EEE 321

Report: Signals and Systems Lab 2

Yigit Turkmen – 22102518

March 1, 2024

Introduction:

This lab report investigates the principles of convolution and cross-correlation in signal processing, emphasizing their application and analysis using MATLAB. It aims to enhance students' understanding of these concepts' significance in real-world scenarios, like noise reduction and speech recognition, while also refining their analytical and programming skills in electrical and electronics engineering contexts.

Part 1

1.1 Defining Convolution:

For this part see the last part of the report **(appendix)**, for ease of use.

1.2 Evaluating Convolution:

For this part see the last part of the report **(appendix)**, for ease of use.

Part 2

2.1 Implementing Convolution

See code 1

```
function [y] = ConvFUNC(x,h)
    Nx = length(x);
    Nh = length(h);
    Ny = Nx + Nh - 1;
    y = zeros(1, Ny);

    for i = 1:Ny
        k = max(1, i+1-Nh):min(i, Nx);
        y(i) = sum(x(k) .* h(i-k+1));
    end
end
```

Code 1: Related function to calculate the convolution

2.2 Testing the Convolution Function

See code 2 and the figures 1-3. Note that here $\xi[n]$ represents both x[n] and h[n]. Also $\Psi[n]$ represents the y[n].

```
x = [0 0 1 1 1 0 0 0 1 1 1 ];
h = [0 0 1 1 1 0 0 0 1 1 1 ];
y = conv(x, h);
z = 3 : 5;

t = ConvFUNC(x,h);
subplot(2,2,1)
stem(x)
xlabel("n")
ylabel("ξ[n]")
title("ξ[n]")

subplot(2,2,2)
stem(h)
xlabel("n")
ylabel("ξ[n]")
title("ξ[n]")

subplot(2,2,3)
stem(y)
xlabel("n")
ylabel("Ψ[n]")
title("Ψ[n] (product of convolution)")
```

Code 2: Whole code for this part

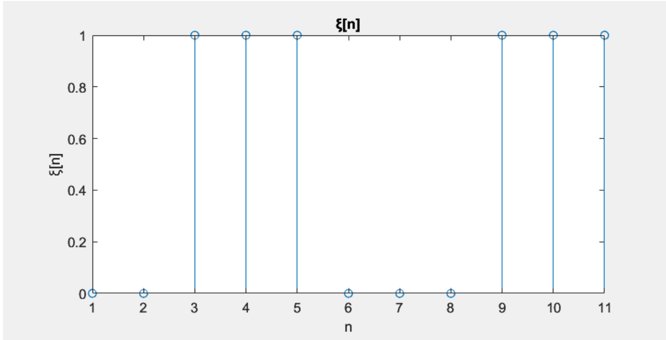


Figure 1: Plot of ξ[n] = x[n], h[n]

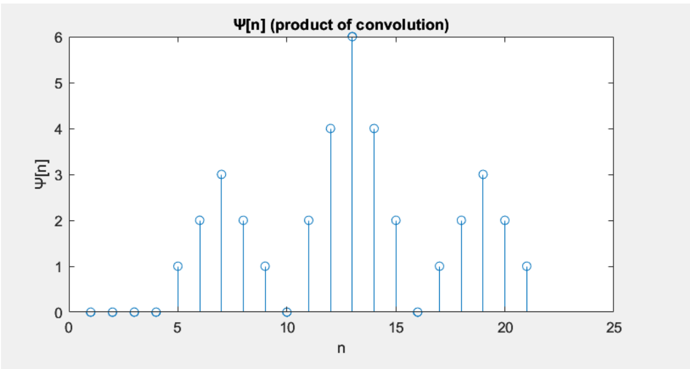


Figure 2: Plot of Ψ[n] = y[n]

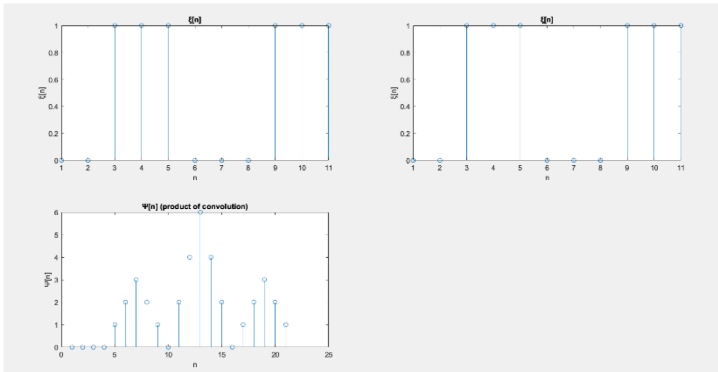


Figure 3: Whole plots in one frame

Part 3

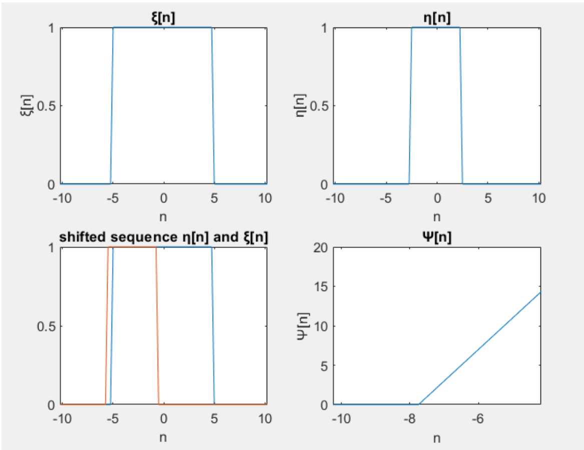
3.1 Creating Convolution Animation

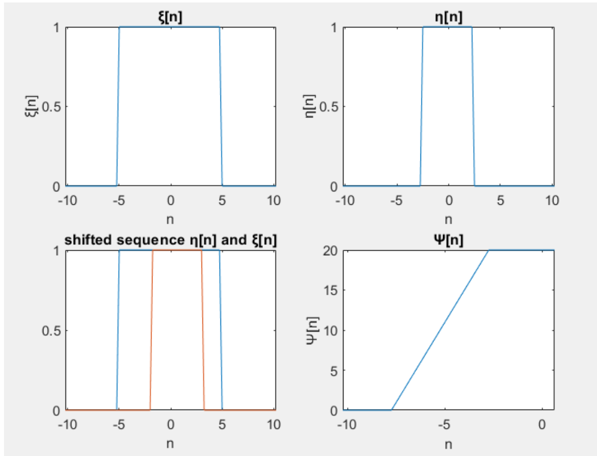
See code 3 and figures 4-6.

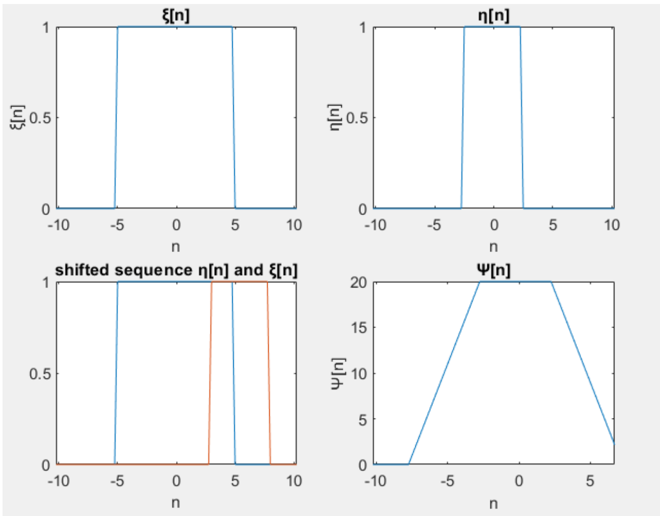
```
t = -10.25:0.25:10.25;
u = @(t) double(t>=0);
e = u(t+5)-u(t-5);
n = u(t+2.5)-u(t-2.5);
```

```
mu = ConvFUNC(e,n);
ts = -20.5:0.25:20.5;

flipped_sequence = fliplr(n);
new_flipped_sequence = zeros(1,83);
new_flipped_sequence(2:21) = flipped_sequence(33:52);
shifted_sequence = new_flipped_sequence;

for  i = 1:165
shift_length = min(i, length(n));
subplot(2,2,1)
plot(t,e)
xlabel("n")
ylabel("ξ[n]")
title("ξ[n]")

subplot(2,2,2)
plot(t,n)
xlabel("n")
ylabel("η[n]")
title("η[n]")

subplot(2,2,3)
plot(t,e)
xlabel("n")
title("shifted sequence η[n] and ξ[n]")
hold on

plot(t,shifted_sequence)
hold off
subplot(2,2,4)

plot(ts,mu)
xlabel("n")
ylabel("Ψ[n]")
title("Ψ[n]")
xlim([-10.25 -10.25 + shift_length*0.32])
ylim([0 20])

shifted_sequence = [zeros(1, shift_length), new_flipped_sequence(1:end-
shift_length)];
drawnow
pause(0.1);
end
```

Code 3: Code for part 3







Figures 4,5 and 6: For animation demonstration

Part 4

4.1 Defining Cross-Correlation

For this part see the last part of the report (appendix), for ease of use.

Question: How can you explain this discrete convolution operation in plain language as if you explain it to an early high school student?

Imagine you have two lists of numbers, one called the "signal" and the other the "filter." Discrete convolution is like blending these lists together to form a new list. First, you flip the filter list backwards. Then, you slide this flipped filter across the signal, multiplying corresponding numbers at each position and summing them up to get a single number for the new list. You keep sliding the filter one step at a time, repeating the multiply-and-add process, until you've moved across the entire signal. Each step gives you a new number for the final list, resulting from blending the original signal through the unique perspective of the filter. This process, akin to viewing a landscape through a moving window and recording a story at each step, is crucial in fields like signal processing and image manipulation, capturing the essence of blending effects and features detection.

4.2 Building a Basic Speech Recognition Algorithm

See figures 7, 8, 9, and 10 . These figures are related to my actual voice recording

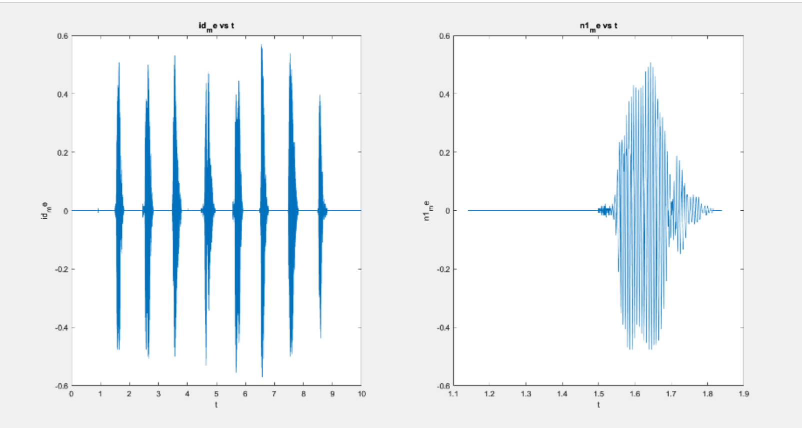


Figure 7: Left is total recording, right is only n1 = 2

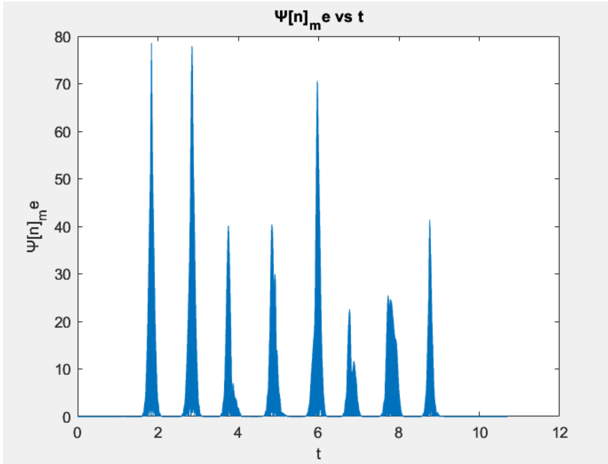


Figure 8: Ψ[n]_me vs t

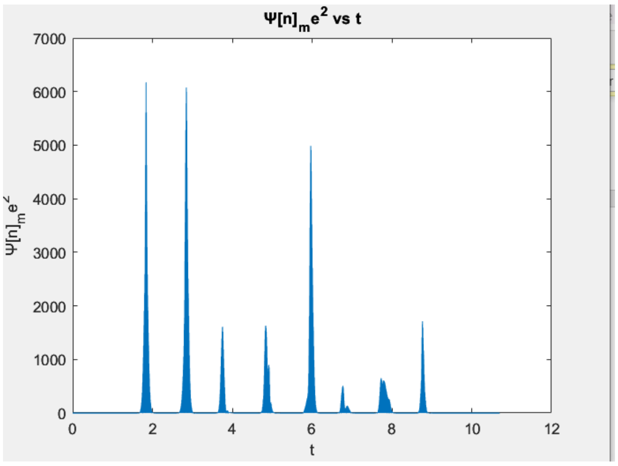


Figure 9: Ψ[n]^2_me vs t

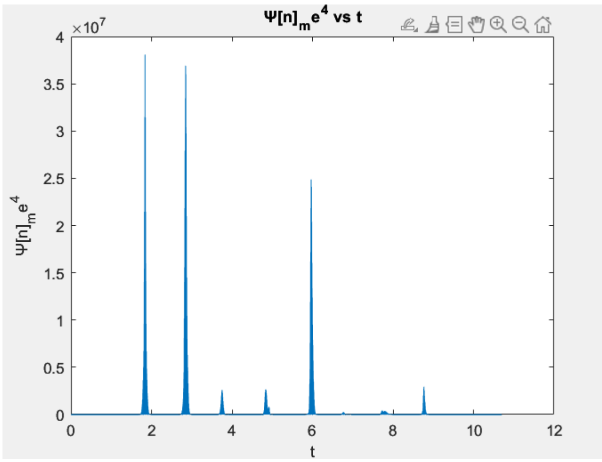


Figure 10: Ψ[n]^4_me vs t

Questions: What did you observe as you have changed Ψ[n] to Ψ[n]^2 and Ψ[n]^4? Were you able to detect each occurrence of *n*1 in your whole speech signal? Why? If you were to record your voice saying *n*1, what might have changed in Ψ[n]?

As the plots change to the square and square of square terms, the peaks where my n1 recording located, becomes much clearer than the previous plots.

Yes, I was able to detect the each occurence of n1 in my whole speech signal, since there were more than ten attempts to record my voice to make these plots clearer and as a result, the longer peaks (first three) are, as can be observed, my n1 peaks.

By recording "n1" directly, as opposed to finding it within a larger recording, I basically simplify the process of signal extraction and potentially increase the accuracy of the recognition algorithm. This is because the signal of interest, "n1," will be isolated and possibly recorded with consistent quality and minimal background noise, making it easier to compare with the reference signal or to identify within a larger dataset. Therefore, this could lead to to the larger and sharper peaks in $|\Psi[n]|$.

4.2 Part 2: ***TotalNumber***

See figures 11-14 . These figures are related to the python recordings.

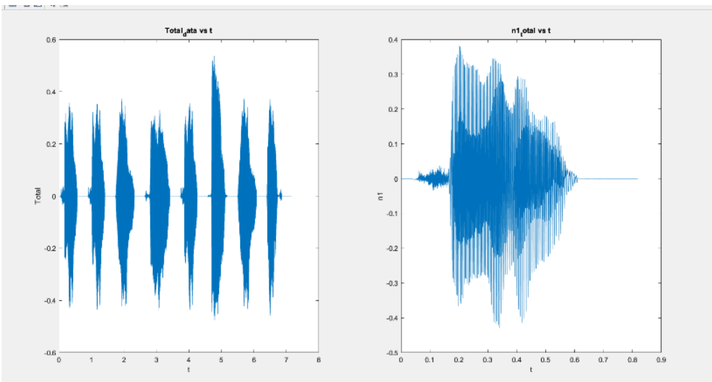


Figure 11: : Left is total recording, right is only n1 = 2

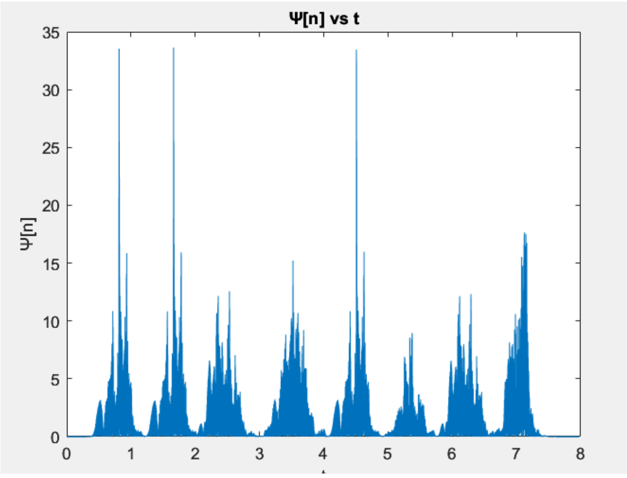


Figure 12: Ψ[n] vs t

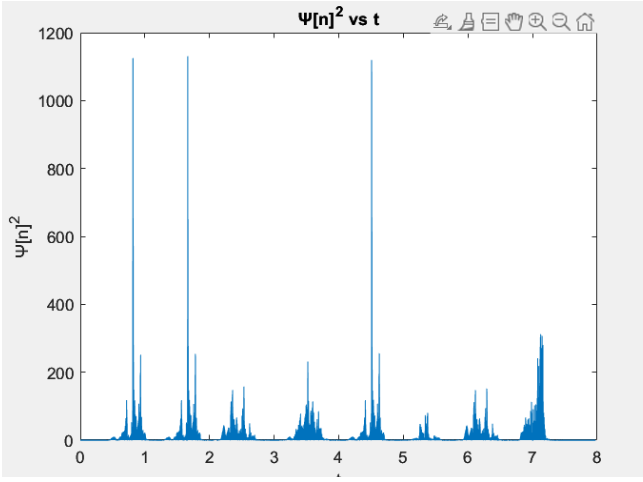


Figure 13: Ψ[n]^2 vs t

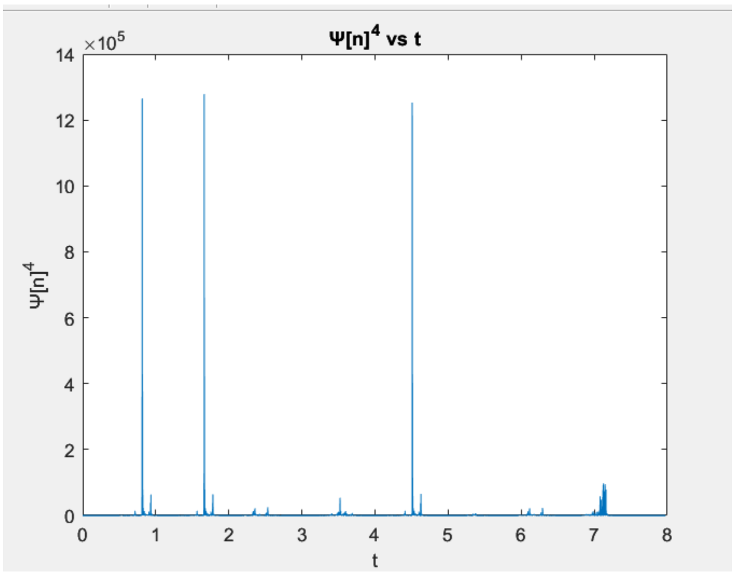


Figure 14: Ψ[n]^4 vs t

Questions: Compare the result of the text-to-speech signal with the result you have obtained in the previous paragraph. If there exists a difference, what might have caused this difference? Instead of finding the cross-correlation operation, what would have changed in Ψ[n] if you had used **ConvFUNC** directly without using the relation you have obtained in Part 4.1?

Comparing text-to-speech and human recordings reveals differences mainly because digital voices are clearer and more consistent, while my voice can vary and include background noise. Plus, my speech has emotional tones that text-to-speech lacks, affecting how signals match up during analysis. These factors contribute to why we see varied results between the two types of speech signals e.g. Python voice recordings are much sharper than the mines.

No detection observed when ConvFUNC directly used, see the figure 15. Therefore, it is important to use cross correlation for this assignment.

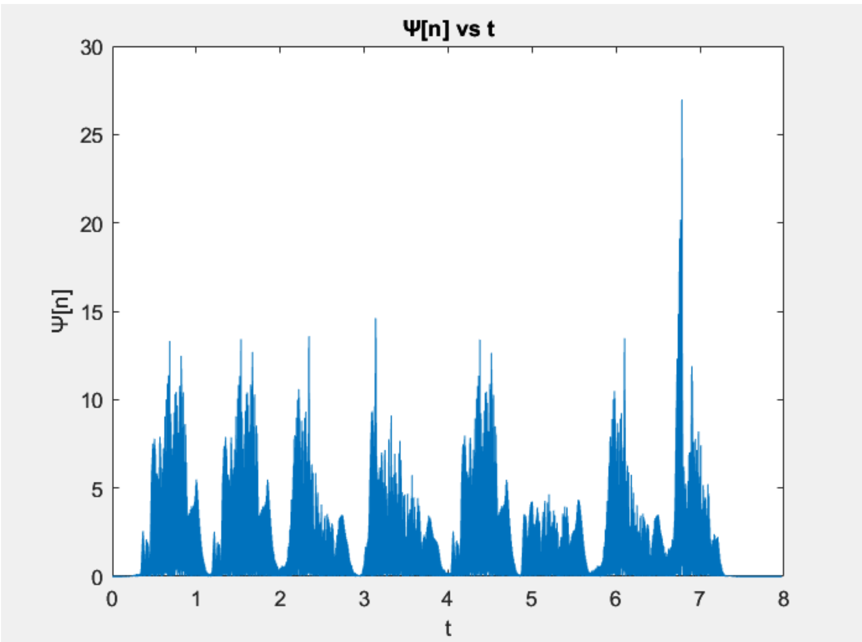


Figure 15: ConvFUNC directly used: Ψ[n] vs t

Questions: Search for *n*2 that you have obtained from the text-to-speech algorithm in *TotalNumber* obtained from the text-to-speech algorithm. Does the result make sense?

After searching the specific *n*1s indicated in the preceding paragraphs, search for *n*1 that you obtained from the text-to-speech algorithm in your own speech signal and check for Ψ[n]^4. Repeat this by searching for *n*1 that you have obtained from your own speech signal in *TotalNumber* obtained from the text-to-speech algorithm. If there exists a difference, which of these searches were more successful? Why? What do you think would make the less successful method better? Indicate your comments.

For n2, there is no insance in audio. Yes, the result makes sense since n2=9 and there is no "9" in my id. Since my voice is not coherent with the robotic sound, both of the combination's results are not good to pick the n2 see the figure 16. Noise reduction and signal enhancements method and signal enhancement techniques can be used with the Machine Learning models to make it better.

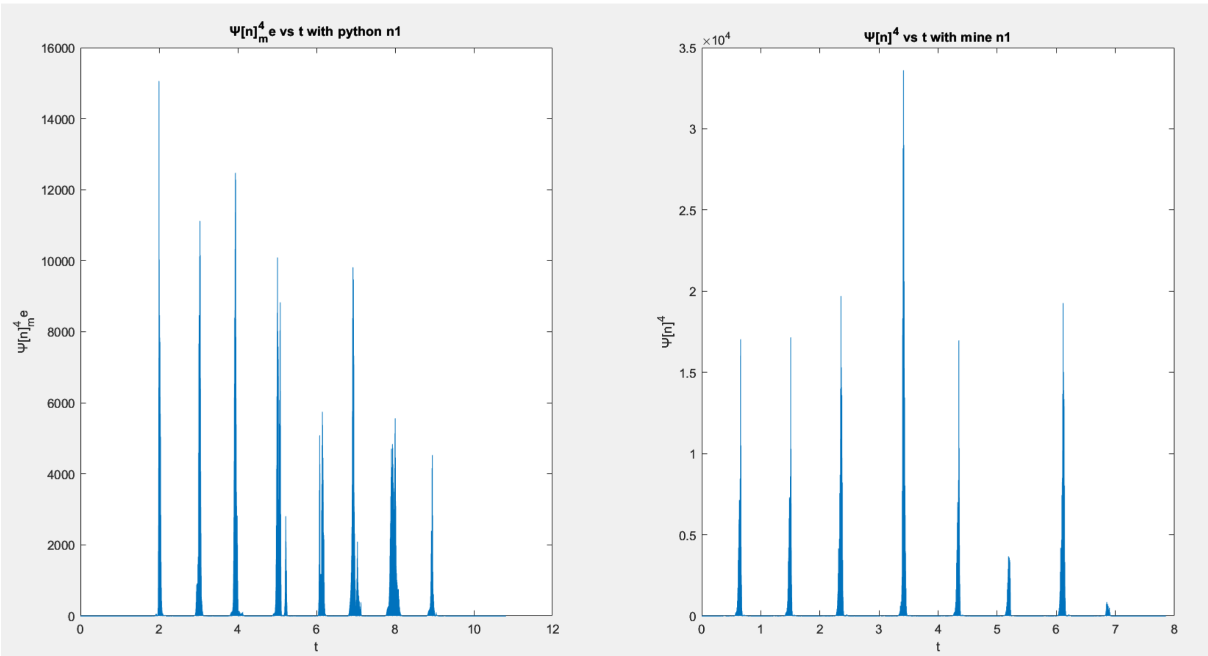


Figure 16: Left one is my id voice with python n1, right one is python id voice with my n1 voice

Code for Part 4 and the searching part:

```
%22102518
y = 0:1:9;
p = [2 1];
lambda = [3 4 6 7 9];
delta = mod(22102518,7);
deltap = mod(delta, length(p)) + 1;
deltalambda = mod(delta, length(lambda)) + 1;
n1 = p(deltap);
n2 = lambda(deltalambda);

%recObj = audiorecorder(8192,16,1);
%recDuration = 10;
%disp("Begin speaking.")
%recordblocking(recObj,recDuration);
%disp("End of recording.")

data = audioread('my_digit_sound_b.flac');
t = 0 + 1/8192 :1/8192:10;
t_mu = 0 + 1/8192 :1/8192:30;

plot(t, data)
n1data = data([9338:15074]);
n1datacross = flipud(n1data);

mu = abs(ConvFUNC(n1datacross, data));

figure(1)
subplot(1,2,1)
plot(t,data)
title("id_me vs t")
xlabel("t")
ylabel("id_me")

subplot(1,2,2)
plot(t([9338:15074]),n1data)
xlabel("t")
ylabel("n1_me")
title("n1_me vs t")

figure(2)
plot(t_mu([1:length(mu)]),mu)
xlabel("t")
ylabel("Ψ[n]_me")
title("Ψ[n]_me vs t")

figure(3)
mu_sq = mu .* mu;
plot(t_mu([1:length(mu)]),mu_sq)
xlabel("t")
ylabel("Ψ[n]_me^2")
title("Ψ[n]_me^2 vs t")


figure(4)
mu_quad = mu_sq .* mu_sq;
plot(t_mu([1:length(mu)]),mu_quad)
```

```
xlabel("t")
ylabel("Ψ[n]_me^4")
title("Ψ[n]_me^4 vs t")

audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
dataTotal = resample(originalAudioData, targetSampleRate, originalSampleRate);

audioFilePath = 'n1.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
datan1py = resample(originalAudioData, targetSampleRate, originalSampleRate);


datan1_flip = flipud(datan1py);
%datan1_flip = datan1py;
mu_total = abs(ConvFUNC(datan1_flip, dataTotal));


figure(5)
subplot(1,2,1)
plot(t_mu([1:58606]),dataTotal)
xlabel("t")
ylabel("Total")
title("Total_data vs t")

subplot(1,2,2)
plot(t([1:length(datan1py)]),datan1py)
xlabel("t")
ylabel("n1")
title("n1_total vs t")

figure(6)
plot(t_mu([1:length(mu_total)]),mu_total)
xlabel("t")
ylabel("Ψ[n]")
title("Ψ[n] vs t")

figure(7)
mu_sqtotal = mu_total .* mu_total;
plot(t_mu([1:length(mu_total)]),mu_sqtotal)
xlabel("t")
ylabel("Ψ[n]^2")
title("Ψ[n]^2 vs t")


figure(8)
mu_quadtotal = mu_sqtotal .* mu_sqtotal;
plot(t_mu([1:length(mu_total)]),mu_quadtotal)
xlabel("t")
ylabel("Ψ[n]^4")
title("Ψ[n]^4 vs t")
```

“General code for the part 4 without searching n1 etc.”

```
y = 0:1:9;
p = [2 1];
lambda = [3 4 6 7 9];
delta = mod(22102518,7);
deltap = mod(delta, length(p)) + 1;
deltalambda = mod(delta, length(lambda)) + 1;
n1 = p(deltap);
n2 = lambda(deltalambda);

data = audioread('my_digit_sound_b.flac');
t = 0 + 1/8192 :1/8192:10;
t_mu = 0 + 1/8192 :1/8192:30;


n1datacross = flipud(n1data);

mu = abs(ConvFUNC(datan1_flip, data));
mu_sq = mu .* mu;
figure(1)
subplot(1,2,1)
mu_quad = mu_sq .* mu_sq;
plot(t_mu([1:length(mu)]),mu_quad)
xlabel("t")
ylabel("Ψ[n]^4_me")
title("Ψ[n]^4_me vs t with python n1")

audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
dataTotal = resample(originalAudioData, targetSampleRate, originalSampleRate);

audioFilePath = 'n1.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
datan1py = resample(originalAudioData, targetSampleRate, originalSampleRate);


datan1_flip = flipud(datan1py);
%datan1_flip = datan1py;
mu_total = abs(ConvFUNC(n1datacross, dataTotal));
mu_sqtotal = mu_total .* mu_total;

subplot(1,2,2)
mu_quadtotal = mu_sqtotal .* mu_sqtotal;
plot(t_mu([1:length(mu_total)]),mu_quadtotal)
xlabel("t")
ylabel("Ψ[n]^4")
title("Ψ[n]^4 vs t with mine n1")
```

“Code for the searching n1 in python voice and the vice versa”

Part 5

5.1 Observing the Effects of SNR

```
p_signal =

    0.0092
```

Value of the p_signal obtained from MATLAB command line after running the code.

```
p_noise =

   9.2250e-04
```

Value of the p_noise obtained from MATLAB command line after running the code. SNR = 10.

Code for part 5.1:

```
audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
audio_array = resample(originalAudioData, targetSampleRate, originalSampleRate);

audio_len = length(audio_array);
snr = 0.001;
p_signal = sum(audio_array .* audio_array) / audio_len;
p_noise = p_signal / snr;

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

noisy_audio = awgn + audio_array;
```

Questions: What do you hear? Repeat this process for the SNR values of 0.1 and 0.001. Listen to your noise-corrupted signals for these SNR values as well. How does the sound change? Can you hear the numbers?

When SNR = 10, I hear my id number but with some noises in background, as SNR decreases the amount of the noise increases and the id sounds diminish. The id sound is very light when SNR = 0.1, and absolutely not hearable at SNR = 0.001.

SNR: 0.1:

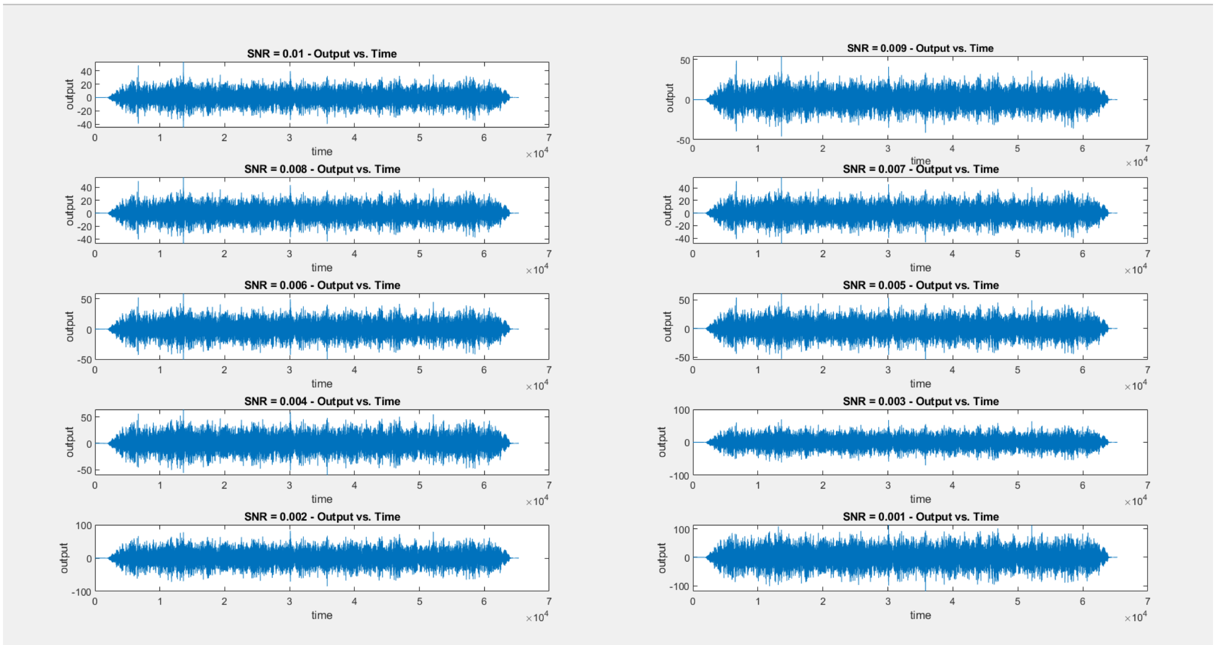
```
p_noise =

    0.0922
```

SNR: 0.001:

```
p_noise =

    9.2250
```

5.2 Detecting the SNR limit

Plots related to each SNR value



Questions: By observing the resulting plots, determine if your system can detect the number pronounced in the audio file of your *filter*. At what SNR value does your system start to fail?

Yes, my system can detect the numbers pronounced in the audio file of the filter. You can check it by looking the peaks at the plots.

From my observation, starting from SNR = 0.003 the peaks become invisible and thus, system starts to fail.

Code for part 5.2:

```
audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
audio_array = resample(originalAudioData, targetSampleRate, originalSampleRate);
audio_len = length(audio_array);

audioFilePath = 'n1.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
filter = resample(originalAudioData, targetSampleRate, originalSampleRate);

for snr = 0.01:-0.001:0.001
    p_signal = sum(audio_array .* audio_array) / audio_len;
    p_noise = p_signal / snr;
    rng (5)
    awgn = sqrt ( p_noise ).* randn ([ audio_len , 1]);
    noisy_audio = awgn + audio_array;
    filter_flip = flipud(filter);
    output = ConvFUNC(noisy_audio, filter_flip);
    i = 11 - snr * 1000;
    subplot(5,2,i)
    plot(output)
    xlabel("time")
    ylabel("output")
    title(['SNR = ', num2str(snr), ' - Output vs. Time']);
end
```

Appendix for part 1.1 and 1.2 and 4.1:

1.1 Defining Convolutions

$$\Psi[n] = (\xi[n]u[n]) * (\eta[n]u[n]) = \sum_{-\infty}^{\infty} (\xi[k]u[k])(\eta[n-k]u[n-k])$$

$$= \sum_{k=\max(0,n-N_\eta+1)}^{\infty} \xi[n](\eta[n-k]u[n-k]) = \sum_{k=\max(0,n-N_\eta)}^{\min(N_\xi-1,n)} \xi[k]\eta[n-k]$$

$\rightarrow u[n-k]=0$ for $k>n$

$\xi[k]=0$ for $k>N_\xi-1$

$u[k]=0$ for $k<0$

$\eta[n-k]=0$ for $k<n-N_\eta+1$

$$\Psi[n] = \sum_{k=\max(0,n-N_\eta+1)}^{\min(N_\xi-1,n)} \xi[k]\eta[n-k]$$

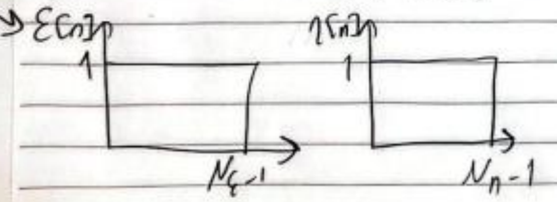
minimum n value that gives nonzero result is $n=0$, start and end point summation became equal and zero.

$\Rightarrow \Psi[0] = \sum_{0}^{0} \xi[k]\eta[-k] = \xi[0]\eta[0]$. maximum value giving nonzero result is when $n-N_\eta+1 = N_\xi-1 \Rightarrow n = N_\eta+N_\xi-2$.

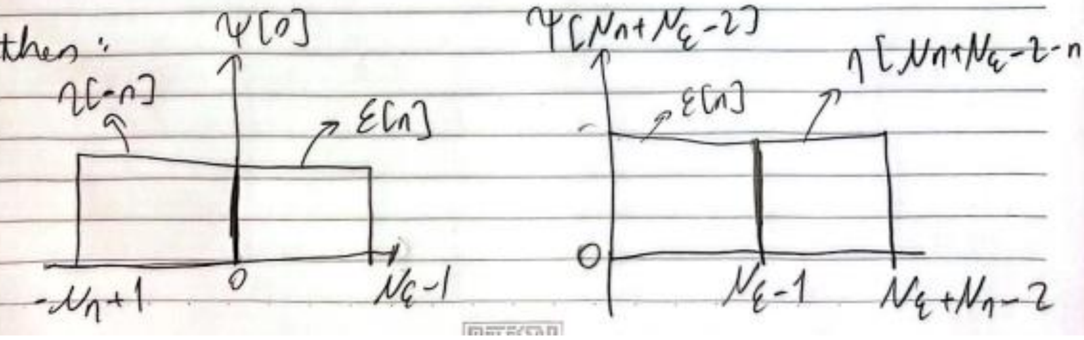
$$\Rightarrow \Psi[N_\eta+N_\xi-2] = \sum_{k=N_\xi-1}^{N_\xi-1} \xi[k]\eta[N_\eta+N_\xi-2-k] = \xi[N_\xi-1]\eta[N_\eta-1]$$

Assume:

from $n=0$ to $n=N_\eta+N_\xi-2$ length of $\psi[n]$ is $\boxed{N_\eta+N_\xi-1}$.



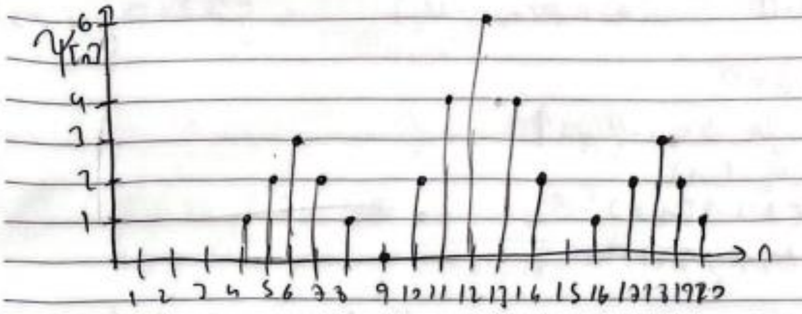
then:



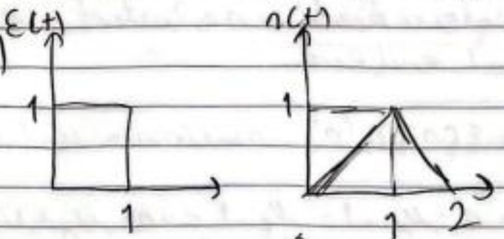
1.2)

a) $\psi[n] = \varepsilon[n] * \varepsilon[n] = \sum_{k=-\infty}^{\infty} \varepsilon[k]\,\varepsilon[n-k]$

$= \varepsilon[n-2] + \varepsilon[n-3] + \varepsilon[n-4] + \varepsilon[n-8] + \varepsilon[n-9] + \varepsilon[n-10]$



b)



$\psi(t) = \varepsilon(t) * \eta(t) * \varepsilon(t)$

$= \varepsilon(t) * \varepsilon(t)\, \eta(t)$

$\varepsilon(t) * \varepsilon(t) = \int_{-\infty}^{\infty} \varepsilon(\tau)\,\varepsilon(t-\tau)\,d\tau = \int_0^1 \varepsilon(t-\tau)\,d\tau = \int_0^1 u(t-\tau) - u(t-\tau-1)\,d\tau$

$= -r(t-\tau) + r(t-\tau-1)\Big|_0^1 = -r(t-1) + r(t-2) + r(t) - r(t-1)$

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

$\eta(t) * \eta(t) = \int_{-\infty}^{\infty} \eta(\tau)\,\eta(t-\tau)\,d\tau = \int_0^2 \eta(t-\tau)\,d\tau$

$= \int_0^2 r(t-\tau) - 2r(t-\tau-1) + r(t-\tau-2)\,d\tau.$

$\int r(t)\,dt = \frac{t^2}{2}u(t),$ → go next page.

$$\Rightarrow \; = -\frac{(t-\tau)^2}{2} u(t-\tau) + \frac{2(t-\tau-1)^2}{2} u(t-\tau-1) - \frac{(t-\tau-2)^2}{2}$$

$$\times u(t-\tau-2) \Big|_0^2$$

$$= -\frac{(t-2)^2}{2} u(t-2) + \frac{(t-3)^2}{2} u(t-3) - \frac{(t-4)^2}{2} u(t-4) +$$

$$+ \frac{t^2}{2} u(t) - (t-1)^2 u(t-1) + \frac{(t-2)^2}{2} u(t-2)$$

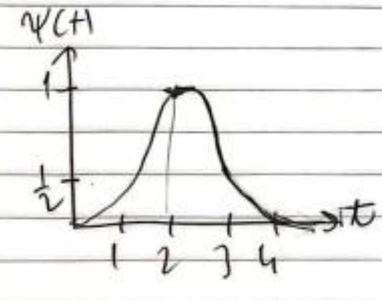
$$= \frac{t^2}{2} u(t) - (t-1)^2 u(t-1) + (t-3)^2 u(t-3) - \frac{(t-4)^2}{2} u(t-4).$$

$$\Rightarrow \; = \frac{t^2}{2} \quad \text{for } 0 \le t \le 1$$

$$-\frac{t^2}{2} + 2t - 1 \quad \text{for } 1 \le t \le 3$$

$$\frac{t^2}{2} + 3t + 8 \quad \text{for } 3 \le t \le 4$$

$$0 \quad \text{for } t \ge 4$$



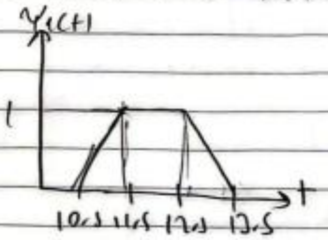
c) $\varepsilon(t) = u(t-1.5) - u(t-2.5)$, $\eta(t) = u(t-9) - u(t-11)$

$$\psi_1(t) = \varepsilon(t) * \eta(t) = \int_{-\infty}^{\infty} \varepsilon(\tau)\,\eta(t-\tau)\,d\tau = \int_{1.5}^{2.5} \eta(t-\tau)\,d\tau$$

$$= \int_{1.5}^{2.5} u(t-\tau-9) - u(t-\tau-11)\,d\tau = -r(t-\tau-9) + r(t-\tau-11)\Big|_{1.5}^{2.5}$$

$$= -r(t-11.5) + r(t-13.5) + r(t-10.5) - r(t-12.5)$$

$$= +r(t-10.5) - r(t-11.5) - r(t-12.5) + r(t-13.5)$$



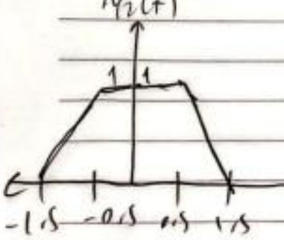
$$\psi_2(t) = \varepsilon(t+12) * \eta(t+10) = \int_{-\infty}^{\infty} \varepsilon(\tau+12)\,\eta(t+10-\tau)\,d\tau$$

$$= \int_{0.5}^{0.5} \eta(t-\tau+10)\,d\tau, \quad \tau = x+10, \; d\tau = dx \;\rightarrow\; \int_{-12.5}^{-9.5} \eta(t-x)\,dx$$

$$= \left[-r(t-x-9) + r(t-x-11)\right]\Big|_{-12.5}^{-9.5} = -r(t+0.5) + r(t-1.5) + r(t+1.5) - r(t-0.5)$$

$$= r(t-1.5) - r(t-0.5) - r(t+0.5) + r(t+1.5)$$

$$\Rightarrow \psi_2(t) = \psi_1(t+12), \; \psi_1(t) = \psi_2(t-12) = \psi_1(t)$$



$\psi_1(t)$ and $\psi_2(t)$ are time shifted versions of each other and $\psi_1(t)$ and $\psi_2(t)$ are identical functions.

d) $\varepsilon(t) = e^{-\frac{t^2}{2}}$, $\eta(t) = 2e^{-\frac{t^2}{2}}$

$$\therefore \psi(t) = \varepsilon(t) * \eta(t) = \int_{-\infty}^{\infty} \varepsilon(\tau)\,\eta(t-\tau)\,d\tau = \int_{-\infty}^{\infty} e^{-\frac{\tau^2}{2}}\, 2e^{-\frac{(t-\tau)^2}{2}}\, d\tau$$

$$= 2\int_{-\infty}^{\infty} e^{\frac{-2\tau^2+2t\tau-t^2}{2}}\, d\tau = 2e^{-\frac{t^2}{2}} \underbrace{\int_{-\infty}^{\infty} e^{-\tau^2} e^{t\tau}\, d\tau}$$

$$2e^{-\frac{t^2}{2}} \cdot \sqrt{\pi}\, e^{t^2/4}$$

$$= 2\sqrt{\pi}\, e^{-\frac{t^2}{4}}$$

convolving two sided decaying exponential function results is a slower decaying two sided exponential function with a higher peak value at origin.

4.1)

$$\psi[n] := \varepsilon[n]\,u[n] \star (\eta[n]\,u[n]) = \sum_{k=-\infty}^{\infty} \varepsilon^*[k]\,u^*[k]\,\eta[n+k]\,u[n+k]$$

$\varepsilon^*[k] = \varepsilon[k]$ and $u^*[k] = u[k]$ since both sequences have real values

$$= \sum_{k=\max(-n,0)}^{\infty} \varepsilon[k]\,\eta[n+k] = \sum_{k=\max(0,-n)}^{\min(N_\varepsilon-1,\,N_\eta-n-1)} \varepsilon[k]\,\eta[n+k]$$

$n = -N_\varepsilon + 1$ is the min. value n can be.

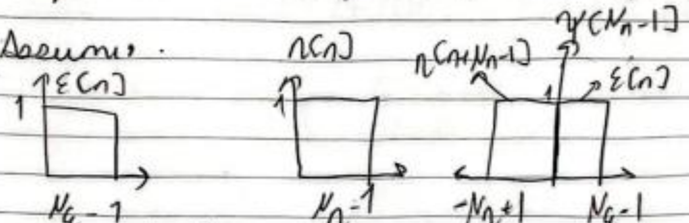
$$\psi[-N_\varepsilon+1] = \sum_{N_\varepsilon-1}^{N_\varepsilon-1} \varepsilon[k]\,\eta[-N_\varepsilon+1+k] = \varepsilon[N_\varepsilon-1]\,\eta[0]$$

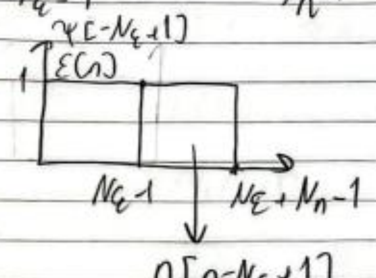
$n = N_\eta - 1$ is the max. value n can be.

$$\psi[N_\eta-1] = \sum_{0}^{0} \varepsilon[k]\,\eta[N_\eta-1+k] = \varepsilon[0]\,\eta[N_\eta-1]$$

length of $\psi[n]$ is $N_\eta - 1 - (-N_\varepsilon+1) + 1 = N_\eta + N_\varepsilon - 1$

Assume:





Cross corr. is equivalent of convolution that does not apply flipping to the one of the sequences.

Therefore, we can flip one of the sequences and conv. operation to calculate cross corr.

$$(\varepsilon[n]\,u[n]) * (\eta[-n]\,u[n]) = (\varepsilon[n]\,u[n]) \star (\eta[n]\,u[n]).$$

Whole Codes for the Assignment:

```
x = [0 0 1 1 1 0 0 0 1 1 1 ];
h = [0 0 1 1 1 0 0 0 1 1 1 ];
y = conv(x, h);
z = 3 : 5;

t = ConvFUNC(x,h);
subplot(2,2,1)
stem(x)
xlabel("n")
ylabel("ξ[n]")
title("ξ[n]")

subplot(2,2,2)
stem(h)
xlabel("n")
ylabel("ξ[n]")
title("ξ[n]")

subplot(2,2,3)
stem(y)
xlabel("n")
ylabel("Ψ[n]")
title("Ψ[n] (product of convolution)")
```

"lab2part2.m"

```
function [y] = ConvFUNC(x,h)
    Nx = length(x);
    Nh = length(h);
    Ny = Nx + Nh - 1;
    y = zeros(1, Ny);

    for i = 1:Ny
        k = max(1, i+1-Nh):min(i, Nx);
        y(i) = sum(x(k) .* h(i-k+1));
    end
end
```

"ConvFUNC.m"

```
t = -10.25:0.25:10.25;
u = @(t) double(t>=0);
e = u(t+5)-u(t-5);
n = u(t+2.5)-u(t-2.5);
mu = ConvFUNC(e,n);
ts = -20.5:0.25:20.5;

flipped_sequence = fliplr(n);
new_flipped_sequence = zeros(1,83);
new_flipped_sequence(2:21) = flipped_sequence(33:52);
shifted_sequence = new_flipped_sequence;

for  i = 1:165
shift_length = min(i, length(n));
subplot(2,2,1)
plot(t,e)
xlabel("n")
ylabel("ξ[n]")
title("ξ[n]")

subplot(2,2,2)
plot(t,n)
xlabel("n")
ylabel("η[n]")
title("η[n]")

subplot(2,2,3)
plot(t,e)
xlabel("n")
title("shifted sequence η[n] and ξ[n]")
hold on

plot(t,shifted_sequence)
hold off
subplot(2,2,4)

plot(ts,mu)
xlabel("n")
ylabel("Ψ[n]")
title("Ψ[n]")
xlim([-10.25 -10.25 + shift_length*0.32])
ylim([0 20])

shifted_sequence = [zeros(1, shift_length), new_flipped_sequence(1:end-
shift_length)];
drawnow
pause(0.1);
end
```

"lab2part3.m"

```matlab
%22102518
y = 0:1:9;
p = [2 1];
lambda = [3 4 6 7 9];
delta = mod(22102518,7);
deltap = mod(delta, length(p)) + 1;
deltalambda = mod(delta, length(lambda)) + 1;
n1 = p(deltap);
n2 = lambda(deltalambda);

%recObj = audiorecorder(8192,16,1);
%recDuration = 10;
%disp("Begin speaking.")
%recordblocking(recObj,recDuration);
%disp("End of recording.")

data = audioread('my_digit_sound_b.flac');
t = 0 + 1/8192 :1/8192:10;
t_mu = 0 + 1/8192 :1/8192:30;

plot(t, data)
n1data = data([9338:15074]);
n1datacross = flipud(n1data);

mu = abs(ConvFUNC(n1datacross, data));

figure(1)
subplot(1,2,1)
plot(t,data)
title("id_me vs t")
xlabel("t")
ylabel("id_me")

subplot(1,2,2)
plot(t([9338:15074]),n1data)
xlabel("t")
ylabel("n1_me")
title("n1_me vs t")

figure(2)
plot(t_mu([1:length(mu)]),mu)
xlabel("t")
ylabel("Ψ[n]_me")
title("Ψ[n]_me vs t")

figure(3)
mu_sq = mu .* mu;
plot(t_mu([1:length(mu)]),mu_sq)
xlabel("t")
ylabel("Ψ[n]_me^2")
title("Ψ[n]_me^2 vs t")


figure(4)
mu_quad = mu_sq .* mu_sq;
plot(t_mu([1:length(mu)]),mu_quad)
xlabel("t")
ylabel("Ψ[n]_me^4")
title("Ψ[n]_me^4 vs t")
```

```
audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
dataTotal = resample(originalAudioData, targetSampleRate, originalSampleRate);

audioFilePath = 'n1.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
datan1py = resample(originalAudioData, targetSampleRate, originalSampleRate);


datan1_flip = flipud(datan1py);
%datan1_flip = datan1py;
mu_total = abs(ConvFUNC(datan1_flip, dataTotal));


figure(5)
subplot(1,2,1)
plot(t_mu([1:58606]),dataTotal)
xlabel("t")
ylabel("Total")
title("Total_data vs t")

subplot(1,2,2)
plot(t([1:length(datan1py)]),datan1py)
xlabel("t")
ylabel("n1")
title("n1_total vs t")

figure(6)
plot(t_mu([1:length(mu_total)]),mu_total)
xlabel("t")
ylabel("Ψ[n]")
title("Ψ[n] vs t")

figure(7)
mu_sqtotal = mu_total .* mu_total;
plot(t_mu([1:length(mu_total)]),mu_sqtotal)
xlabel("t")
ylabel("Ψ[n]^2")
title("Ψ[n]^2 vs t")


figure(8)
mu_quadtotal = mu_sqtotal .* mu_sqtotal;
plot(t_mu([1:length(mu_total)]),mu_quadtotal)
xlabel("t")
ylabel("Ψ[n]^4")
title("Ψ[n]^4 vs t")
```

“lab2part4_without_comparing_n1.m”

```
y = 0:1:9;
p = [2 1];
lambda = [3 4 6 7 9];
delta = mod(22102518,7);
deltap = mod(delta, length(p)) + 1;
deltalambda = mod(delta, length(lambda)) + 1;
n1 = p(deltap);
n2 = lambda(deltalambda);

data = audioread('my_digit_sound_b.flac');
t = 0 + 1/8192 :1/8192:10;
t_mu = 0 + 1/8192 :1/8192:30;


n1datacross = flipud(n1data);

mu = abs(ConvFUNC(datan1_flip, data));
mu_sq = mu .* mu;
figure(1)
subplot(1,2,1)
mu_quad = mu_sq .* mu_sq;
plot(t_mu([1:length(mu)]),mu_quad)
xlabel("t")
ylabel("Ψ[n]^4_me")
title("Ψ[n]^4_me vs t with python n1")

audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
dataTotal = resample(originalAudioData, targetSampleRate, originalSampleRate);

audioFilePath = 'n1.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
datan1py = resample(originalAudioData, targetSampleRate, originalSampleRate);


datan1_flip = flipud(datan1py);
%datan1_flip = datan1py;
mu_total = abs(ConvFUNC(n1datacross, dataTotal));
mu_sqtotal = mu_total .* mu_total;

subplot(1,2,2)
mu_quadtotal = mu_sqtotal .* mu_sqtotal;
plot(t_mu([1:length(mu_total)]),mu_quadtotal)
xlabel("t")
ylabel("Ψ[n]^4")
title("Ψ[n]^4 vs t with mine n1")
```

“lab2part4_with_comparing_n1.m”

```
audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
audio_array = resample(originalAudioData, targetSampleRate, originalSampleRate);

audio_len = length(audio_array);
snr = 0.001;
p_signal = sum(audio_array .* audio_array) / audio_len;
p_noise = p_signal / snr;

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

noisy_audio = awgn + audio_array;
```

“lab2part5_1.m”

```
audioFilePath = 'TotalNumber.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
audio_array = resample(originalAudioData, targetSampleRate, originalSampleRate);
audio_len = length(audio_array);

audioFilePath = 'n1.flac';
[originalAudioData, originalSampleRate] = audioread(audioFilePath);
targetSampleRate = 8192;
filter = resample(originalAudioData, targetSampleRate, originalSampleRate);

for snr = 0.01:-0.001:0.001
    p_signal = sum(audio_array .* audio_array) / audio_len;
    p_noise = p_signal / snr;
    rng (5)
    awgn = sqrt ( p_noise ).* randn ([ audio_len , 1]);
    noisy_audio = awgn + audio_array;
    filter_flip = flipud(filter);
    output = ConvFUNC(noisy_audio, filter_flip);
    i = 11 - snr * 1000;
    subplot(5,2,i)
    plot(output)
    xlabel("time")
    ylabel("output")
    title(['SNR = ', num2str(snr), ' - Output vs. Time']);
end
```

“lab2part5_2.m”