

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.

Part 1.1

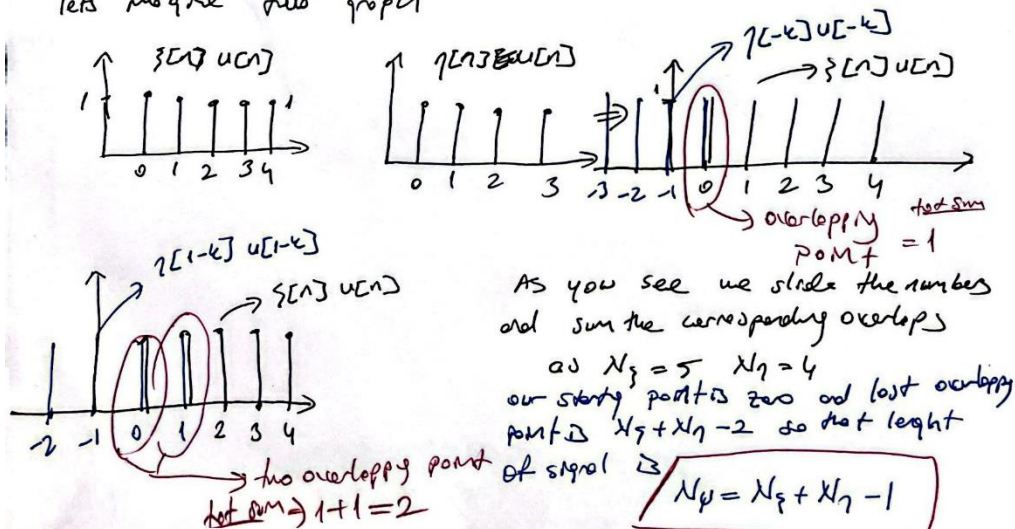
$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau) h(t-\tau) d\tau$$

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k] h[n-k]$$

$\{x[n]\}$ and $\{h[n]\}$ are length N_x and N_h

$$y[n] = (\{x[n]\} u[n]) * (\{h[n]\} u[n])$$

$\{x[n]\} u[n]$ and $\{h[n]\} u[n]$ signals are multiplied by unit step function, namely there are defined starting point of zero ("0") and to the positive time axis. Discrete convolution means, imagine we have two group of numbers and using one group to scan another. You can think this like sliding one group of number onto another. If two number overlapping this means we multiply them. Then take the sum of the numbers, this gives us to discrete convolution. As we said our starting point is $n=0$ lets imagine two graphs



lets show this graph on to the graph of final signal

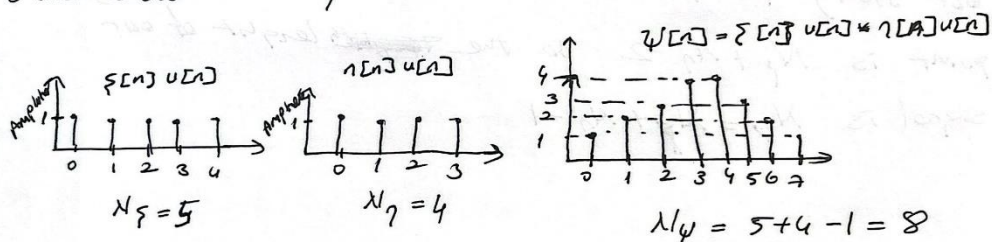


figure 1 : Part 1.1

Part 1.2

a) $\psi[n] = f[n] * f[n] \Rightarrow N_\psi = 2 \times \underbrace{N_f}_{11} - 1 = 2 \times 11 - 1 = 21$

$$\psi[n] = \sum_{k=0}^{21} f[k] f[n-k]$$

$n=0 \rightarrow 0$	$n=11 \rightarrow 4$
$n=1 \rightarrow 0$	$n=12 \rightarrow 6$
$n=2 \rightarrow 0$	$n=13 \rightarrow 4$
$n=3 \rightarrow 0$	$n=14 \rightarrow 2$
$n=4 \rightarrow 1$	$n=15 \rightarrow 0$
$n=5 \rightarrow 2$	$n=16 \rightarrow 1$
$n=6 \rightarrow 3$	$n=17 \rightarrow 2$
$n=7 \rightarrow 2$	$n=18 \rightarrow 3$
$n=8 \rightarrow 1$	$n=19 \rightarrow 2$
$n=9 \rightarrow 0$	$n=20 \rightarrow 1$
$n=10 \rightarrow 2$	$n=21 \rightarrow 0$

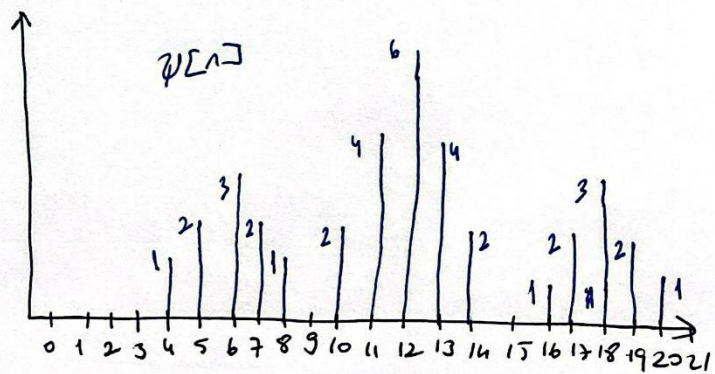
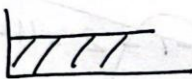
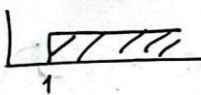


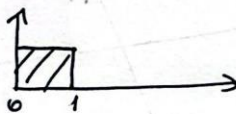
figure 2 : Part 1.2 (a)

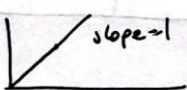
$$b) \quad \xi(t) = u(t) - u(t-1)$$

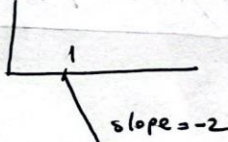
$$\eta(t) = r(t) - 2r(t-1) + r(t-2)$$

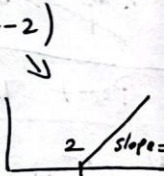
$$\bullet u(t) \rightarrow$$


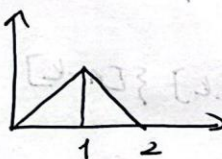
$$\bullet u(t-1) \rightarrow$$


$$\bullet \xi(t) \rightarrow$$


$$\bullet r(t) \rightarrow$$


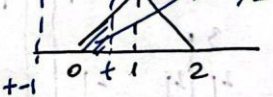
$$\bullet 2r(t-1) \rightarrow$$


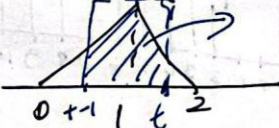
$$\bullet r(t-2) \rightarrow$$


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


$$\psi(t) = \xi(t) * \eta(t) * \xi(t) \rightarrow \text{for } t < 0 \text{ no overlap}$$

$$\text{first compute } \xi(t) * \eta(t)$$

$$\text{for } 0 \leq t < 1 \rightarrow$$


$$\text{for } 1 \leq t < 2 \rightarrow$$


$$\Rightarrow 1 - \left[\frac{(2-t)^2}{2} + \frac{(t-1)^2}{2} \right]$$

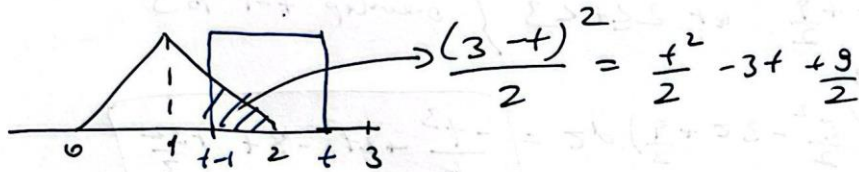
$$= 1 - \left[\frac{2t^2 - 6t + 5}{2} \right]$$

$$= -t^2 + 3t - \frac{3}{2}$$

figure 3 : Part 1.2 (b)

Part 1.2 b cont

for $2 \leq t < 3$



for $t > 3$ no overlap

then to get $\psi(t)$ convolving our result with $\xi(t)$

lets say $f(t) = \xi(t) * \gamma(t)$

$$\psi(t) = f(t) * \xi(t) = \int_{-\infty}^{\infty} f(\tau) \xi(t-\tau) d\tau$$

for $0 \leq t < 1$ } $\Rightarrow \psi(t) = \int_0^t \frac{\tau^2}{2} d\tau = \frac{t^3}{6}$
 $f(t) = \frac{t^2}{2}$ } only first part of $f(t)$ overlaps

for $1 \leq t < 2 \Rightarrow \xi(t)$ overlaps with $f(t)$ for the first two parts
 do our integration Δ ($f(t) = \frac{t^2}{2}$ for $0 \leq t < 1$, $f(t) = -t^2 + 3t - 3/2$ for $1 \leq t < 2$)

$$\psi(t) = \int_{t-1}^1 \frac{\tau^2}{2} d\tau + \int_1^t (-\tau^2 + 3\tau - 3/2) d\tau = \left[-\frac{t^3}{3} + 2t^2 - 2t + \frac{2}{3} \right]$$

for $2 \leq t < 3$ the $\xi(t)$ overlaps with the second and third part
 of $f(t)$ where part 2 ($1 \leq t < 2$) and part 3 ($2 \leq t < 3$)

$$f(t) = -t^2 + 3t - 3/2 \text{ at } (1 \leq t < 2) \quad \text{overlap at } \begin{cases} t-1 \text{ to } 2 \\ 2 \text{ to } t \end{cases}$$

$$f(t) = \frac{t^2}{2} - 3t + \frac{9}{2} \text{ at } (2 \leq t < 3)$$

$$\psi(t) = \int_{t-1}^2 (-\tau^2 + 3\tau - 3/2) d\tau + \int_2^t \left(\frac{\tau^2}{2} - 3\tau + \frac{9}{2} \right) d\tau = \left[\frac{t^3}{2} - 4t^2 + 10t - \frac{22}{3} \right]$$

figure 4 : Part 1.2 (b) continued

for $3 \leq t < 4$ only part 3 of $f(t)$ overlaps with $g(t)$

$$f(t) = \frac{t^2}{2} - 3t + \frac{9}{2} \quad \text{at } 2 \leq t < 3 \quad \left\{ \text{overlap } t=1 \text{ to } 3 \right.$$

$$\psi(t) = \int_{t-1}^3 \left(\frac{\tau^2}{2} - 3\tau + \frac{9}{2} \right) d\tau = \boxed{-\frac{t^3}{6} + 2t^2 - 8t + \frac{32}{3}}$$

for $t \geq 4$ there is no overlap btw $f(t)$ and $g(t)$ so

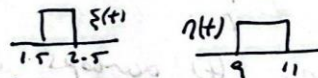
$$\boxed{\psi(t \geq 4) = 0}$$

final result of 1.2(b)

$$\psi(t) = \begin{cases} 0 & t < 0 \\ t^3/6 & 0 \leq t < 1 \\ -t^3/2 + 2t^2 - 2t + \frac{2}{3} & 1 \leq t < 2 \\ t^3/2 - 4t^2 + 10t - \frac{22}{3} & 2 \leq t < 3 \\ -t^3/6 + 2t^2 - 8t + \frac{32}{3} & 3 \leq t < 4 \\ 0 & t \geq 4 \end{cases}$$

c) $\xi(t) = u(t) - u(t-2)$, $\eta(t) = u(t-9) - u(t-11)$
 $\xi(t) = u(t-1.5) - u(t-2.5)$

i) $\psi_1(t) = \xi(t) * \eta(t)$



for $t < 10.5$ there is no overlap $\psi_1(t) = 0$

for $10.5 \leq t < 11.5$ overlap is $\Rightarrow \psi_1(t) = t - 10.5$

for $11.5 \leq t < 12.5$ overlap is $\Rightarrow \psi_1(t) = 1$

for $12.5 \leq t < 13.5$

for $t \geq 13.5$

there is no overlap

$$\psi_1(t) = 0$$

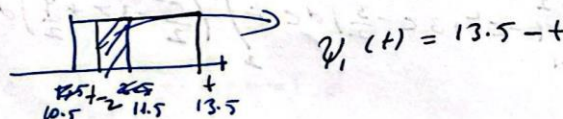
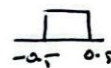
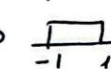


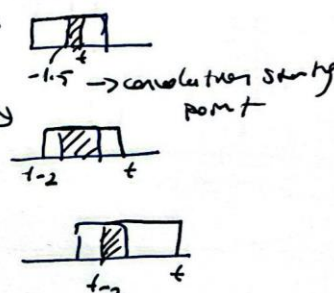
figure 5: Part 1.2 (b-c)

Part 1.2 c cont

$$\psi_1(t) = \begin{cases} 0 & t < 10.5 \\ t-10.5 & 10.5 \leq t < 11.5 \\ 1 & 11.5 \leq t < 12.5 \\ 13.5-t & 12.5 \leq t < 13.5 \\ 0 & t \geq 13.5 \end{cases}$$

ii) $\psi_2(t) = \xi(t+2) * \eta(t+10)$ and then find $\psi_3(t) = \psi_2(t-12)$
using the same logic with part i $\xi(t+2) \rightarrow$  $\eta(t+10) \rightarrow$ 

$$\psi_2(t) = \begin{cases} 0 & t < -1.5 \\ t+1.5 & -1.5 \leq t < -0.5 \\ 1 & -0.5 \leq t < 0.5 \\ 1.5-t & 0.5 \leq t < 1.5 \\ 0 & 1.5 \leq t \end{cases}$$



$-1.5 \rightarrow$ convolution starting point

we can see that

$$\psi_2(t) = \psi_1(t+12)$$

$$\psi_2(t-12) = \psi_1(t-12+12) = \psi_1(t) = \psi_3(t)$$

d) $\xi(t) = e^{-\frac{t^2}{2}}$ and $\eta(t) = 2e^{-\frac{t^2}{2}} \Rightarrow \psi(t) = \xi(t) * \eta(t)$

$$\begin{aligned} \psi(t) &= \int_{-\infty}^{\infty} e^{-\frac{c^2}{2}} 2e^{-\frac{(t-c)^2}{2}} dc = 2 \int_{-\infty}^{\infty} e^{-\left[\frac{c^2 + (t-c)^2}{2}\right]} dc \\ &= 2 \int_{-\infty}^{\infty} e^{-\frac{1}{2}(2c^2 - 2tc + t^2)} = 2 \int_{-\infty}^{\infty} e^{-c^2 + tc - \frac{t^2}{2}} = \begin{cases} -c^2 + tc - \frac{t^2}{2} \\ = -\left(c - \frac{t}{2}\right)^2 + \frac{t^2}{4} \end{cases} \\ &= 2e^{-\frac{t^2}{4}} \int_{-\infty}^{\infty} e^{-(c-\frac{t}{2})^2} dc \end{aligned}$$

The integral of Gaussian function is known $\rightarrow \int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$

figure 6 : Part 1.2 (c)

Apply R2B to our integral

$$\int_{-\infty}^{\infty} e^{-(\tau - \frac{t}{2})^2} d\tau = \sqrt{\pi} \Rightarrow y(t) = 2e^{-\frac{t^2}{4}} \sqrt{\pi} = 2\sqrt{\pi} e^{-\frac{t^2}{4}}$$

Comment \Rightarrow convolution of two gaussian function resulted
 M ~~other~~ another gaussian function

figure 7 : Part 1.2 (c) continued

Part 2

Part 2.1: Implementing Convolution

We turned the convolution operator we mentioned in Part 1 into a Matlab function. Using 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.

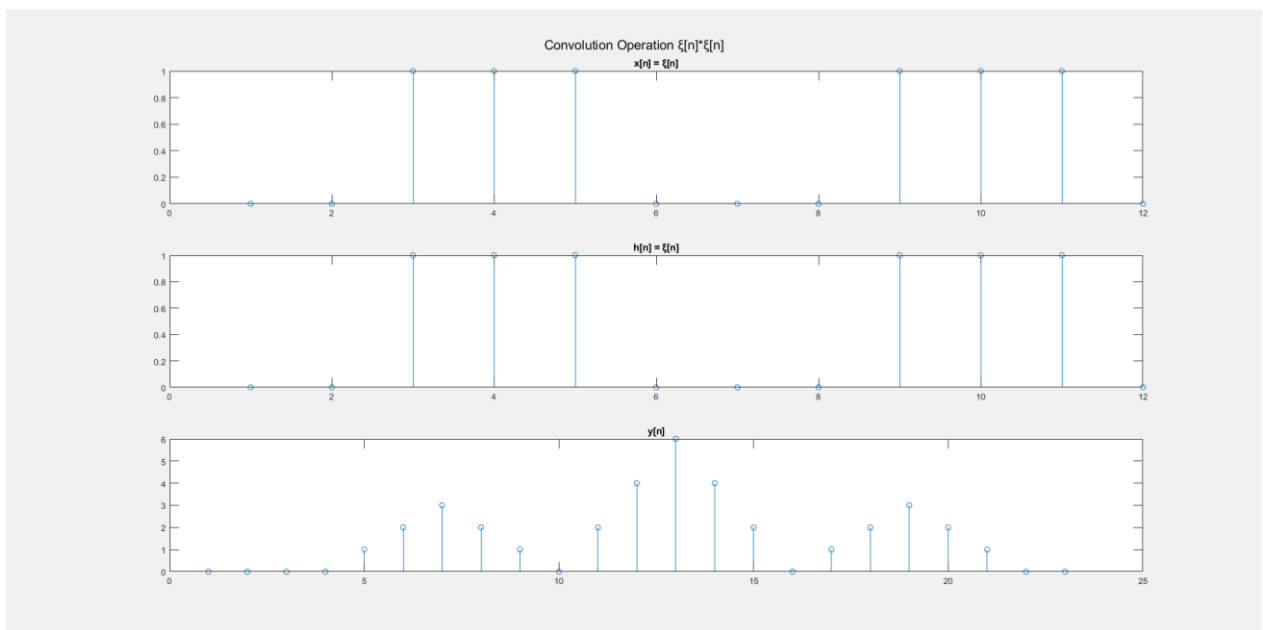


figure 8 : Part 2.2 convolution operation two same discrete signal

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 τ 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](#)

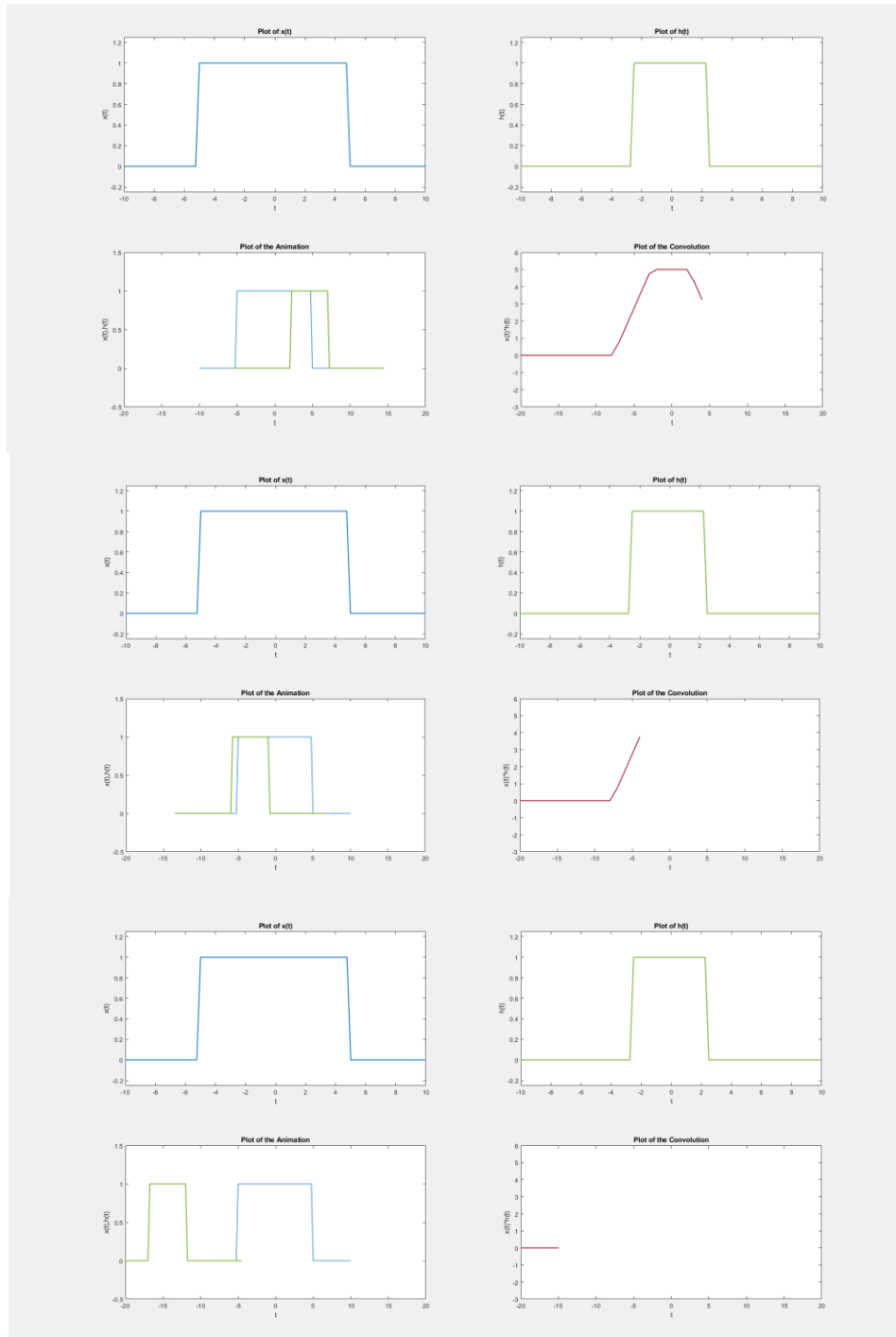


figure 9 : Part 3 Some sections from the convolution animation

Part 4:

Part 4.1 Defining Cross-Correlation

Part 4

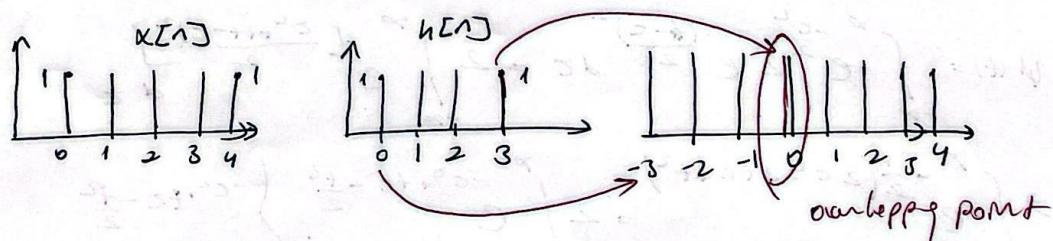
discrete cross correlation operator

$$y[n] = x[n] \star h[n] = \sum_{k=-\infty}^{\infty} \overline{x[k]} h[n+k]$$

$\{x[n]u[n]\}$ and $\{h[n]u[n]\}$ have lengths N_x, N_h respectively

$$y[n] = (\{x[n]u[n]\} \star \{h[n]u[n]\})$$

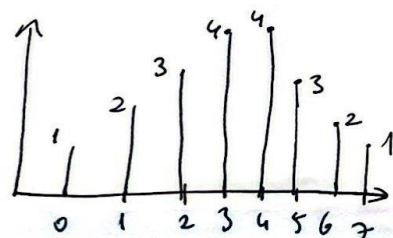
using similar approach we can say $\{x[n]u[n]\}$ and $\{h[n]u[n]\}$ signal starts from zero ("0") point. However, for cross correlation we observe how much $h[n]$ shifted along the x-axis to make it identical to x . As opposed to convolution we slides directly $h[n]$ signal through $x[n]$ (not flip). Take these two signal (see part 1)



as you can see we use similar manner with convolutions but with not mirrored $h[n]$ signal. Starting point is zero and the end point is $N_x + N_h - 2$ so that the length of $y[n]$ is $N_y = N_x + N_h - 1$

figure 10 : Part 4 Cross Correlation Definition

we use cross-correlation operation to measure of similarity of two signal and ~~for~~ signal transformation we use convolution. However our given signals are discrete with functions so that we cannot observe any difference in both length and output of the ~~signal~~ signal whether use convolution or cross correlation operation. $h[n]$ do not change whether it is mirrored or not.



$$\text{total length } 4 + 5 - 1 = 8$$

$$x_1 + x_2 - 1 = 8$$

$$x[n] * h[n]$$

$$x[n] * h[n]$$

where

$$x[n] = \{x[n] u[n]\}$$

$$h[n] = \{x[n] u[n]\}$$

figure 11 : Part 4 Cross Correlation Definition (Continued)

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

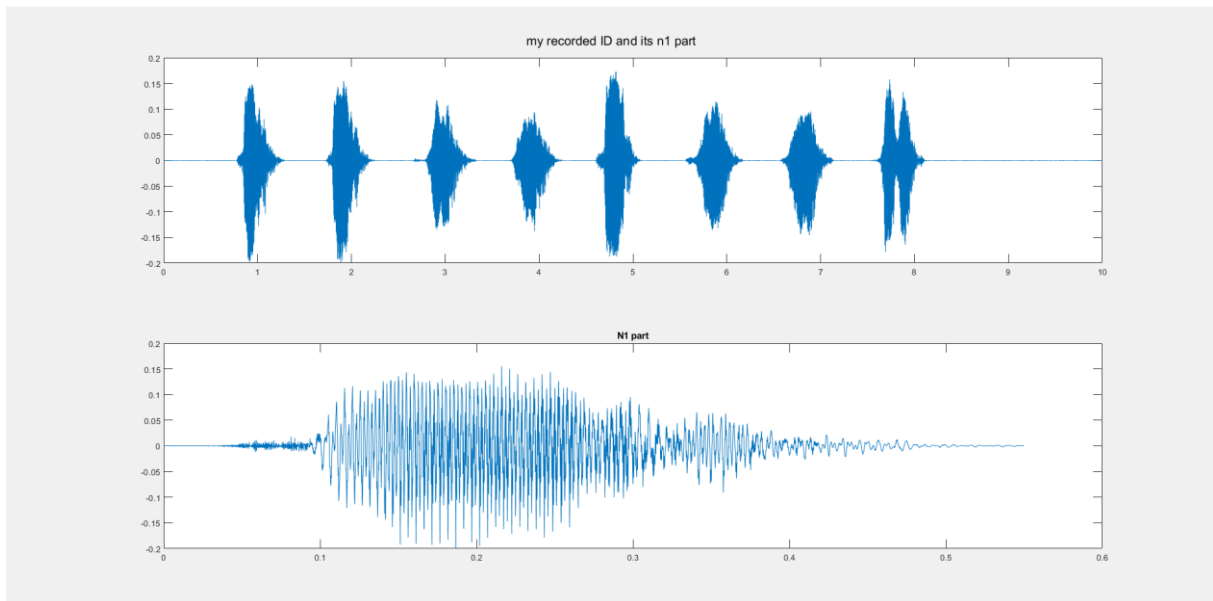


figure 12 : Plot of My Recorded ID and its n1 part

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.

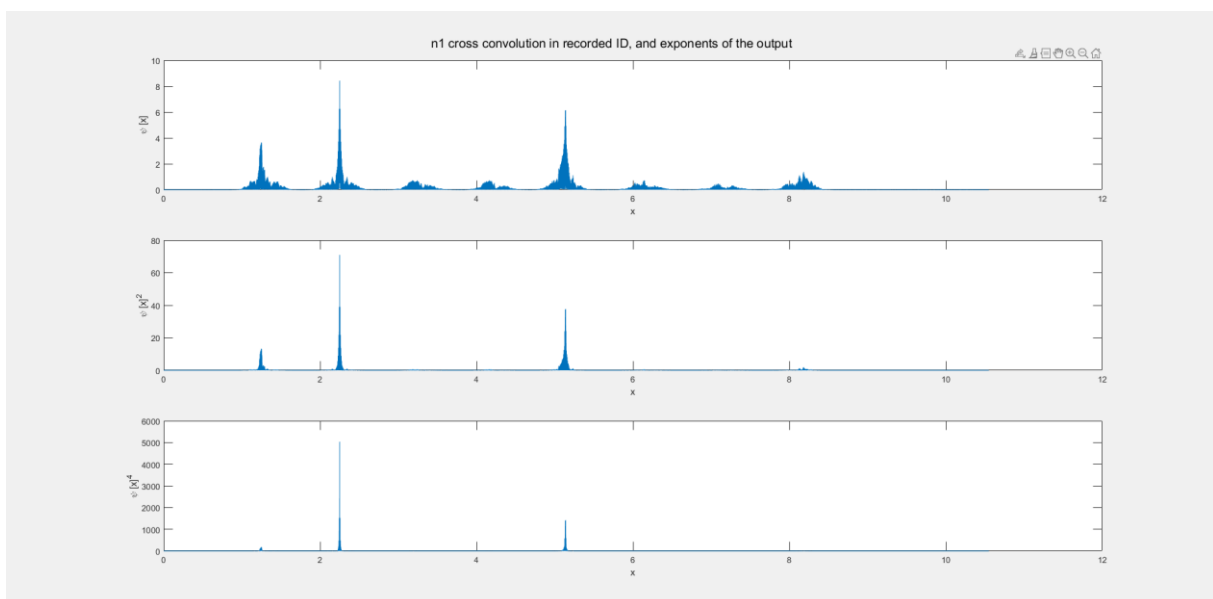


figure 13 : n1 cross convolution in recorded ID and exponents of output

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.

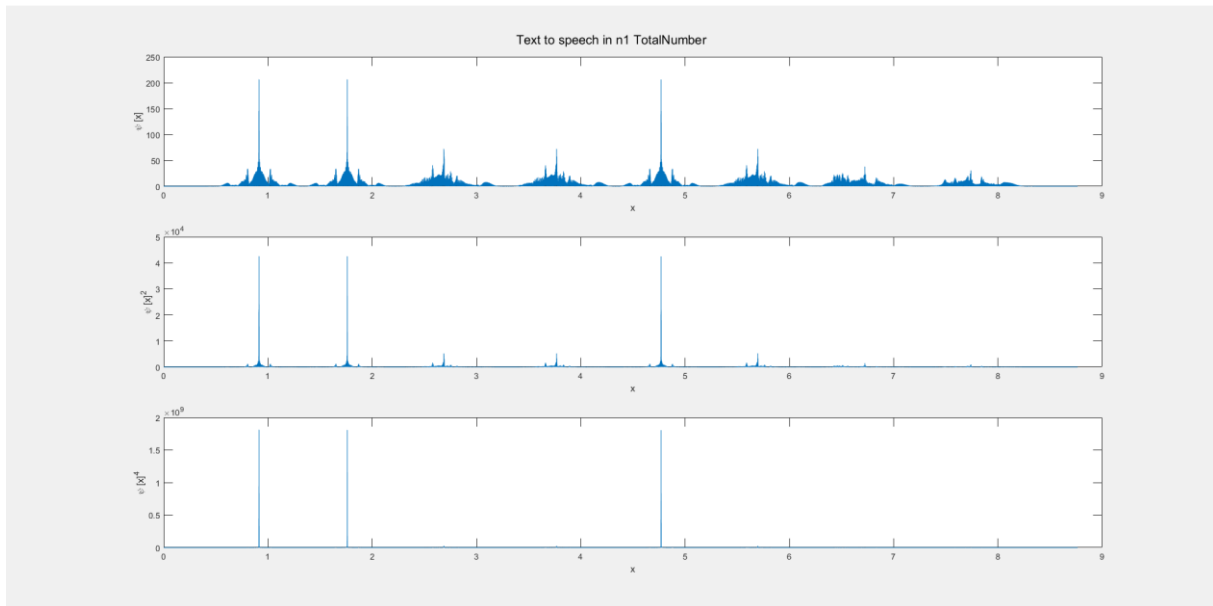


figure 14 : $n1$ cross convolution text to speech $n1$ and TotalNumber and its exponents

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 $n1$.

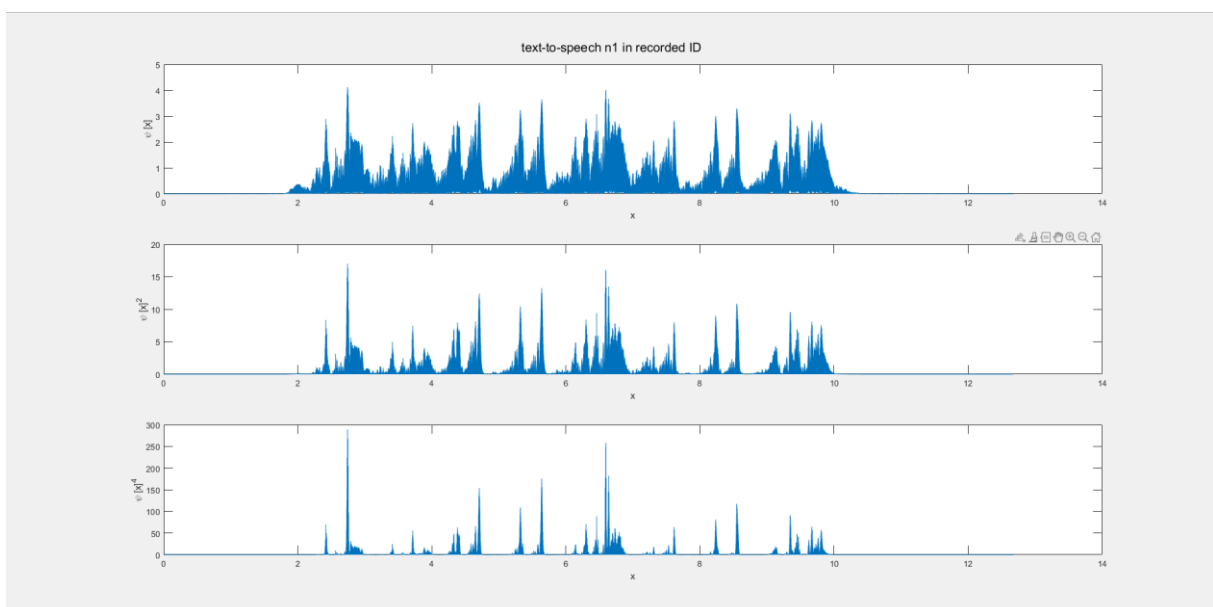


figure 15 : $n1$ cross convolution text to speech $n1$ and recorded ID and its exponents

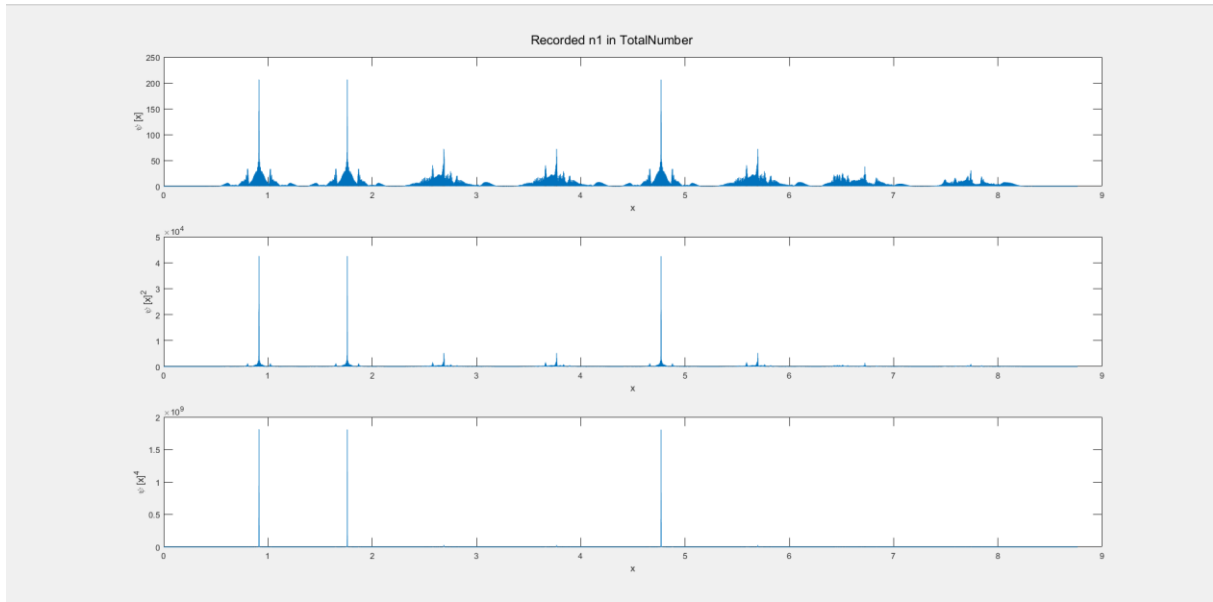


figure 16 : n1 cross convolution recorded n1 and TotalNumber and its exponents

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 length as audio_len to generate noise signal with the same length. Then we calculate power of signal and store as p-signal by using following calculation:

$$P_{signal} = \frac{1}{N_x} \sum_{i=1}^{N_x} (x[i])^2$$

N_x = Number of samples

$x[i]$ = sample of sequence

To generate noise we use additive white Gaussian noise (AWGN) To generate this in matlab we use following code part and use previously calculated values audio_len.

```
rng(5)
awgn = sqrt(p_noise).*randn([audio_len, 1]);
```

figure 17 : AWGN noise signal generation in MATLAB

Then we add this noise signal to our audio signal ("TotalNumber"), plot and listen these 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.

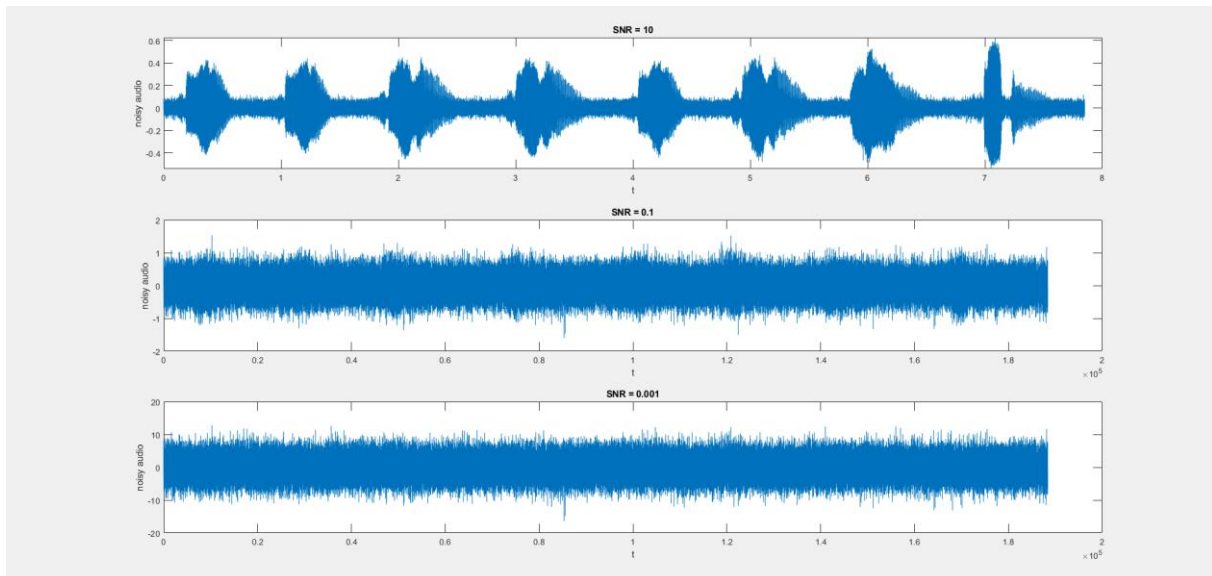


figure 18 : Adding Noise signal TotalNumber and with respect to SNR values

```
p_signal =
    0.0094

p_noise_SNR10 =
    9.4066e-04
    0.0094

p_noise_SNR01 =
    0.0941

p_noise_SNR0001 =
    9.4066
```

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.

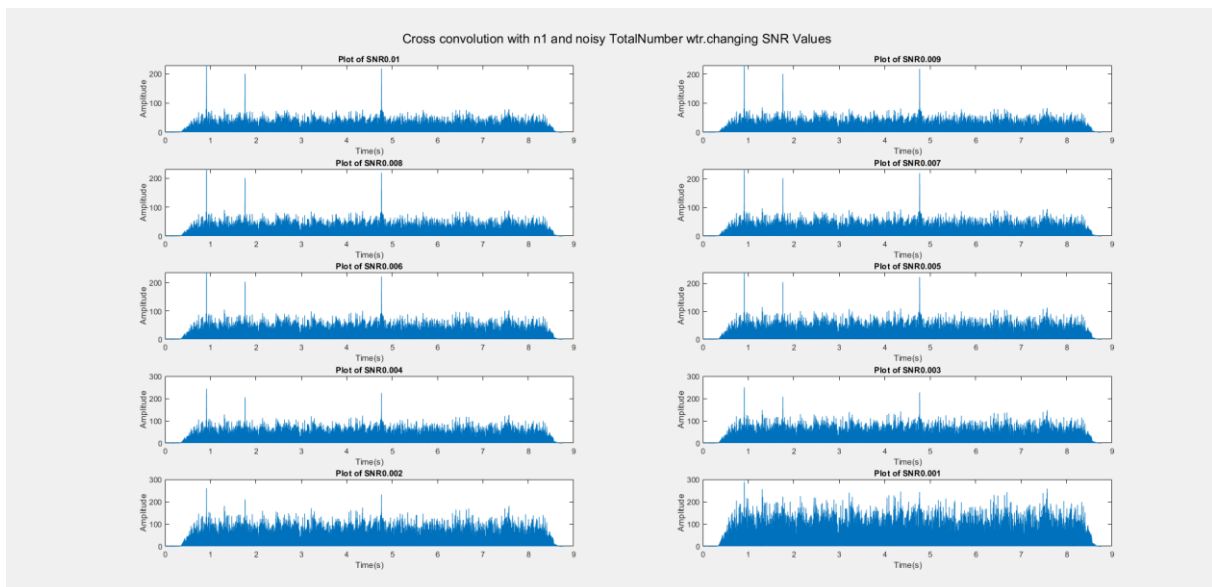


figure 20 : Cross Convolution operation Google n1 and noisy TotalNumber with changing SNR values

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
    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  $\xi[n]*\xi[n]$ ')
subplot(3,1,1)
stem(x);
title('x[n] =  $\xi[n]$ ');
subplot(3,1,2)
stem(h);
title('h[n] =  $\xi[n]$ ');
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];

%lengths
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), flipplr(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

```