S.THINESH

E/15/366

EE387 – Signal Processing

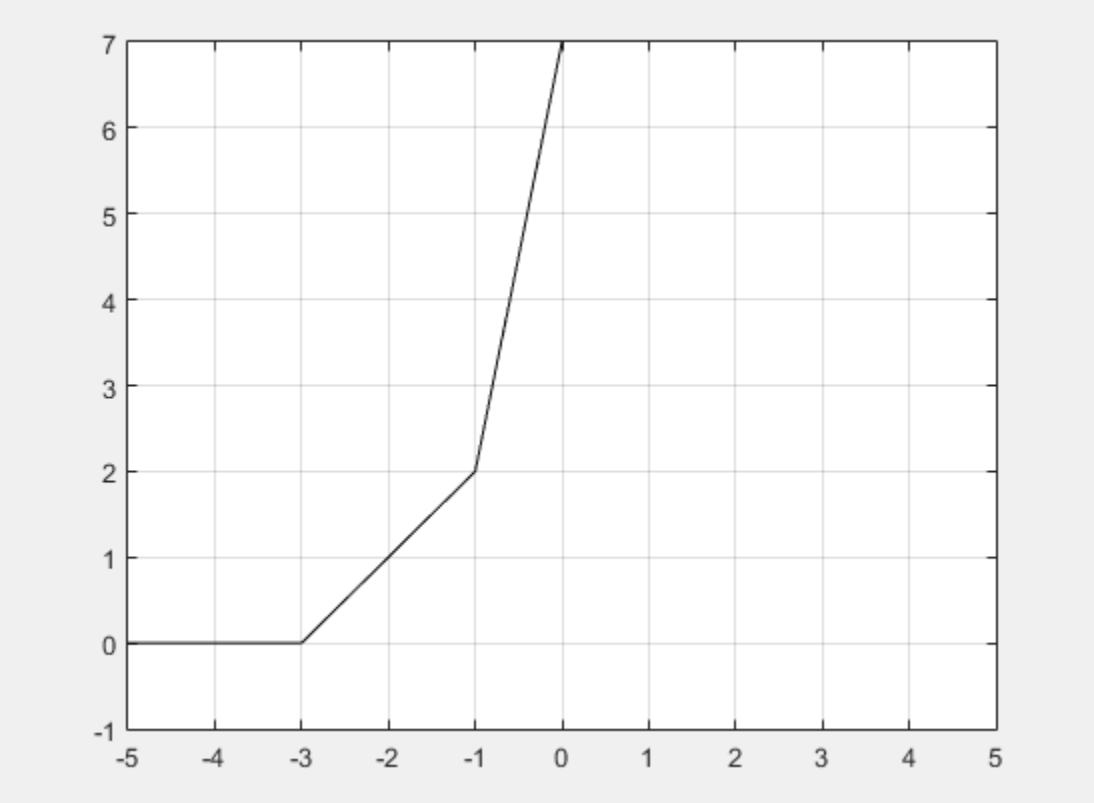
Lab01

PART 1: Basic Signal Representation in MATLAB

1.

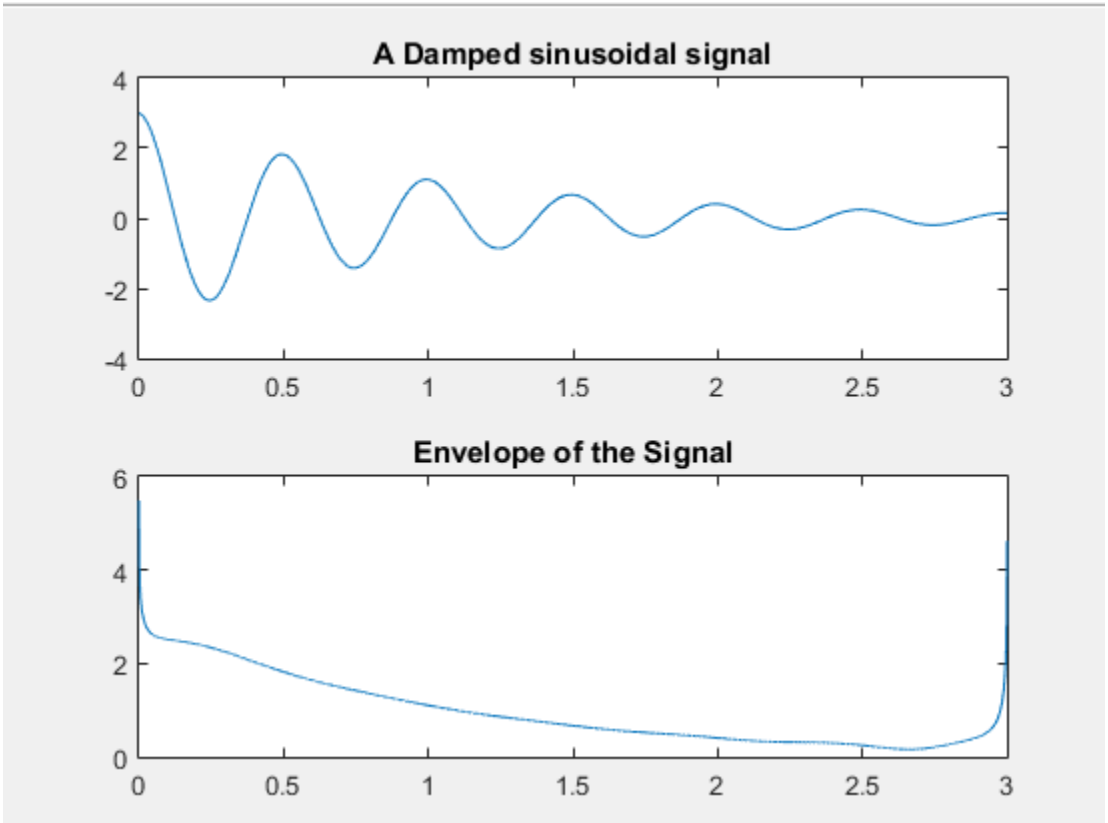
```
% Function for Ramp signal function
% t = length of time
% m = slope of the ramp function
% ad = advance (positive), delay (negative) factor


function y = ramp(t,m,ad)

n = length(t);
y = zeros(1,n);
for i = 1:n,
    if t(i)>= -ad,
    y(i) = m*(t(i) + ad);
    end

end


%Function for Unit Step functions
% t = length of time
% ad = advance (positive), delay (negative) factor


function y = ustep(t,ad)
n = length(t);
y = zeros(1,n);
for i = 1:n,
    if t(i)>= -ad
    y(i) = 1;
    end
end
```

```
%Write a Matlab program and necessary functions to generate the
following signal:
%y(t) = r(t+3) - 2r(t+1) +3r(t) - u(t-3)
%Then plot it and verify analytically that the obtained figure
is correct.

clear all;
Ts=0.01; t= -5:Ts:5;
y1 = ramp(t,1,3);
y2 = ramp(t,-2,1);
y3 = ramp(t,3,0);
y4 = ustep(t,-3);
y = y1-2*y2+3*y3-y4;
plot(t,y,'k');
axis([-5 5 -1 7]); grid
```



2. For the damped sinusoidal signal x(t) = 3e-tcos(4πt) write a MATLAB program to generate x(t) and its envelope, then plot.

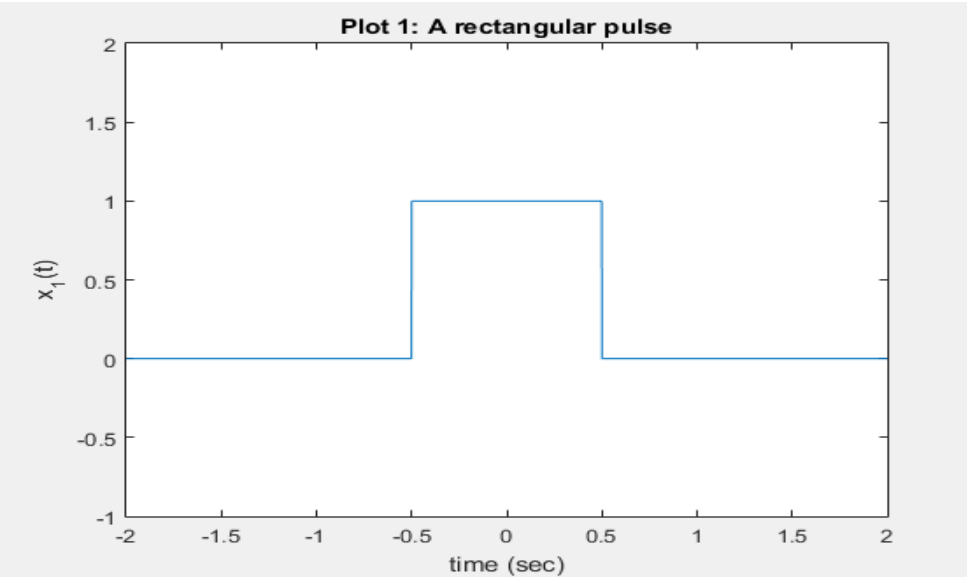
```
%For the damped sinusoidal signal x(t) = 3e-tcos(4?t) write a
MATLAB program to generate x(t) and its envelope, then plot.

%Damped sinusoidal signal
t = 0:1/1000:3;
x_t = 3.*exp(-t).*cos(4*pi*t);
subplot(2,1,1)
plot(t,x_t)
title('Damped sinusoidal signal')

%use command hilbert to find the envelope of a signal
r = abs(hilbert(x_t));
subplot(2,1,2)
plot(t,r)
title('Envelope of the Signal')
```



PART 2: Time-Domain Convolution

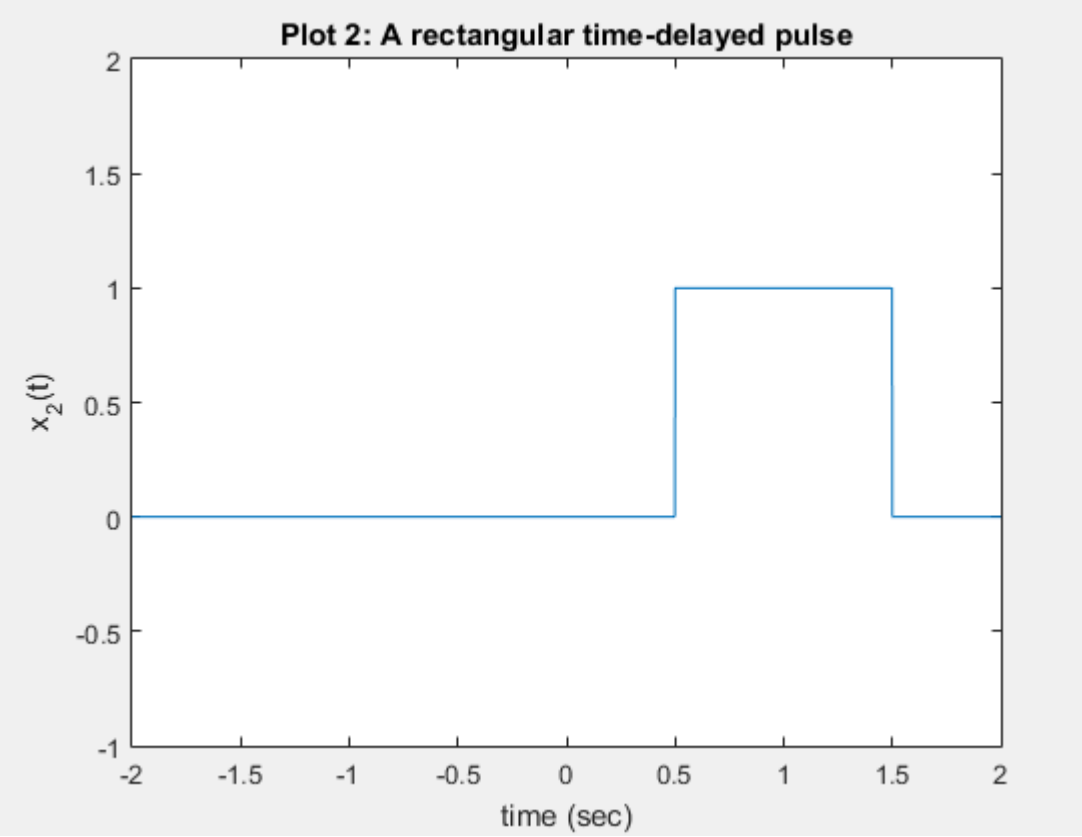
Creating a rectangular pulse in MATLAB

```
% Function for rectangular pulse function
function x = rect(t)

n = length(t);
x = zeros(1, n);
    for i = 1:n
       if t(i) >= -0.5 && t(i) < 0.5
        x(i) = 1;
       end
    end
end

% Main function for plotting time-delayed signal
% This function takes in a vector t of sample instants and
outputs the
% corresponding rectangular pulse contained in the function x
fs = 1000;
Ts = 1/fs;

t = -5:Ts:5;
x1 = rect(t);
plot(t,x1);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_1(t)');
title('Plot 1: A rectangular pulse');
```

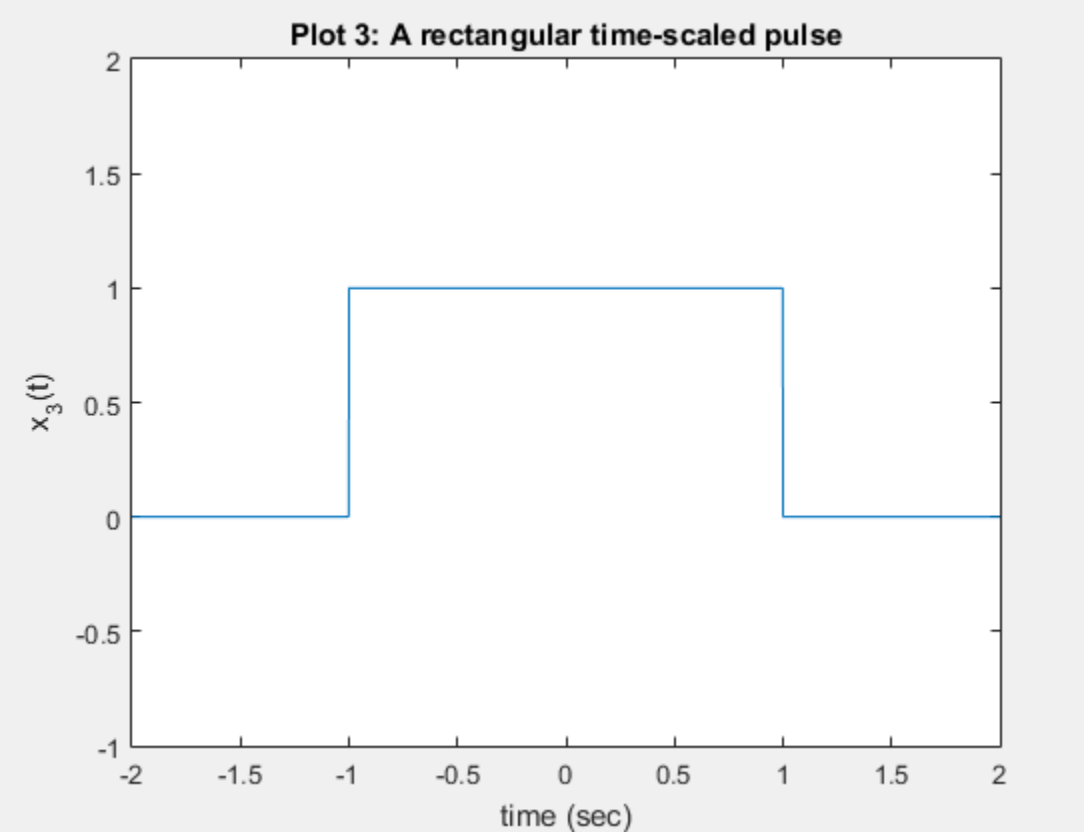


Elementary Signal Operations

```
% First let's create and plot the time-delayed signal,
% x2(t) = rect(t-1)

fs = 1000;
Ts = 1/fs;
t = -5:Ts:5;
x2 = rect(t-1);

plot(t,x2);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_2(t)');
title('Plot 2: A rectangular time-delayed pulse');
```

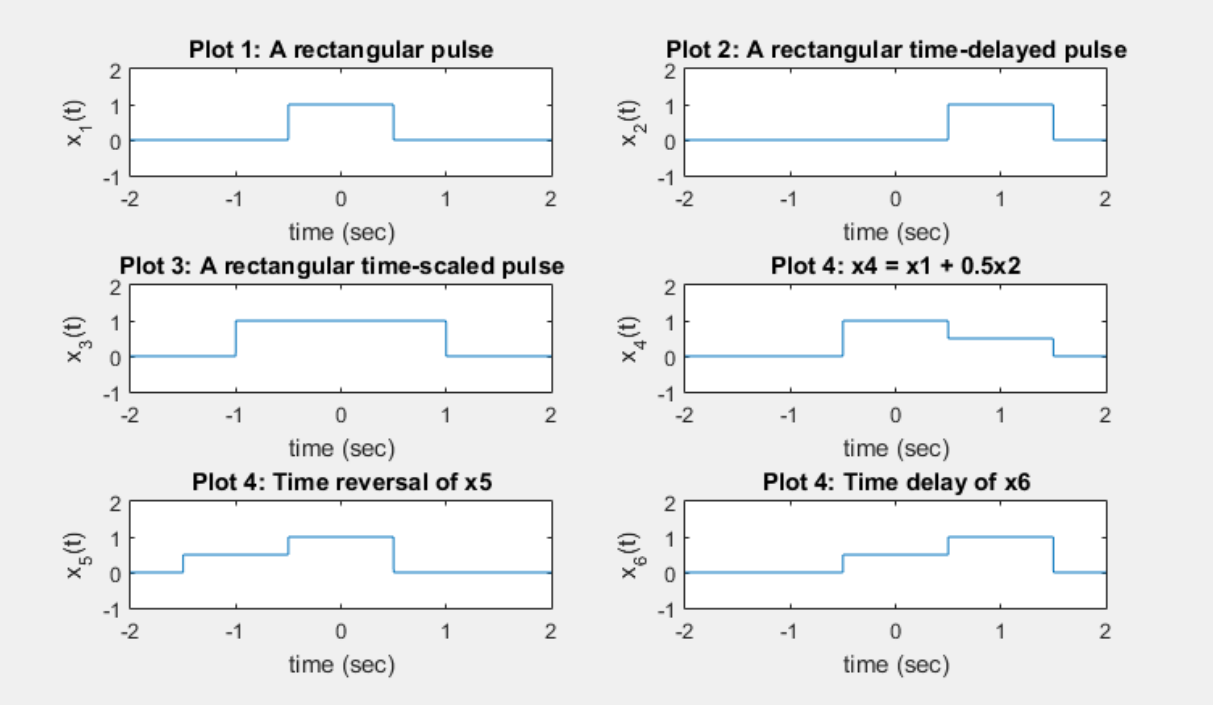


```
% Now let's try to make the time-scaled signal x3(t) = rect(t/2)
fs = 1000;
Ts = 1/fs;
t = -5:Ts:5;
x3 = rect(t/2);

plot(t,x3);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_3(t)');
title('Plot 3: A rectangular time-scaled pulse');
```



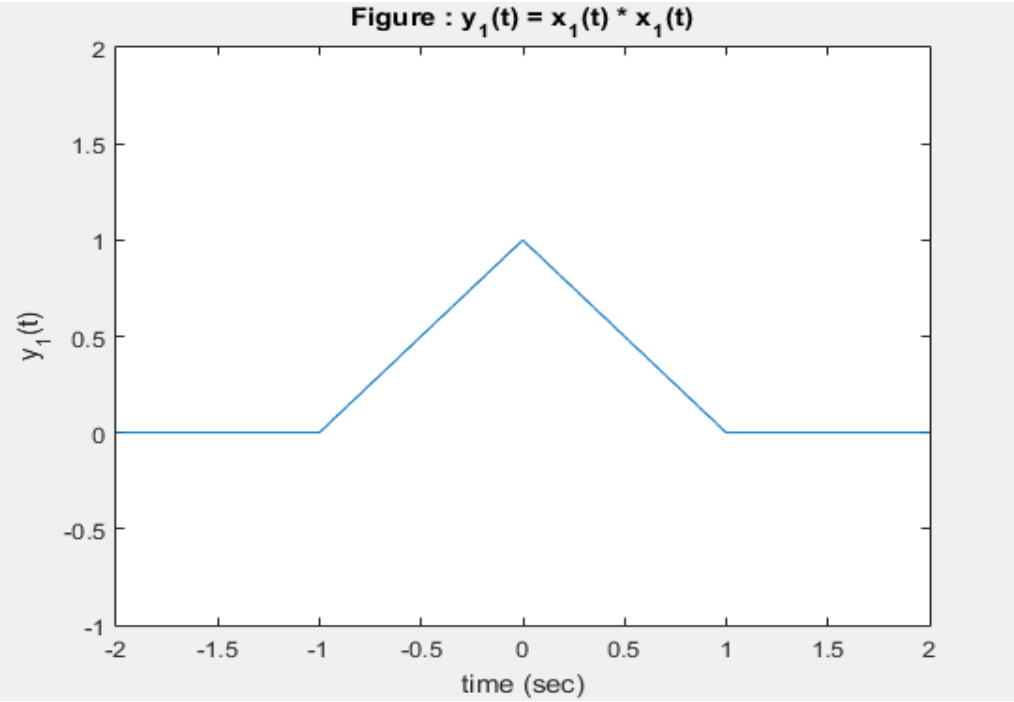
```
%Also create the following signal x4(t) = rect(t)+(1/2)rect(t-1)
%and perform a time reversal as: x5(t) = x4(-t) = rect(-
t)+(1/2)rect(-t-1).
%Finally you can also create the signal: x6(t) = x4(1-t) =
rect(1-t)+(1/2)rect(-t).
% use the MATLAB subplot command to plot all these signals in a
single plot

fs = 1000;
Ts = 1/fs;
t = -5:Ts:5;

% Rectangular pulse
x1 = rect(t);
subplot(3,2,1)
plot(t,x1);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_1(t)');
title('Plot 1: A rectangular pulse');

% A rectangular time-delayed pulse
x2 = rect(t-1);
subplot(3,2,2)
plot(t,x2);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_2(t)');
title('Plot 2: A rectangular time-delayed pulse');

% A rectangular time-scaled pulse
x3 = rect(t/2);
subplot(3,2,3)
plot(t,x3);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_3(t)');
title('Plot 3: A rectangular time-scaled pulse');

% x4(t) addition of x1 and half of x2
x4 = x1 + 0.5 * x2;
subplot(3,2,4);
plot(t,x4);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_4(t)');
title('Plot 4: x4 = x1 + 0.5x2');
```

```
% x5(t) time reversal of x4(t)
x5 = rect(-t) + 0.5 * rect(-t-1);
subplot(3,2,5);
plot(t,x5);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_5(t)');
title('Plot 4: Time reversal of x5');

% x6(t) time delay of x5(t)
x6 = rect(1-t) + 0.5 * rect(-t);
subplot(3,2,6);
plot(t,x6);
axis([-2 2 -1 2])
xlabel('time (sec)');
ylabel('x_6(t)');
title('Plot 4: Time delay of x6');
```



Convolution

```
%Convolution
fs = 1000;
Ts = 1/fs;
t = -5:Ts:5;
x1 = rect(t);

close all;

y = conv(x1, x1);

length(y)
length(t)
%we need to create a separate time axis for the signal y as
ty = -10:Ts:10;
y1 = Ts * conv(x1, x1);

plot(ty, y1);
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('y_1(t)');
title('Figure : y_1(t) = x_1(t) * x_1(t)');
```

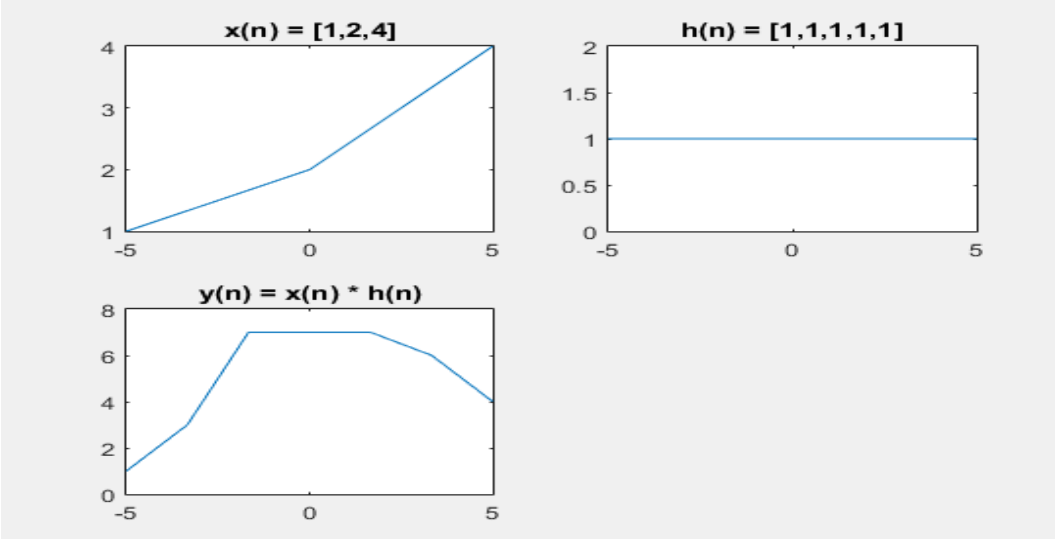


Exercise

Perform convolution on discrete time signals x(n) and h(n), i.e., y(n) = x(n)*h(n) using MATLAB. For each set of signals, plot x(n), h(n) and y(n) as subplots in the same figure.

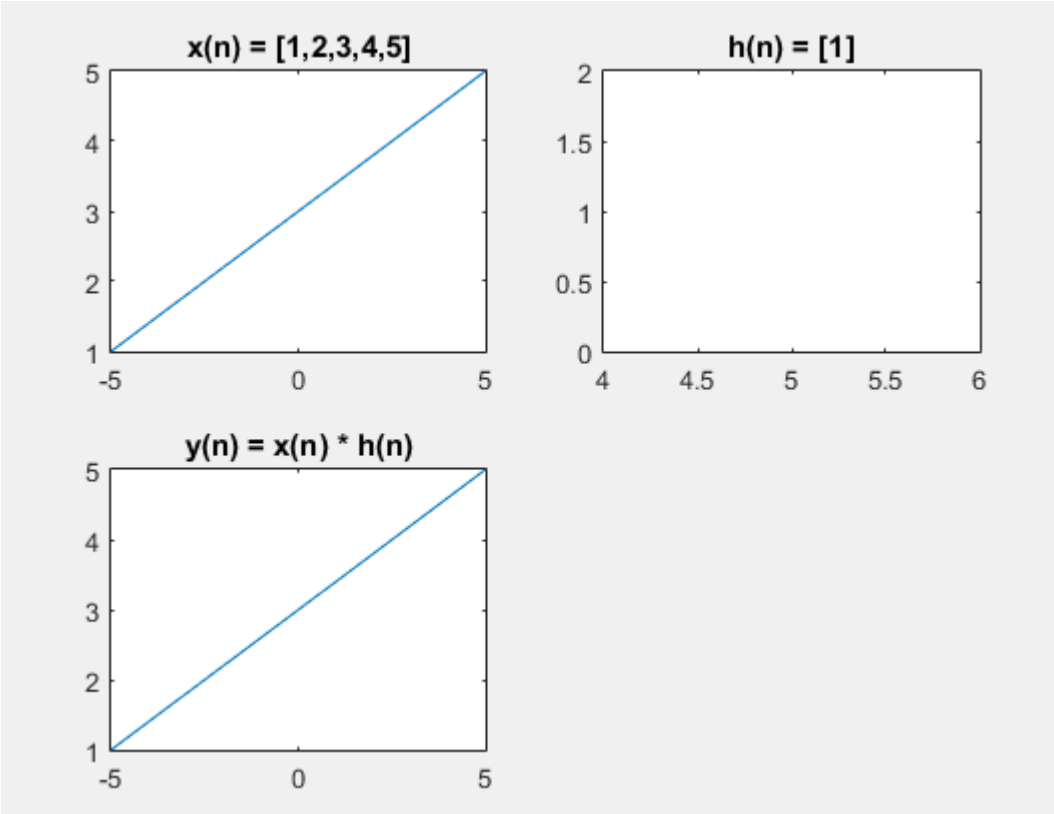
i . x(n) = { 1,2,4 }, h(n) = {1,1,1,1,1}
ii . x(n) = { 1,2,3,4,5 }, h(n) = {1}
iii . x(n) = h(n) ={ 1,2,0,2,1}

```
i)
x1 = [1, 2, 4];
h1 = [1, 1, 1, 1, 1];
y1 = conv(x1, h1);
t1 = linspace(-5,5,3);
t2 = linspace(-5,5,5);
t3 = linspace(-5,5,7);

subplot(2,2,1);
plot(t1,x1);
title('x(n) = [1,2,4]');

subplot(2,2,2);
plot(t2,h1);
title('h(n) = [1,1,1,1,1]');
subplot(2,2,3);
plot(t3,y1);
title('y(n) = x(n) * h(n)');
```

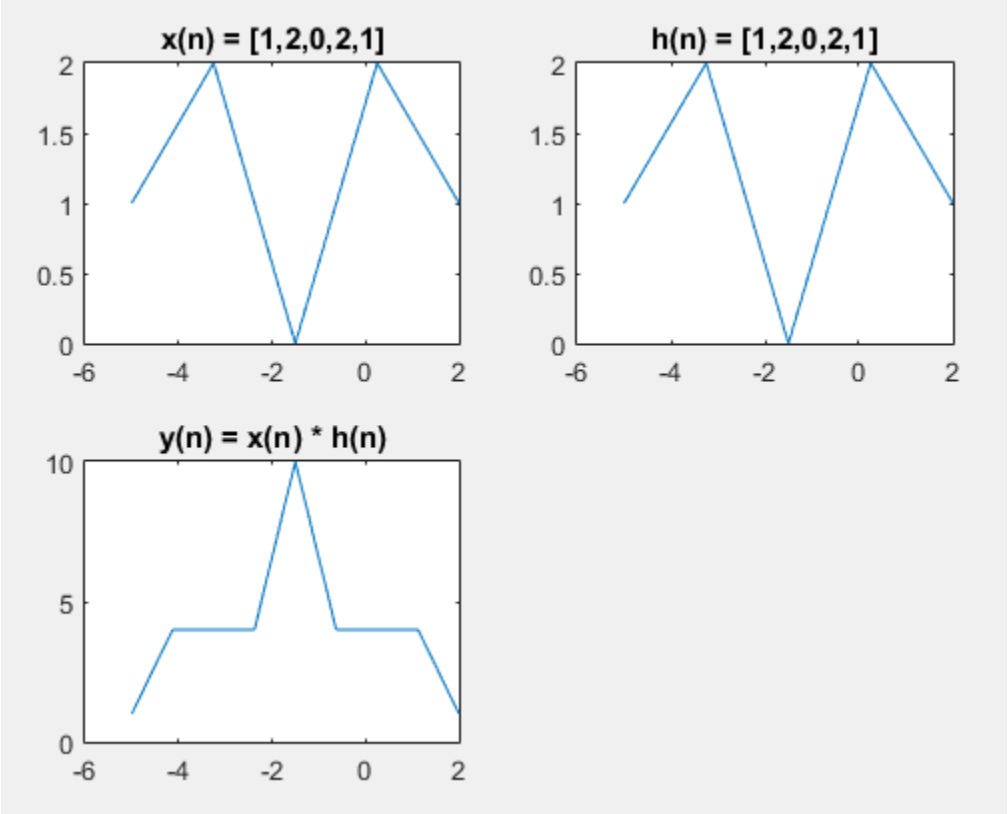


ii)

```
x2 = [1, 2, 3, 4, 5];
h2 = 1;
y2 = conv(x2, h2);
t1 = linspace(-5,5,5);
t2 = linspace(-5,5,1);
t3 = linspace(-5,5,5);

subplot(2,2,1);
plot(t1,x2);
title('x(n) = [1,2,3,4,5]');

subplot(2,2,2);
plot(t2,h2);
title('h(n) = [1]');

subplot(2,2,3);
plot(t3,y2);
title('y(n) = x(n) * h(n)');
```



```
iii)
x3 = [1, 2, 0, 2, 1];
h3 = [1, 2, 0, 2, 1];
y3 = conv(x3, h3);
t1 = linspace(-5,2,5);
t2 = linspace(-5,2,5);
t3 = linspace(-5,2,9);

subplot(2,2,1);
plot(t1,x3);
title('x(n) = [1,2,0,2,1]');

subplot(2,2,2);
plot(t2,h3);
title('h(n) = [1,2,0,2,1]');

subplot(2,2,3);
plot(t3,y3);
title('y(n) = x(n) * h(n)');
```



2. Assume a system with the following impulse response:

h(n) = (0.5)n for 0<= n < 4
= 0 elsewhere

Determine the input x (n) that will generate the output sequence y(n) = {1, 2, 2.5, 3, 3, 3, 2, 1,0...}. Plot h(n), y (n) and x (n) in one figure.

```
t1 = linspace(0,5,5);
h1 = h(t1);
y1 = [1, 2, 2.5, 3, 3, 3, 2, 1, 0];
x1 = deconv(y1, h1);

t2 = linspace(0,5,9);
t3 = linspace(0,5,5);

subplot(2,2,1);
plot(t1,h1);
title('h(n)');
subplot(2,2,2);
plot(t2,y1);
title('y(n)');
subplot(2,2,3);
plot(t3,x1);
title('x(n) is deconvolution of y1,h1');
```