DIGITAL CONTROL SYSTEMS

MAE 444/544

Project 2

Professor- Dr. Aaron Estes

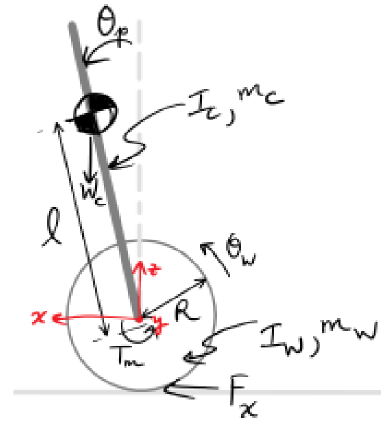
By- Nikhil Arora

UB Person No- 50320282

Part One: Designing a regulator for a self-balancing robot. For this part of the project, you will design a controller that keeps this robot upright, by commanding the wheels to spin at the right speed.

Vehicle Parameters:

Stepper Motor Mass, m_{SM}	300 g (each)
Motor Rotor Inertia, I_R	54 g·cm^2 (each)
Tire Mass, m_T	52 g (each)
Tire Inertia, I_T	520 g·cm^2 (each)
Tire Radius, R	4 cm
Frame Mass, m_F	160 g
Arduino and Breadboard, m_A	100 g
Battery Mass, m_B	120 g



Combined Parameters:

Chassis Combined Mass, m_C	980 g
Chassis Combined Inertia about Axle, I_C	92,336 g·cm^2
Distance from Axle to CG of Chassis, ℓ	6 cm
Wheel and Rotor Combined Mass, m_W	150 g
Wheel and Rotor Combined Inertia, I_W	1150 g·cm^2

The linearized equation of motion for the self-balancing robot is given by

$$\alpha\ddot{\theta}_p - \eta\theta_p = -\gamma\ddot{\theta}_w \tag{1}$$

where

1. $\alpha = I_C + m_C R\ell$
2. $\eta = m_C g\ell$ (where $g = 981$ cm/s^2)
3. $\gamma = I_W + (m_W + m_C)R^2$

We measure **two outputs**: the pitch rate of the robot, $\dot{\theta}_p$, and the angular velocity of the wheels, $\dot{\theta}_w$. The system has **one control input**: the angular acceleration of the wheels, $\ddot{\theta}_w$. Therefore,

$\mathbf{y} = \begin{bmatrix} \dot{\theta}_p \\ \dot{\theta}_w \end{bmatrix}$ and $u = \ddot{\theta}_w$

Project Deliverables: Part One

1. Derive the A_c and B_c matrices from the continuous-time state-space model for the system, assuming that the system states are given by

$$\mathbf{x} = \begin{bmatrix} \theta_p \\ \dot{\theta}_p \end{bmatrix} \tag{2}$$

   You may write these matrices in terms of the parameters α, η, and γ (you don't have to plug in all the numbers right now, in other words).

2. Using your results from the previous part, derive the A and B matrices for the discrete-time state-space model, assuming that the sampling frequency is $f_s = 100$ Hz. You should do this in MATLAB, using the provided numerical values for each parameter.

3. Add the wheel velocity, $\dot{\theta}_w(k)$ as an additional state to your discrete-time state-space model, where

$$\dot{\theta}_w(k) = \dot{\theta}_w(k-1) + \ddot{\theta}_w(k-1) \cdot T \tag{3}$$

   so that the state vector becomes

$$\mathbf{x}(k) = \begin{bmatrix} \theta_p(k) \\ \dot{\theta}_p(k) \\ \dot{\theta}_w(k) \end{bmatrix} \tag{4}$$

   Derive new A and B matrices to accommodate this additional state.

4. Now, define the control input as

$$\ddot{\theta}_w(k) = u(k) = -K\mathbf{x}(k) \tag{5}$$

   and calculate the feedback gain matrix, K, such that the closed-loop system settling time is approximately 1.5 s, with 30% overshoot.

5. Next, we need to design an estimator to give $\hat{\theta}_p(k)$, $\hat{\dot{\theta}}_p(k)$, and $\hat{\dot{\theta}}_w(k)$ at each time step. We measure $\dot{\theta}_p(k)$ with a gyroscope, and let us assume that we also measure $\dot{\theta}_w(k)$ (we know the wheel velocity fairly accurately because we regulate the stepper motor timing). Therefore, assume that our output equation is

$$\mathbf{y}(k) = \begin{bmatrix} \dot{\theta}_p(k) \\ \dot{\theta}_w(k) \end{bmatrix} = C\mathbf{x}(k) \tag{6}$$

   Appropriately define C, then use the `place` command in MATLAB to calculate the L matrix so that the state estimation error has a settling time of 0.5 s, and an overshoot of 20%. (The reason we use the `place` command instead of `acker` is that `acker` only works for systems with one output. Here, we have two outputs.) You can use the following syntax: `L = place(A',C',betastar)'` where `betastar` is a vector of your estimator poles.

6. Given your controller and your estimator, simulate the closed-loop system response in MATLAB. Assume that the true initial state vector is

$$\mathbf{x}(0) = \begin{bmatrix} \theta_p(0) \\ \dot{\theta}_p(0) \\ \dot{\theta}_w(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.2 \\ 0 \end{bmatrix} \tag{7}$$

and that the estimated initial state vector is

$$\hat{\mathbf{x}}(0) = \begin{bmatrix} \hat{\theta}_p(0) \\ \hat{\dot{\theta}}_p(0) \\ \hat{\dot{\theta}}_w(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

For this simulation, include the following plots:

(a) The two system outputs, $\mathbf{y}(k)$, as a function of time

(b) The system control input, $u(k)$, as a function of time

(c) The state estimation error, $\tilde{\mathbf{x}}(k) = \hat{\mathbf{x}}(k) - \mathbf{x}(k)$, as a function of time.

Part One

$m_c = 980\,g$ $\quad$ $R = 4\,cm$

$I_c = 92,336\ g\text{-}cm^2$

$l = 6\,cm$

$m_w = 150\,g$

$I_w = 1150\ g\text{-}cm^2$

$$\alpha \ddot{\theta}_p - \eta \theta_p = -\gamma \ddot{\theta}_w \qquad -(1)$$

1. $\alpha = I_c + m_c R l$

$\alpha = 115856$

2. $\eta = m_c g l$

$\eta = 5768280$

3. $\gamma = I_w + (m_w + m_c) R^2$

$\gamma = 19230$

(i) From (1)

$$\alpha \ddot{\theta}_p = \eta \theta_p - \gamma \ddot{\theta}_w$$

$$\dot{x} = Ax + Bu$$

$$\begin{bmatrix} \dot{\theta}_p \\ \ddot{\theta}_p \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ \eta/\alpha & 0 \end{bmatrix}}_{A} \begin{bmatrix} \theta_p \\ \dot{\theta}_p \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -\gamma/\alpha \end{bmatrix}}_{B} u$$

Also, $y = Cx + Du$

$$\begin{bmatrix} \dot{\theta}_p \\ \dot{\theta}_w \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}}_{C} x + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{D} u$$

(ii) from MATLAB

$$A_c = \begin{bmatrix} 0 & 1 \\ 49.7884 & 0 \end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 \\ 0.166 \end{bmatrix}$$

Similarly

$$A_D = \begin{bmatrix} 1.0025 & 0.0100 \\ 0.4983 & 1.0025 \end{bmatrix}$$

$$B_D = \begin{bmatrix} 0 \\ -0.0017 \end{bmatrix}$$

(iii) $\dot{\theta}_w(k) = \dot{\theta}_w(k-1) + \ddot{\theta}_w(k-1)T$

$$\begin{bmatrix} \theta_p(k) \\ \dot{\theta}_p(k) \\ \ddot{\theta}_w(k) \end{bmatrix} = \underbrace{\begin{bmatrix} 1.0025 & 0.01 & 0 \\ 0.4983 & 1.0025 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{A_{new}} \begin{bmatrix} \theta_p(k) \\ \dot{\theta}_p(k) \\ \ddot{\theta}_w(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ -0.0017 \\ T \end{bmatrix}}_{B_{new}} r$$

(10) $t_s = 1.5s$

$M_p = 30\%$

$$M_p = e^{-\pi\zeta/\sqrt{1-\zeta^2}}$$

$$e^{-\pi\zeta/\sqrt{1-\zeta^2}} = 0.3$$

$$\ln(0.3) = \frac{-\pi\zeta}{\sqrt{1-\zeta^2}}$$

$$+1.2 = +\frac{\pi\zeta}{\sqrt{1-\zeta^2}}$$

$$1.44 - 1.44\zeta^2 = \pi^2\zeta^2$$

$$\zeta = 0.363$$

$$t_s = \frac{4}{\zeta\omega_n} = 1.5$$

$$\zeta\omega_n = 2.66$$

$$\lambda_c = -\zeta\omega_n \pm \omega_d i$$
$$= -2.66 \pm 6.96i$$

$$z^* = e^{\lambda_c T}$$
$$= 0.97 \pm 0.067i$$

From MATLAB

$$K = \begin{bmatrix} -707.76 & -67.0084 & -2.855 \end{bmatrix}$$

(v) $y(k) = \begin{bmatrix} \theta_p(k) \\ \theta_w(k) \end{bmatrix} = Cx(k)$

$$\begin{bmatrix} \dot\theta_p(k) \\ \dot\theta_w(k) \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{C} \begin{bmatrix} \theta_p(k) \\ \dot\theta_p(k) \\ \ddot\theta_w(k) \end{bmatrix}$$

We know, $M_p = 20\%$

$t_s = 0.5$

$$M_p = e^{-\pi\zeta / \sqrt{1-\zeta^2}}$$

$$\zeta = -0.455$$

$$t_s = \frac{4}{\zeta\omega_n}$$

$$\zeta\omega_n = 4/0.5 = 8$$

$$\lambda_c = -\zeta\omega_n \pm \omega_d i$$

$$= -8 \pm 15.615i$$

$$\beta^* = 0.9119 \pm 0.1436i$$

$$L = \begin{bmatrix} 0.0678 & 0.1812 & 0 \\ 0 & 0 & 0.0769 \end{bmatrix}$$

```
clear all; close all; clc;
%% Declaring intial values
Tfinal = 1.5;
frequency = 100;
T = 1/frequency;
t = (0:T:Tfinal)';
N = length(t);
n = 5768280;
alpha = 115856;
gamma = 19230;
Ts = 1.5;
Mp = 0.3;
zeta = sqrt((log(Mp))^2/(pi^2+(log(Mp))^2));
zeta_omegaN = 4/Ts;
omega_N = zeta_omegaN/zeta;
omega_D = omega_N*sqrt(1-zeta^2);
Roots_cont = [-zeta_omegaN+omega_D*1i -zeta_omegaN-omega_D*1i];
Roots_disc = exp(Roots_cont*T);
z_star = [Roots_disc(1) Roots_disc(2)];
Ac = [0 1;n/alpha 0];   %continous state space model
Bc = [0 -gamma/alpha]';
A = expm(Ac*T);         %continous to discrete state space
B = (A - eye(length(A)))*Ac^-1*Bc;
A(1:2,3) = 0;
A(3,1:2) = 0;
A(3,3) = 1;
B(3,1) = T;
K_matrix = acker(A,B,[Roots_disc(1), Roots_disc(2), abs(Roots_disc(1))]); %
calculating k matrix
C = [0 1 0;0 0 1];
Ts_estimate = 0.5;
Mp_estimate = 0.2;
zeta_estimate = sqrt((log(Mp_estimate))^2/(pi^2+(log(Mp_estimate))^2));
zeta_omegaN_estimate = 4/Ts_estimate;
omega_N_estimate = zeta_omegaN_estimate/zeta_estimate;
omega_D_estimate = omega_N_estimate*sqrt(1-zeta_estimate^2);
Roots_cont_estimate = [-zeta_omegaN_estimate+omega_D_estimate*1i ...
    -zeta_omegaN_estimate-omega_D_estimate*1i];
Roots_disc_estimated = exp(Roots_cont_estimate*T);
beta_star = [Roots_disc_estimated(1) Roots_disc_estimated(2)];
char_Equation = poly([beta_star]);
L_matrix = place(A', C',[Roots_disc_estimated(1) Roots_disc_estimated(2) ...
    abs(Roots_disc_estimated(1))])';
%% Simulation
x = zeros(3,N);
x_hat = zeros(3,N);
y = zeros(2,N);
y_hat = zeros(2,N);
u = zeros(1,N);
```

```
x(:,1) = [0;0.2;0];
x_hat(:,1) = [0;0;0];
y_hat(1) = C(1,:)*x_hat(:,1);
u(1) = -K_matrix*x(:,1);
u(1) = 0;
error(:,1) = x_hat(:,1)-x(:,1);
for k = 2:N
    x(:,k) = A*x(:,k-1)+B*u(k-1);
    x_hat(:,k) = A*x_hat(:,k-1) + B*u(k-1) - L_matrix * (y_hat(:,k-1)-y(:,k-1));
    y_hat(:,k) = C(:,:)*x_hat(:,k);
    y(:,k) = C(:,:)*x(:,k);
    u(k) = -K_matrix*x_hat(:,k);
    error(:,k) = x_hat(:,k) - x(:,k);
end

%% Plots
figure(1)
plot(t,y(1,:)*180/pi, 'k-o', t,y(2,:)*180/pi,'r-o', 'linewi',1,'MarkerSize',4)
grid on
title('Measured Angle')
xlabel('Time')
ylabel('\theta/s')
legend('pitch rate','wheel angular veliocity')

figure(2)
stairs(t,u*180/pi,'k-','linewi',1, 'MArkersize',4)
grid on
title('Angular Acceleration')
xlabel('Time')
ylabel('\theta/s^2')
legend('wheel angular acceleration')

figure(3)
plot(t,error(1,:),'k-*', t,error(2,:),'r-*', t,error(3,:),'g-*','linewi',↙
1,'MarkerSize',4)
grid on
title('State Estimation Error')
xlabel('Time')
ylabel('Error')
legend('Error 1','Error 2','Error 3')
```

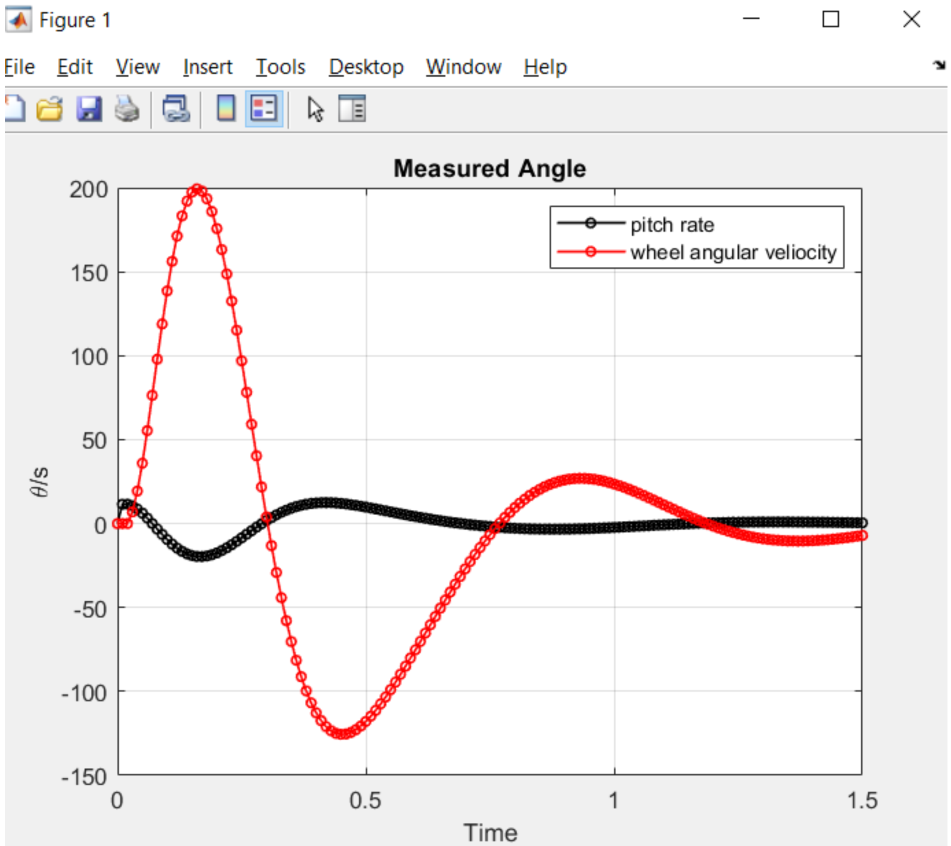
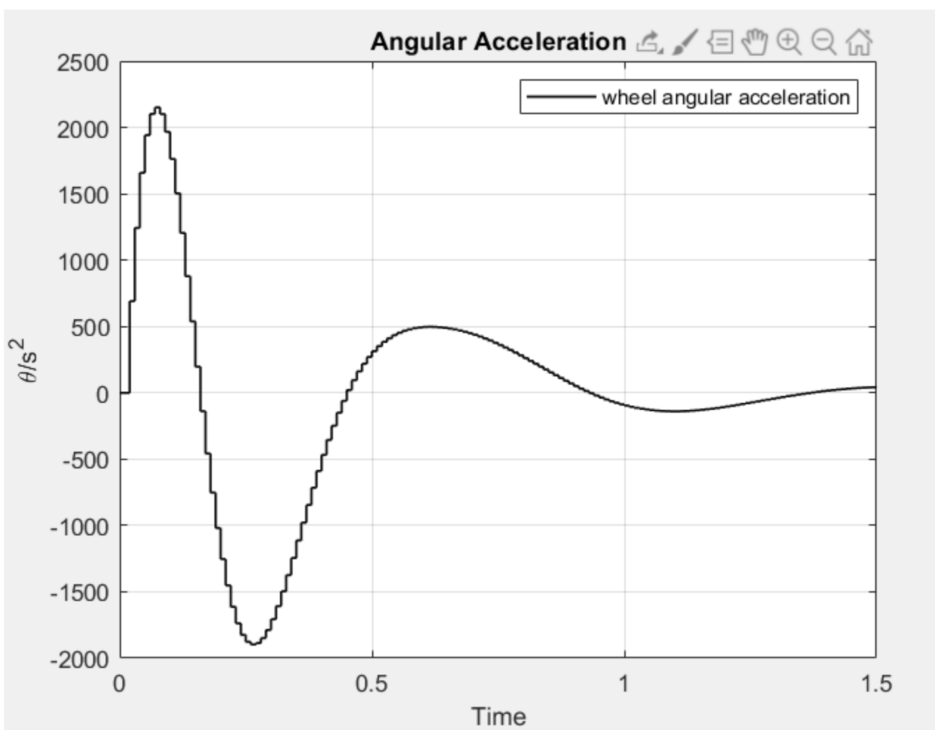
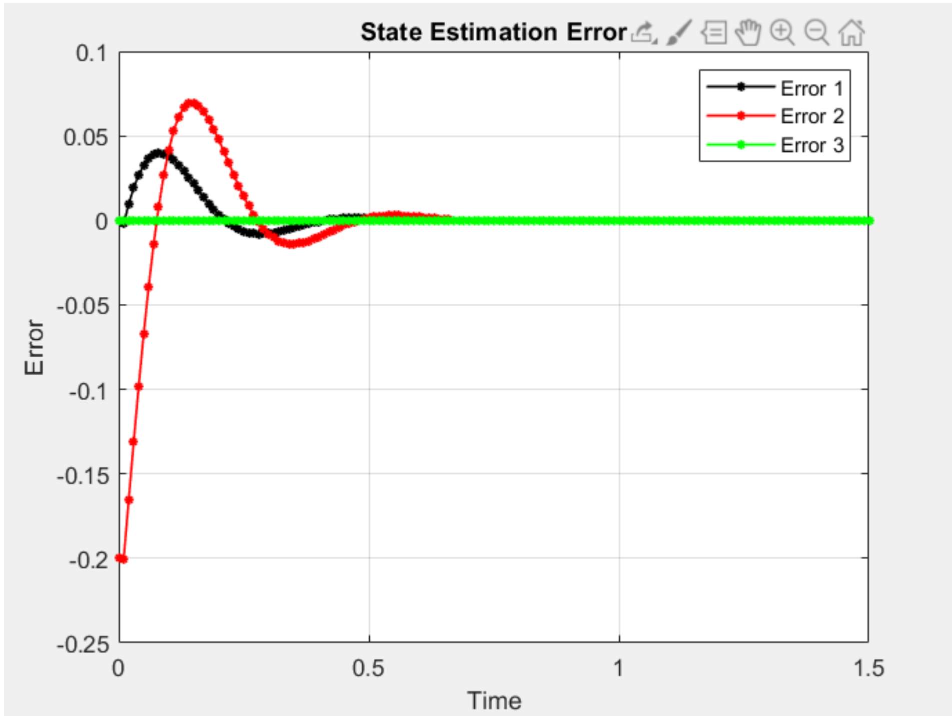


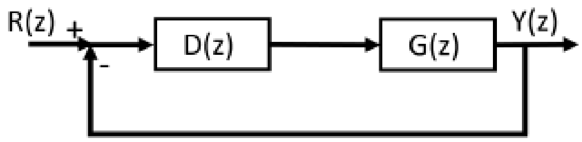
Figure 1
File Edit View Insert Tools Desktop Window Help
Measured Angle
pitch rate
wheel angular veliocity
θ/s
Time
200
150
100
50
0
-50
-100
-150
0
0.5
1
1.5


Angular Acceleration
wheel angular acceleration
θ/s2
Time
2500
2000
1500
1000
500
0
-500
-1000
-1500
-2000
0
0.5
1
1.5



State Estimation Error
Error 1
Error 2
Error 3
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
-0.25
Error
0
0.5
1
1.5
Time

Part Two: The goal of this part of the project is to design a tracking controller for the 6V DC motor we have used as a demo in class, using state-feedback control, with an integrator.



The transfer function for the motor is given by

$$G(z) = \frac{Y(z)}{U(z)} = \frac{1.4432}{z - 0.2299} \tag{9}$$

where $y(kT)$ is the motor angular velocity in rad/s and $u(kT)$ is the input voltage in volts. The sampling frequency for this transfer function is $f_s = 20$ Hz.

1. Put this system in controllable-canonical state-space form. (It will be a first-order system)
2. Calculate the reference state for a reference signal of 2π rad/s
3. Augment the system state vector to include the integral of the output error. Derive new A and B matrices to accommodate this additional state. Call these new matrices A_{AUG} and B_{AUG}.
4. Calculate the feedback gain matrix so that the following performance criteria are satisfied (remember to choose 2 poles, because the integrator has added a second state)
   (a) $t_s = 0.6$ s
   (b) $M_p = 0.2$
5. Derive an estimator that estimates just the first state (not the integral state) for your system. Calculate L such that the settling time of your estimator is 0.2 s. Note: your estimator is first order, so you have to pick one, real, target pole. L will be a scalar.
6. Simulate the system in MATLAB when a reference signal of $r = 2\pi$ rad/s is provided, and generate the following plots (remember to define your control signal in such a way that your output will track the reference! This is not a regulator problem):
   (a) The system output, $y(k)$, as a function of time
   (b) The system control input, $u(k)$, as a function of time
   (c) The state estimation error, $\tilde{x}(k) = \hat{x}(k) - x(k)$, as a function of time.

Part 2

$$G(z) = \frac{Y(z)}{U(z)} = \frac{1.4432}{z - 0.2299}$$

$\because f_s = 20Hz$

$T = 1/20 = 0.05$

(i) Controllable cannonical state space model

$A = [0.2299]$

$B = [1]$

$C = [1.4432]$

$D = b_0 = 0$

$\therefore \quad x_k = 0.2299\, x_{k-1} + u_{k-1}$

and $\quad y_k = 1.4432\, x_k$

(ii) reference signal $= 2\pi$ rad/s

We know $\lim_{k \to \infty} y_{ss} = r(k)$ — (1)

Also $\lim_{k \to \infty} u_k = u_{ss} = N_u - r(k)$

$$\lim_{k \to \infty} x_k = x_{ss} = N_x \, r(k)$$

From (i)

$$\lim_{k \to \infty} C x_k = C N_x r = r$$

$$C N_x = 1$$

At steady state we know,

$$x(k+1) = A x(k) + B x(k) \quad -(1)$$

and

$$x_{ss} = A x_{ss} + B u_{ss} \quad -(2)$$

From (1) and (2)

$$\begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} A-I & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} 0.6929 \\ 0.5338 \end{bmatrix}$$

Therefore reference state

$$x\_r = N_x \cdot r$$
$$= 4.3536$$

(iii) $x_I(k) = x_I(k-1) + r(k-1) - Cx(k-1)$

$$\begin{bmatrix} x(k) \\ x_I(k) \end{bmatrix} = \begin{bmatrix} A & 0 \\ -C & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ x_I(k-1) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k-1)$$

$$+ \begin{bmatrix} 0 \\ 1 \end{bmatrix} r(k-1)$$

$$\Rightarrow \begin{bmatrix} 0.2299 & 0 \\ -1.4432 & 1 \end{bmatrix} \begin{bmatrix} x(k-1) \\ x_I(k-1) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k-1)$$

$$A_{AUG} = \begin{bmatrix} 0.2299 & 0 \\ -1.4432 & 1 \end{bmatrix}$$

$$B_{AUG} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(10) $t_s = 0.6$

$M_p = 0.2$

$$M_p = e^{-\pi\zeta / \sqrt{1-\zeta^2}}$$

$$0.2 = e^{-\pi\zeta / \sqrt{1-\zeta^2}}$$

$$\zeta = 0.4559$$

$$t_s = \frac{4}{\zeta\omega_n}$$

$$\zeta\omega_n = \frac{4}{0.6} = 6.67$$

$$\lambda = -\zeta\omega_n \pm \omega_d i$$
$$-6.67 \pm 13.01i$$

$$z^* = e^{\lambda T}$$
$$= 0.5701 \pm 0.434i$$

from MATLAB

$$K = \begin{bmatrix} 0.0896 & -0.2586 \end{bmatrix}$$

(v) $M_p = 0.2$

$t_s = 0.2$

$$M_p = e^{-\pi\zeta/\sqrt{1-\zeta^2}}$$

$$\zeta = 0.4559$$

$$t_s = \frac{4}{\zeta\omega_n}$$

$$\zeta\omega_n = \frac{4}{0.2} = 20$$

$$\lambda = -\zeta\omega_n \pm \omega_d i$$

$$= -20 \pm 39.039 i$$

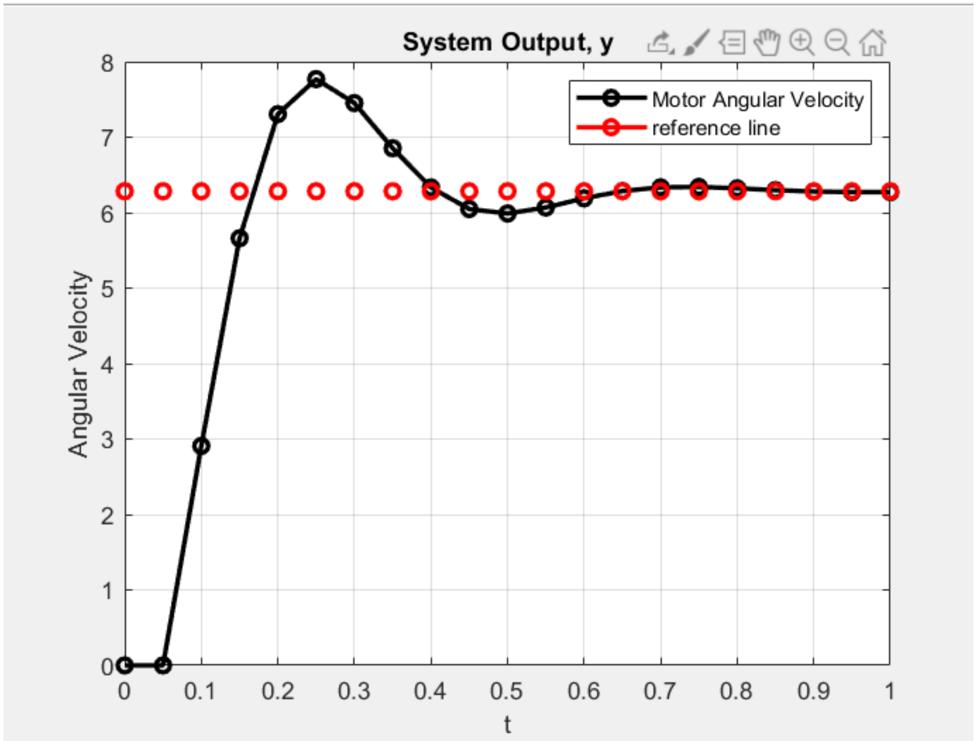
$$\beta^* = 0.3679$$

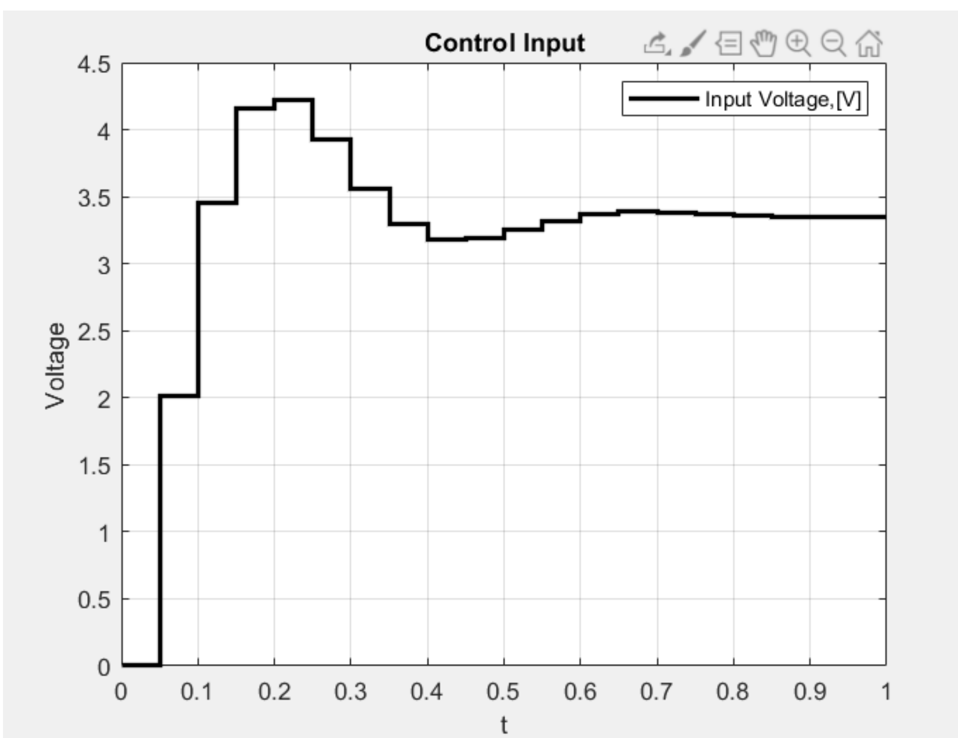
from MATLAB

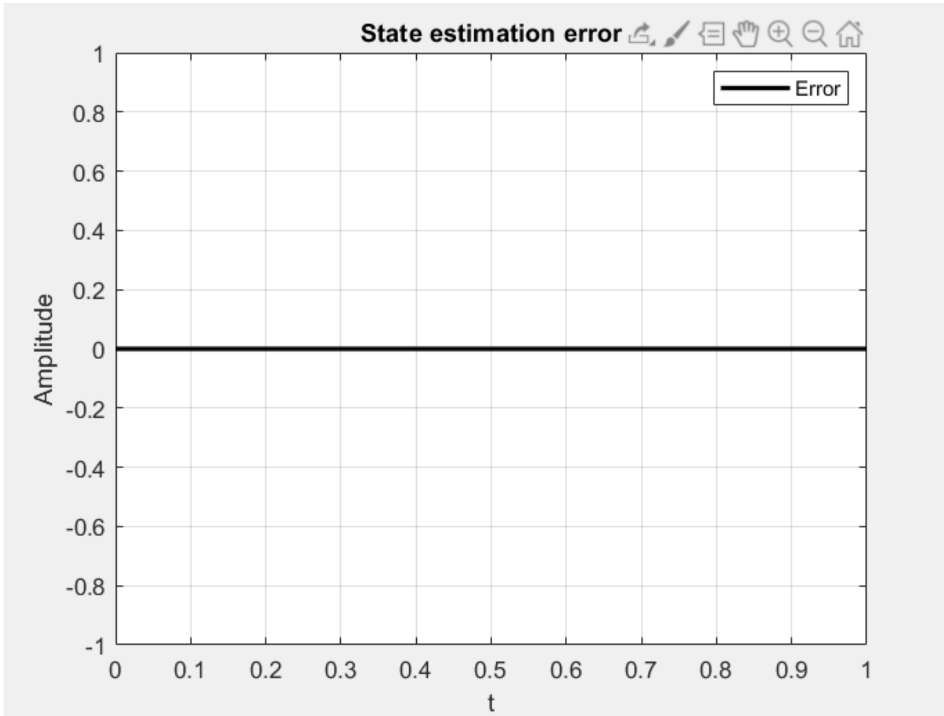
$$L = \begin{bmatrix} -0.0956 \end{bmatrix}$$

```
close all; clear all; clc;
%% declaring intial values
frequency = 20;
T = 1/frequency;
Tfinal = 1;
t = (0:T:Tfinal)';
N = length(t);
r = 2*pi;
A = 0.2299;
B = 1;
C = 1.4432;
Nvec = [(A-eye(size(A))) B;C 0]\[zeros(size(A));1];
N_x = Nvec(1);
%% reference state from reference signal
reference_state = N_x*r;
%% new augmentation matrices
A_AUG = [A zeros(size(A));-C 1];
B_AUG = [B;0];
%% K matrix
Ts = 0.6;
Mp = 0.2;
zeta = sqrt((log(Mp))^2/(pi^2+(log(Mp))^2));
zeta_omegaN = 4/Ts;
omega_N = zeta_omegaN/zeta;
omega_D = omega_N*sqrt(1-zeta^2);
Roots_cont = [-zeta_omegaN+omega_D*1i -zeta_omegaN-omega_D*1i];
Roots_disc = exp(Roots_cont*T);
z_star = [Roots_disc(1) Roots_disc(2)];
K_matrix = acker(A_AUG,B_AUG,[z_star]);
%% L matrix
Ts_estimate = 0.2;
zeta_omegaN_estimate = 4/Ts_estimate;
Roots_cont_estimate = [-zeta_omegaN_estimate];
Roots_disc_estimate = exp(Roots_cont_estimate*T);
beta_star = [Roots_disc_estimate];
L_matrix = acker(A', C',beta_star)';

%% Simulation
x = zeros(1,N);
x_hat = zeros(1,N);
y = zeros(1,N);
u = zeros(1,N);
e = zeros(1,N);
xI = zeros(1,N);
error = zeros(1,N);
x_hat(:,1) = zeros(1,1);
x(:,1) = zeros(1,1);
y(1) = C*x(:,1);
e(1) = r-y(1);
```

```
xI(1) = 0;
u(1) = 0;
for k=2:N
    x(k) = A*x(k-1)+B*u(k-1);
    y(k) = C*x(k);
    x_hat(k) = (A-L_matrix*C)*x_hat(k-1)+B*u(k-1)+L_matrix*y(k-1);
    e(k) = r-y(k);
    xI(k) = xI(k-1)+e(k-1);
    u(k) = K_matrix*([reference_state; 0]-[x_hat(k); xI(k)]);
    error(k) = x_hat(k) - x(k);
end

%% Plots
figure(1)
plot(t,y(1,:),'k-o', t,r,'r-o', 'linewi',2, 'MArkersize', 6)
grid on
hold on
title('System Output, y')
xlabel('t')
ylabel('Angular Velocity')
legend('Motor Angular Velocity', 'reference line')

figure(2)
stairs(t,u, 'k-', 'linewi',2)
grid on
hold on
title('Control Input')
xlabel('t')
ylabel('Voltage')
legend('Input Voltage,[V]')

figure(3)
plot(t,error,'k-','linewi',2)
ylim([-1 1])
grid on
title('State estimation error')
xlabel('t')
ylabel('Amplitude')
legend('Error')
grid on
```


System Output, y
Motor Angular Velocity
reference line
Angular Velocity
t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
1
2
3
4
5
6
7
8


Control Input
Input Voltage,[V]
Voltage
t
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5


State estimation error
Error
Amplitude
t
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1