In this problem, you will work with a ball suspended magnetically in air.

Table of Contents

You will go through the full process starting from taking the model of the system, determining its stability, solving the pole placement problem, using the Input Gain to satisfy steady-state tracking specification, designing the observer and simulating the system with the observed state feedback.

1. Modeling

The model of this system is given below. The current through the coils induces a magnetic force which can balance the force of gravity and cause the ball (which is made of a magnetic material) to be suspended in midair.

The equations for the system are given by:

$$M\frac{d^2h}{dt^2} = Mg - \frac{Ki^2}{h}$$

$$V = L\frac{di}{dt} + iR$$

where `h` is the vertical position of the ball, `i` is the current through the electromagnet, `V` is the applied voltage, `M` is the mass of the ball, `g` is gravity, `L` is the inductance, `R` is the resistance, and `K` is a coefficient that determines the magnetic force exerted on the ball. For simplicity, we will choose values `M = 0.05 Kg`, `K = 0.0001`, `L = 0.01 H`, `R = 1 Ohm`, `g = 9.81 m/sec^2`. The system is at equilibrium (the ball is suspended in midair) whenever `h = K i^2/Mg` (at which point `dh/dt = 0`). We linearize the equations about the point `h = 0.01 m` (where the nominal current is about 7 amp) and get the state space equations. The state variables are deviations in h, derivative of h and current i (a 3x1 vector), `u` is the input voltage (delta `V`), and `y` (the output), is the deviation in h. The system matrices for the linearized system are given below.

```
close all

A = [ 0   1   0
     980  0  -2.8
      0   0  -100 ];
```

```
B = [ 0
      0
      100 ];

C = [ 1 0 0 ];
```

2. Stability

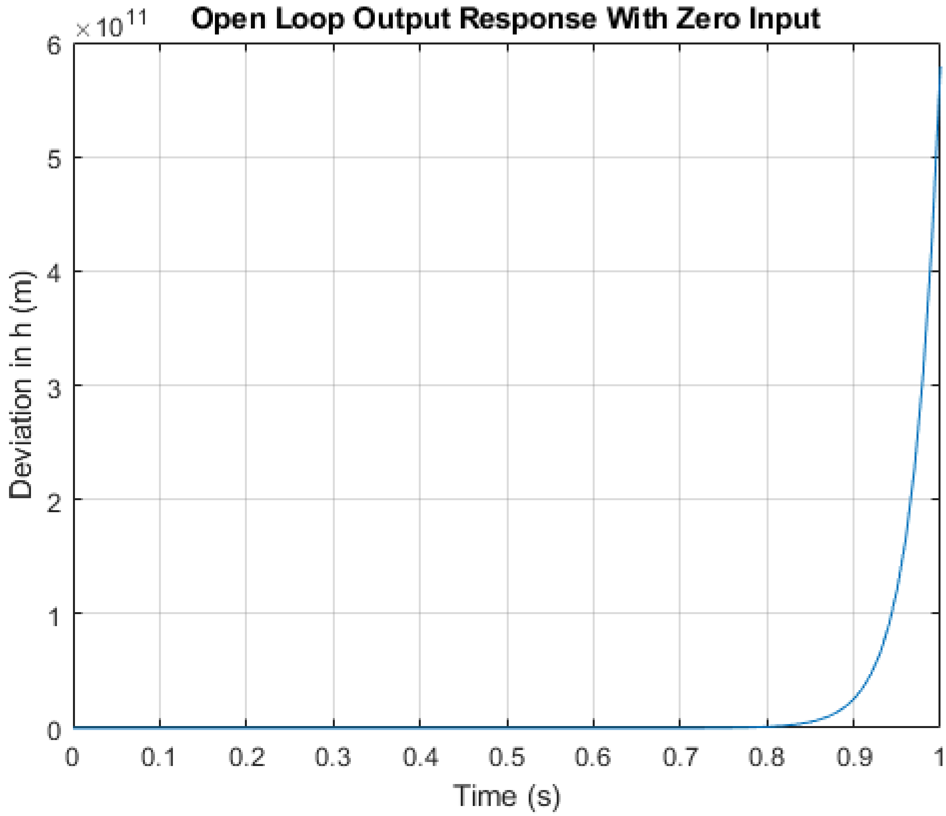
One of the first things we want to do is analyze whether the open-loop system (without any control) is stable. Determine the stability of the system.

```
eigen = eig(A)
if all(eigen < 0)
    disp('Stable');
else
    disp('Not Stable');
end
```

```
eigen =

   31.3050
  -31.3050
 -100.0000

Not Stable
```

Check your stability conclusion by simulating the system response to a nonzero initial condition (zero input). Take this state as your initial condition: x0 = [0.01 0.5 -5] (and use it for the remainder of this question as needed). Plot the output response to this initial condition and comment on your results.

```
D = 0;
K_zero = zeros(1,length(A));
x0 = [0.01 0.5 -5];
tspan = [0:0.01:1];
R_zero = zeros(1,length(A));

[t,x] = ode45(@(t,x) sys(t,x,A,B,K_zero),tspan,x0);

figure(1);
plot(t,x(:,1));
title('Open Loop Output Response With Zero Input')
xlabel('Time (s)');
ylabel('Deviation in h (m)');
grid on
```



3. Controllability and Observability

Check controllability and observability of your system. Comment on your results.

```
P = ctrb(A,B);
if rank(P) == length(A)
    disp('System is controllable');
else
    disp('System is uncontrollable');
end

Q = obsv(A,C);
if rank(Q) == length(A)
    disp('System is observable');
else
    disp('System is observable');
end
```

```
System is controllable
System is observable
```

4. Control Design Using Pole Placement

Next, build a state feedback controller for this system using pole placement. Determine the gain matrix K to place the poles at the following locations: -20 + 20i, -20 - 20i, -100

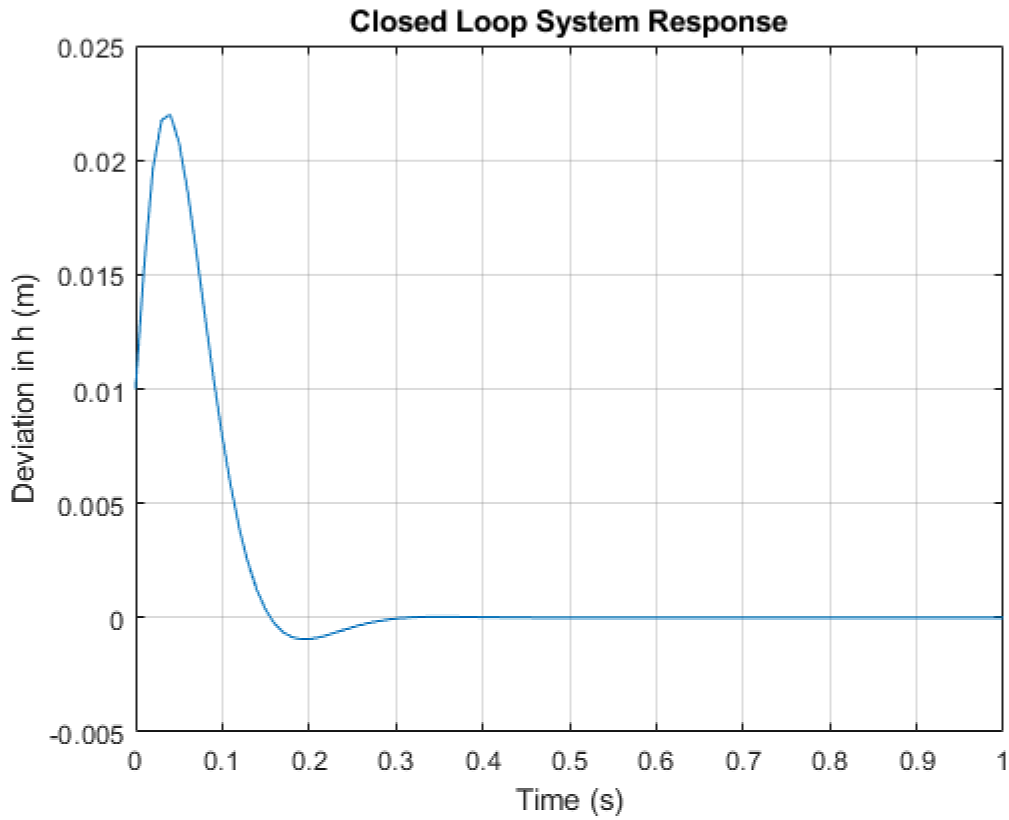
```
poles = [-20+20j -20-20j -100];
K = place(A,B,poles)
```

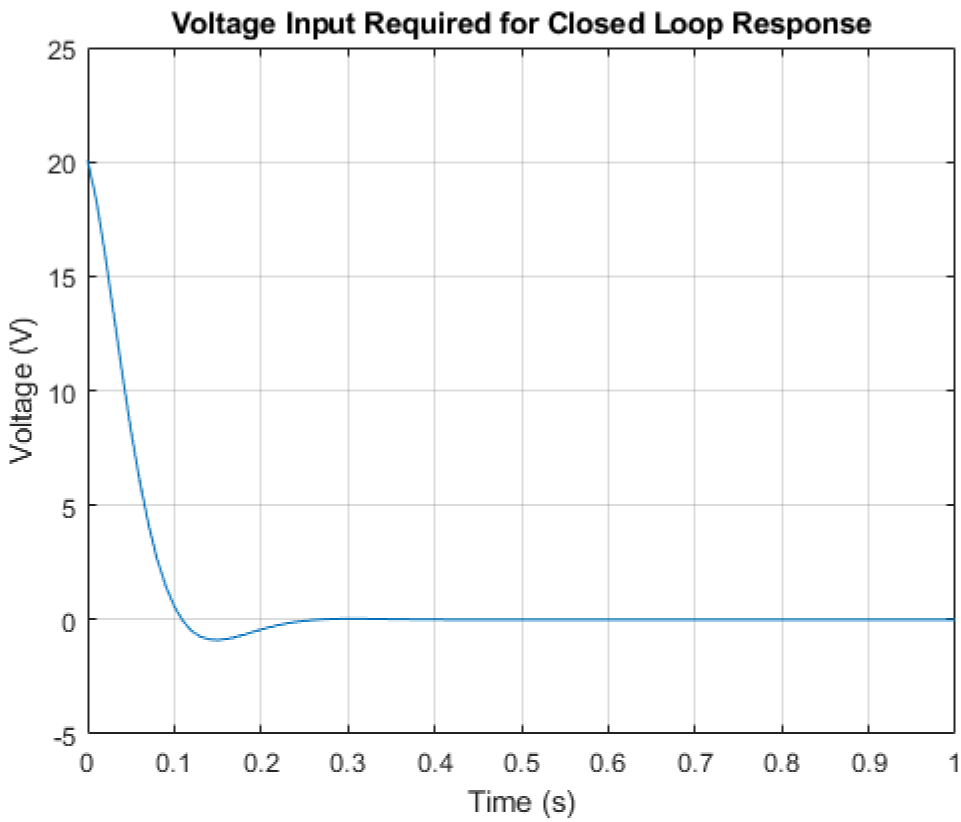
```
K =

 -775.7143  -20.6429    0.4000
```

Simulate the response of the system to an initial condition x0 = [0.01 0.5 -5] to demonstrate closed-loop transient performance. Plot the output response. Use the appropriate time scale. What is is the setting time of your system (approximately)? Plot the control effort (voltage input) required.

```
[t,x_closed] = ode45(@(t,x) sys(t,x,A,B,K),tspan,x0);

figure(2)
plot(t,x_closed(:,1));
title('Closed Loop System Response');
xlabel('Time (s)');
ylabel('Deviation in h (m)');
grid on;

disp('Setting time is approximately 0.3 seconds');

for j = 1:length(x_closed)        %Solve u for all values of x
   u(j,1) = -K*x_closed(j,:)';    %Transpose x as ode45 output is
 vertical
end

figure(3)
plot(t,u);
title('Voltage Input Required for Closed Loop Response');
xlabel('Time (s)');
ylabel('Voltage (V)');
grid on
```

```
Setting time is approximately 0.3 seconds
```


Closed Loop System Response
0.025
0.02
0.015
0.01
0.005
0
-0.005
Deviation in h (m)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time (s)

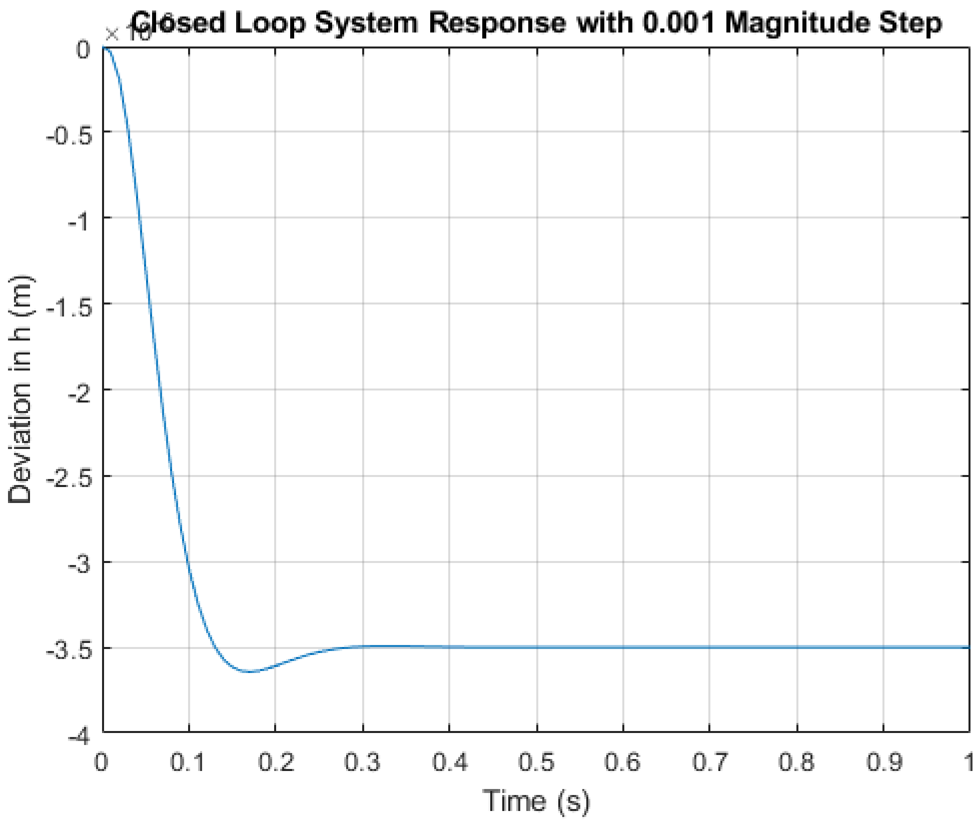

Voltage Input Required for Closed Loop Response
25
20
15
10
5
0
-5
Voltage (V)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Time (s)

5. Introducing the Reference Input

Now, you will take the control system as defined above and apply a step input (we choose a small value for the step, so we remain in the region where our linearization is valid). Use a step input of magnitude 0.001 and simulate the response of the system to this input. What do you observe?

```
R = .001;
iniCon = [0, 0, 0];
[t,x_ref] = ode45(@(t,x) sys2(t,x,A,B,K,1,R),tspan,iniCon);

figure(4)
plot(t,x_ref(:,1));
grid on
ylabel('Deviation in h (m)');
xlabel('Time (s)');
title('Closed Loop System Response with 0.001 Magnitude Step');

disp('We see that the response time (0.3s) is similar to the initial
 condition test done previously')
```

We see that the response time (0.3s) is similar to the initial condition test done previously

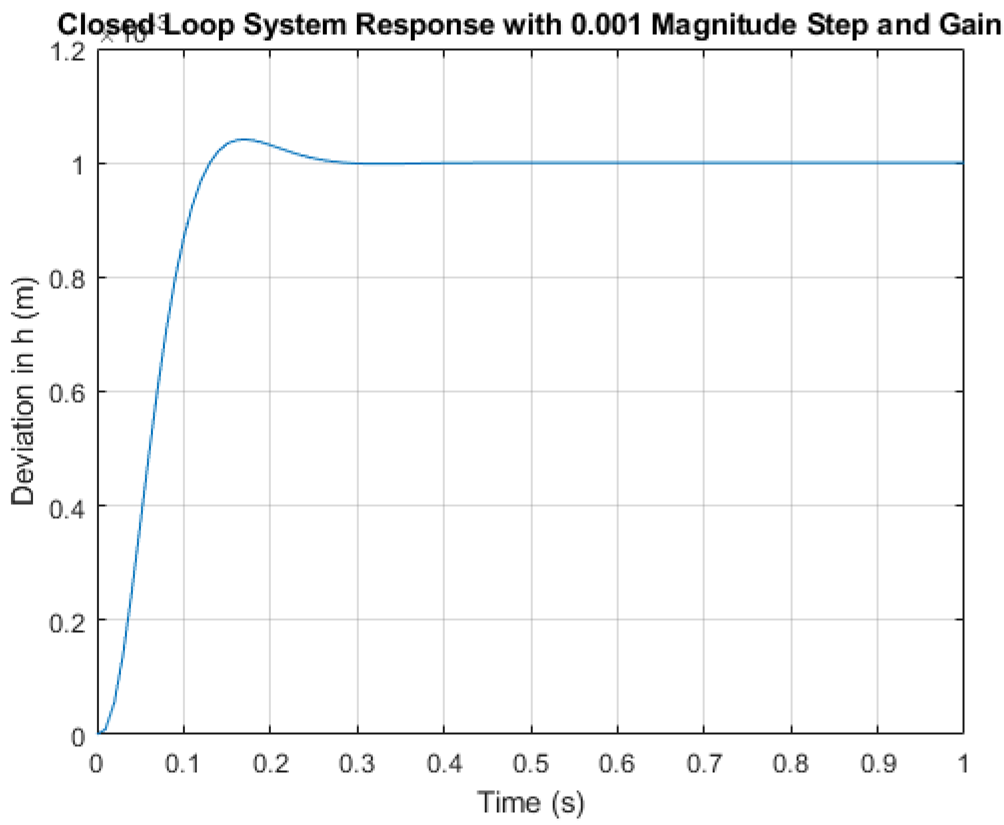


Use Input Gain method to determine the gain to achieve tracking of the step input. Simulate the system with the same input as before and comment on your results.

```
G = -1 * inv(C*inv(A-B*K)*B)
```

```
[t,x_ref] = ode45(@(t,x) sys2(t,x,A,B,K,G,R),tspan,iniCon);

figure(5)
plot(t,x_ref(:,1));
grid on
ylabel('Deviation in h (m)');
xlabel('Time (s)');
title('Closed Loop System Response with 0.001 Magnitude Step and
 Gain');

disp('Due to negative gain, height is increased rather than decreased,
 and a similar settling time is seen. However, the overshoot is much
 smaller.')
```

```
G =

 -285.7143

Due to negative gain, height is increased rather than decreased,
 and a similar settling time is seen. However, the overshoot is much
 smaller.
```



6. Observer Design

Next, we will build an **observer** to estimate the states, while measuring only the output `y = C x`. First, we need to choose the observer gain L. Since we want the dynamics of the observer to be much faster than the system itself, we need to place the poles at least five times farther to the left than the dominant poles of the system. Place the observer poles at -100, -101, -102.

```
p_obs = [-100 -101 -102];

L = place(A',C',p_obs)
```

```
L =

   1.0e+04 *

    0.0203
    1.1282
         0
```

Define the combined equations for the system plus observer using the original state `x` plus the error state: `e = x - \hat{x}`.

```
A_hat = A - L*C;
```

7. Now, we will simulate the response of the closed-loop combined system

to a nonzero initial condition with no reference input. We typically assume that the observer begins with zero initial condition, \hat{x} = 0. Use the following initial condition for the state: x0 = [0.01 0.5 -5]

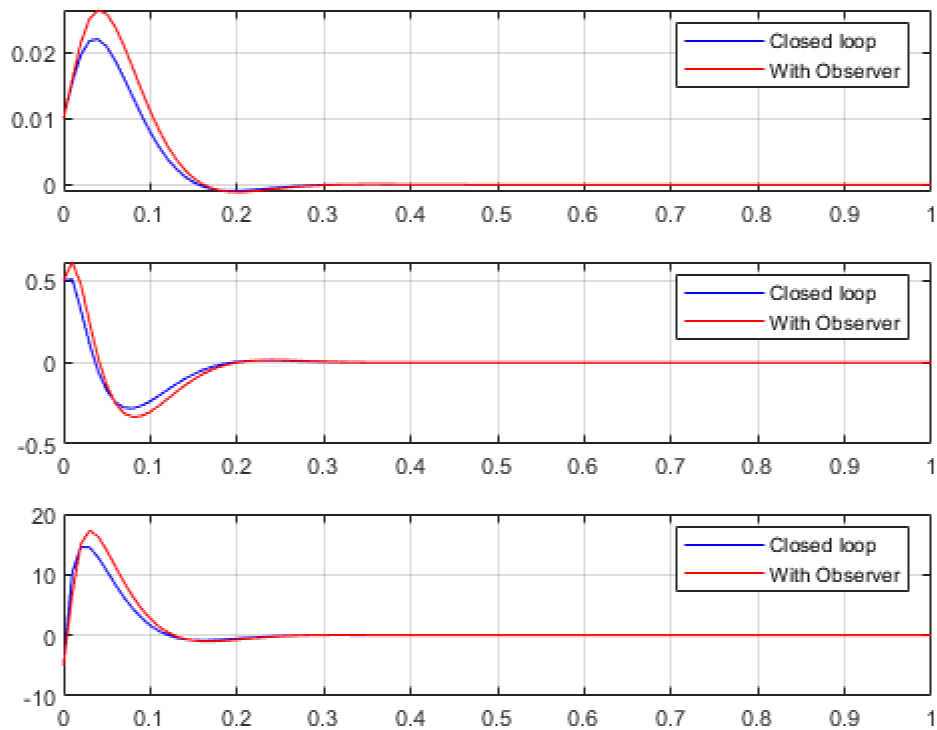
```
Xr0 = [0.01 0.5 -5 0.01 0.5 -5];
Ar = [(A-B*K) B*K; zeros(size(A)) (A-L*C)];
Br =[B; zeros(size(B))];
Cr = [C zeros(size(C))];
Dr = D;
JbkRr = ss(Ar,Br,Cr,Dr);

r = [zeros(size(t))];

[Yr,t,Xr] = lsim(JbkRr,r,t,Xr0);
```

We would like to verify the performance of the observer. Plot each state with the corresponding estimated state and comment on these results.

```
figure(6)
subplot(3,1,1)
plot(t,x_closed(:,1),'b',t,Xr(:,1),'r')
legend('Closed loop', 'With Observer')
grid on
```

```
subplot(3,1,2)
plot(t,x_closed(:,2),'b',t,Xr(:,2),'r')
legend('Closed loop', 'With Observer')
grid on

subplot(3,1,3)
plot(t,x_closed(:,3),'b',t,Xr(:,3),'r')
legend('Closed loop', 'With Observer')
grid on

disp('From the graphs shown, the observer closely follows the Closed-loop with a small overshoot but similar settling times.')
```

From the graphs shown, the observer closely follows the Closed-loop with a small overshoot but similar settling times.



ODE45 function

```
function dx = sys(t,x,A,B,K)
    dx = zeros(length(A),1);
    u = -K*x;
    dx = A*x + B*u;
end

%Second ode45 function setup to use gain and reference inputs
function dx2 = sys2(t,x,A,B,K,G,r)
```

```
    dx2 = zeros(length(A),1);
    u = -K*x + G*r;
    dx2 = A*x + B*u;
end
```

Published with MATLAB® R2020a