COMPUTATIONAL LAB FILE
SGTB KHALSA COLLEGE
UNIVERSITY OF DELHI

Statistical Mechanics

Teachers Remarks

Signature

01/14/2023 00:03:32

Class :	Semester 6 BSc H Physics	Paper :	Statistical Mechanics
Roll number :	2020PHY1122	Name :	GAURAV CHANDRA

Aim :

study of ensembles for coins tossing experiment

Code :

```
'''
Roll No.: 2020PHY1122
Name: Gaurav Chandra
'''
```

```
import numpy as np
import math
import pandas as pd
import matplotlib.pyplot as plt
import random
```

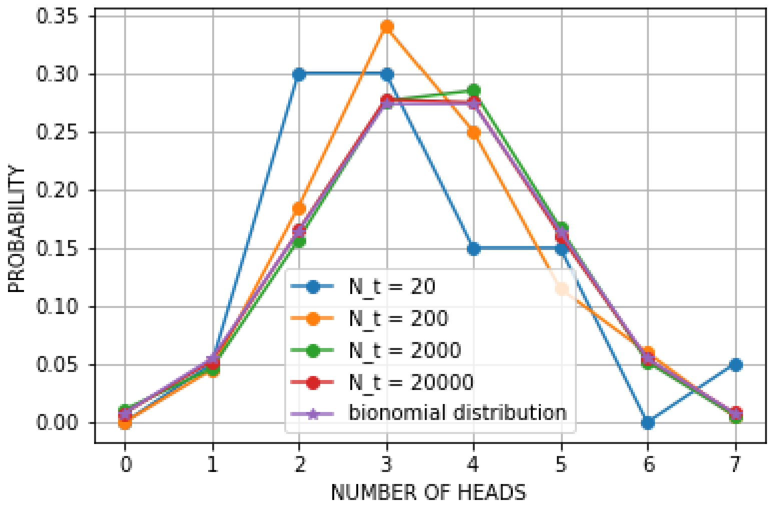
```
def Generator(N_c, N_t):

  out_arr = []   #outcomes_array

  for i in range(N_t):
    out = []  #outcomes

    for j in range(N_c):

      out.append(random.randint(0,1))
    out_arr.append(out)
  n_heads = []  # counting the number of heads and tails
  for n in out_arr:
    n_heads.append(sum(n))
  n_heads = np.array(n_heads)
  n_tails = N_c - n_heads
  freq_count = []  # frequency count of macro-states
  for i in range(N_c + 1):
    freq_count.append(list(n_heads).count(i))
  freq_count = np.array(freq_count)
  probability = freq_count/N_t
  binomial_distri_prob = [] # data from bionomial distribtion
  Nc_arr = np.arange(0, N_c+1)
  for i in range(len(Nc_arr)):
    binomial_distri_prob.append(math.comb(N_c, i)/(2**N_c))
  p = []   # calculating p and q
  for i in range(len(n_heads)):
    p.append(np.sum(n_heads[:i+1])/((i+1)*N_c)
  p = np.array(p)
  q = 1-p
  table = pd.DataFrame({'Trials': np.arange(
    1, N_t + 1), 'Outcomes': out_arr, 'No. of Heads': n_heads, 'No. of Tails':
n_tails, 'p': p, 'q': q})
  table.set_index('Trials', inplace=True)
  return table, probability, binomial_distri_prob
```

```
# plot1

N_c = 7
N_t = 20
while N_t <= 20000:

  y = Generator(N_c, N_t)[1]
  x = np.arange(0, N_c+1)

  plt.plot(x, y, label=f'N_t = {N_t}', marker='o')
  N_t *= 10

y_bd = Generator(N_c, N_t)[2]
plt.plot(x, y_bd, label='bionomial distribution', marker='*')
plt.legend()
plt.grid()
plt.xlabel('NUMBER OF HEADS')
plt.ylabel('PROBABILITY')
plt.savefig('PLOT1_1122')
plt.title('TRIALS VARIATION PLOT')
plt.show()
```

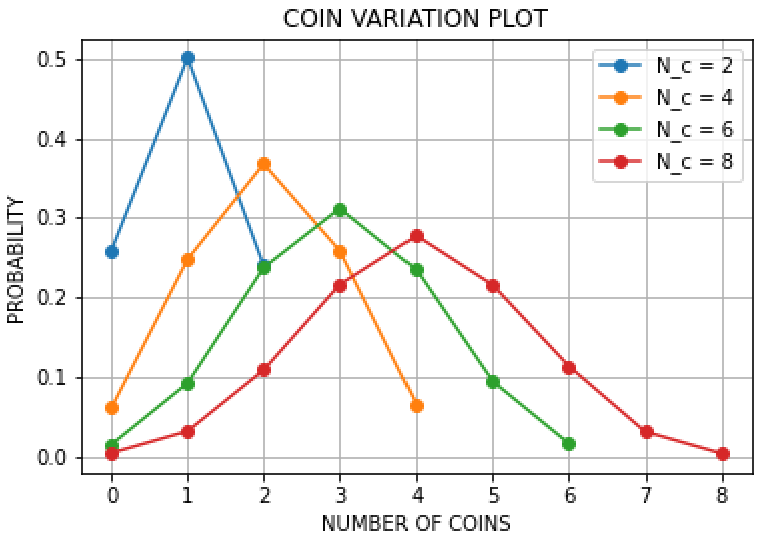
```
# plot2

N_t = 20000
for coins in range(2, 10, 2):

  x = np.arange(0, coins+1)
  y = Generator(coins, N_t)[1]

  plt.plot(x, y, label=f'N_c = {coins}', marker='o')

plt.legend()
plt.grid()
plt.xlabel('NUMBER OF COINS')
plt.ylabel('PROBABILITY')
plt.title('COIN VARIATION PLOT')
plt.savefig('PLOT2_1210')

plt.show()
```

```
# plot3

N_c = 3
N_t = 10000

data = Generator(N_c, N_t)[0]
y1 = data['p'].to_numpy()
y2 = data['q'].to_numpy()

x = np.arange(0, N_t)

plt.plot(x, y1, label='p')
plt.plot(x, y2, label='q')
plt.legend()
plt.grid()
plt.xlabel('NUMBER OF TRIALS')
plt.ylabel('p, q')
plt.title('CUMULATIVE PLOT')
plt.savefig('PLOT3_1122')

plt.show()
```

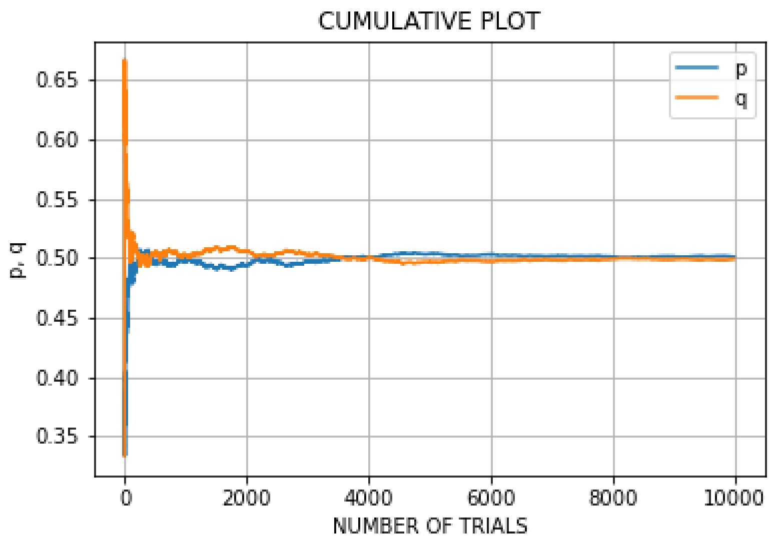
PLOTS :



PROBABILITY
NUMBER OF HEADS
N_t = 20
N_t = 200
N_t = 2000
N_t = 20000
bionomial distribution



COIN VARIATION PLOT
PROBABILITY
NUMBER OF COINS
N_c = 2
N_c = 4
N_c = 6
N_c = 8



CUMULATIVE PLOT
p, q
NUMBER OF TRIALS
p
q

Comments :

COMPUTATIONAL LAB FILE
SGTB KHALSA COLLEGE
UNIVERSITY OF DELHI

Statistical Mechanics

TEACHERS REMARKS

SIGNATURE

01/18/2023 16:51:57

CLASS :	Semester 6 BSc H Physics	PAPER :	Statistical Mechanics
ROLL NUMBER :	2020PHY1122	NAME :	GAURAV CHANDRA

AIM :

TO UNDERSTAND THE DISTRIBUTION FUNCTIONS LIKE MAXWELL-BOLTZMANN ,BOSE EINSTEIN AND FERMI DIRAC DISTRIBUTIONS

CODE :

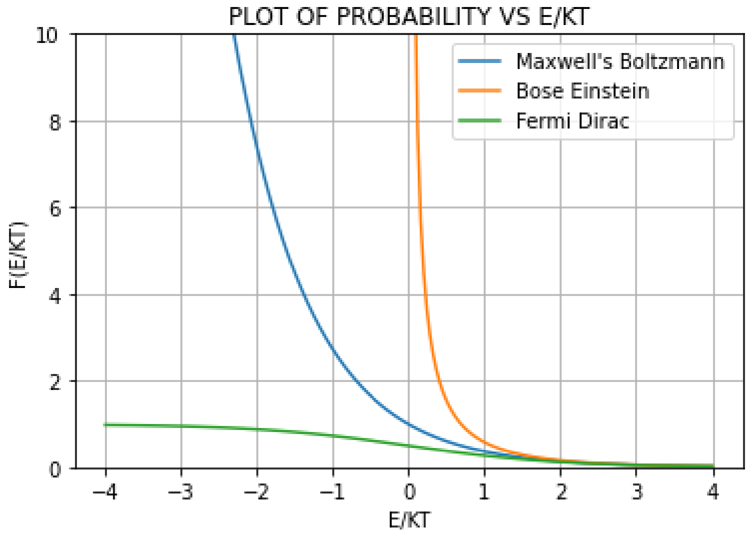
```
'''
Roll No.: 2020PHY1122
Name: Gaurav Chandra
'''

import numpy as np
import matplotlib.pyplot as plt

k = 8.6173 * 10**(-5)
X = np.linspace(-4,4,100)
alpha = 0
X1 = np.linspace(alpha+10**(-5),4,100)
def f_mb(X):

   Y = np.exp(-X)
   return Y

def f_be(X,alpha):

   Y = 1/(np.exp(alpha)*np.exp(X) - 1)
   return Y

def f_fd(X,alpha):

   Y = 1/(np.exp(alpha)*np.exp(X) + 1)
   return Y


#plot 1
plt.plot(X,f_mb(X),label = "Maxwell's Boltzmann")
plt.plot(X1,f_be(X1,alpha),label = "Bose Einstein")
plt.plot(X,f_fd(X,alpha),label = "Fermi Dirac")
plt.ylim([0,10])
plt.xlabel("E/KT")
plt.ylabel("F(E/KT)")
plt.title("PLOT OF PROBABILITY VS E/KT")
plt.grid()
plt.legend()
plt.show()
```

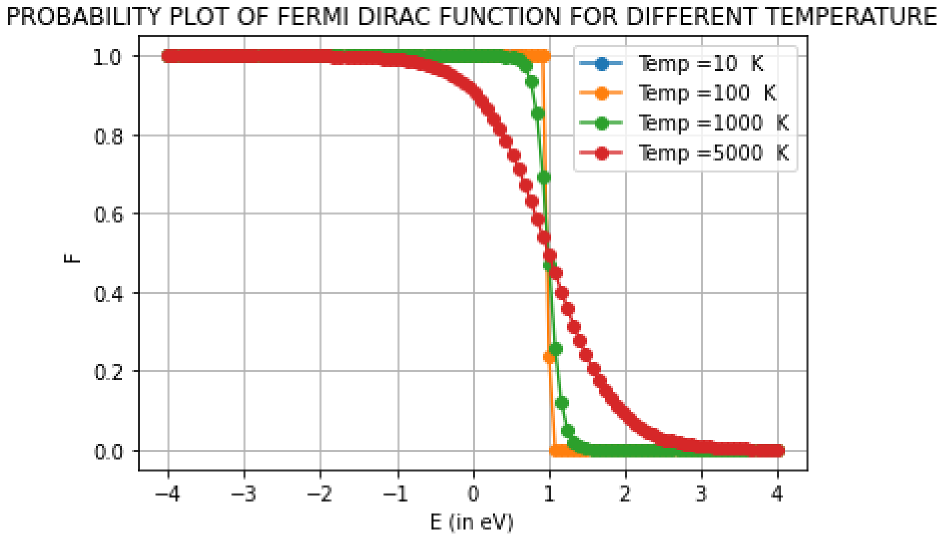
```
#plot 2
e_f = 1  #eV
T = [10,100,1000,5000]
e = np.linspace(-4,4,100)

for i in T:
    plt.plot(e,f_fd(e/(k*i) , alpha = -e_f/(k*i)),marker = 'o',label = 'Temp ='+str(i)+'  K')

plt.xlabel("E (in eV)")
plt.ylabel("F")
plt.legend()
plt.title("PROBABILITY PLOT OF FERMI DIRAC FUNCTION FOR DIFFERENT TEMPERATURE")
plt.grid()
plt.show()
```

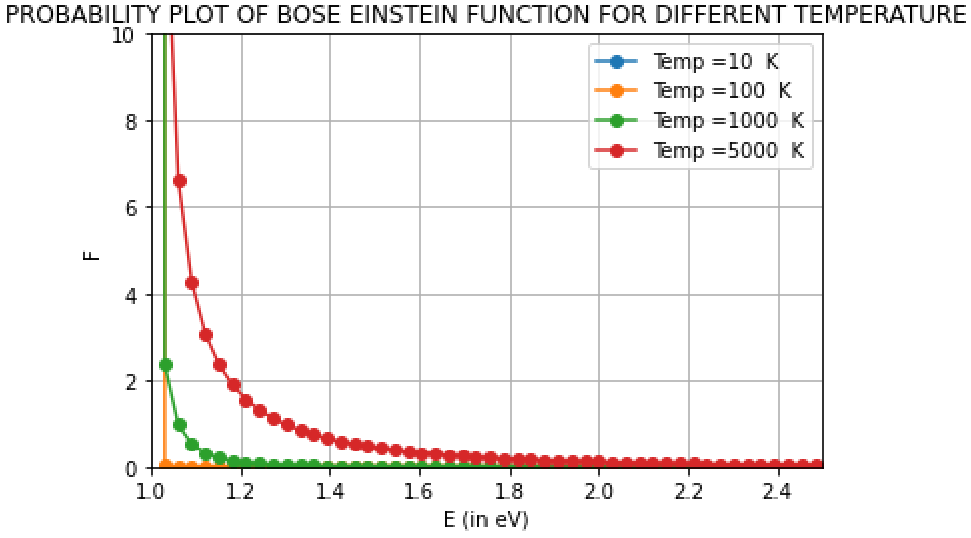
```
#plot 3
U = 1  #eV
T = [10,100,1000,5000]
e = np.linspace(U+10**(-5),4,100)


for i in T:
    x = e/(k*i)
    alpha = -U/(k*i)

    plt.plot(e,f_be(x,alpha),marker = 'o',label = 'Temp ='+str(i)+'  K')

plt.xlabel("E (in eV)")
plt.ylabel("F")
plt.legend()
plt.ylim([0,10])
plt.xlim([U,2.5])
plt.title("PROBABILITY PLOT OF BOSE EINSTEIN FUNCTION FOR DIFFERENT TEMPERATURE")
plt.grid()
plt.show()
```

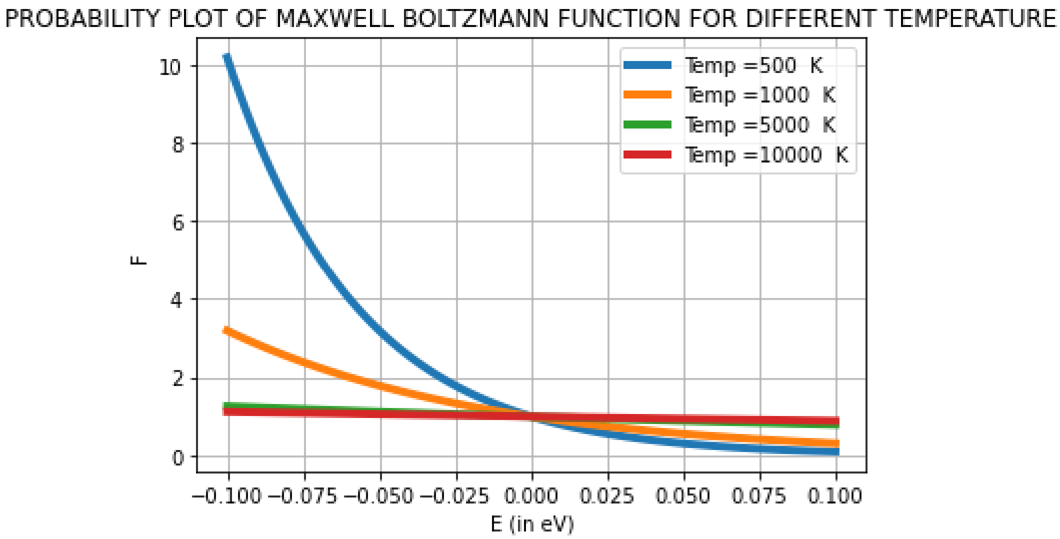
```
#plot 4

T = [500,1000,5000,10000]
e = np.linspace(-0.1,0.1,100)

for i in T:
   X = e/(k*i)
   plt.plot(e,f_mb(X),linewidth=4,label = 'Temp ='+str(i)+'  K')

plt.xlabel("E (in eV)")
plt.ylabel("F")
plt.legend()
plt.title("PROBABILITY PLOT OF MAXWELL BOLTZMANN FUNCTION FOR DIFFERENT TEMPERATURE")
plt.grid()
plt.show()
```

PLOTS :	


PLOT OF PROBABILITY VS E/KT
F(E/KT)
E/KT
Maxwell's Boltzmann
Bose Einstein
Fermi Dirac


PROBABILITY PLOT OF FERMI DIRAC FUNCTION FOR DIFFERENT TEMPERATURE
F
E (in eV)
Temp =10 K
Temp =100 K
Temp =1000 K
Temp =5000 K


PROBABILITY PLOT OF BOSE EINSTEIN FUNCTION FOR DIFFERENT TEMPERATURE
F
E (in eV)
Temp =10 K
Temp =100 K
Temp =1000 K
Temp =5000 K


PROBABILITY PLOT OF MAXWELL BOLTZMANN FUNCTION FOR DIFFERENT TEMPERATURE
Temp =500 K
Temp =1000 K
Temp =5000 K
Temp =10000 K
F
E (in eV)
10
8
6
4
2
0
−0.100
−0.075
−0.050
−0.025
0.000
0.025
0.050
0.075
0.100

Comments :

Computational Lab File

1 message

Google Forms <forms-receipts-noreply@google.com> Tue, 28 Feb, 2023 at 5:51 pm
To: chandra.gaurav2018@gmail.com

Google Forms

Thanks for filling out Computational Lab File

Here's what was received.

Computational Lab File

Email *

chandra.gaurav2018@gmail.com

Class *

Semester 6 BSc H Physics

Paper *

Statistical Mechanics

Name *

GAURAV CHANDRA

Roll No. *

2020PHY1122

Aim *

The Laws of Radiation -
Stefan-Bolztmann Law (Radiant Flux)

Code1 *

(about 10 lines)

```
#name:gaurav chandra
#name : 2020phy1122

import matplotlib.pyplot as plt
import numpy as np
from scipy import integrate
from scipy import stats


h = 6.626*10**(-34)
c = 3*10**(8)
k = 8.61733 *10**(-5)* 1.6 *10**(-19)


x = np.linspace(0.01,12,100)

def f_p(a):
   return (a**3)/(np.exp(a)-1)

plt.plot(x,f_p(x))
plt.xlabel("x")
plt.ylabel("f_p(x) = x**3/(e**x -1)")
plt.grid()
plt.title("PLOT OF F_p VS X FOR PEAK")
plt.savefig("fig_1_a5")
plt.show()
```

Code2

(about 10 lines)

```
i = list(f_p(x)).index(max(f_p(x)))
xp = x[i]
print("the value of peak(dimentionless) is : ",xp)

b = (h*c)/(k*xp)
print("the value of b is : ",b)


#part b

inte = integrate.quad(f_p,0.1,100)[0]
print("The value of integration obtained using python is ",inte)

inte_cal = (np.pi**4 /15)

print("The value of integration obtained using the numerical method is ",inte_cal)

def C(T) :
   l = h*c/(k*T)
   C = 8*np.pi*(k*T)/l**3
   return C

Temp = np.arange(100,10000,500)
C_t = C(Temp)
#print(C_t)

U = inte*(C_t)
F = c*U/4
```

Code3

```
plt.plot(Temp,F,'o-')
plt.xlabel("TEMPERATURE")
plt.ylabel("RADIANT FLUX")
plt.title("PLOT OF RADIANT FLUX VS TEMPERATURE")
plt.grid()
plt.savefig("fig_2_a5")
plt.show()


plt.plot(np.log(Temp),np.log(F),'o-')
plt.xlabel("TEMPERATURE")
plt.ylabel("RADIANT FLUX")
plt.title("LOG PLOT OF RADIANT FLUX VS TEMPERATURE")
plt.xlim([-1,10])
plt.ylim([-20,20])
plt.grid()
```

```
plt.savefig("fig_3_a5")
plt.show()

result = stats.linregress(np.log(Temp),np.log(F))
print("THE SLOPE IS :",result.slope,"AND THE INTERCEPT IS :",result.intercept)

sigma_cal = c*8*np.pi**5*k**4/(4*15*(c*h)**3)
sigma_num = np.exp(result.intercept)

print("THE VALUE OF SIGMA CALCULATED FROM STEFAN BLOTZMANN LAW IS :",sigma_cal," Wm**-2 * T**-4")
print("THE VALUE OF SIGMA CALCULATED FROM THE PLOT IS :",sigma_num," Wm**-2 * T**-4")
if np.round(sigma_num,10) == np.round(sigma_cal,10):
   print("AS THE SIGMA CALCULATED AND SIGMA FROM PLOT ARE EQUAL,SO THE STEFAN BOLTZMANN LAW IS PROVED")
```

Code4

(about 10 lines)

```
#to calculate mean point



a = 0.001
b = 12
x = np.linspace(a,b,11)
for i in x:
   int_l = integrate.quad(f_p,a,i)[0]
   int_r = integrate.quad(f_p,i,b)[0]

   if abs(int_l - int_r)/int_r <= 0.1:
      x_mean = i
      print("THE X VALUE WHICH DIVIDES THE AREA IN EQUAL PARTS IS :",x_mean)
      break



b_mean = (h*c)/(k*x_mean)
print("THE MEAN B VALUE IS ;",b_mean)
```

Code5

(about 10 lines)

Code6

(about 10 lines)

Plot1

Submitted files

fig_1_a5 - Gaurav Chandra.png

Plot2

Submitted files

fig_2_a5 - Gaurav Chandra.png

Plot3

Submitted files

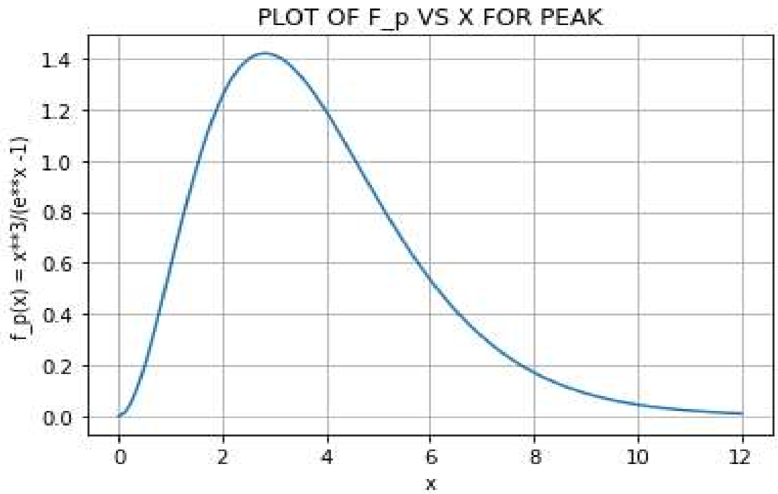
fig_3_a5 - Gaurav Chandra.png

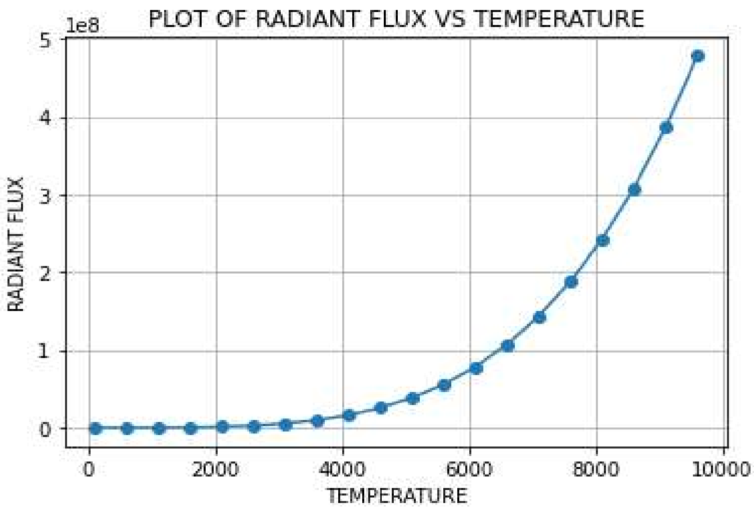
Plot4

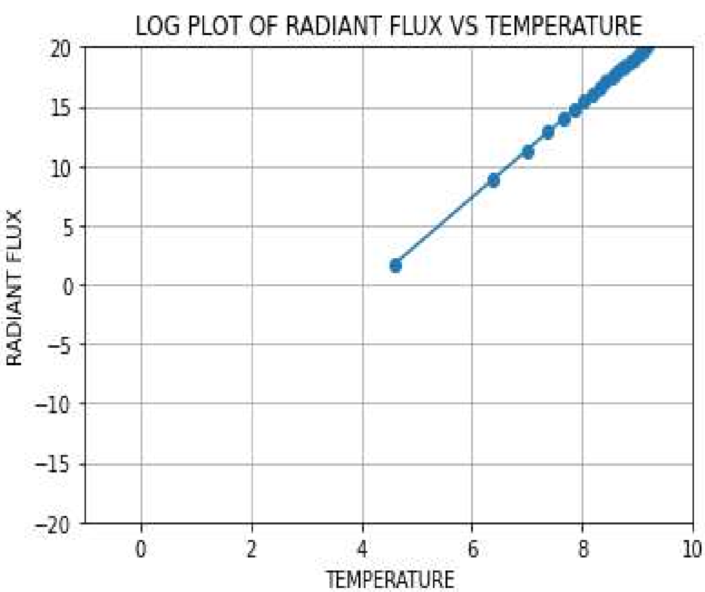
No files submitted

Comments

OUTPUT:--
the value of peak(dimentionless) is : 2.7955555555555556
the value of b is : 0.005157174798518598
The value of integration obtained using python is 6.493618402286659
The value of integration obtained using the numerical method is 6.493939402266828
THE SLOPE IS : 4.0 AND THE INTERCEPT IS : -16.69226675748456
THE VALUE OF SIGMA CALCULATED FROM STEFAN BLOTZMANN LAW IS : 5.6319932389868827e-08 Wm**-2 * T**-4
THE VALUE OF SIGMA CALCULATED FROM THE PLOT IS : 5.6317148456102276e-08 Wm**-2 * T**-4
AS THE SIGMA CALCULATED AND SIGMA FROM PLOT ARE EQUAL,SO THE STEFAN BOLTZMANN LAW IS PROVED
THE X VALUE WHICH DIVIDES THE AREA IN EQUAL PARTS IS : 3.6007
THE MEAN B VALUE IS ; 0.004003990518224171


PLOT OF F_p VS X FOR PEAK
f_p(x) = x**3/(e**x -1)
x


PLOT OF RADIANT FLUX VS TEMPERATURE
1e8
RADIANT FLUX
TEMPERATURE


LOG PLOT OF RADIANT FLUX VS TEMPERATURE
RADIANT FLUX
TEMPERATURE

Computational Lab File

1 message

Google Forms <forms-receipts-noreply@google.com> Thu, 9 Mar, 2023 at 8:13 pm
To: chandra.gaurav2018@gmail.com

Google Forms

Thanks for filling out Computational Lab File

Here's what was received.

Computational Lab File

Email *

chandra.gaurav2018@gmail.com

Class *

Semester 6 BSc H Physics

Paper *

Statistical Mechanics

Name *

GAURAV CHANDRA

Roll No. *

2020PHY1122

Aim *

to analyse plots of partition function,internal energy,entropy,population density plots for high and low temperature

Code1 *

(about 10 lines)

```
#name : gaurav chandra
#rollno : 2020phy1122

import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

k = 8.617 * 10**(-5) # eV/K

def Z(g,e,T):
    n = len(e)
    part = []

    for j in (T):

        z = 0
        for i in range(n):
            z =z+ g[i] * np.exp(-e[i]/(k*j))
        part.append(z)
    return part

def frac(g,e,T):
    n = len(e)
    FRAC = np.zeros(n*len(T) ).reshape(n, len(T))
    z = Z(g,e,T)
    for i in range(len(T)):
        for j in range(n):
            f1 = (g[j]*np.exp(-e[j]/(k*T[i]))) / z[i]
            FRAC[j][i] = f1
    return FRAC
```

```
T1 = np.linspace(0.0001,5000,1000)
T2 = np.linspace(5000,10**6,1000)
```

Code2

(about 10 lines)

```
#2level
e = [0,1]
g=[1,1]
#3level
e_3 = [0,1,2]
g_3 = [1,1,1]
plt.subplot(2,2,1)
plt.plot(T1,Z(g,e,T1),c='r',label="2lvl and low T")
plt.xlabel("TEMPERATURE")
plt.ylabel("Z")
plt.grid()
plt.legend()

plt.subplot(2,2,2)
plt.plot(T2,Z(g,e,T2),c='y',label="2lvl and high T")
plt.xlabel("TEMPERATURE")
plt.ylabel("Z")
plt.grid()
plt.legend()

plt.subplot(2,2,3)
plt.plot(T1,Z(g_3,e_3,T1),c='b',label="3lvl and low T")
plt.xlabel("TEMPERATURE")
plt.ylabel("Z")
plt.grid()
plt.legend()

plt.subplot(2,2,4)
plt.plot(T2,Z(g_3,e_3,T2),c='violet',label="3lvl and high T")
plt.xlabel("TEMPERATURE")
plt.ylabel("Z")
plt.grid()
```

Code3

```
plt.suptitle("PLOT OF Z VS TEMP")
plt.savefig("ass6_1.png")
plt.legend()
plt.show()

#fractional plot
e_mid = 1/len(e)
```

```
e_mid_3 = 1/len(e_3)
E_mid ,E_mid_3= np.full( shape = len(T2) ,fill_value = e_mid),np.full( shape = len(T2) ,fill_value = e_mid_3)

plt.subplot(2,2,1)
for i in range(len(e)):
    plt.plot(T1,frac(g,e,T1)[i], label = "energy = "+str(e[i]) + " eV")

plt.xlabel("TEMPERATURE")
plt.ylabel("N_j / N")
plt.legend(loc=6)
plt.grid()

plt.subplot(2,2,2)
for i in range(len(e)):
    plt.plot(T2,frac(g,e,T2)[i], label = "energy = "+str(e[i]) + " eV")

plt.plot(T2,E_mid,'--')
plt.xlabel("TEMPERATURE")
plt.ylabel("N_j / N")
plt.legend(loc="best")
plt.grid()

plt.subplot(2,2,3)
for i in range(len(e_3)):
    plt.plot(T1,frac(g_3,e_3,T1)[i], label = "energy = "+str(e_3[i]) + " eV")

plt.xlabel("TEMPERATURE")
plt.ylabel("N_j / N")
plt.legend(loc=6)
plt.grid()
plt.subplot(2,2,4)
for i in range(len(e_3)):
    plt.plot(T2,frac(g_3,e_3,T2)[i], label = "energy = "+str(e_3[i]) + " eV")

plt.plot(T2,E_mid_3,'--')
plt.xlabel("TEMPERATURE")
plt.ylabel("N_j / N")
plt.legend(loc="best")
plt.grid()
plt.suptitle("PLOT N_j/N VS TEMPERATURE ")
plt.savefig("ass6_2.png")
plt.show()
```

Code4

(about 10 lines)

```
#INTERNAL ENERGY
population_1 = frac(g,e,T1)
population_2 = frac(g,e,T2)
population_3 = frac(g_3,e_3,T1)
population_4 = frac(g_3,e_3,T2)
U_1,U_2,U_3,U_4 = 0,0,0,0
for i in range(len(population_1)) :
```

```
    U_1 += population_1[i]*e[i]

for i in range(len(population_2)) :
    U_2 += population_2[i]*e[i]

for i in range(len(population_3)) :
    U_3 += population_3[i]*e_3[i]

for i in range(len(population_4)) :
    U_4 += population_4[i]*e_3[i]

plt.subplot(2,2,1)
plt.plot(T1,U_1,c='r',label="2lvl and low T")
plt.xlabel("TEMPERATURE")
plt.ylabel("U/N")
plt.grid()
plt.legend()

plt.subplot(2,2,2)
plt.plot(T2,U_2,c='y',label="2lvl and high T")
plt.xlabel("TEMPERATURE")
plt.ylabel("U/N")
plt.grid()
plt.legend()
plt.subplot(2,2,3)
plt.plot(T1,U_3,c='b',label="3lvl and low T")
plt.xlabel("TEMPERATURE")
plt.ylabel("U/N")
plt.grid()
plt.legend()

plt.subplot(2,2,4)
plt.plot(T2,U_4,c='g',label="3lvl and high T")
plt.xlabel("TEMPERATURE")
plt.ylabel("U/N")
plt.grid()
plt.legend()
plt.savefig("ass6_3.png")
plt.suptitle("U/N VS TEMPERATURE")
plt.show()
```

Code5

(about 10 lines)

```
#ENTROPY
z1 = Z(g,e,T1)
z2 = Z(g,e,T2)
z3 = Z(g_3,e_3,T1)
z4 = Z(g_3,e_3,T2)
N = 1
S1 = N*k*np.log(np.array(z1) / N) + U_1 / T1
S2 = N*k*np.log(np.array(z2) / N) + U_2 / T2
S3 = N*k*np.log(np.array(z3) / N) + U_3 / T1
```

```
S4 = N*k*np.log(np.array(z4) / N) + U_4 / T2

plt.subplot(2,2,1)
plt.plot(T1,S1,c='r',label="2lvl and low T")
plt.xlabel("TEMPERATURE")
plt.ylabel("ENTROPY")
plt.legend()
plt.grid()

plt.subplot(2,2,2)
plt.plot(T2,S2,c='y',label="2lvl and high T")
plt.xlabel("TEMPERATURE")
plt.ylabel("ENTROPY")
plt.legend()
plt.grid()

plt.subplot(2,2,3)
plt.plot(T1,S3,c='b',label="3lvl and low T")
plt.xlabel("TEMPERATURE")
plt.ylabel("ENTROPY")
plt.legend()
plt.grid()

plt.subplot(2,2,4)
plt.plot(T2,S4,c='g',label="3lvl and high T")
plt.xlabel("TEMPERATURE")
plt.ylabel("ENTROPY")
plt.grid()
plt.suptitle("PLOT ENTROPY VS TEMPERATURE")
plt.legend()
plt.savefig("ass6_4.png")
plt.show()

#HELMHOLTZ
F1 = -N*k*np.array(T1) * np.log(np.array(z1))
F2 = -N*k*np.array(T2) * np.log(np.array(z2))
F3 = -N*k*np.array(T1) * np.log(np.array(z3))
F4 = -N*k*np.array(T2) * np.log(np.array(z4))
```

Code6

(about 10 lines)

```
plt.subplot(2,2,1);plt.plot(T1,F1,c='r',label="2lvl and low T")
plt.xlabel("TEMPERATURE");plt.ylabel("HELMHOLTZ FNC")
plt.legend();plt.grid()

plt.subplot(2,2,2)
plt.plot(T2,F2,c='y',label="2lvl and high T")
plt.xlabel("TEMPERATURE");plt.ylabel("HELMHOLTZ FNC")
plt.grid();plt.legend()

plt.subplot(2,2,3)
plt.plot(T1,F3,c='b',label="3lvl and low T")
```

```
plt.xlabel("TEMPERATURE");plt.ylabel("HELMHOLTZ FNC")
plt.legend();plt.grid()

plt.subplot(2,2,4)
plt.plot(T2,F4,c='g',label="3lvl and high T");plt.xlabel("TEMPERATURE")
plt.ylabel("HELMHOLTZ FNC");plt.grid()
plt.legend();plt.suptitle("PLOT F VS TEMP FOR HIGH TEMP")
plt.savefig("ass6_5.png");plt.show()

result_1,result_2 = stats.linregress(T2,F2),stats.linregress(T2,F4)
print("The slope of the plot of F vs T for high temp for 2lvl system is :",result_1.slope)
print("The value obtained for entropy at high temperature :",np.max(S2))
print('')
print("The slope of the plot of F vs T for high temp for 3lvl system is :",result_2.slope)
print("The value obtained for entropy at high temperature :",np.max(S4))
```

Plot1

Submitted files

ass6_1 - Gaurav Chandra.png

Plot2

Submitted files

ass6_2 - Gaurav Chandra.png

Plot3

Submitted files

ass6_3 - Gaurav Chandra.png

Plot4

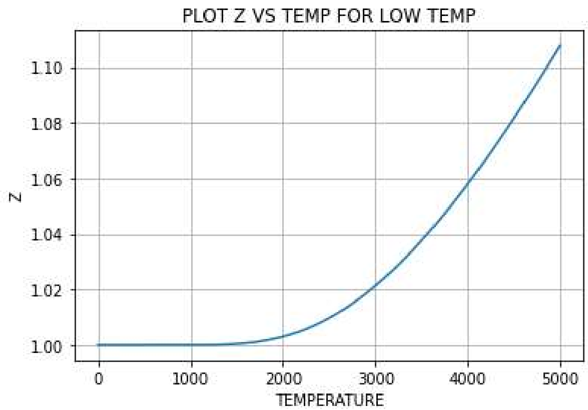
Submitted files

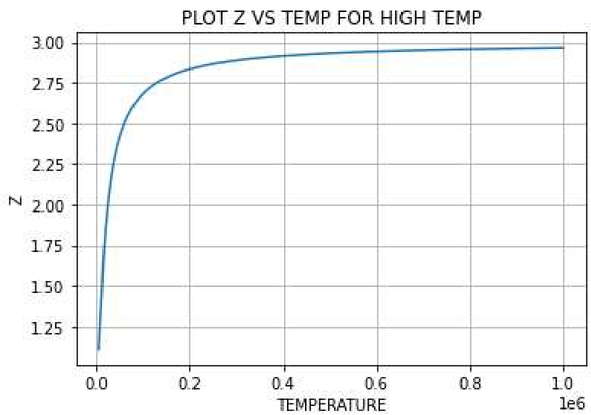
ass6_4 - Gaurav Chandra.png

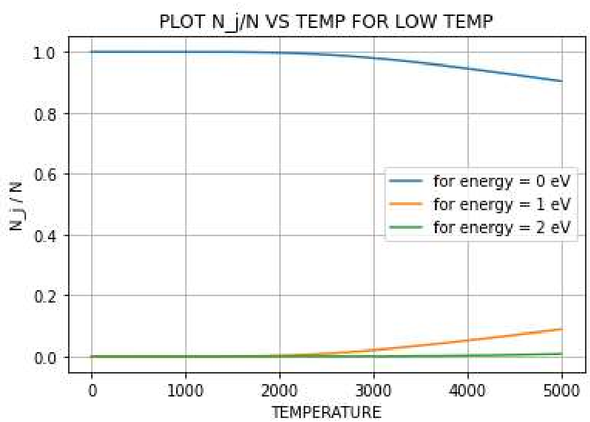
Comments

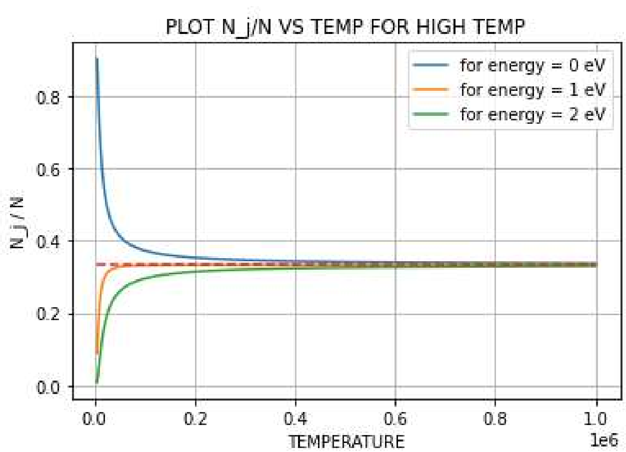
The slope of the plot of F vs T for high temp for 2lvl system is : -5.969921003642963e-05
The value obtained for entropy at high temperature : 5.972704195240474e-05

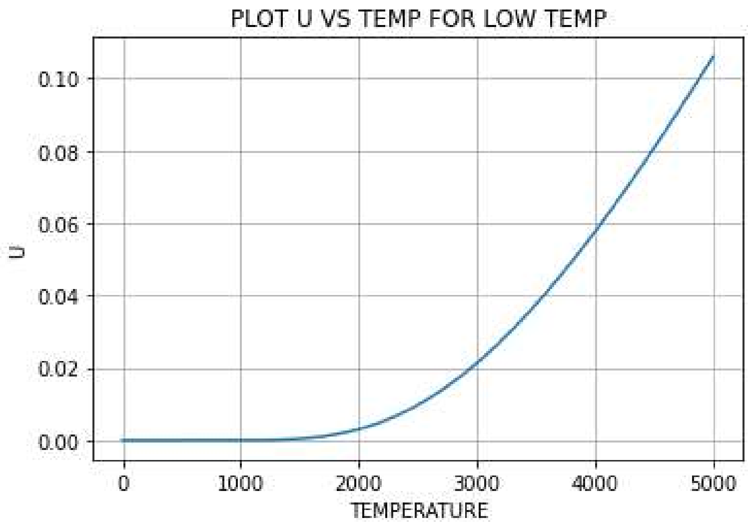
The slope of the plot of F vs T for high temp for 3lvl system is : -9.459123224292288e-05
The value obtained for entropy at high temperature : 9.466355272246002e-05

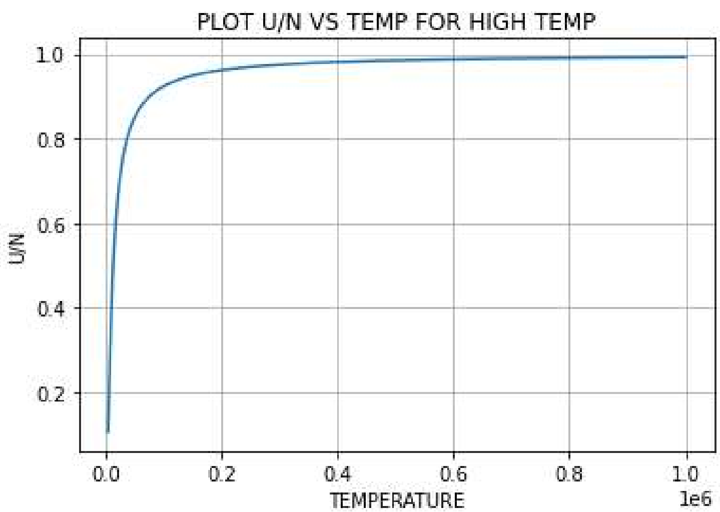

PLOT Z VS TEMP FOR LOW TEMP
Z
TEMPERATURE

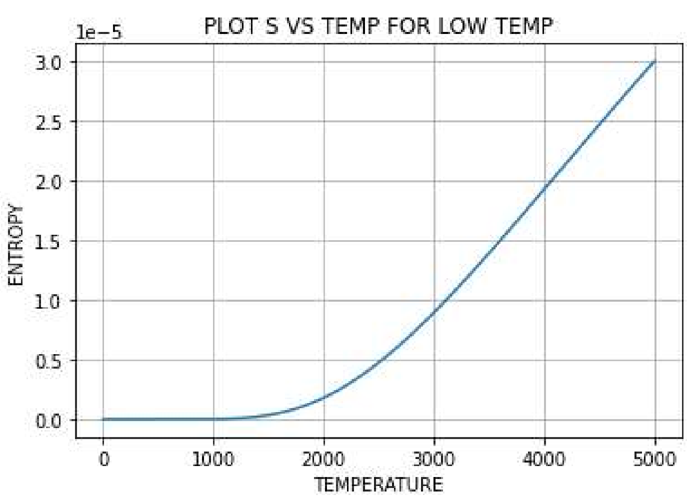

PLOT Z VS TEMP FOR HIGH TEMP
Z
TEMPERATURE
1e6

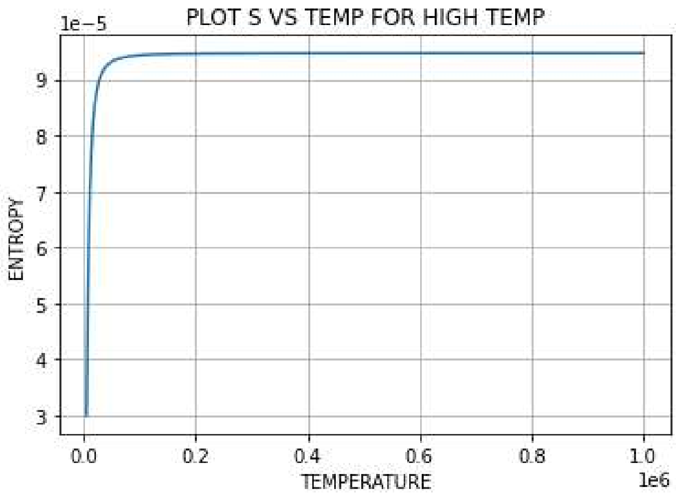

PLOT N_j/N VS TEMP FOR LOW TEMP
N_j / N
TEMPERATURE
for energy = 0 eV
for energy = 1 eV
for energy = 2 eV

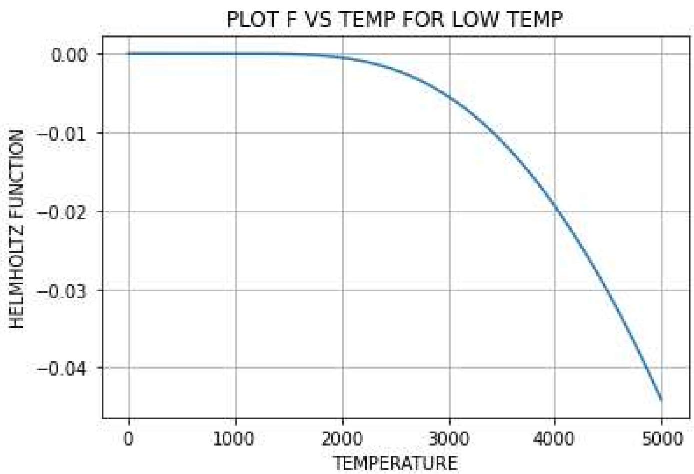

PLOT N_j/N VS TEMP FOR HIGH TEMP
N_j / N
TEMPERATURE
1e6
for energy = 0 eV
for energy = 1 eV
for energy = 2 eV

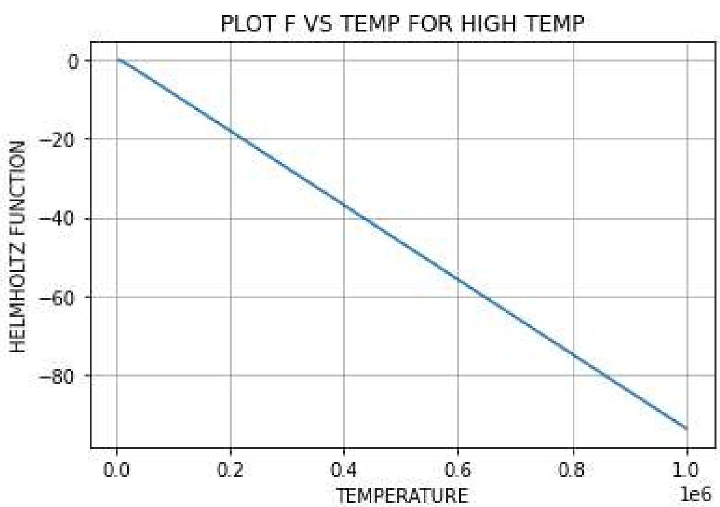

PLOT U VS TEMP FOR LOW TEMP
U
0.00
0.02
0.04
0.06
0.08
0.10
0
1000
2000
3000
4000
5000
TEMPERATURE


PLOT U/N VS TEMP FOR HIGH TEMP
U/N
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
1e6
TEMPERATURE


PLOT S VS TEMP FOR LOW TEMP
1e−5
ENTROPY
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
1000
2000
3000
4000
5000
TEMPERATURE


PLOT S VS TEMP FOR HIGH TEMP
1e−5
ENTROPY
TEMPERATURE
1e6
3
4
5
6
7
8
9
0.0
0.2
0.4
0.6
0.8
1.0


PLOT F VS TEMP FOR LOW TEMP
HELMHOLTZ FUNCTION
TEMPERATURE
0.00
−0.01
−0.02
−0.03
−0.04
0
1000
2000
3000
4000
5000


PLOT F VS TEMP FOR HIGH TEMP
HELMHOLTZ FUNCTION
TEMPERATURE
1e6
0
−20
−40
−60
−80
0.0
0.2
0.4
0.6
0.8
1.0

Computational Lab File
1 message

Google Forms <forms-receipts-noreply@google.com> Tue, 28 Mar, 2023 at 11:36 pm
To: chandra.gaurav2018@gmail.com

Google Forms

Thanks for filling out Computational Lab File

Here's what was received.

Computational Lab File

Email *

chandra.gaurav2018@gmail.com

Class *

Semester 6 BSc H Physics

Paper *

Statistical Mechanics

Name *

GAURAV CHANDRA

Roll No. *

2020PHY1122

Aim *

To study the canonical Ensemble-Maxwell Boltzmann

Code1 *

(about 10 lines)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import integrate
from scipy import stats

def forward_d(x,y):
  derive=[]
  for i in range(len(x)-1):
    dy=y[i+1]-y[i]
    dx=x[i+1]-x[i]
    derive.append(dy/dx)
  return np.array(derive)

k = 1.38*10**(-23)
h = 6.626*10**(-34)
N_a = 6.022*10**(23)
m = 1.6*10**(-27)
V = np.linspace(20*10**(-3),50*10**(-3),50)
T = np.linspace(150,450,50)
matrix = np.zeros(len(V)*len(T)).reshape(len(T),len(V))

for i in range(len(V)):
 for j in range(len(T)):
  v = V[i]
  t = T[j]

  def z(n):
   z = (np.pi/2) * (n**2) * np.exp(-h**2 * n**2/(8*m*v**(2/3)*k*t))
   return z

  I = integrate.quad(z,0,10**(11))[0]
```

```
    matrix[i][j] = I

log_z = np.log(matrix)

#print(matrix)
```

Code2

(about 10 lines)

```
# plot of log z vs temp
fig=plt.figure()
fig.set_figheight(6)
fig.set_figwidth(10)
plt.subplot(1,2,1)
plt.title("plot of log(z) vs T ")
plt.scatter(T,log_z[:,0],label = "for V= "+str(np.round(V[0],3)))
plt.scatter(T,log_z[:,4],label = "for V= "+str(np.round(V[4],3)))
plt.scatter(T,log_z[:,8],label = "for V= "+str(np.round(V[8],3)))
plt.xlabel("T")
plt.ylabel("log(Z)")
plt.grid()
plt.legend(loc='best')

plt.subplot(1,2,2)
plt.title("plot of log(z) vs log(T) ")
plt.scatter(np.log(T),log_z[:,0],label = "for log(V)= "+str(np.round(np.log(V[0]),3)))
plt.scatter(np.log(T),log_z[:,4],label = "for log(V)= "+str(np.round(np.log(V[4]),3)))
plt.scatter(np.log(T),log_z[:,8],label = "for log(V)= "+str(np.round(np.log(V[8]),3)))
plt.xlabel("log(T)")
plt.ylabel("log(Z)")
plt.grid()
```

Code3

```
plt.legend(loc='best')
plt.show()
plt.savefig('fig7_1.png')
# plot of log z vs volume
fig=plt.figure()
fig.set_figheight(6)
fig.set_figwidth(11)
plt.subplot(1,2,1)
plt.title("plot of log(z) vs V ")
plt.scatter(V,log_z[0],label = "for T= "+str(np.round(T[0],3)))
plt.scatter(V,log_z[4],label = "for T= "+str(np.round(T[4],3)))
plt.scatter(V,log_z[8],label = "for T= "+str(np.round(T[8],3)))
plt.xlabel("V")
plt.ylabel("log(Z)")
plt.grid()
plt.legend(loc='best')
```

```
plt.subplot(1,2,2)
plt.title("plot of log(z) vs log(V) ")
plt.scatter(np.log(V),log_z[0],label = "for T= "+str(np.round(T[0],3)))
plt.scatter(np.log(V),log_z[4],label = "for T= "+str(np.round(T[4],3)))
plt.scatter(np.log(V),log_z[8],label = "for T= "+str(np.round(T[8],3)))
plt.xlabel("log(V)")
plt.ylabel("log(Z)")
plt.grid()
plt.legend(loc='best')
plt.show()
plt.savefig('fig7_2.png')
```

Code4

(about 10 lines)

```
# pressure matrix
pressure = []
for i in range(len(T)):
   der = forward_d(V, log_z[i])
   P = N_a*k*T[i] * der
   pressure.append(P)
pressure = np.array(pressure).reshape(len(T),len(V)-1)
#print(pressure)

fig=plt.figure()
fig.set_figheight(6)
fig.set_figwidth(12.5)
plt.subplot(1,2,1)
plt.title("plot of Pressure vs Volume ")
plt.scatter(V[:len(V)-1],pressure[0],label = "for T= "+str(np.round(T[0],3)))
plt.scatter(V[:len(V)-1],pressure[4],label = "for T= "+str(np.round(T[4],3)))
plt.scatter(V[:len(V)-1],pressure[8],label = "for T= "+str(np.round(T[8],3)))
plt.xlabel("V")
plt.ylabel("Pressure")
plt.grid()
plt.legend(loc='best')
```

Code5

(about 10 lines)

```
plt.subplot(1,2,2)
plt.title("plot of Pressure vs Temperature ")
plt.scatter(T,pressure[:,0],label = "for V= "+str(np.round(V[0],3)))
plt.scatter(T,pressure[:,4],label = "for V= "+str(np.round(V[4],3)))
plt.scatter(T,pressure[:,8],label = "for V= "+str(np.round(V[8],3)))
plt.xlabel("Temperature")
plt.ylabel("Pressure")
plt.grid()
plt.legend(loc='best')
plt.show()
```

```
plt.savefig('fig7_3.png')
# energy matrix
cv=[]
for i in range(3):
    energy = []
    der = forward_d(T, log_z[:,i])

    #energy.append(der)
    for j in range(len(T)-1):
        energy.append(k*T[j]**2 *der[j])
    plt.scatter(T[:len(T)-1],energy,label = "for V= "+str(np.round(V[i],4)))
    cv.append(stats.linregress(T[:len(T)-1],energy)[0])

plt.title("plot of Energy vs Temperature ")
plt.xlabel("Temperature")
plt.ylabel("energy")
plt.grid()
plt.legend(loc='best')
plt.show()
plt.savefig('fig7_4.png')


print('the specific heat of this ideal gas obtained is ',np.average(cv))
print('the specific heat calculated using formula is ',1.5*k)
```

Code6

(about 10 lines)

Plot1

Submitted files

fig7_1 - Gaurav Chandra.png

Plot2

Submitted files

fig7_2 - Gaurav Chandra.png

Plot3

Submitted files

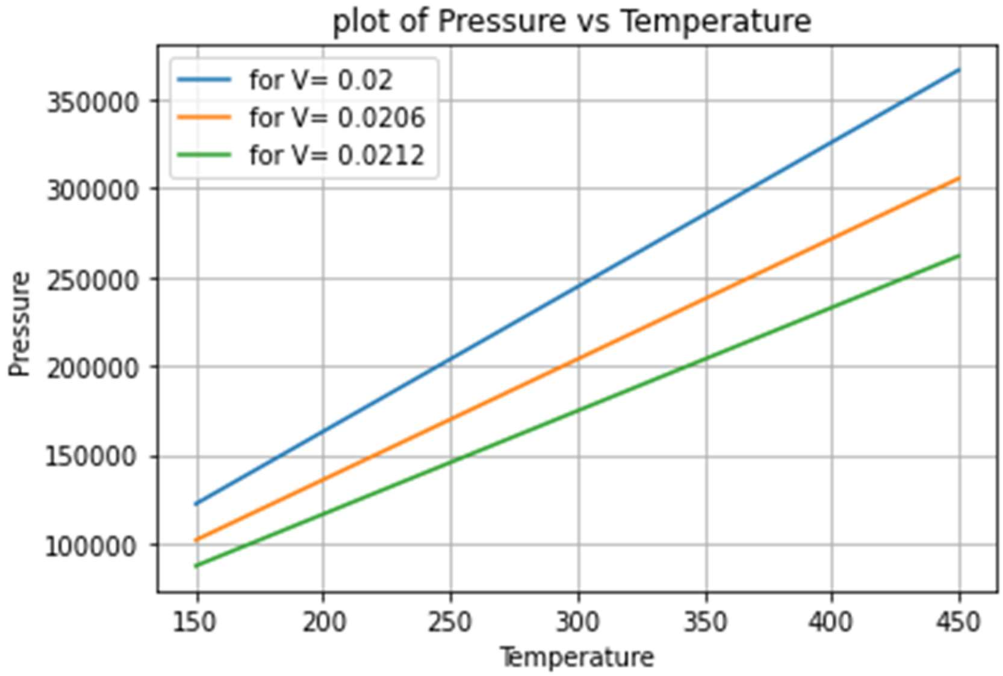
fig7_3 - Gaurav Chandra.png

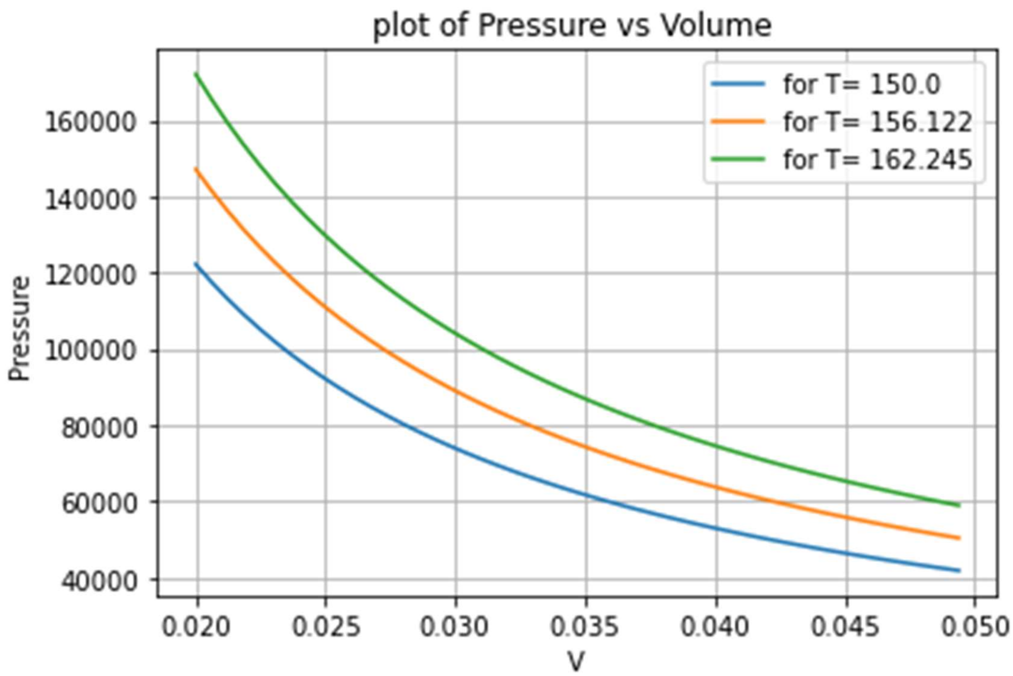
Plot4

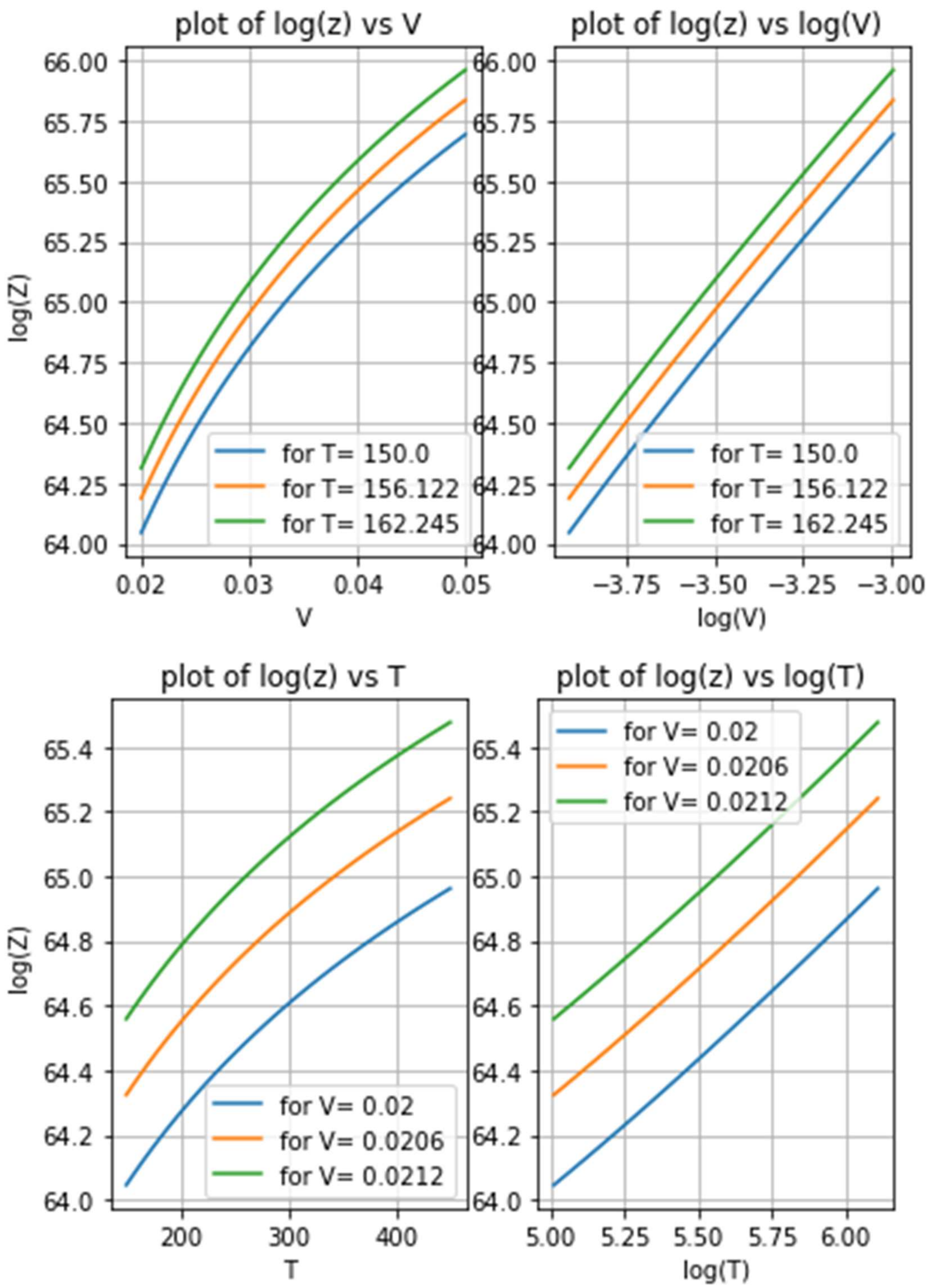
Submitted files

fig7_4 - Gaurav Chandra.png

Comments


plot of Pressure vs Temperature
for V= 0.02
for V= 0.0206
for V= 0.0212
Pressure
Temperature
350000
300000
250000
200000
150000
100000
150
200
250
300
350
400
450


plot of Pressure vs Volume
for T= 150.0
for T= 156.122
for T= 162.245
Pressure
V
160000
140000
120000
100000
80000
60000
40000
0.020
0.025
0.030
0.035
0.040
0.045
0.050



plot of log(z) vs V
plot of log(z) vs log(V)
log(Z)
V
log(V)
for T= 150.0
for T= 156.122
for T= 162.245
plot of log(z) vs T
plot of log(z) vs log(T)
T
log(T)
for V= 0.02
for V= 0.0206
for V= 0.0212

Computational Lab File
1 message

Google Forms <forms-receipts-noreply@google.com> Sat, 8 Apr, 2023 at 6:36 pm
To: chandra.gaurav2018@gmail.com

Google Forms

Thanks for filling out Computational Lab File

Here's what was received.

Computational Lab File

Email *

chandra.gaurav2018@gmail.com

Class *

Semester 6 BSc H Physics

Paper *

Statistical Mechanics

Name *

GAURAV CHANDRA

Roll No. *

2020PHY1122

Aim *

To study the distribution of particles for energies for bosons and fermions and calculate internal energy and specific heat using this.

Code1 *

(about 10 lines)

```
#name : gaurav chandra
#rollno : 2020PHY1122


import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from scipy import integrate
import warnings

warnings.filterwarnings('ignore')
c =3*10**(8)
h = 4.1357*10**(-15) #eV s
k = 8.617 * 10**(-5) # eV/K
N=6.022 *10**(23)
m =1 ; V=1
#for bose-einstein
def f_be(X,alpha):

   Y = 1/(np.exp(alpha)*np.exp(X) - 1)
   return Y

#for fermi-dirac
def f_fd(X,alpha):

   Y = 1/(np.exp(alpha)*np.exp(X) + 1)
   return Y
```

Code2

(about 10 lines)

```
#distribution of particles
def dN_de(e,T,case1,case2):  #case can be either R(relativistic) or NR(non relativistic)
    alpha=-U/(k*T)
    alpha2 =-ef/(k*T)
    X = e/(k*T)
    if case1 == 'R' and case2 =='B':
        return (4*V*np.pi/(h*c)**3)*f_be(X,alpha)*(e**2)

    elif case1 == 'NR' and case2 =='B':
        return (2*V*np.pi*(2*m)**(3/2)/h**3)*f_be(X,alpha)*(e**0.5)

    elif case1 == 'R' and case2 =='F':
        return (4*V*np.pi/(h*c)**3)*f_fd(X,alpha2)*(e**2)

    elif case1 == 'NR' and case2 =='F':
        return (2*V*np.pi*(2*m)**(3/2)/h**3)*f_fd(X,alpha2)*(e**0.5)

    else:
        print('ERROR? plz only enter the valid cases i.e (N,NR,B,F)for(relativistic,non-relativistic,bosons and
fermions)')
```

Code3

```
def internal(T,case1,case2):
    internal_energy = []
    for i in T:
        alpha1=-U/(k*i)
        alpha2 =-ef/(k*i)

        if case1 == 'R' and case2 =='B':

            f=lambda e:(4*V*np.pi/(h*c)**3)*(1/(np.exp(alpha1)*np.exp(e/(k*i)) - 1))*(e**3)
            internal_energy.append(integrate.quad(f,U+0.0001,10)[0])

        elif case1 == 'NR' and case2 =='B':
            f=lambda e:(2*V*np.pi*(2*m)**(3/2)/h**3)*(1/(np.exp(alpha1)*np.exp(e/(k*i)) -1))*(e**1.5)
            internal_energy.append(integrate.quad(f,U+0.0001,2)[0])

        elif case1 == 'R' and case2 =='F':
            f=lambda e:(4*V*np.pi/(h*c)**3)*(1/(np.exp(alpha2)*np.exp(e/(k*i)) +1))*(e**3)
            internal_energy.append(integrate.quad(f,0.0001,10)[0])

        elif case1 == 'NR' and case2 =='F':
            f=lambda e:(2*V*np.pi*(2*m)**(3/2)/h**3)*(1/(np.exp(alpha2)*np.exp(e/(k*i)) +1))*(e**1.5)
            internal_energy.append(integrate.quad(f,0.0001,10)[0])
```

```
        else:
            print('ERROR? plz only enter the valid cases i.e (N,NR,B,F)for(relativistic,non-relativistic,bosons and fermions)')

    return internal_energy
```

Code4

(about 10 lines)

```
#plot of dN/de vs e for fermions
e = np.linspace(0,20,100)
T =[1000,20000,50000]
U=0;ef=5
print('The characteristic temperature used here is 1/k = ',1/k)
fig=plt.figure()
fig.set_figheight(7)
fig.set_figwidth(9)

plt.subplot(2,2,1)
plt.plot(e,dN_de(e, T[0], case1='NR', case2='F'),'o-',c='r',markersize=5,label='low T= '+str(T[0]))
plt.legend()
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')
plt.title("FOR NON-RELATIVISTIC FERMIONS")
plt.xlim(0,10)

plt.subplot(2,2,2)
plt.plot(e,dN_de(e, T[0], case1='R', case2='F'),'o-',c='r',markersize=5,label='low T= '+str(T[0]))
plt.legend(loc='best')
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')
plt.title("FOR RELATIVISTIC FERMIONS")
plt.xlim(0,10)
plt.subplot(2,2,3)
plt.plot(e,dN_de(e, T[1], case1='NR', case2='F'),'o-',c='violet',markersize=5,label='high T= '+str(T[1]))
plt.legend(loc='best')
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')

plt.subplot(2,2,4)
plt.plot(e,dN_de(e, T[1], case1='R', case2='F'),'o-',c='violet',markersize=5,label='high T= '+str(T[1]))
plt.legend(loc='best')
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')
```

Code5

(about 10 lines)

```
plt.suptitle('PLOT OF DISTRIBUTION OF PARTICLES VS ENERGY FOR FERMIONS')
plt.show()

plt.subplot(2,2,2)
plt.plot(T[10:]/1000,internal(T[10:], case1='NR', case2='F'),'o-',c='g',markersize=4,label='non-relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
plt.title("FOR HIGH TEMPERATURE")
plt.legend()
plt.suptitle('PLOT OF INTERNAL ENERGY VS TEMPERATURE FOR FERMIONS ')


plt.subplot(2,2,3)
plt.plot(T[:10]/1000,internal(T[:10], case1='R', case2='F'),'o-',c='r',markersize=4,label='relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
plt.legend()

plt.subplot(2,2,4)
plt.plot(T[10:]/1000,internal(T[10:], case1='R', case2='F'),'o-',c='g',markersize=4,label='relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
plt.suptitle('PLOT OF INTERNAL ENERGY VS TEMPERATURE FOR FERMIONS ')
plt.legend()
plt.show()

#plot of dN/de vs e for bosons
U=0
e = np.linspace(U+0.0001,7,100)

T =[1500,10000,50000]

fig=plt.figure()
fig.set_figheight(7)
fig.set_figwidth(9)
plt.subplot(2,2,1)
```

Code6

(about 10 lines)

```
plt.plot(e,dN_de(e, T[0], case1='NR', case2='B'),'o-',c='r',markersize=5,label='low T= '+str(T[0]))
plt.legend()
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')
plt.xlim(0,1)
plt.ylim(0,1e44)
plt.title("FOR NON-RELATIVISTIC BOSONS")

plt.subplot(2,2,2)
plt.plot(e,dN_de(e, T[0], case1='R', case2='B'),'o-',c='r',markersize=5,label='low T= '+str(T[0]))
plt.legend(loc='best')
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')
plt.xlim(0,2)

plt.title("FOR RELATIVISTIC BOSONS")

plt.subplot(2,2,3)
plt.plot(e,dN_de(e, T[1], case1='NR', case2='B'),'o-',c='violet',markersize=5,label='high T= '+str(T[1]))
plt.legend(loc='best')
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')
plt.xlim(0,1)
plt.ylim([0,1e45])

plt.subplot(2,2,4)
plt.plot(e,dN_de(e, T[1], case1='R', case2='B'),'o-',c='violet',markersize=5,label='high T= '+str(T[1]))
plt.legend(loc='best')
plt.grid()
plt.xlabel('e (in eV)')
plt.ylabel('dN / de ')

#plt.title("FOR RELATIVISTIC BOSONS")
plt.suptitle('PLOT OF DISTRIBUTION OF PARTICLES VS ENERGY FOR BOSONS')

plt.show()
T=np.linspace(10,200000,200)
```

Plot1

Submitted files

test1 - Gaurav Chandra.png

Plot2

Submitted files

test2 - Gaurav Chandra.png

Plot3

Submitted files

test3 - Gaurav Chandra.png

Plot4

Submitted files

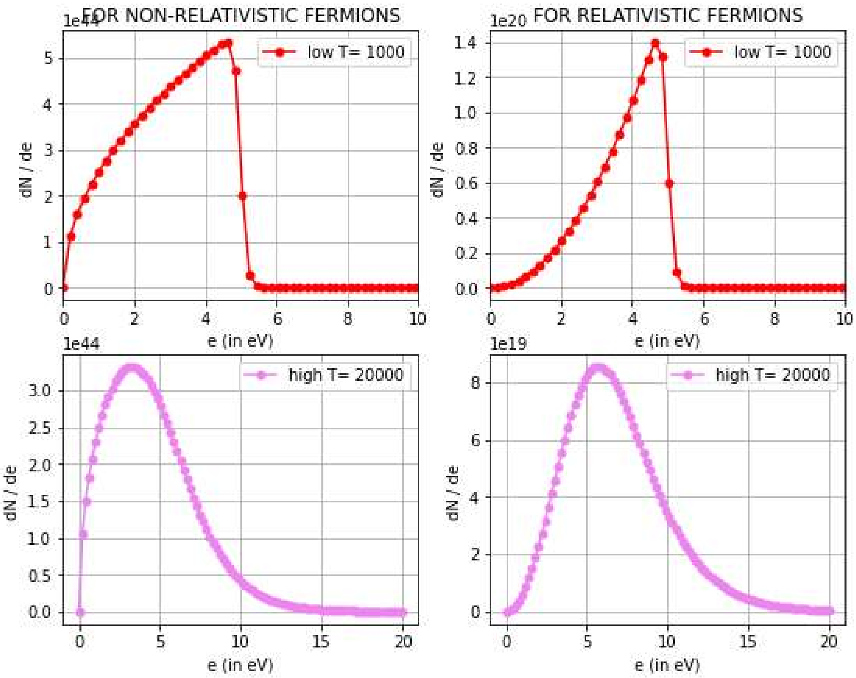
test4 - Gaurav Chandra.png

Comments

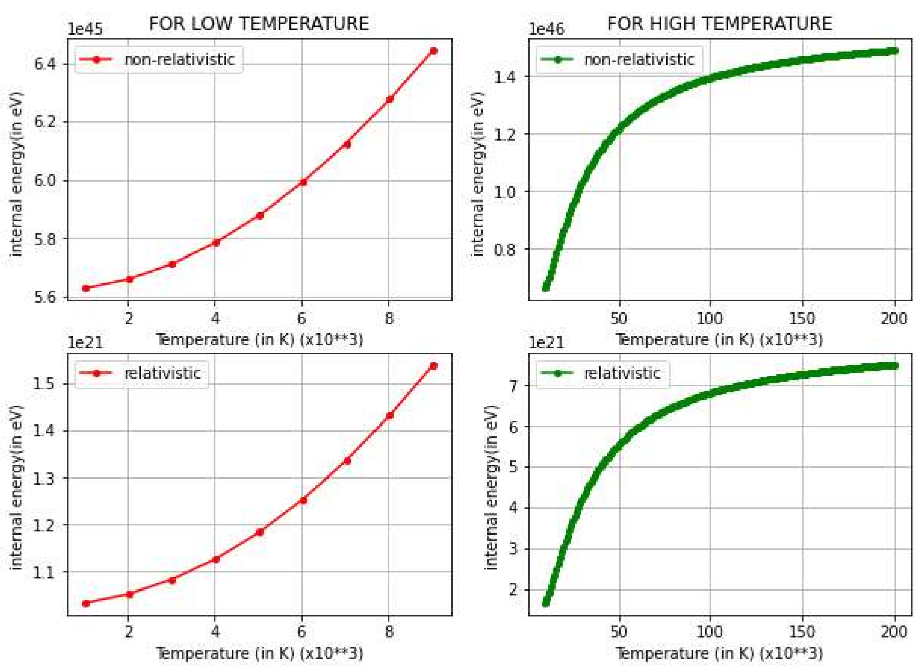
```
fig=plt.figure()
fig.set_figheight(7)
fig.set_figwidth(10)
plt.subplot(2,2,1)
plt.plot(T[:10]/1000,internal(T[:10], case1='NR', case2='B'),'o-',c='r',markersize=4,label='non-relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
plt.title("FOR LOW TEMPERATURE")
plt.legend()
plt.subplot(2,2,2)
plt.plot(T[10:]/1000,internal(T[10:], case1='NR', case2='B'),'o-',c='g',markersize=4,label='non-relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
plt.title("FOR HIGH TEMPERATURE")
plt.legend()
plt.suptitle('PLOT OF INTERNAL ENERGY VS TEMPERATURE FOR BOSONS ')


plt.subplot(2,2,3)
plt.plot(T[:10]/1000,internal(T[:10], case1='R', case2='B'),'o-',c='r',markersize=4,label='relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
```

```
#plt.title("FOR LOW TEMPERATURE")
plt.legend()

plt.subplot(2,2,4)
plt.plot(T[10:]/1000,internal(T[10:], case1='R', case2='B'),'o-',c='g',markersize=4,label='relativistic')
plt.grid()
plt.xlabel('Temperature (in K) (x10**3)')
plt.ylabel('internal energy(in eV) ')
#plt.title("FOR HIGH TEMPERATURE")
plt.suptitle('PLOT OF INTERNAL ENERGY VS TEMPERATURE FOR BOSONS ')
plt.legend()
plt.show()




slope1=stats.linregress(T[1:],internal(T[1:], case1='NR', case2='F'))[0]
slope2=stats.linregress(T[1:],internal(T[1:], case1='R', case2='F'))[0]
slope3=stats.linregress(T,internal(T, case1='NR', case2='B'))[0]
slope4=stats.linregress(T,internal(T, case1='R', case2='B'))[0]
specific_heat=[slope1,slope2,slope3,slope4]

print('The specific heat for fermi gas for non-relativistic fermions is ',slope1)
print('The specific heat for fermi gas for relativistic fermions is ',slope2)
print('The specific heat for boson gas for non-relativistic bosons is ',slope3)
print('The specific heat for boson gas for relativistic bosons is ',slope4)


#references :Thermal Physics by SC Garg ,R M Bansal & C K Ghosh book pageno:595 section 14.2,14.3,14.4
```

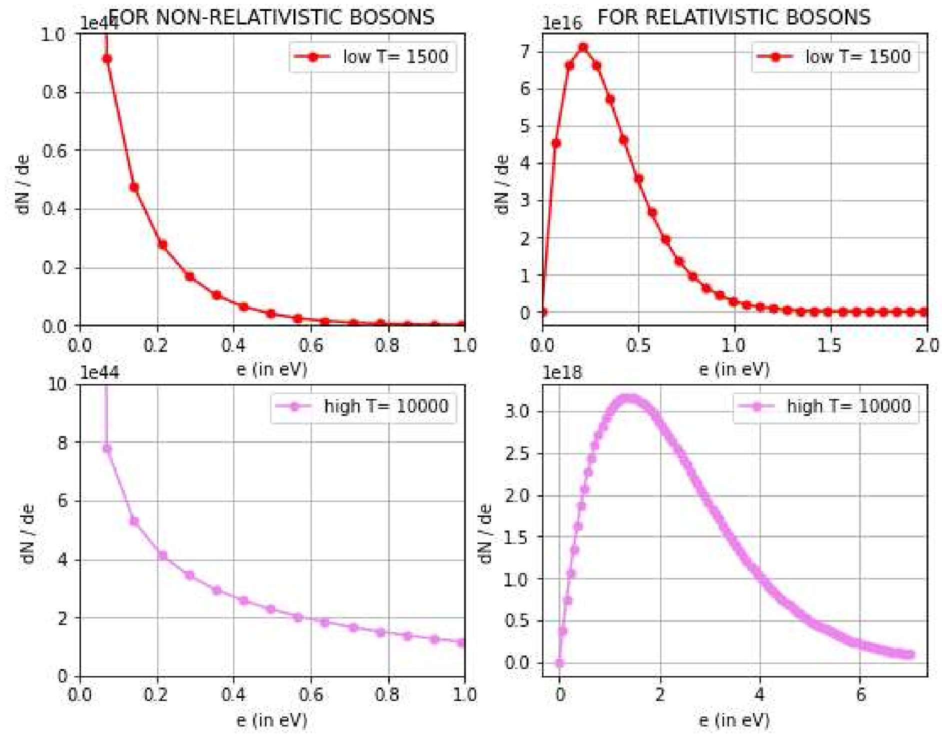
PLOT OF DISTRIBUTION OF PARTICLES VS ENERGY FOR FERMIONS



PLOT OF INTERNAL ENERGY VS TEMPERATURE FOR FERMIONS



PLOT OF DISTRIBUTION OF PARTICLES VS ENERGY FOR BOSONS



PLOT OF INTERNAL ENERGY VS TEMPERATURE FOR BOSONS