EE2703 - Applied Programming Lab Assignment-6: Trapezoidal rule integration: Python, Cython and Numpy

Prajwal Vijay EE23B057

October 2024

1 Introduction

In this assignment our goal was to perform and optimize the trapezoidal integration method, using cython, python and numpy. Here we try to perform the definite integration of a function, by approximating the area under the function f(x), using a bunch of trapezoids. Divide the interval [a,b] into n sub-intervals and sum them up to get a trapezoid.

2 Problem Description

We are given a function $f(x)$, the interval $[a, b]$ and the number of trapezoids n. Given all this info, our tasks are as follows:

- Find the definite integral over the given parameters, for the given function $f(x)$
- This integral must be calculated using cython, python and numpy.
- For cython, try to optimize the code as much as possible by using syntaxx thats closer to C, for numpy use the *np.trapz* function.
- Test the accuracy of your results against the expected results of analytical integration.

3 Details of the Implementation

- In python, I just divide the interval [a,b],into n parts,and find the area of trapezoids, under each of them.
- I have written the entire cython code, in one cell, since this maintains the benefits of cdef, keeps related code together, and avoids linking issues.

- I have defined a pointer to the function_type data. This helps in passing functions as parameters in C, python functions cannot be directly passed this way. Only functions defined by cdef, can be passed.
- In order to access our Cython functions from outside, we define wrapper functions
- Numpy based functions were also defined, they use the np.trapz function for performing integration

4 Results

All times mentioned below are per loop.

$f(x)$	n	a	b	$time_{py}$	$time_{cy}$	$time_{np}$
x^2	10^7	0	1	3.63 s ± 76.2 ms	15 ms ± 126 µs	143 ms ± 258 µs
$sin(x)$	10^7	0	π	15.2 s ± 203 ms	220 ms ± 1.47 ms	231 ms ± 1.02 ms
e^x	10^7	0	1	14.7 s ± 239 ms	151 ms ± 584 µs	196 ms ± 1.21 ms
$1/x$	10^7	1	2	2.99 s ± 78.5 ms	24.1 ms ± 328 µs	142 ms ± 1.13 ms

Table 1: Timing results for different functions

$f(x)$	n	a	b	$Accuracy_{py}(\%)$	$Accuracy_{cy}(\%)$	$Accuracy_{np}(\%)$
x^2	10^7	0	1	7.693e-12	7.693e-12	5.162e-13
$sin(x)$	10^7	0	π	2.398e-12	2.375e-12	8.992e-13
e^x	10^7	0	1	1.421e-12	1.421e-12	5.169e-14
$1/x$	10^7	1	2	2.053e-11	2.053e-11	1.762e-13

Table 2: Accuracy results for different functions

$f(x)$	n	a	b	$time_{py}$	$time_{cy}$	$time_{np}$
x^2	10^7	0	10	4.11 s ± 188 ms	15.9 ms ± 116 µs	158 ms ± 1.37 ms

Table 3: Accuracy results for the given test case

$f(x)$	n	a	b	$Accuracy_{py}(\%)$	$Accuracy_{cy}(\%)$	$Accuracy_{np}(\%)$
x^2	10^7	0	10	7.693e-12	7.693e-12	5.162e-13

Table 4: Accuracy results for the given test case

5 Challenges Encountered in the process of optimization

- Initially I performed the basic optimization procedure of just using cython and annotating the cell with it, but it did not give great results
- Then I tried to cdef the function itself, but this meant I could not access the function from outside the cell, so wrappers were defined.
- Also I had to figure out a way to pass functions in C, this was done by defining a new **func_type** datatype.

6 References

1. Creating Latex Document

2. The Python3 Documentation

3. The Cython Documentation

4. Details on the implementation of function pointers in C