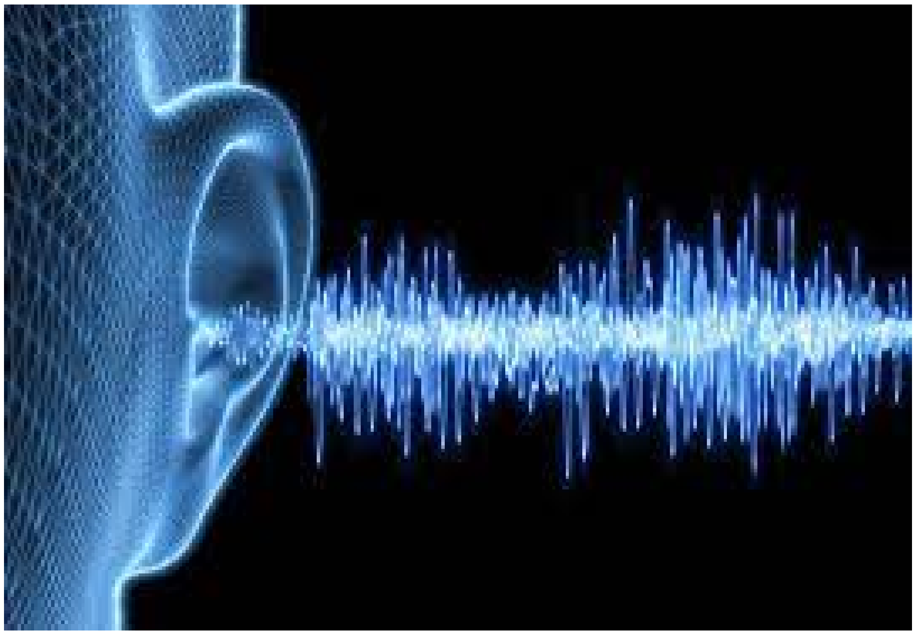


Hearing Aid using Digital Signal Processing Techniques

EE338: Application Assignment

Group 20
Aditya Anavkar, 19d070004
Amrit Rao, 19d070008
Mitali Singal, 19d070036

April 18, 2022

Contents

1 Problem Aspect **3**

2 Our Approach using DSP Methods **3**

3 Wiener filter **4**
3.1 Our Analysis: 4

4 Discrete Wavelet Transformation **7**
4.1 Our Analysis: 10

5 Spectral Subtraction using Short-Time Fourier Transform (STFT) **14**
5.1 Short-Time Fourier Transform 14
5.2 The Vuvuzela sound denoising algorithm and our analysis . . . 16
5.3 Proposed Lowpass Filter 19

6 Hardware Implementation **20**

7 Conclusions and Relevance to Aatmanirbhar Bharat **20**

8 Code: **22**

1 Problem Aspect

A person is said to be suffering from hearing impairment if he/she has a hearing loss of 40db below the threshold for hearing. According to World Health Health Organisation, over 1.5 billion people live with hearing impairment globally which amounts to an annual cost of US$ 980 billion to global economy.Around 63 million people in India are suffering from Significant Auditory Impairment which places the estimated prevalence at 6.3% in the Indian population. Children suffering from hearing loss learn to communicate very late and are often given no schooling in developing countries like India. Due to poor investment in healthcare, hearing aids continue to be inaccessible to a large population.
Effective and affordable hearing aids manufactured in India have been the need of decade and would be a great push for Indian healthcare status and healthcare industry serving the purpose of Aatmanirbhar Bharat.
Digital Signal Processing based techniques have proven to be very cost-efficient and fast in solving the problem of speech processing as compared to other ML based techniques. Our goal is to use Digital Signal Processing techniques to implement a hearing aid which would amplify and de-noise the speech signals.

2 Our Approach using DSP Methods

Our approach using DSP techniques are as follows:

- We have tried a series of techniques to de-noise our speech signals and evaluate them against MSE (quality of de-noising), computational cost, time-efficiency and real-time applicability.
- We initially used **Wiener filters** as our primary de-noising filter. However, we were not getting the results that we expected from this filter. Hence, we looked into **Discrete Wavelet Transforms**.
- Using the discrete wavelet transform, we decompose a signal into a set of mutually orthogonal wavelet basis functions. The DWT is invertible, so that the original signal can be completely recovered from its DWT representation. As the DWT transform is very costly and can't be applied in real-time we look into **Short Term Fourier Transform** (STFT).

- We use STFT and spectral subtraction techniques to enable real-time and cost-efficient de-noising. An attenuation map is computed based on the noise-spectrum in a small window of signal. The original signal is restored by computing the inverse-STFT, lastly a Butterworth low pass filter is added to further improve the quality of audio.

3 Wiener filter

Wiener filter is a local adaptive filter that aims to eliminate noise via averaging the signal over a certain window frame. The Wiener gain [2] per sample in a window frame is defined as:

$$\text{Wiener Gain} = \frac{\text{Apriori SNR}}{\text{Apriori SNR } +1}$$

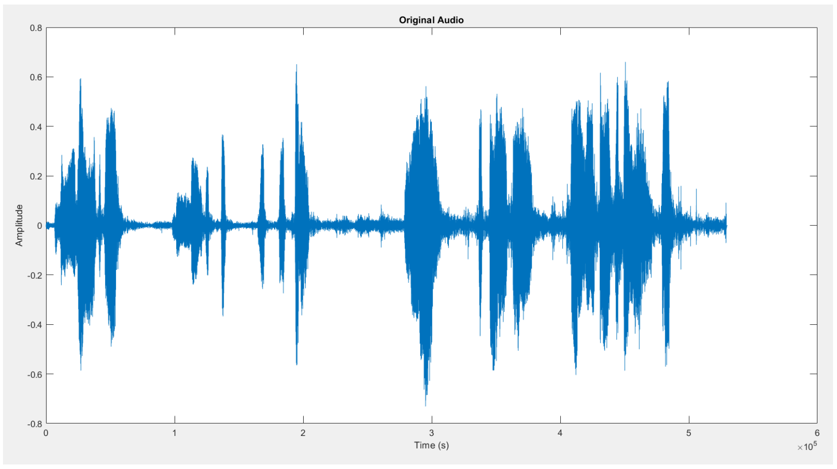
where Apriori SNR is the estimated Signal to Noise Ratio after N samples. After calculating the gain, it is multiplied to the entire noisy signal to mitigate the noise. If the Apriori SNR is high (Speech is dominating Noise), the gain would be close to 1 and the signal would not be attenuated too much, leading to the actual speech being passing through this filter. If the Apriori SNR is low (Noise is dominating Speech), the gain would be close to 0 and the noise would be supressed. For the next window frame, the filter updates its SNR as:

$$\text{New Apriori SNR} = \alpha * \text{(Previous Apriori SNR)} + (1-\alpha) * \max(\text{Aposterior SNR -1,0})$$

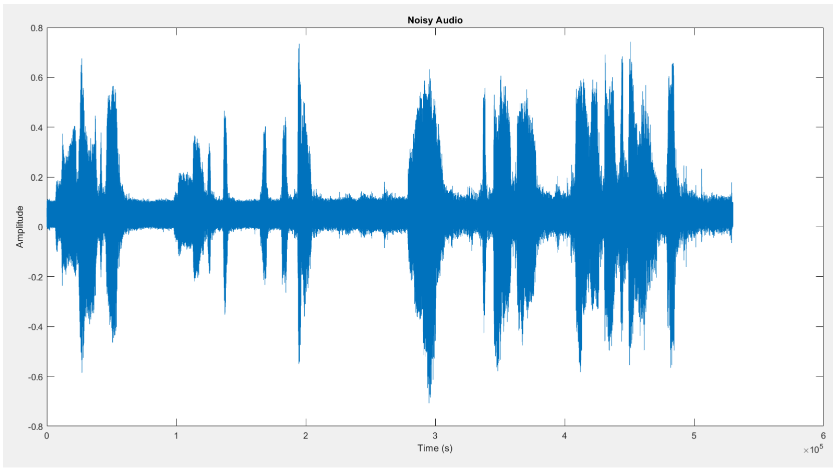
Aposterior SNR can be easily computed after applying the filter over the audio and computing the SNR of the output. α and the inital Noise Power and parameters of this filter which must be tuned to get good denoising. As this filter calculated the SNR over a window frame and continuously updates the SNR , it is called a **Local Adaptive Filter**.

3.1 Our Analysis:

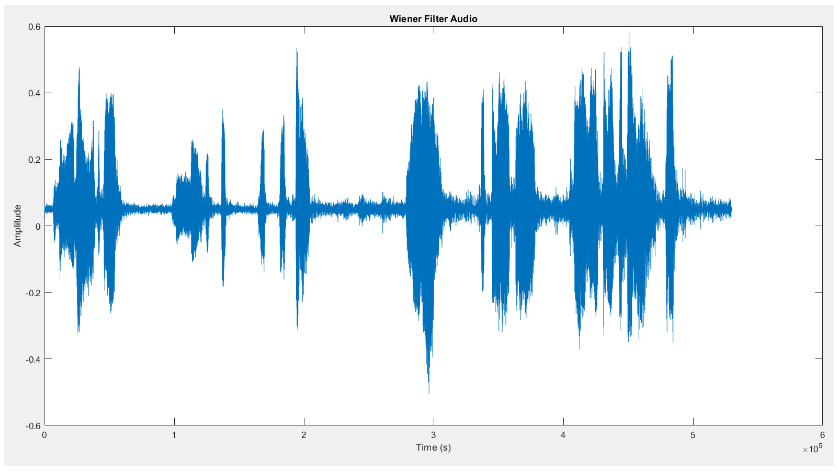
We have used the in-built wiener filter in MATLAB for now but we plan to implement the actual algorithm in MATLAB. The plots of original, noisy audio and the audio after passing through the wiener filter are:



Original audio



Noisy audio



Wiener Filter

We see that the underlying noise-band (± 0.1 band) has been removed by this method and the audio is little clearer. The MSE of this method is 0.0723, which is a great benchmark. However, this filter has some shortcomings which are listed below:

1. The gain of the wiener filter ($\frac{\text{Apriori SNR}}{\text{Apriori SNR} + 1}$)is quite linear and hence the difference between the gain of pure noise and actual speech is less (Eg: If the entire speech is Noise, SNR would be 1 but the wiener gain would be $\frac{1}{2}$ instead of 0)

2. As this filter is applied in the time-domain (point-wise multiplication of samples), actual parts of the speech are also attenuated by this filter, which should ideally be unaffected.

The wiener filter does have some advantages:

1. Calculating the gain per window frame is very fast and hence less computationally expensive

2. As it adapts its estimated SNR, it can adapt to varying Noise conditions and is a more robust choice.

4 Discrete Wavelet Transformation

DWT offers advantage over FFT as it is time localised along with frequency localisation and thus helps in denoising the audio. Wavelets are localised waves which have their energy concentrated in time. The word 'wave' means oscillatory in nature and 'let' means decaying quickly. A wavelet is a type of "brief oscillation". In Figure 1 we can see the difference between a wave and a wavelet.

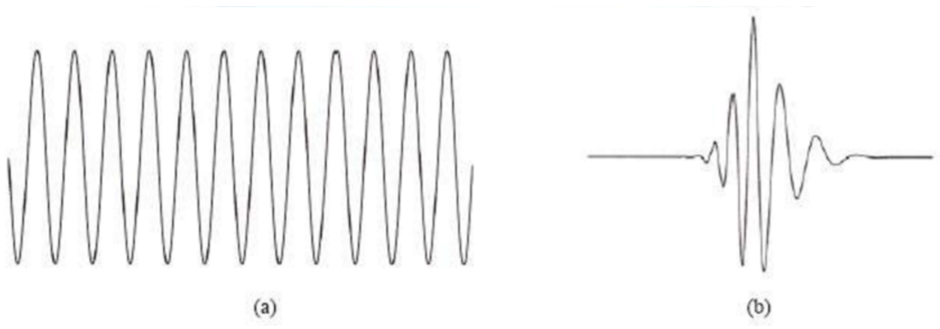


Figure 1: Wave vs Wavelet
Source

There are several different types of wavelets based on application and the requirement. Figure 2 shows various types of popular wavelets. We use wavelets to analyse the frequency spectrum of a signal in a specific window. Along with that wavelet transform is useful in several other ways. The basic formula of wavelet transform is

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t)\psi^*(\frac{t-\tau}{s})dt \tag{1}$$

Here, ψ is the wavelet function, s is the scale parameter (1/frequency) and τ is the translation. The wavelet transform breaks down a signal into a series of basis functions. We can change the width and central frequency of wavelet as we move it across our signal by changing s. This is called scaling.

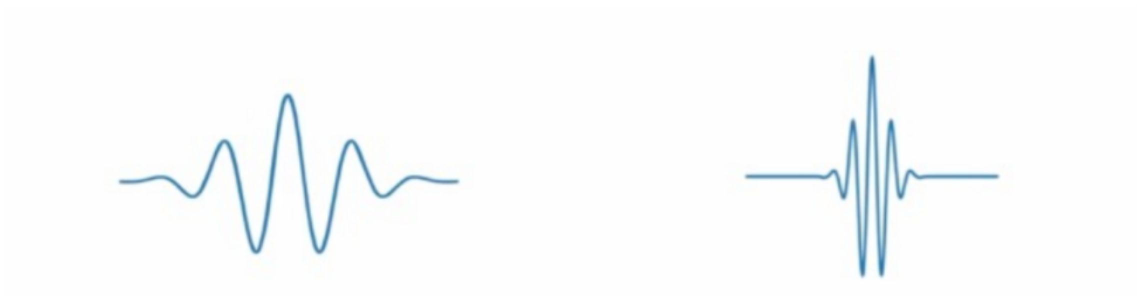


Figure 2: (a) Expanded wavelet is better at resolving low frequency components of the signal with bad time resolution i.e. large values of s. (b) Shrunken wavelet is better at resolving high frequency components of the signal with good time resolution i.e. small values of s

Wavelets are created by moving and scaling a single wavelet termed the Mother wavelet. The low frequency wavelets picks up the low frequency components of the signal and the high frequency wavelets picks up the high frequency components of the signal as we multiply the wavelet to various components of the signal. In a way it is quite similar to Fourier transform, here instead of taking dot product with complex exponential we take dot product with a Mother wavelet function.
The Discrete wavelet transform as the name suggest, transforms a discrete time signal to discrete wavelet representation. The commonly used mother wavelets are Haar Wavelet, Daubechies Wavelet. Daubechies Wavelets are referred as db n wavelet, where n denotes the number of zero-crossings of a wavelet.

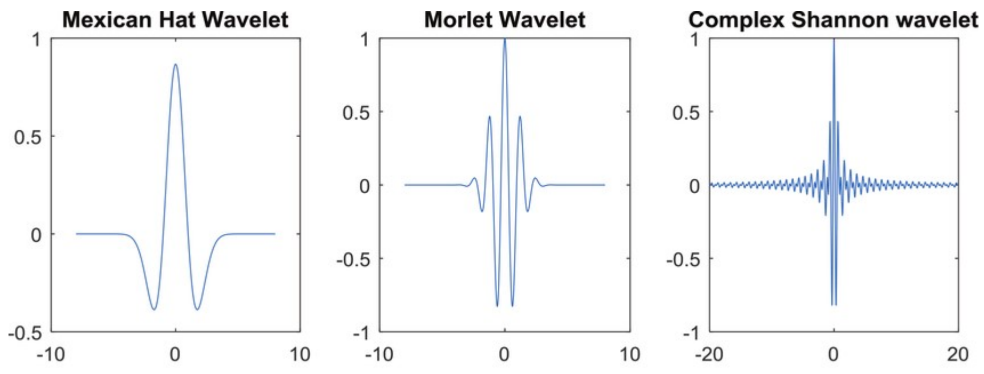


Figure 3: Different Classes of Wavelets
Source

In DWT, we keep s and τ discrete and in the powers of 2 (dyadic) to make the analysis efficient and more accurate.

$$D[a,b] = \frac{1}{\sqrt{b}} \sum_{m=0}^{p-1} f[t_m]\psi[\frac{t_m - a}{b}]$$

$$a = k2^{-j}\ b = 2^{-j}$$

Computationally, Discrete wavelet transform is computed using **Multilevel-Decomposition**. The signal is passed through low-pass and high-pass filters. The low pass filter passes the low frequency components and rejects the high frequency components. In the fig 4, A_i represent our approximation coefficients and D_j represent our detailed coefficients. The low-pass coefficients i.e. approximations coefficients are iteratively filtered i.e. they keep going through low-pass filers at each level rejecting the high-frequency components. Similar, process is done for detailed coefficients (high-pass portions). Thus, at each **decomposition level**, the total number of coefficients (appproximation + detailed) are halved. This is called the decimated-discrete wavelet transform. So, after passing through a fixed number of decomposition levels (number of levels is our choice), we end up with a set of approximation and detailed coefficients.

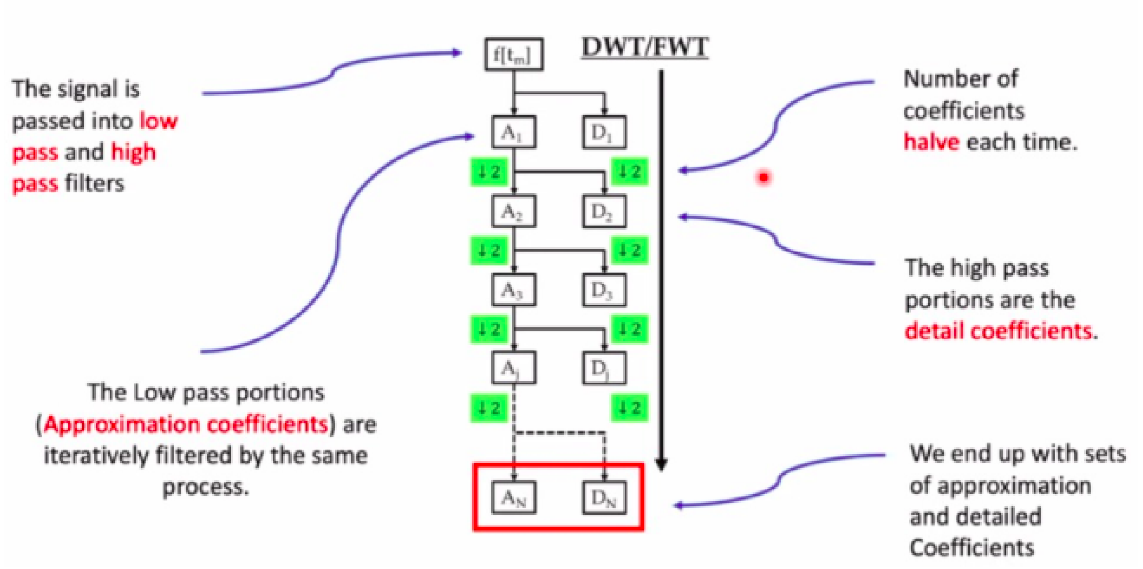

f[t_m]
DWT/FWT
The signal is passed into low pass and high pass filters
A1
D1
↓2
A2
D2
A3
D3
Aj
Dj
AN
DN
Number of coefficients halve each time.
The high pass portions are the detail coefficients.
The Low pass portions (Approximation coefficients) are iteratively filtered by the same process.
We end up with sets of approximation and detailed Coefficients

Figure 4: Working of Discrete Wavelet Transform

Using the wavelet transform we can decompose the real signal, remove its noise part and then recompose the original signal from there. Typically the coefficients corresponding to noise are smaller than the ones corresponding to original signal, thus, for speech de-noising, while reconstructing the signal from it's DWT coefficients we reject coefficients corresponding to noise (which are decided by some threshold fine-tuned by us) and reconstruct the signal from remaining coefficients. We can see an example of signal de-noising by wavelet transform thresholding in the Figure 3. Thus due to the frequency localised and time localised property of wavelet transform it is being adopted for various applications. We shall be using Discrete Wavelet transform in our Application assignment for cleaning of the audio signal.

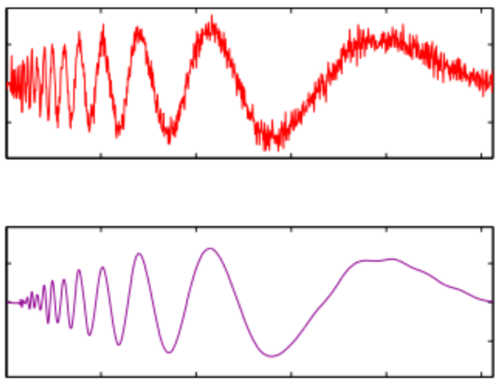
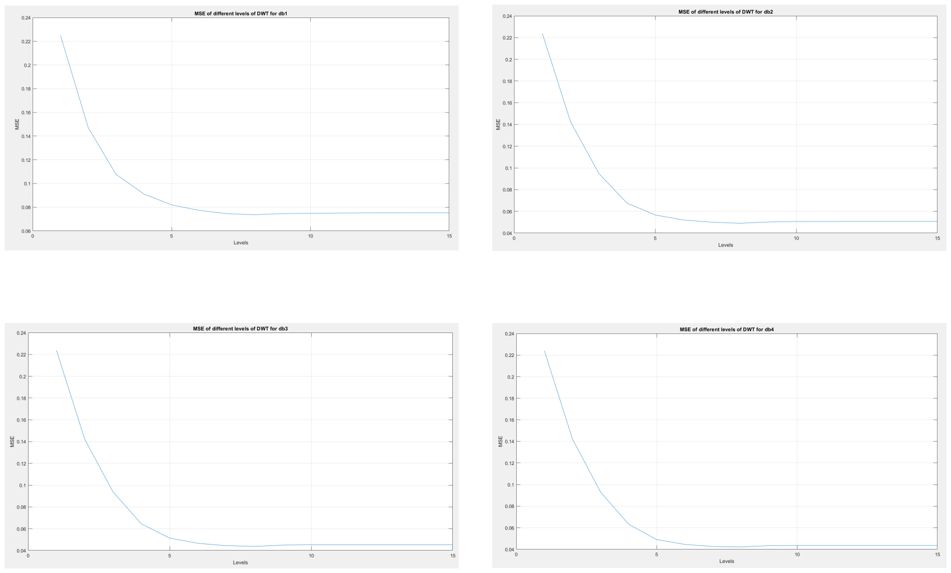


Figure 5: Denoising with Wavelets
Source

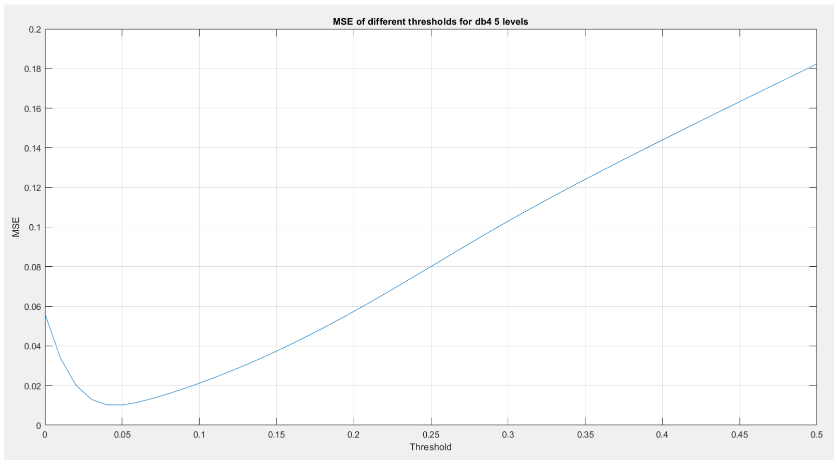
4.1 Our Analysis:

We have used Discrete Wavelet Transform to de-noise the corrupted audio. To choose the number of levels, we plotted the MSE of different levels for 4 different Mother Wavelets (db1 (Haar), db2, db3, db4). The plots are shown below (for threshold value was chosen to be 0.18):



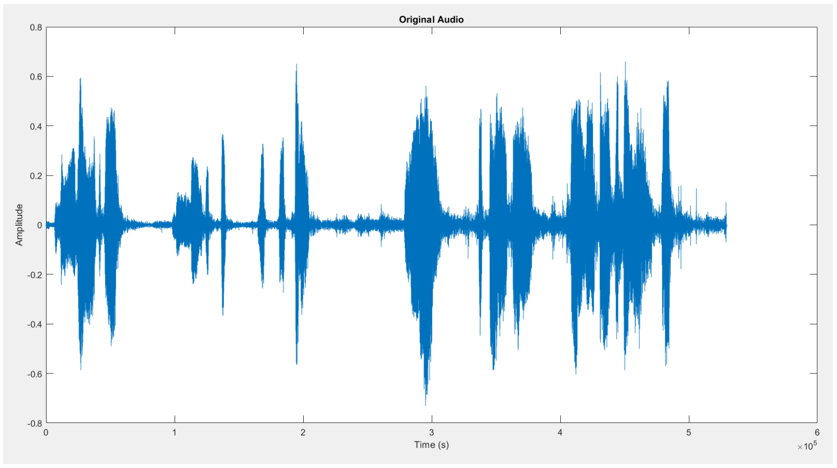
Plots of MSE vs Levels for different Mother Wavelets

Based on the above plots, decomposition of the signal after 5 levels does not show much improvement, and increases the vanishing moments after db4 shows little improvement. We now choose 5 Levels and db4 (Debauchies Wavelet with 4 vanishing moments) Mother Wavelet for our analysis. We also vary the threshold value and plot the MSE vs threshold values for the given noisy signal:

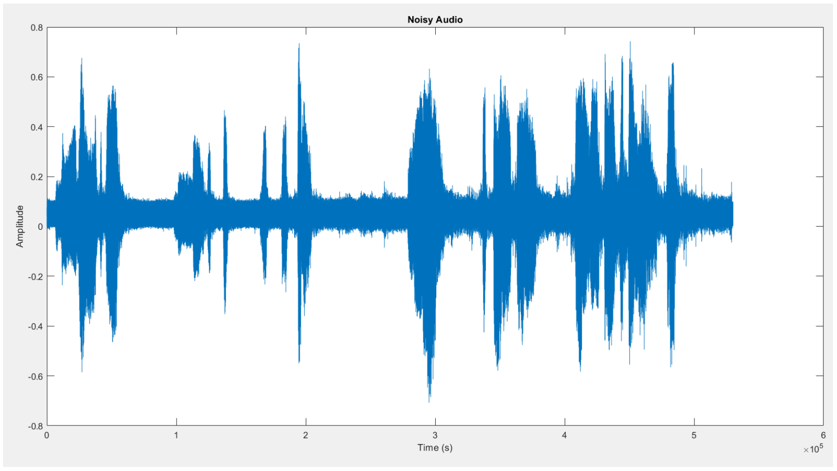


MSE vs Threshold values

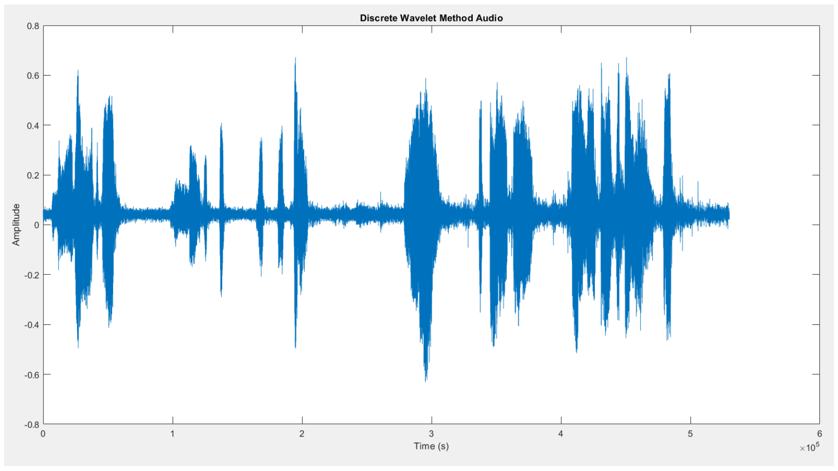
Based on the above plot, we see that a threshold value of 0.05 yields the lowest MSE. The original, noisy and the denoised audios are shown below:



Original audio



Noisy audio



Discrete Wavelet Transform Method

We see that the underlying noise present (± 0.1 band) has been removed with this method. The MSE incurred by this method is 0.0102, which is much better than the wiener filter MSE. The audio recovered is much clearer and this method is capable of removing most of the noise. It is also highly tunable and more fine-tuning can yield better results. However, its disadvantages are:

1. It is computationally expensive to constantly find the DWT coefficients
2. This method cannot adapt to changing noise if we fix the threshold value, and fine-tuning the threshold value is quite challenging.
3. This method also requires the entire signal to be present in order to compute the DWT coefficients, which is not possible for real-time de-noising.

This method, however, gives us insight for our next filter. Recall the formula for wavelet transform :

$$F(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(t)\psi^*(\frac{t-\tau}{s})dt$$

Instead of convolving the signal from $-\infty$ to ∞, we can compute the integral over a very small window frame. This is essentially the idea of Short Time Fourier Transform, which we will see next.

5 Spectral Subtraction using Short-Time Fourier Transform (STFT)

5.1 Short-Time Fourier Transform

The STFT was developed to overcome the poor time resolution of the Fourier Transform. It gives us a time-frequency representation of the signal. Broadly speaking, with STFT we assume some portion of the non-stationary signal is stationary. We then take a Fourier Transform of each stationary portion along signal and add them up. In fig 8, we divide the signal into stationary parts i.e. each part has a constant frequency although the frequency of overall signal is increasing.

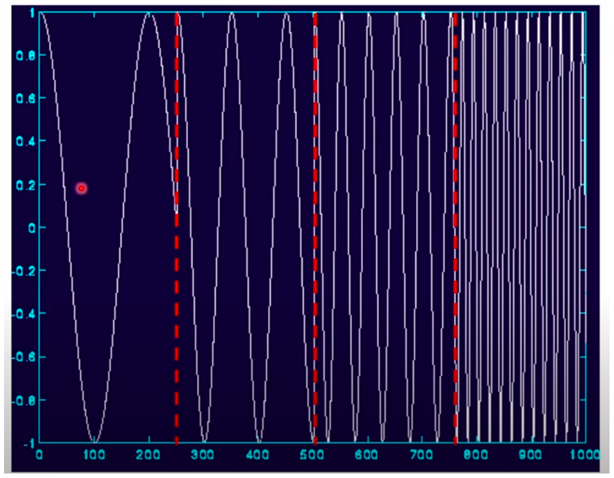


Figure 8: Different stationary sections of a signal [3]

We take a window function of fixed length and move it from start to end and take a FT at each stationary section. Our window function is a rectangle which is zero outside the given interval, so when we multiply the window function to our signal, we only get the signal in that interval and rest is zero which allows us to take Fourier transform of that given section i.e. STFT.

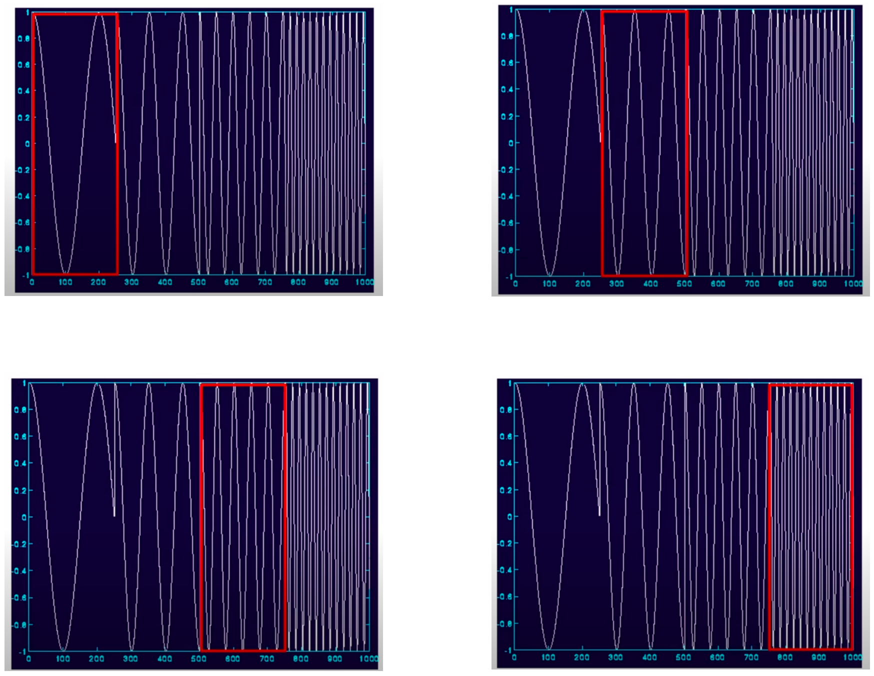


Figure 10: Moving window across various sections in STFT

Mathematically, STFT is given by

$$F(\tau, w) = \int_{-\infty}^{\infty} f(t)w(t-\tau)e^{-iwt}dt$$

where w is our window function which can be translated in time by our translation parameter τ. We see that STFT offers us both time and frequency localisation where as FT only gives frequency localisation.

$$STFT: \quad F(\tau, w) = \int_{-\infty}^{\infty} f(t)w(t-\tau)e^{-iwt}dt$$

$$FT: \quad F(w) = \int_{-\infty}^{\infty} f(t)e^{-iwt}dt$$

But as the window length is finite, so frequency resolution decreases in STFT. Fig. 11 shows frequency-time plane for STFT. We see that at all times for all frequencies we get fixed frequency and time resolutions given by the square regions. Frequency and time resolution are related by following law of physics: *We can't know what frequencies exist at what time instance, but we can know what frequency bands exist at what time intervals*

$$\Delta t \Delta f \geq \frac{1}{4\pi}$$

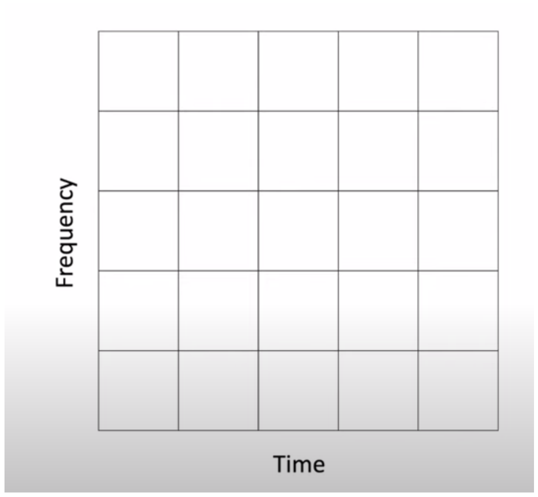


Figure 11: Fixed frequency and time-resolution in STFT

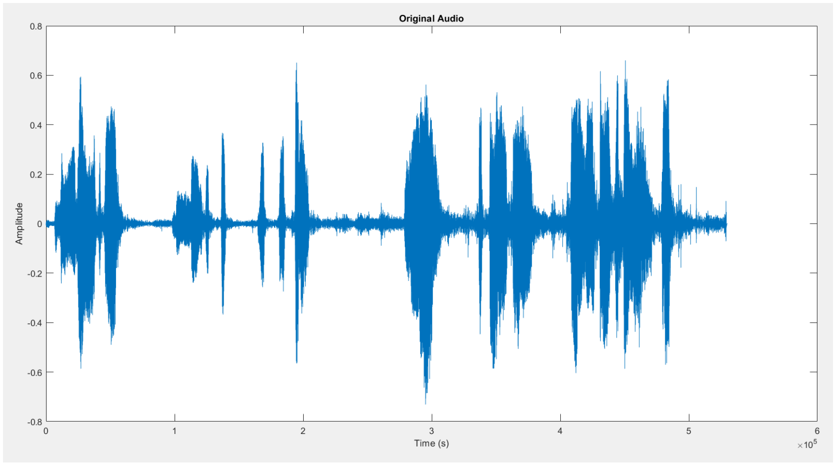
Therefore, a narrow window would give good time resolution but poor frequency resolution and a wide window will give poor time resolution but good frequency resolution. It is generally observed that low frequency components often last a long period of time so a high frequency resolution is required where as high frequency components often appear as short bursts, invoking the need for a higher time resolution. This problem of resolution is solved with Discrete Wavelet Transform but the computational cost incurred in DWT are much higher than in the STFT, thus by using some additional thresholding algorithms we try to improve the results obtained via STFT.

5.2 The Vuvuzela sound denoising algorithm and our analysis

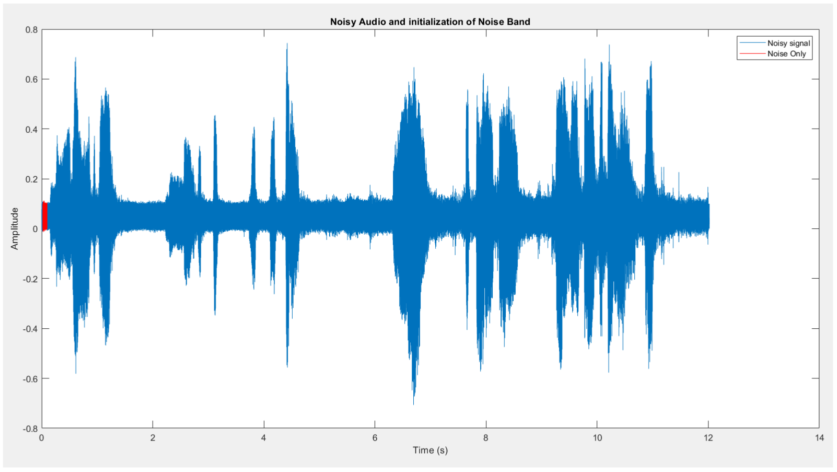
Based on the spectrum of the vuvuzela sound [1], this denoising technique simply computes an attenuation map in the time-frequency domain. It first assumes a noise region in order to compute the noise power for estimating the SNR. We assume the first few 100ms to be the noise region, which is usually a good estimate. The formula for the attentuation map is given by:

$$\text{Att. Map} = \max((1 - \lambda(\frac{1}{SNR+1})^{\beta_1}))^{\beta_2}, 0);$$

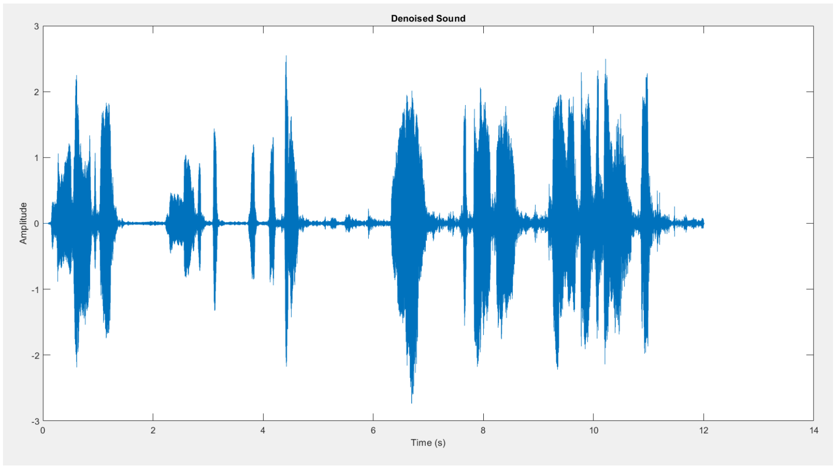
Here, λ, β_1, and β_2 are parameters to be tuned. Intuitively, when SNR is high (Speech), $\frac{1}{SNR+1}$ would be very low and (1 - $\lambda*(\frac{1}{SNR+1})^{\beta_1}$) would be mostly a positive quantity. On the other hand, when SNR is low (Noise), $\frac{1}{SNR+1}$ would be very high and (1 - $\lambda * (\frac{1}{SNR+1})^{\beta_1}$) would be mostly a negative quantity, which would then be clipped to 0. This attentuation map is very non-linear when compared to the Wiener filter and hence we can expect better results when using this method. The SNR is computed by Short Time Fourier Transform (Computing the Fourier Transform of a window frame independent of other window frames). This method is very efficient and can also be implemented in real time. This attentuation map is multiplied with the signal and audio signal is restored by computing the inverse STFT. The plots of original, noisy and the restored audios are given below:



Original audio



Noisy audio with initial Noise Band

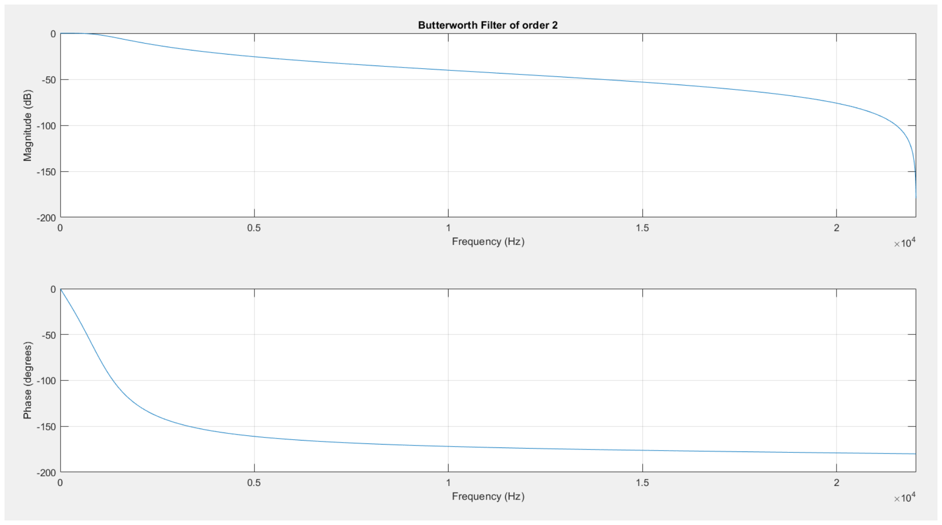


STFT Method

The MSE yielded by this method is 0.0104 after tuning $\beta_1(0.5)$, $\beta_2(1)$ and $\lambda(2)$, which is very close to DWT method. This method, however, can work in real-time and is less computationally expensive than DWT. This method also works well and the audio recovered is more comprehensible than the noisy audio.

5.3 Proposed Lowpass Filter

The drawback of the above method, however, is the final audio, although much clearer than the original noisy audio has a ringing effect in the background,possibly due to the highly non-linear of the attentuation map. To overcome this, we propose a lowpass filter after this particular filter. As speech signals and image signals mostly have all their information in the **phase** and not in the **magnitude**, we will be using the **Butterworth filter** as it has the most linear phase. The cutoff frequency of the Butterworth Filter is set to 2400Hz (Normal human speech has a fundamental frequency of 200Hz and we choose the cutoff frequency to be around 12 octaves higher) and the order is chosen to be 2 considering the computational power. The frequency response of the Butterworth filter is:

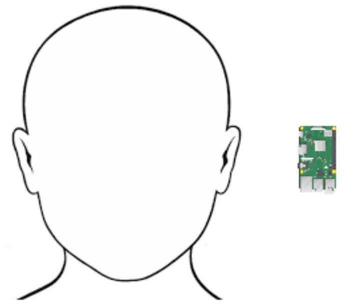


Magnitude and Phase response of Butterworth Filter

After applying this lowpass filter, the ringing noise in the background is now completely removed and denoising to a great extent has been achieved. The link to the audios are given in the Code section of this report.

6 Hardware Implementation

Hearing aid devices need to small, compact and light weight as they need to be used on a daily basis. Thus hardware implementation of our project pose several challenges. To proceed with hardware implementation the following path is proposed; initially the system can be tested with Raspberry Pi as it is compatible with MATLAB. The Raspberry Pi chip comes with sizes as small as 65mm x 35mm which would help building up towards problems that would come when implementing a small hardware package.



Size Comparison: Raspberry Pi Zero 2 vs Human

Once this model is functioning we can then move on to converting the code to HDL and building FPGA which will further reduce the cost and size of the module. Very low cost ADC and DAC with sampling rate of 48Khz can be then incorporated to complete the hearing aid design.

7 Conclusions and Relevance to Aatmanirbhar Bharat

We explored various DSP techniques to efficiently de-noise the speech audio signals for the hearing-aid. Firstly, an adaptive and fast low-pass filter was implemented using Weiner filter algorithm which gave an MSE of 0.0723. Then, a more advance and computationally expensive method - DWT is implemented. DWT gave a very low MSE of 0.0102 which was out-standing but could neither give real-time functioning nor was adaptive. Finally, spectral subtraction algorithm using STFT and Vuvuzela sound de-noising algorithm is implemented which gave an MSE of 0.0104. This not only works fast in real-time but also has a low-computational cost, thus, achieving the goal of devising a low cost algorithm for hearing aids for **Atmanirbhar Bharat**. Fig. 12 shows the final architecture of our real-time de-noising algorithm. We add a Butterworth low-pass filter which greatly improves the quality of audio.

DSP Algorithm	**Mean Square Error**
Weiner Filter	0.0723
Discrete Wavelet Transform	0.0102
STFT and Vuvuzela Algorithm	0.0104

Table 1: Comparision of different methods

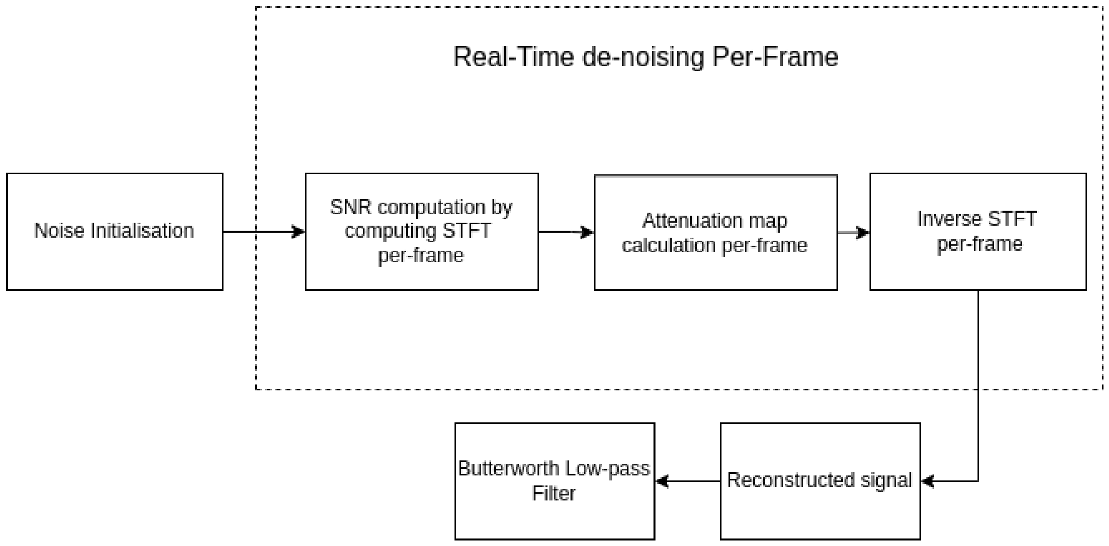

Real-Time de-noising Per-Frame
Noise Initialisation
SNR computation by computing STFT per-frame
Attenuation map calculation per-frame
Inverse STFT per-frame
Reconstructed signal
Butterworth Low-pass Filter

Figure 12: Architecture of final real-time de-noising system designed

Our solution is cost-effective, fast and practical. If used in actual **Made in India** hearing aids, this can help millions of children with hearing impairment access quality education and help the elderly by giving them access to cheap and good quality hearing aids, making India more **Aatmanirbhar** in health and education sector. For future works, one can improve upon the algorithm by automating the process of fine-tuning parameters. The hardware realisation of the algorithm also remains a task for future work.

8 Code:

Github link for code and audio files can be found here (run them on MATLAB): Github link

References

[1] Choqueuse Vincent (2022). Vuvuzela sound denoising algorithm (https://www.mathworks.com/matlabcentral/fileexchange/27912-vuvuzela-sound-denoising-algorithm), MATLAB Central File Exchange. Retrieved April 18, 2022.

[2] Y. Ephraim and D. Malah, “Speech enhancement using a minimum mean square error short-time spectral amplitude estimator,” IEEE. Transactions in Acoust., Speech, Signal Process., vol. 32, no. 6, pp. 1109–1121, Dec. 1984

[3] A nice explanation of STFT, WT and DWT by Andrew Nicoll: https://www.youtube.com/watch?v=kuuUaqAjeoA