Київський національний університет імені Тараса Шевченка

Факультет комп’ютерних наук та кібернетики

Чисельні методи в інформатиці

Лабораторна робота №1

“Розв’язок нелінійного рівняння”

Варіант №7

Виконала студентка групи ІПС-31

Сенечко Дана Володимирівна

Київ - 2025

Постановка задачі

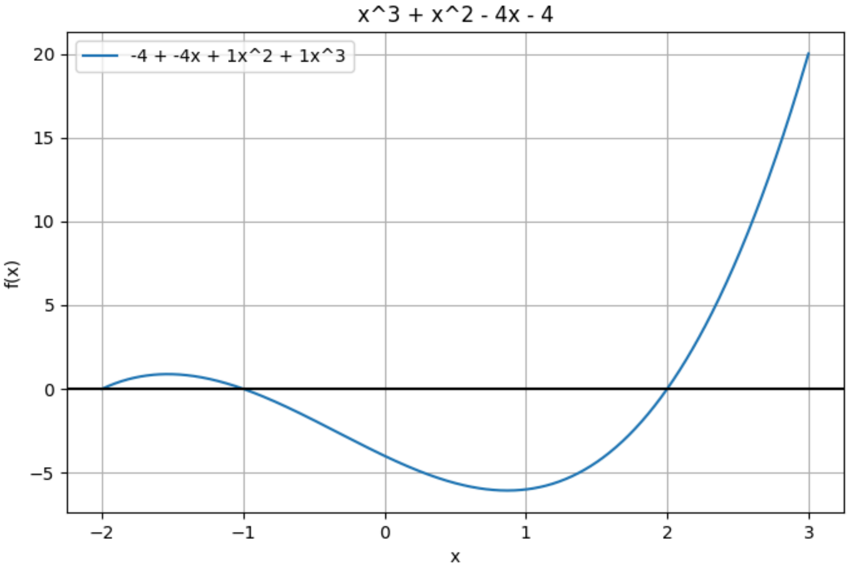
Знайти розв'язок рівняння вказаним методом з точністю $\varepsilon = 10^{-3}$. Дати можливість користувачу ввести іншу точність. Номер варіанту – 7.

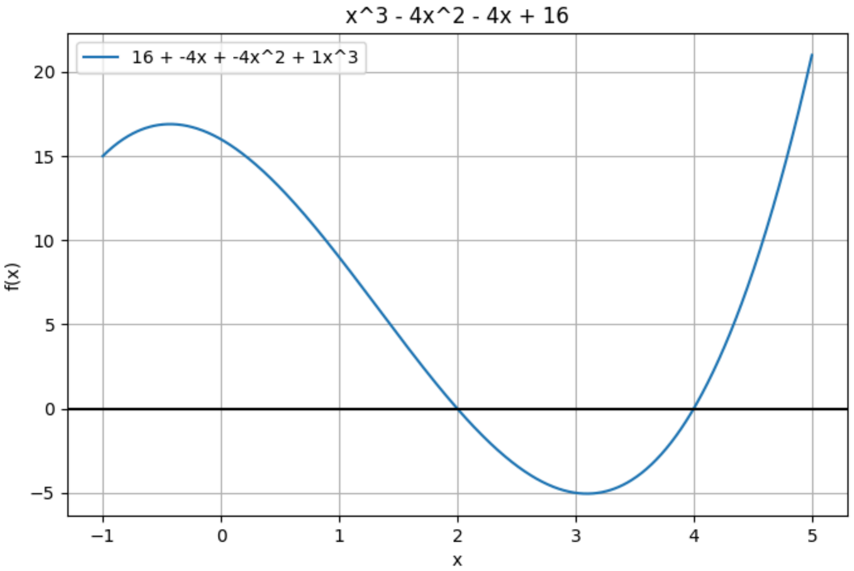
Знайти розв'язок $x^3 + x^2 - 4x - 4 = 0$ методом Ньютона.

Знайти розв'язок $x^3 - 4x^2 - 4x + 16 = 0$ методом простої ітерації.

Програма має виконувати потрібну кількість ітерацій за вказаним методом, перед цим розрахувавши їх апріорно необхідну кількість, в програмі також можна додати перевірку умов теореми або допоміжні розрахунки для звіту.

Графіки рівнянь:





Теоретичні відомості

Метод Ньютона (дотичних).

Ітераційна формула: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.

Достатні умови збіжності: $f(a)f(b) < 0$, $f''(x)$ – знакостала на проміжку, $f'(x) \neq 0$. Якщо $f(x_0)f''(x_0) > 0$, то вибір початкового наближення правильний.

Метод простої ітерації.

$x = \varphi(x) = x + \Psi(x)f(x)$, $x_{n+1} = \varphi(x_n)$.

Умова збіжності: $max|\varphi'(x)| \leq q < 1$ на інтервалі.

Обґрунтування вибору проміжку:

Якщо $f(a)f(b) < 0$ (значення на кінцях різних знаків), то за ознакою наявності кореня на відрізку існує хоча б один корінь на проміжку $[a; b]$. Вибрані проміжки задовольняють цю умову.

Апріорна кількість ітерацій.

Використовується метод ділення навпіл (дихотомія): $n \geq [log_2 \frac{b-a}{\varepsilon}]$.

При [1; 3] та $\varepsilon = 0.001$: $n > log_2 \frac{2}{0.001} = log_2(2000) \approx 10.97 \Rightarrow n = 11$.

При [1; 4] та тій самій точності: $n > log_2 \frac{3}{0.001} \approx 11.55 \Rightarrow n = 12$.

Пам'ятаємо, що апріорна оцінка – не апостеріорна, тобто реальна кількість кроків для обох алгоритмів буде меншою, адже дані методи збігаються швидше.

Хід роботи

Мова програмування – Python; використані бібліотеки: numpy (робота з масивами, векторизовані обчислення многочлена, прості перетворення), math (для розрахунку апріорної кількості ітерацій), matplotlib (для побудови графіків рівнянь).

Отримані таблиці результатів ітерацій:

```
dunnaya@MacBook-Air-Dana lab1 % python3 lab1v7.py

 Newton Method:

Enter interval (1, 3) (format a,b) or press enter:
Enter desired accuracy 0.001 or press enter:

Solving -4 + -4x + 1x^2 + 1x^3 using Newton with accuracy 0.001
Interval: (1, 3)
Initial guess: 2.000
Step    Approximation   Function Value
0       2.000           0.000
Solution found: 2.000 with accuracy 0.001
```

Метод Ньютона. Початкове наближення $x_0 = 2$. Вже на першій ітерації отримуємо точний корінь $x = 2$, бо $f(2) = 0$. Повторно запустимо програму, ввівши інший проміжок – $[1.5; 5]$:

```
dunnaya@MacBook-Air-Dana lab1 % python3 lab1v7.py

 Newton Method:

Enter interval (1, 3) (format a,b) or press enter: 1.5,5
Enter desired accuracy 0.001 or press enter:

Solving -4 + -4x + 1x^2 + 1x^3 using Newton with accuracy 0.001
Interval: (1.5, 5.0)
Initial guess: 3.250
Step    Approximation   Function Value
0       2.434           6.612
1       2.080           1.000
2       2.003           0.041
3       2.000           0.000
4       2.000           0.000
Solution found: 2.000 with accuracy 0.001
```

Тут початкове наближення $x_0 = 3.25$, за 4 ітерації знаходимо корінь $x = 2$.

```
 Simple Iteration Method:

Enter interval (1, 4) (format a,b) or press enter:
Enter desired accuracy 0.001 or press enter:

Solving 16 + -4x + -4x^2 + 1x^3 using simple iteration with accuracy 0.001
Interval: (1, 4)
Initial guess: 2.500
Step    Approximation   Function Value
0       2.837           -4.710
1       3.308           -4.803
2       3.789           -2.187
3       4.007           0.090
4       3.998           -0.019
5       4.000           0.004
6       4.000           -0.001
Solution found: 4.000 with accuracy 0.001
```

Метод простої ітерації. Початкове наближення $x_0 = 2.5$. Отримано корінь $x \approx 4$ при $f(x) \approx 0$.

Висновки

Теоретичні оцінки кількості ітерації співпали з практичними – реальна кількість кроків виявилась значно меншою за очікувану.

Код та згенеровані графіки можна переглянути на моєму [GitHub].