# PH10102 Python Programming Coursework April 2019

**21518**

**Submitted: 17/04/19**

1. Secant method to obtain the Lagrange point between the Earth and the Moon.

(a) Net forces on satellite are as shown,

$$F_{G-Earth} - F_{G-moon} = F_{Centripetal-Sat}$$

$$\frac{GM_E m_{Sat}}{r^2} - \frac{Gm_m m_{Sat}}{(R-r)^2} = m_{Sat}\omega^2 r.$$

Divide through by the mass of the satellite, giving the desired equation;

$$\frac{GM}{r^2} - \frac{Gm}{(R-r)^2} = \omega^2 r\,.$$

(b) To begin with, the formula is rearranged to equal zero, such that;

$$\frac{GM}{r^2} - \frac{Gm}{(R-r)^2} - \omega^2 r = 0$$

this can now be applied to the secant method in python as shown.

```
1 #Data
2 G=6.6741*10**-11
3 R_e=6.371*10**6
4 R_m=1.7371*10**6
5 M=5.9722*10**24
6 m=7.3420*10**22
7 R=3.8440*10**8
8 w=2.6617*10**-6
9
10 r1=10000
11 r2=40000
12 delta=10
13
14
15 def func(r):
16     return (G*M/r**2)-(G*m/(R-r)**2)-(w**2*r)
17
18 def deriv(r1, r2):
19     return (func(r2)-func(r1))/(r2-r1)
20
21 while(abs(delta)>0.0001):
22     delta= -func(r2)/deriv(r1,r2)
23     r3=r2+delta
24     r1=r2
25     r2=r3
26 r=r3/10**8
27 print('The lagrangian point from the centre of the Earth'' is {:1.4f}*10^8 m'.format(r))
```

Lines 1 to 8 store the values of all the data give, whilst lines 10 and 11 are the initial guesses for the secant method. The formula will iterate around these two points until a root is obtained, with line 12 giving any number as an initial delta for the code to loop round. Line 15/16 states the function we derived. Line 18/19 defines the derivative of the function in terms of the secant method, essentially $\frac{\Delta y}{\Delta x}$. Then lines 21 to 25 iterate the function in a while loop, stopping when delta is less than 0.0001, in other words the error. Lines 26/27 are formatting on the answer, giving it to 5 significant figures.

```
The lagrangian point from the centre of the Earth is 3.2605*10^8 m
```

Above is the python console, showing the Lagrange point, $r$ being $3.2605 \times 10^8 m$ from the centre of the Earth.

## 2. Bessel Function

(a) Writing a python function to integrate the Bessel function using the trapezium rule with 10000 strips.

The Bessel functions, $J_m(x)$ are,

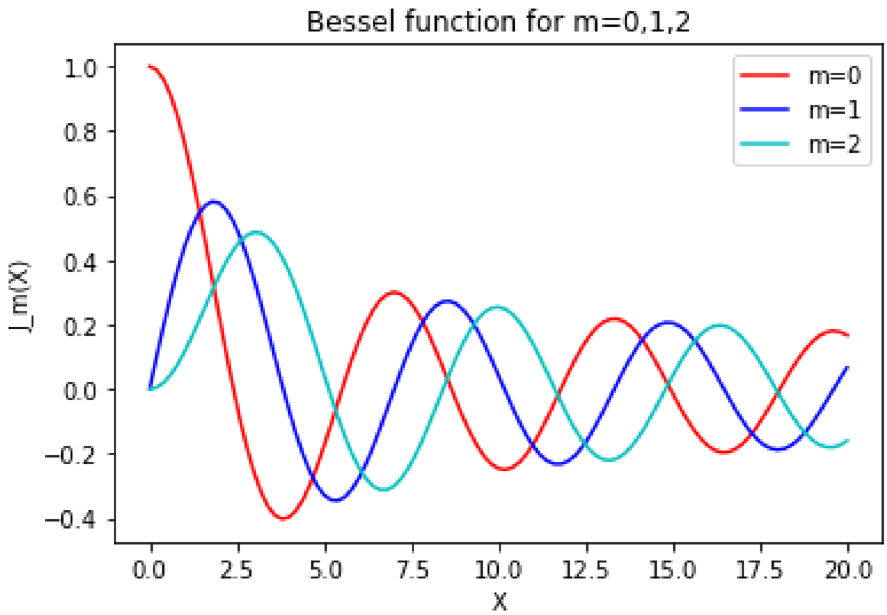
$$J_m(x) = \frac{1}{\pi}\int_0^{\pi} \cos(m\theta - x\sin(\theta))\, d\theta$$

where, $m$ is a non-negative integer and $x \geq 0$. The desired functions are where $m = 0,1,2$;

```
import numpy as np
import matplotlib.pyplot as plt

def func(theta, x, m):
    return (1/np.pi)*(np.cos(m*theta - x*(np.sin(theta))))

j_0 = []
j_1 = []
j_2 = []

#Trap
N=10000
a=0
b=np.pi
h=(b-a)/N

#Bessel
A=0
B=20
#Smoothness coeff
S=100
H=(B-A)/S
X=A+H*np.arange(0,S+1)


while A<=B:
    total=0
    n=1
    while n<N:
        total = total + func((a+n*h), A, 0)
        n=n+1
    j_0.append(0.5*h*(func(a, A, 0)+func(b, A, 0)+2*(total)))
    A=A+H
```

```
plt.title('Bessel function for m=0,1,2')
plt.plot(X,j_0, 'r-', label='m=0')
plt.plot(X,j_1, 'b-', label='m=1')
plt.plot(X,j_2, 'c-', label='m=2')
plt.xlabel('X')
plt.ylabel('J_m(X)')
plt.legend(loc='upper right')
```

Lines 4/5 define the Bessel function in terms of theta, x and m; lines 7-9 define the lists used for the functions. Lines 12-15 define the parameters for the trapezium rule, and lines 18-23 define the limits of the Bessel function for x and the increments on the x-axis (the smoothness coeff as it's called on the code). Lines 26-33 are a while loop for the trapezium rule, the trapezium rule is applied to the Bessel function over theta and then values of x applied to this integrated function. Shown is m=0 and this is repeated for m=1 and m=2. This approach could be simplified for analysis of other m values by implementing a while loop that allows a choice of m, but for the case of just three it seems viable to simply repeat the code. Lines 55-61 are formatting of the plot; the graph is as shown,




(b) Diffraction pattern

The intensity of light is given by,

$$I(r) = I_0\left(\frac{2J_1(x)}{x}\right)^2$$

where, $x = ka\sin(\theta) = \frac{2\pi}{\lambda}a\frac{r}{R}$. The focal ratio has been given as $\frac{R}{2a} = 10$, and the wavelength used was $500nm$ as this is a value of visible light. The values of $r$ for the diffraction pattern are over the range of $\pm 25\mu m$. As shown in the formula, the value of m used is 1 in this case, so the trapezium rule will work over the Bessel function as such.

```python
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def func(theta, x, m):
5     return (np.cos(m*theta - x*(np.sin(theta))))
6 def x_val(r):
7      return (np.pi/lamb)*(r/10)
8 def inten(r,j):
9     return I_0*((2*j)/x)**2
10
11 #Trap
12 N=10000
13 a=0
14 b=np.pi
15 h=(b-a)/N
16
17 j_1 = []
18 r = np.linspace(-25,25,101)
19 theta=np.linspace(0,np.pi,N)
20
21 #Diffraction
22 lamb=0.5
23 I_0=1
24 Intensity=[]
25
26 for i in range(0,len(r-1)):
27     total=0
28     x=x_val(r[i])
29     n=0
30     if r[i] == 0:
31         Intensity.append(1)
32     else:
33         for n in range (0, N-1):
34             total = total + func((a+n*h), x, 1)
35         Integral= h*(0.5*func(theta[1], b,1)+ total)
36         Bessel= (1/np.pi)*Integral
37         intensity_val= inten(x_val(r[i]), Bessel)
38         Intensity.append(intensity_val)
39
40 plt.title('Diffraction intensity for Bessel m=1')
41 plt.plot(r,Intensity, 'r-')
42 plt.xlabel('r (m*10^-6)')
43 plt.ylabel('I(r)')
```

Lines 4-9 define all the functions used, with line 7 giving the simplified formula for x. Lines 12-15 are the same trapezium rule parameters as before, but this time a slightly different approach was used rather than the while loop. Lines 19 and 19 use the linspace function to define a list of values before the bounds stated. With, 101 values of r before -25 and 25, then 10000 values of theta between 0 and pi. Now, the trapezium rule is within a for/if/else loop. With line 26 defining what range to apply this loop, then lines 27-29 stating initial values. Lines 30 and 31 give and if statement for the intensity function at r=0, the function cannot compute a value divided by zero, so the if statement forces it to output 1 as a maximum. The else loop in lines 32-38 compute the trapezium rule on the Bessel function and apply values of r to the integrated function. This is then put into a list and plotted using the formatting lines in 40-43. This approach was an improvement from the difficulties in using the while loop imbedded in other loops and proved successful after many attempts of different methods.