Simulation 2: Inverted Pendulum

Brandon Reddish
MAE 271

This lab consists of modeling the system of an inverted pendulum that is being vibrated vertically. In addition to simulating the model in MATLAB, a set of frequencies and magnitudes will be determined that will stabilize the system such that the pendulum oscillates back and forth in an upright position. The figure shows the diagram that will be modeled.

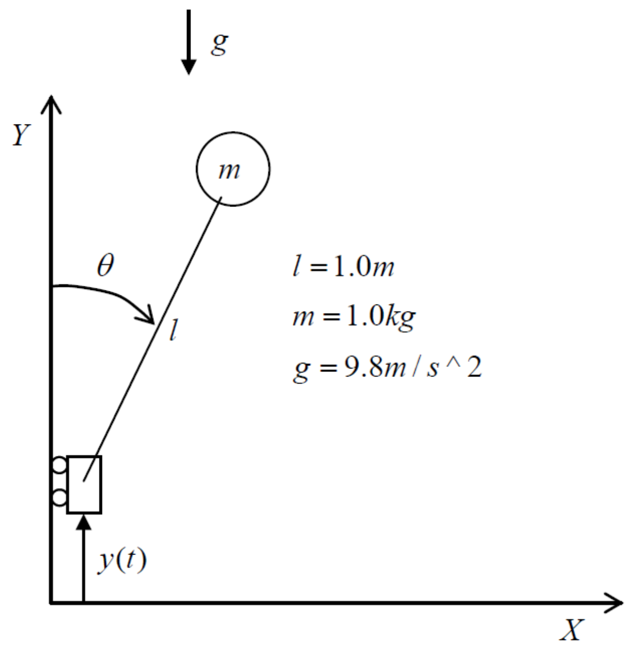


Figure 1 – Inverted Pendulum System

Foe this system, there is a gravity force acting in the negative Y direction. The base of the rod will be vibrated with the following sinusoidal function.

$$y(t) = Ysin(\omega t)$$

By breaking down the system into parts, it can be simulated with the following bond graph.

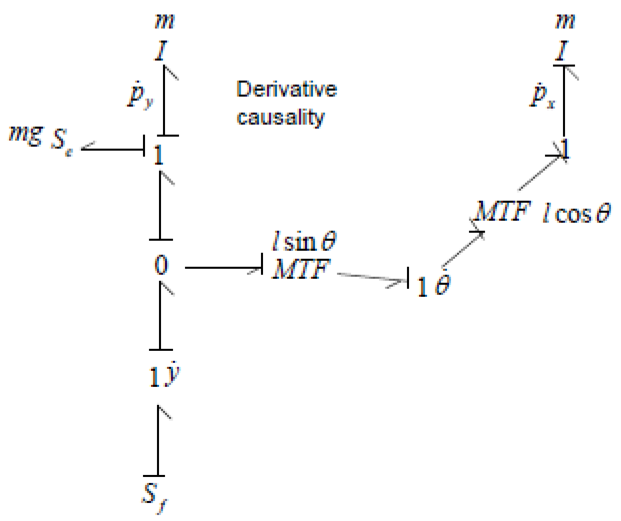


Figure 2 – Inverted Pendulum Bond Graph

Using this bond graph and kinematic relations, the following equations will be used to get the rate of change of momentum in the X direction and theta.

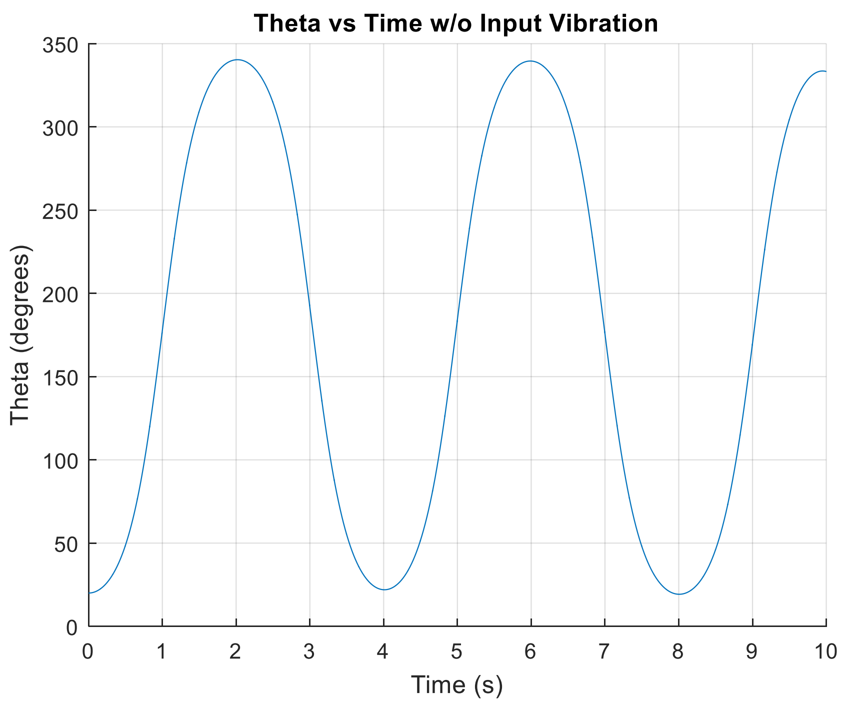
$$\dot{p}_x = mg\sin\theta\cos\theta + m\,\ddot{y}\sin\theta\cos\theta - \frac{\sin\theta}{\cos\theta}\dot{\theta}\,p_x$$

$$\dot{\theta} = \frac{1}{l\cos\theta}\frac{p_x}{m}$$

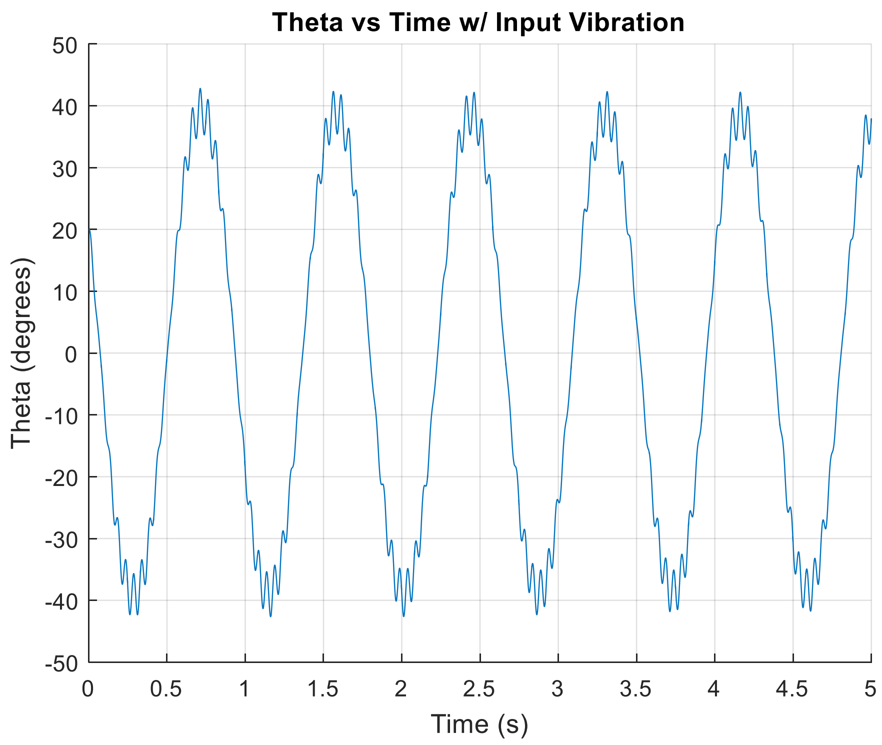
Differentiating the position equation for the base of the rod twice results in the following equation.

$$\ddot{y} = -A\omega^2 sin\omega t$$

With these three equations and the ode solver in MATLAB, the simulation can now be run. With an amplitude of zero (no oscillation) and a slight offset of the pendulum (20 degrees) the pendulum will swing to the bottom and up to -20(+340) degrees and back for the entirety of the simulation. The system has no damping, so the pendulum will never settle on the bottom. The following plot shows the value of theta with no damping or input oscillation.



When an arbitrary input frequency of 20Hz and an amplitude of 0.1m the solution is as follows.



With this input, the system exhibits an interesting behavior. The pendulum rights itself, but continues to swing to the left to about -40 deg and then back to 40 deg. The goal now is to find a frequency and amplitude that will stabilize a pendulum with a 10, 20 and 30 degree initial offset.

Using the approximate solution developed by J P DenHartog provides some guidance when trying to find a good frequency and amplitude combination. The following figure plots the stable regions for the pendulum.

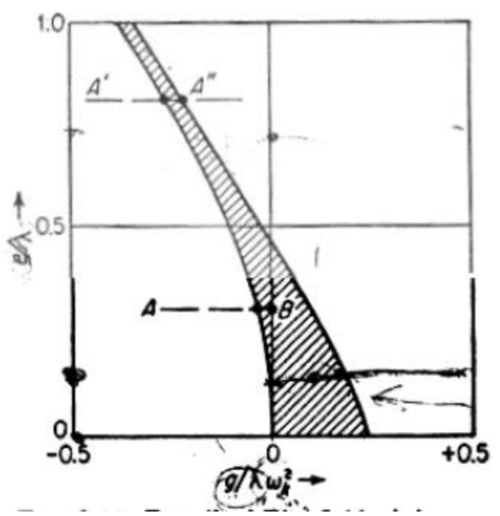


Figure 3 – Approximate Stable Regions

Based on this plot, the values of e will be less than 0.5 so that for values of the x-axis that are close to zero will be stable. Using an amplitude of 0.2 and an angular frequency of 90 radians per second. For this combination of input parameters, the system will keep the pendulum upright against the force of gravity.

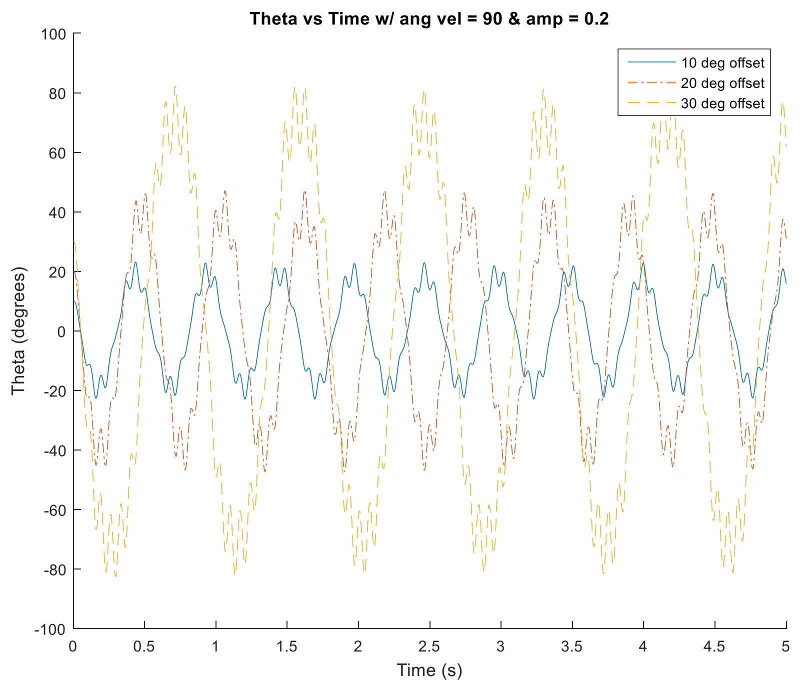
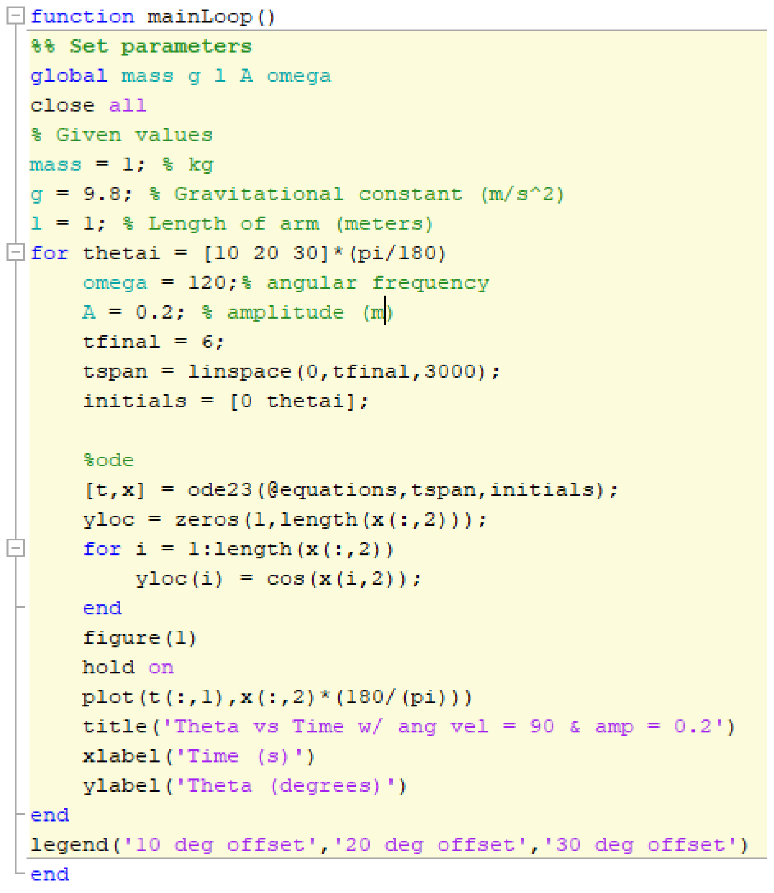


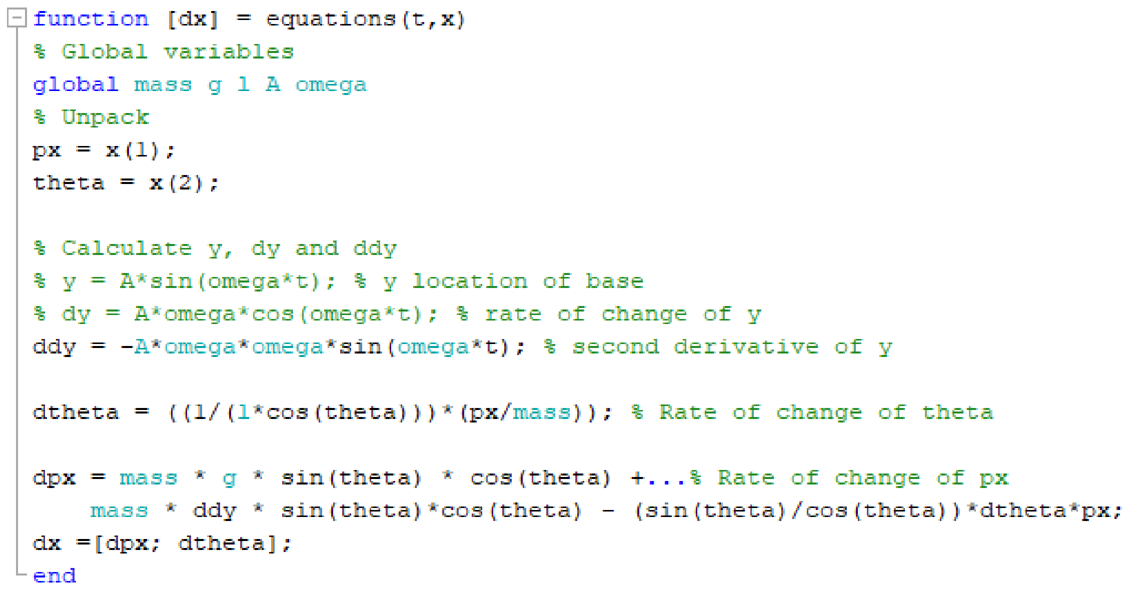
Figure 4 – Stabilized Pendulum with Three Different Initial Conditions

Appendix A – MATLAB Code for Inverted Pendulum



```
function mainLoop()
%% Set parameters
global mass g l A omega
close all
% Given values
mass = 1; % kg
g = 9.8; % Gravitational constant (m/s^2)
l = 1; % Length of arm (meters)
for thetai = [10 20 30]*(pi/180)
    omega = 120;% angular frequency
    A = 0.2; % amplitude (m)
    tfinal = 6;
    tspan = linspace(0,tfinal,3000);
    initials = [0 thetai];

    %ode
    [t,x] = ode23(@equations,tspan,initials);
    yloc = zeros(1,length(x(:,2)));
    for i = 1:length(x(:,2))
        yloc(i) = cos(x(i,2));
    end
    figure(1)
    hold on
    plot(t(:,1),x(:,2)*(180/(pi)))
    title('Theta vs Time w/ ang vel = 90 & amp = 0.2')
    xlabel('Time (s)')
    ylabel('Theta (degrees)')
end
legend('10 deg offset','20 deg offset','30 deg offset')
end
```



```
function [dx] = equations(t,x)
% Global variables
global mass g l A omega
% Unpack
px = x(1);
theta = x(2);

% Calculate y, dy and ddy
% y = A*sin(omega*t); % y location of base
% dy = A*omega*cos(omega*t); % rate of change of y
ddy = -A*omega*omega*sin(omega*t); % second derivative of y

dtheta = ((1/(l*cos(theta)))*(px/mass)); % Rate of change of theta

dpx = mass * g * sin(theta) * cos(theta) +...% Rate of change of px
    mass * ddy * sin(theta)*cos(theta) - (sin(theta)/cos(theta))*dtheta*px;
dx =[dpx; dtheta];
end
```