EE387 – BASIC SIGNAL REPRESENTATION AND CONVOLUTION IN MATLAB

IRFAN M.M.M.
E/15/138
SEMESTER 06
28/03/2020

EE 387: Signal Processing

Lab 1 : Basic Signal Representation and Convolution in MATLAB
Reg No : E/15/138
Name : M.M.M Irfan

1. 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.

Unit step Function (utsep.m)

```
function y = ustep(t,ad)
% t: length of time
% ad: advance (positive), delay (negative) factor% Write your code

% shorthand conditional statment which gives 1 or 0 based on the
condition
% this conditional statment is evaluated for each element in t
 y = (t+ad) >= 0;
endfunction
```

Ramp Function (ramp.m)

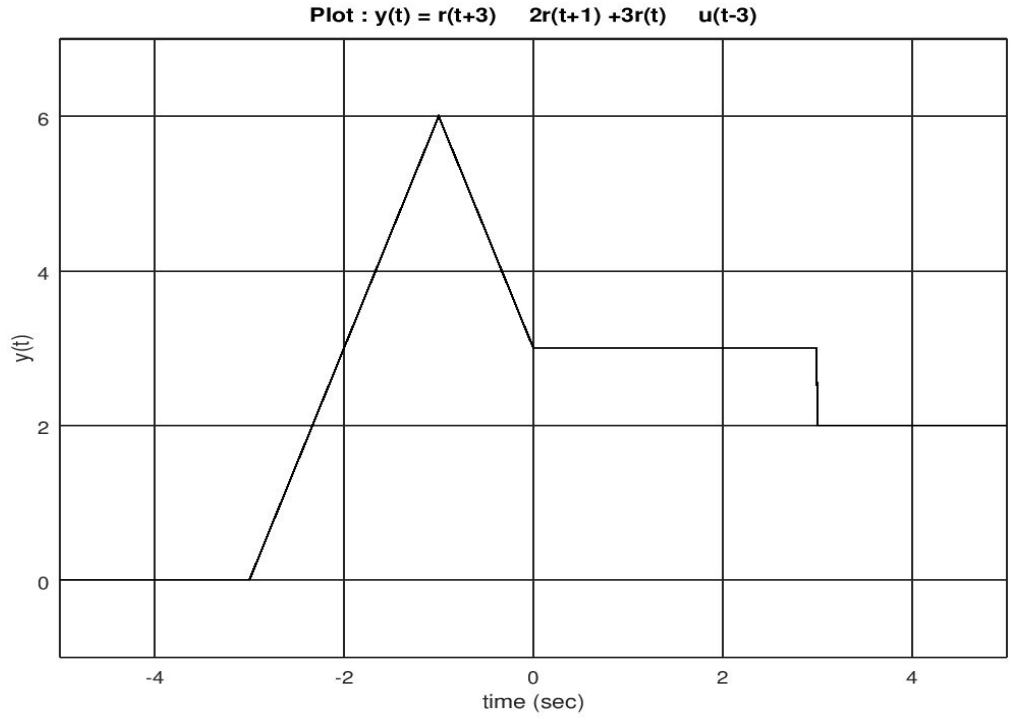
```
function y = ramp(t,m,ad)
% t: length of time
% m: slope of the ramp function
% ad: advance (positive), delay (negative) factor

% generate the required ramp with the slope and the starting point
 y1 = m*(t+ad);

% multiply the generated function with the unit step of the same t,ad
% so that the negative part will be eliminated

 y = y1.*ustep(t,ad);
endfunction
```

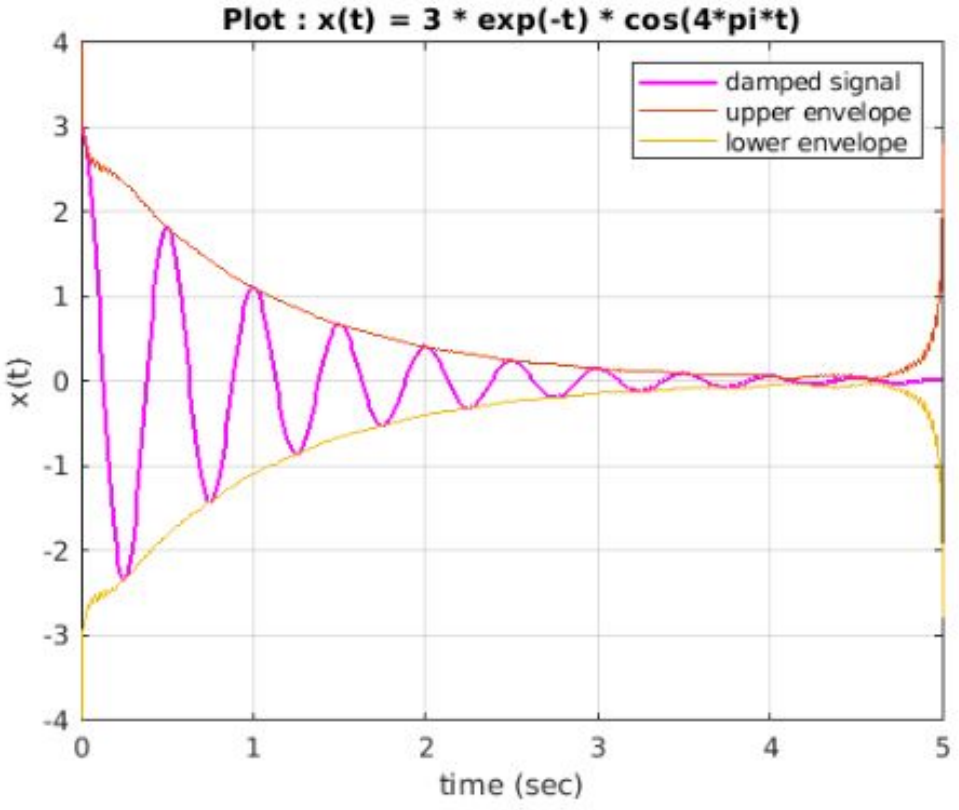
Complete Script for Exercise 1 (code snippet 1)

```
clear all;
Ts=0.01;
t= -5:Ts:5;
r = 3;
y1 = ramp(t,r,3); % y = r(t+3)
```

```
y2 = ramp(t,-2*r,1); % y = -2*r(t+1)
y3 = ramp(t,r,0); % y = 3rt
y4 = ustep(t,-3); % y = u(t-3)

% y(t) = r(t+3) - 2r(t+1) +3r(t) - u(t-3)
y = y1+y2+y3-y4;

% plot the graph and set grid, axis, labels,and the title
plot(t,y,'k');
axis([-5 5 -1 7]);
grid on;

xlabel( 'time (sec)' ) ;
ylabel( 'y(t)' ) ;
title ('Plot : y(t) = r(t+3) - 2r(t+1) +3r(t) - u(t-3)');

% plot the damped sinusoidal grpah in another window
figure;
Ts=0.01;
t= 0:Ts:5;
% construct the damped signal
xt = 3*exp(-t).*cos(4*pi*t);
plot(t,xt,'m','linewidth',1.5);
grid on;
xlabel( 'time (sec)' ) ;
ylabel( 'x(t)' ) ;
title ('Plot : x(t) = 3 * exp(-t) * cos(4*pi*t)');

% get the upper and lower envelopes
[up,lo] = envelope(xt);

% plot on the same graph
hold on;
plot(t,up,t,lo)
legend('damped signal','upper envelope','lower envelope')
hold off;
```

Plots obtained by the above code (code snippet 1)





PART 2: Time-Domain Convolution

Creating a rectangular pulse in MATLAB

Rectangular pulse function (rect.m)

```
function xt = rect(t)

 % xt = 1 when (t < 0.5) and (t>=-0.5)
 % elsewhere 0
 % shorthand conditional statment which gives 1 or 0 based on the
condition
 % this conditional statment is evaluated for each element in t
 xt = (t < 0.5) & (t>=-0.5);
 endfunction
```

Script to plot the rectangular pulse and elementary operations on it (code snippet 2)

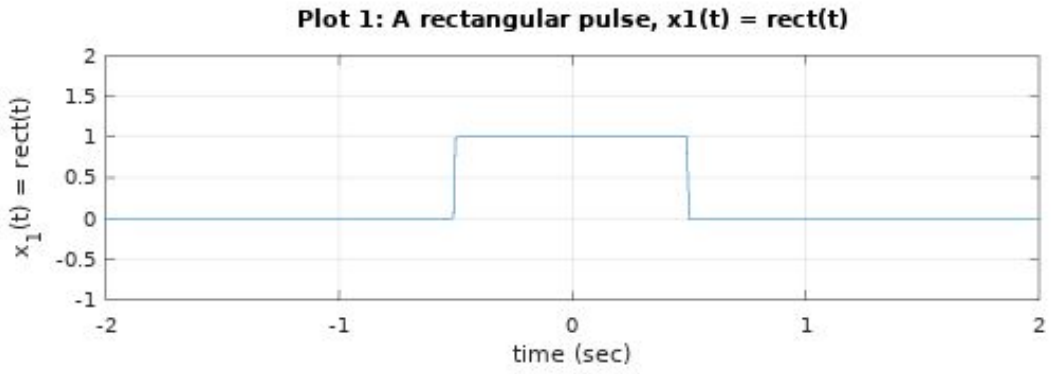
```
% sampling frequency
f_s = 100;

% sampling period
T_s = 1/f_s;

t = [-5:T_s:5];

% generate rectangle wave
x1 = rect(t);
figure;
plot(t,x1);
grid on;
axis( [-2 2 -1 2]);

xlabel( 'time (sec)' ) ;
ylabel( 'x_1(t)' ) ;
title ('Plot 1: A rectangular pulse');


% First let's create and plot the time- delayed signal, x 2 (t) =
rect(t-1)
x2 = rect(t-1);
figure;
plot(t,x2);
axis( [-2 2 -1 2]);
```

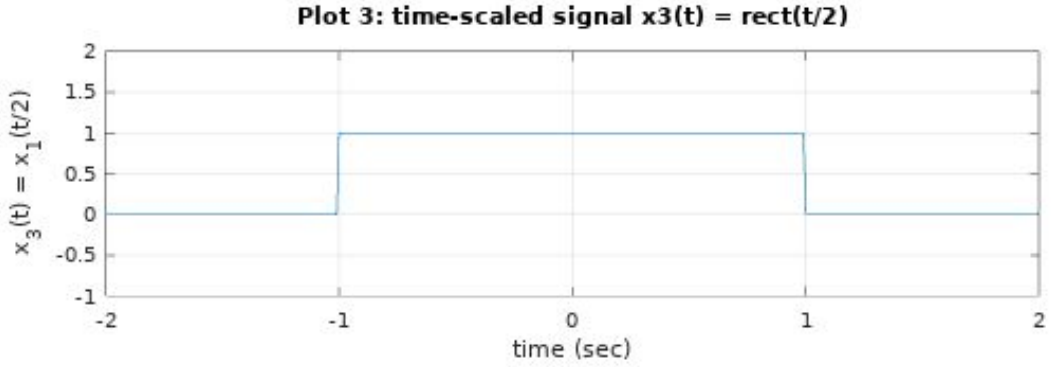
```
% Now let's try to make the time-scaled signal x 3 (t) = rect(t/2)
x3 = rect(t/2);
figure;
plot(t,x3);
axis( [-2 2 -1 2]);

%create another signal
x4 = rect(t)+(0.5*rect(t-1));

%create time reversal of x4
x5 = rect(-t)+(0.5*rect(-t-1));

%create another signal
x6 = rect(1-t)+(0.5*rect(-t));

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

subplot(3,2,2);
plot(t,x2);
grid on;
axis([-2 2 -1 2]);
xlabel( 'time (sec)' );
ylabel('x_2(t) = x_1(t-1)');
title('Plot 2: time- delayed signal, x2(t) = rect(t-1)');

subplot(3,2,3);
plot(t,x3);
grid on;
axis([-2 2 -1 2]);
xlabel( 'time (sec)' );
ylabel('x_3(t) = x_1(t/2)');
title('Plot 3: time-scaled signal x3(t) = rect(t/2)');
```

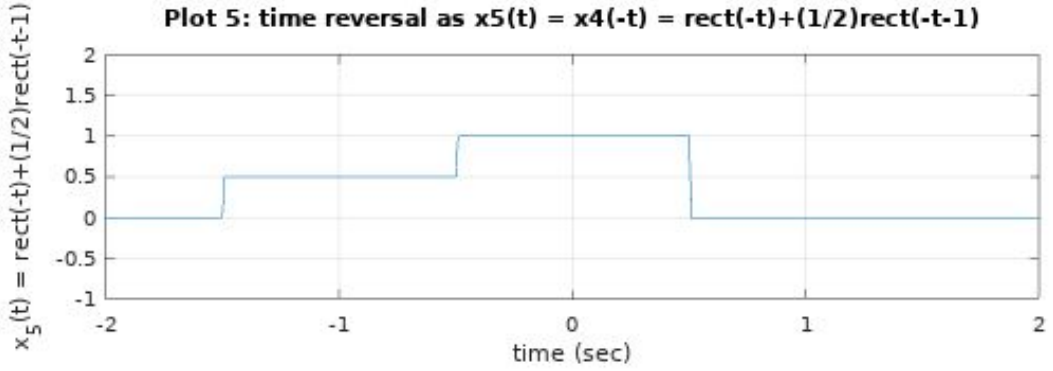
```
subplot(3,2,4);
plot(t,x4);
grid on;
axis([-2 2 -1 2]);
xlabel( 'time (sec)' );
ylabel('x_4(t) = rect(t)+(1/2)rect(t-1)');
title('Plot 4: x4(t) = rect(t)+(1/2)rect(t-1)');


subplot(3,2,5);
plot(t,x5);
grid on;
axis([-2 2 -1 2]);
xlabel( 'time (sec)' );
ylabel('x_5(t) = rect(-t)+(1/2)rect(-t-1)');
title('Plot 5: time reversal as x5(t) = x4(-t) =
rect(-t)+(1/2)rect(-t-1)');



subplot(3,2,6);
plot(t,x6);
grid on;
axis([-2 2 -1 2]);
xlabel( 'time (sec)' );
ylabel('x_6(t) = x_1(t/2)');
title('Plot 6: x6(t) = x4(1-t) = rect(1-t)+(1/2)rect(-t)');
```

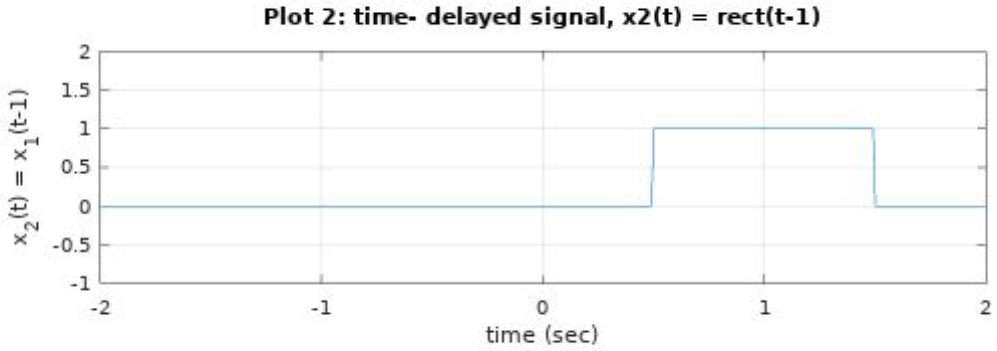
Plots obtained by the above code (code snippet 2)

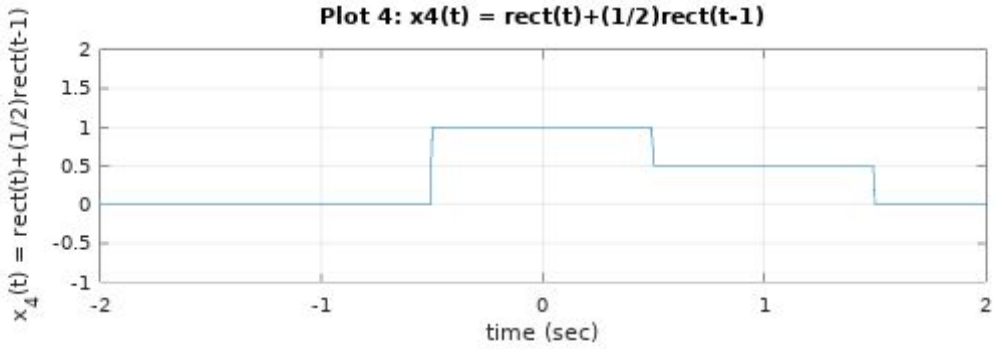
Note: Actual plot had 3 rows and 2 columns since it not very clear, the image has been split into two images

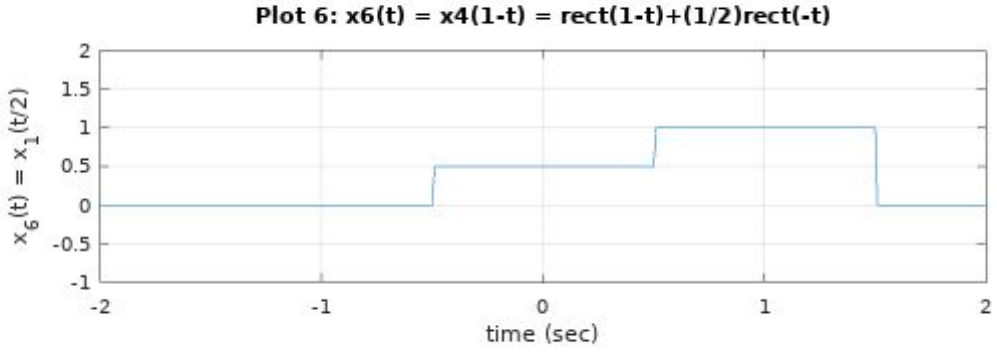












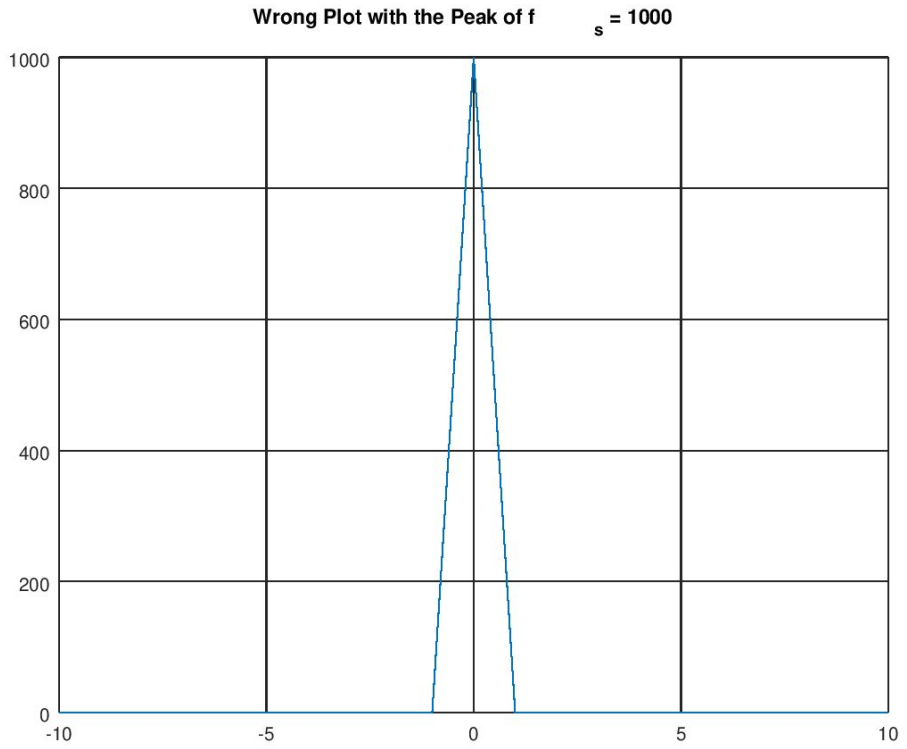
Convolution

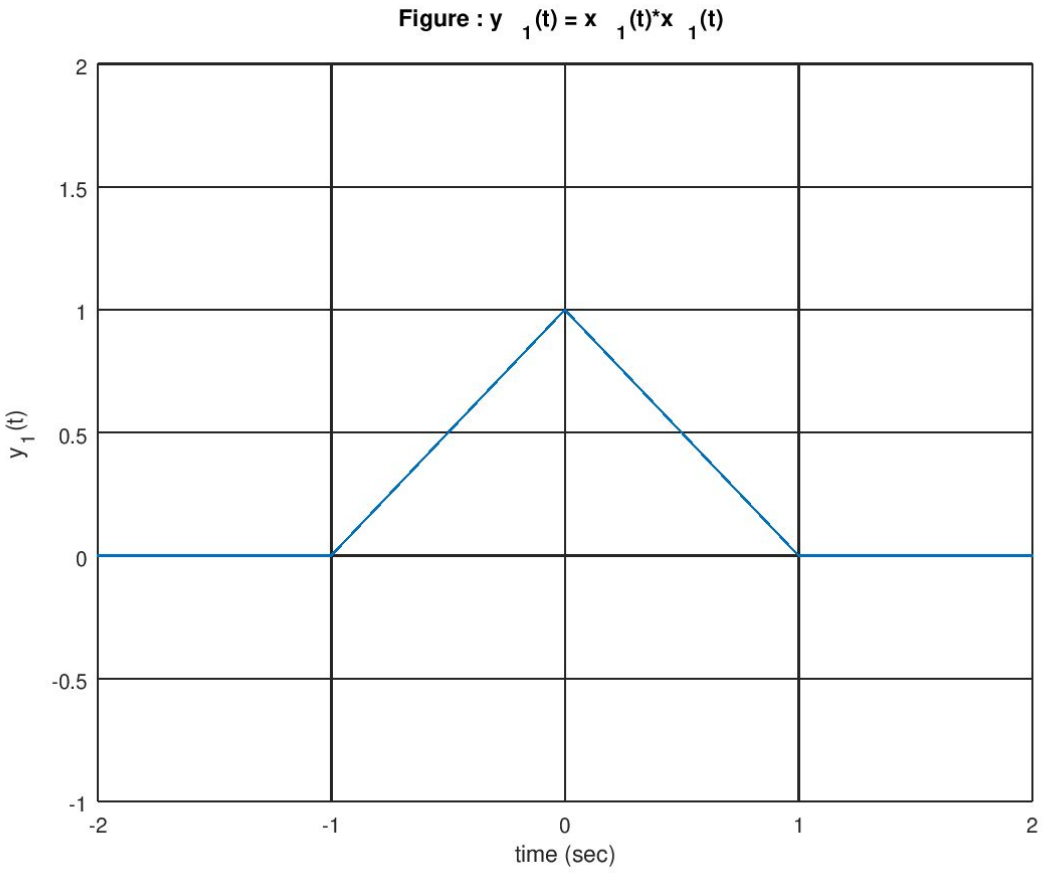
Script for the convolution exercise (code snippet 3)

```
clear all;
f_s = 1000; % sampling frequency 1000 hz
T_s = 1/f_s;
t = -5:T_s:5;

% generate rectangle wave
x1 = rect(t);

y = conv(x1,x1); % get the convolution
```

```
%plot(t,y); will give an error
length(y); % 2001
length(t); % 1001
% so it satisfies, length(y) = 2*length(t)-1
% separate time axis for the signal y as:
t_y = -10:T_s:10;

figure;
plot(t_y, y);
title("Wrong Plot with the Peak of f_s = 1000");
grid on;
y1 = T_s*conv(x1,x1);
figure;
plot(t_y, y1);
grid on;
axis( [-2 2 -1 2] ) ;
xlabel( 'time (sec)');
ylabel('y_1(t)');
title('Figure : y_1(t) = x_1(t)*x_1(t)');
```

Plots obtained by the above code (code snippet 3)





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

For the above purpose, I ‘ve added a custom function which takes x(n), h(n) as inputs and plot the corresponding graphs.

The custom function (custom_plot_convolution.m)

```
function custom_plot_convolution (xn, hn)

% will plot grpahs in a new window
figure;

% get n values equally from -6 to 6
n_1 = linspace(-6,6,length(xn));

% plot in the 1st index in the 3 row 1 column subplot
subplot(3,1,1);
```

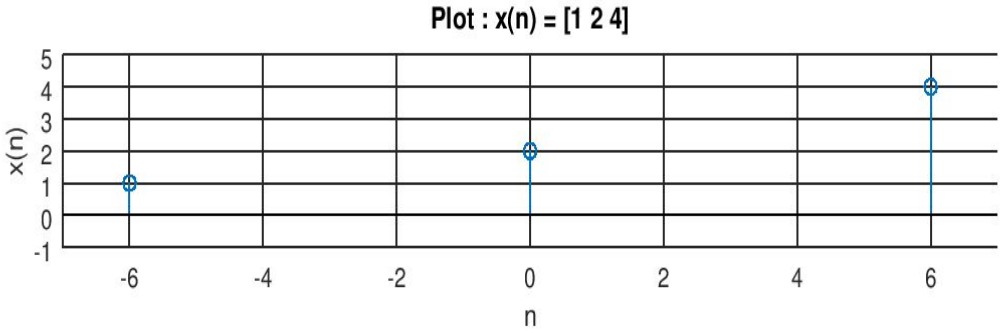
```
% used stem function since its a discrete time signal
stem(n_1, xn);

% set the axis and lables and etc..
grid on;
axis( [-7 7 -1 5] ) ;
xlabel('n');
ylabel('x(n)');
title(['Plot : x(n) = ' mat2str(xn)'']) 

% similary plot h(n)
n_2 = linspace(-6,6,length(hn));
subplot(3,1,2);
stem(n_2, hn);
grid on;
axis( [-7 7 -1 5] ) ;
xlabel('n');
ylabel('h(n)');
title(['Plot : h(n) = ' mat2str(hn)'']) 

% get the convolution of x(n) and h(n)
yn = conv(xn,hn);

% plot y(n)
n_3 = linspace(-6,6,length(yn));
subplot(3,1,3);
stem(n_3, yn);
grid on;
axis( [-7 7 -1 9] ) ;
xlabel('n');
ylabel('y(n)');
title('Plot : y(n) = x(n)*h(n)');

endfunction
```

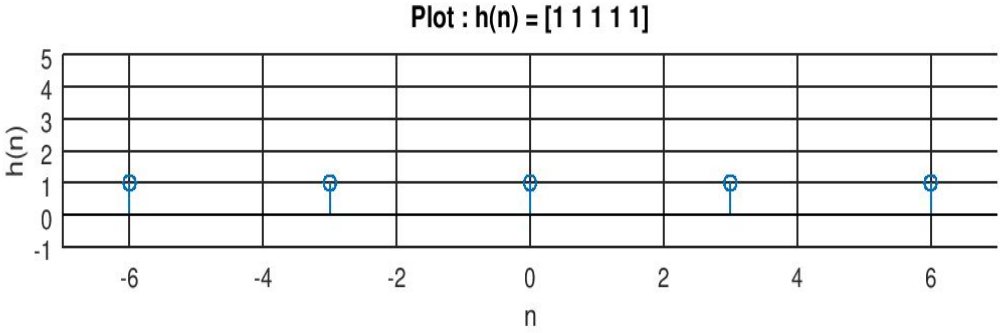
Script to invoke the function for various x(n), h(n) combinations (code snippet 4)

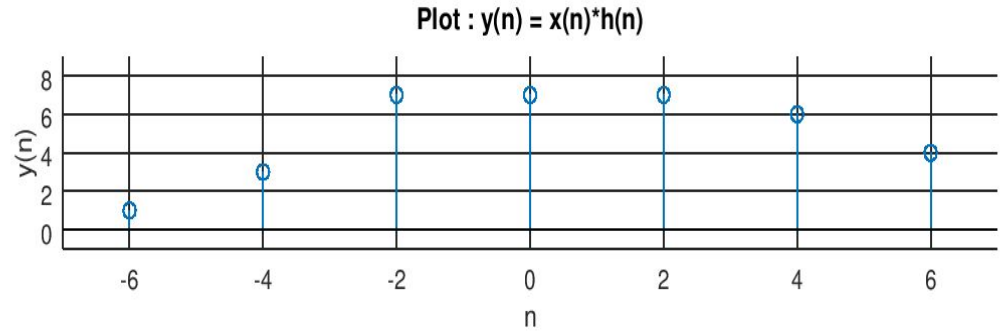
```
% define x(n), h(n)
xn = [1,2,4];
hn = [1,1,1,1,1];

% plot the corrsponding plots
custom_plot_convolution(xn,hn);
```

```
xn = [1,2,3,4,5];
hn = [1];

custom_plot_convolution(xn,hn);

xn = hn = [1,2,0,2,1];

custom_plot_convolution(xn,hn);
```

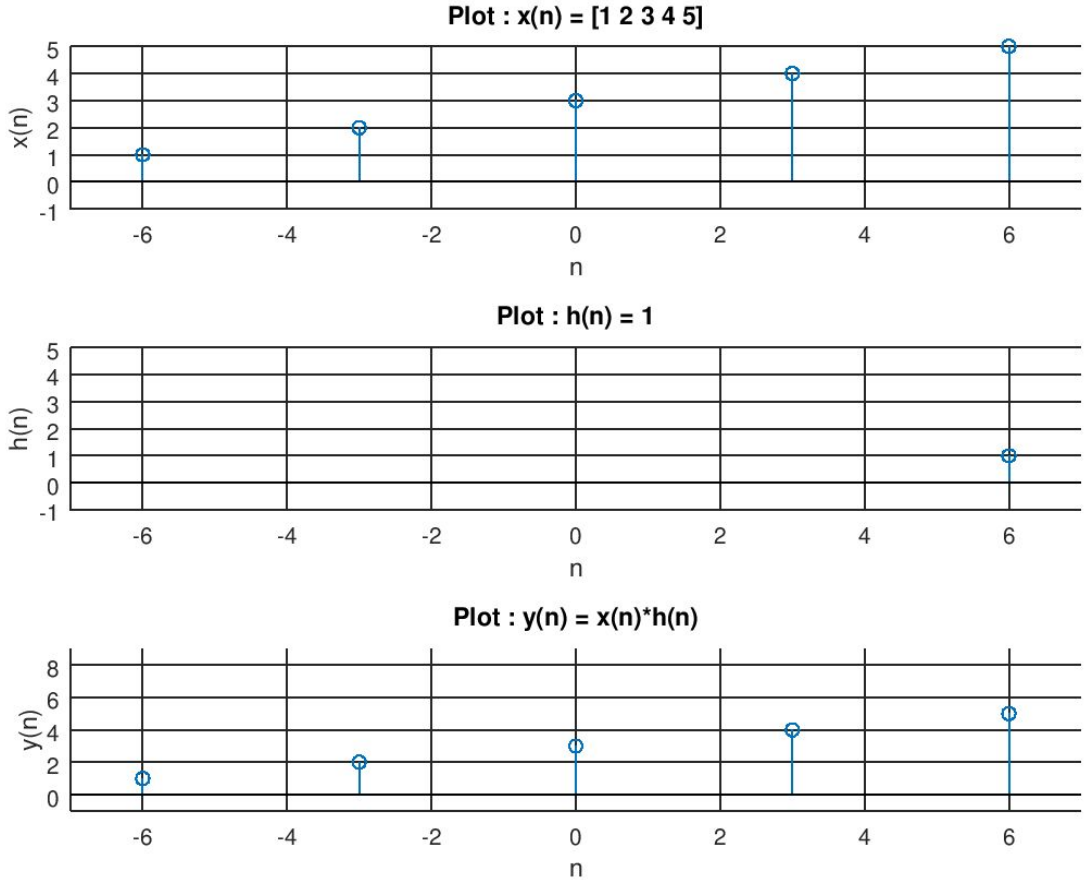
Plots obtained by the above code (code snippet 4)



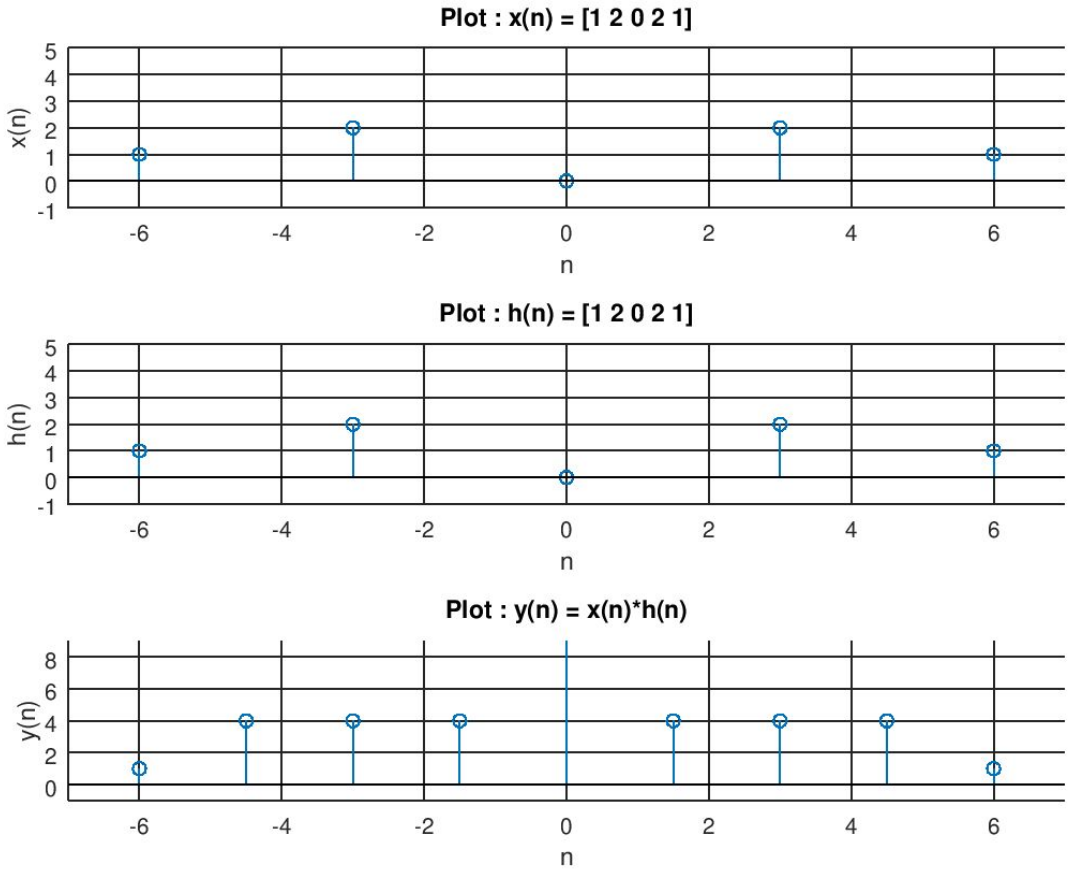




Plots for x(n) = { 1,2,4 }, h(n) = {1,1,1,1,1}



Plots for x(n) = { 1,2,3,4,5 }, h(n) = {1}



Plots for x(n) = h(n) ={ 1,2,0,2,1}

2. Assume a system with the following impulse response:

$$h(n) = (0.5)^n \quad \text{for } 0 \le n < 4$$
$$= 0 \quad \text{elsewhere}$$

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.

Script to find out x(n) (code snippet 5)

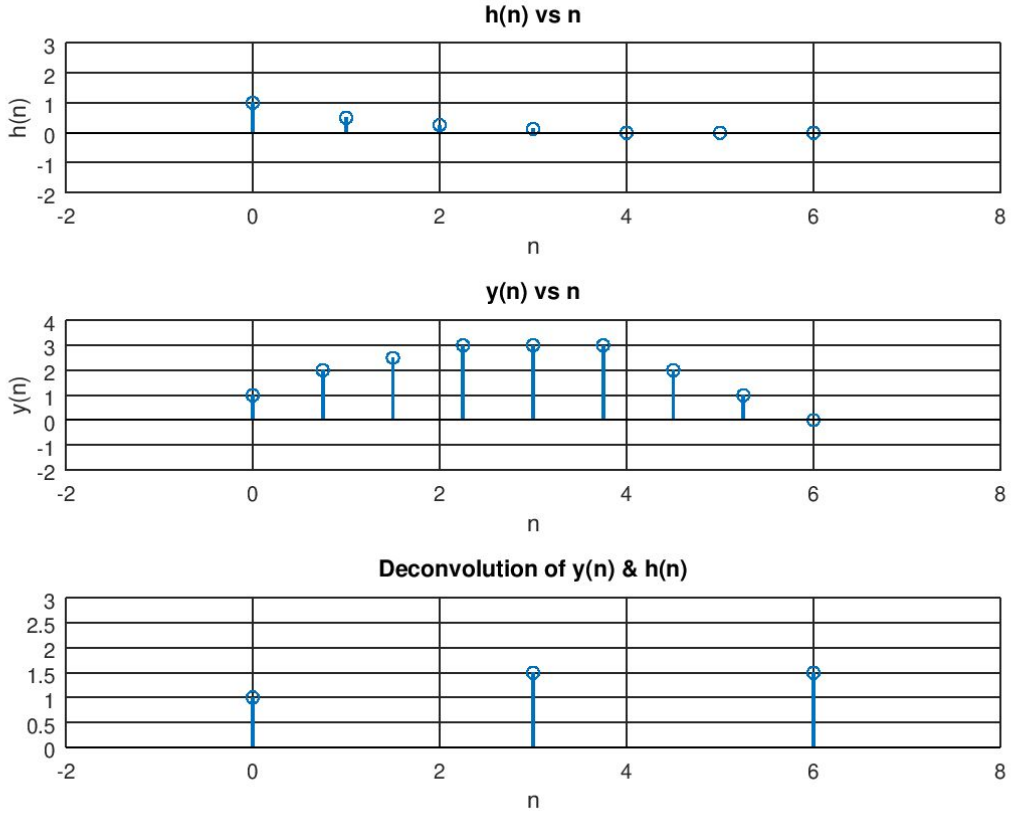
```
clear all;
l = 7;
n = linspace(0,6,l);
% code to generate the function h(n)
for k = 1 : l
  if(n(k)>=0 && n(k)<4)
     hn(k) = 0.5^n(k);
  else
```

```
    hn(k) = 0;
  endif
endfor

subplot(3,1,1);
% used stem since its a discrete time signal
stem(n,hn,'linewidth',1.3);
% set grid, labels,title and axis as usual
grid on;
axis([-2 8 -2 3]);
title('h(n) vs n');
xlabel('n');
ylabel('h(n)');

% initialize the yn function given
yn = [1,2,2.5,3,3,3,2,1,0];

n_1 = linspace(0,6,length(yn));
subplot(3,1,2);

% plot y(n)
stem(n_1,yn,'linewidth',1.3);
grid on;
axis([-2 8 -2 4]);
title('y(n) vs n');
xlabel('n');
ylabel('y(n)');

% obtain the deconvolution for y(n) and h(n)
xn = deconv(yn,hn);
n_2 = linspace(0,6,length(xn));
subplot(3,1,3);
stem(n_2,xn,'linewidth',1.3);
grid on;
axis([-2 8 0 3]);
title('Deconvolution of y(n) & h(n)');
xlabel('n');

% input xn that will generate the output sequence y(n) = {1, 2, 2.5, 3,
3, 3, 2, 1,0...}.
display(xn);
```

Output of the above code

```
xn =
   1.0000   1.5000   1.5000
```

Plots obtained by the above code (code snippet 5)



Remarks

The complete code for the lab can be found at
https://github.com/irfanm96/EE387-Signal-Processing/tree/master/Lab01