Full Name: Orkun İbrahim Kök

Department: EEE

Course Code: EEE321

Section: 02

Experiment Number: 02

Date: 28.02.2024

Lab 2 Report

Introduction:

The purpose of this experiment is understanding and implementing convolution and cross-correlation operations both in discrete and continuous time cases.

Analysis:

Part 1.1 Defining Convolution

The computation and derivation for this part can be found at the end of this report.

In order to provide graphical results, let's consider ξ[n]u[n] = u[n] – u[n-3] and η[n]u[n] = u[n] – u[n-4]. Convolution operation is represented as Ψ[n] = ξ[n]u[n] * η[n]u[n]. Length of η[n]u[n] (N_ξ) will be three (3) and length of η[n]u[n] (N_η) will be four (4). Figure 1 shows the convolution of these two discrete sequences and the output.

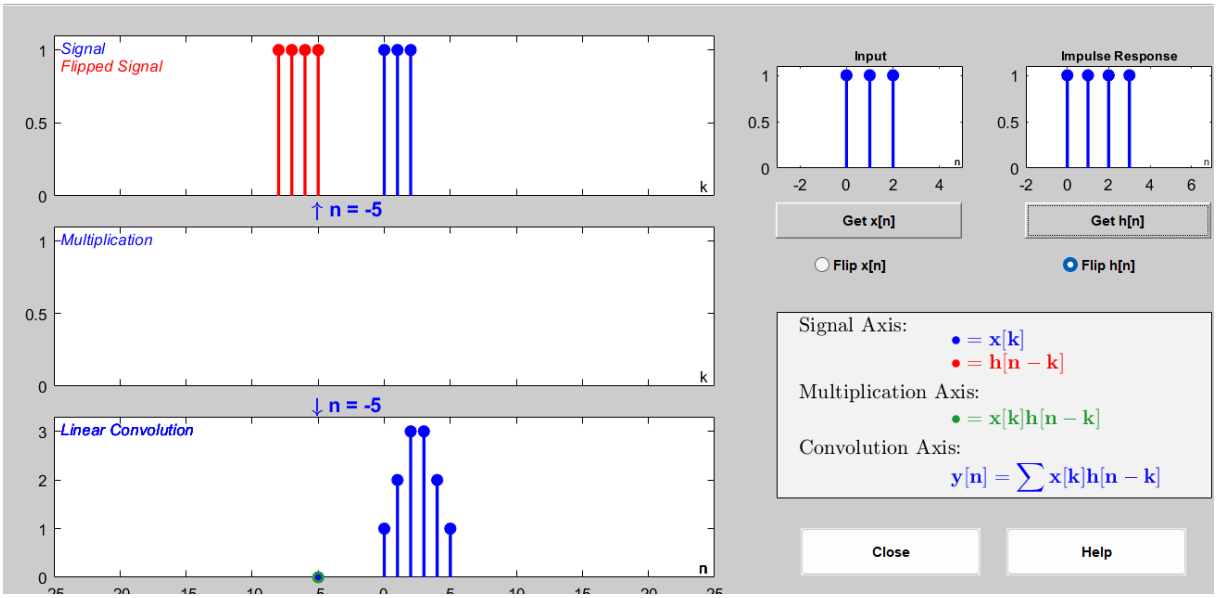


*Figure 1: Ψ[n] = ξ[n]u[n] * η[n]u[n]*

As seen from the graph, length of Ψ[n] turned out to be 6. Therefore, we can generalize it and can say length of the convolution is equal to $N_\xi + N_\eta - 1$. To explain what is discrete convolution in plain language, I can say that it is the open summation of two sequences, where one of the sequence is flipped with respect to n = 0 axis.

$$\Psi[n] = \xi[n]u[n] * \eta[n]u[n]$$

$$\sum_{k=0}^{\min(N\xi,\ N\eta)} \xi[k] \times \eta[n-k]$$

Here, because both ξ[n] and η[n] sequences are multiplied with u[n] (unit-step), open summation starts from 0 rather than - infinity.

Part 1.2 Evaluating Convolution

All derivation and computation for this part again can be seen at the end of the report.

Part 2.1 Implementing Convolution

For this part, I wrote a MATLAB function which takes two discrete sequences as its input and evaluates the convolution of these sequences. The MATLAB code for this part is available at Appendix. The code basically calculates the length of the convolution, and from that, uses open summation of multiplication of two sequences from 1 to length of the convolution.

Part 2.2 Testing the Convolution Function

In this part, question in Part 1.2 a) is wanted to be solved with using ConvFUNC function. The input sequences (ξ[n]) are same with each other and given below.

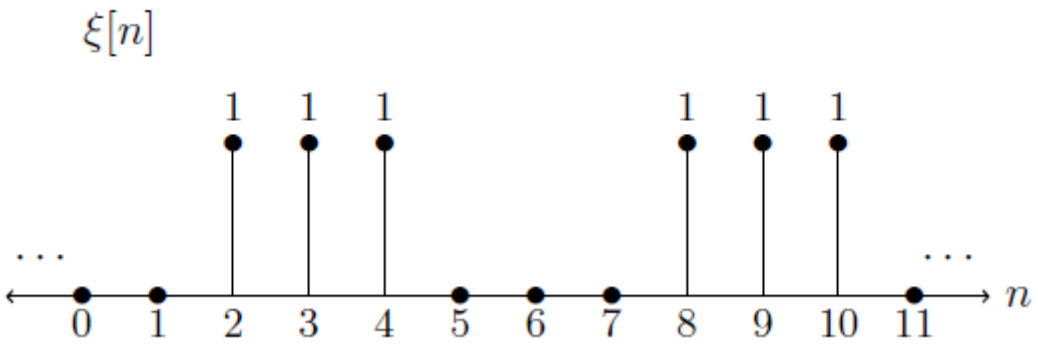


Figure 2: ξ[n] input sequence

Ψ[n] = ξ[n] * ξ[n], result of the convolution graph is given in below figure.

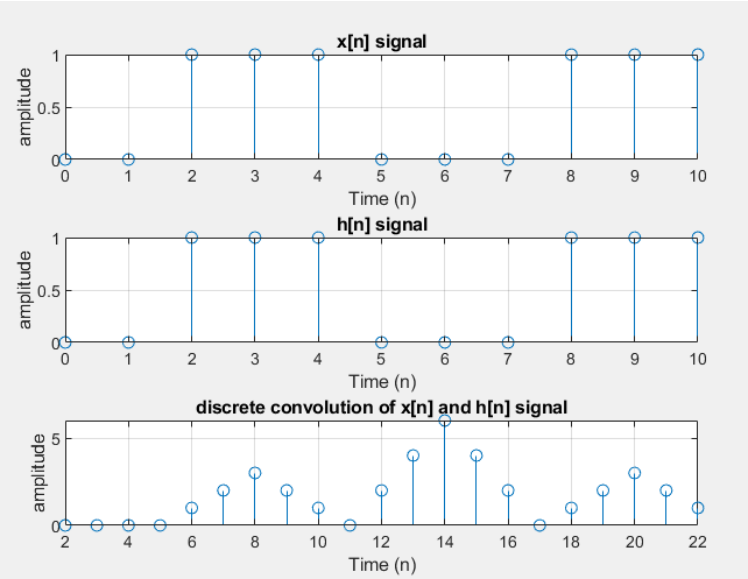


Figure 3: Result of the convolution

Part 3.1 Creating Convolution Animation

In this part, it is asked to create an animation on the MATLAB plot which shows how convolution is performed step-by-step. To do it, I first defined the signals which are going to be convolved. First signal is x(t) = u(t+5)-u(t-5) and second signal is h(t) = u(t+2.5)-u(t-2.5). As seen, these signals are in continuous time case, thereforei in order to use my function ConvFUNC, I used sampling in order to convert them to discrete time sequences. The two graphs at the top shows the input signals. Bottom-left corner graph shows both of the signals at the same time. In this figure, h(t) signal is flipped with respect to t=0 axis, and it is animated as shifted to right. Convolution is simply the area that both signals coincides. Lastly, bottom-right corner graph shows the final result of the convolution as a graph.

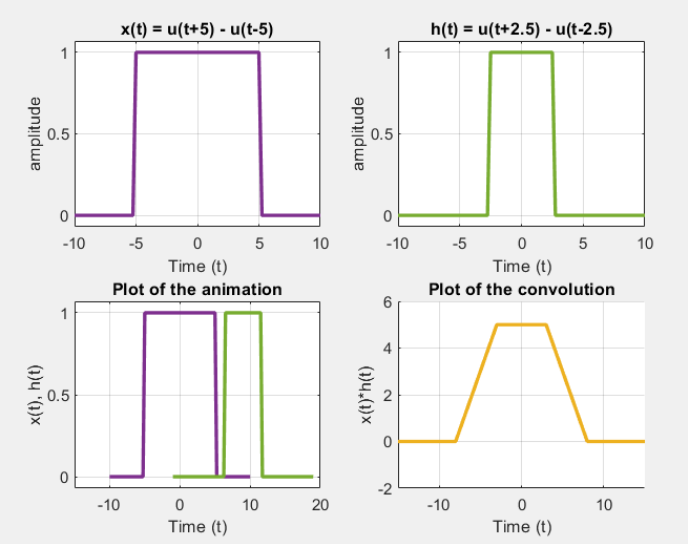


Figure 4: Part 3.1 Graph

Part 4.1 Defining Cross-Correlation

The computation and derivation for this part can be seen in Appendix.

The only difference of cross correlation from convolution is flipping one of the inputs. Cross correlation is showns as $\Psi[n] = x[n] \star h[n]$. Let's consider input sequences as $\xi[n]u[n]$ and $\eta[n]u[n]$. The length of $\xi[n]u[n]$ is N_ξ and length of $\eta[n]u[n]$ is N_η. As in convolution, again the length of the cross correlation $\Psi[n]$ will be $N_\xi + N_\eta - 1$. The relation between cross correlation and convolution is shown below.

$$(x \star h)[n] = \left(x * h_{flipped}\right)[n]$$

$$where\ h_{flipped} = h[-n]$$

The open summation form of cross correlation is shown below (Because of the presence of unit step (u[n]) sequence, open summation starts from 0 rather than - infinity.

$$\Psi[\mathrm{n}] = \xi[\mathrm{n}]\mathrm{u}[\mathrm{n}] \star \eta[\mathrm{n}]\mathrm{u}[\mathrm{n}]$$

$$\sum_{k=0}^{\min(N\xi,\ N\eta)} \xi[\mathrm{k}] \times \eta[\mathrm{k} - \mathrm{n}]$$

Part 4.2 Building a Basic Speech Recognition Algorithm

In this part, a basic speech recognition algorithm is built in MATLAB. Doing the calculation written in the requirements, following results are found (ID = 22003656):

$$\rho = \{0, 2, 6\}$$

$$\lambda = \{1, 4, 7, 8, 9\}$$

$$\Delta = 3$$

$$\Delta_\rho = 1$$

$$\Delta_\lambda = 4$$

From these values n_1 and n_2 results are found.

$$n_1 = 0$$

$$n_2 = 8$$

After finding these values, I recorded my voice while pronouncing my ID number for 10 seconds long. Then, using Python, I generated a text-to-speech version of my ID.

Then I extracted the n1 pronounciation from the original ID voice signal. As seen şn the figure below, I found the start and end points in order to extract that portion of the sound.

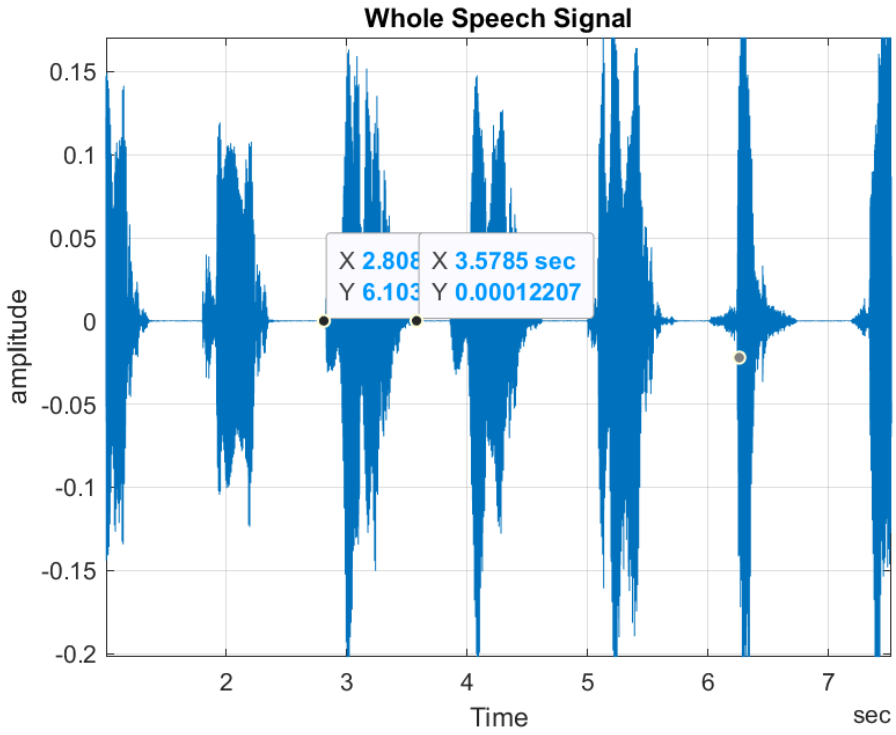


Figure 5: Start and end points of n1

Figure 6 shows the plots of actual speech signal, extracted n1 signal, and flipped n1 signal.

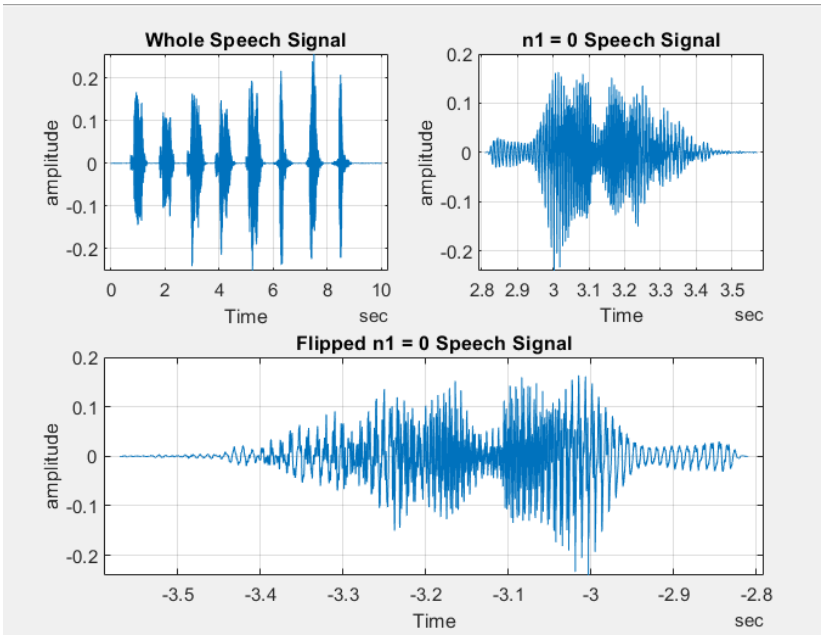


Figure 6: Figure of speech signal and extracted n1

After finding the extracted n1 sound, I used ConvFUNC function to compute cross correlation of whole speech signal and extracted n1 signal. As mentioned above in Part 4.1, while doing cross correlation, one of the input signals should be flipped, therefore, I flipped extracted n1 sound signal.

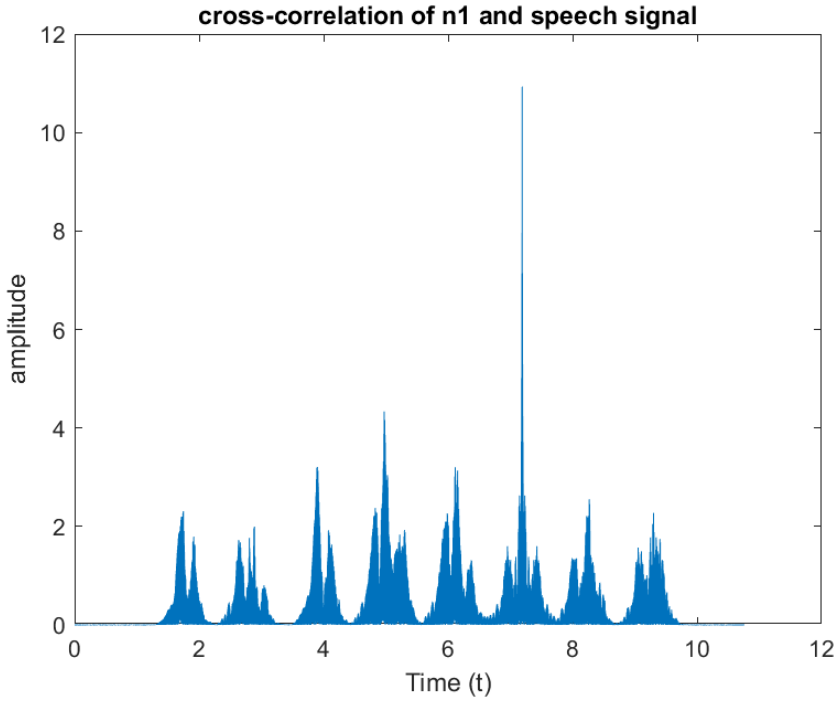


Figure 7: Cross correlation of whole speech signal and n1 signal

I performed the same cross correlation opeation but this time I observed squared and quadrupled cases. Following two figures will depict the graphs respectively.

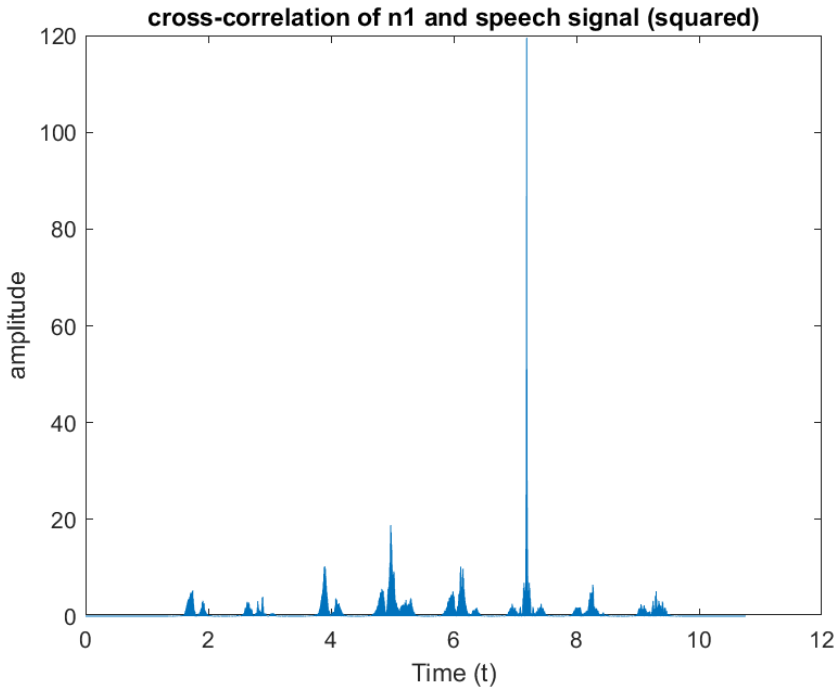


Figure 8: Squared cross correlation graph

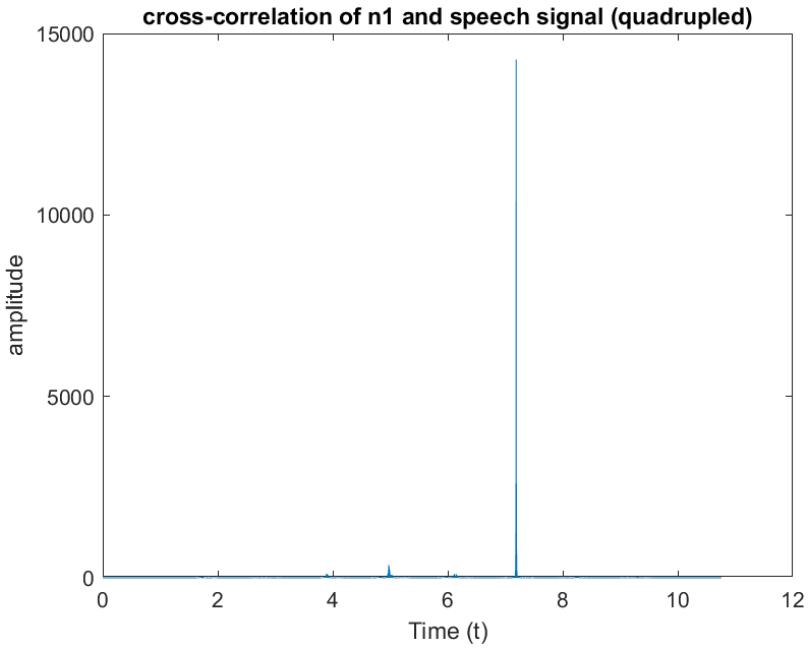


Figure 9: Quadrupled cross correlation graph

As seen from the graphs, there is a peak point. This peak point corresponds to flipped n1 signal. In Figure 7, speech signal's waveforms can be seen clearly, and the peak points amplitude is seen as around 11. In Figure 8, because it is squared, the amplitude of the peak point is increased to 11 squared which is something around 121. The other waveforms of the speech signal has less amplitude in magnitude, therefore these waveforms got smaller compared to peak point. And lastly, in Figure 9, because of the 4^{th} power, amplitude value increased to 4^{th} power of 11 which is something around 14600. Again, because the amplitude of the peak point is increased more rapidly due to nonlinear increase compared to whole speech signal, Remained waveforms of the speech signal now become invisible to see.

The figure below shows the convolution operation between speech signal and n1 signal.

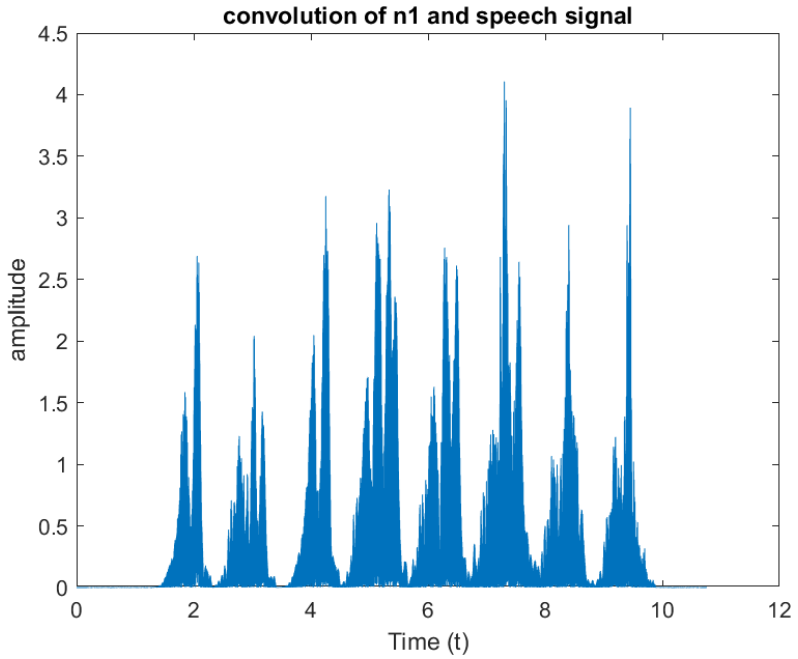


Figure 10: Convolution of speech signal and n1

Observed from the graph, when I performed convolution operation rather than cross correlation, peak point is not that obvious compared to cross correlation graph (Figure 7).

Figure 11 shows the text-to-speech algorith created ID signal.

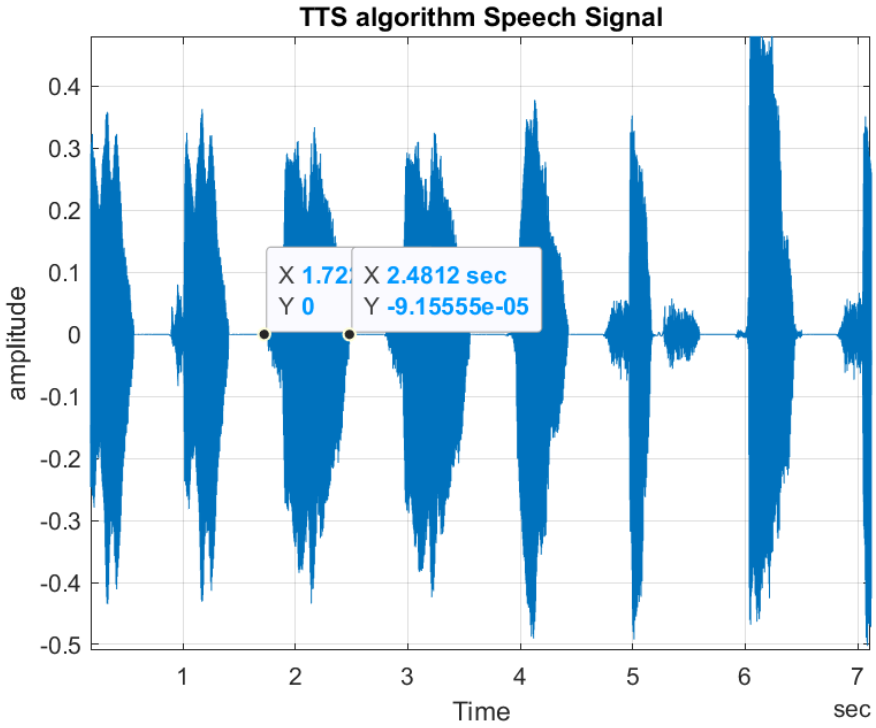


Figure 11: TTS Speech signal

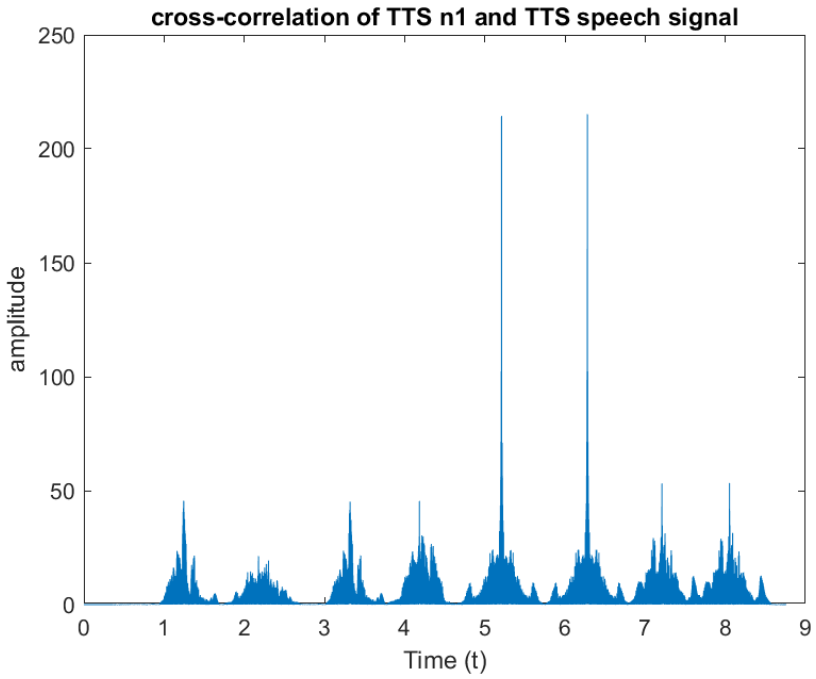


Figure 12: Cross realtion of TTS n1 and TTS speech signal

As seen in Figure 12, there are 2 peak points. The reason behind that is I assume because the sound is generated from computer, repeating sounds are exatcly the same with each other,

there cross correlation of two same things ended up generating a peak point as seen in figure above. Comparing the results with Figure 7, which is the cross correlation of speech signal and n1 both generated by me, ended up with having only one peak point. I think this may be caused because of the envoirenment noise and lack of experimenter's sensitivity while pronouncing the same numbers.

Part 5.1 Observing the Effects of SNR

In this part, the effect of SNR (ratio of the power of signal to the power of noise) is observed. The equation of SNR is given below.

$$SNR = \frac{P_{signal}}{P_{noise}}$$

I loaded the speech signal of my ID that I generated. Then I calculated the power of my sound signal (Equation is given below).

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

$$where\ N_x\ is\ nubmer\ of\ sample$$

After calculating it, I used AWGN (additive white Gaussian noise) as the noise signal. I generated it by using a MATLAB code. After that, I added the noise signal and speech signal to get the final results of the noisy speech audio.

I repeated the process for different SNR values varying from 10 to 0.1 to 0.001. The power of the noise (P_{noise}) values changes acccordingly to varying SNR values. The corresponding P_{noise} results are given for different SNR values below:

$$P_{noise} = 0.000778\ for\ SNR = 10$$

$$P_{noise} = 0.0778\ for\ SNR = 0.1$$

$$P_{noise} = 7.7756\ for\ SNR = 0.001$$

As SNR values decrease, power of the noise increases, which result in less hearable audios.

Part 5.2 Detecting the SNR Limit

This part is similar to Part 5.1. The only difference is showing the plots of noisy audio signal for SNR values from 0.01 to 0.001 decrementing by 0.001. The loaded audios are speech signal generated by me and n1 audio generated by text-to-speech algorithm.

The corresponding plot is available in Figure 13.

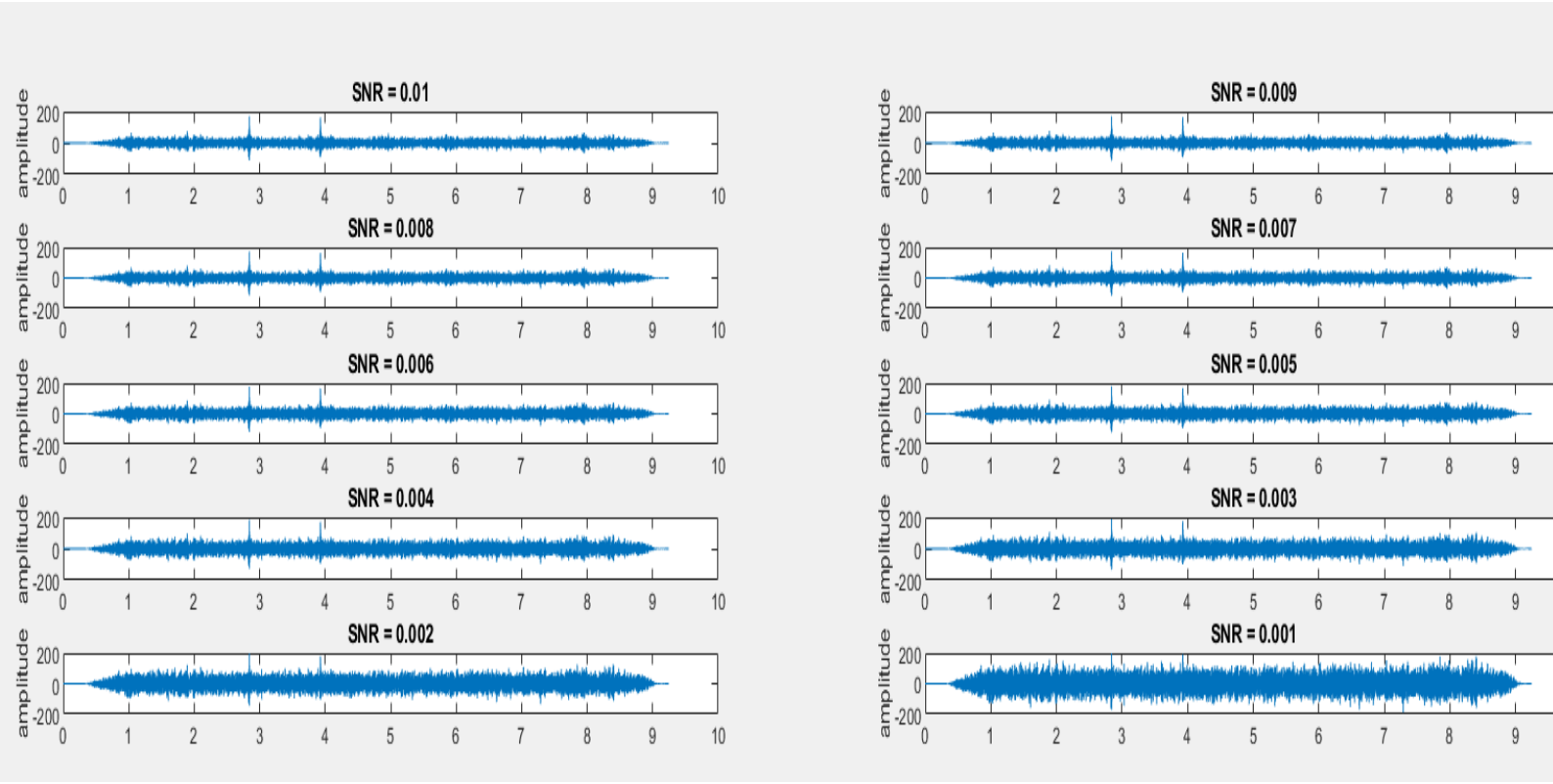


Figure 13: Varying SNR values and corresponding plots for cross correlation

The x-axis shows the time values for the audio signal. (Labeling x axis distorts the image therefore i cannot label x axis as time).

As seen from Figure 13, from SNR values 0.1 to 0.002, there are two (2) peak points at t=2.8 and t=3.8. These are the points where I say n1 number in my speech signal. However, when SNR value is 0.001, as seen from the bottom right corner plot, peak points cannot be distinguished. Therefore, when SNR value is equal to 0.001, n1 number (0) cannot be detected.

Appendix:

Part 2.1

```
function y = ConvFUNC(x, h)
    len_x = length(x);
    len_h = length(h);
    len_conv = len_x + len_h - 1;
    y = zeros(1, len_conv);
    for n = 1:len_conv
        k_val = max(1, n – len_h + 1):min(len_x, n);
        y(n) = sum(x(k_val) .* h(n - k_val + 1));
    end
end
```

Part 2.2

```
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 = ConvFUNC(x, h);
disp(y)

figure;
subplot(3,1,1);
stem(0:length(x)-1, x)
title('x[n] signal')
xlabel('Time (n)')
ylabel('amplitude')
grid on;

subplot(3,1,2);
stem(0:length(h)-1, h)
title('h[n] signal')
xlabel('Time (n)')
ylabel('amplitude')
grid on;

subplot(3,1,3);
stem(2:length(y)+1, y)
title('discrete convolution of x[n] and h[n] signal')
xlabel('Time (n)')
ylabel('amplitude')
grid on;
```

Part 3

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

tx_start = -5;
tx_end = 5;
x = (t>=tx_start & t<=tx_end);
th_start = -2.5;
th_end = 2.5;
h = (t>=th_start & t<=th_end);

n = -10:1:10;
```

```
x_n = (n>=tx_start & n<=tx_end);
h_n = (n>=th_start & n<=th_end);

y = ConvFUNC(x_n, h_n);

figure;
subplot(2,2,1);
plot(t, x, 'LineWidth', 2, 'Color', "#7E2F8E")
title('x(t) = u(t+5) - u(t-5)');
xlabel('Time (t)')
ylabel('amplitude')
ylim padded
grid on;

subplot(2,2,2);
plot(t, h, 'LineWidth', 2, 'Color', "#77AC30")
title('h(t) = u(t+2.5) - u(t-2.5)');
xlabel('Time (t)')
ylabel('amplitude')
ylim padded
grid on;

yeni_n = -20:1:20;
subplot(224)
title('Plot of the convolution')
xlabel('Time (t)')
ylabel('x(t)*h(t)')
grid on;
ylim([-2 6])
xlim([-15 15])
g = animatedline('LineWidth', 2 ,'Color', "#EDB120");
for k = 1:length(yeni_n)
    addpoints(g, yeni_n(k), y(k))
    drawnow;
    pause(0.1);
end

subplot(2,2,3);
x_plot = plot(t, x, 'LineWidth', 2, 'Color', "#7E2F8E");
hold on;
h_plot = plot(t, h, 'LineWidth', 2, 'Color', "#77AC30");
hold off;
xlabel('Time (t)')
ylabel('x(t), h(t)')
title('Plot of the animation')
xlim([-15 20])
ylim padded;
grid on;

ii = 100;
for i = 1:ii
    t_shift = i * (24 / ii);
    new_t = -(t - t_shift+15);
    set(x_plot, 'XData', t, 'YData', x)
    set(h_plot, 'XData', new_t, 'YData', h)
    pause(0.1)
end
```

Part 4.2 (Speech signal generated by me cross convoution)

```
ID = 22003656;
p = {0, 2, 6};
a = {1, 4, 7, 8, 9};
x = mod(ID, 7);
x_p = mod(x, length(p)) + 1;
x_a = mod(x, length(a)) + 1;
n1 = p(x_p);
n2 = a(x_a);

Fs = 8192;
ch = 1;
nbits = 16;
recDuration = 10;
fileName = 'ID_speak.flac';
recorder = audiorecorder(Fs, nbits, ch);
disp('start speaking');
recordblocking(recorder, recDuration);
disp('end')
audioData = getaudiodata(recorder);
audiowrite(fileName, audioData, Fs)


file_location = 'C:\Users\DELL\Desktop\EEE321 Labs\Lab2\audio\ID_speak.flac';
Ts = 1/8192;
fs = 8192;
[y, Ts] = audioread(file_location);

n = length(y);
disp(n/fs)

t = seconds(0:1/fs:(size(y,1)-1)/fs);

start_time = 2.81;
end_time = 3.57;
t1 = seconds(2.81:1/fs:3.57);
y1 = y(start_time*fs:end_time*fs);
soundsc(y1,fs)

subplot(221)
plot(t, y)
grid on;
title('Whole Speech Signal')
ylabel('amplitude')
xlabel('Time')

subplot(222)
plot(t1, y1)
grid on;
title('n1 = 0 Speech Signal')
xlabel('Time')
ylabel('amplitude')

subplot(2,2,3:4)
y2 = y(end_time*fs:-1:start_time*fs);
t2 = seconds(-3.57:1/fs:-2.81);
plot(t2, y2)
grid on;
```

```
title('Flipped n1 = 0 Speech Signal')
xlabel('Time')
ylabel('amplitude')

figure;
v = abs(ConvFUNC(y2, y));
t_v = (0:length(v)-1)/fs;
plot(t_v,flip(v))
title('cross-correlation of n1 and speech signal')
ylabel('amplitude')
xlabel('Time (t)')
```

Part 4.2 (TEXT-TO-SPEECH cross convolution code)

```
file_location = 'C:\Users\DELL\Desktop\EEE321 Labs\Lab2\audio\TotalNumber.flac';
Ts = 1/24342;
fs = 24342;
[y, Ts] = audioread(file_location);

n = length(y);
disp(n/fs)

t = seconds(0:1/fs:(size(y,1)-1)/fs);

start_time = 1.72;
end_time = 2.48;
t1 = seconds(1.72:1/fs:2.48);
y1 = y(start_time*fs:end_time*fs);
soundsc(y1,fs)


subplot(221)
plot(t, y)
grid on;
title('Whole Speech Signal')
ylabel('amplitude')
xlabel('Time')

subplot(222)
plot(t1, y1)
grid on;
title('n1 = 0 Speech Signal')
xlabel('Time')
ylabel('amplitude')



figure;
v = abs(ConvFUNC(flip(y1), y));
t_v = (0:length(v)-1)/fs;
plot(t_v,flip(v))
title('cross-correlation of TTS n1 and TTS speech signal')
ylabel('amplitude')
xlabel('Time (t)')
```

Part 5.1

```
file_path = 'C:\Users\DELL\Desktop\EEE321 Labs\Lab2\audio\TotalNumber.flac';
Ts = 1/8192;
[audio_array, Ts] = audioread(file_path);

audio_len = length(audio_array);
p_signal = (1/audio_len)*sum(audio_array(1:audio_len).*audio_array(1:audio_len));

p_noise = p_signal/0.001;
disp(p_noise);
rng (5)
awgn = sqrt ( p_noise ).* randn ([ audio_len , 1]);
noisy_audio = audio_array + awgn;
soundsc(audio_array, Ts)


%as SNR value decreases, it is getting more difficult to hear the sound.
% 0.000778(SNR = 10)
% 0.0778(SNR = 0.1)
% 7.7756(SNR = 0.001)
```

Part 5.2

```
file_location1 = 'C:\Users\DELL\Desktop\EEE321 Labs\Lab2\audio\TotalNumber.flac';
file_location2 = 'C:\Users\DELL\Desktop\EEE321 Labs\Lab2\audio\0.flac';
Ts = 1/8192;
[audio_array, Ts] = audioread(file_location1);
[filter, Ts] = audioread(file_location2);
snr = 0.01:-0.001:0.001;
for i = 1:length(snr)
    snr_values = snr(i);
    audio_len = length(audio_array);
    p_signal = (1/audio_len)*sum(audio_array.^2);
    p_noise = p_signal/snr_values;
    rng (5)
    awgn = sqrt ( p_noise ).* randn ([ audio_len , 1]);
    noisy_audio = audio_array + awgn;

    output = ConvFUNC(noisy_audio, flip(filter));
    t_output = (0:length(output)-1)/Ts;
    subplot(length(snr), 2, i)
    plot(t_output, output)
    title(['SNR = ' num2str(snr_values)]
end
```

Part 1.1 $\psi[n] = \xi[n]u[n] * \eta[n]u[n]$

$$= \sum_{k=-\infty}^{\infty} \xi[k]u[k] \cdot \eta[n-k] \cdot u[n-k]$$

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

length of $\psi[n] = N_\xi + N_\eta - 1$

Part 4.1

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

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

flip one of the inputs and do the convolution operation

$$= \sum_{k=0}^{\min(N_\xi, N_\eta)} \xi^*[k] \cdot \eta[n+k]$$

length of $\psi[n] = N_\xi + N_\eta - 1$ (same as discrete convolution)

If we time-reverse one of the sequences say $\eta[n]u[n]$, and call the time-reversed sequence $\eta_R[n]$. Then, doing the convolution between $\xi[n]u[n]$ and $\eta_R[n]u[n]$, the result is discrete cross-correlation.

$N_1 = 0 \quad N_2 = 8$

↳ 1st one 2.81 sec – 3.57 sec

↳ 2nd one 3.81 sec – 4.61 sec

Part 1.2:

a) $\xi[n]$: $\delta[n-2]+\delta[n-3]+\delta[n-4]+\delta[n-8]+\delta[n-9]+\delta[n-10]$

$\xi[n] * \xi[n]$ 0.5.1

$=\delta[n+4]+\delta[n-5]+\delta[n-6]+\delta[n-10]+\delta[n+11]+\delta[n-12]$

$+\delta[n-5]+\delta[n-6]+\delta[n-7]+\delta[n-11]+\delta[n-12]+\delta[n-13]$

$+\delta[n-6]+\delta[n-7]+\delta[n-8]+\delta[n-12]+\delta[n-13]+\delta[n-14]$

$+\delta[n-10]+\delta[n-11]+\delta[n-12]+\delta[n-16]+\delta[n-17]+\delta[n-18]$

$+\delta[n-11]+\delta[n-12]+\delta[n-13]+\delta[n-17]+\delta[n-18]+\delta[n-19]$

$+\delta[n-12]+\delta[n-13]+\delta[n-14]+\delta[n-18]+\delta[n-19]+\delta[n-20]$

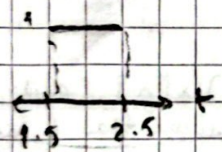
$=\delta[n-4]+2\delta[n-5]+3\delta[n-6]+2\delta[n-7]+\delta[n-8]+2\delta[n-10]$

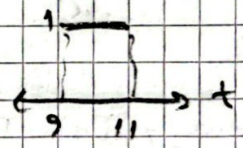
$+4\delta[n-11]+6\delta[n-12]+4\delta[n-13]+2\delta[n-14]+\delta[n-16]+2\delta[n-17]$

$+3\delta[n-18]+2\delta[n-19]+\delta[n-20]$

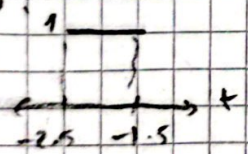
c) $\xi(t)$: $u(t-1.5)-u(t-2.5)$ $\quad$ $\eta(t)$: $u(t-9)-u(t-11)$

i)

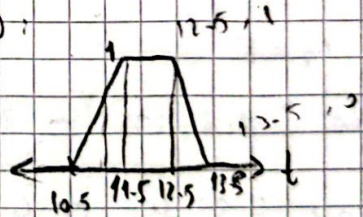




$\xi(-t)$:

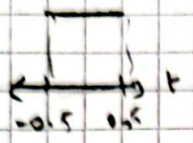


$\xi(t) * \eta(t)$:

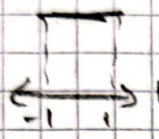


$$\gamma_1(t):\begin{cases} 0 & \text{for } t<10.5 \\ t-10.5 & \text{for } 10.5\le t<11.5 \\ 1 & \text{for } 11.5\le t<12.5 \\ -t+13.5 & 12.5<t<13.5 \\ 0 & \text{for } t\ge 13.5 \end{cases}$$

ii) $\varepsilon(t+2) = u(t+0.5) - u(t-0.5)$



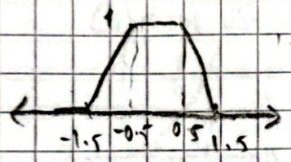
$\eta(t+0) = u(t-1) - u(t+1)$



$t+1.5$

$\eta(-2)$ = same as $\eta(2)$

$\varepsilon(t+2) * \eta(t+0)$:



$$\psi_2(t): \begin{cases} 0 \text{ for } t < -1.5 \text{ and } t > 1.5 \\ t+1.5 \text{ for } -1.5 < t < -0.5 \\ 1 \text{ for } -0.5 < t < 0.5 \\ -t+1.5 \text{ for } 0.5 < t < 1.5 \end{cases}$$

$\psi_3(t) \rightarrow$ shift $\psi_2(t)$ 12 points right

$\psi_3(t)$ =

same as $\psi_1(t)$

So, shifting $\varepsilon(t)$ and $\eta(t)$ separately is same with shifting the convolution result for the same amount.

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

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

$\lambda(t) =$: (c)

$\lambda(t) = \xi(t) * \eta(t) * \xi(t) = \underbrace{\xi(t) * \xi(t)} * \eta(t)$

$= \int_{-\infty}^{\infty} \xi(\tau)\xi(2-t+\tau)\,d\tau$

$= \int_{0}^{1} \xi(\tau)\xi(2-t+\tau)\,d\tau = \int_{0}^{1} u(\tau)u(1-2+t-\tau)\,d\tau$

$= \int_{0}^{t-1} d\tau = t-1 \Big|_{t=2}^{t=1} = (1-t+2)r(t-2) - r(t-1) =$

$= (3-t)r(t-2) - r(t-1) =$

$\lambda(t) = (t-1)r(t-1) - (t-2)r(t-2) - (3-t)r(t-2)$

$(\eta(t) = \eta(t)) \quad \eta(t) * \eta(t)$

$= \int_{-\infty}^{\infty} \eta(\tau)\eta(2-t+\tau)\,d\tau = \int_{0}^{2} \eta(2-t+\tau)\,d\tau$

$= \int_{0}^{1} [(2-t+\tau)r + (1-2+t-\tau)2 - (2-t+\tau)]\,d\tau$

$-z^2-(t^2+2tz-z^2)$

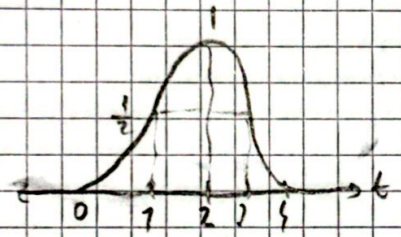
a) known $\int r(t) = \frac{t^2}{2}u(t)$

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

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

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

$$\psi(t) = \begin{cases} \frac{t^2}{2} & \text{for } 0 \le t \le 1 \\ -\frac{t^2}{2}+2t-1 & \text{for } 1 < t \le 3 \\ \frac{t^2}{2}-8t+8 & \text{for } 3 < t \le 4 \end{cases}$$



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

$$\psi(t) = \int_{-\infty}^{\infty} \varepsilon(z)\,\eta(t-z)\,dz = \int_{-\infty}^{\infty} e^{-\frac{z^2}{2}} \cdot 2e^{\frac{-(t-z)^2}{2}}\,dz$$

$$= 2\int_{-\infty}^{\infty} e^{\frac{-z^2-(t-z)^2}{2}}\,dz = 2\int_{-\infty}^{\infty} e^{\frac{-2z^2+2tz-t^2}{2}}\,dz$$

$$= 2\cdot e^{\frac{-t^2}{2}}\int_{-\infty}^{\infty} e^{\frac{-2z^2+2tz}{2}}\,dz = 2e^{\frac{-t^2}{2}}\sqrt{\pi}\,e^{\frac{t^2}{4}}$$

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

Gaussian