Applying Complex Algebra to Enhance Wi-Fi Performance for Smart City Connectivity

Samuel Gerrard Hamonangan Girsang and 13523064[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523064@mahasiswa.itb.ac.id, [2]gerrardgrs@gmail.com

***Abstract*—The concept of Smart Cities has gained global prominence as an innovative solution to urban challenges. By using technologies, data analytics and the concept of Internet of Things (IOT), Smart Cities are built for the purpose of enhancing urban infrastructure, optimizing resources, and improving quality of life. Smart City techniques use smart signal processing and other systems, such as complex numbers, to find improved performance. Overall, these methods allow for effective data transfer, strong communication, and enhanced decision-making systems. The use of complex numbers may aid design implementation for higher reliability systems, noise immunity, and supportive real-time application. The integration of performance metrics and advanced signal processing methods further supports optimal resource utilization, sustainable development, and proliferation of smart and connected urban environments.**

***Keywords*—Smart Cities, complex number, performance, signal**

I. Introduction

During the new era of technological revolution, many countries are racing to develop more advanced technologies. A lot of countries have considered making their ideal form of Smart Cities. This idea was formed because of the challenges that the world is facing. Those problems are traffic congestion, resource depletion, and environmental degradation. Smart Cities are aimed to not only fix these problems, but also provide a better quality of life for individuals.

The connectivity backbone for all devices and systems in Smart Cities is the internet – the most crucial enabler of Smart Cities. Such interconnected systems require a strong and continuous internet connection to work seamlessly, be it IoT-enabled infrastructure, real-time monitoring, autonomous vehicles, or digital governance. One of the most prevalent wireless internet mediums, WiFi is essential for connectivity and data transfer between a range of technologies in a Smart City. WiFi networks need a strong and reliable signal to work, however.

At this point, the concept of Complex Algebra enters the picture. With the aid of Complex Algebra, we can effectively model and analyze the behavior of the WiFi signals to ensure that we have a metric of optimum coverage and reliability. Focusing on these, researchers can enhance signal strength, minimize interference, and maintain seamless connectivity through understanding and manipulation of the properties comprising complex numbers, such as amplitude and phase. Data is vital to Smart Cities, and this particular application of mathematics is absolutely key in closing the divide between theory and practice.

II. Theoretical Basis

A. Complex Numbers

The concept of Complex Numbers emerges from the need of mathematicians to solve equations that do not have real numbers such as $x^2 + 1 = 0$. In the 16th century, mathematician Gerolamo Cardano, began exploring solutions to cubic equations. This exploration led to the discovery of imaginary numbers. The concept of imaginary numbers was initially regarded with skepticism because of the absurdity of the concept. But after that, the imaginary numbers were formalized by known mathematicians like Euler and Gauss. Both demonstrated that imaginary numbers can be utilized to solve real-world problems. An imaginary number is represented by the symbol “i” also known as “iota” representing the square root of negative one or $\sqrt{-1}$.

To leverage the usage of imaginary numbers, mathematicians created the concept of complex numbers that integrate both real numbers and imaginary numbers. Complex number is represented by the form

$$z = a + bi$$

Where a is the real part (Re(z) = a) and b is the imaginary part (Im(z) = b) and i as $\sqrt{-1}$. The set of complex numbers includes all possible combinations of real and imaginary components. This definition allows numbers that cannot be represented on the traditional real number line to exist, expanding mathematics into the complex plane.

We can represent complex numbers using The Argand Diagram. The Argand Diagram is a graphical representation of complex numbers on a two-dimensional plane. It provides an intuitive way to visualize and interpret complex numbers as points or vectors in the complex plane.

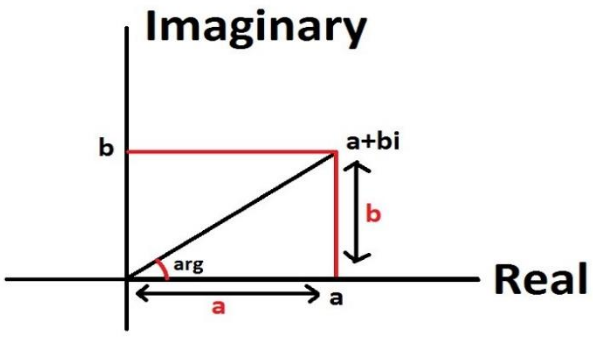


Figure 1.1 Argand Diagram
Source: https://www.advancedhighermaths.co.uk/argand-diagrams/

The length of the vector, also called as modulus, is the distance from the origin (0,0) to the point representing the complex number. The modulus is calculated as

$$|z| = \sqrt{a^2 + b^2}$$

The angle θ formed by the line segment connecting the complex number to the origin with the positive real axis is also called by the argument (arg(z)) which calculated as

$$\theta = \tan^{-1}\left(\frac{b}{a}\right)$$

There are key operations for complex numbers to use for their mathematical usage. Those key operations are

1. Conjugation
The conjugation of a complex number is denoted by z* that is represented by the form

$$z * = a - bi$$

2. Addition
It is the addition of two complex numbers.

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

3. Subtraction
It is the subtraction of two complex numbers.

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

4. Multiplication
It is the multiplication of two complex numbers. The multiplication is based on the distribution property of complex numbers.
.

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$

5. Division
It is the division of two complex numbers. It utilizes the conjugation of the denominator.

$$\frac{a + bi}{c + di} = \frac{(a + bi)(c - di)}{c^2 + d^2}$$

Complex numbers also have axioms that mathematicians could follow to further help the operation of complex numbers. Those axioms are

1. Closure:
For all z_1 and z_2
Addition $z_1 + z_2 \in \mathbb{C}$
Multiplication $z_1 z_2 \in \mathbb{C}$

2. Identity
For each z there is an inverse element -z and 1/z such that
Addition z + (-z) = -z + z = 0
Multiplication $z\left(\frac{1}{z}\right) = \left(\frac{1}{z}\right)z = 1 \;\; (z \neq 0)$

3. Associativity
For all z_1, z_2, and z_3
Addition $z_1 + (z_2 + z_3) = (z_1 + z_2) + z_3$
Multiplication $z_1(z_2z_3) = (z_1z_2)z_3$

4. Distributivity
For all z_1, z_2, and z_3
$z_1(z_2 + z_3) = z_1z_2 + z_1z_3$
$(z_1 + z_2)z_3 = z_1z_3 + z_2z_3$

B. Complex Number as Rotor

Rotor is the rotation axis of an object. Complex numbers have the interpretation of rotors. If a complex number z = a + bi is multiplied by i, with each multiplication, the number undergoes a cyclical transformation, demonstrating the geometrical interpretation of complex numbers as rotors. This repetitive multiplication highlights the rotational property of complex numbers, where every multiplication by i results in a 90-degree counterclockwise rotation of the complex number.

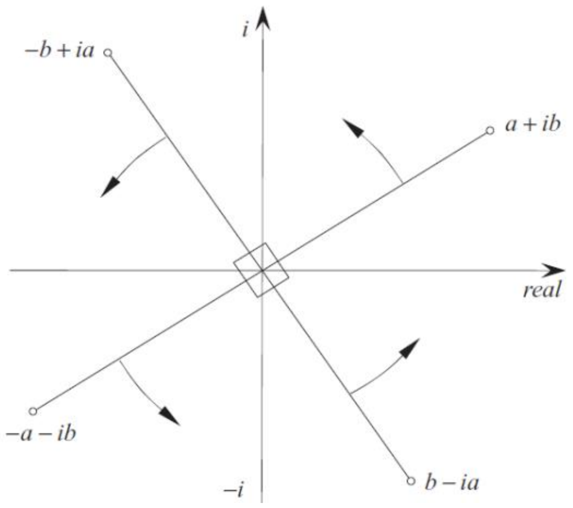


Figure 1.2 Complex Number as Rotor
Source:
https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-24-Aljabar-Kompleks-2023.pdf

The idea of complex numbers as rotor leads us to the Euler equation. The Euler equation provides a mathematical framework to understand how complex numbers operate as rotors. The equation is formed as

$$e^{\pm ix} = \cos(x) \pm \sin(x)$$

This equation is also known as the Euler Formula.

C. Vectors in Complex Plane

Vector plane that has complex numbers as its elements is called a Complex Vector Plane. Complex vector planes of order n are denoted by C^n. Each vector in C^n has n-components, that is v = (v_1, v_2, …, v_n), in the complex form, it translates to

$$v = (\text{v1}, \text{v2}, \dots, \text{vn}) = (a_1 + b_1 i, a_2 + b_2 i, \dots, a_n + b_n i)$$

That also can be interpreted as

$$v = (a_1, a_2, \dots, a_{n)} + i(b_1, b_2, \dots, b_n)$$
$$v = Re(v) + i\, Im(v)$$

D. Quadrature Phase Shift Keying

The Quadrature Phase Shift Keying (QPSK) is a digital modulation technique that transmits two bits of data per symbol. The equation then doubles the data rate. The QPSK maps two bits of data onto one of four distinct phase. These four phases are typically separated by 90 degrees on the complex plane. This idea allows us to represent four possible combinations of two bits using distinct phase shifts. The data inserted will be mapped into phases with this format

- 00 → Phase 0°
- 01 → Phase 90°
- 10 → Phase 180°
- 11 → Phase 270°

The mathematical representation of Quadrature Phase Shift Keying is

$$s(t) = A \cdot e^{j(2\pi f_c t + \phi)}$$

Where:

- **s(t)** is the modulated signal at time **t**,
- **A** is the amplitude of the signal (a constant value), ensuring consistent signal strength,
- **fc** is the carrier frequency, which acts as the foundation for the signal,
- **φ** is the phase shift, which varies according to the data bits being transmitted,
- **j** represents the imaginary unit (j2=−1j^2 = -1j2=−1).

The modulation process of QPSK follows these steps:

1. **Binary Data Mapping**: Each 2-bit sequence of the data is being mapped following the format before.
2. **Phase Shift**: The carrier signal then is shifted in phase according to the binary data to encode the information. So for example, if the binary data is 00, then the phase shift will be 0°, which there is no shift, and if the binary data is 10, the phase shift will be 180°.
3. **Complex Representation**: Each symbol is represented as a complex number in the form of

$$s(t) = A \cdot (\cos(\theta) + j\sin(\theta))$$

Where A is the amplitude and θ is the phase determined by the two-bit inputs.

QPSK role in Wi-Fi connectivity is to enable higher data transmission rate within the same bandwidth. But for modern Wi-Fi such as the current age are using a much-advanced algorithm.

E. Fast Fourier Transform

Fast Fourier Transform (FFT) are the faster versions of Discrete Fourier Transform (DFT) algorithm. This algorithm is commonly used for signal processing in Computer Science.

The Discrete Fourier Transform (DFT) is a tool used to analyze signals in the frequency domain. For example, if we take an audio wave, it is a complex wave made up by many different frequencies. DFT helps us to separate that audio wave into its individual frequency components. This algorithm is usually used for audio processing, image processing, telecommunication, and scientific research.

Fast Fourier Transform (FFT) takes the DFT algorithm and turns it into a much less complex algorithm. The low complexity of the algorithm makes it faster to process data hence the name Fast Fourier Transform. The FFT is highly efficient because of the “divide and conquer” concept it utilized. FFT divides the input signal into smaller and overlapping segments, then it “conquers” the data by computing the DFT of those smaller segments recursively. After the process is done, the algorithm then reconstructs the DFT of the original signal from the DFTs that it gets from the smaller segment. This effectively reduces the time complexity of the DFT algorithm from O(n^2) into O(n log n).

The process of FFT algorithm can be represented as the same mathematical formula as DFT algorithm that is

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{\frac{-j2\pi kn}{N}}$$

Where:

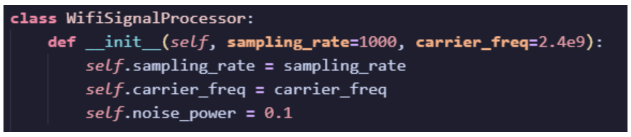
- **X(k)** is the output of the FFT algorithm at k-th frequency bin.
- **N** is the total number of samples in the signal
- **x(n)** is the input signal in the time domain (in range of 0 to N-1)
- $e^{\frac{-j2\pi kn}{N}}$ is the complex exponential function, it represents the sinusoidal basis function for the Fourier Transform

This is a simple representation of the Fast Fourier Transform. For the complex problem and the real-world usage of FFT requires a more complex program and algorithm. Just like DFT, FFT is used for audio processing, image processing, telecommunication, and scientific research. For this paper, Fast Fourier Transform is used to process the signal of wireless internet connections. This algorithm enables us to efficiently analyze spectrums, select channels, and do advanced modulation techniques and beamforming.

III. IMPLEMENTATION

We can make a simple implementation for WiFi performance enhancement by making a program that simulates the process of enhancing internet signals. In this paper, we are making a simple program in python using the provided library such as numpy for mathematical functions, scipy to simulate signals, random to give us a random value that can represent a signal, and matplotlib for data review purposes.

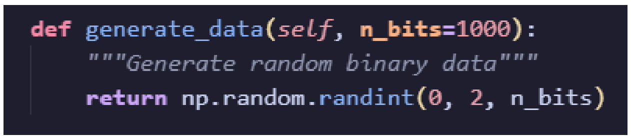
We can divide the program into each function, but first we initialize an object that is WifiSignalProcessor



```python
class WifiSignalProcessor:
    def __init__(self, sampling_rate=1000, carrier_freq=2.4e9):
        self.sampling_rate = sampling_rate
        self.carrier_freq = carrier_freq
        self.noise_power = 0.1
```

Figure 3.1 Object Initialization
Source: Author Documentation

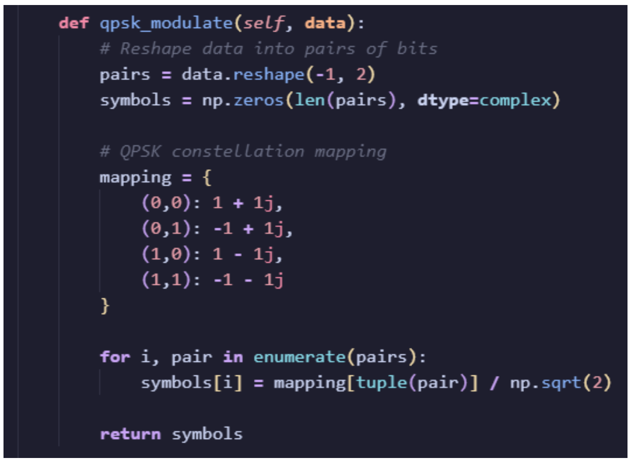
This initializes an object with three attributes, the first attribute is the sampling rate which represents how many samples per second to process and we set 1000 Hz as the default value. The second attribute is the carrier frequency which represents the WiFi carrier frequency with 2.4 GHz as the default value. And the third attribute is noise power which represents the base noise level with 0.1 as the default value.



```python
def generate_data(self, n_bits=1000):
    """Generate random binary data"""
    return np.random.randint(0, 2, n_bits)
```

Figure 3.2 Generate Random Data
Source: Author Documentation

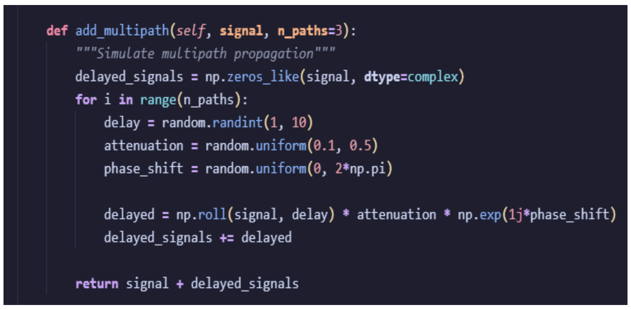
This part of code generates a random bit sequence that acts as a dummy for a WiFi signal that we will use as data.



```python
def qpsk_modulate(self, data):
    # Reshape data into pairs of bits
    pairs = data.reshape(-1, 2)
    symbols = np.zeros(len(pairs), dtype=complex)

    # QPSK constellation mapping
    mapping = {
        (0,0): 1 + 1j,
        (0,1): -1 + 1j,
        (1,0): 1 - 1j,
        (1,1): -1 - 1j
    }

    for i, pair in enumerate(pairs):
        symbols[i] = mapping[tuple(pair)] / np.sqrt(2)

    return symbols
```

Figure 3.3 QPSK Code
Source: Author Documentation

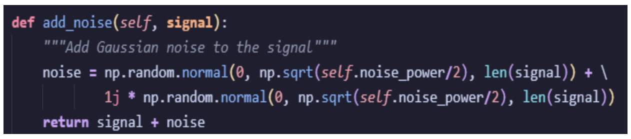
The code then performs QPSK modulation with data input. This function will take the binary data input then map it into complex numbers. The function then normalize the symbols by dividing it by $\sqrt{2}$.



```python
def add_multipath(self, signal, n_paths=3):
    """Simulate multipath propagation"""
    delayed_signals = np.zeros_like(signal, dtype=complex)
    for i in range(n_paths):
        delay = random.randint(1, 10)
        attenuation = random.uniform(0.1, 0.5)
        phase_shift = random.uniform(0, 2*np.pi)

        delayed = np.roll(signal, delay) * attenuation * np.exp(1j*phase_shift)
        delayed_signals += delayed

    return signal + delayed_signals
```

Figure 3.4 Add Multipath
Source: Author Documentation

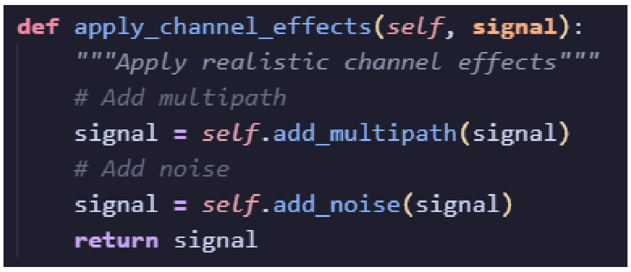
This part of code simulates the multipath interference by creating random delay, random signal weakening, and random phase shift. In general, this code simulates real-world signal interference.



```python
def add_noise(self, signal):
    """Add Gaussian noise to the signal"""
    noise = np.random.normal(0, np.sqrt(self.noise_power/2), len(signal)) + \
            1j * np.random.normal(0, np.sqrt(self.noise_power/2), len(signal))
    return signal + noise
```

Figure 3.5 Add Noise
Source: Author Documentation

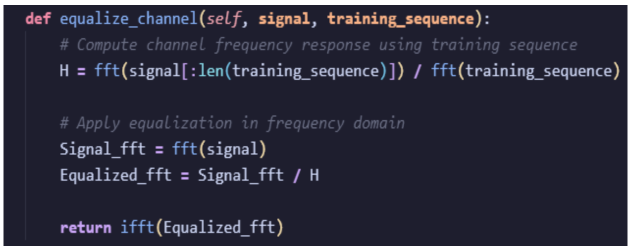
This part of code adds a random noise into the signal data to further simulate real-world signal interference.



```python
def apply_channel_effects(self, signal):
    """Apply realistic channel effects"""
    # Add multipath
    signal = self.add_multipath(signal)
    # Add noise
    signal = self.add_noise(signal)
    return signal
```

Figure 3.6 Apply Channel Effect

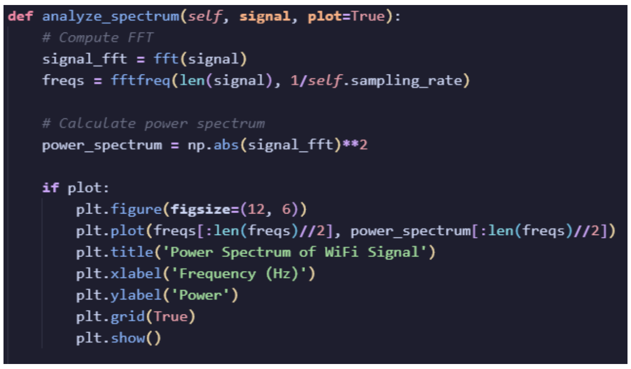
This code applies the previous two functions and combines the multipath and signal noise for real signal condition.



```python
def equalize_channel(self, signal, training_sequence):
    # Compute channel frequency response using training sequence
    H = fft(signal[:len(training_sequence)]) / fft(training_sequence)

    # Apply equalization in frequency domain
    Signal_fft = fft(signal)
    Equalized_fft = Signal_fft / H

    return ifft(Equalized_fft)
```

Figure 3.7 Equalizing Channel
Source: Author Documentation

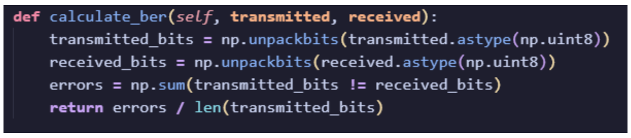
This part of code is where the Fast Fourier Transform algorithm makes its play. The FFT is used to compute the frequency response of channel H that was estimated by the training sequence. This code used a defined Fast Fourier Transform function from python library to calculate the result of the FFT equation.



```python
def analyze_spectrum(self, signal, plot=True):
    # Compute FFT
    signal_fft = fft(signal)
    freqs = fftfreq(len(signal), 1/self.sampling_rate)

    # Calculate power spectrum
    power_spectrum = np.abs(signal_fft)**2

    if plot:
        plt.figure(figsize=(12, 6))
        plt.plot(freqs[:len(freqs)//2], power_spectrum[:len(freqs)//2])
        plt.title('Power Spectrum of WiFi Signal')
        plt.xlabel('Frequency (Hz)')
        plt.ylabel('Power')
        plt.grid(True)
        plt.show()
```

Figure 3.8 Analyze Spectrum
Source: Author Documentation

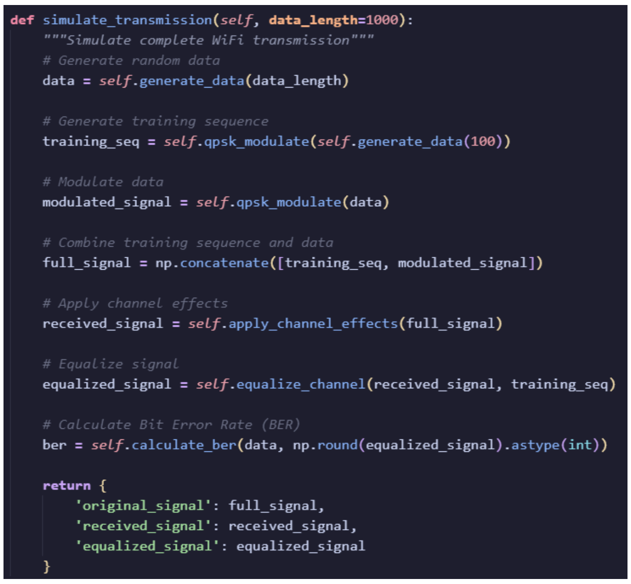
This code performs spectral analysis of the signal also using the Fast Fourier Transform (FFT) algorithm. This code will identify the dominant frequencies and spectral characteristics like the bandwidth or the noise. Then at the end, it visualizes the power spectrum for interference detection.



```python
def calculate_ber(self, transmitted, received):
    transmitted_bits = np.unpackbits(transmitted.astype(np.uint8))
    received_bits = np.unpackbits(received.astype(np.uint8))
    errors = np.sum(transmitted_bits != received_bits)
    return errors / len(transmitted_bits)
```

Figure 3.9 Calculate Bit Error Rate
Source: Author Documentation

This code calculates the Bit Error Rate of the signal to determine the performance of the signal. This code purpose is for later comparison.



```python
def simulate_transmission(self, data_length=1000):
    """Simulate complete WiFi transmission"""
    # Generate random data
    data = self.generate_data(data_length)

    # Generate training sequence
    training_seq = self.qpsk_modulate(self.generate_data(100))

    # Modulate data
    modulated_signal = self.qpsk_modulate(data)

    # Combine training sequence and data
    full_signal = np.concatenate([training_seq, modulated_signal])

    # Apply channel effects
    received_signal = self.apply_channel_effects(full_signal)

    # Equalize signal
    equalized_signal = self.equalize_channel(received_signal, training_seq)

    # Calculate Bit Error Rate (BER)
    ber = self.calculate_ber(data, np.round(equalized_signal).astype(int))

    return {
        'original_signal': full_signal,
        'received_signal': received_signal,
        'equalized_signal': equalized_signal
    }
```

Figure 3.10 Simulate Transmission
Source: Author Documentation

This part is the main code where we take all the previous functions to simulate a real Wi-Fi signal processing and enhancement.

IV. Results and Discussion

After running the program with a random signal bit as the data. We get the result of two plots.

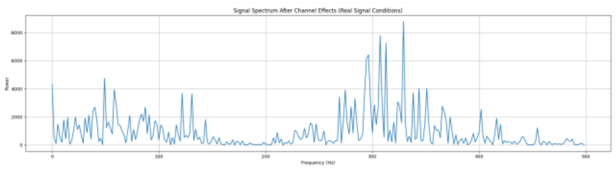


Figure 4.1 Simulation of Real Signal Condition
Source: Author Documentation

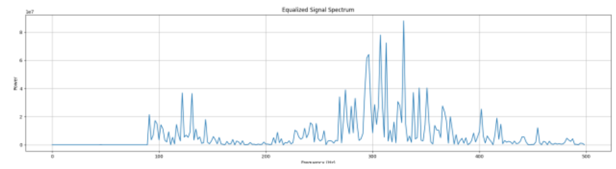


Figure 4.2 Equalized Channel
Source: Author Documentation

Those graphics have parameter power as the y-axis and frequency as the x-axis. Power represents the magnitude of signal's amplitude at each frequency bin. While frequency represents the frequency of the signal. The first graphic shows the power to frequency graph for the real signal condition. The real signal condition is our random generated bit data added by the multipath and the noise. The second graphic is the representation of the signal after being equalized using the Fast Fourier Transform algorithm.

Both graphics may look the same in general, but what we can't see from the picture is the equalized channel has increased power. The first graphic has a peak power of 8.820 Power, while the second graphic has the peak power of 8.820.000 Power. This indicates in power increasement by 1,000,000%. While it might sound good, if the power is too high it can lead to other problems. But in this case, the increased power can indicate stronger signals in specific frequency, meaning there is a significant boost in signal performance. In wireless communications and signal processing, higher power levels might indicate a stronger and more reliable signal. This reliable signal can help reduce noise and interference, leading to better performance.

In general, this result already shows that the Fast Fourier Transform, and the concept of complex numbers, can be utilized to enhance Wi-Fi performance. But there is one factor that indicates that this result is not good enough and it's the BER or the Binary Error Rate. This result gets a Binary Error rate of 49.27% meaning in the processing of the signal, around 49% of the original signal is lost or not processed properly, this indicates in result being not entirely accurate and there are more that should've been taken into account for this program. But overall, this result already show a positive sign regarding enhancing signal performance in wireless communication such as Wi-Fi that will be massively used in Smart Cities.

V. Conclusion

With this result, it can conclude that applying complex algebra concepts can indeed enhance Wi-Fi performance that later will be used for developing Smart Cities. Using the concept of complex algebra, we can use the Fast Fourier Transform algorithm to boost the power of the internet signal for better performance.

But even after that, it still can have weaknesses. Smart Cities are big and require a lot of internet connection and also a strong one. A high power is good but also can cause another problem, so enhancing the Wi-Fi performance also requires other regulations to make it really work in utilizing it for developing Smart Cities.

VI. Acknowledgment

With this, the author expresses the deepest gratitude to various parties who have assisted the author in the process of creating this paper. First, the author is grateful to God Almighty for His blessings that have guided and strengthened throughout the learning and writing process of this paper. The author also sincerely thanks the lecturer of class K-02 for the Linear Algebra and Geometry IF2123 course, Dr. Ir. Rinaldi Munir, M.T., for his guidance and teaching throughout the semester. Additionally, the author would like to extend heartfelt thanks to family and friends who have continuously supported and assisted the author in the learning process during the semester.

References

[1] W. T. Cochran *et al*., "What is the fast Fourier transform?," in *Proceedings of the IEEE*, vol. 55, no. 10, pp. 1664-1674, Oct. 1967, doi: 10.1109/PROC.1967.5957.

[2] D. Saha and T. G. Birdsall, "Quadrature-quadrature phase-shift keying," in *IEEE Transactions on Communications*, vol. 37, no. 5, pp. 437-448, May 1989, doi: 10.1109/26.24595.

[3] Cioni, M., Marzocchi, M., Biagi, S., & Scoppetta, S. (2012). Analysis of QPSK Modulation in Wireless Communication Systems. *The European Physical Journal Special Topics*, *217*(1), 13-22. https://doi.org/10.1140/epjst/e2012-01703-3J.

[4] Sharma, Toyesh & Sharma, Etisha. (2024). COMPLEX NUMBERS An Essential Tool for Mathematics.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 27 Desember 2024

ttd

Nama dan NIM