МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА «ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ ІМЕНІ А. В. ДАБАГЯНА»

ЗВІТ

З спринту №3

«ІНТЕРПОЛЯЦІЯ»

ВИКОНАВ:

студент групи КН-422ч

Максим БЕЛОШИЦЬКИЙ

ПЕРЕВІРИВ:

Завідувач каф. ІСтТ

Олена НІКУЛІНА

Харків – 2023

**ЗМІСТ**

[МЕТА 2](#_Toc167194382)

[Завдання 1.1 3](#_Toc167194383)

[Код завдання 1.1: 3](#_Toc167194384)

[lagrange\_interpolation.m 3](#_Toc167194385)

[Результати виконання програми 3](#_Toc167194386)

[Завдання 1.2 5](#_Toc167194387)

[Код завдання 1.2: 5](#_Toc167194388)

[newton\_interpolation.m 5](#_Toc167194389)

[Результати виконання програми 5](#_Toc167194390)

[Завдання 1.3 7](#_Toc167194391)

[Код завдання 1.3: 7](#_Toc167194392)

[differentiate\_interpolation.m 7](#_Toc167194393)

[Результати виконання програм 8](#_Toc167194394)

[Завдання 1.4 10](#_Toc167194395)

[Код завдання 1.4: 10](#_Toc167194396)

[approximate\_differentiation.m 10](#_Toc167194397)

[Результати виконання програми 11](#_Toc167194398)

[Завдання 1.5, 1.6 12](#_Toc167194399)

[Код завдання 1.5, 1.6: 12](#_Toc167194400)

[CalculatorOfInterpolations.mlapp 12](#_Toc167194401)

[Результати виконання програми 19](#_Toc167194402)

[ВИСНОВОК 21](#_Toc167194403)

# **МЕТА**

Мета спринту з чисельних методів полягає у вивченні та розумінні основних принципів чисельних методів, їх застосуванні для апроксимації та аналізу функцій. У ході цієї роботи студенти мають можливість ознайомлюватися з різними методами інтерполяції та диференціювання, що дозволяє їм здобувати уявлення про те, як ці методи використовуються для наближення функцій та обчислення похідних.

Окрім теоретичного освоєння методів, мета також включає практичну складову. Студенти отримують можливість реалізувати ці методи у вигляді програмного коду та перевіряти їх ефективність на конкретних прикладах. Крім того, створення програми з графічним інтерфейсом дозволяє студентам отримувати досвід роботи з реальними даними та візуалізацією результатів обчислень.

Отже, мета лабораторної роботи полягає у засвоєнні теоретичних знань та практичних навичок з чисельних методів, а також у розвитку вмінь аналізу, програмування та використання програмного забезпечення для чисельних обчислень.

# Завдання 1.1

Написати програму чисельного інтерполювання, який використовує інтерполяційну формулу Лагранжа, програму оформити у вигляді функції. Програму можна писати будь-якою мовою програмування.

Вхідні параметри: х – заданий набір значень аргументів, f – відповідний до х заданий набір значень функції, Х – набір значень аргументів, для якого знаходиться набір значень функції. Вихідні параметри: F – набір обчислених значень функції у точках, що інтерполюються.

## Код завдання 1.1:

### lagrange\_interpolation.m

function F = lagrange\_interpolation(x, f, X)

n = length(x);

m = length(X);

F = zeros(1, m);

for k = 1:m

L = ones(1, n);

for i = 1:n

for j = 1:n

if j ~= i

L(i) = L(i) \* (X(k) - x(j)) / (x(i) - x(j));

end

end

end

F(k) = sum(f .\* L);

end

end

## Результати виконання програми

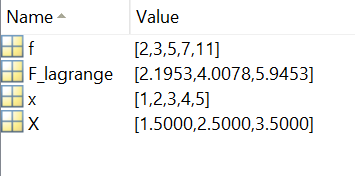


Рисунок 1.1 – «Розв’язок за методом Лагранжа»

Програма реалізує метод інтерполяції Лагранжа, який використовується для побудови інтерполяційного полінома, що проходить через задані точки. Інтерполяційний поліном Лагранжа є лінійною комбінацією базисних поліномів Лагранжа. Цей метод дозволяє знайти значення функції в будь-якій точці на основі заданих значень у деяких інших точках. Вхідними параметрами є вектори x, f та X. x містить координати точок, через які повинен пройти поліном, f містить відповідні значення функції в цих точках, а X містить точки, в яких потрібно обчислити значення інтерполяційного полінома. Вихідним параметром є вектор F, який містить обчислені значення полінома в точках з X.

На початку функції визначається кількість точок інтерполяції та кількість точок, де потрібно обчислити поліном. Вектор F ініціалізується нулями для зберігання результатів. Далі, для кожної точки з X, обчислюється значення полінома Лагранжа. Ініціалізується вектор базисних поліномів Лагранжа одиницями. Для кожної точки інтерполяції обчислюється базисний поліном. Для кожної іншої точки інтерполяції коригується значення базисного полінома, враховуючи відстань між точками. Після цього обчислюється значення полінома в точці X(k) як сума добутків значень функції на відповідні базисні поліноми.

Цей метод добре працює для невеликої кількості точок, але для великої кількості точок він може стати неефективним через зростання обчислювальної складності та можливість числової нестійкості. Для зменшення складності можна використовувати матрицю Вандермонда або інші більш ефективні методи інтерполяції для великих даних. Окрім цього, замість вкладених циклів можна використовувати векторизовані операції для підвищення ефективності. Важливо також додати перевірки для унеможливлення ділення на нуль або коригування для випадків, коли координати точок збігаються.

Загалом, ця функція демонструє базові принципи методу Лагранжа для інтерполяції, що є важливим інструментом у чисельному аналізі та наближенні функцій.

# Завдання 1.2

Написати програму чисельного інтерполювання, який використовує інтерполяційну формулу Ньютона, програму оформити у вигляді функції. Програму можна писати будь-якою мовою програмування.

Вхідні параметри: х – заданий набір значень аргументів, f – відповідний до х заданий набір значень функції, Х – набір значень аргументів, для якого знаходиться набір значень функції. Вихідні параметри: F – набір обчислених значень функції у точках, що інтерполюються.

## Код завдання 1.2:

### newton\_interpolation.m

function F = newton\_interpolation(x, f, X)

n = length(x);

divided\_diff = zeros(n, n);

divided\_diff(:, 1) = f(:);

for j = 2:n

for i = 1:(n-j+1)

divided\_diff(i, j) = (divided\_diff(i+1, j-1) - divided\_diff(i, j-1)) / (x(i+j-1) - x(i));

end

end

coefficients = divided\_diff(1, :);

m = length(X);

F = zeros(1, m);

for k = 1:m

P = coefficients(n);

for j = (n-1):-1:1

P = coefficients(j) + (X(k) - x(j)) \* P;

end

F(k) = P;

end

end

## Результати виконання програми

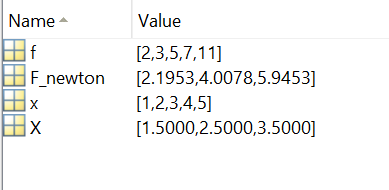


Рисунок 1.2 – «Обчислення за методом Ньютона»

Програма реалізує метод інтерполяції Ньютона, який використовується для побудови інтерполяційного полінома на основі заданих точок. Цей метод базується на використанні поділених різниць для побудови інтерполяційного полінома. Вхідними параметрами є вектори x, f та X. Вектор x містить координати точок, через які проходить поліном, f містить відповідні значення функції в цих точках, а X містить точки, в яких потрібно обчислити значення інтерполяційного полінома. Вихідним параметром є вектор F, який містить обчислені значення полінома в точках з X.

На початку функції визначається кількість точок інтерполяції n. Ініціалізується матриця поділених різниць divided\_diff нулями, після чого перший стовпець цієї матриці заповнюється значеннями функції f. Далі, для кожного наступного стовпця матриці обчислюються поділені різниці за рекурсивною формулою, де кожне значення поділеної різниці визначається як різниця між сусідніми значеннями попереднього стовпця, поділена на різницю відповідних значень з вектора x.

Після обчислення матриці поділених різниць перший рядок цієї матриці містить коефіцієнти інтерполяційного полінома. Коефіцієнти зберігаються у векторі coefficients. Далі визначається кількість точок, де потрібно обчислити поліном m, і ініціалізується вектор F нулями для зберігання результатів.

Для кожної точки з вектора X обчислюється значення полінома за методом Ньютона. Спочатку значення полінома P встановлюється рівним останньому коефіцієнту. Потім для кожного попереднього коефіцієнта поліном коригується, враховуючи відповідні точки з вектора x. Це дозволяє поступово побудувати значення полінома у вигляді, який відповідає заданим точкам. Обчислені значення зберігаються у векторі F.

Цей метод є ефективним для обчислення інтерполяційних поліномів, оскільки дозволяє легко додавати нові точки, не перераховуючи повністю всі коефіцієнти. Він також зручний для числової реалізації та має меншу обчислювальну складність у порівнянні з деякими іншими методами інтерполяції, наприклад, з методом Лагранжа.

Цей код демонструє важливий інструмент чисельного аналізу, який широко використовується для наближення функцій та розв'язання прикладних задач, де потрібне інтерполяційне наближення.

# Завдання 1.3

Написати програму диференціювання таблично заданої функції за допомогою інтерполяції для п’яти вузлів, програму оформити у вигляді функції. Програму можна писати будь-якою мовою програмування.

Вхідні параметри: х – заданий набір значень аргументів, f – відповідний до х заданий набір значень функції, Х – набір значень аргументів, для якого знаходиться набір значень похідної функції. Вихідні параметри: F – набір обчислених значень похідної функції у точках, що інтерполюються.

## Код завдання 1.3:

### differentiate\_interpolation.m

function F = differentiate\_interpolation(x, f, X)

n = length(x);

m = length(X);

F = zeros(1, m);

for k = 1:m

idx = find\_nearest\_index(x, X(k));

nodes = max(1, idx - 2):min(n, idx + 2);

coefficients = lagrange\_coefficients(x(nodes), f(nodes));

F(k) = differentiate\_polynomial(x(nodes), coefficients, X(k));

end

end

function idx = find\_nearest\_index(x, val)

[~, idx] = min(abs(x - val));

end

function coefficients = lagrange\_coefficients(x, f)

n = length(x);

coefficients = zeros(1, n);

for i = 1:n

numerator = 1;

denominator = 1;

for j = 1:n

if j ~= i

numerator = numerator \* (x(i) - x(j));

denominator = denominator \* (x(i) - x(j));

end

end

coefficients(i) = f(i) / numerator;

end

end

function F = differentiate\_polynomial(x, coefficients, X)

n = length(x);

m = length(X);

F = zeros(1, m);

for k = 1:m

P = 0;

for i = 1:n

prod = 1;

for j = 1:n

if j ~= i

prod = prod \* (X(k) - x(j));

end

end

P = P + coefficients(i) \* prod;

end

F(k) = P;

end

end

## Результати виконання програм

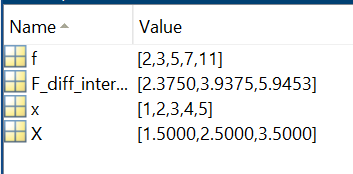


Рисунок 1.3 – «Обчислення differentiate\_interpolation.m»

Цей код реалізує метод диференціювання таблично заданої функції за допомогою інтерполяції для п’яти вузлів. Програма отримує три вхідні параметри: x – вектор значень аргументів, f – вектор значень функції, що відповідають значенням з x, і X – вектор значень аргументів, для яких необхідно знайти значення похідної функції. Вихідним параметром є вектор F, який містить обчислені значення похідної функції у точках з вектора X.

Спочатку програма визначає кількість вихідних точок n та кількість точок для обчислення похідної m. Вектор F ініціалізується нулями. Потім для кожної точки X(k) з вектора X виконується наступний алгоритм.

Програма знаходить найближчу вихідну точку до X(k) за допомогою функції find\_nearest\_index. Потім визначається інтервал з п'яти сусідніх точок, що включає найближчу точку. Індекси цих точок обираються таким чином, щоб уникнути виходу за межі вектора x.

Далі, використовуючи функцію lagrange\_coefficients, обчислюються коефіцієнти інтерполяційного полінома Лагранжа на основі п'яти обраних точок. Ця функція обчислює коефіцієнти, які використовуються для побудови полінома, що проходить через задані точки.

Після цього, функція differentiate\_polynomial обчислює похідну інтерполяційного полінома в точці X(k). Ця функція використовує обчислені коефіцієнти та обрані точки для знаходження значення похідної.

Підсумовуючи, метод розв'язку, представлений у цьому коді, складається з таких основних етапів:

1. Знаходження найближчої вихідної точки до кожної точки X(k).
2. Вибір інтервалу з п'яти сусідніх точок.
3. Обчислення коефіцієнтів інтерполяційного полінома Лагранжа для цих точок.
4. Обчислення похідної інтерполяційного полінома в точці X(k).

Цей метод дозволяє ефективно знаходити значення похідної функції у точках, що знаходяться між заданими вихідними точками. Він є корисним у випадках, коли аналітичне обчислення похідної є складним або неможливим, а також дозволяє отримати наближене значення похідної з високою точністю.

# Початок форми

# Завдання 1.4

Написати програму диференціювання таблично заданої функції за допомогою інтерполяції для п’яти вузлів, програму оформити у вигляді функції. Програму можна писати будь-якою мовою програмування.

Вхідні параметри: х – заданий набір значень аргументів, f – відповідний до х заданий набір значень функції, Х – набір значень аргументів, для якого знаходиться набір значень похідної функції. Вихідні параметри: F – набір обчислених значень похідної функції у точках, що апроксимуються.

## Код завдання 1.4:

### approximate\_differentiation.m

function F = approximate\_differentiation(x, f, X)

n = length(x);

m = length(X);

F = zeros(1, m);

% Check if x is uniformly spaced

if any(diff(x) ~= x(2) - x(1))

error('x must be uniformly spaced');

end

h = x(2) - x(1); % Step size, assuming uniform spacing

for k = 1:m

% Find the index of the closest node to X(k)

[~, idx] = min(abs(x - X(k)));

% Adjust index to ensure it's not too close to the edges

if idx < 3

idx = 3;

elseif idx > n - 2

idx = n - 2;

end

% Select five nodes around the closest index

nodes = idx-2:idx+2;

xx = x(nodes);

ff = f(nodes);

% Fit a polynomial of degree 4 to the selected nodes

poly = polyfit(xx, ff, 4);

% Differentiate the polynomial

dpoly = polyder(poly);

% Evaluate the derivative at X(k)

F(k) = polyval(dpoly, X(k));

end

end

## Результати виконання програми

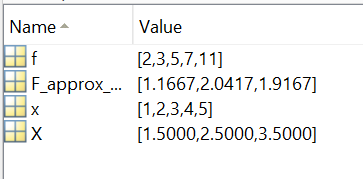


Рисунок 1.4 – «Обчислення approximate\_differentiation»

Цей метод для обчислення похідних функцій на основі інтерполяції використовує інтерполяцію поліномом в малих областях, що оточують точки, в яких потрібно знайти похідну.

Спочатку для кожної точки, в якій потрібно обчислити похідну, знаходиться найближчий вузол у вхідному наборі точок. Потім вибирається п'ять вузлів навколо цього найближчого вузла, щоб створити малий інтервал, де буде проводитися інтерполяція.

Далі виконується інтерполяція поліномом четвертого ступеня через ці п'ять вузлів. Поліном обчислюється за допомогою методу найменших квадратів.

Після отримання полінома обчислюється його похідна, яка відповідає похідній функції в даній точці.

Цей підхід дозволяє обчислити наближені значення похідних в точках, де немає аналітичного виразу для похідної, використовуючи інтерполяцію поліномом через набір вузлів.

# Завдання 1.5, 1.6

Реалізувати графічний інтерфейс користувача програмного продукту. Програмний продукт призначений для вирішення двох завдань, тому при запуску програми користувач вибирає - інтерполювання та диференціювання для таблично заданих функцій. Передбачити можливість вибору будь-якого методу для вирішення обраного завдання, а також читання початкових даних із файлів. Файли формує сам розробник програмного продукту.

Доопрацювати та протестувати розроблений програмний продукт. Для розв’язання задач передбачити як табличне вивидення результатів, так і графічний. На графіку будуються як початкове значення таблично заданої функції, так і отриманий результат. Надати зведену таблицю з результатами обчислень для методів для однієї заданої функції. Проаналізувати отримані результати. Зробити висновки щодо роботи методів.

## Код завдання 1.5, 1.6:

### CalculatorOfInterpolations.mlapp

classdef CalculatorOfInterpolations < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

UIFigure matlab.ui.Figure

XEditField matlab.ui.control.EditField

XEditFieldLabel matlab.ui.control.Label

fEditField matlab.ui.control.EditField

fEditFieldLabel matlab.ui.control.Label

xEditField matlab.ui.control.EditField

xEditField\_2Label matlab.ui.control.Label

UITable\_2 matlab.ui.control.Table

UITable matlab.ui.control.Table

SolveButton matlab.ui.control.Button

InputFileButton matlab.ui.control.Button

MethodDropDown matlab.ui.control.DropDown

MethodDropDownLabel matlab.ui.control.Label

OperationsButtonGroup matlab.ui.container.ButtonGroup

DifferentiationButton matlab.ui.control.RadioButton

InterpolationButton matlab.ui.control.RadioButton

UIAxes matlab.ui.control.UIAxes

end

properties (Access = private)

xData double = []

fData double = []

XData double = NaN

end

methods (Access = private)

% Lagrange interpolation function

function F = lagrange\_interpolation(~, x, f, X)

n = length(x);

F = zeros(size(X));

for k = 1:length(X)

L = ones(n, 1);

for i = 1:n

for j = 1:n

if i ~= j

L(i) = L(i) \* (X(k) - x(j)) / (x(i) - x(j));

end

end

end

F(k) = sum(f .\* L');

end

end

% Newton interpolation function

function F = newton\_interpolation(~, x, f, X)

n = length(x);

b = zeros(n, n);

b(:,1) = f(:);

for j = 2:n

for i = j:n

b(i,j) = (b(i,j-1) - b(i-1,j-1)) / (x(i) - x(i-j+1));

end

end

F = zeros(size(X));

for k = 1:length(X)

F(k) = b(n,n);

for j = n-1:-1:1

F(k) = F(k) \* (X(k) - x(j)) + b(j,j);

end

end

end

% Differentiation using interpolation function for five nodes

function F = differentiate\_interpolation(app, x, f, X)

n = length(x);

m = length(X);

F = zeros(1, m);

for k = 1:m

idx = app.find\_nearest\_index(x, X(k));

nodes = max(1, idx - 2):min(n, idx + 2);

coefficients = app.lagrange\_coefficients(x(nodes), f(nodes));

F(k) = app.differentiate\_polynomial(x(nodes), coefficients, X(k));

end

end

function idx = find\_nearest\_index(~, x, val)

[~, idx] = min(abs(x - val));

end

function coefficients = lagrange\_coefficients(~, x, f)

n = length(x);

coefficients = zeros(1, n);

for i = 1:n

numerator = 1;

denominator = 1;

for j = 1:n

if j ~= i

numerator = numerator \* (x(i) - x(j));

denominator = denominator \* (x(i) - x(j));

end

end

coefficients(i) = f(i) / numerator;

end

end

function F = differentiate\_polynomial(~, x, coefficients, X)

n = length(x);

F = 0;

for i = 1:n

prod = 1;

for j = 1:n

if j ~= i

prod = prod \* (X - x(j));

end

end

F = F + coefficients(i) \* prod;

end

end

% Approximate differentiation function

function F = approximate\_differentiation(~, x, f, X)

n = length(x);

m = length(X);

F = zeros(1, m);

% Check if x is uniformly spaced

if any(diff(x) ~= x(2) - x(1))

error('x must be uniformly spaced');

end

h = x(2) - x(1); % Step size, assuming uniform spacing

for k = 1:m

% Find the index of the closest node to X(k)

[~, idx] = min(abs(x - X(k)));

% Adjust index to ensure it's not too close to the edges

if idx < 3

idx = 3;

elseif idx > n - 2

idx = n - 2;

end

% Select five nodes around the closest index

nodes = idx-2:idx+2;

xx = x(nodes);

ff = f(nodes);

% Fit a polynomial of degree 4 to the selected nodes

poly = polyfit(xx, ff, 4);

% Differentiate the polynomial

dpoly = polyder(poly);

% Evaluate the derivative at X(k)

F(k) = polyval(dpoly, X(k));

end

end

% Function to read data from a .mat file

function loadDataFromFile(app)

[file, path] = uigetfile('\*.mat');

if isequal(file, 0)

disp('User selected Cancel');

else

fullPath = fullfile(path, file);

data = load(fullPath);

if isfield(data, 'x') && isfield(data, 'f')

app.xData = data.x;

app.fData = data.f;

app.UITable.Data = [data.x(:), data.f(:)];

if isfield(data, 'X')

app.XData = data.X;

else

app.XData = NaN;

end

else

disp('Invalid file format. The file should contain variables x and f, and optionally X.');

end

end

end

% Function to update the plot

function updatePlot(app, x, f, X, F)

% Clear existing plot

cla(app.UIAxes);

% Plot original data points

plot(app.UIAxes, x, f, 'bo', 'DisplayName', 'Original Data');

hold(app.UIAxes, 'on');

% Plot interpolation or differentiation result

plot(app.UIAxes, X, F, 'r-', 'DisplayName', 'Computed Result');

% Add legend

legend(app.UIAxes, 'show');

% Release the hold

hold(app.UIAxes, 'off');

end

end

% Callbacks that handle component events

methods (Access = private)

% Selection changed function: OperationsButtonGroup

function OperationsButtonGroupSelectionChanged(app, event)

% Show or hide the method selection based on the selected operation

if app.InterpolationButton.Value

app.MethodDropDownLabel.Visible = 'on';

app.MethodDropDown.Visible = 'on';

% Adjust method options for interpolation

app.MethodDropDown.Items = {'Lagrange''s', 'Newton''s'};

elseif app.DifferentiationButton.Value

app.MethodDropDownLabel.Visible = 'on';

app.MethodDropDown.Visible = 'on';

% Adjust method options for differentiation

app.MethodDropDown.Items = {'Differentiate', 'Approximate'};

else

app.MethodDropDownLabel.Visible = 'off';

app.MethodDropDown.Visible = 'off';

end

end

% Button pushed function: SolveButton

function SolveButtonPushed(app, event)

if isempty(app.xData) || isempty(app.fData)

disp('Data is empty or not in the expected format.');

return;

end

x = app.xData;

f = app.fData;

X = app.XData;

if isnan(X)

X = str2double(app.XEditField.Value);

if isnan(X)

disp('Invalid value for X');

return;

end

end

if app.InterpolationButton.Value

method = app.MethodDropDown.Value;

if strcmp(method, 'Lagrange''s')

F = app.lagrange\_interpolation(x, f, X);

elseif strcmp(method, 'Newton''s')

F = app.newton\_interpolation(x, f, X);

end

elseif app.DifferentiationButton.Value

method = app.MethodDropDown.Value;

if strcmp(method, 'Differentiate')

F = app.differentiate\_interpolation(x, f, X);

elseif strcmp(method, 'Approximate')

F = app.approximate\_differentiation(x, f, X);

end

end

app.UITable\_2.Data = [X(:), F(:)];

% Update the plot with the original and computed data

app.updatePlot(x, f, X, F);

end

% Button pushed function: InputFileButton

function InputFileButtonPushed(app, event)

app.loadDataFromFile();

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create UIFigure and hide until all components are created

app.UIFigure = uifigure('Visible', 'off');

app.UIFigure.Position = [100 100 965 601];

app.UIFigure.Name = 'MATLAB App';

% Create UIAxes

app.UIAxes = uiaxes(app.UIFigure);

title(app.UIAxes, 'Results')

xlabel(app.UIAxes, 'X')

ylabel(app.UIAxes, 'Y')

zlabel(app.UIAxes, 'Z')

app.UIAxes.Position = [41 356 300 185];

% Create OperationsButtonGroup

app.OperationsButtonGroup = uibuttongroup(app.UIFigure);

app.OperationsButtonGroup.SelectionChangedFcn = createCallbackFcn(app, @OperationsButtonGroupSelectionChanged, true);

app.OperationsButtonGroup.Title = 'Operations';

app.OperationsButtonGroup.Position = [422 463 123 78];

% Create InterpolationButton

app.InterpolationButton = uiradiobutton(app.OperationsButtonGroup);

app.InterpolationButton.Text = 'Interpolation';

app.InterpolationButton.Position = [11 32 88 22];

app.InterpolationButton.Value = true;

% Create DifferentiationButton

app.DifferentiationButton = uiradiobutton(app.OperationsButtonGroup);

app.DifferentiationButton.Text = 'Differentiation';

app.DifferentiationButton.Position = [11 10 96 22];

% Create MethodDropDownLabel

app.MethodDropDownLabel = uilabel(app.UIFigure);

app.MethodDropDownLabel.HorizontalAlignment = 'right';

app.MethodDropDownLabel.Visible = 'off';

app.MethodDropDownLabel.Position = [404 152 45 22];

app.MethodDropDownLabel.Text = 'Method';

% Create MethodDropDown

app.MethodDropDown = uidropdown(app.UIFigure);

app.MethodDropDown.Items = {'Lagrange''s', 'Newton''s', 'Differentiate', 'Approximate'};

app.MethodDropDown.Visible = 'off';

app.MethodDropDown.Position = [464 152 100 22];

app.MethodDropDown.Value = 'Lagrange''s';

% Create InputFileButton

app.InputFileButton = uibutton(app.UIFigure, 'push');

app.InputFileButton.ButtonPushedFcn = createCallbackFcn(app, @InputFileButtonPushed, true);

app.InputFileButton.BackgroundColor = [0 0.4471 0.7412];

app.InputFileButton.FontSize = 14;

app.InputFileButton.FontColor = [1 1 1];

app.InputFileButton.Position = [434 198 100 25];

app.InputFileButton.Text = 'Input File';

% Create SolveButton

app.SolveButton = uibutton(app.UIFigure, 'push');

app.SolveButton.ButtonPushedFcn = createCallbackFcn(app, @SolveButtonPushed, true);

app.SolveButton.BackgroundColor = [0 0.4471 0.7412];

app.SolveButton.FontSize = 18;

app.SolveButton.FontColor = [1 1 1];

app.SolveButton.Position = [404 49 168 42];

app.SolveButton.Text = 'Solve';

% Create UITable

app.UITable = uitable(app.UIFigure);

app.UITable.ColumnName = {'x'; 'f'};

app.UITable.RowName = {};

app.UITable.Position = [41 49 302 185];

% Create UITable\_2

app.UITable\_2 = uitable(app.UIFigure);

app.UITable\_2.ColumnName = {'X'; 'F'};

app.UITable\_2.RowName = {};

app.UITable\_2.Position = [633 49 302 185];

% Create xEditField\_2Label

app.xEditField\_2Label = uilabel(app.UIFigure);

app.xEditField\_2Label.HorizontalAlignment = 'right';

app.xEditField\_2Label.Position = [414 389 25 22];

app.xEditField\_2Label.Text = 'x';

% Create xEditField

app.xEditField = uieditfield(app.UIFigure, 'text');

app.xEditField.Position = [454 389 100 22];

% Create fEditFieldLabel

app.fEditFieldLabel = uilabel(app.UIFigure);

app.fEditFieldLabel.HorizontalAlignment = 'right';

app.fEditFieldLabel.Position = [414 345 25 22];

app.fEditFieldLabel.Text = 'f';

% Create fEditField

app.fEditField = uieditfield(app.UIFigure, 'text');

app.fEditField.Position = [454 345 100 22];

% Create XEditFieldLabel

app.XEditFieldLabel = uilabel(app.UIFigure);

app.XEditFieldLabel.HorizontalAlignment = 'right';

app.XEditFieldLabel.Position = [414 301 25 22];

app.XEditFieldLabel.Text = 'X';

% Create XEditField

app.XEditField = uieditfield(app.UIFigure, 'text');

app.XEditField.Position = [454 301 100 22];

% Show the figure after all components are created

app.UIFigure.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = CalculatorOfInterpolations

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.UIFigure)

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

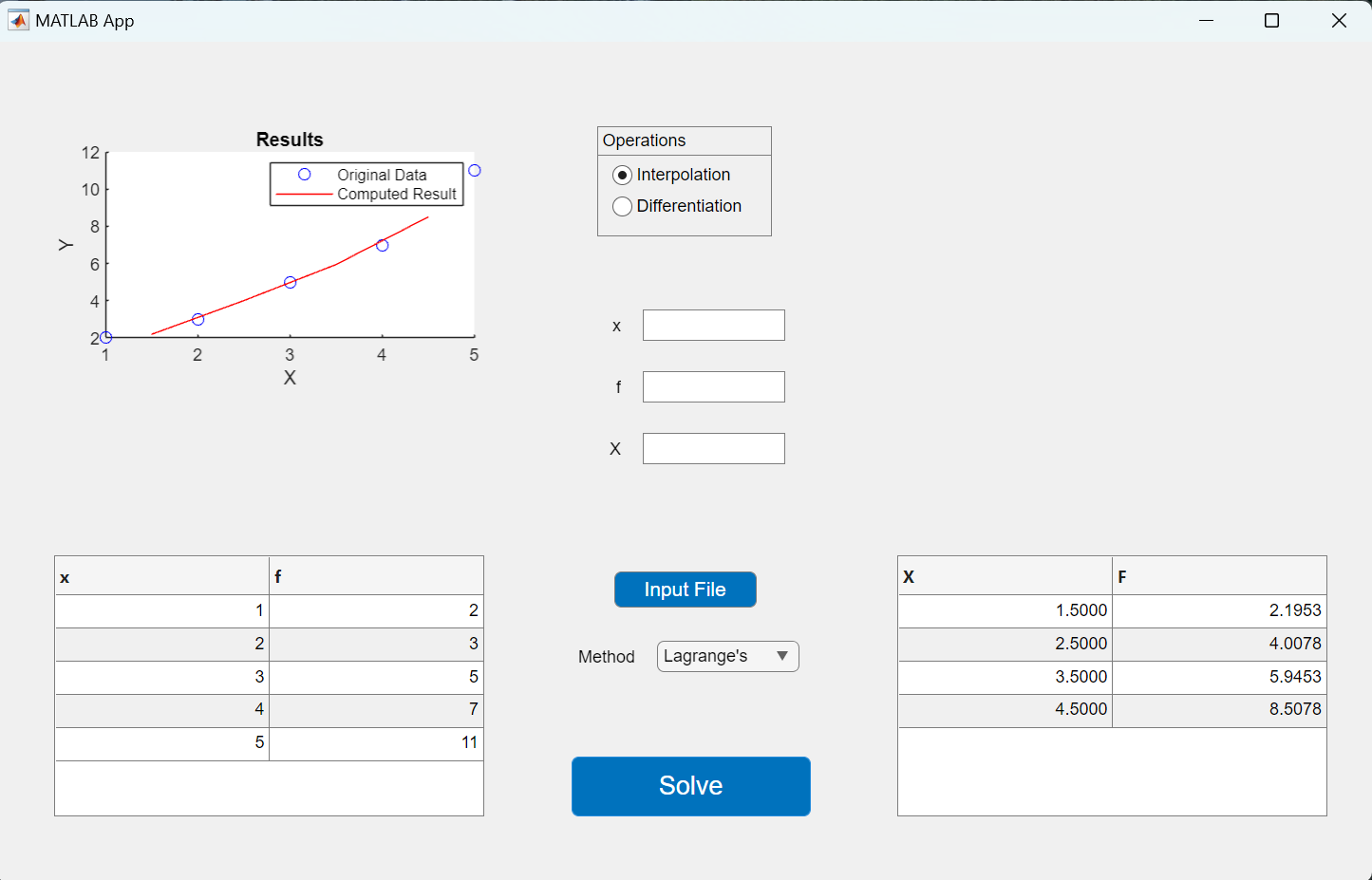
delete(app.UIFigure)

end

end

end

## Результати виконання програми



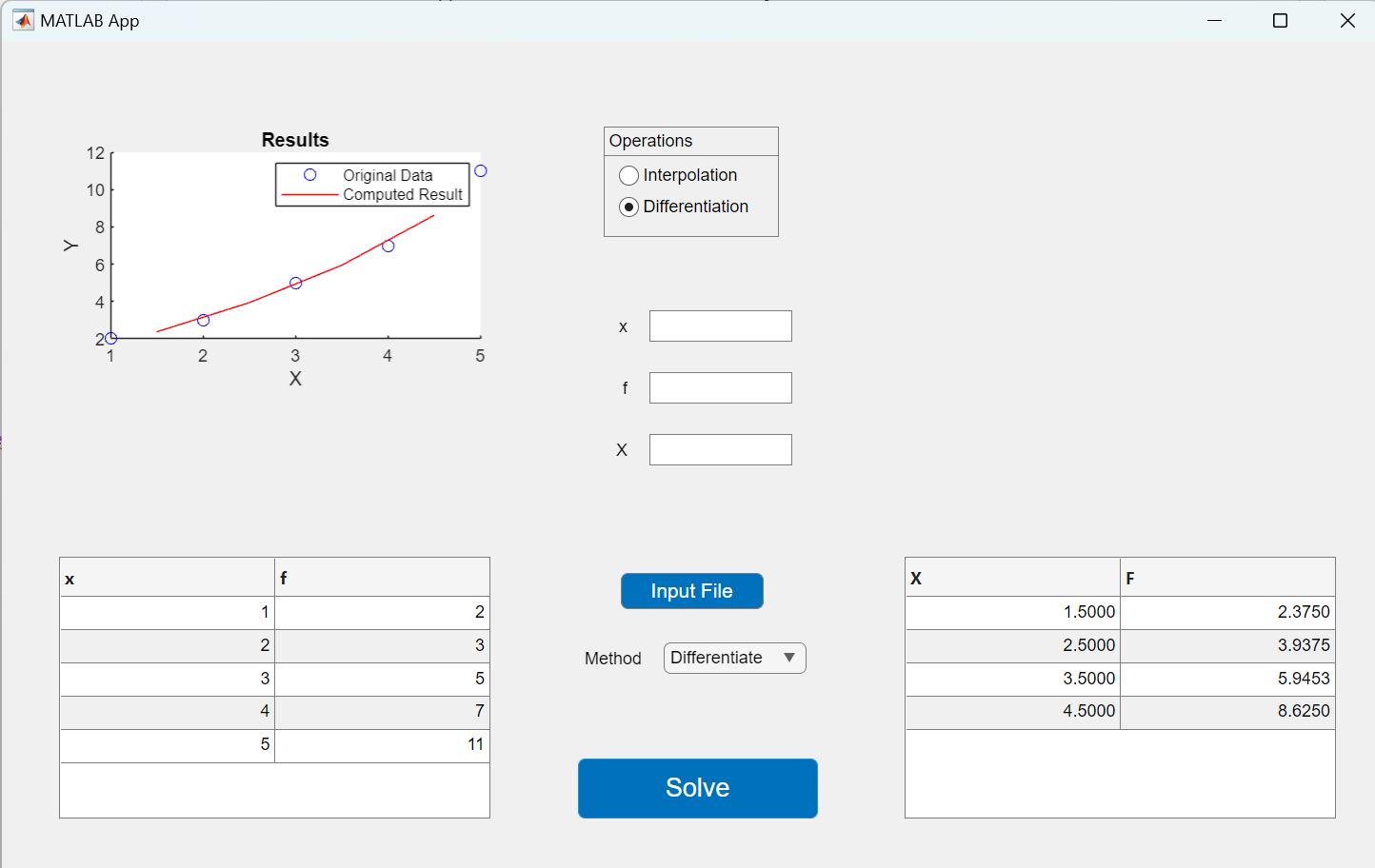


Рисунок 1.5, 1.6 – «Результати обчислень програми з різними методами»

Розроблена програма графічного інтерфейсу для лабораторної роботи забезпечує можливість виконання інтерполяції та диференціювання таблично заданих функцій. Вона має користувацький інтерфейс, що дозволяє вибирати методи інтерполяції та диференціювання, завантажувати дані з файлів, вводити дані вручну та відображати результати обчислень на графіку.

Ця програма включає наступні основні компоненти:

1. Вікно програми: Відображається графічний інтерфейс, який користувач може використовувати для взаємодії з програмою.
2. Група кнопок операцій: Користувач може вибрати одну з операцій: інтерполяція або диференціювання.
3. Кнопки вибору методу: Для кожної операції доступні варіанти методів, які можна вибрати з випадаючого списку.
4. Кнопка завантаження файлу: Дозволяє користувачеві завантажити дані з файлу для обробки.
5. Кнопка розв'язання: Після вибору операції та методу користувач може натиснути цю кнопку для виконання обчислень.
6. Таблиця введених даних: Показує введені дані для перегляду та редагування.
7. Графік результатів: Відображає результати обчислень у вигляді графіку для візуалізації.

Програма використовує методи інтерполяції Лагранжа та Ньютона, а також методи диференціювання, щоб обчислити значення функції в заданих точках. Вона надає зручний інтерфейс для роботи з таблично заданими даними та виконання обчислень без необхідності написання складних кодів.

# **ВИСНОВОК**

У ході спринту з чисельних методів було розглянуто та використано різноманітні методи для інтерполяції та диференціювання таблично заданих функцій. Для інтерполяції були використані методи Лагранжа та Ньютона, які дозволяють апроксимувати значення функції в проміжних точках за допомогою значень, відомих в деяких вузлових точках. Для диференціювання був реалізований метод за допомогою інтерполяції для п'яти вузлів, а також метод апроксимації, який використовує апроксимацію похідної поліномів.

Для побудови програми було використано середовище MATLAB та його графічний інтерфейс App Designer. Програма має користувацький інтерфейс, який дозволяє зручно вводити дані, вибирати методи та відображати результати обчислень на графіку. Було розроблено клас MATLAB для інтерактивного використання програми, що спрощує процес обробки та аналізу числових даних.

Отже, лабораторна робота дозволила вивчити та застосувати чисельні методи для розв'язання завдань інтерполяції та диференціювання, а також надала практичний досвід у використанні середовища MATLAB для побудови програм з графічним інтерфейсом.