Комп'ютерний практикум №1
Розв'язання нелінійних рівнянь

Виконав:
Студент 3 курсу ФТІ
групи ФІ-92
Поночевний Назар Юрійович
Варіант 14

Завдання 1:
Здійснити в якості допрограмового етапу аналіз та відокремлення коренів за допомогою теорем. Зокрема, визначити кількість дійсних коренів рівняння (теорема Гюа, теорема Штурма), відокремити дійсні корені рівняння (теорема про верхню межу). До аналізу комплексних коренів застосувати теорему про кільце. Результатом цього етапу повинна бути послідовність проміжків, кожен із яких містить лише один дійсний корінь рівняння;

$-x^4 + 3x^3 - 2x + 4 = 0$

Теорема про кільце: $A = 4$, $B = 3$

$$\frac{|a_0|}{B + |a_0|} = \frac{4}{3+4} \leq |x| \leq \frac{1+4}{1} = \frac{|a_4| + A}{|a_4|}$$

$$0,5714 \leq |x| \leq 5 \Rightarrow \left[\begin{array}{l} -5 \leq x \leq -0,5714, \\ 0,5714 \leq x \leq 5; \end{array}\right.$$

Теорема про верхню межу

$x^4 - 3x^3 + 2x - 4 = 0 \Rightarrow a_n = 1 > 0,\ m = 3,\ n = 4,\ B_1 = 4$

а) верхня межа дод. коренів: $R = 1 + \sqrt[4-3]{\frac{4}{1}} = 5$

б) нижня межа дод. коренів: $x = \frac{1}{y}$

$-4y^4 + 2y^3 - 3y + 1 = 0 \quad | \cdot (-1)$ $\quad a_n = 4,\ B = +2,\ m = 3$

$4y^4 - 2y^3 + 3y - 1 = 0$ $\quad y^+ = \frac{1}{x^+} \leq 1 + \sqrt[4-3]{\frac{2}{4}} = 1,5$

$x^+ \geq \frac{1}{1,5} = 0,(6) = R_1$

в) нижня межа від'ємних коренів: $x = -x$

$x^4 + 3x^3 - 2x - 4 = 0;\ B = 4,\ a_n = 1,\ m = 1$

$-x^- \leq 1 + \sqrt[4-1]{\frac{4}{1}} = 2,5874;\ x^- \geq -2,5874 = R_2$

г) верхня межа від. коренів: $x = -\frac{1}{y}$

$4y^4 + 2y^3 - 3y - 1 = 0;\ B = 3,\ m = 1,\ a_n = 4$

$y^+ = -\frac{1}{x^-} \leq 1 + \sqrt[4-1]{\frac{3}{4}} = 1,9086;$

$x^- \leq -0,5239 = \frac{1}{R_3}$ $\quad$ Рез.: $\left[\begin{array}{l} 0,6667 \leq x^+ \leq 5 \\ -2,5874 \leq x^- \leq -0,5239 \end{array}\right.$

Теорема Штурма

$$f_0 = x^4 - 3x^3 + 2x - 4;$$

$$f_1 = 4x^3 - 9x^2 + 2;$$

$$f_2 = \frac{27}{16}x^2 - \frac{3}{2}x + \frac{29}{8};$$

$$f_3 = \frac{1088}{81}x - \frac{3328}{243};$$

$$f_4 = const < 0.$$

	-2	-1	2	3
f_0	+	−	−	+
f_1	−	−	−	+
f_2	+	+	+	+
f_3	−	−	+	+
f_4	−	−	−	−

КЗЗ 3 2 2 1

$\underbrace{3 \quad 2}_{3-2 = 1 \text{ корінь}}$ $\underbrace{2 \quad 1}_{2-1 = 1 \text{ корінь}}$

Завдання 2:

Програмний етап полягає в тому, щоб уточнити корені рівняння методом бісекції, методом хорд, методом дотичних;

```python
"""
Solving polynoms

"""

# ----------- Input -----------


def f(x):
    return -x**4 + 3 * x**3 - 2 * x + 4
```

```
def df(x):
    return -4 * x**3 + 9 * x**2 + 2


INTERVALS = [(-2, 0), (2, 4)]
ACCURACY = 5


# ------------ Code ------------


def bisect(function, derivative, number, a, b, acc, counter=1, debug=True) -> float:
    e = 10**-acc
    mid = (a + b) / 2
    if debug:
        print(f"Iteration: {counter}, Root #{number}: {round(mid, acc)}, Abs. len.:
{round(abs(a - b), acc)}")
    while abs(a - b) >= e:
        if function(a) * function(mid) <= 0:
            b = mid
        elif function(mid) * function(b) <= 0:
            a = mid
        mid = (a + b) / 2
        counter += 1
        if debug:
            print(f"Iteration: {counter}, Root #{number}: {round(mid, acc)}, Abs. len.:
{round(abs(a - b), acc)}")
    return mid


def secant(function, derivative, number, a, b, acc, counter=1, debug=True) -> float:
    e = 10 ** -acc
    mid = (a + b) / 2
    x, x_prev = mid, mid + 2 * e
    if debug:
        print(f"Iteration: {counter}, Root #{number}: {round(x, acc)}, Abs. change.:
{round(abs(x - x_prev), acc)}")
    while abs(x - x_prev) >= e:
        x, x_prev = x - function(x) / (function(x) - function(x_prev)) * (x - x_prev), x
        counter += 1
        if debug:
            print(f"Iteration: {counter}, Root #{number}: {round(x, acc)}, Abs. change.:
{round(abs(x - x_prev), acc)}")
    return x


def newton(function, derivative, number, a, b, acc, counter=1, debug=True) -> float:
    e = 10 ** -acc
    mid = (a + b) / 2
    x, x_prev = mid, mid + 2 * e
    if debug:
        print(f"Iteration: {counter}, Root #{number}: {round(x, acc)}, Abs. change.:
{round(abs(x - x_prev), acc)}")
    while abs(x - x_prev) >= e:
```

```python
        x, x_prev = x - function(x) / derivative(x), x
        counter += 1
        if debug:
            print(f"Iteration: {counter}, Root #{number}: {round(x, acc)}, Abs. change.: 
{round(abs(x - x_prev), acc)}")
    return x


def find_roots(function, derivative, intervals, method, accuracy=5, debug=True) -> list:
    roots = []
    for i, (a, b) in enumerate(intervals):
        root = round(method(function, derivative,
                            i + 1, a, b, accuracy), accuracy)
        roots.append(root)
        if debug:
            print()
    return roots


def main():
    print("\n------------ Bisect method ------------")
    roots = find_roots(f, df, INTERVALS, bisect, ACCURACY, True)
    print("Roots:", roots)

    print("\n------------ Secant method ------------")
    roots = find_roots(f, df, INTERVALS, secant, ACCURACY, True)
    print("Roots:", roots)

    print("\n------------ Newton method ------------")
    roots = find_roots(f, df, INTERVALS, newton, ACCURACY, True)
    print("Roots:", roots)


if __name__ == "__main__":
    main()
```

```
------------ Bisect method ------------
Iteration: 1, Root #1: -1.0, Abs. len.: 2
Iteration: 2, Root #1: -1.5, Abs. len.: 1.0
Iteration: 3, Root #1: -1.25, Abs. len.: 0.5
Iteration: 4, Root #1: -1.125, Abs. len.: 0.25
Iteration: 5, Root #1: -1.1875, Abs. len.: 0.125
Iteration: 6, Root #1: -1.15625, Abs. len.: 0.0625
Iteration: 7, Root #1: -1.14062, Abs. len.: 0.03125
Iteration: 8, Root #1: -1.14844, Abs. len.: 0.01562
Iteration: 9, Root #1: -1.15234, Abs. len.: 0.00781
Iteration: 10, Root #1: -1.15039, Abs. len.: 0.00391
Iteration: 11, Root #1: -1.14941, Abs. len.: 0.00195
Iteration: 12, Root #1: -1.14893, Abs. len.: 0.00098
Iteration: 13, Root #1: -1.14917, Abs. len.: 0.00049
Iteration: 14, Root #1: -1.14929, Abs. len.: 0.00024
Iteration: 15, Root #1: -1.14923, Abs. len.: 0.00012
Iteration: 16, Root #1: -1.14926, Abs. len.: 6e-05
Iteration: 17, Root #1: -1.14928, Abs. len.: 3e-05
Iteration: 18, Root #1: -1.14927, Abs. len.: 2e-05
Iteration: 19, Root #1: -1.14927, Abs. len.: 1e-05

Iteration: 1, Root #2: 3.0, Abs. len.: 2
Iteration: 2, Root #2: 2.5, Abs. len.: 1.0
Iteration: 3, Root #2: 2.75, Abs. len.: 0.5
Iteration: 4, Root #2: 2.875, Abs. len.: 0.25
Iteration: 5, Root #2: 2.9375, Abs. len.: 0.125
Iteration: 6, Root #2: 2.90625, Abs. len.: 0.0625
Iteration: 7, Root #2: 2.92188, Abs. len.: 0.03125
Iteration: 8, Root #2: 2.92969, Abs. len.: 0.01562
Iteration: 9, Root #2: 2.92578, Abs. len.: 0.00781
Iteration: 10, Root #2: 2.92773, Abs. len.: 0.00391
Iteration: 11, Root #2: 2.92676, Abs. len.: 0.00195
Iteration: 12, Root #2: 2.92627, Abs. len.: 0.00098
Iteration: 13, Root #2: 2.92603, Abs. len.: 0.00049
Iteration: 14, Root #2: 2.92615, Abs. len.: 0.00024
Iteration: 15, Root #2: 2.92609, Abs. len.: 0.00012
Iteration: 16, Root #2: 2.92606, Abs. len.: 6e-05
Iteration: 17, Root #2: 2.92607, Abs. len.: 3e-05
Iteration: 18, Root #2: 2.92606, Abs. len.: 2e-05
Iteration: 19, Root #2: 2.92607, Abs. len.: 1e-05

Roots: [-1.14927, 2.92607]
```

```
------------ Secant method ------------
Iteration: 1, Root #1: -1.0, Abs. change.: 2e-05
Iteration: 2, Root #1: -1.18182, Abs. change.: 0.18182
Iteration: 3, Root #1: -1.14322, Abs. change.: 0.03861
Iteration: 4, Root #1: -1.14905, Abs. change.: 0.00583
Iteration: 5, Root #1: -1.14927, Abs. change.: 0.00022
Iteration: 6, Root #1: -1.14927, Abs. change.: 0.0

Iteration: 1, Root #2: 3.0, Abs. change.: 2e-05
Iteration: 2, Root #2: 2.93104, Abs. change.: 0.06896
Iteration: 3, Root #2: 2.92642, Abs. change.: 0.00462
Iteration: 4, Root #2: 2.92607, Abs. change.: 0.00035
Iteration: 5, Root #2: 2.92607, Abs. change.: 0.0

Roots: [-1.14927, 2.92607]
```

```
------------ Newton method ------------
Iteration: 1, Root #1: -1.0, Abs. change.: 2e-05
Iteration: 2, Root #1: -1.13333, Abs. change.: 0.13333
Iteration: 3, Root #1: -1.14622, Abs. change.: 0.01289
Iteration: 4, Root #1: -1.14867, Abs. change.: 0.00245
Iteration: 5, Root #1: -1.14915, Abs. change.: 0.00049
Iteration: 6, Root #1: -1.14925, Abs. change.: 0.0001
Iteration: 7, Root #1: -1.14927, Abs. change.: 2e-05
Iteration: 8, Root #1: -1.14927, Abs. change.: 0.0

Iteration: 1, Root #2: 3.0, Abs. change.: 2e-05
Iteration: 2, Root #2: 2.92, Abs. change.: 0.08
Iteration: 3, Root #2: 2.92728, Abs. change.: 0.00728
Iteration: 4, Root #2: 2.92584, Abs. change.: 0.00143
Iteration: 5, Root #2: 2.92611, Abs. change.: 0.00027
Iteration: 6, Root #2: 2.92606, Abs. change.: 5e-05
Iteration: 7, Root #2: 2.92607, Abs. change.: 1e-05

Roots: [-1.14927, 2.92607]
```

Завдання 3:

Порівняти отримані результати, зробити висновки, який метод приводить до меншої кількості ітерацій і чим це зумовлено.

1) Теорема про верхню границю + уточнення способом Лагранжа дало більш точні проміжки для пошуку коренів у моєму варіанті, ніж теорема про кільце, а теорема Штурма з кроком 1 змогла гарно розділити корені у цих проміжках;
2) Метод хорд у моєму варіанті справився за найменшу кількість ітерацій. Думаю, що це зумовлено виглядом функції у даних проміжках. На майже вертикальній прямій метод хорд трохи швидше ніж метод Ньютона і значно швидше ніж метод бісекцій зійшовся до кореня.