


Course Title:	Signals and Systems I
Course Number:	ELE532
Semester/Year (e.g.F2016)	F2023

Instructor:	Brendan Wood
--------------------	--------------

<i>Assignment/Lab Number:</i>	4
<i>Assignment/Lab Title:</i>	The Fourier Transform: Properties and Application

<i>Submission Date:</i>	December 3rd 2023
<i>Due Date:</i>	December 3rd 2023

Student LAST Name	Student FIRST Name	Student Number	Section	Signature*
Sooriyapperuma	Lathika	500904706	1	

*By signing above you attest that you have contributed to this written lab report and confirm that all work you have contributed to this lab report is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, an "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: <https://www.torontomu.ca/content/dam/senate/policies/pol60.pdf>

Table of Contents

Table of Contents	2
Objective	3
Section A: The Fourier Transform and its Properties.	3
Problem A.1.....	5
Problem A.2.....	5
Problem A.3.....	5
Problem A.4.....	5
Problem A.5.....	7
Problem A.6.....	7
Section B: Application of Fourier Transform	9
Problem B.1.....	9
Appendix	11
References.....	18

Objective

The purpose of this lab was to analyze, observe and study frequency domain characteristics of time waveforms. In doing so the properties of the Fourier Transform are observed, and in part B, the Fourier transform is used in a real-world scenario in which an audio waveform, a voice clip, is transmitted through a band-pass communications channel.

Section A: The Fourier Transform and its Properties.

Initially, several MATLAB functions and techniques which are used throughout the lab were noted.

Plotting the rectangular pulse seen in the lab manual [1, Figure 1] is done with the code presented below in Figure 1.

```
6      %% Create and plot rect pulse
7
8      N = 100;    %N = the number of samples for the signal
9      PulseWidth = 10;
10     t = [0:1:(N-1)]; %creates 100 element array w/ from 0 to 99 inclusive
11     x = [ones(1,PulseWidth), zeros(1,N-PulseWidth)];
12
13     figure('Name', 'Graph of Rect Pulse');
14     stairs(t,x); grid on; axis([-1,30,-0.1,1.1]);
15
```

Figure 1: Creating and Plotting Rectangular Pulse MATLAB Code.

Note that N , the number of samples for the signal, will be required since MATLAB cannot store continuous signals. Furthermore, the 100-element array t , which serves as the signal's values on its time axis, has the same length as the number of samples so that there is an input for every output. Finally, the y-axis values of the signal which are 1s and 0s are defined with simple and familiar MATLAB array definition tools, and the signal is plotted (as seen in Figure 9 on page 7, top left) with the **stairs()** function as opposed to **plot()** to better reflect the nature of impulses with their vertical (infinite) slope.

One might notice that the initial impulse from $y=0$ to $y=1$ at time $t=0$ cannot be seen. This is due to the nature of the stairs function, which displays the vertical line only when there is a transition. Since the signal begins at 1 at time 0, there is no change and thus the vertical line at $t=0$ cannot be seen.

The Fourier transform is applied to the rectangular pulse with the code depicted below in Figure 2. The in-built function **fft()** is used to populate the vector X_f which represents the Fourier-transformed signal. When the Fourier transform is applied to a function, it is no longer in the time domain. Instead, it is in the frequency domain. To reflect this in MATLAB, another vector of the same length as the time domain must be made but centered at a frequency of 0. In the code, this is done in lines 21 and 22, with 22 being an optional additional step to convert to radial frequency which may be more familiar in this context. The Fourier spectra for the pulse can be seen to the right of its corresponding pulse in the time domain on the top left of figure 9 on page 7.

Using MATLAB, the inverse Fourier Transform can also be taken using an inbuilt function. Figure 3 below demonstrates this. It is expected that the inverse Fourier Transform of $x(t)$ be same as the original $x(t)$ function. This is the case practically speaking, however computational rounding errors mean

that they are not quite the same. It is show that the discrepancies that do exist are insignificant by using the **areVectorsNearlyIdenticalFunction()** shown in Figure 4. Judging by the output seen in Figure 3, we note that the vectors are indeed nearly identical.

```

16    %% Obtain Fourier transformed function and plot fourier spectra
17
18    Xf = fft(x); %Perform fourier transform on rectangular pulse x.
19
20    %define the complex func's domain, w
21    f = [-(N/2):1:(N/2)-1]*(1/N);
22    w = 2*pi*f; %convert radial freq (rad/s)
23
24    figure('Name', 'Fourier Spectra for Pulse');
25    subplot(211); plot(w,fftshift( abs(Xf))); grid on; %plot amplitude spectra
26    subplot(212); plot(w,fftshift(angle(Xf))); grid on; %plot phase spectra
27

```

Figure 2:MATLAB code to obtain $F\{x(t)\}$ and plot its Spectra.

```

28    %% Perform inverse fourier transform and compare to orginal function x
29
30    xhat = ifft(Xf);    %perform the inverse fourier transform on Xf
31
32    %compare the vectors
33    result = areVectorsNearIdentical(x, xhat, 1e-5);
34    if result
35        disp('Vectors are near identical.');

Command Window



Vectors are near identical.



fx >>


```

Figure 3: Inverse Fourier Transform and comparison to original signal.

```

1  function areNearIdentical = areVectorsNearIdentical(vec1, vec2, tolerance)
2      % Check if vec1 and vec2 are near identical w/in the given tolerance.
3
4      % Check if the vectors have the same length
5      if length(vec1) ~= length(vec2)
6          error('Vectors must have the same length.');

```

Figure 4: MATLAB **areVectorsNearIdentical()** Function.

Problem A.1

The signal $x(t)$, the rectangular pulse previously mentioned, was analytically convolved with itself, and plotted, as seen in Figures 14-17 in the Appendix.

Problem A.2

As an alternative to convolution in the time domain, multiplication in the Fourier domain also yields the same result. Using the MATLAB code depicted in figure 5 below, $Z(\omega)$, $z(t)$'s counterpart in the Fourier domain is obtained.

```
27 %% A.2: compute  $Z(\omega) = X(\omega)X(\omega)$   
28 Xf = fft(x); %% Perform fourier transform on x  
29 Zf = Xf .* Xf; % Use fourier property to obtain covolution equivalent. |  
30
```

Figure 5: MATLAB Code to compute $Z(\omega)$, using Fourier Property.

Problem A.3

Then, the magnitude and phase spectra of $z(t)$ are plotted. See figure 6 below.

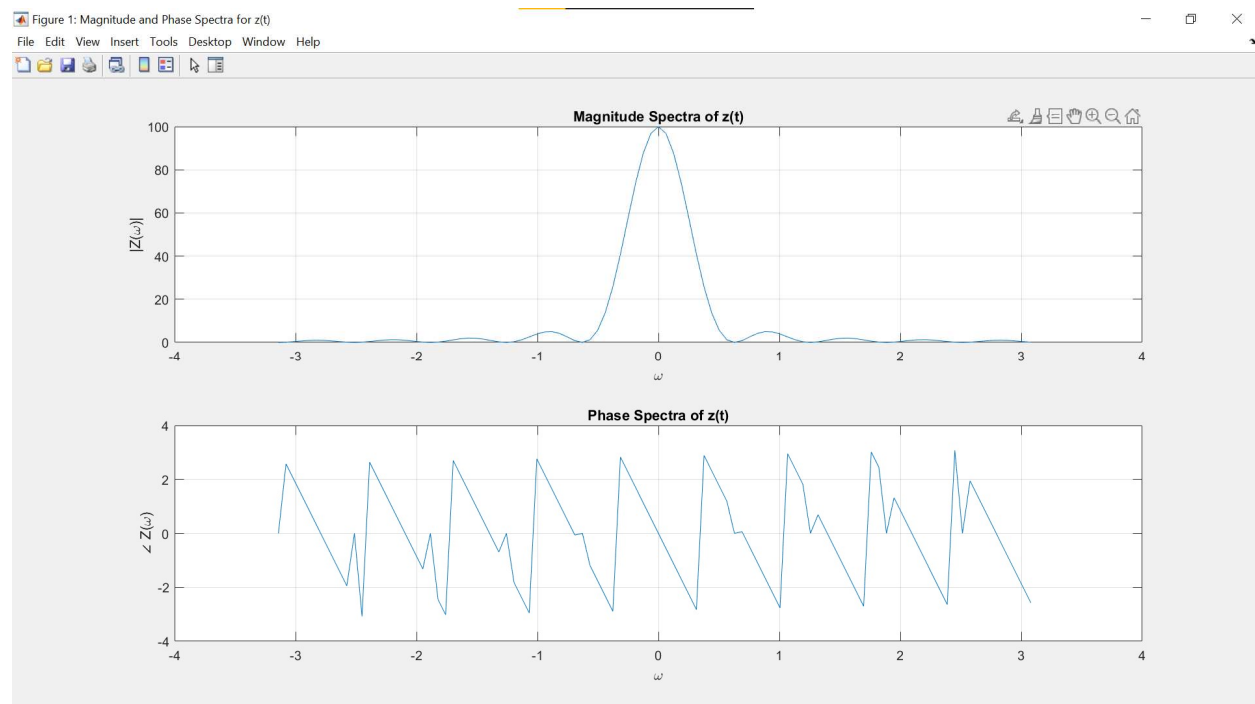


Figure 6: Magnitude and Phase Spectra of $z(t)$.

Problem A.4

MATLAB also has an in-built function to compute convolutions in the time domain, rather than transforming to the frequency domain and multiplying as previously done. The two convolutions can be compared to each other to demonstrate they are in fact the same. The code below in Figure 7 accomplishes this, and the resulting plots can be followed in the figure that follow, Figure 8.

```

43 % A.4 Comparing time vs. phasor domain MATLAB operations to find z(t)
44 figure('Name', 'z(t) found using time vs. fourier domain ops');
45
46 z = conv(x,x);
47 z = z(1:100); %remove the *irrelavant parts vector z.(2)
48
49 % Subplot graph of z(t) found using time domain operation
50 subplot(211); plot(t,z); grid on; axis([-10,110,-0.1,1.1]);
51 axis tight;
52 xlabel('t'); ylabel('z(t)');
53 title('z(t) found using MATLAB conv func');
54
55 %Perform the inverse fourier transform to Z(w)
56 zhat = ifft(Zf);
57
58 % Subplot graph of z(t) found using Fourier property (from A.2)
59 subplot(212); plot(t,zhat); grid on; axis([-10,110,-0.1,1.1]);
60 axis tight;
61 xlabel('t'); ylabel('z(t)');
62 title('z(t) found using Fourier Property');

```

Figure 7: MATLAB code to compare z(t) using time domain vs phasor domain operations.

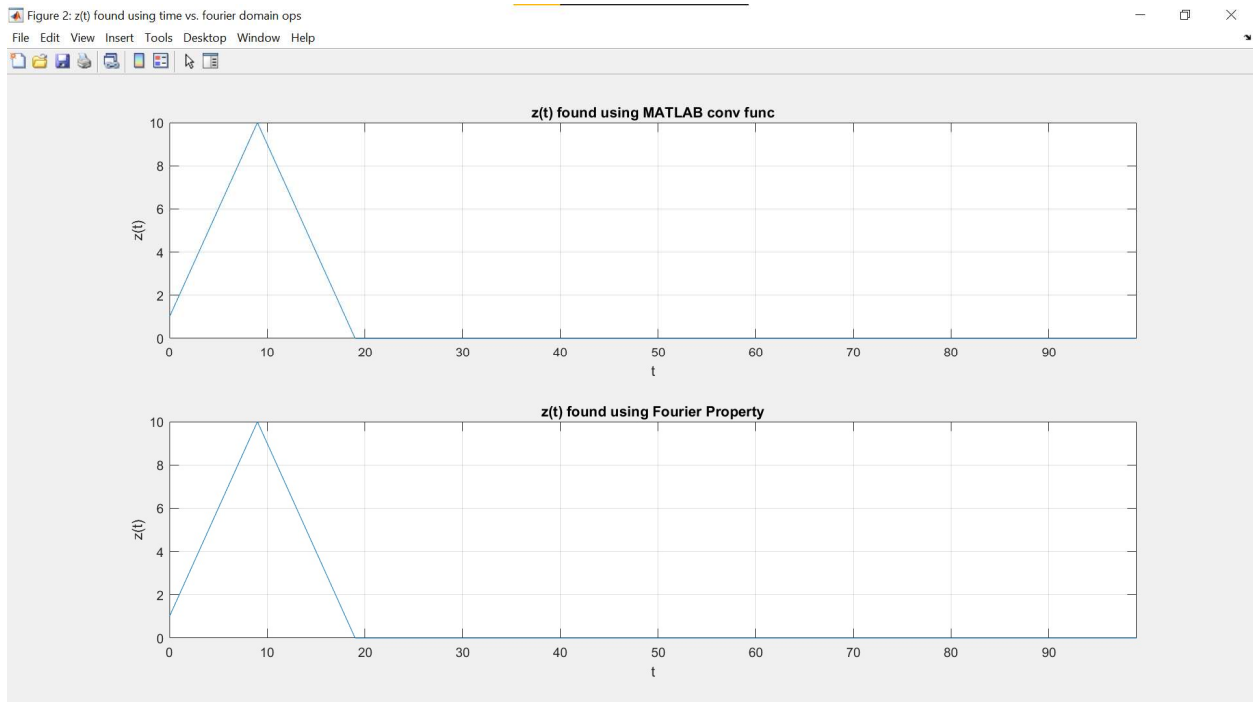


Figure 8: Comparison of convolutions using time domain vs phasor domain operations.

In Figure 8, the resulting convolutions are identical. They both differ slightly from the analytically computed Fourier transform, and the required time shift in both MATLAB computations is exactly 1 unit to the right. The Fourier property demonstrated is the convolution property, which is given by:

$$F\{x(t) * x(t)\} = X(\omega)X(\omega)$$

Problem A.5

Using the code seen in Figure 18, the Fourier Spectra of 3 rectangular pulses, each with varying pulse widths are plotted below in Figure 9.

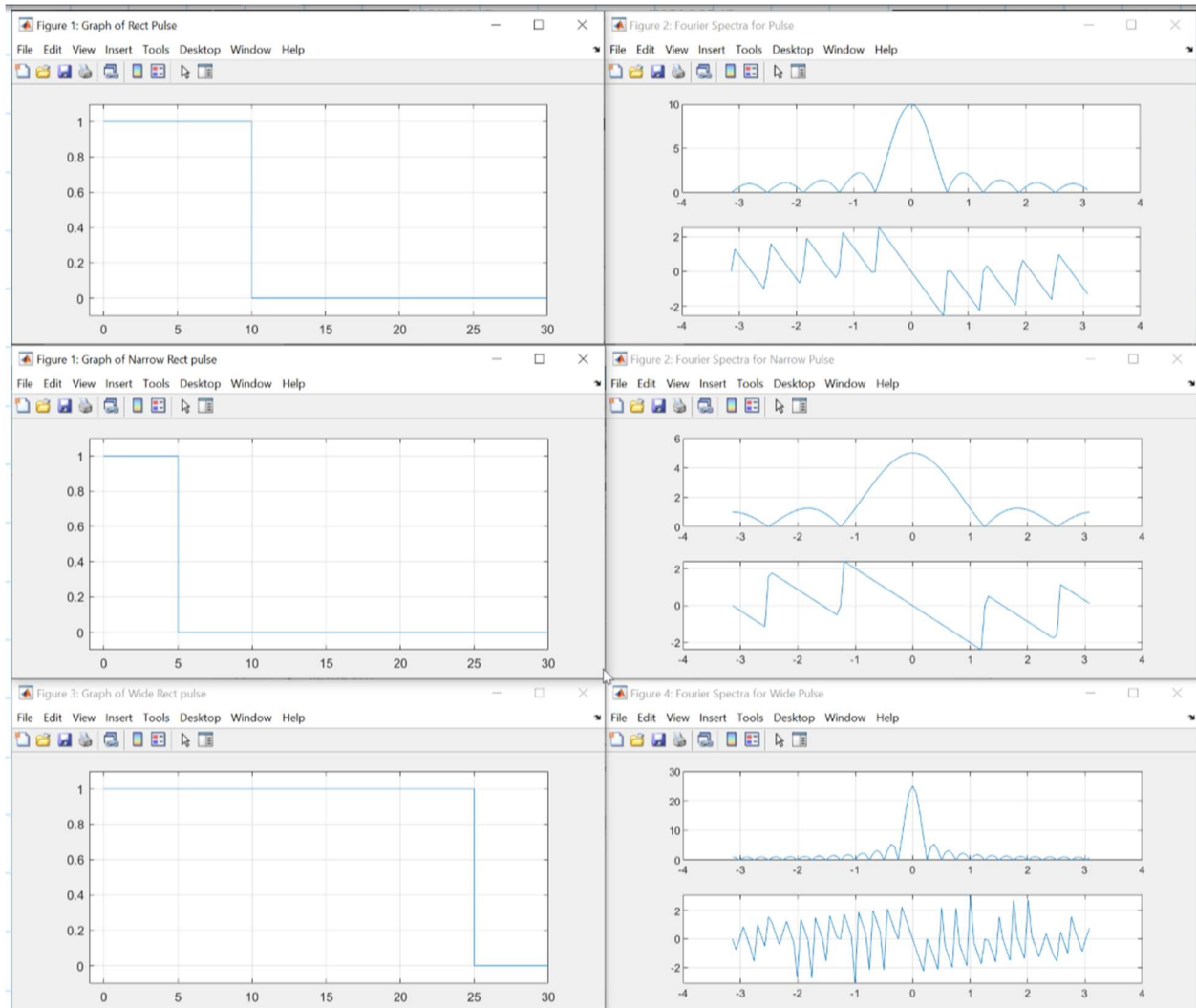


Figure 9: Comparison of Fourier Spectra of Rectangular pulses with different widths.

When ordering the pulses from narrowest to widest, it can be observed that the pulse width is inversely proportional to the signal's bandwidth, or in other words, the range of frequencies over which the signal is distributed. This demonstrates the Heisenberg Uncertainty Principle. The property demonstrated is time scaling.

Problem A.6

Three signals, $w_+(t) = x(t)e^{j(\pi/3)t}$, $w_-(t) = x(t)e^{-j(\pi/3)t}$ and $w_c = x(t)\cos(\pi/3)t$ are computed using the code shown in Figure 19-20 in the appendix. Their Fourier spectra are plotted and compared in the same code, by plotting them side by side as seen in Figure 10. The first two signals demonstrate the frequency shift property of the Fourier transform, given by the following equation:

$$F\{x(t)e^{j\omega_0 t}\} = X(\omega - \omega_0)$$

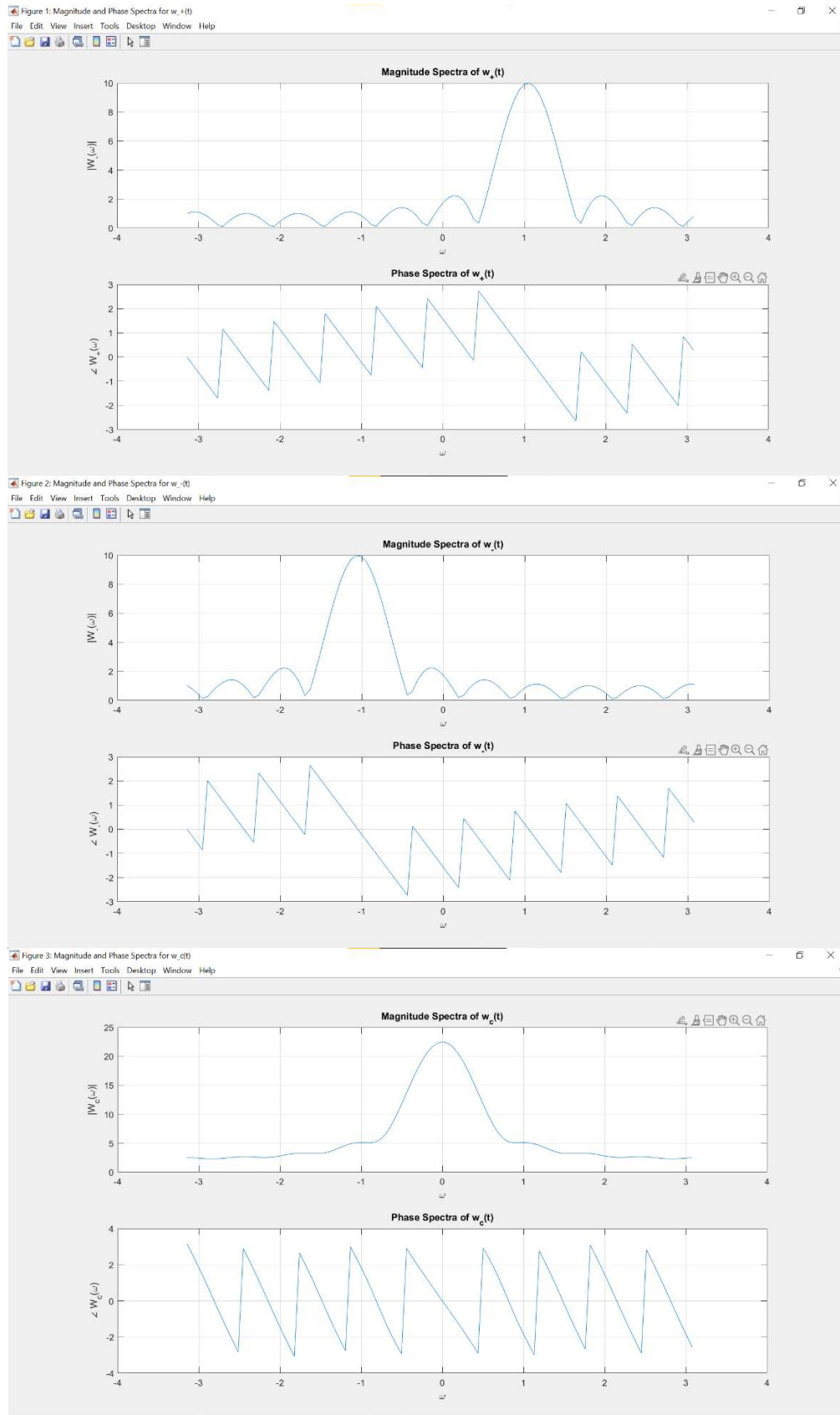


Figure 10: Fourier Spectra of w_+ , w_- , w_c .

Section B: Application of Fourier Transform

Problem B.1

The first step was to design an encoding system that will allow the transmission of the signal speech over the channel. This process is shown in block diagram form in Figure 11 below.

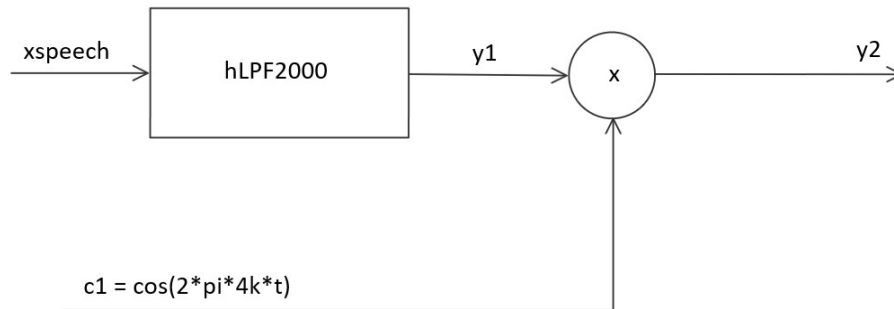


Figure 11: Block Diagram form encoding system.

The audio clip is convolved with the 2kHz low pass filter to fit it into a band of 4kHz width, which required since the width of the bandpass transmission channel is 4kHz (2kHz to 6 kHz).

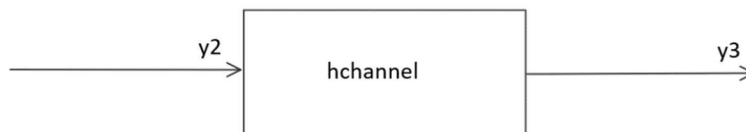


Figure 12: Block Diagram of Transmission.

The transmission of the signal can be modelled by convolving the compressed signal with the transmission channel signal.

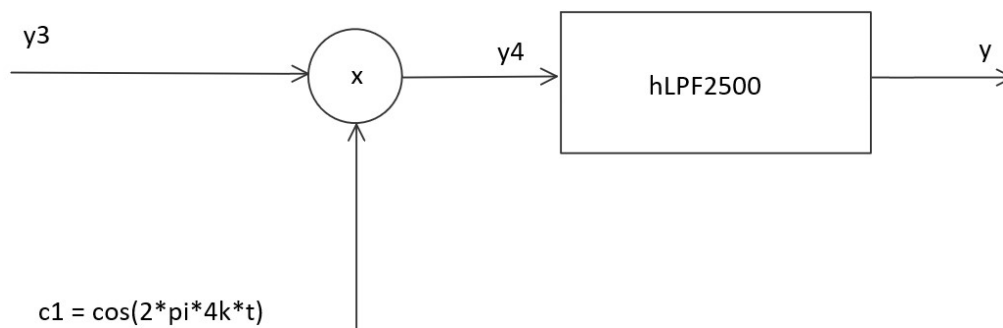


Figure 13: Block Diagram of Decoding.

Finally, the signal is decoded, as seen in figure 13. De-modulation is performed first and is done so using a technique known as coherent demodulation. This involves the multiplication of the signal with the original carrier signal. This type of de-modulation produces mirrored high frequency bands which need to be removed. To not needlessly degrade the quality of the audio clip by using the 2kHz low-pass filter, the de-modulated signal is instead convolved with the wider 2.5kHz low pass filter to obtain the final recovered audio clip, which can be listened to by running B1.m. The audio is clearly compressed- has a muffled sound but is still more perfectly understandable and more than serviceable.

This entire process is simulated in MATLAB with the code found in B1.m. The code for B1 is included in the appendix, and the B1 script itself is included in the zip file to easily be tested.

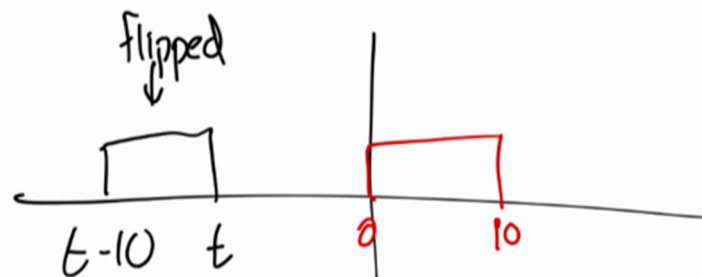
Appendix

$$z(t) = x(t) * x(t)$$

$$x(t) = \begin{cases} 1 & 0 < t < 10 \\ 0 & \text{o.w.} \end{cases}$$

$$z(t) = \int_{-\infty}^{+\infty} x(\tau) x(t-\tau) d\tau$$

Case 1: $t+10 < 0 \Rightarrow y(t) = 0$

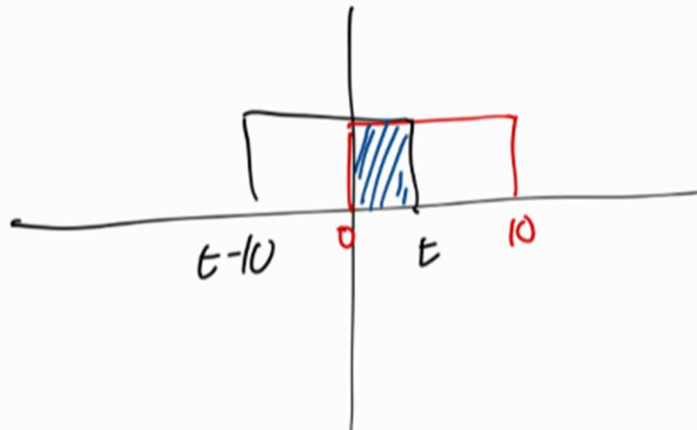


$$y(t) = \int_{-\infty}^0 x(\tau) x(t-\tau) d\tau$$

$= 0$

Figure 14: Analytical Calculation and Plot of $z(t)$ 1/4.

Case 2: $t < 10$



$$y(t) = \int_0^t x(\tau) x(t-\tau) d\tau$$

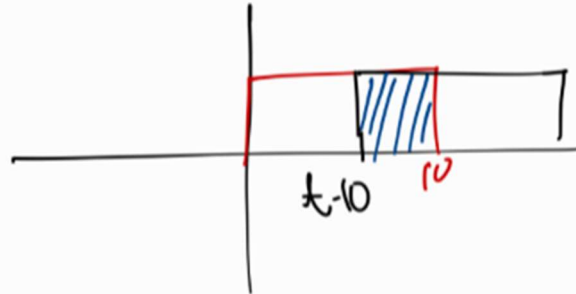
$$= \int_0^t (1)(1) d\tau$$

$$= [\tau]_0^t$$

$$y(t) = t$$

Figure 15: Analytical Calculation and Plot of $z(t)$ 2/4.

Case 3 : $0 < t-10 < 10$
 $10 < t < 20$



$$y(t) = \int_{t-10}^{10} x(\tau) x(t-\tau) d\tau$$

$$= \int_{t-10}^{10} (1) (1) d\tau$$

$$= [\tau]_{t-10}^{10}$$

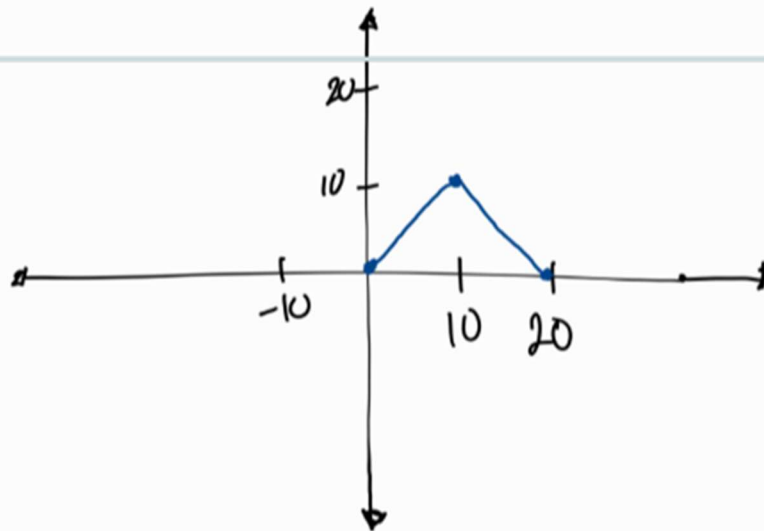
$$= 10 - (t-10)$$

$$= 10 - t + 10$$

$$\boxed{y(t) = -t + 20}$$

Figure 16: Analytical Calculation and Plot of $z(t)$ 3/4.

Graph of $z(t) = x(t) * x(t)$



$$z(t) = \begin{cases} t & 0 < t < 10 \\ -t+20 & 10 < t < 20 \\ 0 & \text{e.w.} \end{cases}$$

Figure 17: Analytical Calculation and Plot of $z(t)$ 4/4.

```

%A.5: Compute/plot Fourier Transform/Spectra of narrower and wider pulses

%% Initialization
clear; clc; close all;

%% Create narrower rect pulse (pulse width = 5)

%create rect pulse
N = 100; PulseWidth = 5;
t = [0:1:(N-1)];
x1 = [ones(1,PulseWidth), zeros(1,N-PulseWidth)];

%Plot narrow pulse w/ same axis as wide pulse for easy comparison
figure('Name', 'Graph of Narrow Rect Pulse');
stairs(t,x1); grid on; axis([-1,30,-0.1,1.1])

%% Obtain Fourier transformed function and plot fourier spectra

Xf1 = fft(x1); %to get the freq domain func

%define the complex func's domain, w
f = [-(N/2):1:(N/2)-1]*(1/N);
w = 2*pi*f; %conver to rad/s

figure('Name', 'Fourier Spectra for Narrow Pulse');
subplot(211); plot(w,fftshift(abs(Xf1))); grid on; %plot amplitude spectra
subplot(212); plot(w,fftshift(angle(Xf1))); grid on; %plot phase spectra

%% Create wider rect pulse (pulse width = 25)

%create rect pulse
N = 100; PulseWidth = 25;
t = [0:1:(N-1)];
x2 = [ones(1,PulseWidth), zeros(1,N-PulseWidth)];

%Plot wide pulse w/ same axis as narrow pulse for easy comparison
figure('Name', 'Graph of Wide Rect Pulse');
stairs(t,x2); grid on; axis([-1,30,-0.1,1.1])
%% Obtain Fourier transformed function and plot fourier spectra

Xf2 = fft(x2); %to get the freq domain func

%define the complex func's domain, w
f = [-(N/2):1:(N/2)-1]*(1/N);
w = 2*pi*f; %conver to rad/s

figure('Name', 'Fourier Spectra for Wide Pulse');
subplot(211); plot(w,fftshift(abs(Xf2))); grid on; %plot amplitude spectra
subplot(212); plot(w,fftshift(angle(Xf2))); grid on; %plot phase spectra

```

Figure 18: A5 Code.


```

%A.6: Compute Fourier Transform of narrower pulse

%% Initialization
clear; clc; close all;

%create rect pulse
N = 100; PulseWidth = 10;
t = [0:1:(N-1)];
x = [ones(1,PulseWidth), zeros(1,N-PulseWidth)];

% Perform fourier transform on x
Xf = fft(x); %to get the freq domain func

%define the complex function Xf's domain.
f = [-(N/2):1:(N/2)-1]*(1/N);
w = 2*pi*f; %convert to rad/s

%% Compute  $w_+(t) = x(t)e^{j(\pi/3)t}$  and it's Fourier Transform
w_plus = x .* exp( ( 1i*(pi/3)).*t );
W_plus = fft(w_plus);

%% Compute  $w_-(t) = x(t)e^{-j(\pi/3)t}$  and its fourier Transform
w_minus = x .* exp( ( -1*1i*(pi/3)).*t );
W_minus = fft(w_minus);

%% Compute  $w_c(t) = x(t)\cos(\pi/3)t$  and its fourier Transform
w_c = x .* cos(pi/3) .* t;
W_c = fft(w_c);

%% Plot and compare the magnitude and phase Spectra

figure('Name', 'Magnitude and Phase Spectra for  $w_+(t)$ ');
%plot amplitude (magnitude) spectra
subplot(211); plot(w,fftshift( abs(W_plus))); grid on;
xlabel('\omega'); ylabel('|W_{+}(\omega)|');
title('Magnitude Spectra of  $w_+(t)$ ');
%plot phase spectra
subplot(212); plot(w,fftshift( angle(W_plus))); grid on;
xlabel('\omega'); ylabel('\angle W_{+}(\omega)');
title('Phase Spectra of  $w_+(t)$ ');

figure('Name', 'Magnitude and Phase Spectra for  $w_-(t)$ ');
%plot amplitude (magnitude) spectra
subplot(211); plot(w,fftshift( abs(W_minus))); grid on;
xlabel('\omega'); ylabel('|W_{-}(\omega)|');
title('Magnitude Spectra of  $w_-(t)$ ');
%plot phase spectra
subplot(212); plot(w,fftshift( angle(W_minus))); grid on;
xlabel('\omega'); ylabel('\angle W_{-}(\omega)');
title('Phase Spectra of  $w_-(t)$ ');

```

Figure 19: A6 Code 1/2.

```

figure('Name', 'Magnitude and Phase Spectra for w_c(t)');
%plot amplitude (magnitude) spectra
subplot(211); plot(w,fftshift(abs(W_c))); grid on;
xlabel('\omega'); ylabel('|W_{c}(\omega)|');
title('Magnitude Spectra of w_{c}(t)');
%plot phase spectra
subplot(212); plot(w,fftshift(angle(W_c))); grid on;
xlabel('\omega'); ylabel('\angle W_{c}(\omega)');
title('Phase Spectra of w_{c}(t)');

```

Figure 20: A6 Code 2/2.

```

%B1 | Lathika Sooriyapperuma | 500904706 | Lab 4 Part B
% Transmitting radio broadcast through communications channel

%% Initialization
clear
clc
close all
load('Lab4_Data.mat');

%% Define constants
N = 80000; %80k samples at 32 kHz sample rate = 2.5 second audio clip

%% Convolve original signal with low pass filter
y1 = conv(xspeech,hLPF2000);
y1 = y1(1:80000); %remove irrelevant parts (1)

%% Create Carrier Signal w/fundamental frequency=4khz
c1 = osc(4000,N);

%% Modulate signal
y2 = y1 .* c1;

%% Transmit through transmission band
y3 = conv(y2, hChannel);
y3 = y3(1:80000);

%% De-modulation
y4 = y3 .* c1;
y4 = y4(1:80000);

%% Remove high frequencies by using Low-Pass filter
y = conv(y4,hLPF2500);
y = y(1:80000);

%% Debug recovered Sound
sound(y,32000);

```

Figure 21: B1 MATLAB Code.

References

[1] "ELE532 Signals and Systems I, Laboratory Assignment 4, The Fourier Transform: Properties and Applications." Department of Electrical, Computer and Biomedical Engineering, Faculty of Engineering and Architectural Science, Toronto Metropolitan University, Toronto