[PHYS-GA2000] Problem Set 4

Dylan Lane
Github: dl4729

October 03, 2023

Problem 1

Methods and Results

This problem follows immediately from the equation given:

$$C_v = 9V\rho k_b(\frac{T}{\theta_D})^3 \int_0^{\frac{\theta_D}{T}} \frac{x^4 e^x}{(e^x-1)^2} dx \tag{1}$$

(1). V is the volume, ρ is number density of the sample, k_b is Boltzmann's constant, and θ_D is the Debye temperature (in our case, 428K). The integrand is already expressed in dimensionless quantities, so Gaussian quadrature can be applied directly. Let $f = \frac{x^4 e^x}{(e^x-1)^2}$, and let w_k, x_k be the weights used in Gaussian quadrature and the roots of the N-th order Legendre polynomial. Then the integral in equation (1) can be very well approximated as

$$\int_0^{\frac{\theta_D}{T}} \frac{x^4 e^x}{(e^x-1)^2} dx \approx \sum_{k=1}^{N} w_k f(x_k) \tag{2}$$

The code provided in the textbook (1) (3) computes the weights and evaluation points for Nth order Gaussian quadrature; call these w and x. Since w and x are expressed as numpy arrays, this can be accomplished by np.sum(). Part (b) is simply repeated applications of this result. Qualitatively, it looks correct; the specific heat asymptotes at high temperature but looks similar to T^3 at low temperature.

Figures

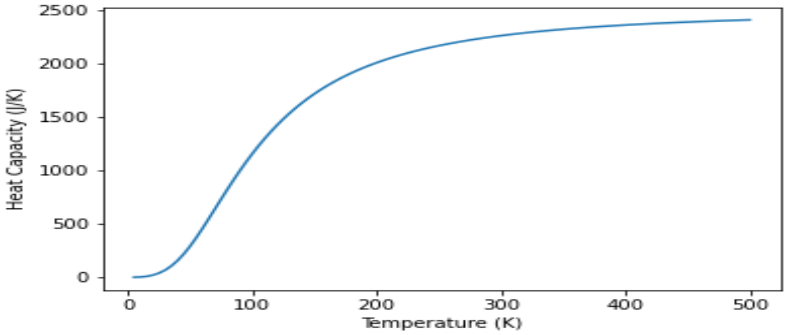

Figure 1: Heat Capacity (J/K) vs Temperature (K). Calculated using Gaussian quadrature with $N = 50$ function evaluations.

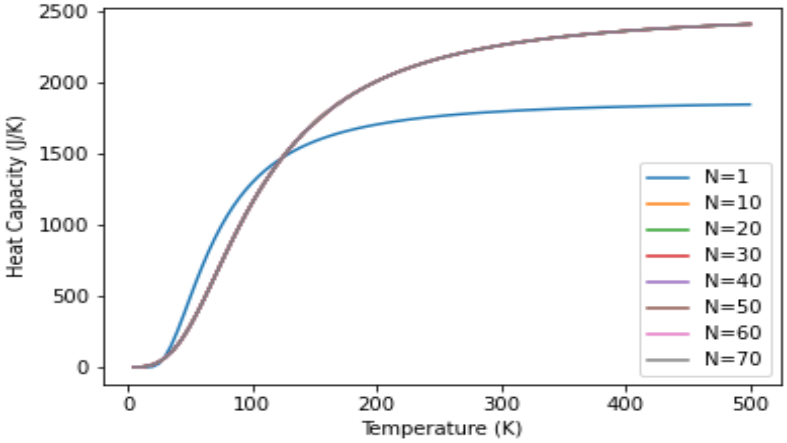

Figure 2: (Problem 2) Heat Capacity (J/K) vs Temperature (K). Calculated using Gaussian quadrature with $N = 1$ to $N = 70$ function evaluations.

Problem 2

Methods and Results

Consider the energy of an oscillator such that $E = \frac{1}{2}m(\frac{dx}{dt})^2 + V$ and let a be the amplitude. It follows that $E = V(a)$. Solving for $\frac{dx}{dt}$, we find

$$\frac{dx}{dt} = \sqrt{\frac{2(V(a) - V(x))}{m}} \tag{3}$$

Then, rearranging to isolate dt,

$$dt = \sqrt{\frac{m}{2}} \frac{dx}{\sqrt{V(a) - V(x)}} \tag{4}$$

As the oscillating particle travels from 0 to a, it takes a time $T/4$. Integrating both sides,

$$T/4 = \sqrt{\frac{m}{2}} \int_0^a \frac{dx}{\sqrt{V(a) - V(x)}} \tag{5}$$

Then as expected

$$T = \sqrt{8m} \int_0^a \frac{dx}{\sqrt{V(a) - V(x)}} \tag{6}$$

In this form (with all constants outside), we can apply Gaussian quadrature. As in the previous problem, $f(x) = \frac{1}{\sqrt{V(a)-V(x)}}$, where we use $V(x) = x^2$. We use the textbook's code (1) to again derive the roots and weights for 20th-order Gaussian quadrature and summing the product of the weights and the function evaluations at the roots of the 20th Legendre polynomial. This is a function of a; the resulting plot of the period between $a = 0$ and $a = 2$ is shown in figure (3).

As the textbook notes (1), as a goes to 0 the period diverges, while as a increases the period is slower (that is, the particle completes its motion faster). This is understandable from the x^4 potential. Recall from the description of the problem that the particle starts at position a so there is no question of overcoming the potential. Suppose first that a is large; the particle will have a very high energy since the potential is so steep. Then in the flat area $|x| < 1$, the potential is very low so the kinetic energy and thus speed is very high. Moreover, in the region $|x| > 1$ the gradient of the potential (force) is proportional to x^3, so the particle is accelerated very quickly in the opposite direction. These together make the particle's oscillations fast for sufficiently large a. Now suppose a is small, in particular $a < 1$. The gradient being proportional to x^3 means the particle is accelerated towards the center of the potential well very slowly, so the motion takes a very long time. In the limit of $a = 0$, this is clearly an equilibrium point with no force so no oscillation occurs at all (this is the same notion as an infinite period). Thus the behavior of figure (2) emerges from the choice of potential $V(x) = x^4$.

Figures

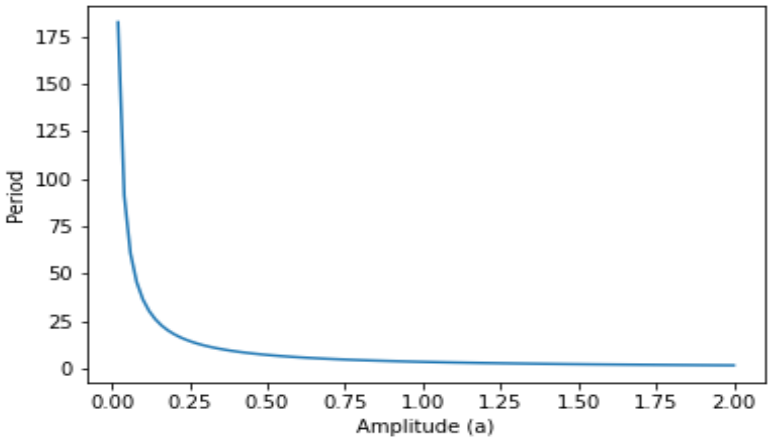

Figure 3: Period (units not specified in problem) vs. Amplitude (a)

Problem 3

Methods and Results

The functions of interest are the solutions to the quantum harmonic oscillator

$$\psi_n(x) = \frac{1}{\sqrt{2^n n! \sqrt{\pi}}} e^{-x^2/2} H_n \tag{7}$$

where H_n is the n-th Hermite polynomial. These can be generated from the recursion relation $H_{n+1} = 2xH_n(x) - 2nH_{n-1}(x)$. To code this, begin with $H_0 = 1$ and $H_1 = 2x$. Then create a list and append H_2 with the formula in terms of H_0 and H_1 and repeat this process in a for loop until the nth Hermite polynomial has been generated. The plots of the 0th, 1st, 2nd, and 3rd wavefunctions are shown in figure (3) and the 30th wavefunction in figure (4). Note that the recursion method provided in the hints was substantially slower at generating the 30th wavefunction than the for loop method I employed.

To compute the uncertainty in position, consider

$$< x_n^2 >= \int_{-\infty}^{\infty} x^2 |\psi_n(x)|^2 dx = \frac{1}{2^n n! \sqrt{\pi}} \int_{-\infty}^{\infty} x^2 H_n^2(x) e^{-x^2} dx \tag{8}$$

. The integrand is our $f(x)$ on which we want to employ Gaussian quadrature, however the bounds must first be adjusted. One map from $(-\pi/2, \pi/2) \to (-\infty, \infty)$ is x=$tan(z)$, $z \in (-\pi/2, \pi/2)$. Then $dx = sec^2(z)dz = \frac{dz}{cos^2(z)}(1)$. Substituing X in the integrand, we can now use the same method as problems 1 and 2 to find the weights and evaluation points (which range from $(-\pi/2, \pi/2)$). I used N=100; for $n = 5$, the resulting uncertainty in x is 2.3452078797796547 which agrees with the book's expected value of around 2.3 (1).

However, a closer inspection of equation (8) reveals that Gaussian quadrature may not be the best method for integration. Observe that $f(x)$ is of the form $f(x) = g(x)e^{-x^2}$, where $g = x^2 H_n^2(x)$. Evidently this is the form for which Gauss-Hermite quadrature is optimized for (1)(2). Using scipy.special's Hermite root function, for N points,

$$< x^2 > \approx \frac{1}{2^n n! \sqrt{\pi}} \sum_{k=1}^{N} w_k g(x_k) \tag{9}$$

where w_k, x_k are the k-th weights and roots for Gauss-Hermite quadrature of degree N. Gauss-Hermite already integrates from $-\infty$ to ∞ so no rescaling of the integral bounds is necessary as it was for Gaussian quadrature. For $n = 5$ and $N = 7$, the uncertainty was calculated to be 2.3452078799117184.

This should be an exact calculation; As defined above, $g(x)$ is a polynomial of degree $2n + 2$ where n is the order of the Hermite polynomial of interest. Gauss-Hermite should therefore be able to perfectly integrate $g(x)$ provided N (the number of roots) is sufficiently large (need $2N - 1 > 2n + 2$)(1)(2). If $n = 5$, so long as $N \geq 7$, the calculation should be perfect. However, I am unconvinced that my code succeeded in this; there is a nontrivial difference (order of 10^{-15}, a change bigger than machine error) between the uncertainty calculated using $N = 7$ and $N = 100$ or even higher.

Figures

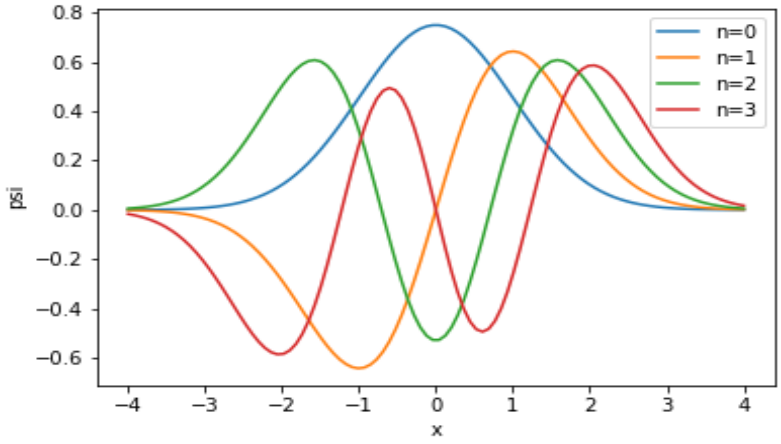

Figure 4: Wavefunction Solutions to Quantum Oscillator: n=0,1,2,3. Units of x, ψ unspecified.

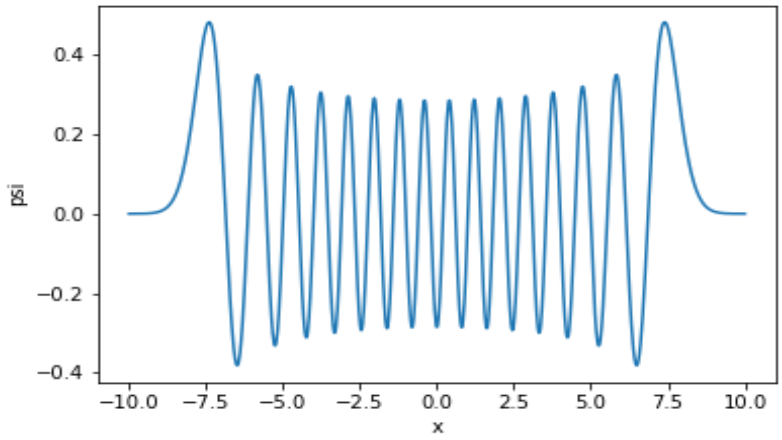

Figure 5: 30th Wavefunction Solution to Quantum Oscillator. Units of x, ψ unspecified.

References

[1] Newman, M. 2012, Computational Physics (Createspace Independent Pub)

[2] `https://blanton144.github.io/computational-grad/`

[3] `https://github.com/mcmorre/computational_TA/blob/main/PS2_hints.ipynb`