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. MPI. ВЫЧИСЛЕНИЕ ДВОЙНЫХ ИНТЕГРАЛОВ. МЕТОДЫ ЯЧЕЕК, ТРАПЕЦИЙ, СТАТИСТИЧЕСКИХ ИСПЫТАНИЙ.**

ЛАБОРАТОРНАЯ РАБОТА №6

студента 4 курса 451 группы

направления 09.03.04 — Программная инженерия

факультета КНиИТ

Голикова Артема Олеговича

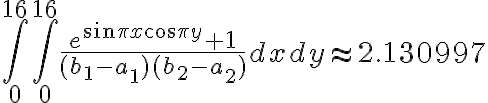
Проверил

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

Саратов 2020

**Цель работы:** изучить использование потоков MPI 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:

// Work12, метод ячеек

#include <time.h>

#include <iostream>

#include <mpi.h>

#define PI 3.1415926535897932384626433832795

int NProc, ProcId;

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;

double x0, y0, x1, y1;

for (x1 = a1 + h; x1 < b1; x1 += h)

{

x0 = x1 - h;

for (y1 = a2 + (ProcId + 1) \* h; y1 < b2; y1 += NProc \* h)

{

y0 = y1 - h;

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

}

}

sum \*= h \* h;

MPI\_Reduce(&sum, res, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

}

double experiment(double\* res)

{

double stime; // 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

if (ProcId == 0)

{

stime = MPI\_Wtime();

}

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

if (ProcId == 0)

{

return MPI\_Wtime() - stime;

}

}

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

MPI\_Init(NULL, NULL);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &NProc);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcId);

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

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

{

time = experiment(&res);

if (ProcId == 0)

{

avg\_time += time;

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

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

}

}

// вывод результатов эксперимента

if (ProcId == 0)

{

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;

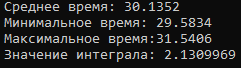
}

MPI\_Finalize();

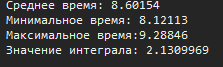
return 0;

}

Результат выполнения в однопоточном режиме:



Результат выполнения в многопоточном режиме:



Код программы реализующий метод трапеций:

// Work12, метод трапеций

#include <time.h>

#include <iostream>

#include <mpi.h>

#define PI 3.1415926535897932384626433832795

int NProc, ProcId;

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 = ProcId; j <= m; j+=NProc)

{

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);

}

}

MPI\_Reduce(&sum, res, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

\*res \*= h \* h;

}

double experiment(double\* res)

{

double stime; // 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

if (ProcId == 0)

{

stime = MPI\_Wtime();

}

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

if (ProcId == 0)

{

return MPI\_Wtime() - stime;

}

}

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

MPI\_Init(NULL, NULL);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &NProc);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcId);

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

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

{

time = experiment(&res);

if (ProcId == 0)

{

avg\_time += time;

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

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

}

}

// вывод результатов эксперимента

if (ProcId == 0)

{

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;

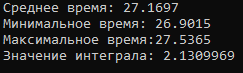
}

MPI\_Finalize();

return 0;

}

Результат выполнения в однопоточном режиме:



Результат выполнения в многопоточном режиме:



Код программы реализующий метод статистический испытаний:

// Work12, метод Монте-Карло

#include <time.h>

#include <iostream>

#include <mpi.h>

#define PI 3.1415926535897932384626433832795

int NProc, ProcId;

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;

}

int N = 100000000;

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

{

int n = 0;

double sum = 0.0;

for (int i = ProcId; i < N; i += NProc)

{

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);

}

}

MPI\_Reduce(&sum, res, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

int nn = 0;

MPI\_Reduce(&n, &nn, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

\*res \*= b1 \* b2 / nn;

}

double experiment(double\* res)

{

double stime; // 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

if (ProcId == 0)

{

stime = MPI\_Wtime();

}

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

if (ProcId == 0)

{

return MPI\_Wtime() - stime;

}

}

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

MPI\_Init(NULL, NULL);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &NProc);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &ProcId);

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

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

{

time = experiment(&res);

if (ProcId == 0)

{

avg\_time += time;

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

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

}

}

// вывод результатов эксперимента

if (ProcId == 0)

{

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;

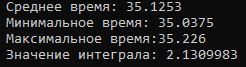
}

MPI\_Finalize();

return 0;

}

Результат выполнения в однопоточном режиме:



Результат выполнения в многопоточном режиме:



Сравнение результатов:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Последовательная  реализация | Параллельная  реализация | Ускорение |
| Метод ячеек | Мин. время: 29,5834  Макс. время: 31,5406  Сред. время: 30,1352 | Мин. время: 8,12113  Макс. время: 9,28846  Сред. время: 8,60154 | 3,5035 |
| Метод трапеций | Мин. время: 26,9015  Макс. время: 27,5365  Сред. время: 27,1697 | Мин. время: 7,04057  Макс. время: 7,59895  Сред. время: 7,19886 | 3,7741 |
| Метод Монте-Карло | Мин. время: 35,0375  Макс. время: 35,226  Сред. время: 35,1253 | Мин. время: 9,38058  Макс. время: 10,2864  Сред. время: 9,76475 | 3,5972 |

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 3,5 – 3,7 раза, в зависимости от метода вычисления.

**Вывод:** изучил использование потоков MPI приводящих к уменьшению вычисления двойных интегралов на примере методов ячеек, трапеций, статистических испытаний; сравнил время выполнения алгоритмов численного интегрирования с использованием разных методов.