**Практическая работа №5**

**Вариант №3.7**

**Формулировка задания.**

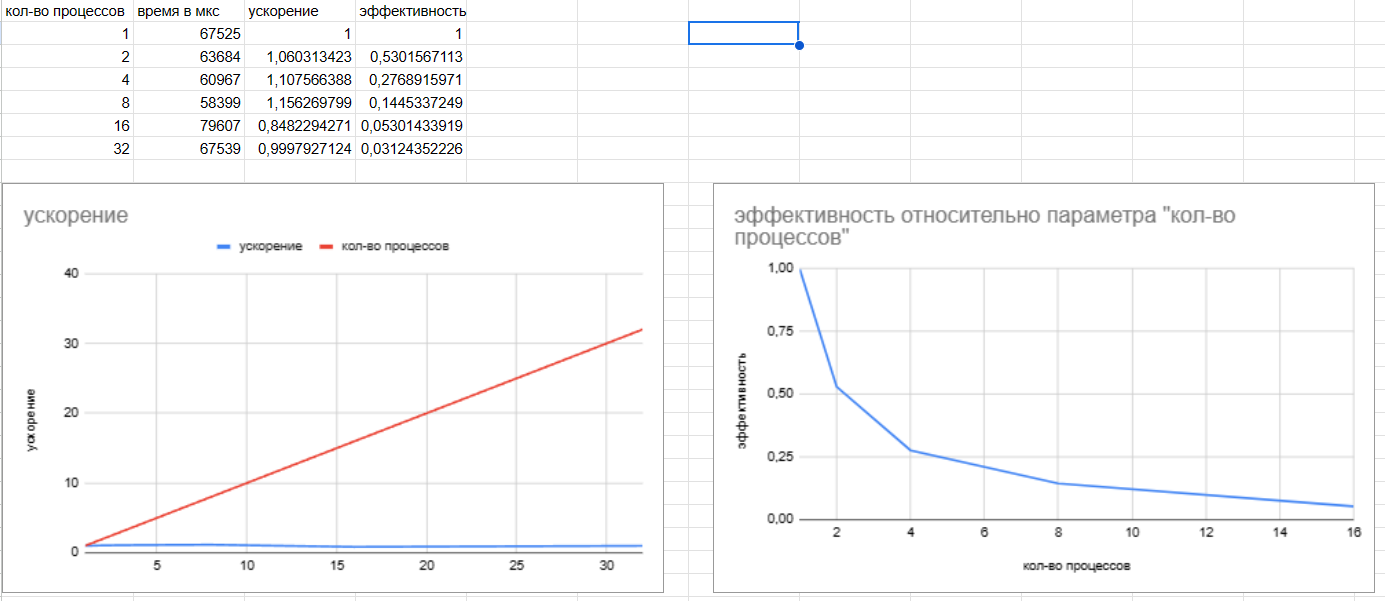
Написать параллельную MPI программу приближенного вычисления определенного интеграла с точностью e = 10-5. Вид обобщенной формулы численного интегрирования определяется вашим вариантом (мсп, мт, мс). Для контроля точности приближенного вычисления использовать расчет в системе MathCad или аналогичной. Оценить ускорение и эффективность параллельной программы, подготовить отчет.

**Ход работы.**

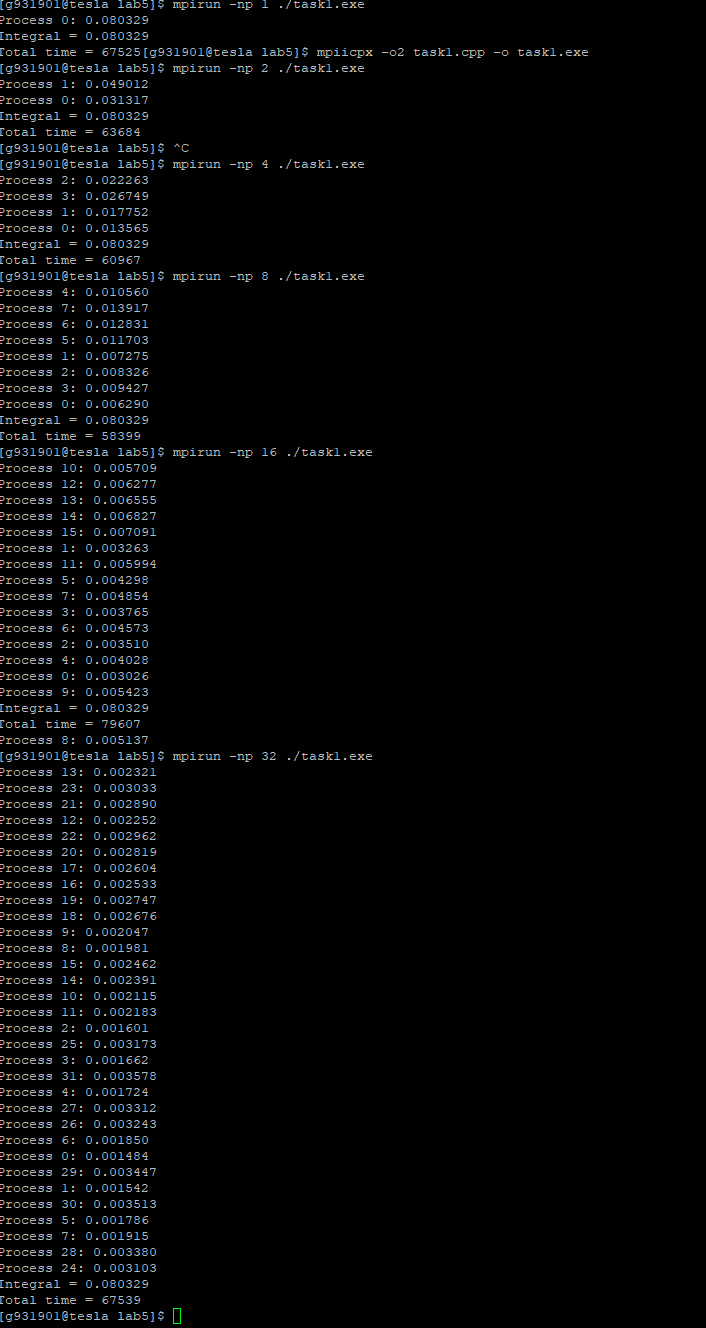
Схема вычислений:

Сначала мы определяем шаг на который мы будем делить наш отрезок [a,b]. И раздаем каждому процессу свой отрезок. После чего подсчета каждого интеграла мы собираем сумму с помощью MPI\_Reduce на нулевом процессе.

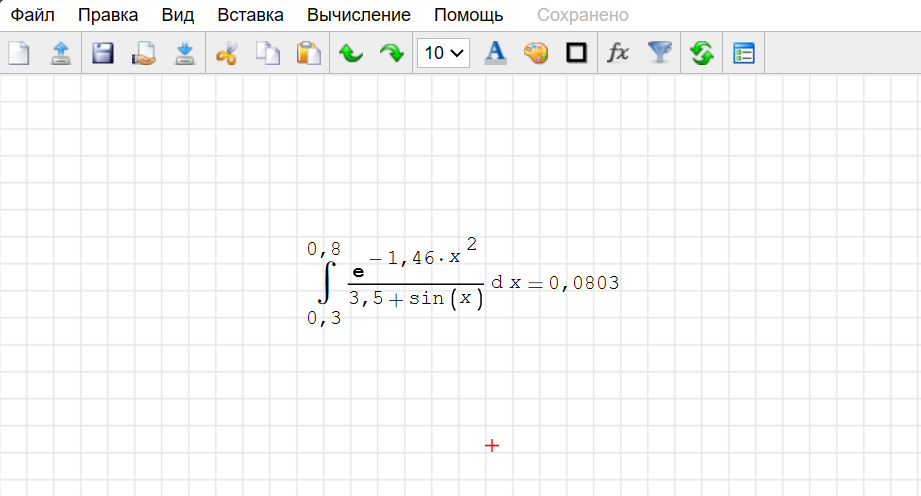
График ускорение и эффективности



Пример(size = 1, 4, 8, 16, 32):



Решение онлайн сервиса подобному Mathcad([SMath Studio Cloud](https://freeanalogs.ru/Online/SMathCloud)):



**Заключение**

В данной работе была написана программа для распараллеливание векторных операций по процессам, 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 double E = 10e5;

double Function(double x) {

return exp(-1.46 \* x \* x) / (3.5 + sin(x));

}

double IntegrateWithSimpsonMethod(double a, double b, int n) {

double width = (b - a) / n;

double simpson\_integral = 0;

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

double x1 = a + step \* width;

double x2 = a + (step + 1) \* width;

simpson\_integral += (x2 - x1) / 6.0 \* (Function(x1) + 4.0 \* Function(0.5 \* (x1 + x2)) + Function(x2));

}

return simpson\_integral;

}

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);

double global\_a = 0.3;

double global\_b = 0.8;

double global\_step = (global\_b - global\_a) / size;

double local\_a = global\_b - global\_step \* (rank + 1);

double local\_b = global\_b - global\_step \* rank;

int start\_time = clock();

double local\_integral = IntegrateWithSimpsonMethod(local\_a, local\_b, E);

printf("Process %d: %f\n", rank, local\_integral);

double global\_integral;

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

int finish\_time = clock();

if(rank == 0){

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

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

}

rc = MPI\_Finalize();

}

**Ссылка на код на кластере**

/home/g931901/g932204/Sobol/lab5/task1.cpp