Exercise #3 - Numerical Integration

Computational Physics Lab

Instructor: Prof. Sean Dobbs

January 30, 2020

*Due by 11 PM, February 7th**

Submission

For each problem, you will write one or more python programs. These programs should follow the Python coding and formatting conventions outlined for the course. Many problems will have additional questions to answer, or will ask for a record of the terminal showing the output when the program is run with various arguements. These answers can be given either in block comments in the prologue of the corresponding assignment, or in clearly labeled text files. The terminal output should contain some information on the username and machine you are running on (this will often be in the command prompt).

You will be evaluated based on the files contained in your remote GitHub repository at the due date. You should and commit all the files you created to your local repository on a regular basis. You should do this by adding any new or modified files using "git add", and then finalizing changes using "git commit". Remember that "git status" will give you information on which files in your repository. have been changed. Remember to add short, but useful comments when performing a commit. When you are finished with the exercise, push the current status of your local repository to the remote repository on GitHub using the command "git push -u". You are encouraged to push your local files to the remote repository periodically before you are done, and certainly well before the deadline, if possible.

1. Simpson's Rule

(a) Write a program to calculate an approximate value for the integral $\int_0^2 (x^4 - 2x + 1)\, dx$ from Example 5.1, but using Simpson's rule with 10 slices instead of the trapezoidal rule. You may wish to base your program on the trapezoidal rule program on page 142.

(b) Run the program and compare your result to the known correct value of 4.4. What is the fractional error on your calculation?

(c) Modify the program to use a hundred slices instead, then a thousand. Note the improvement in the result. How do the results compare with those from Example 5.1 for the trapezoidal rule with the same numbers of slices?

For full credit turn in your program, plus your results and a brief discussion of how they compare with the trapezoidal rule.

2. Gaussian Error Function

Consider the integral

$$E(x) = \int_0^x e^{-t^2}\, dt.$$

a) Write a program to calculate $E(x)$ for values of x from 0 to 3 in steps of 0.1. Choose the trapezoidal method for performing the integral and choose a suitable number of slices.

b) When you are convinced your program is working, extend it further to make a graph of $E(x)$ as a function of x. If you want to remind yourself of how to make a graph, you should consult Section 3.1, starting on page 88.

Note that there is no known way to perform this particular integral analytically, so numerical approaches are the only way forward.

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

3. Adaptive Integration

Consider the integral

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

a) Write a program that uses the adaptive trapezoidal rule method of Section 5.3 and Eq. (5.34) to calculate the value of this integral to an approximate accuracy of $\epsilon = 10^{-6}$ (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 N, until the target accuracy is reached. (Hint: You should find the result is around $I = 0.45$.)

b) Now modify your program to evaluate the same integral using the adaptive Simpson's rule method. Have your program print out a similar table of values as in part(a). 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 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 discussion from part (b).

4. Heat Capacity of a Solid

Debye's theory of solids gives the heat capacity of a solid at temperature T to be

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

where V is the volume of the 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.

(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 Gaussian quadrature to evaluate the integral, with $N = 50$ sample points.

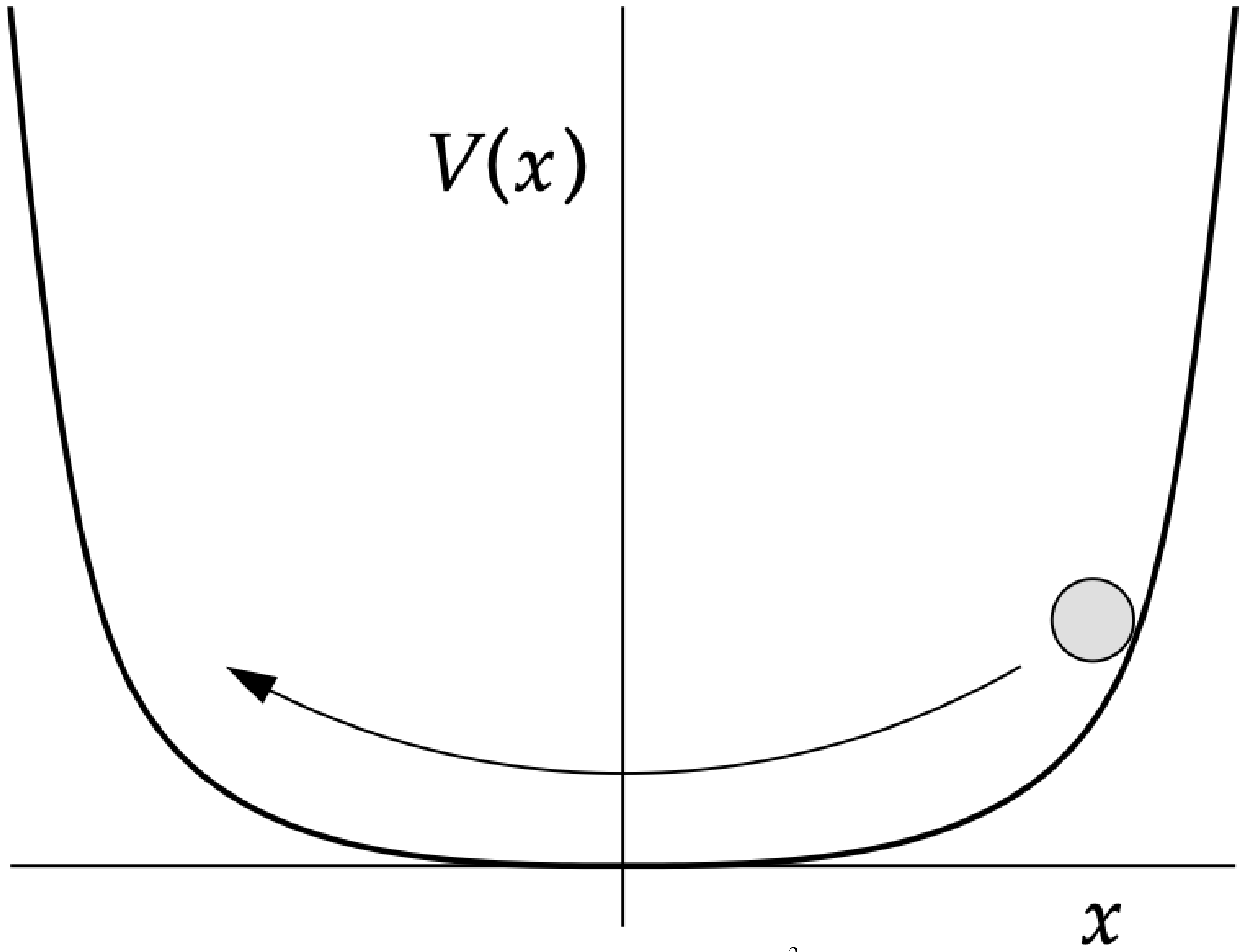
(b) Use your function to make a graph of the heat capacity as a function of temperature from $T = 5$ K to $T = 500$ K.

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

EC. Period of an anharmonic oscillator - Extra Credit (+5)

The simple harmonic oscillator crops up in many places. Its behavior can be studied readily using analytic methods and it has the important property that its period of oscillation is a constant, independent of its amplitude, making it useful, for instance, for keeping time in watches and clocks. Frequently in physics, however, we also come across anharmonic oscillators, whose period varies with amplitude and whose behavior cannot usually be calculated analytically.

A general classical oscillator can be thought of as a particle in a concave potential well. When disturbed, the particle will rock back and forth in the well:



The harmonic oscillator corresponds to a quadratic potential $V(x) \propto x^2$. Any other form gives an anharmonic oscillator. (Thus there are many different kinds of anharmonic oscillator, depending on the exact form of the potential.)

One way to calculate the motion of an oscillator is to write down the equation for the conservation of energy in the system. If the particle has mass m and position x, then the total energy is equal to the sum of the kinetic and potential energies thus:

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

Since the energy must be constant over time, this equation is effectively a (nonlinear) differential equation linking x and t.

Let us assume that the potential $V(x)$ is symmetric about $x = 0$ and let us set our anharmonic oscillator going with amplitude a. That is, at $t = 0$ we release it from rest at position $x = a$ and it swings back towards the origin. Then at $t = 0$ we have $dx/dt = 0$ and the equation above reads $E = V(a)$, which gives us the total energy of the particle in terms of the amplitude.

(a) When the particle reaches the origin for the first time, it has gone through one quarter of a period of the oscillator. By rearranging the equation above for dx/dt and then integrating with respect to t from 0 to $\frac{14}{T}$, show that the period T is given by

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

(b) Suppose the potential is $V(x) = x^4$ and the mass of the particle is $m = 1$. Write a Python function that calculates the period of the oscillator for given amplitude a using Gaussian quadrature with $N = 20$ points, then use your function to make a graph of the period for amplitudes ranging from $a = 0$ to $a = 2$.

(c) You should find that the oscillator gets faster as the amplitude increases, even though the particle has further to travel for larger amplitude. And you should find that the period diverges as the amplitude goes to zero. How do you explain these results?

For full extra credit turn your program, the derivation for part(a), the graph for part (b), and the discussion for part (c).