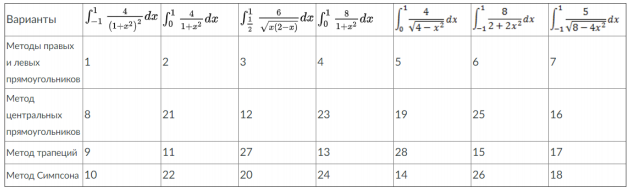
Петрушишин Александр

ИВТ-11М

Лабораторная задание №1

Вариант - 5



1. Код программы я прикрепил на GitHub.

По заданию необходимо решить интеграл, где нет первообразных элементарных функций. Решить его можно, следуя моему варианту, методом правых и левых прямоугольников (типо как дискретизация).

Далее нужно определить время просчётов, аккуратность и разбить это всё на количество интервалов (100, 1000, 10000, 100000, 1000000).

Моя функция (или как её видит компилятор):



Согласно калькулятору (будем честны), результат вышел: 2\*pi / 3

К сути, вывод последовательной программы без новорочек:

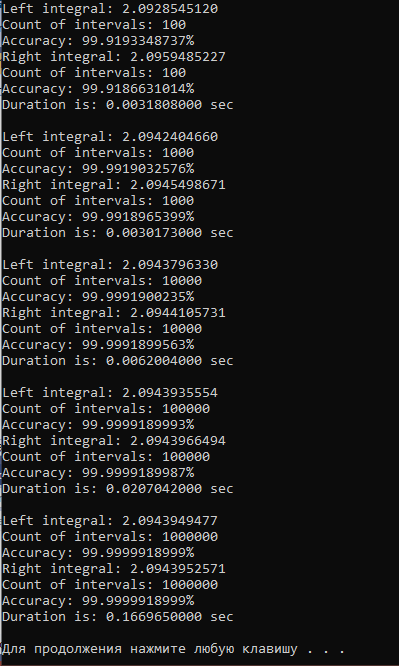


Рисунок 1 - Послед. подсчёт итераций

1. Теперь отключим векторизацию строкой  :

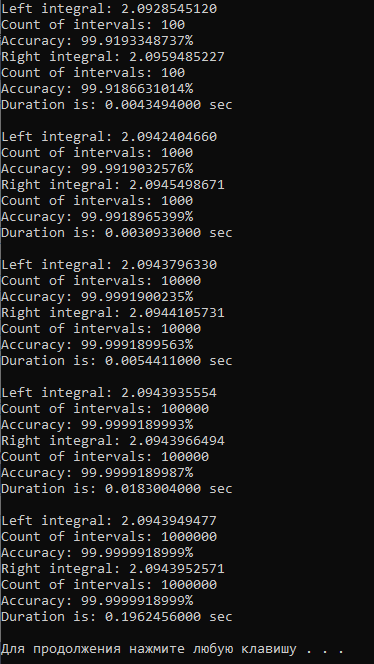


Рисунок 2 – Отключение векторизации

Как видно из скриншота, длительность обработки где-то стала выше, где-то ниже. Создаётся впечатление, что оно не работает…

С другой стороны, можно заметить, что точность не изменилась.

1. Теперь попробуем выполнить ключевое (если верить названию предмета) действие: распараллелить циклы так, чтобы потоки процессора умудрились просчитать вычисления быстрее за счёт разбиения их на кусочки и, одновременно «пережевывая».

Используем метод /Qpar (и по желанию /Qpar-report:2).

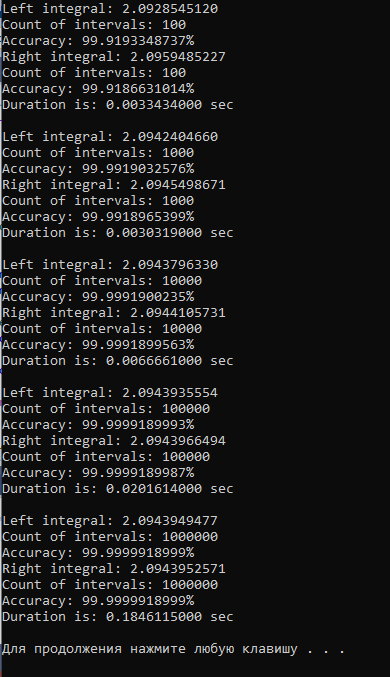


Рисунок 3 – Включаем супер-мега-режим распараллеливания

Вроде как стало даже быстрее. Исходя из того, что мой i5’ый имеет 2 ядра = 4 потока (скажем спасибо hyper threading), pragma loop я задал так: 

Но, откровенно говоря, не уверен, если оно сработало правильно, так как при каждом запуске программы, параметры меняются.

1. Немного запутался в разнице между распараллеливанием и разбиением на «треды». Но суть в том, что при подключении последних, скорость увеличилась в разы.

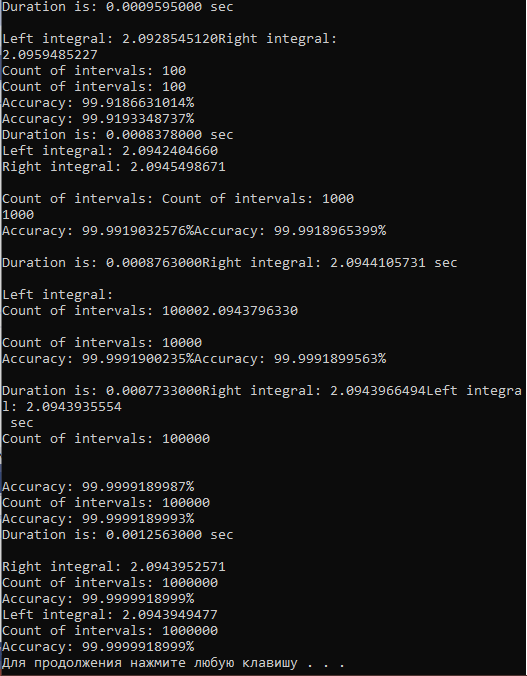


Рисунок 4 – Разбиение на треды

Вывод немного поплыл (строки имею в виду) в следствие одновременного расчёта на потоках, если правильно понимаю.

По итогу можно сказать, что распараллеливание – это здорово и полезно. Профит на глазах.

Сравнивая 1й и 4й скриншоты можно сказать, что производительность выросла в ~14 раз.