CHAPTER 1

INTRODUCTION

Enhancement of Speech signal is a technique which deals in processing of the noisy speech signals and aims in improving their intelligibility and quality or their proper decoding by machines. It is required that the noise signal is introduced in clean speech signal is to be minimized. This noise reduction technique finds applications in multidisciplinary areas such as, Communication Systems and Speech recognition Hearing aids, etc. Hence, the quality of speech signal can be improved using speech enhancement technique.

The essential purpose of enhancement of speech is to minimize the noise in original speech signal and enhance the one or more intuitive facets of speech signal such as quality and comprehensible. Therefore, to increase speech intelligibility, the speech enhancement techniques are broadly classified into, single and multi channel enhancement techniques. In speech enhancement by single channel technique both clean speech and noises are present together. In multi channel speech enhancement technique both noise signal and clean speech are present separately. The motivating factor for using filter banks in speech enhancement is, it is possible to control magnitude distortion and phase distortion resulting due to transform domain processing. These are not possible from conventional methods of speech enhancement.

In recent years, multi-channel speech recognition has been applied on devices used in daily life, such as Amazon Echo and Google Home. The recognition accuracy is greatly improved by exploiting Cascaded filter bank arrays when compared to single channel filters. However, satisfactory performance is still not achieved in noisy everyday environments.

The advances in speech recognition have led to performance improvement in challenging scenarios such as noisy and far field conditions. However, speech recognition systems still perform poorly when the audio of interest is recorded in crowded environments, i.e., with interfering audio signal in the foreground or background.

Signal Processing is one of the important research fields that is used widely these days in different aspects of our lives. Filter bank is an array of band-pass filters that separates the input signal into multiple components, each one carrying a single frequency sub-band of the original signal. The process of decomposition performed by the filter bank is called analysis (meaning analysis of the signal in terms of its components in each sub-band); the output of analysis is referred to as a sub band signal with as many sub bands as there are filters in the filter bank. The reconstruction process is called synthesis, meaning reconstitution of a complete signal resulting from the filtering process.

CHAPTER 2

LITERATURE SURVEY

A literature survey is basically a compilation of significant research published an a topic by accredited scholars and researchers. In this chapter, a number of IEEE research papers on the topic of Multirate filtering were studied and their summary is given below.

- L.lin and E.Ambikairajah study was to improve the noisy speech quality by removing the white and colored noise in each sub band channels. The author has used the auditory filter bank structure. It consists of narrow band pass filters and placed according to equivalent rectangular bandwidth (ERB). The proposed speech denoising system consists of analysis filter and synthesis filter. The denoising is achieved by multiplying the output of each filter by scaling factor and calculated signal and variances. This method can gives better quality speech denoising even in present of colored noise. This algorithm performs better in presence of both white noise and colored noise.

- Ying Deng et.al., the author focused on design of the low delay non-uniform Quadrature Mirror Filter (QMF) banks, to achieve the acute transition band and high stop band attenuation. Both acute transition band and high stop band attenuation allow the non-uniform processing of the sub bands signals without introducing notable aliasing distortion into the reconstruction signal. The author has proved that synthesis prototype has to be extended version of analysis prototype filter, in order to remarkable aliasing cancellation to occur in the less delay uniform QMF banks and to accomplish notable distortion cancellation in less delay non uniform filter bank, But this method shows slightly greater residual noise in segmented SNR levels as compared to Canon's bark extended wavelet packet decomposition.

- Bastian Sauert et.al., proposed warped low delay filter bank using Discrete Fourier Transformer(DFT).In this method was used time and frequency dependent approach which intensifies far end speech signal to retrieve a target signal to noise ratio(SNR).This algorithm consists of analysis synthesis filter bank, implemented by overlap add method with non uniform filter bank equalizer.In non uniform filter bank equalizer designed by all pass transformation, it changes both magnitude and phase response of filter. The proposed algorithm was evaluated the term of speech intelligibility index, apart from approximately calculation of spectral weights in bark scaled frequency band for the non uniform frequency resolution of human ear.

- Asha Vijaykumar author suggested novel unimodular filter bank algorithm which is the noisy input signal, processed by an M-channel one regular unimodular filter bank. The each M sub band channel output are scaled by scaling factor and computer. with varying input signal, by using unimodular filter bank we have to reduce the sub band based noise suppression, after that we get speech and noise database and the database used for automatic speech recognition. The output of filter provides the effective noise reduction as well as improve recognition rate at less delay speech signal. The demerit of this filter bank is produce highly non stationary noise.

- Yu Cai et.al., author conveyed the sub band spectral subtraction algorithm based on over sampled DFT modulated filter bank .Here the input signal is filtered by k channel analysis filter and it is decimated by factor M, the decimated signal transforms into M sub band channels. These sub band signals are processed and interpolated by synthesis filter, these filter can remove the musical noise in speech enhancement and output of synthesis filter can sum together to get enhancement of speech signal, the enhanced speech signal gives better result rather than multiband method. But this filter bank also gives the same result as Kamath method in speech enhancement. The proposed algorithm may suppress the noise, to preserve relatively integrated speech component. This algorithm is also adaptable method that can acquaint into many audio applications.

- Alexander Schase et.al., author presented cascaded filter bank algorithm which increases the frequency resolution without changing original filter bank system. The proposed algorithm, A time-domain signal is divided into oversampling samples which weighted by a analysis prototype low pass filter and these samples processed by sub band signal samples to get the frequency bins it depends on the sampling frequency of time domain signal. The frequency resolution in the low frequency range of the analysis system can be increased by decomposing sub band signals, if filter can cascaded it can reduce the spectral gain, but in this system gains of the a frequency domain wiener filter. The additional delay caused by cascaded system can be reduced by low delay filterbank.

- Jiri Malek et.al.anticipated compare trained bank (L-CFB) and manually constructed bank(M-CFB) algorithms are used in real time scenario for noise free recording of mobile phones. This method described here scans the noise free recording, this data processed block by blocks it may overlap. In beginning stage the filter bank is empty. The algorithm computes target cancelation filter in the first block and it puts into the filter bank if output variance is smaller than threshold it detect the object. If variance is higher than, the algorithm is processed in the next block, this is repeated until a fir target cancellation filter is added to the filter bank. Here each block is in parallel; filtered by all cancellation filters in the current filter bank and their output are

compared. If cancellation filter bank yield the minimum output variance over first object, a new cancellation filter computed for the block of signal. The resulting filter bank is added to the filter bank. The resulting bank will henceforth be referred to as one pass target cancellation filter bank.

- Kristian Timm Andersen et.al., author proposed Fast Fourier transformer algorithm is used to an adaptive time-frequency analysis/synthesis scheme for processing of audio less delay and less complexity. The adaptive time and frequency scheme is used for noise suppression gain factor by analysis/synthesis method. The adaptive time-frequency can adopt its frequency resolution and develop the suitable time delay. The analysis filter is incepted into the frame of the DFT modulated filter bank and these filters are used to determine the gain factors while synthesis uses DFT filter bank . The synthesis method uses an asymmetric window it is mixed with twiddle filter bank that maintains less delay of the reference filter bank but improves the frequency response. Additional delay can be reduced by applying the gain to signal. The complexity of the filter bank decreased in frequency by the FFT algorithm in the DFT filter bank and it is calculated by number of frequency band which selected in FFT algorithm.

- Pinki, Sahil Gupta author proposed Spectral Subtraction method. This method operates on the frequency domain; the input signal may be presented as the addition of the speech signal and noise signal. In this method, first we have to estimates spectrum of the background musical noise and eliminating the noise spectrum from input speech signal. To perform the frequency domain processing, the continuous time signal cab be split into overlapping pieces is called frames. These frames can operate; the frames can reassemble to create a non-discrete output signal. To avoid the spectral effects, we have to multiply the window function before processing the fast Fourier transformer and again after evaluating inverse fast Fourier transformer. In this technique the speech spectrum is divided into number of frequency bands and spectral subtraction is operated independently and this algorithm can readjust the over subtraction factor in every frequency band, to estimate the clean speech magnitude spectrum. This algorithm removes the additive noise and remnant noise using spectral subtractive type algorithm.

We can summarize the chapter that, based on the research papers published by the above authors, the research suggests that the performance of Multirate system is better than traditional filters in various domains. The focus of this project will be to verify the same in the case of audio processing.

CHAPTER 3

PROBLEM IDENTIFICATION

The first step in the problem solving and decision making process is to identify and define the problem. A problem can be regarded as a difference between the actual situation and the desired situation. This chapter attempts to do the same.

The common filters used in the electronic devices today are Finite Impulse Response Filters and Infinite Impulse Response Filters. Most notable ones are Butterworth, Chebyshev, Bessel, Elliptical filters. However the FIR filters sometimes have the disadvantage that they require more memory for calculation to achieve a given filter response characteristic also certain responses are not practical to implement with FIR filters.

Similarly, IIR filters are more susceptible to problems of finite-length arithmetic, such as noise generated by calculations, and limit cycles. (This is a direct consequence of feedback: when the output isn't computed perfectly and is fed back, the imperfections can compound.), are harder (slower) to implement using fixed-point arithmetic and don't offer the computational advantages of FIR filters for Multirate (decimation and interpolation) applications.

These can be overcome by using Multirate filters. Potential advantages of Multirate signal processing are reduced computational work load, lower filter order, lower coefficient sensitivity and noise, and less stringent memory requirements. Disadvantages are more complex design, aliasing and imaging errors and a more complex algorithm. Hence this project aims to demonstrate the performance of Cascaded Multirate systems.

The most immediate reason is when you need to pass data between two systems which use incompatible sampling rates. For example, professional audio systems use 48 kHz rate, but consumer CD players use 44.1 kHz; when audio professionals transfer their recorded music to CDs, they need to do a rate conversion.

But the most common reason is that Cascaded Multirate can greatly increase processing efficiency (even by orders of magnitude), which reduces system cost. This makes the subject of Cascaded Multirate DSP vital to all professional DSP practitioners.

We can summarize this chapter by saying that there are currently numerous problems associated with audio processing in various domains, such as low efficiency, high system cost and other problems.

CHAPTER 4

OBJECTIVE

An objective describes the desired results of a project, which often includes a tangible item. An objective is specific and measurable, and must meet time, budget, and quality constraints. This chapter strives to define the objectives for this project.

The objectives of this project is to design and demonstrate a digital filter that is more effective and can overcome the limitations of the conventional Finite Impulse Response Filters and Infinite Impulse Response Filters, the main focus being to develop an algorithm that can be used.

The primary objectives of this project are as follows:

- To eliminate the noise associated with clear speech.
- To minimize the musical noise resulting due to transform domain processing.
- To increase the intelligibility and quality of speech signal.

The design and implementation objectives of this project are as follows:

- To implement FIR, IIR and Multirate filters.
- To design and write code required to realize these filters.
- To compare the performance of the aforementioned filters on the basis of noise reduction
- To visualize the outputs of the aforementioned filters on application of the same input sample and plot the graphs.

Hence, this chapter can be summarized as the objectives of this project are mainly to design and implement filters that can filter the noise efficiently from the audio samples and to plot the graphs of the filtered output, in order to visualize the performance.

CHAPTER 5

METHODOLOGY

This chapter deals with explanation of various filters currently being used and the Multirate filtering. Methodology is the systematic, theoretical analysis of the methods applied to a field of study. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge. Typically, it encompasses concepts such as paradigm, theoretical model, phases and quantitative or qualitative techniques. The details for each are as explained below.

There are two major types of filters that are currently being used in the industry and applications. They are: Finite Impulse Response Filters (FIR) and Infinite Impulse Response (IIR). They are further explained with block diagrams in detail as follows:

5.1 Finite Impulse Response Filter

FIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP). "FIR" means "Finite Impulse Response." If you put in an impulse, that is, a single "1" sample followed by many "0" samples, zeroes will come out after the "1" sample has made its way through the delay line of the filter. In the common case, the impulse response is finite because there is no feedback in the FIR. A lack of feedback guarantees that the impulse response will be finite. Therefore, the term "finite impulse response" is nearly synonymous with "no feedback". However, if feedback is employed yet the impulse response is finite, the filter still is a FIR. An example is the moving average filter, in which the Nth prior sample is subtracted (fed back) each time a new sample comes in. This filter has a finite impulse response even though it uses feedback: after N samples of an impulse, the output will always be zero. Fig 5.1 is the block diagram of Finite Impulse Response filter.

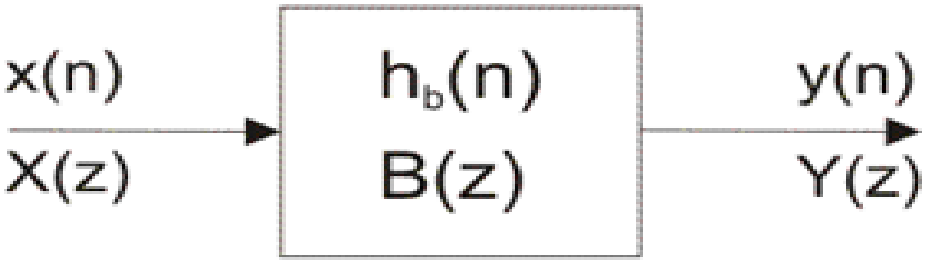


Fig 5.1: Block Diagram of Finite Impulse Response Filter

5.2 Infinite Impulse Response Filter

The impulse response of IIR Filter is “infinite” because there is feedback in the filter; if you put in an impulse (a single “1” sample followed by many “0” samples), an infinite number of non-zero values will come out (theoretically.) IIR filters can achieve a given filtering characteristic using less memory and calculations than a FIR filter. Figure 5.2 is the block diagram of Finite Impulse Response filter.

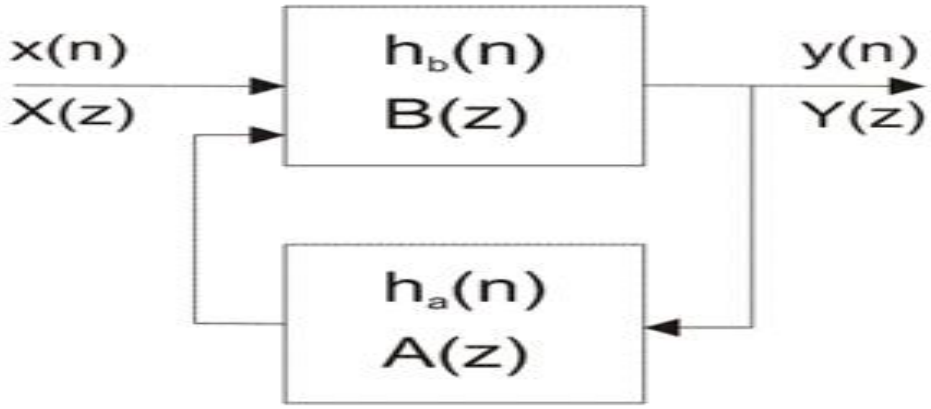


Fig 5.2: Block Diagram of Infinite Impulse Response Filter

5.3 Multirate Filter

Multirate simply means “multiple sampling rates”. A multirate DSP system uses multiple sampling rates within the system. Whenever a signal at one rate has to be used by a system that expects a different rate, the rate has to be increased or decreased, and some processing is required to do so. Strictly speaking, "multirate" means you are processing or sampling the signals at different rates in the same system. Usually the rates are multiples of one another. Fig 5.3 is the block diagram of Cascaded Multirate System.

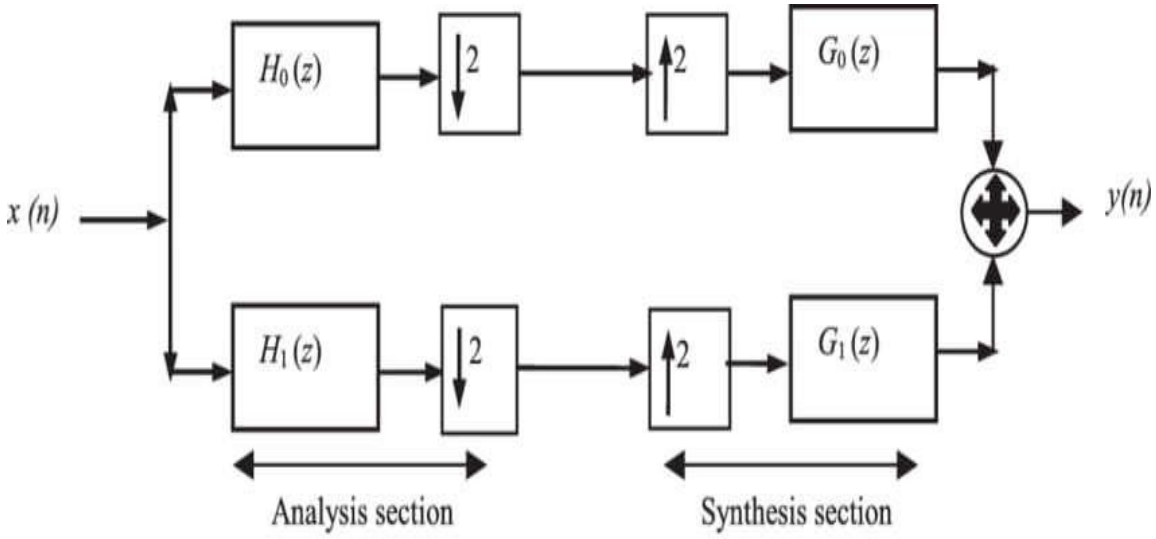


Fig 5.3: Block Diagram of Cascaded Multirate System

Multirate system is a filter that provides speech enhancement and it consists of 3 main steps:

5.3.1 DECIMATION

"Decimation" is the process of reducing the sampling rate. In practice, this usually implies low pass-filtering a signal, then throwing away some of its samples. The most immediate reason to decimate is simply to reduce the sampling rate at the output of one system so a system operating at a lower sampling rate can input the signal. But a much more common motivation for: Decimation is to reduce the cost of processing the calculation and/or memory required to implement a DSP system generally is proportional to the sampling rate, so the use of a lower sampling rate usually results in a cheaper implementation. This process, along with the low pass filter is also called the Analysis section, which is shown in Fig 5.3.

5.3.2 INTERPOLATION

"Up sampling" is the process of inserting zero-valued samples between original samples to increase the sampling rate. (This is called "zero-stuffing".) Up sampling adds to the original signal undesired spectral images which are centered on multiples of the original sampling rate. "Interpolation", in the DSP sense, is the process of up sampling followed by filtering. The result is as if you had just originally sampled your signal at the higher rate. The primary reason to interpolate is simply to increase the sampling rate at the output of one system so that another system operating at a higher sampling rate can input the signal. This process, along with the high pass filter is also called the Synthesis section, which is shown in Fig 5.3.

5.3.3 RESAMPLING

"Resampling" means combining interpolation and decimation to change the sampling rate by a rational factor. Resampling is usually done to interface two systems which have different sampling rates. If the ratio of two system's rates happens to be an integer, decimation or interpolation can be used to change the sampling rate; otherwise, interpolation and decimation must be used together to change the rate. The interpolation factor is simply the ratio of the output rate to the input rate. Given that the interpolation factor is L and the decimation factor is M, the resampling factor is L / M.

This chapter explains the various details and differences between various filters, such as Finite Impulse Filter, Infinite Impulse Filters and Multirate Filters, that are to be implemented in this project. The concepts of Multirate filters, such as decimation, interpolation and resampling are explained extensively in this chapter.

CHAPTER 6

SOFTWARE DESIGN

This chapter deals with the various types of software used and how the same is used in the project.

6.1 Operating System (OS)

Linux or Windows can be used as the software environment for running this project. The relevant details about the aforementioned OS's are as follows:

6.1.1 Linux

Linux is one of the most prominent examples of free and open-source software collaboration. The source code may be used, modified and distributed commercially or non-commercially. Linux is typically packaged in a Linux distribution (or distro for short). Distributions include the Linux kernel and supporting system software and libraries, many of which are provided by the GNU Project. Linux also runs on embedded systems, i.e. devices whose operating system is typically built into the firmware and is highly tailored to the system. This includes routers, automation controls, televisions, digital video recorders, video game consoles, and smart watches.

6.1.2 Windows

The Microsoft Store package is a simple installation of Python that is suitable for running scripts and packages, and using IDLE or other development environments. It requires Windows 10, but can be safely installed without corrupting other programs.

6.2 Software Used

6.2.1 Python

Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp. Python's developers strive to avoid premature optimization. Additionally, they reject patches to non-critical parts of the CPython reference implementation that would provide improvements on speed. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C or PyPy, a just-in-time compiler. Cython is also available. It translates a Python script into C and makes direct C-level API calls into the Python interpreter.

6.2.2 Python Applications

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

- Web development.
- Scientific programming.
- Desktop GUIs.
- Network programming.
- Game programming.

6.2.3 Python Syntax

Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

6.2.4 Python Version

The latest stable version (as of Winter 2019) is Python 3.7.2. This project was built for the same. However, depending upon the changes in syntax and other factors, the code may work or break when used with other versions of Python.

6.3 Python Libraries used

6.3.1 Numpy

It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays. Latest stable version of numpy(1.8.2) is used. The following are the numpy functions used in this project:

6.3.1.1 random

This creates an array of specified shape and fills it with random values.NumPy random numbers. An important part of any simulation is the ability to generate random numbers. For this purpose, NumPy provides various routines in the submodule random . It uses a particular algorithm, called the Mersenne Twister, to generate pseudorandom numbers.

6.3.1.2 log10

Return the base 10 logarithm of the input array, element-wise. The logarithm to the base 10 of x, element-wise. NaNs are returned where x is negative.

6.3.1.3 int16

int16 Integer (-32768 to 32767) it is easy to processing and storage.

6.3.1.4 array

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

6.3.1.5 array_equal.

Returns True if two arrays have the same shape and elements, False otherwise.

6.3.1.6 asarray

Converts the input to an array.

6.3.1.7 cos

Trigonometric cosine element-wise.

6.3.1.8 sin

Trigonometric sine, element-wise.

6.3.1.9 absolute

Calculates the absolute value element-wise.

6.3.1.10 arrange

Return evenly spaced values within a given interval.

6.3.1.11 array

Create an array of elements.

6.3.1.12 randint

Return random integers from low (inclusive) to high (exclusive).

6.3.1.13 randn

Return a sample (or samples) from the standard normal distribution.

6.3.1.14 roll

Roll array elements along a given axis. Elements that roll beyond the last position are re-introduced at the first.

6.3.1.15 kron

Kronecker product of two arrays. Computes the Kronecker product, a composite array made of blocks of the second array scaled by the first.

6.3.1.16 zeros

Return a new array filled with zeroes.

6.3.1.17 floor

Return the floor of the input, element-wise. The floor of the scalar x is the largest integer i, such that i <= x.

6.3.1.18 sinc

Return the sinc function. The sinc function is sin(pi x)/(pi x).

6.3.1.19 kaiser

Return the Kaiser window. The Kaiser window is a taper formed by using a Bessel function.

6.3.1.20 mod

Return element-wise remainder of division. Computes the remainder complementary to the floor_divide function. It is equivalent to the Python modulus operator ‘x1 % x2’ and has the same sign as the divisor x2. The MATLAB function equivalent to np.remainder is mod.mod is an alias of remainder.

6.3.1.21 r_

Translates slice objects to concatenation along the first axis. This is a simple way to build up arrays quickly.

6.3.1.22 ceil

Return the ceiling of the input, element-wise. The ceil of the scalar x is the smallest integer i, such that i >= x.

6.3.2 SciPy

SciPy (pronounced “Sigh Pie”) is open-source software for mathematics, science, and engineering. It is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling systems such as MATLAB, IDL, Octave, R-Lab, and SciLab.

The additional benefit of basing SciPy on Python is that this also makes a powerful programming language available for use in developing sophisticated programs and specialized applications. Scientific applications using SciPy benefit from the development of additional modules in numerous niches of the software landscape by developers across the world.

Everything from parallel programming to web and data-base subroutines and classes have been made available to the Python programmer. Scipy version 1.3.0 is the version used.

The sub-modules used in the project are as follows:

6.3.2.1 Scipy.io:

The Scipy.io (Input and Output) package provides a wide range of functions to work around with different format of files. Some of these formats are : Matlab, IDL,Matrix Market, Wave, Arff, Netcdf, etc. So out of this wave file is used since wave file is a simple implementation of audio storage. In this sub-module, the following functions are used:

6.3.2.1.1 wavefile.read

Open a WAV file and return the sample rate (in samples/sec) and data from a WAV file. This function cannot read wav files with 24-bit data, which is a high quality bit rate. Hence we use 16-bit wav files as audio data, sampled at 44100 Hz or 44.1 kHz, since it is better for processing and storage, when compared to 24-bit wav files.

6.3.2.1.2 wavefile.write

Write a numpy array as a WAV file. Writes a simple uncompressed WAV file.To write multiple-channels, a 2-D array of shape (Nsamples, Nchannels) can be used. The bits-per-sample and PCM/float will be determined by the data-type. Hence, we use int-16 to write 16-bit wav files, to preserve standardization between the input and output of the program.

6.3.2.2 Scipy.signal

This module contains all the functions that are related to manipulation of signals. The functions used are as follows:

6.3.2.2.1 freqz

The scipy library of matplotlib in Python 3 contains a function scipy.signal.freqz which computes the frequency response of a digital filter.

6.3.2.2.2 Iirfilter

This filter class is capable to do low or high or bandpass and stopband filterings with different filter designs: Butterworth or Chebyshev Type I/II. Design an Nth-order digital or analog filter and return the filter coefficients.

6.3.2.2.3 butter

To create a bandpass Butterworth filter. The Butterworth filter is a type of signal processing filter designed to have a frequency response as flat as possible in the passband. It is also referred to as a maximally flat magnitude filter.

6.3.2.2.4 lfilter

Linear filters process time-varying input signals to produce output signals, subject to the constraint of linearity. This results from systems composed solely of components (or digital algorithms) classified as having a linear response. Most filters implemented in analog electronics, in digital signal processing, or in mechanical systems are classified as causal, time invariant, and linear signal processing filters. The general concept of linear filtering is also used in statistics, data analysis, and mechanical engineering among other fields and technologies

6.3.3 pylab

Pylab combines the pyplot functionality (for plotting) with the numpy functionality (for mathematics and for working with arrays) in a single namespace, making that namespace (or environment) even more MATLAB-like. For example, one can call the sin and cos functions just like you could in MATLAB, as well as having all the features of pyplot.pylab interface is convenient for interactive calculations and plotting.

It is a "magic function" that you can call within IPython, or Interactive Python. By invoking it, the IPython interpreter will import matplotlib and NumPy modules such that you'll have convenient access to their functions. Some of Pylab's functions used are figure, clf, plot, xlabel, ylabel, xlim, ylim, title, grid, axes, show, legend, suptitle.

6.3.4 random

The random module generates random numbers. Currently, it uses the Mersenne Twister PRING. The only function that are important are getrandbits, random, shuffle, seed and randint.

6.3.5 fractions

Fraction module supports rational number arithmetic. Using this module, we can create fractions from integers, floats, decimal and from some other numeric values and strings. There is a concept of Fraction Instance. It is formed by a pair of integers as numerator and denominator.

The class fractions.Fractionis used to create a Fraction object. It takes Numerator and Denominator. The default value of the numerator is 0 and denominator is 1. It raises ZeroDivisionError when the denominator is 0.

6.4 Global Program Variables

6.4.1 Nyquist Rate

In signal processing, the Nyquist rate, is twice the bandwidth of a band limited function or a band limited channel. This term means two different things under two different circumstances:

1. as a lower bound for the sample rate for alias-free signal sampling (not to be confused with the Nyquist frequency, which is half the sampling rate of a discrete-time system) and
2. as an upper bound for the symbol rate across a bandwidth-limited baseband channel such as a telegraph line or passband channel such as a limited radio frequency band or a frequency division multiplex channel.

6.4.2 Kaiser Window and Beta

The Kaiser window is an approximation to the prolate spheroidal window, for which the ratio of the main lobe energy to the side lobe energy is maximized.

The β parameter of the Kaiser window provides a convenient continuous control over the fundamental window trade-off between side-lobe level and main-lobe width. Larger β values give lower side-lobe levels, but at the price of a wider main lobe. Widening the main lobe reduces frequency resolution when the window is used for spectrum analysis. Reducing the side lobes reduces “channel cross talk'” in an FFT-based filter-bank implementation.

6.4.3 Firwin

FIR filter design using the window method. This function computes the coefficients of a finite impulse response filter. The filter will have linear phase; it will be Type I if numtaps is odd and Type II if numtaps is even.

Type II filters always have zero response at the Nyquist frequency, so a Value Error exception is raised if firwin is called with numtaps even and having a pass band whose right end is at the Nyquist frequency.

6.4.4 Lfilter

Filter data along one-dimension with an IIR or FIR filter. Filter a data sequence, *x*, using a digital filter. This works for many fundamental data types (including Object type). The filter is a direct form II transposed implementation of the standard difference equation.

6.5 Implementation Of Python Program:

- Check the system requirements and install the required software.
- Install the required libraries.
- Set up the project environment to facilitate the development of this project.
- Download the audio samples to verify the working of the filters.
- Import the required input sample into the program and add random noise to it. This signal is X, which is the noisy signal.
- Setup global variables, such as Sampling frequency, Nyquist rate, BandPass cutoff is 20Hz to 20kHz.

6.5.1 Create FIR Filter

- Choose the appropriate type of window for Band Pass FIR filter.
- Get the implementation of the FIR filter by calling the appropriate function with the window choosen. Input the global variables to get the order of the filter and any other parameters that are necessary.
- Preserve this implementation for filter of the noisy signal X

6.5.2 Create IIR Filter

- Choose the appropriate type of Band Pass IIR filter.
- Get the implementation of the IIR filter by calling the appropriate function.Input the global variables to get the order of the filter and any other parameters that are necessary.
- Preserve this implementation for filter of the noisy signal X.

6.5.3 Create Multirate Filter

- Create the Low Pass FIR filter implementation for the Multirate System, taking cutoff frequency to be 20kHz. Refer for the steps in 3.1.5.1 .
- Design the function to implement decimation, interpolation and resampling.
- Create the High Pass FIR filter implementation for the Multirate System, taking cutoff frequency to be 20Hz. Refer for the steps in 3.1.5.1 .
- Preserve this implementation for later use.

6.5.4 Create cascaded Multirate Filter

- Select the number of stages to be implemented (N).
- Decide the band of frequency which each stage should filter.
- Using the above implementaion of the Multirate filter, set the coefficients required.
- After the completion of each stage, add up the outputs of all the filters.
- Preserve this implementation for filter of the noisy signal X.

In summary, in this chapter the various software requirements, programming language, the libraries required and other global variables are discussed. Along with these, the implementation of the project and the implementation of the filters are also discussed.

CHAPTER 7

ALGORITHM & FLOW CHART

This chapter explains the actual flow of the program by means of tools like algorithms and flowcharts. Both algorithm and flowchart for this project can be used to implement the project in any programming language.

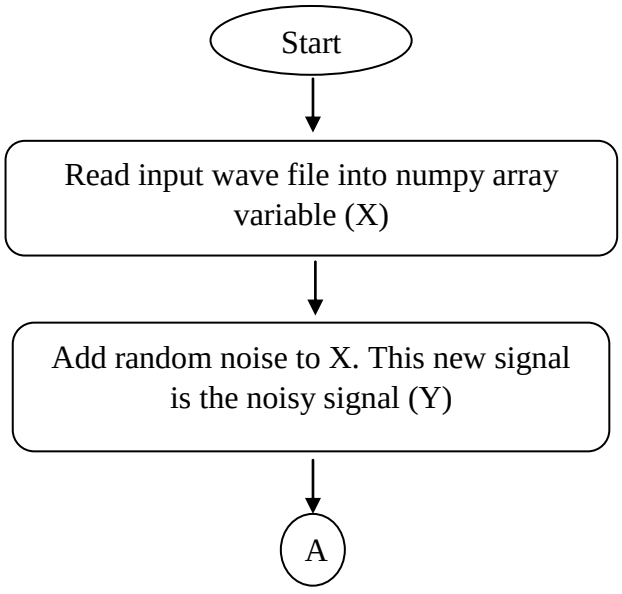
7.1 Algorithm

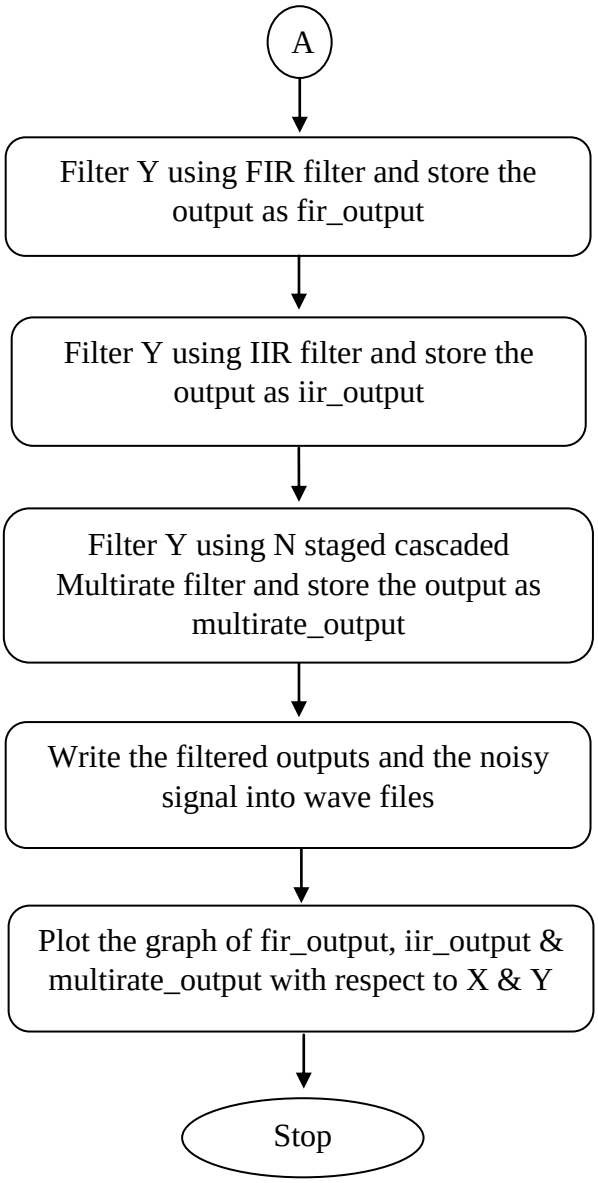
An algorithm is a single, specific way of performing a complex task efficiently and repeatedly. The algorithm works by performing all of the smaller steps you need to do to perform the complex task, and is guaranteed to work

Below is the algorithm of the program:

- Start.
- Get the required global variables, parameters and filter implementations.
- Filter the noisy signal X using FIR filter implementation and get the output.
- Filter the noisy signal X using IIR filter implementation and get the output.
- Filter the noisy signal X using Multirate filter implementation and get the output.
- Write the noisy signal X and the filtered outputs to audio files.
- Plot the required graphs using the filtered outputs.
- End.

7.2 Flow Chart





The above algorithm and flowchart for implementing the project in any required programming language. In this implementation, the project has been developed in the Python programming environment. Hence, we can conclude this chapter by saying that usage of the above algorithm and flowchart can be helpful to realize this project in any programming language.

CHAPTER 8

RESULT AND DISCUSSION

In the previous chapter algorithm and flowchart for the project was designed. This chapter provides the steps for running the program and puts emphasis on the results obtained after executing the program. Further, the results of the project are also discussed.

The steps involved for executing the program are as follows:

1) Run source venv/bin/activate whilst in the project directory using a terminal for Linux OS. This step can be skipped in Windows OS.
2) Use python3 saigo.py to run the script.
3) Enter the name of the wav file from the samples in wav/ directory. If no name is given, default one is assumed.
4) The relevant graphs will then show up.
5) Along with the graphs, the filtered output of all the filters are exported to individual wav files and can be played back for checking the performance of the filters.

The results obtained after implementing the algorithm of the filters are discussed below:

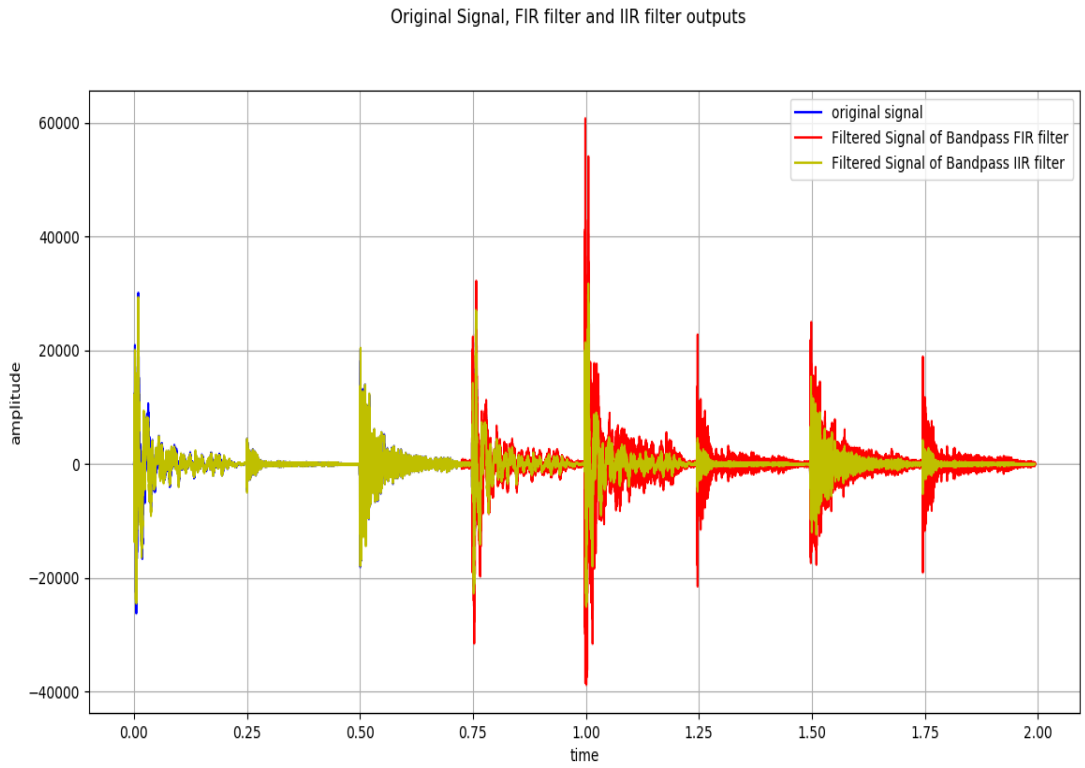


Fig 8.1: Plot of Original signal,FIR Filter and IIR Filter outputs

As seen in the Fig 8.1, the performance of bandpass IIR filter (yellow) for the same input noisy signal is better than that of bandpass FIR filter (red). Note that the IIR filter's waveform closely follows that of the original signal's waveform (dark blue). In Fig 8.1, note the waveform of FIR filter. There are a lot of random spikes, which are due to the random noise that is added to the original signal.

Hence, we can infer that the performance of IIR filter is better than FIR filter, in audio processing.

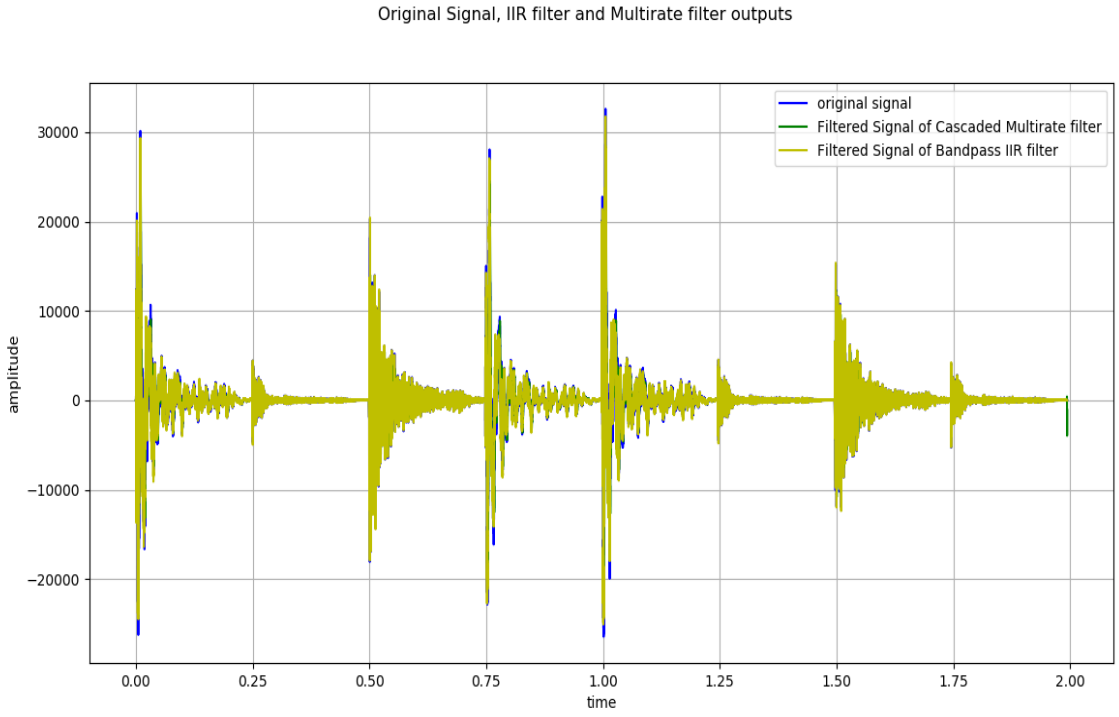


Fig 8.2: Plot of Original Signal, Bandpass IIR Filter and Multirate filter

In Fig 8.2, the performance of Bandpass IIR (yellow) and Cascaded Multirate filter (green) is shown. Note from the figure, the waveform of the IIR filter closely follows that of the original signal. However, the gain of the IIR filter is slightly less than that of the original signal. This is because of the fact that the noise is reduced to some extent when passed through the IIR filter. But the noise is still noticeable when the IIR filtered output is heard.

When compared to both the IIR filter and the original signal, the waveform of Cascaded Multirate Filter output is more smoother. The Multirate filter reduces the noise greatly because of the fact that each stage is focused on a particular sub-band, therefore the performance is better. Hence the smoothening of the waveform is observed in the case of Cascaded Multirate Filter.

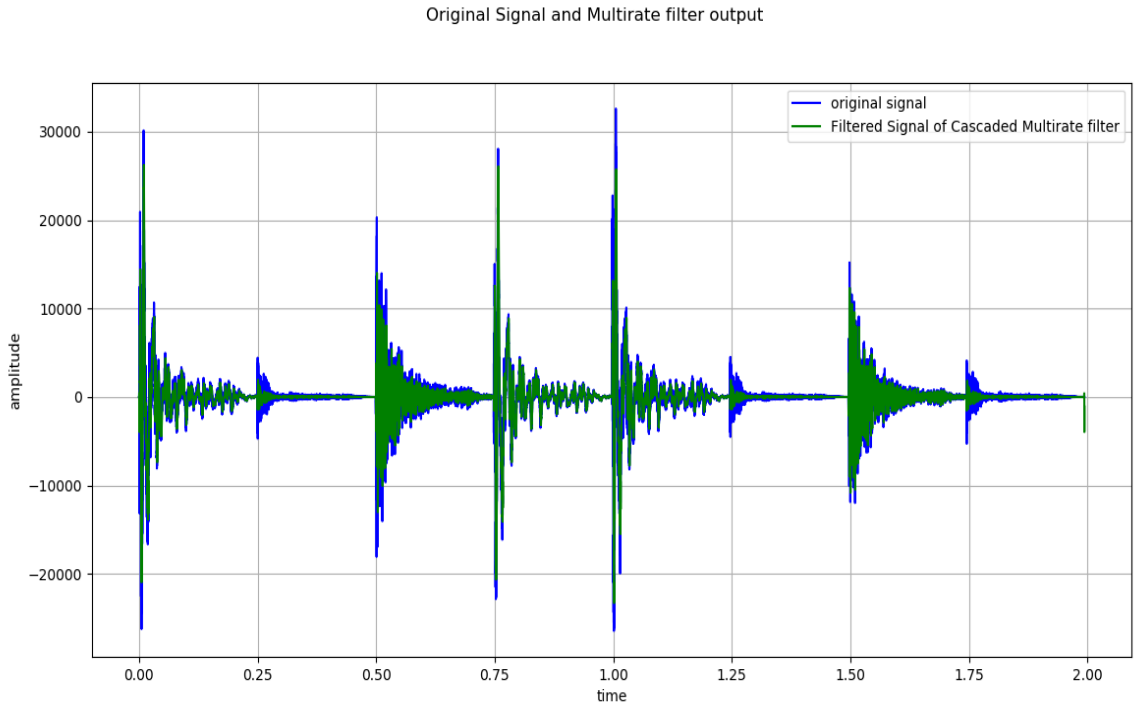


Fig 8.3: Plot of Original Signal and Multirate Filter

In Fig 8.3, the plot of waveforms of Original signal (dark blue) and that of the Cascaded Multirate filter output (green) are plotted. Note that the Cascaded Multirate filter's waveform appears to be smaller than Original signal. This is because of the fact that the Original signal may itself contain noise which was present during recording.

The performance can be more efficiently determined by the audio files of the filter outputs. The filters are designed to be operated in the 20 Hz to 20 kHz, hence the difference can be more easily identified upon listening to the audio files.

We can summarize this chapter by saying that for audio processing, the performance of IIR filters is better than that of FIR filters. However, the filtered output still contains noticeable degradation of signal quality. The waveform of the Cascaded Multirate filter output is much more smoother than that of IIR filter. Not only the added noise, but the noise that was recorded was also attenuated to a greater extent in the output audio samples. Hence we can conclude that the performance of Cascaded Multirate Filter is better than traditional filters, such as FIR and IIR filters.

CHAPTER 9

ADVANTAGES, DISADVANTAGES AND APPLICATIONS

This chapter deals with the advantages, disadvantages and applications of this project.

9.1 Advantages

Some of the advantages are:

- With interpolation and decimation, the computational and/or memory requirements of the resampling filtering can sometimes be greatly reduced by using multiple stages.
- Sampling rate conversion of a digital signal can be accomplished in two methods. One method is to pass the digital signal through a D/A converter, filter it and then resample the resulting analog signal at the desired rate. The second method is to perform the sampling rate conversion entirely in the digital domain. The advantage of the first method is that the new sampling rate can be arbitrarily selected and needn't have any special relationship to the old sampling rate.
- Multirate Digital signal processing is more efficient, distortion less and flexible type of signal processing.

9.2 Disadvantages

Some of the disadvantages are:

- This system is more complex design, has the phenomenon of aliasing.
- It has imaging errors.
- The algorithm is more complex than traditional filters.

9.3 Applications

- Used for the design of phase shifters.
- Interfacing of digital systems with different sampling rates .
- Subband coding of speech signals.
- Quadrature mirror filters (QMF).
- Oversampling A/D and D/A conversion.

This chapter summarizes that the Cascaded Multirate filters have more advantages, fewer disadvantages and more applications when compared to traditional filters.

CHAPTER 10

CONCLUSION AND FUTURE SCOPE

10.1 Conclusion

In short the performance of Cascaded Multirate filter is better than other traditional filters. This point is explained in more detail as follows:

- FIR filters are more powerful than IIR filters, but also require more processing power and more work to set up the filters. They are also less easy to change "on the fly" as you can by tweaking (say) the frequency setting of a parametric (IIR) filter.
- Their greater power means more flexibility and ability to finely adjust the response of your active loudspeaker. Since the nature and type of noises is unpredictable, it is seen that the more general purpose IIR filters perform better.
- However, the performance of Cascaded Multirate Filter Bank, which has been the focus of this project. This is due to the fact that each stage of the filter is focused to filter only a specific band of the filter, thereby the performance of the entire filter increases as the number of filters increases.

Hence in conclusion, we can say that Cascaded Multirate Filter Bank is better than FIR or IIR filters.

10.2 Future Scope

Future scope defines the next direction that the project should head into and the additional steps to be taken to improve upon the work done in this project.

The future scopes of this project are as follows:

- Further work needs to be done to optimize the source code to make better use of computer resources.
- After optimization, the code can be bundled into a library or a module, so that the filter functions can be easily used by other programmers.
- The code should be made flexible enough to incorporate the advances in Multirate signal processing concepts

We can therefore summarize the future scope that additional work needs to be done on this code so that it can be implemented in real world situations.