**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практической работе №8**

**по дисциплине «Вычислительная математика»**

Тема: Формула Гаусса

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Киреев К.А. |
| Преподаватель |  | Щеголева Н.Л. |

Санкт-Петербург

2019

**Цель работы.**

Изучение и сравнение различных методов численного интегрирования на примере квадратурной формулы Гаусса.

**Основные теоретические положения.**

В квадратурной формуле Гаусса

(1)

узлы *xj* и веса подобраны так, чтобы формула была точна для всех многочленов степени . Для приближённого вычисления интеграла по конечному отрезку  выполняется замена переменной:

(2)

тогда квадратурная формула Гаусса принимает вид:

(3)

Узлы квадратурной формулы Гаусса *xj*являются корнями полинома Лежандра:

*.* (4)

Полином можно выразить рекуррентно :

, (5)

причем первые две функции имеют вид *P0(x)* =1 и *P1(x) =x*

Производную полинома можно найти как:

. (6)

Корни полинома Лежадра можно вычислить итеративно, используя метод Ньютона:

, (7)

причем начальное приближение для *j*-го корня берется по формуле:

. (8)

Гауссовы веса вычисляются по формуле:

. (9)

Если подынтегральная функция достаточно гладкая, то квадратурная формула Гаусса обеспечивает очень высокую точность при небольшом числе узлов, так как для погрешности *Rn* формулы Гаусса с n узлами справедлива оценка:

. (10)

**Постановка задачи.**

Используя квадратную формулу Гаусса (2*n*-1)-го порядка точности вычислить приближенное значение заданного интеграла:

. (11)

**Выполнение работы.**

Составлена функция *F* для вычисления значения подынтегральной функции. После чего были составлены подпрограммы *P* и *P1* для вычисления значения полинома Лежандра n-степени и его производной соответственно. Данные функции необходимы для вычисления узлов, а также весов в формуле Гаусса, для чего были составлены отдельные функции. Корни полинома Лежандра были вычислены методом Ньютона. Также была составлена подпрограмма функция *GAUSS*(см. Приложение А) для вычисления интеграла по формуле Гаусса.

Используя все написанные функции создана головная программа, которая вызывает функцию *GAUSS* с разными аргументами и выводит искомое значение.

Значения представлены в таблице 1:

Таблица 1 – Приближенные значения интегралов

|  |  |
| --- | --- |
| Порядок точности | Приближенное значение интеграла |
| 2 | 0.270241 |
| 5 | 0.273818 |
| 7 | 0.273788 |
| 9 | 0.273788 |
| 11 | 0.273788 |
| 12 | 0.273788 |

Таблица 2 – Погрешность

|  |  |
| --- | --- |
| Порядок точности | Погрешность |
| 2 | 0.00785674 |
| 5 | 0.000795046 |
| 7 | 0.000342825 |
| 9 | 0.000182899 |
| 11 | 0.000110748 |
| 12 | 8.90973e-05 |

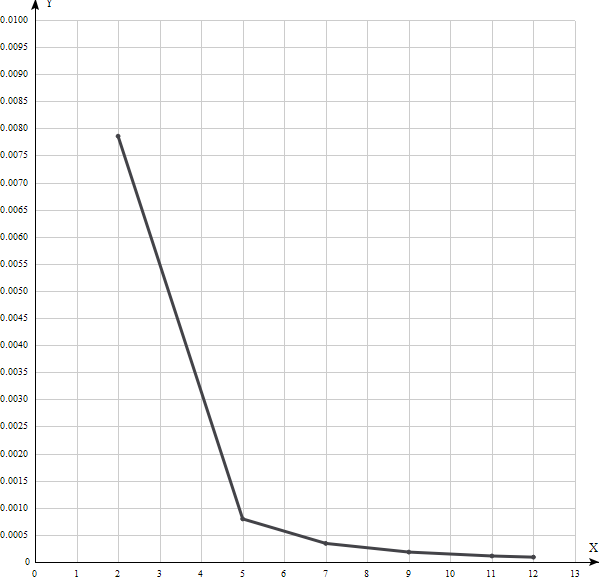
На основе полученных данных построен график зависимости погрешности вычислений от порядка точности

Рисунок 1 – График зависимости погрешности от порядка точности(*n*)

**Выводы.**

Изучен метод нахождения значения интеграла с помощью формы Гаусса. Написана программа для получения приближенных значений интеграла в зависимости от порядка точности. В сравнении с прошлыми способами вычисления интегралов, можно сделать вывод, что точность расчёта по формуле Гаусса в разы выше, чем для методов прямоугольников, трапеций и Симпсона. Вычисление интегралов с помощью квадратурных формул Гаусса дает очень высокую точность.

Приложение А

ТЕКСТ ОСНОВНОЙ ПРОГРАММЫ

#include <iostream>

#include <fstream>

#include <cmath>

using namespace std;

double F(double x); // вычисление значения подынтегральной функции

double P(double n, double x); // вычисление значения полинома Лежандра n-ой степени

double P1(double n, double x); // вычисление значения производной полинома Лежандра n-ой степени

double xj\_for\_next\_k(double n, double x0, double Eps); // вычисление корней полинома Лежандра

double xj\_0(double n, double j); // начальное приближение для j-го корня

double wj(double n,double x); // вычисление Гауссовых весов

double Rn(double n); // вычисление погрешности

double GAUSS(double n); // вычичление интеграла по формуле Гаусса

int main()

{

    int N[6] = { 2, 5, 7, 9, 11, 12 };

    for (int i = 0; i < 6; i++)

    {

        cout << "Порядок точности: " << N[i] << endl;

        cout << "Приближенное значение интеграла: " << GAUSS(N[i]) << endl;

        cout << "Погрешность: " << Rn(N[i]) << endl << endl;

    }

    return 0;

}

double F(double x)

    { return (x\*cos(x)/(1+(x\*x))); }

double P(double n, double x)

{

    if (n == 0) return 1;

    if (n == 1) return x;

    return (((2 \* (n - 1) + 1) / ((n - 1) + 1)) \* x \* P(n - 1, x) - ((n - 1) / ((n - 1) + 1)) \* P(n - 2, x));

}

double P1(double n, double x)

    { return((n / (1 - x \* x)) \* (P(n - 1, x) - x \* P(n, x))); }

double xj\_for\_next\_k(double n, double x0, double Eps)

{

    double Y, Y1, DX;

    do

    {

        Y = P(n, x0);

        if (Y == 0.0) break;

        Y1 = P1(n, x0);

        DX = Y / Y1;

        x0 -= DX;

    } while (fabs(DX) > Eps);

    return x0;

}

double xj\_0(double n, double j)

    { return cos(3.14 \* (4 \* j - 1) / (4 \* n + 2)); }

double wj(double n,double x)

    { return (2 / ((1 - pow(x, 2)) \* pow(P1(n, x), 2))); }

double Rn(double n)

    { return ((1.0) / (2.5 \* sqrt(n)) \* pow(((1.0) / (3 \* n)), 2) \* 1.0); }

double GAUSS(double n)

{

    double result = 0.0, a = 0.0, b = 1.0;

    for (double j = 1; j <= n; j++)

    {

        double Tj = ((a + b) / 2 + ((b - a) / 2 \* xj\_for\_next\_k(n, xj\_0(n, j), Rn(n))));

        double Wj = wj(n, xj\_for\_next\_k(n, xj\_0(n, j), Rn(n)));

        double f = F(Tj);

        result += Wj \* f;

    }

    return ((b - a) / 2) \* result;

}