МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра математической кибернетики и компьютерных наук

**ПАРАЛЛЕЛЬНОЕ И РАСПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ. OPENMP. ВЫЧИСЛЕНИЕ ДВОЙНЫХ ИНТЕГРАЛОВ. МЕТОДЫ ЯЧЕЕК, ТРАПЕЦЙ, СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ.**

ЛАБОРАТОРНАЯ РАБОТА №3

студента 4 курса 451 группы

направления 09.03.04 — Программная инженерия

факультета КНиИТ

Голикова Артема Олеговича

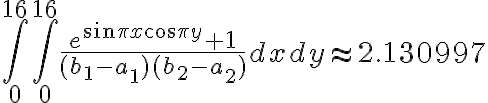
Проверил

канд. физ-мат. н., доцент \_\_\_\_\_\_\_\_\_\_\_\_ К. П. Савина

Саратов 2020

**Цель работы:** изучить использование потоков OpenMP (директивы #pragma omp paralel) и операции редукции, приводящих к уменьшению вычисления двойных интегралов на примере методов ячеек, трапеций, статистических испытаний; сравнить время выполнения алгоритмов численного интегрирования с использованием разных режимов.

**Ход работы:**



Код программы реализующий метод ячеек для последовательной реализации:

// Work03, метод ячеек, последовательная версия

#include <time.h>

#include <iostream>

#define PI 3.1415926535897932384626433832795

using namespace std;

double f(double x, double y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / (16.0\*16.0);

}

void integral(double a1, double b1, double a2, double b2, double h, double\* res)

{

double sum = 0.0;

int n = (b2 - a2) / h;

int m = (b1 - a1) / h;

double dx = (b1 - a1) / m;

double dy = (b2 - a2) / n;

double x0, y0, x1, y1;

for (int i = 1, x0 = a1; i < m + 1; i++)

{

x1 = x0 + dx;

for (int j = 1, y0 = a2; j < n + 1; j++)

{

y1 = y0 + dy;

sum += f((x0 + x1) / 2.0, (y0 + y1) / 2.0);

y0 = y1;

}

x0 = x1;

}

sum \*= dx \* dy;

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 1; // количество запусков программы

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

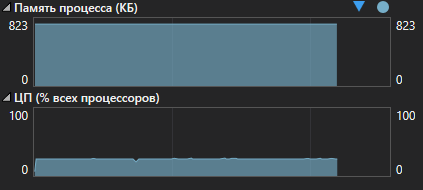
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

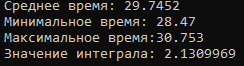
return 0;

}

Использование ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод ячеек для параллельной реализации:

// Work03, метод ячеек, параллельная версия

#include <time.h>

#include <iostream>

#include <omp.h>

#define PI 3.1415926535897932384626433832795

using namespace std;

double f(double x, double y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / (16.0 \* 16.0);

}

void integral(double a1, double b1, double a2, double b2, double h, double\* res)

{

double sum = 0.0;

int n = int((b2 - a2) / h);

int m = int((b1 - a1) / h);

double dx = (b1 - a1) / m;

double dy = (b2 - a2) / n;

double x0 = a1, y0 = a2, x1, y1;

#pragma omp parallel for private(x1, y1) reduction(+: sum)

for (int i = 1; i < m + 1; i++)

{

x1 = x0 + dx;

for (int j = 1; j < n + 1; j++)

{

y1 = y0 + dy;

sum += f((x0 + x1) / 2.0, (y0 + y1) / 2.0);

y0 = y1;

}

x0 = x1;

}

sum \*= dx \* dy;

\*res = sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 10; // количество запусков программы

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

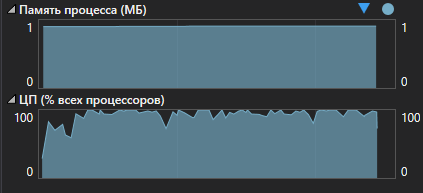
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

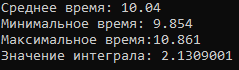
return 0;

}

Использование ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод трапеций для последовательной реализации:

// Work03, метод трапеций, последовательная версия

#include <time.h>

#include <iostream>

#define PI 3.1415926535897932384626433832795

using namespace std;

double f(double x, double y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / (16.0 \* 16.0);

}

void integral(double a1, double b1, double a2, double b2, double h, double\* res)

{

double sum = 0.0;

int n = int((b2 - a2) / h);

int m = int((b1 - a1) / h);

double x = a1, y = a2;

double q = 0;

for (int i = 0; i <= n; i++)

{

x = a1 + i \* h;

for (int j = 0; j <= m; j++)

{

if (i > 0 && i < n && j > 0 && j < m) q = 1;

else if ((i == 0 || i == m) && (j == 0 || j == n)) q = 0.25;

else q = 0.5;

y = a2 + j \* h;

sum += q \* f(x, y);

}

}

\*res = h \* h \* sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 10; // количество запусков программы

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

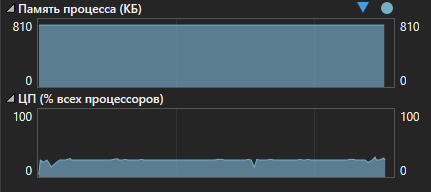
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

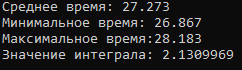
return 0;

}

Использования ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод трапеций для параллельной реализации:

// Work03, метод трапеций, параллельная версия

#include <time.h>

#include <iostream>

#include <omp.h>

#define PI 3.1415926535897932384626433832795

using namespace std;

double f(double x, double y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / (16.0 \* 16.0);

}

void integral(double a1, double b1, double a2, double b2, double h, double\* res)

{

double sum = 0.0;

int n = int((b2 - a2) / h);

int m = int((b1 - a1) / h);

double x = a1, y = a2;

double q = 0;

#pragma omp parallel for private(x, y) reduction(+: sum)

for (int i = 0; i <= n; i++)

{

x = a1 + i \* h;

for (int j = 0; j <= m; j++)

{

if (i > 0 && i < n && j > 0 && j < m) q = 1;

else if ((i == 0 || i == m) && (j == 0 || j == n)) q = 0.25;

else q = 0.5;

y = a2 + j \* h;

sum += q \* f(x, y);

}

}

\*res = h \* h \* sum;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 10; // количество запусков программы

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

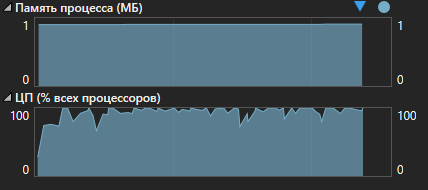
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

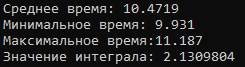
return 0;

}

Использования ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод статистический испытаний для последовательной реализации:

// Work03, метод статистических испытаний (Монте-Карло), последовательная версия

#include <time.h>

#include <iostream>

#define PI 3.1415926535897932384626433832795

using namespace std;

double f(double x, double y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / (16.0 \* 16.0);

}

double random(double min, double max)

{

return (double)(rand()) / RAND\_MAX \* (max - min) + min;

}

void integral(double a1, double b1, double a2, double b2, double h, double\* res)

{

int n = 0;

int N = 10000000;

double sum = 0.0;

for (int i = 0; i < N-1; i++)

{

double x = random(a1,b1);

double y = random(a1, b1);

if ((x >= 0 && x <= 16.0) && (y >= 0 && y <= 16.0))

{

n++;

sum += f(x, y);

}

}

\*res = b1 \* b2 \* sum / n;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 10; // количество запусков программы

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

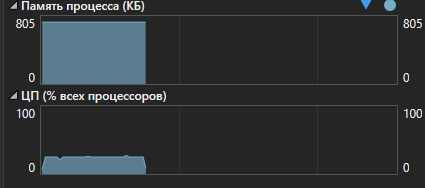
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

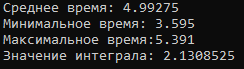
return 0;

}

Использования ЦП во время выполнения:



Результат выполнения:



Код программы реализующий метод статистический испытаний для параллельной реализации:

// Work03, метод статистических испытаний (Монте-Карло), параллельная версия

#include <time.h>

#include <iostream>

#include <omp.h>

#define PI 3.1415926535897932384626433832795

using namespace std;

double f(double x, double y)

{

return (exp(sin(PI \* x) \* cos(PI \* y)) + 1) / (16.0 \* 16.0);

}

double random(double min, double max)

{

return (double)(rand()) / RAND\_MAX \* (max - min) + min;

}

void integral(double a1, double b1, double a2, double b2, double h, double\* res)

{

int n = 0;

int N = 10000000;

double sum = 0.0;

double x, y;

#pragma omp parallel for private(x,y) reduction (+: sum, n)

for (int i = 0; i < N - 1; i++)

{

x = random(a1, b1);

y = random(a1, b1);

if ((x >= 0 && x <= 16.0) && (y >= 0 && y <= 16.0))

{

n++;

sum += f(x, y);

}

}

\*res = b1 \* b2 \* sum / n;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a1 = 0.0, a2 = 0.0; // левая граница интегрирования

double b1 = 16.0, b2 = 16.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

stime = clock();

integral(a1, b1, a2, b2, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

setlocale(LC\_ALL, "Russian");

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 10; // количество запусков программы

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

if (max\_time < time) max\_time = time;

if (min\_time > time) min\_time = time;

}

// вывод результатов эксперимента

cout << " Среднее время: " << avg\_time / numbExp << "\n Минимальное время: " <<

min\_time << "\n Максимальное время:" << max\_time << endl;

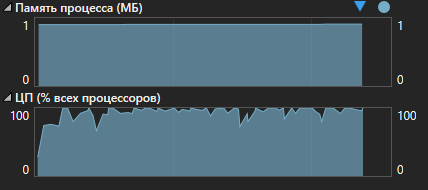
cout.precision(8);

cout << " Значение интеграла: " << res << endl;

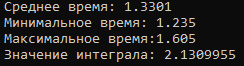
return 0;

}

Использования ЦП во время выполнения:



Результат выполнения:



Сравнение результатов:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Последовательная  реализация | Параллельная  реализация | Ускорение |
| Метод ячеек | Мин. время: 28,47  Макс. время: 30,753  Сред. время: 29,7452 | Мин. время: 9,854  Макс. время: 10,861  Сред. время: 10,04 | 2,9627 |
| Метод трапеций | Мин. время: 26,867  Макс. время: 28,183  Сред. время: 27,273 | Мин. время: 9,931  Макс. время: 11,187  Сред. время: 10,4719 | 2,6044 |
| Метод Монте-Карло | Мин. время: 3,595  Макс. время: 5,391  Сред. время: 4,99275 | Мин. время: 1,235  Макс. время: 1,605  Сред. время: 1,3301 | 3,7537 |

Вычисления происходили на процессоре Intel Core i5-2310 (4 ядра, 4 потока). По результатам сравнения видно, что распараллеливание при вычислении двойных интегралов функции дает прирост по быстродействию в 2,5 – 3,7 раза, в зависимости от метода вычисления.

**Вывод:** изучил использование потоков OpenMP (директивы #pragma omp paralel) и операции редукции, приводящих к уменьшению вычисления двойных интегралов на примере методов ячеек, трапеций, статистических испытаний; сравнил время выполнения алгоритмов численного интегрирования с использованием разных методов.