Wprowadzenie do sztucznej inteligencji

Ćwiczenie nr. 1

Bartosz Latosek

1. Opis ćwiczenia

Zadanie polega na zaimplementowaniu dwóch sposobów wyznaczania minimum funkcji Himmelblau. Badane metody to Spadek Gradientu oraz metoda Newtona. Zostaną one porównane ze sobą pod względem ogólnego działania, czasu wykonywania i skuteczności.

Użyte technologie

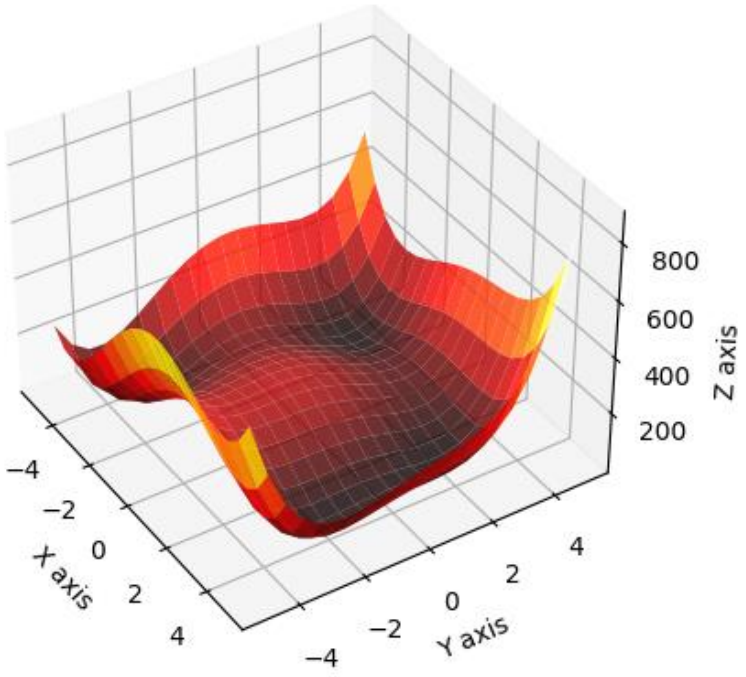
Python (3.9.0) wraz z dwoma bibliotekami:

> *matplotlib (3.4.3)*

> *numpy (1.21.2)*

Opis badanej funkcji (Himmelblau)

$f(x, y) = (x * x+y-11)\ 2 + (x+y * y-7)\ 2$, $gdzie -5 <= x, y <= 5$



Pochodne 1-go i 2-go rzędu:

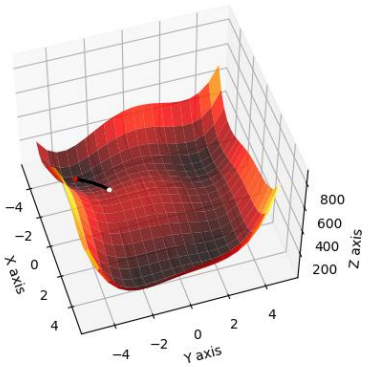
$$\frac{\partial f}{\partial x}(x,y) = 4x(x^2 + y - 11) + 2(x + y^2 - 7)$$
$$\frac{\partial f}{\partial y}(x,y) = 2(x^2 + y - 11) + 4y(x + y^2 - 7)$$

$$\frac{\partial f}{\partial x \partial x}(x,y) = 12x^2 + 4y - 42$$
$$\frac{\partial f}{\partial x \partial y}(x,y) = 4x + 4y$$
$$\frac{\partial f}{\partial y \partial x}(x,y) = 4x + 4y$$
$$\frac{\partial f}{\partial y \partial y}(x,y) = 12y^2 + 4x - 26$$

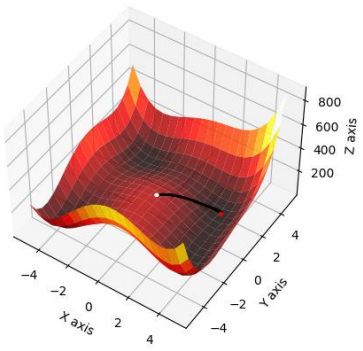
Fig.1 Pochodne 1-go i 2-go rzędu.

2. Badanie metody spadku gradientu

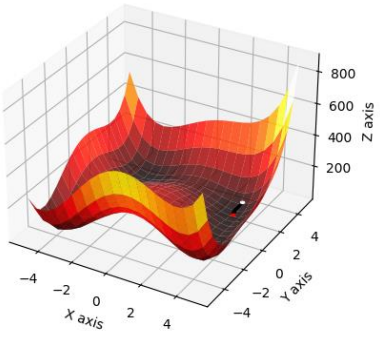
Przykładowe wyniki działania:



$x_0 = 1.52 \quad y_0 = -3.77$



$x_0 = 0 \quad y_0 = 0$



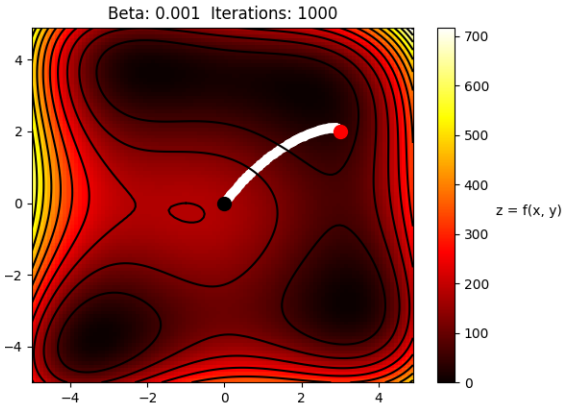
$x_0 = 3 \quad y_0 = 3$

Na powyższych przykładach punkt startowy jest reprezentowany poprzez biały punkt na wykresie, natomiast czerwony punkt jest znalezionym punktem minimalnym. Parametr Beta był ustawiony na 0.001 natomiast algorytm działał w 1000 iteracji.

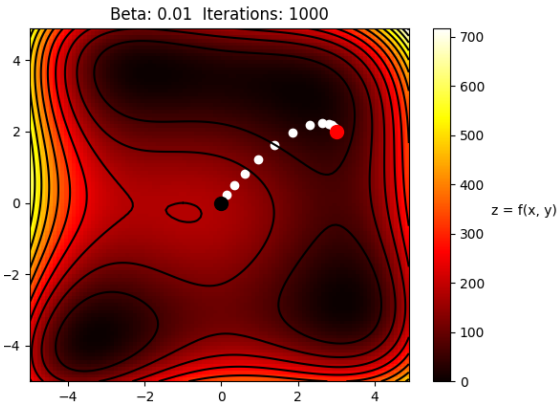
Algorytm:

Algorytm polega na obliczaniu gradientu funkcji w kolejnych punktach w każdej iteracji. Gradient liczony jest przez funkcję *calculateGradient()* , zwracającą wartość pochodnej **dx** i **dy** w punkcie **(x, y)**

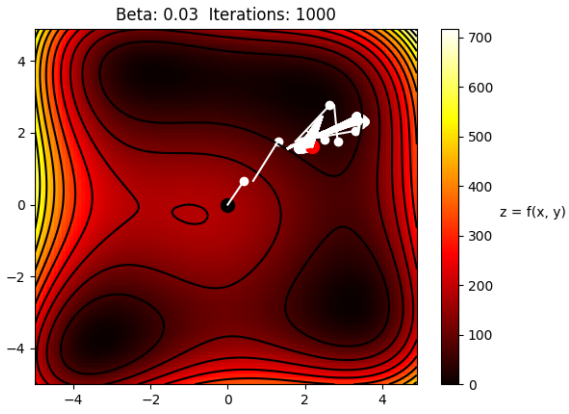
Poniżej prezentacja zmian w działaniu algorytmu przy zmienianiu liczby iteracji oraz wartości parametru Beta. Za punkt startowy przyjmuję punkt **(0, 0).**



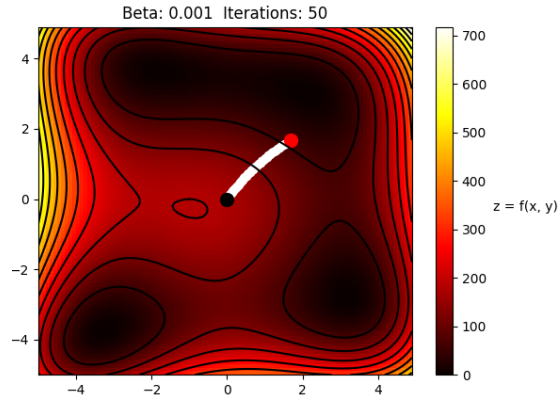
Rysunek 1



Rysunek 2



Rysunek 3



Rysunek 4

Wnioski:

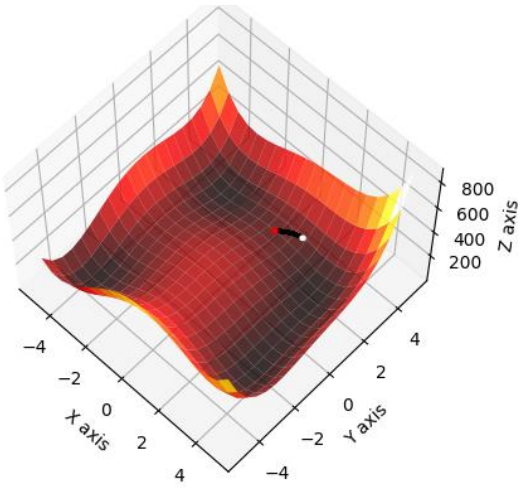
Na *1 rysunku* przedstawione są poprawnie dobrane parametry. Kroki są małe, ale po 1000 iteracji algorytm odnajduje punkt minimalny. Podobnie jest w

przypadku 2 rysunku, tutaj natomiast Beta jest znacznie większa, przez co algorytm „dochodzi” do punktu minimalnego znacznie szybciej.

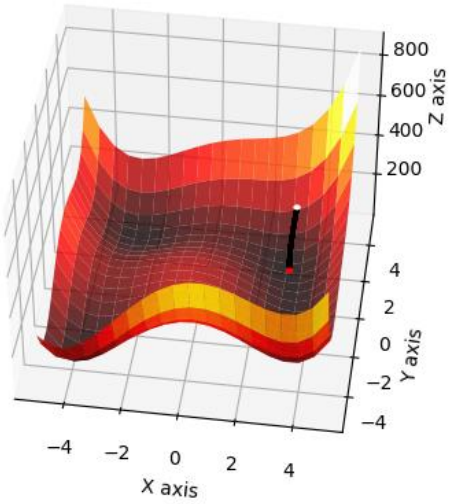
W przypadku 3 rysunku parametr Beta został ustawiony na zbyt dużą wartość, przez co algorytm po zbliżeniu się do poszukiwanego punktu wpada w oscylacje przez co nigdy go nie osiągnie. Na 4 rysunku liczba iteracji jest zbyt mała, aby algorytm „zdążył” zbliżyć się do punktu minimalnego.

3. Badanie metody Newtona

Przykładowy wyniki działania:



$x_0 = 1.33\ y_0 = 3.71$



$x_0 = 3\ y_0 = 4$

Na powyższych przykładach punkt startowy jest reprezentowany poprzez biały punkt na wykresie, natomiast czerwony punkt jest znalezionym punktem minimalnym. Parametr Beta był ustawiony na 0.001 natomiast algorytm działał w 5000 iteracji. Działanie algorytmu na lewym obrazku postaram się lepiej wytłumaczyć w dalszej części sprawozdania.

Algorytm:

Do działania algorytmu niezbędne jest obliczenie Hesjanu funkcji Himmelblau. (Fig. 1)

$$\mathrm{H} = \begin{bmatrix} \frac{\partial \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{x}} & \frac{\partial \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{y}} \\ \frac{\partial \mathrm{f}}{\partial \mathrm{y}\,\partial \mathrm{x}} & \frac{\partial \mathrm{f}}{\partial \mathrm{y}\,\partial \mathrm{y}} \end{bmatrix} \qquad (1)$$

$$\mathrm{H} = \begin{bmatrix} 12x^2 + 4y - 42 & 4x + 4y \\ 4x + 4y & 12y^2 + 4x - 26 \end{bmatrix} \qquad (2)$$

$$H^{-1} = \frac{1}{det(\boldsymbol{H}(\boldsymbol{x},\boldsymbol{y}))} \begin{bmatrix} \frac{\partial \mathrm{f}}{\partial \mathrm{y}\,\partial \mathrm{x}}(x,y) & -\frac{\partial \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{y}}(x,y) \\ -\frac{\partial \mathrm{f}}{\partial \mathrm{y}\,\partial \mathrm{x}}(x,y) & \frac{\partial \mathrm{f}}{\partial \mathrm{x}\,\partial \mathrm{x}}(x,y) \end{bmatrix} \qquad (3)$$

$$\begin{bmatrix} temp\_x \\ temp\_y \end{bmatrix} = H^{-1} \times \begin{bmatrix} \partial x(x,y) \\ \partial y(x,y) \end{bmatrix} \qquad (4)$$

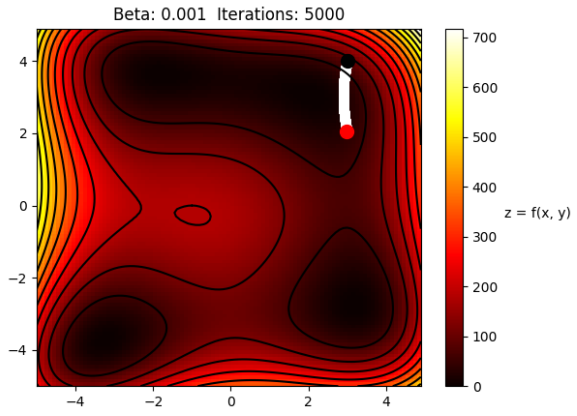
Hesjan wyliczany jest na podstawie wzoru (1). W przypadku funkcji Himmelblau będzie on miał postać (2). Aby wyznaczyć macierz odwrotną do hesjanu korzystamy ze wzoru (3). W algorytmie w każdej iteracji wykonywana jest poniższa instrukcja:

$$x -= B * temp\_x$$

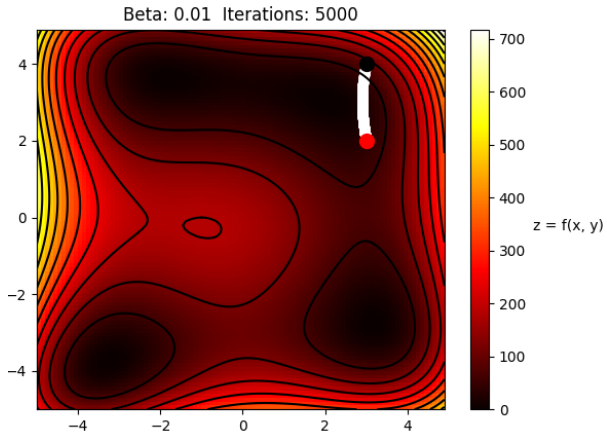
$$y -= B * temp\_y$$

Gdzie B = wsp. Beta, a temp_x i temp_y obliczane są ze wzoru (4).

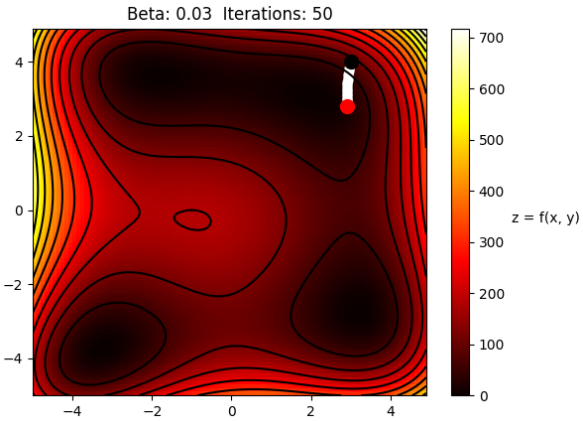
Poniżej prezentacja zmian w działaniu algorytmu przy zmienianiu liczby iteracji oraz wartości parametru Beta. Za punkt startowy przyjmuję punkt **(3, 4).**



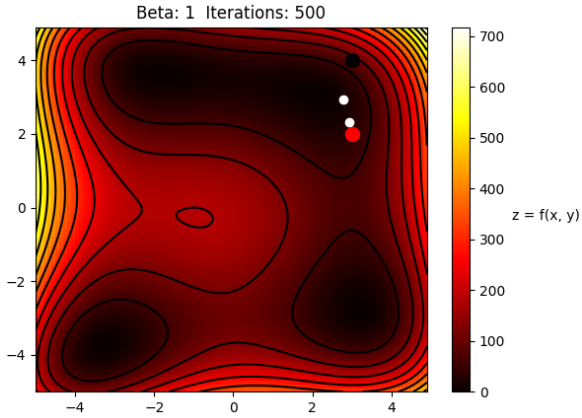
Rysunek 1



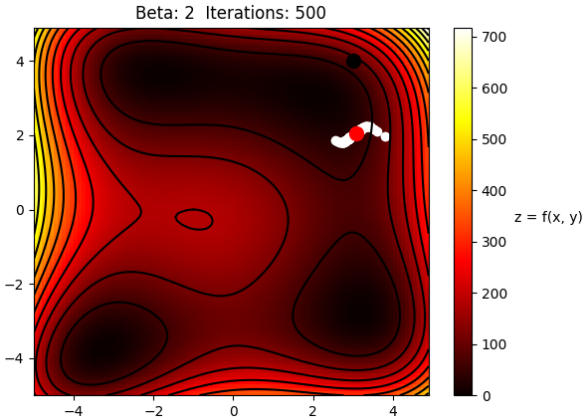
Rysunek 2



Rysunek 3



Rysunek 3



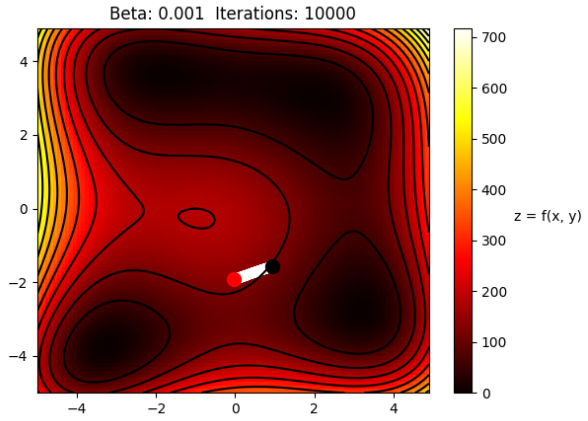
Rysunek 5

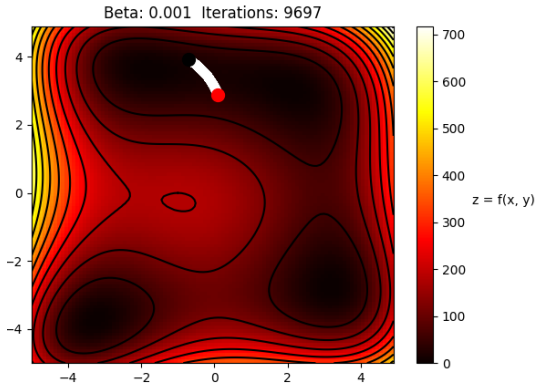
Na rysunku 1 i 2, algorytm zachowuje się podobnie do algorytmu spadku gradientu. Na rysunku 3, liczba iteracji jest za mała aby algorytm „zdążył” dotrzeć do punktu minimalnego. Na rysunku 4 parametr Beta jest znacznie większy, niż w pozostałych przypadkach, przez co liczba widzialnych kroków jest znacznie mniejsza. Na rysunku 5 natomiast parametr Beta jest zbyt duży, przez co algorytm wpada w oscylacje i nigdy nie osiąga dokładnie szukanego minimum.

UWAGI:

Podczas testowania działania algorytmu Newtona, wielokrotnie zdarzało się, że nie zatrzymywał się on w punkcie minimalnym, a w tak zwanym punkcie siodłowym funkcji.

Przykłady:





Punkt siodłowy funkcji, to punkt w którym jej pochodna jest bliska zeru. Metoda Newtona nie potrafi rozróżnić ich od minimów, więc czasami zdarza się, że zamiast szukać punktu minimalnego, szuka ona właśnie punktu siodłowego.

4. Czasy wykonania:

Pomiar wykonywany za pomocą biblioteki time.

(Punkt startowy (0, 0), Beta = 0.001, ε ~ 0)

Metoda Gradientu:

Liczba Iteracji	Czas wykonania [ms]
100	1041
500	859
1000	2169
2500	998
10000	1999

Metoda Newtona:

Liczba Iteracji	Czas wykonania [s]
100	1042
500	1963
1000	9744
2500	11451
10000	89945

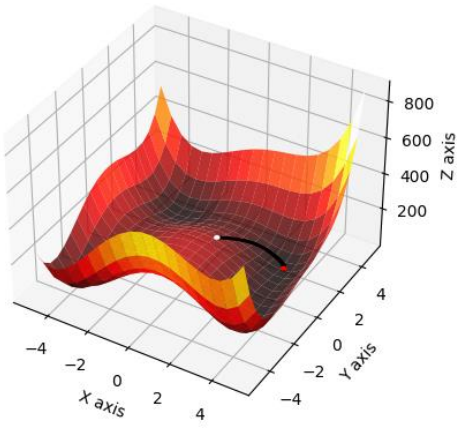
Wnioski:

Wyniki czasowe pokrywają się z założeniami. Metoda Newtona wykonuje więcej obliczeń, przez co czas wykonania jest znacznie dłuższy niż ten w Metodzie gradientu.

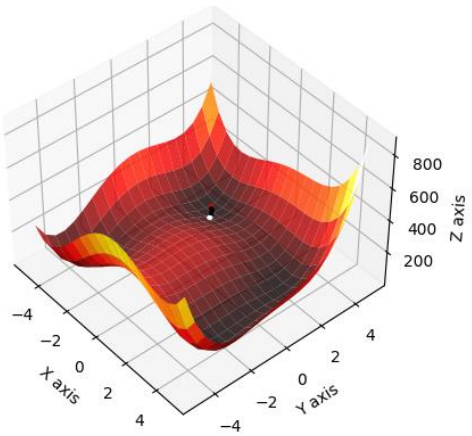
5. Wnioski ogólne:

Metoda gradientu jest szybsza od metody Newtona, ponieważ w każdej iteracji wykonywane jest mniej obliczeń. Jest ona też bardziej niezawodna, gdyż nie jest podatna na punkty siodłowe.

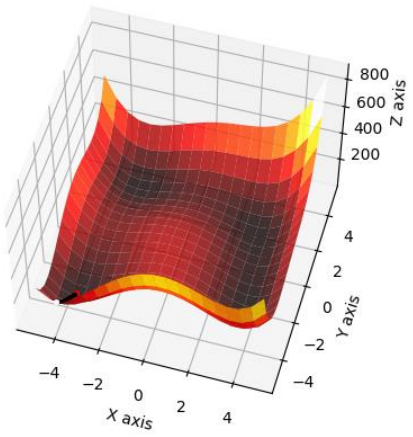
Znalezione punkty minimalne funkcji Himmelblau:



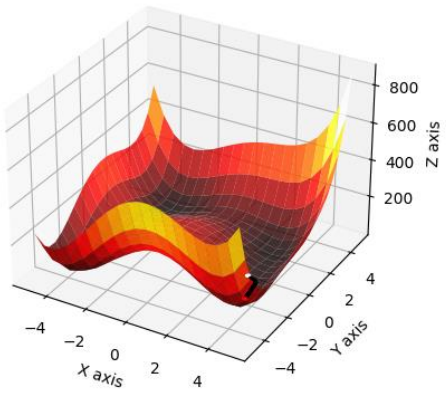
Xo = 3.00 Yo = 2.00



Xo = -2.82 Yo = 3.13



Xo = -3.78 Yo = -3.28



Xo = 3.58 Yo = -1.85

Przy ostatecznym wyznaczaniu punktów minimalnych używałem zgodnie z wywnioskowanymi faktami metody spadku gradientu. Punkty startowe są wybierane losowo, a znalezione punkty minimalne pokrywają się z tymi obliczonymi matematycznie.