Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

# Лабораторна робота №1

## **Чисельні методи**

*Розв’язування нелінійних рівнянь*

Варіант 17

Виконала
студентка групи ІТ-91:

Луцай К. А.

Перевірила:

ас. Тимофєєва Ю. С.

Київ 2021

**Мета:** навчитися розв’язувати нелінійні рівняння за допомогою мови програмування Python, використати три методи (половинного ділення, ньютона, простої ітерації).

**Теоретичні відомості:**

Розв'язують рівняння у два етапи:
1) знаходження наближених значень коренів (відокремлення коренів)
2) уточнення наближених значень коренів.

Загальних методів відокремлення коренів не існує. Методи уточнення наближених значень коренів при розв'язуванні нелінійних рівнянь такого типу є прямі й ітераційні. В ітераційних методах задається процедура розв'язування у вигляді багаторазового застосування певного алгоритму. Отриманий розв'язок завжди є наближеним, хоча може бути скільки завгодно близьким до точного.

На першому етапі необхідно знайти відрізки з області визначення функції f (x) , всередині яких міститься тільки один корінь розв'язуваного рівняння. Іноді обмежуються розглядом лише певної частини області визначення, що викликає інтерес. На цьому етапі використовуються графічні або аналітичні способи. При аналітичному способі відділення коренів є корисною така теорема

Теорема 1.1. Безперервна строго монотонна функція f (x) має, причому єдиний, нуль на відрізку [a,b] тоді й тільки тоді, коли на його кінцях вона набуває значення різних знаків.

Достатньою ознакою монотонності функції f (x) на відрізку [a,b] є збереження знака похідної функції.

Графічний спосіб виділення коренів доцільно використовувати тоді, коли є можливість побудувати графік функції y = f (x). Наявність графіка вихідної функції дає безпосереднє уявлення про кількість і розташування нулів функції, що дозволяє визначити проміжки, всередині яких міститься тільки один корінь. Якщо побудувати графік функції y = f (x) складно, часто виявляється, що зручно привести рівняння (1.1) до еквівалентного вигляду $f1(x) = f2(x)$ та побудувати графіки функцій $y = f1(x)$ і $y = f2(x)$ . Абсциси точок перетину цих графіків будуть значеннями коренів розв'язуваного рівняння.

Так чи інакше, при завершенні першого етапу, необхідно визначити проміжки, на кожному з яких міститься тільки один корінь рівняння.

1. Метод половинного ділення

Алгоритм складається з таких операцій. Спочатку обчислюються значення функцій у точках, розташованих через рівні інтервали на осі x. Це робиться

доти, поки не буде знайдено два послідовні значення функції $f(x_п)$ та $f(x_п+i)$, що мають протилежні знаки (нагадаємо, якщо функція неперервна, то зміна знака вказує на існування кореня).
Потім за формулою обчислюється середнє значення x в інтервалі значень [ $x_n$, $x_п+i$ ] та відшукується значення функції $f(x_c)$. Якщо знак $f(x_c)$ збігається зі знаком $f(x_п)$, то надалі замість $f(x_п)$ використовується $f(x_c)$. Якщо ж $f(x_c)$ має знак, протилежний знаку $f(x_п)$, тобто її знак збігається зі знаком $f(x_п+i)$, то на $f(x_c)$ замінюється це значення функції. У результаті інтервал, у якому міститься значення кореня, звужується. Якщо $f(x_c)$ достатньо близьке до нуля, процес закінчується, в іншому випадку він продовжується

2. Метод Ньютона

Члени, що містять $H$, при другій та вищих степенях похідних відкидаються; використовується співвідношення $x_n + H = x_n+i$. Передбачається, що перехід від $x_n$ до $x_п+_1$ наближає значення функції до нуля так, що при деякому $n$ $f(x_n + H) = 0$. Тоді

$$x_{n+1} = x_n + h = x_n - \frac{f(x_n)}{f`(x_n)}$$

Значення $x_п+i$ відповідає точці, в якій дотична до кривої в точці $x_n$ перетинає вісь $x$. Оскільки крива $f(x)$ відрізняється від прямої, те значення функції $f(x_п+_1)$ швидше за все не буде точно дорівнювати нулю. Тому вся процедура повторюється, причому замість $x_n$ використовується $x_п+i$. Процес припиняється після досягнення достатньо малого значення $f(x_п+i)$.

3. Метод простої ітерації

Щоб застосувати цей метод, рівняння $f(x) = 0$ подається у вигляді $x = g(x)$. Відповідна ітераційна формула має вигляд

$$x_{n+1} = g(x_n)$$

Простота методу простої ітерації приваблює, проте не слід забувати, що і цей метод має вади, оскільки він не завжди забезпечує збіжність. Тому для будь-якої програми, в якій використовується цей алгоритм, необхідно передбачати контроль збіжності та припиняти розрахунок, якщо збіжність не забезпечується.

**Розв'язок задачі в аналітичній формі:**

$4^x - 5x - 2 = 0$

$f(x) = 4^x - 5x - 2$

$f`(x) = 4^x ln4 - 5$ похідна

$4^x ln4 - 5 = 0$ точка у якій похідна функції змінює знак, отже один з коренів знаходиться зліва, а інший – справа від неї.

Отримуємо два інтервали для перевірки:

$(-\infty,\ \log_4 \frac{5}{\ln 4})$

$(\log_4 \frac{5}{\ln 4}, \infty+)$

Оскільки координати абцис коренів функції імовірно знаходяться в межах 100 від початку координат, то замість нескінченності в інтервал доцільно поставити 100, а після першої перевірки навіть зменшити це число до 10.

$f\left(\log_4 \frac{5}{\ln 4}\right) < 0$

f(10) > 0, f(-10) > 0

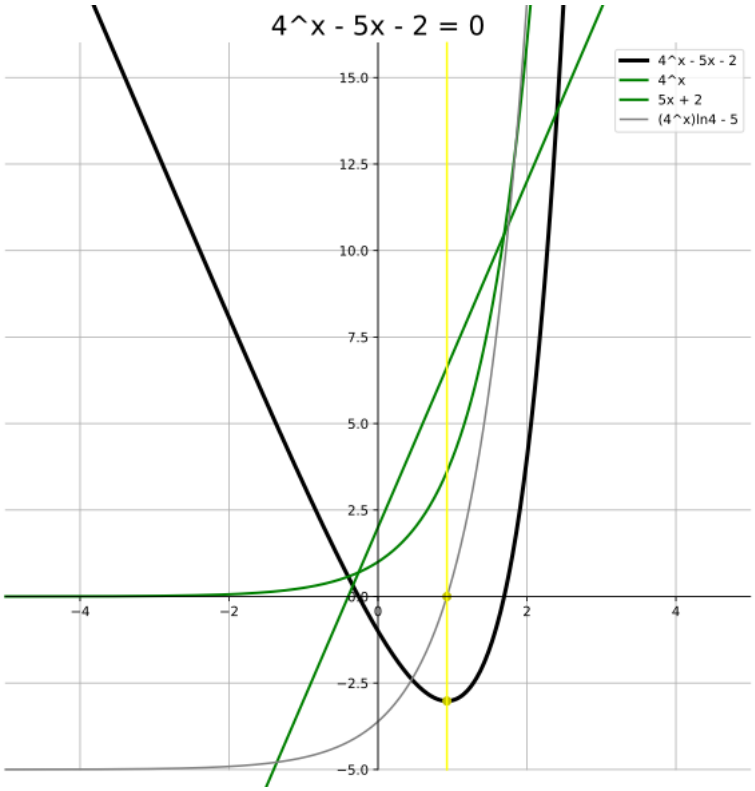
тож кінцевими інтервалами для уточнення коренів будуть:

$(-10,\ \log_4 \frac{5}{\ln 4})$

$(\log_4 \frac{5}{\ln 4}, 10)$

Перевіримо це за допомогою графічного способу:

$4^x = 5x + 20$




Чорним показано основну функцію;
Зеленим її представлення через дві інші функції;
Сірим похідну основної функції;
Жовтим точку, у якій, похідна дорівнює нулю, та лінію що розмежовую координатну площину на два інтервали, у кожному з яких знаходиться лише один корінь.

Координати абцис перетину двох функцій знаходяться в межах інтервалу (-2, 2), отже попередні розрахунки вірні. Тепер можна приступати до уточнення за допомогою обчислювальних алгоритмів конкретних виділених коренів із вказаною точністю.

**Лістинги програм розв'язування задачі:**

В якості аргументів методи приймають початок та кінець інтервалу для пошуку кореня, степінь уточнення (кількість знаків після коми). Повертають пару чисел у вигляді x, y координат округлених до обраного знаку після коми.

```
import matplotlib.pyplot as plt
import math
import numpy as np

def poldil(a, b, n):
    xRes, eps = interval(a, b, n)

    while xRes[1] - xRes[0] > eps:
        xS = (xRes[0]+xRes[1])/2.0
        if func17(xS) * func17(xRes[0]) > 0:
            xRes[0] = xS
        else:
            xRes[1] = xS

    print("Проміжок x координати:", xRes)
    yRes = [func17(xRes[0]), func17(xRes[1])]
    print("Проміжок y координати:", yRes)
    return round(xS, n), round(func17(xS), n)

def newton(a, b, n):
    xRes, eps = interval(a, b, n)

    x = xRes[0]
    h = func17(x)/der17(x)
    while math.fabs(h) > eps:
        h = func17(x)/der17(x)
        x -= h

    print("Наближення на", h, "при похідній", der17(x))
    return round(x, n), round(func17(x), n)

def simpiter_1(a, b, n):
    xRes, eps = interval(a, b, n)

    xS = (xRes[0] + xRes[1])/2.0
    x = lin17_1(xS)
    while math.fabs(x - xS) >= eps:
        xS = x
        x = lin17_1(x)

    print("Значення x = math.log(5*x + 2, 4) подання:", lin17_1(x))
    return round(x, n), round(func17(x), n)
```

```
def simpiter_2(a, b, n):
    xRes, eps = interval(a, b, n)

    xS = (xRes[0] + xRes[1])/2.0
    x = lin17_2(xS)
    while math.fabs(x - xS) >= eps:
        xS = x
        x = lin17_2(x)

    print("Значення x = (4**x - 2)/5 подання:", lin17_2(x))
    return round(x, n), round(func17(x), n)
```

Допоміжна функція визначення інтервалу для уточнення коренів:

```
def interval(a, b, n):
    eps = 10**-n
    x1 = a
    y1 = func17(x1)
    for x2 in range(a+1, b+1):
        y2 = func17(x2)
        if y1 * y2 > 0:
            y1 = y2
            x1 = x2
        else:
            break
    xRes = [x1, x2]
    print("Проміжок для пошуку кореня:", xRes, "з точністю до", eps)
    return xRes, eps
```

Функції використані у методах наведених вище:

```
def func17(x):
    return 4**x - 5*x - 2

def der17(x):
    return (4**x)*math.log(4) - 5

def lin17_1(x):
    return math.log(5*x + 2, 4)

def lin17_2(x):
    return (4**x - 2)/5
```

Функція для побудови графіка та коренів:
Приймає координати обох коренів, колір та назву графіку.

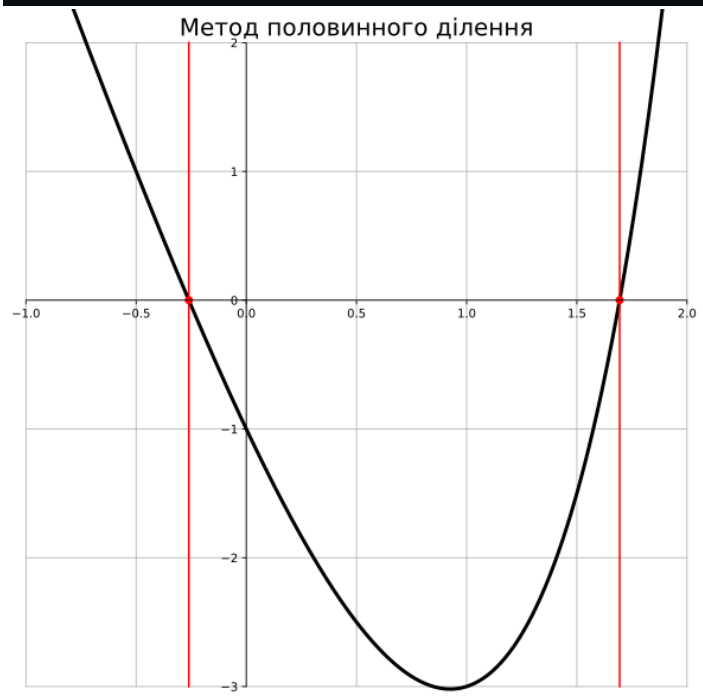
```
def graph(x1, y1, x2, y2, clr, ttl):
    x = np.linspace(-5, 5, 1000)
    fig = plt.figure(figsize=(10, 10))
    ax = plt.subplot(ylim=(-3, 2), xlim=(-1, 2))
    ax.plot(x, 4**x - 5*x - 2, color="black", lw=3)
    ax.plot(x1, y1, clr[0]+"o")
    ax.plot(x2, y2, clr[0]+"o")
    ax.grid(True)
    ax.spines["left"].set_position("zero")
    ax.spines["right"].set_color("none")
    ax.spines["bottom"].set_position("zero")
    ax.spines["top"].set_color("none")
    plt.title(ttl , fontsize=20)
    plt.axvline(x1, color=clr)
    plt.axvline(x2, color=clr)
    plt.show()
```

Виклик методів для уточнення коренів:
Було обрано два інтервали для пошуку коренів.

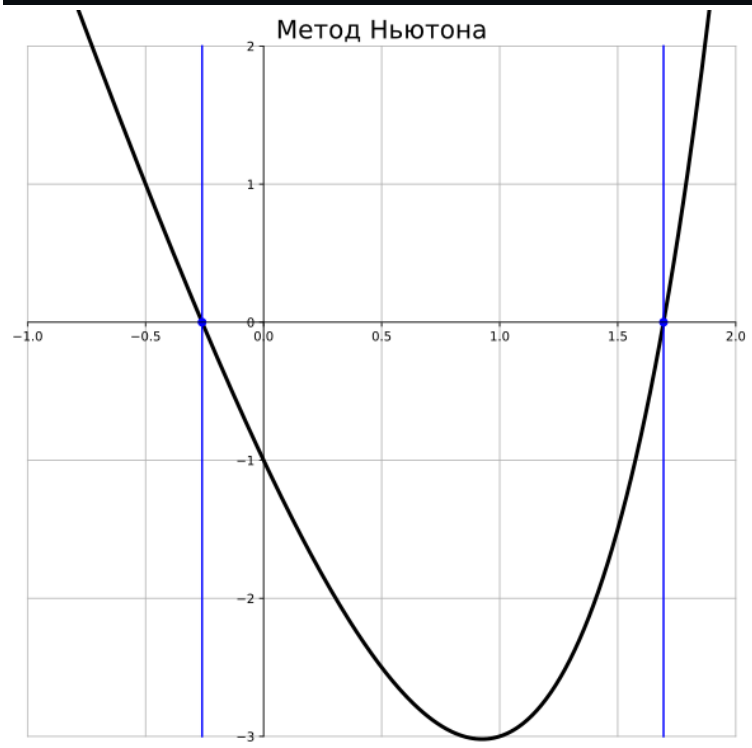
```
xGorb = math.log(5/math.log(4), 4)

x1, y1 = poldil(-10, int(xGorb), 4)
print("Корінь x1:", x1, y1)
x2, y2 = poldil(int(xGorb), 10, 4)
print("Корінь x2:", x2, y2)
graph(x1, y1, x2, y2, "red", "Метод половинного ділення")

x1, y1 = newton(-10, int(xGorb), 4)
print("Корінь x1:", x1, y1)
x2, y2 = newton(int(xGorb), 10, 4)
print("Корінь x2:", x2, y2)
graph(x1, y1, x2, y2, "blue", "Метод Ньютона")

x1, y1 = simpiter_2(-10, int(xGorb), 4)
print("Корінь x1:", x1, y1)
x2, y2 = simpiter_1(int(xGorb), 10, 4)
print("Корінь x2:", x2, y2)
graph(x1, y1, x2, y2, "green", "Метод простої ітерації")
```

**Результати виконання програм:**

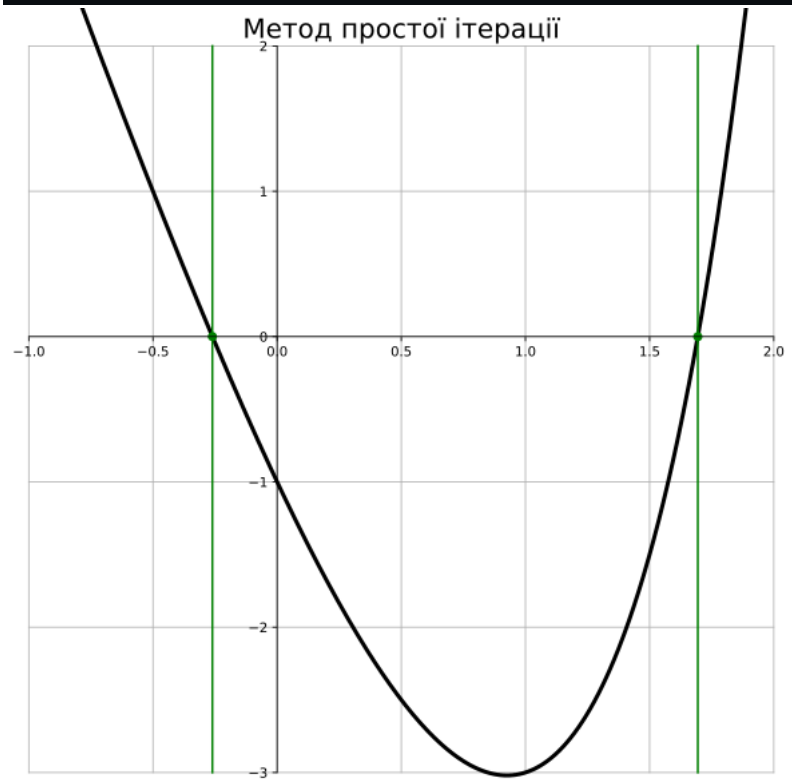
```
Проміжок для пошуку кореня: [-1, 0] з точністю до 0.0001
Проміжок x координати: [-0.26068115234375, -0.2606201171875]
Проміжок y координати: [0.00011939306745700762, -0.00012682940317287894]
Корінь x1: -0.2607 0.0001
Проміжок для пошуку кореня: [1, 2] з точністю до 0.0001
Проміжок x координати: [1.694091796875, 1.69415283203125]
Проміжок y координати: [-0.0005021170757828486, 7.863586834844227e-05]
Корінь x2: 1.6942 0.0001
```




```
Проміжок для пошуку кореня: [-1, 0] з точністю до 0.0001
Наближення на -1.1593342921737182e-08 при похідній -4.034110193436775
Корінь x1: -0.2607 0.0
Проміжок для пошуку кореня: [1, 2] з точністю до 0.0001
Наближення на 5.345360472872561e-05 при похідній 9.51550409164349
Корінь x2: 1.6941 0.0
```



```
Проміжок для пошуку кореня: [-1, 0] з точністю до 0.0001
Значення x = (4**x - 2)/5 подання: -0.26065353415156534
Корінь x1: -0.2607 0.0
Проміжок для пошуку кореня: [1, 2] з точністю до 0.0001
Значення x = math.log(5*x + 2, 4) подання: 1.6941302919238068
Корінь x2: 1.6941 -0.0004
```




**Висновки:** було написано програму для розв’язування нелінійного рівняння на мові Python: було виділено корені за допомогою аналітичного та графічного методів, було реалізовано три програмних методи для уточнення цих коренів.