Exploring Numerical Computation Methods in Python

24554
Department of Physics, University of Bath,
Bath BA2 7AY, UK

Submitted 06.05.2022

Introduction

The aim of this project is to demonstrate concise programme design skills in order to solve a number of numerical physics problems as well as present them graphically in order to define relationships between parameters.

Obtaining Wein's Constant from Planck's Displacement Law

Plancks Radiation Law describes the density of electromagnetic radiation emitted by a black body assuming that there is no overall flow of radiation between the black body and its immediate surroundings.[1] The relationship between density of radiation ρ and its wavelength λ is given by

$$\boldsymbol{\rho(\lambda)d\lambda = \frac{8\pi hc}{\lambda^5(\exp\left(\frac{hc}{\lambda kT}\right)-1)}d\lambda} \qquad \textbf{(1)}$$

where *h* is Planck's constant, *c* is the speed of light, *k* is the Boltzmann constant and *T* is the temperature of the black body. [1]

In order to obtain Wien's constant from equation 1 it must be differentiated, set equal to zero and subsequently solved for its roots. [2]

By using the substitution

$$\boldsymbol{x = \frac{hc}{\lambda kT}} \qquad \textbf{(2)}$$

and rearranging for λ

$$\boldsymbol{\lambda = \frac{hc}{xkT}} \qquad \textbf{(3)}$$

λ can be removed entirely from Planck's Displacement Law leaving behind

$$\boldsymbol{\rho(x) = \frac{8\pi(xkT)^5}{(hc)^4(\exp(x)-1)}} \qquad \textbf{(4)}$$

which can then be differentiated by hand using the product rule which finally gives

$$\boldsymbol{\frac{d\rho}{dx} = e^x(5-x)-5} \qquad \textbf{(5)}$$

This equation can be equated to zero which corresponds to the turning point of the original function ρ(λ).

The programme designed on python uses the Newton-Raphson method to obtain the roots of this function by solving for *x*.

It followed a four-step structure. The parameters were defined, and the relevant python libraries imported, then increasingly complicated functions were defined. Provisional graphs were plotted to better visualise the problem and finally relevant parameters were substituted into the main function and the roots therefore identified.

First, $\frac{d\rho}{dx}$ was plotted as a function of *x* to roughly see where the roots lie in order to have a more accurate estimate of the variable *x* in the subsequent Newton-Raphson programme.

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

def dev_p(x):
    return (np.exp(x)*(5-x))-5

xvalues=np.linspace(-1,5,100)
yvalues=dev_p(xvalues)

plt.axhline(0, color='black',lw=0.5)
plt.plot(xvalues,yvalues)
plt.xlabel("x values",fontsize=12)
plt.ylabel("dp/dx",fontsize=12)
plt.savefig("q1 graph")
plt.show()
```

The plt.axhline() adds a line at y=0 so it is easier to see where the function crosses the x-axis therefore revealing the general location of the roots.

The font size of the axis labels was also increased to make it more legible.

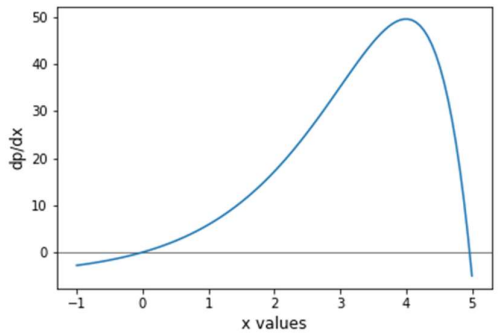

Figure 1. A graph of $\frac{d\rho}{dx}$ plotted against its corresponding x values to estimate the two roots of the first derivative.

It is clear to see that $x = 0$ or $x \approx 5$. From this a more accurate value for the initial "guess" of the root can be obtained and therefore a more accurate value for the actual value can be calculated.

The Newton Raphson method is a technique based on the Taylor series expansion which finds increasingly more accurate approximations of the roots of a real-valued function. Assuming that the initial "guess" of the root is x_0, the actual value is

$$(\boldsymbol{x} + \boldsymbol{\delta}) \qquad \textbf{(6)}$$

where

$$\boldsymbol{\delta} \approx \frac{-f(x_1)}{f'(x_1)}. \qquad \textbf{(7)}$$

The following code was then used to solve $\frac{d\rho}{dx}$.

```
import numpy as np

error=10**-10

def dev_p(x):
    return (np.exp(x)*(5-x))-5

def secdev_p(x):
    return (4-x)*np.exp(x)
```

The mathematical operations package numpy was imported.

Then two functions were defined: ρ'(x) and ρ''(x) respectively. dev_p(x) is the function for which the roots were obtained and secdev_p(x) which is the derivative of dev_p(x) needed to calculate δ in the Newton-Raphson method.

Then the function which performs the Newton Raphson calculation was defined.

```
def newtonraphson(x0):
    error=10**(-10)
    for i in range(1000):
        xn = x0 - (dev_p(x0)/secdev_p(x0))
        if abs(xn-x0)<error:
            break
        x0=xn
    return xn

print(newtonraphson(4.5))
```

A 'for' loop was used to run through 1000 increasingly accurate guesses for the root. The 'if' statement ensures that if the absolute difference between two successive guesses is greater than the allowed error (defined inside the function), the loop will break and the final value for the root will be printed.

This programme obtained the value x=4.965114231744276 which is indeed close to 5 as it was originally predicted in Fig.1.

Wien's Displacement Law is given by

$$\boldsymbol{\lambda_{max}} = \frac{\boldsymbol{b}}{\boldsymbol{T}} \qquad \textbf{(8)}$$

where b is Wien's constant [2]. Reusing the substitution in equation 2 it is found that

$$\boldsymbol{b} = \boldsymbol{\lambda_{max} T} = \frac{\boldsymbol{hc}}{\boldsymbol{kx}}. \qquad \textbf{(9)}$$

The programme continues as follows:

```
h=6.626E-34
c=299792458
k= 1.3806503E-23

constant=((h*c))/(k*newtonraphson(4.5))
print(constant)
```

The final value of Wien's constant, b=0.0028977385480924624mK or b=2.898E-3mK which aligns with the literature value. [3]

Calculating the diffraction pattern of a telescope using the trapezium method of integration.

The first exercise of this section involves calculating values for the Bessel function

$$J_m(x) = \frac{1}{\pi}\int_0^{\pi} \cos(m\theta - x\sin\theta)\, d\theta \quad (10)$$

where m is a constant and x is a non-negative integer.

The trapezium method of integration involves dividing the integration interval from a to b into N equal trapezia of width h where

$$h = \frac{b-a}{N}, \quad (11)$$

and summing them.

The following code was used to achieve this:

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

def func(m,o,x):
    return np.cos((m*o)-(x*np.sin(o)))

a=0.0
b=np.pi


def trapezium(a,b,m,x):
    N=10000
    h=(b-a)/N
    fab=0.5*(func(m, a, x)+func(m, b, x))
    extra=0
    for k in range(1,N):
        extra+=func(m, a+k*h, x)
    return h*(fab+extra)

def J(m,x):
    return (1/np.pi)*trapezium(a, b,m,x)
```

Firstly, the function being integrated was defined as func(m,o,x). Then the function which applies the trapezium rule for integration was defined as trapezium(a,b,m,x) with parameter N=10000 indicating that there will be 10000 trapeziums all summed together.

Finally the full Bessel function was defined as J(m,x) and for values m = 0,1,2 and $0 \leq x \leq 20$ graphs were plotted.

```
x=np.linspace(0,21,1000)

fig = plt.figure(figsize=(10,6))
ax = fig.add_subplot(1,1,1)

plt.plot(x,J(0,x),label="$J_0(x)$")
plt.plot(x,J(1,x),label= "$J_1(x)$")
plt.plot(x,J(2,x),label ="$J_2(x)$")
plt.xlabel("x values",fontsize=15)
plt.ylabel("Bessel Function $J_m(x)$",fontsize=15)
plt.legend()
plt.savefig("Bessel Graph")
plt.show()
```

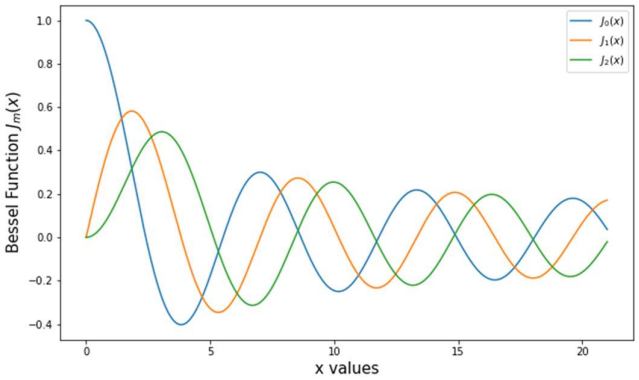

Figure 2. Plot of the Bessel functions for increasing *m* number.

It is interesting to note that as the m number increases, the overall amplitude of the function decreases. This shows that as $x \to \infty$, the function goes to zero. Putting this into the context of the light intensity from a source in the radial direction, it shows that the further away the subsequent fringe is, the less intense it will be. As x increases the individual amplitude of the function decreases.

When light of wavelength λ, passes through a circular aperture, the image produced on the focal plane is a diffraction pattern made up of a bright central circular fringe surrounded by concentric rings whose brightness decreases as the distance from the centre fringe increases [4]. This pattern is known as an Airy Disk.

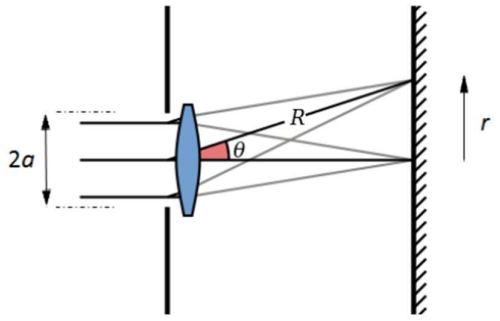

Figure 3. Diagram detailing the internal setup of a telescope with a circular aperture.

The intensity of the light in this phenomenon is described by

$$\boldsymbol{I(r) = I_0 \left(\frac{2J_1(x)}{x}\right)^2} \qquad \textbf{(12)}$$

where I_0 is the maximum intensity which, for the purpose of this exercise, is set to 1. [4]

Using the substitution

$$\boldsymbol{x = k\, a\, sin\theta = \frac{2\pi}{\lambda} a \frac{r}{R}}, \qquad \textbf{(13)}$$

as well as the fact that the focal ratio, $\frac{R}{2a} = 10$ and λ = 550nm, it can be determined that

$$\boldsymbol{x = \frac{\pi r}{10\lambda}}. \qquad \textbf{(14)}$$

The following programme calculated the diffraction pattern of the light passing through the telescope where -25≤ r ≤25 µm.

```
r=np.linspace(-25E-6,25E-6,10000)
lambda_val=550E-9
x_val=(np.pi*r)/(10*lambda_val)
I_0=1.0

def Intensity(x_val):
    return I_0*((2*J(1,x_val)/x_val))**2

plt.plot(r,Intensity(x_val))
plt.axhline(color='black', lw=0.5)
plt.axvline(color='black', lw=0.5)
plt.xlabel("Distance from the central fringe (m)",fontsize=10)
plt.ylabel("Intensity of Light (W/$m^2$)",fontsize=10)
plt.savefig("Diffraction Pattern")
plt.show()
```

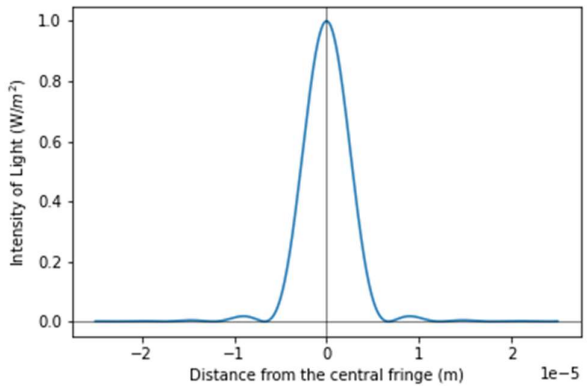

Figure 4. Diffraction Pattern in the r plane of light entering a telescope with a circular aperture. Shown with a Gaussian profile.

The distance from the central fringe, r, was defined as an array of 10000 equally spaced values in the range of -25 to 25µm using the command np.linspace().

The intensity function was defined to return equation 12 with the relevant substitutions made where the parameter x in equation 14 was defined as x_val in the code.

Two lines representing the x and y axis were also added to aid with visually representing the diffraction pattern. It is clearer to see that the intensity peaks at r=0 and never falls below I=0.

Figure 4 can be used to determine the location of the dark fringes by finding the r values when the intensity $I(r)$=0.

It is interesting to note that the subsequent peaks after the central peak are dramatically smaller. This shows that the light is not dispersed around the telescope but rather focused onto the eyepiece of the observer suggesting that when the focal ratio $\frac{R}{2a} = 10$, the image seen through the telescope has a higher resolution.

References

[1] Planck M, Masius M. *The Theory of Heat Radiation.* 2nd Edition. Philadelphia P. Blakiston's Son & Co; 1914

[2] Das B. Obtaining Wein's displacement law from Planck's law of radiation. *The Physics Teacher*. 2002; 40 pp.148-149

[3] The NIST Reference on Constants, Units and Uncertainty. *Wien wavelength displacement law constant.* [Internet] 2019 [cited 5 May 2022] Available from https://physics.nist.gov/cgi-bin/cuu/Value?bwien

[4] Born M, Wolf E *Principles of Optics*, 7th Edition. Cambridge, Cambridge University Press; 1999