Furuta Pendulum Project

By:

John Grenard

Travis Gubbins

MECA 482 Control System Design

Spring 2022

05/20/2022

California State University, Chico

Department of Mechanical and Mechatronic Engineering and Advanced Manufacturing

Chico, CA 95929

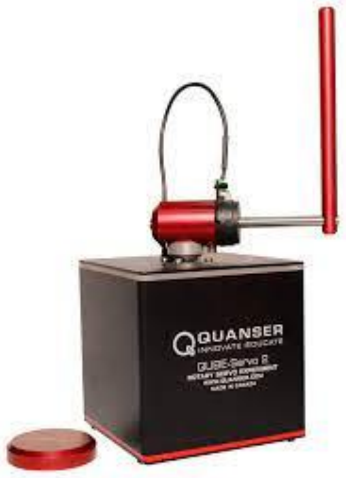

Figure 1. “Furuta Pendulum” – or rotational inverted pendulum by Quanser

Introduction:

The purpose of this project is to create a mathematical model of a Furuta Pendulum or rotational inverted pendulum, shown above in figure 1, which can be used along with a 3-D model and Simulink to simulate its response. A Furuta Pendulum is a system in where an inverted pendulum is held upright on a horizontal arm which is moved by the torque of a motor spinning the system about its vertical axis. Once the pendulum arm is brought up in the upright position the system must then stabilize it and hold it in this position. This is accomplished by a motor controlled by a programable controller with many know parameters. This parameters are important as the mathematics of the system are heavily dependent on the moment of inertia of the pendulum as well as the lengths of each arm and pendulum. Show below in figure 2 is an example of all of the parameters necessary in the modeling of the system. Where, θ_0 is the angular position of the spinning arm and θ_1 is the angular position of the pendulum measured with respect to the upright position. τ is the torque produced by the motor that is used to spin the system. I_0 is the moment of inertia of the spinning arm about its end and L_0 is the spinning arm’s length. m_1, L_1, l_1, and J_1 are the mass, length of arm, the center of mass location, and the moment of inertia of the pendulum, respectively. Lastly, g is the acceleration due to gravity.

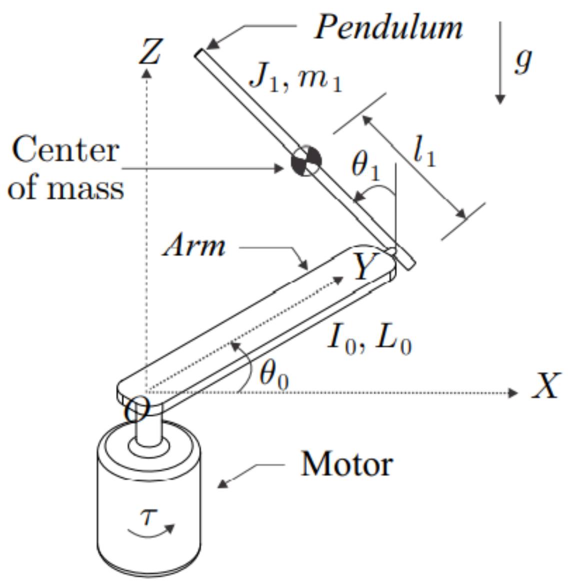

Figure 2. All known parameters of the Furuta Pendulum

This process of simulation is done in the real world because it is much cheaper to simulate a systems and its response than it is to build a real system. This is also done because the engineers are able to predict failures that might be unforeseen until the a real system is built, this as well also saves a great deal of money and time. In the following report you will be shown how the system was mathematical modeled and simulated. The MATLAB codes will be available in the appendix, as well as the functional viewpoint mapping used in the Simulink simulation. This report will also show you what type of hardware would be used and how the hardware would communicate with the controlling software.

Modeling:

The Lagrange equation of the system (L) is shown below in Eq.(1), where K is the kinetic energy and P potential energy of the pendulum arm,

$$L = K - P \tag{1}$$

Due to the systems two degrees of freedom the Lagrange equations of motion are,

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_0}\right) - \frac{\partial L}{\partial \theta_0} \tag{2}$$

$$0 = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}_1}\right) - \frac{\partial L}{\partial \theta_1} \tag{3}$$

The kinetic energy of the arm (K_0) in shown below in Eq.(4), and the kinetic energy of the pendulum arm (K_1) is shown below in Eq.(5), where v_1 is the linear velocity of the center of mass for the pendulum arm,

$$K_0 = \frac{1}{2} I_0 \dot{\theta}_0^2 \tag{4}$$

$$K_1 = \frac{1}{2} J_1 \dot{\theta}_1^2 + \frac{1}{2} m_1 v_1^T v_1 \tag{5}$$

Because this is being represented in 3-D space, we must determine the location of the center of mass for the pendulum arm in state space form by using Eq.(6) below,

$$x = \left[x_x, x_y, x_z\right]^T \tag{6}$$

The values of x_x, x_y, x_z are defined below,

$$x_x = L_0 \cos(\theta_0) - l_1 \sin\,(\theta_1) \sin\,(\theta_0)$$

$$x_y = L_0 \sin(\theta_0) + l_1 \sin\,(\theta_1) cos\,(\theta_0)$$

$$x_z = l_1 \cos\,(\theta_1)$$

The same analysis must be done regarding the velocity of the center of mass of the pendulum arm, shown below in Eq.(7),

$$v_1 = \left[\dot{x}_x, \dot{x}_y, \dot{x}_z\right]^T \tag{7}$$

The values of $\dot{x}_x, \dot{x}_y, \dot{x}_z$ are defined below, which are just the derivatives of x_x, x_y, x_z above,

$$\dot{x}_x = -\dot{\theta}_0 L_0 \sin(\theta_0) - l_1\left[\dot{\theta}_0 \sin(\theta_1)\cos(\theta_0) + \dot{\theta}_1 \sin(\theta_0)\cos(\theta_1)\right]$$

$$\dot{x}_y = \dot{\theta}_0 L_0 \cos(\theta_0) + l_1\left[\dot{\theta}_0 \cos(\theta_0)\cos(\theta_1) - \dot{\theta}_0 \sin(\theta_0)\sin(\theta_1)\right]$$

$$\dot{x}_z = -\dot{\theta}_0 l_1 \sin(\theta_1)$$

To find the real value of the kinetic energy of the pendulum arm (K_1), Eq.(7) must be plugged into Eq.(5) giving us,

$$K_1 = \frac{1}{2}J_1\dot{\theta}_1^2 + \frac{1}{2}m_1\left[\left(\dot{\theta}_0 L_0\right)^2 + \left(l_1\dot{\theta}_0 sin(\theta_1)\right)^2 + \left(l_1\dot{\theta}_1\right)^2 + 2\dot{\theta}_0\dot{\theta}_1 L_0 l_1 \cos(\theta_1)\right]$$

To find the kinetic energy of the Lagrange equation of the system (L) we use Eq.(8) below, which adds our new kinetic energy of the pendulum arm (K_1) with the already known kinetic energy of the arm (K_0),

$$K = K_0 + K_1 \tag{8}$$

The kinetic energy of the Lagrange equation of the system (L) is defined below,

$$K = \frac{1}{2}I_0\dot{\theta}_0^2 + \frac{1}{2}J_1\dot{\theta}_1^2 + \frac{1}{2}m_1\left[\left(\dot{\theta}_0 L_0\right)^2 + \left(l_1\dot{\theta}_0 sin(\theta_1)\right)^2 + \left(l_1\dot{\theta}_1\right)^2 + 2\dot{\theta}_0\dot{\theta}_1 L_0 l_1 \cos(\theta_1)\right]$$

Because the arm is only acting in the horizontal plane we only need to consider the potential energy of the pendulum arm, which is defined in Eq.(9) below,

$$P = -hm_1 g = m_1 g l_1(\cos(\theta_1) - 1) \tag{9}$$

Now that the kinetic energy (K) and potential energy (P) have been found, these values can be plugged into the Lagrange equation of the system. When the Lagrange equation of the system is plugged into the Lagrange equations of motion and derived we find the dynamics of the Furuta Pendulum, shown below in Eq.(10) & Eq.(11),

$$\tau = \alpha\ddot{\theta}_0 + \beta\dot{\theta}_0\dot{\theta}_1 + \gamma\ddot{\theta}_1 - \sigma\dot{\theta}_1^2 \tag{10}$$

$$0 = \gamma\ddot{\theta}_0 + (m_1 l_1^2 + J_1)\ddot{\theta}_1 - \tfrac{1}{2}\beta\dot{\theta}_0^2 - m_1 g l_1 \sin(\theta_1) \tag{11}$$

Where alpha, beta, gamma, and sigma are defined in Eqs.(12-15) respectively,

$$\alpha = I_0 + m_1 L_0^2 + m_1 l_1^2 sin^2(\theta_1) \quad (12)$$

$$\beta = m_1 l_1^2 \sin{(2\theta_1)} \quad (13)$$

$$\gamma = m_1 L_0 l_1 \cos(\theta_1) \quad (14)$$

$$\sigma = m_1 L_0 l_1 \sin(\theta_1) \quad (15)$$

Table 1. Assumed values for the pendulum system.

Parameter	**Value**	**Parameter**	**Value**
Spinning Arm Length (L_0)	6.6 (*cm*)	Spinning Arm Mass (M_0)	.380 (*kg*)
Pendulum Arm Length (L_1)	14.6 (*cm*)	Pendulum Arm Mass (M_1)	.054 (*kg*)
Center of Mass Location (l_1)	7.3 (*cm*)	Pendulum Inertia (J_1)	3.53x10^4 (*kg*-m^2)

Sensor Calibration:

The only calibration required for a Furuta Pendulum is setting the angle of zero degrees in the stepper motor. This calibration though, is not very necessary for the vertical axis motor, it is only done if you want it to start from a specific position. So to do this calibration there are a few different methods that could be used, the sensor method, the dedicated motor drive chip with stall detection method, and the stop block method. The sensor method requires a rotation position sensor to be installed on the system at the zero degree position. When the spinning arm gets to the desired position the sensor will tell the controller and the driver that it has gotten to its desired position. This method is reliable and accurate but requires extra pieces to be purchased and installed. The dedicated motor drive chip with stall detection method is a external chip the install to the motor driver that detects when the motor has stopped moving. So you are required to block the spinning arm from moving at its desired position and it will determine its zero degrees position. This method requires a certain stepper motor current and has bad versatility. The stop block method just requires that put a block at the desired position and the motor need to be run into it from a larger angle. When the motor hits the block and stops it finds the zero degree position. This method is very simple and requires no extra pieces, but it does do harm to the motor as the motor will try to power though the block. Hoping that the calibration only needs to be preform one time, the stop block method is the best option for calibration of the Furuta Pendulum.

Controller Design and Simulation:

The controller for the Furuta Pendulum must be one that is linear, this is why all the MATLAB code has been linearized. When the system gets an initial input value telling the controller where the pendulum arm is it then runs the initial value through the mathematical model equations. This equations will then give and output value for the torque and the position sensor will be feedback to the summing junction through the full state feedback loop. Then the new input will go through the controller and will send the motor driver a new torque value to use. This process is repeated until the pendulum arm is in the upright position.

Appendix A

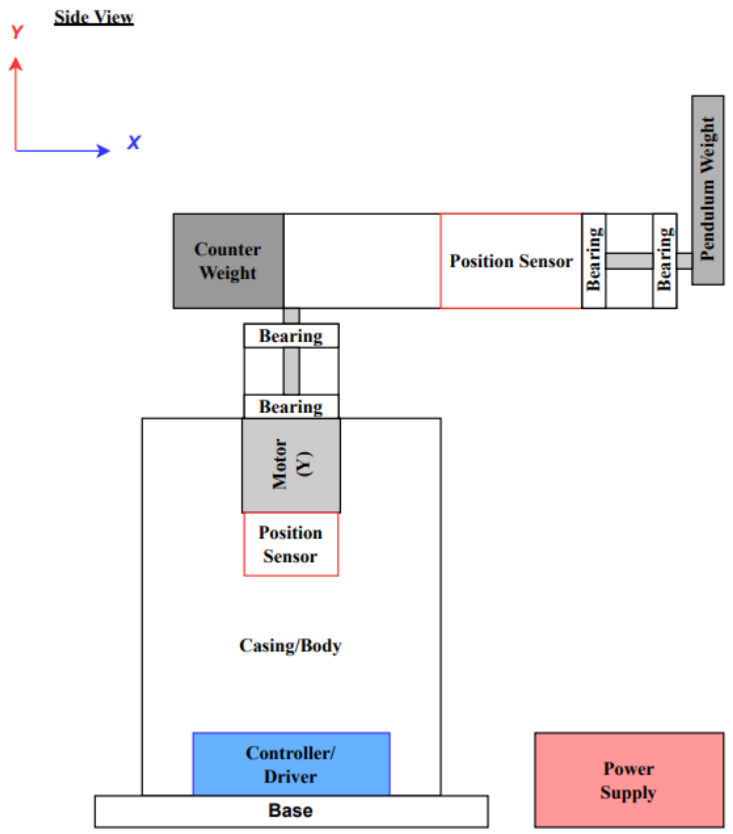

Figure 1. Side View of the Operational Viewpoint

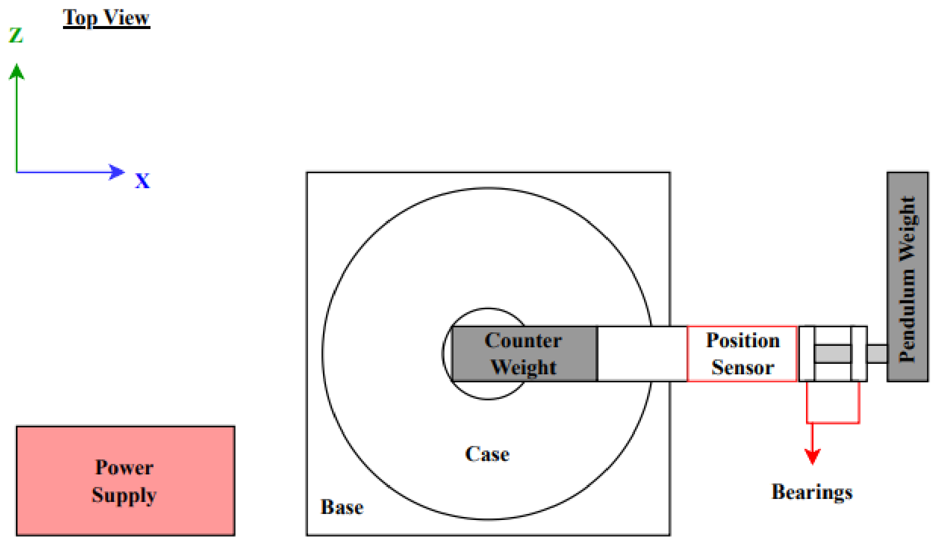

Figure 2. Top View of the Operational Viewpoint

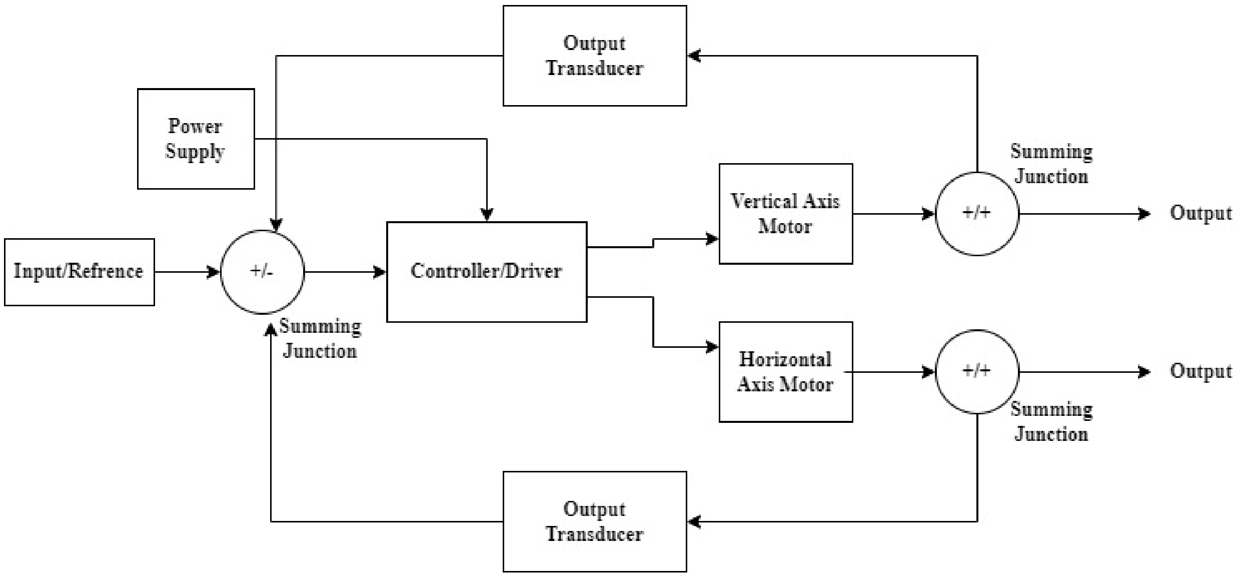

Figure 3. Functional Viewpoint of Furuta Pendulum

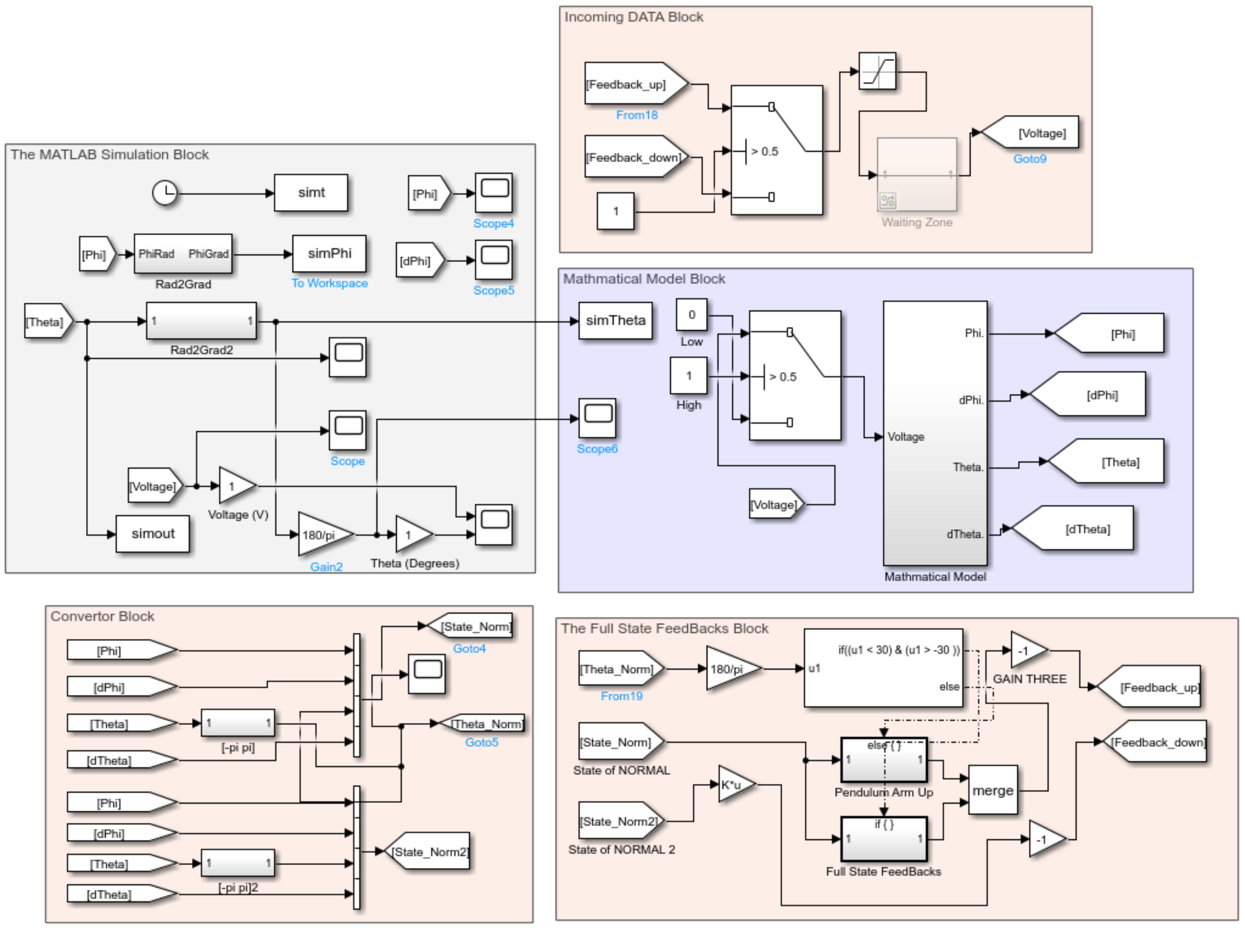

Figure 4. Snapshot of the Simulink

MATLAB Code for the Constants:

```
clear
clc
%% Define the Parameters of The System
g = 9.81;       % (Meter/Second^2)
m1 = 0.380;     % (Kilograms)
m2 = 0.054;     % (Kilograms)
L1 = 0.066;     % (Meters)
L2 = 0.146;     % (Meters)
J = 3.5256e-4;  % (Kilogram-Meter^2)
Re = 14.5;      % Potential energy
ke = 0.5;       % Kinetic Energy
M = 0.044;      % Sin/Cos Values
kb_p = 4.7940e-04 ;
kb_m = 6.75e-4 ;
%% Equations for Lagrange Equation of the system
alpha = J + (M + m1/3 + m2) * L1^2;
beta = (M + m2/3) * L2^2;
gamma = (M + m2/2) * L2*L1;
sigma = (M + m2/2) * g * L2;
%% Define the Parameters of The Simulation
Zn = 3;
Ts = .001;
StepX = 10;
dtDisc = 0.01;
initial_state = pi;
distrub = 12;
disturb = distrub*pi/180;
Reference = [0 0 0 0];
%% Making the Linearization of The System
% Making Matrix A
A = zeros(4,4);
A(1,2) = 1;
A(2,3) = -(sigma * gamma)/(alpha * beta - gamma^2);
A(3,4) = 1;
A(4,3) = (alpha * sigma)/(alpha * beta - gamma^2);
% Making Matrix B
B = zeros(4,2);
B(2,1) = beta/(alpha * beta - gamma^2);
B(2,2) = -gamma/(alpha * beta - gamma^2);
B(4,1) = -gamma/(alpha * beta - gamma^2);
B(4,2) = alpha/(alpha * beta - gamma^2);
% C matrix
C = [0 0 1 0; 0 0 0 1];
%% Linear of The System Matrixs
% Making New Matrix A
Ap = zeros(4,4);
Ap(1,2) = 1;
Ap(2,1) = 0;      Ap(2,2) = -B(2,1) * (ke^2/Re + kb_m);
Ap(2,3) = A(2,3); Ap(2,4) = -B(2,2) * kb_p;
Ap(3,4) = 1;
Ap(4,1) = 0;      Ap(4,2) = -B(4,1) * (ke^2/Re + kb_m);
Ap(4,3) = A(4,3); Ap(4,4) = -B(4,2) * kb_p;
```

```
% Making New Matrix B
Bp = zeros(4,1);
Bp(2) = B(2,1) * ke/Re;
Bp(4) = B(4,1) * ke/Re;
%% State Feedback Control
K = place(Ap,Bp,[-5 -4 -2+2j -2-2j]);
Q = [0.1 0 0 0; 0 0.01 0 0; 0 0 100 0; 0 0 0 10];
R = 10;
[K, ~, E] = lqr(Ap,Bp,Q,R);
% Making Controls & Observability
Control = rank(ctrb(Ap,Bp));
Observ = rank(obsv(Ap,C));
%% Pendulum Arm in The Swing Down Position
% Swing Down Matrix A
Ap2 = zeros(4,4);
Ap2(1,2) = 1;
Ap2(2,1) = 0;      Ap2(2,2) = -B(2,1) * (ke^2/Re + kb_m);
Ap2(2,3) = A(2,3); Ap2(2,4) = B(2,2) * kb_p;
Ap2(3,4) = 1;
Ap2(4,1) = 0;      Ap2(4,2) = B(4,1) * (ke^2/Re + kb_m);
Ap2(4,3) = -A(4,3); Ap2(4,4) = -B(4,2) * kb_p;
% Swing Down Matrix B
Bp2 = zeros(4,1);
Bp2(2) = B(2,1) * ke/Re;
Bp2(4) = -B(4,1) * ke/Re;
K2 = place(Ap2,Bp2,[-5 -4 -2+2j -2-2j]);
R2 = 1;
Q2=[1 0 0 0; 0 10 0 0; 0 0 1000 0; 0 0 0 10];
[K2, ~, E] = lqr(Ap2,Bp2,Q2,R2);
```

MATLAB Code for the Simulation:

```
clc
%% Create the 3-D Simulation Space
view(135,20)
AL = .2;
grid on
%% Redefine the Parameters
L1 = 0.066; %(Meters)
L2 = 0.146; %(Meters)
s = 8;
theta = 0;
phi = 0;
c = [0 0 0];
Xh = [0; L1]';
Yh = [0; 0]';
Zh = [0; 0]';
Xv = [Xh(2); L1]';
Yv = [Yh(2); 0]';
Zv = [Zh(2); -L2]';
hold on
Harm  = fill3(Xh,Yh,Zh,'b');
Varm  = fill3(Xv,Yv,Zv,'g');
M = scatter3(Xv(2),Yv(2),Zv(2),s);
axis([-AL AL -AL AL -AL AL]);
```

```
TXT = title('Time: ');
for t = 1:17:size(simTheta,1)
    TXT2 = sprintf('Time:%.2f',simt(t));
    set(TXT,'String',TXT2);
    phi = simPhi(t);
    theta = -simTheta(t);
    Xva = 0;
    Yva = L2 * sin(theta);
    Zva = -L2 * cos(theta);
    Xvb = Xva * cos(phi) - Yva * sin(phi) + L1 * cos(phi);
    Yvb = Xva * sin(phi) + Yva * cos(phi) + L1 * sin(phi);
    Zvb = Zva;
    Xh(2) = L1 * cos(phi);
    Yh(2) = L1 * sin(phi);
    Xv = [Xh(2); Xvb]';
    Yv = [Yh(2); Yvb]';
    Zv = [0; Zvb]';
    set(Harm,'XData',Xh);
    set(Harm,'YData',Yh);
    set(Harm,'ZData',Zh);
    set(Varm,'XData',Xv);
    set(Varm,'YData',Yv);
    set(Varm,'ZData',Zv);
    rem(t,30)
    set(M,'XData',Xv(2));
    set(M,'YData',Yv(2));
    set(M,'ZData',Zv(2));
    drawnow;
end
```

References:

[1] Norman S. Nise - Control Systems Engineering-Wiley (2015) 7th Edition

[2] Lecture and reference videos from H. Sinan Bank's Blackboard page

[3] Wikipedia, Furuta Pendulum, Retrieved by Apr., 19, 2022 from https://en.wikipedia.org/wiki/Furuta_pendulum

[4] IEEEXPLORE, Modeling, Simulation, and Construction of a Furuta Pendulum Test-Bed, Retrieved by Apr., 25, 2022 from https://ieeexplore.ieee.org/document/7086928

[5] Control System Tutorials for MATLAB and Simulink, Retrieved by Apr, 22, 2022 from http://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling

[6] GitHub Docs, Creating and highlighting code blocks, Retrieved by May, 02, 2022 from https://docs.github.com/en/get-started/writing-on-github

[7] Quanser, QUBE – Servo 2, Retrieved by May, 02, 2022 from https://www.quanser.com/products/qube-servo-2/

[8] Screencast-O-Matic, Sites used on May, 19, 2022 from https://screencast-o-matic.com/