Computing Tasks

May 13, 2018

Exercise 1

Integration Practice

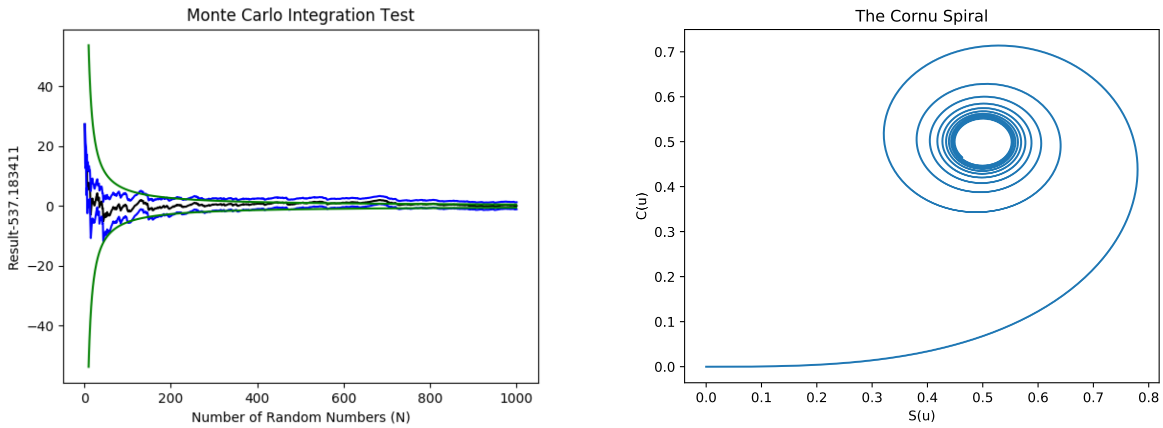

Figure 1

0.0.1 IntegrateMK2.py Code

```
import numpy as np
import operator
import matplotlib.pyplot as plt
def function(coordinate):
        return np.sin(coordinate)

def Monte_Gradual_integrate(Lower_Limit=0,Upper_Limit=np.pi/8,Dimensions=8,N=10):
        count_in=0; square_count_in=0;Results=[];Error=[]
        Volume = (Upper_Limit-Lower_Limit)**Dimensions

        for i in range(1,int(N)+1):
                coordinate = 0
                for j in range(int(Dimensions)):
                        coordinate += np.random.rand()*(Upper_Limit-Lower_Limit) + Lower_Limit
                count_in+=float(function(coordinate)); square_count_in += float((function(coordinate
                #print(count_in, square_count_in)
                Results.append(Volume*count_in/i)
                Error.append(Volume*(np.power(((-np.power(((count_in)/i),2)+(square_count_in/i))/i),
        return [x*10**6 for x in Results], [x*(10**6) for x in Error]
N=1000
Results, Error = Monte_Gradual_integrate(N=N)


plt.plot(range(1,len(Results)+1),Results,color='black')
plt.plot(range(1,len(Results)+1),list(map(operator.sub, Results, Error)),color='blue')
NError = [-x for x in Error]
plt.plot(range(1,len(Results)+1),list(map(operator.sub,Results,NError)),color='blue')
plt.xlabel('Number of Random Numbers (N)')
plt.title('Monte Carlo Integration Test')
plt.savefig('New_Plot.pdf')
plt.clf()

plt.plot(range(1,len(Results)+1),[x-537.1873411 for x in Results],color='black')
plt.plot(range(1,len(Results)+1),[x-537.1873411 for x in list(map(operator.sub, Results, Error))],co
NError = [-x for x in Error]
```

```
plt.plot(range(1,len(Results)+1),[x-537.1873411 for x in list(map(operator.sub,Results,NError))],col
plt.xlabel('Number of Random Numbers (N)')
plt.ylabel('Result -537.183411')
plt.title('Monte Carlo Integration Test')
plt.savefig('Comparative_Plot.pdf')
u = [537.183411/x for x in range(10,1000)]
l = [-537.183411/x for x in range(10,1000)]
plt.plot(range(10,1000),u,color='green')
plt.plot(range(10,1000),l,color='green')
plt.savefig('Compartative_Plot_Add_Lines.png')
```

0.0.2 CornuIntegrate.py Code

```
import scipy
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import numpy.random as rand
from scipy.integrate import quad

def cosser(x):
        return np.cos(np.pi*(x**2)/2)

def sinner(x):
        return np.sin(np.pi*(x**2)/2)

def Integrator(u):
        C = quad(cosser,0,u)[0]
        S = quad(sinner,0,u)[0]
        return C,S

x=[]; y=[]
for i in np.linspace(0,2*np.pi,num=1000):
        C,S = Integrator(i)
        y.append(S); x.append(C)

plt.plot(x,y)
plt.xlabel('S(u)')
plt.ylabel('C(u)')
plt.title('The Cornu Spiral')
plt.savefig('Cornu_Spiral.pdf')
```

Apertures

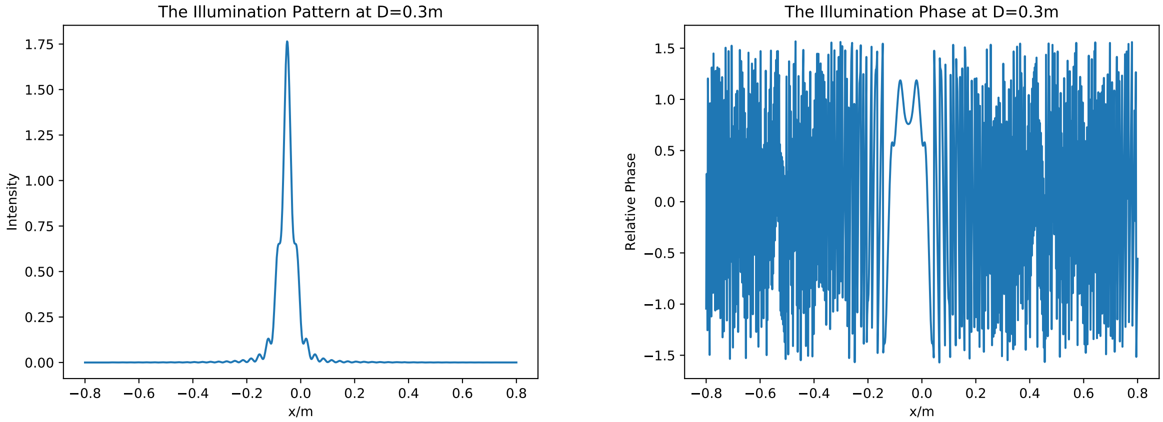

Figure 2

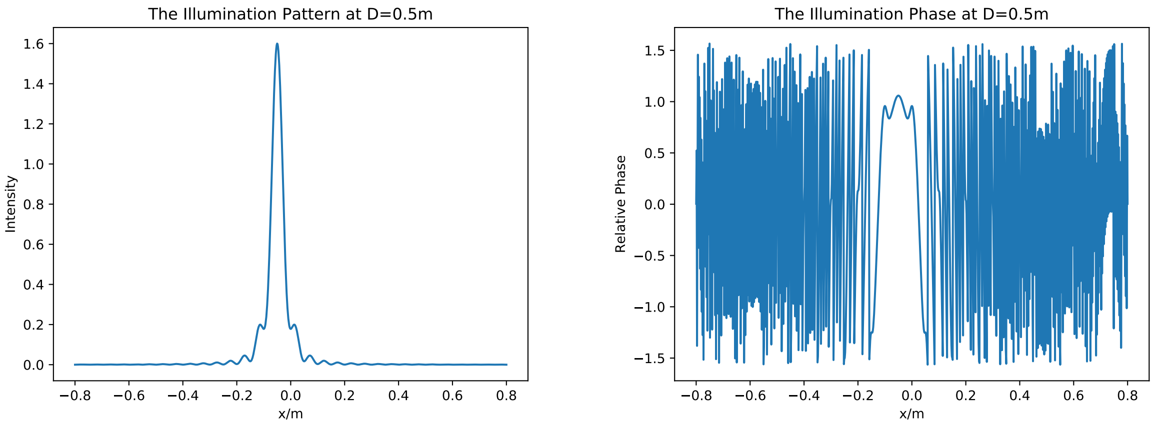

Figure 3

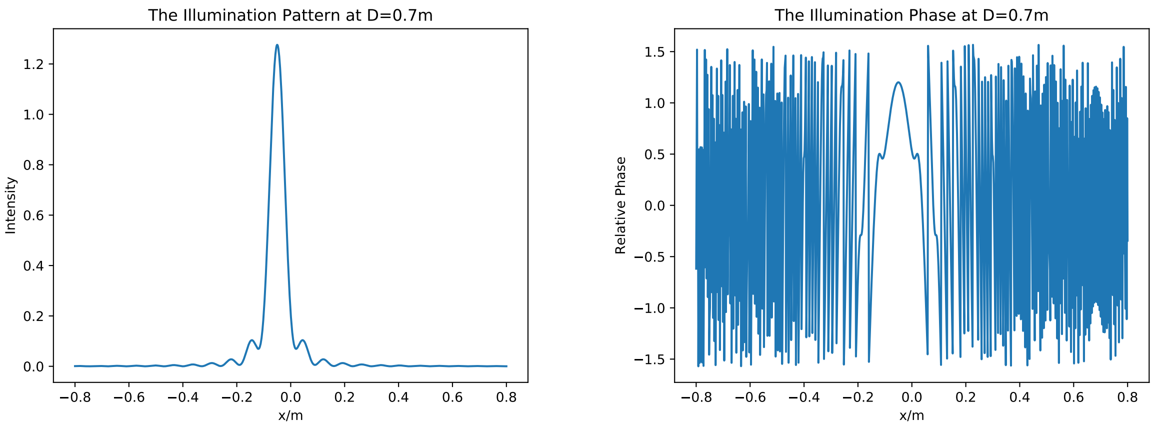

Figure 4

Aperture Code

```
import scipy
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import numpy.random as rand
from scipy.integrate import quad

def cosser(x):
        return np.cos(np.pi*(x**2)/2)

def sinner(x):
        return np.sin(np.pi*(x**2)/2)

def Integrator(u):
        C = quad(cosser,0,u)[0]
        S = quad(sinner,0,u)[0]
        return C,S

x=[]; y=[]

def Two_Ended(x0,x1,lam=0.01,D=0.3):
        Scaling = (2/(lam*D))**0.5
        u2 = x1*Scaling; u1 = x0*Scaling
        imag1, real1 = Integrator(u1)
        imag2, real2 = Integrator(u2)
        imag = -imag1+imag2
        real = -real1+real2
        mag = (imag**2+real**2)*0.5
        arg = np.arctan(imag/real)
        return mag, arg


def Plot_Mag_Arg(d=0.1,lam = 0.01,D=0.3):
        mag_list = []; arg_list =[]
        x = np.linspace(-0.8,0.8,num=1000)
        for i in x:
                mag, arg = Two_Ended(i,i+d,D=D,lam=lam)
                mag_list.append(mag); arg_list.append(arg)
        plt.plot(x,mag_list)
        plt.xlabel('x/m')
        plt.ylabel('Intensity')
        plt.title('The Illumination Pattern at D='+str(D)+'m')
        plt.savefig('Apperture_Pattern_'+str(D)+'_'+'.pdf')
        plt.clf()
        plt.plot(x,arg_list)
        plt.xlabel('x/m')
        plt.ylabel('Relative Phase')
        plt.title('The Illumination Phase at D='+str(D)+'m')
        plt.savefig('Phase_Pattern_'+str(D)+'_.pdf')
        plt.clf()

Plot_Mag_Arg()
Plot_Mag_Arg(D=0.5)
Plot_Mag_Arg(D=0.7)
```

Exercise 2

The effect of order

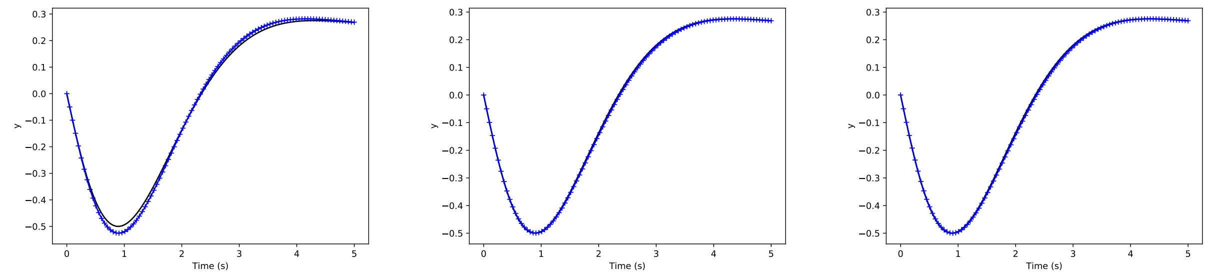

Figure 5

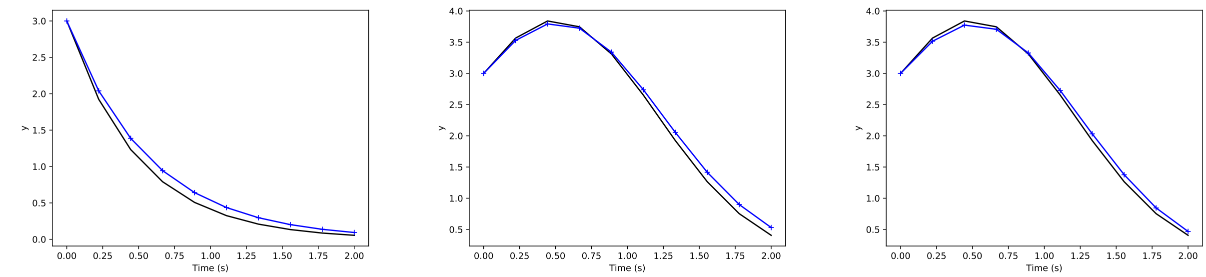

Figure 6

The effect of time-step

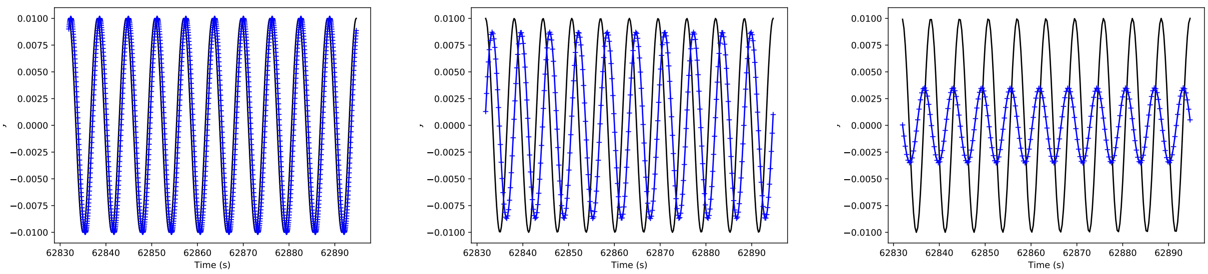

Figure 7

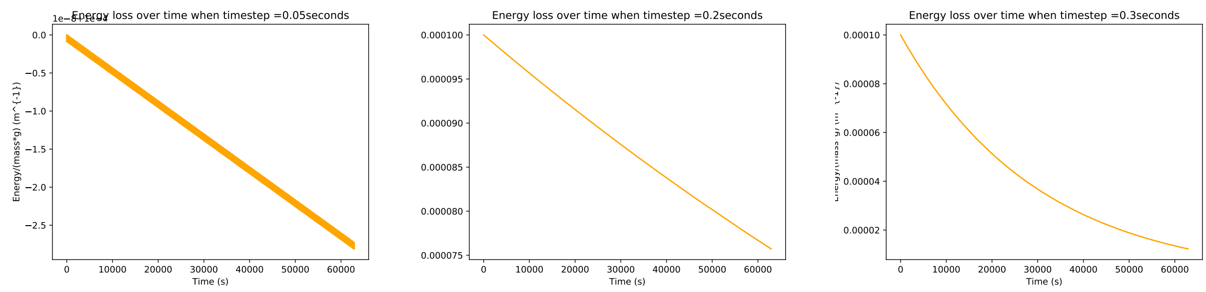

Figure 8

Light Damping

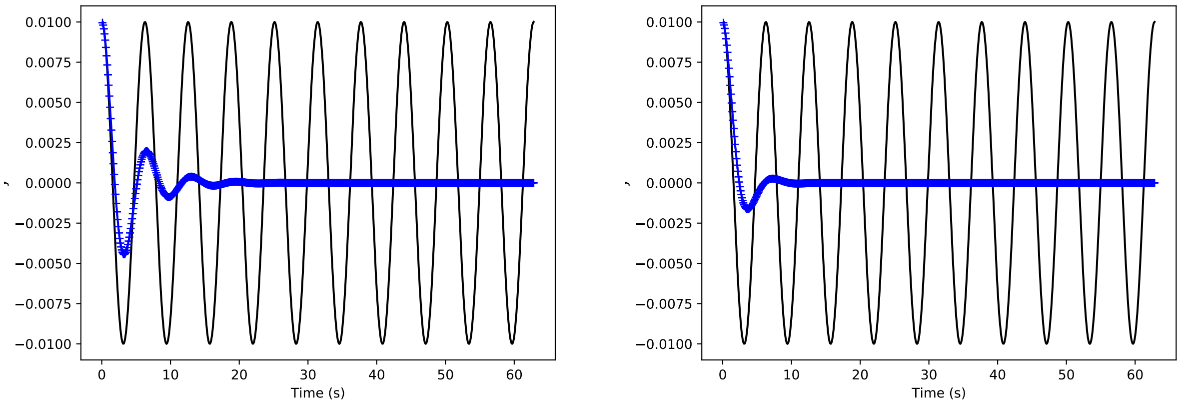

Figure 9

Overdamping

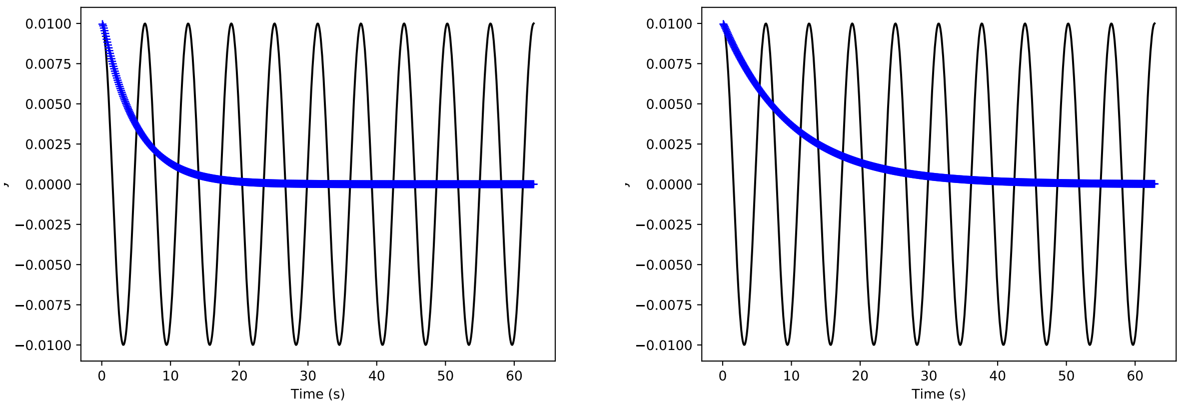

Figure 10

Forced

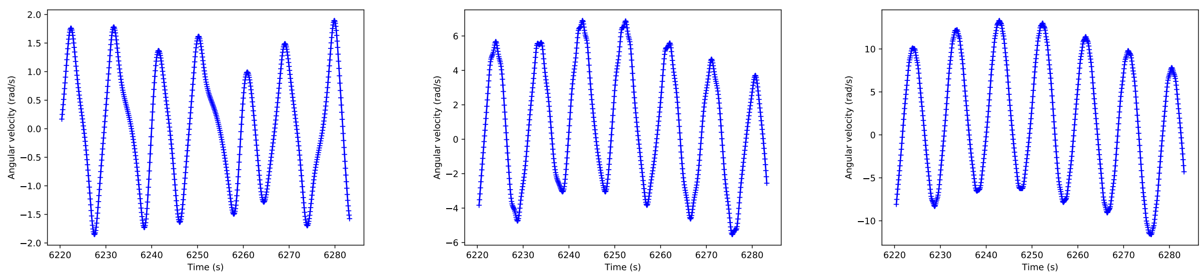

Figure 11: Apparent period of oscillation (when Forcing=0.5) (from sign changes of ang velocity): 9.42477796077 Apparent period of oscillation (when Forcing=1.2) (from sign changes of ang velocity): 11.0879740715 Apparent period of oscillation (when Forcing=1.44) (from sign changes of ang velocity): 9.42477796077 Apparent period of oscillation (when Forcing=1.465) (from sign changes of ang velocity): 9.42477796077

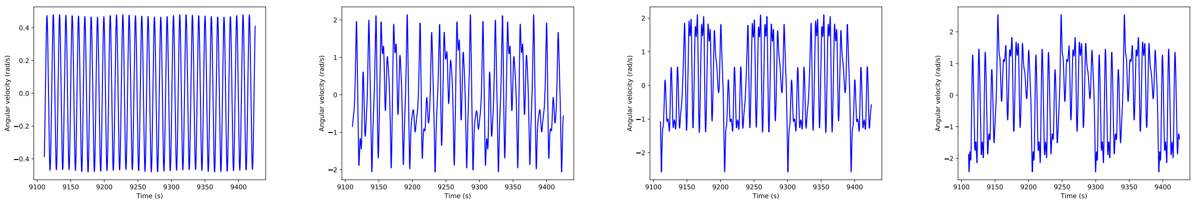

Figure 12

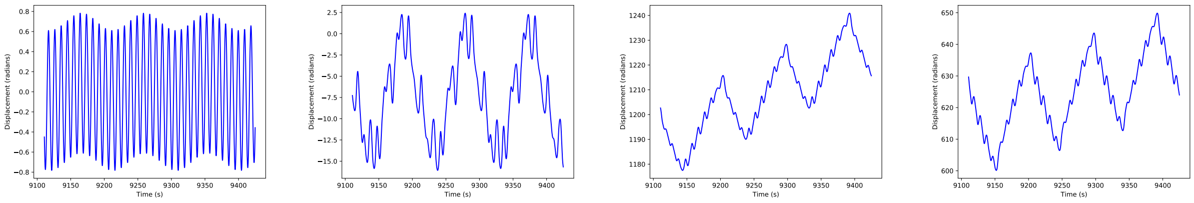

Figure 13

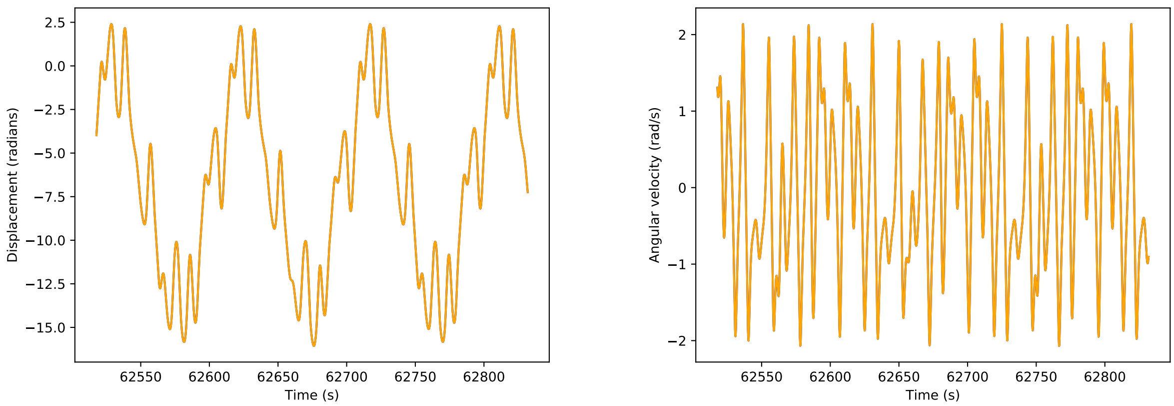

Figure 14

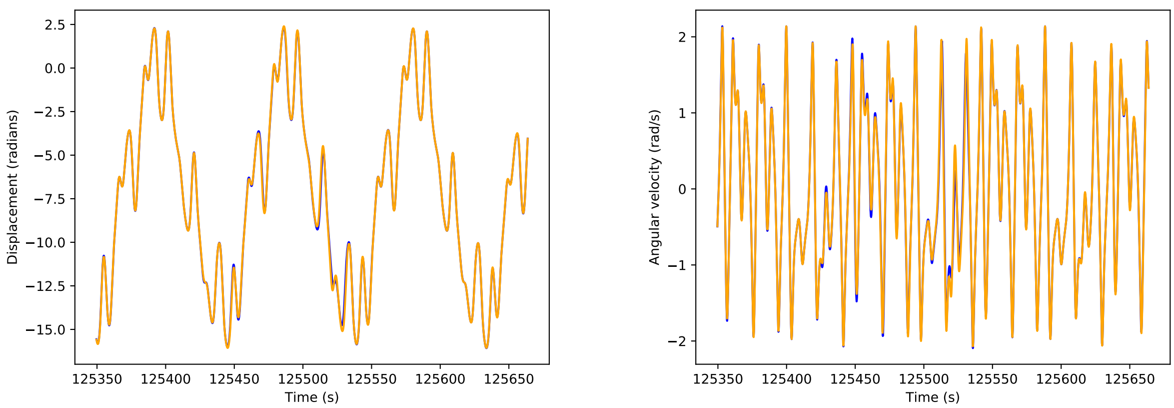

Figure 15

Exercise 3

Task 1

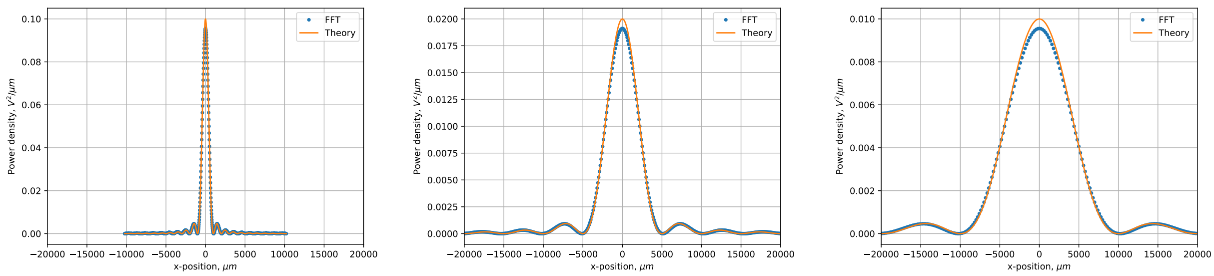

Figure 16

Task 2

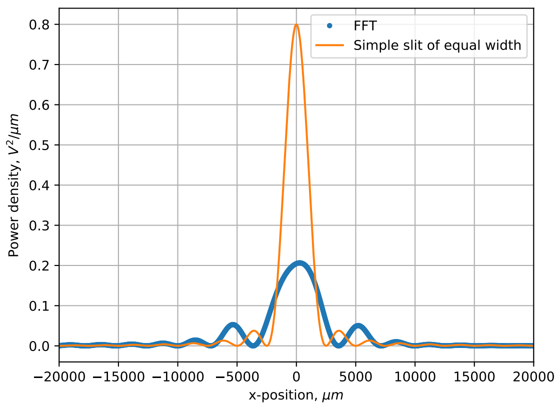

Figure 17

Task 3

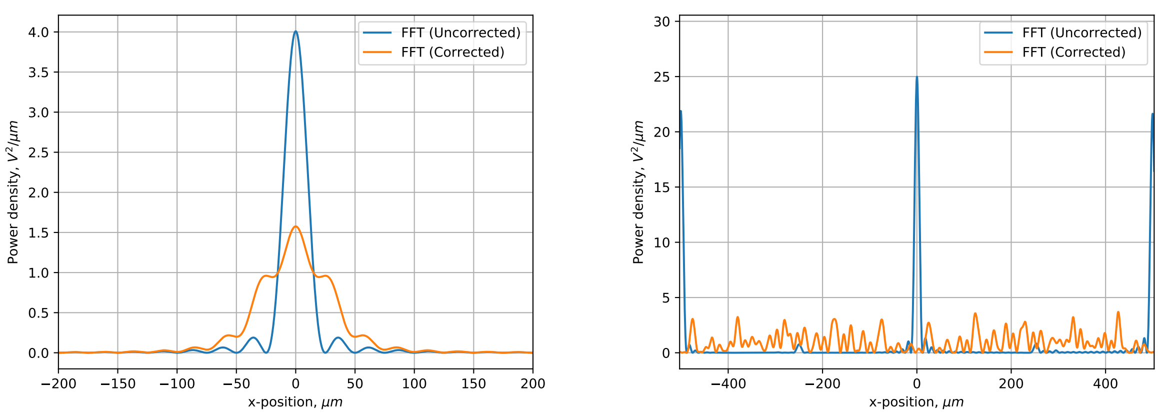

Figure 18