Міністерство освіти і науки України
Національний Технічний Університет України
Київський політехнічний інститут імені Ігоря Сікорського
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №4

з дисципліни «Чисельні методи»

Тема «Інтерполяційні поліноми»

Варіант №22 (2)

Студента 2-го курсу НН ІАТЕ гр. ТР-12
Ковальова Олександра
Перевірила: к.т.н., доц. Залевська О. В.

КИЇВ 2022

Мета роботи. Набути навичок у використанні інтерполяційних поліномів. Написати багаточлен Ньютона за заданими точками. Описати функцію методом кубічних сплайнів за заданими точками. Порівняти отримані функції з даною.

Теоретична частина.

Інтерполяція – в обчислювальній математиці спосіб знаходження проміжних значень величини по наявному дискретному наборі відомих значень.

Нехай маємо n значень x_i, кожному з який відповідає своє значення y_i. Потрібно знайти таку функцію F, що

$$F(x_i) = y_i, i = 0, \dots, n. (1)$$

При цьому x_i називаються вузлами інтерполяції; пари (x_i, y_i) – точками даних; функцію $F(x)$ – інтерполянтом.

Інтерполянти, як правило, будуються у вигляді лінійних комбінацій деяких елементарних функцій:

$$y = \sum_{k=0} c_k \Phi_{\,k}(x)^n ,$$

де $\Phi_k(x)$ – фіксовані лінійно незалежні функції; $c_0, \dots, c_n$ – не визначені поки що коефіцієнти.

З умови (1) отримуємо систему n+1 рівнянь відносно коефіцієнтів c_k:

$$\sum_{k=0} c_k \Phi_k(x_i) = y_i, i = 0, \dots, n^n,$$

В якості системи лінійно незалежних функцій $\Phi_k(\mathrm{x})$ частіше за все обирають: степеневі функції $\Phi_k(\mathrm{x}) = x^k$ (в цьому випадку $F = P_n(x)$ – поліном ступеня n); тригонометричні функції.

Поліном Лагранжа.

Будемо шукати інтерполяційний поліном у вигляді:

$$P_{\text{т}}(x) = \sum_{k=0}^{n} c_k x^k. \qquad (2)$$

Звідси отримуємо систему рівнянь:

$$\begin{gathered} c_0 + c_1\, x_0 + \ldots + c_n x_0^n = y_0 \\ \ldots \\ c_0 + c_1\, x_n + \ldots + c_n x_n^n = y_n \end{gathered}$$

Ця система має єдиний розв'язок, а отже і інтерполяційний поліном вигляду (2) також єдиний. Форм запису його існує багато.

Лагранж запропонував наступну форму поліному, в основі якої лежить базис поліномів Лагранжа $l_k(x)$ ступеня n.

Поліноми Лагранжа мають вигляд:

$$l_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i} = \frac{(x - x_0)(x - x_1)\dots(x - x_n)}{(x_k - x_0)(x_k - x_1)\dots(x_k - x_n)}. \qquad (3)$$

Тоді поліном $P_n(x)$ набуде вигляду:

$$P_n(x) = \sum_{k=0}^{n} y_k \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i} \qquad (4)$$

Цей поліном має ступінь не вищу за n та $P_n(x_i) = y_i$. Формулу (4) називають формулою Лагранжа. Кількість арифметичних дій для обчислення за (4) пропорційна n^2.

Поліном Ньютона.

При використанні інтерполяційного поліному Ньютона застосовується поняття роздільної різниці:

роздільна різниця першого роду: $y(x_i, x_j) = \frac{y(x_i) - y(x_j)}{x_i - x_j}$, роздільна різниця другого роду $y(x_i, x_j, x_k) = \frac{y(x_i, x_j) - y(x_j, x_k)}{x_i - x_k}$ і т. д.

Якщо $y(x) = P_n(x)$ – поліном ступеню n, то для нього перша роздільна різниця $P(x, x_0)$ – поліном $n - 1$ ступеню, друга роздільна різниця P(x,x0,x1) – поліном $n - 2$ ступеню і т.д., так що $(n + 1) -$ а роздільна різниця дорівнює нулю. Із визначення роздільних різниць отримуємо:

$$P(x) = P(x_0) + (x - x_0)P(x, x_0)$$
$$P(x, x_0) = P(x_0, x_1) + (x - x_1)P(x, x_0, x_1)$$
$$P(x, x_0, x_1) = P(x_0, x_1, x_2) + (x - x_2)P(x, x_0, x_1, x_2)$$
$$\dots$$

Звідси отримуємо формулу для $P_n(x)$:

$$P(x) = P(x_0) + (x - x_0)P(x_0, x_1) + (x - x_0)(x - -x_1)P(x_0, x_1, x_2) + \dots + (x - x_0)(x - x_1)\dots(x - x_{n-1})P(x_0, x_1, \dots, x_n) \qquad (5)$$

Якщо $P_n(x)$– інтерполяційний поліном для функції $y(x)$, то його роздільні різниці співпадають із роздільними різницями функції. Тоді можна записати:

$$F(x) = y_0 + \sum_{k=1}^{n}(x - x_0)(x - x_1)\ldots(x - x_{k-1})y(x_0, x_1, \ldots, x_k)$$

Частіше використовують поліном Ньютона у формі Горнера (перед цим необхідно обчислити всі роздільні різниці):

$$F(x) = y(x_0) + (x - x_0)\big[y(x_0, x_1) + (x - x_1)[y(x_0, x_1, x_2) + \ldots]\big] \quad (6)$$

Обчислення $F(x)$ для кожного x потребує n множень та $2n$ додавань або віднімань.

Сплайн-інтерполяція.

Розглянемо спеціальний випадок кусково-поліноміальної інтерполяції, коли між будь-якими сусідніми вузлами інтерполяції функція інтерполюється кубічним поліномом (кубічна сплайн-інтерполяція). Його коефіцієнти на кожному інтервалі визначаються з умов сполучення у вузлах:

$$F(x_i) = y_i, \qquad (7)$$
$$F'(x_i - 0) = F'(x_i + 0), \quad (8)$$
$$F''(x_i - 0) = F''(x_i + 0), i = 1, \ldots, n - 1. \quad (9)$$

Крім того, на границях при $x = x_0$ та $x = x_n$ встановлюються умови:

$$F''(x_0) = 0, F''(x_n) = 0.$$

Будемо шукати кубічний поліном у вигляді:

$$F_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3, i = 1, \ldots, n - 1. \quad (10)$$

Тоді $F_i(x_i) = a_i, F'{}_i(x_i) = b_i, F''{}_i(x_i) = c_i$. Для виконання умови неперервності: $F_i(x_{i-1}) = y_{i-1}$. Звідси отримуємо формули для обчислення коефіцієнтів сплайну, підставивши (10) у рівняння (7), (8), (9) (тут $h_i = x_i - x_{i-1}$).

$$a_i = y_i$$
$$h_i c_{i-1} + 2(h + h_{i+1})c_i + h_{i+1}c_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_1 - y_{i-1}}{h_i}\right)$$
$$d_i = \frac{c_i - c_{i-1}}{h_i}$$
$$b_i = \frac{1}{2}h_i c_i - \frac{1}{6}h_i^2 d_i + \frac{y_i - y_{i-1}}{h_i}$$

Якщо врахувати, що $c_1 = c_{n+1} = 0$, то обчислення коефіцієнтів c можна провести за допомогою методу прогону для трьохдіагональної матриці.

Метод прогону

Більшість технічних задач зводиться до розв’язування систем ЛАР, в яких матриці містять багато нульових елементів, а ненульові елементи розміщені за спеціальною структурою (стрічкові квазітрикутні матриці).

Задачі побудови інтерполяційних сплайнів, різницевих методів розв’язування крайових задач для диференціальних рівнянь зводяться до розв’язування систем ЛАР з трьохдіагональною матрицею А. В матриці А всі елементи, що не лежать на головній діагоналі і двох сусідніх паралельних діагоналях, дорівнюють нулю.

В загальному вигляді такі системи записують так:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \quad (1)$$
$$1 \le i \le n; a_1 = 0; c_n = 0$$

або в розгорнутому вигляді:

$$\begin{cases} b_1x_1 + c_1x_2 = d_1 \\ a_2x_1 + b_2x_2 + c_2x_3 = d_2 \\ a_3x_2 + b_3x_3 + c_3x_4 = d_3 \\ \dots \dots \dots \dots \dots \dots \dots \\ a_ix_{i-1} + b_ix_i + c_ix_{i+1} = d_i \\ a_{n-1}x_{n-2} + b_{n-1}x_{n-1} + c_{n-1}x_n = d_{n-1} \\ a_nx_{n-1} + b_nx_n = d_n \end{cases} \quad (2)$$

Вибір найбільшого елемента при виключенні невідомих за методом Гауса в таких системах робити не можна, оскільки перестановка рядків руйнує структуру матриці. Найчастіше для розв’язку системи з трьохдіагональною матрицею використовують метод прогону, який є частковим випадком методу Гауса.

Прямий хід прогону (алгоритм прямого ходу методу Гауса).

Кожне невідоме x_iвиражається через $x_{i+!}$з допомогою прогоночних коефіцієнтів A_i та B_i

$$x_i = A_i x_{i+1} + B_i; i = \overline{1, n} \quad (3)$$

Наприклад, з першого рівняння системи (2) знайдемо:

$$\left.\begin{array}{l} x_1 = -\dfrac{c_1}{b_1} x_2 + \dfrac{d_1}{b_1} \\ x_1 = A_1 x_2 + B_1 \end{array}\right\}, \text{ звідки } \quad A_1 = -\frac{c_1}{b_1} \quad B_1 = \frac{d_1}{b_1} \quad (4)$$

З другого рівняння системи (3) виразимо x_2 через x_3, замінюючи x_1 формулою (3) або (4)

$$a_2x_1 + b_2x_2 + c_2x_3 = a_2(A_1x_2 + B_1) + b_2x_2 = d_2$$

Звідси знайдемо

$$x_2 \frac{d_2 - c_2 x_3 - a_2 B_1}{a_2 A_1 + b_2}, \text{або } x_2 = A_2 x_3 + B_2$$

$$A_2 = -\frac{c_2}{a_2 A_1 + b_2}; B_2 = -\frac{d_2 - a_2 B_1}{a_2 A_1 + b_2}, \text{беручи за } e_2 = a_2 A_1 + b_2$$

Запишемо:

$$A_2 = -\frac{c_2}{e_2}; \quad B_2 = \frac{d_2 - a_2 B_1}{e_2}$$

Аналогічно для кожного і прогоночні коефіцієнти з рівняння $x_i = A_i x_{i+1} + B_i$ мають вигляд:

$$A_i = -\frac{c_i}{e_i}; \quad B_i = \frac{d_i - a_i B_{i-1}}{e_i} \quad (5)$$

$$e_i = a_i A_{i+1} + b_i; i = \overline{1, n}$$

При цьому враховуючи, що $a_1 = c_n = 0$, приймаємо

$$A_0 = 0; B_0 = 0 \quad (6)$$

В розгорнутому вигляді формула (5) буде мати вигляд формули (7). Значення прогоночних коефіцієнтів можна одержати і таким шляхом. В рівнянні (3) понизимо індекс на одиницю та підставимо значення xi-1 в і-е рівняння системи (1)

$$x_{i-1} = A_{i-1} \cdot x_i + B_{i-1}$$

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i \Rightarrow a_i(A_{i-1} x_i + B_{i-1}) + b_i x_i + c_i x_{i+1} = d_i$$

$$x_i = \frac{d_i - c_i x_{i+1} - a_i B_{i-1}}{a_i A_{i-1} + b_i} = -\frac{c_i}{a_i A_{i-1} + b_i} x_{i+1} + \frac{d_i - a_i B_{i-1}}{a_i A_{i-1} + b_i} = A_i x_{i+1} + B_i \quad (7)$$

Обернений хід прогонки (аналог оберненого ходу методу Гауса).

Він полягає в послідовному обчисленні невідомих xi. Спочатку знаходять цього формулу (7) запишемо при i = n (враховуючи, що Cn = 0)

$$x_n = -\frac{c_n}{a_n A_{n-1} + b_n} x_{n+1} + \frac{d_n - a_n B_{n-1}}{a_n A_{n-1} + b_n} = \frac{d_n - a_n B_{n-1}}{a_n A_{n-1} + b_n} = B_n$$

Долі використовуючи формулу (3) знаходимо послідовно всі невідомі xn-1, xn-2, … , x1.

Майже у всіх задачах, що приводять до розв'язку системи (2) з трьохдіагональною матрицею, забезпечується умова переважання діагональних коефіцієнтів

$$|b_i| \geq |a_i| + |c_i|$$

Це забезпечує існування єдиного розв'язку та достатню стійкість методу прогону відносно похибок заокруглення.

Для запису коефіцієнтів ai, bi, та прогоночних коефіцієнтів Ai-1, Bi-1 використати один і той же масив.

Завдання.

Створити програму, яка для заданої функції по заданим точкам будує інтерполяційний поліном $P_n(x)$ у формі Лагранжа або Ньютона, а також здійснює інтерполяцію кубічними сплайнами.

Програма має розраховувати значення похибки $\varepsilon = |P_n(x) - y(x)|$, для чого потрібно вивести на графік із кроком (графік можна будувати допоміжними засобами, наприклад, у Mathcad), меншим у 5-6 разів, ніж крок інтерполяції, відповідні значення поліному та точної функції. Якщо похибка дуже мала, застосувати масштабування.

Знайти кубічний інтерполяційний сплайн для заданої функції у Mathcad. Вивести графік результатів.

Функція:

$$\sin\frac{\alpha}{2}x + \sqrt[3]{x\alpha}$$

α – остання цифра номеру групи, отже $\alpha = 2$, і функція матиме вигляд:

$$\sin x + \sqrt[3]{x}$$

Вузли інтерполяції:

$$-5 + k, \qquad -3 + k, \qquad -1 + k, \qquad 1 + k, \qquad 3 + k$$

$k = №_{\text{варіанту}} - 1 = 1$, отже у кінцевому вигляді вузли інтерполяції такі:

$$-4, \qquad -2, \qquad 0, \qquad 2, \qquad 4$$

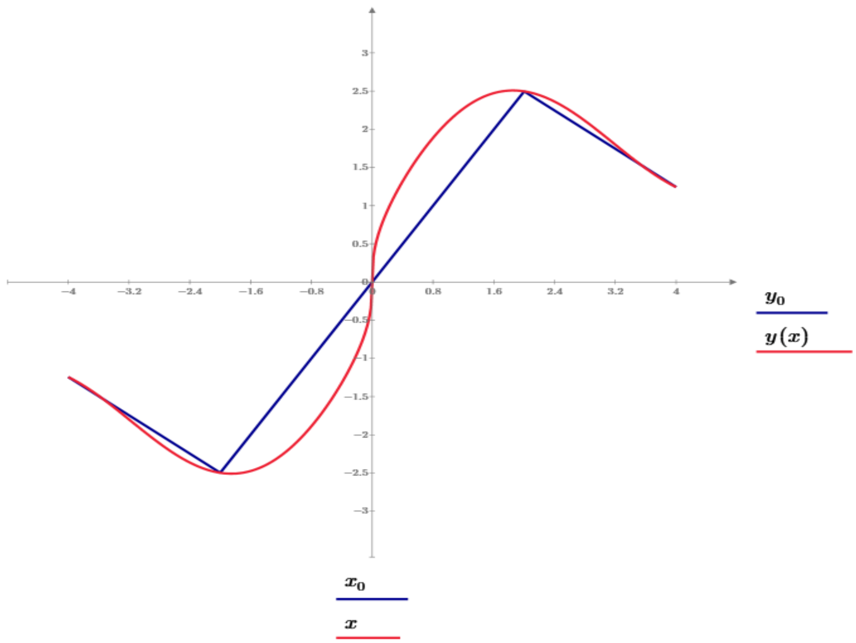
Хід роботи.

Таблиця значень у вузлах інтерполяції:

x	-4	-2	0	2	4
y	$-1,2432$	$-2,4967$	0	2,4967	1,2432

Графік функції.

$y(x)$ – точний графік функції (червоним), y_0 – крива, побудована за точками інтерполяції (синім):



Вигляд поліному Ньютона.

Розділені різниці:

- першого роду: $y(x_i, x_j) = \frac{y(x_i) - y(x_j)}{x_i - x_j}$
- другого роду: $y(x_i, x_j, x_k) = \frac{y(x_i, x_j) - (x_j, x_k)}{x_i - x_k}$
- третього роду: $y(x_i, x_j, x_k, x_l) = \frac{y(x_i, x_j, x_k) - (x_j, x_k, x_l)}{x_i - x_l}$
- четвертого роду: $y(x_i, x_j, x_k, x_l, x_m) = \frac{y(x_i, x_j, x_k, x_l) - (x_j, x_k, x_l, x_m)}{x_i - x_m}$

Тоді, формула інтерполяційного поліному Ньютона:

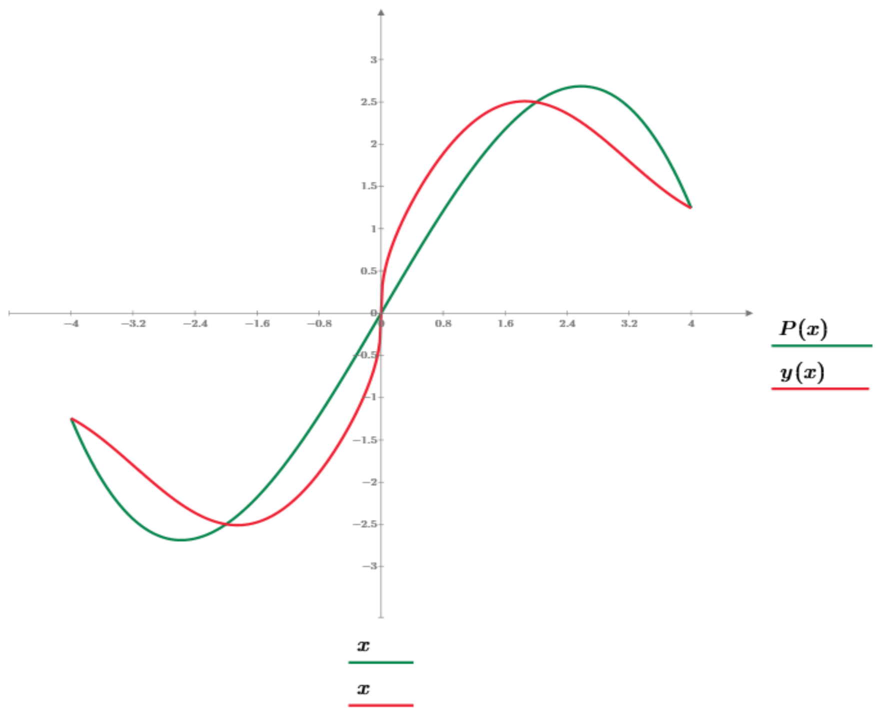
$$P(x) = y(x_0) + (x - x_0) * y(x_0, x_1) + (x - x_0)(x - x_1) * y(x_0, x_1, x_2) + (x - x_0)(x - x_1)(x - x_2) * y(x_0, x_1, x_2, x_3) + (x - x_0)(x - x_1)(x - x_2)(x - x_3) * y(x_0, x_1, x_2, x_3, x_4)$$

$$P(x) = y(-4) + (x + 4) * y(-4, -2) + (x + 4)(x + 2) * y(-4, -2, 0) + (x + 4)(x + 2)(x - 0) * y(-4, -2, 0, 2) + (x + 4)(x + 2)(x - 0)(x - 2) * y(-4, -2, 0, 2, 4)$$

$$P(x) = -1{,}2432 + (x + 4) * (-0{,}6268) + (x^2 + 6x + 8) * 0{,}4688 + (x^3 + 6x^2 + 8x) * (-0{,}0781) + (x^4 + 4x^3 - 4x^2 - 16x) * 0 = -0{,}0781x^3 + 0{,}0002x^2 + 1{,}5612x$$

$$P(x) = -0{,}0781x^3 + 0{,}0002x^2 + 1{,}5612x$$

Порівняльний графік функції та поліному:



Приклад роботи програми.

Програма для обчислень була написана мовою програмування C# у оточенні розробки JetBrains Rider. Версія мови – C# 11, платформа – .NET 7.

Розрахунок похибок для значень з кроком меншим у 5 разів ніж крок інтерполяції, тобто $\frac{2}{5} = 0{,}4$:

```
Function: y(x) = sin(x) + (2*x)^(1/3)
Interpolation nodes: -4, -2, 0, 2, 4

Newton's polynom:
-1.2432 - (x + 4) * 0.62675 + (x^2 + 6x + 8) * 0.468775 - (x^3 + 6x^2 + 8x) * 0.0781292

x = -4
y(x) = -1.243197505
P(x) = -1.243197505
Error = 0

x = -3.6
y(x) = -1.488458326
P(x) = -1.973923229
Error = 0.4854649036

x = -3.2
y(x) = -1.79826139
P(x) = -2.434634594
Error = 0.6363732036

x = -2.8
y(x) = -2.110796154
P(x) = -2.655333193
Error = 0.5445370392
```

```
x = 3.6
y(x) = 1.488458326
P(x) = 1.973923229
Error = 0.4854649036

x = 4
y(x) = 1.243197505
P(x) = 1.243197505
Error = 1.554312234e-15
```

Інтерполяція кубічними сплайнами, робота програми по знаходженню коефіцієнтів.

Вигляд i-того сплайнового інтерполяційного поліному:

$$S_{i(x)} = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, i = \overline{1, n-1}$$

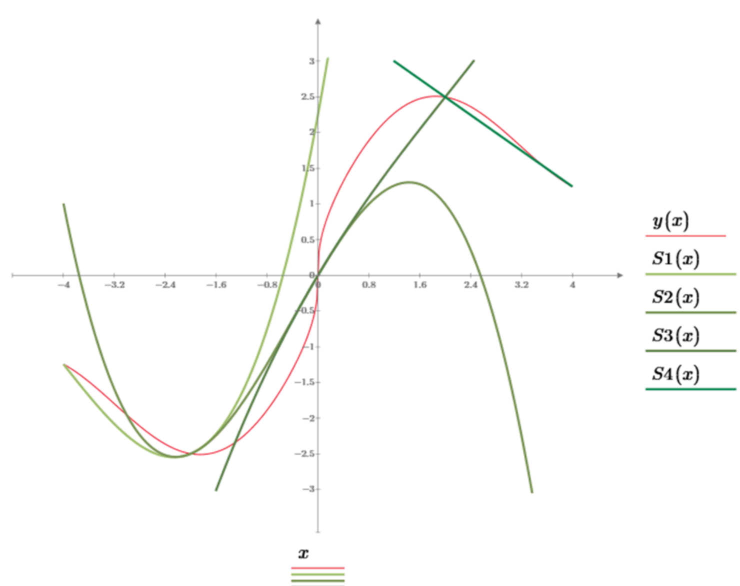
Коефіцієнти $a, b, c, d, i = \overline{1, n-1}$:

```
S1:
a =   -1.24319750469207 b =   -1.12677708077033
c =                   0 d =   0.125006648429856

S2:
a =   -2.49669847879388 b =   0.373302700387946
c =   0.750039890579138 d =   -0.15625831053732

S3:
a =                   0 b =    1.49836253625665
c = -0.187509972644785 d = 0.0312516621074641

S4:
a =    2.49669847879388 b = -0.626750487050905
c =                   0 d =                  -0
```

Порівняльний графік функції та сплайн-інтерполяції:

$y(x)$ – графік функції (червоним)

$S_i(x)$ – графік сплайн-інтерполяції (різнокольоровий)



Інтерполяція сплайнами у програмному забезпеченні Mathcad, порівняльні графіки.

$y(x)$ – графік функції (червоним)

$\mathrm{P}(x)$ – графік поліному Ньютона (зеленим)

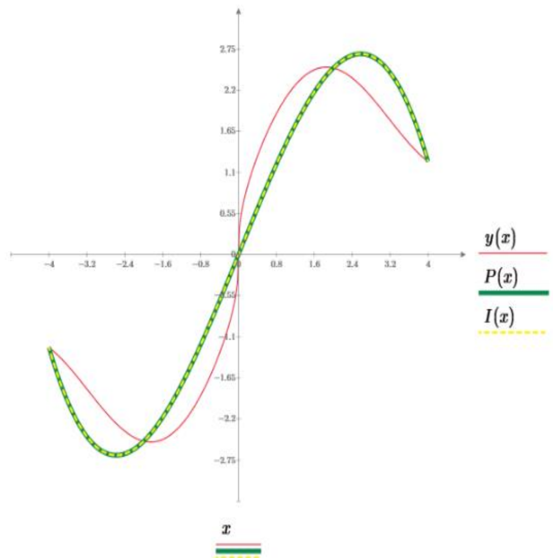
$I(t)$ – графік інтерполяції сплайнами Mathcad (жовтим)

$x_0 := [-4 \ \ -2 \ \ 0 \ \ 2 \ \ 4]^T$

$y_0 := \sin(x_0) + \sqrt[3]{2\,x_0}$

$s := \mathrm{cspline}(x_0, y_0)$

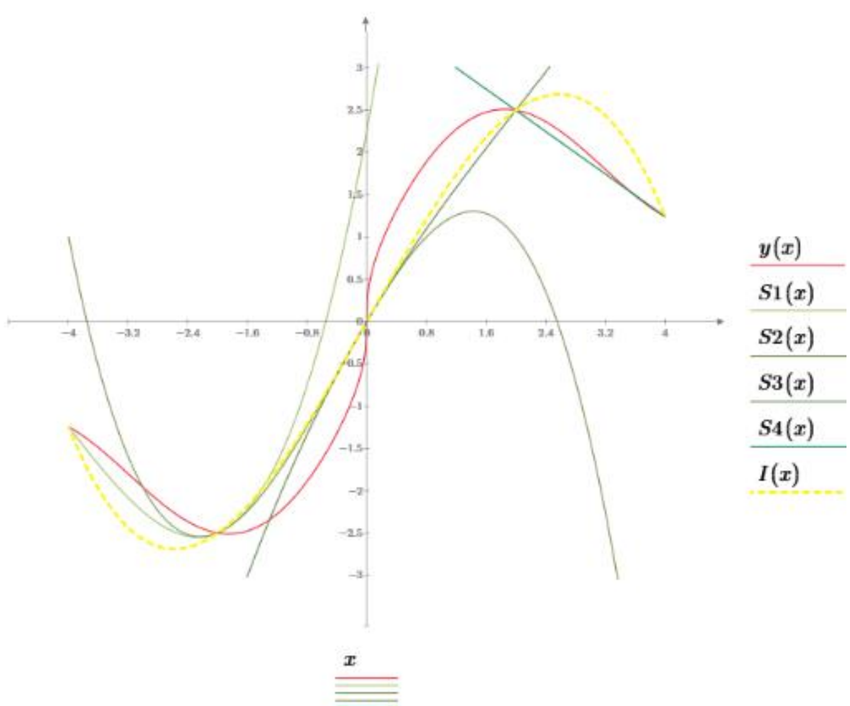
$I(t) := \mathrm{interp}(s, x_0, y_0, t)$



$y(x)$ – графік функції (червоним)

$S_i(x)$ – графік сплайн-інтерполяції(зеленим)

$I(t)$ – графік інтерполяції сплайнами Mathcad(жовтим)



Висновок. Під час виконання лабораторної роботи було освоєно обчислення поліному Ньютона за його точками, а також робота з інтерполяційними кубічними сплайнами. Були також побудовані графіки поліному, і порівняні графіки сплайнів та функції.

Додатки.

Код програми (Replit): https://replit.com/join/xdrdlknbuy-kovalevalex

Код програми:

Program.cs

```
namespace Lab4;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("--- Kovalov Alex, TP-12 ---");
        Console.WriteLine("        -- Lab 4 --      \n");

        Console.WriteLine(" Function: y(x) = sin(x) + (2*x)^(1/3)");
        Console.WriteLine(" Interpolation nodes: -4, -2, 0, 2, 4");

        Console.WriteLine();

        int n = 5;
        var x = new double[n];
        var y = new double[n];
        var step = -4.0;
        for (int i = 0; i < n; i++) {
            x[i] = step;
            step += 2;
            y[i] = Math.Sin(x[i]) + Math.Cbrt(2 * x[i]);
        }

        Calculations.NewtonsPolynom(x, y, n);
        Calculations.CSpline(x, y, n);
    }
}
```

Calculations.cs

```
namespace Lab4;

public static class Calculations
{
    public static double DividedDifference(double[] x, double[] y, int i, int k)
    {
        if (i == k) return y[i];

        return (DividedDifference(x, y, i, k - 1) - DividedDifference(x, y, i + 1, k)) / (x[i] - x[k]);
    }

    public static double CalcNewtonsPolynom(double x, double[] dd, Polynom[] pn, int n) {
        double rez = dd[0];

        for (int i = 1; i < n - 1; i++) {
            if (dd[i] != 0) rez += dd[i] * pn[i - 1].Calculate(x);
        }

        return rez;
```

```
        }

        public static void CalcNewtonsPolynomError(double[] x, double[] dd,
Polynom[] pn, int n) {
            double step = (x[1] - x[0]) / 5;

            for (int i = 0; i < n * 4.2; i++) {
                double xi = x[0] + step * i;
                Console.WriteLine($" x = {xi:F2}");

                double y = Math.Sin(xi) + Math.Cbrt(2 * xi);
                Console.WriteLine($" y(x) = {y:F10}");

                double p = CalcNewtonsPolynom(xi, dd, pn, n);
                Console.WriteLine($" P(x) = {p}");

                Console.WriteLine($" Error = {Math.Abs(p - y)}");

                Console.WriteLine();
            }
        }

        public static void NewtonsPolynom(double[] x, double[] y, int n) {
            var dd = new double[n];

            for (int i = 0; i < n; i++) {
                dd[i] = DividedDifference(x, y, 0, i);
            }

            Polynom[] pn = new Polynom[n - 1];

            for (int i = 0; i < n - 1; i++) {
                pn[i] = new Polynom(1, 1, null);
                pn[i].Next = new Polynom((-1) * x[i], 0, null);

                if (i <= 0) continue;

                pn[i] = pn[i].Product(pn[i - 1]);
                pn[i].Simplify();
            }

            Console.WriteLine(" Newton's polynom:");
            for (int i = 0; i < n; i++)
            {
                if (i == 0)
                {
                    Console.Write($" {dd[i]}");
                }
                else if (dd[i] != 0) {
                    if (dd[i] > 0) Console.Write(" + ");
                    else Console.Write(" - ");

                    Console.Write("(");
                    pn[i - 1].Print();
                    Console.Write($") * {Math.Abs(dd[i])}");
                }
            }

            Console.WriteLine("\n");

            CalcNewtonsPolynomError(x, dd, pn, n);
        }
```

```
            public static void ACoefs(double[] a, double[][] system, double[] y, int n)
{
                a[0] = 0.0;
                for (int i = 1; i < n - 2; i++) {
                    a[i] = ((-1.0) * system[i - 1][2]) / (system[i - 1][0] * a[i - 1] +
system[i - 1][1]);
                }
            }

            public static void BCoefs(double[] b, double[] a, double[][] system,
double[] y, int n) {
                b[0] = 0.0;
                for (int i = 1; i < n - 2; i++) {
                    b[i] = (system[i - 1][3] - system[i - 1][0] * b[i - 1]) / (system[i
- 1][0] * a[i - 1] + system[i - 1][1]);
                }
            }

            public static double CalcCSpline(double xi, double[] a, double[] b, double[]
c, double[] d, double[] x0, int n) {
                double x = 0, ai = 0, bi = 0, ci = 0, di = 0;

                for (int i = 1; i < n; i++) {
                    x = xi - x0[i - 1];
                    ai = a[i - 1];
                    bi = b[i - 1];
                    ci = c[i - 1];
                    di = d[i - 1];
                    if (xi < x0[i]) break;
                }
                double rez = ai + bi * x + ci * Math.Pow(x, 2) + di * Math.Pow(x, 3);
                return rez;
            }

            public static void CSplinePolynomError(double[] x, double[] a, double[] b,
double[] c, double[] d, int n) {
                double step = (x[1] - x[0]) / 5;
                for (int i = 0; i < n * 4.2; i++) {
                    double xi = x[0] + step * i;
                    Console.WriteLine($" x = {xi:F2}");

                    double y = Math.Sin(xi) + Math.Cbrt(2 * xi);
                    Console.WriteLine($" y(x) = {y:F10}");

                    double p = CalcCSpline(xi, a, b, c, d, x, n);
                    Console.WriteLine($" S(x) = {p}");

                    Console.WriteLine($" Error = {Math.Abs(p - y)}");

                    Console.WriteLine();
                }
            }

            public static void CSpline(double[] x, double[] y, int n) {
                double[] a = new double[n - 1];
                double[] b = new double[n - 1];
                double[] c = new double[n - 1];
                double[] d = new double[n - 1];
                double[] h = new double[n - 1];

                for (int i = 0; i < n - 1; i++) {
                    h[i] = x[i + 1] - x[i];
                    if (i == 0 || i == n - 2) c[i] = 0.0;
                }
```

```
            var system = new double[n - 3][];
            for (int i = 0; i < n - 3; i++) {
                system[i] = new double[4];
                for (int j = 0; j < 4; j++) {
                    switch (j) {
                        case 0: system[i][j] = h[i + 1]; break;
                        case 1: system[i][j] = 2 * (h[i + 1] + h[i + 2]); break;
                        case 2: system[i][j] = h[i + 2]; break;
                        case 3: system[i][j] = 3 * (((y[i + 2] - y[i + 1]) / h[i +
2]) - ((y[i + 1] - y[i]) / h[i + 1])); break;
                    }
                }
            }

            ACoefs(a, system, y, n);
            BCoefs(b, a, system, y, n);

            for (int i = 0; i < n - 3; i++) system[i] = null;

            for (int i = n - 3; i > 0; i--) c[i] = a[i] * c[i + 1] + b[i];

            for (int i = 0; i < n - 1; i++) {
                a[i] = y[i];
                if (i == n - 2) {
                    d[i] = ((-1.0) * c[i]) / (3 * h[i]);
                    b[i] = ((y[i + 1] - y[i]) / h[i]) - ((2 * h[i] * c[i]) / 3);
                } else {
                    d[i] = (c[i + 1] - c[i]) / (3 * h[i]);
                    b[i] = ((y[i + 1] - y[i]) / h[i]) - ((h[i] * (c[i + 1] + 2 *
c[i])) / 3);
                }

                Console.WriteLine($" S{i+1}:");

                Console.Write($" {$"a = {a[i]}", 18}");
                Console.WriteLine($" {$"b = {b[i]}", 18}");

                Console.Write($" {$"c = {c[i]}", 18}");
                Console.WriteLine($" {$"d = {d[i]}", 18}");
            }

            CSplinePolynomError(x, a, b, c, d, n);

            Console.WriteLine();
        }
    }
```

Polynom.cs

```
namespace Lab4;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("--- Kovalov Alex, TP-12 ---");
        Console.WriteLine("        -- Lab 4 --      \n");

        Console.WriteLine(" Function: y(x) = sin(x) + (2*x)^(1/3)");
        Console.WriteLine(" Interpolation nodes: -4, -2, 0, 2, 4");

        Console.WriteLine();

        int n = 5;
```

```
        var x = new double[n];
        var y = new double[n];
        var step = -4.0;
        for (int i = 0; i < n; i++) {
            x[i] = step;
            step += 2;
            y[i] = Math.Sin(x[i]) + Math.Cbrt(2 * x[i]);
        }

        Calculations.NewtonsPolynom(x, y, n);
        Calculations.CSpline(x, y, n);
    }
}
```