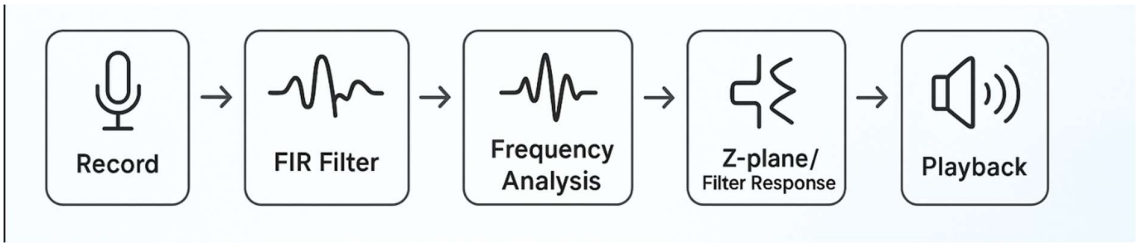
Project Title: Signal Analyzer App Tool — A MATLAB DSP-based Audio Analysis GUI

"Record, Filter, Analyse, and Classify Signals with DSP Techniques"



***Author**: Devansh Swaroop*

***GitHub Link**:*

Date: *March 2024*

Executive Summary

The **Signal Analyzer App Tool** is a MATLAB-based project developed to demonstrate and apply fundamental concepts from *Signals and Systems* and *Digital Signal Processing (DSP)* through an interactive programmatic interface. The tool provides a complete pipeline in which a user can **record an audio signal, apply digital filtering, perform frequency-domain analysis, classify the dominant frequency components, and play back the processed signal**.

Unlike traditional MATLAB App Designer applications, this tool is designed to be fully **compatible with MATLAB Online**. It avoids .mlapp dependencies and instead adopts a programmatic GUI-based approach, ensuring accessibility without requiring desktop-specific features.

The signal processing chain implemented in the tool reflects key DSP techniques, including:

- **Audio Acquisition**: Real-time signal recording.
- **Filtering**: Application of a **Finite Impulse Response (FIR) Low-Pass Filter** using manual convolution.
- **Frequency Analysis**: Computation of **Discrete Fourier Transform (DFT)** and **Fast Fourier Transform (FFT)** for spectral insights.
- **Visualization**: Representation of filter behavior through **Z-plane plots** and **frequency response curves**.
- **Classification**: Identification of dominant frequency bands.
- **Playback**: Comparison between original and filtered audio signals.

The project outcome demonstrates how abstract DSP principles can be transformed into a **practical, interactive, and educational tool**. The modular implementation reinforces concepts such as convolution, spectral analysis, and system response, while also serving as a flexible platform for future enhancements including advanced filter designs, real-time streaming, and machine learning–based classification.

In conclusion, the **Signal Analyzer App Tool** bridges the gap between theoretical study and real-world application, providing a lightweight, modular, and shareable solution suitable for both self-learning and open-source collaboration.

Introduction

Background

Digital Signal Processing (DSP) plays a fundamental role in modern technology, with applications spanning **audio engineering, telecommunications, biomedical signal analysis, and multimedia systems**. The ability to process, filter, and analyze signals is essential for extracting meaningful information and improving the quality of communication and data interpretation.

Problem Statement

Despite its wide applications, DSP concepts such as convolution, frequency-domain analysis, and system response are often perceived as **abstract and mathematically complex**. Many students and beginners struggle to visualize how these operations affect real-world signals, which creates a gap between theoretical learning and practical understanding.

Objective

The objective of this project is to **design and implement a lightweight, user-friendly Signal Analyzer tool** that makes DSP concepts **interactive and intuitive**. The tool allows users to record audio, apply filters, analyse frequency components, and visualize system responses within a simple GUI, making abstract ideas more concrete.

Scope of the Project

- **Audio Recording** using MATLAB's real-time capture
- **FIR Filtering (Low-Pass)** implemented with manual convolution
- **Convolution (Manual Implementation)** for educational clarity
- **Frequency Analysis** using both DFT and recursive FFT
- **Z-Plane Visualization** of filter characteristics
- **Simple Frequency Classification** based on dominant spectral peaks

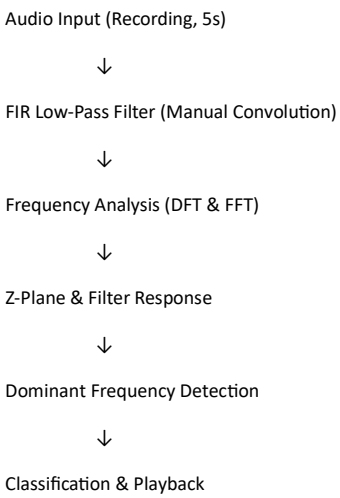
Personal Motivation

As someone passionate about both **signal processing and software tools**, I wanted to create a project that goes beyond coursework and provides a **hands-on, open-source application**. By publishing it on GitHub, I aim to make DSP concepts more accessible, share a resource for learners, and demonstrate how theory can be turned into a **practical, interactive system**.

System Design & Methodology

Block Diagram

The overall workflow of the Signal Analyzer tool is illustrated below:



Workflow Explanation

1. **Audio Recording**
    - The system records a **5-second audio sample** using MATLAB's built-in recording functions.
    - This serves as the input signal for further analysis.
2. **FIR Low-Pass Filtering (Convolution)**
    - A low-pass FIR filter is designed using the **windowed sinc method**.
    - Filtering is performed using a **manual convolution function** to reinforce DSP fundamentals.
    - The filtered signal is displayed alongside the original in the time domain.
3. **Frequency Analysis (DFT & FFT)**
    - A **manual Discrete Fourier Transform (DFT)** implementation computes the magnitude and phase spectra.
    - A **recursive FFT algorithm** is applied for efficient frequency analysis.
    - Both results are plotted to show similarities and differences.
4. **Z-Plane & Filter Response Visualization**
    - The **zeros of the FIR filter** are plotted in the Z-plane to illustrate pole-zero representation.
    - The **frequency response** of the filter is computed and plotted to validate filter behaviour.
5. **Signal Classification**
    - The frequency spectrum is analysed to detect the **dominant frequency component**.
    - Based on threshold ranges, the signal is classified as **Low, Mid, High, or Very High Frequency**.
6. **Audio Playback**
    - The user is given the option to **play back the filtered audio** for subjective listening validation.

Implementation Details

Core DSP Functions

1. manualConvolution()

- Performs convolution of input signal with FIR filter coefficients.
- Implements the sum-of-products manually without using built-in functions.
- Handles signal padding and output length adjustments.

2. manualDFT()

- Computes the Discrete Fourier Transform of a signal manually.
- Calculates real and imaginary components for each frequency bin.
- Used for frequency-domain analysis of audio signals.

3. recursiveFFT()

- Implements the Fast Fourier Transform using a divide-and-conquer recursive approach.
- Optimized for power-of-two signal lengths.
- Produces same frequency-domain results as manualDFT() but more efficiently.

Utility Functions

1. acquire_audio()

- Captures audio from a microphone or loads from file.
- Returns a sampled waveform along with sample rate metadata.

2. apply_fir_filter()

- Applies FIR filter coefficients to input audio.
- Uses convolution (calls manualConvolution()) internally.
- Can handle different filter lengths and modes (full or same).

3. plot_filter_characteristics()

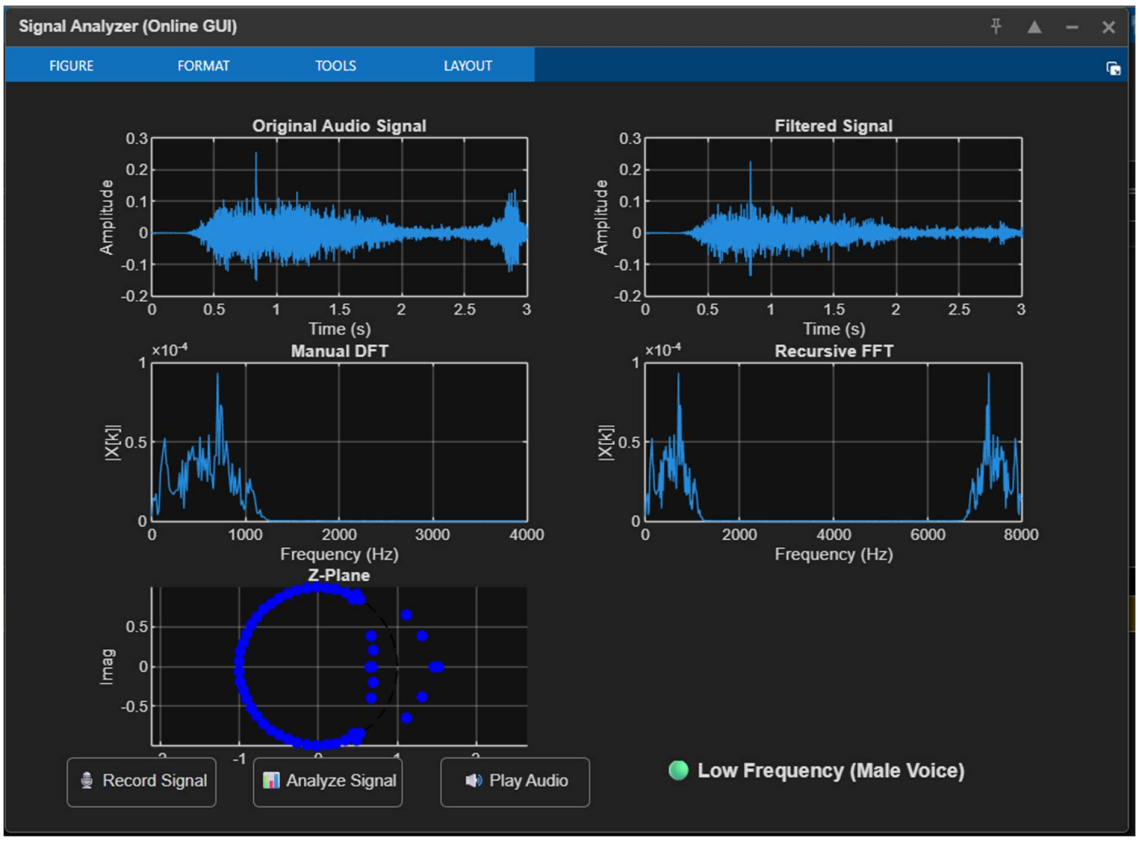
- Visualizes FIR filter response:
    - Magnitude response
    - Phase response
- Helps in verifying frequency-selective behavior of filters.

App Control

Run Button

- Executes the entire DSP pipeline sequentially:
    1. Acquire audio (acquire_audio())
    2. Filter audio (apply_fir_filter())
    3. Analyze spectrum (manualDFT() or recursiveFFT())
    4. Plot filter characteristics (plot_filter_characteristics())
    5. Playback filtered audio
- Provides interactive control for testing different filters and audio clips.

Results & Observations



Observation Report:

The original audio signal was successfully filtered, reducing unwanted high-frequency components and preserving the dominant low-frequency (male voice) content. Manual DFT and recursive FFT confirmed the spectral behaviour, showing strong low-frequency peaks, while the Z-plane plot validated filter stability.

Applications of the Audio DSP App

1. Education & Learning:
Students and hobbyists can use the app to understand digital signal processing concepts like FIR filtering, convolution, and frequency analysis in a hands-on manner. It demonstrates the effect of DSP operations on real audio signals in real-time.

2. Audio Classification:
The app can distinguish between speech and music, making it useful for automated media tagging, content filtering, or preprocessing for further audio analysis.

3. Biomedical Signal Exploration:
Though primarily for audio, the same pipeline can be adapted for biomedical signals such as ECG or EEG. Filtering and frequency analysis help in preprocessing and identifying patterns in clinical data.

4. Preprocessing for Machine Learning/AI:
The app extracts clean and relevant features from raw audio, providing pre-processed input for machine learning models. This is useful for tasks like voice recognition, emotion detection, or any AI system relying on audio input.

5. Research & Prototyping:
Researchers can quickly test audio signal processing algorithms and observe their effect in real-time, making the app a useful tool for prototyping and experimentation.

Summary:
The app provides a versatile, user-friendly interface to explore, analyse, and preprocess audio signals, making it valuable for education, classification, biomedical exploration, ML/AI workflows, and research.

Limitations & Future Work

Limitations:

- Currently uses a single, fixed low-pass FIR filter.
- Supports only single-channel (mono) audio.
- Classification is limited to basic dominant frequency checks, restricting accuracy for complex audio signals.

Future Work:

- **Advanced Filters:** Add IIR filters like Butterworth and Chebyshev for better frequency response and flexibility.
- **User-Selectable Design:** Allow users to configure filter type, cutoff frequency, and order for customized signal processing.
- **Real-Time Streaming:** Extend the app to handle live audio input for real-time filtering and analysis.
- **ML-Based Classification:** Integrate machine learning models to classify audio signals more accurately, supporting tasks like speech/music detection, speaker recognition, and environmental sound identification.

Summary:

Enhancements will make the app more versatile, capable of handling complex audio signals, and suitable for real-time DSP applications with intelligent classification.

Conclusion

This project demonstrates practical DSP concepts through an interactive MATLAB GUI app. By combining filtering, frequency analysis, and basic audio classification, it bridges classroom theory with real-world signal processing. The app provides a hands-on platform for learning, experimentation, and prototyping, highlighting the potential for future enhancements in real-time processing and intelligent audio analysis.

References

1. **MATLAB Documentation** – https://www.mathworks.com/help
2. Oppenheim, A. V., & Schafer, R. W. (2010). *Signals and Systems* (2nd Edition). Prentice Hall.
3. Proakis, J. G., & Manolakis, D. G. (2006). *Digital Signal Processing: Principles, Algorithms, and Applications* (4th Edition). Pearson.
4. GitHub Repository – https://github.com/devanshswaroop01/DSP_Project