WSI Ćwiczenie 1
Gradient prosty

Autor:
Jakub Kowieski

Przedmiot:
WSI, semestr: 21Z

Numer Albumu:
310765

Celem ćwiczenia była implementacja algorytmu gradientu prostego oraz zastosowanie go do znalezienia minimum funkcji f i g. W tej dokumentacji będą zawarte eksperymenty, wnioski, opis i wyniki.

Założenia i stałe:

RANGE = 1000 - liczba iteracji algorytmu
param_f - jest to lista krotek, której elementami są wektor początkowy x (start x) oraz wielkość kroku algorytmu (step size) dla funkcji f(x)
param_g - jest to lista krotek, której elementami są wektor początkowy x (start x) oraz wielkość kroku algorytmu (step size) dla funkcji g(x)

Posiadanie modułów:

- numpy
- Matplotlib
- pytest-3

Legenda:

(Start X, Step Size) - taki zapis to odwołanie się do eksperymentu dla zadanych parametrów
Na wykresach czerwone punkty oznaczają pośrednie minima, a zielony punkt oznacza ostatnie

---

Testowanie i symulacje:

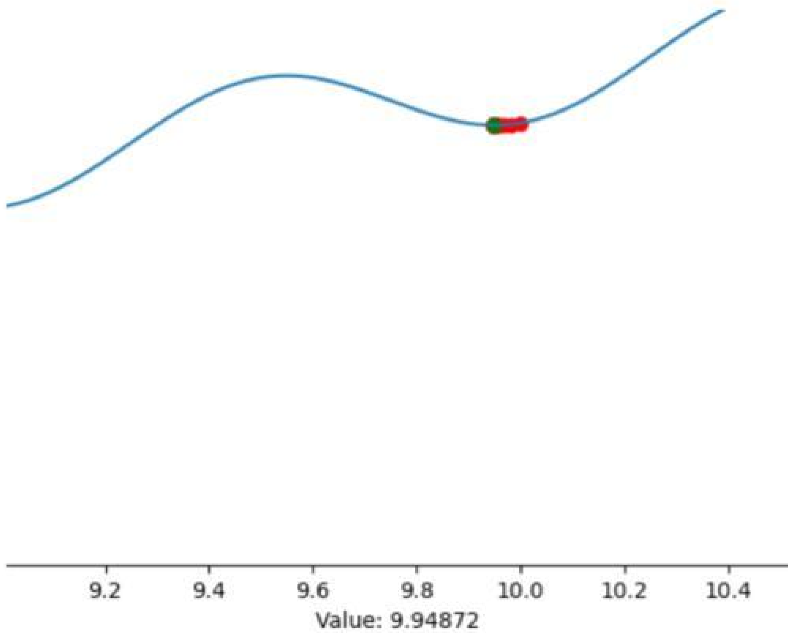
pytest-3 tests - bardzo proste testy czy algorytm działa poprawnie
python3 tables.py - tworzy dwa pliki csv w folderze ./csv z wynikami eksperymentów dla danych parametrów startowych
python3 2d.py - wyświetla wykresy funkcji g(x) dla różnych parametrów params_g (w tej samej kolejności jak w pliku csv)
python3 3d.py - wyświetla wykresy funkcji f(x) dla różnych parametrów params_f (w tej samej kolejności jak w pliku csv)

1. Funkcja g(x)

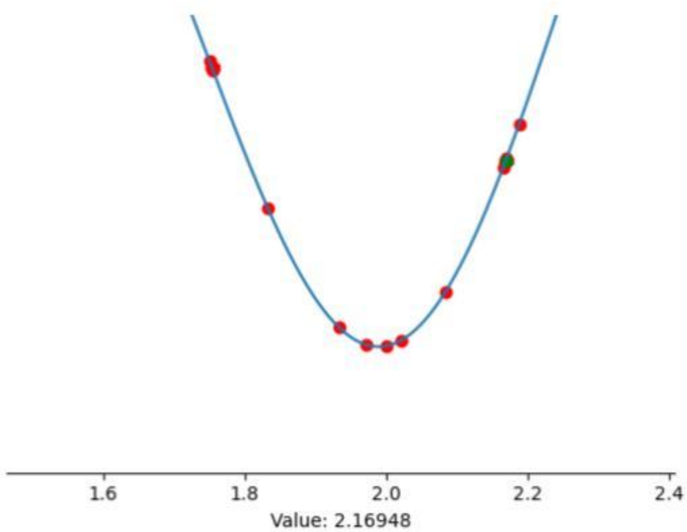
Eksperymenty oraz niektóre wykresy:

Start X	Step Size	Minimum	Value
0	0.001	0.0	0.0
10	0.001	9.94872	99.49
-23	0.003	-22.87023	526.19
2	0.007	2.16948	9.86
25	0.007	24.85474	621.64
-4	0.01	-2.63271	23.65
30	0.03	0.22046	8.2
5	0.1	-2.61666	24.28
-25	0.5	5.81743	39.73
10	1	-372.75869	138958.5

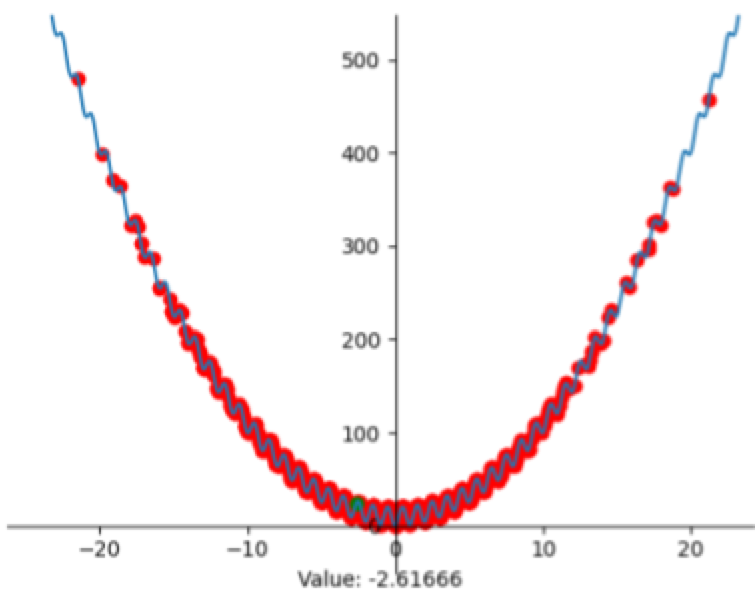
Zielony - znaleziono minimum lokalne
Czerwony - nie znaleziono minimum lokalnego



Wykres dla start_x=10, step_size=0.001

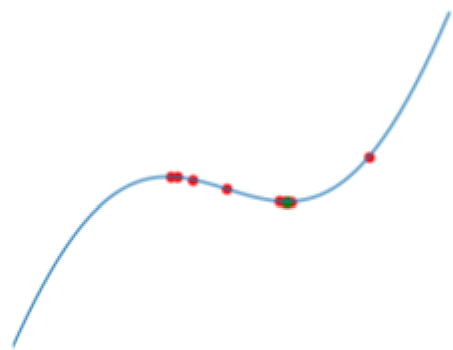


Wykres dla start_x=2, step_size=0.007



Wykres dla start_x=5, step_size=0.1

Algorytm znajdował minima lokalne tylko wtedy, gdy parametr step_size < 0.01. Dla małych kroków algorytm znajdował coraz bliższe x i skok co iteracje się zmniejszał. Dla większych kroków algorytm oscylował i zazwyczaj oscylacja się nie stabilizowała co doprowadzało do nieznajdowania minimum lokalnego. Jednak w przypadku (25, 0.007) oscyluje pomiędzy minimum, ale ostatecznie się stabilizuje i znajduje minimum. Nie tylko wielkość kroku decyduje o pozytywnym wyniku co dobrze obrazuje porównanie (25, 0.007) i (2, 0.007). Wielkość Kroku jest taka sama, ale to odpowiedni punkt początkowy pozwala na znalezienie minima funkcji. Ostatecznie przez to, że funkcja posiada tak dużo minim lokalnych to przy większych wartościach parametru step_size algorytmowi praktycznie co skok zmienia się kierunek i porusza się po całej funkcji (tak jak na wykresie (5, 0.1)).

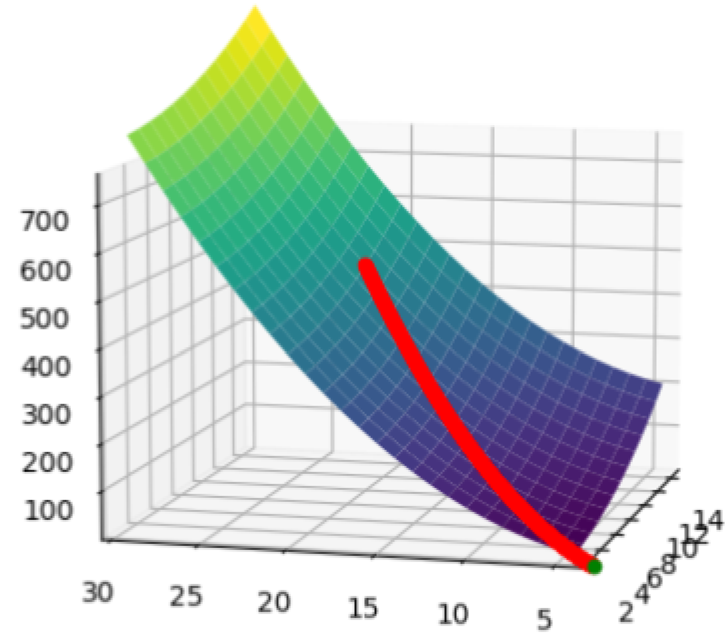


Wykres dla start_x=25, step_size=0.007

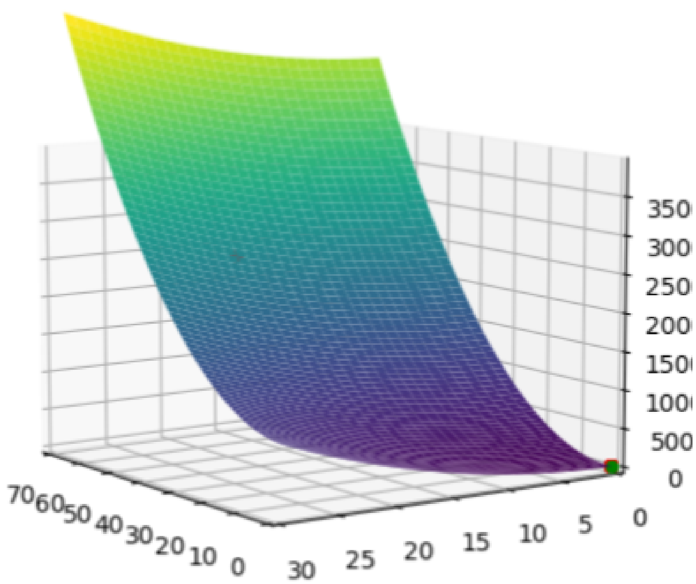
2. Funkcja f(x)

Start X	Step Size	Minimum	Value
(10, 20)	0.001	[1.35065 2.70129]	9.12
(-25, -50)	0.005	[-0.00108 -0.00216]	0.0
(-7, 9)	0.01	[-0. 0.]	0.0
(50, 100)	0.01	[0. 0.]	0.0
(10, -20)	0.03	[0. -0.]	0.0
(-10, 5)	0.05	[-0. 0.]	0.0
(-10, -100)	0.1	[-0. -0.]	0.0
(47, 20)	0.5	[0. 0.]	0.0
(20, 20)	0.8	[0. 0.]	0.0
(-4, 4)	1	[-4. 4.]	32.0

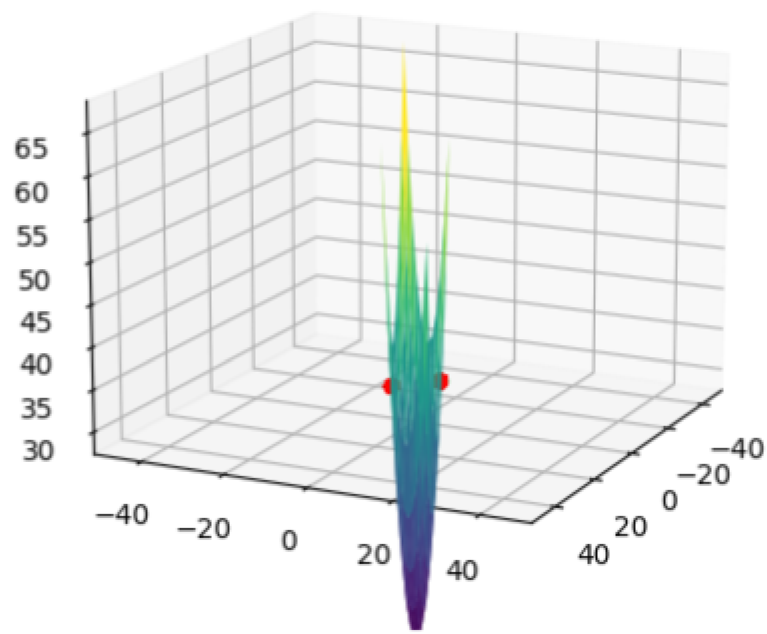
Zielony - znaleziono minimum lokalne
Czerwony - nie znaleziono minimum lokalnego



Wykres dla start_x=(10, 20), step_size=0.001

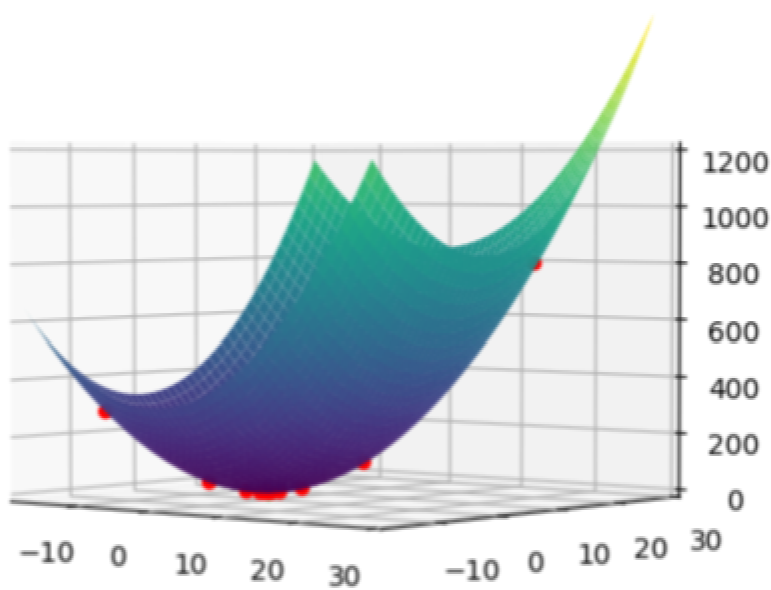


Wykres dla start_x=(47, 20), step_size=0.5



Wykres dla start_x=(-4, 4), step_size=1

Funkcja f(x) posiada tylko jedno minimum i jest ono globalne, więc rozmiar kroku nie powinien mieć tak dużego znaczenia jak w funkcji g(x). Jednakże dla ((10, 20), 0.001) wielkość kroku jest za mała, bo algorytm dąży do minimum, ale niestety za wolno i nie starcza iteracji. Większość eksperymentów jest bardzo podobnych niezależnie od punktu początkowego, wolniej lub szybciej (zależnie od step_size) algorytm znajduje minimum. Dla ((47, 20), 0.5) funkcja po pierwszym skoku od razu znajduje minimum przez, to że gradient_*f(x) * step_size = (47, 20)*, i po odjęciu daje nam x = (0, 0). Oscylacja występuje tylko dla ((20, 20), 0.8), ale ostatecznie się stabilizuje i algorytm znajduje minimum. Jednak gdy step_size=1 na przykład dla ((-4 ,4), 1), algorytm co iteracje zmienia się z (x0, x1) na (-x0, -x1) i odwrotnie. Ostatecznie dzięki posiadaniu jednego minimum dla większości przypadków minimum jest odnajdywane.



Wykres dla start_x=(20, 20), step_size=0.8

Podsumowanie:

Algorytm gradientu prostego sprawdza się najlepiej dla funkcji, w których występuje mało minim lokalnych i są od siebie znacznie oddalone. Wielkość kroku ma duże znaczenie i trzeba je odpowiednio dobierać do każdej funkcji. Jeden raz zwiększenie pozwoli nam na zmniejszenie czasu wykonania algorytmu, ale może też spowodować jego złe działanie. Wektor początkowy x nie ma aż takiego znaczenia jak wielkość kroku, ale też odpowiednio dobrany może pozwolić nam na zwiększenie step_size co w wyniku przyśpieszy nasz algorytm.