Christina Schumm

PHY 905 Final Project

Throughout this course I have learned a lot about coding through C++, but I did not become too comfortable with any of the other coding languages. Many students in class would mention their love for Python, which I have only ever seen a handful of times and at a very basic level. I thought it would be interesting to use this final project to teach myself some Python. In particular, to compare and contrast C++ by taking an already made code from class and converting it into another language. I chose to convert derivative\_test.cpp into derivative\_test\_error.py and added derivative\_test.py to calculate and graph the function e^-x, the exact derivative and the approximated derivatives using the 3 methods learned from class.

I ended up discovering why people prefer Python over C++. Although it took me a while to figure out the language, my Python code ended up being more concise, and in general just easier to read. I also expected Python to have different precision than C++, but if you compare the error graphs that I created, you will see that they have the exact same precision. I had heard that C++ is “computationally better”, so I expected the precision to be better. I assumed that C++ would be able to approximate the derivatives with much smaller step sizes than Python, but that did not happen. In conclusion, they yield the same results, but I prefer Python because it is easier to understand.

The errors also scale as expected. The forward difference has a slope of 1, central difference has a slope of 2 and extrapolated difference has a slope of 4. This follows for both the C++ error plot and the Python error plot. These are both expected according to the notes when using double precision. I created derivative\_test.py to be changed by the user. The point is to use derivative\_test\_error.py to choose a function and then find the optimum step size, once the optimum step size is found for each method, the user can plug those values into derivative\_test.py. This should give them the best results for the function derivative they are trying to estimate.

If I were to choose between C++ and Python, I would choose Python. At least when calculating or estimating derivatives because they have the same precision. The code may execute slightly slower compared to C++, but for a code that is short it is not a huge problem. I imagine that C++ is better for very computationally rigorous calculations. Python also does not require a make-file and I learned how to easily incorporate pyplot. Having Python directly plot graphs was much easier than having a separate .plt file the way we had to do it for C++. Overall, for shorter codes, Python is the way to go.