Kacper Tomczykowski  
Nr albumu: 318736

**Minimalizacja funkcji celu i porównanie wyników dla metody najszybszego spadku gradientu i metody Newtona**

# Wersja języka:

* Python - 3.9.7

# Używane biblioteki:

* Numpy - 1.22.3
* Matplotlib - 3.6.1
* Random
* Timeit

# Funkcją celu:

Funkcja bananowa Rosenbrocka postaci:

f(x,y) = (1-x)2 + 100(y-x\*x)2 , -5 <= x <= 5, -5 <= y <= 5

# Warunek stopu

W obu metodach zastosowałem taki sam warunek stopu:

* Jeżeli ilość iteracji przekroczy zadana
* Jeżeli kolejny x i y różni się od poprzedniego o mniejszą wartość niż epsilon

# Metoda najszybszego spadku gradientu

## Gradient





∇f(x, y)=[ 2 (-1 + x + 200 x3 - 200 x y) , 200 (-x2+ y)]

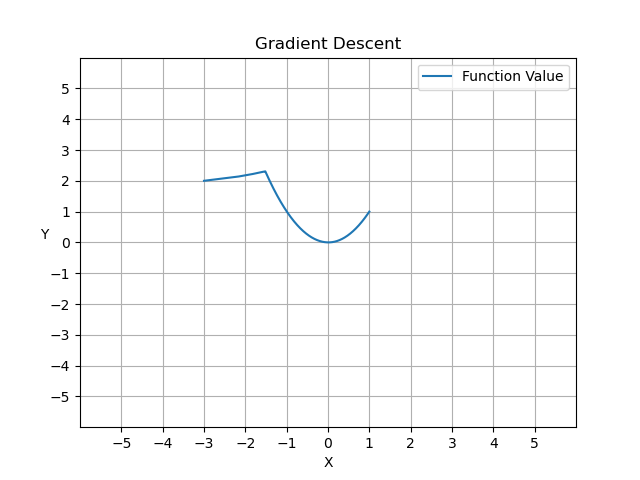
## Pomiary

Wartości w komórkach ilość iteracji \ czas wykonywania

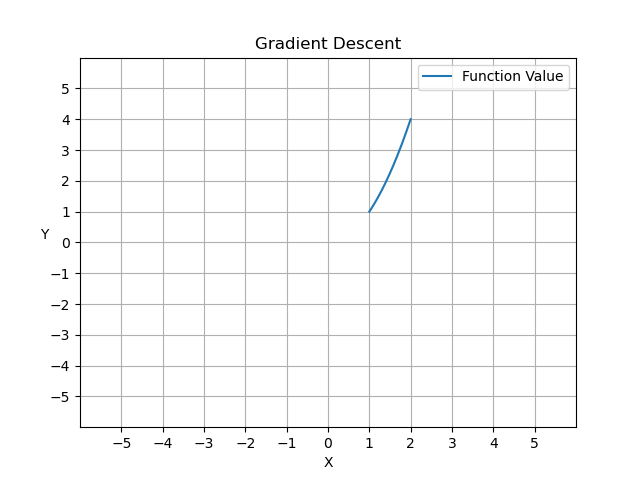
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| B(x, y)\punkt | (-5, -2) | (4, 0) | (2, 3) | (-2, 3) | (-1, 5) |
| 0,0001 | 425618\1,1 | 408959\1,05 | 484116\1,27 | 446538\1,15 | 457336\1,16 |
| 0,0005 | rozbieżność | 224411\0,59 | 250719\0,64 | 231939\0,6 | 237267\0,6 |
| 0,0015 | rozbieżność | rozbieżność | 104813\0,27 | 97343\ 0,25 | 99344\ 0,25 |
| 0,002 | rozbieżność | rozbieżność | 51634\ 0,13 | 49845\ 0,13 | 51032\ 0,13 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| b(x, y) | (1, 4) | (2, -3) | (2, 0) | (3, 0) | (-2, -4) |
| 0,0001 | 498271\1,27 | 425733\1,09 | 414000\1,05 | 409141\1,03 | 425770\1,1 |
| 0,0005 | 257667\0,66 | 221542\0,56 | 215987\0,55 | 216323\0,56 | 221557\0,57 |
| 0,0015 | 107434\0,27 | 93204\ 0,24 | 92135\ 0,23 | 93682\0,24 | 93204\0,24 |
| 0,002 | 51642\ 0,s13 | rozbieżność | 48989\0,12 | rozbieżność | rozbieżność |

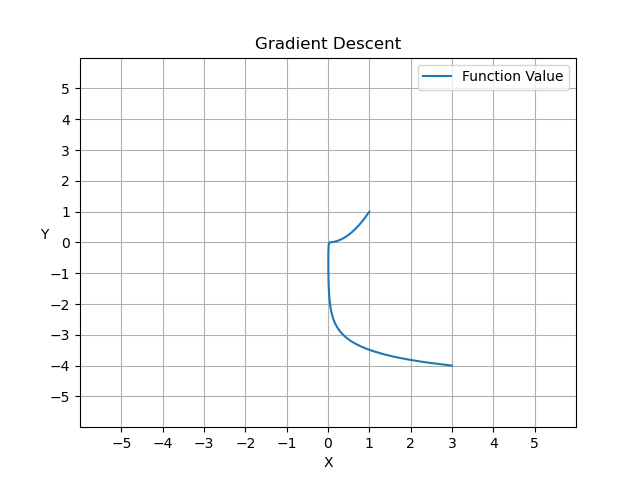
# Wykres przebiegu algorytmu najszybszego spadku gradientu



Przebieg algorytmu dla punktu początkowego (-3, 2) i b=0.0001



Przebieg algorytmu dla punktu początkowego (2,4) i b=0.0001



Przebieg algorytmu dla punktu początkowego (3,-4) i b=0.0004

## Obserwacje

Metoda najszybszego spadku gradientu w pierwszej kolejności dąży do najbliższego punktu paraboli zbliżonej do paraboli y=x2. Na tej paraboli znajdują się punkty bliskie minimum globalnemu (1, 1). W następnej kolejności metoda porusza się po, wyżej wspomnianej paraboli w kierunku minimum globalnego.

# Metoda Newtona

## Hesjan

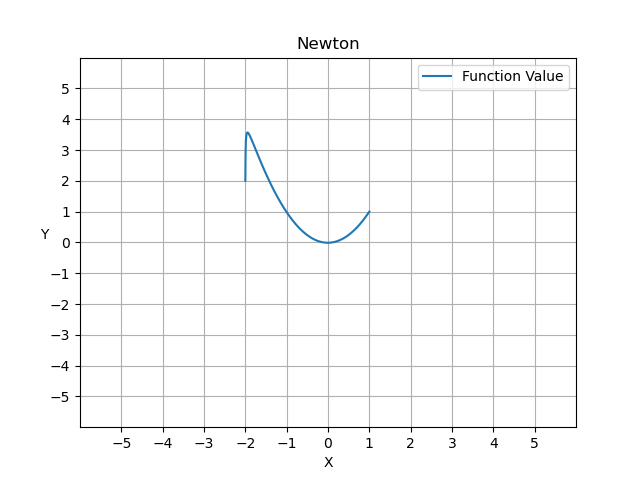
∇2f(x, y)= [1200x2-400y+2, -400x ]

[-400x, 200 ]

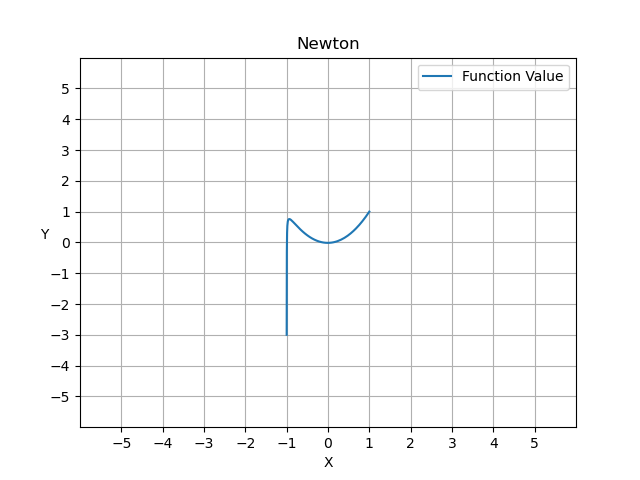
## Pomiary:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| b(x, y) | (-1, 1) | (-2, 3) | (1, 2) | (4, 2) | (-2, 1) |
| 0,01 | 2586\0.03 | 3102\0.03 | 2293\0.02 | 3363\0.04 | 3211\0.03 |
| 0,1 | 305\0.01 | 355\0.01 | 242\0.01 | 379\0.01 | 365\0.01 |
| 0,5 | 59\0.01 | 73\0.01 | 40\0.0 | 77\0.01 | 75\0.01 |
| 1 | 3\0.01 | 7\0.01 | 2\0.01 | 6\0.01 | 7\0.01 |

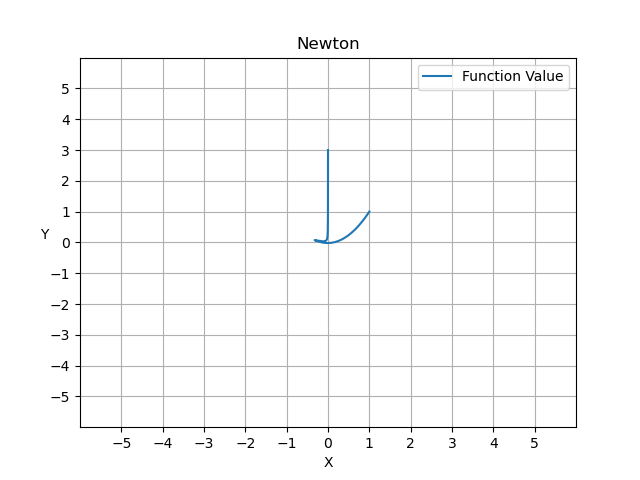
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| b(x, y) | (2, -4) | (1, -3) | (3, -2) | (-5, 4) | (2, 0) |
| 0,01 | 3112\0.03 | 2431\0.03 | 3251\0.03 | 3627\0.04 | 3043\0.03 |
| 0,1 | 331\0.01 | 255\0.01 | 357\0.01 | 432\0.01 | 325\0.01 |
| 0,5 | 61\0.01 | 42\0.01 | 70\0.01 | 94\0.01 | 60\0.01 |
| 1 | 6\0.01 | 2\0.01 | 6\0.01 | 6\0.01 | 6\0.01 |



Przebieg algorytmu dla punktu początkowego (-2,2) i b=0.1



Przebieg algorytmu dla punktu początkowego (-1,-3) i b=0.2



Przebieg algorytmu dla punktu początkowego (0,3) i b=0.5

# Podsumowanie

Powyższe pomiary pozwoliły na zaobserwowanie dużej korelacji między wybraną betą a ilością iteracji i szybkością wykonania pomiaru. Pomiary udały się dla wszystkich wylosowanych punktów. Im większą betę przyjmiemy tym mniejsza jest ilość iteracji i czas znalezienia minimum, ale większa szansa na oscylację. Dla każdego punktu możliwe jest wybranie lepszej bety, która zmniejszy ilość iteracji i zredukuje czas wykonywania programu. Niestety nie jest to takie proste, ponieważ sprowadza się to do ręcznego wyszukiwania bety za pomocą metody prób i błędów.

Metoda Newtona jest dużo szybsza i wykonuje się w mniejsza ilość iteracji, kosztem większej ilości obliczeń (musimy obliczyć Hesjan, który dodatkowo trzeba pomnożyć razy gradient). Beta w metodzie Newtona może być nawet kilka tysięcy razy większa co prowadzi do znaczącego przyspieszenia programu.