Project-2

Aim: The goal of this part of the project is to design a tracking controller for the 6V DC motor using state-feedback control, with an integrator.

Constraints provided:

1) 0.6 < Settling time(ts) < 0.8
2) Overshoot (Mp) < 0.2
3) Rise time (tr) < 0.5
4) Input voltage signal u(Kt) cannot exceed 6v when the step reference of r(Kt) = 4pi rad/sec
5) Static position error constant: Kp = ∞
6) Static velocity error constant (Kv), should be made as large as possible to minimize error when tracking a ramp reference.

For the above constraints we need to find an open loop pole and also a pole has to be positioned at z = 1 for Kp = ∞.

TASK 1 HAS BEEN SOLVED ON NEXT PAGE

TASK1:

PROJECT-2

(1)

Transfer function $G(z) = \frac{Y(z)}{U(z)} = \frac{1.796}{z - 0.5134}$

[Q1] Put System in Controllable - Canonical state space form.

General $\Rightarrow \quad \frac{Y(z)}{U(z)} = \frac{b_0 z^n + b_1 z^{n-1} + \cdots b_n}{z^n + a_1 z^{n-1} + \cdots + a_n}$

In this question $z^n = z^1$ so A matrix is 1×1

$$\frac{Y(z)}{U(z)} = \frac{0z + 1.796}{z - 0.5134}$$

(b_0 = 0, b_1 = 1.796; a_1 = 0.5134)

So, $a_1 = -0.5134$

$b_0 = 0$

$b_1 = 1.796$

$A = [-a_1] = [0.5134]$, $B_{1\times1} = [1]$

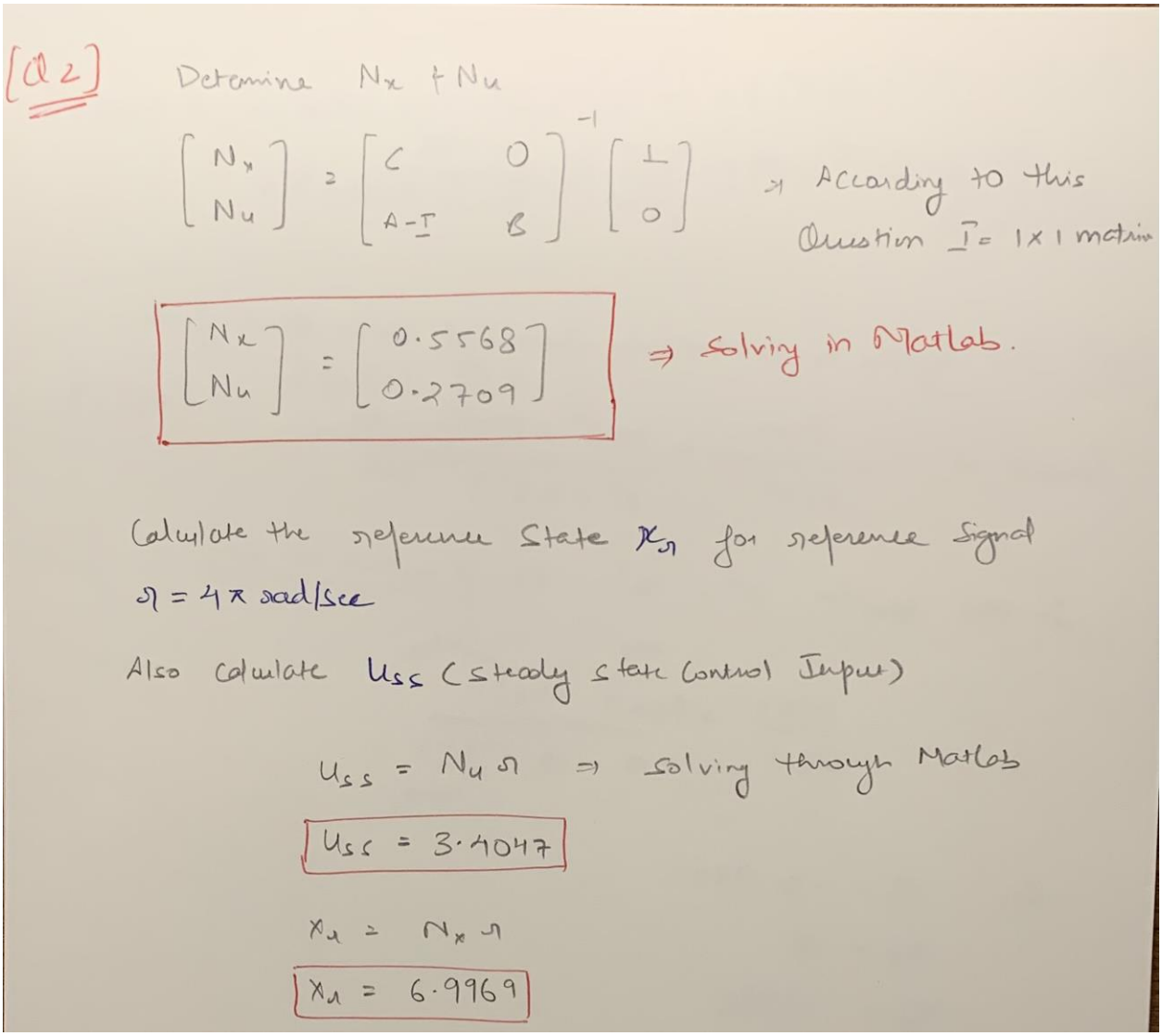
$C_{1\times1} = [b_1 - a_1 b_0] = [1.796 - (-0.5134)(0)]$

$= [1.796]$

$D_{1\times1} = b_0 = [0]$

$A = [0.5134]$
$B = [1]$
$C = [1.796]$
$D = [0]$

Task 2:

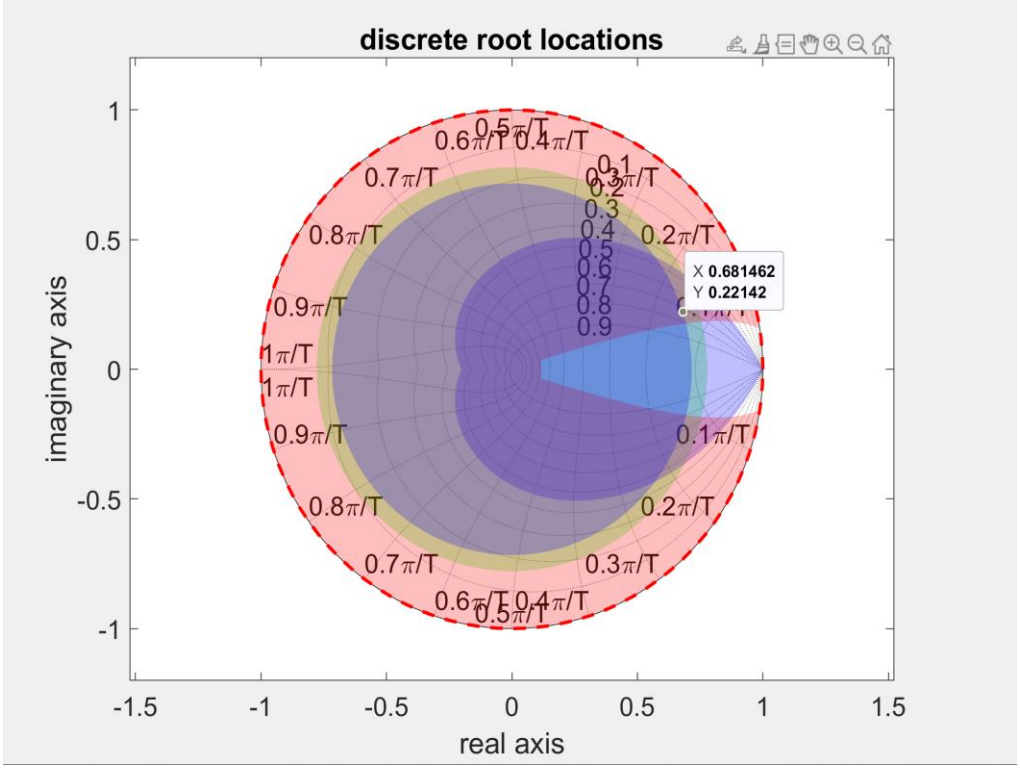


After calculating Nx, Nu, Uss and Xr, I used the given constraints to calculate the value of the open loop pole and to find whether the original controller has any steady state error or not.

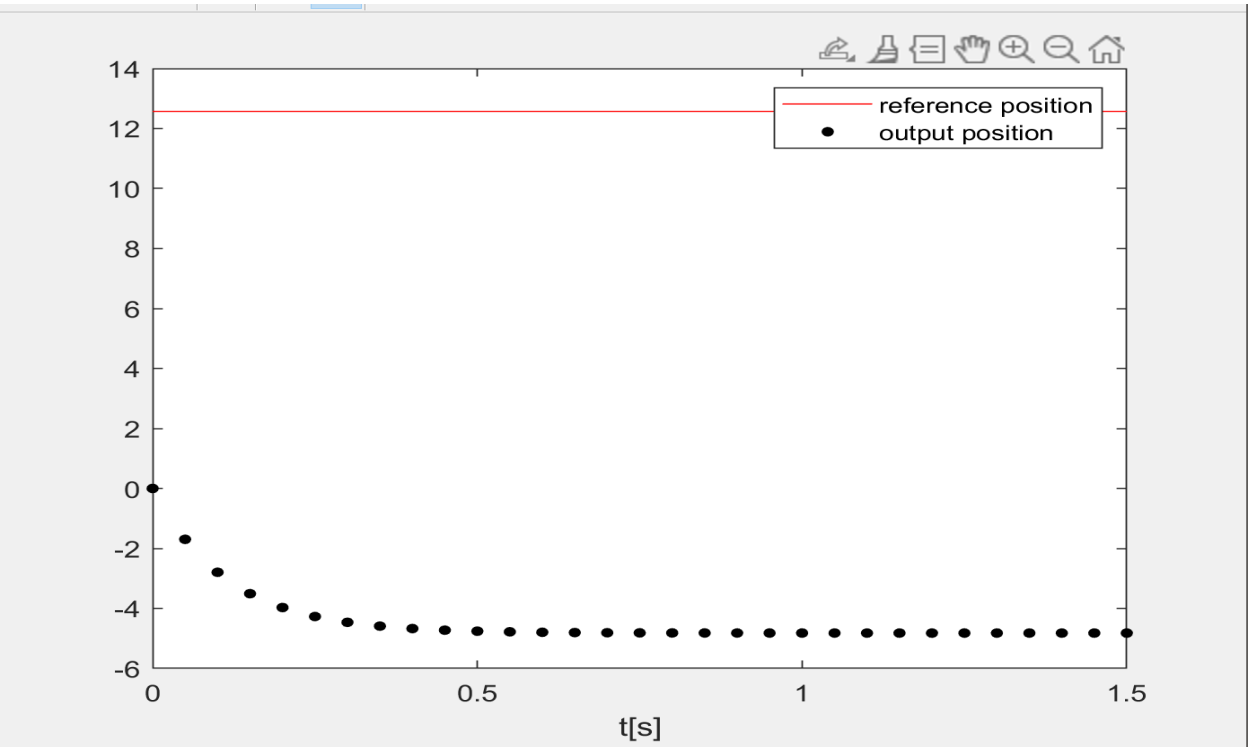
I founded the open loop pole by picking up the value from the intersection of rise time plot, settling time and maximum overshoot.

The value I picked initially for zstar = 0.618462 + 0.22142i, 0.618462 - 0.22142i

As shown below:



Once the value was picked the feedback gain matrix K was calculated and output and reference we plotted.



From the above plot we can see that there is a huge steady state error when a step reference of 4pi was given.

To solve the above issue we need to use TASK 3.

Task 3:

In this task we will augment the A matrix and the B matrix so that we can include the integral of the output error.

Using Matlab we can find the A_aug and B_aug using the formula”

A_aug = [A, zeros(1,1); -C, 1];
B_aug = [B; -D]

Where A,B,C,D are the controllable-canonical state-space form matrices.

```
A_aug =

    0.5134         0
   -1.7960    1.0000
```

```
B_aug =

     1
     0
```

On calculating the eigen values of the A_aug we can see that we hava an open loop pole at Z = 1.

```
>> eig(A_aug)

ans =

    1.0000
    0.5134
```

Task 4: Since the integrator has added a second state we need to add a new pole, and that is the benefit of the state space that we can add a new pole at any location we want to but in root locus the third pole cannot be controlled.

So after following the constraints as shown above I use the pole

zstar = 0.618462 + 0.22142i, 0.618462 - 0.22142i

and the third pole I used is the absolute value of the zstar, so that it doesn’t dominate the controller and make the system slower and also it shouldn’t be too close to the system that the system has overshoot.

```
>> abs(zstar)

ans =

    0.7165
```

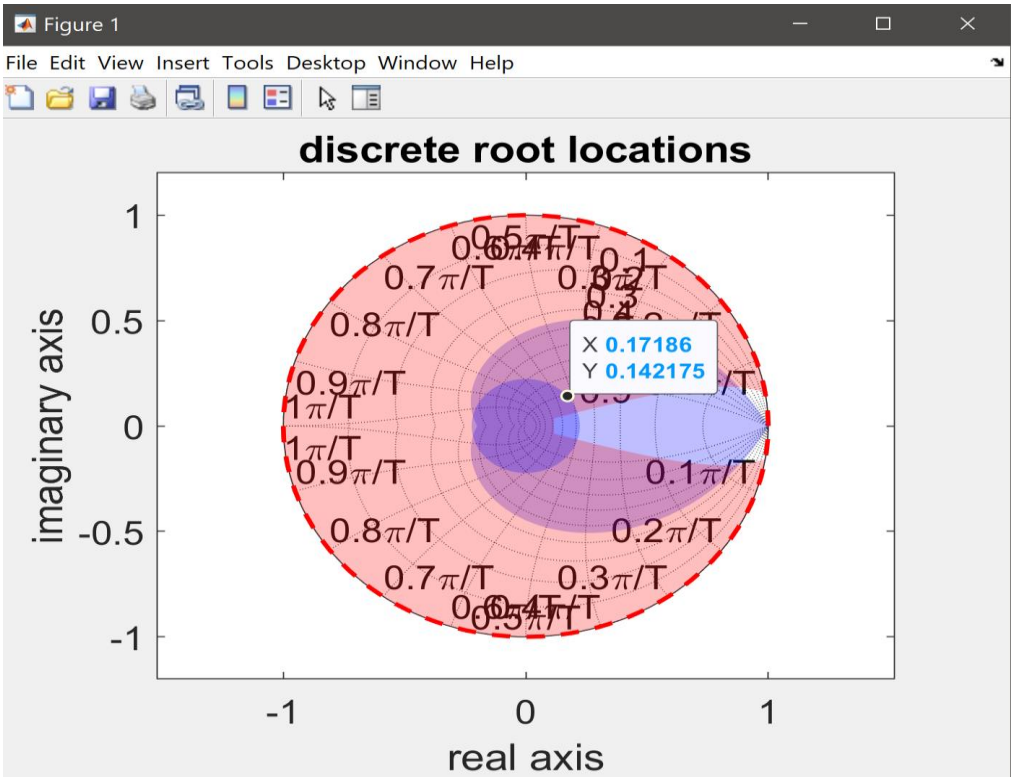
Once we are done with choosing the poles, augmenting matrices with the integral of output error and also fixing the open loop pole at Z=1, we were able to get the zero steady state error.

But in real world we never know the real state, we always have some noise that follows it and for that we need to add an estimator.

Task-5: For deriving an estimator we keep the settling time 4 to 6 times less than the original controller so that it doesn't dominate the system at all and keep giving the values of estimated state in advance to the system. Otherwise if it will settle at a lower rate, then it will not be able to feed the system with the estimated values.

So the settling time Ts for the estimator = 0.8/6 = 0.1333

And keeping all the other conditions same we were able to pick the required root for the estimator as shown below.



So, root for the estimator betastar is betastar = [0.1719+0.1422i];
Using acker command to get the estimator matrix, which is a scalar in this case
L = acker(A',C',betastar)';

```
L =

    0.1901
```

We can verify that the estimator has betastar component by using the Matlab command.

eig(A-L*C)

```
>>  eig(A-L*C)

ans =

    0.1719
```

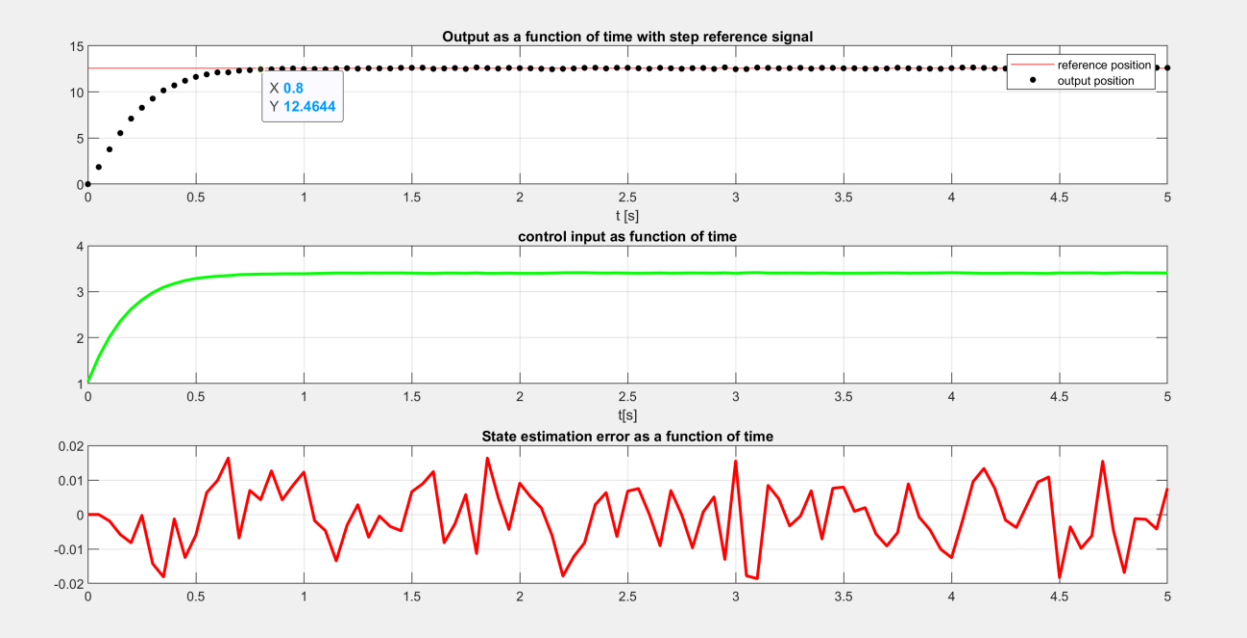
Task 6: Now finally after adding the estimator to our controller we can track the step reference for output, control input and state estimation error as function of time and also adding noise to it as asked in the task.

Matlab code shown below to get the required plots:

```
u(1) = -K_aug*([xhat(:,1); xI(1)] - [xr(:,1); 0]);
e(1) = r(1)-y(1);

for k = 2:N
    x(:,k) = A*x(:,k-1) + B*u(k-1);
    xhat(:,k) = A*xhat(:,k-1) + B*u(k-1)- L*(yhat(k-1)-y(k-1));
    xI(k) = xI(k-1) + e(k-1); %integrated state
    u(k) = -K_aug*([xhat(:,k); xI(k)] - [xr(:,k); 0]);
    y(k) = C*x(:,k) + D*u(k) + 0.05*randn(1);
    yhat(k) = C*xhat(:,k) + D*u(k);
    e(k) = r(k) - y(k);
end

figure
subplot(311),plot(t,r,'r-',t,y,'k.','markersize',14),grid on
title('Output as a function of time with step reference signal')
xlabel('t [s]'), legend('reference position','output position')
subplot(312),plot(t,u,'g-','linewi',2),grid on
title('control input as function of time')
xlabel('t[s]')
subplot(313),plot(t,xhat-x,'r-','linewi',2),grid on
title('State estimation error as a function of time')
```



Task 6 Observation:

First plot : From the first plot we can see that the settling time of the system is less than 0.8 and also greater than 0.6 (0.6<Ts<0.8)

Also, the system has no overshoot and rise time is less than 0.5 seconds as at 0.5 the system reached 11.6304 which is way beyond than 90% requirement for the rise time.

Second plot shows the voltage requirement of the motor and which is very low as compared to the limit which is 6v.

Third plot shows the shows the error between the estimated and the real state and the lower it is the better it is and mostly is stays between the limit of 0.01 to 0.02 and which is also good considering the noise as well.

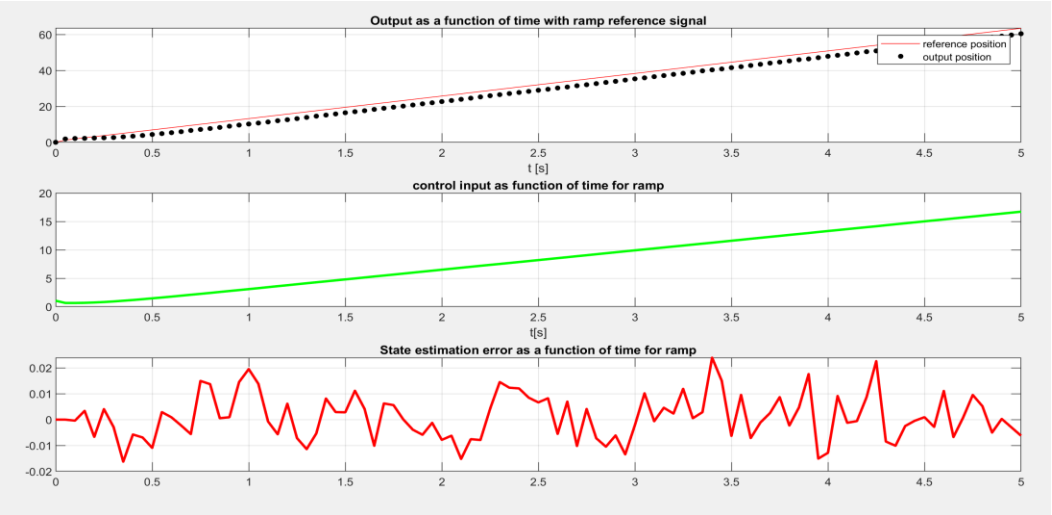
Task 7: This task focuses on tracking the ramp response.

Matlab code for the ramp response is shown below.

```
for k = 2:N
    r(k) = k*pi*4*T;
    xr(k) = Nx*r(k); %reference state for ramp reference
    uss = Nu*r(k); %steady state control input for ramp reference

    x(:,k) = A*x(:,k-1) + B*u(k-1);
    xhat(:,k) = A*xhat(:,k-1) + B*u(k-1)- L*(yhat(k-1)-y(k-1));
    xI(k) = xI(k-1) + e(k-1);
    u(k) = -K_aug*([xhat(:,k); xI(k)] - [xr(:,k); 0]);
    y(k) = C*x(:,k) + D*u(k) + 0.05*randn(1);
    yhat(k) = C*xhat(:,k) + D*u(k);
    e(k) = r(k) - y(k);
end

figure
subplot(311),plot(t,r,'r-',t,y,'k.','markersize',14),grid on
title('Output as a function of time with ramp reference signal')
xlabel('t [s]'), legend('reference position','output position')
subplot(312),plot(t,u,'g-','linewi',2),grid on
title('control input as function of time for ramp')
xlabel('t[s]')
subplot(313),plot(t,xhat-x,'r-','linewi',2),grid on
title('State estimation error as a function of time for ramp')
```



Observation of plot:

First plot: From the above plots we can see that the ramp response is tracked well and the steady state error is very less

Third Plot: the state estimation error stayed within the limit of 0.01 and just over 0.02, also with the noise of 0.05.

So, according to my observations and calculations this controller should work for the given system and under given constraints.