**Signal Processing – EC5201**

**Final Course Project**

Team – **SRH**

Team Members – **Harshvardhan Singh** (*2022112004*),

**Rupak Antani** (*2022102045*), **Soham Vaishnav** (*2022112002*)

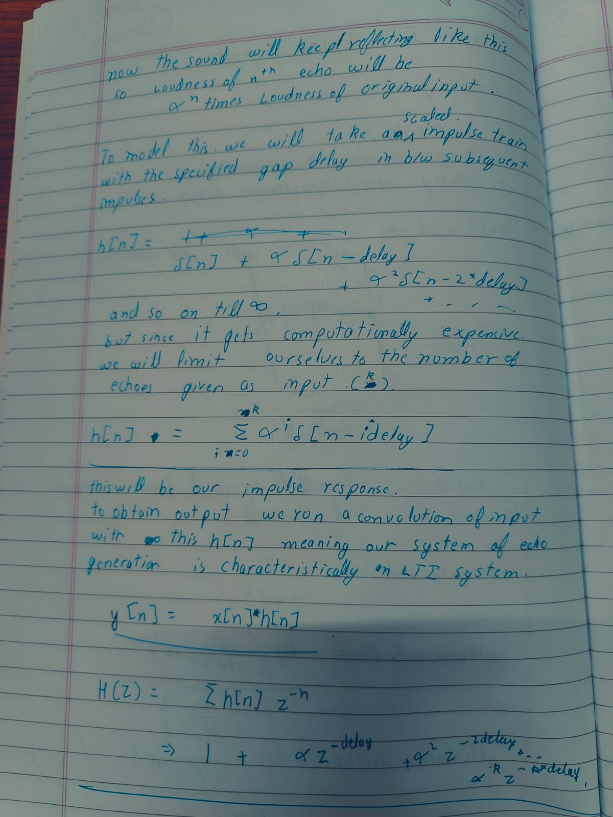
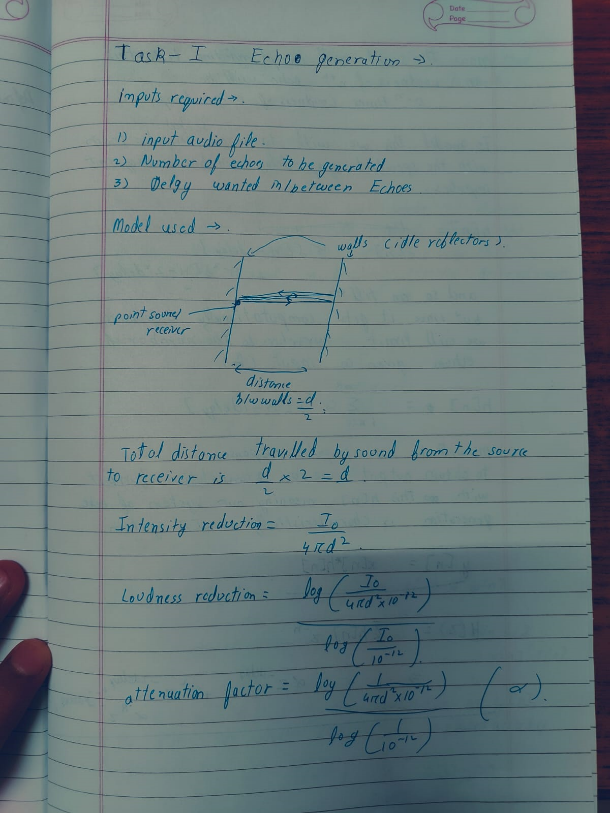
**Task 1 – Echo Creation**

**Aim:** To introduce echo in a given audio\speech signal

**Inputs:** Hindi\_2s.wav and IPL audiofiles given in the viva and test files given beforehand.According to code, we require number of echoes to be produced and desired delay between two echoes.

**Problem solving approach:**

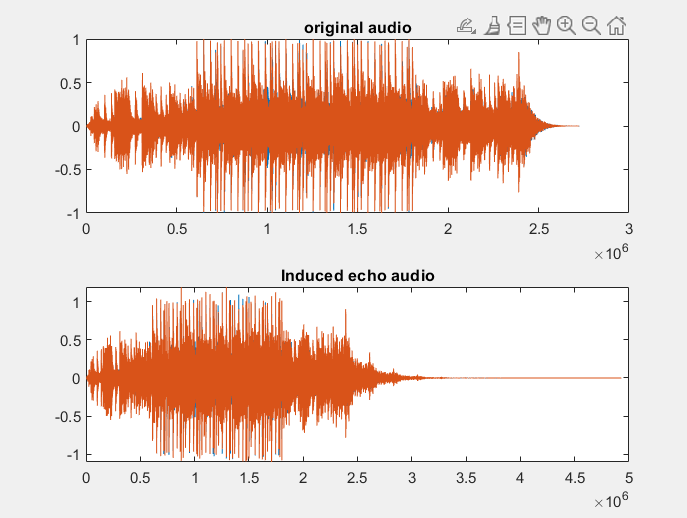
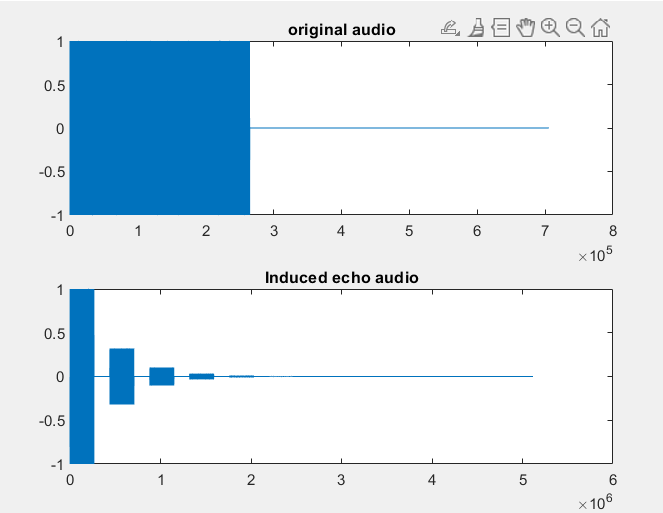
Echo is fundamentally addition of a delayed and attenuated version of the source signal over and over again for a sustained period of time. This occurs due to the sound being reflected off a surface and the attenuation occurs as a result of loss of energy at the contact or during travel of the signal. Here is the modelling of echo -



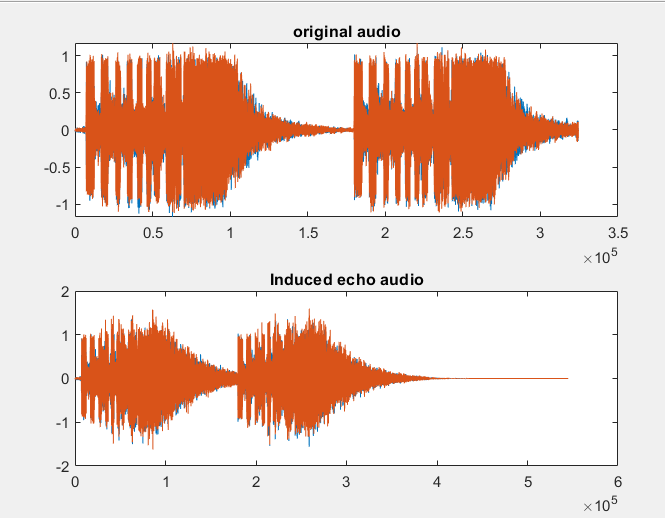
*Images showing explanation and modelling of the algorithm for task-1*

**Results – Plots**

The following images show the outputs of the files given as an input (mentioned above) by following the approach shown in the above images. Clearly, visually although it might seem difficult to observe the echo, but actually it has been induced quite successfully.

*Images showing the implementation on the given test audio files q1.wav and q1\_hard.wav*

The viva included performing the task on the speech and music files one of which is shown below –



*Image showing echo being induced in the iconic IPL tune*

**Task 2 – Cancel the Echo**

**Aim:** To cancel the echo from the signal – be it uniform or non-uniform

**Inputs:** Input audio files such as med\_echo and max\_echo. Other than that we require fs as an input. Rest is derived internally by one or the other function.

Echo cancellation is one of the most challenging signal processing tasks because it involves identification and separation of over-lapping signals induced due to the phenomenon of echo.

* **Consideration**

1. Echo creation as we have seen occurs when the delayed and attenuated versions of source signal are heard simultaneously. This may or may not include overlapping and therefore the difficulty of the task differs for each case. For overlapping echoes, the problem becomes more complex and it involves implementing sophisticated methods, whereas, for the non-overlapping case it is relatively simpler.
2. Another major factor that comes into play is regarding delays which can be of two types – uniform and non-uniform. By uniform and non-uniform delays we refer to the periodicity in echoes, i.e. the duration between two echoes.

It is important to be mindful of the fact that in real life situations there are many reflectors and thus the echoes arrive at different intervals in time.

These factors play a crucial role in deciding the approach that we take to tackle the issue of cancelling the echo.

* **Approach(es)**

There are different approaches for different situations – uniform and non-uniform.

1. **Uniform Delays**

We can model the uniform delay situation as follows:

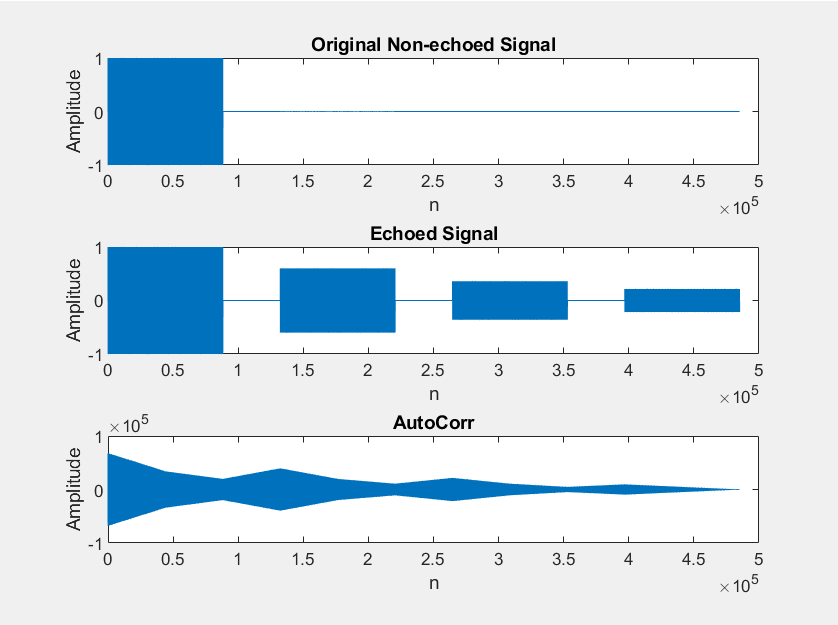
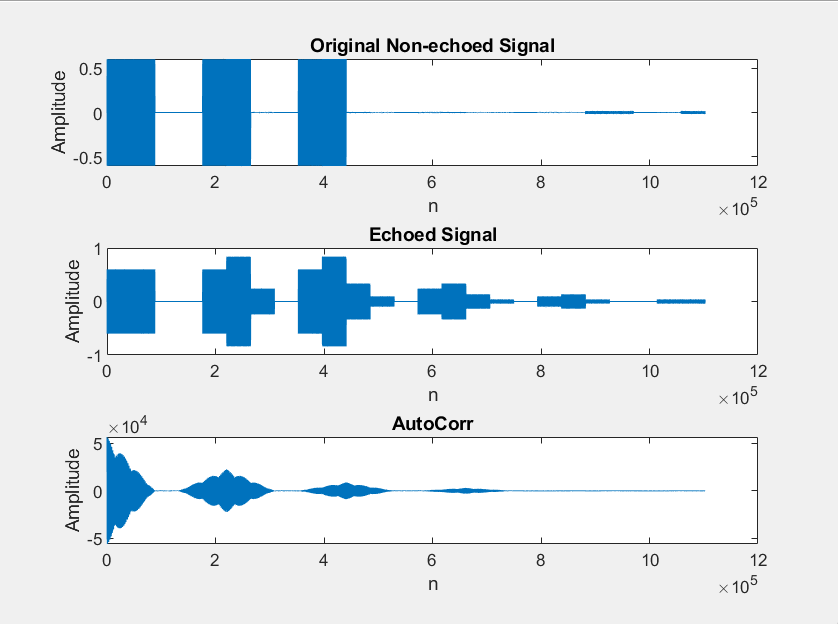
Which can be re-written in a more compact form as follows –

It is important to note that we have assumed uniform attenuation across the length of the echoed signal, and by that we mean that the attenuation factor decays in a geometric progression.

**Steps taken to remove echo –**

1. Take auto-correlation of the given echoed signal using **xcorr** which will render us peaks at the places where the original signal is repeated

This shows that as and when *l* is equal to the delay N, the signal will be the square of itself multiplied by the attenuation factor for that echo

*Images showing peaks where the echoes occur*

1. Extract the peaks by using the **findpeaks** function in MATLAB and setting the minimum peak distance to be 2\*fs\*0.1-1 units of time. This particular distance was chosen keeping in mind the physics of acoustics which says that if an echo is to be heard and perceived by a human then it has to occur after at least 0.1s have passes between the sound being released by the source and it being received by the human ear (considering speed of sound in air is 343m/s)
2. Filter those peaks to get the (nearly) exact peaks in the auto-correlation graph where the echoes have occurred
3. Since the delays and attenuation are both uniform, we take the ratio of the amplitude of the first peak to the maximum value of the auto-correlation graph to find attenuation α and multiply it with the version of the original signal delayed by the number of units that correspond to the first peak using **delayseq**

Where x is the original non-echoed signal and N is the position where the first peak occurs.

1. **Non-uniform Delays**

Let us model the non-uniform delays as follows –

The assumption of uniform attenuation is carried here as well, else modelling the problem becomes more complex because multiple coefficients are required. Clearly, we cannot form the compact form of the above equation as we did for the uniform delays’ case.

We therefore suggest two possible algorithms to solve the problem and in the process we have been to identify the flaws or constraints in using them. The third method is to solve it using **NLMS-Adaptive filtering**

1. **Top-to-Bottom –** This is a recursive algorithm which starts from the first peak and moves towards the last peak (apparently). Here, while removing each peak, we *induce* new peaks or new echoes due to non-uniform delays. In the first iteration we remove all the original peaks and then start locating for the new induced peaks. This involves taking auto-correlation of the signal in every iteration.

**Problems involved –** Firstly, this method does not guarantee any convergence in terms of removal of peaks and secondly, it is computationally expensive due to calculation of auto-correlation in every iteration which may be many.

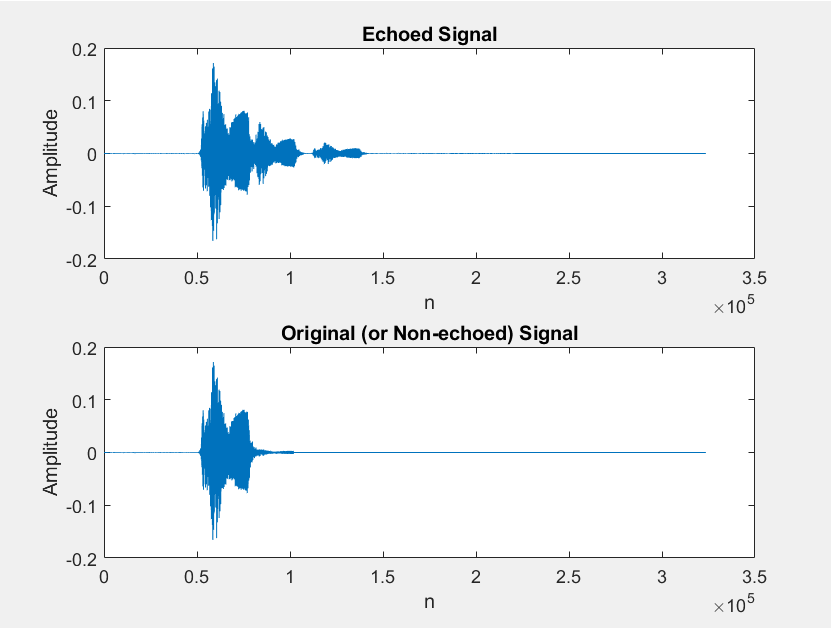
1. **Bottom-to-Top –** This method involves starting from the *measurably* last peak and moving towards the first peak. The logic behind this is that all the echoes induced due to the last peak in any iteration will definitely be more insignificant than the original peaks and so the signal amplitude can therefore be made 0 thereafter.

**Problems involved –** Firstly, if the number of peaks is too many then it becomes computationally expensive and secondly, if the original signal is too long such that it has more than one echo within its length then while making the later peaks 0, we may lose data!

For ease, let’s consider the signal with just two echoes –

First iteration beginning from the last –

Therefore, if we see closely, two new echoes have been added at N1+N2 and 2N2 which are definitely later in time that the echo at N2 which was removed. We therefore conveniently make xe[n] equal to 0 beginning from the th instant in time (discrete).

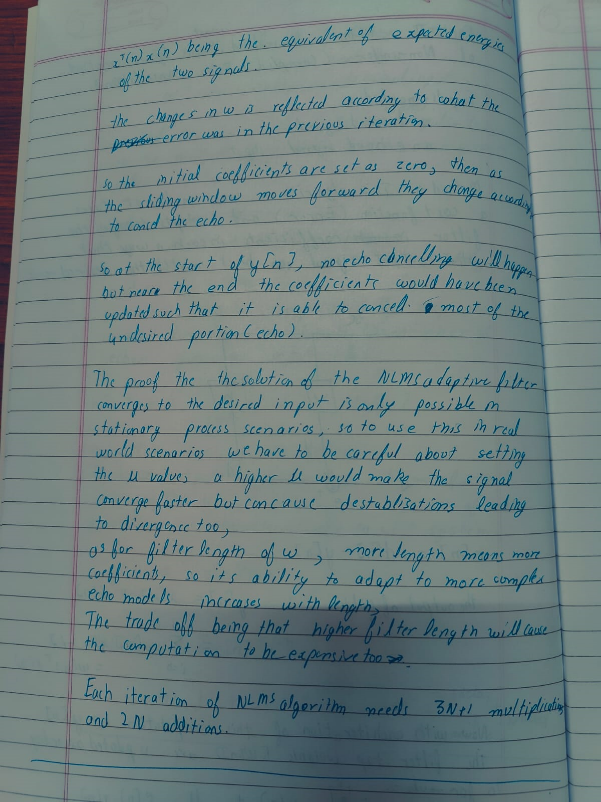
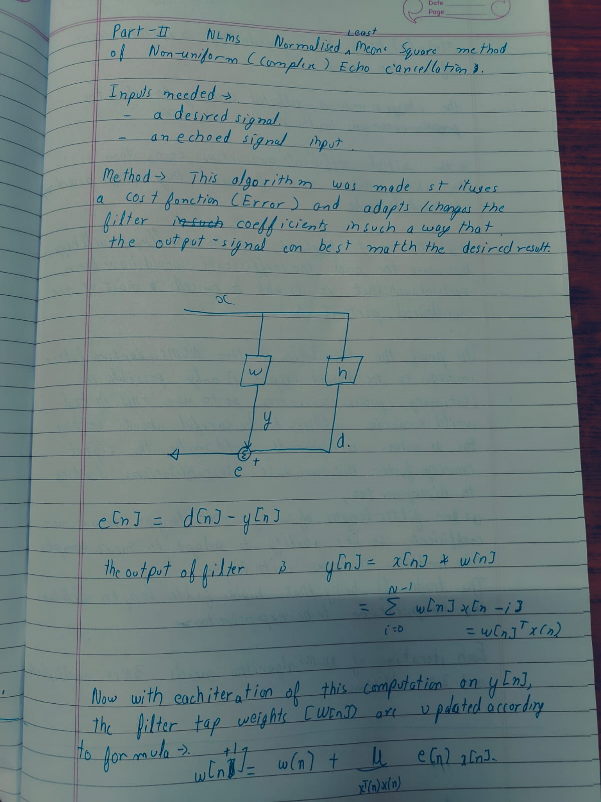


*Image showing filtering of a self-recorded and echoed sample*

*using Bottom-to-Top method*

1. **NLMS – Adaptive Filtering –**

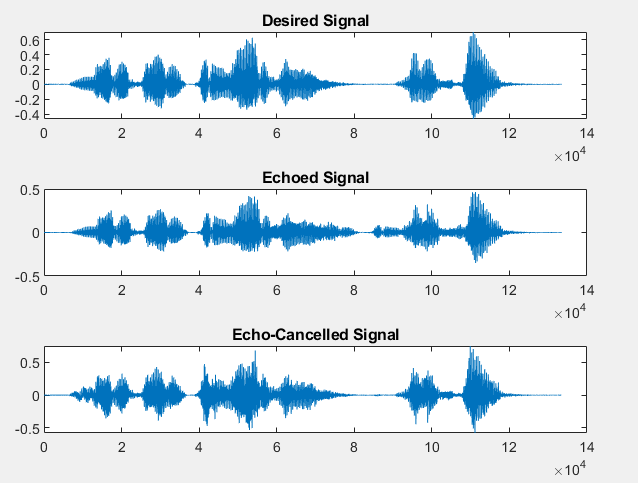
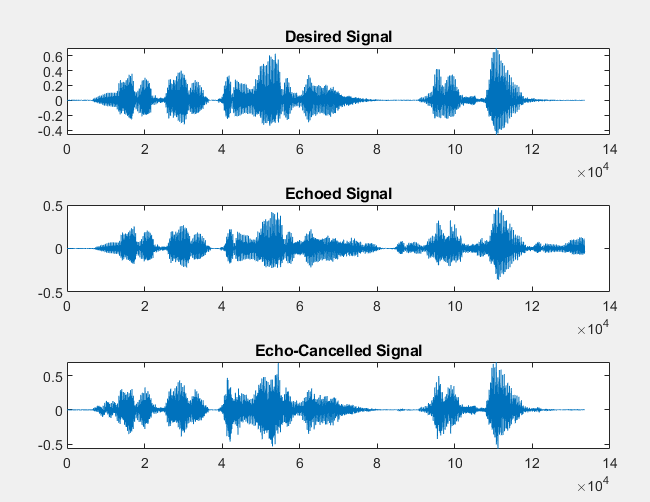
This method has found varied and widespread industrial applications in terms of noise and echo cancellation because of ease of implementation and accuracy. Following is the math and intuition behind its working –



*Image showing the math and intuition behind the working of*

*NLMS Adaptive Filtering*

In the viva we were given two kinds of echoed signals – max\_echo and med\_echo – both of which were speech signals. By using this filter on those we get –



*Images showing removal of echoes in the max\_echo.wav and med\_echo.wav files resp.*

As we can see at the start of the echo cancelled signal it is imperfect since the filter hasn’t updated the coefficients sufficiently yet, but after some samples and iterations the rest of the signal is very close to the desired signal. Therefore, while listening to the files,

**Results –** Shown with the approaches itself

**Task 3 – What is this noise?**

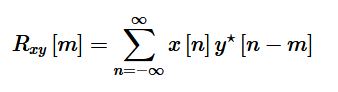
**Aim:** to identify the type of background noise present in music recordings without filtering out the noise from the music. The noise can be of the following four types:

1. Fan
2. Traffic
3. Pressure Cooker
4. Water Pump

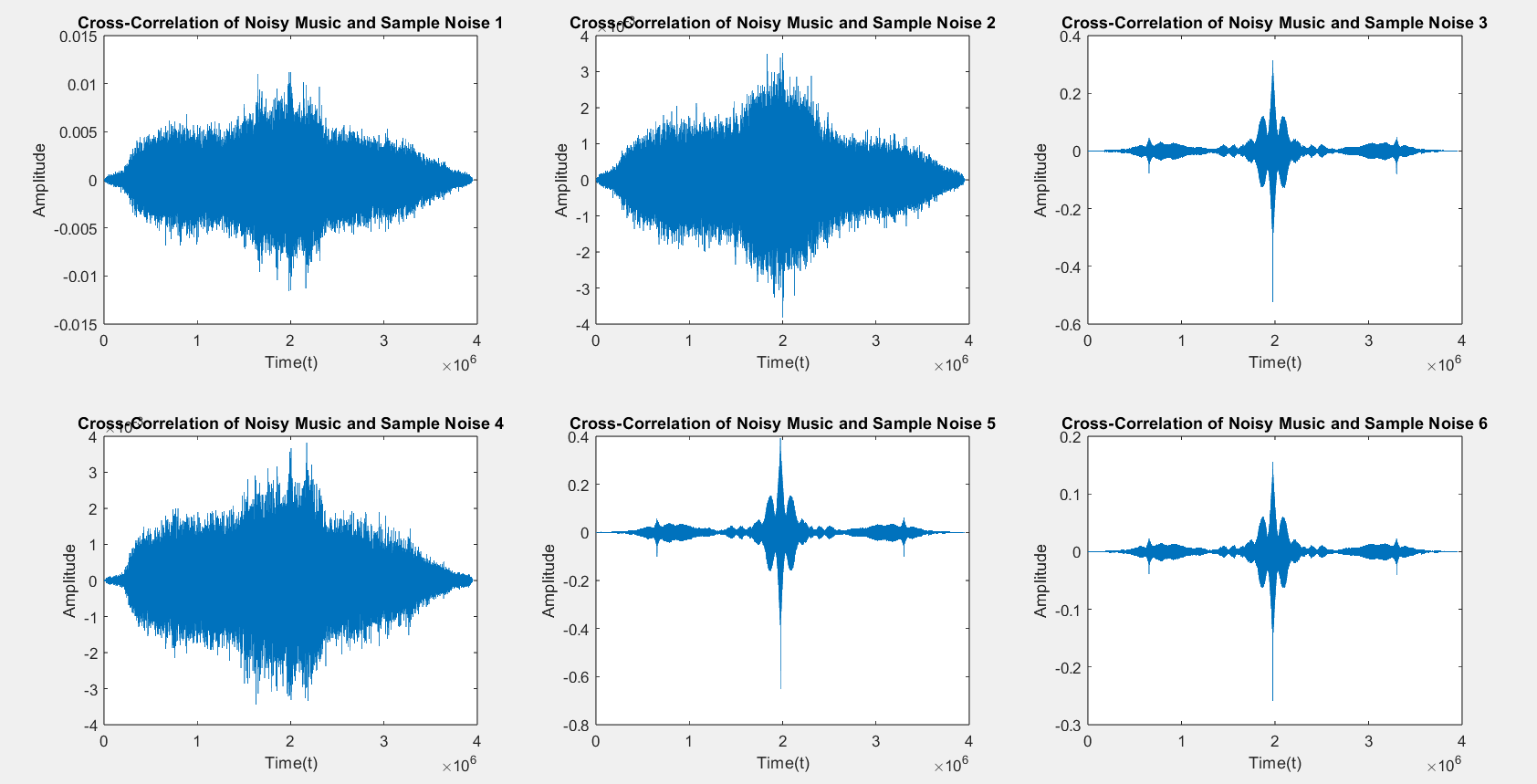
**Inputs:** Just audio files.

**Logic and its Implementation**

In order to classify the different types of noise, we first take the noisy music audio file from which we need to classify the noise as well as a set of six sample noise audio files. These sample audio files have been made by subtracting one type of noise from another. Therefore, they contain various combinations of the four types of noise. We perform the cross-correlation of the given music signal with the set of sample noise signals. Cross – correlation of two signals gives the measure of the similarity between a signal and the delayed form of the other signal. The cross correlation (Rxy[m]) between two discrete signals x[n] and y[n] is given by the formula below:



The more similar two signals are, the greater will be the peak values obtained by their correlation. So, by correlating the music signal and the sample noise signals, we can obtain the extent of similarity between the noise in music signal and the sample noise files. For example, I take the input signal as a music signal with background noise of water pump. On cross - correlating this signal with the set of noisy signals I obtain the plots as shown below:



From the plots, we can observe that we get higher peak values in case of sample noise 3, 5 and 6. This is because the water pump noise is present in those audio signals but not in the rest.

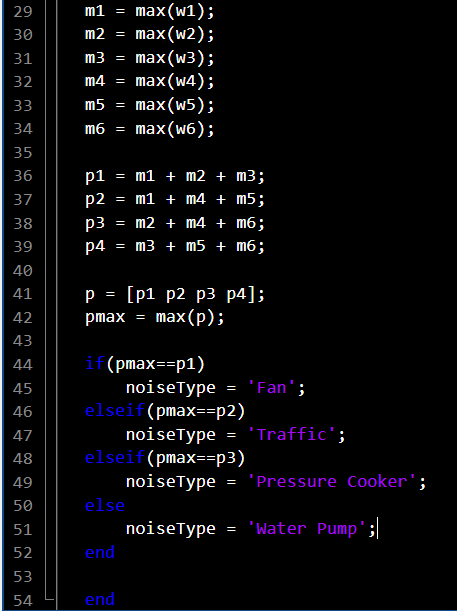
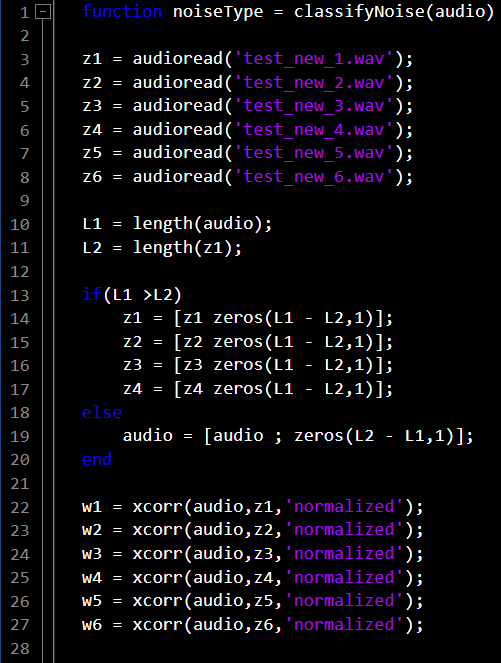
Next, I take the peak or maximum values of each of the cross – correlated outputs. Then, I take four combinations of these peak values based on the combinations of different types of noises present in the set of sample noise audios. For example, in this case I take the combinations of noise signals to be (1,2,3), (1,4,5), (2,4,6) and (3,5,6) because I have made the noise files accordingly. Now, I find the maximum of these four values. Lastly, I compare and find that for which combination am I getting that maximum value and based on that I classify the type of noise.

In the case that I have considered, if the combination (1,2,3) has the highest value, then it corresponds to fan noise, (1,4,5) corresponds to traffic noise, (2,4,6) corresponds to pressure cooker noise and (3,5,6) corresponds to the water pump noise.

This algorithm can be applied to any type of music signal having any one the four possible types of noise.

**Explanation of Code**

**P.T.O**



The **classifyNoise** function is shown above. The ‘audio’ signal which we are taking as input for the function is the noisy music signal which is given to us. We also load the set of six sample noise files which is made up of various combinations of the different types of noise. Then, we cross-correlate the ‘audio’ file with the set of noise signals using the **xcorr ()** function. We normalise the cross - correlation so that the range of the magnitude of cross - correlated signal is between -1 to 1. Next, we find the peak values of each cross – correlated output using the **max ()** function. After this, we take four combinations of these peak values and find the maximum of these values by again using the **max ()** function. Lastly, we match this maximum value with all the four combination values and identify that for which type of noise we are getting this maximum value using the if-else conditions. This will give us the type of noise present in the music audio.

**Results:** Mentioned above with the explanations itself.

**Conclusion –**

We realised that echo creation, removal and identification of noise are some of the most challenging and crucial problems that Signal Processing deals with. To create echo, we had to model the reflector as a system to get proper delays and attenuations. To remove echo, we had to model the system which showed us what the delays are and consequently, where the echo occurs. Along with this, we had to find the attenuation and using that and the delays we were able to eliminate the echoes. The challenges faced thereon were mentioned in the respective sections. Now, to the third part where we classify noise, we used the already given data-set to cross correlate with the new audios and compare different combinations of amplitudes to classify noise.