Zadanie Numeryczne 17

Mateusz Wojtyna

1. Wstęp

Należało znaleźć minimum funkcji jednej zmiennej danej wzorem:

$$f(x) = \frac{1}{4}x^4 - \frac{1}{2}x^2 - \frac{1}{16}x$$

metodą złotego podziału oraz metodą Brenta z dokładnością do 10^{-6}. Należało również porównać zbieżności obu metod.

2. Opis

Na początku ograniczamy minimum funkcji trzema punktami $a < b < c$ tak, że $f(a) > f(b)$ oraz $f(b) < f(c)$. Następnie wybieramy punkt d na podstawie jakiejś metody i odpowiednio przesuwamy punkty a, b, c. Ustalamy znowu punkt d biorąc pod uwagę nowe a, b, c, i tak dalej. Iterację zatrzymujemy jeżeli $|c - a| < \varepsilon(|b| + |d|)$.

2.1. Metoda złotego podziału

W metodzie złotego podziału punktu d szukamy w większym z dwóch podprzedziałów $[a, b]$ i $[b, c]$. Załóżmy, że $[a, b]$ jest większy, wtedy możemy dążyć do takiego podziału, że

$$\frac{|b - a|}{|c - a|} = \frac{|c - b|}{|b - a|}$$

korzystając z warunków:

$$\begin{aligned} &\text{jeżeli } |b - a| > |c - b| \text{ to } d = a + w \cdot |b - a| \\ &\text{jeżeli } |b - a| < |c - b| \text{ to } d = b + w \cdot |c - b| \end{aligned}$$

gdzie $w = \frac{3-\sqrt{5}}{2}$. Ta metoda jest zbieżna liniowo.

2.2. Metoda Brenta

Metoda Brenta wykorzystuje interpolację paraboliczną, aby znaleźć minimum. Przez punkty $(a, f(a)), (b, f(b)), (c, f(c))$ przeprowadzamy parabolę, a jako punkt d bierzemy jej minimum:

$$d = \frac{1}{2} \cdot \frac{a^2(f(c) - f(b)) + b^2(f(a) - f(c)) + c^2(f(b) - f(a))}{a(f(c) - f(b)) + b(f(a) - f(c)) + c(f(b) - f(a))}$$

Jeżeli nie zachodzi $a < d < c$ to ustawiamy $d = \frac{c+a}{2}$. Metoda ta jest zbieżna szybciej niż liniowo, ale wolnej niż kwadratowo.

3. Kod

Program napisano w Pythonie z użyciem pakietu *NumPy*.

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

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


def f(x: float):
    return 0.25 * x**4 - 0.5 * x**2 - 0.0625 * x


def initial_interval(x_0: num, delta=0.1, limit=100):
    a = x_0
    fa = f(a)
    b = a + delta
    fb = f(b)

    if fb > fa:
        # zmiana kierunku
        delta *= -1
        b = a + delta
        fb = f(b)
        assert fa > fb

    for _ in range(limit):
        c = b + delta
        fc = f(c)

        if fc > fb:
            return a, b, c
```

```
        a = b
        fa = fb
        b = c
        fb = fc
        delta *= 2

    raise RuntimeError("Nie znaleziono przedziału")


def golden_ratio(a: num, b: num, c: num, limit=100, eps=1e-6):
    w = (3 - np.sqrt(5)) / 2

    for i in range(limit):
        if abs(b - a) > abs(c - b):
            d = a + w * abs(b - a)
        else:
            d = b + w * abs(c - b)

        if abs(c - a) < eps * (abs(b) + abs(d)):
            return d, i + 1

        fd = f(d)
        fb = f(b)
        if fd < fb:
            if d < b:
                c = b
                b = d
            else:
                a = b
                b = d
        else:
            if d < b:
                a = d
            else:
                c = d

    raise RuntimeError("Nie znaleziono punktu (złoty podział)")


def brent(a: num, b: num, c: num, limit=100, eps=1e-6):
    prev = abs(c - a)
    prev2 = prev

    for i in range(limit):
        fa = f(a)
```

```
        fb = f(b)
        fc = f(c)
        old_b = b
        d = 0

        accept = False
        denominator = a * (fc - fb) + b * (fa - fc) + c * (fb - fa)
        if abs(denominator) > 1e-16:
            nominator = a * a * (fc - fb) + b * b * (fa - fc) + c * c * (fb - fa)
            d = nominator / (2 * denominator)

            if a < d < c and max(abs(d - a), (c - d)) < 0.5 * prev2:
                accept = True

        if not accept:
            d = (c + a) / 2

        if abs(c - a) < eps * (abs(old_b) + abs(d)):
            return d, i + 1

        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 = prev
        prev = abs(c - a)

    raise RuntimeError("Nie znaleziono punktu (brent)")


def main():
    a, b, c = initial_interval(0)
    print("Punkty", a, b, c)
    golden_value, golden_steps = golden_ratio(a, b, c)
    print(f"Złoty podział: {golden_value} po {golden_steps} krokach")
    brent_value, brent_steps = brent(a, b, c)
```

```
    print(f"Brent: {brent_value} po {brent_steps} krokach")

    a, b, c = initial_interval(-2)
    print("\nPunkty", a, b, c)
    golden_value, golden_steps = golden_ratio(a, b, c)
    print(f"Złoty podział: {golden_value} po {golden_steps} krokach")
    brent_value, brent_steps = brent(a, b, c)
    print(f"Brent: {brent_value} po {brent_steps} krokach")


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

4. Wyniki ($\varepsilon = 10^{-6}$)

Dla punktów $a = 0.4$, $b = 0.8$, $c = 1.6$:

$$\begin{aligned} \text{złoty podział: } x_{\min} &\approx 1.029895, \quad \text{iteracje} = 34 \\ \text{metoda Brenta: } x_{\min} &\approx 1.029896, \quad \text{iteracje} = 19 \end{aligned}$$

Dla punktów $a = -1.6$, $b = -1.2$, $c = -0.4$:

$$\begin{aligned} \text{złoty podział: } x_{\min} &\approx -0.967149, \quad \text{iteracje} = 39 \\ \text{metoda Brenta: } x_{\min} &\approx -0.967149, \quad \text{iteracje} = 16 \end{aligned}$$

Widać, że metoda Brenta zbiega się prawie dwukrotnie szybciej od metody złotego podziału.