EE 387 – Signal Processing
Lab 1: Basic Signal Representation and Convolution in MATLAB
E/20/420 – WANASINGHE J.K.

Contents

PART 1: Basic Signal Representation in MATLAB

1. Write a Matlab program and necessary functions to generate the following signal:

$$y(t) = r(t+3) - 2r(t+1) + 3r(t) - u(t-3)$$

```
clear all; %To clear all variables

Ts = 0.01; %Sampling time
t = -5:Ts:5; %Time vector where the signal is defined

y1 = ramp(t, 1, 3);    % r(t+3): shift=3, slope=1
y2 = ramp(t, 1, 1);    % r(t+1): shift=1, slope=1
y3 = ramp(t, 1, 0);    % r(t):   shift=0, slope=1
y4 = ustep(t, -3);      % u(t-3): shift=3

y = y1 - 2*y2 + 3*y3 - y4;
plot(t, y, 'r'); axis([-5 5 -1 7]); grid

function y = ramp(t, slope, shift)
    y = slope * max(0, t + shift);
end

function y = ustep(t, shift)
    y = double(t >= shift); % unit step function

    %t>=shift creates a logical array of 1s and 0s
end
```

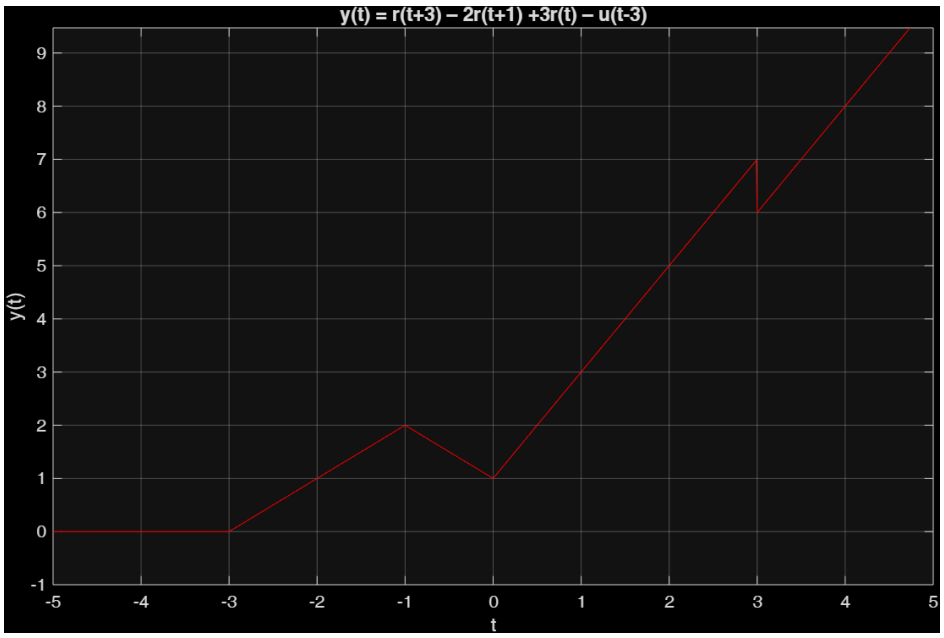
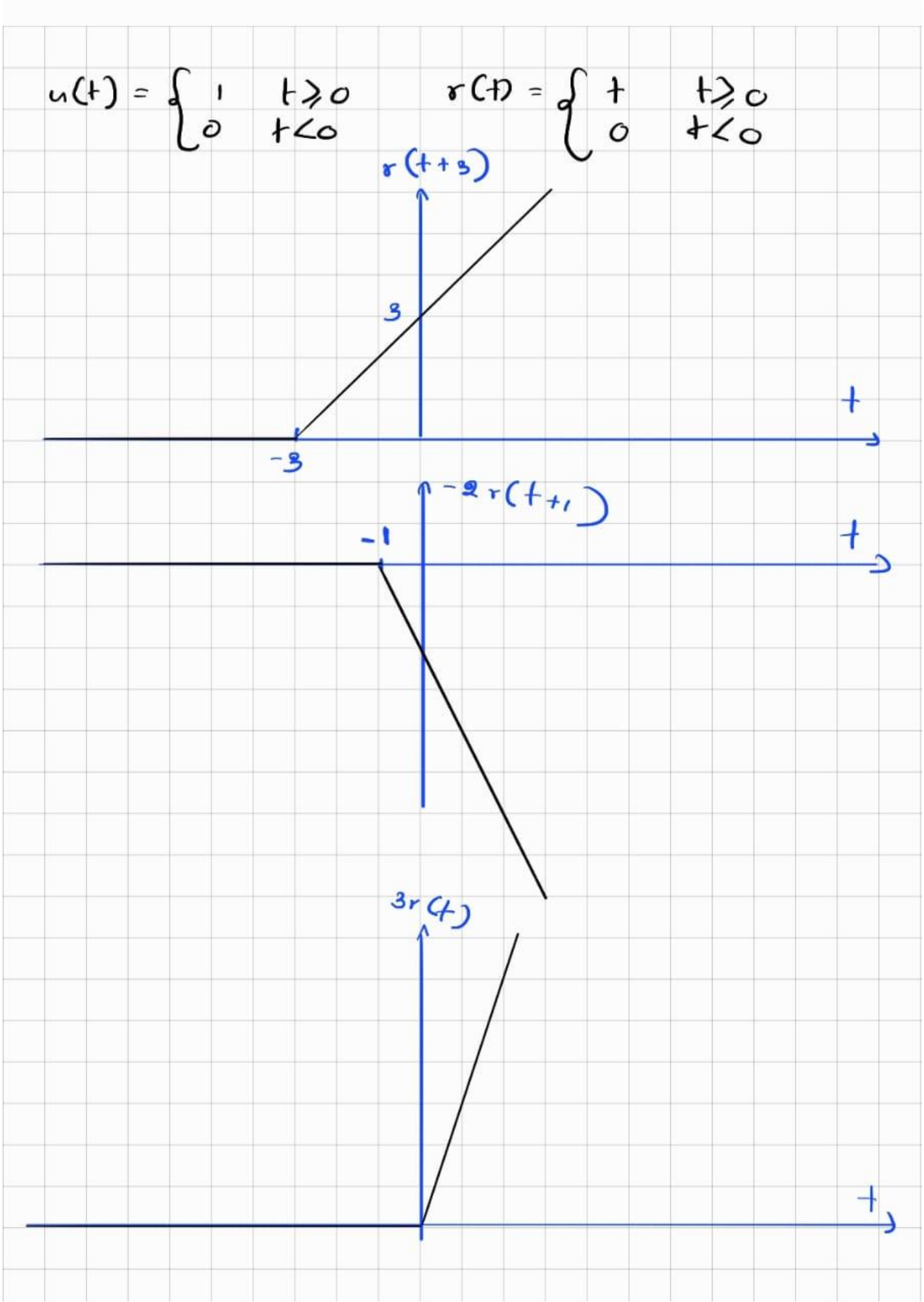


Figure 1: y(t) = r(t+3) – 2r(t+1) +3r(t) – u(t-3)

Analytic verification

$$u(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases} \qquad r(t) = \begin{cases} t & t \geq 0 \\ 0 & t < 0 \end{cases}$$



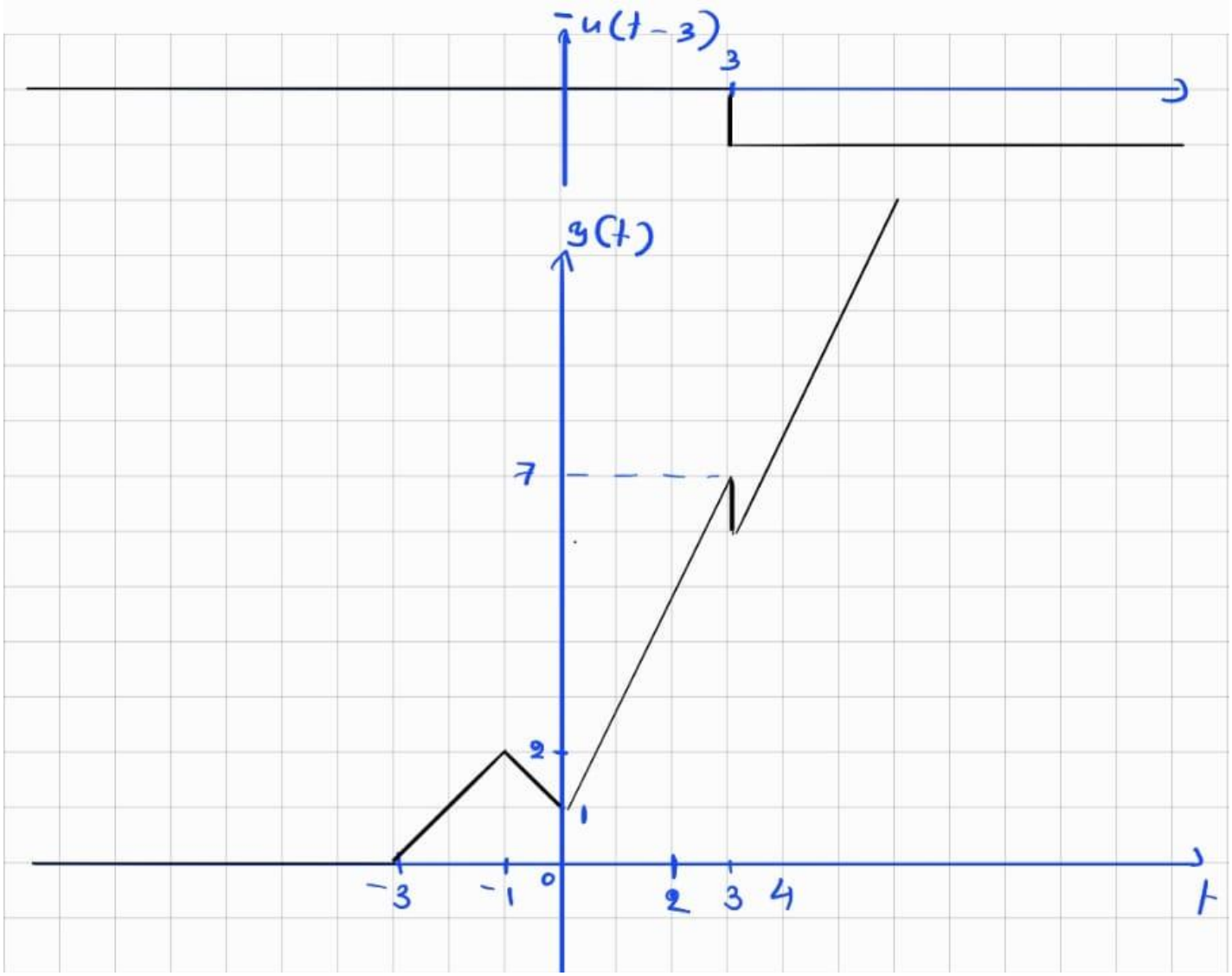


Figure 2: ANALYTIC VERIFICATION

2. For the damped sinusoidal signal $x(t) = 3e^{-t}\cos(4\pi t)$ write a MATLAB program to generate x(t) and its envelope, then plot.

```
clear all; %To clear all variables

Ts = 0.01; %Sampling time
t = -5:Ts:5; %Time vector where the signal is defined
x = 3*exp(-t).*cos(4*pi*t); % x(t) = 3e^-tcos(4*pi*t)

env = 3*exp(-t); % Envelope of the signal

subplot(2,1,1); % First subplot: signal with envelope
plot(t, x, 'r'); hold on;
plot(t, env, 'w--');
plot(t, -env, 'w--');
axis([-3 4 -50 50]);
grid;
xlabel('Time (s)');
ylabel('Amplitude');
```

```
title('Signal and Envelope');
legend('Signal', 'Envelope');

subplot(2,1,2); % Second subplot: envelope only
plot(t, env, 'w-'); hold on;
plot(t, -env, 'w-');
axis([-3 4 -50 50]);
grid;
xlabel('Time (s)');
ylabel('Amplitude');
title('Envelope');
legend('Envelope');
hold off;
```

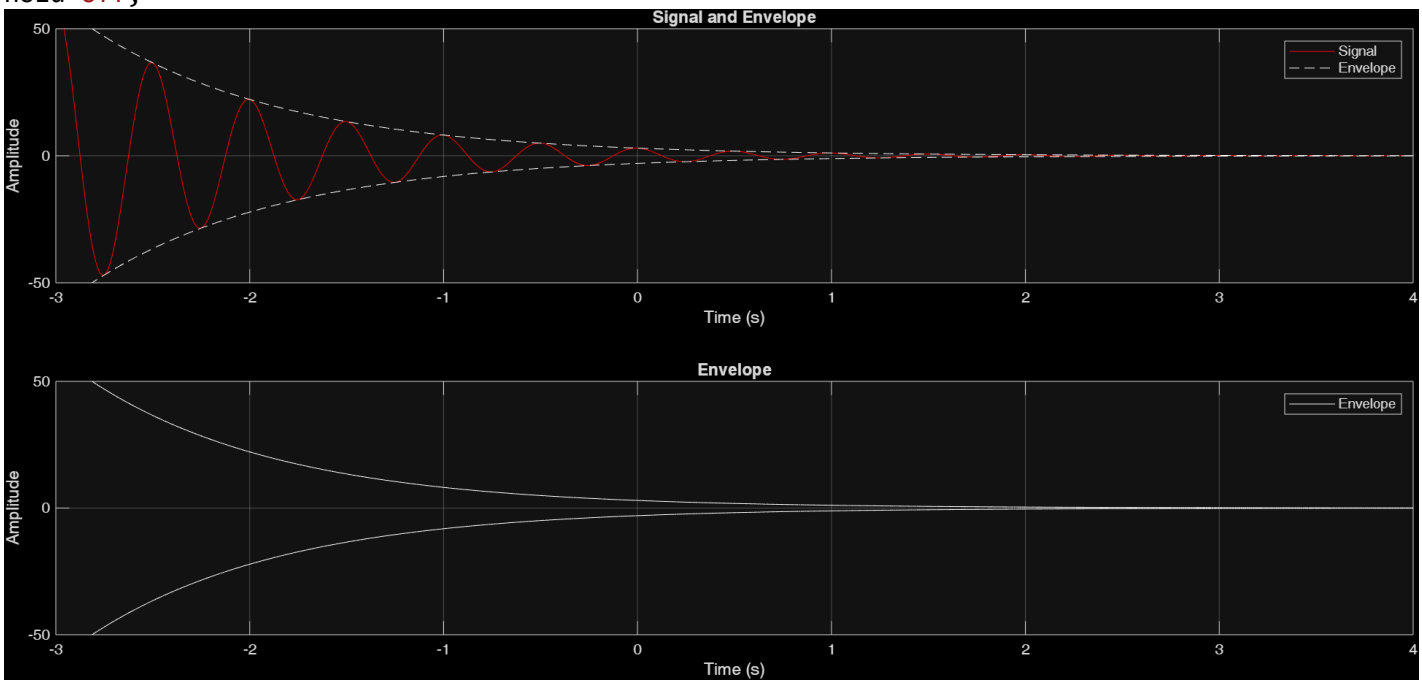


Figure 3: SIGNAL AND ENVELOPE PLOT

PART 2: Time-Domain Convolution

Creating a rectangular pulse in MATLAB

```
clear all;
f_s = 100; %Sampling frequencyt
t_s = 1/f_s;
t = -5:t_s:5; %Time vector where the signal is defined

function x = rect(t)
% RECT rectangular pulse
% Usage: x = rect(t)
% This function takes in a vector t of sample instants and outputs the
% corresponding rectangular pulse contained in the function x
    x = double(t >= -0.5 & t < 0.5);
end

x1 = rect(t);
plot(t,x1,'-w');
axis( [-2 2 -1 2]); %this sets the axis limits of x as [-2 2] and y as [-1 2]
xlabel( 'time (sec)' );
ylabel( 'x_1(t)' ) ;
```

```
title ('Plot 1: A rectangular pulse');
```

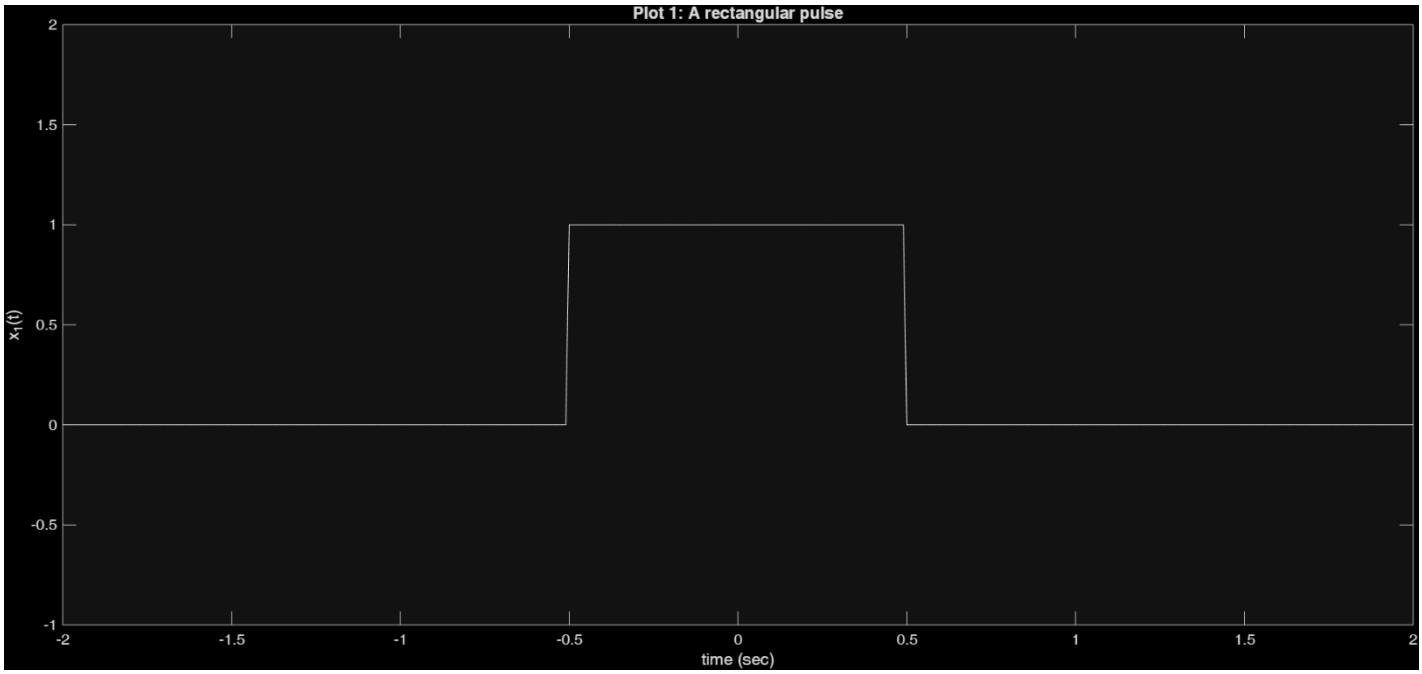


Figure 4: RECTANGULAR PULSE PLOT

Elementary signal operations

Operations, such as time-delay, time-scaling, and time-reversal are performed below using MAT-LAB.

$x_1(t) = rect(t)$ as defined in the above section

$x_2(t) = rect(t-1)$

$x_3(t) = rect(\frac{t}{2})$

$x_4(t) = rect(t) + \frac{1}{2}rect(t-1)$

$x_5(t) = x_4(-t) = rect(-t) + \frac{1}{2}rect(-t-1)$

$x_6(t) = x_4(1-t) = rect(1-t) + \frac{1}{2}rect(-t)$

```
clear all;

f_s = 100; % Sampling frequency
t_s = 1/f_s;
t = -5:t_s:5; % Time vector

function x = rect(t)
    x = double(t >= -0.5 & t < 0.5);
end

% Elementary Signal Operations
x1 = rect(t);
x2 = rect(t-1);
x3 = rect(t/2);
x4 = rect(t) + 0.5*rect(t-1);
x5 = rect(-t) + 0.5*rect(-t-1);      % x5(t) = x4(-t)
x6 = rect(-t+1) + 0.5*rect(-t);      % x6(t) = x4(1-t) = x5(1+t)

figure('Position',[100 100 1000 600]); % Wider figure for better label fit

subplot(3,2,1);
plot(t,x1,'w-');
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('x_1(t) = rect(t)');
set(gca, 'FontSize', 10);

subplot(3,2,2);
plot(t,x2,'w-');
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('x_2(t) = rect(t-1)');
set(gca, 'FontSize', 10);

subplot(3,2,3);
plot(t,x3,'w-');
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('x_3(t) = rect(t/2)');
set(gca, 'FontSize', 10);

subplot(3,2,4);
plot(t,x4,'w-');
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('x_4(t) = rect(t) + 0.5rect(t-1)');
set(gca, 'FontSize', 10);

subplot(3,2,5);
plot(t,x5,'w-');
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('x_5(t) = rect(-t) + 0.5rect(-t-1)');
set(gca, 'FontSize', 10);

subplot(3,2,6);
plot(t,x6,'w-');
axis([-2 2 -1 2]);
xlabel('time (sec)');
ylabel('x_6(t) = rect(-t+1) + 0.5rect(-t)');
set(gca, 'FontSize', 10);
```

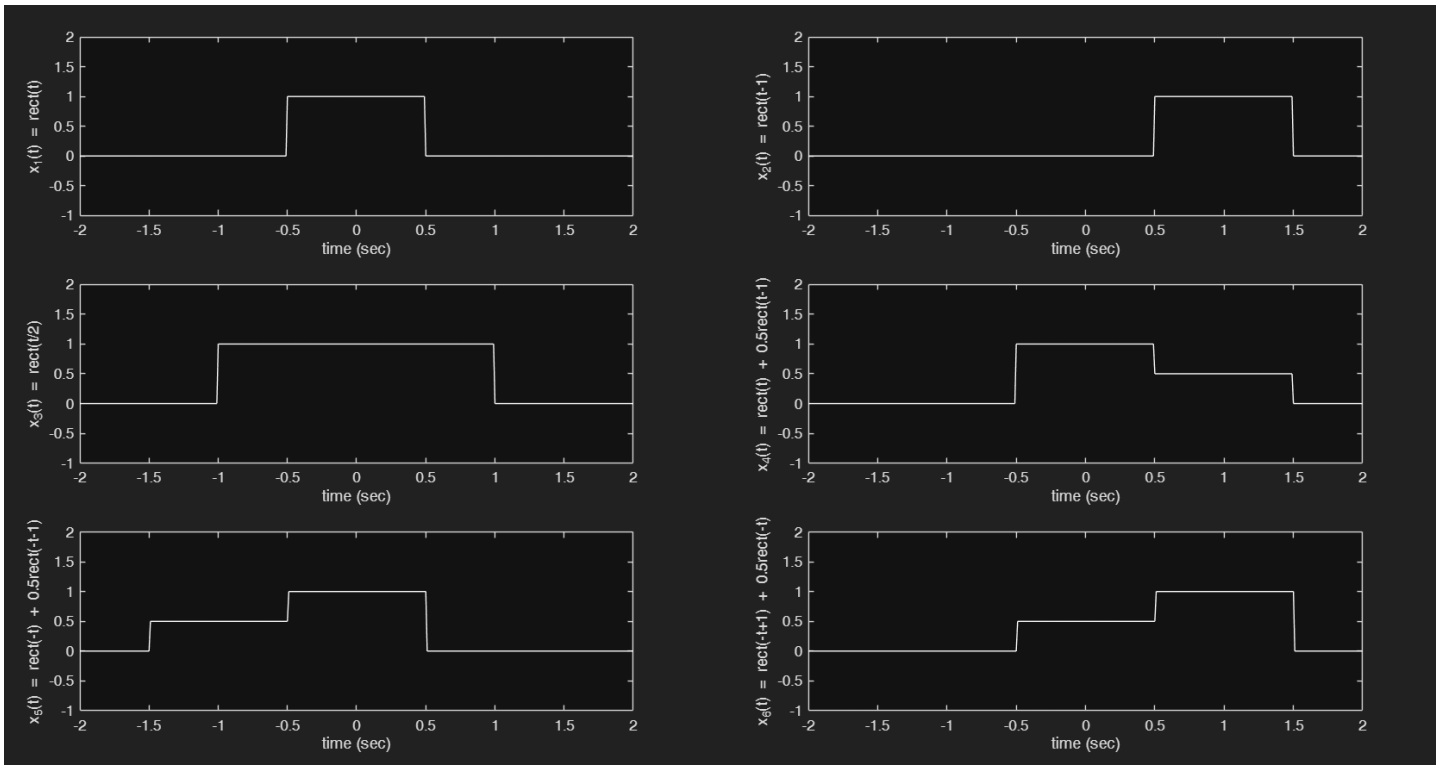


Figure 5: ELEMENTARY SIGNAL OPERATIONS

Convolution

MATLAB can approximate continuous-time convolution by using the `conv()` function.

```
y = conv(x1,x1);
```

When it is tried to be plotted from MATLAB using the following command;

```
close all;
plot(t,y);
```

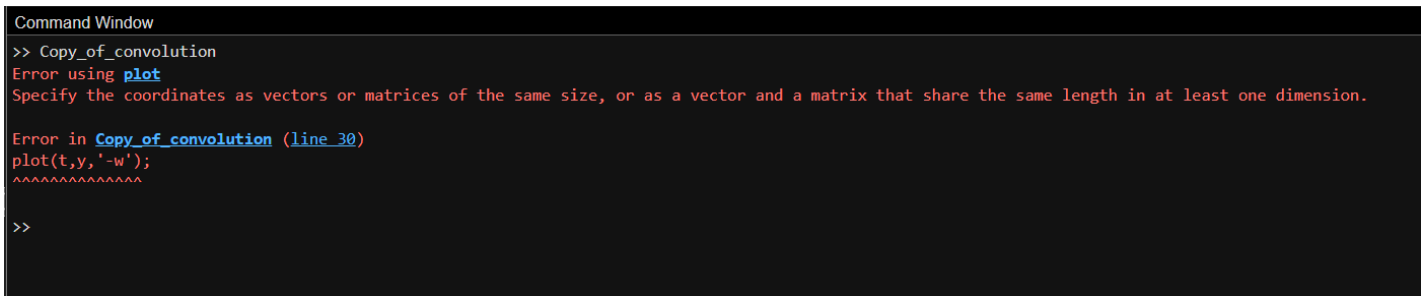


Figure 6: ERROR GENERATED DUE TO VECTOR INCOMPATIBILITY

This is because the two vectors are not in the same length, It can be seen when `length()` function.

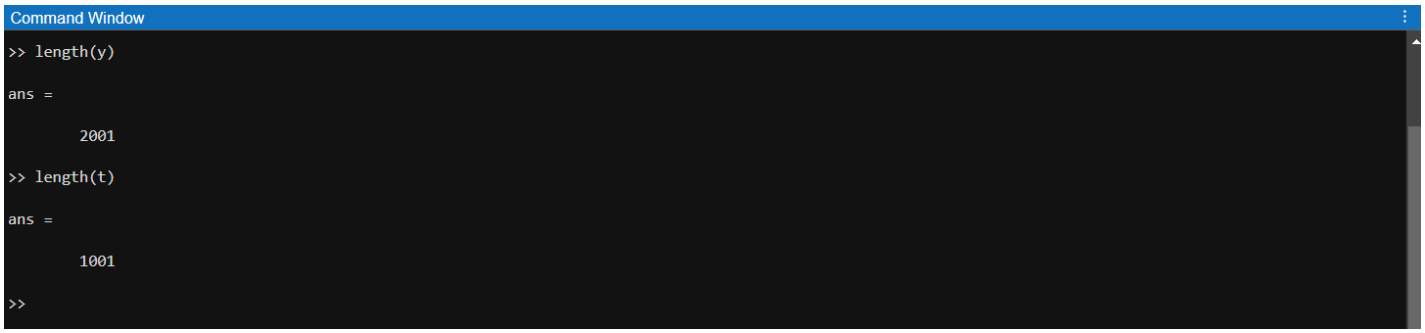


Figure 7: COMPARISON OF CONVOLUTED VECTOR AND TIME VECTOR

Due to this incompatibility, MATLAB generates an error.
Therefore, a new time axis is created to address this incompatibility.

```
t_y = -10:t_s:10;
```

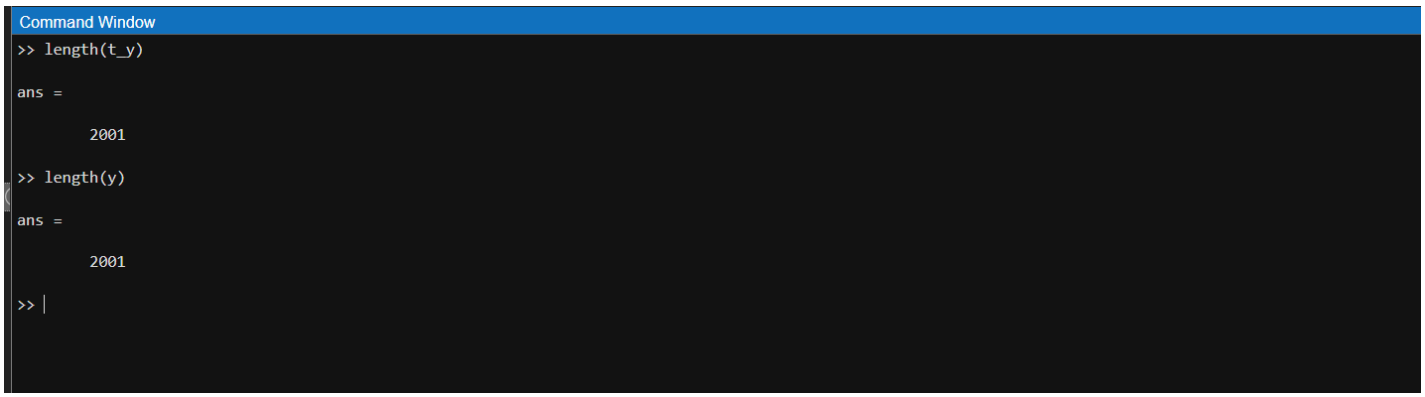


Figure 8: COMPARISON OF CONVOLUTED VECTOR AND NEW TIME VECTOR

Now this is compatible for convolution.

However, convolved answer should be multiplied by the sampling time to get the correct continuous time approximation because in discrete-time convolution (as implemented by MATLAB's conv), the sum approximates the continuous-time convolution integral. To make this approximation accurate, each term in the sum must be multiplied by the sampling interval t_s (which is Δt). Therefore, correct convolution syntax would be `y=t_s*conv(x1,x1); not y=conv(x1,x1);`

Final Convolution

```
clear all;

f_s = 100; %Sampling frequencyt
t_s = 1/f_s;
t = -5:t_s:5; %Time vector where the signal is defined

function x = rect(t)
% RECT rectangular pulse
% Usage: x = rect(t)% This function takes in a vector t of sample instants and outputs the
% corresponding rectangular pulse contained in the function x
    x = double(t >= -0.5 & t < 0.5);
end
```

```
x1 = rect(t);
y=t_s*conv(x1,x1);

%The number of elements are now differnet there fore it is not possible to
%be plotted against the before mentioned t range; new number of t is 2*length(ts)- 1

t_y = -10:t_s:10;

plot(t_y,y,'-w');
axis( [-2 2 -1 2]); %this sets the axis limits of x as [-2 2] and y as [-1 2]
xlabel( 'time (sec)' );
ylabel( 'y_1(t)' ) ;
title ('Figure : y_1(t) = x_1(t)*x_1(t)');
```

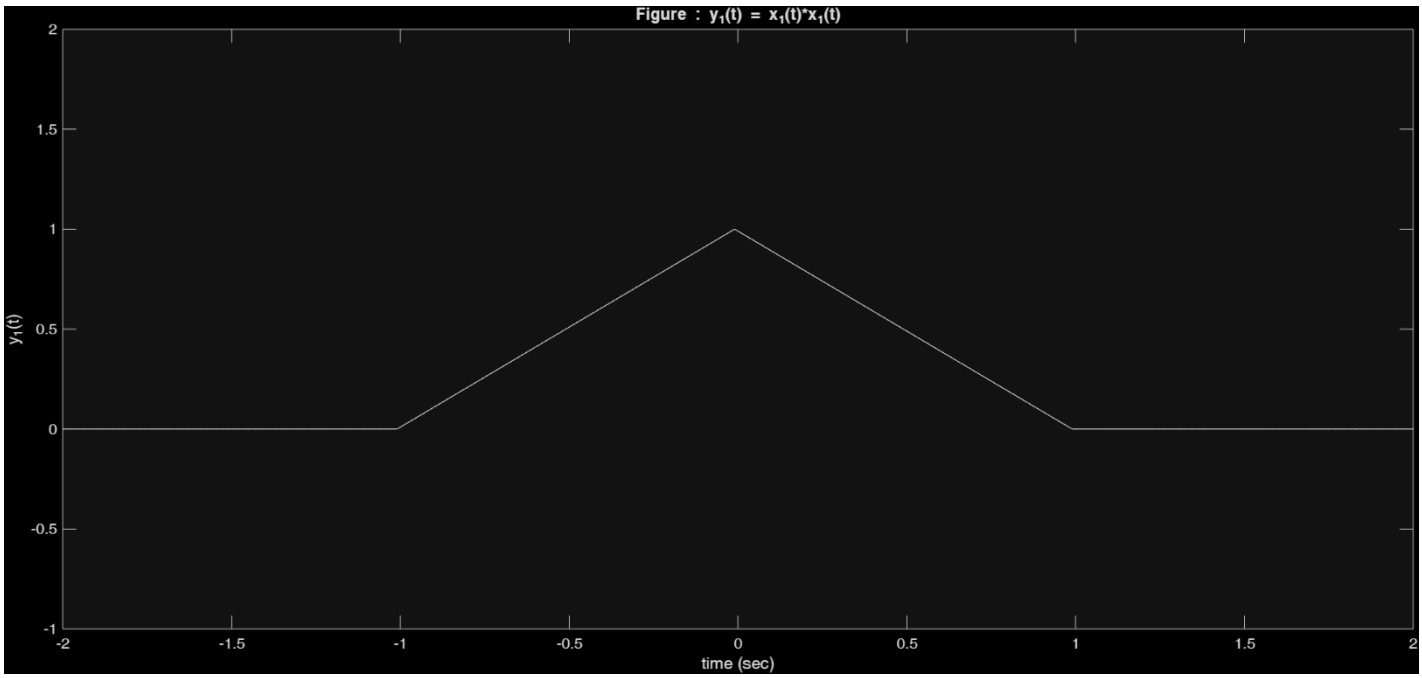


Figure 9: CONVOLUTION RESULT PLOT

Exercise

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.

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

```
% Define discrete time signals x(n) and h(n)
x1 = [1 2 4];
h1 = [1 1 1 1 1];

% Time indices
n_x1 = 0:length(x1)-1;
n_h1 = 0:length(h1)-1;

% Perform convolution
y1 = conv(x1, h1);
n_y1 = 0:length(y1)-1;
```

```
% Find the global x-axis limits
x_min = min([-1, -1, -1]);
x_max = max([n_x1+1, n_h1+1, n_y1+1]);

% Plot using stem for discrete signals
figure;
subplot(3,1,1);
stem(n_x1, x1, 'filled','-w');
title('Signal x(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(x1)+1]);

subplot(3,1,2);
stem(n_h1, h1, 'filled','-w');
title('Signal h(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(h1)+1]);

subplot(3,1,3);
stem(n_y1, y1, 'filled','-w');
title('Convolution y(n) = x(n) * h(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(y1)+1]);
```

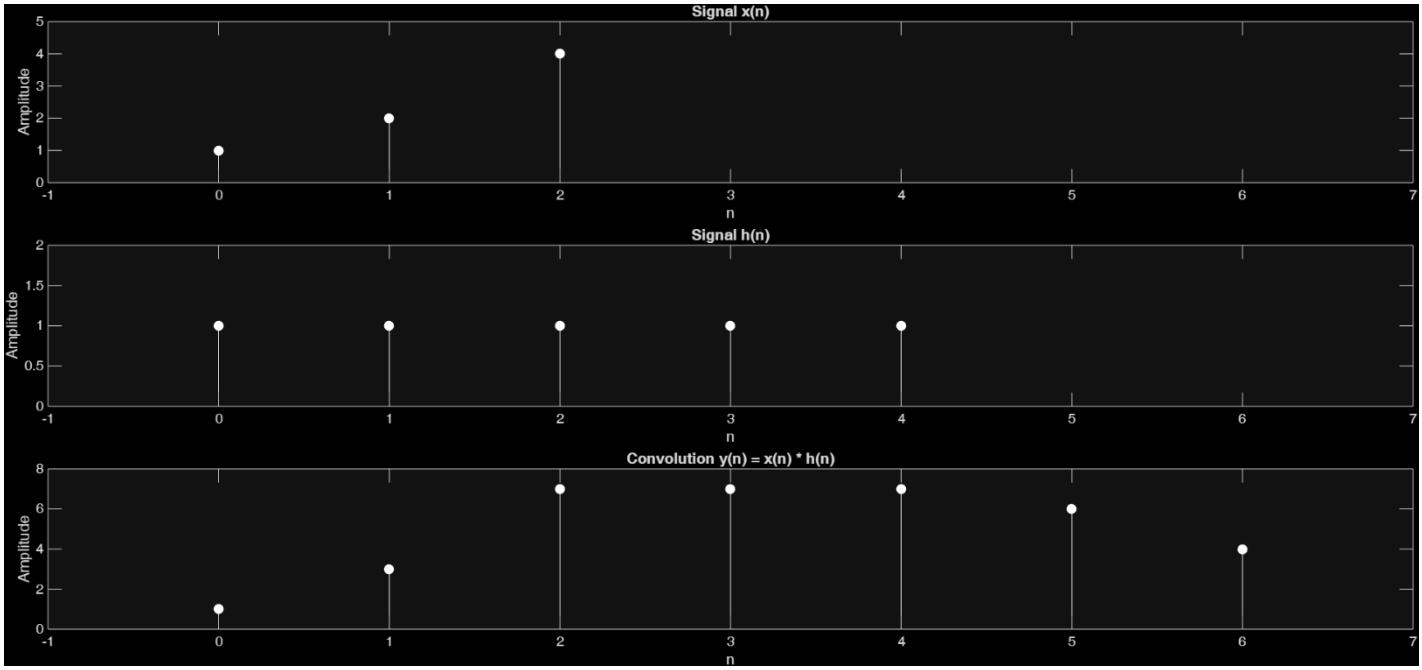


Figure 10: SIGNALS AND CONVOLUTION PLOT

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

```
% Define discrete time signals x(n) and h(n)
x1 = [1 2 3 4 5];
h1 = 1;
```

```
% Time indices
n_x1 = 0:length(x1)-1;
n_h1 = 0:length(h1)-1;

% Perform convolution
y1 = conv(x1, h1);
n_y1 = 0:length(y1)-1;

% Find the global x-axis limits
x_min = min([-5, -5, -5]);
x_max = max([5, 5, 5]);

% Plot using stem for discrete signals
figure;
subplot(3,1,1);
stem(n_x1, x1, 'filled','-w');
title('Signal x(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(x1)+1]);

subplot(3,1,2);
stem(n_h1, h1, 'filled','-w');
title('Signal h(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(h1)+1]);

subplot(3,1,3);
stem(n_y1, y1, 'filled','-w');
title('Convolution y(n) = x(n) * h(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(y1)+1]);
```

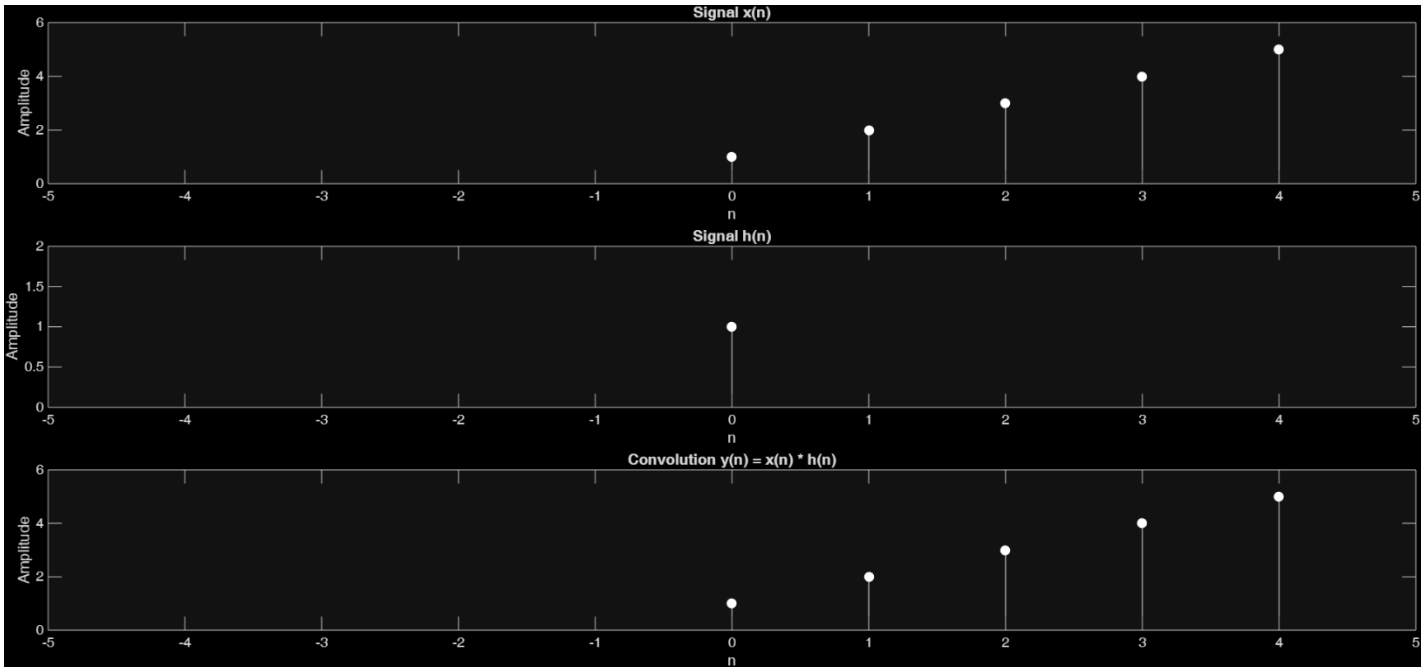


Figure 11: SIGNALS AND CONVOLUTION PLOT

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

```
% Define discrete time signals x(n) and h(n)
x1 = [1 2 0 2 1];
h1 = [1 2 0 2 1];

% Time indices
n_x1 = 0:length(x1)-1;
n_h1 = 0:length(h1)-1;

% Perform convolution
y1 = conv(x1, h1);
n_y1 = 0:length(y1)-1;

% Find the global x-axis limits
x_min = min([-2, -2, -2]);
x_max = max([10, 10, 10]);

% Plot using stem for discrete signals
figure;
subplot(3,1,1);
stem(n_x1, x1, 'filled','-w');
title('Signal x(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(x1)+1]);

subplot(3,1,2);
stem(n_h1, h1, 'filled','-w');
title('Signal h(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(h1)+1]);

subplot(3,1,3);
stem(n_y1, y1, 'filled','-w');
title('Convolution y(n) = x(n) * h(n)');
xlabel('n'); ylabel('Amplitude');
xlim([x_min x_max]);
ylim([0 max(y1)+1]);
```

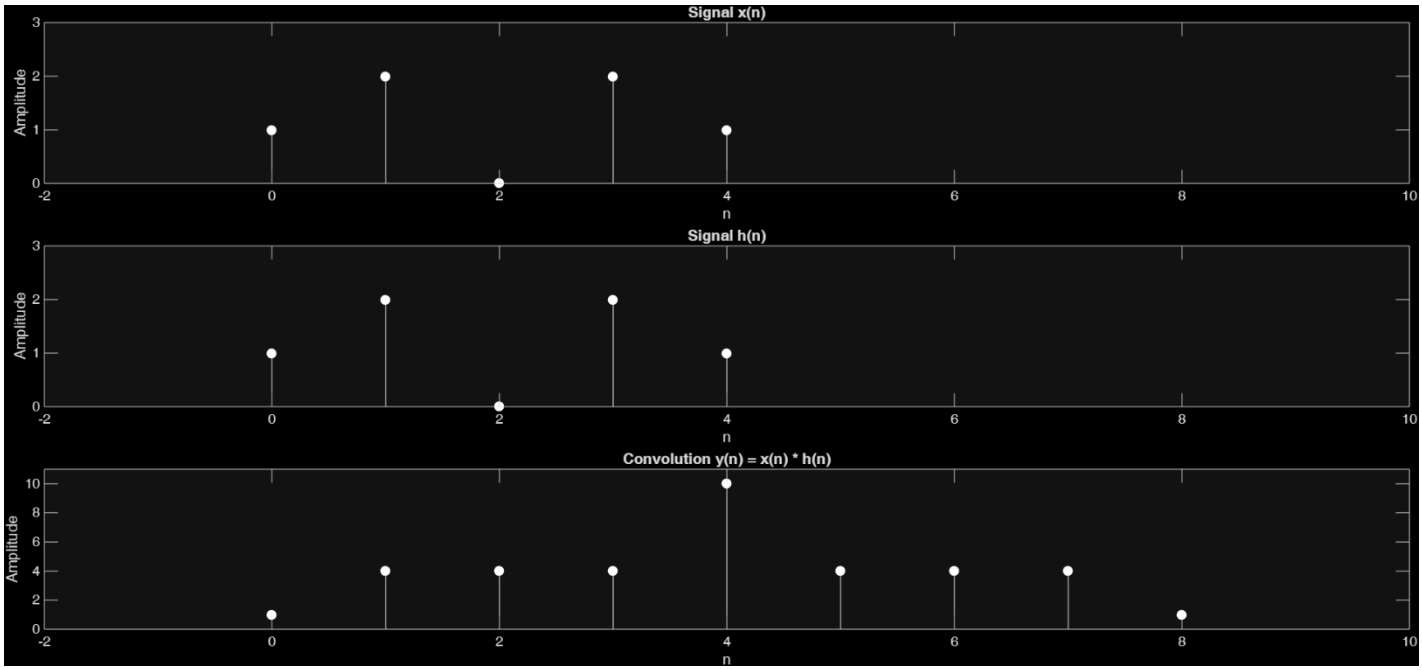


Figure 12: SIGNALS AND CONVOLUTION PLOT

2. Finding Input Signal from the Output and Impulse Response

Assume a system with the following impulse response:

$$h(n) = 0.5^n \; for \; 0 \; \leq n < 4; \; 0 \; elsewhere$$

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.

```
% Define impulse response
n_h = 0:3; % define n_h from 0 to 3
h = 0.5.^n_h; % h(n) = 0.5^n for n = 0, 1, 2, 3, h[n] = [1, 0.5, 0.25, 0.125]

% Given output
y = [1, 2, 2.5, 3, 3, 3, 2, 1, 0];
n_y = 0:length(y)-1;

% Find input x(n) using deconvolution
[x, r] = deconv(y, h); %deconvolution to find x(n) from y(n) and h(n)
n_x = 0:length(x)-1;

disp(x); %x[n] is produced here
% deconv(y, h) finds the input sequence x such that when x is convolved with h, you get y
(i.e., y = conv(x, h)).
% x is the quotient (the estimated input signal).
% r is the remainder (the part of y that cannot be explained by convolution with h).

% Find global x-axis limits for alignment
x_min = min([-1, -1, -1]);
x_max = max([9, 9, 9]);

% Plot all three signals with aligned x-axes
figure;
subplot(3,1,1);
stem(n_h, h, 'filled','-w');
title('Impulse Response h(n)');
xlabel('n'); ylabel('h(n)');
```

```
xlim([x_min x_max]);

subplot(3,1,2);
stem(n_y, y, 'filled','-w');
title('Output y(n)');
xlabel('n'); ylabel('y(n)');
xlim([x_min x_max]);

subplot(3,1,3);
stem(n_x, x, 'filled','-w');
title('Input x(n)');
xlabel('n'); ylabel('x(n)');
xlim([x_min x_max]);
```

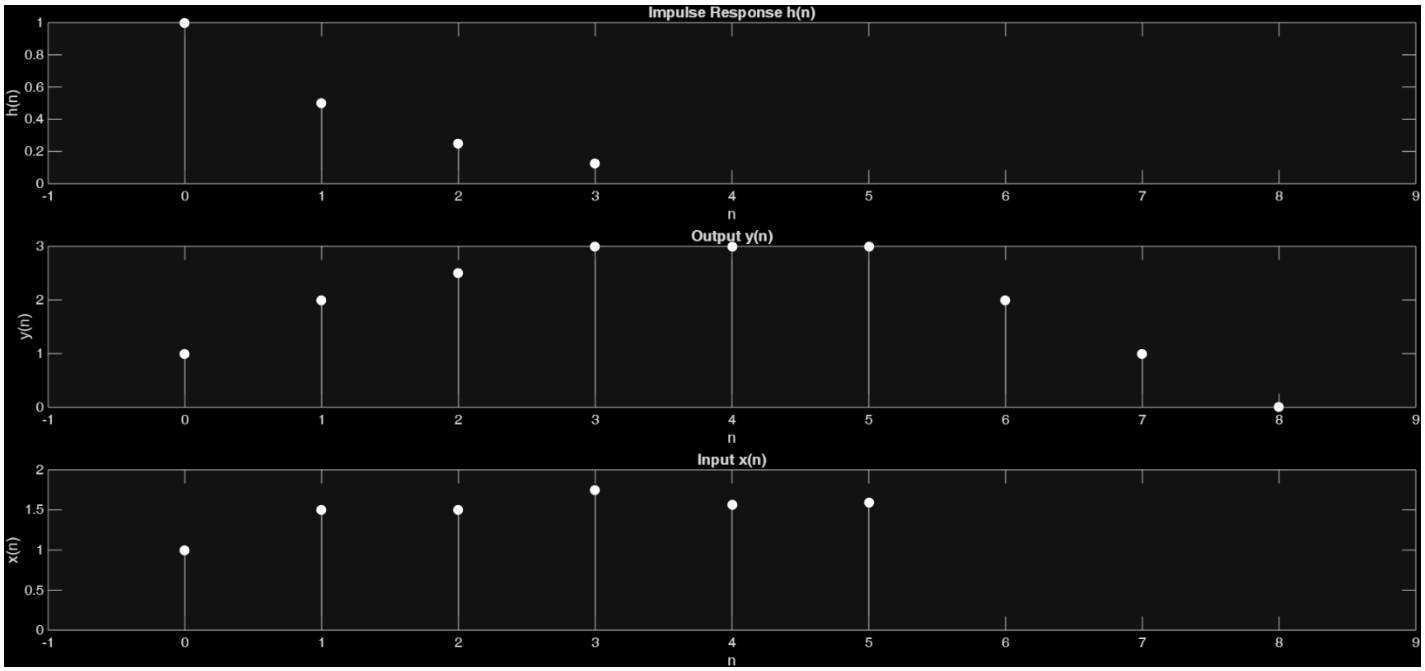


Figure 13: IMPULSE RESPONSE, OUTPUT AND DECONVOLUTED INPUT