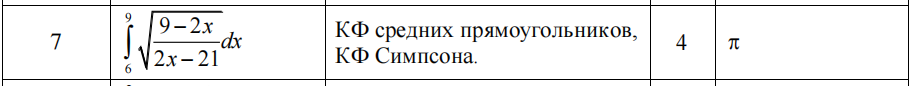
**Лабораторная работа № 3**

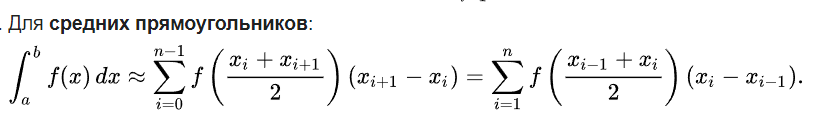
**«Приближенное вычисление интегралов»**

Вариант 7

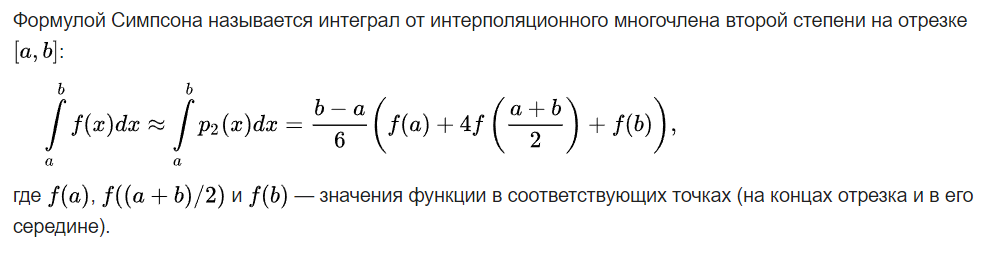
Исполнитель: Кохан Даниил, 11 группа 2024 год

Вариант 7:



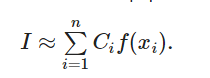
Формулы для вычислений:  
  


Формула Симпсона:



Оценка погрешности:



Для задания 2:

Преобразование узлов на отрезок:

x[i] = a + (b - a) \* (x[i-1] + 1) / 2

Преобразование весов

W[i] = (b-a)/2 \* W[i-1];

**Задание 1:**

Вычисление КФ средних прямоугольников,

КФ Симпсона.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Квадратурная формула | Число разбиений | Шаг | Приближенное значение интеграла | Оценка погрешности | Абсолютная погрешность |
| КФ средних прямоугольников | 2  4  8  16  32  64  128  256  512  1024  2048 | 1.5  0.75  0.375  0.1875  0.09375  0.046875  0.0234375  0.0117188  0.00585938  0.00292969  0.00146484 | 3.09839  3.12994  3.13861  3.14084  3.1414  3.14155  3.14158  3.14159  3.14159  3.14159  3.14159 | 0.010517  0.002891 0.000743976  0.000187443  4.69537e-05  1.17443e-05  2.93643e-06  7.34131e-07  1.83534e-07  4.58836e-08 | 0.0116551  0.0029821  0.00075017  0.00018784  4.69786e-05  1.17458e-05  2.93653e-06  7.34137e-07  1.83535e-07  4.58837e-08 |
| КФ Симпсона | 2  4  8  16  32  64  128  256 | 1.5  0.75  0.375  0.1875  0.09375  0.046875  0.0234375  0.0117188 | 3.14294  3.1417  3.1416  3.14159  3.14159  3.14159  3.14159  3.14159 | 0.000137302  6.54675e-06  4.43138e-07  2.83274e-08  1.78085e-09  1.11468e-10  6.96912e-12 | 0.000105302  7.10048e-06  4.53408e-07  2.84963e-08  1.78352e-09  1.11508e-10  6.97087e-12 |

Вывод:

Исходя из представленных результатов, можно сделать вывод о том, что метод Симпсона обеспечивает более точное вычисление интеграла по сравнению с методом средних прямоугольников.

Листинг программы с комментариями

#include<vector>

#include<cmath>

#include <iostream>

using namespace std;

const double M\_PI = 3.14159265358979323846;

int SIZE = 2;

const double a = 6;

const double b = 9;

double h = (b - a) / SIZE;

//вывод вектора

void printVector(vector<double> vec)

{

for (int i = 0; i < vec.size(); i++)

{

cout << vec[i] << " ";

}

cout << endl;

}

//функция вычисления узлов Средних прямоугольников

vector<double> getPoints(double a, double b)

{

vector<double> points;

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

{

points.push\_back(a + h \* (i + 0.5));

}

return points;

}

//функция получения узлов Симпсона четных

vector<double> getPointsSimpson(double a, double b)

{

std::vector<double> points(SIZE+1);

for (int i = 0; i < SIZE+1; ++i) {

points[i] = a + i \* h;

}

return points;

}

double f(double x)

{

return sqrt((9 - 2 \* x) / (2 \* x - 21));

}

//I culculation

double Iculc1(vector<double>& vec)

{

double I = 0;

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

{

I+=f(vec[i]);

}

return I;

}

//оценка погрешности составных квадратных формул

double errorMark(double I1, double I2, int p)

{

return abs(I1 - I2) / (pow(2, p) - 1);

}

double Iculc2(vector<double>& vec)

{

double I = 0;

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

{

I += (f(vec[i]) + 4\*f(vec[i]+h/2) + f(vec[i+1]));

}

return I;

}

int main()

{

//вычисление интегралов методом Квадратурных формул(КФ средних прямоугольников,КФ Симпсона)

setlocale(LC\_ALL, "rus");

vector<double> points = getPoints(a, b);

//printVector(points);

double I = Iculc1(points);

I = I \* h;

cout << h << endl;

cout << "I = " << I << endl;

cout << "I точное = " << M\_PI << endl;

SIZE \*= 2;

h = (b - a) / SIZE;

vector<double> points1\_ = getPointsSimpson(a, b);

double I1\_ = Iculc2(points1\_);

I1\_ = I1\_ \* h / 6;

cout << "Error оценка = " << errorMark(I, I1\_, 2) << endl;

cout << "Error = " << abs(M\_PI - I) << endl;

cout<< endl;

SIZE /= 2;

h = (b - a) / SIZE;

vector<double> points2 = getPointsSimpson(a, b);

double I2 = Iculc2(points2);

I2 = I2 \* h / 6;

cout << h << endl;

cout << "I2 = " << I2 << endl;

cout << "I2 точное = " << M\_PI << endl;

SIZE \*= 2;

h = (b - a) / SIZE;

vector<double> points2\_ = getPointsSimpson(a, b);

double I2\_ = Iculc2(points2\_);

I2\_ = I2\_ \* h / 6;

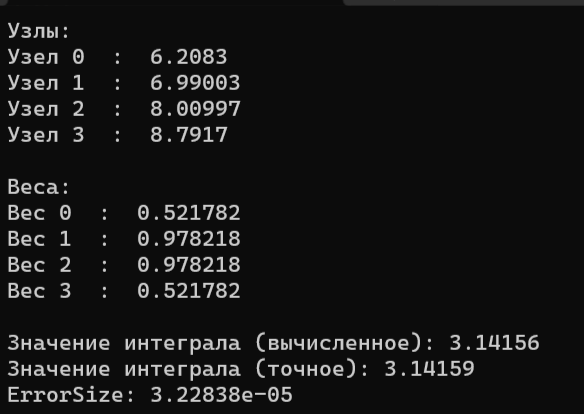
cout << "Error оценка = " << errorMark(I2, I2\_, 2) << endl;

cout << "Error2 = " << abs(M\_PI - I2) << endl;

return 0;

}

Задание 2



Листинг программы с комментариями

#include <vector>

#include <cmath>

#include <iostream>

const double M\_PI = 3.14159265358979323846;

const int stage = 4;

using namespace std;

double f(double x)

{

return sqrt((9 - 2 \* x) / (2 \* x - 21));

}

int main() {

setlocale(LC\_ALL, "rus");

// Узлы и веса КФ НАСТ

vector<double> nativePoints = { -0.86113631, -0.33998104, 0.33998104, 0.86113631 };

vector<double> values\_weight = { 0.34785485, 0.65214515, 0.65214515, 0.34785485 };

// Преобразование узлов на отрезок [6, 9]

// по ф-ле x[i] = a + (b - a) \* (x[i-1] + 1) / 2

vector<double> points;

for (double point : nativePoints) {

points.push\_back(6 + (9 - 6) \* (point + 1) / 2);

}

// Преобразование весов

vector<double> weights\_transformed;

for (double weight : values\_weight) {

weights\_transformed.push\_back((9 - 6) \* weight / 2);

}

double integral = 0;

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

integral += weights\_transformed[i] \* f(points[i]);

}

cout << "Узлы: "<<endl;

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

cout << "Узел " << i << " : " << points[i] << endl;

}

cout << endl;

cout << "Веса: "<<endl;

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

cout << "Вес " << i << " : " << weights\_transformed[i]<<endl;

}

cout << endl;

cout <<"Значение интеграла (вычисленное): "<<integral<<endl;

cout << "Значение интеграла (точное): " << M\_PI << endl;

cout << "ErrorSize: " << abs(M\_PI - integral) << endl;

return 0;

}