МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

САМАРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Лабораторная работа №2

Численное интегрирование

Цель работы: реализация алгоритма численного интегрирования функции двумя способами — однопоточным и многопоточным, проведение измерений времени выполнения, сравнение экспериментальных результатов с теоретическими предсказаниями по законам Амдала и Густафсона.

Задачи работы:

1. Реализовать численное интегрирование функции методом прямоугольников, трапеций, Монте-Карло.
2. Реализовать графический интерфейс на Windows Forms.
3. Реализовать две версии вычисления интеграла. Однопоточное и многопоточное.
4. Провести серию экспериментов, измеряя время выполнения при разном количестве потоков.
5. Рассчитать теоретическое ускорение по закону Амдала и закону Густафсона.
6. Сравнить экспериментальные результаты с теоретическими предсказаниями и сделать выводы.

Задание на лабораторную работу:

Написать программу, вычисление численного интеграла для однопоточного и многопоточного режима, вычислить ускорение по Амдалу и Густафсону.

Выбрать одну из функций для интегрирования из предложенных (по списку фамилий распределяются варианты):

$1 -\ f(x) = x^2 + 3x + 2$ - полиномиальная

$2 -\ f(x) = \sin(x) \cdot e^{-x/5}$ – осцилляторная

$3 -\ f(x) = \ln(1 + x)$ – логарифмическая

$4 -\ f(x) = e^{-x^2}$ - гауссова кривая

Разработать Windows Forms приложение с элементами управления (поля ввода для a, b, N и количества потоков, выпадающий список для выбора метода интегрирования, две кнопки: «Вычислить однопоточно» и «Вычислить многопоточно», поле вывода результата и времени выполнения, таблица для сравнения времени работы с разным числом потоков, кнопка «Очистить»).

Обязательно реализовать однопоточное вычисление (циклический проход по точкам) и многопоточное (с использованием Task, Thread). Провести тестирование с разными значениями потоков N=1, 2, 4, 8, 16. Записать время выполнения каждого варианта.

Рассчитать теоретическое ускорение по законам Амдала и Густафсона. Построить графики зависимости ускорения от числа потоков. Сравнить экспериментальные и теоретические результаты, сделать выводы.

В реализации вашей программы не использовать Thread.Sleep() для синхронизации потоков.

Требования к графическому интерфейсу:

1. Запрет ввода букв в числовые поля
2. Проверка на пустые поля

Содержание отчета:

1. Титульный лист
2. Экранные формы работы программы и листинг кода

3. Таблица с результатами исполнения.
4. График сравнения теоретического ускорения и практического.

Теоретические сведения:

Численное интегрирование – это метод приближенного вычисления определённого интеграла вида:

$$I = \int_a^b f)x)dx$$

Методы интегрирования

Метод прямоугольников - простейший метод, разбиение области на прямоугольники:

$$I \approx \sum_{i=0}^{N-1} f(x_i) \cdot \Delta x$$

Метод трапеций - учитывает среднее значение функции между соседними точками:

$$I \approx \sum_{i=0}^{N-1} \frac{f(x_i) + f(x_{i+1})}{2} \cdot \Delta x$$

Метод Монте-Карло - использует случайные точки для оценки интеграла:

$$I \approx \frac{(b-a)}{N} \sum_{i=1}^{N} f(x_i)$$

где x_i — случайные точки в диапазоне [a, b]

Законы Амдала и Густафсона рассматривали на лекции, формулы возьмите из лекционного материала.

Численное интегрирование можно реализовать с помощью обычных циклов Операторы итерации —for, foreach, do и while - C# reference | Microsoft Learn

Thread — это базовый способ работы с потоками. Он предоставляет полный контроль над созданием и управлением потоками, но требует явного управления их жизненным циклом.

Task — более высокоуровневый API для асинхронного программирования. Управление потоками происходит автоматически, и задачи могут выполняться в ThreadPool.

Thread Класс (System.Threading) | Microsoft Learn.

ThreadPool Класс (System.Threading) | Microsoft Learn.

Task.Run Метод (System.Threading.Tasks) | Microsoft Learn - Позволяет выполнять операции в отдельных потоках без явного управления потоками.

Для каждой задачи создается отдельный Task, выполняющий вычисления. Task.Run запускает поток, который выполняет какое-то действие. Task.WaitAll гарантирует, что выполнение продолжится только после завершения всех задач.

Task.WaitAll Метод (System.Threading.Tasks) | Microsoft Learn - Позволяет дождаться завершения всех задач перед продолжением выполнения кода.

Чтобы улучшить Ваш код, можно использовать механизм lock, он блокирует доступ к ресурсу, пока один из потоков его использует.

Инструкция блокировки — синхронизация доступа к общим ресурсам - C# reference | Microsoft Learn

Вместо List<T> можно использовать ConcurrentBag<T>, ConcurrentQueue<T>, ConcurrentDictionary<TKey, TValue>, которые специально разработаны для многопоточной работы.

ConcurrentBag<T> Класс (System.Collections.Concurrent) | Microsoft Learn

ConcurrentDictionary<TKey,TValue> Класс (System.Collections.Concurrent) | Microsoft Learn

ConcurrentQueue<T> Класс (System.Collections.Concurrent) | Microsoft Learn

Они не требуют использования явного lock.

Замер времени рекомендуется сделать с помощью Stopwatch Класс (System.Diagnostics) | Microsoft Learn

Для ввода матриц использовать DataGridView.