**Практическая работа №8**

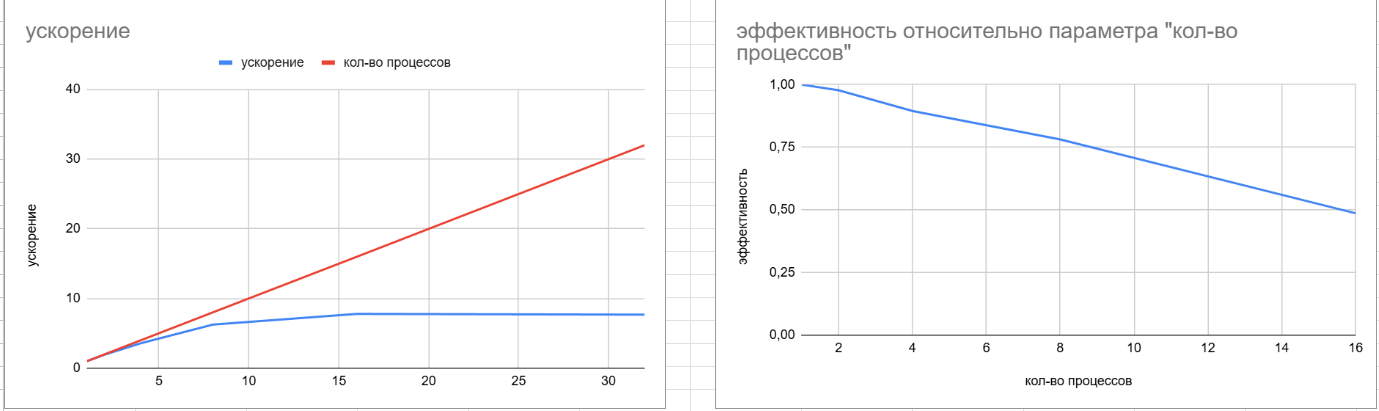
**Вариант №3.7**

**Формулировка задания.**

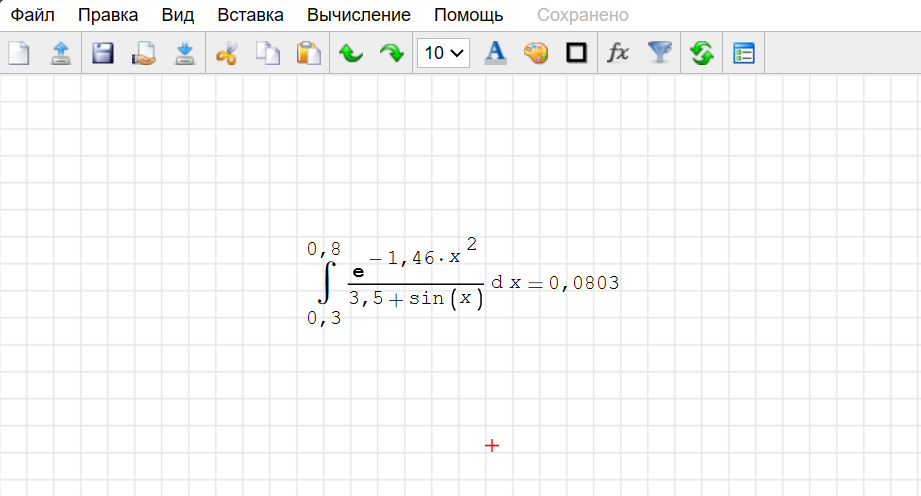
Написать параллельную OpenMP программу приближенного вычисления определенного интеграла с точностью e = 10-5. Вид обобщенной формулы численного интегрирования с помощью метода Симпсона. Для контроля точности приближенного вычисления использовать расчет в системе MathCad или аналогичной. Оценить ускорение и эффективность параллельной программы, подготовить отчет.

График ускорение и эффективности

|  |  |  |  |
| --- | --- | --- | --- |
| кол-во процессов | время в мкс | ускорение | эффективность |
| 1 | 41270 | 1 | 1 |
| 2 | 21104 | 1,95555345 | 0,9777767248 |
| 4 | 11531 | 3,579047784 | 0,8947619461 |
| 8 | 6602 | 6,251136019 | 0,7813920024 |
| 16 | 5301 | 7,785323524 | 0,4865827202 |
| 32 | 5376 | 7,67671131 | 0,2398972284 |



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



**Заключение**

В данной работе мы вычислили определённый интеграл методом Симпсона, используя библиотеку   
OpenMP и оценили ускорение и эффективность параллельной программы.

**Код программы**

#include <iostream>

#include <omp.h>

#include <cmath>

#define SIZE 32

int const n = 10e8;

double f(double x) {

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

}

void CheckAnswer(double actual) {

using namespace std;

double exception = 0.08;

cout << "Ответ: " << actual << '\n';

cout << "Ожидаемый ответ: " << exception << '\n';

cout << "Погрешность: " << abs(actual - exception) << '\n';

}

int main()

{

setlocale(LC\_ALL, "Russian");

omp\_set\_num\_threads(SIZE);

double a = 0.3;

double b = 0.8;

double integral = 0.0;

double h = (b - a) / n;

clock\_t start\_time = clock();

#pragma omp parallel

{

#pragma omp for reduction(+ : integral)

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

double x1 = a + step \* h;

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

integral += (x2 - x1) / 6.0 \* (f(x1) + 4.0 \* f(0.5 \* (x1 + x2)) + f(x2));

}

}

CheckAnswer(integral);

std::cout << "Время: " << (double)(clock() - start\_time) << '\n';

}