./

Control Systems - Report



Version Number: 1.0

Name: Pushkar Antony

Module: Control Systems

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ver. Rel. No.** | **Release Date** | **Prepared. By** | **Reviewed By** | **Approved By** | **Remarks/Revision Details** |
| 1.0 | 14/04/2021 | Pushkar Antony | Shiva Kumar | Dr.Prithvi Sekhar | Individual report- Control Systems |

**Document History**

Contents

[1(a) First Order Equation with open loop and without controller 5](#_Toc68626744)

[Plant Description 5](#_Toc68626745)

[Code………… 5](#_Toc68626746)

[Math Analysis 10](#_Toc68626747)

[Comparison Analysis:(Speed, Accuracy and stability): 10](#_Toc68626748)

[1(b) First Order Equation with open loop and controller 10](#_Toc68626749)

[Plant Description 10](#_Toc68626750)

[Code…. 10](#_Toc68626751)

[Math Analysis 14](#_Toc68626752)

[Comparison Analysis:(Speed, Accuracy and stability): 14](#_Toc68626753)

[1(c) First Order Equation 15](#_Toc68626754)

[Plant Description 15](#_Toc68626755)

[Code… 15](#_Toc68626756)

[Math Analysis 24](#_Toc68626757)

[Comparison Analysis:(Speed, Accuracy and stability): 24](#_Toc68626758)

[2(a) Second Order MSD Equation 24](#_Toc68626759)

[Plant Description 24](#_Toc68626760)

[Code… 24](#_Toc68626761)

[Math Analysis: 30](#_Toc68626762)

[Comparison Analysis:(Speed, Accuracy and stability): 30](#_Toc68626763)

[2(b) Second Order MSD Equation 31](#_Toc68626764)

[Plant Description 31](#_Toc68626765)

[Code… 31](#_Toc68626766)

[Math Analysis: 35](#_Toc68626767)

[Comparison Analysis:(Speed, Accuracy and stability): 36](#_Toc68626768)

[2(c) Second Order MSD Equation 36](#_Toc68626769)

[Plant Description 36](#_Toc68626770)

[Code… 37](#_Toc68626771)

[Math Analysis: 44](#_Toc68626772)

[Comparison Analysis:(Speed, Accuracy and stability): 45](#_Toc68626773)

[2(d) Movement of Poles. 46](#_Toc68626774)

[Description: Movement of poles is shown along the real and imaginary axis 46](#_Toc68626775)

[Analysis 47](#_Toc68626776)

[2(e) Roots of the Standard Equation 47](#_Toc68626777)

[Code… 47](#_Toc68626778)

[Comparison Analysis: 54](#_Toc68626779)

[2(f) PID Analysis 55](#_Toc68626780)

[First Order System PID Analysis 55](#_Toc68626781)

[Second Order System PID Analysis 56](#_Toc68626782)

[Comparison Analysis: 57](#_Toc68626783)

[3(a) Second Order Exponential Decay system 58](#_Toc68626784)

[Plant Description 58](#_Toc68626785)

[Without Controller 58](#_Toc68626786)

[Open Loop with Controller (P) 60](#_Toc68626787)

[Open Loop with Controller (I) 61](#_Toc68626788)

[Closed Loop- Negative feedback with Controller (D) 63](#_Toc68626789)

[Closed Loop- Positive feedback with Controller (D) 65](#_Toc68626790)

[Math Analysis 67](#_Toc68626791)

[Comparison Analysis 67](#_Toc68626792)

[4(a) 1st Order Differential Equation Model 68](#_Toc68626793)

[4(b) 2nd Order Differential Equation Model 69](#_Toc68626794)

[4(c) Exponential Decay- Radioactive Material Model 69](#_Toc68626795)

[5(a) Migration of Scripts to GNU Octave 70](#_Toc68626796)

[5(b) Migration of Model to OpenModelica 71](#_Toc68626797)

# 1(a) First Order Equation with open loop and without controller

## Plant Description

The Mass-damper first order system is taken as Plant. Equation: f= Bv + M v' f = force; B= coefficient of friction; M = mass ; v= velocity. Values: B1= 0.4 M1=1000; B2= 0.5 M2= 500; B3= 1.7 M3= 340;

## Math Analysis

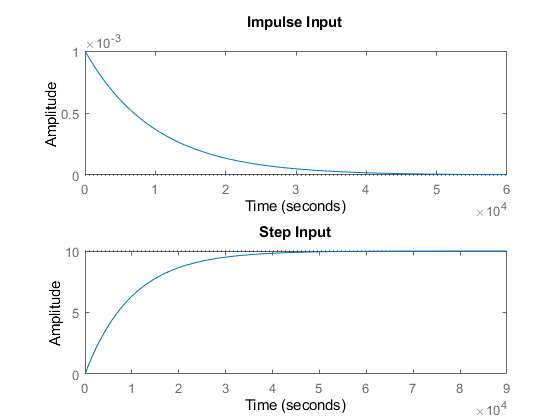
Independent: Time(t) Dependent: Velocity(v) and Force(f) Constant: Mass(M) and Frictional Coefficient(B)

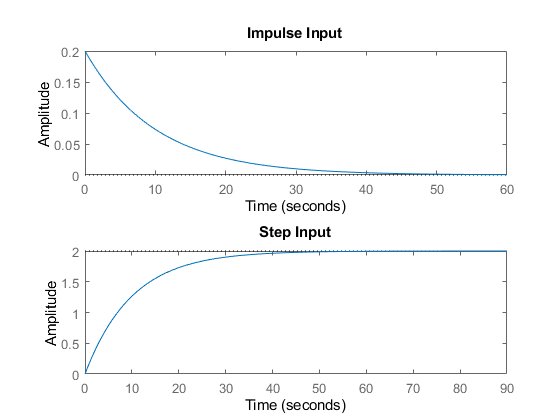
% Roots:(-B)/M  
  
% IVT:  
% 1. For step input: 0  
% 2. For impulse input: 1/M  
  
% FVT:  
% 1. For step input: 1/B  
% 2. For impulse input: 0  
  
% Time Response Results:  
% Rise Time :4tau = (4M)/B; where tau = M/B

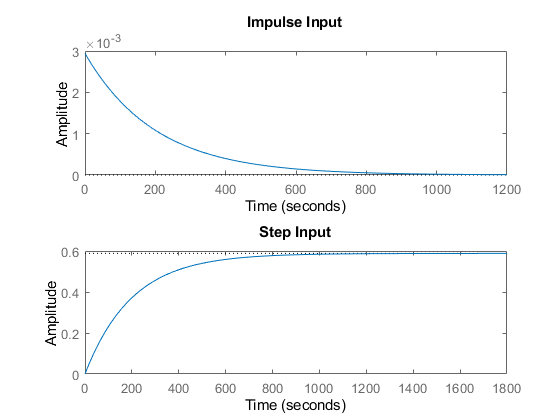
## Tool Analysis:

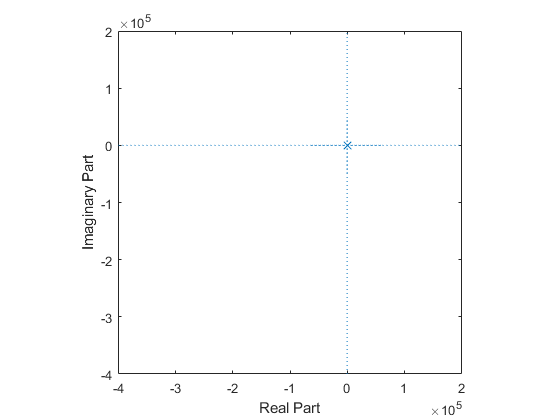
clc;  
B1= ([0.1 0.5 1.7]);  
M1=([1000 5 340]);  
for i=1:3  
 sys = tf([1/M1(i)],[1,B1(i)/M1(i)])  
 figure(i);  
 subplot(2,1,1);  
 impulse(sys);  
 title('Impulse Input');  
 subplot(2,1,2);  
 step(sys);  
 title('Step Input');  
 [z,p,k]= tf2zp([1/M1(i)],[1,B1(i)/M1(i)])  
 figure(4);  
 zplane(z,p);  
 xlim([-4\*1e5 2\*1e5]);  
 ylim([-4\*1e5 2\*1e5]);  
 hold on;  
 S = stepinfo(sys)  
end

sys =  
   
 0.001  
 ----------  
 s + 0.0001  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -1.0000e-04  
  
  
k =  
  
 1.0000e-03  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 2.1970e+04  
 SettlingTime: 3.9121e+04  
 SettlingMin: 9.0450  
 SettlingMax: 9.9997  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 9.9997  
 PeakTime: 1.0546e+05  
  
  
sys =  
   
 0.2  
 -------  
 s + 0.1  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.1000  
  
  
k =  
  
 0.2000  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 21.9701  
 SettlingTime: 39.1207  
 SettlingMin: 1.8090  
 SettlingMax: 1.9999  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 1.9999  
 PeakTime: 105.4584  
  
  
sys =  
   
 0.002941  
 ---------  
 s + 0.005  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.0050  
  
  
k =  
  
 0.0029  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 439.4013  
 SettlingTime: 782.4149  
 SettlingMin: 0.5321  
 SettlingMax: 0.5882  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 0.5882  
 PeakTime: 2.1092e+03









## Comparison Analysis:

%Stability: System 2 is more stable as roots are more negetive.  
%Accuracy: Settling time of System 2 is lowest so System 2 is Accurate.  
%Speed: Rise time of System 2 is lowest so System 2 has the highest speed.

# 1(b) First Order Equation with open loop and controller

## Plant Description Plant Description

The Mass-damper first order system is taken as Plant. Equation: f= Bv + M v' f = force; B= coefficient of friction; M = mass ; v= velocity. Values: B1= 0.4 M1=1000; B2= 0.5 M2= 500; B3= 1.7 M3= 340;

## Math Analysis

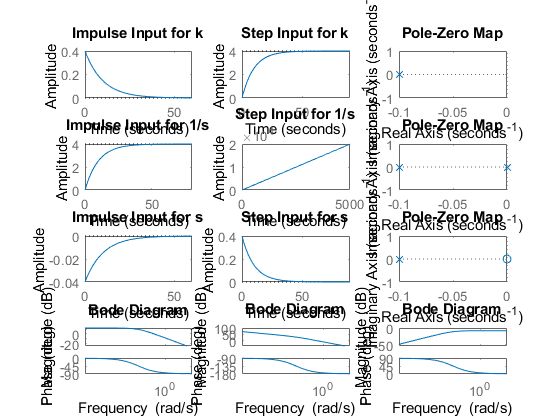
Independent: Time(t) Dependent: Velocity(v) and Force(f) Constant: Mass(M) and Frictional Coefficient(B)

% Roots:(-B)/M  
  
% Poles and Zero Calculation:  
  
  
% IVT:  
% 1. For step input: 0  
% 2. For impulse input: 1/M  
  
% FVT:  
% 1. For step input: 1/B  
% 2. For impulse input: 0  
  
% Time Response Results:  
% Rise Time :4tau = (4M)/B; where tau = M/B

## Tool Analysis:

clc;  
B1= 0.5;  
M1= 5;  
P = 2;  
  
 sys = tf([P/M1],[1,B1/M1])  
 subplot(4,3,1);  
 impulse(sys);  
 title('Impulse Input for k');  
 subplot(4,3,2);  
 step(sys);  
 title('Step Input for k');  
 subplot(4,3,3);  
 [z,p,k]= tf2zp([P/M1],[1,B1/M1])  
 pzmap(sys)  
 subplot(4,3,10);  
 bode(sys)  
  
 hold on;  
 S = stepinfo(sys)  
  
sys = tf([P/M1],[1,B1/M1,0])  
subplot(4,3,4);  
impulse(sys);  
title('Impulse Input for 1/s');  
subplot(4,3,5);  
step(sys);  
title('Step Input for 1/s');  
subplot(4,3,6);  
[z,p,k]= tf2zp([P/M1],[1,B1/M1,0])  
pzmap(sys)  
 subplot(4,3,11);  
 bode(sys)  
hold on;  
S = stepinfo(sys)  
sys = tf([P/M1,0],[1,B1/M1])  
subplot(4,3,7);  
impulse(sys);  
title('Impulse Input for s');  
subplot(4,3,8);  
step(sys);  
title('Step Input for s');  
subplot(4,3,9);  
[z,p,k]= tf2zp([P/M1,0],[1,B1/M1])  
pzmap(sys)  
 subplot(4,3,12);  
 bode(sys)  
hold on;  
S = stepinfo(sys)

sys =  
   
 0.4  
 -------  
 s + 0.1  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.1000  
  
  
k =  
  
 0.4000  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 21.9701  
 SettlingTime: 39.1207  
 SettlingMin: 3.6180  
 SettlingMax: 3.9999  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 3.9999  
 PeakTime: 105.4584  
  
  
sys =  
   
 0.4  
 -----------  
 s^2 + 0.1 s  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0  
 -0.1000  
  
  
k =  
  
 0.4000  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
sys =  
   
 0.4 s  
 -------  
 s + 0.1  
   
Continuous-time transfer function.  
  
  
z =  
  
 0  
  
  
p =  
  
 -0.1000  
  
  
k =  
  
 0.4000  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 21.9701  
 SettlingTime: 39.1207  
 SettlingMin: 1.0521e-05  
 SettlingMax: 0.0382  
 Overshoot: 0  
 Undershoot: 7.2058e+17  
 Peak: 0.4000  
 PeakTime: 0



## Comparison Analysis:

Propotional controller will have no effect on Rising time and setting time of the system hence there response parameters remains same for all values of p and only the amplitude of the system get affected by P times.

## %Integrator controller is used to eliminate steady state error from the %system , but as a pole is added system is going from stable to marginally %stable. %Differentiator control on system adds a zero to the system and makes %unstable system stable in our case %Stability: Both System 1 and System 3 are stable as roots are negative and have same system responses. %Accuracy: Settling time of System 1 and System 3 is lowest so they are Accurate. %Speed: Rise time of System 1 and System 3 is lowest so they are speed system.

# First Order Equation CL

## 

## Plant Description

The Mass-damper first order system is taken as Plant. Equation: f= Bv + M v' f = force; B= coefficient of friction; M = mass ; v= velocity. Values: B1= 0.4 M1=1000; B2= 0.5 M2= 500; B3= 1.7 M3= 340;

## Math Analysis

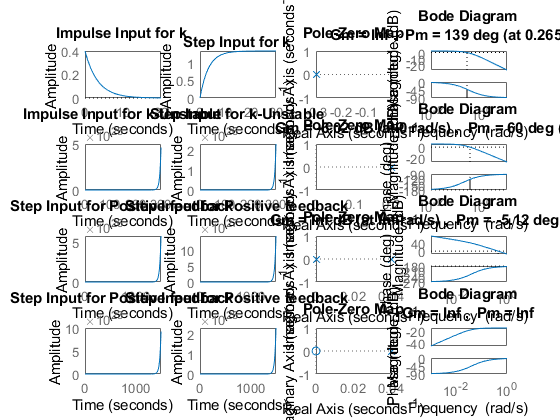
Independent: Time(t) Dependent: Velocity(v) and Force(f) Constant: Mass(M) and Frictional Coefficient(B)

% Roots:(-B)/M  
  
% Poles and Zero Calculation:  
  
% IVT:  
% 1. For step input: 0  
% 2. For impulse input: 1/M  
  
% FVT:  
% 1. For step input: 1/B  
% 2. For impulse input: 0  
  
% Time Response Results:  
% Rise Time :4tau = (4M)/B; where tau = M/B

## Tool Analysis:

%Negative Feedback using gain input  
clc;  
%Stable P control  
B1= 0.5;  
M1= 5;  
P = 2;  
  
sys = tf([P],[M1,B1+1])  
subplot(4,4,1);  
impulse(sys);  
title('Impulse Input for k');  
subplot(4,4,2);  
step(sys);  
title('Step Input for k');  
subplot(4,4,3);  
[z,p,k]= tf2zp([P],[M1,B1+1])  
pzmap(sys)  
subplot(4,4,4)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)  
  
%Unstable P controll  
B2= -2;  
M2= 5;  
P2 = 2;  
  
sys = tf([P2],[M2,B2+1])  
subplot(4,4,5);  
impulse(sys);  
title('Impulse Input for k-Unstable');  
subplot(4,4,6);  
step(sys);  
title('Step Input for k-Unstable');  
subplot(4,4,7);  
[z,p,k]= tf2zp([P2],[M2,B2+1])  
pzmap(sys)  
subplot(4,4,8)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)  
  
%Positive Feedback using integral input  
B3= 0.8;  
M3= 5;  
  
  
sys = tf([1],[M3,B3-1,0])  
subplot(4,4,9);  
impulse(sys);  
title('Step Input for Positive feedback');  
subplot(4,4,10);  
step(sys);  
title('Step Input for Positive feedback');  
subplot(4,4,11);  
[z,p,k]= tf2zp([1],[M3,B3-1,0])  
pzmap(sys)  
subplot(4,4,12)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)  
  
% Positive Feedback using differentiator input  
B4= 0.8;  
M4= 5;  
  
  
sys = tf([1,0],[M4,B4-1])  
subplot(4,4,13);  
impulse(sys);  
title('Step Input for Positive feedback');  
subplot(4,4,14);  
step(sys);  
title('Step Input for Positive feedback');  
subplot(4,4,15);  
[z,p,k]= tf2zp([1,0],[M4,B4-1])  
pzmap(sys)  
subplot(4,4,16)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)

sys =  
   
 2  
 ---------  
 5 s + 1.5  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.3000  
  
  
k =  
  
 0.4000  
  
  
Gm =  
  
 Inf  
  
  
Pm =  
  
 138.5925  
  
  
Wcg =  
  
 NaN  
  
  
Wcp =  
  
 0.2646  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 7.3234  
 SettlingTime: 13.0402  
 SettlingMin: 1.2060  
 SettlingMax: 1.3333  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 1.3333  
 PeakTime: 35.1528  
  
  
sys =  
   
 2  
 -------  
 5 s - 1  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0.2000  
  
  
k =  
  
 0.4000  
  
  
Gm =  
  
 0.5000  
  
  
Pm =  
  
 59.9993  
  
  
Wcg =  
  
 0  
  
  
Wcp =  
  
 0.3464  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
sys =  
   
 1  
 -------------  
 5 s^2 - 0.2 s  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0  
 0.0400  
  
  
k =  
  
 0.2000  
  
Warning: The closed-loop system is  
unstable.   
  
Gm =  
  
 Inf  
  
  
Pm =  
  
 -5.1214  
  
  
Wcg =  
  
 Inf  
  
  
Wcp =  
  
 0.4463  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
sys =  
   
 s  
 ---------  
 5 s - 0.2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0  
  
  
p =  
  
 0.0400  
  
  
k =  
  
 0.2000  
  
Warning: The closed-loop system is  
unstable.   
  
Gm =  
  
 Inf  
  
  
Pm =  
  
 Inf  
  
  
Wcg =  
  
 NaN  
  
  
Wcp =  
  
 NaN  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf



## Comparison Analysis:

%Negetive feedback system:  
%for negetive feedback system poles lie in -0.3  
%and hence the system is stable, and when the controller is added to the  
%system rasie time and settiling time of the response is reduced and  
%system is becoming more fast and stable.  
  
%Positive feedback:  
%In positive feedback system the roots are lying in the  
%right hand side of the s plane and the system is unstable.  
%Bode Plot:  
%phase margin-139degree and gain margin-infinite  
%when the phase of the loop gain goes below -180 degree then gain margin is  
%infinite Wcp=0.264.  
%phase margin:-60 degree and gain margin:-6.02dB  
%phase margin is infinite because the magnitude plot goes below 0dB at all  
%frequencies this indicates the system will have trouble tracking various  
%reference signal without excessive error.  
% Rise time of Sys2 is lowest so Sys2 has the highest speed.]  
% Settling time of Sys2 is lowest so Sys2 is most stable.  
%the gain margin is infinite for all values if Kp since the gain is zero at  
%gain cross over frequency

# 2(a) Second Order MSD Equation

## Plant Description

The Mass-damper Spring Second order system is taken as Plant. It is used in as suspension.

% Equation: Mx''(t)+ Bx'(t) + Kx(t)= Kf(t).  
% f = force; B= coefficient of friction; M = mass ; v= velocity; k=spring constant.  
% Values: K1= 0.9 B1= 0.4 M1=1000 Wn=0.03 ; K2= 1 B2= 0.5 M2= 500 Wn=0.44; K3= 3 B3= 1.7 M3= 340 Wn=0.09;

## Math Analysis:

Independent: Time(t) Dependent: Velocity(v) and Force(f) Constant: Mass(M), Frictional Coefficient(B), Spring constant(K)

% Roots:((-B/M)+-sqrt(sq(B/M)-4K/M))/2  
  
  
% IVT:  
% 1. For step input: 0  
% 2. For impulse input: 0  
  
% FVT:  
% 1. For step input: 1  
% 2. For impulse input: K/M

## Tool Analysis:

B1= ([0.1 0.5 1.7]);  
M1=([1000 5 340]);M1=([1000 5 340]);  
K1 = ([0.9 1 3]);  
for i=1:3  
 sys = tf([K1(i)/M1(i)],[1,B1(i)/M1(i),K1(i)/M1(i)])  
 figure(i);  
 subplot(2,1,1);  
 impulse(sys);  
 title('Impulse Input');  
 subplot(2,1,2);  
 step(sys);  
 title('Step Input');  
 [z,p,k]= tf2zp([K1(i)/M1(i)],[1,B1(i)/M1(i),K1(i)/M1(i)])  
 figure(4);  
 zplane(z,p);  
 xlim([-5\*1e5 3\*1e5]);  
 ylim([-5\*1e5 3\*1e5]);  
 hold on;  
 S = stepinfo(sys)  
end

sys =  
   
 0.0009  
 -----------------------  
 s^2 + 0.0001 s + 0.0009  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.0001 + 0.0300i  
 -0.0001 - 0.0300i  
  
  
k =  
  
 9.0000e-04

License checkout failed.  
License Manager Error -10  
Your license for Signal\_Toolbox has expired.   
If you are not using a trial license contact your License Administrator to obtain an updated license.   
Otherwise, contact your Sales Representative for a trial extension.  
  
Troubleshoot this issue by visiting:   
<a href="https://www.mathworks.com/support/lme/R2020b/10">https://www.mathworks.com/support/lme/R2020b/10</a>  
  
Diagnostic Information:  
Feature: Signal\_Toolbox   
License path: C:\Users\99003728\AppData\Roaming\MathWorks\MATLAB\R2020b\_licenses;C:\Program Files\MATLAB\R2020b\licenses\license.dat;C:\Program Files\MATLAB\R2020b\licenses\trial\_8681003\_R2020b.lic   
Licensing error: -10,32.  
  
Error in sec\_order\_sys\_MSD (line 52)  
 zplane(z,p);

## Comparison Analysis:

Time Response Results:

%system  
% K1= 0.9 B1= 0.4 M1=1000  
%RiseTime: 34.7791  
%SettlingTime: 7.8226e+04  
%percentage Overshoot: 99.4778  
%PeakTime: 104.7198  
  
  
%K2= 1 B2= 0.5 M2= 500  
%RiseTime:2.5448  
%SettlingTime: 78.1524  
%percentage Overshoot: 70.2118  
%PeakTime: 70.2118  
  
%K3= 3 B3= 1.7 M3= 340  
%RiseTime:1.5426e+03  
%SettlingTime: 0.1540  
%percentage Overshoot: 70.2118  
%PeakTime: 33.4448  
%Speed: System 2 is having low raise time and is therefore speed system.  
%Stability: Settling time of system of system 3 is less and therefore is  
%the stable system.  
%Accuracy: Settling time of system of system 3 is less and therefore is  
%more accurate.

# 2(b) Second Order MSD Equation

## Plant Description

The Mass-damper Spring Second order system is taken as Plant. It is used in as suspension.

% Equation: Mx''(t)+ Bx'(t) + Kx(t)= Kf(t).  
% f = force; B= coefficient of friction; M = mass ; v= velocity; k=spring constant.  
% Values: K1= 0.9 B1= 0.4 M1=1000 Wn=0.03 ; K2= 1 B2= 0.5 M2= 500 Wn=0.44; K3= 3 B3= 1.7 M3= 340 Wn=0.09;

## Math Analysis:

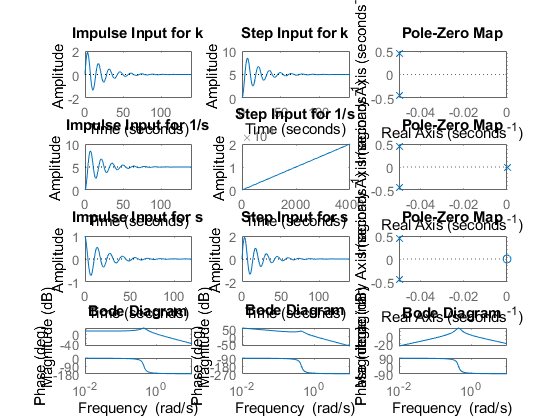
Independent: Time(t) Dependent: Velocity(v) and Force(f) Constant: Mass(M), Frictional Coefficient(B), Spring constant(K)

% Roots:((-B/M)+-sqrt(sq(B/M)-4K/M))/2  
  
% Poles and Zero Calculation:  
  
  
% IVT:  
% 1. For step input: 0  
% 2. For impulse input: 0  
  
% FVT:  
% 1. For step input: 1  
% 2. For impulse input: K/M  
  
% Time Response Results:  
  
% K1= 1 B1= 0.5 M1= 500  
% P controller:-  
  
% Rise Time :2.5448  
% settling time:78.1524  
% Overshoot:70.2118  
% PeakTime: 7.0248  
  
% I controller:-  
  
% Rise Time :NaN  
% settling time:NaN  
% Overshoot:NaN  
% PeakTime:Inf  
  
% D controller:-  
  
% Rise Time :0  
% settling time:81.5509  
% Overshoot: Inf  
% PeakTime: 3.5124

## Tool Analysis:

B1= 0.5  
M1= 5;  
K1 =1;  
P=5;  
sys = tf([P\*K1/M1],[1,B1/M1,K1/M1])  
subplot(4,3,1);  
impulse(sys);  
title('Impulse Input for k');  
subplot(4,3,2);  
step(sys);  
title('Step Input for k');  
subplot(4,3,3);  
[z,p,k]= tf2zp([P\*K1/M1],[1,B1/M1,K1/M1])  
pzmap(sys)  
subplot(4,3,10);  
bode(sys)  
hold on;  
S = stepinfo(sys)  
  
sys = tf([P\*K1/M1],[1,B1/M1,K1/M1,0])  
subplot(4,3,4);  
impulse(sys);  
title('Impulse Input for 1/s');  
subplot(4,3,5);  
step(sys);  
title('Step Input for 1/s');  
subplot(4,3,6);  
[z,p,k]= tf2zp([P\*K1/M1],[1,B1/M1,K1/M1,0])  
pzmap(sys)  
subplot(4,3,11);  
bode(sys)  
hold on;  
S = stepinfo(sys)  
  
sys = tf([P\*K1/M1,0],[1,B1/M1,K1/M1])  
subplot(4,3,7);  
impulse(sys);  
title('Impulse Input for s');  
subplot(4,3,8);  
step(sys);  
title('Step Input for s');  
subplot(4,3,9);  
[z,p,k]= tf2zp([P\*K1/M1,0],[1,B1/M1,K1/M1])  
pzmap(sys)  
subplot(4,3,12);  
bode(sys)  
hold on;  
S = stepinfo(sys)

B1 =  
  
 0.5000  
  
  
sys =  
   
 1  
 -----------------  
 s^2 + 0.1 s + 0.2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.0500 + 0.4444i  
 -0.0500 - 0.4444i  
  
  
k =  
  
 1  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 2.5448  
 SettlingTime: 78.1524  
 SettlingMin: 2.5361  
 SettlingMax: 8.5106  
 Overshoot: 70.2118  
 Undershoot: 0  
 Peak: 8.5106  
 PeakTime: 7.0248  
  
  
sys =  
   
 1  
 ---------------------  
 s^3 + 0.1 s^2 + 0.2 s  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0.0000 + 0.0000i  
 -0.0500 + 0.4444i  
 -0.0500 - 0.4444i  
  
  
k =  
  
 1  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
sys =  
   
 s  
 -----------------  
 s^2 + 0.1 s + 0.2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0  
  
  
p =  
  
 -0.0500 + 0.4444i  
 -0.0500 - 0.4444i  
  
  
k =  
  
 1  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 0  
 SettlingTime: 81.5509  
 SettlingMin: -1.3280  
 SettlingMax: 1.8877  
 Overshoot: Inf  
 Undershoot: Inf  
 Peak: 1.8877  
 PeakTime: 3.5124



## Comparison Analysis:

%P-Controller:  
%Using p controller the system responses doesnt change and only magnitude  
%of response is increased and as the poles are complex conjugate and negetive hence system is stable  
%I-controller:  
%Using I controller a pole is added at zero and makes the system  
%marginally stable so we dont get raise time,settling time and overshoot.  
%D-controller:  
%Using differentiator controller a zero is added and as we know zero is  
%root of the numerator of the transfer function and it introduces a  
%pronounced peak to peak system response and hence the systems peak  
%overshoot may increase.  
  
%Speed: System 3 is having low raise time and is therefore speed system.  
%Stability: Settling time of system of system 1 is less and therefore is  
%the stable system.  
%Accuracy: Settling time of system of system 1 is less and therefore is  
%more accurate.

# 2(c) Second Order MSD Equation

## Plant Description

The Mass-damper Spring Second order system is taken as Plant. It is used in as suspension.

% Equation: Mx''(t)+ Bx'(t) + Kx(t)= Kf(t).  
% f = force; B= coefficient of friction; M = mass ; v= velocity; k=spring constant.  
% Values: K1= 0.9 B1= 0.4 M1=1000 Wn=0.03 ; K2= 1 B2= 0.5 M2= 500 Wn=0.44; K3= 3 B3= 1.7 M3= 340 Wn=0.09;

## Math Analysis:

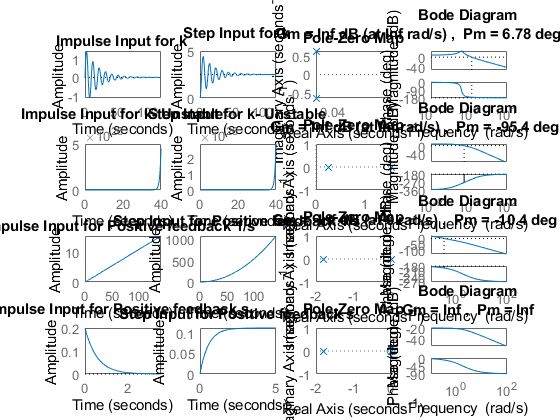
Independent: Time(t) Dependent: Velocity(v) and Force(f) Constant: Mass(M), Frictional Coefficient(B), Spring constant(K)

% Roots:((-B/M)+-sqrt(sq(B/M)-4K/M))/2  
  
% Poles and Zero Calculation:  
  
  
% IVT:  
% 1. For step input: 0  
% 2. For impulse input: 0  
  
% FVT:  
% 1. For step input: 1  
% 2. For impulse input: K/M

## Tool Analysis:

clc;  
%For negative feedback  
B1= 0.5  
M1= 5;  
K1 =1;  
P=5;  
  
sys = tf([P\*K1],[M1,B1,2\*K1])  
subplot(4,4,1);  
impulse(sys);  
title('Impulse Input for k');  
subplot(4,4,2);  
step(sys);  
title('Step Input for k');  
subplot(4,4,3);  
[z,p,k]= tf2zp([P\*K1],[M1,B1,2\*K1])  
pzmap(sys)  
subplot(4,4,4)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
  
hold on;  
S = stepinfo(sys)  
  
B2= -9  
M2= 5;  
K2=1;  
P2=5;  
  
sys = tf([P2\*K2],[M2,B2,2\*K2])  
subplot(4,4,5);  
impulse(sys);  
title('Impulse Input for k- Unstable');  
subplot(4,4,6);  
step(sys);  
title('Step Input for k- Unstable');  
subplot(4,4,7);  
[z,p,k]= tf2zp([P2\*K2],[M2,B2,2\*K2])  
pzmap(sys)  
subplot(4,4,8)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)  
  
% For Positive feedback using I & D  
  
B3= 9  
M3= 5;  
K3=1;  
  
  
sys = tf([K3],[M3,B3,0,0])  
subplot(4,4,9);  
impulse(sys);  
title('Impulse Input for Positive feedback 1/s ');  
subplot(4,4,10);  
step(sys);  
title('Step Input for Positive feedback 1/s');  
subplot(4,4,11);  
[z,p,k]= tf2zp([K3],[M3,B3,0,0])  
pzmap(sys)  
subplot(4,4,12)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)  
  
  
B4= 9  
M4= 5;  
K4=1;  
  
  
sys = tf([K4,0],[M4,B4,0])  
subplot(4,4,13);  
impulse(sys);  
title('Impulse Input for Positive feedback s ');  
subplot(4,4,14);  
step(sys);  
title('Step Input for Positive feedback s');  
subplot(4,4,15);  
[z,p,k]= tf2zp([K4,0],[M4,B4,0])  
pzmap(sys)  
subplot(4,4,16)  
bode(sys)  
margin(sys)  
[Gm,Pm,Wcg,Wcp] = margin(sys)  
hold on;  
S = stepinfo(sys)

B1 =  
  
 0.5000  
  
  
sys =  
   
 5  
 -----------------  
 5 s^2 + 0.5 s + 2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.0500 + 0.6305i  
 -0.0500 - 0.6305i  
  
  
k =  
  
 1  
  
  
Gm =  
  
 Inf  
  
  
Pm =  
  
 6.7782  
  
  
Wcg =  
  
 Inf  
  
  
Wcp =  
  
 1.1803  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 1.7526  
 SettlingTime: 75.6433  
 SettlingMin: 0.9814  
 SettlingMax: 4.4486  
 Overshoot: 77.9429  
 Undershoot: 0  
 Peak: 4.4486  
 PeakTime: 4.9673  
  
  
B2 =  
  
 -9  
  
  
sys =  
   
 5  
 ---------------  
 5 s^2 - 9 s + 2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 1.5403  
 0.2597  
  
  
k =  
  
 1  
  
Warning: The closed-loop system is  
unstable.   
  
Gm =  
  
 Inf  
  
  
Pm =  
  
 -95.4008  
  
  
Wcg =  
  
 Inf  
  
  
Wcp =  
  
 0.5531  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
B3 =  
  
 9  
  
  
sys =  
   
 1  
 -------------  
 5 s^3 + 9 s^2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0  
 0  
 -1.8000  
  
  
k =  
  
 0.2000  
  
Warning: The closed-loop system is  
unstable.   
  
Gm =  
  
 0  
  
  
Pm =  
  
 -10.4065  
  
  
Wcg =  
  
 0  
  
  
Wcp =  
  
 0.3306  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
B4 =  
  
 9  
  
  
sys =  
   
 s  
 -----------  
 5 s^2 + 9 s  
   
Continuous-time transfer function.  
  
  
z =  
  
 0  
  
  
p =  
  
 0  
 -1.8000  
  
  
k =  
  
 0.2000  
  
Warning: The closed-loop system is  
unstable.   
  
Gm =  
  
 Inf  
  
  
Pm =  
  
 Inf  
  
  
Wcg =  
  
 NaN  
  
  
Wcp =  
  
 NaN  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 1.2206  
 SettlingTime: 2.1734  
 SettlingMin: 0.1005  
 SettlingMax: 0.1111  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 0.1111  
 PeakTime: 5.8588



## Comparison Analysis:

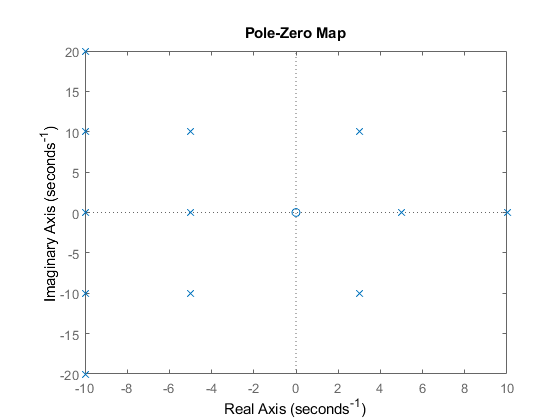
%Negetive feedback with P-controller:  
%Using p controller the system responses doesnt change and only magnitude  
%of response is increased and as the poles are complex conjugate and negetive hence system is stable  
  
%Positive feedback with P-controller:  
%here poles are complex conjugate and are lying on the right hand side of s  
%plane and hence the system is unstable.  
  
%Positive feedback with I\_controller:  
%Using I controller a pole is added at zero and makes the system  
%unstable as %GM=9;PM=negetive hence stability of system is affetced and becomes a  
%unstable system.  
  
%positive feedback with D-controller:  
%Using differentiator controller a zero is added at the origin and  
%therefore the system becomes stable.  
  
  
%Speed: System 1 is having low raise time and is therefore speed system.  
%Stability: Settling time of system of system 4 is less and therefore is  
%the stable system.  
%Accuracy: Settling time and raise time are less of system 4 is less and therefore is  
%more accurate.

# 2(d) Movement of Poles.

## Description: Here the movement of poles is shown along the real and imaginary axis .

poles = [-10+20i -10-20i -5+10i -5-10i -10+10i -10-10i 3+10i 3-10i -5+0i +5+0i -10+0i +10-0i ];  
  
zeros = [0 0];  
  
gain = 0.9;  
  
s=zpk(zeros,poles,gain);  
  
pzplot(s)  
  
[wn,zeta] = damp(s)

wn =  
  
 5.0000  
 5.0000  
 10.0000  
 10.0000  
 10.4403  
 10.4403  
 11.1803  
 11.1803  
 14.1421  
 14.1421  
 22.3607  
 22.3607  
  
  
zeta =  
  
 1.0000  
 -1.0000  
 1.0000  
 -1.0000  
 -0.2873  
 -0.2873  
 0.4472  
 0.4472  
 0.7071  
 0.7071  
 0.4472  
 0.4472



## Analysis :

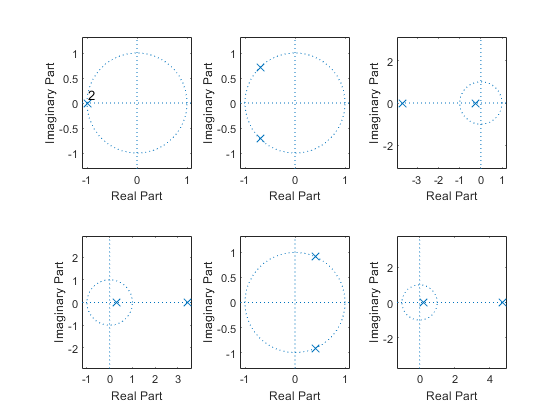
1. If we move the roots along Wn, then frequency will increase and overshoot remains same. 2. If we move the poles along jw axis, then overshoot increases and frequency also increases. 3. If we move along zetaWn axis or x-axis. 3a. If we move to right hand side then overshoot increases and frequency decreases. 3b. If we move to left hand side of s-plane overshoot decreases and frequency increases.

# 2(e) Roots of the Standard Equation

## Code

clc;  
zeta=1;  
TF=tf([1],[1,(2\*zeta),1])  
sys = tf([1],[1,(2\*zeta),1])  
figure  
subplot(2,3,1)  
S = stepinfo(sys)  
[z,p,k]= tf2zp([1],[1,(2\*zeta),1])  
zplane(z,p)  
  
zeta=0.7 ;  
TF=tf([1],[1,(2\*zeta),1])  
sys = tf([1],[1,(2\*zeta),1])  
subplot(2,3,2)  
S = stepinfo(sys)  
[z,p,k]= tf2zp([1],[1,(2\*zeta),1])  
zplane(z,p)  
  
zeta=2;  
TF=tf([1],[1,(2\*zeta),1])  
sys = tf([1],[1,(2\*zeta),1])  
subplot(2,3,3)  
S = stepinfo(sys)  
[z,p,k]= tf2zp([1],[1,(2\*zeta),1])  
zplane(z,p)  
  
zeta=-1.85;  
TF=tf([1],[1,(2\*zeta),1])  
sys = tf([1],[1,(2\*zeta),1])  
subplot(2,3,4)  
S = stepinfo(sys)  
[z,p,k]= tf2zp([1],[1,(2\*zeta),1])  
zplane(z,p)  
  
zeta=-0.4;  
TF=tf([1],[1,(2\*zeta),1])  
sys = tf([1],[1,(2\*zeta),1])  
subplot(2,3,5)  
S = stepinfo(sys)  
[z,p,k]= tf2zp([1],[1,(2\*zeta),1])  
zplane(z,p)  
  
zeta=-2.45;  
TF=tf([1],[1,(2\*zeta),1])  
sys = tf([1],[1,(2\*zeta),1])  
subplot(2,3,6)  
S = stepinfo(sys)  
[z,p,k]= tf2zp([1],[1,(2\*zeta),1])  
zplane(z,p)

TF =  
   
 1  
 -------------  
 s^2 + 2 s + 1  
   
Continuous-time transfer function.  
  
  
sys =  
   
 1  
 -------------  
 s^2 + 2 s + 1  
   
Continuous-time transfer function.  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 3.3579  
 SettlingTime: 5.8339  
 SettlingMin: 0.9000  
 SettlingMax: 0.9994  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 0.9994  
 PeakTime: 9.7900  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -1  
 -1  
  
  
k =  
  
 1  
  
  
TF =  
   
 1  
 ---------------  
 s^2 + 1.4 s + 1  
   
Continuous-time transfer function.  
  
  
sys =  
   
 1  
 ---------------  
 s^2 + 1.4 s + 1  
   
Continuous-time transfer function.  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 2.1268  
 SettlingTime: 5.9789  
 SettlingMin: 0.9001  
 SettlingMax: 1.0460  
 Overshoot: 4.5986  
 Undershoot: 0  
 Peak: 1.0460  
 PeakTime: 4.4078  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -0.7000 + 0.7141i  
 -0.7000 - 0.7141i  
  
  
k =  
  
 1  
  
  
TF =  
   
 1  
 -------------  
 s^2 + 4 s + 1  
   
Continuous-time transfer function.  
  
  
sys =  
   
 1  
 -------------  
 s^2 + 4 s + 1  
   
Continuous-time transfer function.  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 8.2308  
 SettlingTime: 14.8789  
 SettlingMin: 0.9017  
 SettlingMax: 0.9993  
 Overshoot: 0  
 Undershoot: 0  
 Peak: 0.9993  
 PeakTime: 27.3269  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 -3.7321  
 -0.2679  
  
  
k =  
  
 1  
  
  
TF =  
   
 1  
 ---------------  
 s^2 - 3.7 s + 1  
   
Continuous-time transfer function.  
  
  
sys =  
   
 1  
 ---------------  
 s^2 - 3.7 s + 1  
   
Continuous-time transfer function.  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 3.4064  
 0.2936  
  
  
k =  
  
 1  
  
  
TF =  
   
 1  
 ---------------  
 s^2 - 0.8 s + 1  
   
Continuous-time transfer function.  
  
  
sys =  
   
 1  
 ---------------  
 s^2 - 0.8 s + 1  
   
Continuous-time transfer function.  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0.4000 + 0.9165i  
 0.4000 - 0.9165i  
  
  
k =  
  
 1  
  
  
TF =  
   
 1  
 ---------------  
 s^2 - 4.9 s + 1  
   
Continuous-time transfer function.  
  
  
sys =  
   
 1  
 ---------------  
 s^2 - 4.9 s + 1  
   
Continuous-time transfer function.  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 4.6866  
 0.2134  
  
  
k =  
  
 1



## Comparison Analysis:

% 1st value lise on negative x axis means: Critically-damped case & stable  
% 2nd value lise in 2nd & 3rd quadrant means: Under-damp case & stable  
% 3rd value lise on negative x axis means: Overdamped case & stable  
% 4th value lise on positive x axis means: unstable  
% 5th value lise on 1st & 4th quadrant means: unstable  
% 6th value lise on positive x axis means: unstable

# 2(f) PID Analysis Second Order PID controller:

Author: Mohammed Ijaz PS Number: 99003728 Date: 7th April 2021. Version: Matlab 2020

Second Order System PID 1

Analysis of Second Order system with PID controllers 2

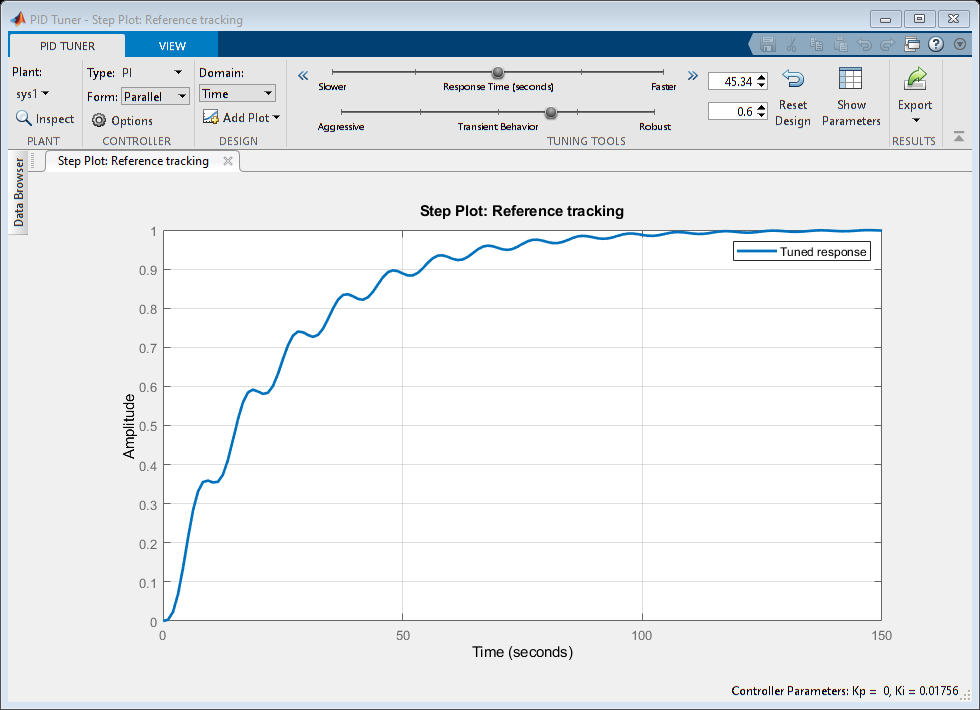
First Order System PID Code 3

Analysis of First Order system with PID controllers 4

## Second Order System PID

B1= 0.5  
M1= 5;  
K1 =1;  
P1=5;  
  
sys1 = tf([P1\*K1],[M1,B1,2\*K1])  
pidTuner(sys1)

B1 =  
  
 0.5000  
  
  
sys1 =  
   
 5  
 -----------------  
 5 s^2 + 0.5 s + 2  
   
Continuous-time transfer function.



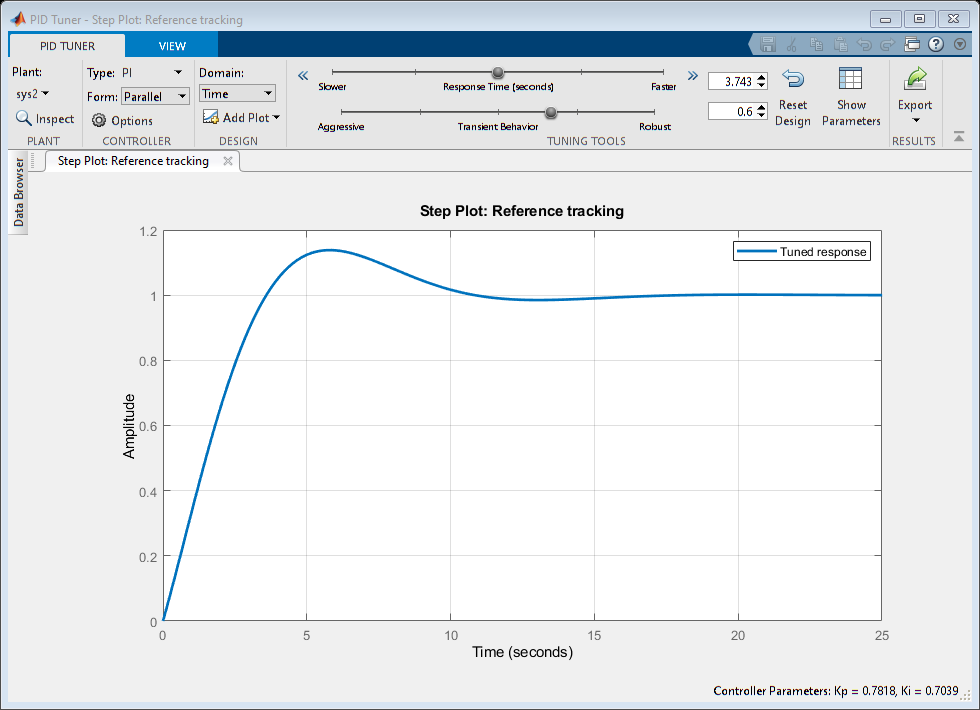
## Analysis of Second Order system with PID controllers

% PI Controller  
% Ideal values given by the pid tuner:  
% Kp =0 Ki =0.0174  
% Rise time=50.4s  
% Settling time= 93.4s  
% Overshoot= 0.00872%  
  
  
% To make 0% overshoot we are decreasing the speed of the system  
  
  
% The new values are given below:  
% Kp =0 Ki =0.015  
% Rise time=51.1s  
% Settling time= 94.3s  
% Overshoot= 0  
  
  
% PD Controller  
% Ideal values given by the pid tuner:  
% Kp =27.35 Kd =6.251  
% Rise time=0.017s  
% Settling time= 1.35s  
% Overshoot= 24.7%  
  
  
% To make overshoot 0% we are increasing the speed.  
% The new values are given below:  
  
  
% Kp =26.97 Kd =6.34  
% Rise time=0.0179s  
% Settling time= 0.13s  
% Overshoot= 0%  
  
  
% PID Controller  
% Ideal values given by the pid tuner:  
  
  
% This is the best possible reponse given by the system i.e. we cannot  
% decrease the overshoot further more.  
  
  
% Kp =26.97 Kd =6.34 Ki =0.68  
% Rise time=0.495s  
% Settling time= 9.3s  
% Overshoot= 12.4%

## First Order System PID Code

B2= 0.5;  
M2= 5;  
P2 = 2;  
  
  
sys2 = tf([P2],[M2,B2+1])  
pidTuner(sys2)

sys2 =  
   
 2  
 ---------  
 5 s + 1.5  
   
Continuous-time transfer function.



## Analysis of First Order system with PID controllers

PI Controller Ideal values given by the pid tuner: Kp =0.781 Ki =0.70 Rise time=2.7s Settling time= 9.87s Overshoot= 13.8%

% Changing the values of Kp and ki to get the best possible stability of  
% the system i.e decreasing the overshoot.  
  
  
% The new values are given below:  
% Kp =1.25 Ki =0.46  
% Rise time=3.59s  
% Settling time= 5.39s  
% Overshoot= 1.33%  
  
  
% PD Controller  
% Ideal values given by the pid tuner:  
% Kp =53.18 Kd =0  
% Rise time=0.102s  
% Settling time= 0.181s  
% Overshoot= 0%  
  
  
% As we are getting the overshoot 0% so we don't need to change any  
% parameters.  
  
  
% PID Controller  
% Ideal values given by the pid tuner:  
  
  
% Kp =1.07 Kd =0 Ki =0.53  
% Rise time=3.04s  
% Settling time= 10.6s  
% Overshoot= 6.08%  
  
  
% The above parameters gives the best response for the system so we don't  
% need to change any parameters.

# 3(a) Second Order Exponential Decay system

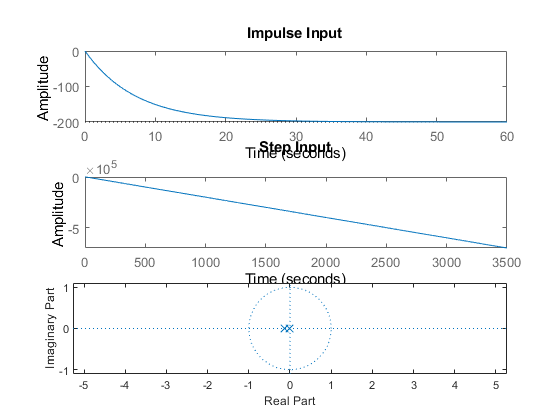
## Plant Description

It is a exponential decay system of a radioactive material Equation- dM/dt=-kA(e^-kt) M=mass, k=constant, A= non zero constant, t=time Values- k=0.14, A=200

## Without Controller

clc;  
k= 0.14;  
A= 200;  
sys = tf([-k\*A],[1,k,0])  
figure(1);  
subplot(3,1,1);  
impulse(sys);  
title('Impulse Input');  
subplot(3,1,2);  
step(sys);  
title('Step Input');  
[z,p,k] = tf2zp([-k\*A],[1,k,0])  
subplot(3,1,3);  
zplane(z,p);  
S = stepinfo(sys)

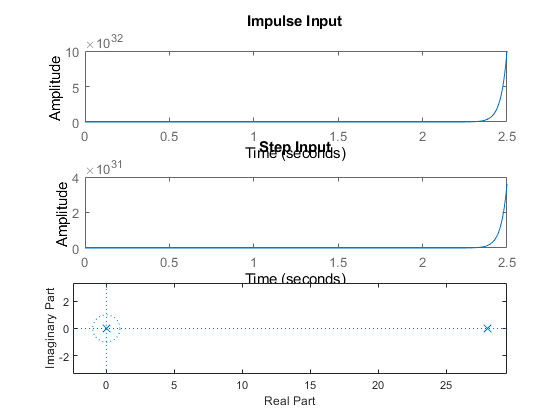
sys =  
   
 -28  
 ------------  
 s^2 + 0.14 s  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0  
 -0.1400  
  
  
k =  
  
 -28.0000  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf



## Open Loop with Controller (P)

P= 2;  
sys = tf([P\*(-k)\*A],[1,k,0])  
figure(2);  
subplot(3,1,1);  
impulse(sys);  
title('Impulse Input');  
subplot(3,1,2);  
step(sys);  
title('Step Input');  
[z,p,k] = tf2zp([P\*(-k)\*A],[1,k,0])  
subplot(3,1,3);  
zplane(z,p);  
S = stepinfo(sys)

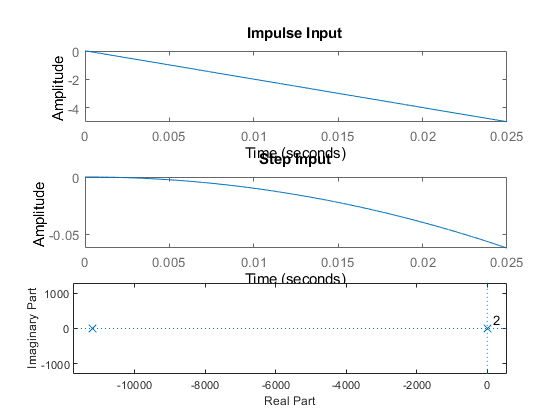
sys =  
   
 1.12e04  
 ----------  
 s^2 - 28 s  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 0  
 28.0000  
  
  
k =  
  
 1.1200e+04  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf



## Open Loop with Controller (I)

sys = tf([(-k)\*A],[1,k,0,0])  
figure(3);  
subplot(3,1,1);  
impulse(sys);  
title('Impulse Input');  
subplot(3,1,2);  
step(sys);  
title('Step Input');  
[z,p,k] = tf2zp([-k\*A],[1,k,0,0])  
subplot(3,1,3);  
zplane(z,p);  
S = stepinfo(sys)

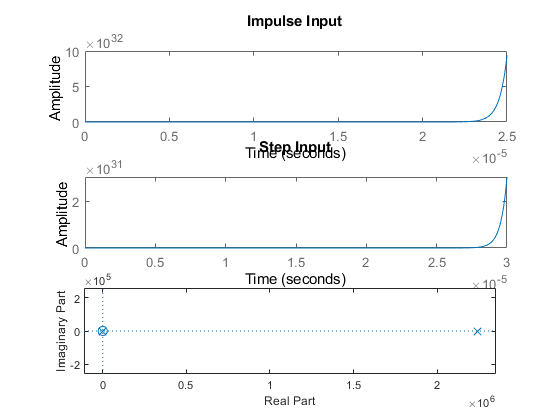
sys =  
   
 -2.24e06  
 -----------------  
 s^3 + 1.12e04 s^2  
   
Continuous-time transfer function.  
  
  
z =  
  
 0×1 empty double column vector  
  
  
p =  
  
 1.0e+04 \*  
  
 0  
 0  
 -1.1200  
  
  
k =  
  
 -2.2400e+06  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf



## Closed Loop- Negative feedback with Controller (D)

sys = tf([(-k)\*A,0],[1,k,(-k)\*A])  
figure(4);  
subplot(3,1,1);  
impulse(sys);  
title('Impulse Input');  
subplot(3,1,2);  
step(sys);  
title('Step Input');  
[z,p,k] = tf2zp([-k\*A,0],[1,k,(-k)\*A])  
subplot(3,1,3);  
zplane(z,p);  
S = stepinfo(sys)

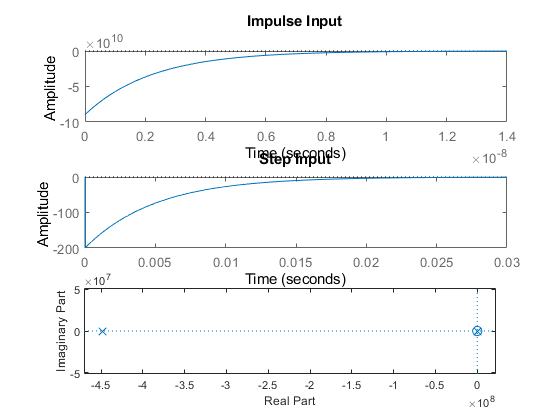
sys =  
   
 4.48e08 s  
 -------------------------  
 s^2 - 2.24e06 s + 4.48e08  
   
Continuous-time transfer function.  
  
  
z =  
  
 0  
  
  
p =  
  
 1.0e+06 \*  
  
 2.2398  
 0.0002  
  
  
k =  
  
 4.4800e+08  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: NaN  
 SettlingTime: NaN  
 SettlingMin: NaN  
 SettlingMax: NaN  
 Overshoot: NaN  
 Undershoot: NaN  
 Peak: Inf  
 PeakTime: Inf



## Closed Loop- Positive feedback with Controller (D)

sys = tf([(-k)\*A,0],[1,k,k\*A])  
figure(5);  
subplot(3,1,1);  
impulse(sys);  
title('Impulse Input');  
subplot(3,1,2);  
step(sys);  
title('Step Input');  
[z,p,k] = tf2zp([-k\*A,0],[1,k,k\*A])  
subplot(3,1,3);  
zplane(z,p);  
S = stepinfo(sys)

sys =  
   
 -8.96e10 s  
 -------------------------  
 s^2 + 4.48e08 s + 8.96e10  
   
Continuous-time transfer function.  
  
  
z =  
  
 0  
  
  
p =  
  
 1.0e+08 \*  
  
 -4.4800  
 -0.0000  
  
  
k =  
  
 -8.9600e+10  
  
  
S =   
  
 struct with fields:  
  
 RiseTime: 0  
 SettlingTime: 0.0196  
 SettlingMin: -199.9633  
 SettlingMax: -0.2598  
 Overshoot: Inf  
 Undershoot: Inf  
 Peak: 199.9633  
 PeakTime: 9.2103e-07



## Math Analysis

Independent: Time(t) Dependent: Mass(M) Constant: Non-zero constant(A), Constant(A)

## Comparison Analysis

1) System without controller behaves exactly like an exponential decay with the system decaying exponentially.

2) On adding a proportionality controller to system, the system becomes unstable.

3) On adding a Integrator controller to system, the response times have decreased hugely, making the system reach stability faster than a P controller.

4) Integrator controller adds a pole to zero also.

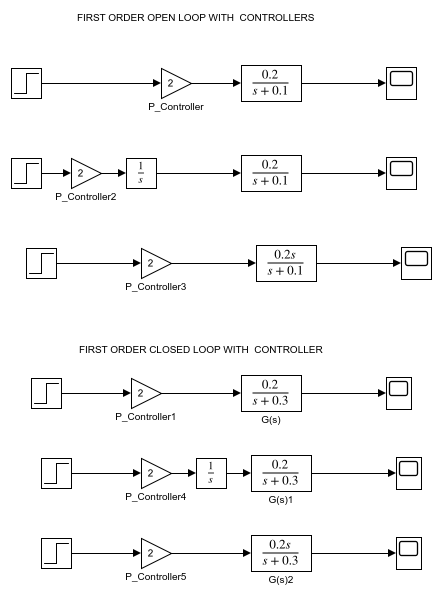
5) On addition of a differentiator controller in negative feedback the system becomes unstable.

6) A zero gets added at origin due to the differentiator.

7) On addition of a differentiator controller in positive feedback the system becomes stable.

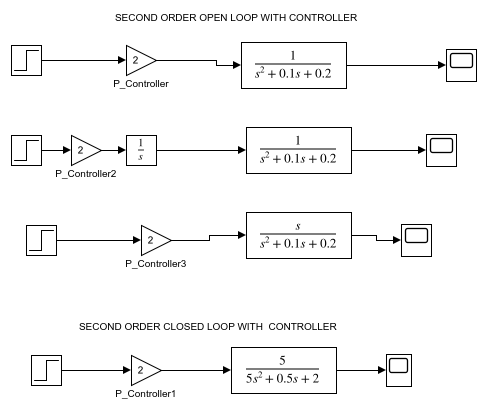
# 4(a) 1st Order Differential Equation Model

This was done in Simulink of MATLAB R2020b.



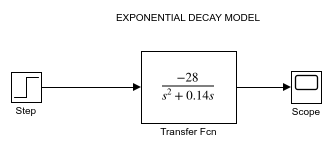
# 4(b) 2nd Order Differential Equation Model

This was done in Simulink of MATLAB R2020b.



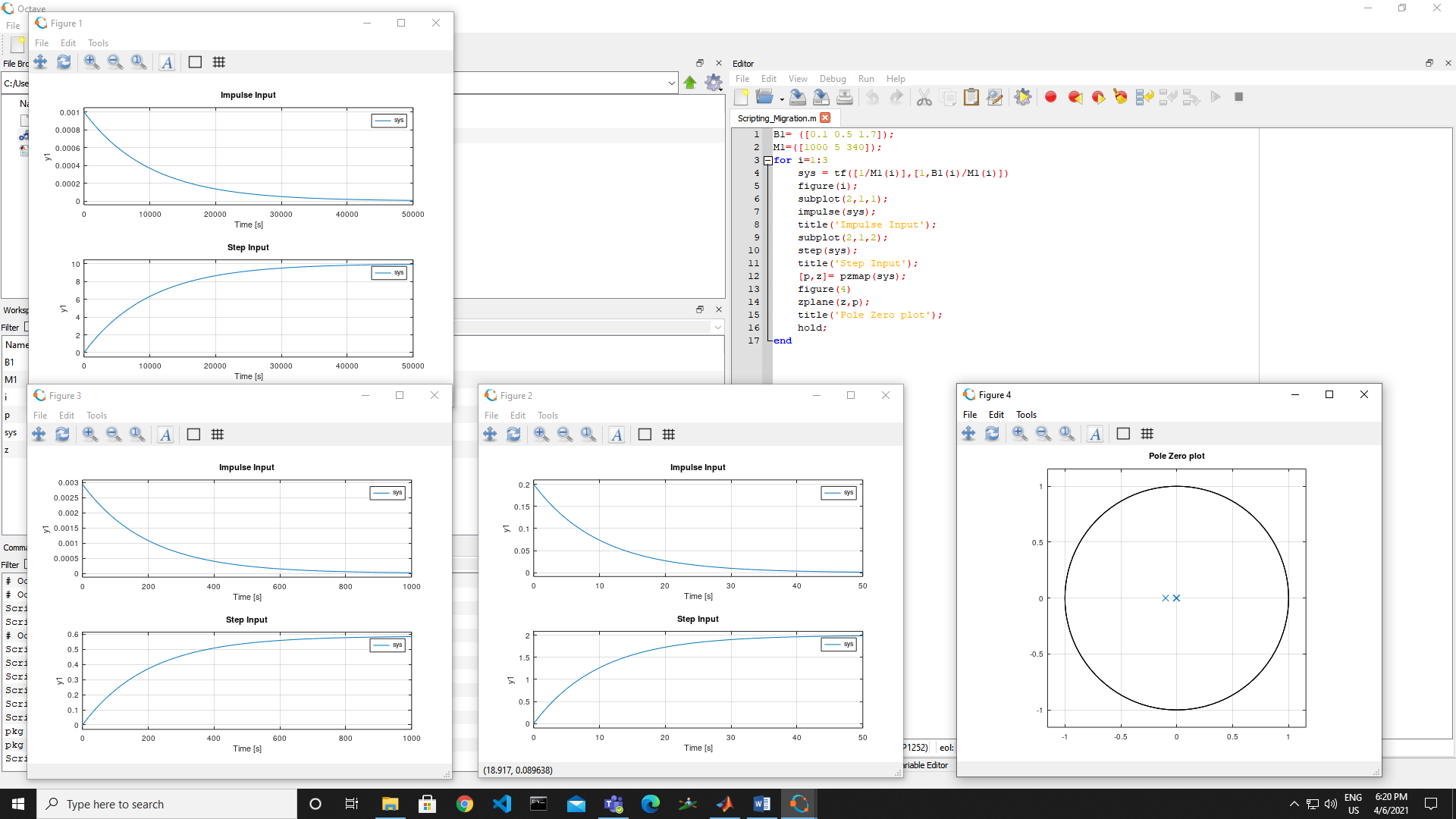
# 4(c) Exponential Decay- Radioactive Material Model

This was done in Simulink of MATLAB R2020b.



# 5(a) Migration of Scripts to GNU Octave

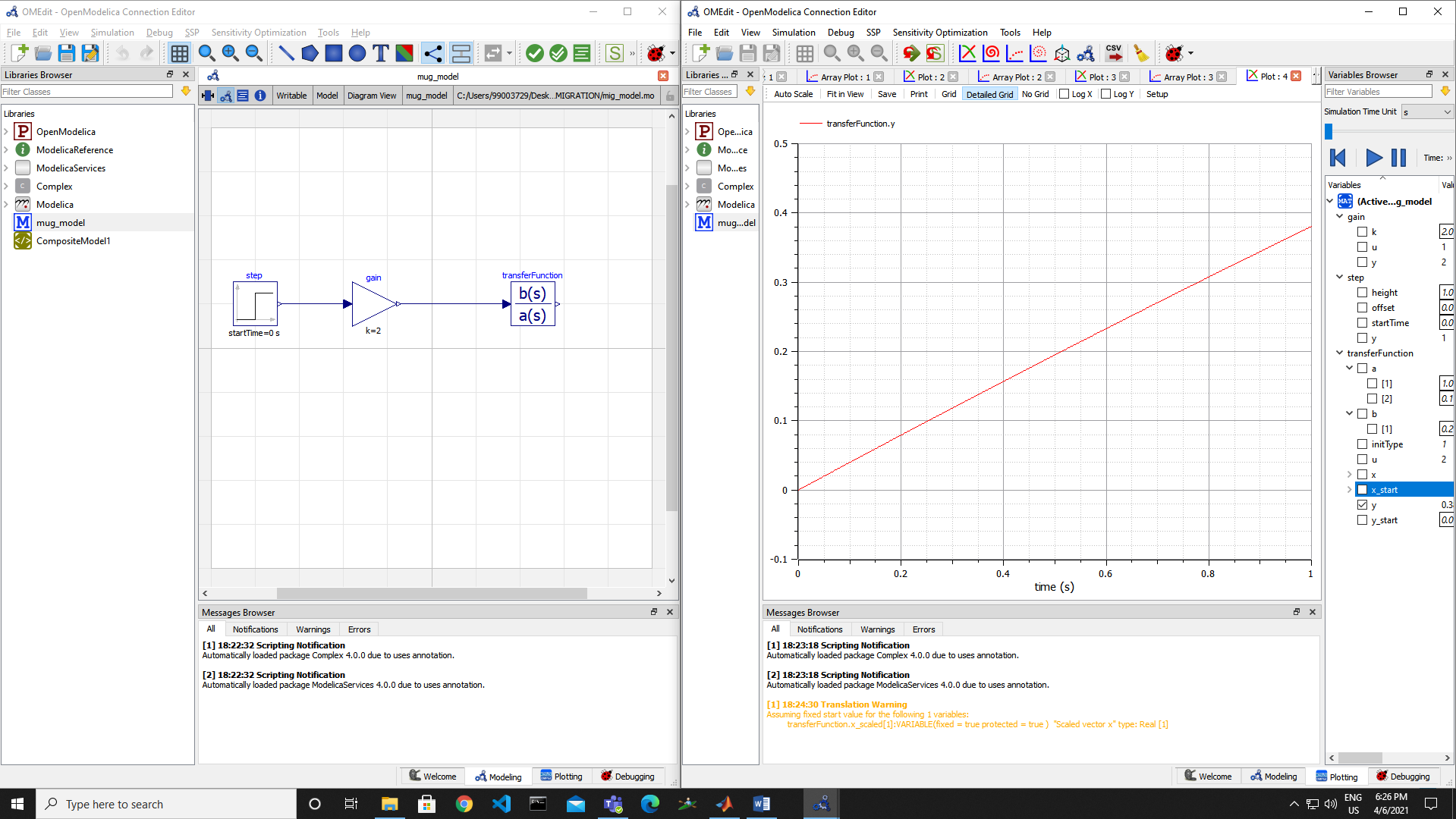
The results matched with the scripts executed in the MATLAB R2020b



GNU Octave is software featuring a high-level programming language, primarily intended for numerical computations. Octave helps in solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB.

# 5(b) Migration of Model to OpenModelica

The results matched with the models executed in Simulink of MATLAB R2020b.



OpenModelica is a free and open source environment based on the Modelica modeling language for modeling, simulating, optimizing and analyzing complex dynamic systems. This software is actively developed by Open Source Modelica Consortium, a non-profit, non-governmental organization.