Zadania:

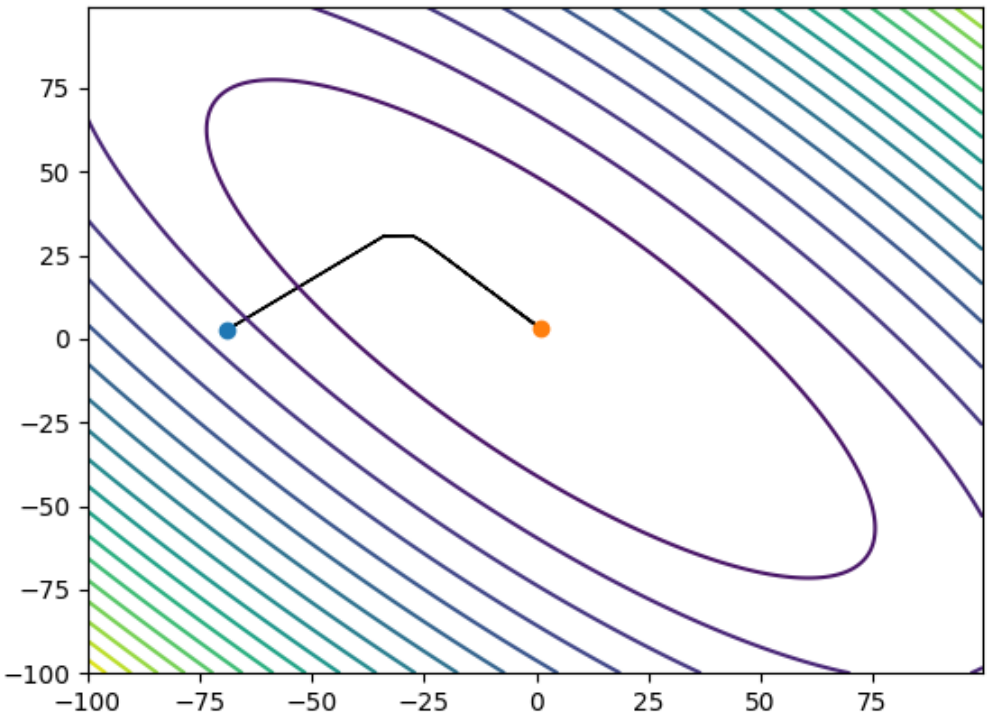
- Zaimplementować metodę najszybszego wzrostu/spadku (minimalizacja, spodziewam się stałego współczynnika kroku, jeśli jednak ktoś chce zrobić więcej i zastosować zmienny współczynnik to ma taką możliwość). Gradient wyliczamy numerycznie.
- Narysować zachowanie algorytmu (kolejne kroki algorytmu jako strzałki na tle poziomic funkcji celu). Uwaga: w praktycznych zadaniach optymalizacji nie da się narysować funkcji celu ponieważ zadania mają wiele wymiarów (np. 100), oraz koszt wyznaczenia oceny jednego punktu jest duży.
- Zastosować metodę do znalezienia optimum funkcji booth w 2 wymiarach, po czym do znalezienia optimum funkcji o numerach od 1 do 3 z CEC 2017 w 10 wymiarach (na wykresie narysować kroki w wybranych 2 wymiarach z 10). Ograniczenia kostkowe przestrzeni to -100, 100.

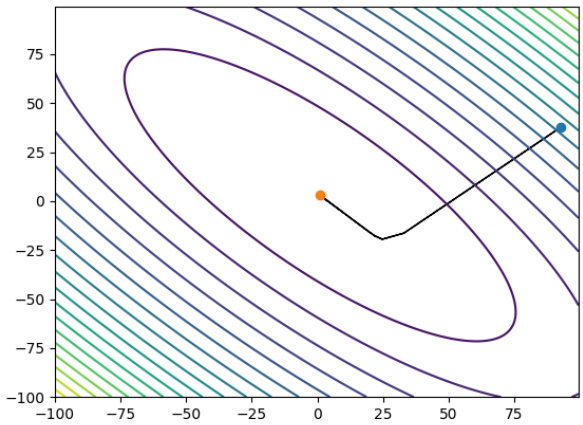
W sprawozdaniu należy zawrzeć wykresy uzyskane stworzonym oprogramowaniem (np. po 3 dla każdej funkcji, dla różnych punktów startowych). Należy podać wartość funkcji celu w punkcie uznanym za optimum.
Pytania:

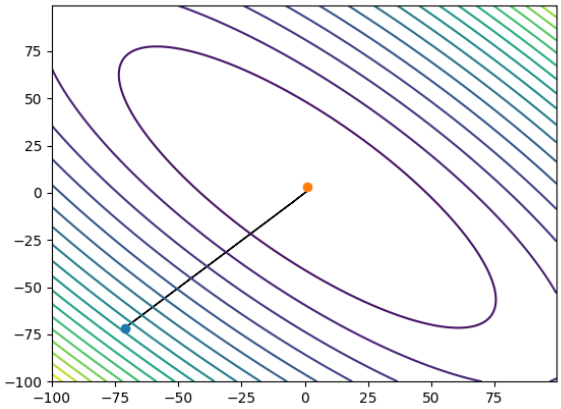
1. Jak wartość parametru beta wpływa na szybkość dojścia do optimum i zachowanie algorytmu? Jakiej bety użyto dla każdej z funkcji?
2. Zalety/wady algorytmu?
3. Wnioski

FUNKCJA BOOTH

Znalezione minimum: 0 (w przybliżeniu)

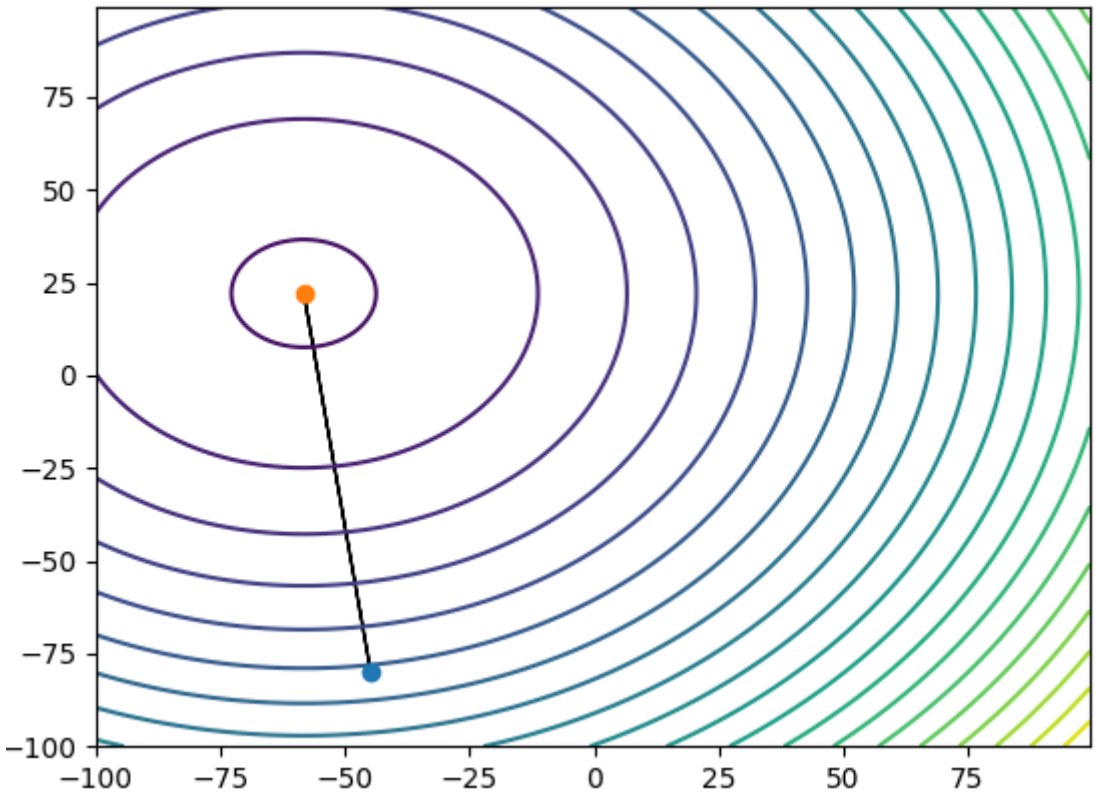


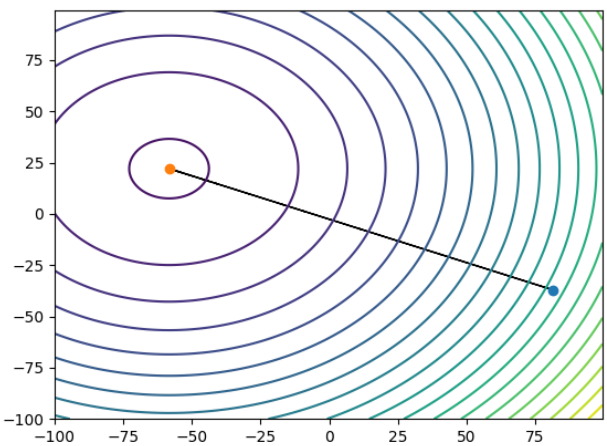


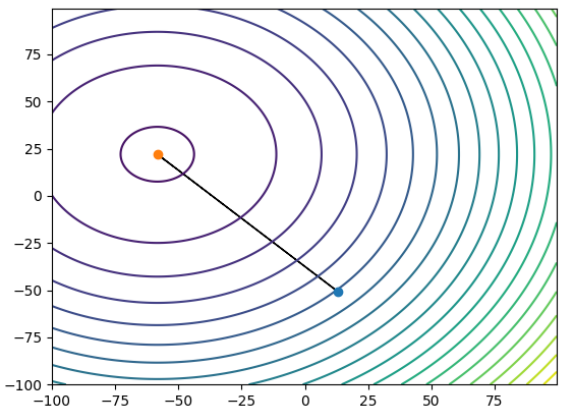


FUNKCJA F1

Znalezione minimum: 105.06

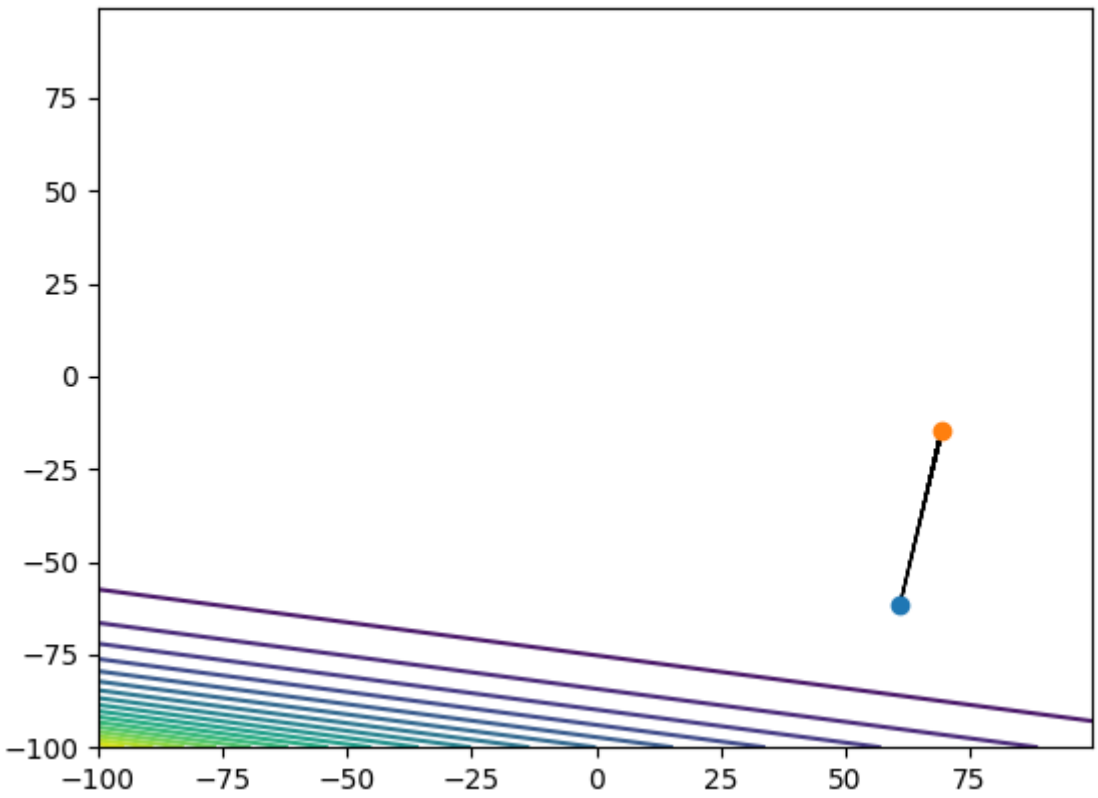


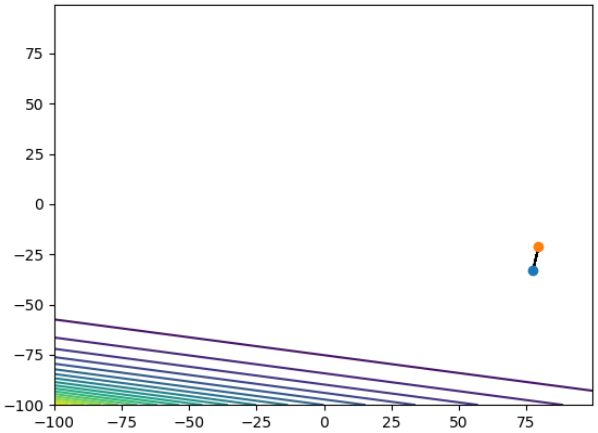


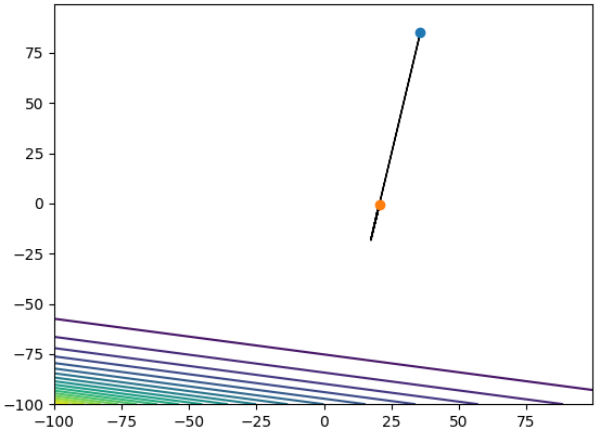


FUNKCJA F2

Znalezione minimum: ok. 10^13

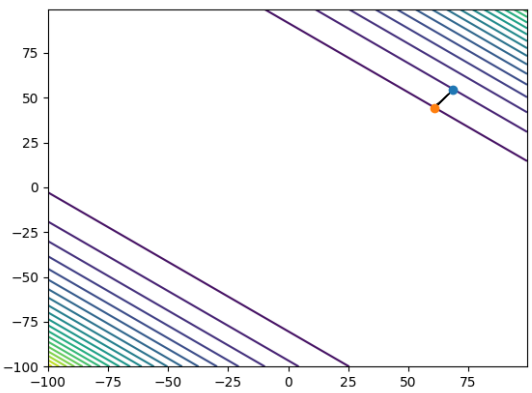


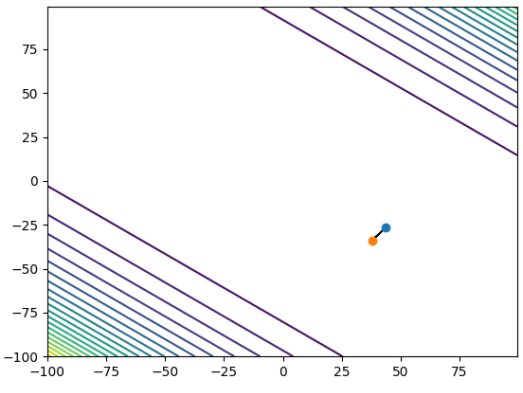


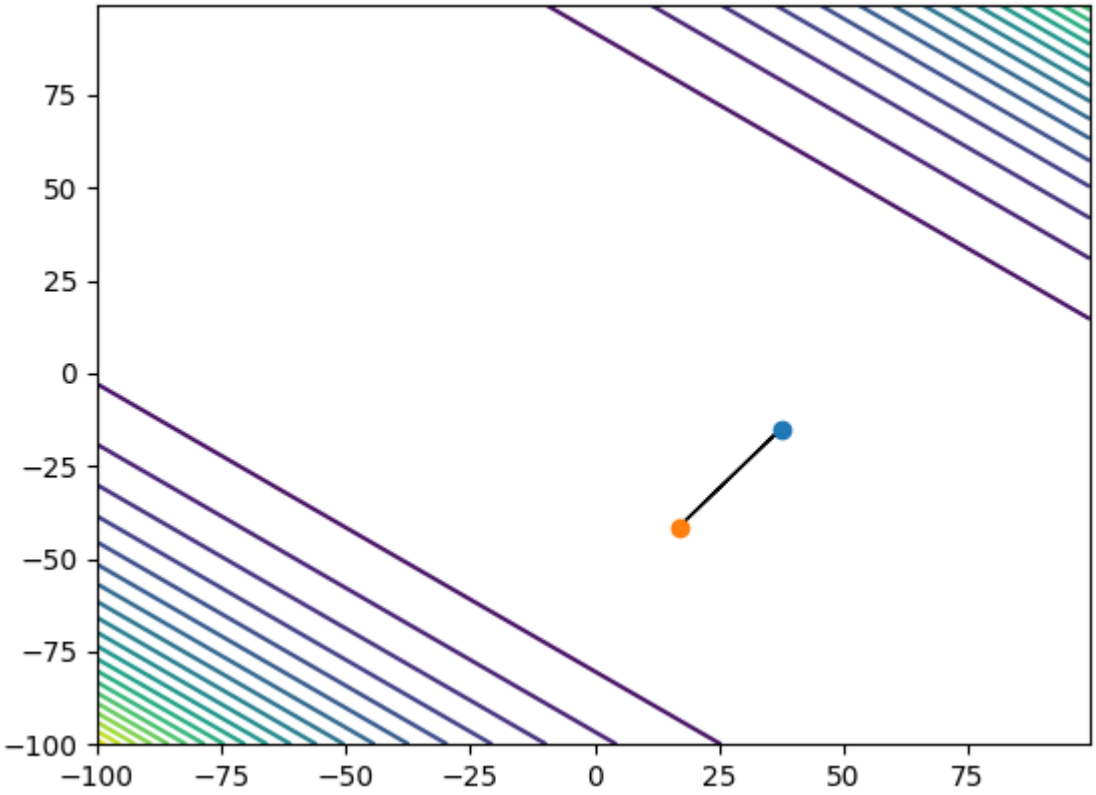


FUNKCJA F3

Znalezione minimum: ok. 40000







1. **Jak wartość parametru beta wpływa na szybkość dojścia do optimum i zachowanie algorytmu? Jakiej bety użyto dla każdej z funkcji?**

Im większa beta, tym szybciej algorytm dąży do optimum, przy czym za duża beta może spowodować że algorytm „przestrzeli” optimum zbyt dużym krokiem, co może spowodować że nigdy nie znajdziemy optimum lub znajdziemy później niż z niższą wartością bety.

Wartości bety użyte dla konkretnych funkcji:

Booth – 0.05

F1 – 8*10^(-9)

F2 – 2*10^(-17)

F3 – 9*10^(-9)

2. **Zalety/wady algorytmu?**

Zalety:

- Algorytm prosty w implementacji jeśli umiemy policzyć gradient funkcji celu
- Możemy go zastosować do wielowymiarowych funkcji

Wady:

- Musimy ręcznie dobrać parametr beta
- Dla bardziej skomplikowanych funkcji ciężko jest dobrać stały parametr beta, który umożliwiłby znalezienie optimum
- Algorytm może „utknąć” w optimum lokalnym

3. Wnioski:

Algorytm bardzo dobrze nadaje się do badania prostych funkcji z jednym optimum, czego przykładem jest funkcja booth, algorytm znajduje minimum w mniej niż 100 krokach

Przy bardziej skomplikowanych funkcjach pojawiają się problemy:

Dla f1 jest w stanie znaleźć optimum, ale często zwraca wynik w okolicach (np. 2000), zwiększenie liczby kroków nie pomaga

Funkcja f2 jest bardzo daleka od optymalizacji tą metodą, zwiększenie bety powoduje kompletną zapaść algorytmu, czyli wartości dążą do nieskończoności

Funkcja f3 jest również ciężka do zoptymalizowania tą metodą i tutaj również jest taki sam problem z betą jak przy funkcji f2.

Podejrzewam, że dla funkcji f1-f3 należałoby zastosować zmienną betę, np. rosnącą wraz z kolejnymi krokami algorytmu. Być może wówczas wyniki byłyby o wiele bardziej zadowalające.