Andrzej Pultyn

325213

WSI – zadanie 1

• Wstęp

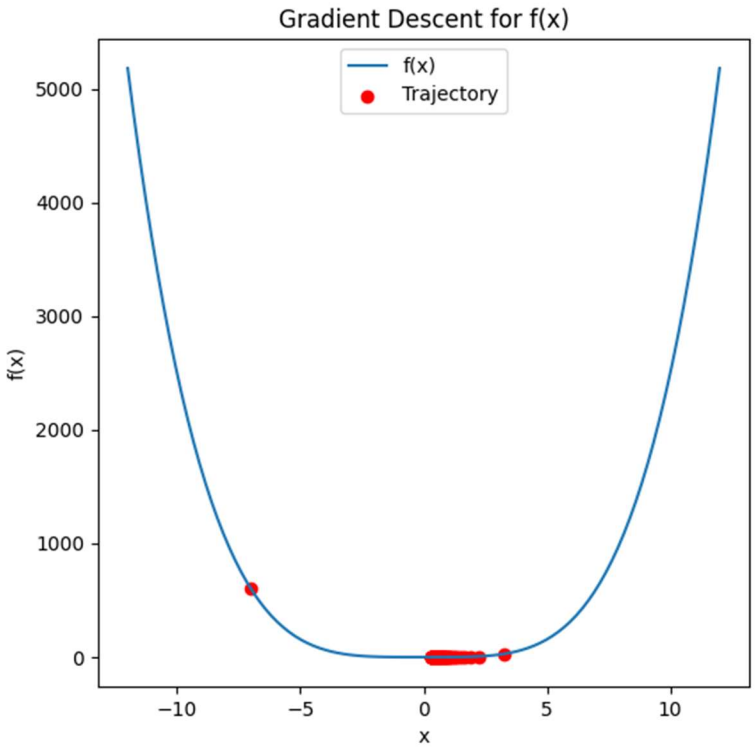
Celem zadania było zaimplementowanie algorytmu gradientu prostego, wykorzystaniu go do znalezienia minimum dwóch podanych funkcji, oraz zbadaniu wpływu rozmiaru początkowego kroku dla losowych punktów początkowych.

Algorytm realizowany jest poprzez klasę *Gradient_descent*, gdzie obiekty tworzymy, podając im funkcję, jej gradient, długość początkowego kroku, współczynnik zmniejszania tego kroku, oraz precyzję, z jaką algorytm powinien przybliżać minimum. Następnie, chcąc przybliżyć minimum funkcji, używamy metody *solve*, jako parametr podając jej punkt startowy. Funkcja zwraca tabelę zawierającą kolejne kroki algorytmu.

• Przykłady działania

Pierwszym przykładem jest optymalizacja funkcji f(x) z parametrami:

- Punkt początkowy: x = -7
- Początkowa długość kroku: 3
- Współczynnik redukcji kroku: 0.1
- Precyzja optymalizacji: wartość gradientu w punkcie < 0.001

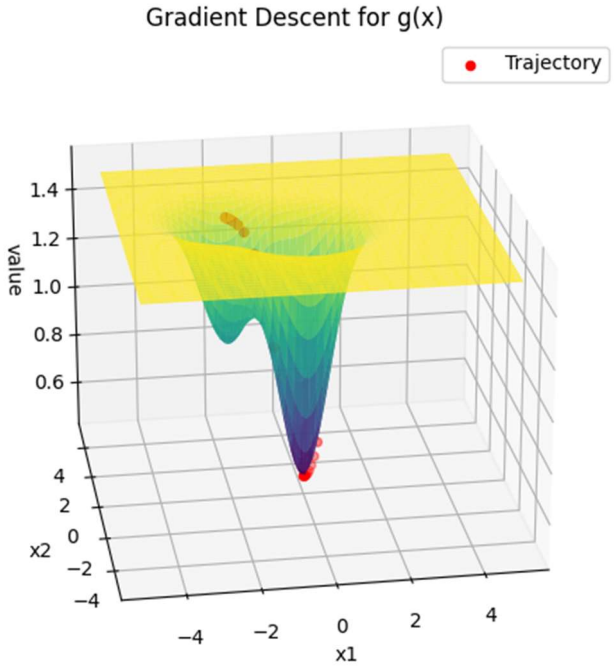


Jak widać, algorytm wykonywał coraz mniejsze kroki, aż do miejsca w którym wartość gradientu była mniejsza od zadanej precyzji, czyli w tym przypadku x ≈ 0.3214. Warte zaznaczenia jest też, że gdy znalazł się po drugiej stronie minimum niż punkt startu, to zawrócił się i ciągle dążył do owego minimum. Algorytm jest też zabezpieczony przed sytuacją, w której przy dużym rozmiarze kroku wartość w punkcie docelowym jest większa niż w poprzednim.

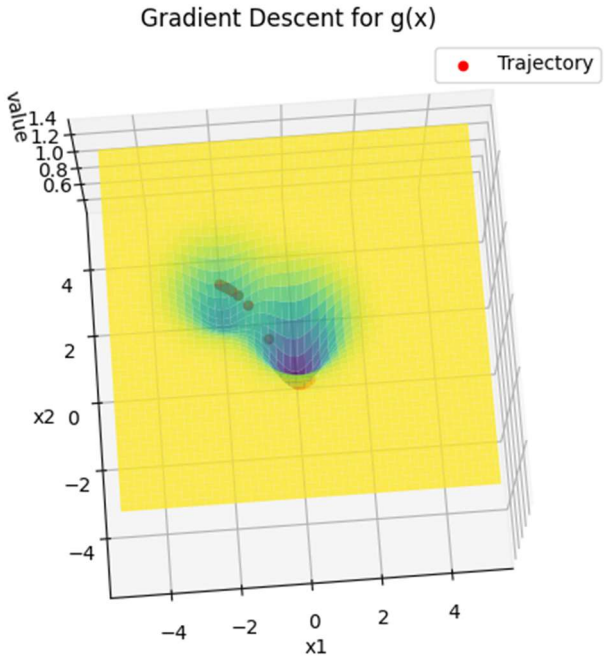
Drugim przykładem jest optymalizacja funkcji g(x) z parametrami:

- Punkt początkowy: (-2, 1)
- Początkowa długość kroku: 1,5
- Współczynnik redukcji kroku: 0.1
- Precyzja: 0.001

Widok od boku:



Widok z góry:



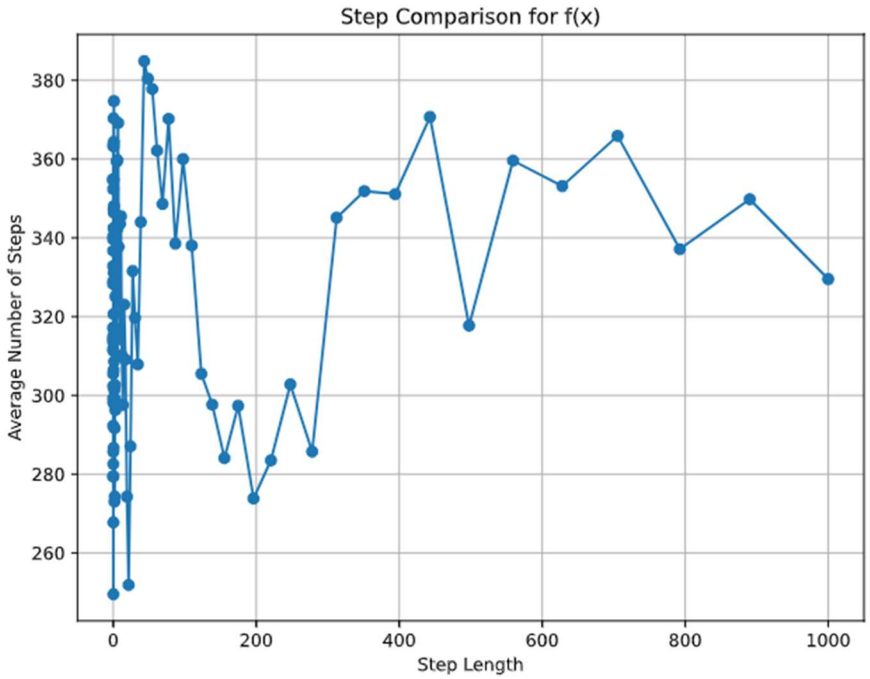
Tutaj również widać jak algorytm reaguje na „przestrzelenie” celu. Widać też, że im większe nachylenie w punkcie (wartość bezwzględna gradientu jest większa), algorytm wykonuje większe kroki. Tutaj punktem końcowym jest (0.0066, -0.009).

- # Działanie algorytmu w zależności od długości początkowego kroku

Badanie wpływu długości kroku na ilość wykonanych kroków wykonane zostało poprzez uzyskanie średniej ilości wykonanych kroków z 50 różnych punktów początkowych dla każdej długości.

Na początku wykres dla f(x), gdzie:

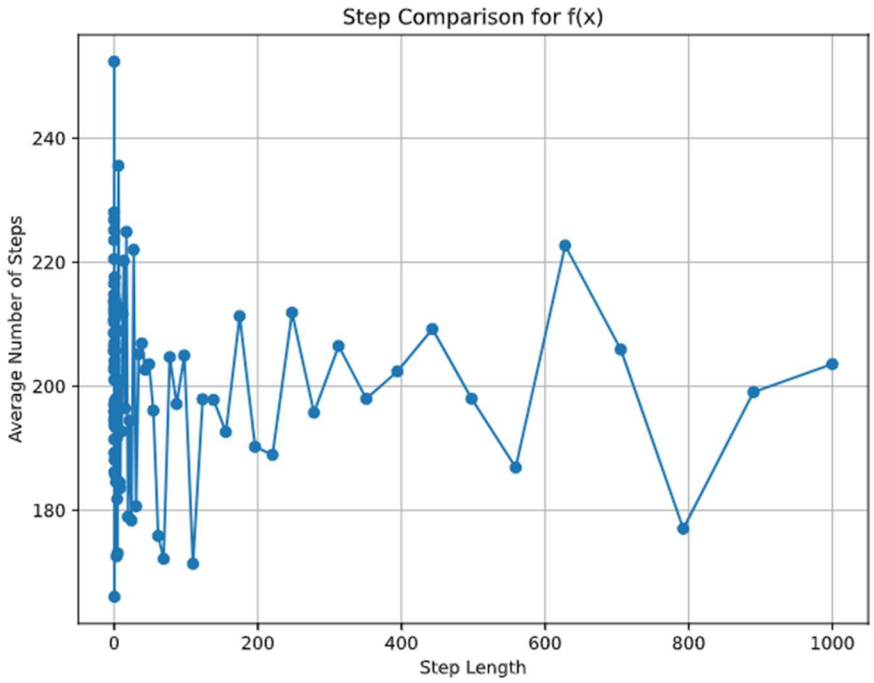
- Zakres punków początkowych: od x = -30 do x = 30
- Długości kroków: 100 różnych długości z przestrzeni logarytmicznej od 0.01 do 1000
- Współczynnik redukcji kroku: 0.1
- Precyzja: 0.001



Wyniki:

- Średnia ilość kroków: 324.557
- Najmniejsza ilość kroków: 248.9
- Największa ilość kroków: 372.2

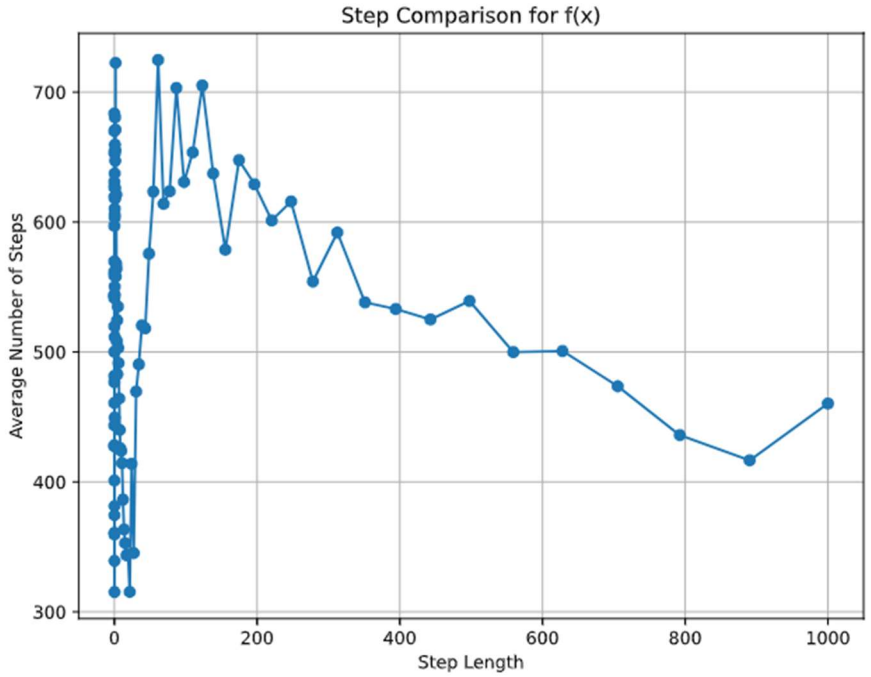
Badanie dla podobnych warunków, tylko z mniejszą redukcją długości kroku = 0.8:



Wyniki:

- Średnia ilość kroków: 201,5325
- Najmniejsza ilość kroków: 166.03
- Największa ilość kroków: 252.35

I jeszcze jedno badanie, tylko tym razem współczynnik redukcji kroku jest mniejszy niż pierwotnie = 0.01:



Wyniki:

- Średnia ilość kroków: 531,05

- Najmniejsza ilość kroków: 315,27
- Największa ilość kroków: 724.81

• Wnioski dla f(x)

Wykresy pokazują, że algorytm nie ma żadnego wyraźnego trendu wydajnościowego. Dla niektórych wartości ilość kroków jest mniejsza, dla innych większa, w zależności od tego, ile kroków musi wykonać od jednego zmniejszenia długości kroku do drugiego.

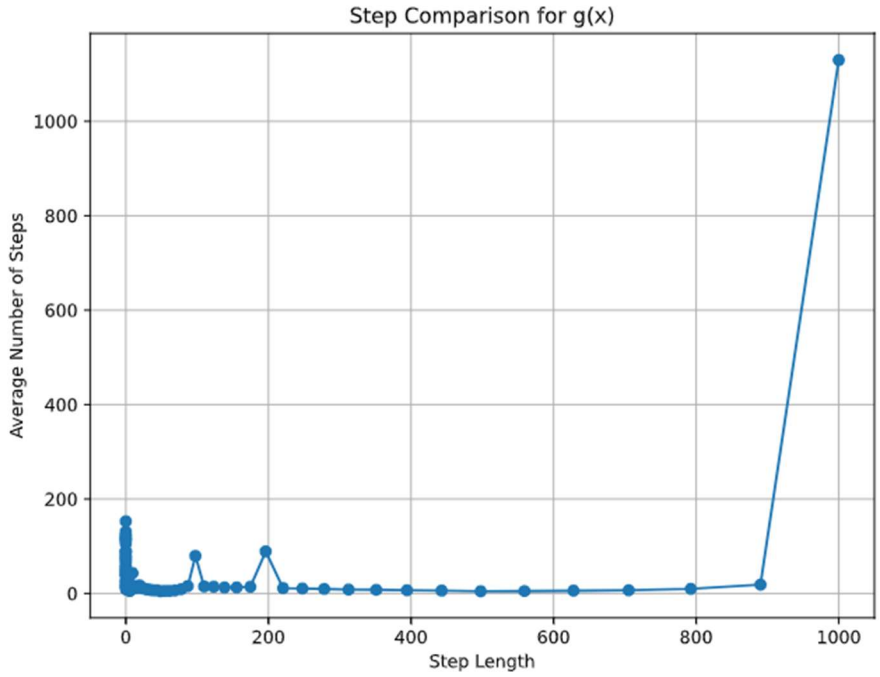
Zmniejszenie redukcji kroku (czyli zwiększenie współczynnika) sprawia, że algorytm wykonuje średnio mniej kroków, ale więcej iteracji kończy się tylko zmniejszeniem kroku. Natomiast zwiększenie redukcji kroku zdecydowanie zwiększa ilość kroków, ale większość iteracji kończy się wykonaniem kroku.

• Badanie dla g(x)

Badanie dla g(x) przeprowadzone zostało tak samo jak poprzednio, z parametrami:

- Zakres punków początkowych: x1, x2 należą do zakresu [-2, 2]
- Długości kroków: 100 różnych długości z przestrzeni logarytmicznej od 0.01 do 1000
- Precyzja: 0.001

Na początku współczynnik redukcji kroku wynosił 0.1:

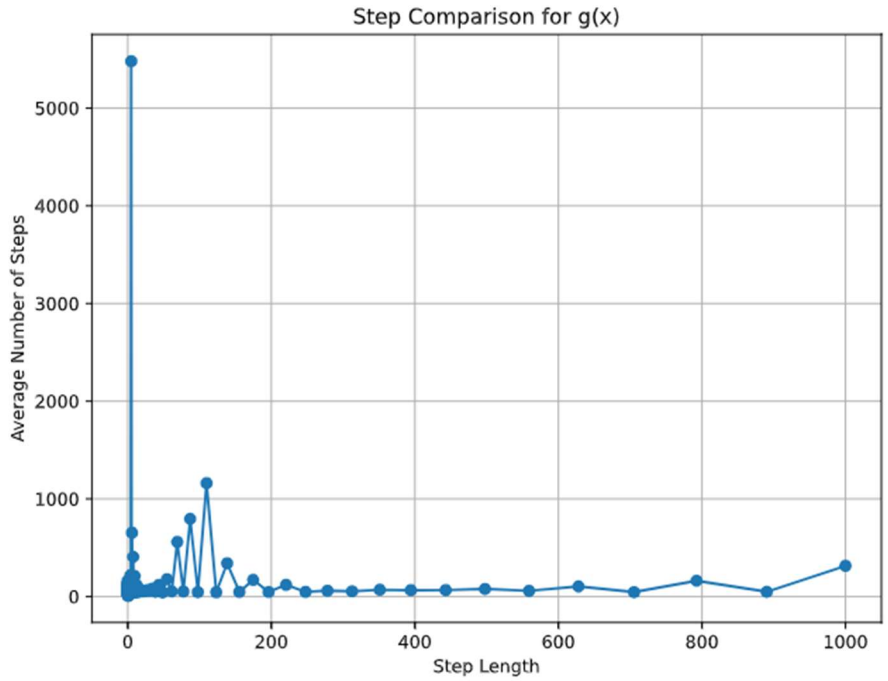


Wyniki:

- Średnia ilość kroków: 42,46

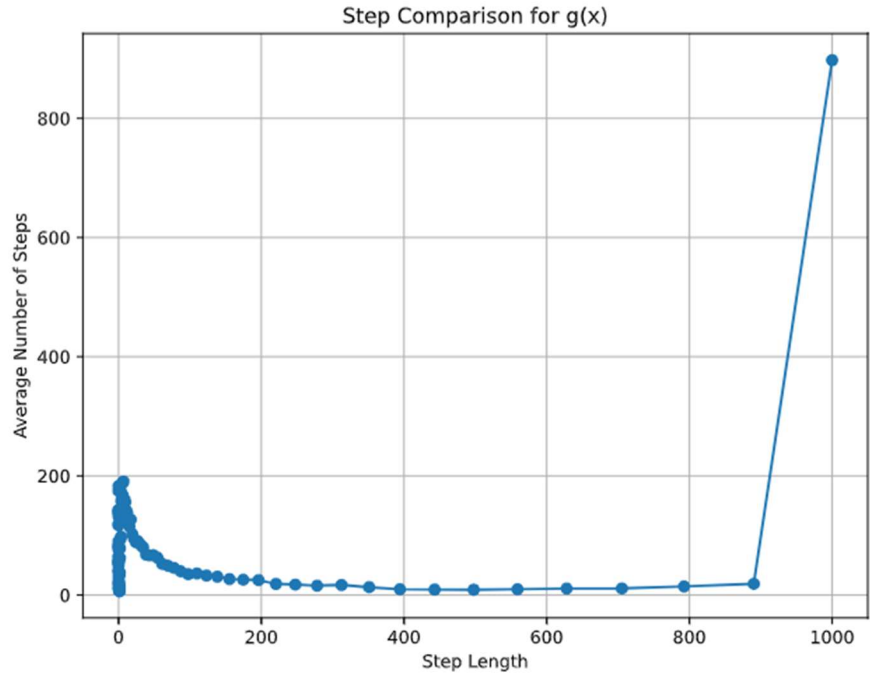
- Najmniejsza ilość kroków: 4.9
- Największa ilość kroków: 1095

Później z współczynnikiem redukcji kroku = 0.8:



- Średnia ilość kroków: 157.09
- Najmniejsza ilość kroków: 8.64
- Największa ilość kroków: 5480.82

I współczynnik 0.01:



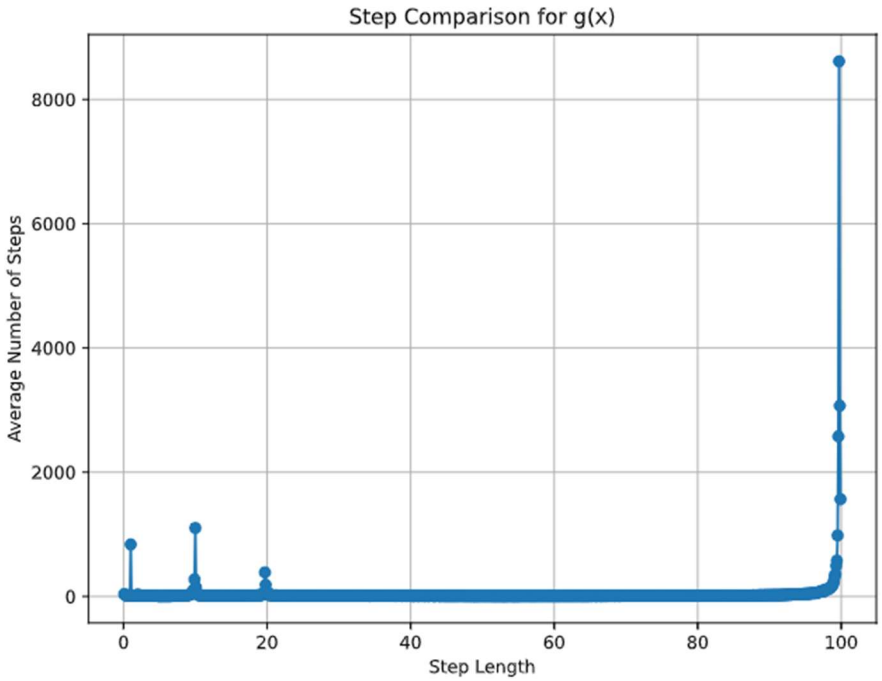
- Średnia ilość kroków: 76.35

- Najmniejsza ilość kroków: 6.32
- Największa ilość kroków: 897.68

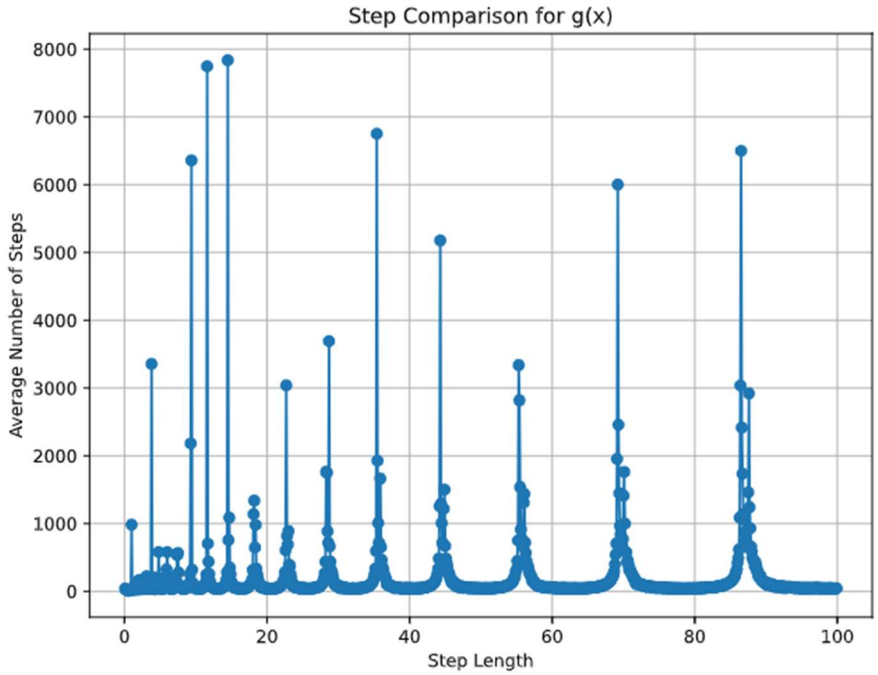
Jak widać, dla każdego współczynnika redukcji kroku następuje ogromny skok w pewnym momencie. Zauważyłem, że np. dla współczynników które są potęgami 10, te skoki następują również dla długości kroków które są potęgami 10. Pokażą to następne wykresy.

W tym badaniu długości kroku nie są skalą logarytmiczną, a po prostu wartościami od 0.1 do 100, co 0.1.

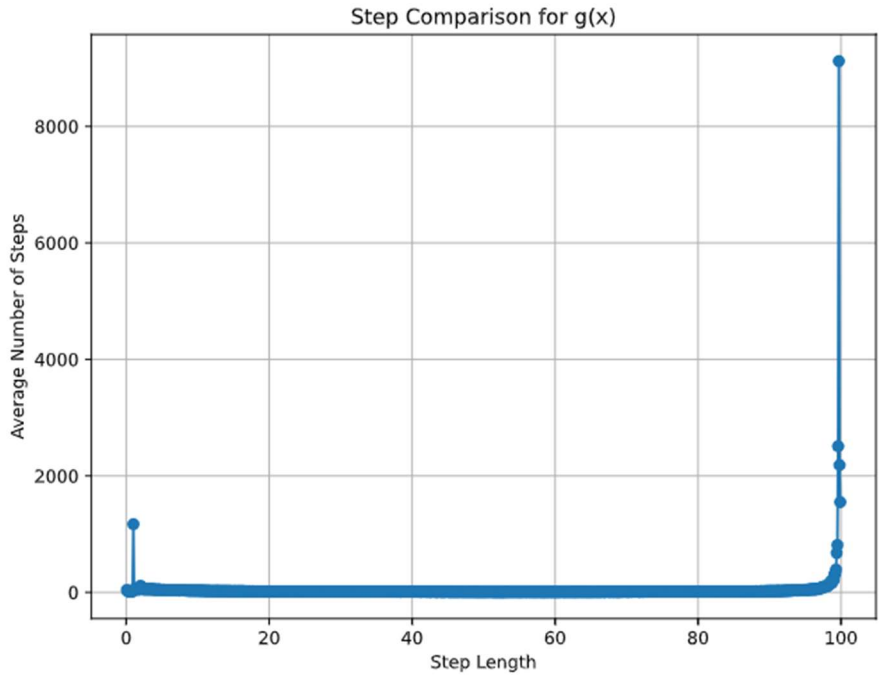
Najpierw wybrałem współczynnik redukcji 0.1:



Później 0.8:



I wreszcie 0.01:



Zwiększenie współczynnika redukcji spowodowało znacznie częstsze skoki. Za to przy współczynniku 10 do potęgi -1, skoki nastąpiły w okolicach 10 do 0, do 1 i do 2. A przy 10 do -2, skoki nastąpiły tylko w okolicach 10 do 0 i 10 do 2.

- Ogólne wnioski na koniec

Ten algorytm bardzo zależy od parametrów, które mu się dobierze. Prostsza funkcja, jak f, zachowywała się zdecydowanie bardziej przewidywalnie niż g. Dodatkowo, funkcję g da się optymalizować tym algorytmem tylko dla zakresu który podałem, ponieważ dla większych wartości wynik funkcji jest rzędu e do potęgi kilkudziesięciu, a sam wykres bardzo się spłaszcza.