Sztuczna inteligencja dla optymalizacji oprogramowania 2021

Kamil Celejewski 239642 239642@edu.p.lodz.pl
Łukasz Starosta 239711 239711@edu.p.lodz.pl

Zadanie 1.: Optymalizacja jednowymiarowa

1. Cel

Celem zadania było zaimplementowanie oraz porównanie dwóch metod optymalizacji jednowymiarowej. Zaimplementowane i porównane zostały metoda bisekcji oraz Fibonacci. Implementacja została wykonana przy użyciu języka Python w wersji 3.9.2.

2. Wprowadzenie

2.1. Sprawdzenie unimodalności

Unimodalność [4] przedziału wprowadzonego przez użytkownika została sprawdzona za pomocą algorytmu, który przeszukuje dziedzinę funkcji oraz oblicza jej wartości w zadanych punktach. Funkcja jest podejrzewana o bycie unimodalną, jeśli spełniony jest warunek (1). Wzór (2) służy do wyliczenia przedziału dla danej funkcji.

$$f(a) > f(x) \ \ and \ \ f(b) > f(x) \tag{1}$$

- x – punkt początkowy
- f – rozważana funkcja
- a – początek przedziału
- b – koniec przedziału

$$f(x-d) > f(x) \ \ and \ \ f(x+d) > f(x) \tag{2}$$

- x – punkt początkowy
- d – krok
- f – rozważana funkcja

2.2. Metoda bisekcji

Metoda bisekcji [1] pozwala znaleźć minimum lokalne poprzez eliminację połowy rozważanego przedziału w każdej iteracji. Przy pierwszej iteracji należy wyliczyć cztery wartości, zgodnie ze wzorami (3), (4), (5) i (6). Gdzie x_m jest środkiem przedziału $[a, b]$ a x_1 i x_2 są odpowiedni środkami przedziałów $[a, x_m]$ i $[x_m, b]$.

$$x_1 = a + L/4 \tag{3}$$

$$x_m = (a + b)/2 \tag{4}$$

$$x_2 = a + L/4 \tag{5}$$

$$L = b - a \tag{6}$$

- a – początek przedziału
- b – koniec przediału

Następnie należy wybrać nowy przedział an podstawie warunków. Jeśli warunek (7) został spełniony, poszukiwania minimum należy kontynuować w przedziale $[a, x_m]$. Jeśli warunek (8) jest prawdziwy, jako nowy przedział wybierany jest przedział $[x_m, b]$. Jeśli warunek (7) i (8) nie są spełnione, jako nowy przedział należy wybrać $[x_1, x_2]$.

$$f(x_1) < f(x_m) \tag{7}$$

$$f(x_2) < f(x_m) \tag{8}$$

- x_1 – wartość wyliczona ze wzoru (3)
- x_2 – koniec przediału ze wzoru (5)
- x_m – koniec przediału ze wzoru (4)

Obliczenia kontynuowane są dopóki różnica między początkiem przedziału a jego końcem nie będzie mniejsza bądź równa od 2ϵ.

$$|b - a| \leqslant 2\varepsilon \tag{9}$$

Jako wartość znalezionego przedziału przyjmuje się wartość x_m wyliczoną ze wzoru (4), gdzie wartości a i b są odpowiedni początkiem i końcem ostatniego znalezionego przedziału.

2.3. Metoda Fibonacci

W metodzie Fibonacciego [2] [3] metoda zmniejszania przedziału następuje w inny sposób niż w metodzie bisekcji. Do obliczenia proporcji wykorzystywany jest ciąg Fibonacciego, zdefiniowany następująco (10):

$$F_0 = F_1 = 1, F_k = F_{k-1} + F_{k-2} \ \ dla \ \ k = 2,3,... \tag{10}$$

Dla funkcji unimodalnej $f(x)$ w przedziale $[a,b]$ przyjmujemy dwa warunki stopu: na ilość iteracji, oraz na dokładność ϵ, czyli szerokość przedziału poszukiwań. Pierwszym krokiem jest znalezienie n takiego, że:

$$\frac{b-a}{F_n} \leqslant \epsilon \tag{11}$$

W tym celu należy korzystać z kolejnych liczb w ciągu Fibonacciego. Następnie wyznaczamy punkty c i d:

$$c = a + \frac{F_{n-2}}{F_n}(b-a) \tag{12}$$

$$d = a + \frac{F_{n-1}}{F_n}(b-a) \tag{13}$$

oraz obliczamy wartości funkcji $fc = f(c)$ i $fd = f(d)$.

Następnie, jeżeli $n = 2$, wtedy możemy od razu wyznaczyć minimum w punkcie $\frac{a+b}{2}$. Natomiast jeżeli $n > 2$ wtedy zmniejszamy n:

$$n = n - 1 \tag{14}$$

i porównujemy wartości fc oraz fd, w celu zmniejszenia przedziału. Jeżeli $fc < fd$, dokonujemy następujących operacji, w celu odrzucenia przedziału po prawej stronie (między d a b):

$$\begin{aligned} b &= d \\ d &= c \\ fd &= fc \\ c &= a + \frac{F_{n-2}}{F_n}(b-a) \\ fc &= f(c) \end{aligned} \tag{15}$$

w przypadku, gdy $fc \geqslant fd$, odrzucamy przedział po lewej stronie (między a a c), wykonując:

$$\begin{aligned} a &= c \\ c &= d \\ fc &= fd \\ d &= a + \frac{F_{n-1}}{F_n}(b-a) \\ fd &= f(d) \end{aligned} \tag{16}$$

Po tym kroku wracamy ponownie do porównania wartości n i kończymy algorytm lub wykonujemy znowu kroki (14), (15) oraz (16), aż do znalezienia minimum lub osiągnięcia limitu powtórzeń iteracji.

3. Opis implementacji

Aplikacja została napisana w języku Python, w wersji 3.9.2. Interfejs graficzny został stworzony przy użyciu *tkinter*. Główny plik *zad1.py* zawiera definicje elementów GUI oraz główną metodę służącą do narysowania wykresu podanej funkcji, sprawdzenia unimodalności przedziału i zastosowania wybranego przez użytkownika sposobu znajdowania minimum.

Do ewaluacji podanej przez użytkownika funkcji używana jest wbudowana metoda *eval*. Aby umożliwić operacje matematyczne takie jak *sin* czy *cos*, niezbędne jest zaimportowanie biblioteki *math* oraz przekazanie jej metod jako drugi parametr funkcji *eval*:

```
from inspect import getmembers
import math


def eval_math_fn(function, name_dict):
    math_name_dict = dict(getmembers(math))

    # Uncomment to use custom function
    # x = name_dict['x']
    # return (x ** 2) - x

    return eval(function, {**name_dict, **math_name_dict})
```

Przykładowe wywołanie podanej przez użytkownika funkcji w punkcie $x = 5$ odbywa się poprzez użycie:

```
eval_math_fn(fn, {'x': 5})
```

gdzie fn jest funkcją zapisaną jako *string*.

Odpowiednie algorytmy zostały napisane w osobnych plikach w folderze *variants*, natomiast metody wspólne dla obu sposobów, takie jak zaznaczanie przedziałów, zawarte są w folderze *utils*.

Repozytorium github: `https://github.com/kcc112/SIDOO-zad1`

4. Materiały i metody

W celu zbadania poprawności metod bisekcji oraz Fibonacci zostały wykonane eksperymenty na dwóch funkcjach o wzorach (17) oraz (18):

$$f(x) = x^2 \tag{17}$$

$$f(x) = \frac{x^3}{3} - \frac{x^2}{2} - x - 1 \tag{18}$$

W trakcie eksperymentów modyfikowane były kryterium dokładności oraz ilość iteracji. Wyniki dla obu metod zostały na koniec porównane z wynikami zwracanymi przez *scipy.optimize.minimize_scalar* [5] dla tych samych parametrów.

4.1. Eksperyment pierwszy

Eksperyment miał na celu porównanie wpływu ilości iteracji na dokładność znalezionego minimum. Dla stałej wartości ϵ oraz stałego przedziału, zwiększana była ilość iteracji od 1, co 5 aż do wartości 30 w celu sprawdzenia wpływu tego parametru na dokładność znalezionego minimum.

Wykorzystane parametry:
- $[-5, 12]$ - badany przedział dla funkcji (17)
- $[1, 2]$ - badany przedział dla funkcji (18)
- 0.1 - wykorzystany ϵ
- 1 - początkowa ilość iteracji
- 30 - końcowa ilość iteracji

4.2. Eksperyment drugi

Eksperyment miał na celu porównanie wpływu współczynnika dokładności ϵ na wartość znalezionego minimum. Dla stałej ilości iteracji oraz niezmiennego przedziału modyfikowana była wartość ϵ w celu sprawdzenia wpływu tego na dokładność znalezionego minimum.

Wykonane zostało 5 prób gdzie wartość ϵ była stopniowo zmniejszana z 0.5 do wartości 0.1.

Wykorzystane parametry:
- $[-5, 12]$ - badany przedział dla funkcji (17)
- $[1, 2]$ - badany przedział dla funkcji (18)
- 0.5 - początkowa wartość ϵ
- 0.1 - końcowa wartość ϵ
- 30 - ilość iteracji

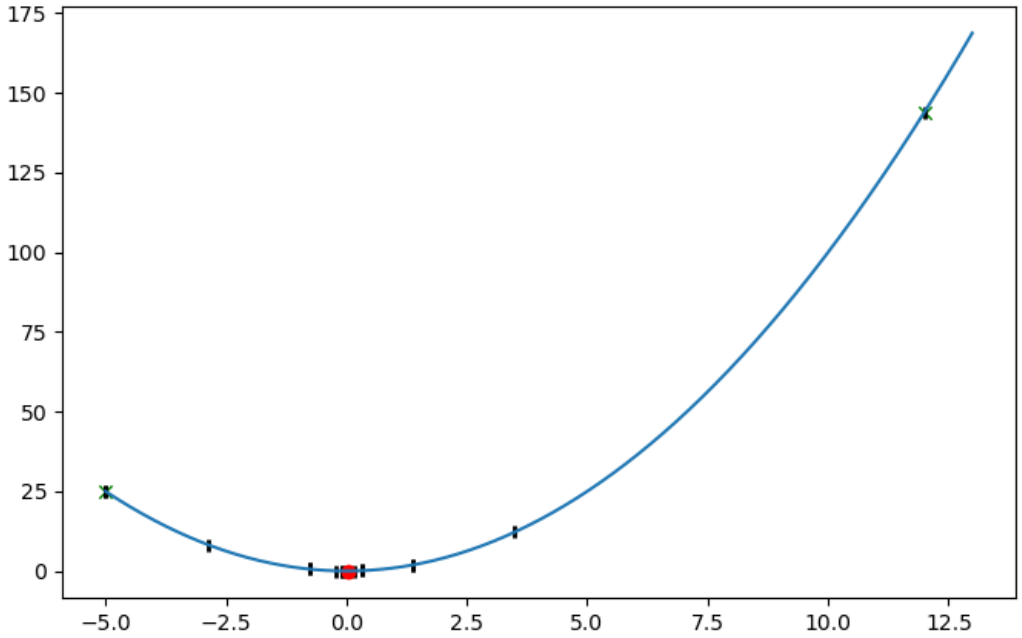
5. Wyniki

5.1. Eksperyment pierwszy

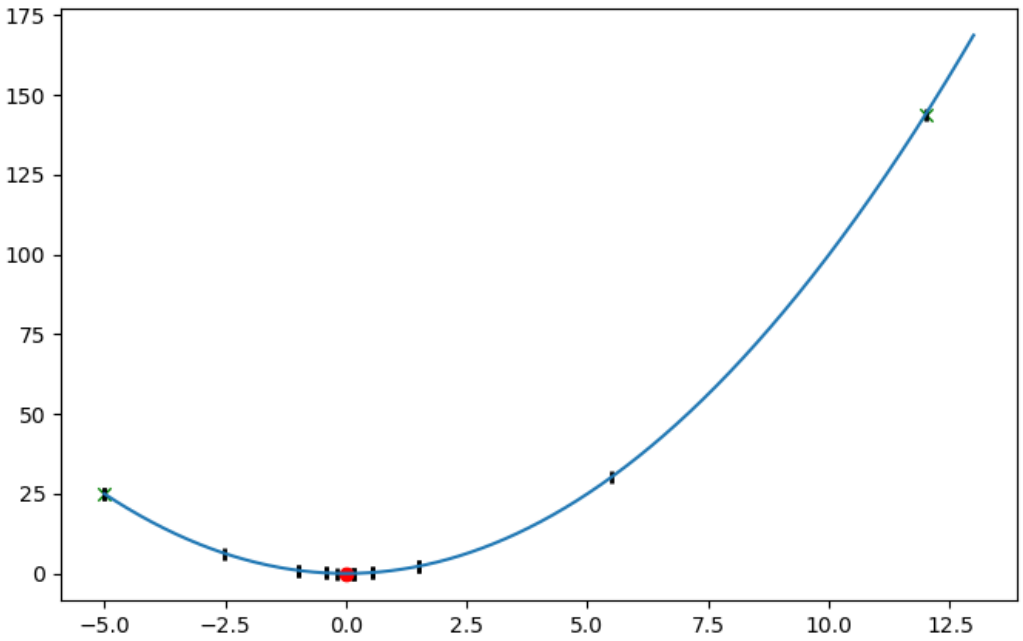
Rozważając zmianę iteracji dla funkcji (17) i obu metod otrzymaliśmy następujące wyniki, zaprezentowane w Tabeli 1. Dla funkcji (18) otrzymaliśmy wyniki zaprezentowane w Tabeli 2.

Ilość iteracji	Bisekcja	Fibonacci	*minimize_scalar*
1	3.50	0.25	1.493
5	-0.22	-0.22	0
10	-0.02	0.03	0
15	-0.02	-0.002	0

Tabela 1. Porównanie wyników pierwszego eksperymentu dla funkcji (17)



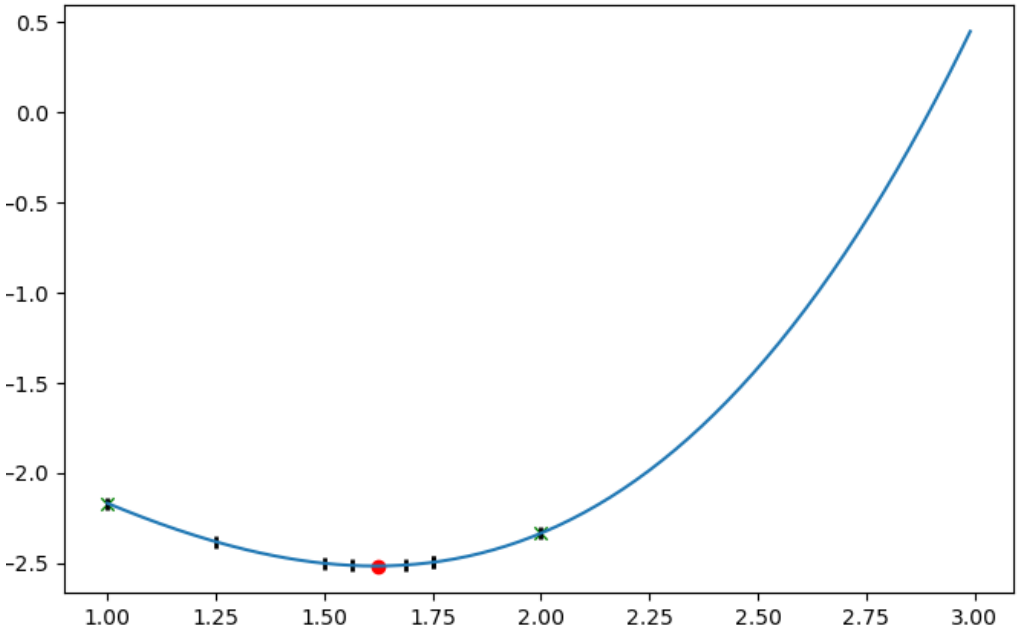
Rysunek 1. Wynik pierwszego eksperymentu dla funkcji (17) i metody bisekcji



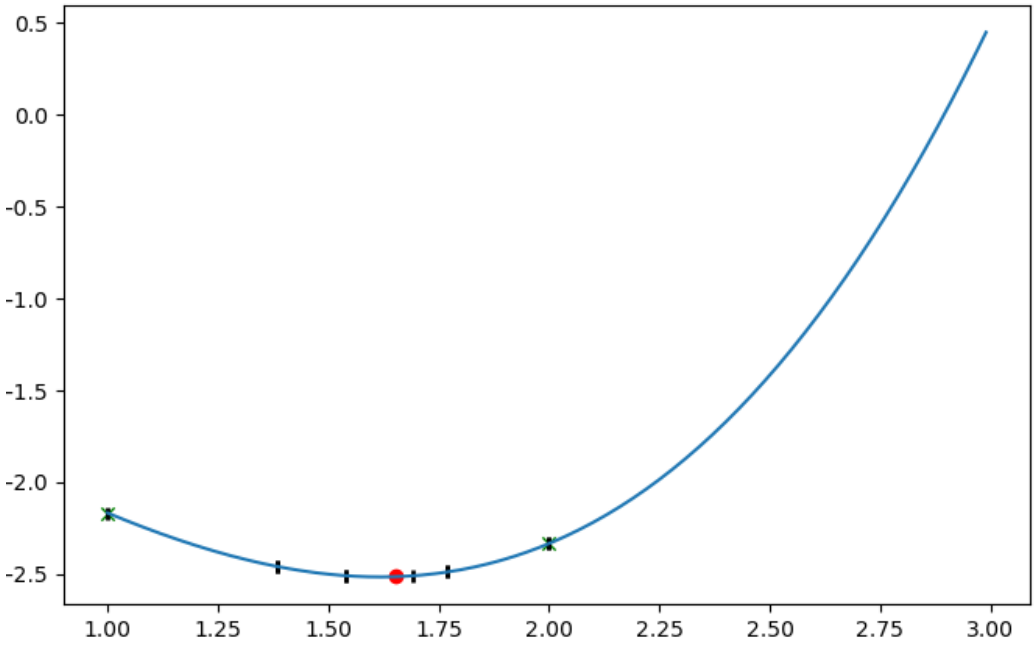
Rysunek 2. Wynik pierwszego eksperymentu dla funkcji (17) i metody Fibonacci

Ilość iteracji	Bisekcja	Fibonacci	*minimize_scalar*
1	1.5	1.69	1.618
5	1.63	1.65	1.618
10	1.63	1.654	1.618

Tabela 2. Porównanie wyników pierwszego eksperymentu dla funkcji (18)



Rysunek 3. Wynik pierwszego eksperymentu dla funkcji (18) i metody bisekcji



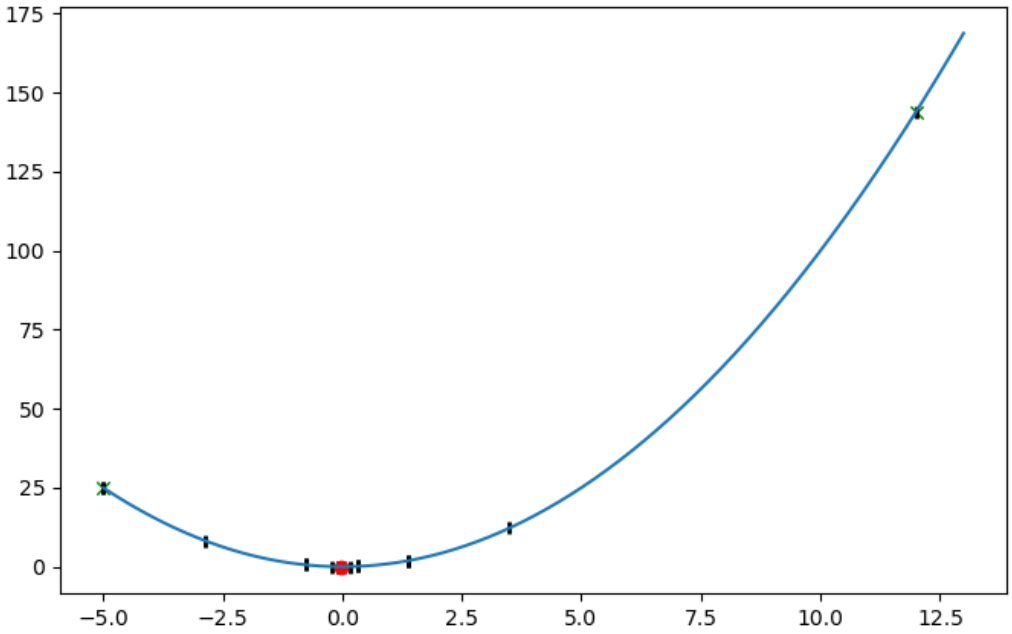
Rysunek 4. Wynik pierwszego eksperymentu dla funkcji (18) i metody Fibonacci

5.2. Eksperyment drugi

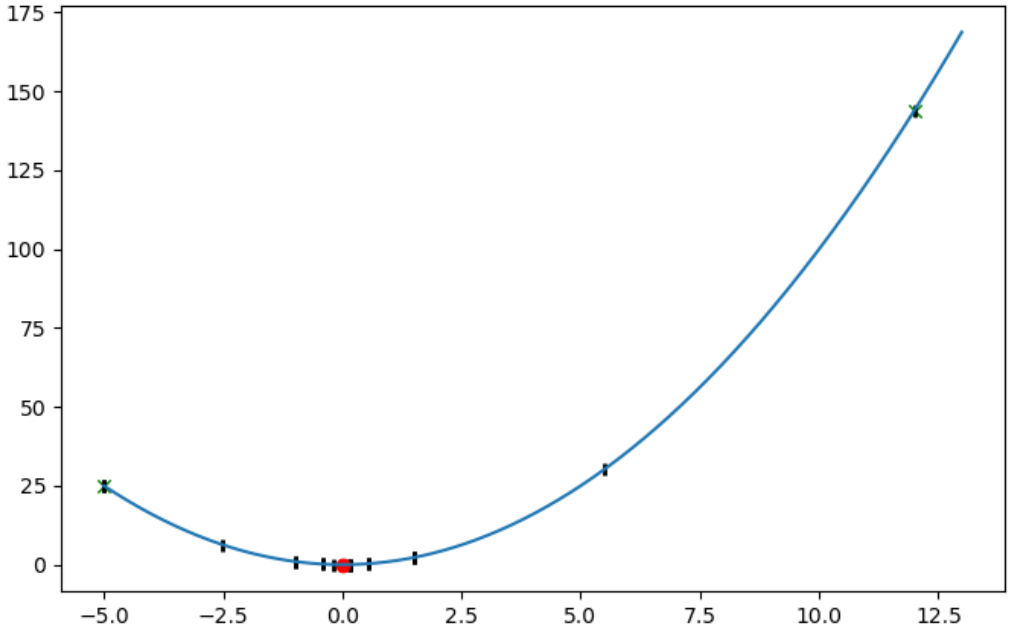
Rozważając zmianę iteracji dla funkcji (17) i obu metod otrzymaliśmy następujące wyniki, zaprezentowane w Tabeli 3. Dla funkcji (18) otrzymaliśmy wyniki zaprezentowane w Tabeli 4.

Dokładność	Bisekcja	Fibonacci	*minimize_scalar*
0.5	0.047	0.1	0
0.4	0.047	0.1	0
0.3	0.047	0.06	0
0.2	0.047	0.06	0
0.1	-0.019	-0.002	0

Tabela 3. Porównanie wyników drugiego eksperymentu dla funkcji (17)



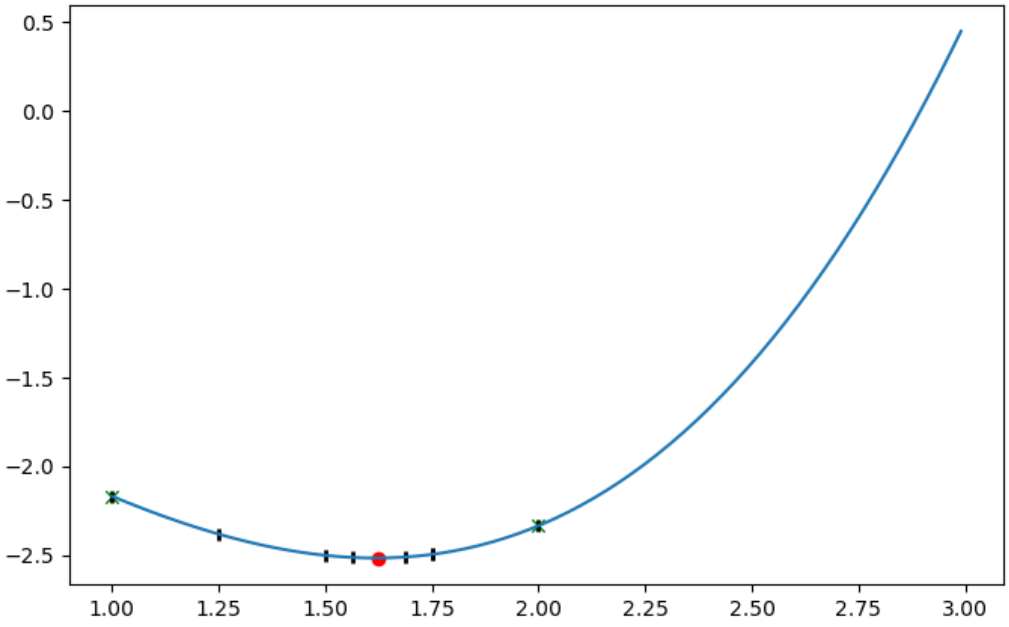
Rysunek 5. Wynik drugiego eksperymentu dla funkcji (18) i metody bisekcji



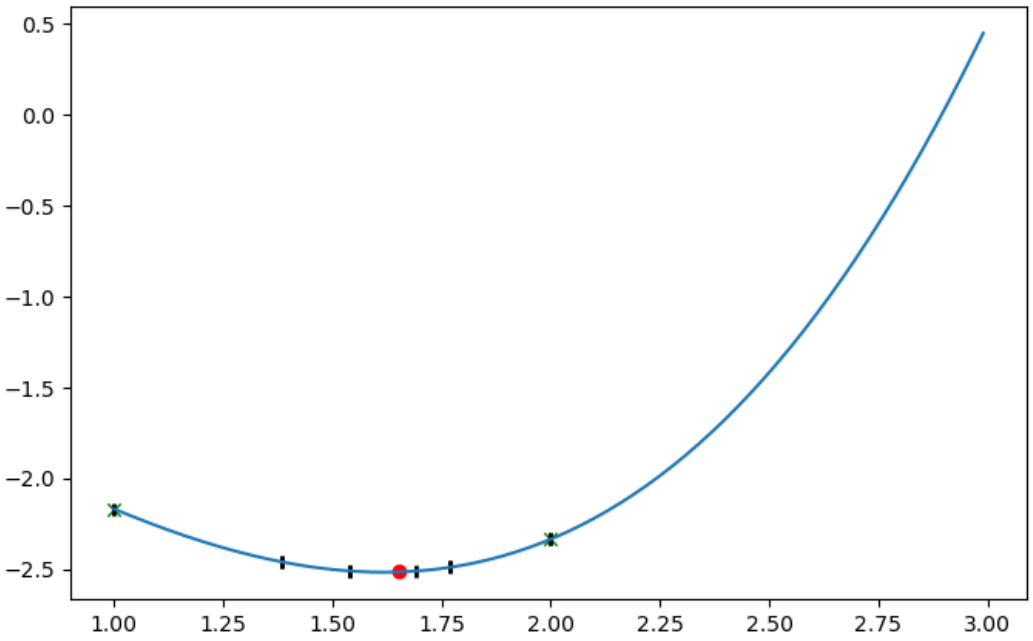
Rysunek 6. Wynik drugiego eksperymentu dla funkcji (18) i metody Fibonacci

Dokładność	Bisekcja	Fibonacci	*minimize_scalar*
0.5	1.5	1.83	1.618
0.4	1.5	1.83	1.618
0.3	1.5	1.7	1.618
0.2	1.625	1.686	1.618
0.1	1.625	1.654	1.618

Tabela 4. Porównanie wyników drugiego eksperymentu dla funkcji (18)



Rysunek 7. Wynik drugiego eksperymentu dla funkcji (18) i metody bisekcji



Rysunek 8. Wynik drugiego eksperymentu dla funkcji (18) i metody Fibonacci

6. Dyskusja

6.1. Eksperyment pierwszy

Dla funkcji (17) metoda minimize_scalar osiągnęła najlepszy wynik wg. Tabeli 3 przy najmniejszej liczbie iteracji. Drugą z kolej metodą, która osiągnęła najlepszy wynik, była metoda Fibonacci. Natomiast wynik metody bisekcji przestał się zmieniać pomimo zwiększonej liczby iteracji.

Podobne wyniki z Tabeli 3 zostały uzyskane dla funkcji (18). Metoda minimize_scalar po pierwszej iteracji znalazł minimum, a jej wynik nie zmienił się w miarę wzrostu liczby iteracji. Metoda Fibonacci osiągnęła wynik najbardziej zbliżony do metody minimize_scalar oraz wraz ze wzrostem liczby iteracji jej wynik coraz bardziej przybliżał się do wyniku metody minimize_scalar. Natomiast wynik metody bisekcji pomimo zwiększonej ilości iteracji w pewnym momencie przestał się zmieniać.

Powyższe wyniki pokazują, że metoda minimize_scalar jest w stanie znaleźć minimum lokalne przy najmniejszej liczbie iteracji oraz że metoda Fibonacci przy tej samej liczbie iteracji co metoda bisekcji znajduje minimum, które jest bliższe rzeczywistej wartości.

6.2. Eksperyment drugi

Dla funkcji (17) metoda minimize_scalar osiągnęła najlepszy wynik wg. Tabeli 4 przy wykorzystaniu największej wartości ϵ. Drugą z kolej metodą, która osiągnęła najlepszy wynik, była metoda Fibonacci, która przy $\epsilon = 0.1$ uzyskała wynik -0.002. Natomiast wynik metody bisekcji wyniósł jedynie -0.019.

Podobne wyniki z Tabeli 4 zostały uzyskane dla funkcji (18). Metoda minimize_scalar znalazła jako pierwsza minimum lokalne przy wykorzystaniu największej wartości ϵ. Jednak w tym przypadku metoda bisekcji osiągnęła wartość bardziej zbliżoną do wartości metody minimize_scalar niż metoda Fibonacci.

Powyższe wyniki pokazują, że metoda minimize_scalar jest w stanie znaleźć minimum lokalne przy podanej największej wartości ϵ oraz że metoda Fibonacci przy tej samej wartości ϵ co metoda bisekcji niekoniecznie znajduje minimum, które jest bliższe rzeczywistej wartości niż minimum znalezione przez metodę bisekcji. W tym przypadku możemy uznać, że metoda Fibonacci potrzebuje więcej iteracji lub mniejszej wartości współczynnika dokładności.

7. Wnioski

- Metoda Fibonacci jest w stanie znaleźć minimum lokalne przy mniejszej liczbie iteracji niż metoda bisekcji
- Metoda Fibonacci niekoniecznie daje lepsze wyniki niż metoda bisekcji dla takiej samej wartości ϵ
- Najlepsze wyniki zostały uzyskane przy wykorzystaniu metody do wyszukiwania minimum lokalnego z zewnętrznej biblioteki

- Metoda Fibonacci wymaga wywołania funkcji ewaluującej jednokrotnie w każdej iteracji, natomiast metoda bisekcji - dwukrotnie, co może przekładać się na szybkość działania
- Metoda Bisekcji jest prostsza w implementacji

Literatura

[1] dr Alicji Romanowicz, *Przedziały unimodalności - fragment skryptu*, dostępny online. https://ftims.edu.p.lodz.pl/pluginfile.php/162147/mod_resource/content/1/Przedzia%C5%82y%20unimodalno%C5%9Bci.

[2] dr Alicji Romanowicz, *Minimalizacja funkcji jednej zmiennej - fragment*, dostępny online. https://ftims.edu.p.lodz.pl/pluginfile.php/162148/mod_resource/content/1/Minimalizacja%20funkcji%20jednej%20zmiennej.

[3] Oscar Veliz, *Fibonacci Search*, dostępny online. https://www.youtube.com/watch?v=GAafWFRGP7k.

[4] *Funkcja unimodalna*, dostępny online. https://pl.wikipedia.org/wiki/Funkcja_unimodalna.

[5] *Minimize scalar*, dostępny online. https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize_scalar.html?fbclid=IwAR1LAbLuGxceOBsc-evHqckJ7YK_23SMU6Tl9Exhy13Bm7D4ilWwqx89vR8.