

# Raport LAB 1 – Jakub Kwaśniak 331396

## 1. Treść polecenia

Znana jest funkcja celu:

- $f(x)=Ax+B\sin(x)$   
 $x \in (-4\pi, 4\pi)$
- $g(x,y)=Cxy/(e^{(x^2+y^2)})$   
 $x \in (-2, 2)$   
 $y \in (-2, 2)$

Gdzie A, B, C to kolejne dodatnie cyfry numeru indeksu poczynając od jego końca. Np. dla 321037 A=7, B=3, C=1

- W moim rozwiązaniu A=6, B=9, C=3

Zaimplementować metodę gradientu prostego opisaną na wykładzie.  
Użyć zaimplementowany algorytm do wyznaczenia ekstremów funkcji.

Zbadać wpływ następujących parametrów na proces optymalizacji:

- długość kroku uczącego
- limit maksymalnej liczby kroków algorytmu
- rozmieszczenie punktu startowego

Zinterpretować wyniki w kontekście kształtu badanej funkcji.

## **2. Cel i opis eksperymentu:**

W tym zadaniu badana jest skuteczność metody gradientu prostego dla wyznaczania ekstremów funkcji (optymalizacji funkcji kosztu/straty)

Wyniki zostały zbadane ze względu na 3 parametry:

- długość kroku uczącego:
  - długość kroku uczącego ‘alpha’ mówi nam o rzędzie wielkości o jaki zmienione zostaną badane argumentu w 1 kroku optymalizacji funkcji
- limit maksymalnej liczby kroków algorytmu:
  - liczba kroków uczących algorytmu, czyli obrotów pętli – wpływa na to ile optymalizacji argumentów funkcji zostanie przeprowadzone w celu znalezienia ekstremów, mamy na celu zbadać jaka liczba kroków pozwoli uzyskać wynik bliski dokładnemu
- rozmieszczenie punktu startowego:
  - Punkt startowy dla funkcji każdorazowo wybierany był losowo z podanych w treści zadania przedziałów.

Jakość działania algorytmu dla dobranych parametrów można określić na podstawie wykresów z zaznaczoną ‘trajektorią’ znajdowania ekstremów oraz na podstawie porównania wyników z wynikami obliczonymi ręcznie bądź za pomocą dokładnych narzędzi matematycznych takich jak ‘wolfram alpha’.

### **3. Odtworzenie wyników:**

- Aktywacja środowiska wirtualnego oraz instalacja potrzebnych bibliotek:

```
python3 -m venv venv  
source venv/bin/activate  
pip install -r requirements.txt
```

- Uruchomienie skryptu:

```
python3 ..//WSI-JAKUB-KWASNIAK/lab1/lab1_Jakub_Kwasniak.py  
--function ['f'/'g']
```

- Argument wywołania ‘function’ przyjmuje tylko 2 możliwe wartości – f lub g (w zależności od funkcji, którą chcemy optymalizować), wartość domyślna to ‘f’

- Analiza wyników:

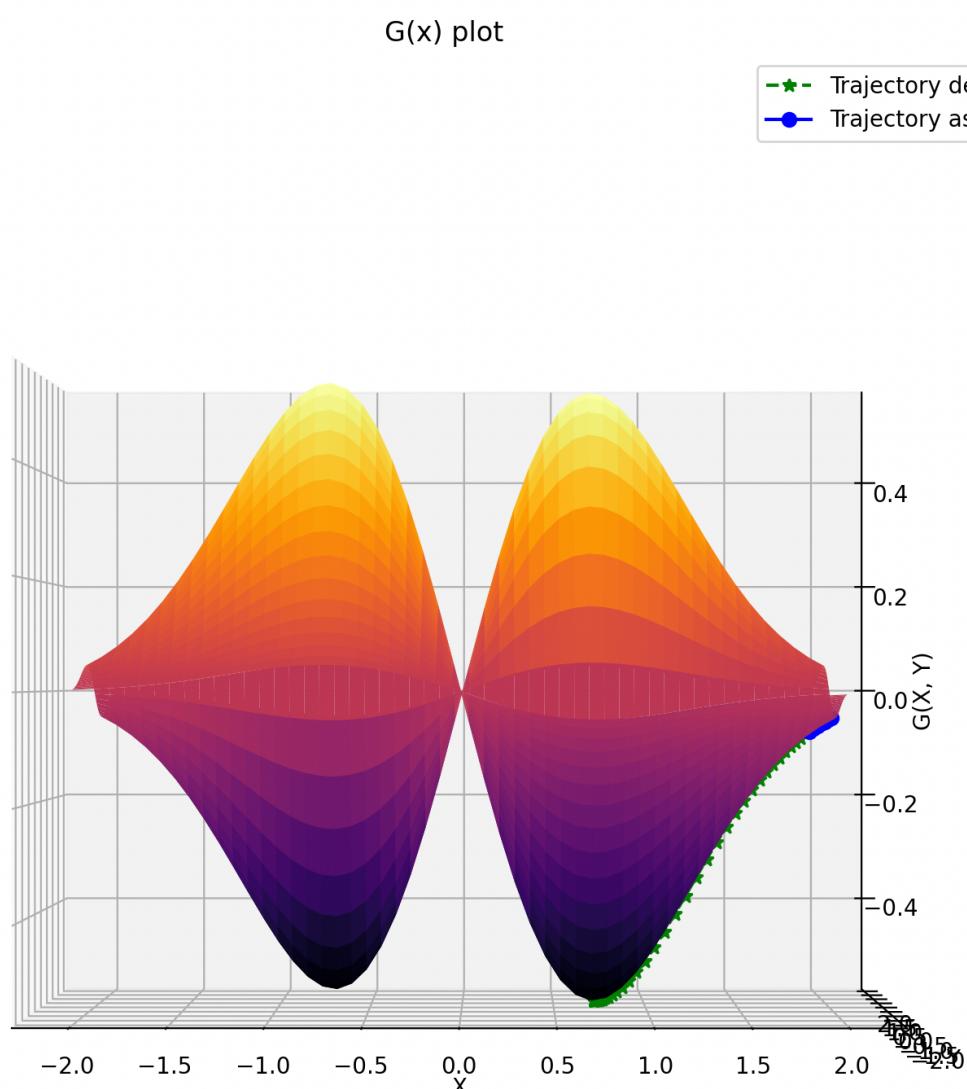
- W terminalu wyświetcone zostaną wylosowane argumenty początkowe funkcji oraz wyznaczone metodą gradientu prostego ekstrema w notacji ([współrzędne], ekstremum)

#### 4. Wyniki przeprowadzonego eksperymentu:

- Przykład 1:
  - Długość kroku uczącego ‘alpha’: 0.1
  - Limit maksymalnej liczby kroków algorytmu: 10 000

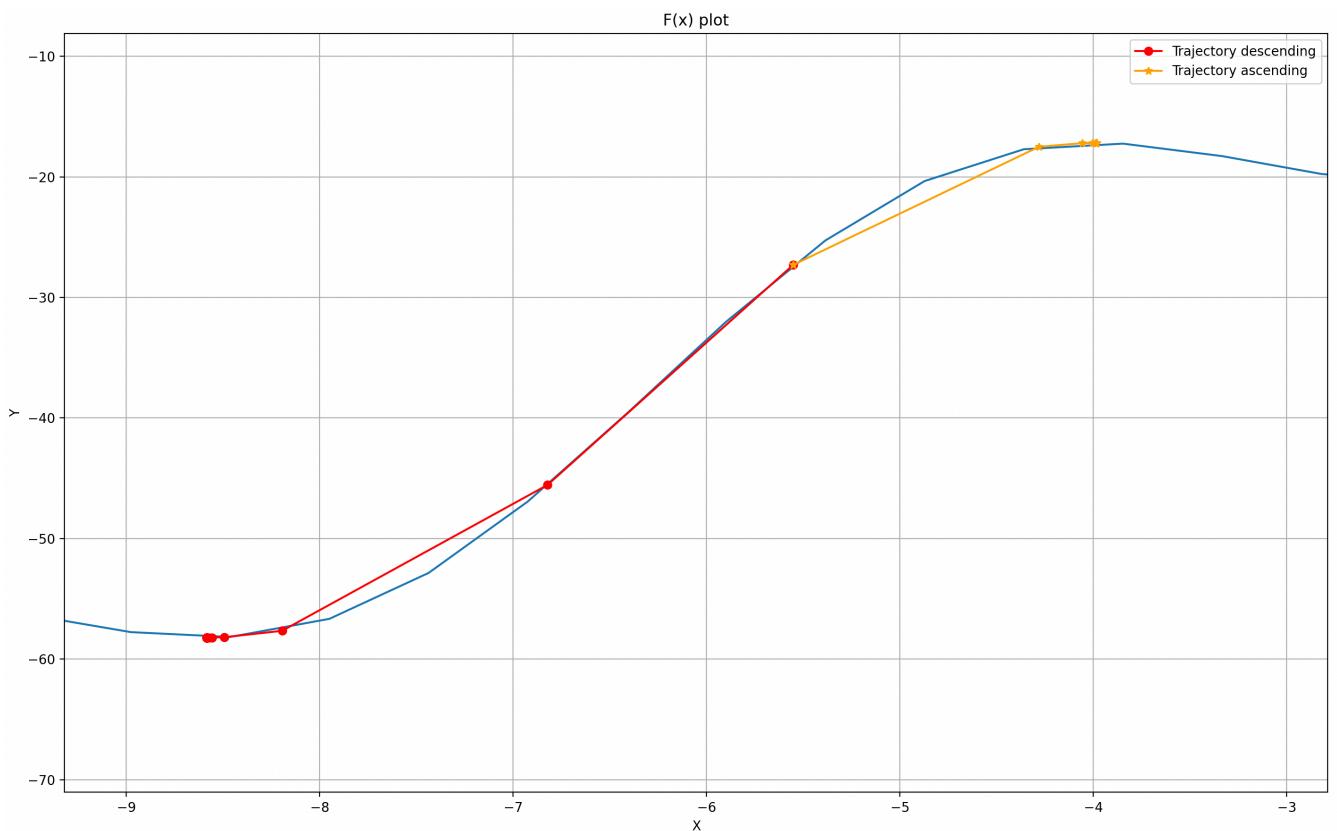
Funckja  $G(x, y)$ :

```
Starting point: [1.8408411644654512, -0.5772178114746165]
Function g has minimum ([ 0.70710678 -0.70710678], -0.5518191617571636)
Function g has maximum ([ 1.98390852 -0.5494406 ], -0.04721935420016936)
```



Funkcja F(x):

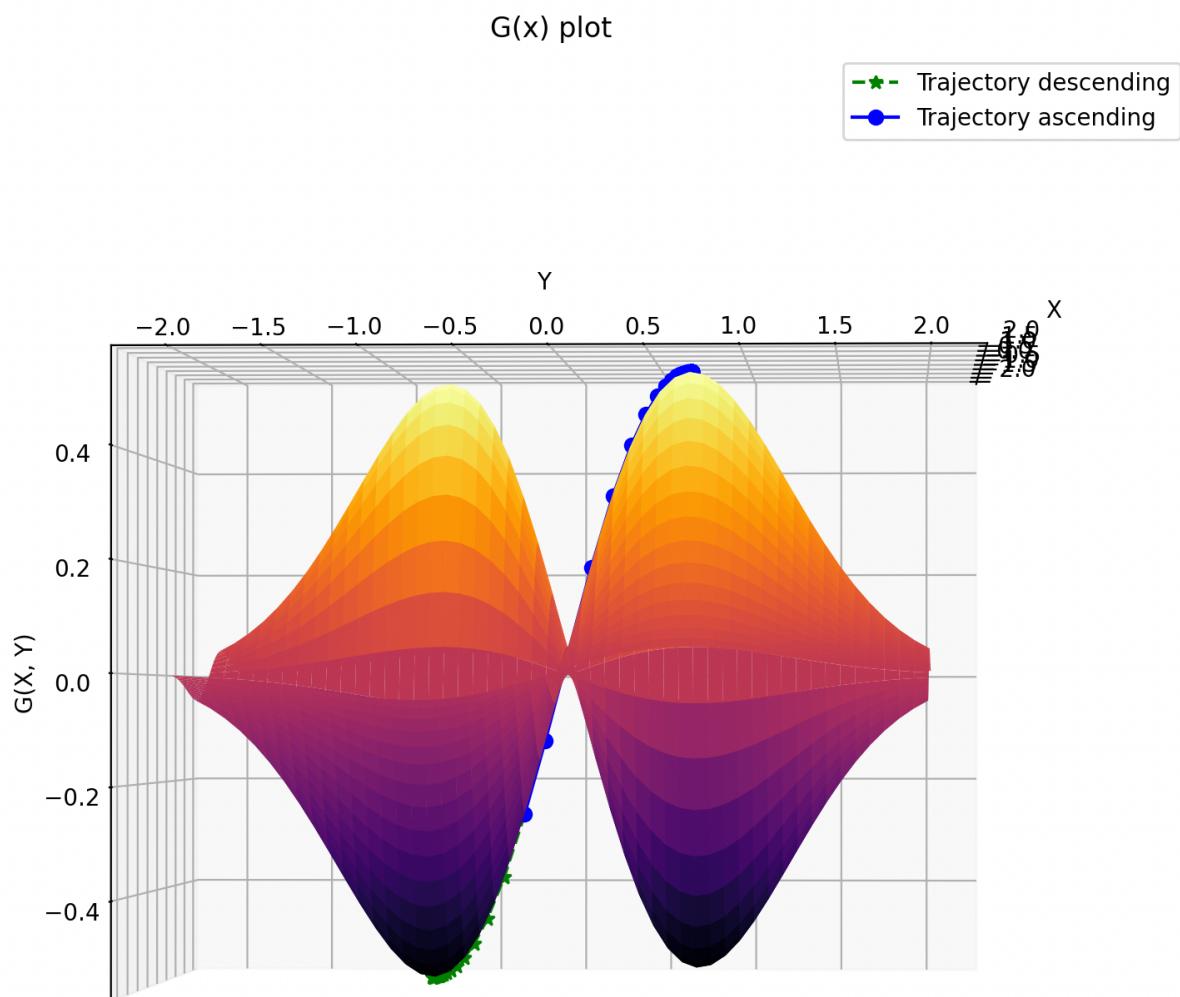
```
Starting point: [-5.552069732711495]
Function f has minimum([-8.58370929], [-58.21045967])
Function f has maximum([-3.98266132], [-17.18776401])
```



- Przykład 2:
  - Długość kroku uczącego 'alpha': 0.1
  - Limit maksymalnej liczby kroków algorytmu: 1 000

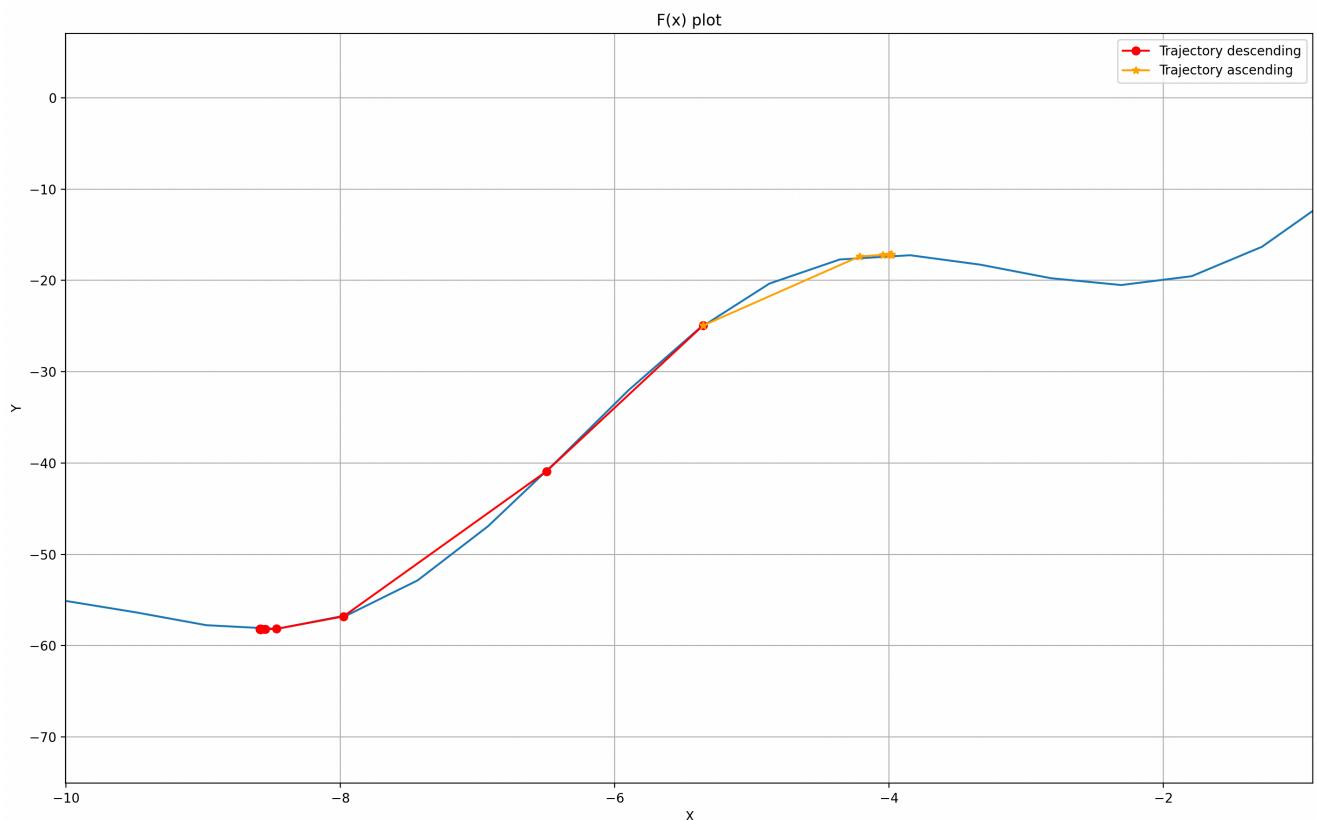
Funckja  $G(x, y)$ :

```
Starting point: [0.6578833227900365, -0.20775847638702127]
Function g has minimum ([ 0.70710678 -0.70710678], -0.5518191617571634)
Function g has maximum ([0.70710678 0.70710678], 0.5518191617571634)
```



Funkcja F(x):

```
Starting point: [-5.357276874424164]
Function f has minimum([-8.58370929], [-58.21045967])
Function f has maximum([-3.98266132], [-17.18776401])
```



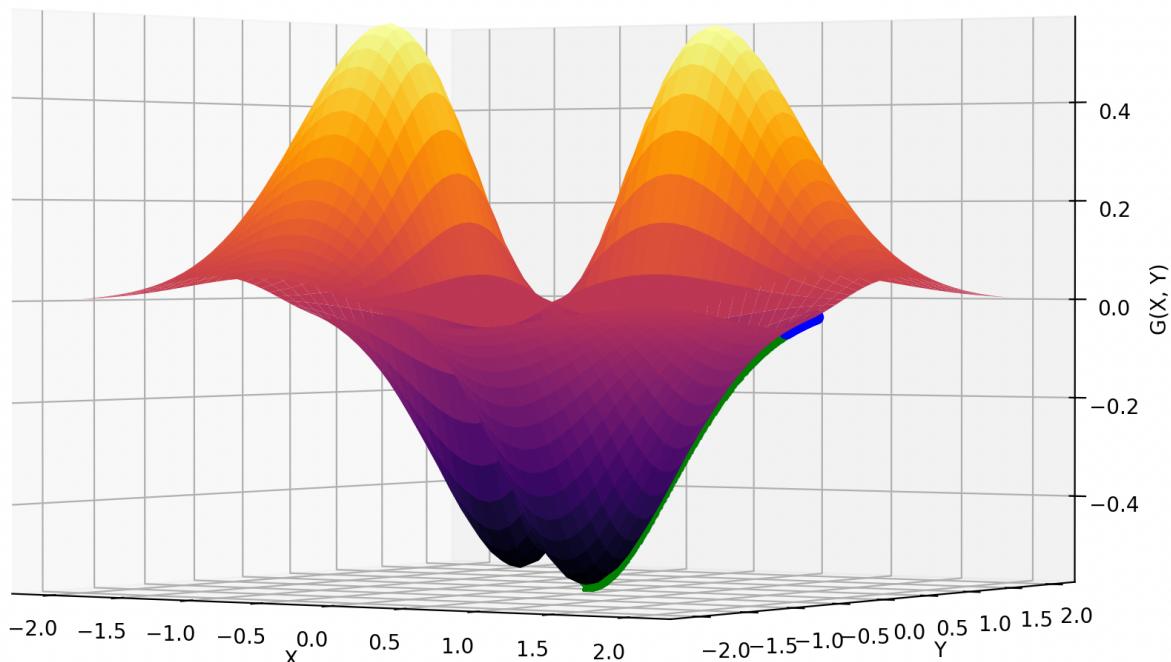
- Przykład 3:
  - Długość kroku uczącego ‘alpha’: 0.01
  - Limit maksymalnej liczby kroków algorytmu: 10 000

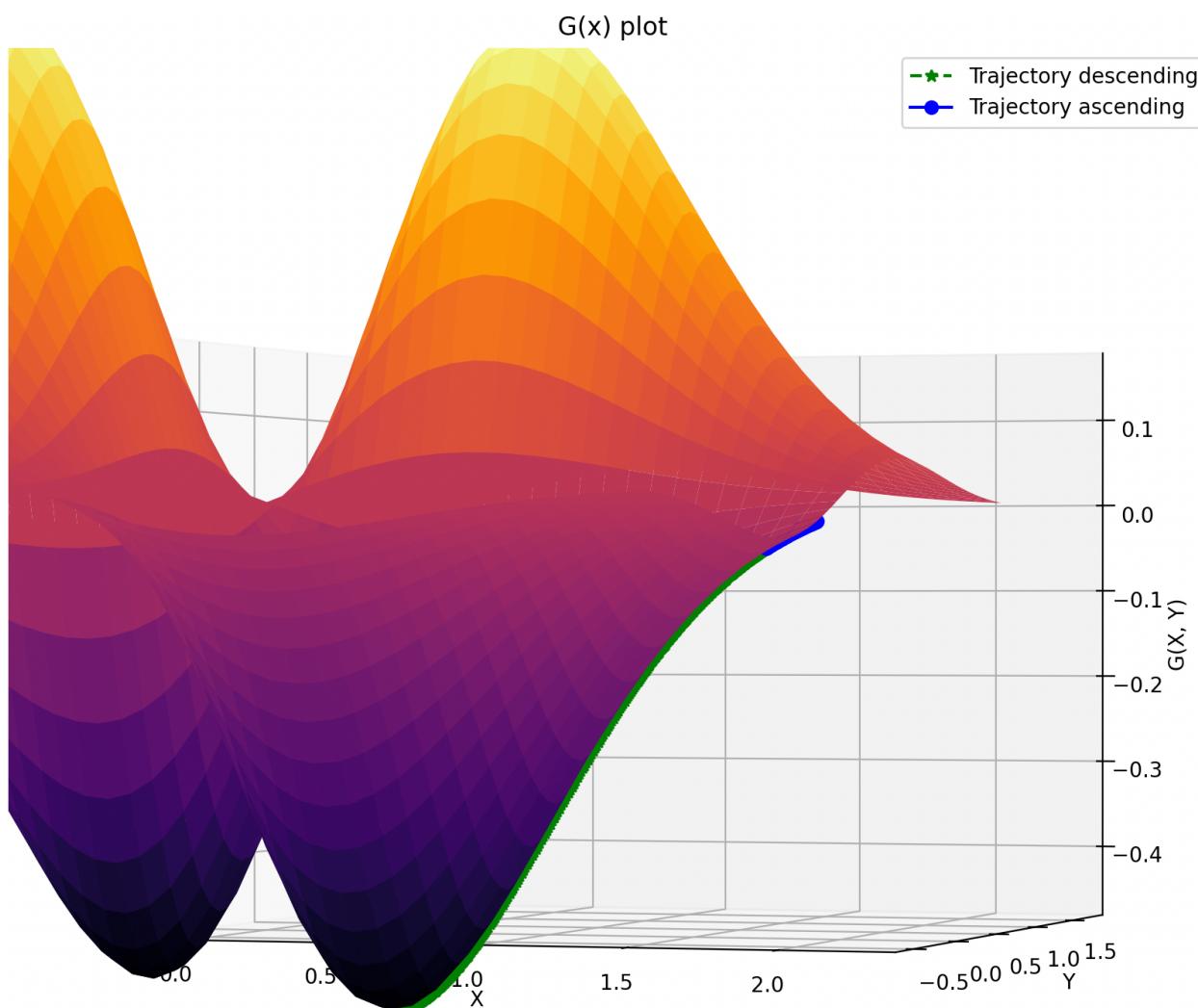
Funckja  $G(x, y)$ :

Starting point: [1.8612201826568189, -0.36897521385380294]  
 Function g has minimum ([ 0.70710678 -0.70710678], -0.5518191617571636)  
 Function g has maximum ([ 1.99985528 -0.26565282], -0.027218171330163594)

$G(x)$  plot

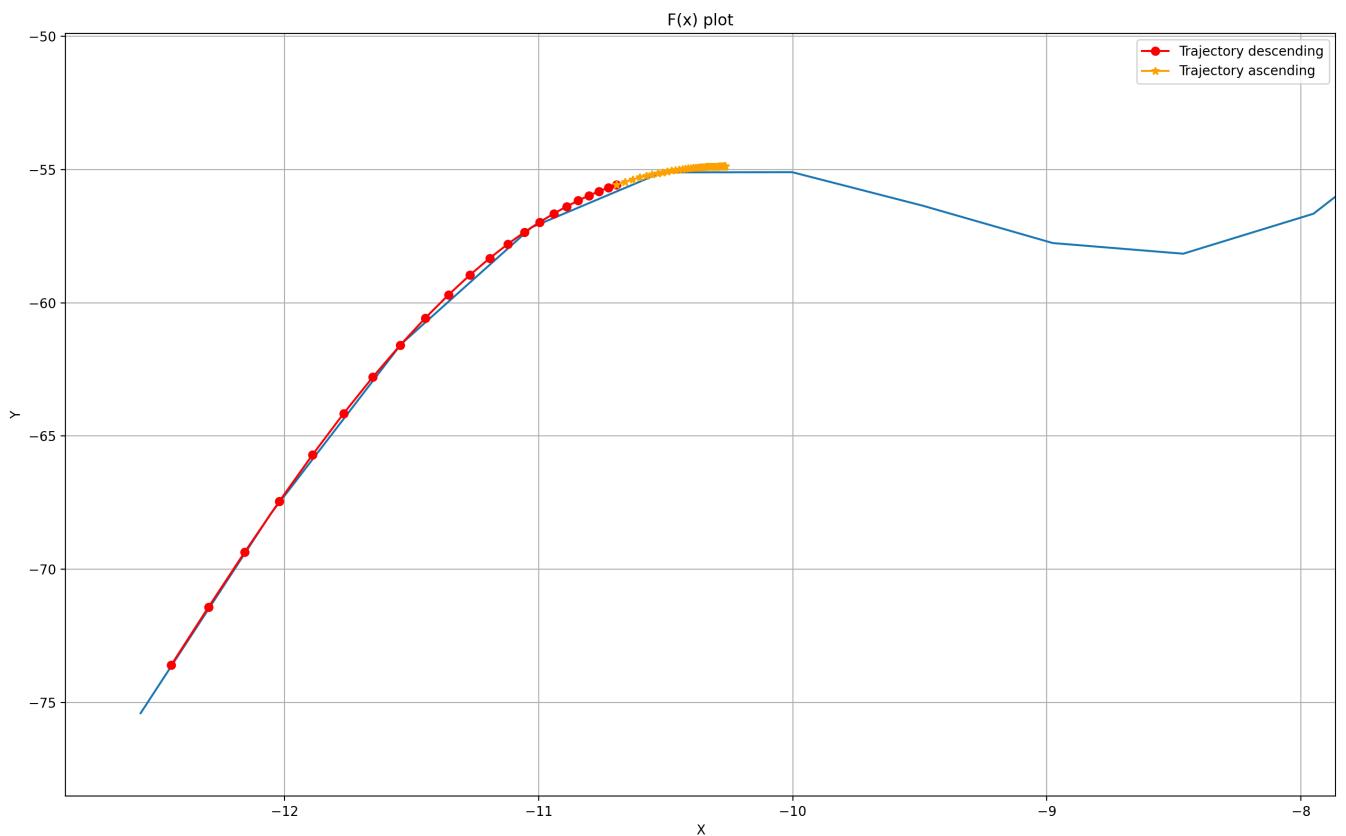
-★-	Trajectory descending
-●-	Trajectory ascending





Funkcja F(x):

Starting point: [-10.692754782346336]  
Function f has minimum ([ -12.44625416], [ -73.59907457])  
Function f has maximum ([ -10.26584663], [ -54.88687586])



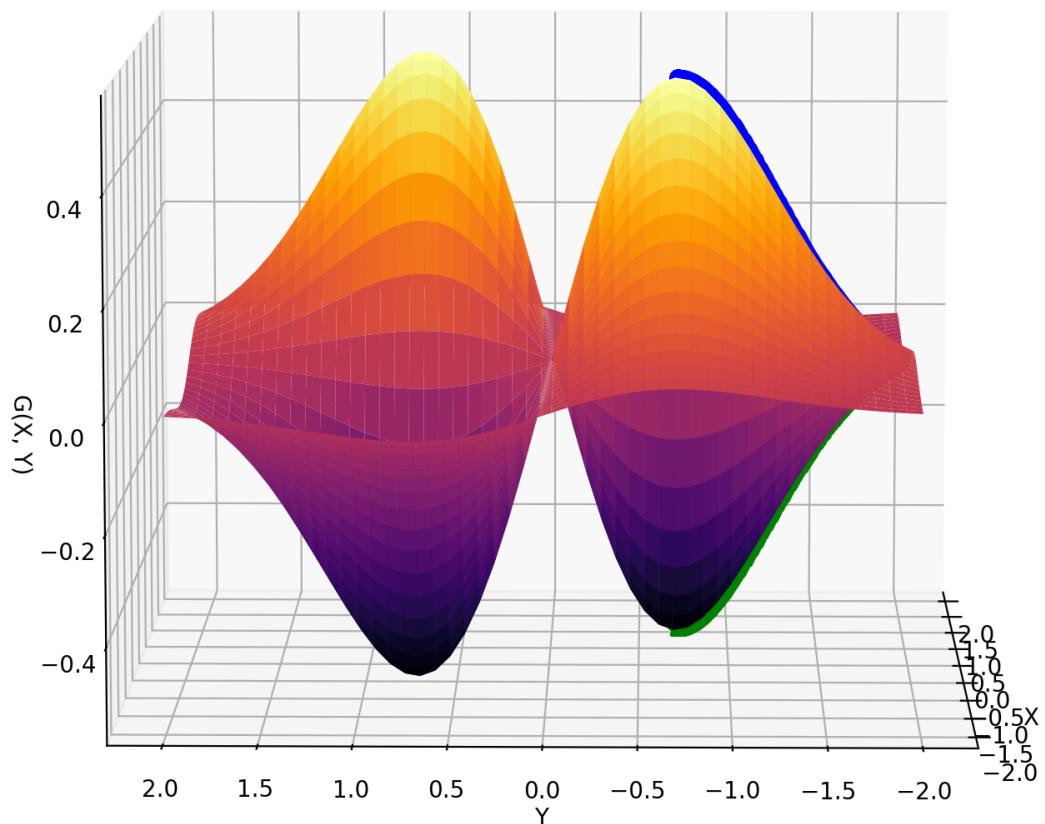
- Przykład 4:
  - Długość kroku uczącego ‘alpha’: 0.01
  - Limit maksymalnej liczby kroków algorytmu: 1 000

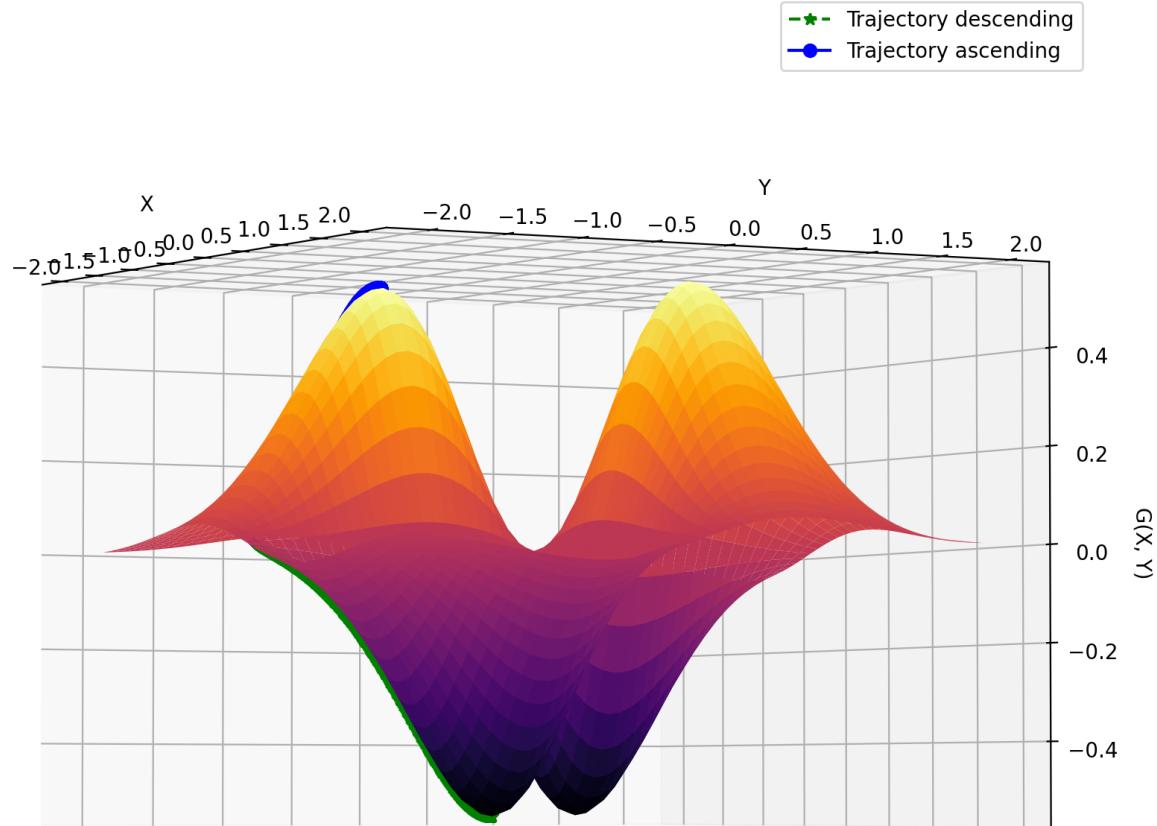
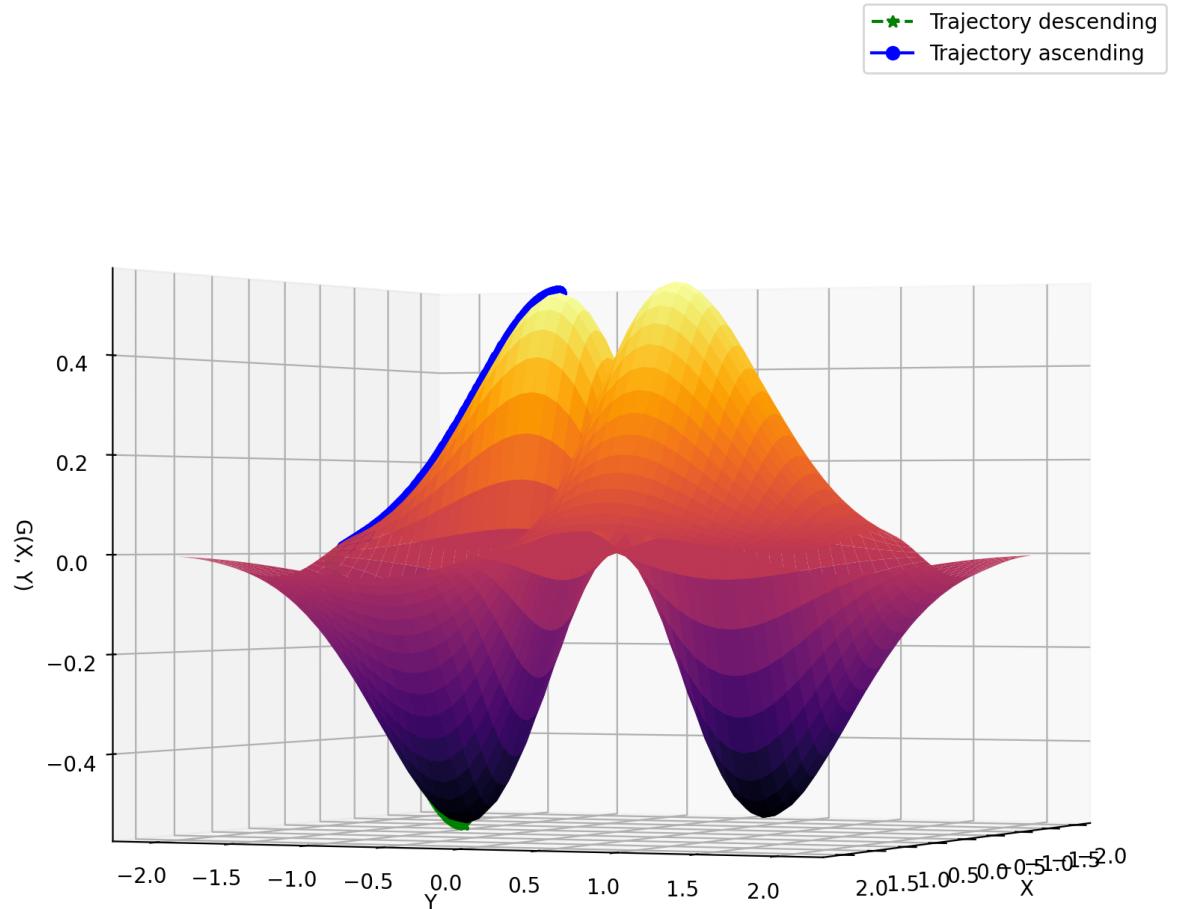
Funckja  $G(x, y)$ :

Starting point: [-0.07560943862421388, -1.9183131824109867]  
 Function g has minimum ([ 0.70710654 -0.70710838], -0.5518191617542678)  
 Function g has maximum ([ -0.70710676 -0.70710693], 0.5518191617571395)

$G(x)$  plot

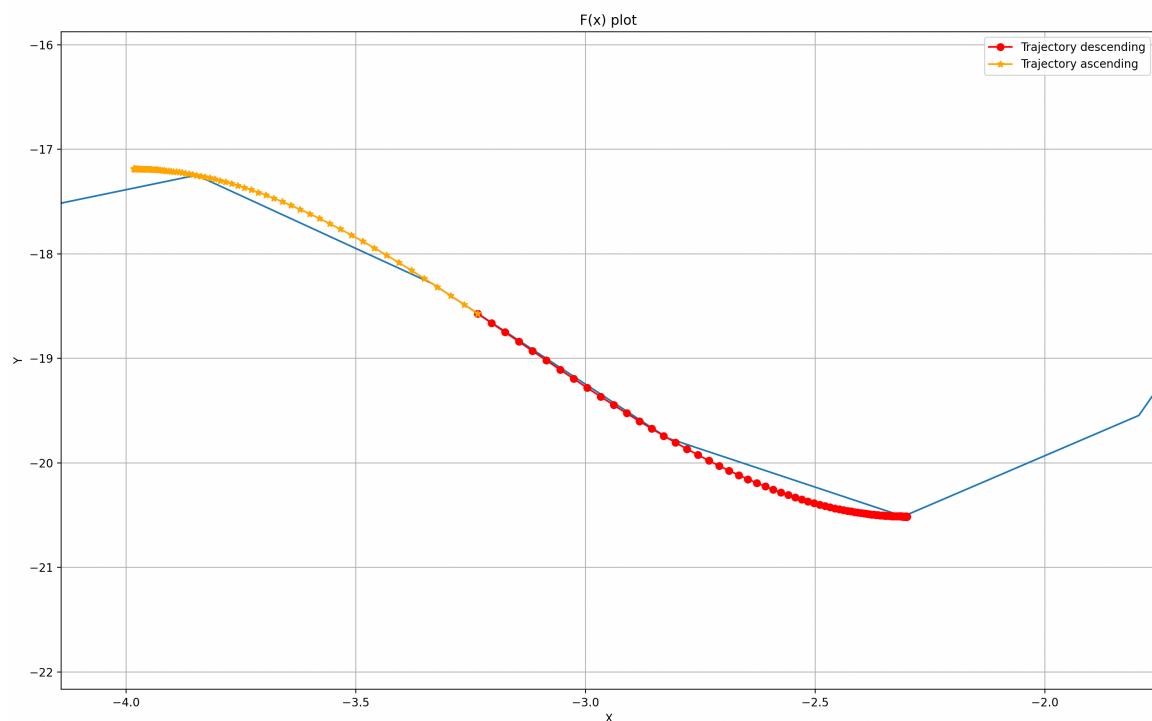
-★-	Trajectory descending
-●-	Trajectory ascending





Funkcja F(x):

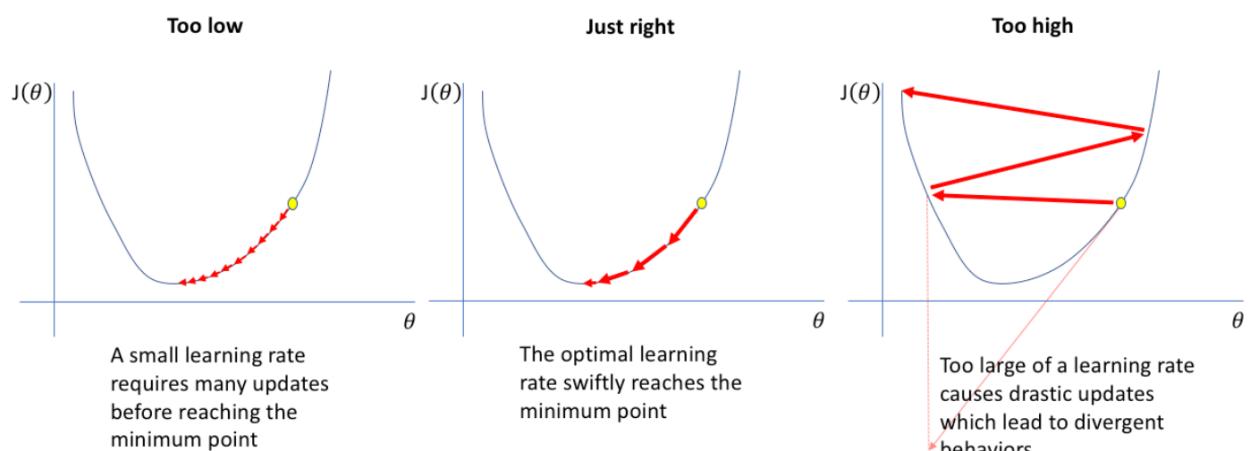
```
Starting point: [-3.2345250340891933]
Function f has minimum([-2.30052398], [-20.51134783])
Function f has maximum([-3.98266132], [-17.18776401])
```



## 5. Omówienie wyników eksperymentu - wnioski:

- Długość kroku uczącego:
  - Optymalizując funkcję należy zbadać wiele różnych współczynników uczenia 'alpha' i dobrać najlepszy do badanej funkcji – dla każdej funkcji optymalna wartość współczynnika będzie inna.
  - Zbyt długi krok uczący spowoduje duże odchylenia między kolejnymi punktami wyznaczanej trajektorii poszukiwania ekstremów co może spowodować pominięcie ekstrema i wyznaczenie błędного wyniku, z kolei zbyt krótki krok uczący spowoduje znaczący wzrost czasu wykonywania programu oraz będzie wymagał wykonania większej ilości iteracji w celu wyznaczenia ekstremów.

## Współczynnik uczenia (learning rate) $\alpha$



Źródło: „Regresja” Bootcamp Golem 2023, autorstwo: Weronika Piotrowska

- Limit maksymalnej liczby kroków algorytmu:
  - Za mała ilość iteracji/kroków algorytmu może spowodować uzyskanie błędnie wyznaczonego ekstremum - algorytm nie dojdzie do lokalnego bądź globalnego ekstremum a zatrzyma się w 'trakcie drogi'.
  - Za duża ilość iteracji/kroków algorytmu może spowodować wzrost czasu wykonywania algorytmu oraz marnowanie mocy obliczeniowej - gdy ekstremum zostanie aproksymowane z dużą dokładnością -> gradient będzie bliski zeru, a więc kolejne iteracje spowodują nieznaczące zmiany w wyznaczonym ekstremum (wartość będzie oscylowała wokół już wyznaczonej).

- Rozmieszczenie punktu startowego:
  - Rozmieszczenie wylosowanego punktu startowego wpływa na długość wyznaczonej trajektorii dojścia algorytmu do ekstremum, czyli na ilość operacji potrzebną do wyznaczenia ekstremów
  - Rozmieszczenie wylosowanego punktu startowego wpływa również na to jakie ekstrema zostaną wyznaczone – lokalne czy globalne