ME 134 Lab 6a Report

Orion V, Tatsuyoshi Kurumiya, Eric Bermudez
Section 201
April 26, 2024

1 Purpose

The two sessions for Lab 6 are the culmination of the labwork we have been doing for the whole semester. In this lab we have the objective of achieving simulateneous control of the angular position of the pendulum and the horizontal position of the cart on the track using full-state feedback. We will be looking at small perterbations that we apply to the pendulum and a sine wave reference input that the cart tracks for position. We are trying to control both the position of the cart and the angle of the pendulum, so this is our first experience controlling a SIMO (single input multiple output) system using only a single parameter, motor voltage.

2 Pre-Lab

2.1 Equations of Motion of the Mechanical System

The complete free body diagram for our system is shown in Figure 1. The parameters of the system are given in Figure 2.

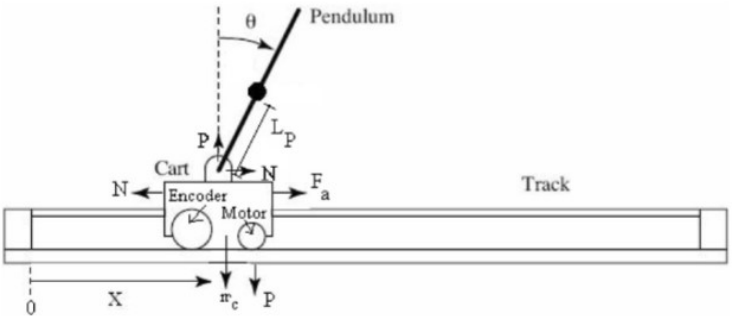

Figure 1: Complete free body diagram for inverted pendulum system

Parameter	Value	Description
	439.6 counts/cm	Resolution of the cart position encoder
	651.9 counts/rad	Resolution of the angle encoder
M	0.94 kg	Mass of cart and motor
m	0.230 kg	Mass of pendulum
L_p	0.3302 m	Pendulum distance from pivot to center of mass
I_c	$m\,L_p^2/3$	Moment of inertia of pendulum about its center
I_e	$4m\,L_p^2/3$	Moment of inertia of pendulum about its end
K_t	$7.67 \cdot 10^{-3}$ Nm/A	Motor torque constant
K_m	$7.67 \cdot 10^{-3}$ Vs/rad	Motor back EMF constant
K_g	3.71	Motor gearbox ratio
R_m	2.6 Ω	Motor winding resistance
r	$6.36 \cdot 10^{-3}$ m	Radius of motor gear
J_m	$3.9 \cdot 10^{-7}$ kg m^2	Motor moment of inertia

Figure 2: Parameters and constants for inverted pendulum system

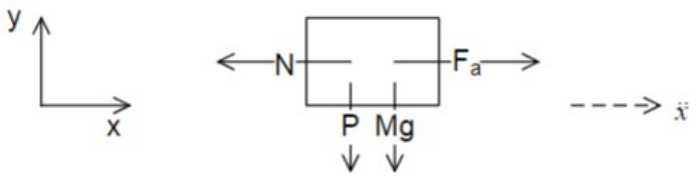

Figure 3: Individual free body diagram for cart

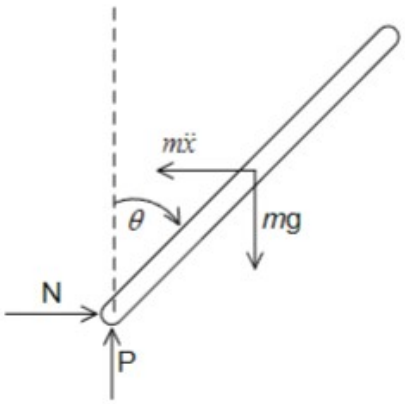

Figure 4: Individual free body diagram for pendulum

Using the free body diagrams of the cart and the pendulum shown in Figures 3 and 4, and knowledge of the small angle approximation that says $\sin\theta \approx \theta$ and $\cos\theta \approx 1$, we derive the following equations of motion for the inverted pendulum-cart system:

Equation of motion of the cart $\rightarrow$ no motion in y: $\sum F_{\mathrm{x}} = M\ddot{x} \rightarrow F_{\mathrm{a}} - N = M\ddot{x}$

where N is reaction force of pendulum on cart.

Horizontal acceleration of pendulum :$a_{\mathrm{p}} = a_{\mathrm{cart}} + a_{\mathrm{tang}} = \ddot{x} + L_{\mathrm{p}}\ddot{\theta}$

with reaction force $N = m(\ddot{x} + L_{\mathrm{p}}\ddot{\theta})$

Now: $F_{\mathrm{a}} - m(\ddot{x} + L_{\mathrm{p}}\ddot{\theta}) = M\ddot{x} \rightarrow F_{\mathrm{a}} = M\ddot{x} + m\ddot{x} + mL_{\mathrm{p}}\ddot{\theta} = (M+m)\ddot{x} + mL_{\mathrm{p}}\ddot{\theta}$

$F_{\mathrm{a}} = (M+m)\ddot{x} + mL_{\mathrm{p}}\ddot{\theta}$

Tangential acceleration $\rightarrow a_{\mathrm{tang}} = a_{\mathrm{t}} = L_{\mathrm{p}}\ddot{\theta}$

Horizontal acceleration $\rightarrow a_{\mathrm{horizontal}} = \ddot{x} + L_{\mathrm{p}}\ddot{\theta}$

Torque around pivot from gravity $\rightarrow \tau_{\mathrm{grav}} = -mgL_{\mathrm{px}}\sin\theta \approx mgL_{\mathrm{p}}\theta$

Torque due to F_{a} accelerating the cart $\rightarrow \tau_{\mathrm{grav}} = mL_{\mathrm{p}}a_{\mathrm{px}} = -(mL_{\mathrm{p}}(\ddot{x} + L_{\mathrm{p}}\ddot{\theta}))$

Now, Newton's 2nd Law: $I\ddot{\theta} = \tau_{\mathrm{net}} = \tau_{\mathrm{cart}} + \tau_{\mathrm{grav}}$

Moment of inertia about a pivot at the end of a rod: $I = \frac{1}{3}mL_{\mathrm{p}}^2$

Now: $\frac{1}{3}mL_{\mathrm{p}}^2\ddot{\theta} = \tau_{\mathrm{net}} = -mL_{\mathrm{p}}\ddot{x} - mL_{\mathrm{p}}^2\ddot{\theta} + mgL_{\mathrm{p}}\theta$

$mL_{\mathrm{p}}\ddot{x} + \frac{4}{3}mL_{\mathrm{p}} - mgL_{\mathrm{p}}\theta = 0$

2.2 Full System Dynamics of Linearized System

We use the motor dynamics derived in Lab 3 in the form $F_\mathrm{a} = f(V, x, \dot{x})$ and substitute them into the linearized cart-pendulum dynamics we just derived to obtain the complete system dynamics. We are interested in outputs of x, the position of the cart, and θ, the pendulum angle. Our control input is V, the voltage applied to the motor. As we have a single input and multiple outputs, this is a SIMO system. The full dynamics are derived as follows:

From Lab 3,

$$F_\mathrm{a} = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V - \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x} - \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2} \ddot{x}$$

So now,

$$(M+m)\ddot{x} + m L_\mathrm{p} \ddot{\theta} = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V - \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x} - \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2} \ddot{x}$$

This gives us the complete dynamics for our system:

$$\left(M + m + \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2}\right) \ddot{x} + m L_\mathrm{p} \ddot{\theta} = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V + \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x} \tag{1}$$

$$m L_\mathrm{p} \ddot{x} + \frac{4}{3} m L_\mathrm{p}^2 \ddot{\theta} - m g L_\mathrm{p} \theta = 0 \tag{2}$$

Now we derive the state space model symbolically for the complete system. We use $\mathbf{x} = [x, \dot{x}, \theta, \dot{\theta}]^T$ as our state vector. Our A and B matrices will be in the form:

$$A = \begin{bmatrix} 0 & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & 0 & a_{34} \\ 0 & a_{42} & a_{43} & 0 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix}$$

Our derivation is as follows, using equations 1 and 2 from above:

$$\text{We have: } \left(M + m + \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2}\right) \ddot{x} + m L_\mathrm{p} \ddot{\theta} = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V + \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x}$$

$$\text{and: } m L_\mathrm{p} \ddot{x} + \frac{4}{3} m L_\mathrm{p}^2 \ddot{\theta} - m g L_\mathrm{p} \theta = 0$$

Solving equation 2 for $\ddot{\theta}$:

$$\ddot{\theta} = \frac{m g L_p \theta - m L_p \ddot{x}}{\frac{4}{3} m L_p^2} = \frac{3}{4 L_p}(g\theta - \ddot{x}) = \frac{3 g \theta}{4 L_p} - \frac{3 \ddot{x}}{4 L_p}$$

Plugging into equation 1:

$$\left(M + m + \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2}\right) \ddot{x} + m L_\mathrm{p} \left(\frac{3 g \theta}{4 L_p} - \frac{3 \ddot{x}}{4 L_p}\right) = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V + \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x}$$

$$\left(M + m + \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2}\right) \ddot{x} + m L_\mathrm{p} \left(\frac{3 g \theta}{4 L_p} - \frac{3 \ddot{x}}{4 L_p}\right) \ddot{x} + \frac{3}{4} g m \theta - \frac{3}{4} m \ddot{x} = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V + \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x}$$

$$\left(M + m + \frac{J_\mathrm{m} K_\mathrm{g}^2}{r^2} - \frac{3}{4} m\right) \ddot{x} = \frac{K_\mathrm{g} K_\mathrm{T}}{r R_\mathrm{m}} V + \frac{K_\mathrm{g}^2 K_\mathrm{m} K_\mathrm{T}}{r^2 R_\mathrm{m}} \dot{x}$$

$$\sigma_{\text{EI}} = EI \times \sqrt{\left(2\frac{\sigma_{f_n}}{f_n}\right)^2 + \left(\frac{\sigma_M}{M}\right)^2 + \left(3\frac{\sigma_L}{L}\right)^2}$$

Now,

$$\ddot{x} = \frac{1}{(M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m)} \left(\frac{K_{\text{g}} K_{\text{T}}}{r R_{\text{m}}} V + \frac{K_{\text{g}}^2 K_{\text{m}} K_{\text{T}}}{r^2 R_{\text{m}}} \dot{x} - \frac{3}{4} g m \theta\right)$$

And from before,

$$\ddot{\theta} = \frac{3}{4L_p}\left[g\theta - \frac{1}{(M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m)} \left(\frac{K_{\text{g}} K_{\text{T}}}{r R_{\text{m}}} V + \frac{K_{\text{g}}^2 K_{\text{m}} K_{\text{T}}}{r^2 R_{\text{m}}} \dot{x} - \frac{3}{4} g m \theta\right)\right]$$

$$\ddot{\theta} = \frac{3}{4L_p}\left[g\theta + \left(\frac{\frac{3}{4} g m \theta}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m}\right)\right] + \frac{\frac{3}{4L_p}}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m} \left(\frac{K_g^2 K_m K_T}{r^2 R_m} \dot{x} - \frac{K_g K_T}{r R_m} V\right)$$

This gives us the state space representation:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \dfrac{-\frac{K_g^2 K_m K_T}{r^2 R_m}}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m} & -\dfrac{\frac{3}{4} g m}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{\frac{3}{4L_p}}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m}\left[\frac{K_g^2 K_m K_T}{r^2 R_m}\right] & \dfrac{3g}{4L_p}\left(1 + \dfrac{\frac{3}{4}m}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m}\right) & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \dfrac{\frac{K_g K_T}{r R_m}}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m} \\ 0 \\ -\dfrac{\frac{3}{4L_p}}{M + m + \frac{J_m K_g^2}{r^2} - \frac{3}{4}m}\left[\frac{K_g K_T}{r R_m}\right] \end{bmatrix} C = \begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Now we plug everything into MATLAB to make it easier to change parameters.

2.3 Analysis and Controller Design

We determine the eigenvalues of the state matrix A and the poles of the state space representation using MATLAB. We also check the stability, controllability and observability of the system. The code to do this is shown in Listing 1.

Listing 1: Optimized Parameters for MHD Drive

```
1
2 % Define the system constants
3 M = 0.94;
```

```
m = 0.230;
L_p = 0.3302;
J_m = 3.9e-7;
K_g = 3.71;
K_t = 7.67e-3;
K_m = 7.67e-3;
r = 6.36e-3;
R_m = 2.6;
g = 9.81;

% Calculate the denominator for the dynamics
denominator = M + m + (J_m * K_g^2)/r^2 - 3/4 * m;

% State-space matrices A and B
A = [0, 1, 0, 0;
     0, -(K_g^2 * K_m * K_t / (r^2 * R_m)) / denominator, -3*m*g/(4*denominator), 0;
     0, 0, 0, 1;
     0, (3/(4*L_p) * (K_g^2 * K_m * K_t / (r^2 * R_m))) / denominator, 3*g/(4*L_p)*(1 + (3/4*m)
     / denominator), 0];

B = [0;
     K_g * K_t / (r * R_m * denominator);
     0;
     -3/(4*L_p*denominator) * (K_g * K_t / (r * R_m))];

% Output matrices C and D
C = [1, 0, 0, 0;   % Output x
     0, 0, 1, 0];  % Output theta

D = [0;
     0];

% Eigenvalues of A
eigenvalues = eig(A);

% Check for stability
isStable = all(real(eigenvalues) < 0);

% Controllability
Co = ctrb(A,B);
isControllable = rank(Co) == size(A,1);

% Observability
Ob = obsv(A,C);
isObservable = rank(Ob) == size(A,1);

% Display results
disp('Eigenvalues of A:');
disp(eigenvalues);
disp(['Is the system stable? ', num2str(isStable)]);
disp(['Is the system controllable? ', num2str(isControllable)]);
disp(['Is the system observable? ', num2str(isObservable)]);
```

Running the code outputs the following results:

```
State-space matrix A:
     0    1.0000         0         0
     0   -6.8123   -1.4973         0
     0         0         0    1.0000
```

```
     0   15.4731   25.6828         0

State-space matrix B:
         0
    1.5226
         0
   -3.4583

Output matrix C:
     1     0     0     0
     0     0     1     0

Direct transmission matrix D:
     0
     0

Eigenvalues of A:
         0
   -7.5515
   -4.1290
    4.8682

Is the system stable? 0
Is the system controllable? 1
Is the system observable? 1
```

We can see that the system has poles at 0, -7.45, -4.46, and 4.09. Since there is a pole in the right half plane, the system is not internally/BIBO stable. However, the system is controllable and observable.

We simulate the output response of the system for a step input in the motor voltage. From the cart, physically we expected the position of the cart to accelerate in an attempt to prevent the pendulum arm from swinging but fails because of system instability. As for the angle of the pendulum, we physically expect the pendulum to drop and maintain a position in stable equilibrium. However, the system's outputs are state variables in the input. This being the case, the simulated response takes this as an unbounded increase due to the instability of the system. In short, the difference in the physical and simulated system is due to the linearity imposed by the small angle approximation. Due to this discrepancy, the simulated systems response shows a exponential increase in cart position and an exponential decrease in angle position. The MATLAB code is shown in Listing 2.

Listing 2: MATLAB code to run simulation of step input

```
1
2  % Define the system with state-space matrices A, B, C, D
3  sys = ss(A, B, C, D);
4
5  % Time vector for simulation
6  t = 0:0.01:10;  % Time vector
7
8  % Titles and labels for each plot
9  titles = {'Step Response: Position of Cart (x)', 'Step Response: Angle of Pendulum (theta)'};
10 ylabels = {'Position (meters)', 'Angle (radians)'};
11
12 % Create a single figure for all subplots
13 figure;
14
```

```
% Generate step response for each output using subplots
for i = 1:size(C, 1)
    subplot(2, 1, i);
    step(sys(i), t);
    title(titles{i});
    ylabel(ylabels{i});
    xlabel('Time');
    grid on;
end
```

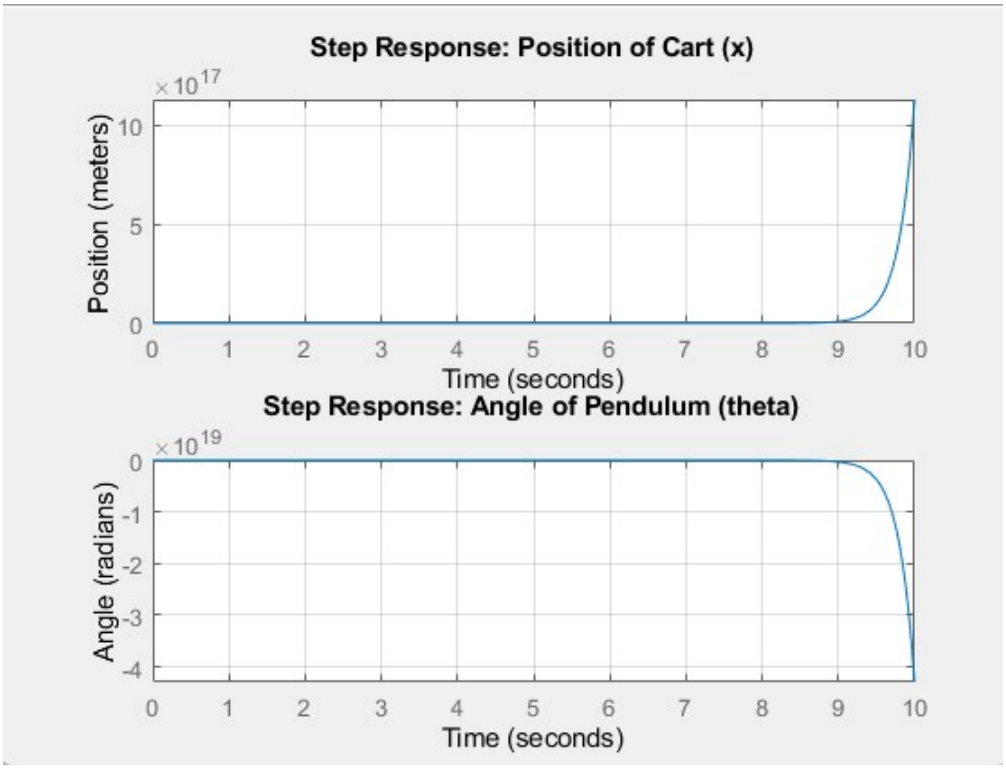

Figure 5: Position of cart and angle of pendulum step response

Now we will use full state-feedback to control the system so we achieve the performance specifications. We assume all state variables are available for measurement and can use them for feedback. Our full-state feedback controller is $u = -Kx$. The gain matrix K is chosen such that the closed-loop eigenvalues lie at some desired values. These desired values are found based on performance specifications and implemented in the system using pole placement. For our purposes in this lab, we want our closed-loop eigenvalues to lie at $s_{1,2} = -2.0 \pm 10j$ and $s_{3,4} = -1.6 \pm 1.3j$

Now we find the feedback gain matrix K of the form $K = [k_1, k_2, k_3, k_4]$. The closed-loop system matrix is given by $A_K = A - BK$. We use the A and B matrices that we found previously to compute A_K as a function of k_i. The code used to accomplish this is shown in Listing 3. We compute the characteristic polynomial $P(K; s) = det(sI - A_K)$ of the closed loop system as a function of k_1 through k_4 and the desired characteristic polynomial $P_{des}(s) = \prod_{i=1}^{4}(s - s_i)$ determined by the desired locations of the closed-loop poles.

Listing 3: MATLAB code find characteristic equation, K matrix

```
% Define symbolic variables
syms k1 k2 k3 k4 s
```

```
K = [k1, k2, k3, k4];  % Feedback gain matrix

% Closed-loop system matrix A_k
A_k = A - B * K;  % A - BK

% Characteristic polynomial of A_k
charPoly = det(s * eye(4) - A_k);

expression = charPoly;

% Simplify expression
simplifiedExpr = simplify(expression);

% Use variable precision arithmetic to reduce the size of numbers
vpaExpr = vpa(simplifiedExpr, 5);  % Display with 5 significant digits

disp('charPoly');
disp(vpaExpr);

% Desired characteristic polynomial from specified eigenvalues
desiredPoly = expand((s - (-2 + 10*j))*(s - (-2 - 10*j))*(s - (-1.6 + 1.3*j))*(s - (-1.6 - 1.3*
    j)))

% Equate coefficients and solve for k1, k2, k3, k4
equations = coeffs(charPoly, s, 'All') == coeffs(desiredPoly, s, 'All');
solutions = solve(equations, [k1, k2, k3, k4]);

k1_numeric = double(solutions.k1);
k2_numeric = double(solutions.k2);
k3_numeric = double(solutions.k3);
k4_numeric = double(solutions.k4);

% Display the numeric results
disp('Numeric values for K:');
disp([k1_numeric, k2_numeric, k3_numeric, k4_numeric]);
K = [k1_numeric, k2_numeric, k3_numeric, k4_numeric];

% Desired pole locations
desired_poles = [-2 + 10i, -2 - 10i, -1.6 + 1.3i, -1.6 - 1.3i];

% Calculate K using the place command
K_placed = place(A, B, desired_poles);

% Display the calculated K
disp('Calculated K using place:');
disp(K_placed);

disp('Difference between placed K and calculated K:');
disp(K_placed - K);
```

The code returns:

```
Simplified charPoly:
1.6187e-16*k3*s - 151.79*s - 33.926*k2*s - 33.926*k1 + 1.5226*k1*s^2 + 1.5226*k2*s^3
- 3.4583*k3*s^2 + 1.6187e-16*k4*s^2 - 3.4583*k4*s^3 - 25.683*s^2 + 6.8123*s^3 + s^4

desiredPoly =
```

```
s^4 + (36*s^3)/5 + (2421*s^2)/20 + (1749*s)/5 + 442

Numeric values for K:
  -13.0283  -14.7848  -48.1649   -6.6214

Calculated K using place:
  -13.0283  -14.7848  -48.1649   -6.6214

Difference between placed K and calculated K:
   1.0e-14 *

   -0.7105         0   -0.7105    0.5329
```

In this code, we also compared the coefficients of $P(K;s)$ and $P(s)$ and determined the system of linear equations that the gains $k_1, ..., k_4$ have to satisfy. We used MATLAB to solve for K with our numerical values. We then verified the results using the `place` command to place the poles and found the difference between the K calculated with placed poles and the K calculated from the characteristic polynomials. Our results show us that these are essentially the same.

Assuming that we are designing our controller with full state feedback with the input $u = K(r - x)$, the dynamics of the closed loop system are $\dot{x} = Ax + Bu = A_K x + BK_r$, where r is our reference input. Using our calculated gain matrix K, we use MATLAB to calculate the closed loop transfer function from the first component in the reference position input, $\mathbf{r}$ to the output x, the position output of the cart. We plot a step response for this transfer function and show that the DC gain for the transfer function is unity. This code is shown in Listing 4. The plot is shown in Figure 6.

Listing 4: MATLAB code to find closed loop transfer function and DC gain

```
% A, B, C, D as previously defined

% Matrices for the closed-loop system
A_cl = A - B * K;
B_cl = B * K(:, 1);  % first component of K only for the position reference
C_cl = [1, 0, 0, 0];  % Output matrix focused on the position of the cart
D_cl = 0;

% state-space model for the closed-loop system
sys_cl = ss(A_cl, B_cl, C_cl, D_cl);

% transfer function from r (position reference) to x (position of the cart)
T_cl = tf(sys_cl);

disp('Closed-loop Transfer Function from r (position reference) to x (position of the cart):');
disp(T_cl);

% Plot the step response
figure;
step(T_cl);
title('Step Response of Closed-Loop System from Position Reference');
xlabel('Time');
ylabel('Position of Cart (x)');
grid on;

% Check the DC Gain
dc_gain = dcgain(T_cl);
disp('DC Gain of the transfer function:');
disp(dc_gain);
```

The code returns:

```
  Closed-loop Transfer Function from r (position reference) to x (position of the cart):
tf with properties:

    Numerator: {[0 0 -19.8367 -3.5237e-14 442.0000]}
    Denominator: {[1 7.2000 121.0500 349.8000 442.0000]}

    DC Gain of the transfer function:
    1.0000
```

This gives us the transfer function:

$$TF = \frac{-19.8s^2 - 3.52e^{-14}s + 442}{s^4 + 7.2s^3 + 121.1s^2 + 350s + 442}$$

Letting the position reference input, $R_{pos}(s)$, be a sine wave, we plotted the position output using frequencies

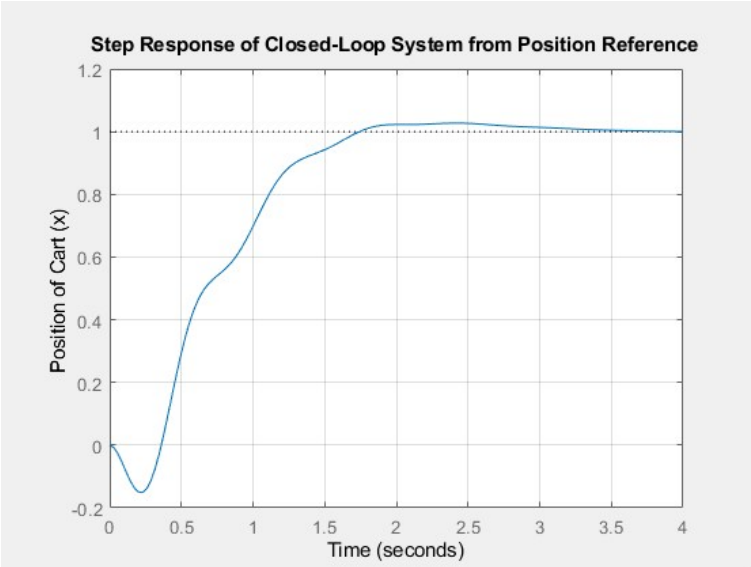

Figure 6: Step response for closed loop transfer function from reference to output

$\omega = 1, 2, 5, rads/s$. We plot these on separate plots with their corresponding reference input superimposed. We also compute the gain from the ratio of the amplitude of the system's output to the amplitude of the input sine wave. We decided to use time-domain and showed 3 periods of data. The code for this task is given in Listing 5. The plots are shown in Figure 7.

Listing 5: MATLAB code to plot position output with sine wave reference signal

```
% Ensure B_cl is configured for a single input
B_cl = B * K(:, 1);  % Assuming K affects only the first state

% Create the state-space object
sys_cl = ss(A - B * K, B_cl, [1 0 0 0], 0);

% Define the frequencies for sinusoidal input
omegas = [1, 2, 5];  % rad/s

% in time domain
for i = 1:length(omegas)
    omega = omegas(i);
```

```
13
14      % time vector to cover at least 3 periods
15      T_final = 2 * pi * 3 / omega;  % Three periods
16      t = linspace(0, T_final, 1000);  % Define time vector
17
18      % Define the sinusoidal input signal
19      R_pos = sin(omega * t)';
20
21      % Ensure R_pos is a properly formatted column vector
22      if size(R_pos, 2) > 1
23          R_pos = R_pos';  % Make sure R_pos is a column vector
24      end
25
26      % Simulate the system response
27      try
28          [Y, T, X] = lsim(sys_cl, R_pos, t);
29      catch ME
30          disp('Error using lsim:');
31          disp(ME.message);
32          continue;  % Skip to the next loop iteration if error occurs
33      end
34
35      % Plot the results
36      subplot(length(omegas), 1, i);
37      plot(T, R_pos, 'r--'); hold on;
38      plot(T, Y, 'b-');
39      title(['Response to Sinusoidal Input at \omega = ', num2str(omega), ' rad/s']);
40      xlabel('Time (s)');
41      ylabel('Position of Cart (x)');
42      legend('Input R_{pos}(t)', 'Output x(t)', 'Location', 'best');
43      grid on;
44
45      % Calculate and display the gain
46      input_amplitude = max(abs(R_pos));
47      output_amplitude = max(abs(Y));
48      gain = output_amplitude / input_amplitude;
49      disp(['Gain for omega = ', num2str(omega), ' rad/s: ', num2str(gain)]);
50  end
```

3 Lab

In the lab portion, we implement the state-feedback controller that we described in the pre-lab and run it on the hardware.

3.1 Implementing the Controller in Simulink

We build our Simulink model for the state feedback system with the hardware blocks that we need. As in previous labs, we use the HIL Initialize, Write Analog, and Read Encoder Timebase blocks to connect to the cart-pendulum system hardware. We use dx/dt blocks to approximate $\dot{x}$ and $\dot{\theta}$ since the encoders provide only x and θ. We use Mux blocks to combine these 4 signals into a "vector" signal that carries them together, with gain blocks to implement K and to-workspace blocks to pull our data together in MATLAB. We put a saturation block set to $\pm 6V$ before the Analog Output block in order to prevent gear slipping and this was the only measure that needed to be taken for prevention. Our Simulink model is shown in Figure 8.

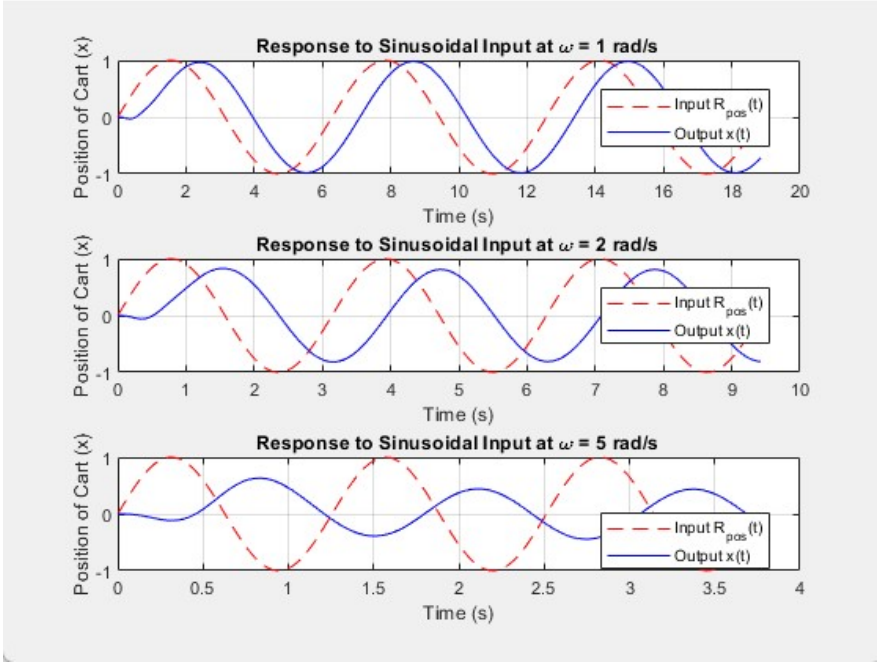

Figure 7: Position outputs from various sinusoidal signals as reference inputs

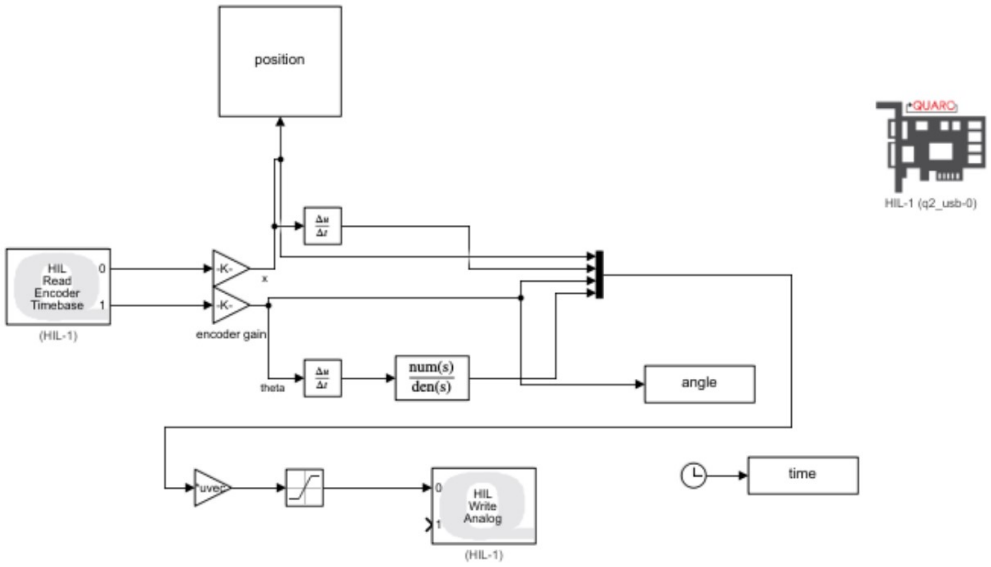

Figure 8: Simulink model for implementing controller on hardware

3.2 Running the Controller on Hardware

3.2.1

We are now ready to run the controller on the hardware. We have set our reference $\mathbf{r}$ to $\mathbf{r} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$ and when we ran the controller, the system was able to balance right away. We made sure to hold the pendulum exactly vertical as the system starts because when the encoder powers up, it takes the current angular position to be $\theta = 0$ and then tries to balance the hardware around that angle. If the hardware connected with the pendulum in stable equilibrium the stable equilibrium position, the cart would not move because there would be no motion in the pendulum. Formally, a body is at a point of stable equilibrium when the body's mass is bellow the body's pivot point. Essentially, the angle $\theta = 0$ would represent pendulum arm pointing downward, and there is no control necessary to provide stability. We heard a little motor jittering during the initial balancing, so we implemented a low-pass filter on the angular position signal which can be seen on the Simulink model in Figure 8. Low pass filters have the general transfer function $\frac{K}{1+s/\omega_c}$. Our GSI provided values as a good starting point for tuning, where $K = 3.5$ and $\omega_c = 100$. These values worked well for us and we didn't need to tune them for our system.

3.2.2

We applied small perturbations to the pendulum and watched the system's response. In perturbing the top of the pendulum in a given direction, we noted that the controller actuated the cart in the same direction as the pendulum was tipping in order to compensate for the movement and restore balance. We plot the variation of the cart and pendulum position with time while introducing small perturbations about the equilibrium value. The cart position over time is shown in Figure 9, the pendulum position is shown in Figure 10. Our code for plotting is given in Listing 6.

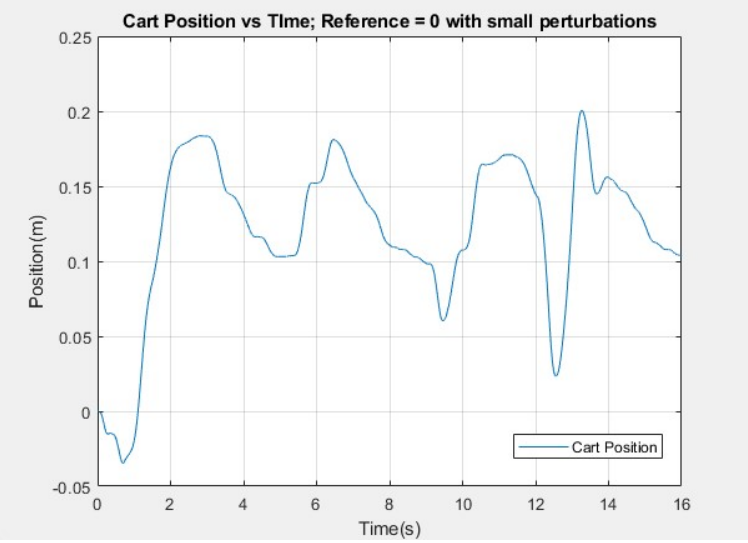

Figure 9: Plot of cart position over time with small perturbations

Listing 6: MATLAB code to plot cart and pendulum position over time with perturbations

```
1
2 K=[  -13.0283  -14.7848  -48.1649   -6.6214];
3 close all;
4 freq = 5;
5 amp = .1;
6 xlabel("Time [s]")
7 figure
8 plot(time, position)
```

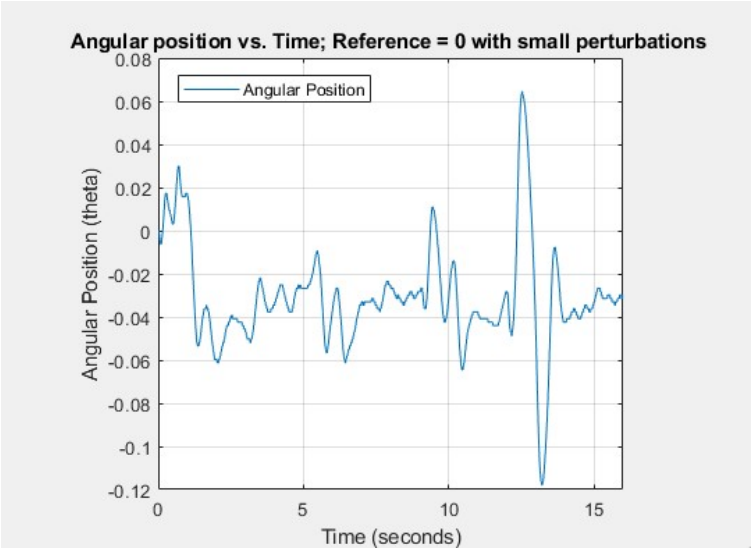

Figure 10: Plot of pendulum position (angle) over time with small perturbations

```
title(["Position vs. Time with Sin input, freq = " + freq + "rad/s, Amplitude =" + amp "m"]);
figure
plot(time, angle)
title(["Angle vs. Time with Sin input, freq = " + freq + "rad/s, Amplitude =" + amp "m"]);
figure
plot(time, velocity)
title(["Velocity vs. Time with Sin input, freq = " + freq + "rad/s, Amplitude =" + amp "m"]);
figure
plot(time, omega)
title(["Angular Velocity vs. Time with Sin input, freq = " + freq + "rad/s, Amplitude =" + amp
    "m"]);
```

Overall, the controllers general performance was satisfactory. Though it continued to oscillate slightly after perturbation, the pendulum maintained an inverted position and the within center of the track. We believe that the pendulum sustained oscillations after perturbations because the system was undergoing a dynamic mode of unstable equilibrium. That is, the system was dynamic in nature, but bound within certain limits such that desired performance was still met.

3.2.3

Now we introduce a sine wave reference signal to our system. Our input is $\mathbf{r} = \begin{bmatrix} M \sin \omega t & 0 & 0 & 0 \end{bmatrix}^T$. The Simulink model is shown in Figure 11. We gave the signal an amplitude of $M = 0.1m$ and considered the response of the system for the frequencies $\omega = 1, 2, 5 \ rad/s$. Our system seemed to be taking a little bit to get balanced, so we added a Transport Delay block with a value of 2 seconds to our system to ensure that the system had time to stabilize before data was extracted into MATLAB. Our system responses for cart position, cart velocity and pendulum angle are plotted in the following subplots; $\omega = 1 \ rad/s$ in Figure 12, $\omega = 2 \ rad/s$ in Figure 13, and $\omega = 5 \ rad/s$ in Figure 14. General code used to format the plots is shown in Listing 7.

Listing 7: General code used to format the saved MATLAB figures

```
%% Lab 6a plotting %%

% Load the figure
fig = openfig(['4.2.2Position.fig'], 'reuse');
```

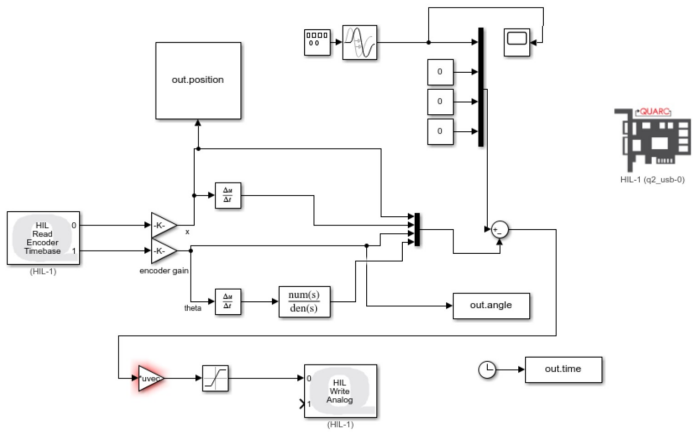

Figure 11: Simulink model for sinusoidal reference input

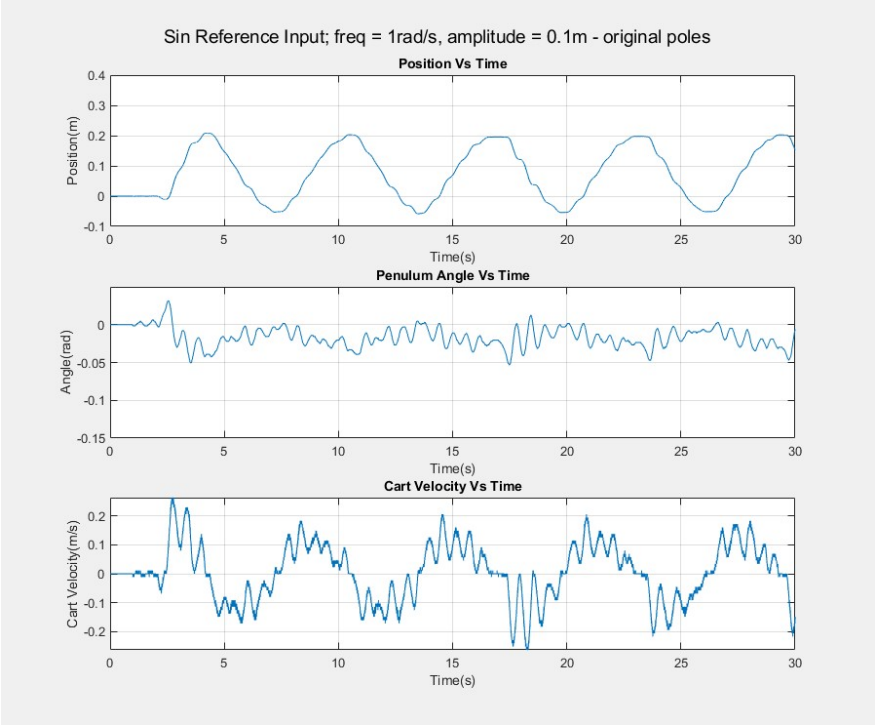

Figure 12: Subplots for $x, \dot{x}, \theta$; where $\omega = 1rad/s$

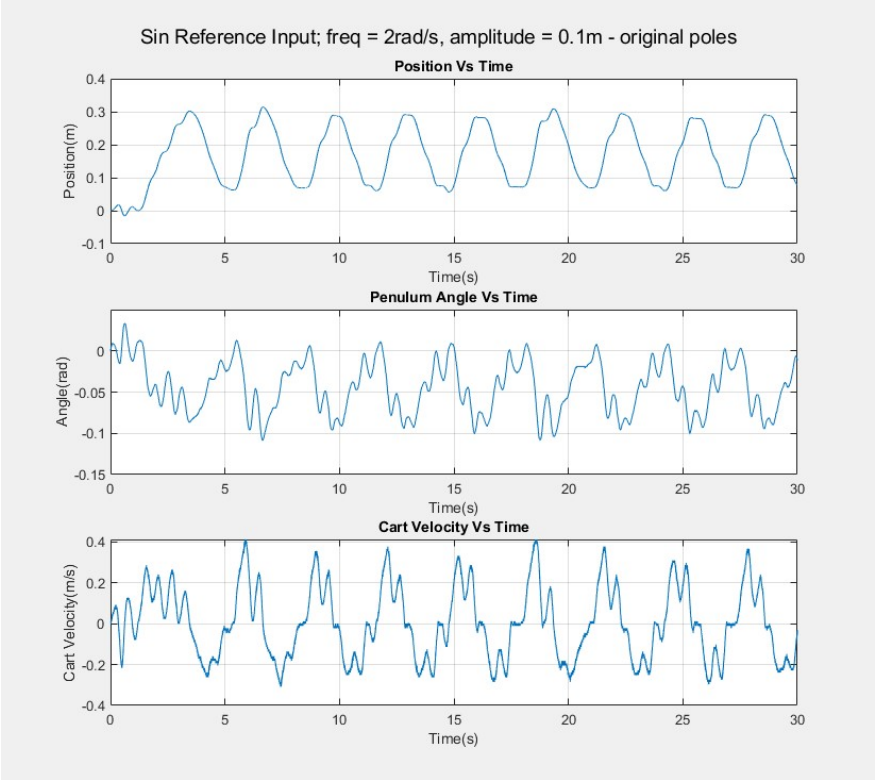

Figure 13: Subplots for $x, \dot{x}, \theta$; where $\omega = 2rad/s$

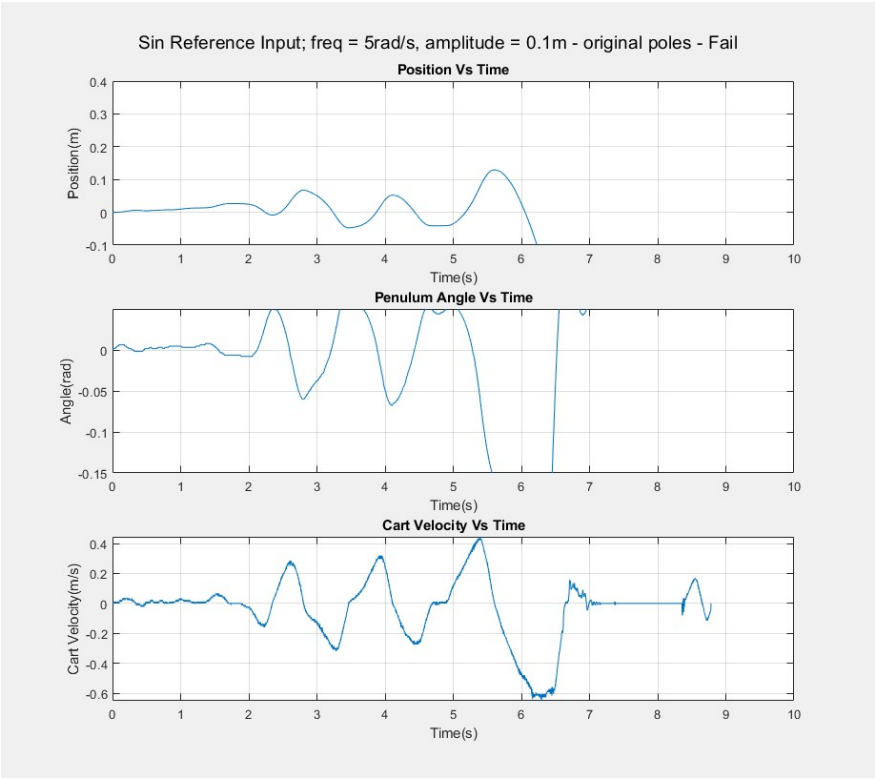

Figure 14: Subplots for $x, \dot{x}, \theta$; where $\omega = 5rad/s$. Failure to stabilize.

```
% Access the axes handle
ax = gca;
grid(ax, 'on');

% Change title and axis labels
xlabel(ax, 'Time(s)');
ylabel(ax, 'Position(m)');
title('Cart Position vs TIme; Reference = 0 with small perturbayion')

% Add or modify legend
legend(ax, 'show', 'Location', 'best');

% Set new axis limits
 xlim(ax, [0 16]);
% ylim(ax, [ymin ymax]);

%% saved subplot needs changes

fig = openfig('5.2.3asubplotsw=5.fig', 'reuse');

% Find all axes in the figure
allAxes = findall(fig, 'type', 'axes');
sgtitle('Sin Reference Input; freq = 5rad/s, amplitude = 0.1m - original poles - Success');

title(allAxes(4), 'Cart Position Vs Time');
xlabel(allAxes(4), 'Time(s)');
ylabel(allAxes(4), 'Position(m)');
xlim(allAxes(4), [0 30]);
ylim(allAxes(4), [-0.05 0.15]);
grid(allAxes(4), 'on');

% Modify second subplot
title(allAxes(3), 'Pendulum Angle Vs Time');
xlabel(allAxes(3), 'Time(s)');
ylabel(allAxes(3), 'Angle(rad)');
xlim(allAxes(3), [0 30]);
ylim(allAxes(3), [-0.09 0.05]);
grid(allAxes(3), 'on');

% Modify third subplot
title(allAxes(2), 'Cart Velocity Vs Time');
xlabel(allAxes(2), 'Time(s)');
ylabel(allAxes(2), 'Velocity(m/s)');
xlim(allAxes(2), [0 30]);
ylim(allAxes(2), [-0.21 0.25]);
grid(allAxes(2), 'on');

% Modify fourth subplot
title(allAxes(1), 'Pendulum Angular Velocity Vs Time');
xlabel(allAxes(1), 'Time(s)');
ylabel(allAxes(1), 'Angular Velocity(rad/s)');
xlim(allAxes(1), [0 30]);
ylim(allAxes(1), [-0.35 0.40]);
grid(allAxes(1), 'on');
%
%% subplot from multiple figures

fig1 = openfig('4.2.3aAnglew=5.fig', 'reuse');
ax1 = get(fig1, 'Children');  % Get the axes of the figure
```

```
data1x = get(get(ax1, 'Children'), 'XData');  % Get x-data
data1y = get(get(ax1, 'Children'), 'YData');  % Assuming you want to replot y-data

fig2 = openfig('4.2.3aVelocityw=5.fig', 'reuse');
ax2 = get(fig2, 'Children');
data2x = get(get(ax2, 'Children'), 'XData');  % Get x-data
data2y = get(get(ax2, 'Children'), 'YData');

fig3 = openfig('4.2.3aPositionw=5.fig', 'reuse');
ax3 = get(fig3, 'Children');
data3x = get(get(ax2, 'Children'), 'XData');  % Get x-data
data3y = get(get(ax3, 'Children'), 'YData');

newFig = figure;
sgtitle('Sin Reference Input; freq = 5rad/s, amplitude = 0.1m - original poles - Fail');

subplot(3, 1, 1);
plot(data3x, data3y);
grid on
title('Position Vs Time');
xlabel('Time(s)');
ylabel('Position(m)');
xlim([0 10])
ylim([-0.1, 0.4])

subplot(3, 1, 2);
plot(data1x, data1y);
grid on
title('Penulum Angle Vs Time');
xlabel('Time(s)');
ylabel('Angle(rad)');
xlim([0 10])
ylim([-.15 0.05])

subplot(3, 1, 3);
plot(data2x, data2y);
grid on
title('Cart Velocity Vs Time');
xlabel('Time(s)');
ylabel('Cart Velocity(m/s)');
xlim([0 10])
```

Our system was able to balance without problem for sine wave inputs of $\omega = 1\ rad/s$ and $\omega = 2\ rad/s$ but would quickly become unstable and would need to be stopped when running it at $\omega = 5\ rad/s$. We began our debugging process with adding another low-pass filter to the cart position signal in addition to the filter on the angular position signal. We ran this setup at $\omega = 5\ rad/s$ and were able to easily achieve stability in the system. The updated Simulink model is show in Figure 15. The results from this trial with the addition low-pass filter are shown in Figure 16.

Analyzing the response from the $1\ rad/s$, $2\ rad/s$, $5\ rad/s$ reference inputs in the prelab, we get approximate gains of 1, 0.8, and 0.6 respectively. For the actual system the gains for the 0.1 amplitude input and frequencies of $1\ rad/s$, $2\ rad/s$, $5\ rad/s$ we get gains of approximately 0.7, 0.9, and 0.9, respectively. Comparing the results we see they vary quite significantly. In all cases in the simulated response the oscillations were centered about initial position of the cart. However, the hardware response has the oscillations centered away from the carts initial position. This is likely due to some non-linearity within the system that causes a sort of over shot to correct for the initial displacement caused by the input. In the case of the $1\ rad/s$ input, the difference is slight but noticeable, with a lower gain in the hardware system than

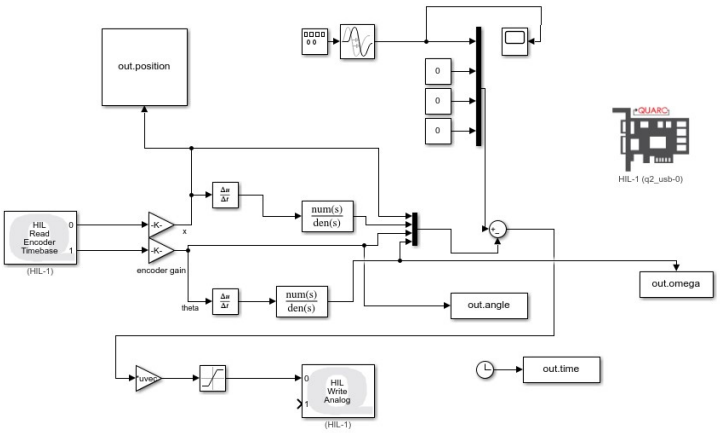

Figure 15: Updated block diagram with 2 low-pass filters

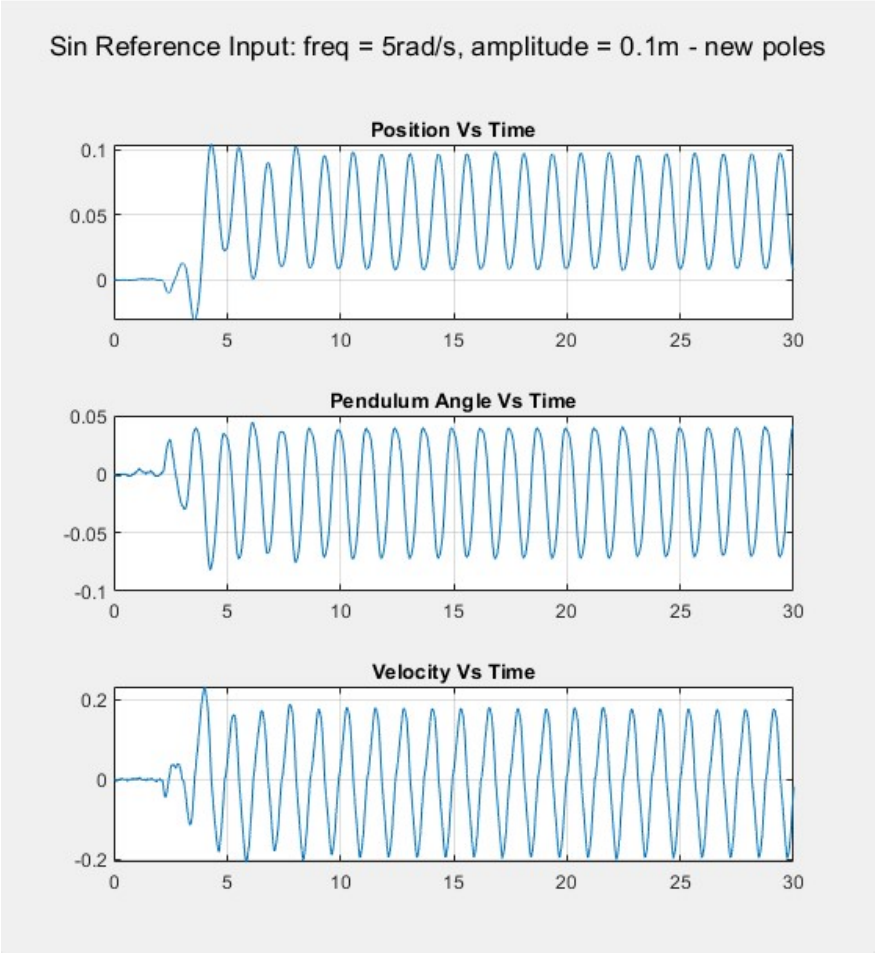

Figure 16: Subplots for $x, \dot{x}, \theta, \dot{\theta}$; where $\omega = 5rad/s$. Corrected and stabilized.

in the simulation. In the case of the $2/enspacerad/s$ input, the difference in gain is larger in the hardware implementation than in the simulation. In the case of the $5/enspacerad/s$ input, the gain was also notably larger than the simulated gain. Additionally, it is noteworthy to mention that the gain calculated for $5/enspacerad/s$ input in the hardware was only able to run for a short while before becoming unstable. We believe these discrepancies to be due to overshoots in the actual hardware system that are unaccounted for in simulation. This causes could arise from non-linearities and time variant phenomena such as backlash, friction, saturation, and other such system imperfections.
Next we picked a frequency that was a little easier for our system to control, $\omega = 1\ rad/s$, and chose different closed-loop poles. Previously, our poles had been located at $-2+j10, -2-j10, -1.6+j1.3, -1.6-j1.3$, giving us $K = [-13.0283 - 14.7848 - 48.1649 - 6.6214]$. For our first trial, we decided to move the poles slightly more negative than they previously were. Our new poles were $-3 + j10, -3 - j10, -2 + j1.3, -2 - j1.3$. Running this through the code shown in Listing 3, we end up with $K = [-18.2812 - 18.3319 - 55.5783 - 8.9927]$. The plots are shown in Figure 17.

For our second trial, we used poles at $-5 + j10, -5 - j10, -3 + j1.3, -3 - j1.3$ and this resulted in $K = [-39.3871 - 29.7320 - 81.3526 - 15.7467]$. The plots are shown in Figure 18. Changing the location of the poles changes the transient characteristics of the response. Initially we had chosen poles that corresponded to a desired system behavior, but also calculated a gain such that this behavior was realized. By keeping the same gain and chaining the position of the desired poles we change the transient behavior of the system's response. Looking at the plots corresponding to the position, our first change of poles resulted in stabilization of the system. The second change of poles resulted in a reduction in the oscillation of the cart position. Additionally, the response of the angle and velocity show a sort of superposition of two wave forms. One with a larger amplitude and smaller frequency. The another that looks like noise, but is more likely to be a more frequent sinusoidal signal with a smaller amplitude. This pattern repeats itself when the we changed the poles a third time. However the amplitude of the lower frequency wave form is reduced while the amplitude of the higher frequency wave form looks increased. Another noteworthy observation is that the gain looks to be approximately unity in all instances where the poles were changed.

3.2.4

Finally, we plot the cart's velocity $\dot{x}$ and pendulum's angular velocity $\dot{\theta}$ from the x and θ signals obtained with numerical differentiation blocks in the model. We used a sine input with an amplitude of $0.1m$ and frequency of $\omega = 1\ rad/sec$. We observed that the signal is smoothed by the implementation of the numerical differentiation blocks and is shown in Figure 18. Closer inspection of these plots seems to show a sort of phase shift in the velocity response of the system. Additionally, the angular velocity response does not seem to have a phase shift but has a drastic reduction in amplitude.

4 Conclusion

In this lab, we incorporated full-state feedback control of an inverted pendulum on a cart, starting from deriving the dynamics of the system. We linearize and translate the dynamics of the system in state space form, with our states being the cart's position and velocity and the pendulum's angular position and angular velocity. Using a full-state feedback controller in the form $u = -Kx$, we were able to stabilize the otherwise unstable system by placing the system's eigenvalues at our desired positions, in the left hand plane. Translating our results into the hardware, we used a similar K matrix as derived from the pre-lab to control our system.
One difference from the simulation and the hardware set-up was the need to implement a low-pass filter since we measure the linear and angular velocity values by taking the derivative of our position and angular position signals, which will cause an inevitably noisy reading. During lab 6a, we were unsuccessful in stabilizing the pendulum with a sinusoidal input of 5 rad/s. However, upon retrying our system in lab 6b, we were able to stabilize the system at 5 rad/s by using a second low-pass filter. The successful results from the modified full state feedback controller will be included and elaborated upon in lab report 6b.
The greatest challenges we faced with the hardware were in identifying which parameters to change to improve the response of our system. Since the K-values we used closely matched an ideal system, we iterated instead on the cutoff frequencies and gains of the low-pass filters to account for the imperfections of our

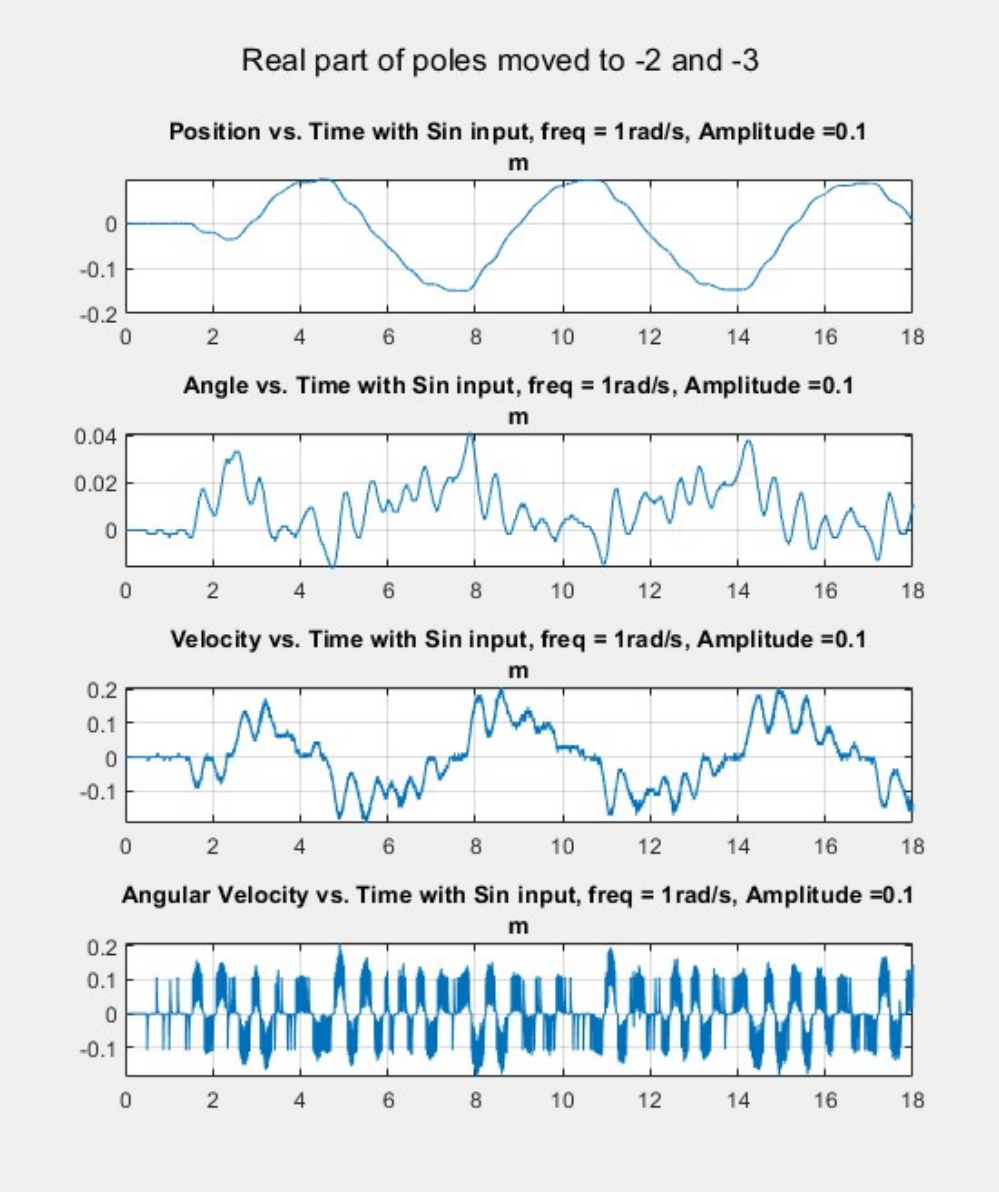

Figure 17: Subplots for $x, \dot{x}, \theta, \dot{\theta}$ with poles moved slightly negative.

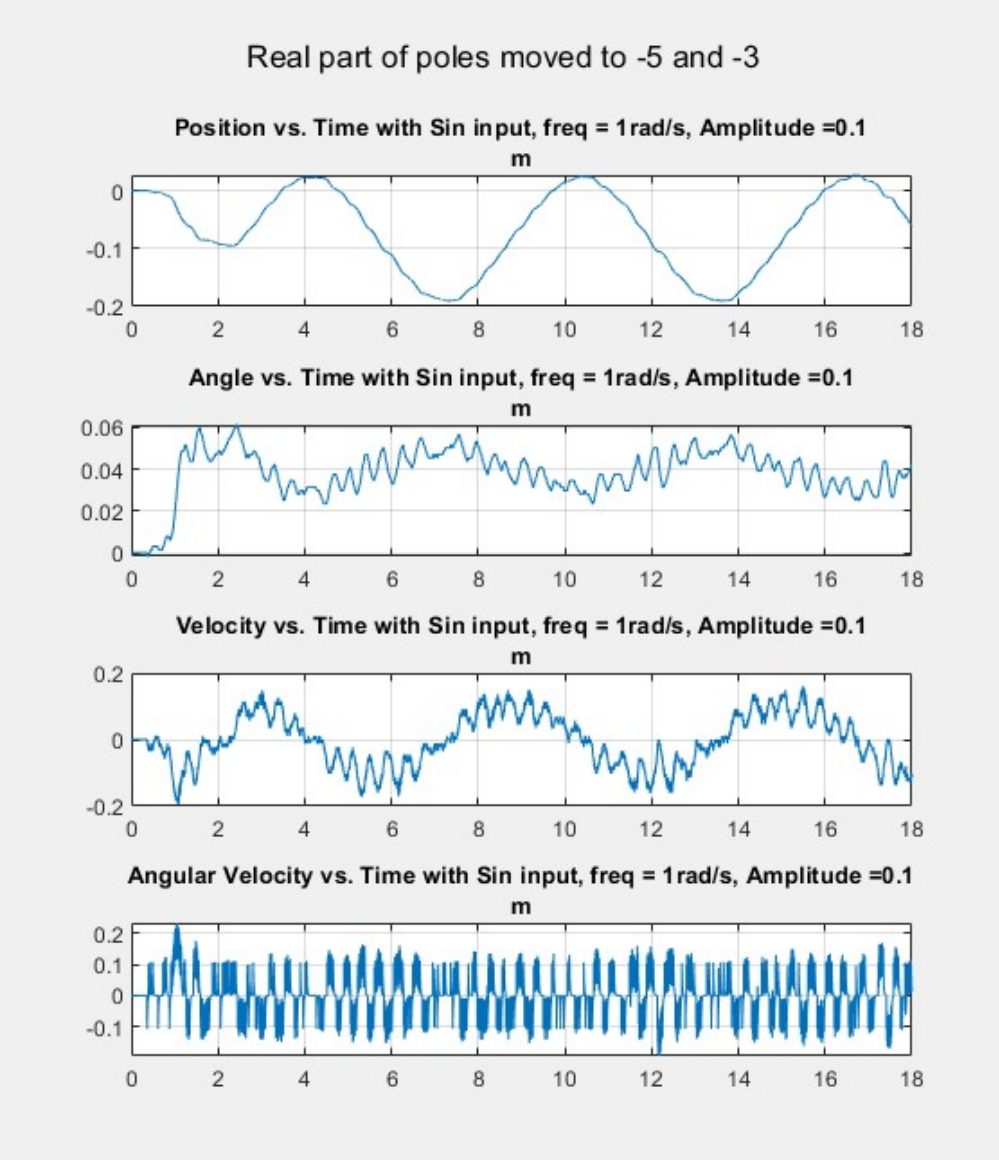

Figure 18: Subplots for $x, \dot{x}, \theta, \dot{\theta}$ with poles moved even more negative.

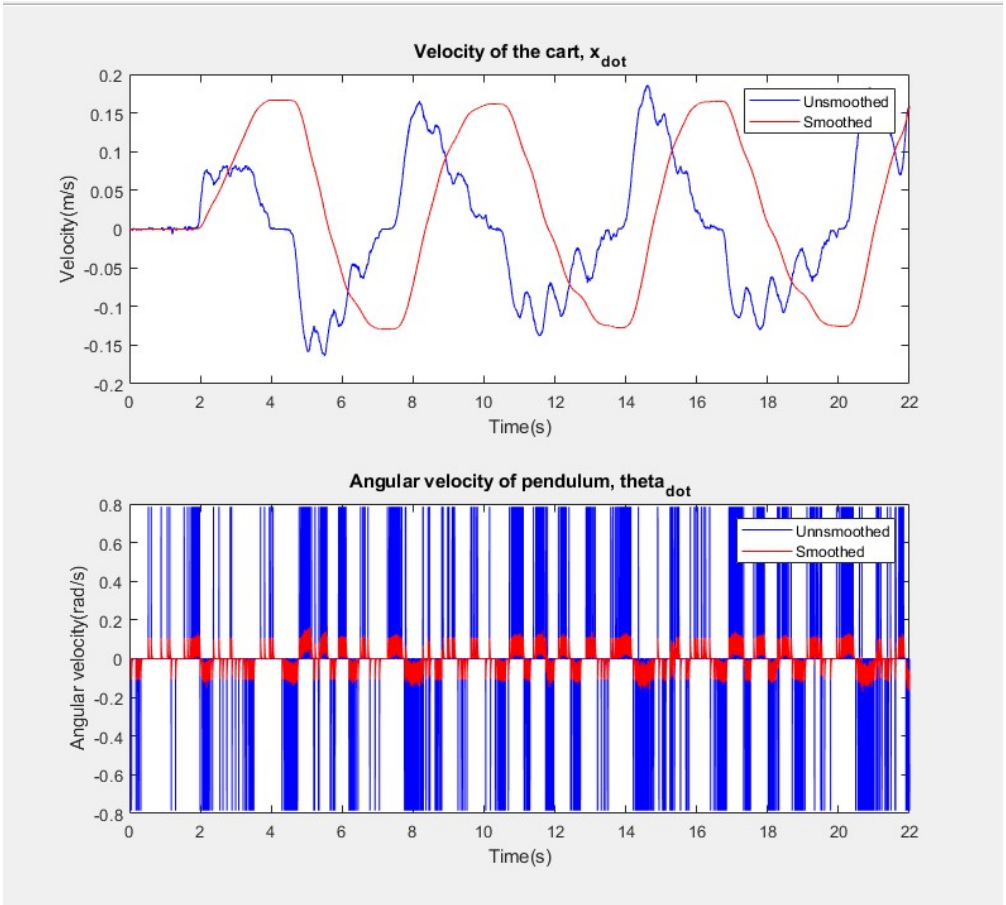

Figure 19: Cart velocity and angular velocity obtained numerically with smoothed signal

signal processing. This improved our controller and ultimately led to the success for every input signal. This lab was instrumental in teaching us the steps and details of creating a controller from scratch, following the process of calculating the dynamics, deriving ideal controller gains, collecting and analyzing data, and tuning our controller parameters to ensure the successful resposne for our system.