Bangladesh University of Engineering and Technology

Department of Electrical & Electronic Engineering

Course No.: EEE 312
Course Title: Digital Signal Processing I Laboratory

Experiment No.: 02

Name of the Experiment:

Time Domain Analysis of Discrete Time Signals and Systems

Assignment 2-b: Convolution and LCCDE

Date of Performance: December 07, 2022

Date of Submission: December 15, 2022

Prepared by:

Tasnimun Razin
Student ID: 1906044

Partner: Tasmin Khan
Student ID: 1906055

Section: A2
Level:3, Term:1

Problem 1

To solve the problem, we will first declare some custom user defined functions for our convenience. The functions will help to do Signal folding, multiplication, shifting and convolution. These are very common operations used in discrete signal processing. This way, we can avoid repetitive coding and just pass the input to get desired output.

MATLAB Code for Function declaration:

Function for Signal Folding:

```
function [y,n] = fold(x,n)
    y = fliplr(x);
    n = -fliplr(n);
```

Function for Signal Shifting:

```
function [y,n] = shift(x,o,n0)
    n = o + n0;
    y = x;
```

Function for Multiplication of Two Signals:

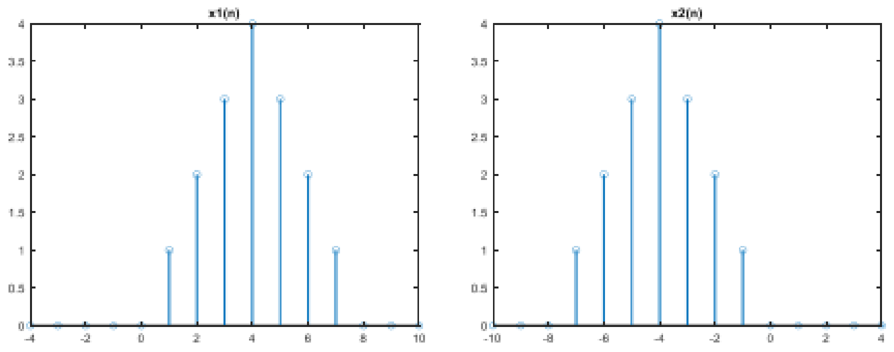
```
function [y,n]=product_1(x1,n1,x2,n2)
    n = min(n1(1),n2(1)) : max(n1(end),n2(end));

    x1_i = zeros(1,length(n));
    x2_i = zeros(1,length(n));

    x1_i(find(n>=n1(1) & n<=n1(end))) = x1;
    x2_i(find(n>=n2(1) & n<=n2(end))) = x2;

    y = x1_i.*x2_i;
```

Function for Convolution of Two Signals:

```
function [y,n] = cnvl(x1,n1,x2,n2)
    k_min = n1(1) + n2(1);
    k_max = n1(end) + n2(end);
    n = k_min:k_max;
    y = conv(x1,x2);
```

Question:

1. Two signals are provided: $x_1(n)$ and $x_2(n)$. They are shifted triangular signals with a peak value of P. $x_1(n)$ starts from 0 and $x_2(n)$ ends at 0. The figure below illustrates the two signals graphically with $P = 4$. The x-axis (sample axis) can be set to any arbitrarily large range as long as the entire signal is included.



Now, for $P = (5 + \boldsymbol{Last\ Digit\ of\ your\ ID})$, perform the following operations:

a. $x_1(n) * x_2(n)$.

b. $x_1(P-n) * x_2(P-n)$.

c. $x_1(n) * [x_2(n) * x_2(-n)]$.

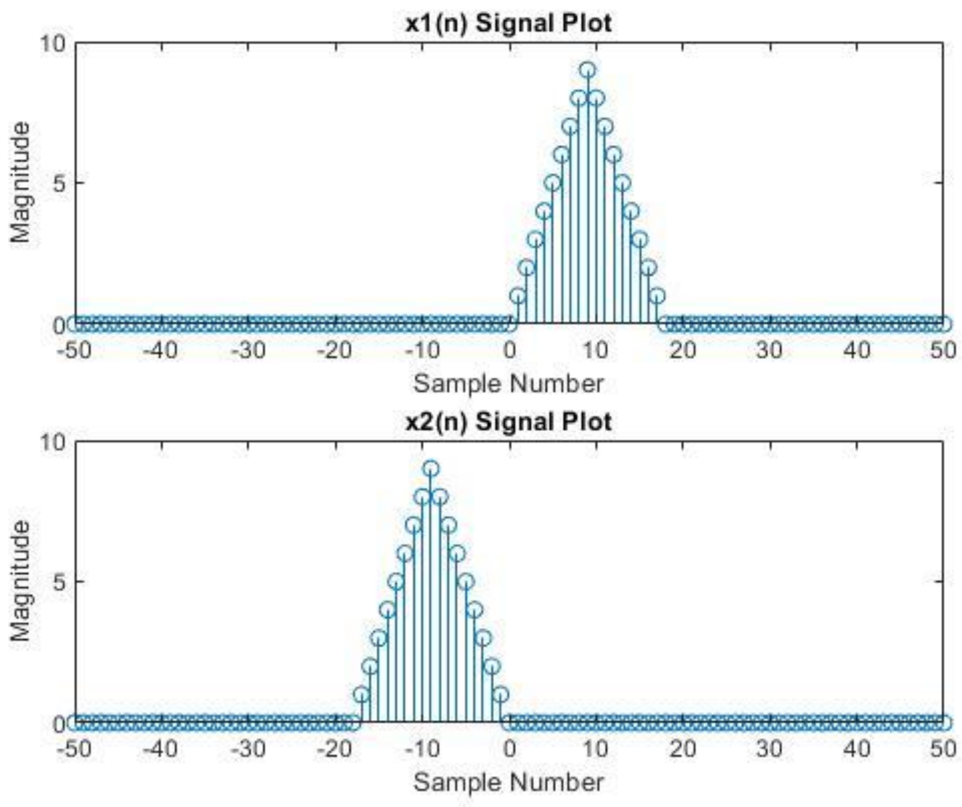
d. $[x_1(n).u(3-n)] * [x_2(n).u(n-3)]$.

MATLAB Code for Input Declaration and Plotting:

```
clc;
clear all;
close all;

ID = 4; %Student ID = 1906044
P = 5 + ID;
n = 1:P;
step = 1.*(n>=0);

%Function Generation
%General triangular function from convolution

[y,n_y] = cnvl(step,n,step,n);
[sig_1,n1] = shift(y,n_y,-1);
[sig_2,n2] = shift(y,n_y,-2*P-1);
```

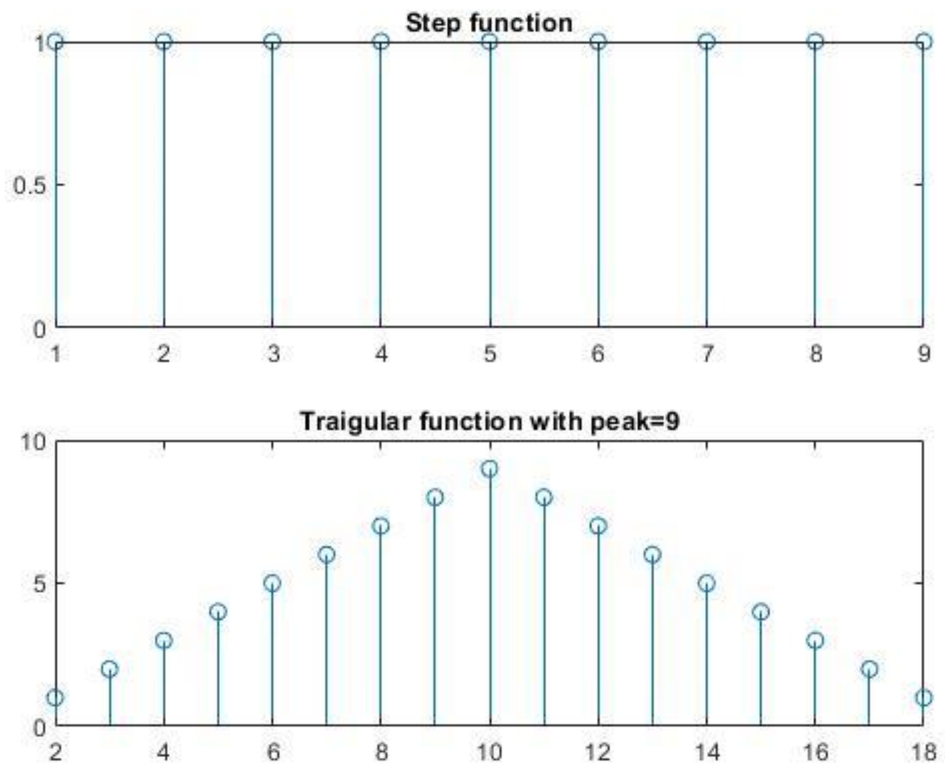
```
n_x = -50:50;

% x1(n)
x1 = zeros(1,length(n_x));
x1(find(n_x>=n1(1) & n_x<=n1(end))) = sig_1;

% x2(n)
x2 = zeros(1,length(n_x));
x2(find(n_x>=n2(1) & n_x<=n2(end))) = sig_2;

%Plotting
figure(1)
subplot(3,1,1)
stem(n_x,x1);
title('x1(n) Signal Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n_x,x2);
title('x2(n) Signal Plot')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

Here we used step functions to create a triangular function through convolution. The peak value we wanted for our triangular function was set as the upper limit in the x axis for step function. And the lower limit was one, thus creating a triangle with no zero value.



Once we got the triangle, we shifted and added zero value paddings to create our final $x_1(n)$ and $x_2(n)$ functions with a common range of sample number. The range was selected arbitrarily, and quite a large range will later help us to do later operations with much ease.

Problem 1 (a):

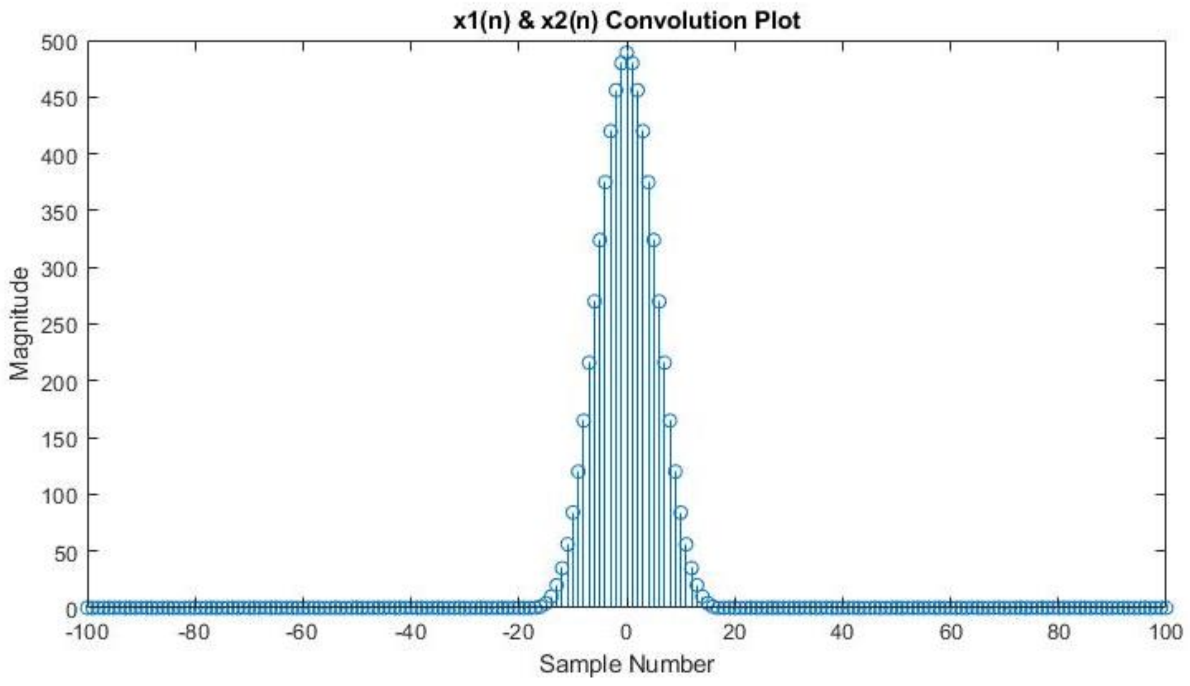
Question:

a. $x_1(n) * x_2(n)$.

MATLAB code:

```
%% 1_a
% x1(n)*x2(n)
[x_a,n_xa] = cnvl(x1,n_x,x2,n_x);

%Plotting
stem(n_xa,x_a)
title('x1(n) & x2(n) Convolution Plot')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

We know that convolution of two triangular signals result in a gaussian signal as we can see from the plot.

Problem 1 (b):

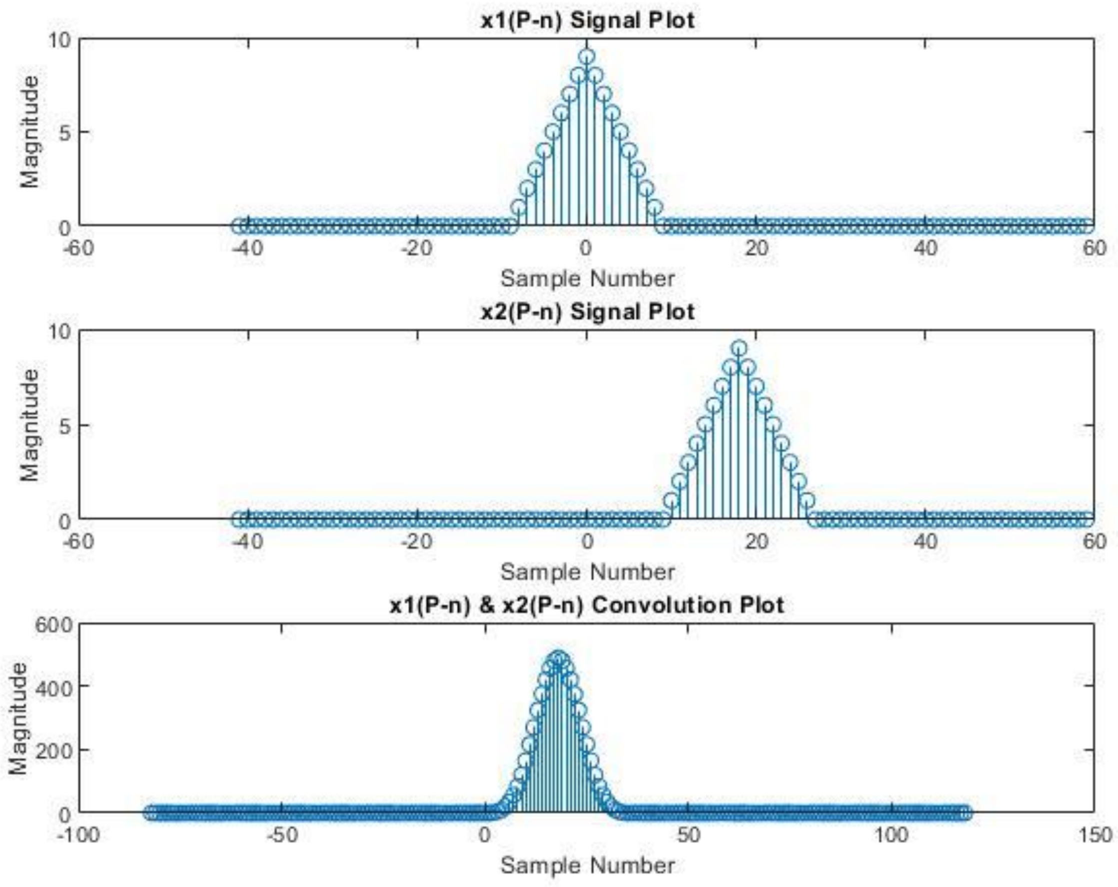
Question:

b. $x_1(P-n) * x_2(P-n)$.

MATLAB code:

```
%% 1_b

%Folding
% x1(-n)
[x1_fold,nx1_fold] = fold(x1,n_x);
% x2(-n)
[x2_fold,nx2_fold] = fold(x2,n_x);

%Shifting
% x1(P-n)
[x1_fold_shift,nx1_fold_shift] = shift(x1_fold,nx1_fold,P);
% x2(P-n)
[x2_fold_shift,nx2_fold_shift] = shift(x2_fold,nx2_fold,P);

% x1(P-n)*x2(P-n)
[x_b,n_xb] = cnvl(x1_fold_shift,nx1_fold_shift,x2_fold_shift,nx2_fold_shift);

%Plotting
figure(2)
subplot(3,1,1)
stem(nx1_fold_shift,x1_fold_shift)
title('x1(P-n) Signal Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(nx2_fold_shift,x2_fold_shift)
title('x2(P-n) Signal Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n_xb,x_b)
title('x1(P-n) & x2(P-n) Convolution Plot')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

The convolution is once again a gaussian curve resulting from two shifted triangular signals.

Problem (c):

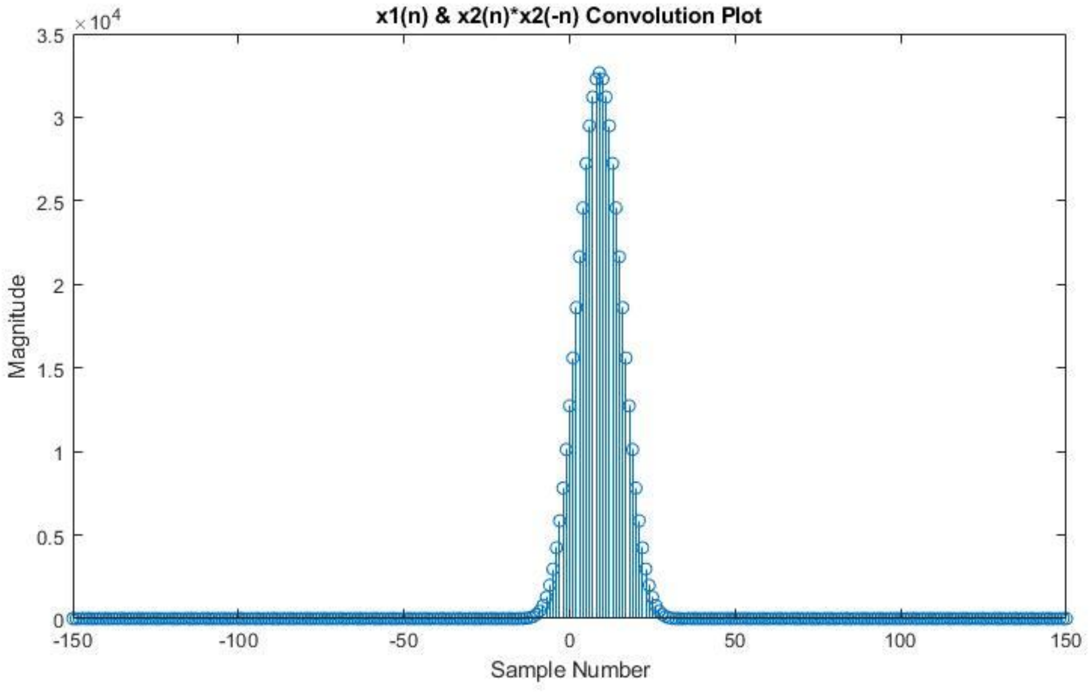
Question:

c. $x_1(n) * [x_2(n) * x_2(-n)]$.

MATLAB code:

```
%% 1_c
% x2(-n)
[x2_fold,nx2_fold] = fold(x2,n_x);

% x2(n)*x2(-n)
[x_ci,n_xci]=cnvl(x2,n_x,x2_fold,nx2_fold);

% x1(n)*[x2(n)*x2(-n)]
[x_c,n_xc] = cnvl(x1,n_x,x_ci,n_xci);

%Plotting
stem(n_xc,x_c);
title('x1(n) & {x2(n)*x2(-n)} Convolution Plot')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

The convolution of three triangular signals gives us a gaussian curve, this time the curve is much more compressed.

Problem (d):

Question:

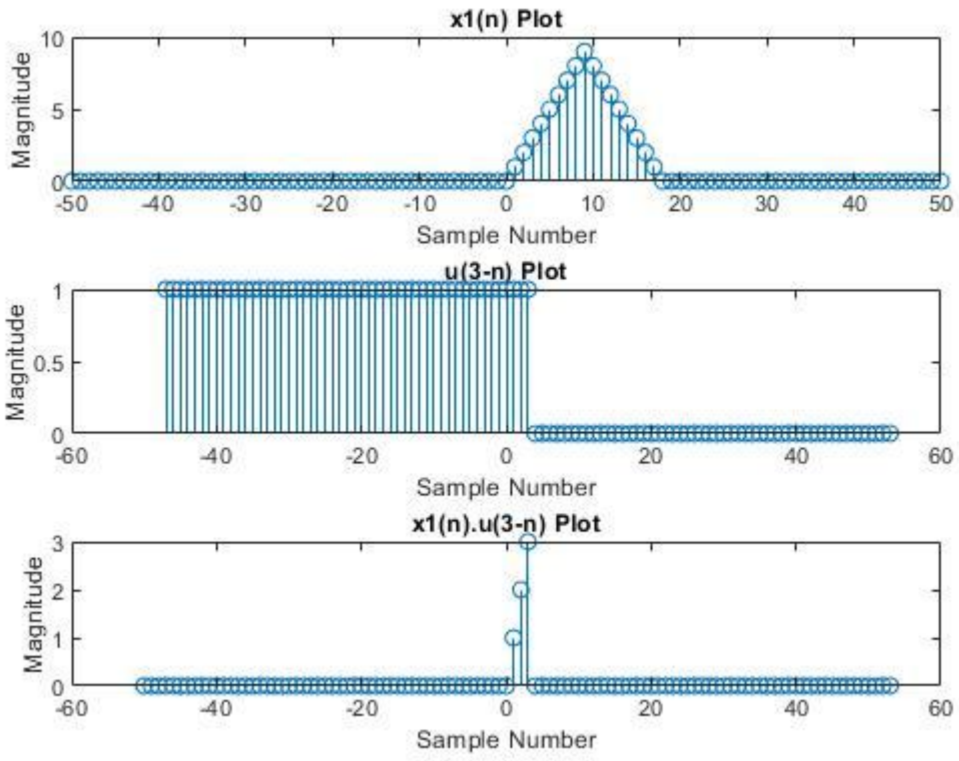
d. $[x_1(n).u(3-n)] * [x_2(n).u(n-3)]$.

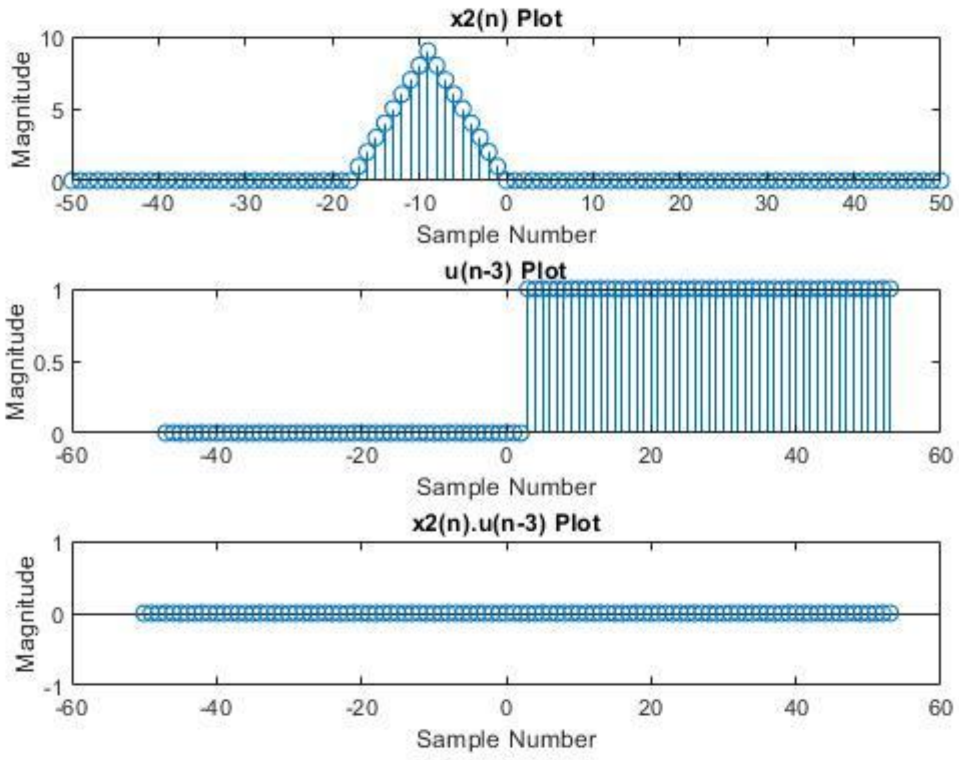
MATLAB code:

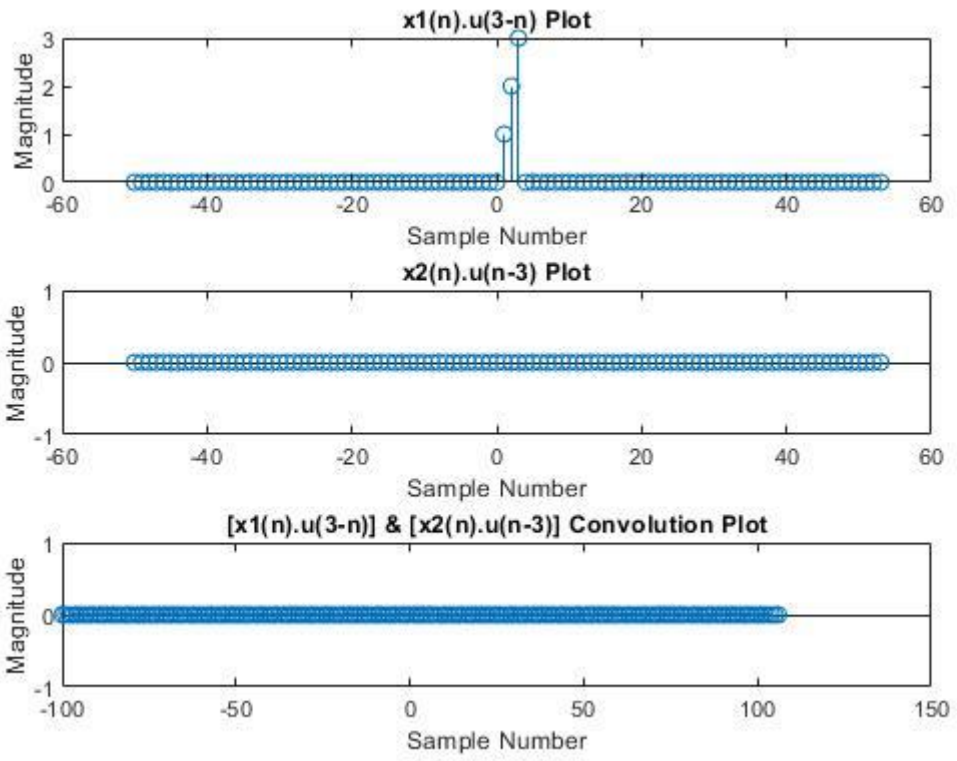
```
%% 1_d
step = 1.*(n_x>=0);

% u(3-n)
[step1_fold,n1_fold] = fold(step,n_x);
[step1_fold_shift,n1_fold_shift] = shift(step1_fold,n1_fold,3);

% u(n-3)
[step2_shift,n2_shift] = shift(step,n_x,3);

% x1(n).u(3-n)
[x_d1,n_d1] = product_1(x1,n_x,step1_fold_shift,n1_fold_shift);

% x2(n).u(n-3)
[x_d2,n_d2] = product_1(x2,n_x,step2_shift,n2_shift);

%[x1(n).u(3-n)]*[x2(n).u(n-3)]
[x_d,n_xd] = cnvl(x_d1,n_d1,x_d2,n_d2);


%Plotting

figure(1)

subplot(3,1,1)
stem(n_x,x1)
title('x1(n) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n1_fold_shift,step1_fold_shift)
title('u(3-n) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n_d1,x_d1)
title('x1(n).u(3-n) Plot')
xlabel('Sample Number')
ylabel('Magnitude')
```

```
figure(2)

subplot(3,1,1)
stem(n_x,x2)
title('x2(n) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n2_shift,step2_shift)
title('u(n-3) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n_d2,x_d2)
title('x2(n).u(n-3) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

figure(3)

subplot(3,1,1)
stem(n_d1,x_d1)
title('x1(n).u(3-n) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,2)
stem(n_d2,x_d2)
title('x2(n).u(n-3) Plot')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(3,1,3)
stem(n_xd,x_d)
title('[x1(n).u(3-n)] & [x2(n).u(n-3)] Convolution Plot')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:


x1(n) Plot
Magnitude
Sample Number
u(3-n) Plot
Magnitude
Sample Number
x1(n).u(3-n) Plot
Magnitude
Sample Number


x2(n) Plot
Magnitude
Sample Number
u(n-3) Plot
Magnitude
Sample Number
x2(n).u(n-3) Plot
Magnitude
Sample Number



Discussion:

Since, u(n-3) and $x_2(n)$ do not have any common region with non-zero values, their product function gives zero output. Even though, the product of u(3-n) and $x_1(n)$ is nonzero for a certain range, convoluting this with zero results in a zero signal at the final step.

Problem 2

Question:

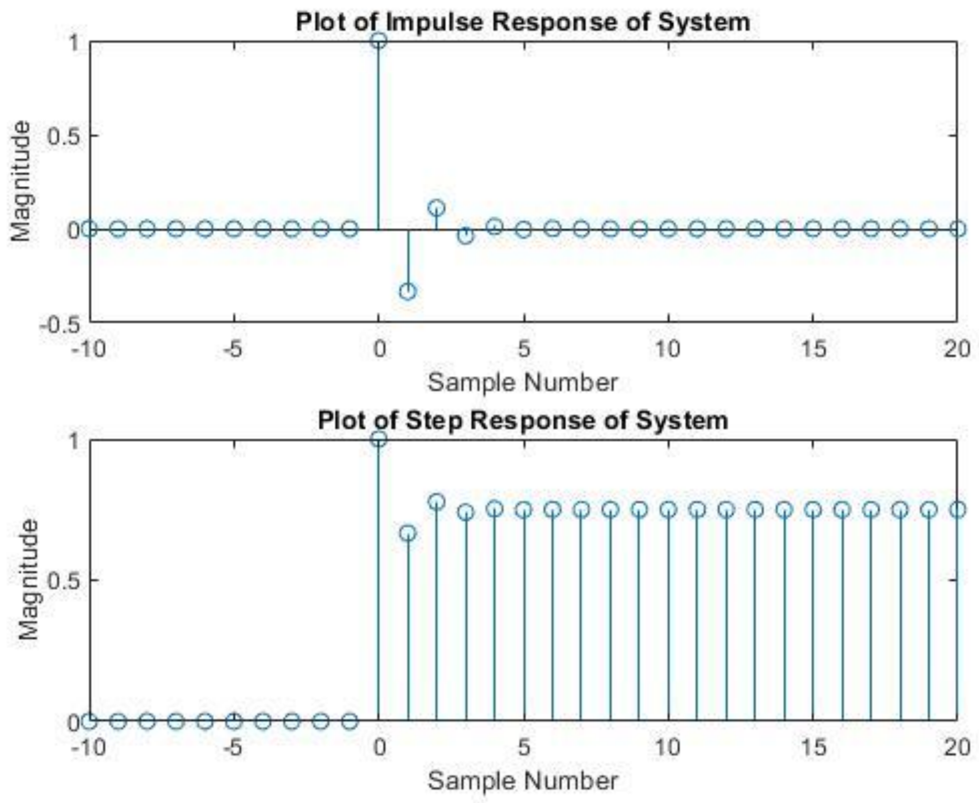
2. Consider the following difference equation. You **must** do this task manually with loops. Find the system function.

$$y(n) + \frac{4y(n-1)}{Last\ Digit\ of\ your\ ID + 8} = x(n)$$

MATLAB Code:

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

ID = 4; %Student ID = 1906044
coeff = 4/(ID+8);

% Assumption: Initially Relaxed LTI
%% Finding Impulse Response
n = -10:20;
x=1.*(n==0);
for i=1:length(n)
    if n(i)<0
        h(i)=0;
    else
        h(i) = -coeff*h(i-1)+x(i);
    end
end

%% Finding Step Response
n = -10:20;
x=1.*(n>=0);
for i=1:length(n)
    if n(i)<0
        u(i)=0;
    else
        u(i) = -coeff*u(i-1)+x(i);
    end
end

%Plotting
subplot(2,1,1)
stem(n,h);
title('Plot of Impulse Response of System')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(n,u);
title('Plot of Step Response of System')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

In a relaxed system, y(n) = 0 when n < 0

To get the system response we used an impulse input and a unit step input.

The impulse response follows the input impulse starting with a sharp peak at n=0 and a few decaying oscillations. Soon the output signal dies out eventually giving zero output imitating the nature of input impulse. Thus, this is a stable response.

The unit step response follows the input step function with some transient fluctuations at the start when n = 0. Soon, the output starts to give constant value 1, holding onto this for as long as n goes. Thus, this is a marginally stable response.

Problem 3

Question:

3. Consider the system described by the following difference equation:

$$y(n) + A_1 y(n-1) + A_2 y(n-2) = x(n) + 3x(n-3); \;\; A_1, A_2 \geq 0$$

Use a bounded input of your choice. Find the stability status of the system by varying:

a. The coefficient A_1 and plot the transition points. For this, assume $A_2 = 0.5$.

b. The coefficient A_2 and plot the transition points. For this, assume $A_1 = 0.5$.

a) Varying A$_1$ and Fixing A$_2$ ->

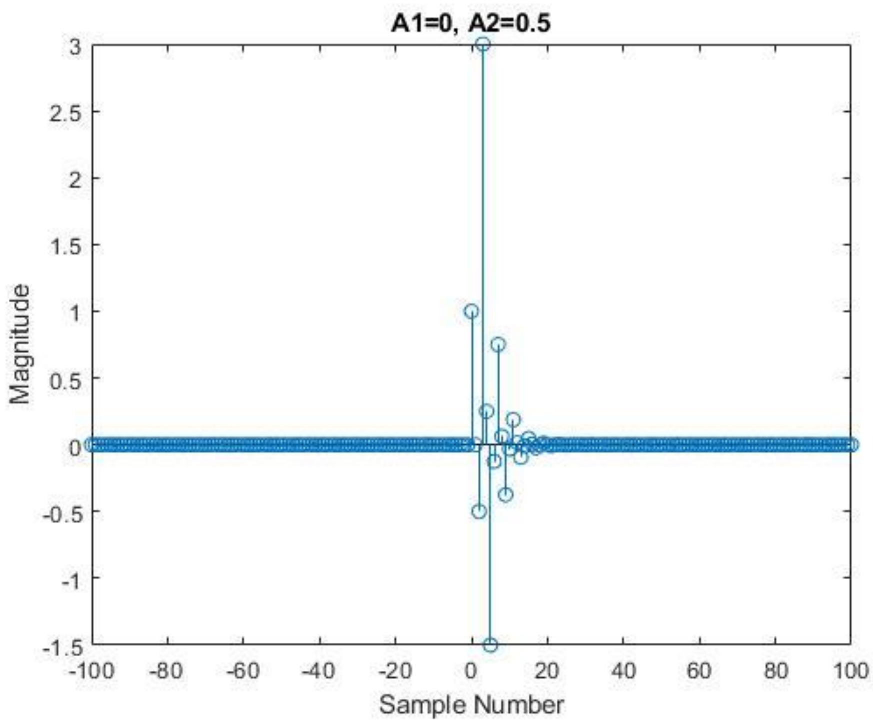
MATLAB Code:

```
clc;
clear all;
close all;

n = -20:50;
%Taking Bounded Impulse Input
x_del = 1.*(n==0);

% Varying A1 & Fixing A2
A2 = 0.5;

%% A1 = 0
A1 = 0;
a = [1 A1 A2]; %Coefficients of y
b = [1 0 0 3]; %Coefficients of x
y = filter(b,a,x_del);

stem(n,y);
title(['A1=' num2str(A1) ', A2=0.5'])
xlabel('Sample Number')
ylabel('Magnitude')
```

Here we will only plot for the critical values of A1.

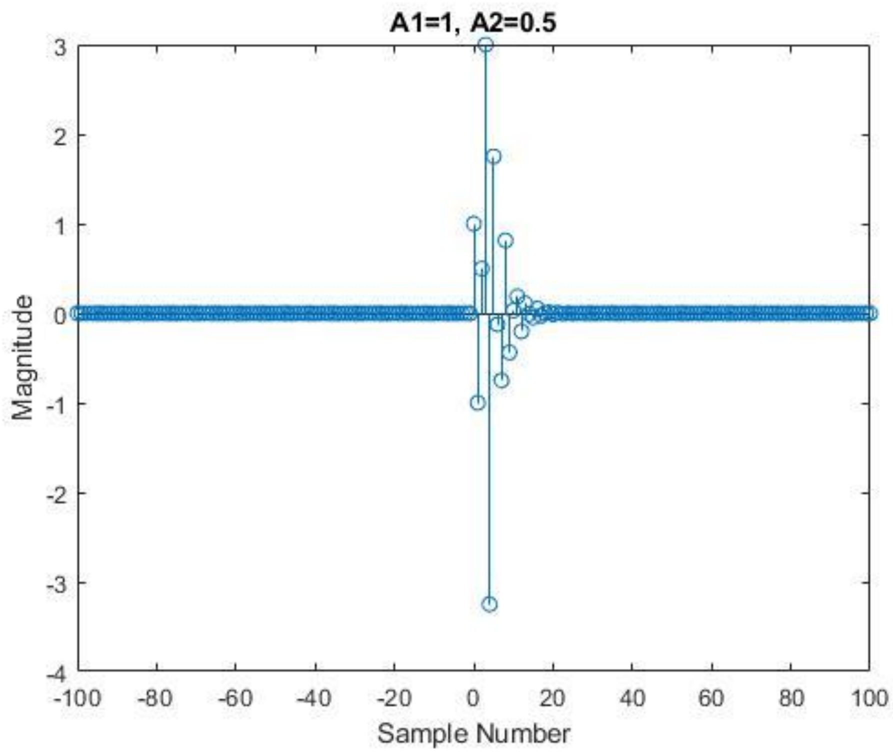
Plot for Different Values of A$_1$:

- ***$A_1 = 0$***



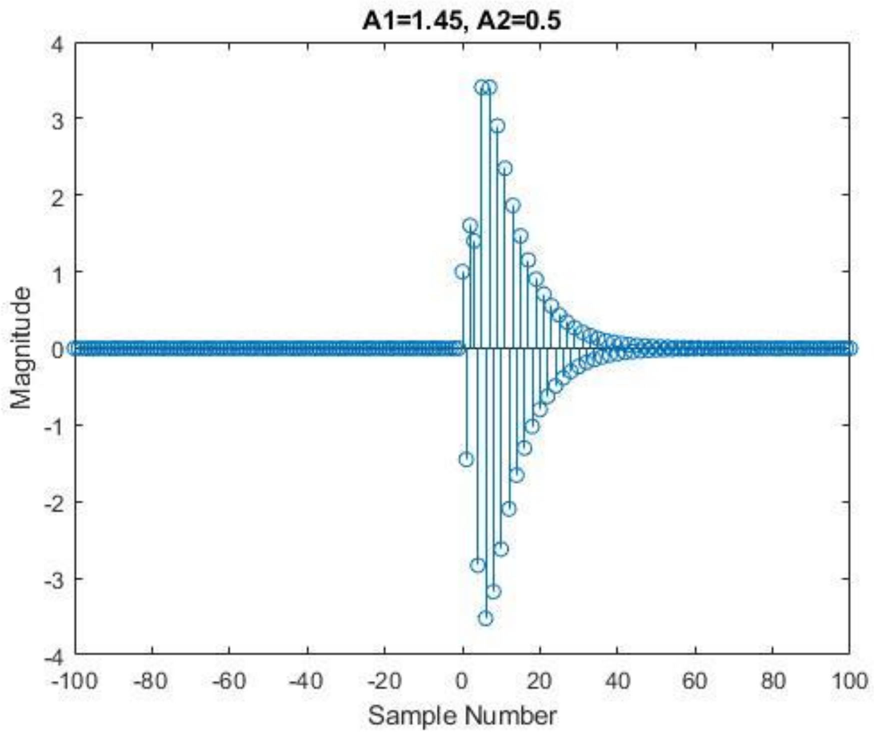
Discussion: As we can see the output signal quickly dies out, so the LCCDE is stable.

- ***$A_1 = 1$***



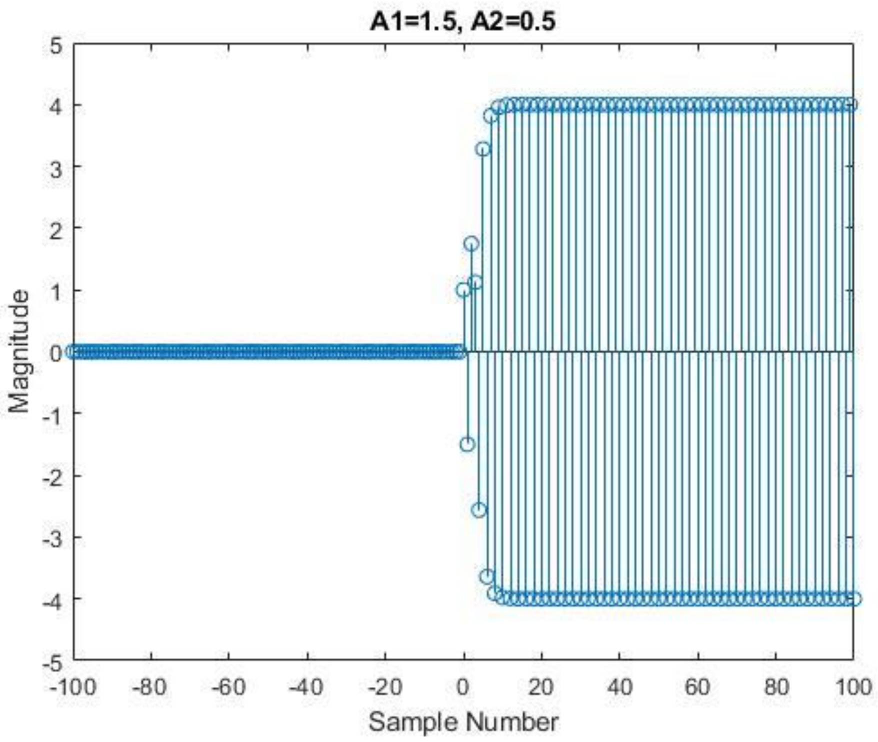
Discussion: Output eventually dies out; the system remains stable.

- ***$A_1 = 1.45$***



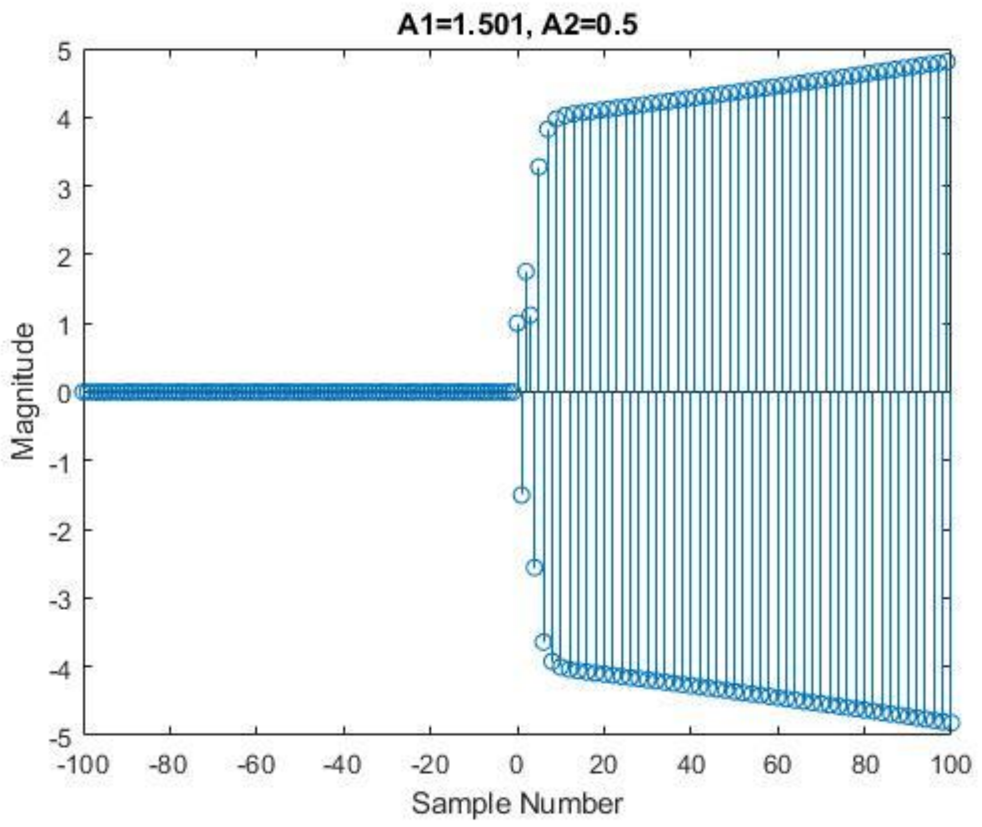
Discussion: Output dies out but takes comparatively more time. The system is still stable.

- ***$A_1 = 1.5$***



Discussion: The system is marginally stable, since the output does not die out but holds a constant value 4 for all values of n.

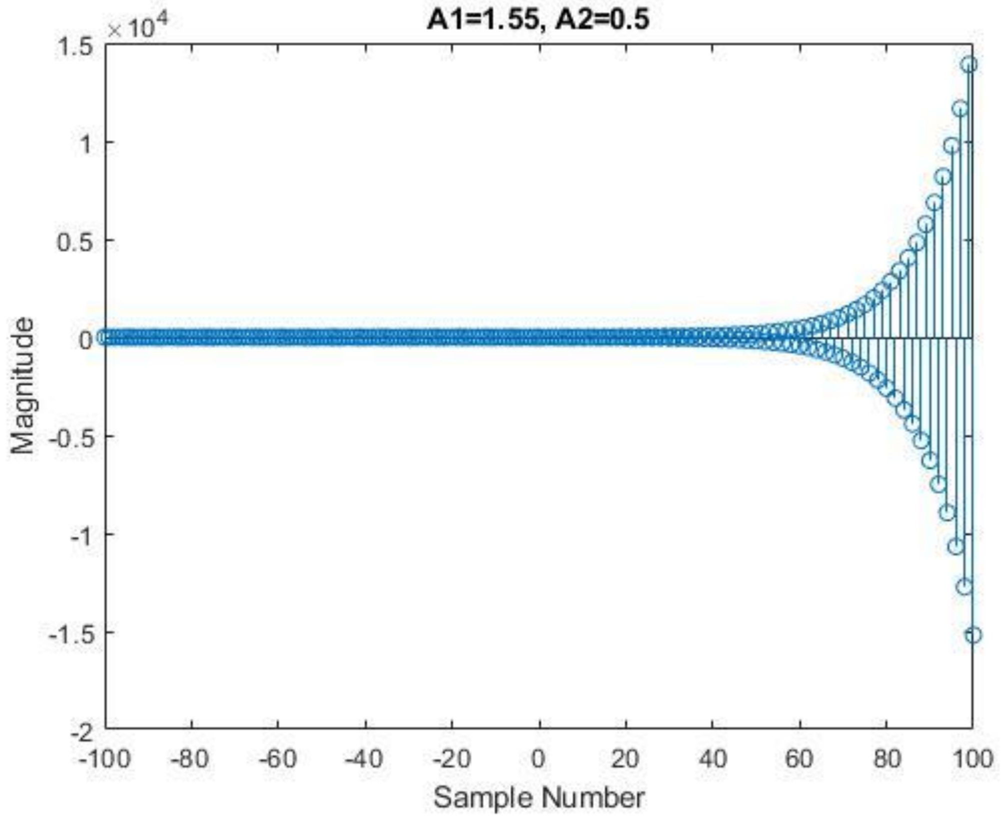
- ***A_1 = 1.501***



Discussion:

Output becomes unstable for a very small increase in A1. It is growing in nature and will continue to increase as long as n goes. Thus, the system output is not bounded rather is an unstable system.

- ***A_1 = 1.55***



Discussion:

As we increase the value of A1, the system output increases more rapidly with n becoming more unstable.

Conclusion:

A1 = 1.5 is the critical point. For A1 < 1.5, the system remains stable. Beyond this value the system is unstable and for A1 = 1.5, the system is marginally stable.

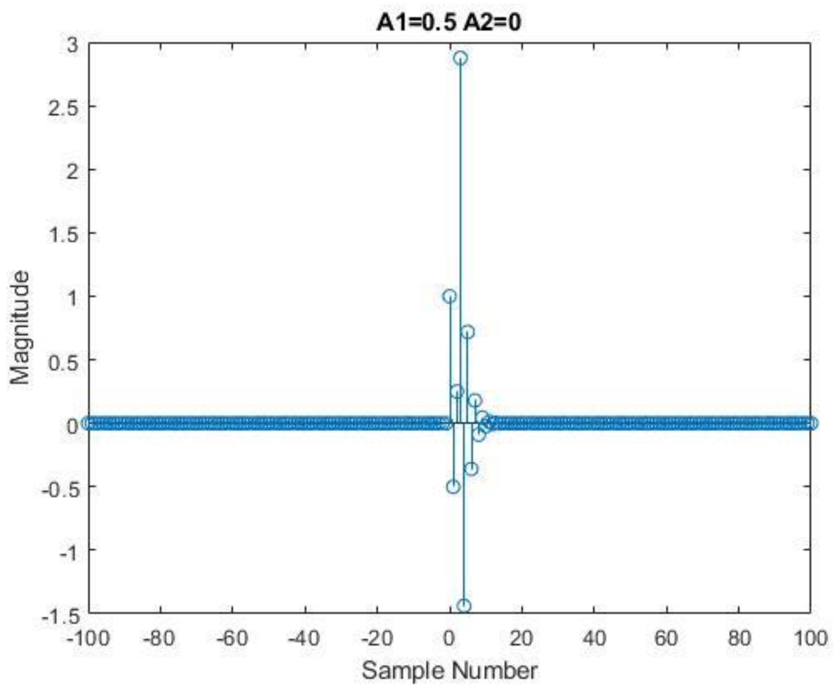
b) Varying A_2 and Fixing A_1 ->

MATLAB Code:

```
clc;
clear all;
close all;
n = -20:50;
%Taking Bounded Impulse Input
x_del = 1.*(n==0);

% Varying A1 & Fixing A2
A2 = 0.5;
%% A1 = 0
A1 = 0;
a = [1 A1 A2]; %Coefficients of y
b = [1 0 0 3]; %Coefficients of x
y = filter(b,a,x_del);

stem(n,y);
title(['A1=0.5 A2=' num2str(A2)])
xlabel('Sample Number')
ylabel('Magnitude')
```

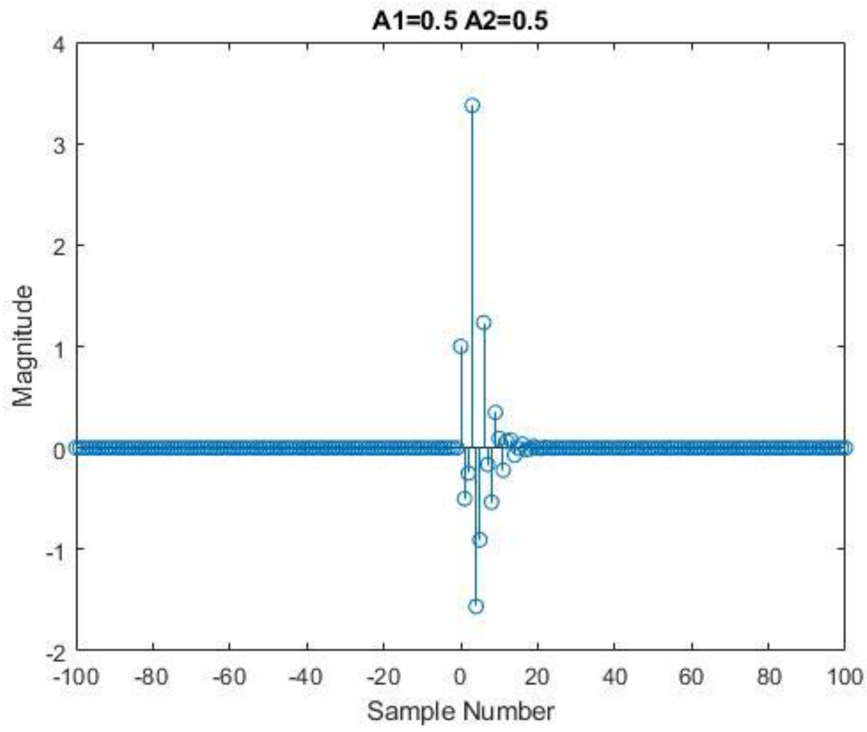
Plot for Different Values of A_2:

- ***$A_2 = 0$***



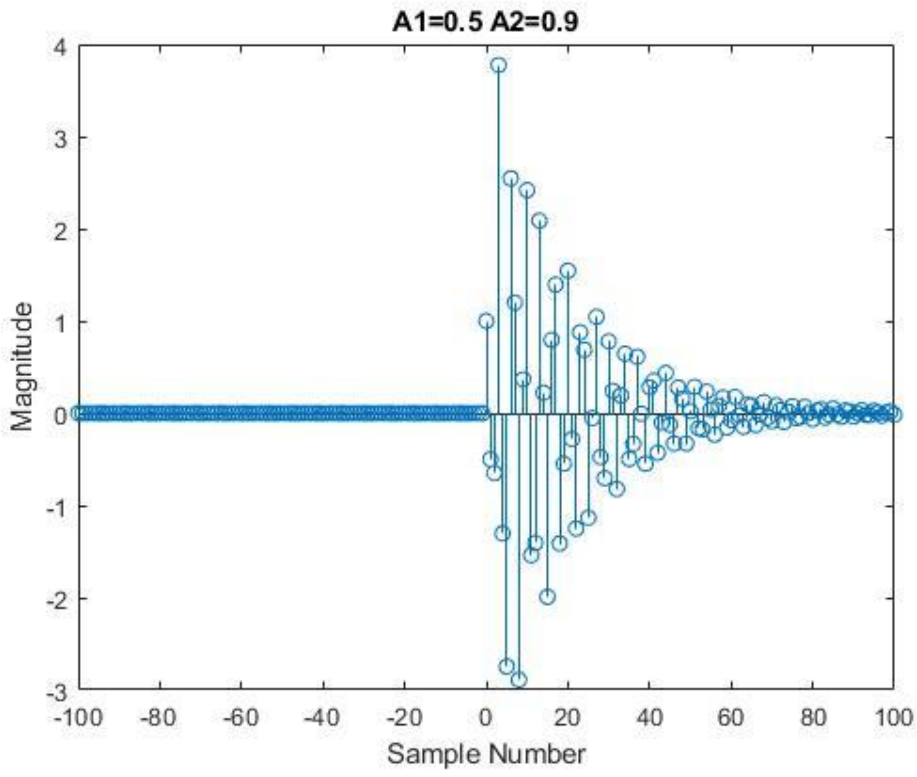
Discussion: As we can see the output signal quickly dies out, so the LCCDE is stable.

➢ ***$A_2 = 0.5$***



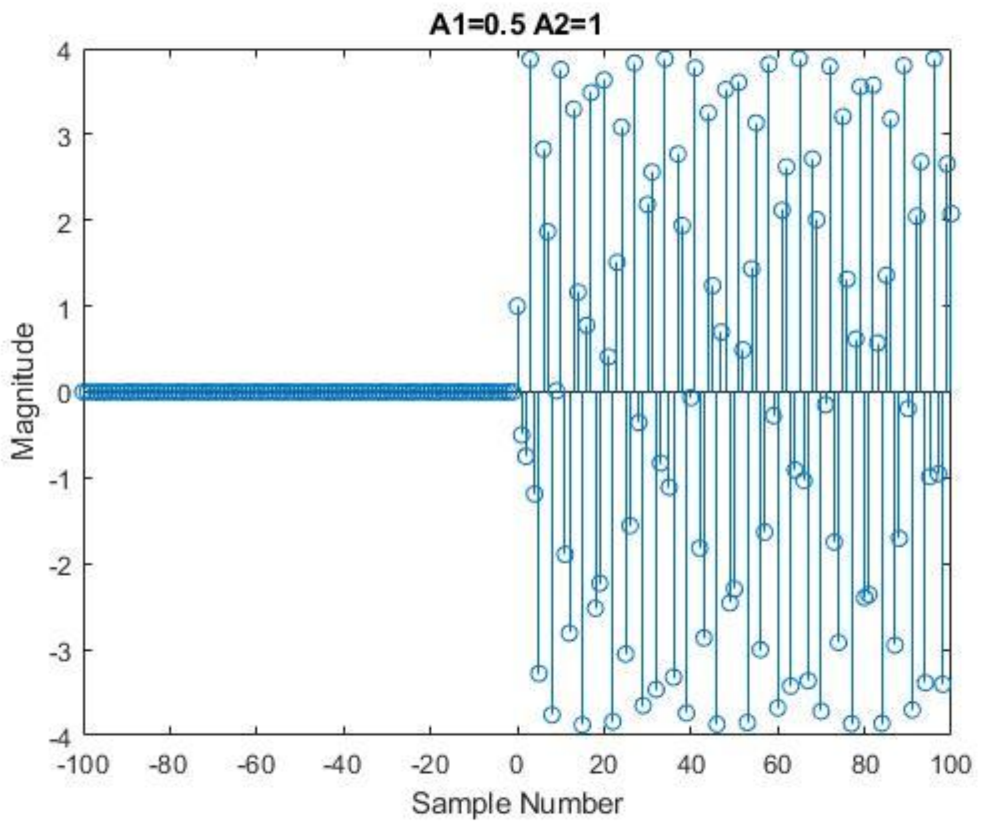
Discussion: Output eventually dies out; the system remains stable.

➢ ***$A_2 = 0.9$***



Discussion: Output dies out but takes comparatively more time. The system is still stable.

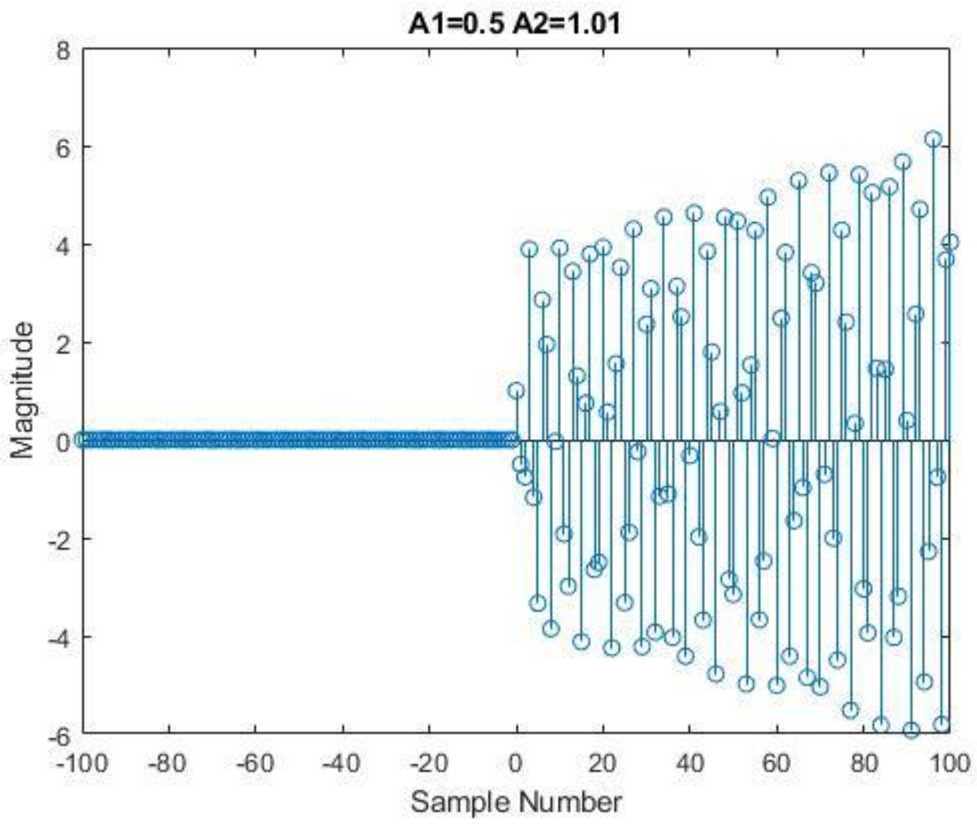
- ***$A_2 = 1$***



Discussion:

The system is marginally stable, since the output does not die out but stays below a constant value for all values of n.

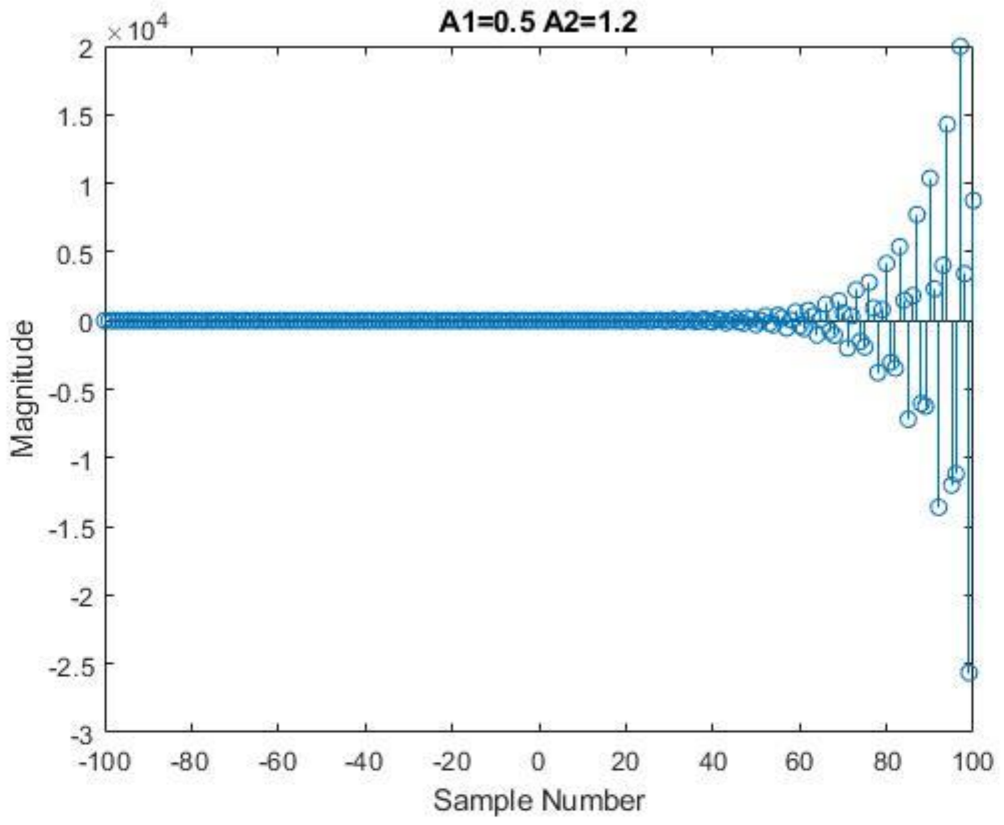
- ***$A_2 = 1.01$***



Discussion:

Output becomes unstable for a very small increase in A2. It is growing in nature and will continue to increase as long as n goes. Thus, the system output is not bounded rather is an unstable system.

- ***A_2 = 1.2***



Discussion:

As we increase the value of A2, the system output increases more rapidly with n becoming more unstable.

Conclusion:

A2 = 1 is the critical point. For A2 < 1, the system remains stable. Beyond this value the system is unstable and for A2 = 1, the system is marginally stable.

Problem 4

Question:

4. Consider the two systems described by the following difference equations:

$$(i)\quad y(n)+\frac{y(n-1)}{6.5-Last\ Digit\ of\ your\ ID}+\frac{y(n-2)}{7.5-Last\ Digit\ of\ your\ ID}=4x(n)$$

$$(ii)\quad y(n)+\frac{5y(n-1)}{Last\ Digit\ of\ your\ ID-1.5}+\frac{y(n-2)}{Last\ Digit\ of\ your\ ID-1.5}=x(n)+x(n-3)$$

In the range of $-10\le n\le 50$ for both described systems:

MATLAB Code for Basic Parameter Declaration:

```
clc;
clear all;
close all;

n = -10:50;
ID = 4; %Student ID = 1906044

coef1 = 1/(6.5-ID);
coef2 = 1/(7.5-ID);
coef3 = 5/(ID-1.5);
coef4 = 1/(ID-1.5);

%System 1
a_c1 = [1 coef1 coef2];
b_c1 = 4;

%System 2
a_c2 = [1 coef3 coef4];
b_c2 = [1 0 0 1];
```

Problem (a):

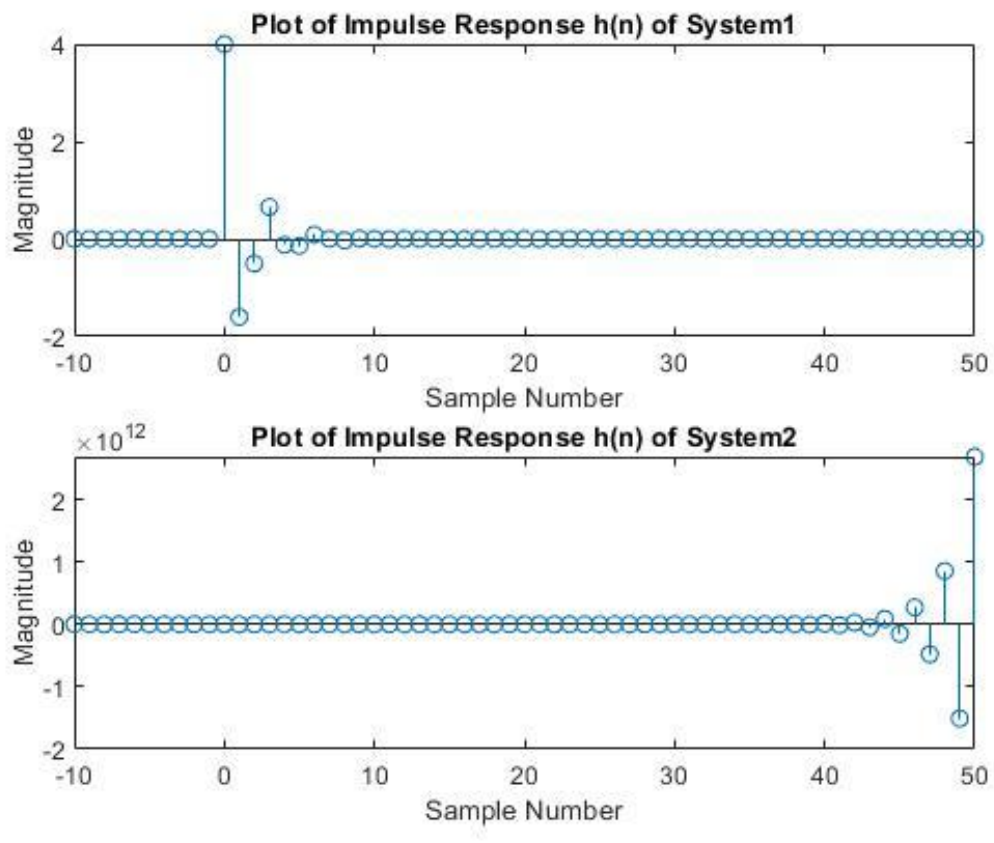
Question:

a. Plot $h(n)$.

MATLAB code:

```
delta =1.*(n==0);
h1 = filter(b_c1,a_c1,delta);
h2 = filter(b_c2,a_c2,delta);

%Plotting
subplot(2,1,1)
stem(n,h1);
title('Plot of Impulse Response h(n) of System1')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(n,h2);
title('Plot of Impulse Response h(n) of System2')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion:

System 1 is BIBO stable, and the output quickly decays. But system 2 is unstable with a growing output.

Problem (b):

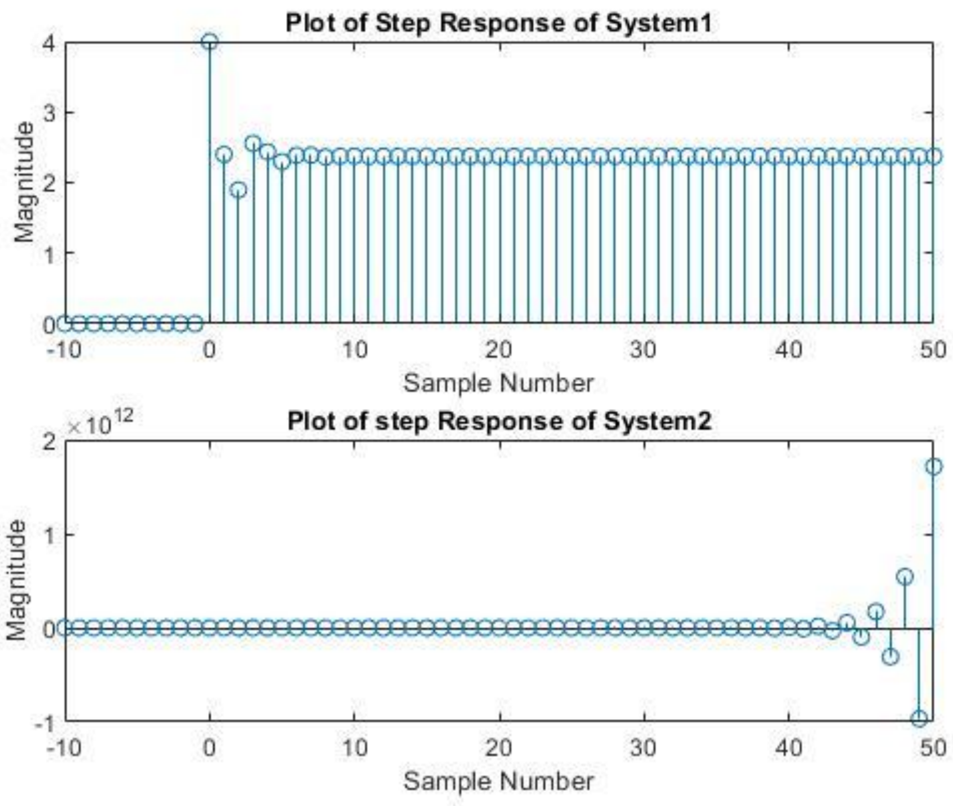
Question:

b. Plot the unit step response.

MATLAB code:

```
%% 4_b
step = 1.*(n>=0);
u1 = filter(b_c1,a_c1,step);
u2 = filter(b_c2,a_c2,step);

%Plotting
subplot(2,1,1)
stem(n,u1);
title('Plot of Step Response of System1')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(n,u2);
title('Plot of step Response of System2')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion: System 2 is marginally stable, and the output quickly decays. But system 2 is unstable with a growing output.

Problem (c):

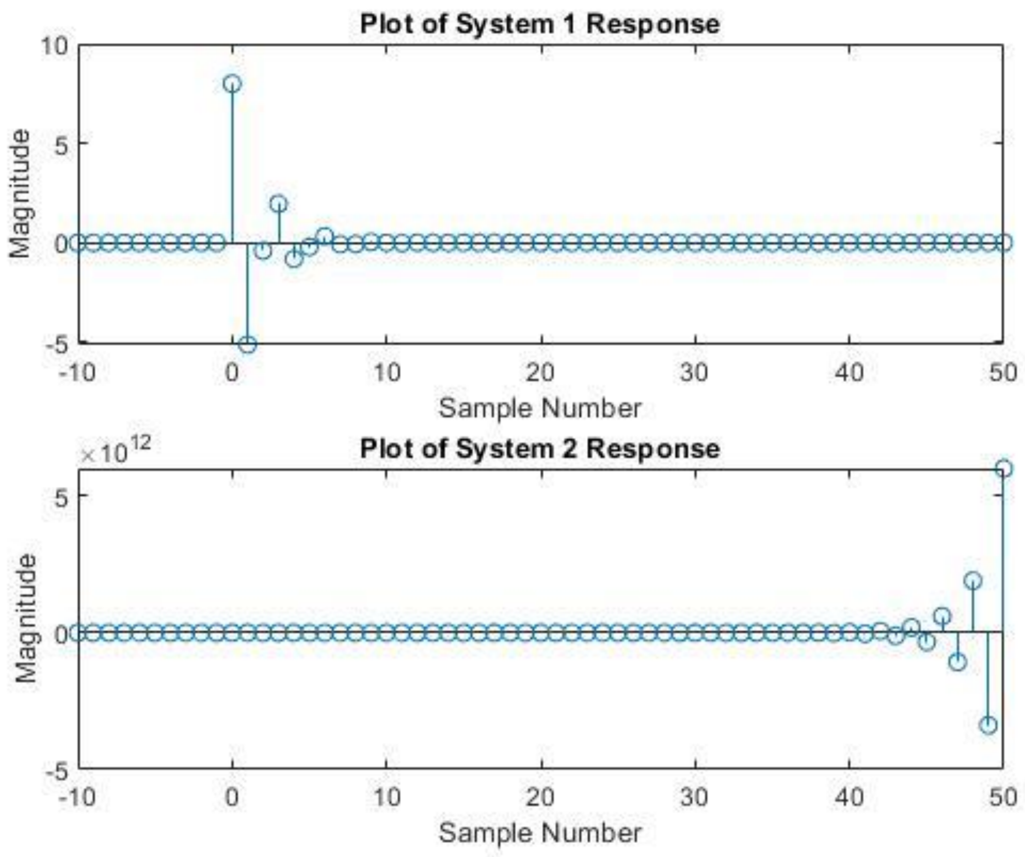
Question:

c. Plot the response if your input is $2e^{-n}.\cos(4n)\,.\,u(n)$.

MATLAB code:

```
%% 4_c
u = 1.*(n>=0); %Taking Step Input
x_c = 2.*exp(-n).*cos(4*n).*u;

y_c1 = filter(b_c1,a_c1,x_c);
y_c2 = filter(b_c2,a_c2,x_c);

%plotting
subplot(2,1,1)
stem(n,y_c1)
title('Plot of System 1 Response')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(n,y_c2)
title('Plot of System 2 Response')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion: System 1 is stable, and the output quickly decays. But system 2 is unstable with a growing output.

Problem (d):

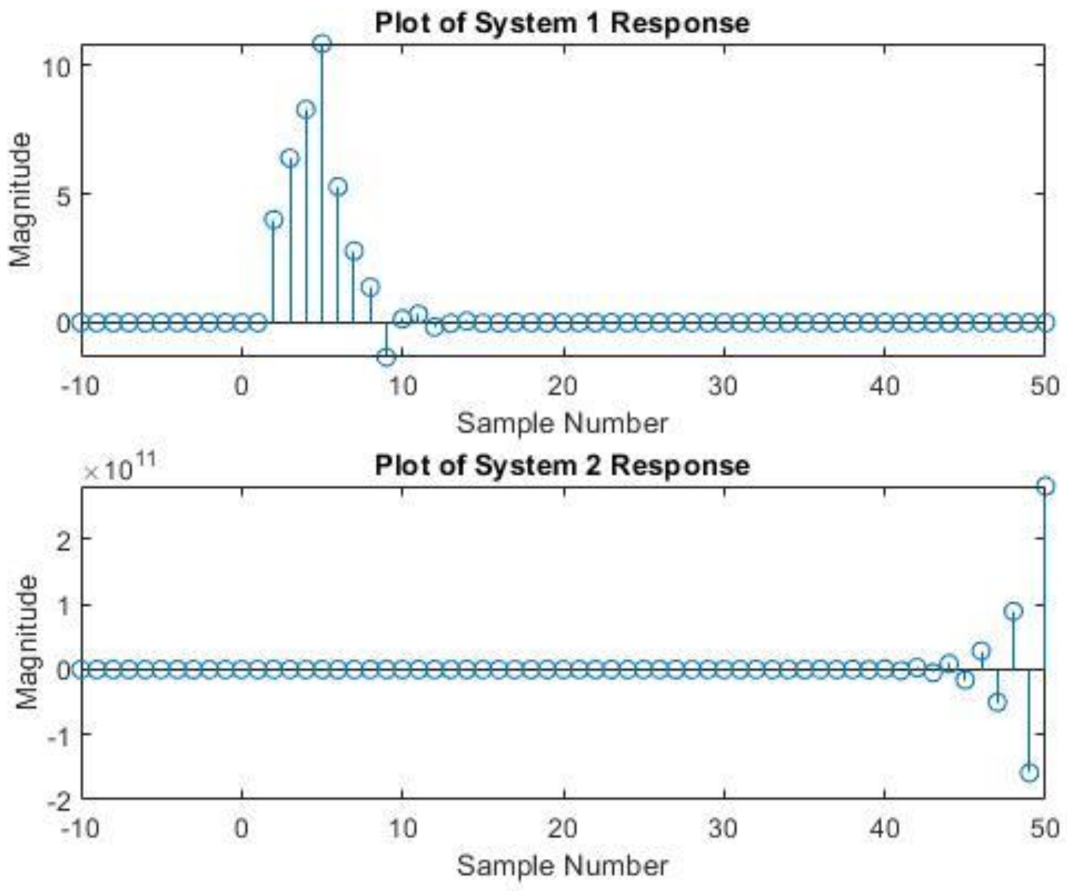
Question:

d. Plot the response if your input is $x_1(n)$ [from Q-1] with $P = 4$.

MATLAB code:

```
%% 4_d
P = 4;
n_d1 = 1:P;
step_d = 1.*(n_d1>=0);

[x_d1,nx_d1] = cnvl(step_d,n_d1,step_d,n_d1);
[sig_d1,n_d2] = shift(x_d1,nx_d1,0);

x_d = zeros(1,length(n));
x_d(find(n>=n_d2(1) & n<=n_d2(end))) = sig_d1;
y_d1 = filter(b_c1,a_c1,x_d);
y_d2 = filter(b_c2,a_c2,x_d);

% Plotting
subplot(2,1,1)
stem(n,y_d1)
title('Plot of System 1 Response')
xlabel('Sample Number')
ylabel('Magnitude')

subplot(2,1,2)
stem(n,y_d2)
title('Plot of System 2 Response')
xlabel('Sample Number')
ylabel('Magnitude')
```

Plot:



Discussion: System 1 is stable, and the output gradually decays. But system 2 is unstable with a growing output.

Problem (e):

Question:

e. Are any of the systems BIBO stable?

Discussion:

BIBO stable means for a certain system a bounded input will result in a bounded output. It as expressed in equation as,

For, input $x[n] \leq M < \infty$ we get output $y[n] \leq B < \infty$

In this problem, all the inputs were bounded input, but the outputs were not always bounded. System 2 was always unstable. Irrespective of the input, it gave a non-bounded growing output.

System 1 on the other hand gave bounded outputs for all the inputs. Even though, it was marginally stable for unit step input, the output held on to a constant value and did not grow further. Hence, system 1 is a BIBO stable system.

Conclusion:

In this experiment, we learnt about convolution and observed different graphs of convolution of different types of functions. We then got responses of s system solving its LCCDE equation in a manual way as well as using the function filter.

The filter() function helps to filter a range of data based on the coefficient and input function criteria we give. It then solves the equation and gives us the consequent output function.

Finally, we used several different inputs for two different systems to identify the stability. We learnt that BIBO stability comes from system constraint and always gives bounded output if the system is stable.

Thus, the experiment can be deemed successful in learning about convolution, LCCDE and system stability.