EGB 242 Assignment 2

Message From Mars-242 Mission Control

Group Assignment

Joseph Haddad	n10535268
Ryan Brooker	n10730036
Mitchell Hatton-Whimp	n10222502

Introduction:

The mission at hand as assigned by BASA’s chief engineers involves working with the live MARS - 242 mission. In which we need to apply previous skills and knowledge attained by the previous missions in Assignments 1A and 1B. In order for this Mars mission to succeed, the Astronauts physical and psychological health needs to be monitored. Whilst ensuring selection of an appropriate landing point for them is important so that they can safely land.

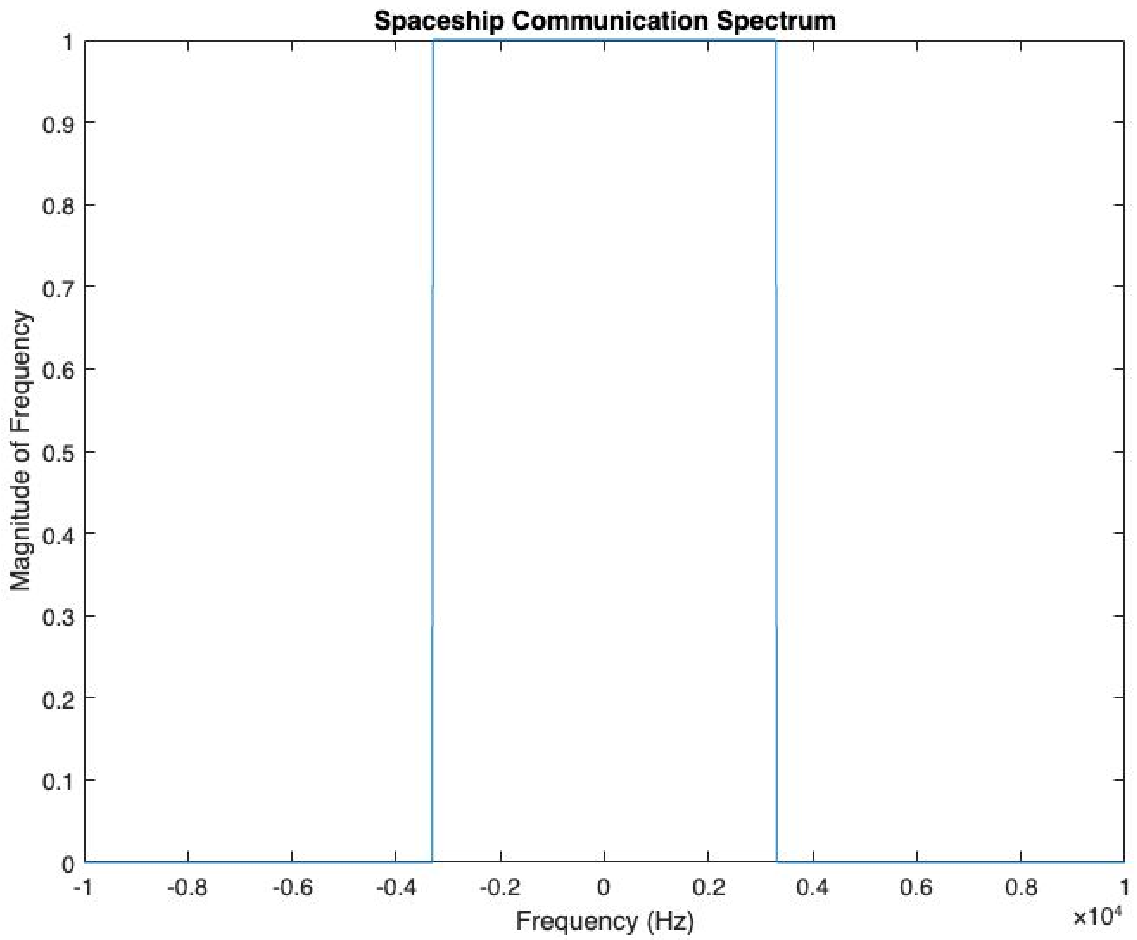
Section 1:

1.1 - Rover Spectrum

The spectrum of the spaceship's communication module is modelled as a constant magnitude function between the generated frequencies of 3300Hz. This can be mathematically modelled as.

$$S(F) = A * rect(\frac{f}{6600})$$

Figure 1.1: The Spaceship's communication module in the frequency domain.



The figure above shows that the Spaceships communication spectrum is between 3300 Hz.

1.2 - Rover Spectrum

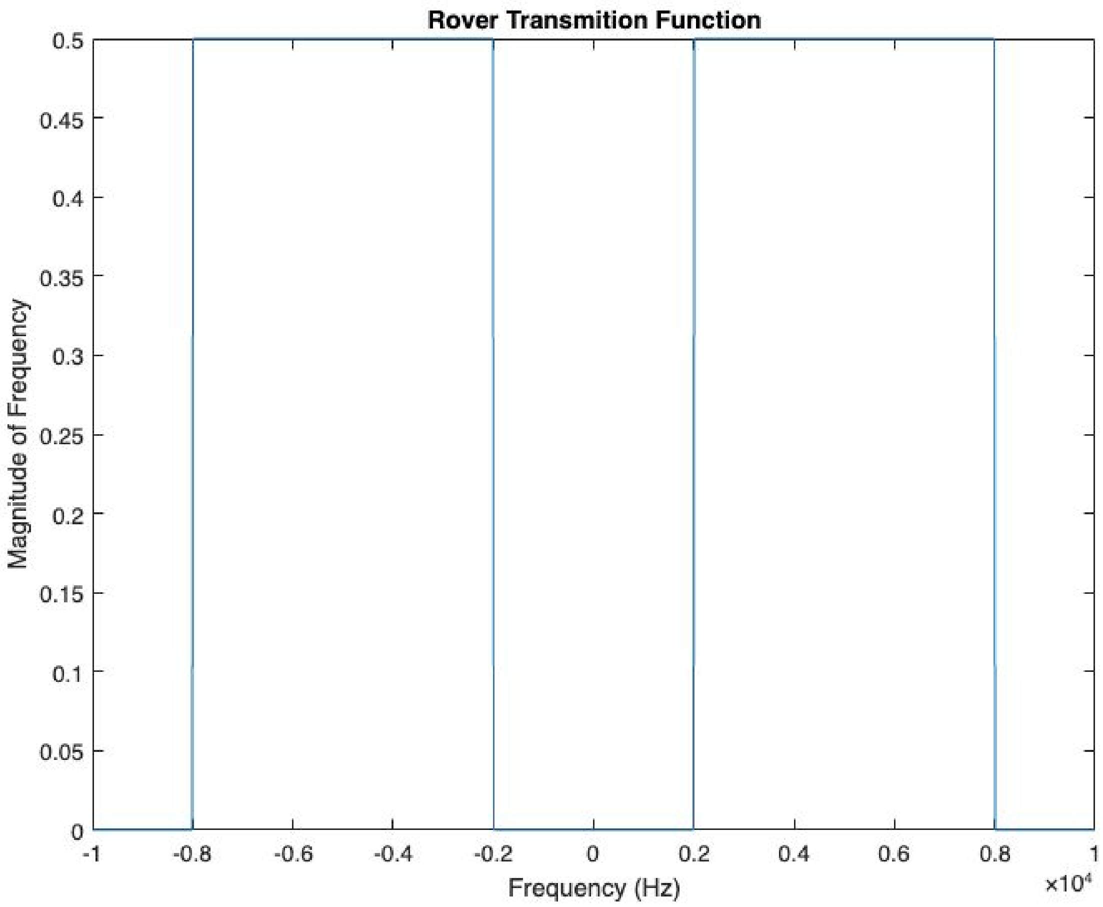
The function that models the rover's transmission module can be expressed as.

$$Srov(t) = 6000\,Sinc(6000 \cdot t)\sin\left(10 \cdot \pi \cdot 10^3 + \pi/2\right)$$

From this function the Fourier Transform could be derived to get the Rover's transmission function in the frequency domain. The calculations of the are shown in appendix A. The function can be shown as.

$$\mathrm{Srov(F)} = \frac{1}{2}\left[rect\left(\frac{t}{6000} - 5000\right) + rect\left(\frac{t}{6000} + 5000\right)\right]$$

Figure 1.2: The Rover's Transmission Function.



The figure above shows that the Rover's transmission function is between the frequencies 2000 Hz and 8000 Hz and has a magnitude of 0.5 units. The information in the graph will be used to create a filter to ensure that the Rover's transmissions are received.

1.3 - Rover Spectrum

The ideal filter in order to eliminate the spaceships broadcast information would be a High Pass Filter (HPF). The HPF will be able to eliminate all signals with the frequency that coincide with the Spaceship's communication module whilst allowing for the Rover's transmission to be received. The transmission function for this ideal filter can be expressed as.

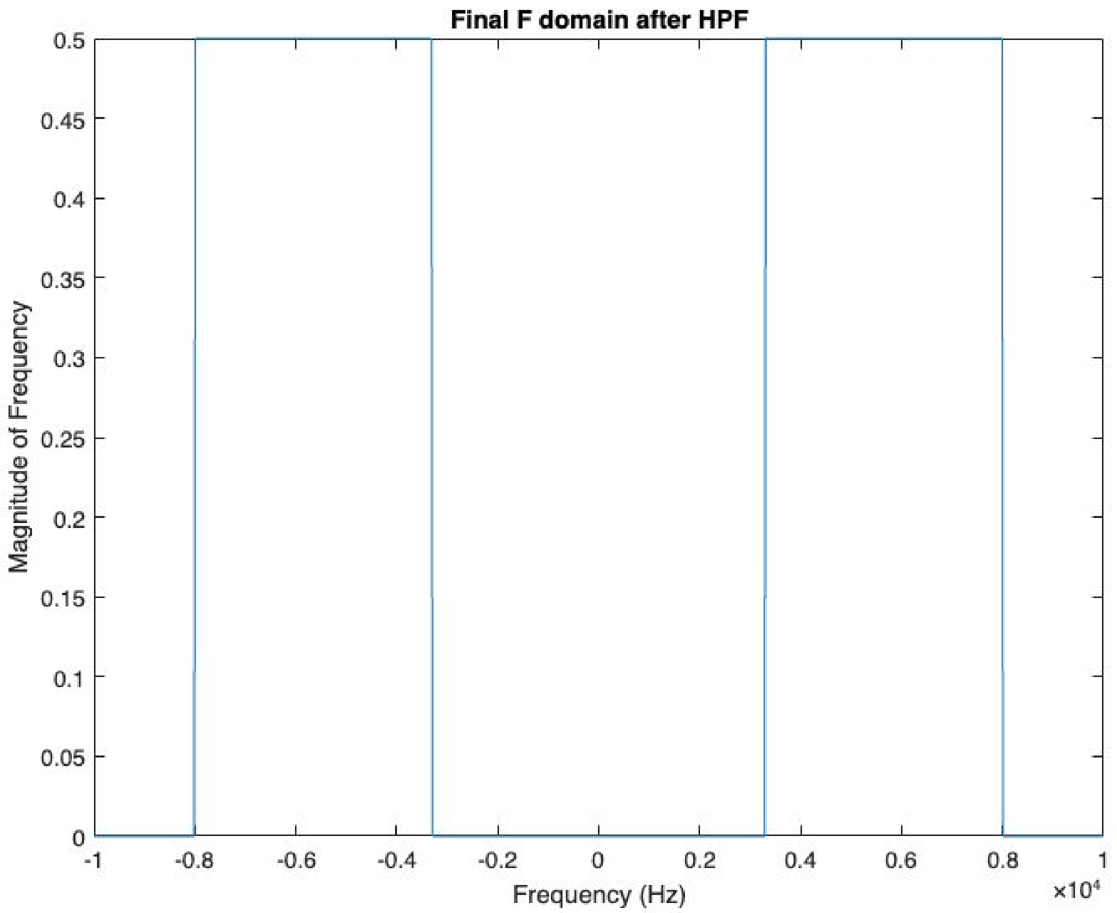
$$Hfilt = \begin{cases} 1 & if\ \ f \leq -3300 \\ 0 & otherwise \\ 1 & if\ \ f \geq 3300 \end{cases}$$

The HPF is the simplest filter to be used in this situation. This is due to it only needing to eliminate the lower frequencies between 3300 Hz. These lower frequencies are created by the Spaceships communication module, therefore, for the Rover's transmission signal to be received these lower frequencies will be eliminated. The filter does not eliminate any other frequencies. This makes the HPF the simplest filter to create as there will be no higher frequencies that will need to be eliminated outside the Rover's transmission. Therefore, the HPF is most efficient filter for this situation.

1.4 - Rover Spectrum

The figure below shows the frequency domain after the ideal HPF is applied.

Figure 1.3: The frequency domain after HPF is applied.



The Figure above shows that the Spaceships communication channel has been eliminated from the total signal received. The only signal frequency left is the frequency that has been broadcasted by the Rover. Although the HPF allows for frequencies above 8000Hz, the Rover's broadcast frequencies stop at 8000Hz, this implies there was no need to have a more complicated Band Pass Filter to eliminate the higher frequencies above 8000Hz.

1.5 - Rover Spectrum

The impacts that the HPF will have on the information transmitted by the Rover between the frequencies 2000Hz and 3300Hz is that there will be a loss of information as the Spaceship's communication spectrum frequencies overlaps these frequencies with the Rover's transmission frequencies. As the mission was to eliminate the Spaceship's communication channel the filter also eliminated some frequencies the Rover was transmitting. This means that some information the Rover's was broadcasting is lost due to the HPF. To mitigate the loss of information you could have the rover broadcast its information at a higher frequency where the spaceships communication channel does not overlap and therefore, the HPF will not eliminate the information being transmitted by the rover.

The practical use of a non-ideal filter will mean that the HPF will eliminate more information as the filter cannot transition between the magnitude of 0 and 0.5 instantaneously. A non-ideal filter has a transition band where the HPF doesn't instantaneously transition from the stop to pass band. This means that the transition band allows for some of the spaceships communication frequencies to be present and some loss of some of the rover's transmission frequencies after the HPF is applied. The non-ideal HPF will have this transition band, therefore, when the filter gets to 3300Hz, the filter will not instantaneously pass the frequencies and more information will be eliminated and some Spaceship transmission frequencies will be passed. This means when using the non-ideal HPF, there will need to be consideration of the transition period from the stop to pass bands as there will be frequencies transmitted from the Spaceship's broadcast and some Rover's transmission frequencies magnitudes will be reduced during this transition band.

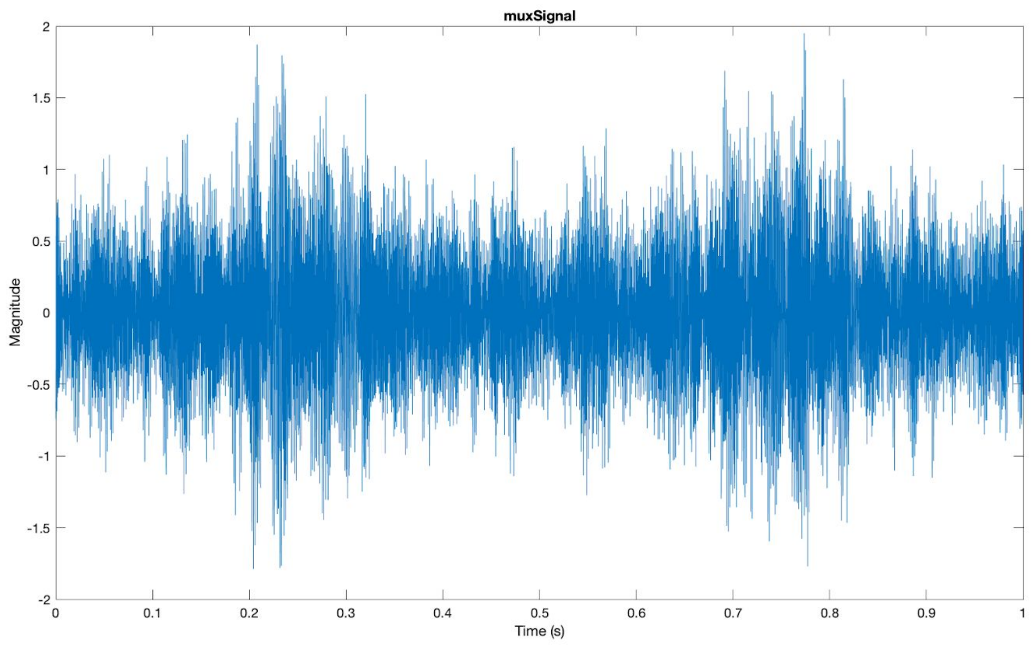
Section 2:

2.1 - Represent the Spectrum Analyser

The first steps towards fixing the faulty filter module is to determine some basic parameters, then to compute and plot the Fourier transform of the unfiltered signal.

The time-step (sampling period) of the signal is simply the inverse of the sampling frequency which is known. Now, knowing the sampling period and the length of the signal, a time vector can be constructed. This will allow for the plotting of the original unfiltered signal which is shown in figure 2.1 and directly below it, the code to produce the sampling period and time vector.

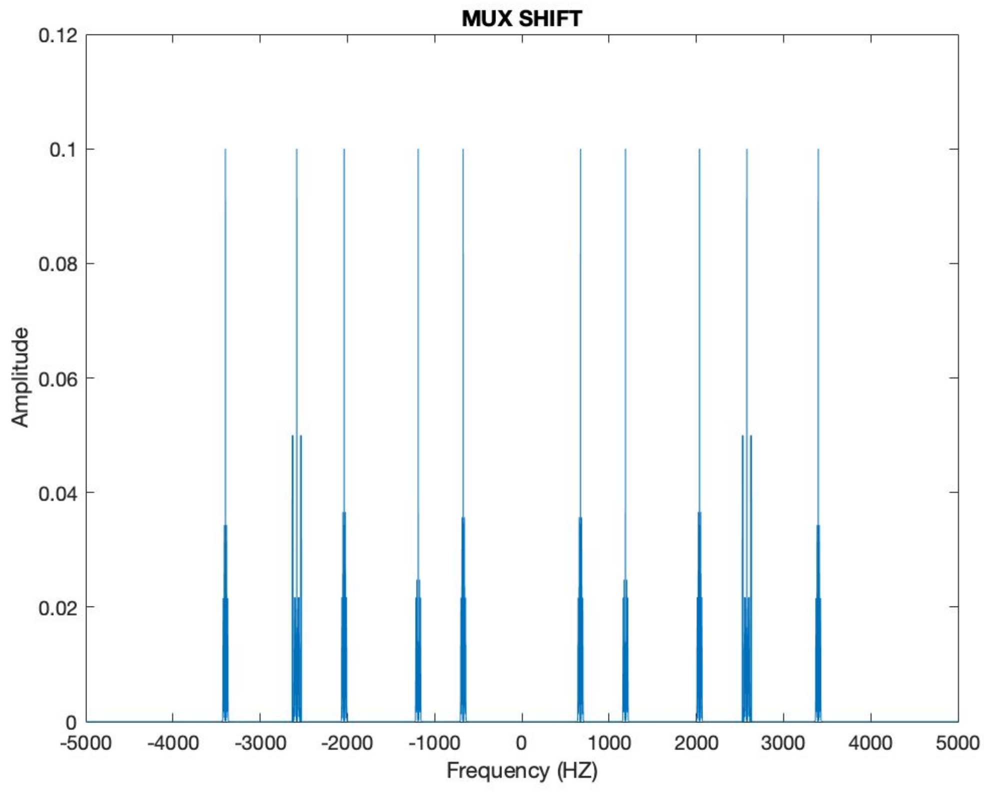
Figure 2.1 - Graph of the given signal in the time domain.



```
Ts = 1/fs;  % time step
Ls = length(muxSignal);
t = 0:Ts:Ls*Ts; t(end) = [];    % time vector
```

The next step is to compute a Fourier transform of the signal and create a magnitude plot. This will allow for visualisation of the frequency shifts. To complete this step, a frequency vector must also be created. This vector is created such that the endpoints are +/- the Nyquist frequency. The magnitude plot is shown in figure 2.2 below with the corresponding code directly beneath it, noting that the magnitude has been shifted and scaled in the plotting process.

Figure 2.2 - Graph of the given signal in the frequency domain.



```
k = linspace(-fs/2, fs/2, Ls + 1); k(end) = [];
MUXSIG = fft(muxSignal);
```

2.2 - Determining Demultiplexing Parameters

In section 2.1, it became clear that there are numerous frequency shifts in the signal all of equal or similar magnitude. The next step is to exactly locate these frequency shifts and their corresponding magnitudes. The code to complete this is shown below, noting that it is essentially the same process used to complete tasks previously assigned by BASA and that the magnitudes are now shifted and scaled.

```
MUXSIG_shift = abs(fftshift(MUXSIG)/fs);
[x_pos, y_pos] = findpeaks(MUXSIG_shift,'MinPeakHeight',0.09);

fshift = k(y_pos);
fshift = fshift(fshift >= 0);

% find the magnitude spectrum
Mag = abs(x_pos(k(y_pos)>0));

% find the phase spectrum
Phase = fftshift(angle(MUXSIG));
Phase = Phase(y_pos(6:end));
```

2.3 - Remove Frequency Shifts

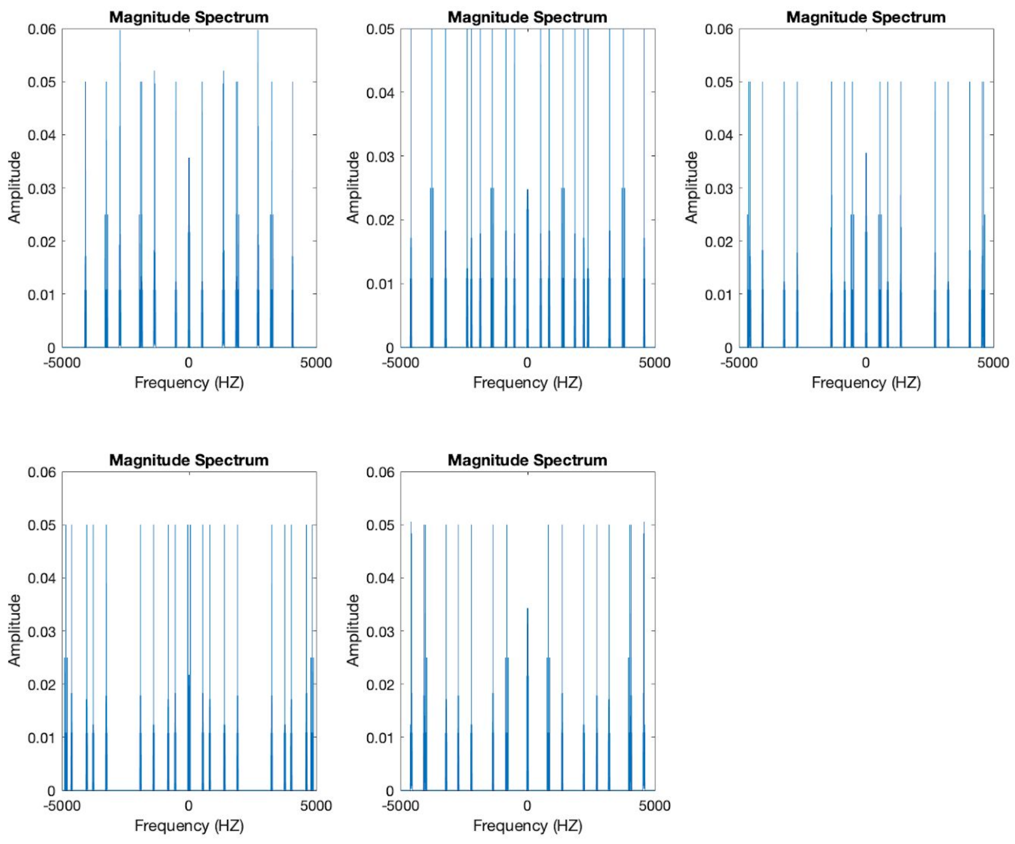
In a previous task assigned by BASA, a frequency shifting module was created but here the module has been given, hence, the next step is to pass the parameters determined in previous sections to the module. This will remove the frequency shifts previously identified.

```
xdm = FDMDemux(muxSignal.',t,Mag.',fshift,Phase.');
```

2.4 - Review

Now that the frequency shifts have been removed, the next step is to compute the Fourier transform of each shift and plot the magnitude spectrum. This is shown in figure 2.3 below and the corresponding code is shown directly below the figure.

Figure 2.3 - Graph of Magnitude Spectrum of XDM.



```
% Computing the fourier transform
XDM(1,:) = fft(xdm(1,:));
XDM(2,:) = fft(xdm(2,:));
XDM(3,:) = fft(xdm(3,:));
XDM(4,:) = fft(xdm(4,:));
XDM(5,:) = fft(xdm(5,:));

% Calculating the Magnitude Spectrum
XDM_MAG(1,:) = fftshift(abs(XDM(1,:))/fs);
XDM_MAG(2,:) = fftshift(abs(XDM(2,:))/fs);
XDM_MAG(3,:) = fftshift(abs(XDM(3,:))/fs);
XDM_MAG(4,:) = fftshift(abs(XDM(4,:))/fs);
XDM_MAG(5,:) = fftshift(abs(XDM(5,:))/fs);
```

2.5 - Mathematical Analysis

The next step is to filter the unwanted frequencies, but to complete this step, the most suitable filter must be selected. Analysing the given transfer functions will allow for calculation of the “poles and zeros” of each filter, thereby allowing for the immediate exclusion of unstable filters.

System 1:

```
                          1.689e04 s
                    ----------------------
                    s^2 - 4222 s + 1.783e07

                      After Factorising

                         16889.2021s
       --------------------------------------------------------
       (s - 2111.1503 - j3656.6195)(s - 2111.1503 + j3656.6195)
```

After factorising as shown above, it’s clear that the first system has two complex poles and a single zero at s = 0. The first complex pole has a positive real and imaginary part, whereas the second has a positive real and negative imaginary part. This defines an unstable system and hence it is not appropriate to use.

System 2:

```
                            2.157e07
                    ----------------------
                    s^2 + 6568 s + 2.157e07

                      After Factorising

                        21571664.2999
       --------------------------------------------------------
       (s + 3284.1791 + j3284.1791)(s + 3284.1791 - j3284.1791)
```

After factorising as shown above, it’s clear that the second system has two complex poles and no zeros. The first complex pole has a negative real and imaginary part, whereas the second has a negative real and positive imaginary part. This defines a stable system and hence it is potentially appropriate to use.

System 3:

```
                     1.141e04 s
              ----------------------
              s^2 + 1257 s + 1.783e07

                  After Factorising

                    11410.2645s
  ------------------------------------------------------
  (s + 628.3185 + j4175.2889)(s + 628.3185 - j4175.2889)
```

After factorising as shown above, it’s clear that the third system has two complex poles and a single zero at s = 0. The first complex pole has a negative real and imaginary part, whereas the second has a negative real and positive imaginary part. This defines a stable system and hence it is potentially appropriate to use.

System 4:

```
                       3.669e05
              ----------------------
              s^2 + 1211 s + 3.669e05

                  After Factorising

                      366909.7996
             ----------------------------
             (s + 605.7308)(s + 605.7308)
```

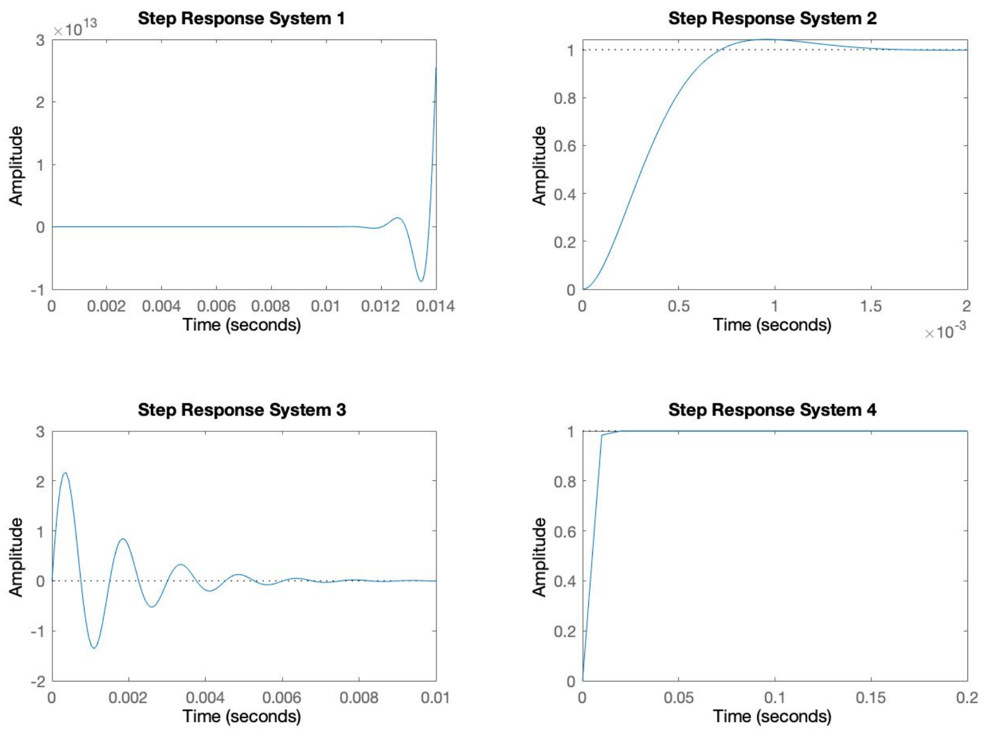
After factorising as shown above, it’s clear that the third system has two real poles and no zeros. The first and second real pole has a negative real part. This defines a stable system and hence it is potentially appropriate to use.

This mathematical analysis has eliminated one potential filter meaning that further analysis is required to determine which of the remaining three is the most appropriate.

2.6 - System Analysis

Using MATLABs LTI Viewer, further analysis can be performed to choose the most appropriate filter. The step response of each filter is shown in figure 2.4 below. Note that despite Filter 1's inclusion in the following figures, it has already been eliminated as a candidate in the previous section.

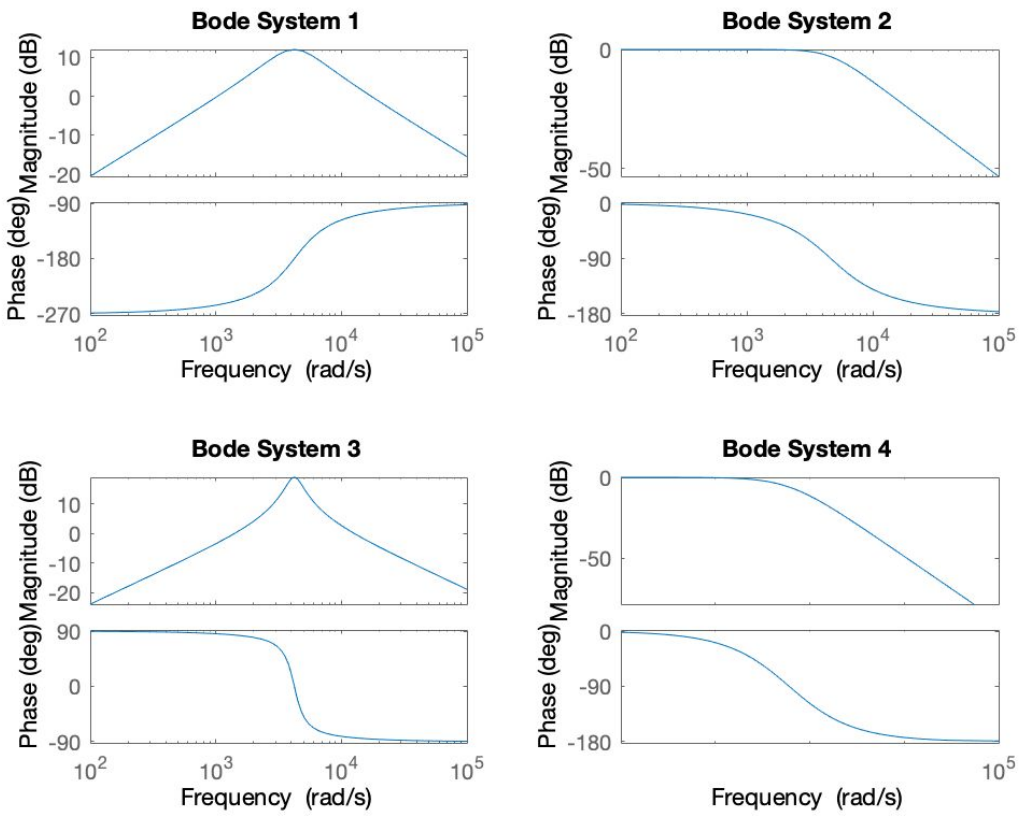
Figure 2.4 Step response of various filters



The step response analysis confirms what was determined in the mathematical analysis (i.e. that Filter 1 is inappropriate) as its step response tends towards infinity. The step response of filters 2 and 4 tends towards 1 whereas filter 3's response tends towards 0.

The bode plot of each system is shown below in figure 2.5

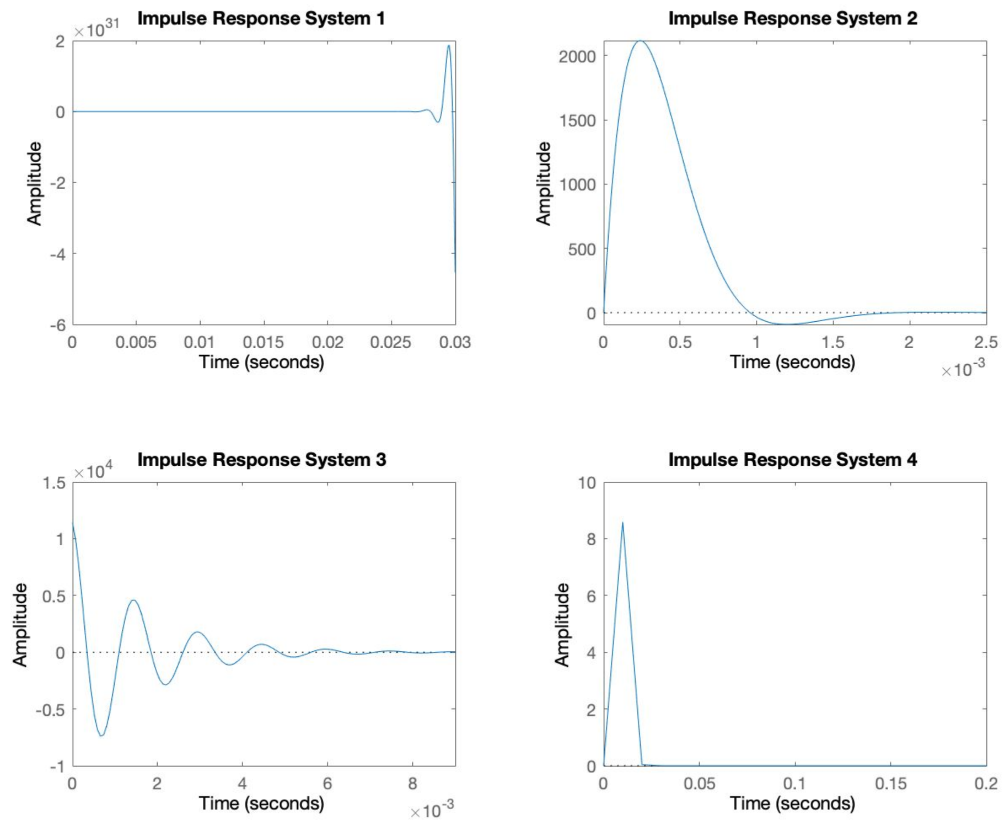
Figure 2.5 Bode plot of various filters



System 2 seems to begin attenuating frequencies of above approximately 750Hz. The gain in the passband is approximately 0. System 3 seems to begin attenuating frequencies below 575Hz and above approximately 780Hz and has a maximum gain of approximately 19.1dB in its passband. System 4 seems to begin attenuating frequencies above approximately 400Hz and has a gain of approximately 0 in its passband.

The impulse response of each system is shown below in figure 2.6.

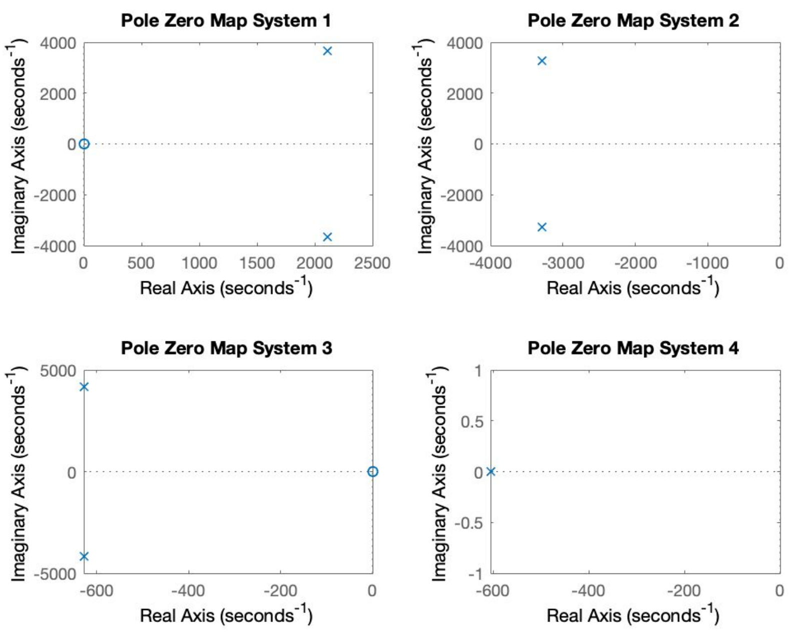
Figure 2.6 Impulse response of various filters



The impulse responses of filters 2, 3, and 4 all eventually tend towards zero. Systems 2 and 4 have a peak soon after 0s before they begin tending towards zero.

The pole-zero map of each system is shown below in figure 2.7.

Figure 2.7 Pole-zero map of various filters



The pole-zero map simply confirms what was calculated in section 2.5.

2.7 - Recommendation

System 4 is the most appropriate filter for this application. Firstly, it is a stable filter as per the pole-zero map and mathematical analysis. Secondly, filter 2 attenuates frequencies below 750Hz which includes too much of the interfering frequencies. Thirdly, filter 3 attenuates frequencies below 575Hz which would attenuate the frequencies we wish to keep. Using filter 4 would mean that frequencies above 400Hz would begin to be attenuated, as well as the fact that its step response holds at an amplitude of 1 and impulse response has a peak in the beginning before it tends towards 0.

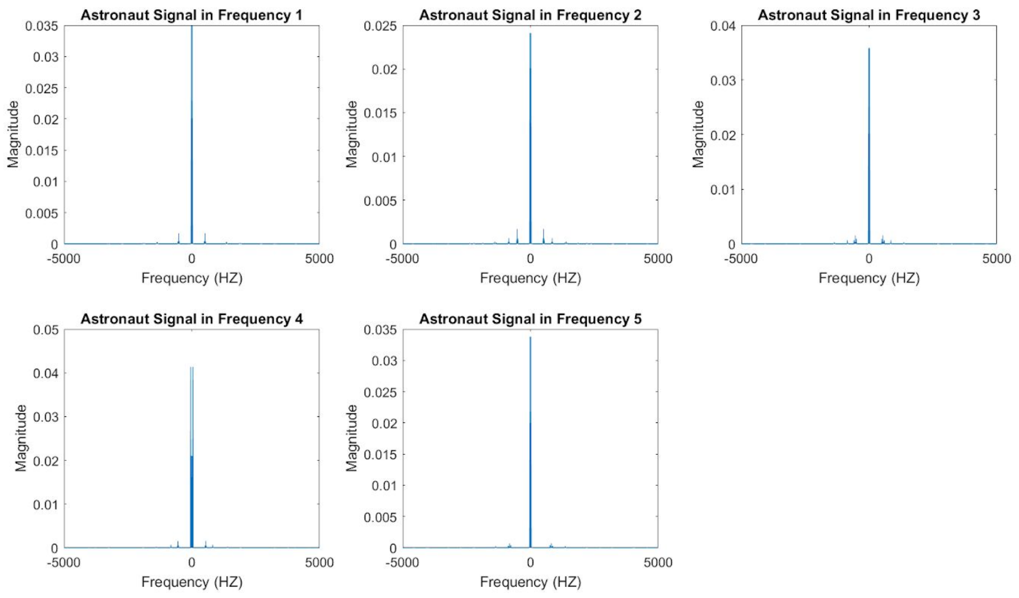
2.8 - Filter the Signal

Now that a filter has been selected, the next step is to filter the signal. This is done by first obtaining the impulse response of the system in the time domain, then applying a Fourier transform to this response and scaling it to bring it into the frequency domain. Now, since the original signal has been Fourier transformed and passed through the frequency shifting module, each of the data streams can be multiplied by the impulse response (since they're both in the frequency domain). The code to complete this is shown below.

```
system_response = impulse(sys(4),t)';
system_response = fft(system_response)/fs;

EEG(1,:) = system_response .* XDM(1,:);
EEG(2,:) = system_response .* XDM(2,:);
EEG(3,:) = system_response .* XDM(3,:);
EEG(4,:) = system_response .* XDM(4,:);
EEG(5,:) = system_response .* XDM(5,:);
```

Figure 2.8 Magnitude plot of the filtered streams

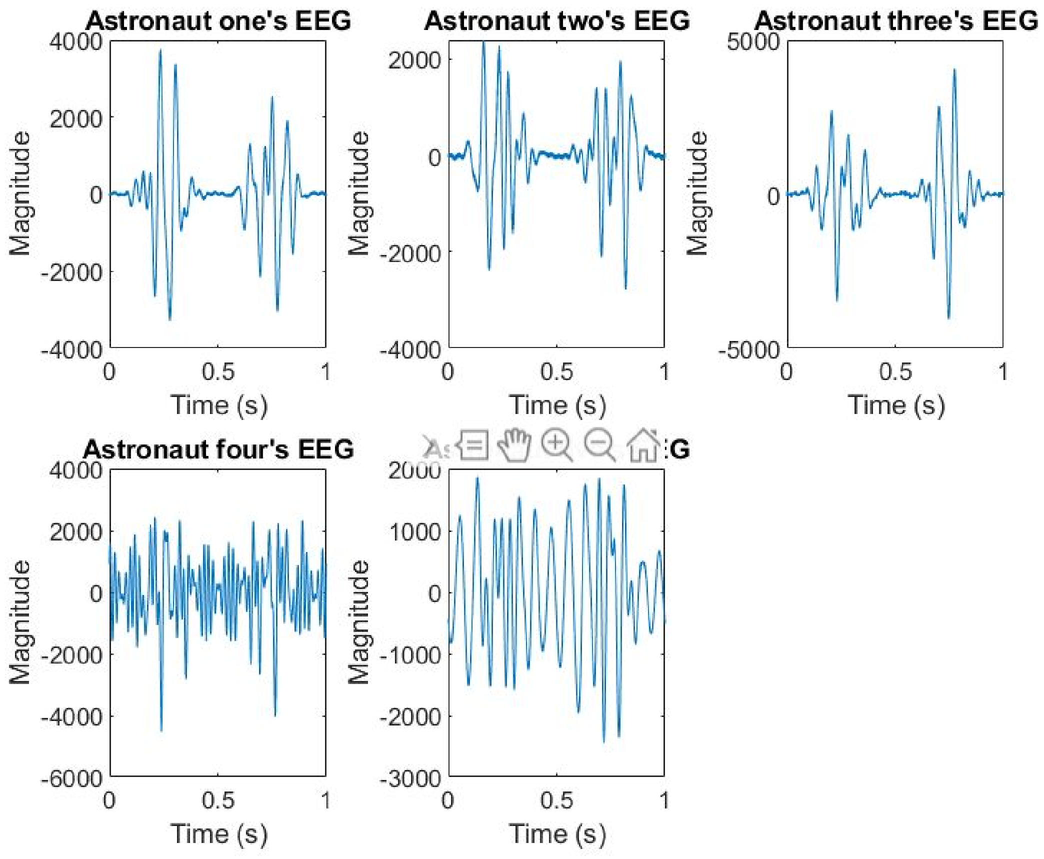


Now, these streams have been filtered and are ready to be returned to the time domain. To complete this step, an inverse Fourier transform is applied with appropriate shifting and rescaling. Also noting that only the real part is taken and that detrend is applied in MATLAB to remove the DC offset. The code to complete this step is shown below.

```
eeg(1,:) = detrend(real(ifft(ifftshift(EEG(1,:))*fs)));
eeg(2,:) = detrend(real(ifft(ifftshift(EEG(2,:))*fs)));
eeg(3,:) = detrend(real(ifft(ifftshift(EEG(3,:))*fs)));
eeg(4,:) = detrend(real(ifft(ifftshift(EEG(4,:))*fs)));
eeg(5,:) = detrend(real(ifft(ifftshift(EEG(5,:))*fs)));
```

Now, the astronauts' EEGs have been successfully filtered and can be viewed in the time domain. This is shown in figure 2.9 below.

Figure 2.9 Astronauts' EEGs in the time domain after filtering with system 4



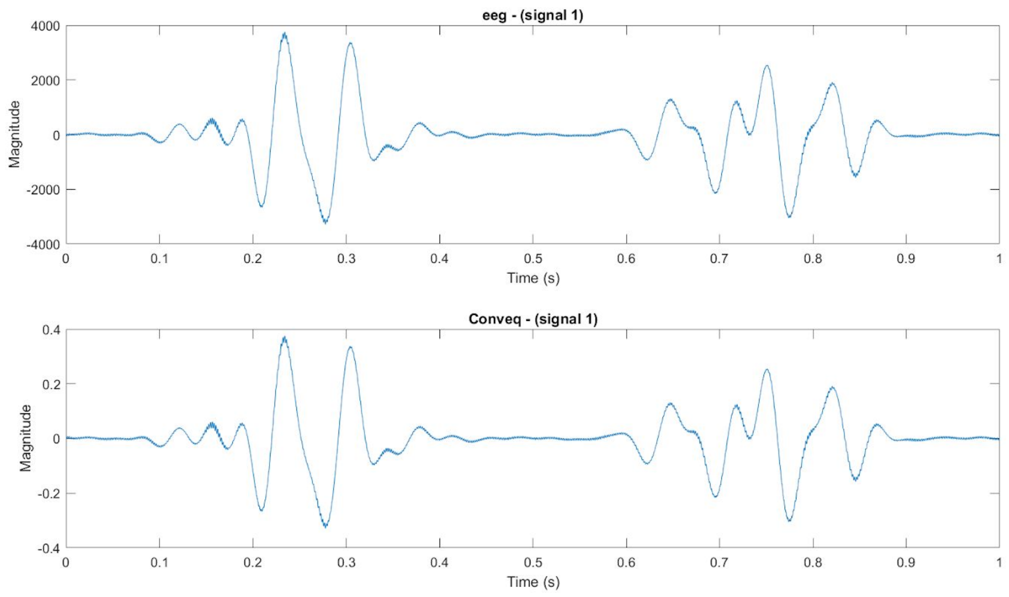
2.9 - Equivalence With Convolution

Mathematically, multiplication in the frequency domain is equivalent to convolution in the time domain. This fact can be used to check the results obtained in section 2.8. Now, this means that the impulse response of system 4 in the time domain must be convoluted with the astronauts' EEGs in the time domain. The code to complete this step is shown below.

```
impresp = impulse(sys(4),t);
Conveq = conv(impresp, xdm(1,:)/fs);
Conveq = Conveq(1: length(t));
```

Now, the resulting Conveq vectors can be plotted in MATLAB for comparison with section 2.8. Astronaut 1's EEG by convolution in the time domain and the same astronaut's EEG but by the methods outlined in section 2.8 is shown in figure 2.10 below.

Figure 2.10 Method comparison using astronaut 1



The signals look the exact same. This is expected since mathematically they should be equivalent.

2.10 - Compare

Astronaut 1, 2, and 3 all have EEGs in the time domain extremely similar to the example in figure two of the project brief. The minor differences can likely be attributed to the fact that EEGs vary slightly across people since the human body isn't an exact system. It would be unreasonable to expect that astronaut 1, 2, and 3's EEGs would match the example exactly, rather, it should be expected that they follow the general pattern in the example which they do. As for astronaut 4 and 5, further filtering is needed in section 2.11 to determine whether the significant differences in the time domain is due to remaining interference.

As for the frequency domain, after rescaling the plot in figure 2.8, it's clear that the magnitude plots are again very similar, however, the plots of the astronauts seem to have larger peaks as if they've been stretched vertically. The signal is also concentrated almost entirely in -50Hz to 50Hz, just like the example. As for astronauts 4 and 5, the differences in the frequency domain are significant and again, further analysis to rule out additional interference is required.

2.11 - Digital - denoising

As shown in previous sections, some interference remains in the filtered signals. It has been discovered that this is due to single-frequency interference noises. It has been decided that a rectangular pulse in the frequency domain will be multiplied with the signals to remove this remaining noise. The code to complete this step is shown below.

```
B = 50;    % cutoff frequency selected
 low_pass_filter = rectpuls(k,(B*2));
EEG(1,:) = (fftshift(EEG(1,:))/fs) .* low_pass_filter;
EEG(2,:) = (fftshift(EEG(2,:))/fs) .* low_pass_filter;
EEG(3,:) = (fftshift(EEG(3,:))/fs) .* low_pass_filter;
EEG(4,:) = (fftshift(EEG(4,:))/fs) .* low_pass_filter;
EEG(5,:) = (fftshift(EEG(5,:))/fs) .* low_pass_filter;
```

This will completely remove any frequencies larger than 50Hz from the signal. 50Hz was chosen since almost all of the EEG data is concentrated in this range and hence, it's reasonable to suspect any frequencies outside of this range is unwanted interference.

2.12 - Visual Analysis

Astronauts 1, 2, and 3 all have EEGs in the time and frequency domain that are very similar to the example given in the project brief.

Astronauts 4 and 5 have EEGs that do not share characteristics with each other or the example given in the brief and hence can be deemed abnormal.

Section 3:

3.1 - View the Noisy Image

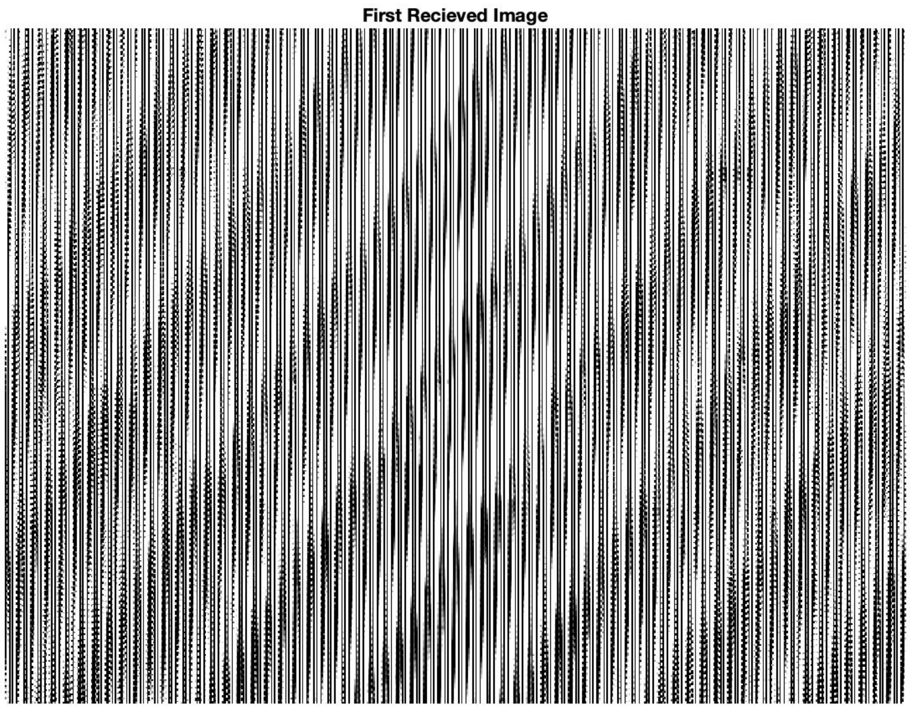


Figure 3.1 - Image of the first received image in its noisy state.

3.2 - Reference Vectors

Since the sample rate is (fs = 1000 pixels/second) this can be used to find Ts by doing (1/fs). This allows for construction of the t and k - (time and frequency) variables as shown in the code below.

```
Fs = 1000;  % pixels per seconds
Ts = 1/Fs;
Ls = length(sig);   % length of the signal

t = 0:Ts:Ls*Ts; t(end) = [];    % time vector
k = linspace(-500, 500, Ls + 1); k(end) = [];   % frequency vector
```

3.3 & 3.4 - Visualising the Received Signal - Estimating Periodic Noise

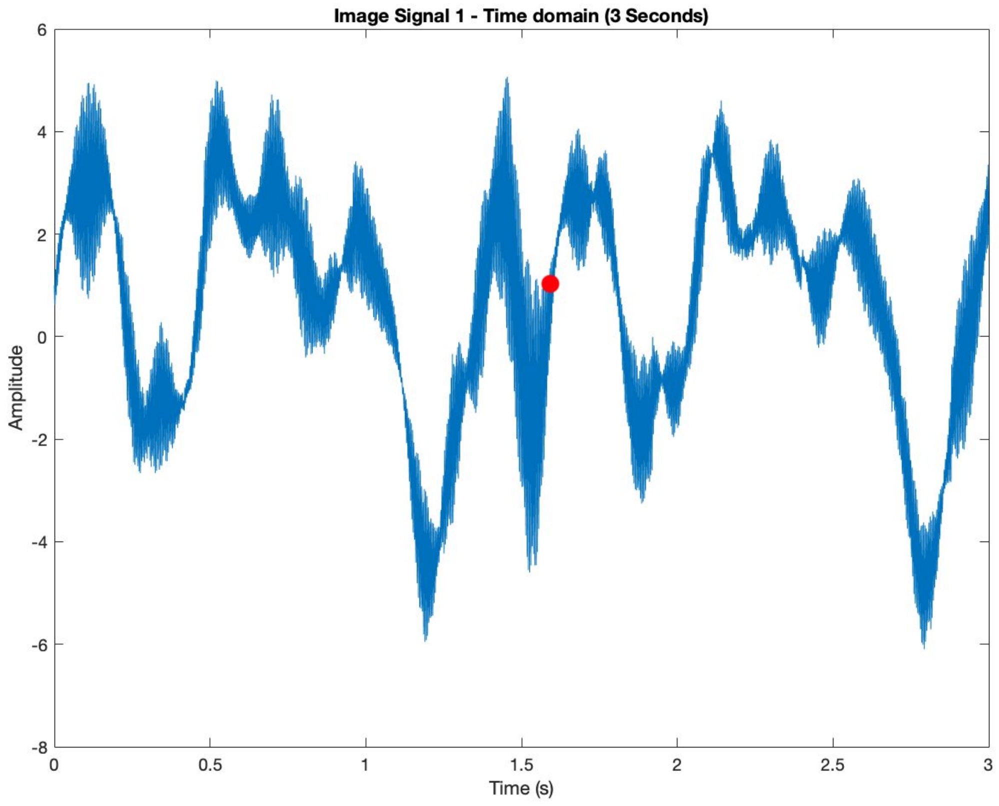


Figure 3.2 - Image of the first received image in the time domain. (Noisy)

As pictured above in figure 3.2 graphically over three seconds there is a visible period. The position of it is marked by a red dot. Through use of the given candidateT variable the closest number to this position is 1.5950. This in turn is the estimated periodic noise in which the variable (T) is set to.

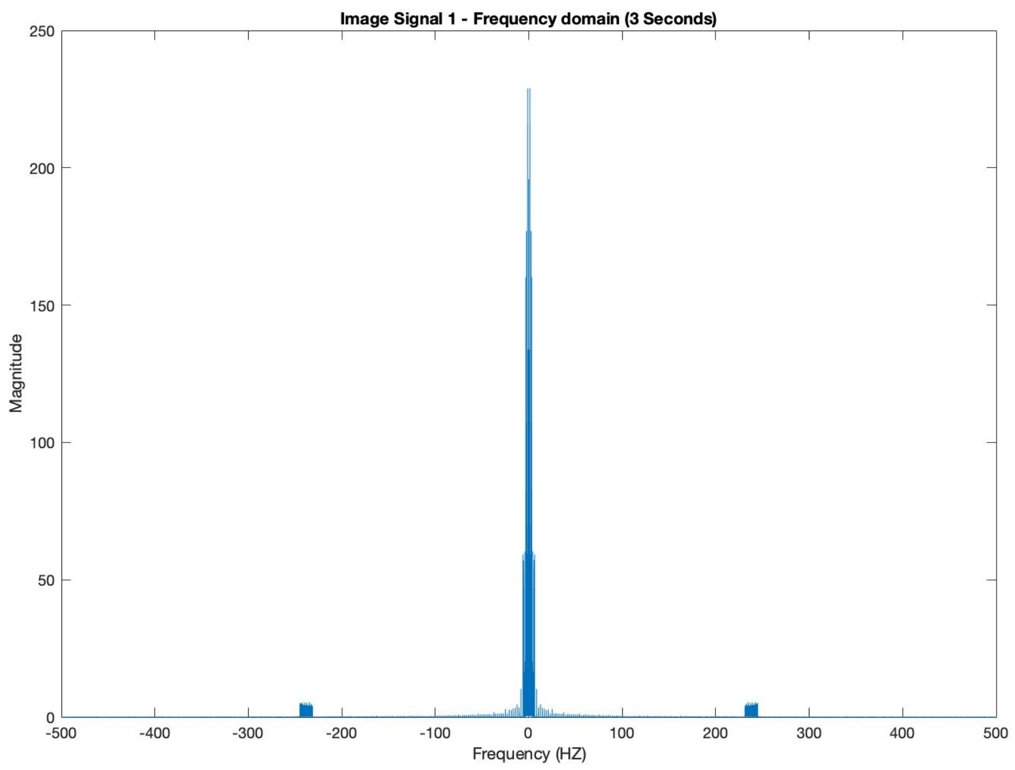


Figure 3.3 - Image of the first received image in the frequency domain. (Noisy)

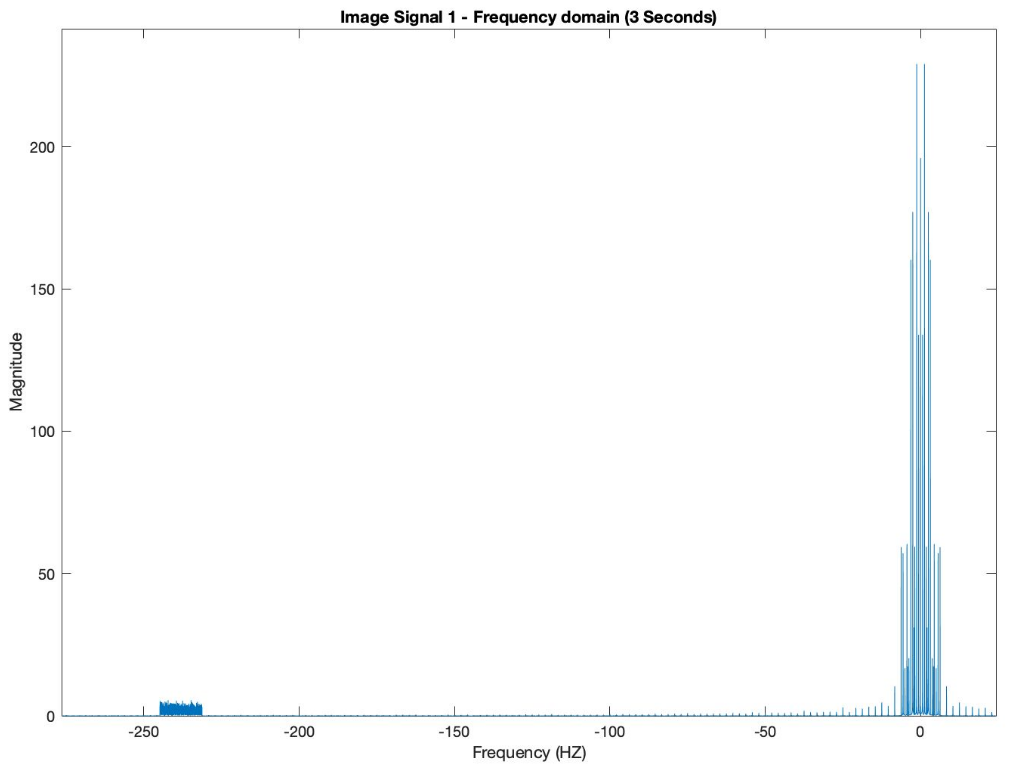


Figure 3.4 - Image of the first received image in the frequency domain. (Zoomed into the bandwidth)

In the above figures 3.3 & 3.4 are images of the signal in the frequency domain. These show the bandwidth of the signal which is approximately 220hz.

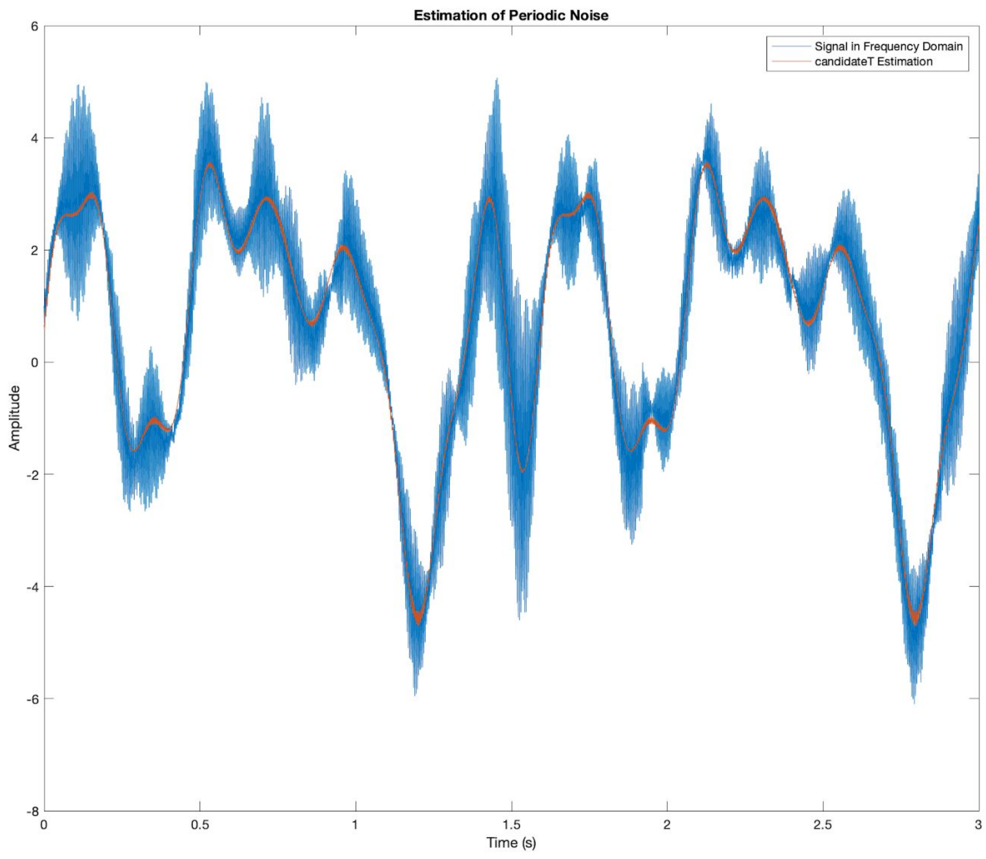


Figure 3.5 - Estimation of the Periodic Noise

In the above figure 3.5 is the estimation. This has been computed by selection of a candidateT. As shown by the orange lines overlaid on top of the blue. The approximation developed by the complex transform. Accurately follows the signal.

3.5, 3.6 & 3.7 - Modelling the Periodic Noise - DC Bias - Generation of the Approximation

```
N = 25;  % Harmonics

nTrig = (1:N).';

a0 = 1/T * sum(sig(1,:)) * Ts;
an = 2/T * sig(1,:) * cos(2*pi*(1/T)*nTrig*t).' * Ts;
bn = 2/T * sig(1,:) * sin(2*pi*(1/T)*nTrig*t).' * Ts;

c0 = 0; % make zero to remove DC offset

cn_pos = 1/2 * (an-1j*bn);
cn_neg = 1/2 * (an + 1j*bn);

cn_neg = fliplr(cn_neg);

cn = [cn_neg c0 cn_pos];

nComp = (-N:N).';
s_approx = cn * exp(1j * 2 * pi * (1/T) * nComp * t);

Noisesig_fs = s_approx./197;    % generated approximation
```

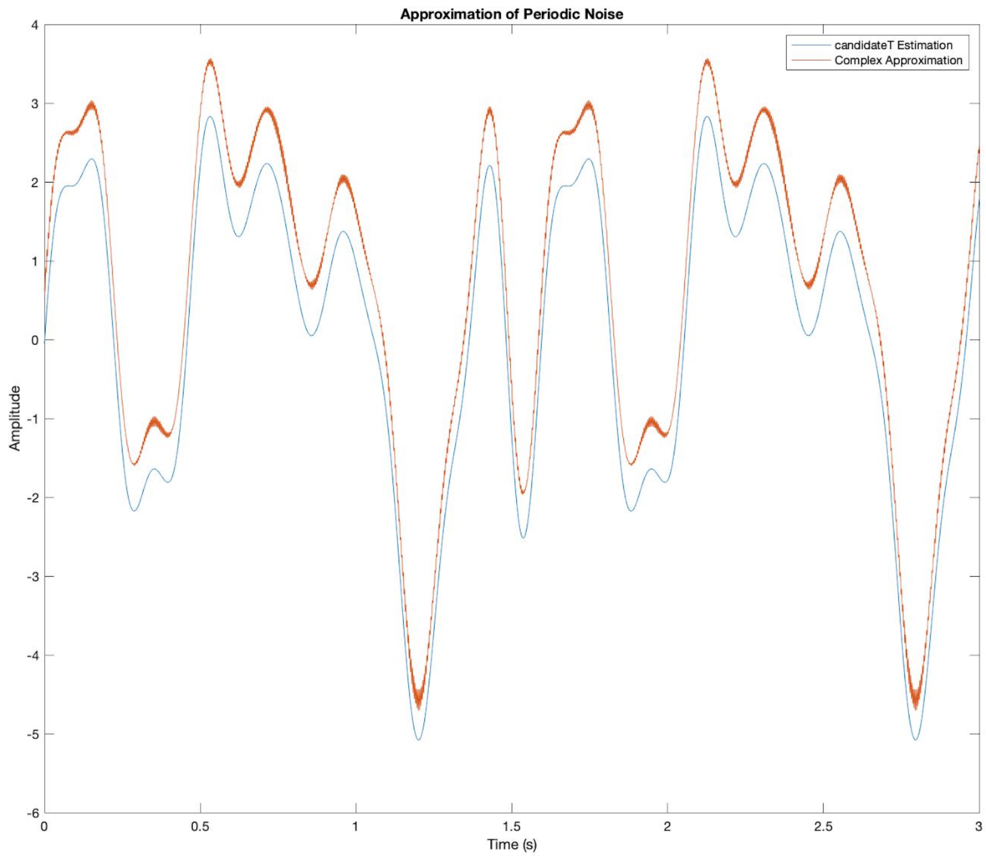


Figure 3.6 - Approximation of Periodic Noise.

Through use of a complex fourier transform an approximation as shown in orange is developed. There are two adjustments to the variables that have been made in order to achieve the best approximation. In order to remove the DC offset a0 = c0 = 0. Since this is the DC component of the approximation. It means that the DC Bias in the signal is removed. Furthermore, in order to account for scale. s_approx is multiplied by 197. Which through testing different values proved to be the optimal value providing the least distortion. In the graph above 25 harmonics were used. This provided sufficient accuracy. Any more harmonics would be a waste of computing time as they do not provide a better resulting signal.

3.9 & 3.10 - De-noise - Remove Bandlimited Noise

```
figure(7)
im1(1,:) = fft(sig(1,:) - real(Noisesig_fs));
im1(2,:) = fft(sig(2,:) - real(Noisesig_fs));
im1(3,:) = fft(sig(3,:) - real(Noisesig_fs));
im1(4,:) = fft(sig(4,:) - real(Noisesig_fs));

im1(1,:) = fftshift(im1(1,:));
im1(2,:) = fftshift(im1(2,:));
im1(3,:) = fftshift(im1(3,:));
im1(4,:) = fftshift(im1(4,:));

plot(k, im1(1,:))

B = 220;    % cutoff frequency - (Bandwidth)
low_pass_filter = rectpuls(k,(B*2));

im2(1,:) = im1(1,:) .* low_pass_filter;   %% remove noise in frequency domain
im2(2,:) = im1(2,:) .* low_pass_filter;
im2(3,:) = im1(3,:) .* low_pass_filter;
im2(4,:) = im1(4,:) .* low_pass_filter;
```

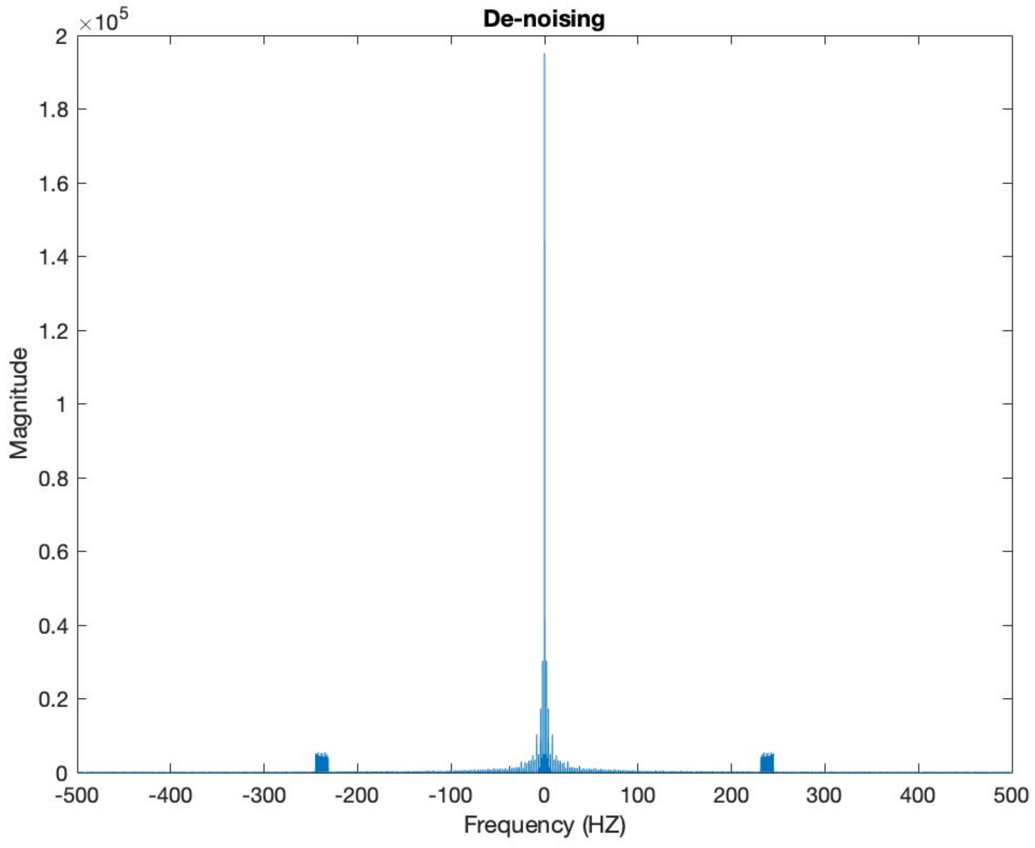


Figure 3.7 - Denoised Signal.

After adjusting the fourier coefficients. Figure 3.7 is the best denoising achievable. There is still a small amount of noise as shown by the small dark blue spot in the middle of the graph. However, with the current system this is the best achievable. The image quality as shown by the next stages is adequate enough to show the landing sites and their numbers.

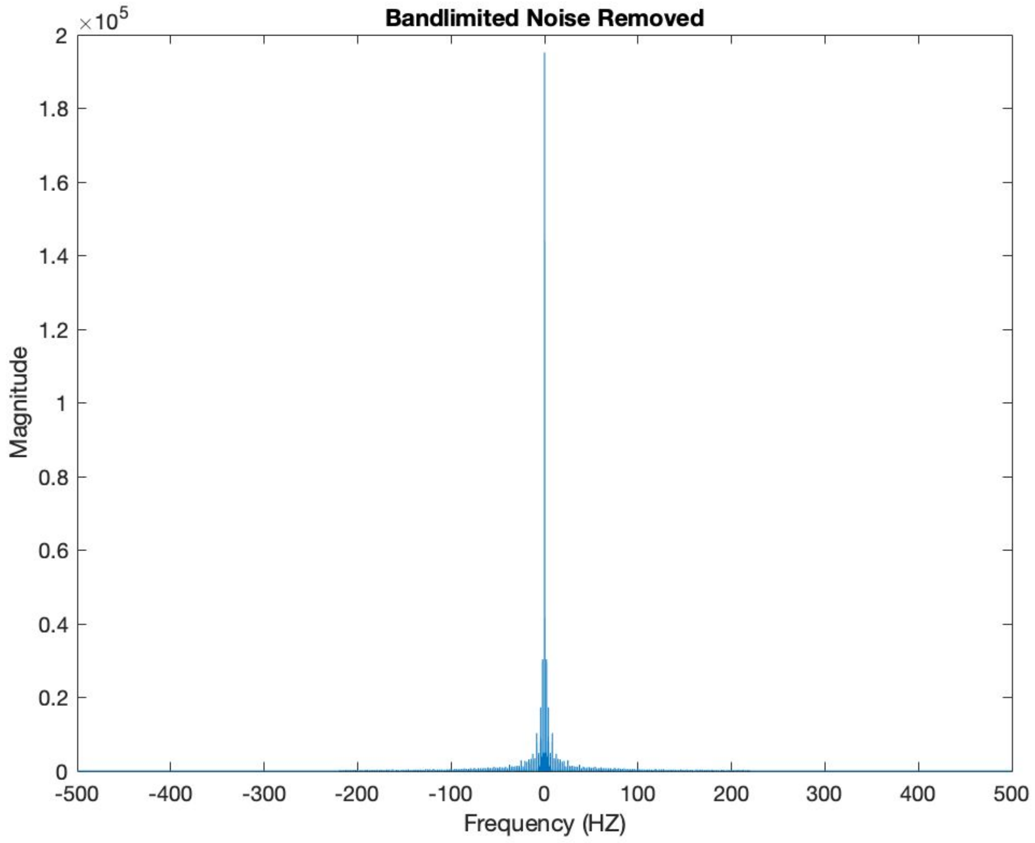


Figure 3.8 - Denoised signal with bandlimited noise removed.

The graph shown in figure 3.8 shows the signal now with the bandlimited noise removed. Now the signal is ready to be brought back to the time domain and viewed. As all the denoising has occurred through the generation of a lowpass filter. This low pass filter was generated with a bandwidth of 220hz. This allows it to function the best whilst keeping all the useful frequency content.

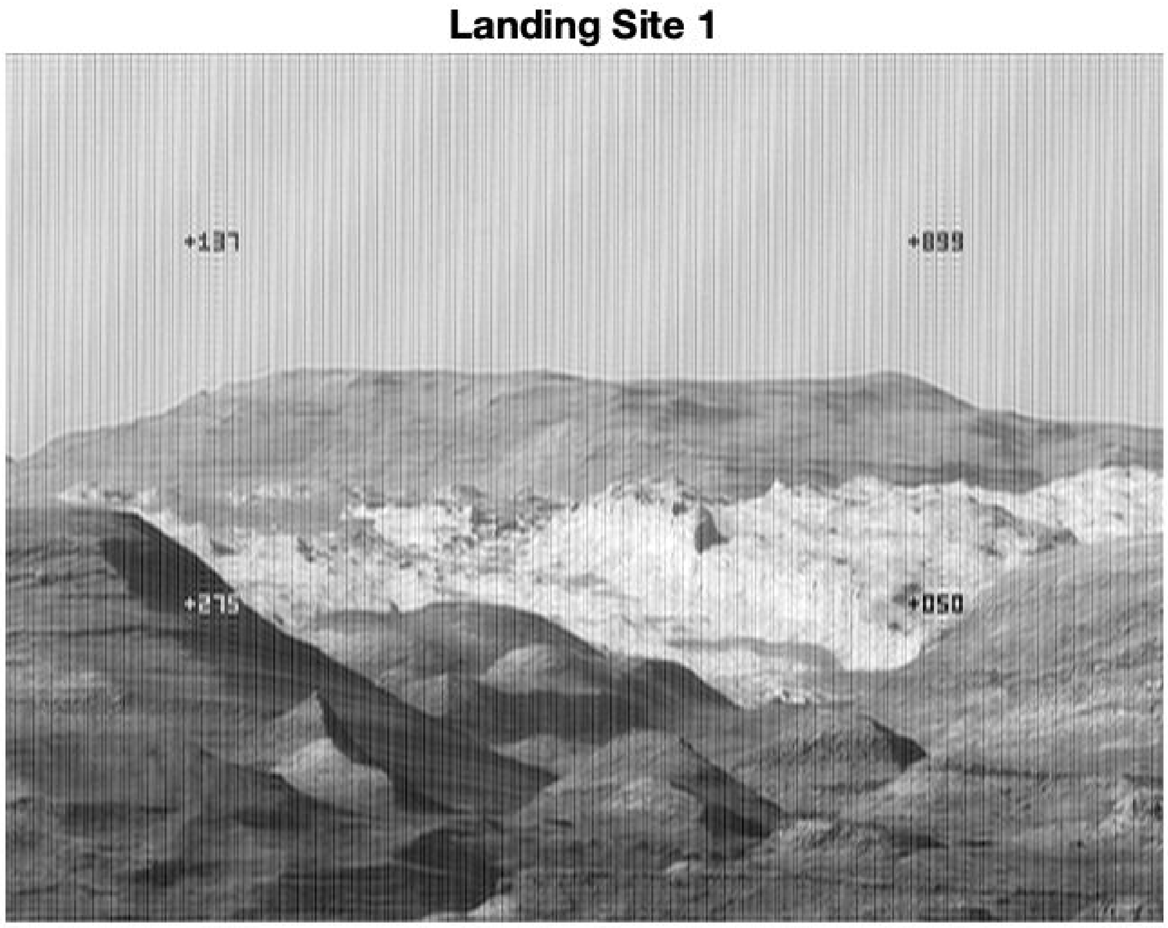


Figure 3.9 - Landing Site 1

Landing Site Numbers

Top Left: 137
Bottom Left: 275
Top Right: 899
Bottom Right: 050

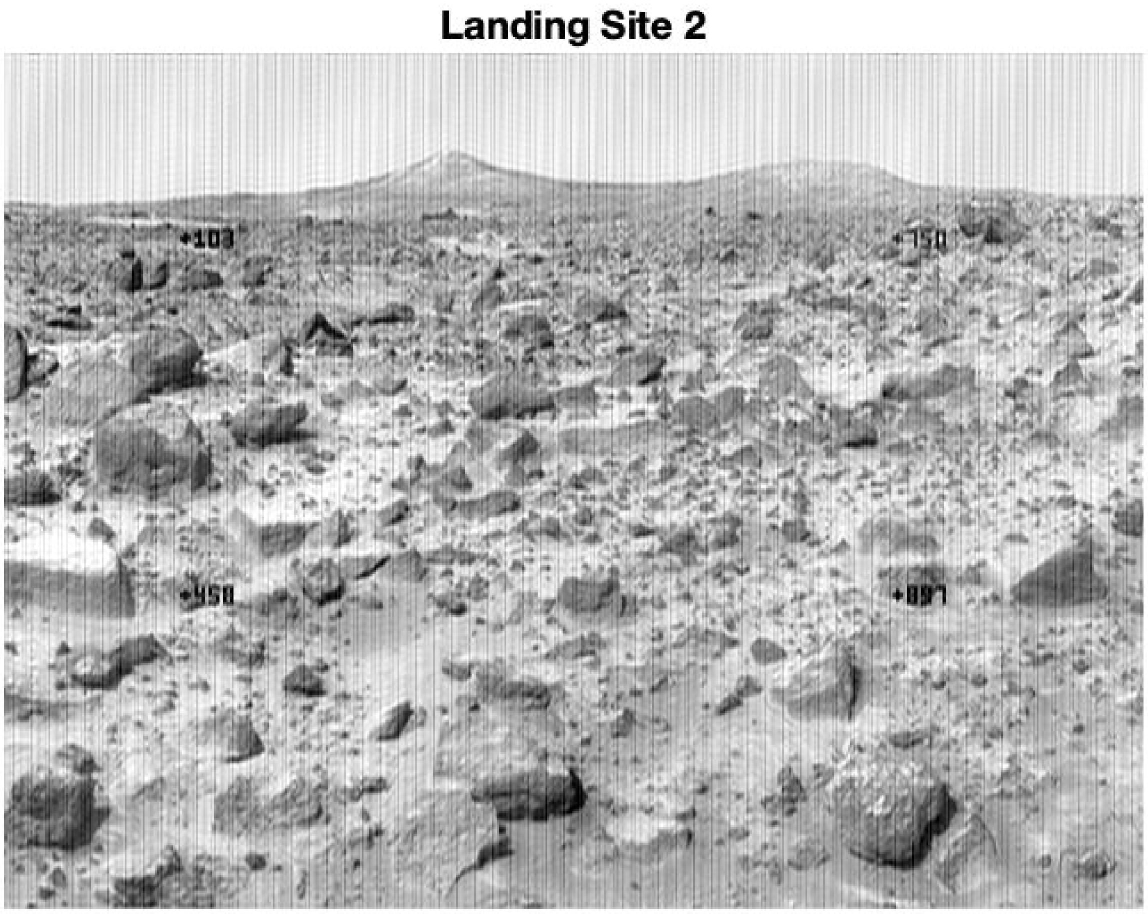


Figure 3.10 - Landing Site 2

<u>Landing Site Numbers</u>

Top Left: 473
Bottom Left: 761
Top Right: 122
Bottom Right: 630

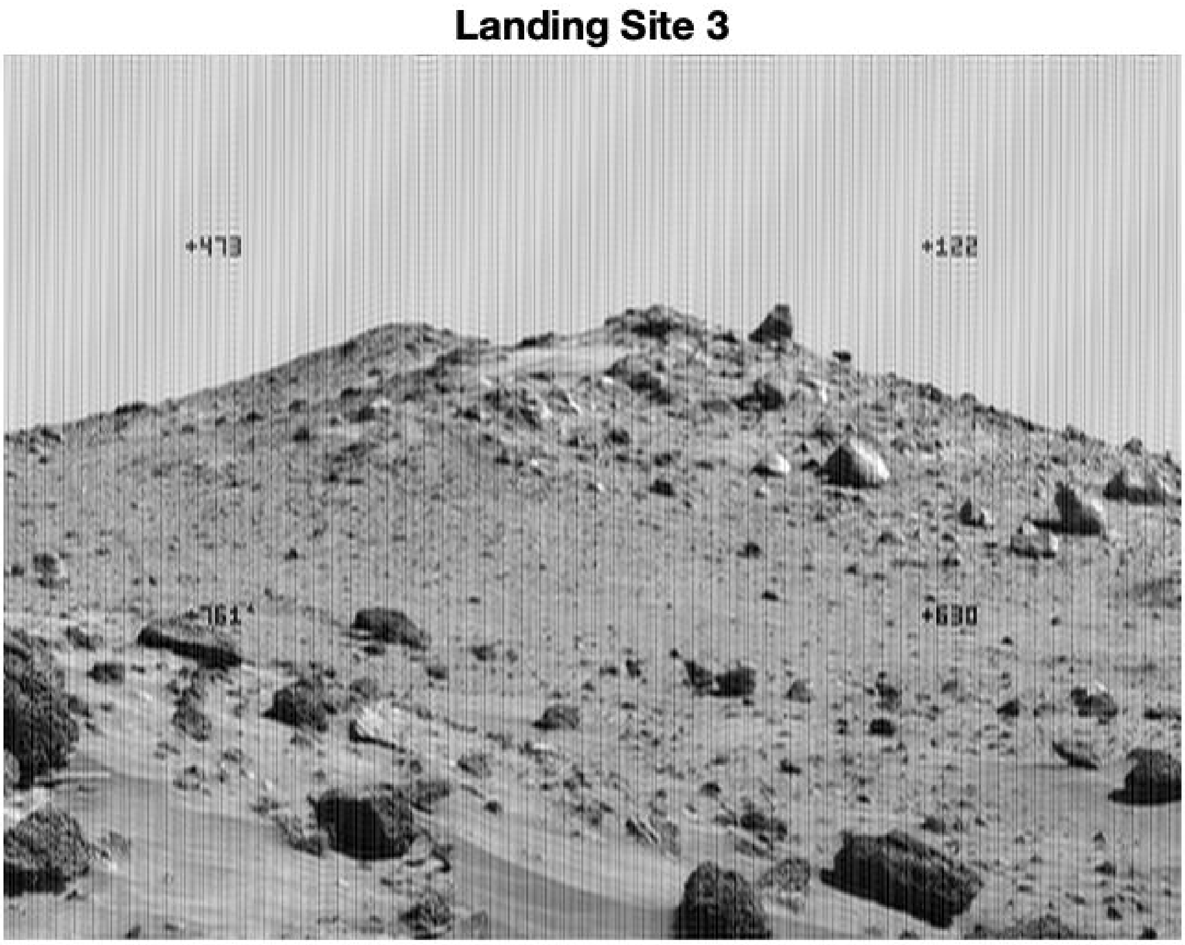


Figure 3.11 - Landing Site 3

<u>Landing Site Numbers</u>
Top Left: 103
Bottom Left: 458
Top Right: 750
Bottom Right: 897

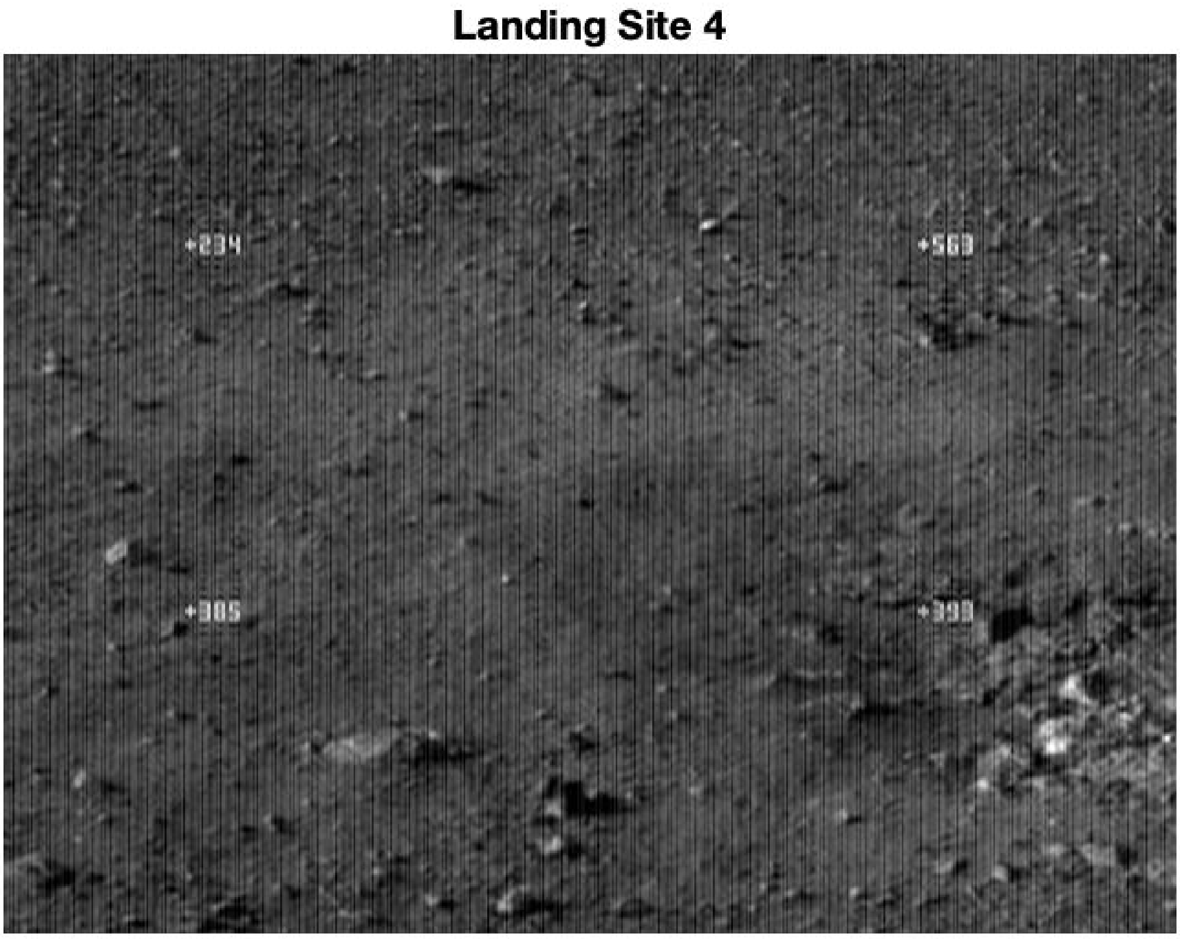


Figure 3.12 - Landing Site 4

Landing Site Numbers
Top Left: 234
Bottom Left: 385
Top Right: 563
Bottom Right: 393

Selection of the Best Landing Site

Out of all the landing sites pictured. It is apparent that landing site 4 is the most appropriate. Its ground overall is quite level and allows for all 4 legs of the craft to easily land and be stable.

Reflection:

With this second assignment being so long it was great to be able to work in a group, with other like minded engineers in order to solve the given problems. A big takeaway for us as a group was communication. Effective communication between us as group mates allowed for the successful completion of this assignment. We all assigned ourselves and each other different sections of each task in order to allow each group member to tackle a different part of the criteria. Sometimes we felt that we could have communicated better to each other so that we did not double up on doing work. Allowing for more efficient competition of tasks.

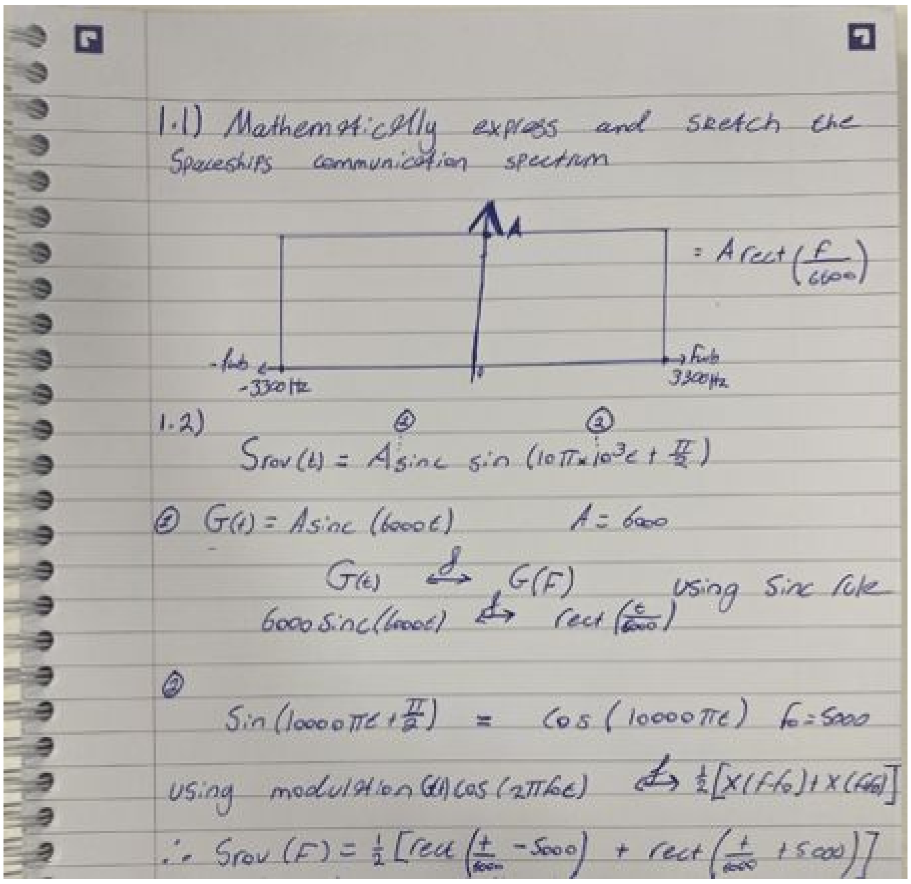
Something that we would do differently next time to avoid the challenges we face is to create an online checklist. This would allow us to ensure we check of all the parts that we complete so that each member can know where they need to start, what the previous member has done and what they should check/proofread.

Conclusion:

To conclude this mission, we have been successfully able to find an appropriate landing site namely ‘Landing Site 4’. As it is the one with the flattest ground out of all the other sites. We have been able to pinpoint the numerical positions pictured to allow the craf to successfully and safely land in the area. Furthermore, to account for the Astronauts' health, we have been able to filter the received EEG signals from their space suits, to allow for monitoring of their health. Allowing analysis of Astronauts 1,2 and 3 to be deemed normal, whilst 4 and 5 have been deemed abnormal. They should be checked up on to see whether they are still fit for the task.

Appendix:

Calculation for Section 1



A
$= A\,\mathrm{rect}\left(\frac{f}{6600}\right)$
$-f_{cb}$
$-3300\,\mathrm{Hz}$
0
f_{cb}
$3300\,\mathrm{Hz}$

1.1) Mathematically express and sketch the Spaceships communication spectrum

1.2)

$$S_{rov}(t) = \overset{①}{A\,\mathrm{sinc}}\ \overset{②}{\sin}\left(10\pi \times 10^3 t + \frac{\pi}{2}\right)$$

① $G(t) = A\,\mathrm{sinc}\,(6000t)$ $\quad A = 6000$

$$G(t) \xleftrightarrow{\mathcal{F}} G(F)$$

$$6000\,\mathrm{Sinc}(6000t) \xleftrightarrow{\mathcal{F}} \mathrm{rect}\left(\frac{f}{6000}\right)$$ using Sinc rule

②

$$\sin\left(10000\pi t + \frac{\pi}{2}\right) = \cos\,(10000\pi t) \quad f_0 = 5000$$

using modulation $G(t)\cos(2\pi f_0 t) \xleftrightarrow{\mathcal{F}} \frac{1}{2}\left[X(f-f_0) + X(f+f_0)\right]$

$$\therefore S_{rov}(F) = \frac{1}{2}\left[\mathrm{rect}\left(\frac{f}{6000} - 5000\right) + \mathrm{rect}\left(\frac{f}{6000} + 5000\right)\right]$$

MATLAB Code Task 1

```
% Section 1
clc;
close all;
clear;
%% 1.1 Spaceships comunication spectrum
fwb = 3300; % frequency given in Hz
faxis = -10000:10:10000; % creating the frequency axis
spectrum = rectangularPulse(-fwb,fwb,faxis);

figure
plot(faxis, spectrum)
title("Spaceship Communication Spectrum")
xlabel("Frequency (Hz)")
ylabel("Magnitude of Frequency")
%% 1.2
% rover transmition function in the frequency domain
sroverf = 1/2*(rectangularPulse(-8000,-2000,faxis) +
rectangularPulse(2000,8000,faxis));

% ploting the rover transmition function
figure
plot(faxis, sroverf)
title("Rover Transmition Function")
xlabel("Frequency (Hz)")
ylabel("Magnitude of Frequency")

%% 1.3 Rover HPF
% creating filter to eliminate the spaceships broadcast information
% between -3300hz & 3300hz
filter = heaviside(-faxis - 3300) + heaviside(faxis - 3300);

% ploting the filter
figure
plot(faxis, filter)
title("H Filter (HPF)")
xlabel("Frequency (Hz)")
ylabel("Magnitude of Frequency")

%% 1.4 Applying HPF
finalF = filter.*sroverf; % aplying the HPF to the rovers transmission
figure % ploting the results of the transfer
plot(faxis, finalF)
title("Final F domain after HPF")
xlabel("Frequency (Hz)")
ylabel("Magnitude of Frequency")
```

MATLAB Code Task 2

```
%% Assignment 2, Part 2 (EEG signal analysis)
%  Do not change before line 35
%  You will need to have generated A2P2Data.mat from
%  GenerateAssignment2Data.m before working with this file.

%  Clearing and preparing the workspace
clear; clc; close all;

%  Load assignment data from A2P1Data.mat.
load('A2P2Data.mat');

%=================================================================%
%
% Refer to the assignment sheet for details.
% Names of the variables are important,
% e.g. 'a1' is considered a different variable to 'A1'.
% Make sure variables have been declared as they appear in the brief.
%
% Ts - sampling period
% t - time domain vector
% MUXSIG - Fourier Transform
% k - frequency vector
% fshift - frequency shifts
% Mag - magnitude
% Phase- phase
% xdm - EEG data
% XDM - Fourier Transform
% freqResponse - frequency response of systems
% impresp - impulse response of chosen system
% EEG - filtered signals
% eeg - time domain of filtered signals
% Conveq - convolution
%
%====Enter your code below this line================================



%% Question 1 - (Represent the Spectrum Analyser)

Ts = 1/fs;  % time step
Ls = length(muxSignal);
t = 0:Ts:Ls*Ts; t(end) = [];    % time vector

figure(1)
plot(t, muxSignal)
title('muxSignal')
ylabel('Magnitude')
xlabel('Time (s)')

k = linspace(-fs/2, fs/2, Ls + 1); k(end) = [];
```

```matlab
MUXSIG = fft(muxSignal);

figure(2)
plot(k, abs(fftshift(MUXSIG)/fs))
title('MUX SHIFT')
xlabel('Frequency (HZ)')
ylabel('Amplitude')

%% Question 2
MUXSIG_shift = abs(fftshift(MUXSIG)/fs);
[x_pos, y_pos] = findpeaks(MUXSIG_shift,'MinPeakHeight',0.09);

fshift = k(y_pos);
fshift = fshift(fshift >= 0);

% find the magnitude spectrum
Mag = abs(x_pos(k(y_pos)>0));

% find the phase spectrum
Phase = fftshift(angle(MUXSIG));
Phase = Phase(y_pos(6:end));

%% Question 3

xdm = FDMDemux(muxSignal.',t,Mag.',fshift,Phase.');

%% Question 4 - (Compute the fourier transform of each data stream)

% Computing the fourier transform
XDM(1,:) = fft(xdm(1,:));
XDM(2,:) = fft(xdm(2,:));
XDM(3,:) = fft(xdm(3,:));
XDM(4,:) = fft(xdm(4,:));
XDM(5,:) = fft(xdm(5,:));

% Calculating the Magnitude Spectrum
XDM_MAG(1,:) = fftshift(abs(XDM(1,:))/fs);
XDM_MAG(2,:) = fftshift(abs(XDM(2,:))/fs);
XDM_MAG(3,:) = fftshift(abs(XDM(3,:))/fs);
XDM_MAG(4,:) = fftshift(abs(XDM(4,:))/fs);
XDM_MAG(5,:) = fftshift(abs(XDM(5,:))/fs);

% Plotting the Magnitude Spectrum
figure(3)
subplot(2,3,1)
plot(k, XDM_MAG(1,:))
title('Magnitude Spectrum')
xlabel('Frequency (HZ)')
ylabel('Amplitude')
subplot(2,3,2)
plot(k, XDM_MAG(2,:))
```

```
title('Magnitude Spectrum')
xlabel('Frequency (HZ)')
ylabel('Amplitude')
subplot(2,3,3)
plot(k, XDM_MAG(3,:))
title('Magnitude Spectrum')
xlabel('Frequency (HZ)')
ylabel('Amplitude')
subplot(2,3,4)
plot(k, XDM_MAG(4,:))
title('Magnitude Spectrum')
xlabel('Frequency (HZ)')
ylabel('Amplitude')
subplot(2,3,5)
plot(k, XDM_MAG(5,:))
title('Magnitude Spectrum')
xlabel('Frequency (HZ)')
ylabel('Amplitude')


%% Question 5 & 6 & 7

% Explanations for Questions 5,6 & 7 will be entierly located in
% the report.

%System_1 = ltiview(sys(1));
%System_2 = ltiview(sys(2));
%System_3 = ltiview(sys(3));
%System_4 = ltiview(sys(4)); % This is the best System

% impulse graphs
figure(4)
subplot(2,2,1)
impulse(sys(1))
title('Impulse Response System 1')
subplot(2,2,2)
impulse(sys(2))
title('Impulse Response System 2')
subplot(2,2,3)
impulse(sys(3))
title('Impulse Response System 3')
subplot(2,2,4)
impulse(sys(4))
title('Impulse Response System 4')

% step graphs
figure(5)
subplot(2,2,1)
step(sys(1))
title('Step Response System 1')
subplot(2,2,2)
```

```matlab
step(sys(2))
title('Step Response System 2')
subplot(2,2,3)
step(sys(3))
title('Step Response System 3')
subplot(2,2,4)
step(sys(4))
title('Step Response System 4')

% bode graphs
figure(6)
subplot(2,2,1)
bode(sys(1))
title('Bode System 1')
subplot(2,2,2)
bode(sys(2))
title('Bode System 2')
subplot(2,2,3)
bode(sys(3))
title('Bode System 3')
subplot(2,2,4)
bode(sys(4))
title('Bode System 4')

% pole zero map graphs
figure(7)
subplot(2,2,1)
pzmap(sys(1))
title('Pole Zero Map System 1')
subplot(2,2,2)
pzmap(sys(2))
title('Pole Zero Map System 2')
subplot(2,2,3)
pzmap(sys(3))
title('Pole Zero Map System 3')
subplot(2,2,4)
pzmap(sys(4))
title('Pole Zero Map System 4')

%% Question 8 - (Filtering the Signals)

system_response = impulse(sys(4),t)';
system_response = fft(system_response)/fs; % Take response to frequency domain

EEG(1,:) = system_response .* XDM(1,:);
EEG(2,:) = system_response .* XDM(2,:);
EEG(3,:) = system_response .* XDM(3,:);
EEG(4,:) = system_response .* XDM(4,:);
EEG(5,:) = system_response .* XDM(5,:);

figure(8)
```

```
subplot(2,3,1)
plot(k, abs(fftshift(EEG(1,:))/fs))
title('Astronaut Signal in Frequency 1')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,2)
plot(k, abs(fftshift(EEG(2,:))/fs))
title('Astronaut Signal in Frequency 2')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,3)
plot(k, abs(fftshift(EEG(3,:))/fs))
title('Astronaut Signal in Frequency 3')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,4)
plot(k, abs(fftshift(EEG(4,:))/fs))
title('Astronaut Signal in Frequency 4')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,5)
plot(k, abs(fftshift(EEG(5,:))/fs))
title('Astronaut Signal in Frequency 5')
ylabel('Magnitude')
xlabel('Frequency (HZ)')

% Take signals to time domain
eeg(1,:) = detrend(real(ifft((EEG(1,:))*fs)));
eeg(2,:) = detrend(real(ifft((EEG(2,:))*fs)));
eeg(3,:) = detrend(real(ifft((EEG(3,:))*fs)));
eeg(4,:) = detrend(real(ifft((EEG(4,:))*fs)));
eeg(5,:) = detrend(real(ifft((EEG(5,:))*fs)));

figure(9)
subplot(2,3,1)
plot(t, eeg(1,:))
title("Astronaut one's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,2)
plot(t, eeg(2,:))
title("Astronaut two's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,3)
plot(t, eeg(3,:))
title("Astronaut three's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,4)
plot(t, eeg(4,:))
```

```
title("Astronaut four's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,5)
plot(t, eeg(5,:))
title("Astronaut five's EEG")
ylabel('Magnitude')
xlabel('Time (s)')

%% Question 9 & 10 - Convolution Stuff
% Questions 10's comparison will be entirely in the report.

impresp = impulse(sys(4),t);
Conveq = conv(impresp, xdm(1,:)/fs);
Conveq = Conveq(1: length(t));

figure(10)
subplot(2,1,1)
plot(t,eeg(1,:))
title('eeg - (signal 1)')
ylabel('Magnitude')
xlabel('Time (s)')

subplot(2,1,2)
plot(t, real(Conveq))
title('Conveq - (signal 1)')
ylabel('Magnitude')
xlabel('Time (s)')

%% Question 11 - (Digital de-noising)

% using a second filter through rectpuls

B = 50;     % cutoff frequency selected
low_pass_filter = rectpuls(k,(B*2));

EEG(1,:) = (fftshift(EEG(1,:))/fs) .* low_pass_filter;
EEG(2,:) = (fftshift(EEG(2,:))/fs) .* low_pass_filter;
EEG(3,:) = (fftshift(EEG(3,:))/fs) .* low_pass_filter;
EEG(4,:) = (fftshift(EEG(4,:))/fs) .* low_pass_filter;
EEG(5,:) = (fftshift(EEG(5,:))/fs) .* low_pass_filter;

figure(11)
subplot(2,3,1)
plot(k, abs(EEG(1,:)))
title('Signal 1')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,2)
plot(k, abs(EEG(2,:)))
title('Signal 2')
```

```
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,3)
plot(k, abs(EEG(3,:)))
title('Signal 3')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,4)
plot(k, abs(EEG(4,:)))
title('Signal 4')
ylabel('Magnitude')
xlabel('Frequency (HZ)')
subplot(2,3,5)
plot(k, abs(EEG(5,:)))
title('Signal 5')
ylabel('Magnitude')
xlabel('Frequency (HZ)')


eeg(1,:) = detrend(real(ifft(ifftshift(EEG(1,:))*fs)));
eeg(2,:) = detrend(real(ifft(ifftshift(EEG(2,:))*fs)));
eeg(3,:) = detrend(real(ifft(ifftshift(EEG(3,:))*fs)));
eeg(4,:) = detrend(real(ifft(ifftshift(EEG(4,:))*fs)));
eeg(5,:) = detrend(real(ifft(ifftshift(EEG(5,:))*fs)));

figure(12)
subplot(2,3,1)
plot(t, eeg(1,:))
title("Astronaut one's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,2)
plot(t, eeg(2,:))
title("Astronaut two's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,3)
plot(t, eeg(3,:))
title("Astronaut three's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,4)
plot(t, eeg(4,:))
title("Astronaut four's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
subplot(2,3,5)
plot(t, eeg(5,:))
title("Astronaut five's EEG")
ylabel('Magnitude')
xlabel('Time (s)')
```

MATLAB Code Task 3

```
%% Assignment 2, Part 3 (Choosing a landing site)
```

```
%  Do not change before line 32
%  You will need to have generated A2P3Data.mat from
%  GenerateAssignment2Data.m before working with this file.

%  Clearing and preparing the workspace
clear; clc; close all;

%  Load assignment data from A2P3Data.mat.
load('A2P3Data.mat');

%=================================================================%
%
% Refer to the assignment sheet for details.
% Names of the variables are important,
% e.g. 'a1' is considered a different variable to 'A1'.
% Make sure variables have been declared as they appear in the brief.
%
% t - time domain vector
% k - frequency vector
% SIG - Fourier Transform
% T - selected value
% Noisesig - estimated noise signal
% a0,an,bn - Trigonometric Fourier Series Coefficients
% OR
% c0,cn - Complex Fourier Series Coefficients
% Noisesig_fs - approximation of noise
% im1 - image 1
% im2 - image 2
%
%====Enter your code below this line================================

%% Question 1 - (View the noisy image)

% displaying the first image - (original re recieved image)
figure(1)
imshow(reshape(sig(1,:) , 480, 640))
title('First Recieved Image')

%% Question 2 & 3 - (Reference Vectors) & (Plotting)

% constructing appropriate time vector.
% t = time vector
% k = frequency vector

Fs = 1000;  % pixels per seconds
Ts = 1/Fs;
Ls = length(sig);   % length of the signal

t = 0:Ts:Ls*Ts; t(end) = [];    % time vector
k = linspace(-500, 500, Ls + 1); k(end) = [];   % frequency vector
```

```
figure(2)
plot(t,(sig(1,:)))
xlim([0 3])
title('Image Signal 1 - Time domain (3 Seconds)')
xlabel('Time (s)')
ylabel('Amplitude')

SIG(1,:) = fft(sig(1,:));   % converting to frequency domain
figure(3)
plot(k, abs(fftshift(SIG(1,:)))/Fs)
title('Image Signal 1 - Frequency domain (3 Seconds)')
xlabel('Frequency (HZ)')
ylabel('Magnitude')

%% Question 4 - (Estimate the Periodic Noise)

% closest option from candidateT
T = candidateT(7)*10^-3;
Noisesig = estimateNoise(sig(1,:), T*Fs).'; % 1 period of the signal

figure(4)
plot(t,sig(1,:))
xlim([0 3])
hold on

plot(t(1:length(repmat(Noisesig, [1 2]))),repmat(Noisesig, [1 2]))
title('Estimation of Periodic Noise')
xlabel('Time (s)')
ylabel('Amplitude')
legend('Signal in Frequency Domain','candidateT Estimation')

%% Question 5 & 6 - (Complex Fourier Series) & (Removing DC Offset)

N = 50;  % Harmonics

nTrig = (1:N).';

a0 = 1/T * sum(sig(1,:)) * Ts;
an = 2/T * sig(1,:) * cos(2*pi*(1/T)*nTrig*t).' * Ts;
bn = 2/T * sig(1,:) * sin(2*pi*(1/T)*nTrig*t).' * Ts;


c0 = 0; % make zero to remove DC offset

cn_pos = 1/2 * (an-1j*bn);
cn_neg = 1/2 * (an + 1j*bn);

cn_neg = fliplr(cn_neg);

cn = [cn_neg c0 cn_pos];
```

```
nComp = (-N:N).';
s_approx = cn * exp(1j * 2 * pi * (1/T) * nComp * t);

%% Question 7 - (Approximation)

Noisesig_fs = s_approx./197;    % generated approximation

%% Question 8 - (Compare the approximation)

% plotting
figure(5)
plot(t(1:length(real(Noisesig_fs))),real(Noisesig_fs))
xlim([0 3])
hold on
plot(t(1:length(repmat(Noisesig, [1 2]))),repmat(Noisesig, [1 2]))

title('Approximation of Periodic Noise')
xlabel('Time (s)')
ylabel('Amplitude')
legend('candidateT Estimation','Complex Approximation')

%% Question 9 - (Denoise)
figure(6)
imshow(reshape(sig(1,:) - real(Noisesig_fs) , 480, 640))    % displays current
image.
title('First Recieved Image First Filter')

% Filtering will be used to further denoise the signal.

im1(1,:) = fft(sig(1,:) - real(Noisesig_fs));
im1(2,:) = fft(sig(2,:) - real(Noisesig_fs));
im1(3,:) = fft(sig(3,:) - real(Noisesig_fs));
im1(4,:) = fft(sig(4,:) - real(Noisesig_fs));

im1(1,:) = fftshift(im1(1,:));
im1(2,:) = fftshift(im1(2,:));
im1(3,:) = fftshift(im1(3,:));
im1(4,:) = fftshift(im1(4,:));

figure(7)
plot(k, abs(im1(1,:)))
title('De-noising')
xlabel('Frequency (HZ)')
ylabel('Magnitude')

B = 220;    % cutoff frequency - (Bandwidth)
low_pass_filter = rectpuls(k,(B*2));

im2(1,:) = im1(1,:) .* low_pass_filter;   %% remove noise in frequency domain
im2(2,:) = im1(2,:) .* low_pass_filter;
im2(3,:) = im1(3,:) .* low_pass_filter;
```

```
im2(4,:) = im1(4,:) .* low_pass_filter;

figure(8)
plot(k, abs(im2(1,:)))
title('Bandlimited Noise Removed')
xlabel('Frequency (HZ)')
ylabel('Magnitude')

%% View the filtered image

% return to time domain
im2(1,:) = real(ifft(ifftshift(im2(1,:))));
im2(2,:) = real(ifft(ifftshift(im2(2,:))));
im2(3,:) = real(ifft(ifftshift(im2(3,:))));
im2(4,:) = real(ifft(ifftshift(im2(4,:))));

figure(9)
subplot(2,2,1)
imshow(reshape(im2(1,:) , 480, 640))
title('Landing Site 1')
subplot(2,2,2)
imshow(reshape(im2(2,:) , 480, 640))
title('Landing Site 2')
subplot(2,2,3)
imshow(reshape(im2(3,:) , 480, 640))
title('Landing Site 3')
subplot(2,2,4)
imshow(reshape(im2(4,:) , 480, 640))
title('Landing Site 4')
```