ESE 441

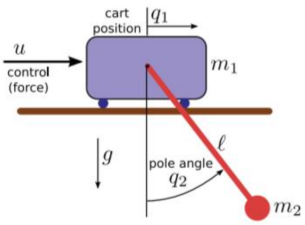
Final Project

Prince John
502013

Collaborated with: George Mitrev, Blue Truong

May 10, 2023

1 Cart-Pendulum Simulation and Stabilization



$$\ddot{q}_1 = \frac{\ell\, m_2 \sin(q_2)\, \dot{q}_2^2 + u + m_2\, g\, \cos(q_2)\, \sin(q_2)}{m_1 + m_2\,(1 - \cos^2(q_2))}$$

$$\ddot{q}_2 = -\frac{\ell\, m_2 \cos(q_2)\, \sin(q_2)\, \dot{q}_2^2 + u\, \cos(q_2) + (m_1 + m_2)\, g\, \sin(q_2)}{\ell\, m_1 + \ell m_2\,(1 - \cos^2(q_2))}$$

Figure 1: Cart Pendulum model from the Assignment

1.1 State Space Representation of the System

The dynamics of this system are described by non-linear equations, therefore I have the non-linear state space representation described below.

The state for this system can be represented by x,

$$x = \begin{pmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{pmatrix}$$

Where q_1 is the cart position from the origin, q_2 is the pole angle, $\dot{q}_1$ is the velocity of the cart, and $\dot{q}_2$ is the angular velocity of the pole pendulum.

I can then represent $\dot{x}$ as,

$$\dot{x} = f(x(t), u(t)) \quad \text{with} \quad f : \mathbb{R}^{(4+1)} \longrightarrow \mathbb{R}^4 \quad ; \quad f = (f_1, f_2, f_3, f_4)^T$$

and,

$$f_1 = \dot{q}_1 = x_3 \,;\, f_2 = \dot{q}_2 = x_4$$

$$f_3 = \ddot{q}_1 = \frac{l m_2\, sin(q_2)\, \dot{q}_2^2 + u + m_2\, g\, cos(q_2)\, sin(q_2)}{m_1 + m_2(1 - cos^2(q_2))}$$

$$f_4 = \ddot{q}_2 = -\frac{l\, m_2\, cos(q_2)\, sin(q_2)\, \dot{q}_2^2 + u\, cos(q_2) + (m_1 + m_2)\, g\, sin(q_2)}{l\, m_1 + l\, m_2(1 - cos^2(q_2))}$$

1.2 Visualization of the cart system

The anonymous function `cartpendulum(t,x)` represents the $\dot{x}(t)$ state vector. Therefore, I have written out the equations for $\ddot{q}_1$ and $\ddot{q}_2$ for the empty rows of the function. Since this is the autonomous case, I've set $u = 0$. My implementation for the no-input cart pendulum system is given below.

```
cartpendulum = @(t,x) [x(3);
                       x(4);
                       (l*m2*sin(x(2))*x(4)^2 + m2*g*cos(x(2))*sin(x(2)))/(m1+m2
    *(1 - cos(x(2))^2));
```

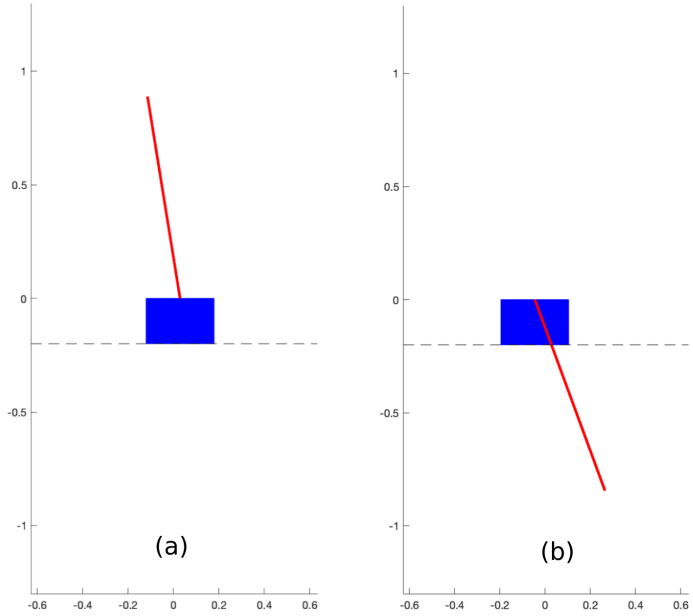


Figure 2: Visualization of the autonomous system response. (a) close to the start of the simulation, (b) mid swing

```
                    (l*m2*cos(x(2))*sin(x(2))*x(4)^2 + (m1 + m2)*g*sin(x(2)))/(
    l*m1 + l*m2*(1 - cos(x(2))^2))];
```

Observations from the visualization:

The visualization starts with the pendulum near the top just slightly off center. This can be seen in figure 2 (a). The blue cart then moves right as the pendulum swings down. The pendulum then swings through under the cart and goes up to the other side at the same height as the original location. The cart goes to the left while the pendulum goes back up to the other side.

This oscillation of the cart and the pendulum continues indefinitely since we have not modeled any air resistance in the state space model. The cart keeps oscillating about the 0 point in the x axis and the pendulum kept oscillating between $\pi + 0.1$ rad and $\pi - 0.1$ rad.

1.3 System Visualization with provided inputs

I created a wrapper function to use `ODE45` with a set of inputs. Since the `ODE45` function evaluates only few points along the time span I used `interp1(t_span,u,t)` function to interpolate which u value will be the closest to the sampled t value. My wrapper function implements the original `cartpendulum` function with an interpolated u value instead of having the u terms from the $\ddot{x}$ vector set to 0. My implementation is shown below.

```
function x = cartpendulum_wrapper(t, x, t_span, u)
  m1=1; m2=0.3; l=0.5; g=9.81;
  u = interp1(t_span,u,t);
  x = [x(3);
       x(4);
```

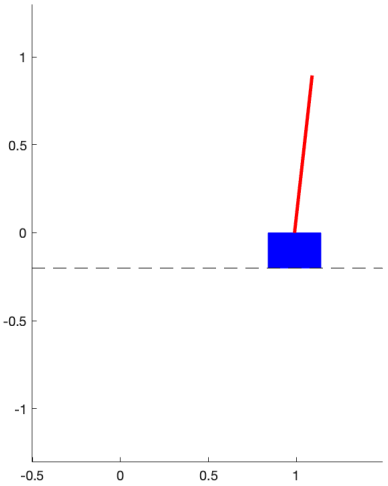


Figure 3: The final state of the cart pendulum with provided inputs.

```
6         (l*m2*sin(x(2))*x(4)^2 + u+ m2*g*cos(x(2))*sin(x(2)))/(m1+m2*(1 - cos(x(2))
      ^2));
7       -1*(l*m2*cos(x(2))*sin(x(2))*x(4)^2 + u*cos(x(2))+ (m1 + m2)*g*sin(x(2)))/(l
      *m1 + l*m2*(1 - cos(x(2))^2))];
8
9   end
```

Observations from the visualization:
I used the visualization script from earlier to create an animation with the set of inputs provided with the assignment. The visualization starts with the pendulum at the bottom. It shows the cart moving back and forth repeatedly to use the swing of the pendulum to propel itself to an upright state. The inputs nudge the pendulum with every back and forth motion to swing with a greater angular displacement. At the end of the visualization the pendulum reaches a state which is slightly offset from the perfect upright state. This state is shown in figure 3

1.4 Linearization of the System

We can linearlize the system about equilibrium points x_e, u_e. The state $x(t)$ can then be written as,

$$x(t) = x_e + \Delta x(t)$$

and the input signal $u(t)$ as,

$$u(t) = u_e + \Delta u(t)$$

The selection of x_e, u_e is such that $f(x_e, u_e) = 0$. Then,

$$\begin{aligned}\dot{x}(t) &= f(x(t), u(t)) = f(x_e + \Delta x(t), u_e + \Delta u(t)) \\ &= \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{pmatrix} \Delta x(t) + \begin{pmatrix} \frac{\partial f_1}{u} \\ \frac{\partial f_2}{u} \\ \frac{\partial f_3}{u} \\ \frac{\partial f_4}{u} \end{pmatrix} \Delta u(t)\end{aligned}$$

This Jacobian matrix was computed manually on *many sheets of paper* and then the given equilibrium point $(0, \pi, 0, 0)^T$ was substituted in to get the linearlized form,

$$\dot{x}(t) = \underbrace{\begin{pmatrix} 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1.0000 \\ 0 & -2.9430 & 0 & 0 \\ 0 & 25.5060 & 0 & 0 \end{pmatrix}}_{\text{A}} \Delta x(t) + \underbrace{\begin{pmatrix} 0 \\ 0 \\ 1 \\ 2 \end{pmatrix}}_{\text{B}} \Delta u(t)$$

I then used `MATLAB` to compute the eigenvalues of the A matrix, the eigenvalues are

$$\lambda_1 = 0,\ \lambda_2 = 0,\ \lambda_3 = 5.0503,\ \lambda_4 = -5.0503$$

1.5 Stabilizing static state feedback controller

The state feedback controller I designed is a simple static controller with

$$u = -Fx(t)$$

The gain values for F can be found in a number of ways, all we care about is placing the poles of the resultant controller in the open left half plane(OLHP). To keep it simple I choose my poles to be at -1, -2, -3, and -4.

I found the gain values for F using the `MATLAB` function `place` as shown below,

```
F = place(A,B,[-1,-2,-3,-4]);
```

Which gave me, $F = \begin{pmatrix} -0.7645 & 30.6353 & -1.5928 & 5.7964 \end{pmatrix}$

The implemented controller with these gain values gave a very reliable stabilization about the set equilibrium point. I also used Ackermann's formula to find the gain values which resulted in different values for F, those values however failed to simulate because they caused `ODE45` to take an extremely long time to solve the trajectory.

1.6 Implementing the controller

1.6.1 Implementation

I implemented the controllers into the non-linearlized system by slightly modifying the original `cartpendulum` system. I replaced the u terms with $-F\Delta x$ terms. To realize Δx I

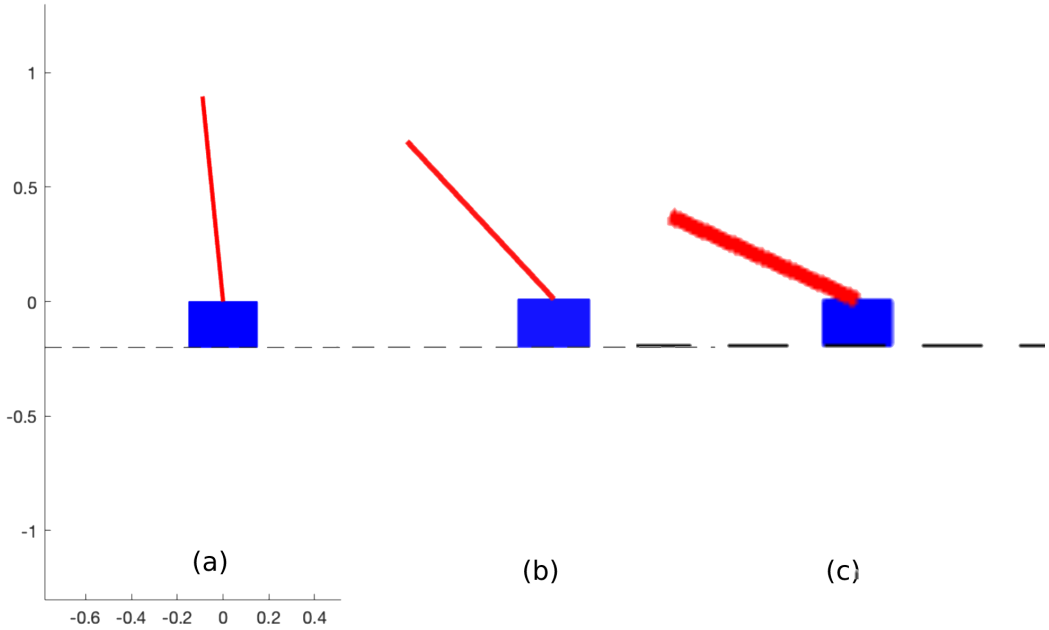


Figure 4: Initial conditions from which the static controller was able to stabilize the pendulum to an upright position. (a) 0.1 rad, (b) 0.75 rad, (c) 1.15 rad

simply subtracted the equilibrium point from the current state vector. My implementation is shown below.

```
eq_point = [0;pi;0;0];

cartpendulum_stablized = @(t,x) [x(3);
                        x(4);
                        (l*m2*sin(x(2))*x(4)^2 - (F*(x-eq_point)) + m2*g*cos(x(2))*
    sin(x(2)))/(m1+m2*(1 - cos(x(2))^2));
                        -1*(l*m2*cos(x(2))*sin(x(2))*x(4)^2 - (F*(x-eq_point))*cos(
    x(2))+ (m1 + m2)*g*sin(x(2)))/(l*m1 + l*m2*(1 - cos(x(2))^2))];

[t,x_traj] = ode45(cartpendulum_stablized,[0,15],[0;pi+0.1;0;0]);
```

1.6.2 Results

My implementation for the static stabilizing controller worked quite well. The chosen pole locations in the OLHP resulted in a controller that very quickly stabilized the upright pendulum from the initial location of 0.1 rad from the equilibrium point.

To test the limits of my controller I progressively increased the initial deviation without imparting any linear or angular velocity. My controller was reasonably successful in quickly returning to the upright position up to an initial angle of 0.75 angle (figure 4 (b)). After which it still worked but took some time to stabilize. The absolute maximum deviation from the equilibrium point was 1.15 rad, shown in figure 4 (c). Any increase after this did not result in a stable system.

1.6.3 Combined Stabilization

Just for fun I linked the two parts of this assignment to get the cart to start with the pendulum at an angle of 0 rad and get itself upright with the provided inputs and then hand over the control to the static controller to maintain the stabilization.

I did this by using the final x state after the inputs were executed as the initial state for the static controller simulation. I then combined the two trajectory vectors to use the provided animation script to animate the simulation.

The output for this and rest of the code can be found at my GitHub repository for this project here: https://github.com/Prince-John/final_project_ese441

2 Stabilizing Roll Dynamics of Aircraft

2.1 Part 1

I used the notes for observability to calculate the gains for the Output feedback controller but my calculations resulted in an unstable system. Experimentally, I found that the system behaved reasonably well with the following values,

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix};\ B = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 0 \\ 0 \end{pmatrix};\ D = 2$$

I kept the feed-forward gain at 1 because any other positive value resulted in very big overshoots, my calculated values for the feed-forward were obviously incorrect.

2.2 Bode Plots

The bode plot for this system was found using the following MATLAB script.

```
A = [0 1; 0 0];
B = [0;1];
C = [1 0];
D =0;

B_ = [0; 1];
C_ = [0 0]; D_ = 2;

sys_1 = ss(A, B, C, D);
sys_2 = ss(A, B_, C_, D_);

transfer_1 = tf(sys_1);
transfer_2 = tf(sys_2);

final_transfer = transfer_1/(1+transfer_1*transfer_2);
%final_transfer = transfer_1*transfer_2;
bode(final_transfer)
```

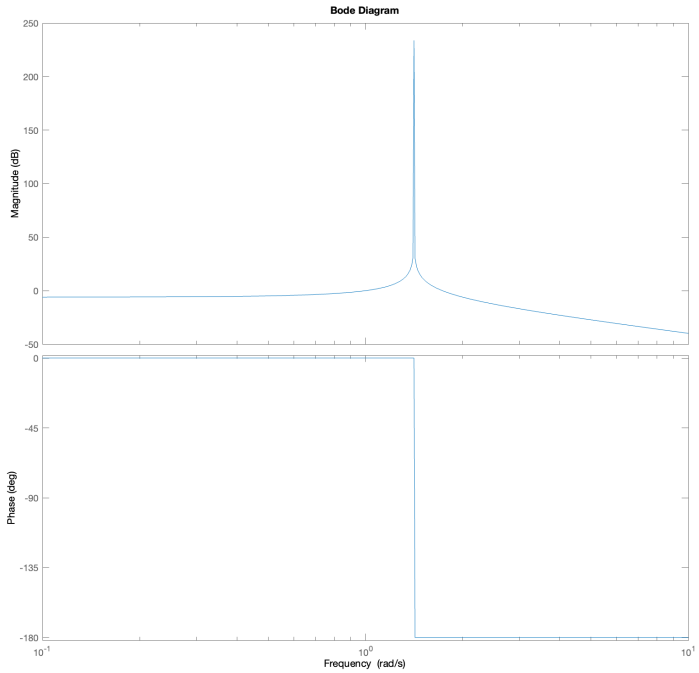


Figure 5: Bode plot for the Feedback controller

The bode plot in figure 5 shows that apart from the resonant peak this system behaves like a low-pass filter. This is somewhat expected because even though we have not modeled measurement noise, that is usually high frequency which will get filtered out by this system design. The inputs to this system are step inputs which are at a much lower frequency which is kept intact by this system.