*Санкт-Петербургский Национальный Исследовательский Университет*

*Информационных Технологий, Механики и Оптики*

ФКТиУ, кафедра Вычислительной Техники

Лабораторная работа №2

по дисциплине

«Вычислительная математика»

*«Интегрирование методом Симпсона»*

Санкт-Петербург

1. г.
2. Метод парабол (Симпсона). Идея метода исходит из того, что на частичном промежутке дуга некоторой параболы в общем случае теснее прилегает кривой *y = f(x),* чем хорда, соединяющая концы дуги этой кривой (метод трапеций). Поэтому значения площадей соответствующих элементарных трапеций, ограниченных сверху дугами парабол, являются более близкими к значениям площадей соответствующих частичных криволинейных трапеций, ограниченных сверху дугой кривой *y = f(x),* чем значения площадей соответствующих прямолинейных трапеций.

*O*

*x*

*y*

*y = f(x)*

*a*

*A*

*p*

*b*

*c*

*B*

*q*

*P*

*C*

*Q*

*h1*

*h1*

Рис. Геометрическая интерпретация метода парабол.

Рассмотрим функцию *y = f(x).* Будем считать, что на отрезке [*a; b*] она положительна и непрерывна. Найдем площадь криволинейной трапеции aABb (рис.).

Для этого разделим отрезок [*a, b*] точкой *c = * пополам и в точке *C*(*c*, *f(c)*) проведем касательную к линии *y = f(x).* После этого разделим [*a, b*] точками *p* и *q* на три равные части и проведем через них прямые *x = p* и *x = q*. Пусть *P* и *Q* – точки пересечения этих прямых с касательной. Соединив *A* с *P* и *B* с *Q*, получим три прямолинейные трапеции *aAPp, pPQq, qQBb*. Тогда площадь трапеции *aABb* можно приближенно посчитать по следующей формуле

,где .

Откуда получаем

.

Заметим, что *aA = f(a)*, *bB = f(b)*,а *pP + qQ = 2f(c)* (как средняя линия трапеции), в итоге получаем малую формулу Симпсона



В данном случае дуга *ACB* заменяется параболой, проходящей через точки *A, P, Q, B.*

Малая формула Симпсона дает интеграл с хорошей точностью, когда график подынтегральной функции мало изогнут, в случаях же, когда дана более сложная функция, малая формула Симпсона непригодна. Тогда, чтобы посчитать интеграл заданной функции нужно разбить отрезок [*a, b*] на *n* частей и к каждому из отрезков применить формулу (8).

Обязательным требованием, вытекающим из геометрического смысла метода парабол, является то, что *n* должно быть четным. Пусть , точки деления будут *х0=а, x1, x2, …xn-2, xn-1, xn=b,* а *y0, y1, …yn* – соответствующие значения подынтегральной функции на отрезке [*a, b*]*.*

Тогда, применяя малую формулу Симпсона к каждой паре получившихся отрезков, имеем



Тогда . (9)

Заметим, что во всех выражениях  первый множитель равен :



Сделав замену по формулам (10), вынося общий множитель  за скобку, в (9) получаем:



группируем слагаемые

.

Таким образом, получаем «большую» формулу Симпсона, которая имеет вид:

 (11)

Предлагаем для запоминания следующий вид формулы:

, где *Yкр = y0 + yn,* *Yнеч = y1 + y3 + … + yn-1,*

*Yчет = y2 + y4 + … + yn-2,* а .

1. Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Sympson

{

class Program

{

static void Main(string[] args)

{

int func; //номер функции

int low, high; //пределы

double ac; //точность

func = Number();

low = Limit("нижний"); // проверка ввода

high = Limit("верхний"); // проверка ввода

ac = Accuracy(); // проверка ввода

Calculate cal = new Calculate(func, low, high, ac);

cal.Sympson\_sMethod(); // вызов численного метода

cal.Print(); // печать результатов

}

static public double Accuracy()

{

string s;

double ac;

bool mistake = false;

do

{

Console.WriteLine("Введите точность: ");

s = Console.ReadLine();

if (!double.TryParse(s, out ac) || ac <= 0) // проверка корректности введенных данных

{

mistake = true;

Console.WriteLine("Некорректный ввод. Попробуйте снова");

}

else mistake = false;

}

while (mistake);

return ac;

}

static public int Limit(string text)

{

string s;

int lim;

bool mistake = false;

do

{

Console.WriteLine("Введите " + text + " предел интегрирования:");

s = Console.ReadLine();

if (!int.TryParse(s, out lim))

{

mistake = true;

Console.WriteLine("Некорректный ввод. Попробуйте снова");

}

else mistake = false;

}

while (mistake);

return lim;

}

static public int Number()

{

string s;

int num;

bool mistake = false;

do

{

Console.WriteLine("Выберите функцию, интеграл которой вы бы хотели вычислить: \n 1. 2\*x^2 + 3\*x - 1 \n 2. 1 / (1 + x) \n 3. sqrt(1 + x ^2)");

s = Console.ReadLine();

if (!int.TryParse(s, out num) || num < 1 || num > 3)

{

mistake = true;

Console.WriteLine("Некорректный ввод. Попробуйте снова");

}

else mistake = false;

}

while (mistake);

return num;

}

}

class Calculate

{

double num, low, high, p; // номер функции, нижний и верхний пределы, количество шагов

double In, I2n, h, ac, uac, rez; // интеграл 1, интеграл 2, размер шага,

// введенная пользователем точность, вычисленная погрешность, итоговое значение интеграла

bool flag;

public Calculate(int f, double low, double high, double ac) // конструктор

{

this.num = f;

if (high > low) //меняем местами пределы, если нижний предел > верхнего

{

this.low = low;

this.high = high;

}

else

{

flag = true;

this.low = high;

this.high = low;

}

this.ac = ac;

}

public double Sympson\_sMethod()

{

if (high != low)

{

for (int n = 4; n <= 10000; n += 2)

{

double sum1 = 0, sum2 = 0;

h = (high - low) / n; //вычисление размера шага

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

{

sum1 += 4 \* Point(low + i \* h);

++i;

sum1 += 2 \* Point(low + i \* h);

}

In = (sum1 + Point(low) - Point(high)) \* h / 3; // вычисление интеграла с количеством шагов = n

h = (high - low) / (2 \* n); // увеличение количества шагов в 2 раза

for (int i = 1; i < 2 \* n; i++)

{

sum2 += 4 \* Point(low + i \* h);

++i;

sum2 += 2 \* Point(low + i \* h);

}

I2n = (sum2 + Point(low) - Point(high)) \* h / 3;// вычисление интеграла с количеством шагов = 2n

if ((Math.Abs(I2n - In)/15) < ac) //проверка, заданная пользователем точность меньше вычисленной погрешности?

{

rez = I2n;

p = n;

uac = Math.Abs(I2n - In) / 15;

break;

}

if (n == 10000) { Console.WriteLine("Заданная точность не достигнута. Интеграл не имеет решения."); p = 0; }

}

}

else { rez = 0; p = 0; Console.WriteLine("Пределы интегрирования равны, результат вычисления будет равен 0 в любом случае."); }

// учитывание условия равенства нижнего и верхнего пределов

return rez;

}

public double Point(double x) // вычисление значения выбранной пользователем функции в точке х

{

if (num == 1) return 2 \* Math.Pow(x, 2) + 3 \* x - 1; // 1

if (num == 2) return 1 / (1 + x); // 2

else return Math.Sqrt(1 + Math.Pow(x, 2)); // 3

}

public void Print() // вывод полученных значений

{

if (p != 0)

{

if (flag == true) rez = rez \* (-1);

Console.WriteLine("Значение полученного интеграла: " + Math.Round(rez, 9));

Console.WriteLine("Количество шагов: " + p);

Console.WriteLine("Погрешность: " + Math.Round(uac, 9));

}

Console.ReadKey();

}

}

}

1. Блок – схема программы

Конец

Вывод данных в консоль

Да

n = n+2

Выведение информации об ошибках

Да

Нет

Верно ли введены данные?

Ввод данных с клавиатуры

Начало

Нет

Достигнута необходимая точность?

Вычисление погрешности методом Рунге

Integral = (sum+F(a)-F(b))/3

Нет

i<n

i=1

sum += 4\*F(a+h)

i=i+1

sum += 2\*F(a+h)

i=i+1

Вычисление размера шага h=(a-b)/n

H=