**EXERCISE 2.1: INTEGRATION USING SIMPSON’S RULE**

1. Starting with Example 2.3 in the notes, add a function to perform Simpson’s rule integration.

Code :

import numpy as np

def Simpson(function, a, b, N):

'''

Parameters

----------

function : function of a single variable

a, b : interval of integration [a,b]

N : number of subintervals of [a,b]

Returns

----------

Approximation of the integral of f(x) from a to b using the

Simpson rule with N subintervals of equal length.

Examples

----------

>>> Simpson(lambda x: x\*\*4 - 2\*x + 1, 0.0, 2.0, 10)

4.50656

>>> Simpson(lambda x: x\*\*4 - 2\*x + 1, 0.0, 2.0, 1000)

4.40001066667

>>> Simpson(lambda x: np.sin(x), 0.0, 2.0, 10)

1.4161463644981664

'''

h = (b - a)/N

s = 0.5\*function(a) + 0.5\*function(b)

for k in range(1, N):

s += function(a+k\*h)

return h\*s

1. Use the integral , for which the exact value is 4.4, to compare the accuracy of the trapezium rule and Simpsons rule for the same values of N.

Code implementation for Trapezium’s rule:

def Trapezium(function, a, b, N):

'''

Parameters

----------

function : function of a single variable

a , b : interval of integration [a,b]

N : number of subintervals of [a,b]

Returns

-------

Approximation of the integral of f(x) from a to b using the

Trapezium rule with N subintervals of equal length.

Examples

--------

>>> Trapezium(lambda x: x\*\*4 - 2\*x + 1, 0.0, 2.0, 10)

4.50656

>>> Trapezium(lambda x: x\*\*4 - 2\*x + 1, 0.0, 2.0, 1000)

4.4000106666656

'''

x = np.linspace(a, b, N+1)

y = function(x)

y\_right = y[1:] # right endpoints

y\_left = y[:-1] # left endpoints

dx = (b - a)/N

T = (dx/2) \* np.sum(y\_right + y\_left)

return T

>>> Trapezium(lambda x: x\*\*4 - 2\*x + 1, 0.0, 2.0, 1000)

4.4000106666656

>>> Simpson(lambda x: x\*\*4 - 2\*x + 1, 0.0, 2.0, 1000)

4.40001066667

Conclusion : when increasing N(number of subintervals of [a,b]), in the case of quadratic functions, the Simpsons method gave the best approximation and the Trapezoidal provided the worst.

**EXERCISE 2.2 MAXWELL SPEED DISTRIBUTION**

1. Write a function that returns a value of f(v) from Maxwell Speed distribution, given an input of v, m and T

Code :

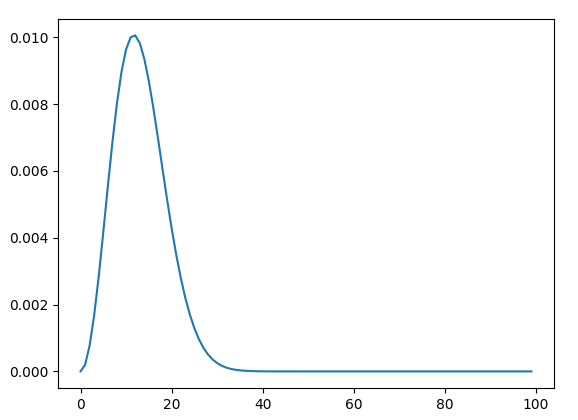
def MaxwellSpeedDistribution(v, m, T):

return 4\*PI\*((m/(2\*PI\*kB\*T))\*\*(1.5))\*v\*v\*math.exp(-(m\*v\*v)/(2\*kB\*T))

1. Calculate and plot f(v) at the temperatures of 300K an 1000K for:
2. A gas of He atoms, m = 6.65 x kg

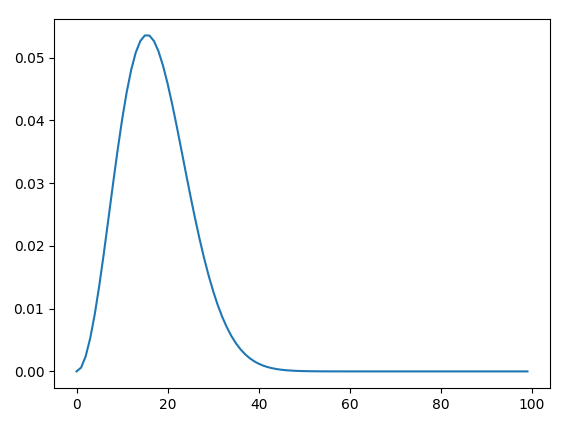
Use above function, we have :

f(v) = 1.6243 x v2 exp(-8.0314 x v2)



1. A gas of N2 molecules, m = 4.65 x kg

f(v) = 4.9352 x v2 exp(-1.6848 xv2)



1. Calculate the probability of molecule having a speed between v1 and v2 using
2. Simpson’s rule ( Exercis 2.1)

A gas of He atoms, m = 6.65 x kg

* f(v) = 1.6243 x v2 exp(-8.0314 x v2)
* P(15 < v < 25) = 0.045484531105908416

A gas of N2 molecules, m = 4.65 x kg

* f(v) = 4.9352 x v2 exp(-1.6848 xv2)
* P(15 < v < 25) = 0.44146966504554

1. Scipy.integrate.quad ()

P(15 < v < 25) = 0.04548452792973004

P(15 < v < 25) = 0.44146969867934865

**EXERCISE 2.4: DIFFRACTION LIMIT OF A TELESCOPE**

Write a python function J(m,x) that calculates the value of Jm(x) from Eqn. (5)

Code :

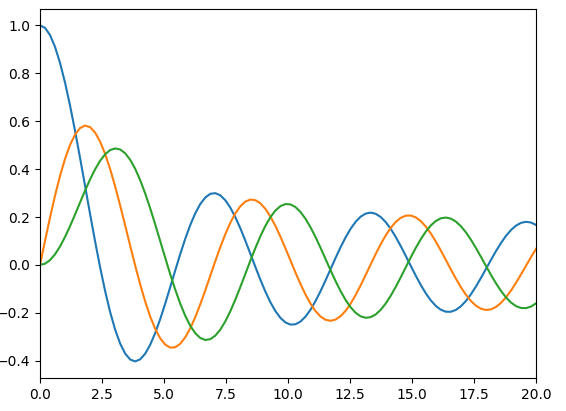
def Bessel(theta, m, x):

return (1/pi)\*(cos(m\*theta - x\*sin(theta)))

def J(m, x):

return quad(Bessel, 0, pi, args=(m, x))[0]

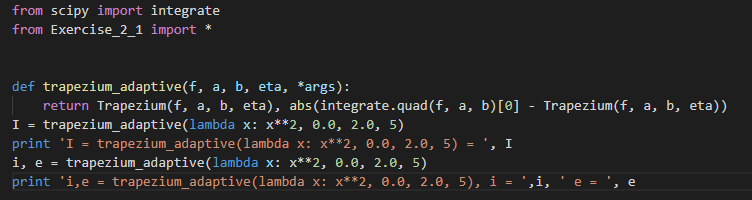
1. Plot J0(x), J1(x) and J2(x) over the range from x = 0 to x = 20



1. Fdf

**EXERCISE 2.5: ERRORS ON INTEGRALS AND ADAPTIVE INTEGRATION**

1. Write a user defined function trapezium\_adaptive(f, a, b, eta, \*args)

Code :

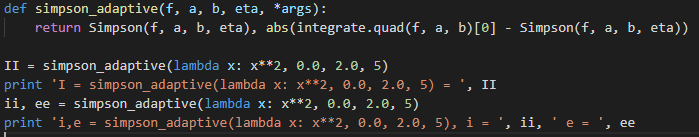
Result :



b)

1. Write a user-defined function simpson\_adaptive(f, a, b, eta, \*args)

Code :

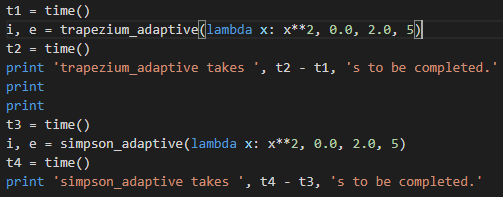


c)

Result :



1. Compare the time it takes for the two functions to calculate a given integral function.

Code :

 Result :