Signal Processing Project Report

Team Name:

DeNocifiers

Team:

K Sri Rama Rathan Reddy - 2022102072

Ch Vishnu Priya - 2022102023

S Rohit - 2022102001

Part 1 - Echo creation:

Let x[n] be the input signal and y[n] be the output signal.

So, the output signal would be $y[n] = x[n] + \sum_{i=1}^{N} a_i \cdot x[n - D_i]$ where N is the number of echos to be added to the Input Signal, $a_i (0 < a_i \leq 1)$ is the scaling factor of amplitude in the attenuation. and D_i is the Delay at which The new Echo starts. N is the number of echos we want to add to the Signal.

Here $N,\ a_i\ and\ D_i$ can be changed.

Now taking the z transform of the above Equation, we get:

$$Y = X + \sum_{i=1}^{N} a_i \cdot X \cdot z^{-D_i}$$

$$H(z) = \frac{Y}{X} = 1 + \sum_{i=1}^{N} a_i \cdot z^{-D_i}$$

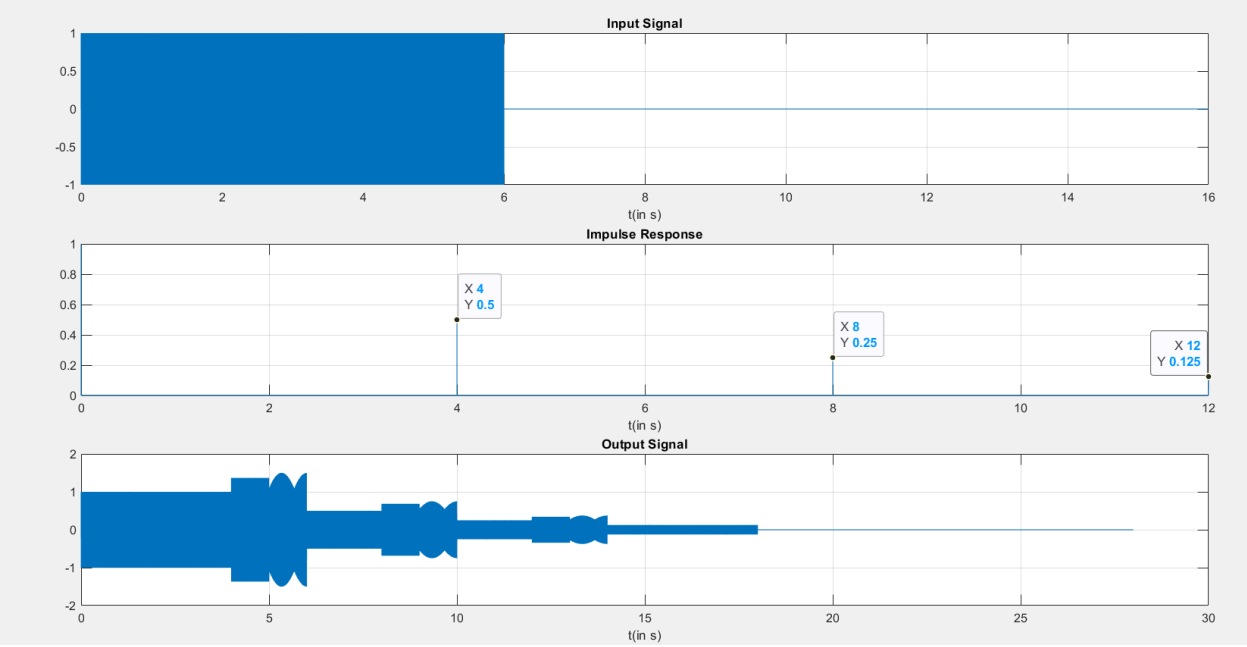
Let the numerator be b and denominator be a.

So, we get :

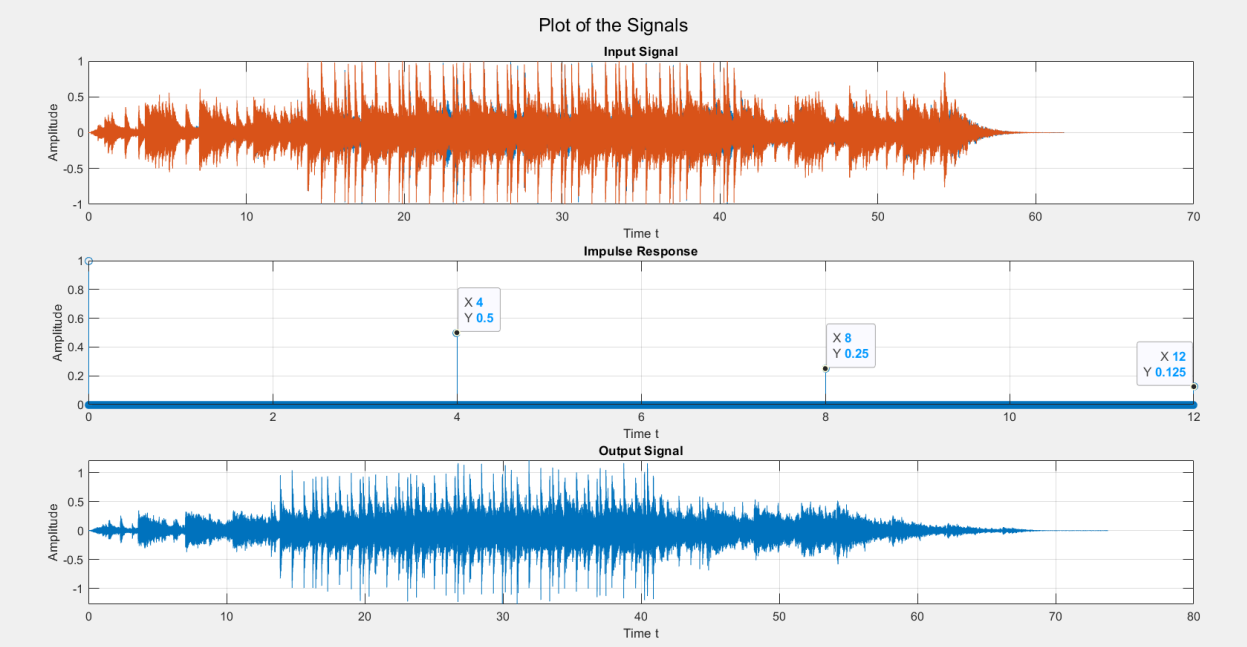
$$b = 1 + \sum_{i=1}^{N} a_i \cdot z^{-D_i}\ and\ a = 1$$

using the impz(b,a) function from MATLAB, we get the impulse response of the system H(z) and let it be h[n].

Now, Convolving the Impulse response h[n] with input signal x[n] using conv(x,y,”full”) command we get the output y[n] which is the signal with Echo effect.



Plot For q1.wav



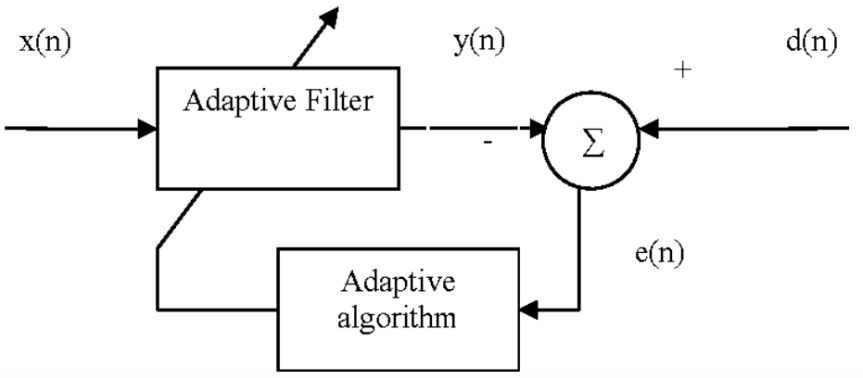
Plot for q1_hard.wav

The above plots are for $N = 3;\ a_i = (0.5)^2; D_i = 176400 (as\ t = 4sec)\ for\ i = 1, 2, 3$

Part 2 - Cancel the echo:

Adaptive Filters Algorithm are used for the Echo cancellation.

An adaptive filter is a filter that self-adjusts its transfer function according to an optimization algorithm driven by an error signal.



Adaptive Filter

Adaptive filters are self-learning filters, whereby an FIR or IIR filter is designed based on the characteristics of the input signal to adapt to its environment. The environment will be defined by the input signal x[n] and desired signal d[n]. Adaptive filters have self-regulation and tracking capabilities. An adaptive filter finds its essence in applications such as Echo Cancellation, Noise Cancellation, System Identification, and many others.

There are many algorithms like LMS(Least Mean Square), NLMS (Normalized Least Mean Square), and RLS(Recursive Least Square) algorithms.

The basis of the LMS and NLMS algorithm is to minimize the expectation of the square of the error of Output and Desired in every iteration so that the weights can be updated for calculating the error. so that the output reaches the desired signal.

LMS Algorithm:

The LMS algorithm is by far the most widely used in adaptive filtering. It is a type of adaptive filter known as a stochastic gradient-based algorithm as it utilizes the gradient vector of the filter tap weights to converge on the optimal wiener solution. In each iteration of the algorithm, the filter taps weights are updated as per Equation (3) where w[n] represents the adaptive filter weight vector at time n, x[n] represents time-delayed input signal samples, e[n] represents error signal to be minimized and μ represents step size or convergence factor.

Output: $y[n] = w^h \cdot x[n]$

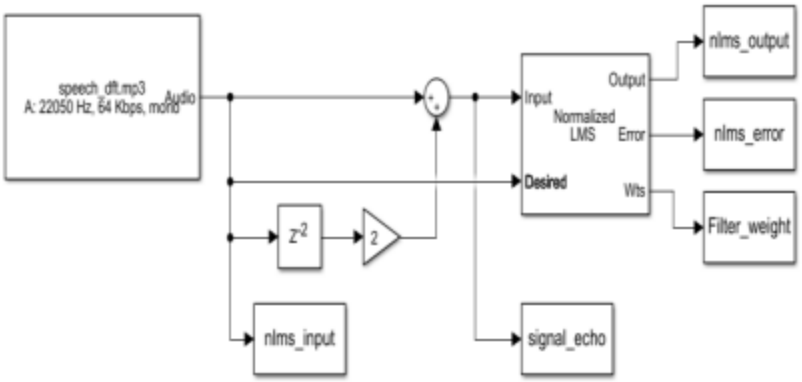
Error: $e[n] = d[n] - y[n]$

weight: $w[n+1] = w[n] + \mu x[n] \cdot e[n]$

The step size parameter controls the influence of the updating factor. If μ is chosen to be very small then the algorithm converges very slowly. A large value of μ may lead to a faster convergence but the adaptive filter becomes less stable around the minimum value and its output diverges.

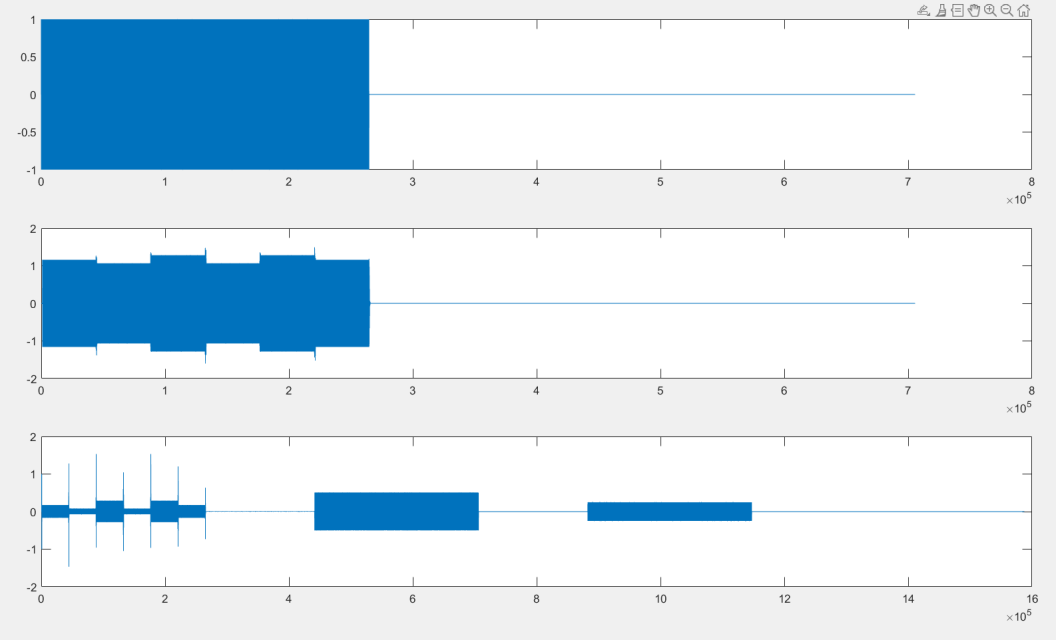
NLMS Algorithm:

One of the primary disadvantages of the LMS algorithm is having a fixed step size parameter for every iteration. In the LMS algorithm the weight adjustment is directly proportional to the amplitude of input vector samples therefore, when the vector x [n] is large, the LMS suffers from a gradient noise amplification problem. To overcome this problem, the adjustment applied to the weight vector at each iteration is normalized. The normalized least mean square algorithm (NLMS) is an extension of the LMS algorithm which bypasses this issue by calculating maximum step size μ[n]. This step size is proportional to the inverse of the total expected energy of the instantaneous values of the coefficients of the input vector x[n]. This sum of the expected energies of the input samples is also equivalent to the dot product of the input vector with itself, and the trace of input vectors auto-correlation matrix, R. In each iteration of the NLMS algorithm, the filter tap weights are updated as per Equation.

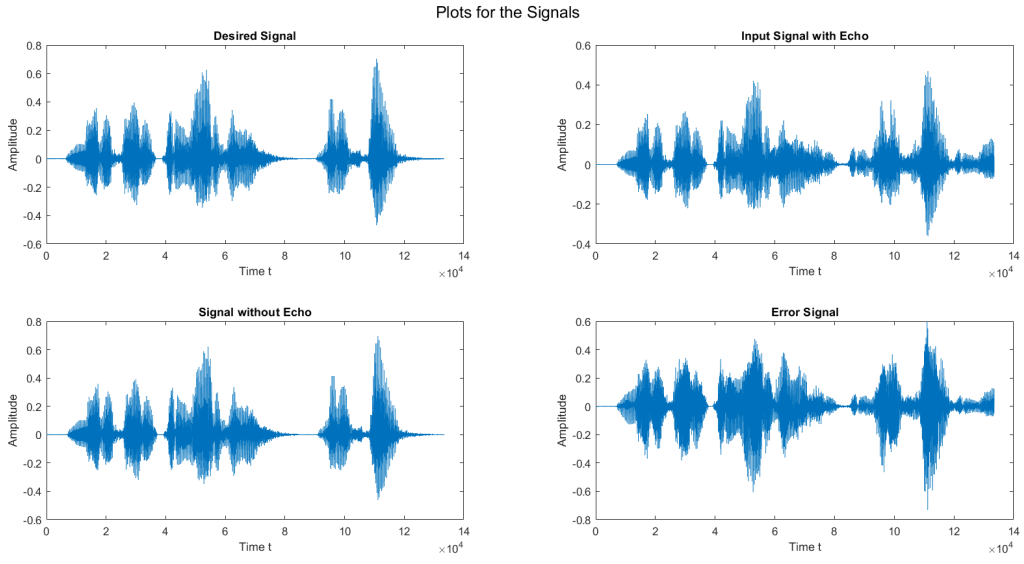


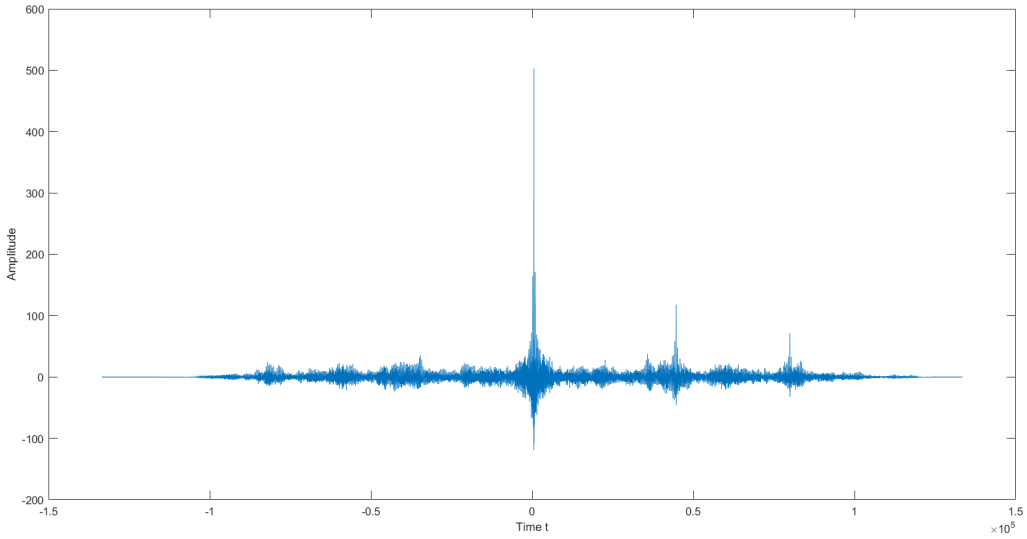
Block Diagram for NLMS Algorithm

Plots for Desired signal as q1.wav and Input signal as echo signal generated from q1.wav from Part1



Plots for Desired signal as q1_hard.wav and Input signal as echo signal generated from q1.wav from Part1





Cross Co-rrellation of Original Signal and Output Signal

Part 3: What is this noise?

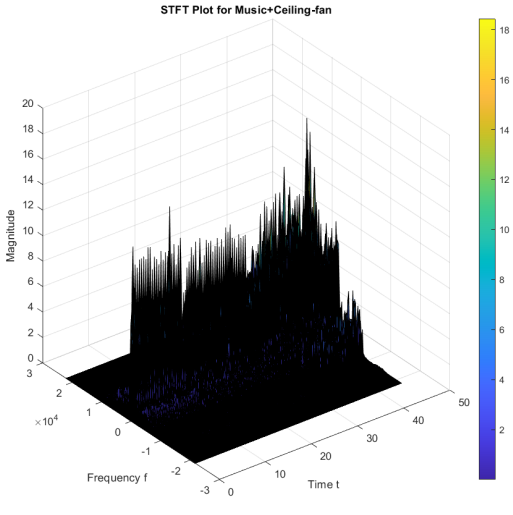
Frequency-domain representations such as the DTFT and the DFT are useful, they both are obtained by summing the time function x[n] from $-\infty$ to ∞. This means that the DTFT and DFT describe frequency components in the signal averaged over all times.

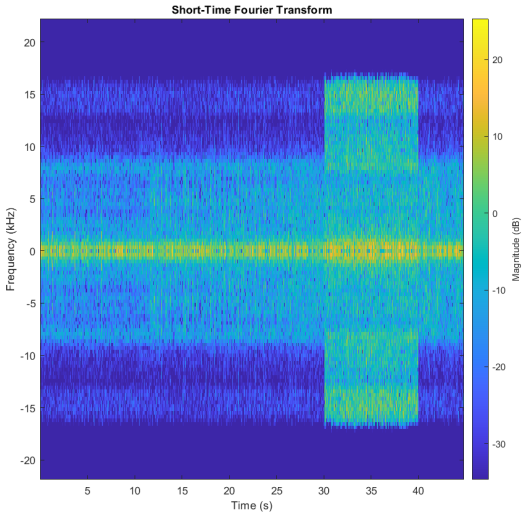
But for signals like Music and speech, we are interested in analyzing the variation of Frequency components concerning time.

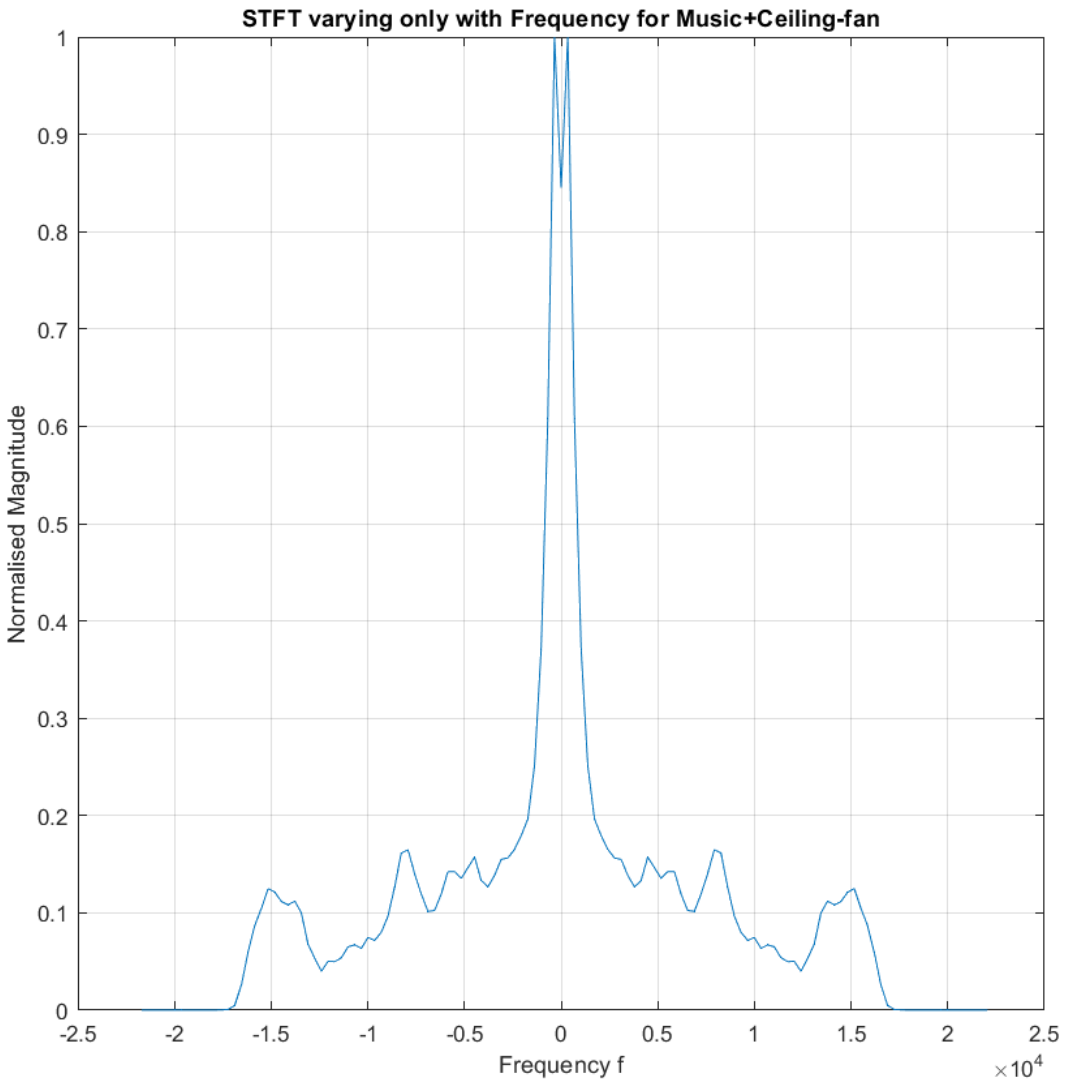
For this, we use a Short-time Fourier transform.

STFT Plots:

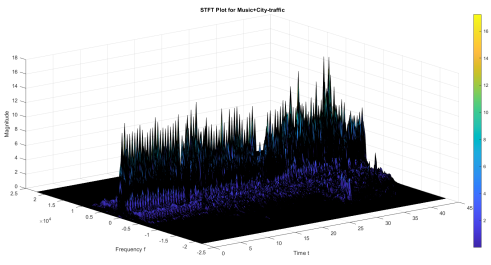
Plots for Ceiling Fan:

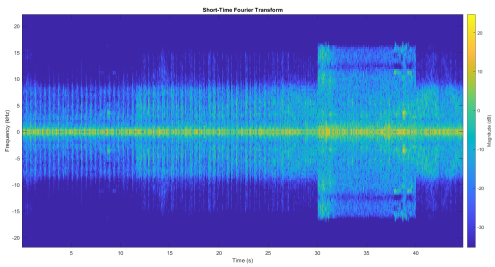

STFT Plot for Music+Ceiling-fan
Magnitude
Frequency f
Time t
×10^4

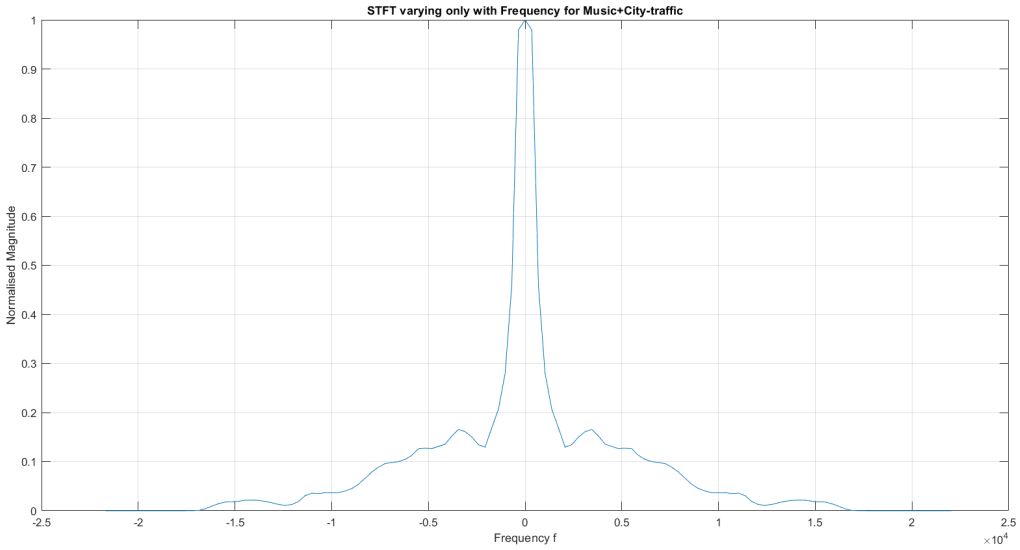

Short-Time Fourier Transform
Frequency (kHz)
Time (s)
Magnitude (dB)



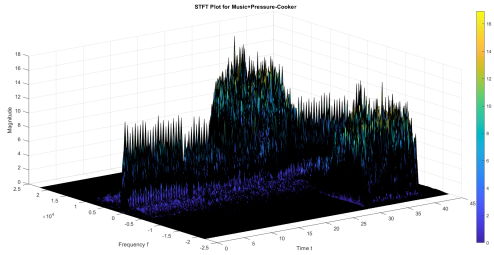
Plots for City Traffic:

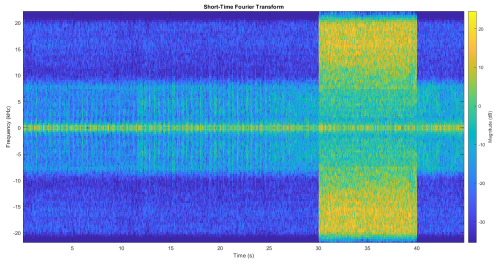


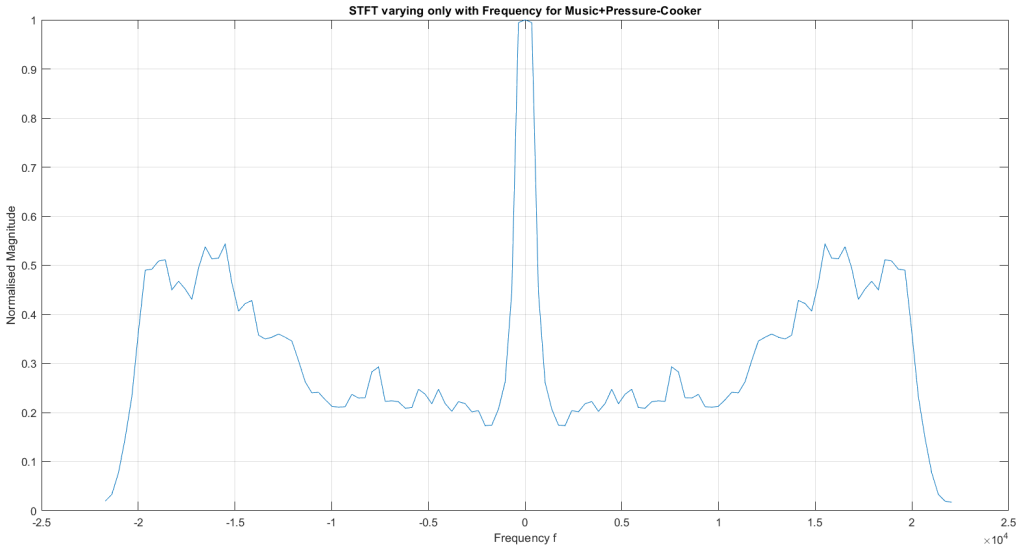




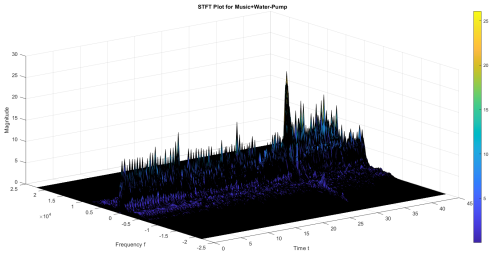
Plots for Pressure Cooker:

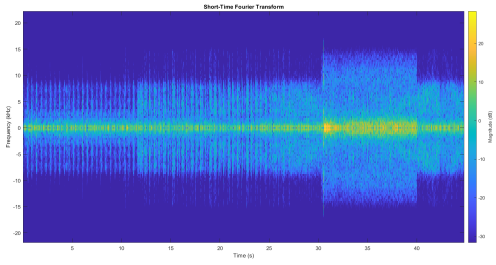


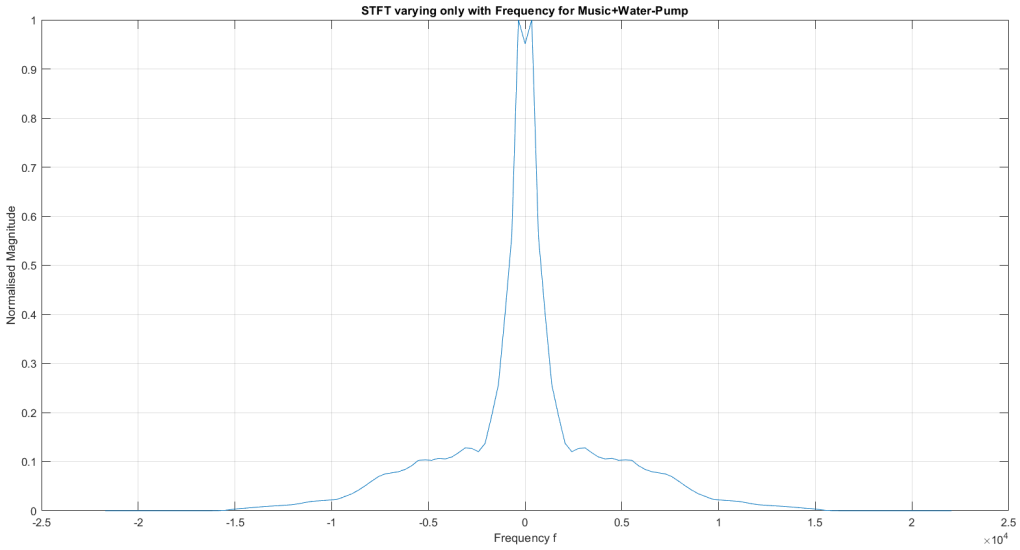




Plots for Water Pump:







In DFT and DTFT, we generally average over the entire time range which is very large. Noise Generally has a very small Amplitude and averaging over such a large time might not consider the effect of noise would be much less in DFT and DTFT.

The STFT considers only a short-duration segment of a longer signal and computes its Fourier transform which accounts for the effect of noise much better than FFT and STFT.

Taking these short time intervals is achieved by using Filters.

We make a normalized magnitude of s versus frequency plot and compare all the reference files to the given input files using Root Mean Squared Error. and print the Noise using that.

We compare high-frequency terms so that only noise is compared irrespective of the input music.

We can Also Separate Noise from the Noise of the Given Input Signal by using a 3D STFT plot.

From the plots, we can observe that only some parts have high-frequency components that correspond to noise we can isolate that part of the signal and analyze using the Fourier transform to find the noise.

Then this separated noise can be used to find noise by using the dynamic time warping method.