**Furkan Büyüksarıkulak**

**22002097**

**26.02.2025**

**EEE 321**

**Signals and Systems**

**Spring 2024-2025**

**Lab Assignment 2**

In this lab task, we aimed to learn how to implement convolution and cross-correlation operations in signals. Then we will examine how the voice recognition algorithm works using these operators.

**Part 1**

**Part 1.1: Defining Convolution**

**Part 1.2: Evaluating Convolution**

**Note:** Since Part 1.1 and Part 1.2 were made on paper, large figures were placed starting from the next page for easier reading.

metin, el yazısı, doküman, belge, mektup, harf içeren bir resim

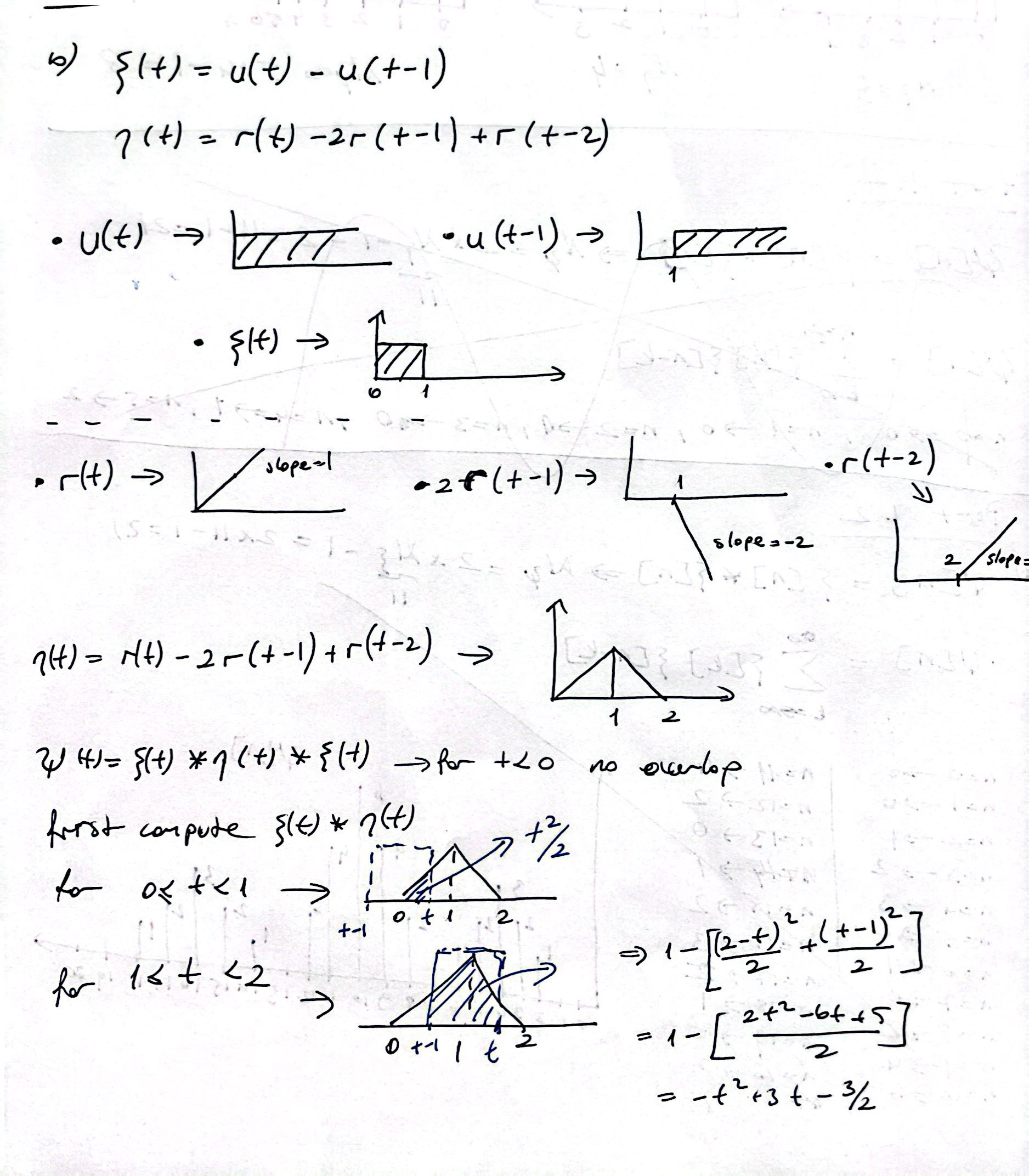
Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, el yazısı, yazı tipi, hat sanatı, kaligrafi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

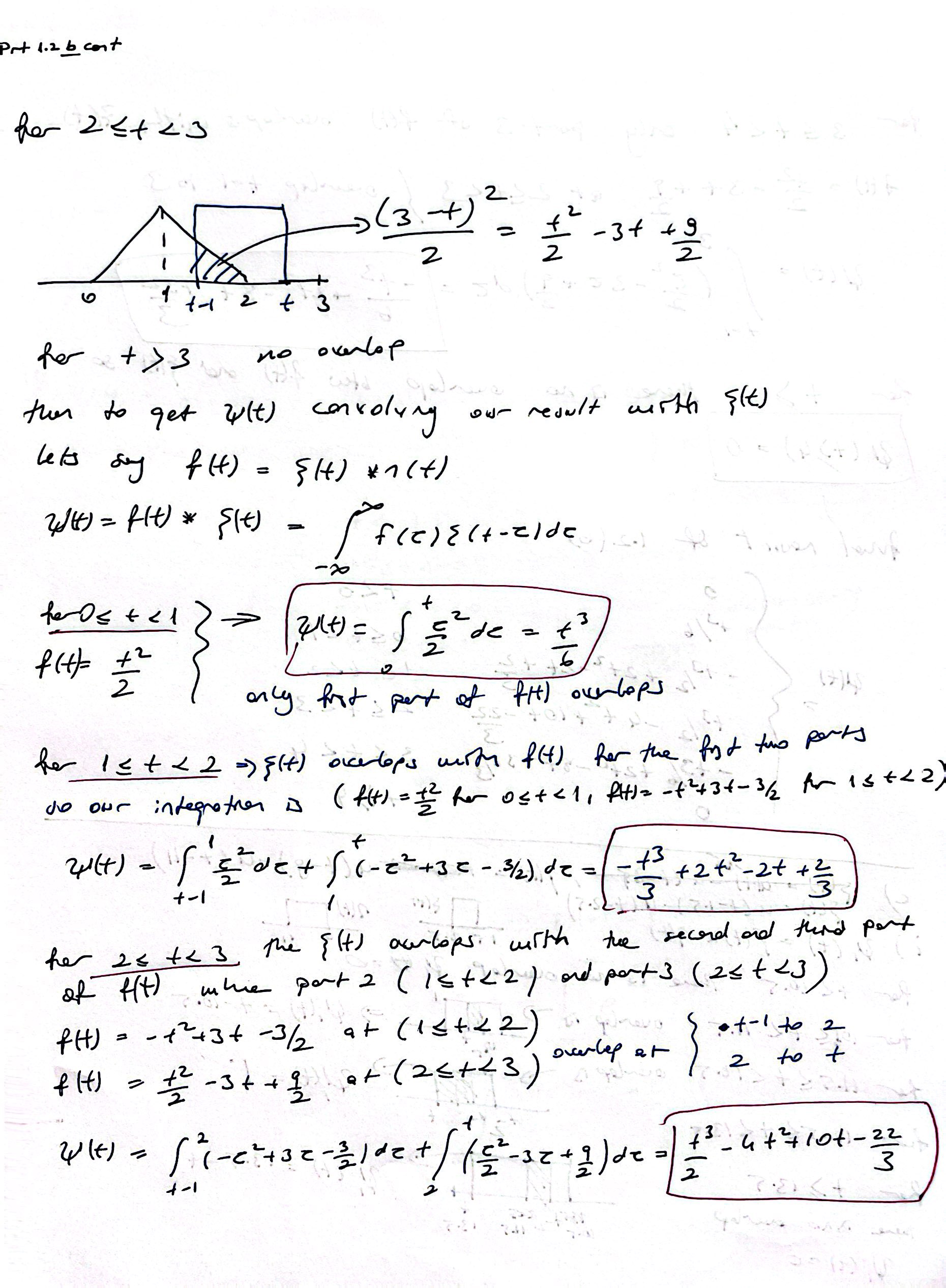
*figure 1 : Part 1.1*



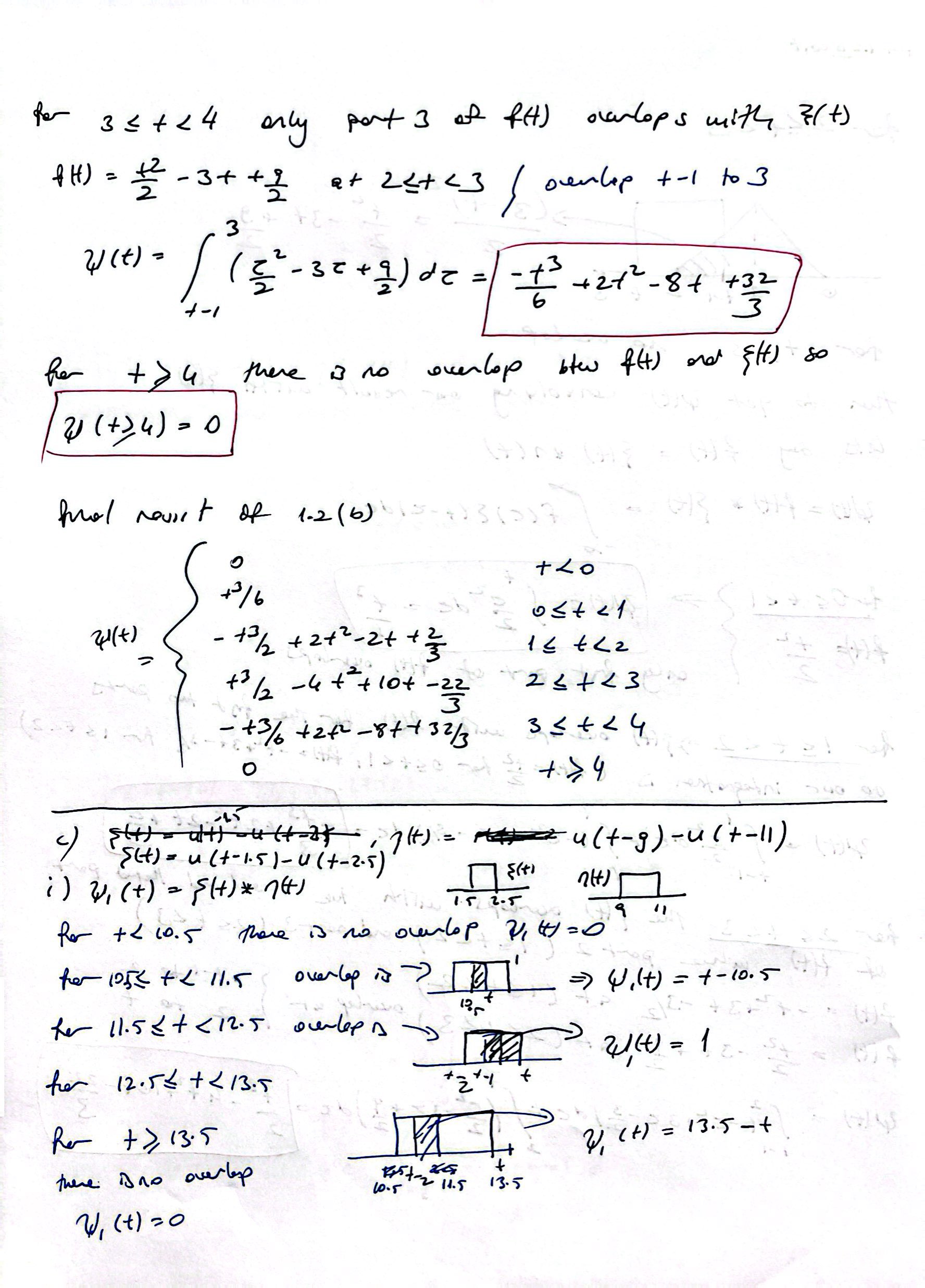
*figure 2 : Part 1.2 (a)*



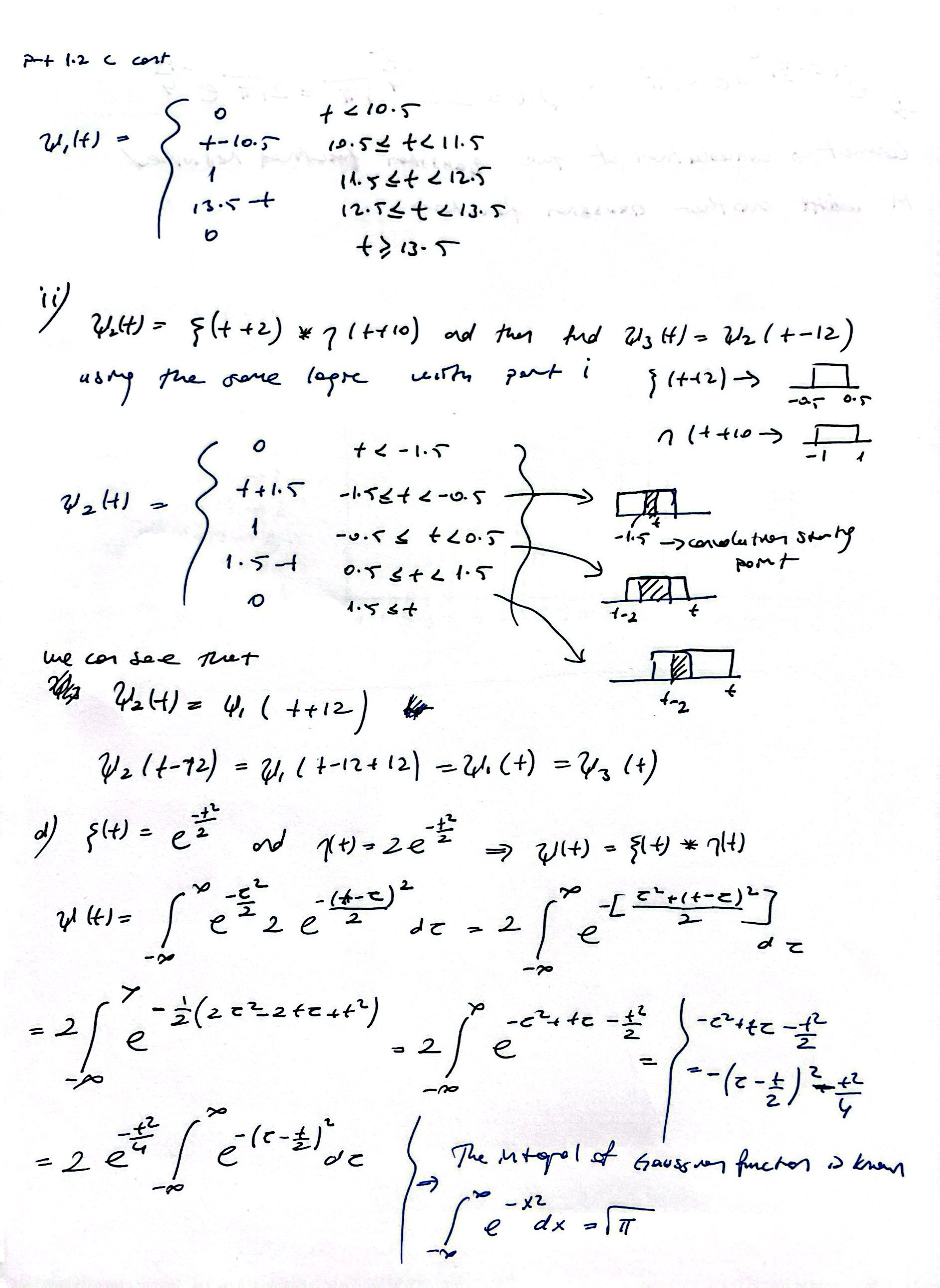
*figure 3 : Part 1.2 (b)*



*figure 4 : Part 1.2 (b) continiued*



*figure 5 : Part 1.2 (b-c)*



*figure 6 : Part 1.2 (c)*



*figure 7 : Part 1.2 (c) continiued*

**Part 2**

**Part 2.1: Implementing Convlution**

We turned the convolution operator we mentioned in Part 1 into a Matlab function. Using 1 for loop, we wrote a function that would give output by adding the points in the signal where the signal we flipped and shifted in accordance with the convolution rule and saved it as a separate script.

**Part 2.2: Testing the Convolution Function**

We tested the convolution function we wrote by implementing it on the signals we examined in part 1.2 and the result we obtained is as follows.

metin, çizgi, diyagram, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 8 : Part 2.2 convolution operation two same discrete signal*

**Part 3: Creating Convolution Animation**

In this section, we aimed to visually examine the convolution operator we learned. We implemented the given x(t) and h(t) CT signals via matlab. First, we tested the accuracy of these signals by plotting them. Then, we performed the convolution operation by calling the convFUNC operator we had written before. In order to turn this operation into an animation, we used a for loop in the tau range at a resolution of 0.25 and graphed the outputs we obtained as an animation. You can find some figures belonging to the created animation below. You can also access the gif version of the animation [**HERE**](https://drive.google.com/file/d/1wTdRC920326SqYOsdWasptWqw6782rUd/view?usp=sharing)

metin, diyagram, dikdörtgen, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, diyagram, dikdörtgen, plan içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.metin, diyagram, ekran görüntüsü, dikdörtgen içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 9 : Part 3 Some sections from the convolution animation*

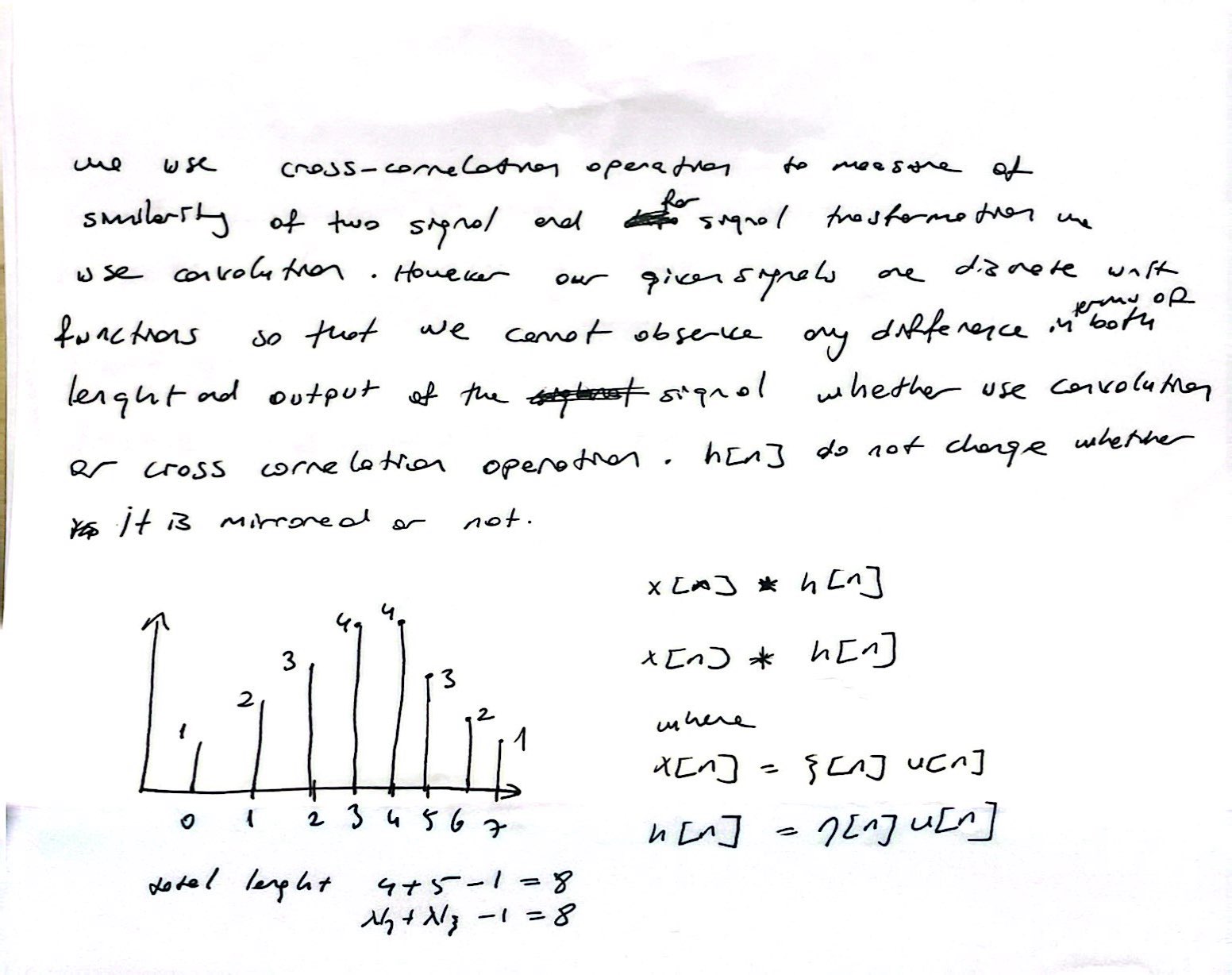
**Part 4:**

**Part 4.1 Defining Cross-Correlation**

metin, el yazısı, mektup, harf, doküman, belge içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 10 : Part 4 Cross Correlation Definition*

****

*figure 11 : Part 4 Cross Correlation Definition ( Continiued)*

**Part 4.2 : Building a Basic Speech Recognition Algorithm**

In this part, we created a voice recognition algorithm based on numbers. My ID number is 22002097 and when I followed the instructions and calculated, the numbers I got are as follows.

n1 = 2

n2 = 8

First, I learned how to record my own voice as a flac file. To do this, I recorded the voice using the audiorecorder and recordblocking functions in matlab in a different script file and saved the file with the audiowrite command. Then, using the given Python code, I downloaded the audio files corresponding to my ID from Google's text to speech library. After that, I recorded the n1 number ("2") we determined in a different flac. file by determining the appropriate interval within my own voice. My recorded voice and the n1 part in it look like this

metin, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 12 : Plot of My Recorded ID and its n1 part*

As we mentioned in Section 4.1, the cross-convolution operation was used to detect similarities between two signals. Using this logic, I put the two signals into the convFUNC function in order to find n1 in my own voice, but since this is a convolution function, I performed transpose and flip operations to convert it to cross-convolution. I plotted the 2nd power and 4th power of the output signal we obtained using subplot. The importance of the output power here is that it reduces the amplitude of the signals detected as less similar in the two signals and allows the signals we detect to be seen more clearly.

metin, çizgi, diyagram, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 13 : n1 cross convolution in recorded ID and exponents of output*

Here we observe the 3 ("2") sound waves that we are looking for in the first two graphs, but since the first one is not very strong, its amplitude is considerably reduced in the 4th power.

metin, çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 14 : n1 cross convolution text to speech n1 and TotalNumber and its exponents*

When we cross-correlate the n1 and TotalNumber obtained from Google, we observe that we get very clear signals.

However, when we do this operation with n2, we do not expect to see additional increases because n2 is not included in TotalNumber.

Now we will scan our own voice with TotalNumber and then with Google's text-to-speech n1.

metin, ekran görüntüsü, yazı tipi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 15 : n1 cross convolution text to speech n1 and recorded ID and its exponents*

metin, diyagram, ekran görüntüsü, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 16 : n1 cross convolution recorded n1 and TotalNumber and its exponents*

Probably because my own voice has multiple instabilities, the result I got was not clear enough, so the configuration where we use n1 and TotalNumber belonging to text to speech is more optimal.

**Part 5:**

**Part 5.1 Observing the Effects of SNR**

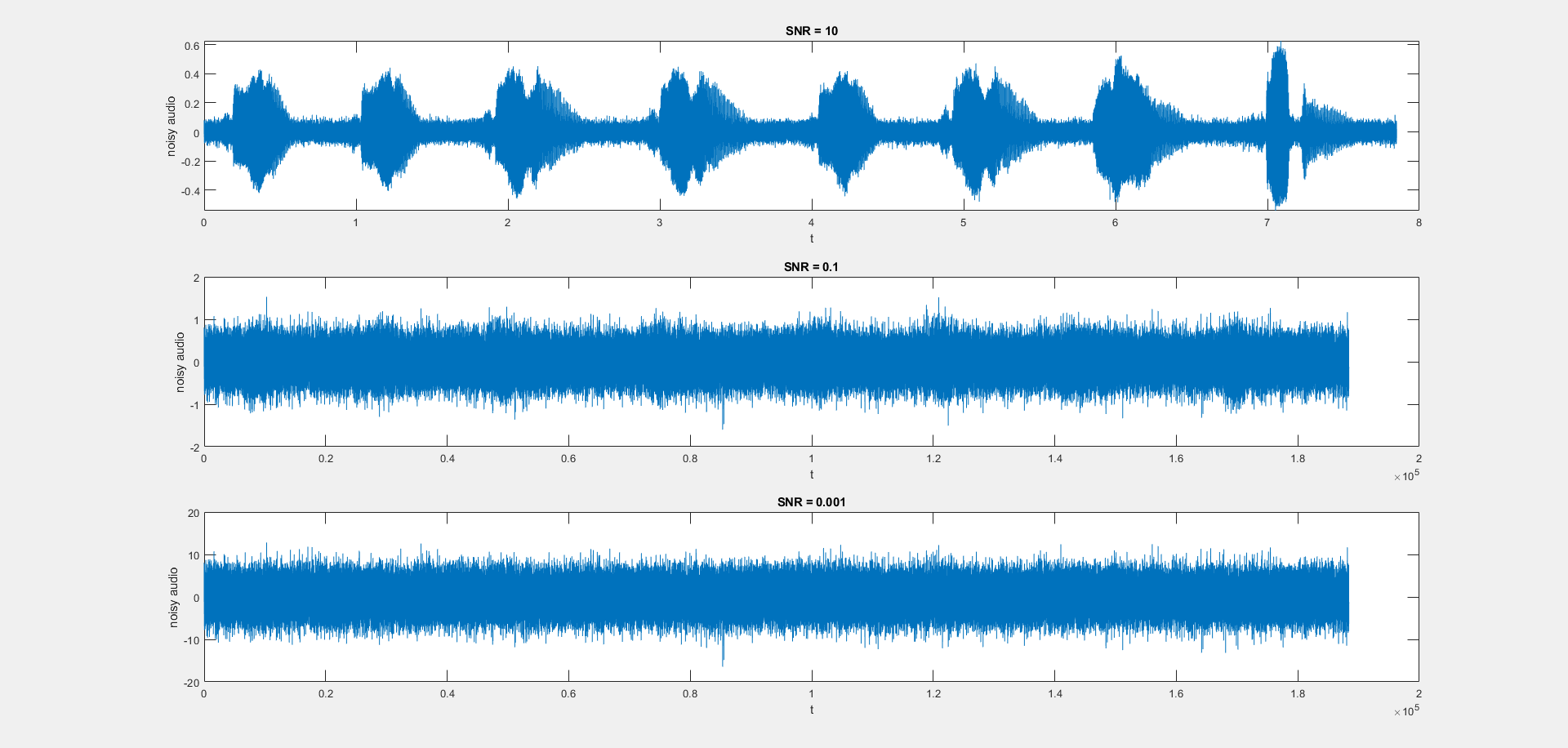
In this part we observe the effect of Signal to Noise Ratio on voice recognition. First we get the TotalNumber data as audio\_array and get its lenght as audio\_len to generate noise signal with the same lenght. Then we calculate power of signal and store as p-signal by using following calculation:

To generate noise we use additive white Gaussian noise (AWGN) To generate this in matlab we use folllowing code part and use previously calculated values audio\_len.



*figure 17 : AWGN noise signal generation in MATLAB*

Then we add this noise signal to our audio signal (“TotalNumber”), plot and listen theese signal. We repeat this process with different SNR values of (10, 0.1, 0.001). We can barely hear the numbers with the noise signal. We get least noise with 10 and the noise increase with decreasing SNR value, so most noise is at SNR value of 0.001. The corresponding plots of signals and power values of noises (“p\_noise”) as follows.



*figure 18 : Adding Noise signal TotalNumber and with respect to SNR values*

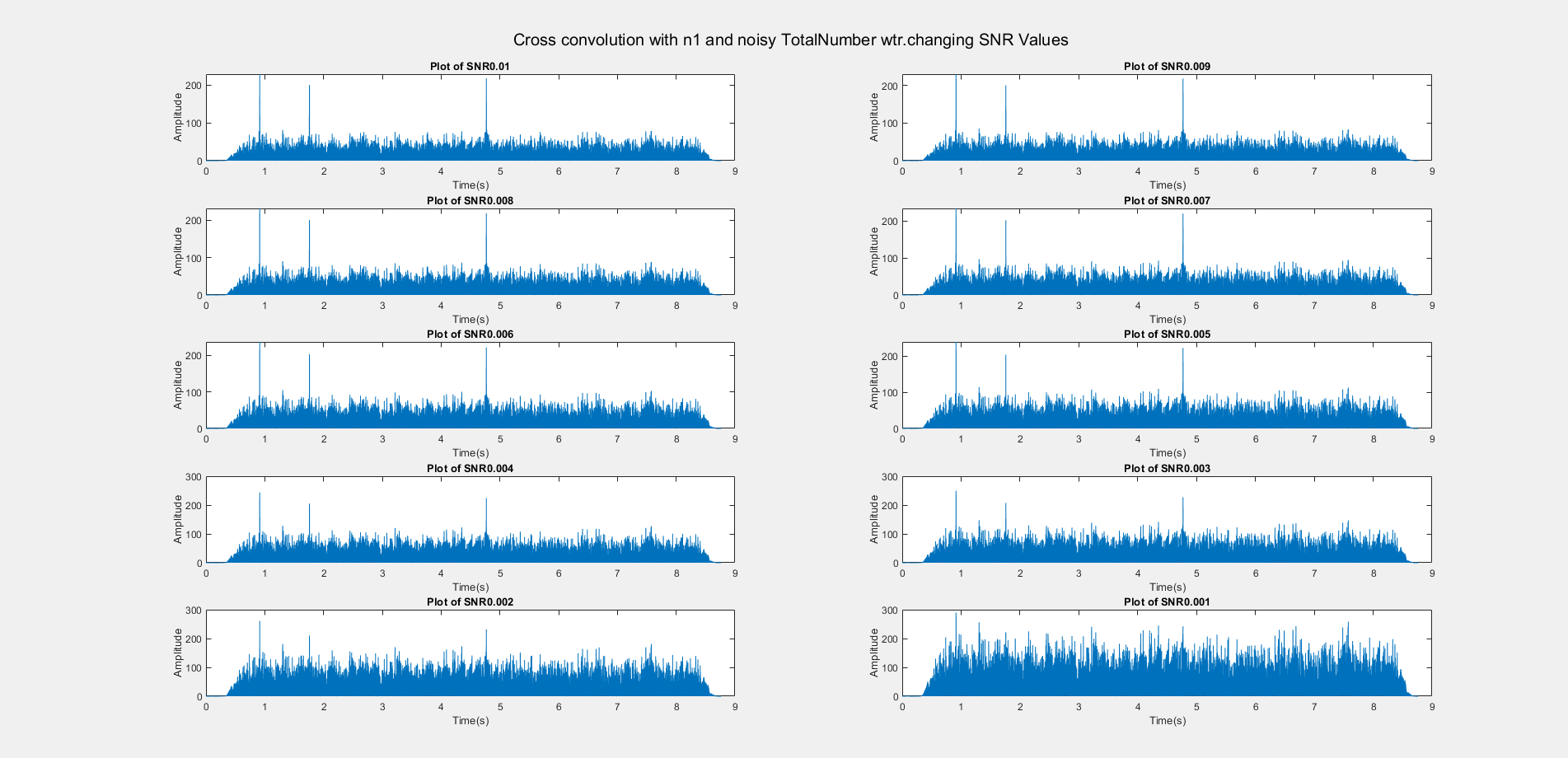
metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

*figure 19 : Power values of signal and noise signal with changing SNR values*

**Part 5.2 Detecting the SNR Limit**

In this part, we will try to detect similarity by using the cross-convolution operation on the noisy signal we created, as we did in part 4.2. Our aim here is to observe how effectively the voice recognition algorithm can work on noisy signals. We cross-convolved Google's n1 value and then created these graphs using for loop and subplot with changing SNR values. The resulting graph is as follows.



*figure 20 : Cross Convolution operation Google n1 and noisy TotalNumber with changing SNR values*

As far as we observe here, we can distinguish the n1 value from the graph for SNR values ​​between 0.01 and 0.002, but when the SNR value is 0.001, it becomes impossible to distinguish it.

**Conclusion**

In this lab task, we examined how convolution and cross-convolution operators are calculated and how they are used with signals. Using these operators, we developed a voice recognition algorithm over numbers and detected certain numbers over our own voice. We also created a noise signal, added noise to the voice we had and examined how this situation affected the voice recognition algorithm. Each stage specified in the lab task was completed successfully.

**Codes**

**Part 2.1: Implementing Convolution**

function [y] = convFUNC(x, h)

Nx = length(x);

Nh = length(h);

Ny = Nx + Nh - 1;

y = zeros(1, Ny);

flp\_h = fliplr(h); % Flip h for convolution operation

new\_h = [zeros(1, Nh-1), x, zeros(1, Nh-1)];

%perform convolution

for n = 1:Ny

y(n) = sum(new\_h(n : n+Nh-1) .\* flp\_h); %Compute the convolution by sum n's

end

end

**Part 2.2: Testing the Convolution Function**

%The signals

x = [0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0];

h = [0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0];

y=convFUNC(x,h); % call function

%plot the signal as discrete

sgtitle('Convolution Operation ξ[n]\*ξ[n]')

subplot(3,1,1)

stem(x);

title('x[n] = ξ[n]');

subplot(3,1,2)

stem(h);

title('h[n] = ξ[n]');

subplot(3,1,3)

stem(y);

title('y[n]');

**Part 3: Creating Convolution Animation**

filename = 'animation.gif'; % Name of the gif file

Si=0.25; %resolution

tlim = 10;

tau = -tlim : Si : tlim;

tau\_out = -20 : Si : 20;

Nt = length(tau);

%unit step function

u = @(t) double(t>=0);

%define sequences

x = @(t) u(t+5) - u(t-5); %x(t)

h = @(t) u(t+2.5) - u(t-2.5); %h(t)

x\_t = x(tau);

h\_t = h(tau);

y= convFUNC(x\_t,h\_t);

h\_t\_flip = h(-tau);

figure;

subplot(3,1,1);

plot(y);

title('Plot of y(t)');

xlabel('t');

ylabel('x(t)');

subplot(3,1,2);

plot(x\_t);

title('Plot of x(t)');

xlabel('t');

ylabel('x(t)');

subplot(3,1,3);

plot(h\_t);

title('Plot of h(x)');

xlabel('t');

ylabel('h(t)');

for ii = 1:length(tau)

% Update the plot

subplot(2, 2, 1);

plot(tau, x\_t, 'Color', "#0072BD", 'LineWidth', 1.5);

xlim([-tlim, tlim]); %tau limits

ylim([-0.25, 1.25]);

title('Plot of x(t)');

xlabel('t');

ylabel('x(t)');

subplot(2, 2, 2);

plot(tau, h\_t, 'Color', "#77AC30",'LineWidth', 1.5);

xlim([-tlim, tlim]);

ylim([-0.25, 1.25]);

title('Plot of h(t)');

xlabel('t');

ylabel('h(t)');

subplot(2, 2, 3);

plot(tau, x\_t, 'Color', "#0072BD", 'LineWidth', 1);

hold on;

plot(tau - 20 + ii/2, h\_t\_flip, 'Color', "#77AC30", 'LineWidth', 1.5);

hold off;

title('Plot of the Animation');

xlabel('t');

xlim([-20, 20])

ylabel('x(t),h(t)');

ylim([-0.5, 1.5]); % Set y-axis limits

subplot(2, 2, 4);

plot(tau\_out(1:4:2\*ii-1), y(1:4:2\*ii-1) / 4,'Color', "#A2142F", 'LineWidth', 1.5);

title('Plot of the Convolution');

xlabel('t');

ylabel('x(t)\*h(t)');

xlim([-20, 20]); % Set x-axis limits

ylim([-3, 6]);

% Capture the frame

frame = getframe(gcf);

im = frame2im(frame);

[imind, cm] = rgb2ind(im, 256); % Convert to indexed image

% Write to GIF file

if ii == 1

imwrite(imind, cm, filename, 'gif', 'Loopcount', inf, 'DelayTime', 0.05);

else

imwrite(imind, cm, filename, 'gif', 'WriteMode', 'append', 'DelayTime', 0.05);

end

end

**Part 4**

**Audio Recorder**

recObj = audiorecorder(8192, 16, 1);

recDuration = 10;

disp("Recording Start.")

recordblocking(recObj,recDuration);

disp("File ID\_record.flac recorded.")

play(recObj);

audioData = getaudiodata(recObj);

audiowrite('ID\_recordd.flac', audioData, 8192);

**Part 4.2: Building a Basic Speech Recognition Algorithm**

%variables

y = [0,1,2,3,4,5,6,7,8,9];

ID = 22002097;

p = [2,0];

lamda = [1,3,4,5,6,8];

%lenghts

Np = length(p);

Nlamda = length(lamda);

delta = mod(ID,7);

deltap = mod(delta,Np)+1;

deltalamda = mod(delta,Nlamda)+1;

%n1 and n2

n1 = 2;

n2 = 8;

f\_s= 8192;

bitsample = 16;

channel = 1;

figure;

%---my recorded ID and its n1 part

sgtitle('my recorded ID and its n1 part');

[ID, fs] = audioread("ID\_record.flac");

%t = 0:length(ID) - 1 / fs;

subplot(2, 1, 1);

title("mine ID");

%soundsc(ID,fs)

Ts2 = 1/f\_s;

Temptau2 = 1:length(ID);

t\_2 = Temptau2 \* Ts2;

plot(t\_2,ID);

%----set boundaries of n1 in recorded ID

t0 =find(t\_2 >= 1.7, 1);

t1 =find(t\_2<= 2.25,1,'last');

% Extract the instance into another .flac file

mine\_n1 = ID(t0:t1);

audiowrite('n1\_mineee.flac', mine\_n1, f\_s);

%---

soundsc(mine\_n1, f\_s);

subplot(2,1,2);

t\_1 = (0: length(mine\_n1)-1) \* (1/f\_s);

plot(t\_1, mine\_n1);

title("N1 part");

%---get TotalNumber and googles n1 and n2

figure;

[n1\_bot\_2, fs\_1] = audioread("2.flac");

[n2\_bot, fs\_2] = audioread("8.flac" );

[bot, FS] = audioread("TotalNumber.flac");

%soundsc(bot,FS)

Ts1 = 1/24000;

Temptau1 = 1:length(bot);

tau1 = Temptau1 \* Ts1;

plot(tau1,bot);

figure;

%cross concolution transformation of convolution

flp\_mine\_n1 =fliplr (transpose(mine\_n1));

out\_1 = convFUNC(transpose(ID), flp\_mine\_n1);

%n1 cross convolution in recorded ID, and exponents of the output

sgtitle('n1 cross convolution in recorded ID, and exponents of the output');

t\_5 = (0: length(out\_1)-1) \* (1/f\_s);

subplot(3,1,1);

plot(t\_5,abs(out\_1));

ylabel("\psi [x]");

xlabel("x");

subplot(3,1,2);

plot(t\_5,abs(out\_1.^2));

ylabel("\psi [x]^2");

xlabel("x")

subplot(3,1,3);

plot(t\_5,abs(out\_1.^4))

ylabel("\psi [x]^4");

xlabel("x");

%convolution of n2 with TotalNumber

flp\_n2\_bot = fliplr(transpose(n2\_bot));

out\_3 = convFUNC(transpose(bot), flp\_n2\_bot);

% Plot of text-to-speech n1 in recorded ID

figure;

%cross concolution transformation of convolution

flp\_n1\_bot\_2 = fliplr(transpose(n1\_bot\_2));

out\_4 = convFUNC(transpose(ID), flp\_n1\_bot\_2);

sgtitle('text-to-speech n1 in recorded ID');

t\_8 = (0: length(out\_4)-1) \* (1/fs);

subplot(3,1,1);

plot(t\_8,abs(out\_4));

ylabel("\psi [x]");

xlabel("x");

subplot(3,1,2);

plot(t\_8,abs(out\_4.^2));

ylabel("\psi [x]^2");

xlabel("x")

subplot(3,1,3);

plot(t\_8,abs(out\_4.^4))

ylabel("\psi [x]^4");

xlabel("x");

%Plot of Recorded n1 in TotalNumber

figure;

%cross concolution transformation of convolution

flp\_mine\_n1 = fliplr(transpose(n1\_bot\_2));

out\_5 = convFUNC(transpose(bot), flp\_mine\_n1);

sgtitle('Recorded n1 in TotalNumber');

t\_9 = (0: length(out\_5)-1) \* (1/FS);

subplot(3,1,1);

plot(t\_9,abs(out\_5));

ylabel("\psi [x]");

xlabel("x");

subplot(3,1,2);

plot(t\_9,abs(out\_5.^2));

ylabel("\psi [x]^2");

xlabel("x")

subplot(3,1,3);

plot(t\_9,abs(out\_5.^4))

ylabel("\psi [x]^4");

xlabel("x");

%Plot of Text to speech in n1 TotalNumber

figure;

%cross concolution transformation of convolution

flp\_n1\_bot\_2 = fliplr(transpose(n1\_bot\_2));

out\_6 = convFUNC(transpose(bot), flp\_n1\_bot\_2);

sgtitle('Text to speech in n1 TotalNumber');

t\_9 = (0: length(out\_6)-1) \* (1/FS);

subplot(3,1,1);

plot(t\_9,abs(out\_6));

ylabel("\psi [x]");

xlabel("x");

subplot(3,1,2);

plot(t\_9,abs(out\_6.^2));

ylabel("\psi [x]^2");

xlabel("x")

subplot(3,1,3);

plot(t\_9,abs(out\_6.^4))

ylabel("\psi [x]^4");

xlabel("x");

**Part 5**

**Part 5.1: Observing the Effects of SNR**

[audio\_array, FS] = audioread("TotalNumber.flac" );

t\_3 = (0: length(audio\_array)-1) \* (1/FS);

%soundsc(audio\_array, FS);

audio\_len = length(audio\_array);

%power calculations

p\_signal = (1/audio\_len) \* sum(audio\_array.^2);

p\_noise\_SNR10 = p\_signal / 10;

display(p\_signal);

display(p\_noise\_SNR10);

disp(p\_signal);

rng (5)

awgn = sqrt ( p\_noise\_SNR10 ) .\* randn ([ audio\_len , 1]) ;

noisy\_audio = audio\_array + awgn;

%soundsc(noisy\_audio, FS); pause(10);

%SNR = 10

subplot(3, 1, 1);

plot(t\_3, noisy\_audio);

title("SNR = 10");

xlabel("t");

ylabel("noisy audio");

%SNR = 0.1

p\_noise\_SNR01 = p\_signal / 0.1;

display(p\_noise\_SNR01);

rng(5);

awgn\_2 = sqrt(p\_noise\_SNR01) .\* randn([audio\_len, 1]);

noisy\_audio\_2 = audio\_array + awgn\_2;

soundsc(noisy\_audio\_2, FS); pause(10);

subplot(3, 1, 2);

plot(noisy\_audio\_2);

title("SNR = 0.1");

xlabel("t");

ylabel("noisy audio");

%SNR = 0.001

p\_noise\_SNR0001 = p\_signal / 0.001;

display(p\_noise\_SNR0001);

rng(5);

awgn\_3 = sqrt(p\_noise\_SNR0001) .\* randn([audio\_len, 1]);

noisy\_audio\_3 = audio\_array + awgn\_3;

soundsc(noisy\_audio\_3, FS);

subplot(3, 1, 3);

plot(noisy\_audio\_3);

title("SNR = 0.001");

xlabel("t");

ylabel("noisy audio");

**Part 5.2: Detecting the SNR Limit**

[audio\_array,FS] = audioread("TotalNumber.flac");

addpath('C:\Users\furka\Desktop\Sinyal\LAB2');

[fltr, f\_fltr] = audioread("2.flac");

audio\_len = length(audio\_array);

p\_signal = 2/audio\_len\*sum(audio\_array.^2);

disp(p\_signal);

figure;

%for loop for SNR dependent plot

for i = 1:10

snr = 0.011-0.001\*i;

p\_noise = p\_signal / snr;

rng(5)

awgn = sqrt(p\_noise) .\* randn([audio\_len, 1]);

noisy\_audio = awgn + audio\_array;

%cross-convolution operation

filterr = convFUNC(transpose(noisy\_audio), fliplr(transpose(fltr)));

t = (0: length(filterr)-1) \* 1/FS;

sgtitle('Cross convolution with n1 and noisy TotalNumber wtr.changing SNR Values');

subplot(5,2,i);

plot(t, abs(filterr));

title(['Plot of SNR',num2str(snr)]);

xlabel('Time(s)');

ylabel('Amplitude');

end