Name:. Ravneet kaur

CollegeRollNo:. 2020PHY1064

AimofComputationalExperiment

AIM : TO FIND ENERGY STATE OF A FINITE POTENTIAL WELL USING GARRET'S METHOD.

#ALGORITHM

The most significant method in Quantum Mechanics is the Garrets Method.

Firstly in the given problem we found the value of Energy denoted by E, followed by finding the value of delta and from these two we found the value of the length. In general for different energy levels, we found the corresponding values of delta and lengths. For that initially we checked whether the value of the potential is positive or not. If it is positive, it will proceed forward else it will return a message, "V should be greater than E". Then we are appending all the new values of energy, delta and lengths and after finding these again we are getting the new values of energies. We are also calculating the $\Delta E = En(i+1) - En(i)$. Then we have to find the ratio of the energy given by $r = \Delta E(i) / En(i)$. If the value of this ratio is greater than 0.01 then we will replace i by i+1 and the loop will again work else we will return the output value of the Enegry Calculated denoted by En(i+1).

TheProgramCode

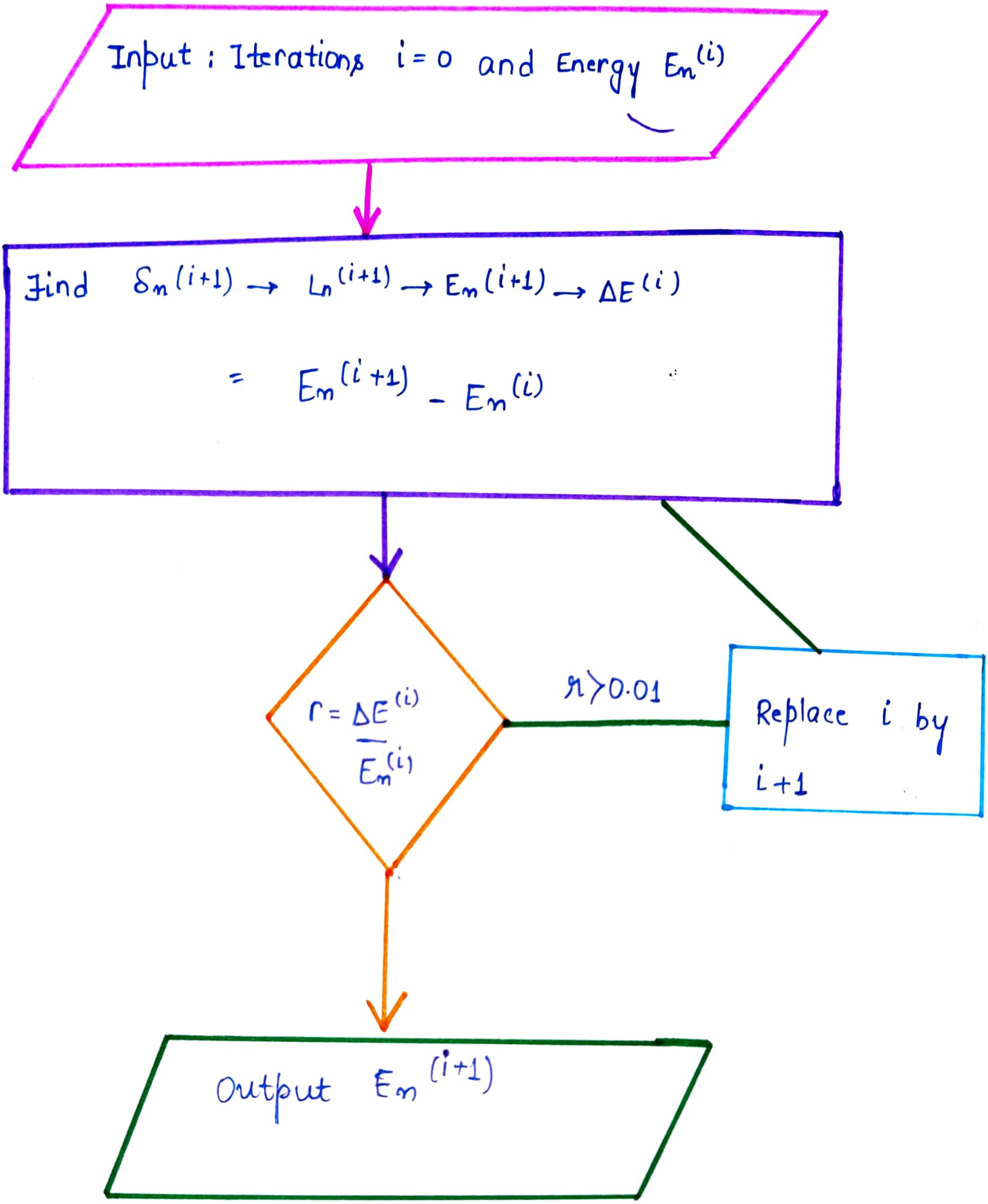
```
#Importing Modules
import numpy as np
import matplotlib.pyplot as plt
from scipy.constants import hbar,e
import sys
import pandas as pd
#Function for Energy
def Energy(n,m,L):
    return n*n*(np.pi**2)*hbar**2/(2*m*L**2)#Function for Delta
def delta(En):
    if V>En: #We are checking wheather the value of the V is positive or not
        pass
    else:
        sys.exit("V should be greater than E")
    delta = hbar/np.sqrt(2*m*(V - En))
    return delta
#Function for the Length
def Ln(L,delta):
    return L+2*delta#Now Defining a new function for iterations and Tolernace
def Function(En,n,m,V,L,Tol):#Now we will make empty lists in order to append the values
    En_array = [En]
    Ln_array = []
    i_array = []
    r_array = []
    Del_array = []
    for i in range(1,1000):
        i_array.append(i) #all value of i
        Delta= delta(En) #new delta w.r.t new Energy
        Del_array.append(Delta) #appended all values of new Delta
        Lnew = Ln(L,Delta) #New Value of Ln
        Ln_array.append(Lnew) #appended new length in the empty list
        En = Energy(n,m,Lnew) #Energy
        En_array.append(En) #Appended the new Energy values in En_array
        del_E = abs(En_array[-1]-En_array[-2])
        #Here del_E means the absolute difference b/w the last second last last value of Energy
        r = del_E/En_array[-2]#ration of Delta by the second last value of Energy
        r_array.append(r)#appended the values of r in the empty list r_array
        '''Now we will check the condition for iteration with the tolernace '''
        if r<Tol:  #Here if the value of r is less than the tolerance then it would break and the loop will again have to run
            break
        else:  #else it will continue to run the loop since the value of r>Tol
            continue
    return En_array,i_array,r_array,i
n = float(input("Enter the value of n: "))
Len = float(input("Enter the value of L: "))
L = Len*10**-10 #E = mc^2
r_m = float(input("Enter the value of r_m: "))
m = (r_m*10**6*1.6*10**-19)/(3*10**8)**2 #kg
Pot = float(input("Enter the value of V: "))
V = Pot*e
T = float(input("Enter the value of Tolerance: "))
Tol = T*10**-10
#Plotting for different values of n
for n in [1,2,3]:
   En = Energy(n,m,L)
   En_array,i_array,r_array,i = Function(En,n,m,V,L,Tol)
   plt.plot([0]+i_array,En_array,label = "E_{0}".format(n),marker="o")
   plt.scatter([0]+i_array,En_array)
plt.legend()
plt.title("Energy vs No. of Iterations")
plt.grid()
plt.xlabel("No. of Iterations I")
plt.ylabel("Energy Joules $E_n$")
plt.show()
print("Given Tolerance :", Tol)
print("Total number of iteration :", i)
```

```
#PLOTTING
plt.plot(i_array,r_array, c="red")
plt.scatter(i_array,r_array)
plt.title("Ratio of Energy vs No. of Iterations")
plt.grid()
plt.xlabel("No. of Iterations I")
plt.ylabel("Ratio = $\Delta$ E/ $E_{i}$")
plt.show()
#Table
data ={"Energy (New)":En_array}
print(pd.DataFrame(data))
```

TheProgramCode

TheProgramCode

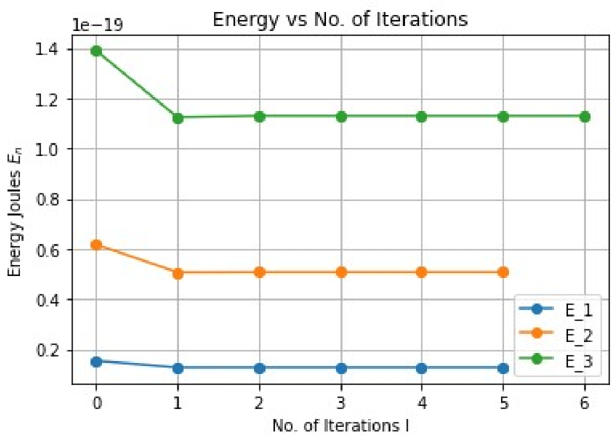
ALGORITHM→



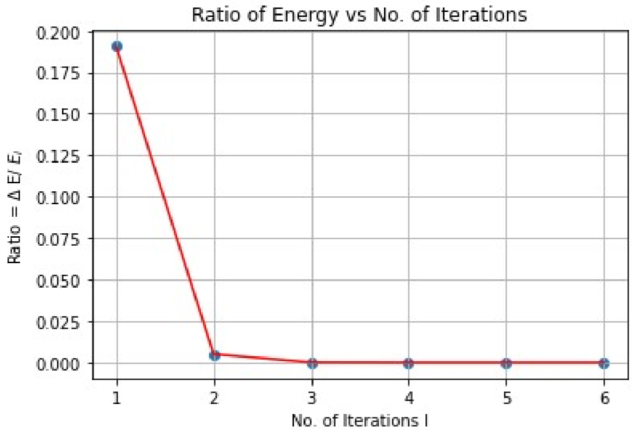
$$\boxed{E = \frac{n^2 \pi^2 h^2}{2mL^2}} \rightarrow \boxed{\delta_n = \frac{h}{\sqrt{2m(V - E_n)}}} \rightarrow$$

$$\boxed{L = L + 2\delta_n}$$

```
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03)
[MSC v.1928 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more
information.

IPython 7.33.0 -- An enhanced Interactive Python.

C:
\Users\hp\AppData\Local\Programs\Spyder\pkgs\numpy\_distribu
tor_init.py:30: UserWarning: loaded more than 1 DLL from
.libs:
C:
\Users\hp\AppData\Local\Programs\Spyder\pkgs\numpy\.libs\lib
openblas.EL2C6PLE4ZYW3ECEVIV3OXXGRN2NRFM2.gfortran-
win_amd64.dll
C:
\Users\hp\AppData\Local\Programs\Spyder\pkgs\numpy\.libs\lib
openblas.QVLO2T66WEPI7JZ63PS3HMOHFEY472BC.gfortran-
win_amd64.dll
  warnings.warn("loaded more than 1 DLL from .libs:"

In [1]: runcell(0, 'C:/Users/hp/Desktop/QUANTUM PHYSICS/SEM
5 CODES/QM Assignment 1.py')
Enter the value of n: 1
Enter the value of L: 20
Enter the value of r_m: 0.5
Enter the value of V: 4
Enter the value of Tolerance: 10
```




```
Given Tolerance : 1e-09
Total number of iteration : 6
```




```
   Energy (New)
0  1.389175e-19
1  1.124169e-19
2  1.129925e-19
3  1.129804e-19
4  1.129807e-19
5  1.129807e-19
6  1.129807e-19
```

In [2]: