Національний авіаційний університет
Факультет кібербезпеки, комп’ютерної та програмної інженерії

ЗВІТ
по лабораторній роботі No 5
Чисельне інтегрування

Дисципліна: «Обчислювальні методи»

Кафедра: прикладної математики
ОС: бакалавр
Спеціальність: 113 «Прикладна математика»
ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 3 курсу. 351 групи
Архіпов Олексій Тімурович

Перевірив: Віталій Богданович Василик

Київ 2022

Тема: Чисельне інтегрування.

Мета роботи: Опанувати метод чисельного інтегрування.

Завдання:

1. Запрограмувати квадратурну формулу Сімпсона. Вхідні параметри: відрізок [a ; b], підінтегральна функція f(x), кількість точок квадратурної формули. Вихідні параметри: значення інтегралу, крок (h).

2. Запрограмувати квадратурну формулу Сімпсона з автоматичним контролем точності за принципом Рунге. Вхідні параметри: відрізок [a ; b], підінтегральна функція f(x), точність (ε). Вихідні параметри: значення інтегралу, крок (h).

Порядок обчислень

Задача чисельного інтегрування полягає в обчисленні визначеного інтеграла

$$I = \int_a^b f(x)dx$$

у випадках, коли аналітичне обчислення неможливе або дуже складне. Методи чисельного обчислення інтеграла засновані на тому, що в якості наближеного значення інтеграла береться значення інтеграла від інтерполюючої для f(x) функції, побудованої по точках розбиття відрізка [a,b].

Метод Сімпсона

Відрізок [a,b] розбивається на n частин з кроком $h = \frac{b-a}{n}$, при цьому точки розбиття x_i визначаються за формулою x_i = a + i × h, i = 0,n, тобто x_0 = a, x_n = b.

Формула Сімпсона в загальному випадку має вигляд

$$S = \frac{h}{3}(f(a) + f(b) + 4\sum_{i=1}^{n-1} f(x_i) + 2\sum_{j=1}^{n-2} f(x_j))$$

Правило Рунґе

У кожній конкретній задачі необхідно визначити число точок поділу n, необхідне обчислення інтеграла з необхідною точністю ε.

Для визначення n зручно наступне правило Рунге. Нехай ε – задана точність обчислення інтеграла, тоді крок h має задовольняти умову $h = \sqrt[4]{\varepsilon}$.

За цим значенням h із співвідношення $h = \frac{b-a}{n}$ визначається n. При цьому для методу Сімпсона як n береться найближче парне ціле число, що перевищує $\frac{b-a}{h}$.

Використовуючи правило Рунґе, можна побудувати процедуру наближеного обчислення інтеграла із заданою точністю ε. Потрібно, почавши обчислення з деякого значення кроку h, послідовно зменшувати це значення вдвічі, щоразу обчислюючи наближене значення I_{h_i}. Обчислення припиняються тоді, коли результати двох наступних обчислень відрізнятимуться менше, ніж ε.

Код програми

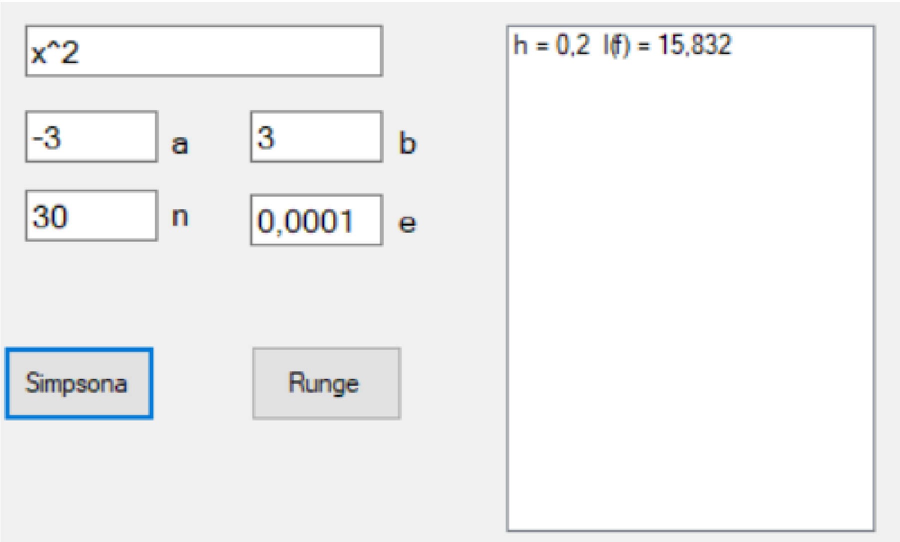
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Net.Mime.MediaTypeNames;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace lab3._5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        string s;
        double a, b, n, e;

        private void btnRunge_Click(object sender, EventArgs e)
        {
            Runge();
        }

        private void btnSimpsona_Click(object sender, EventArgs e)
        {
            Simpsona();
        }

        void InitExpr()
        {
            s = tbExp.Text;
            a = Convert.ToDouble(tbA.Text);
            b = Convert.ToDouble(tbB.Text);
            n = Convert.ToDouble(tbN.Text);
            e = Convert.ToDouble(tbE.Text);
        }
        void Simpsona()
        {
            InitExpr();
            listBox1.Items.Clear();
            int n1 = Convert.ToInt32(n);
            double h = (b - a) / n;
            double[] X = new double[n1];
            double[] Y = new double[n1];
            MathExpression f = new MathExpression(s);
            X[0] = a;
            Y[0] = f.Calculate(X[0]);
            for (int i = 1; i < n1; i++)
            {
                X[i] = X[i - 1] + h;
                Y[i] = f.Calculate(X[i]);
            }
```

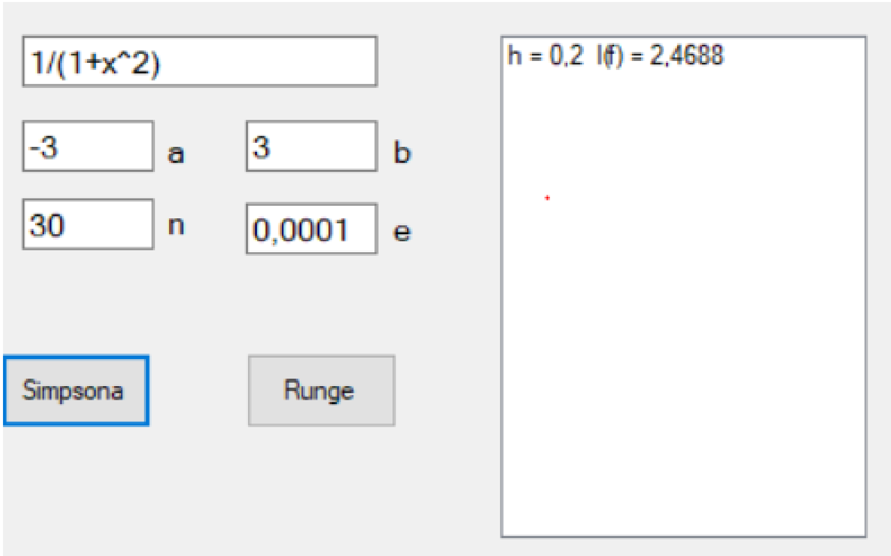
```
            double S1 = Y[0] + Y[n1 - 1];
            double S2 = 0;
            double S3 = 0;
            for (int i = 1; i < n1 - 1; i++)
            {
                if (i % 2 != 0) S2 += Y[i];
                else S3 += Y[i];
            }
            double S = h / 3 * (S1 + 4 * S2 + 2 * S3);
            listBox1.Items.Add("h = " + h.ToString() + "  I(f) = " + Math.Round(S,
4).ToString());
        }
        void Runge()
        {
            InitExpr();
            int z = Convert.ToInt32(numZeroesAfterPoint(e));
            listBox1.Items.Clear();
            List<double> S_all = new List<double>();
            double h1 = Math.Pow(e, 1 / 4);
            int n1 = 0;
            //if (n1 % 2 != 0) n1++;
            double h = 0;
            for (int j = 0; j < 200000; j++)
            {
                if(j == 0)
                {
                    n1 = (int)((b - a) / h1);
                    if (n1 % 2 != 0) n1++;
                    h = (b - a) / n1;
                }
                double[] X = new double[n1];
                double[] Y = new double[n1];
                MathExpression f = new MathExpression(s);
                X[0] = a;
                Y[0] = f.Calculate(X[0]);
                for (int i = 1; i < n1; i++)
                {
                    X[i] = X[i - 1] + h;
                    Y[i] = f.Calculate(X[i]);
                }
                double S1 = Y[0] + Y[n1 - 1];
                double S2 = 0;
                double S3 = 0;
                for (int i = 1; i < n1 - 1; i++)
                {
                    if (i % 2 != 0) S2 += Y[i];
                    else S3 += Y[i];
                }
                double S = h / 3 * (S1 + 4 * S2 + 2 * S3);
                S_all.Add(S);
                if (j >= 1)
                {
                    if (Math.Abs(S_all[j-1] - S_all[j]) / 15 <= e)
                    {
                        listBox1.Items.Add("h = " + h.ToString());
                        listBox1.Items.Add("I(f) = " + Math.Round(S_all[j - 1],
z).ToString());
                        break;
                    }
                    else h = h / 2;
                }
                else h = h / 2; n1 = n1 * 2;
            }
        }
        double numZeroesAfterPoint(double x)
```

```
        { // функція повертає кількість нулів, перед тим як ідуть числа
            if ((x % 1) == 0)
            {
                return 0;
            }
            else
            {
                return -1 - Math.Floor(Math.Log10(x % 1));
            }
        }
    }
}
```

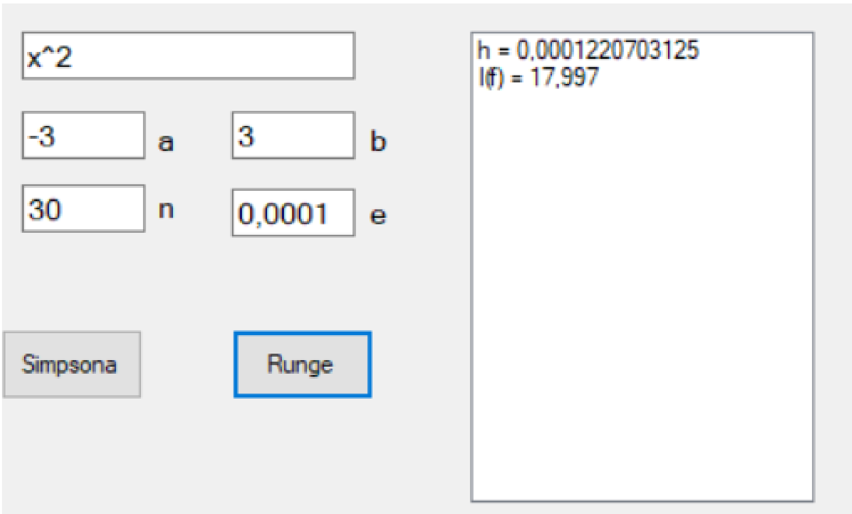
Результати тестування

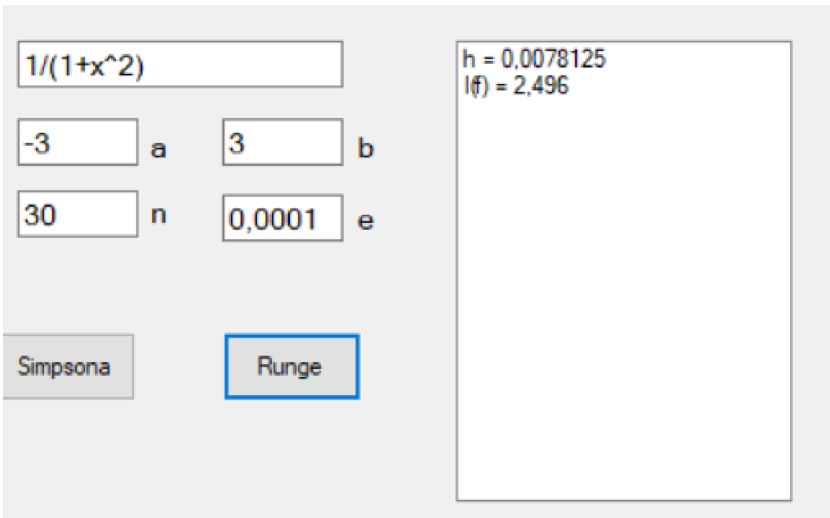
Результат роботи програми за методом Сімпсона.





Результат роботи програми за методом Сімпсона з автоматичним контролем точності за принципом Рунге.





Висновки

1. Я опанував метод чисельного інтегрування.
2. Реалізував квадратурну формулу Сімпсона.
3. Реалізував квадратурну формулу Сімпсона з автоматичним контролем точності за принципом Рунге.