B.E. PROJECT ON

Designing of Finite Impulse Response Filter using Artificial Neural Network

Submitted by

Himanshu Singh	072/EC/15
Ishan Dua	074/EC/15
Mohit Yadav	103/EC/15
Pranav Khurana	127/EC/15

Under the Guidance of
Dr. Tarun Rawat
Mr. Abhay Sharma

A Project in partial fulfillment of requirement for the award of B.E. in
Electronics & Communication Engineering

Department of Electronics & Communication Engineering, (NSIT)
Now upgraded to NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY
NEW DELHI-110078

2019

CERTIFICATE

This is to certify that the report entitled "**Designing of Finite Impulse Response Filter using Artificial Neural Network**" being submitted by Himanshu Singh, Ishan Dua, Mohit Yadav and Pranav Khurana to the Department of Electronics and Communication Engineering, NSIT, (now upgraded to Netaji Subhas University of Technology), for the award of bachelor's degree of engineering, is the record of the bonafide work carried out by them under our supervision and guidance. The results contained in this report have not been submitted either in part or in full to any other university or institute for the award of any degree or diploma.

Supervisors

Dr. Tarun Rawat
Department of ECE
NSIT, New Delhi, India
(now upgraded to NSUT)

Mr. Abhay Sharma
Department of ECE
NSIT, New Delhi, India
(now upgraded to NSUT)

ACKNOWLEDGEMENT

We wish to convey our profound gratefulness to Dr. Tarun Kumar Rawat , Department of Electronics and Communication Engineering, Netaji Subhas university of Technology, Delhi for giving us the opportunity to work on this project. Without their thoughtful encouragement, careful supervision and unfailing support, this thesis would have never taken shape. The skills and knowledge which we have gained through the project form a valuable component in our future career development. We would like to express our special gratitude towards our seniors, for their kind cooperation and encouragement which helped us in the completion of this project. We are grateful to our parents for supporting us through the course of this project. We extend our thanks and appreciation to the staff members and people who helped us during the course of the project.

HIMANSHU SINGH (72EC15)
ISHAN DUA (74EC15)
MOHIT YADAV (103EC15)
PRANAV KHURANA (127EC15)

Bachelor of Engineering
Division of Electronics & Communication Engineering
Netaji Subhas University of Technology

Abstract

Digital filters are usually used to eliminate the additional noise in signal which can affect negatively on the required portion. In many cases as in telephone systems, there is no need to transmit very high frequencies since most speech falls within the band of 400 to 3400 Hz. Filtering out or rejecting all the frequencies above and below the desired band is mandatory in most Digital Signal Processing (DSP) systems. Digital filters fall into two types; Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). Linearity and stability properties of FIR give the reason of using this type in most communication systems. Artificial neural networks (ANN) are computing systems vaguely inspired by the biological neural networks that constitute animal brains.Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. A new method for the calculation of coefficients of an FIR filter has been introduced which involves the use of Neural Networks. The initial objective of the project which was to find an alternative to computation intensive designing methods like windowing has been achieved as the proposed method uses multiple iterations and back propagation of errors to refine the initialised values of the required vector(vector of filter coefficients in our application) in order to make them as ideal as possible.

Contents

List of Figures

List of Tables

1 Introduction

Digital filters are usually used to eliminate the additional noise in signal which can affect negatively on the required portion. In many cases as in telephone systems, there is no need to transmit very high frequencies since most speech falls within the band of 400 to 3400 Hz. Filtering out or rejecting all the frequencies above and below the desired band is mandatory in most Digital Signal Processing (DSP) systems. Digital filters fall into two types; Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). Linearity and stability properties of FIR give the reason of using this type in most communication systems. Based on the form of frequency transfer function, the FIR digital filter can be classified into the following types: low-pass, high-pass, band-pass and band-stop filter. Output of FIR filter is given by :

$$\begin{aligned} y[n] &= b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N] \\ &= \sum_{i=0}^{N} b_i \cdot x[n-i] \end{aligned} \tag{1}$$

where, N is the order of the filter, x[n] is the input of the signal and b represent the coefficients of the FIR filter. The most famous approach in FIR filter design, before Artificial Neural Networks (ANNs) was the windowing method. Many different types of window had been used in the FIR implementation process such as Kaiser, Hamming, Hanning, rectangular etc. The concept of the window method is based on transforming an ideal duration of infinite impulse response to a finite duration impulse response filter design. All the methods for FIR design that used in the DSP field are approximated approaches, so they do not give optimal results in the design of the FIR, therefore, the new researches proceed to use the ANN approach to solve this problem.

Artificial neural networks (ANN) are computing systems motivated by the biological neural networks that constitute animal brains. The neural network itself is not an algorithm, but preferably a framework for many different machine learning algorithms to operate together and process complex data inputs. Such systems master to perform tasks by analysing examples, generally without being processed with any task-specific rules.

1.1 FIR Filters

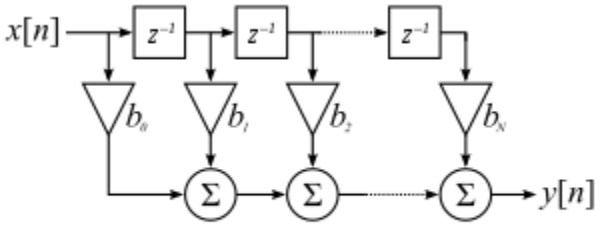
In signal processing, a Finite Impulse Response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying).

The impulse response of an Nth-order discrete-time FIR filter lasts exactly N + 1 samples (from first nonzero element through last nonzero element) before it then settles to zero. For a causal discrete-time FIR filter of order N, each value of the output sequence is a weighted sum of the most recent input values:

$$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] \ldots\ldots\ldots b_Nx[n-N] \quad (2)$$

where,
$x[n]$ is the input signal
$y[n]$ is the output signal
N is the filter order
bi are filter coefficients

The$x[n-i]$ in these terms are commonly referred to as taps, based on the structure of a tapped delay line that in many implementations or block diagrams provides the delayed inputs to the multiplication operations.



Direct form structure of an Nth order FIR filter

Figure 1: Direct form structure

1.1.1 Properties of FIR filters -

An FIR filter has a number of useful properties which sometimes make it preferable to an infinite impulse response (IIR) filter. FIR filters:

● Require no feedback. This means that any rounding errors are not compounded by summed iterations. The same relative error occurs in each calculation. This also makes implementation simpler.

● Are inherently stable, since the output is a sum of a finite number of finite multiples of the input values.

● Can easily be designed to be linear phase by making the coefficient sequence symmetric.

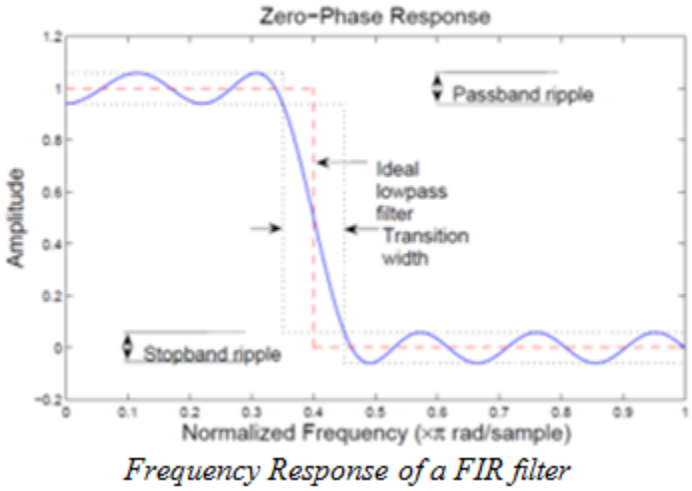


Figure 2: Frequency Response of a FIR filter

1.1.2 Design of FIR filters -

An FIR filter is designed by finding the coefficients and filter order that meet certain specifications, which can be in the time domain and/or the frequency domain. One of the most common methods to design an FIR filter is the Windowing method. In the window design method, one first designs an ideal IIR filter and then truncates the infinite impulse response by multiplying it with a finite length window function. The result is a finite impulse response filter whose frequency response is modified from that of the IIR filter. Multiplying the infinite impulse by the window function in the time domain results in the frequency response of the IIR being convolved with the Fourier transform (or DTFT) of the window function. The ideal response is usually rectangular, and the corresponding IIR is a sinc function. The result of the frequency domain convolution is that the edges of the rectangle are tapered, and ripples appear in the passband and stopband.

1.2 Neural Networks.

Neural Networks or more specifically Artificial neural networks (ANN) or connectionist systems are computing systems inspired by the biological neural networks that constitute animal brains.The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules. An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In common ANN implementations, the signal at a connection between artificial neurons is a real number, and the output of each artificial neuron is computed by some non-linear function of the sum of its inputs. The connections between artificial neurons are called 'edges'. Artificial neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Artificial neurons may have a threshold such that the signal is only sent if the aggregate signal crosses that threshold. Typically, artificial neurons are aggregated into layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times

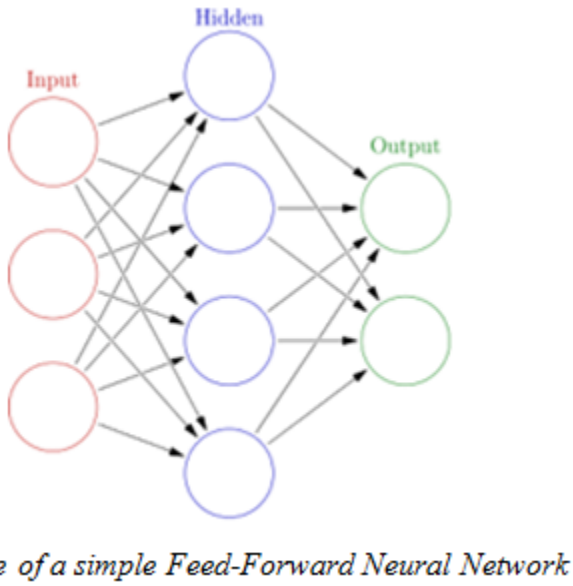


Figure 3: Structure of a simple Feed-Forward Neural Network

2 Digital Signal Processing

2.1 Introduction

Since the Second World War technicians have considered on the applicability of digital techniques to perform signal processing tasks, for instance, at the end of the 1940s, Shannon, Bode and other researchers at the Bell Telephone Laboratories studied the chance of using digital circuit elements to produce filter functions. At this time, there was, regrettably, no proper hardware available. Hence, cost, size and reliability strongly preferred current — analog implementations. While the middle of the 1950s, Professor Linville at MIT discussed digital filtering at graduate seminars. By then, control theory, based somewhat on works by Hurewicz had become established as a discipline, and sampling and its spectral effects were well appreciated. Several mathematical tools, such as the z-transform, which had existed since Laplace's time, were now used in the electronics engineering community. Technology at that point, nevertheless, was only able to deal with low-frequency control problems or low-frequency signal processing problems. While seismic scientists made fundamental use of digital filter concepts to solve problems, it was not until the midst of the 1960s that a more precise theory of digital signal processing commenced to emerge. During this period, the arrival of the silicon integrated circuit technology made whole digital systems possible, but still quite costly. Kaiser produced the first significant contribution in the area of digital filter synthesis at Bell Laboratories. His work showed how to design useful filters using the bilinear transform. Further,around 1965, the famous paper by Cooley and Turkey was published. In this paper, FFT (fast Fourier transform), an efficient and fast method of performing the DFT (discrete Fourier transform), was explained, At this time, hardware better suited for realising digital filters was developed and affordable circuits began to be commercially available.

2.2 FIR filter

2.2.1 Characteristics Of Fir Filters

- The impulse response of an FIR, if given symmetric or anti-symmetric, will generate a linear phase function, the result implying that signals passing through the filter do not have their phases dispersed.
- Arbitrarily steep roll-offs may be had by making the impulse response correspondingly long. The cost is in computation, since linear convolution in the time domain of two sequences of length N requires about N2 multiplications.This can often be alleviated by the use of frequency domain techniques.
- Arbitrary pass characteristics (i.e., other than standard lowpass, highpass, notch, or bandpass) may readily be generated.
- FIRs are inherently stable, meaning that a finite-valued input signal will never lead to an unbounded output signal.

2.2.2 Effect of filter length

Let N be thelength of a simple lowpass filter having impulse response [ones(1, N)]. By increasing N, the number of samples in the filter, we can observe the effect on frequency response, especially on steepness of roll-off. Figure 1.1 shows three such filters of increasing length in subplots (a), (c), and (e), and their frequency responses in plots (b), (d), and (f), respectively. From Fig. 1.1 we can readily deduce that a filter can be made more selective or have a steeper roll-off by increasing its length. This may be explained by noting that as the filter length is increased, there are more distinct integral-valued frequencies (or correlators) between 0 and π radians (normalized frequencies of 0 and 1.0).

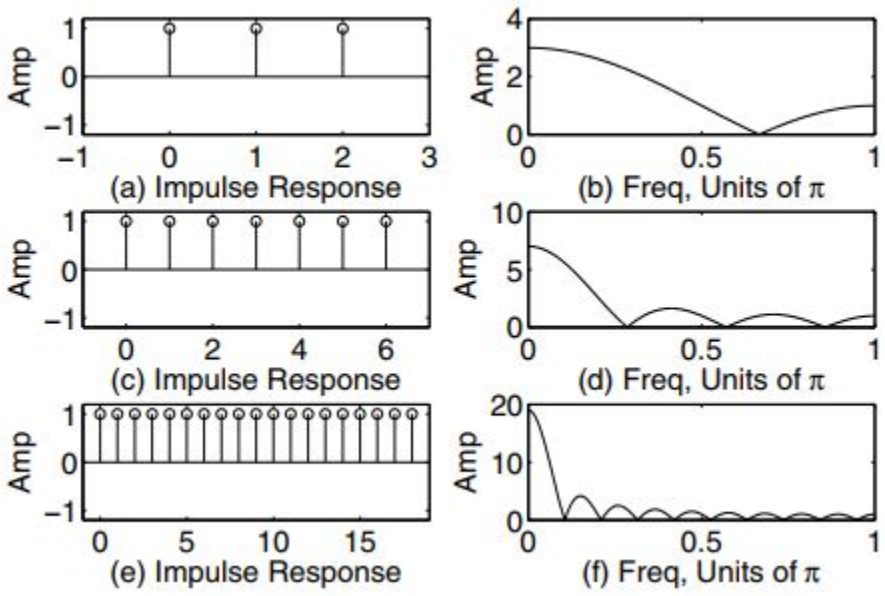


Figure 4: (a) 3-pt Rectangular Impulse Response; (b) Magnitude of frequency response of signal at (a); (c) 7-pt Rectangular Impulse Response; (d) Magnitude of frequency response of signal at (c); (d) 19-pt Rectangular Impulse Response; (e) Magnitude of frequency response of signal at (e).

2.2.3 Effect of windowing

It can be seen in Fig. 1 that sustained impulse responses, although more frequency selective, suffer from scalloping in the frequency response. Fortunately, this can be mitigated by using the identical kind of windows on the filter impulse response that we used on signals prior to taking the DFT. Figure 1.2 shows the same three impulse responses after smoothing with a hamming window, and the corresponding frequency responses. The improvement is dramatic.

In Fig. 1.3, plot (a), a multi-cycle cosine (essentially rectangularly-windowed) has been employed as the impulse response. As can be seen in plot (c), the expected

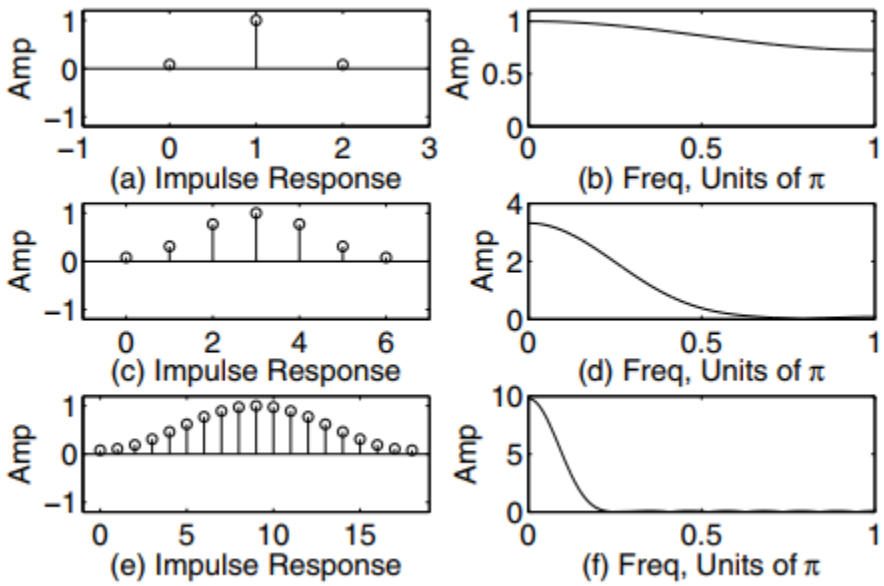


Figure 5: (a) 3-pt Hamming-windowed Impulse Response; (b) Magnitude of frequency response of signal at (a); (c) 7-pt Hamming-windowed Impulse Response; (d) Magnitude of frequency response of signal at (c); (d) 19-pt Hamming-windowed Impulse Response; (e) Magnitude of frequency response of signal at (e).

scalloped response results. Figure 1.3, plot (d), shows the result when a hamming window is applied to the impulse response (plot (b)). Note that the steepness of roll-off of the main lobe of the response is decreased with application of the hamming window, opposed to the rectangular window. You should discern that the rectangular window result has a main lobe width narrower than that of the hamming window example, as can be readily seen, but the rectangular window's side lobe amplitude is very distinguished (i.e., the stopband attenuation is poor), producing the rectangular window unacceptable for most applications.

General Rule: For a given filter length, the vaster the stopband attenuation, the smaller the main lobe roll-off must be. Said conversely, for a given filter length, the steeper the main (or central) lobe roll-off, the poorer will be the final stopband attenuation. For a presented stopband attenuation, the roll-off may be enhanced by increasing the filter length

2.2.4 Linear phase

- FIRs can be adjusted to have a Linear Phase response, which means that a diagram of the phase shift presented by the filter versus frequency is a straight line, or at least piece-wise linear.
- With linear phase shift, all frequencies remain in phase, and consequently the filter acts like a bulk delay line in which all frequencies reside in time alignment. Such a filter is stated to be Non-Dispersive.

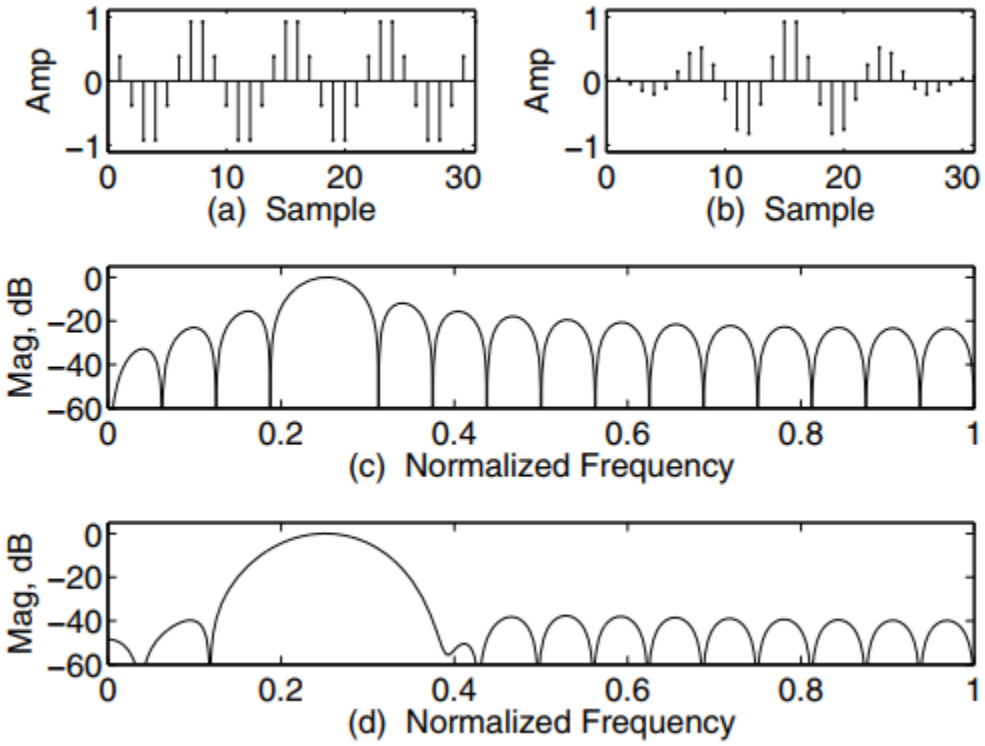


Figure 6: (a) Impulse response, a cosine, inherently windowed with a Rectangular window; (b) Same cosine as (a), multiplied by a Hamming window; (c) Magnitude of frequency response in dB of sequence at (a); (d) Magnitude of frequency response in dB of sequence at (b)

To see why a mere delay line has a linear phase delay characteristic (and vice versa), assume a signal simply going through a delay line that grants a delay time of τ. A given frequency f0 has a period of 1/f0 and is therefore delayed by $\tau/(1/\text{f0}) = \text{f0} \cdot \tau$ cycles, which is simply a linear function of frequency If f0 doubles, for illustration, so does the number of cycles of delay, and so on.

2.2.5 Impulse response requirement

Impulse response coefficients be either symmetrical or anti-symmetrical about the centre of the impulse response.

2.2.6 Basic types of FIR filter

Linear phase impulse responses must be either symmetric or anti-symmetric, and any impulse response must have either an even length (evenly divisible by two) or an odd length.There are distinct differences between even and odd length filters, as well as symmetric and anti-symmetric filters.

Type I, Symmetric, Odd Length This is perhaps the most used of linear phase FIR filter types because it is suitable for lowpass, highpass, bandpass, and bandstop filters. Correlator basis functions are cosines, meaning that the impulse response is generated by weighting and summing cosines of different frequencies.

Type II: Symmetric, Even Length This filter type cannot be used as a highpass or bandstop filter since the cosine, at the Nyquist limit (necessary in the impulse response for highpass or bandstop filters), cannot be symmetrical in an even length. Lowpass and bandpass filters are possible.

Type III: Anti-Symmetric, Odd Length The correlator basis functions aresines.The basis functions are identically zero at DC and theNyquist limit, so this type of filter cannot be used for lowpass or highpass characteristics. Bandpass filters are possible, but the primary use is in designing Hilbert transformers and differentiators. Hilbert transformers shift the phase of all frequencies in a signal by $90\circ$, or $\pi/2$ radians. This is useful in certain communications applications, such as generating single sideband signals, demodulating quadrature modulated signals, and so forth.

Type IV: Anti-Symmetric, Even Length Like the Type III filter, this filter uses sines as the basis correlating functions. It is not suitable for a lowpass characteristic, but is suitable for Hilbert transformers and differentiators.

2.2.7 Zero location in Linear phase FIR filters

A linear phase FIR's zeros adhere to certain conditions. Zeros having magnitude other than 1.0 (i.e., not on the unit circle) need to always be balanced with a zero having the same frequency but the reciprocal magnitude. Any complex zero, of course, must also be matched with its complex conjugate to ensure real-only coefficients.

2.2.8 Linear Phase FIR frequency content and response

Since linear phase FIRs adhere to specific forms for the impulse response, the frequency response can be formulated for each of the four FIR types as an expression involving cosines or sines.The Impulse Response for Type I filters adheres to the rule that

$$h[n] = h[L - n - 1] \tag{3}$$

where the impulse response length is L and n = 0:1:L − 1.
The Frequency Response for Type I and II filters adheres to the general form

$$H(\omega) = H_r(\omega)e^{-j\omega M} \tag{4}$$

where M = (L − 1)/2, and H r(ω) is a real function that can be positive or negative and is accordingly called the Amplitude Response, while the complex exponential

describes a linear phase factor. For the Type I filter (odd length), the amplitude response is given as

$$H_r(\omega) = h[M] + 2\sum_{n=0}^{M-1} h[n]\cos(\omega[M-n]) \tag{5}$$

which is equivalent to

$$H_r(\omega) = h[M] + 2\sum_{n=0}^{M-1} h[n]\cos(\omega[M-n]) \tag{6}$$

This can be written as

$$H_r(\omega) = 2\sum_{n=0}^{L/2-1} h[n]\cos(\omega[n]) \tag{7}$$

2.2.9 Design methods for FIR filters

- Window Method: In this technique, a truncated ideal lowpass channel having a particular transmission capacity is presented, and after that a selected window is connected to perform a certain stopband weakening. Channel length can be adjusted to achieve a required move off rate in the change band. Channel types other than lowpass, for example, highpass, bandpass can be accomplished by a few methods that begin with windowed, truncated perfect lowpass channels.

- Frequency Sampling Method: In this technique, equitably distributed examples of an ideal frequency response are made, and the IDFT is processed to get a impulse response. In expansion to utilizing the IDFT to achieve this, there are equations that do the proportional, i.e., superposing symmetrical cosines or sines to create a drive reaction. The two variations of this strategy will be investigated underneath. Instead of making a drive reaction to be executed either in Direct Form or Linear Phase Form, it is imaginable to utilize the recurrence tests to legitimately appreciate the channel utilizing this current strategy's namesake, the Frequency Sampling Realization Strategy, which will be investigated in the activities toward the finish of the section.

- Equiripple Method: This strategy plans a FIR having evened out desirable amplitudes in the passband, and evened out desirable in the stopband. The desirable dimensions might be autonomously controlled, permitting incredible adaptability. It is understandable with equiripple plan, for instance, for a given channel length, to increment stopband reducing by letting passband swell increment, and the other way around. The past two FIR plan techniques don't allow this level of control.

2.2.10 Filter Specification

• Figure 4 illustrates a typical design specification for an FIR lowpass filter. There is generally a particular amount of ripple in both passbands and stopbands. In Fig. 4, the highest deviation (recognised as an acceptable tolerance) from the average value in the passband is designated δ_P , while the variation from zero in the stopband is designated as δ_S. This manner of defining the conditions for passband and stopband ripple is called an absolute specification; another manner (more common) is to define the levels of ripple in decibels relative to the maximum magnitude of response, which is $1 + \delta_P$.

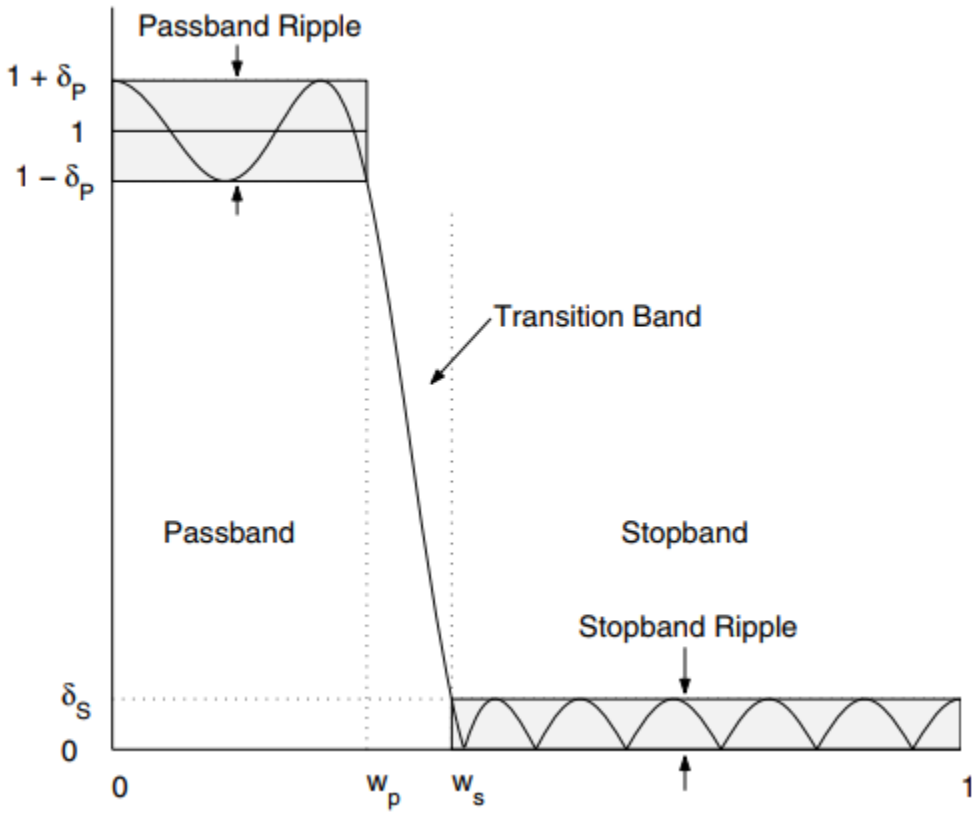


Figure 7: Design criteria for an FIR filter

Figure 4 represents another lowpass filter design specification in relative terms. The values of R and A are in decibels, and express the passband ripple amplitude (or minimum passband response when the maximum filter/passband response is 0 db) and minimum stopband attenuation, respectively. The correlation between δ_P and δ_S and R and A are

$$R = -20 \log 10 \left(\frac{1 - \delta_P}{1 + \delta_P} \right) \tag{8}$$

$$A = -20 \log 10 \left(\frac{\delta_S}{1 + \delta_P} \right) \tag{9}$$

To determine δ_P when R and A are given, use

$$\delta_P = \frac{1 - 10^{-R/20}}{1 + 10^{-R/20}} \tag{10}$$

$$\delta_S = (1 + \delta_P)\, 10^{-A/20} \tag{11}$$

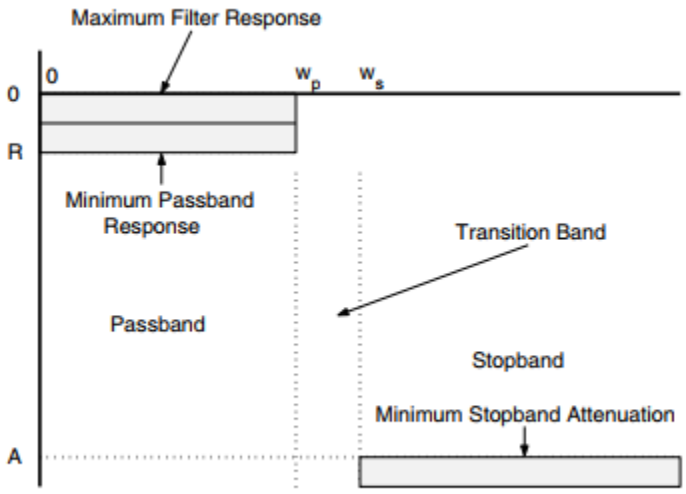


Figure 8: A relative filter design specification, with the (horizontal) frequency axis at the top, and (vertical) logarithmic amplitude (dB) axis at the left

2.2.11 Windows

Any finite-length (i.e., truncated) variant of the ideal lowpass impulse response may be considered as the product of the infinite-length lowpass impulse response and a window function W, which has a limited number of contiguous nonzero-valued samples

$$b = \left(\frac{\sin\left(\omega_c [n - M]\right)}{\pi [n - M]} \right) \left(W_L [n - M] \right) \tag{12}$$

where the window length is L, M = (L − 1)/2, 0 ≤ n ≤ L − 1, and W_L[n] is generally a function F_E[n] having even symmetry about M defined as

$$W_L[n] = \begin{cases} F_E[n] & n = 0:1:L-1 \\ 0 & \text{otherwise} \end{cases}$$

The right side of Eq. (2.2) is the product of an infinite-length, ideal lowpass filter with a function that is nonzero only above a finite number of contiguous samples. The effect is a finitelength or trimmed lowpass filter. Since W is chosen to have even symmetry around M, the window symmetrically trims the infinite-length function (which is itself symmetrical about M), resulting in a finite-length symmetrical sequence as the net filter impulse response. We found windows as a tool for reducing leakage in the DFT. We address some of the basic knowledge on standard windows here for accessibility. The simplistic window is the rectangular window R[n], which is defined as

$$R[n] = \begin{cases} 1 & 0 \le n \le L-1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

The windowing process, utilising a rectangular window, is shown in Fig. 2.3. In subplot (a), a part of an infinite-length, ideal lowpass filter is shown, centered on M; in subplot (b), a part of the infinite-length (rectangular) window is shown, with its contiguous, nonzero-valued samples centered on M, and eventually, in subplot (c), the product of the two sequences in (a) and (b), the trimmed lowpass filter, is shown.

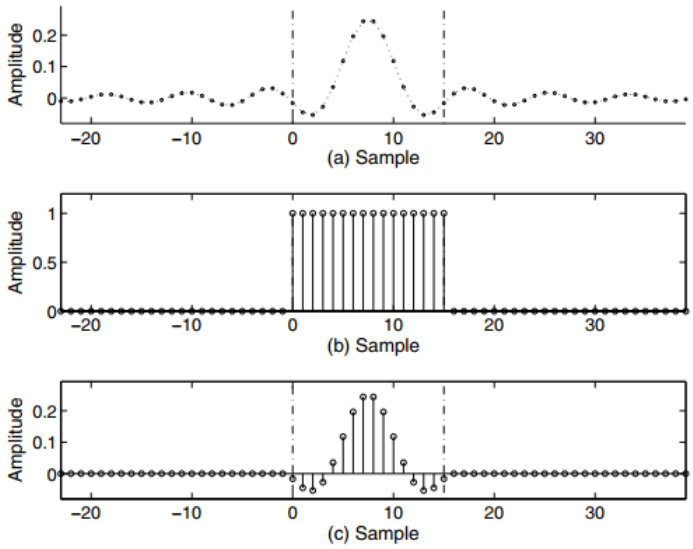


Figure 9: (a) Samples of an Ideal Lowpass filter impulse response, delayed by M = (L − 1)/2, where L is the desired digital filter length; (b) A Rectangular window of length L; (c) The product of the window and the Ideal Lowpass filter, yielding, for samples n = 0:1:(L − 1), a length-L symmetrical impulse response as the lowpass digital filter finite impulse response.

The Hanning, Hamming, and Blackman windows are described by the general raised-cosine formula:

$$W[n] = \begin{cases} a - b\cos\left(2\pi\frac{n}{L-1}\right) + c\cos\left(4\pi\frac{n}{L-1}\right) & n = 0:1:L-1 \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

where L is the window length, and where, for the Hanning window, a = 0.5, b = 0.5, and c = 0; for the Hamming window, a = 0.54, b = 0.46, and c = 0; and for the Blackman window, a = 0.42, b = 0.5, and c = 0.08.

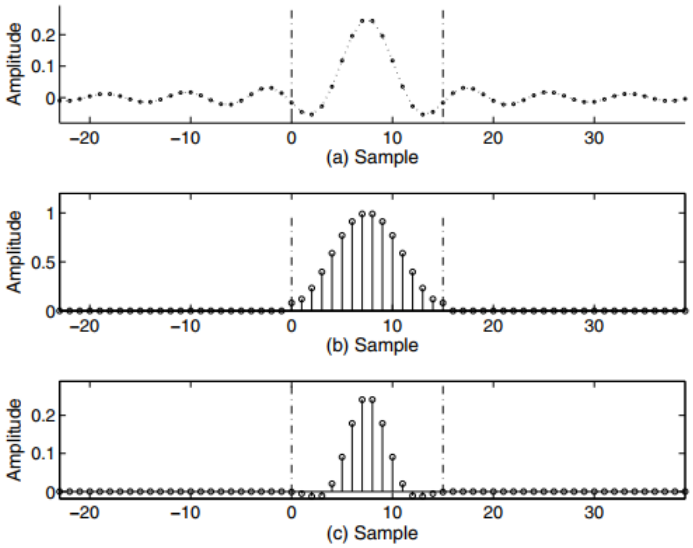


Figure 10: (a) Samples of an Ideal Lowpass filter impulse response, delayed by M = (L − 1)/2, where L is the desired digital filter length;(b) A Hamming window of length L;(c)The product of the window and the Ideal Lowpass filter, yielding, for samples n = 0:1:(L − 1), a length-L symmetrical impulse response as the lowpass digital filter finite impulse response

The Kaiser window is described by the formula

$$W[n] = \begin{cases} \frac{I_0\left[\beta\left(1-(n-M)/M]^2\right)^{0.5}\right]}{I_0(\beta)} & n = 0:1:L-1 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

for n = 0:1:L-1, where L is the window length, M = (L − 1)/2, and I0 represents the modified Bessel function of the first kind.

2.2.12 Implication of various windows on responses

Table 1: Implications of various Windows on frequency response.

S. no.	Name of window	Implication
1	Rectangular	finest main lobe and scantiest transition width, but the worst stopband attenuation.
2	Hanning	It has extensive main lobes, extensive transition widths, but enhanced stopband attenuation.
3	Hamming	It has extensive main lobes, extensive transition widths, but enhanced stopband attenuation.
4	Blackmann	It has extensive main lobes, extensive transition widths, but enhanced stopband attenuation.
5.	Kaiser	It is adaptable, allowing a chosen compromise between transition width and stopband attenuation through the selection of the parameter β.

3 Neural Networks

3.1 History

In 1943, neurophysiologist Warren McCulloch moreover mathematician Walter Pitts composed a paper on how neurons may function. To depict how neurons in the cerebrum may operate, they displayed a basic neural system utilizing electrical circuits.

In 1949, Donald Hebb created The Organization of Behavior, a work which called attention to the way that neural pathways are reinforced each time they are utilized, an idea on an elementary level fundamental to the methods by which people learn. If two nerves fire in the meantime, he contended, the association between them is improved.

As PCs turned out to be further developed in the 1950s, it was at last conceivable to reenact a theoretical neural system. The initial move towards this was made by Nathanial Rochester from the IBM examine research facilities. Shockingly for him, the first endeavour to do as such fizzled.

In 1959, Bernard Widrow and Marcian Hoff of Stanford created models called "ADALINE" and "MADALINE." In a run of the mill show of Stanford's affection for acronyms, the names originate from their utilization of Multiple ADAptive LINear Elements. ADALINE was created to perceive parallel examples so that on the off chance that it was perusing spilling bits from a telephone line, it could anticipate the following piece. MADALINE was the first neural system connected to a genuine issue, utilizing a versatile channel that kills echoes on telephone lines. While the framework is as antiquated as aviation authority frameworks, similar to airport regulation frameworks, it is still in business use.

In 1962, Widrow and Hoff built up a learning methodology that inspects the incentive before the weight changes it (for example 0 or 1) as indicated by the standard: Weight Change = (Pre-Weight line esteem) * (Error/(Number of Inputs)). It depends on the possibility that while one dynamic perceptron may have a significant blunder, one can modify the weight esteems to appropriate it over the system, or if nothing else to nearby perceptrons. Applying this standard still outcomes in a blunder if the line before the weight is 0, although this will in the end right itself. If the mistake is rationed, so every last bit of it is dispersed to the majority of the loads than the blunder is wiped out.

Regardless of the later achievement of the neural system, customary von Neumann engineering assumed control over the processing scene, and neural research was abandoned. Unexpectedly, John von Neumann himself proposed the impersonation of neural capacities by utilizing broadcast transfers or vacuum tubes.

In a similar timeframe, a paper was composed that recommended there couldn't be an augmentation from the single-layered neural system to a numerous layered neural system. Also, multiple individuals in the field were utilizing a learning capacity that was on a fundamental level imperfect since it was not differentiable over the whole line. Thus, research and financing went down.

This was combined with the way that the early achievements of some neural prompted an embellishment of the capability of neural systems, particularly thinking

about the down to earth innovation at the time. Guarantees went unfulfilled, and on occasion more noteworthy philosophical inquiries prompted dread. Journalists considered the impact that the alleged "figuring machines" would have on people, thoughts which are still around today.

The possibility of a PC which programs itself is exceptionally engaging. On the off chance that Microsoft's Windows 2000 could reconstruct itself, it may almost certainly fix a great many bugs that the programming staff made. Such thoughts were engaging; however, extremely hard to execute. Also, von Neumann engineering was picking up in prevalence. There were a couple of advances in the field, however generally look into was rare.

In 1972, Kohonen and Anderson built up a similar system autonomously of each other, which we will talk about increasingly about later. The two of them utilized network science to depict their thoughts; however, did not understand that what they were doing was making a variety of simple ADALINE circuits. The neurons should actuate a lot of yields rather than only one.

The first multilayered organize was created in 1975, an unsupervised system.

In 1982, enthusiasm for the field was restored. John Hopfield of Caltech introduced a paper to the National Academy of Sciences. His methodology was to make increasingly helpful machines by utilizing bidirectional lines. Beforehand, the associations between neurons were just a single way.

That equivalent year, Reilly and Cooper utilized a "Half breed organize" with numerous layers, each layer using an alternative critical thinking technique.

Likewise, in 1982, there was a joint US-Japan meeting on Cooperative/Competitive Neural Networks. Japan declared another Fifth Generation exertion on neural systems, and US papers created stress that the US could be abandoned in the field. (Fifth era figuring includes automated reasoning. Original utilized switches and wires, the second era used the transistor, the third state utilized strong state innovation like coordinated circuits and more elevated amount programming dialects, and the fourth era is code generators.) therefore, there was all the more financing and subsequently more research in the field.

In 1986, with various layered neural systems in the news, the issue was the way to stretch out the Widrow-Hoff principle to numerous layers. Three free gatherings of scientists, one of which included David Rumelhart, a previous individual from Stanford's brain science office, thought of comparative thoughts which are presently gotten back to proliferation systems since it disseminates design acknowledgement blunders all through the system. Half and half systems utilized only two layers. These back-spread systems use many. The outcome is that back-proliferation systems are "moderate students," requiring perhaps a considerable number of emphasis to learn.

Presently, neural systems are utilized in a few applications, some of which we will depict later in our introduction. The initial thought behind the idea of neural networks is that on the off chance that it works in nature, it must most likely work in PCs. The eventual fate of neural systems, however, lies in the improvement of equipment. Much like the propelled chess-playing machines like Deep Blue, quick, effective neural systems rely upon apparatus being indicated for its possible use.

Research that focuses on creating neural systems is moderately moderate. Because

of the constraints of processors, neural networks take a long time to learn. A few organizations are attempting to make what is known as a "silicon compiler" to create a particular kind of coordinated circuit that is upgraded for the use of neural systems. Computerized, simple, and optical chips are the various types of chips being created. One may promptly limit simple flag as a relic of past times. Anyway, neurons in the cerebrum work more like a simple sign than advanced sign. While advanced warning has two unmistakable states (1 or 0, on or off), simple sign change among least and most extreme qualities, it might be for a little while, however, before optical chips can be utilized in business applications.

3.2 Models

3.2.1 Components of an artificial neural network

- **Neurons:-**

A neuron with name$j$ getting an info $p_j(t)$ from forerunner neurons comprises of the accompanying components:

an initiation$a_j(t)$, the neuron's state, contingent upon a discrete time parameter,

potentially an edge θ_j, which remains fixed except if changed by a learning capacity,

an actuation work f that processes the new initiation at a given time $t+1$ from $a_j(t)$,θ_j and the net info$p_j(t)$ offering ascend to the connection

$$a_j(t+1) = f(a_j(t), p_j(t), \theta_j) \tag{16}$$

,

furthermore, a yield workf_{out} figuring the yield from the initiation

$$o_j(t) = f_{out}(a_j(t)) \tag{17}$$

.

Regularly the yield work is essentially the Identity work.

An info neuron has no ancestor however fills in as information interface for the entire system. Likewise a yield neuron has no successor and accordingly fills in as yield interface of the entire system.

- **Connections, weights and biases:-**

The system comprises of associations, every association exchanging the yield of a neuron i to the contribution of a neuron j. In this sense$i$ is the ancestor of j and$j$ is the successor of i . Every association is alloted a weight w_{ij}. Sometimes a predisposition term is added to the complete weighted total of contributions to fill in as an edge to move the enactment work.

- **Propagation function:-**

The propagation function computes the input $p_j(t)$ to the neuron$j$ from the outputs $o_i(t)$ of predecessor neurons and typically has the form

$$p_j(t) = \sum_i o_i(t) w_{ij} \tag{18}$$

When a bias value is added with the function,

$$p_j(t) = \sum_i o_i(t) w_{ij} + w_{0j} \tag{19}$$

- **Learning rule:-**

The learning rule modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This learning process typically amounts to modifying the weights and thresholds of the variables within the network.

3.2.2 Activation Functions

- **Sigmoid Function :-**

It is a function which is plotted as 'S' formed diagram.

Condition :$A = 1/(1 + e - x)$

Nature : Non-direct. Notice that X extrems lies between - 2 to 2, Y esteems are steep. This implies, little changes in x would likewise achieve enormous changes in the estimation of Y.

Range : 0 to 1

Utilizations : Usually utilized in yield layer of a parallel grouping, where result is either 0 or 1, as incentive for sigmoid capacity lies somewhere in the range of 0 and 1 just along these lines, result can be anticipated effectively to be 1 if esteem is more noteworthy than 0.5 and 0 generally.

- *Tanh* **Function :-** The actuation that works quite often superior to anything sigmoid capacity is Tanh work additionally knows as Tangent Hyperbolic capacity. It's entirely moved rendition of the sigmoid capacity. Both are comparative and can be obtained from one another.

Condition :-

$$f(x) = tanh(x) = 2/(1 + e - 2x) - 1 \tag{20}$$

Or on the other hand

$$tanh(x) = 2 * sigmoid(2x) - 1 \tag{21}$$

Esteem Range :-$-1to + 1$

Nature :- non-direct

Utilizations :- Usually utilized in concealed layers of a neural system as it's qualities lies between - 1 to 1 consequently the mean for the hidden layer turns out be 0 or near it, subsequently helps in focusing the information by conveying mean near 0. This makes learning for the following layer a lot simpler.

- **RELU :-** Stands for Rectified linear units. It is the most broadly utilized activation function. Mainly executed in hidden layers of Neural network.

Condition :- $A(x) = max(0, x)$. It gives a yield x if x is certain and 0 generally.

Range :-$[0, inf)$

Nature :- non-straight, which implies we can without much of a stretch backpropagate the errors and have various layers of neurons being initiated by the ReLU work.

Utilizations :- ReLu is less computationally costly than tanh and sigmoid in light of the fact that it includes less complex numerical tasks. At once just a couple of neurons are actuated making the system meager making it proficient and simple for calculation.In straightforward words, RELU adapts a lot quicker than sigmoid and Tanh work.

- **Picking the right activation function**

The fundamental principle guideline is on the off chance that you truly don't have the foggiest idea what actuation capacity to utilize, at that point basically use RELU as it is a general initiation work and is utilized as a rule nowadays.On the off chance that your yield is for paired grouping, at that point, sigmoid capacity is exceptionally normal decision for yield layer.

3.2.3 Learning

Supervised learning: Supervised learning utilises a set of example pairs

$(x, y), x \in X, y \in Y(x, y), x \in X, y \in Y$

and the intention is to find a function $f : X \to Y f : X \to Y$ in the conceded class of functions that matches the examples. In other words, we want to mention the mapping indicated by the data; the cost function is relevant to the mismatch among our mapping and the data, and it inevitably contains earlier understanding about the problem field.

A generally used cost is the mean-squared error, which attempts to reduce the average squared error between the network's output, f(x), and the target value y over all the sample pairs. Reducing this cost using gradient descent for the class of neural networks termed as multilayer perceptrons (MLP), unfolds the backpropagation algorithm for training neural networks.

Tasks that fall within the criterion of supervised learning are pattern recognition (also perceived as classification) and regression (also regarded as function approximation). The supervised learning paradigm is further applicable to sequential data (e.g., for handwriting, speech and gesture recognition). This can be conceived of as learning with a "teacher", in the style of a function that presents constant feedback on the quality of solutions achieved thus far.

Unsupervised learning: In unsupervised learning, some data x is given and the cost function to be reduced, that can be any function of the data x and the network's output, f

The cost function is reliant on the task (the model domain) and any apriori hypotheses (the inherent properties of the model, its parameters and the observed variables).

Tasks that settle within the criterion of unsupervised learning are in general prediction problems; the pertinence includes clustering, the calculation of statistical distributions, compression and filtering.

3.2.4 Learning algorithms

1. **Gradient descent:**Gradient descent is a first-order iterative optimization algorithm for determining the minimum of a function. To obtain a local minimum of a function utilizing gradient descent, one needs steps equivalent to the negative of the gradient (or approximate gradient) of the function at the current point. If alternately, one apprehends steps proportional to the positive of the slope, one approximates to a local maximum of that function; the scheme is then known as gradient ascent.Gradient descent is also perceived as steepest descent. Nevertheless, gradient descent should not be mixed with the method of most precipitous decline for approximating integrals.

2. **Levenberg-Marquardt Algorithm:**The Levenberg-Marquardt (LM) algorithm is the most extensively employed optimization algorithm. It defeats simple gradient descent and other conjugate gradient methods in a extensive variation of problems. This document tries to produce an intuitive description for this algorithm. The LM algorithm is originaly shown to be a blend of vanilla gradient descent and Gauss-Newton iteration. Afterwards, different outlook on the algorithm is produced by viewing it as a trust-region method.The dilemma for which the LM algorithm presents an answer is termed Nonlinear Least Squares Minimization. This means that the function to be reduced is of the following special form :

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{22}$$

3.3 Algorithm in code

3.3.1 Phase 1: Propagation

Every spread includes the accompanying advances:

Spread forward through the system to produce the yield value(s)

Figuring of the cost (mistake term)

Spread of the yield initiations back through the system utilizing the preparation design focus to create the deltas (the contrast between the focused on and real yield esteems) of all yield and concealed neurons.

3.3.2 Phase 2: Weight Update

For each weight, the accompanying advances must be pursued:

The weight's yield delta and information initiation are duplicated to discover the slope of the weight.

A proportion (rate) of the weight's slope is subtracted from the weight.

This proportion (rate) impacts the speed and nature of learning; it is known as the learning rate. The more noteworthy the proportion, the quicker the neuron trains, however the lower the proportion, the more precise the preparation is. The indication of the angle of a weight shows whether the mistake changes straightforwardly with, or conversely to, the weight. In this manner, the weight must be refreshed the other way, "plunging" the inclination.

3.4 Backpropagation

Backpropagation algorithms are a class of methods used to efficiently train artificial neural networks (ANNs) following a gradient descent approach that utilises the chain rule. The main highlight of backpropagation is its iterative, recursive and efficient method for determining the weights updates to develop in the network until it is able to accomplish the task for which it is being trained.It is closely associated to the Gauss–Newton algorithm.

Backpropagation needs the derivatives of activation functions to be known at network design time. Automatic differentiation is a method that can automatically furthermore analytically render the derivatives to the training algorithm. In the context of learning, backpropagation is ordinarily used by the gradient descent optimization algorithm to fit the weight of neurons by estimating the gradient of the loss function.

If a constant 1 is fed into the output unit and the network is run backwards. Incoming information to a node is added and the result is multiplied by the value stored in the left part of the unit. The result is transmitted to the left of the unit. The result collected at the input unit is the derivative of the network function with respect to x.

3.5 Modes of learning

Two forms of learning are possible: stochastic and batch. In stochastic learning, each input produces a weight adjustment. In batch learning weights are customised based on a batch of inputs, accumulating errors over the batch. Stochastic learning adds "noise" into the gradient descent process, using the local gradient estimated from one data point; this decreases the possibility of the network getting stuck in

local minima. Nevertheless, batch learning typically generates a faster, more stable descent to a local minimum, since every update is presented in the direction of the average error of the batch. A well-known trade-off choice is to use "mini-batches", indicating small batches and with samples in every batch chosen stochastically from the complete data set

4 Implementation

4.1 Supervised learning

Initially, a simple feedforward network with 2 inputs, 100 outputs and 15 neurons in the hidden layer was implemented. The output of the network, a 99 order FIR filter, was trained under supervision using a sample set of 3000 different FIR filters designed using Kaiser window with random frequencies and values of β. Levenberg-Marquardt algorithm was used to train the network because of its robustness and estimated time to converge. The Artificial neural network was trained for 94 epochs while reaching 6 test case validation after which it achieved the targeted mean square error value. Hence, the artificial neural network was trained resulting in 4.95e-5 mean square error and 9.95e-1 regression.

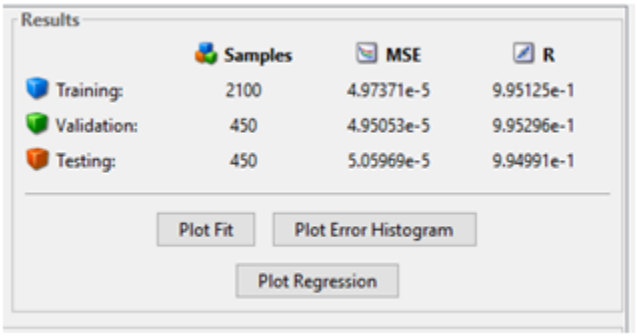


Figure 1

Figure 11: MSE and Regression values in various stages

Implementation 1 results

- The mean square error of the output from the desired output was plotted.

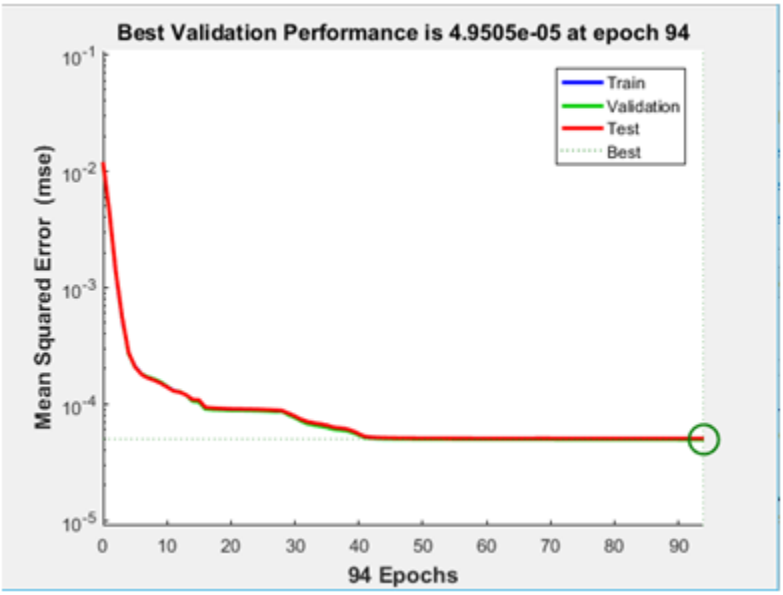


Figure 12: Mean square error of the output

● The gradient, Mu and validation checks used while training the artificial neural networks have been plotted. [Figure 3]

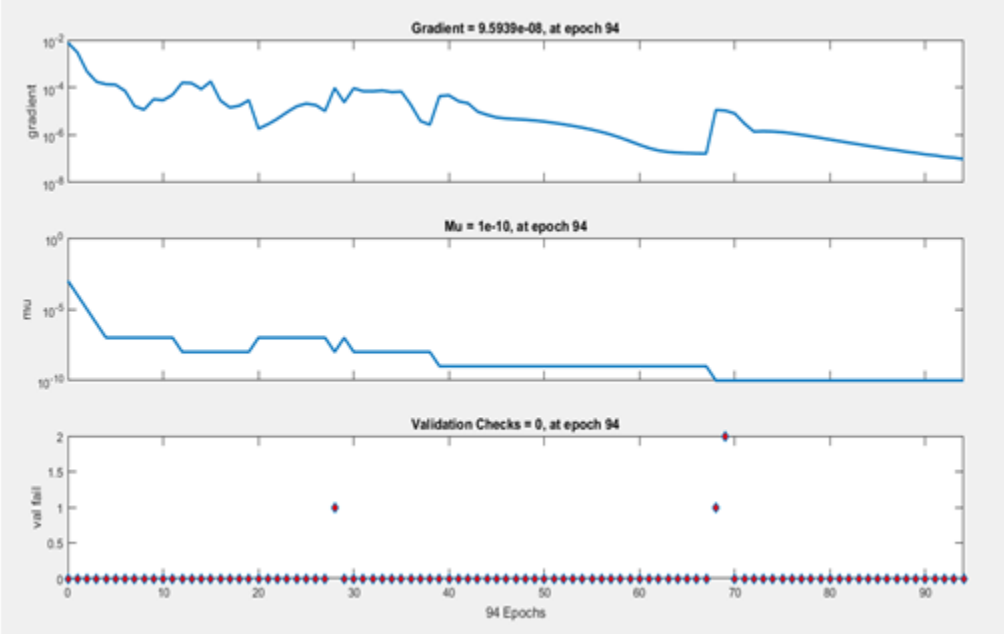


Figure 13: (a)Gradient (b)mu (c)Validation check

● Error histogram was plotted for the in training neural network, neural network in validation phase and neural network in testing phase. [Figure 4]

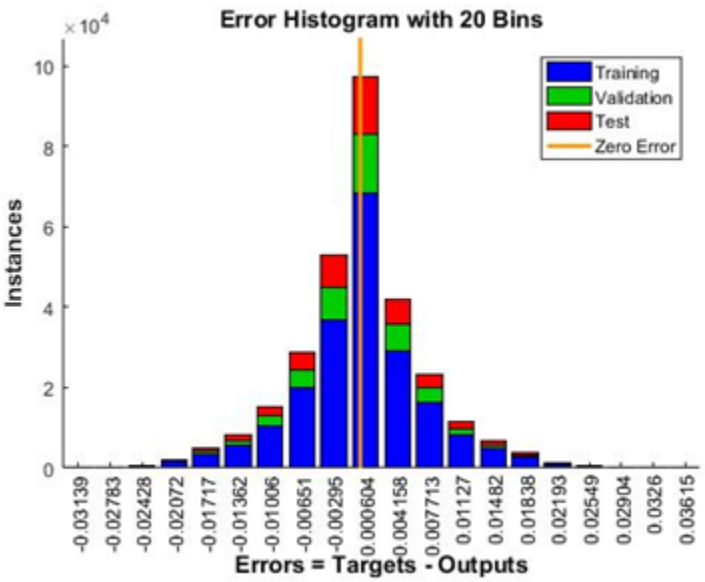


Figure 14: Instances of errors

● Regression was potted where a dot shows the regression for each instances of each dataset value for in training neural network, neural network in validation phase and neural network in testing phase. [Figure 5]

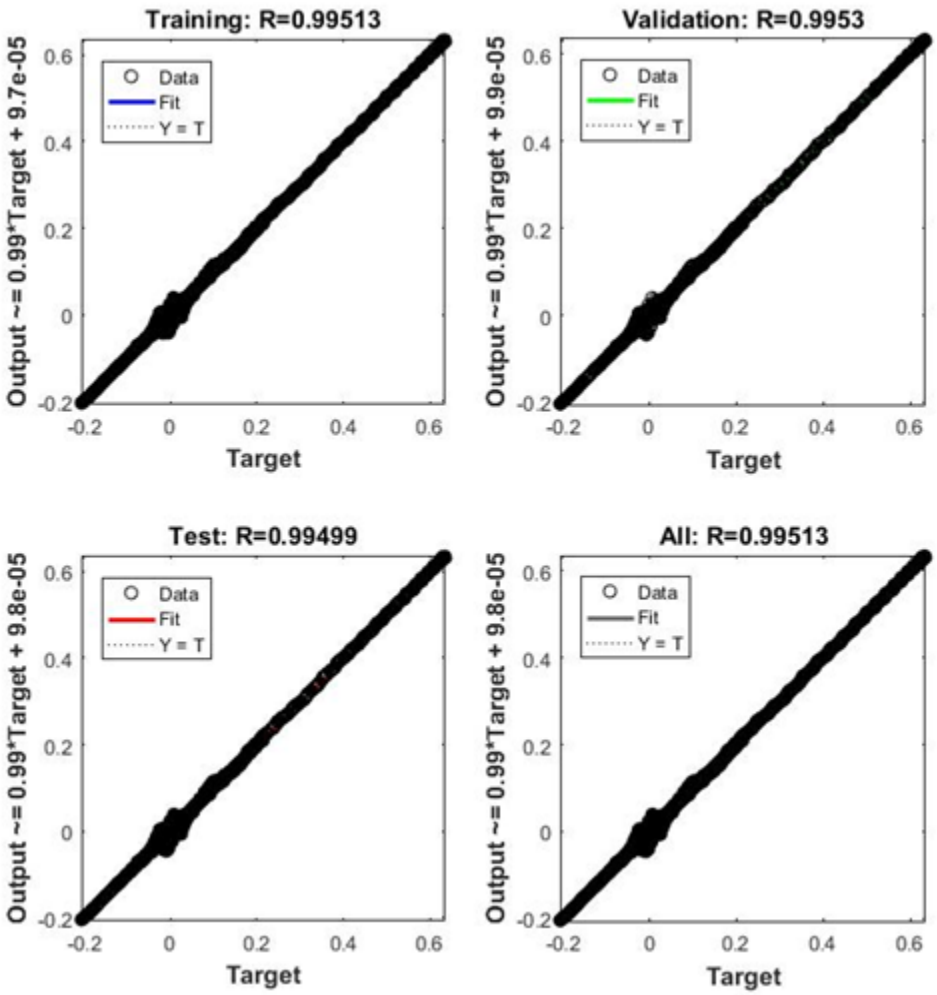


Figure 15: Output regression

● Finally, the trained network was used to design an FIR filter for 0.3 normalised frequency using the the trained ANN.Hence, -24 dB peak stopband ripple was seen with 3-dB frequency at 0.29 normalised frequency. [Figure 6]

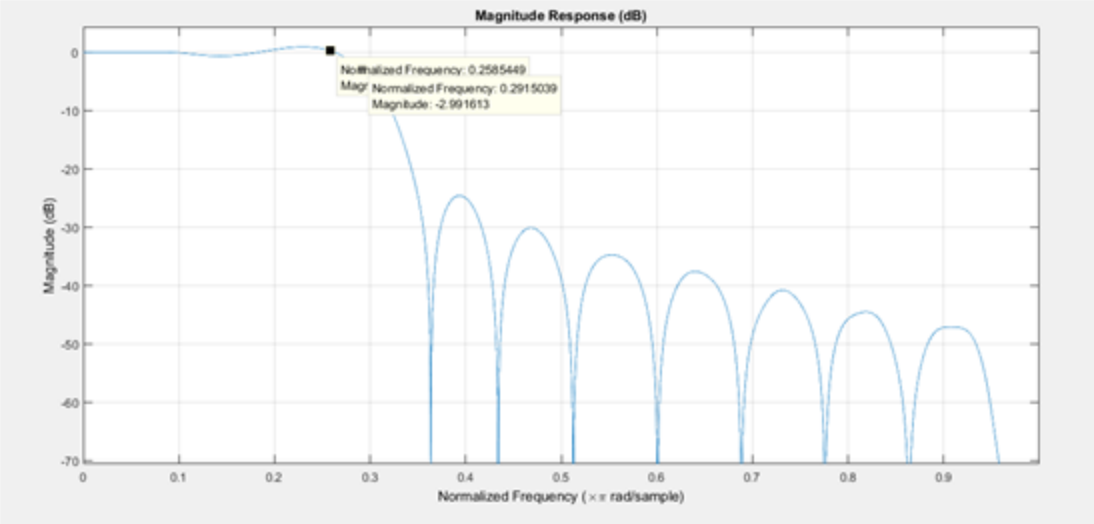


Figure 16: Magnitude Response

4.1.1 Implementation 1 Problems

● The current method of FIR filter design using ANN provides us with results which are either on par or slightly worse than existing methods of FIR filter design. This is because the dataset used to train the network is the best possible outcome that the network can achieve.

● The current network only guesses a value based on the training dataset. No mathematical computations are involved in generating the coefficients for a filter not in its training dataset.

The next approach is an attempt to overcome the problems of this implementation.

Code To Generate 3000 Random Samples

$n = 99;$
$h = zeros(3000, 50);$
$s2 = zeros(3000, 2);$
$for\ i = 1 : 3000$
$s = 0.1 + 0.85 * rand(1, 1);$
$b = 0.1 + 5 * rand(1, 1);$
$s2(i, 1) = s;$
$s2(i, 2) = b;$
$window2 = kaiser(n + 1, b);$
$d4 = fir1(n, s, window2);$
$forj = 1 : 100$
$h(i, j) = d4(j);$
end
end

4.2 Unsupervised learning

As per the aim of the project , we first randomly initialised the filter coefficients and then trained them towards a near ideal response using Artificial Neural Network algorithms.

As is a well known fact, the transfer function of an (N-1)th order FIR filter is given by

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n} \tag{23}$$

If h(n) is the impulse response, N is odd and N is the length of the filter, then for a linear phase , the following condition holds true.

$$\mathrm{h(n)} = \mathrm{h(N-1-n)} \tag{24}$$

The amplitude frequency response in such a case can be given by

$$\hat{H}(\Omega) = \sum_{n=0}^{\frac{N-1}{2}} w_n \cos(n\Omega) \tag{25}$$

$$w_0 = h\left(\frac{N-1}{2}\right), w_n = 2h\left(\frac{N-1}{2} - n\right), n = 1, 2, \ldots, \frac{N-1}{2} \tag{26}$$

We see from Eq. (23) that the amplitude-frequency response of FIR filters with a linear phase is a Fourier series, and wn is the Fourier coefficients. Once having wn , we can acquire h(n) via (24)

$$h\left(\frac{N-1}{2}\right) = w_0 \tag{27}$$

$$h\left(\frac{N-1}{2} - n\right) = \frac{w_n}{2} \tag{28}$$

The algorithm used in our approach is based on the activation matrix C^T produced by cosine basis functions in order to obtain w_n

4.2.1 Algorithm Description

Suppose

$$\mathbf{W} = [w_0, w_1, \cdots, w_{N-1}]^T \, \mathbf{\Omega} = \left[\Omega_0, \Omega_1, \cdots, \Omega_{\frac{N-1}{2}}\right]^T \tag{29}$$

$$\hat{\mathbf{H}} = \left[\hat{H}\left(\Omega_0\right), \hat{H}\left(\Omega_1\right), \cdots, \hat{H}\left(\Omega_{\frac{-W-1}{2}}\right)\right]^T \tag{30}$$

Thus , the equation (30) can be expressed as

$$\hat{\mathbf{H}} = C^T W(3) \tag{31}$$

As per the model of the neural network shown in fig.17. The output of the neural network is expressed by

$$H^k = C^T W^k \tag{32}$$

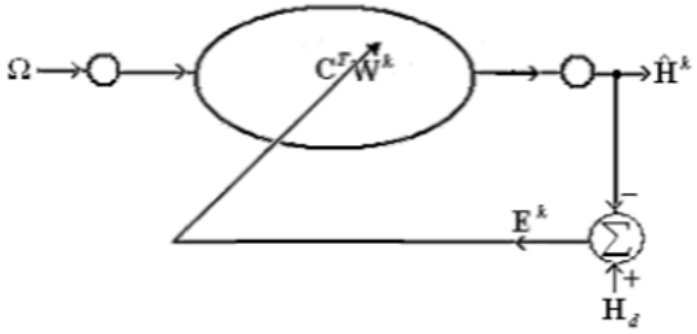


Figure 17: The model of neural network

where $\hat{\mathbf{H}}^k$ is the output of the ANN and Hd is the output of the desired filter W^K is the weight vector of the neural network, C^T is the activation matrix of the hidden units of neural network, A Cost function is defined as

$$J = \frac{1}{2} \left\| \mathbf{E}^k \right\|^2 \tag{33}$$

where $\mathbf{E}^k = \mathbf{H}_d - \hat{\mathbf{H}}^k$

E^k is the Error vector between the desired and actual outputs and $\left\| \mathrm{E}^k \right\|^2$ is the Frobenius norm of the vector. To minimize J , the W^k is recursively calculated by using Gradient Descent algorithm, So the updated weights are given as - $\mathbf{W}^{k+1} = \mathbf{W}^k - \eta \frac{\partial J}{\partial \mathbf{W}^k}$

where $\eta > 0$ is the learning rate.

After differentiating (5) we have ,

$$\frac{\partial J}{\partial \mathbf{W}^k} = \frac{\partial \mathbf{H}}{\partial \mathbf{W}^k} \frac{\partial \mathbf{E}^k}{\hat{\mathbf{H}}} \frac{\partial J}{\partial \mathbf{E}^k} = -\mathbf{C}\mathbf{E}^k \tag{34}$$

Thus , the updated weights are given as

$$\mathbf{W}^{k+1} = \mathbf{W}^k + \eta \mathbf{C}\mathbf{E}^k \tag{35}$$

Thus we can design an FIR digital filter with a linear phase by training the amplitude-frequency response vector Hd of an ideal FIR digital filter to obtain the weight vector Wk of neural network, i.e. Fourier coefficient vector W of FIR digital filter according to the parallel algorithm of neural network.

So ,to conclude, following are the steps involved in designing a filter using ANN

Step 1- Sample the desired frequency response$\left|H_d\left(e^{i\Omega_l}\right)\right|$ equably to obtain the training sample vector H_d of the neural network where$\Omega_l = \frac{2\pi}{N-1}l$, 1 = 0, 1, 2, .. (N_1) /2 and

$$\mathbf{\Omega} = \left[\Omega_0, \Omega_1, \cdots, \Omega_{\frac{N-1}{2}}\right]^T \tag{36}$$

Initialise the random weight vector W using arbitrary values and define an arbitrarily small number Tol(tolerance). Define the activation matrix C for the hidden units of the Neural Network and initialise the learning rate

Step 2- Produce new predicted output vector $\hat{\mathbf{H}}^k$ of neural network using

Step 3- Calculate the error vector Ek and J

Step 4- Update the weight vector Wk+1 using

Step 5- If J>Tol, go to step 2 or else, stop training the network.

Step 6- Once the weight vector W is obtained , obtain the filter coefficients using

.

4.2.2 Implementation Of The Algorithm And Results

Designing The Filter An Nth order low pass FIR filter (take for instance N = 2000) is to be designed using the artificial neural network approach. The following steps show the detailed working as well as the MATLAB code along with the results.

- Say the filter has the following characteristics $\left|H_d\left(e^{j\Omega}\right)\right| = \quad 1, |\Omega| \leq$ 1,0, otherwise

Here N = 2001. We sample$H_d\left(e^{j\Omega}\right)$ to obtain the training sample vectorH_d of $((N+1)/2)$ samples (in this case 1001) in the range of $\Omega \in [0, \pi]$,i.e. l (in general $\Omega_l = \pi/((N-1)/2)l$) and l = 0,1,2.....1000 (in general till (N-1)/2). Thereafter we produce the random weight vector W , define Tolerance (Tol) to a small value (say 10^{-20}). Depending upon the characteristics of the required filter, the cutoff frequency must reflect the index of the Hd array where the transition occurs. For this we equate $\Omega_l = \pi/((N-1)/^2)$ l to 1 which gives us $l = (N-1)/2\pi$

The MATLAB code for this is as follows :

$N = 2001;$
$cx = int16((N-1)/(2*pi));$
$Hd = [ones(1, cx), 0.78, 0.25, zeros(1, ((N-1)/2) - cx - 1)];$
H_d=[zeros(1,638),0.25,0.78,ones(1,361)]
H_m=[ones(1,320),0.78,0.25,zeros(1,316),0.25,0.78,ones(1,361)]
$Tol = 1e-20;$
$w = rand(((N-1)/2)+1, 1);$

- We now define the activation matrix C and calculate the learning rate (η) by taking the square of its Euclidean Norm. The MATLAB code for this is as follows :

$C = zeros(((N-1)/2)+1, ((N-1)/2)+1);$
$C(1,:) = ones(1, ((N-1)/2)+1);$
$fori = 2 : ((N-1)/2)+1$
$forj = 1 : ((N-1)/2)+1$
$C(i,j) = cos((i-1) * (2 * pi/(N-1)) * (j-1));$
end
$endlearnrate = 1.35/((norm(C))^2);$

- Having all the initial variables, we now code the actual back propagation algorithm as shown below.

$Hd = Hd';$
$while1$
$Ct = transpose(C);$
$Hk = Ct * w;$
$Ek = Hd - Hk;$
$J = ((norm(Ek))^2)/2;$
$if(J < Tol)$
$break;$
end
$w = w + learn_rate * C * Ek;$
end

As is evident, the weight vector W is being updated according to the rule

$$W^{(k+1)} = W^k + \eta C E^k \tag{37}$$

After every iteration, the cost function J is calculated. As soon as it goes below our defined tolerance value, the algorithm is halted.

- Once we have the weight vector, we can plot the response using the formula

$$\hat{H}(\Omega) = \sum_{n=0}^{\frac{N-1}{2}} w_n \cos(n\Omega) \tag{38}$$

The MATLAB code for this is as follows :

$omega = zeros(1, ((N-1)/2)+1);$
$forx = 1 : ((N-1)/2)+1$
$omega(1,x) = (2 * pi/(N-1)) * (x-1);$
end
$Homega = zeros(((N-1)/2)+1, ((N-1)/2)+1);$
$fory = 1 : ((N-1)/2)+1$
$Homega(y,:) = w(y,1) * cos((y-1) * omega);$

end
$Hfinal = zeros(1, ((N-1)/2)+1);$
$forq = 1 : ((N-1)/2)+1$
$Hfinal = Hfinal + Homega(q,:);$
end
$plot(omega, Hfinal)$
$xlabel(`NormalisedFrequency')$
$ylabel(`Magnitude')$

The values of $\hat{H}(\Omega)$ are stored in the H_{final} vector. The omega vector has also been defined which would be plotted against the X axis. Finally the response is plotted by calling the plot(omega,H_{final}) function. Following are the plots obtained for different values of N.

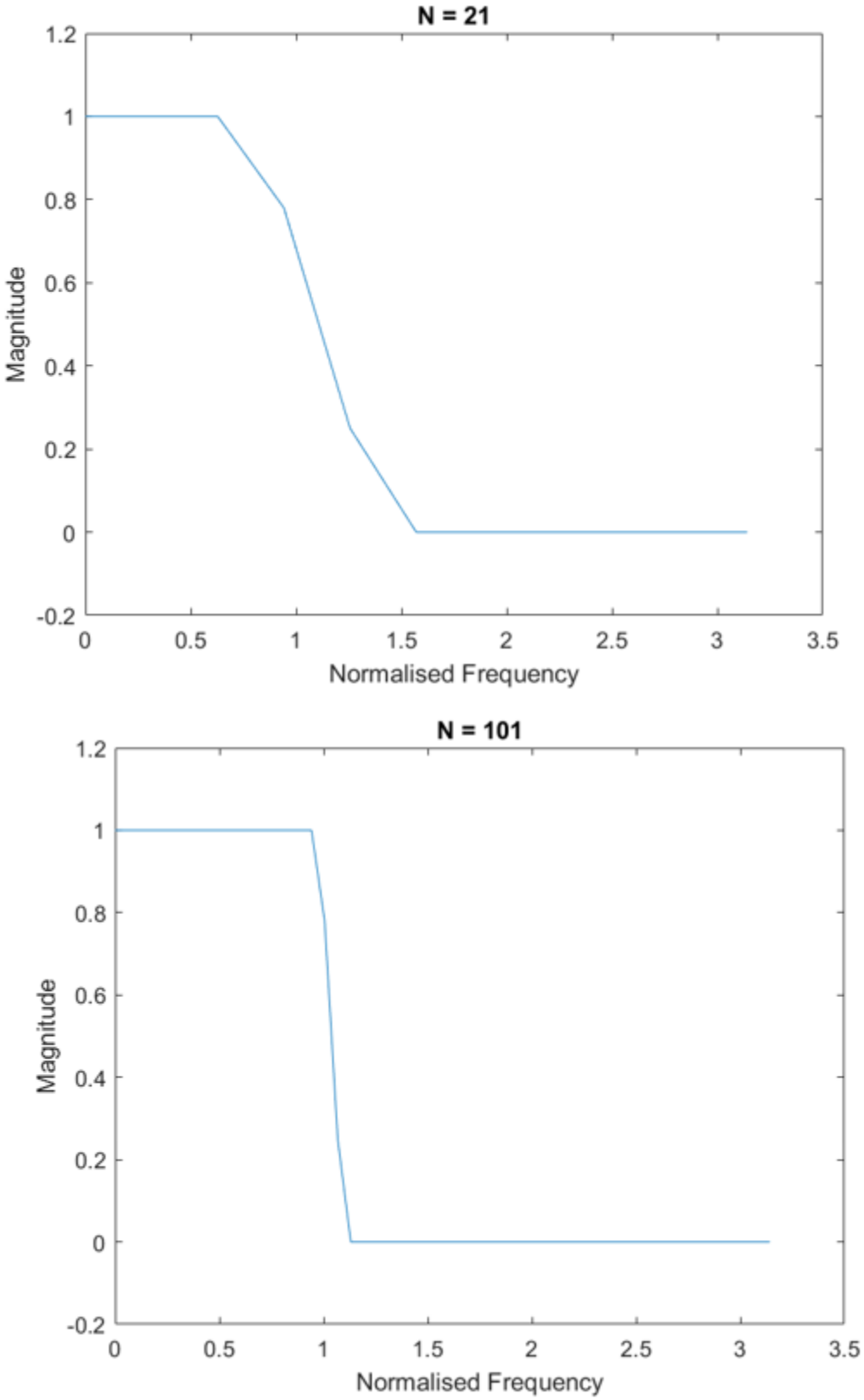
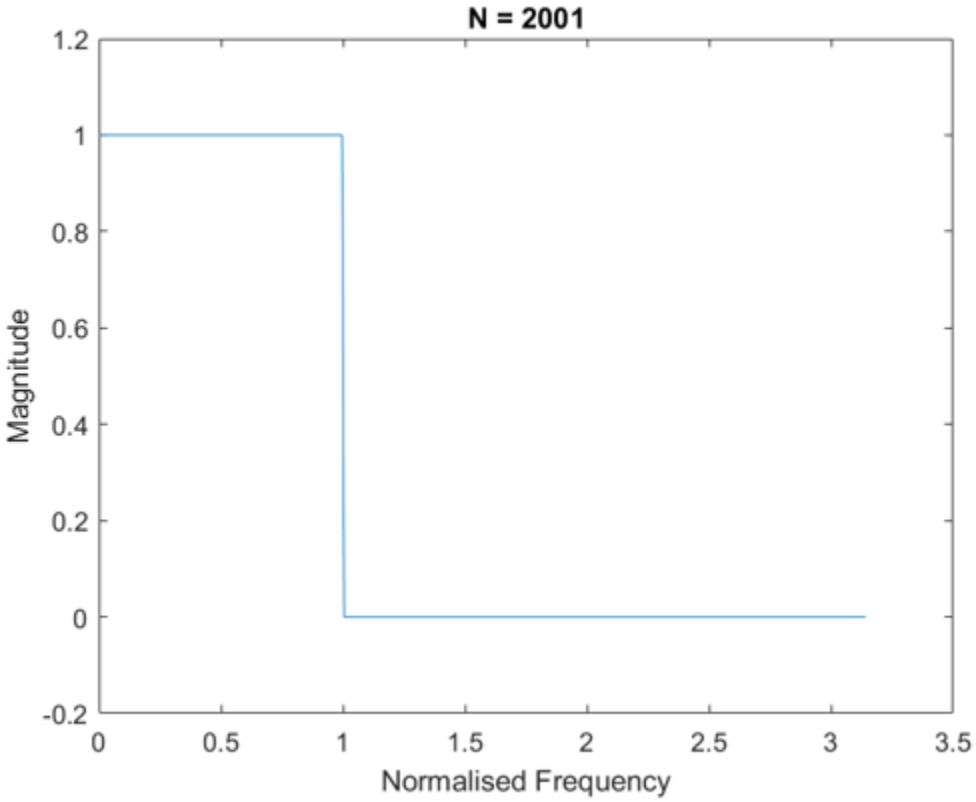


Figure 18: (a)Magnitude response for N = 21 (b)Magnitude response N=101

As is evident, the smoothness of the response increases as the order increases. Also, if the tolerance is reduced, the response worsens as well. The following is the response for N=2001 for Tol = 0.1



Magnitude response

These responses however are theoretical. In order to compute the practical responses, fvtool is used. This tool takes in the filter coefficients and plots the magnitude response by actually passing waves of multiple frequencies through the filter and recording the result , thus giving a more practical view of the response. However , we have the fourier weights Wn but not the filter coefficients. Thus, we need to use to obtain the coefficients first. The MATLAB code for the same is as follows :

$hfilter = zeros(N, 1);$
$hfilter(((N-1)/2)+1, 1) = w(1, 1)$
$fordf = 1 : ((N-1)/2)$
$hfilter((((N-1)/2)+1) - df, 1) = w(df+1, 1)/2;$
end
$forrt = 1 : ((N-1)/2)$
$hfilter(N+1-rt, 1) = hfilter(rt, 1);$
end

$fvtool(hfilter)$;

$hfilter$ is the vector of filter coefficients.
fvtool(hfilter) would plot the practical response of the required filter.
Following are the responses obtained for various values of N.

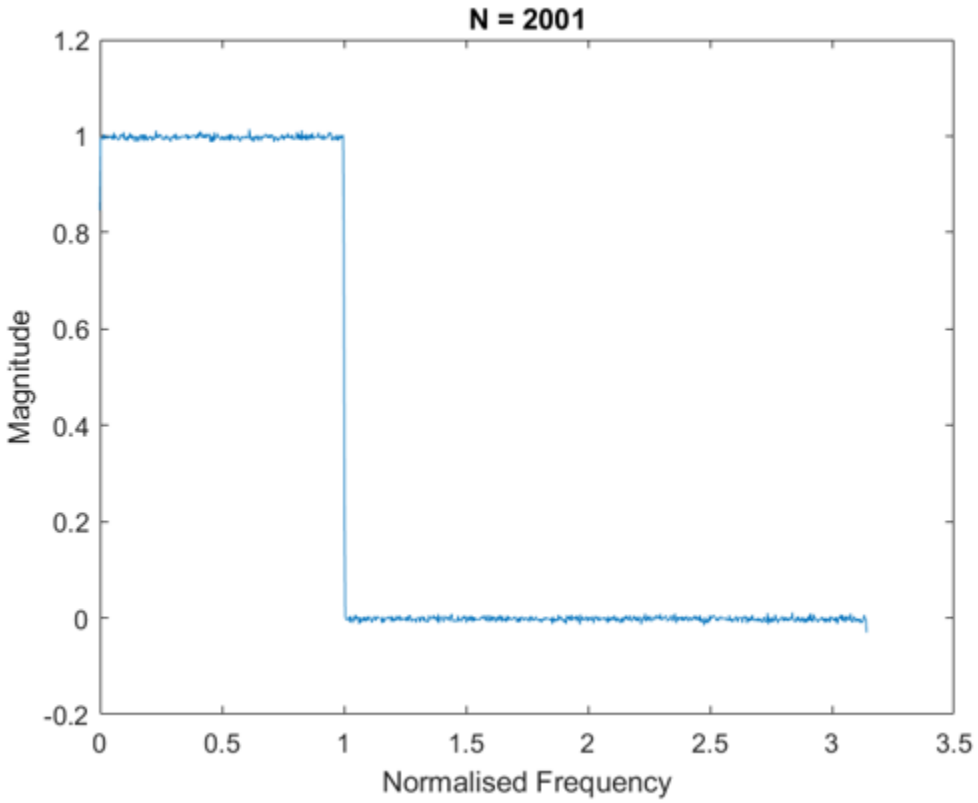


Figure 19: Magnitude Response for N = 2001 (with noise)

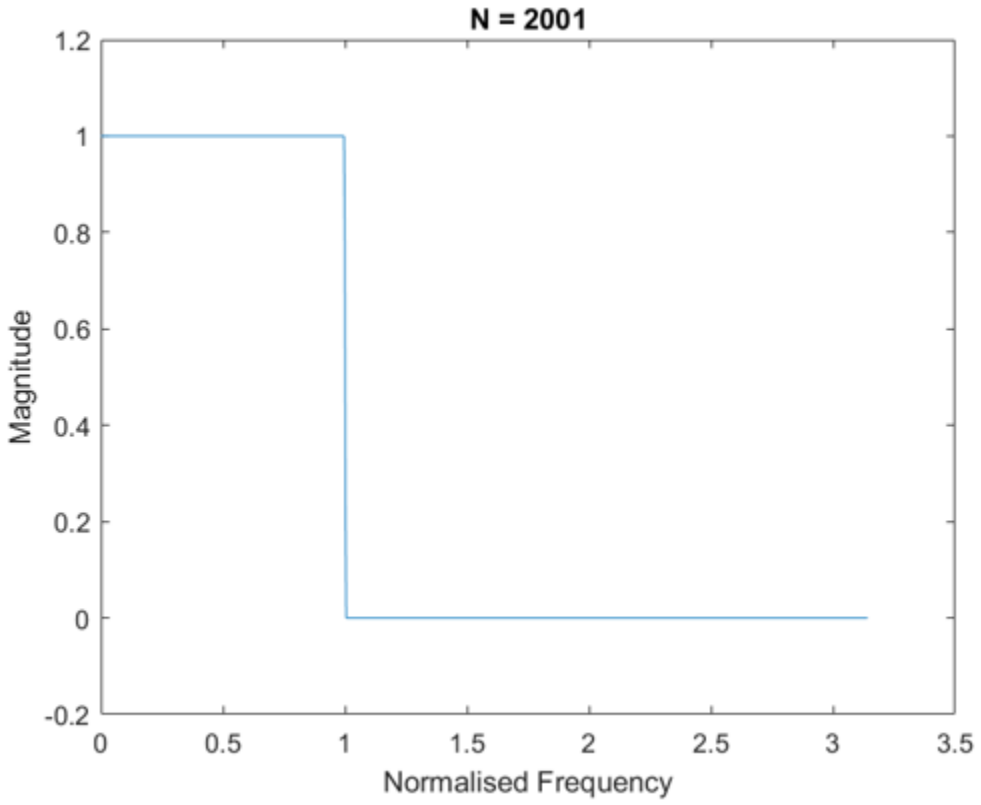


Figure 20: Magnitude Response for N = 2001 (without noise)

As is evident from these graphs, with increasing order, the transition width reduces and the stop band attenuation also increases

.

4.2.3 Testing The Designed Filter

Once the algorithm has been used to design the filter, it must be tested in order to check whether it is blocking the undesired frequencies. For this purpose, we denormalise the filter by taking a sampling frequency Fs= 7000 and passing waves of frequencies in the passband as well as stop band and note the results. After denormalisation, the magnitude reponse of the N=2001 filter looks like the following :

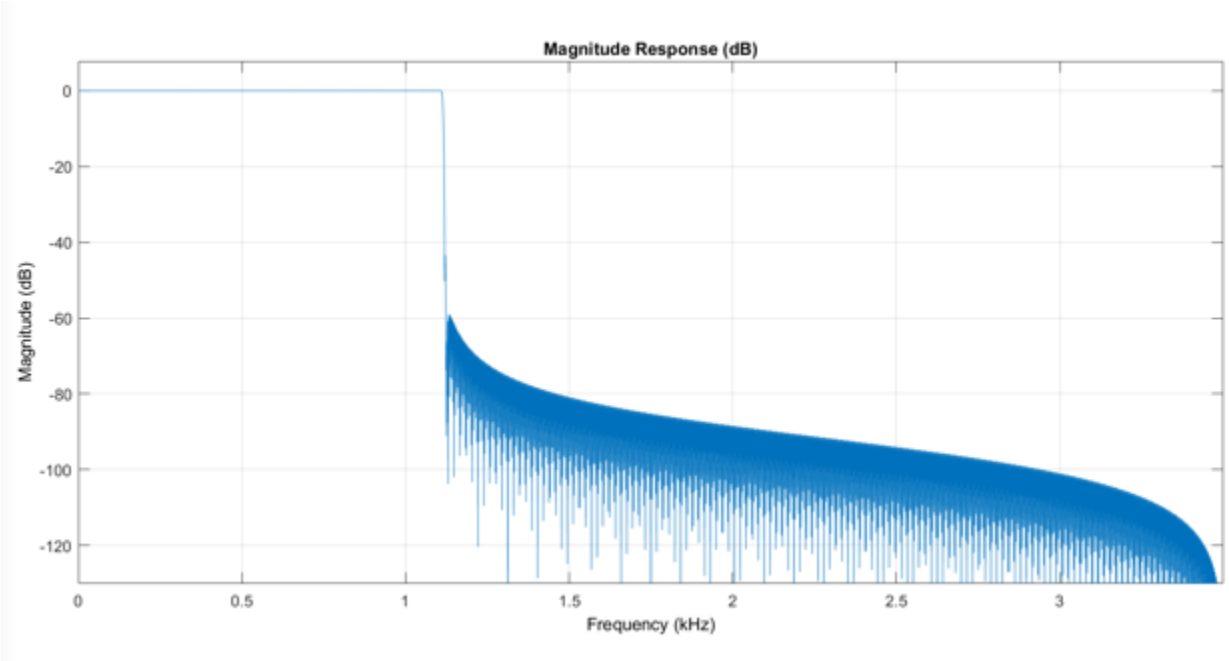
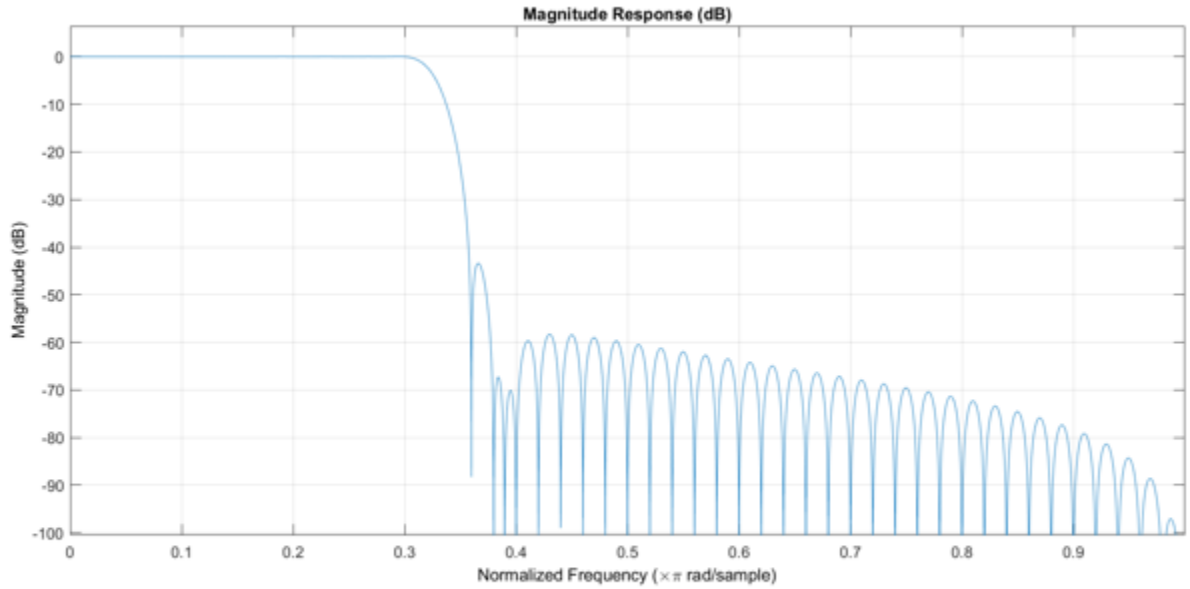


Figure 21: Magnitude Response

As evident, the cutoff frequency is somewhere around 1200 Hz. The following MATLAB code is used to pass and record the repose of Sinusoidal waveforms.

$Fs = 7000;$
$t = linspace(0, 1, Fs);$
$x = cos(2 * pi * 20 * t) + cos(2 * pi * 2750 * t)$;
$subplot(2, 1, 1);$
$plot(t, x)$
$y = filter(hfilter, 1, x)$
$subplot(2, 1, 2);$
$plot(t, y)$



$N = 201$

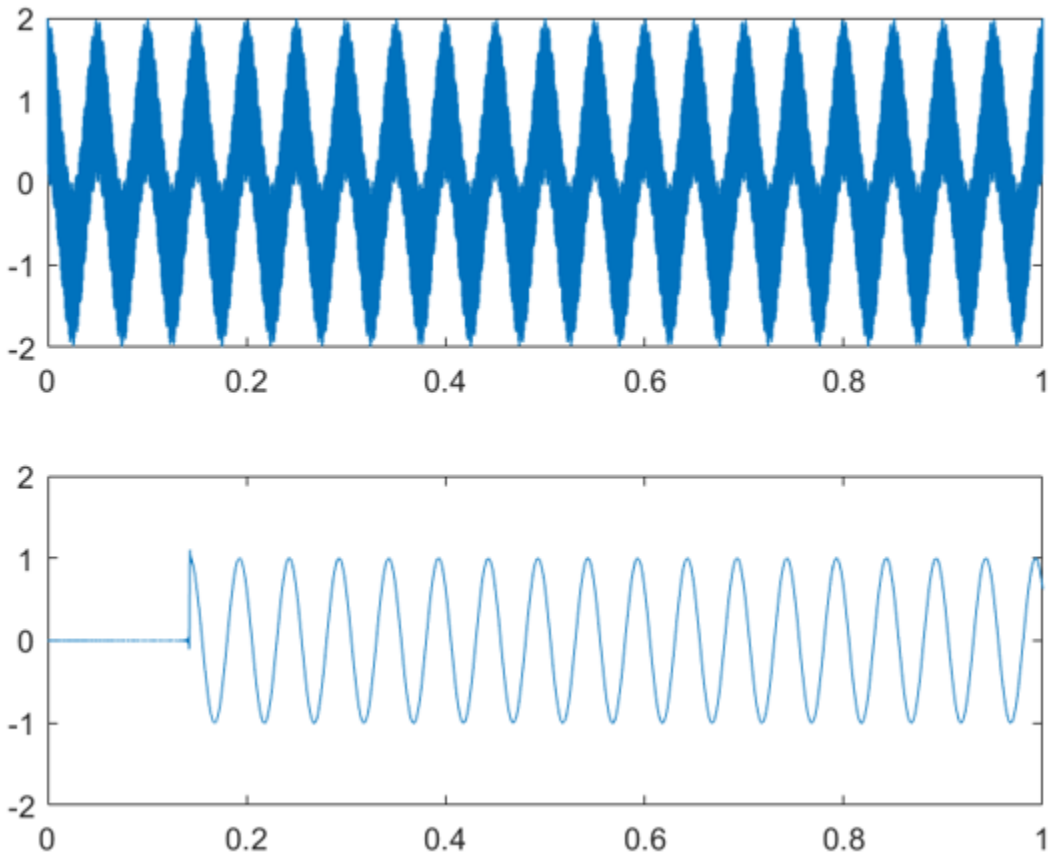


Figure 22: Magnitude Response

As is evident from these graphs, the input waveform consists of two frequencies, 20 Hz(lying in the passband) and 2750Hz (lying in the stopband). The output filtered waveform contains only the 20Hz waveform. Thus, the filter works fine. The same can be verified by taking the Fourier Transform of the input and output waveforms. The following code is used to take the fft of the filtered and unfiltered waveforms and plot the same.

$n = length(x);$
$fshift = (-n/2 : n/2 - 1) * (7000/n);$
$yshift = fftshift(xfft);$
$subplot(2, 1, 1);$
$plot(fshift, abs(yshift))$
$xfft = fft(y);$
$n = length(y);$
$fshift = (-n/2 : n/2 - 1) * (7000/n);$
$yshift = fftshift(xfft);$
$subplot(2, 1, 2);$
$plot(fshift, abs(yshift))$

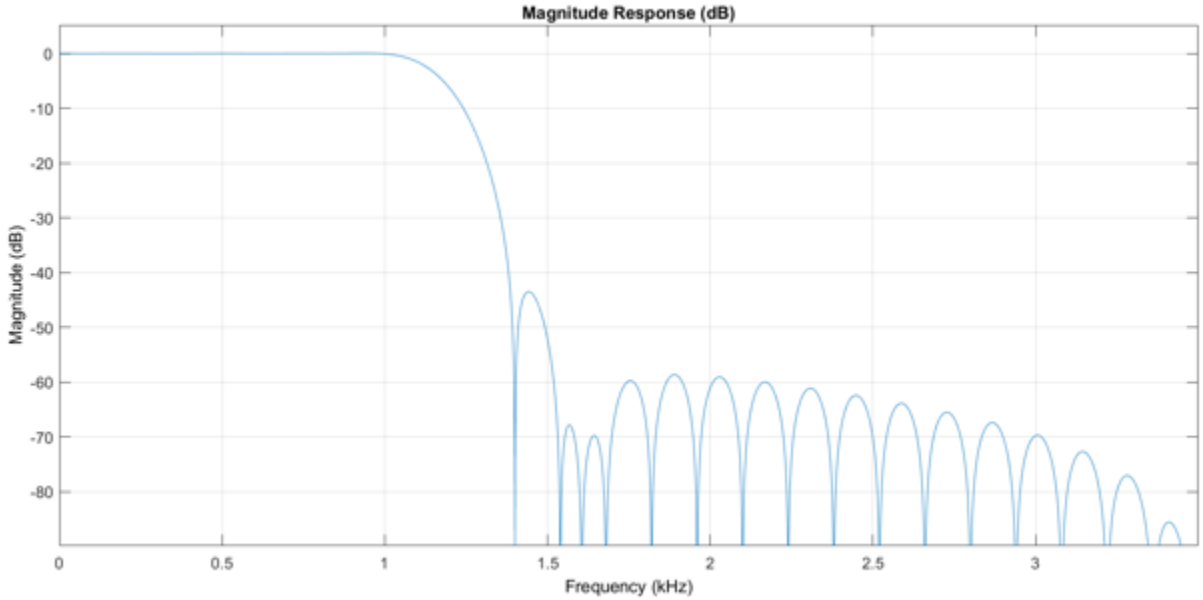


Figure 23: Magnitude Response

For intelligibility, the two frequencies taken are 200Hz and 2750Hz

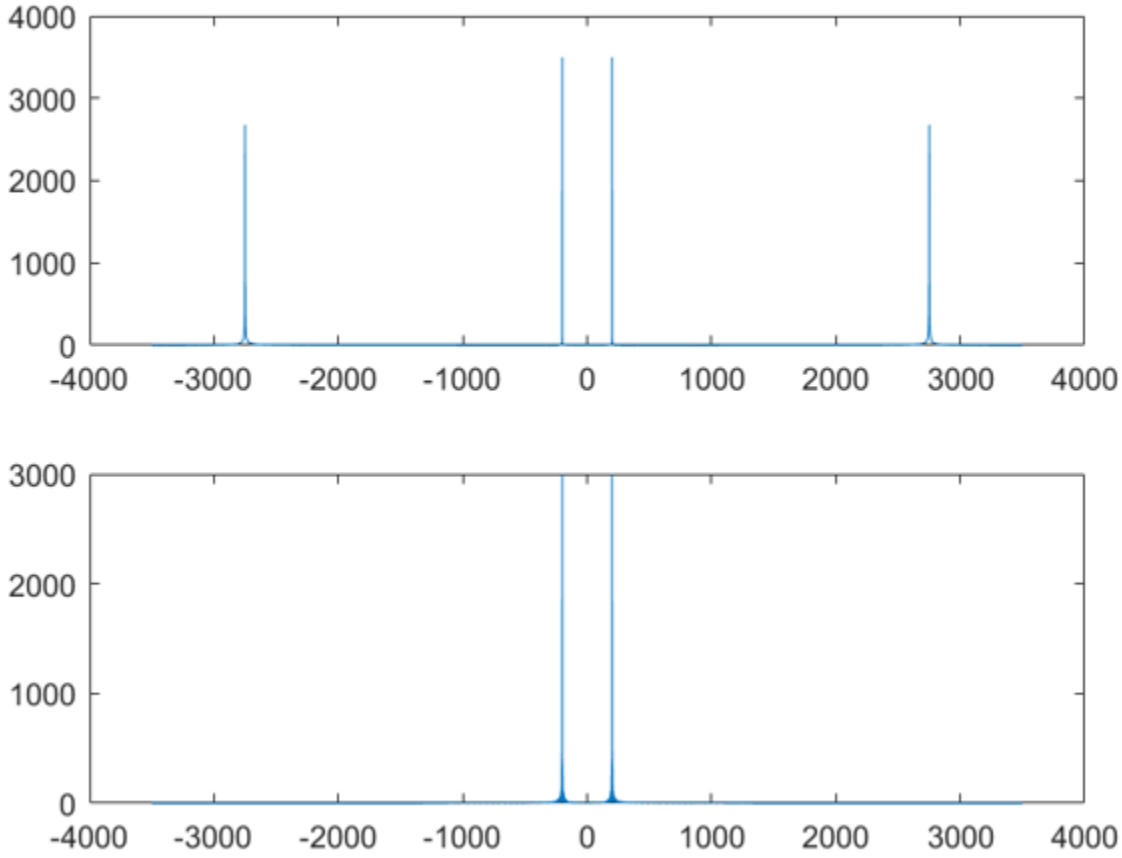


Figure 24: Magnitude Response

It is clear from the above plots that the filtered waveform only contains the 200Hz frequency while the 2750Hz gets blocked.

A further validation involves ensuring that the frequencies lying in the transition band get appropriately attenuated. For this we have taken an N= 51 filter as the transition band is sufficiently wide.

A sinusoidal waveform of amplitude 5 and frequency 1175 Hz is passed through this filter and the response is recorded.

The input and output waveforms are as follows:

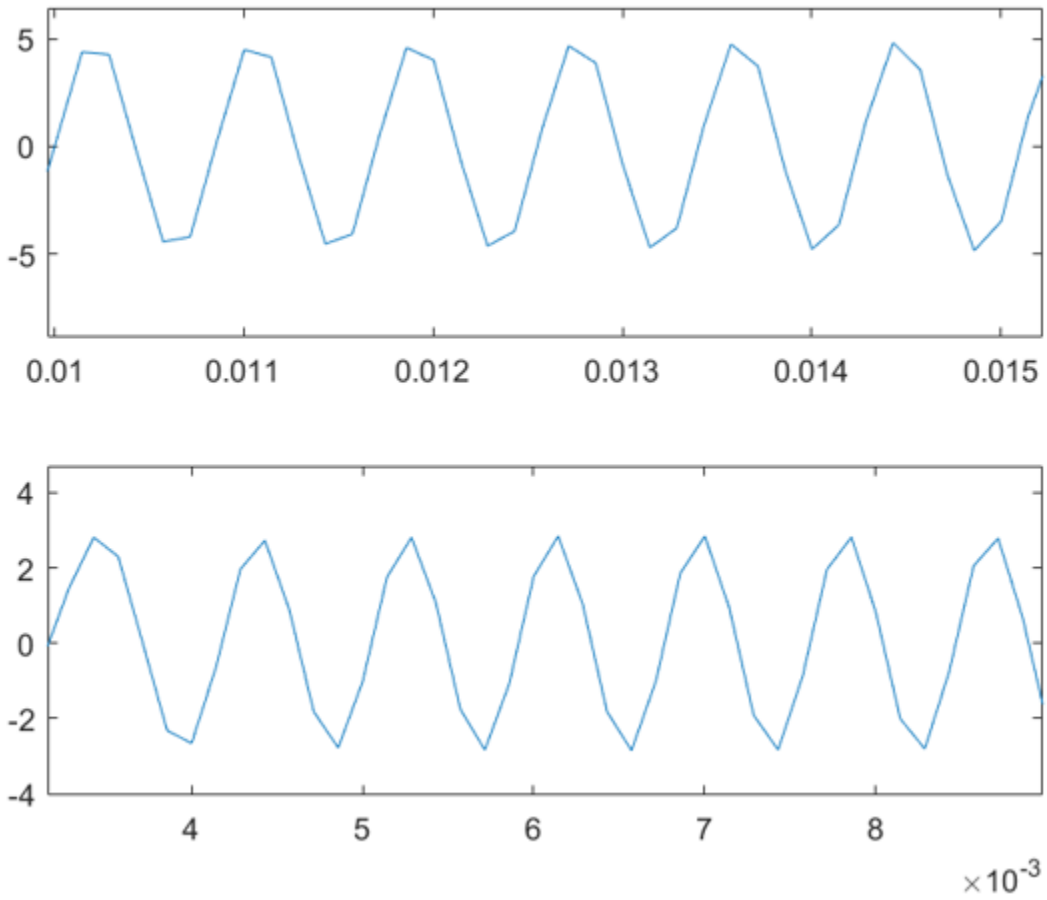


Figure 25: Magnitude Response

As is evident, a wave having frequency lying in the transition band of the filter is being appropriately attenuated. Thus, it can be safely concluded that the algorithm works perfectly and the results are satisfactory.

5 CONCLUSION

In this project, a new method for the calculation of coefficients of an FIR filter has been introduced which involves the use of Neural Networks. The initial objective of the project which was to find an alternative to computation intensive designing methods like windowing has been achieved as the proposed method uses multiple iterations and back propagation of errors to refine the initialised values of the required vector(vector of filter coefficients in our application) in order to make them as ideal as possible.

After running the simulations for different values of N(order) and observing the results, it can be concluded that the proposed method of designing FIR filters with Artificial Neural Network is less computation intensive than the existing methods and gives correct and satisfactory results consistent with theoretical expectations.

Additionally, the proposed method allows easy programmability to incorporate user defined cutoff, order and type of filter. Implementation on MATLAB allows easy integration with Xilinx Design Suite which would facilitate seamless implementation of the filter on FPGA.

References

[1] Rawat, T. (2015). *Digital signal processing*. New Delhi, India: Oxford University Press.

[2] Fausett, L. (2004). *Fundamentals of neural networks*. Delhi, India: Pearson Education.

[3] Wang, X. and He, Y. (2008). A neural network approach to FIR filter design using frequency-response masking technique. Signal Processing, 88(12), pp.2917-2926.

[4] Alwahab, D., Zaghar, D. and Laki, S. (2018). FIR Filter Design Based Neural Network. 2018 11th International Symposium on Communication Systems, Networks & Digital Signal Processing (CSNDSP).

[5] Z. Zeng, Y. Chen and Y. Wang, "Optimal Design Study of High-Order FIR Digital Filters Based on Neural-Network Algorithm", 2006 International Conference on Machine Learning and Cybernetics, 2006.

[6] Oppenheim, A. and Schafer, R. (n.d.). *Discrete-time signal processing.*Delhi, India: Pearson Education.