SEMESTER 2 2018

MARS-242 MISSION CONTROL

EGB242 - ASSIGNMENT 1B

INDIVIDUAL

KEVIN DUONG – N9934731

Introduction

The MARS-242 team is approaching Mars closer and closer every day. The team has been used to the spaceship environment; getting used to eating and sleeping in compact rooms of the spaceship, and they usually spend their time conducting experiments and preliminary analysis. Due to the length of the journey, entertainment for the astronauts is the option to listen to music, or play music and broadcast it along the journey.

This report investigates the use of Fourier transform and its properties in section 1, where handwritten examples of problem solving questions are given. Section 2 introduces Fourier transform in MATLAB, dealing with the signals and their respective Fourier transformed frequencies in section 1, by calculating and plotting them to visually demonstrate the time and frequency domain. Lastly, section 3 introduces a scenario that requires extracting the signals off a raw data by applying the knowledge learnt from Fourier transform and primarily using MATLAB.

B0 - BASA Headquarters: Preparation

The test signals are defined as below:

$$s1(t) = (t-3)e^{-t}u(t-3)$$

$$s2(t) = s1(t) \times \cos(2\pi m t)$$

$$s3(t) = s2(t) \times A\cos(2\pi m t + \phi)$$

Where m, A and ϕ will be assigned values in Sections B2 and B3, but are constants in Section B1.

B1 - BASA Headquarters: Problem Solving

B1.1 – Verifying Fourier transform of s1(t) using first principles

The signal handwritten below uses first principles to find out the Fourier transform of s1(t). No table property is used to help solve the problem.

B1.1

$$s_1(t) = (t-3)e^{-t}u(t-3)$$

$$S_1(f) = \int_{-\infty}^{\infty} s_1(t)e^{-j2\pi ft}\,dt$$

$$= \int_{-\infty}^{\infty} (t-3)e^{-t}u(t-3)e^{-j2\pi ft}\,dt$$

$$u(t-3) = 1 \quad \text{for } t \geq 3$$
$$= 0 \quad \text{for } t < 3$$

$$= \int_{3}^{\infty} (t-3)e^{-t}\cdot e^{-j2\pi ft}\,dt$$

$$= \int_{3}^{\infty} (t-3)e^{-(1+j2\pi f)t}\,dt$$

Integration by parts

$$\int uv' = uv - \int vu'$$

$$= \underbrace{\int_{3}^{\infty} \overset{}{\underset{u}{t}}\,\underset{v'}{e^{-(1+j2\pi f)t}}\,dt}_{LHS} - 3\underbrace{\int_{3}^{\infty} \underset{v'}{e^{-(1+j2\pi f)t}}\,dt}_{RHS}$$

$$\text{Let } u = t \qquad v' = e^{-(1+j2\pi f)t}$$
$$u' = 1 \qquad v = \frac{e^{-(1+j2\pi f)t}}{-(1+j2\pi f)}$$

LHS :

$$\int uv' = uv - \int vu'$$

$$= t \cdot \frac{e^{-(1+j2\pi f)t}}{-(1+j2\pi f)} \boxed{- \int \frac{e^{-(1+j2\pi f)t}}{-(1+j2\pi f)}} \cdot 1 \, dt$$

$$\Rightarrow \frac{1}{(1+j2\pi f)} \cdot \frac{e^{-(1+j2\pi f)t}}{-(1+j2\pi f)} = -\frac{e^{-(1+j2\pi f)t}}{(1+j2\pi f)^2}$$

RHS :

$$= -3 \int e^{-(1+j2\pi f)t} \, dt$$

$$= -3 \cdot \frac{e^{-(1+j2\pi f)t}}{-(1+j2\pi f)} = \frac{3e^{-(1+j2\pi f)t}}{(1+j2\pi f)}$$

Define Limits

$$t \cdot \frac{e^{-(1+j2\pi f)t}}{-(1+j2\pi f)} - \frac{e^{-(1+j2\pi f)t}}{(1+j2\pi f)^2} \Bigg|_3^\infty + \frac{3e^{-(1+j2\pi f)t}}{(1+j2\pi f)} \Bigg|_3^\infty$$

$$= \left[\frac{e^{-(1+j2\pi f)t}}{(1+j2\pi f)} \left\{ -t - \frac{1}{(1+j2\pi f)} + 3 \right\} \right]_3^\infty$$

$$= \left[\frac{e^{-(1+j2\pi f)}}{(1+j2\pi f)^2} \left\{ -t(1+j2\pi f) - 1 + 3(1+j2\pi f) \right\} \right]_3^\infty$$

$$= \frac{e^{-3(1+j2\pi f)}}{(1+j2\pi f)^2} \left\{ \underset{=}{-} \left(\cancelto{0}{-3} \cancelto{0}{- 3j2\pi f} \underset{=}{-1} \cancelto{0}{+3} + \cancelto{0}{3j2\pi f} \right) \right\}$$

$$-(-1) = 1$$

$$\therefore S_1(f) = \frac{e^{-3(1+j2\pi f)}}{(1+j2\pi f)^2}$$

B1.2 – Calculating Fourier transform of s2(t)

The Fourier transform is perfomed on s2(t). The property used for this case is frequency shift.

B1.2

$$S_2(t) = s_1(t) \times \cos(2\pi m t)$$

$$= s_1(t)\left[\frac{e^{j2\pi mt} + e^{-j2\pi mt}}{2}\right]$$

Euler's formula $\frac{e^{j\omega} + e^{-j\omega}}{2}$

$$= \frac{1}{2} s_1(t)\, e^{j2\pi mt} + \frac{1}{2} s_1(t)\, e^{-j2\pi mt}$$

Using frequency shift property   Let $f_0 = m$

$$s_1(t)\, e^{j2\pi f_0 t} \xrightarrow{\mathcal{F}} S_1(f - f_0)$$

$$\therefore S_2(f) = \frac{1}{2} S_1(f-m) + \frac{1}{2} S_1(f+m)$$

$$= \frac{1}{2} \cdot \frac{e^{-3(1+j2\pi(f-m))}}{(1+j2\pi(f-m))^2} + \frac{1}{2} \cdot \frac{e^{-3(1+j2\pi(f+m))}}{(1+j2\pi(f+m))^2}$$

$$\therefore S_2(f) = \frac{e^{-3(1+j2\pi(f-m))}}{2(1+j2\pi(f-m))^2} + \frac{e^{-3(1+j2\pi(f+m))}}{2(1+j2\pi(f+m))^2}$$

B1.3 - Calculating Fourier transform of s3(t)

The s3(t) equation is Fourier transformed with the help of S2f. The property used to calculate S3(f) was frequency shift.

B1.3

$$S_3(t) = S_2(t) \times A\cos(2\pi m t + \phi)$$

$$= S_2(t) \times A\left[\frac{e^{j(2\pi m t + \phi)} + e^{-j(2\pi m t + \phi)}}{2}\right]$$

Eulers Formula

$$\frac{e^{j\phi} + e^{-j\phi}}{2} = \cos(\phi)$$

$$= \frac{A}{2} S_2(t)\left[e^{j2\pi m t} e^{j\phi} + e^{-j2\pi m t} \cdot e^{-j\phi}\right]$$

$$= \frac{A}{2}\left[e^{j\phi} \cdot S_2(t) e^{j2\pi m t} + e^{-j\phi} \cdot S_2(t) e^{-j2\pi m t}\right]$$

Using frequency Shift Property

$$\underset{{}^{*}S_2f}{(f \pm m)} = \underset{S_3f}{(f \pm m \pm m)}$$

$$\therefore S_3(f) = \frac{A}{2}\left[e^{j\phi} S_2(f-m) + e^{-j\phi} S_2(f+m)\right]$$

Simplification

$$S_3(f) = \frac{A}{2}\left[e^{j\phi} \cdot \underbrace{\left(\frac{e^{-3(1+j2\pi(f-m-m))}}{2(1+j2\pi(f-m-m))^2} + \frac{e^{-3(1+j2\pi(f+m-m))}}{2(1+j2\pi(f+m-m))^2}\right)}_{S_2(f-m)} + e^{-j\phi} \cdot \underbrace{\left(\frac{e^{-3(1+j2\pi(f-m+m))}}{2(1+j2\pi(f-m+m))^2} + \frac{e^{-3(1+j2\pi(f+m+m))}}{2(1+j2\pi(f+m+m))^2}\right)}_{S_2(f+m)}\right]$$

$$= \frac{A}{2}\left[e^{j\phi} \cdot \left(\frac{e^{-3(1+j2\pi(f-2m))}}{2(1+j2\pi(f-2m))^2} + \frac{e^{-3(1+j2\pi f)}}{2(1+j2\pi f)^2}\right) + e^{-j\phi} \cdot \left(\frac{e^{-3(1+j2\pi f)}}{2(1+j2\pi f)^2} + \frac{e^{-3(1+j2\pi(f+2m))}}{2(1+j2\pi(f+2m))^2}\right)\right]$$

$$= \frac{A}{2}\left[\frac{e^{-3(1+j2\pi(f-2m)+j\phi)}}{2(1+j2\pi(f-2m))^2} + \boxed{\frac{e^{-3(1+j2\pi f+j\phi)}}{2(1+j2\pi f)^2} + \frac{e^{-3(1+j2\pi f-j\phi)}}{2(1+j2\pi f)^2}} + \frac{e^{-3(1+j2\pi(f+2m)-j\phi)}}{2(1+j2\pi(f+2m))^2}\right]$$

$$\rightarrow \frac{e^{-3(1+j2\pi f+j\phi)}}{2(1+j2\pi f)^2} + \frac{e^{-3(1+j2\pi f-j\phi)}}{2(1+j2\pi f)^2} = \frac{e^{-3(1+j2\pi f-j\phi)} + e^{-3(1+j2\pi f+j\phi)}}{2(1+j2\pi f)^2}$$

Factor $e^{j\phi}$

$$\rightarrow \frac{e^{-3(1+j2\pi f)} \cdot (e^{j\phi} + e^{-j\phi})}{2(1+j2\pi f)^2}$$

let $e^{j\phi} + e^{-j\phi} = 2\cos(\phi)$

Euler's Formula

$$\rightarrow \frac{e^{-3(1+j2\pi f)} \cdot 2\cos(\phi)}{2(1+j2\pi f)^2}$$

Answer:

$$\therefore S_3(f) = \frac{A}{2}\left[\frac{e^{-3\left(1+j2\pi(f-2m)+j\phi\right)}}{2\left(1+j2\pi(f-2m)\right)^2} + \frac{e^{-3\left(1+j2\pi f\right)} \cdot 2\cos(\phi)}{2\left(1+j2\pi f\right)^2} + \frac{e^{-3\left(1+j2\pi(f+2m)-j\phi\right)}}{2\left(1+j2\pi(f+2m)\right)^2} \right]$$

B2 - BASA Headquarters: Training Exercise

The m value is generated to be **10** in MATLAB by running 'GenerateDataAssignment1B.m' file. It is set up as a variable, along with the established amplitude of A = 4, and phi = π.

```matlab
%% Setting up common variables
m = 10;
A = 4;
phi = pi;
```

B2.1 – Constructing S1(f)

A frequency vector of 10000 points with an interval of $-50Hz \leq f < 50Hz$ is created to hold our given Fourier transform of S1(f). These are then are used to plot two subplots of the magnitude and phase of S1(f) using stem and plot respectively (Figure 1.1).

```matlab
% 10000 points
samples = 10000;

% Frequency vector
f1 = linspace(-50,50,samples+1); f1(end) = [];

% Fourier transform S1(f)
S1f = (exp(-3*(1+1i*2*pi*f1))./((1+1i*2*pi*f1).^2);
```

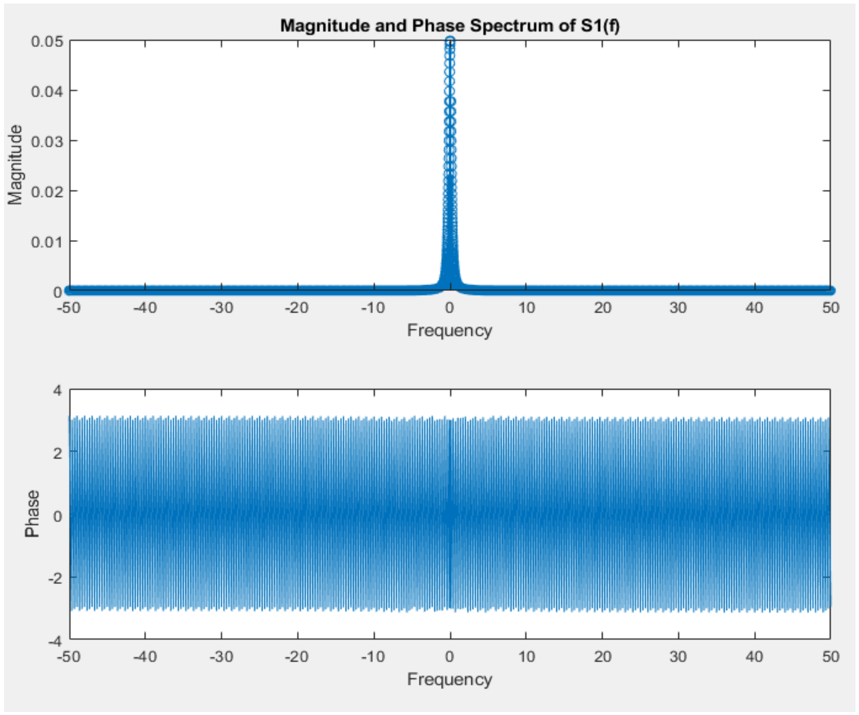
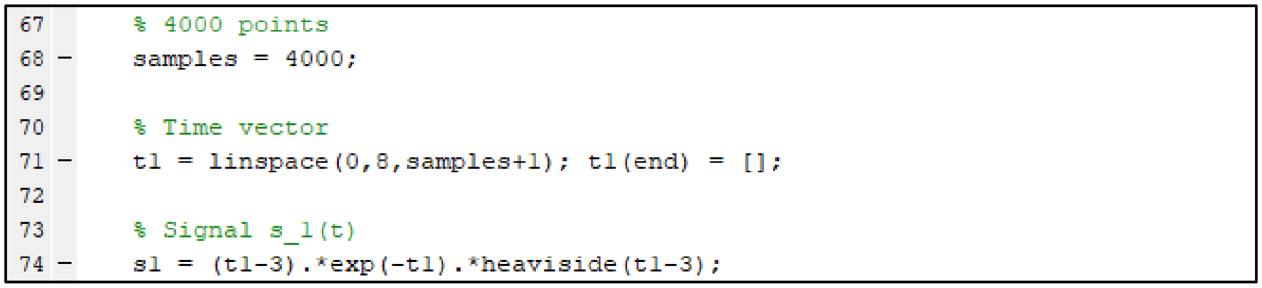


Figure 1.1: S1(f) magnitude and phase spectrum.

B2.2 – Time Vector for s1

Time vector t1 is made to hold our original s1(t) with 4000 samples for an interval of $0 \leq t < 8$. A plot is created to hold this signal against the time domain t1 (Figure 1.2). A heaviside function is applied to s1(t) for u(t-3), when t < 3 is 0 and 1 when t > 3.



```
% 4000 points
samples = 4000;

% Time vector
t1 = linspace(0,8,samples+1); t1(end) = [];

% Signal s_1(t)
s1 = (t1-3).*exp(-t1).*heaviside(t1-3);
```

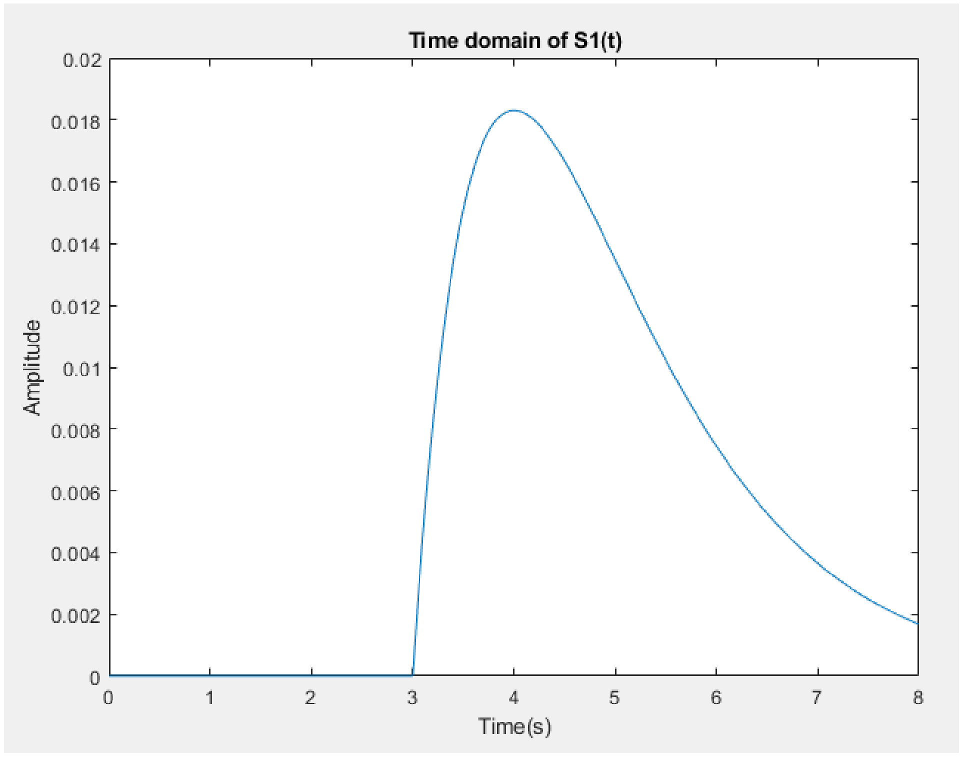
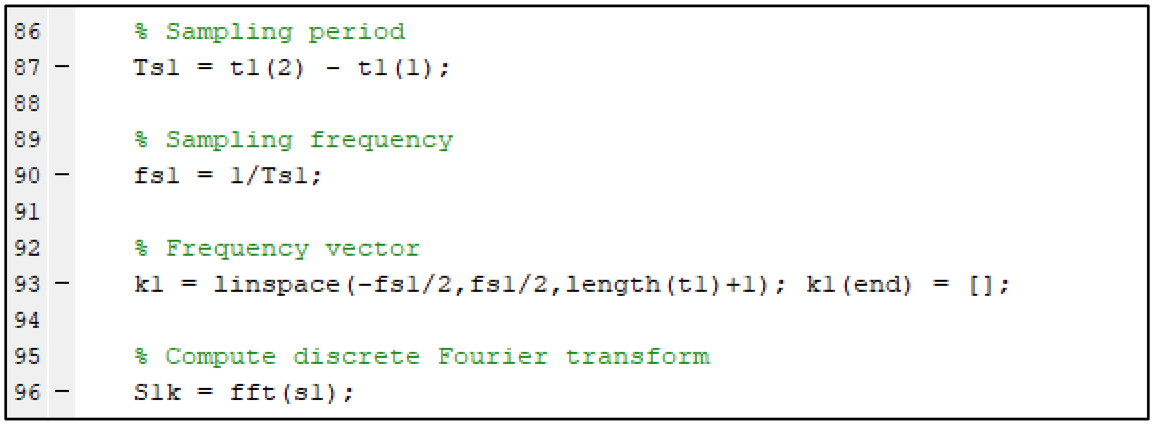


Figure 1.2: s1(t) plot on time domain.

A sampling frequency fs1 is used to calculate the sampling period of Ts1. This was then used to apply on a new frequency vector named k1 for intervals $-\frac{fs1}{2} \leq f < \frac{fs1}{2}$. A new Fourier transform S1k is created using a MATLAB function ‘fft’ onto S1.



```
    % Sampling period
    Ts1 = t1(2) - t1(1);

    % Sampling frequency
    fs1 = 1/Ts1;

    % Frequency vector
    k1 = linspace(-fs1/2,fs1/2,length(t1)+1); k1(end) = [];

    % Compute discrete Fourier transform
    S1k = fft(s1);
```

B2.3 – Plotting S1k

A new figure is created with two subplots to display the magnitude and phase of S1k (Figure 1.3).

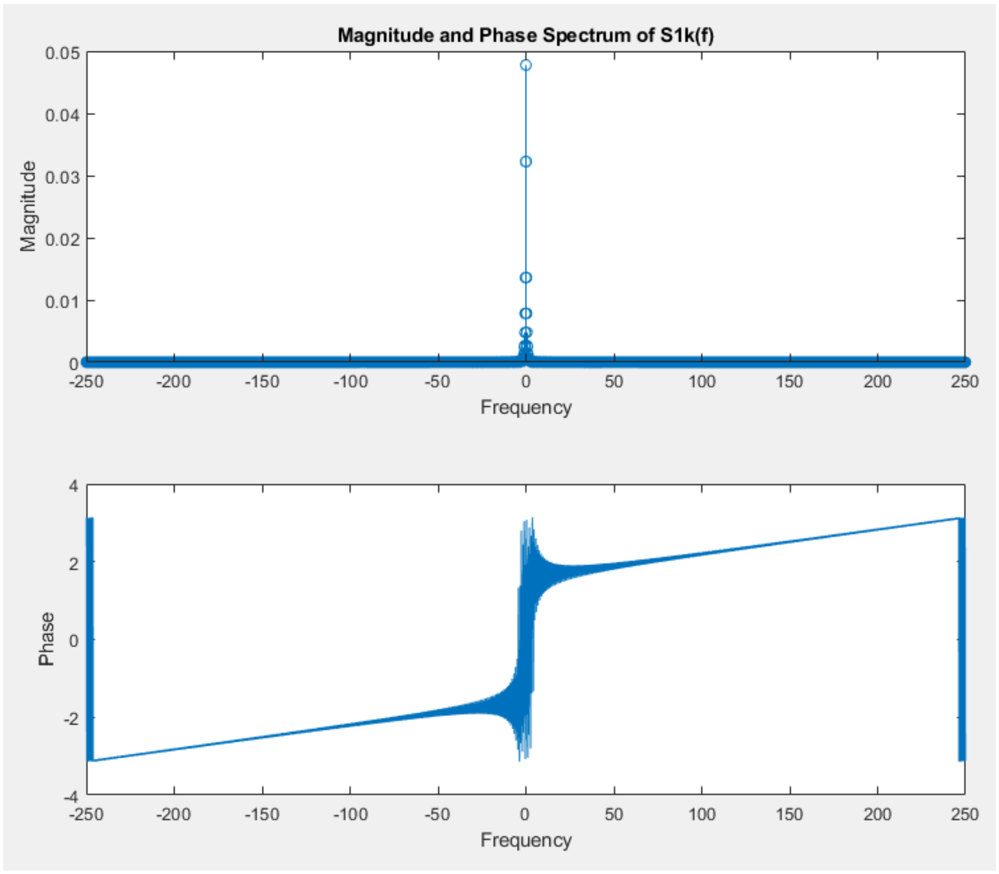
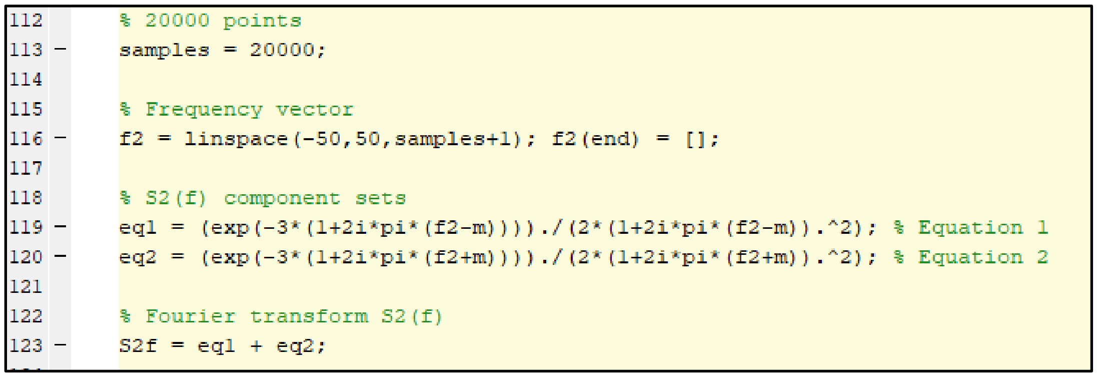


Figure 1.3: S1k(f) magnitude and phase spectrum.

The magnitude spectrum of S1k (MATLAB fft) has less stem plot activity compare to S1f (actual numbers), with an additional mention that S1k's phase plot does not fill up the frequency vector with its phase, only having frequency activity between -50 to 50. This could be due to sampling size, where S1f's frequency vector f1 contains 10000 points and k1 holds only 4000 points. It should also worth mentioning that the frequency vector of S1k has a larger interval with a range of 250 compared to S1f's frequency vector interval of 50.

B2.4 – Constructing S2(f) and S2k

A frequency vector f2 containing 20000 points over the interval $-50Hz \leq f < 50Hz$ has been set up for the Fourier transform S2(f), defined as S2f in MATLAB.



```
% 20000 points
samples = 20000;

% Frequency vector
f2 = linspace(-50,50,samples+1); f2(end) = [];

% S2(f) component sets
eq1 = (exp(-3*(1+2i*pi*(f2-m))))./(2*(1+2i*pi*(f2-m)).^2); % Equation 1
eq2 = (exp(-3*(1+2i*pi*(f2+m))))./(2*(1+2i*pi*(f2+m)).^2); % Equation 2

% Fourier transform S2(f)
S2f = eq1 + eq2;
```

A plot has been made to showcase the magnitude spectrum of S2f (Figure 1.4).

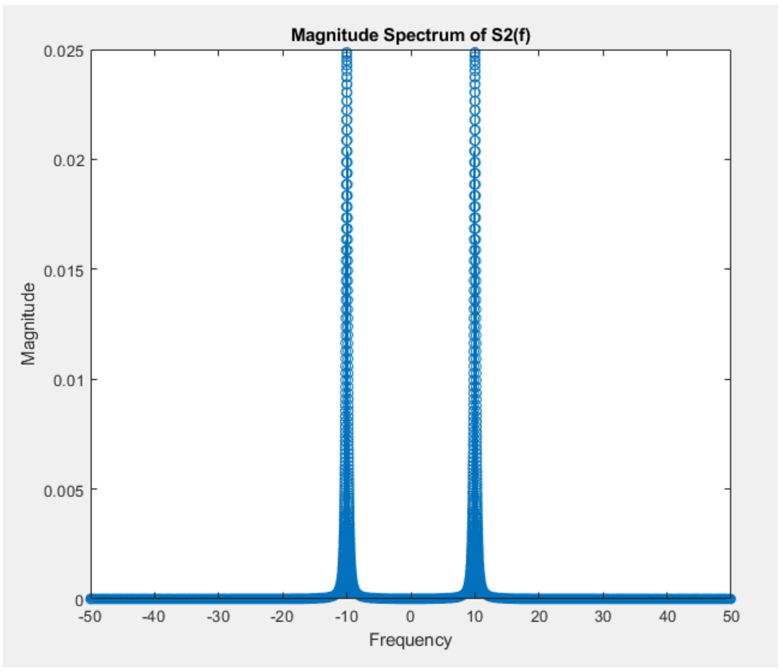
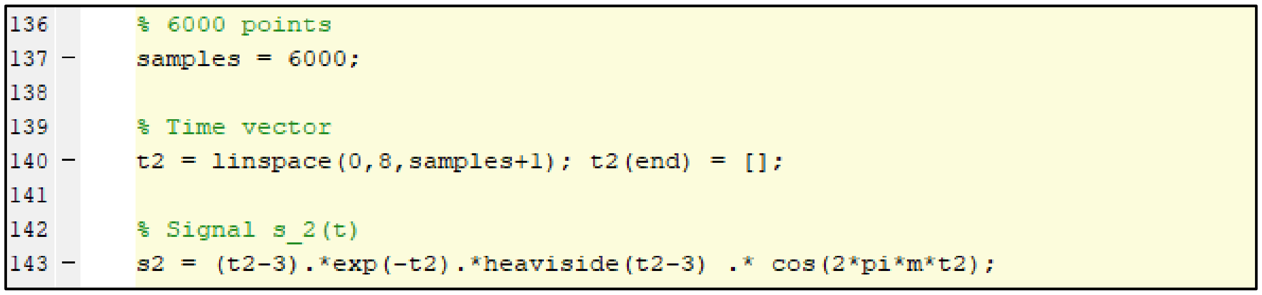


Figure 1.4: Magnitude spectrum of S2(f).

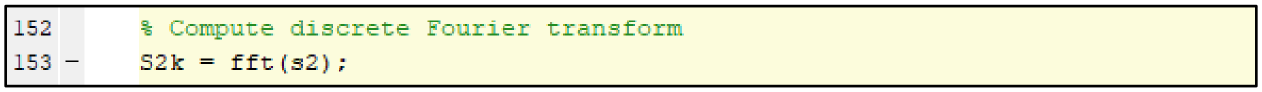
A time vector t2 with 6000 sample points is created to hold our s2(t) for an interval of $0 \leq t < 8$. A plot is created to show this signal's time domain (Figure 1.5). Along with this is a computed fft of s2(t) now defined as S2k.



```
% 6000 points
samples = 6000;

% Time vector
t2 = linspace(0,8,samples+1); t2(end) = [];

% Signal s_2(t)
s2 = (t2-3).*exp(-t2).*heaviside(t2-3) .* cos(2*pi*m*t2);
```



```
% Compute discrete Fourier transform
S2k = fft(s2);
```

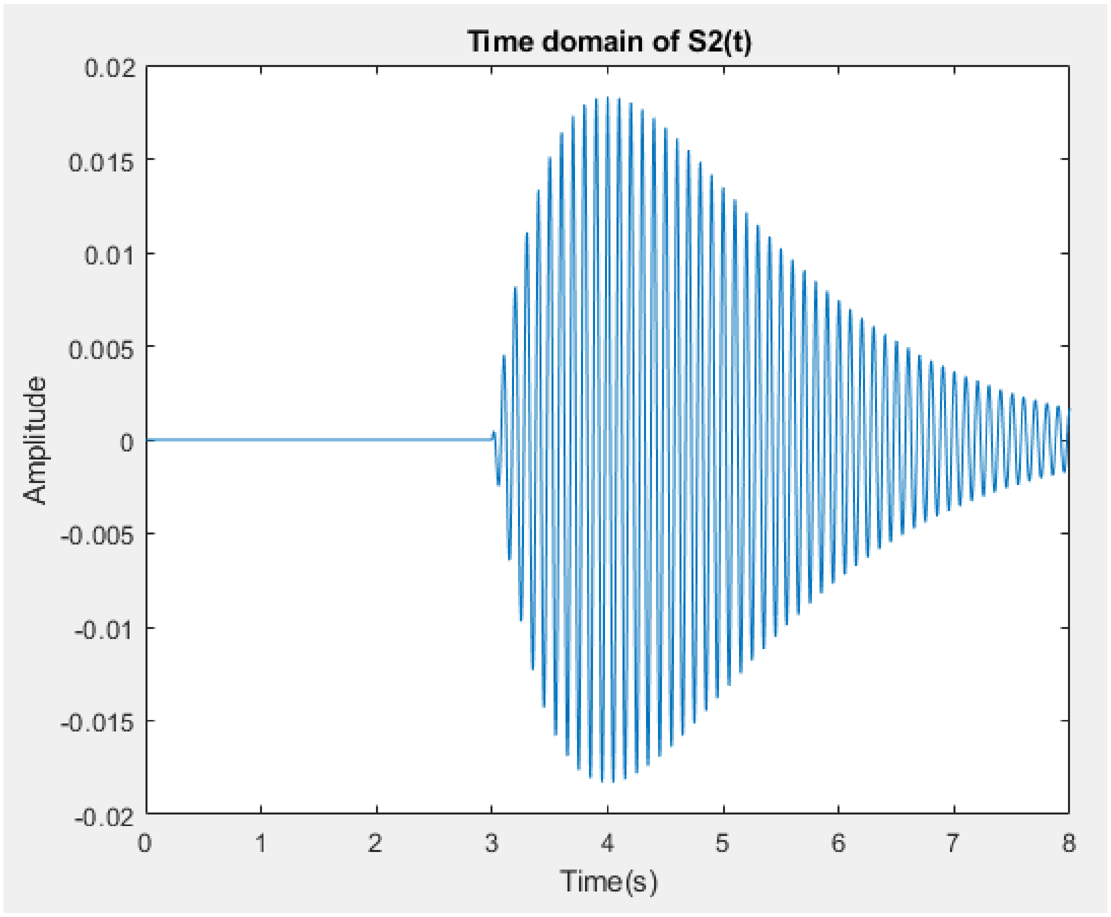
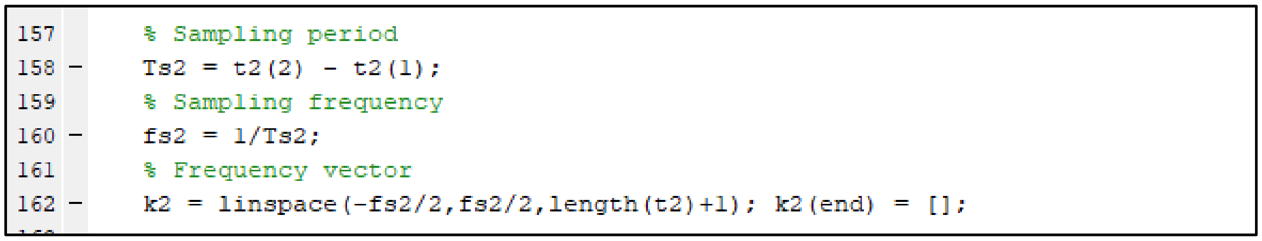


Figure 1.5: s2(t) plot on Time domain.

B2.5 – Developing a New Figure for S2k and S2f

A plot is created to compare both S2k (in red) and S2f (in blue). A custom frequency vector k2 is developed to handle both of these variables to maintain its vector length (Figure 1.6)



```
% Sampling period
Ts2 = t2(2) - t2(1);
% Sampling frequency
fs2 = 1/Ts2;
% Frequency vector
k2 = linspace(-fs2/2,fs2/2,length(t2)+1); k2(end) = [];
```

.

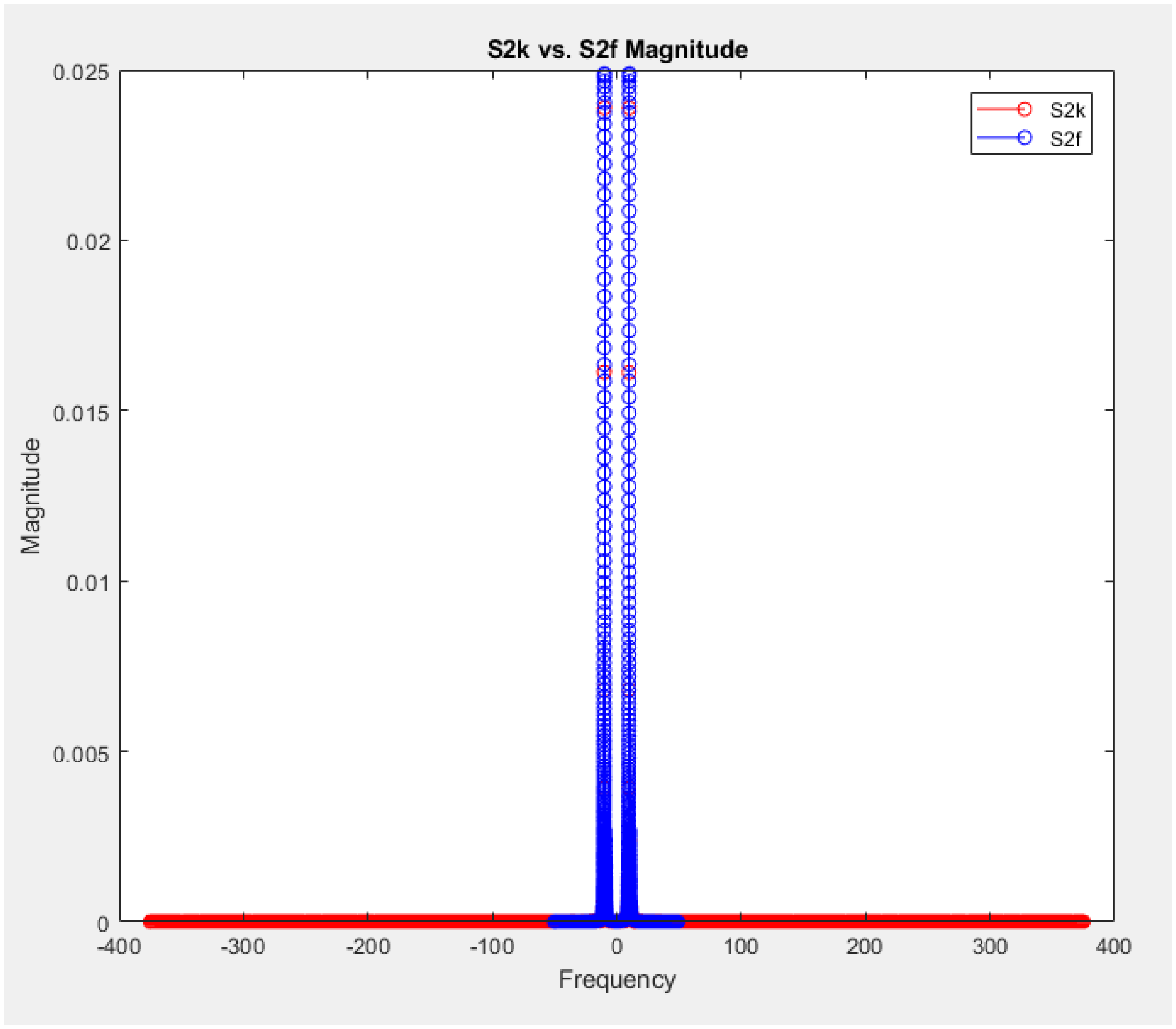
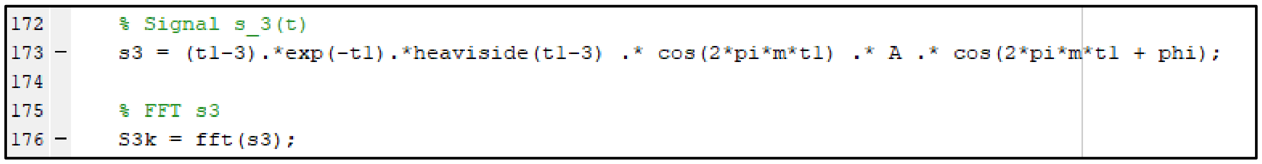


Figure 1.6: S2k and S2f plot in magnitude spectrum.

We can see that S2f, our handwritten Fourier transform version, matches our developed S2k variable that is calculated on MATLAB using fft. The only difference is the length spectrum of the frequency vectors developed for these two Fourier transforms, given that S2f is limited to a range interval of $-50Hz \leq f < 50Hz$, while S2k reaches at an interval of $-\frac{fs1}{2} \leq f < \frac{fs1}{2}$.

B2.6 – Generation of S3

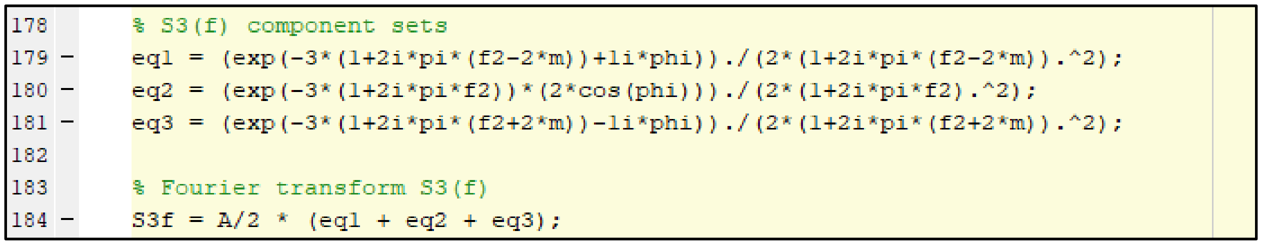
A new signal s3(t) is established and applies the current time vector t1 to be generated. Then a Fourier transform is to be used on s3(t) to generate and be stored as S3k.



```
% Signal s_3(t)
s3 = (t1-3).*exp(-t1).*heaviside(t1-3) .* cos(2*pi*m*t1) .* A .* cos(2*pi*m*t1 + phi);

% FFT s3
S3k = fft(s3);
```

The set up of S3f; it is broken down into parts to simplify the equation and making it easier to use.



```
% S3(f) component sets
eq1 = (exp(-3*(1+2i*pi*(f2-2*m))+1i*phi))./(2*(1+2i*pi*(f2-2*m)).^2);
eq2 = (exp(-3*(1+2i*pi*f2))*(2*cos(phi)))./(2*(1+2i*pi*f2).^2);
eq3 = (exp(-3*(1+2i*pi*(f2+2*m))-1i*phi))./(2*(1+2i*pi*(f2+2*m)).^2);

% Fourier transform S3(f)
S3f = A/2 * (eq1 + eq2 + eq3);
```

The magnitude spectrum is plotted with S3k and also S3f that is supported using frequency vector f2. A comparison of these two is shown below to display the differences (Figure 1.7).

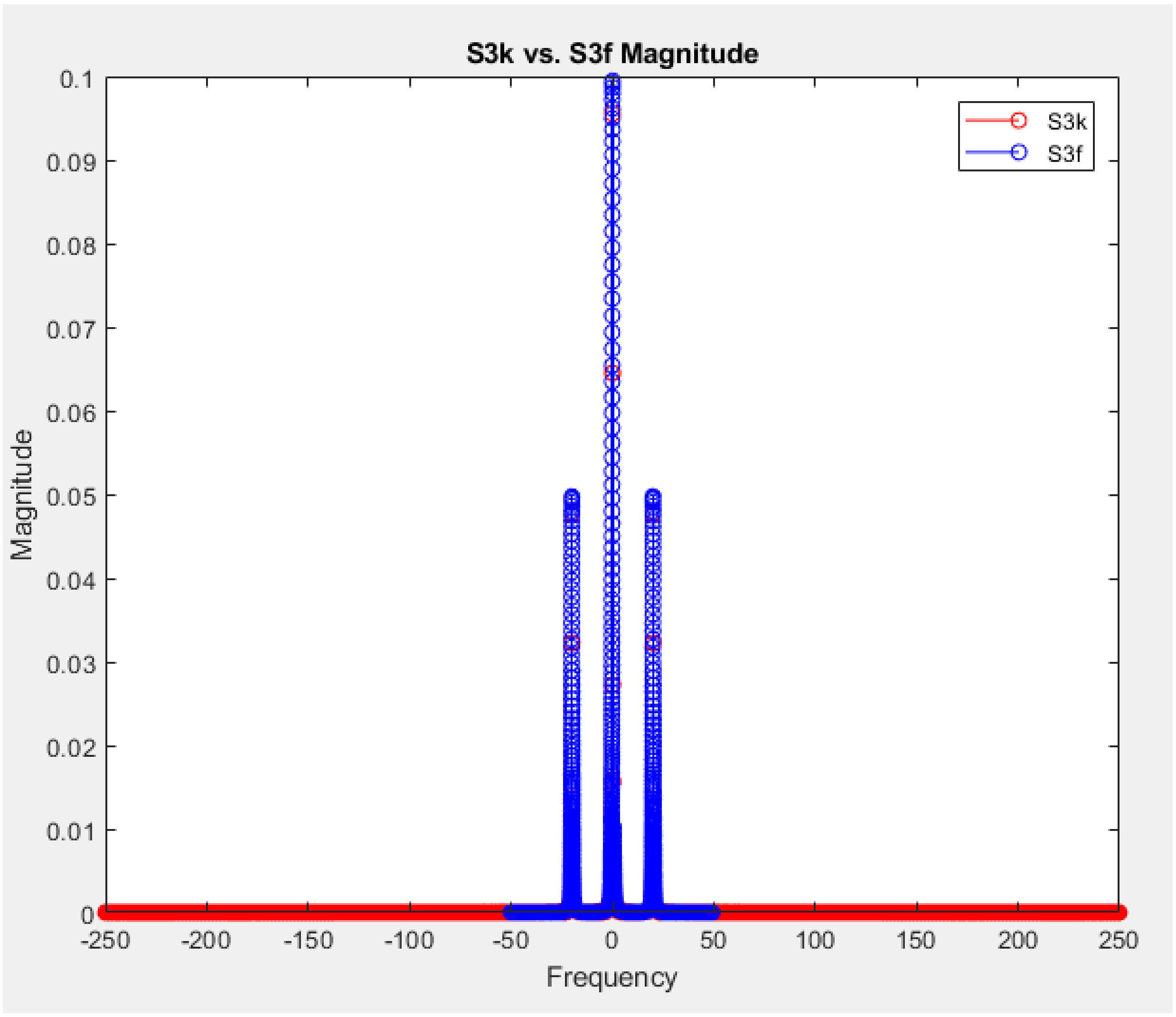


Figure 1.6: S3k and S3f plot in magnitude spectrum.

B2.7 – Similarities and Differences of S1k and S2k

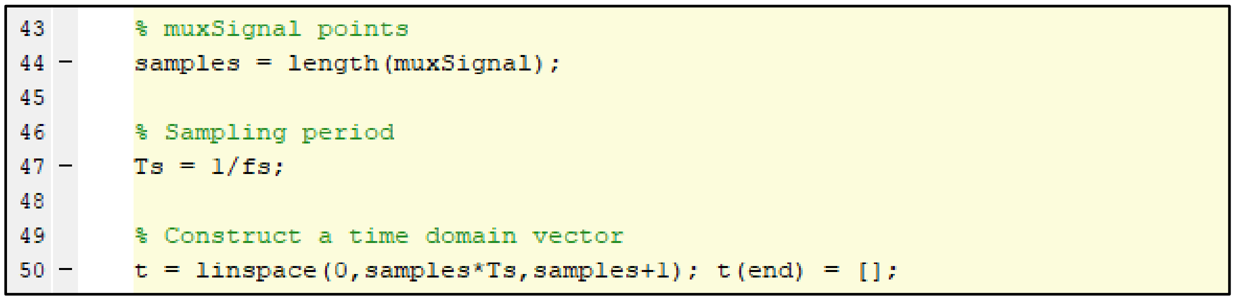
When comparing both of S1k and S2k, there is a difference in the number of magnitude peaks and their respective amplitude height. First of all, S2k contains 2 peaks which reside next to each other and is also approximately half the height that of S1k. The first Fourier transform has only 1 peak, and stands tall at about 0.05 at the centre of the frequency spectrum. This occurs due to applying a cosine after cosine equations to different level of signals – it shifts the frequency both positive and negative.

B3 – Mars Mission: Combining Signals

The hardware module that can remove frequency shifts and extract music and text has malfunctioned. A sampling frequency ‘fs’ and the raw data stream of music and text signals ‘muxSignal’ is provided to solve extract the signals with the use of MATLAB.

B3.1 – Constructing muxSignal

The muxSignal is developed by creating a sampling period Ts and constructing a time domain vector t which holds the raw signal.



```
% muxSignal points
samples = length(muxSignal);

% Sampling period
Ts = 1/fs;

% Construct a time domain vector
t = linspace(0,samples*Ts,samples+1); t(end) = [];
```

A plot is made for muxSignal which corresponds to its time vector t (Figure 2.1).

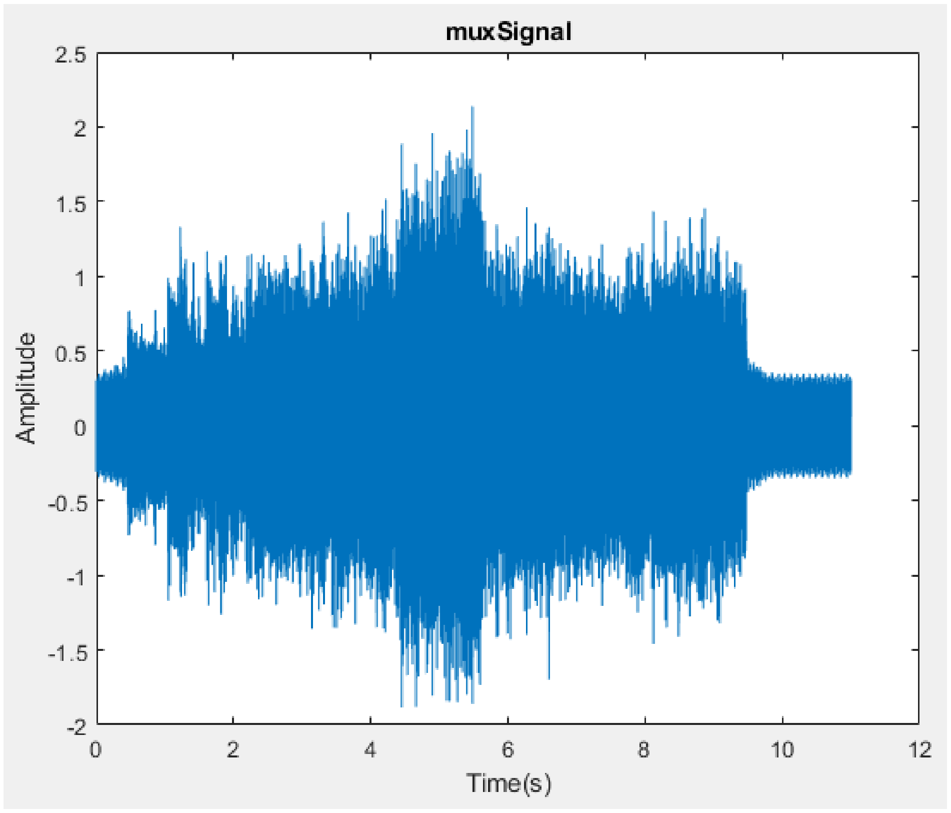
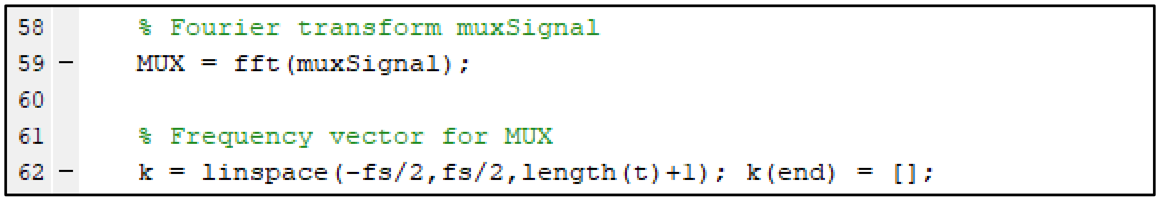


Figure 2.1: Display of raw muxSignal.

B3.2 – Fourier transform of muxSignal

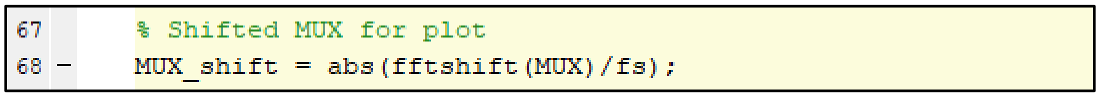
MATLAB computes the Fourier transform of muxSignal using the fft function. Our generated FT is defined to be MUX, and a frequency vector called k is set up to help plot the Fourier transformed version of muxSignal.



```
% Fourier transform muxSignal
MUX = fft(muxSignal);

% Frequency vector for MUX
k = linspace(-fs/2,fs/2,length(t)+1); k(end) = [];
```

To plot MUX appropriately in our plot, we have to define a new variable MUX_shift to perform an ‘abs’ and ‘fftshift’ command on the established MUX variable. We also have to scale our amplitude using the sampling frequency ‘fs’.



```
% Shifted MUX for plot
MUX_shift = abs(fftshift(MUX)/fs);
```

A plot is generated to display the magnitude spectrum of MUX (Figure 2.2).

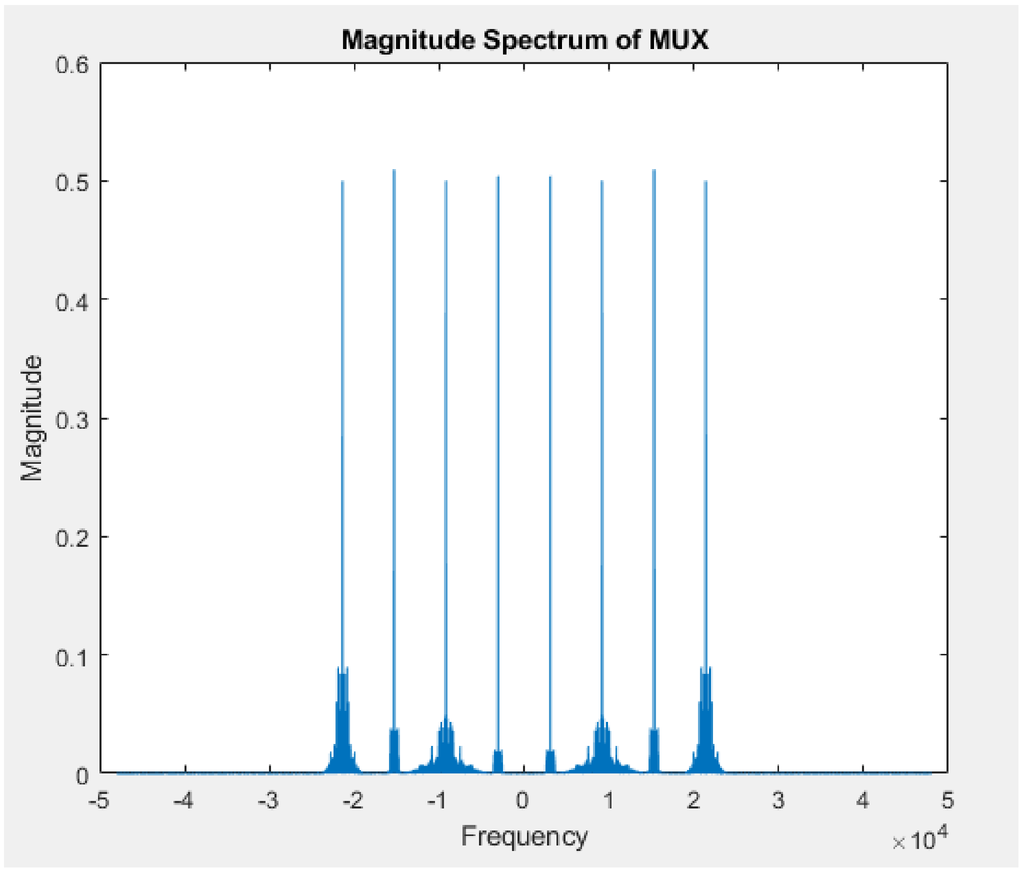
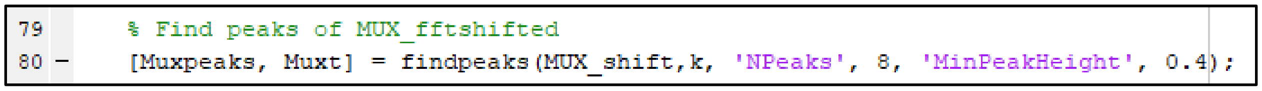


Figure 2.2: The magnitude spectrum of muxSignal.

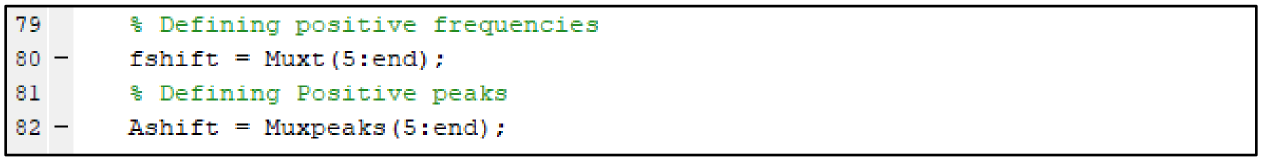
B3.3 – Identifying Peaks

Each peak in the magnitude spectrum provides the locations of the frequency shifts. The peaks are identified using an inbuilt MATLAB function called ‘findpeaks’. The function can give out the frequency and amplitude coordinates which are paired together to show where the peak is. Since there are lots of peaks that aren’t necessary, a condition is set to find peaks above the minimum amplitude.



```
% Find peaks of MUX_fftshifted
[Muxpeaks, Muxt] = findpeaks(MUX_shift,k, 'NPeaks', 8, 'MinPeakHeight', 0.4);
```

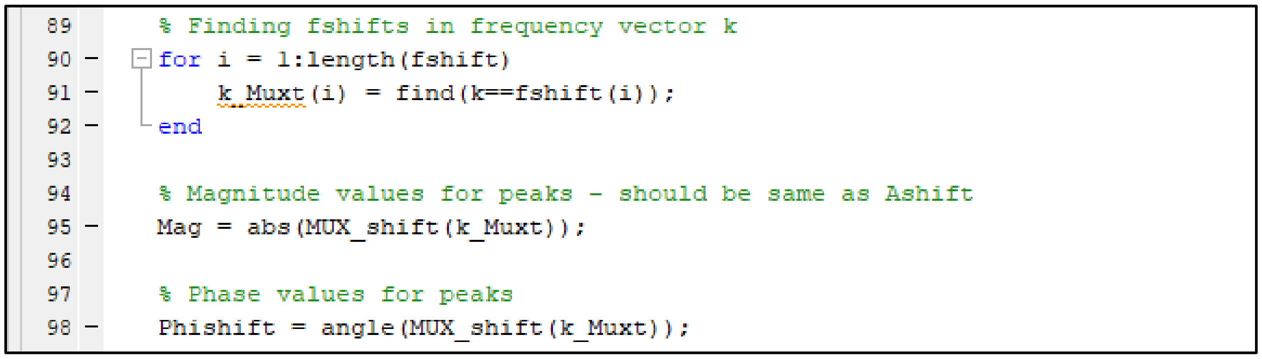
Positive frequency peaks are only needed. Two vectors are created to hold onto these values: fShift for frequencies, and Ashift for peaks. The fshift vector is already established to have whole numbers and has had its elements displayed in an ascending order. Since our MUX shift has 8 peaks, we set the findpeak function to collect 8 elements - the second half of the vector is positive and so fshift stores those values and not negative ones.



```
% Defining positive frequencies
fshift = Muxt(5:end);
% Defining Positive peaks
Ashift = Muxpeaks(5:end);
```

B3.4 – Magnitude and Phase of Peaks

The magnitude and phase of each peak is calculated and are stored in variables Mag and Phishift respectively, which both correspond to their respective frequencies in fshift. The location of these frequency shifts are located in MUX_shift’s frequency vector, k. A for loop is designed to find these values.



```
% Finding fshifts in frequency vector k
for i = 1:length(fshift)
    k_Muxt(i) = find(k==fshift(i));
end

% Magnitude values for peaks - should be same as Ashift
Mag = abs(MUX_shift(k_Muxt));

% Phase values for peaks
Phishift = angle(MUX_shift(k_Muxt));
```

The coordinates of 4 peaks are plotted as green circles which overlay our current MUX magnitude spectrum plot (Figure 2.3).

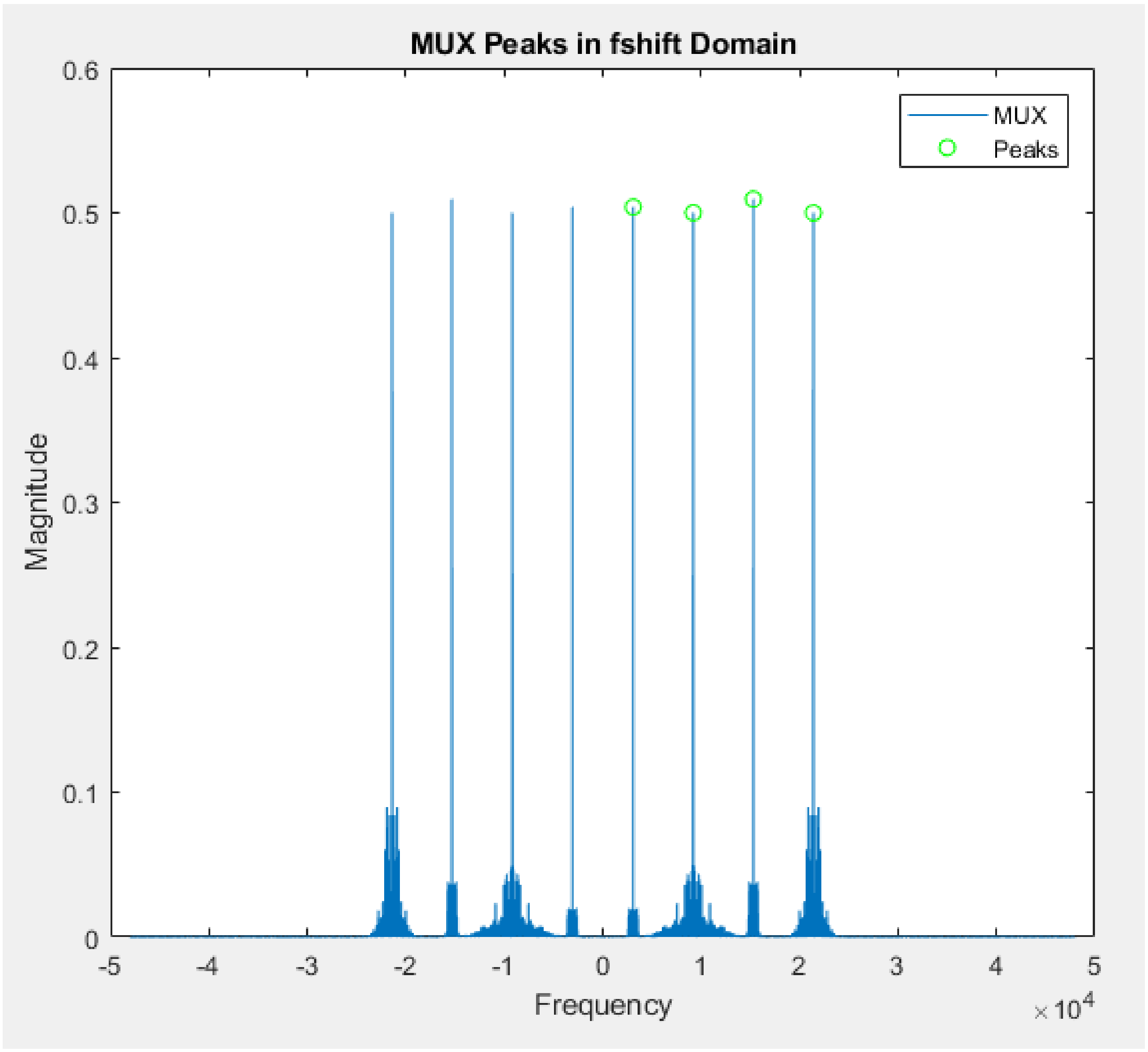


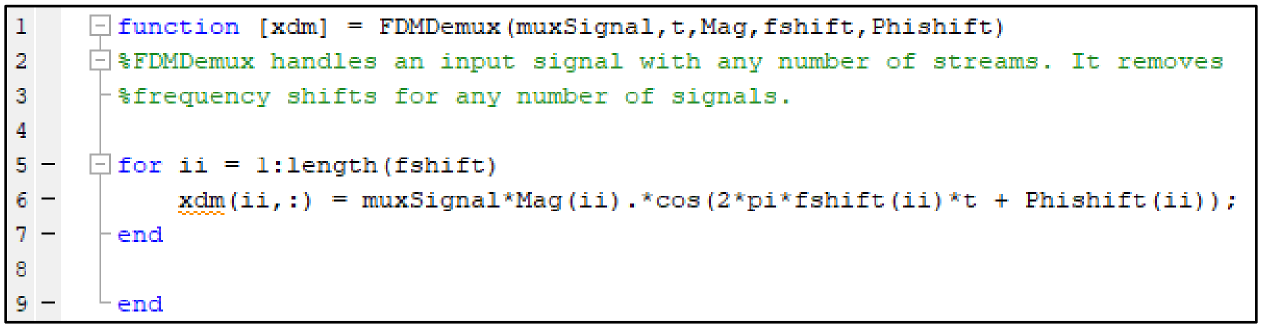
Figure 2.3: Magnitude peaks in the positive frequency, indicated by a green circle.

B3.5 – The FDMDemux function

A frequency shifting module ‘FDMDemux.m’ is developed as a function to handle an input signal with any number of streams. In this case the function is called to remove frequency shifts for 4 signals, and input variables muxSignal, t, Mag, fshift and Phishift are to be inserted as row vectors. When executed, each row of the xdm matrix will contain the data for 1 signal with the frequency shift removed. An equation will be used during the matrix for loop to be able to shift the frequencies based on part of s3(t):

$$A\cos(2\pi mt + \phi)$$

Where A is the peaks multiplied by each of muxSignal element, m is the fshift, t is the vector and ϕ is phishift.

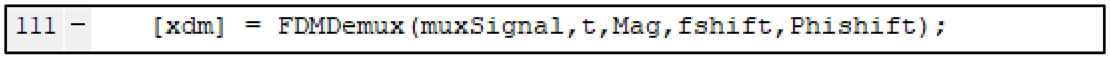


```
function [xdm] = FDMDemux(muxSignal,t,Mag,fshift,Phishift)
%FDMDemux handles an input signal with any number of streams. It removes
%frequency shifts for any number of signals.

for ii = 1:length(fshift)
    xdm(ii,:) = muxSignal*Mag(ii).*cos(2*pi*fshift(ii)*t + Phishift(ii));
end

end
```

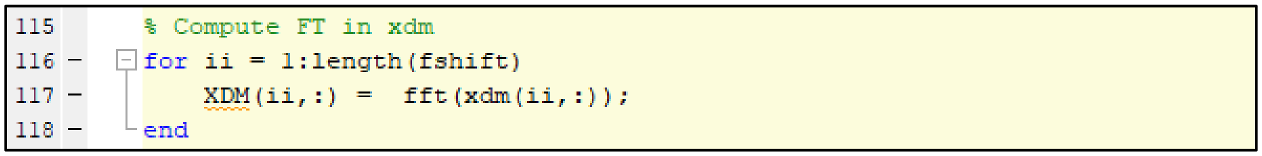
Then, our function is called externally to give us the xdm values.



```
[xdm] = FDMDemux(muxSignal,t,Mag,fshift,Phishift);
```

B3.6 – Fourier transform on data stream

The xdm outputs 4 data streams from our FDMDemux function as a matrix. Each row is then computed with the MATLAB Fourier transform function ‘fft’ using for loop.



```
% Compute FT in xdm
for ii = 1:length(fshift)
    XDM(ii,:) =  fft(xdm(ii,:));
end
```

A plot is coded in to display all 4 data stream magnitude spectrums at once (Figure 2.4).

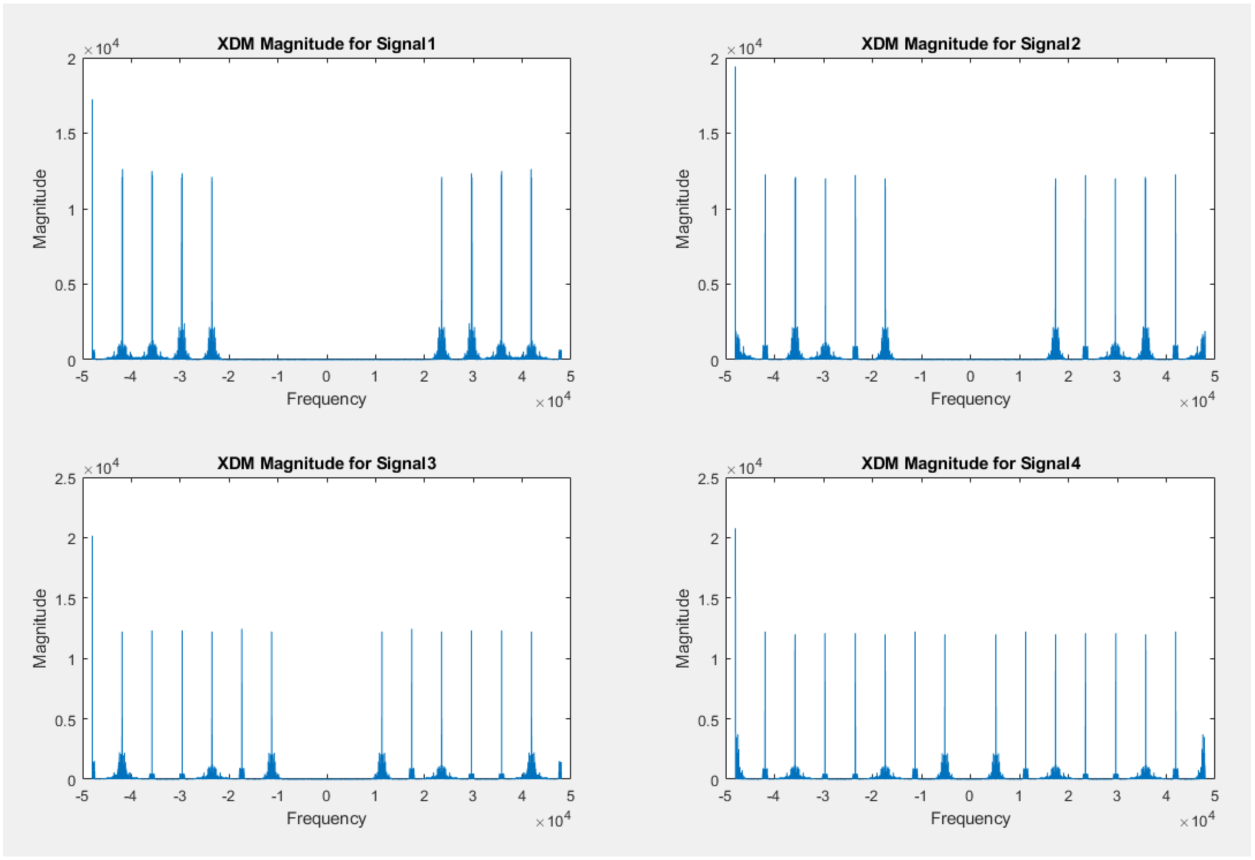
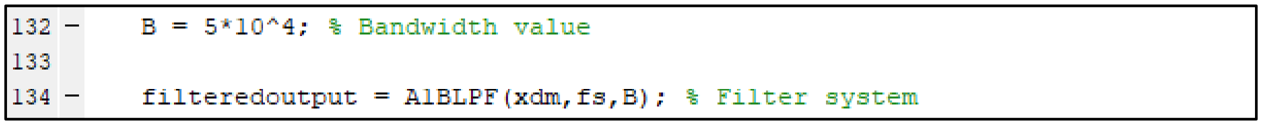


Figure 2.4: Four data streams and their magnitude spectrums.

The streams are unsuitable to listen to or decode in their current state due to a lot of unwanted noise within the signals. A filter is needed to remove the noise in order to receive quality sound and proper decoding.

B3.7 – Determining a bandwidth value

A bandwidth value of $50kHz$ is determined to be used for all streams since they are all placed against the same frequency spectrum with an interval of $-50kHz \leq f < 50kHz$. The A1BLPF.p is used to extract the desired components of each data stream, taking the time domain signal (xdm), sample rate (fs) and single cutoff frequency (B) as inputs.



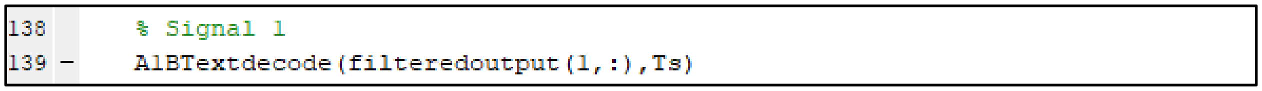
```
    B = 5*10^4; % Bandwidth value

    filteredoutput = A1BLPF(xdm,fs,B); % Filter system
```

B3.8 – Audio stream playback

Each data stream is tested to see whether they output as an audio or text. To test if it's an audio stream, we call MATLAB's function sound() which takes an input of filteredoutput and its chosen row, and the sampling frequency that is fs. For text decoding, an external function `A1BTextdecode` is used which requires the inputs of filteredoutput row and sampling frequency Ts.

<u>SIGNAL 1:</u>

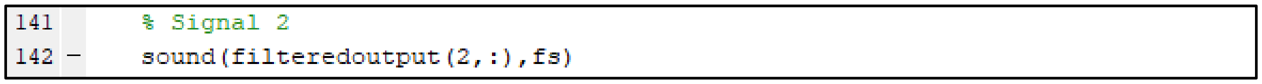


```
% Signal 1
A1BTextdecode(filteredoutput(1,:),Ts)
```

The first signal is a stream of text which reads:

'Young misses their pet llama'.

<u>SIGNAL 2:</u>

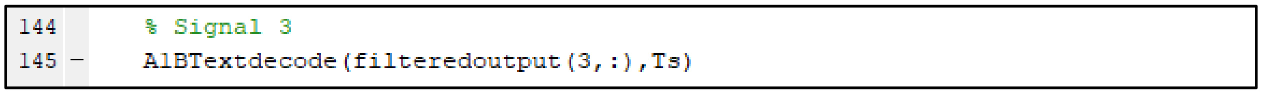


```
% Signal 2
sound(filteredoutput(2,:),fs)
```

This signal played a well-known music called 'Danger Zone' by Kenny Loggins, notably used as a movie theme song for 'Top Gun'[1]. We can hear a sample of the lyrics:

"Highway to the danger zone, gonna take a ride into the danger zone…"

<u>SIGNAL 3:</u>

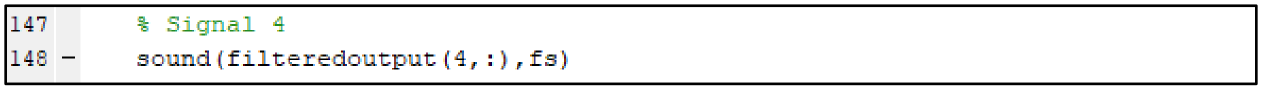


```
% Signal 3
A1BTextdecode(filteredoutput(3,:),Ts)
```

This signal is a text decoded stream that displays the following text:

'Crew members names are : Degen-Portnoy, Lin, Pien, Young and Martinez.'

<u>SIGNAL 4:</u>



```
% Signal 4
sound(filteredoutput(4,:),fs)
```

Our final signal is an audio stream that plays an instrumental music song. The music was composed by Frédéric Chopin titled: 'Piano Sonata No 2 in B flat minor'[2]. Popularly known as a funeral song due to its sad/depressing theme.

A4 – Reflection

The effects of multiplying a time domain signal by a sinusoidal equation leads to shifting the frequencies further and further in both spectrums. This happens when the multiplication of signal with bandwidth B by a cosine or sine function of frequency f_c, shifts the existing spectra to occupy new locations at $f_c \pm B$ and $-f_c \pm B$. This technique is associated with Modulation, and in the time domain, we can see the visual differences between s1(t) and s2(t), where the second signal has added a cosine equation to its main equation. Another cosine is also added to s3(t) in continuation from s2(t), which when viewed on the frequency domain, develops a third shifting frequency. It is 'modulated' by a preceeding signal function when they are multiplied in time. Multiplication in the time domain also corresponds to convolution in the frequency domain. A good example of this being used practically is radio communications; for radio frequencies can propagate long distances. A good quality antenna if it was fed by $100V$, $1MHz$ sine wave, can produce radio waves that can reach to other people's houses, countries, or even other planets. The process of modulation is merging two signals to form a third signal, which then allows progagating at long distances to be able to carry audio and information[3]. Many modulation schemes have been developed to produce better radio communications, and even television broadcasting.

When working on this assignment, I did not really understand or grasp the idea of Fourier transform and how to apply the concept. Further along the way when I started getting through the task and progressing, I started understanding the topic more and more, which helped me learn and appreciate how Fourier transform is beneficial to practical uses. I fairly understood how to use the Foutier Transform properties to help calculate the signals which needed to be transformed; identifying the ideal signal patterns that could help convert from time domain to frequency domain. This further improved my understanding when I input the numbers that I calculated by hand into MATLAB, and then compared it to MATLAB's Fourier transform function which then I made a plot to display the two magnitude spectrums – as soon as I realise they were overlaying each other, I knew this was correct. I also learned more of MATLAB's functions and commands which is great for applying Fourier transform concepts, and this overall made me enjoy my overall experience with MATLAB. If I were to do things differently, I would've study more and ask for help of the content before taking on the assignment, which would've helped me save more time and further increase my likelihood of doing better along the way.

References

[1] "TOP GUN – DANGER ZONE (Music Video)" [Online]. Available: https://www.youtube.com/watch?v=kUsFWO08CO0

[2] "Frédéric Chopin's Piano Sonata No 2 in B flat minor, Funeral March" [Online]. Available: https://www.youtube.com/watch?v=hZY5DBmgC_A

[3] "Multiplying Signals (Amplitude Modulation)" [Online]. Available: https://www.dspguide.com/ch10/5.htm