Computer Assignment 2

EEE391- Basics of Signals and Systems

Efe Acer
21602217

Bilkent University, CS

Efe Acer
21602217

Contents

The *convolution sum* is defined for discrete signals $x[n]$ and $y[n]$ as follows:

$$z[n] = x[n] * y[n] = \sum_{\ell=-\infty}^{\infty} x[\ell] \cdot y[n-\ell] \tag{1}$$

If both $x[n]$ and $y[n]$ are finite sequences, meaning that they have a finite number of nonzero values, i.e. their supports are finite; we can reduce the summation in (1) by restricting the summation interval to the support of the sequences. Recall that a sequence's support refers to the subset of the sequence's domain containing the elements that are all mapped to nonzero values. Let the set of integers $\{-M_1, -M_1+1, ..., 0, ..., M_2-1, M_2\}$ contain the support of $x[n]$ where M_1 and M_2 are both positive integers. Similarly, let the set $\{-N_1, -N_1+1, ..., 0, ..., N_2-1, N_2\}$ contain the support of $y[n]$, again N_1 and N_2 are positive integers. Then, the *convolution sum* can be rewritten as:

$$z[n] = x[n] * y[n] = \sum_{\ell=-M_1}^{M_2} x[\ell] \cdot y[n-\ell] \tag{2}$$

or equivalently:

$$z[n] = x[n] * y[n] = \sum_{k=-N_1}^{N_2} x[n-k] \cdot y[k] \tag{3}$$

The equivalence of (2) and (3) is a direct implication of the transitivity property of the *convolution sum*:

$$\sum_{\ell=-\infty}^{\infty} x[n] \cdot y[n-\ell] \overset{\text{k=n}-\ell}{\longleftrightarrow} \sum_{k=\infty}^{-\infty} x[n-k] \cdot y[k] = \sum_{k=-\infty}^{\infty} x[n-k] \cdot y[k] \tag{4}$$

The derivation above is translated into a `Matlab` function, namely `convolve`, as specified in the assignment sheet. The `convolve` function together with the complete `Matlab` code for the assignment are located in the Appendix section of this report. The digital signals given in the assignment sheet, $x_1[n]$ and $x_2[n]$, are stored in `Matlab` arrays after implementing their defining functions, then the requested convolutions are computed using the `convolve` function. The results are plotted after the time axis is properly adjusted, and their correctness are checked using `Matlab`'s built-in `conv` function. The following pages display the given digital signals $x_1[n]$ and $x_2[n]$ together with the convolutions, for parts a (1), b (2) and c (3).

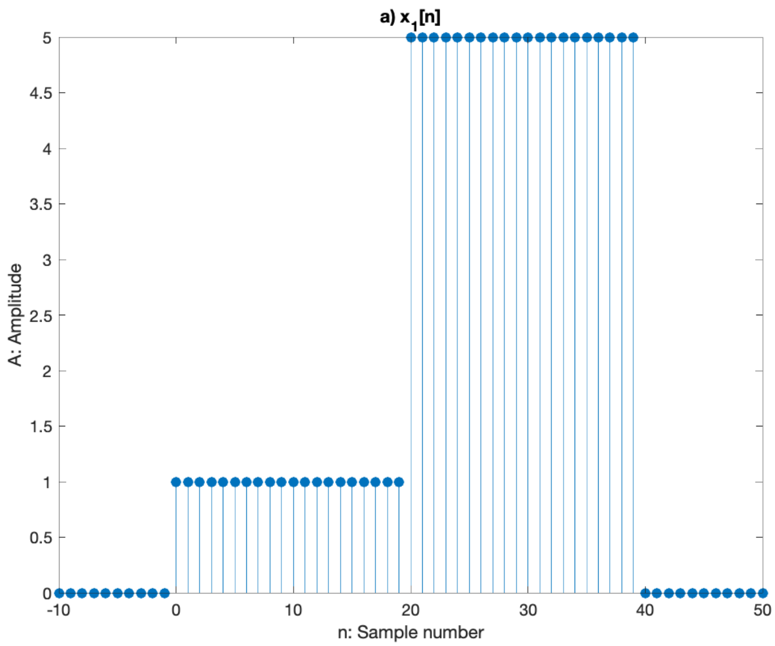

Figure 1: $x_1[n]$ for part a

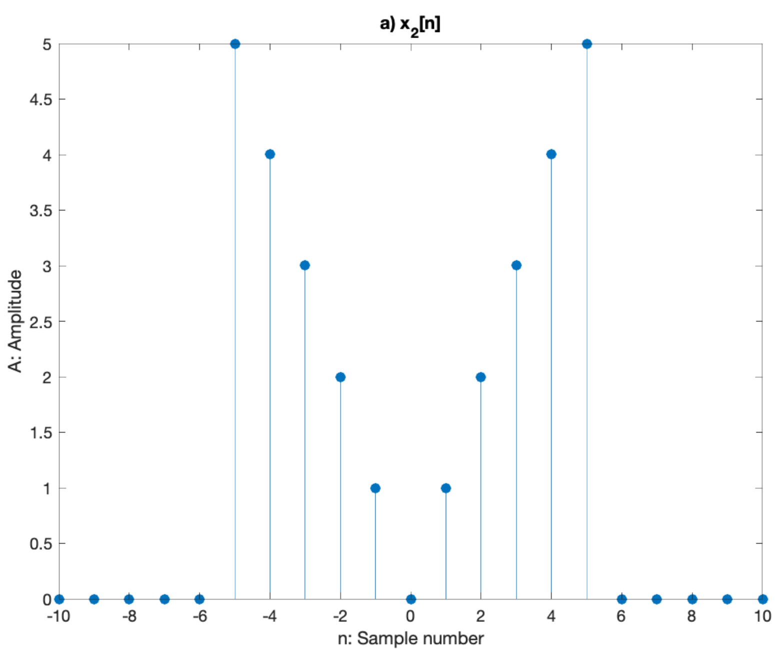

Figure 2: $x_2[n]$ for part a

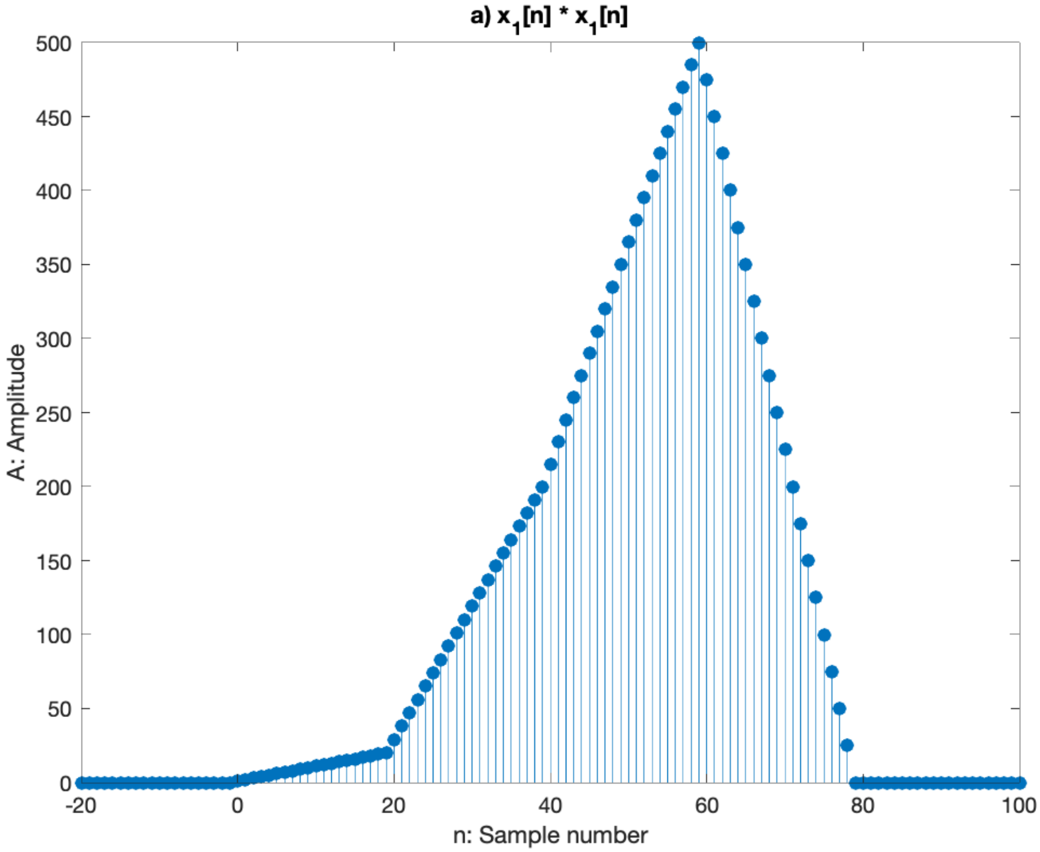

Figure 3: $x_1[n] * x_1[n]$ for part a

First nonzero time index of $x_1[n] * x_1[n]$ is 0.

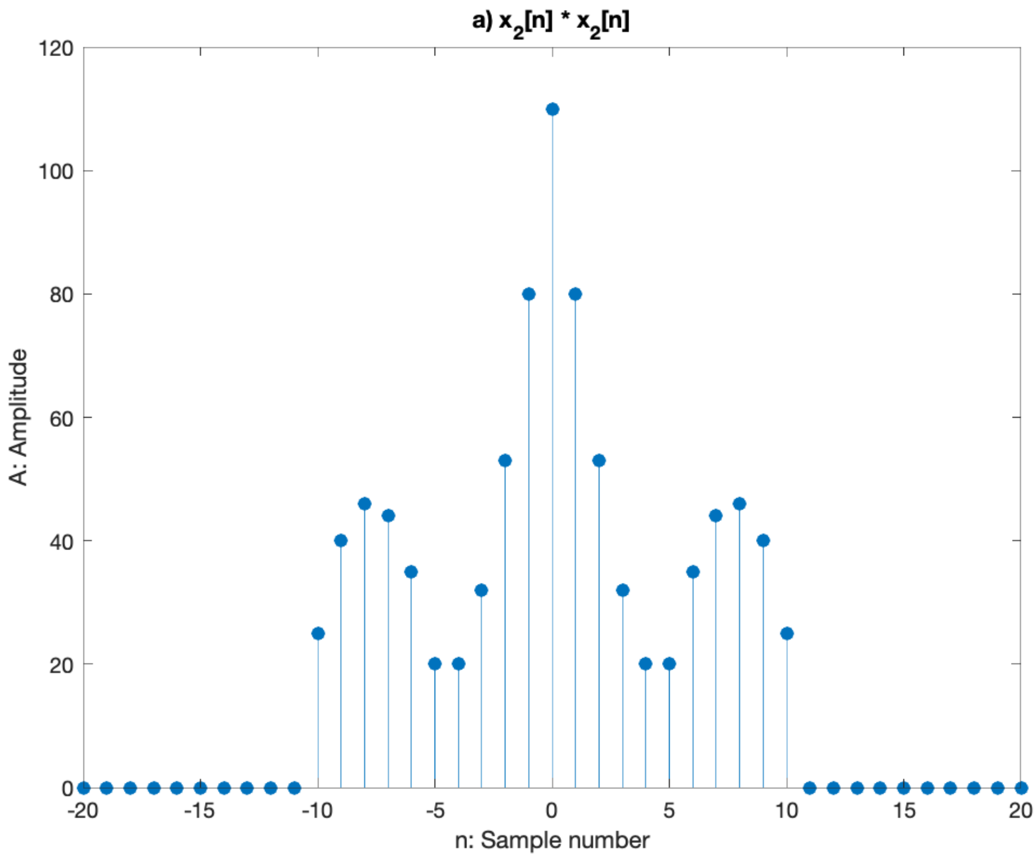

Figure 4: $x_2[n] * x_2[n]$ for part a

First nonzero time index of $x_2[n] * x_2[n]$ is -10.

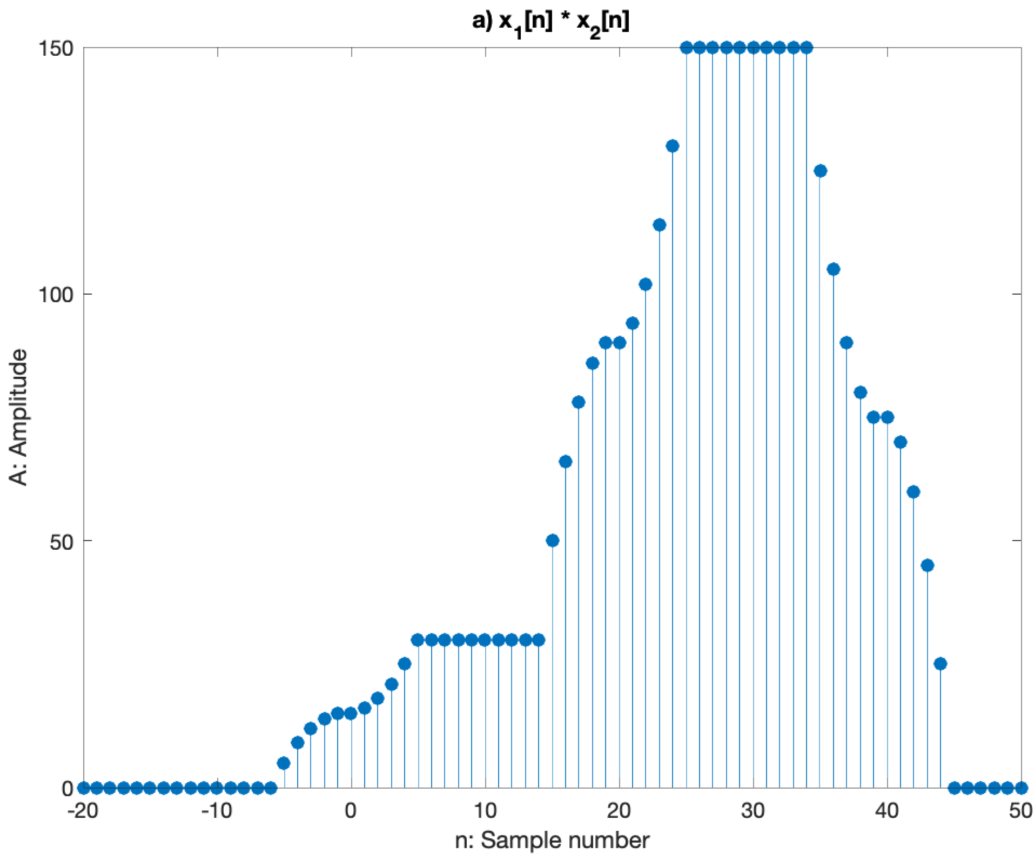

Figure 5: $x_1[n] * x_2[n]$ for part a

First nonzero time index of $x_1[n] * x_2[n]$ is -5.

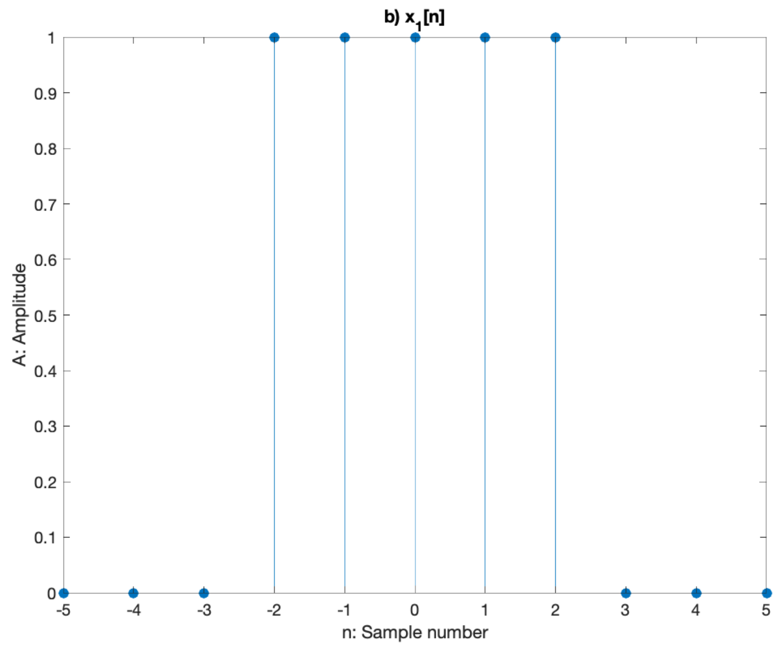

Figure 6: $x_1[n]$ for part b

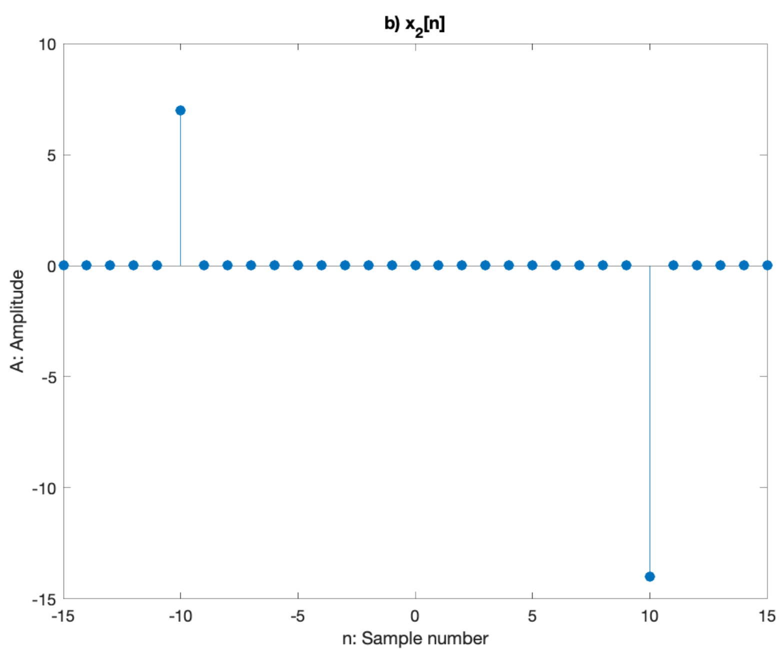

Figure 7: $x_2[n]$ for part b

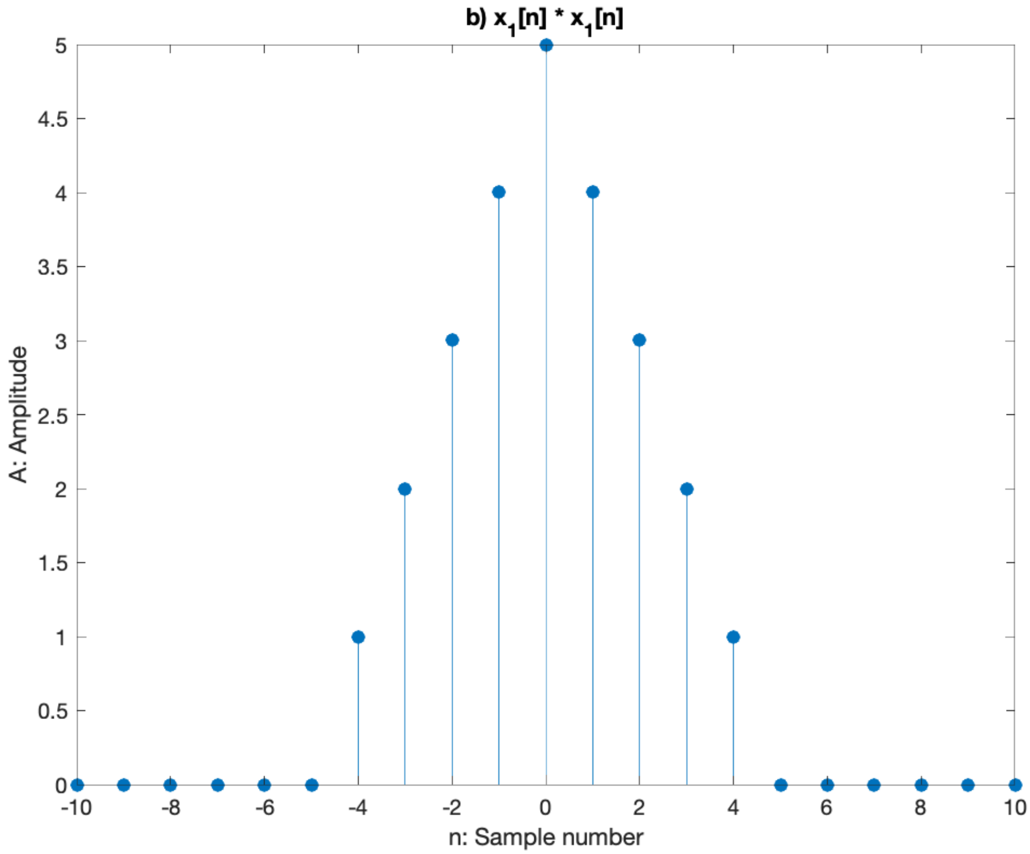

Figure 8: $x_1[n] * x_1[n]$ for part b

First nonzero time index of $x_1[n] * x_1[n]$ is -4.

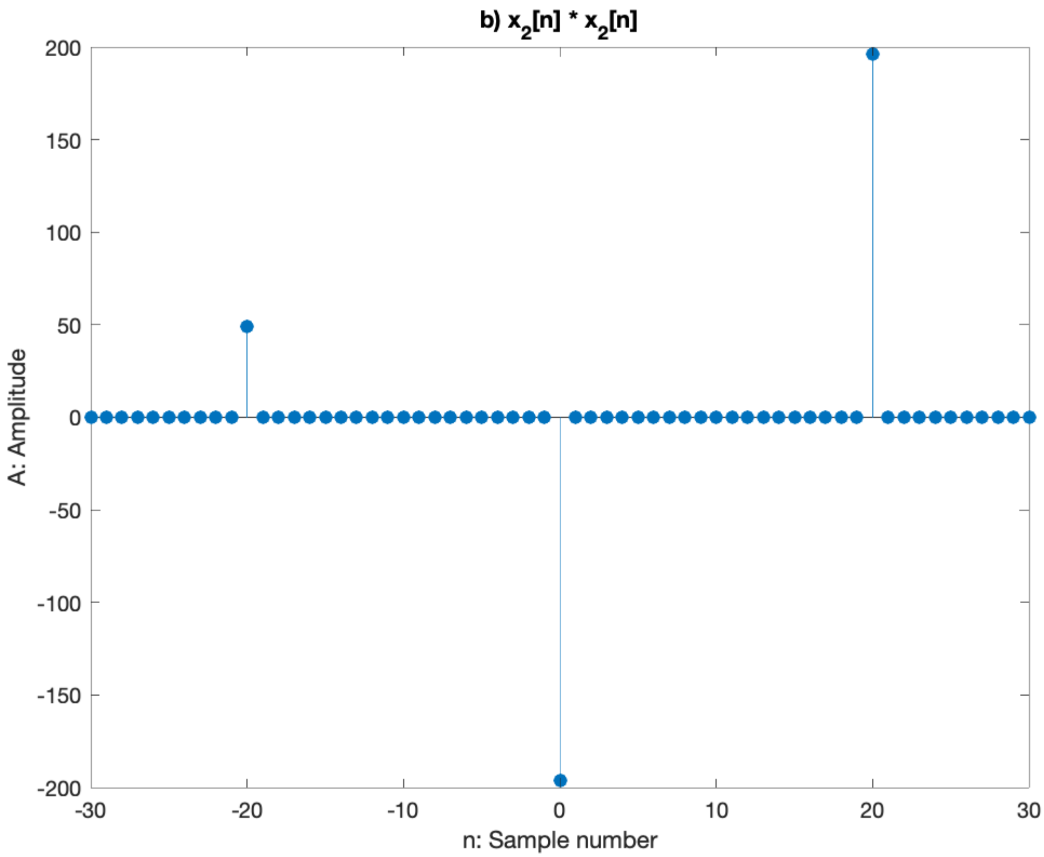

Figure 9: $x_2[n] * x_2[n]$ for part b

First nonzero time index of $x_2[n] * x_2[n]$ is -20.

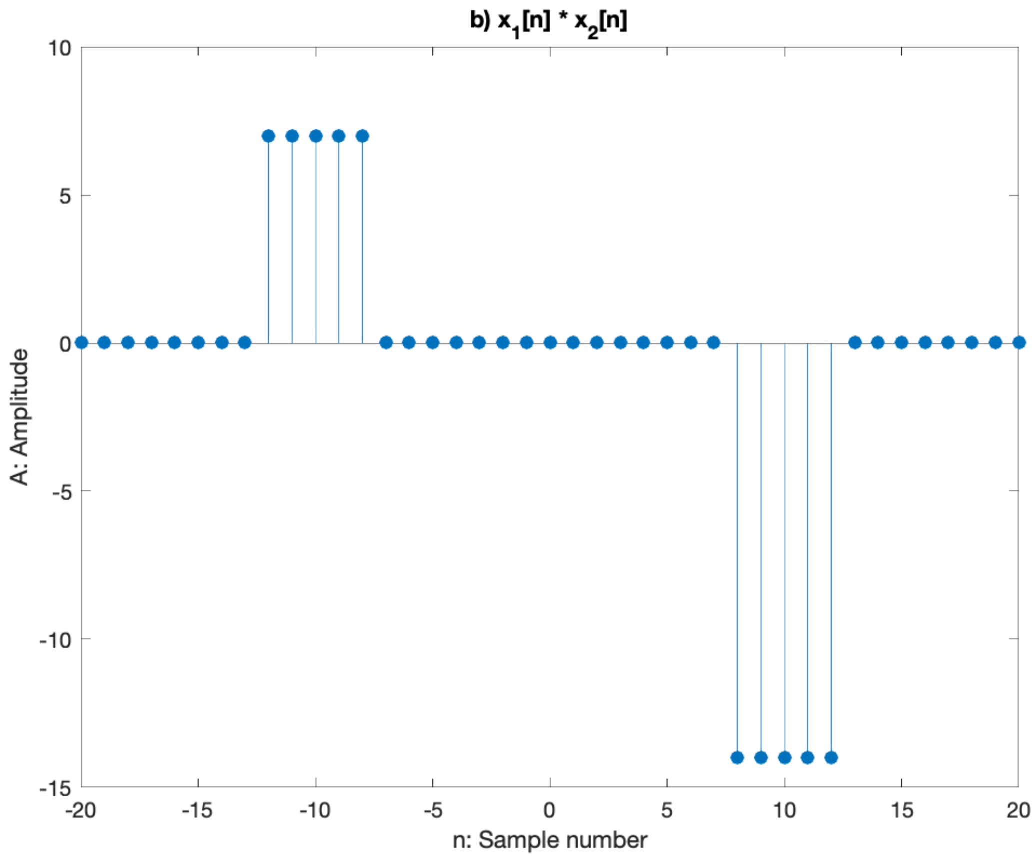

Figure 10: $x_1[n] * x_2[n]$ for part b

First nonzero time index of $x_1[n] * x_2[n]$ is -12.

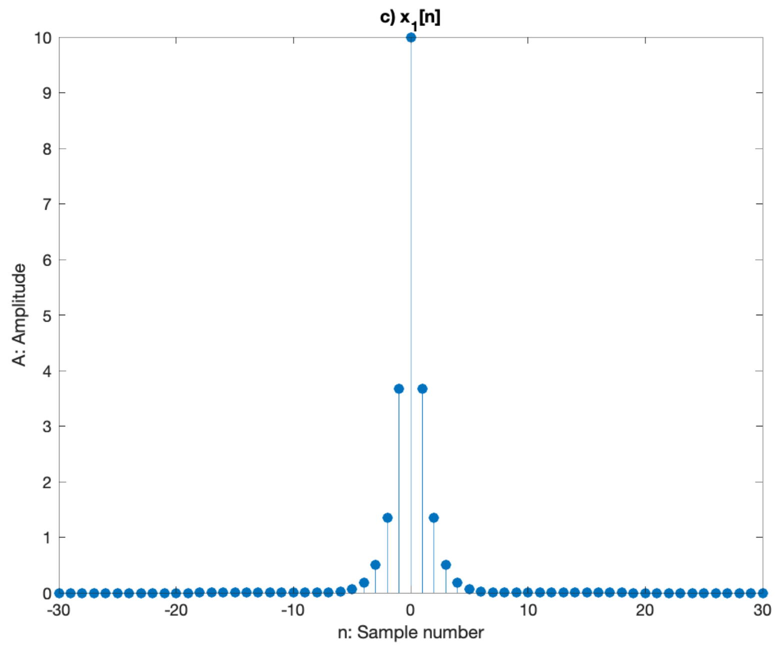

Figure 11: $x_1[n]$ for part c

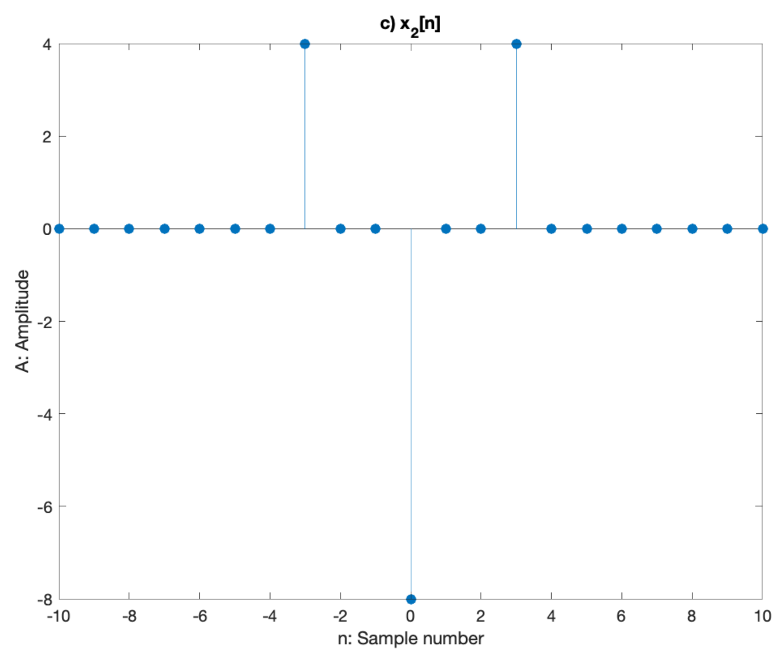

Figure 12: $x_2[n]$ for part c

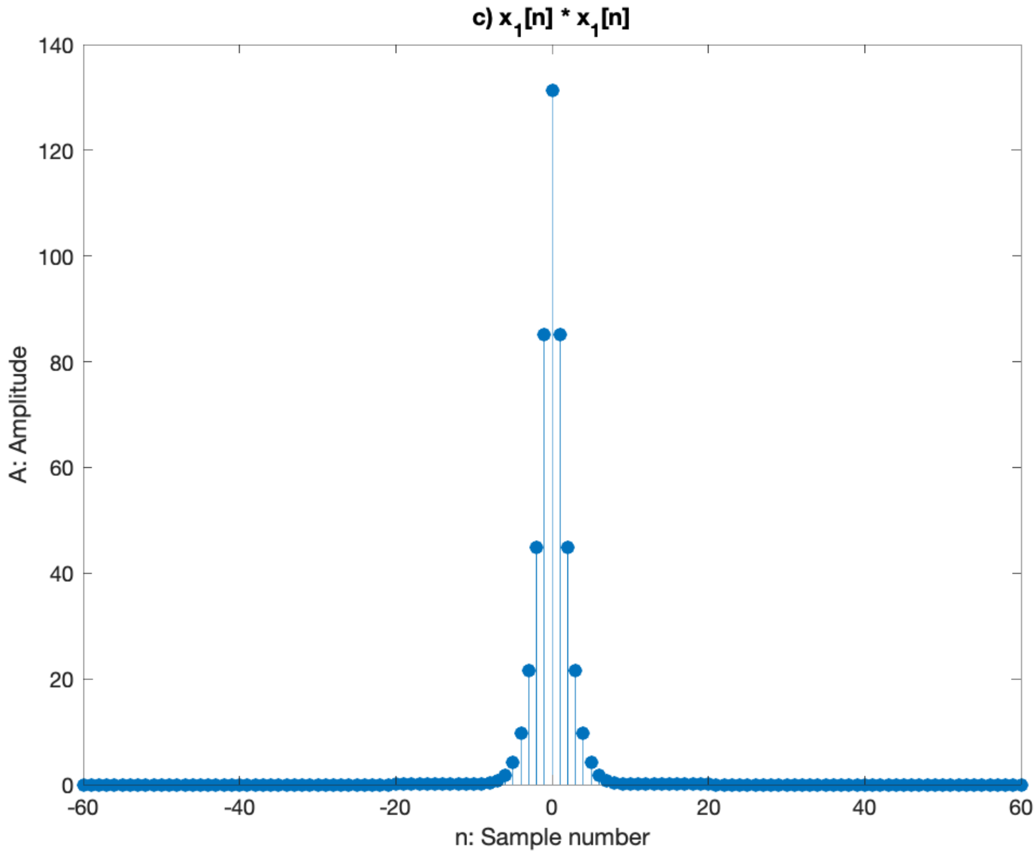

Figure 13: $x_1[n] * x_1[n]$ for part c

First nonzero time index of $x_1[n] * x_1[n]$ is -50, however the corresponding value is very close to zero.

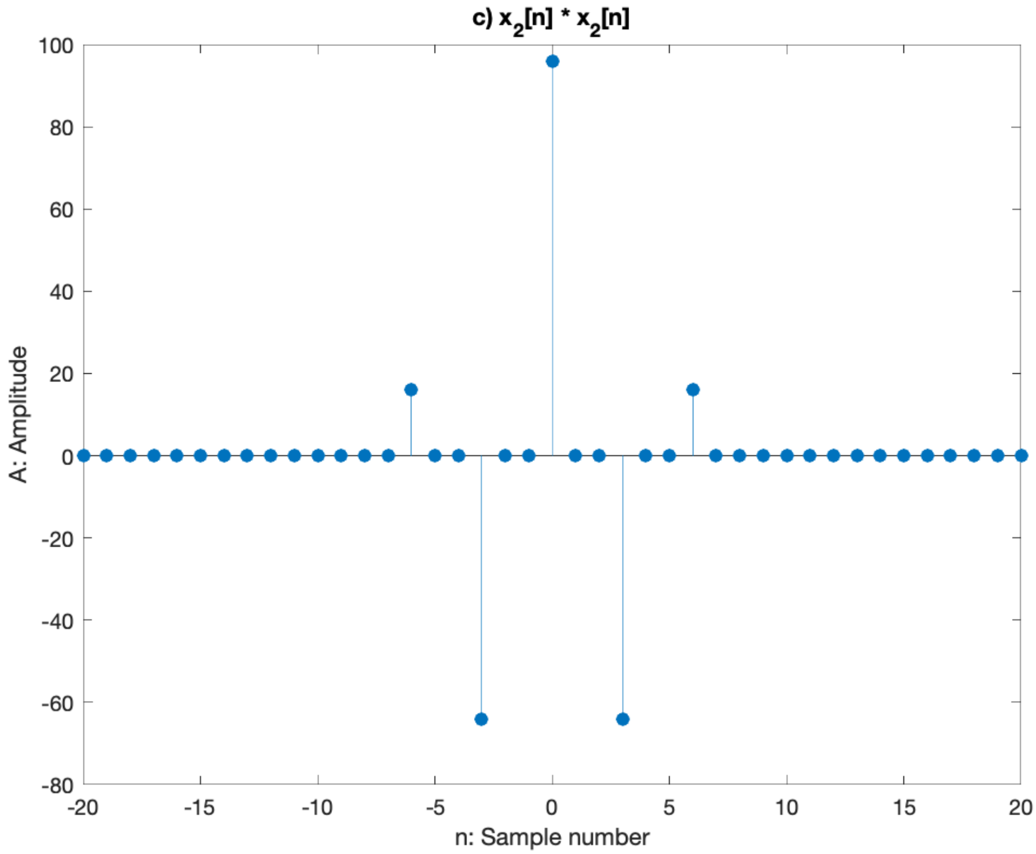

Figure 14: $x_2[n] * x_2[n]$ for part c

First nonzero time index of $x_2[n] * x_2[n]$ is -6.

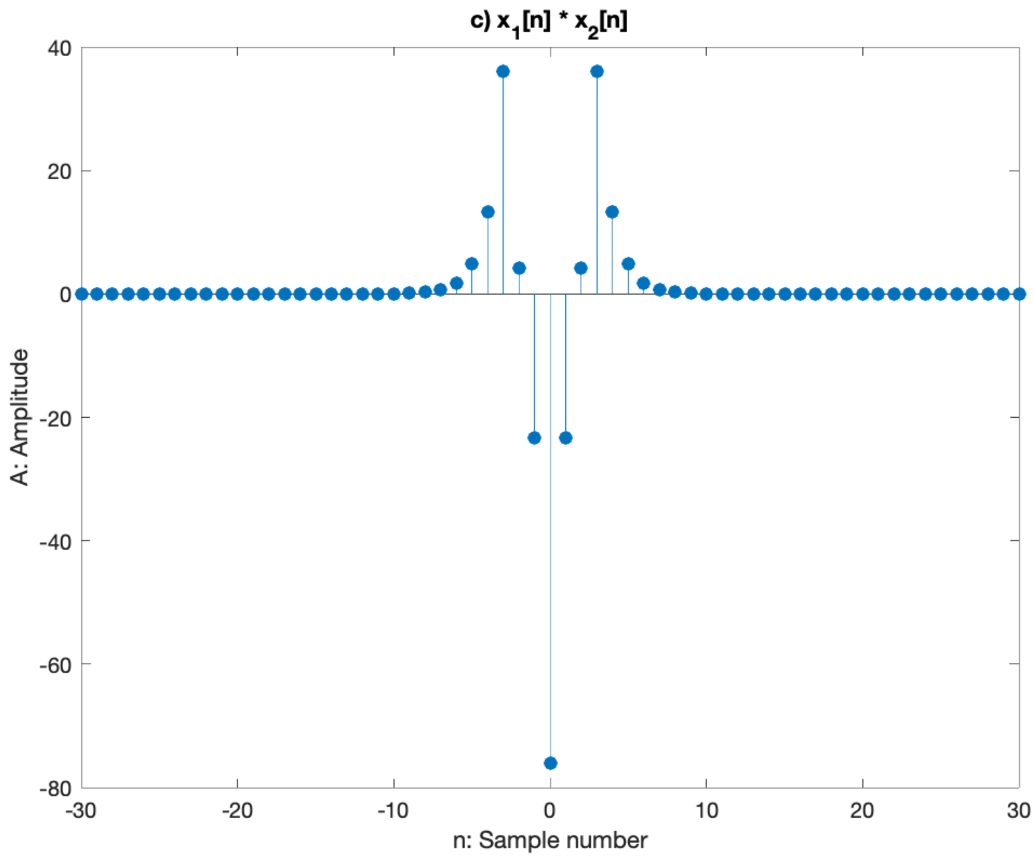

Figure 15: $x_1[n] * x_2[n]$ for part c

First nonzero time index of $x_1[n] * x_2[n]$ is -28, however the corresponding value is very close to zero.

Complete Matlab Code

```
% EEE391 Computer Assignment 2
% Author: EFE ACER

%% a)

offset = 101;
x1 = zeros(1, 201);
for n = -100:100
    if n >= 0 && n <= 19
        x1(n + offset) = 1;
    elseif n >= 20 && n <= 39
        x1(n + offset) = 5;
    end
end
n_x1 = 0;

figure(1);
stem(-100:100, x1, 'filled');
xlim([-10, 50]);
title('a) x_1[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

x2 = zeros(1, 201);
for n = -100:100
    if abs(n) <= 5
        x2(n + offset) = abs(n);
    end
end
n_x2 = -5;

figure(2);
stem(-100:100, x2, 'filled');
xlim([-10, 10]);
title('a) x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x1, x1);
[z, n_z] = convolve(x1, x1, n_x1, n_x1);
figure(3);
idx = find(z ~= 0, 1, 'first');
```

```
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-20, 100])
title('a) x_1[n] * x_1[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x2, x2);
[z, n_z] = convolve(x2, x2, n_x2, n_x2);
figure(4);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-20, 20])
title('a) x_2[n] * x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x1, x2);
[z, n_z] = convolve(x1, x2, n_x1, n_x2);
figure(5);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-20, 50])
title('a) x_1[n] * x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%% b)

x1 = zeros(1, 201);
for n = -100:100
    if -2 * n + 4 >= 0
        x1(n + offset) = x1(n + offset) + 1;
    end
    if -n - 3 >= 0
        x1(n + offset) = x1(n + offset) - 1;
    end
end
n_x1 = -2;

figure(6);
stem(-100:100, x1, 'filled');
```

```
xlim([-5, 5])
title('b) x_1[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

x2 = zeros(1, 201);
for n = -100:100
    if -n - 10 == 0
        x2(n + offset) = 7;
    elseif -n + 10 == 0
        x2(n + offset) = -14;
    end
end
n_x2 = -10;

figure(7);
stem(-100:100, x2, 'filled');
xlim([-15, 15])
title('b) x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x1, x1);
[z, n_z] = convolve(x1, x1, n_x1, n_x1);
figure(8);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-10, 10])
title('b) x_1[n] * x_1[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x2, x2);
[z, n_z] = convolve(x2, x2, n_x2, n_x2);
figure(9);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-30, 30])
title('b) x_2[n] * x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

```

```
%z = conv(x1, x2);
[z, n_z] = convolve(x1, x2, n_x1, n_x2);
figure(10);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-20, 20])
title('b) x_1[n] * x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%% c)

x1 = zeros(1, 201);
for n = -100:100
    if abs(n) <= 25
        x1(n + offset) = 10 * exp(-abs(n));
    end
end
n_x1 = -25;

figure(11);
stem(-100:100, x1, 'filled');
xlim([-30, 30])
title('c) x_1[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

x2 = zeros(1, 201);
for n = -100:100
    if n + 3 == 0
        x2(n + offset) = 4;
    elseif n == 0
        x2(n + offset) = -8;
    elseif n - 3 == 0
        x2(n + offset) = 4;
    end
end
n_x2 = -3;

figure(12);
stem(-100:100, x2, 'filled');
xlim([-10, 10])
title('c) x_2[n]');
```

```
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x1, x1);
[z, n_z] = convolve(x1, x1, n_x1, n_x1);
figure(13);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-60, 60])
title('c) x_1[n] * x_1[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x2, x2);
[z, n_z] = convolve(x2, x2, n_x2, n_x2);
figure(14);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-20, 20])
title('c) x_2[n] * x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%z = conv(x1, x2);
[z, n_z] = convolve(x1, x2, n_x1, n_x2);
figure(15);
idx = find(z ~= 0, 1, 'first');
time_vals = (1:length(z)) - (idx - n_z);
stem(time_vals, z, 'filled');
xlim([-30, 30])
title('c) x_1[n] * x_2[n]');
xlabel('n: Sample number');
ylabel('A: Amplitude');

%% Discrete Convolution Sum Implementation:

function [z, n_z] = convolve(x, y, n_x, n_y)
% CONVOLVE Computes the discrete convolution sum of two finite supported
% sequences x and y. n_x and n_y denotes the first time indices of the
% sequences which correspond to nonzero values. z returns the convolution
% result and n_z denotes the first time index of z's support.
    z = zeros(1, length(x) + length(y) - 1);
```

```
    for i = 1:length(x)
        for j = 1:length(y)
            z(i + j - 1)= z(i + j - 1) + x(i) * y(j);
        end
    end
    n_z = n_x + n_y;
end
```