MTH 3300
Fall 2018
E. Fink
October 7, 2018

Homework 5

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load these template files, and write the following programs. Submit your source code to me via Blackboard, in `.py` format; do NOT send any other files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

Be sure to read the SPECIFICATIONS carefully for all problems! And write comments!

1) trapezoid.py

If you've taken Calculus II, you've probably encountered trapezoidal rule for approximating definite integrals:

$$\int_A^B f(x)dx \approx \frac{\Delta x}{2}\left(1f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + \ldots + 2f(x_{N-1}) + 1f(x_N)\right)$$

where

$$\Delta x = \frac{B-A}{N}$$

and

$$x_i = A + i\Delta x$$

Here, N is the number of pieces you break the interval $[A, B]$ into; the more pieces, the better. Pay attention to the coefficients: 1 for the first term and the last term; all the rest are two. (We'll discuss in class why this approximates the integral well).

So, for example: if I want to approximate $\int_1^2 x^2 dx$ with $N = 4$ steps, I have:

$$\Delta x = \frac{2-1}{4} = \frac{1}{4}$$

$$x_0 = 1,\ x_1 = 1 + \frac{1}{4} = 1.25,\ x_2 = 1 + \frac{2}{4} = 1.5,\ x_3 = 1 + \frac{3}{4} = 1.75,\ x_4 = 1 + \frac{4}{4} = 2$$

$$\int_1^2 x^2 dx \approx \frac{1/4}{2}\left(1(1)^2 + 2(1.25)^2 + 2(1.5)^2 + 2(1.75)^2 + 1(2)^2\right) = 2.34375$$

The exact answer is of course $\frac{7}{3} = 2.33333333\ldots$, so we're pretty close.

I want you to create a program that asks the user to enter in an integer N, and uses the trapezoidal rule with N intervals to approximate the value of

$$\int_{-1}^{1} f(x)dx$$

where $f(x)$ is defined by

$$f(x) = \begin{cases} (1+x)\cos(\pi x), & \text{if } x < 0 \\ (1-x)\cos(\pi x), & \text{if } x \geq 0 \end{cases}$$

You should write a Python function to represent this mathematical function $f(x)$. Warning – it's probably a bad idea to have this function do anything other than take a value of x as input, and return $f(x)$. This function should not contain any loops within its definition; nor should its definition have anything to do with N or the limits of integration. (It *should* contain an if statement, though.)

Don't be intimidated by this problem – it's just a big sum! But do be careful about the details, like initializing your sum, using the right number of terms, and providing the right coefficient for each term.

Finally, **make sure you understand the trapezoidal rule well enough to perform some calculations by hand**. It can be tedious, sure, but if you can't use the trapezoidal rule to estimate an integral by hand, you'll never be able to code it.

Test values: when $N = 10$, you should have exactly 0.4188854 (rounded to 7 digits). If you're slightly off, you might be missing a term!

Specifications: your program must

- ask the user for a value of N, where N is a positive integer (you may assume the user complies).
- use a Python function to represent the function $f(x) = \begin{cases} (1+x)\cos(\pi x), & \text{if } x < 0 \\ (1-x)\cos(\pi x), & \text{if } x \geq 0 \end{cases}$.
- print an estimate for the value of $\int_{-1}^{1} f(x)dx$ using the trapezoidal rule with N steps, with $f(x)$ the function given above.

***Challenge**: instead of asking the user for an integer N, ask the user to enter an error tolerance. Then, have your program determine an N on its own, so that the error between the trapezoidal rule estimate and the true value of the integral is no more than the given error tolerance. There is an inequality related to the trapezoidal rule that can help you be sure about whether your N is sufficient.*

---

2) subway.py

You have been hired to fix the subways! Your first task is to see if you can build a preliminary model to understand exactly how they are behaving.

Your preliminary model will be super-simplified:

- Each train will make 20 stops before reaching its final destination, where everyone gets off. It will start out empty.
- At each of the 20 stations, a random number of people between 1 and 100 will attempt to get on the train. (Each number between 1 and 100 is equally likely.) Nobody exits.
- The train has capacity 1000. Once the train has 1000 passengers, all further passengers will be refused entry. They will then be consider *late*, because they have to wait for the next train!

Your job: using a simulation, estimate the average number of late passengers (summed over all stations) per train.

Here is a bit more direction: you should simulate 100,000 trains. For each simulated train, keep track of the current load and the total number of late passengers, which should both start out as 0. You should then pick 20 different random numbers – each one should be between 1 and 100, with each value equally likely – which represent the number of passengers at each station. For each one, add that random number onto the current load. If this brings the current load to over 1000, the current load should be reset to be 1000, and the excess should be added to the number of late passengers.

As you proceed through these 100,000 simulated trains, you should keep track of the total number of late passengers. Then, the average number of late passengers per train is simply this total divided by 100,000.

After the simulations are complete, print out your program's estimates.

Specifications: your program must

- NOT ask the user for any input.
- print out an estimate of the average number of late passengers per train, using the above model.
- **use 100,000 simulations with random numbers – no credit for theoretical solutions**. Your answers should **not** be exactly the true average, and you should **not** even get the same answers each time you run it. If this point isn't obvious, please ask for clarification!

***Challenge**: display a histogram showing the frequencies of each amount of late passengers. Or, find a 95% confidence interval for the true value of the average based on your simulations. Or do both. (You might want to use Google to learn how to create a nice histogram or compute a confidence interval.)*