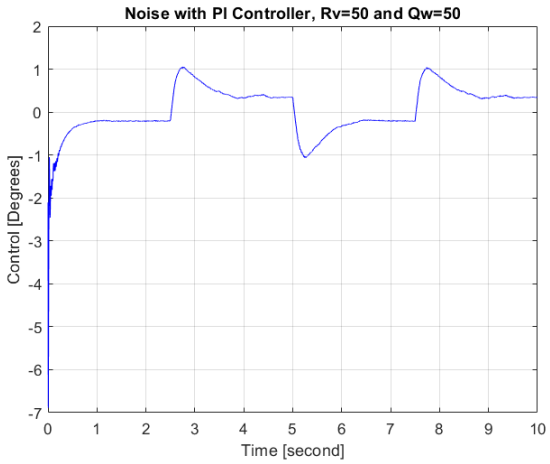


(b) Control Output

Figure 6: Noise (QwRv) Tuning with PI controller, R1000, Qw=100 and Rv=50

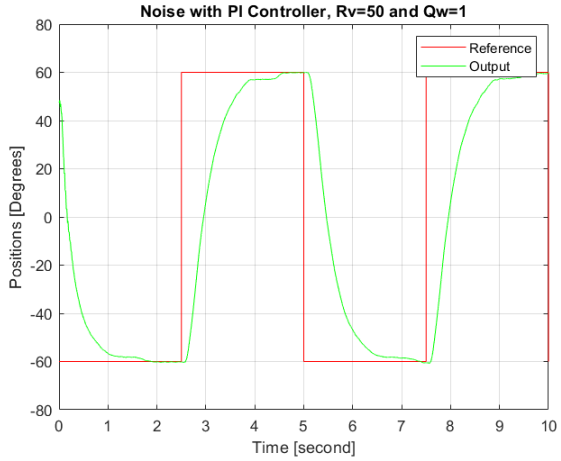


(a) Position of the flexible arm

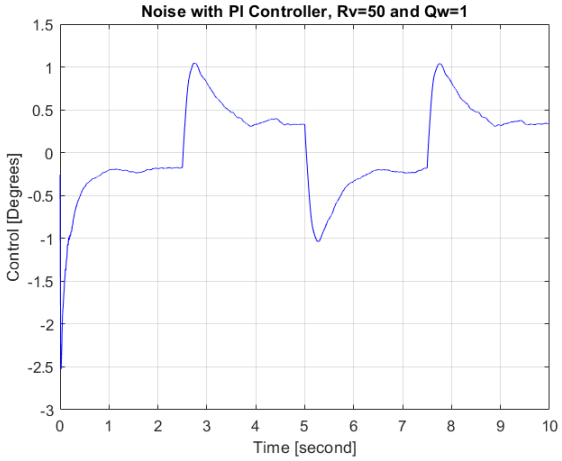


(b) Control Output

Figure 7: Noise (QwRv) Tuning with PI controller, R1000, Qw=50 and Rv=50

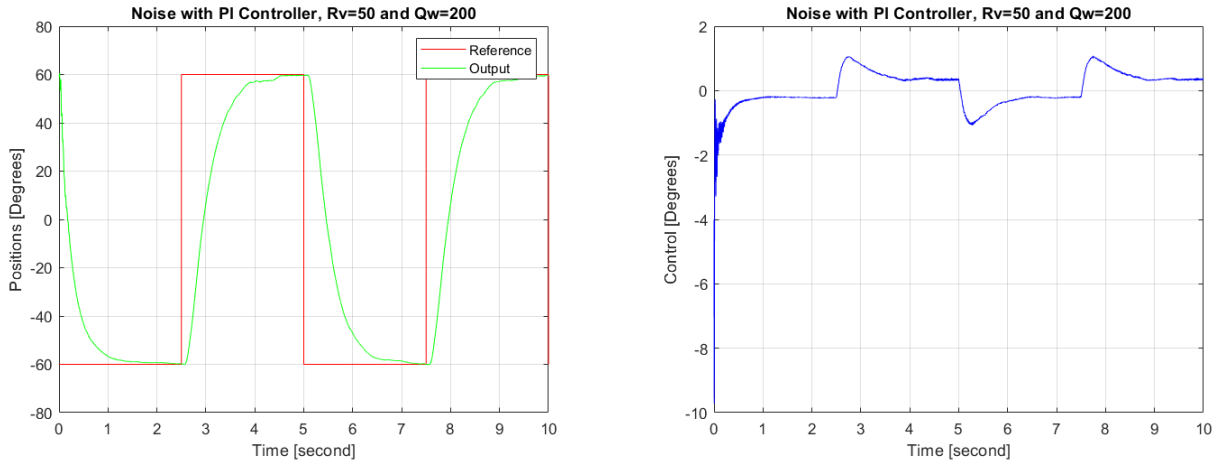


(a) Position of the flexible arm



(b) Control Output

Figure 8: Noise (QwRv) Tuning with PI controller, R1000, Qw=1 and Rv=50



(a) Position of the flexible arm

(b) Control Output

Figure 9: Noise (QwRv) Tuning with PI controller, R1000, Qw=200 and Rv=50

The choice of noise covariance matrices (Q and R) in the LTR framework is critical for achieving the desired control system performance. Properly tuning these matrices is essential for accurate state estimation, system stability, convergence of the LTR process, robustness, and efficient computational load. Based on this we tried out different values for Qw and Rv. We started with $Q_w = 100$ and $R_v = 1$. This gave a very noisy control value. We systematrically increased the values of R_v till we obtained a good control input when $R_v = 50$. Thereafter, we tuned the values of Q_w. We observed that reducing the value a lot gives a state estimation that is not smooth enough. However, increasing the value beyond $Q_w = 100$ doesn't give any noticeable effect on the plant.

Summarily, When Qw was sufficiently high, the system was able to follow the given signal as observed in Figure 9. When Rv was chosen to be very low the system is unable to stabilize itself as seen in Figure 4 and Figure 5. We obtained the best results with Rv = 50 and Qw = 100 as seen in Figure 6.

The Root Locus for the system is provided in Figure 10.

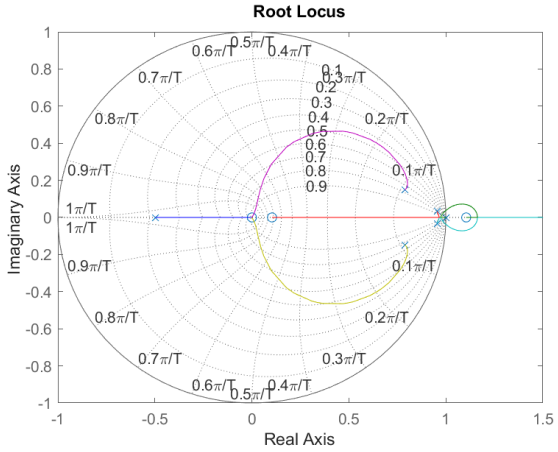


Figure 10: Root Locus Diagram

4.3 Frequency and Time Responses

Time and frequency responses of a system are important tools used for analyzing the behaviour of a system. The time response provides crucial insights into the behaviour of a system when transitioning from an initial state to a new state. This behaviour makes apparent some important metrics for assessing the performance of the system such as the rising time, the settling time, the overshoot, etc. The frequency response, on the other hand, provides some additional information about the system's characteristics. For example, with the Bode plot, information about the gain and phase margins -which are pivotal to the stability and robustness of a system- can be obtained. However, to determine the system responses, the transfer function is needed.

In this section, we will provide or derive the state-space equations for a system to be analyzed. Thereafter, we would utilize the MATLAB $tf()$ function to obtain the transfer function. We would analyze the close-loop plant, the plant with a P controller and the plant with a PI controller. We will also identify the positions of the closed-loop poles and zeros.

4.3.1 The Uncontrolled Closed-Loop Plant

In Section 2, we introduced the model for the uncontrolled plant and the identified parameters. The state-space equation for the plant is presernted in Equation 3and was used to derived the closed-loop transfer function[2]. Thereafter, we obtained the Pole-Zero plot in Figure 11 and the response plots in Figure 12 from the transfer function.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{3}$$

The Pole-Zero Diagram in Figure 11 indicates a pole at $z = 1$ which gives an accumulator behaviour. This pole will be the dominant pole and will cause the flexibar to spin continuously

[2]Further details are available in the script. See Figure 18

in the absence of a controller. This expected behaviour is obtained in the time response in Figure 12a. Likewise, we observe a high gain margin the frequency response in Figure 12b but this is not important yet. To prevent the endless spinning and make the system follow a specific reference, we introduce a controller in the closed loop.

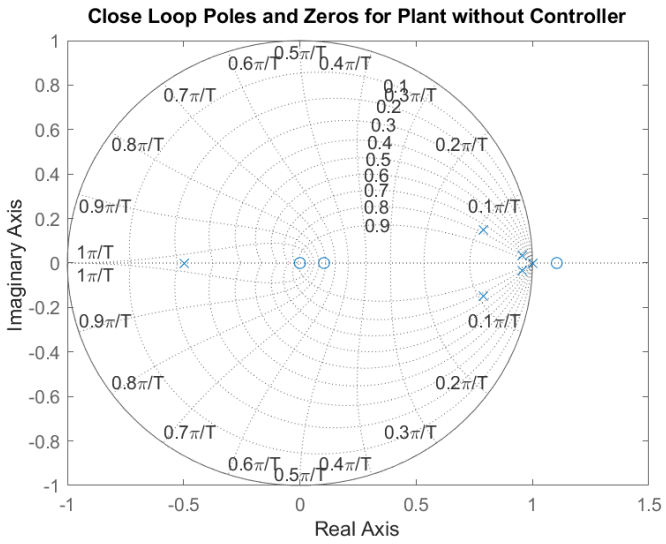
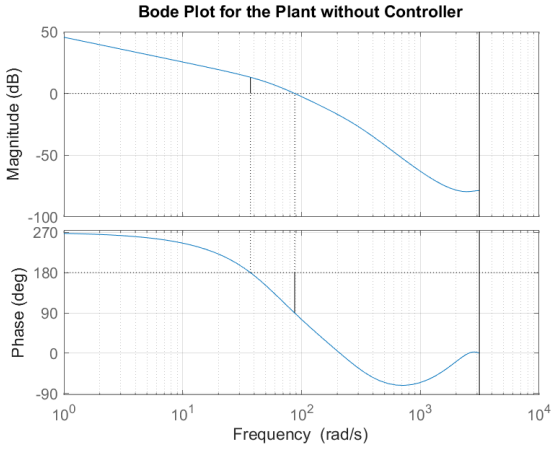


Figure 11: Poles and Zeros Diagram for the Uncontrolled Closed-Loop Plant

(a) Time Response for Step Input



(b) Frequency Response (Bode Plot).

Figure 12: Closed-Loop Responses for The Uncontrolled Plant

4.3.2 The Controlled Plant: Proportional Controller

To make the system stable, we introduce a Proportional Controller (henceforth referred to as P-Controller), $u_k = -Kx_k$. But since we do not have a full observation of the states, an observer is introduced (and the necessary analysis is done as already explained in Section

4.2) and the control law for the P-Controller is modified as $u_k = -K\hat{x}_k$[3]. Then, to make the controller track a reference, we modify the control law as $u_k = r_k - K\hat{x}_k$.

However, to analyse this system, we would derive a state-space model where the input is the reference r_k while the output is the measurement y_k. The equation for the plant and the current observer are derived in Equation 4 and presented in matrix form in Equation 5.

$$\begin{aligned} u_k &= r - K\hat{x}_k \\ x_{k+1} &= Ax_k + Bu_k \\ &= Ax_k + Br_k - BK\hat{x}_k \\ \hat{x}_{k+1} &= A\hat{x}_k + Bu_k + M(y_{k+1} - C\hat{x}_{k+1}) \\ &= A\hat{x}_k + Br_k - BK\hat{x}_k + MC(x_{k+1} - \hat{x}_{k+1}) \\ &= A\hat{x}_k + Br_k - BK\hat{x}_k + MCA(x_k - \hat{x}_k) \\ &= MCAx_k + (A - BK - MCA)\hat{x}_k + Br_k \\ y_k &= Cx_k \end{aligned} \tag{4}$$

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} &= \begin{bmatrix} A & -BK \\ MCA & A - BK - MCA \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} + \begin{bmatrix} B \\ B \end{bmatrix} r_k \\ y_k &= \begin{bmatrix} C & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} \end{aligned} \tag{5}$$

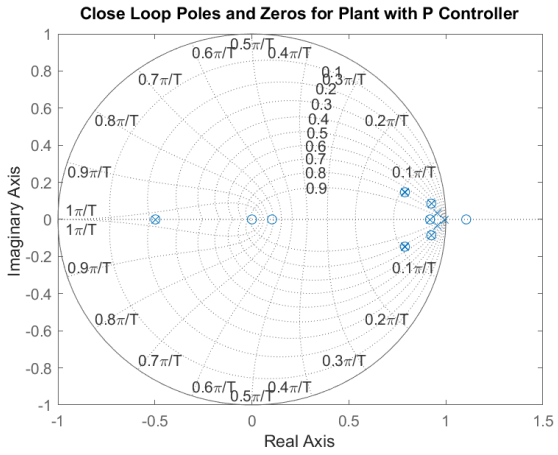
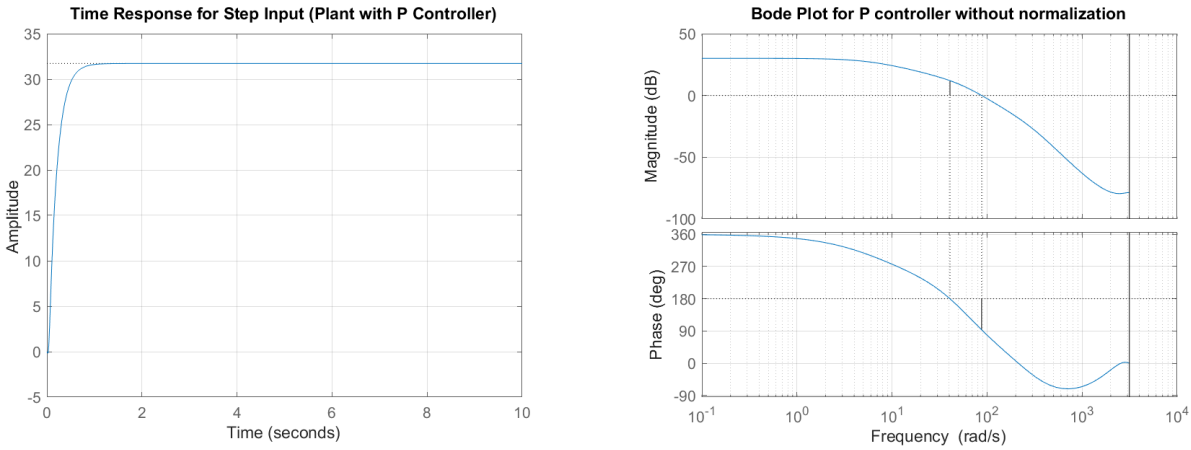


Figure 13: Poles and Zeros Diagram for the P-Controller in the Closed-Loop Plant

With the state-space model in Equation 5, we obtain the closed-loop transfer function in MATLAB and the three plots: pole-zero plot, time response and frequency response as shown in Figures 13, 14a and 14b. In the Pole-Zero plot in Figure 13, we now have all the poles within

[3]The controllability analysis was performed before continuing with the controller design. See the script in Figure 1

the unitary circle which signifies stability. In the time response plot in Figure 14a, we also have a "good" output - a good response shape and absence of oscillation. However, the time response amplitude is not 1 unit and the frequency response in Figure 14b signifies a positive gain margin which shows non-stability. This controller was not tested on the flexibar bar for this reason.



(a) Time Response for Step Input

(b) Frequency Response (Bode Plot).

Figure 14: Close Loop Responses for the P Controller without Normalized Gain

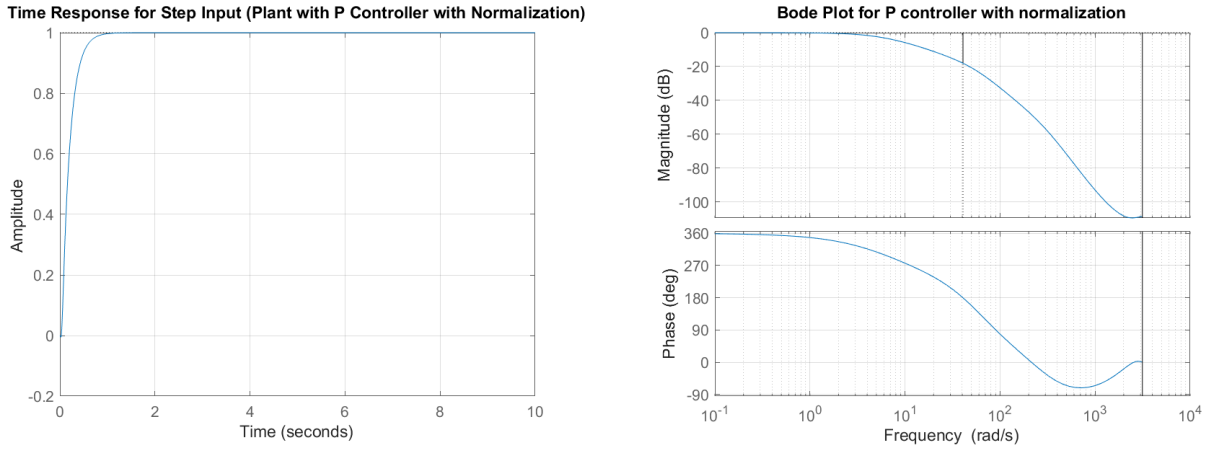
4.3.3 The Controlled Plant: Normalized Proportional Controller

To enhance the static gain of the closed-loop system, a compensatory gain N is introduced, ensuring that the resulting static gain becomes unitary. The computation of this gain is performed within the Controller Script, and the modified P-Controller model is represented by Equations 6 and 7.

$$\begin{aligned} u_k &= Nr - K\hat{x}_k \\ x_{k+1} &= Ax_k + BNr_k - BK\hat{x}_k \\ \hat{x}_{k+1} &= MCAx_k + (A - BK - MCA)\hat{x}_k + BNr_k \\ y_k &= Cx_k \end{aligned} \tag{6}$$

$$\begin{aligned} \begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} &= \begin{bmatrix} A & -BK \\ MCA & A - BK - MCA \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} + \begin{bmatrix} BN \\ BN \end{bmatrix} r_k \\ y_k &= \begin{bmatrix} C & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} \end{aligned} \tag{7}$$

Upon analyzing the normalized P-Controller, the time response depicted in Figure 15a illustrates that the simulated system response adequately tracks the specified reference signal. Additionally, the frequency response plot in Figure 15b now displays a negative gain margin, indicating system stability. This controller eventually on the plant and the results will be presented in the subsequent sections of the report: Pre-Filter Analysis and Performance Evaluation.



(a) Time Response for Step Input

(b) Frequency Response (Bode Plot).

Figure 15: Close Loop Responses for the P Controller with Normalized Gain

4.3.4 The Controlled Plant: Proportional-Integral Controller

During the first testing phase of the controller, we observed a steady-state error in the output of the plant. The plant tracks the specified reference but with an amount of offset. To correct this problem, we opted for a Propotional-Intergal Controller (hereafter referred to as PI-Controller). The PI-Controller is defined by the control law $u_k = -Kx_k - K_I(y_k - r_k)$. The second term of the control law compensates for the offsets. To analyse the response of this controller, we derived the modified state-space equations as shown in Equations 8 and 9. The state-space equations are converted to transfer function and thereafter used to obtain the time and frequency responses in Figure 17. The time and frequency responses are very similar to the ones obtained from Normalized P-Controller - the static gain appears to be unitary and the bode plot shows a negative gain margin. However, their performances on the plant are significantly different. This would be made apparent in later sections of the report.

$$\begin{aligned}
u_k &= -K\hat{x}_k - K_I(y_k - r_k) \\
x_{k+1} &= Ax_k + Bu_k \\
&= Ax_k - BK\hat{x}_k - BK_Iy_k + BK_Ir_k \\
&= (A - BK_IC)x_k - BK\hat{x}_k + BK_Ir_k \\
\hat{x}_{k+1} &= A\hat{x}_k + Bu_k + M(y_{k+1} - C\hat{x}_{k+1}) \\
&= A\hat{x}_k - BK\hat{x}_k - BK_Iy_k + BK_Ir_k + MCA(x_k - \hat{x}_k) \\
&= A\hat{x}_k + Br_k - BK\hat{x}_k + MCA(x_k - \hat{x}_k) \\
&= (MCA - BK_IC)x_k + (A - BK - MCA)\hat{x}_k + BK_Ir_k \\
y_k &= Cx_k
\end{aligned} \tag{8}$$

$$\begin{aligned}
\begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} &= \begin{bmatrix} A - BK_IC & -BK \\ MCA - BK_IC & A - BK - MCA \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} + \begin{bmatrix} BK_I \\ BK_I \end{bmatrix} r_k \\
y_k &= \begin{bmatrix} C & \mathbf{0} \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix}
\end{aligned} \tag{9}$$

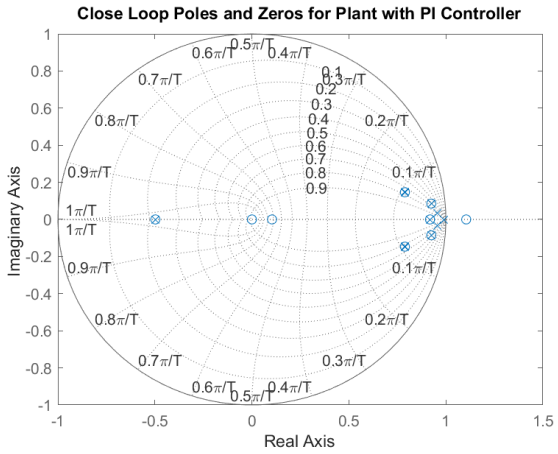
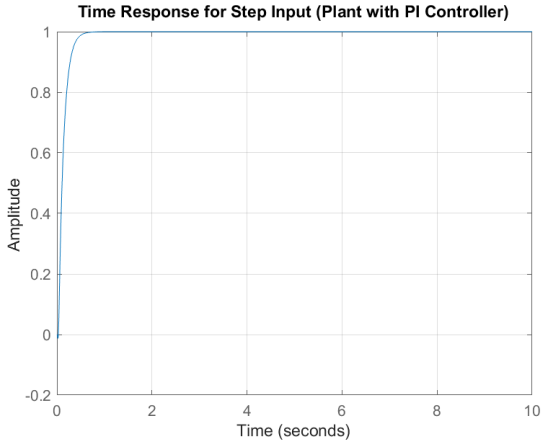
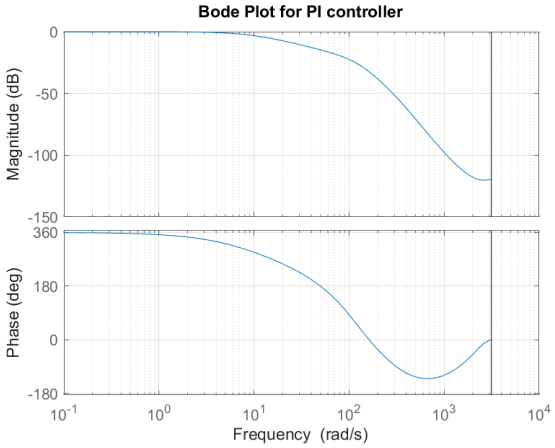


Figure 16: Poles and Zeros Diagram for the PI-Controller in the Closed-Loop Plant



(a) Time Response for Step Input



(b) Frequency Response (Bode Plot).

Figure 17: Close Loop Responses for the PI Controller

ControllerAnalysisScript2.m

```
close all; clc;

% Load Controller Parameters
load('controller_params')

%% Time and Frequency Response

% Plant Transfer Fxn
plant_sys = ss(A, B, C, D, Ts);

fig = figure;
step(feedback(plant_sys, 1), 10)
grid
title('Time Response for Step Input (Plant without Controller)')
saveas(gcf,'ResponsePlots\TimeResponse_Plant.png');

fig = figure;
bodeplot(feedback(plant_sys, 1))
grid
title('Bode Plot for the system without controller')
saveas(gcf,'ResponsePlots\BodePlot_Plant.png');

%% Close Loop Transfer Fxn (Plant, Observer and P Controller)

A_est = A - M*C*A;
B_est = B - M*C*B;

A_p = [A, zeros(size(A));
    M*C*A, A_est - B_est*Kx];
B_p = [B; M*C*B];
C_p = [zeros(size(Kx)) Kx];
D_p = 0;

p_contr_sys = ss(A_p, B_p, C_p, D_p, Ts);

fig = figure;
step(feedback(p_contr_sys, 1), 10)
grid
title('Time Response for P controller without normalization')
saveas(gcf,'ResponsePlots\TimeResponse_P_Controller.png');

fig = figure;
bode(feedback(p_contr_sys, 1))
grid
title('Bode Plot for P controller without normalization')
saveas(gcf,'ResponsePlots\BodePlot_P_Controller.png');
```

(a) Script from Line 1 - 46

ControllerAnalysisScript2.m

```
%% Close Loop Transfer Fxn (Plant, Observer and P Controller with Norm)

A_pnorm = [A, -B*Kx;
    M*C*A, A_est - B_est*Kx - M*C*B*Kx];
B_pnorm = [B; B_est+M*C*B];
C_pnorm = [C, zeros(size(C))];
D_pnorm = 0;

pnorm_contr_sys = ss(A_pnorm, B_pnorm, C_pnorm, D_pnorm, Ts);

fig = figure;
step(Nbar*feedback(p_contr_sys, 1), 10)
grid
title('Time Response for P controller with normalization')
saveas(gcf,'ResponsePlots\TimeResponse_P_Controller_Norm.png');

fig = figure;
bode(Nbar*feedback(p_contr_sys, 1))
grid
title('Bode Plot for P controller with normalization')
saveas(gcf,'ResponsePlots\BodePlot_P_Controller_Norm.png');

%% Close Loop Transfer Fxn (Plant, Observer and PI Controller)

A_pi = [A, zeros(size(A)), zeros(n_states, 1);
    M*C*A-B_est*Ki_int*C, A_est - B_est*Kx_int, B_est*Ki_int;
    zeros(1, 2*n_states), 1];
B_pi = [B; M*C*B; 0];
C_pi = [Ki_int*C, Kx_int, -Ki_int];
D_pi = 0;

pi_contr_sys = ss(A_pi, B_pi, C_pi, D_pi, Ts);

fig = figure;
step(feedback(pi_contr_sys, 1), 10)
grid
title('Time Response for Step Input (Plant with PI Controller)')
saveas(gcf,'ResponsePlots\TimeResponse_PI_Controller.png');


fig = figure;
bode(feedback(pi_contr_sys, 1))
grid
title('Bode Plot for PI controller')
saveas(gcf,'ResponsePlots\BodePlot_PI_Controller.png');
```

(b) Script from Line 48 - 92

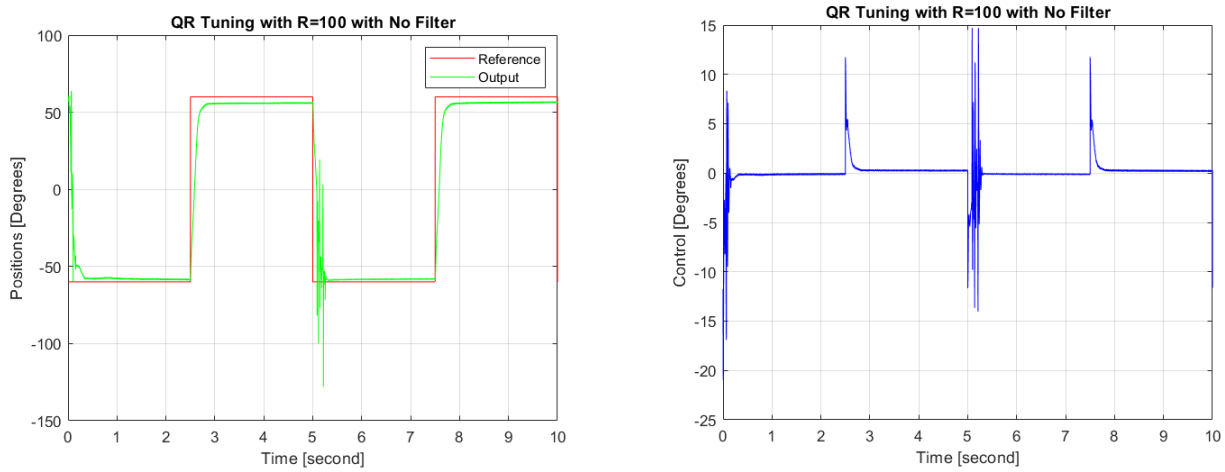
Figure 18: Scripts for Plotting the Time and Frequency Responses

4.4 Effect of the Inclusion of a Pre-Filter

One major drawback of the P-Controller when it is tested on the plant is that it responds quickly to sudden changes in the reference signal and causes the plant to have repeated cycles of vibration before stabilizing. This behaviour is observed in Figure 19. It can be observed that the control variable u_k has a very noisy response to this situation. This behavior occurs during a sudden transition of the reference signal from a value to another. Hence, to mitigate this undesired effect, we introduce a filter (with a static gain) that smoothen the transition of the reference signal from the initial value to the new value. This gives the controller a better signal to track and improves the response of the plant. The block diagram for the filter has been provided in the Simulink block diagram (see Figure 3a).

We tested the use of a first-order and a second-order filter. The transfer function for the filter is given in Equation 10 where n is the order of the filter. We selected a value of $\alpha = 0.9$.

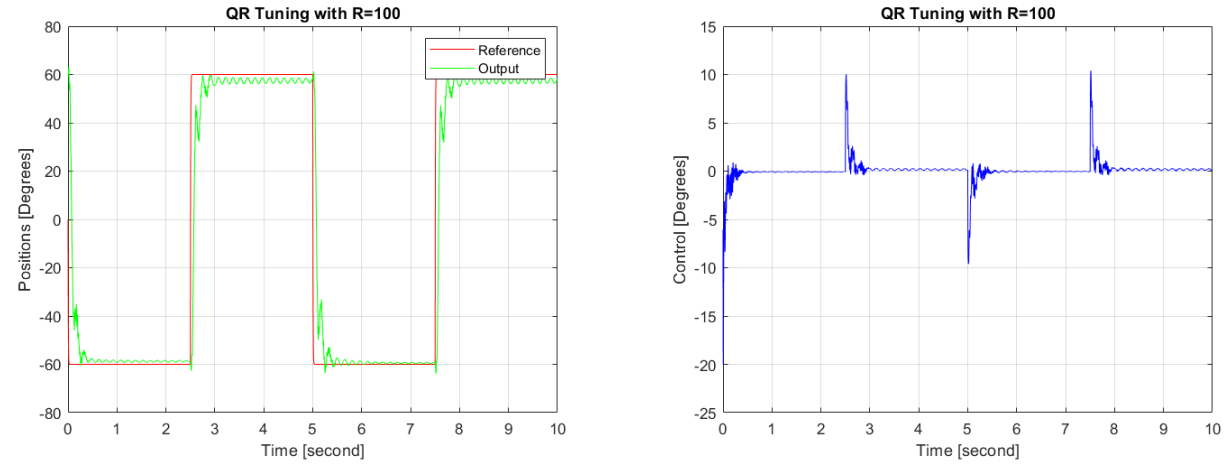
$$G(z) = \frac{(1-\alpha)^n}{(z-\alpha)^n} \tag{10}$$



(a) Position of the flexible arm **(b)** Control Output

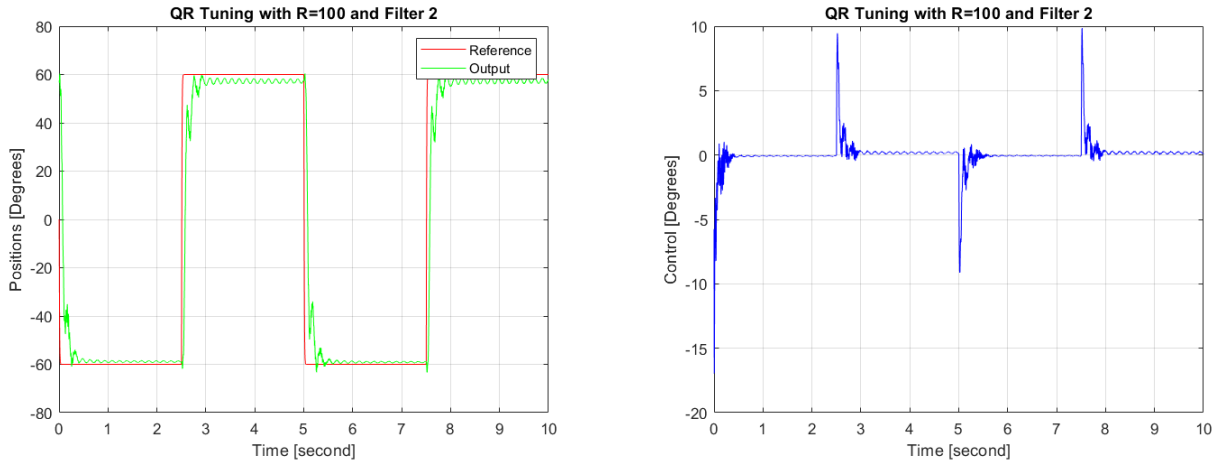
Figure 19: LQR Tuning with P controller, R100 and No Filter

We tested the first-order and the second-order filter on the plant with a (Normalized) P-Controller and the resulting system response is given in Figures 20 and 21 respectively. Some noise is observed at the beginning in the plot of the control variable but this is neglected because the noise is only due to the inertia of the system at the rest state. In consequent state transitions, it is observed that the system moved towards the reference systematically with intermittent stoppages along its path to avoid erratic behaviour. In the situation, it can be seen that the controller is less noisy which signifies that the actuator would be less stressed compared to the earlier situation when the filter was absent. This response is observed in the first-order and the second-order filters.



(a) Position of the flexible arm **(b)** Control Output

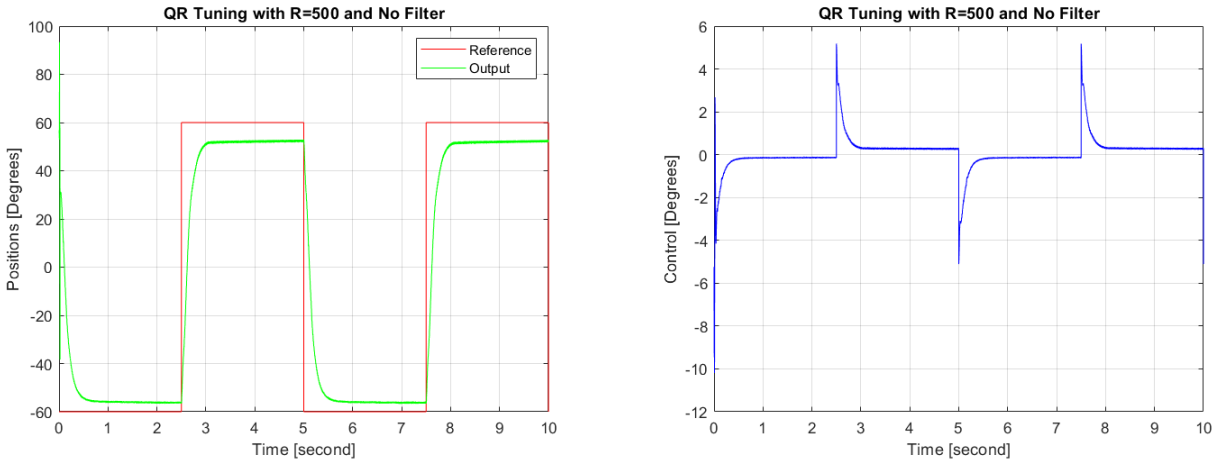
Figure 20: LQR Tuning with P controller, R100 and Filter 1



(a) Position of the flexible arm **(b)** Control Output

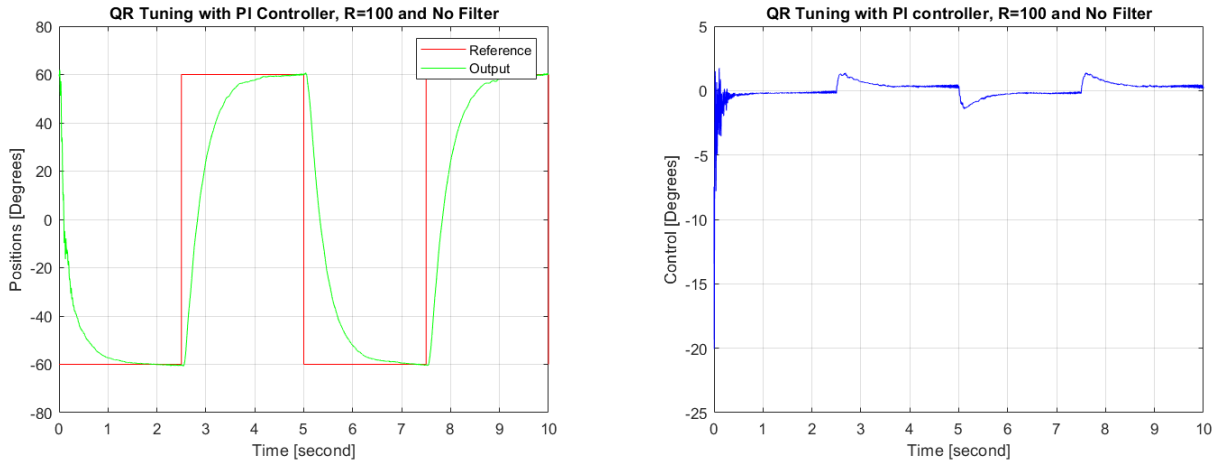
Figure 21: LQR Tuning with P controller, R100 and Filter 2

However, instead of using a filter, we observed that putting more penalty on the controller by increasing the value of R also improves the system response. This is shown in Figure 22. Likewise, we observed that our PI Controller does not require a filter as it is robust to the transitions (perhaps due to the presence of the integrator in the controller).



(a) Position of the flexible arm **(b)** Control Output

Figure 22: LQR Tuning with P controller, R500 and No Filter



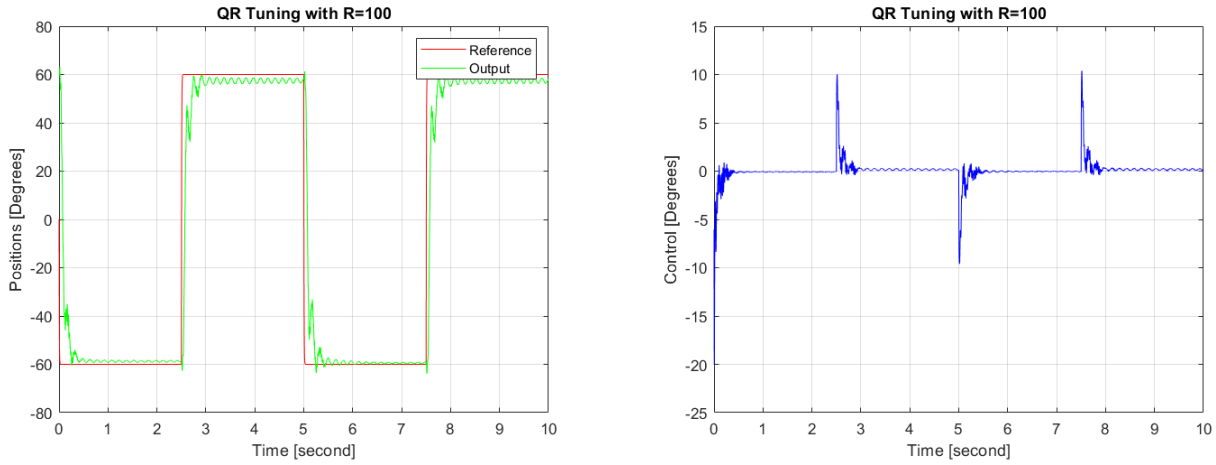
(a) Position of the flexible arm **(b)** Control Output

Figure 23: LQR Tuning with PI controller, R100 and No Filter

Summarily, the pre-filter is important when a P-Controller is used with a fast response. Without the pre-filter in this situation, the plant gives an erratic behavior and could lead to internal or mechanical defects in the system. The pre-filter helps to prevent this situation by smoothing the change in the reference over a window of time. However, the erratic behavior could also be avoided by penalizing the controller. Also, the pre-filtering is not necessary for the PI Controller since it is already robust to this scenario.

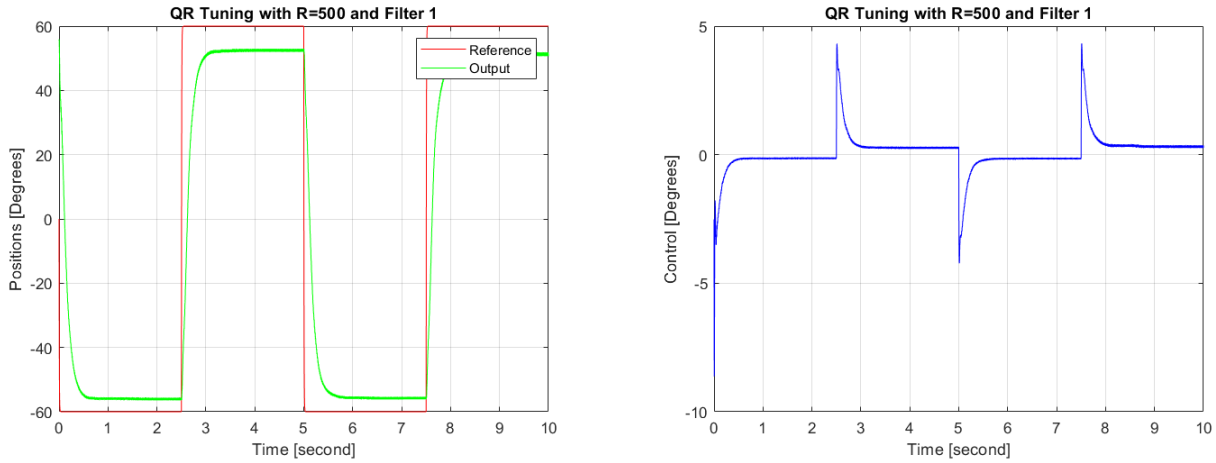
5 Performance Evaluation

Despite the inherent limitations of the plant, which make it unsuitable for applications requiring rapid position changes, it is noteworthy how swiftly and precisely this system can be manipulated using various control theory techniques. However, it should be understood that the plant's dynamics prevent it from accurately tracking a reference signal, such as a square wave, due to the need to accelerate the mass and account for the ruler's motion. The goal to make the system reach the desired value with minimum number of oscillations was achieved by tuning LQR parameters particularly the gain $\mathbf{R}$ and adding either a first order filter or a second order filter. Experiments were performed using different values of R, type of controller and filter and the resulting output was observed. Figure 24 and Figure 25 shows a plot obtained from using a first order filter (filter 1) and only a proportional controller with different values of R. While it is effective in quickly reducing error, it tends to introduce noise and may not entirely eliminate steady-state error. However increasing the cost on R improves the controller. But the steady-state error becomes more apparent. The choice of LQR tuning parameters and the use of a filter can help mitigate some of these issues, but there is often a trade-off between noise reduction, steady-state performance, and control effort. Fine-tuning the controller and considering additional control strategies like Integral (I) or Derivative (D) control can further improve system performance.



(a) Position of the flexible arm **(b)** Control Output

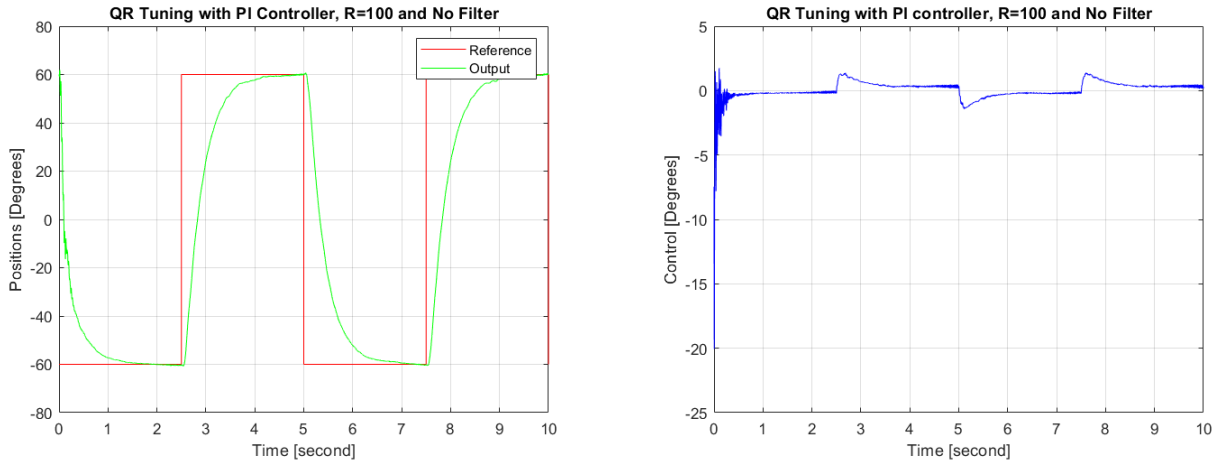
Figure 24: LQR Tuning with P controller, R100 and Filter 1



(a) Position of the flexible arm **(b)** Control Output

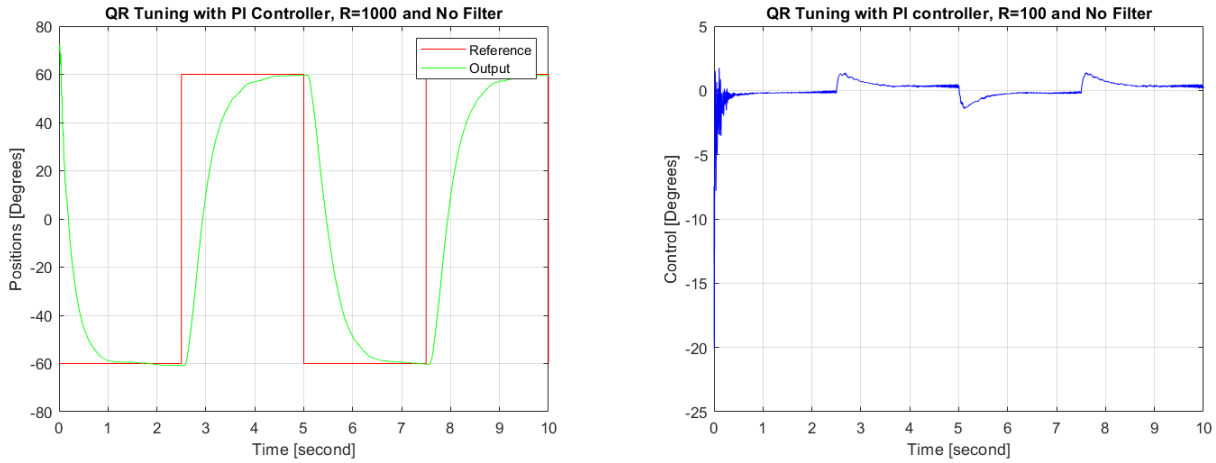
Figure 25: LQR Tuning with P controller, R500 and Filter 1

The output plot 26 and 27 shows that the PI controller effectively regulates the system's response and brings it to the desired setpoint. The noise observed at the beginning of the response is temporary and can be considered negligible because it does not significantly impact the overall performance of the control system. This is a common characteristic of control systems during the transient phase when the controller is actively working to establish stability and accurate tracking of the setpoint. The PI controller quickly corrects any deviations from the setpoint and ensures that the system tracks the desired trajectory accurately.



(a) Position of the flexible arm **(b)** Control Output

Figure 26: LQR Tuning with PI controller, R100 and No Filter



(a) Position of the flexible arm **(b)** Control Output

Figure 27: LQR Tuning with PI controller, R1000 and No Filter

6 Conclusion

In conclusion, we developed a controller that was able to adequately track a given reference. We used a PI-Controller because it is very robust to sudden change in the reference and it corrects steady-state errors. We recommend setting the value of the LQR to $R = 1000$ and the tuning of the Kalman Filter to $Q_w = 50$ and $R_v = 50$. These are the best values we obtained.