Course Project

Signal Processing

Aryaman Mahajan (2022102034)
Manas Deshmukh (2022102040)
Yash Nitin Dusane (2022102078)

Convolution of Signals :

The convolution of two signals, often denoted as $x(t)$ and $h(t)$, is a mathematical operation that produces a third signal, usually denoted as $y(t)$. The convolution operation is defined as:

$$y(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t - \tau)\, d\tau$$

In discrete time, for discrete signals $x[n]$ and $h[n]$, the convolution is given by:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \cdot h[n - k]$$

Use of Convolution concept in Echo creation :

We have transfer function as

$$y[n] = x[n] + \alpha x[n-k] + \alpha^2 x[n-2k] + \dots$$

in case of echo creation,

$\therefore$ Taking the Z transform on both sides

$$Y(z) = X(z) + \alpha z^{-k} X(z) + \alpha^2 z^{-2k} X(z) + \dots$$

$$\therefore Y(z) = X(z)\left(1 + \alpha z^{-k} + \alpha^2 z^{-2k} + \dots\right)$$

$$\therefore \frac{Y(z)}{X(z)} = 1 + \alpha z^{-k} + \alpha^2 z^{-2k} + \dots$$

Considering finite no. of echoes (a)

$\therefore \quad H(z) = 1 + \alpha z^{-K} + \alpha^2 z^{-2K} + \ldots + \alpha^a z^{-aK}$

Taking inverse z-transform

$$h[n] = \delta[n] + \alpha\,\delta[n-K] + \alpha^2\delta[n-2k] + \ldots + \alpha^a\delta[n-ak]$$

$\therefore$ h[n] being the transfer function,

As $H(z) = \frac{Y(z)}{X(z)}$

$\therefore \quad Y(z) = X(z) \cdot H(z)$

$\therefore$ From properties of z-transform

$y[n] = x[n] * h[n]$

$\therefore$ We could see use of convolution in getting echoed signal with uniform delay

{ $\alpha \Rightarrow$ Attenuation factor
Assuming α^K times attenuation for K^{th} echoed signal }

Thus in the matlab code :

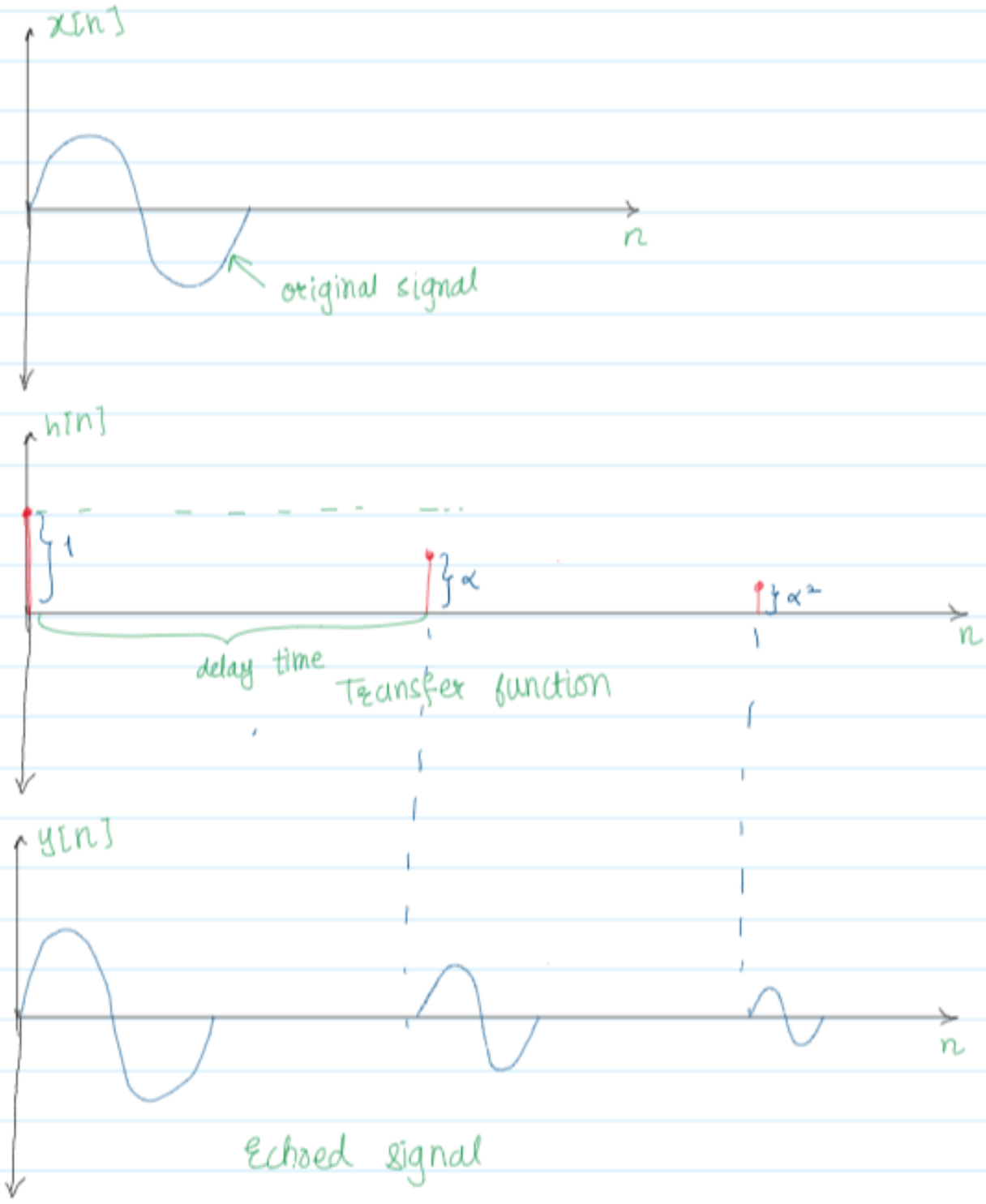
$h[n] = [\,1 \;\; \text{zeroes}(1, d) \;\; \alpha \;\; \text{zeroes}(1, d) \;\; \alpha^2 \ldots]$ (a times)

($1 \uparrow \delta[n]$, $\alpha \uparrow \alpha\delta[n-K]$, $\alpha^2 \uparrow \alpha^2\delta[n-2K]$)

$\Rightarrow$ d = delay time
a = no. of echoes required

$\therefore$ Just convolution of signal with h[n] could give us the required echoed signal

Graphical Representation



⇒ Another Method

If you know the system function,

$$y[n] = x[n] + \alpha x[n-K] + \alpha^2 x[n-2K]$$

Taking z transform, we get

$$H(z) = \frac{1 + \alpha z^{-K} + \alpha^2 z^{-2K}}{1}$$

∴ In form of

$$H(z) = \frac{\sum_{K=0}^{M} b_K z^{-K}}{\sum_{K=0}^{N} a_K z^{-K}}$$

We can just implement the system, using 1-D filter function

b = array of all b_K's

a = array of all a_K's

y = filter(b, a, x)

This one command would compute echoed signal y[n]

Here a = [1]

b = h[n] = [1 zeros(1,(n-1)*d) α zeros(1,(n-1)*d) α^2]

SIGNAL PROCESSING MONSOON PROJECT REPORT:-
PROBLEM 2:-

Part 2 - Cancel the echo

In this section, you will be presented with an audio signal containing an echo with non-uniform delays. Your task is to implement an echo cancellation algorithm to remove the echo components from the signal, leaving only the original, clean audio.

Task :

Input Audio Signal:

You will receive an audio signal in a specific format (e.g., WAV or MP3), which contains both the original audio and the echo with non-uniform delays.

Echo Cancellation:

Your task is to design and implement an algorithm to cancel the echo in the signal.

The echo cancellation algorithm should effectively remove the echo components, leaving behind the original, clean audio.

THE LMS APPROACH:-

- **The Least Mean Squares (LMS) algorithm is a widely used adaptive filter algorithm in the field of digital signal processing. Adaptive filters are systems that can automatically adjust their parameters based on the input signals to optimize some criteria. The LMS algorithm is particularly popular for applications such as noise cancellation, system identification, and equalization**

The Least Mean Squares (LMS) method finds applications in various fields due to its adaptability and effectiveness in solving problems related to signal processing, system identification, and adaptive filtering. Here are some common applications of the LMS method:

1. **Adaptive Filtering:**
    - **Noise Cancellation:** LMS is used for adaptive noise cancellation in audio signals. It can adaptively filter out unwanted noise from a signal, enhancing the quality of the desired signal.
    - **Echo Cancellation:** In telecommunications, LMS is applied to cancel echo in voice communication systems.
2. **Communication Systems:**
    - **Equalization:** LMS is employed in adaptive equalization of communication channels. It helps mitigate the effects of channel distortions and improve the quality of transmitted signals.
    - **Channel Identification:** LMS can be used for identifying and adapting to changing characteristics of communication channels in real-time.
3. **System Identification:**
    - **Modeling and Prediction:** LMS is used for system identification and modeling. It helps in predicting the behavior of dynamic systems by adapting to changes in the system parameters.
    - **Control Systems:** LMS is applied in adaptive control systems to adjust control parameters based on changing system dynamics.
4. **Acoustic Signal Processing:**
    - **Speech Processing:** LMS is utilized in speech processing applications, such as speech enhancement and speech recognition, where adaptive filtering is essential for handling variations in speech signals.
5. **Biomedical Signal Processing:**
    - **Biomedical Signal Filtering:** In biomedical applications, LMS can be used for adaptive filtering of signals, such as removing artifacts from electroencephalogram (EEG) or electrocardiogram (ECG) signals.
6. **Financial Time Series Analysis:**
    - **Stock Market Prediction:** LMS can be applied for adaptive modeling and prediction in financial time series analysis. It helps in adapting to changing market conditions for more accurate predictions.
7. **Smart Antenna Systems:**
    - **Beamforming:** In wireless communication systems with smart antennas, LMS is used for adaptive beamforming. It helps in adjusting the antenna pattern to focus on a specific direction for better signal reception.
8. **Seismic Signal Processing:**
    - **Seismic Data Processing:** In geophysical applications, LMS is employed for adaptive filtering of seismic signals. It aids in identifying and filtering out noise from seismic data.
9. **Image Processing:**
    - **Image Restoration:** LMS can be used in image processing for adaptive filtering, particularly in scenarios where images are degraded by noise, and adaptive methods are required for effective restoration

Applications

1. **Automobiles:** Noise attenuation inside vehicles and electronic mufflers for exhaust and induction system
2. **Industrial:** Cancelling noise from almost any industrial appliance
3. **Appliances:** Noise from daily appliances like ACs, exhausts, etc can also be controlled using ANC

Theory

We start by defining some of the concepts of probability theory that are used in understanding adaptive filters and the various algorithms.

- Expectation and variance

  The expected value or mean of a discrete random variable X is a weighted average of the possible values that x can take. It is written as $E(X)$. It can be calculated as follows

$$E(X) = \sum_{i=1}^{\infty} x_i \; p(x_i)$$

  where x_i is one of the possible values random variable X can take and $p(x_i)$ is the probability of X having the value x_i

  Variance is a measure of the spread of the possible values of the random variable. It is represented by σ^2. The variance of a discrete random variable X is calculated as follows

$$\sigma^2(X) = E[(X - E(X))^2]$$

$$= E\ [X^2] - (E[X])^2$$

- Covariance

  Covariance is the measure of joint variability of 2 random variables. For 2 jointly distributed real values random variables X and Y, the covariance is defined as the expected value of the product of deviations from their expected value. For real-valued random variables covariance is defined as follows

$$Cov_{XY} = E[(X - E[X])(Y - E[Y])]$$

$$= E[(XY] - E[X]E[Y]]$$

  For complex-valued random variables

$$Cov_{XY} = E[(X - E[X])(Y - E[Y])^*] \quad \text{(Complex conj. of 2nd factor is taken)}$$

- Correlation

  Correlation is any statistical relationship between 2 random variables. Mathematically, it is defined as the quality of least-squares fitting to the original data. It is obtained by taking the ratio of the covariance of the two variables in the question of our numerical dataset, normalized to the square root of their variances.

$$Corr_{XY} = \frac{Cov_{XY}}{\sigma_X \sigma_Y}$$

  We can say 2 random variables are uncorrelated if their covariance is zero.

- Stationary process

  We ideally model random processes such as noise as stationary processes. This means that the randomness of this variable is homogeneously distributed throughout the samples. The fluctuations in the values are not localized. This means that the e=mean and variance of the data remain constant throughout time.

  $E[x(n)] = \mu_n = \mu$ First order stationary

  $E[x(n)x^*(n-k)] = function\ of\ k = r(k)$ Second order stationary/Wide sense stationary if both conditions are satisfied

  Here, $r(k)$ can be considered a correlation function known as auto-correlation function.

- Hermitian transpose or conj. transpose

  For an $n \times m$ matrix A the hermitian transpose is defined as

$$(A^H)_{ij} = A^*_{ji}$$

Using the above equation we can obtain the following result for autocorrelation function $r(k)$

$$r^*(-k) = E[x^*(n)\, x(n+k)] = r(k)$$

- Power spectral density

  Power spectral density is the Fourier transform of the autocorrelation function.

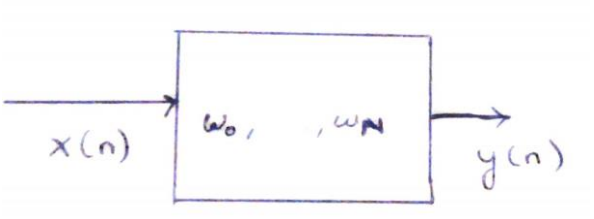
$$S_{xx}(\omega) = \int_{-\infty}^{\infty} R_{XX}(\tau) e^{-jw\tau}\, d\tau$$

where S_{XX} is power spectral density and R_{xx} is the autocorrelation function

Now we can start exploring adaptive filters. We begin by looking at a Wiener filter.

Wiener filter

For a filter as shown in the figure the output $y(n)$ is given by

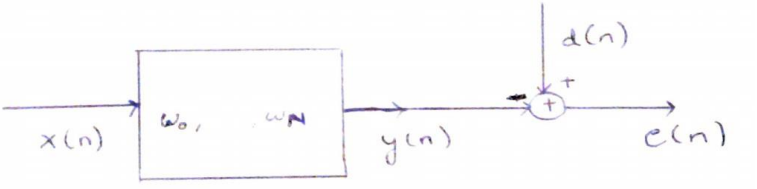


$$y(n) = w_0 x(n) + w_1 x(n-1) + \cdots + w_p x(n-N)$$

$$= w^T x(n)$$

as w and x(n) are both column vectors and to produce required result we need transpose of w

Now we consider as system as shown where we want y(n) to be an estimate of a target signal d(n)



If x(n) is stationary then y(n) is also stationary. x(n) and d(n) will be jointly stationary.

$$e(n) = d(n) - y(n)$$

Taking autocorrelation we get $R_{XX} = E[x(n)\, x^T(n)] = R$
Taking cross-correlation we get P a column vector such that $P = E[d(n)x(n)]$
We need to minimize $e(n)$ to ensure $y(n)$ is a close estimate of $d(n)$. Taking mean squared error or expected value of $e^2(n)$

$$E[e^2(n)] = E[\,(d(n) - w^T x(n))\,(d(n) - w^T x(n))^T\,]$$ substituting $y(n)$ as $w^T x(n)$

$$E[e^2(n)] = E[d^2(n)] - 2E[w^T x(n)\, d(n)] + E[w^T x(n)\, x^T(n)\, w\,]$$

Using properties of expectation and substituting cross correlation and autocorrelation

$$E[e^2(n)] = E[d^2(n)] - 2w^T P + w^T Rw$$

Taking gradient of this vector we get

$\nabla_w E[e^2(n)] = 0 - 2\nabla_w A + \nabla_w B$ where A and B are $w^T P$ and $w^T Rw$ respectively

Here,

$$A = w^T P = w_0 P(0) \cdots + w_k P(k) + \cdots + w_N P(N)$$

where,

$$\frac{\delta A}{\delta w_k} = P(k)$$

Gradient of B will be the vector containing the partial derivatives of B with respect to each w i.e., w_k for all values of k from 1 to N

$$\therefore \nabla_w A = P$$

$$B = w^T Rw$$

$$B = \sum_{i=0}^{N} w_i (Rw)_i$$

$$= \sum_{i=0}^{N} w_i \sum_{j=0}^{N} R_{ij} w_j$$

$$= \sum_{i=0}^{N} \sum_{i \neq k} w_i \sum_{j=0}^{N} R_{ij} w_j + w_k \sum_{j=0}^{N} R_{kj} w_j$$

$$\frac{\delta B}{\delta w_k} = \sum_{i=0}^{N} \sum_{i \neq k} R_{ki} w_i + \sum_{j=0}^{N} R_{kj} w_j + R_{kk} w_k$$

$$= 2 \sum_{i=0}^{N} R_{ki} w_i$$

Gradient of B will be the vector containing the partial derivatives of B with respect to each w i.e., w_k for all values of k from 1 to N

$$\therefore \nabla_w B = 2Rw$$

Substituting $\nabla_w A$ and $\nabla_w B$ into the gradient equation we get

$$\nabla_w E[e^2(n)] = -2P + 2Rw$$

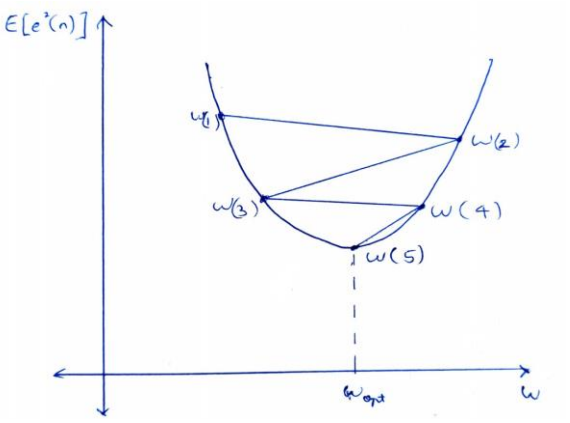
Therefore, for the optimum filtering condition where the error is minimized

$$w = w_{opt} = R^{-1} P$$

where $R = E[x(n)\, x^T(n)]$ and $P = E[x(n)\, d(n)]$

Steepest descent algorithm

We know that for this system $E[e^2(n)]$ is a function is filter coefficients.



We take the $E[e^2(n)]$ plot and find the gradient at the point of filter weights we are currently at. If the gradient at that point is positive then we should subtract a value proportional to the gradient from the current filter weights in order to arrive at the minima. If the gradient is negative then we add a term proportional to the gradient. This algorithm continues to go back and forth about the minima till it finally converges at the minima point. This process is shown in the figure.

Mathematically, steepest descent algorithm can be represented as-

$$\begin{aligned} w(i+1) &= w(i) + \frac{\mu}{2} \nabla_w E[e^2(n)]\big|_{w=w(i)} \\ &= w(i) - \frac{\mu}{2}[2Rw - 2P]_{w=w(i)} \\ &= w(i) + \mu[P - Rw(i)] \end{aligned}$$

This expression is for offline computation of filter weights. In order to carry this out in real time, we need P and R also in real time.

$$R = E[x(n)\, x^T(n)]$$

Taking a crude estimate of R,

$$R \approx x(n)x^T(n)$$

Similarly,

$$P = E[x(n)d(n)] \approx x(n)d(n)$$

$$\begin{aligned} w(n+1) &= w(n) + \mu[P - Rw(i)] \\ &= w(n) + \mu[x(n)\, d(n) - x(n)\, x^T(n)\, w(n)] \\ &= w(n) + \mu x(n)[d(n) - x^T(n)\, w(n)] \\ &= w(n) + \mu\, x(n)[d(n) - x^T(n)w(n)] \\ &= w(n) + \mu\, x(n)\,[d(n) - y(n)] \\ w(n+1) &= w(n) + \mu x(n)\, e(n) \end{aligned}$$

This derivation is for real valued $x(n)$. For complex case we can derive the exact same expression for filter weights but we have to consider conjugate transpose instead of the normal transpose taken in R.

Implementation

1. Adaptive filter controlled by LMS algorithm

The implementation of the adaptive filter using the LMS algorithm is quite simple and straightforward. We generate a random noise signal and pass it through the acoustic path. For the purpose of demonstration, we have assumed a transfer function P(n) for this acoustic path. The result of the convolution of the noise signal and acoustic path transfer function is the signal we need to cancel using our adaptive filter. The adaptive filter weights are controlled by the LMS algorithm and we have the algorithm summarised as follows:

$$y(n) = w^T(n)\, x(n)$$

$$e(n) = d(n) - y(n)$$

$$w(n+1) = w(n) + \mu x(n) e(n)$$

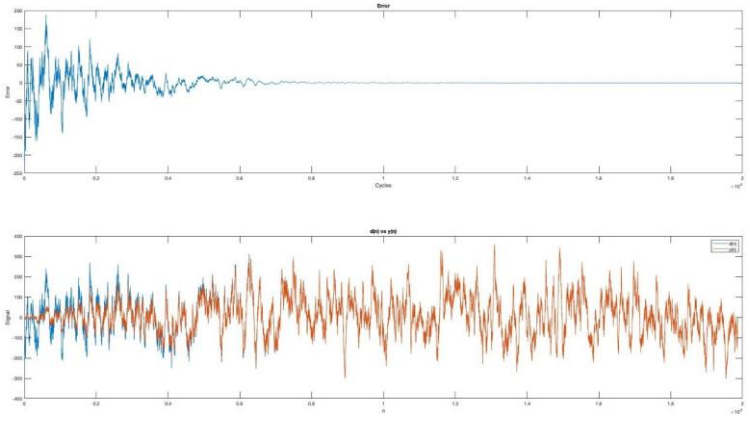
Here, w(n) is the weights vector in the instant n, y(n) and x(n) are the input and output signals respectively. e(n) is the filter's error. w(n+1) is the weights vector at the instant n+1, implying that it is constantly changing.

Here, μ is the step size

$$\mu \propto Convergence\ speed$$

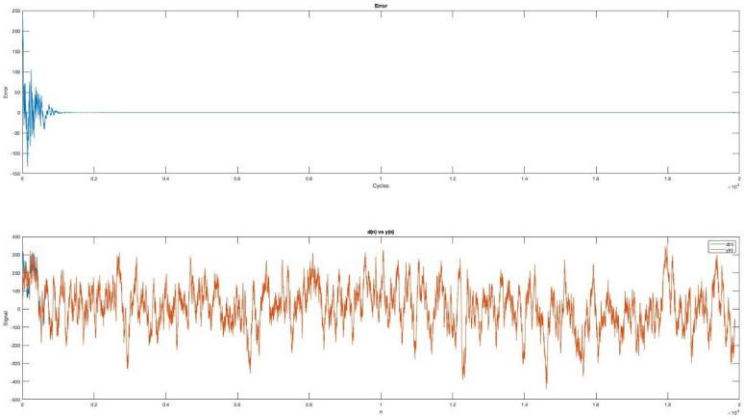
Running this algorithm in cycles through our noise signal causes our adaptive filter to converge to P(n) as we have shown in the previous section of this report. When the filter converges we are able to cancel out maximum noise.

Results



For step size (μ) = 0.01, the error plot is as given above. The error signal fluctuates initially before finally reaching zero after approximately 7500 cycles. The error plot is a direction representation of the convergence of d(n) and y(n) signals. The desired input signal (d(n)) and filter output (y(n)) converge around the same time as the error signal reaches zero

$$\mu = 0.05$$



From the resulting plots, we can clearly observe the adaptive nature of the filter as the generated signal y(n) slowly converges to the desired signal d(n). We also observe that for a very low value of step size the filter takes a lot longer or a lot more cycles to converge to d(n).

QUESTION 3

Problem Statement:

We are required to classify the kind of noise given in a file into one of the four possible types of noises presented to us, i.e.,

1. Traffic noise
2. Ceiling Fan Noise
3. Pressure Cooker Noise
4. Water Pump Noise

Solution:

The solution developed uses cross correlation to identify the noise present in the input file. Correlation helps quantify how much one signal resembles another. There are different types of correlation used in signal processing, including cross-correlation and auto-correlation.

Cross-Correlation:

Definition: Cross-correlation measures the similarity between two signals as a function of the time lag between them. It involves sliding one signal over the other and computing the sum of products at each position.

Application: Cross-correlation is commonly used in applications such as pattern recognition, communication systems, and signal synchronization. In pattern recognition, it can be used to identify a specific pattern within a larger signal.

Cross-correlation finds applications in various fields due to its ability to quantify the similarity between two signals. Some notable applications include:

1. **Communication Systems:**

In wireless communication, cross-correlation is used for signal synchronization. It helps align received signals with the transmitted signals, allowing for accurate data recovery.

2.**Radar Systems:**

radar systems use cross-correlation to compare transmitted and received radar pulses. This is crucial for target detection, tracking, and range measurement.

3.**Audio and Speech Processing:**

Cross-correlation is employed in audio and speech processing for tasks such as speaker recognition, speech recognition, and echo cancellation. It helps identify similarities between the received signal and reference patterns.

4.**Image Processing:**

In image processing, cross-correlation is used for template matching. It helps locate a specific pattern or object within an image by comparing it to a reference template.

5.**Biomedical Signal Processing**:

Cross-correlation is applied in biomedical signal processing for tasks like electrocardiogram (ECG) signal analysis. It aids in aligning and comparing heartbeats for diagnostic purposes.

6.**Seismology:**

Seismologists use cross-correlation to compare seismic signals recorded at different locations. This technique helps identify seismic events and locate their epicentres.

7.**Robotics and Computer Vision:**

Cross-correlation is utilized in robotics and computer vision for object recognition and tracking. It assists in matching features within images or video frames.

8.**Structural Health Monitoring:**

In structural health monitoring of buildings and bridges, cross-correlation can be used to compare ambient vibration signals over time. Changes in the correlation function may indicate structural damage.

9.**Financial Signal Processing:**

Cross-correlation is applied in financial signal processing to analyse the relationship between financial time series, such as stock prices. It helps identify patterns and correlations in market data.

10.**Wireless Positioning Systems:**

Cross-correlation is used in global navigation satellite systems (GNSS) for accurate positioning. By comparing signals from different satellites, receivers can calculate precise location information.

These applications highlight the versatility of cross-correlation in extracting meaningful information from signals, aligning patterns, and facilitating various technological advancements across different domains.

Implementation:

The approach is to find how much correlation is between the input signal and the template noises we have.

The template noise with which the signal is most highly correlated is the noise present essentially in the input file.

For generating the template noise with which we plan to obtain the correlation was generated using various methods. The best meth among this which worked with the highest accuracy and lowest error rate is the method discussed below for generating template noises:

We obtain 4C2 = 6 combinations of the 4 input files. We know that the background music behind all the noise samples is exactly the same. Also, the length of all 4 files is also known to be the same. Assuming the nature of noise is purely additive in all the 4 given music samples, the subtraction of the data points of two of these 4 files cancels the music content and leaves behind only the noise content. Hence, we finally generate 6 ‘only noise’ samples in the following manner:

fan_pump = fan – pump

fan_traffic = fan – traffic

fan_cooker = fan – cooker

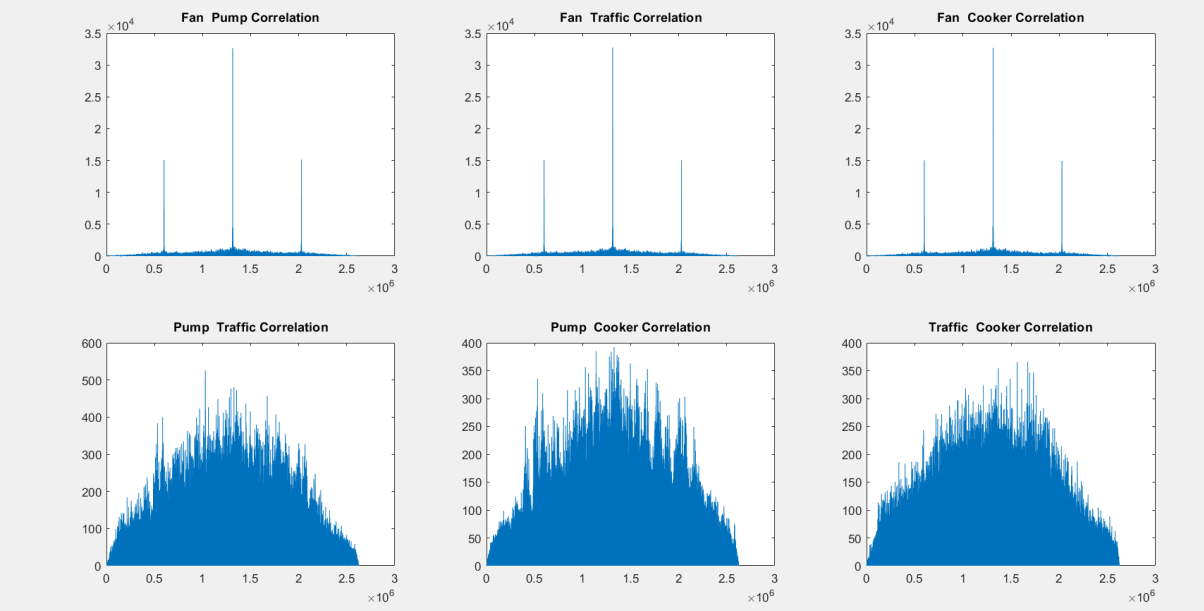
pump_traffic = pump-traffic

pump_cooker = pump-cooker

traffic_cooker = traffic – cooker

We now obtain the cross correlation using ‘xcorr’ function in MATLAB of each of these 6 signals with the input file. We further take the absolute value of these correlation graph and plot it.

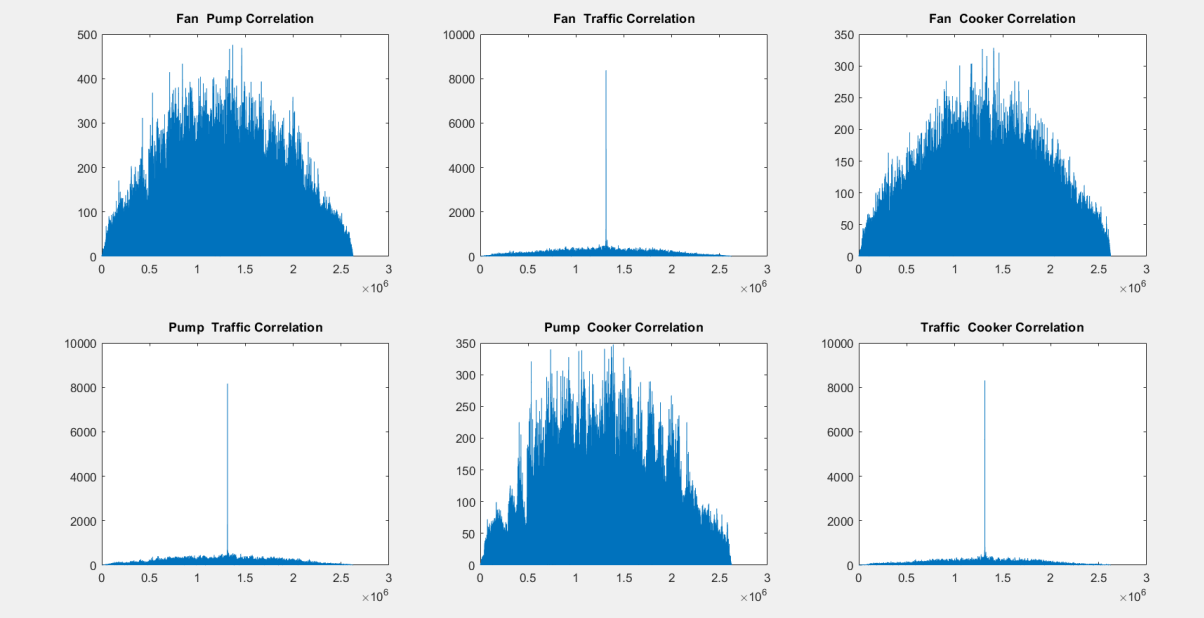
Observation:



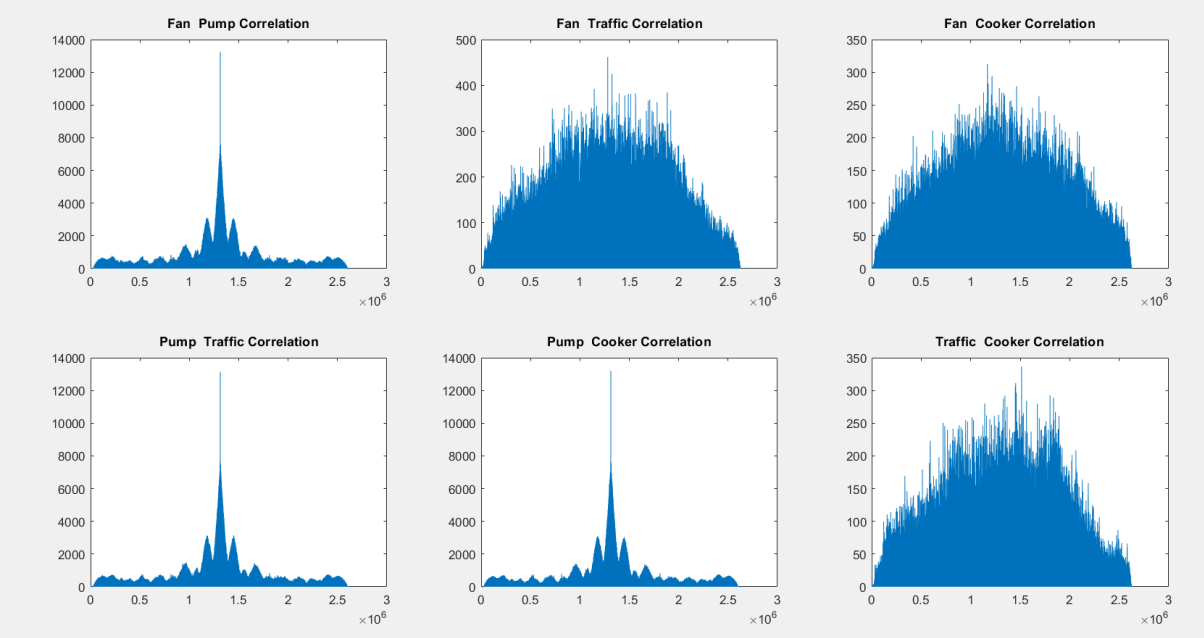
When the input file contains ceiling fan noise, we can clearly observe the peaks in the correlation graphs have much higher magnitude than the other graphs. This clearly indicates the presence of fan noise in the input file.

Similarly we can generate graphs for other inputs:

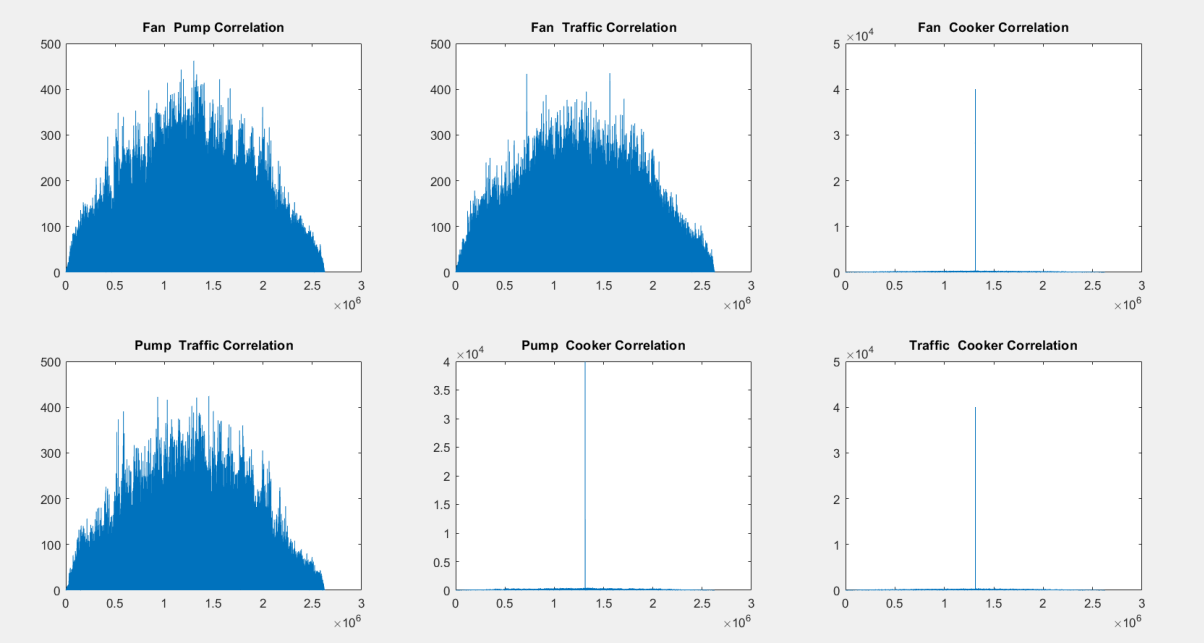
Traffic Noise



Pump Noise



Pressure Cooker Noise



As we observe, there are clear and high valued peaks in three out of the 6 correlation graphs. It indicates that the common term among these 3 corresponds to the category of noise present in the input.

Thus, we calculate the max value in each of thee 6 correlation vectors using the ‘max’ function. We then store these max values in an array. The three highest value in this array are checked using if-else and the chosen indexing as to which noise category do they have in common.

That noise is printed.

Further Comments:

For obtaining better results, we pay much attention to the generation of template noises, Thus, for the speech signal cases, we use the noisy speech templates and for the music input signal we take the music files as the basic templates.

Note that this is just to have a higher accuracy.