EE208:CONTROL ENGINEERING LAB 07

State feedback controller design on MATLAB platform.

Course Instructor : Dr. Sanjay Roy

Date : 8th March 2022

Group Number - 14

Raj Kumar Deb	2020eeb1025
Tushar Sharma	2020eeb1325
Tiya Jain	2020eeb1213

TABLE OF CONTENTS

1. OBJECTIVE

1.1) PART 1

To design the state feedback gain matrix for a given digital state-space system for which the closed loop discrete time eigenvalue specifications are as follows :

- Application #1 $0.1, 0.4 \pm j0.4$
- Application #2 $0.4, 0.6 \pm j0.33$
- Application #3 All three eigenvalues at the origin.

1.2) PART 2

To discuss the response of the closed loop system for various inputs (zero, step and ramp inputs), for each case with initial state vector as:

$$\mathbf{x}_0 = [1\ 1\ 1]^T$$

2. GIVEN INFORMATION

- Sampling time = 0.01s.
- Discrete State Matrices (Open Loop) :

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

- We have the following state equation for discrete time domain:

$$x[k+1] = F * x[k] + g * u[k],$$

where F is the System Feedback Matrix and g is the System Gain Matrix

3.BLOCK DIAGRAM

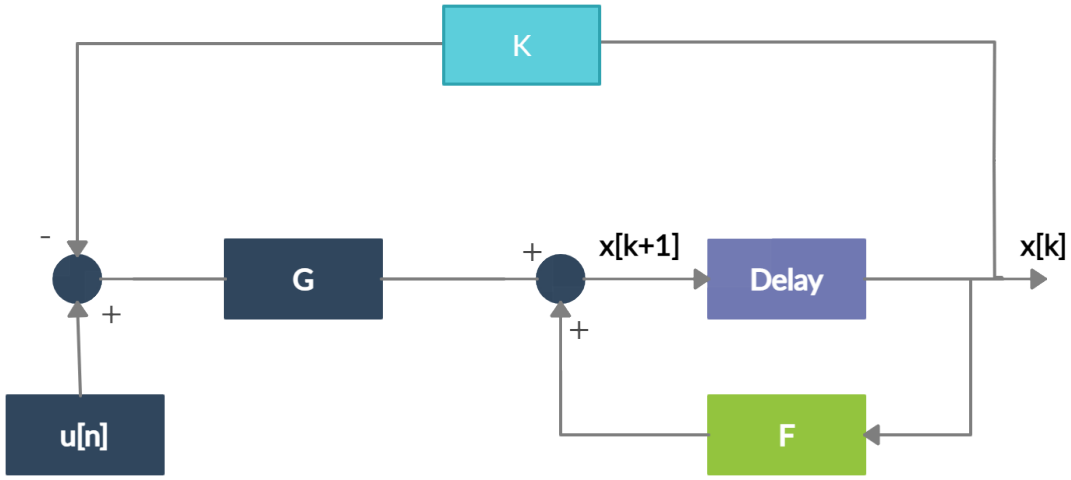


Figure 3.1 Block diagram for the CLTF of the given system

4. METHODOLOGY

Here, we will analyse the matrix given that is provided to us:

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

While, we can observe that the standard form of state space model for a DC motor (below example is for continuous time case, **only for reference**) has the following distribution of its physical parameters (*like J is rotor inertia, b is damping coefficient, K_T is torque and so on*) in matrix:

$$\underbrace{\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{b}{J} & \frac{K_T}{J} \\ 0 & -\frac{K_e}{L} & -\frac{R}{L} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ \frac{1}{L} \end{bmatrix}}_{B} u + \underbrace{\begin{bmatrix} 0 \\ -\frac{1}{J} \\ 0 \end{bmatrix}}_{W} d$$

We won't be able to convert $0 = e^{aT}$ values, as this would give us a = -inf. This delivers us the idea of how the state space is provided to us. The state space doesn't necessarily represent the angle rotated, angular velocity or the armature current, but a mix of these states (*States are interdependent*!!). So in the following observations we will comment only with respect to the given states.

4.1) PART 1 - Designing the state feedback gain matrix for all the 3 applications

Using Pole Placement Method

This method requires us to nominate poles of the closed loop system and it constructs the feedback gain matrix consequently. We take the eigenvalues specifications as p1 , p2 and p3 and then using the *MATLAB place* function we find the State feedback gain matrices for Application #1 and #2, but since all the poles of Application #3 are equal, the *place* function won't work here. For this reason, we use the *acker* function, which gives us the approximation of the poles to be placed.

Shown is the *MATLAB* code for above method:

```
clc

% Given Matrices
F = [
    [1.0 0.10000 0.0000];
    [0.0 0.99500 0.0095];
    [0.0 -0.0947 0.8954];
    ];

g = [
    [1.662*1e-6];
    [4.821*1e-4];
    [9.468*1e-2];
    ];

% Poles to place
p1 = [0.1, 0.4 + 1i*0.4, 0.4- 1i*0.4];
p2 = [0.4, 0.6 + 1i*0.33, 0.6- 1i*0.33];
p3 = [0.0 0.0 0.0];

% Initial State
xn_0 = [[1];[1];[1];];

% Feedback Matrices
K1 = place(F,g,p1);
K2 = place(F,g,p2);
K3 = acker(F,g,p3);
```

4.2) PART 2 - Discussing the response of the closed loop system for various inputs, for each case (Application #1 , #2 and #3) with initial state vector as: $x_0 = [1\ 1\ 1]^T$

We approached this problem by keeping our system in the digital domain only.

We did not assume any C matrix in our analysis.

The code snippet that we used is:

```
% Defining Model
TS = 0.01;
n = 100000;
DATA = zeros(9,n);
n_timestamps = linspace(0*TS,(n-1)*TS,n);

DATA(1:3,1) = xn_0;
DATA(4:6,1) = xn_0;
DATA(7:9,1) = xn_0;

for i = 2:n

%   x(n+1)          :   (A-B*K)*x(n) + g*u(n);
%   u(n) = 0        :   For Zero input response
%   u(n) = 1        :   For Step input response
%   u(n) = n*TS     :   For Ramp input response

%   Calculting response
    DATA(1:3,i) = (F-g*K1)*DATA(1:3,i-1) + g*i*TS;
    DATA(4:6,i) = (F-g*K2)*DATA(4:6,i-1) + g*i*TS;
    DATA(7:9,i) = (F-g*K3)*DATA(7:9,i-1) + g*i*TS;
end

state1_points = [DATA(1,:);DATA(4,:);DATA(7,:);];
state2_points = [DATA(2,:);DATA(5,:);DATA(8,:);];
state3_points = [DATA(3,:);DATA(6,:);DATA(9,:);];
```

For observing the responses we initialise the *DATA* variable (2D array) for the different applications and define the different inputs that we have been given as :

- For Zero Input: u[n] = 0
- For Step Input: u[n] = 1
- For Ramp Input: u[n] = n*$Sampling\ Time$

Given $x_0 = [1\ 1\ 1]^T$,here the x[n+1] represents the next state and x[n] is the present state.

So, we have initialised the first column in the *DATA* array for each of the applications (three rows each for a given application) as the elements of x_0. Further columns of the *DATA* arrays were filled accordingly with the help of the digital domain state equation. We take an iterative approach and find the next values of *DATA* array.

All the different inputs were defined separately and applied to all the three applications given in the problem. After that, we plotted graphs of responses for each case.

```
% Plotting Responses
t = tiledlayout(3,1);
nexttile;
stairs(n_timestamps,state1_points(1,:),'-',LineWidth=1);
grid;
ylabel('Amplitude of State 1');
nexttile;
stairs(n_timestamps,state1_points(1,:),'-',Color='#D95319',LineWidth=1);
grid;
ylabel('Amplitude of State 2');
nexttile;
stairs(n_timestamps,state1_points(1,:),'-',Color='#EDB120',LineWidth=1);
grid;
ylabel('Amplitude of State 3');

% Labelling graph
xlabel(t,'Time Stamps');
title(t,'Ramp Input Response');
```

We plot the zero input, step and ramp responses for every state of every application using the *stairs* function in *MATLAB*. We take the x axis as “time stamps” and the y axis helps us define the value of the states.

5. OBSERVATIONS & THEIR ANALYSIS

5.1) PART 1 - Observations while pole placement for all cases

The state feedback matrix $K = [k1\ k2\ k3]$ obtained for all the three applications are :

Application	$k1$	$k2$	$k3$
1	4.9268×10^3	1.4258×10^3	0.0137×10^3
2	1.6985×10^3	0.6959×10^3	0.0101×10^3
3	1.0527×10^4	0.2613×10^4	0.0017×10^4

Table 4.1 : The state feedback matrices

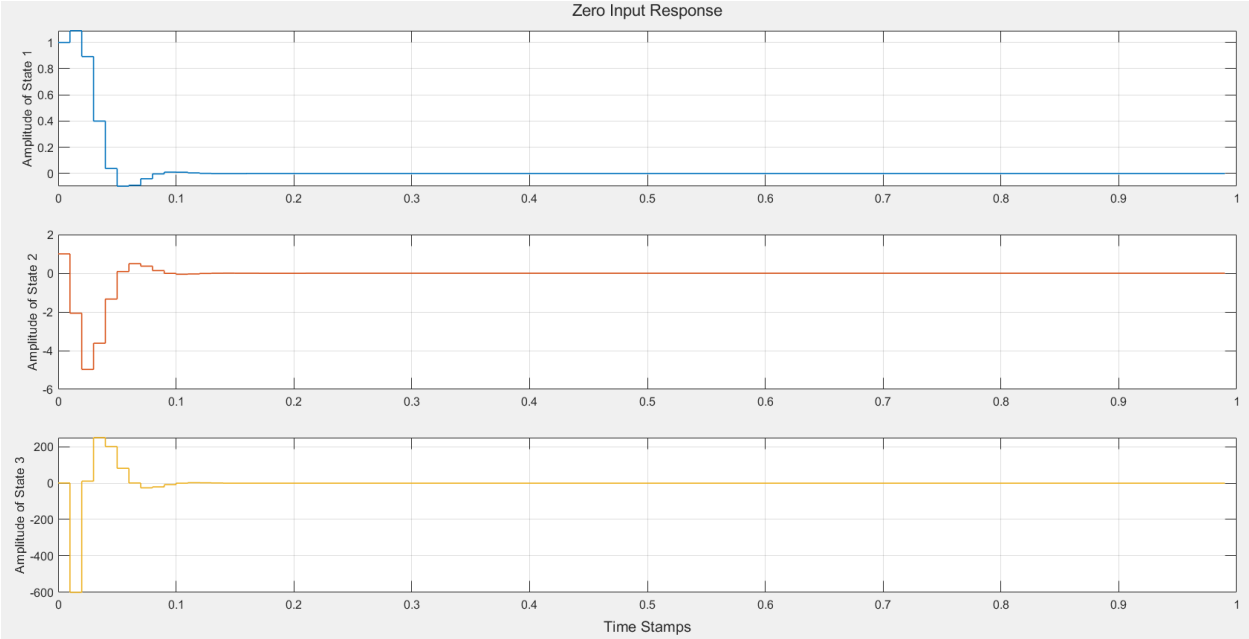
Specific observations :

- For the first and second application, the *place* function gives direct results but when we use this on the third application (where we had all eigenvalues equal to 0) the result was unable to comprehend.
- We received an error and we observed one of the limitations of the *place* function - this function can't place poles with the same eigenvalues. Hence, we tried to search for an alternative approach which led us to discovery of the *acker* function.
- The *acker* function however could place the poles only to a close approximation to the exact eigenvalues(very close to 0 with negligible difference for practical purposes, but not the exact one) for application #3.

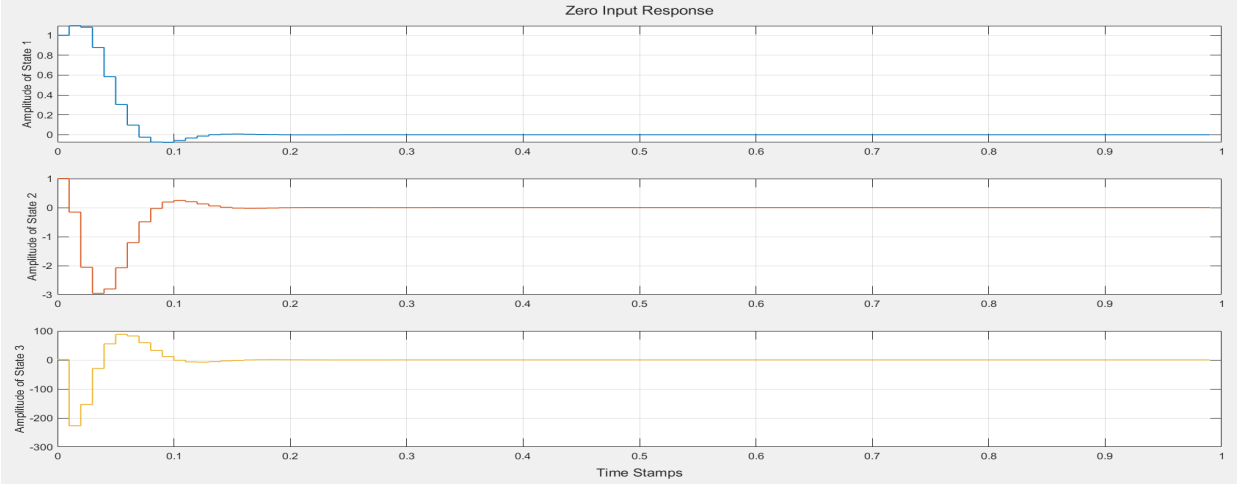
5.2) PART 2 - Observing and analysing the response of the closed loop system for various inputs, for each case (Application #1 , #2 and #3) with initial state vector as: x0 $=[1\ 1\ 1]^T$

5.2.1) Time Domain Responses for ZERO I/P CASES with reference to specific dynamic features characteristic of each application

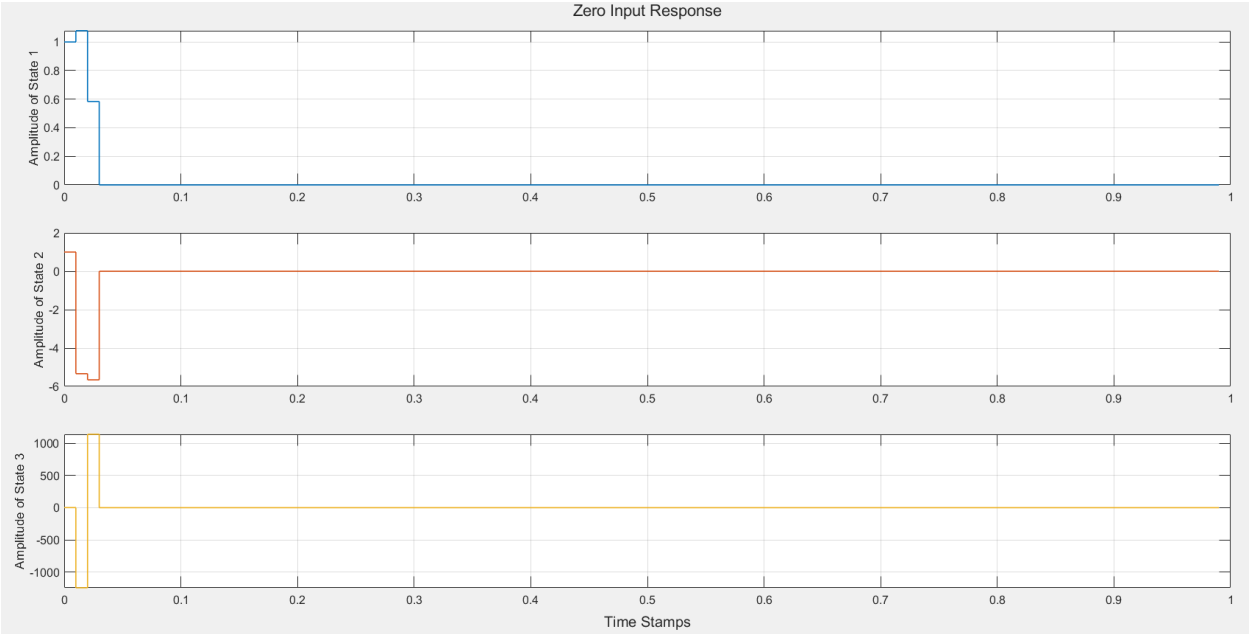
Application #1



Application #2



Application #3



Specific Dynamic feature Observations for zero i/p cases:

- Settling times (in seconds) :

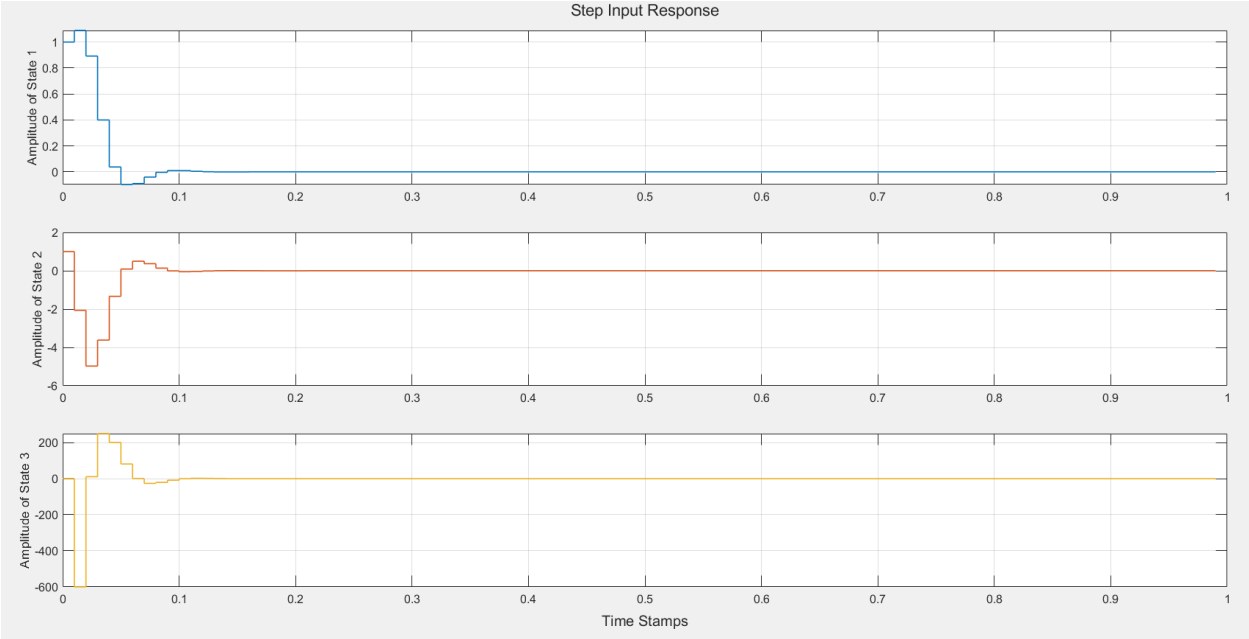
	Application #1	Application #2	Application #3
State 1	0.09	0.13	0.03
State 2	0.13	0.19	0.03
State 3	0.22	0.28	0.03

- Peak Time (in seconds) :

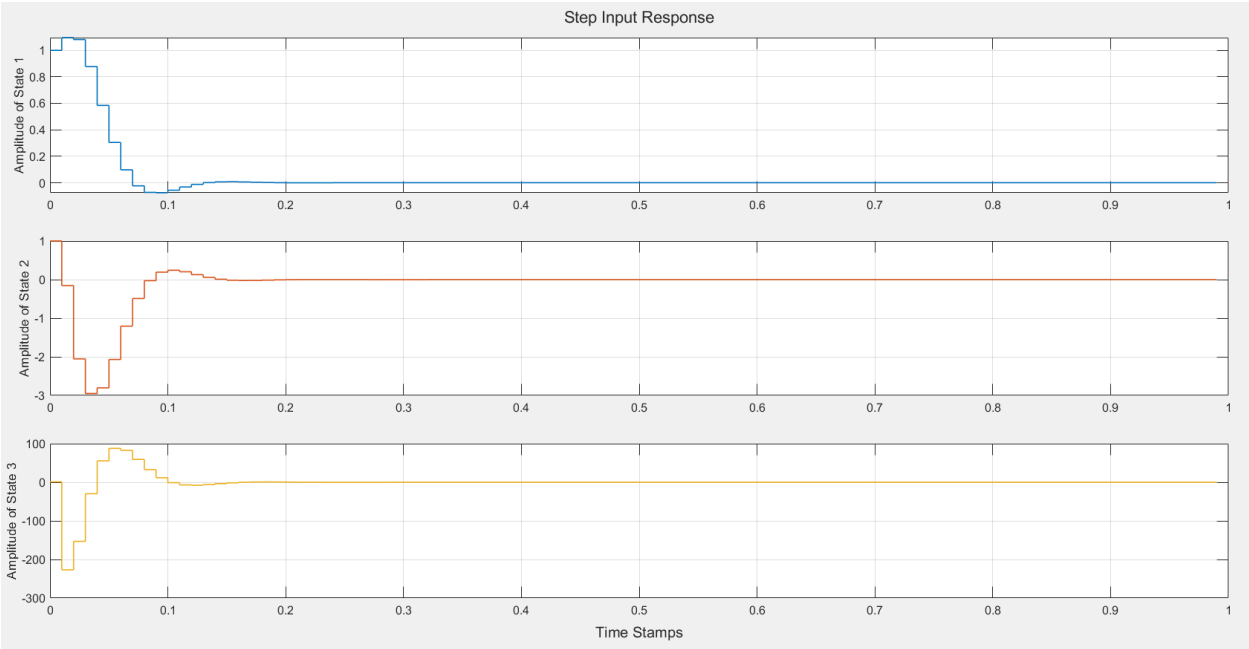
	Application #1	Application #2	Application #3
State 1	0.01	0.01	0.01
State 2	0.02	0.03	0.02
State 3	0.01	0.01	0.01

5.2.2) Time Domain Responses for STEP I/P CASES with reference to specific dynamic features characteristic of each application

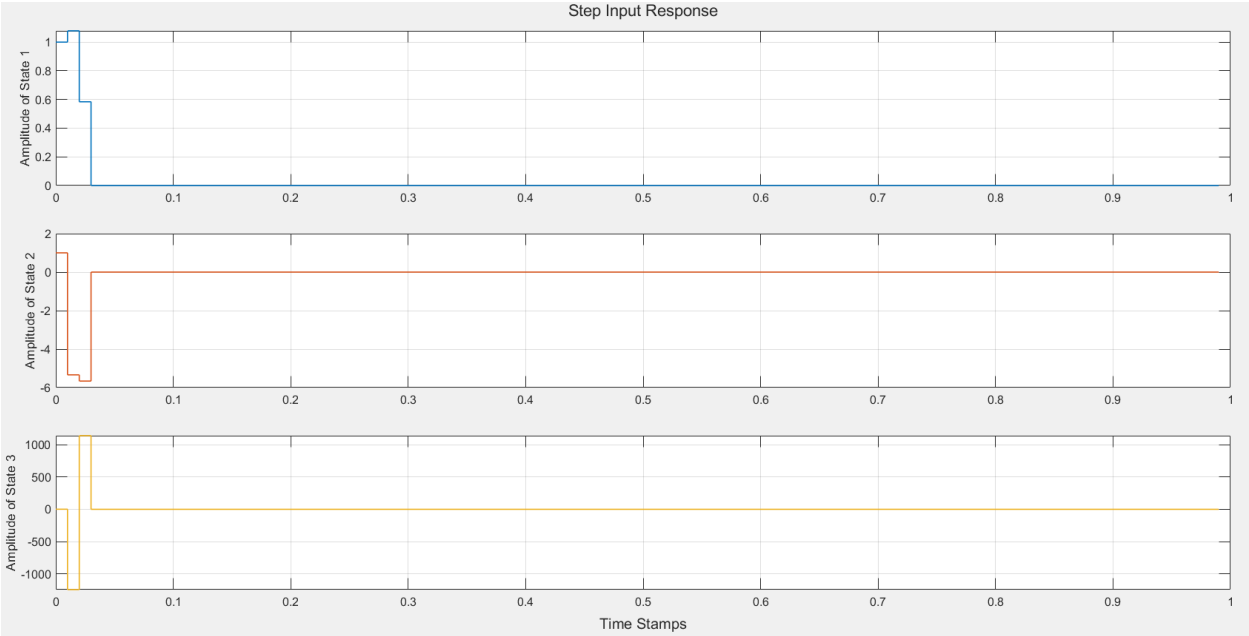
Application #1



APPLICATION #2



APPLICATION #3



Specific Dynamic feature Observations for step i/p cases:

- Settling times(in seconds) :

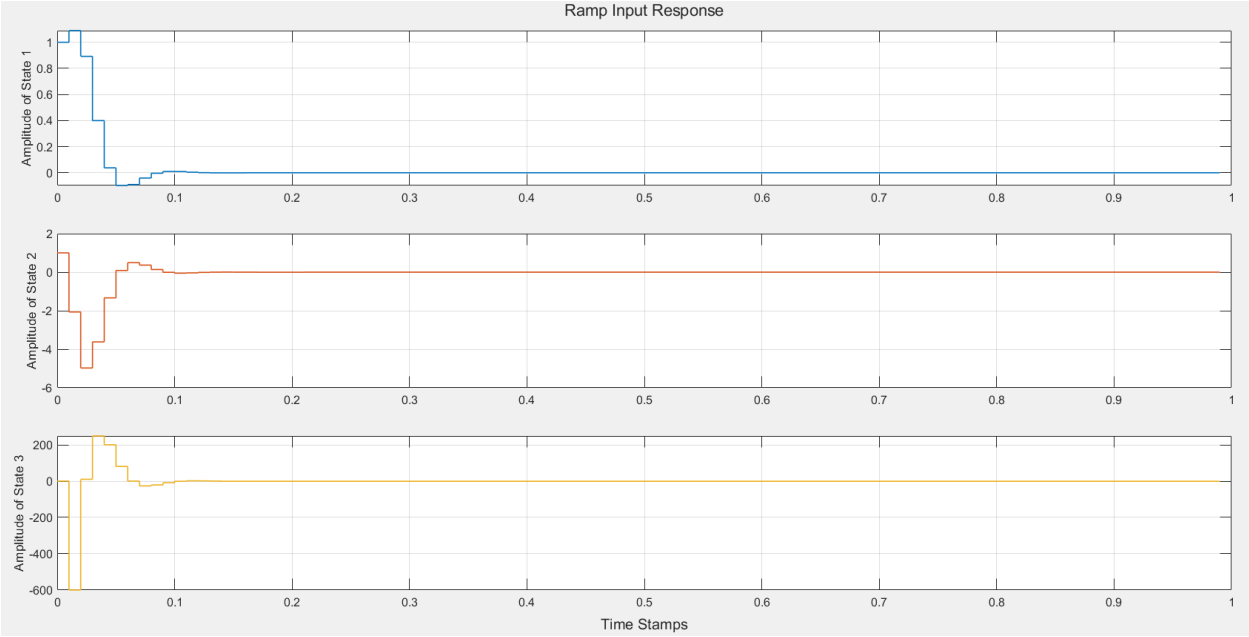
	Application #1	Application #2	Application #3
State 1	0.19	0.28	0.03
State 2	0.23	0.32	0.03
State 3	0.30	0.43	0.03

- Peak Time(in seconds) :

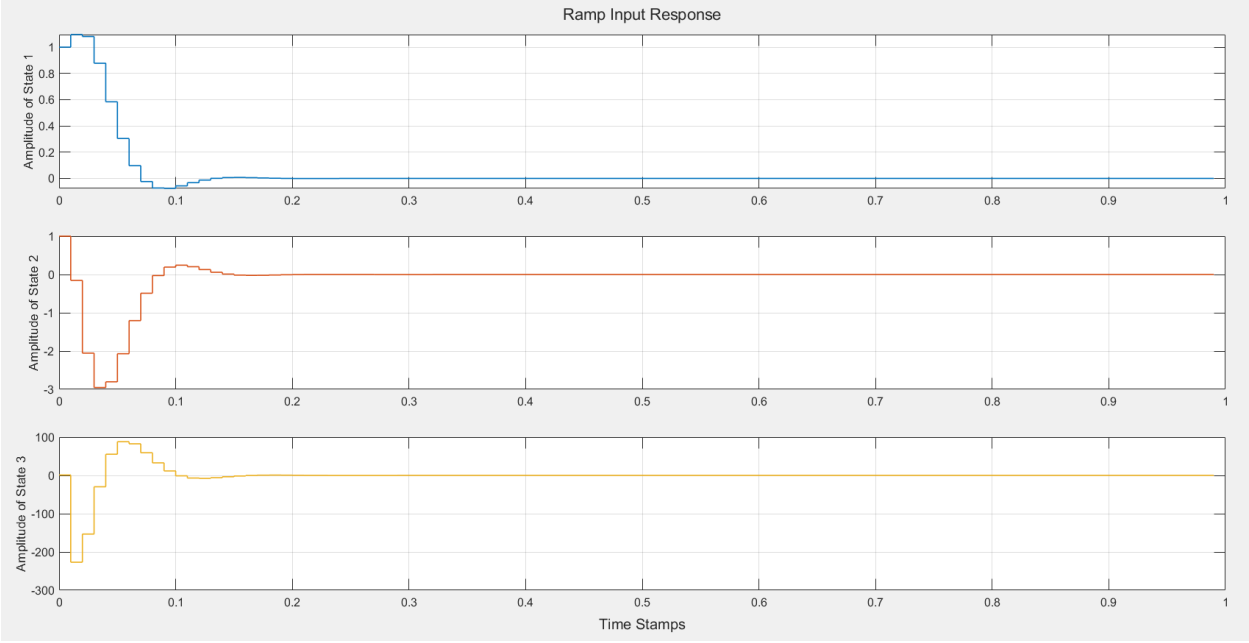
	Application #1	Application #2	Application #3
State 1	0.01	0.01	0.01
State 2	0.02	0.03	0.02
State 3	0.01	0.01	0.01

5.2.3) Time Domain Responses for RAMP I/P CASES with reference to specific dynamic features characteristic of each application

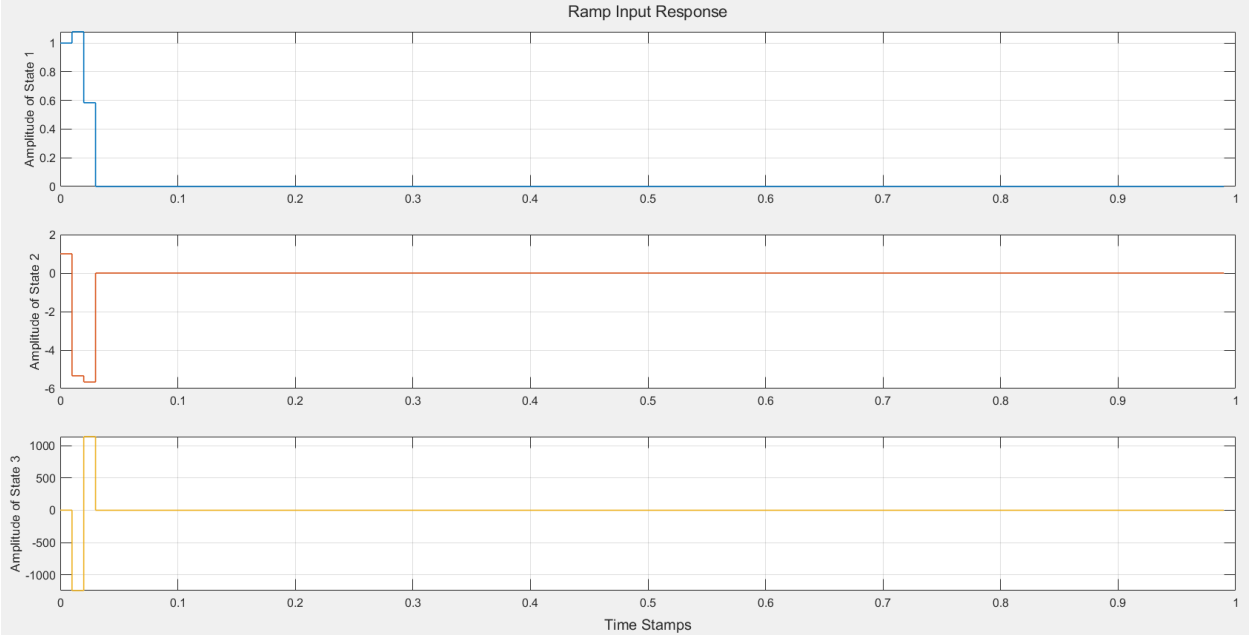
APPLICATION #1



APPLICATION #2



APPLICATION #3



Specific Dynamic feature Observations for ramp i/p cases:

At first glance we can notice that the states seem to settle at t = 0.1s for application #1, t = 0.15s for application #2, t = 0.04s for application #3. We have explained this in the 2nd point of *section* 5. 2. 4.

5.2.4) General Observations and trends:

1. Settling time trend across the states for **zero and step inputs** is :

   Application #2 > *Application* #1 > *Application* #3

   This is due to the locations of eigenvalues for the given applications, as we move our poles (or eigenvalues) closer to the origin (*i. e.* distance of eigenvalues for application #3 is less than that of #1 followed by #2), we get the result that the value of the states tends to settle down faster.

2. We also made a peculiar observation which depends on the type of input as we increase the range of timestamps in our observations (n):
    - **For zero input ($u[n] = 0$) -** We observe that the amplitude of the states settle down to a constant value and after that there occurs no variation in the constant line (approximately $y = 0$ for all states). There is no change in amplitude for all states because of the '0' factor.
    - **For step input ($u[n] = 1$) -** Again the amplitude stabilised and this time the constant line seems to be constant at a non-zero value (though very close to $y = 0$). The factor g results in this observation.
    - **For ramp input ($u[n] = n$) -** The amplitude stabilised again. ***But, only for a short period of time.*** As the value of n increases, so does the ramp input, which leads to continuous rise in amplitude (which once seemed to have stabilised apparently) and hence begins to rise like a ramp as we increase the value of n to some considerably high power of 10 (like around 10^3). The factor of $g * n * Ts$ is an increasing function of n, which is the culprit behind this.

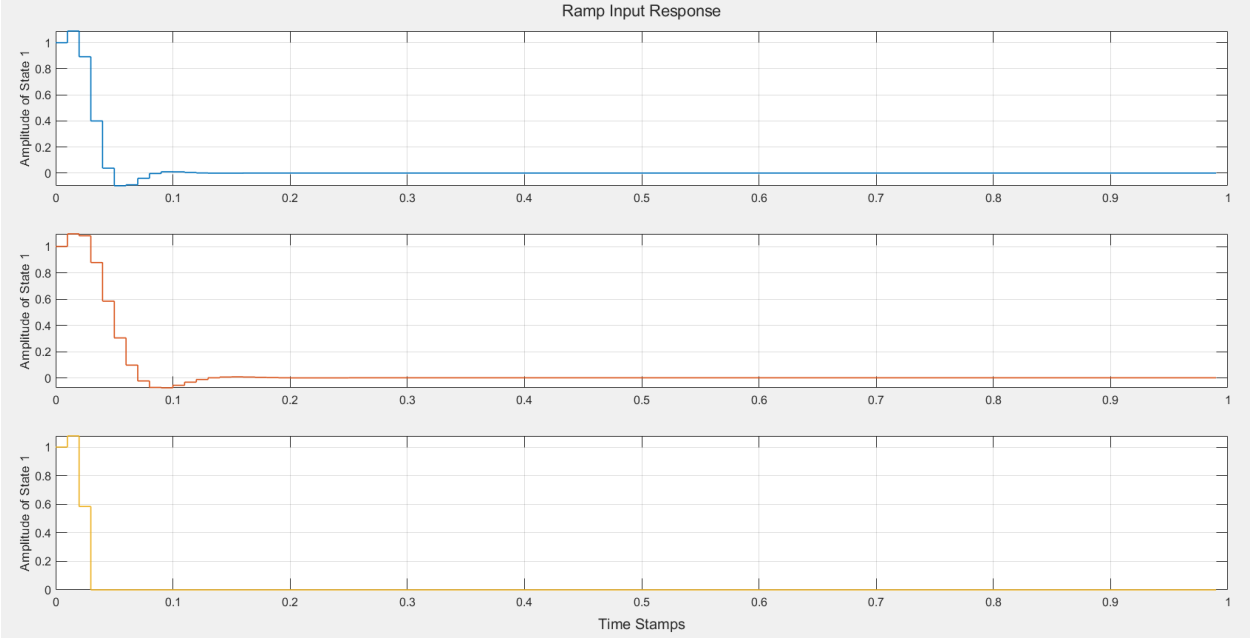


Figure 1 : Ramp response for $n = 0$ to $n = 100$

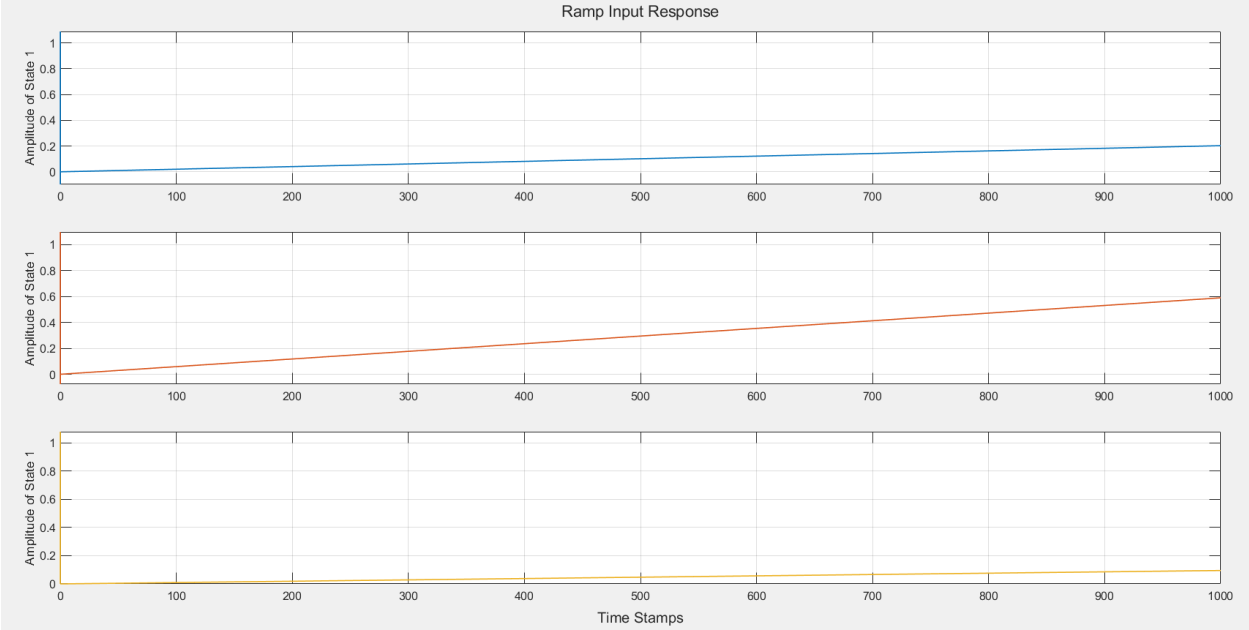


Figure 2 : Ramp response for n = 0 to n=100000

3. According to the trends observed, the graphs for state 1 always register a positive peak, from the initial given state x_0 (whose each element has a value equal to 1) it rises a little and then decreases, whereas for states 2 and 3 one observes negative peaks. Given initial state x_0 for each of these cases, the state value decreases the very next instant (t = 0.01s).

4. We observed from the above graphs that the damping is higher in case of application#2. We further derive this result by narrowing down the angle between the points of the complex poles given in Application#1 and Application#2 :

Application No.	Complex poles	Angle between these poles	Damping ratio
#1	0.4 ± j0.4	45	0.707
#2	0.6 ± j0.33	28.81	0.876

We observed and calculated the angles between the complex poles and the damping ratio, we concluded that they are inter - related.

$$\xi = cos(\theta) = \frac{\alpha}{\sqrt{\alpha^2+\beta^2}}$$

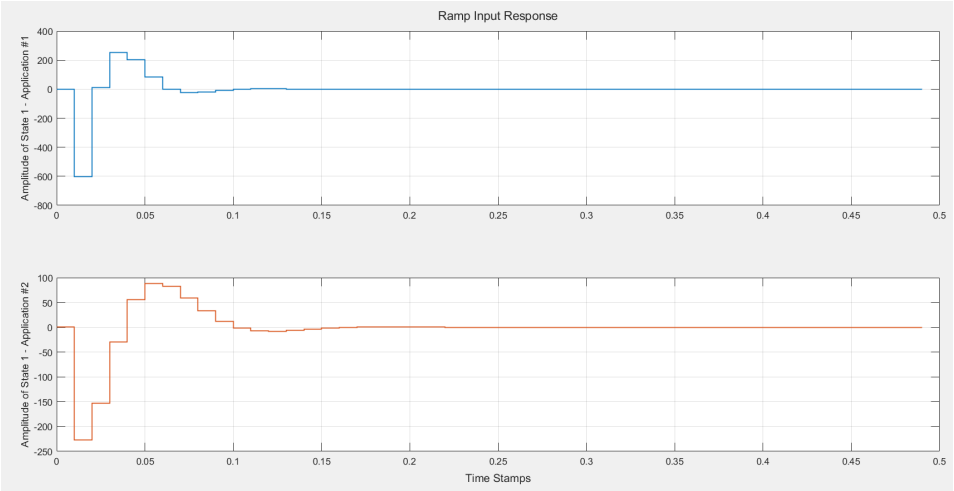


Figure 3: Ramp Input Response for Application #1 and Application #2

The Application having lower damping ratio, will suffer larger oscillations. So, in our case, Application #2 has higher damping ratio compared to Application#1.

Whereas in case of Application #3 all the poles were supposed to be placed at origin but in MATLAB they are not exactly placed at origin and hence we did not infer anything about the damping in this case.

6. CONCLUSION

- We concluded the values of the state feedback matrix $K = [k1\ k2\ k3]$ for all Application #1, #2 and #3 using the *place* and *acker* function. The corresponding values were found as :
  - $K = [4.9268 \times 10^3 \quad 1.4258 \times 10^3 \quad 0.0137 \times 10^3]$ for application #1
  - $K = [1.6985 \times 10^3 \quad 0.6959 \times 10^3 \quad 0.0101 \times 10^3]$ for application #2
  - $K = [1.0527 \times 10^4 \quad 0.2613 \times 10^4 \quad 0.0017 \times 10^4]$ for application #3
- We concluded the responses of the closed loop system for various inputs (zero , step and ramp inputs) , for each case (Application #1 , #2 and #3) with initial state vector as: $x_0 = [1\ 1\ 1]^T$.
- We analysed the graphs plotted using the *stairs* function and came to conclusion regarding a few trends that were observed and have been mentioned in the above sections.