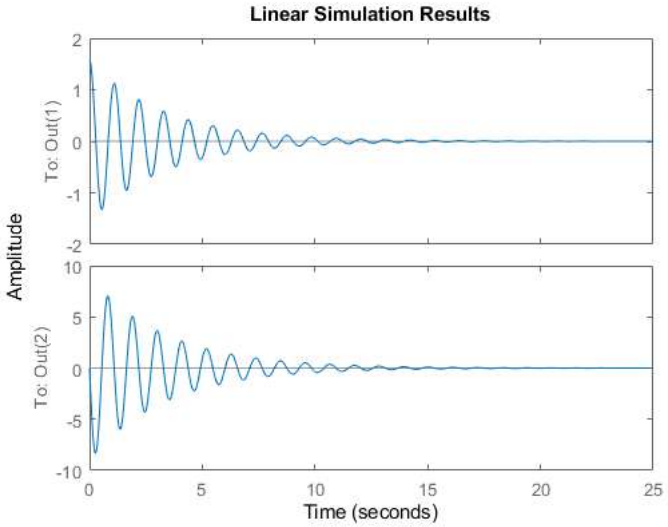
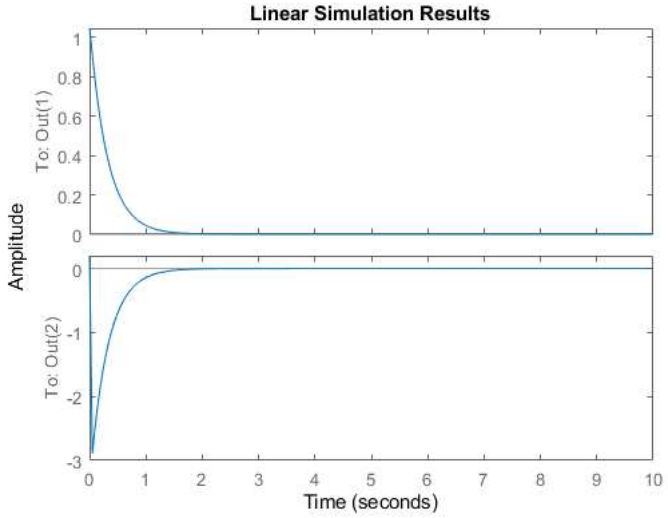
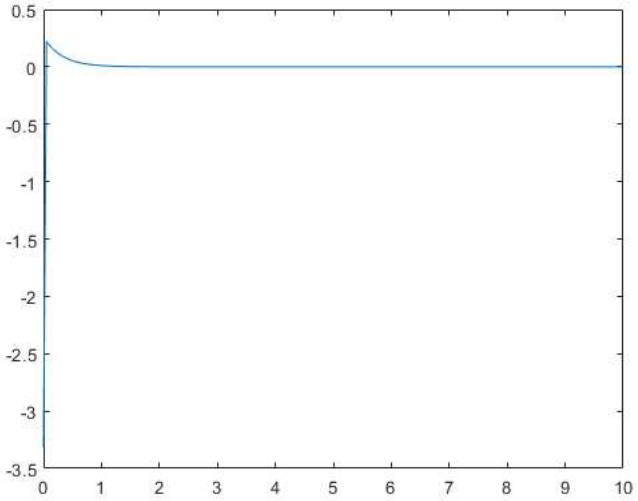
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Info
% Author: Taylor Smith, Christian Williams
% Version: 1.0
% Created: 5/3/2023
%
% Description
%   Script to model the hanging down of a pendulum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
close all;
%pkg load control
%pkg load signal

% Measured Parameters
mass_of_copter = 0.0132;
mass_of_rod = 0.0234;
length_of_rod = 0.5021;
length_of_rod_to_pivot = 0.29;
radius_of_copter = 0.01;

% Damping Coefficient
b = 0.60;

% Constants and Basic Calculated Parameters
g = 9.81;
density_of_rod = mass_of_rod/length_of_rod;
length_of_back_rod = length_of_rod - length_of_rod_to_pivot;
mass_of_rod_to_pivot = density_of_rod * length_of_rod_to_pivot;
total_mass_of_rod = density_of_rod * length_of_rod;
mass_of_back_rod = density_of_rod * length_of_back_rod;

% Moment of Inertia
inertia = 1/3*mass_of_rod_to_pivot*length_of_rod_to_pivot^2 + 1/3*mass_of_back_rod*length_of_back_rod^2 + 2/5 * mass_of_copter * radius_of_copter^2 + mass_of_copter

% State Space Model
A = [[0,1];[-(g*length_of_rod_to_pivot*(mass_of_copter+mass_of_rod_to_pivot/2))/inertia,-b]];
B = [[0];[length_of_rod_to_pivot/inertia]];
C = eye(2);
D = [0];

% Create State Space Model
model = ss(A,B,C,D);

% Simulate results
t=0:0.05:25;
forcing = zeros(size(t));
angle = pi/2;
omega = 0;
figure('Name','Angle vs Time');
lsim(model,forcing,t,[angle;omega]);

% Display Eigenvectors and Eigenvalues
disp('Eigenvectors and Eigenvalues:')
[eigenvectors, eigenvalues] = eig(model.a)

% Observability
observability = obsv(A,C);
rank_of_observability = rank(observability)

% Controlability
controlabilty = ctrb(A,B);
rank_of_controlability = rank(controlabilty)

% Determine Desired Poles Using LQR
Q = [10 0;0 1];
R = [1];
Gain = lqr(A,B,Q,R)

% Create Model
Ac = A-B*Gain;
Bc = [0;0];
Cc = C;
Dc = 0;
controlled_model = ss(Ac,Bc,Cc,Dc);

% Simulate Results
t = 0:0.05:10;
forcing = zeros(size(t));
angle = pi/3;
omega = 0.2;
figure();
lsim(controlled_model, forcing, t, [angle;omega]);
[Y,T,X] = lsim(controlled_model, forcing, t, [angle;omega]);

% Display Eigenvector and Eigenvalues
disp('Eigenvectors and eigenvalues:')
[eigenvectors,eigenvalues] = eig(controlled_model.a)
```

```
% Display the Results
figure();
plot(t,-Gain*Y');
```

```
Eigenvectors and Eigenvalues:

eigenvectors =

  -0.0089 - 0.1710i  -0.0089 + 0.1710i
   0.9852 + 0.0000i   0.9852 + 0.0000i


eigenvalues =

  -0.3000 + 5.7449i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.3000 - 5.7449i


rank_of_observability =

     2


rank_of_controlability =

     2


Gain =

    2.9725    1.0139

Eigenvectors and eigenvalues:

eigenvectors =

    0.3006   -0.0059
   -0.9537    1.0000


eigenvalues =

   -3.1725         0
         0 -168.8093
```







Published with MATLAB® R2022a