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

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ, ФОРМУЛА СИМПСОНА, ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ И КРАТНЫХ ИНТЕГРАЛОВ

ОТЧЕТ О ПРАКТИКЕ

Студента 3 курса 311 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Аношкина Андрея Алексеевича

Проверил
Старший преподаватель _____ М. С. Портенко

Саратов 2024

СОДЕРЖАНИЕ

1 Work 02

Задание

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

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

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.

Численное интегрирование по методу прямоугольников для двумерной функции

$$\begin{cases} J = \int_{a_1}^{b_1}\int_{a_2}^{b_2} f(x,y)\,dx\,dy \approx \sum_{i=0}^{N-1}\sum_{j=0}^{M-1}(h_1\cdot h_2\cdot f(x_i,y_i)) \\ x_i = a_1 + i\cdot h_1 + \frac{h_1}{2} \\ y_i = a_2 + i\cdot h_2 + \frac{h_2}{2} \end{cases} \tag{2}$$

где N — количество отрезков интегрирования по оси x, M — количество отрезков интегрирования по оси y, $h_1 = (b_1 - a_1)/N$ и $h_2 = (b_2 - a_2)/N$.

Реализация

Фрагмент кода решения приведен ниже:

```
#include <iostream>
#include <omp.h>
#include <time.h>
#include <cmath>
#define PI 3.14159265358979323846264338327950

using namespace std;

double f1(const double x, const double y, const double a1, const double b1, const double a2, const double b2) {
	return (1 + exp(sin(PI * x) * cos(PI * y))) / (b1 - a1) / (b2 - a2);
}

```

```
13 void integral_posl(const double a1, const double b1, const double a2, const double b2, const double h1, const
   ↪ double h2, double* res) {
14         double sum = 0;
15         int n = (int)((b1 - a1) / h1);
16         int m = (int)((b2 - a2) / h2);
17 
18         for (int i = 0; i < n; ++i) {
19                 double x = a1 + i * h1 + h1 / 2;
20                 for (int j = 0; j < m; ++j) {
21                         double y = a2 + j * h2 + h2 / 2;
22                         sum += f1(x, y, a1, b1, a2, b2) * h1 * h2;
23                 }
24         }
25 
26         *res = sum;
27 }
28 
29 void integral_paral_1(const double a1, const double b1, const double a2, const double b2, const double h1,
   ↪ const double h2, double* res) {
30         double sum = 0;
31         int n = (int)((b1 - a1) / h1);
32         int m = (int)((b2 - a2) / h2);
33 
34         #pragma omp parallel for reduction(+: sum)
35         for (int i = 0; i < n; ++i) {
36                 double x = a1 + i * h1 + h1 / 2;
37                 for (int j = 0; j < m; ++j) {
38                         double y = a2 + j * h2 + h2 / 2;
39                         sum += f1(x, y, a1, b1, a2, b2) * h1 * h2;
40                 }
41         }
42 
43         *res = sum;
44 }
45 
46 void integral_paral_2(const double a1, const double b1, const double a2, const double b2, const double h1,
   ↪ const double h2, double* res) {
47         double sum = 0;
48         int n = (int)((b1 - a1) / h1);
49         int m = (int)((b2 - a2) / h2);
50 
51         #pragma omp parallel for reduction(+: sum)
52         for (int i = 0; i < n; ++i) {
53                 double x = a1 + i * h1 + h1 / 2;
54                 #pragma omp parallel for reduction(+: sum)
55                 for (int j = 0; j < m; ++j) {
56                         double y = a2 + j * h2 + h2 / 2;
57                         sum += f1(x, y, a1, b1, a2, b2) * h1 * h2;
58                 }
59         }
```

```
60
61          *res = sum;
62  }
63
64  double experiment(double* res, void f(const double a1, const double b1, const double a2, const double b2,
    ↪   const double h1, const double h2, double* res)) {
65          double stime = 0, ftime = 0;
66          double a1 = 0, b1 = 16;
67          double a2 = 0, b2 = 16;
68          double h1 = 0.005, h2 = 0.005;
69          stime = clock();
70          f(a1, b1, a2, b2, h1, h2, res);
71          ftime = clock();
72
73          return (ftime - stime) / CLOCKS_PER_SEC;
74  }
75
76  void calculate(void f(const double a1, const double b1, const double a2, const double b2, const double h1,
    ↪   const double h2, double* res)) {
77          double avg_time = 0;
78          double min_time = 0;
79          double max_time = 0;
80          double res = 0;
81          int numbExp = 10;
82
83          min_time = max_time = experiment(&res, f);
84          avg_time = min_time / numbExp;
85          for (int i = 0; i < numbExp - 1; ++i) {
86                  double time = experiment(&res, f);
87                  if (time > max_time)
88                          max_time = time;
89                  if (time < min_time)
90                          min_time = time;
91
92                  avg_time += time / numbExp;
93          }
94
95          cout << "Execution time: " << avg_time << "; " << min_time << "; " << max_time << "\n";
96          cout.precision(8);
97          cout << "Integral value: " << res << "\n";
98  }
99
100 int main() {
101
102         bool calcPosl = true, calcParal_1 = true, calcParal_2 = true;
103
104         // Последовательное вычисление
105
106         if (calcPosl) {
107                 cout << "Posl:\n";
```

```
108             calculate(integral_posl);
109             cout << "\n";
110         }
111
112         // Параллельное вычисление (1)
113
114         if (calcParal_1) {
115             cout << "Paral_1:\n";
116             calculate(integral_paral_1);
117             cout << "\n";
118         }
119
120         // Параллельное вычисление (2)
121
122         if (calcParal_2) {
123             cout << "Paral_2:\n";
124             calculate(integral_paral_2);
125             cout << "\n";
126         }
127
128         system("pause");
129         return 0;
130 }
```

Результаты работы

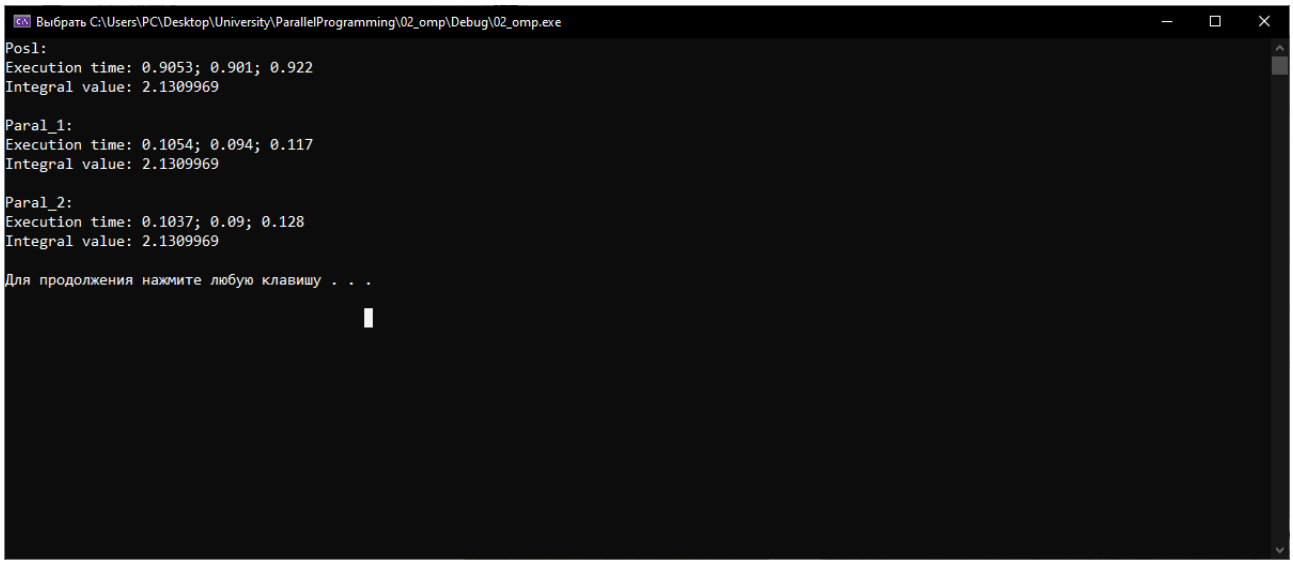

Рисунок 1 – Work-2

Значение ускорения выполнения программы при переходе к многопоточной версии:

$$a = \frac{timeSeq}{timePar} = \frac{0.9053}{0.1054} \approx 8.5892$$

Значение ускорения выполнения программы при переходе к многопоточной версии (2):

$$a = \frac{timeSeq}{timePar} = \frac{0.9053}{0.1037} \approx 8.73$$

Характеристики устройства

Процессор: Intel(R) Core(TM) i5-10400F

Ядер: 6

Оперативная память: 16 Гб