**ЛАБОРАТОРНА РОБОТА № 4**

**Методи чисельного інтегрування**

***Мета:*** вивчення найбільш простих квадратних формул (прямокутників, трапецій, Сімпсона) та методу Монте-Карло

**Зміст роботи**

Завдання 1. Було вивчено методи розрахунку площ фігур

Завдання 2. Було розроблено програму обчислення визначеного інтегралу по різним формулам та методом Монте-Карло.

Завдання 3. Було обчислено визначений інтеграл виразу за різних значень кількості прямокутників/трапецій/дуг та десятків точок у методі Монте-Карло.

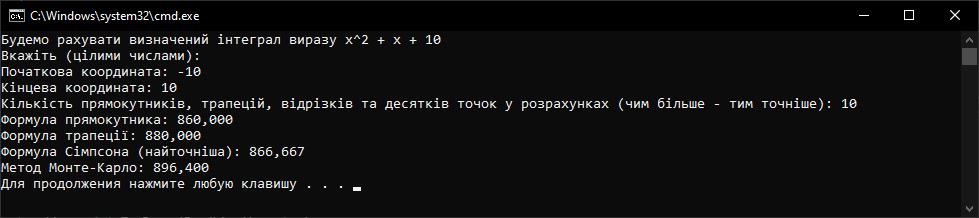


Рис.4.1 – 10 відрізків та десятків точок

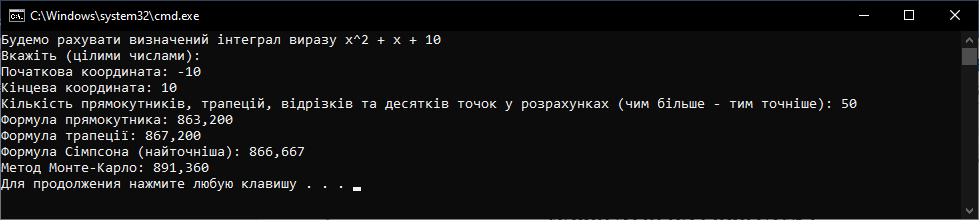


Рис.4.2 – 50 відрізків та десятків точок

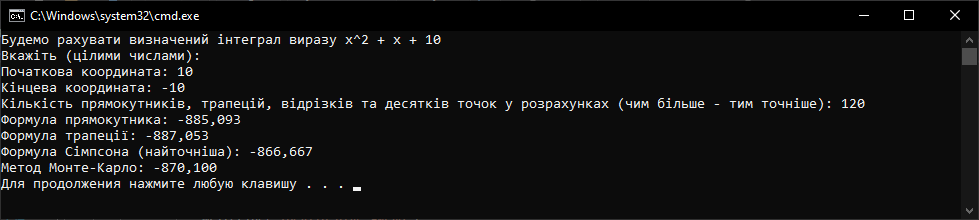


Рис.4.3 – 120 відрізків та десятків точок та перевірка проміжку інтегрування



Рис.4.4 – обчислення інтегралу у мобільному додатку

Лістинг програми:

using System;

using System.Text;

using static System.Console;

using static System.Math;

namespace Formulas

{

class Program

{

static double Rectangle(int start, int end, int n)

{

double symbol; //знак інтегралу

if(start > end)

{

int tmp = start;

start = end;

end = tmp;

symbol = -1; //якщо початкова точка менша за кінцеву, то їх змінюють місцями, а інтегралу змінюють знак

}

else

{

symbol = 1;

}

double step = (end - start) / (double)n; //відстань кроку, з яким йдемо по інтегралу, та відстань однієї зі сторін прямокутника

double result = 0;

for (double i = start; i < end; i += step)

{

result += (step \* (Pow(i, 2) + i + 10)); //обчислення площі прямокутника, де крок - одна сторона, вираз - друга. результат додається до підсумку

}

return result \*= symbol; //множенням на знакову змінну дає точну відповідь

}

static double Trapeze(int start, int end, int n)

{

double symbol;

if (start > end)

{

int tmp = start;

start = end;

end = tmp;

symbol = -1;

}

else

{

symbol = 1;

}

double step = (end - start) / (double)n;

double result = 0;

for (double i = start; i < end; i += step)

{

double first = Pow(i, 2) + i + 10; //одна сторона трапеції

double second = Pow(i + step, 2) + (i + step) + 10; //друга сторона трапеції

result += ((first + second) / 2) \* step; //її площа, що додається до підсумку

}

return result \* symbol;

}

static double SimpsonMethod(int start, int end, int n)

{

double symbol;

if (start > end)

{

int tmp = start;

start = end;

end = tmp;

symbol = -1;

}

else

{

symbol = 1;

}

double step = (end - start) / (double)n;

double E1 = 0;

for (double i = start + step; i < end; i += 2 \* step)

{

E1 += Pow(i, 2) + i + 10; //розрахунок першої суми згідно формули

}

double E2 = 0;

for(double i = start + 2 \* step; i < end; i += 2 \* step)

{

E2 += Pow(i, 2) + i + 10; //розрахунок другої суми згідно формули

}

return ( ( (Pow(start, 2) + start + 10) + (Pow(end, 2) + end + 10) + 4 \* E1 + 2 \* E2 ) \* step / 3 ) \* symbol;

//підсумок згідно формули

}

static double MethodMonteCarlo(int start, int end, int n)

{

double symbol;

if (start > end)

{

int tmp = start;

start = end;

end = tmp;

symbol = -1;

}

else

{

symbol = 1;

}

double height = double.MinValue; //висота прямокутника, у якому "розсипаємо" крапки

for(double i = start; i < end; i += 0.001)

{

if(Pow(i, 2) + i + 10 > height)

{

height = Pow(i, 2) + i + 10; //визначення найбільшої точки майбутнього прямокутника

}

}

Random random = new Random();

int length = (end - start); //відстань між краями інтегралу

int N = length \* n \* 10; //кількість точок, які будуть "розсипані"

int num = 0; //майбутня кількість точок, що потрапили у зону нашого інтегралу

double area = height \* length; //площа прямокутника

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

{

double x = random.NextDouble() \* (length) + start; //випадкова координата х

double y = random.NextDouble() \* height; //випадкова координата у

if (y <= Pow(x, 2) + x + 10) //перевірка на потрапляння у потрібну зону інтегралу

{

num++; //збільшення кількості точок фактично відповідних умовам

}

}

return area / N \* num \* symbol; //власне розрахунок інтегралу: площа прямокутника, ділена на кількість точок взагалі, дає "площу" однієї

//цю "площу" точки множимо на кількість точок, що потрапили у зону інтегралу, отримуючи приблизну площу-інтеграл

}

static void Main(string[] args)

{

InputEncoding = Encoding.Unicode;

OutputEncoding = Encoding.Unicode;

Write("Будемо рахувати визначений інтеграл виразу x^2 + x + 10\nВкажіть (цілими числами):\n");

bool flag;

int start, end;

Write("Початкова координата: ");

do

{

flag = int.TryParse(ReadLine(), out start);

if (!flag)

{

WriteLine("Уведіть ціле число");

}

}

while (!flag);

Write("Кінцева координата: ");

do

{

flag = int.TryParse(ReadLine(), out end);

if (!flag)

{

WriteLine("Уведіть ціле число");

}

}

while (!flag);

int n;

Write("Кількість прямокутників, трапецій, відрізків та десятків точок у розрахунках (чим більше - тим точніше): ");

do

{

flag = int.TryParse(ReadLine(), out n);

if (!flag)

{

WriteLine("Уведіть ціле число");

}

}

while (!flag);

WriteLine($"Формула прямокутника: {Rectangle(start, end, n):F3}");

WriteLine($"Формула трапеції: {Trapeze(start, end, n):F3}");

WriteLine($"Формула Сімпсона (найточніша): {SimpsonMethod(start, end, n):F3}");

WriteLine($"Метод Монте-Карло: {MethodMonteCarlo(start, end, n):F3}");

}

}

}

***Висновок:*** серед методів інтегрування найкращим виявився метод за формулою Сімпсона, а точність обчислень, як і вважалося, пропорційна кількості частин, на які ділиться інтеграл, та точок, які випадковим чином «розкидують» в області обчислення інтегралу.