Scilab case study project on

A Scilab GUI for Real-Time Audio Analysis and Manipulation

Fahad Ali

Visvesvaraya National Institute of Technology

July 25, 2024

Abstract

This case study explores the application of Scilab for audio processing, emphasizing the creation of a graphical user interface (GUI) for a range of audio manipulation tasks. It requires a thorough understanding of Scilab's handling of audio files, including concepts such as file storage, sampling rate, and digital filters. The primary objective is to develop a system capable of loading, playing, and visualizing audio files, alongside performing operations like pitch modification, noise filtering, and combining multiple audio files. Key concepts examined include the storage and sampling of audio files, the implementation of digital filters, and the use of Fourier Transform for frequency analysis. This comprehensive study not only delves into the theoretical aspects of audio processing but also addresses practical implementation challenges, demonstrating Scilab's capability in audio analysis and manipulation.

1. Problem Statement

The project aims to develop a GUI-based system with the following capabilities:

- Load and read audio files.
- Play audio files without distortion or at different frequencies.
- Generate an Amplitude vs. Time graph.
- Create an Amplitude vs. Frequency graph.
- Combine at least two audio files.
- Modify the pitch of audio files.
- Apply noise filtering to audio files.

Issues Faced

- Firstly, in GUI mode the graph cannot be deleted by the standard command instead xset(”auto clear”,”on”) is used. In changing pitch if the user modifies the pitch by more than 2 times or less than 2 times the data in the audio file gets down sampled/up sampled more than its sampling rate so the user is only given an option between 0.5-2 times.
- Secondly, In change pitch the original file’s pitch is copied and then only changed otherwise the same problem as above would occur as the down sampling or up sampling would get compounded.
- Thirdly, the file is transposed while changing the pitch so it is transposed back otherwise it cannot be plotted

2. Basic concepts related to the topic

1. Sampling Rate and Bandwidth

Sample rate is the number of samples of audio carried per second, measured in Hz or kHz (one kHz being 1000 Hz). For example, 44100 samples per second can be expressed as either 44100 Hz, or 44.1 kHz. Bandwidth is the difference between the highest and lowest frequencies carried in an audio stream. The sample rate determines the maximum audio frequency that can be reproduced. Theoretically the maximum frequency that can be represented is half the sample rate (known as the Nyquist frequency). In practice, the limit is a little lower, so the practical upper frequency limit for a sample rate of 44100 Hz, is a little over 20000 Hz, but less than22050Hz. The term bandwidth may be applied to the frequency content of an audio 3 signal stream, or the frequency ability of audio hardware or software. Although an audio interface may support a very high sample rate, for example 192000 Hz (192 kHz), which could practically support frequencies above 80 kHz, there is no guarantee that the audio interface does support such high frequencies. It is not uncommon for high quality audio interfaces to deliberately filter out extreme high frequencies to improve rejection of electromagnetic interference.

2. Fourier Transform

The Fourier transform (FT) decomposes a function often a function of the time, or a signal into its constituent frequencies. Where f is frequency and t is time period.

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i x\xi}dx$$

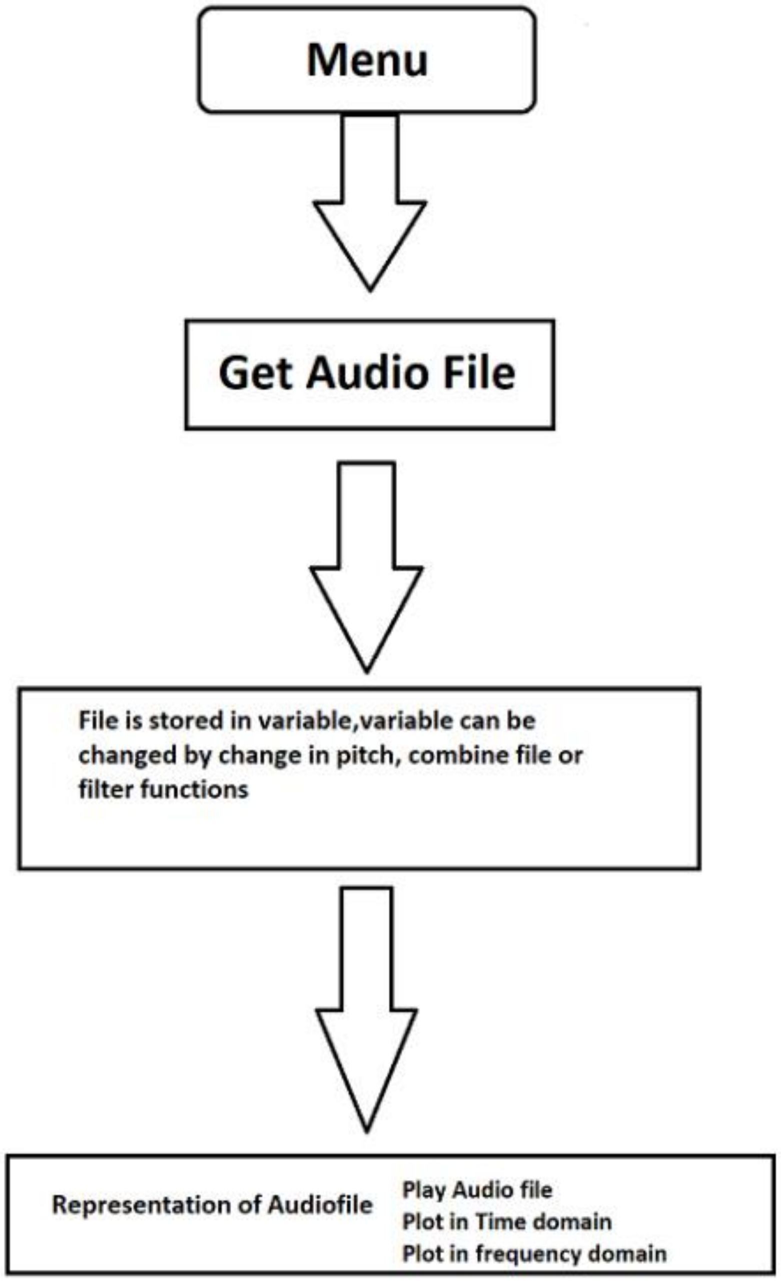
3. Fast Fourier Transform

A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields, but computing it directly from the definition is often too slow to be practical. An FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors. As a result, it manages to reduce the complexity of computing the DFT from O(N2) N is the data size.

4. Filter

In signal processing, a filter is a device or process that removes some unwanted components or features from a signal. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some frequencies or frequency bands. However, filters do not exclusively act in the frequency domain; especially in the field of image processing many other targets for filtering exist. Correlations can be removed for certain frequency components and not for others without having to act in the frequency domain. Filters are widely used in electronics and telecommunication, in radio, television, audio recording, radar, control systems, music synthesis, image processing, and computer graphics.

3. Flowchart


Menu
Get Audio File
File is stored in variable,variable can be changed by change in pitch, combine file or filter functions
Representation of Audiofile
Play Audio file
Plot in Time domain
Plot in frequency domain

4. Software/Hardware used

Hardware: An inbuilt microphone was used to record the audio.

Software:

- Scilab 2024.1.0 on Windows 10
- Audacity was also used to record and convert the audio file

5. Procedure of execution

1. Load Audio File: Click on "Get Sound File". This function presents the user with an explorer to browser for the audio file. The audio file must be of WAV type.

2. Play Audio: The user can play the audio file by clicking on "Play".

3. Analyze in Time Domain: The user can choose to view the audio signal in the time domain by clicking on "Analyze in Time Domain".

4. Analyze in Frequency Domain: The user can analyze the audio signal in the frequency domain by clicking on "Analyze in Frequency Domain".

5. Combine Audio Files: Click on "Combine Audio Files". This will combine the first audio file that the user selected in "Get Sound File" with the second audio file that the user selects.

6. Change Pitch: The user can change the pitch of the audio file by clicking on the options presented. It will multiply the pitch by the selected factor.

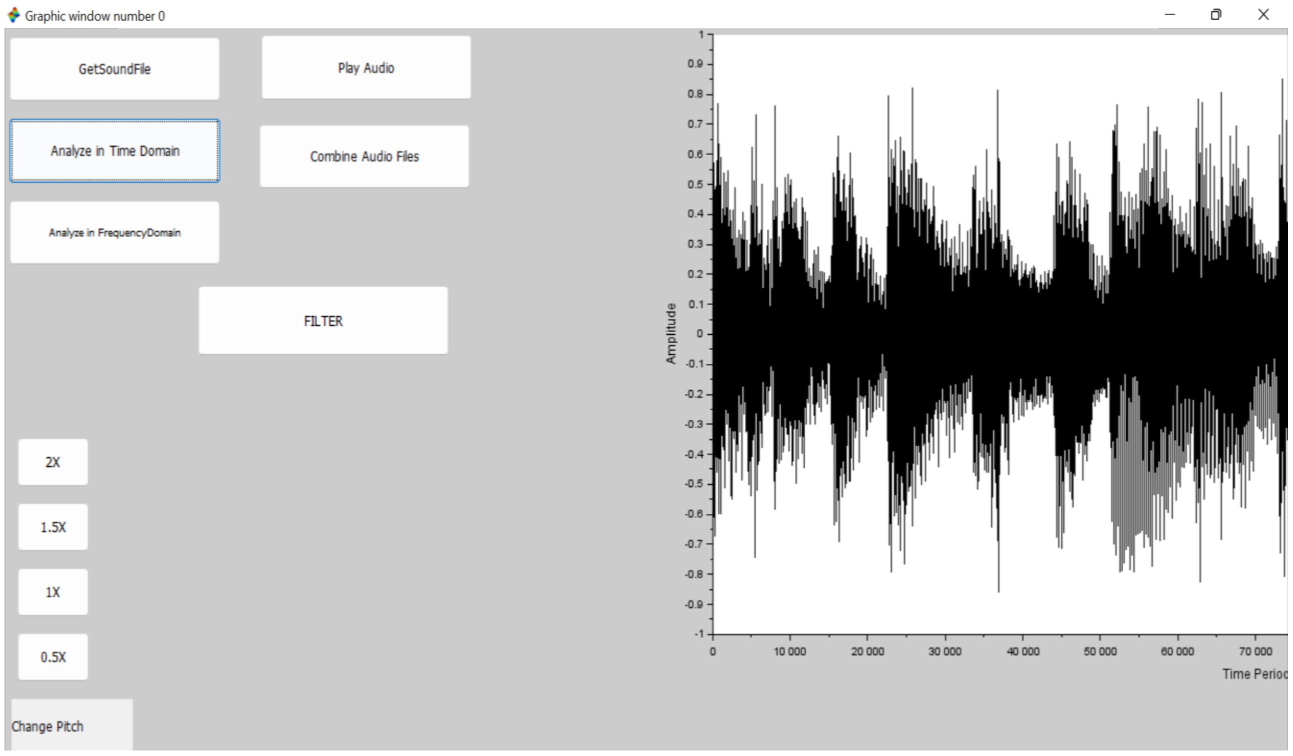
7. Filter Audio: The user can filter the audio by clicking on "Filter". They are then presented with a filter menu.

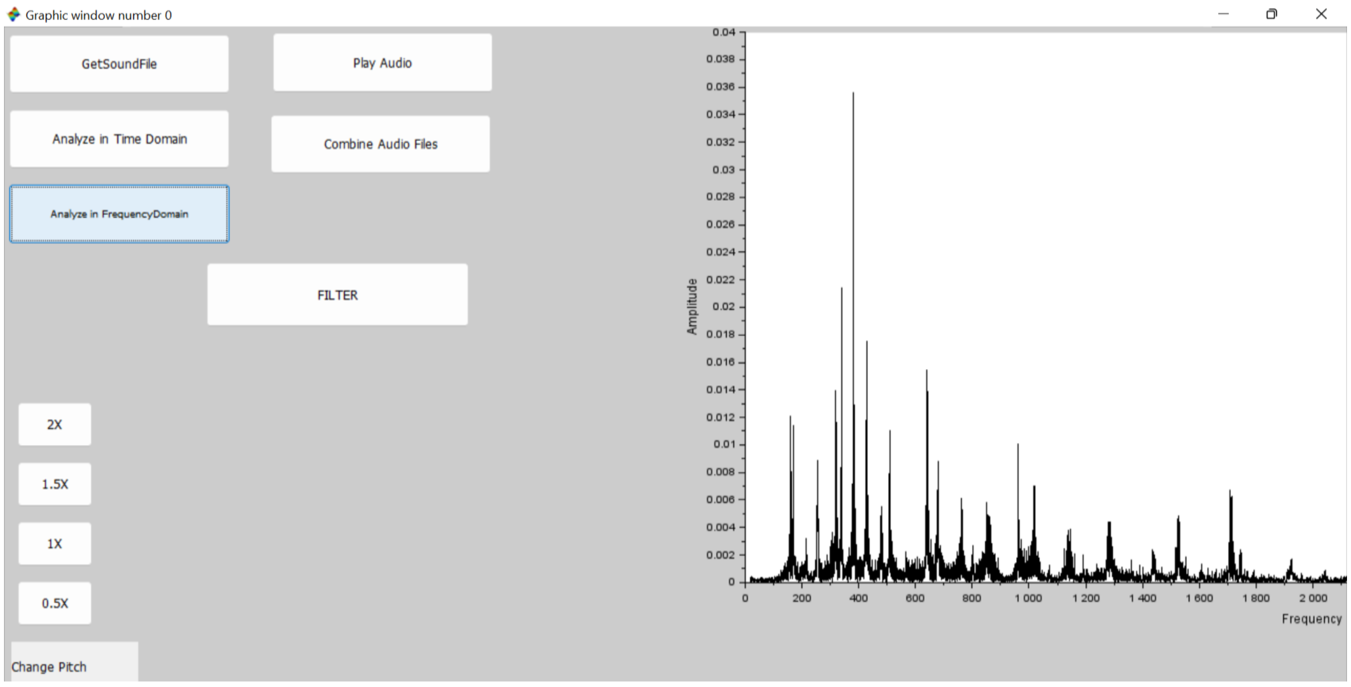
8. Select Sampling Rate: The user must select 44100 as the sampling rate or any other rate if different.

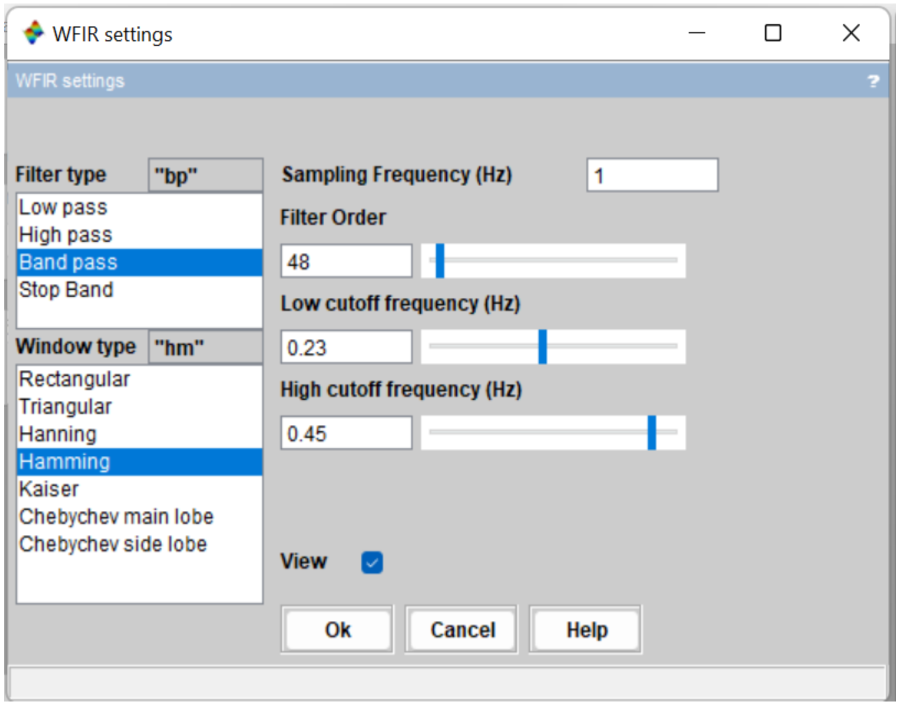
9. Select Filter Type: The user must select the type of filter. By default, the user can select the Low Pass Rectangular Filter.

10. Apply Filter: Click "OK" and wait for some time for the filter to process.

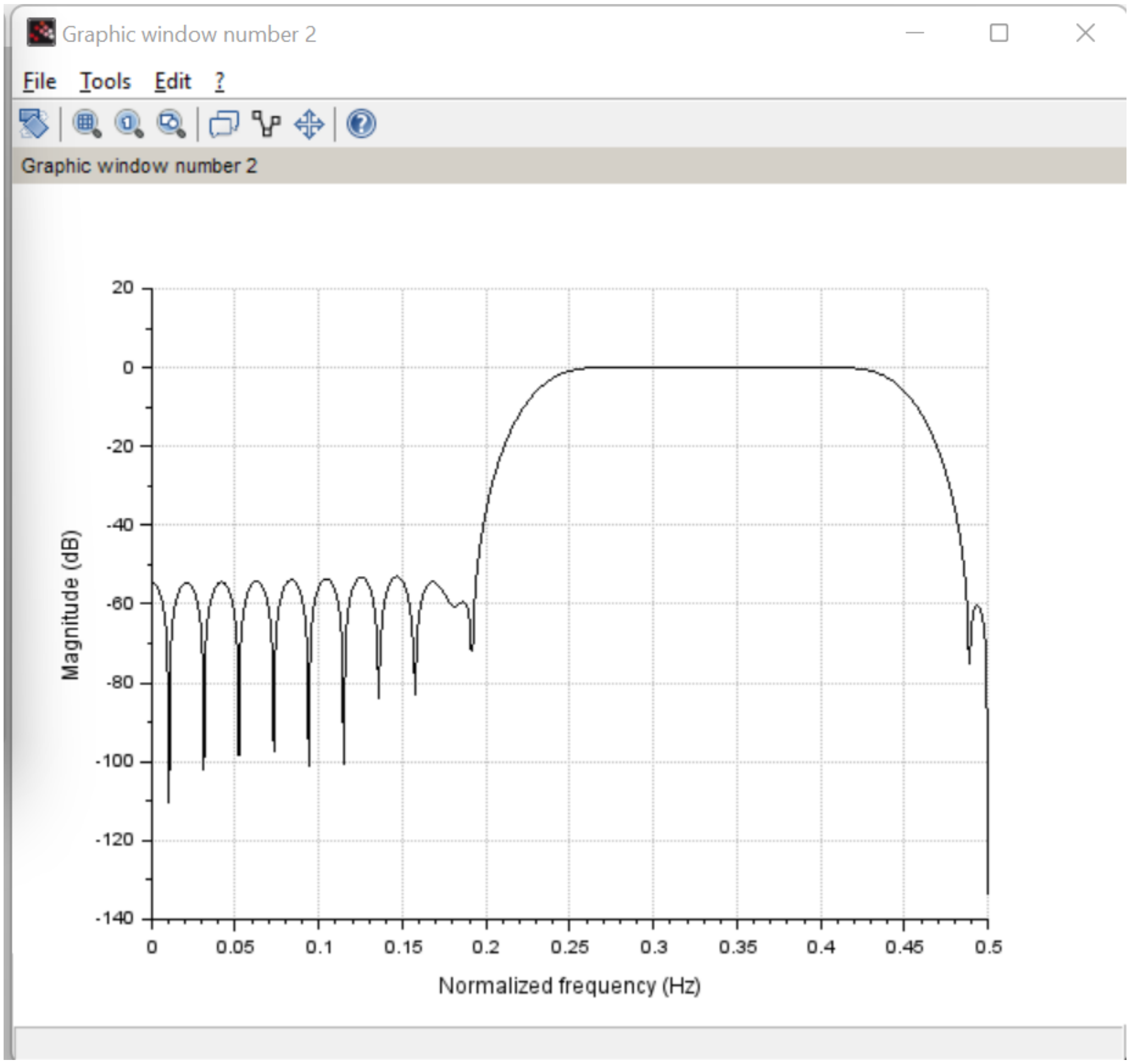
6. Result







Filter type "bp" — Band pass
Window type "hm" — Hamming
Sampling Frequency (Hz) 1
Filter Order 48
Low cutoff frequency (Hz) 0.23
High cutoff frequency (Hz) 0.45
View ✓



7. References

(1)Digital Signal Processing Paperback by Oppenheim/ Schafer

(2) https://scilab.in/resources

(3) 2.3.4 and 3.1 Development of Real Time Audio Equalizer Application using MATLAB App Designer By Johannes Langelaar ,Adam Str¨omme Mattsson,Filip Natvig Uppsala Universitet