

Tutorial problems for Lectures 1, 2, and 3. Due Monday April 23, 2018.

Problem 1: Make a new text file with your name, email address, what you're working on for your honours project, plus a few things you'd like to learn in the course. Initialize a git repository and commit this file. Please name the repository Tutorial_1 (5)

Problem 2: Put your answers to the other tutorial questions into another text file. Add this to the repository also. (5)

Problem 3: Make a github account and push the repository onto github. Email me (jonathan.sievers@gmail.com) your github name so I can have a look at your file/answers. (10)

Problem 3: Write a python script to make a vector of n evenly spaced numbers between 0 and $\pi/2$. i.e. $x[0]=0$, $x[-1]=\pi/2$ (5)

Use this vector to integrate $\cos(x)$ from 0 to $\pi/2$ for a range of number of points using the simple method. Include 10,30,100,300,1000 points between 0 and $\pi/2$. How does error scale with number of points? Your answer should look like $err \propto n^\alpha$, say what α is (5)

Problem 4: Python supports array slicing - $x[5:10:2]$ will take points 5,7,9 from x . $x[5::2]$ will take points 5,7,9 from x . How can I take all odd points from an array? How can I take all even points from an array, but skipping the first and last points? (5)

Problem 5: Write a python function to integrate this vector using Simpsons rule. How does error scale with number of points? How many points did we need to use in part 2 to get same accuracy as 11 points with Simpsons rule? (10)

Problem 6: Plot the errors as a function of number of points using Simpsons rule and standard sum. You will want to use a log scale here - look at logplot.py in the github distribution (5)

Bonus: The scipy module has built in integration functions in `scipy.integrate`. The `quad` routine will do numerical integrals. `quad` will try to put its effort where the function changes quickly, which can save huge amounts of computation for some problems.

B1: Look at `scipy_quad_example.py`, which uses `scipy` to integrate our Gaussian function over two different ranges. The integrals should be (almost) identical - yet they are not. Can you figure out why? (5)

B2: Can you write another function that will always give the correct answer to this integral? (5) Hint - you may want to do two integrals instead of one.