­BASA: Mission to Mars

# De-Noising a Received Signal with Periodic Noise Interference

## EGB 242 Signal Analysis

## Liam Andrews – N10528873

# Section A2

## Introduction:

This report contains within it a method and procedure that was undertaken to de-noise a speech signal that was received on the Mars-242 spaceship. Which contained two instances of periodic noise interference within the received signals two halves. Additionally, the MATLAB code created to de-noise the signal is attached below in the appendix.

## Received Signal:

The received signal was originally recorded at a sampling rate of 44100 samples per second over the twenty second duration. When the received signal was played, the speech signal was unintelligible and therefore requiring identification and removal of the interfering noise. The received noise signal can be seen to have two distinct noise patterns that divide the signal into two identifiable parts as in figure 1.

Chart, line chart

Description automatically generated

Figure 1: The Received Speech Signal with Noise

The received signal can be split into two halves which last for ten seconds each. The first half of the signal was identified as being corrupted by the function and the second half was identified as being corrupted by the function .

As all the functions identified by the senior engineer at BASA had an offset of zero further factor were needed to identify which functions were affecting the speech signal. This was done by plotting the identified functions and comparing their period and shape with that of the received signal. The function identified as corrupting the first half of the signal was recognised as due to the triangular shaped nature the signal produced when plotted. Additionally, it was noticed that initial spike in the graph from the first half of the piecewise function closely matched that of the received signal, this can be seen below in figure 2. The second function was identified in a similar method to the first function as the shape of the graph resembled the shaped of the received signal seen in figure 1, also due to the received signals linear shape it ruled out the function as it contained and exponential which would have resulted in the received function curving within the second half. As seen in figure 1 the received signal had a duration of twenty seconds and as both functions had a duration of five seconds both noises would appear twice within each half of the received signal.

The Additive Noise functions are displayed below with the MATLAB code used to generate the graphs.

Chart, line chart

Description automatically generated

Figure 2: Identified Additive Noise Signals

% Period

samples = lgthNS \* 0.25;

T = 5;

% Time Vector

kPeriod = linspace(0, T, samples + 1); kPeriod(end) = [];

% Time Vector in Halves

kPeriodHalf = linspace(0, T/2, samples/2 + 1); kPeriodHalf(end) = [];

kPeriodHalf2 = linspace(T/2, T, samples/2 + 1); kPeriodHalf2(end) = [];

% The Piecewise function s3(t)

PW1 = (A\*kPeriodHalf -1.25\*A); % First half of the piecewise function

PW2 = (-A\*kPeriodHalf2 + 3.75\*A); % Second half of the piecewise function

% Addition of Noise Vectors

addNoise1st = [PW1 PW2]; % Piece Wise Function

addNoise2nd = kPeriod/A; % Upward Sloping Function

The second periodic noise signal has been determined to be an odd function as it satisfies the mathematical definitions shown below.

Which states that a function of x when time is negative is the same as minus one multiplied by x of the same time, for example :

Since this rule is satisfied the function is proven to be odd.

## Analysis of Signal:

Complex Fourier Series was chosen as the method to model the periodic noise signals. It was chosen over Trigonometric Fourier Series as it requires less computation in MATLAB due to the number of lines of code for the approximation to be calculated. As Trigonometric Fourier Series requires two coefficients and a constant, which when compared to Complex Fourier Series which only requires one constant and one coefficient, it means that the chance of coding errors is increased and since the process is performed twice for the two noise signals. Additionally, using exponentials in verification by hand calculations is easier as the exponential dose not change when integrated.

The first additive noise signal was modelled using Complex Fourier Series that was calculated by hand and plotted against the received signal. The Complex Fourier Series model when calculated by hand uses the following rule.

The worked solutions for the above equations can be found in the appendix. The results of using the above equations are:

Therefore, the hand calculated approximation of is:

MATLAB was to evaluate this expression. The following plot is of the previous expression against the first additive noise from the received signal.

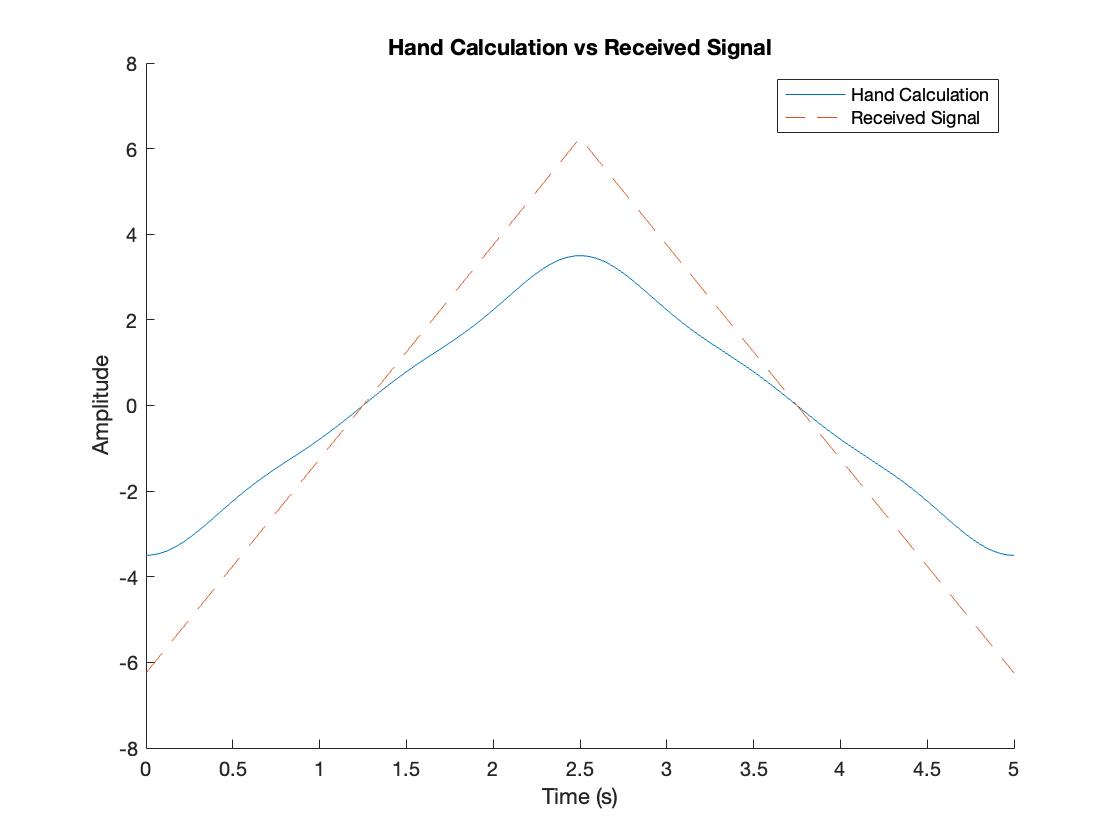


Figure 3: Hand Calculation vs First Additive Noise

% Complex Fourier Series Calculated by Hand

CHc0 = 0;

CHcn = (7.5 .\* (exp(1j .\* -pi .\* compN) - 1) ./ (pi.^2 .\* compN.^2));

% Zeros Vector Required for For Loop

CHapprox = zeros(1, samples);

% For Loop

for n = compN

CHapprox = CHapprox + (7.5 .\* (exp(1j .\* pi .\* n) - 1) ./ (pi.^2 .\* n.^2))...

.\* exp(1j .\* 2 .\* pi .\* n .\* f0 .\* kPeriod);

end

% Add the value of n = 0

CHapprox = CHapprox + CHc0;

The Second Additive noise was approximated using MATLAB and the plotted against the received Signal. This approximation also used the same number of harmonics as what was stated in the hand calculations. The following is a plot of the computer approximation against the received second additive noise signal.

Chart, line chart

Description automatically generated

Figure 4: Computer Complex Fourier Series Approximation vs Received Signal

% Complete Complex Fourier Series

% Number of Harmonics

Harm = 5;

compN = -Harm:Harm;

compN(Harm + 1) = []; % Used to remove n = 0 as c0 is present

% Time step

Ts = kPeriod(2) - kPeriod(1);

% Fundamental Frequency

f0 = 1/T;

% Complex Fourier Series

c0 = 1/T \* sum(addNoise2nd) \* Ts;

cn = 1/T \* addNoise2nd \* exp(1j \* -2 \* pi \* f0 \* compN' \* kPeriod).' \* Ts;

FS2nd = c0 + cn \* exp(1j \* 2 \* pi \* f0 \* compN' \* kPeriod);

## De-Noising the Speech Signal:

The first step in de-noising the original speech signal required the identification of the noise signals that were interfering with the speech. ­Once these were identified they had to be modelled over a single period of the received signal. From this Complex Fourier Series Approximations were created using MATLAB for each of the identified noise functions. Before they were combined to be subtracted away from the received signal to produce the original speech signal without noise.

% Fourier Series Approximations of the Additional Noise

% Additional Noise Vectors

Noise1st = [PW1 PW2]; % Piece Wise Function S3(t)

Noise2nd = kPeriod/A; % Upward Sloping Function S1(t)

% Complex Fourier Series of the 1st Noise

FS1c0 = 1/T \* sum(addNoise1st) \* Ts;

FS1cn = 1/T \* addNoise1st \* exp(1j \* -2 \* pi \* f0 \* compN' \* kPeriod).' \* Ts;

FS1st = FS1c0 + FS1cn \* exp(1j \* 2 \* pi \* f0 \* compN' \* kPeriod);

% Complex Fourier Series of the 2nd Noise

FS2c0 = 1/T \* sum(Noise2nd) \* Ts;

FS2cn = 1/T \* Noise2nd \* exp(1j \* -2 \* pi \* f0 \* compN' \* kPeriod).' \* Ts;

FS2nd = FS2c0 + FS2cn \* exp(1j \* 2 \* pi \* f0 \* compN' \* kPeriod);

Chart, line chart

Description automatically generated

Figure 5: Complex Fourier Series Approximations of the Interfering Noise

The above figure displays the two different noise patterns that were originally identified to be interfering with the speech signal after they have been modelled using a Complex Fourier Series Approximation with the number of harmonics displayed in the graph’s legend. The following graph is a plot of the combined Fourier Series Approximation of the two additive noises when plotted with the same duration as the original signal.

Chart, line chart

Description automatically generated

Figure 6: Combined Fourier Series Approximations

During the process of Complex Fourier Series Approximation, the number of harmonics used in the approximation was determined to only affect the computational time by increase the codes run time as harmonics ranging from , to , and to . As any changes in the quality and clarity of the speech were not heard when it was subsequently played at the differing range of harmonics. Despite this with greater harmonics the accuracy of the signal does improve as the step changes are smaller and closer together.

After de-noising the process had occurred the speech signal looked as follows.

Chart

Description automatically generated

Figure 7: The De-Noised Speech Signal

As seen in the above plot the speech signal is now undistorted by noise and takes on the form expected of a speech signal. Now that the speech signal is free of noise the message can be heard. A transcript of the message is entered below.

***“T-minus 10, 9, 8, 7, 6, 5, all three engines up and burning, 2, 1, 0, and lift off. The final lift off, of Atlantis, on the shoulders of the space shuttle, America will continue to Dream.”***

# Section A3

## Reflection:

During this assessment I had two opportunities to display my knowledge and understanding of Fourier Series Approximations and integration. First with MATLAB Grader which required lengthy hand calculations involving multiple instances of integration by parts. In both parts the coding skills developed within the tutorials was on display, with methods such a ‘for loop’ which can be easily used to compute the summations within the either Complex Fourier Series or Trigonometric Fourier Series Approximation when using hand calculations. Furthermore, within the Section A2 I was able to correctly identify the additive noise signals by writing code to plot and perform Complex Fourier Series. From which the Complex Fourier Series Approximations were used to de-noise the signal and recover the original speech and transcribe it.

In Section A1 of the assignment, it was brought to my attention that there gaps in my knowledge and understanding as I was unable to complete every objective of the first three questions on MATLAB Grader. Despite this I was able to achieve the identification of the interfering signals, before completing Complex Fourier Series Approximation as when undertaking MATLAB Grader, the code written for calculating c0 and cn were marked as correct which meant correcting the approximation line of code. I was able to make the required correction after reconsulting the provided tutorial content. Additionally, I did not effectively plan out how this assignment was going to be undertaken and in what order. Which meant wasted time during the initial weeks and a crunch period in the final weeks.

# Appendix:

## MATLAB Grader Question 1:

### Trigonometric Fourier Series Calculations:

Text, letter

Description automatically generated

Text, letter

Description automatically generated

# Text, letter Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

## MATLAB Grader Question 2:

### Complex Fourier Series Calculations:

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

## Mission Question 5:

### Complex Fourier Series Calculations:

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

Diagram

Description automatically generated

Text, letter

Description automatically generated

Text, letter

Description automatically generated

## MATLAB Grader Code:

%% Matlab Grader Question 1

function [a0, an, bn, s1t\_approx, sid] = trigFS(sid)

syms t, syms n, pi = sym('pi');

% Enter your student ID as sid:

sid = 10528873;

% As a symbolic expression, save the expressions

% for the trigonometric coefficients below:

a0 = 3/2

an = (3\*sin((3\*pi\*n)/2)/(pi\*n)) - (3\*sin((pi\*n)/2)/(pi\*n))

bn = (-2\*sin((pi\*n)/2)/(pi^2 \* n^2)) + ((3/(pi\*n))\*(cos((pi\*n)/2) - cos((3\*pi\*n)/2)))

% Find the trigonometric FS, evaluate the FS with

% 2 harmonics and save the expression below:

f = 1;

nn = 2;

a1 = (3\*sin((3\*pi\*n)/2)/(pi\*n)) - (3\*sin((pi\*n)/2)/(pi\*n));

b1 = (-2\*sin((pi\*n)/2)/(pi^2 \* n^2)) + ((3/(pi\*n))\*(cos((pi\*n)/2) - cos((3\*pi\*n)/2)));

a2 = (3\*sin((3\*pi\*nn)/2)/(pi\*nn)) - (3\*sin((pi\*nn)/2)/(pi\*nn));

b2 = (-2\*sin((pi\*nn)/2)/(pi^2 \* nn^2)) + ((3/(pi\*nn))\*(cos((pi\*nn)/2) - cos((3\*pi\*nn)/2)));

s1t\_approx = a0 + a1 + b1 + a2 + b2

end

%% Matlab Grader Question 2

function [c0, cn, s2t\_approx, sid] = compFS(sid)

syms n t; e = sym(exp(sym(1))); pi = sym('pi');

% Enter your student ID as sid:

sid = 10528873;

% As a symbolic expression, save the expressions

% for the complex coefficients below:

c0 = ((exp(3) - exp(-3/2 + 3))/3 - 1);

cn = (((exp(3) - exp(3/2 + 1j\*pi\*n))/(3 - 1j\*2\*pi\*n)) + ((1/(1j\*pi\*n)) \* (exp(-1j\*pi\*n) - 1)));

% Find the complex FS, evaluate the FS up to

% second harmonic and save the expression below:

s2t\_approx = (((exp(3) - exp(3/2 + 1j\*pi\*-2))/(3 - 1j\*2\*pi\*-2)) + ((1/(1j\*pi\*-2)) \* (exp(-1j\*pi\*-2) - 1))) + (((exp(3) - exp(3/2 + 1j\*pi\*-1))/(3 - 1j\*2\*pi\*-1)) + ((1/(1j\*pi\*-1)) \* (exp(-1j\*pi\*-1) - 1))) + ((exp(3) - exp(-3/2 + 3))/3 - 1) + (((exp(3) - exp(3/2 + 1j\*pi))/(3 - 1j\*2\*pi)) + ((1/(1j\*pi)) \* (exp(-1j\*pi) - 1))) + (((exp(3) - exp(3/2 + 1j\*pi\*2))/(3 - 1j\*2\*pi\*2)) + ((1/(1j\*pi\*2)) \* (exp(-1j\*pi\*2) - 1)));

end

%% Matlab Grader Question 3

function [t, s2\_hinf, sid] = noiseFunc(sid);

% Enter you student ID below:

sid = 10528873;

% Save the appropriate outputs below as defined above:

N = 5;

T = 1;

samples = 100;

time = linspace(-T/2 , T/2, samples+1); time(end) = [];

t = linspace(0,5,500+1); t(end) = [];

s2 = exp(3.\*time + 3);

s2(time >= 0) = -2;

s2\_hinf = repmat(s2, [1,N]);

end

%% Matlab Grader Question 4

function [a0, an, bn, s\_approx, T] = trigFS(s\_hinf, t, N)

Ts = t(2) - t(1);

T = t(end) - t(1) + Ts;

f0 = 1/T;

n\_trig = 1:N;

a0 = 1/T \* sum(s\_hinf) \* Ts;

an = 2/T \* s\_hinf \* cos(2\*pi\*f0\*n\_trig'\*t).' \* Ts;

bn = 2/T \* s\_hinf \* sin(2\*pi\*f0\*n\_trig'\*t).' \* Ts;

s\_approx = a0 + an \* cos(2\*pi\*f0\*n\_trig'\*t) + bn \* sin(2\*pi\*f0\*n\_trig'\*t);

end

%% Matlab Grader Question 5

function [c0, cn, s\_approx, T] = complexFS(s\_hinf, t, N)

Ts = t(2) - t(1);

T = t(end) - t(1) + Ts;

f0 = 1/T;

n\_comp = -N:N;

c0 = 1/T \* sum(s\_hinf) \* Ts;

cn = 1/T \*s\_hinf \* exp(1j \* -2 \* pi \* f0 \*n\_comp' \* t).' \* Ts;

s\_approx = cn \* exp(1j \* 2 \* pi \* f0 \*n\_comp' \* t);

end

## Mission Code:

%% Assignment 1 Part A - Section A2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%.%

% Do not change before line 30.

% If you have not generated Data1A from GenerateDataAssignment1A.m,

% do that now.

% Clearing and preparing the workspace

clear; clc; close all;

% Load assignment data.

load Data1A;

% VARIABLES:

% k - Time vector

% kPeriod - Time vector for 1 period of the interfering noise waveform

% T - Period of signal

% addNoise1st - Your first noise waveform

% addNoise2nd - Your second noise waveform

% c0, cn - Complex Fourier series coefficient vectors

% OR

% a0, an, bn - Trig Fourier series coefficient vectors

% FS1st - Fourier series approximation vector of first noise waveform

% FS2nd - Fourier series approximation vector of second noise waveform

% FSTotal - Fourier series approximation vector of total noise waveform

% dnMsg - De-noised resulting wave (with 5 harmonics)

%==================================================================

% Refer to the assignment sheet for details on variable naming.

% Names of the variables are important,

% e.g. 'a1' is considered a different variable to 'A1'.

%====Enter your code below this line================================

%% Question 1: Plotting the Received

% Length of Noise Signal

lgthNS = length(noiseSound);

% Time vector

t = linspace (0 ,20 ,lgthNS + 1);

t (end) = [ ];

% Plotting received noise

% Sound ( noiseSound ,44100);

figure(1)

plot(t, noiseSound) % Plot of the received noise over time

title('Received Speech Signal')

xlabel('Time (s)')

ylabel('Amplitude')

%% Question 2: Noise Signals Plotted over 1 Period

% Period

samples = lgthNS \* 0.25;

T = 5;

% Time Vector

kPeriod = linspace(0, T, samples + 1); kPeriod(end) = [];

% Time Vector in Halves

kPeriodHalf = linspace(0, T/2, samples/2 + 1); kPeriodHalf(end) = [];

kPeriodHalf2 = linspace(T/2, T, samples/2 + 1); kPeriodHalf2(end) = [];

% The Piecewise function s3(t)

PW1 = (A\*kPeriodHalf -1.25\*A); % First half of the piecewise function

PW2 = (-A\*kPeriodHalf2 + 3.75\*A); % Second half of the piecewise function

% Addition of Noise Vectors

addNoise1st = [PW1 PW2]; % Piece Wise Function

addNoise2nd = kPeriod/A; % Upward Sloping Function

% Plot of the above

hold off

figure(2)

hold on

subplot(1, 2, 1)

plot(kPeriod, addNoise1st)

title('First Additive Noise')

xlabel('Time (s)')

ylabel('Amplitude')

legend('S3(t)')

subplot(1, 2, 2)

plot(kPeriod, addNoise2nd)

title('Second Additive Noise')

xlabel('Time (s)')

ylabel('Amplitude')

legend('S1(t)')

hold off

%% Question 3 Classifying Signals

% Results inside the report

%% Question 4 Fourier Series Approximations of AddNoise2nd

% Complete Complex Fourier Series

% Number of Harmonics

Harm = 5;

compN = -Harm:Harm;

compN(Harm + 1) = []; % Used to remove n = 0 as c0 is present

% Time step

Ts = kPeriod(2) - kPeriod(1);

% Fundamental Frequency

f0 = 1/T;

% Complex Fourier Series

c0 = 1/T \* sum(addNoise2nd) \* Ts;

cn = 1/T \* addNoise2nd \* exp(1j \* -2 \* pi \* f0 \* compN' \* kPeriod).' \* Ts;

FS2nd = c0 + cn \* exp(1j \* 2 \* pi \* f0 \* compN' \* kPeriod);

% Plotting the Complex Fourier Series

hold off

figure(3)

hold on

plot(kPeriod, real(FS2nd), 'b') % Only plots real numbers

plot(kPeriod, addNoise2nd, 'm')

title('Original Signal vs Complex Fourier Series Approximation')

xlabel('Time (s)')

ylabel('Magnitude')

legend('CFS Approximation', 'S1(t)')

hold off

%% Question 5

% Complex Fourier Series Calculated by Hand

CHc0 = 0;

CHcn = (7.5 .\* (exp(1j .\* -pi .\* compN) - 1) ./ (pi.^2 .\* compN.^2));

% Zeros Vector Required for For Loop

CHapprox = zeros(1, samples);

% For Loop

for n = compN

CHapprox = CHapprox + (7.5 .\* (exp(1j .\* pi .\* n) - 1) ./ (pi.^2 .\* n.^2))...

.\* exp(1j .\* 2 .\* pi .\* n .\* f0 .\* kPeriod);

end

% Add the value of n = 0

CHapprox = CHapprox + CHc0;

% Plotting the Complex Fourier Series Calculated by Hand

hold off

figure(4)

hold on

plot(kPeriod, real(CHapprox))

plot(kPeriod, real(addNoise1st), '--')

title('Hand Calculation vs Received Signal')

xlabel('Time (s)')

ylabel('Amplitude')

legend('Hand Calculation', 'Received Signal')

%% Question 6

% Fourier Series Approximations of the Additional Noise

% Additional Noise Vectors

Noise1st = [PW1 PW2]; % Piece Wise Function S3(t)

Noise2nd = kPeriod/A; % Upward Sloping Function S1(t)

% Complex Fourier Series of the 1st Noise

FS1c0 = 1/T \* sum(addNoise1st) \* Ts;

FS1cn = 1/T \* addNoise1st \* exp(1j \* -2 \* pi \* f0 \* compN' \* kPeriod).' \* Ts;

FS1st = FS1c0 + FS1cn \* exp(1j \* 2 \* pi \* f0 \* compN' \* kPeriod);

% Complex Fourier Series of the 2nd Noise

FS2c0 = 1/T \* sum(Noise2nd) \* Ts;

FS2cn = 1/T \* Noise2nd \* exp(1j \* -2 \* pi \* f0 \* compN' \* kPeriod).' \* Ts;

FS2nd = FS2c0 + FS2cn \* exp(1j \* 2 \* pi \* f0 \* compN' \* kPeriod);

% Plotting the Additional Noise Approximation

hold off

figure(5)

hold on

subplot(1, 2, 1)

plot(kPeriod, real(FS1st))

title('First Additional Noise Approximation')

xlabel('Time (s)')

ylabel('Amplitude')

legend('-5 < n < 5')

subplot(1, 2, 2)

plot(kPeriod, real(FS2nd))

title('Second Additional Noise Approximation')

xlabel('Time (s)')

ylabel('Amplitude')

legend('-5 < n < 5')

%% Question 7

% The Fourier Series Total

FSTotal = [FS1st FS1st FS2nd FS2nd];

% Plot of Fourier Series Total

hold off

figure(6)

hold on

plot(t, real(FSTotal))

title('Fourier Series Total')

xlabel('Time (s)')

ylabel('Amplitude')

%% Question 8

% De-Noising the Original Signal

dnMsg = noiseSound - real(FSTotal).';

%% Question 9

% Results inside the report

%% Question 10

% Plotting the De-Noised Signal

hold off

figure(7)

hold on

plot(t, dnMsg)

title('The De-Noised Speech Signal')

xlabel('Time (s)')

ylabel('Amplitude')

% load chirp.mat

% sound(real(dnMsg), 44100)