Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 5

з дисципліни «Спеціальні розділи математики-2.  
Чисельні методи»

Виконав:

студент гр. ІС-34

Колосов Ігор

Викладач:

доц. Рибачук Л.В.

Київ – 2024

Тема: Розв’язання нелінійних рівнянь

Постановка задачі:

1. Допрограмовий етап: визначити кількість дійсних коренів рівняння, відокремити корені рівняння (письмово) (див. теореми про верхню та нижню границі, Гюа, метод поліномів Штурма). Результатом є висновок: перший корінь належить проміжку […], другий корінь належить проміжку […] і т.д.
2. Програмний етап: уточнити корені рівняння:
   1. методом бісекції;
   2. методом хорд;
   3. методом Ньютона (дотичних).

Критерієм закінчення ітераційного процесу мають бути нерівності:

- для методу бісекції (інтервальний метод; a та b - кінці інтервалу)

| b - a | < e та | f(xk) | < e

- для методiв хорд та дотичних

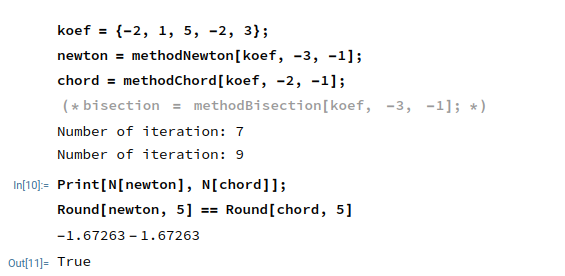
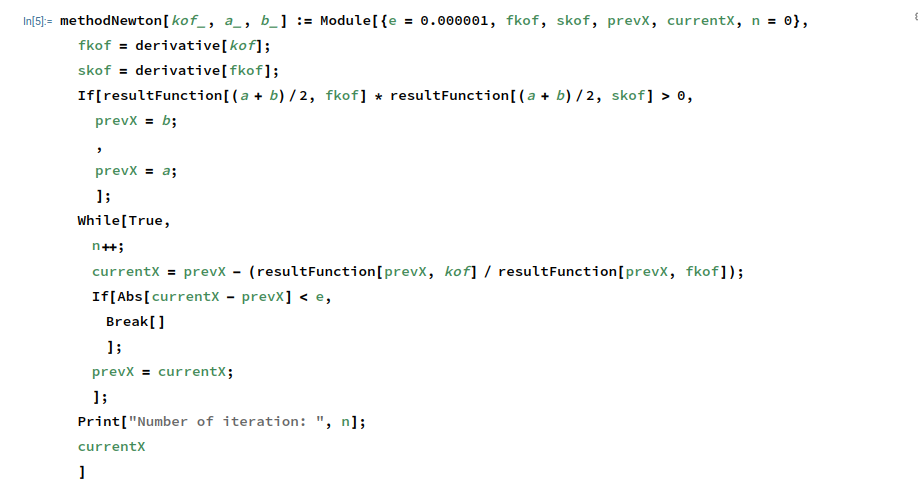
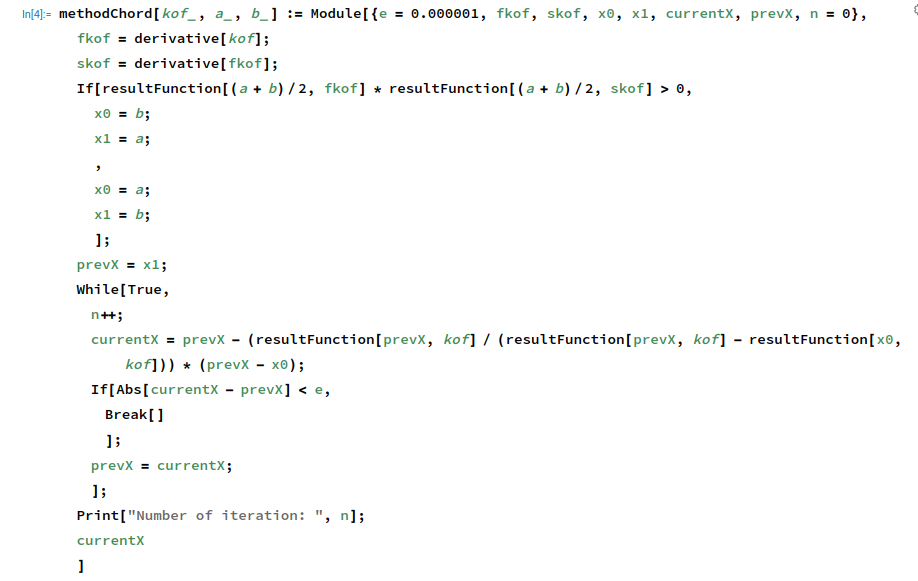
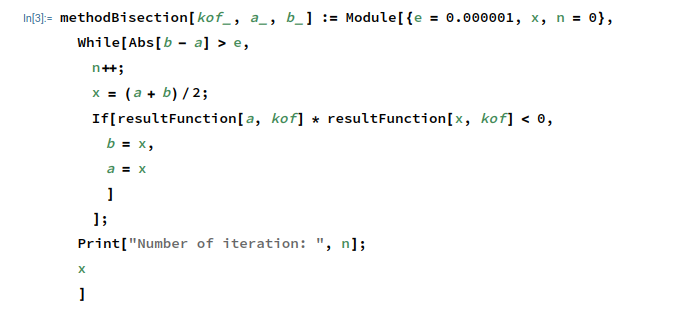
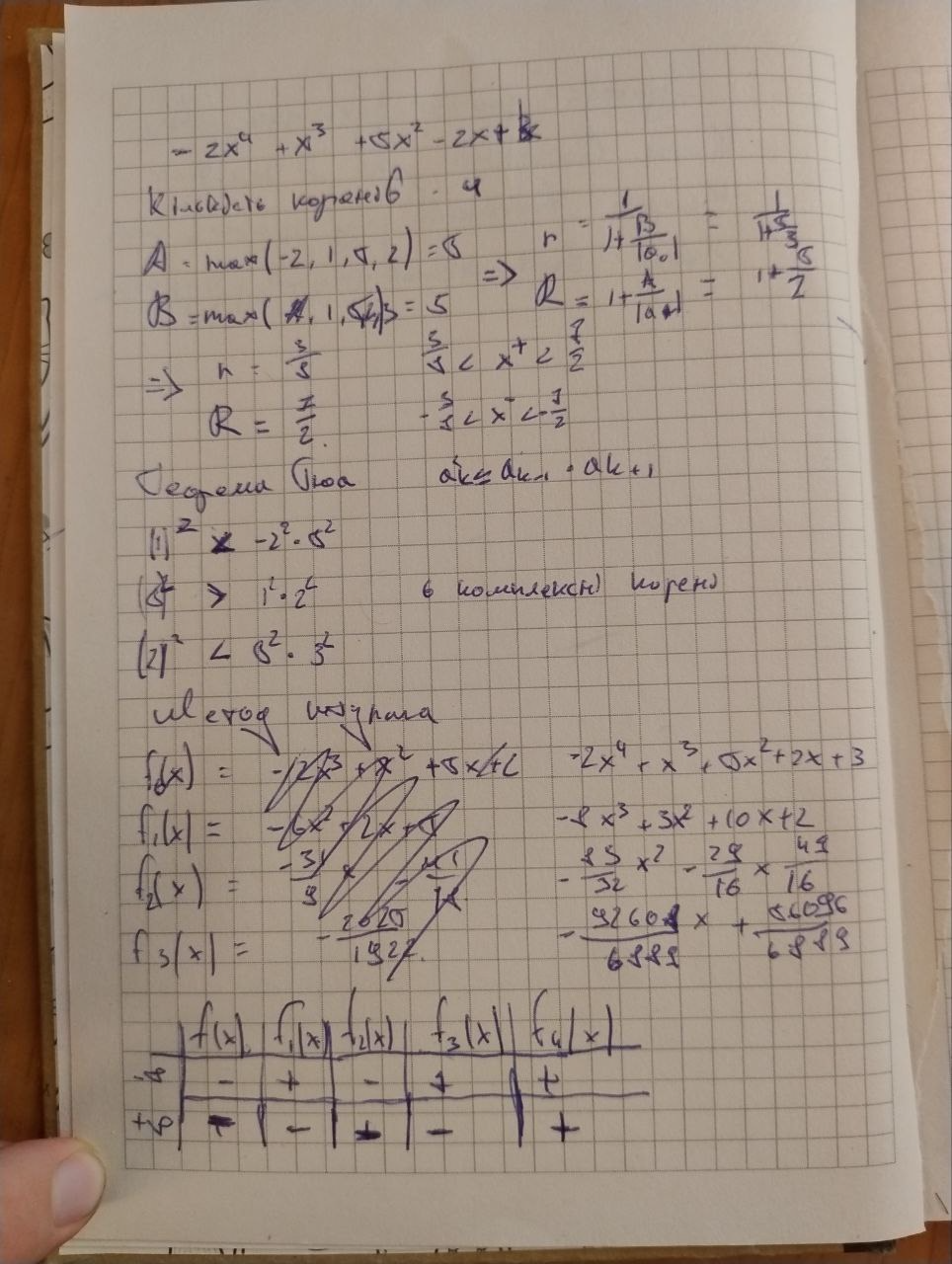
| xk - xk-1 | < e та | f(xk) | < e .

1. Порівняти отримані результати, зробити висновки, який метод приводить до меншої кількості ітерацій і чим це зумовлено.

Вигляд рівння:

16x^5-3x^4+x^3+2x^2-4x+5=0

Розв’язок:



Лістинг програми:

import math

*def* result\_function(*a*, *kof*):

result = 0

for i in range(len(kof)):

if i == len(kof) - 1:

result += kof[i]

else:

result += kof[i] \* math.pow(a, len(kof) - i - 1)

return result

*def* derivative(*kof*):

new\_kof = [0] \* (len(kof) - 1)

for i in range(len(kof) - 1):

new\_kof[i] = kof[i] \* (len(kof) - 1 - i)

return new\_kof

*def* method\_bisection(*kof*, *a*, *b*):

e = 0.000001

x = 0

n = 0

while abs(b - a) > e:

n += 1

x = (a + b) / 2

if result\_function(a, kof) \* result\_function(x, kof) < 0:

b = x

else:

a = x

print("Number of iteration:", n)

return x

*def* method\_chord(*kof*, *a*, *b*):

e = 0.000001

fkof = derivative(kof)

skof = derivative(fkof)

if result\_function((a + b) / 2, fkof) \* result\_function((a + b) / 2, skof) > 0:

x0 = b

x1 = a

else:

x0 = a

x1 = b

current\_x = 0

prev\_x = x1

n = 0

while True:

n += 1

current\_x = prev\_x - result\_function(prev\_x, kof) / (result\_function(prev\_x, kof) - result\_function(x0, kof)) \* (prev\_x - x0)

if abs(current\_x - prev\_x) < e:

break

prev\_x = current\_x

print("Number of iteration:", n)

return current\_x

*def* method\_newton(*kof*, *a*, *b*):

e = 0.000001

fkof = derivative(kof)

skof = derivative(fkof)

if result\_function((a + b) / 2, fkof) \* result\_function((a + b) / 2, skof) > 0:

prev\_x = b

else:

prev\_x = a

current\_x = 0

n = 0

while True:

n += 1

current\_x = prev\_x - result\_function(prev\_x, kof) / result\_function(prev\_x, fkof)

if abs(current\_x - prev\_x) < e:

break

prev\_x = current\_x

print("Number of iteration:", n)

return current\_x

alhpa = 3

cooficients\_lab:*float* = [0.0, -2.0, 1.0, 5.0, -2.0, 1.0]

a5, a4, a3, a2, a1, a0 = cooficients\_lab

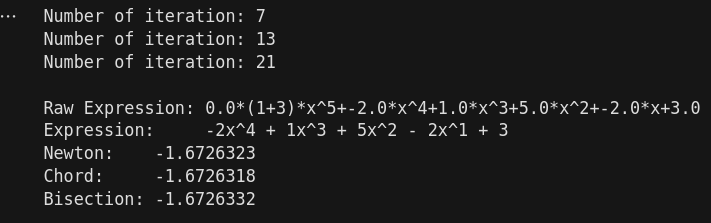
k = 3

expression = *f*"{a5}\*(1+{alhpa})\*x^5+{a4}\*x^4+{a3}\*x^3+{a2}\*x^2+{a1}\*x+{k\*a0}"

simplified = "-2x^4 + 1x^3 + 5x^2 - 2x^1 + 3"

koef = [-2, 1, 5, -2, 3]

Скріншоти виконання програми:



Висновок

У ході виконання лабораторної роботи було помічено, що метод хорд та метод бісекції схожі, але метод хорд дає швидше сходження до точки і тому у загальному випадку в нього менше ітерацій.

Кількість ітерацій при використанні методу Ньютона напряму залежить від вказаного початкового наближення. У кожному з випадків було перевірено чи однаковий знак у даного рівняння і другої похідної. Якщо знак співпадає, то початкове наближення дорівнює початку проміжку уточненого кореня, якщо ж ні, то його кінцю. При такому наближенні метод Ньютона виявився найшвидшим.