```python
# -*- coding: utf-8 -*-
"""
Created on Tue Jan 22 14:03:28 2019

@author: sfg17
"""

import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import hilbert
from scipy.optimize import curve_fit
import scipy.fftpack as spf
import os

path = "/Users/ShaunGan/Desktop/interferometry/Tungsten"
os.chdir(path)

##############
###Analysis###
##############

#Load in the data
fname = 'yellow_tungsten_3_good.txt'
f = open(fname,'r')
signal=[]
t_sec=[]
t_usec=[]
position=[]

lines=f.readlines()
for line in range(len(lines)):
    signal.append(lines[line].split()[0])
    t_sec.append(lines[line].split()[1])
    t_usec.append(lines[line].split()[2])
    position.append(lines[line].split()[3])


#Perform Hilbert Transform to fit envelope
position = np.array(position)
position = position.astype(float)
signal = np.array(signal)

analytic_signal = hilbert(signal,axis= -1)

analytic_imag_signal = hilbert((np.imag(analytic_signal)))
amplitude_envelope = np.abs(analytic_imag_signal)

plt.figure()
plt.ylabel("Signal (a.u.)")
plt.xlabel("Postion in mm")
plt.plot(position,np.imag(analytic_signal))
plt.plot(position,amplitude_envelope)
plt.plot(position,-amplitude_envelope)


# Curve fit to fit beating
y = amplitude_envelope
x = position
ref = np.imag(analytic_signal)

def beating(x,amplitude,scale,phase):
    return amplitude * np.sinc(scale * x - phase)

guess_amplitude = 3000
guess_scale = 1.094
guess_phase = 0.63

p0 = [guess_amplitude,guess_scale,guess_phase]
fit = curve_fit (beating,x,y,p0=p0,maxfev = 1000000)
data_fit = beating(x , *fit[0])

plt.figure()
plt.plot(x,y)
plt.figure()
plt.plot(x,ref,label='data') #plots the curve along the y axis
plt.plot (x, data_fit,'r:')


#Find Tophat Function
nsamp = len(x)
sampling_speed = 0.005e-3 #m/s
dsamp = 2 * sampling_speed / 50 #times two as something due to the path

N= nsamp
Fs=1
Ts=1/Fs
t= np.linspace(-1,Ts,50)
f=5

func_y = amplitude_envelope
fy=(spf.fft(func_y,N))
fr = np.multiply(np.arange(0,N-1,1),Fs/N)

plt.figure()
plt.title("Experimental")
plt.plot(fr,spf.fftshift(abs(fy))[:nsamp-1])
plt.xlabel('Reciprocal distance in mm^(-1) ')
plt.ylabel(' Magnitude')

#%%
```

```
########################################################
###Exptected outcome - Simulation using SINC Function###
########################################################
Fs=1
Ts=1/Fs
t= np.linspace(-1,Ts,50)
f=5
func_y = beating(t*f, *fit[0])
N=nsamp

fy=(spf.fft(func_y,N))
fr = np.multiply(np.arange(0,N-1,1),Fs/N)

plt.figure()
plt.title("Theoretical")
plt.plot(fr,spf.fftshift(abs(fy))[:N-1])
plt.xlabel('Reciprocal distance in mm^(-1) ')
plt.ylabel('Magnitude')
```