Ahmet Faruk Çolak

22102104

2024-2025 Spring

EE-321: SIGNALS & SYSTEMS
LAB 2

Introduction

In this lab, convolution operation was studied. Convolution is used for linear time invariant systems, and it helps to understand how systems react other systems. Let's assume we have a basic sinusoid signal, and we have also a low pass filter modeled signal. If we convolve sinusoid with LPF, we basically get reaction of sinusoid to our LPF. That's why convolution operation is a helpful mathematical concept to understand how signals act, basically. We also built a speech recognition algorithm and detect our speech. This was the fun part of the Lab.

Part 1.1

In this part, we are asked to find the length of $\psi[n]$.

$$\psi[n] = (\xi[n]u[n]) * (\eta[n]u[n])$$

Let us determine the length for $(\xi[n]u[n])\ and\ (\eta[n]u[n])$ as $N_\xi\ and\ N_\eta$ respectively.

Length for the convolution determined as:

$$N_\xi\ +\ N_\eta - 1$$

In Figure.1 we can clearly see the length of the convolution operation.

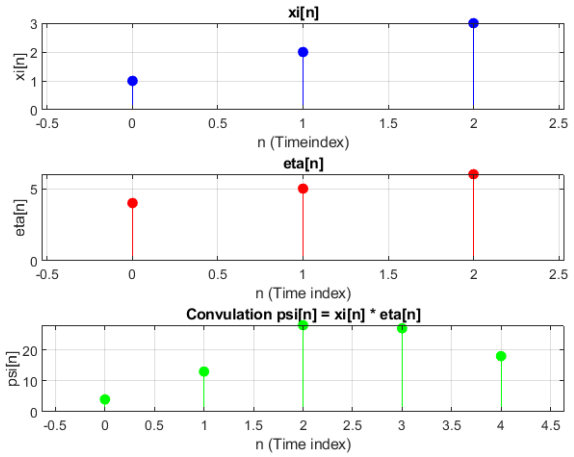


Figure.1: Length of convolution

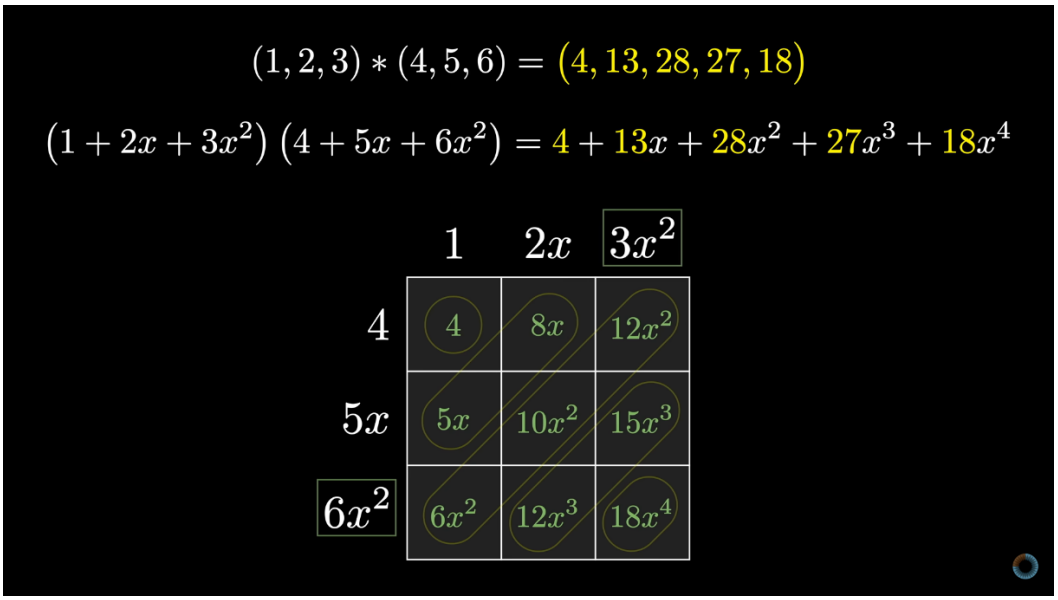


Figure.2: Convolution operation in a unique perspective

In Figure.2, YouTube channel *3Blue1Brown* explains convolution operation, with acting each element of the set is like a coefficient of a polynomial. This may be a bit easy way to understand this operation.

Part 1.2

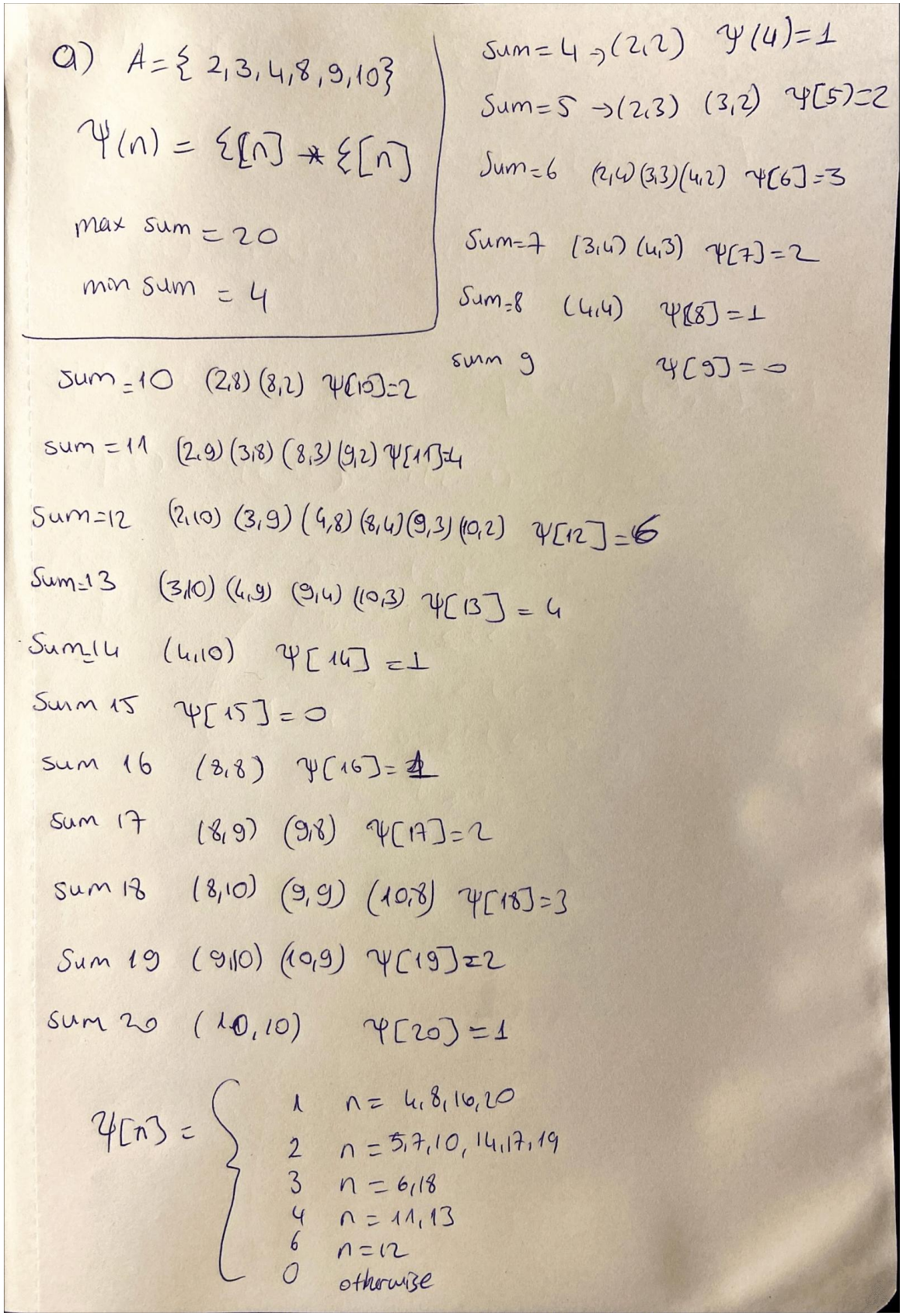


Figure.3: Part1.2 a

b) $\xi(t) = u(t) - u(t-1)$

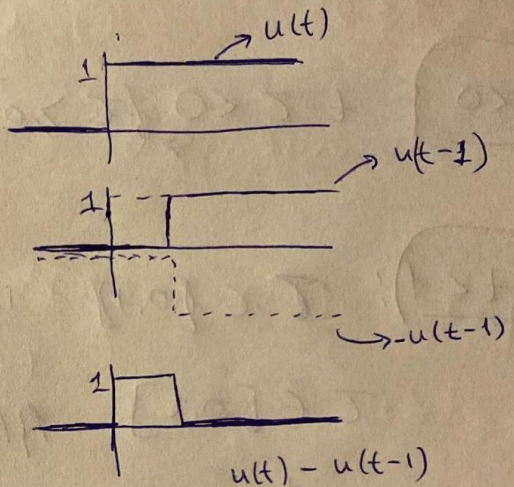
$r(t) = r(t) - 2r(t-1)$

(1) $\xi(t) = u(t) - u(t-1)$

$t < 0 \; ; \; \xi(t) = 0$

$0 \leq t < 1 \; ; \; \xi(t) = 1$

$t \geq 1 \; ; \; \xi(t) = 0$



(2) $\eta(t) = r(t) - 2r(t-1)$

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

$r(t-1) = (t-1)\,u(t-1)$

$\eta(t)$

$t < 0 \; ; \; \eta(t) = 0$

$0 \leq t < 1 \; ; \; r(t) = t \; , \; r(t-1) = 0 \rightarrow \eta(t) = t$

$t \geq 1 \; ; \; r(t) = t \; , \; r(t-1) = t-1$

$\eta(t) = t - 2(t-1) = t - 2t + 2 = 2 - t$

$\eta(t) = \begin{cases} 0, & t < 0 \\ t, & 0 \leq t < 1 \\ 2-t & t \geq 1 \end{cases}$

Figure.4: Part1.2 b

$$\xi * \eta(t) = \int_{-\infty}^{+\infty} \xi(\tau)\,\eta(t-\tau)\,d\tau$$

$\xi(\tau) = 1$ only $\tau \in [0,1]$ so $\int_0^1 \eta(t-\tau)\,d\tau$

$t < 0$ $\quad 0 < t-\tau$ $\quad (0 \le \tau) \rightarrow 0 = \eta(t-\tau)$ $\quad 0 = \eta * \xi(t)$

$0 \le t < 1$; $\tau \in [0,1]$ if $\tau < t$, $0 < t-\tau$, $\eta(t-\tau) = 0$

$t-\tau = (t-\tau)\eta$, $\quad \tau \in [0,t]$

$$\xi * \eta(t) = \int_0^t d\tau = \left[\tau\right]_{\tau=0}^{\tau=t} = t - 0 = t$$

$\frac{t^2}{2}$; $\frac{t^2}{2} = \frac{t}{2} \cdot t$

$1 \le t < 2$;

$\tau \in [0, t-1] \rightarrow 1 < t-\tau \leftarrow \eta(t-\tau) = 1-(t-\tau)$

$\tau \in [t-1, 1] \rightarrow 1 > t-\tau \ge 0 \leftarrow \eta(t-\tau) = t-\tau$

$$\xi * \eta(t) = \int_0^{t-1} d\tau + \int_{t-1}^{1} (t-\tau)\,d\tau$$

Figure.5: Part1.2 b-2

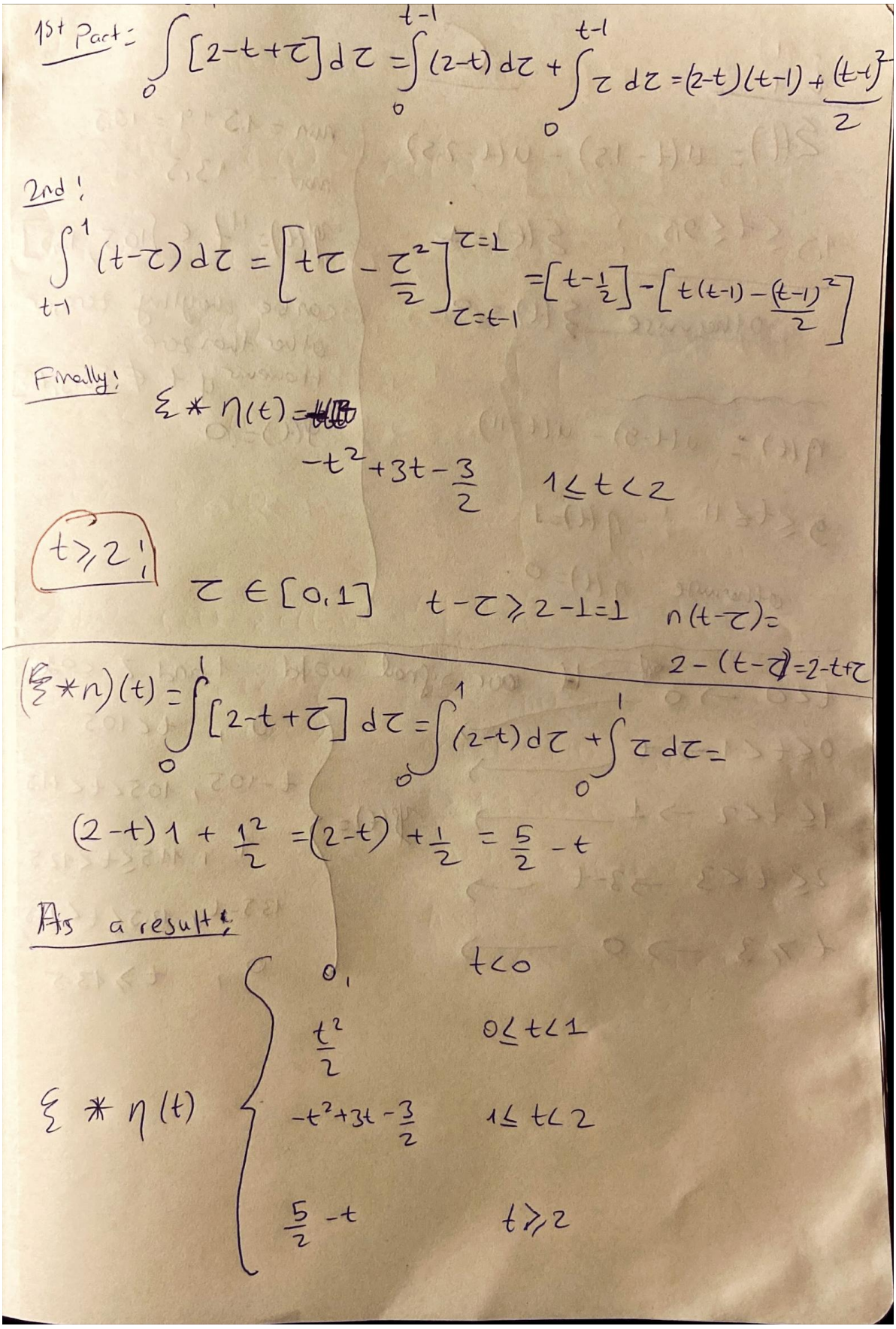


Figure.6: Part1.2 b-3

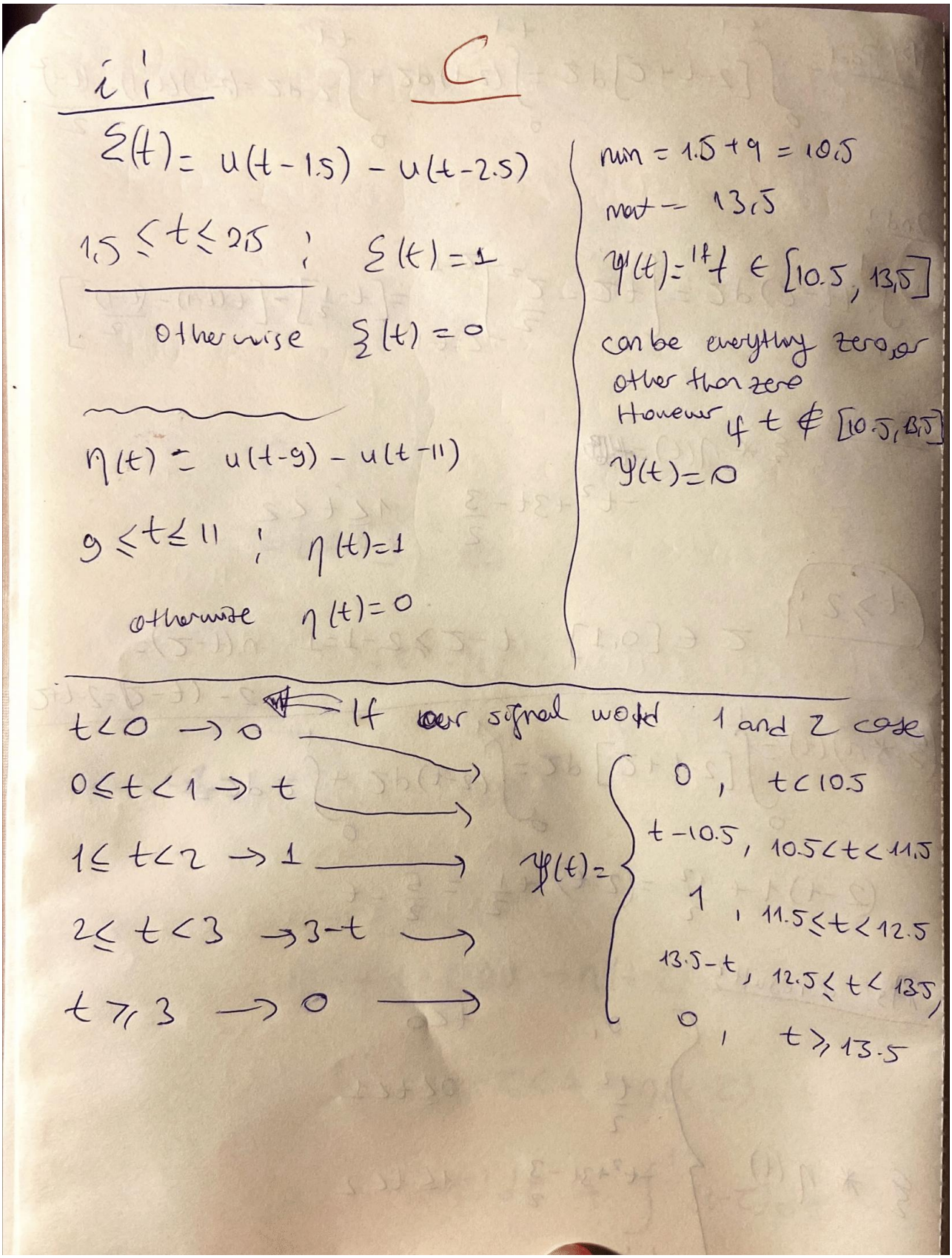


Figure.7: Part1.2 c-i

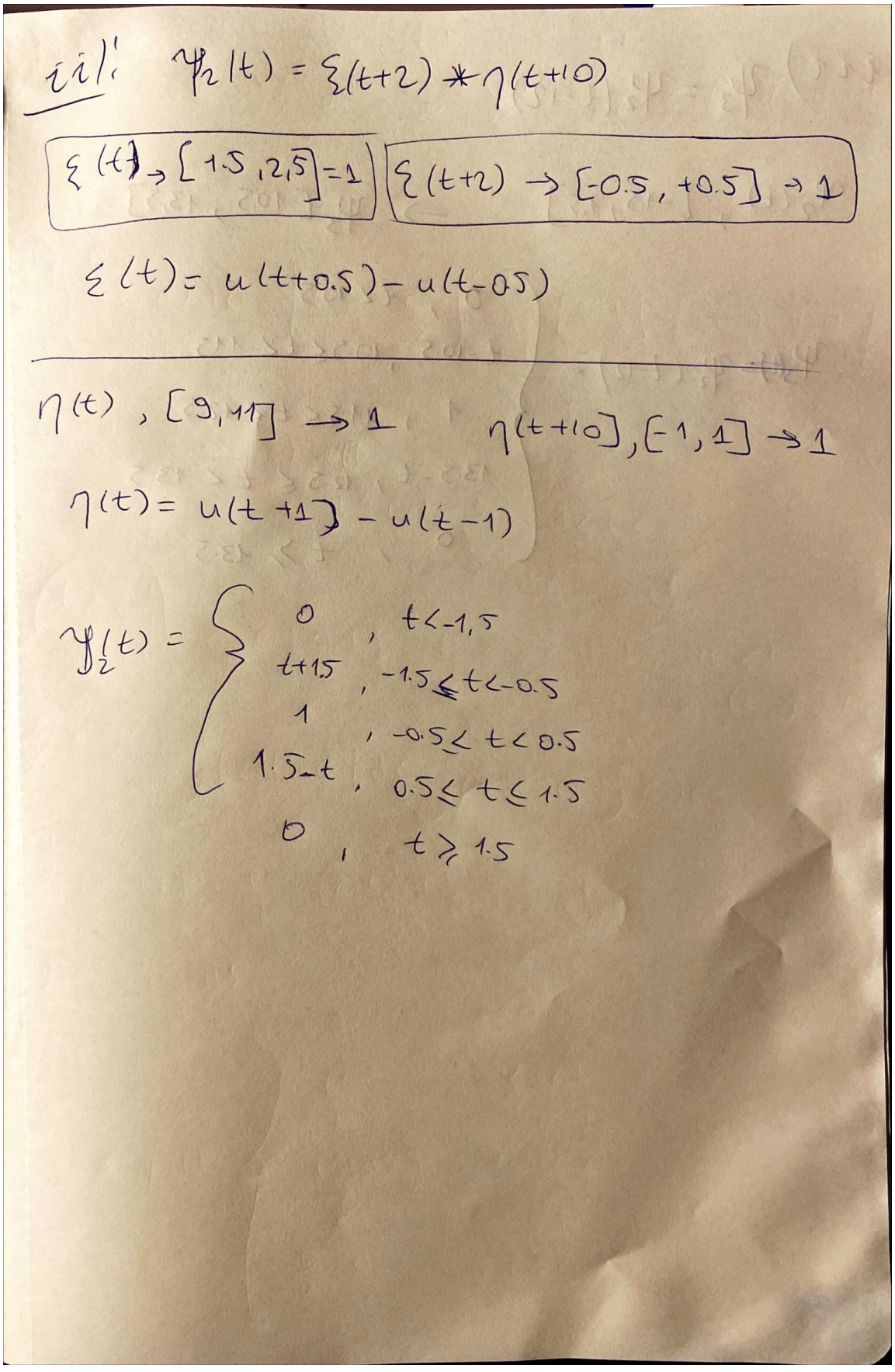


Figure.8: Part1.2 c-ii

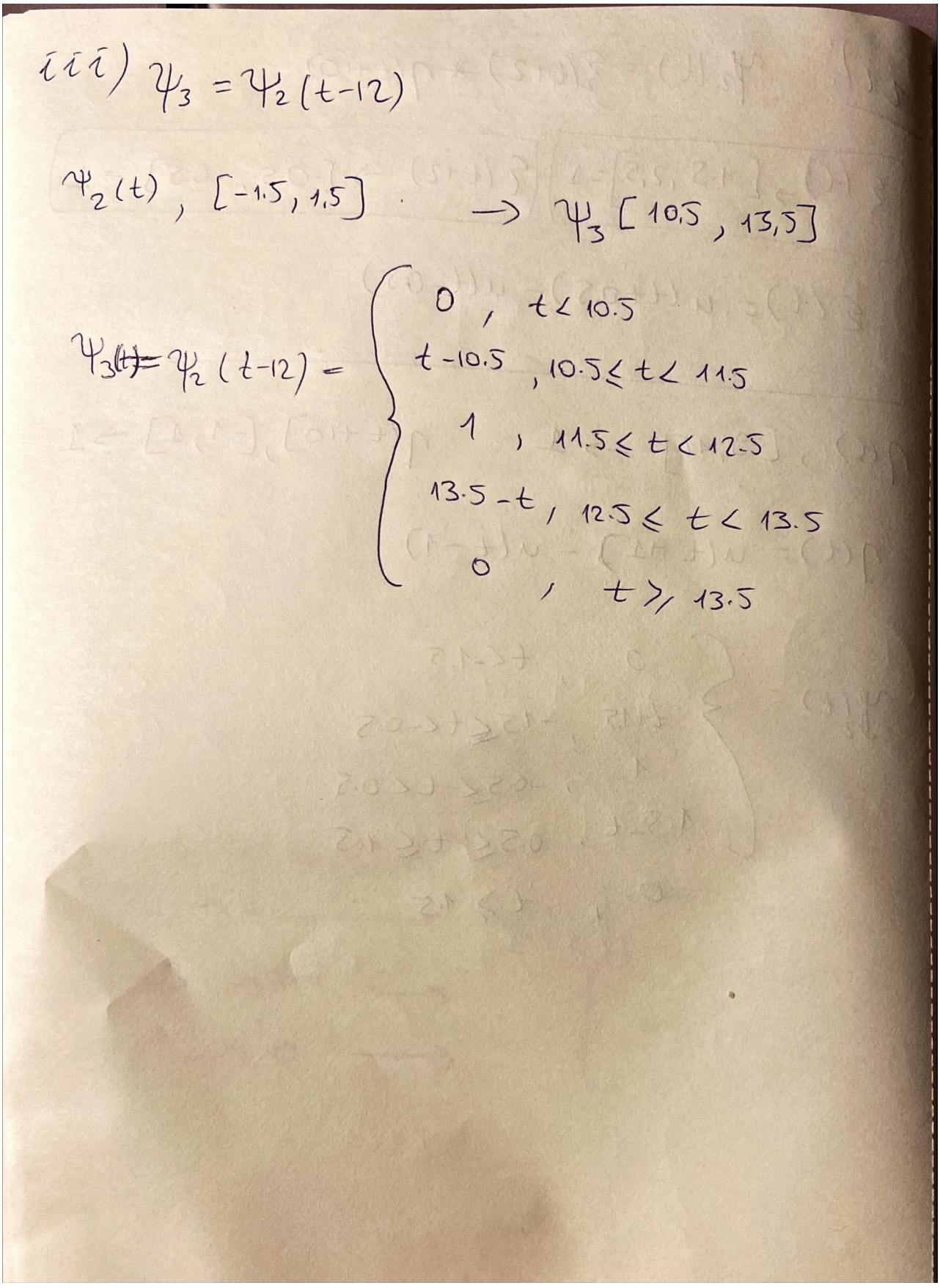


Figure.9: Part1.2 c-iii

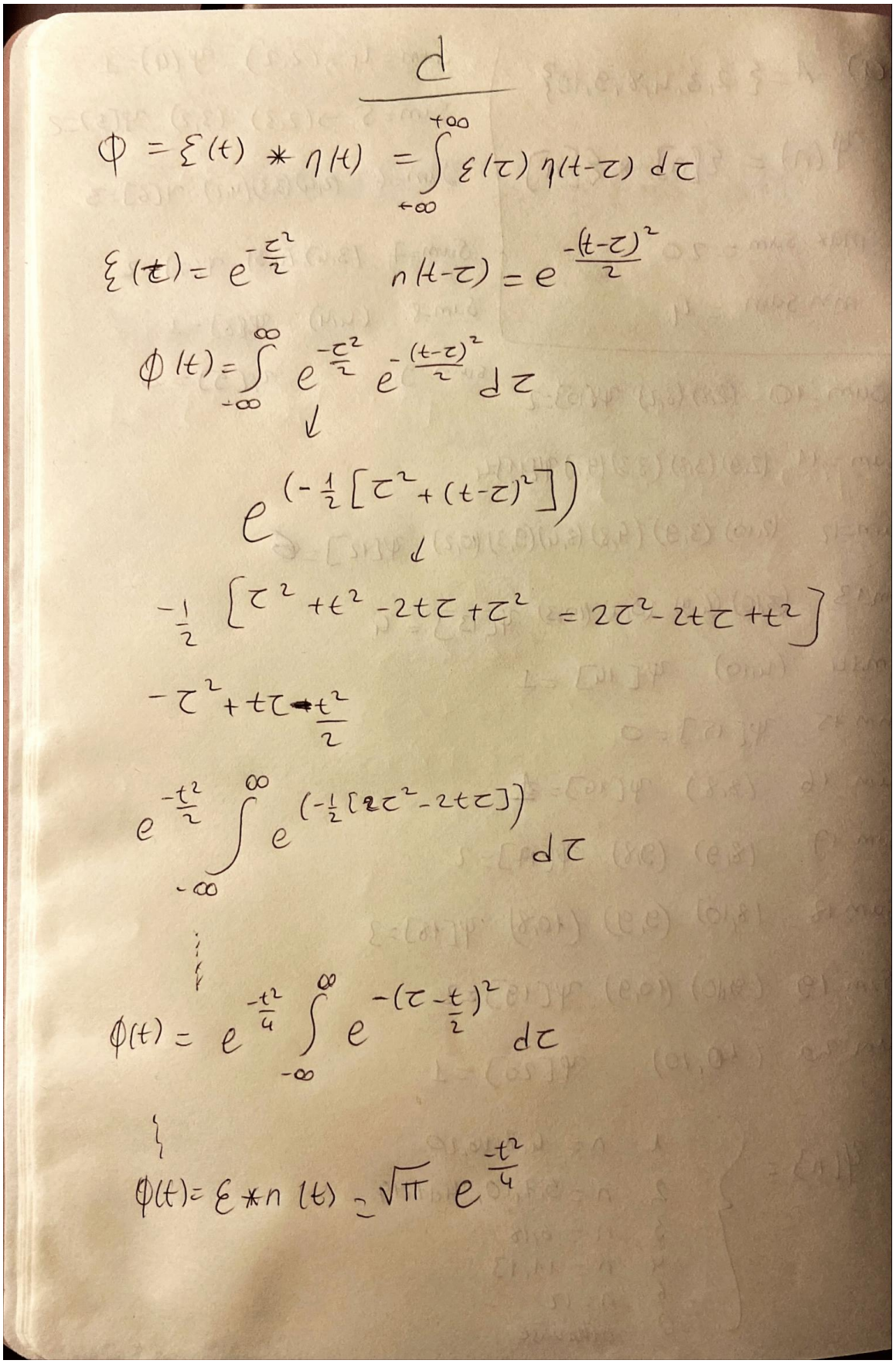


Figure.10: Part1.2 d

Part 2

2.1 Implementing Convolution

In this part, according to mathematical derivations, we developed a MATLAB code for convolution operation.

```
function y = ConvFUNC(x, h)
% ConvFUNC: Tek for döngüsü kullanarak iki dizinin konvolüsyonunu hesaplar.

    Nx = length(x);
    Nh = length(h);
    Ny = Nx + Nh - 1;
    y  = zeros(1, Ny);


    for n = 1:Ny

        kmin = max(1, n - Nh + 1);
        kmax = min(n, Nx);


        k      = kmin : kmax;
        x_part = x(k);
        h_part = h(n - k + 1);

        y(n) = sum( x_part .* h_part );
    end
end
```

2.2 Testing the Convolution Function

```
% a)
n = 0:5;
x = ones(1,6);
ya = ConvFUNC(x, x);
subplot(4,1,1)
stem(0:length(ya)-1, ya)
title('a')
xlabel('n')
ylabel('Amplitude')

% b)
dt = 0.01;
t = 0:dt:4;
psi = double(t>=1.5 & t<2.5) - double(t>=2.5 & t<3.5);
r = @(x) x.*(x>=0);
nb = r(t) - 2*r(t-1) + r(t-2);

yb = ConvFUNC(psi, nb) * dt;
tb = 0:dt:(length(yb)-1)*dt;
subplot(4,1,2)
plot(tb, yb)
title('b)')
xlabel('Time')
ylabel('Amplitude')
```

```
% c)
t = 0:dt:15;
xc = double(t>=1.5 & t<2.5);
psic = double(t>=13.5 & t<14.5);

yc = ConvFUNC(xc, psic) * dt;
tc = 0:dt:(length(yc)-1)*dt;
subplot(4,1,3)
plot(tc, yc)
title('c)')
xlabel('Time')
ylabel('Amplitude')

% d)
t = 0:dt:2;
xd = exp(t.^2);
nd = exp(2*t.^2);

yd = ConvFUNC(xd, nd) * dt;
td = 0:dt:(length(yd)-1)*dt;
subplot(4,1,4)
plot(td, yd)
title('d)')
xlabel('Time')
ylabel('Amplitude')
```

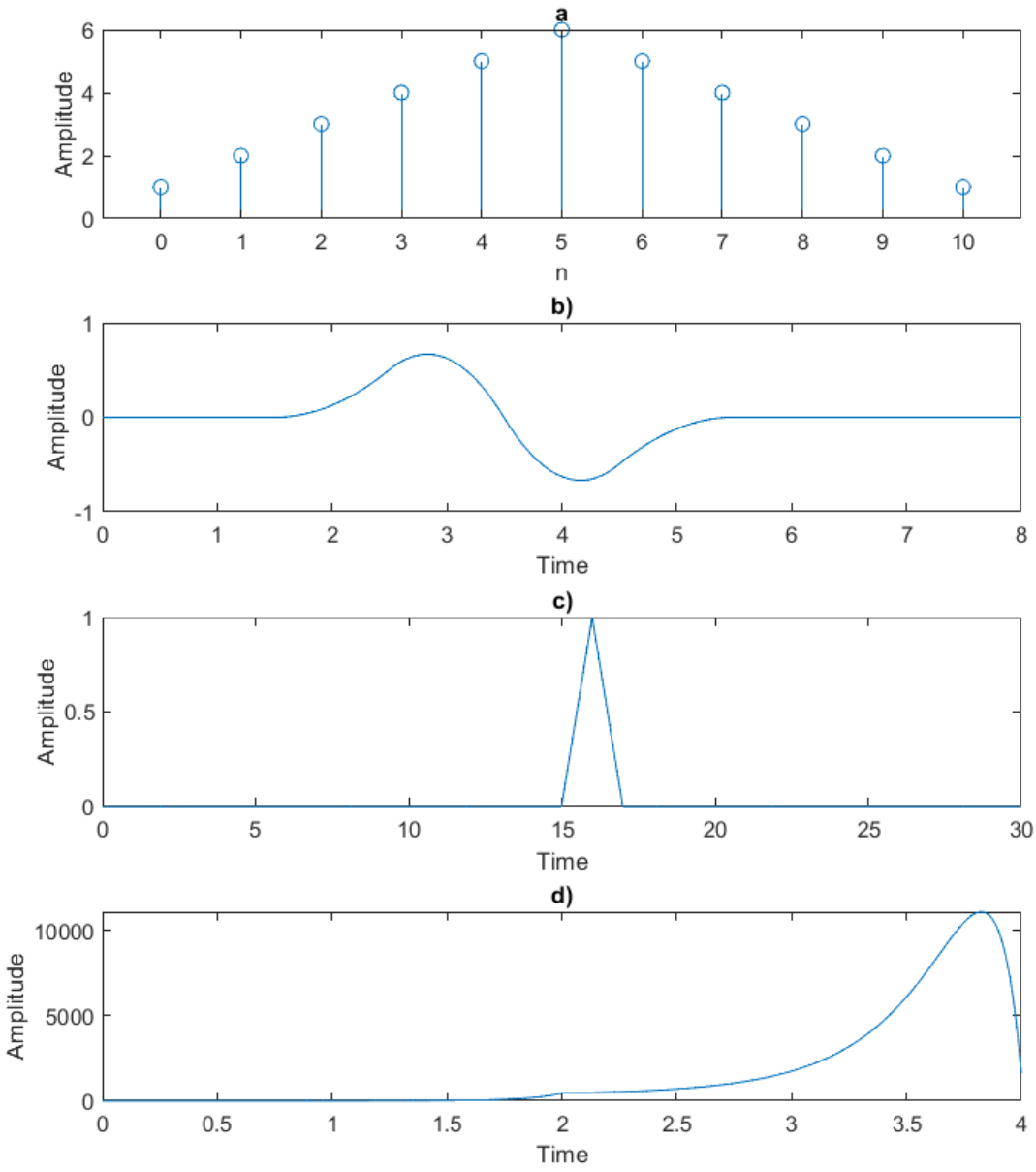


Figure.11: Graphs of a,b,c,d

Part 3

In this part, for better understanding, we developed an animation for convolution.

```
Si = 0.25;
t = -10:Si:10;

xi = double(t >= -5 & t < 5);
eta = double(t >= -2.5 & t < 2.5);

figure;
subplot(3,1,1); stem(t, xi, 'filled'); title('\xi[n]'); xlabel('t'); ylabel('Amplitude');
subplot(3,1,2); stem(t, eta, 'filled'); title('\eta[n]'); xlabel('t'); ylabel('Amplitude');

psi = conv(xi, eta) * Si;
t_psi = (min(t)+min(t)) : Si : (max(t)+max(t));

subplot(3,1,3); stem(t_psi, psi, 'filled');
title('Convolution \psi[n] = \xi[n]*\eta[n]'); xlabel('t'); ylabel('Amplitude');

eta_flipped = fliplr(eta);

figure;
Npsi = length(psi);
numSteps = Npsi;

for ii = 1:numSteps
    shift_amount = (ii - 1) * Si;
    t_eta_shifted = t + shift_amount;

    subplot(2,2,1);
    stem(t, xi, 'filled');
    title('\xi[n]');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t) max(t) + (numSteps-1)*Si]);

    subplot(2,2,2);
    stem(t, eta, 'filled');
    title('\eta[n] (orijinal)');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t) max(t) + (numSteps-1)*Si]);

    subplot(2,2,3);
    stem(t_eta_shifted, eta_flipped, 'filled');
    title('\eta[n] (Time reversed & flipped)');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t) max(t) + (numSteps-1)*Si]);

    subplot(2,2,4);
    stem(t_psi(1:ii), psi(1:ii), 'filled');
    title('Cumulative \psi[n]');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t_psi) max(t_psi)]);

    pause(0.5);
end
```

Part 4

In this part, we developed a basic speech detection algorithm using MATLAB. We recorded our ID numbers and then extracted each number. Most repeated number for me is 2 and then our algorithm detected 2's in the speech. As we seen before, while two same signals convolving each other, due to similarity of them, we got higher amplitudes. This property of the convolution operation allows us to detect similar signals.

```
[totalSignal, fs] = audioread('TotalNumber.flac');
t_total = (0:length(totalSignal)-1) / fs;

[n1Signal, fs_n1] = audioread('n1_extracted.flac');
t_n1 = (0:length(n1Signal)-1) / fs_n1;

n1_preprocessed = flip(conj(n1Signal));

psi = ConvFUNC(n1_preprocessed, totalSignal);
t_psi = (0:length(psi)-1) / fs;

figure;
subplot(2,1,1);
plot(t_total, totalSignal, 'b');
title('Total Speech Signal');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(2,1,2);
plot(t_n1, n1Signal, 'r');
title('Extracted n1 Signal');
xlabel('Time (s)');
ylabel('Amplitude');

figure;
plot(t_psi, abs(psi), 'k');
title('Magnitude of Cross-Correlation |\psi[n]|');
xlabel('Time (s)');
ylabel('|\psi[n]|');

figure;
subplot(2,1,1);
plot(t_psi, abs(psi).^2, 'm');
title('Magnitude Squared |\psi[n]|^2');
xlabel('Time (s)');
ylabel('|\psi[n]|^2');

subplot(2,1,2);
plot(t_psi, abs(psi).^4, 'g');
title('Magnitude to the Fourth |\psi[n]|^4');
xlabel('Time (s)');
ylabel('|\psi[n]|^4');

function [y] = ConvFUNC(x, h)
    x = x(:).';
    h = h(:).';
    Nx = length(x);
    Nh = length(h);
    Ny = Nx + Nh - 1;
    y = zeros(1, Ny);
    for n = 1:Ny
        k_min = max(1, n - Nh + 1);
        k_max = min(n, Nx);
        k = k_min:k_max;
        y(n) = sum( x(k) .* h(n - k + 1) );
    end
end
```

Those are the graphs for Google's voice:

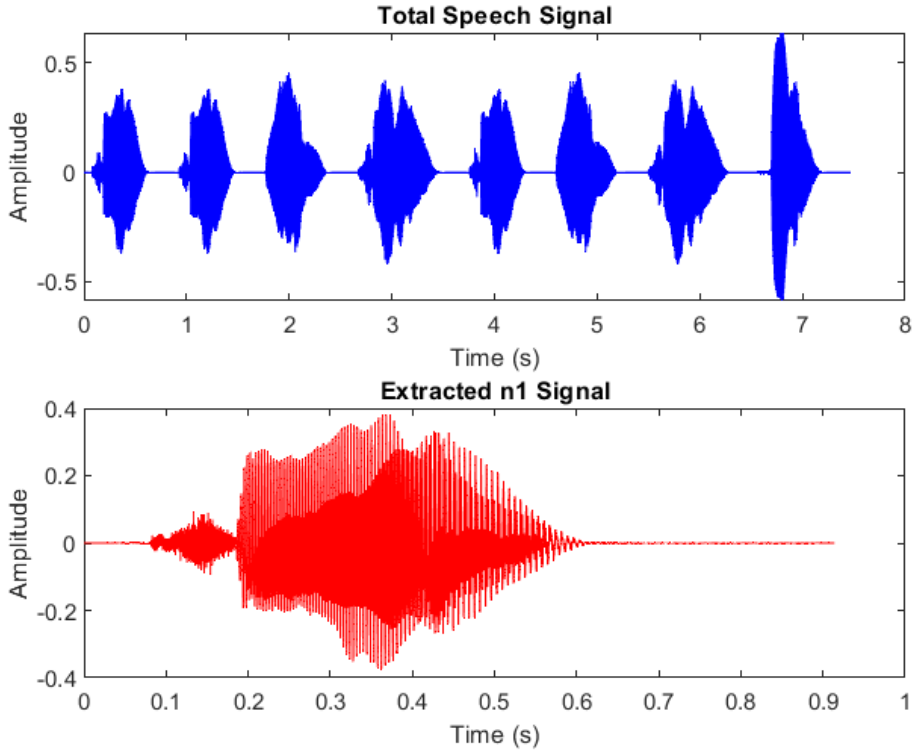


Figure.12: Speech signal and extracted n1 (Google)

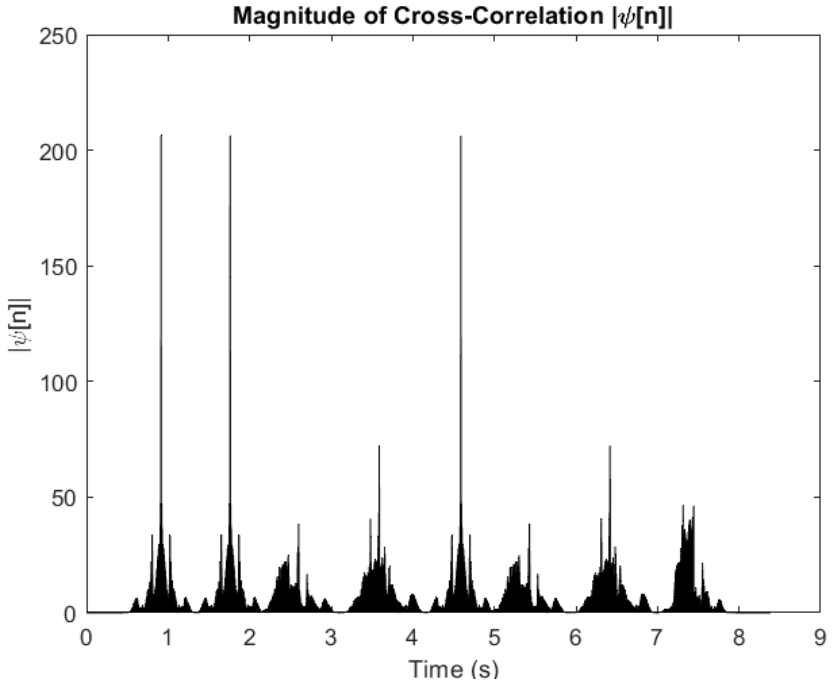


Figure.13: Detected n1 (Google)

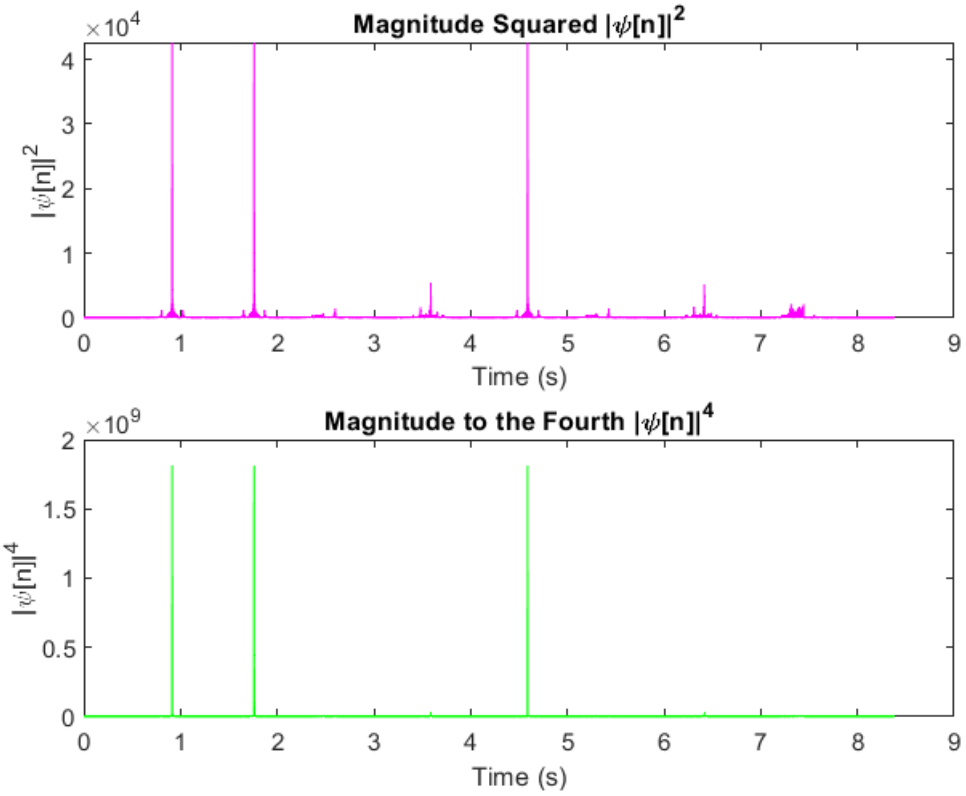


Figure.14: Magnitude of the squared and fourth (Google)

Those are the graphs for my own voice:

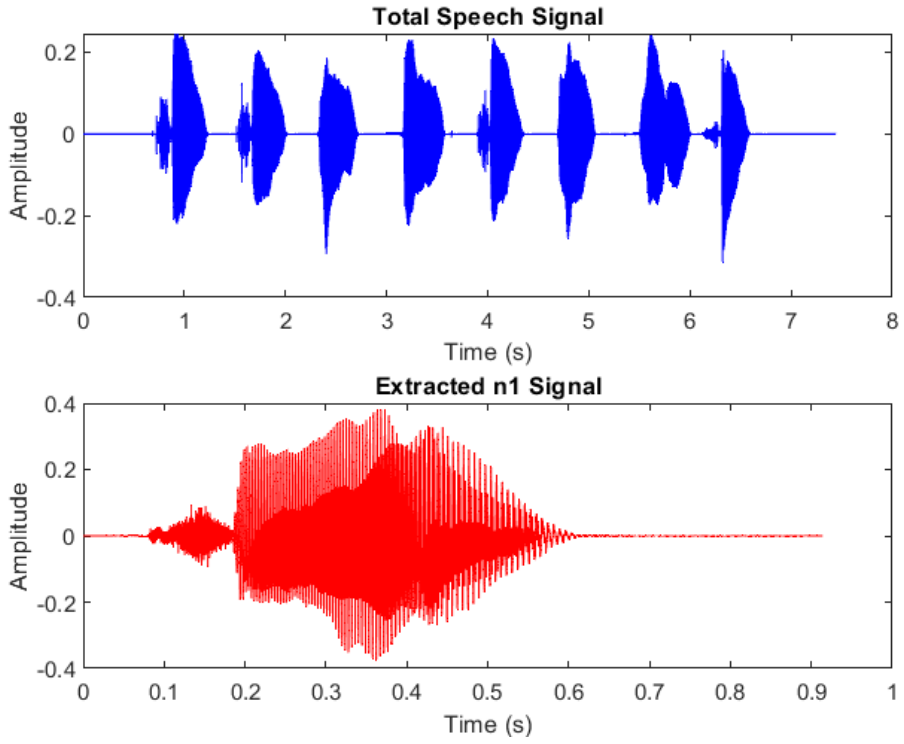


Figure.15: Speech signal and extracted n1 (My own)

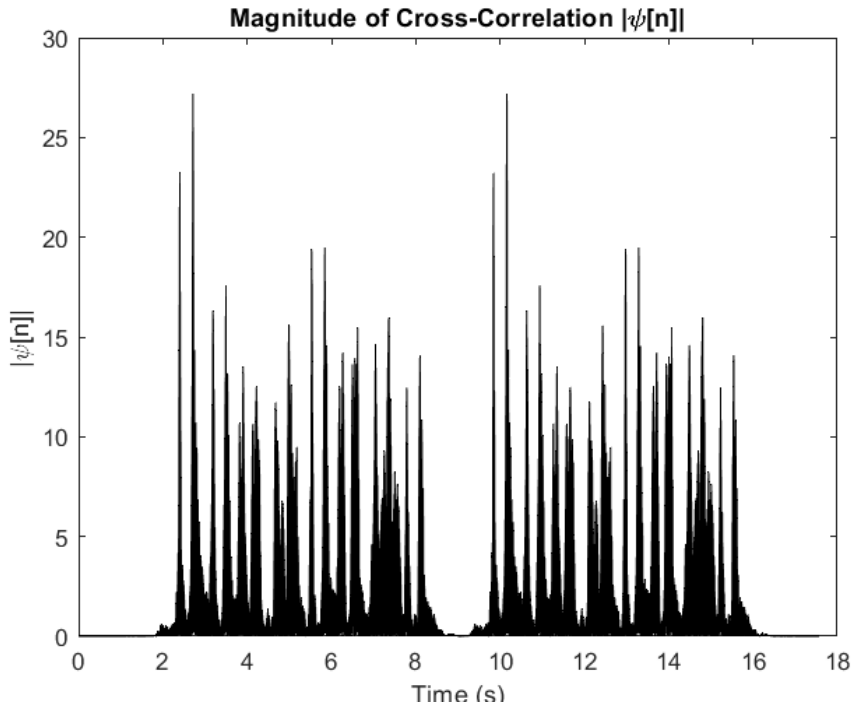


Figure.16: Detected n1 (My own)

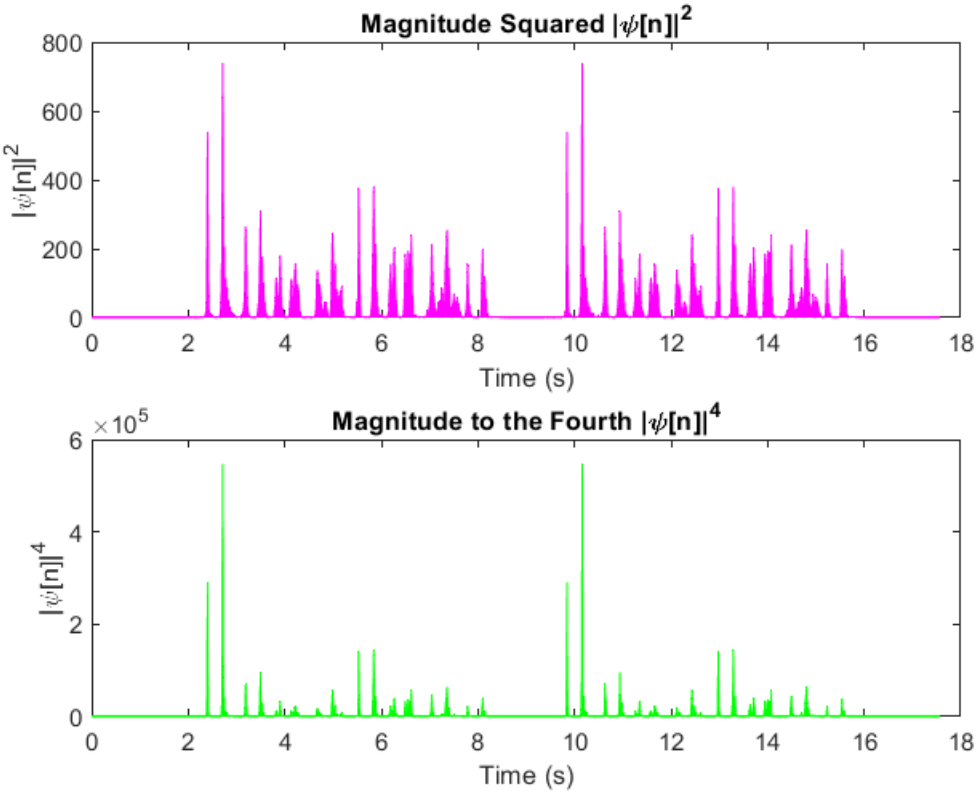


Figure.17: Magnitude of the squared and fourth (My own)

We can see that detecting my own voice is not that successful, but still we can somehow understand the 2's in my own voice.

Part 5

In this part, we generated a noise for our speech signals. To do this, we need the average power of a signal. The average power of the signal divided by average noise gives us SNR. We have different kinds of SNRs and generate a variety of noise signals for each SNR.

```
Si = 0.25;
t = -10:Si:10;

xi = double(t >= -5 & t < 5);
eta = double(t >= -2.5 & t < 2.5);

figure;
subplot(3,1,1); stem(t, xi, 'filled'); title('\xi[n]'); xlabel('t'); ylabel('Amplitude');
subplot(3,1,2); stem(t, eta, 'filled'); title('\eta[n]'); xlabel('t'); ylabel('Amplitude');

psi = conv(xi, eta) * Si;
t_psi = (min(t)+min(t)) : Si : (max(t)+max(t));

subplot(3,1,3); stem(t_psi, psi, 'filled');
title('Tam Konvolüyon \psi[n] = \xi[n]*\eta[n]'); xlabel('t'); ylabel('Amplitude');

eta_flipped = fliplr(eta);

figure;
Npsi = length(psi);
numSteps = Npsi;

for ii = 1:numSteps
    shift_amount = (ii - 1) * Si;
    t_eta_shifted = t + shift_amount;

    subplot(2,2,1);
    stem(t, xi, 'filled');
    title('\xi[n]');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t) max(t) + (numSteps-1)*Si]);

    subplot(2,2,2);
    stem(t, eta, 'filled');
    title('\eta[n] (orijinal)');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t) max(t) + (numSteps-1)*Si]);

    subplot(2,2,3);
    stem(t_eta_shifted, eta_flipped, 'filled');
    title('\eta[n] (ters çevrilmiş & kaydırılmış)');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t) max(t) + (numSteps-1)*Si]);

    subplot(2,2,4);
    stem(t_psi(1:ii), psi(1:ii), 'filled');
    title('Kümülatif \psi[n]');
    xlabel('t'); ylabel('Amplitude');
    xlim([min(t_psi) max(t_psi)]);

    pause(0.5);
end
```

As shown in figures, while SNR ratio increases, noise also increases. At 0.10 SNR, it was very hard to hear our speech

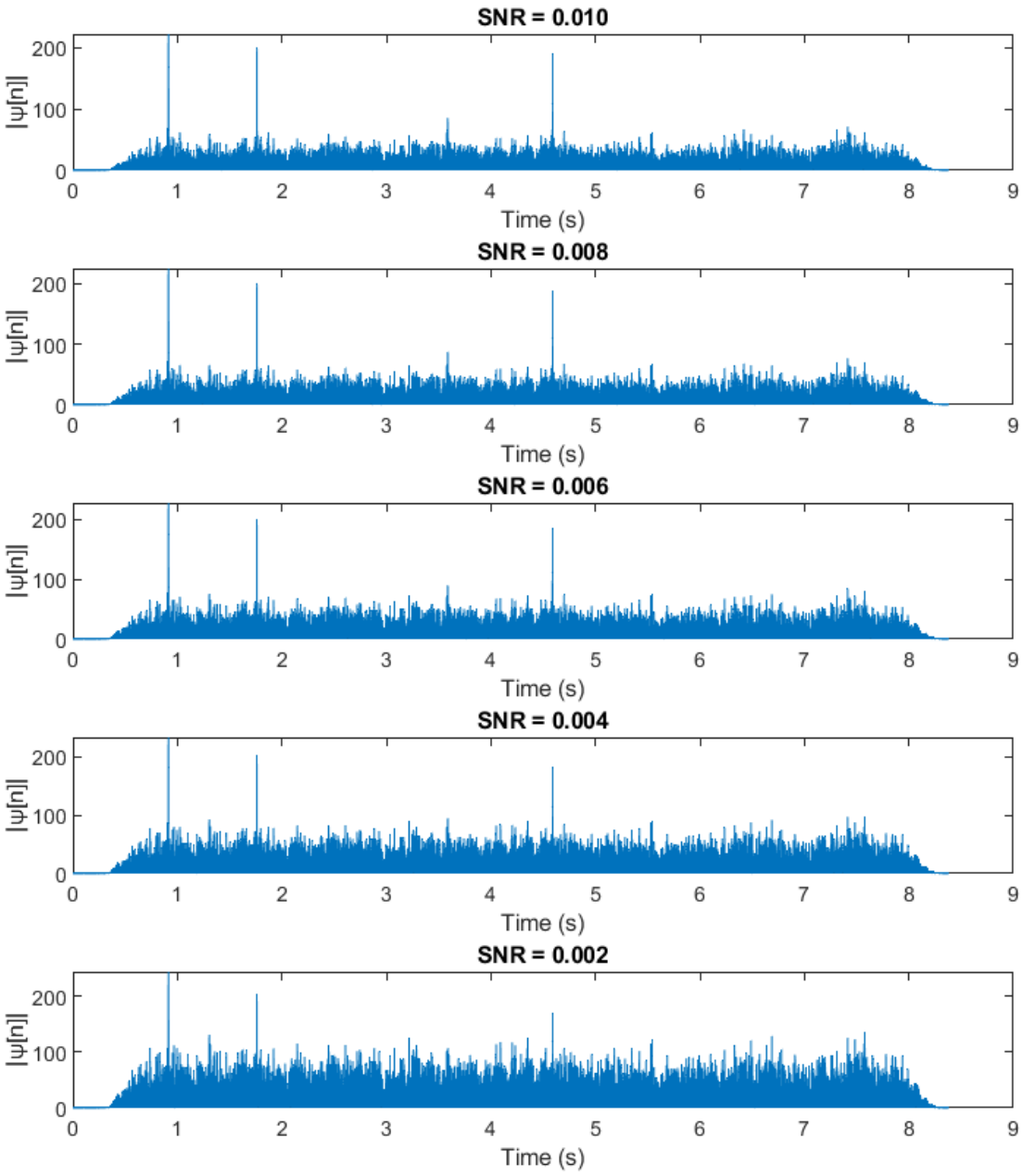


Figure.18: Different SNR and Noise - 1

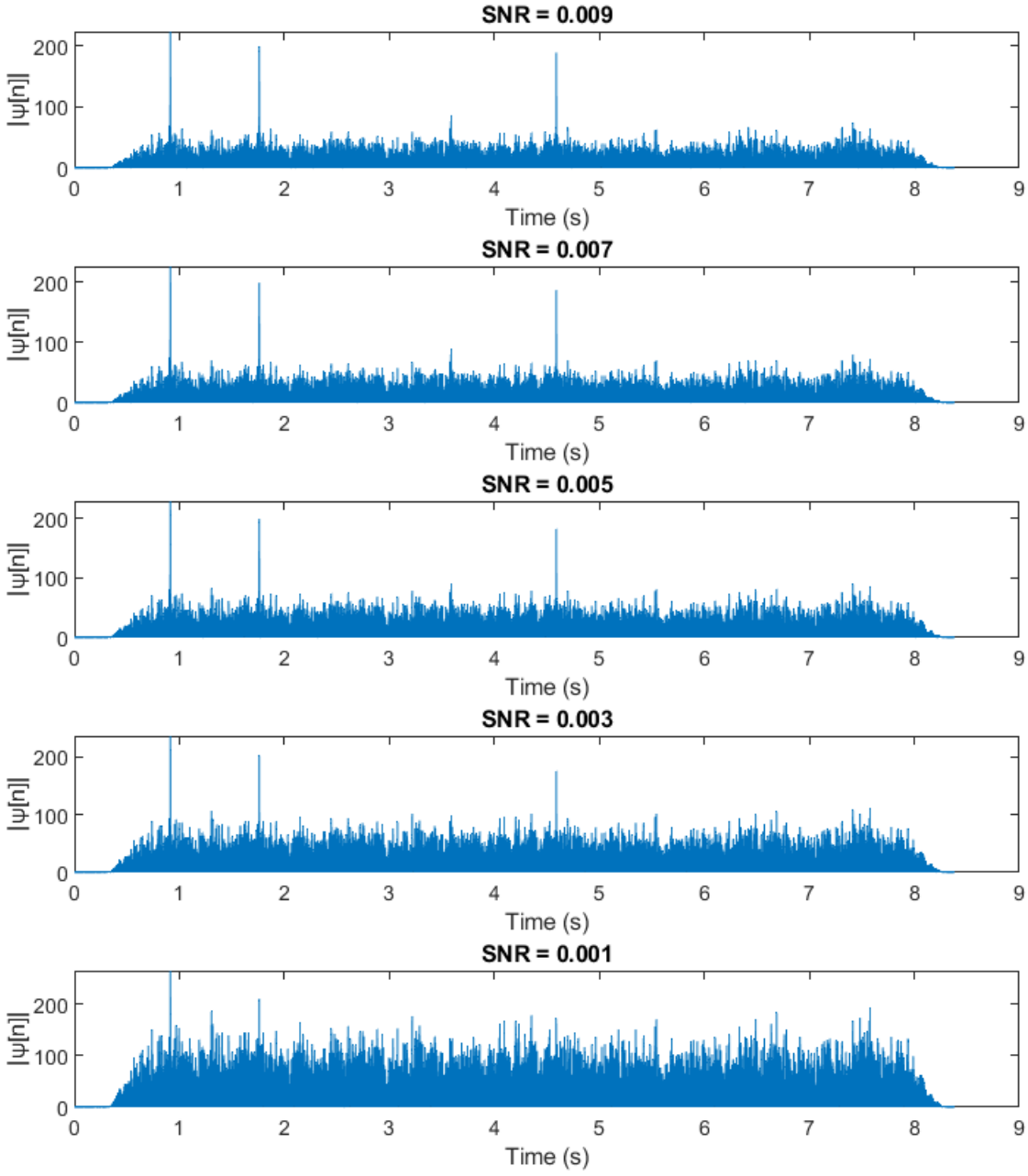


Figure.19: Different SNR and Noise - 2

Conclusion

This lab provided a hands-on exploration of convolution in signal processing. It began by examining the theory behind convolution, including how to determine the length of the resulting signal and drawing parallels with polynomial coefficients. Next, a MATLAB-based convolution function was implemented and tested using various input signals, which helped confirm the theoretical concepts with practical examples. An animation was also created to visually demonstrate the time-reversal and shifting effects inherent in convolution, making the dynamic process more intuitive. Additionally, a speech detection algorithm was developed to identify repeated digits in voice recordings, successfully detecting patterns in a Google voice sample, although with less accuracy in a personal recording. Lastly, the effect of noise was analyzed by generating signals with different SNR levels, highlighting how increased noise degrades signal clarity. Overall, the lab effectively showcased the fundamental role of convolution in both theoretical analysis and practical applications in signal processing.