Lab Report 5

By Seunghyun Park 1003105855 & Juann Jeon 1005210166
Question 1 done by Seunghyun Park
Question 2 done by Seunghyun Park
Question 3 done by Juann Jeon
Lab Report done by Juann Jeon

Q1 a)

Pseudocode

```
import scipy
import matplotlib.pyplot
import numpy

#Assign Mass, in $kg$
$m = 1$
#Assign Spring constant, in $N/m$
$k = 12$
#Assign Speed of light, in $m/s$
$c = speed\ of\ light$
#$x_c$ from Lab03 2c), approximately $8.65 \times 10^7 m$
 $x_c = c\sqrt{\frac{m}{k}}$

#Time for at least 10 periods, with number of samples in power of 2
$t$ = total time
#Compute Time step, needs to be small enough to have stable solution
$dt$ = time step

#define a function that calculates x and dx/dt using Euler-Cromer method
def function(initial x value, time, time step):
      Assign zero list with length of t for $u_1$, $u_1$ = x
      Assign zero list with length of t for $u_2$, $u_2$ = $\dot{u}_1$
      Assign zero list with length of t for $\dot{u}_1$, $\dot{u}_1$ = $u_2$
      Assign zero list with length of t for $\dot{u}_2$, $\dot{u}_2$ = $\frac{-k}{m}u_1(1 - u_2^2/c^2)^{3/2}$
      Assign initial value of $u_1$ in first index, $u_1$ = initial x value
      for index starts from 0 to (length of time - 1)
          #Calculate the position and velocity of relativistic spring
           using Euler-Cromer method
           $\dot{u}_2$ = $\frac{-k}{m}u_1(1 - u_2^2/c^2)^{3/2}$
           $u_2$ = $u_2$+ $\dot{u}_2$*dt
           $\dot{u}_1$ = $u_2$
           $u_1$ = $u_1$ + $\dot{u}_1$*dt
```

```
    return $u_1$, $u_2$

Calculate the position and velocity when x0 = 1m using defined function
above
Calculate the position and velocity when x0 = x_c using defined function
above
Calculate the position and velocity when x0 = 10*x_c using defined function
above

Plot position x when x0 = 1m vs time
Plot position x when x0 = x_c vs time
Plot position x when x0 = 10*x_c vs time

Plot velocity of spring when x0 = 1m vs time
Plot velocity of spring when x0 = x_c vs time
Plot velocity of spring when x0 = 10x_c vs time
```

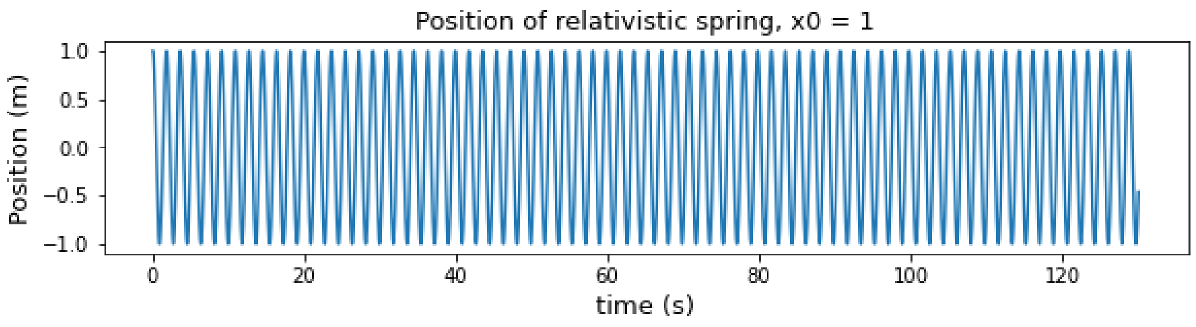

Figure 1: Distance of relativistic spring system with initial position $x_0 = 1m$

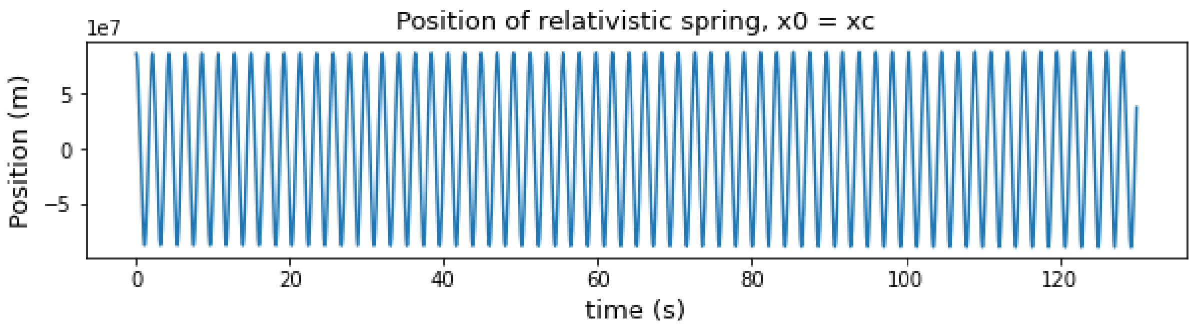

Figure 2: Distance of relativistic spring system with initial position $x_0 = x_c$

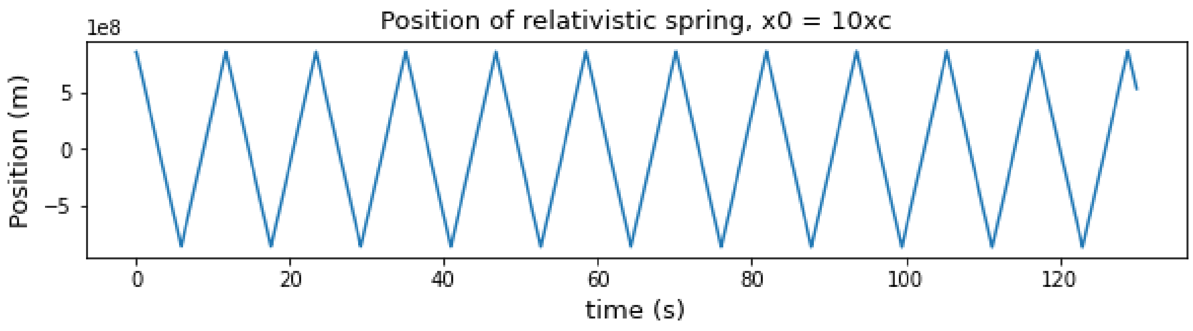

Figure 3: Distance of relativistic spring system with initial position $x_0 = 10x_c$

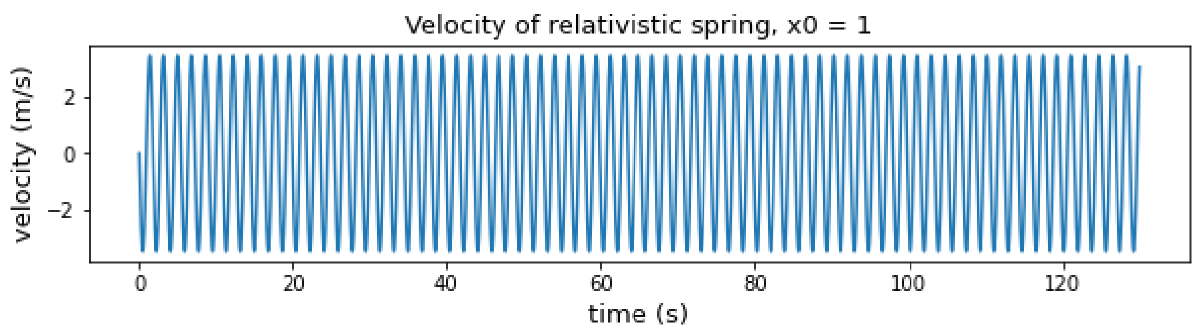

Figure 4: Velocity of relativistic spring system with initial position $x_0 = 1m$

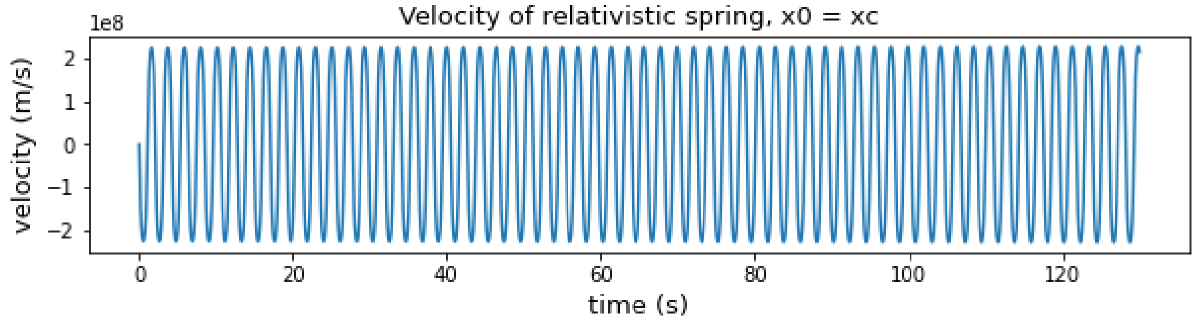

Figure 5: Velocity of relativistic spring system with initial position $x_0 = x_c$

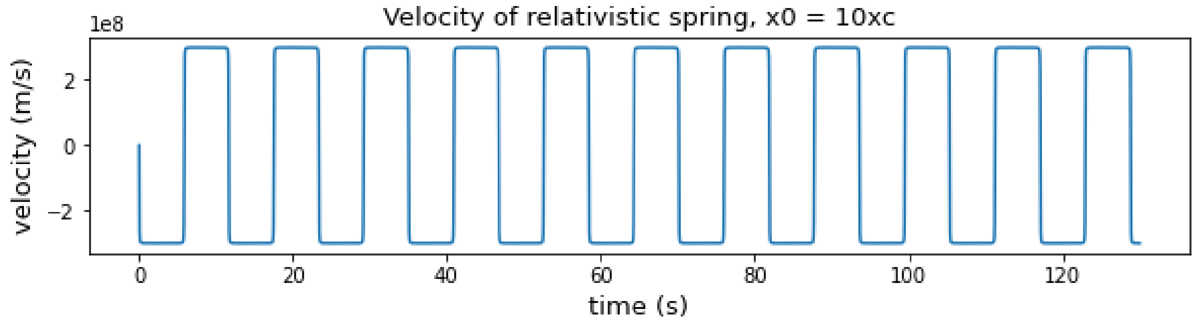

Figure 6: Velocity of relativistic spring system with initial position $x_0 = 10x_c$

To simulate the relativistic spring system from Lab 03, we used following parameters: $u_1 = x$, $u_2 = \dot{x}$, $\dot{u}_1 = u_2$ and $\dot{u}_2 = \frac{-k}{m} u_1 (1 - u_2^2/c^2)^{3/2}$.

Q1 b)

...continued from Q1 a) Pseudocode

```
Apply fourier transform on position, with x0 = 1m
Apply fourier transform on position, with x0 = x_c
Apply fourier transform on position, with x0 = 10*x_c
Find the Frequency corresponding to fourier coefficient

Plot Fourier coefficient of position when x0 = 1m divided by its max vs
frequency
Plot Fourier coefficient of position when x0 = x_c divided by its max vs
frequency
Plot Fourier coefficient of position when x0 = 10*x_c divided by its max vs
frequency
```

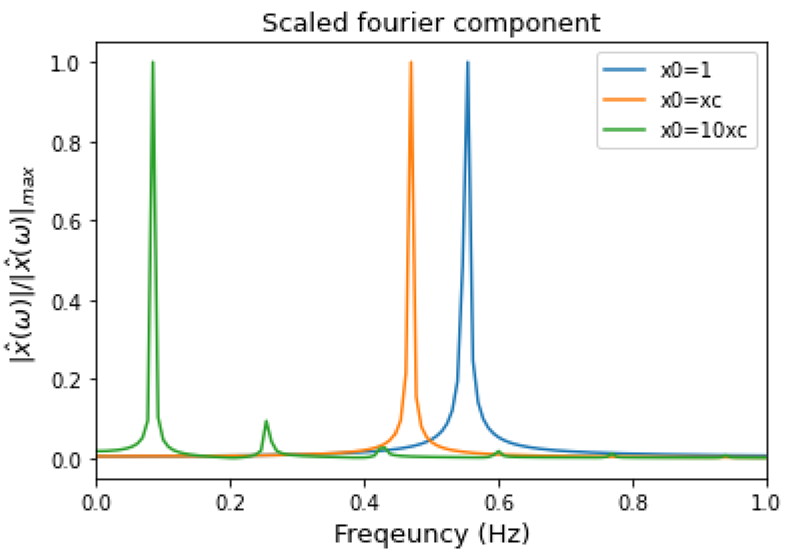

Figure 7: FFT of relativistic spring system with different initial position x_0 scaled by $\frac{|\hat{x}(\omega)|}{|\hat{x}(\omega)|_{max}}$

Figure 7 shows the Fast Fourier Transform of position of relativistic spring scaled by $\frac{|\hat{x}(\omega)|}{|\hat{x}(\omega)|_{max}}$, where $\hat{x}$ is Fourier component and ω is an angular frequency. Notice that for each FFT, peaks are located at different places where lower x_0 has a peak at the right part of the plot due to higher frequency and higher x_0 has a peak at the left part of the plot due to lower frequency as shown from Figures 1, 2, and 3.

Q1 c)

…continued from Q1 b) Pseudocode

```
Import gaussxw

#Define the integrand that was used in Lab03
Define integrand(position, initial position):
      return $\frac{4}{c}\{\frac{2[mc^2+k(x_0^2-x^2)/2]}{k(x_0^2-x^2)[2mc^2+k(x_0-x^2)/2}\}^{1/2}$

Assign Lower boundary of integral
Assign upper boundary of integral
Assign number of sample for Gaussian quadrature

Calculate sample point and weight using gaussxw

Map the sample point and weight to required integration domain
$x_p = \frac{1}{2}(b - a)x + \frac{1}{2}(b + a)$
$w_p = \frac{1}{2}(b - a)w$

#Use for loop to calculate Period of relativistic spring
for i = 0 to N:
      Sum up all components of $w_p[i]$ * $g(x_p[i], b)$
      Output = Period obtained by Gaussian quadrature integration

#Convert the period to frequency
frequency = 1 / Period

Repeat this procedure for $b = x_c$ and $b = 10x_c$

plot frequencies corresponding to $b = x_0$, $b = x_c$ and $b = 10x_c$ with vertical
lines
plot Scaled Fourier coefficient of position with x0 = 1m, x0 = x_c, and x0
= 10x_c divided by its max vs frequency
```

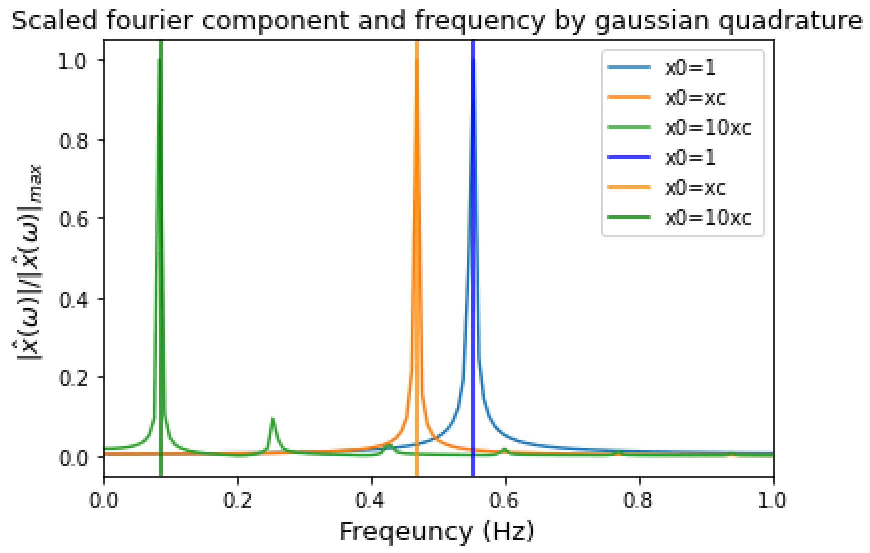

Figure 8: Scaled fourier component and frequencies obtained by gaussian quadrature

Figure 8 shows the frequencies of the oscillations obtained by Gaussian quadrature integration that was used in Lab03 over Figure 7. The plot represents the scaled quantity of FFT and the vertical lines represent the frequencies obtained by gaussian quadrature integration, i.e. $\frac{1}{T}$ where T is a period of the relativistic spring obtained from gaussian quadrature integration. Notice that vertical lines are placed at the peak of scaled FFT, meaning that quantitatively both methods shows the identical frequency to respected initial position x_0.

Q2 b)

Pseudocode

```
Read 'GraviteaTime.wav'
Obtain 'sample' and 'data' from 'GraviteaTime.wav'

#Get channel_0 and channel_1 from 'data'
channel 0 is the first column of 'data'
channel 1 is the second column of 'data'

$N$ = length of channel
dt = time step, equals to 1/sample
$t$ = time #Time starting from 0s to N/sample with time step dt


plot channel_0 in time
plot channel_1 in time
```

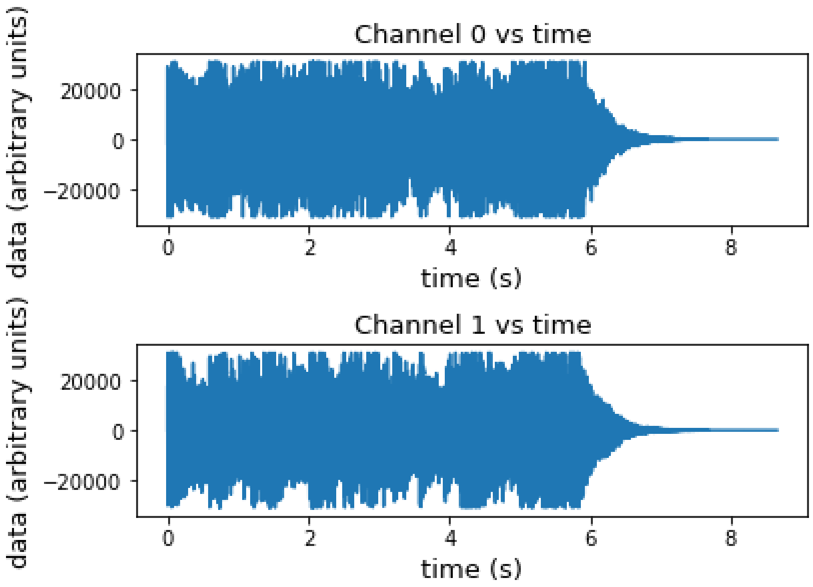

Figure 9: Plots of original time series

Q2 c)

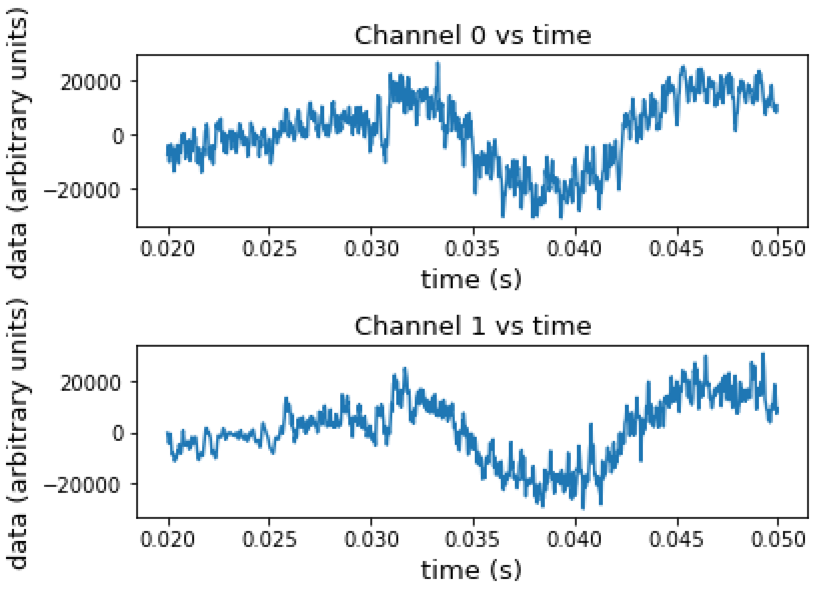

Figure 10: Zoomed version of original time series, time = [20,50] ms

Q2 d)

...continued from Q2 b) Pseudocode

```
#Find fourier coefficient of both channel
Apply fourier transform on channel 0
Apple fourier transform on channel 1
Find the Frequency corresponding to fourier coefficient

#Plot fourier coefficient of both channel
Plot Fourier transform of channel 0 vs Frequency
Plot Fourier transform of channel 1 vs Frequency

#Define a function that sets the fourier coefficients of frequency greater
than 880Hz to zero.
def filter(fourier coefficients, frequency):
     for i = 0 to (length of frequency - 1)
         if absolute value of frequency > 880Hz
             set corresponding fourier coefficient = zero
     return fourier coefficient

#Filter the fourier coefficient
Filter fourier transform of channel 0 using defined filter function
Filter fourier transform of channel 1 using defined filter function

#Convert the filtered fourier coefficient to time domain using inverse
fourier transform
Apply inverse fourier transform on filtered fourier coefficient of channel
0 to convert to time domain
Apply inverse fourier transform on filtered fourier coefficient of channel
1 to convert to time domain
#Plot the filtered fourier coefficient
Plot Filtered Fourier coefficient of channel 0
Plot Filtered Fourier coefficient of channel 1

#Plot the original data
Plot Original time series of channel 0 with t =[0.02,0.05]s
Plot Original time series of channel 1 with t =[0.02,0.05]s

#Plot the filtered data
Plot Filtered time series of channel 0 with t = [0.02,0.05]s
Plot Filtered time series of channel 1 with t = [0.02,0.05]s
```

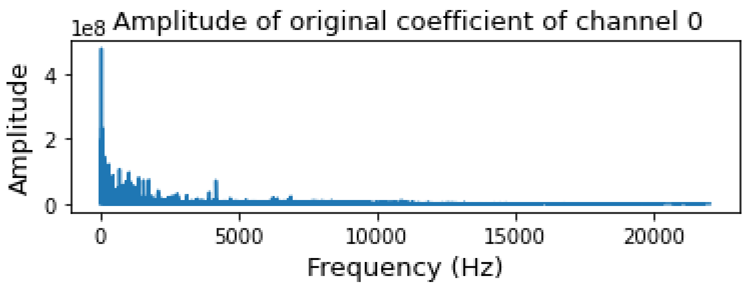
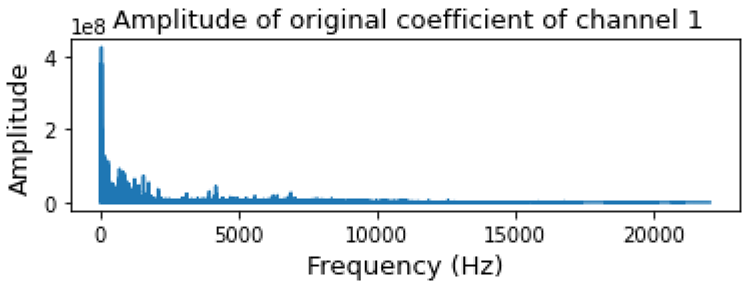

Figure 11: Amplitude of the original Fourier coefficients

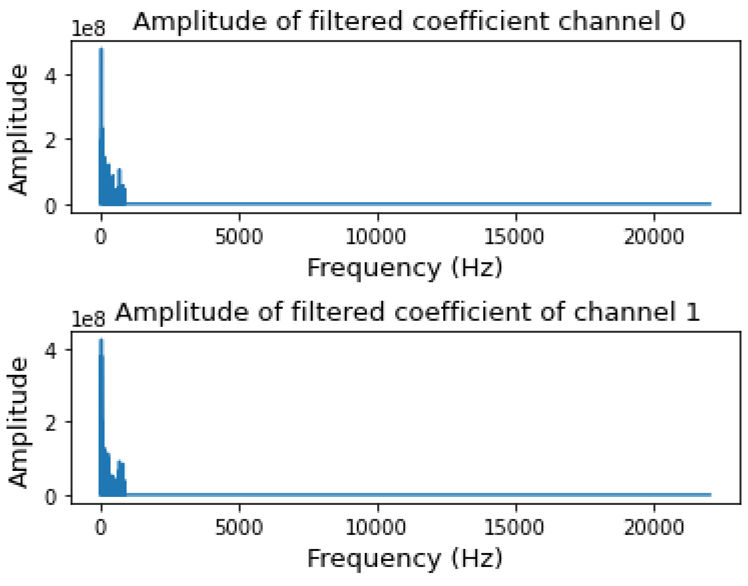

Figure 12: Amplitude of the filtered Fourier coefficients

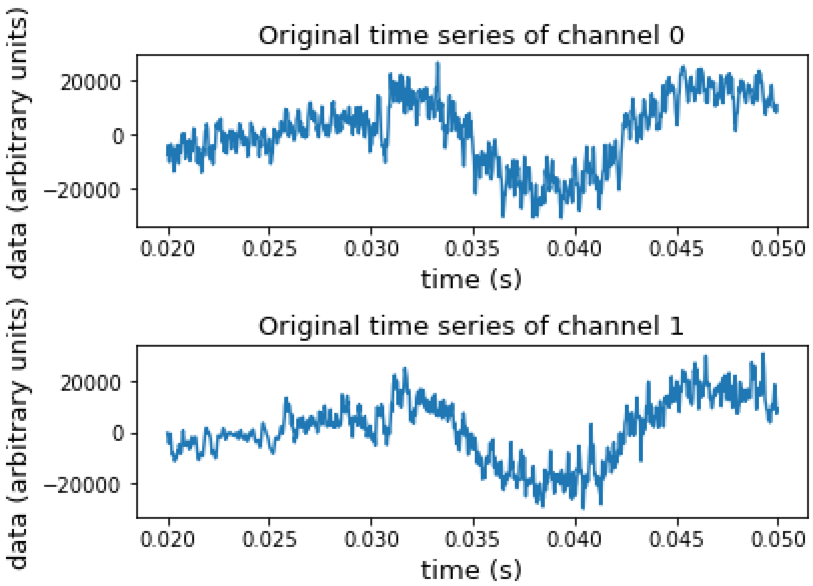

Figure 13: Original time series over the 20ms to 50ms (same as Figure 5)

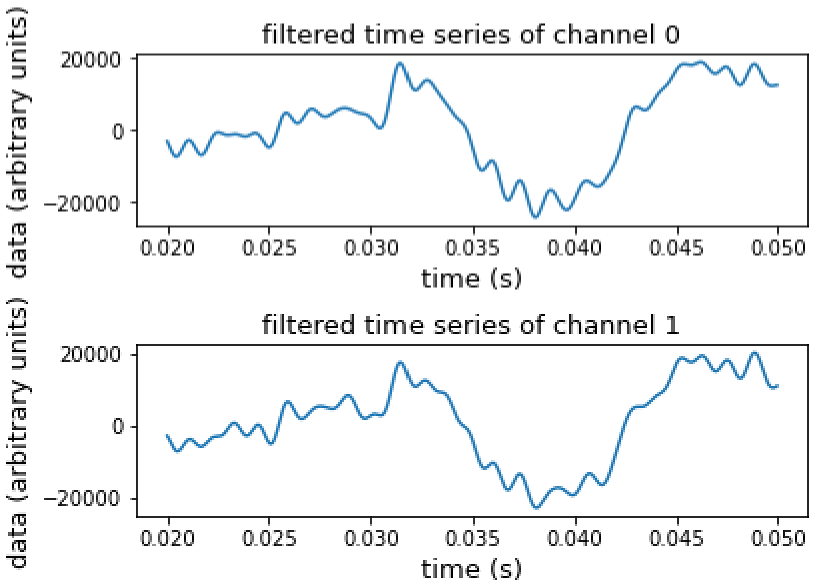

Figure 14: Filtered time series over the 20ms to 50ms

Q3 a)

Pseudocode

```
import numpy
import matplotlib.pyplot

Read 'SLP' text file
Read 'Ion' text file
Read 'times' text file

(1) Apply 2D Fourier Transform on 'SLP' data

(2) Create new numpy array that fills all entries to 0 except at Fourier
wavenumber $m = 3$

(3) Apply 2D inverse Fourier Transform

Repeat task (1),(2),(3) for wavenumber $m = 4$ and $m = 5$

plot contour map of SLP
plot contour map of SLP at wavenumber $m = 3$
plot contour map of SLP at wavenumber $m = 4$
plot contour map of SLP at wavenumber $m = 5$
```

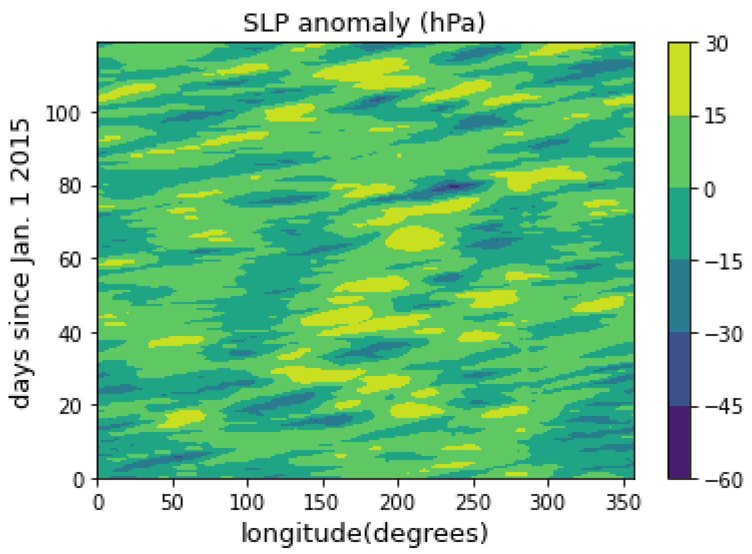

Figure 15: SLP anomaly in hPa for given SLP file

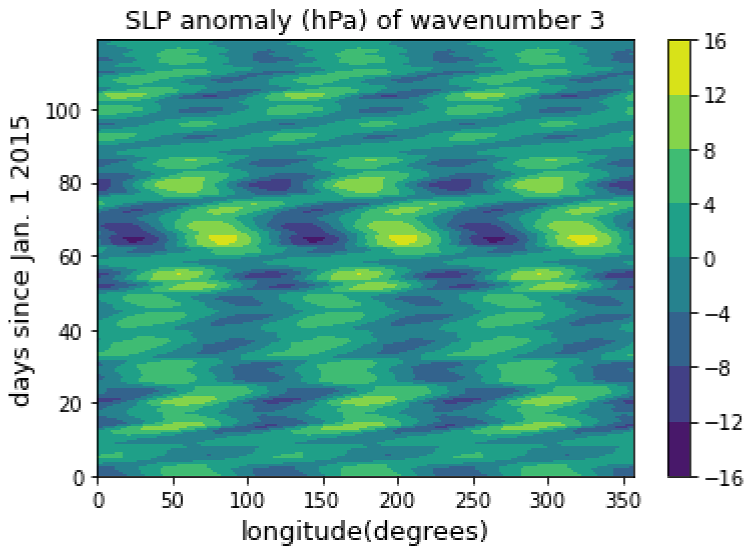

Figure 16: SLP anomaly in hPa of wavenumber $m = 3$

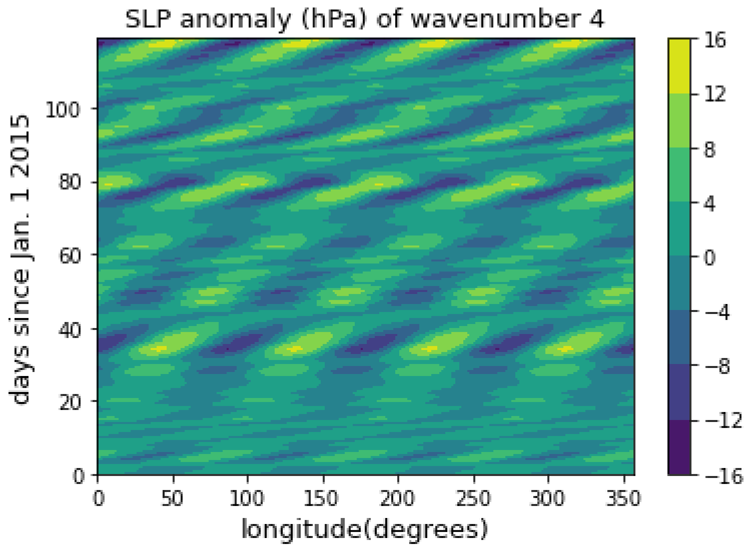

Figure 17: SLP anomaly in hPa of wavenumber $m = 4$

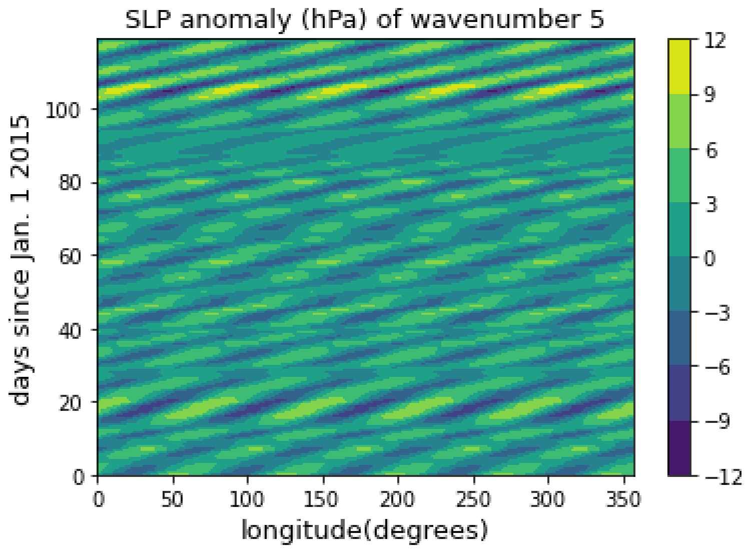

Figure 18: SLP anomaly in hPa of wavenumber $m = 5$

Q3 b)

Theory of atmospheric wave propagation states that phase speed depends on the wavelength, where wavelength is proportional to wavenumber, i.e. for wavelength λ and wavenumber m, $\lambda = \frac{1}{m}$. For the case of given SLP data, theory suggests that shorter waves travel eastward faster than longer waves, meaning plots with higher wavenumbers are expected to travel eastward faster than lower wavenumbers. As theory suggests, Figure 16, 17 and 18 shows a trend that atmospheric waves are propagating eastward (from left to right as days are passing) as wavenumbers are higher.