DSP Course

William Jackson

09/09/2016

Contents

1 Part 1

1.1 Trends Fitting

For the first supplied data set, the trend was a simple DC offset requiring baseline subtraction for the removal. This is shown in Figure 1.1

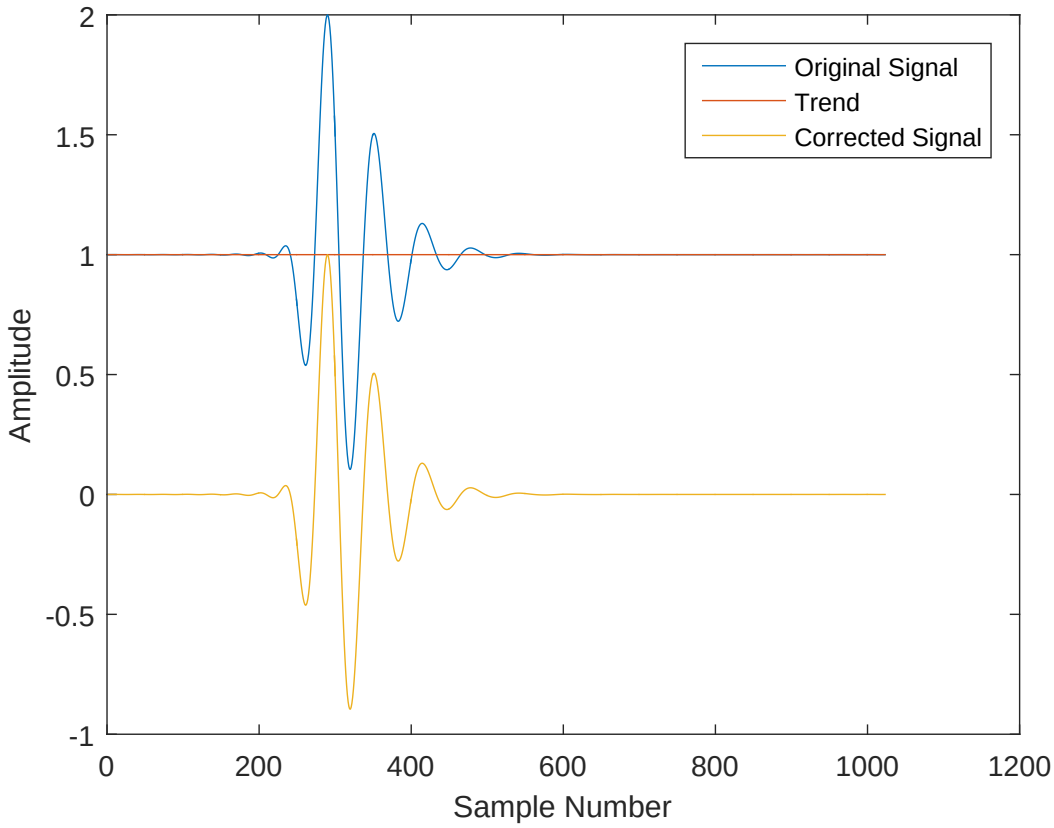


Figure 1.1: Trand Data 1

The second supplied data set contained a 1st order trend, polyfit was used to find the coefficients and this was subtracted as shown in Figure 1.2

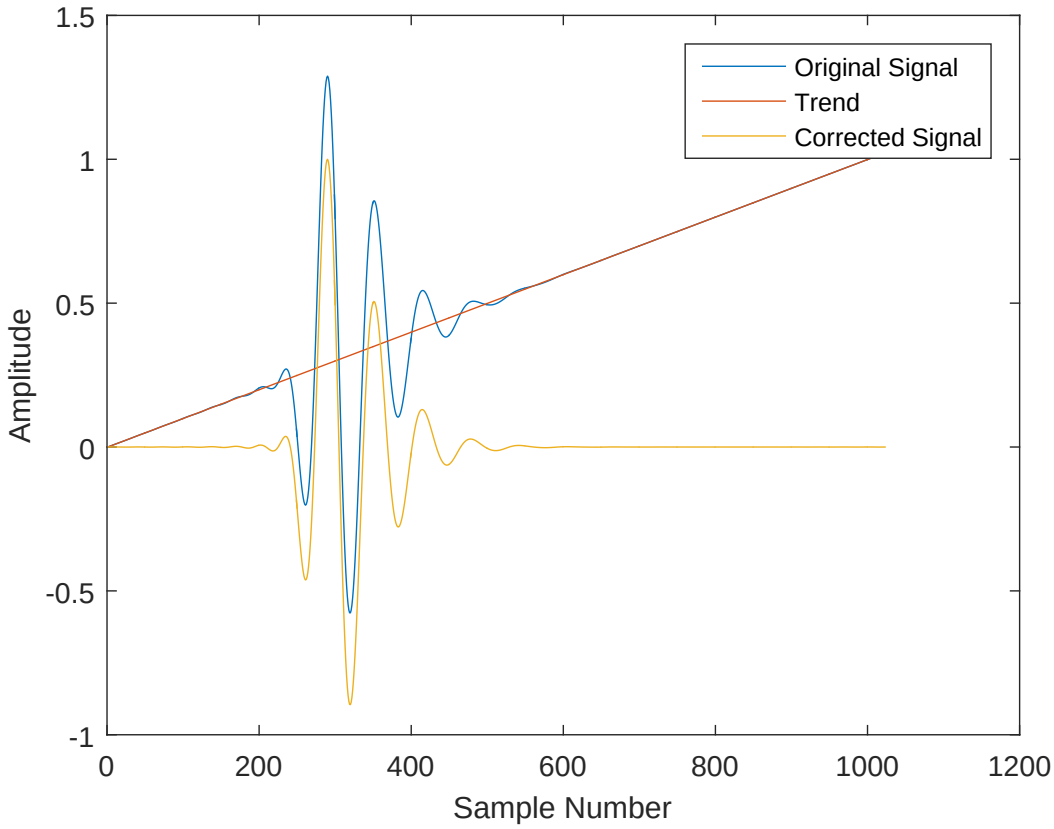


Figure 1.2: Trend Data 2

The third supplied data set contained a 2nd order trend, polyfit was used to find the coefficients and this was subtracted as shown in Figure 1.3

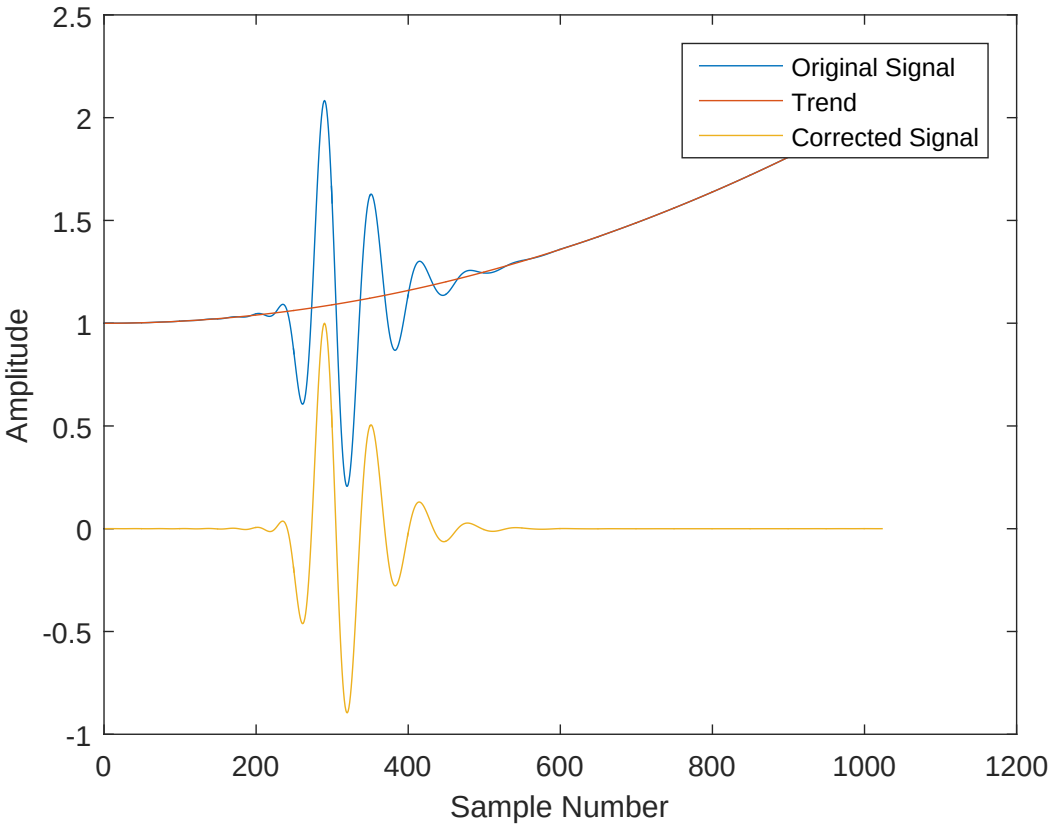


Figure 1.3: Trend Data 3

1.2 Coherent Noise

The following code was used to demonstrate the coherent averaging demonstrating the noise is reduced by a factor a $\sqrt{M}$

The error between the expected standard deviation and estimated were within an expected tolerance as shown in the following code snippet 1

Listing 1: MATLAB Coherent Averaging

```
%Ex1_2
close all
clear
N = 100;
M = 1000;
for ii =  1:M
    X(ii,:) = rand(1,N);
end
X_sd = std(X);
X_sd_mean = mean(X_sd);
X_c = sum(X)/M;
X_c_mean = std(X_c);
fprintf('Expected Value: %f \nActual Value: %f\n',sqrt(M),X_sd_mean/X_c_mean)

%Expected Value: 31.622777
%Actual Value: 28.970080
```

1.3 Correlating Signals

Two cycles of in phase cosines were integrated using point wise multiplication as shown in Figure 1.4

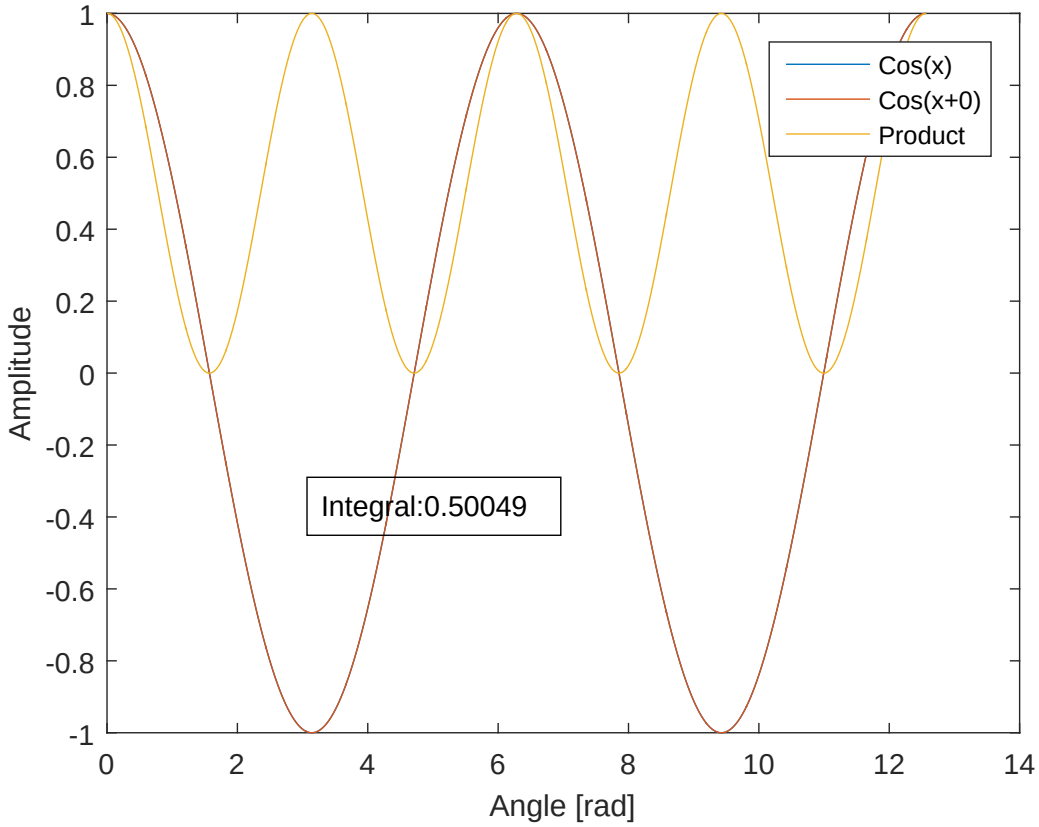


Figure 1.4: In phase cosine

This process was repeated with 20 cycles of cosine for the secondary signal, it is noted that the integral result has tended to a much lesser value as expected shown in Figure 1.5

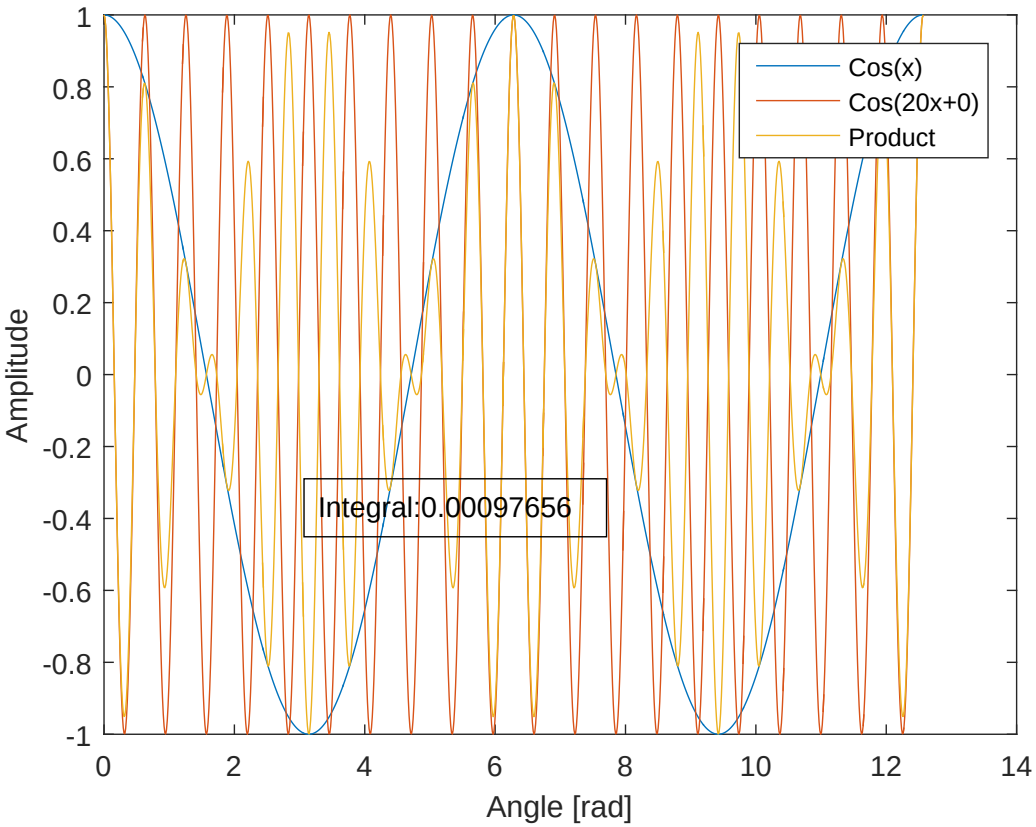


Figure 1.5: Twenty Cycles Cosine Product

Finally, this was repeated with a cosine and sinusoid as expected with a signal out of phase by 90 degrees the integral value tended to zero. As shown in Figure 1.6

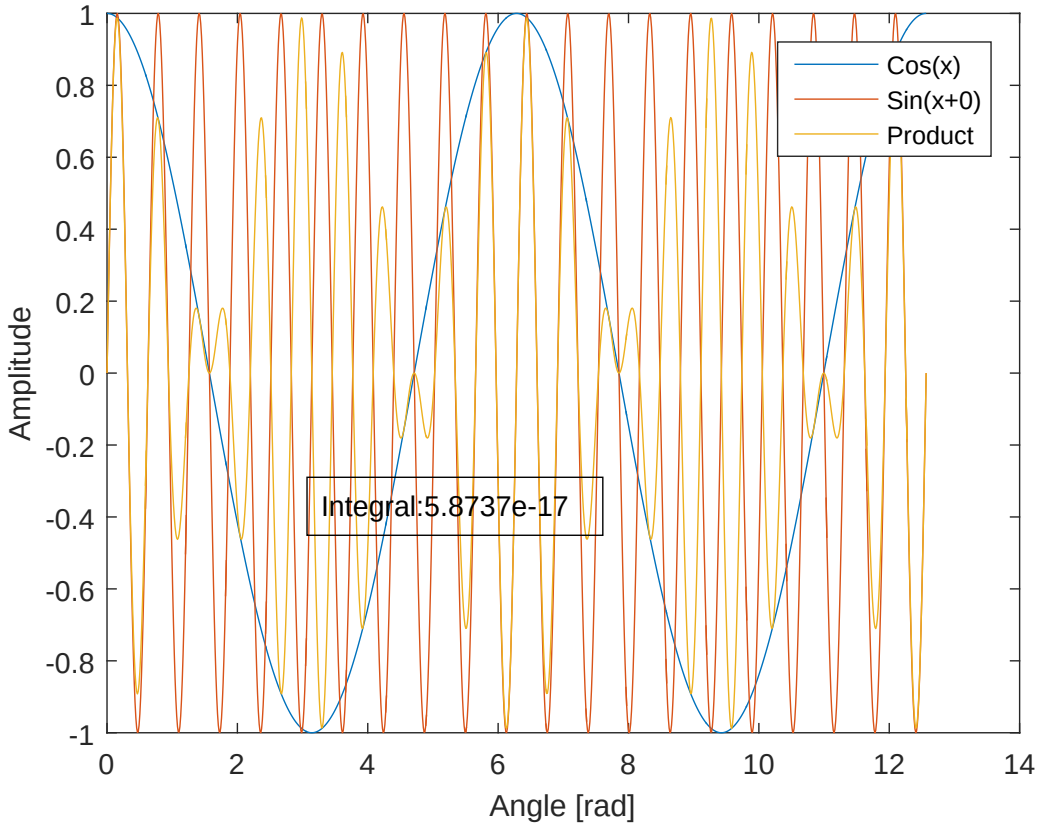


Figure 1.6: Cosine & Sin Product

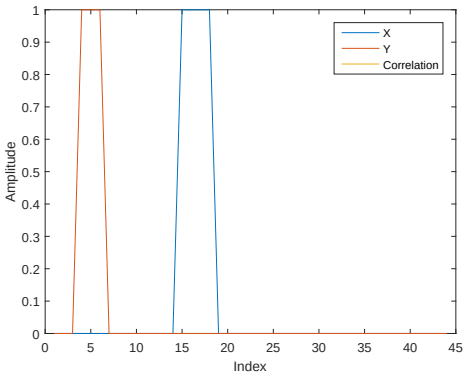
1.4 Cross-Correlation

A cross correlation algorithm was implemented, this consisted of a padding exercise where the signals are padded on both sides to twice the length of the longer signal with zeros. This followed by the summation of the dot product between signals as they are shifted over each other in a cyclic fashion. Two signals (x & y) as follows:

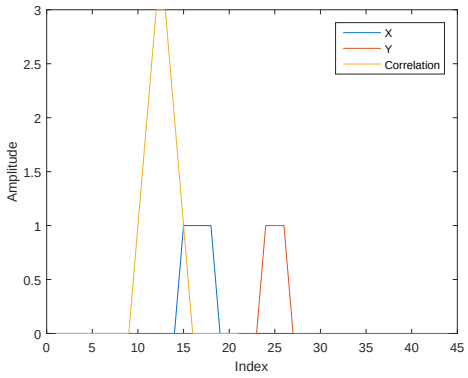
$$X = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

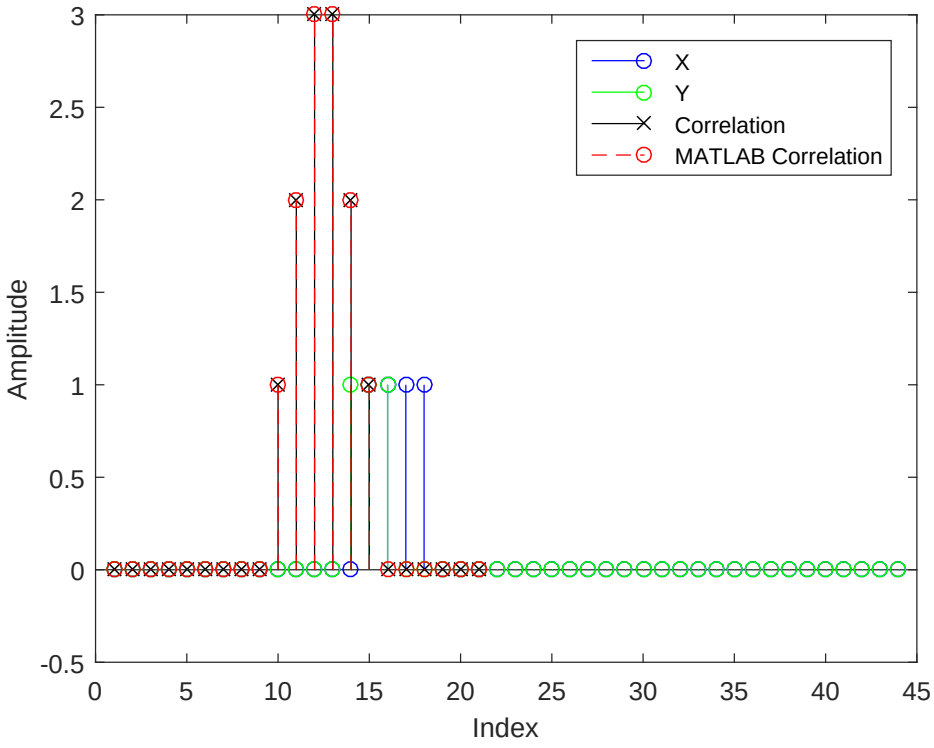
Are shown as examples for this exercise which is graphically shown with the first itteration (1.7a), final iteration(1.7b) and comparison of MATLAB's inbuilt function & the self-written code(1.7c). It should be noted that if only one variable is passed into the function autocorrelation is performed.



(a)



(b)



(c)

Figure 1.7: Correlation Results

1.5 FFT Analysis

A 16-point array of the form:

$A \cos nk\frac{2\pi}{N}$

$B \sin nk\frac{2\pi}{N}$ $(N = 16)$

Were used to demonstrate the use of the FFT. A cosine with $A = 1$ and $k = 1$ is shown in Figure 1.8. The resulting FFT is shown in Figure 1.9. The FFT illustrates the energy of the signal is located in the DFT bin corresponding with the k value. With the imaginary part being negative in the negative frequency and positive for the positive frequency. The real part is positive for both. With the A value doubled the energy within the FFT is doubled as shown in Figure 1.10.

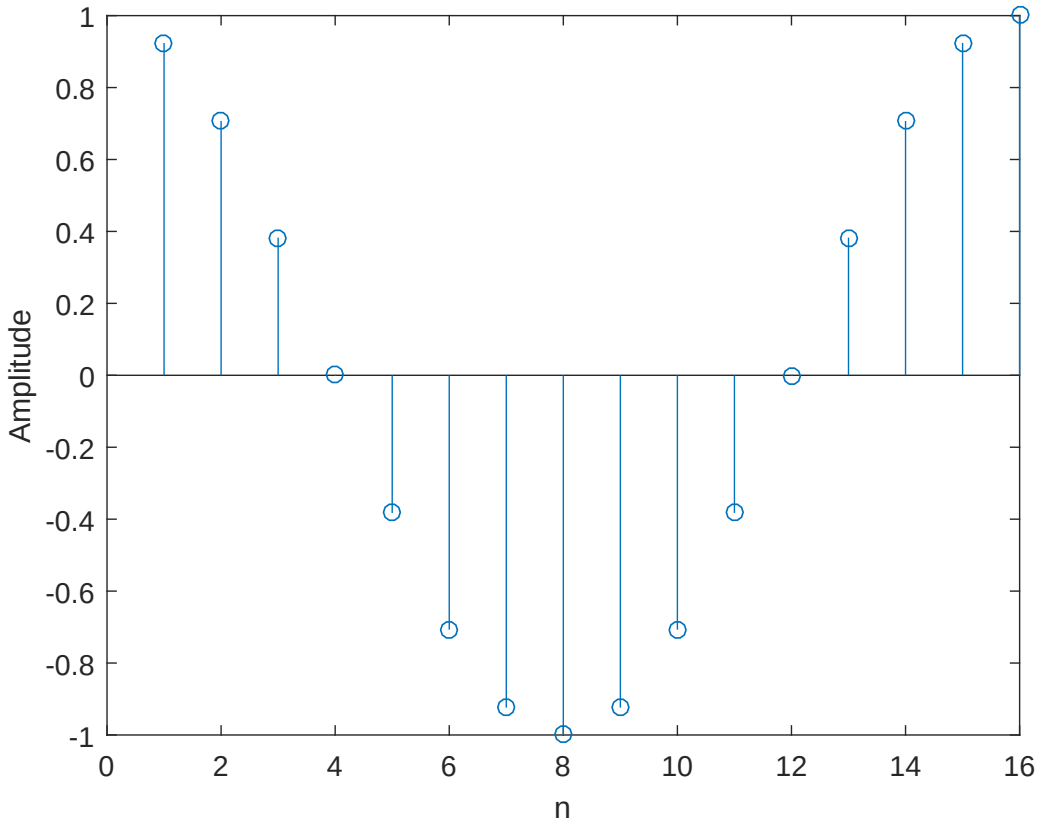


Figure 1.8: First Harmonic Cosine $A = 1$

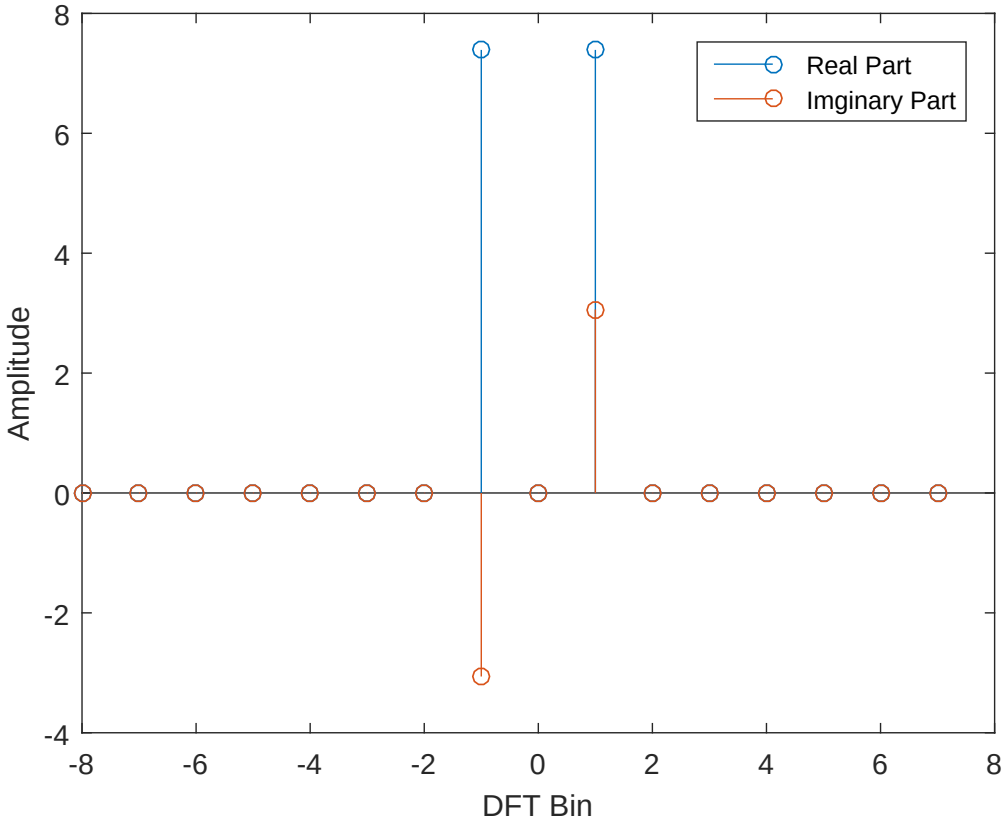


Figure 1.9: FFT of First Harmonic Cosine

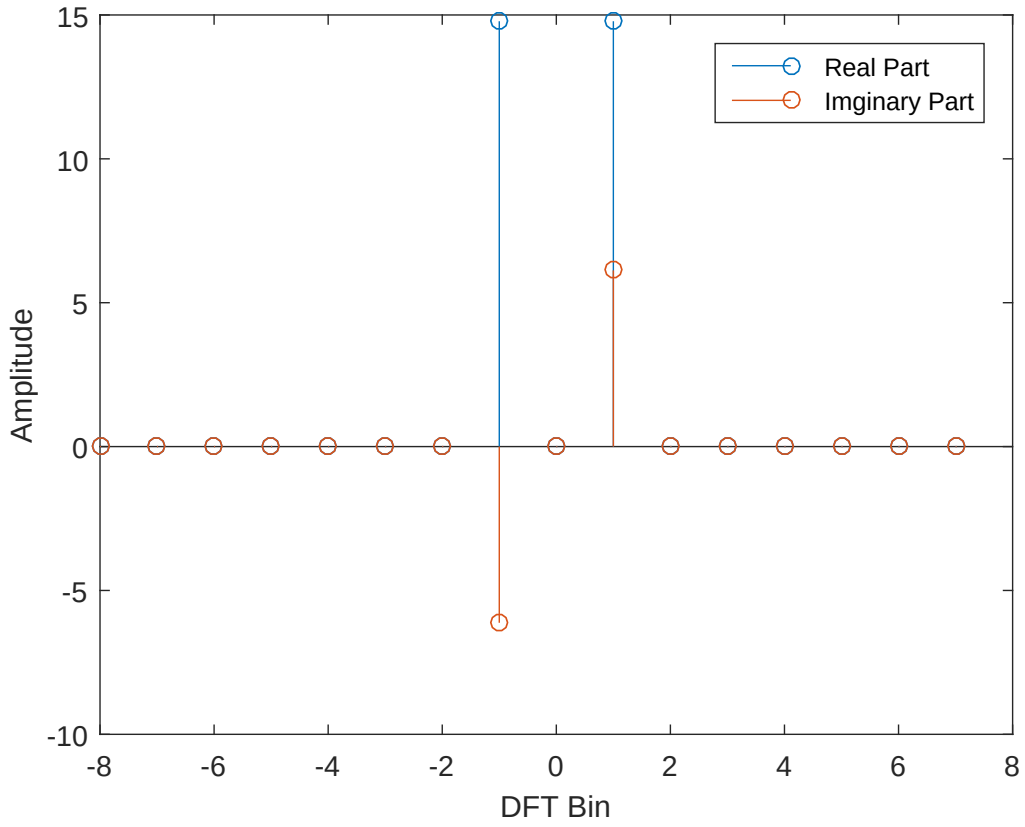


Figure 1.10: FFT of First Harmonic Cosine $A = 2$

The process was repeated for a Sinusoid with $B = 1$ and $k = 1$, shown in Figure 1.11 with the corresponding FFT shown in Figure 1.12. It is observed that the imaginary part of the sinusoidal signal is the complex conjugate of the Cosine.

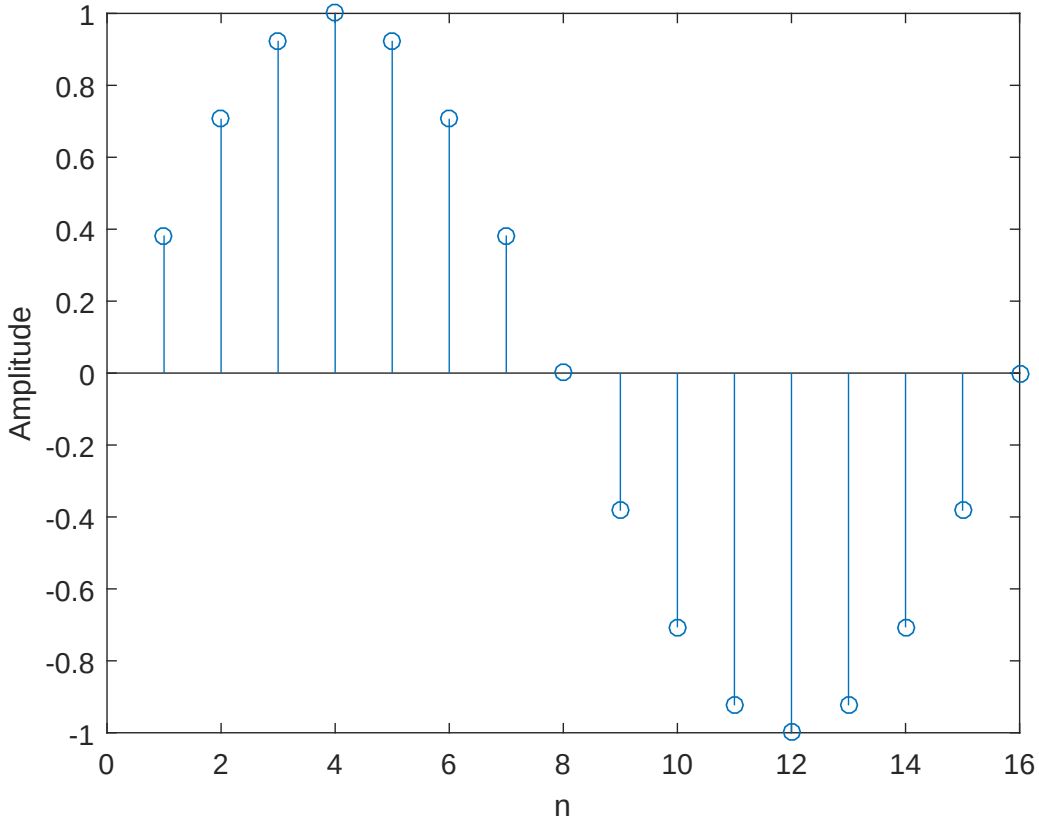


Figure 1.11: First Harmonic Sinusoid $B = 1$

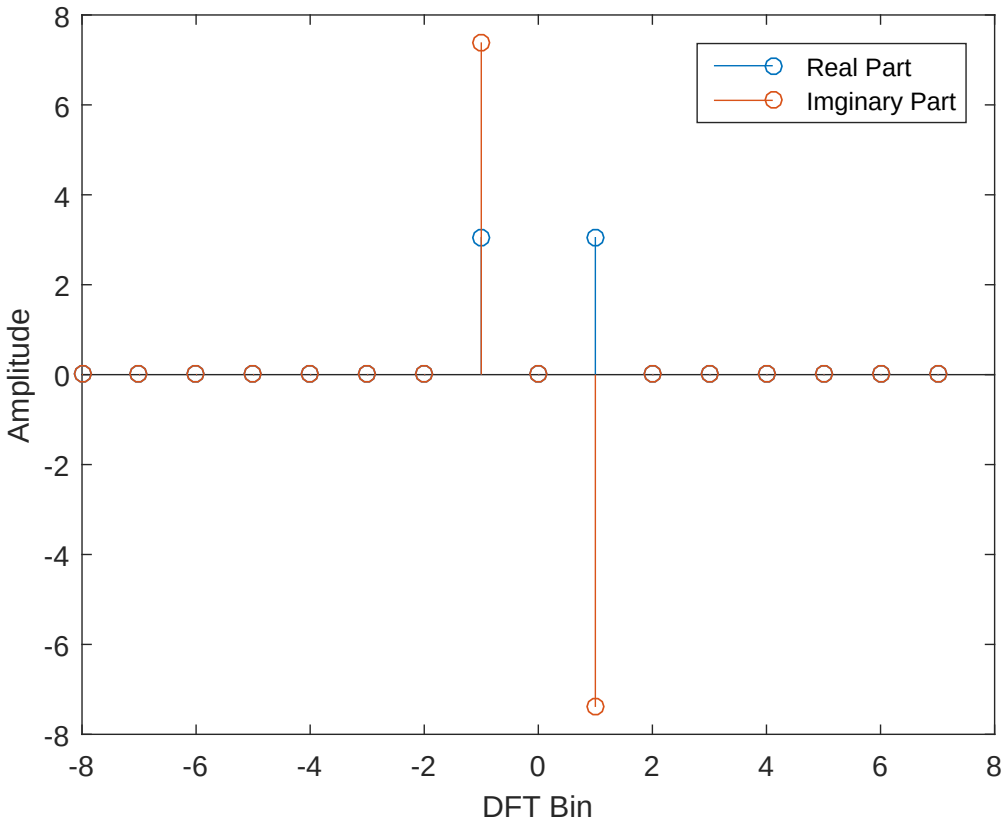


Figure 1.12: FFT of First Harmonic Sinusoid $B = 1$

As $N = 16$ the highest harmonic that can be observed is that of $k = 8$ due to Nyquist's theorem the cosine of this signal is shown in Figure 1.13 & the FFT in Figure 1.14

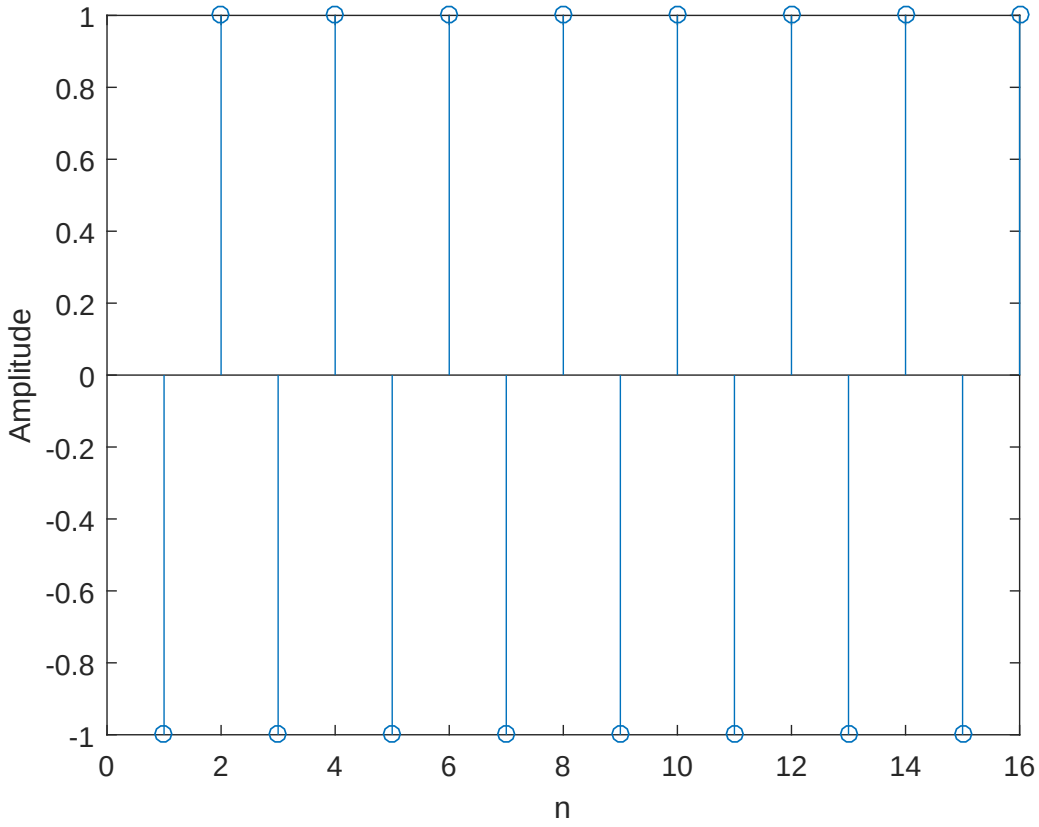


Figure 1.13: Eighth Harmonic Cosine $A = 1$

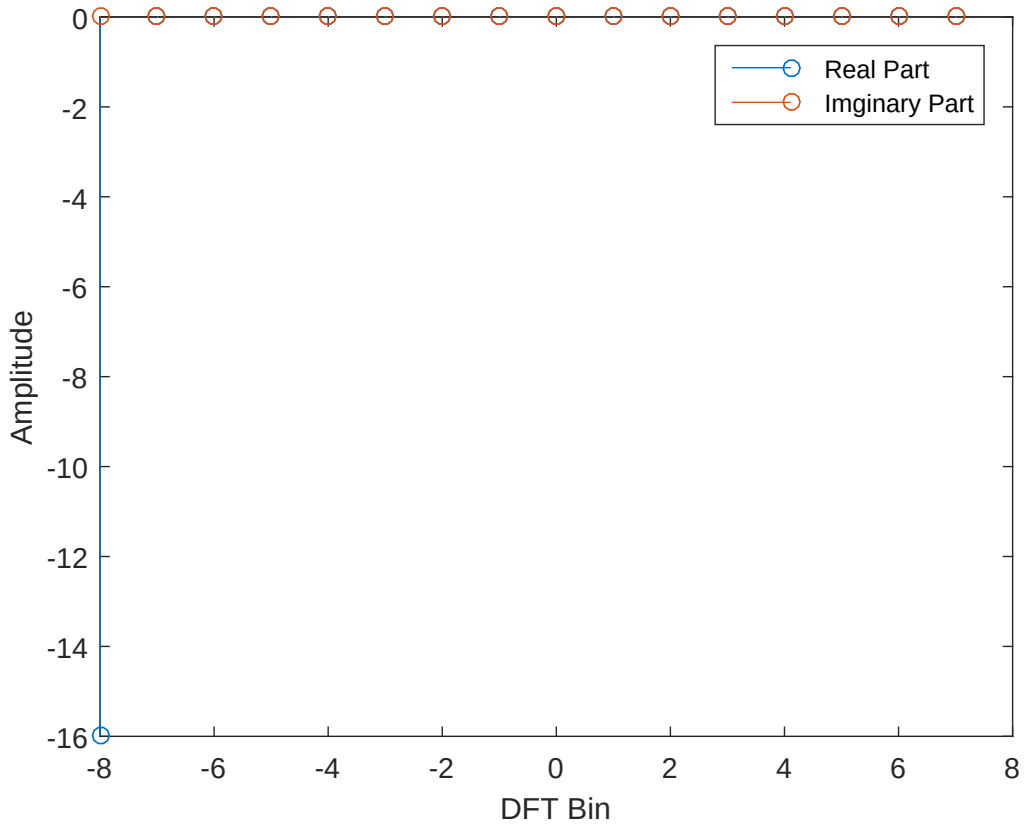


Figure 1.14: FFT of Eighth Harmonic Cosine $A = 1$

The effects of this are observed with a value of $k = 9$ the signal is observed to aliased to having energy in the 7th DFT bins shown in Figure 1.15. This is due to the FFT being continuous and repeating every N samples. With $k = 9$ being greater than $\frac{N}{2}$ the signal has wrapped around & aliased to $k = 7$.

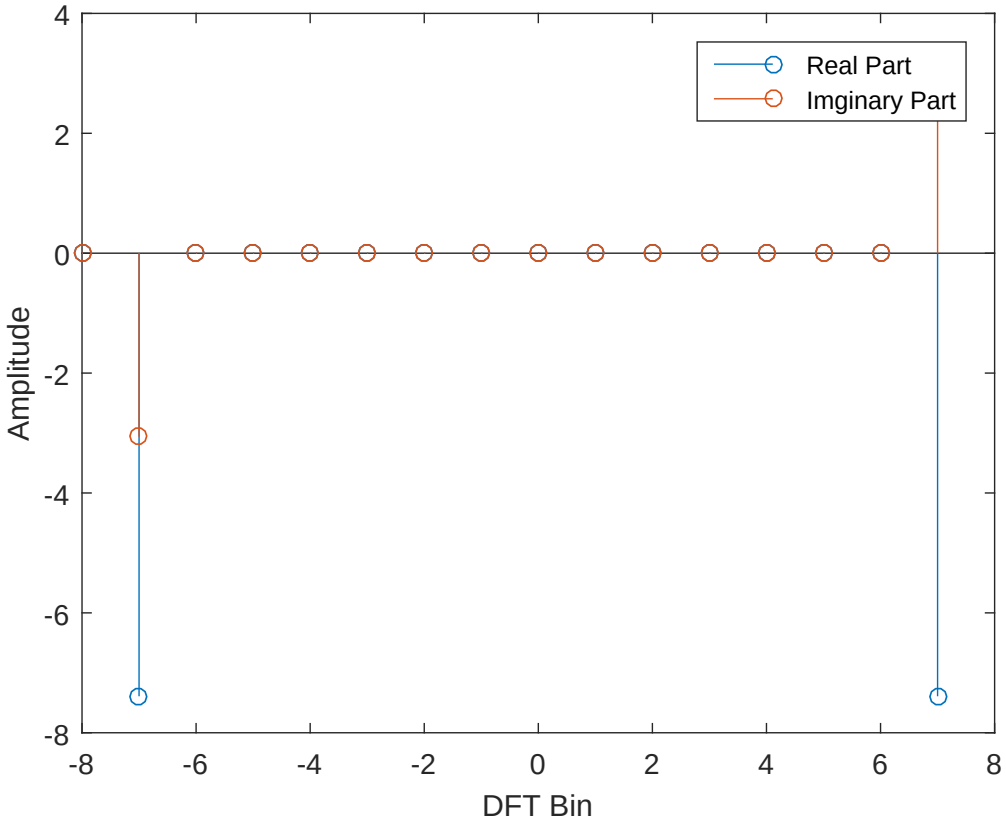


Figure 1.15: FFT of Ninth Harmonic Cosine $A = 1$

1.6 Zero Pad Interpolation in Freq. Domain

Interpolation in the time domain can be completed by padding the frequency domain signal with zero value samples. This can be thought of as adding samples into the time domain signal as the zeros are inserted at the midpoint or fs/2 which is the maximum positive frequency which has a value close to zero, the samples do not have much other effect than carrying out interpolation. This method is best documented with commented example code for interpolation of a ultrasonic signal, shown in listing 2. The resultant output is shown in Figure 1.16, it is noted that this method exhibits Gibbs' phenomenon where a signal constructed by Fourier series behaves at a jump discontinuity, this is observed in the following subsection of Figure 1.16 in Figure 1.17.

Listing 2: FFT Zero Pad Interpolation

```
% Use of the FFT interpolation
close all
clear
fs = 40*1e6;%fs known
newfs = 160*1e6; %desired fs known
ratio = newfs/fs; %calculate ratio assume even
raw = importdata('ex1_8_interp.txt'); %import data
raw_fft = fft(raw); %calculate fft
%first section of signal = first half of fft
int_fft(1:length(raw_fft)/2+1)= raw_fft(1:length(raw_fft)/2+1);
%insert number zeros to create signal of length ratio*original signal in
%the mid point of signal
%length
int_fft((length(raw_fft)/2+1):(length(raw)*ratio)-length(raw_fft)/2) = 0;
%later section of signal equals the second half of the original fft
int_fft(length(int_fft):length(int_fft)+length(raw_fft)/2) = raw_fft(length(
    raw_fft)/2:end);
%take the ifft
int_ifft = ifft(int_fft);
%take real part of signal (small errors may cause imaginary parts)
% & scale amplitude of signal by the ratio
int_sig = real(ratio*int_ifft);
%create x values for plotting
x =1:length(int_ifft);
x2 = 1:ratio:length(int_ifft);
%plot signals
plot(x,int_sig);
hold on
stem(x2,raw);
xlabel('Index')
ylabel('Amplitude')
legend('Interpolated signal','Raw Signal')
xlim([0 256])
```

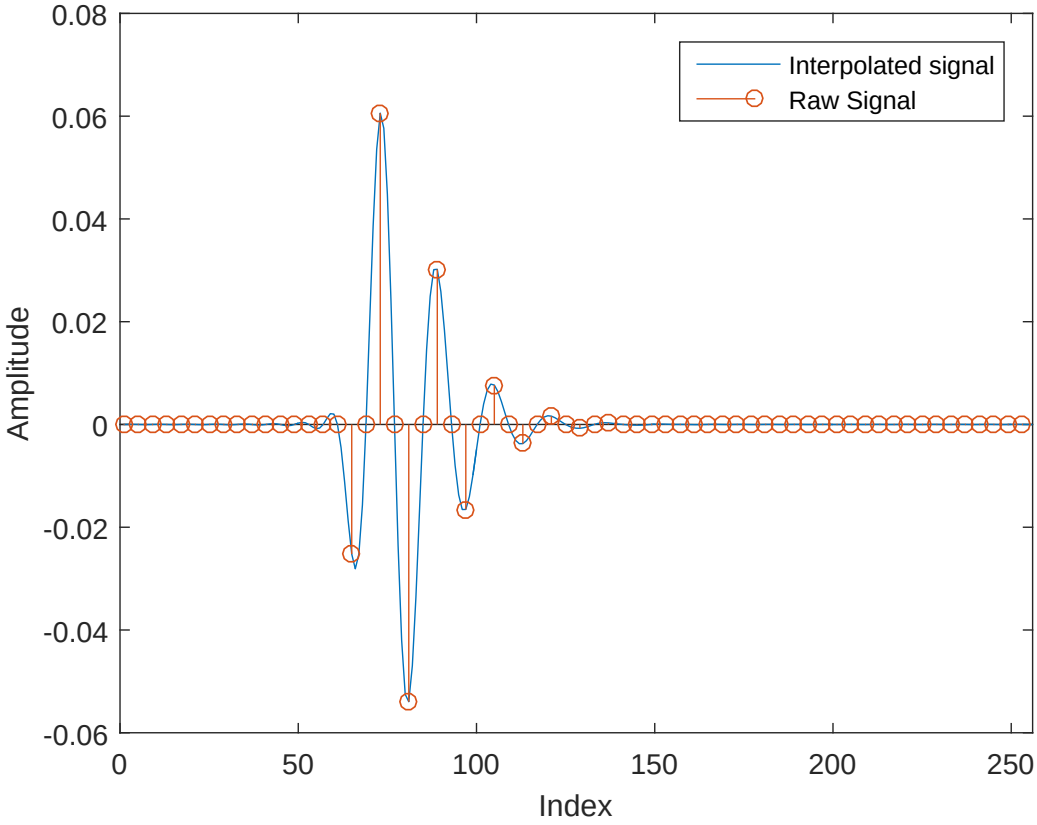


Figure 1.16: FFT Zero Pad Interpolation

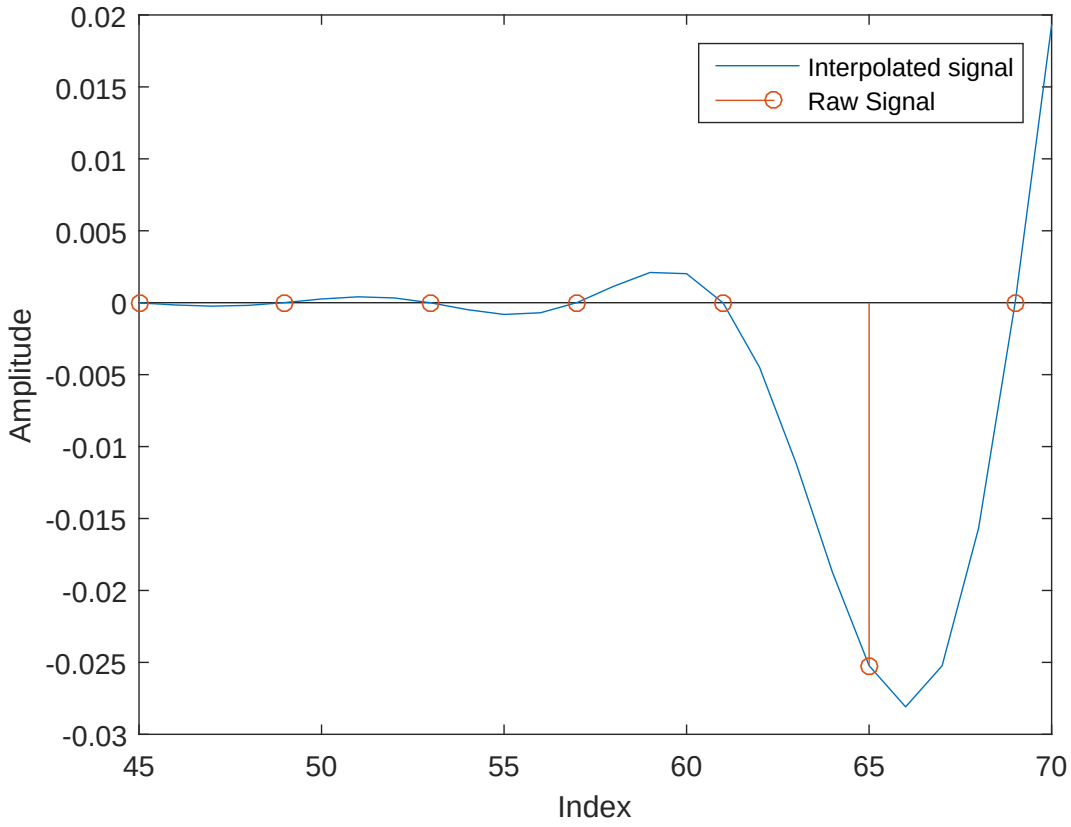


Figure 1.17: FFT Zero Pad Interpolation Gibbs' Phenomenon

1.7 FFT Interpretation of Signal

A digitised ultrasonic signal (shown in Figure 1.18) Consists of the main signal component centred at 4.4 MHz from the FFT. The frequency measured for the signal using time domain analysis (taking the period between peaks) gave a result of 5.5 MHz. The signal also has random near white noise & a signal frequency element also visible in the FFT (Figure 1.19. at 20 MHz.

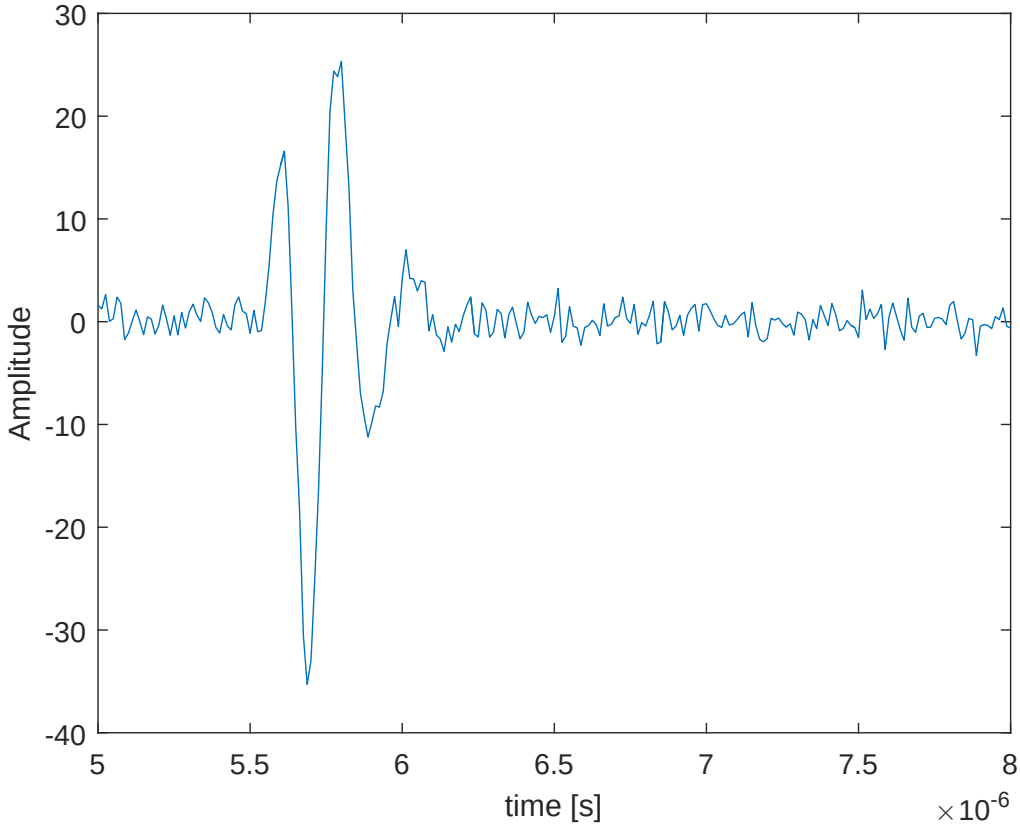


Figure 1.18: Ultrasonic Signal in Time Domain

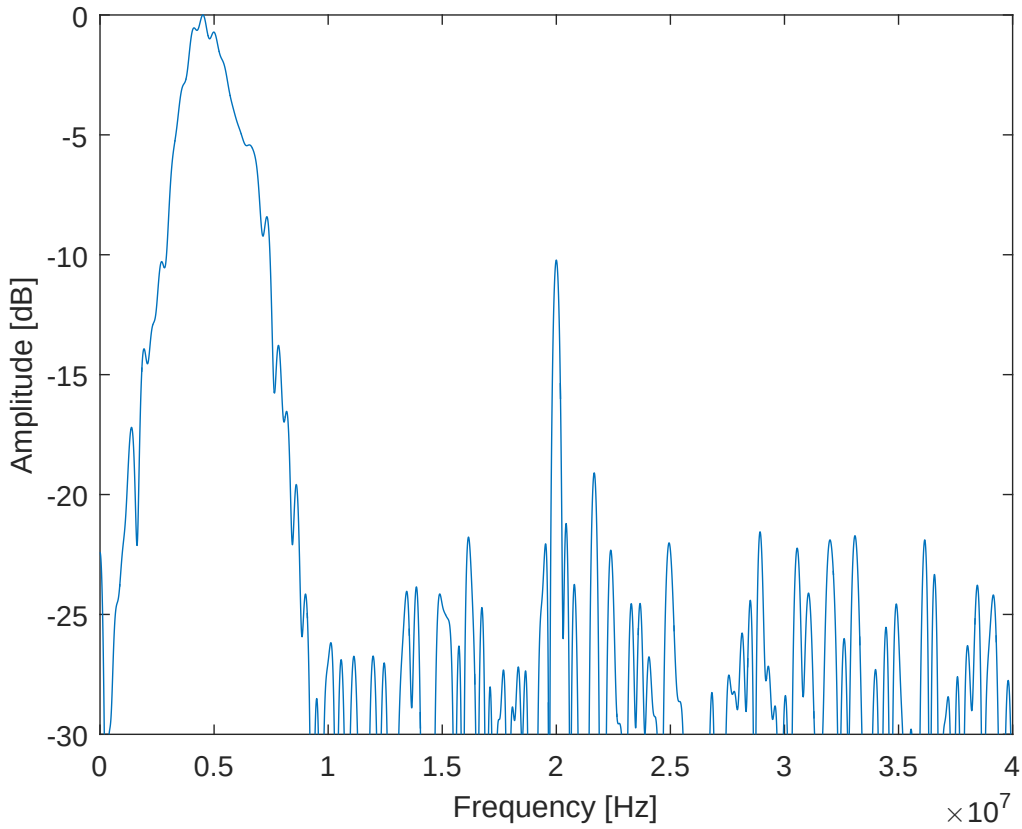


Figure 1.19: FFT of Ultrasonic Signal

1.8 Implementation of a single pole filter

For this exercise a simple single pole filter was implemented. Of the form shown in equation 1

$$H(k) = \frac{k_o}{k_o + jk} \tag{1}$$

$$k = \frac{f}{fs}N \tag{2}$$

$$k_o = \frac{f_o}{fs}N \tag{3}$$

These were implemented into MATLAB with a k_o value of $0.2 * fs$ The resulting frequency response is that shown in Figure 1.20. The original signal and filtered signal are shown in Figure 1.21

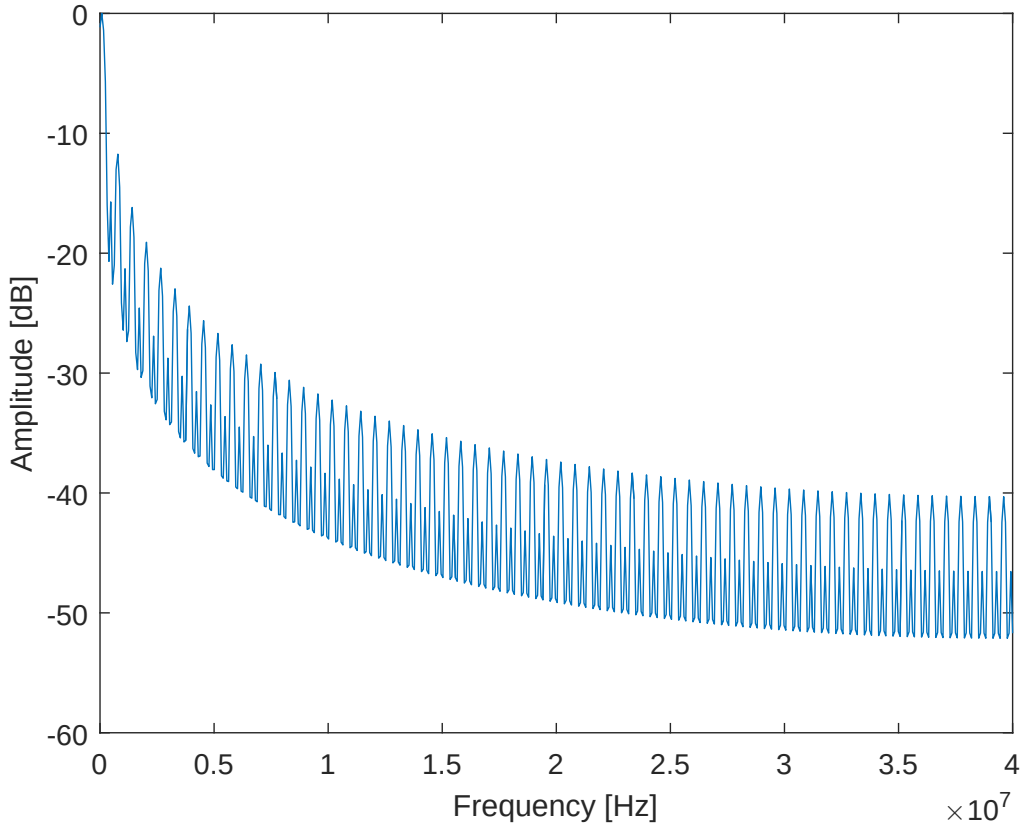


Figure 1.20: Frequency Response of Filter

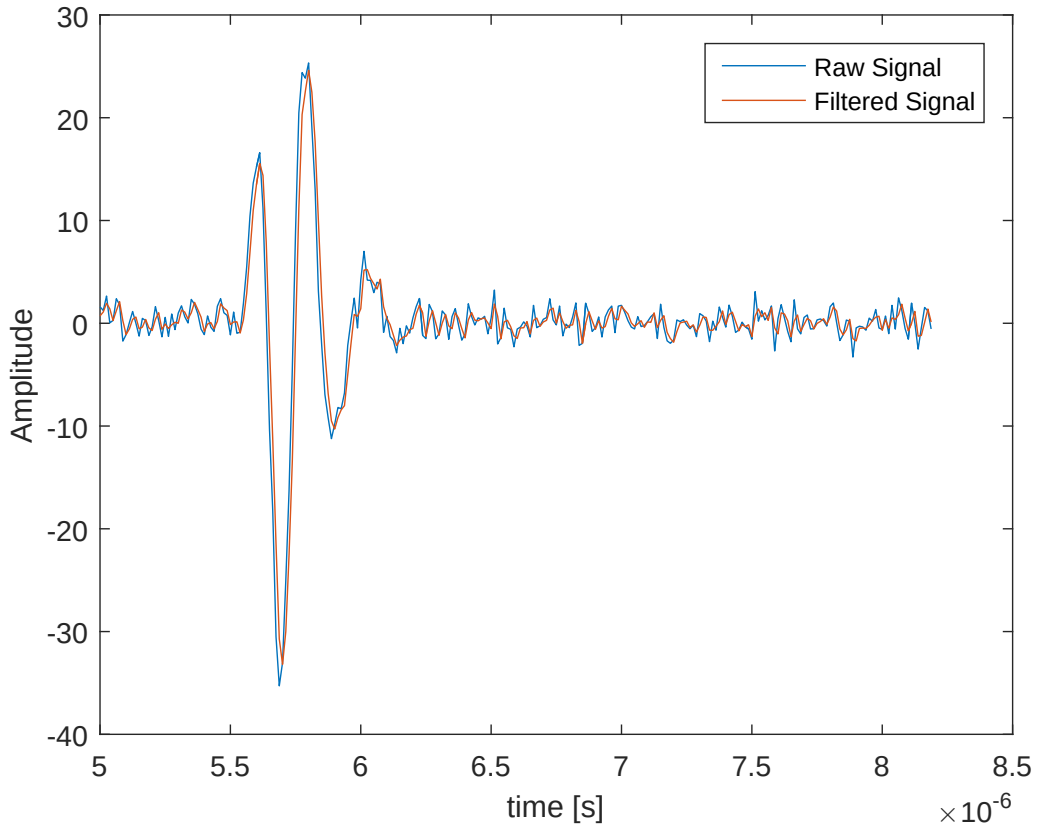


Figure 1.21: Filtered Signal

1.9 Convolution in the Time Domain

In this exercise a simple filter was created & applied in the time domain using convolution. the filter used is that shown in equation 4. The resulting signals are shown in Figure 1.22. The frequency response is shown in Figure 1.23. It is visible the filter has removed some high frequencies however it is clear from the frequency analysis of the signal that it is not the most effective filter with relatively small dB change.

$$h(n) = e^{-n2\pi f_o/f_s} \tag{4}$$

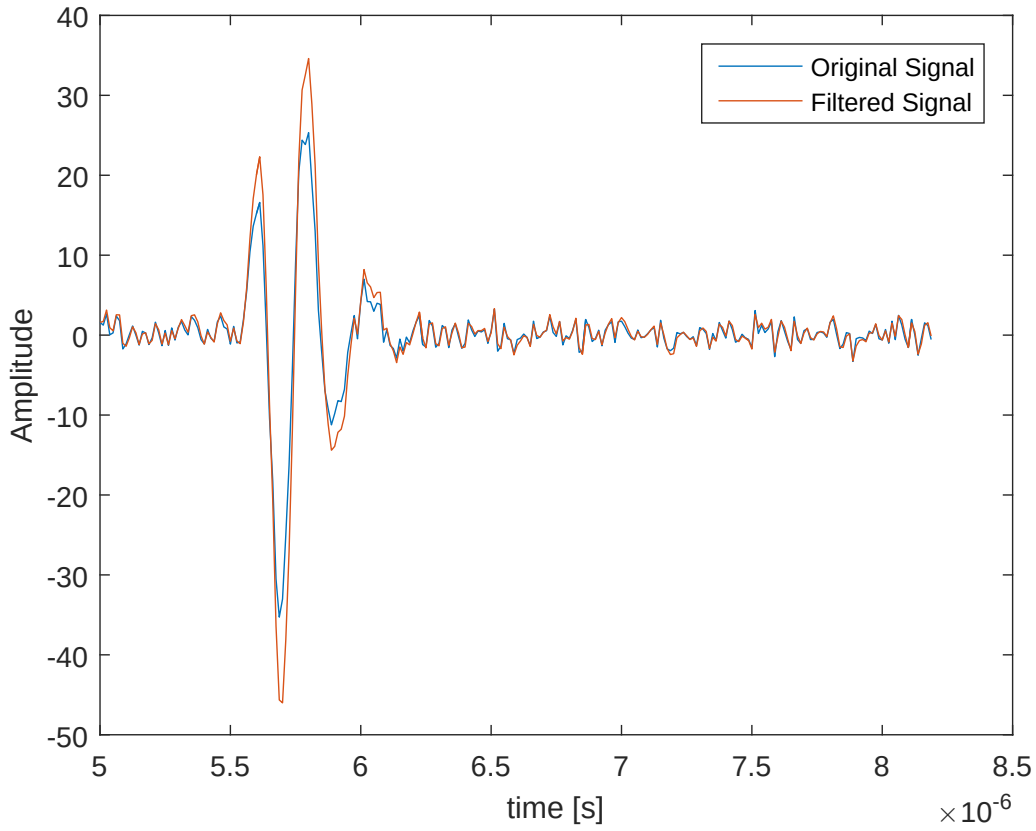


Figure 1.22: Original Signal & Signal Convolved with Filter

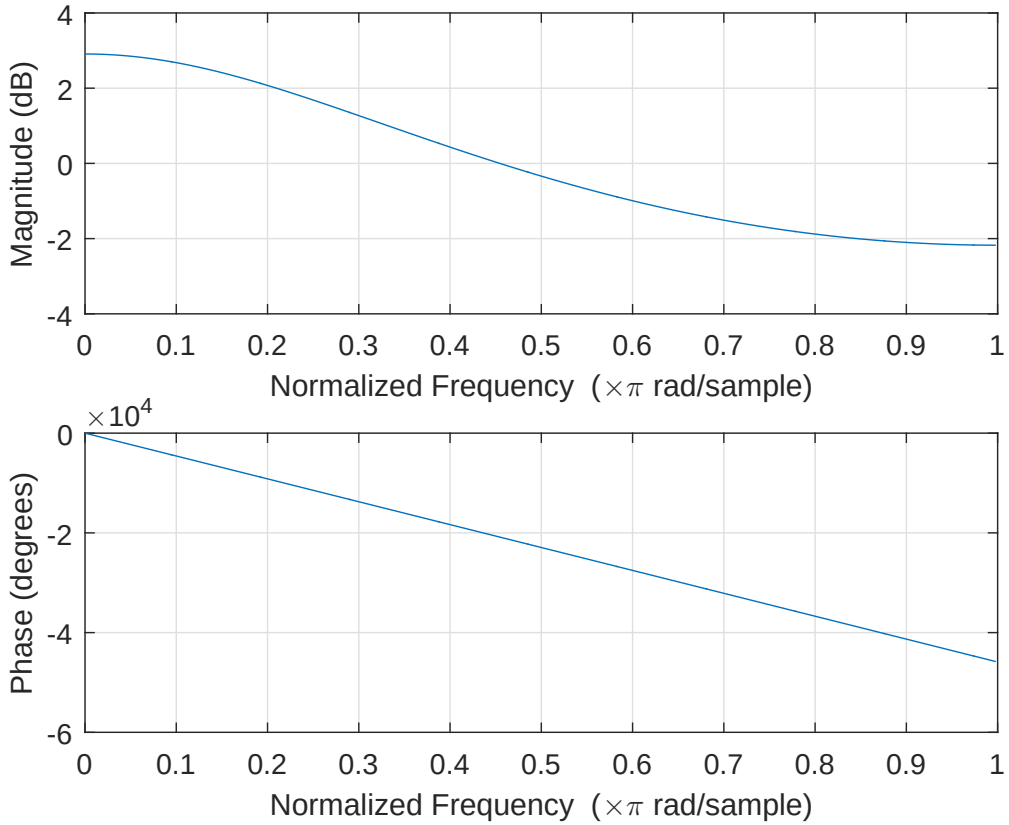


Figure 1.23: Frequency Response of Filter

1.10 Digital Filter Implementation

A digital filter of form shown in 5 was implemented in MATLAB in two ways, firstly by using the inbuilt function and the b & a parameters derived from the equation. Secondly by apply the tap delay line method within a for loop, both shown in the following code snippet 3. The result can be seen in Figure 1.25, frequency response in Figure 1.24 & FFT of filtered and unfiltered signal which shows some removal of interfering frequency but mainly attenuation towards the sampling frequency as expected from the frequency response.

$$H(z) = \frac{z+1}{z(1+\frac{2CR}{T}) - (\frac{2CR}{T} - 1)} \tag{5}$$

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

$Y(z) = X(z) + X(z)z^{-1} + \frac{2CR}{T}Y(z)^{-1}Y(z)^{-1}$

$y[n] = x[n] + x[n-1] + \frac{2CR}{T}y[n-1] - y[n-1]$

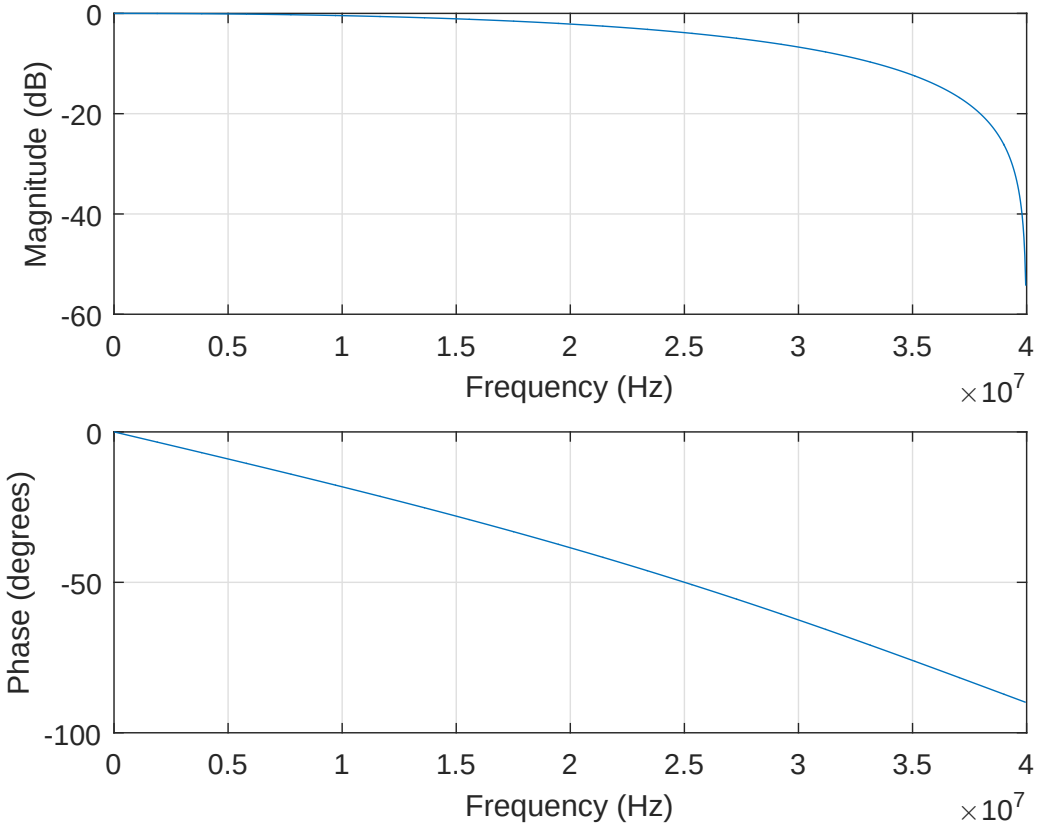


Figure 1.24: Frequency Response of Filter

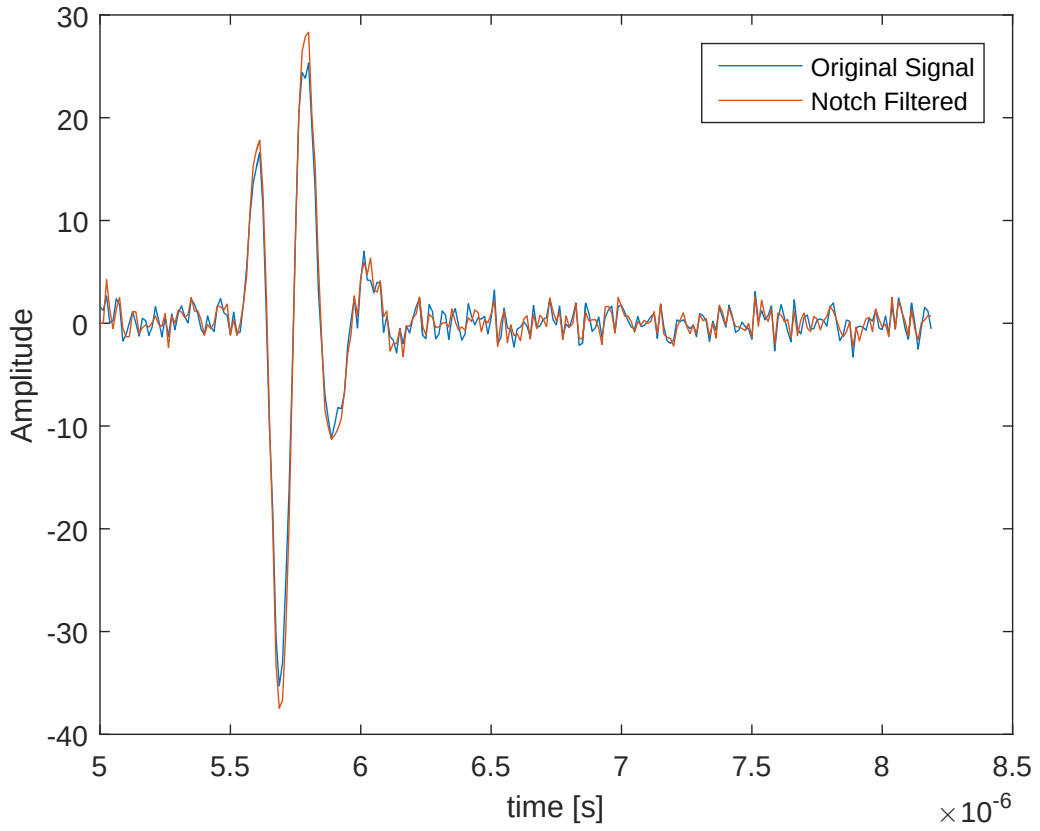


Figure 1.25: Original and Filtered Signal

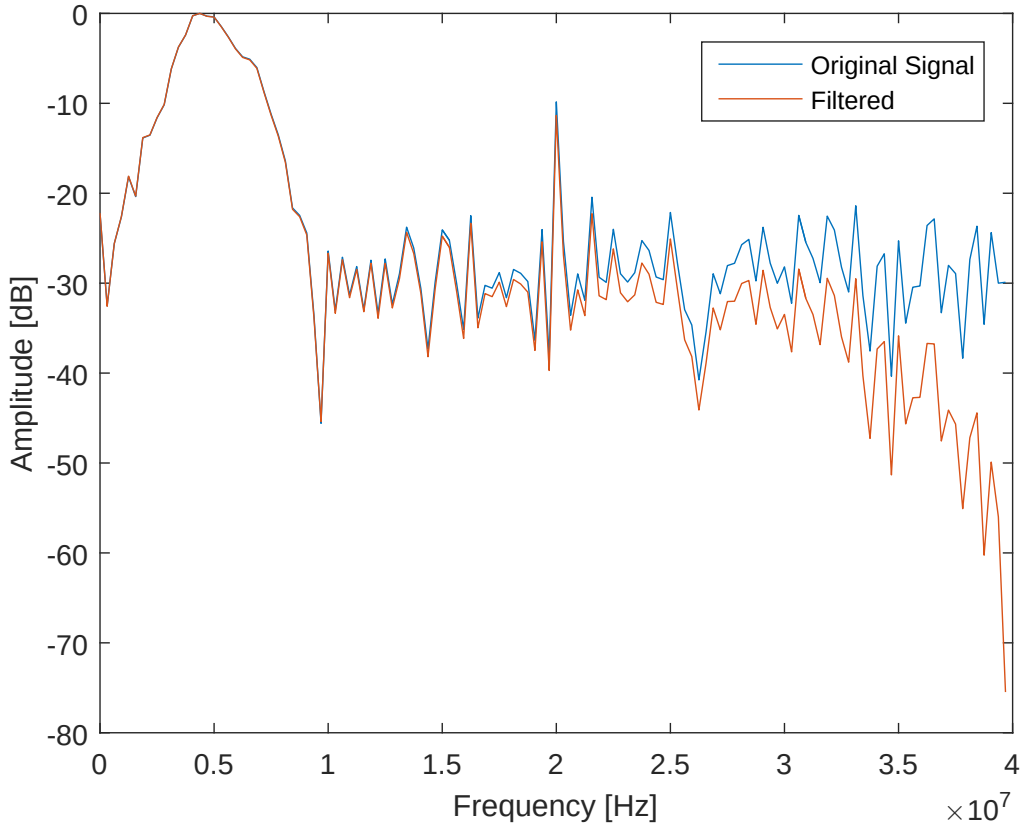


Figure 1.26: Original and Filtered Signal FFT

Listing 3: Filter Implementation

```
m = (2:length(x)); %start from two to remain within signal index
b(1) = 1; %coefficients of filter
b(2) = 1;
a(1) = (1+c);
a(2) = (-c+1);
m_filt = filter(b,a,x); %apply filter
for mm = m(1:end)
y(mm) = x(mm) + x(mm-1) + y(mm-1)*(c-1);
end
```

1.11 Moving Average Filter

A moving average filter was implemented using the equation 6 where N is the number of elements in the averaging window. This was applied to an input signal with results shown in Figure 1.27.

$$y[n] = y[n-1] + x[n] - x[n-N] \tag{6}$$

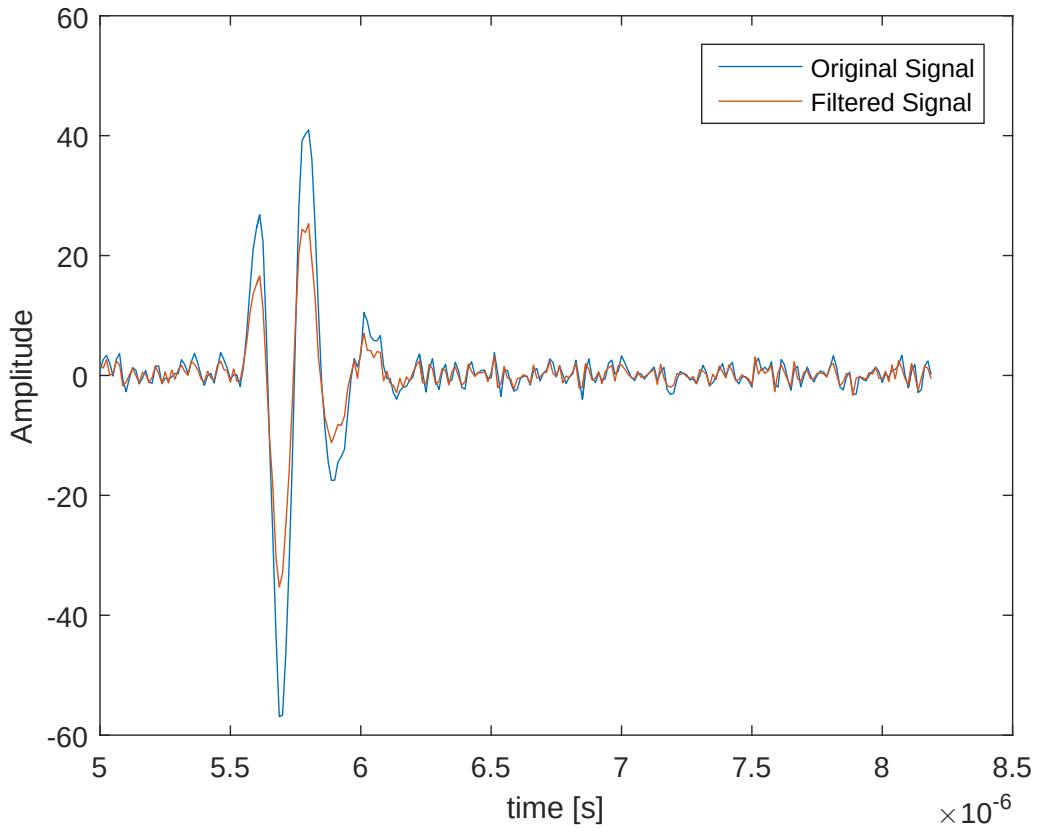


Figure 1.27: Original and Filtered Signal

1.12 Band & High Pass Filter Implementation

A simple band and high pass filter were implemented from equations 8 & 7 respectively. The filters were applied and are shown in Figure 1.28

$$y[n] = -y[n-2] + x[n] - x[n-4] \tag{7}$$

$$y[n] = y[n-1] + x[n] - x[n-4] \tag{8}$$

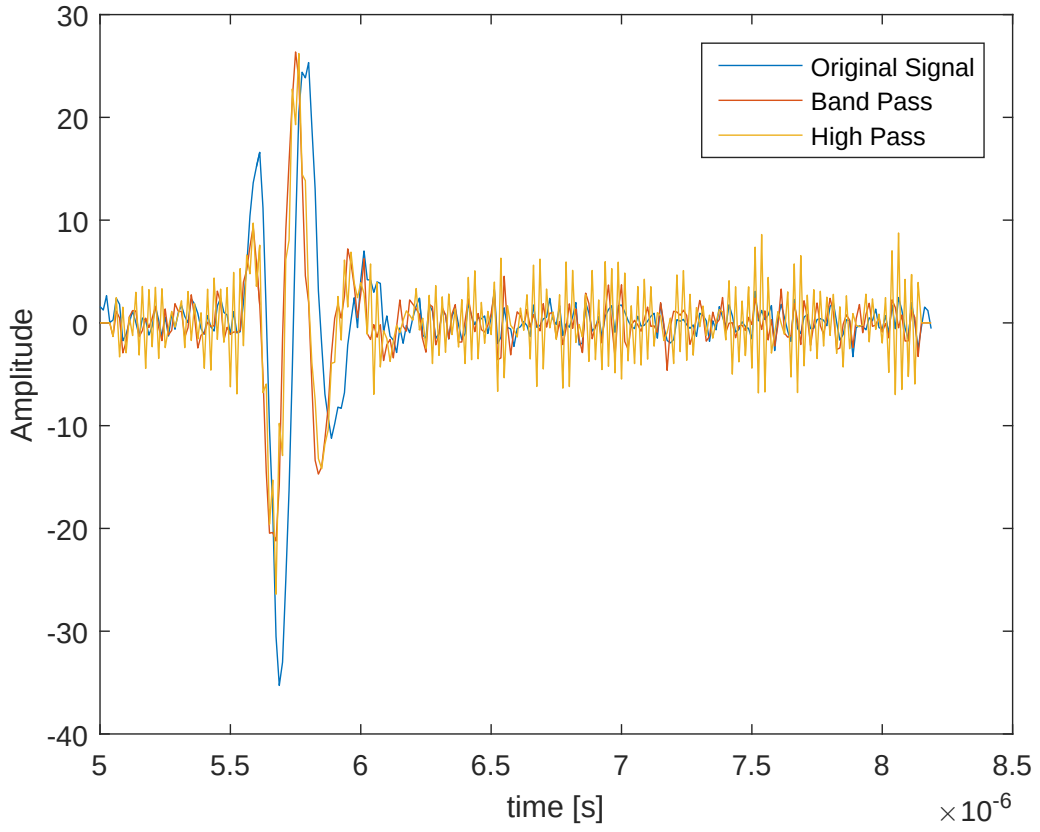


Figure 1.28: Original and Filtered Signals

1.13 Notch Filter

A simple notch filter was designed of the form shown in equation 9. The resulting frequency response of the filter was calculated by convolving the filter with an impulse, the result is shown in Figure 1.29. The filter was then applied to the signal of interest with the resulting output shown in time domain if Figure 1.30 & in the frequency domain in Figure 1.31. The result shows the filter is the most effective used at this point.

$$y[n] = y[n-1] + x[n] - x[n-4] \tag{9}$$

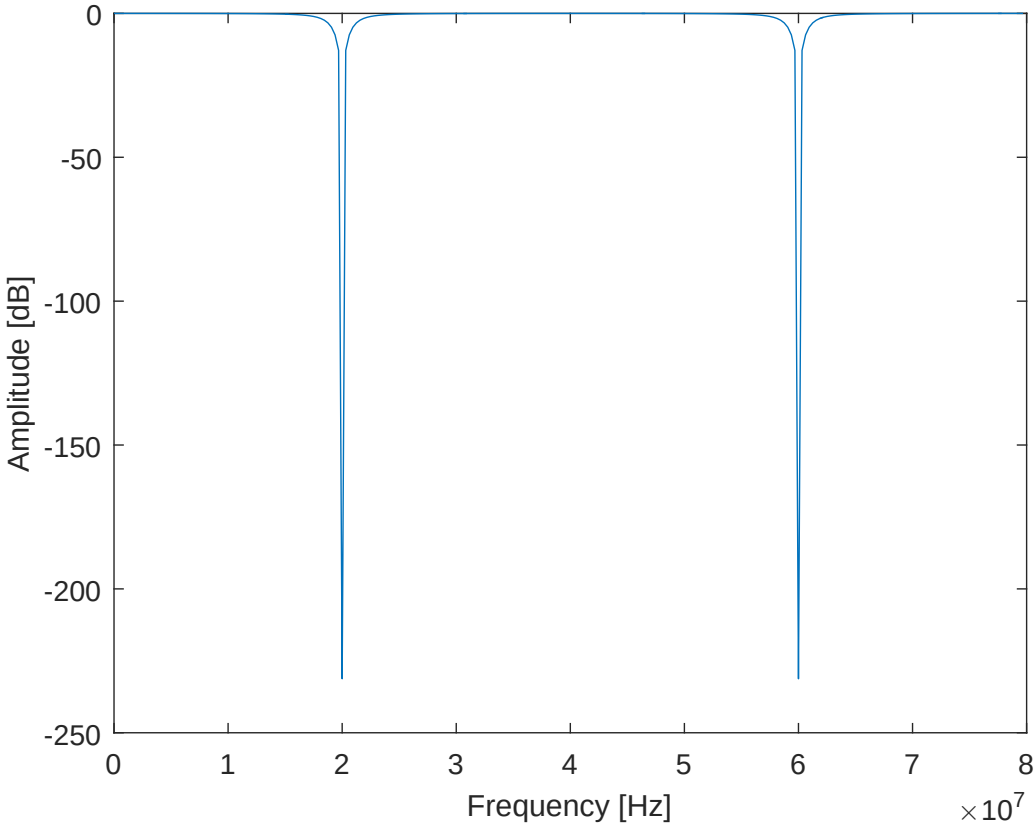


Figure 1.29: Impulse Response of Notch Filter

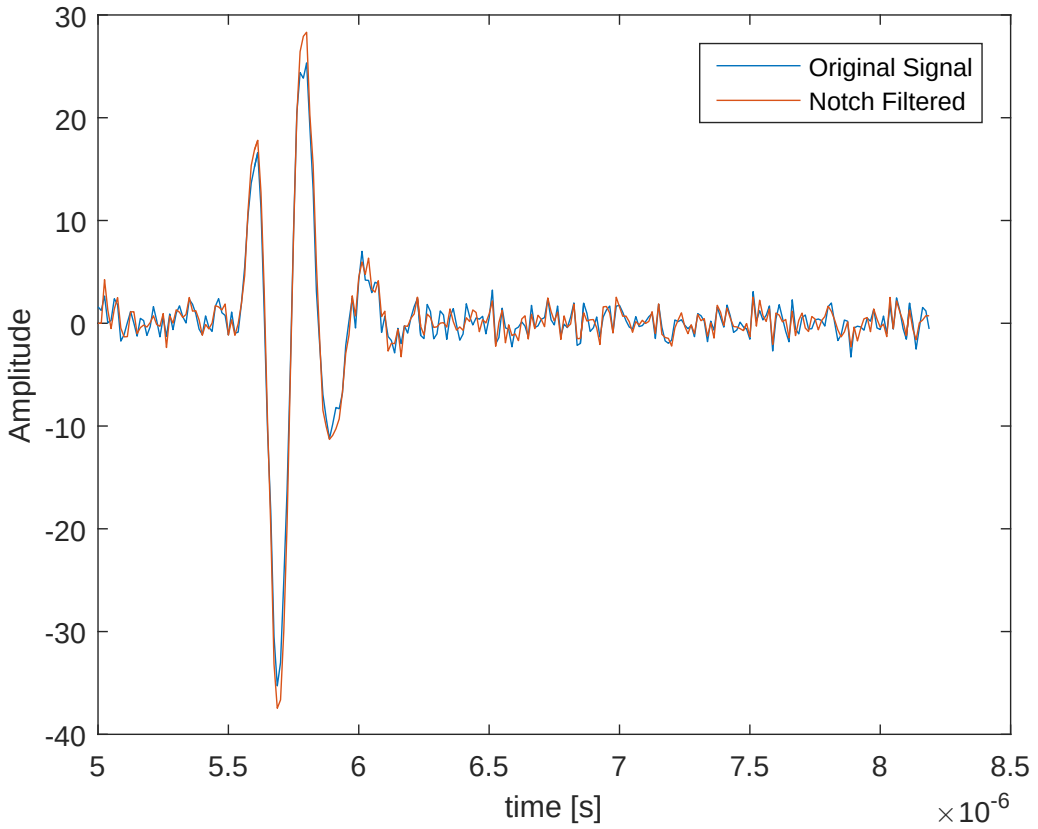


Figure 1.30: Original and Filtered Signals

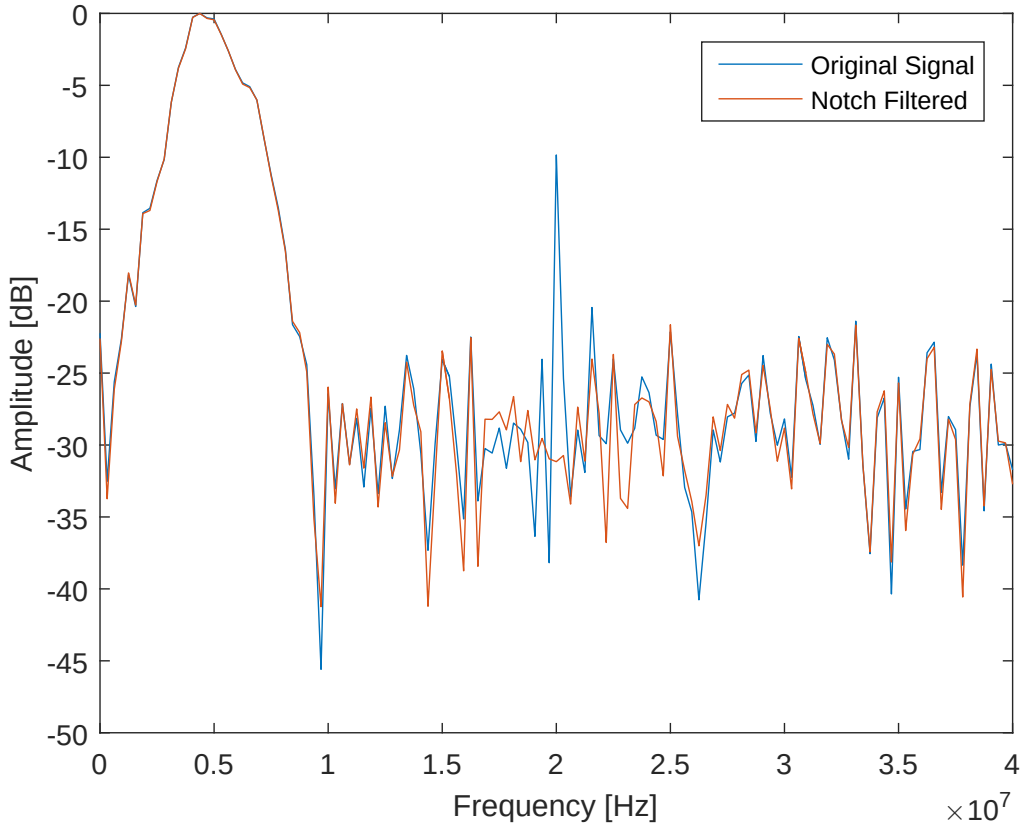


Figure 1.31: Original and Filtered Signals Frequency Domain

2 NDE of an Aerospace Composite Component

2.1 Introduction

A time domain signal of an A-Scan from a 4 mm thick aerospace component is provided as shown in Figure 2.1. The quality of the component can be inferred by the attenuation coefficient.

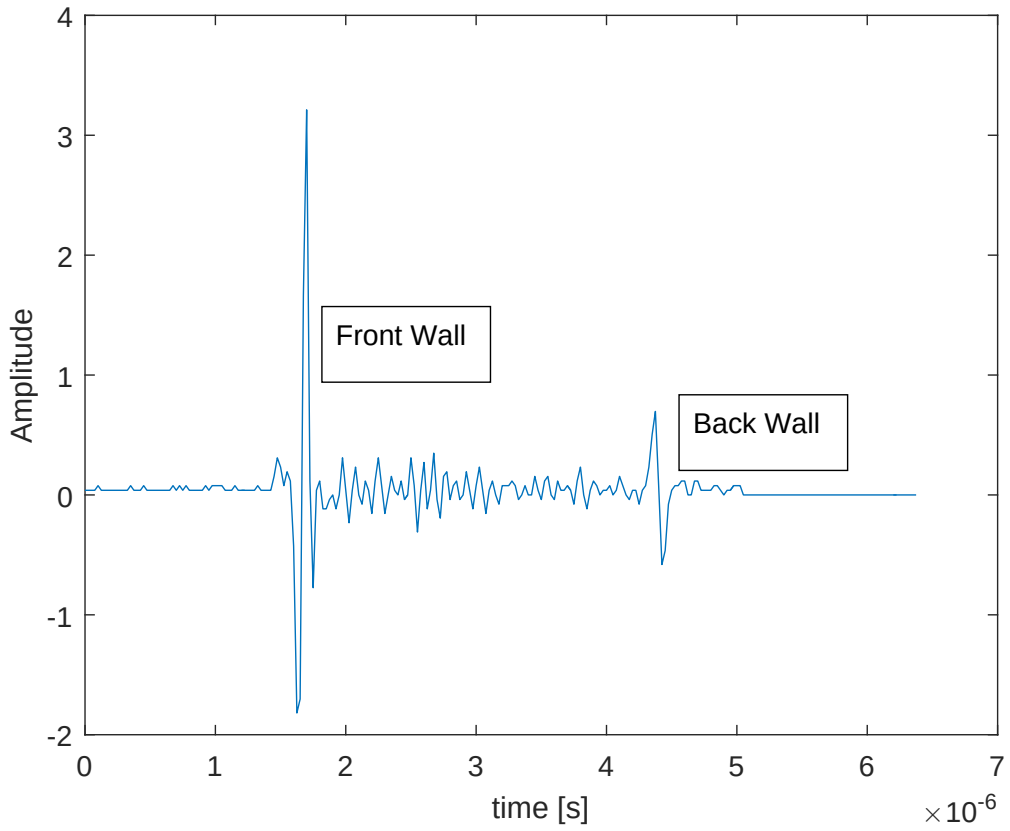


Figure 2.1: A-Scan Line of 4 mm Aerospace Component

2.2 Windowing of The Signal

The sections of the signal to undergo further analysis were windowed using a Tukey (tapered cosine) window [1], this window is desirable as it has a flat top which maintained to amplitude of the front and back wall echoes while minimising the surrounding noise. The parameters of the window are the length of the window and the taper value where 0 is equivalent to that of a rectangular window & 1 is equivalent to a Hann window this is represented in Figure 2.2 The window with taper value of 0.8, as well as the signals for the front & back wall echoes are shown in Figure 2.3.

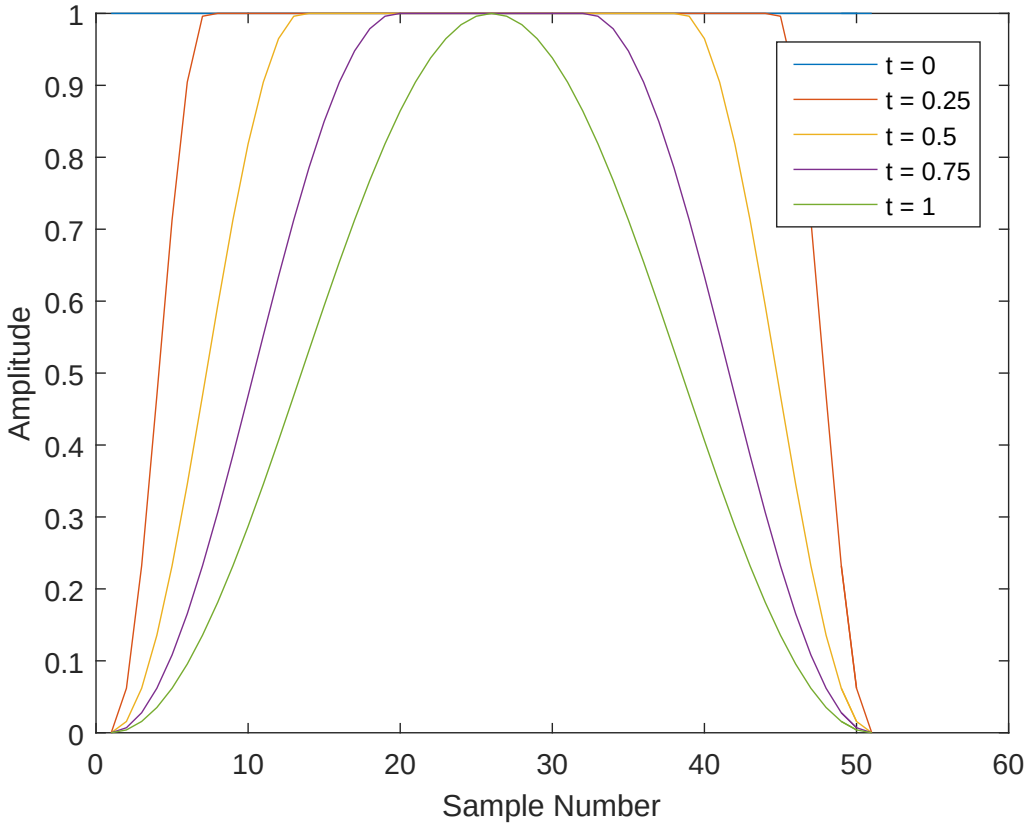
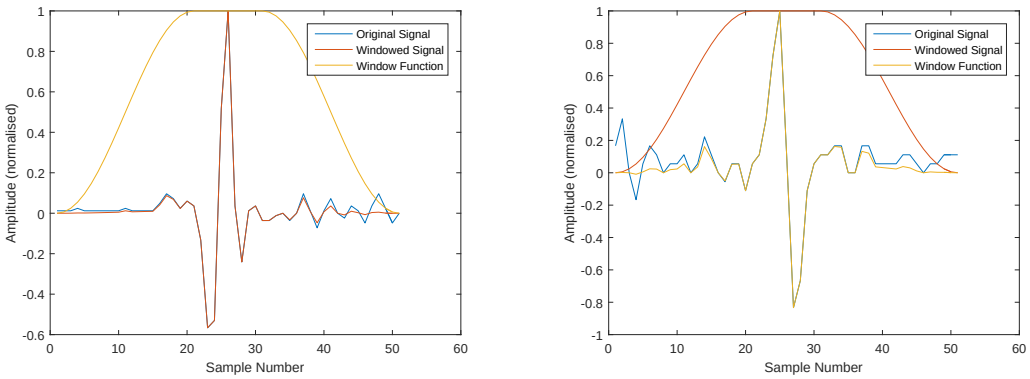


Figure 2.2: Varying Taper Values for Tukey Windows



(a) Front Wall Windowed Signal

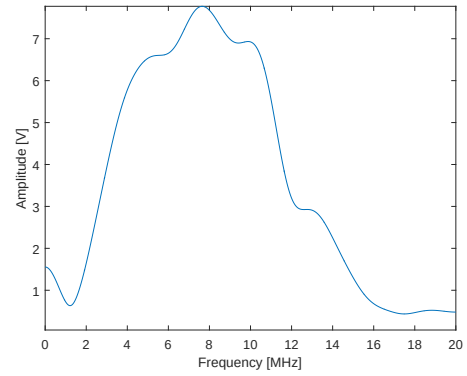
(b) Back Wall Windowed Signal

Figure 2.3: Windowing of Back and Front Wall Reflections

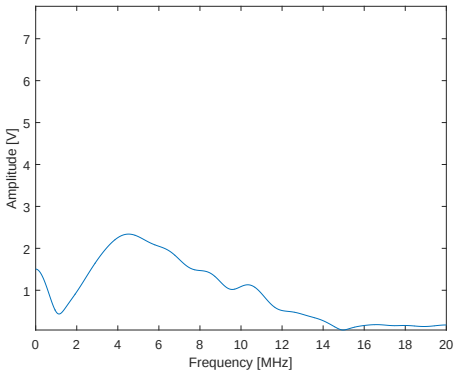
2.3 Discrete Fourier Transform of the Windowed Signals

The signals are to be analysed in the Fourier Domain, to achieve greater resolution between the Discrete Fourier Transform (DFT) Bins a technique known as zero padding will be applied to the signal in the time domain to interpolate in the Fourier Domain [2].

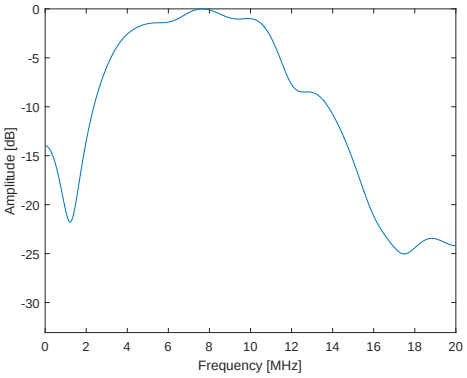
The Fast Fourier Transform (FFT) will be calculated using MATLAB's in built function [3] to arrive at the DFT. The DFT for the frontwall & backwall echos are shown in Figure 2.4 in Voltage and in in dB.



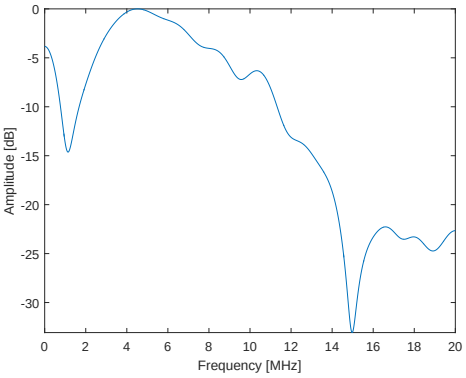
(a) Front Wall DFT Voltage



(b) Back Wall DFT Voltage



(c) Front Wall DFT dB



(d) Back Wall DFT dB

Figure 2.4: DFT of Back and Front Wall Reflections

2.4 Analysis of Attenuation

To calculate the attenuation, the DFT of the front wall echo was simply divided by that of the back wall echo. This was then divided by the thickness to convert the scale to that of a per meter regime and finally converted into a decibel scale. The resultant plot is shown in Figure 2.5. The result can then be compared with a known response of a sample to determine the quality of the component.

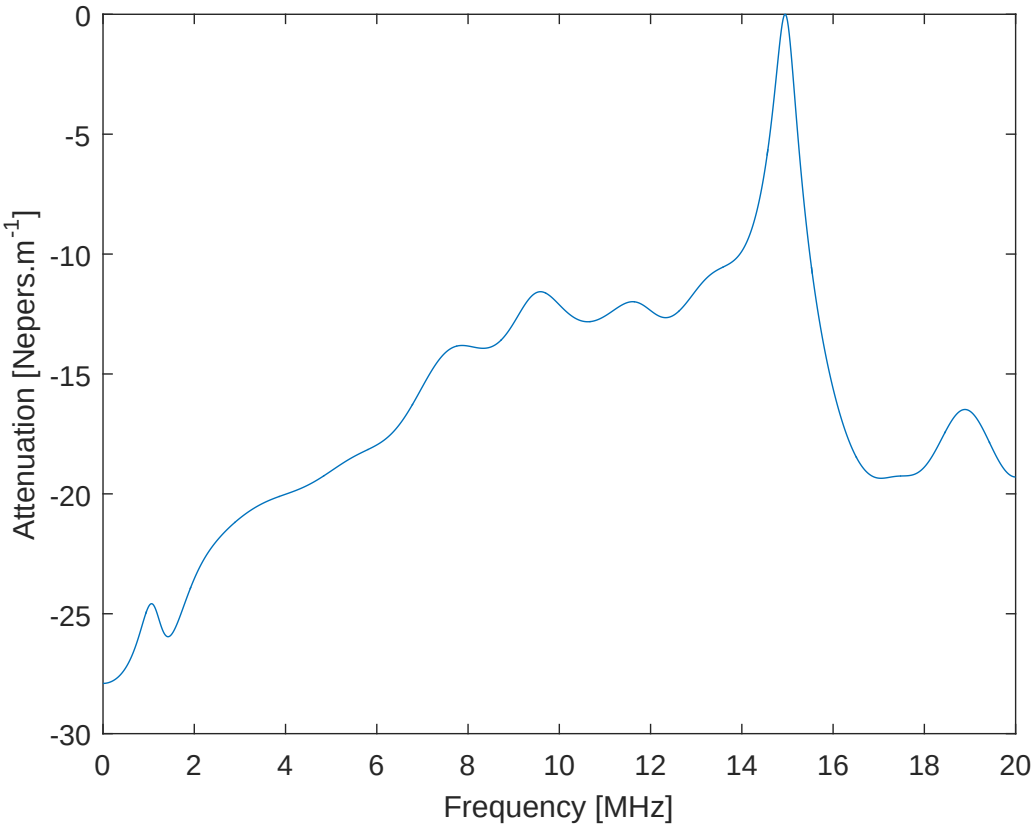


Figure 2.5: Frequency Dependent Attenuation

2.5 Conclusion

This work has detailed the windowing and extraction of front and back wall echoes. The extracted signals have then been transformed to the frequency domain using the FFT algorithm. Within this domain it was possible to calculate the attenuation coefficient for the frequency range of $f = 0 : \frac{fs}{2} Hz$. From this the quality of the component can be evaluated.

3 An Acoustic Emission Experiment

3.1 Introduction

The following work covers the analysis of an acoustic emission experiment which was designed to monitor condition of a complex plant components. The signals provided consist of near white noise shown in Figure 3.1 & the result of this signal when convolved with the plant system shown in Figure 3.2. The standard deviation, autocorrelation, cross correlation & DFT of these signals will be analysed.

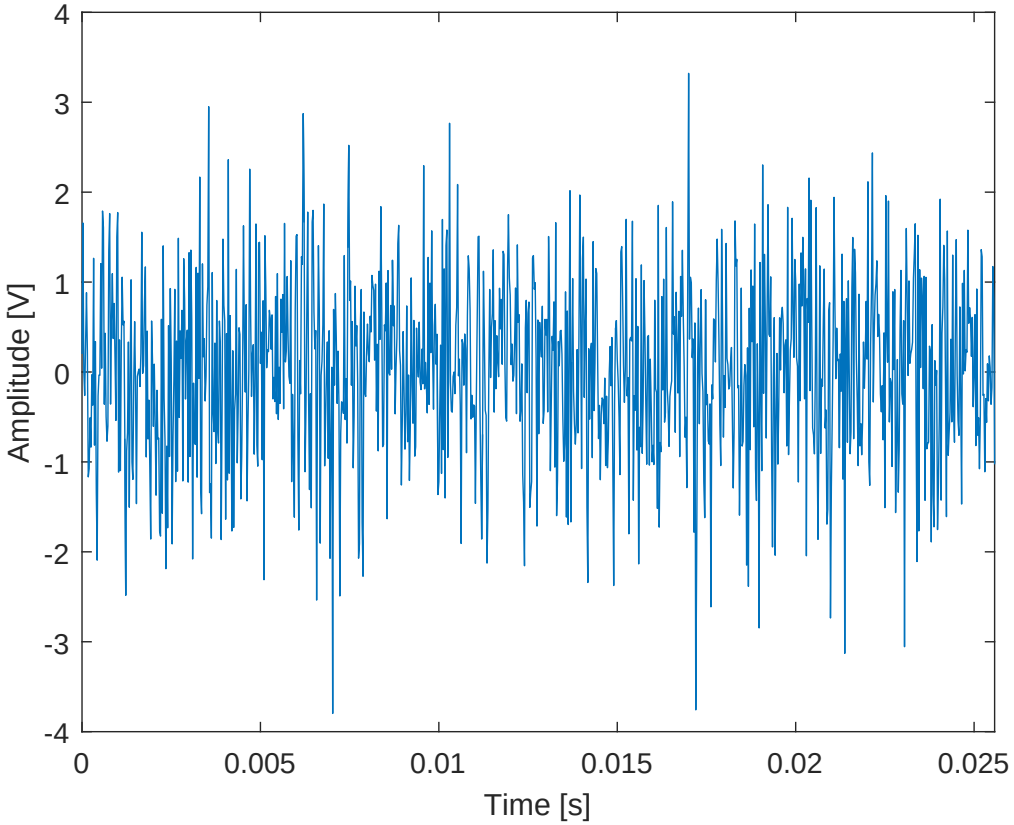


Figure 3.1: Signal Record of X1

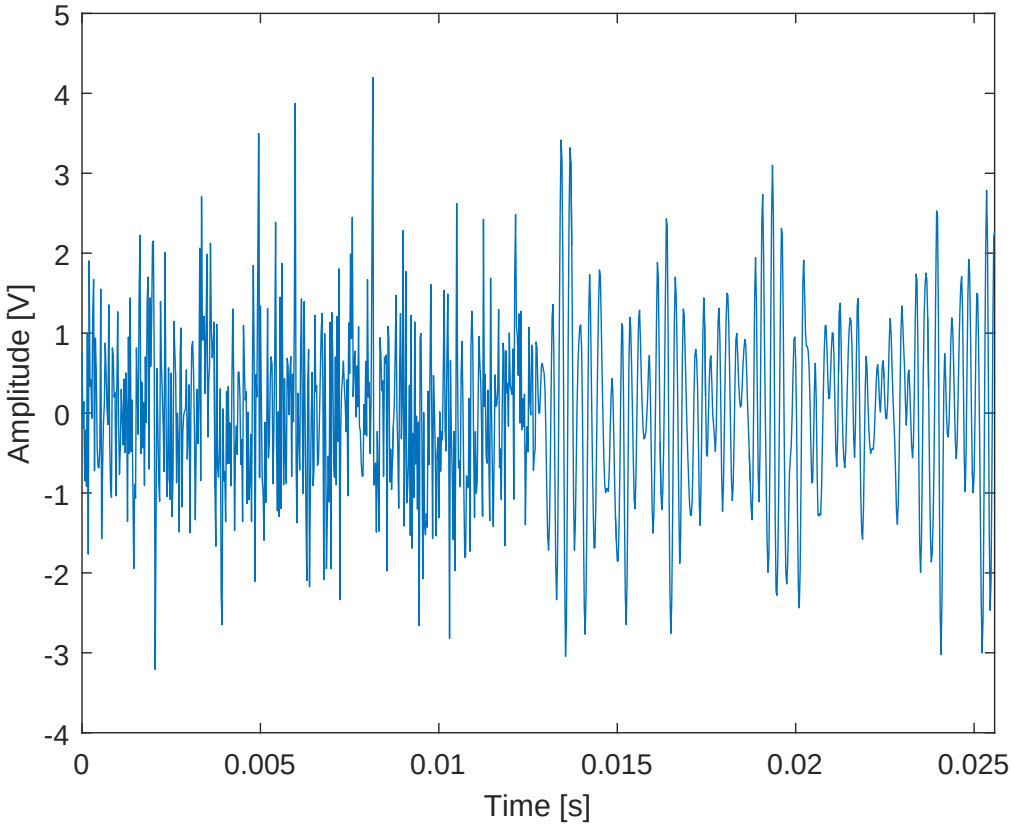


Figure 3.2: Signal Record of X2

3.2 Standard Deviation & Auto Correlation Function

The standard deviation of the signal was calculated by using the MATLAB function [4] for single observation deviation returns a result of 0.9.

The auto correlation function of the signal was calculated using equation 11 of [5]. The resulting plot is shown in Figure 3.3. It is observed the normalised auto correlation value is below 0.2 throughout the system which indicates that the signal has no elements of self-periodicity within itself and is random in nature.

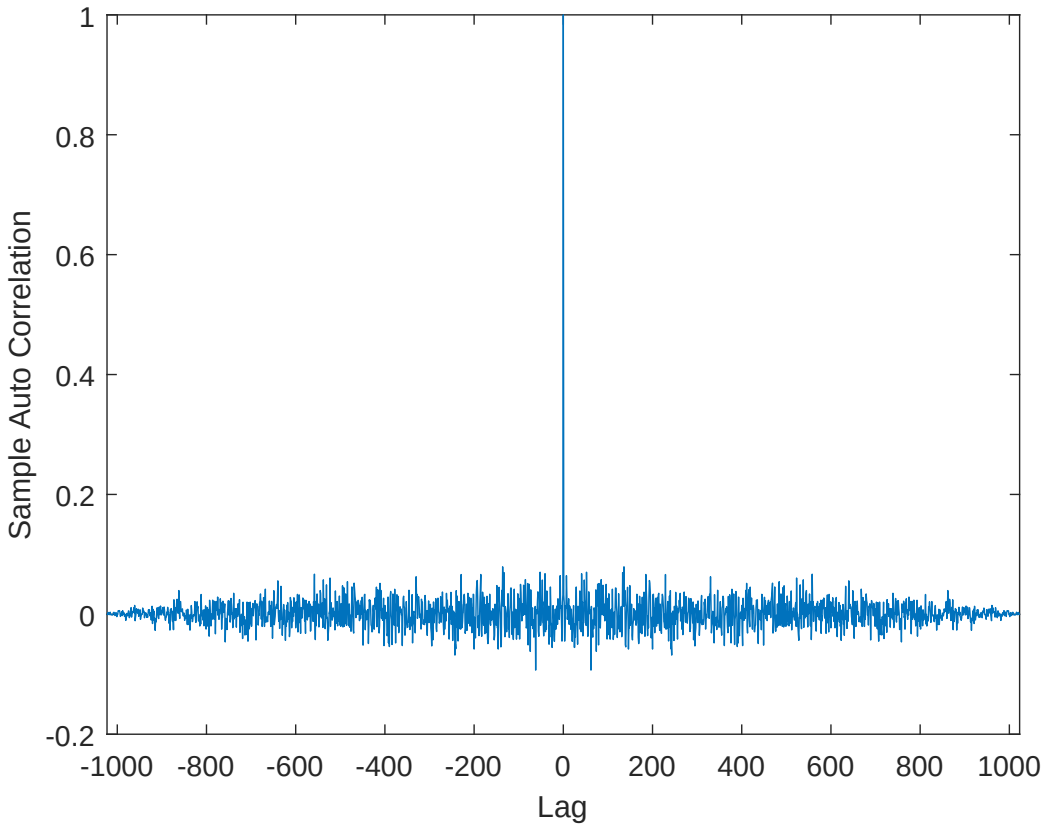


Figure 3.3: Auto Correlation of X1

3.3 Cross Correlation and Frequency Analysis

The signals shown in Figure 3.1 & Figure 3.2 were then cross correlated to determine the impulse response which will lead to the time delay between the first recording position of the signal and the latter as there will be a maximum when the signals are overlapped with the lag value representing this. The resulting correlation is shown in Figure 3.4. The maximum value is observed at a lag of 515 samples, where $Fs = 40kHz$ this correlates to $12.9ms$. The FFT of x1 before it has passed through the system consisting of near white noise will therefore have a frequency response with the amplitude being relative equal across the entire spectrum. This is shown in Figure 3.5. The FFT of the signal after it has been convolved with the system (x2) is shown in Figure 3.6. The FFT of X2 shows a clear introduction of a signal around

$4kHz$ which is not present in the signal prior to convolution with the system. Monitoring of the frequency response & delay could lead to an NDE regime if changes can be meaningfully correlated to known defects, possibly through more simulation of the system.

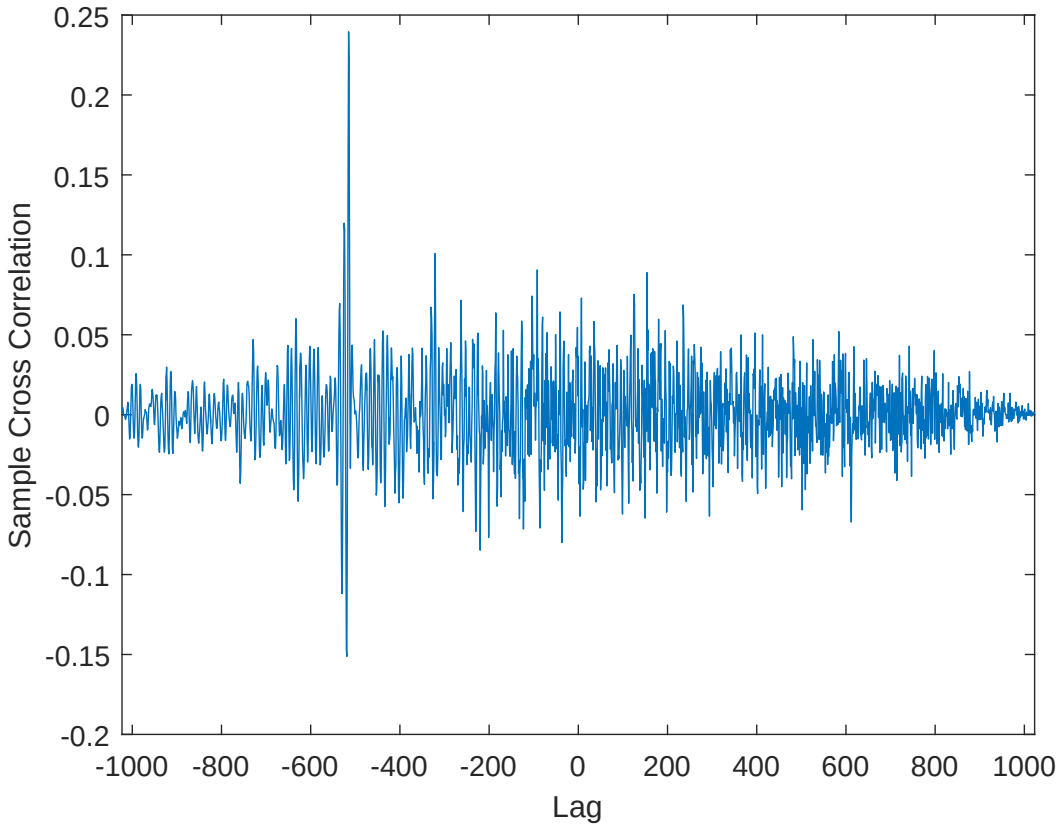


Figure 3.4: Cross Correlation of X1 X2

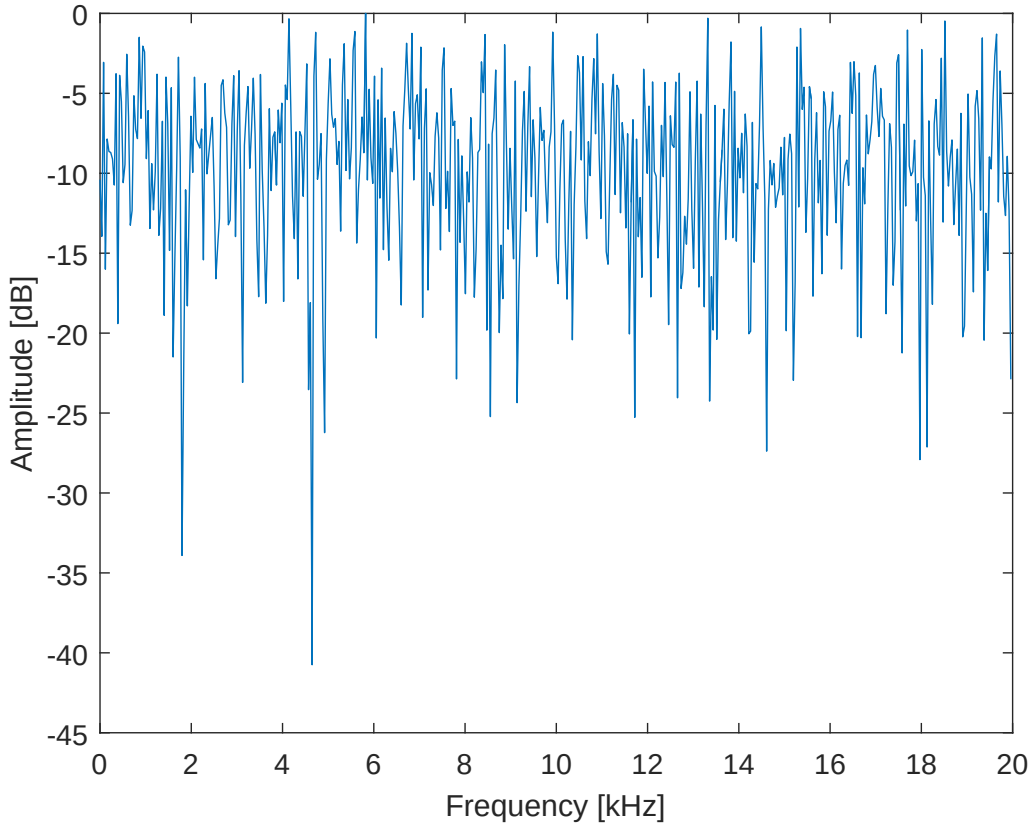


Figure 3.5: FFT of X1

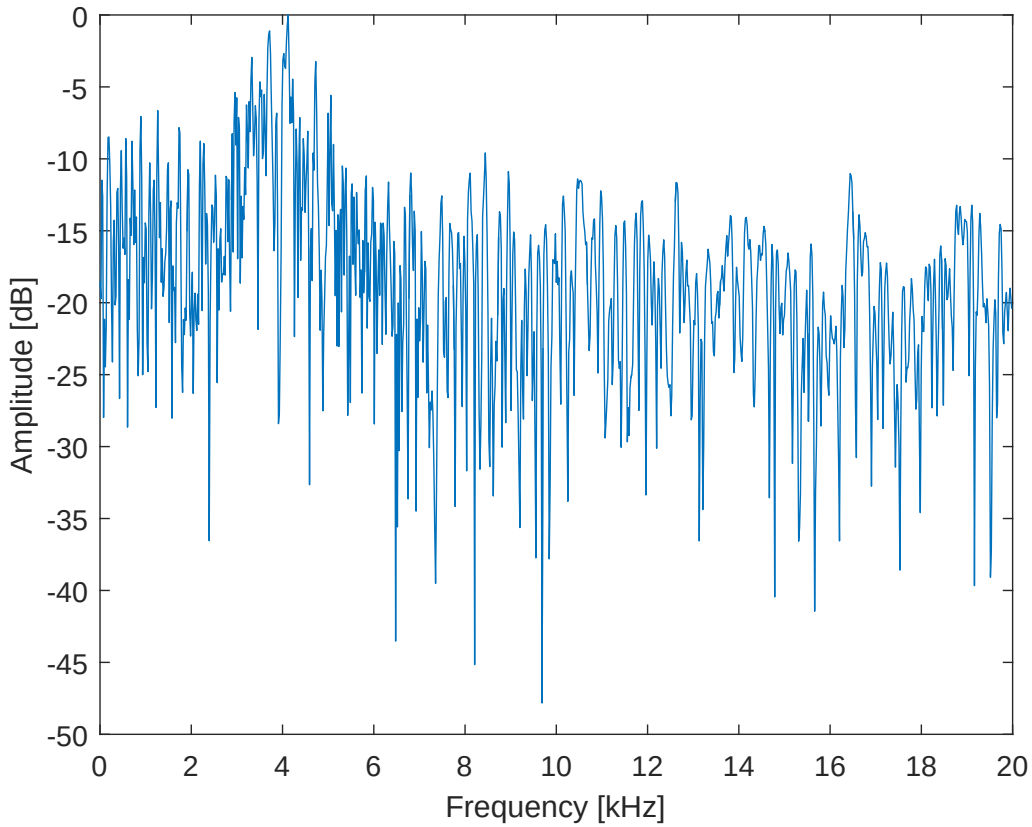


Figure 3.6: FFT of X2

3.4 Conclusion

The analysis of an acoustic emission has been carried out, the input signal was analysed for periodicity to determine if it was near white noise as described. Once this was confirmed the signal was analysed after it had been convolved with a function being the parts within a complex plant. The resulting signal was correlated with the original input signal which resulted in a lag value of $12.9ms$. The signal was then analysed in the frequency domain which led to the conclusion that a sinusoidal component of around $4kHz$ was introduced into the system. The outcome being that the quality of the plant may be assessed by this method if the lag and frequency spectrum can correctly be correlated with known defects possibly with finite element simulations being carried out to determine the response under simulated conditions.

References

[1] "Tukey (tapered cosine) window - MATLAB tukeywin - MathWorks United Kingdom." [Online]. Available: http://uk.mathworks.com/help/signal/ref/tukeywin.html

[2] "Reasons for Zero Padding (Spectral Interpolation)." [Online]. Available: https://ccrma.stanford.edu/~jos/ReviewFourier/Reasons_Zero_Padding_Spectral.html

[3] "Fast Fourier transform - MATLAB fft - MathWorks United Kingdom." [Online]. Available: http://uk.mathworks.com/help/matlab/ref/fft.html

[4] "Standard deviation - MATLAB std - MathWorks United Kingdom." [Online]. Available: http://uk.mathworks.com/help/matlab/ref/std.html

[5] R. Challis and S. Sharples, "Introduction to signal processing for NDE."