National Institute of Technology
Silchar

Department of Computer Science and Engineering

Properties of Discrete Fourier Transform and Noise Reduction from Input Signal using DFT

Project submitted to:
Dr. Anish Kumar Saha

Bachelor of Technology Mini-Project
Signals and Data Communication, SDC
CS-207

Swagat S. Bhuyan, Anamitra Saikia
18-1-5-059, 18-1-5-060

July 2020
Silchar, Assam

Contents

0.1 Abstract

The DFT is a mathematical technique to decompose a signal into sinusoidal components of various frequencies ranging from 0 frequency (i.e., constant) up to the maximum frequency (i.e., just below the half the sampling rate). The DFT is the most important discrete transform, used to perform Fourier analysis in many practical applications. In digital signal processing, the function is any quantity or signal that varies over time, such as the pressure of a sound wave, a radio signal, or daily temperature readings, sampled over a finite time interval (often defined by a window function). In image processing, the samples can be the values of pixels along a row or column of a raster image. The DFT is also used to efficiently solve partial differential equations, and to perform other operations such as convolutions or multiplying large integers. In this Project, we will be discussing about the properties of DFT that will ensure success in Noise Reduction in a given input signal to attain the original signal.

Noise reduction to recover a target signal from an input waveform is important in a number of fields. We usually use a frequency spectrum to remove noise from the input waveform. Although it is difficult to distinguish a signal from the noise in the time domain, this task tends to become easier in the frequency domain. We would be manually adding noise using Random Function coding and then trying to retrieve the original waveform by Noisy Waveform Restoration using DFT.

The aim of this project is to understand the properties of Discrete Fourier Transform and then using DFT to reduce noise from an input waveform to obtain original waveform on transmission.

Chapter 1

Introduction

The Fourier representation of signals is crucial for understanding how filters work and what a spectrum is, but it is not a practical tool because it is a continuous function of frequency and therefore its computation would on paper require an infinite number of operations. Hence we resort to a new representation of discrete-time signals, the Discrete Fourier Transform (DFT), which is closely related to the discrete-time Fourier transform, and can be implemented either in digital hardware or in software. The DFT is of great importance as an efficient method for computing the discrete-time convolution of two signals, as a tool for filter design, for measuring spectra of discrete-time signals, and as we will progress in this project, Noise Reduction of Input Signals to obtain original waveform. While computing the DFT of a signal is generally easy (requiring no more than the execution of a simple program) the interpretation of these computations can be tough to comprehend because the DFT only provides a complete representation of finite-duration signals.

The Discrete Fourier Transform computes the correlation of a signal with sinusoidal signals of discrete, completely unrelated frequencies which are uniformly sampled in the independent variable. For instance, a sinusoidal signal of frequency ω will be perfectly correlated (i.e., something like "1" or an otherwise significant value) with sinusoidal signals of frequency ω and completely uncorrelated (i.e., "0" or an otherwise insignificant value) with the other sinusoidal signals of discrete frequencies chosen in the discrete Fourier transform.

What DFT simply does is use a number of sample points, say N, across the spectrum at regular intervals, say T. Doing such helps facilitate compression of higher degree signals in terms of complex sine and cosine terms without having to render at infinite points present in the signal. This opens the door to numerous applications as discussed, noise reduction being one of the pivotal uses in order to restore native signal waveform.

1.1 History

The Fourier Series and the Fourier Transforms, both pivotal to the world of Signal Processing and Data Communication, have no doubt revolutionized many fields of science, especially the fields of computer science related to signal processing, data communication, etc. It was named after Joseph Fourier, who introduced the Fourier series and transform as he solved for a mathematical approach to describe how heat transfers in a metal plate, publishing his initial results in his 1807 Mémoire sur la propagation de la chaleur dans les corps solides (Treatise on the propagation of heat in solid bodies), and publishing his Théorie analytique de la chaleur (Analytical theory of heat) in 1822. The Mémoire introduced Fourier analysis, specifically Fourier series. Through Fourier's research the fact was established that an arbitrary (considered continuous) function can be represented by a trigonometric series of sines and cosines. Joseph Fourier was solving a equation of Unsteady Heat Conduction having no internal heat generation (this is known as Diffusion Equation) problem in metal plate and in the end he discovered a series of trigonometric function.

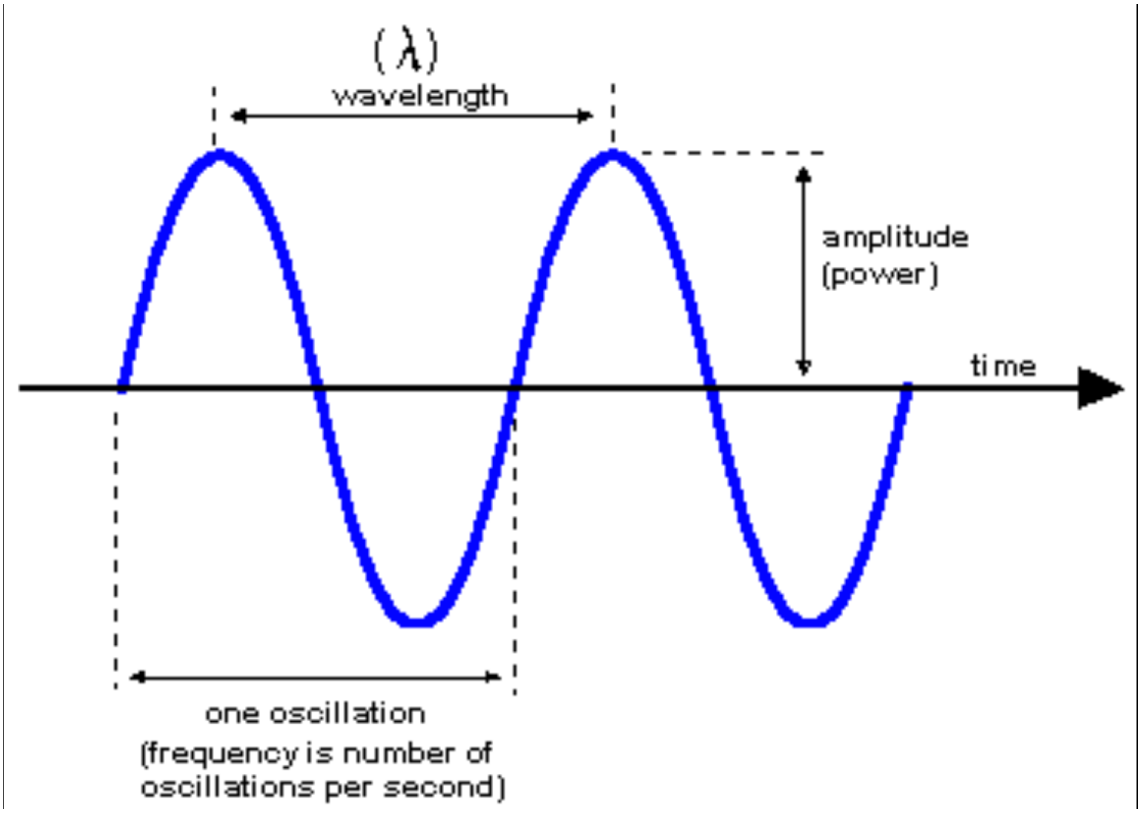
Fig.: Joseph Fourier (1768-1830)

Chapter 2

Signals

2.1 Introduction

A signal is a function that conveys information about a phenomenon. In electronics and telecommunications, it refers to any time varying voltage, current or electromagnetic wave that carries information.In a communication system, a transmitter encodes a message to create a signal, which is carried to a receiver by the communications channel.



In a broader sense , Signal is an action of conveying information via various means. It is a part of evolution. In modern age signals are used for many purposes and it is an essential part of humankind.

A signal can be defined as a function of time. The common signals which are mathematically generated are:

- constant (or static or DC) signal
- the unit step signal
- the unit ramp signal
- a rectangular pulse signal
- a sinusoidal signal

In the real world, Signals are not determined by mathematical formulas as signals are used for transmission of information over a medium and thus it varies according to the data. Signals are used for FM/AM radio, Cable TV signal, Audio signal ,Video signal , Telephone network , Digital electronics etc.

In a more simplistic viewpoint, signals are basically a piece of information or data which is conveyed by the use of mathematical functions.

Ex :- Suppose height of water column (H) in a tank is decreasing at the rate of 0.2 m/s then the result of a sensor sensing this height will also differ with time and the signal so produced by the sensor can be expressed as:

H(t) = X(0.2) t meters,

Here X is a proportionality constant of sensor having dimensional analysis m/s. Thus the information of rate of decrease of water column is a expressed in form of mathematical information.

Signals are represented mathematically as functions of one or more independent variables. e.g.- a speech signal can be represented mathematically by something called acoustic pressure as a function of time, and a picture can be represented by brightness as a function of two spatial variables.

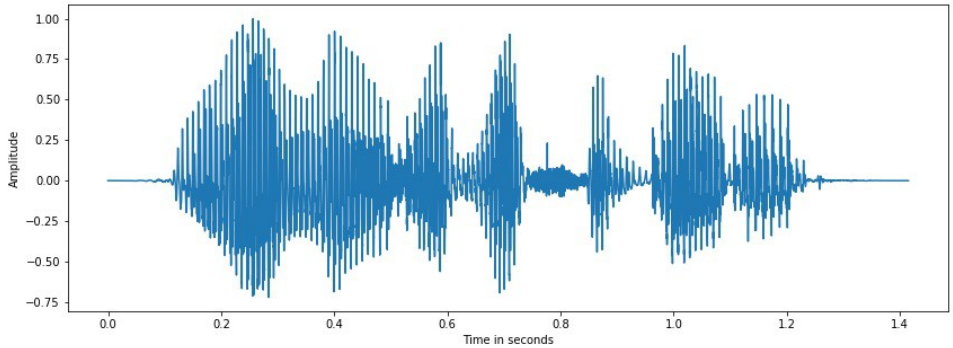


Fig.: Speech Signal Plot

2.2 Characteristics

The Characteristics of a signal defines the signal and gives information about the nature of the signal. The Characteristics of a signal are:

- **Amplitude:** The amplitude of a wave refers to the maximum amount of displacement of a particle on the medium from its rest position.
- **Frequency:** Frequency is the rate of Oscillation of a signal's waveform in a second. The unit of Frequency is hertz (Hz).
- **Time Period:** The time period of a signal is the time in which the wave completes its one full cycle. The unit of the time period is Second. The time period is denoted by 'T' and it is the inverse of frequency. I.e. T=1/F
- **Phase:** In electronic signaling, phase is a definition of the position of a point in time (instant) on a waveform cycle.
- **Size of a signal:** The size of a signal is a number that shows the strength or largeness of that signal. According to the size of the signal, there are two parameters:

  Signal Energy, Signal Power

2.3 Classification of Signals

Signals can be classified into different categories based on their nature and properties :

2.3.1 Amplitude based

Analog Signal: An analog signal is any continuous signal for which the time-varying feature (variable) of the signal is a representation of some other time-varying quantity, i.e., analogous to another time-varying signal. For example, in an analog audio signal, the instantaneous voltage of the signal varies continuously with the pressure of the sound waves.

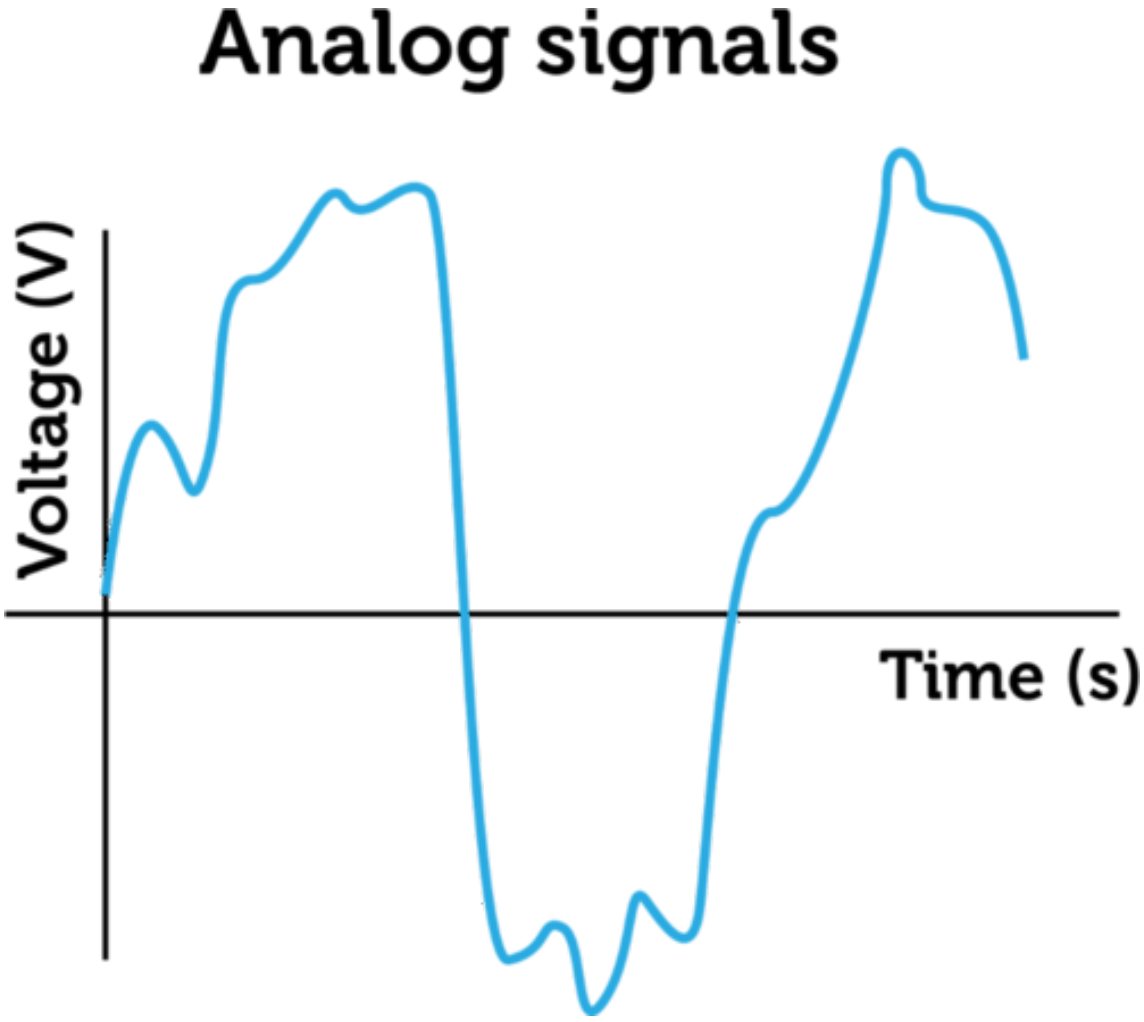


Fig.: Analog Signal

Digital Signal: A digital signal is a signal that is being used to represent data as a sequence of discrete values; at any given time it can only take on one of a finite number of values. A special case is a logic signal or a binary signal , which varies between a low and a high signal level. Because discretization, relatively small changes to the analog signal levels do not leave the discrete envelope, and as a result are ignored by signal state sensing circuitry. As a result, digital signals have noise immunity.

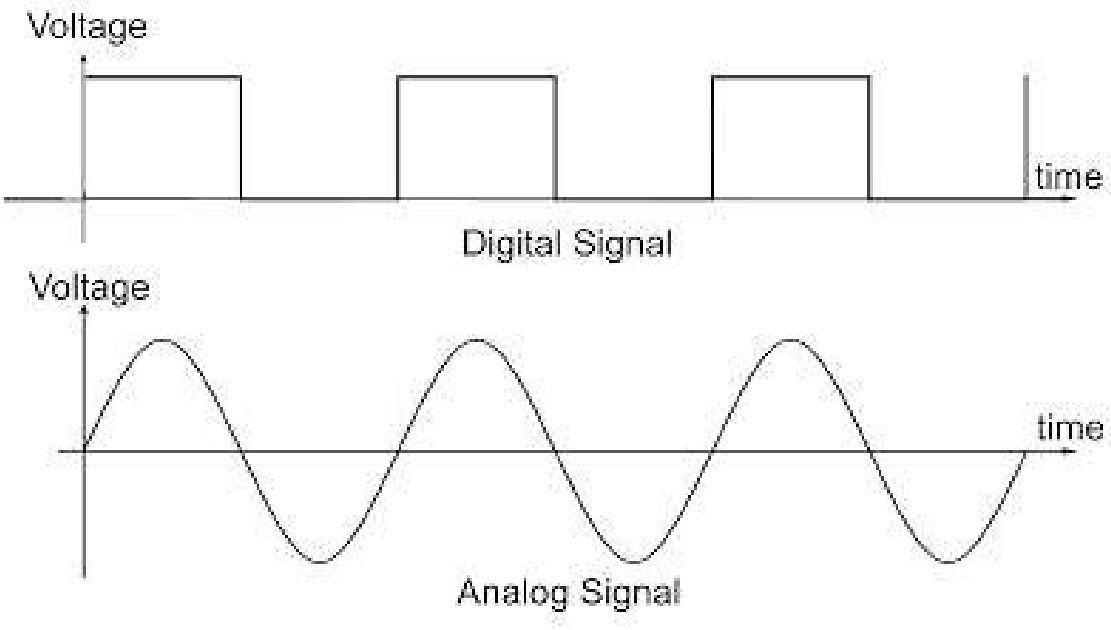


Fig.: Analog Signal vs Digital Signal

2.3.2 Time based

Continuous Time Signal: Continuous-time signal is the "function of a continuous-time variable that has an infinite set of numbers in its sequence". The continuous-time signal can be represented and defined at any instant of the time in its sequence.

Discrete Time Signal: Discrete-time signal is the "function of a discrete-time variable that has a countable or finite set of numbers in its sequence". The discrete-time signal can be represented and defined at certain instants of time in its sequence.

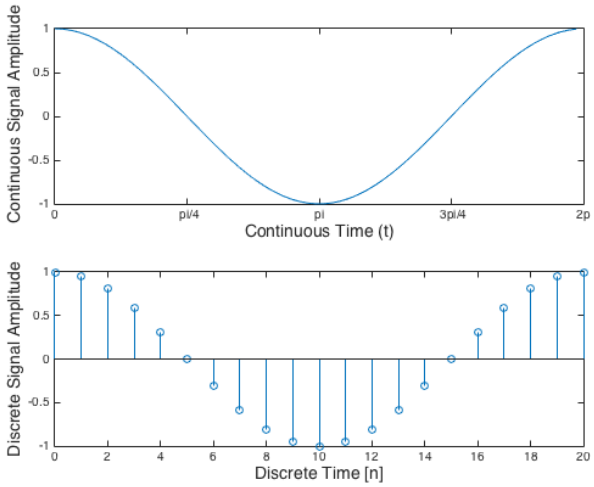


Fig.: Cont. Time Signal vs Discrete Time Signal

2.3.3 Periodicity based

Periodic Signal: signal is a periodic signal if it completes a pattern within a measurable time frame, called a period and repeats that pattern over identical subsequent periods. The completion of a full pattern is called a cycle. A period is defined as the amount of time required to complete one full cycle.

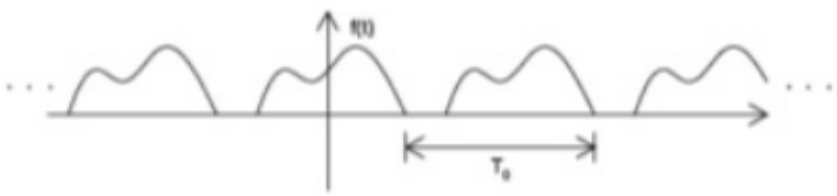


Fig.: Periodic Signal

Aperiodic Signal: A signal that does not repeat itself after a specific interval of time is called an aperiodic signal. aperiodic function can be considered similar to a periodic function with an infinite period.

Fig.: Aperiodic Time Signal

2.3.4 Energy and Power

Energy Signal: signal is said to be an energy signal if it has a finite amount of energy associated with it. This finite energy, when averaged over an infinite amount of time, will result in zero power.

Power Signal: signal is said to be a power signal if it has a finite amount of power associated with it. This finite power when accumulated over an infinite amount of time, will result in infinite energy.

2.3.5 Deterministic Nature

Deterministic Signal: A signal is said to be deterministic if there is no uncertainty with respect to its value at any instant of time. Or, signals which can be defined exactly by a mathematical formula are known as deterministic signals.

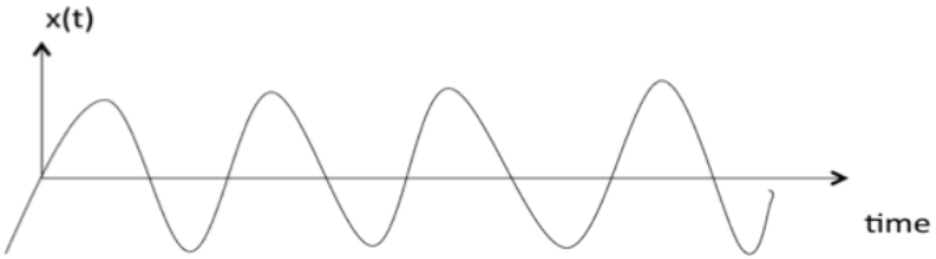


Fig.: Deterministic Signal

Non-Deterministic Signal: A signal is said to be non-deterministic if there is uncertainty with respect to its value at some instant of time. Non-deterministic signals are random in nature hence they are called random signals. Random signals cannot be described by a mathematical equation. They are modelled in probabilistic terms.

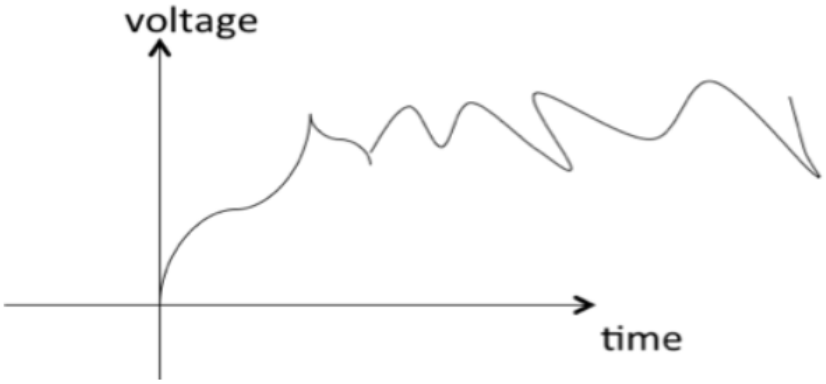


Fig.: Non-Deterministic Signal

2.4 Applications of Signal Processing

1. Telecommunication: The Telecommunication network is based on signals.Cellular networks, Internet, WiFi, Bluetooth, NFC operated on signals of different types which varies on frequency and amplitude which affects the rate of data transmission and range.

2. Healthcare, Pharmacy and medicine: Analysis of the ECG, EEG are just some examples of medicine using signal processing. Further, a lot of non-invasive techniques for health monitoring are being developed which involve signal processing strategies on Ultra Wide Band signals.

3. Meteorology: Analysis of a vast extent of collected data involves signal processing methods as well, and hence these help to perform accurate weather prediction.

4. Radars, Sonar , military equipment and surveillance systems involve signal processing.

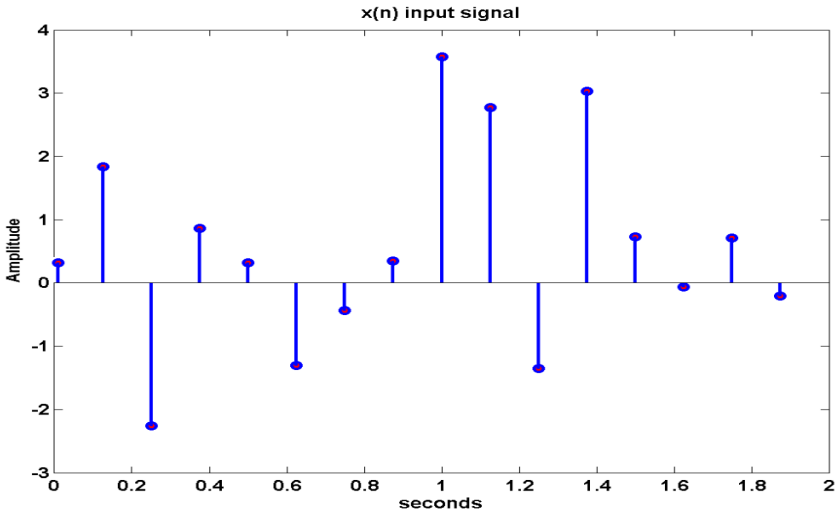
5. Seismology: scientists use spectral analysis to determine periodicity and anomalies in the recorded data. It gives them a better insight into the nature of earthquakes and what follows before and after an earthquake. Recently, after the 2011 Tohoku earthquake, it has been revealed that slow quakes which are basically like noise in the grand scheme of things as recorded by a seismograph are being used to predict big earthquakes near subduction zones. This involves a lot of efficient and clever signal processing techniques.

Chapter 3

Discrete Fourier Transform

3.1 Introduction

The Discrete Fourier Transform is a mathematical technique to decompose a signal into sinusoidal components of various frequencies ranging from 0 frequency (i.e., constant) up to the maximum frequency (i.e., just below the half the sampling rate). DFT computes the correlation of a signal with sinusoidal signals of discrete, completely unrelated frequencies which are uniformly sampled in the independent variable. For instance, a sinusoidal signal of frequency ω will be perfectly correlated (i.e., something like "1" or an otherwise significant value) with sinusoidal signals of frequency ω and completely uncorrelated (i.e., "0" or an otherwise insignificant value) with the other sinusoidal signals of discrete frequencies chosen in the discrete Fourier transform. Oftentimes, it is convenient to express a signal as a sum of sine waves.



3.2 Properties of DFT

- **Linearity:** The Linearity property of DFT states that The transform of a sum is the sum of the transforms: DFT(x+y) = DFT(x) + DFT(y). Let $x_0, x_1, \ldots\ldots x_{n-1}$ and $(y_0, y_1 \ldots\ldots y_{n-1}$ be two sets of discrete samples with corresponding DFTs given by X(m) and Y(m) . Then DFT of the sample set $x_0 + y_0, x_1 + y_1 \ldots\ldots x_{n-1} + y_{n-1}$ is given by X(m) + Y(m).

$$ax[n] + by[n] \leftarrow DFT \rightarrow NaX[k] + bY[k]$$

- **Periodicity:** The Periodicity property of DFT states that If the expression that defines the DFT is evaluated for all integers k instead of just for k=0,. . . , N-1 then the resulting infinite sequence is a periodic extension of the DFT, periodic with period N.

$$X[k + N] = X[k] \ \forall k$$

- **Symmetry:** The DFT of a real-valued discrete-time signal has a special symmetry, in which the real part of the transform values are DFT even symmetric and the imaginary part is DFT odd symmetric

$$X[k] = X * [kmodN].$$

  Thus If x[n] is real and circularly even, then X[k] is also real and circularly even.

- **Time Reversal:** The Time reversal property states that if

$$X(n) \xleftrightarrow[N]{DFT} x(k) \text{ And}$$

$$\text{Then } x((-n))_N = x(N-n) \xleftrightarrow[N]{DFT} x((-k))_N = x(N-k)$$

  It means that the sequence is circularly folded; its DFT is also circularly folded.

- **Time Scaling:** Time Scaling deals with the effect on the frequency-domain representation of a signal if the time variable is altered.

- **Time Shifting:** Time Shifting property states that a shift in time is equivalent to a linear phase shift in frequency. Since the frequency content depends only on the shape of a signal, which is unchanged in a time shift, then only the phase spectrum will be altered.

$$F[X[m - m_0]] = e^{jm_0\omega}X(e^{j\omega})$$

- **Frequency Shifting (Modulation):** Frequency shifting enables us to shift a signal to a different frequency, allows us to take advantage of different parts of the electromagnetic spectrum is what allows us to transmit television, radio and other applications through the same space without significant interference.

$$F[X[m]e^{jm_0\omega}] = X(e^{j(\omega-\omega_0)})$$

- **Convolution:** The convolution theorem states that convolution in time domain corresponds to multiplication in frequency domain and vice versa:

$$x_1(n)*x_2(n) \leftrightarrow \mathbf{X}_1(e^{j\omega})\ \mathbf{x}\ X_2(e^{j\omega})$$

- **Multiplication:** The Multiplication property states that if:

$$x_1(n)\ \mathbf{x}\ x_2(n) \leftrightarrow \mathbf{X}_1(e^{j\omega})*X_2(e^{j\omega})$$

  It means that multiplication of two sequences in time domain results in circular convolution of their DFT s in frequency domain.

- **Parseval's Theorem:** The Parseval's theorem states:

$$\sum_{n=0}^{N-1} X(n)\, y^*(n) = 1/N \sum_{n=0}^{N-1} x(k)\ y^*(k)$$

  This equation gives energy of finite duration sequence in terms of its frequency components.

- **Duality:** The Duality Property tells that if x(t) has a Fourier Transform X(ω), then if we form a new function of time that has the functional form of the transform, X(t), it will have a Fourier Transform x(ω) that has the functional form of the original time function (but is a function of frequency)

$$x(t) \leftrightarrow X(\omega)$$
$$X(t) \leftrightarrow 2\pi\, x(-\omega)$$

3.3 DFT Errors

3.3.1 Aliasing

If the initial samples are not sufficiently closely spaced to represent high-frequency components present in the underlying function, then the DFT values will be corrupted by aliasing. As before, the solution is either to increase the sampling rate (if possible) or to pre-filter the signal in order to minimise its high frequency spectral content.

3.3.2 Leakage

If we attempt to complete the DFT over a non-integer number of cycles of the input signal, then we might expect the transform to be corrupted in some way.

Consider the case of an input signal waveform which is a sinusoidal wave with a fractional number of cycles in the data samples. The DFT for this case (given for n = 0, n = N/2) is shown below:

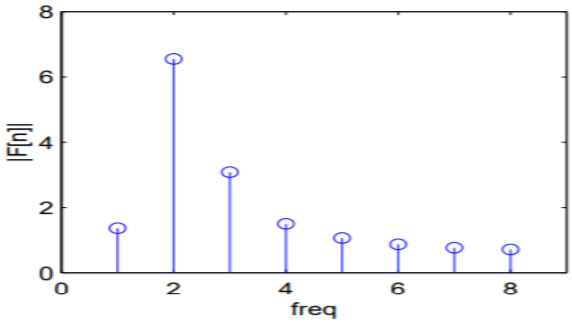


Figure 7.5: *Leakage.*

Here, we would not be surprised to find the Discrete Fourier Transform to give an result at just the quantified frequencies on either sides of the true frequency band. This definitely does happen but we also find non-zero results at all other frequency bands. This smearing effect, which is known as leakage, arises because we are effectively calculating the Fourier series for the waveform in the above plot, which has major discontinuities.

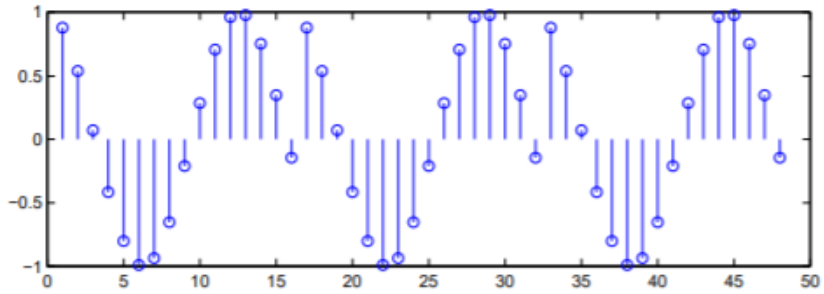


Figure 7.6: *Leakage. The repeating waveform has discontinuities.*

Most sequences of real data from the sample space of big data are much more complex than the sinusoidal sequences exemplified above and so it will not be possible to avoid inculcating discontinuities when using a finite number of sampling points from the sequence in order to calculate the Discrete Fourier Transform. The solution is to use one of the window functions which we faced in the design of FIR filters (ex:- the Hamming windows). These window functions taper the samples towards zero values at both endpoints, and so there is no discontinuity with a hypothetical next period. Hence the leakage of spectral content away from its true location is much reduced, as shown below:

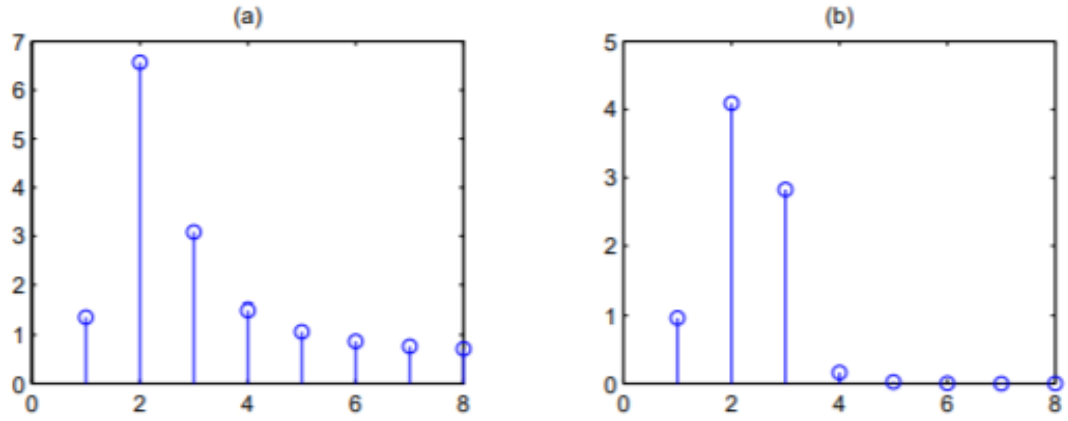


Figure 7.7: *Leakage is reduced using a Hanning window.*

3.4 Fast Fourier Transform Algorithm

The time taken to evaluate a Discrete Fourier Transform on a digital computer depends principally on the number of multiplications involved, since these are the slowest operations. With the DFT, this number is directly related to N^2 (matrix multiplication of a vector), where N is the length of the transform. On a usual practical scenario, N is chosen to be at least 256 in order to get a reasonable approximation for the spectrum of the sequence under consideration – hence computational speed becomes a major consideration. Highly optimized computer algorithms for calculating Discrete Fourier Transforms have been developed for a very long time now (from the 1960s). These are known as Fast Fourier Transform (FFT) algorithms.

In this project, we have used Fast Fourier Transform, fft() to perform Discrete Fourier Transform on the given time vector related transmitted noise-corrupted signal waveform. This enables us to cipher out the low amplitude frequencies once we plot out the noisy response. This plot is now at our hands for modification. We then simply eliminate this noise using zeros() and return the response to the signal. We then make the signal undergo Inverse Discrete Fourier Transform using Inverse Fast Fourier Transform, ifft() in MATLAB. We then obtain the original transmitted signal we began with before randomizing its waveform using a random function to simulate noise infestation.

Fast Fourier transform has hence proven to be a reliable means to obtain discrete Fourier transform of a signal transmitted, so as to isolate out the low amplitude frequency bands in order to eliminate noise and retrieve native signal waveform.

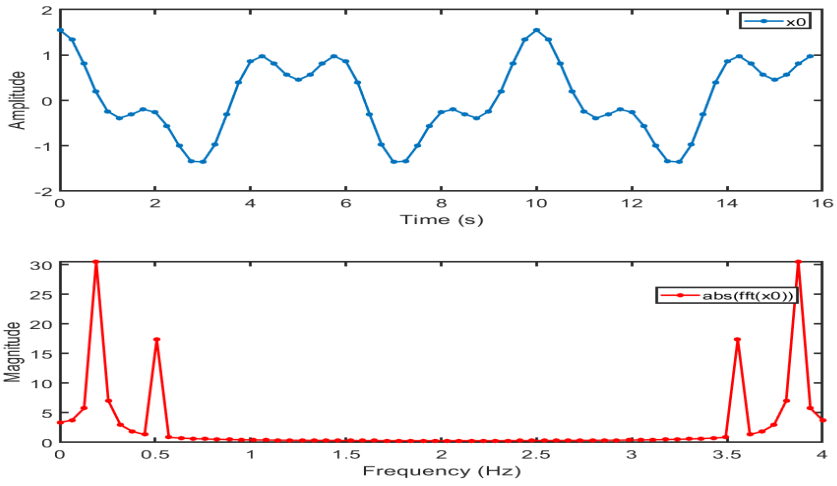


Fig.: Depiction of FFT over a signal

Chapter 4

Noise Reduction for Native Signal Restoration

As discussed earlier, Noise can infest any given signal while transmission in a lot of different ways. The compression and transmission of signals across large distances is rendered meaningless if the receiver is not able to correctly identify and analyse the required signal. This leads to huge cases of data loss in many data communication projects that rely on long distance signal transmission. Noise Reduction hence plays a pivotal role in making sure input signal waveform is intact and ready for analysis once received at the required destination receiver end. Noise reduction to recover a target signal from an input waveform is hence important. We usually use a frequency spectrum to remove noise from the input waveform. Although it is difficult to distinguish a signal from the noise in the time domain, this task tends to become easier in the frequency domain.

4.1 Methodology

4.1.1 Noise Influx

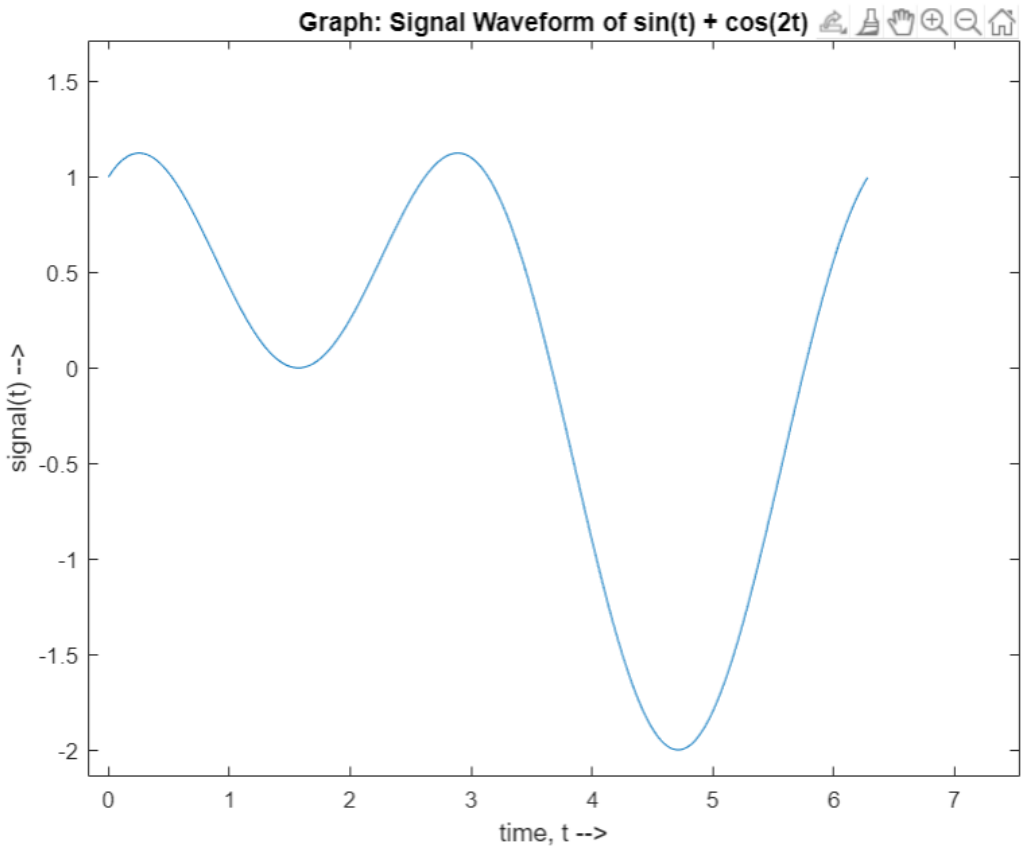
To recreate the problem of Noise Infestation on a large scale, long distance transmission will be required, which may get out of the scope if this mini project. Hence, to simulate Noise incursion in a given native signal, we use MATLAB to mathematically use random signals created using a random function generator to successfully induce noise into a target signal waveform.

Let us take an example of a native signal:

$$signal = \sin(t) + \cos(2t)$$

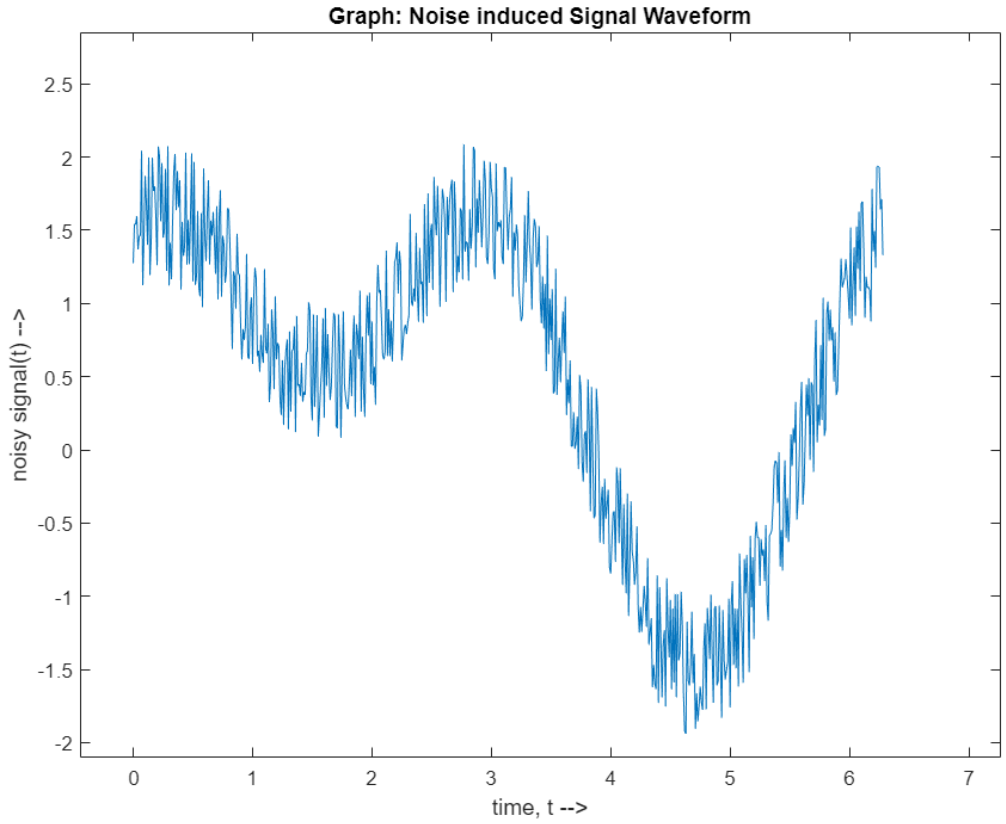
where t is the time axis, (array of values in MATLAB) that vary from 0 to 2π with an interval of 0.01 to maintain continuity.

When we plot a graph of signal in terms of time t, we get the following graph rendered on MATLAB:



As seen in the above graph, the x-axis is the time axis, t and the y-axis is the signal axis with respect to time. This can now be considered as the 'pure' signal waveform being subjected to random functions to simulate the effect of noise influx in a transmitted signal.

After acquiring the given native signal waveform, we use rand() function in MATLAB to generate a random function of vector for the sole purpose of Noise Influx into the Input Signal Waveform. This generates a successful Noisy Signal Waveform with minutely varying amplitudes, hence a perfectly noise-induced signal waveform as shown:



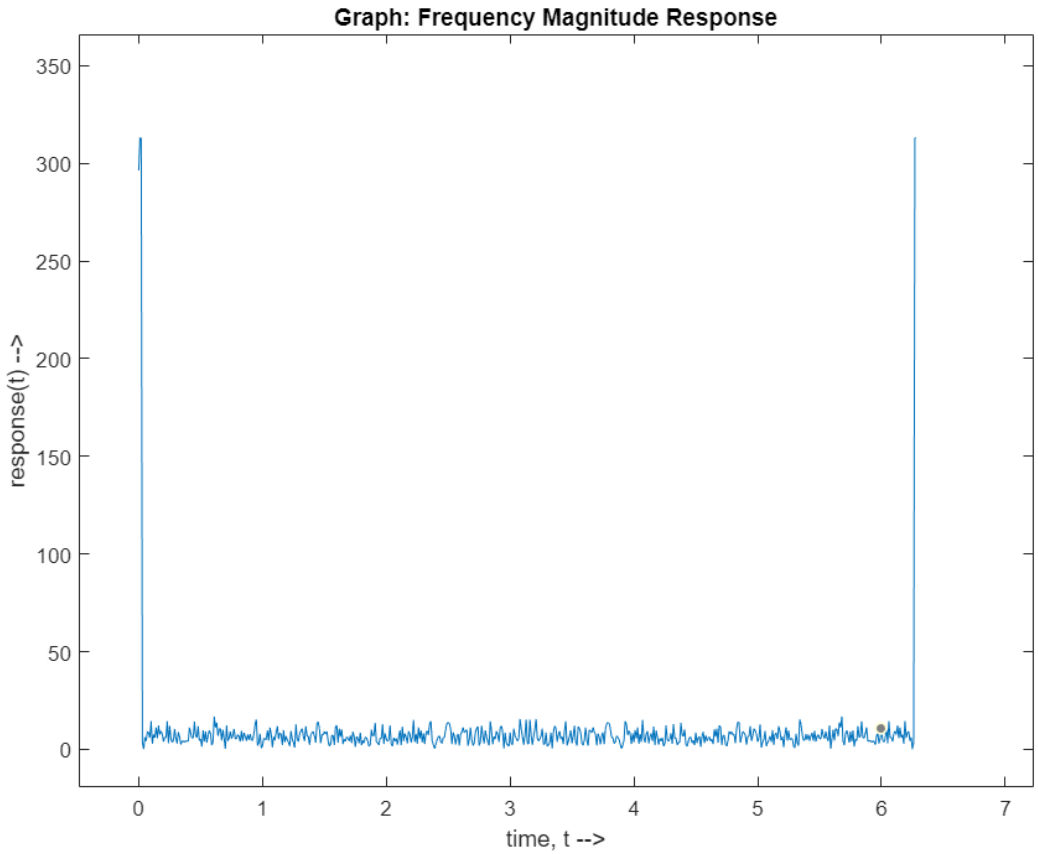
As seen in the above graph, the x-axis is the time axis, t and the y-axis is the noisy signal axis with respect to time. This can now be considered as the noise-infested signal waveform to be subjected to Fast Discrete Fourier Transform algorithms to enable native signal waveform restoration.

4.1.2 Fast Fourier Transform Algorithm

We obviously cannot use Continuous Fourier Transform on a signal in Computer as it is simply impractical to solve this in an infinitely large sampling point vector. Here, Discrete Fourier Transform comes into our help. We take the help of sampling our given noisy signal waveform to take multiple sample across the signal at intervals and perform DFT.

Fast Fourier Transform (FFT) is a Discrete Fourier Transform algorithm that is very efficient in calculating the DFT of a given noisy signal waveform to establish the corresponding DFT signal to separate out the noisy components of the signal to obtain the original signal waveform.

Using the fft() function in MATLAB, we can perform DFT with ease in the given noise induced signal waveform. We then obtain the Frequency Magnitude Response of the Noisy Signal after having performed fft() on the signal, as shown below:

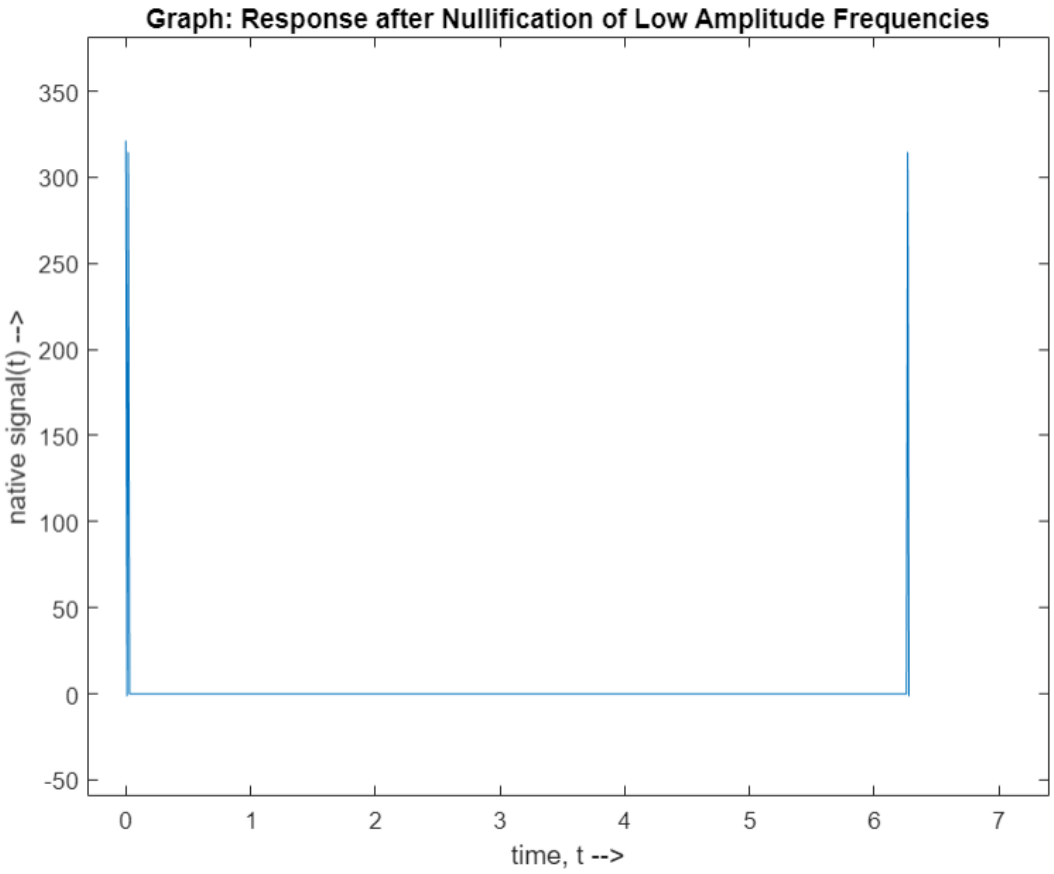


As seen in the above graph, the x-axis is the time axis, t and the y-axis is the response. This is the response highlighting the low amplitude noise that has been detected in the transmitted signal waveform and hence has to be eliminated to obtain native signal waveform.

4.1.3 Noise Nullification using zeros()

Now that we have quantify-ably obtained the low amplitude frequencies of the noise induced signal waveform, we can easily nullify our findings from the signal to obtain the required native target signal waveform. This can be done using zeros() MATLAB function to find the low amplitude frequencies and then turn them into 0s (nullify).

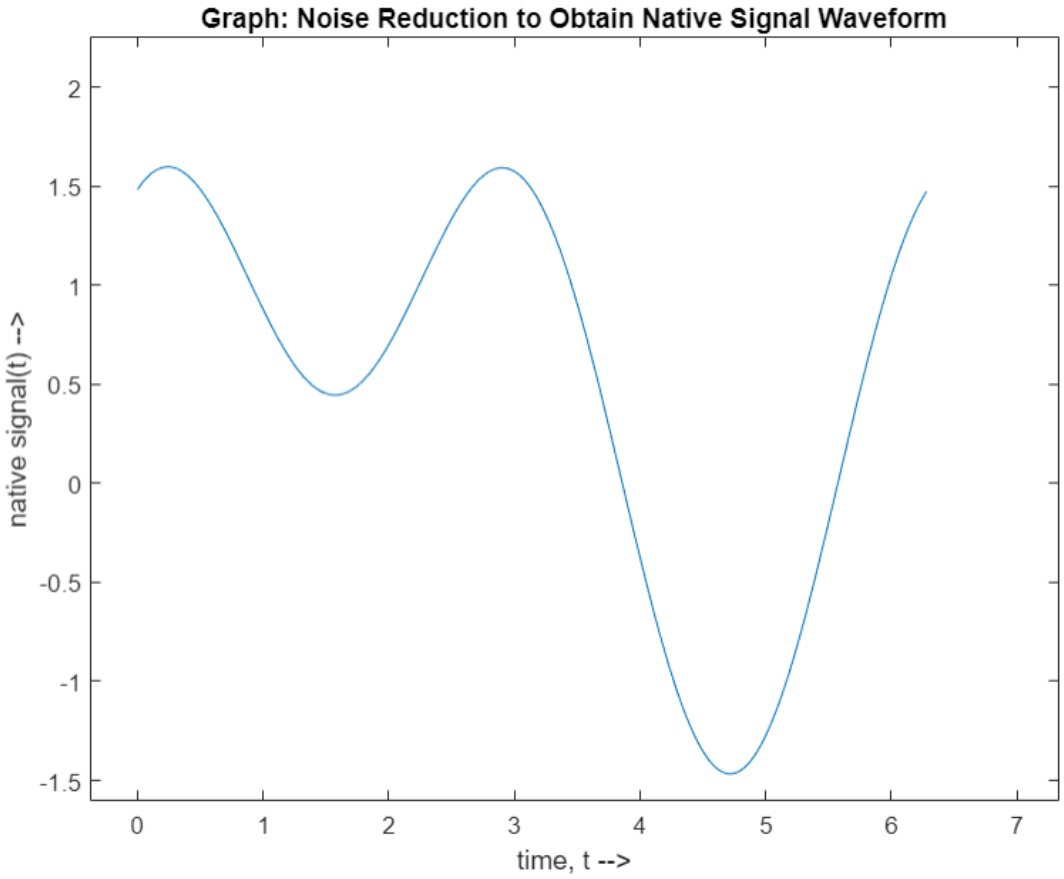
The Frequency Magnitude Response Graph is thus modified to cancel out all noise from the lower amplitude bands, as shown below:



As seen in the above graph, the x-axis is the time axis, t and the y-axis is the response. This is the response highlighting the low amplitude noise that has been detected in the transmitted signal waveform and hence have been eliminated to obtain a straight response, as they cease to exist in this graph.

4.1.4 Inverse Discrete Fourier Transform using ifft()

Now that the noise has been removed from the input signal waveform, we can proceed to obtain the native signal waveform from the acquired frequency magnitude response. To obtain this, we must perform inverse Discrete Fourier Transformation on the signal vector using the Inverse Fast Fourier Transform Algorithm that inversely performs what the fft() algorithm did in the previous subsections.



As seen in the above graph, the x-axis is the time axis, t and the y-axis is the noise-free signal resembling the native signal we started out with. This ensures that we have successfully recovered our target signal waveform after deliberate noise influx into the transmitted input signal waveform, using Fast Fourier Transform Algorithm which uses Discrete Fourier Transform as at means to undergo seamless noise reduction of Received Signal waveforms.

4.2 MATLAB Code Implementation

The above Methodology has been used in proceeding with the project to code in MATLAB. MATLAB coding has helped in the signal processing immensely due to its easy to understand UI and its Visual Graphs to depict our progress in each step of the signal processing task of noise corruption and noise reduction for restoration of native signal waveform.

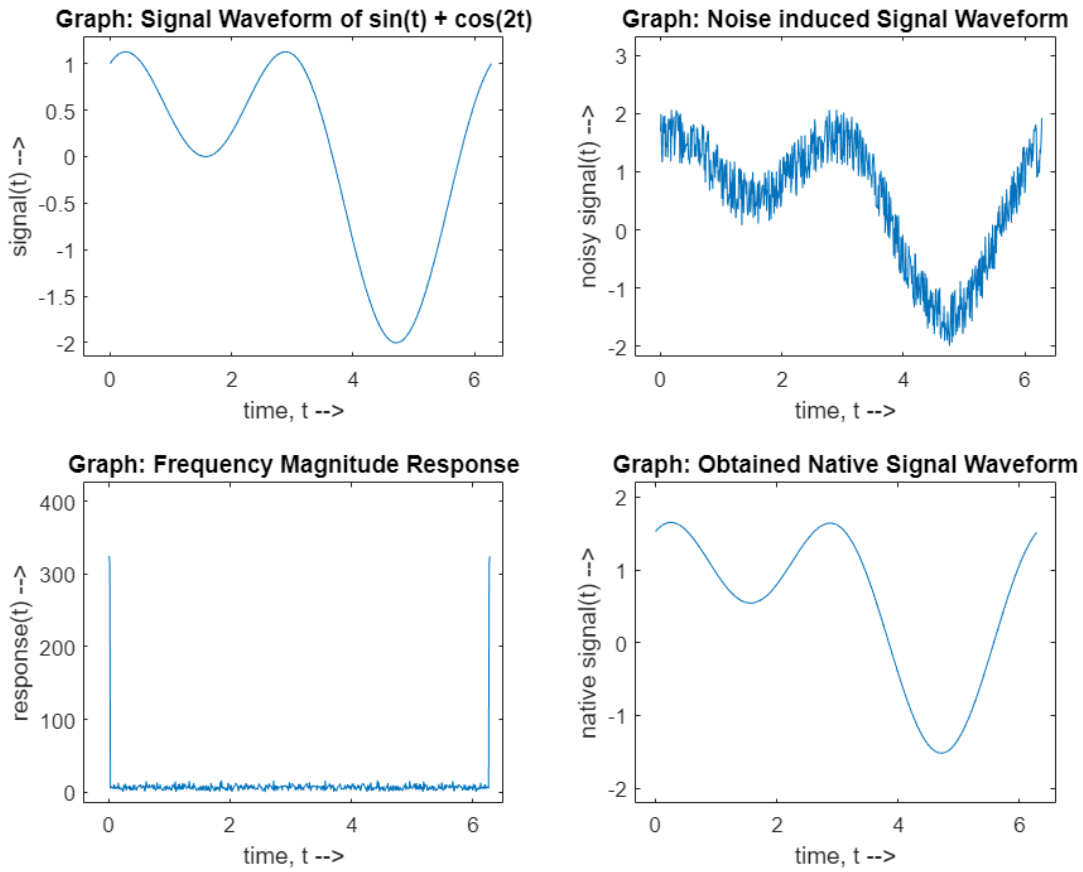
The following is a Code Snippet from MATLAB performing the required Noise Reduction process using DFT:

4.2.1 MATLAB CODE:

```
time = 0:0.01:2*pi;
    signal = sin(time) + cos(2*time);

    subplot(2, 2, 1);
    plot(time, signal);
    title("Graph: Signal Waveform of sin(t) + cos(2t)");
    xlabel("time, t -");
    ylabel("signal(t) -");

    randnoise = rand(1, length(time));
    noisySignal = signal + randnoise;

    subplot(2, 2, 2);
    plot(time, noisySignal);
    title("Graph: Noise induced Signal Waveform");
    xlabel("time, t -");
    ylabel("noisy signal(t) -");

    response = fft(noisySignal);

    subplot(2, 2, 3);
    plot(time, abs(response));
    title("Graph: Frequency Magnitude Response");
    xlabel("time, t -");
    ylabel("response(t) -");

    noise = find(abs(response) ¡ 50);
    response(noise) = zeros(size(noise));

    nativesignal = ifft(response)

    subplot(2, 2, 4);
    plot(time, nativesignal);
    title("Graph: Obtained Native Signal Waveform");
    xlabel("time, t -");
    ylabel("native signal(t) -");
```

The MATLAB codes to plot the Noise Cancelled Frequency Magnitude Response has been commented out, but can be easily plotted back if required.

4.2.2 MATLAB Plot Window:



4.3 Explanation

4.3.1 Signal Transmission Setup

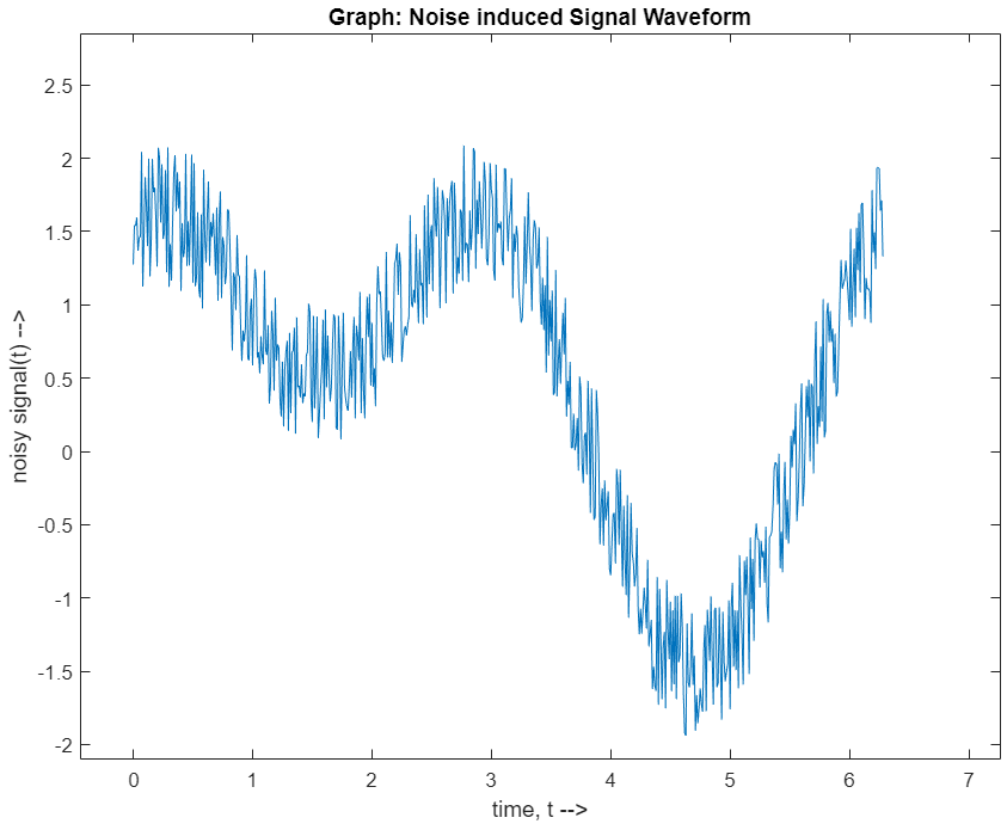
The signal to be hypothetically transmitted has been given a time axis from 0 to $2pi$ with a sampling interval of 0.01 for a smoother graph plot. we take signal vector called signal as a function of time

$$signal = \sin(t) + \cos(2t)$$

This is a rather simple sine cosine function plot taken for signal transmission, but is effective to explain noise reduction as it contains both sine and cosine signals. The given signal function is hence to undergo Discrete Fourier Transform, DFT via fft() MATLAB function, but first, it has to be infested by noise.

4.3.2 Noise Influx

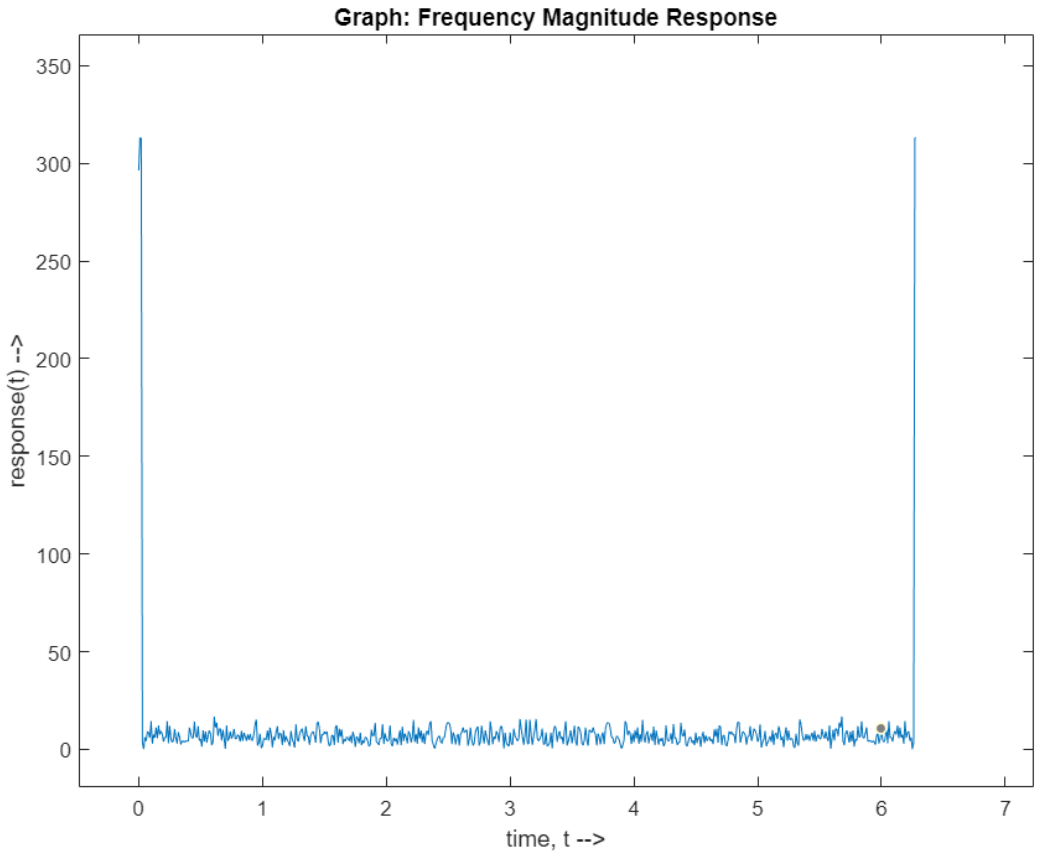
After acquiring the given native signal waveform, we use rand() function in MATLAB to generate a random function of vector for the sole purpose of Noise Influx into the Input Signal Waveform. This generates a successful Noisy Signal Waveform with minutely varying amplitudes, hence a perfectly noise-induced signal waveform as shown:



As seen in the above graph, the x-axis is the time axis, t and the y-axis is the noisy signal axis with respect to time. This can now be considered as the noise-infested signal waveform to be subjected to Fast Discrete Fourier Transform algorithms to enable native signal waveform restoration.

4.3.3 Performing DFT using fft()

Using the fft() function in MATLAB, we perform DFT with ease in the given noise induced signal waveform. We then obtain the Frequency Magnitude Response of the Noisy Signal after having performed fft() on the signal, as shown below:

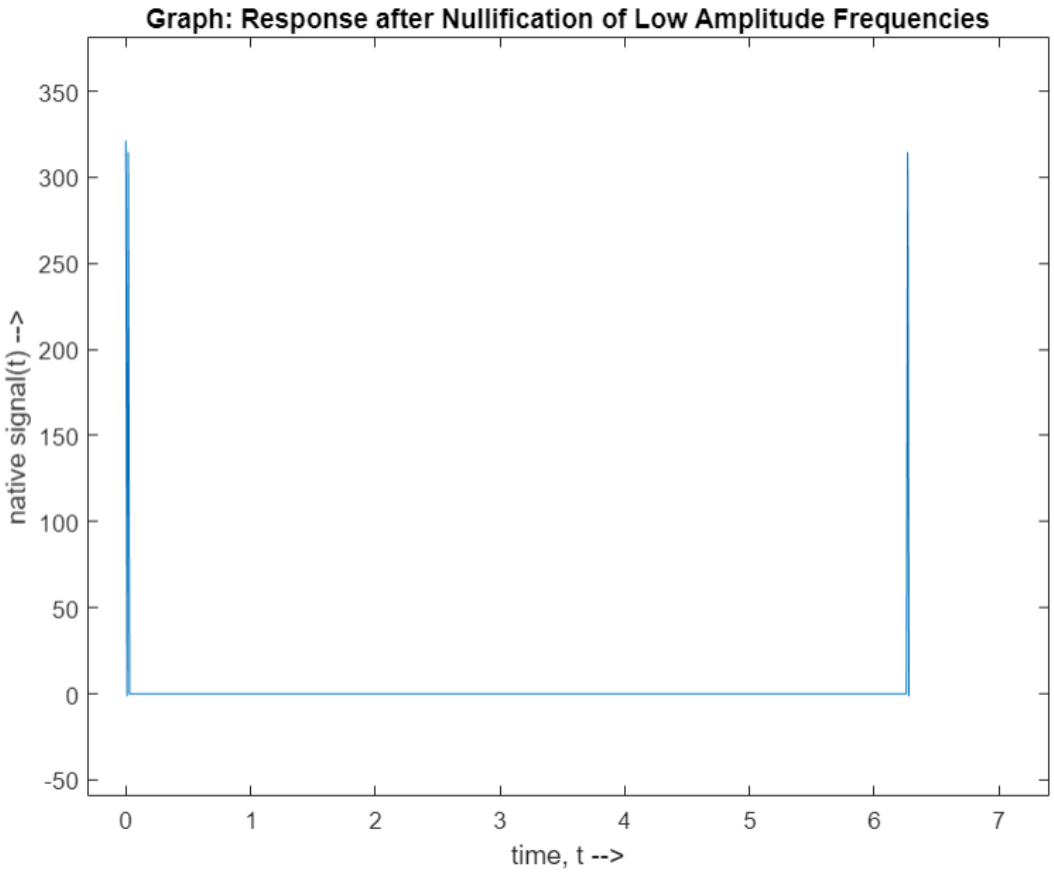


As seen in the above graph, the x-axis is the time axis, t and the y-axis is the response. This is the response highlighting the low amplitude noise that has been detected in the transmitted signal waveform and hence has to be eliminated to obtain native signal waveform.

The noise obtained is plotted by using plot on the absolute value of response, as we are not concerned with the negative values of the response plot.

Now that we have quantify-ably obtained the low amplitude frequencies of the noise induced signal waveform, we easily nullify our findings from the signal to obtain the required native target signal waveform. This is done using zeros() MATLAB function to find the low amplitude frequencies and then turn them into 0s (nullify).

The Frequency Magnitude Response Graph is thus modified to cancel out all noise from the lower amplitude bands, as shown below:

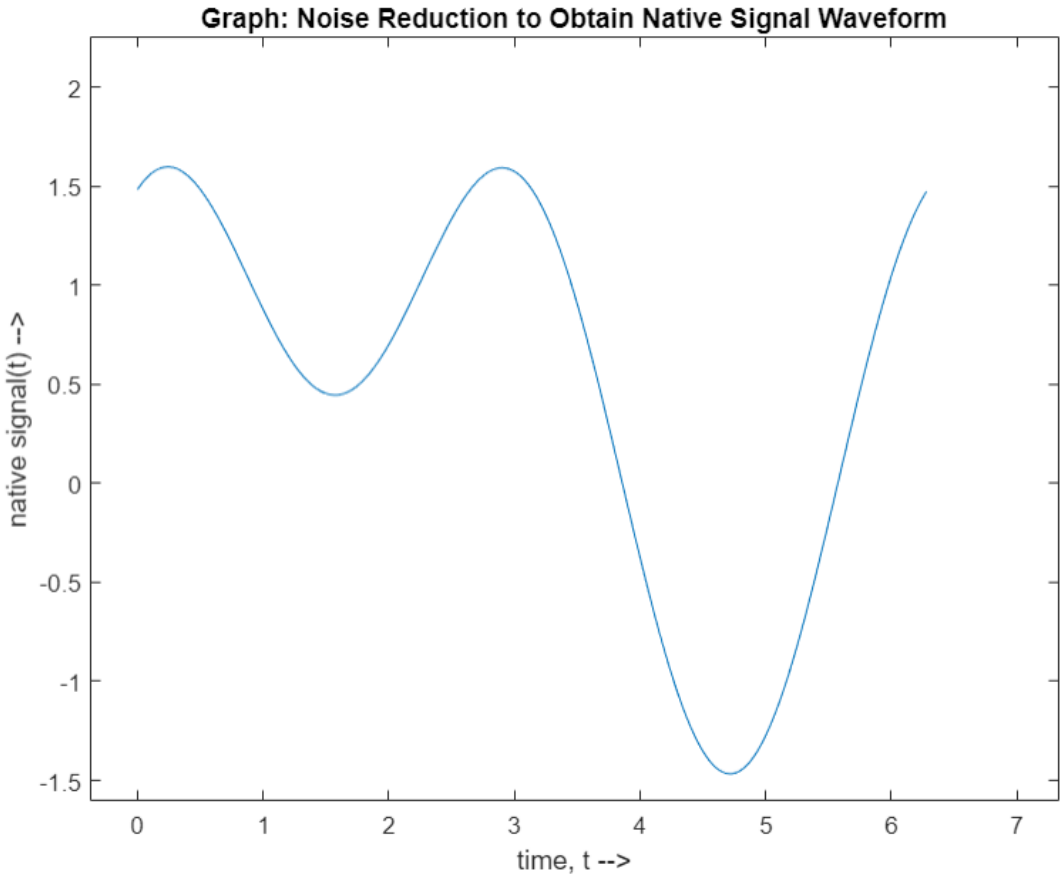


As seen in the above graph, the x-axis is the time axis, t and the y-axis is the response. This is the response highlighting the low amplitude noise that has been detected in the transmitted signal waveform and hence have been eliminated to obtain a straight response, as they cease to exist in this graph.

We use 50 as a thresh hold amplitude to determine whether or not to nullify amplitudes below the thresh hold of 50. It is usually advised to calculate the average of the amplitudes in case of more complex signal transmissions, but in this case, 50 was a safe bet, anything lesser would have resulted in partially noise reduced signal waveform.

4.3.4 Performing Inverse DFT using ifft()

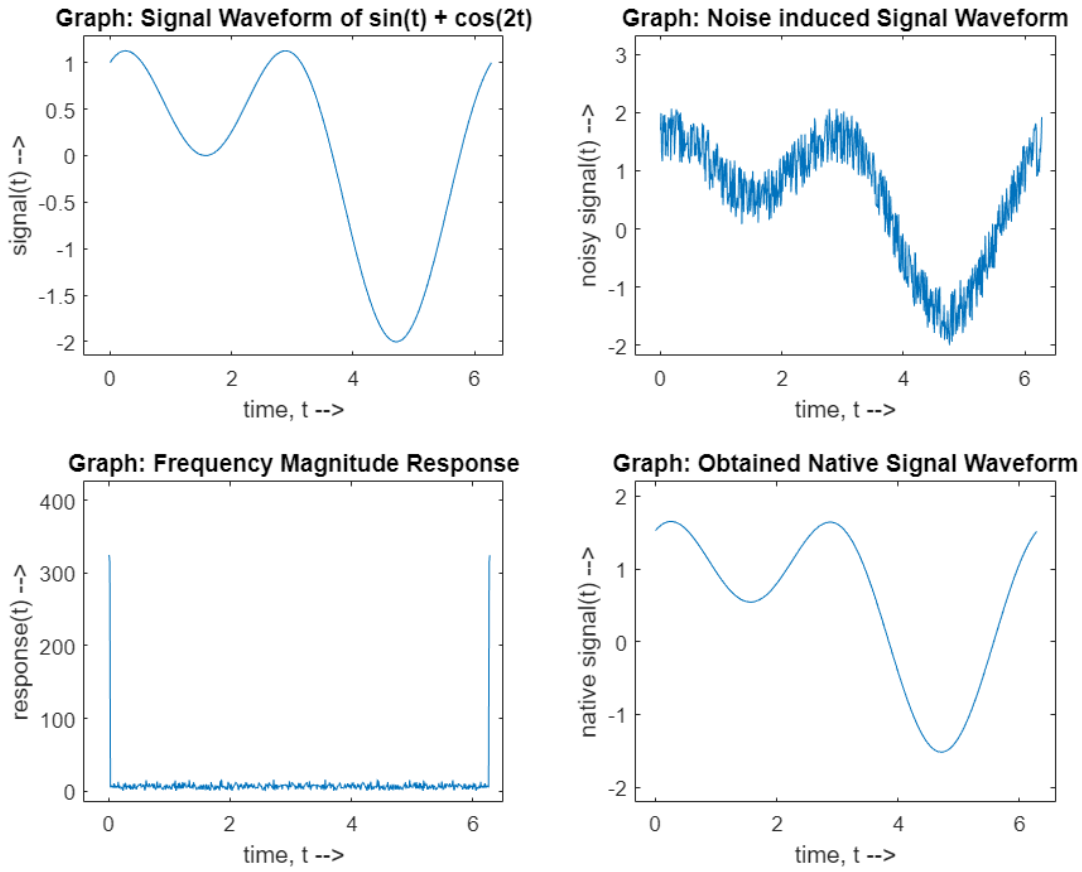
Then inverse Discrete Fourier Transformation was performed on the signal vector using the Inverse Fast Fourier Transform Algorithm that inversely performs what the fft() algorithm did in the previous subsections.



As seen in the above graph, the x-axis is the time axis, t and the y-axis is the noise-free signal resembling the native signal we started out with. This ensures that we have successfully recovered our target signal waveform after deliberate noise influx into the transmitted input signal waveform, using Fast Fourier Transform Algorithm which uses Discrete Fourier Transform as at means to undergo seamless noise reduction of Received Signal waveforms.

4.3.5 Final MATLAB plot

The subplot function was used judiciously to accommodate all the acquired MATLAB plots into a single plot window with decent titles and labels. The following subplot arrangement shows how the signal underwent all the above mentioned steps to finally come back to its native waveform:



4.4 Importance

Noise Reduction has proven to be crucial when it comes to seamless signal transmission over long distances. Being able to revert back to the original signal waveform after rigorous compression and expansion helps prevent huge data loss scenarios. Noise Reduction hence plays a pivotal role in making sure input signal waveform is intact and ready for analysis once received at the required destination receiver end. Noise reduction to recover a target signal from an input waveform is hence important. We usually use a frequency spectrum to remove noise from the input waveform. Although it is difficult to distinguish a signal from the noise in the time domain, this task tends to become easier in the frequency domain.

Signal conversions may become tougher as the transmission gets more complex. Sometimes, signals may not be in simple terms of sines and cosines. In those cases, different algorithms must be used in order to reduce noise and try to achieve a higher percentage of cleaner signal waveform. In

such cases, many other factors also come into play, such as noise infestation means, optimal noise reduction algorithm, etc.

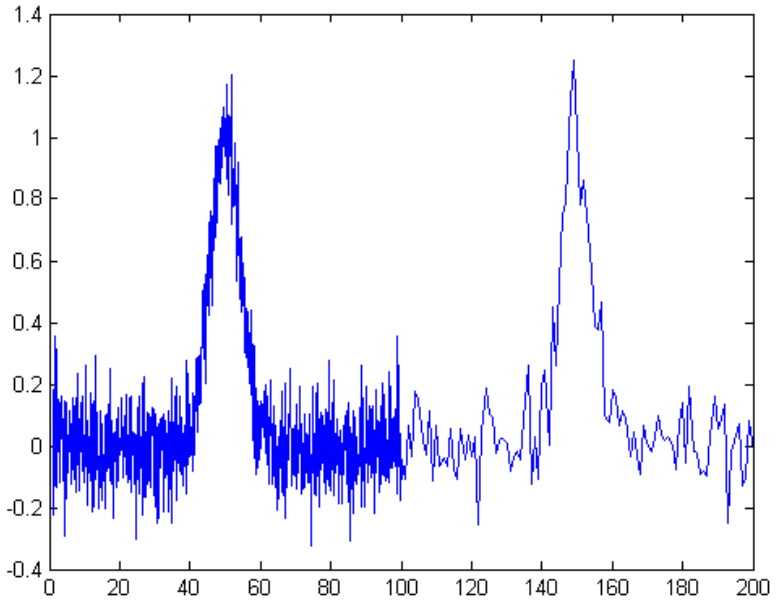
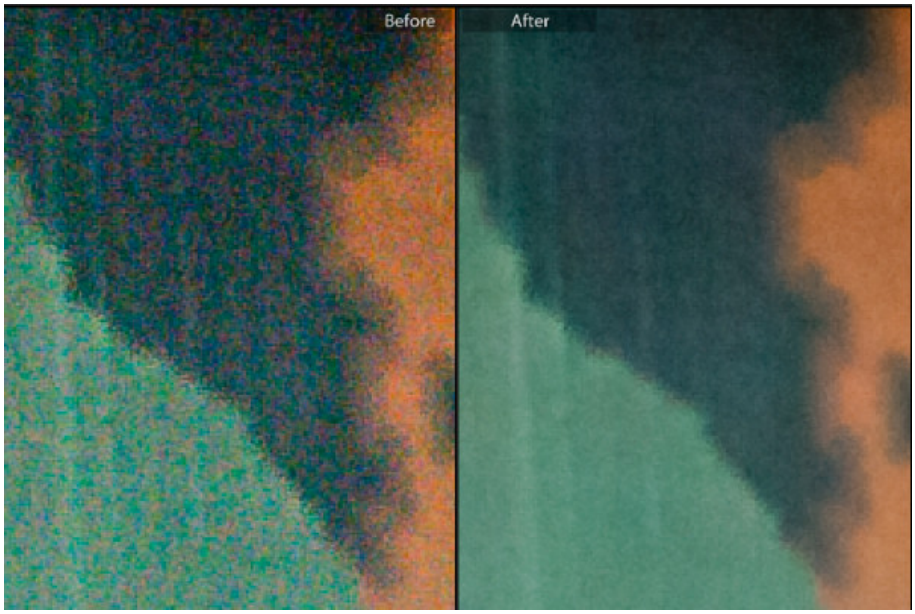


Image Noise Reduction may be a different criteria over all, but it also faces the same problem of poor compression and data loss due to noise infestation. Many other De-Noising Algorithms exist for Image Noise Reduction using Filtering Techniques, Image Processing Tools, etc. Huge Data loss in prevented, especially since images are the new big data currency powering many data analytical machine learning algorithms that depend on seamless noise free images for their training and efficiency.



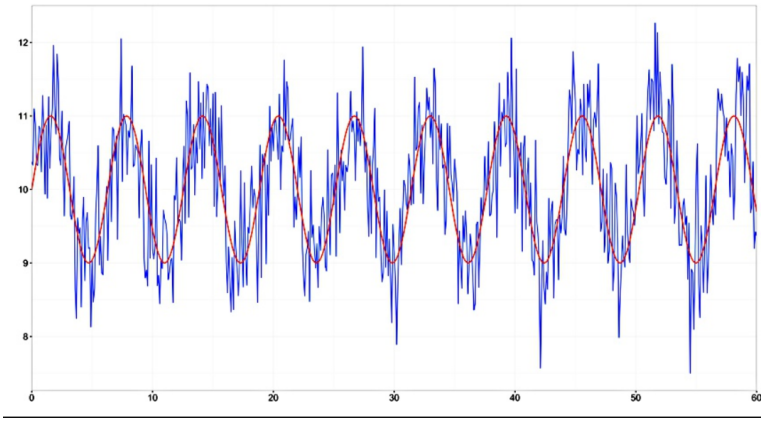
Chapter 5

Real World Problems and Applications of Noise Reduction

5.1 Noise Reduction Problems

Digital signal processing is widely used to manipulate, modify, enhance or filter signals such as speech, audio, image and telecommunication signal. The audio signal noise reduction has become a fundamental area of research for many real world applications. It is vital for engineers of inserting paradigms, processors, and tools focusing on the audio noise reduction system which is the system that's accustomed take away the noise from the audio signal.

A common challenge faced in data analysis is, in signal processing, how to isolate noise from the pre-existing signal with maximum efficiency and optimisation in the resulting signal waveform.



The plot above exemplifies an observed signal (in blue) with noise and the underlying signal without noise (in red).

Unlike the example above, which is amenable to visual analysis, in most real life problems of Noise Reduction in Signal Processing, filtering the noise to determine the signal is not feasible via simple visual analysis. More importantly, given the volume of the number of time series, it is not practical to carry out visual analysis. It is imperative to carry data analysis in an algorithmic approach.

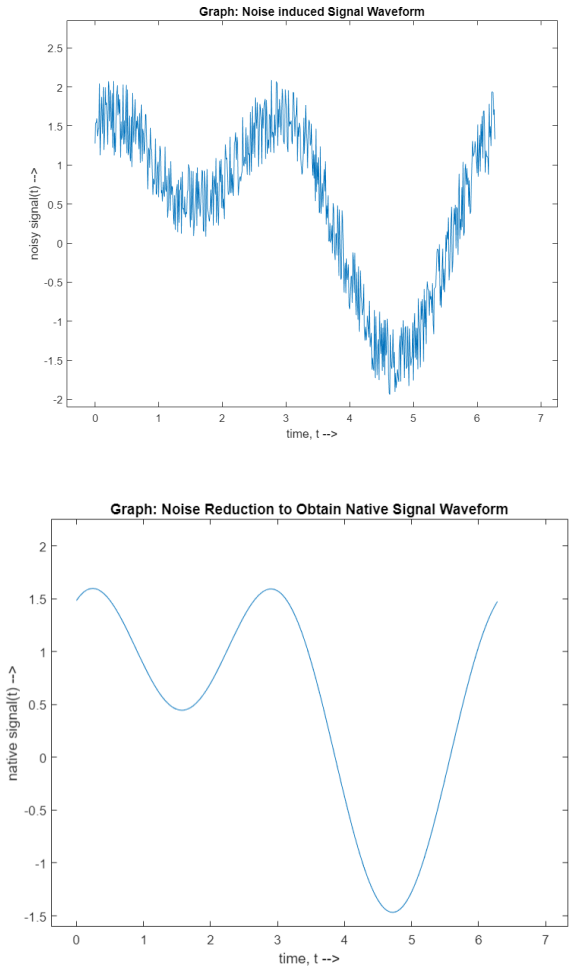
5.2 How to Reduce Noise (TIME DOMAIN)

Over multiple decades of dedicated work in the field of Signals and Data Communication, a large amount of progress has been made to improve the signal-to-noise ratio (SNR). Noise reduction plays a key role is large set of applications beyond operations, e.g., image/audio/video processing. A wide variety of filters have been proposed to address noise reduction. Broadly speaking, filters can be classified into two categories:

- Low pass filter: It passes signals with a frequency lower than a certain thresh-hold frequency and attenuates signals with frequencies higher than the thresh-hold frequency. In the context of a time series, a simple moving average (SMA) exemplifies a low pass filter.
- High pass filter: It passes signals with a frequency higher than a certain thresh-hold frequency and attenuates signals with frequencies lower than the thresh-hold frequency.

5.3 How to Reduce Noise (FREQUENCY DOMAIN)

Noise reduction can be achieved in both the time domain as well as frequency domain. In case of the frequency domain noise reduction, Fourier Transform of the given signal is obtained and subsequently an appropriate filter is applied. This portion in the vast possibilities of noise Reduction has been explained in details in this mini Project with diagrams, plots and MATLAB coding.



The transition can be clearly seen above, using DFT to reduce Noise from Noisy Transmitted Signal waveform to obtain the noise-free Signal Waveform.

5.4 Applications of Noise Reduction

- Noise can infest any given signal while transmission in a lot of different ways. The compression and transmission of signals across large distances is rendered meaningless if the receiver is not able to correctly identify and analyse the required signal. This leads to huge cases of data loss in many data communication projects that rely on long distance signal transmission. Noise Reduction hence plays a pivotal role in making sure input signal waveform is intact and ready for analysis once received at the required destination receiver end. Noise reduction to recover a target signal from an input waveform is hence important.

- Noise Corruption is not just a problem pertaining to Audio Signal Transmission. Images have large cases of Noise Influx too. Salt and

pepper noise, Gaussian noise, etc. have a history of corrupting quality image-driven data during transmission. This leads to large scale data loss, as images contribute to a bulk in the big data spectrum, and hence, are crucial to preserve for better understanding of data and all its corollaries. hence Image Noise Reduction plays a huge part in eliminating data corruption.

- Noise Reduction, especially in the field of image processing, drastically improves the quality of numerous machine learning and deep learning models based on big data, including audio signals and image data. This helps predict more accurate Artificial Intelligence based machinery that have proven to be crucial in many industries across the globe.

Fig.: Image with Gaussian Noise

Chapter 6

Conclusion

Noise can infest any given signal while transmission in a lot of different ways. The compression and transmission of signals across large distances is rendered meaningless if the receiver is not able to correctly identify and analyse the required signal. This leads to huge cases of data loss in many data communication projects that rely on long distance signal transmission. Noise Reduction hence plays a pivotal role in making sure input signal waveform is intact and ready for analysis once received at the required destination receiver end. Noise reduction to recover a target signal from an input waveform is hence important.

Noise Reduction has proven to be crucial when it comes to seamless signal transmission over long distances. Being able to revert back to the original signal waveform after rigorous compression and expansion helps prevent huge data loss scenarios. Signal conversions may become tougher as the transmission gets more complex. Sometimes, signals may not be in simple terms of sines and cosines. In those cases, different algorithms must be used in order to reduce noise and try to achieve a higher percentage of cleaner signal waveform. In such cases, many other factors also come into play, such as noise infestation means, optimal noise reduction algorithm, etc.

Hence, in this project, we have used Fast Fourier Transform, fft() to perform Discrete Fourier Transform on the given time vector related transmitted noise-corrupted signal waveform. This enables us to cipher out the low amplitude frequencies once we plot out the noisy response. This plot is now at our hands for modification. We then simply eliminate this noise using zeros() and return the response to the signal. We then make the signal undergo Inverse Discrete Fourier Transform using Inverse Fast Fourier Transform, ifft() in MATLAB. We then obtain the original transmitted signal we began with before randomizing its waveform using a random function to simulate noise infestation.

Chapter 7

Reference

7.1 OPEN SOURCE TOOLS:

- **MATLAB Online Open Source IDE**
  : https://matlab.mathworks.com/
- **Overleaf LaTex Editor**
  : https://www.overleaf.com/project
- **Octave Online Open Source Compiler**
  : https://octave-online.net/

7.2 PROJECT DATA MATERIAL:

- **Wikipedia** : https://www.wikipedia.org/
- **Quora** : https://www.quora.com/
- **Signals and Systems - Alan V. Oppenheim, Alan S. Willsky**
- **Lecture - Stephen P. Boyd**
- **Fourier Transform**
  https://web.eecs.umich.edu/
- **Electrical Technology**
  https://www.electricaltechnology.org/
- **Information Engineering, University of Oxford**
  : http://www.robots.ox.ac.uk/

- **Tutorials Point**
  : https://www.tutorialspoint.com/
- **Medium** : https://medium.com/
- **MATLAB documentations**
  : https://in.mathworks.com/
- **IEEE Xplore**
  : https://ieeexplore.ieee.org/Xplore/home.jsp
- **Research Gate**
  : https://www.researchgate.net/
- **Global Vision Press**
  : http://gvpress.com/journals/
- **Towards Data Science**
  : https://towardsdatascience.com/
- **Britannica** : https://www.britannica.com/
- **Google Scholars Search Engine**
  : https://scholar.google.co.in/
- **Oxford Academic Journals**
  : https://academic.oup.com/journals
- **MIT official**
  : http://web.mit.edu/
- **Wolters Kluwer Journals**
  : https://journals.lww.com/pages/default.aspx
- **Science Direct**
  : https://www.sciencedirect.com/
- **Catchpoint Blogs**
  : https://blog.catchpoint.com/