Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота № 4

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

Виконав:

студент гр. ІС-34

Колосов Ігор

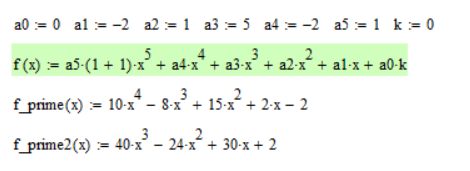
Викладач:

доц. Рибачук Л.В.

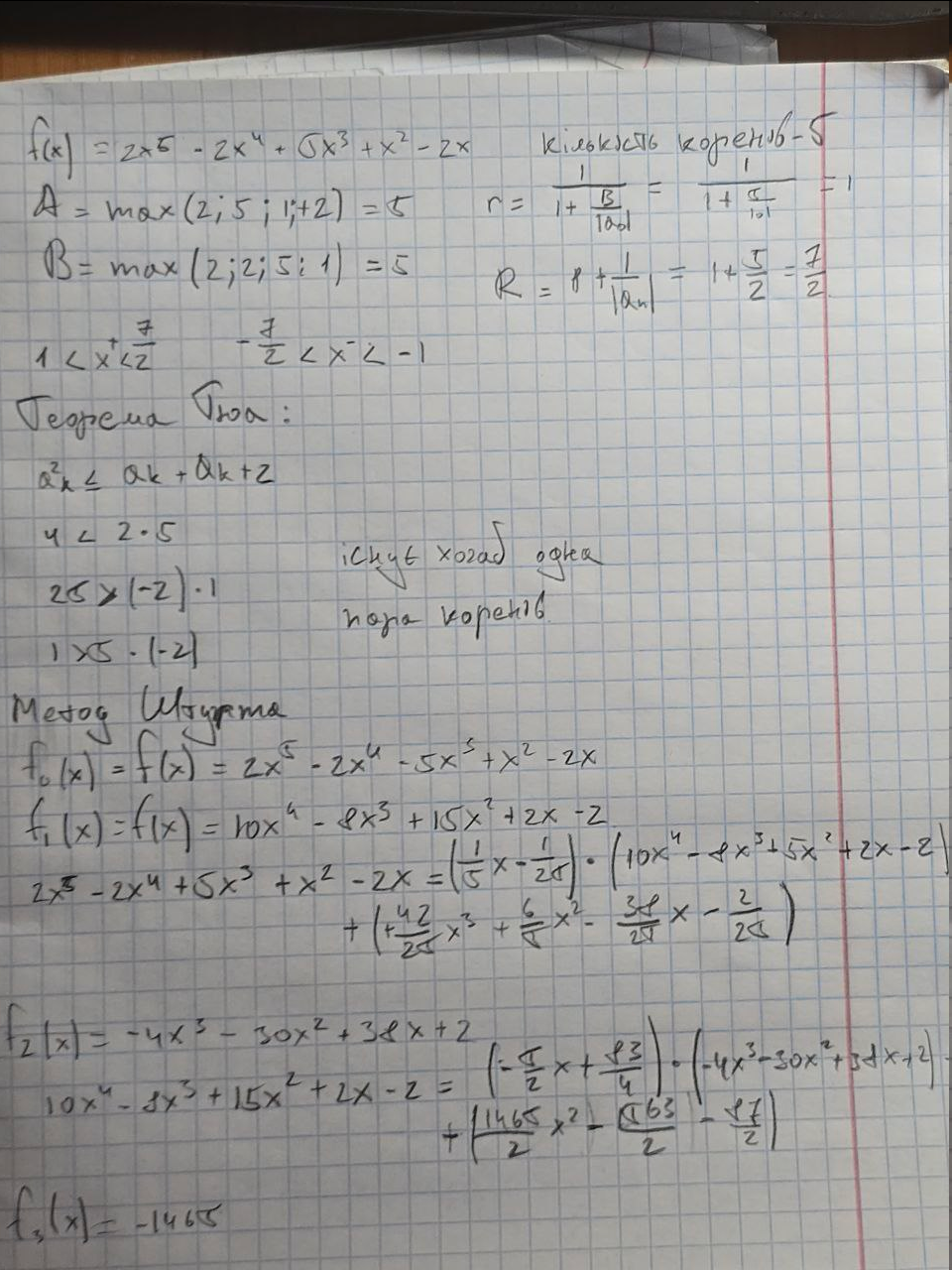
Київ – 2024

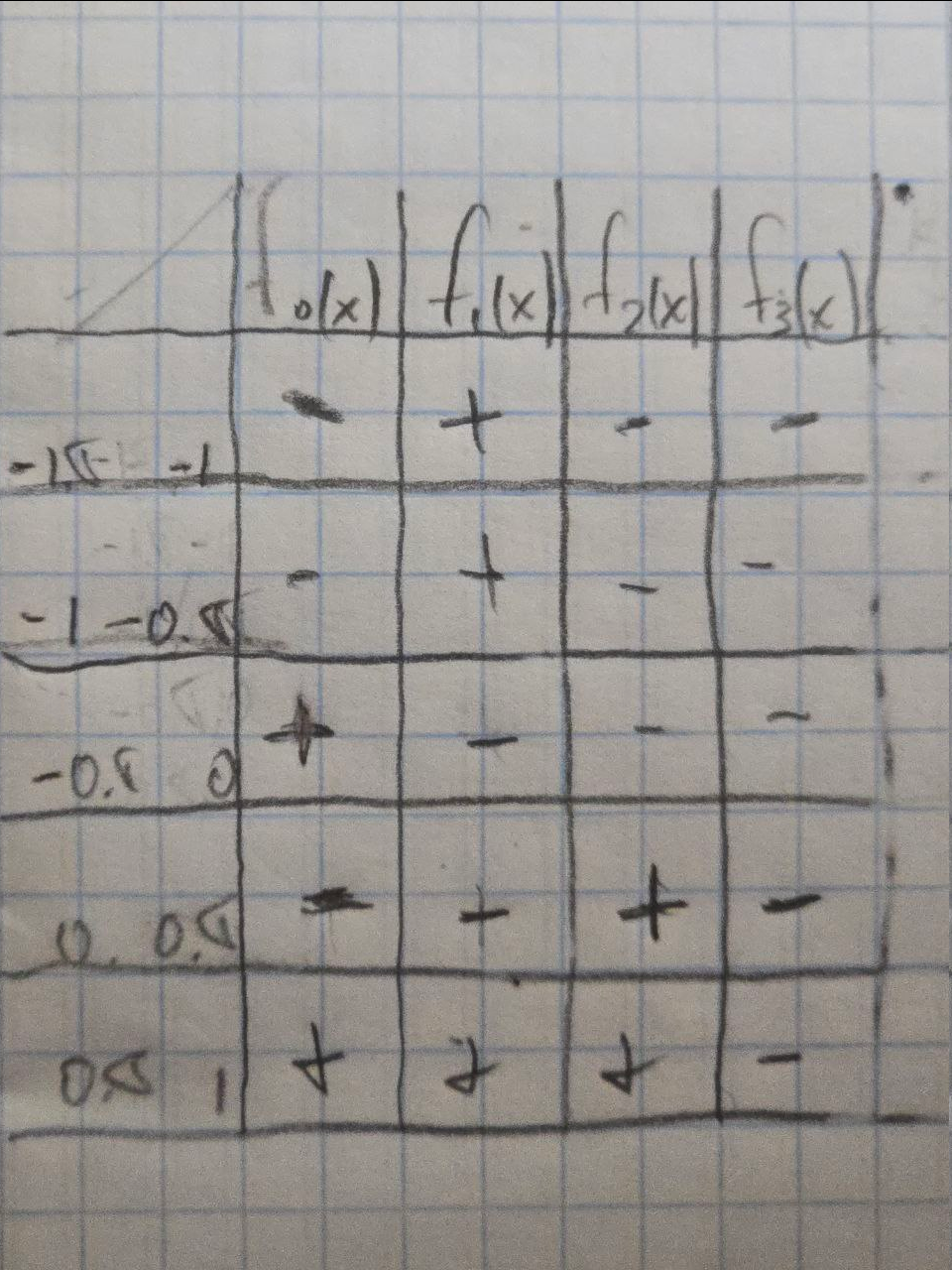
Зміст

1. **Постановка задачі у вигляді вихідного рівняння**

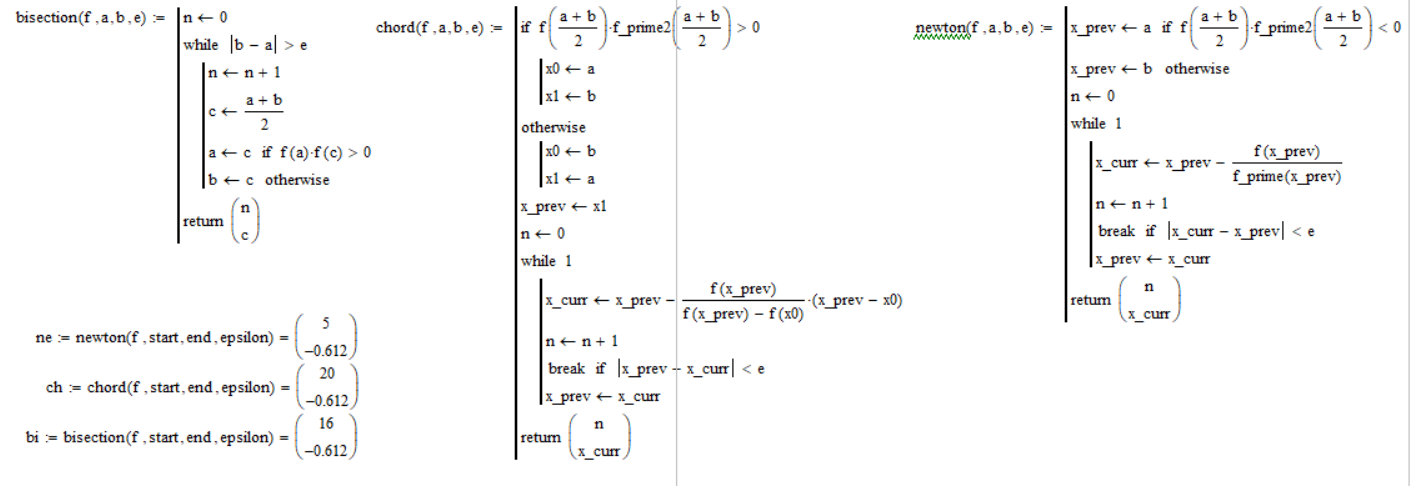


1. **Виконання допрограмового етапу, результатом якого повинні бути проміжки, щодо яких проводиться уточнення**

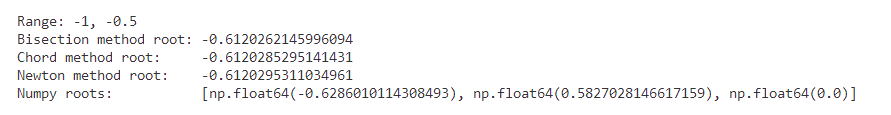
****

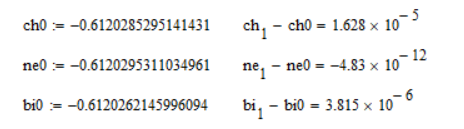
****

1. **Розв’язок уточнення коренів за методами бісекції, хорд, дотичних у Mathcad**

****

1. **Порівняння власного результату з розв’язком у Mathcad**

****

****

1. **Висновки**

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.

1. **Лістинг програми.**

| import numpy as np  def f(x, k=0, alpha=1):  a0, a1, a2, a3, a4, a5 = 0, -2, 1, 5, -2, 1  return a5\*(1+alpha)\*x\*\*5 + a4\*x\*\*4 + a3\*x\*\*3 + a2\*x\*\*2 + a1\*x + a0\*k  def bisection(f, a=-10, b=+10, eps=1e-5):  while abs(b - a) > eps:  c = (a + b) / 2  if f(c) == 0:  return c  elif f(c) \* f(a) < 0:  b = c  else:  a = c  return (a + b) / 2  def chord(f, a=-10, b=+10, eps=1e-5):  c = 1.0  while abs(f(c)) > eps:  c = (a \* f(b) - b \* f(a)) / (f(b) - f(a))  if f(c) \* f(b) > 0:  a = c  else: b = c  return c  def derivative(f, x, eps=1e-5):  return (f(x + eps) - f(x)) / eps  def newton\_method(f, x0, eps=1e-5):  x = x0  x\_old = x - 1  while abs(x\_old - x) > eps:  x\_old = x  x = x - f(x) / derivative(f, x, eps=eps)  return x  a, b = -1, -0.5  bisection\_root = bisection(f, a, b)  chord\_root = chord(f, b,a)  newton\_root = newton\_method(f, a)  coeffs = [1, -2, 5, 1, -2, 0]  np\_roots = np.roots(coeffs)  np\_roots = [root.real for root in np\_roots if root.imag == 0]  x\_vals = np.linspace(a, b, 1000)  y\_vals = f(x\_vals)  print(f"""  Range: {a}, {b}  Bisection method root: {bisection\_root}  Chord method root: {chord\_root}  Newton method root: {newton\_root}  Numpy roots: {np\_roots}  """) |
| --- |