Subject: Segway Control Term Project – Part 1
Student: Howard Chiang
Class: ME5659 – Control and Mechatronics
Professor: Rifat Sipahi
Date: 4/24/2015

Introduction

Below are the chosen values for constants that are to be used in this report.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Variable Initialization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% theta;            % ~ % Platform deviation from vertical
% x;                % ~ % Horizontal displacement of Segway vehicle
% V;                % ~ % Motor voltage
% T;                % ~ % Motor torque
% d;                % ~ % Horizontal displacement of rider load/mass (m)

R = 0.5;            % C % Wheel radius (m)
L = 0.4;            % C % Distance from wheel centerline to center of mass (m)
l = 1;              % C % Distance from wheel centerline to rider load/mass (m)
M = 27;             % C % System (vehicle) mass (~120 lbs)
m = 33.72;          % C % Rider (load) mass(N) (~150 lbs)
k_t = 0.75;         % C % Torque constant (N*m/A)
k_bemf = 0.5;       % C % Back emf constant (V*sec)
r_a = 1.4;          % C % Armature resistance
c1 = 0.01;          % C % Small rotational damping with appropriate units
c2 = 0.01;          % C % Small linear damping with appropriate units
g = 9.81;           % C % Gravitational acceleration (m*s^2)
J = M*L^2+m*l^2;            % C % Moment of inertia related to all the mass rotating
around the wheels (kg*m^2)(calculated)
alpha = k_t/r_a;            % C % Motor constant (calculated)
beta = k_t*k_bemf/(R*r_a);  % C % Motor constant (calculated)
```

Figure 1 – Variables, constants, and Equations

Task 1

This tasks focuses on finding a lead compensator that would stabilize the Segway system. The poles and roots of the open loop, closed loop, with & without disturbance transfer functions are analyzed. The poles and zeroes were found using the 'pole' and 'zero' function in Matlab.

System Type	Poles	Zeroes
Open Loop w/o Disturbance	3.3905, -3.3907, -0.0088	-8.1301e-05
Closed Loop w/o Disturbance	3.3915, 3.3905, -3.3917, -3.3907 -0.0088, -0.0088	3.3905, -3.3907, -0.0088, -0.0001
Open Loop w/ Disturbance	3.3905, 3.3905, -3.3907, -3.3907, -0.0088	3.3905, -3.3907, -0.0088
Closed Loop w/ Disturbance	3.3905, 3.3905, 3.3905, 1.6784, -3.3907, -3.3907, -3.3906, -1.6785 -0.0088, -0.0088	3.3905, 3.3905, 3.3905, -3.3907, -3.3907, -3.3906, -0.0088, -0.0088

Table 1 – Poles & Zeroes

Many iterations of the root locus plot was attempted for each type of system. However, a stable system could not be found. There were cases where one locus required the gain to be low to remain on the left hand side of the imaginary axis, while another locus would require a high gain, thus neither conditions can be settled.

The closest the system to stability was a closed loop system without including disturbance in the system. The lead compensator values that were chosen were:

Poles	Zeroes
-10	1, 20

Table 2 – Poles & Zeroes of Lead Compensator

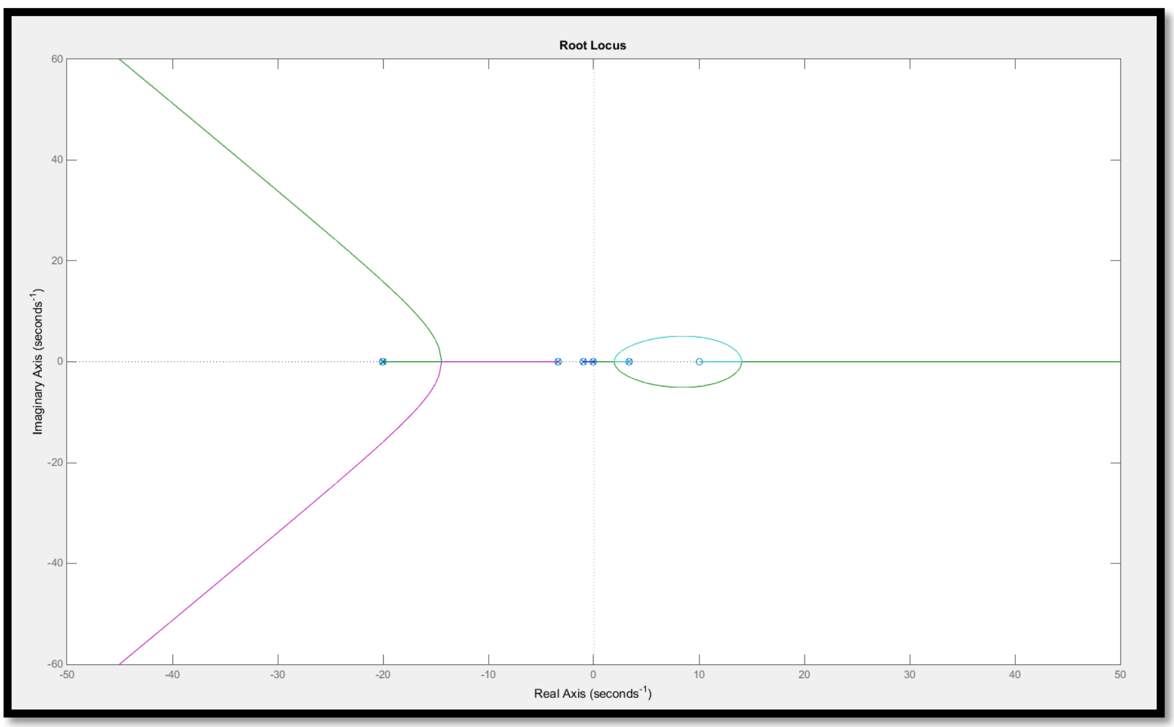

Figure 2 – Root Locus with Lead Compensator

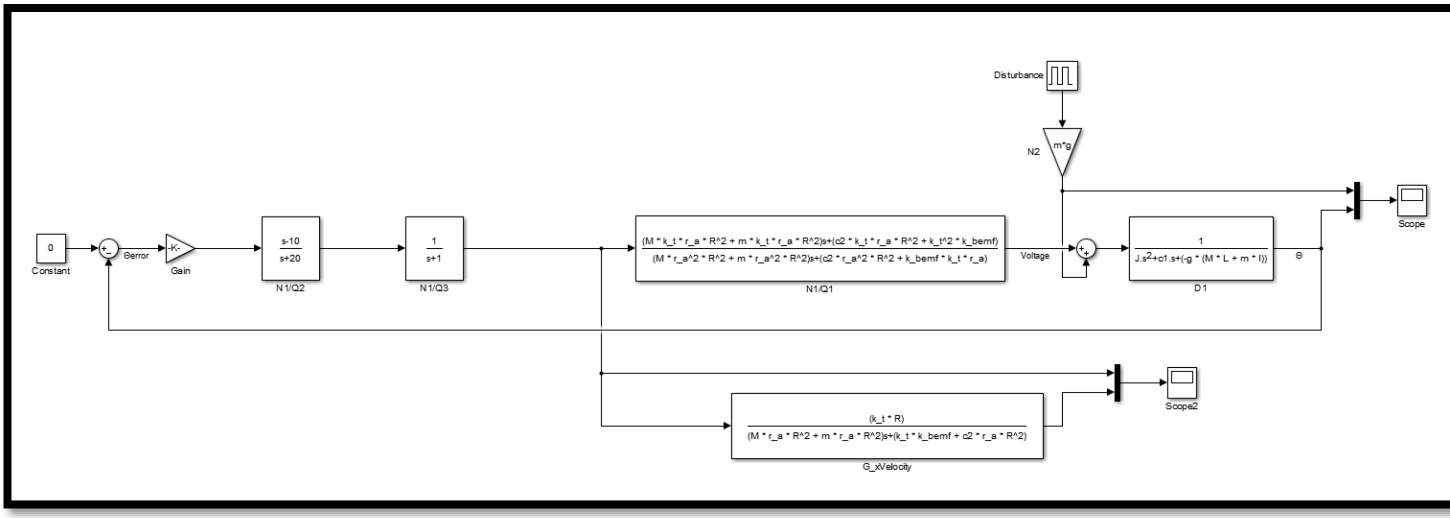

Figure 3 – System Model with Lead Compensator

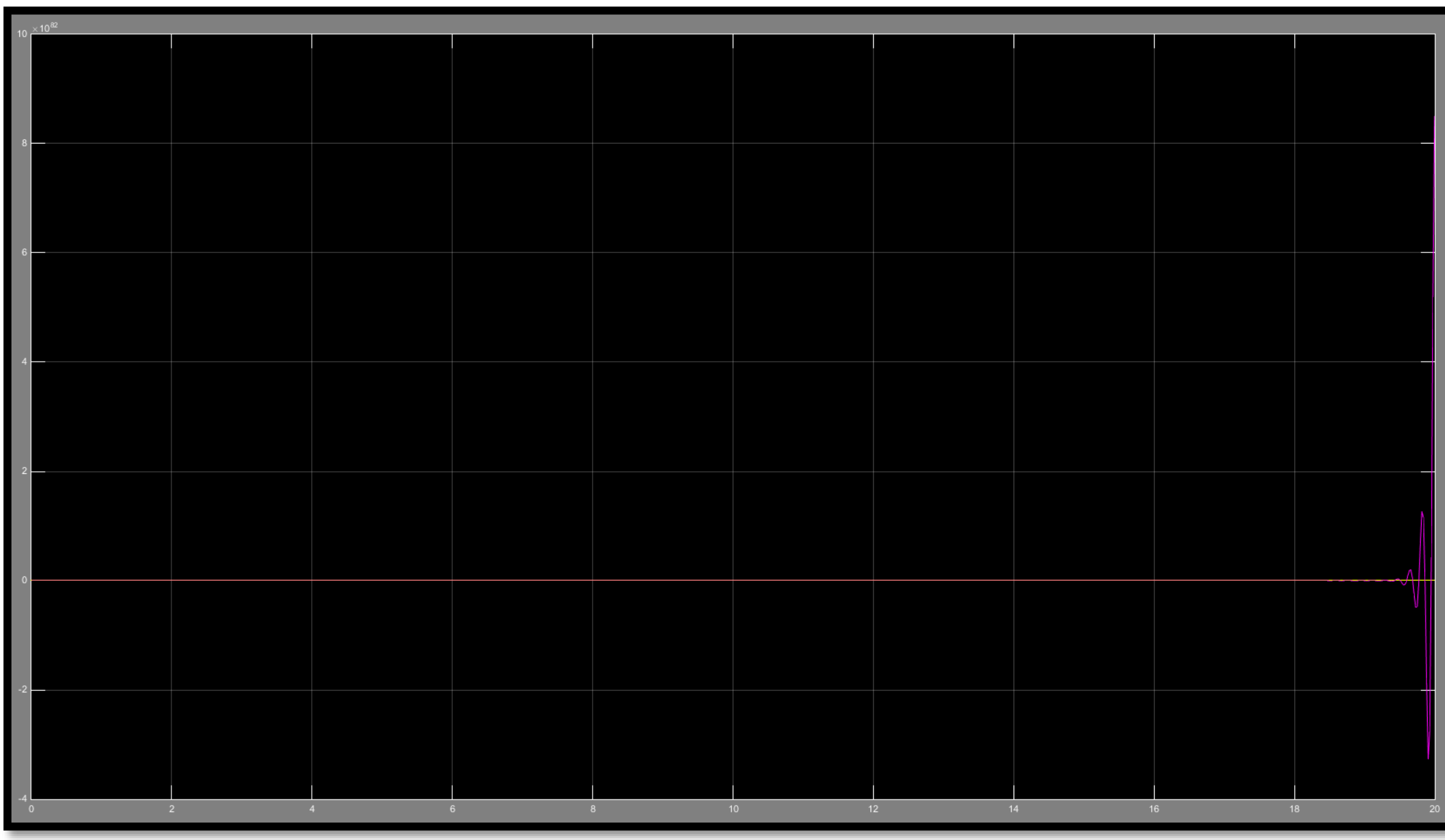

Figure 4 – Scope of System Model with Lead Compensator

Task 2

The governing dynamics was evaluated in state space form. Where 'A' and 'B' are state and control matrices, 'u' is the controller, and 'T' is the disturbance. The complete maple code used to derive these matrices can be found in the appendix.

```
> # This section creates the state space governing dynamics
z := Matrix(3, 1, [[θ], [θd], [xd]]);
```

$$\begin{bmatrix} \theta \\ \theta d \\ xd \end{bmatrix}$$

```
A := simplify(Matrix(3, 3, [[0, 1, 0], [coeff(θdd, θ), coeff(θdd, θd),
    coeff(θdd, xd)], [coeff(xdd, θ), coeff(xdd, θd), coeff(xdd, xd)]])
    );
```

$$\begin{bmatrix} 0 & 1 & 0 \\ \frac{g L (M+m)}{J} & -\frac{c_1}{J} & \frac{k_t k_{bemf}}{R r_a J} \\ 0 & 0 & -\frac{k_t k_{bemf} + c_2 R^2 r_a}{R^2 r_a (M+m)} \end{bmatrix}$$

```
B := simplify(Matrix(3, 1, [[0], [coeff(θdd, V)], [coeff(xdd, V)]]));
```

$$\begin{bmatrix} 0 \\ -\frac{k_t}{r_a J} \\ \frac{k_t}{r_a R (M+m)} \end{bmatrix}$$

```
T := simplify(Matrix(3, 1, [[0], [coeff(θdd, d)], [coeff(xdd, d)]]));
```

$$\begin{bmatrix} 0 \\ \frac{m g}{J} \\ 0 \end{bmatrix}$$

```
C := Matrix(1, 3, [1, 0, 0]);
```

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

```
zd := simplify(Az + B·u + T·d);
```

$$\left[\left[\begin{array}{c}\theta d\end{array}\right],\right.$$

$$\left[\frac{1}{R\, r_a\, J}\left(g\, L\, \theta\, R\, r_a\, M + g\, L\, \theta\, R\, r_a\, m - c_1\, \theta d\, R\, r_a + xd\, k_t\, k_{bemf} - u\, k_t\, R + m\, g\, d\, R\, r_a\right)\right],$$

$$\left.\left[-\frac{xd\, k_t\, k_{bemf} + xd\, c_2\, R^2\, r_a - u\, k_t\, R}{R^2\, r_a\, (M+m)}\right]\right]$$

$y := Cz;$

$$\left[\begin{array}{c}\theta\end{array}\right]$$

Figure 5 – Governing Dynamic Matrices

The stability can be determined by taking the Eigen values of matrix 'A'. If all of the calculated Eigen values are negative, it can be inferred that the open loop system will be stable. However, the derived Eigen values, shown below, has a positive value in its first cell. Thus, it can be implied that the open loop system is not stable on its own.

> $Eigenvalues(A);$

$$\left[\begin{array}{c}2.506831634\\-0.008792102204\\-2.506961842\end{array}\right]$$

Figure 6 – Eigen Values of Governing Dynamics Matrix A

Task 3

By using Matlab, the system will analyzed to determine if it can be controllable and the states are observable. To do so the 'ctrb' and 'obsv' functions were utilized from Matlab and then subtracted from the rank. A short script was written to determine the controllability and observability. From these results the system is controllable, with a controllability of 3. There are also no unobservable states in this system.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Task 3 - Controllability & Observability
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Co = ctrb(A,B);
controllability = rank(Co);
unco = length(A) - controllability;

if unco == 0
    disp('System is controllable.');
else
    disp('System is not controllable.');
end
disp('Controllability = ');
disp(controllability);
disp(' ');


Ob = obsv(A,C);
Unob = length(A) - rank(Ob);        % Number of unobservable states
if Unob ~= 0
    disp('System is not observable.');
    disp('Number of unobservable states');
    disp(Unob);
else
    disp('There are no unobservable states.');
end
```

Figure 7 – Controllability & Observability Script

Task 4

The state space matrices have been rewritten in control canonical and modal canonical form. This is done by utilizing the 'canon' functions in Matlab. The full script used to compute these matrices are included in the appendix.

```
Control Canonical A -
      0          0     0.0553
   1.0000        0     6.2845
      0      1.0000   -0.0089

Control Canonical B -
   1
   0
   0

Control Canonical C -
      0   -0.0070    0.0001

Control Canonical D -
   0
```

Figure 8 – Control Canonical Form

```
Modal Canonical A -
   2.5068        0          0
      0      -2.5070        0
      0          0     -0.0088

Modal Canonical B -
  -0.0524
  -0.0527
   0.0044

Modal Canonical C -
   0.0265   -0.0265   -0.0022

Modal Canonical D -
   0
```

Figure 9 – Modal Canonical Form

Task 5

Using the ‘acker’ function in Matlab a control law, ‘K’, can be determined for the desired poles for the closed loop function. The chosen poles for this system are shown below, along with the determined gains for the control law. The code used to perform this task is shown below in the appendix.

Poles	-10, -10, -50
Acker Evaluated Gains	-158600, -124090, -91330

Table 3 – Chosen Pole & Derived Gains for the Control Law

Task 6

Two different state space system were created in Simulink; one which includes disturbance input, and one which utilizes the state space block. The results from the state space diagram seems unsteady in comparison to the model which uses transfer functions and a PID controller (Phase 1 of the term project). It also seems that the state space model is able to take larger time steps than, thus demanding less computational resources from the computer. This would be highly beneficial in a system that must scale.

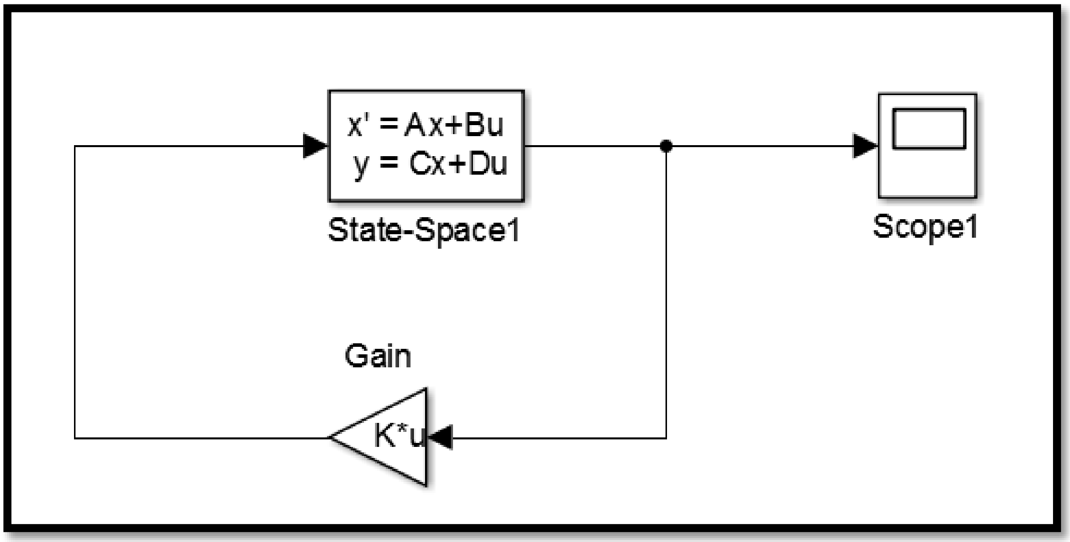

Figure 10 – State Space Model with State Space Block

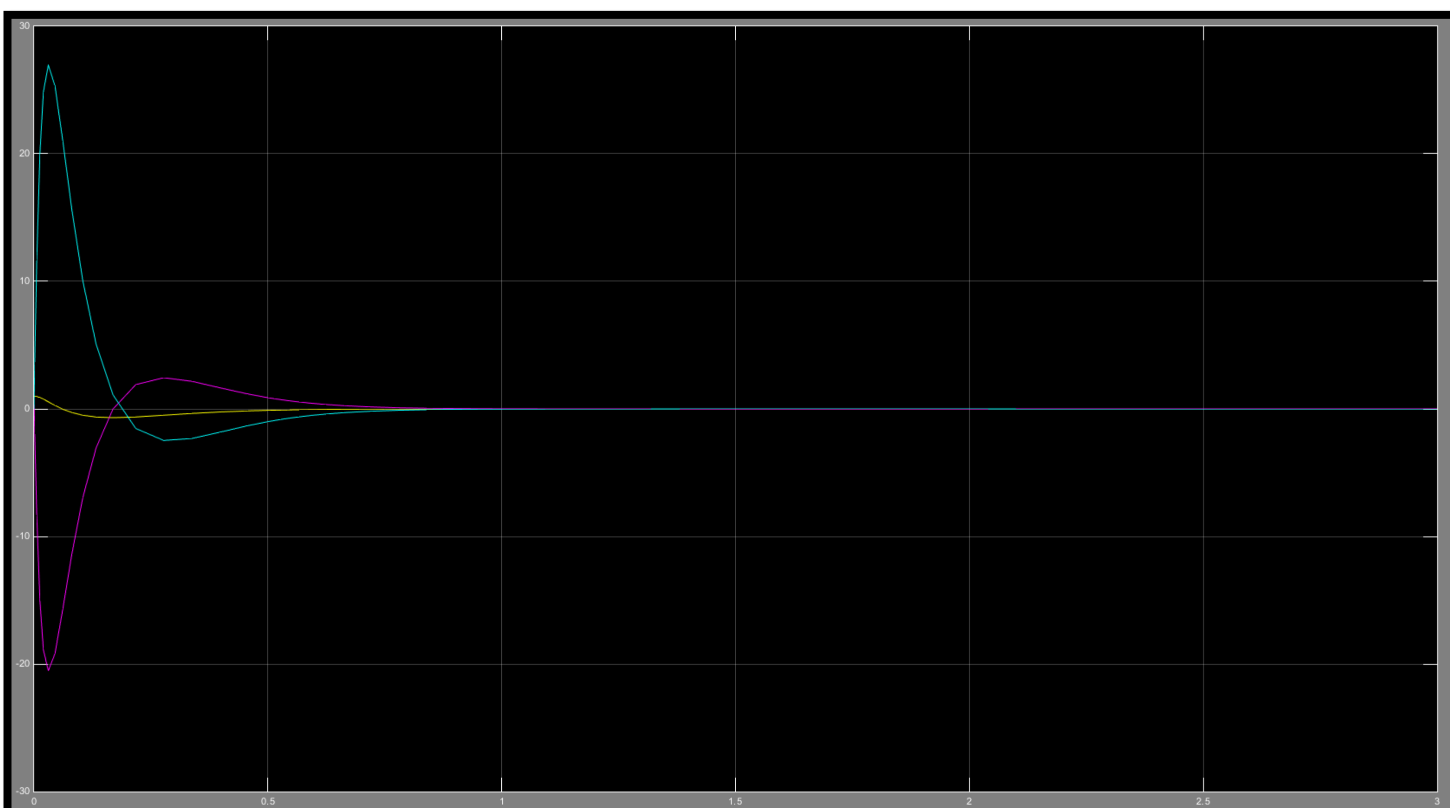

Figure 11 – Scope of State Space Model with State Space Block

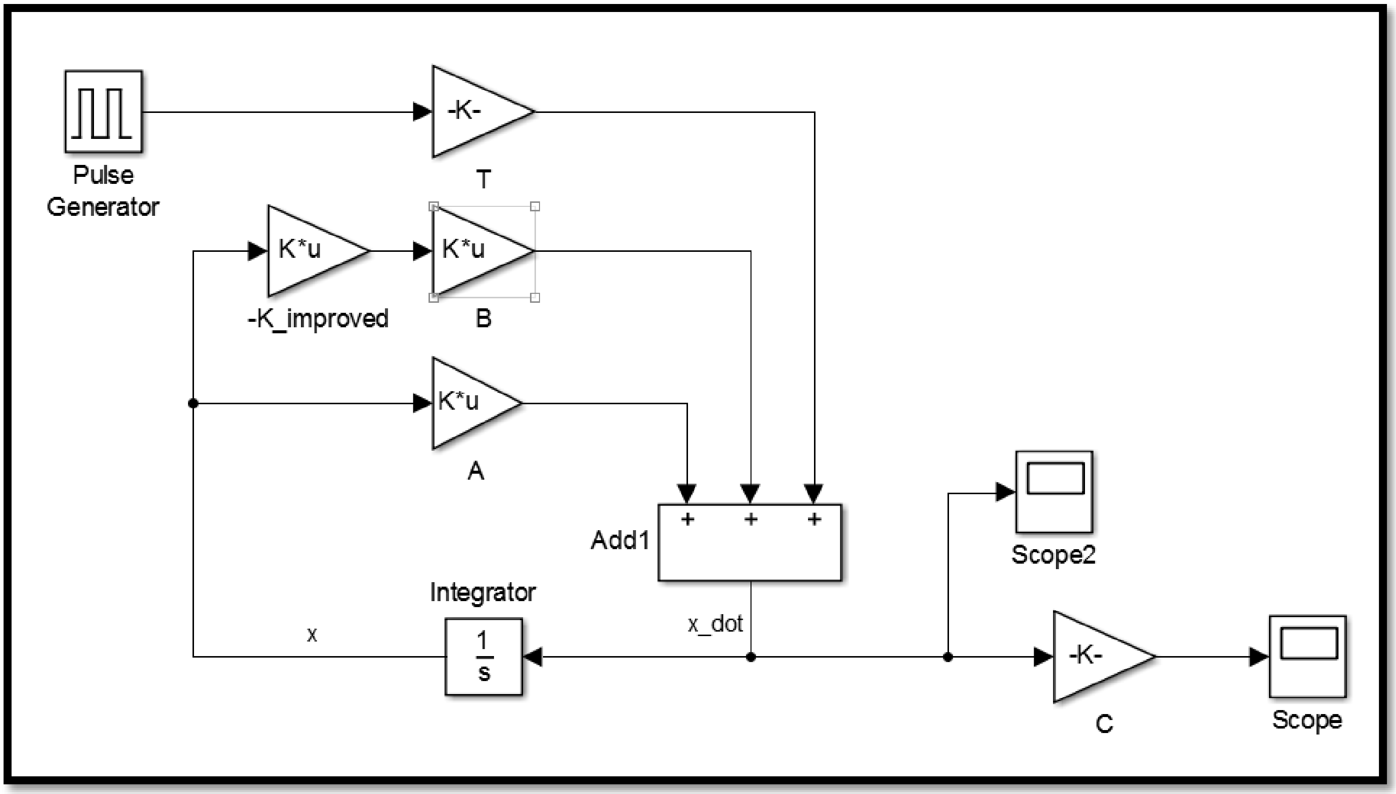

Figure 12 – State Space Model with Disturbance Input

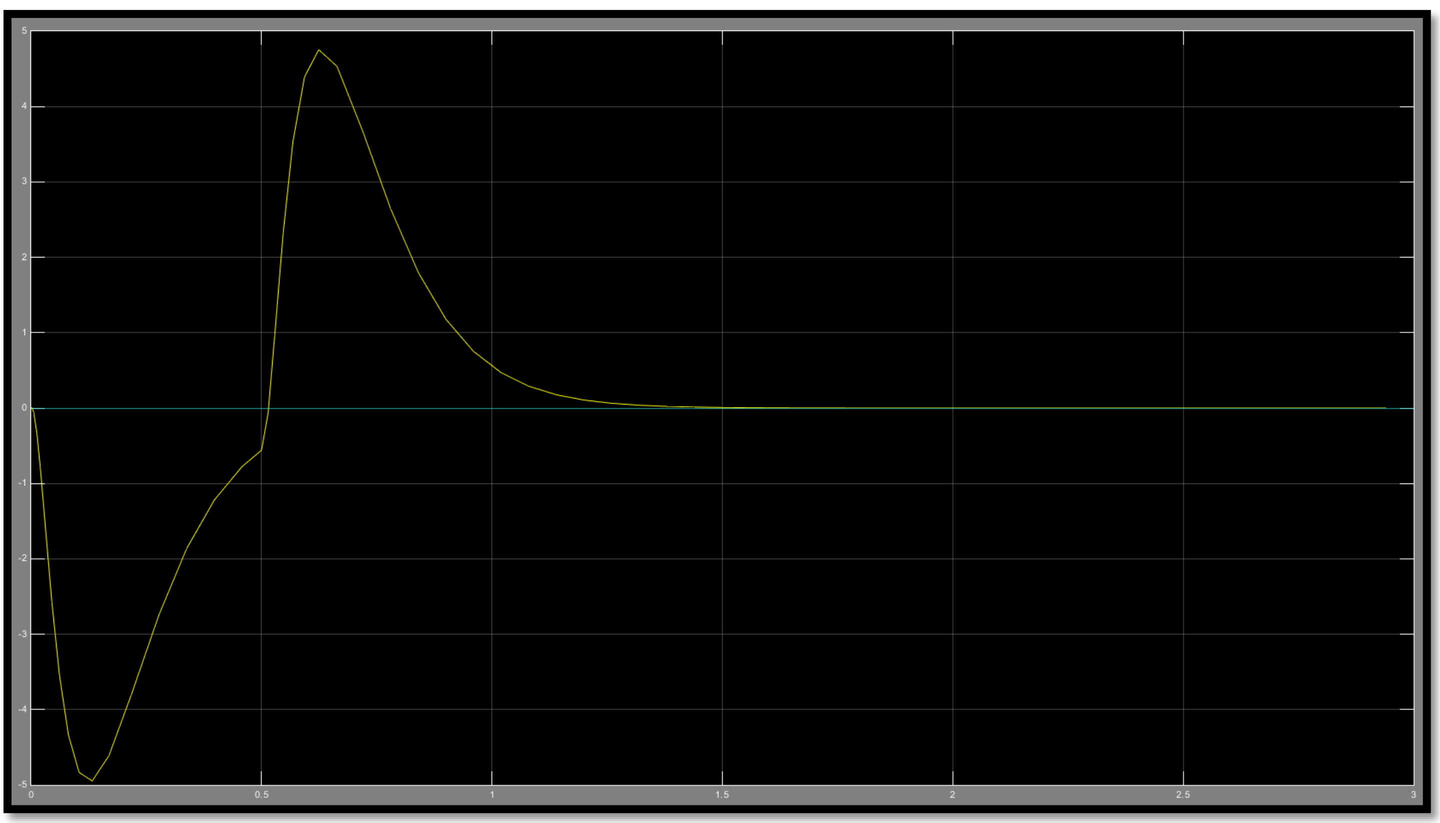

Figure 13 – Scope of State Space Model with Disturbance Input

Bonus

Instead of reading the velocity of the Segway directly from the sensor, a observer system was created to take its place. The observer was created with poles that are 10 times larger than the original system because the observer must respond much faster than the actual system to provide reliable values of the velocity. Where the velocity of the system was input into the state space system, is now replaced by an observer, which will estimate the velocity over time. The full code used to determine the gain for the observer is included in the appendix.

Poles	-100 + 5i,	-100 + 5i,	-500
Acker Evaluated Gains	699.9911,	110030,	718450000

Table 4 – Chosen Poles & Derived Gains for the Observer

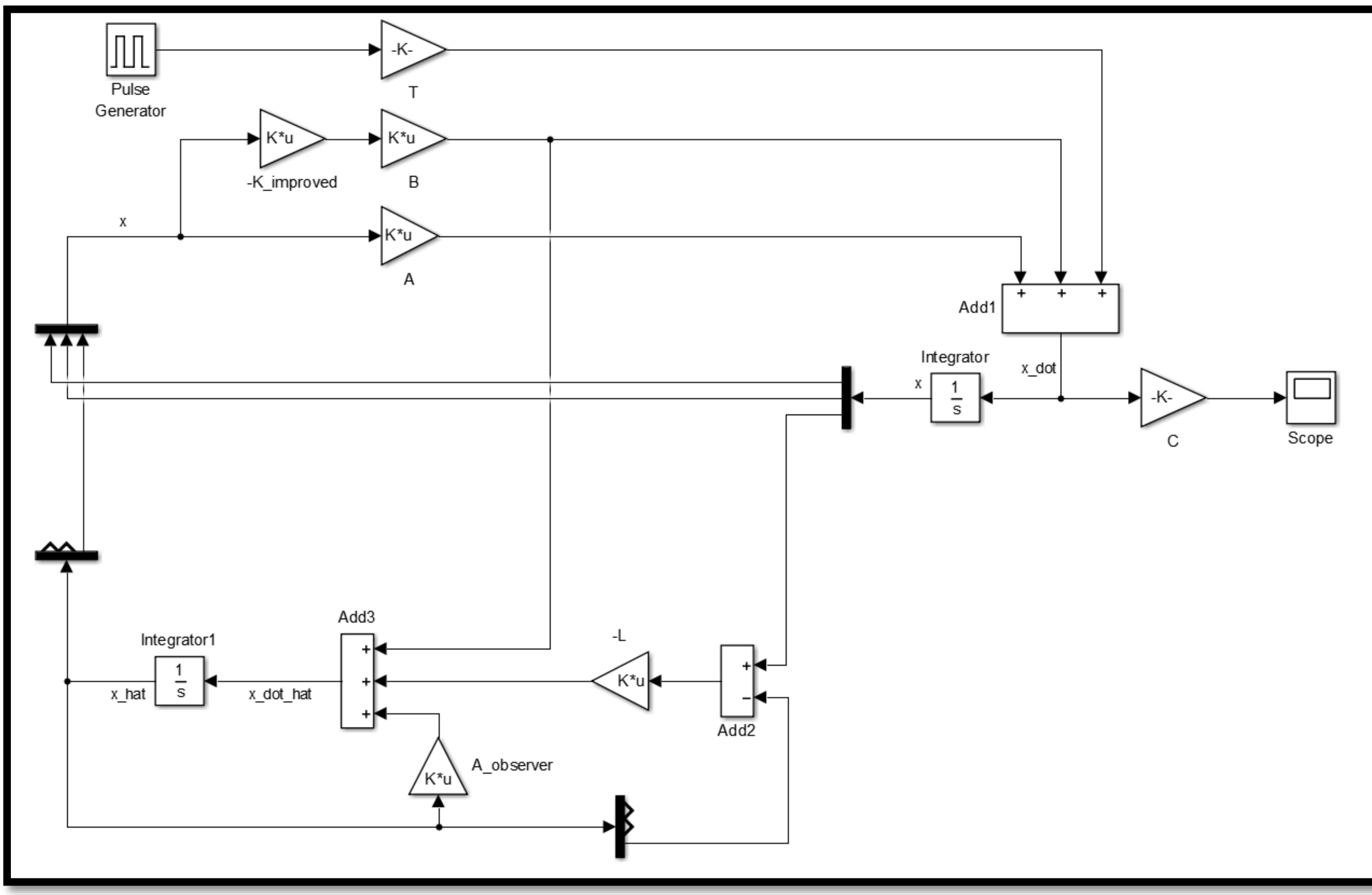

Figure 14 – Simulink Model of Velocity Observer

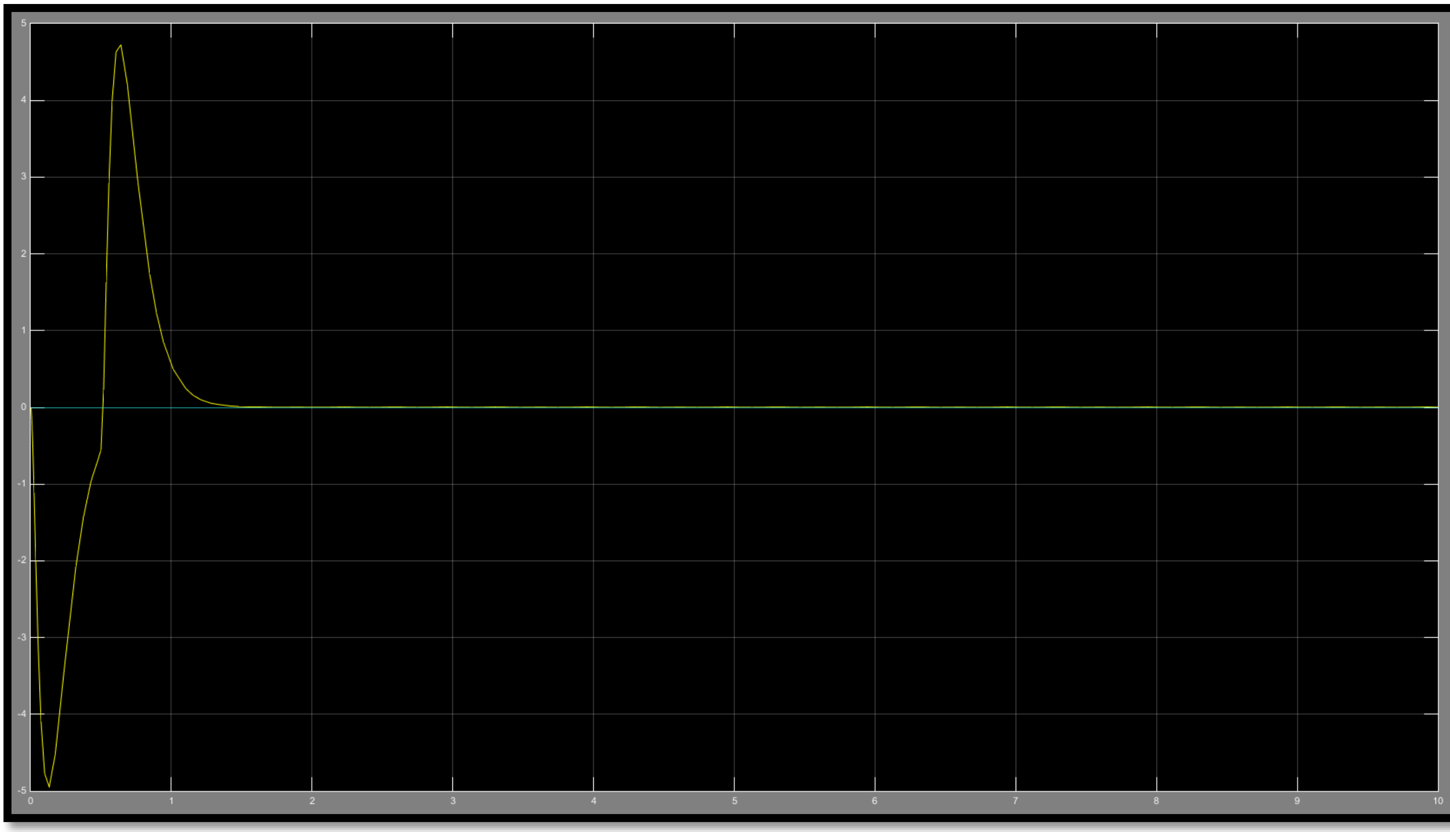

Figure 15 – Scope of system utilizing Velocity Observer

Appendix

(i) Calculate Governing Dynamics Matrices in State Space Form

restart;

 # Include Libraries

with(*inttrans*);

with(*DynamicSystems*);

with(*LinearAlgebra*);

 # summation of all moments

$$sumEqn_{\theta} := J\cdot\theta\cdot s^2 = M\cdot g\cdot L\cdot\theta + m\cdot g\cdot l\cdot\theta - T - c_1\cdot\theta\cdot s + m\cdot g\cdot d;$$

$$J\,\theta\,s^2 = M\,g\,L\,\theta + m\,g\,l\,\theta - T - c_1\,\theta\,s + m\,g\,d$$

 # summation of all forces in the x-direction

$$sumEqn_{velocity} := (M+m)\cdot x\cdot s^2 = \frac{T}{R} - c_2\cdot x\cdot s;$$

$$(M+m)\,x\,s^2 = \frac{T}{R} - c_2\,x\,s$$

 # Predefined Equations

$$\beta := \frac{k_t\cdot k_{bemf}}{R\cdot r_a};$$

$$\frac{k_t\,k_{bemf}}{R\,r_a}$$

$$\alpha := \frac{k_t}{r_a};$$

$$\frac{k_t}{r_a}$$

$$T := \alpha\cdot V - \beta\cdot x\cdot s;$$

$$\frac{k_t\,V}{r_a} - \frac{k_t\,k_{bemf}\,x\,s}{R\,r_a}$$

 # Simplified Governing dynamics

$$\theta dd := \frac{(M\cdot g\cdot L + m\cdot g\cdot L)}{J}\cdot\theta + \frac{\beta}{J}\cdot xd - \frac{\alpha}{J}\cdot V + \frac{m\cdot g}{J}\cdot d - \frac{c_1}{J}\cdot\theta d;$$

$$xdd := \frac{\alpha}{R\cdot(M+m)}\cdot V - \frac{\frac{\beta}{R} + c_2}{M+m}\cdot xd;$$

$$\frac{(M\,g\,L + m\,g\,L)\,\theta}{J} + \frac{k_t\,k_{bemf}\,xd}{R\,r_a\,J} - \frac{k_t\,V}{r_a\,J} + \frac{m\,g\,d}{J} - \frac{c_1\,\theta d}{J}$$

$$\frac{k_t\,V}{r_a\,R\,(M+m)} - \frac{\left(\frac{k_t\,k_{bemf}}{R^2\,r_a} + c_2\right)xd}{M+m}$$

 # This section creates the state space governing dynamics

$$z := Matrix(3, 1, [[\theta], [\theta d], [xd]]);$$

$$\begin{bmatrix}\theta\\ \theta d\\ xd\end{bmatrix}$$

$A := simplify\Big(Matrix\big(3, 3, \big[[0, 1, 0], \big[coeff(\theta dd, \theta), coeff(\theta dd, \theta d),$
$coeff(\theta dd, xd)\big], \big[coeff(xdd, \theta), coeff(xdd, \theta d), coeff(xdd, xd)\big]\big]\big)$
$\Big);$

$$\begin{bmatrix} 0 & 1 & 0 \\ \frac{g\,L\,(M+m)}{J} & -\frac{c_1}{J} & \frac{k_t\,k_{bemf}}{R\,r_a\,J} \\ 0 & 0 & -\frac{k_t\,k_{bemf} + c_2\,R^2\,r_a}{R^2\,r_a\,(M+m)} \end{bmatrix}$$

$B := simplify\big(Matrix\big(3, 1, \big[[0], [coeff(\theta dd, V)], [coeff(xdd, V)]\big]\big)\big);$

$$\begin{bmatrix} 0 \\ -\frac{k_t}{r_a\,J} \\ \frac{k_t}{r_a\,R\,(M+m)} \end{bmatrix}$$

$T := simplify\big(Matrix\big(3, 1, \big[[0], [coeff(\theta dd, d)], [coeff(xdd, d)]\big]\big)\big);$

$$\begin{bmatrix} 0 \\ \frac{m\,g}{J} \\ 0 \end{bmatrix}$$

$C := Matrix(1, 3, [1, 0, 0]);$

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$zd := simplify(A.z + B \cdot u + T \cdot d);$

$$\left[\begin{bmatrix} \theta d \end{bmatrix}, \left[\frac{1}{R\,r_a\,J} \big(g\,L\,\theta\,R\,r_a\,M + g\,L\,\theta\,R\,r_a\,m - c_1\,\theta d\,R\,r_a + xd\,k_t\,k_{bemf} - u\,k_t\,R + m\,g\,d\,R\,r_a \big) \right], \left[-\frac{xd\,k_t\,k_{bemf} + xd\,c_2\,R^2\,r_a - u\,k_t\,R}{R^2\,r_a\,(M+m)} \right] \right]$$

$y := C.z;$

$$\begin{bmatrix} \theta \end{bmatrix}$$

$R := 0.5;$
$L := 0.4;$
$l := 1;$
$M := 55;$
$m := 68;$
$J := M \cdot L^2 + m \cdot l^2;$
$k_t := 0.75;$
$k_{bemf} := 0.5;$
$r_a := 1.4;$
$c_1 := 0.01;$
$c_2 := 0.01;$
$g := 9.81;$

Eigenvalues(A);

$$\begin{bmatrix} 2.506831634 \\ -0.008792102204 \\ -2.506961842 \end{bmatrix}$$

 # Since the first cell of the Eigenvalues is not negative, it can be infered that the system with no disturbance and no controller will be unstable.

(ii) Matlab Program – [Task 3-5]

```
A = [0 1 0; (L*(M*g+m*g)/J) (-c1/J) (k_t*k_bemf/(R*r_a*J)); 0 0 -
(k_t*k_bemf+c2*R^2*r_a)/(R^2*r_a*(M+m))];
B = [0; (-k_t/(r_a*J)); (k_t/(r_a*R*(M+m)))];
T = [0 m*g/J 0];



C = [1 0 0];
D = 0;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Task 2 - Stability
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
e = eig(A);
disp('Eigen values of matrix A - ');
disp(e);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Task 3 - Controllability & Observability
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Co = ctrb(A,B);
controllability = rank(Co);
unco = length(A) - controllability;

if unco == 0
    disp('System is controllable.');
else
    disp('System is not controllable.');
end
disp('Controllability = ');
disp(controllability);
disp(' ');
```

```
Ob = obsv(A,C);
Unob = length(A) - rank(Ob);          % Number of unobservable states
if Unob ~= 0
    disp('System is not observable.');
    disp('Number of unobservable states');
    disp(Unob);
else
    disp('There are no unobservable states.');
end
disp(' ');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Task 4 - Control & Modal Canonical Form
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sys = ss(A,B, C, D);
ccsys = canon(sys, 'control');
cmsys = canon(sys, 'modal');

disp('Control Canonical A -');
disp(ccsys.A);
disp('Control Canonical B -');
disp(ccsys.B);
disp('Control Canonical C -');
disp(ccsys.C);
disp('Control Canonical D -');
disp(ccsys.D);


disp('Modal Canonical A -');
disp(cmsys.A);
disp('Modal Canonical B -');
disp(cmsys.B);
disp('Modal Canonical C -');
disp(cmsys.C);
disp('Modal Canonical D -');
disp(cmsys.D);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Task 5 - ACKER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
p1 = -10;
p2 = -10;
p3 = -50;

K = acker(A,B,[p1 p2 p3]);
disp('acker K gains - ')
disp(K);
```

(iii) Observer Gains Script

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bonus - Observer Design
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

A = [0 1 0; (L*(M*g+m*g)/J) (-c1/J) (k_t*k_bemf/(R*r_a*J)); 0 0 -
(k_t*k_bemf+c2*R^2*r_a)/(R^2*r_a*(M+m))];
B = [0; (-k_t/(r_a*J)); (k_t/(r_a*R*(M+m)))];
T = [0 m*g/J 0];

C = [1 0 0];
D = 0;


At = transpose(A);
Ct = transpose(C);


p1 = -100 + 5i;
p2 = -100 - 5i;
p3 = -500;

K1 = transpose(acker(At,Ct,[p1 p2 p3]));

disp('Observer gains');
disp(K1(1,1));
disp(K1(2,1));
disp(K1(3,1));
```