English Alphabet Character Voice-frequency Encoder/Decoder Systems Electrical & Computer Engineering Department – Birzeit University

Abd Khuffash[1], *Hatem Hussein*[2]

1200970@student.birzeit.edu[1], 1200894@student.birzeit.edu[2]

Abstract

This paper presents a two-phase project aimed at designing and implementing an English alphabet character voice-frequency encoder and decoder system. In Phase One, an encoder is developed to represent each English character using a combination of three voice-band frequency components (low, middle, and high). Additionally, a Graphical User Interface (GUI) is built to allow users to encode English strings, offering the choice to either play the generated signal or save it as a (.wav) audio file. In Phase Two, a decoder is introduced to recover the original text string from the encoded multi-frequency signal. Two decoding approaches are implemented: one utilizing frequency analysis (e.g., Fourier transform) and the other employing bandpass filters to extract frequencies. A GUI is developed to facilitate user-friendly decoding of audio files, displaying the recovered text string. The paper concludes with comprehensive testing of both decoding systems, evaluating accuracy by recognizing letters within the encoded strings. The presented system offers a unique perspective on voice-frequency encoding and decoding, contributing to the advancement of audio signal processing technologies.

Index Terms: encoder, decoder, frequency

1. Introduction

In the realm of audio signal processing, our project focuses on the design and implementation of a novel voice-frequency encoder for English alphabet characters. This encoding system aims to represent each character through a distinctive combination of low, middle, and high voice-band frequencies, opening new possibilities for secure communication and data representation.

Within the domain of audio signal processing, our project embarks on a transformative journey, focusing on the creation of an innovative voice-frequency encoder tailored for English alphabet characters. This cutting-edge encoding system seeks to uniquely represent each character through a carefully crafted combination of low, middle, and high voice-band frequencies. This novel approach not only enhances the richness of audio data but also introduces new dimensions for secure communication and data representation.

Table “1” presents the encoding frequencies meticulously designed for each English character, outlining the low, middle, and high frequency components. For instance, character 'a' is encoded with frequencies 100Hz (low), 1100Hz (middle), and 2500Hz (high). The systematic mapping of characters to distinct voice-band frequencies lays the foundation for a robust and expressive communication medium.

Beyond encoding, the project places a strong emphasis on user interaction by developing an intuitive graphical interface. This interface serves as a user-friendly gateway for encoding English strings, offering an immersive experience in transforming text into auditory signals.

Table 1. *Encoding frequency for each voice character*

Character	Low frequency	Middle Frequency	High Frequency
a	100	1100	2500
b	100	1100	3000
c	100	1100	3500
d	100	1300	2500
e	100	1300	3000
f	100	1300	3500
g	100	1500	2500
h	100	1500	3000
i	100	1500	3500
j	300	1100	2500
k	300	1100	3000
l	300	1100	3500
m	300	1300	2500
n	300	1300	3000
o	300	1300	3500
p	300	1500	2500
q	300	1500	3000
r	300	1500	3500
s	500	1100	2500
t	500	1100	3000
u	500	1100	3500
v	500	1300	2500
w	500	1300	3000
x	500	1300	3500
y	500	1500	2500
z	500	1500	3000
space	500	1500	3500

To ensure the versatility of our system, we introduce two distinct decoding approaches. The first leverages frequency analysis, employing techniques such as Fourier transform, to discern the underlying frequencies and recover the original text. [4] The second employs bandpass filters, selectively isolating frequencies, to achieve the same objective. These decoding mechanisms, coupled with the encoding system, constitute a comprehensive venture into the unexplored potential of voice-frequency encoding and decoding. [1]

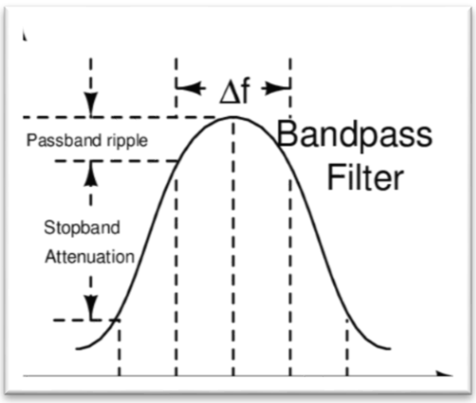

Figure 1: *Response of a bandpass filter.* [3]

In essence, this project pioneers a novel approach to audio data communication and representation, unlocking possibilities that extend beyond conventional methods. The strategic fusion of encoding intricacies, user-centric design, and advanced decoding techniques positions our venture at the forefront of audio signal processing innovations.

2. Problem specification

Within the domain of audio signal processing, our project addresses the challenge of efficient encoding and decoding of voice signals to represent English alphabet characters. The general problem involves developing a robust system capable of translating characters into distinctive voice-band frequencies for secure communication. To tackle this, our well-defined technical goal is to design, implement, and test a voice-frequency encoder that accurately translates English characters into a combination of low, middle, and high frequencies. Subsequently, we aim to create a corresponding decoder system that can reliably recover the original text from the encoded voice signals. This technical goal ensures precision in audio data representation and opens avenues for secure communication applications.

3. Data

The development and evaluation of the voice-frequency encoding and decoding system revolve around user-centric interactions. The system seamlessly accommodates two primary types of user input:

3.1. Encoding Input

Users can input English strings directly into the system for encoding. This input method allows for real-time testing of the system's ability to transform textual information into voice-frequency-encoded signals. Taking into account different test cases such as:

3.1.1. Varying String Length

Strings of different lengths, from short phrases to longer sentences and paragraphs, to evaluate the system's scalability and efficiency.

3.1.2. Customized Input Strings

Synthetic signals with customized input strings to assess the system's handling of specific linguistic patterns, special characters, and non-standard vocabulary.

3.2. Decoding Input

For decoding, users specify the name of the audio file containing the encoded voice signal. This straightforward input method enables users to recover the original text from previously encoded audio files.

3.2.1. Varying Signal Length

Signals of different lengths, from short signals representing one character to longer signals representing sentences, to evaluate the system's scalability and efficiency.

3.2.2. Noised Signals

Signals noised by other characters that are not English, or special characters that are not considered in our systems to see what the output of the decoder system as there is some noise in the signal.

This diverse set of signals ensures a comprehensive evaluation of the system, considering various linguistic, environmental, and contextual factors. The chosen signals aim to validate the system's effectiveness and applicability across a wide range of scenarios.

4. Evaluation Criteria

We will thoroughly evaluate the performance of the voice-frequency encoding and decoding system through a systematic analysis based on specific criteria. Our chosen criteria are carefully selected to be objective, quantitative, and capable of discerning improvements in both functionality and efficiency. The evaluation will concentrate on the following essential metrics:

4.1. Accuracy of Decoding

We will assess how precisely the system decodes the original text from previously encoded audio files. The evaluation involves quantifying the percentage of characters that are correctly recovered by the system compared to the total characters present in the original string. This measure serves as a direct indicator of the system's proficiency in faithfully reproducing the intended content during the decoding process.

4.2. Accuracy of Filters

Assess the fidelity of the decoded audio signals compared to the original input. Especially when using the bandpass filter, as the power ratio used to evaluate the efficiency of the filter that is used in comparison between the frequencies bounds used in the decoder system.

4.3. Encoding/Decoding Processing Speed

The time taken for encoding and decoding tasks across signals of varying lengths will be quantified. The processing speed will be measured to ensure efficient real-time performance.

4.4. Errors Handling

In assessing error handling, the system's robustness to distorted input signals and unexpected variations in encoded data will be scrutinized. This evaluation extends to handling non-English characters, examining the system's proficiency in encoding diverse linguistic nuances. Additionally, the system's capability to process special characters beyond the standard space will be examined to ensure versatility. Furthermore, the assessment includes scrutinizing the system's competence in managing filter cut-off frequencies, with particular attention to the relationship between the second (fc2) and first (fc1) cut-off frequencies. This holistic evaluation aims to reinforce the system's adaptability, accommodating a broad spectrum of linguistic inputs, symbols, and frequency parameters, thus enhancing its reliability in diverse real-world scenarios.

4.5. GUI Application and User Experience

The evaluation of the GUI application and user experience will focus on the ease of system use, the intuitiveness of the graphical interface, and overall user experience. The emphasis will be on evaluating the clarity and simplicity of the interface and assessing the informativeness of displayed outputs within the application. This approach ensures a comprehensive examination of the system's user interaction aspects for refinement and optimization.

4.6. Complexity Checking

The system's scalability, under the umbrella of Complexity Checking, will be assessed by encoding and decoding signals (strings) of varied complexities and lengths. This evaluation includes testing the system with hard strings, featuring intricate linguistic patterns and special characters, as well as long strings and sentences. The goal is to measure the system's performance in handling challenging inputs, ensuring its efficiency and endurance as the complexity and length of the input data scale. This approach guarantees that the system remains effective across a broad spectrum of real-world scenarios.

Utilizing objective, quantitative, and discriminatory criteria ensures a comprehensive and measurable assessment of the voice-frequency encoding and decoding system. The outcomes will offer valuable insights into the system's strengths, areas for enhancement, and its overall performance in practical, real-world situations.

5. Approach

Multiple approaches has been taken in order to hit the goal of the project. Firstly, the encoder system has been built by using one simple way. On the other hand, the decoder system has been built using multiple approaches in order to decode the input voice signal. The first one is using the decoder based on frequencies system and the other is by using the bandpass filter. The bandpass filter has been implemented using three ways and the results has been compared to get the best structure of the decoder based on filters system.

5.1. Encoder System

In the encoder system we used a simple way based on defining a table in Matlab that contains all the values represents in Table "1". Moreover, a simple GUI application has been built to take the input string from the user and encode it into the appropriate signal. We used the cosine in order to construct the signal based on the low, middle, and high frequencies of each character to decode. The below is the GUI used for the encoder system:

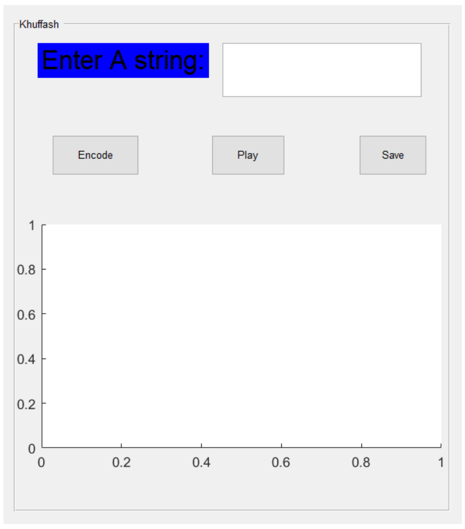

Figure 2: *Encoder GUI*

5.2. Decoder System Based on Frequency

In this system the frequency is used to decode the encoded text in the first phase. The system based on the function called "Frequency" that takes an input signal and calculates the frequencies of the three highest peaks in its frequency spectrum using the Fast Fourier Transform (FFT). The function operates under a sampling frequency of 8000 Hz and considers a signal length of 320 samples. By performing FFT on the input signal, it identifies the three highest peaks and retrieves their corresponding frequencies. The resulting frequencies are stored in an array and sorted in ascending order, providing a sorted list of the top three dominant frequencies in the input signal. The function then returns this array as its output.

Moreover, the main function called "Decoder" processes an input audio file, extracting fragments of 320 samples each. For each fragment, it uses the `Frequency` function to identify the three highest peaks in its frequency spectrum. The resulting frequencies are then adjusted to match predefined frequency ranges for low (L), mid (M), and high (H) frequency bands. The adjusted frequencies are compared with a predefined table, associating each frequency combination with a corresponding character. The identified characters are concatenated to form a decoded string, representing the message encoded in the input audio file. The final decoded string is displayed as output, providing the information encoded in the audio file. The function also includes error handling to continue processing if the identified frequencies fall outside the predefined ranges.

A simple GUI application has been built that enables the user to enter the file name and displays the decoded string on the screen.

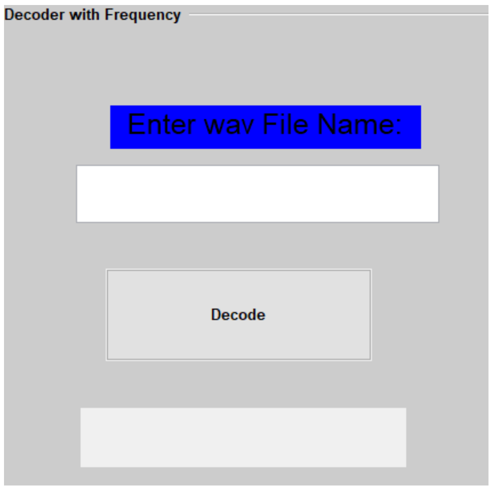

Figure 3: Decoder GUI

5.3. Decoder System Based on Filters

In this system multiple approaches has been used in order to compare between the results got from them and to determine the best way to design the system with the least percentage of error. The first method is based on thresholds values based on some experimentally tests has been taken. The other is by using the power ratios of the output filter with the input signal.

5.3.1. Thresholds Method

In this method, the experimental testing played a crucial role in order to determine the thresholds values in order to determine what values filters should pass and what values should not. The system contains the function `DecoderWithBPF` as a main function that reads an audio signal from a file and divides it into fragments of 320 samples each. It then applies three bandpass filters to each fragment, targeting low, mid, and high frequencies, and calculates the ratio of the filtered signals. Based on these ratios, the function decodes the signal by associating it with predefined frequency ranges for characters. The decoded characters are appended to a string, forming the final decoded output. The function iterates through all fragments, decoding each one, and prints the resulting string. The bandpass filter frequencies and decoding thresholds are customized for the specific task of signal decoding.

In addition, the system consists of the function called `BPF` that calculates a ratio based on the amplitude range of a signal after applying a bandpass filter. It takes an input signal `x`, lower and upper cutoff frequencies `fc1` and `fc2`, and the sampling frequency `fs`. The function first validates the input parameters, ensuring the correct number and numeric nature of arguments. It then designs a bandpass filter using the specified cutoff frequencies and applies it to the input signal. The function defines thresholds based on the filtered signal's amplitude and assigns ratios depending on the amplitude range. It returns a ratio of 0.7 for frequencies within a certain range, 1 for higher amplitudes, and 0 for out-of-range frequencies. The function incorporates error handling to address invalid input conditions.

Multiple BPFs has been tested. Firstly, we used the FIR BPF using the 'designfilter' function. Then, the 'Fdatool' has been used in order to design more accurate and customized filters. Three BPF Window Hann filters has been built with different cut-off frequencies that cover the required ranges in an appropriate way. The below is an example of a Hann filter with fc1=1 and fc2=200 (Low frequency BPF).

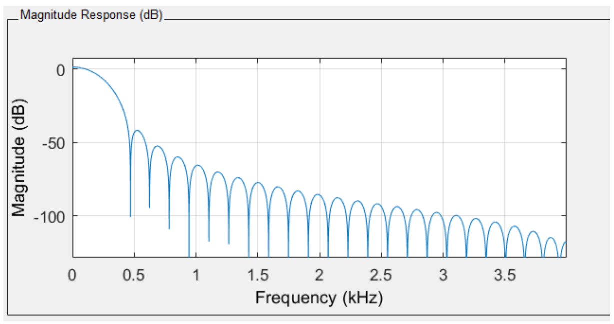

Figure 4: Hann BPF

Moreover, the BPF Window Blackman Type has been used in order to compare it with the Hann type. The below is an example of the BPF Blackman with fc1 = 1090 and fc2=1110.

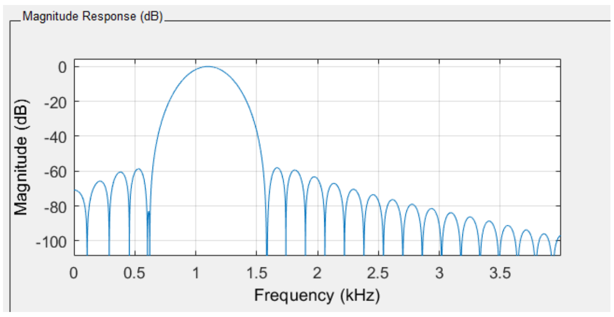

Figure 5: Blackman BPF

5.3.2. Power-Ratio Method

In this method we needed more experimental tests to make the system works very accurate. The system consists of the same two functions that illustrated in the previous section, but with changing the BPF function. The BPF function is based on implementing a bandpass filter to process an input signal `x` within specified frequency cutoffs (`fc1` and `fc2`). Using a bandpass FIR filter design, it filters the input signal and computes the power of the filtered and original signals. The function then calculates the ratio of the power of the filtered signal to the power of the original signal, providing a measure of the signal's frequency content within the specified band. The filter design and power ratio calculation are configured for precise signal analysis, and the function can be used to assess the dominance of certain frequencies within the input signal. That required us more experimental test cases and we came into a conclusion where some signals' power ratios conflict with other signals. For example, at the high frequency bound the power ratio of the letter 'c' is higher than the power ratio of the letter 'b' even though that the high frequency of 'b' (3000) is smaller than the high frequency of 'c' (3500) which is unexpected. Some special cases has been taken into consideration in the main function that is called 'DecoderWithBPF2' that causes conflicts as illustrated above. The same Decoder GUI has been used to test this method.

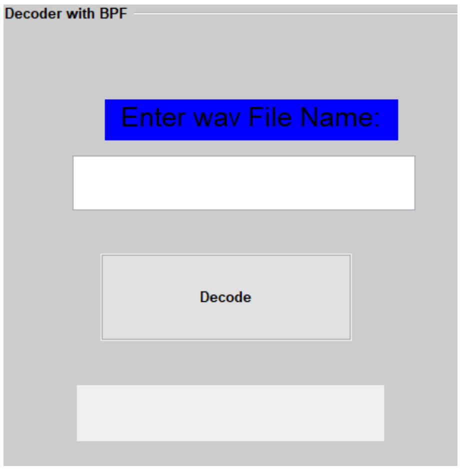

Figure 6: Decoder Based on Filters GUI

6. Results and Analysis

6.1. Encoder System

The encoder system has been tested using multiple input strings taking into account the special character "space", long words, and long sentences. The below test case "birzeit university" represents taking a simple sentence and display its generated signal to the user with enabling the user to play the signal and hear its sound:

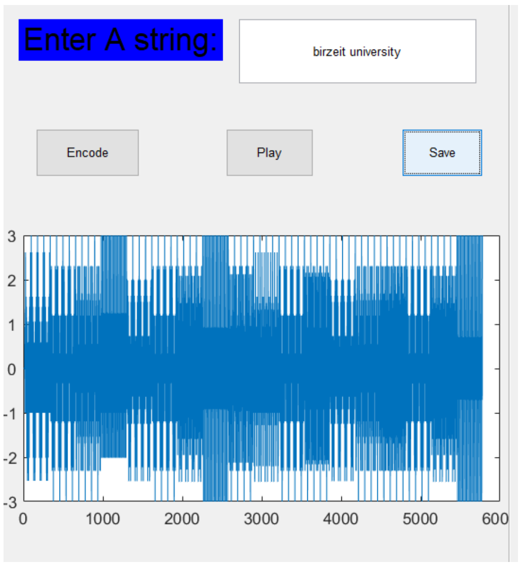

Figure 7: Encoder system test case one

Taking another test case where the input string contains capital letter and some special characters that are not inserted into the table of the system:

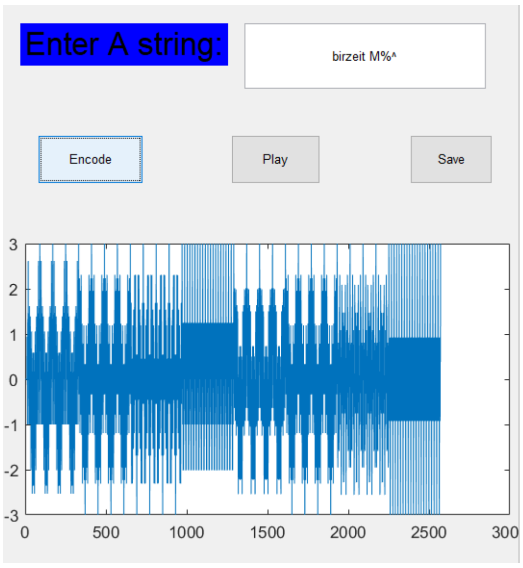

Figure 8: Encoder system test case two

As we can see from the below output that the system encoded the input string even though some of them are not included in the system; and it took them as attenuation to this signal. That part will be clarified using the decoder system to see the output.

6.2. Decoder System

6.2.1. Thresholds Method Testing

In this method we didn't get a good accuracy results neither in the BPF Hann nor the BPF Blackman. The system was good with dealing with small strings. The below test case has been applied to the system after we encode a letter "a":

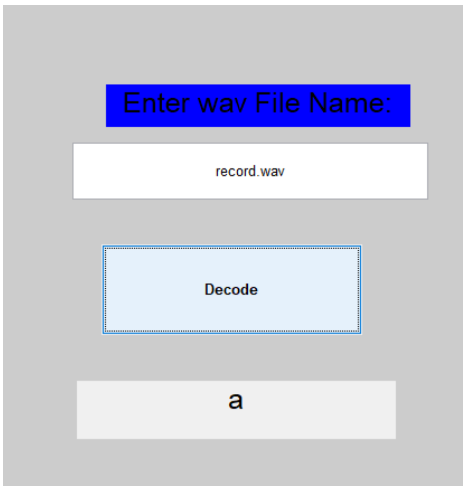

Figure 9: Decoder System BPF-thresholds test case

The issue we faced was dealing with the filters orders, cut-off frequencies, and the thresholds values selection. Even though we went through all the steps step by step and multiple types of BPFs has been used we got an accuracy result of 50% for this method.

6.2.2. *Power-Ratio Method Testing*

This method got a better results than the previous method. We entered a longer strings and strings containing spaces. A better accuracy result has been got due to the special conditions that has been added in order to ignore all the conflicts. The below is a simple test case containing one word:

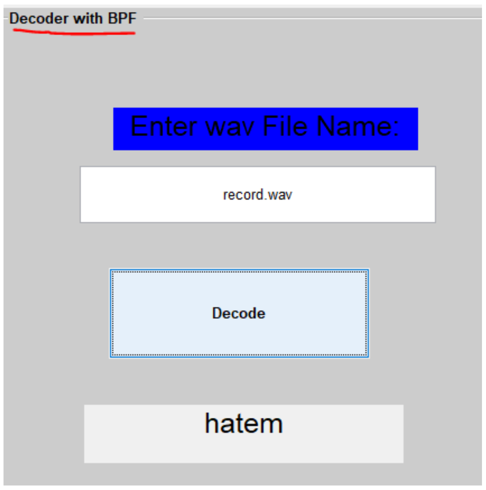

Figure 10: Decoder Based on BPF and Power-Ratio test case one

Taking another test case contains only “space“, the below results has been got:

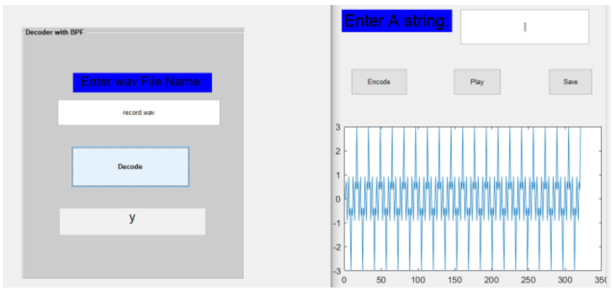

Figure 11: Decoder Based on BPF and Power-Ratio test case two

As can be seen from the previous results that the output got was “y” which incorrect due to some conflict we noticed In those two characters where is the power ratio of the “y” is very close to power ratio of the “space”.

Although the system got some percentage of error but a better accuracy than the threshold approach has been got. The problem with this method is the calculated values we had to do based on each frequency bound.

7. Development

As illustrated before we faced an issue dealing with the filters. So we went through another approach based on the Power-Ratio and we got a better results with a good percentage of accuracy. Although the approach was better but it has many side effects, especially in the part when many calculations has been taken into account in order to use them in the special cases conditions in the main function “DecoderWithBPF2”.

They really helped to get some string accurate, but still some long strings didn’t encoded them correctly due to some conflict in the generated input signal.

Some improvements could be done in order to generalize the system to fit at more languages and more special characters and to get rid of the special cases’ conditions in the Matlab code.

8. Conclusions

In the progression of this project, insights have been gained into the effectiveness of voice-frequency encoding for representing English alphabet characters. Versatility has been demonstrated through the handling of various inputs, encompassing different languages and complex patterns. Efficient decoding of signals of diverse lengths and complexities has been showcased, emphasizing the practical utility of the system in secure communication. Overall, the project has not only addressed the initial problem but has also underscored the system's reliability and potential applications in audio data representation.

9. References

[1] *BPF - allaboutcircuits*. (2024). Retrieved from https://www.allaboutcircuits.com/textbook/alternating-current/chpt-8/band-pass-filters/

[2] BPF - mathworks. (2023). Retrieved from https://www.mathworks.com/help/signal/ref/bandpass.html

[3] BPF - researchgate. (2020). Retrieved from https://www.researchgate.net/figure/A-typical-frequency-response-of-a-bandpass-filter_fig3_27543058

[4] FFT - mathworks. (2023). Retrieved from https://www.mathworks.com/help/matlab/ref/fft.html