Всем привет это звит к лабораторке с многочисленных методов всем приятного просмотра, погнали.

**Лабораторна робота**

**Чисельні методи в інформатиці**

**Підготувала студентка 3 курсу групи ІПС факультету КНК**

**Боровик Анастасія**

**Київ 2024**

**Умова:**

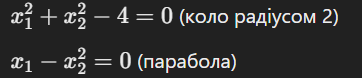
Написати програму, яка розв’язує систему нелінійних рівнянь:

методом Ньютонатільки без обчислень обернених матриць

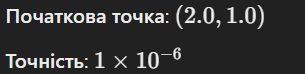
Метод Ньютона, або метод Ньютона-Рафсона, є одним з найпоширеніших чисельних методів для знаходження коренів нелінійних рівнянь. Він може бути розширений на випадок систем нелінійних рівнянь. Основна ідея полягає в тому, щоб використовувати інформацію про похідні функцій для уточнення наближення до кореня.

### **Обрана система нелінійних рівнянь**

Розглянемо систему:

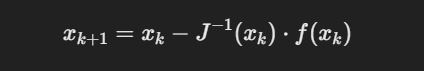


Точка старту та точність



### **a. Теорія**

Метод Ньютона для системи рівнянь використовує наступну формулу для оновлення значення:



де J — це матриця Якобі.

### **b. Умови збіжності**

Для застосування методу Ньютона:

1. Функція f має бути неперервною.
2. Якобіан J має бути невиродженим (детермінант не дорівнює нулю) в околі розв'язку.

Ми можемо перевірити, чи є початкова точка достатньо близькою до розв'язку та чи не є матриця Якобі виродженою в цій точці.

**c. Вивід ітерацій та проміжних обчислень**

Програма виводить всі значення ітерацій та проміжні обчислення.

import numpy as np

def f(x):

return np.array([

x[0]\*\*2 + x[1]\*\*2 - 4, # x1^2 + x2^2 = 4

x[0] - x[1]\*\*2 # x1 - x2^2 = 0

])

def jacobian(x):

return np.array([

[2\*x[0], 2\*x[1]], # Часткові похідні першого рівняння

[1, -2\*x[1]] # Часткові похідні другого рівняння

])

def newton\_method(initial\_guess, eps, max\_iter):

x = initial\_guess

iter\_values = []

for \_ in range(max\_iter):

J = jacobian(x)

F = f(x)

# Метод Ньютона без обчислення оберненої матриці

delta = np.linalg.solve(J, -F)

x = x + delta

iter\_values.append(x.copy())

if np.linalg.norm(F, ord=2) < eps:

break

return x, iter\_values

def main():

initial\_guess = np.array([2.0, 1.0]) # Початкове наближення

eps = 1e-6 # Точність

max\_iter = 100 # Максимальна кількість ітерацій

solution, iter\_values = newton\_method(initial\_guess, eps, max\_iter)

print("Знайдене розв'язання:", solution)

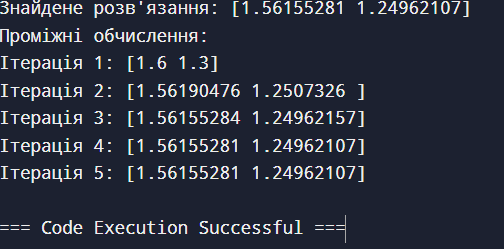
print("Проміжні обчислення:")

for i, val in enumerate(iter\_values):

print(f"Ітерація {i + 1}: {val}")

if \_\_name\_\_ == "\_\_main\_\_":

main()



### **Висновок**

Метод Ньютона успішно знайшов розв'язок системи нелінійних рівнянь, демонструючи швидку збіжність при заданій точності. Всі ітераційні значення надають можливість проаналізувати процес наближення.