Table of Contents


```
clc; clear all; close all;
```

Storing values of given parameters

```
m = 0.23; M = 0.5; l = 0.321; g = 9.8;
a = 1/(m+M);
A = [ 0 1 0 0; 3*g/(4*l-3*m*l*a) 0 0 0; 0 0 0 1; -3*m*a*g/(4-3*m*a) 0
 0 0];
B = [0; 3*a/(3*m*l*a-4*l); 0; 4*a/(4-3*m*a)];
C = [1 0 0 0];
```

Defining the open-loop transfer function

```
s = tf('s');
gs = C*(inv(s*eye(4)-A))*B


gs =

           -4.191
  ------------------------
  s^2 - 1.51e-14 s - 29.98

Continuous-time transfer function.
```

Finding parameters and dominant poles by given specifications

```
Mp = 0.2; set_time = 1;
     % Given Peak overshoot and settling time
zeta = sqrt((log(Mp)^2/(pi^2+(log(Mp))^2)))
     % Mp = exp((-pi*zeta)/(sqrt(1-zeta^2)))
w_n = 4/(set_time*zeta)
     % settling time = 4/(w_n*zeta)
d_p1 = -w_n*zeta + j*w_n*sqrt(1-zeta^2)
     % desired dominant pole 1
```

```
d_p2 = -w_n*zeta - j*w_n*sqrt(1-zeta^2)
     % desired dominant pole 2


zeta =

    0.4559


w_n =

    8.7729


d_p1 =

  -4.0000 + 7.8079i


d_p2 =

  -4.0000 - 7.8079i
```

Angle contribution by desired dominant poles and compensator

```
P = pole(gs); Z = zero(gs);
     % to store the open-loop poles and zeroes respectively
mm = length(Z); n = length(P);
     % to store the number of zeroes and poles respectively
% angle contribution = (sum of angles made with the open-loop zeroes)
 - (sum of angles made with the open-loop poles)
phi = sum(angle(d_p1-Z)) - sum(angle(d_p1-P));
     % phi is in radians
phi*180/pi
     % Printing phi (in degrees)


ans =

 -219.8095
```

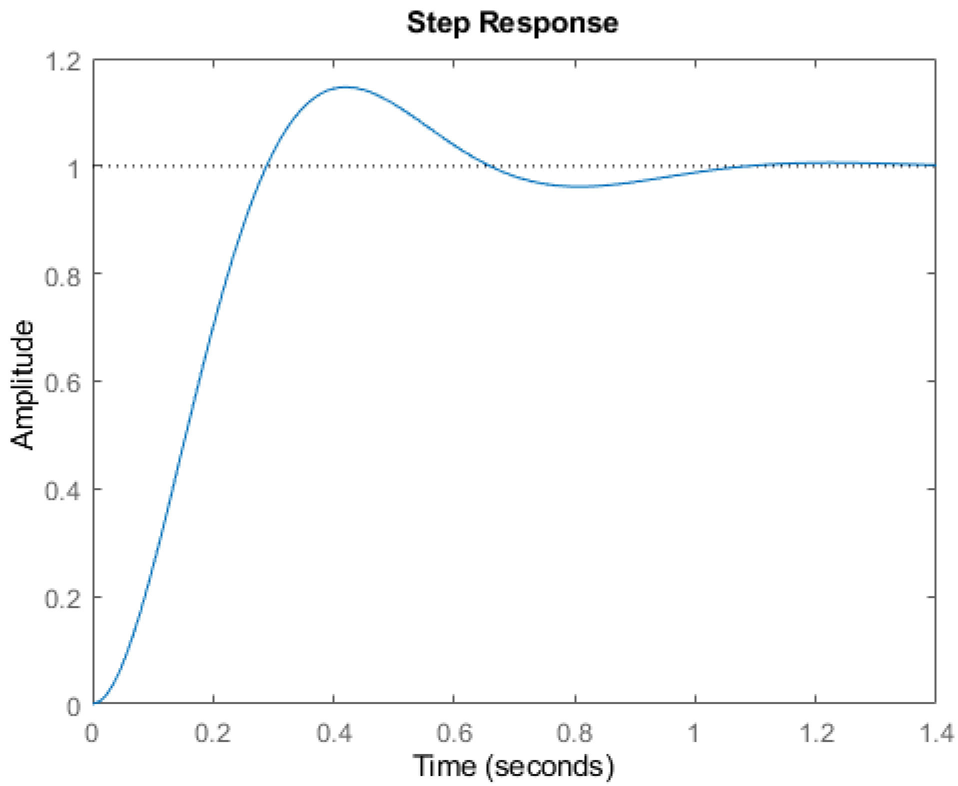
Designing Lead Compensator

```
cz = -5.0;
     % choosing appropriate compensator zero
phi = phi + angle(d_p1-cz);
       % adding angle made by the dominant pole with the compensator
 zero
```

```
cp_angle = phi + pi;
     % angle made by the dominant pole with the compensator pole
cp = real(d_p1)-(imag(d_p1)/tan(cp_angle))
     % to find the compensator pole (considering cp_angle is +ve)

cs = tf((s-cz)/(s-cp))
     % compensator (without K)
ls = cs*gs
     % loop tranfer function (without K)
K = real(evalfr(-1/ls, d_p1))
     % to find the value of gain in compensator, K = 1/|L(s)|

ss = feedback(K*ls, 1)
     % The overall closed loop system thus formed

pole(ss)
     % Closed loops poles of the combined system
Gain = evalfr(ss, 0)
     % Feed Forward gain
figure(1);
stepplot(ss/Gain);
     % to plot the step response
stepinfo(ss/Gain)
     % to get the information of the step response
```

```
cp =

  -12.4047


cs =

   s + 5
  --------
  s + 12.4

Continuous-time transfer function.


ls =

          -4.191 s - 20.95
  --------------------------------
  s^3 + 12.4 s^2 - 29.98 s - 371.9

Continuous-time transfer function.


K =

  -33.9263
```

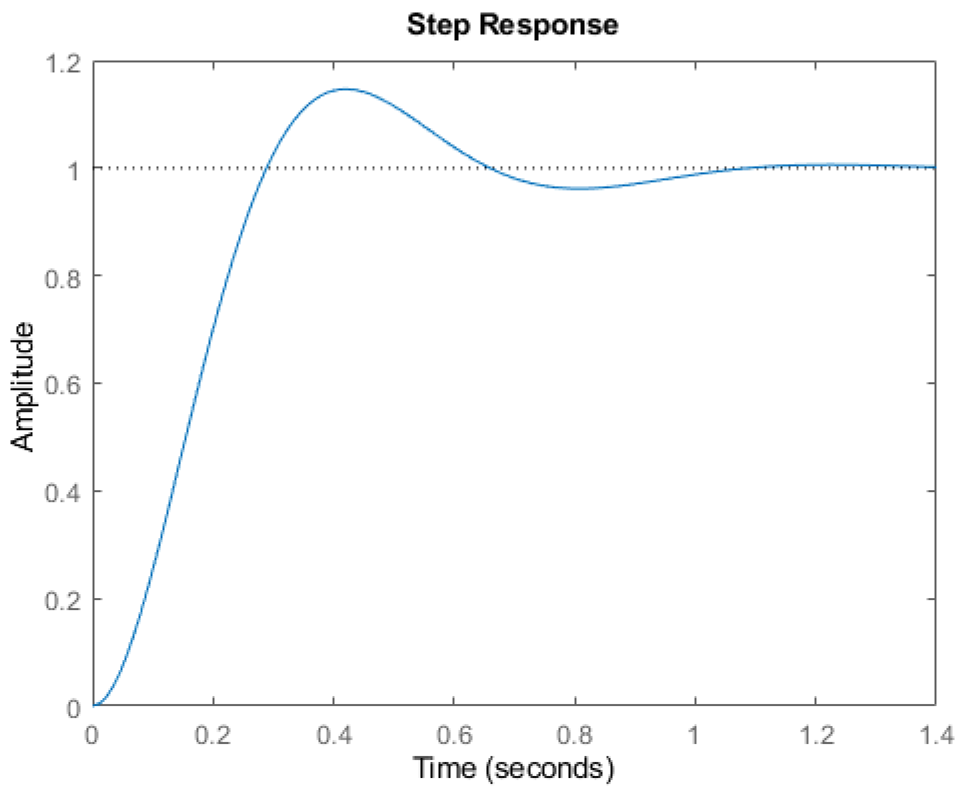
```
ss =

         142.2 s + 710.9
  ------------------------------
  s^3 + 12.4 s^2 + 112.2 s + 339

Continuous-time transfer function.


ans =

  -4.0000 + 7.8079i
  -4.0000 - 7.8079i
  -4.4047 + 0.0000i


Gain =

    2.0971


ans =

  struct with fields:

        RiseTime: 0.1946
    SettlingTime: 0.9548
     SettlingMin: 0.9352
     SettlingMax: 1.1465
       Overshoot: 14.6460
      Undershoot: 0
            Peak: 1.1465
        PeakTime: 0.4260
```

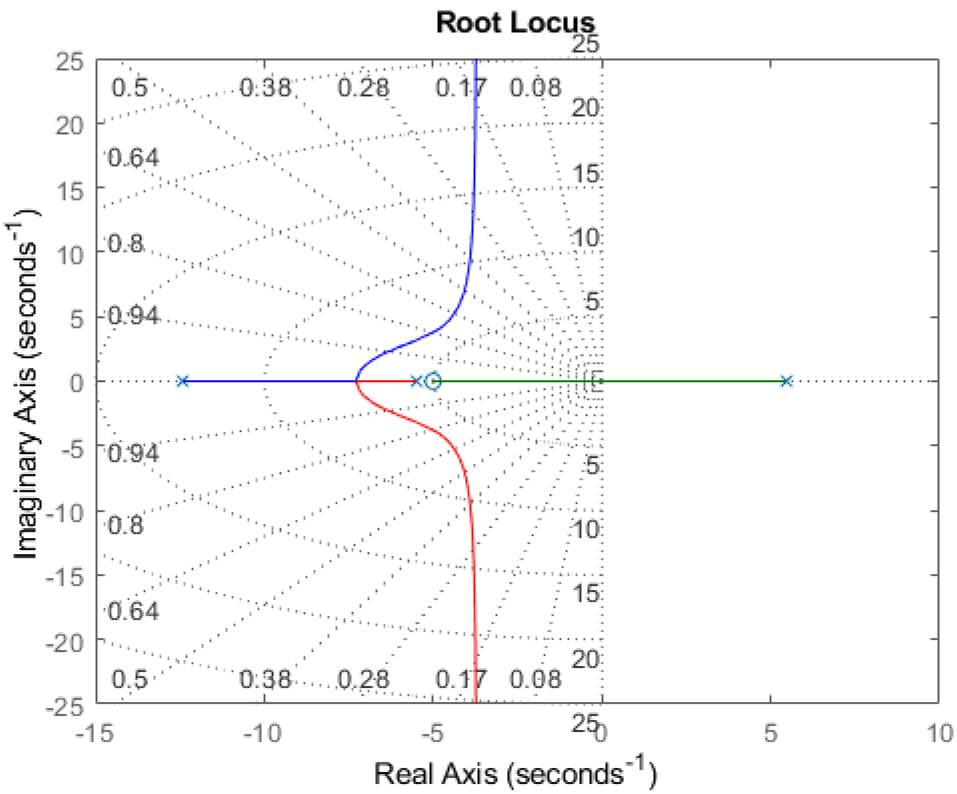


Root Locus of the compensated system

```
figure(2);
rlocus(-ls);
grid on;
delta = (length(pole(ls))-length(zero(ls)));
     % number of poles - number of zeroes
% Centroid
centroid = (sum(pole(ls))-sum(zero(ls)))/delta
      % centroid = {sum of poles - sum of zeroes}/(number of poles -
 number of zeroes)
% Asymptote angles
for i=1:delta
    asym(i)=(2*(i-1)+1)*180/delta;
end
asym
```

```
centroid =

   -3.7023


asym =

    90   270
```


Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4

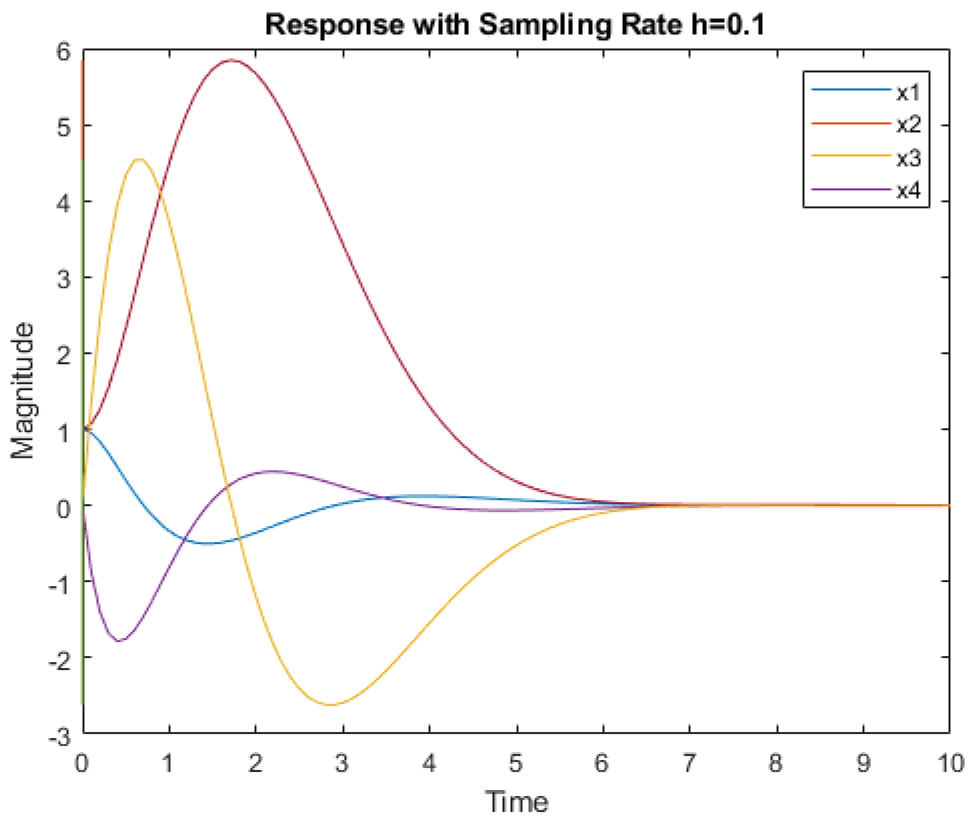


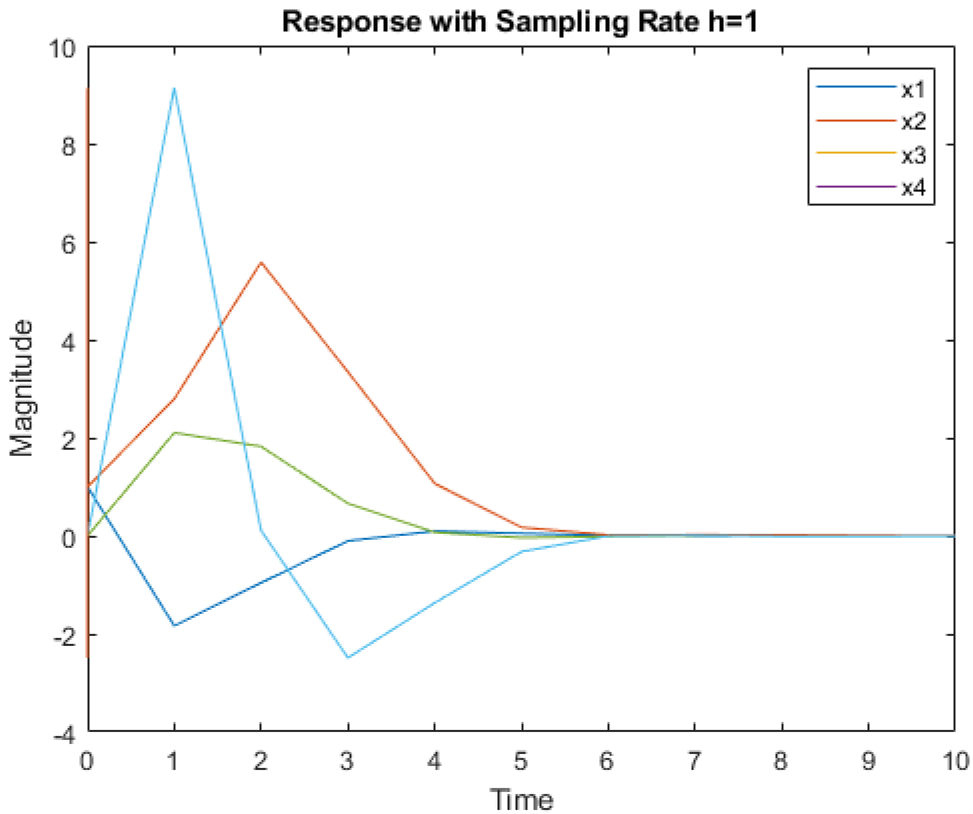
Section (c) - Runge Kutta Method

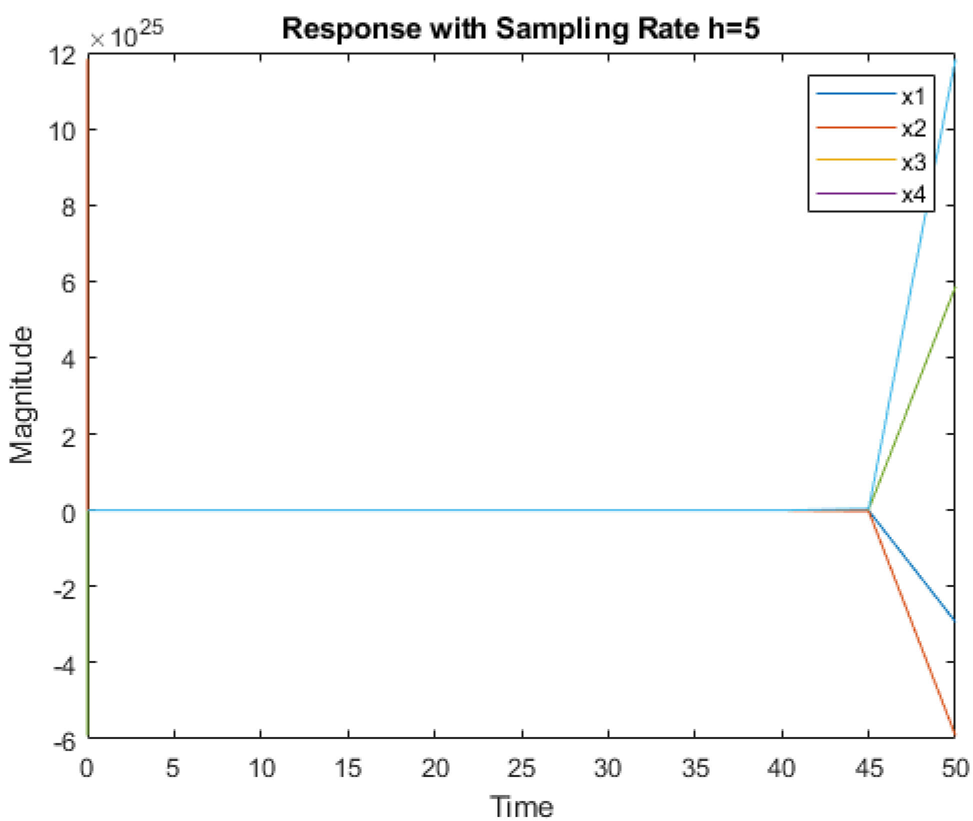
```
Kr = [-9.5818 -1.2973 -0.0974 -0.2435];
f = A - B*Kr;

% Part (a)
h = 0.1;
x = zeros(4,101);
time = zeros(101);
x(:,1) = [1; 0; 1; 0];
time(1)=0;
for i=1:100
    m0 = f*x(:,i);
    m1 = f*(x(:,i)+h*m0/2);
    m2 = f*(x(:,i)+h*m1/2);
    m3 = f*(x(:,i)+h*m2);
    x(:,i+1) = x(:,i) + h*(m0+2*m1+2*m2+m3)/6;
    time(i+1)=time(i)+h;
end
figure(3);
for i=1:4
    plot(time, x(i,:))
    hold on;
end
hold off;
```

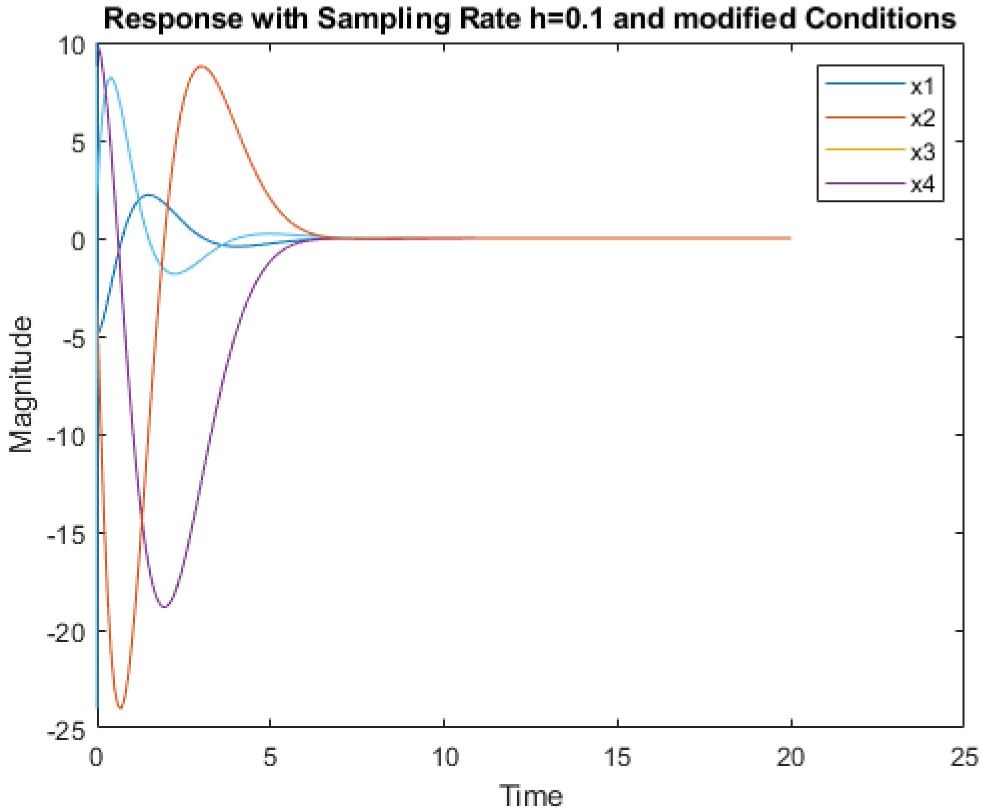
```
legend('x1', 'x2', 'x3', 'x4');
title('Response with Sampling Rate h=0.1');
ylabel('Magnitude');
xlabel('Time')

% Part (b)
h = 1;
x = zeros(4,11);
time = zeros(11);
x(:,1) = [1; 0; 1; 0];
time(1)=0;
for i=1:10
    m0 = f*x(:,i);
    m1 = f*(x(:,i)+h*m0/2);
    m2 = f*(x(:,i)+h*m1/2);
    m3 = f*(x(:,i)+h*m2);
    x(:,i+1) = x(:,i) + h*(m0+2*m1+2*m2+m3)/6;
    time(i+1)=time(i)+h;
end
figure(4);
for i=1:4
    plot(time, x(i,:))
    hold on;
end
hold off;
legend('x1', 'x2', 'x3', 'x4');
title('Response with Sampling Rate h=1');
ylabel('Magnitude');
xlabel('Time')

%Sampling rate increased further to check if the response goes
 unbounded
h = 5;
x = zeros(4,11);
time = zeros(11);
x(:,1) = [1; 0; 1; 0];
time(1)=0;
for i=1:10
    m0 = f*x(:,i);
    m1 = f*(x(:,i)+h*m0/2);
    m2 = f*(x(:,i)+h*m1/2);
    m3 = f*(x(:,i)+h*m2);
    x(:,i+1) = x(:,i) + h*(m0+2*m1+2*m2+m3)/6;
    time(i+1)=time(i)+h;
end
figure(5);
for i=1:4
    plot(time, x(i,:))
    hold on;
end
hold off;
legend('x1', 'x2', 'x3', 'x4');
title('Response with Sampling Rate h=5');
ylabel('Magnitude');
```

```
xlabel('Time')

% Part (c)
h = 0.1;
x = zeros(4,201);
time = zeros(201);
x(:,1) = [-5; 2; 10; -3];
time(1)=0;
for i=1:200
    m0 = f*x(:,i);
    m1 = f*(x(:,i)+h*m0/2);
    m2 = f*(x(:,i)+h*m1/2);
    m3 = f*(x(:,i)+h*m2);
    x(:,i+1) = x(:,i) + h*(m0+2*m1+2*m2+m3)/6;
    time(i+1)=time(i)+h;
end
figure(6);
for i=1:4
    plot(time, x(i,:))
    hold on;
end
hold off;
legend('x1', 'x2', 'x3', 'x4');
title('Response with Sampling Rate h=0.1 and modified Conditions');
ylabel('Magnitude');
xlabel('Time')
```




Response with Sampling Rate h=1
Magnitude
Time
x1
x2
x3
x4
10
8
6
4
2
0
-2
-4
0
1
2
3
4
5
6
7
8
9
10


Response with Sampling Rate h=5
×10^25
Magnitude
Time
x1
x2
x3
x4
12
10
8
6
4
2
0
-2
-4
-6
0
5
10
15
20
25
30
35
40
45
50



Published with MATLAB® R2020b