FFT-Based Spectrum Analyzer

Abstract

In this project, an FFT-based spectrum analyzer is designed and implemented to analyze the frequency content of time-domain signals. The limitation of time-domain analysis in identifying dominant frequency components motivates the use of frequency-domain techniques. Using the Fast Fourier Transform (FFT), sampled signals are converted into the frequency domain, where spectral characteristics such as dominant frequencies, noise floor, and spectral leakage are studied. The effect of additive noise and windowing techniques is also analyzed to understand real-world signal behavior. The complete system is implemented and validated in MATLAB.

1. Introduction

Signals encountered in practical communication and signal processing systems often contain multiple frequency components and noise. While time-domain representation shows how a signal varies with time, it does not explicitly reveal the frequency content of the signal. Frequency-domain analysis is therefore essential for applications such as spectrum sensing, interference detection, filtering, and modulation analysis.

A spectrum analyzer is a fundamental tool that displays signal amplitude as a function of frequency. In digital systems, the Fast Fourier Transform (FFT) provides an efficient way to compute the Discrete Fourier Transform (DFT) of sampled signals. This project focuses on designing an FFT-based spectrum analyzer to bridge the gap between time-domain observations and frequency-domain interpretation.

2. Problem Statement

Time-domain representation of a signal does not provide explicit information about its frequency components, bandwidth, or spectral distribution, especially in the presence of noise or multiple sinusoidal components. This makes it difficult to analyze signal characteristics using time-domain analysis alone. Therefore, an efficient frequency-domain analysis technique is required to extract and visualize the spectral content of a given signal.

3. Aim and Objectives

Aim

The aim of this project is to design and implement an FFT-based spectrum analyzer to convert a time-domain signal into its frequency-domain representation for effective spectral analysis.

Objectives

- To generate and sample time-domain signals
- To apply FFT and obtain the frequency spectrum
- To construct a meaningful frequency axis in Hertz
- To analyze the effect of noise on the frequency spectrum
- To study spectral leakage and its reduction using windowing techniques

4. Theory Background

4.1 Time Domain vs Frequency Domain

Time-domain analysis shows signal amplitude as a function of time, whereas frequency-domain analysis represents signal amplitude as a function of frequency. Frequency-domain representation is crucial for identifying dominant frequency components and interference.

4.2 Sampling

A continuous-time signal is converted into a discrete-time signal by sampling. According to the Nyquist criterion, the sampling frequency must be at least twice the maximum signal frequency to avoid aliasing.

4.3 Discrete Fourier Transform and FFT

The Discrete Fourier Transform (DFT) converts discrete-time signals into discrete frequency components but is computationally expensive. FFT is an efficient algorithm to compute the DFT with reduced computational complexity, making it suitable for practical applications.

4.4 Spectral Leakage

FFT operates on finite-length signals, which introduces discontinuities at the signal boundaries, leading to spectral leakage. Windowing techniques are used to reduce this effect.

5. System Model and Methodology

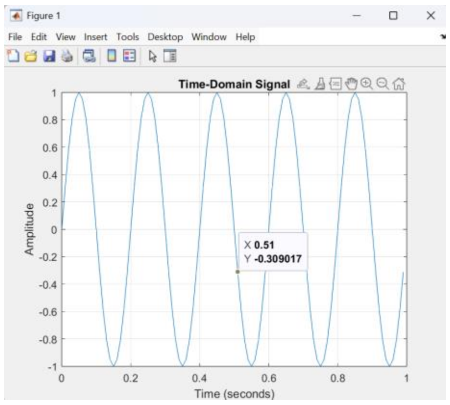
The implementation follows the steps below:

1. Generation of a sinusoidal signal
2. Sampling the signal to obtain discrete-time samples
3. Time-domain visualization
4. FFT computation
5. Frequency axis construction
6. Single-sided spectrum extraction
7. Noise addition and analysis
8. Windowing and leakage reduction

6. Implementation Details

6.1 Signal Generation

A sinusoidal signal of known frequency is generated and sampled using a predefined sampling frequency. The discrete-time signal represents the sampled version of the continuous signal.



6.2 FFT Computation

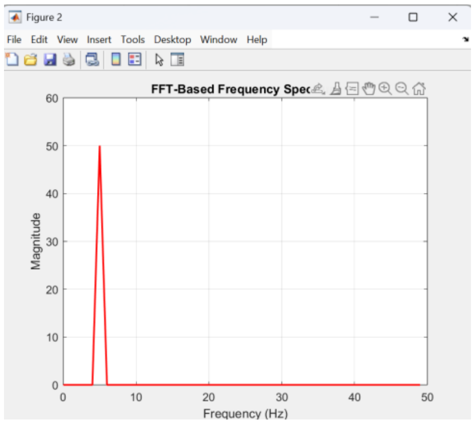
FFT is applied to the discrete-time signal to obtain frequency-domain components. The magnitude spectrum is computed from the complex FFT output.

6.3 Frequency Axis Mapping

FFT output indices are mapped to actual frequencies using the relation:

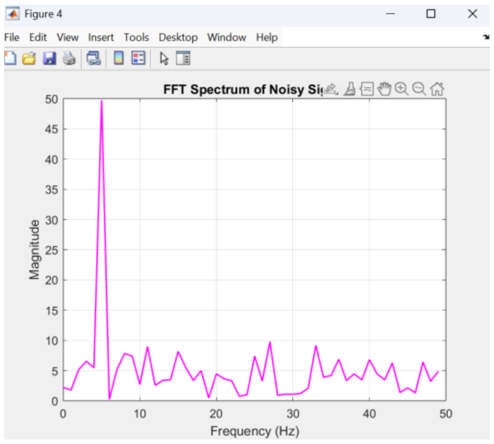
f_k = kFs / N

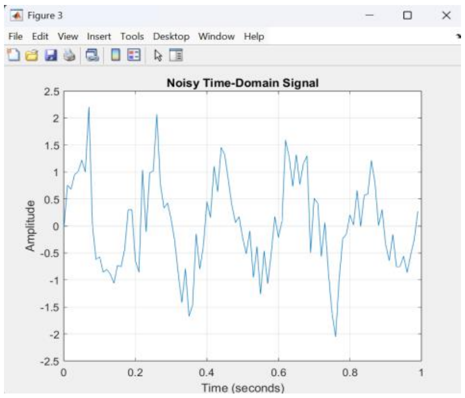
where Fs is the sampling frequency and N is the FFT length.



6.4 Noise Analysis

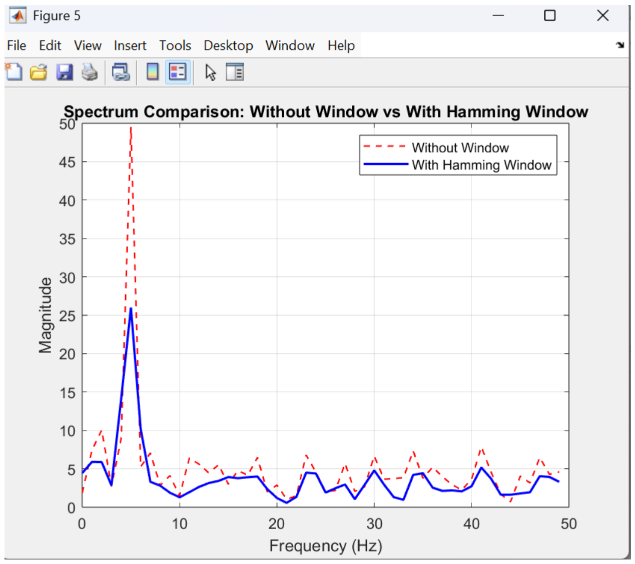
White Gaussian noise is added to the signal to simulate real-world conditions. The effect of noise is observed in both time and frequency domains.





6.5 Windowing

A Hamming window is applied to the noisy signal before FFT computation to reduce spectral leakage and improve frequency estimation accuracy.



7. Results and Observations

- The FFT spectrum of a clean signal shows a sharp peak at the signal frequency.
- When noise is added, the dominant frequency remains visible, while noise spreads across all frequencies forming a noise floor.
- Applying a Hamming window reduces spectral leakage and produces a cleaner frequency spectrum.

8. Conclusion

An FFT-based spectrum analyzer was successfully designed and implemented in MATLAB. The project demonstrated how FFT enables effective frequency-domain analysis of time-domain signals. The impact of noise and the importance of windowing techniques were clearly observed. This project highlights the practical significance of FFT in digital signal processing and communication systems.

9. Applications

- Spectrum sensing in wireless communication
- Interference detection and analysis
- Audio and speech signal processing
- Signal monitoring and diagnostics

10. Future Scope

- Extension to multi-tone and real-world signals
- Real-time spectrum analysis
- Comparison of different windowing techniques
- Integration with communication system simulations

11. References

1. A. V. Oppenheim and R. W. Schafer, *Signals and Systems*
2. J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*
3. MATLAB Documentation on FFT and Signal Processing