Furuta Pendulum Project

MECA 482 - Fall 2021

Team members:

Riczi Cano
Greta Fischer
Flor Contreras
Timothy Grice

Department of Mechanical and Mechatronic Engineering
and Advanced Manufacturing
California State University, Chico
Chico, CA 95929-0789

Table of Contents:

1.Introduction

Deliverables:

- Build a webpage using GitHub
- Mathematical model of the system using MATLAB
- Control system should be provided using a high-level programming language. Must show that the control algorithm will give the requirements for the system.
- Have a simulation with the control system and mathematical model by connecting CoppeliaSim to MATLAB
- The design hardware should have explanations such as hardware and software relationships.

The purpose of this project is to analyze the control system used for the Furuta Pendulum. The Furuta Pendulum was studied, modeled and simulated in this project. The pendulum is a nonlinear system and it is difficult for it to stay upright. It is used in control systems to highlight different techniques. The pendulum consists of an arm that rotates in the horizontal plane and a pendulum is attached to it that can rotate in the vertical plane. The goal of this project is to have

the arm in the horizontal plate balance vertically by controlling the motors and the required force to do so. The project will be done virtually, using MATLAB and CoppeliaSim.

2. Modeling

The dynamics of the Furuta Pendulum will utilize Simulink. The program will allow the computation of dynamic properties as well as visual representation of the working system. The code used in Simulink will be from Matlab. The torque applied to the pendulum via a DC motor will be an important aspect of the pendulum.

Table 1: Nomenclature

x	Position of arm
Θ	Angle of pendulum
F	Applied force
m	Mass of pendulum
l	Length of pendulum
g	gravity
fo	Friction coefficient

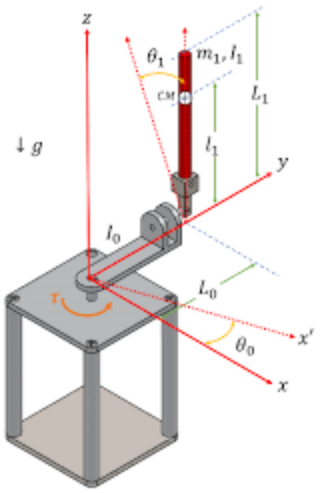


Figure 2: Example of a Furuta Pendulum

In the figure above the two angles show the degrees of freedom for the pendulum. The motor helps to guide the angles to be positive and keep it upright.

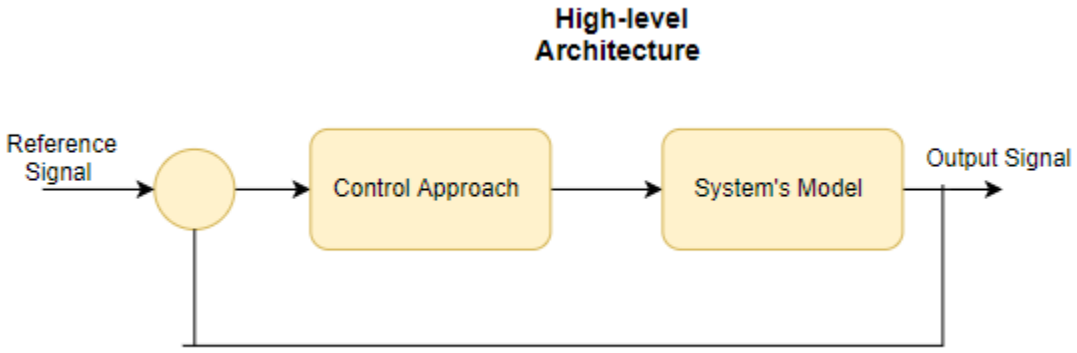


Figure 3: High level architecture for the pendulum.

The equations below are the basic transfer functions. These functions would help to determine the system poles and zeros. They model the output for the possible inputs of the system.

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \ldots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \ldots + a_1 s + a_0}$$

$$H(s) = \frac{N(s)}{D(s)} = K \frac{(s - z_1)(s - z_2) \ldots (s - z_{m-1})(s - z_m)}{(s - p_1)(s - p_2) \ldots (s - p_{n-1})(s - p_n)},$$

We can represent the linear motion functions in state-space form. They can be put into the matrices that are shown below. The g in the matrices is gravity and the R and the km are the resistance and torque constants.

$$A = \begin{bmatrix} 0 & 1 & 0 \\ \bar{d}_{11}\bar{m}g & 0 & 0 \\ \bar{d}_{21}\bar{m}g & 0 & 0 \end{bmatrix}$$

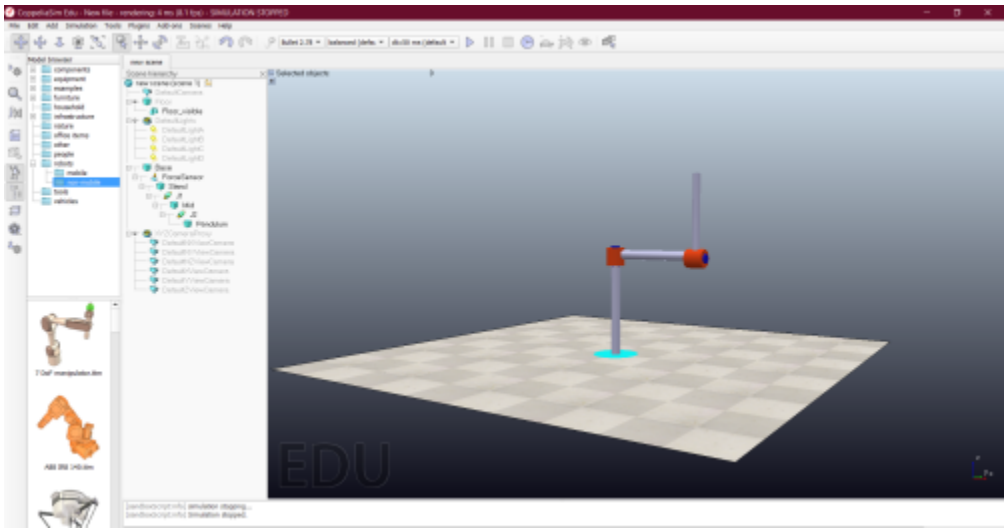
$$\mathcal{B} = \begin{bmatrix} 0 \\ \bar{d}_{12} \\ \bar{d}_{22} \end{bmatrix} \frac{k_m}{R}.$$

$$D = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \begin{bmatrix} m_1 l_{c1}^2 + m_2 l_1^2 + I_1 + I_2 & I_2 \\ I_2 & I_2 \end{bmatrix}$$

3. Sensor Calibration

No sensor calibration is needed due to this project being completely virtual.

4. Controller Design Simulation



5. Simulation Code

```
% Initial Conditions:
Jp=.5;Mp=6;Lr=12;Jr=.6;Lp=15;g=9.81;Br=10;Bp=15;Kg=16;kt=2;km=3;Rm=.3;
Jt=Jp*Mp*Lr^2+Jr*Jp+(1/4)*Jr*Mp*Lp^2;
% State Space Representation

A = [0 0 1 0;0 0 0 1;0 (1/(4*Jt))*Mp^2*Lp^2*Lr*g (-1/Jt)*Br*(Jp+1/4*Mp*Lp^2)
(-1/(2*Jt))*Bp*Mp*Lp*Lr;0 .5*Mp*Lp*g*(Jr+Mp*Lr^2)/Jt (1/(2*Jt))*Mp*Lp*Lr*Br
(-1/Jt)*Bp*(Jr+Mp*Lp^2)]; % Add here linearized system model instead of zero
B = [0;0;(1/Jt)*(Jp+.25*Mp*Lp^2);(1/(2*Jt))*Mp*Lp*Lr]; % Add here input vector instead of
zero
C = eye(2, 4);   % Shows we have two outputs
D = zeros(2,1);

% Add actuator dynamics (if you would consider to add)
A(3,3) = A(3,3)-Kg^2*kt*km/Rm*B(3); % Add here actuator dynamics instead of zero
A(4,3) = A(4,4)-Kg^2*kt*km/Rm*B(4); % Add here actuator dynamics instead of zero
B = Kg * kt * B / Rm;

poles = eig(A);
% Load into state-space system
sys_FURPEN_ol = ss(A,B,C,D) % Open loop system model
```

```
function K = control_FURPEN(A,B,zeta,wn,d_p3,d_p4)

   % Location of dominant poles along real-axis
   sigma = zeta * wn;
   % Location of dominant poles along img axis (damped natural freq
wn*(1-zeta)^0.5)
   wd = wn*(1-zeta)^0.5
   % Desired poles (-30 and -40 are given)
   poles = [-sigma+j*wd, -sigma-j*wd, d_p3, d_p4];
   % Find control gain using MATLAB pole-placement command (acker or
place)
   K = acker(A, B, poles)

end
```

6. Conclusion

The group was able to get the Matlab code and to create a model of the Furuta Pendulum on Coppelia. We were unable to connect the Coppelia model with our Matlab code.

7. References

1. Quanser, Rotary Inverted Pendulum, https://www.quanser.com/products/rotary-inverted-pendulum/
2. M. W. Spong, P. Corke, and R. Lozano, Nonlinear control of the reaction wheel pendulum,Automatica, vol. 37, no. 11, pp. 1845–1851, 2001
3. https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling