МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №7

з дисципліни: «Чисельні методи»

16 Варіант

СПЕЦІАЛЬНОСТІ 121 – Інженерія програмного забезпечення

Виконав: Левак О. О.

Група: ІТ-91

Перевірила: Тимофєєва Ю.С.

Київ 2020

**Лабораторна робота №7. Розв’язування систем нелінійних рівнянь**

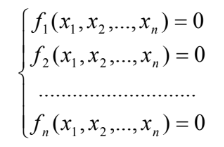
**Мета:** навчитися розв'язувати систему нелінійних рівнянь

методами простих ітерацій і Ньютона. За допомогою бібліотеки matplotlib

графічно визначити початкове наближення.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

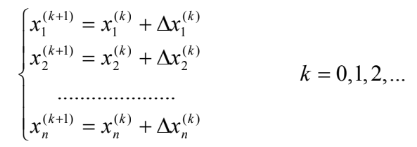
Систему нелінійних рівнянь із n невідомими можна записати у вигляді:



Зрідка для розв’язування такої системи вдається застосувати метод послідовного виключення невідомих і звести розв’язування вихідної системи до розв’язування одного нелінійного рівняння з одним невідомим. Значення інших невідомих величин знаходять відповідною підстановкою в конкретні вирази. Однак у переважній більшості випадків для розв’язування систем нелінійних рівнянь використовуються ітераційні методи. Надалі припускається, що шукається ізольований розв’язок нелінійної системи.

Як і у випадку одного нелінійного рівняння, локалізація розв’язку може здійснюватися на основі специфічної інформації з конкретного розв'язуваного завдання (наприклад, з фізичних міркувань), і за допомогою методів математичного аналізу. Для розв’язування системи двох рівнянь, досить часто зручним є графічний спосіб, у якому розташування коренів визначається як точки перетину.

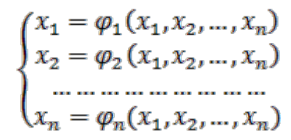
**Метод Ньютона**. Якщо визначено початкове наближення, ітераційний процес знаходження розв’язку системи методом Ньютона можна подати у вигляді:



Для реалізації алгоритму методу Ньютона в більшості випадків ліпше не обчислювати обернену матрицю − а знайти із системи значення приростів і обчислити нового наближення за (7.3). Щоб розв’язати такі лінійні системи, можна залучати різноманітні методи, як прямі, так і ітераційні (див. лабораторну роботу No 6), з урахуванням розмірності n розв'язуваної задачі та специфіки матриць Якобі J(x) (наприклад, симетрії, розрідженості тощо).

Щоб використовувати метод Ньютона, необхідно, щоб функції диференційовні і матриця Якобі була невиродженою. У разі, якщо початкове наближення обрано в досить малому околі шуканого кореня, ітерації збігаються до точного розв’язку, причому збіжність квадратична.

Метод простої ітерації. Коли використовується метод простої ітерації, система рівнянь приводиться до еквівалентної системи спеціального вигляду:

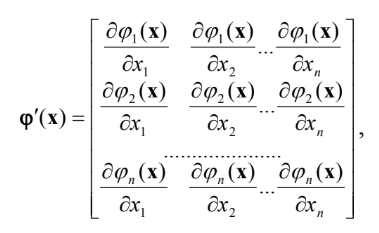


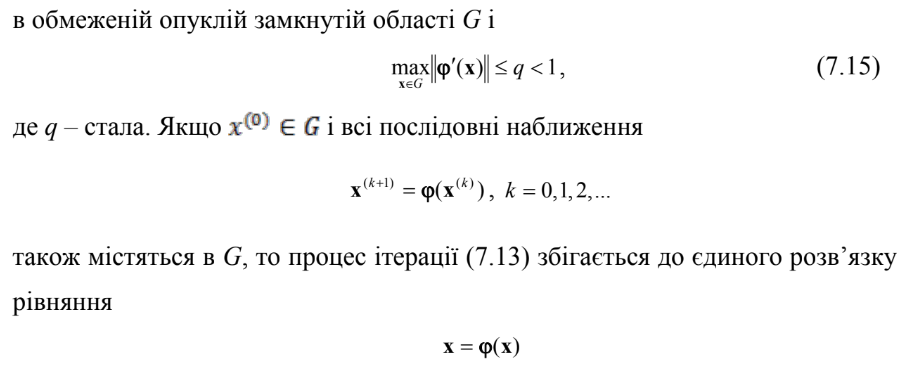
Якщо послідовність збігається, то вона збігається до розв’язку.

Достатня умова збіжності ітераційного процесу формулюється в такий спосіб:

Теорема 7.1.

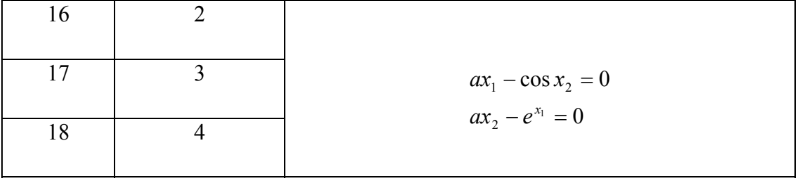
Нехай вектор-функція (x) неперервна разом зі своєю похідною



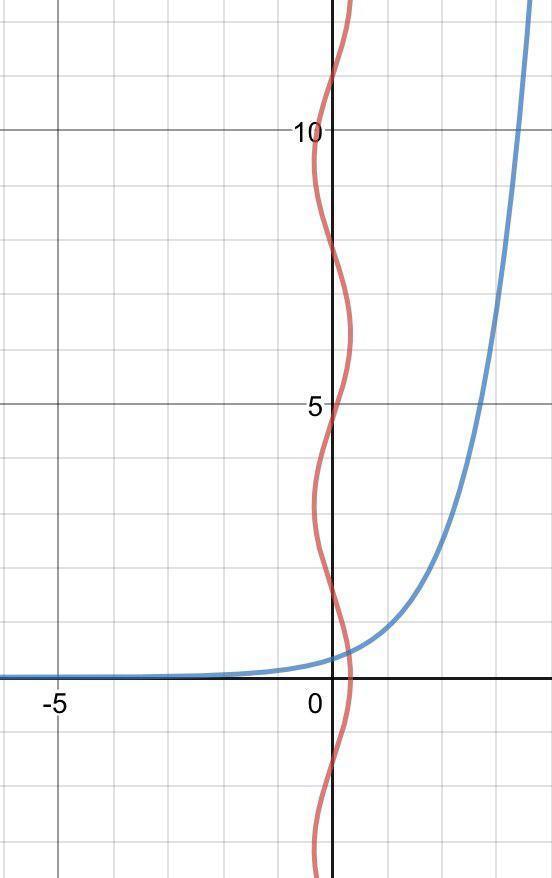


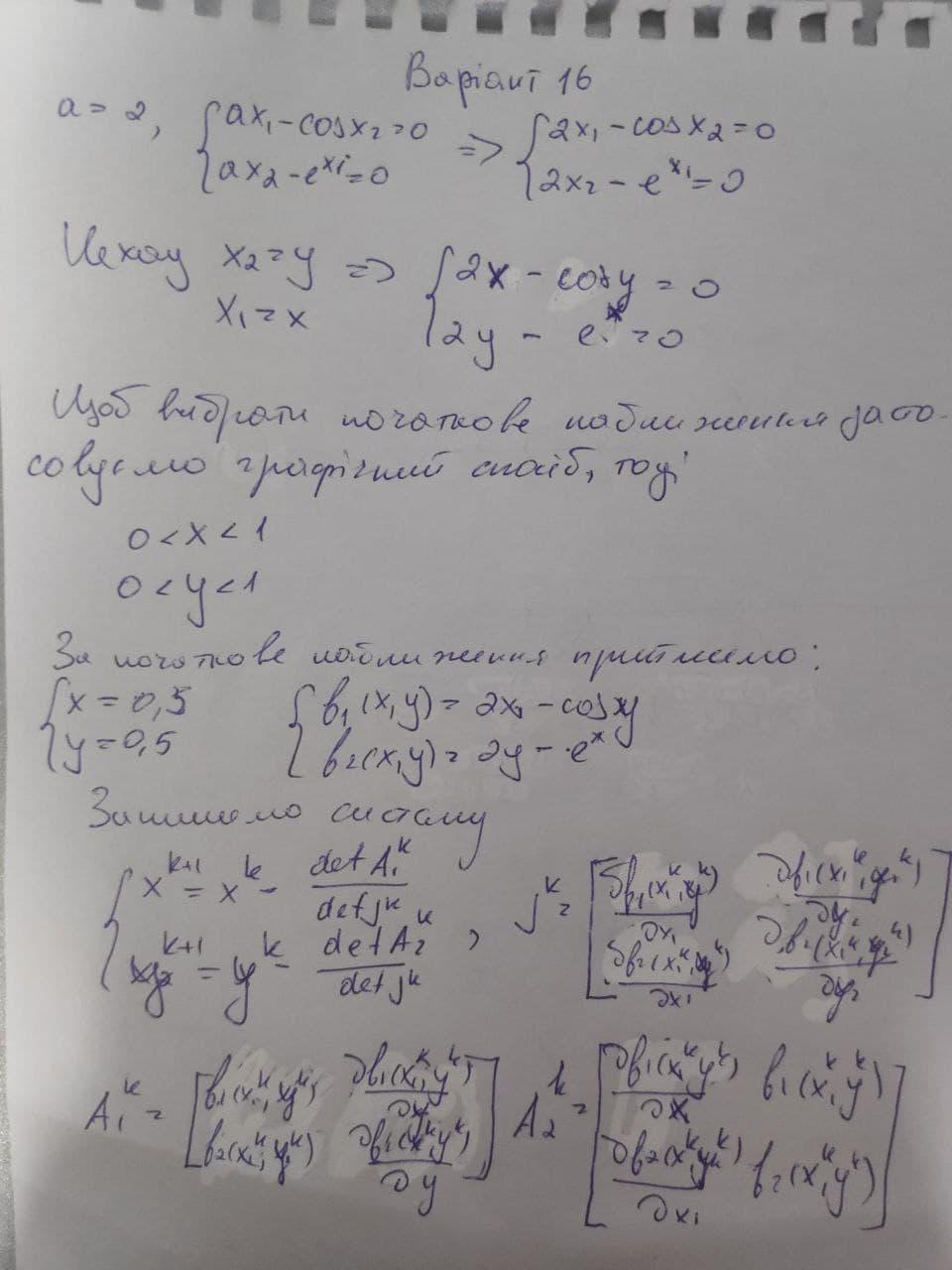
**РОЗВ’ЯЗОК ЗАДАЧІ В АНАЛІТИЧНІЙ ФОРМІ**

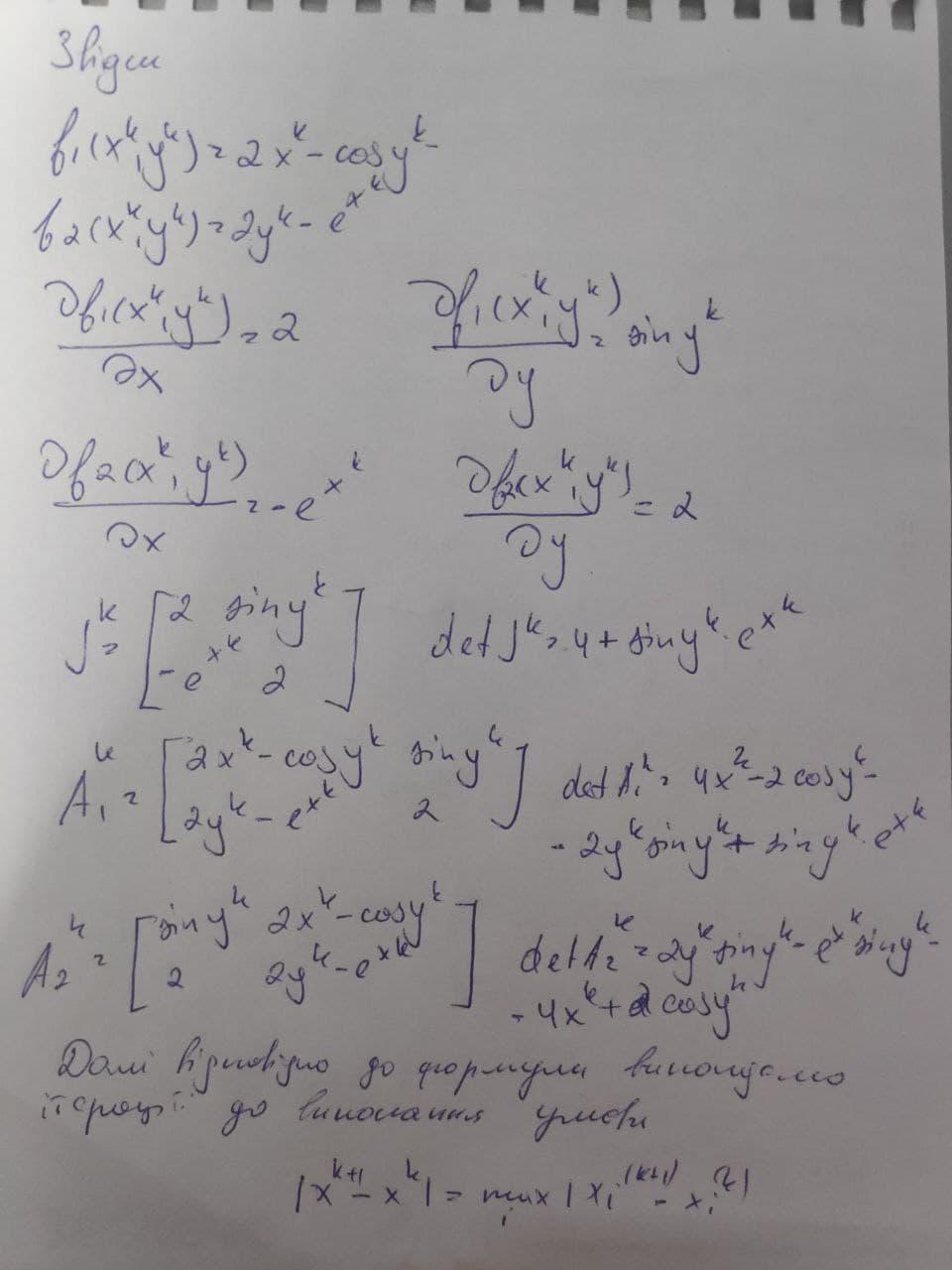
Запишемо умову:

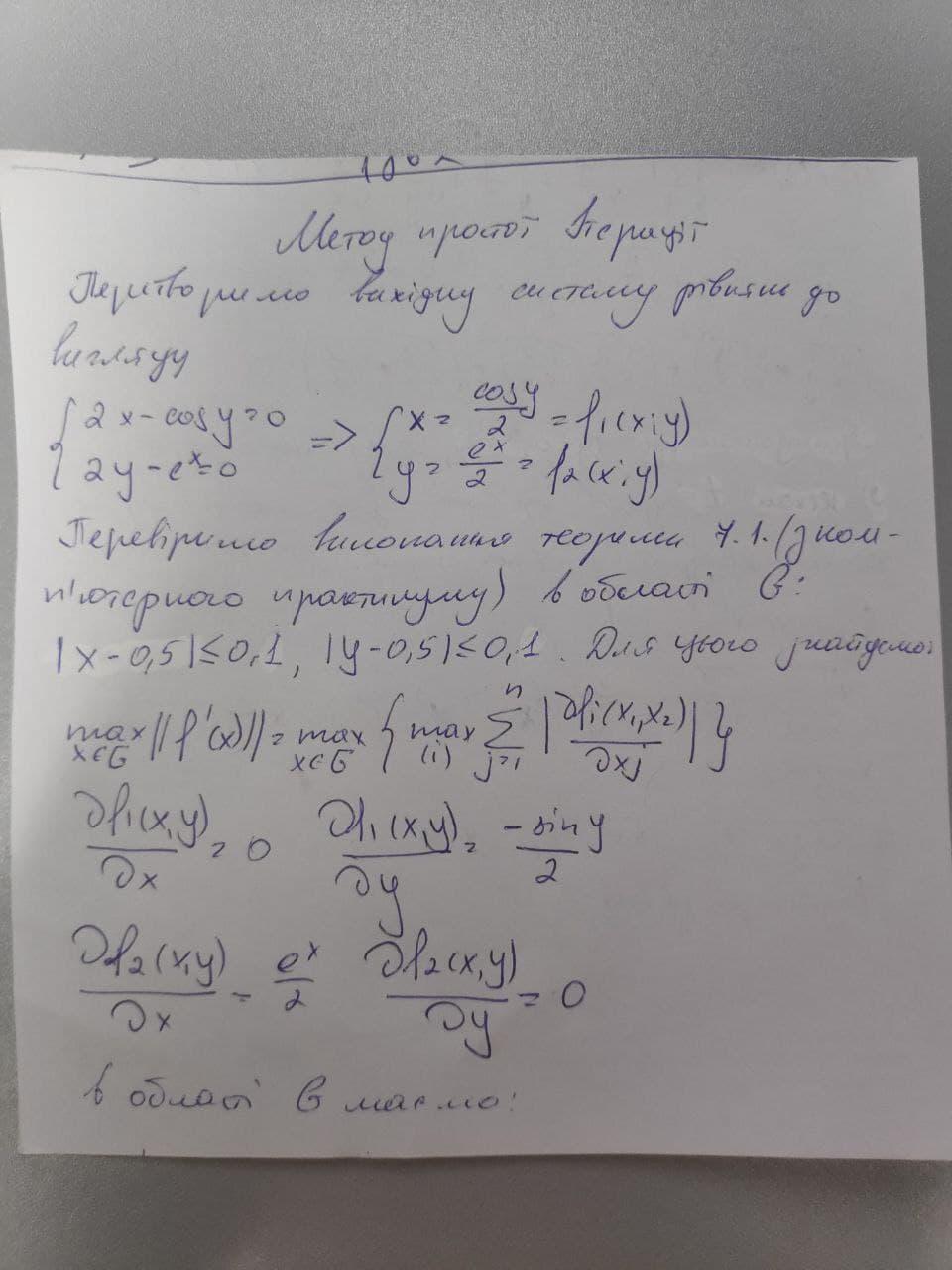


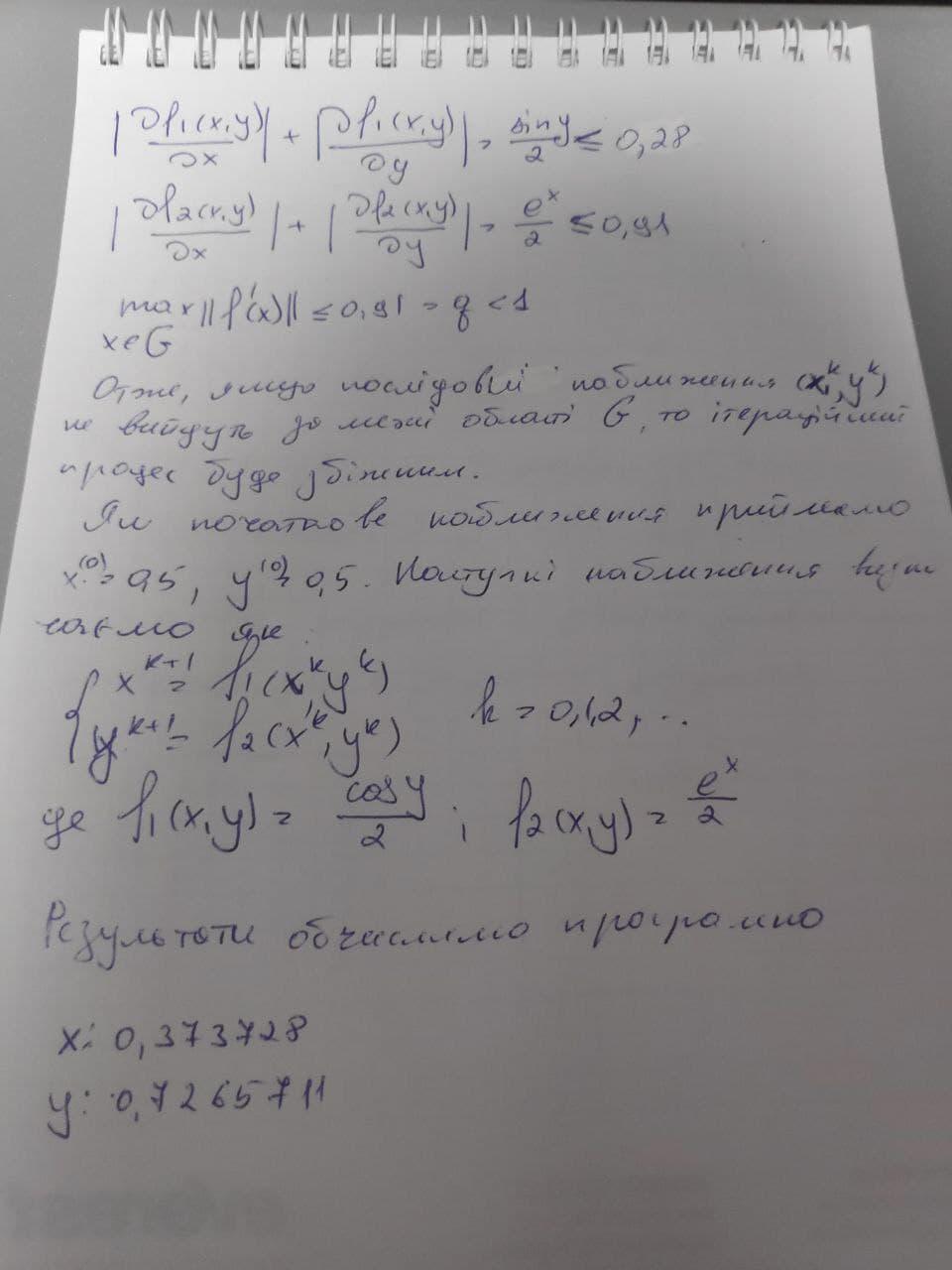
Точний розв’язок:



****

****

****

****

**ЛІСТИНГ ПРОГРАМИ РОЗВ’ЯЗУВАННЯ ЗАДАЧІ**

**from sympy import \***

**import math**

**def getDet(x, y, fns):**

**mainDiag = 1**

**diag = 1**

**for i in range(len(fns)):**

**fn = fns[i]**

**fnResult = fn(x, y)**

**if i <= 1:**

**mainDiag \*= fnResult**

**else:**

**diag \*= fnResult**

**result = mainDiag - diag**

**return result**

**def newton(x, y, fns, diffFns, e):**

**x1, y1 = x, y**

**while true:**

**J = getDet(x1, y1, diffFns)**

**A1 = getDet(x1, y1, [fns[0], diffFns[1], diffFns[2], fns[1]])**

**A2 = getDet(x1, y1, [diffFns[0], fns[1], fns[0], diffFns[3]])**

**x1 = x - (A1 / J)**

**y1 = y - (A2 / J)**

**if abs(x - x1) <= e: break**

**x, y = x1, y1**

**return [x1, y1]**

**def iteration(x, y, fns, e):**

**x1, y1 = x, y**

**while true:**

**x1 = fns[0](x1, y1)**

**y1 = fns[1](x1, y1)**

**if abs(x - x1) <= e: break**

**x, y = x1, y1**

**return [x1, y1]**

**def f1(x, y):**

**return 2\*x - math.cos(y)**

**def f2(x, y):**

**return 2\*y - math.e\*\*x**

**def deffFunc1X1(x, y):**

**return 2**

**def deffFunc1X2(x, y):**

**return math.sin(y)**

**def deffFunc2X1(x, y):**

**return -math.e\*\*x**

**def deffFunc2X2(x, y):**

**return 2**

**def f1Iter(x, y):**

**return math.cos(y) / 2**

**def f2Iter(x, y):**

**return math.e\*\*x / 2**

**def deffFuncIter1X1(x, y):**

**return 0**

**def deffFuncIter1X2(x, y):**

**return -(math.sin(y) / 2)**

**def deffFuncIter2X1(x, y):**

**return math.e\*\*x / 2**

**def deffFuncIter2X2(x, y):**

**return 0**

**x = 0.5**

**y = 0.5**

**e = 0.0001**

**diffFns = [deffFunc1X1, deffFunc2X2, deffFunc1X2, deffFunc2X1]**

**fns = [f1, f2]**

**iterFns = [f1Iter, f2Iter]**

**x1 = newton(x, y, fns, diffFns, e)[0]**

**y1 = newton(x, y, fns, diffFns, e)[1]**

**x2 = iteration(x, y, iterFns, e)[0]**

**y2 = iteration(x, y, iterFns, e)[1]**

**print("Метод Ньютона")**

**print("x: ", x1)**

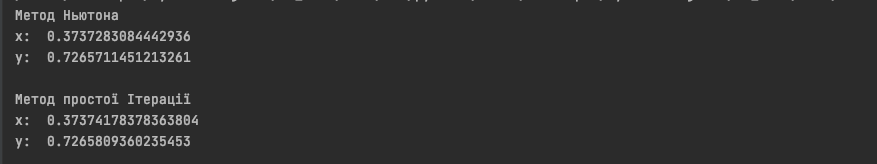
**print("y: ", y1)**

**print("\nМетод простої Ітерації")**

**print("x: ", x2)**

**print("y: ", y2)**

**РЕЗУЛЬТАТИ ВИКОНАННЯ ПРОГРАМ**

****

**Висновки :**

У ході виконання даної лабораторної роботи я навчилася знаходити корені нелінійної системи рівнянь різними методами, а саме: Ньютона і простої ітерації за допомогою мови програмування Python. Також я використала бібліотеку scipy, щоб порівняти результати мого аналітичного розрахунку та практичного з дійсним. Похибки в розв’язках відносно малі, це свідчить про те, що програма працює коректно.