EE 208

Control Engineering Lab

Experiment-7: State feedback features in digital domain.

Group Number: 20

Vedansh: 2020EEB1218

Aniket Arya: 2020EEB1157

Chirag Rathi: 2020EEB1165

Professor: Dr. S. Roy

Date: 13/03/2022

OBJECTIVE: -

To design state feedback for a given digital state-space system, so as to realise performance specifications for different applications.

Given: -

The discretised state open-loop space representation for an armature-controlled DC motor is described by:

$$F = \begin{bmatrix} 1.0 & 0.1 & 0.0 \\ 0.0 & 0.9995 & 0.0095 \\ 0.0 & -0.0947 & 0.8954 \end{bmatrix} \qquad G = \begin{bmatrix} 1.622 \times 10^{-6} \\ 4.821 \times 10^{-4} \\ 9.468 \times 10^{-2} \end{bmatrix}$$

The corresponding sampling time is 0.01s.
The three applications for which the state feedback matrix need to be designed for which the closed loop discrete time eigenvalue specifications are:
1) 0.1, 0.4 ± 0.4j
2) 0.4, 0.6 ± 0.33j
3) All values at origin
Initial state vector of the system is:

$$x_0 = [1 \quad 1 \quad 1]^T$$

The transfer functions corresponding to original system without feedback gain matrix are:

```
1.622e-06 z^2 + 4.514e-05 z + 4.823e-05
---------------------------------------
  z^3 - 2.895 z^2 + 2.791 z - 0.8959
```

```
0.0004821 z + 0.0004678
-----------------------
z^2 - 1.895 z + 0.8959
```

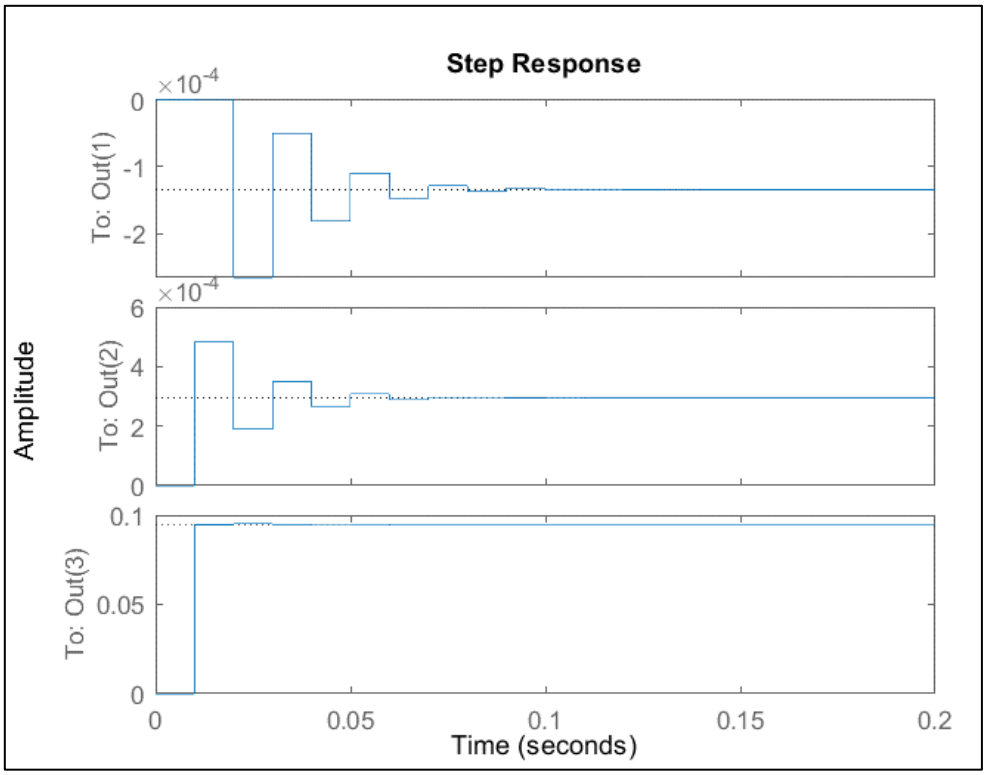
```
 0.09468 z - 0.09468
----------------------
z^2 - 1.895 z + 0.8959
```

The **eigenvalue**s of the **open-loop system** are {**1.0, 0.9855, 0.9054**}. Given a set of eigenvalues, it is clear that the open-loop system is slightly stable at z = 1.0 due to the presence of poles.

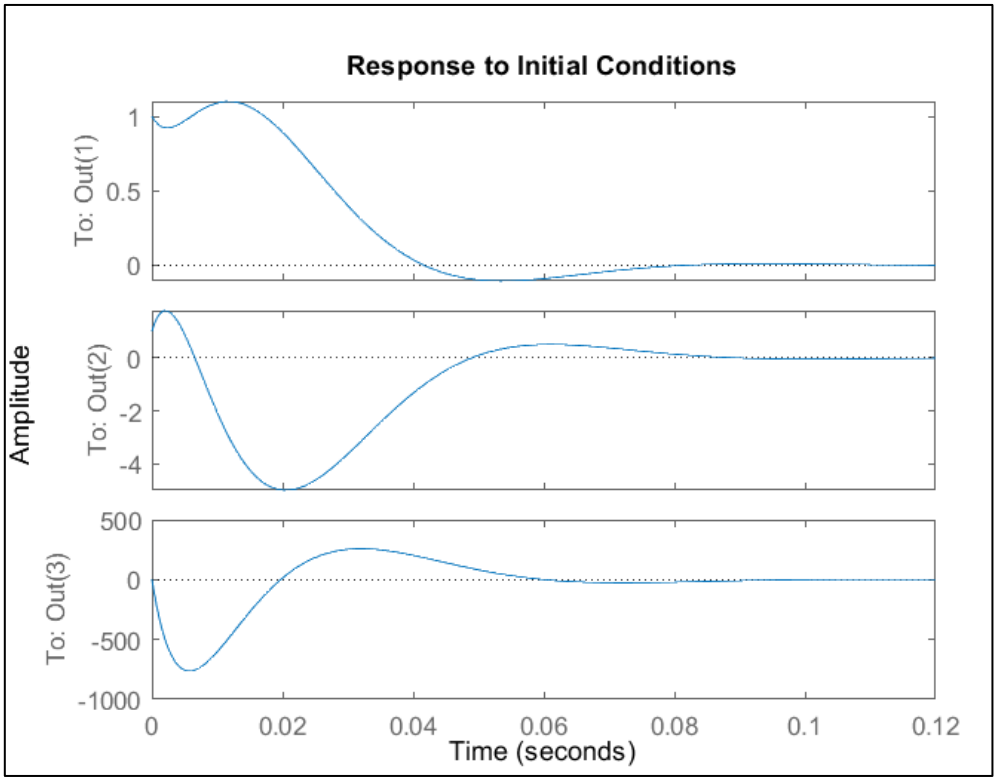
The third application makes us place all the closed loop poles to be placed at 0, but as placing pole at 0 in z domains is equivalent to the placing s at infinity. We applied Ackerman approximation to place multiple poles at 0 in z domain.

Observations:

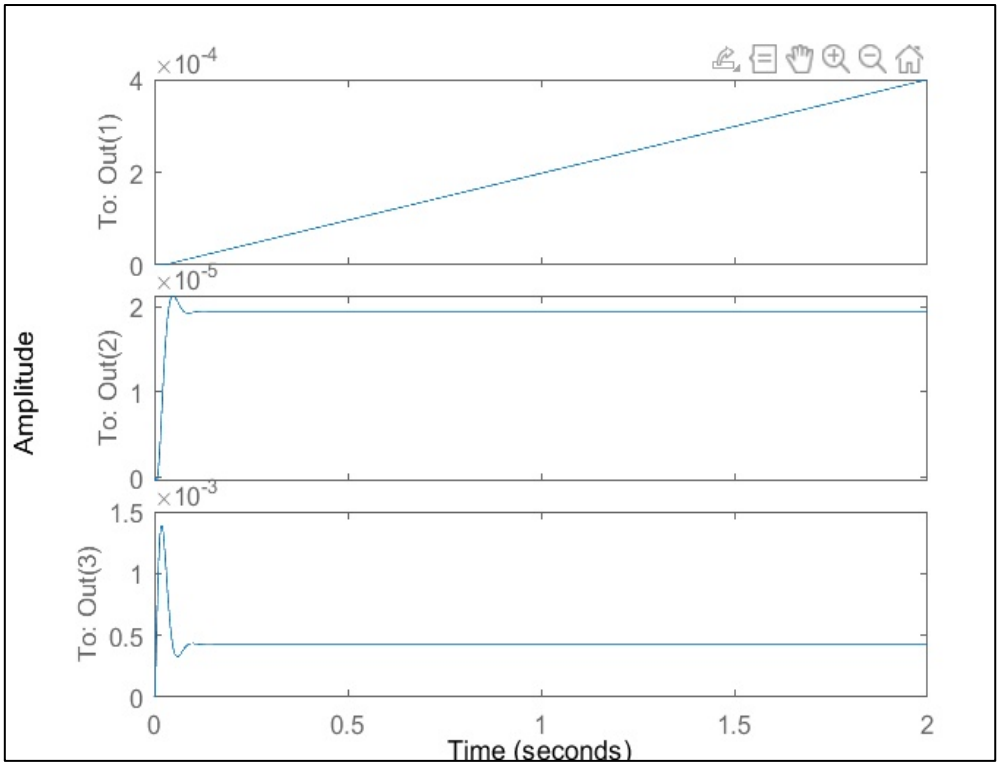
Case 1: With closed loop discrete time eigenvalue: 0.1, 0.4 + j0.4, 0.4 - j0.4



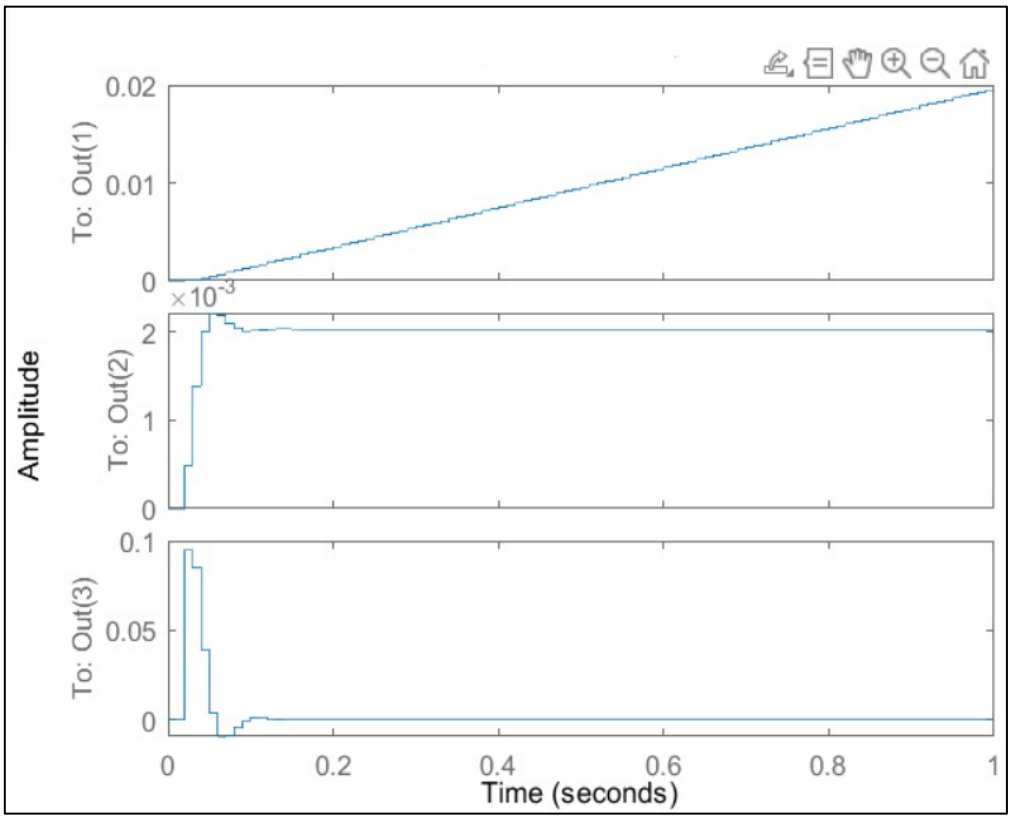
Graph-1: Step Response for Application 1 for discrete time



Graph-2: Step Response for Application 1 for continuous time



Graph-3: Ramp Response for Application 1 for continuous

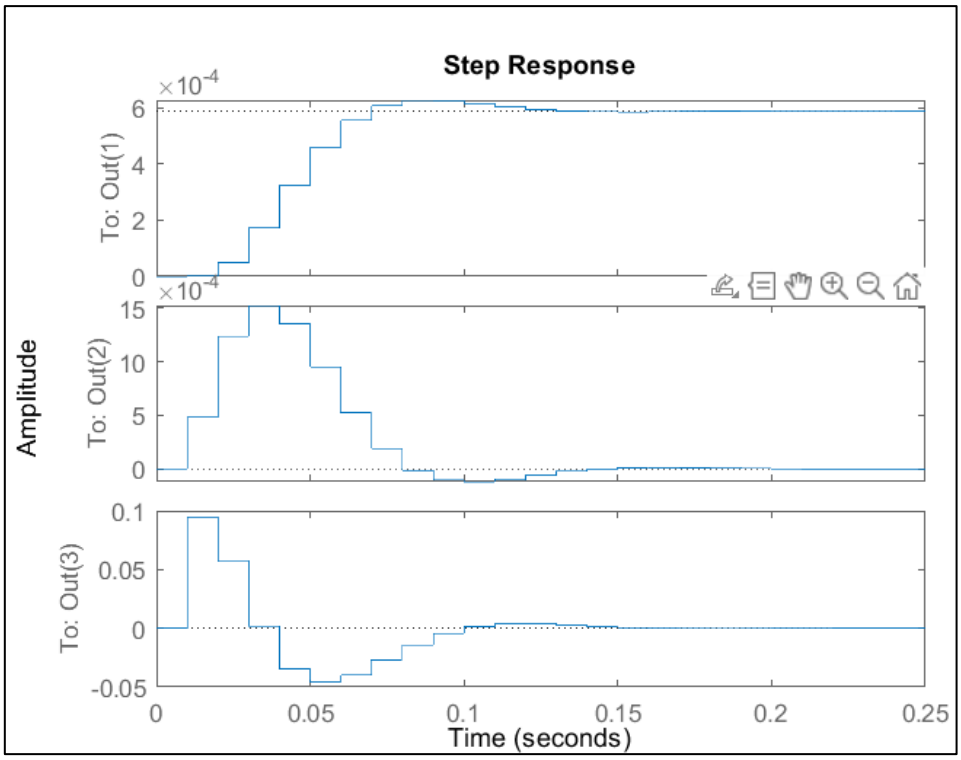


Graph-4: Ramp Response for Application 1 for discrete time

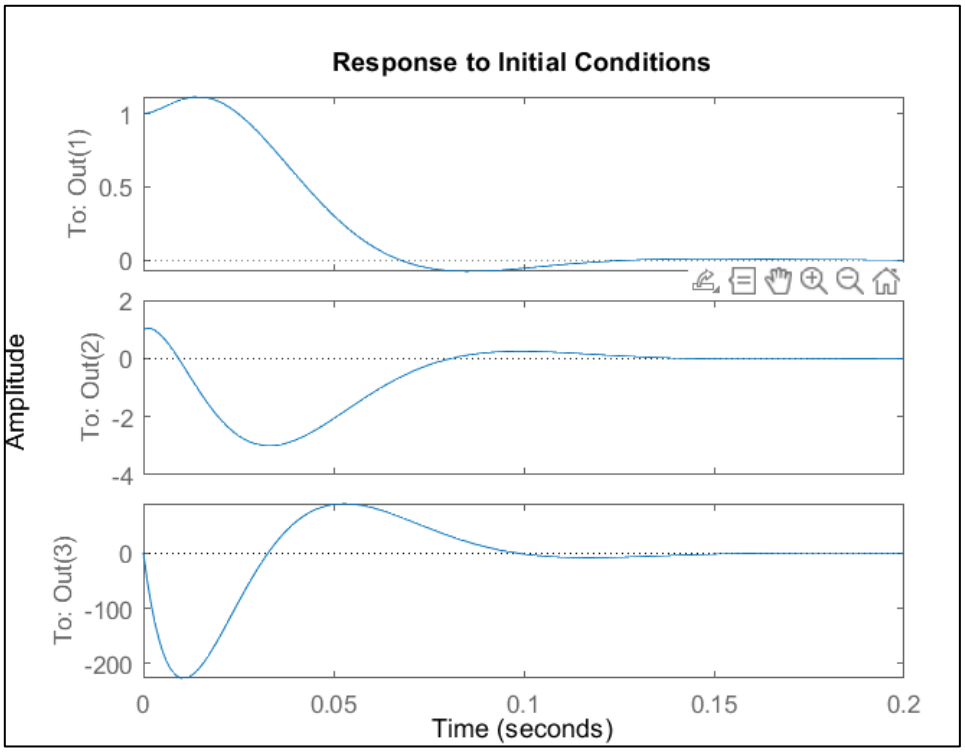
The feedback gain matrix obtained for this application is as follow:

$$\mathbf{k} = \begin{bmatrix} \mathbf{4926.8} \\ \mathbf{1432.4} \\ \mathbf{13.7} \end{bmatrix}$$

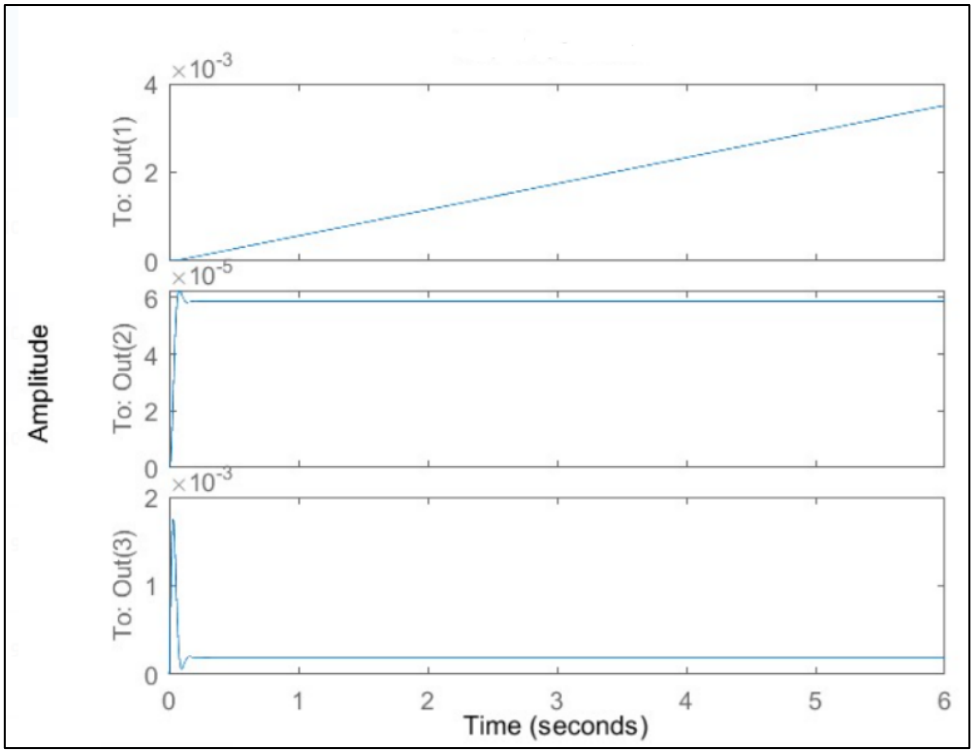
Case 2: With closed loop discrete time eigenvalue: 0.4, 0.6 + j0.33, 0.6 - j0.33



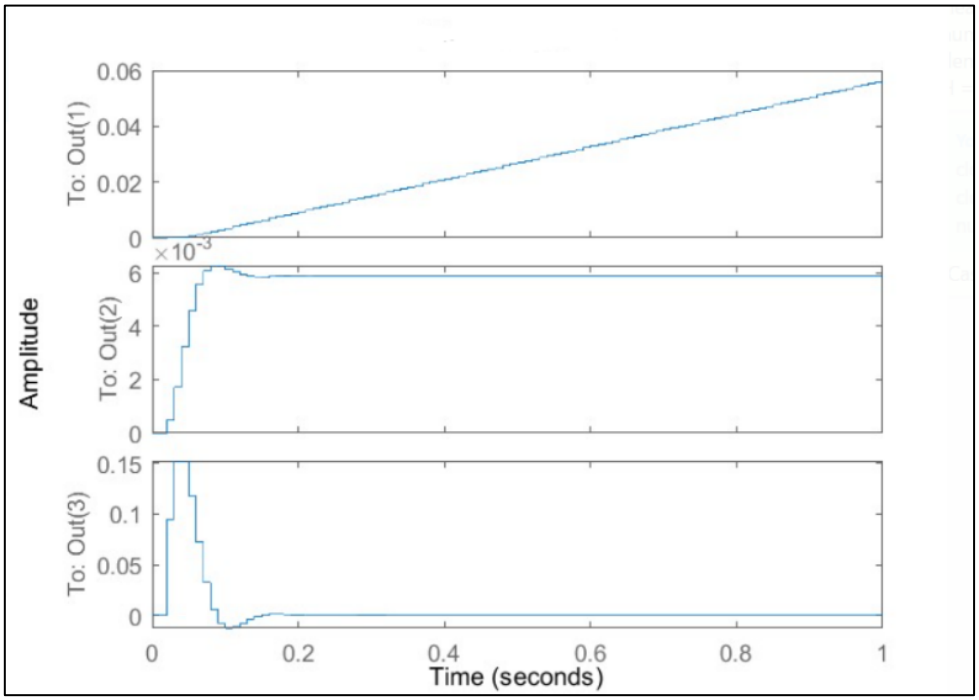
Graph-5: Step Response for Application 2 for discrete time



Graph-6: Step Response for Application 2 for continuous time



Graph-7: Ramp Response for Application 2 for continuous time

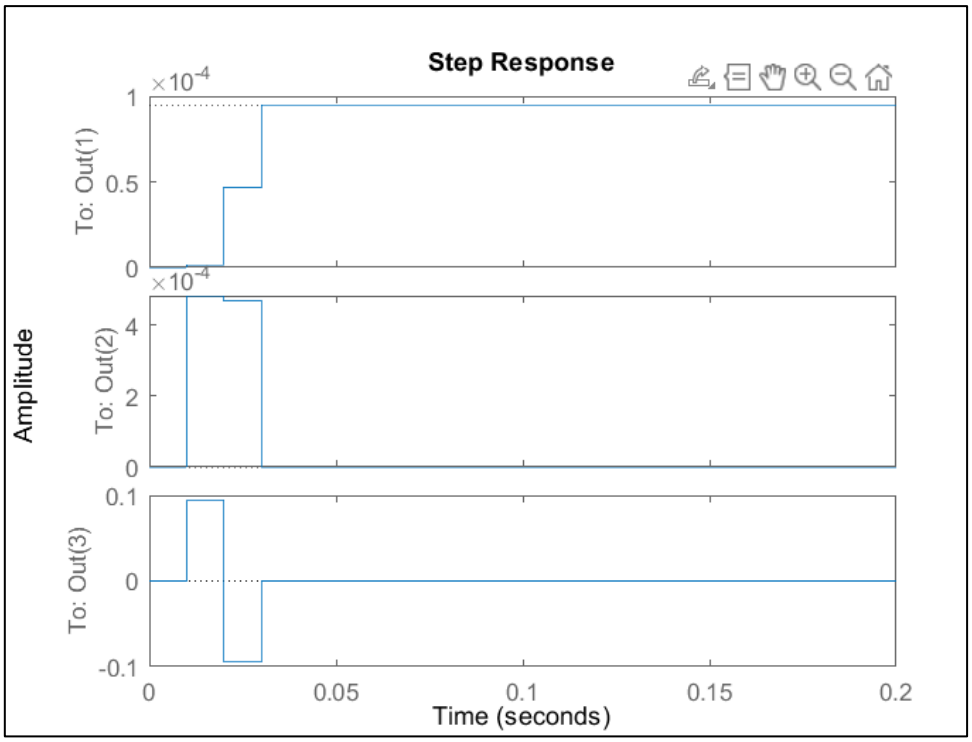


Graph-8: Ramp Response for Application 2 for discrete time

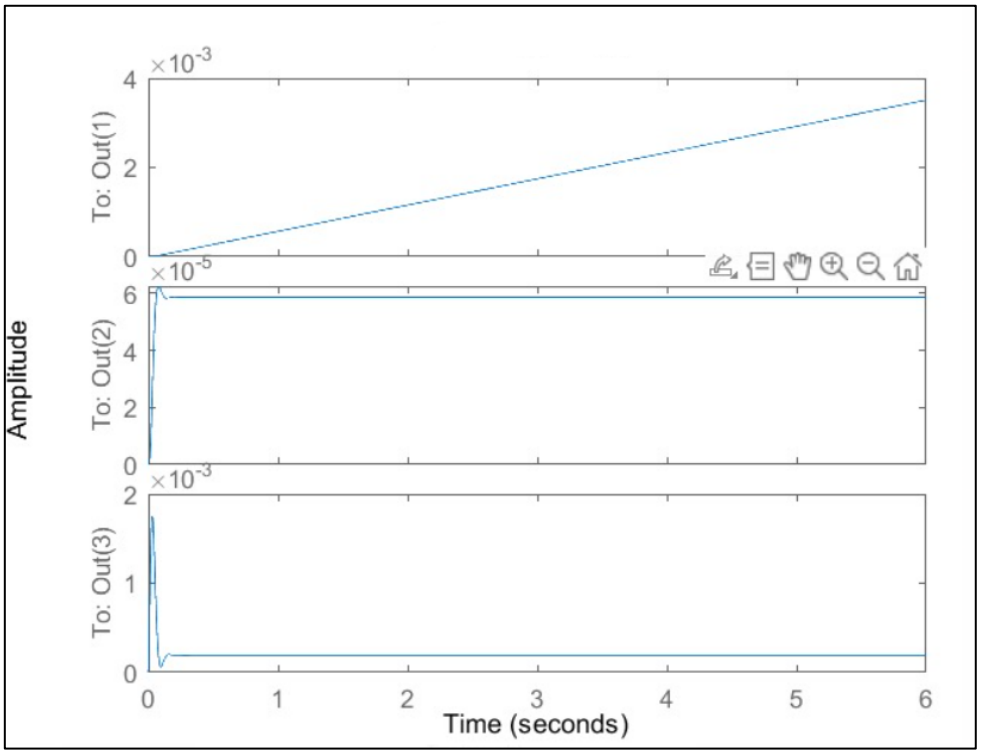
The feedback gain matrix for the above application comes out to be:

$$\mathbf{k} = \begin{bmatrix} \mathbf{1698.5} \\ \mathbf{700.9} \\ \mathbf{10.1} \end{bmatrix}$$

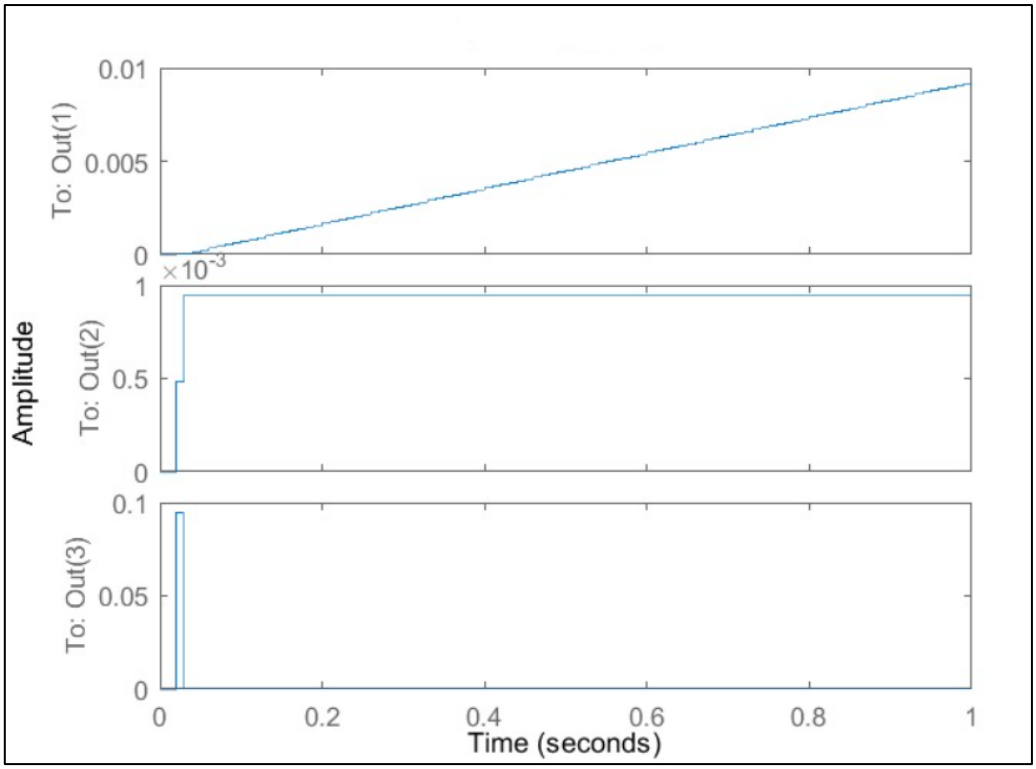
Case 3: With closed loop discrete time eigenvalue: 0, 0, 0



Graph-9: Step Response for Application 3 for discrete time



Graph-10: Ramp Response for Application 3 for continuous time



Graph-11: Ramp Response for Application 3 for discrete time

The feedback gain matrix for the above application comes out to be:

$$\mathbf{k} = \begin{bmatrix} \mathbf{10527} \\ \mathbf{2621} \\ \mathbf{17} \end{bmatrix}$$

We chose C matric to be Identity matrix so as to get the output corresponding to all the states.

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

As the system depicts the armature-controlled DC motor so we can assume that the three states given in the statement corresponds to:

1) Pivot Angle
2) Omega or frequency of rotation
3) Armature Current

But since in the F matrix each row contains more than two non-zero values, the output is dependent on the multiple states. **We cannot analyse the output for the corresponding single states.**

Case1: Eigen Values {0.1, 0.566, 0.566}

Rise Time	Settling Time	Overshoot	Peak Time
0	0.09	98.0903	0.02
0	0.07	63.8278	0.01
0	0.01	0.1978	0.02

Case2: Eigen Values {0.4, 0.685, 0.685}

Rise Time	Settling Time	Overshoot	Peak Time
0.03	0.12	6.4505	0.08
0	NaN	5.9593×10^{18}	0.03
0	NaN	3.4207×10^{18}	0.01

Case3: Eigen Values {0, 0, 0}

Rise Time	Settling Time	Overshoot	Peak Time
0.01	0.03	2.2204×10^{-14}	0.05
0	NaN	1.0289×10^{19}	0.01
0	NaN	3.4207×10^{18}	0.01

Dynamic Analysis and Observations:

- With eigen values of **0.1, 0.4 + j0.4, 0.4 - j0.4**
- The system is quite responsive as peak time is low.
- The system settles down.
- The closed loop system is stable.

- With eigen values of **0.4, 0.6 + j0.33, 0.6 - j0.33:**
- The system is again responsive.
- The system settles in more time than of case1 since the settling time depends on the real part of s domain.
- The closed loop system is not stable.

- With eigen values of **0, 0, 0:**
- Since the poles are located at infinity in s domain, high overshoot is observed.
- The system has negligible rise time.
- Since the rise time, settling time are quite less, the continuous domain provides us with more accurate results.
- The corresponding closed loop system is not stable.

Conclusion:

- The state space system was analysed for different eigen values of the closed loop system with the step response and ramp response and the corresponding outputs were recorded.
- The output corresponds to multiple states so the individual state cannot be analysed.
- The system is quite responsive with low settling time and peak time but the overshoot is quite high.

MATLAB Script:

```
A=[1.0 0.1 0.0;0.0 0.9995 0.0095; 0.0 -0.0947 0.8954 ];
B=[1.622e-6;4.821e-4;9.468e-2];
C=[1 0 0; 0 1 0; 0 0 1];
D=0;
sys_mimo = ss(A,B,C,D,0.01);
tf(sys_mimo);
p=[0.1, 0.4+0.4j, 0.4-0.4j];
k=place(A,B,p);
%k=place();
Anew=A-B*k;
%eig(Anew)
x=[1;1;1];
sys_new = ss(Anew,B,C,D,0.01);
z=stepinfo(sys_new)
stepplot(sys_new)
sys_c=d2c(sys_new)
initial(sys_c,x)
s=tf('s');
z=tf('z');
step(sys_c/s);
step(sys_new*((1/(z-1))),0:0.01:1)
%step(sys_c/s);
%damp(sys_new)
```

Script-1: Pole placement method

```
A=[1.0 0.1 0.0;0.0 0.9995 0.0095; 0.0 -0.0947 0.8954 ];
B=[1.622e-6;4.821e-4;9.468e-2];
C=[1 0 0; 0 1 0; 0 0 1];
D=0;
sys_mimo = ss(A,B,C,D,0.01);
tf(sys_mimo);
p=[0, 0, 0];
k=acker(A,B,p);
%k=place();
Anew=A-B*k;
%eig(Anew)
x=[1;1;1];
sys_new = ss(Anew,B,C,D,0.01);
tf(sys_new);
%sn=initial(sys_new,x)
%z=stepinfo(sys_new)
%stepplot(sys_new)
z=tf('z');
step(sys_new*((1/(z-1))),0:0.01:1)
%s=tf('s');
%step(sys_c/s);
%hold on
%damp(sys_new)
```

Script-2: Ackerman Method

Thank You!