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. ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ, ФОРМУЛА СИМПСОНА, ВЫЧИСЛЕНИЕ ОПРЕДЕЛЕННЫХ И КРАТНЫХ ИНТЕГРАЛОВ.**

ЛАБОРАТОРНАЯ РАБОТА №10

(Вариант №4)

студента 4 курса 451 группы

направления 09.03.04 — Программная инженерия

факультета КНиИТ

Голикова Артема Олеговича

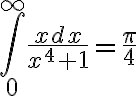
Проверил

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

Саратов 2020

**Цель работы:** изучить использование потоков MPI, приводящих к уменьшению вычисления интегралов, на примере методов прямоугольников и Симпсона; сравнить время выполнения алгоритмов численного интегрирования с использованием разных режимов.

**Ход работы:**



Код программы реализующий метод прямоугольников:

// Work10 - Вариант 4, метод прямоугольников

#include <time.h>

#include <iostream>

#include <mpi.h>

using namespace std;

int NProc = 0, ProcId = 0;

double f(double x)

{

return (x / (x \* x \* x \* x + 1));

}

void integral(const double a, const double b,

const double h, double\* res)

{

int n;

double sum = 0.0;

double x;

n = (int)((b - a) / h);

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

{

x = a + i \* h + h / 2.0;

sum += f(x) \* 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 a = 0.0; // левая граница интегрирования

double b = 1000000.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

if (ProcId == 0)

{

stime = MPI\_Wtime();

}

integral(a, b, h, res); // вызов функции интегрирования

if (ProcId == 0)

{

return (MPI\_Wtime() - stime);

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

MPI\_Init(NULL, NULL);

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

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

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

double max\_time; // максимальное время работы

double avg\_time; // среднее время работы

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);

if (ProcId == 0)

{

avg\_time += time;

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

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

}

}

if (ProcId == 0)

{

// 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;

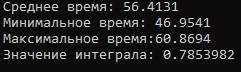
}

MPI\_Finalize();

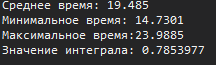
return 0;

}

Результат выполнения программы в однопоточном режиме:



Результат выполнения программы в многопоточном режиме:



Код программы реализующий метод Симпсона:

// Work10 - Вариант 4, метод Симпсона

#include <time.h>

#include <iostream>

#include <mpi.h>

using namespace std;

int NProc = 0, ProcId = 0;

double f(double x)

{

return (x / (x \* x \* x \* x + 1));

}

void integral(const double a, const double b,

const double h, double\* res)

{

int n;

double sum1 = 0.0, sum2 = 0.0, s1 = 0.0, s2 = 0.0;

double x;

n = (int)((b - a) / (2 \* h));

for (int i = 1; i <= n; i += NProc)

{

x = (a + (2 \* i - 1) \* h);

sum1 += 4 \* f(x);

}

MPI\_Reduce(&sum1, &s1, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

for (int i = 1; i < n; i += NProc)

{

x = a + 2 \* i \* h;

sum2 += 2.0 \* f(x);

}

MPI\_Reduce(&sum2, &s2, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (ProcId == 0)

{

\*res = (h / 3) \* (s1 + s2 + ((a / (a \* a \* a \* a + 1)) + (b / (b \* b \* b \* b + 1))));

}

}

double experiment(double\* res)

{

double stime; // время начала и конца расчета

double a = 0.0; // левая граница интегрирования

double b = 1000000.0; // правая граница интегрирования

double h = 0.001; // шаг интегрирования

if (ProcId == 0)

{

stime = MPI\_Wtime();

}

integral(a, b, h, res); // вызов функции интегрирования

if (ProcId == 0)

{

return (MPI\_Wtime() - stime);

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

MPI\_Init(NULL, NULL);

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

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

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

double max\_time; // максимальное время работы

double avg\_time; // среднее время работы

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);

if (ProcId == 0)

{

avg\_time += time;

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

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

}

}

if (ProcId == 0)

{

// 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;

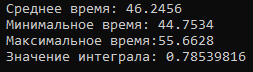
}

MPI\_Finalize();

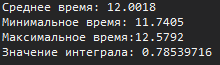
return 0;

}

Результат выполнения в однопоточном режиме:



Результат выполнения в многопоточном режиме:



Сравнение результатов:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Последовательная  реализация | Параллельная  реализация | Ускорение |
| Метод прямоугольников | Мин. время: 46,4131  Макс. время: 60,8694  Сред. время: 56,4131 | Мин. время: 14,7301  Макс. время: 23,9885  Сред. время: 19,485 | 2,8952 |
| Метод Симпсона | Мин. время: 44,7534  Макс. время: 55,6628  Сред. время: 46,2456 | Мин. время: 11,7405  Макс. время: 12,5792  Сред. время: 12,0018 | 3,8532 |

Вычисления происходили на процессоре Intel Core i5-2310 (4 ядра, 4 потока). По результатам сравнения видно, что распараллеливание при вычислении интегралов дает прирост по быстродействию в 2 – 4 раза.

**Вывод:** изучил использование потоков MPI, приводящих к уменьшению вычисления интегралов, на примере методов прямоугольников и Симпсона; сравнил время выполнения алгоритмов численного интегрирования с использованием разных режимов.