Звіт до лабораторної роботи №1
з курсу “Методи обробка даних та чисельні методи”
студента 2 курсу 1 групи Данька Олександра

Завдання

Розглянемо простір $L_2[a,b]$ зі скалярним добутком $(f(x),g(x))=\frac{1}{b-a}\int_a^b f(x)g(x)dx$ і нормою $\|f(x)-g(x)\|=\sqrt{\frac{1}{b-a}\int_a^b (f(x)-g(x))^2 dx}$. Оберемо в якості базисних функцій на $[-1,1]$ ортогональні поліноми Лежандра, що задаються формулами:

$$P_0(x)=1 \;;\quad P_1(x)=x \;;\quad P_2(x)=\frac{1}{2}(3x^2-1) \;;$$

$$P_{n+1}(x)=\frac{2n+1}{n+1}xP_n(x)-\frac{n}{n+1}P_{n-1}(x) \;.$$

Побудувати на $[-1,1]$ найкраще наближення вигляду $\sum_{j=0}^{3} c_j P_j(x)$ для функції $y=\sin x$ в сенсі заданої норми. Інтеграли рахувати наближено, використовуючи формулу прямокутників або трапецій.

Хід роботи

Щоб побудувати найкраще наближення для функції $y=\sin x$, потрібно щоб $\|y(x)-\tilde{y}(x)\|=min$ або $\|y(x)-\tilde{y}(x)\|^2=min$. Для цього продиференціюємо функцію $\|y(x)-\tilde{y}(x)\|^2$ і знайдемо вирах для коефіцієнтів.

$$\|y(x)-\tilde{y}(x)\|^2=\frac{1}{2}\int_{-1}^{1}\left(y(x)-\sum_{i=0}^{3}c_iP_i(x)\right)^2 dx \;;$$

$$\frac{\partial\sigma^2}{\partial c_j}=-\int_{-1}^{1}\left(y(x)-\sum_{i=0}^{3}c_iP_i(x)\right)P_j(x)dx=-\frac{2}{2}\left[\int_{-1}^{1}y(x)P_j(x)dx-c_j\|P_j\|^2\right]=0 \;.$$

Звідси $c_j=\frac{1}{\|P_j\|^2}(y,P_j)$ або, розписавши скалярний добуток та норму,

$$c_j=\frac{\int_a^b \frac{1}{b-a}\sin(x)P_j(x)dx}{\int_a^b \frac{1}{b-a}P_j^{\,2}(x)dx} \;.$$

Апроксимація ортогональними поліномами має перевагу в тому, що покращення апроксимації шляхом додавання нового члена $c_{j+1}P_{j+1}(x)$ не змінює раніше обчислених коефіцієнтів $c_0,c_1,c_2,\ldots,c_j$.

Всі функції, які безпосередньо стосуються обчислення значення в точці за допомогою наближення, для зручності помістимо в окрему бібліотеку. Такий варіант зручний тим, що цю бібліотеку можна підключити до різних програм, в яких планується використання цих функцій, і не потрібно змінювати код для кожної конкретної програми (наприклад, консольна програма та програма з графічним інтерфейсом).

Для виконання поставленої задачі було створено бібліотеку `LegendreApproximation_lib` до якої увійшли наступні функції:

- `LegendreP(double x, int n0)` — повертає значення поліному Лежандра степеня `n` в точці `x`.
- `WeightF()` — вагова функція (в нашому випадку це коефіцієнт).
- `CoeffNum(int degree)` — обчислення чисельника в формулі для знаходження коефіцієнта.
- `CoeffDen(int degree)` — обчислення знаменника в формулі для знаходження

коефіцієнта.

- `CalcCoeff(int degree)` — обчислення коефіцієнта для формули наближеного обчислення функції.
- `CalcAllCoeffs()` — обчислення всіх коефіцієнтів.
- `FunctionInPoint(double x)` — наближене обчислення функції в точці `x`.

Доступним для зовнішнього використання є лише остання функція, а всі інші є внутрішніми. Таким чином у своїй програмі можна підключити заздалегідь написану бібліотеку і проводити обчислення викликом всього однієї функції.

Код бібліотеки

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LegendreApproximation_lib
{
    public class LegendreApprox
    {
        private int factor = 4000;
        private int n; // кількість поліномів
        private double a, b; // Межі
        private double[] coeffs;
        private Function f; // Делегат для виклику функції

        public LegendreApprox(Function inputFunc, double a_,
        double b_, int n_) // Конструктор класу
        {
            f = inputFunc;
            a = a_;
            b = b_;
            n = n_;
            coeffs = new double[n + 1];
            CalcAllCoeffs();
        }
        private double LegendreP(double x, int n0) // Розрахунок
        значення поліному Лежандра
        {
            if (n0 == 0)
                return 1.0;
            else if (n0 == 1)
                return x;
            else if (n0 == 2)
                return (x * x * 3 - 1) / 2;
            else if (n0 > 2)
            {
                return ((2 * (n0 - 1) + 1) * x * LegendreP(x, n0 -
                1) / n0) - ((n0 - 1) * LegendreP(x, n0 - 2) / n0);
            }
            else
                throw (new System.Exception("The degree of
```

```
            polinominal is out of range.")); // Виключення,
            яке виникає при отриманні недопустимого значення
            степеня полінома
    }
    public delegate double Function(double x); // Делегат для
    функцій, які потрібно апроксимувати
    private double WeightF() // Вагова функція (в нашому
    випадку -- коефіцієнт)
    {
        return 1 / (b - a);
    }
    private double CalcCoeff(int degree) // Розрахунок
    коефіцієнта c
    {
        return CoeffNum(degree) / CoeffDen(degree);
    }
    private void CalcAllCoeffs()
    {
        for (int i = 0; i <= n; i++)
        {
            coeffs[i] = CalcCoeff(i);
        }
    }
    private double CoeffNum(int degree) // Чисельник виразу за
    допомогою якого розраховується коефіцієнт
    {
        double step = (b - a) / factor;
        double result = 0, temp;
        for (int i = 0; i < (factor - 1); i++)
        {
            temp = WeightF() * f(a + i * step) * LegendreP(a +
            i * step, degree);
            temp += WeightF() * f(a + (i + 1) * step) *
            LegendreP(a + (i + 1) * step, degree);
            temp /= 2;
            temp *= (a + i * step) - (a + (i - 1) * step);
            result += temp;
        }
        return result;
    }
    private double CoeffDen(int degree) // Знаменник виразу за
    допомогою якого розраховується коефіцієнт
    {
        double step = (b - a) / factor;
        double result = 0, temp;
        for (int i = 0; i < (factor - 1); i++)
        {
            temp = WeightF() * LegendreP(a + i * step, degree)
            * LegendreP(a + i * step, degree);
            temp += WeightF() * LegendreP(a + (i + 1) * step,
            degree) * LegendreP(a + i * step, degree);
            temp /= 2;
            temp *= (a + i * step) - (a + (i - 1) * step);
```

```
                result += temp;
            }
            return result;
        }
        public double FunctionInPoint(double x)
        {
            double result = 0;
            for (int i = 0; i <= n; i++)
            {
                result += coeffs[i] * LegendreP(x, i);
                //Console.WriteLine(coeffs[i]);
            }
            return result;
        }
        /*public double DiffInPoint(Function f, double x)
        {
            double diff = 1;
            diff = f(a)
        }*/
    }
}
```

Демонстрацію роботи створеної бібліотеки для апроксимації функцій проведемо за допомогою консольної програми.

Код програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using LegendreApproximation_lib;

namespace Legendre
{
    class LegendreApproximation
    {
        static void Main(string[] args)
        {
            LegendreApprox lga = new LegendreApprox(Math.Sin, -1,
            1, 3);
            double step = 0.08, xmin = -1.0, xmax = 1.0, x = xmin;
            int i = 1;
            Console.WriteLine("Point\tx\t\tSin(x)\t\tApprox\t\tDif
            f");
            while(x <= xmax)
            {
                Console.WriteLine("{0}\t{1:0.00}\t\t{2:0.0000}\t\t
                {3:0.0000}\t\t{4:0.0000}", i, x, Math.Sin(x),
                lga.FunctionInPoint(x), Math.Abs(Math.Sin(x) -
                lga.FunctionInPoint(x)));
                i++;
                x += step;
```

```
            }
            Console.ReadKey();
        }
    }

}
```

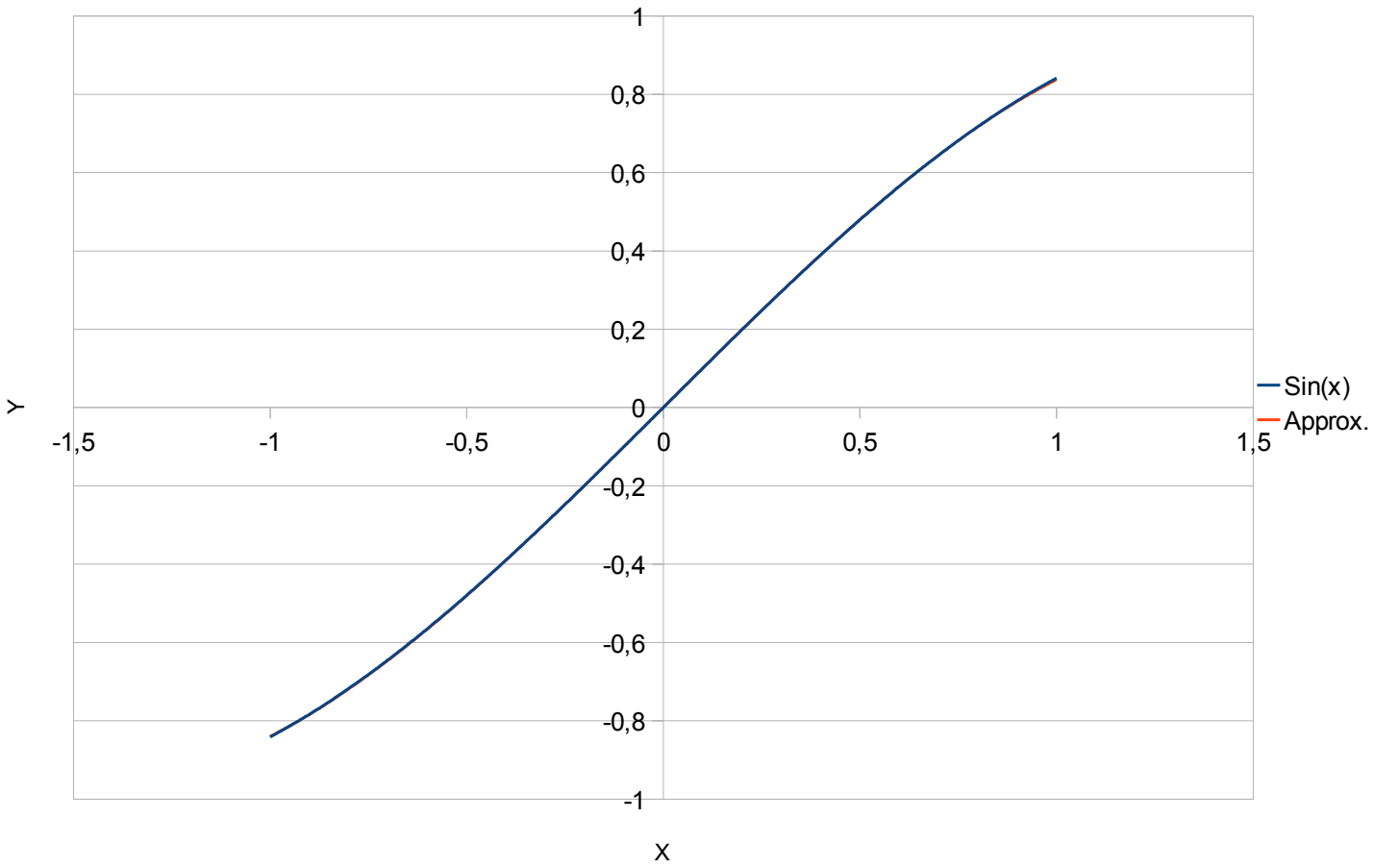
В результаті отримаємо наступні результати:

Point	x	Sin(x)	Approx.	Diff.
1	-1,00	-0,8415	-0,8402	0,0013
2	-0,92	-0,7956	-0,7957	0,0001
3	-0,84	-0,7446	-0,7455	0,0008
4	-0,76	-0,6889	-0,6900	0,0011
5	-0,68	-0,6288	-0,6299	0,0012
6	-0,60	-0,5646	-0,5657	0,0010
7	-0,52	-0,4969	-0,4977	0,0008
8	-0,44	-0,4259	-0,4264	0,0005
9	-0,36	-0,3523	-0,3525	0,0003
10	-0,28	-0,2764	-0,2764	0,0000
11	-0,20	-0,1987	-0,1986	0,0001
12	-0,12	-0,1197	-0,1195	0,0002
13	-0,04	-0,0400	-0,0397	0,0003
14	0,04	0,0400	0,0403	0,0003
15	0,12	0,1197	0,1201	0,0004
16	0,20	0,1987	0,1991	0,0004
17	0,28	0,2764	0,2768	0,0004
18	0,36	0,3523	0,3528	0,0005
19	0,44	0,4259	0,4265	0,0005
20	0,52	0,4969	0,4974	0,0005
21	0,60	0,5646	0,5651	0,0005
22	0,68	0,6288	0,6291	0,0003
23	0,76	0,6889	0,6889	0,0001
24	0,84	0,7446	0,7439	0,0008
25	0,92	0,7956	0,7936	0,0020
26	1,00	0,8415	0,8377	0,0038

В першій колонці знаходиться номер точки, в наступних, відповідно, точка, значення функції, що визначене безпосередньо, наближене значення, різниця між двома попередніми значеннями.

Також в даній програмі можна замінити функцію, яку потрібно апроксимувати, на іншу без зміни коду бібліотеки. Це досягнуто за допомогою використання делегатів.

Графік функцій



Як бачимо на графіку, відхилення функцій на графіку практично непомітне.

Висновок

- Знайдено формулу для визначення коефіцієнтів для підрахунку наближеного значення функції.
- Даний метод дає досить непогані результати. Відхилення від безпосередньо визначеного значення функції спостерігається у третьому знаку після коми.
- Під час написання програми було виявлено, що метод трапецій дає більш точні результати, ніж метод прямокутників за однакових умов.