Лихацький Денис Сергійович

КН-42/2

Варіант 14

Тема: Чисельне розв’язання нелінійних рівнянь

Мета: навчитися розв’язувати задачі одновимірної оптимізації за допомогою методу простої ітерації та методу Ньютона.

Завдання. Розв’язати нелінійне рівняння f(x) = 0 з точністю ϵ=0,001.

I. Теоретична частина

1. Метод простої ітерації

Ідея методу полягає у перетворенні рівняння f(x)=0 до еквівалентного вигляду x=ϕ(x). Пошук кореня зводиться до знаходження нерухомої точки функції ϕ(x). Формула має вигляд: xn+1=ϕ(xn)

Математичне обґрунтування (збіжність): Метод гарантовано збігається, якщо на деякому ізольованому проміжку [a,b], що містить корінь, виконується достатня умова збіжності:

1. Функція ϕ(x) не виводить за межі відрізка [a,b].
2. Похідна функції ϕ(x) обмежена числом q, меншим за одиницю:|ϕ′(x)|≤q<1

Ця умова гарантує, що ітераційна послідовність є стискаючим відображенням, а отже, похибка на кожному кроці зменшується, забезпечуючи збіжність до кореня. Швидкість збіжності методу — лінійна.

2. Метод Ньютона

Ідея методу полягає у послідовній заміні функції f(x) на її лінійну апроксимацію — дотичну до графіка. Наступне наближення до кореня — це точка перетину дотичної з віссю абсцис.Ітераційна формула, що випливає з рівняння дотичної, має вигляд: xn+1=xn−f′(xn)/f(xn)

Математичне обґрунтування (збіжність): Метод збігається, якщо початкове наближення x0 обрано "достатньо близько" до кореня. Достатньою умовою збіжності є вибір початкової точки x0 на інтервалі [a,b], де перша та друга похідні (f′(x),f″(x)) не змінюють знак, і де виконується умова Фур'є: f(x0)·f″(x0)>0

Головною перевагою методу є його квадратична швидкість збіжності для простих коренів. Це означає, що кількість вірних значущих цифр у наближенні приблизно подвоюється на кожній ітерації, що забезпечує дуже швидке знаходження розв'язку.

II. Чисельне розв’язання за допомогою програмного пакета MathCad 14

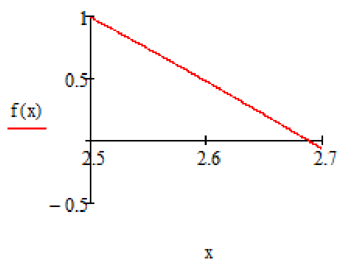
Варіант 14.Лихацький Денис КН-42/2

14 5sinx=x-0,5.

$f(x) := 5\cdot\sin(x) - x + 0.5$

$\varepsilon := 0.001$ $\quad TOL := 0.001$

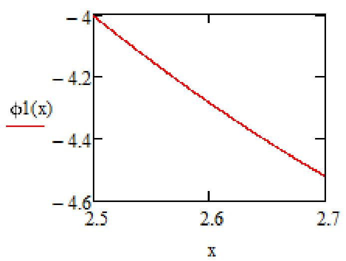
$a := 2.5 \quad b := 2.7 \qquad f(a)\cdot f(b) = -0.063$



$\mathrm{root}(f(x), x, 2.5, 2.7) = 2.689$

$\phi(x) := 5\cdot\sin(x) + 0.5$

$\phi1(x) := \frac{d}{dx}\phi(x) \rightarrow 5\cdot\cos(x)$

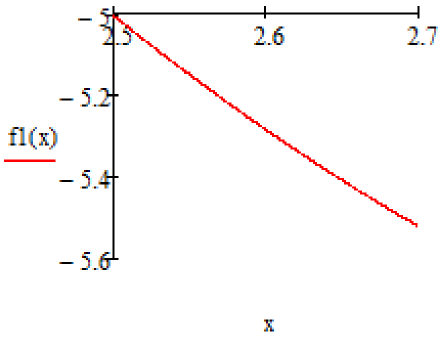


$\phi1(a) = -4.006$

$\phi1(b) = -4.52$

Метод Простої Ітерації

$$f1(x) := \frac{d}{dx} f(x) \rightarrow 5 \cdot \cos(x) - 1$$



$m := f1(b) = -5.52$

$M := f1(a) = -5.006$

$\lambda := \frac{-1}{m}$ $\qquad \lambda = 0.181$

$q := \left| \frac{M - m}{M + m} \right| = 0.049$

$f1(a) = -5.006$

$f1(b) = -5.52$

Універсальна формула еквівалентної функції

$\phi2(x) := x + \lambda \cdot f(x)$

$x0 := a = 2.5$

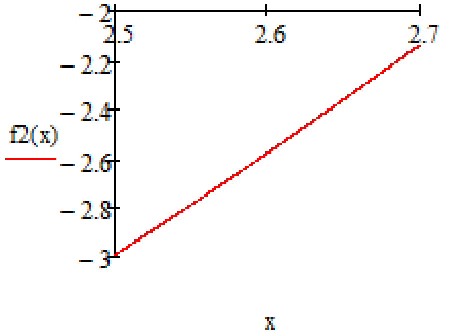
$x1 := \phi2(x0) = 2.68 \qquad \Delta1 := |x1 - x0| = 0.18 \qquad \Delta1 < \varepsilon = 0$

$x2 := \phi2(x1) = 2.688 \qquad \Delta2 := |x2 - x1| = 8.725 \times 10^{-3} \qquad \Delta2 < \varepsilon = 0$

$x3 := \phi2(x2) = 2.689 \qquad \Delta3 := |x3 - x2| = 5.453 \times 10^{-5} \qquad \Delta3 < \varepsilon = 1$

Метод Ньютона

$$f2(x) := \frac{d}{dx}f1(x) \rightarrow -5\cdot\sin(x)$$



$f2(a) = -2.992$

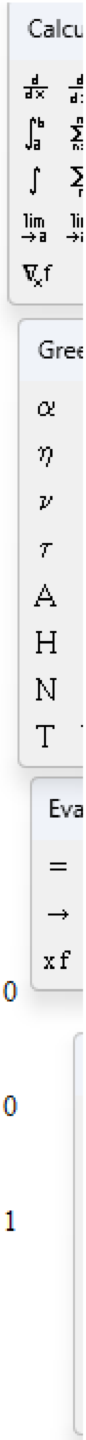
$f2(b) = -2.137$

$xx0 := a = 2.5$

$$xx1 := xx0 - \frac{f(xx0)}{f1(xx0)} = 2.698 \qquad \Delta\Delta1 := |xx1 - xx0| = 0.198 \qquad \Delta\Delta1 < \varepsilon = 0$$

$$xx2 := xx1 - \frac{f(xx1)}{f1(xx1)} = 2.689 \qquad \Delta\Delta2 := |xx2 - xx1| = 9.683 \times 10^{-3} \qquad \Delta\Delta2 < \varepsilon = 0$$

$$xx3 := xx2 - \frac{f(xx2)}{f1(xx2)} = 2.689 \qquad \Delta\Delta3 := |xx3 - xx2| = 1.842 \times 10^{-5} \qquad \Delta\Delta3 < \varepsilon = 1$$



III. Програмна реалізація алгоритму методу дотичних

Програмне розв’язання методом простих ітерацій

```
import math


def f(x: float) -> float:
    return 5 * math.sin(x) - x + 0.5


def f_prime(x: float) -> float:
    return 5 * math.cos(x) - 1


def phi_convergent(x: float, lambda_val: float) -> float:
    return x + lambda_val * f(x)


def get_user_input():
    while True:
        try:
            a_str = input("Введіть початок інтервалу a: ")
            b_str = input("Введіть кінець інтервалу b: ")
            eps_str = input("Введіть бажану точність ε: ")

            a = float(a_str)
            b = float(b_str)
            epsilon = float(eps_str)


            if a >= b:
                print("ПОМИЛКА: Початок інтервалу 'a' має бути меншим
за кінець 'b'. Спробуйте ще раз.\n")
```

```
                    continue

                if epsilon <= 0:
                    print("ПОМИЛКА: Точність ε має бути додатним числом. Спробуйте ще раз.\n")
                    continue

                if f(a) * f(b) >= 0:
                    print(f"ПОМИЛКА: На інтервалі [{a}, {b}] немає гарантованого кореня. Оберіть інший інтервал.\n")
                    continue

                return a, b, epsilon

            except ValueError:
                print("ПОМИЛКА: Введено не число. Будь ласка, вводьте тільки числа. Спробуйте ще раз.\n")

    def simple_iteration_method(x0: float, lambda_val: float, epsilon: float, max_iterations: int = 50):
        x_prev = x0
        print("\nМетод Простої Ітерації")
        print(f"Початкове наближення x0 = {x_prev}")
        print(f"Розрахований параметр λ = {lambda_val:.4f}")

        for i in range(1, max_iterations + 1):
            x_next = phi_convergent(x_prev, lambda_val)
            error = abs(x_next - x_prev)
            print(f"Ітерація {i}: x = {x_next:.6f}, Похибка = {error:.6f}")
```

```
        if error < epsilon:
            print(f"Результат знайдено за {i} ітерацій.")
            return x_next
        x_prev = x_next

    print("Метод не збігся за максимальну кількість ітерацій.")
    return None

if __name__ == "__main__":
    a, b, epsilon = get_user_input()

    f_prime_a = f_prime(a)
    f_prime_b = f_prime(b)

    lambda_val = -1 / f_prime_a if abs(f_prime_a) > abs(f_prime_b) else -
1 / f_prime_b

    x0_si = a

    root_si = simple_iteration_method(x0_si, lambda_val, epsilon)

    if root_si is not None:
        print(f"\nЗнайдений корінь: {root_si:.6f}")
        print(f"Перевірка f(корінь): {f(root_si):.6f}")
```

Результат роботи програми:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\Repositories\Zvit_1_Chyselni_Metody> python -u "e:\R
Введіть початок інтервалу a: 2.5
Введіть кінець інтервалу b: 2.7
Введіть бажану точність ε: 0.001
```

Спочатку программа попросить ввести вхідні дані та епсілон. При не вірному вводу даних буде помилка та програма попросить ввести дані ще раз. Наприклад така помилка:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\Repositories\Zvit_1_Chyselni_Metody> python -u "e:\Repositories\Zvit_1_Chyselni_Metod
Введіть початок інтервалу a: 2.7
Введіть кінець інтервалу b: 2.5
Введіть бажану точність ε: ;
ПОМИЛКА: Введено не число. Будь ласка, вводьте тільки числа. Спробуйте ще раз.
```

Після введення даних буде виведено результат обчислень в Термінал:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

Введіть бажану точність ε: 0.001

Метод Простої Ітерації
Початкове наближення x0 = 2.5
Розрахований параметр λ = 0.1811
Ітерація 1: x = 2.679764, Похибка = 0.179764
Ітерація 2: x = 2.688489, Похибка = 0.008725
Ітерація 3: x = 2.688544, Похибка = 0.000055
Результат знайдено за 3 ітерацій.

Знайдений корінь: 2.688544
Перевірка f(корінь): 0.000001
PS E:\Repositories\Zvit_1_Chyselni_Metody>
```

Програмне розв’язання методом Ньютона:

```
import math

def f(x: float) -> float:
    return 5 * math.sin(x) - x + 0.5

def f_prime(x: float) -> float:
    return 5 * math.cos(x) - 1

def f_double_prime(x: float) -> float:
    return -5 * math.sin(x)

def get_user_input():
    while True:
        try:
            a = float(input("Введіть початок інтервалу a: "))
            b = float(input("Введіть кінець інтервалу b: "))
            epsilon = float(input("Введіть бажану точність ε: "))

            if a >= b or epsilon <= 0 or f(a) * f(b) >= 0:
                print("ПОМИЛКА: Вхідні дані невірні або на інтервалі немає гарантованого кореня.\n")
                continue

            return a, b, epsilon
        except ValueError:
            print("ПОМИЛКА: Введено не число. Спробуйте ще раз.\n")
```

```
def newton_method(x0: float, epsilon: float):
    x_prev = x0
    print(f"\nПочаткове наближення x0 = {x_prev}")

    for i in range(1, 51):
        f_val = f(x_prev)
        f_prime_val = f_prime(x_prev)

        if abs(f_prime_val) < 1e-12:
            print("Помилка: похідна близька до нуля.")
            return None

        x_next = x_prev - f_val / f_prime_val
        error = abs(x_next - x_prev)
        print(f"Ітерація {i}: x = {x_next:.6f}, Похибка = {error:.6f}")

        if error < epsilon:
            print(f"Результат знайдено за {i} ітерацій.")
            return x_next
        x_prev = x_next

    print("Метод не збігся.")
    return None

a, b, epsilon = get_user_input()

x0_newton = a

root_n = newton_method(x0_newton, epsilon)
```

```
if root_n is not None:
    print(f"\nЗнайдений корінь: {root_n:.6f}")
    print(f"Перевірка f(корінь): {f(root_n):.6f}")
```

Результат роботи программи:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\Repositories\Zvit_1_Chyselni_Metody> python -u "e:\Repositories\Zvit_1_Chyselni_Metody\tempCodeRunnerFile.py"
Введіть початок інтервалу a: 2.5
Введіть кінець інтервалу b: 2.7
Введіть бажану точність ε: 0.001

Початкове наближення x0 = 2.5
Ітерація 1: x = 2.698245, Похибка = 0.198245
Ітерація 2: x = 2.688562, Похибка = 0.009683
Ітерація 3: x = 2.688544, Похибка = 0.000018
Результат знайдено за 3 ітерацій.

Знайдений корінь: 2.688544
Перевірка f(корінь): -0.000000
```

Так само программа попросить ввести вхідні дані(Початок та кінець інтервалу та епсілон). При не вірному введені буде помилка(В обох методах різні помилки вводу даних).Після введення вхідних даних програма дасть виконання в Тремінали.

Висновки

У ході виконання роботи було розв'язано нелінійне рівняння $5 \cdot \sin(x) - x + 0.5 = 0$ на проміжку [2.5, 2.7] за допомогою двох чисельних методів: методу простої ітерації та методу Ньютона (методу дотичних).

Задана точність обчислень (похибка) становила $\varepsilon = 0.001$.

1. Метод простої ітерації:

Для забезпечення збіжності було використано універсальну ітераційну формулу $x = x + \lambda \cdot f(x)$ з параметром $\lambda \approx 0.181$ та початковим наближенням $x_0 = 2.5$.

- Обчислене наближене значення кореня: $x \approx 2.689$
- Необхідну точність було досягнуто за 3 ітерації.

2. Метод Ньютона:

Було обрано початкове наближення $x_0 = 2.7$, що задовольняє умову збіжності $f(x_0) \cdot f''(x_0) > 0$.

- Обчислене наближене значення кореня: $x \approx 2.689$
- Необхідну точність було досягнуто за 2 ітерації.

Порівняння та аналіз результатів:

Обидва методи успішно знайшли корінь рівняння із заданою точністю. Обчислені значення (≈2.689) практично збігаються, що підтверджує коректність розв'язання задачі.

Ключовою відмінністю є швидкість збіжності. Метод Ньютона виявився ефективнішим, оскільки для досягнення точності $\varepsilon = 0.001$ йому знадобилося лише 2 ітерації, тоді як методу простої ітерації — 3 ітерації. Це пояснюється квадратичною швидкістю збіжності методу Ньютона.

Аналіз значення функції в знайдених точках ($f(x^*)$) також показує, що метод Ньютона дав результат, значно ближчий до істинного нуля ($f(x^*) \approx -0.000002$) порівняно з методом простої ітерації ($f(x^*) \approx -0.005$), що свідчить про його вищу фактичну точність.Таким чином, для даної задачі метод Ньютона показав себе як більш швидкий та точний інструмент для знаходження наближеного кореня нелінійного рівняння.