Problem Set 4

Debendro Mookerjee
GitHub Account Name: Debendro-dm5790

October 2, 2023

Abstract

The three exercises in this problem set concern numerical integration via Gaussian and related quadratures. The first two problems exclusively deal with Gauss-Legendre quadrature and the last one also involve Gauss-Hermite quadrature. From these exercises we learn various ways to appropriately integrate in Python, such as rescaling the limits of integration and recasting the integrand with dimensionless variables.

1 Introduction

An integral can be numerically computed by approximating it as a finite sum with each term being the product of the integrand evaluated at a sample point and the weight at that point. Some approximation methods involve weights with the same value, such as the rectangular rule, while other schemes involve weights with different values, such as the trapezoidal and Simpson's rules. Gauss-Legendre and Gauss-Hermite quadratures fall into the latter category. In these integration methods, the sample points x_i are the roots of the Legendre and Hermite polynomials of a particular order. The weights depend on these polynomials as well. The sample points and weights have to be rescaled to fit the limits of integration.

Before using these powerful integration techniques, it is important to rewrite the integral we want to compute. This can involve expressing the integrand with unitless variables, rescaling the limits of integration into dimensionless ones, and rewriting integrals ranging over an infinite interval to one ranging over a finite interval. A combination of these techniques have been used in the exercises.

We will first describe the methods we used to answer the three exercises and then we will provide the suitable outputs.

2 Methods

In these exercises we use two functions implemented by Mark Newman: the function gaussxw() which calculates and returns the sample points and the weights for a particular order N, which is the number of desired sample points, and the function gaussxwab() which calculates and returns the sample points and weights scaled to the desired integration limits. These two functions are used for Gauss-Legendre quadrature.

2.1 Exercise 1

For this exercise we wish to determine the heat capacity of a solid according to the Debye's theory of solids:

$$C_V = 9V\rho k_B \left(\frac{T}{\Theta_D}\right)^3 \int_0^{\Theta_D/T} \frac{x^4 e^x}{(e^x-1)^2} dx$$

where V is the volume of the sample, ρ is the number density, k_B is the Boltzmann constant, Θ_D is the Debye temperature, and T is the temperature.

We implement a function called integrand that returns the dimensionless integrand presented in the exercise. It does not return the extra dimensionful factors that are multiplied to the integral. We then implement a function called Cv that calculates the heat capacity of our simulated system. If first uses the function gaussxw to get the sampled values and weights for Gauss-Legendre quadrature and then rescales them to the bounds presented in the exercise description by calling the gaussxwab function. Using the rescaled sampled points and weights, we compute the integral and then we multiply it by the multiplicative factor made from the sample volume, the number density, the Boltzmann constant, the temperature, and the Debye temperature which are the function's input parameters vol, rho, kB, temp, and debT, respectively. The function also has an input parameter called N, which is the number of sample points we want.

We define the constants V, rho debTemp, N, and kB as the sample volume, the number density, the Debeye temperature, the number of sample points, and the Boltzmann constant, respectively. We use 50 sample points. We also define an array of temperatures ranging from 5 to 500 Kelvin. We store it in a variable called TArray. We also define an array to store the corresponding heat capacities. We initialize it as an array of zeros with the same size as the array TArray. We store this array in a variable called heatCaps.

With a for loop, we fill the array heatCaps with the corresponding heat capacity and print a plot of the heat capacity versus temperature for our simulated sample of Aluminum. We save the plot as a .png file called HeatCap.png.

We repeat this procedure for sample sizes of 10,20,30,40,50,60, and 70, which we store in an array called NArray. For heat sample size, we determine the corresponding heat capacity curve and store it into a list called heatCapList. This list is two dimensional. We plot the individual heat capacitity plots onto a single point to see whether these curves converge. They indeed do and the output is saved as a .png file HeatCapConvergenceOverall.png. This plot tests overall convergence.

We use the two-dimensional list to determine whehter the computed heat capacity at a certain temperature converge as the number of sample points vary. We make plots and save them as .png files. We indeed see that the values are precise.

2.2 Exercise 2

In this exercise we wish to determine the period of an an-harmonic oscillator, a system whose potential energy is not $\propto x^2$, where x is position. We will assume that the potential energy

$V(x)$ is even with respect to the origin. The energy of the system

$$E = \frac{1}{2}m\left(\frac{dx}{dt}\right)^2 + V(x)$$

is conserved and if the system starts from rest at $x = a$, then $E = V(a)$. From this we see the following:

$$\frac{dx}{dt} = \pm\sqrt{\frac{2}{m}(V(a) - V(x))} \rightarrow dt = \pm\frac{dx}{\sqrt{\frac{2}{m}(V(a) - V(x))}} = \sqrt{\frac{m}{2}}\frac{dx}{\sqrt{V(a) - V(x)}}$$

Here we note that the integrand is non-negative and time is non-negative. Therefore $\pm$ must be $+$. Since we assumed an even potential, the time it takes for the system to go from $x = a$ to $x = 0$ is one-fourth of the period of oscillator. Thus we see

$$T = 4\sqrt{\frac{m}{2}}\int_0^a \frac{dx}{\sqrt{V(a) - V(x)}} = \sqrt{8m}\int_0^a \frac{dx}{\sqrt{V(a) - V(x)}}$$

In our case $V(x) = x^4$ and so

$$T = \sqrt{8m}\int_0^a \frac{dx}{\sqrt{a^4 - x^4}} = \frac{\sqrt{8m}}{a^2}\int_0^a \frac{dx}{\sqrt{1 - (x/a)^4}} = \frac{\sqrt{8m}}{a}\int_0^1 \frac{dy}{\sqrt{1 - y^4}}$$

We have now re-scaled the limits and re-casted the integrand is a way that they do not depend on any physical constants. This form is suitable for numerical integration. We define the rescaled version of the integrand, which scales out the a^4 term in the original integrand's denominator, in a function called integrand(). We also define a function called Period() which takes as input the amplitude, mass, and the number of sample points. These correspond to the input variables a, mass, and N. The function computes the scaled integral using Gauss-Legendre quadrature and multiplies it by the appropriate constants. The period is returned. We consider a sample size of 20 and a mass of 1. We make a plot of amplitude versus period and save it as a .png file called Anharmonic.png.

Since the energy is conserved, we can think of the system as a particle rolling along the potential landscape. We note that $V(x) = x^4$ gets steeper as x increases at a much faster rate than $V(x) = x^2$ and so it should roll with greater speed as its amplitude increases. The larger speed compensates the extra distance, making the period smaller and smaller. However, when the particle starts with smaller amplitudes, the potential landscape gets flatter and flatter, and so its initial speed gets smaller and smaller. Although distance does decrease, so does its speed and it takes the particle longer and longer to travel. Very close to the origin, the particle rolling in the potential landscape has a speed very close to zero. It takes it extremely long to go back and forth.

2.3 Exercise 3

The wavefunctions of the quantum harmonic oscillator is defined as follows in terms of the Hermite polynomials H_n:

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

where the Hermite polynomials are defined recursively as

$$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$$

We define a function called H(n,x) which dermines the value of the n-th Hermite polynomial at x. We test to see whether out coded Hermite polynomials match the actual Hermite polynomials for the case $n = 5$. They indeed do and we store this verification in a .png file called HermiteTest.png. Using the coded Hermite functions we plot the first four wavefunctions of the harmonic oscillator in the range

$$-4 < x < 4$$

and the 31st wavefunction in the range $-10 < x < 10$. These correspond to $n = 0, 1, 2, 3$, and 30. We save the plots as .png files.

After testing our coded Hermite polynomials and plotting some of the wavefunctions, we are now ready to determine the uncertainty by calculating the following integral:

$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 |\psi_n(x)|^2 dx$$

The integrand is independent of any physical constants and so all that needs to be done is to re-scale the limits. Let

$$x = \frac{z}{1-z^2} \rightarrow dx = \frac{1+z^2}{(1-z^2)^2} dz$$

Then the following transformation occurs:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{-1}^{1} \frac{1+z^2}{(1-z^2)^2} f\left(\frac{z}{1-z^2}\right) dz$$

Thus we see

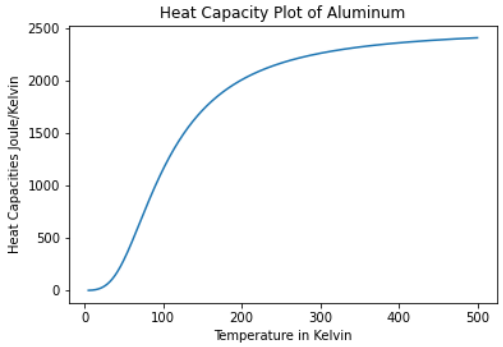
$$\langle x^2 \rangle = \int_{-\infty}^{\infty} x^2 |\psi_n(x)|^2 dx = \int_{-1}^{1} \frac{(1+z^2)z^2}{(1-z^2)^4} \left|\psi_n\left(\frac{z}{1-z^2}\right)\right|^2 dz = \frac{1}{2^n n! \sqrt{\pi}} \int_{-1}^{1} \frac{(1+z^2)z^2}{(1-z^2)^4} e^{-\frac{z^2}{(1-z^2)^2}} \left|H_n\left(\frac{z}{1-z^2}\right)\right|^2 dz$$

This form is suitable for numerical integration. We first rescale the integration limits to -1 and 1 and rewrite the integrand. This is implemented in the function called integrand(z,n), where n is the index of the wavefunction and the Hermite polynomial. We use Gauss-Legendre and Gauss-Hermite quadrature to compute the root neab square position for n = 5 and 100 sample points. For each case we get a value of approximate 2.345.

3 Results

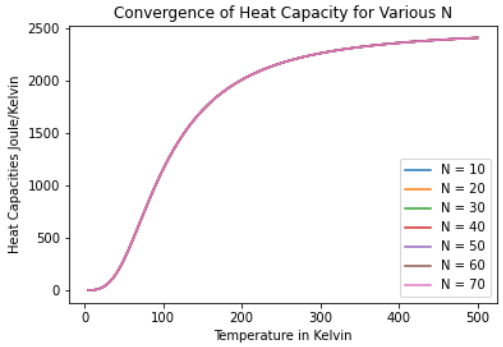
3.1 Exercise 1

Here is the the plot of the heat capacity.



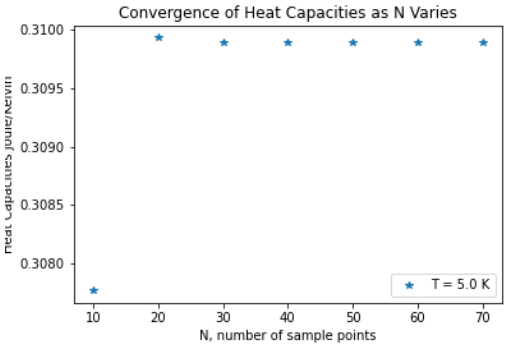
Heat Capacity of Aluminum sample according to the Debye theory.

Below is the plot showing overall convergence.

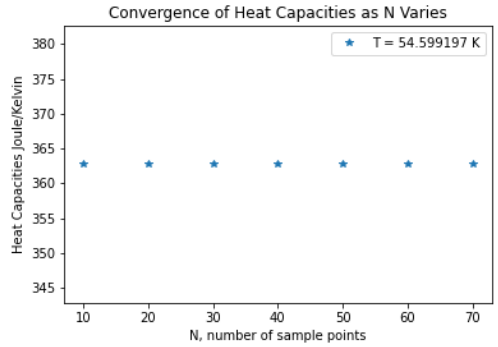


Testing the converge of the various heat capacity plots for the various sample sizes.

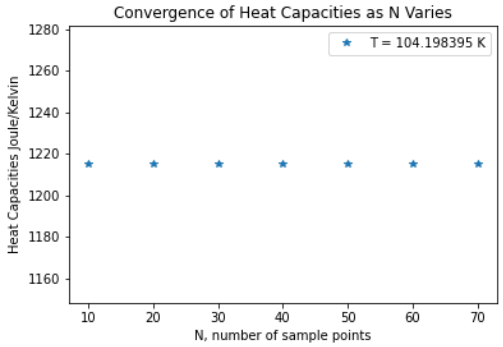
Finally, we present the plots testing individual, selected convergence.



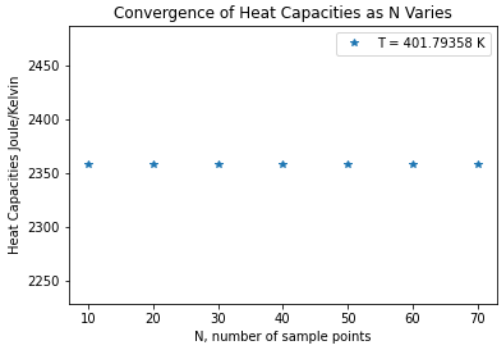
Convergence test for 5 Kelvin



Convergence test for 54.6 Kelvin



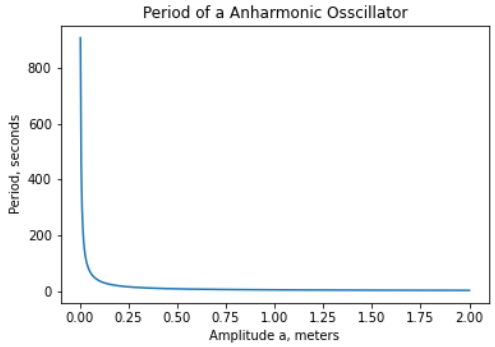
Convergence test for 104.2 Kelvin



Convergence test for 401.8 Kelvin

3.2 Exercise 2

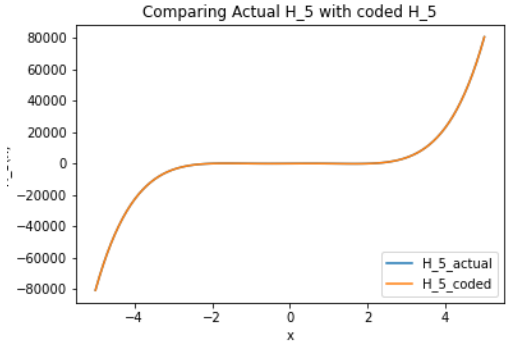
Below is a plot of the period versus amplitude. We explained the derivation of the period formula and the nature of this plot in the corresponding methods section.



Plot of the period versus starting position for the an-harmonic oscillator $V(x) = x^4$. For fun, we made the amplitude have units of meters and assumed that the mass was 1 kg. Then the period would have units have seconds.

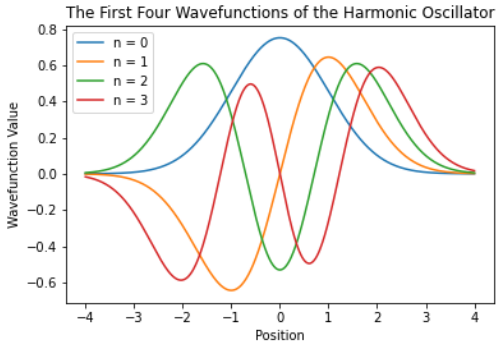
3.3 Exercise 3

Here is the plot showing us testing our Hermite polynomial generator.

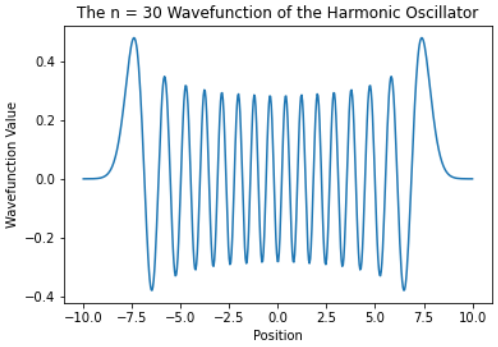


Comparing our $H_5(x)$ and the actual $H_5(x)$

Here is a plot of the first 4 wavefunctions, followed by a plot of the 31st wavefunction.



The ground state and the first three excited states of the harmonic oscillator.



Plot of ψ_{30}

Here is the result of $\langle x^2 \rangle$ for 100 sample points computed with Gauss-Legendre and Gauss-Hermite quadratures. This was done for $\psi_{n=5}(x)$. Both results match and agree with the problem's claim that the result should hover around 2.3.

```
The uncertainity in position for the n = 5 state is 2.3452078737858173 via Gauss-Legendre
quadrature.
The uncertainity in position for the n = 5 state is 2.3452078737858173 via Guass-Hermite
quadrature
```

The uncertainty in position for the $n = 5$ quantum state.

To decide whether the value of the uncertainty in position is close to exact, we decided to evaluate the aforementioned integral by using Simpson's rule with 800,001 sample points. The approximation error due to Simpson's rule for the integral $\int_a^b f(x)dx$ is given by equation 5.24 (see *Computational Physics* by Mark Newman)

$$\epsilon = \frac{h^4}{90}\left(f'''(a) - f'''(b)\right)$$

where h is the spacing between adjacent points. Because we chose 800,001 data points that are roughly between -1 and 1 and since the approximation error is of the order $\frac{h^4}{90}$, we see that the approximation error in the integral is ≈ 0 and so the resulting uncertainty should be very close to exact. Below is the uncertainty evaluated by Simpson's rule. We can see that it is pretty close to the ones computed with Gauss-Legendre and Gauss-Hermite quadratures.

```
In [9]: xSimpson = np.linspace(-0.99999999, 0.99999999, 800001)
   ...: n = 5
   ...: ySimpson = integrand(xSimpson, n)
   ...: integralSimp = integrate.simps(ySimpson, xSimpson)
   ...: rmsExact = np.sqrt((1/(2**n*m.factorial(n)*np.sqrt(np.pi)))*integralSimp)
   ...: print('The more exact value of the uncertainty in positon is ' + str(rmsExact))
The more exact value of the uncertainty in positon is 2.3452078799117153
```

The uncertainty in position for the $n = 5$ quantum state via Simpson's rule.