Лабораторная работа 10
Сравнение методов оптимизации кода

Выполнил Жохов Даниил (502506)

В ходе работы создали 5 вариантов кода:

1) Без оптимизации
2) Оптимизация потоками
3) Оптимизация процессами
4) Оптимизация Cython
5) Оптимизация с помощью многопоточности/многопроцессности, Cython (nogil)

```
        Интегрирование cos(x) от 0 до 100000
Число итераций: 10,000,000
================================================================================
==========

 Однопоточные реализации:
Python (обычный):          0.7240 с, ошибка = 9.5e-01
Cython (с GIL):            0.6480 с, ошибка = 9.5e-01

================================================================================
==========
n_jobs    | Потоки (GIL)    | Потоки (nogil)    | Процессы       | Ошибка (потоки nogil)
--------------------------------------------------------------------------------
2         | 0.6561          | 0.0483            | 0.4260         | 9.5e-01
4         | 0.6214          | 0.0493            | 0.3333         | 9.5e-01
6         | 0.6343          | 0.0459            | 0.2657         | 9.5e-01
8         | 0.6427          | 0.0466            | 0.2565         | 9.5e-01

================================================================================
==========
 Многопроцессная версия с Cython (ProcessPool + Cython):
   2 процессов: 0.3952 с, ошибка = 9.5e-01
   4 процессов: 0.2676 с, ошибка = 9.5e-01
   6 процессов: 0.2613 с, ошибка = 9.5e-01
   8 процессов: 0.2657 с, ошибка = 9.5e-01

================================================================================
```

Выводы:
- Потоки с GIL не дают особого преимущества, в отличие от процессов
- Чем больше процессов, тем быстрее вычисления
- Однопоточный Cython ускоряет в ~1.1×
- Потоки с nogil дают реальный параллелизм (обходят GIL)
- Они сравнимы по скорости с ProcessPool, но без накладных расходов на процессы.

Самым быстрым вариантом оказался Cython с 6 потоками без GIL