24-774 Advanced Control System Integration
Lab 2: State Space Control

James Hong (*peizeh*) Aomeng Li (*aomengl*) Kathy Tang (*chuyit*)
Dajun Tao (*dajunt*) Shaobo Wang (*shaobow*)

October 1, 2022

Abstract

In this lab, an observer was added to enable the pendulum arm to maintain the upright position. A newly linearized plant model was created at the upright position and was controlled by a LQR controller. After the initial simulation, the closed-loop model was tested on the Quanser hardware platform. Due to hardware issues, only a video was recorded to capture the inverted pendulum tracking result. Further Simulink tasks were performed.

1 Introduction

The purpose of this lab is to practice hardware implementation of state space controllers by designing both a state estimator and a LQR controller for inverse pendulum control. Basically, the team worked through this lab with some degree of parallelization. The responsibility was assigned as follows:

Team Member	Tasks	People
1	State Estimator 1	Shabo Wang
2	State Estimator 2	Kathy Tang
3	LQR 1	James Hong
4	LQR 2	Dajun Tao
5	LQR 3	Aomeng Li

Table 1: Work Breakdown

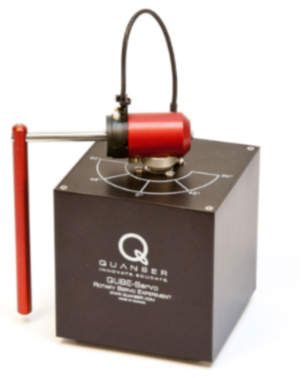


Figure 1: Quanser Qube Servo System.

2 State Estimator Design

2.1 Observer Design

In order to implement a full state-feedback controller to the inverted pendulum system, we need to rely on the observer to estimate the two unmeasured angular velocity states. We purposed two methods for states estimation: Luenberger Observer and Kalman filter. The design process of both methods are discussed in the following sections. The performance of observer-based methods is compared with the derivative and low-pass filter method through simulations and hardware tests.

2.1.1 Observer Design Process

Due to the separation principle, we can individually design the controller and the observer. To better evaluate the performance of the observer, a stable system with zero control input is used. The performance is evaluated base on the difference in the time response between states ground truth and the estimated states.

For the downward position model a definition of downward zero position is used so that the steady state of the linearized LTI system can remain as the downward position. This move allows us to directly use the LTI model in our observer design. Otherwise, we need to manually apply offset to convert the pendulum arm angle definition. Similarly, the upright position will be used for the inverted pendulum model and its controller design.

The observer can be interpret as a MIMO state feedback regulator which takes the estimated states as its states and regulates the error between the measured output y and the estimated output $\hat{y}$. The block diagram is shown in Figure 2.

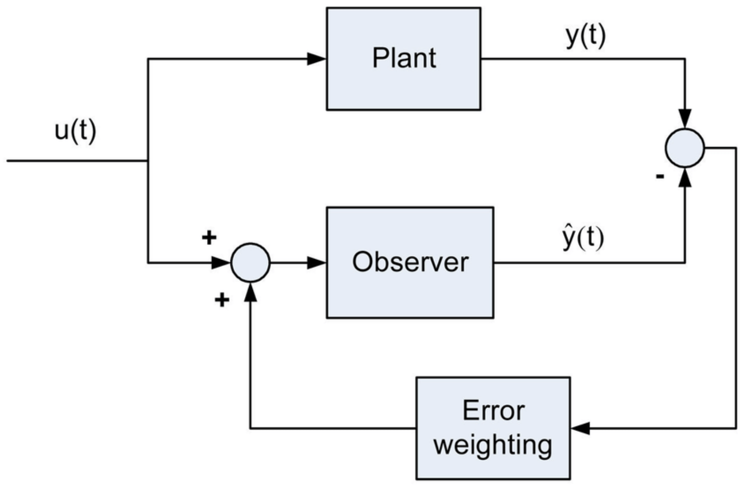


Figure 2: The Luenberger Observer Model Estimated States by Regulating Errors between Measured and Estimated Output.

The entire observer can also be represented as a MIMO LTI system whose state space model is:

$$\dot{\hat{x}} = (A - LC)\hat{x} + \begin{bmatrix} B & L \end{bmatrix} \begin{bmatrix} u \\ y \end{bmatrix}$$

$$\begin{bmatrix} \hat{y} \\ \hat{x} \end{bmatrix} = \begin{bmatrix} C \\ I \end{bmatrix} \hat{x}$$

The main goal is to design the weight matrix L for the above system to perform good state estimations.

Luenberger Observer Design

The Luenberger observer rely on pole placement to design the weight matrix L. For our fourth-order pendulum system, we can modify the system response by placing four poles at different LHP positions. A guideline of having observer poles $4 - 10$ times faster than the controller pole is used in the inverted pendulum control. Therefore, since the LQR controller of the inverted pendulum system has dominant close-loop system poles near -5.6 given a rough design for the later sections, we place the pole at $-50, -51, -52, -53$ to capture the dynamics of the current system.

Kalman Filter Design

The Kalman filter modeled the process noise and the measurement noise as Gaussian distributed. Based on our assumptions on the white noise signal, we can design the Q and R matrices as:

$$Q = \frac{cov(t)}{T_s}, \; R = \begin{bmatrix} \frac{cov(\theta)}{T_s} & O \\ O & \frac{cov(\alpha)}{T_s} \end{bmatrix}$$

In real case, we need to run experiments on the hardware to acquire the covariance of the measurements or input. Therefore, it is kind of unorthodox to directly use the noise information. But if the noise info we acquire from testings is accurate enough, then we can expect the performance to be similar to our simulation.

2.1.2 Comparison with LPF

The Simulink model is created to evaluate the performance between different state estimation methods. The white noise in both input and measurement are included to better mimic the real scenario.
The white noise is assumed to have a sampling time of 2 ms and a covariance (energy) of 10^{-9}.The low-pass filter uses the default filter frequency of 50 rad/s. Since we are not using the LQR controller for this simulation, the Luenberger observer's poles are placed at $-20, -21, -22, -23$ instead. The Q and R matrices for the Kalman filter design is designed according to the white noise properties assumption. The initial pendulum arm angle is set to be 10 Deg to better fit into the linear range of the system dynamics.

```
% Downward Init condition
IC = [0; 0; pi/18; 0];
% Noise info
Tn = 2e-3;
cov_u = 1e-9;
cov_theta = 1e-9;
cov_alpha = 1e-9;
% LPF
w = 50;
% CT Luenberger Observer
P_LB = [-20 -21 -22 -23];
```

```
L_LB = place(A_down', C', P_LB)';
Obs_LB = ss(A_down-L_LB*C,[B_down L_LB], eye(4), []);
% CT Kalman Filter
Q = cov_u/Tn;
R = [cov_theta/Tn 0; 0 cov_alpha/Tn];
[¬,L_KF,¬] = kalman(sys_down, Q, R);
Obs_KF = ss(A_down-L_KF*C,[B_down L_KF], eye(4), []);
```

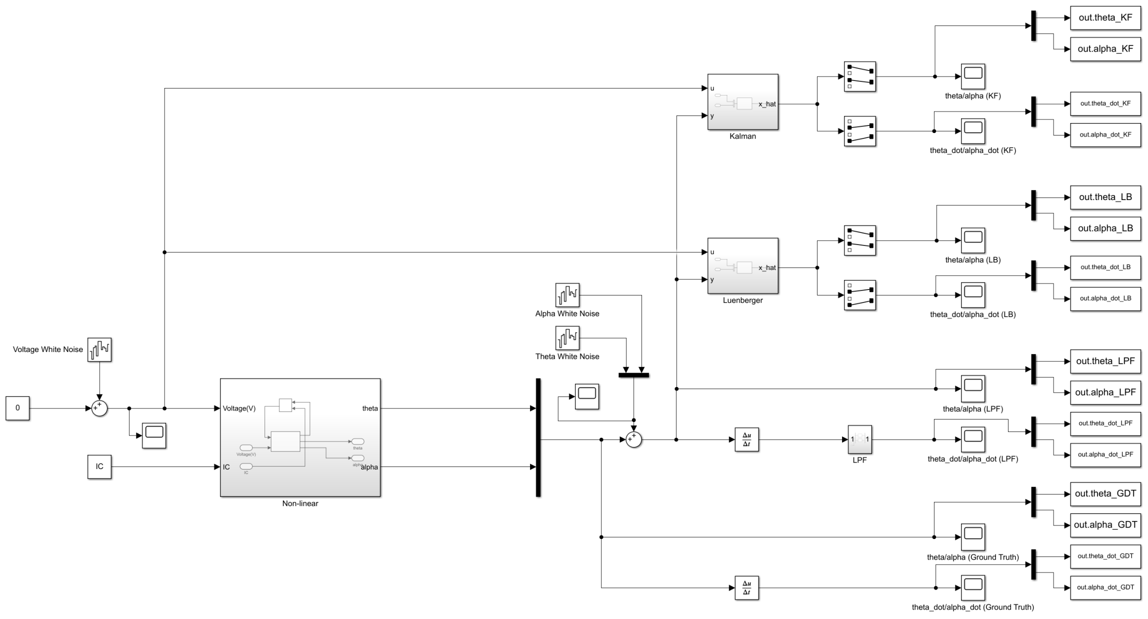


Figure 3: This is the Layout of Simulink Model for State Estimation Methods Evaluations including Both Luenberger Observer and Kalman Filter.

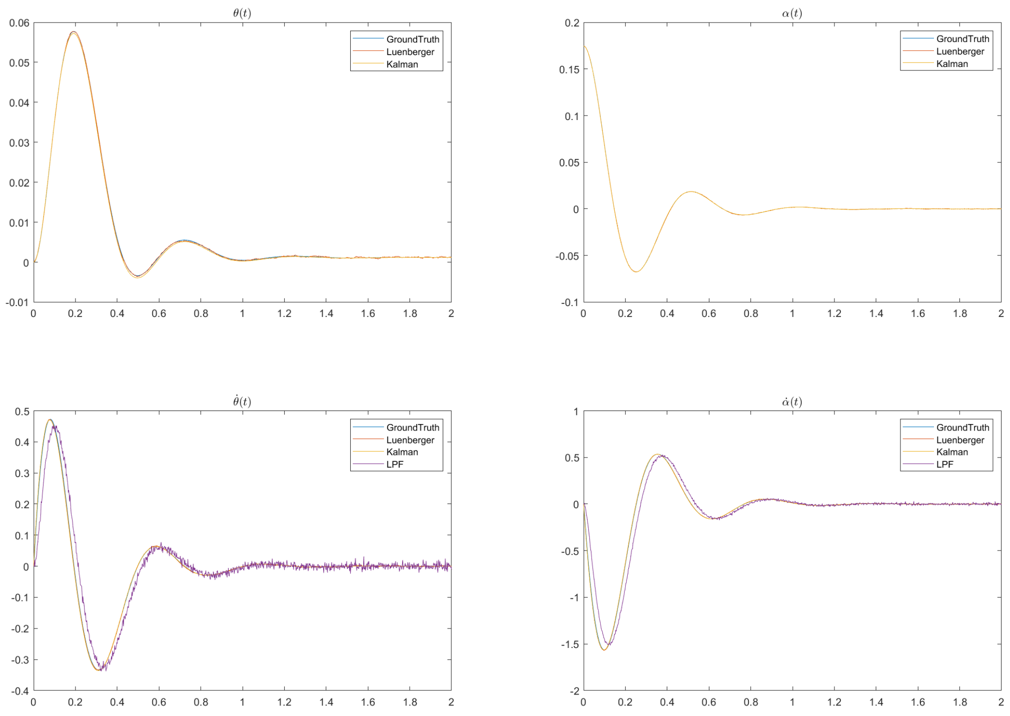


Figure 4: For the Zero-input System Time Response, the Simulation Shows the Results for Ground Truth Measurement, Luenberger Observer Estimation, Kalman Filter Estimation.

The simulation results for the downward-pendulum with zero-input is shown in Figure 4. For a initial condition like this, both Kalman filter and Luenberger observer perform well in improving the measurement quality and estimating the unmeasured state. It can be seen that the state estimation is almost identical to the ground truth where no white noise is applied. For the LPF, it will absolutely introduce some phase shift to the system, and its performance is limited by the energy of the noise. When we keep increasing the noise covariance, then the LPF will fail to give reasonable result. But if the noise energy is small, which is usually the case, then the LPF is still acceptable.

If we keep increasing the initial pendulum arm angle, then it is observed that the Kalman filter will begin to perform worse. Part of this is because the linaerized model is no longer close to the real system model, which makes the prediction process even worse. Also take into account the fact that we do not have the information of noise for granted, we should choose the Luenberger observer for the hardware testings.

The simulation results for the upright balancing and tracking pendulum is shown in Figure 5. It is noticed that based on the current noise energy level, the LPF is still sufficient enough to maintain stable state feedback control. The Luenberger observer will in this case require much faster poles to work together with the LQR controller. The Kalman filter is observed to be quite sensitive to the initial conditions, but it will always converge back to the ground truth even the prior correction can be intense.

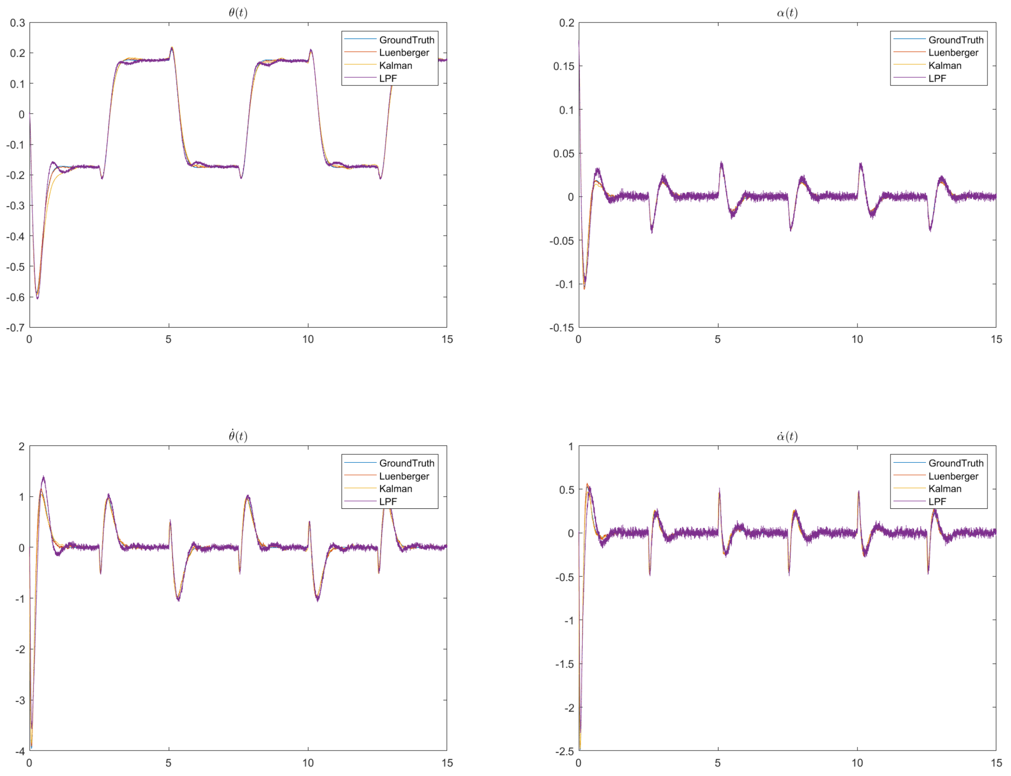


Figure 5: For the Balancing and Tracking System Time Response, the Simulation Shows the Results for Ground Truth Measurement, Luenberger Observer Estimation, Kalman Filter Estimation.

2.2 Quantization Effect

During A/D and D/A conversions, quantization errors would be introduced into the system. This error is caused by the process of rounding of the fractional part of a signal and also rounding of the overflow of the signal range. To simulate the quantization effect in the observer model, two quantization blocks are added to both the motor (θ) and pendulum (α) angles as shown below:

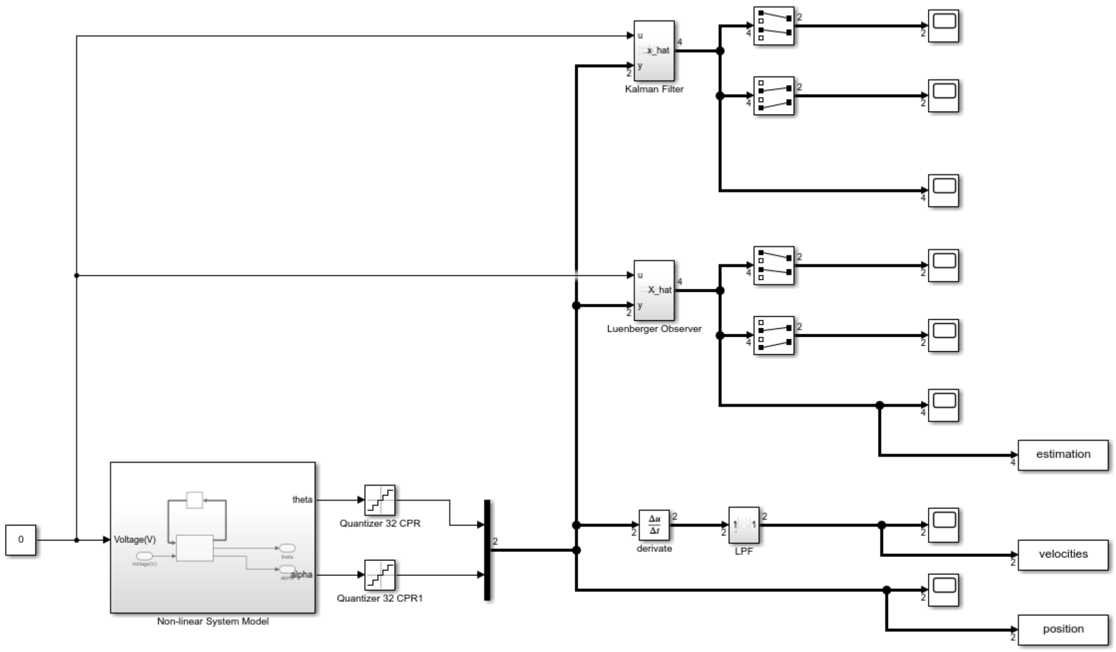
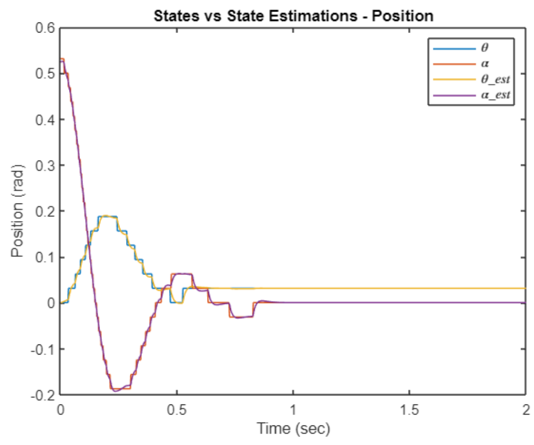


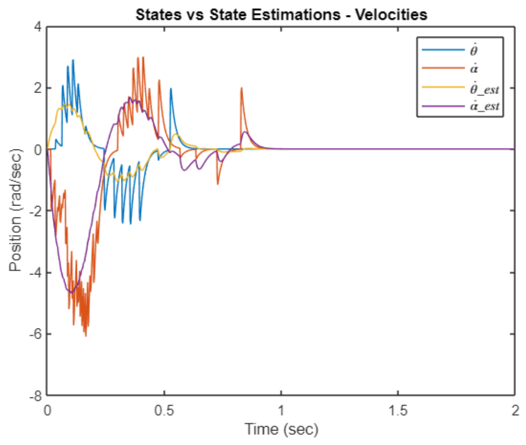
Figure 6: Two quantization blocks are added to both motor angle (θ) and pendulum angle (α) to simulate the quantization effect on the system and system observer model. To realize different counters per revolution of encoders, we set different values of quantization interval (q).

2.2.1 Effect of Encoder Resolution

For the encoder with 32 counts per revolution, each quantization interval (q) is represented by a 5-bit word. Similarly, for the encoder with 1024 counts per revolution, each quantization interval is represented by a 10-bit word. The observer's converge for encoders with each CPR is shown below:

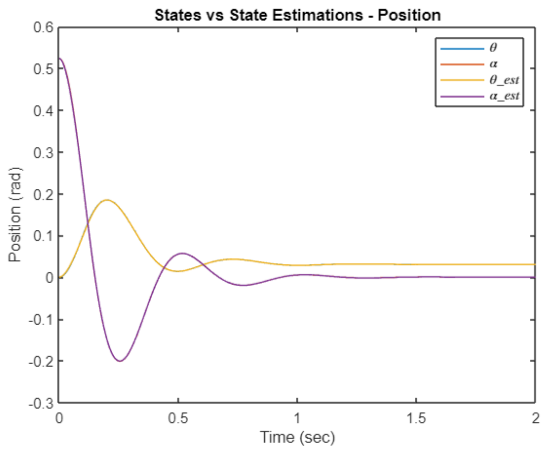


((a)) Position Measurement and Estimation

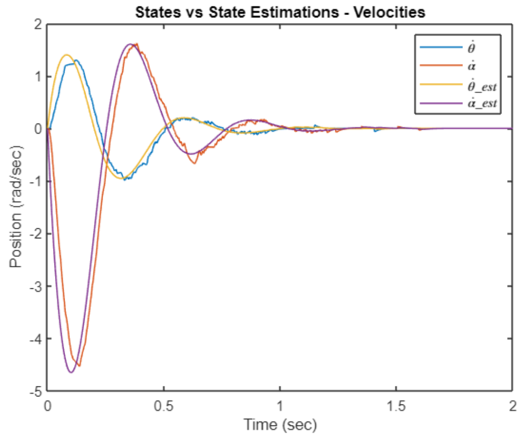


((b)) Velocity Measurement and Estimation

Figure 7: This is the comparison between between actual output states and output estimation when we simulates an encoder with 32 CPR. Some quantization errors are introduced to the system model for both position and velocity measurement and estimation.



((a)) Position Measurement and Estimation



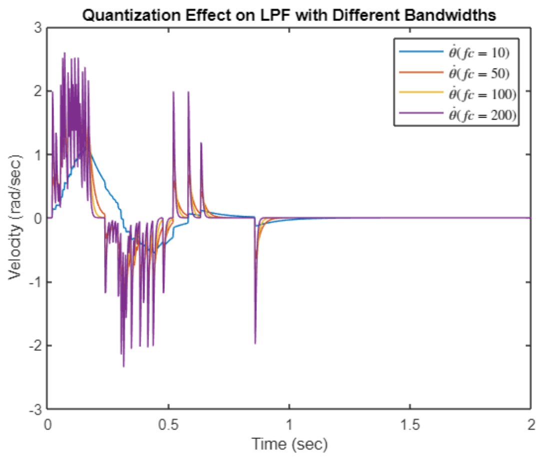
((b)) Velocity Measurement and Estimation

Figure 8: This is the comparison between between actual output states and output estimation when we simulates an encoder with 1024 CPR. Some quantization errors are introduced to the system model for both position and velocity measurement and estimation.

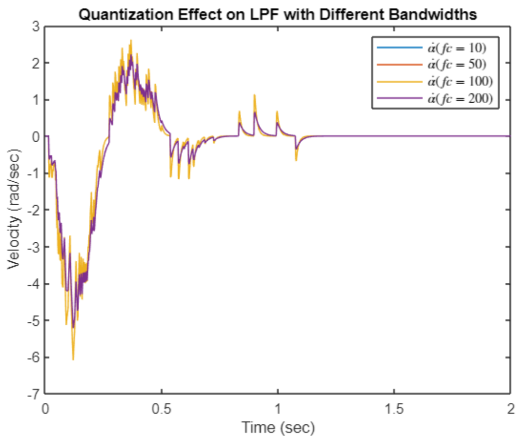
As we can see from the Figures 7 and 8, the higher CPR of encoder, the smaller quantization intervals, the smaller quantization errors. Therefore, if we were to redesign our observer given these quantization levels, the observer with lower CPR encoder resolution should be designed to handle more accuracy errors between actual and estimated states, partially due to more quantization errors per interval and their accumulation caused by encoder resolution.

2.2.2 Quantization Effect on LPF Design

Comparison between actual velocity outputs and state estimations for velocities in Figure 7 (b) and 8 (b) shows the effect of quantization on the accuracy of observer design vs the the derivate plus low pass filter. Although both observer and the derivate + LPF both take into accounts of quantization effect, the derivate + LPF has more oscillations in output responses.
We set low pass filter frequency at 50 rad/sec with quantization interval q at 0.01. The quantization effect on different low pass filter bandwidths at 10 rad/sec, 50 rad/sec, 100 rad/sec, and 200 rad/sec is shown below:



((a)) theta



((b)) alpha

Figure 9: To illustrate the quantization effect on LPF, system responses of the derivatives of both motor (θ) and pendulum (α) angles with different LPF bandwidths are compared.

With the same quantization effect, the higher LPF bandwidth, the more quantization error was introduced to the observer estimation. The quantization interval should set small enough that introduces less quantization error to the system with LPF. In other words, the low pass filter bandwidth needs to set small enough for prevent more quantization errors.

3 LQR Controller Design for Inverted Pendulum

3.1 LQR Controller Design

To make the pendulum stay in the upright position and also track the setpoint changes in motor angle, we need to use the linearized continuous time state space model to design an LQR controller to stabilizes the system.

```
Q = diag([10 0, 1, 0]); % states weights
R = 1e0*eye(num_ctl);   % control usage weights

K_lqr = lqr(A, B, Q, R);
eig(A-B*K_lqr)          % checks poles

L_lqr = sys*K_lqr;
size(L_lqr)
sys_lqr = ss(A-B*K_lqr, B, C, D);
```

```
step(sys_lqr)             % checks step response
```

The purpose of LQR controller is to minimize the quadratic cost function of $\int(x^TQx + u^TRu)dt$. The parameters Q and R can be used as design parameters to penalize the state variables and the control effort. We choose large Q because we are more concern about the change of state and R small because the cost of energy like electricity usage is not important in this stage. We want to stabilize the system as soon as possible.

In this control problem, we set the motor to track square-wave angle for 30 degree in 0.2 hz frequency with initial pendulum angle at $\pi/6$. While motor is tracking the reference, the pendulum need to stand in upright position.

```
placed_poles = [-50 -51 -52 -53];       % placed poles
L_luen = place(A', C', placed_poles)';  % places poles
% Luenberger Observer model
obs_luen = ss(A-L_luen*C,[B L_luen], eye(4), []);
eig(A-L_luen*C)                         % checks poles
```

In this problem, we used Luenberger observer to estimate the states. We need to place the observer poles at least 5 time faster than the slowest pole of our controller. Therefore, we placed the poles on -50, -51, -52 and -53 respectively.

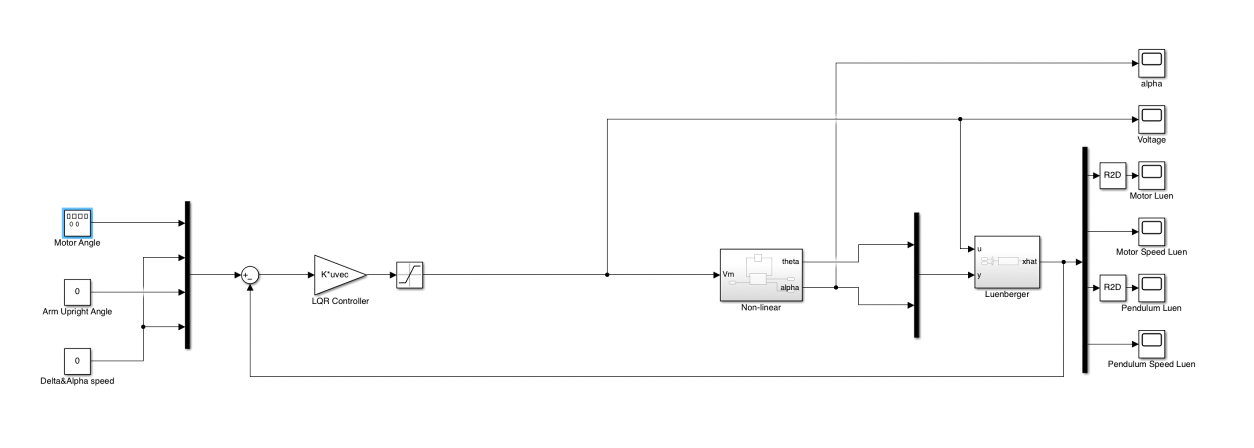


Figure 10: This is the layout of Simulink model of Luenberger Observer with LQR controller.

The Simulink model of the system is shown above. The LQR controller will minimize undesired deviations between estimate states and our desired states.

Result:

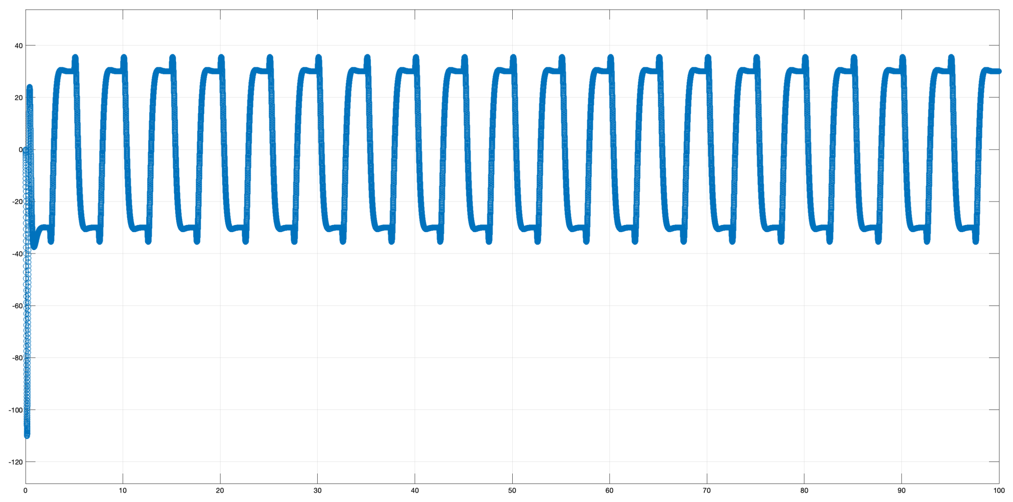


Figure 11: The simulation shows the result of motor angle tracking.

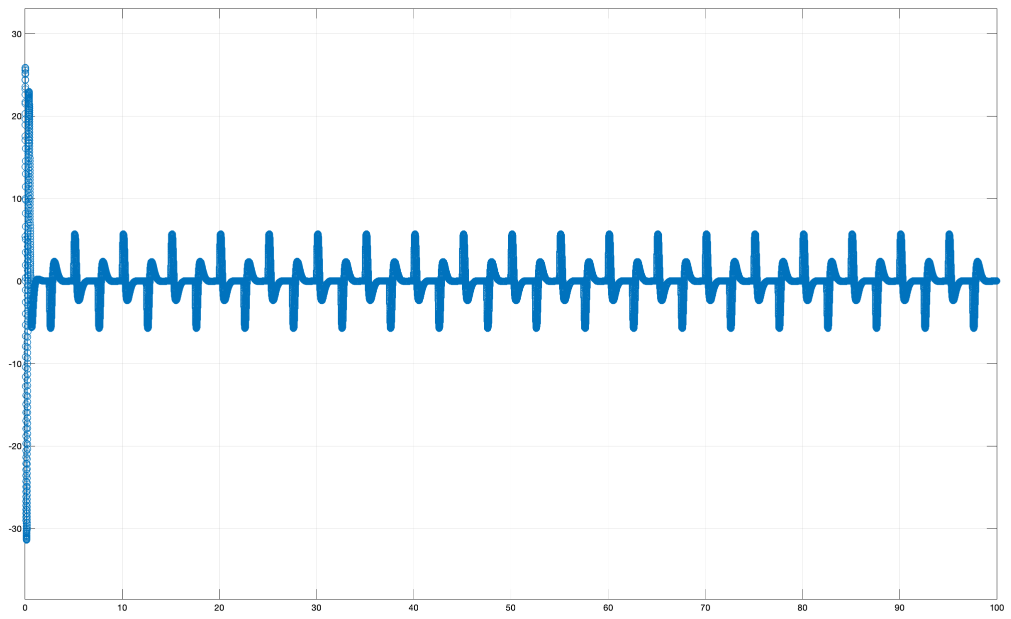


Figure 12: The simulation shows the result of pendulum angle tracking.

According to the result above, this model shows good tracking ability in both motor angle and pendulum angle in 0.2hz frequency. It is worth noting that at the beginning, to make the pendulum swing from $\pi/6$ back to upright position, the motor need to suddenly rotate to maximum -110 degree, which is opposite to tilted direction of the pendulum. After this, the whole system shows good stability and tracking ability.

3.2 LQR Controller Implementation

In this section, the results from the previous sections were combined into a Simulink model and tested on the hardware. Due to the hardware issue, only a video was recorded without response data. The additional Simulink task was performed to create a linear plant model for the upright position and a PD controller to test the upright position response.

3.2.1 Simulink LQR Hardware Implementation (partial)

The LQR controller and Luenberger observer with pole placement, created in the previous tasks, were combined and connected with the hardware. A ”trigger” block was also created to only activate the controller when the absolute pendulum arm angle value is within 30° to the vertical position. The model is shown in Figure 13.

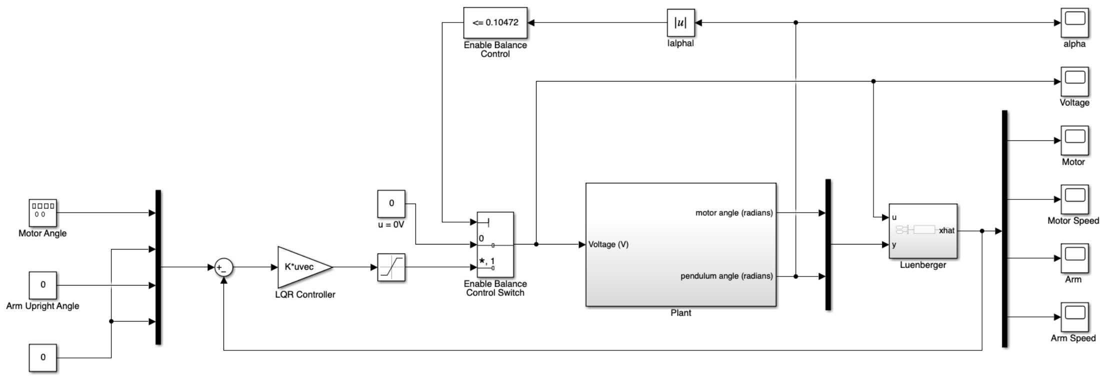


Figure 13: The LQR Hardware Simulink Model was Created with a LQR Controller, a Linearized plant, a ”Trigger” block, and a Luenberger Observer

This model was tested with the hardware and achieved reasonable motor angle tracking and inverted upright pendulum stability. A video was recorded: `https://www.youtube.com/shorts/4IWSIiAZmL0`. Due to hardware issues, the team was unable to record the response of motor angle tracking, pendulum arm upright angle, and trigger.

3.2.2 PD Controller Simulink Implementation

The linearized upright model still uses the same non-linear system dynamics that chooses equilibrium points at 0° which, in this scenario, is defined as the upright pendulum position. The reason to choose the upright position as 0° is to simplify the trigger condition for the hardware test. To accommodate this change, the pendulum angle $alpha$ in the state vector was changed to $alpha - \pi$. The resulting Simulink block is shown in Figure 13. The detailed MATLAB implementation is shown in Appendix: 5.1.2 Upright Position Modeling 5.2.

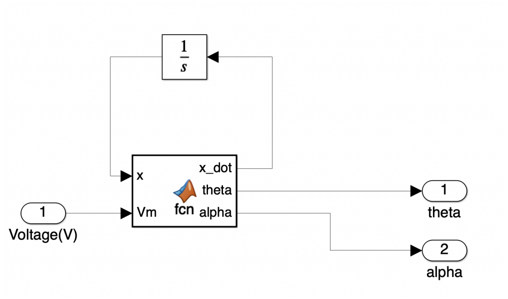


Figure 14: Linearized Model with Equilibrium Points at the Upright Position

Different from the previous tasks, this Simulink simulation utilized a PD controller instead of a LQR controller. The Simulink model is shown in Figure 15. The arm reference angle was set to 0° as the upright position, and the initial condition was set to $\pi/6$ (30°) which is shown in Figure 16. The PD values were calculated by the Simulink Controller Tuning which is shown in Figure 17. The results were achieved by reducing the rise time and increasing the robust value.

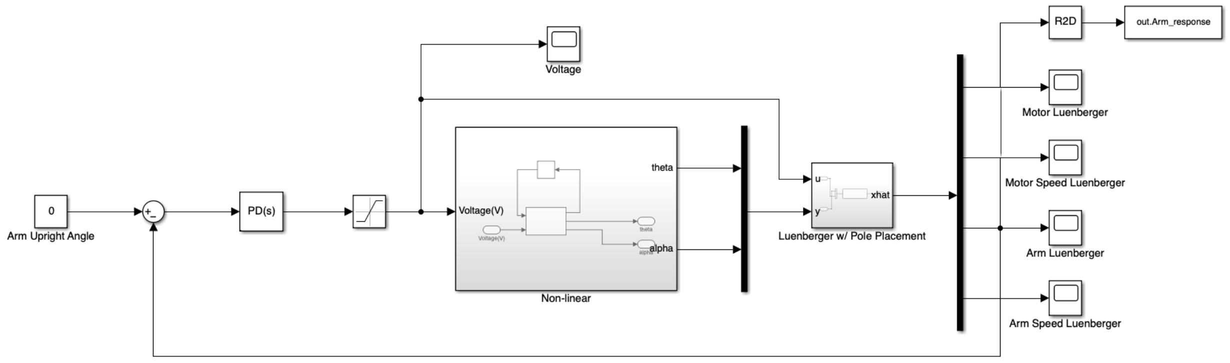
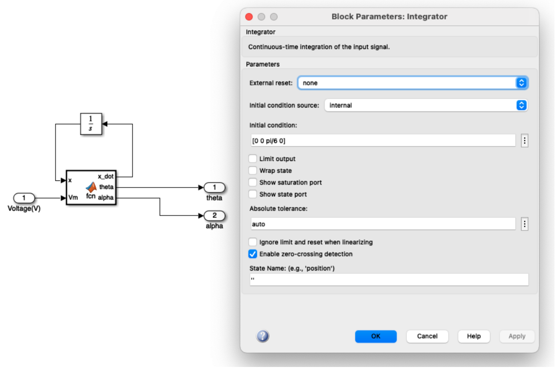


Figure 15: The PD Hardware Simulink Model was Created with a PD Controller, a Linearized Plant, and a Luenberger Observer



Block Parameters: Integrator

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

[0 0 pi/6 0]

- [ ] Limit output
- [ ] Wrap state
- [ ] Show saturation port
- [ ] Show state port

Absolute tolerance:

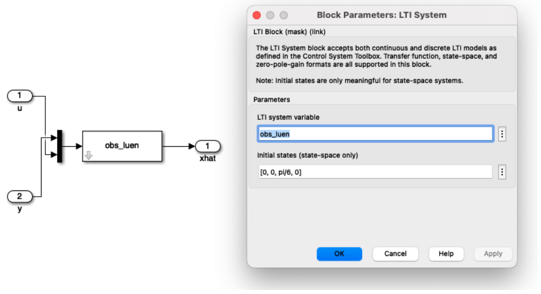
auto

- [ ] Ignore limit and reset when linearizing
- [x] Enable zero-crossing detection

State Name: (e.g., 'position')

''

((a)) Plant Initial Condition



Block Parameters: LTI System

LTI Block (mask) (link)

The LTI System block accepts both continuous and discrete LTI models as defined in the Control System Toolbox. Transfer function, state-space, and zero-pole-gain formats are all supported in this block.

Note: Initial states are only meaningful for state-space systems.

Parameters

LTI system variable

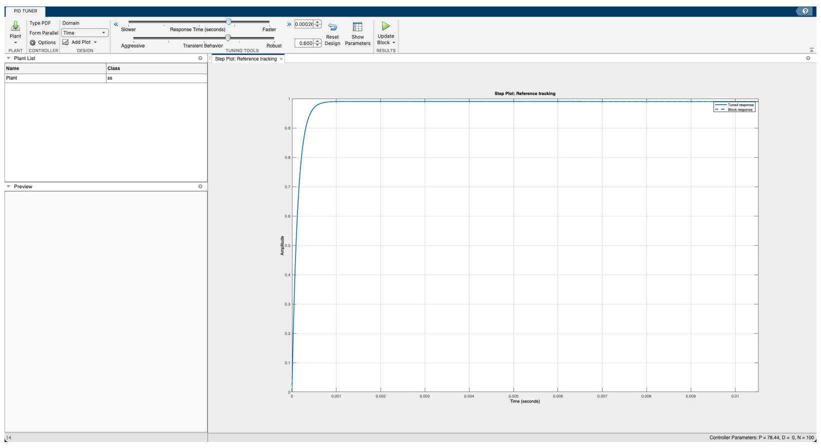
obs_luen

Initial states (state-space only)

[0, 0, pi/6, 0]

((b)) Observer Initial Condition

Figure 16: Initial Pendulum Arm 30°(π/6) Condition



Controller Parameters: P = 78.44, D = 0, N = 100

Figure 17: PD Simulink Control Tuning Shows the Tuning Result Matches the Expected Response

The pendulum arm response is shown below in Figure 18. The response is very fast and matches the expectation as the pendulum arm angle starts at 30°(π/6) and quickly goes down to 0° (the upright position) and stays at 0°.

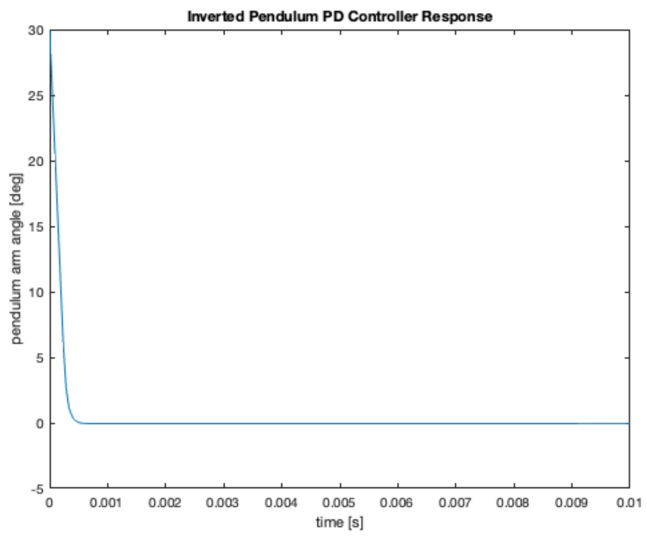


Figure 18: PD Inverted Pendulum Response Starts at 30 Degrees and Converges to 0 Degrees

To find the stable range of the initial pendulum angle, the initial condition was increased from 60° to 180°. However, the PD controller could always stabilize the system and make the pendulum inverted, as Figure 19 shows at 180°. The only differences are the time needs to achieve the 0° inverted position and the overshoot amount. The larger the initial angle, the longer it takes to converge and the more the overshoot.

```
arm_ini = deg2rad(mod(60, 360));     % initial arm condition
IC = [0; 0; arm_ini; 0];             % initial conditions
```

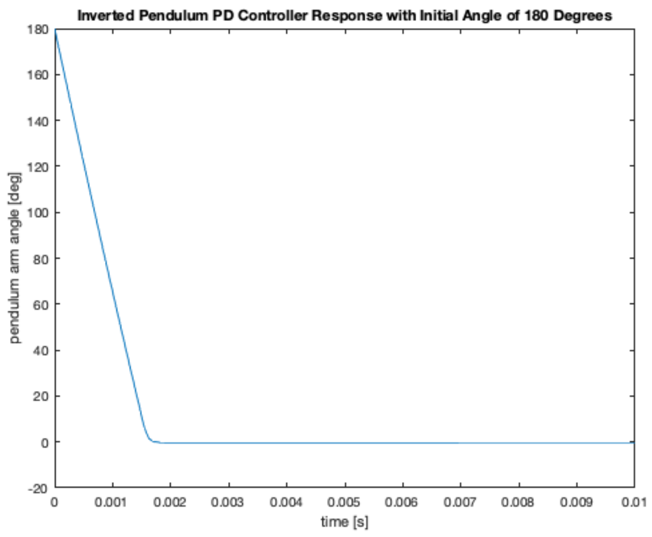


Figure 19: PD Inverted Pendulum Response Starts at 180 Degrees and Converges to 0 Degrees

3.3 LQR controller design for discrete-time system

To design a LQR controller based on a discrete-time system, we first discretized the original system with a sampling rate of $T_s = 1/f_s = 0.005\,[s]$.

```
sysCT = ss(Ac,Bc,Cc,Dc);
Ts = 1/200;
sysDT = c2d(sysCT,Ts,'zoh');

Ad = sysDT.A;    Bd = sysDT.B;    Cd = sysDT.C;
```

Then we simply set the weights and use `dlqr` to design a reasonable LQR controller.

```
Q = diag([1 0 10 0]);
R = 1e-2;
Kd_lqr = dlqr(Ad,Bd,Q,R);
```

Then we can formulate the Simulink scheme to implement the controller.

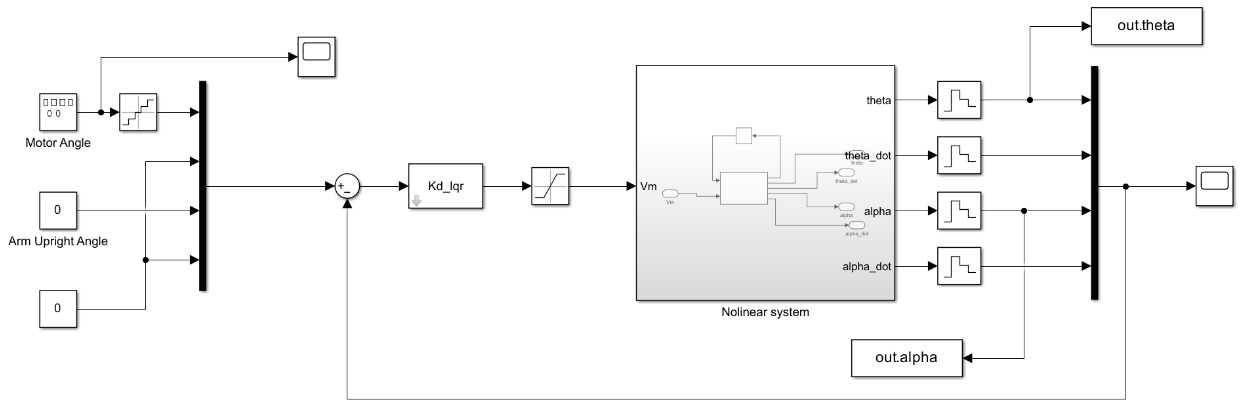


Figure 20: This is the layout of Simulink model for the discrete ssystem with the LQR controller.

Let's compare the feedback gain for both DT and CT:

$$K_{CT} = \begin{bmatrix} 3.1623 & 0.3731 & 0.4314 & 0.1403 \end{bmatrix} \tag{1}$$

$$K_{DT} = \begin{bmatrix} -2.9604 & -1.6289 & 38.0084 & 2.6843 \end{bmatrix} \tag{2}$$

Generally, the absolute value of each entry in K_{DT} is greater than that in K_{CT} except for the θ term, which matches the expectation as more efforts are required to bring the states towards the reference in a discrete time system. Similarly, when we focus on α, we noticed that the control effort increases greatly because we have to counteract the behavior caused by inertia.

The results for a sampling rate $200\,[Hz]$ is shown below. We found that it's really close to the result of a continuous system since it has a high sampling rate.

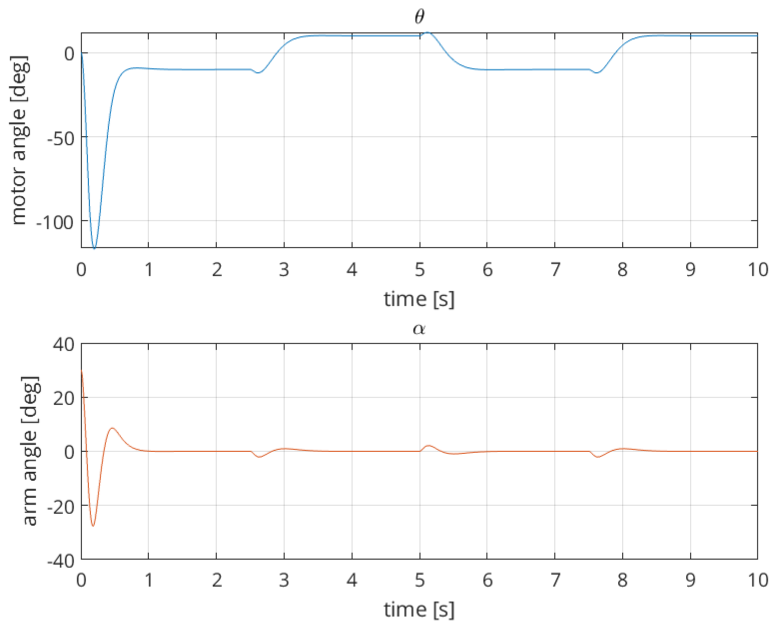


Figure 21: Angle Outputs with a Sampling Rate of 200 [Hz]

Now we move on to find the lowest sampling rate that guarantees the stability of the system. After some tuning, we found that a sampling rate of $0.05\,[s]$ i.e. $20\,[Hz]$ will be critical for the system to be stable.

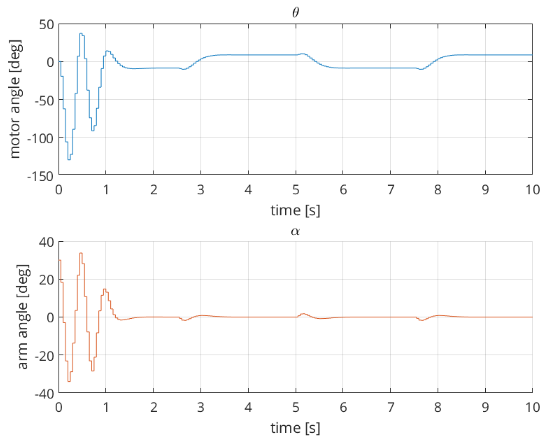


Figure 22: Sampling Rate $20\,[Hz]$

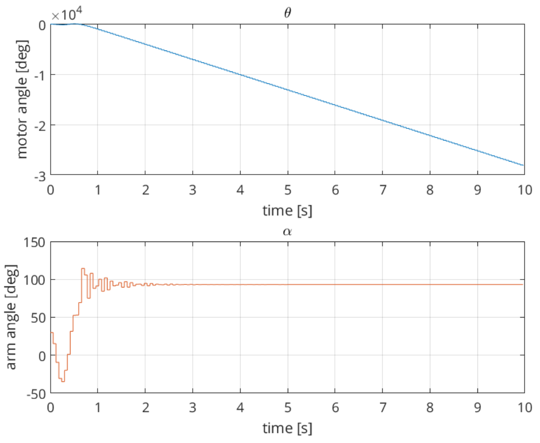


Figure 23: Sampling Rate $16.67\,[Hz]$

Figure 24: Angles Response with Different Sampling Rates

4 Conclusions

What were the main results?

The team successively achieved a reasonably good inverted pendulum response on the hardware with a LQR controller and two types of observers. Though the hardware issues prevent us to do further testing and tuning, the Simulink link model has already proven to achieve the expected results. The initial observer testing shows that both Luenberger Observer and Kalman Filter could achieve great accuracy in the unmeasured states. However, when the initial pendulum angle increases, the Kalman Filter performance might decrease. Interestingly, for the PD controller with Luengerber Observer, no matter the initial angle, the pendulum arm would stay inverted. For the quantization effect, the higher the encoder counts, the more analog or continuous the measurement would be. Finally, the team also explored how discretization would affect the model stability and performance, and also the lowest sampling rate. The results show discretization might increase the control efforts

What did you learn (if anything) by completing the lab?

It is usually typical that we do not have the full knowledge of state measurement. Therefore, we need to rely on the state observer to estimate the state information. During this lab, we experiment with Luenberger Observer, Kalman Filter, and derivative with LPF. Both Luenberger observer and Kalman filter can not only provide estimation towards unmeasured state, but also filter noise from measured signal. Thanks to the separation principle, we can design the observer separately. The only thing we need to make sure is that the observer should be faster than the controller. However, since we are relying on the linearized model to perform prediction, it will be sensitive to dynamics in nonlinear region, but it will guarantee to converge if the system is stable.

What suggestions do you have to make the lab better or more interesting?

Basically, it would be better if we could implement our controller on Arduino. We spent extra time on debugging the code and finally realized that it was the matter of the deficiency of the hardware. In addition, we noticed that the pendulum was quite easily to detach from the arm. Apart from that, we managed to design Leunburger Observer, Kalman Filter as well as LPF to achieve various interesting results in this lab, and it would be more fun if we could add some external disturbances, for example, a poke on the pendulum, so that we could check the response of the system to these interference and perhaps design a more rigorous and general system.

5 Appendix

5.1 State Estimator Design

```
% Downward Init condition
IC = [0; 0; pi/18; 0];

% Noise info
Tn = 2e-3;
cov_u = 1e-9;
cov_theta = 1e-9;
cov_alpha = 1e-9;

% LPF
w = 50;

% Downward CT Luenberger Observer
P_LB = [-20 -21 -22 -23];
L_LB = place(A_down', C', P_LB)';
Obs_LB = ss(A_down-L_LB*C,[B_down L_LB], eye(4), []);
eig(A_down-L_LB*C)

% Downward CT Kalman Filter
Q = cov_u/Tn;
R = [cov_theta/Tn 0; 0 cov_alpha/Tn];
[¬,L_KF,¬] = kalman(sys_down, Q, R);
Obs_KF = ss(A_down-L_KF*C,[B_down L_KF], eye(4), []);
eig(A_down-L_KF*C)

% Upright CT Luenberger Observer
P_LB = [-50 -51 -52 -53];
L_LB = place(A_up', C', P_LB)';
Obs_LB = ss(A_up-L_LB*C,[B_up L_LB], eye(4), []);
eig(A_up-L_LB*C)

% Upright CT Kalman Filter
Q = cov_u/Tn;
R = [cov_theta/Tn 0; 0 cov_alpha/Tn];
[¬,L_KF,¬] = kalman(sys_up, Q, R);
Obs_KF = ss(A_up-L_KF*C,[B_up L_KF], eye(4), []);
eig(A_up-L_KF*C)
```

5.2 Upright Position Modeling

```
% parameters
Ts = 0.01;
syms theta theta_dot alpha alpha_dot real
syms theta_ddot alpha_ddot real
syms Vm real
R_m = 8.4;
K_m = 0.042;
M_r = 0.095;
```

```
L_r = 0.085;
J_r = 5.72e-5;
D_r = 0.0015;
M_p = 0.024;
L_p = 0.129;
J_p = 3.33e-5;
D_p = 0.0005;
g = 9.81;

% non-linear system dynamics
func(1) = (M_p*L_r^2+1/4*M_p*L_p^2*(cos(alpha))^2+J_r)*theta_ddot
          + 1/2*M_p*L_p*L_r*cos(alpha)*alpha_ddot
          + 1/2*M_p*L_p^2*sin(alpha)*cos(alpha)*theta_dot*alpha_dot
          -1/2*M_p*L_p*L_r*sin(alpha)*alpha_dot^2
          - K_m*(Vm - K_m*theta_dot)/R_m + D_r*theta_dot;
func(2) = 1/2*M_p*L_p*L_r*cos(alpha)*theta_ddot
          +(J_p+1/4*M_p*L_p^2)*alpha_ddot
          -1/4*M_p*L_p^2*cos(alpha)*sin(alpha)*theta_dot^2
          +1/2*M_p*L_p*g*sin(alpha) + D_p*alpha_dot;

% changes pendulum arm angle(alpha) to make upright position as 0 degree
func = subs(func, [alpha, alpha_dot], [alpha-pi, alpha_dot]);
[theta_ddot, alpha_ddot] = solve(func == [0;0], [theta_ddot, alpha_ddot]);

% creates functions to solve for acceleration variables
theta_ddot = simplify(theta_ddot);
matlabFunction(theta_ddot, 'File', 'get_theta_ddot', 'vars',
               {theta theta_dot alpha alpha_dot, Vm});

alpha_ddot = simplify(alpha_ddot);
matlabFunction(alpha_ddot, 'File', 'get_alpha_ddot', 'vars',
               {theta theta_dot alpha alpha_dot, Vm});

% state vectors
x = [theta; theta_dot; alpha; alpha_dot];
x_dot = [theta_dot; theta_ddot; alpha_dot; alpha_ddot];

% 4 states, 1 control
num_state = 4;
num_ctl = 1;

% use Jacobian to linearize around equilibrium points
A = double(subs(jacobian(x_dot, x), [theta; theta_dot; alpha; alpha_dot], [0;0;0;0]));
B = double(subs(jacobian(x_dot, Vm), [theta; theta_dot; alpha; alpha_dot], [0;0;0;0]));
C = [1,0,0,0; 0,0,1,0]; % [0, 0, 1, 0] to include the pendulum arm angle output
D = [0;0];
sys = ss(A,B,C,D);
```

5.3 LQR Controller Design for Inverse Pendulum Control

```
Q = diag([10 0, 1, 0]); % states weights
R = 1e0*eye(num_ctl);   % control usage weights

K_lqr = lqr(A, B, Q, R);
```

```
eig(A-B*K_lqr)          % checks poles

L_lqr = sys*K_lqr;
size(L_lqr)
sys_lqr = ss(A-B*K_lqr, B, C, D);
step(sys_lqr)           % checks step response
```