

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ
ОТЧЕТ О ПРАКТИКЕ

студента 3 курса 311 группы
направления 02.03.02 — Фундаментальная Информатика и Информационные
Технологии
факультета КНиИТ
Вильцева Данила Денисовича

Проверил
Старший преподаватель

М. С. Портенко

СОДЕРЖАНИЕ

1	Work 01	3
1.1	Условие задачи	3
1.2	Решение	3
1.3	Сравнение времени работы	4
2	Work 02	5
2.1	Условие задачи	5
2.2	Решение	5
2.3	Сравнение времени работы	6
3	Work 03	7
3.1	Условие задачи	7
3.2	Решение	7
3.3	Сравнение времени работы	7

1 Work 01

1.1 Условие задачи

Реализуйте параллельные алгоритмы, использующие метод прямоугольников и формулу Симпсона для подсчета интегралов. Точные значения интегралов указаны для проверки численных вычислений. В случае, если в верхнем пределе интегрирования указан знак бесконечности, то в расчете необходимо заменить его на 10^6 .

Сравните время численного интегрирования для последовательной и параллельной реализации. Какое ускорение выполнения программы предоставляет переход к многопоточной версии?

Вариант 1.

$$\int_0^1 \frac{dx}{\sqrt{1-x^2}} = \frac{\pi}{2}$$

1.2 Решение

Параллельные вычисление были применены к циклу for функции integral(): согласно схеме разбиения данных при численном интегрировании, цикл можно распараллеливать.

```
#pragma omp parallel for private(x) reduction(+: sum)
for (i = 0; i < n; i++)
{
    x = a + i * h + h / 2.0;
    sum += func(x) * h;
}
```

Рисунок 1 – Метод прямоугольников

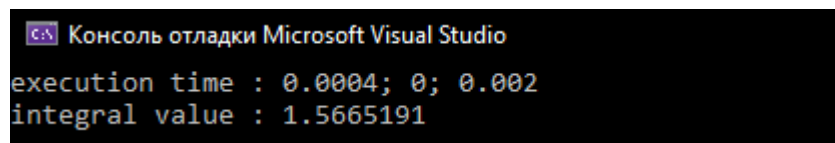
```
#pragma omp parallel for private(k) reduction(+: integration)
for (i = 1; i <= n - 1; i++)
{
    k = a + i * h;

    if (i % 2 == 0)
    {
        integration = integration + 2.0 * (func(k));
    }
    else
    {
        integration = integration + 4.0 * (func(k));
    }
}
```

Рисунок 2 – Метод Симпсона

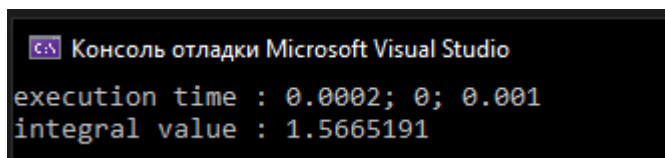
1.3 Сравнение времени работы

Ускорение: 2



```
Консоль отладки Microsoft Visual Studio
execution time : 0.0004; 0; 0.002
integral value : 1.5665191
```

Рисунок 3 – Последовательная версия



```
Консоль отладки Microsoft Visual Studio
execution time : 0.0002; 0; 0.001
integral value : 1.5665191
```

Рисунок 4 – Параллельная версия

2 Work 02

2.1 Условие задачи

Модифицируйте разработанную ранее программу по методу прямоугольников для численного интегрирования тестовой функции:

$$\int_0^{16} \int_0^{16} \frac{e^{\sin \pi x \cos \pi y} + 1}{(b_1 - a_1)(b_2 - a_2)} dx dy \approx 2.130997$$

Предложите несколько способов распараллеливания базового алгоритма метода прямоугольников для двумерной функции с использованием директивы `#pragma omp parallel for`. Реализуйте предложенные способы на примере тестовой функции.

Сравните время численного интегрирования для последовательной и параллельных реализаций и параллельных реализаций между собой.

2.2 Решение

Параллельные вычисление были применены к циклу `for` функции `integral()`: согласно схеме разбиения данных при численном интегрировании, цикл можно распараллеливать.

Параллельная версия №1:

```
double func(double x, double y, const double a1,
const double b1, const double a2, const double b2) {
    return ((exp(sin(PI * x) * cos(PI * y)) + 1) /
((b1 - a1) * (b2 - a2)));
}

#pragma omp parallel for private(x) reduction(+: sum)
for (i = 0; i < N; i++) {
    x = a1 + i * h1 + h1 / 2;
    #pragma omp parallel for private(y) reduction(+: sum)
    for (j = 0; j < M; j++) {
        y = a2 + j * h2 + h2 / 2;
        sum += h1 * h2 * func(x, y, a1, b1, a2, b2);
    }
}
```

Параллельная версия №2:

```
for (i = 0; i < N; i++) {
    #pragma omp parallel for private(x, y) reduction(+: sum)
    for (j = 0; j < M; j++) {
        x = a1 + j * h1 + h1 / 2;
```

```

        y = a2 + j * h2 + h2 / 2;
        sum += h1 * h2 * func(x, y, a1, b1, a2, b2);
    }
}

```

2.3 Сравнение времени работы

Ускорение версии №1: 6.199

Ускорение версии №2: 1.256

```

execution time : 0.8227; 0.131; 7.016
integral value : 2.1309969

```

Рисунок 5 – Последовательная версия

```

execution time : 0.1327; 0.13; 0.135
integral value : 2.1309969

```

Рисунок 6 – Параллельная версия №1

```

execution time : 0.6548; 0.189; 4.745
integral value : 2.0634834

```

Рисунок 7 – Параллельная версия №2

3 Work 03

3.1 Условие задачи

В качестве методов приближенного вычисления двойных интегралов рассмотрим параллельные реализации метода:

2. трапеций

3.2 Решение

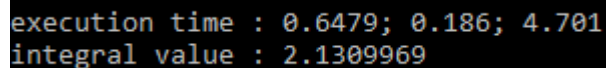
Параллельные вычисления были применены к циклу for функции trapezIntegral(): согласно схеме разбиения данных при численном интегрировании, цикл можно распараллеливать.

```
#pragma omp parallel for private(x, y, w) reduction(+: sum)
for (int i = 0; i <= N; i++) {
    for (int j = 0; j <= M; j++) {
        x = a1 + i * h1;
        if (i > 0 && i < N && j > 0 && j < M) {
            w = 1;
        }
        else {
            if ((i == 0 || i == N) && (j == 0 || j == M))
                w = 0.25;
            else
                w = 0.5;
        }

        y = a2 + j * h2;
        sum += w * (exp(sin(PI * x)) * cos(PI * y)) + 1);
    }
}
```

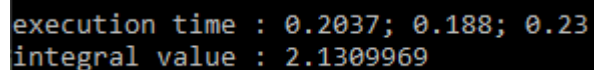
3.3 Сравнение времени работы

Ускорение: 3.18



```
execution time : 0.6479; 0.186; 4.701
integral value : 2.1309969
```

Рисунок 8 – Последовательная версия



```
execution time : 0.2037; 0.188; 0.23
integral value : 2.1309969
```

Рисунок 9 – Параллельная версия