Project 1

Aim: To design a controller for the 6V DC motor for the given constraints.

Constraints:

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 pf r(Kt) = 2pi 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.

Steps to building the required controller:

1) I took the first 3 constraints and then plot the area from which I have to pick up the roots**.** Shown below**:**

   Using the settling time constraint

   The code I used for getting the settling time constraint is:

```
%% Settling time constraint for ts = 0.8

mag_ts1 = exp(-(4/ts1)*T); %% magnitude of our discrete roots for settling time condition
real_ts1 = mag_ts1*cos(theta);
imag_ts1 = mag_ts1*sin(theta);
%%above lines draw another circle with different magnitude
p2 = fill(real_ts1,imag_ts1,'g','LineStyle','none'); %%fill the shape with the colour
alpha(0.25); %%makes the shape transparent varies from 0 to 1


%% Settling time constraint for ts = 0.6

mag_ts0 = exp(-(4/ts0)*T); %% magnitude of our discrete roots for settling time condition
real_ts0 = mag_ts0*cos(theta);
imag_ts0 = mag_ts0*sin(theta);
%%above lines draw another circle with different magnitude
p3 = fill(real_ts0,imag_ts0,'b','LineStyle','none'); %%fill the shape with the colour
alpha(0.25); %%makes the shape transparent varies from 0 to 1

%end of settling time constraint = 0.6
```

   Calculations behind the code shown below:

Steps for meeting the settling time constraints:-

Settling time $\Rightarrow$ It is the time taken by the system to reach and to stay within $\pm 2\%$ of the reference signal.

$$t_s \text{ (settling time)} \approx 4/\delta\omega_n$$

So the constraint is that $t_s > 0.6$ sec

So $\Rightarrow$ $t_{s_0} < \frac{4}{\delta\omega_n}$ where $t_{s_0} = 0.6$ sec

(1) $\Leftarrow$ $\delta\omega_n < \frac{4}{t_{s_0}}$

$\delta \Rightarrow$ Damping Ratio

$\omega_n \Rightarrow$ Damped Natural frequency.

To make it in terms of setting as a constraint so that It can be used to find roots we simplify the above inequality

Magnitude: $|\lambda_d| = e^{-\delta\omega_n T}$

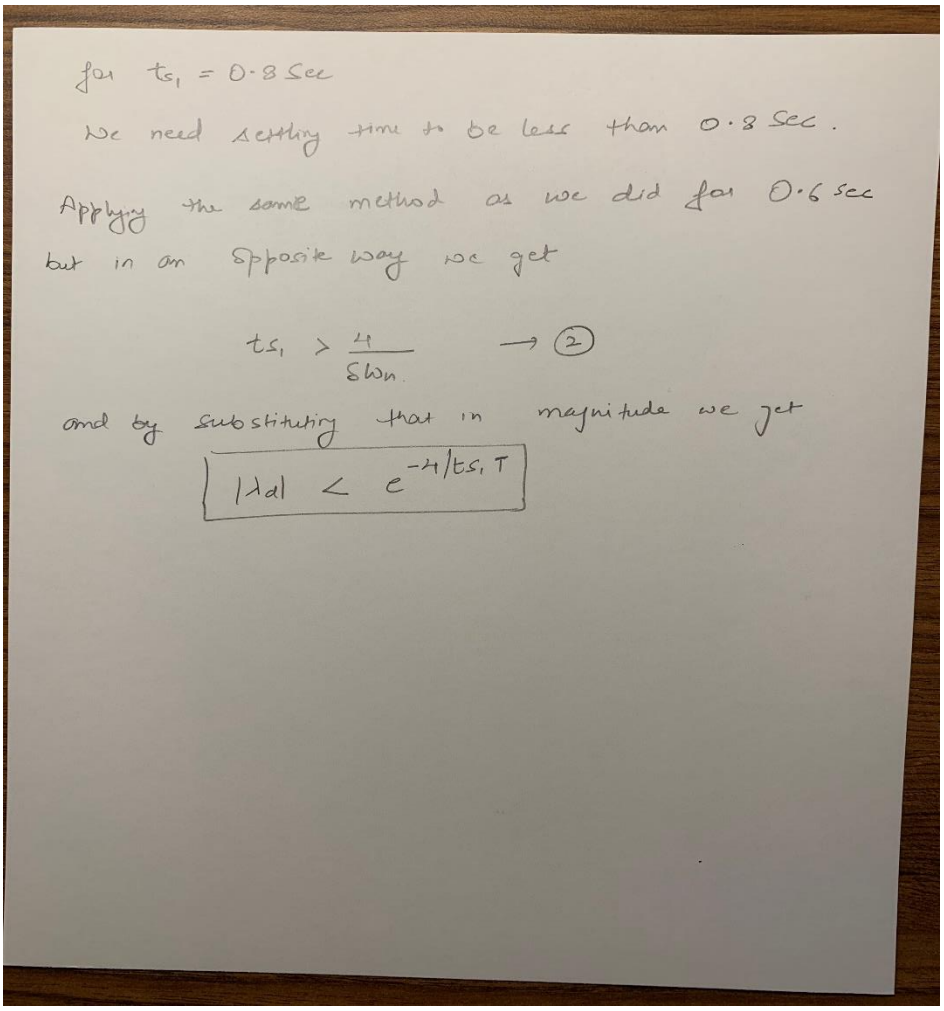
$\Rightarrow \lambda_d =$ discrete root

$T \Rightarrow$ Sampling Rate.

$\omega_d =$ Damped frequency

Substituting value from (1)

$$\boxed{|\lambda_d| > e^{-4/t_{s_0} T}} \Rightarrow \text{for } 0.6 \text{ sec.}$$

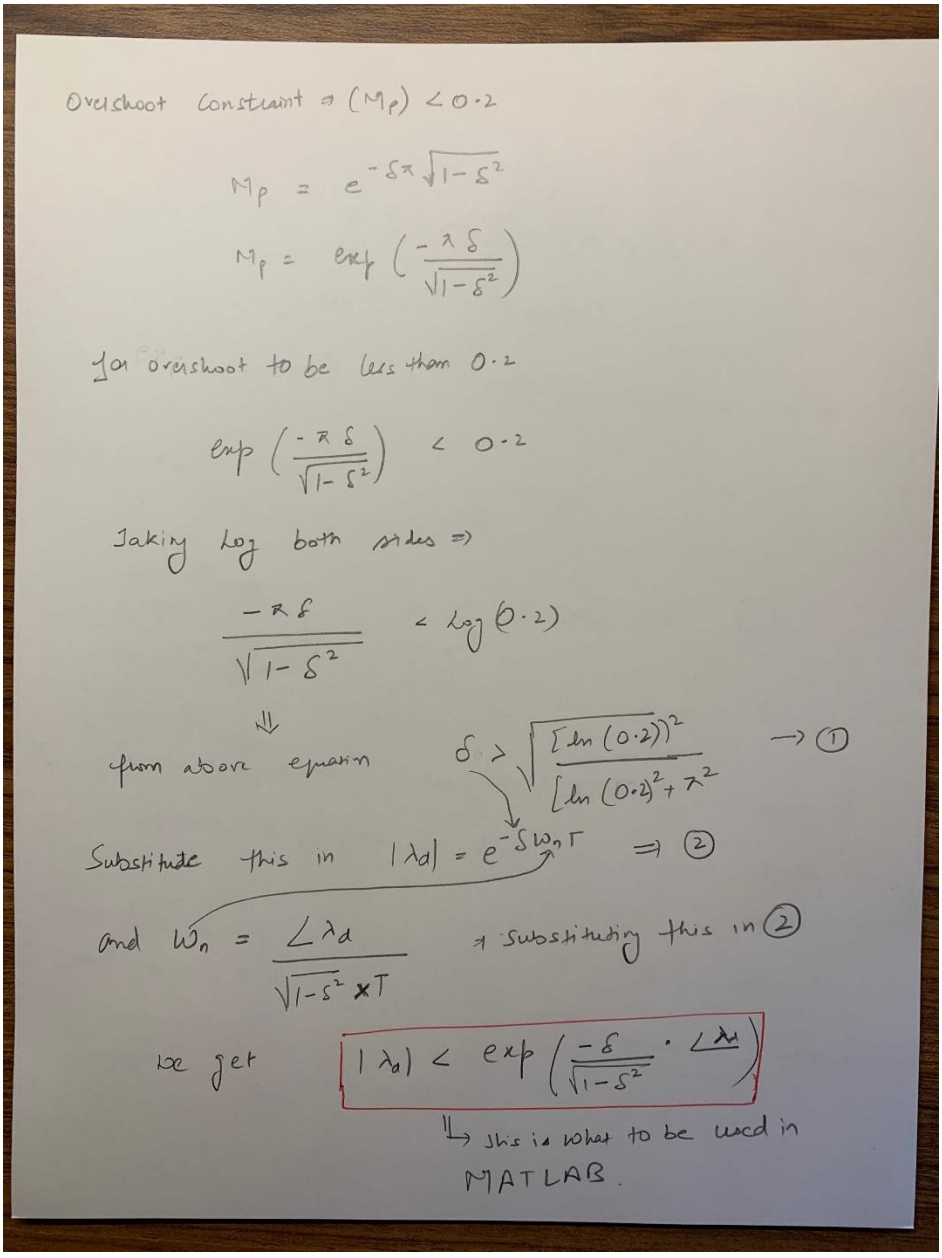


Using the Maximum overshoot constraint : where Overshoot (Mp) < 0.2

MATLAB CODE FOR THE OVERSHOOT:

```
%% constraint for maximum overshoot
theta2 = (0:pi/50:pi)';
zeta_Mp = sqrt(log(Mp0)^2/(pi^2 + log(Mp0)^2)); %%zeta_mp is eqivalent to zeta0 in the notes
mag_Mp = exp(-zeta_Mp/sqrt(1-zeta_Mp^2)*theta2); %%this magnitude is now a vector as theta value varies
real_Mp = mag_Mp.*cos(theta2);
imag_Mp = mag_Mp.*sin(theta2);
real_Mp_2 = [real_Mp; flipud(real_Mp)];
imag_Mp_2 = [imag_Mp; flipud(-imag_Mp)];
p4 = fill(real_Mp_2,imag_Mp_2,'b','LineStyle','none'); %%fill the shape with the colour
alpha(0.25)
```

Calculations behind the MATLAB code:



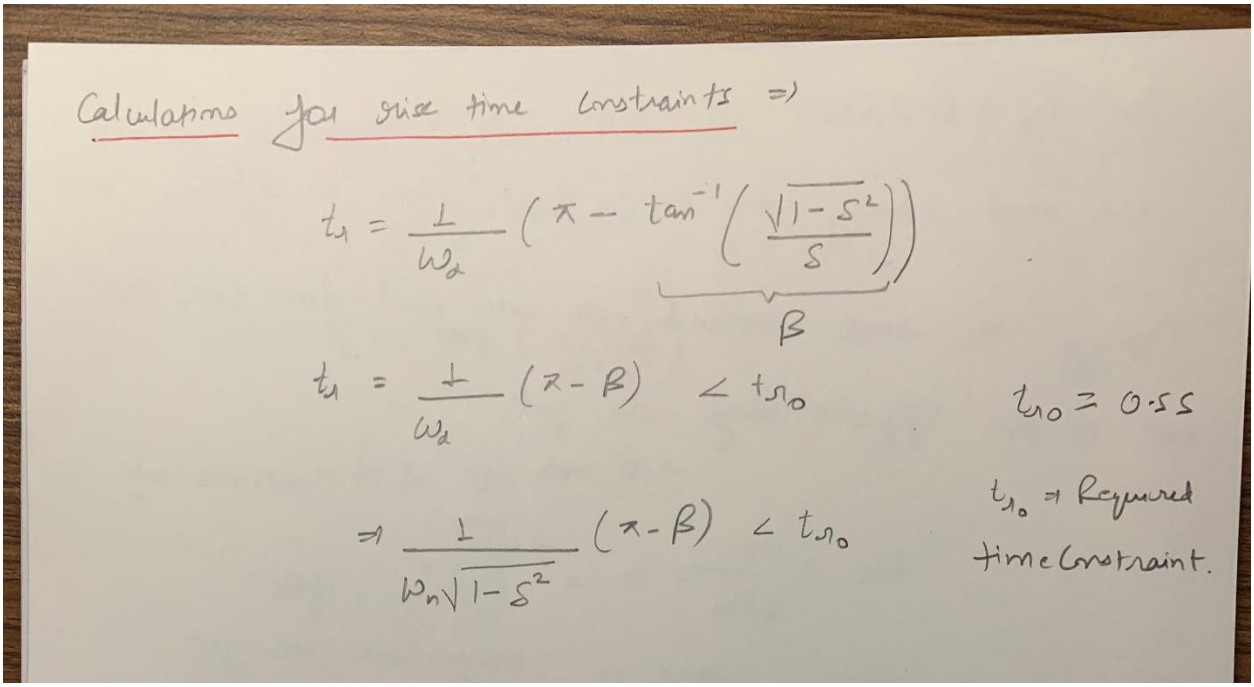
Using the rise time constraint:

Code for rise time constraint:

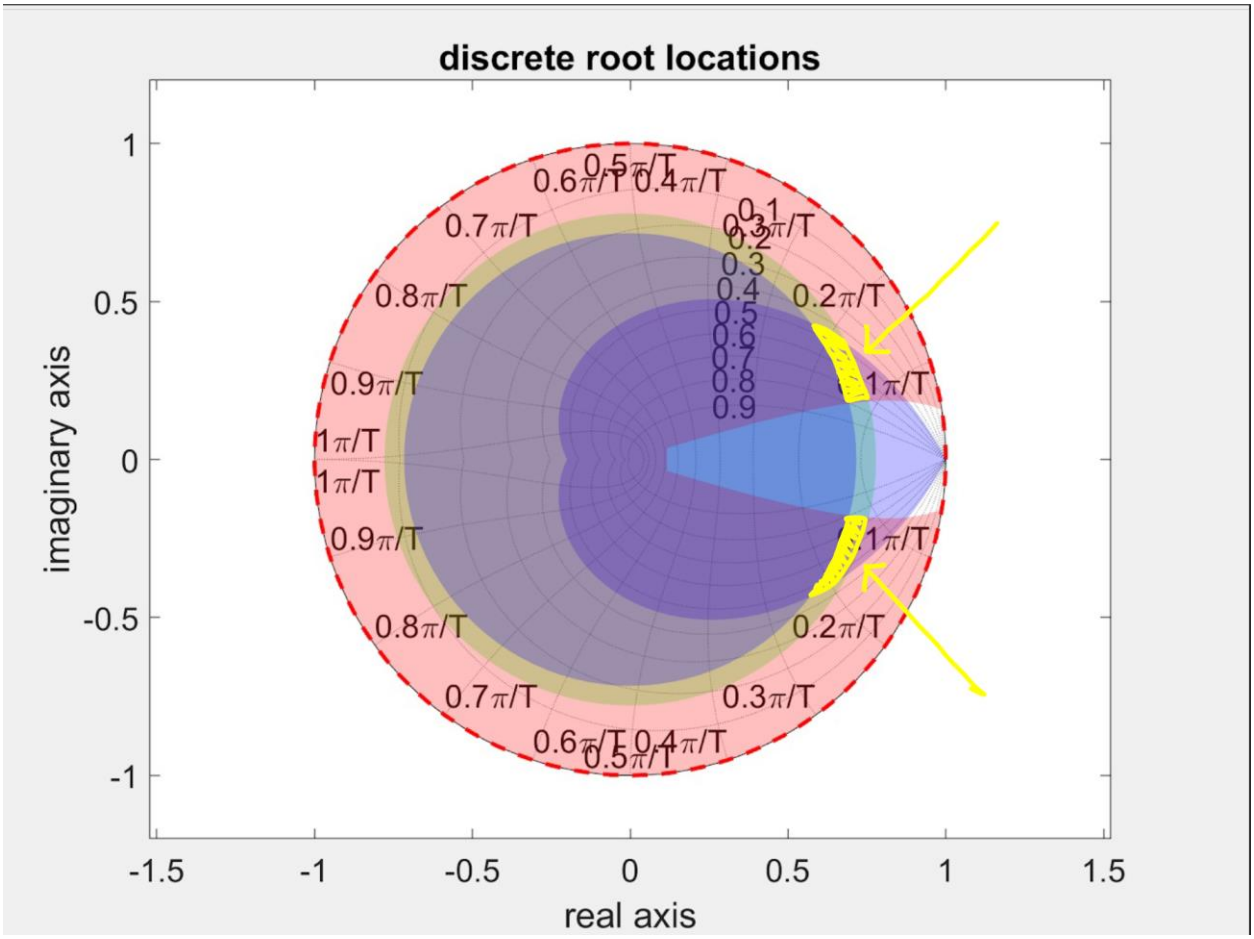
```
zeta_tr = linspace(0,0.99,501)';
beta = atan(sqrt(1-zeta_tr.^2)./zeta_tr);
omegan_tr = (pi-beta)./(tr0*sqrt(1-zeta_tr.^2));
omegad_tr = omegan_tr.*sqrt(1-zeta_tr.^2);
```

```
mag_tr = exp(-zeta_tr.*omegan_tr*T);
ang_tr = omegad_tr*T;
real_tr = mag_tr.*cos(ang_tr);
imag_tr = mag_tr.*sin(ang_tr);

real_tr_2 = [flipud(real_tr); 1*cos((ang_tr(1):0.1:pi)')];
imag_tr_2 = [flipud(imag_tr); 1*sin((ang_tr(1):0.1:pi)')];

real_tr_3 = [real_tr_2; flipud(real_tr_2)];
imag_tr_3 = [imag_tr_2; flipud(-imag_tr_2)];

p5 = fill(real_tr_3,imag_tr_3,'r-','LineStyle','none');
alpha(0.25);
```

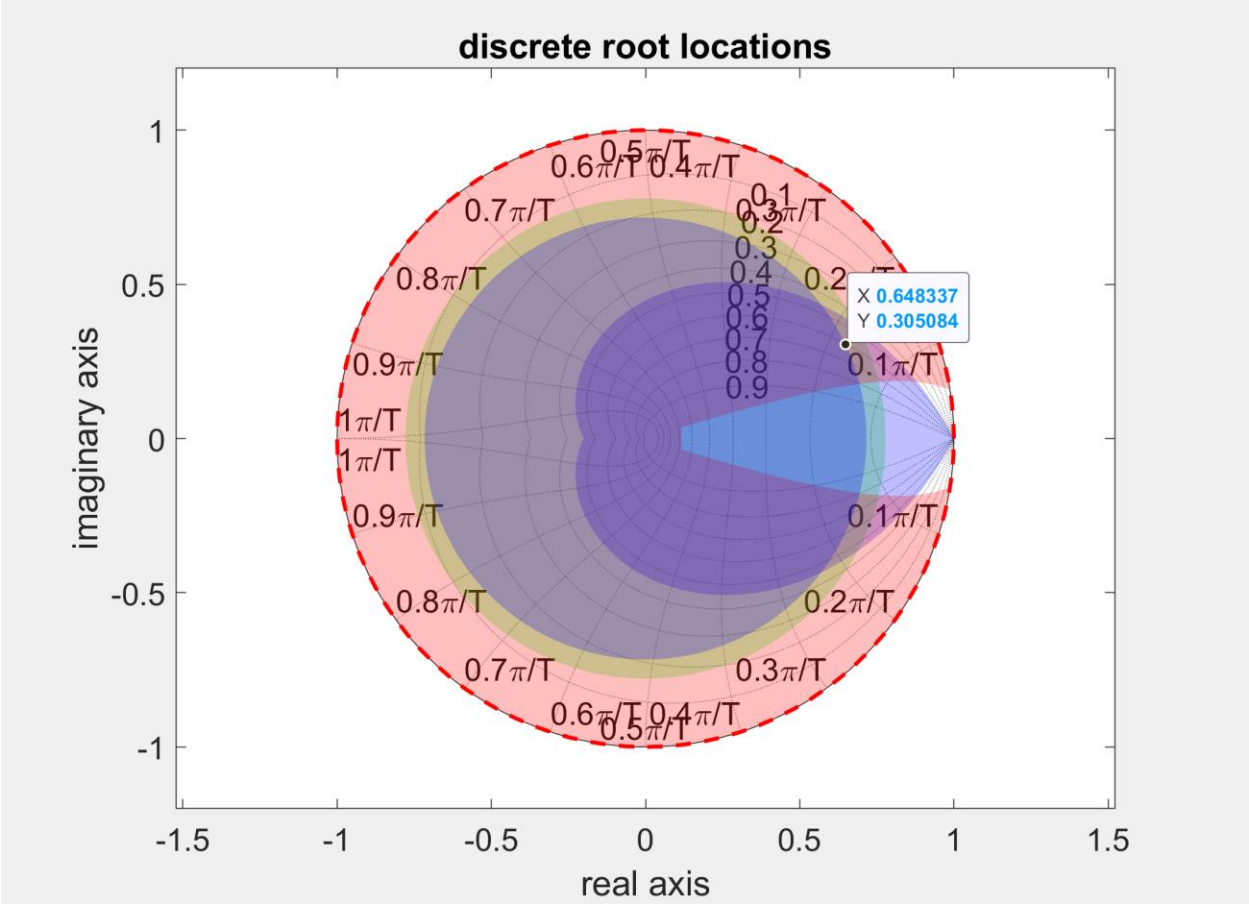
Calculations behind the MATLAB code:



The area after applying the constraints, we got an area from which we can pick the root that would satisfy the rise time, maximum overshoot and settling time.



2) Once I was done with finding out the area from which I have to pick zstar (root that can give me a suitable response, I picked a root on the boundary of settling time = 0.6sec so that even if my system goes a little off and is not able to settle till 0.6, it is settling till 0.8 for sure.
Zstar that I took is shown below.



2.1 The next step we need to find is the find poles and zeros for the controller which meets rest of the remaining three criteria
The transfer function we have is:

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

The form of the controller we need:

$$D(z) = \mathbb{K}\frac{z - z_1}{(z - p_1)(z - p_2)}$$

So in the above controller we can see that we need two poles and one zero, according to our z = zstar, that should match the other requirements of Kp, Kv, and Control input voltage.

2.2 To start with we need Kp = ∞, and Kp can only infinity when we have a pole at z = 1, since the system of type 1 and pole at z = 1 can satisfy the requirement.

So, now since we found one pole, we have our controller as
$D(z) = k\,(z - z1)/[(z-p1)(z-p2)]$, with $p2 = 1$.

To make the calculations and coding easy let's shift (z-p2) to combine with the plant transfer function as this won't change the final response of the system.

So our updated plant transfer function is:

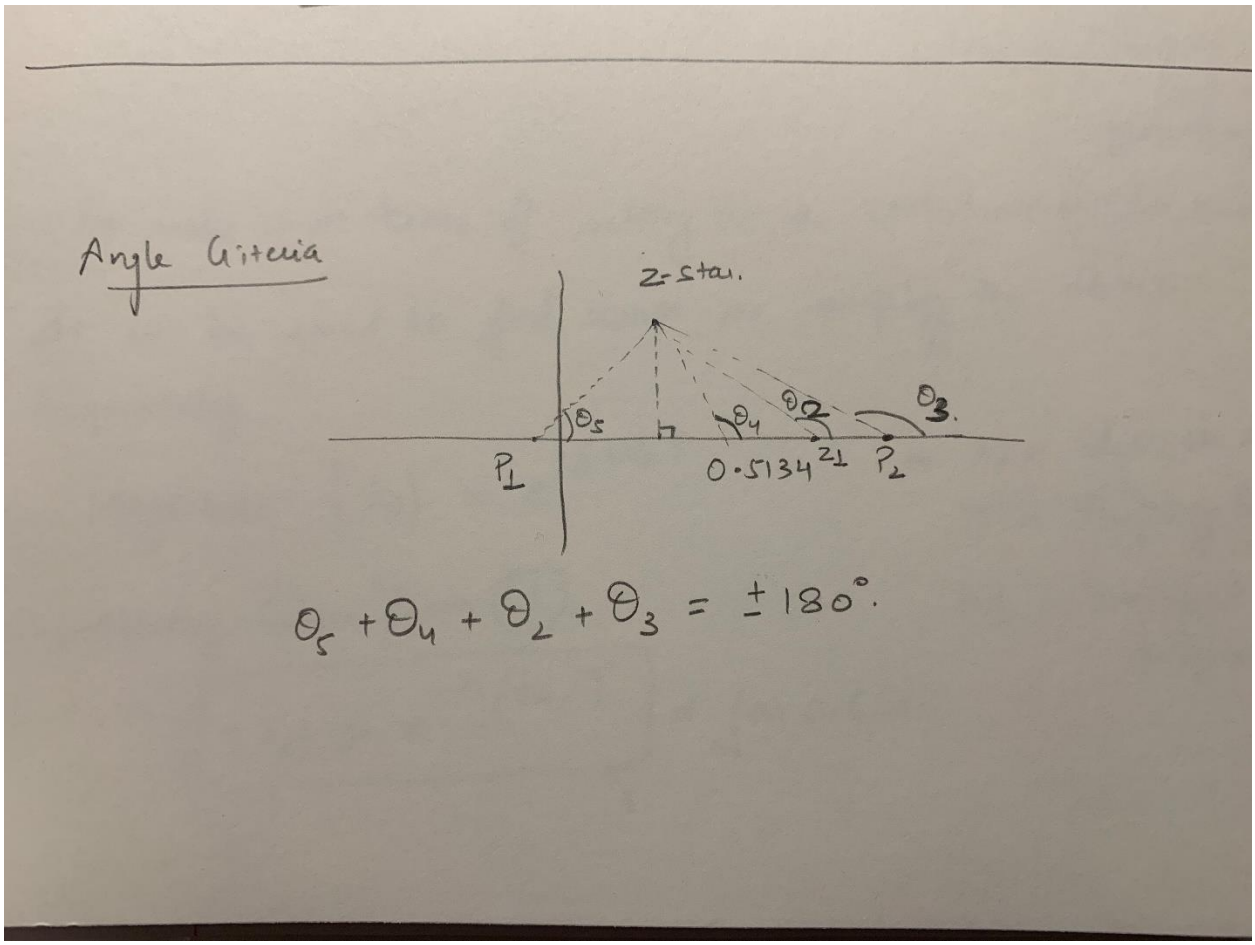
$G(z) = 1.796/[(z-0.5134)(z-1)]$

The undated controller we are left with to work on is:
$D(z) = k\,(z - z1)/(z-p1)$

2.3 Now we will apply magnitude and angle criteria to find the poles, zeros and gain of the controller.
According to the angle criteria which we use to find the poles and zeros according to the zstar, the sum of the angles of poles and zeros made with the conjugate or imaginary root of zstar should be equal to +180 degrees or -180 degrees, where the

Angle contribution of poles is negative and angle contribution of zeros is positive.
We use the geometric formula for tangent calculating all the axis from x – axis and taking clockwise angle as the angle measuring criteria.

Calculations for angle criteria shown below:



2.4 And using the magnitude criteria we get the value of k(gain).

According to this the absolute value of
G(zstar)*D(zstar) = 1
Where D(zstar) = K * D_no_k(zstar)
Substituting that we get the value for k = 1/abs(G(zstar) * D_no_K(zstar))

2.5 So, the points that satisfy the angle and magnitude criteria can be many and we can run a for loop to change the poles, zeros and values of gain to get the number of controllers and test their output to get the correct response.

After we get atleast 10 controllers within the allowed limit of magnitude and angle criteria the closed loop poles are calculated and those poles which we already know won't give us the correct response are rejected (like the closed loop poles on negative axis and the poles that have a magnitude greater than 1 and will make the system go unstable)

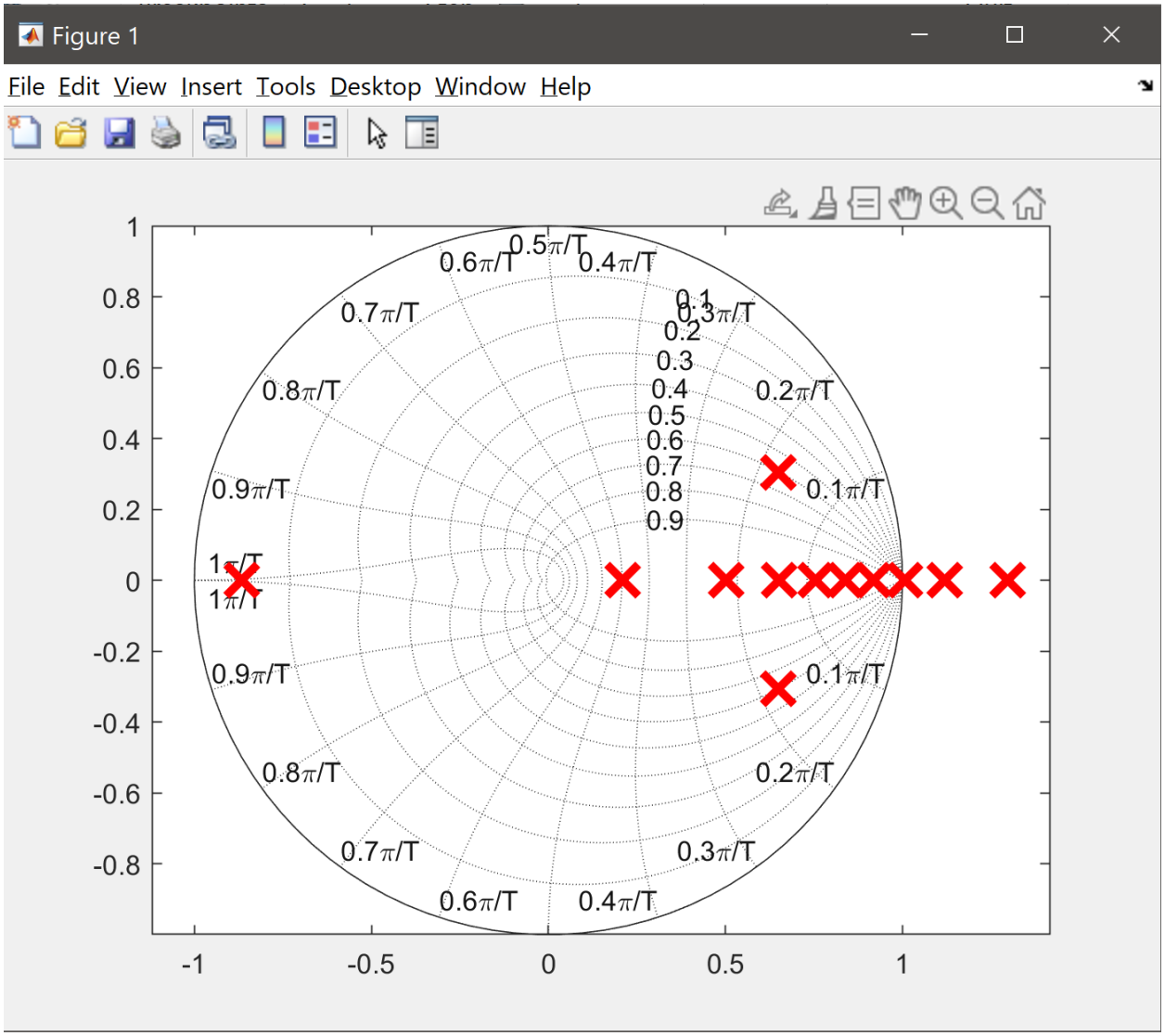
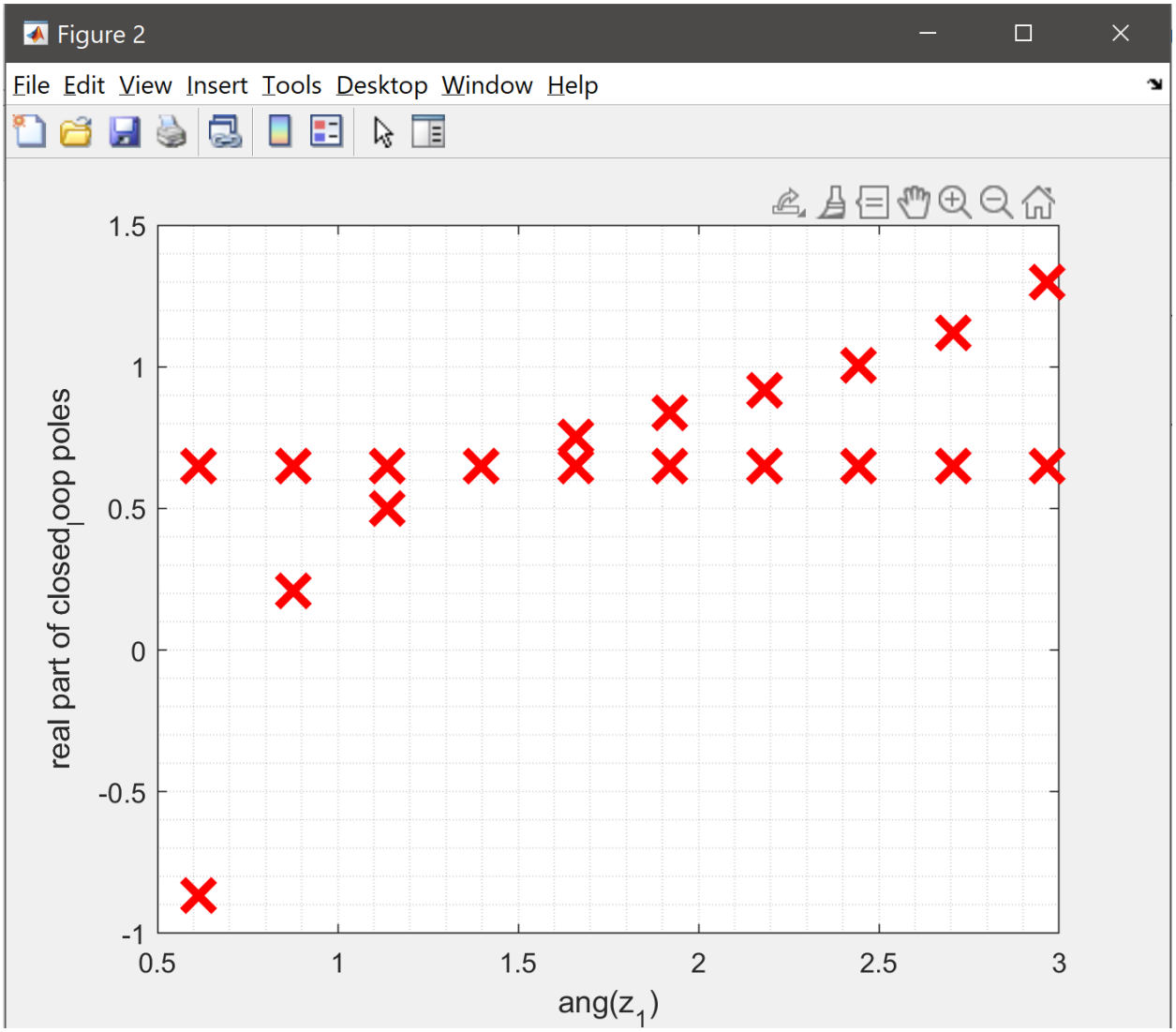
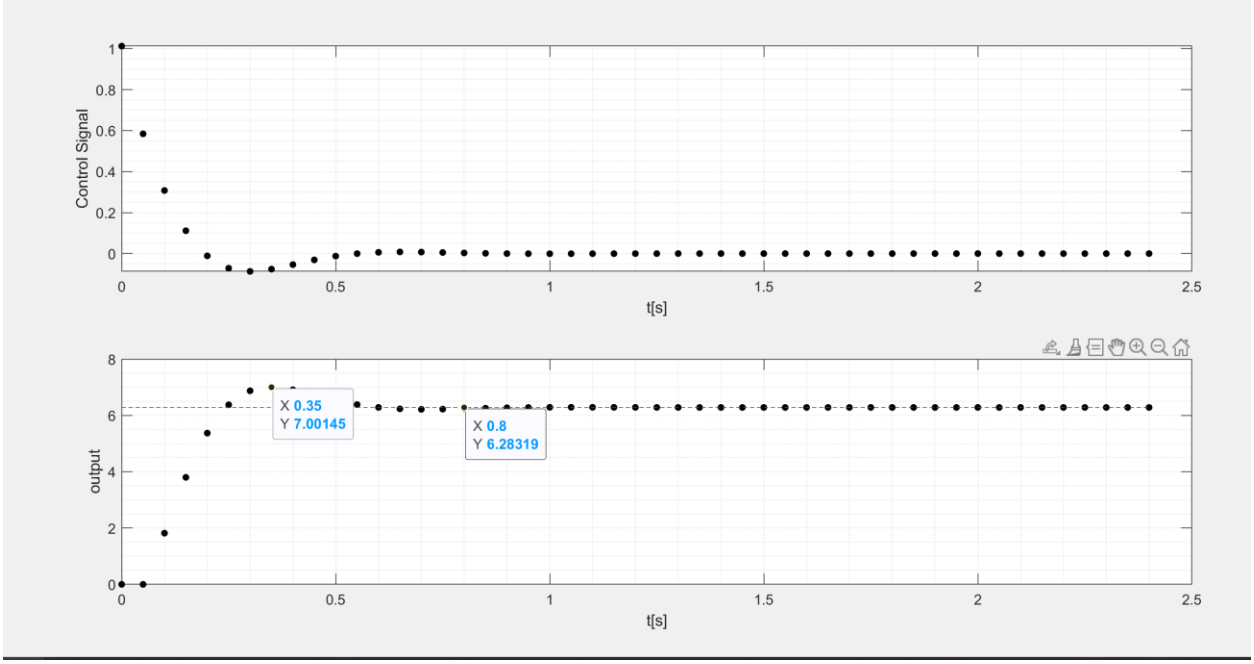


Figure 1
File Edit View Insert Tools Desktop Window Help
0.5π/T
0.6π/T
0.4π/T
0.7π/T
0.3π/T
0.8π/T
0.2π/T
0.9π/T
0.1π/T
1π/T
0.9π/T
0.1π/T
0.8π/T
0.2π/T
0.7π/T
0.3π/T
0.6π/T
0.4π/T
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-0.5
0
0.5
1



From the above two figures we can come to the conclusion that our angle range of zero z1 can be from 0.6 – 1.7 and based on the the poles angle are also restricted to give us the best response.

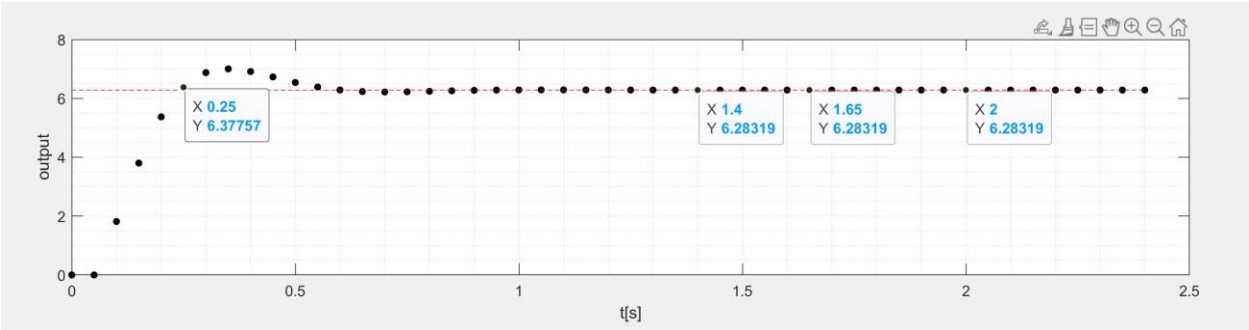
3) Now we need to check the response of our controller and also the control input voltage which should not exceed 6volts. And also keeping the step reference = 2*pi



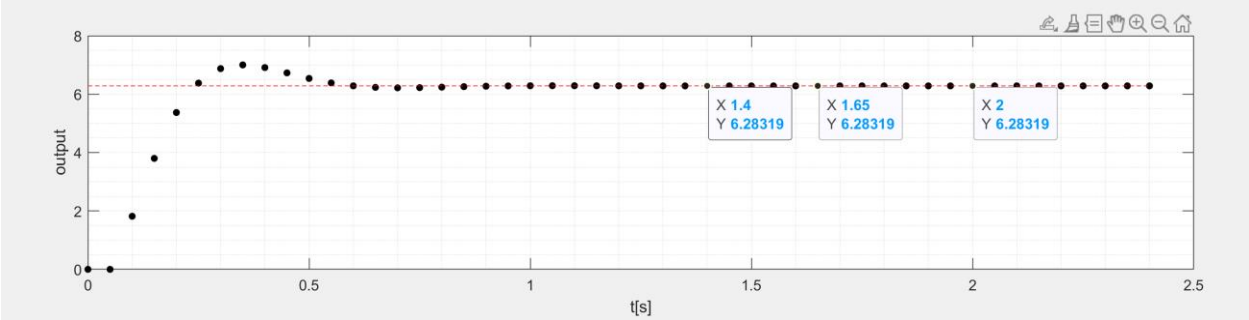
From the output figure it can be seen that the **maximum overshoot is 7.00145** which is well under the required limit of 20%

Also, from the output figure we can see the **settling time is less than 0.8 sec,** which is also within the required limit of 0.6 and 0.8.

The control signal graph shows that **the maximum voltage requirement when the reference is kept at 2pi is around 1 volt**, which is very well below the limit of 6 volts



The graph above shows that the rise time for the system is 0.25 sec, however the requirement was of less than 0.5 secs which is also satisfied.



The graph above shows that the steady error for the step output is zero as the reference and the output line matches exactly, **there Ess = 0, that means Kp is infinity as Ess = (2pi/(1+Kp))**, this requirement is also satsfied.

So summarizing the above points:

3.1 Maximum overshoot (Mp) = 0.11431 which is less than 20%

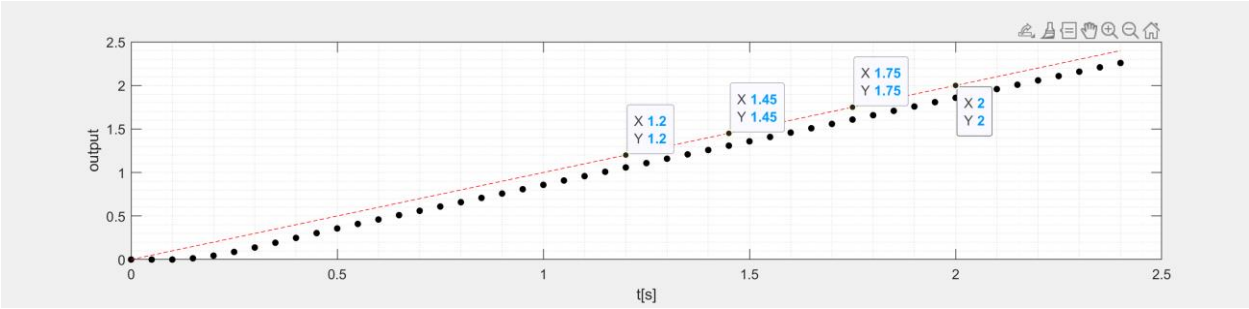
3.2 Settling time (ts) < 0.8 sec

3.3 Rise time = 0.25 sec

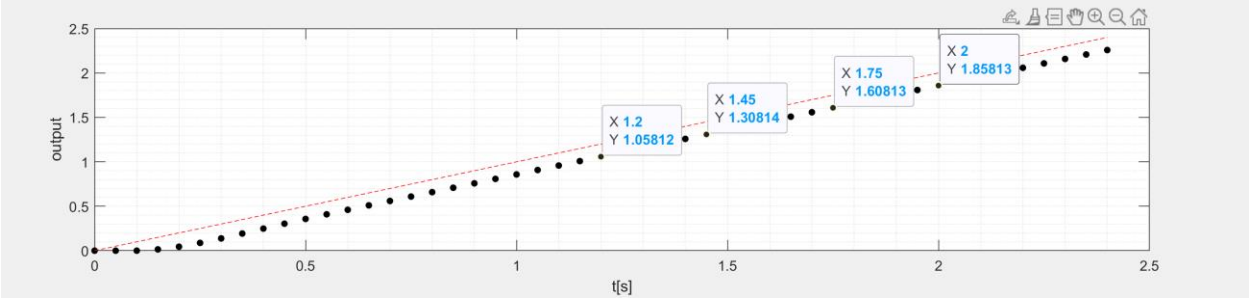
3.4 Kp is infinity

3.5 Control input voltage is around 1 volt.

4)



The above figure the red dotted line shows the reference for ramp response.

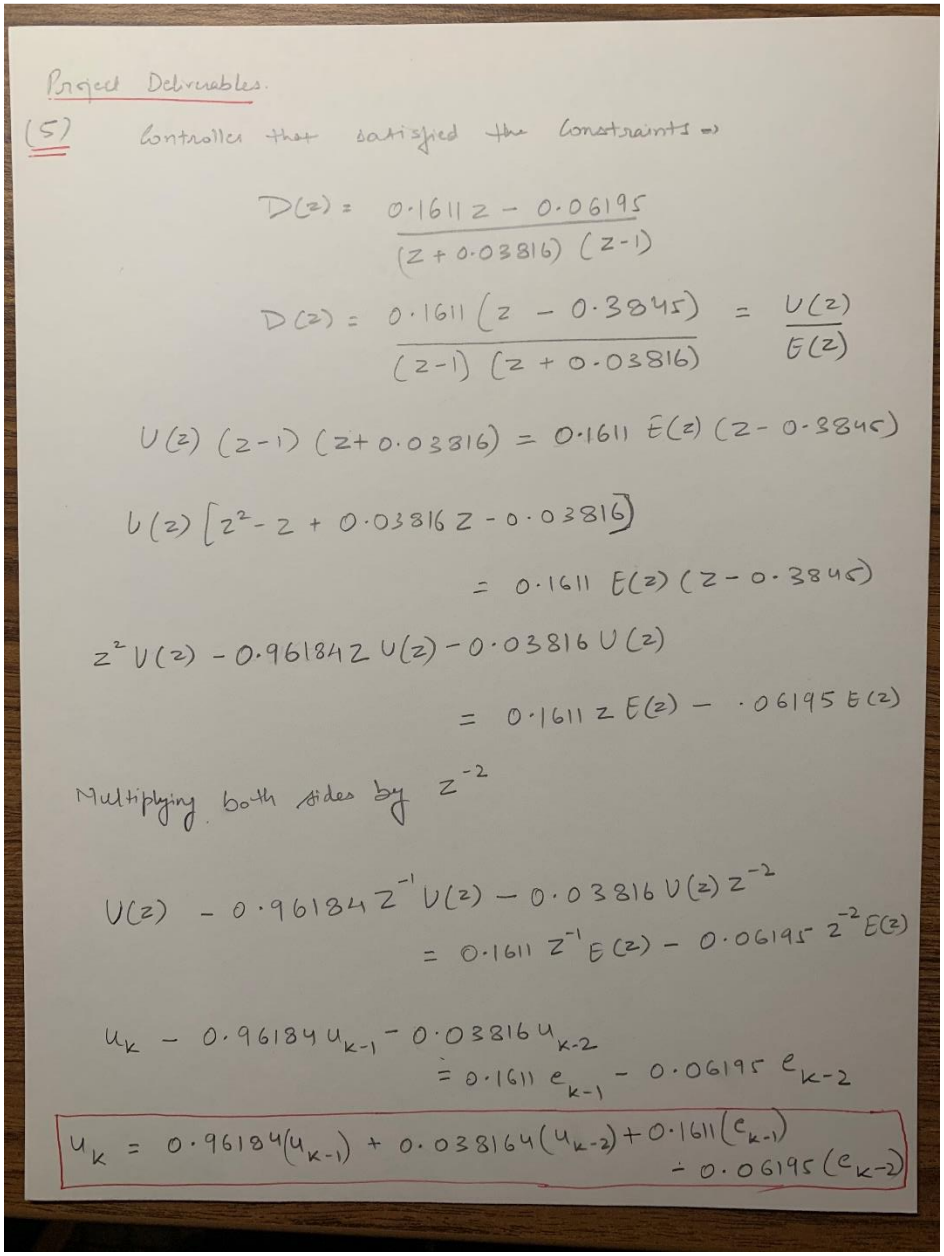


The above figure shows the response of the system with ramp response.

From the above two figures we can calculate the Kv (static velocity error constant) :
The steady state error for the 4 points marked above are:
0.14179, 0.14186, 0,14187, 0.14187

So Kv is 1/Ess = 7.0487 (for Ess = 0.14187 as the system seems to be stable at this value of error)

5) I used the calculation below to get the final equation that can be put into the microcontroller to test my controller.



Project Deliverables.

(5) Controller that satisfied the constraints =>

$$D(z) = \frac{0.1611z - 0.06195}{(z+0.03816)(z-1)}$$

$$D(z) = \frac{0.1611(z - 0.3845)}{(z-1)(z+0.03816)} = \frac{U(z)}{E(z)}$$

$$U(z)(z-1)(z+0.03816) = 0.1611\,E(z)(z-0.3845)$$

$$U(z)[z^2 - z + 0.03816z - 0.03816]$$
$$= 0.1611\,E(z)(z-0.3845)$$

$$z^2U(z) - 0.961842\,U(z) - 0.03816\,U(z)$$
$$= 0.1611z\,E(z) - .06195\,E(z)$$

Multiplying both sides by z^{-2}

$$U(z) - 0.961842z^{-1}U(z) - 0.03816\,U(z)z^{-2}$$
$$= 0.1611z^{-1}E(z) - 0.06195z^{-2}E(z)$$

$$u_k - 0.961842u_{k-1} - 0.03816u_{k-2}$$
$$= 0.1611e_{k-1} - 0.06195e_{k-2}$$

$$u_k = 0.96184(u_{k-1}) + 0.038164(u_{k-2}) + 0.1611(e_{k-1}) - 0.06195(e_{k-2})$$