WSI: Laboratorium

Ćwiczenie 1: Zagadnienie przeszukiwania i podstawowe podejścia do niego

Paweł Skierś, 310895

Semestr zimowy 2021/22

ZADANIE

Celem ćwiczenia jest implementacja algorytmu gradientu prostego oraz zastosowanie go do znalezienia minimum funkcji f i g. Ponadto należy zbadać wpływ rozmiaru kroku oraz różne punkty początkowe.

Funkcje:

$f(x) = x_1^2 + x_2^2$

$g(x) = x^2 - 10\cos(2\pi x) + 10$

DOKUMENTACJA ROZWIĄZANIA

Program będący napisany na potrzebę tego ćwiczenia podzielony został na trzy pliki: descent.py, plot.py oraz main.py. W dalszej części zostanie omówione za co odpowiadają poszczególne pliki oraz zastosowane w nich rozwiązania.

descent.py

Plik ten odpowiedzialny jest za część algorytmiczną programu. Zawiera funkcję gradient_descent() będącą implementacją algorytmu gradientu prostego. Funkcja ta przyjmuje jako argumenty:

- **x** – jest to punkt początkowy od którego zaczyna się przeszukiwanie
- **function** – jest to funkcja, której minimum chcemy znaleźć, powinna być to faktyczna napisana w pythonie funkcja (innymi słowy ten argument być callable)
- **gradient** – jest to funkcja będąca gradientem funkcji, której minimum chcemy znaleźć. Powinna to być faktyczna funkcja jak argument function
- **epsilon** – jest to dokładność, którą przyjmować będzie algorytm. Jeśli wartość bezwzględna różnicy pomiędzy pewnymi dwoma liczbami będzie mniejsza niż wartość tego argumentu to algorytm będzie traktował te liczby jako równe
- **step** – jest to stała, przez którą mnożony będzie „krok" algorytmu obliczany przy pomocy gradientu
- **step_change** – jest to stała o którą zmniejszony zostanie step w przypadku gdy krok wykonany przez algorytm miał zwiększyć wartość minimalną funkcji

Funkcja zwraca krotkę (x, history) gdzie:

- **x** – jest to punkt, w którym znalezione zostało minimum
- **history** – jest to lista punktów które zostały po kolei sprawdzone przez algorytm

Algorytm wykonywany przez funkcję polega na wyznaczeniu następnego punktu do sprawdzenia w poszukiwaniu minimum, przez odjęcie od obecnego punktu pomnożonego przez stałą step, wektora gradientu minimalizowanej funkcji w obecnie sprawdzanym punkcie. Wyszukiwanie następnego punktu powtarzane jest, dopóki nie zostanie znalezione minimum. Dodatkowo, gdyby wartość funkcji w następnym do przeszukania punkcie miała być większa niż wartość w obecnie sprawdzanym punkcie to wartość step zostaje podzielona przez step_change i wyznaczony zostaje nowy punkt.

plot.py

Plik odpowiedzialny za obrazowanie działania algorytmu gradientu prostego. Robione jest to przez przedstawienie historii sprawdzanych punktów na wykresie. Plik zawiera funkcję plot_history() odpowiedzialną za wyżej wymienione

zadania pliku. Warto zaznaczyć, że funkcja potrafi przedstawiać jedynie wykresy co najwyżej 3 wymiarowe. Funkcja przyjmuje argumenty:

- **history** – lista sprawdzonych przez algorytm punktów
- **function** – funkcja, której minimum było szukane

Funkcja nie zwraca niczego. Przedstawia ona natomiast na wykresie drogę którą przebył algorytm w poszukiwaniu minimum, na tle wykresu minimalizowanej funkcji.

main.py
Jest to plik główny programu. Plik zawiera zaimplementowane funkcje f i g jak i ich gradienty. Ponadto znajdują się w nim również zhardcodowane przyjęte wartości epsilona i zmiany kroku (step_change) oraz funkcja main() pozwalająca na wczytanie punktu startowego i kroku (step) dla minimalizacji funkcji f i g, a następnie znalezienie i wpisanie minimum tych funkcji jak i zobrazowanie działania algorytmu.

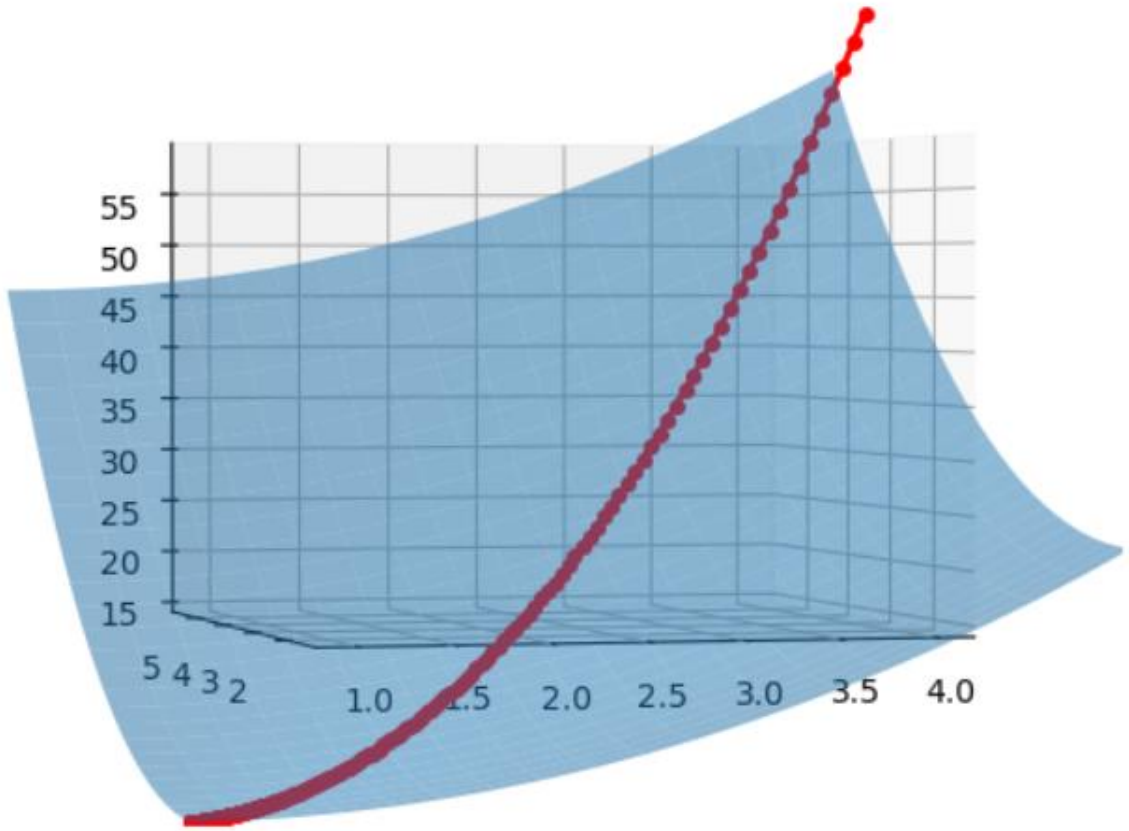
BADANIE WPŁYWU KROKU I PUNKTU POCZĄTKOWEGO

W celu zbadania wpływu kroku i punktu początkowego na działanie algorytmu gradientu prostego przeprowadzono kilka eksperymentów, aby zaobserwować zmiany w wyniku i sposobie do niego dojścia.

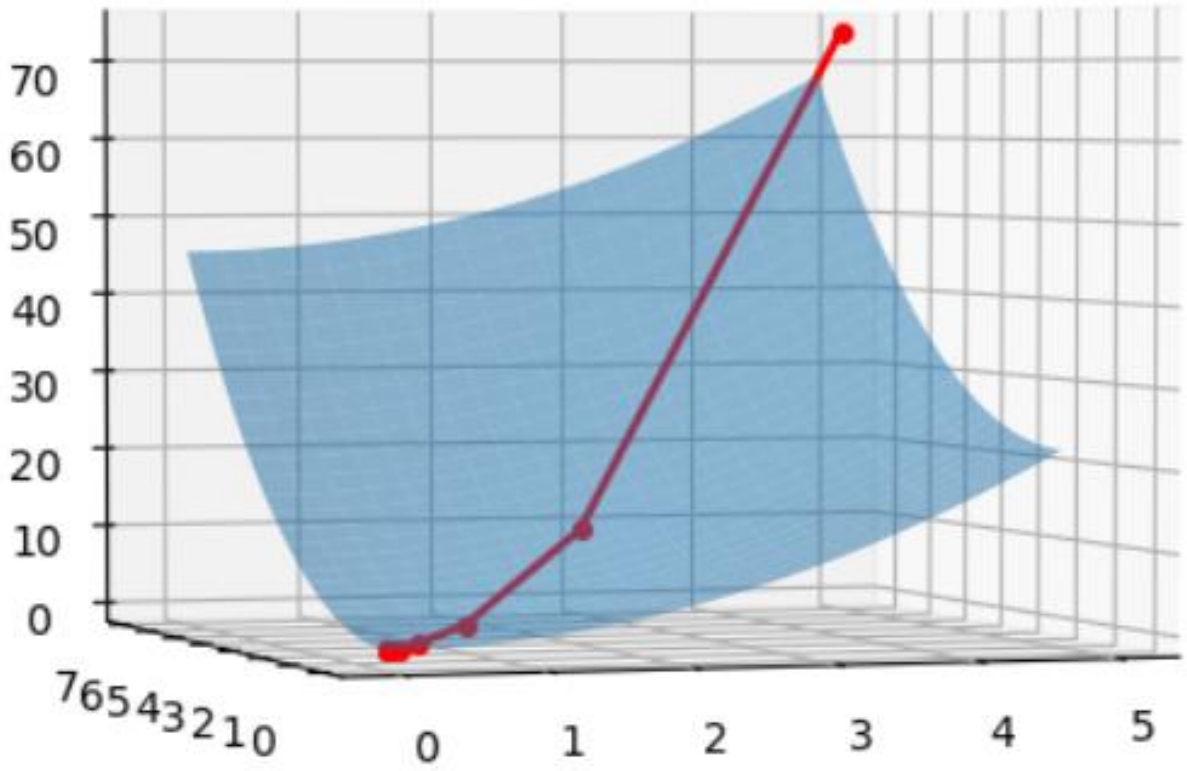
Wpływ kroku

Wpierw zajęto się wpływem kroku. Poniżej widoczne są efekty kilku z wykonanych eksperymentów.

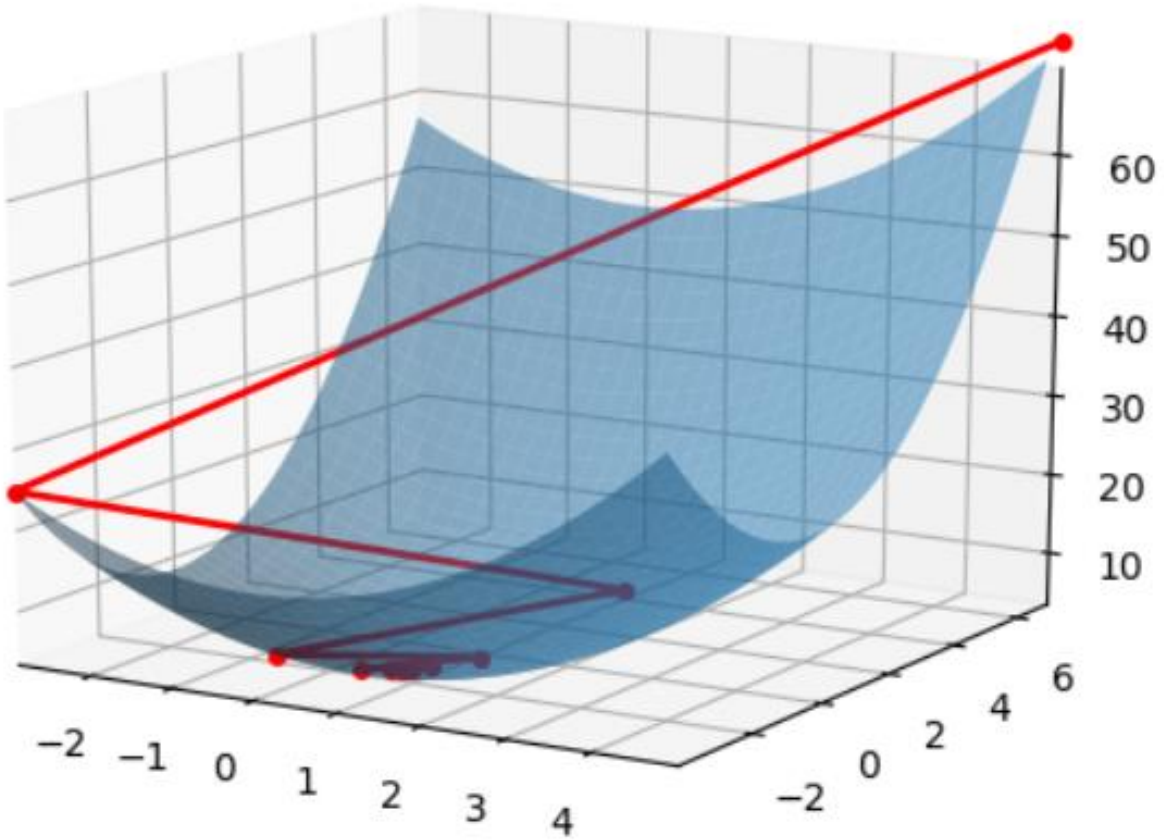
funkcja = f, start = (5,7), krok = 0,01, minimum = (0,0)



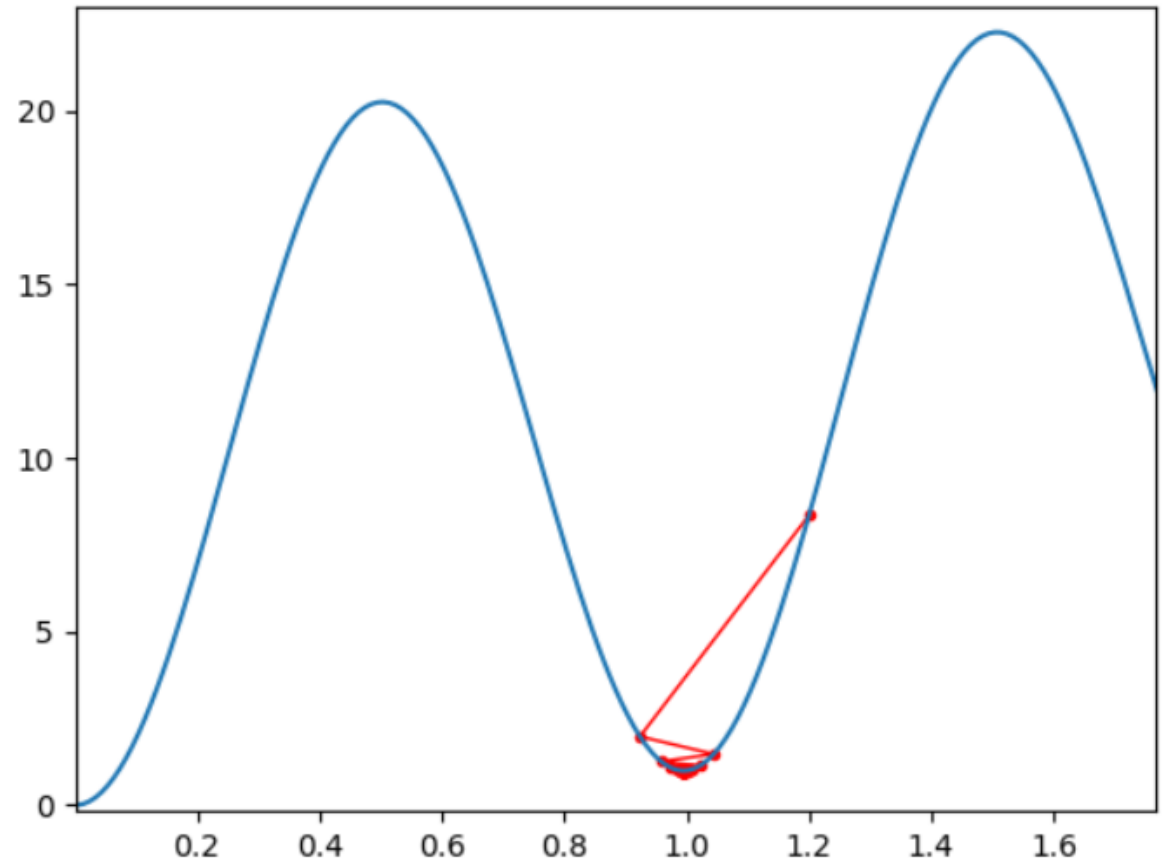
funkcja = f, start = (5,7), krok = 0,3, minimum = (0, 0)



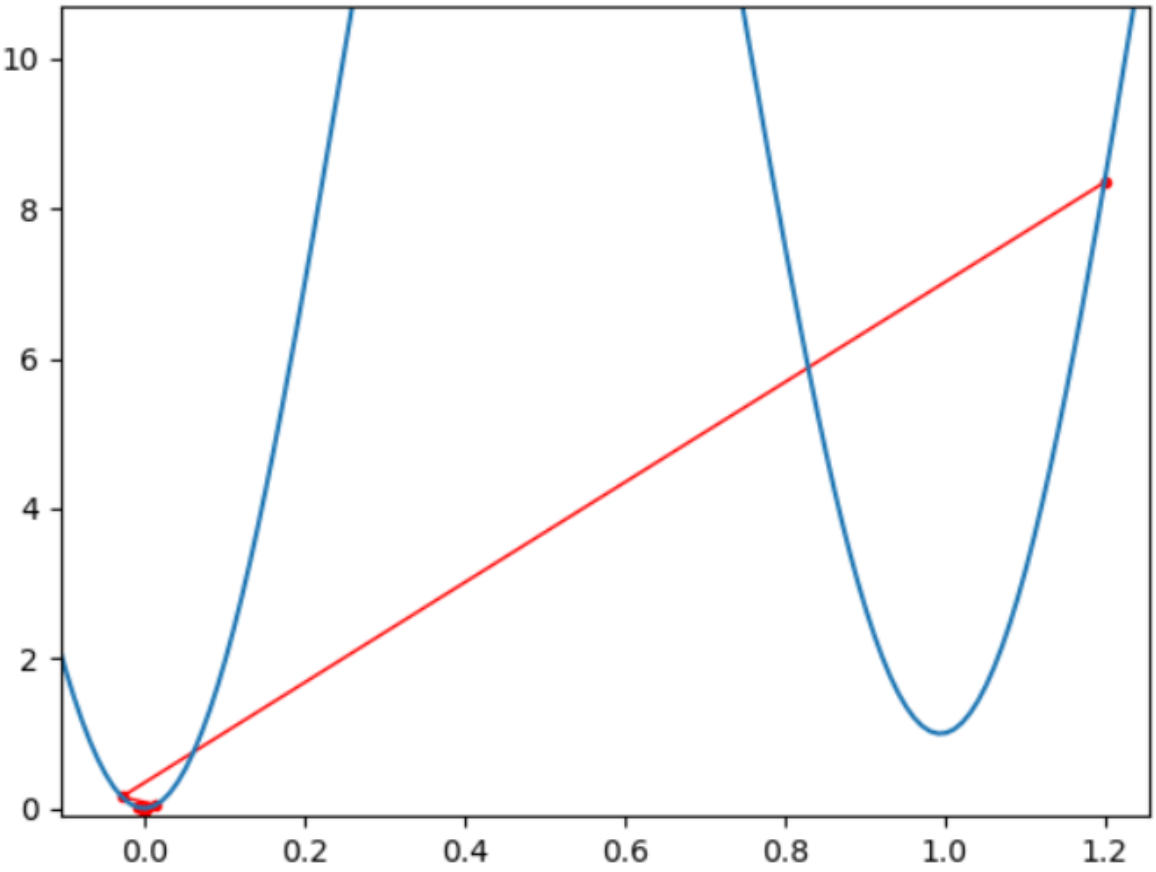
funkcja = f, start = (5,7), krok = 4, minimum = (0,0)



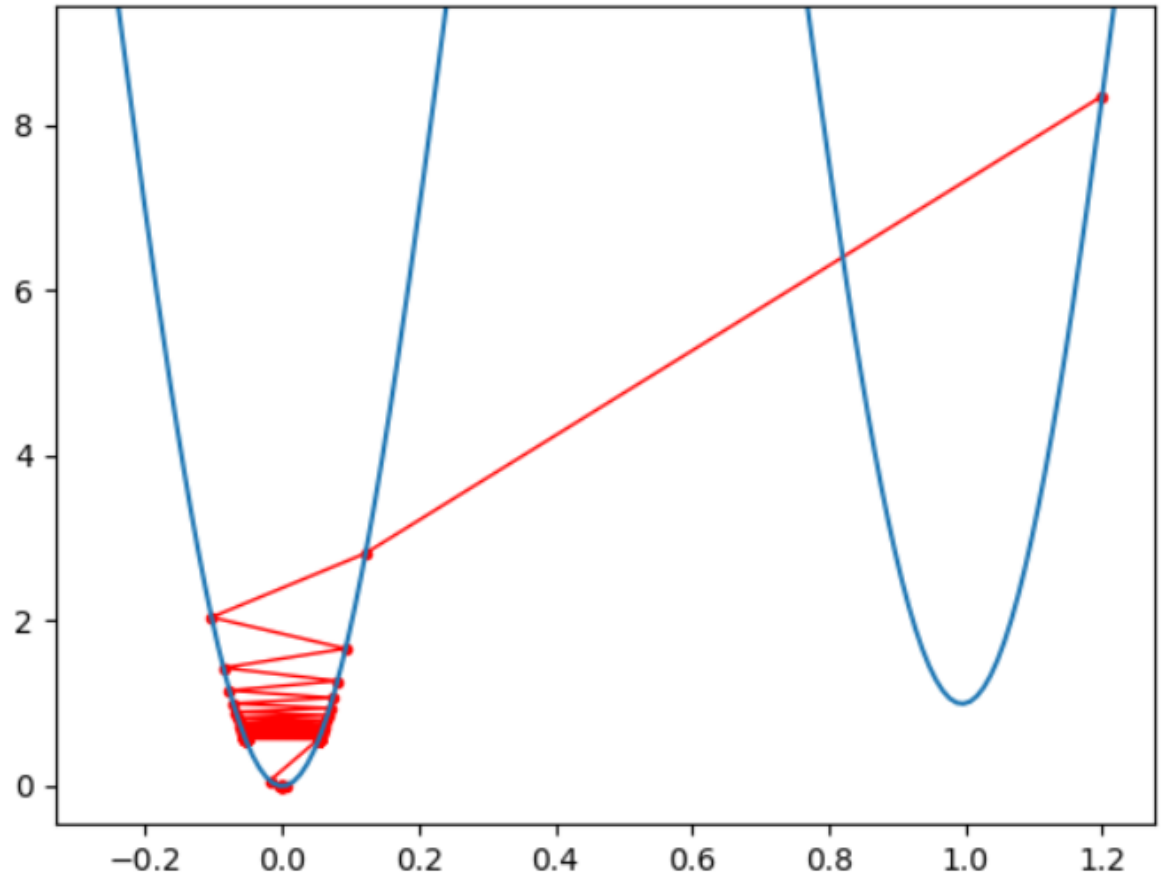
funkcja = g, start = 1,2, krok = 0,01, minimum = 0,99



funkcja = g, start = 1,2, krok = 0.1, minimum = -0,03



funkcja = g, start = 1,2, krok = 1, minimum = 0,03

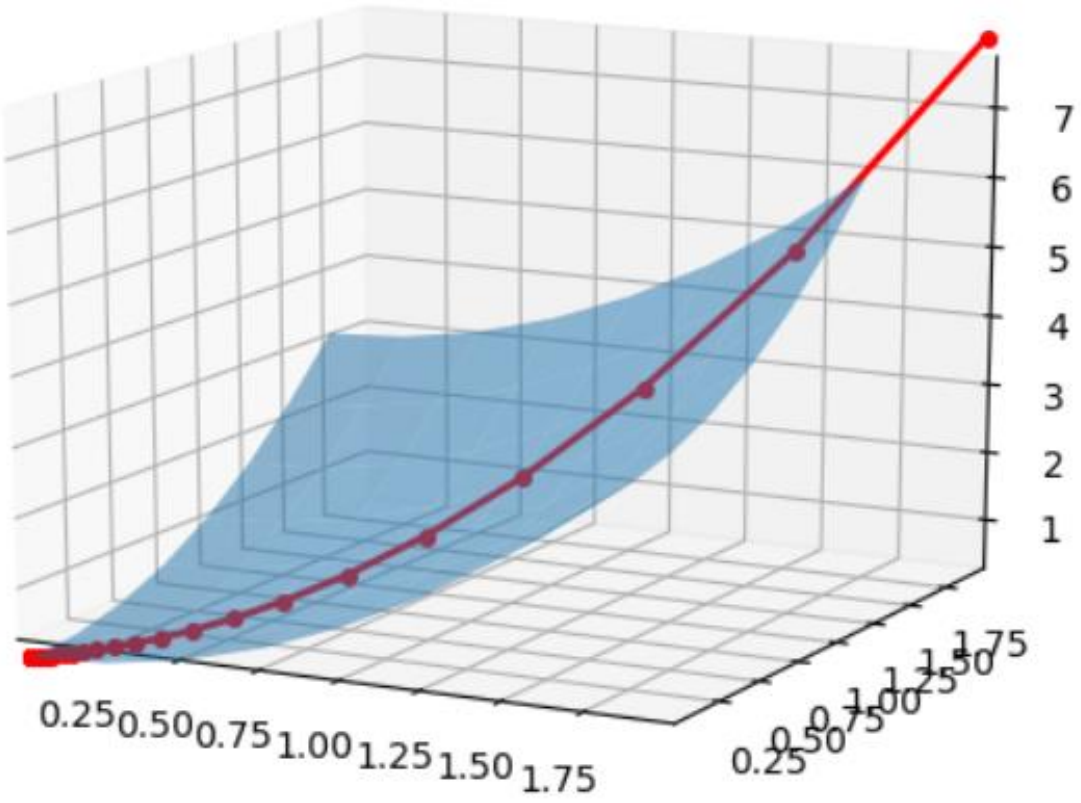


Zauważyć, że algorytm spisuje się najlepiej (najbardziej przewidywalnie) w przypadku gdy krok jest bardzo mały. Wynika to z faktu, że gradient który jest wykorzystywany przy użyciu tego algorytmu liczony jest z definicji dla zmiany współrzędnych równej prawie zero (granica). Dlatego więc gdy zwiększamy wielkość skoku zaobserwować możemy w przypadku funkcji f, że algorytm „przeskakuje” nad minimum, a następnie „wraca się” od miniętego minimum, natomiast w przypadku funkcji g, algorytm przy większym kroku w ogóle przeskakuje najbliższe minimum i osiada w jednm z sąsiednich. Warto jednak zauważyć, zwiększenie kroku, zmniejsza do pewnego momuntu, liczbę kroków potrzebnych do znalezienia minimum, a następnie ją zwiększa. Zwiększenie kroku jest więc uzasadnione dla bardziej „płaskich” funkcji (o małym, mało zmiennym gradiencie). Jeżeli jednak funkcja ma gradient mocno zminiający się, zwiększenie kroku prowadzi do przeskakiwania minimów i ogólnie dość trudnego do przewidzenia działania algorytmu.

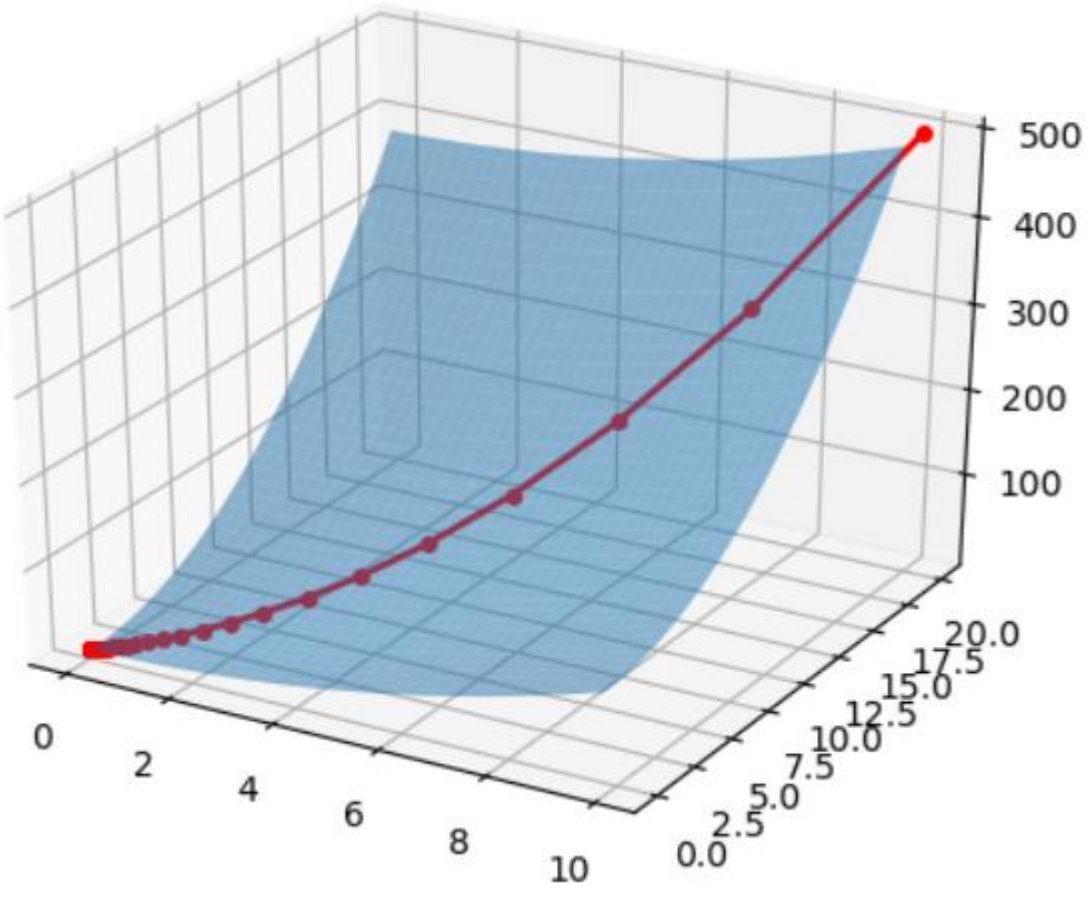
Wpływ punktu startowego

Następnie zajęto się wpływem punktu startowego. Poniżej widoczne są efekty kilku z wykonanych eksperymentów.

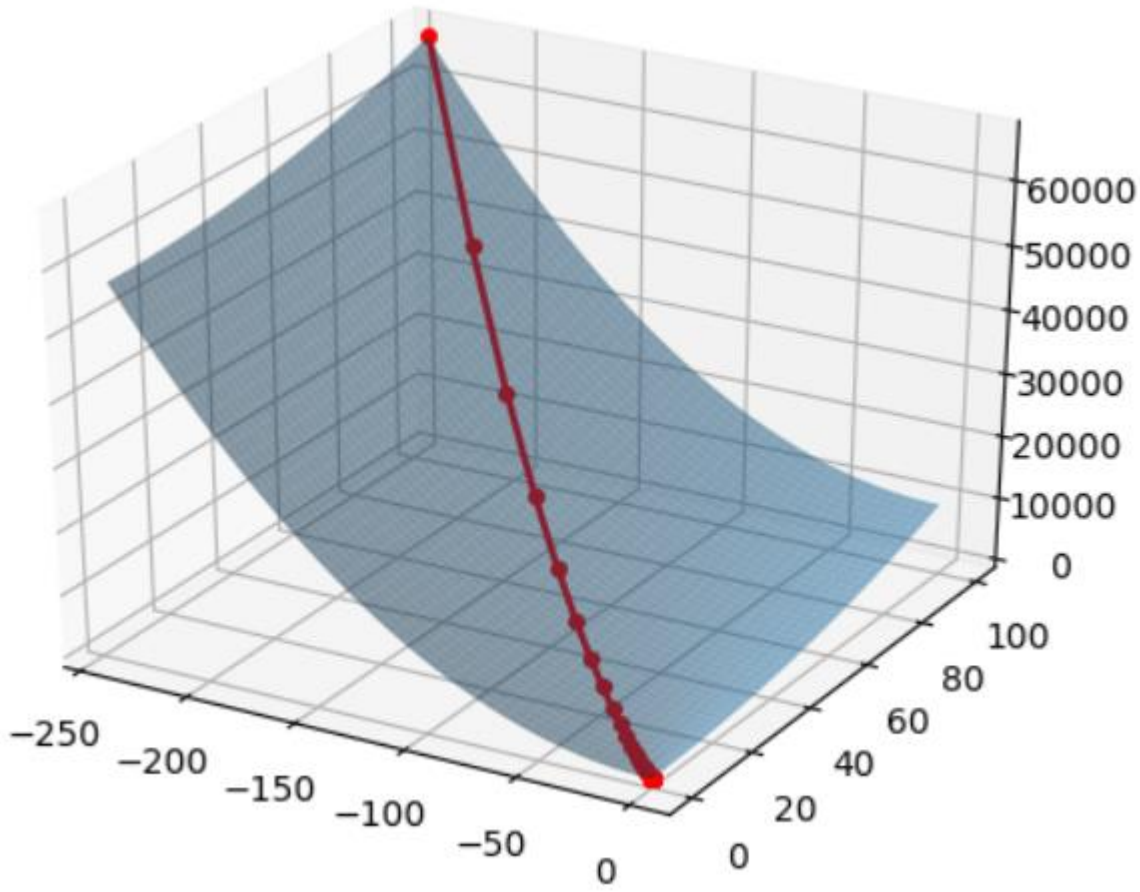
funkcja = f, start = (2,2), krok = 0,1, minimum = (0,0)



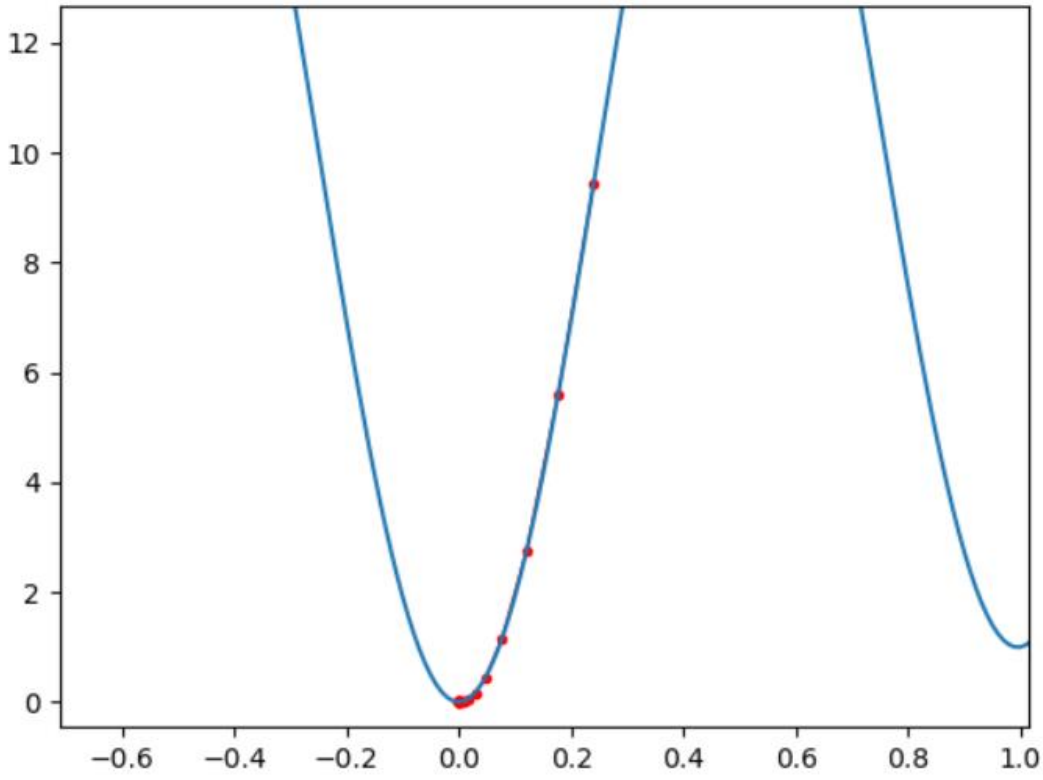
funkcja = f, start = (10,20), krok = 0,1, minimum = (0,0)



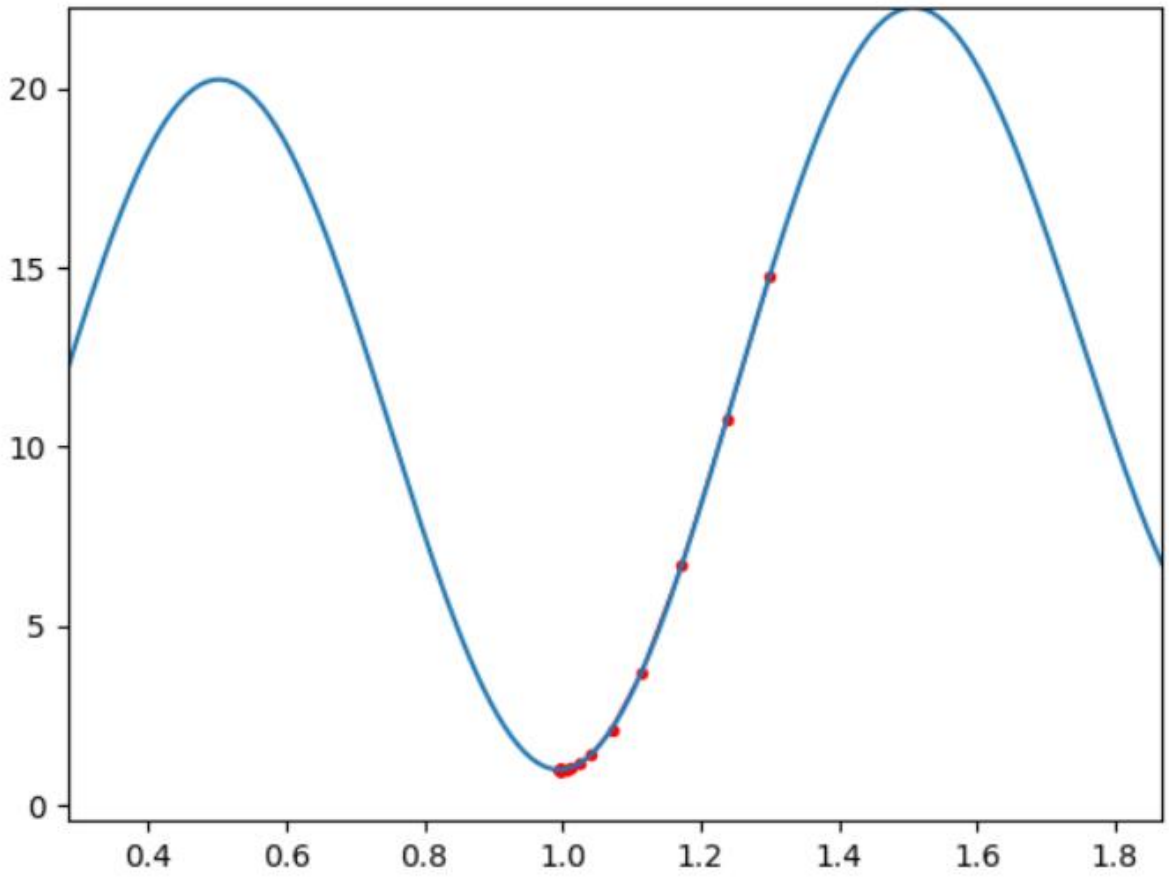
funkcja = f, start = (-240,100), krok = 0,1, minimum = (0,0)



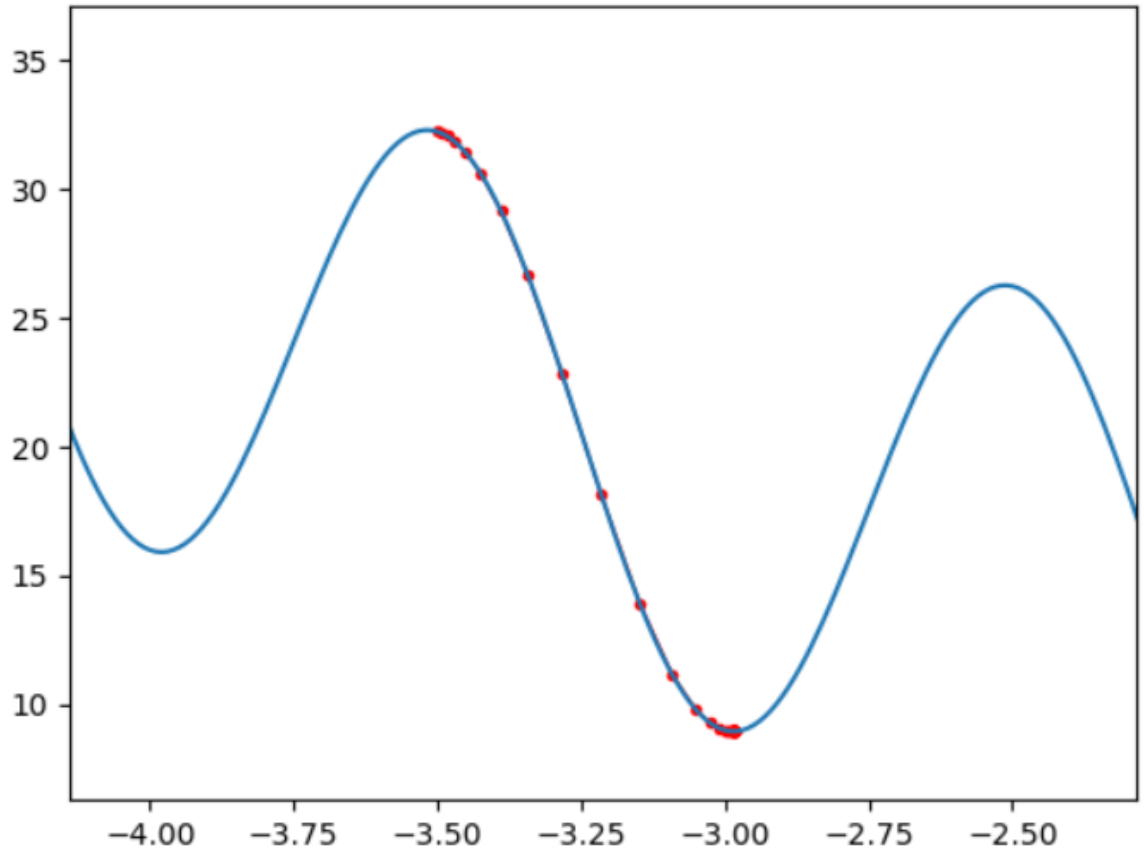
funkcja = g, start = 0,24, krok = 0,001, minimum = 0,02



funkcja = g, start = 1,3, krok = 0,001, minimum = 0,99



funkcja = g, start = -3,5, krok = 0,001, minimum = -2,98



Jak możemy zauważyć, wybór punktu startowego dla funkcji f ma tylko wpływ na to ile kroków musi być wykonane zanim zostanie znalezione minimum tzn. im dalej od minimum był punkt startow tym więcej kroków trzeba było wykonać. Natomiast w przypadku funkcji g to które minimum zostawało znalezione zależało właśnie od punktu startowego. Dzieje się tak dlatego, że algorytm gradientu prostego znajduje (jeśli działa poprawnie czyli gdy krok jest wystarczająco mały) najbliższe punktowi startowego minimum. Dlatego funkcji f – funkcji z jednym minimum, wybór punktu startowego nie ma większego znaczenia, a dla funkcji g ma znaczenie podstawowe.