Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота №7

з дисципліни «Спеціальні розділи математики»

на тему

«**Розв’язання нелінійних рівнянь**»

Виконав:

студент гр. ІС-91

Хмелiнiн Андрiй

Викладач:

доц. Рибачук Л.В.

Київ-2020

**Зміст**

**1.**Постановка задачі у виглядівихідного рівняння.

**2.**Виконання допрограмованого етапу,результатом якого повинні бути проміжки,щодо яких проводяться уточнення.

**3.**Розв’язок уточнення коренів за методами бісекції,хорд,дотичних

**4.** Лістинг прогами

**5.** Висновки

**1.Постановка задачі**

****

x^4-2\*x^3-9\*x^2-3\*x-1

k=1

α=2

a5=0

a4=1

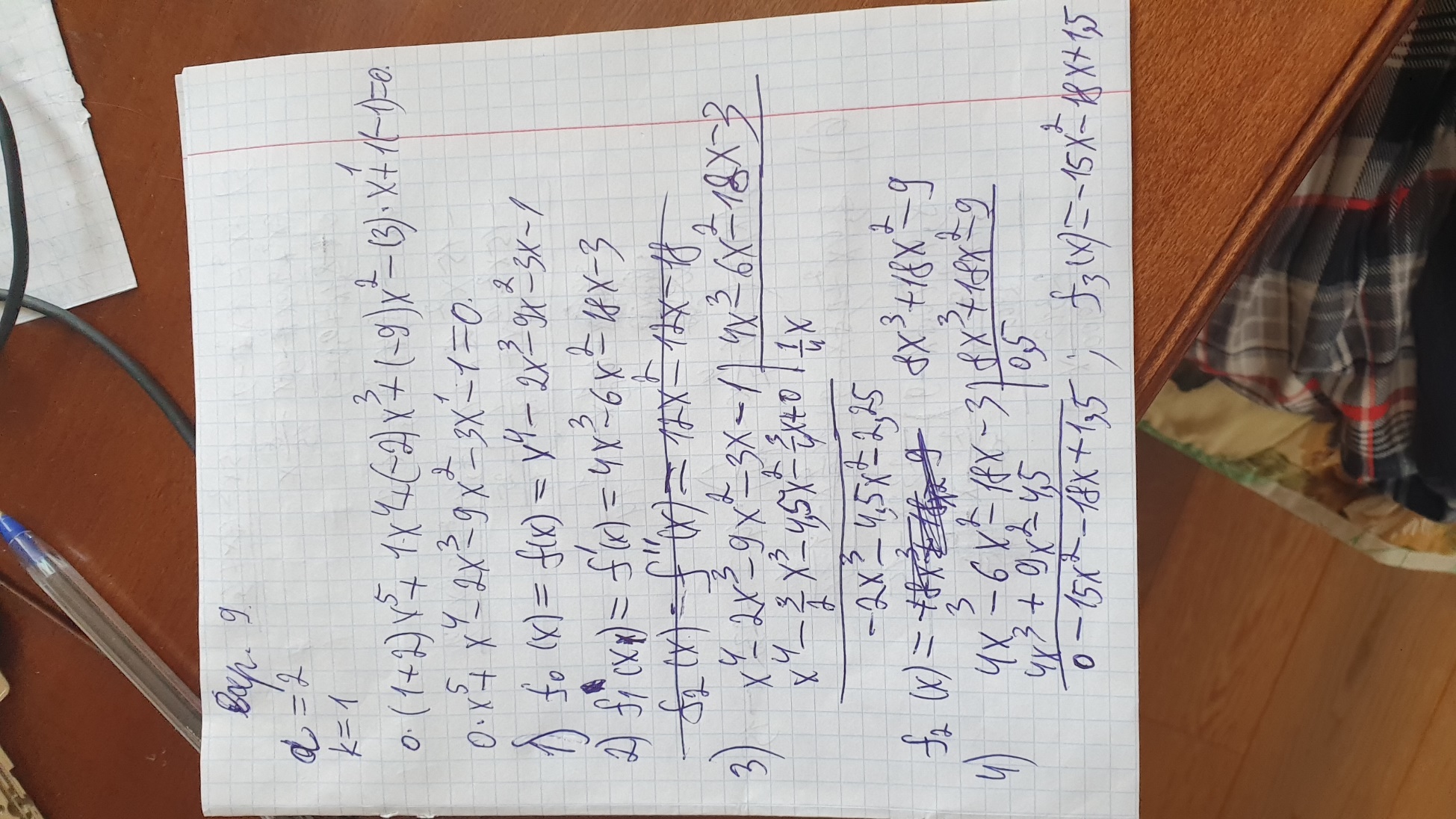
a3=-2

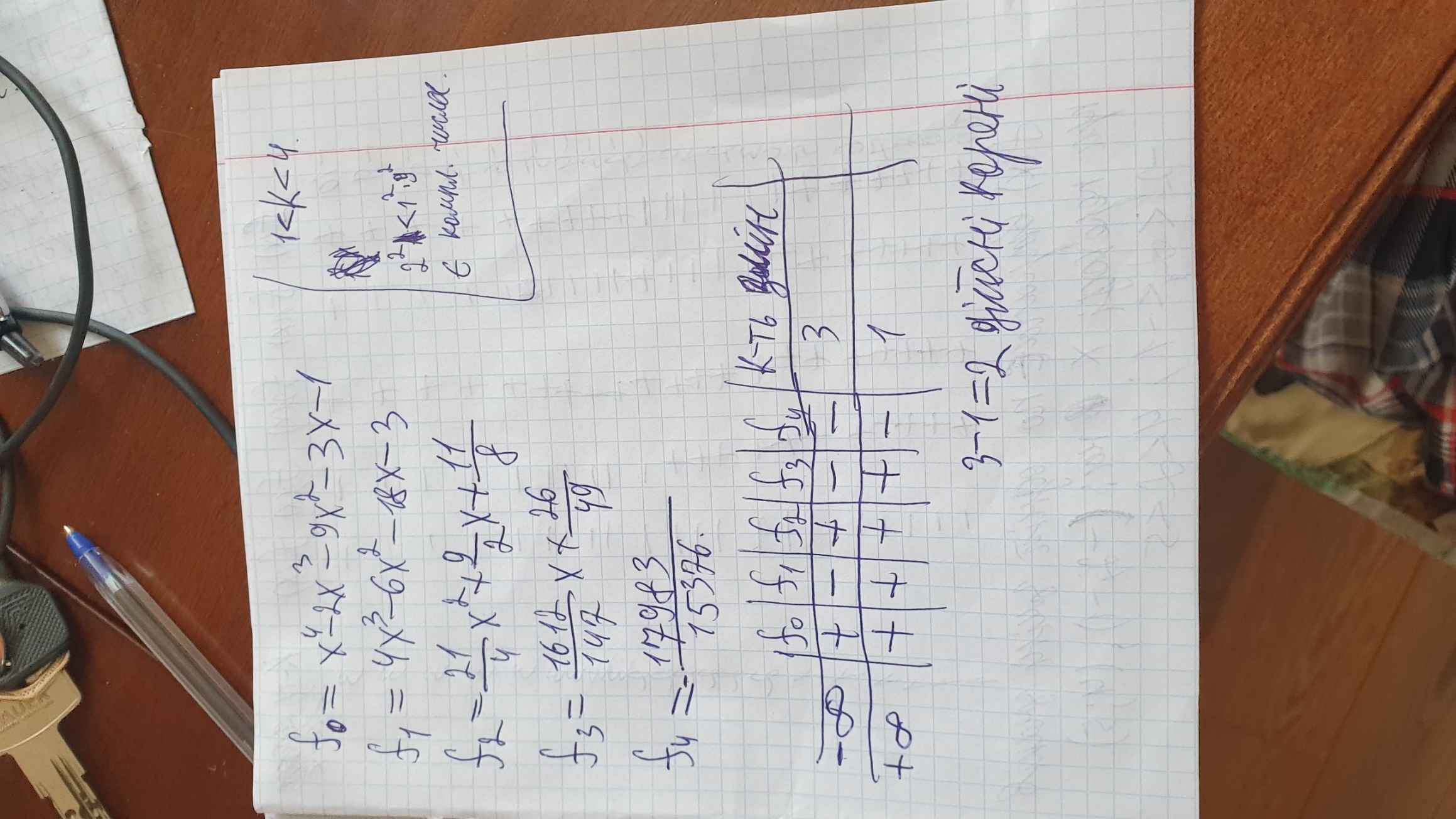
a2=-9

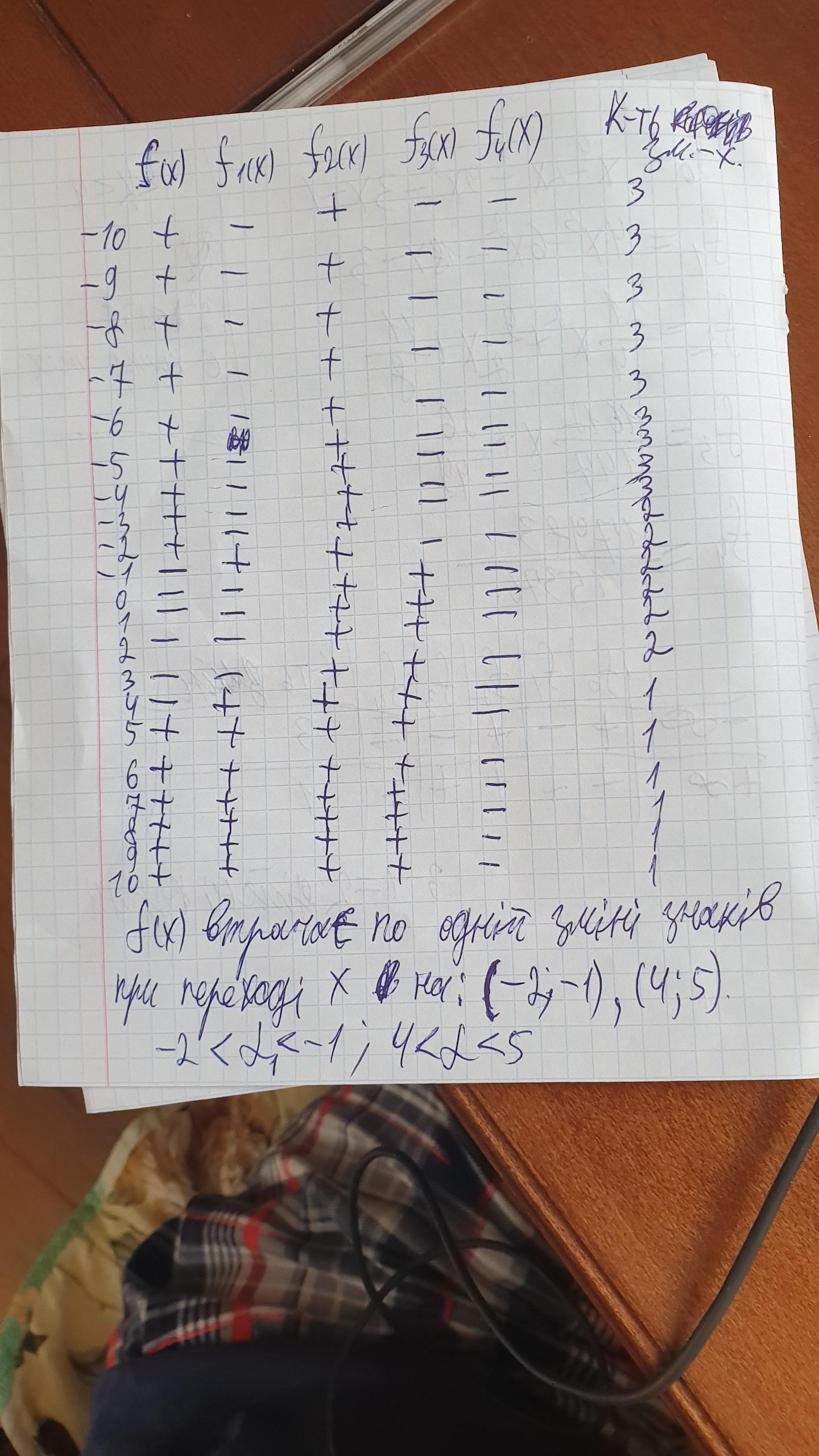
a1=-3

a0=-1

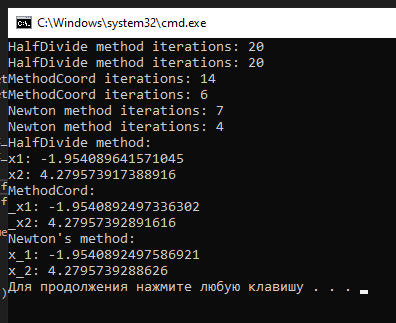
**2.Виконання допрограмованого етапу,результатом якого повинні бути проміжки,щодо яких проводяться уточнення**



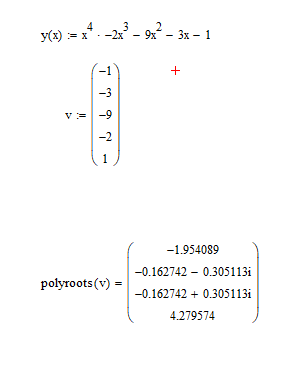




**3.Розв’язок уточнення коренів за методами бісекції, хорд, дотичних**



Рішення в Mathcad



**4.Лістинг програми**

import math

func\_glob = lambda x: (x\*\*4) -2\*(x\*\*3) - 9 \* (x \*\* 2) - 3\*x - 1

func\_glob1 = lambda x: 4 \* (x\*\*3)-6\*(x\*\*2) - 18 \* x-3

func\_glob2 = lambda x: 12\*(x\*\*2)-12\*x-18

a0, b0 = -2.0, -1.0

c0, d0 = 4.0, 5.0

eps = 0.000001

def half\_divide\_method(a, b, f, it1\_global=0):

x = (a + b) / 2

while math.fabs(f(x+eps)) >= eps and math.fabs(b - a) > eps:

x = (a + b) / 2

a, b = (a, x) if f(a) \* f(x) < 0 else (x, b)

it1\_global += 1

print("HalfDivide method iterations:", it1\_global)

return (a + b) / 2

# ----------------------------------------------------------------------------------

def method\_h(func, s, e, it2\_global=0):

while True:

it2\_global += 1

fa = func(s)

fb = func(e)

c = s - fa \* (e - s) / (fb - fa)

fc = func(c)

if fa \* fc < 0:

e = c

else:

e = s

s = c

if (math.fabs(e - s) <= eps and fc <= eps):

break

print("MethodCoord iterations:", it2\_global)

return c

#------------------------------------------------------------------------------------

def newton(a,b,f,f1,counter=0):

x\_prev=(a+b)/2

x=x\_prev-f(x\_prev)/f1(x\_prev)

while math.fabs(x-x\_prev) >= eps or math.fabs(f(x)) >= eps:

x=x\_prev-f(x\_prev)/f1(x\_prev)

x\_prev=x

counter+=1

print("Newton method iterations:", counter)

return x

counter1 = 0

counter2 = 0

x1 = half\_divide\_method(a0, b0, func\_glob, counter1)

x2 = half\_divide\_method(c0, d0, func\_glob)

\_x1 = method\_h(func\_glob, a0, b0, counter2)

\_x2 = method\_h(func\_glob, c0, d0)

x\_1 = newton(a0,b0,func\_glob, func\_glob1)

x\_2 = newton(c0,d0,func\_glob, func\_glob1)

print("HalfDivide method:")

print("x1:", x1)

print("x2:", x2)

print("MethodCord:")

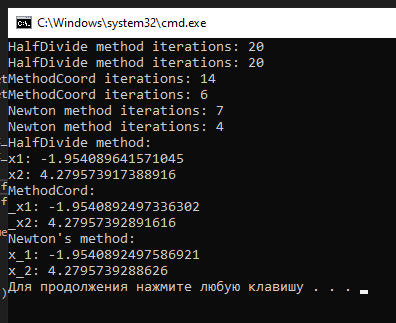
print("\_x1:", \_x1)

print("\_x2:", \_x2)

print("Newton's method:")

print("x\_1:",x\_1)

print("x\_2:",x\_2)



**5.Висновок**

Я визначив кількість дійсних коренів рівняння, відокремивши корені рівняння, порівнявши методи і дійшов до такого висновку, що

1) Метод бісекції ефективний для знаходження більш точного розв’язку рівняння, але це потребує більшої кількості ітерацій, ніж інші методи

2) Метод хорд значно швидший ніж перший, але існують проміжки (відрізки),на яких цей ітеративний метод для конкретної функції потребує значно більшу кількість ітерацій для проведення обчислень з заданою точністю

3) Метод Ньютона дозволяє за малу кількість ітерацій провести обчислення та отримати результат порівняно з попередніми методами. Більш того він здатний самостійно підвищувати точність підрахунку: швидкість наближення до кореня рівняння часом дозволяє знаходити результат з точністю більшою ніж задана. Недолік цього методу полягає в збільшені кількості потрібних ітерацій зі збільшенням степеням полінома від чого інші методи не залежать