Exercise #5 - Numerical Derivatives, Modules, and Classes

Computational Physics Lab ¶

Instructor: Prof. Sean Dobbs

February 18, 2020

*Due by 11 PM, February 26th**

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. Integration Library

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. 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 for additional information and coding expectations.

For full credit turn in your program, plus a copy of the pydoc output for your module.

2. The Gamma Function

A commonly occurring function in physics calculations is the gamma function $\Gamma(a)$, which is defined by the integral

$$\Gamma(a) = \int_0^\infty x^{a-1} e^{-x}\, dx.$$

There is no closed-form expression for the gamma function, but one can calculate its value for given a by performing the integral above numerically. You have to be careful how you do it, however, if you wish to get an accurate answer.

a) Write a program to make a graph of the value of the integrand $x^{a-1} e^{-x}$ as a function of x from $x = 0$ to $x = 5$, with three separate curves for $a = 2$, 3, and 4, all on the same axes. You should find that the integrand starts at zero, rises to a maximum, and then decays again for each curve. Note that that the maximum falls at $x = a - 1$.

b) Most of the area under the integrand falls near the maximum, so to get an accurate value of the gamma function we need to do a good job of this part of the integral. We can change the integral from 0 to ∞ to one over a finite range from 0 to 1 using the change of variables in Eq. (5.67), but this tends to squash the peak towards the edge of the $[0, 1]$ range and does a poor job of evaluating the integral accurately. We can do a better job by making a different change of variables that puts the peak in the middle of the integration range, around $1/2$. We will use the change of variables given in Eq. (5.69), which we repeat here for convenience:

$$z = \frac{x}{c + x}.$$

For what value of x does this change of variables give $z = 1/2$? Hence what is the appropriate choice of the parameter c that puts the peak of the integrand for the gamma function at $z = 1/2$?

c) Before we can calculate the gamma function, there is another detail we need to attend to. The integrand $x^{a-1} e^{-x}$ can be difficult to evaluate because the factor x^{a-1} can become very large and the factor e^{-x} very small, causing numerical overflow or underflow, or both, for some values of x. Write $x^{a-1} = e^{(a-1)\ln x}$ to derive an alternative expression for the integrand that does not suffer from these problems (or at least not so much). Explain why your new expression is better than the old one.

d) Now, using the change of variables above and the value of~c you have chosen, write a user-defined function `gamma(a)` to calculate the gamma function for arbitrary argument a. Use whatever integration method you feel is appropriate. Test your function by using it to calculate and print the value of $\Gamma(\frac{3}{2})$, which is known to be equal to $\frac{1}{2}\sqrt{\pi} \simeq 0.886$.

e) For integer values of a it can be shown that $\Gamma(a)$ is equal to the factorial of $a-1$. Use your Python function to calculate $\Gamma(3)$, $\Gamma(6)$, and $\Gamma(10)$. You should get answers closely equal to $2! = 2$, $5! = 120$, and $9! = 362\,880$.

For full credit turn in your program, and your answers to the above questions.

3. Differential Calculator

Create a differential calculator object class which uses the central difference to calculate the derivative of a given function $f(x)$ at a given value of x. The `__init__` constructor should include arguments for the referencing function `func(x)` and central difference step size h which should have a default value of 1e-08. Define a `__call__(self, x)` member function which implements the central difference of the function at the value x. Within the main part of your program, define a function $f(x)$ that returns the value $1 + 1/2 \tanh 2x$. Declare a differential calculator object for the numerical derivative of $f(x)$. For comparison, define a function for the analytic derivative of $f(x)$ and make a graph with your numerical result and the analytic results on the same plot for the range $-2 \leq x \leq 2$.

For full credit turn in your program and your graph.

4. Radioactive Decays

This problem is a straightforward application of a forward difference to numerically solve a differential equation. Given $N(t)$ radioactive nuclei, they will decay randomly according to the following equation:

$$\frac{dN}{dt} = -\frac{N(t)}{\tau}$$

By replacing dN/dt with the forward difference $(N(t+h) - N(t))/h$, one obtains a numerical solution for $N(t+h)$. Given initial conditions (i.e. $N(t=0)$), one can obtain numerical values of $N(t)$ for all later times. This differential equation can be solved analytically as $N(t) = N(0)\exp(-t/\tau)$, where $N(0)$ is the initial number (or fraction) of radioactive nuclei. This solution allows one to compare our numerical results with the exact solution.

a) Write a program to numerically solve for the time dependence $N(t)$ from $0.0\text{s} \leq t \leq 15.0\text{s}$ assuming $N(0) = 100\%$ and $\tau = 2$ s. Do this for the following values of h: 1.0 s, 0.1 s, and 0.01 s. Graphically compare your results to the exact solution as a function of time. Also study the accuracy by plotting the fractional error vs. t. Overlay on one figure the graphs for the different values of h.

b) Using $h = 0.01$ s, write another program to plot the time dependence of $N(t)$ for τ = 5.0 s, 3.0 s, 1.0 s, 0.1 s, and 0.01 s. Briefly discuss the accuracy of your results. If there are any problems explain.

c) Consider a system of a parent nucleus, **P**, and a daughter nucleus, **D** both radioactive. The coupled equations which describe their decays are as follows:

$$\frac{dN_P}{dt} = -\frac{N_P(t)}{\tau_p}$$

$$\frac{dN_D}{dt} = \frac{N_P(t)}{\tau_p} - \frac{N_D(t)}{\tau_D}$$

Write a program to numerically solve the above coupled equations and plot the time dependence of N_P and N_D for τ_P = 2.0 s and τ_D = 0.02 s, 2.0 s and 200.0 s. Assume you start out with 100% parent nuclei and no daughter nuclei. Graph all results overlaid on one figure. Qualitatively explain the behavior of N_D for situations in which $\tau_P \gg \tau_D$, $\tau_P \approx \tau_D$, and $\tau_P \ll \tau_D$.

For full credit turn in a your 3 final programs, your graphs from parts a, b and c, and the discussions for part b & c.