**Практическая работа №5**

**Вариант №3.7**

**Формулировка задания.**

Написать параллельную MPI программу приближенного вычисления двойного интеграла с точностью e = 10^-5. Для численного интегрирования использовать метод в соответствии с вашим вариантом (ммк, мс, мпи). Вид функции f(x,y) определяется вариантом задания (1-20). Для контроля точности приближенного вычисления использовать расчет в системе MathCad или аналогичной. Оценить ускорение и эффективность параллельной программы, подготовить отчет.

**Ход работы.**

Схема вычислений:

Делим e на части для процессов, после чего запускаем цикл в котором считаем сколько точек попало в нашу область. После чего используя эту форумулу находим интеграл

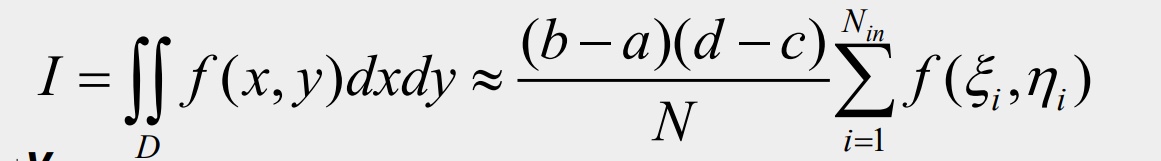
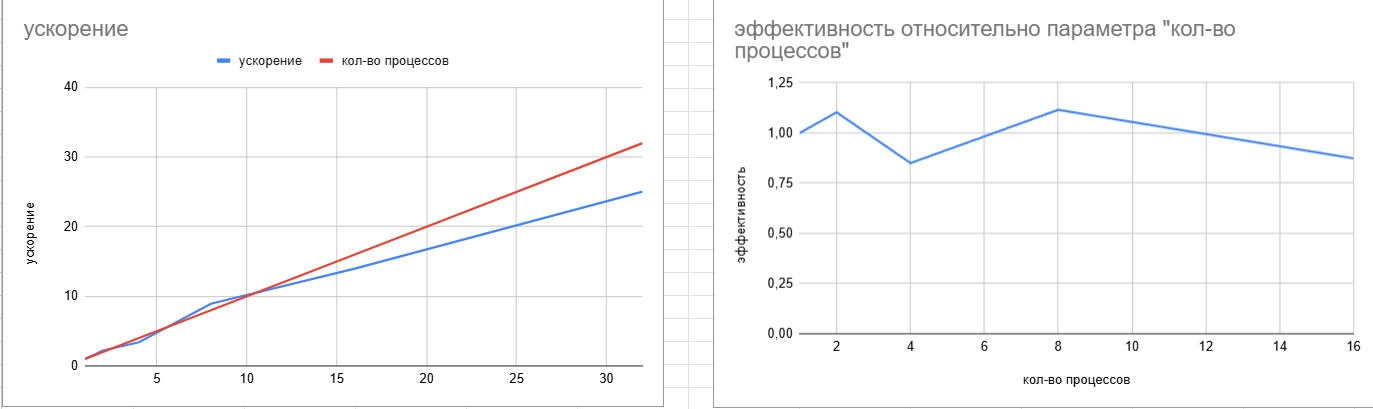
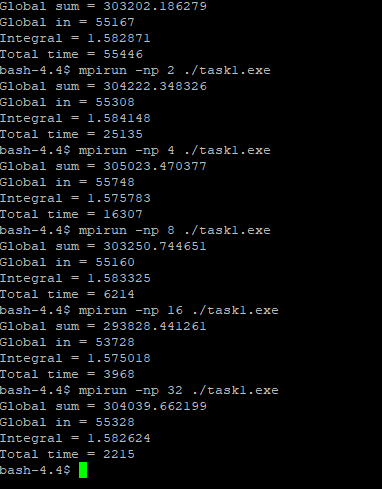


График ускорение и эффективности

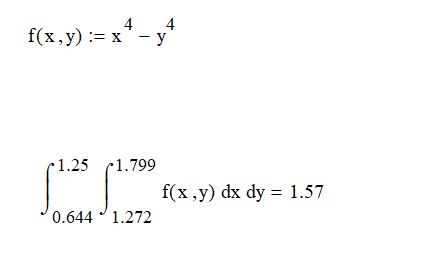
|  |  |  |  |
| --- | --- | --- | --- |
| кол-во процессов | время в мкс | ускорение | эффективность |
| 1 | 55446 | 1 | 1 |
| 2 | 25135 | 2,205927989 | 1,102963994 |
| 4 | 16307 | 3,400134911 | 0,8500337278 |
| 8 | 6214 | 8,922755069 | 1,115344384 |
| 16 | 3968 | 13,97328629 | 0,8733303931 |
| 32 | 2215 | 25,03205418 | 0,782251693 |



Пример(size = 1, 4, 8, 16, 32):



Решение программой MathCad 15:



**Заключение**

В данной работе была написана программа для вычисления кратного интеграла с использованием распараллеливания, с последующим выводом на экран.

**Код программы**

#include <stdio.h>

#include <mpi.h>

#include <cstdlib>

#include <time.h>

#include <math.h>

#include <random>

#include <math.h>

const int E = 10e5;

double Function(double x, double y) {

return pow(x, 4) - pow(y, 4);

}

double IntegrateWithMonteKarloMethod(int n, int& in) {

double sum = 0;

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

{

double x = ((double)rand() / RAND\_MAX) + 1;

double y = ((double)rand() / RAND\_MAX) + 1;

if((1 < x \* y) && (x \* y < 2) && (x \* x - y \* y > 1) && (x \* x - y \* y < 2) && (x > 0)) {

in++;

sum += Function(x, y);

}

}

return sum;

}

int main(int argc, char \*\*argv)

{

int rank, size, rc, tag = 11;

MPI\_Status status;

rc = MPI\_Init(&argc, &argv);

rc = MPI\_Comm\_size(MPI\_COMM\_WORLD, &size);

rc = MPI\_Comm\_rank(MPI\_COMM\_WORLD, &rank);

srand(time(nullptr));

int start\_time = clock();

int global\_n = E;

int local\_n = global\_n / size;

int local\_in = 0;

double local\_sum = IntegrateWithMonteKarloMethod(local\_n, local\_in);

int global\_in = 0;

MPI\_Reduce(&local\_in, &global\_in, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

double global\_sum = 0.0;

MPI\_Reduce(&local\_sum, &global\_sum, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

int finish\_time = clock();

if(rank == 0){

printf("Global sum = %f\n", global\_sum);

printf("Global in = %d\n", global\_in);

double area = 0.6 \* 0.48;

double integral = (area / global\_in) \* global\_sum;

printf("Integral = %f\n", integral);

printf("Total time = %d\n", finish\_time - start\_time);

}

rc = MPI\_Finalize();

}

**Ссылка на код на кластере**

/home/g931901/g932204/Sobol/lab6/task1.cpp