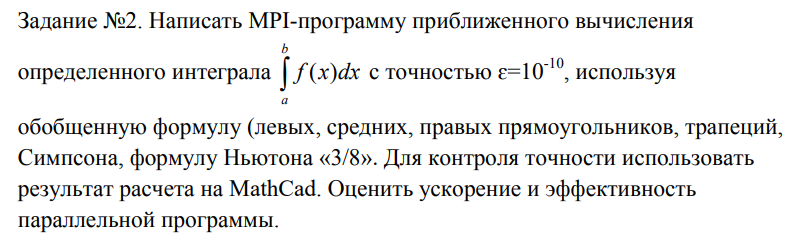
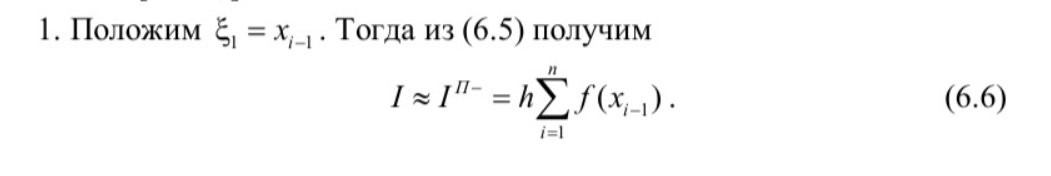
Отчет №4

Гренц Елизавета Алексеевна группа 932220



Описание работы:

Исходную функцию интеграла будем определять в функции f(x). За формулу метода левых прямоугольников будем принимать: реализованную в функции left\_rectangle\_integral(double a, double b, int n) После подсчета локальных сумм мы будем их суммировать с помощью MPI\_Reduce с параметром MPI\_SUM и записывать это в integral в нулевом процессе. За подсчет времени будет отвечать MPI\_Wtime. Подсчет ускорения: Sp = T1 Tp , где T1 – время выполнения программы на одном процессе, а Tp – время выполнения программы на p процессах. Подсчет эффективности: Ep = Sp p

Более подробно :

Интервал [a, b] распределяется между процессами. Каждому процессу назначается часть интервала для вычисления интеграла.  
Локальное вычисление : каждый процесс независимо вычисляет свой локальный интеграл, используя метод левого прямоугольника. Интервал, назначенный каждому процессу, делится на подинтервалы, и к каждому подинтервалу применяется метод левого прямоугольника. Локальный интеграл вычисляется как сумма значений функции на левых концах этих подинтервалов, умноженная на ширину каждого подинтервала.  
После чего локальные интегралы, вычисленные каждым процессом, уменьшаются с использованием MPI\_Reduce функцией с MPI\_SUM операцией. Эта операция объединяет локальные интегралы всех процессов и вычисляет их сумму, в результате чего получается глобальный интеграл.

Код для вставки :

#include <stdio.h>

#include <mpi.h>

#include <math.h>

// РІС‹С‡РёСЃР»СЏРµРј С„СѓРЅРєС†РёСЋ f(x)

double f(double x) {

return (1 / (sqrt(x) \* (exp(0.9 \* x) + 3)));

}

// РјРµС‚РѕРґ Р»РµРІС‹С… РїСЂСЏРјРѕСѓРіРѕР»СЊРЅРёРєРѕРІ

double left\_rectangle\_integral(double a, double b, int n) {

double h = (b - a) / n;

double sum = 0.0;

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

double x = a + i \* h;

sum += f(x); // СЃСѓРјРёСЂСѓРµРј Р·РЅР°С‡РµРЅРёРµ

}

return h \* sum; // РёС‚РѕРіРѕРІРѕРµ РёРЅС‚РµРіСЂР°Р»СЊРЅРѕРµ Р·РЅР°С‡РµРЅРёРµ

}

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

int rank, size;

double a = 0.5, b = 2.0;

int n = 100000000;

double integral, st, fin;

MPI\_Init(&argc, &argv);

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

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

st = MPI\_Wtime();

// РєР°Р¶РґС‹Р№ РїСЂРѕС†РµСЃСЃ РІС‹С‡РёСЃР»СЏРµС‚ СЃРІРѕР№ Р»РѕРєРѕР»СЊРЅС‹Р№ РёРЅС‚РµРіСЂР°Р»

double local\_integral = left\_rectangle\_integral(a + rank \* (b - a) / size, a + (rank + 1) \* (b - a) / size, n/size);

MPI\_Reduce(&local\_integral, &integral, 1, MPI\_DOUBLE, MPI\_SUM, 0, MPI\_COMM\_WORLD);

fin = MPI\_Wtime();

if (rank == 0) {

printf("Integral: %.10lf, Time: %.10lf seconds\n", integral, fin - st);

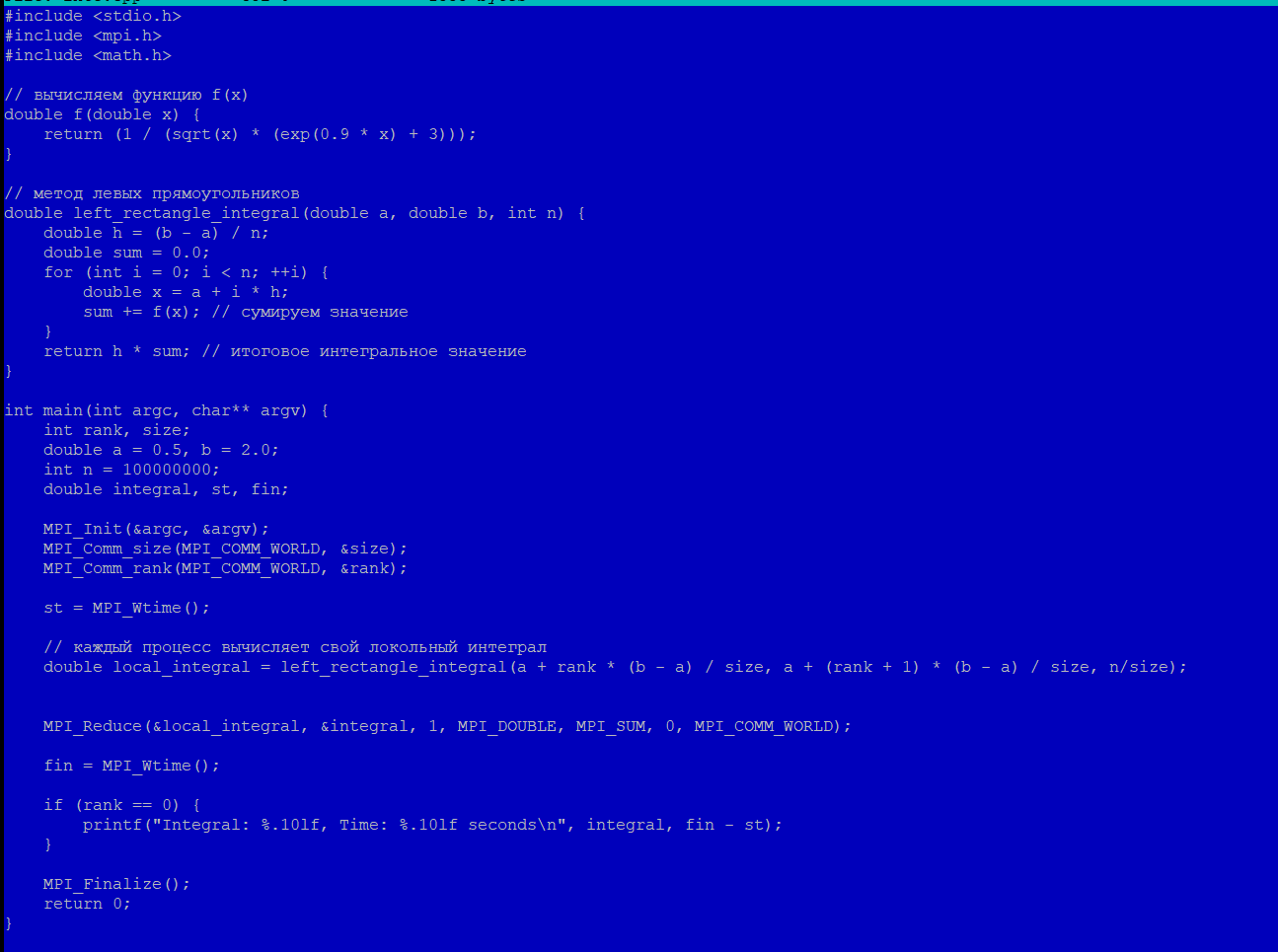
}

MPI\_Finalize();

return 0;

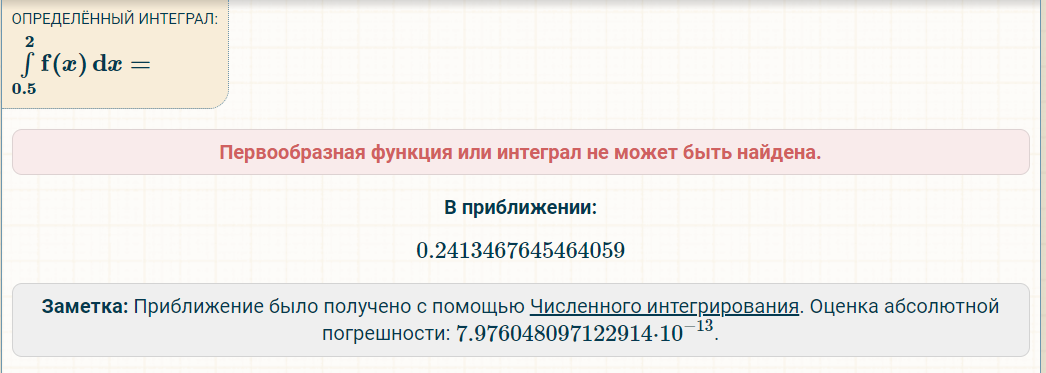
}

Скрин\_для \_просмотра



**Результаты**

Integral: 0.2413467663



**1 процесс**

Time: 1.6881051064 seconds

**2 процесса**

Time: 0.8421561718 seconds

**4 процесса**

Time: 0.5959069729 seconds

**6 процессов**

Time: 0.2886161804 seconds

**8 процессов**

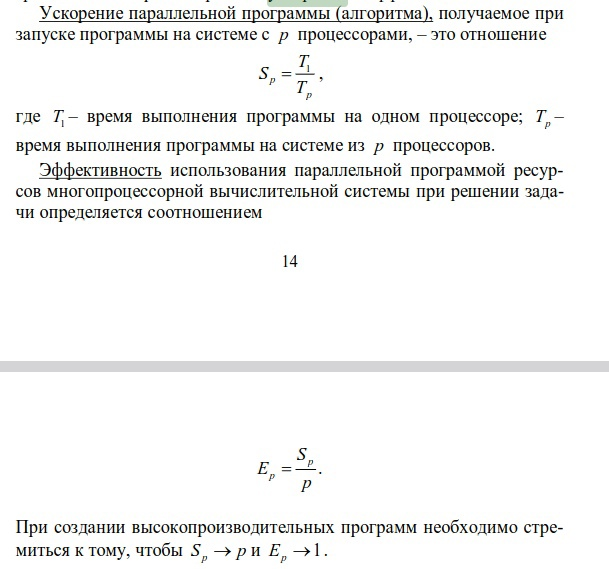
Time: 0.2147240639 seconds

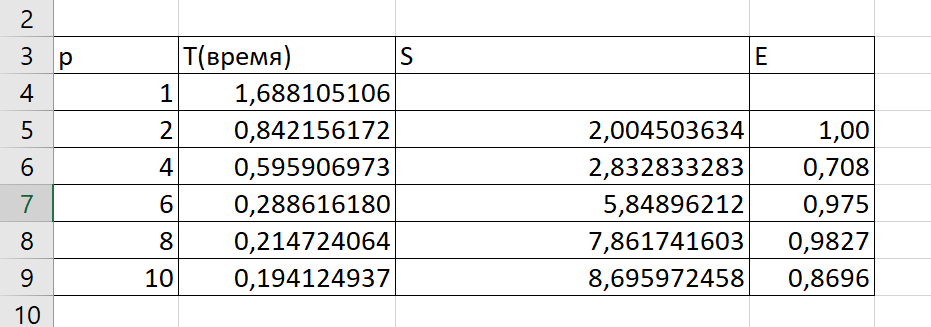
**10 процессов**

Time: 0.1941249371 seconds

**Вывод :**

Оценим Ускорение и Эффективность





С увеличением кол-ва процессов уменьшается время выполнения программы, но самая высокая эффективность достигается не на самом большом числе процессоров, моя программа лучше всего работает на двух процессах. Возможно по мере усложнения вычислений результат может измениться.