Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій



**ЗВІТ**

**Про виконання лабораторної роботи № 7**

«Чисельні методи розв’язування систем нелінійних рівнянь»

**з дисципліни «Чисельні методи»**

**Лектор:**

доцент кафедри ПЗ

Мельник Н.Б.

**Виконав:**

студ. групи ПЗ-15

Марущак А. С.

**Прийняв:**

асистент кафедри ПЗ

Гарматій Г.Ю

«\_\_\_» \_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

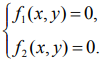
Львів – 2022

**Тема роботи:** Чисельні методи розв’язування систем нелінійних рівнянь.

**Мета роботи:** ознайомлення на практиці з методом ітерацій та методом Ньютона розв’язування систем нелінійних рівнянь.

**Теоретичні відомості**

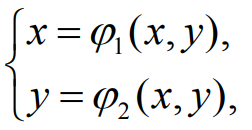
Розглянемо систему двох нелінійних рівнянь з двома невідомими



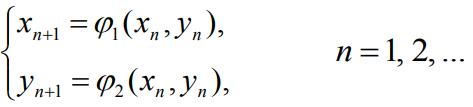
Розв’язком цієї системи є пара чисел , яка перетворює систему рівнянь в тотожність.

**Метод ітерації**

Припустимо, що - наближений розв’язок системи, яку перетворимо до такого вигляду

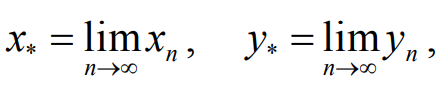


де - неперервно-диференційовані функції за змінними x та y . Розглянемо ітераційний процес



який породжує числові послідовності .

Якщо ітераційний процес збігається, тобто існують границі



То розв’язком початкової системи є пара чисел .

Причому, умовами збіжності цього процесу в деякій замкнутій області D є наступні:

1. функції визначені та неперервно-диференційовані в області D;
2. початкове наближення і всі наступні наближення (n =1, 2,...) належать області D;
3. в області D виконуються нерівності

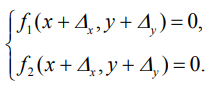
Або аналогічні

**Метод Ньютона**

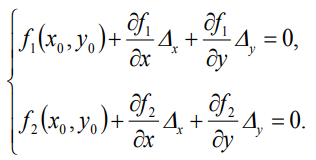
Це найрозповсюдженіший метод розв’язування систем нелінійних рівнянь. Він забезпечує кращу збіжність, ніж метод простої ітерації.

Нехай - наближений розв’язок системи, а Δx, Δy - деякі поправки до точного розв’язку.

Розглянемо систему рівнянь



Розкладемо функції в ряд Тейлора, обмежившись лінійними членами розкладу відносно Δx, Δy



І тоді цю систему перепишемо у вигляді

Тоді, розв’яжемо цю систему відносно будь-яким відомим нам методом розв'язування СЛАР.

І тоді наступне наближення виражається через попереднє наступним чином

Ці операції треба повторювати, допоки не досягнемо потрібної точності.

**Індивідуальне завдання**

1. Ознайомитися з теоретичним матеріалом.
2. Розв'язати систему нелінійних рівнянь з точністю методом ітерацій та методом Ньютона.

**Варіант завдання**



**Хід роботи**

Проведемо деякі обчилення вручну, щоб потім мати змогу використати отримані результати в реалізації програми.

По перше, нам необхідно локалізувати розв’язок системи. Для цього скористаємось графічним методом. Побудуємо графіки цих 2-ох рівнянь на координатній площині :

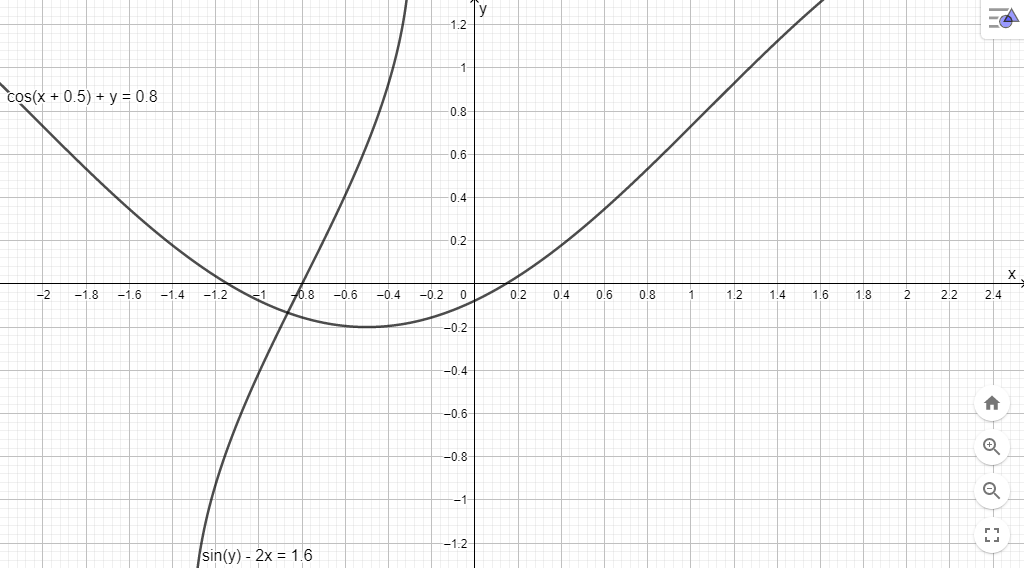


Рис 1.1 Графічна інтерпретація системи рівнянь

З рис 1.1 очевидно, що корінь локалізований в області , а також видно, що за початкове наближення можна взяти точку .

Тепер для того, щоб розв'язати систему методом ітерації, запишемо її у наступному вигляді:

Введемо позначення

і, для зручності, переставимо рядки системи місцями:

Перевіримо збіжність системи на області D:

.

Отож, ітераційний процес

буде збіжним на області при початковому наближені . Для прикладу, проведемо першу ітерацію:

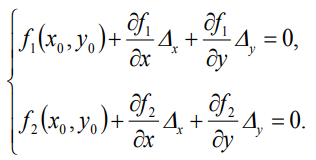
Подальші ітерації виконаємо вже програмно.

Для методу Ньютона перепишемо вихідну систему у вигляді:

Тоді введемо позначення . Отримаємо, що

Тоді, нехай числа такі, що

Тоді, за розкладом Тейлора, обмежившись лінійними членами, цю систему можна записати так:



Або у вигляді

Маємо СЛАР відносно . Для n-ої ітерації маємо:

Де . Або у матричному вигляді:

Де – матриця коефіцієнтів, або так звана матриця Якобі, , . Тоді, згідно з методом оберненої матриці запишемо, що

І, врахувавши формулу переходу від до , до , запишемо ітераційну формулу:

*,*

де .

Наприклад, для першої ітерації . Тоді

Наступна мета – реалізувати подані методи програмно, використовуючи ці дані.

Для подальшої роботи я буду використовувати структуру даних Matrix, яка відповідає матриці заданого розміру, як зрозуміло з назви. Серед її функціональних можливостей – множення та додавання матриць, пошук транспонованої матриці та детермінанту, пошук оберненої матриці. Покажемо це у коді:

public static Matrix operator\*(Matrix a, Matrix b)

{

if (a.ColumnsCount != b.RowsCount)

throw new ArgumentException("Cannot multiply matrices of these sizes.");

Matrix result = new(a.RowsCount, b.ColumnsCount);

for (int i = 0; i < result.RowsCount; i++)

for (int j = 0; j < result.ColumnsCount; j++)

result[i, j] = a.Row(i) \* b.Column(j);

return result;

}

public static Matrix operator+(Matrix a, Matrix b)

{

if (a.ColumnsCount != b.ColumnsCount || a.RowsCount != b.RowsCount)

throw new ArgumentException("Cannot add matrices of different sizes.");

Matrix result = new(a.RowsCount, b.ColumnsCount);

for (int i = 0; i < result.RowsCount; i++)

for (int j = 0; j < result.ColumnsCount; j++)

result[i, j] = a[i, j] + b[i, j];

return result;

}

public Matrix Transponed()

{

Matrix result = new(ColumnsCount, RowsCount);

for (int i = 0; i < RowsCount; i++)

for (int j = 0; j < ColumnsCount; j++)

result[j, i] = this[i, j];

return result;

}

public double Determinant()

{

if (ColumnsCount != RowsCount)

throw new InvalidOperationException("Cannot find determinant for non-square matrices.");

if (ColumnsCount == 1)

return this[0, 0];

double result = 0;

for (int i = 0; i < ColumnsCount; i++)

{

result += Math.Pow(-1, i) \* this[0, i] \* this.WithoutRow(0).WithoutColumn(i).Determinant();

}

return result;

}

public double AlgebraicAddition(int i, int j) => Math.Pow(-1, i+j) \* this.WithoutRow(i).WithoutColumn(j).Determinant();

public Matrix Inversed()

{

if (RowsCount != ColumnsCount)

throw new InvalidOperationException();

Matrix result = new Matrix(RowsCount, ColumnsCount);

for(int i = 0; i < RowsCount; i++)

{

for(int j = 0; j < ColumnsCount; j++)

{

result[i, j] = this.AlgebraicAddition(i, j);

}

}

return (1/this.Determinant()) \* (result.Transponed());

}

І основна програма буде виглядати наступним чином:

using MatrixLib;

Console.Write("Початкове наближення: ");

var numbers = Console.ReadLine().Split().Select(double.Parse).Take(2).ToArray();

var x0 = numbers[0];

var y0 = numbers[1];

var solution = SimpleIteration(x0, y0, out int iters);

Console.WriteLine($"Метод iтерацiї: {solution.Item1:0.000}, {solution.Item2:0.000}; Iтерацiй: {iters}");

var solution2 = NewtonMethod(x0, y0, out int iters2);

Console.WriteLine($"Метод Ньютона: {solution2.Item1:0.000}, {solution2.Item2:0.000}; Iтерацiй: {iters2}");

(double, double) SimpleIteration(double x0, double y0, out int iterations)

{

double x = x0, y = y0;

iterations = 0;

do

{

//Зберігаємо попередні значення

x0 = x;

y0 = y;

//Використовуємо ітераційні формули

x = Math.Sin(y0) / 2 - 0.8;

y = 0.8 - Math.Cos(x0 + 0.5);

iterations++;

} while (Math.Max(Math.Abs(y - y0),Math.Abs(x - x0)) > 0.001); // Допоки не досягнемо заданої точності.

return (x, y);

}

//Обчислення матриці Якобі

Matrix J(Matrix X)

{

Matrix res = new(2, 2);

res[0, 0] = -Math.Sin(X[0, 0] + 0.5);

res[0, 1] = 1;

res[1, 0] = -2;

res[1, 1] = Math.Cos(X[1, 0]);

return res;

}

//Обчислення значень функції

Matrix F(Matrix X)

{

Matrix res = new(2, 1);

res[0, 0] = Math.Cos(X[0, 0] + 0.5) + X[1, 0] - 0.8;

res[1, 0] = Math.Sin(X[1, 0]) - 2 \* X[0, 0] - 1.6;

return res;

}

(double, double) NewtonMethod(double x0, double y0, out int iterations)

{

Matrix X = new(2, 1), X\_prev;

X[0,0] = x0;

X[1, 0] = y0;

iterations = 0;

do

{

//Зберігаємо попередні значення

X\_prev = X.Clone();

//Використовуємо ітераційну формулу

X = X\_prev + (-1) \* J(X\_prev).Inversed() \* F(X\_prev);

iterations++;

} while (Math.Max(Math.Abs(X\_prev[0, 0] - X[0, 0]), Math.Abs(X\_prev[1, 0] - X[1, 0])) > 0.001); // Допоки не досягнемо заданої точності.

return (X[0,0], X[1,0]);

}

Як зазначалось вище, початкове наближення . Подамо його на вхід програми і подивимось на результат.

**Результат** виконання програми:



Рис 1.2 Результат виконання програми.

**Аналіз результатів:**

Як бачимо, результати обчислень обома методами співпали. Для того, щоб перевірити їх дійсність, підставимо їх у початкову систему:

Це означає, що систему було розв’язано вірно.

**Висновок:**

Виконуючи цю лабораторну роботу ми ознайомились на практиці з методом ітерацій та методом Ньютона розв’язування систем нелінійних рівнянь.

За допомогою цих знань ми реалізували програму, що знайшла розв’язок наступної системи:



Перед цим, ми провели деякі обчислення вручну, які потім використали у програмній реалізації.

Ми отримали вектор-стовпець дійсних чисел, що відображає розв' язок цієї системи:

і довели, що цей результат є вірним.