Report file - Problem Set #4

Matteo Dell'Acqua
GitHub: MatteoDellAcqua6121

October 1, 2024

Abstract

This is the report for the problem set #4. Since the problem set is composed of three exercises, we divide the report into three sections, one for each problem. The scripts (labelled as ps_4.'problem number') and the raw file of the images are in this directory.

1 Problem 1

1.1 Formulation of the problem

We are asked to compute the heat capacity of a solid at temperature T according to Debye's theory:

$$C_V = 9V\rho k_B \left(\frac{T}{\theta_D}\right)^3 \int_0^{\theta_D/T} \frac{x^4 e^x}{(e^x-1)^2}\mathrm{d}x, \tag{1}$$

where k_B is Boltzmann's constant, and for a sample consisting of $V = 1000cc$ of aluminium, whose physical parameters are given by[1]:

$$\rho = 6.022 \times 10^{28}\ m^{-3}, \qquad \theta_D = 428\ K. \tag{2}$$

The problem consists of evaluating the integral using Gaussian quadrature and testing the convergence of this method.

1.2 Computational methods

Gaussian quadrature is a method of computing integrals using the approximation:

$$\int_{-1}^{1} f(x)\mathrm{d}x = \sum_{i=0}^{N_1} w_i f(x_i), \tag{3}$$

where x_i are the roots of the Nth Legendre polynomial $P_N(x)$ (you can find them using `np.polynomial.legendre.leggauss`) and the weights w_i are given by:

$$w_i = \frac{2}{(1-x_i^2)(P_N'(x_i))^2}. \tag{4}$$

[1] θ_D is the so-called Debye temperature.

This method has the advantage of providing an exact answer for $f(x)$ a polynomial of degree less than $2N-1$ and converging to the exact answer as $N \to \infty$ for a generic function $f(x)$. In the first part of the problem, we will choose $N = 100$. In order to apply this method to the integral of interest, we need to adjust the integration interval. In general, this can be done with a change of variable, as follows:

$$x' = \frac{b-a}{2}x + \frac{b+a}{2}, \quad \mathrm{d}x' = \frac{b-a}{2}\mathrm{d}x, \quad \frac{b-a}{2}\int_{-1}^{1} f(x)\mathrm{d}x = \int_{a}^{b} f(x')\mathrm{d}x'. \tag{5}$$

The implementation goes as follows[2]:

```
def c_T(T,N):
    #generate root and weights
    xp,wp=np.polynomial.legendre.leggauss(N)
    #rescale them according to the integration interval
    a=0
    b=theta/T
    xp=0.5*(b-a)*xp+0.5*(b+a)
    wp=0.5*(b-a)*wp
    int_temp=np.zeros(N, dtype=np.float32)
    #temporarily store the contributions and use np.sum
    for i in np.arange(N):
        int_temp[i]=wp[i]*f(xp[i])
    return 9*V*k_B*rho*(T/theta)**3/10*int_temp.sum()
```

where the 1/10 factor takes care of the powers of ten (and unit change) present in the various constant factors in order to get an answer expressed in $[J/K]$, and f is the integrand in question:

```
def f(x):
    return x**4*np.exp(x)/(np.exp(x)-1)**2
```

Finally, we test the convergence of the result obtained in this way, as $N \to \infty$ for a fixed value of $T = 5K$[3] by computing the same heat coefficient $C_T(T = 5K, N)$ for $N \in \{10, 20, 30, 40, 50, 60, 70\}$.

1.3 Results

We report the results of our simulations (plotted using `matplotlip.pyplot`) in figs. 1 and 2.

2 Problem 2

2.1 Formulation of the problem

In this problem we are going to study the behaviour of anharmonic oscillators: 1d systems of a single particle in a concave potential well $V(x)$. We are going to work with the simplifying assumption that the potential is symmetric:

$$V(x) = V(-x). \tag{6}$$

[2]We used the `sum` attribute of `np.array` since we have previously showed it is faster than a `for` loop and less prone to roundoff errors.

[3]One could plot C_T vs T for the various N instead, but the functions plotted this way are not distinguishable by

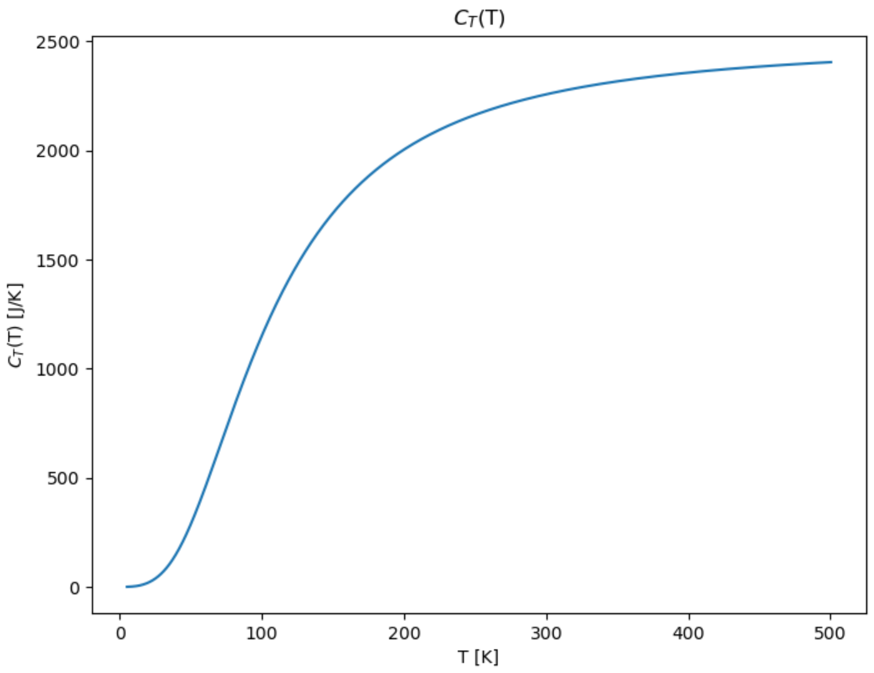

Figure 1: Plot of the specific heat of aluminium as a function of T in the interval $[5K, 500K]$, computed using Gaussian quadrature for $N = 50$ points.

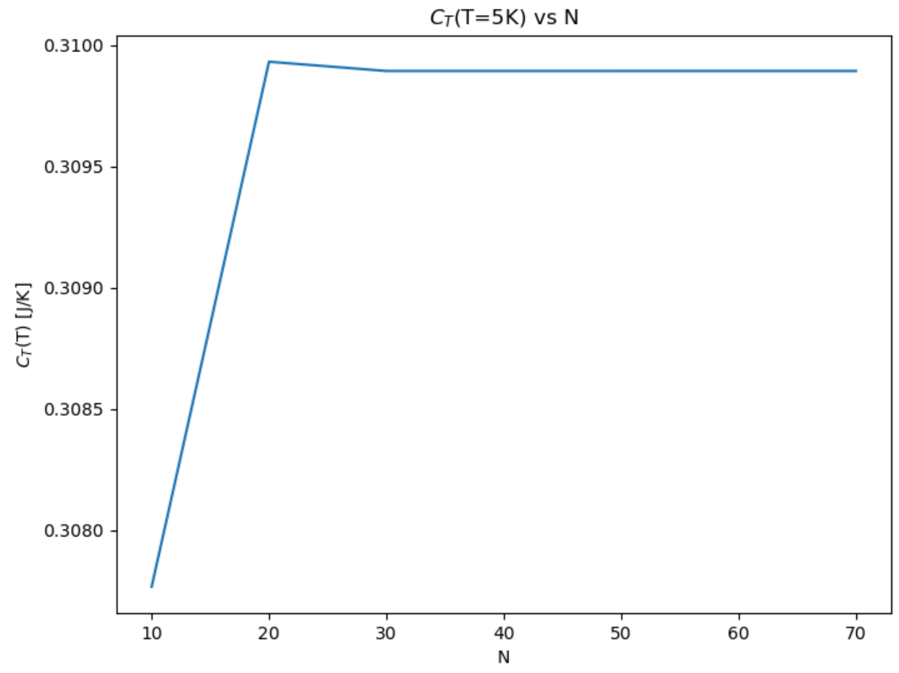

Figure 2: Plot of the specific heat of aluminium at $T = 5K$ as a function of the number N of points used in the Gaussian quadrature.

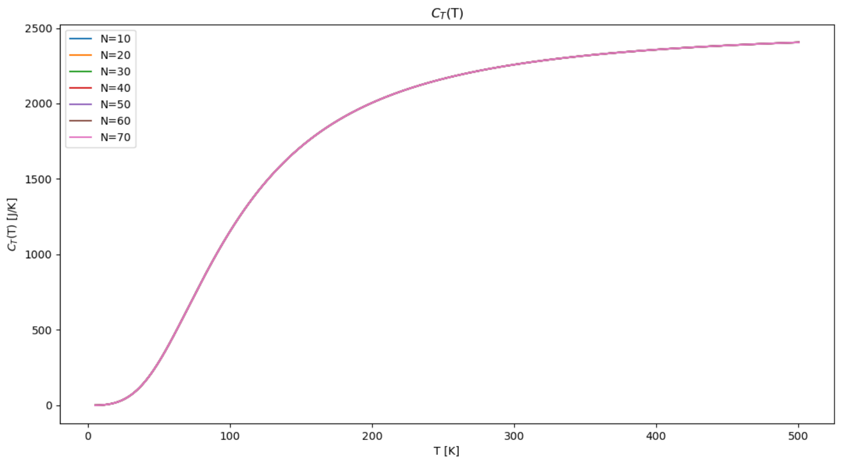

Figure 3: Plot of the specific heat of aluminium as a function of T in the interval $[5K, 500K]$, computed using Gaussian quadrature for $N \in \{10, 20, 30, 40, 50, 60, 70\}$ points.

Theorem 1 *The period T of a symmetric anharmonic potential of amplitude a is given by:*

$$T = \sqrt{8m} \int_0^a \frac{\mathrm{d}x}{\sqrt{V(a) - V(x)}}. \tag{7}$$

Consider a particle in the initial state:

$$v(t=0) = 0, \quad x(t=0) = -a. \tag{8}$$

Due to the symmetries of the problem, the period T is 4 times the time $\bar{t}$ it takes for said particle to reach the position $x = 0$ for the first time:

$$T = 4 \int_0^{\bar{t}} \mathrm{d}t = 4 \int_{-a}^{0} \frac{\mathrm{d}x}{v(x)}, \tag{9}$$

where we performed the change of variables $t \to x(t)$, $\mathrm{d}t \to \mathrm{d}x/\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)$ (and used the facts $x(t=0) = -a$, $x(t=\bar{t}) = 0$).

Imposing the conservation of the energy, we have:

$$V(a) = V(-a) = E(t=0) = E(t = t(x)) = \frac{m}{2} v^2(x) + V(x), \tag{10}$$

or, equivalently:

$$v(x) = \sqrt{\frac{2}{m}} \sqrt{V(a) - V(x)}. \tag{11}$$

The desired theorem is proved by substituting the expression 11 for $v(x)$ into eq. (9). □

In the following we are going to focus on the case $V(x) = x^4$ and $m = 1$ (and work in a unit system where everything is unitless).

2.2 Computational methods

The implementation is, mutatis mutandis (i.e. $N = 20$, integration interval $[0, a]$, function as below), identical to the previous problem, with the following integrand function:

```
def f(x,a):
return 1/np.sqrt(a**4-x**4).
```

2.3 Results

We report the results of our simulations (plotted using `matplotlip.pyplot`) in fig. 4. We find out that the period increases as the amplitude decreases (in particular it diverges as $a \to 0$). One way to understand this fact is the following:

$$\frac{\mathrm{d}v}{\mathrm{d}t} \propto \frac{\mathrm{d}V}{\mathrm{d}x} \quad \Rightarrow \quad \left|\frac{\mathrm{d}v}{\mathrm{d}t}\right| \propto |x^5|. \tag{12}$$

Thus a particle starting still in $x(0) = a >> 1$ will quickly acquire speed much faster than one placed at $x(0) = a << 1$. A counterargument could be that, while the first particle will be

naked eye (see fig. 3.

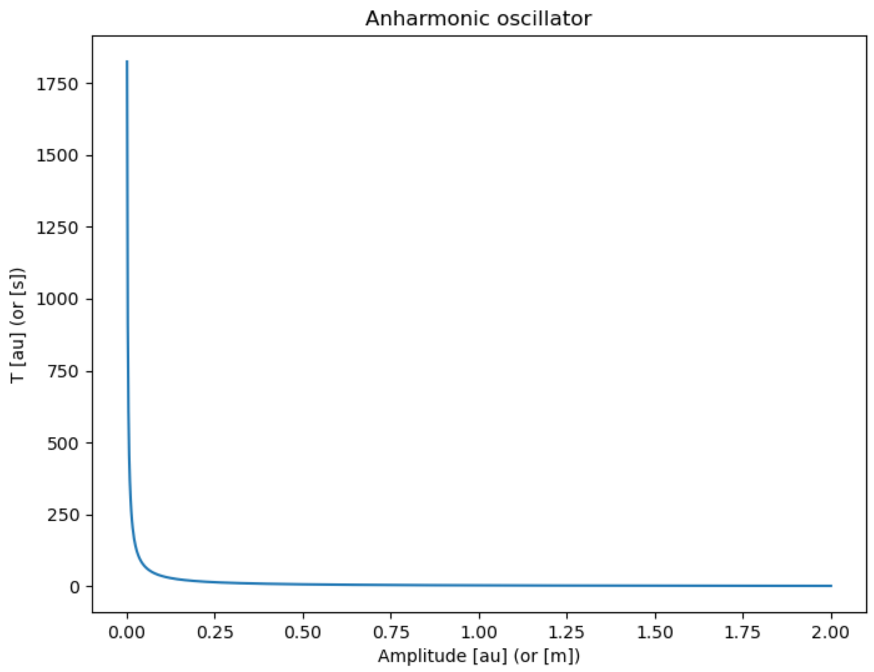

Figure 4: Plot of the period T of an anharmonic oscillator (with potential $V(x) = x^4$) on the amplitude a of the oscillating motion (i.e. the initial position of the particle).

faster, it will have to traverse a greater distance. However, one can (**very roughly**) estimate the time of a quarter period as:

$$v(x) \propto \sqrt{a^4 - x^4} \quad \Rightarrow \quad v \sim a^2 \quad \Rightarrow \quad T \sim \frac{a}{v} \sim \frac{1}{a} \tag{13}$$

where we used the identity 11.

In particular, $x = 0$ is a point of stable equilibrium: a particle initially still and in that position will never move!

3 Problem 3

3.1 Formulation of the problem

In this problem, we are going to study the quantum uncertainty of the harmonic oscillator.

The wavefunction of the nth energy level of a harmonic oscillator is given by [4]:

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} e^{-x^2/2} H_n(x), \tag{14}$$

[4]Here and in the following, we work in units where all the constants are 1.

for $n = 0, \dots \infty$, where $H_n(x)$ is the nth Hermite polynomial.

These polynomials are inductively defined as follows:

$$H_0(x) = 1, \quad H_1(x) = 2x, \quad H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x). \tag{15}$$

The quantum uncertainty of the nth energy level is thus given by:

$$\langle x^2 \rangle = \int_{-\infty}^{+\infty} x^2 |\psi_n(x)|^2 \mathrm{d}x. \tag{16}$$

The goal of the problem is to compute the uncertainty using both Gaussian quadrature and Gauss-Hermite quadrature.

3.2 Computational methods

First of all, we can define the Hermite Polynomials as a function $H(n, x)$ using recursion and dynamic programming (needed to make the computation efficient for large n):

```
def hermite_dp(n, x):
    if n == 0:
        return 1
    elif n == 1:
        return 2 * x

# Create an array to store Hermite values
dp = [0] * (n + 1)
dp[0] = 1
dp[1] = 2 * x

# Fill the table using the recurrence relation
for i in range(2, n + 1):
    dp[i] = 2 * x * dp[i - 1] - 2 * (i - 1) * dp[i - 2]

return dp[n]
```

Then, we compute the root-mean-squared position $\sqrt{\langle x^2 \rangle}$ (c.f.r. eq. (16)) with the same Gaussian square method of the previous two problems. The slight difference is given by the fact that the domain of integration is now infinite. This issue can be solved by an appropriate change of variables:

$$x = \frac{z}{1-z^2}, \quad \mathrm{d}x = \frac{1+z^2}{(1-z^2)^2}\mathrm{d}z, \quad \int_{-\infty}^{+\infty} f(x)\mathrm{d}x = \int_{-1}^{1} \frac{1+z^2}{(1-z^2)^2} f\left(\frac{z}{1-z^2}\right)\mathrm{d}z. \tag{17}$$

Finally, we repeat the computation but using the Gauss-Hermite quadrature. It allows for a more efficient and accurate approximation of integrals of the form:

$$\int_{\infty}^{+\infty} e^{-x^2} f(x) \simeq \sum_{i=0}^{N-1} w_i f(x_i) \tag{18}$$

where now x_i are the roots of the Nth Hermite polynomial $H_N(x)$ (you can find them using `np.polynomial.hermite.hermgauss`) and the weights w_i are accordingly modified. We observe that we are precisely in this setting as $|\psi_n(x)|^2 \propto e^{-x^2}$.

The actual implementation is very similar to that of the Gauss quadrature:

```
def herm_uncertainty(n):
    xp,wp=np.polynomial.hermite.hermgauss(NN)
    int_temp=np.zeros(NN, dtype=np.float32)
    for i in np.arange(NN):
        int_temp[i]=wp[i]*(xp[i])**2*hermite_dp(n,xp[i])**2/(np.sqrt(np.pi)*np.exp2(n)*math.fa
    return int_temp.sum()
```

3.3 Results

We report the plot of the first four and the 30th wavefuntions (realized using `matplotlip.pyplot`) respectively in figs. 5 and 6.

The root-mean-squared position for the 5th wavefunction ψ_5 computed via Gaussian quadrature and Gauss-Hermite quadrature (both on $N = 100$ points) is given by:

$$\sqrt{\langle x^2 \rangle}_{\text{Gauss}} = 2.345208, \qquad \sqrt{\langle x^2 \rangle}_{\text{Hermite}} = 2.345208. \tag{19}$$

Let's compare it with the analytical solution (found with the help of *Wolfram Matematica*):

$$H_5(x) = 32x^5 - 160x^3 + 120x, \quad \sqrt{\langle x^2 \rangle} = \sqrt{\frac{11}{2}} = 2.345208. \tag{20}$$

We can see that in both cases the result is essentially exact (up to round-off errors and the intrinsic problem of representing an irrational number using a finite amounts of bits). This agrees with the fact that Gauss-Hermite quadrature with $N = 100$ points is exact for the integral of polynomials of order up to $2N - 1 = 199$. In our case we are integrating a polynomial of degree 12 ($\propto x^2 H_5(x)^2$): in fact we can verify that the precision of the results for this method is greatly improved going from $N = 6$ to $N = 7$ (and that this last result is not that different from the one we get for $N = 100$), while the ordinary Gauss method behaves differently:

$$(N=5) \quad \sqrt{\langle x^2 \rangle}_{\text{G}} = 0.6117,\ \text{Err}_{\text{G}} = 1.73355 \quad \sqrt{\langle x^2 \rangle}_{\text{H}} = 1.5811,\ \text{Err}_{\text{H}} = 0.76407 \tag{21}$$

$$(N=6) \quad \sqrt{\langle x^2 \rangle}_{\text{G}} = 1.3969,\ \text{Err}_{\text{G}} = 0.94835, \quad \sqrt{\langle x^2 \rangle}_{\text{H}} = 2.345\ \text{Err}_{\text{H}} = -4.9699 \cdot 10^{-8}. \tag{22}$$

The result of the Gauss quadrature, in fact, instead is never going to be exact because of the e^{-x^2} factor in front of the polynomial but it does converge to the actual answer as demonstrated by the above result 19.

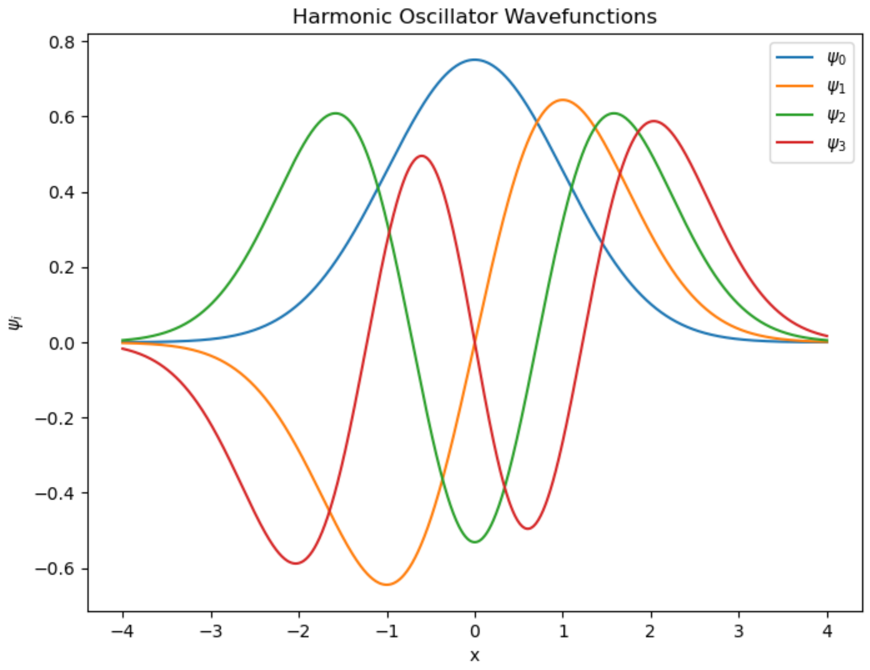

Figure 5: Plot of the first four wavefunctions ψ_i of the harmonic oscillator in the range $x \in [-4, 4]$.

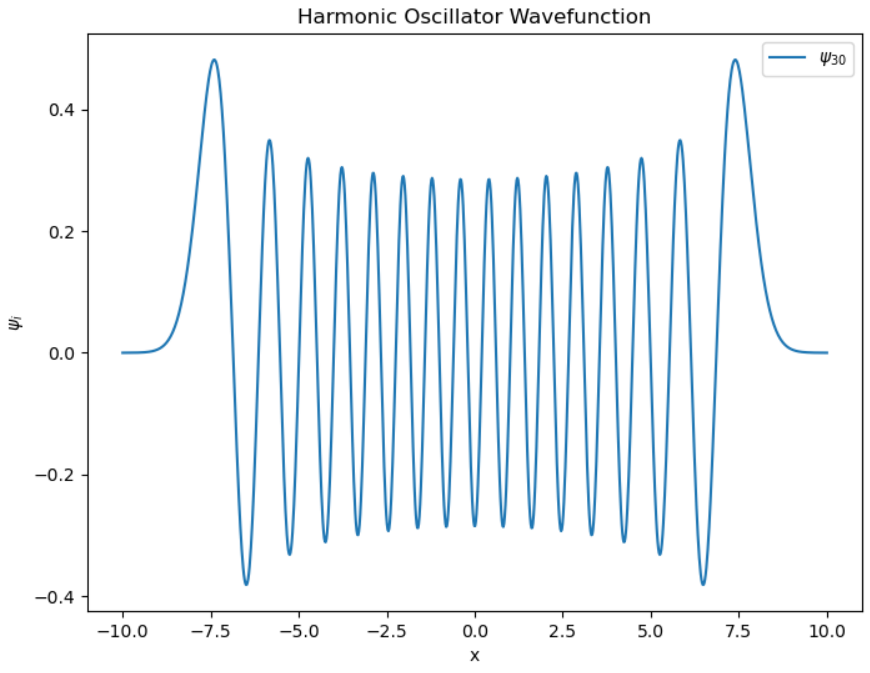

Figure 6: Plot of the 30th wavefunction ψ_{30} of the harmonic oscillator in the range $x \in [-10, 10]$.