Fys2160 Oblig 2

Daniel Heinesen, daniehei

1. oktober 2017

1 Exercise 1)

1.1 a)

The partition function is given as

$$Z = \sum_s e^{-E(s)/kT} = \sum_s e^{-E(s)\beta} \tag{1}$$

We have the energies $\epsilon_1 = \epsilon$ and $\epsilon_2 = \epsilon_3 = \epsilon_4 = 2\epsilon$, giving us the partition function:

$$Z = e^{-\beta\epsilon} + 3e^{-2\beta\epsilon} \tag{2}$$

1.2 b)

From the partition function we are able to find the average energy from the following equation:

$$\langle E \rangle = \sum_s E(s) e^{-E(s)\beta} - \frac{\partial \ln Z}{\partial \beta} \tag{3}$$

So the average energy for our system is:

$$\langle E \rangle = -\frac{\partial}{\partial \beta} \ln\left(e^{-\beta\epsilon} + 3e^{-2\epsilon\beta}\right) \tag{4}$$

$$= -\frac{-\epsilon e^{-\beta\epsilon} - 6\epsilon e^{-2\beta\epsilon}}{e^{-\beta\epsilon} + 3e^{-2\epsilon\beta}} = \epsilon \frac{e^{\beta\epsilon} + 6}{e^{\beta\epsilon} + 3} \tag{5}$$

As a function of temperature:

$$\langle E(T) \rangle = \epsilon \frac{e^{\frac{\epsilon}{kT}} + 6}{e^{\frac{\epsilon}{kT}} + 3} \tag{6}$$

Since we only have one molecule, this is also the total energy U.

1.3 c)

The heat capacity can be found from

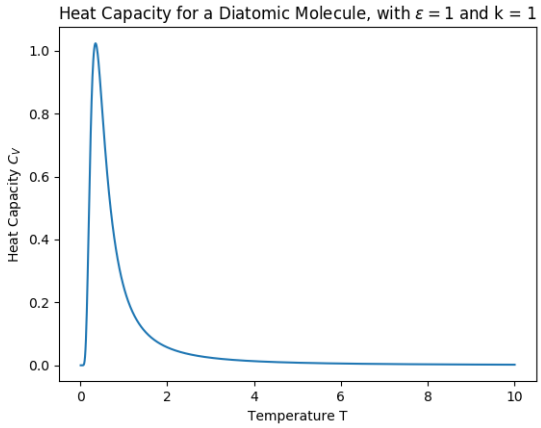
$$C_V = \left(\frac{\partial U}{\partial T}\right)_V = \left(\frac{\partial \langle E(T) \rangle}{\partial T}\right)_V \tag{7}$$

So using the energy found in eq. (1.2), but with the change of variable $1/kT = \beta$

$$C_V = \frac{\partial \beta}{\partial T} \frac{\partial}{\partial \beta} \epsilon \frac{e^{\beta\epsilon} + 6}{e^{\beta\epsilon} + 3} = -\frac{\epsilon}{kT^2} \frac{\epsilon e^{\beta\epsilon}\left(e^{\beta\epsilon} + 3\right) - \epsilon e^{\beta\epsilon}\left(e^{\beta\epsilon} + 6\right)}{\left(e^{\beta\epsilon} + 3\right)^2} \tag{8}$$

$$= \frac{\epsilon^2}{kT^2} \frac{3e^{\epsilon\beta}}{\left(e^{\beta\epsilon} + 3\right)^2} = \frac{\epsilon^2}{kT^2} \frac{3e^{\frac{\epsilon}{kT}}}{\left(e^{\frac{\epsilon}{kT}} + 3\right)^2} \tag{9}$$

For simplicity we are going to set $\epsilon = k = 1$ for the plot of the heat capacity. The code can be found at the end as **c.py**(2.1)



Figur 1: The heat capacity of a simple diatomic molecule.

Above a certain temperature the C_V goes to zeros, which means that, from eq. (1.3), the energy is unchanged even as more and more energy is added.

1.4 d)

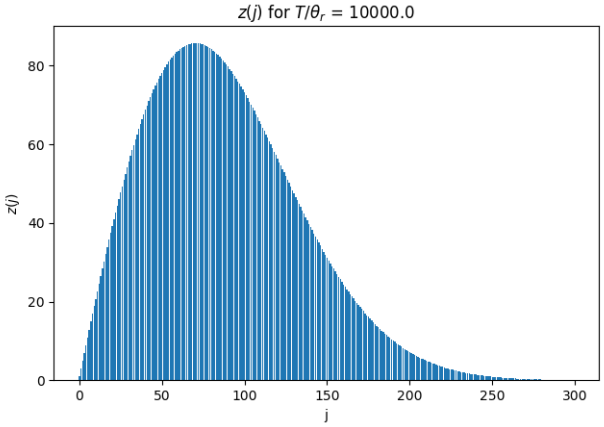
Because of degeneracy eq. (1.1) for the partition function becomes

$$Z_R(T) = \sum_j g(j) e^{-E(j)/kT} = \sum_j (2j+1) e^{-j(j+1)\theta_r/T} \tag{10}$$

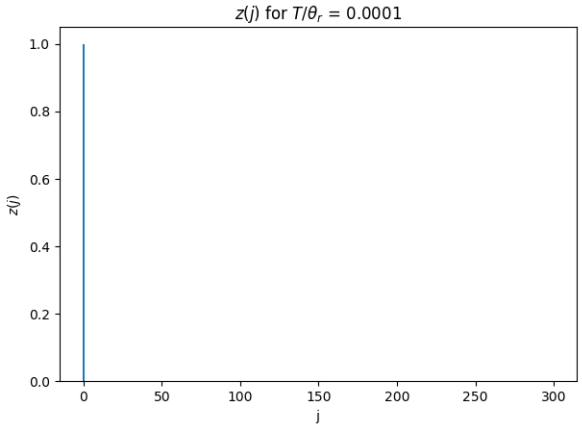
Where $\theta_r = \frac{\hbar^2}{2Ik}$

1.5 e)

The code can be found at the end as **e.py**(2.2):



Figur 2: A bar plot showing the terms of the partition function for $T/\theta_r >> 1$.



Figur 3: A bar plot showing the terms of the partition function for $T/\theta_r << 1$.

We can see that for large values for T/θ_r terms for large values of j contribute, but for small values of T/θ_r the only term that contributes is $z(0) = 1$ and the rest goes to zero.

1.6 f)

We can see in fig. 1.5 that when $T >> \theta_r$ that we can make a good approximation of the terms $z(j)$ as a continuous function, and therefore let the eq. (1.4) go towards an integration:

$$Z_r \approx \int_0^\infty (2j+1)e^{-j(j+1)\theta_r/T}dj \tag{11}$$

To solve this integral we use the change of variable

$$x = j(j+1) = j^2 + j, \qquad \frac{dx}{dj} = 2j+1 \Rightarrow dj = \frac{dx}{2j+1} \tag{12}$$

To the integral becomes

$$Z_r \approx \int_0^\infty e^{-x\theta_r/T}dx = -\frac{T}{\theta_r}e^{-x\theta_r/T}\Big|_0^\infty = \frac{T}{\theta_r} \tag{13}$$

So in the limit $T >> \theta_r$ $Z_r(T) = T/\theta_r$.

1.7 g)

For the limit $T << \theta_r$, we can see from fig. 1.5 that only the first term contribute. So we only need to write out the first few terms of eq. (1.4):

$$Z_r \approx 1 + 3e^{-2\theta_r/T} \tag{14}$$

The rest of the term goes very rapidly to zero, and is unnecessary to use.

1.8 h)

1.8.1 High T

For the high temperature approximation we can use eq. (1.2) to get

$$\langle E\rangle = -\frac{\partial \ln Z}{\partial \beta} = -\frac{1}{Z}\frac{\partial}{\partial \beta}Z = -\beta k\theta_r \frac{\partial}{\partial \beta}\frac{1}{\beta k\theta_r} \tag{15}$$

$$= \beta k\theta_r \frac{1}{\beta^2 k\theta_r} = \frac{1}{\beta} = kT \tag{16}$$

So for high temperature $\langle E\rangle = kT$

1.8.2 Low T

We again use eq. (1.2):

$$\langle E\rangle = -\frac{\partial}{\partial \beta}\ln 1 + 3e^{-2\theta_r k\beta} = \theta_r k \frac{6e^{-2\theta_r k\beta}}{1+3e^{-2\theta_r k\beta}} = \frac{6\theta_r k}{e^{2\theta_r/T}+3} \tag{17}$$

So for low temperatures $\langle E\rangle = \frac{6\theta_r k}{e^{2\theta_r/T}+3}$

1.9 i)

To find the heat capacity we use eq. (1.3):

1.9.1 High T

For high temperatures we use eq. (1.8.1) and find the heat capacity

$$C_v = \frac{\partial}{\partial T}kT = k \tag{18}$$

So for high temperatures the hear capacity is simply $C_v = k$

1.9.2 Low T

For low temperatures we use eq. (1.8.2). First we do a change of variable $\beta = 1/kT$

$$C_v = \frac{\partial \beta}{\partial T}\frac{\partial}{\partial \beta}\frac{6\theta_r k}{e^{2\theta_r k\beta}+3} = \frac{-1}{kT^2}\frac{6\theta_r k e^{2\theta_r k\beta}\cdot 2\theta_r k}{\left(e^{2\theta_r k\beta}+3\right)^2} = \frac{12\theta_r^2 k e^{2\theta_r/T}.}{T^2\left(e^{2\theta_r/T}+3\right)^2} \tag{19}$$

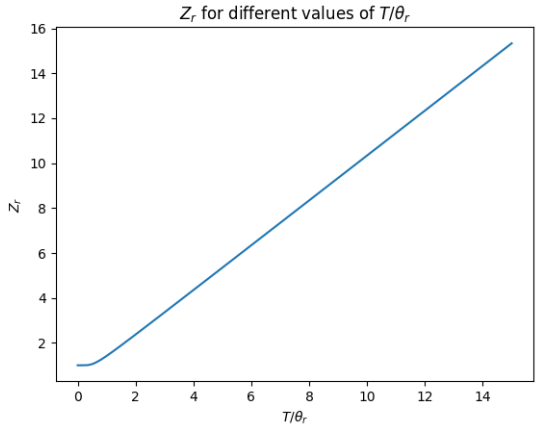
So for low temperatures the heat capacity is $C_v = \frac{12\theta_r^2 k e^{2\theta_r/T}.}{T^2\left(e^{2\theta_r/T}+3\right)^2}$

1.10 j)

The function can be found at the end as **j.py**(2.3).

1.11 k)

This also uses the code found in **j.py**(2.3), and is the first plot:

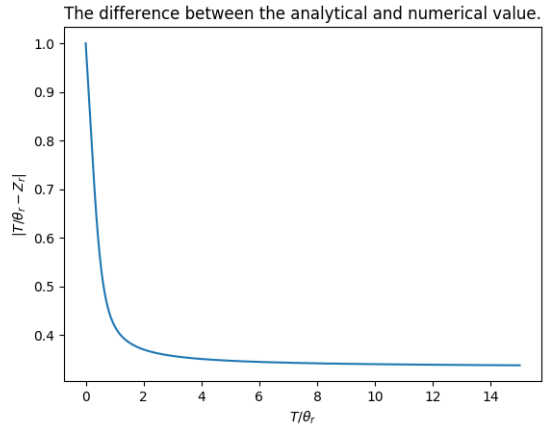


Figur 4: The partition function Z_r for different values of T/θ_r

We can see that near $T/\theta_r = 0$ Z_r is constantly $\backsim 1$, but for larger values of T/θ_r Z_r becomes a linear function. This is exactly what we found in (1.7) and (1.6).

1.12 l)

The code for this is the second plot in **j.py**(2.3), but is plotted with T/θ_r up to 200.



Figur 5: Shows the difference between the analytical value for high temperature and the numerical value of the partition function.

We can see from the above plot that the difference between the analytical value for high temperature and the numerical value for the partition function quickly decreases for higher values of T/θ_r, but seems to flatten out around 1/3. This is because we try to approximate a discrete function with a continuous one, which means that an integral will either over- or undershoot the real value. If we had drawn the continuous function on top of the bar plot in fig. 1.5 we would see that many bars had area outside over that function, meaning that in this case analytical value is about 1/3 lower than the numerical value(This is printed out when running the code).

1.13 m)

To find the energy numerically we use that

$$\langle E \rangle = \frac{\sum_j E(j) e^{-\beta E(j)}}{Z} = \frac{\sum_j j(j+1)\theta_r k(2j+1) e^{-j(j+1)\theta_r/T_i}}{\sum_j (2j+1) e^{-j(j+1)\theta_r/T_i}} = E(T_i) \tag{20}$$

Since the heat capacity (1.3) is defined with a differentiation, we have to use a numerical differentiation to find it.

To get most precision we use the symmetric difference quotient:

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} \tag{21}$$

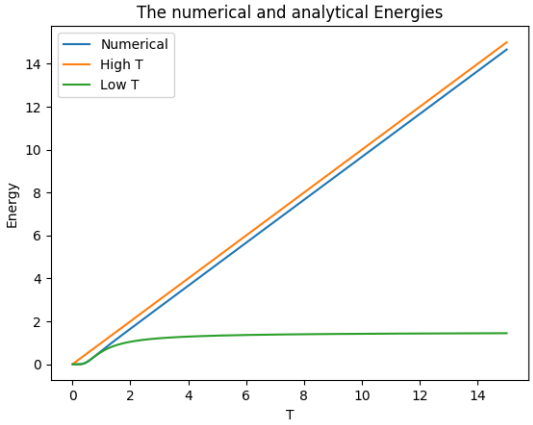
And thus

$$C_v = \frac{1}{2h}(E(T_{i+1}) - E(T_{i-1}) \tag{22}$$

1.14 n)

The code can be found at the end as **n.py**(2.4)

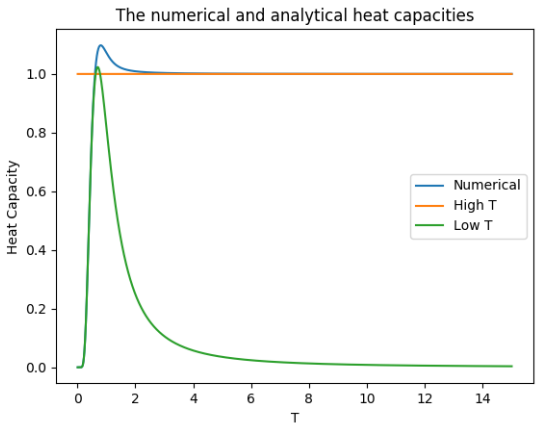
1.14.1 Energy



Figur 6: The numerical and analytic energies for different T

As we can see, for low temperature the analytical and numerical solutions are in agreement, but for our high temperature solution the numerical solution is a constant higher than the analytic. This is stark contrast with the partition function where it was lower.

1.14.2 Energy



Figur 7: The numerical and analytic heat capacities for different T

We can that the analytical and numerical solutions are good fits in their own domains. The constant difference for the high temperature is now gone, this is because for the energies the solutions were linear with a constant difference. Heat capacity is the derivative of the energies, meaning that this constant difference is differentiated away.

2 Codes

2.1 c.py

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

T = np.linspace(0.,10,1000)
eps = 1
k = 1

C = lambda T: (eps**2/(k*T**2))*(3*np.exp(eps/(k*T)))/(np.exp(eps/(k*T)) +
    3)**2


plt.plot(T,C(T))
plt.title(r"Heat Capacity for a Diatomic Molecule, with $\epsilon = ${} and
    k = {}".format(eps,k))
plt.xlabel("Temperature T")
plt.ylabel(r"Heat Capacity $C_V$")
plt.show()
```

2.2 e.py

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

j = np.arange(0,300)
```

```
z = lambda j, theta_over_T: (2*j + 1)*np.exp(-j*(j+1)*theta_over_T)


theta_over_T = [0.0001,10000]

for toT in theta_over_T:
    plt.bar(j,z(j,toT))
    plt.title(r"$z(j)$ for $T/\theta_r$ = {}".format(1./toT))
    plt.ylabel(r"$z(j)$")
    plt.xlabel("j")
    plt.show()
```

2.3 j.py

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



def Z(T_over_theta, j_limit = 1e7, eps = 1e-10):
    """
    j_limit how many j's we  want to use, and is default 1e7.
    The function stops to sum if the term is smaller than some epsilon eps.
    This is 1e-10 by default. The j > 2 is the to ensure that
    the break doesn't happen at the very start.
    """
    result = 0
    for j in range(int(j_limit)):
        term = (2*j + 1)*np.exp(-j*(j+1)*1./T_over_theta)
        result += term

        if abs(term) < eps and j>2:
            break
    return result


if __name__=="__main__":
    N = 10000
    T_over_theta = np.linspace(0.00001,15,N)
    partition = np.zeros(N)
    for i,toT in enumerate(T_over_theta):
        partition[i] = Z(toT)

    #For k)
    plt.plot(T_over_theta,partition)
    plt.title(r"$Z_r$ for different values of $T/\theta_r$")
    plt.xlabel(r"$T/\theta_r$")
    plt.ylabel(r"$Z_r$")
    plt.show()

    #For l)
    plt.plot(T_over_theta,np.abs(T_over_theta-partition))
    plt.title("The difference between the analytical and numerical value.")
    plt.xlabel(r"$T/\theta_r$")
    plt.ylabel(r"$|T/\theta_r-Z_r|$")
    plt.show()
```

```
    print("The difference is about ",T_over_theta[-1]-partition[-1])
```

2.4 n.py

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


def numerical_energy(T, theta=1, k=1, j_limit=1e7, eps=1e-10):
    result = 0
    result_z = 0
    for j in range(int(j_limit)):
        term = j*(j+1)*k*theta*(2*j + 1)*np.exp(-j*(j+1)*theta/float(T))
        term_z = (2*j + 1)*np.exp(-j*(j+1)*theta/float(T))
        result += term
        result_z += term_z

        if abs(term) < eps and j > 2:
            break
    return result/result_z

def numerical_CV(E,h):
    return 1./(2*h)*(E[2:]-E[:-2])


def approx_energy_high(T,k=1):
    return k*T

def approx_energy_low(T,k=1,theta=1):
    return (6*theta*k)/(np.exp(2*theta/T) + 3.)

def approx_cv_high(T,k=1):
    return k

def approx_cv_low(T,k=1,theta=1):
    return (12*theta**2*k*np.exp(2*theta/T))/(T**2*(np.exp(2*theta/T)+3.)
        **2)


if __name__=="__main__":
    k = 1
    theta = 1
    T_start = 0.01
    T_end = 15
    T_N = 10000
    dT = (T_end-T_start)/T_N
    T = np.linspace(T_start,T_end,T_N,endpoint=True)


    #Gets the numerical energies
    num_energies = np.zeros_like(T)
    for i,t in enumerate(T):
        num_energies[i] = numerical_energy(t,j_limit=1e3)

    #Gets the numerical heat capacities
    num_cv = numerical_CV(num_energies,dT)
```

```
    #Get the analytical approximations
    approx_energies_h = approx_energy_high(T)
    approx_energies_l = approx_energy_low(T)
    approx_cvs_l = approx_cv_low(T)
    approx_cvs_h = np.vectorize(approx_cv_high)(T)
    """
    #The np.vectorize is to ensure that the constant k is returned as an
        array
    """


    #Plots for energies:
    plt.plot(T,num_energies,label="Numerical")
    plt.plot(T,approx_energies_h,label="High T")
    plt.plot(T,approx_energies_l,label="Low T")
    plt.legend()
    plt.title("The numerical and analytical Energies")
    plt.xlabel("T")
    plt.ylabel("Energy")
    plt.show()

    #Plots for heat capacities:
    plt.plot(T[1:-1],num_cv,label="Numerical")
    plt.plot(T[1:-1],approx_cvs_h[1:-1],label="High T")
    plt.plot(T[1:-1],approx_cvs_l[1:-1],label="Low T")
    plt.legend()
    plt.title("The numerical and analytical heat capacities")
    plt.xlabel("T")
    plt.ylabel("Heat Capacity")
    plt.show()
```