PHZ4151C: Exercise 5

Computational Physics Lab
Due Feb 22

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>-exercise4/ where <last_name> is your last name and use similar command as provided in earlier exercises.

1. Floating point precision:

(a) Write a program to find the largest positive floating-point number x, to within a factor of 2, such that when you add the value x to the value 1.0 the resulting value is 1.0. The value obtained for x is an approximation (to within a factor of two) for the precision of 64 bit floating point numbers.

(b) Modify your program to use numpy floating point data types: float16, float32, float64, and float128. Obtain the floating point precision, to within a factor of two, for all of the above float data types. The statement " var = numpy.float128(1.0)" will create a 128 bit floating point variable with the value 1.0.

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

2. Calculating derivatives:

Suppose we have a function f (x) and we want to calculate its derivative at a point x. We can do that with pen and paper if we know the mathematical form of the function, or we can do it on the computer by making use of the definition of the derivative:

$$\frac{df}{dx} = \lim_{\delta \to 0} \frac{f(x+\delta) - f(x)}{\delta} \quad .$$

On the computer we can't actually take the limit as δ goes to zero, but we can get a reasonable approximation just by making δ small.

(a) Write a program that defines a function f(x) which returns the value x(x − 1). In the same program define a generic derivative function of a real variable which takes as function arguments: the name of a function, a value of x for which the derivative is calculated at, and a value for the limit variable δ. Next have your program calculate and print the derivative of the function f(x) at the point x = 1 using the formula above with $\delta = 10^{-2}$. Calculate the true value of the same derivative analytically and compare with the answer your program gives. The two will not agree perfectly. Why not?

(b) Create a second program which calculates the derivative for $\delta = 10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}, 10^{-12},$

10^{-14}. 10^{-16}, and 10^{-18}. You should see that the accuracy of the calculation initially gets better as δ gets smaller, but then gets worse again. Why is this?

For full credit, turn in a printout of your two programs, the results from the various calculations, and your answer to the questions in part (a) and part (b)

We will look at numerical derivatives in more detail later in the course, when we will study techniques for dealing with these issues.

3. 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.

(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 number of slices?

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

4. Gaussian error function:

Consider the integral

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

(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 a printout of your program, plus a printout of your graph from part (b) .