# ME 134 Lab 6b Report

Orion V, Tatsuyoshi Kurumiya, Eric Bermudez
Section 201

May 03 2024

## 1 Purpose

Our goal for this lab was to design a full-state observer to estimate the state of an inverted pendulum using only information about cart and pendulum position. Additionally, the lab will give us an experience of implementing this estimate for full state feedback control of a system and practice in how to understand a system's noise. Furthermore, this lab helps to bring about practical implementation of the theoretical topics covered in class, namely full state feed back, pole placement, and observer design.

## 2 Pre-Lab

### 2.1 Controllability and Observability

We start this lab by checking if the state-space representation of the linearized system is controllable and observable. To do so, we were instructed to use built in MATLAB functions ctrb(), osbv(), and ran(). To determine observability we must create the observability matrix from the state space representation of the system. An nth-order plant is completely observable if the observability matrix is of rank n. For a state space. Similarly, the system is said to be completely controllable if the controllability matrix formed from state space matrices is also of rank n.

For system with an nth order plant whose state and output equations are, respectively,

$$\dot{x} = Ax + Bu$$
$$y = Cx$$

The observability matrix is:

$$O_M = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

The controllability matrix is:

$$C_M = [B \ \ AB \ \ A^2 \ \ \cdots \ \ A^{n-1}B]$$

The code to determine controllability and observability is shown in Listing 1.

Listing 1: MATLAB code used to determine the controllability and observability of the system

```
%system dynamics denominator
sigma = M + m + Jm*Kg^2/r^2 - 3*m/4;

% elements of state space matrices
a12 = 1 ; a34 = 1 ;
a22 = -Kg^2*Kt*Km / (Rm*r^2*sigma);
a23 = -3*g*m /(4*sigma);
```

```
a42 = 3*Kt*Kg^2*Km / (4*Lp*Rm*r^2*sigma);
a43 = -9*g*m / (16*Lp*sigma) + 3*g / (4*Lp);
b2 = Kt*Kg / (Rm*r*sigma);
b4 = - 3*Kt*Kg/ (4*Lp*Rm*r*sigma);

% Form state space representation of the plant
A = [0 a12 0 0 ;
     0 a22 a23 0;
     0 0 0 a34;
     0 a42 a43 0];

B = [0 b2 0 b4]';
C = [1 0 0 0 ; 0 0 1 0];
D = [0 0]';

% determine controllability and observabilty
if rank(ctrb(A,B)) == 4 & rank(obsv(A,C)) == 4
    disp('System is controllable and observable.')
else
    disp('System is not controllable or not observable.')
end
```

The code from Listing 1 outputs the following:

```
System is controllable and observable.
```

## 2.2 Observer Design

In the previous lab, we used the same system we assumed to have access to the full state. This assumption was made by estimating velocity states of $\dot{x}$ and $\dot{\theta}$. This result yielded a poor quality estimate of the velocity states due to the amplification of noise which lead to poor controller output. To get better performance in this lab, we will implement a Luenberger observer as an alternative to using derivative blocks to estimate velocity states.

The dynamics of a Luenberger observer are as follows:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - \hat{y})$$

where

$$y = Cx$$

$$\hat{y} = C\hat{x}$$

$$u = K(r - \hat{x})$$

The Luenberger observer is made of two parts, namely the predictor and corrective parts. The predictive part is the first two terms of the equation, $A\hat{x} + Bu$, and the corrective part is the final term, $L(y - \hat{y})$. For the controller gain K we desire closed loop poles $s_{1,2} = -2.0 \pm 10j$ and $s_{3,4} = -1.6 \pm 1.3j$, the same as the previous lab. The observer gain L is chosen such that the eigenvalues of the matrix A-LC are in the left hand plane and whose exact positions govern the rate at which the state estimate $\hat{x}$ converges to the actual state x of the system. Since the matrix A-LC is a 4x4 matrix and the output matrix C is a 2x4 matrix, then the *dimensions of the observer gain matrix L should be 4x2.* Following the general rule of thumb, we chose the eigenvalues for the matrix A-LC such that the error dynamics are an order of magnitude faster than the dynamics of the controlled system. The eigenvalues of the observer are to be placed at $-10 \pm 15j$ and $-12 \pm 17j$. The MATLAB code to obtain the matrix L is shown in Listing 2

Listing 2: MATLAB code to determine observer gain matrix and observer state space model using pole placement.

```
s_obs = [-10+15*i , -10-15*i , -12+7*i , -12-7*i] ; % desired observer eigenvalues
L = place(A',C',s_obs)' ; % pole placement
```

### 2.3 Simulation

Using the newly designed observer we create a Simulink model with the observer placed in feedback around the actual system. We use the estimated states $\hat{x}$ for state feedback and the feedback gain matrix K designed in the previous lab. The Simulink model is shown in Figure 1.

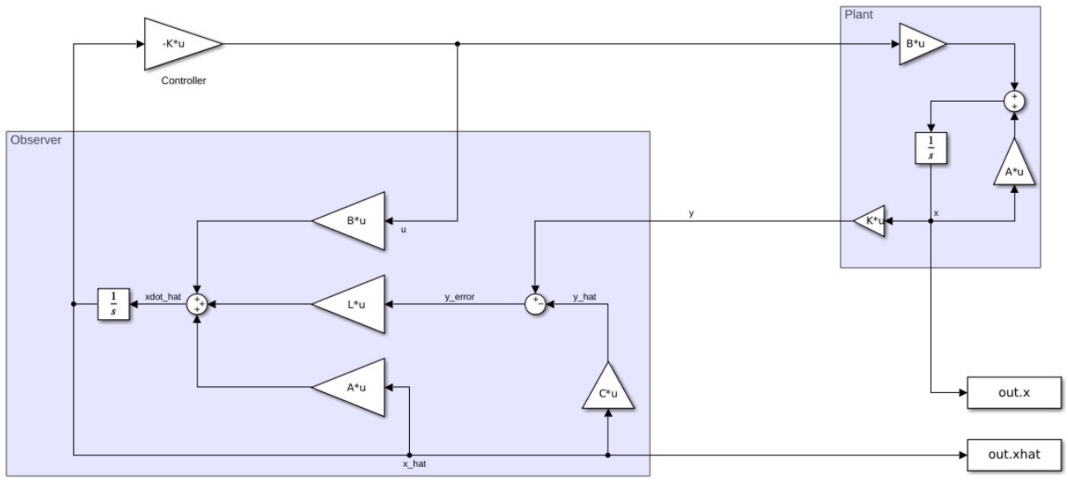


Figure 1: Simulink model of system with observer in feedback

Using the Simulink model shown in Figure 1, we simulate the system's response for position and angular perturbations of 10 cm and 5 degrees, respectively. This was achieved by setting the plant's integrator block with desired initial conditions. However, it is important to note that the systems inputs are in meters and radians, rather than centimeters and degrees. Thus, appropriate conversions must be made. The parameter settings for the integrator block to achieve the initial conditions is shown in the Appendix as Figure 13.

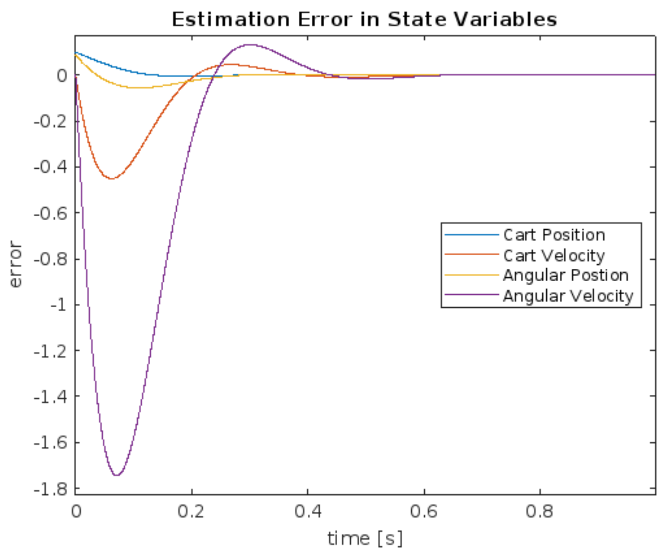


Figure 2: Parameter settings to implement desired initial conditions with appropriate conversions

By using ”to workspace” blocks in the Simulink model, we are able to acquire signal information for the state estimate and state signals. We use this information to create plots and compare the estimated states to the actual states. The plots of the estimated and actual states are shown in Figure 3 and the code used to produce these plots is in the Appendix under Listing 4. It is important to note that in practice this cannot be done with the physical plant as there is no measurement of the actual state.

Next we want to know how the estimation error $e = \hat{x} - x$ varies with time. The plot for the estimation error is shown in Figure 2 and the code to plot this is posted in the appendix under Listing 4 The most apparent and notable thing we noticed is the estimation error for every state tends to zero within approximately half a second. Additionally, The greatest amount of error occurs in the velocity states, and most notably in the angular velocity state. This is interesting because the error in the velocity states are initially at zero error and then jump to large overshoots, but resolve relatively quickly.

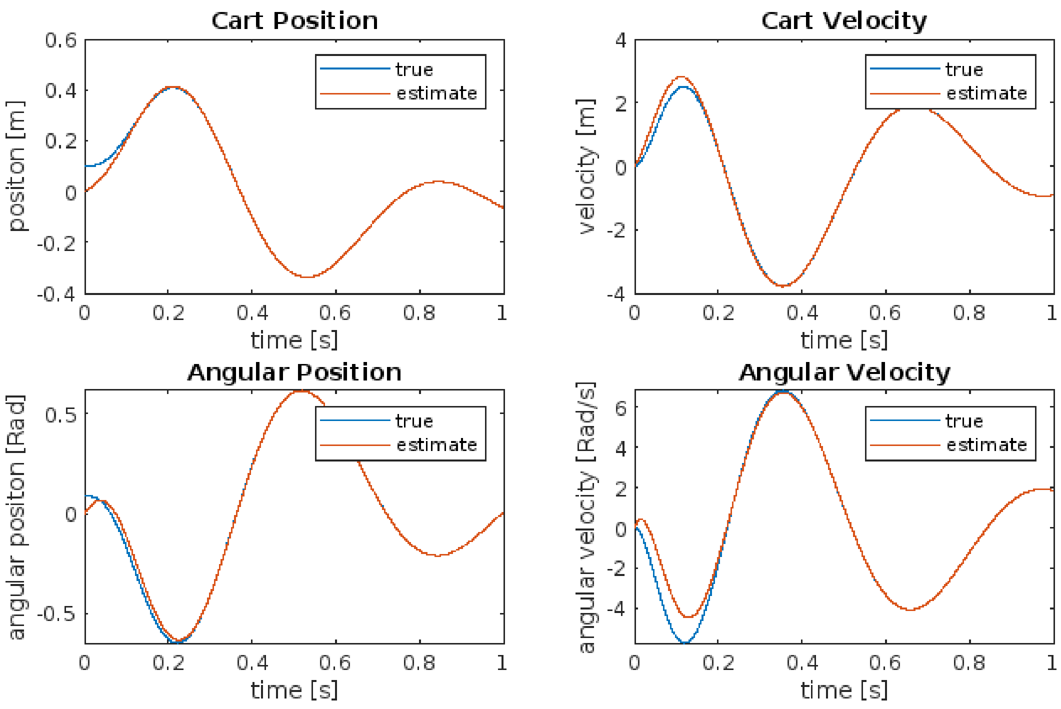


Figure 3: Individual plots comparing actual and estimated states for system response of perturbations of 10 cm and 5 degrees

## 3 Lab

In the actual lab we implement the state feedback controller on the state estimate $\hat{x}$ using the Luenberger observer on the hardware. The Simulink block diagram used for this is shown in Figure 4. First we observed and recorded the output $\hat{y}$ of the observer and the actual measurement y while applying small perturbations. Plots for cart position and pendulum angular position to compare the actual signal and estimated signal are shown in Figure 5.

Next we want to compare the controller from Lab 6a to the newly developed controller in this lab. It is important to note that the closed loop poles of each system are the same, but with different controller schemes. To compare the controller performance, we used the following reference signals: zero reference, zero reference with small manual perturbations, and a sinusoidal position reference with amplitude 5 cm and frequency 1 rad/s. It should be noted that we made a conscious effort to create a repeatable and consistent application of the manual perturbations. The plots associated with each reference are in Figures 6 - 8

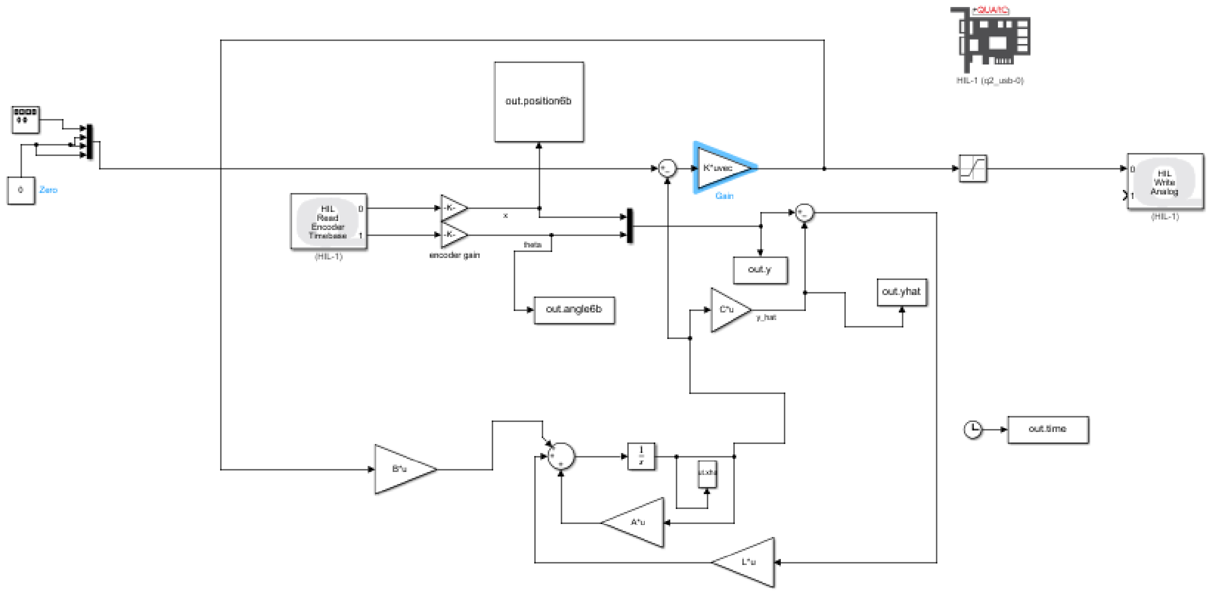


Figure 4: Simulink block diagram for hardware with implemented observer

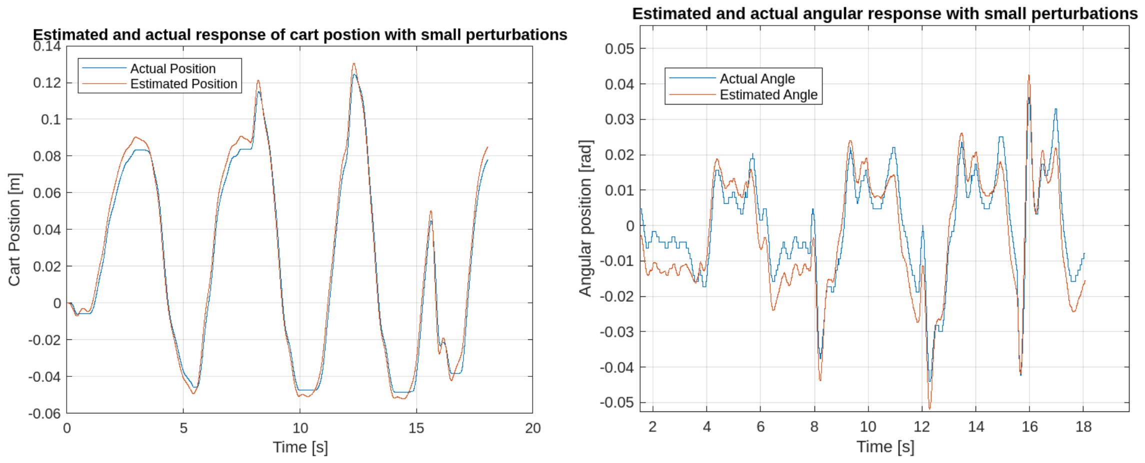


Figure 5: Plots showing estimated and actual signals on same graph for the position of the cart and the angle of the pendulum under small perturbations with a zero reference signal.

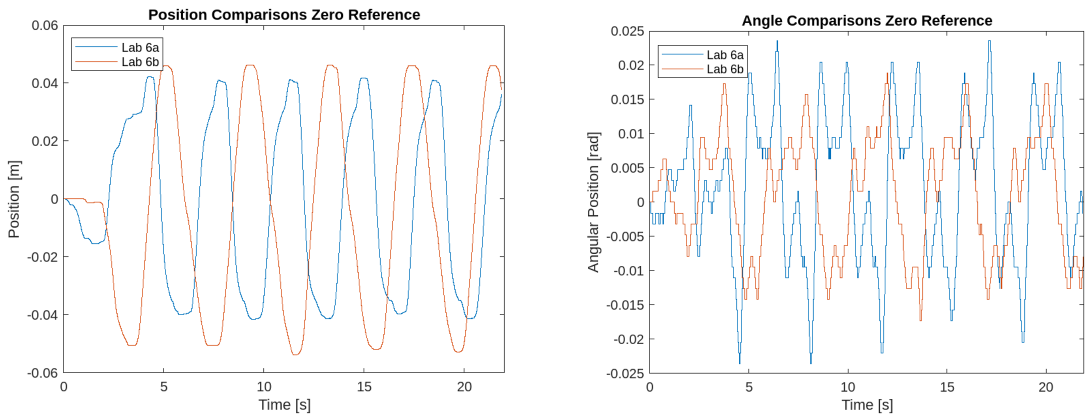


Figure 6: Plots comparing tracking ability of controllers from lab 6a and lab 6b with a zero reference.

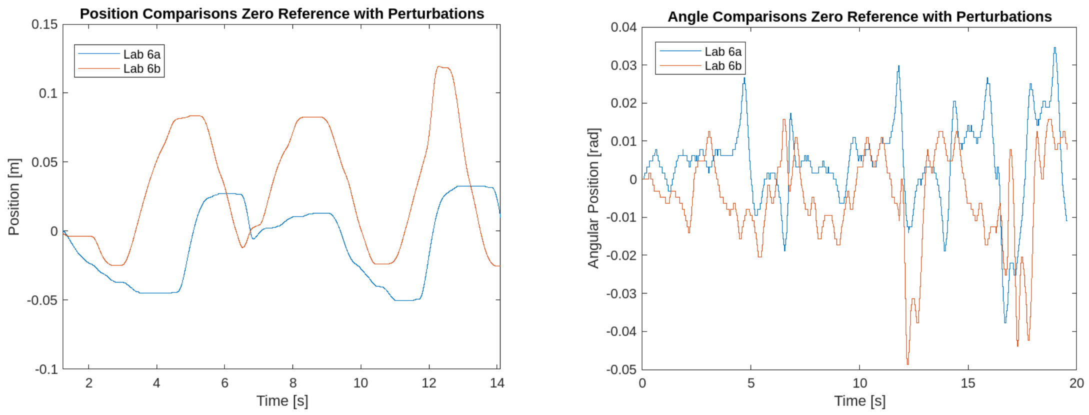


Figure 7: Plots comparing tracking ability of controllers from lab 6a and lab 6b with a zero reference and controlled manual perturbations.

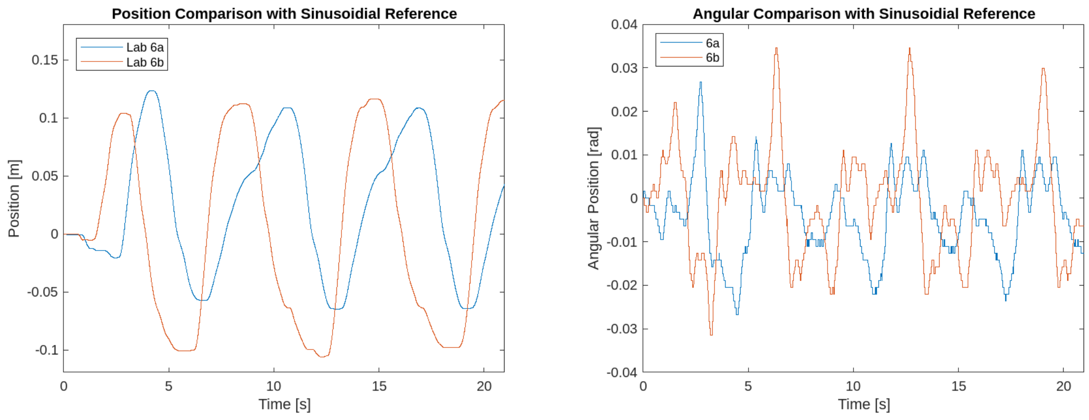
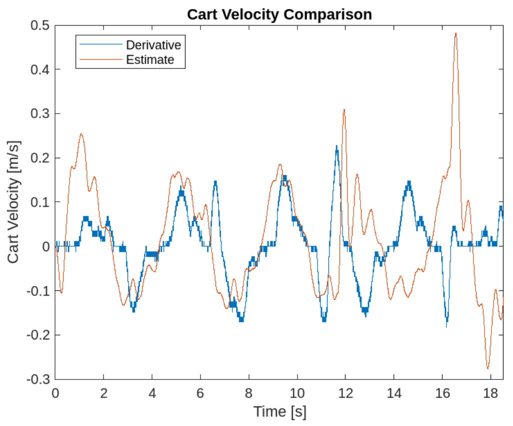


Figure 8: Plots comparing tracking ability of controllers from lab 6a and lab 6b with a sinusoidal reference of $r(t) = .05sin(t)$.

Comparing the *derivative based controller and the observer based controller by inspection*, there are few things to note. In the position based comparisons, the signals seem to match up well, though it looks as if there is a potential phase offset. The most dramatic difference in the position based comparison is in the controlled manual perturbation trial. The cart position state in the derivative based controller from lab 6a (shown in Figure 7) seems to round off the peaks of the overshoots and starts by drifting away from the zero position with out an initial perturbation. Looking at all the position plots in Figures 6 - 8 it shows that the derivative based controller seems to drift towards the negative position. Looking at the angular position for the zero reference, the controllers look like they follow the same general trajectory but with a slight phase shift. This however does not really hold for the controlled perturbation and sinusoidal references for angular position. When looking at Figure 8 for the angular comparison, there seems to be a significantly persistent larger overshoot in the signal from the observer based controller.

Next we take a *closer look at the differences in the controller performances by comparing the cart and pendulum velocities* from the two labs. We compare the estimated velocities from the observer based controller in this lab with the velocities obtained via differentiation of the position measurements from the last lab. The comparisons of which are plotted in Figure 9. The most striking contrast is in the comparison of angular velocities. The derivative based controller seems to show more of an on/off signal instead of the expected wave like curve. Additionally, the cart velocity signal from the derivative based controller seems to be fuzzy as compared to the observer based controller which has a smoother signal. We suspect the fuzziness and sporadic behavior in the derivative based controller comes from its sensitivity to noise.



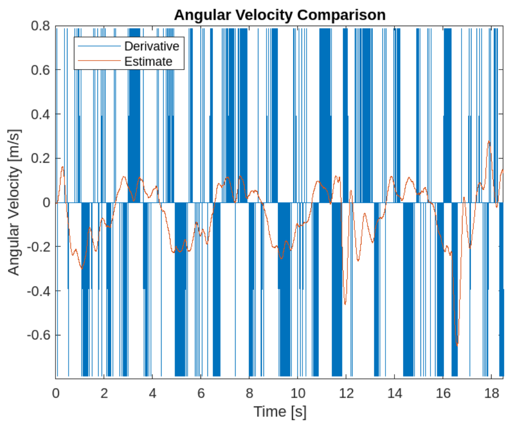


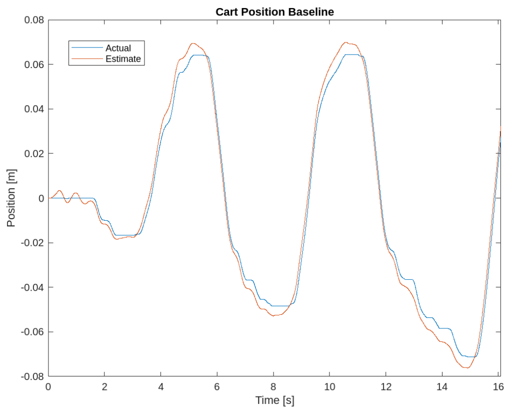
Figure 9: Plots of position and angular velocity comparison of observation and derivative based controllers.

After inspecting and comparing the signals produced from the controllers from each lab we conclude that the *observer based controller gives better performance.* We are inclined to believe this because of the observer based controller's robust resilience to noise. Though the derivative based controller seems to react better to perturbations, it may be more likely to fail due to noise amplification.

## 3.1 Measurement Noise

This section of the lab guided us in ways to understand a systems noise. This is a concept that is directly applicable to practical use cases because designing system resiliency to noise can only be done once the noise of the system is understood. We start by directing our attention to the cart position and evaluate it based on two performance metrics: Position deviation and noise power. Position deviation will be evaluated by using the MATLAB command std() on the system output y for various scenarios. Noise power is a measure of high frequency noise in the system output and was evaluated with the provided code in the lab document and is shown in Listing 5. These metrics were used to evaluate the observer by creating artificial noise into the sensor signals.

We started by letting system run at least 15 seconds using the observer and controller gains from before. This is done under a zero reference condition to determine a baseline for the position deviation and noise power metrics. From this run we want to compare the actual position and the observer's estimated position for the cart and pendulum angle. These results for the cart and pendulum positions and are shown in Figure 10. The result of the plot seem to indicate that the observer is ability to track the actual system state is satisfactory. Using the data from baseline we get calculated position deviation and noise power. The *baseline position deviations calculated were 0.0414* $m^2$ *and 0.0922* $rad^2$. for the cart position and pendulum angular position, respectively. The baseline noise power calculated were 4.3756e-06 and 9.302e-05 for the cart and pendulum position, respectively.



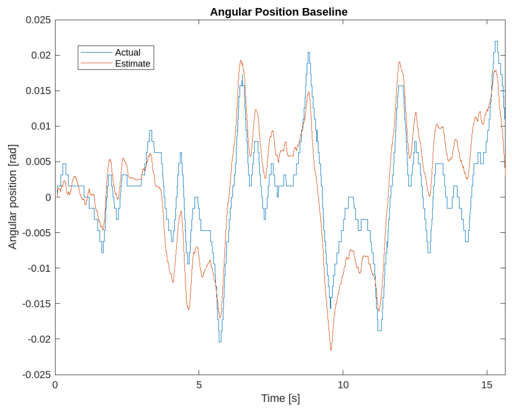
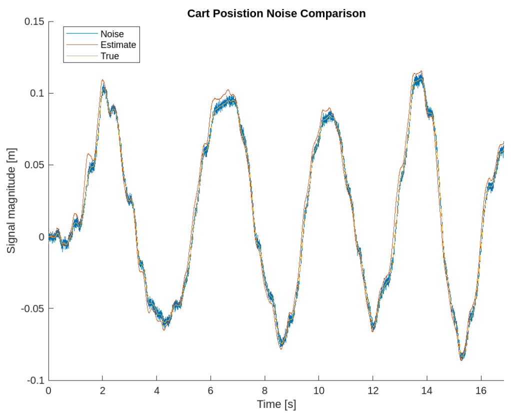


Figure 10: Plots for baseline position deviation and noise power calculations showing true and estimated values plotted on same graph for each respective state.

Next we introduce white noise by implementing Random Number Blocks into the block diagram and summing junction to artificially add noise to the position and angle readouts. We were instructed to set each of the Random Number Block's parameters as follows: zero mean, 0.002 sample time, unique seed per block, 9e-6 $m^2$ for cart variance, and 0.00190386 $rad^2$ for the pendulum angle variance. The plots showing the observer's estimate for cart and pendulum position along side the noisy and non-noisy signals are shown in Figure 11. Using the observer's data from the noisy signal plot, we calculated the *position deviations for the cart and pendulum positions signals to be 0.0619* $m^2$ *and 0.015* $rad^2$ *respectively.* Using the same data from the observer estimated states, we calculated the *noise power for the cart and pendulum position signals to be 2.9027e-6 and 9.0374e-6,* respectively. We observed that by introducing noise, the system had more oscillations during the 15 second interval.



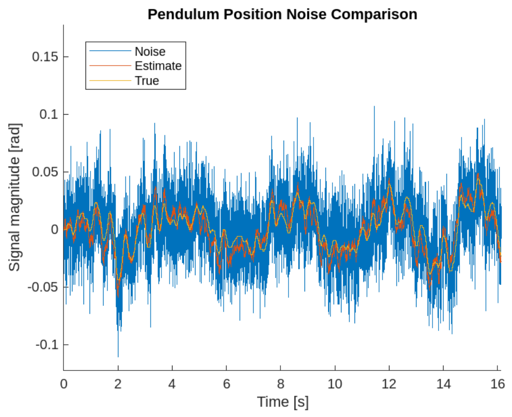
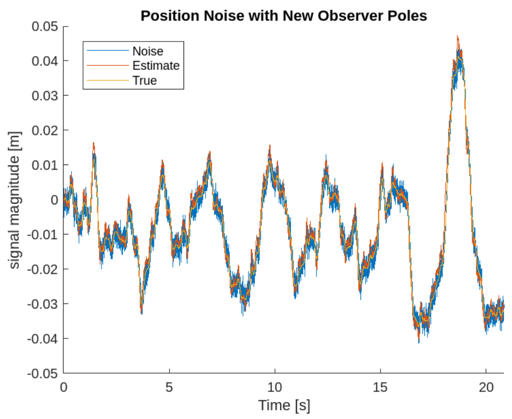


Figure 11: Plots to evaluate the observer's ability to track the actual position signal with induced artificial noise.

Lastly, we created an observer with new desired poles located at $s = -12 \pm 15i, -15 \pm 7i$. This change was done to see how an observer with a smaller damping ratio and increased rise time would react to the introduction of noise. The plots showing the new observer's estimate for cart and pendulum position along side the noisy and non-noisy signals are shown in Figure 12. Using the data from this run we *determined values for the cart and pendulum position deviations to be 0.0385 $m^2$ and 0.087214 $rad^2$ respectively.* For the *noise power, we calculated values of 3.6793e-5 and 2.5908e-5 for the position of the cart and pendulum, respectively.* We observed more oscillation cycles in the new observer poles using our new observer poles. We also noted that his change of observer poles



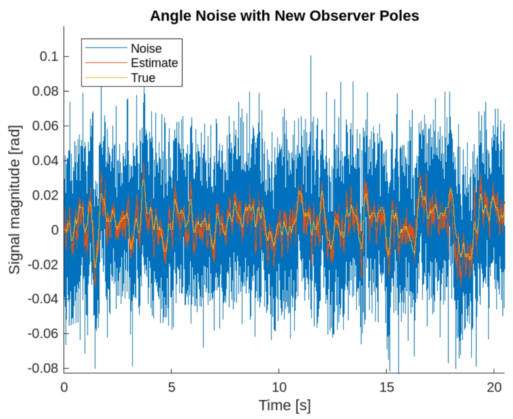


Figure 12: Plots to evaluate the observer's ability to track the actual position signal with induced artificial noise using new observer poles: $s = -12 \pm 15i, -15 \pm 7i$.

Our results are displayed in Table1 and was used to compare the metrics for each of the given conditions. Starting with the position deviation, we noticed that there was an increase in the cart's position deviation, but a significant decrease in the pendulum position deviation. Further, both the position deviation for the cart and the pendulum decreased bellow the baseline using the new observer poles. Now looking at the noise power for both the cart and the pendulum, we notice that noise power increases with the introduction of noise, but decreases using the new observer poles.

Table 1: Table to compare metric values

| Metric | Baseline | Noise | New Observer |
|---|---|---|---|
| Position Deviation: Cart $m^2$ | 0.0414 | 0.0619 | 0.0385 |
| Position Deviation: Pendulum $rad^2$ | 0.0922 | 0.0150 | 0.087214 |
| Noise Power: Cart | 4.3756e-06 | 2.9027e-6 | 3.6793e-5 |
| Noise Power: pendulum | 9.302e-05 | 9.0374e-6 | 2.5908e-5 |

## 4 Conclusion

In this lab we worked through the process of designing a Luenberger observer for the inverted pendulum system used in previous labs. We simulated the observer in feedback with the system before implementing it into the actual system. Using the simulation we analyzed the error in the estimated states from the observer. We found that the error in the estimated states had rather large overshoots initially, but went to zero for every state within 0.5 seconds. We then implemented the Luenberger observer on the hardware to determine how well the observer estimated the state of the system. After satisfactory performance from the hardware implementation we compared the controllers from lab 6a and 6b, understanding that the closed loop poles of the systems were the same. To compare the controllers we evaluated their ability to track signals under different references. We then moved to evaluating the observer's performance while introducing random noise. The performance was rated through two metrics: position deviation and noise power. We used these metrics on position data received from approximately 15 second runs for the system with out noise, with noise, and with noise using new observer poles. The results of which are given in Table 1. We determined that *the observer controller gives better performance over the scheme of estimating states using derivatives* due to its resistance to noise and that the noise is not amplified. Overall, in this lab we applied some theoretical concepts covered in lecture, namely full state feed back, pole placement, and observer design to the inverted pendulum system.

## 5 Appendix

Listing 3: MATLAB code used to define the parameters of the system for use in other code snippets.

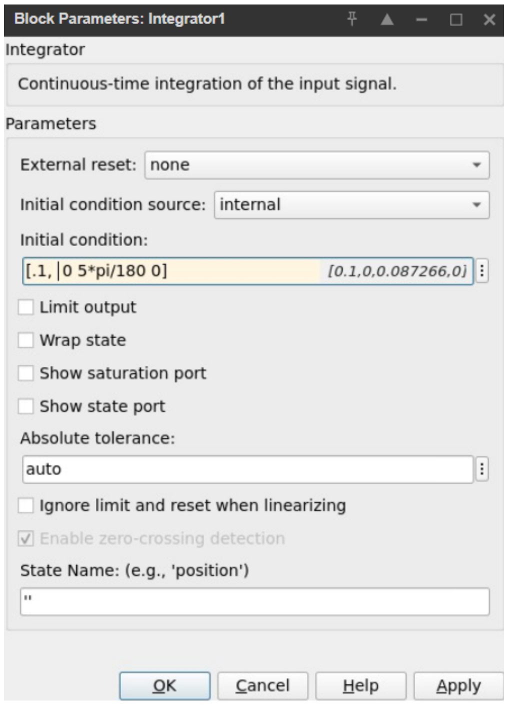
```
encoder_cart = 439.6 ;% [counts/cm]
encoder_angl = 651.9 ;% [counts/rad]
M             = 0.94  ;% [kg] mass of cart and motor
m             = 0.230 ;% [kg] mass of rod
Lp            = 0.3302;% [m] pend half length
Ic          = m*Lp^2/3;% moment of inertia from rod center
Ie        = 4*m*Lp^2/3;% moment of inertia from rod end
Kt            =7.67e-3;% [Nm/A] motor torque constant
Km            =7.67e-3;% [Vs/rad] motor back EMF constant
Kg            = 3.71  ;% Motor gearbox ratio
Rm            = 2.6   ;% [ohm] motor winding resistance
r             =6.36e-3;% [m] motor gear radius
Jm            =3.9e-7 ;% [kg*m^2] Motor moment of inertia
g             = 9.81  ;% [m/s^2] gravitational acceleration
```

Listing 4: MATLAB code to obtain information from simulation and create individual plots for each state and it's estimate.

```
% obtain values from the simulation
t = output.tout;
pos = output.x(:,1);
vel = output.x(:,2);
angpos = output.x(:,3);
angvel = output.x(:,4);
pos_hat = output.xhat(:,1);
vel_hat = output.xhat(:,2);
angpos_hat = output.xhat(:,3);
angvel_hat = output.xhat(:,4);

% plot each state and it's estimate in a separate plots
subplot(2,2,1)
plot(t,pos,t,pos_hat)
title('Cart Position')
ylabel('positon [m]') ; xlabel('time [s]')
legend('true','estimate')

subplot(2,2,2)
plot(t,vel,t,vel_hat)
title('Cart Velocity')
ylabel('velocity [m]') ; xlabel('time [s]')
legend('true','estimate')

subplot(2,2,3)
plot(t,angpos,t,angpos_hat)
title('Angular Position')
ylabel('angular positon [Rad]') ; xlabel('time [s]')
legend('true','estimate')

subplot(2,2,4)
plot(t,angvel,t,angvel_hat)
title('Angular Velocity')
ylabel('angular velocity [Rad/s]') ; xlabel('time [s]')
legend('true','estimate')
```

Listing 5: MATLAB code provided to compute the noise power.

```
function [ret] = relative noise power(cart pos)
fs = 1/2e -3;
n = length(cart pos);
total_band_power = bandpower(cart pos, fs, [df, fs/2-df]);
highfreq_band_power = bandpower(cart pos, fs, [10, fs/2-df]);
ret = highfreqband_power/total_band_power;
```


Block Parameters: Integrator1

Integrator

Continuous-time integration of the input signal.

Parameters

External reset: none

Initial condition source: internal

Initial condition:

[.1, |0 5*pi/180 0] [0.1,0,0.087266,0]

Limit output: ☐

Wrap state: ☐

Show saturation port: ☐

Show state port: ☐

Absolute tolerance: auto

Ignore limit and reset when linearizing: ☐

Enable zero-crossing detection: ✓

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

''

Figure 13: Parameter settings for the integrator block located in the plant section of the system's block diagram shown in Figure 1. This is used to implement desired initial conditions using the appropriate conversions.