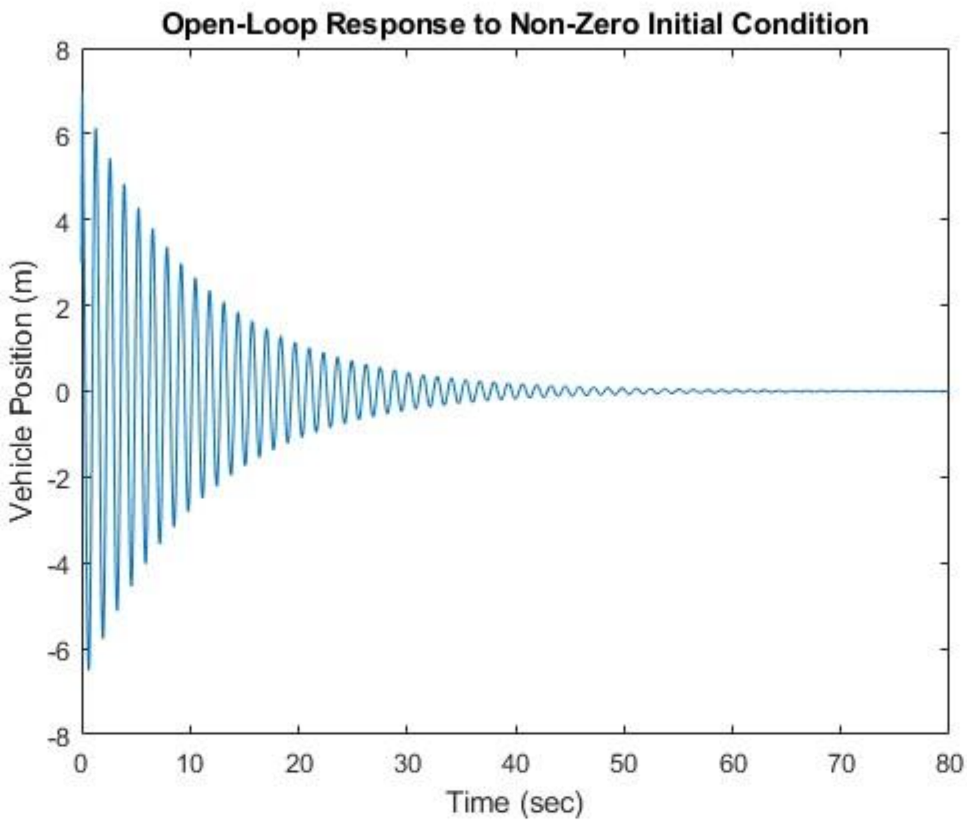
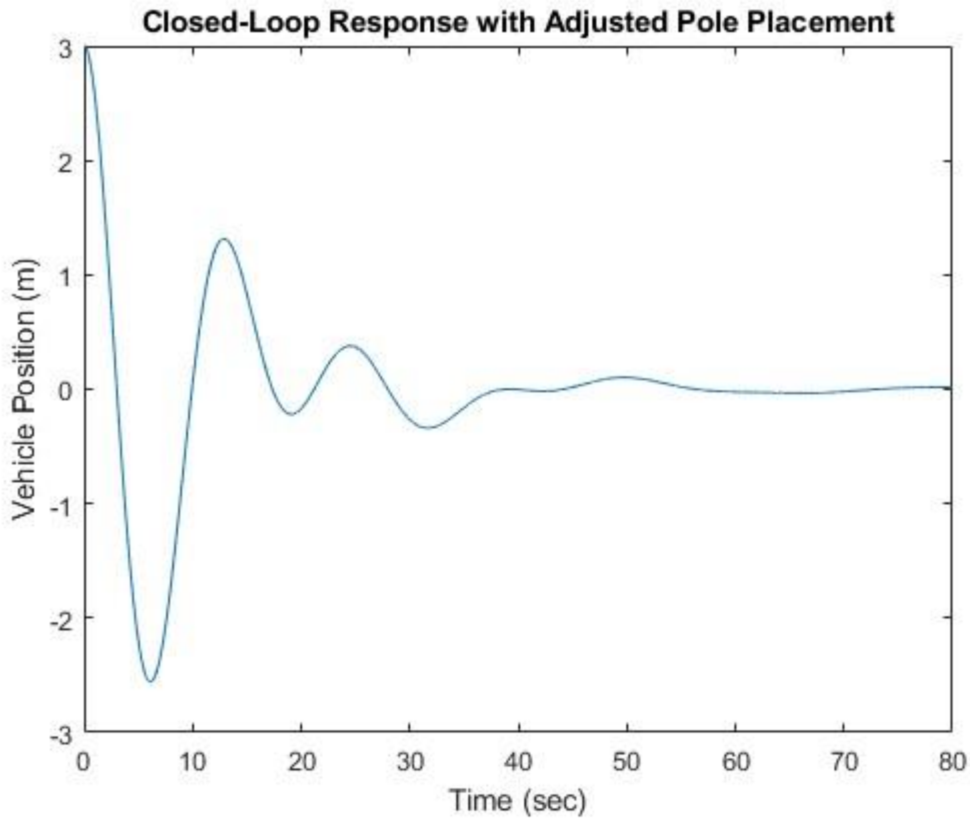
```
% Define system parameters
M1 = 3000;  % Mass 1
M2 = 350;   % Mass 2
K1 = 80000; % Stiffness 1 K2 =
500000;% Stiffness 2 b1 = 350;   %
Damping coefficient 1 b2 = 15020; %
Damping coefficient 2

% Define A matrix
A = [0, 0, 1, 0;
     0, 0, 0, 1;
     -K1/M1, K1/M1, -b1/M1, b1/M1;
     K1/M2, -(K1+K2)/M2, b1/M2, -(b1+b2)/M2];

% Define B matrix
B = [0;
     0;
     1/M1;
     K2/M2];

% Define C matrix
C = [1, -1, 0, 0];

% Define D matrix
D = 0;

% Calculate poles of the open-loop system
poles = eig(A); disp('Poles of the open-
loop system:'); disp(poles);

% Define time vector and initial conditions t = 0:0.01:80;  % Time
vector u = zeros(size(t)); % Input vector (zero input for initial
condition response)
x0 = [8; 5; 0; 0];  % Initial condition for states

% Define state-space system for open-loop system
sys_open = ss(A, B, C, D);

% Simulate the open-loop response to initial conditions
[y_open, t, x_open] = lsim(sys_open, u, t, x0);

% Plot the open-loop response figure; plot(t, y_open);
title('Open-Loop Response to Non-Zero Initial Condition');
xlabel('Time (sec)'); ylabel('Vehicle Position (m)');

% Calculate the step response characteristics of the open-loop system
info_open = stepinfo(y_open, t);
disp('Open-Loop Maximum Overshoot (%):');
disp(info_open.Overshoot); disp('Open-
Loop Settling Time (sec):');
disp(info_open.SettlingTime);
```

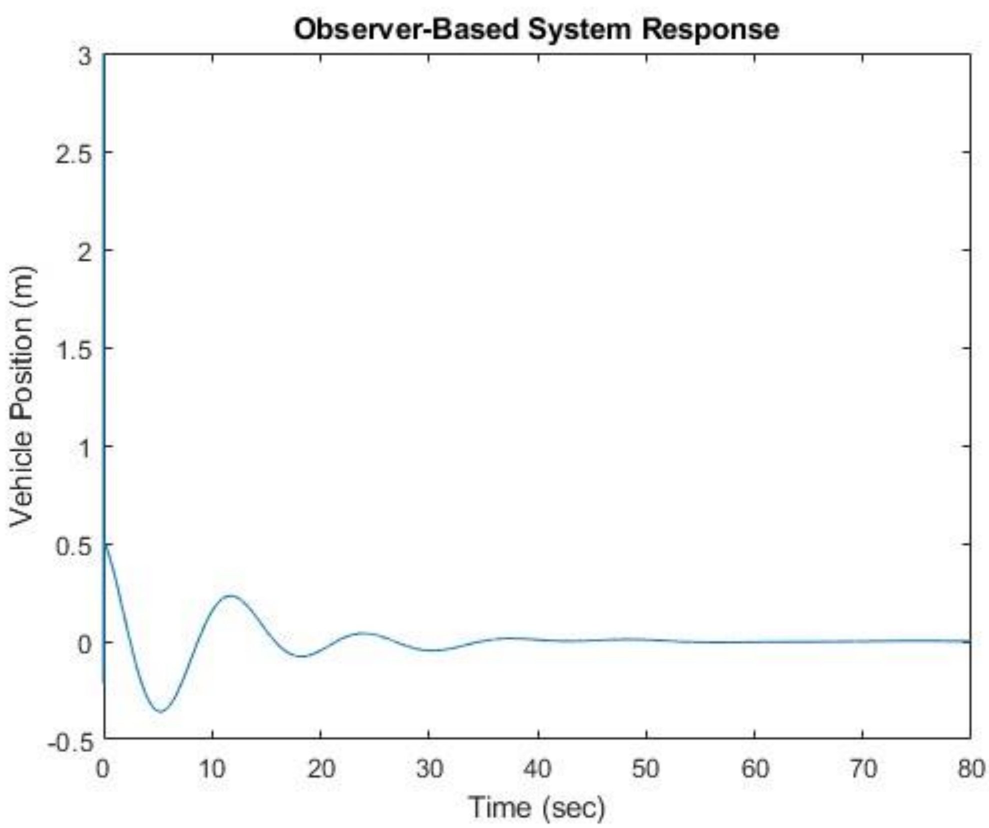
```
% Define desired poles for the closed-loop system with increased damping
desired_poles = [-0.1 + 0.5i, -0.1 - 0.5i, -0.05 + 0.2i, -0.05 - 0.2i];

% Calculate the state feedback gain matrix K for pole placement
K = place(A, B, desired_poles);

% Define the closed-loop system with new pole placement
Acl = A - B*K;
sys_closed = ss(Acl, B, C, D);

% Simulate the closed-loop response with initial conditions
[y_cl, t, x_cl] = lsim(sys_closed, u, t, x0);

% Plot the closed-loop response figure; plot(t, y_cl);
title('Closed-Loop Response with Adjusted Pole Placement');
xlabel('Time (sec)'); ylabel('Vehicle Position (m)');

% Calculate step response characteristics for the closed-loop system
info_cl = stepinfo(y_cl, t); disp('Closed-Loop Maximum Overshoot
(%):'); disp(info_cl.Overshoot); disp('Closed-Loop Settling Time
(sec):'); disp(info_cl.SettlingTime);

% Observer poles and observer design
observer_poles = [-300 + 50i, -300 - 50i, -400 + 30i, -400 - 30i];
L = place(A', C', observer_poles)';

% Define the augmented state-space matrices for the system with observer
At = [ A-B*K, B*K;
       zeros(size(A)), A-L*C ];
Bt = [B; zeros(size(B))];
Ct = [C, zeros(size(C))];

% Define the system with observer
sys_observer = ss(At, Bt, Ct, 0);

% Simulate the observer-based system with initial conditions (adjust initial
states) x0_aug = [x0; x0]; % The augmented initial condition, concatenating
the system and observer initial conditions [y_observer, t, x_observer] =
lsim(sys_observer, zeros(size(t)), t, x0_aug);

% Plot the observer response figure; plot(t, y_observer(:, 1)); %
Plot the first state or output (vehicle position) title('Observer-
Based System Response'); xlabel('Time (sec)'); ylabel('Vehicle
Position (m)');
```

```
Poles of the open-loop system:
 -21.9240 +34.3096i
 -21.9240 -34.3096i
  -0.0915 + 4.7928i
  -0.0915 - 4.7928i

Open-Loop Maximum Overshoot (%):
   1.5291e+05
```

```
Open-Loop Settling Time (sec):
   79.9308

Closed-Loop Maximum Overshoot (%):
   1.8319e+04

Closed-Loop Settling Time (sec):
   79.1962
```




Closed-Loop Response with Adjusted Pole Placement
Vehicle Position (m)
Time (sec)
3
2
1
0
-1
-2
-3
0
10
20
30
40
50
60
70
80


Observer-Based System Response
Vehicle Position (m)
Time (sec)
3
2.5
2
1.5
1
0.5
0
-0.5
0
10
20
30
40
50
60
70
80

Published with MATLAB® R2024a

2) Matlab-code with other desired controller and observer poles and linked slx file.

```
M1 = 3000;
M2 = 350;
K1 = 80000;
K2 = 500000;
b1 = 350;
b2 = 15020;

s = tf('s');
G1 = ((M1+M2)*s^2+b2*s+K2)/((M1*s^2+b1*s+K1)*(M2*s^2+(b1+b2)*s+(K1+K2))-
(b1*s+K1)*(b1*s+K1));

%step(G1)

G2 = (-M1*b2*s^3-M1*K2*s^2)/((M1*s^2+b1*s+K1)*(M2*s^2+(b1+b2)*s+(K1+K2))-
(b1*s+K1)*(b1*s+K1));

%step(0.1*G2)


[a ,b ,c, d]=linmod('modern2')

%controllability check
f=ctrb(a,b);
if rank(f)==rank(a)
   disp('All states are controllable');
else
   disp('All states are not controllable');
   uncontrollable_states = rank(a) - rank(f);
   fprintf('Number of uncontrollable states: %d\n', uncontrollable_states);
end
%desired_controller_poles
desired_c_poles = [-24.05+35.5j,-24.05-35.5j,-0.61+4.9283j,-0.61-4.9283j];
K = place(a, b, desired_c_poles)

%observability check
g=obsv(a,c);
if rank(g)==rank(a)
```

```
    disp('All states are observable');
else
    disp('All states are not observable');
    unobservable_states = rank(a) - rank(g);
    fprintf('Number of unobservable states: %d\n', unobservable_states);
end

%desired_observer_pole
desired_o_poles = [-30,-33,-38,-40];
L = place(a', c', desired_o_poles)'

%closed-loop_controller_analysis
e=a-b*K
charpoly(e)

%observer_analysis
f=a-L*c
charpoly(f)
```