Lab 3: Numerical Integration

Objectives

By the end of this lab you should be able to:

- Write a program to numerically integrate a function `f(x)` from lower limit `a` to upper limit `b`.
- Write a function that has as one of its arguments another function.

Prelab Questions

Read the rest of this lab and only then answer the following questions:

1. In class we worked out the are of one trapezoid (from a to $a+h$) to be $\frac{h}{2}\left(f(a)+f(a+h)\right)$.
    1. Like we did in lecture for the midpoint rule, write down the each of the 4 areas if the region is broken into 4 intervals.
    2. Add the four up and see if you can find a pattern in the terms.
2. Calculate the definite integral of $\frac{1}{3}x^3 - \frac{11}{4}x^2 + 7x$ from x=a to x=b analytically (pencil and paper)(or with a program like Maple or Wolfram Alpha).
3. Sketch the integrand of the Boltzmann integral (see Part 4, below). Feel free to write a small piece of Python code to plot it if you want!

Your pre-lab will be checked by the instructor at the beginning of lab.

Part 1: Set up a test function and integral

- Define a simple function in your notebook called `test_function`, that accepts one variable, `x`, and returns $\frac{1}{3}x^3 - \frac{11}{4}x^2 + 7x$.
- Write a function called `integral_of_test_function` that takes a lower limit `a` and an upper limit `b` and returns the integral of $\frac{1}{3}x^3 - \frac{11}{4}x^2 + 7x$ using the analytical solution you worked out in the pre-lab.
- Verify that `integral_of_test_function` returns the value you expect in a couple of cases. You choose the cases and, in comments in your code, say what value you expected for the integral between the limits you chose and what value your code actually gave you.

Part 2: Implementing the Trapezoidal Rule

- In a separate file called `integrators.py` write a function called `trapezoidal\_rule\_loop` that takes four arguments, one of which is the function to integrate, and returns the integral using the trapezoidal rule. *Use a for loop inside your function.* The arguments are:
    - a lower limit `a`,
    - an upper limit `b`,
    - an argument named `integrand` that will be a function

  - an optional argument `N`, the number of intervals, with a default value of 10.
- In your main program perform the integration using the `trapezoidal_rule_loop` function with a few different values of `N`. Compare to the result you get from `test_function_integral`, and comment.
- Write a function `trapezoidal_rule` that implements the trapezoid rule *without a for loop.*

Part 3: Simpson's Rule

- In `integrators.py` add a function called `simpsons_rule` that takes the same arguments as `trapezoidal_rule` but calculates the integral using Simpson's rule. Though Simpson's rule is described in Newman (Eq. 5.9), your Simpson's rule should simply call the function `simps` from the scipy package. Import it with: `from scipy.integrate import simps`
- You will almost certainly want to google the documentation for that function.
- For several values of `N` calculate the integral of your test function using the trapezoid rule and Simpson's rule, and compare to what you get from the analytic result.
- In your notebook perform the integration using the `trapezoidal_rule` function with a few different values of `N`. Compare to the result you get from `test_function_integral` and `trapezoidal_rule` with the results from `simpsons_rule` and comment.
- In your notebook, in a markdown cell, answer this question: For the same value of `N`, which method is more accurate, trapezoidal rule or Simpson's rule?

Part 4: The Stefan-Boltzmann Integral

- There is a classic physics problem known as the blackbody radiation problem that you will likely see several times as an undergraduate. The correct determination of the total power output of a blackbody radiation source involves evaluation of the Stefan-Boltzmann integral:

$$\int_0^\infty \frac{x^3}{\exp(x) - 1} dx = \frac{\pi^4}{15}$$

- Write a function in your notebook that takes a single argument `x` and returns the value of the *integrand*, i.e. the value of $x^3/(\exp(x) - 1)$,in the Stefan-Boltzmann integral. **Suggestion:** $x = 0$ is a special case where we know the integrand has a limit of 0 but if you try to compute it, it will fail. You can deal with this special case in your function if you wish.

- Choose values for the upper and lower limits of integration and the number of intervals, then calculate the integral using your trapezoidal rule and Sampson's rule.

- In your comments explain why you chose the upper and lower limits you did. **BIG HINT:** In class, we may have discussed re-writing the functions to deal with infinite bounds. In this case, you can tackle the limit being 'tricky' much more easily if you think about the pre-lab question where you sketched the function. What happens to the integral in the limits?
- Adjust your choices for the limits until the result no longer changes significantly. . . .and explain in your comments what you mean by "significantly."
- Compare your numerical result to the expected result and comment on it.

Optional, if time permits: Writing an Adaptive Integrator

- Often when doing an integral numerically what you care about is whether the you have done the integral reasonably accurately, not the exact number of intervals used. In this part you will write a function that automatically changes the number of intervals until changing that number does not change the value of the integral.
- Define a new function called `adaptive_integrator` that takes the arguments below. It should return the value of the integral.
  - a lower limit `a`,
  - an upper limit `b`,
  - an argument named `integrand` that will be the function whose integral you want to find.
  - an argument named **integrating_method** that will be the function you want to use to numerically calculate the integral.
  - **tolerance**, an optional argument with default value of **1e-4**, which is the margin of error you are willing to allow in the calculation of the integral.
- Inside the idea is to have a loop that changes `N` and iterates until successive values differ by an amount less than the tolerance. Start with 10 intervals and have `N` change by a factor of two with each successive approximation. Have the program print both the `N` value and the integral approximation.
- Reduce the tolerance and observe how large `N` must become for Simpson's rule to yield an answer within tolerance. This is a characteristic of the algorithm, and Simpson's rule converges more rapidly than the Trapezoidal rule.

What to turn in at the end of lab:

Submit the Python code in GitHub. Make sure to upload both your notebook and your `integrators.py` code.