Speeding up with Cython

Madhav Soorya Tadepalli ee23b040

October 2024

Objective

Implement and optimize the trapezoidal rule integration method using Python and Cython, and compare it with NumPy's implementation.

Procedure

I started off with a basic implementation of the trapezoidal integration which gave me a base case for the time.
I then realized that the number of function calls could be halved by using an array as there were many repetitions involved. This resulted in the time to drop by nearly (but not exactly) half Further, I realized that we can implement the same without even an array, dropping my time further.

Building on this, I created the Cython implementation, specifying the type wherever possible.

Finally, I implemented the same using Numpy. Ensuring to take into account the time required to create the 'y' array.

Performance

The drop in time from 'py_trapz1' to 'py_trapz2' suggests that most of the time involved was due to the function call.
The drop in time form 'py_trapz2' to 'py_trapz3' is probably because of the decrease in overhead in creating the array.

Cython was better than Python implementation in terms of speed but worse in terms of accuracy (drop in accuracy could be due to the limit in precision of float).

Numpy performed significantly better than Python and Cython in terms of time with only slight decrease in accuracy compared to Python. The major reason for the speed up can be attributed to the fact that I used 'np.linspace' and performed only 1 function call to create the entire 'y' array.

Challenges

I did not use 'cdef' or any of the wrapper functions in the cython implementations as they were producing errors occasionally (with no change in code, it ran sometimes and failed to run other times), further the increase in speed was marginal (within error tolerance).