Opis zaimplementowanego algorytmu

Celem zadanie było zaimplementowanie algorytmu gradientu prostego - służącego do (sub)optymalizowania zadanych funkcji matematycznych wielu zmiennych. Algorytm polega na obliczaniu wartości gradientu w zadanym punkcie - a więc kierunku wzrostu wartości funkcji - i "przejściu" do kolejnego punktu w kierunku przeciwnym do gradientu o wartość iloczynu gradientu i parametru kroku. Kluczowym jest więc odpowiednie dobranie kroku. W mojej implementacji znajdują się dwa rozwiązania - z góry ustalony krok losowany z zadanego przedziału(potencjalnie można uruchomić program dla paru wartości z przedziału) i krok dynamiczny, zmieniający się w każdej iteracji, wyszukiwany przez - "Backtrack line search". Ponieważ wybór punktu ma również ogromne znaczenie na wyniki gradientu prostego, program można uruchomić dla wielu punktów, losowanych z podanej dziedziny

Backtrack line search

Polega na itercyjnym zmniejszaniu potencjalnego kroku w danej iteracji, aż do osiągnięcia zadowalającej optymalizacj z bieżącego miejsca.

Planowane eksperymenty numeryczne

- Uruchomienie obu algorytmów dla zadanego punktu/kroku
- Uruchomienie obu algorytmów dla wielu kroków i punktów
- Uruchomienie obu algorytmów dla funkcji dwóch zmiennych w losowym punkcie, punkcie krytycznym i na "płaskim terenie"

W każdym eksperymencie program dokonuje zapisu obecnego obliczanego punktu, wartości funkcji w punkcie i iteracji, po za tym program mierzy czas wykonania.

Dwa pierwsze eksperymenty zostały uruchomione na proponowanej funkcji:

$$q(x) = \sum_{i=1}^{n} \alpha^{\frac{i-1}{n-1}} x_i^2, x \in [-100, 100]^n \subset \mathbb{R}^n, n = 10$$

Trzeci dla:

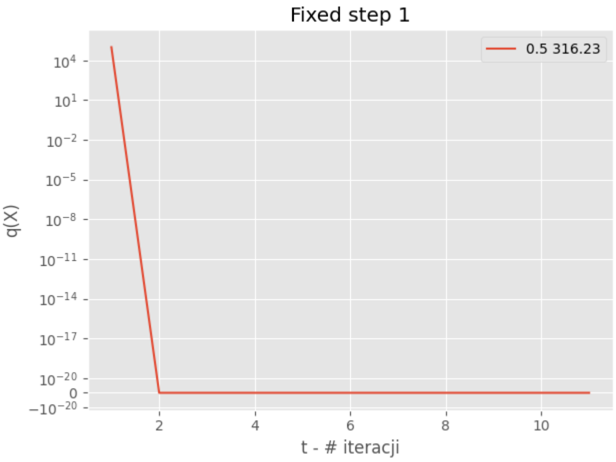
$$q(x,y) = (1 - x^2 + y^3)e^{-(x^2+y^2)}$$

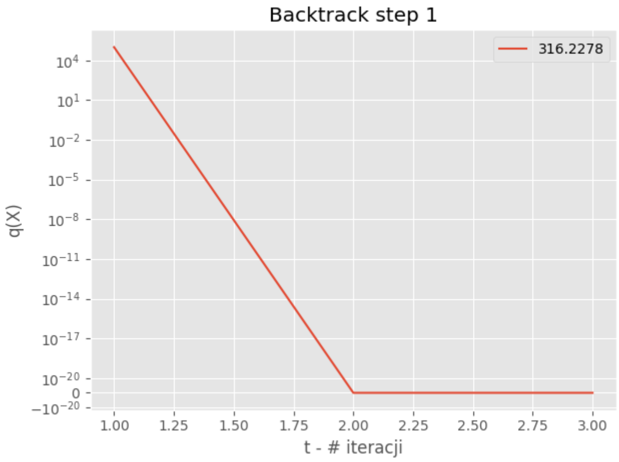
Wyniki

Wywołanie dla zadanego kroku i punktu

Etykieta w przypadku Fixed step stanowi przyjęty krok i odległość punktu startowego od Punktu X=0 - odpowiednio dla Backtrack step

α=1

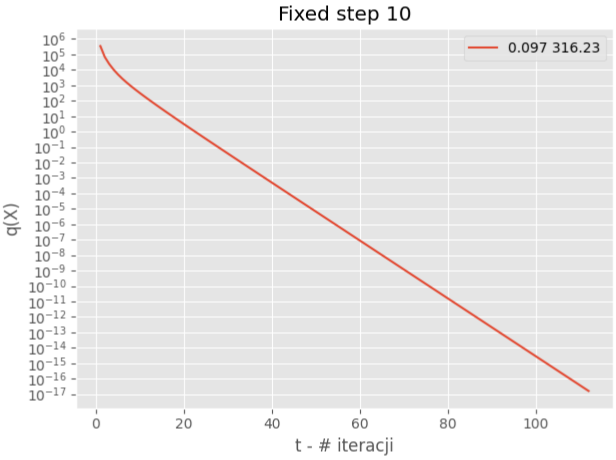


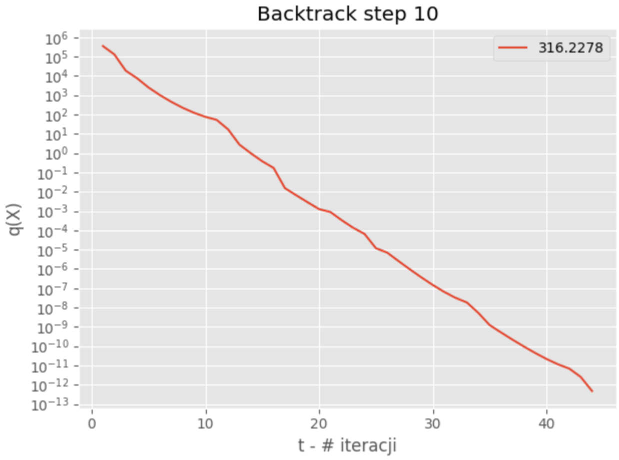


Czas

```
----------
Experiment:Fixed step 0.5 316.23
Result: 0.0
Execution time: 0.010082923996378668
----------
Experiment:Backtrack step 316.22776601683
Result: 0.0
Execution time: 0.009788026996830013
```

α=10

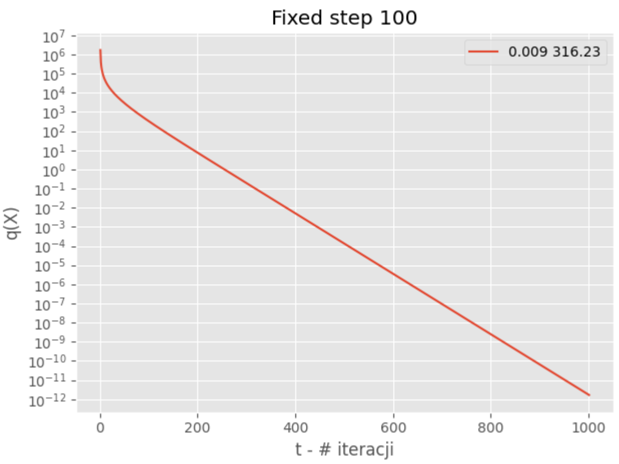


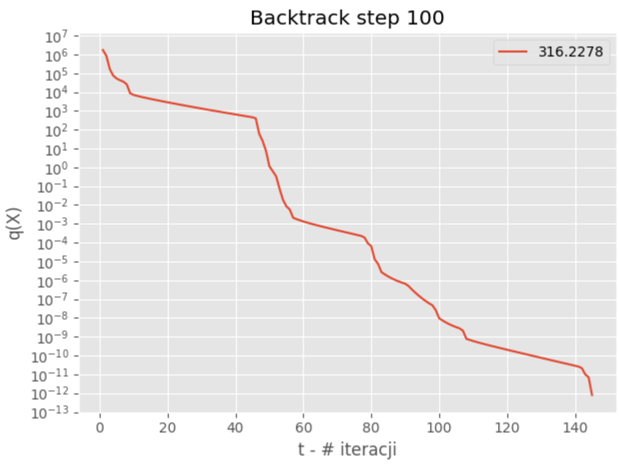


Czas

```
----------
Experiment:Fixed step 0.097 316.23
Result: 1.608346515453692e-17
Execution time: 0.10827062199678039
----------
Experiment:Backtrack step 316.22776601683
Result: 4.793633982835726e-13
Execution time: 0.228224542006501
```

α=100



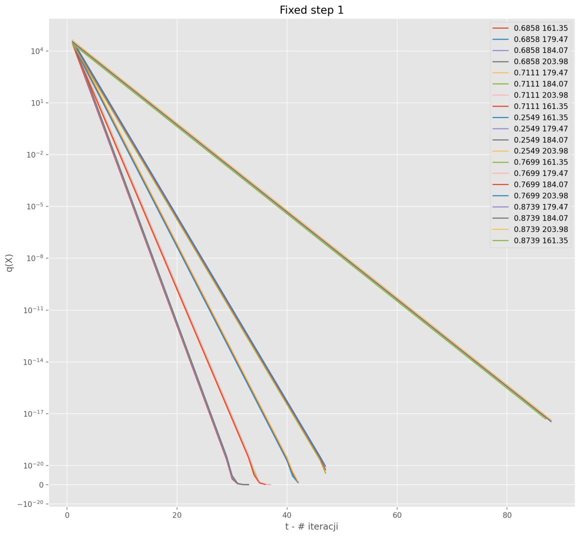


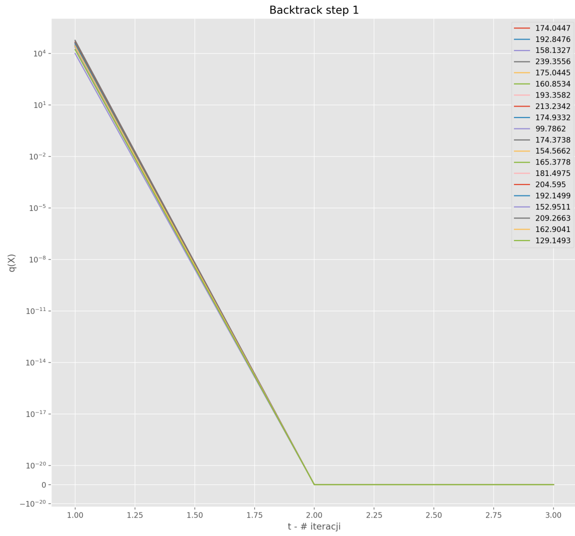
Czas

```
----------
Experiment:Fixed step 0.009 316.23
Result: 1.6709966306609213e-12
Execution time: 0.9561672269992414
----------
Experiment:Backtrack step 316.22776601683
Result: 8.036478306381011e-13
Execution time: 1.0663950169982854
```

Wywołanie z wieloma punktami i krokami

α=1

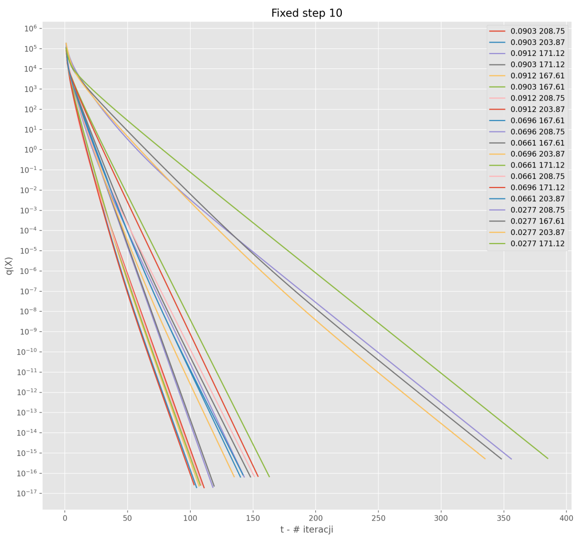


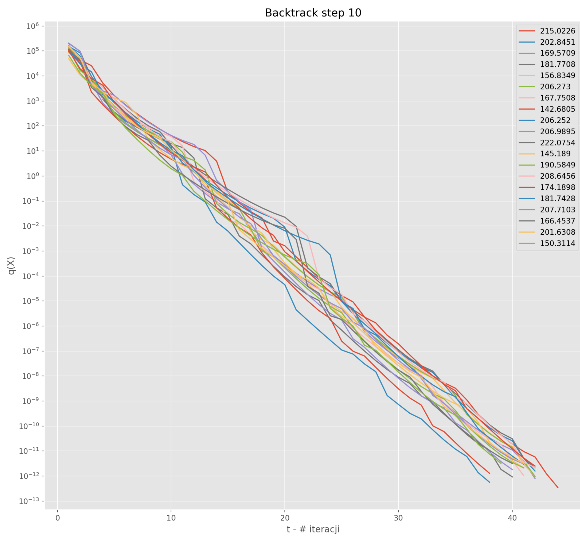


Czas

```
----------
Experiment: Random fixed step
Result: 7.99337139599254e-24
Execution time: 0.4417497599933995
----------
Experiment: Backtrack dynamic step
Result: 0.0
Execution time: 0.10568298600264825
```

α=10

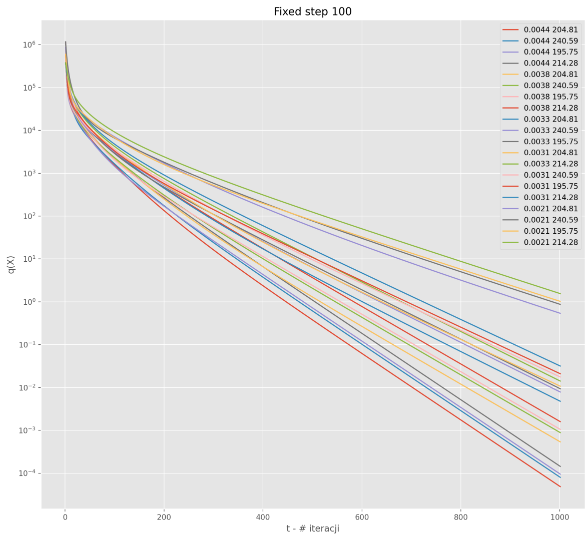


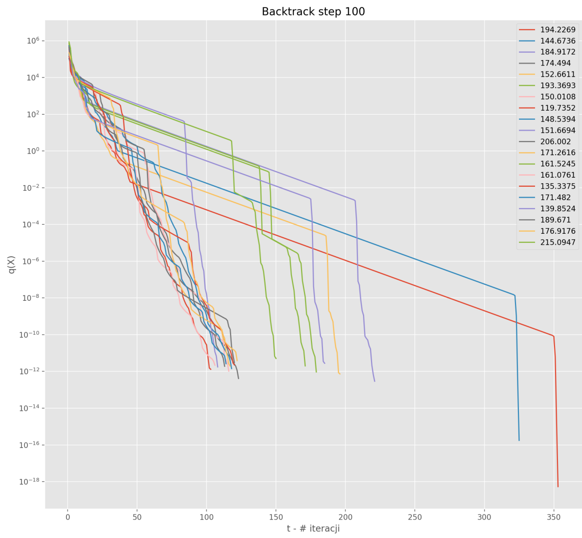


Czas

```
----------
Experiment: Random fixed step
Result: 1.9265017999456796e-17
Execution time: 1.7985471189967939
----------
Experiment: Backtrack dynamic step
Result: 3.496602571879976e-13
Execution time: 1.7700858829994104
```

α=100

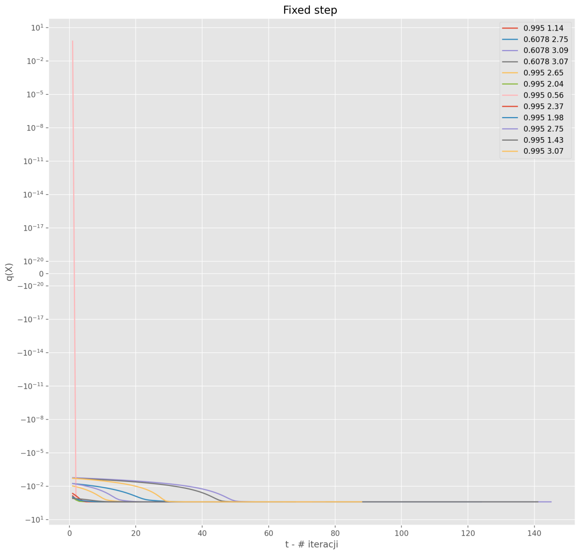


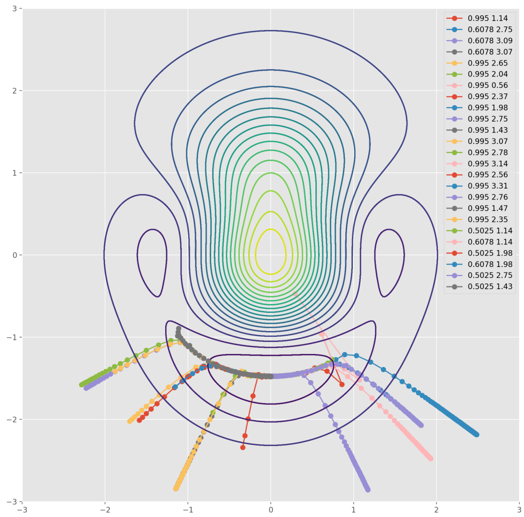


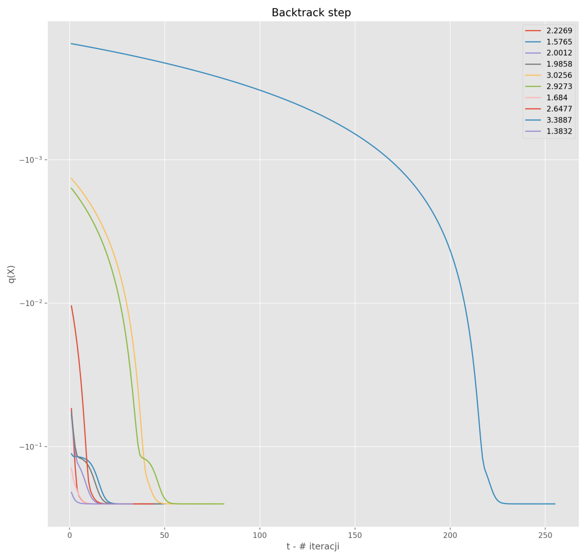
Czas

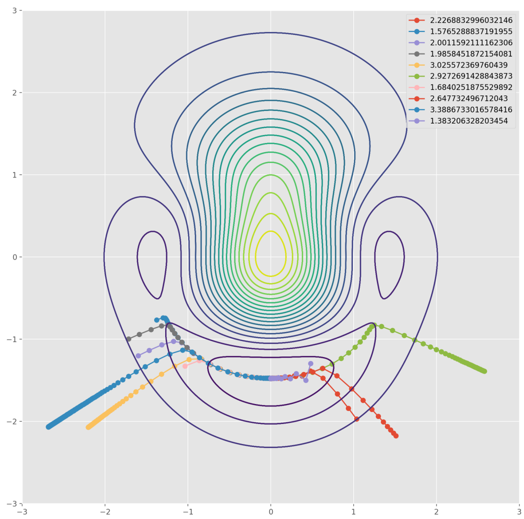
```
----------
Experiment: Random fixed step
Result: 4.863676495054151e-05
Execution time: 9.752461295000103
----------
Experiment: Backtrack dynamic step
Result: 5.361345025677837e-19
Execution time: 10.164777259997209
```

Wywołanie dla funkcji z wieloma punktami krytycznymi i "płaskim" obszarem









Czas

```
----------
Experiment: Random fixed step
Result: -0.2508093291388029
Execution time: 15.847806726997078
----------
Experiment: Backtrack dynamic step
Result: -0.25080932908725107
Execution time: 3.7925941110006534
```

Analiza

Szybkość zbieżności

Przy odpowiednim dobraniu kroku, oba algorytmy w przypadku testowej funkcji osiągają zadowalającą wartość. Stały krok wymaga jednak testowania wielu potencjalnych wartości, stwarza ryzyko wpadnięcia w oscylację, lub dążenia do nieskończoności. Z wykresów można wyciągnąć wniosek o dużym wpływie wartości kroku na szybkość zbieżności - wykres stanowią proste o różnych stopniach nachylenia.

Dokładność

Algorytm gradientu pozwala uzyskać dość dokładną zoptymalizowaną wartość - wartości gradientu zmniejszające się w pobliżu minimum pozwalają w tym obszarze dokładniej zbliżać się do punktu krytycznego - wpływ jednak ponownie wywiera krok, który może być za duży i wymusić na algorytmie ciągłe przeskakiwanie minimum.

Zachowanie w punktach krytycznych

Mankament gradientu prostego stanowi nieodpowiedni wybór punktu startowego - punkt krytyczny, lub "płaski" obszar, sprawiają, że gradient jest bliski zeru, powodując "stanie w miejscu" algorytmu. Próby zwiększania kroku nie przynoszą żadnych rezultatów.

Szybkość wykonania

Ponieważ gradient prosty polega głównie na odejmowaniu wektorów, jest on wydajnym rozwiązaniem. Możliwości równoległego przetwarzania wielu punktów i kroków dodatkowo zmniejszają czasy wykonania. Porównując oba algortymy, "backtracking" średnio wykonywał się szybciej, ponieważ miał możliwość w odpowiednich miejscach wykonywania dłuższego skoku.