WSI Laboratorium 1 Raport

Algorytm gradientu prostego

Miłosz Andryszczuk 331355

1. Wykorzystane biblioteki

Do przeprowadzenia eksperymentu zostały wykorzystane następujące biblioteki:

- matplotlib
  do rysowania wykresów
  https://matplotlib.org/stable/index.html
- numpy
  do obsługi operacji matematycznych
  https://numpy.org/doc/
- autograd
  do obliczania gradientu funkcji
  https://autograd.readthedocs.io/en/latest/

2. Opis eksperymentu

W ramach tego eksperymentu zbadano wpływ rozmiaru kroku oraz liczby iteracji na działanie algorytmu gradientu prostego przy różnych punktach początkowych.

Badania zostały przeprowadzone dla dwóch funkcji: jedno-argumentowej f(x) i dwu-argumentowej g(x1, x2).

Zaimplementowany algorytm gradientu prostego działa dla dowolnej funkcji o dowolnej liczbie argumentów.

3. Przebieg Eksperymentu

3.1. Funkcja f(x)

Dla funkcji f(x) przeprowadzono eksperymenty z trzema różnymi początkowymi wartościami:

- x0 = -4.0
- x0 = 2.0
- x0 = 7.0

Dla każdej wartości początkowej zastosowano różne rozmiary kroków:

- alpha = 0.005
- alpha = 0.01
- alpha = 0.1

Wszystkie przypadki zostały zbadane dla 20 i 200 iteracji.

3.2. Funkcja g(x1, x2)

Podobnie jak dla funkcji f(x), zbadano działanie algorytmu gradientu prostego dla funkcji dwuwymiarowej g(x1, x2) z trzema różnymi początkowymi wartościami:

- [x1, x2] = [-1.0, -3.0]
- [x1, x2] = [0.0, 0.0]
- [x1, x2] = [1.5, -3.0]

Zastosowano różne rozmiary kroków:

- alpha = 0.1
- alpha = 1.0
- alpha = 5.0

Liczba iteracji również wynosiła 20 oraz 200.

4. Wyniki

Wszystkie wyniki w postaci wykresów zapisanych w formacie png dostępne są w folderach **function_f** dla funkcji f i **function_g** dla funkcji g. W przypadku funkcji g zostały sporządzone po 2 wykresy dla każdej kombinacji parametrów: jeden 2D z poziomicami i drugi 3D.

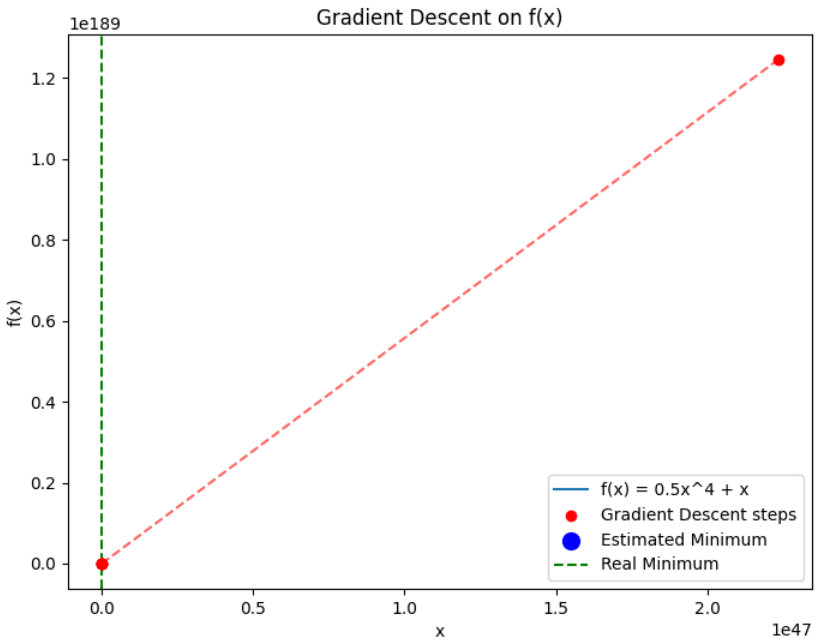
Możliwe jest uruchomienie algorytmu dla konkretnych parametrów i analiza wyniku na interaktywnym wykresie.

Poniżej przedstawiłem tylko kilka wyników, które pozwalają na określenie wpływu parametrów na dokładność przybliżenia ekstremum funkcji.

4.1. Funkcja f(x)

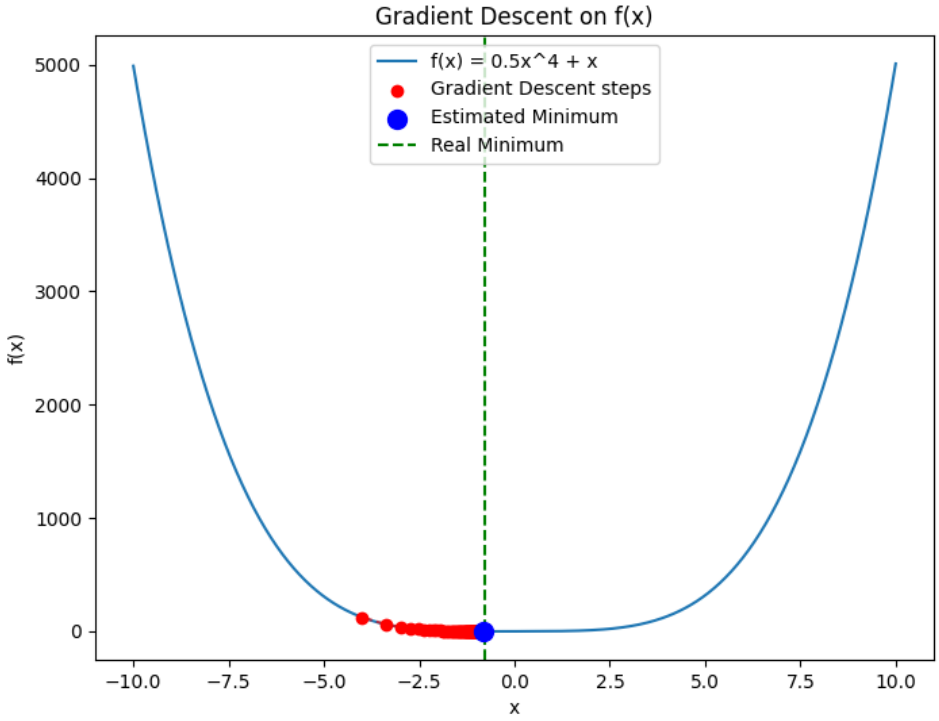
Dla wysokich wartości rozmiaru kroku (aplha), takich jak 0.1 kolejne kroki przeskakują minimum, co powoduje rozbieżność do nieskończoności niezależnie od liczby iteracji.

Alpha: 0.1 x0: -4.0 liczba iteracji: 20



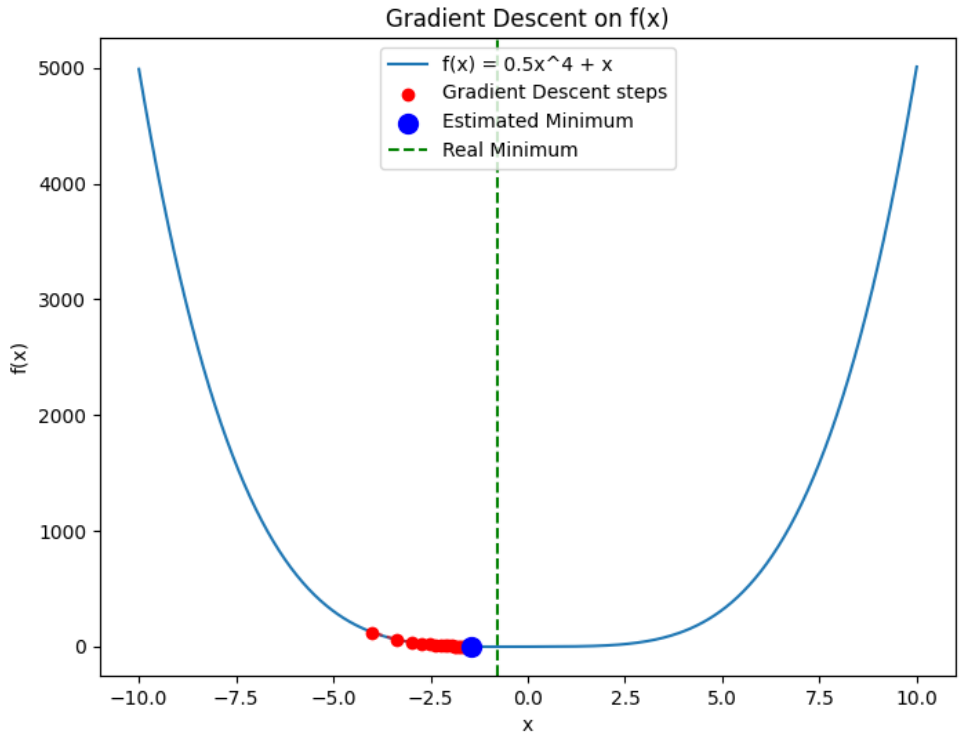
Dla mniejszych wartości alpha procedura zmierza do minimum w sposób stabilny ale powolny. Z tego powodu wymaga znacznie większej liczby iteracji aby dotrzeć do ekstremum. W poniższym przykładzie algorytm zdołał dotrzeć do minimum.

Alpha: 0.005 x0: -4.0 liczba iteracji: 200



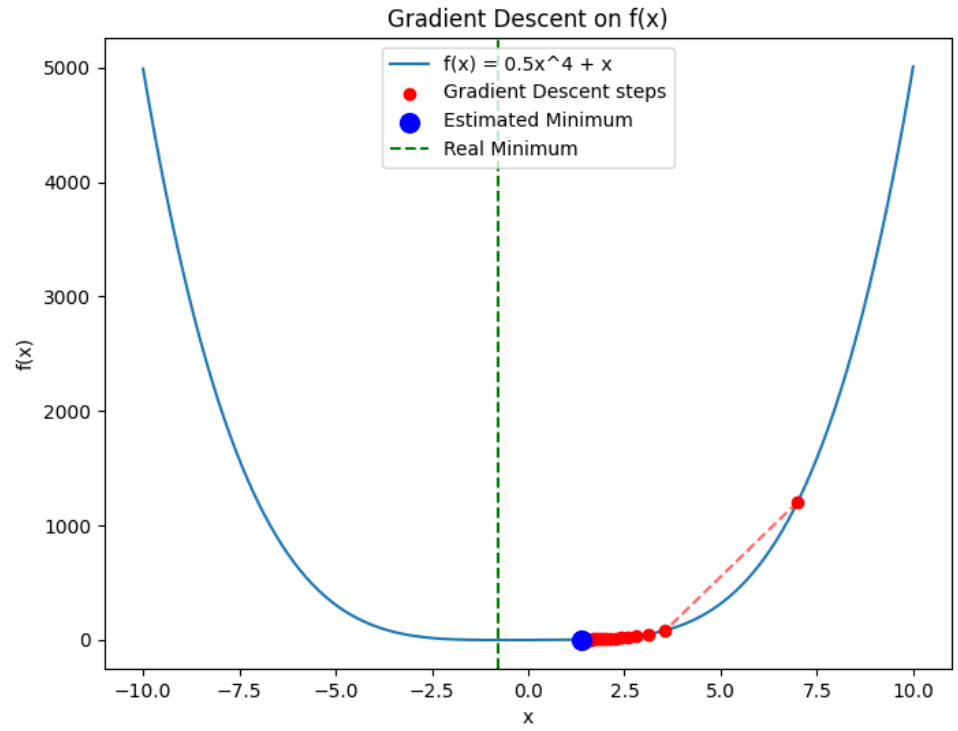
Natomiast dla mniejszej liczby iteracji procedura nie jest w stanie dotrzeć do faktycznego minimum (oznaczonego zieloną linią przerywaną).

Alpha: 0.005 x0: -4.0 liczba iteracji: 20



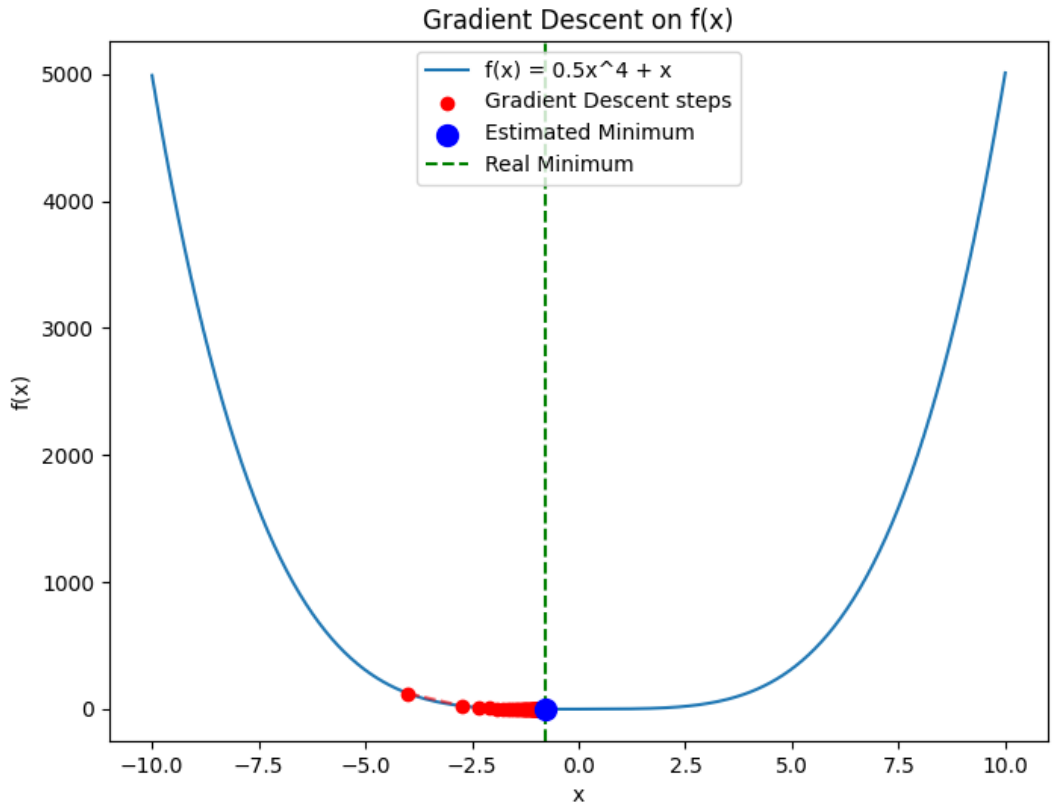
Im dalej od ekstremum znajduje się x0 tym większy jest ten błąd.

Alpha: 0.005 x0: 7.0 liczba iteracji: 20



Dla funkcji f(x) najlepszym doborem parametrów, który pozwala na niezawodne i dokładne znalezienie ekstremów jest duża liczba iteracji (200) i alpa wynoszące 0.01.

Alpha: 0.01 x0: -4.0 liczba iteracji: 200

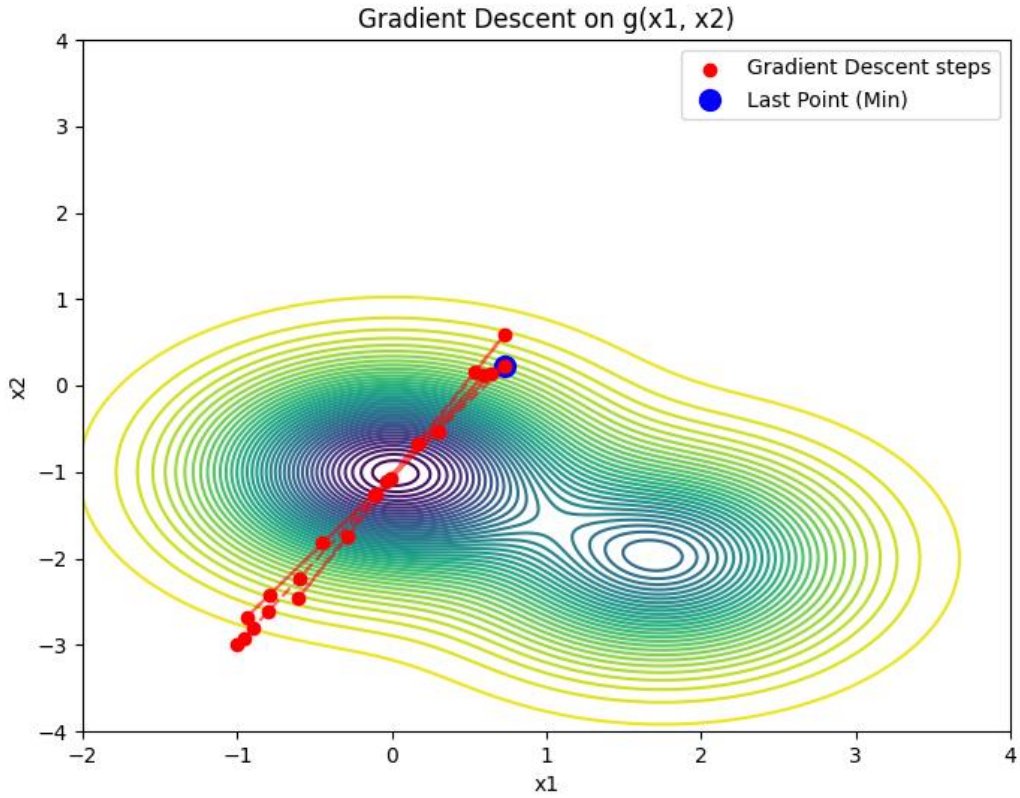


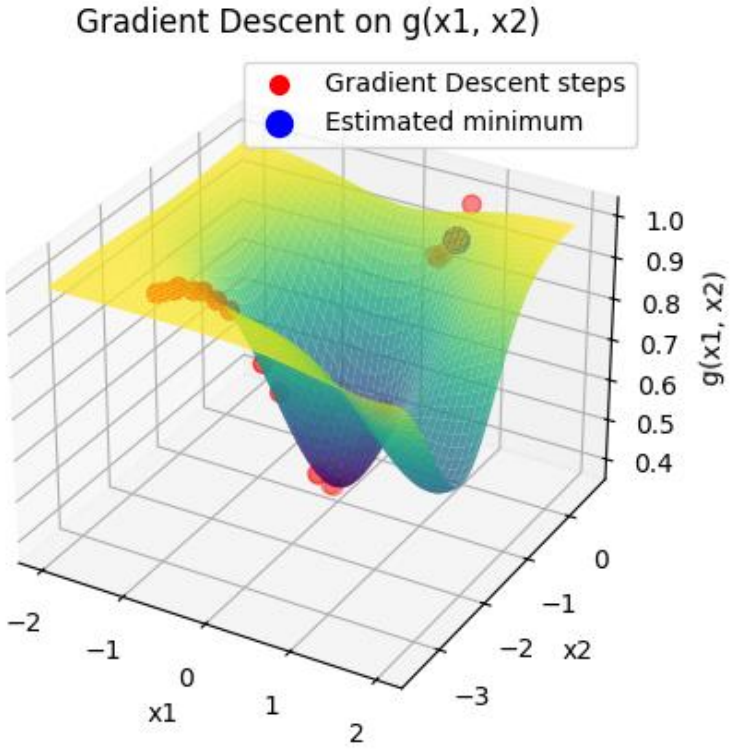
4.2. Funkcja g(x1, x2)

Wynik eksperymentu dla funkcji g(x1, x2) przyniósł podobne rezultaty jak w przypadku funkcji f(x).

Dla bardzo wysokich rozmiarów skoku procedura oscylowała wokół ekstremum funkcji i ostatecznie zakończyła się daleko od faktycznego ekstremum.

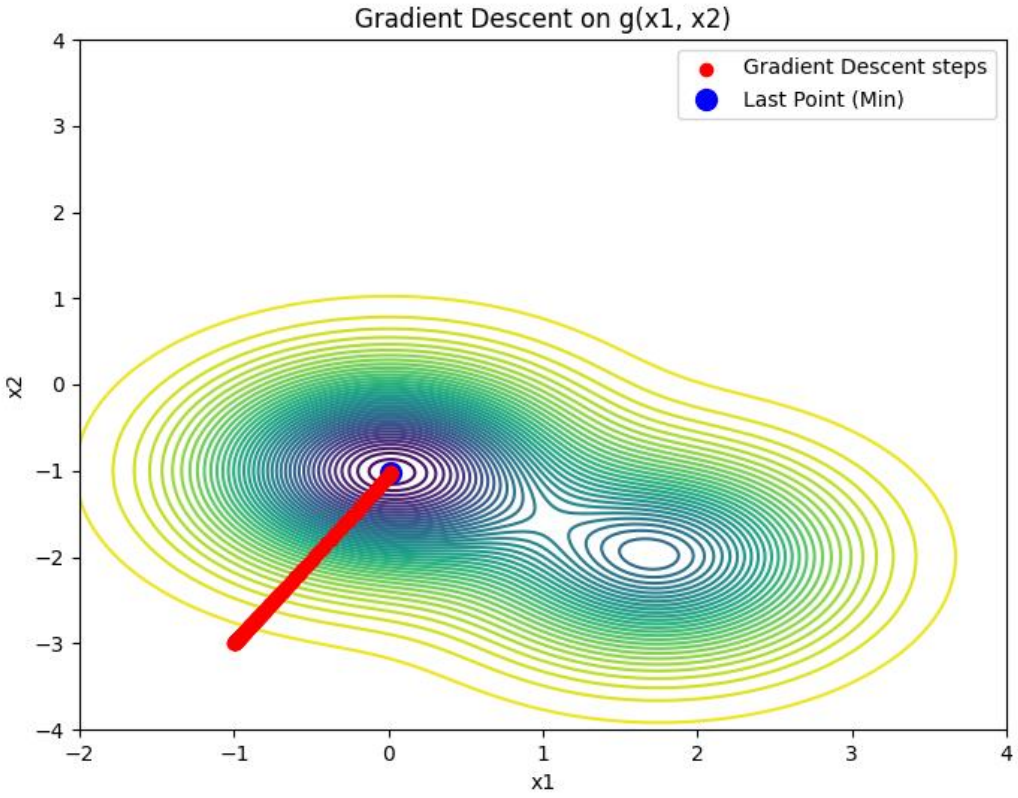
Alpha: 5 x0: [-1, -3] liczba iteracji: 20

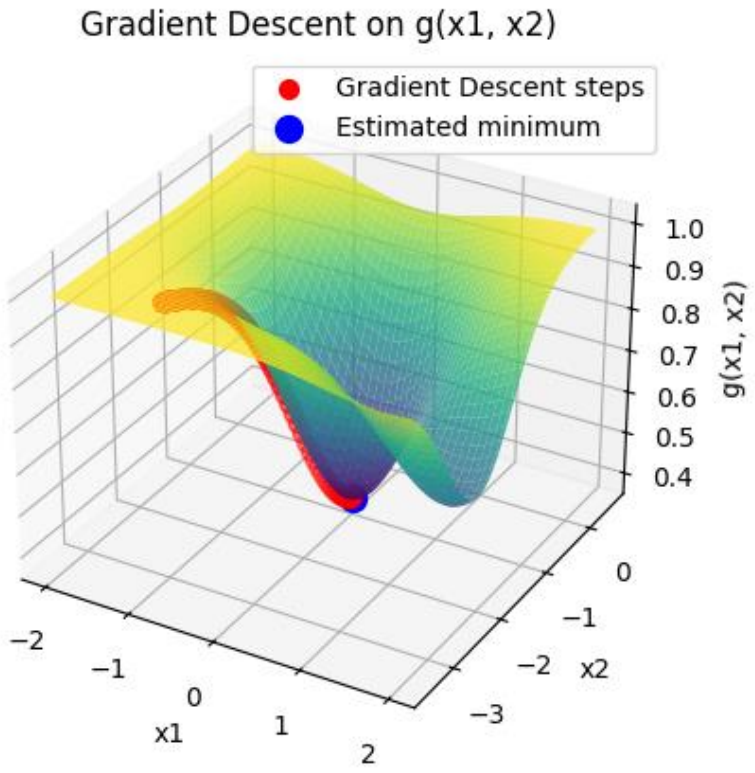




Mały współczynnik alpha sprawia, że algorytm bardzo dokładnie wylicza ekstremum funkcji ale tylko, jeśli wykona odpowiednio dużo skoków.

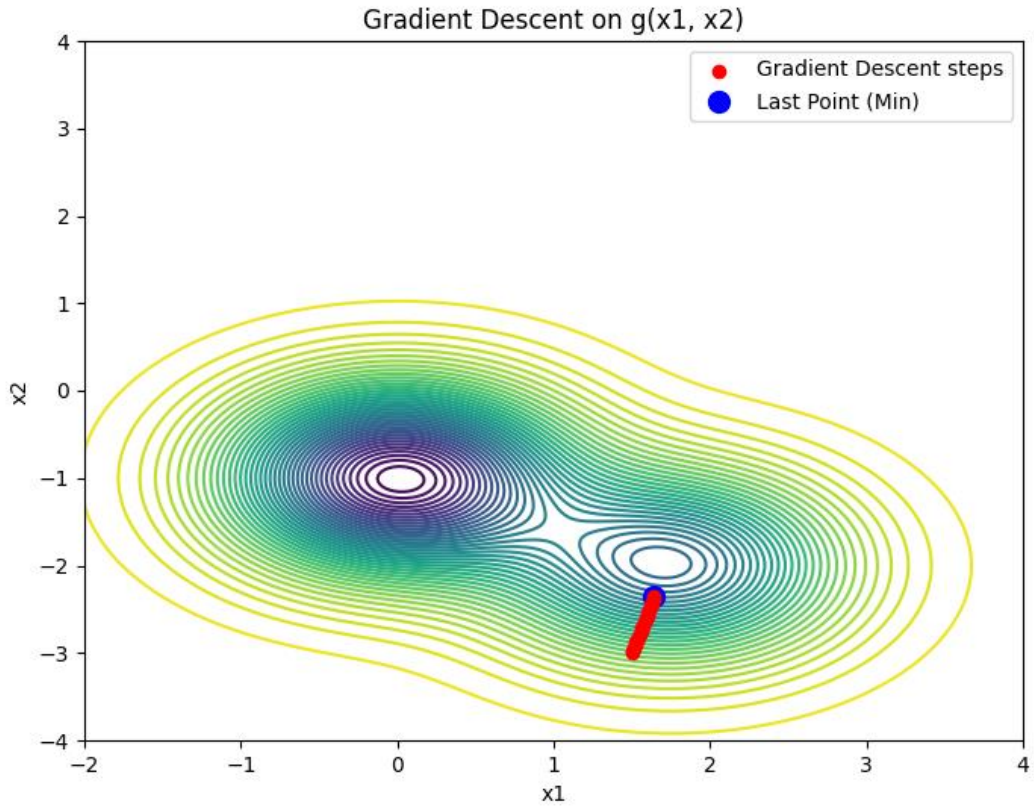
Alpha: 0.1 x0: [-1, -3] liczba iteracji: 200

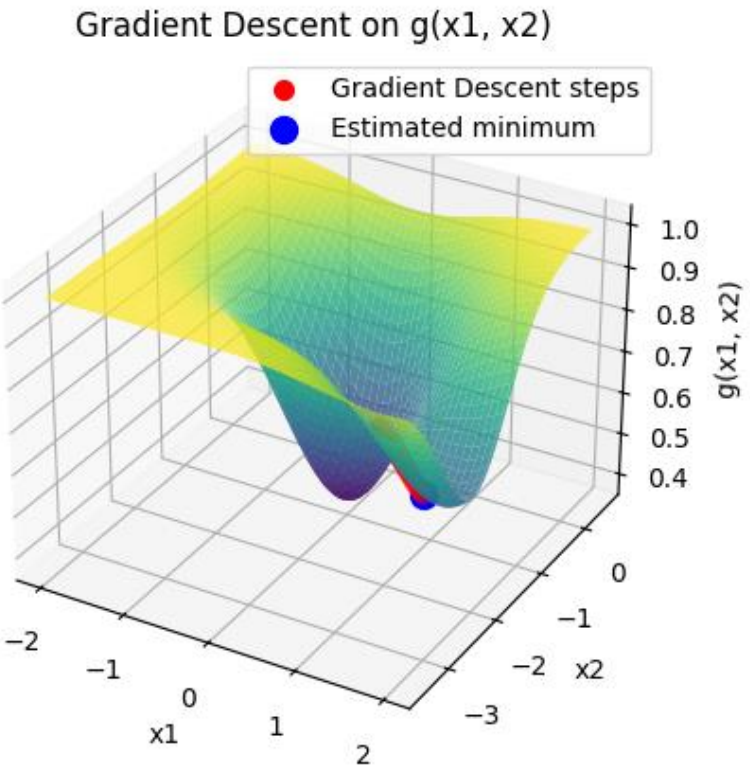




Jednakże dla małej liczby skoków algorytm nie jest w stanie dotrzeć do faktycznego minimum lokalnego.

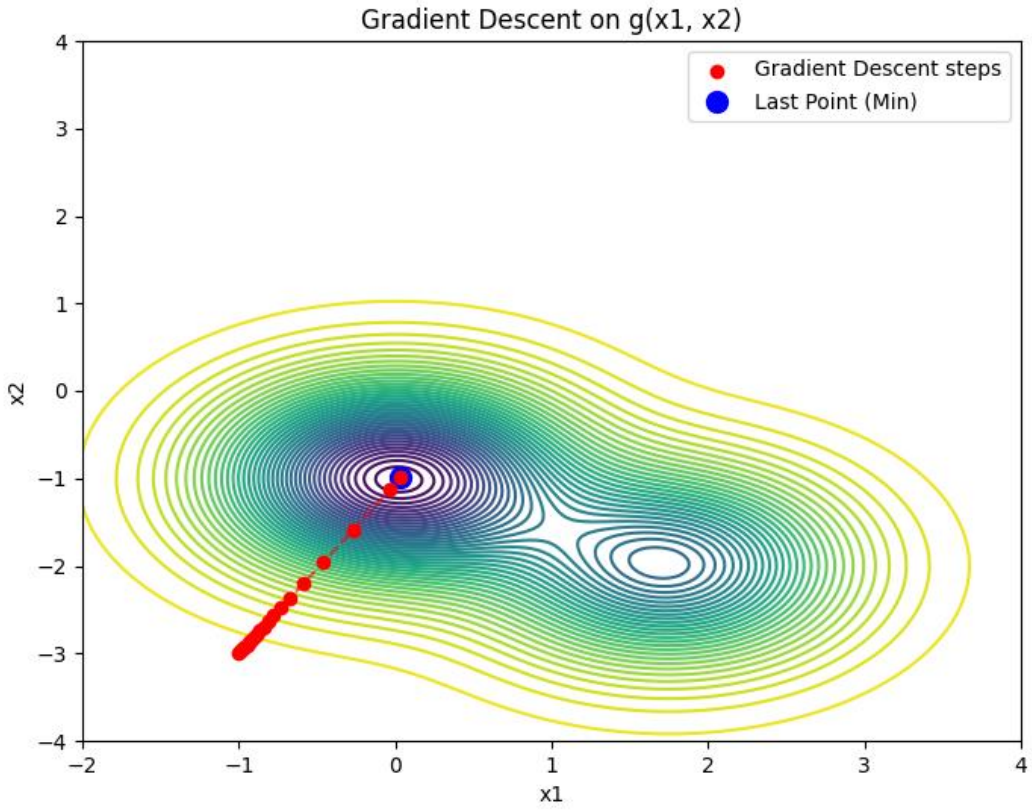
Alpha: 0.1 x0: [1.5, -3] liczba iteracji: 20

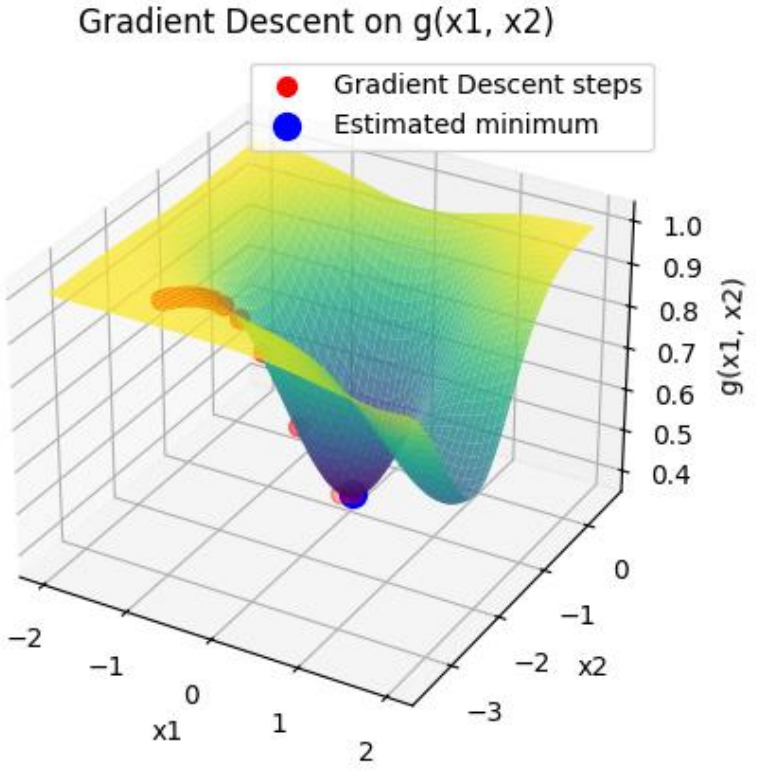




Dla funkcji g(x1, x2) najlepiej zdaje się działać współczynnik alpa równy 1, który nawet przy małej liczbie iteracji (20) potrafi znaleźć ekstremum z bardzo dobrym przybliżeniem.

Alpha: 1 x0: [-1, -3] liczba iteracji: 20





5. Wnioski

Eksperyment wykazał, że algorytm gradientu prostego jest bardzo wrażliwy na rozmiar kroku. Zbyt mały krok spowalnia cały proces i wymaga znacznie więcej iteracji, a tym samym czasu na znalezienie ekstremum. Natomiast zbyt duży krok powoduje oscylacje wokół minimum lub eksplozję (rozbieżność) do nieskończoności.

Jeśli zależy nam na dokładności otrzymanego wyniku, najlepszym rozwiązaniem jest wykonanie dużej liczby kroków o małym rozmiarze, jednak jeśli zależy nam na czasie to trzeba się liczyć z faktem, że zwiększanie tego rozmiaru wiążę się z ryzkiem błędnego wyniku algorytmu.