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

System Information

```
initial_conditions = [-98;20;-100;25]
```

```
initial_conditions = 4×1
   -98
    20
  -100
    25
```

```
tspan = 10
```

```
tspan = 10
```

Writing System

```
syms x1(t) x2(t) x3(t) x4(t) u1(t) u2(t) t 'real'
x = [x1;x2;x3;x4];
u = [u1;u2];
dx = [x2; u1; x4 ; u2/2]
```

```
dx(t) =
```

$$\begin{pmatrix} x_2(t) \\ u_1(t) \\ x_4(t) \\ \frac{u_2(t)}{2} \end{pmatrix}$$

Writing System Dynamics

```
A = [0 1 0 0; 0 0 0 0; 0 0 0 1; 0 0 0 0]
```

```
A = 4×4
     0     1     0     0
     0     0     0     0
     0     0     0     1
     0     0     0     0
```

```
B = [0 0; 1 0; 0 0; 0 0.5]
```

```
B = 4×2
         0         0
    1.0000         0
         0         0
         0    0.5000
```

```
C = [1 0 0 0; -1 0 1 0]
```

```
C = 2×4
```

```
     1     0     0     0
    -1     0     1     0
```

```
D = [0 0;0 0]
```

```
D = 2×2
     0     0
     0     0
```

Checking Controllability

```
CO = ctrb(A,B)
```

```
CO = 4×8
         0         0    1.0000         0         0         0         0         0
    1.0000         0         0         0         0         0         0         0
         0         0         0    0.5000         0         0         0         0
         0    0.5000         0         0         0         0         0         0
```

```
r_CO = rank(CO)
```

```
r_CO = 4
```

```
if(rank(A) == r_CO)
    disp("System is controllable")
else
    disp("System is not controllable")
end
```

```
System is not controllable
```

Designing LQR Controller with provided Q,R

```
Q = diag([2,5,2,5])
```

```
Q = 4×4
     2     0     0     0
     0     5     0     0
     0     0     2     0
     0     0     0     5
```

```
R = diag([1,1])
```

```
R = 2×2
     1     0
     0     1
```

```
[P,K,L] = icare(A,B,Q,R)
```

```
P = 4×4
    3.9569    1.4142   -0.0000   -0.0000
    1.4142    2.7979   -0.0000   -0.0000
   -0.0000   -0.0000    4.6167    2.8284
   -0.0000   -0.0000    2.8284    6.5290
K = 2×4
    1.4142    2.7979   -0.0000    0.0000
   -0.0000    0.0000    1.4142    3.2645
```

```
L = 4×1 complex
  -0.6622 + 0.0000i
  -0.8161 + 0.2026i
  -0.8161 - 0.2026i
  -2.1358 + 0.0000i
```

```
u1 = - K(1,:)*x;
u2 = - K(2,:)*x;
dx = [x2; u1; x4 ; u2/2]
```

```
dx(t) =
```

$$\begin{pmatrix} x_2(t) \\ \frac{1859259797328309\,x_3(t)}{5070602400912917605986812821504} - \frac{6300384224323929\,x_2(t)}{2251799813685248} - \frac{6349385536679153\,x_4(t)}{1622592768292133633915780102} \\ x_4(t) \\ \frac{8198733962548731\,x_1(t)}{4056481920730334084789450257203} - \frac{597279309511843\,x_2(t)}{2028240960365167042394725128601} - \frac{918870491619799}{5629499534213} \end{pmatrix}$$

Testing System Output

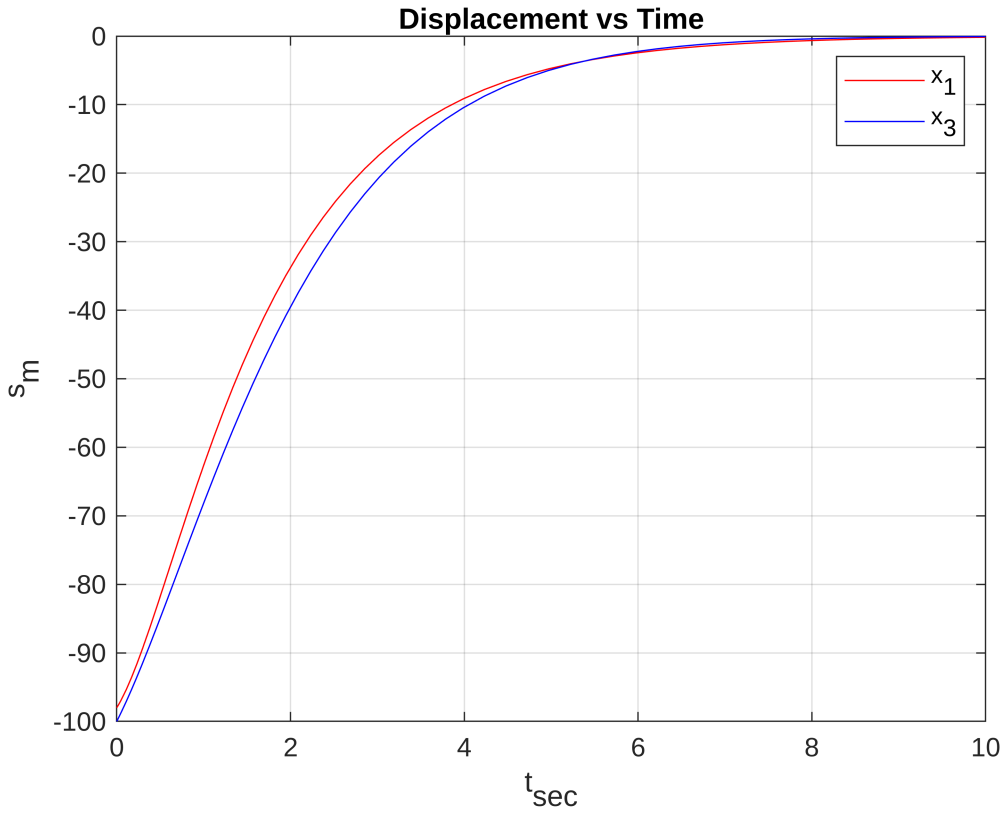
```
vars = [x1(t) x2(t) x3(t) x4(t)];
func_dx = odeFunction(dx,vars);
fdx = @(t,x)func_dx(t,x);
[t,x] = ode45(fdx,[0,tspan],initial_conditions);
for i=1:1:size(t)
    y(i,:) = (C*x(i,:)')';
    inputs(i,:) = (-K*x(i,:)')';
end
```

Plotting Graphs

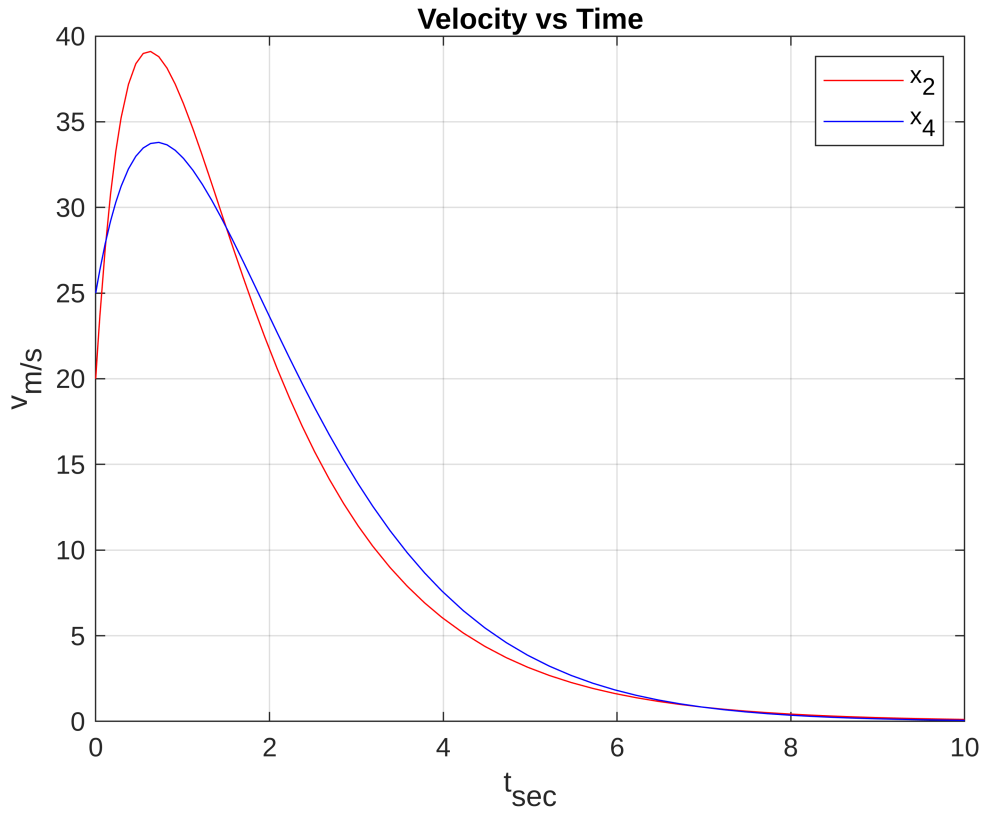
```
disp("Plotting States")
```

```
Plotting States
```

```
plot(t,x(:,1),'r')
hold on;
plot(t,x(:,3),'b')
hold on;
legend ('x_{1}','x_{3}');
grid on;
xlabel("t_{sec}")
ylabel("s_{m}")
title("Displacement vs Time")
hold off;
exportgraphics(gcf,'default_tuned_x1x3.png','Resolution',1200)
```



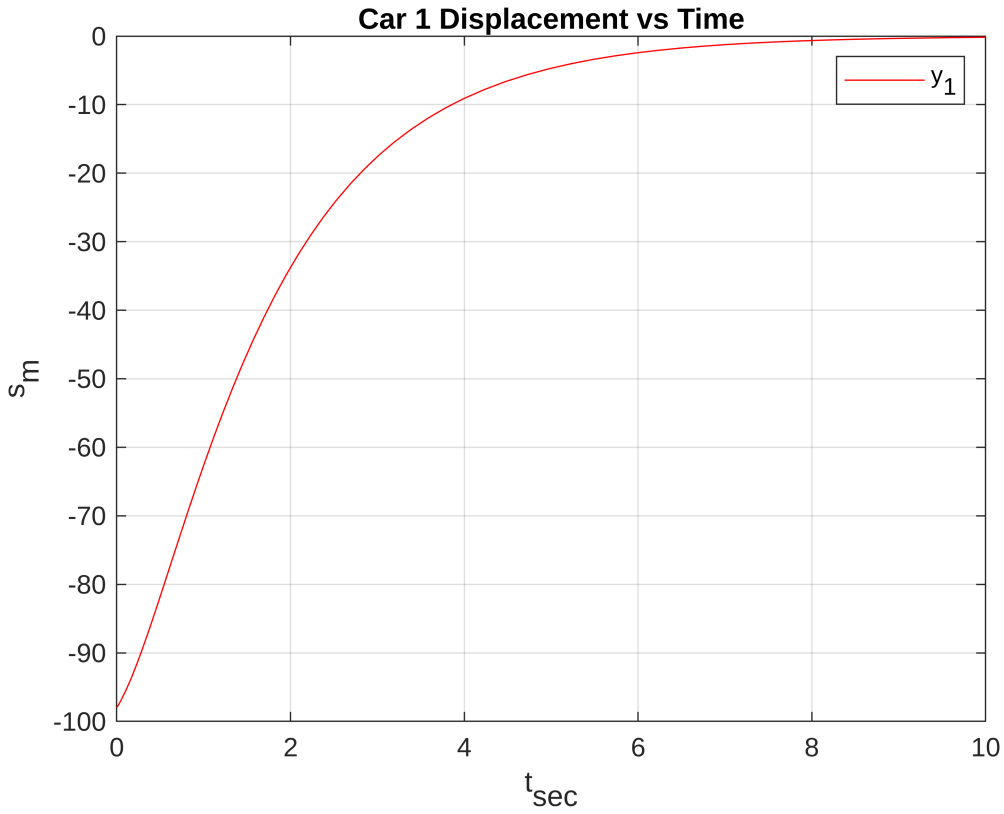

```
plot(t,x(:,2),'r')
hold on;
plot(t,x(:,4),'b')
hold on;
legend ('x_{2}','x_{4}');
grid on;
xlabel("t_{sec}")
ylabel("v_{m/s}")
title("Velocity vs Time")
hold off;
exportgraphics(gcf,'default_tuned_x2x4.png','Resolution',1200)
```



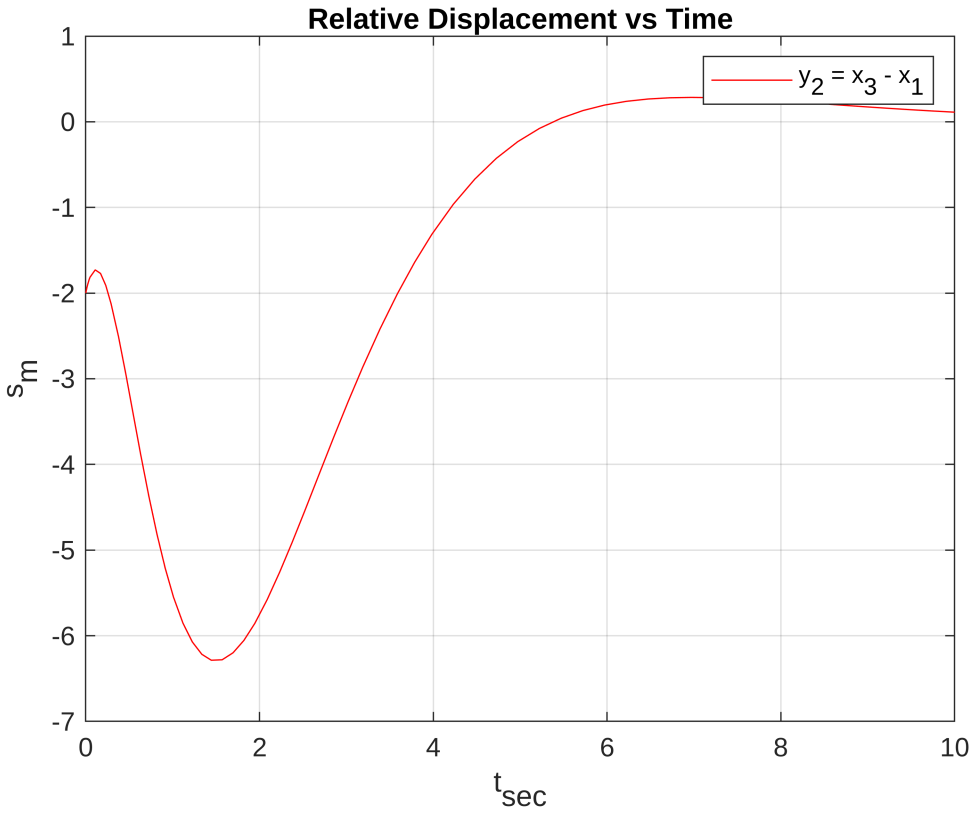

```
disp("Plotting Outputs")
```

```
Plotting Outputs
```

```
plot(t,y(:,1),'r')
legend ('y_{1}');
grid on;
xlabel("t_{sec}")
ylabel("s_{m}")
title("Car 1 Displacement vs Time")
hold off;
exportgraphics(gcf,'default_tuned_y1.png','Resolution',1200)
```



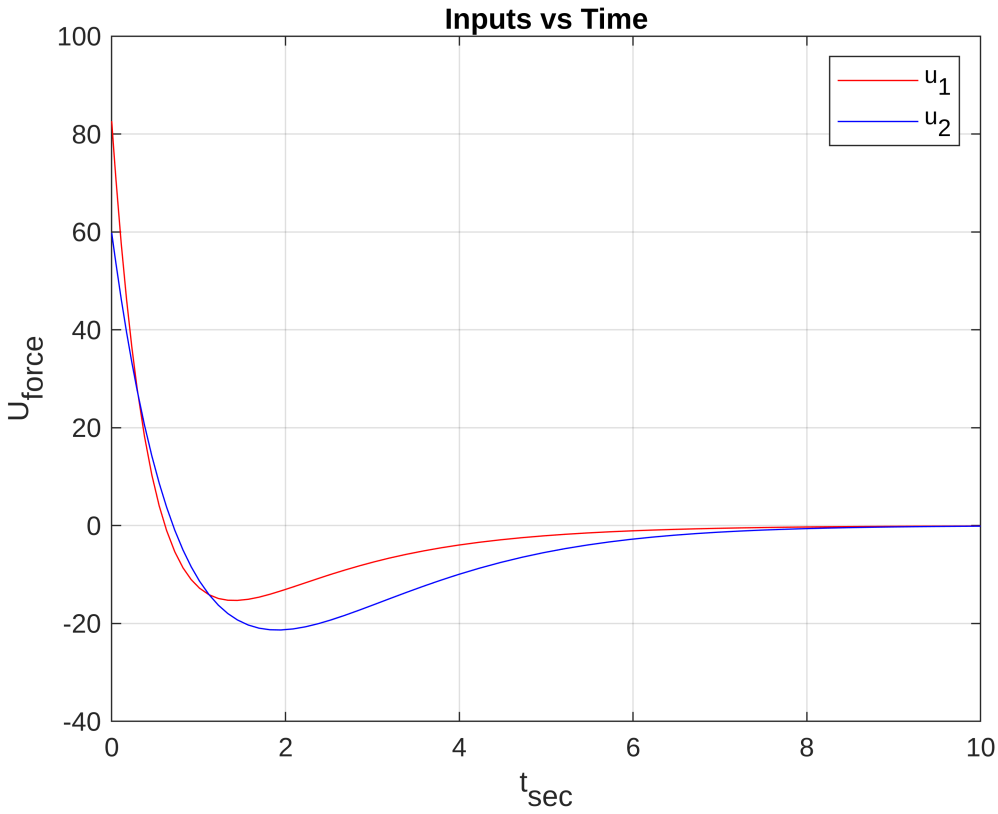

```
plot(t,y(:,2),'r')
legend ('y_{2} = x_{3} - x_{1}');
grid on;
xlabel("t_{sec}")
ylabel("s_{m}")
title("Relative Displacement vs Time")
hold off;
exportgraphics(gcf,'default_tuned_y2.png','Resolution',1200)
```




```
disp("Plotting Inputs")
```

```
Plotting Inputs
```

```
plot(t,inputs(:,1),'r')
hold on;
plot(t,inputs(:,2),'b')
hold on;
legend ('u_{1}','u_{2}');
grid on;
xlabel("t_{sec}")
ylabel("U_{force}")
title("Inputs vs Time")
hold off;
exportgraphics(gcf,'default_tuned_u1u2.png','Resolution',1200)
```



Safety Constraints Check

```
first_violation_time = tspan+1;
for i=1:1:size(t)
    if(y(i,2)>0)
        if(first_violation_time > tspan)
            first_violation_time = t(i)
            [maximum_violation,i] = max(y(:,2));
            maximum_violation
            max_violation_time = t(i)
            break;
        end
    end
end
```

```
first_violation_time = 5.4719
maximum_violation = 0.2846
max_violation_time = 6.9719
```

Tuning LQR for minimum distance Tuning

```
a = 0.0
```

```
a = 0
```

```
b = 1
```

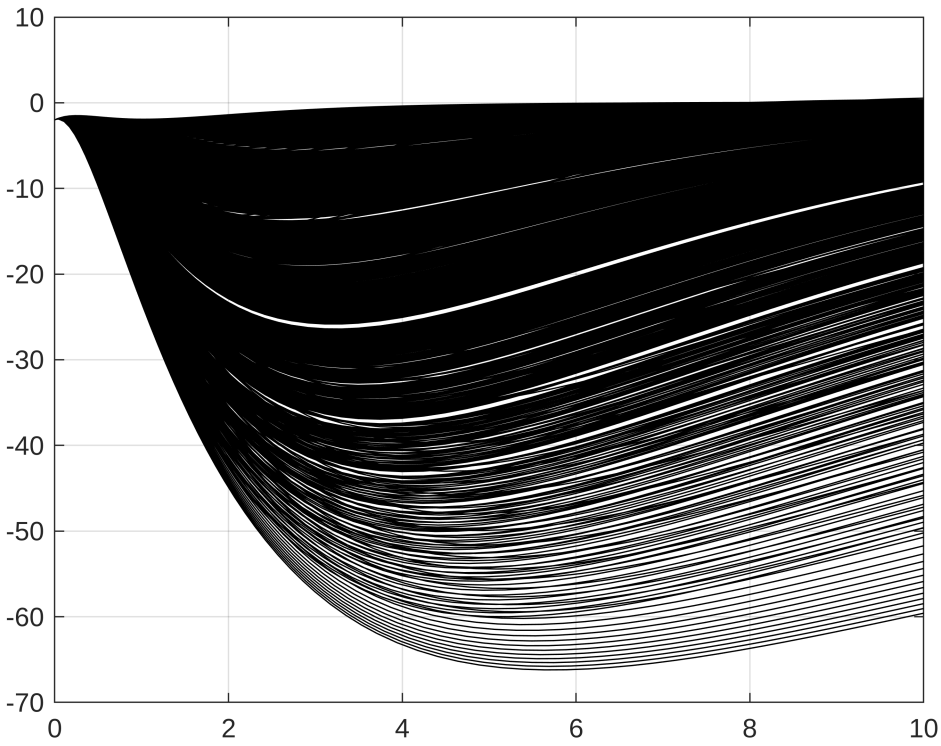
```
b = 1
```

```
first_entry = 1;
warning('off','all')
output = [0 0];
while(min(output(:,2)) < -2 || max(output(:,2))>0 || first_entry)
    first_entry=0;
    syms x1(t) x2(t) x3(t) x4(t) 'real';
    x = [x1;x2;x3;x4];
    a = a+0.05;
    Q = diag([2.5, 10, a, b-a]);
    [P,K,L] = icare(A,B,Q,R);
    u1 = - K(1,:)*x;
    u2 = - K(2,:)*x;
    dx = [x2; u1; x4 ; u2/2];
    vars = [x1(t) x2(t) x3(t) x4(t)];
    func_dx = odeFunction(dx,vars);
    fdx = @(t,x)func_dx(t,x);
    [t,x] = ode45(fdx,[0,tspan],initial_conditions);
    length = size(t);
    length = length(1,1);
    output = zeros(length,2);
    for i=1:1:size(t)
        output(i,:) = (C*x(i,:)')';
    end
    maximum = max(output(:,2));
    minimum = min(output(:,2));
    plot(t,output(:,2),'black')
    hold on;
    grid on;
    if(maximum > 0)
        disp('Failure at')
        b
        b = b+1;
        a = 0;
    end
end
```

```
Failure at
b = 1
Failure at
b = 2
Failure at
b = 3
Failure at
b = 4
Failure at
b = 5
Failure at
b = 6
Failure at
b = 7
Failure at
```

```
b = 8
Failure at
b = 9
Failure at
b = 10
Failure at
b = 11
Failure at
b = 12
Failure at
b = 13
Failure at
b = 14
Failure at
b = 15
Failure at
b = 16
Failure at
b = 17
Failure at
b = 18
Failure at
b = 19
Failure at
b = 20
Failure at
b = 21
Failure at
b = 22
Failure at
b = 23
Failure at
b = 24
Failure at
b = 25
Failure at
b = 26
Failure at
b = 27
```

```
exportgraphics(gcf,'tuning_performance.png','Resolution',1600)
hold off;
```




```
warning('on','all')
disp('Congratulation. System is tuned !!')
```

```
Congratulation. System is tuned !!
```

```
a
```

```
a = 5.5500
```

```
b
```

```
b = 28
```

Displaying Performance on new Tuning

```
syms x1(t) x2(t) x3(t) x4(t) 'real';
```

```
Warning: Can only make assumptions on variable names, not 'x1(t)'.
Warning: Can only make assumptions on variable names, not 'x2(t)'.
Warning: Can only make assumptions on variable names, not 'x3(t)'.
Warning: Can only make assumptions on variable names, not 'x4(t)'.
```

```
x = [x1;x2;x3;x4];
Q = diag([2.5, 10, a, b-a])
```

```
Q = 4×4
    2.5000         0         0         0
         0   10.0000         0         0
         0         0    5.5500         0
```

```
         0         0         0   22.4500
```

```
[P,K,L] = icare(A,B,Q,R)
```

```
P = 4×4
    5.7363    1.5811    0.0000    0.0000
    1.5811    3.6280    0.0000    0.0000
    0.0000    0.0000   13.3003    4.7117
    0.0000    0.0000    4.7117   11.2913
K = 2×4
    1.5811    3.6280    0.0000    0.0000
    0.0000    0.0000    2.3558    5.6457
L = 4×1
   -0.5065
   -0.5091
   -2.3137
   -3.1214
```

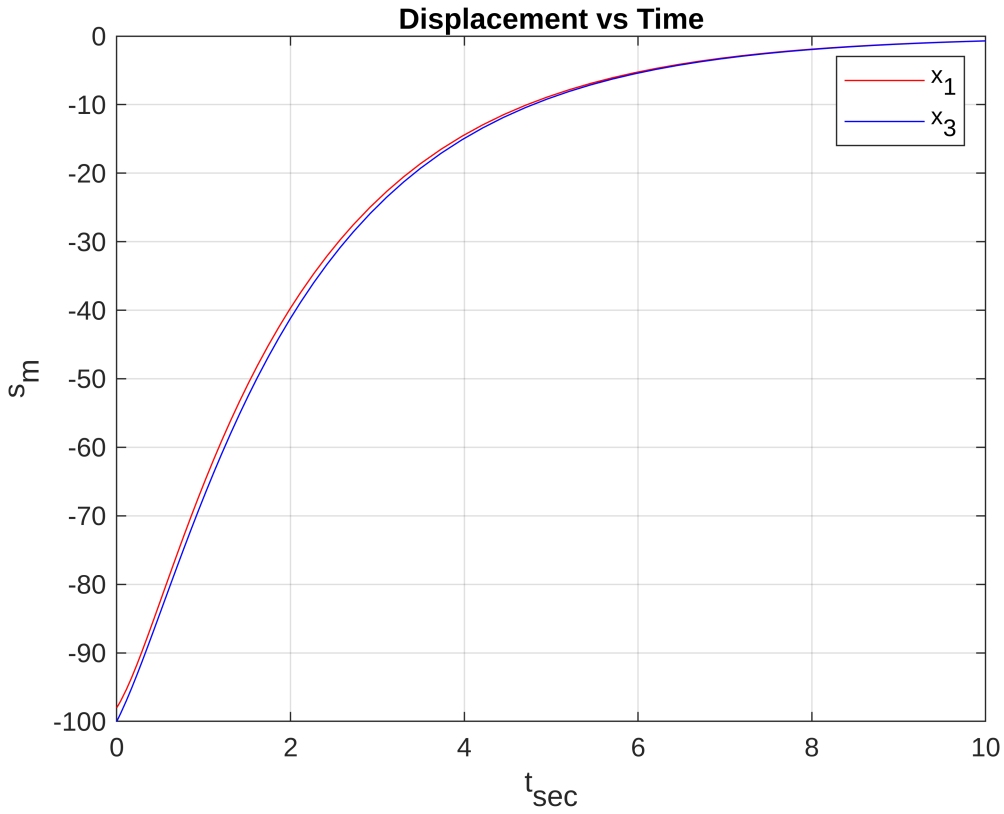
```
u1 = - K(1,:)*x;
u2 = - K(2,:)*x;
dx = [x2; u1; x4 ; u2/2];
vars = [x1(t) x2(t) x3(t) x4(t)];
func_dx = odeFunction(dx,vars);
fdx = @(t,x)func_dx(t,x);
[t,x] = ode45(fdx,[0,tspan],initial_conditions);
length = size(t);
length = length(1,1);
output = zeros(length,2);
for i=1:1:size(t)
    output(i,:) = (C*x(i,:)')';
    inputs(i,:) = (-K*x(i,:)')';
end
```

Plotting Graphs

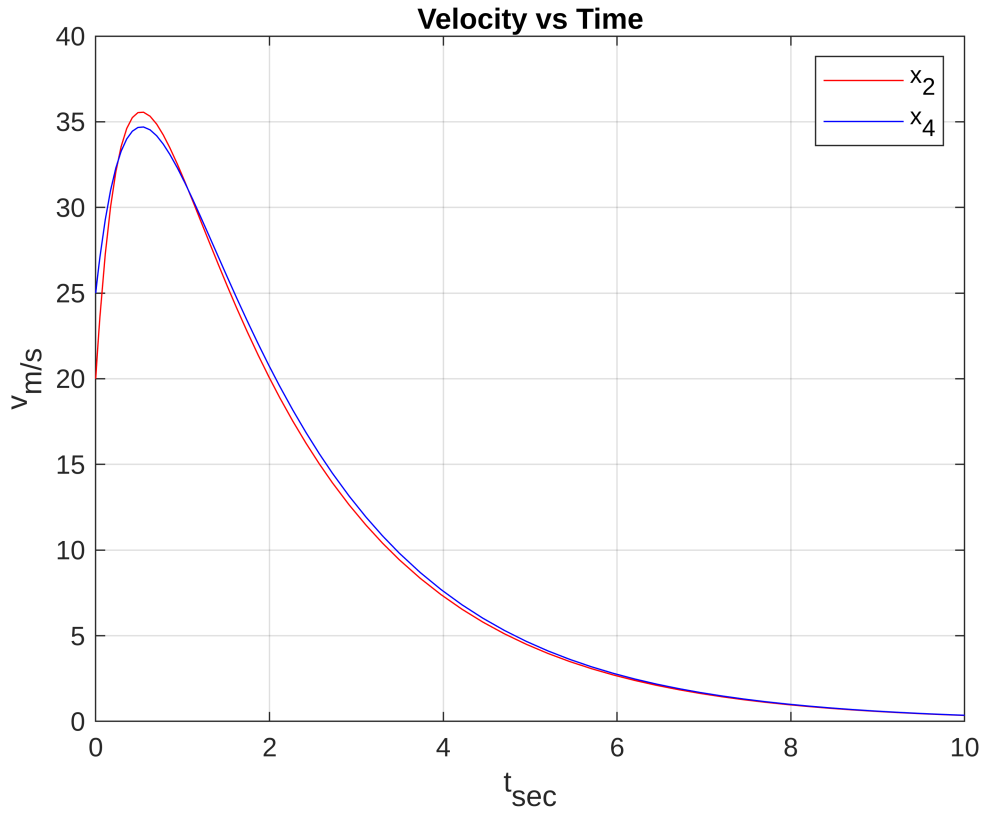
```
disp("Plotting States")
```

```
Plotting States
```

```
plot(t,x(:,1),'r')
hold on;
plot(t,x(:,3),'b')
hold on;
legend ('x_{1}','x_{3}');
grid on;
xlabel("t_{sec}")
ylabel("s_{m}")
title("Displacement vs Time")
hold off;
exportgraphics(gcf,'custom_tuned_x1x3.png','Resolution',1200)
```



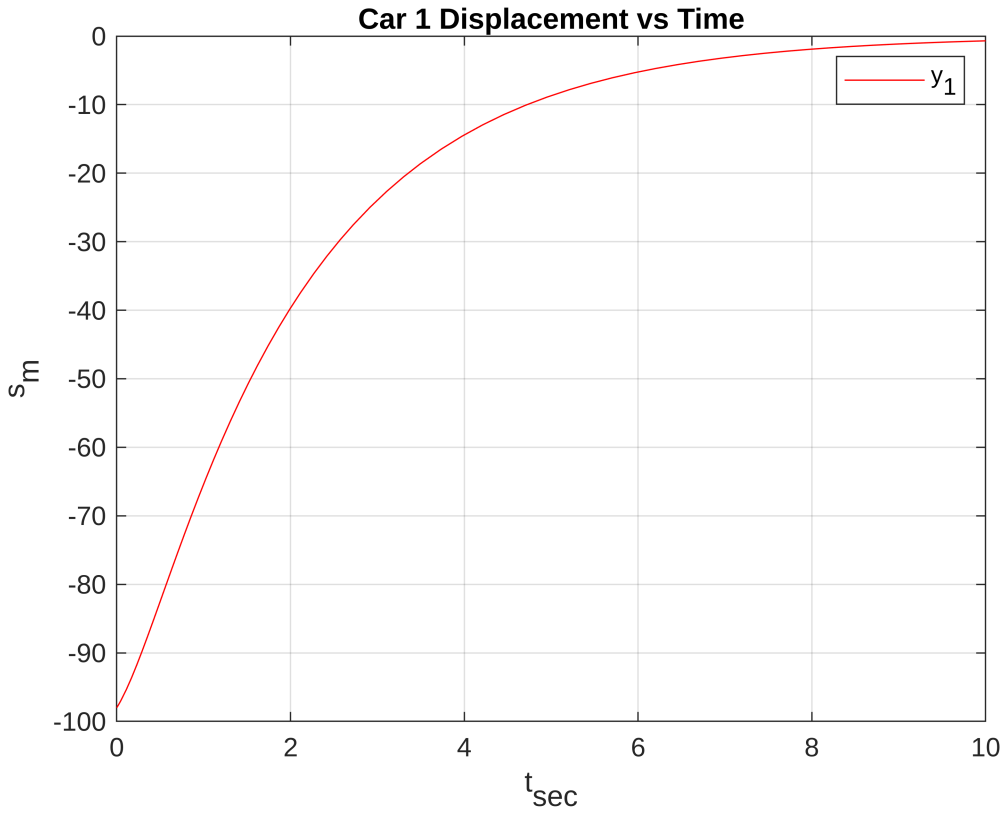

```
plot(t,x(:,2),'r')
hold on;
plot(t,x(:,4),'b')
hold on;
legend ('x_{2}','x_{4}');
grid on;
xlabel("t_{sec}")
ylabel("v_{m/s}")
title("Velocity vs Time")
hold off;
exportgraphics(gcf,'custom_tuned_x2x4.png','Resolution',1200)
```



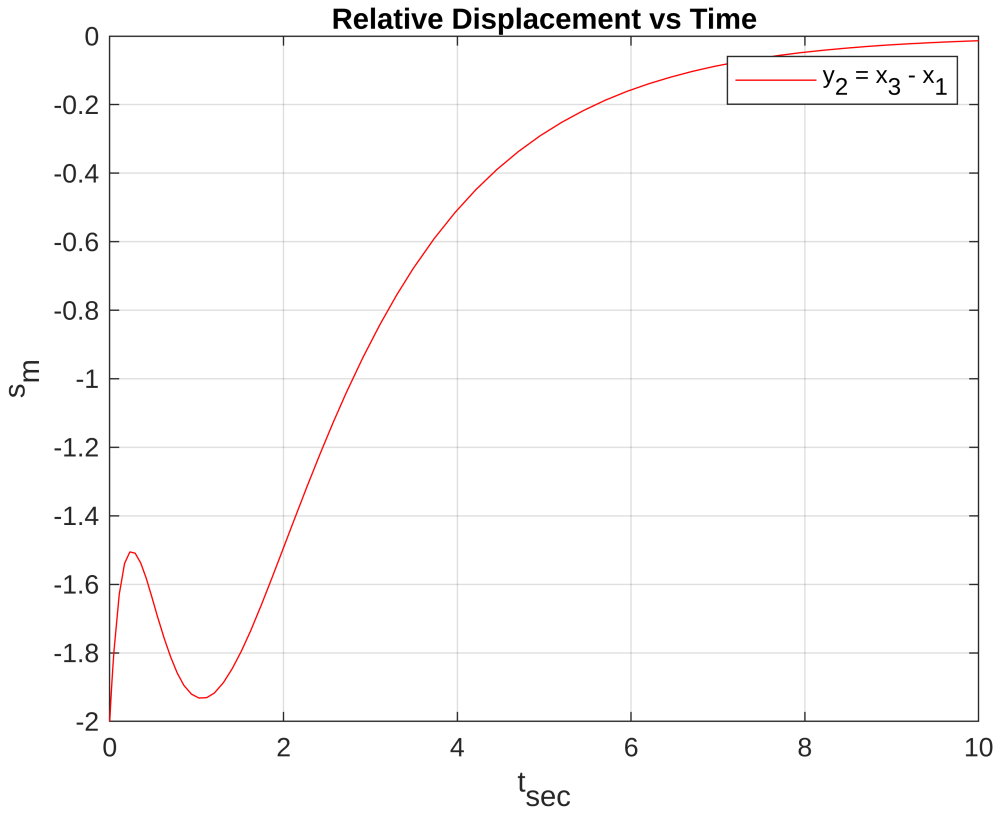

```
disp("Plotting Outputs")
```

```
Plotting Outputs
```

```
plot(t,output(:,1),'r')
legend ('y_{1}');
grid on;
xlabel("t_{sec}")
ylabel("s_{m}")
title("Car 1 Displacement vs Time")
hold off;
exportgraphics(gcf,'custom_tuned_y1.png','Resolution',1200)
```



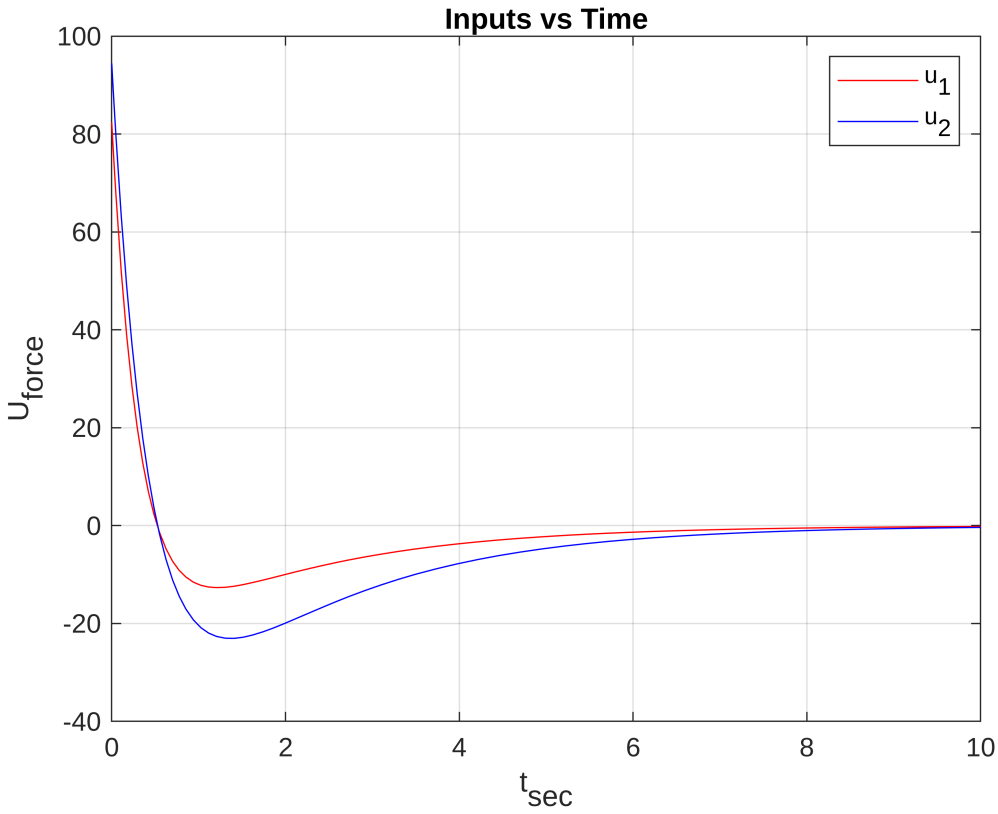

```
plot(t,output(:,2),'r')
legend ('y_{2} = x_{3} - x_{1}');
grid on;
xlabel("t_{sec}")
ylabel("s_{m}")
title("Relative Displacement vs Time")
hold off;
exportgraphics(gcf,'custom_tuned_y2.png','Resolution',1200)
```




```
disp("Plotting Inputs")
```

```
Plotting Inputs
```

```
plot(t,inputs(:,1),'r')
hold on;
plot(t,inputs(:,2),'b')
hold on;
legend ('u_{1}','u_{2}');
grid on;
xlabel("t_{sec}")
ylabel("U_{force}")
title("Inputs vs Time")
hold off;
```




```
exportgraphics(gcf,'custom_tuned_u1u2.png','Resolution',1200)
```