, алеМіністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**Звіт**

Про виконання лабораторної роботи №7

**На тему:**

«Чисельні методи розв’язування систем нелінійних рівнянь»

з дисципліни «Чисельні Методи»

**Лектор:**

доцент каф. ПЗ

Мельник Н. Б.

**Виконав:**

ст. гр. ПЗ-11

Солтисюк Д.А.

**Прийняла:**

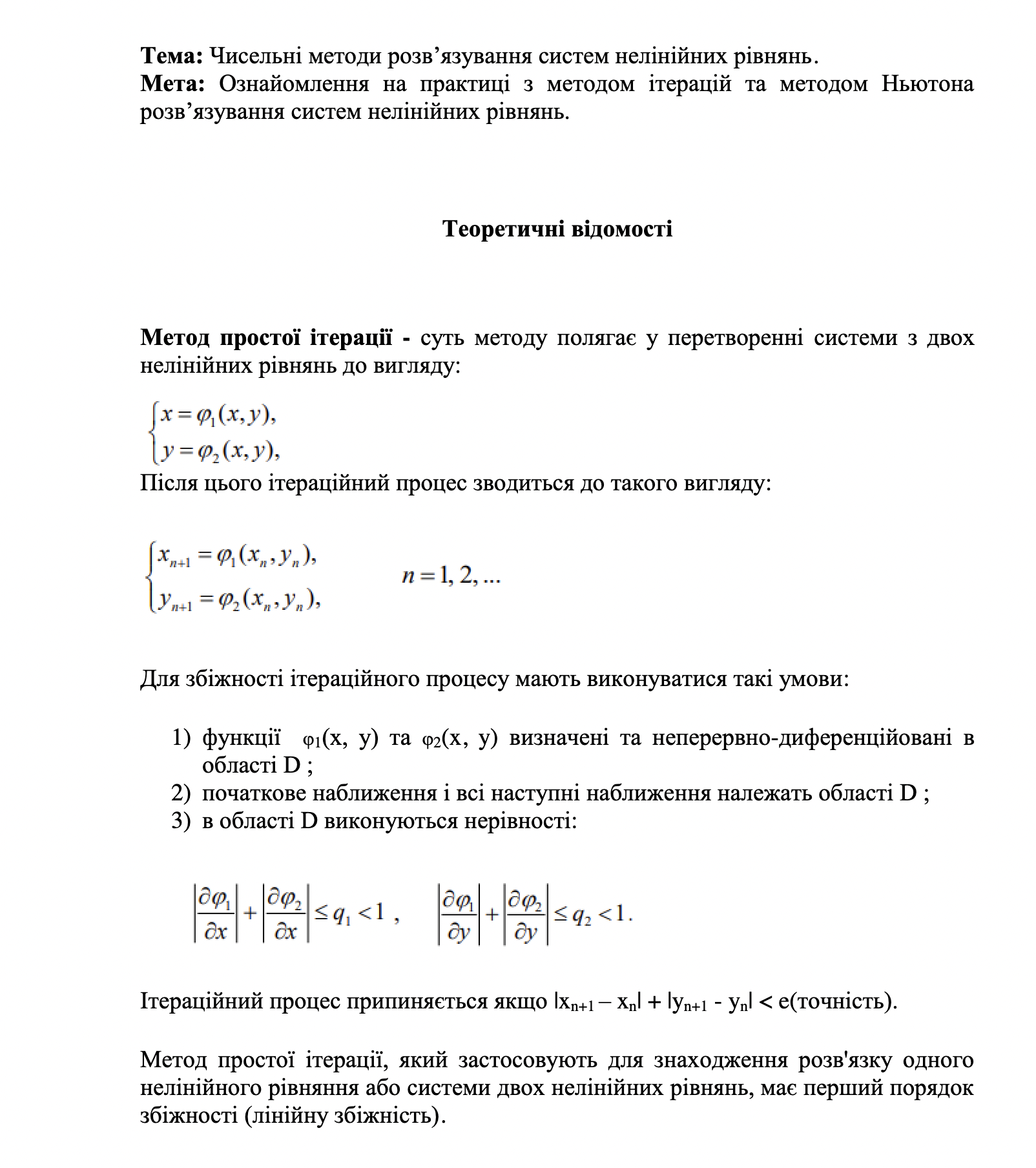
доцент каф. ПЗ

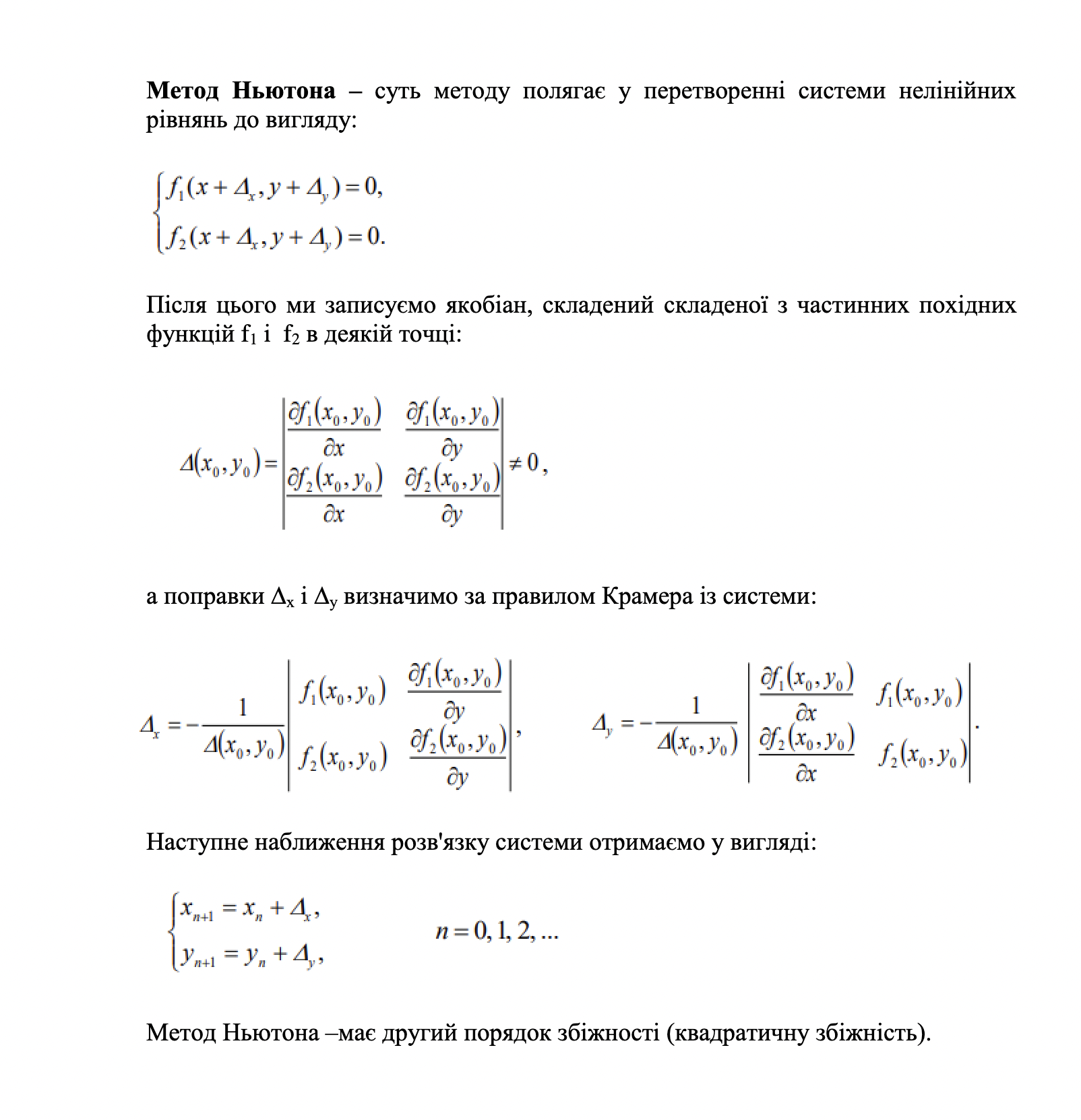
Мельник Н. Б.

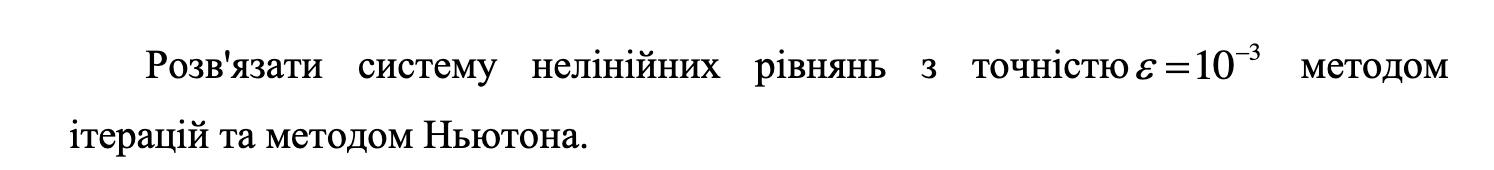
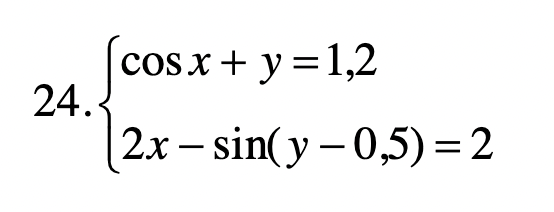
« \_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑ = \_\_\_\_\_ \_\_\_\_\_\_\_\_\_ .

Львів – 2022





**Індивідуальне завдання**

**Код програми**

simple\_iteration.py

**from** typing **import** Callable

**from** common.main **import** NonLinearEqSysOrientedMethod

**from** common.utils **import** print\_header

**from** prettytable **import** PrettyTable

**class** SimpleIterationMethod(NonLinearEqSysOrientedMethod):

**def** \_\_init\_\_(self, eqsys: list[Callable]):

super().\_\_init\_\_("Simple iterations", eqsys)

**def** execute\_method(self, tolerance=1e-4):

"""

1) Making an approximate guess

2) Linearly iterating using equeation function to get next x and y

3) Checking if can be considered solution:

(|x\_next - x| + |y\_next - y| < eps)

"""

x, y = 0, 0

print\_header("Initial guess")

**print**([x, y])

**print**("\n")

x\_next, y\_next = x, y

iterations = 0

f1, f2 = self.eqsys

t = PrettyTable(["Iteration", "X", "Y", "Precision"])

result = **None**

**while True**:

iterations += 1

x\_next, y\_next = f2(y), f1(x)

precision = abs(x\_next - x) + abs(y\_next - y)

x, y = x\_next, y\_next

t.add\_row([iterations, x, y, precision])

**if** precision < tolerance:

result = (x, y)

**break**

**print**(t)

**return** result

newton\_iteration.py

**import** math

**from** typing **import** Callable

**import** numpy **as** np

**from** common.gaussian **import** GaussianEliminationMethod

**from** common.main **import** NonLinearEqSysOrientedMethod

**from** prettytable **import** PrettyTable

**class** NewtonIterationMethod(NonLinearEqSysOrientedMethod):

**def** \_\_init\_\_(self, eqsys: list[Callable]):

super().\_\_init\_\_("Newton iteration", eqsys)

**def** execute\_method(self, tolerance=1e-4):

"""

1) May (x0, y0) -> approximate solution

delta\_x, delta\_y -> corrections

2) Delta(x0, y0) can be found from Jakobian (det of the Jakobi matrix)

3) delta\_x, delta\_y -> using gaussian elimintaion method

4) x\_next = x + delta\_x y\_next = y + delta\_y

"""

X = np.array([0, 0], dtype=float)

**def** f(X):

x, y = X

**return** np.array(

[math.cos(x) + y - 1.2, 2 \* x - math.sin(y - 0.5) - 2])

**def** Jf(X):

x, y = X

**return** np.array([[-math.sin(x + 0.5), 1], [2, math.cos(y - 2)]])

X\_delta = X.copy()

t = PrettyTable(["Iteration", "X", "Y", "Precision"])

result = **None**

iterations = 0

**while True**:

iterations += 1

A = Jf(X)

B = f(X)

X\_delta = GaussianEliminationMethod(A, B).compile(silent=**True**)

X -= X\_delta

x, y = X

norm = np.linalg.norm(B)

t.add\_row([iterations, x, y, norm])

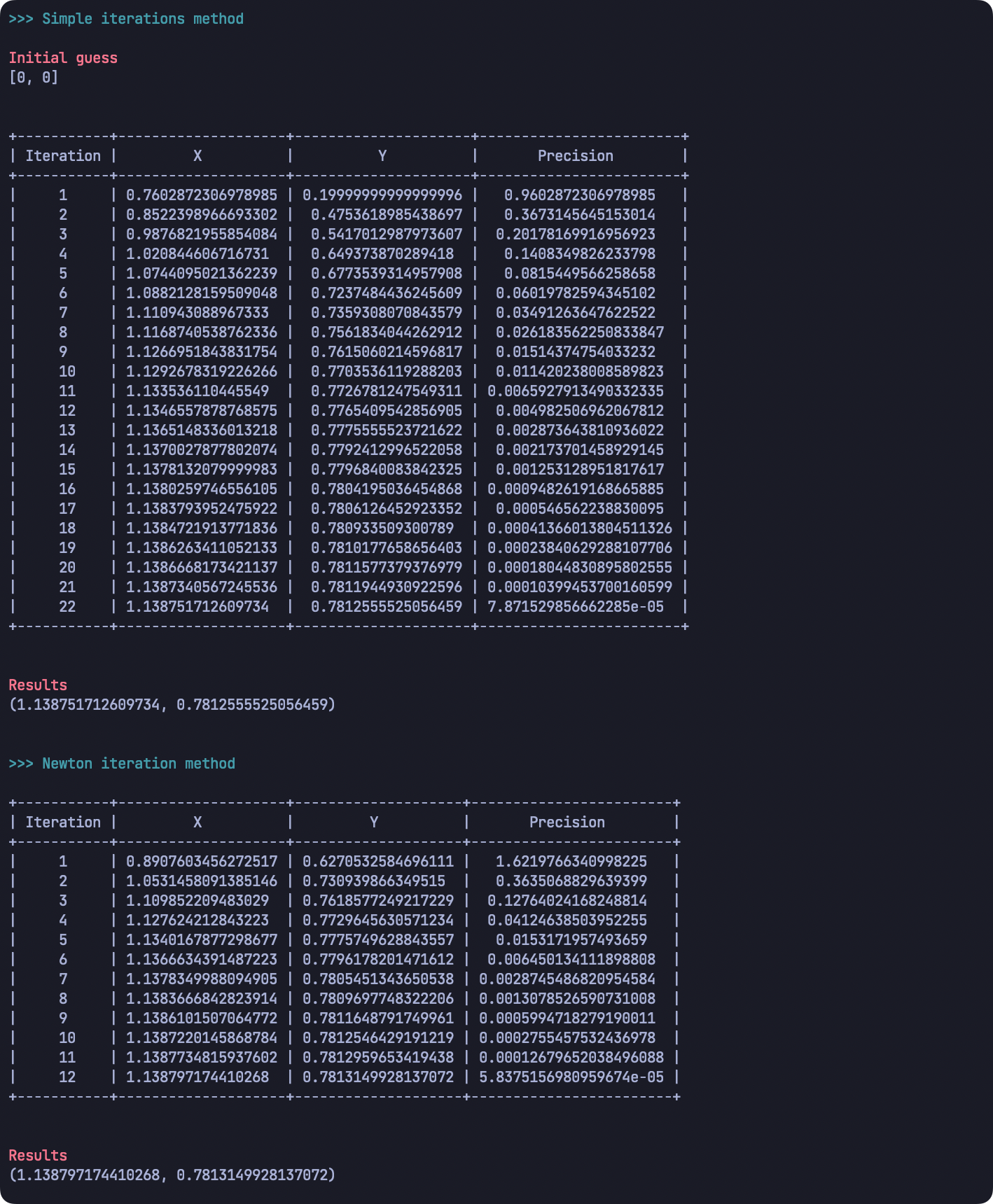
**if** norm < tolerance:

result = (x, y)

**break**

**print**(t)

**return** result

**Протокол роботи**

**Висновки**

Виконуючи лабораторну роботу №7, я навчився програмувати розв’язки систем

нелінійних рівнянь методами простої ітерації та Ньютона.