Trapezoidal Rule Integration: Python, Cython, and NumPy Implementation

Background

The trapezoidal rule is a numerical method used to approximate the definite integral of a function. Given a function $f(x)$ and an interval $[a, b]$, the rule approximates the area under the curve by dividing the interval into n subintervals and summing the areas of the resulting trapezoids.

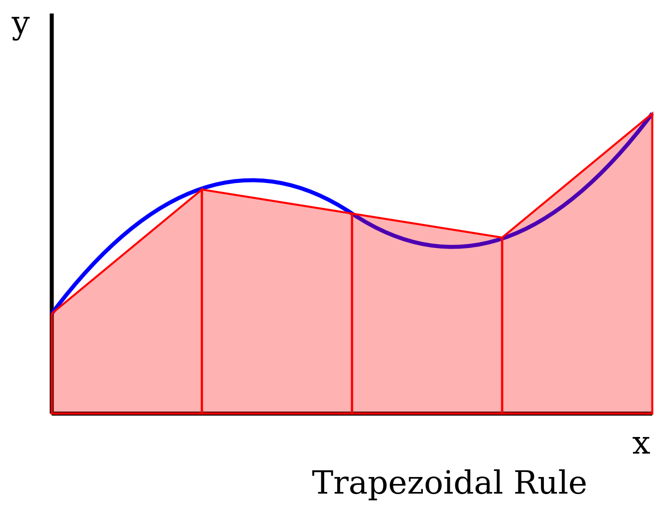

Figure 1: Illustration of Trapezoidal Rule

Objective

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

Tasks

1. Implement a pure Python function `py_trapz(f, a, b, n)` that calculates the definite integral of a function `f` from a to b using n trapezoids.
2. Create a Cython implementation `cy_trapz(f, a, b, n)` that performs the same calculation but utilizes Cython's static typing and C-level optimizations for improved performance.

3. Use the NumPy function `numpy.trapz()` as a reference implementation.
4. Compare the performance and accuracy of all three implementations (Python, Cython, and NumPy) for the following test cases:

    a. $f(x) = x^2$ from 0 to 1
    b. $f(x) = \sin(x)$ from 0 to π
    c. $f(x) = e^x$ from 0 to 1
    d. $f(x) = 1/x$ from 1 to 2

5. Conduct a performance test by integrating $f(x) = x^2$ from 0 to 10 with 10 million trapezoids. Compare the execution times of all three implementations.
6. Document your findings, including:

    - The implementation details of your Python and Cython functions
    - The accuracy of each method compared to the known analytical solutions
    - The performance improvements achieved by your Cython implementation
    - A comparison of your implementations against the NumPy version
    - Any challenges encountered during the Cython implementation and optimization process

Deliverables

- Source code for both Python and Cython implementations
- A brief report (maximum 2 pages) documenting your findings as outlined above

Combine the above into a single ZIP file named as per your roll number, and submit only this file. Your code should contain sufficient comments and be clear enough to understand what you have done, without being unnecessarily verbose. All instructions for running the programs should be self-contained, and should have been tested on the Jupyter server as they will be evaluated there. Make sure the Python and Cython versions you use are compatible with those on the server.

Evaluation Criteria

- Correctness and accuracy of implementations
- Performance improvement achieved with Cython
- Quality of code and documentation
- Depth of analysis in the report