1. Постановка задачі

Задана таблиця значень функції $f(x) = \ln(x^4 - 2x^2 + 3)$ в 9-ти вузлових точках:

x	-1.6	-1,1	-0,6	-0,2	0,2	0,7	0,93	1,2	1,51
$f(x)$	1.48921	0,71496	0,87946	1,07213	1,07213	0,81541	0,70223	0,78554	1.29161

1. Побудувати таблицю значень функції $f(x)$, $x \in [-1.5, 1.5]$ з кроком $h = 0{,}1$:

а) безпосередньо за формулою;

б) застосовуючи лінійну апроксимацію;

в) за допомогою апроксимації поліномом Лагранжа, побудованим за всіма наявними вузлами.

2. Застосовуючи лінійну апроксимацію функції $f(x)$ та формули чисельного диференціювання, побудувати таблиці значень функції $f'(x)$ та $f''(x)$, $x \in [-1.5, 1.5]$ з кроком $h = 0{,}1$. Порівняти з таблицями значень цих функцій, одержаних безпосереднім диференціюванням.

2. Опис методу розв'язку

1.а) Підстановка у функцію $f(x)$ $n = \frac{b-a}{h} + 1$ аргументів (b – верхня межа області визначення, a – нижня межа області визначення, в даному випадку $a = -1.5;\ b = 1.5; h = 0.1$)

Лістинг 1

```
double[] out_x_array = GenereteXarray(from,h,to); // масив аргументів функції
double[] true_y_array = new double[out_x_array.Length];// масив значень функції, отриманих
підстановкою у формулу

for (int i = 0; i < out_x_array.Length; i++)
{
    true_y_array[i] = Math.Log(Math.Pow(out_x_array[i], 4) - 2 * Math.Pow(out_x_array[i], 2)
+ 3);
}
```

Аргументи функції обчислені методом `GenereteXarray()` за формулою:

$$x_i = x_{i-1} + h, \text{ де } i = \overline{0, n},\ x_0 = a$$

Лістинг 2 – GenerateXarray

```
static double[] GenereteXarray(double from,double h,double to)
{
   int n = (int)((to - from) / h)+1;
   double[] x_arr = new double[n];
   x_arr[0] = from;
   for (int i = 1; i < n; i++)
   {
       x_arr[i] = x_arr[i - 1] + h;
   }
   return x_arr;
}
```

1.б) Лінійна інтерполяція

На основі даних в умові 9-ти вузлових точках шляхом лінійної інтерполяції буде отримано наближені значення функції $f(x)$ за наступною формулою:

$$y = a_i x + b_i \,, x_i \le x \le x_{i+1}, \text{де } a = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, b = y_i - a_i x_i$$

$$i = \overline{0, n-1}$$

Зауваження: в даному випадку $n = 9$ – кількість вузлових точок, а $n - 1$ – це кількість відрізків, якими сполучені ці точки.

Лістинг 3 – лінійна інтерполяція

```
static void LinearInterpolation(double[] in_x_arr, double[] in_y_arr, double[] out_x_arr,
double[] out_y_arr)
{
    double[] a = new double[in_x_arr.Length - 1];
    double[] b = new double[in_x_arr.Length - 1];

    for (int i = 0; i < in_x_arr.Length-1; i++)
    {
        a[i] = (in_y_arr[i + 1] - in_y_arr[i]) / (in_x_arr[i + 1] - in_x_arr[i]);
        b[i] = in_y_arr[i] - a[i] * in_x_arr[i];
    }

    int j = 0;
    for (int i = 0; i < out_x_arr.Length; i++)
    {
        for (j = 0; j < in_x_arr.Length-1; j++)
        {
            if (out_x_arr[i] < in_x_arr[j + 1] && out_x_arr[i] > in_x_arr[j]) break;//
            перевірка чи потрапляє x у необхідний відрізок
        }
        out_y_arr[i] = a[j] * out_x_arr[i] + b[j];
    }
}
```

1.в) Інтерполяційний многочлен Лагранжа

Наближені значення функції будуть отримані за допомогою інтерполяційного многочлена Лагранжа:

$$y = \sum_{i=0}^{n} y_i \frac{(x - x_0)(x - x_1) \dots (x - x_{i-1})(x - x_{i+1}) \dots (x - x_n)}{(x_i - x_0)(x_i - x_1) \dots (x_i - x_{i-1})(x_i - x_{i+1}) \dots (x_i - x_n)}$$

Лістинг 4 – інтерполяція методом Лагранжа

```
static void LagrangeInterpolation(double[] in_x_arr, double[] in_y_arr, double[] out_x_arr,
double[] out_y_arr)
{
    double l=1;
    for (int j = 0; j < out_x_arr.Length; j++)
    {
         out_y_arr[j] = 0;
         for (int i = 0; i < in_y_arr.Length; i++)
             {
              for (int k = 0; k < in_y_arr.Length; k++)
                 {
                       if (k != i)
                       l *= (out_x_arr[j] - in_x_arr[k]) /( in_x_arr[i] - in_x_arr[k]);
                 }

              out_y_arr[j] += in_y_arr[i] *l;
              l = 1;
             }
    }
}
```

2. Чисельне диференціювання

2.а) Для знаходження наближеного значення першої похідної $f'(x)$ функції $f(x)$ шляхом чисельного диференціювання на основі значень, отриманих лінійною апроксимацією $f(x)$ використано формулу центральної різницевої похідної:

$$y_i' = \frac{y_{i+1} - y_{i-1}}{2h}, \text{де } i = \overline{1, n-1}$$

Щоб знайти значення в крайніх точках($i = 0, i = n$) використано формули правої та лівої різницевої похідної:

$$y_0' = \frac{y_1 - y_0}{h},\ y_n' = \frac{y_n - y_{n-1}}{h}$$

Для знаходження наближеного значення другої похідної $f''(x)$ використано наступну формулу:

$$y_i'' = \frac{y_{i+1} + y_{i-1} - 2y_i}{h^2}, \text{де } i = \overline{1, n-1}$$

Лістинг 5 – Чисельне диференціювання

```
static void Differentiate(double[] out_x_arr,double[] in_y_arr,double[]
first_diff_y,double[] secon_diff_y,double h )
{
    //для обчислення похідної в крайніх точках використано формули правої та лівої похідної
    first_diff_y[0] = (in_y_arr[1] - in_y_arr[0]) / h;
    first_diff_y[out_x_arr.Length - 1] = (in_y_arr[out_x_arr.Length - 1] -
    in_y_arr[out_x_arr.Length - 2]) / h;

    for (int i = 1; i < out_x_arr.Length - 1; i++)
    {
        first_diff_y[i] = (in_y_arr[i + 1] - in_y_arr[i - 1]) / (2 * h);
    }
```

```
    //для обчислення другої похідної в крайніх точках використано формули правої та лівої
    похідної та значення першої похідної
    secon_diff_y[0] = (first_diff_y[1] - first_diff_y[0]) / h;
    secon_diff_y[out_x_arr.Length - 1] = (first_diff_y[out_x_arr.Length - 1] -
    first_diff_y[out_x_arr.Length - 2]) / h;

    for (int i = 1; i < out_x_arr.Length - 1; i++)
    {
    secon_diff_y[i] = (in_y_arr[i + 1] + in_y_arr[i - 1] - 2 * in_y_arr[i]) / (h * h);
    }
}
```

2.б) Значення першої та другої похідних $f'(x)$, $f''(x)$ отримані підстановкою (аналогічно до пункту 1.а) значень x у наступні формули:

$$f' = \frac{-4x + 4x^3}{3 - 2x^2 + x^4}$$

$$f'' = -\frac{(-4x + 4x^3)^2}{(3 - 2x^2 + x^4)^2} + \frac{-4 + 12x^2}{3 - 2x^2 + x^4}$$

Лістинг 6 – Значення похідних f(x)

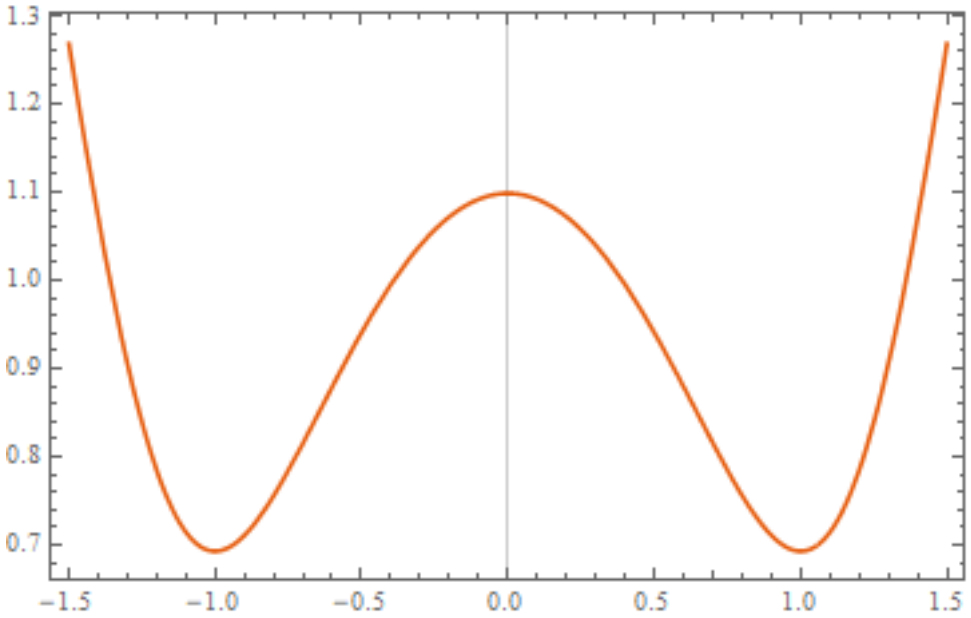
```
double[] true_first_diff_y = new double[out_x_array.Length];//масив істинних значень
похідної функції
for(int i = 0; i < out_x_array.Length; i++)
    {
        true_first_diff_y[i] = (-4 * out_x_array[i] + 4*Math.Pow(out_x_array[i], 3))/
        (Math.Pow(out_x_array[i], 4) - 2 * Math.Pow(out_x_array[i], 2) + 3);
    }

double[] true_second_diff_y = new double[out_x_array.Length];//масив істинних значень
другої похідної функції
for (int i=0;i<out_x_array.Length;i++)
    {
        true_second_diff_y[i] = 4 * (-Math.Pow(out_x_array[i], 6) +
        Math.Pow(out_x_array[i], 4)+ 7*Math.Pow(out_x_array[i], 2)-
        3)/Math.Pow(Math.Pow(out_x_array[i], 4)-2* Math.Pow(out_x_array[i], 2)+3,2);
    }
```

Зауваження: під час лінійної інтерполяції ми наближаємо функцію многочленом першого степеню (в даному випадку з графічної точки зору це деяка ламана), тому якщо до отриманих значень застосувати формули чисельного диференціювання для знаходження першої похідної, отримаємо набір горизонтальних прямих, похідні від яких будуть рівними нулю. Це також випливає з того що похідна другого і вищого порядку від многочлена першого степеню завжди рівна 0. Тому шукати значення другої похідної функції, отримані лінійною інтерполяцією, шляхом чисельного диференціювання недоцільно.

3. Отримані результати

3.1) Значення функції отримані підстановкою у формулу $f(x)$

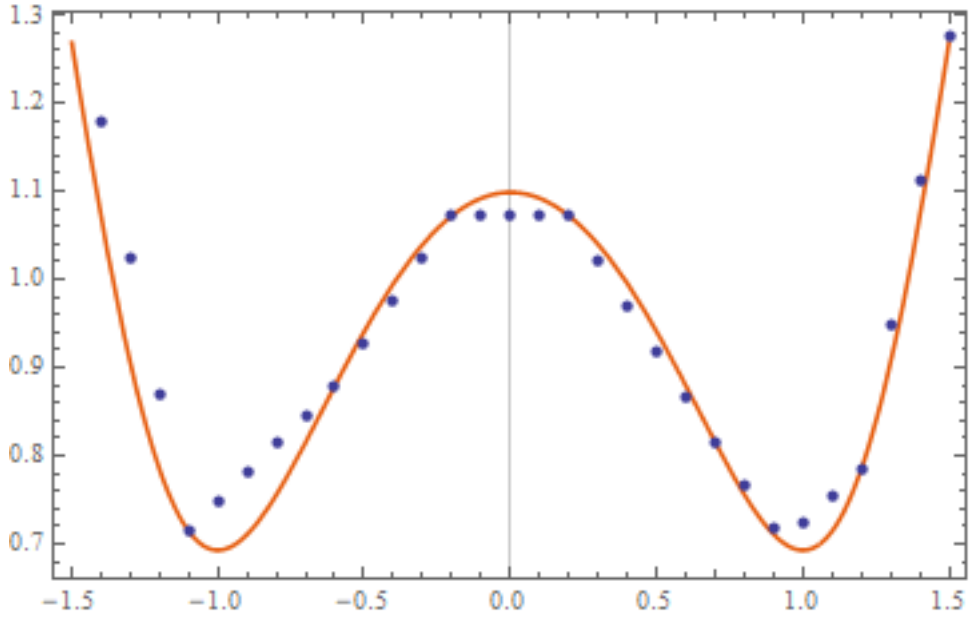
x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	1.27046	1.07213	0.90668	0.78554	0.71496	0.69315	0.71104	0.75593
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.81541	0.87946	0.94098	0.99532	1.03961	1.07213	1.09196	1.09861
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	1.09196	1.07213	1.03961	0.99532	0.94098	0.87946	0.81541	0.75593
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	0.71104	0.69315	0.71496	0.78554	0.90668	1.07213	1.27046	



Графік 1. $f(x)$

3.2) Значення функції отримані лінійною інтерполяцією

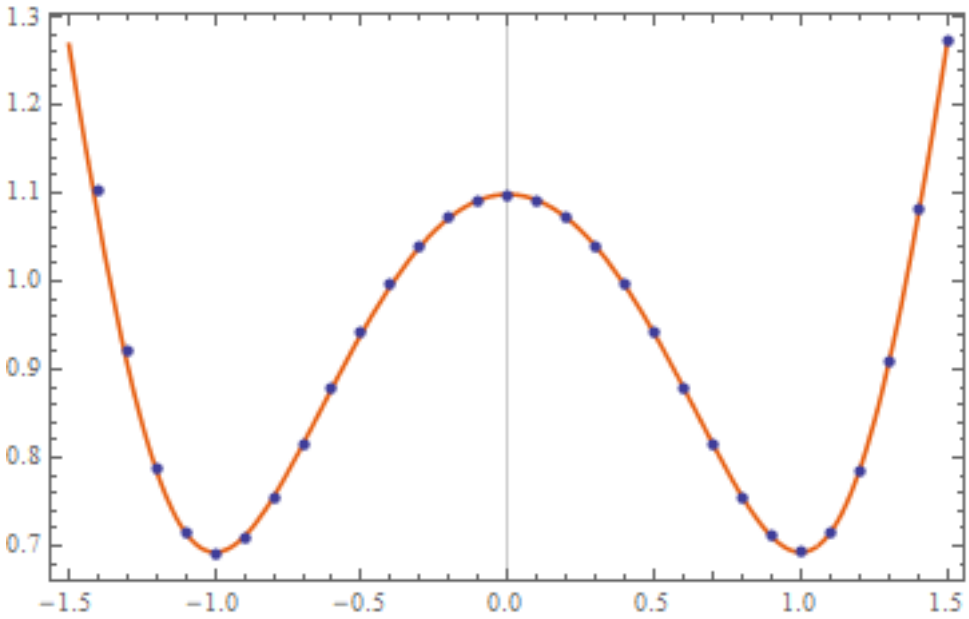
x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	1.33436	1.17951	1.02466	0.86981	0.71496	0.74786	0.78076	0.81366
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.84656	0.87946	0.92763	0.97580	1.02396	1.07213	1.07213	1.07213
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	1.07213	1.07213	1.02079	0.96944	0.91810	0.86675	0.81541	0.76620
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	0.71699	0.72383	0.75468	0.78554	0.94879	1.11204	1.27529	



Графік 2 . $f(x)$ та точки отримані лінійною інтерполяцією

3.3) Значення функції отримані інтерполяцією методом Лагранжа

x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	1.30957	1.10171	0.91991	0.78905	0.71496	0.69219	0.70966	0.75449
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.81448	0.87946	0.94182	0.99645	1.04037	1.07213	1.09122	1.09757
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	1.09121	1.07213	1.04042	0.99662	0.94217	0.88008	0.81541	0.75571
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	0.71098	0.69312	0.71453	0.78554	0.91052	1.08211	1.27329	

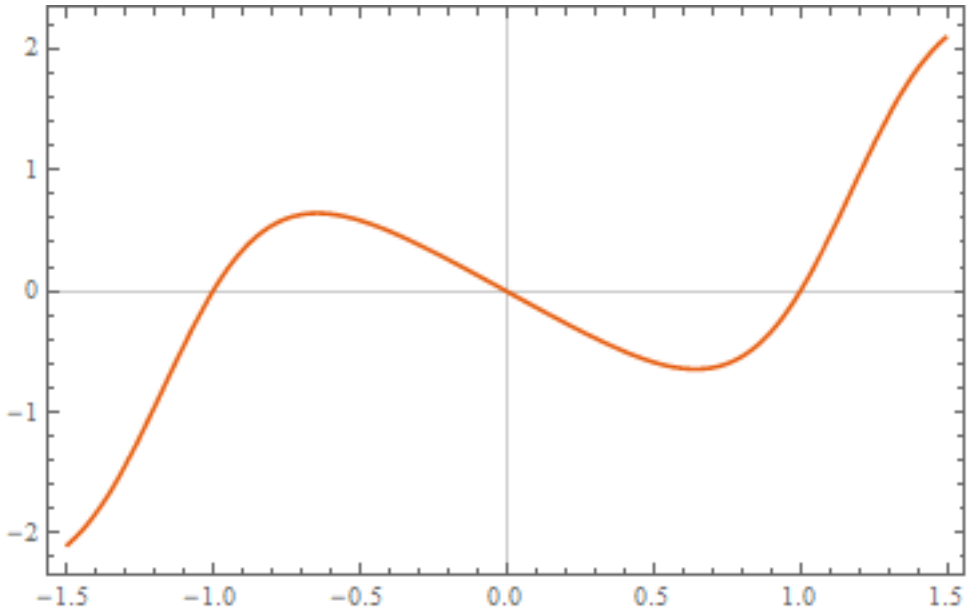


Графік 3 . $f(x)$ та точки отримані інтерполяцією Лагранжа

Порівнюючи таблиці 3.1, 3.2, 3.3, та графіки 1, 2 та 3 можна помітити що найближчими до реальних значень $f(x)$ є значення, отримані інтерполяцією Лагранжа.

3.4) Значення першої похідної функції отримані підстановкою у формулу $f'(x)$

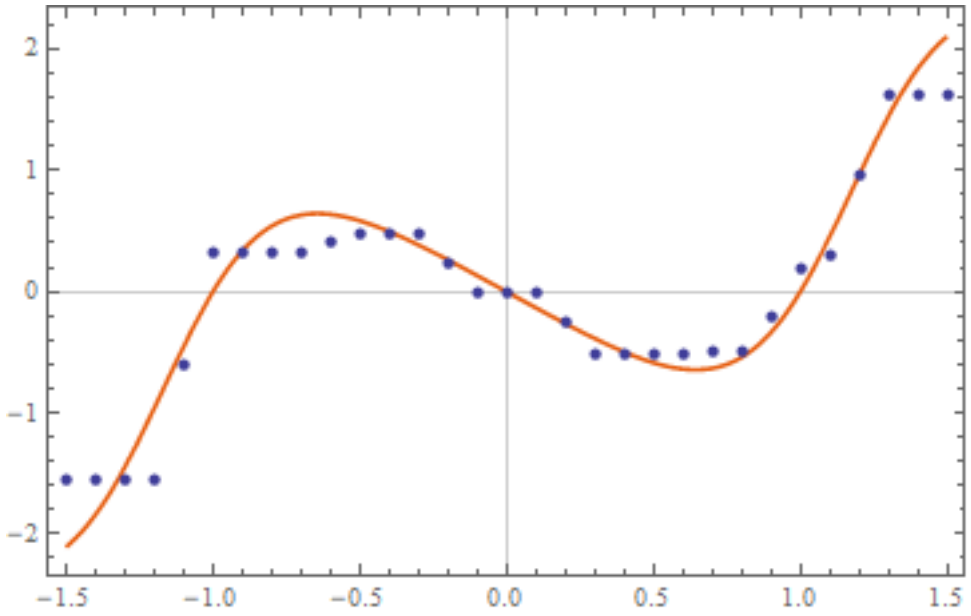
x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	-2.10526	-1.84009	-1.44905	-0.96280	-0.45203	0.00000	0.33594	0.54095
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.63183	0.63745	0.58537	0.49675	0.38612	0.26287	0.13288	0.00000
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	-0.13288	-0.26287	-0.38612	-0.49675	-0.58537	-0.63745	-0.63183	-0.54095
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	-0.33594	0.00000	0.45203	0.96280	1.44905	1.84009	2.10526	



Графік 4 . $f'(x)$

3.5) Наближені значення першої похідної, отримані чисельним диференціюванням

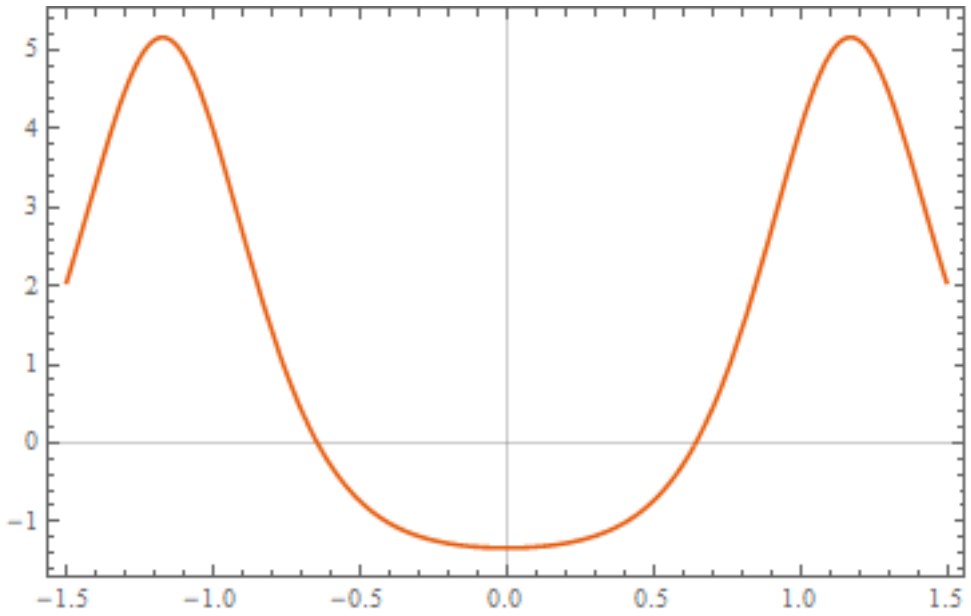
x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	-1.54850	-1.54850	-1.54850	-1.54850	-0.60975	0.32900	0.32900	0.32900
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.32900	0.40534	0.48168	0.48168	0.48167	0.24084	0.00000	0.00000
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	0.00000	-0.25672	-0.51344	-0.51344	-0.51344	-0.51344	-0.50276	-0.49209
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	-0.21186	0.18846	0.30856	0.97052	1.63248	1.63248	1.63248	



Графік 5 . $f'(x)$ та точки отримані чисельним диференціюванням

3.6) Значення другої похідної функції отримані підстановкою у формулу $f''(x)$

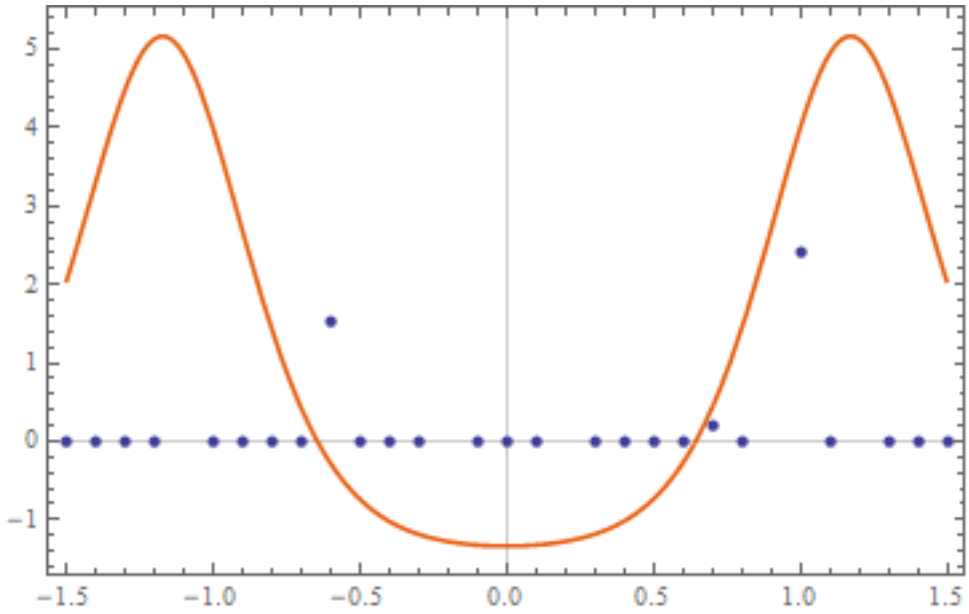
x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	2.02401	3.29535	4.47510	5.12699	4.94219	4.00000	2.69644	1.43540
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.43261	-0.27354	-0.73290	-1.01553	-1.18159	-1.27392	-1.31963	-1.33333
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	-1.31963	-1.27392	-1.18159	-1.01553	-0.73290	-0.27354	0.43261	1.43540
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	2.69644	4.00000	4.94219	5.12699	4.47510	3.29535	2.02401	



Графік 6 . $f''(x)$

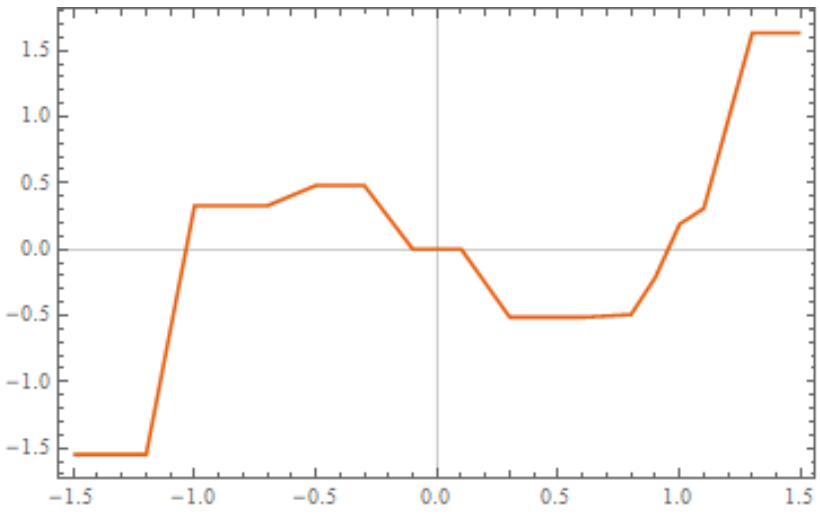
3.7) Наближені значення другої похідної, отримані чисельним диференціюванням

x	-1.5	-1.4	-1.3	-1.2	-1.1	-1.0	-0.9	-0.8
$f(x)$	0.00000	0.00000	0.00000	0.00000	18.77500	0.00000	0.00000	0.00000
x	-0.7	-0.6	-0.5	-0.4	-0.3	-0.2	-0.1	0
$f(x)$	0.00000	1.52675	0.00000	0.00000	0.00000	-4.81675	0.00000	0.00000
x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	0.00000	-5.13440	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
x	0.9	1	1.1	1.2	1.3	1.4	1.5	
$f(x)$	5.60450	2.40193	0.00000	13.23928	0.00000	0.00000	0.00000	



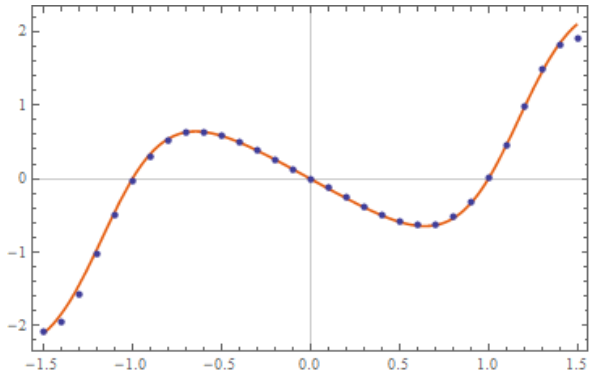
Графік 7 . $f''(x)$ та точки, отримані чисельним диференціюванням лінійно інтерпольованої функції $f(x)$

З таблиці 3.5 та графіку 5 видно, деякі точки наближено дорівнюють істинним значенням першої похідної, майже всі точки утворюють горизонтальні прямі. Усі значення другої похідної функції отримані чисельним диференціюванням лінійно інтерпольованої функції рівні 0, як і передбачалось у п. 2, за винятком деяких точок. Аномальність цих точок можна пояснити, якщо з'єднати точки з таблиці 3.5(наближені значення $f'(x)$) прямим(графік 8). На графіку видно що ламана складається з горизонтальних та похилих прямих відрізків (прямих). Кутові коефіцієнти цих прямих є значеннями другої похідної лінійно інтерпольованої функції. Тому отримано значну кількість нулів (горизонтальні прямі) та отримано деякі числа, які є кутовими коефіцієнтами похилих прямих – це і є «аномальні» точки.

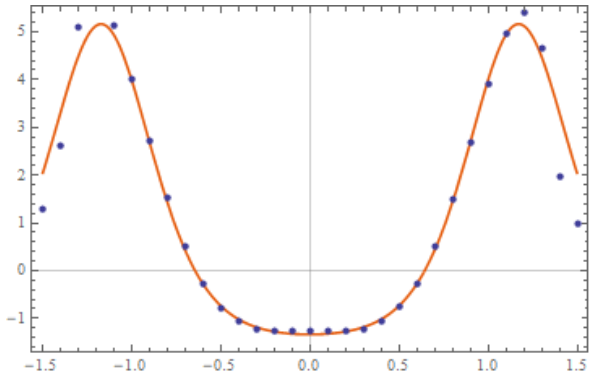


Графік 8.

3.8) Оскільки знаходження другої похідної методом чисельного диференціювання на основі значень лінійної інтерполяції функції не працює, то для перевірки реалізації алгоритму пошуку другої похідної було використано інтерполяцію методу Лагранжа. Результат можна побачити на графіках 9 та 10.



Графік 9. $f'(x)$ та її наближені значення



Графік 10. $f''(x)$ та її наближені значення

3. Висновки

В даній лабораторній роботі були реалізовані мовою C# алгоритм лінійної інтерполяції функції, алгоритм інтерполяції функції поліномом Лагранжа, алгоритми чисельного диференціювання. Побудовано таблиці, які вимагались в постановці задачі, та доведена недоцільність пошуку похідної другого та вищих порядків, на основі значень, отриманих шляхом лінійної інтерполяції функції.

Лістинг 7 – весь код програми

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Globalization;

namespace Alg_Lab1_Var6_Part1_console
{
    class Program
    {
        /// <summary>
        /// Виводить таблицю аргументів і значень функції
        /// </summary>
        /// <param name="x_arr">масив аргументів функці</param>
        /// <param name="y_arr">масив значень функції</param>
        static void PrintTable(double[] x_arr, double[] y_arr)
        {

            Console.WriteLine("\n_____________________________________________________");
            for (int i = 0; i < x_arr.Length; i++)
            {
                Console.Write("{0,-10:f2}", x_arr[i]);
            }
            Console.WriteLine("\n-----------------------------------------------------");
            for (int i = 0; i < y_arr.Length; i++)
            {
                Console.Write("{0,-10:f5}", y_arr[i]);
            }
            Console.WriteLine("\n_____________________________________________________");
        }

        /// <summary>
        /// Заповнює масив x з заданим кроком
        /// </summary>
        /// <param name="from">початок проміжку</param>
        /// <param name="h">крок</param>
        /// <param name="to">кінець проміжку</param>
        /// <returns>масив x</returns>
        static double[] GenereteXarray(double from,double h,double to)
        {
            int n = (int)((to - from) / h)+1;
            double[] x_arr = new double[n];
            x_arr[0] = from;
            for (int i = 1; i < n; i++)
            {
                x_arr[i] = x_arr[i - 1] + h;
            }
            return x_arr;
         }
         /// <summary>
        /// Лінійна інтерполяція функції, заданої точками
        /// </summary>
        /// <param name="in_x_arr">масив аргументів точок вузлів</param>
        /// <param name="in_y_arr">масив значень точок вузлів</param>
        /// <param name="out_x_arr"> масив аргументів функції</param>
        /// <param name="out_y_arr">вихідний масив значень функції</param>
        static void LinearInterpolation(double[] in_x_arr, double[] in_y_arr, double[]
        out_x_arr, double[] out_y_arr)
        {
            double[] a = new double[in_x_arr.Length - 1];
            double[] b = new double[in_x_arr.Length - 1];

            for (int i = 0; i < in_x_arr.Length-1; i++)
            {
                a[i] = (in_y_arr[i + 1] - in_y_arr[i]) / (in_x_arr[i + 1] - in_x_arr[i]);
```

```
            b[i] = in_y_arr[i] - a[i] * in_x_arr[i];
        }

        int j = 0;
        for (int i = 0; i < out_x_arr.Length; i++)
        {
            for (j = 0; j < in_x_arr.Length-1; j++)
            {
                if (out_x_arr[i] < in_x_arr[j + 1] && out_x_arr[i] > in_x_arr[j]) break;
                // перевірка чи потрапляє х у необхідний відрізок
            }
            out_y_arr[i] = a[j] * out_x_arr[i] + b[j];
        }
    }

    /// <summary>
    /// апроксимація функції методом Лагранжа
    /// </summary>
    /// <param name="in_x_arr">масив х координат вузлів</param>
    /// <param name="in_y_arr">масив у координат вузлів</param>
    /// <param name="out_x_arr">вихідний масив аргументів функції</param>
    /// <param name="out_y_arr">вихідний масив значень фукції</param>
    static void LagrangeInterpolation(double[] in_x_arr, double[] in_y_arr, double[]
    out_x_arr, double[] out_y_arr)
    {
        double l=1;
        for (int j = 0; j < out_x_arr.Length; j++)
        {
            out_y_arr[j] = 0;
            for (int i = 0; i < in_y_arr.Length; i++)
            {
                for (int k = 0; k < in_y_arr.Length; k++)
                {
                    if (k != i)
                        l *= (out_x_arr[j] - in_x_arr[k]) /( in_x_arr[i] - in_x_arr[k]);
                }

                out_y_arr[j] += in_y_arr[i] *l;
                l = 1;
            }
        }
    }

    /// <summary>
    /// Обчислює значення першої та другої поїхідної шляхом числового диференціювання
    /// </summary>
    /// <param name="out_x_arr">масив аргументів функції</param>
    /// <param name="in_y_arr">масив значень функції</param>
    /// <param name="first_diff_y">масив значень першої похідної</param>
    /// <param name="secon_diff_y">масив значень другої похідної</param>
    /// <param name="h">крок</param>
    static void Differentiate(double[] out_x_arr,double[] in_y_arr,double[]
    first_diff_y,double[] secon_diff_y,double h )
    {
        //для обислення похідної в крайніх точках використано формули правої та лівої
          похідної
        first_diff_y[0] = (in_y_arr[1] - in_y_arr[0]) / h;
        first_diff_y[out_x_arr.Length - 1] = (in_y_arr[out_x_arr.Length - 1] -
        in_y_arr[out_x_arr.Length - 2]) / h;

        for (int i = 1; i < out_x_arr.Length - 1; i++)
        {
            first_diff_y[i] = (in_y_arr[i + 1] - in_y_arr[i - 1]) / (2 * h);
        }

        //для обчислення другої похідної в крайніх точках використано формули правої та
        лівої похідної та значення першої похідної
        secon_diff_y[0] = (first_diff_y[1] - first_diff_y[0]) / h;
```

```
        secon_diff_y[out_x_arr.Length - 1] = (first_diff_y[out_x_arr.Length - 1] -
        first_diff_y[out_x_arr.Length - 2]) / h;

        for (int i = 1; i < out_x_arr.Length - 1; i++)
        {
           secon_diff_y[i] = (in_y_arr[i + 1] + in_y_arr[i - 1] - 2 * in_y_arr[i]) / (h
           * h);
        }
    }

    static void Main(string[] args)
    {
        try
        {
            double from = -1.5; //початкова межа
            double to = 1.5;    //кінцева межа
            double h = 0.1;     //крок x в таблиці
            int n = 9;          //кількість вузлів

            double[] x_array = { -1.6, -1.1, -0.6, -0.2, 0.2, 0.7, 0.93, 1.2, 1.51 };
            double[] y_array = { 1.48921, 0.71496, 0.87946, 1.07213, 1.07213, 0.81541,
            0.70223, 0.78554, 1.29161 };

            Console.WriteLine("Вузли функції");
            PrintTable(x_array, y_array);

            double[] out_x_array = GenereteXarray(from, h, to); // масив аргументів
            функції
            double[] true_y_array = new double[out_x_array.Length];// масив значень
            функції, отриманих підстановкою у формулу

            for (int i = 0; i < out_x_array.Length; i++)
            {
                 true_y_array[i] = Math.Log(Math.Pow(out_x_array[i], 4) - 2 *
                 Math.Pow(out_x_array[i], 2) + 3);

            }
            Console.WriteLine("\nІстинні значення функції");
            PrintTable(out_x_array, true_y_array);

            double[] lin_y_arr = new double[out_x_array.Length];//масив значень функції
            отриманих лінійною апроксимацією

            LinearInterpolation(x_array, y_array, out_x_array, lin_y_arr);

            Console.WriteLine("\nЗначення функції, отримані лінійною апроксимацією");
            PrintTable(out_x_array, lin_y_arr);

            double[] lag_y_arr = new double[out_x_array.Length];//масив значень функції
            отриманих апроксимацією Лагранжа
            LagrangeInterpolation(x_array, y_array, out_x_array, lag_y_arr);

            Console.WriteLine("\nЗначення функції, отримані апроксимацією Лагранжа");
            PrintTable(out_x_array, lag_y_arr);



            double[] true_first_diff_y = new double[out_x_array.Length];//масив істинних
            значень похідної функції
            for (int i = 0; i < out_x_array.Length; i++)
            {
               true_first_diff_y[i] = (-4 * out_x_array[i] + 4 *
               Math.Pow(out_x_array[i], 3)) / (Math.Pow(out_x_array[i], 4) - 2 *
               Math.Pow(out_x_array[i], 2) + 3);
            }

            Console.WriteLine("\nІстинні значення першої похідної");
            PrintTable(out_x_array, true_first_diff_y);
```

```
                double[] true_second_diff_y = new double[out_x_array.Length];//масив
                істинних значень другої похідної функції
                for (int i = 0; i < out_x_array.Length; i++)
                {
                   true_second_diff_y[i] = 4 * (-Math.Pow(out_x_array[i], 6) +
                   Math.Pow(out_x_array[i], 4) + 7 * Math.Pow(out_x_array[i], 2) - 3) /
                   Math.Pow(Math.Pow(out_x_array[i], 4) - 2 * Math.Pow(out_x_array[i], 2) +
                   3, 2);
                }

                Console.WriteLine("\nІстинні значення другої похідної");
                PrintTable(out_x_array, true_second_diff_y);

                double[] first_diff_y = new double[out_x_array.Length];//масив значень
                похідної функції отриманих числовим диференціюванням
                double[] second_diff_y = new double[out_x_array.Length];//масив значень
                другої похідної функції отриманих числовим диференціюванням

                Differentiate(out_x_array, lin_y_arr, first_diff_y, second_diff_y, h);

                Console.WriteLine("\nЗначення першої похідної");
                PrintTable(out_x_array, first_diff_y);

                Console.WriteLine("\nЗначення другої похідної");
                PrintTable(out_x_array, second_diff_y);
            }
            catch (Exception e)
            {
                Console.WriteLine("Щось пішло не так :(");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```