EE2703: Report

Assignment 6 - Speeding up with Cython

EE23B092

1 Implementation

1.1 Python implementation

The basic idea is to sum up all the areas of the trapeziums. The function accepts 4 parameters - the function to integrate (f), lower bound (a), upper bound (b) and the number of trapeziums (n).

1.2 Cython Implementation

It is similar in structure to the python implementation. It uses cdef for function and variable definitions along with C data types. Some Cython decorators are called to speed up the performance. The function f is made inline (as we call it multiple times) which makes it run faster. We also use a wrapper function which calls the actual function cy_trapz(f, a, b, n) to measure the time taken.

2 Performance and Accuracy on different functions (10 million trapeziums)

2.1 f(x) = x^2 from 0 to 1

2.1.1 Cython implementation

Time taken:- 14.9 ms ± 439 µs per loop — Result:- 0.3333333332358258

2.1.2 Numpy Implementation

Time taken:- 88.3 ms ± 1.61 ms per loop — Result:- 0.33333333333333504

2.1.3 Python Implementation

Time taken:- 1.99 s ± 18.9 ms per loop — Result:- 0.3333333332358258

2.2 f(x) = $sin(x)$ from 0 to π

2.2.1 Cython implementation

Time taken:- 490 ms ± 15 ms per loop — Result:- 2.000000000413601

2.2.2 Numpy Implementation

Time taken:- 94.3 ms ± 775 µs per loop — Result:- 1.999999999999982

2.2.3 Python Implementation

Time taken:- 1.94 s ± 73.3 ms per loop — Result:- 2.000000000413601

2.3 f(x) = e^x from 0 to 1

2.3.1 Cython implementation

Time taken:- 432 ms ± 9.02 ms per loop — Result:- 1.7182818283253474

2.3.2 Numpy Implementation

Time taken:- 83.2 ms ± 514 µs per loop — Result:- 1.718281828459046

2.3.3 Python Implementation

Time taken:- 1.79 s ± 50.1 ms per loop — Result:- 1.7182818283253474

2.4 f(x) = $1/x$ from 1 to 2

2.4.1 Cython Implementation

Time taken:- 14.9 ms ± 338 µs per loop — Result:- 0.6931471804472792

2.4.2 Numpy Implementation

Time taken:- 92.5 ms ± 2.94 ms per loop — Result:- 0.6931471805599465

2.4.3 Python Implementation

Time taken:- 1.44 s ± 87.5 ms per loop — Result:- 0.6931471804472792

3 Conclusion

Numpy is much more accurate than the other approaches and is faster in most cases. Cython seems to be the fastest for polynomial functions. To measure the time, I had to use a wrapper function for the Cython implementation otherwise timeit was throwing an error.