ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. О. ГОНЧАРА
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ ТА МАТЕМАТИЧНОЇ КІБЕРНЕТИКИ

Лабораторна робота №2

на тему «Наближення функцій алгебраїчними многочленами»

з курсу «Методи обчислень»

Варіант № 7

Виконав:

студент групи ПА-19-2

Ільяшенко Єгор

Дніпро, 2021

Зміст

Основні теоретичні відомості

Постановка задачі інтерполяції

Найпростіша задача інтерполяції (або інтерполювання) формулюється так. На проміжку [a, b] задані $(n+1)$ різних між собою точок $x_0, x_1, \dots, x_n$, які називаються вузлами інтерполяції, і відомі значення функції $f(x)$ у цих точках:

$$f(x_0) = y_0, f(x_1) = y_1, \dots, f(x_n) = y_n.$$

Потрібно побудувати функцію $\varphi(x)$ ***(інтерполяційну функцію)***, яка б належала певному класу функцій і набувала у вузлах інтерполяції таких самих значень, як і $f(x)$, тобто таку, що

$$\varphi(x_i) = y_i, i = \overline{0,n} \qquad (4.1)$$

Геометрично це означає, що треба побудувати криву $y = \varphi(x)$ якогось визначеного типу, котра проходить через задану систему точок $M_i(x_i, y_i), i = \overline{0,n}$ (рис. 4.1). Як видно з рис. 4.1, таких функцій можна побудувати багато.

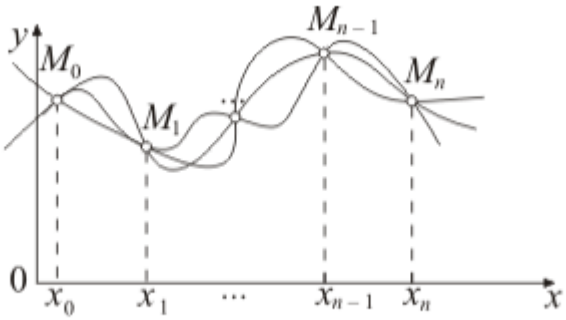


Рис. 4.1. Графіки функцій, що задовольняють умови інтерполювання

У вузлах інтерполяції значення функцій $f(x)$ і $\varphi(x)$ однакові. Але коли x не збігається з жодним вузлом, розбіжність між значеннями цих функцій може бути досить значною, навіть якщо вузлів багато і вони мало віддалені один від одного. Так може статися, наприклад, якщо $f(x)$ сильно звивиста функція і навіть розривна, а $\varphi(x)$ має високий ступінь гладкості. Для того, щоб мати надію одержати задовільний збіг $f(x)$ і $\varphi(x)$ на всьому проміжку [a, b], слід вибір класу інтерполяційних функцій узгоджувати з властивостями вихідної функції $f(x)$. Наприклад, достатньо гладку функцію $f(x)$ вдається 2 добре наблизити алгебраїчними многочленами. Якщо $f(x)$ періодична, то інтерполяційну функцію доцільно шукати серед тригонометричних многочленів з таким самим періодом.

На практиці до задачі інтерполювання звертаються, якщо потрібно багатократно обчислювати одну і ту саму складну функцію в різних точках. У цьому випадку доцільно обчислити раз назавжди її значення за фіксованими точками $x_0, x_1, \dots, x_n$, а в інших точках обчислювати її наближені значення, використовуючи більш просту інтерполяційну функцію. При цьому розрізняють інтерполяцію у вузькому сенсі, коли

$x \in [x_0, x_n]$, та екстраполяцію, коли $x \notin [x_0, x_n]$. Далі, вживаючи термін «інтерполяція», будемо розуміти обидва ці випадки.

Інтерполяція алгебраїчними многочленами

Будемо шукати функцію $\varphi(x)$ у вигляді алгебраїчного многочлена

$$\varphi(x) = \sum_{i=0}^{m} a_i * x^i, \qquad (4.2)$$

тобто у вигляді лінійної комбінації функцій $x^i, i = \overline{0, m}$ з невідомими коефіцієнтами $a_i, i = \overline{0, m}$. Підставимо (4.2) до умов інтерполювання (4.1) і отримаємо СЛАР відносно шуканих коефіцієнтів $a_i, i = \overline{0, m}$.

$$\varphi(x_j) = \sum_{i=0}^{m} a_i * x_j^i = f(x_j), j = \overline{0, n} \qquad (4.3)$$

СЛАР (4.3) має (n+1) рівнянь та (m+1) невідомих. Розглянемо можливі варіанти:

1) m>n, тобто невідомих коефіцієнтів у системі (4.3) більше, ніж рівнянь. СЛАР (4.3) має більше ніж один розв'язок. У цьому випадку серед шуканих коефіцієнтів $a_0, a_1, \ldots, a_m$ можна виділити m+1– r (r – ранг матриці A цієї системи) вільних коефіцієнтів, даючи яким довільні значення, можна відшукувати вже єдині значення решти невідомих коефіцієнтів.

2) m<n, тобто невідомих коефіцієнтів у системі (4.3) менше, ніж рівнянь. У цьому випадку система (4.3) може не мати розв'язку.

3) m=n. Система (4.3) має єдиний розв'язок, якщо визначник її матриці A відрізняється від нуля. Матриця системи (4.3) має вигляд

$$A = \begin{bmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{bmatrix}$$

Визначник цієї матриці є відомим в алгебрі визначником Вандермонда. Він відрізняється від нуля, якщо всі вузли різні.

Отже, надалі приймаємо, що в задачі інтерполювання всі вузли різні та **m** = **n**, тобто степінь алгебраїчного многочлена (4.2) на одиницю менший ніж кількість вузлів. У цьому випадку лінійна ***задача інтерполювання має єдиний розв'язок.***

На практиці інтерполяційні многочлени будують використовуючи прості алгебраїчні міркування, які не вимагають розв'язування СЛАР (4.3). Залежно від того,

для чого призначається інтерполяційний многочлен, його можна записати в різних формах.

Інтерполяційна формула Лагранжа. Залишок

Функцію $\varphi(x)$ шукаємо у вигляді

$$\varphi(x) = \sum_{i=0}^{n} y_i * Q_n^{(i)}(x), \qquad (4.4)$$

де $Q_n^{(i)}(x)$ - алгебраїчний многочлен степеня n, побудований для вузла x_i.

Функція (4.4) буде задовольняти умови (4.1), якщо алгебраїчні поліноми $Q_n^{(i)}(x), i = \overline{0,n}$ мають такі властивості

$$Q_n^{(i)}(x_j) = \begin{cases} 0, & j \neq i, \\ 1, & j = i. \end{cases}$$

Оскільки $Q_n^{(i)}(x)$ - многочлен степеня n з нулями в точках $x_0, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n$ то його можна подати у вигляді

$$Q_n^{(i)}(x) = C_i \prod_{\substack{m=0 \\ m \neq i}}^{n} (x - x_m) \qquad (4.5)$$

З умови $Q_n^{(i)}(x_i) = 1$ дістанемо $C_i = \frac{1}{\prod_{\substack{m=0 \\ m \neq i}}^{n} (x_i - x_m)}$. Тепер поліном (4.5) набуває вигляду

$$Q_n^{(i)}(x) = \prod_{\substack{m=0 \\ m \neq i}}^{n} \frac{x - x_m}{x_i - x_m} \qquad (4.6)$$

Підставивши (4.6) в (4.4), добудемо формулу для інтерполяційного полінома, яка носить ім'я Лагранжа і позначається $L_n(x)$

$$\varphi(x) = \sum_{i=0}^{n} y_i \prod_{\substack{m=0 \\ m \neq i}}^{n} \frac{x - x_m}{x_i - x_m} \equiv L_n(x) \qquad (4.7)$$

Запишемо многочлен Лагранжа трохи в іншому вигляді. Для цього впровадимо функцію $\omega_{n+1}(x) = \prod_{m=0}^{n}(x - x_m)$, обчислимо її похідну $\omega_{n+1}(x) = \sum_{k=0}^{n}\prod_{\substack{m=0 \\ m\neq k}}^{n}(x - x_m)$ та знайдемо значення похідної в точці x_i

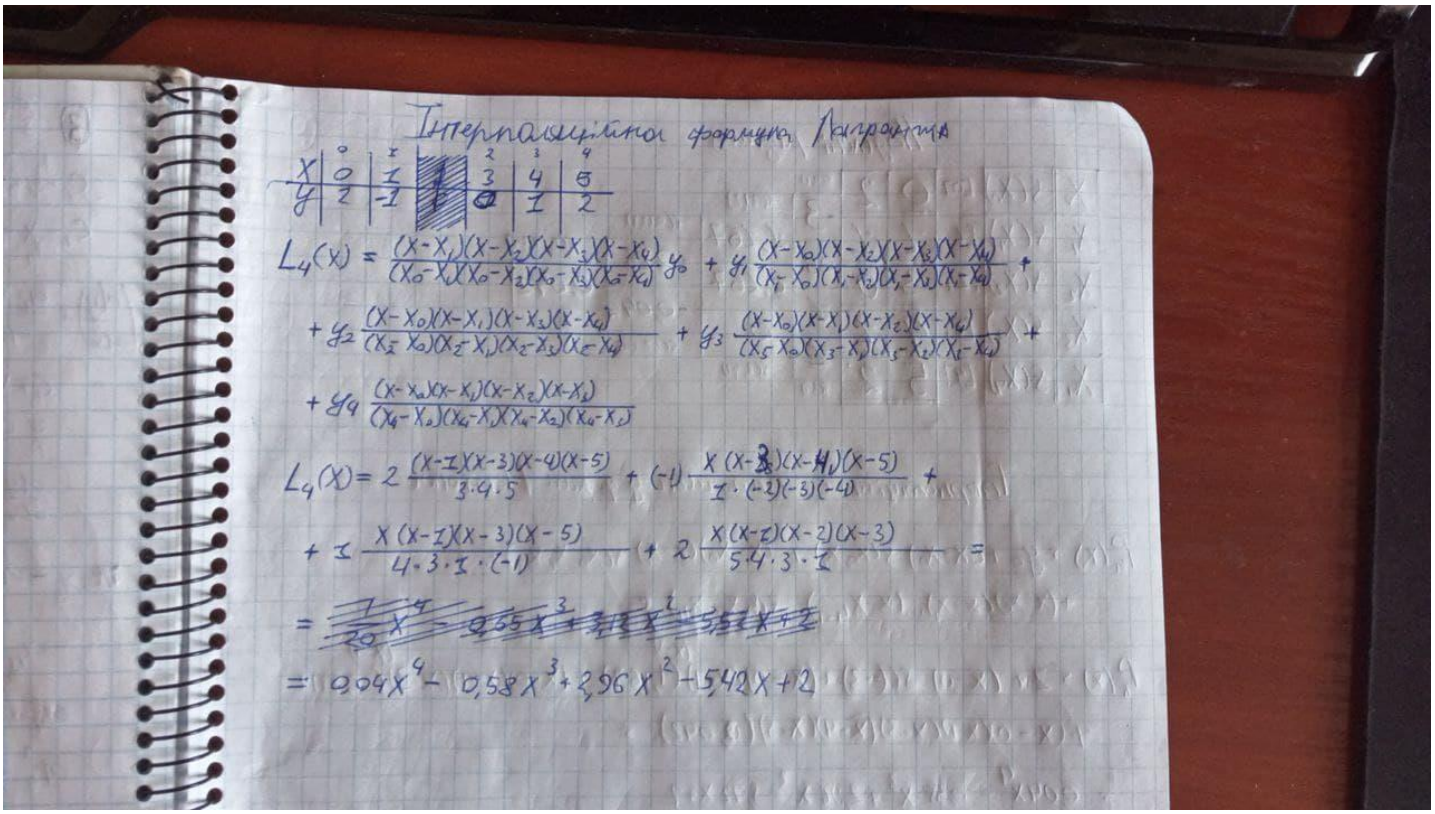
$$\omega_{n+1}(x_i) = \prod_{\substack{m=0 \\ m\neq i}}^{n}(x_i - x_m)$$

Тепер можна записати

$$\prod_{\substack{m=0 \\ m\neq i}}^{n}\frac{x - x_m}{x_i - x_m} = \frac{\omega_{n+1}(x)}{(x - x_i)\omega_{n+1}(x_i)}$$

Отже, інтерполяційний многочлен у формі Лагранжа набуває такого вигляду

$$L_n(x) = \sum_{i=0}^{n} f(x_i)\frac{\omega_{n+1}(x)}{(x - x_i)\omega_{n+1}(x_i)}$$



Поділені різниці та їх властивості

Нехай відомі різні між собою значення аргументу $x_0, x_1, \dots, x_n$ та відповідні значення функції $f(x_0), f(x_1), \dots, f(x_n)$. Для цих значень функції і вузлів обчислимо відношення

$$f(x_0, x_1) = \frac{f(x_1) - f(x_0)}{x_1 - x_0}, f(x_1, x_2) = \frac{f(x_2) - f(x_1)}{x_2 - x_1}, \dots,$$
$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}, \dots, f(x_{n-1}, x_n) = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Ці відношення називаються поділеними (розділеними) різницями першого порядку. Вони визначають середню швидкість зміни функції $f(x)$ (або середнє значення її похідної) на відповідному відрізку $[x_i, x_{i+1}], i = 0,1, \dots, n-1$.

Поділені різниці другого порядку, побудовані за вузлами x_i, x_{i+1}, x_{i+2}, мають вигляд

$$f(x_i, x_{i+1}, x_{i+2}) = \frac{f(x_{i+1}, x_{i+2}) - f(x_i, x_{i+1})}{x_{i+2} - x_i}.$$

Якщо поділені різниці k-го порядку $f(x_i, x_{i+1}, \dots, x_{i+k})$ вже відомі, то поділені різниці (k+1) -го порядку обчислюються за допомогою формули

$$f(x_i, x_{i+1}, \dots, x_{i+k+1}) = \frac{f(x_{i+1}, \dots, x_{i+k+1}) - f(x_i, \dots, x_{i+k})}{x_{i+k+1} - x_i}$$

При практичній побудові поділених різниць їх зручно розташовувати у вигляді таблиці (табл. 4.1). Така таблиця називається ***таблицею поділених різниць***.

Таблиця 4.1

Схема розташування поділених різниць

x_0	$f(x_0)$				
x_1	$f(x_1)$	$f(x_0, x_1)$			
x_2	$f(x_2)$	$f(x_1, x_2)$	$f(x_0, x_1, x_2)$		
...	...	...	...	...	$f(x_0, x_1, \dots, x_n)$
x_{n-1}	$f(x_{n-1})$	$f(x_{n-2}, x_{n-1})$	$f(x_{n-2}, x_{n-1}, x_n)$		
x_n	$f(x_n)$	$f(x_{n-1}, x_n)$			

Відмітимо деякі ***властивості поділених різниць***.

1 Поділену різницю k-го порядку можна виразити через значення функції у вузлах за такою формулою

$$f(x_i, x_{i+1}, \dots, x_{i+k}) = \sum_{j=i}^{i+k} \frac{f(x_j)}{\prod_{\substack{m=i \\ m\neq j}}^{i+k}(x_j - x_m)}. \qquad (4.15)$$

Доведемо цю формулу методом математичної індукції. При $k = 1$ формула правильна, оскільки

$$f(x_i, x_{i+1}) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{f(x_i)}{x_i - x_{i+1}} + \frac{f(x_{i+1})}{x_{i+1} - x_i} = \sum_{j=i}^{i+1} \frac{f(x_j)}{\prod_{\substack{m=i \\ m\neq j}}^{i+1}(x_j - x_m)}$$

Припустимо, що формула (4.15) є правильною при $k = \ell - 1$ і доведемо її правильність при $k = \ell$. Запишемо поділену різницю ℓ-го порядку

$$f(x_i, x_{i+1}, \dots, x_{i+\ell}) = \frac{f\left(x_{i+1,\dots,}x_{i+\ell}\right) - f(x_i, x_{i+1}, \dots, x_{i+\ell-1})}{x_{i+\ell} - x_i} =$$

$$= \frac{1}{x_{i+\ell} - x_i}\left(\sum_{j=i+1}^{i+\ell} \frac{f(x_j)}{\prod_{\substack{m=i+1 \\ m\neq j}}^{i+\ell}(x_j - x_m)} - \sum_{j=i}^{i+\ell-1} \frac{f(x_j)}{\prod_{\substack{m=i \\ m\neq j}}^{i+\ell-1}(x_j - x_m)} \right)$$

В останній формулі при $j = i$ та $j = i + \ell$ відповідні доданки з $f(x_j)$ зустрічаються один раз, причому у потрібному вигляді. При всіх інших значеннях j доданки з $f(x_j)$ зустрічаються двічі. Об'єднаємо ці доданки парами

$$\frac{1}{x_{i+\ell} - x_i} = \left(\frac{f(x_j)}{\prod_{\substack{m=i+1 \\ m\neq j}}^{i+\ell}(x_j - x_m)} - \frac{f(x_j)}{\prod_{\substack{m=i \\ m\neq j}}^{i+\ell-1}(x_j - x_m)} \right) =$$

$$= \frac{1}{x_{i+\ell} - x_i} * \frac{f(x_j)}{\prod_{\substack{m=i+1 \\ m\neq j}}^{i+\ell-1}(x_j - x_m)} \left(\frac{1}{x_j - x_{i+\ell}} - \frac{1}{x_j - x_i} \right) =$$

$$= \frac{f(x_j)}{\prod_{\substack{m=i+1 \\ m\neq j}}^{i+\ell-1}(x_j - x_m)} * \frac{1}{(x_j - x_{i+\ell})(x_j - x_i)} = \frac{f(x_j)}{\prod_{\substack{m=i \\ m\neq j}}^{i+\ell}(x_j - x_m)}$$

Остаточно маємо

$$f(x_i, x_{i+1}, \dots, x_{i+\ell}) = \frac{f(x_i)}{\prod_{\substack{m=i \\ m\neq i}}^{i+\ell}(x_i - x_m)} + \frac{f(x_{i+\ell})}{\prod_{\substack{m=i \\ m\neq j}}^{i+\ell}(x_{i+\ell} - x_m)} + \sum_{j=i+1}^{i+\ell-1} \frac{f(x_j)}{\prod_{\substack{m=i \\ m\neq j}}^{i+\ell}(x_j - x_m)} =$$

$$= \sum_{j=i}^{i+\ell} \frac{f(x_j)}{\prod_{\substack{m=i \\ m \neq j}}^{i+\ell} (x_j - x_m)}$$

Формула (4.15) доведена

Поділена різниця будь-якого порядку є лінійним оператором від $f(x)$. Дійсно, нехай $f(x) = \alpha f_1(x) + \beta f_2(x)$, тоді

$$f(x_i, x_{i+1}, \dots, x_{i+k}) = \sum_{j=i}^{i+k} \frac{f(x_j)}{\prod_{\substack{m=i \\ m \neq j}}^{i+k} (x_j - x_m)}$$

$$= \alpha \sum_{j=i}^{i+k} \frac{f_1(x_j)}{\prod_{\substack{m=i \\ m \neq j}}^{i+k} (x_j - x_m)} + \beta \sum_{j=i}^{i+k} \frac{f_2(x_j)}{\prod_{\substack{m=i \\ m \neq j}}^{i+k} (x_j - x_m)} =$$

$$= \alpha f_1(x_i, x_{i+1}, \dots, x_{i+k}) + \beta f_2(x_i, x_{i+1}, \dots, x_{i+k})$$

Поділена різниця є симетричною функцією своїх аргументів, тобто не змінюється при будь-якій перестановці аргументів. Дійсно, будь-яке переставляння аргументів у поділеній різниці приводить до зміни місць доданків у сумі та зміни місць множників у добутках формули (4.15).

Інтерполяційні формули Ньютона. Залишок

За допомогою поділених різниць можна одержати іншу форму запису інтерполяційного многочлена (4.7). Для цього на проміжку $[a, b]$ задамо функцію $f(x)$ її значеннями $f(x_i)$ у вузлах $x_i \in [a, b]$, $i = \overline{0, n}$ причому $x_i \neq x_j$ при $i \neq j$.

Нехай x – це довільна точка на проміжку $[a, b]$, причому $x \neq x_i, i = \overline{0, n}$. Розглянемо поділену різницю першого порядку, побудовану за точками $\{x, x_0\}$

$$f(x, x_0) = \frac{f(x) - f(x_0)}{x - x_0}$$

Звідси знаходимо

$$f(x) = f(x_0) + (x - x_0) * f(x, x_0) \qquad (4.16)$$

Тут число $f(x_0) = P_0(x)$ є інтерполяційним многочленом нульового степеня, побудованим за одним вузлом $R_0(x) = (x - x_0)f(x, x_0)$ – похибка інтерполювання. Далі підключимо ще один вузол x1 та розглянемо множину точок $\{x, x_0, x_1\}$. За означенням поділеної різниці другого порядку маємо

$$f(x, x_0, x_1) = \frac{f(x, x_0) - f(x_0, x_1)}{x - x_1}$$

Звідси знаходимо

$$f(x, x_0) = f(x_0, x_1) + (x - x_1) * f(x, x_0, x_1)$$

Підставивши знайдений вираз в (4.16), маємо

$$f(x) = f(x_0) + (x - x_0) * f(x_0, x_1) + (x - x_0)(x - x_1) * f(x, x_0, x_1) \qquad (4.17)$$

Тут $P_1(x) = f(x_0) + (x - x_0) * f(x_0, x_1)$ - інтерполяційний поліном першого степеня, побудований за вузлами x_0, x_1; $R_1(x) = (x - x_0)(x - x_1) * f(x, x_0, x_1)$ - похибка інтерполювання.

Якщо порівняти $P_1(x)$ та $P_0(x)$, то бачимо, що

$$P_1(x) = P_0(x) + (x - x_0) * f(x_0, x_1),$$

тобто наступний поліном $P_1(x)$ дорівнює попередньому поліному $P_0(x)$ плюс доданок, що характеризує вплив нового вузла x_1.

Додамо ще один вузол x_2 і скористаємось формулою для поділеної різниці третього порядку

$$f(x, x_0, x_1, x_2) = \frac{f(x, x_0, x_1) - f(x_0, x_1, x_2)}{x - x_2}$$

Знайдемо звідси $f(x, x_0, x_1)$ та підставивши до формули (4.17), одержимо

$$f(x) = P_1(x) + (x - x_0)(x - x_1) * f(x_0, x_1, x_2) + (x - x_0)(x - x_1)(x - x_2) * f(x, x_0, x_1, x_2)$$

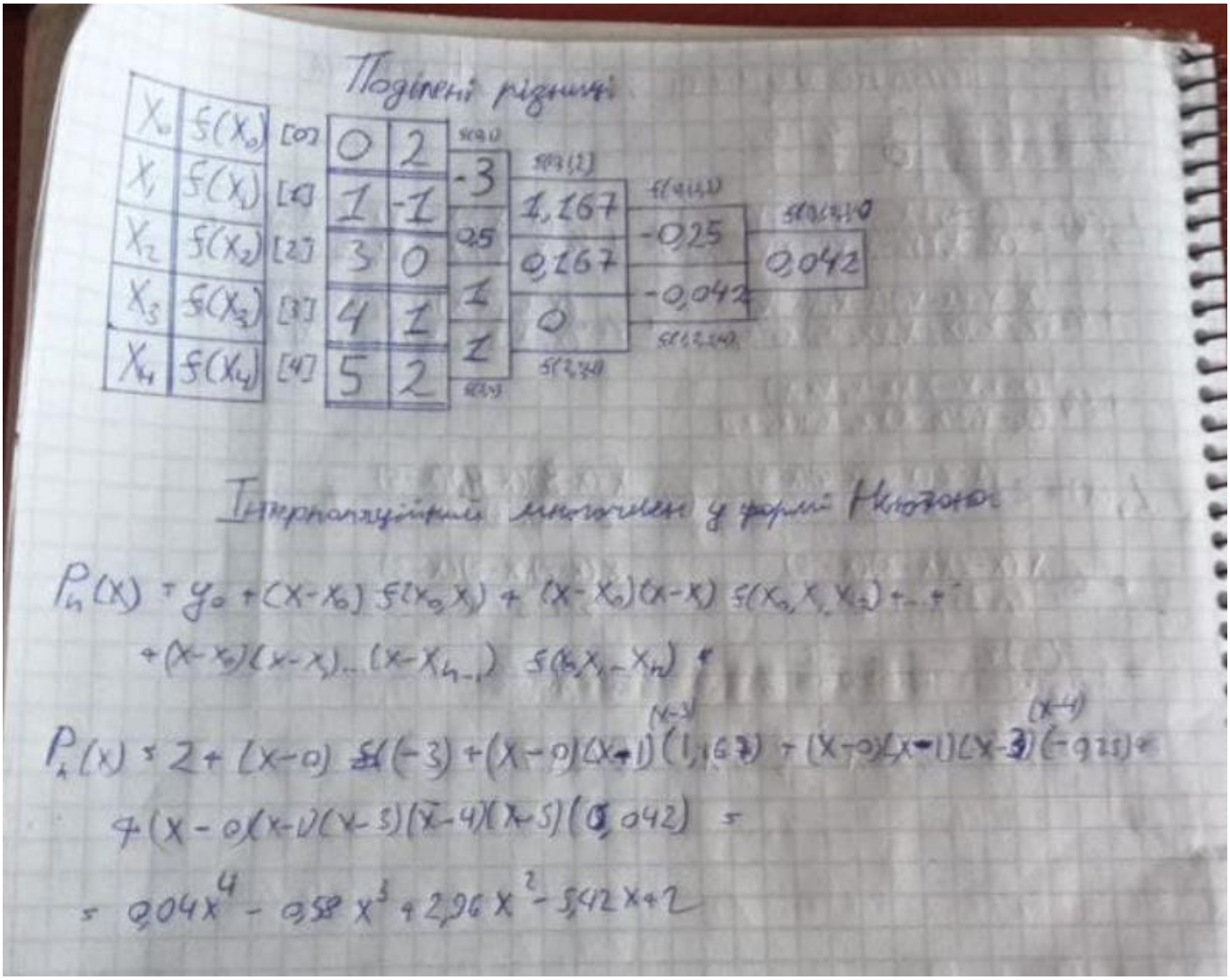
Методом математичної індукції добудемо формулу для інтерполяційного полінома, побудованого за вузлами $x_0, x_1, \dots, x_n$.

$$P_n(x) = f(x_0) + (x - x_0) * f(x_0, x_1) + (x - x_0)(x - x_1) * f(x_0, x_1, x_2) + \cdots + (x - x_0)(x - x_1) \dots (x - x_{n-1}) * f(x_0, x_1, \dots, x_n) \quad (4.18)$$

Похибка цієї інтерполяційної формули має вигляд

$$R_n(x) = \left(\prod_{m=0}^{n} (x - x_m) \right) * f(x, x_0, x_1, \dots, x_n) \quad (4.19)$$

Формулу (4.18) називають формулою ***Ньютона для інтерполяційного многочлена*** у випадку нерівних проміжків між вузлами.



Середньоквадратичне наближення функцій. Похибка

Постановка задачі середньоквадратичного наближення функцій

Задача інтерполювання вимагає від інтерполяційного многочлена $P_n(x)$, щоб він у вузлах інтерполяції x_i , i = 0,1, … , n точно дорівнював значенням $f(x_i)$. Якщо ці значення знайдені наближено (наприклад, з експерименту), то така вимога недоцільна. У цьому випадку кращим буде наближення функції не за точками, а в середньому.

Нехай задана функція $f(x)$, яку будемо наближувати іншою функцією $\varphi(x)$, що належить деякому класу Ф. Постановку задачі сформулюємо для двох випадків завдання функції $f(x)$.

Якщо функція $f(x) \in C[a,b]$, то функцію $\varphi(x) \in \Phi$ будемо шукати з умови, щоб інтеграл

$$J = \int_b^a (f(x) - \varphi(x))^2 dx \to min \qquad (1)$$

набував мінімального значення. Умова (1) означає, що функції $f(x)$ та $\varphi(x)$ на проміжку [a, b] у ***середньому близькі*** одна одній, хоча в окремих точках, або на деяких малих частинах проміжку [a, b], різниця між $f(x)$ та $\varphi(x)$ може бути досить великою.

Для функції $f(x)$, яка відома дискретно, тобто своїми значеннями $f(x_i)$ в точках x_i , i = 0,1, … , n , треба побудувати функцію $\varphi(x) \in \Phi$ так, щоб сума

$$S = \sum_{i=0}^{n} (f(x_i) - \varphi(x_i))^2 \to min \qquad (2)$$

набувала мінімального значення. Якщо припустити, що задані значення $f(x_i)$ мають випадкову похибку, то можна сподіватися, що значення, одержані в результаті апроксимації, будуть кращими ніж задані, тобто, середньоквадратичне наближення буде згладжувати локальні неправильності.

Наближення $\varphi(x)$, при якому інтеграл (1), або сума (2) набувають мінімальних значень, будемо називати найкращим середньоквадратичним наближенням, або наближенням за методом найменших квадратів (МНК).

Алгоритм побудови найкращого середньоквадратичного наближення (неперервний випадок)

Будуємо (або вибираємо) систему координатних (базисних) функцій $\{\psi_i(x)\}_{i=0}^{\infty}$, тобто систему функцій з такими властивостями:

Функції $\{\psi_i(x)\}_{i=0}^{\infty}$ такі, що $\sum_{i=0}^{n} C_i\psi_i(x) \in \Phi$ при будь-якому n = 0, 1, 2, … та при будь-яких числових коефіцієнтах $C_0, C_1, \dots, C_n$

при будь-якому натуральному m система функцій $\{\psi_i(x)\}_{i=0}^{m}$ лінійно незалежна. Це означає, що $\sum_{i=0}^{m} C_i\psi_i(x) = 0 \Leftrightarrow C_i = 0, i = \overline{0,m}$;

система функцій $\{\psi_i(x)\}_{i=0}^{\infty}$ є повною на множині функцій неперервних на відрізку $[a,b]$, тобто для будь-якої функції $g(x) \in C[a,b]$ та для $\forall\varepsilon>0$ знайдеться таке число n та такі коефіцієнти $C_0, C_1, \dots, C_n$, що буде виконуватись нерівність $\max\limits_{x\in[a,b]} |g(x) - \sum_{i=0}^{n} C_i\psi_i(x)| \le \varepsilon$. Іншими словами, будь-яку функцію $g(x) \in C[a,b]$ можна з будь-якою точністю ε наблизити виразом $\sum_{i=0}^{n} C_i\psi_i(x)$ для цього треба тільки відповідним чином вибрати коефіцієнти $C_0, C_1, \dots, C_n$ та їхню кількість.

Апроксимацію $\varphi(x)$ шукаємо у вигляді узагальненого многочлена m-го степеня

$$\varphi(x) = P_m(x) = \sum_{i=0}^{m} C_i\psi_i(x) \qquad (3)$$

Підставимо (3) до (1), інтеграл стане функцією коефіцієнтів$C_i, i = \overline{0,m}$. Виберемо ці коефіцієнти так, щоб інтеграл

$$\int_b^a (f(x) - \sum_{i=0}^{m} C_i\psi_i(x))^2 dx = J(C_0, C_1, \dots, C_m) \qquad (4)$$

набував мінімального значення. Як видно, значення функції (4) будуть невід’ємними. Крім того, функція (4) є поліномом другого степеня відносно своїх аргументів, а значить має одну точку екстремуму, яка є точкою мінімуму. Для того щоб знайти цю точку мінімуму, обчислимо частинні похідні від інтеграла (4) по всіх $C_k, k = \overline{0,m}$ і напишемо умови

$$\frac{\partial J}{\partial C_k} = 2\int_a^b \left(f(x) - \sum_{i=0}^{m} C_i\psi_i(x)\right) * \big(-\psi_k(x)\big)dx = 0, k = \overline{0,m}$$

Добуту систему рівнянь перепишемо у вигляді

$$\int_a^b f(x) * \psi_k(x)dx = \sum_{i=0}^{m} C_i \int_a^b \psi_i(x)\psi_k(x)\,dx,\ k = \overline{0,m} \qquad (5)$$

Позначимо

$$\int_a^b f(x) * \psi_k(x)dx = (f,\psi_k) = \beta_k,\ \int_a^b \psi_i(x) * \psi_k(x)dx = (\psi_i,\psi_k) = \alpha_{ik},$$

тоді система рівнянь (5) запишеться у вигляді

$$\sum_{i=0}^{m} \alpha_{ik} * C_i = \beta_k, k = \overline{0, m} \qquad (6)$$

Визначник СЛАР (6) є визначником Грама

$$\Delta = \begin{vmatrix} (\psi_0, \psi_0) & (\psi_1, \psi_0) & \dots & (\psi_m, \psi_0) \\ (\psi_0, \psi_1) & (\psi_1, \psi_1) & \vdots\vdots & (\psi_m, \psi_1) \\ \dots & \dots & & \dots \\ (\psi_0, \psi_m) & (\psi_1, \psi_m) & \dots & (\psi_m, \psi_m) \end{vmatrix} \qquad (7)$$

Він не дорівнює нулю, бо система функцій $\{\psi_i(x)\}_{i=0}^{m}$ лінійно незалежна. Це означає, що система (4.30) має єдиний розв'язок. Якщо позначити $C_0^*, C_1^*, \dots, C_m^*$ розв'язок цієї системи, то найкраще середньоквадратичне наближення запишеться у вигляді

$$\varphi(x) = \sum_{i=0}^{m} C_i^* \psi_i(x)$$

Середньоквадратичне відхилення (середня похибка) знаходиться за формулою

$$\delta = \sqrt{\frac{\int_b^a (f(x) - \sum_{i=0}^{m} C_i^* \psi_i(x))^2 dx}{b - a}}$$

Середня похибка δ може бути малим числом, але при цьому функції $f(x)$ і $\varphi(x)$ в окремих точках відрізка [a, b] можуть дуже відрізнятися одна від одної.

Алгоритм побудови найкращого середньоквадратичного наближення (дискретний випадок)

Будуємо систему координатних функцій $\{\psi_i(x)\}_{i=0}^{\infty}$, так, як у неперервному випадку.

Апроксимацію $\varphi(x)$ шукаємо у вигляді узагальненого многочлена

$\varphi(x) = \sum_{i=0}^{m} C_i \psi_i(x)$, причому вибираємо m<n.

Коефіцієнти $C_i, i = \overline{0, m}$ знайдемо так, щоб сума

$$\sum_{i=0}^{n} (f(x_i) - \sum_{j=0}^{m} C_j \psi_j(x_i))^2 = S(C_0, C_1, \dots, C_m) \qquad (8)$$

набувала мінімального значення. Обчислюємо частинні похідні від суми (8) по всіх $C_k, k = \overline{0, m}$ і напишемо необхідну умову мінімуму

$$\frac{\partial S}{\partial C_k} = 2\sum_{i=0}^{n}\left(f(x_i) - \sum_{j=0}^{m} C_j\psi_j(x_i)\right) * \left(-\psi_k(x_i)\right) = 0, k = \overline{0,m}$$

Добуту систему рівнянь перепишемо у вигляді

$$\sum_{i=0}^{n} f(x_i) * \psi_k(x_i) = \sum_{j=0}^{m} C_j \sum_{i=0}^{n} \psi_j(x_i)\psi_k(x_i), k = \overline{0,m} \qquad (9)$$

Позначимо

$$\sum_{i=0}^{n} f(x_i) * \psi_k(x_i) = (f, \psi_k) = \beta_k, \sum_{i=0}^{n} \psi_j(x_i) * \psi_k(x_i) = \left(\psi_j, \psi_k\right) = \alpha_{jk}$$

і напишемо систему рівнянь (9) у цих позначеннях

$$\sum_{j=0}^{m} \alpha_{jk} * C_j = \beta_k, k = \overline{0,m} \qquad (10)$$

Визначник СЛАР (10) є визначником Грама (7), він не дорівнює нулю, бо система функцій $\{\psi_i(x)\}_{i=0}^{m}$ лінійно незалежна.

Позначивши розв'язок системи (4.34) $C_0^*, C_1^*, \dots, C_m^*$, напишемо найкраще середньоквадратичне наближення

$$\varphi(x) = \sum_{i=0}^{m} C_i^*\psi_i(x)$$

Середньоквадратичне відхилення обчислюється за формулою

$$\delta = \sqrt{\frac{\sum_{i=0}^{n}(f(x_i) - \sum_{j=0}^{m} C_j^*\psi_j(x_i))^2}{n+1}}$$

При великих n степінь m вибирають, як правило, значно меншим за n. Якщо вибрати m = n, то поліном середньоквадратичного наближення стане інтерполяційним.

Середньоквадратична функція

$y = ax + b$ (степінь 2) ($\leq n$) n - кількість точок

$$\begin{cases} a\sum_{i=1}^{n} x_i + bn = \sum_{i=1}^{n} y_i \\ a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i \end{cases}$$

$$\begin{cases} 13a + 5b = 4 \\ 51a + 13b = 13 \end{cases}$$

По правилу Крамера

$$\begin{vmatrix} 13 & 5 \\ 51 & 13 \end{vmatrix} = -86 \neq 0 \Rightarrow \text{один розв'язок}$$

$$\left(\begin{array}{cc|c} 13 & 5 & 4 \\ 51 & 13 & 13 \end{array}\right)$$

1) $\begin{pmatrix} 4 & 5 \\ 13 & 13 \end{pmatrix} = 4 \cdot 13 - 5 \cdot 13 = -13$

2) $\begin{pmatrix} 13 & 4 \\ 51 & 13 \end{pmatrix} = -35$

$a = \frac{\Delta_1}{\Delta} = \frac{-13}{-86} \approx 0{,}15$

$b = \frac{\Delta_2}{\Delta} = \frac{-35}{-86} \approx 0{,}4$

$y = 0{,}15x + 0{,}4 \quad [m = 2]$

Помилка.

$S = \sum_{i=1}^{n} (y_i - P(x_i))^2$

$S = (2 - 0{,}4)^2 + (-1 + 0{,}55)^2 + (-0{,}85)^2 + (1-1)^2 + (2 - 1{,}15)^2 =$

$= 2{,}56 + 0{,}2025 + 0{,}723 + 0 + 0{,}7225 =$

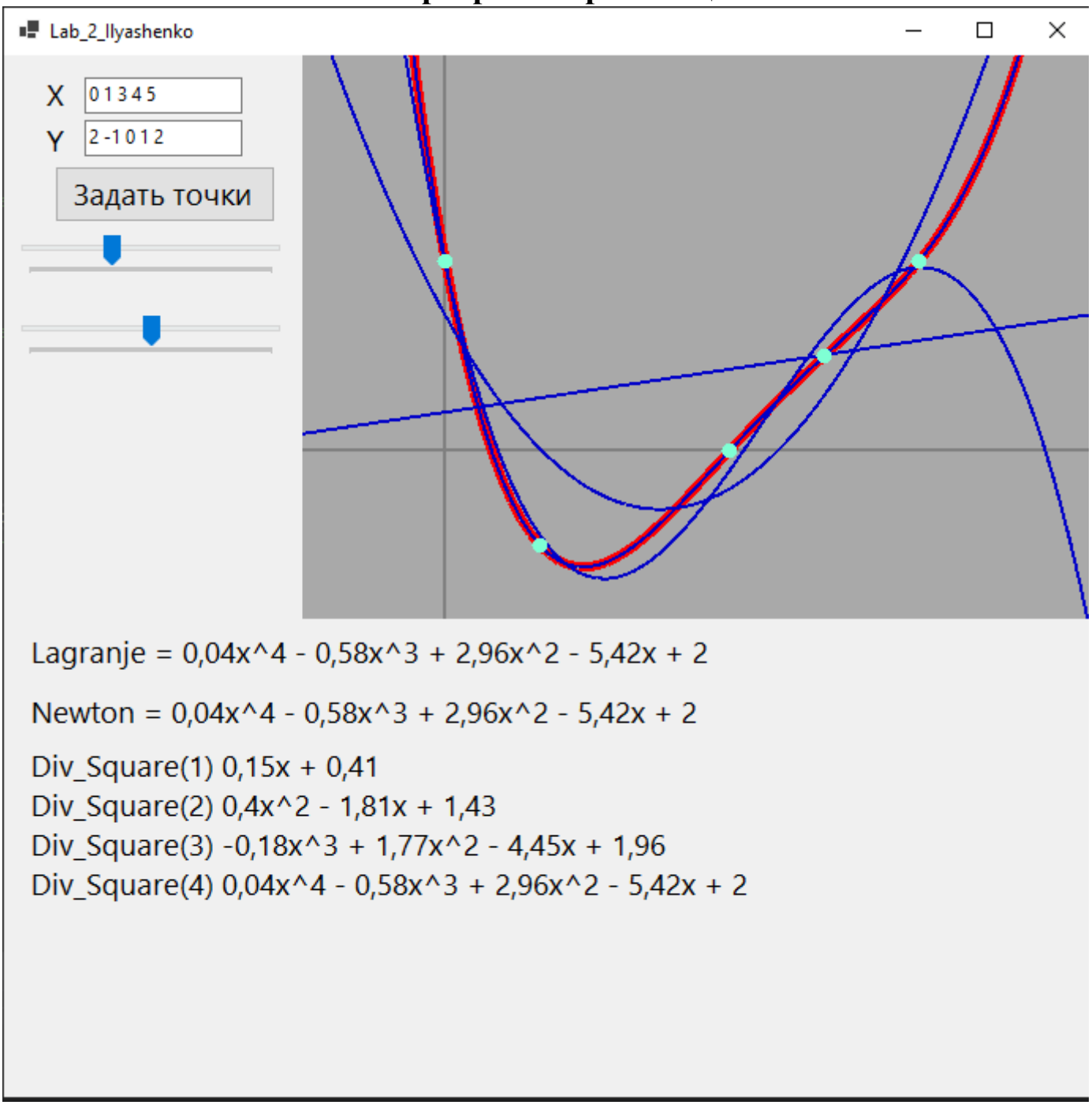
$= 4{,}209$

$\delta = \sqrt{\frac{S}{n+1}} = \frac{4{,}21}{4} = 1{,}05$

Чисельний експеримент та аналіз результатів

Для програмної реалізації усіх методів наближення функцій алгебраїчними многочленами, було обрано C# Windows Forms з бібліотекою Math.net-Numerics. Потрібно ввести таблицю значень функції та нажати на кнопку.

Опис програмної реалізації



Через пробіл задаються точки X і У. Ползулкамі можна зрушити графік. Червоним намальований поліном по функції Лагранжа і Ньютона, а синім - поліном побудований за методом розділених квадратів в ступеня M. (M <= N) (M> = 1)

Аналіз результатів

При інтерполюванні функції методами Лагранжа та Ньютона, отримані функції мають співпадати між собою. Більш того, функція, отримана методом розділених квадратів у степені (N-1), де N – кількість введених точок, теж повинна співпадати с двома іншими.

Висновки

Під час виконання лабораторної роботи номер 2 «Наближення функцій алгебраїчними многочленами» з курсу «Методи обчислень» я розглянув різні наближення функцій алгебраїчними многочленами (інтерполяційна формула Лагранжа, Ньютона та середньоквадратичне наближення функцій). Після ручних розрахунків була створена програма, в якій реалізовані усі вище вказані методи.

Перелік використаних джерел

1. Бойко Л.Т. Основи чисельних методів: навчальний посібник. – Д.: Вид-во ДНУ, 2009. – 244 с.
2. Шахно С.М. Практикум з чисельних методів: навч. посібник [Текст] / С.М. Шахно, А.Т. Дудикевич, С.М. Левицька. – Львів: ЛНУ імені Івана Франка. 2013. – 432 с.

Додаток. Код програми

```
using System;
using System.Collections.Generic;
using System.Text;
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics;
using MathNet.Numerics.Differentiation;

namespace Lab_2_Methods
{
    public class MyInput
    {
        public int[] x_input;
        public int[] y_input;
        public Polynomial lagranje;
        public Polynomial newton;
        public Polynomial div_square;
        public struct P_R
        {
            public double[] podil_rizn;
        }
        public P_R[] pod_r;

        public MyInput()
        {

        }
        public void AddPoints(string x, string y)
        {
            string[] splitted = x.Split(' ');
            x_input = new int[splitted.Length];
            for (int i = 0; i < splitted.Length; i++)
                x_input[i] = int.Parse(splitted[i]);

            splitted = y.Split(' ');
            y_input = new int[splitted.Length];
            for (int i = 0; i < splitted.Length; i++)
                y_input[i] = int.Parse(splitted[i]);
        }
        public void BuildLagranjePolynome()
        {
            lagranje = new Polynomial(0, 0);
            for (int i = 0; i < x_input.Length; i++)
            {
                Polynomial buf = new Polynomial(1, 0);
                buf *= y_input[i];
                for (int j = 0; j < x_input.Length; j++)
                {
                    if (j == i)
                        continue;

                    buf *= new Polynomial(-x_input[j], 1);
                    buf /= (x_input[i] - x_input[j]);
                }
                lagranje += buf;
            }
        }
        public void BuildNewtonPolynome()
        {
            pod_r = new P_R[x_input.Length - 1];
            int lader_count = x_input.Length - 1;
            for (int i = 0; i < pod_r.Length; i++)
            {
                pod_r[i].podil_rizn = new double[lader_count];
                lader_count--;
```

```
            }
            for (int i = 0; i < pod_r.Length; i++)
            {
                if (i == 0)
                {
                    for (int j = 0; j < pod_r[0].podil_rizn.Length; j++)
                    {
                        pod_r[0].podil_rizn[j] = ((double)y_input[j + 1] - y_input[j]) /
((double)x_input[j+1] - x_input[j]);
                    }
                }
                else
                {
                    for (int j = 0; j < pod_r[i].podil_rizn.Length; j++)
                    {
                        pod_r[i].podil_rizn[j] = (pod_r[i-1].podil_rizn[j + 1] - pod_r[i-
1].podil_rizn[j]) / ((double)x_input[j + 1 + i] - x_input[j]);
                    }
                }
            }
            newton = new Polynomial(0, 0);
            Polynomial buf = new Polynomial(0, 0);
            Polynomial buf2;
            newton += y_input[0];
            for (int i = 0; i < x_input.Length - 1; i++)
            {
                buf = new Polynomial(1, 0);

                for (int j = 0; j <= i; j++)
                {
                    buf2 = new Polynomial(-x_input[j], 1);
                    buf *= buf2;
                }

                buf *= pod_r[i].podil_rizn[0];
                newton += buf;
            }
        }
        //public double[,] slar;
        public void BuildDivSquarePolynome(int power)
        {
            //Polynomial line;
            double[,] slar = new double[power+1, power+1];
            double[] right_part = new double[power + 1];
            double[] koef = new double[power + 1];

            for (int i = 0; i < power+1; i++)
            {
                for (int j = 0; j < power + 1; j++)
                {
                    double sum = 0;
                    for (int k = 0; k < x_input.Length; k++)
                    {
                        sum += Math.Pow(x_input[k], power + i - j);
                    }
                    slar[i, j] = sum;
                }
            }
            for (int i = 0; i < power + 1; i++)
            {
                double sum = 0;
                for (int k = 0; k < x_input.Length; k++)
                {
                    sum += y_input[k]*Math.Pow(x_input[k], i);
                }
                right_part[i] = sum;
```

```
    }


    Matrix<double> slar_matrix = Matrix<double>.Build.DenseOfArray(slar);
    double determinant = slar_matrix.Determinant();
    double[,] buf_slar = new double[power + 1, power + 1];
    for (int i = 0; i < power + 1; i++)
    {
        for (int b = 0; b < power + 1; b++)
            for (int g = 0; g < power + 1; g++)
            {
                buf_slar[b, g] = slar[b, g];
            }

        for (int j = 0; j < power + 1; j++)
        {
            buf_slar[j, i] = right_part[j];
        }
        Matrix<double> koef_find = Matrix<double>.Build.DenseOfArray(buf_slar);
        koef[i] = koef_find.Determinant() / determinant;
    }
    double[] koef_buf = new double[koef.Length];
    for (int i = 0; i < koef.Length; i++)
    {
        koef_buf[i] = koef[koef.Length - i - 1];
    }
    div_square = new Polynomial(koef_buf);
}
public string OutputNewton()
{
    string output = "";
    Polynomial out_p = newton.Clone();
    for (int i = 0; i < out_p.Coefficients.Length; i++)
        out_p.Coefficients[i] = Math.Round(out_p.Coefficients[i], 2);
    output = out_p.ToStringDescending();
    return output;
}
public string OutputLagranje()
{
    string output = "";
    Polynomial out_p = lagranje.Clone();
    for (int i = 0; i < out_p.Coefficients.Length; i++)
        out_p.Coefficients[i] = Math.Round(out_p.Coefficients[i], 2);
    output = out_p.ToStringDescending();
    return output;
}
public string OutputDivSquare()
{
    string output = "";
    Polynomial out_p = div_square.Clone();
    for (int i = 0; i < out_p.Coefficients.Length; i++)
        out_p.Coefficients[i] = Math.Round(out_p.Coefficients[i], 2);
    output = out_p.ToStringDescending();
    return output;
}
public (double, double) LagranjePoint(double x, int accuracy)
{
    return (x, Math.Round(lagranje.Evaluate(x), accuracy));
}
public (double, double) NewtonPoint(double x, int accuracy)
{
    return (x, Math.Round(newton.Evaluate(x), accuracy));
}
public (double, double) DivSquarePoint(double x, int accuracy)
{
    return (x, Math.Round(div_square.Evaluate(x), accuracy));
```

```
        }
    }
}
```

```
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 MathNet.Numerics;
using MathNet.Numerics.LinearAlgebra;

namespace Lab_2_Methods
{
    public partial class Lab_2_Methods_Form : Form
    {
        public MyInput input;
        public Lab_2_Methods_Form()
        {
            InitializeComponent();
            input = new MyInput();
            gr = pictureBox1.CreateGraphics();
            gr.Clear(Color.DarkGray);
        }
        Graphics gr;
        private void button1_Click(object sender, EventArgs e)
        {
            gr.Clear(Color.DarkGray);
            input.AddPoints(textBox_X.Text, textBox_Y.Text);

            input.BuildLagranjePolynome();
            label1.Text = "Lagranje = " + input.OutputLagranje();

            input.BuildNewtonPolynome();
            label2.Text = "Newton = " + input.OutputNewton();


            DrawMethods();
        }
        void DrawMethods()
        {

            PointF buf = new PointF(0 + 50, (float)input.LagranjePoint(0 + 50, 3).Item2 * 10);
            Pen pen = new Pen(Color.Red, 6);
            List<PointF> points = new List<PointF>();
            gr.DrawLine(new Pen(Color.Gray, 2), ToMap(new PointF(-500, 0)), ToMap(new PointF(500,
0)));
            gr.DrawLine(new Pen(Color.Gray, 2), ToMap(new PointF(0, 550)), ToMap(new PointF(0, -
500)));
            for (double i = -2; i < 10; i += 0.1)
            {
                points.Add(new PointF((float)i, -(float)input.LagranjePoint(i, 3).Item2));
                //gr.DrawEllipse(pen, new RectangleF((float)i*30+250, -(float)input.LagranjePoint(i,
3).Item2*30+250, 1, 1));
            }
            for (int i = 0; i < points.Count - 1; i++)
            {
```

```
                gr.DrawLine(pen, ToMap(points[i]), ToMap(points[i + 1]));
                //gr.DrawEllipse(pen, new RectangleF((float)i*30+250, -(float)input.LagranjePoint(i,
3).Item2*30+250, 1, 1));
            }

            Random rnd = new Random();
            label3.Text = "";
            for (int k = 1; k < input.x_input.Length; k++)
            {
                pen = new Pen(Color.FromArgb(0,0,200), 2);
                input.BuildDivSquarePolynome(k);
                points.Clear();
                for (double i = -2; i < 10; i += 0.1)
                {
                    points.Add(new PointF((float)i, -(float)input.DivSquarePoint(i, 3).Item2));
                    //gr.DrawEllipse(pen, new RectangleF((float)i*30+250, -
(float)input.LagranjePoint(i, 3).Item2*30+250, 1, 1));
                }
                for (int i = 0; i < points.Count - 1; i++)
                {
                    gr.DrawLine(pen, ToMap(points[i]), ToMap(points[i + 1]));
                    //gr.DrawEllipse(pen, new RectangleF((float)i*30+250, -
(float)input.LagranjePoint(i, 3).Item2*30+250, 1, 1));
                }
                label3.Text += "Div_Square(" + k + ") " + input.OutputDivSquare() + "\n";
            }

            for (int i = 0; i < input.x_input.Length; i++)
            {
                gr.FillEllipse(new SolidBrush(Color.Aquamarine), new RectangleF(new PointF(ToMap(new
PointF(input.x_input[i], -input.y_input[i])).X-5, ToMap(new PointF(input.x_input[i], -
input.y_input[i])).Y-5), new SizeF(10, 10)));
                //gr.DrawLine(new Pen(Color.Aquamarine, 6), ToMap(new PointF(input.x_input[i], -
input.y_input[i])), ToMap(new PointF(input.x_input[i]+0.1f, -input.y_input[i]+0.1f)));
            }

        }
        PointF ToMap(PointF point)
        {
            point.X *= 60;
            point.Y *= 60;
            point.X += 250 + trackBar1.Value;
            point.Y += 250 + trackBar2.Value;
            return point;
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            gr.Clear(Color.DarkGray);
            DrawMethods();
        }

        private void trackBar2_Scroll(object sender, EventArgs e)
        {
            gr.Clear(Color.DarkGray);
            DrawMethods();
        }
    }
}
```