PHZ4151C: Exercise 6

Computational Physics Lab
Due March 1 → extended to Monday March 4

For each problem you will write one or more python programs. These programs should follow the Python 2.7.x coding and formatting conventions outlined for our course. You must hand in copies of the programs and outputs as prescribed in each problem.

In addition, you must submit all Python programs as an archive tgz file via email to phz4151c@hadron.physics.fsu.edu. Place copies of only your Python programs in a directory called <last_name>-exercise6/where <last_name> is your last name and use similar command as provided in earlier exercises. Include in your <last_name>-exercise6/ directory, the mymodules/ directory which includes your integration module.

1. Your Python module for integration:
Create a Python module for your implementation of numerical and Monte Carlo integration techniques. Chose an appropriate name for your module. The module should contain functions for trapezoidal and Simpson methods along with adaptive variants of each. The module should also contain a function for a generalized n-dimensional mean value Monte Carlo integration(see problem 5). Your module should include a test block which when executed as a program, tests and verifies each integration function. This module should be contained in a directory ./mymodules which is a subdirectory of your exercise 5 working directory. You will need to have a PYTHONPATH environmental variable set properly so that you can utilize your new module in programs. See lecture notes on *User Defined Modules* for additional information and coding expectations.

For full credit turn in a printout of your final user defined module, a printout of pydoc output for your module, and follow the proper instructions for electronic submission.

2. Adaptive integration:
(a) Write a program that uses the adaptive trapezoidal rule method from your integration module to calculate the value of the integral I to an approximate accuracy of $\varepsilon = 10^{-6}$ (i.e., correct to six digits after the decimal point). Start with one single integration slice and work up from there to two, four, eight, and so forth. Have your program print out the number of slices, its estimate of the integral, and its estimate of the error on the integral, for each value of the number of slices starting with N=2, until the target accuracy is reached. (Hint: you should find the result is around I = 0.45.)

$$I = \int_0^1 \sin^2(\sqrt{100x})\,dx$$

(b) Now modify your program to add the evaluation of the same integral using the Simpson's rule integration from your module, again to an approximate accuracy of $\varepsilon = 10^{-6}$. Using the formula provided in the lecture notes, starting with one integration slice, and working up from there, print out the results as in part (a) until the required accuracy is reached. You should find you reach the accuracy for a significantly smaller number of slices than with the trapezoidal rule calculation of part (a). Does this agree with expectations? Explain.

For full credit turn in a printout of your final program from part (b), printouts showing both adaptive integration methods in action (part (a) and part (b)) and clearly showing the output values, and the write up for part (b) discussion.

3. Heat capacity of a solid:
Debye's theory of solids gives the heat capacity of a solid at temperature T to be where V is the volume of the solid, ϱ is the number density of atoms, k_B is Boltzmann's constant, and θ_D is the so-called Debye temperature, a property of solids that depends on their density and speed of sound.

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

a) Write a Python function Cv(T) that calculates C_V for a given value of the temperature, for a sample consisting of 1000 cubic centimeters of solid aluminum, which has a number density of $\rho = 6.022 \times 10^{28}\ \mathrm{m}^{-3}$ and a Debye temperature of $\theta_D = 428$ K. Use Simpson's rule implemented in your integration module to evaluate the integral, with N = 50 sample points.

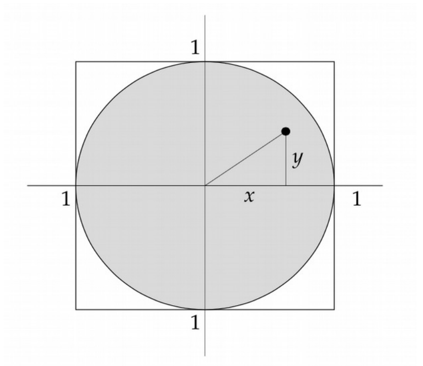
b) Use your function to make a graph of the heat capacity as a function of temperature from T = 5 K to T = 500 K.

For full credit, turn in a printout of your program, and your graph from part (b).

4. Rolling dice:
a) Write a program that generates and prints out two random numbers between 1 and 6, to simulate the rolling of two dice.

b) Modify your program to simulate the rolling of two dice a million times and count the number of times you get a double six. Divide the result by a million to get the fraction of times you get a double six. You should get a value close to, though probably not exactly equal to, 1/36. Your program should print the thrown dice values up to and including the first double six obtained. Have the program print out the number of double sixes, the number of dice throws, the double six fraction, and the number average number of throws per double six combination.

For full credit, turn in a printout of your program, the results from part (b).



5. Volume of a n-dimensional hypersphere:
This exercise asks you to estimate the volume of a sphere of unit radius with any dimensions using a Monte Carlo method. Consider the equivalent problem in two dimensions, the area of a circle of unit radius:

$$I = \int_{-1}^{+1}\int f(x,y)\,dx\,dy$$

The area of the circle, the shaded area shown, is given by the integral *I* where f(x, y) = 1 everywhere inside the circle and zero everywhere outside. In other words,

$$f(x,y) = \begin{cases} 1 & \text{if } x^2+y^2 \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

So if we didn't already know the area of the circle, we could calculate it by Monte Carlo integration. We would generate a set of N random points (x, y), where both x and y are in the range from −1 to 1. Then the two-dimensional version of the mean value integral for this calculation would be

$$I \simeq \frac{(1-(-1))^2}{N}\sum_{i=1}^{N} f(x,y).$$

a) Generalize this method to the n-dimensional case and write a function for your integration module and perform the Monte Carlo calculation with 1 million points of the volume of a sphere of unit radius in two and ten dimensions.

b) Modify your program to calculate the hyper volume as a function of the dimensions from n = 0 to n = 12. Print results to the screen and generate a graph of the hyper volume vs dimension.

If we had to do a ten-dimensional integral the traditional way, it would take a very long time. Even with only 100 points along each axis (which wouldn't give a very accurate result) we'd still have $100^{10} = 10^{20}$ points to sample, which is impossible on any computer. But using the Monte Carlo method we can get a pretty good result with a million points or so.

For full credit, turn in a printout of your program, the results from the various calculations in part (a), and the graph and results from part (b).

6. Gaussian integral via numerical integration

$$\int_{-\infty}^{\infty} e^{-x^2}\,dx = \sqrt{\pi}$$

Write a program which uses your integration module to performs the Gaussian integral $f(x)=e^{-x^2}$ from $-\infty$ to ∞ via numerical integration. The value for integral should be $\sqrt{\pi}$. Have your program compare your results of the numerical integration to the true numerical value.

For full credit, turn in a printout of your program, and the results.