Metody Numeryczne - Zadanie XVIII

Mateusz Kamiński

December 2025

1 Wstęp

W zadaniu należało znaleźć minimum wielomianu interpolacyjnego z zadania 7.

2 Opis

2.1 Równanie

Z rozwiązanego zadania 7 wiemy, że wielomian interpolacyjny jest dany wzorem

$$f(x) = x^7 - x^6 + 3x^3 - 2x^2 + 1$$

2.2 Metoda Brenta

Dla trzech punktów a, b, c wykonywana jest interpolacja paraboliczna, a punkt minimum paraboli wyznaczany jest ze wzoru:

$$d = \frac{1}{2} \cdot \frac{a^2(f_c - f_b) + b^2(f_a - f_c) + c^2(f_b - f_a)}{a(f_c - f_b) + b(f_a - f_c) + c(f_b - f_a)}.$$

Krok interpolacyjny jest akceptowany tylko wtedy, gdy punkt d leży wewnątrz przedziału (a, c) oraz prowadzi do istotnego skrócenia przedziału. W przeciwnym przypadku wykonywany jest krok bisekcji:

$$d = \frac{a + c}{2}.$$

Algorytm kończy działanie po spełnieniu warunku:

$$|c - a| < \varepsilon(|b| + |d|).$$

3 Implementacja

```
import numpy as np
from numpy.typing import NDArray

num = np.float64
vector = NDArray[np.float64]

def horner_value(x: num) -> num:
    # współczynniki z zadania 7
    coeffs = np.array([1.0, -1.0, 0.0, 0.0, 3.0, -2.0, 0.0, 1.0],
dtype=num)

    p = coeffs[0]
    for i in range(1, len(coeffs)):
        p = p * x + coeffs[i]
    return p

def f(x: num) -> num:
    return horner_value(x)


def bracket_minimum(x0, h=0.1, max_iter=50):
    a = x0
    fa = f(a)

    b = a + h
    fb = f(b)

    # zmiana kierunku, jeśli idziemy pod górę
    if fb > fa:
        h = -h
        b = a + h
        fb = f(b)
        if fb > fa:
            raise RuntimeError("Brak kierunku spadku")

    for _ in range(max_iter):
        c = b + h
        fc = f(c)

        if fc > fb:
            return a, b, c

        a, fa = b, fb
        b, fb = c, fc
        h *= 2

    raise RuntimeError("Nie znaleziono przedziału – funkcja monotoniczna")

def brent(a: num, b: num, c: num, tolerance: float = 1e-6) -> tuple[num,
```

```
int]:
    prev_interval = abs(c - a)
    prev2_interval = prev_interval

    i = 0
    while True:
        i += 1

        fa, fb, fc = f(a), f(b), f(c)
        old_b = b

        # interpolacja paraboliczna
        denominator = a * (fc - fb) + b * (fa - fc) + c * (fb - fa)

        accept = False
        if abs(denominator) > 1e-14:
            numerator = a ** 2 * (fc - fb) + b ** 2 * (fa - fc) + c ** 2 *
(fb - fa)
            d = numerator / (2*denominator)

            if a < d < c:
                new_interval = max(abs(d - a), (c - d))
                if new_interval <= 0.5 * prev2_interval:
                    accept = True

        if not accept:
            d = (c + a) / 2.0

        fd = f(d)

        if fd < fb:
            if d < b:
                c, b = b, d
            else:
                a, b = b, d

        else:
            if d < b:
                a = d
            else:
                c = d

        prev2_interval = prev_interval
        prev_interval = abs(c - a)

        if abs(c - a) < tolerance * (abs(old_b) + abs(d)):
            break

    return b, i

def main():
```

```
    a, b, c = bracket_minimum(0.0)
    x_min, iterations = brent(a, b, c)
    print(f"Minimum znaleziono w punkcie x = {x_min} w {iterations}
iteracjach.")

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

4 Wyniki

4.1 Rozwiązania

Dla punktów: a = 0.2, b = 0.4, c = 0.8 znaleziono minimum funkcji

$$x_{\min} = 0.4581139893888733, \quad \text{iteracje} = 15$$