Київський національний університет імені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №1

З дисципліни "Чисельні методи"

Тема: Розв'язування нелінійних рівнянь

Виконав студент 3-го курсу
групи ТТП-31
Рісенгін Владислав

Київ-2024

1 Постановка задачі

Варіант № 7.

Знайти розв'язок рівняння $x^4 + x^3 - 6x^2 + 20x - 16 = 0$ методами простої ітерації та релаксації.

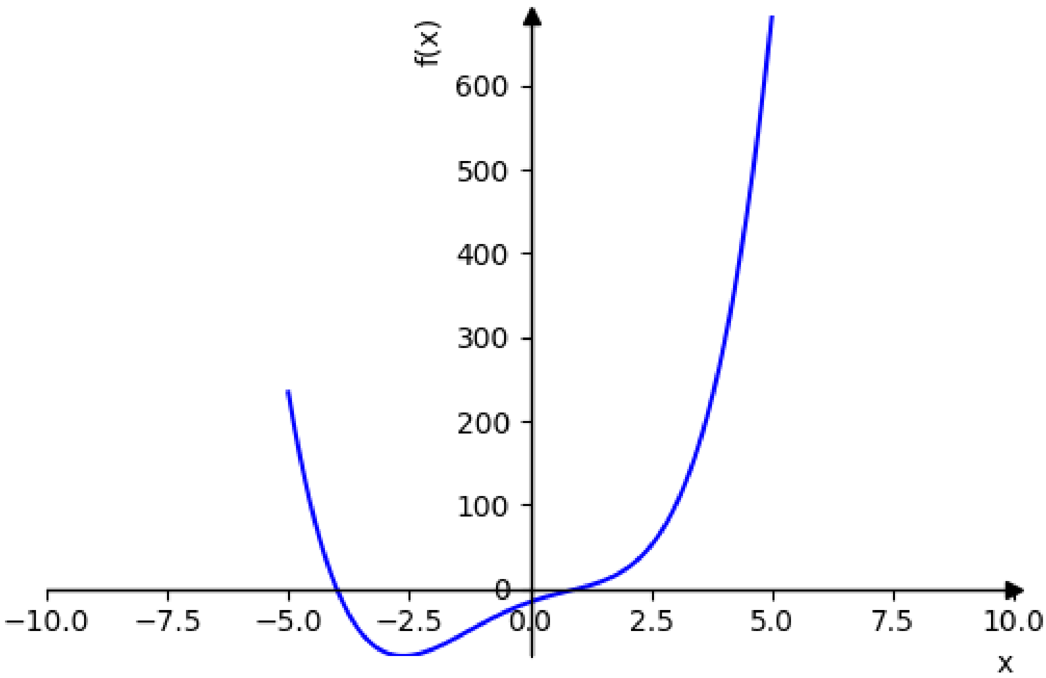

Рис. 1: Графік рівняння

2 Вступ

Метою цієї лабораторної роботи є вивчення методів розв'язування нелінійних рівнянь, які дають можливість вирішення багатьох наукових та інженерних задач. У процесі виконання роботи будуть досліджені та реалізовані метод простої ітерації та метод релаксації для знаходження коренів нелінійних рівнянь.

3 Методи які застосовувались у ході розвязання

Мова програмування: Python

$f(x) = x^4 + x^3 + -6x^2 + 20x - 16 = 0$

$\phi(x) = x + (x^4 + x^3 + -6x^2 + 20x - 16) * \psi(x) = 0$

$\psi(x) = \frac{1}{x-24}$

$f'(x) = 4x^3 + 3x^2 + -12x + 20$

4 Задача

Знайти найменший додатній корінь нелінійного рівняння
$x^4 + x^3 + -6x^2 + 20x - 16 = 0$
методом простої ітерації та методом релаксації з точністю $\epsilon = 10^{-4}$.

Знайти апріорну та апостеріорну оцінку кількості кроків.
Початковий проміжок та початкове наближення обрати однакове для обох методів (якщо) це можливо. Порівняти результати методів між собою.

4.1 Метод простої ітерації

На проміжку $[0; 1.4]$ $f(x)$ монотонно зростає, а також має різні знаки на кінцях. $f'(x)$ на цьому ж проміжку є додатньою

Оберемо проміжок $[a; b] = [0; 1.4]$

Знайдемо $x_0 = (a_0 + b_0)/2$, де $a_0 = a, b_0 = b$

$x_0 = (0 + 1.4)/2 = 0.7$

$\sigma = 0.7$

Підберемо таку функцію $\psi(x)$ щоб модуль похідної від функції $\phi(x) = x + f(x) * \psi(x)$ був < 1 на проміжку $[a; b]$
$\psi(x) = \frac{1}{x-24}$
Обчислимо $\phi(x)$ за формулою:
$\phi(x) = x + (x^4 + x^3 + -6x^2 + 20x - 16) * \psi(x) = 0$

Для визначення q знайдемо критичні точки $\phi'(x) = 0$ на $[0; 1.4]$ – 0.73011, $\phi''(0.73011) \approx -1.011$, отже це точка локального максимума.
$\phi(0.73011) \approx 0.39$

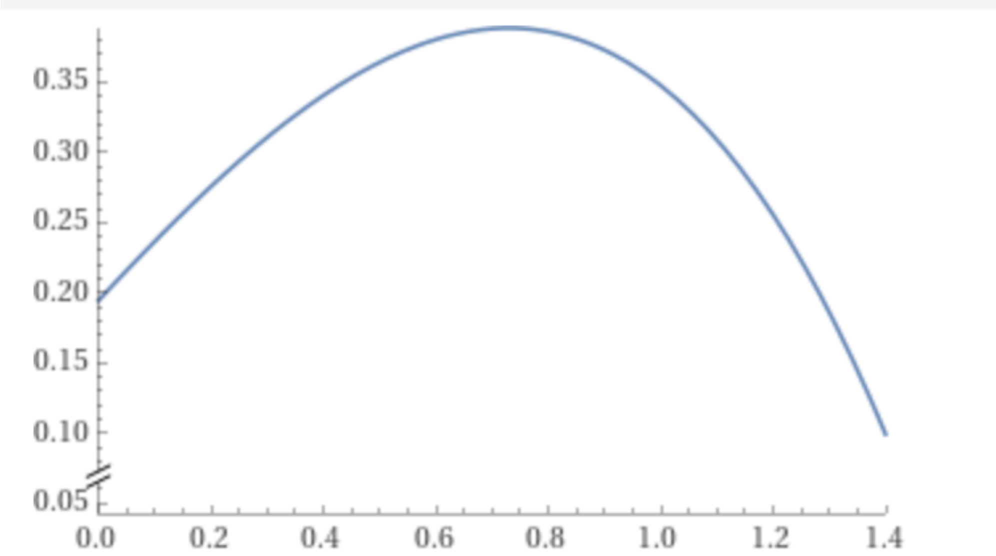

Рис. 2: $\phi'(x)$

Також перевіримо графічно див рис. 2
$q = 0.4$

Перевіримо формулу (12)

$|\phi(x_0) - x_0| <= (1 - q)\sigma$

$\phi(x_0) = 0.886991, x_0 = 0.7$, $(1 - 0.4) * 0.7 = 0.225 => 0.186991 <= 0.42$

Знайдемо апріорну оцінку:

n $\leq \left[\frac{\ln\left(\frac{\phi(x_0 - x_0)}{(1-q)*\epsilon}\right)}{\ln(1/q)}\right] + 1 = \left[\frac{\frac{\ln(0.886991 - 0.7)}{(1-0.4)*10^{-4}}}{\ln(1/0.4)}\right] + 1 = 9$
Апостеріорна оцінка обчислюється за наступною формулою:

$$|x_n - x_*| \leq \frac{q}{1-q} |x_n - x_{n-1}| .$$

4.2 Метод релаксації

На проміжку $[0; 1.4]$ $f(x)$ монотонно зростає, а також має різні знаки на кінцях. $f'(x)$ на цьому ж проміжку є додатньою

Обчисливши першу похідну і прирівнявши до нуля, отримаємо дві критичні точки 0.78 і -1.28.

Якщо перевірити за другою похідною - це будуть локальний мінімум, і локальний максимум відповідно. Отже мінімум обчислим за наступною формулою :

$m_1 = min|f'(x)| = |f'(0.78)| \approx 14.364$

А максимум за f'(a) і f'(b) адже функція зростає на (0.78; 1.4) і спадає на [0; 0.78) (через знак похідної):

$M_1 = max|f'(x)| = |f'(0)| = 20$

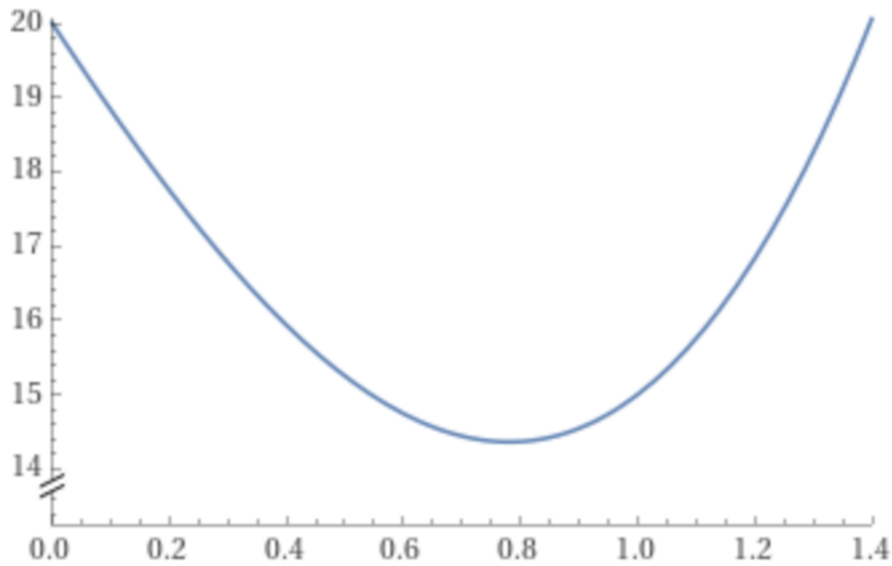

Рис. 3: Похідна функції f

Також перевіримо графічно: див рис. 3

$\tau_{opt} = \frac{2}{M_1+m_1} = \frac{2}{34} = 0.0059$

Умова збіжності $-2 < \tau f'(x) < 0$ виконується.

Додаткові обчислення:

$q = \frac{M_1-m_1}{M_1+m_1} = \frac{20-14}{20+14} = 0.176$

$z_0 = \frac{0+1.4}{2} = 0.7$

Знайдемо апріорну оцінку:

$$n \leq [\frac{ln(|z_0|/\epsilon}{ln(1/q)}] + 1 \leq [\frac{8.8536}{1.734}] + 1 = 6$$

Апостеріорна оцінка обчислюється за наступною формулою:

$$|z_n| \leq q^n * |z_0|$$

5 Результати роботи програми

```
✦ ❯ py program.py
Select algorigh to solve x^4 + x^3 - 6x^2 + 20x - 16 :
 1: Iterative method
 2: Relaxation method
 > 1
n           | x                          | f(x)                          | ao
-----------------------------------------------------------------------------------------------------
0           | 0.7                        | -4.3568999999999996           | -
1           | 0.8869914163090128         | -1.663868967695251            | 0.12466094420600857
2           | 0.9589798362358462         | -0.6105967773120646           | 0.047992279951222246
3           | 0.9854802590891854         | -0.21717890605527934          | 0.017666948568892803
4           | 0.9949168629592726         | -0.07617019679441484          | 0.006291069246724846
5           | 0.9982278788917373         | -0.026572423200374118         | 0.0022073439549764227
6           | 0.9993831126302712         | -0.00925217006948742          | 0.0007701558256893264
7           | 0.9997853701051378         | -0.0032193102743907076        | 0.00026817164991103465
8           | 0.999925338810917          | -0.001119901115444577         | 9.33124705195058e-05
9           | 0.9999740300057041         | -0.00038954789120282385       | 3.246079652467297e-05
```

Рис. 4: таблиця результатів для методу простої ітерації

```
✦ ❯ py program.py
Select algorigh to solve x^4 + x^3 - 6x^2 + 20x - 16 :
 1: Iterative method
 2: Relaxation method
 > 2
n           | x                          | f(x)                          | z
-----------------------------------------------------------------------------------------------------
0           | 0.7                        | -4.3568999999999996           | -
1           | 0.9562882352941176         | -0.6503582689822789           | 0.12352941176470589
2           | 0.994544604057781          | -0.08174246601248925          | 0.02179930795847751
3           | 0.9993529844114568         | -0.009703979294751974         | 0.003846936698554855
4           | 0.9999238067229128         | -0.0011428817422718396        | 0.0006788711820979156
5           | 0.9999910350606935         | -0.00013447384849030186       | 0.00011980079684080865
6           | 0.9999989452870753         | -1.5820690531853643e-05       | 2.114131708955447e-05
```

Рис. 5: Таблиця результатів для методу релаксації

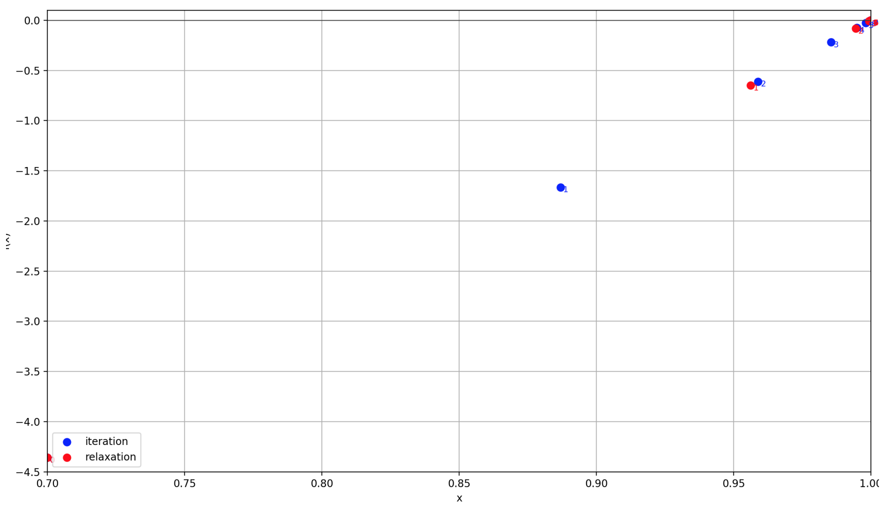

Рис. 6: Таблиця порявняння

6 Висновок

Ми отримали наближений корінь нелінійного рівняння двома різними способами:

$$\begin{aligned} \text{Методом простої ітерації:} \quad & x^* = 0.9999740300057031 \\ \text{Методом релаксації:} \quad & x^* = 0.9999989452870753 \end{aligned}$$

Проте могли зупинитись після 8 кроку для методу простої ітерації так як точність вже досягається,
для методу релаксації точність досягається після 6 кроку - як і за апріорною оцінкою.

Отже, у даному випадку метод релаксації досягнув набагато кращої точності навіть при меншій кількості кроків.
З іншої ж сторони різниця не така вже й велика, адже алгоритм на 9 та 6 кроків - це те що може бути швидко обраховане навіть вручну.
Обидва методи відпрацювали чудово.

7 Вихідний код програми

Посилання на GitHub