МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(МОСКОВСКИЙ ПОЛИТЕХ)**

09.03.01 Информатика и вычислительная техника

Образовательная программа (профиль)

«Интеграция и программирование в САПР»

Кафедра «СМАРТ технологии»

МЕТОДИЧЕСКОЕ ПОСОБИЕ

по дисциплине:

**Проектная деятельность**

на тему:

Приближённое вычисление интегралов

квадратурой Гаусса методом Гаусса-Лежандра.

Преподаватель: / Толстиков А.В., к.т.н. /

*подпись ФИО, уч. звание и степень*

Студент: / Свинцицкий Р.Е. 201-323 /

*подпись ФИО, группа*

Москва, 2022 г.

# ТЕОРИТИЧЕСКАЯ ЧАСТЬ

Под задачей приближённого вычисления интегралов понимают вычисление определённого интеграла при условии, что известны отдельные значения подинтыгральной функции и некоторые её общие свойства.

Разобьём отрезок [*a,b*] точками *a = x0<x1<…<xn=b* на N элементарных отрзков . По свойству адаптивности интеграл I можно представить в виде:

По теореме о среднем значении интегрального исчисления

где , *Rn*[*f*] – погрешность вычисления.

Функция должна быть линейной. Тогда

Приближённое равенство

Если узловые точки не заданы сначала, то можно попытаться определить их таким образом, чтобы получилась формула как можно большего порядка.

Это приводит к *квадратуре Гаусса*. Подставляя в формулу , получим следующую систему уравнений

Узлы интегрирования *xi* суть корни многочлена Эрмитта.

**Многочлен Лежа́ндра** — многочлен, который в наименьшей степени отклоняется от нуля в *смысле среднего квадратического*. Образует ортогональную систему многочленов на отрезке [-1, 1] [ − 1 , 1 ] {\displaystyle [-1,\;1]} в пространстве *L2*L 2 {\displaystyle L^{2}}. Многочлены Лежандра могут быть получены из многочленов *{1, x, x2, x3, …}* { 1 , x , x 2 , x 3 , … } {\displaystyle \{1,\;x,\;x^{2},\;x^{3},\;\ldots \}} ортогонализацией Грама ― Шмидта.

Названы по имени французского математика Адриен Мари Лежандра.

Общее уравнение для многочленов Эрмита имеет вид:

Уравнение H n ( x ) = 0 {\displaystyle H\_{n}(x)=0} имеет n n {\displaystyle n} вещественных корней, попарно симметричных относительно начала системы координат, и модуль каждого из них не превосходит величины n ( n − 1 ) / 2 {\displaystyle {\sqrt {n(n-1)/2}}} .

Для нахождения корней используется метод простых итераций.

Метод итерации — численный метод решения математических задач, используемый для приближённого решения алгебраических уравнений и систем. Суть метода заключается в нахождении по приближённому значению величины следующего приближения (являющегося более точным).

Для применения метода итераций необходимо сначала отделить корни уравнения.

Для этого используется формула , где А – наибольший коэффициент многочлена, а {an} – коэффициент при переменной с наибольшей степенью.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

Основная часть программы содержится в пространстве имён «GausMethodIntegral» и состоит из 2–х классов и функции, интеграл которой требуется найти.

Первый класс называется “Methods”, содержит в себе функции для вычислений систем уравнений методом Гаусса, определённых степенных интегралов, факториала числа, корня полинома Лагранжа и является абстрактной. Рассмотрим процедуру нахождения корня полинома Лагранжа (Листинг 1) подробнее.

Она принимает на вход номер корня (переменная i) и порядок полинома Лагранжа (переменная n). Функция состоит из 3-х частей: цикл нахождения бинома Ньютона по формуле полинома Лагранжа, нахождения n-й производной и поиска корней. Также эта функция использует другую, вспомогательную функцию (Листинг 2), для нахождения решения полинома Лагранжа при заданном x).

Листинг 1

// Поиск корней полинома Лежандра

protected double Gauss\_Legandr(int i, int n, double eps = 0.01f, double e = 0.001f)

{

int cnt;

cnt = (int)Math.Pow(2, n);

double[] a = new double[cnt + 1];

// Бином Ньютона

for (int k = 0; k <= cnt; k++)

{

double t = factorial(n) / (factorial(k) \* factorial(n - k)) \* Math.Pow(-1, k);

a[2 \* (n - k)] = t;

}

for (int j = 0; j < n; j++)

{

for (int c = 0; c <= cnt; c++)

{

if (c == 0)

{

a[c] = 0;

continue;

}

if (a[c] == 0)

continue;

a[c - 1] = a[c] \* c;

a[c] = 0;

}

}

//double C = 1 / (Math.Pow(2, n) \* factorial(n));

// Поиск корней

//поиск отрезков

double max = a[0];

int mp = 0;

for (int c = 1; c <= cnt; c++)

{

if (a[c] > max)

max = a[c];

if (a[c] != 0)

mp = c;

}

double bx = 1 + max / a[mp];

double ax = -bx;

int number = 1;

for (double X = ax; X <= bx; X += e)

{

if ((Math.Abs(Legandr(X, a)) <= eps) && (number == i))

return X;

number++;

}

return -1;

}

Листинг 3

private double Legandr(double x, double[] func)

{

double ret = 0;

for (int i = 0; i < func.Length; i++)

{

ret += Math.Pow(x, i) \* func[i];

}

return ret;

}

Другой класс является наследником первого и содержит в себе переменные:

1. N – порядок полинома Лагранжа и количество точек;
2. a, b – интервал интегрирования;
3. function – ссылка на функцию, интеграл которой требуется найти;

и функции, производящих расчёт. Функция “Solve” принимает данные, проверяет их и записывает в вышеуказанные переменные. Функция “glsn” производит вычисления. Рассмотрим её подробнее.

Функция состоит из 3-х последовательных циклов (листинг 3) и вызова сторонней функции. В первом производится поиск корней полинома Лагранжа, посредством вызова соответствующей функции, и запись их в массив “x”. Во втором, происходит заполнение массивов, которые далее будут переданы функции “LinSystemGauss”, для вычисления корней уравнений. А в третьем, происходит вычисление интеграла по формуле .

Листинг 3

public double gfn()

{

double r = 0;

double[] x = new double[N];

for (int p = 0; p < N; p++)

{

x[p] = Ermit(p+1, N);

}

double[,] Xi = new double[N, N];

double[] add = new double[N];

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

{

for (int j = 0; j < N; j++)

{

Xi[i, j] = Math.Pow(x[j], i);

}

add[i] = PowIntegral(i, a, b);

}

double[] c = LinSystemGauss(Xi, add);

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

{

r += c[i] \* function(x[i]);

}

return r;

}

Остальная часть кода содержит в себе всё необходимое для создания формы и вывода результата на форму (Рисунок 1). Полный код можно изучить приложении А.

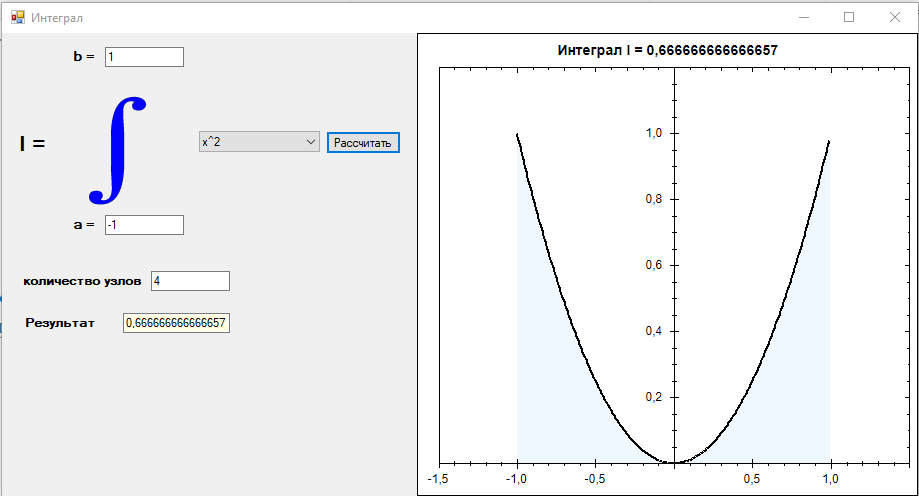


Рисунок 1 – Интерфейс программы.

Основной код, отвечающий за вывод графика и расчёт интеграла (здесь вернее сказать – за вызов функции, рассчитывающей интеграл) и проверку вводимых данных, содержится в обработчике нажатия кнопки (листинг 4).

Листинг 4

private void button1\_Click(object sender, EventArgs e)

{

double res = 0;

zgIntegral.GraphPane.CurveList.Clear();

zgIntegral.GraphPane.XAxis.Title.Text = "x";

zgIntegral.GraphPane.YAxis.Title.Text = "y";

zgIntegral.GraphPane.XAxis.Cross = 0.0;

zgIntegral.GraphPane.YAxis.Cross = 0.0;

zgIntegral.AxisChange();

zgIntegral.Invalidate();

GausMethodIntegral.Function function = fn;

if (cbVariants.Text == "")

return;

if (cbVariants.Text == "x^2")

function = fn;

if (cbVariants.Text == "Sin(x)")

function = fnSin;

if (cbVariants.Text == "x^2 + 2x - 5")

function = fn2;

if (cbVariants.Text == "8x - 3")

function = fn3;

double A, B;

int N;

try

{

A = double.Parse(tbA.Text);

B = double.Parse(tbB.Text);

N = int.Parse(tbN.Text);

}

catch

{

MessageBox.Show("Сам дурак");

return;

}

GausMethodIntegral.GausMethod gausMethod = new GausMethodIntegral.GausMethod();

res = Math.Abs(gausMethod.Solve(function, A, B, N));

PointPairList list = new PointPairList();

for(double x = A; x <= B; x += 0.01)

{

list.Add(x, function(x));

}

LineItem line2 = zgIntegral.GraphPane.AddCurve("", list, Color.Black, SymbolType.None);

line2.Line.Width = 2;

line2.Line.Fill = new Fill(Color.AliceBlue);

zgIntegral.RestoreScale(zgIntegral.GraphPane);

zgIntegral.GraphPane.XAxis.Title.Text = "x";

zgIntegral.GraphPane.YAxis.Title.Text = "y";

zgIntegral.GraphPane.XAxis.Cross = 0.0;

zgIntegral.GraphPane.YAxis.Cross = 0.0;

zgIntegral.GraphPane.XAxis.Title.IsVisible = false;

zgIntegral.GraphPane.YAxis.Title.IsVisible = false;

zgIntegral.GraphPane.YAxis.Scale.IsSkipLastLabel = true;

zgIntegral.GraphPane.YAxis.Scale.IsSkipCrossLabel = true;

zgIntegral.GraphPane.XAxis.Scale.IsSkipLastLabel = true;

zgIntegral.GraphPane.XAxis.Scale.IsSkipCrossLabel = true;

zgIntegral.GraphPane.Title.Text = "Интеграл I = " + res.ToString();

zgIntegral.AxisChange();

zgIntegral.Invalidate();

tbRes.Text = res.ToString();

}**РЕЗУЛЬТАТЫ**

В результате работы сделана программа по вычислению определённых интегралов квадратурой Гаусса методом Гаусса-Эрмитта, и выводящая на экран график функции с вычисленным интегралом.

# ПРИЛОЖЕНИЕ А. Исходный код программы.

using System;

using System.Drawing;

using System.Windows.Forms;

using ZedGraph;

namespace Integral

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

zgIntegral.GraphPane.CurveList.Clear();

zgIntegral.GraphPane.XAxis.Title.Text = "x";

zgIntegral.GraphPane.YAxis.Title.Text = "y";

zgIntegral.GraphPane.XAxis.Cross = 0.0;

zgIntegral.GraphPane.YAxis.Cross = 0.0;

zgIntegral.AxisChange();

zgIntegral.Invalidate();

zgIntegral.GraphPane.Title.Text = "Интеграл I = ";

}

static double fn3(double x)

{

return 8 \* x - 3;

}

static double fn2(double x)

{

return x \* x + 2 \* x - 5;

}

static double fnSin(double x)

{

return Math.Sin(x);

}

static double fn(double x)

{

return x \* x;

}

// public delegate double Func(double x);

private void button1\_Click(object sender, EventArgs e)

{

double res = 0;

zgIntegral.GraphPane.CurveList.Clear();

zgIntegral.GraphPane.XAxis.Title.Text = "x";

zgIntegral.GraphPane.YAxis.Title.Text = "y";

zgIntegral.GraphPane.XAxis.Cross = 0.0;

zgIntegral.GraphPane.YAxis.Cross = 0.0;

zgIntegral.AxisChange();

zgIntegral.Invalidate();

GausMethodIntegral.Function function = fn;

if (cbVariants.Text == "")

return;

if (cbVariants.Text == "x^2")

function = fn;

if (cbVariants.Text == "Sin(x)")

function = fnSin;

if (cbVariants.Text == "x^2 + 2x - 5")

function = fn2;

if (cbVariants.Text == "8x - 3")

function = fn3;

double A, B;

int N;

try

{

A = double.Parse(tbA.Text);

B = double.Parse(tbB.Text);

N = int.Parse(tbN.Text);

}

catch

{

MessageBox.Show("Сам дурак");

return;

}

GausMethodIntegral.GausMethod gausMethod = new GausMethodIntegral.GausMethod();

if(rbErmit.Checked)

res = Math.Abs(gausMethod.Solve(function, A, B, N));

if(rbLagrange.Checked)

res = Math.Abs(gausMethod.Solve(function, -1, 1, N));

PointPairList list = new PointPairList();

for(double x = A; x <= B; x += 0.01)

{

list.Add(x, function(x));

}

LineItem line2 = zgIntegral.GraphPane.AddCurve("", list, Color.Black, SymbolType.None);

line2.Line.Width = 2;

line2.Line.Fill = new Fill(Color.AliceBlue);

zgIntegral.RestoreScale(zgIntegral.GraphPane);

zgIntegral.GraphPane.XAxis.Title.Text = "x";

zgIntegral.GraphPane.YAxis.Title.Text = "y";

zgIntegral.GraphPane.XAxis.Cross = 0.0;

zgIntegral.GraphPane.YAxis.Cross = 0.0;

zgIntegral.GraphPane.XAxis.Title.IsVisible = false;

zgIntegral.GraphPane.YAxis.Title.IsVisible = false;

zgIntegral.GraphPane.YAxis.Scale.IsSkipLastLabel = true;

zgIntegral.GraphPane.YAxis.Scale.IsSkipCrossLabel = true;

zgIntegral.GraphPane.XAxis.Scale.IsSkipLastLabel = true;

zgIntegral.GraphPane.XAxis.Scale.IsSkipCrossLabel = true;

zgIntegral.GraphPane.Title.Text = "Интеграл I = " + res.ToString();

zgIntegral.AxisChange();

zgIntegral.Invalidate();

tbRes.Text = res.ToString();

}

}

}

namespace GausMethodIntegral

{

public delegate double Function(double x);

// Класс вспомогательных математических процедур и функций

abstract public class Methods

{

// Вычисление системы линейных уравнений методом Гаусса

public static double[] LinSystemGauss(double[/\*строка\*/,/\*столбец\*/] system, double[] add)

{

int N = system.GetLength(0);

for (int i = 0; i < N; i++) //строки

{

for (int j = 0; j < i; j++) //столбцы

{

double a = -1;

int k;

for (k = i - 1; k >= 0; k--)

{

if (system[k, j] != 0)

{

a = system[i, j] / system[k, j];

break;

}

}

for (int c = 0; c < N; c++)

{

system[i, c] -= system[k, c] \* a;

}

add[i] -= add[k] \* a;

}

}

double[] res = new double[N];

for (int i = N - 1; i >= 0; i--)

{

double a = add[i];

for (int j = N - 1; j > i; j--)

{

a -= system[i, j] \* res[j];

}

res[i] = a / system[i, i];

}

return res;

}

// Нахождение определённого интеграла целой рациональной функции n-й степени

protected double PowIntegral(int pow, double a, double b)

{

return Math.Pow(b, pow + 1) / (pow + 1) - Math.Pow(a, pow + 1) / (pow + 1);

}

// Вычисление факториала

private int factorial(int n)

{

int a = 1;

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

a \*= i;

return a;

}

// Полином Лежандра

private double Legandr(double x, double[] func)

{

double ret = 0;

for (int i = 0; i < func.Length; i++)

{

ret += Math.Pow(x, i) \* func[i];

}

return ret;

}

// Поиск корней полинома Лежандра

protected double Gauss\_Legandr(int i, int n, double eps = 0.01f, double e = 0.001f)

{

int cnt;

cnt = (int)Math.Pow(2, n);

double[] a = new double[cnt + 1];

// Бином Ньютона

for (int k = 0; k <= cnt; k++)

{

double t = factorial(n) / (factorial(k) \* factorial(n - k)) \* Math.Pow(-1, k);

a[2 \* (n - k)] = t;

}

for (int j = 0; j < n; j++)

{

for (int c = 0; c <= cnt; c++)

{

if (c == 0)

{

a[c] = 0;

continue;

}

if (a[c] == 0)

continue;

a[c - 1] = a[c] \* c;

a[c] = 0;

}

}

//double C = 1 / (Math.Pow(2, n) \* factorial(n));

// Поиск корней

//поиск отрезков

double max = a[0];

int mp = 0;

for (int c = 1; c <= cnt; c++)

{

if (a[c] > max)

max = a[c];

if (a[c] != 0)

mp = c;

}

double bx = 1 + max / a[mp];

double ax = -bx;

int number = 1;

for (double X = ax; X <= bx; X += e)

{

if ((Math.Abs(Legandr(X, a)) <= eps) && (number == i))

return X;

number++;

}

return -1;

}

/\* protected double Gauss\_Chebyshev(double u)

{

return 1.0 / (Math.Sqrt(1.0 - Math.Pow(u, 2)));

}

protected double Gauss\_Lagerr(double u)

{

return Math.Exp(-u);

}\*/

// Поиск корней полинома Эрмитта

protected double Ermit(int i, int n)

{

double[,] poly = new double[n / 2 + 1, 2];

for (int j = 0; j <= n / 2; j++)

{

poly[j, 0] = Math.Pow(-1, j) \* (factorial(n) \* Math.Pow(2, n - 2 \* j)) / (factorial(j) \* factorial(n - 2 \* j));

poly[j, 1] = n - 2 \* j;

}

//поиск отрезков

double max = poly[1, 0];

int mp = (int)poly[0, 1];

for (int j = 1; j <= n / 2; j++)

{

if (Math.Abs(poly[j, 0]) > Math.Abs(max))

max = poly[j, 0];

if (mp < (int)poly[j, 1])

mp = (int)poly[j, 1];

}

double bx = 1 + Math.Abs(max) / Math.Abs(poly[0, 0]);

double eps = 0.001f; // допустимая погрешность вычислений

double e = 0.001f; // шаг итерации

double ax = -bx;

double[] res = new double[mp];

double FuncX = 0;

for (int I = 0; I < mp; I++)

res[I] = ax;

for (double X = ax; X <= bx; X += e)

{

FuncX = 0;

for (int I = 0; I < n / 2 + 1; I++)

{

FuncX += poly[I, 0] \* Math.Pow(X, poly[I, 1]);

}

int k = 0;//max element

for (int I = 1; I < mp; I++)

{

double fk = 0;

double ik = 0;

for (int J = 0; J < n / 2 + 1; J++)

{

fk += poly[J, 0] \* Math.Pow(res[k], poly[J, 1]);

ik += poly[J, 0] \* Math.Pow(res[I], poly[J, 1]);

}

if (Math.Abs(fk) < Math.Abs(ik))

{

k = I;

}

}

double rk = 0;

for (int J = 0; J < n / 2 + 1; J++)

{

rk += poly[J, 0] \* Math.Pow(res[k], poly[J, 1]);

}

if ((Math.Abs(FuncX) < Math.Abs(rk)) && (FuncX) < eps)

res[k] = X;

}

return res[i - 1];

}

}

// Класс для вычислений определённых интегралов квадратурой Гаусса

public class GausMethod : Methods

{

public static int N; // Порядок вычисления

public double a, b; // Границы интегрирования

public Function function; // Ссылка на функцию

// Ввод значений, проверка, выдача решения

public double Solve(Function f, double A, double B, int n)

{

N = n;

a = A;

b = B;

function = f;

if (A >= B || n < 2)

return 0;

else

return gfn();

}

public double SolveLagrange(Function f, int n)

{

N = n;

a = -1;

b = 1;

function = f;

if (n < 2)

return 0;

else

return glfn();

}

// Вычисление определённых интегралов квадратурой Гаусса методом Гаусса-Эрмитта

private double gfn()

{

double r = 0;

double[] x = new double[N];

for (int p = 0; p < N; p++)

{

x[p] = Ermit(p + 1, N);

}

double[,] Xi = new double[N, N];

double[] add = new double[N];

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

{

for (int j = 0; j < N; j++)

{

Xi[i, j] = Math.Pow(x[j], i);

}

add[i] = PowIntegral(i, a, b);

}

double[] c = LinSystemGauss(Xi, add);

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

{

r += c[i] \* function(x[i]);

}

return r;

}

// Вычисление определённых интегралов квадратурой Гаусса методом Гаусса-Лагранжа

private double glfn()

{

double r = 0;

double[] x = new double[N];

for (int p = 0; p < N; p++)

{

x[p] = Gauss\_Legandr(p + 1, N);

}

double[,] Xi = new double[N, N];

double[] add = new double[N];

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

{

for (int j = 0; j < N; j++)

{

Xi[i, j] = Math.Pow(x[j], i);

}

add[i] = PowIntegral(i, a, b);

}

double[] c = LinSystemGauss(Xi, add);

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

{

r += c[i] \* function(x[i]);

}

return r;

}

}

}