# **WSI - ćwiczenie 1.**

## *Zagadnienie przeszukiwania i podstawowe podejścia do niego*

Dokumentacja – Mikołaj Olejnik

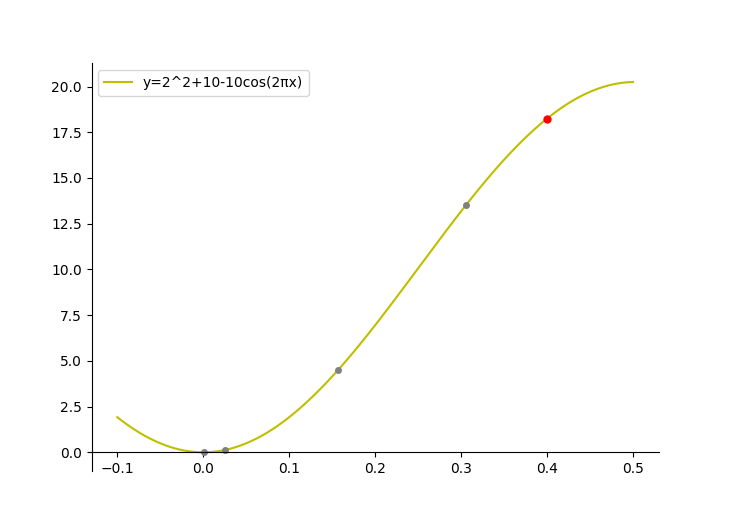
**Treść zadania**

Obraz zawierający tekst

Opis wygenerowany automatycznie

**Wyniki eksperymentów**

Każdą z funkcji przetestowałem 4 razy dla różnych parametrów - wielkości kroku oraz początkowego punktu.

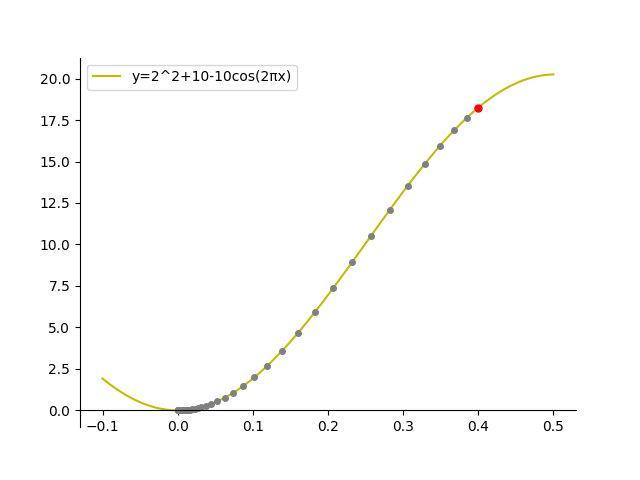


**Wykres 1.**

starting\_x = 0.4

learn\_rate = 0.0025

Liczba iteracji - 4

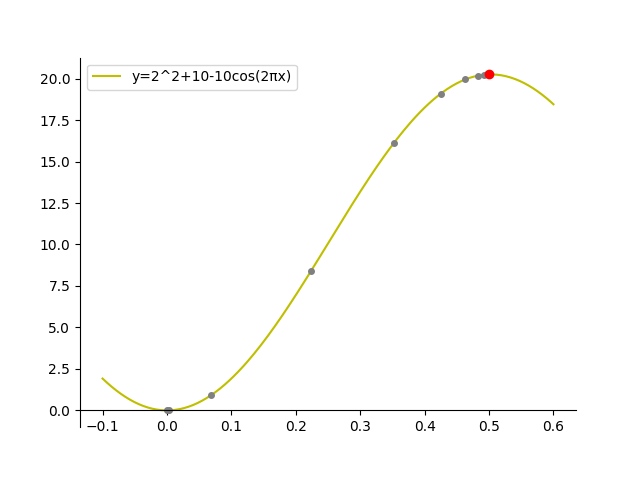


**Wykres 2.**

starting\_x = 0.4

learn\_rate = 0.0004

Liczba iteracji - 44

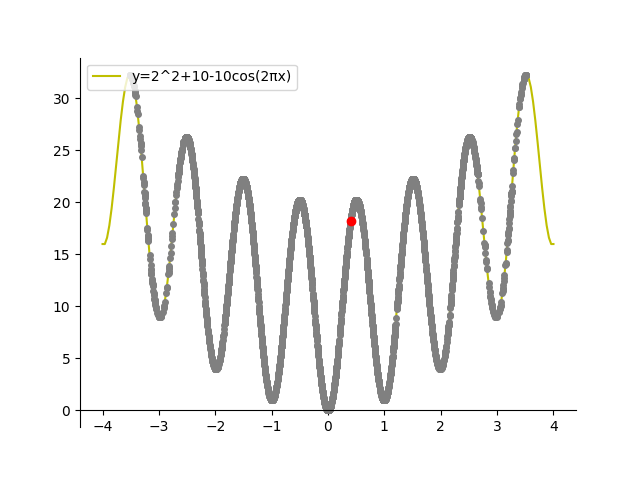


**Wykres 3.**

starting\_x = 0.5

learn\_rate = 0.0025

Liczba iteracji – 10

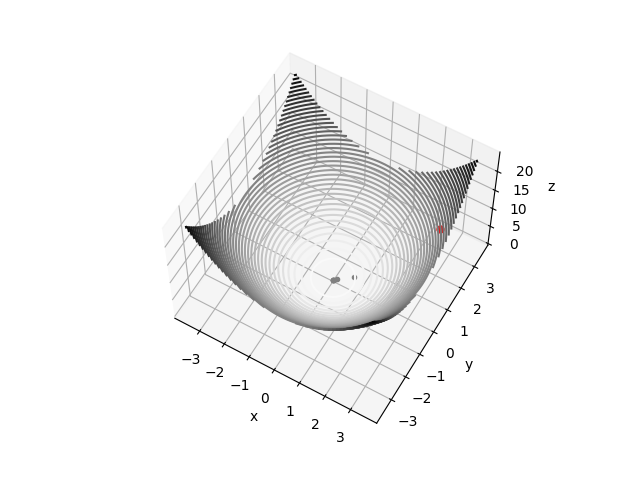


**Wykres 4.**

starting\_x = 0.4

learn\_rate = 0.013

Liczba iteracji – 5000 (przyjęta maksymalna liczba iteracji)

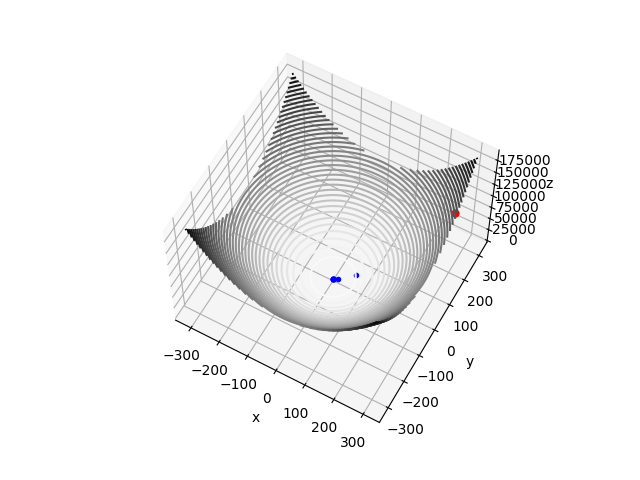


**Wykres 5.**

starting\_x1 = 3, starting\_x2 = 2

learn\_rate = 0.4

Liczba iteracji – 7

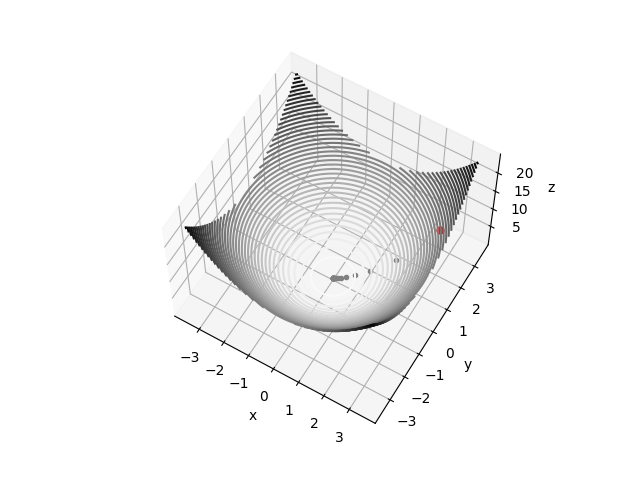


**Wykres 6.**

starting\_x1 = 300, starting\_x2 = 200

learn\_rate = 0.4

Liczba iteracji – 10

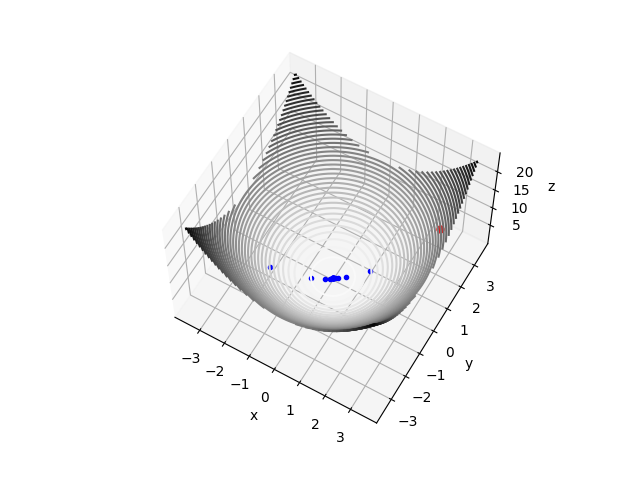


**Wykres 7.**

starting\_x1 = 3, starting\_x2 = 2

learn\_rate = 0.2

Liczba iteracji – 21

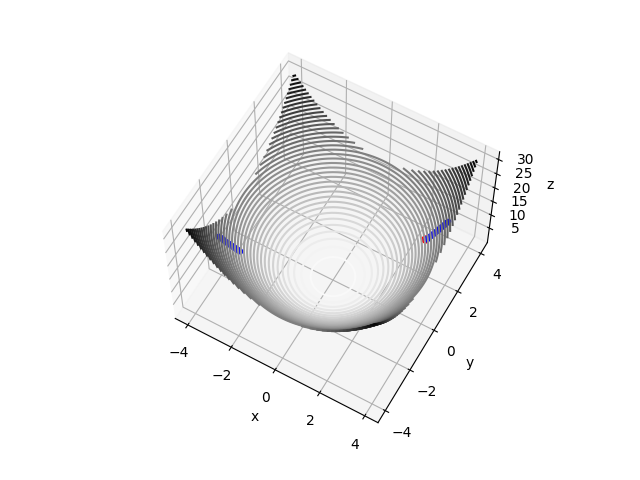


**Wykres 8.**

starting\_x1 = 3, starting\_x2 = 2

learn\_rate = 0.8

Liczba iteracji – 21



**Wykres 9.**

starting\_x1 = 3, starting\_x2 = 2

learn\_rate = 1.001

Liczba iteracji – 100 (przyjęta maksymalna liczba iteracji)

**Obserwacje i wnioski**

Szybkość znalezienia minimum lokalnego zależy od wielkości kroku, więc jakoś zaimplementowanego przeze mnie algorytmu zależy od parametru learn\_rate.

Dobrze dobrana wartość learn\_rate powoduje, że szybko zbliżamy się do minimum, z każdą iteracją znacząco się do niego zbliżając.

Jeśli wybrana wartość wielkości kroku jest za duża, to możliwe jest, że nigdy nie znajdziemy lokalnego minimum. Nawet jeśli mamy trochę szczęścia I algorytm znajdzie jakieś minimum, to może to zająć bardzo dużo iteracji. Sytuację to widać bardzo dobrze na wykresie 4 i 9. W obydwu przypadkach algorytm zatrzymał się po wcześniej zdefiniowanej liczbie kroków – inaczej mógłby działać w nieskończoność nie znajdując żadnego minimum. Ale nawet z za dużą wartością learn\_rate ciągle możliwe jest, że znajdziemy minimum. Widoczne jest to na wykresie 7 i 8.

Jeśli wartość kroku jest za mała, to na pewno znajdziemy minimum, ale będzie to bardzo powolne i zajmie o wiele za dużo iteracji algorytmu niż jest potrzebne. Widać to na wykresie 2. Algorytm potrzebował aż 44 iteracji, aby znaleźć minimum. Dla ponda 6 razy większej wartości learn\_rate algorytm znajduje minimum tylko w 4 iteracje.

To jak szybko znajdziemy minimum i które zależy też od tego jaki punkt wybierzemy jako startowy. Eksperymenty, które przedstawiają wykresy 1 i 3 mają taką samą wartość learn\_rate, ale różny punkt startowy starting\_x. Podejście z starting\_x = 0.5 potrzebuje o 250% iteracji więcej niż to z starting\_x = 0.4. W przypadku funkcji f, punkt startowy również ma znaczenie, ale dość niewielkie. Widać to porównując wynik algorytmu na wykresie 5 i 6.

Ważnym aspektem algorytmu gradientu prostego jest to, że znajduje on minimum lokalne a nie globalne. Jeśli nasz algorytm znajdzie punkt, który znajduje się w obszarze jakiegoś minimum lokalnego, to już z niego nie wyjdzie (przez co nie znajdzie innego minimum – globalnego), chyba że wielkość kroku będzie wystarczająco duża (chociaż raczej chcemy dobierać taką wielkość kroku, żeby algorytm „szedł w dół”).

Minimalizując funkcję zazwyczaj chcemy znaleźć minimum globalne. Niestety ten algorytm nie jest w stanie rozróżnić minimum globalnego od lokalnego.