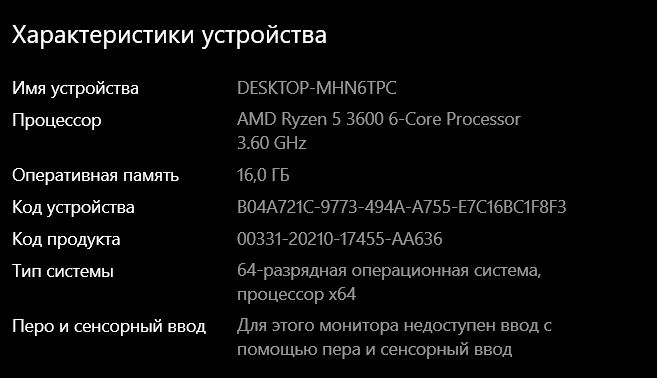
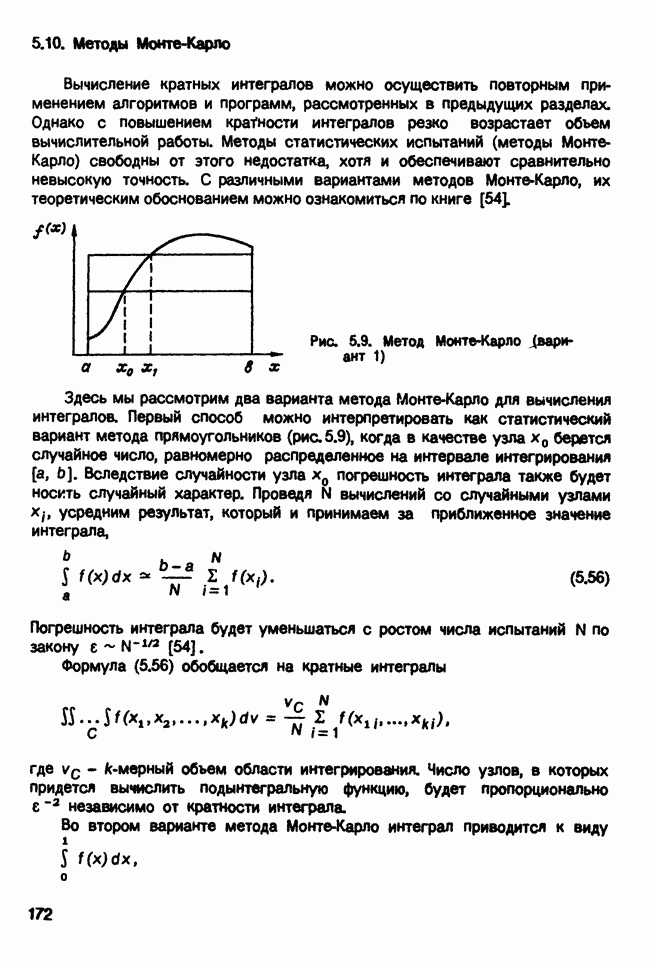
Work 3 Кириленко Константин 351





// 01\_seq.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include <iostream>

#include <omp.h>

#include <cstdlib>

#define PI 3.1415926535897932384626433832795

#define e 2.718281828459045235360287471

using namespace std;

double rand\_range(double a, double b)

{

if (a > b) return rand\_range(b, a);

return a + (b - a) \* rand() / RAND\_MAX;

}

void integral(const double a, const double b, const double a2, const double b2,

const double h, const double h2, double\* res)

{

int i, j, n, n2;

double sum; // локальная переменная для подсчета интеграла

double x, y; // координата точки сетки

n = (int)((b - a) / h); // количество точек сетки интегрирования

n2 = (int)((b2 - a2) / h2); // количество точек сетки интегрирования

sum = 0.0;

#pragma omp parallel for reduction(+ : sum)

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

{

x = rand\_range(a, b);

y = rand\_range(a2, b2);

sum += ((pow(e, sin(PI \* x) \* cos(PI \* y)) + 1) / ((b - a) \* (b2 - a2)));

}

\*res = (b - a) \* (b2 - a2) \* sum / n;

}

double experiment(double\* res)

{

double stime, ftime; // время начала и конца расчета

double a = 0.0; // левая граница интегрирования

double b = 16.0; // правая граница интегрирования

double a2 = 0.0; // левая граница интегрирования

double b2 = 16.0; // правая граница интегрирования

double h = 0.000001; // шаг интегрирования

stime = clock();

integral(a, b, a2, b2, h, h, res); // вызов функции интегрирования

ftime = clock();

return (ftime - stime) / CLOCKS\_PER\_SEC;

}

int main()

{

std::srand(std::time(nullptr));

int i; // переменная цикла

double time; // время проведенного эксперимента

double res; // значение вычисленного интеграла

double min\_time; // минимальное время работы

// реализации алгоритма

double max\_time; // максимальное время работы

// реализации алгоритма

double avg\_time; // среднее время работы

// реализации алгоритма

int numbExp = 10; // количество запусков программы

// первый запуск

min\_time = max\_time = avg\_time = experiment(&res);

// оставшиеся запуски

for (i = 0; i < numbExp - 1; i++)

{

time = experiment(&res);

avg\_time += time;

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

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

}

// вывод результатов эксперимента

cout << "execution time : " << avg\_time / numbExp << "; " <<

min\_time << "; " << max\_time << endl;

cout.precision(8);

cout << "integral value : " << res << endl;

return 0;

}

- парал



- послед

