ECE 459 Project Report

Fall 2023

Project#3: Computer Experiment of The Distortion-Free Communication Theorem and Amplitude Modulation

Jie Wang, Junhao Zhu, Junjie Ren, Ruiqi Zhao, Suhao Wang

Prof. Said Mikki
ZJU-UIUC Institute
Zhejiang University, Haining, Zhejiang, China
Oct.28, 2023

1 Introduction

1.1 Experiment Setup

The second part of this experiment is about Ideal Band-Pass Filter. The experiment requires us to generate a pulse $x(t)$ with arbitrary amplitude first, and then obtain its time domain image and frequency domain image. Unlike the LPF section, in the BPF section, to show the effect of BPF more clearly, the experiment requires the use of DSB AM modulation to shift the spectrum. Finally, the new $x(t)$ is passed through BPF to obtain the frequency domain image $Y(f)$ and the corresponding time domain image $y(t)$.

1.2 Characteristic

LPF: Low-Pass Filter $|H(f)| = \begin{cases} 1 & , f_c - \dfrac{B_c}{2} \leq |f| \leq f_c + \dfrac{B_c}{2} \\ 0 & , otherwise \end{cases}$

BPF: Band-Pass Filter $|H(f)| = \begin{cases} 1 & , 0 \leq |f| \leq B_c \\ 0 & , otherwise \end{cases}$

1.3 Objective

The primary objectives of the project are:

1. **Generation and visualization** of the original pulse and filtered pulse (LPF and BPF) in time domain and frequency domain. .
2. **Discussion** of the details and problems during the experiment.

2 Methodology

Square Pulse Generation: Using Python, we can employ the *numpy* library to generate and plot the pulse for a given amplitude, duration and the center $T1$.

Fourier Transform Calculation: Using the *numpy* library in Python, the Fourier Transform $X(f)$ of the generated triangular pulse is computed by fast Fourier Transform.

Band-Pass Filter and Low-Pass Filter: Using the *numpy* library in Python, set a mask function in the Fourier Transform generator function.

Bandwidth B: using the *math* library and *numpy* library in Python as well as some Mathematical derivation, calculate the 3dB bandwidth of the original signal.

2.1 Methodology 1: convolution in time domain

2.1.1 LPF result

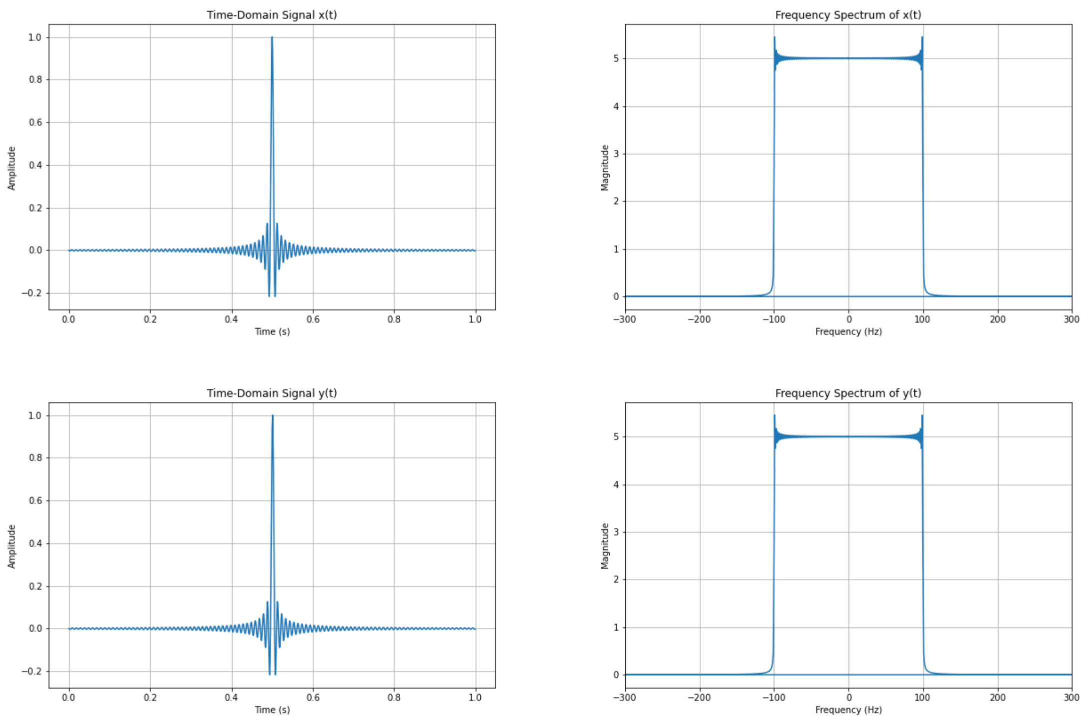

Figure 1: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B < Bc$ (LPF)

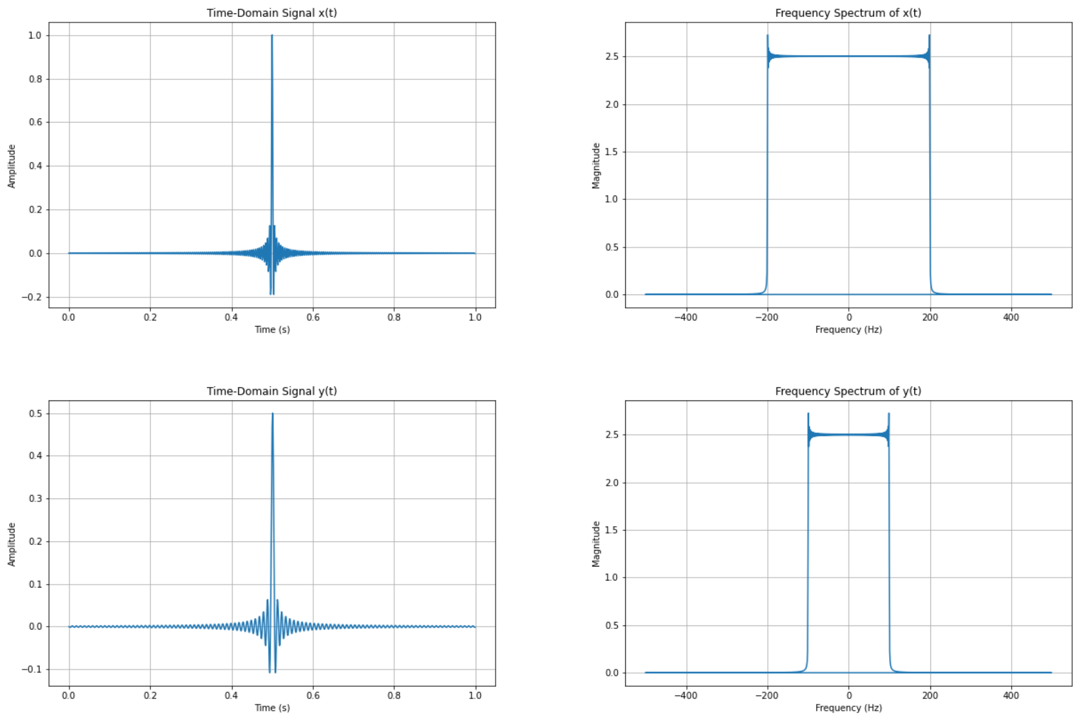

Figure 2: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B > Bc$ (LPF)

2.1.2 LPF answer

(a) $B < B_c$ figures' parameters

- **pulse type:** Nyquist pulse
- **signal bandwidth:** $B = 100$ Hz.
- **cut-off bandwidth:** $B_c = 200$ Hz.

(b) $B > B_c$ figures' parameters

- **pulse type:** Nyquist pulse
- **signal bandwidth:** $B = 200$ Hz.
- **cut-off bandwidth:** $B_c = 100$ Hz.

(c) Compare and comment
Case 1:$B < Bc$ **(Narrow Bandwidth)**
Time Domain ($B < Bc$): In the time domain, the signal (x(t)) exhibits a narrow pulse shape. The narrow pulse has a longer duration in time, and the amplitude is relatively high.
Frequency Domain ($B < Bc$): In the frequency domain, the spectrum (X(f)) shows a broad distribution of frequencies. The spectrum spreads out over a wider range of frequencies due to the narrow pulse in the time domain. There is a significant energy content across a range of frequencies.
Case 2:$B > Bc$ **(Wide Bandwidth)**
Time Domain ($B > Bc$): In the time domain, the signal (x(t)) exhibits a broader and shorter pulse shape. The pulse is relatively shorter in duration, and the amplitude is lower compared to the narrow pulse in Case 1.
Frequency Domain ($B > Bc$): In the frequency domain, the spectrum (X(f)) is more localized and concentrated around the central frequency (near zero). The energy is concentrated around a narrower range of frequencies. The bandwidth in the frequency domain is smaller, reflecting the broader and shorter pulse in the time domain.
Commentary:

When $B < Bc$, the signal x(t) in the time domain is characterized by a narrow pulse, leading to a broader frequency spectrum, with energy spread across a wider range of frequencies. In contrast, when $B > Bc$, the signal x(t) has a wider pulse, resulting in a more concentrated frequency spectrum, with most of the energy located around the central frequency.

These observations are in line with the basic principles of signal processing and the relationship between the time and frequency domains. The bandwidth (B) of a signal determines how spread out or concentrated the spectrum is, and it has a significant impact on the shape and properties of the signal in both domains.

2.1.3 BPF result

2.1.4 BPF answer

(a) $B < B_c$ figures' parameters

- **pulse type:** Nyquist pulse
- **carrier frequency:** $fc = 150$ Hz.
- **signal bandwidth:** $B = 50$ Hz.
- **cut-off bandwidth:** $B_c = 200$ Hz.

(b) $B > B_c$ figures' parameters

- **pulse type:** Nyquist pulse

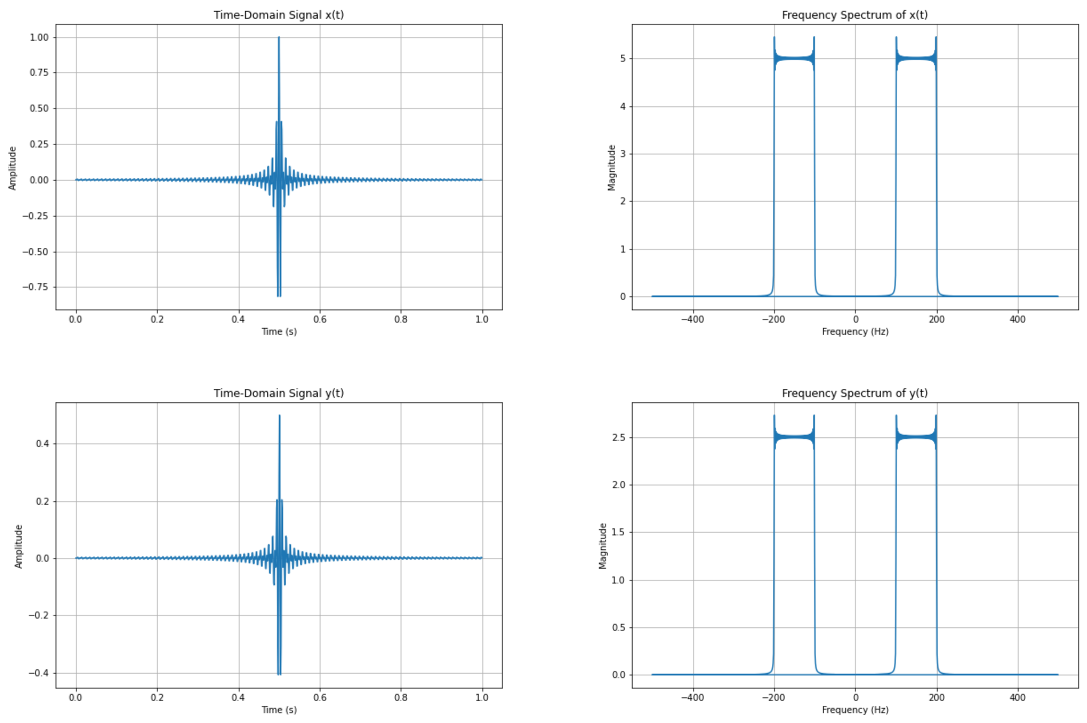

Figure 3: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B < Bc$ (BPF)

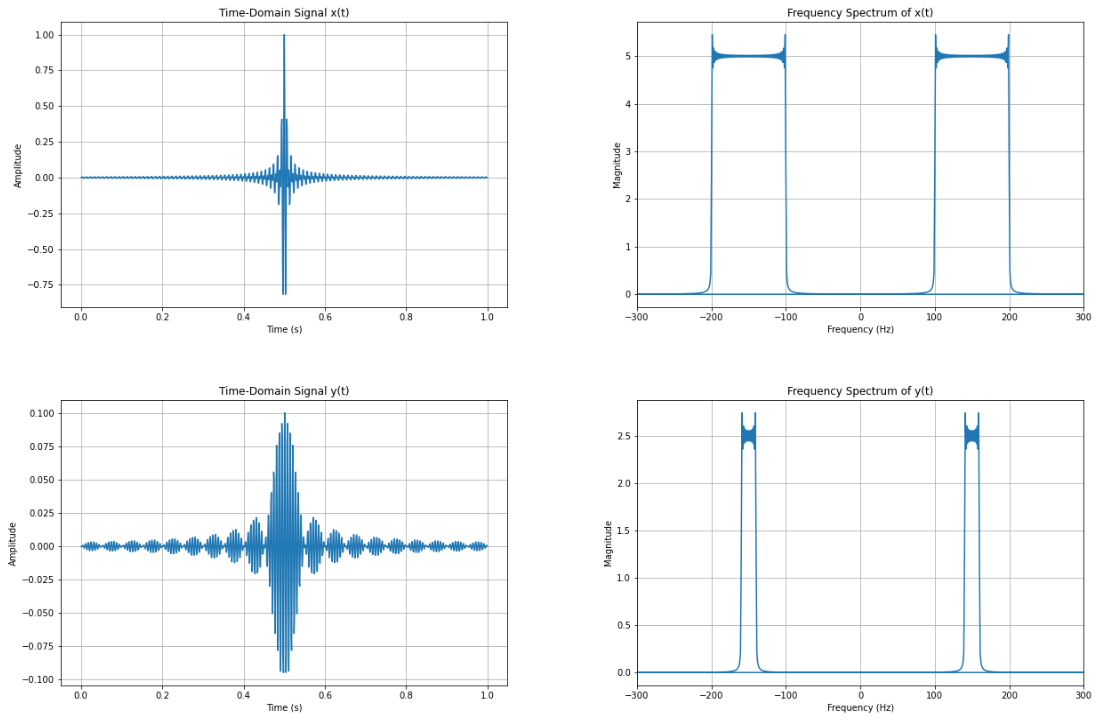

Figure 4: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B > Bc$ (BPF)

- **carrier frequency:** $fc = 150$ Hz.
- **signal bandwidth:** $B = 50$ Hz.
- **cut-off bandwidth:** $B_c = 20$ Hz.

(c) Compare and comment
Comparison:

In the frequency domain: when $Bc < B$, Band-Pass Filter intercepts less frequency domain near f_c. $Y(f)$ is concentrated around a narrower range of frequencies. When $Bc > B$, Band-Pass Filter intercepts more frequency domains near f_c. $Y(f)$ is concentrated around a broader range of frequencies.

In the time domain: when $Bc < B$, the envelope shape of the output signal $y(t)$ is very different from the original signal $x(t)$. The amplitudes of $x(t)$ and $y(t)$ are very different. When $Bc > B$, the envelope shape of the output signal $y(t)$ is closer to the original signal $x(t)$. The amplitudes of $x(t)$ and $y(t)$ are similar.

Commentary:

When the intercepted spectrum covers the main components of the original signal in spectrum, that is, the cut-off bandwidth covers the bandwidth, the new signal can basically restore the original signal in the time domain.

When the cut-off bandwidth cannot cover the 3dB bandwidth, it will take on the shape of a sine wave rather than the original signal. Actually, the original signal can be regarded as a superposition of sinusoidal signals of continuous frequency. In this case, the frequency types of the sinusoidal signal are too simple to show the complex original shape.

2.2 Methodology 2: mask in frequency domain

It is convenient to directly mask some frequencies in the frequency domain to realize LPF and BPF, and the images are placed in the appendix. Also, in this method, the pulse type are changed from Nyquist pulse to rectangular pulse. But the results are similar.

3 Discussion

3.1 Methodological dispute

During the discussion, one method that our team members thought about was to find the impulse response corresponding to BPF, and then convolve it with $x(t)$ to get the output signal. However, Some of us think that the convolution method is not feasible, especially since BPF does not have a corresponding function, but, in fact, we can implement BPF by multiplying the corresponding sinc function of LPF by the cosine equation. That's method 1. At the same time, because of the particularity of the ideal filter, another method could be applied, too. Instead of calculating $h(t)$, it's a good idea to generate the mask function as $H(f)$ using *numpy* directly during the process of generating fft.

3.2 Clarification of Pulse

The pulse used in this experiment is not an ideal pulse, but a random pulse defined by ourselves. Some of us initially thought we should use an ideal pulse for $x(t)$. However, it seems that it is not suitable in practice. Python uses discrete sampling points to generate functions and cannot get an ideal pulse without a duration. At the same time, we have questions about the choice of pulse type. Finally, the rectangular pulse and the Nyquist pulse are chosen. Using multiple pulses and finding solutions helps us understand the project better.

3.3 Bandwidth questions

The bandwidth of $x(t)$ does not take into account the full spectrum of the signal. It actually means 3dB bandwidth. Pulse signals are superimposed by sinusoidal signals of infinite frequencies, and its spectrum extends from negative infinity to infinity. Since most of the frequencies correspond to very small magnitudes, In practice people only focus on The bandwidth of the spectrum range that is 3dB less than the peak power, that is, 50% of the peak.

4 Conclusion

4.1 Summary

In this experiment, our group used the *numpy*, *math* library in python to illustrate the raw signal as well as the signal through BPF and LPF in both time and frequency domain. we conclude that using a cut-off bandwidth greater than bandwidth B can better restore the original signal, and using a cut-off bandwidth less than bandwidth B will result in a large change in the shape of the signal.

4.2 Application

The conclusion of this project can be applied to various fields:

- **AM Radio Broadcasting:** In traditional AM radio, the carrier wave is modulated by the audio signal to produce a DSB-SC signal. The carrier is suppressed, and only the upper and lower sidebands containing the audio information are transmitted. DSB-SC allows for the efficient use of bandwidth in radio broadcasting.
- **Single-Sideband (SSB) Modulation:** DSB-SC modulation can be used to generate a single sideband (either upper or lower) by selectively filtering one sideband while discarding the other. SSB modulation is used in applications where bandwidth efficiency is crucial, such as long-distance voice communication in radio and telephony.
- **Amplitude Shift Keying:** In digital communication, DSB-SC can be used in amplitude shift keying (ASK) modulation. In ASK, the presence or absence of a carrier wave represents binary data. The carrier is suppressed when there is no signal, and it is present when transmitting data.
- **Carrier Recovery:** DSB-SC modulation can be used in carrier recovery circuits, where the original carrier is reconstituted from the sidebands. This is particularly useful in demodulation processes.

5 Appendix

5.1 References

[1] Wikipedia. LPF. `https://en.wikipedia.org/wiki/Low-pass_filter`.

[2] Wikipedia. BPF. `https://en.wikipedia.org/wiki/Band-pass_filter`.

5.2 Code of LPF and BPF

We will submit attached file for this part.

5.3 Used Functions Documentation

- **numpy.linspace():** Generates evenly spaced numbers over a specified interval.
- **numpy.where():** Returns elements based on conditional expression.
- **numpy.fft.fft():** Computes the one-dimensional n-point discrete Fourier Transform.
- **numpy.fft.fftfreq():** Returns the discrete Fourier Transform sample frequencies.
- **numpy.fft.ifft():** compute the inverse discrete Fourier Transform.
- **numpy.logical_and():** used as the filter in frequency domain.

5.4 Task Separation

Task 1: LPF: By Junjie Ren

1. All the task that is related to LPF
2. application in conclusion
3. Check and verify the report

Task 2: BPF: By Suhao Wang

1. All the task that is related to BPF
2. objective in introduction

5.5 Additional images

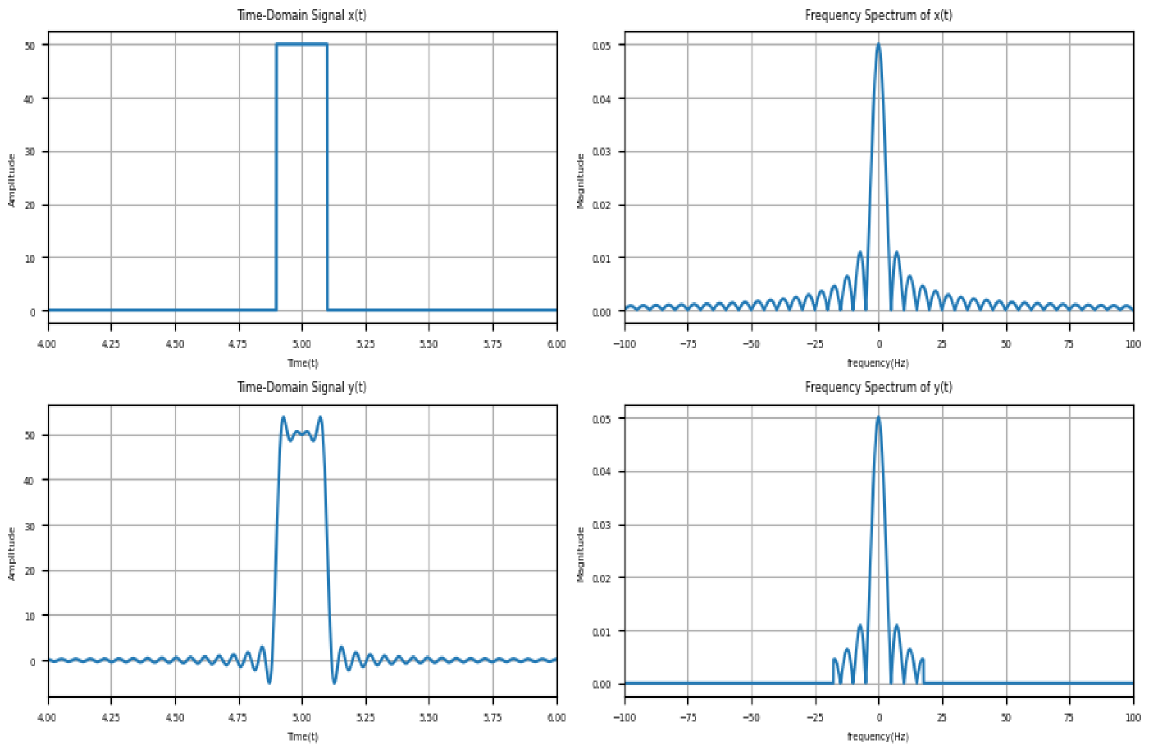

Figure 5: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B = 4.29Hz, Bc = 34.32Hz$ (LPF)

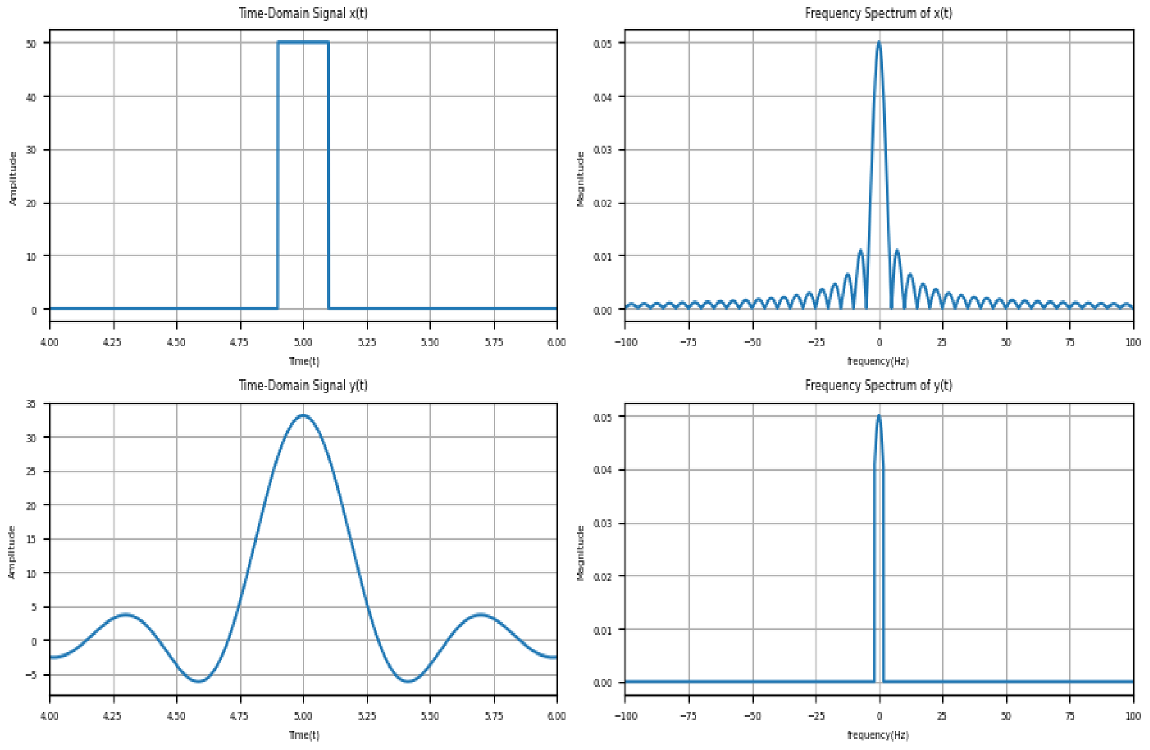

Figure 6: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B = 4.29Hz, Bc = 3.43Hz$ (LPF)

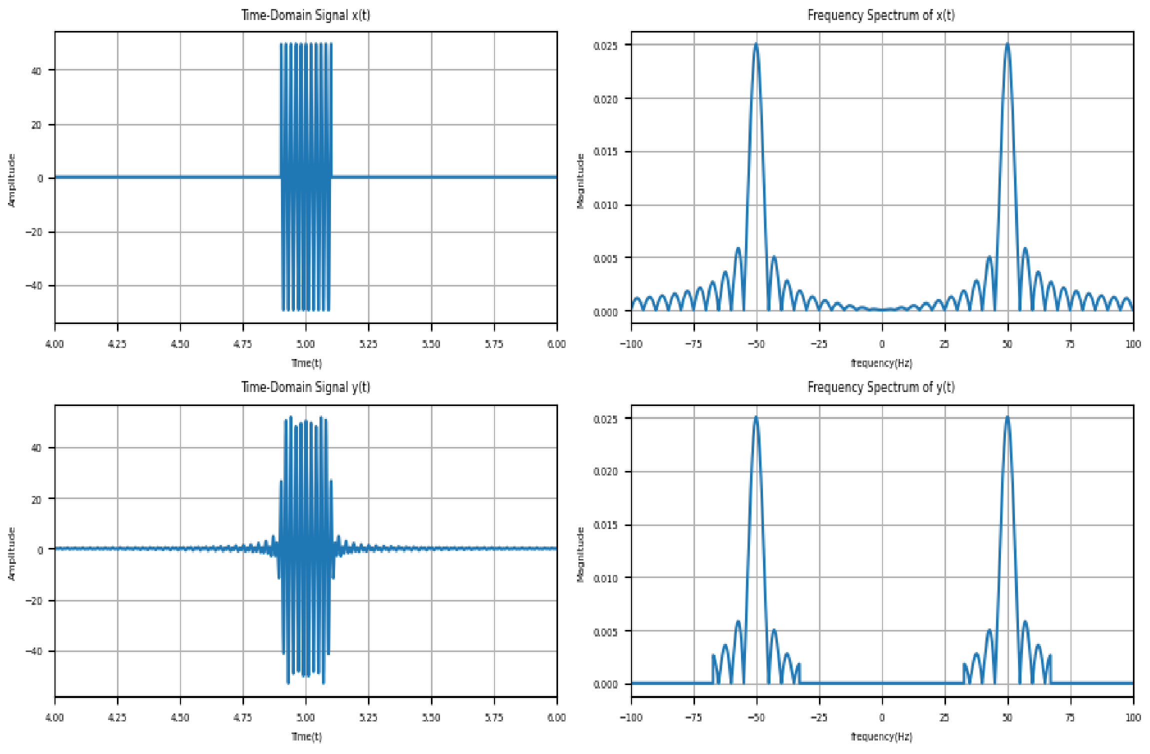

Figure 7: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B = 4.29Hz, Bc = 34.32Hz$ (BPF)

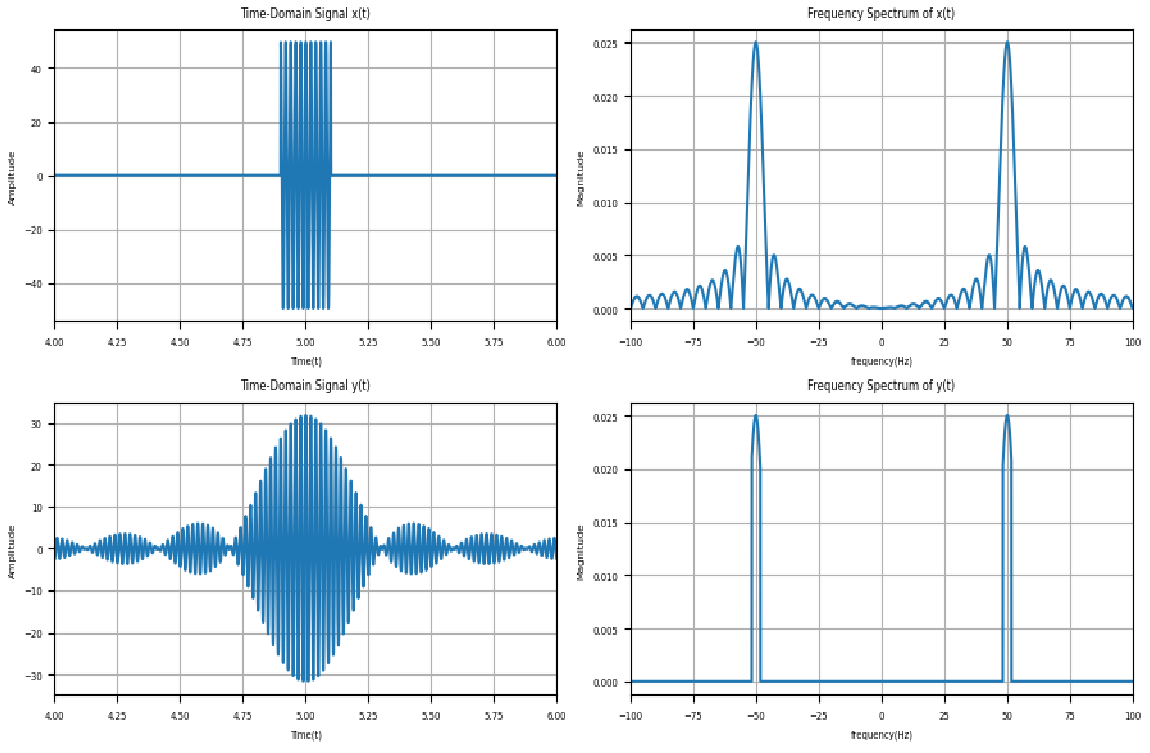

Figure 8: $x(t)$, $X(f)$, $y(t)$, $Y(f)$ when $B = 4.29Hz, Bc = 3.43Hz$ (BPF)