

## Zadanie 1. Dokumentacja: Implementacja algorytmu gradientu prostego

Artem Kukushkin, 317140

### 1. Opis implementowanych algorytmów

W ramach tego projektu zaimplementowano algorytm gradientu prostego do minimalizacji funkcji celu. Algorytm ten iteracyjnie aktualizuje wartości wektora zmiennych decyzyjnych w kierunku przeciwnym do gradientu funkcji celu, zgodnie ze wzorem:

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

Gdzie:

- $x_t$  – wektor zmiennych decyzyjnych w iteracji  $t$ ,
- $\alpha$  – współczynnik uczenia (krok algorytmu),
- $\nabla f(x_t)$  – gradient funkcji celu w punkcie  $x_t$ .

Algorytm został zaprojektowany tak, aby można było zastosować go do różnych funkcji celu. W implementacji wykorzystano bibliotekę **autograd** do obliczania gradientów.

### Warunki stopu

Aby uniknąć nieskończonej liczby iteracji, algorytm kończy działanie, gdy spełniony zostanie jeden z warunków:

- Norma gradientu jest mniejsza od zadanej tolerancji  $\epsilon$  (epsilon),
- Osiągnięta została maksymalna liczba iteracji.

### 2. Opis planowanych eksperymentów numerycznych

Aby zbadać skuteczność algorytmu gradientu prostego, przeprowadzono eksperymenty dla trzech różnych funkcji celu:

#### a. Funkcja kwadratowa:

$$f(x) = \sum x_i^2$$

Jest to funkcja wypukła o minimum globalnym w punkcie (0,0)(0,0).

#### b. Funkcja Rosenbrocka:

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

Znajduje szeroką dolinę prowadzącą do minimum globalnego w punkcie (1,1)(1,1), co czyni ją trudnym problemem optymalizacyjnym.

### c. Funkcja Ackleya:

$$f(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right) - \exp \left( \frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) + 20 + e$$

Funkcja ta posiada wiele minimów lokalnych, a jej globalne minimum znajduje się w punkcie (0,0)(0,0).

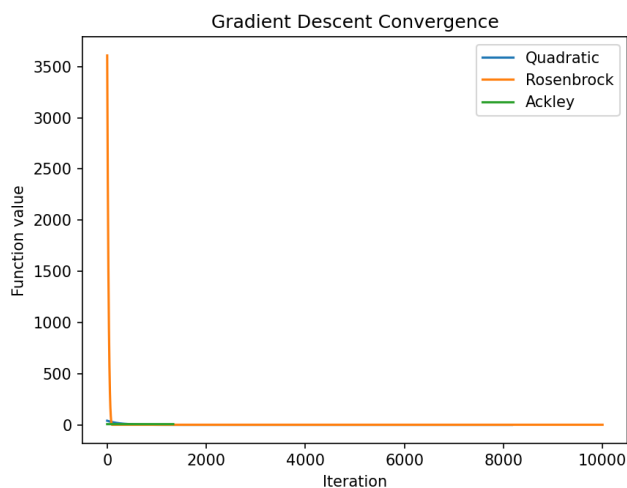
Dla każdej z funkcji przetestowano różne wartości początkowe x0:

[-1.2, 1.0], [-2, 2], [-5, 5], [2, 2], [0, 0], [-10, 10], [100, -100]

Eksperymenty miały na celu zbadanie wpływu wyboru początkowego punktu na zbieżność algorytmu oraz jego skuteczność w minimalizacji funkcji.

### 3. Opis uzyskanych wyników

**X0 = [2, 2]**



**Wyniki dla różnych wartości x0:**

x0	Funkcja kwadratowa	Funkcja Rosenbrocka	Funkcja Ackleya
[-1.2, 1.0]	$[-3.84 \cdot 10^{-7}, 3.20 \cdot 10^{-7}]$ 0.868s	[0.992, 0.984] 2.698s	[-0.968, 0.968], 0.102s
[-2, 2]	$[-3.53 \cdot 10^{-7}, 3.53 \cdot 10^{-7}]$ 0.894s	[0.989, 0.978] 2.722s	[-1.974, 1.974], 0.091s
[-5, 5]	$[-3.53 \cdot 10^{-7}, 3.53 \cdot 10^{-7}]$ 1.061s	[-2.693, 7.284] 2.666s	[-4.986, 4.986], 0.090s

[2, 2]	$[3.53 \cdot 10^{-7}, 3.53 \cdot 10^{-7}]$ 0.878s	[1.032, 1.066], 2.737s	[1.974, 1.974], 0.099s
[0, 0]	[0, 0], 0.002s	[0.994, 0.989], 2.890s	[nan, nan], 3.681s
[-10, 10]	$[-3.53 \cdot 10^{-7}, 3.53 \cdot 10^{-7}]$ 1.005s	[-4, 15.956], 2.657s	[-9.995, 9.995], 0.085s
[100, -100]	[1.838, -1.838], 1.084s	[0.106, -0.010] 2.612s	[100, -100], 0.000s

### Analiza wyników

- **Funkcja kwadratowa** – algorytm dobrze radził sobie z minimalizacją niezależnie od wartości  $x_0$ .
- **Funkcja Rosenbrocka** – gradient prosty miał trudności z dojściem do optymalnego rozwiązania dla dużych wartości  $x_0$ , co wynika z jej specyfiki.
- **Funkcja Ackleya** – wartości  $x_0 = [0, 0]$  i  $x_0 = [100, -100]$  pokazały problemy numeryczne i brak zbieżności.

### 4. Wnioski z przeprowadzonych badań

1. Zbieżność algorytmu zależy od wyboru funkcji celu i punktu startowego.
2. Gradient prosty może nie radzić sobie dobrze z funkcjami o wielu minimach lokalnych (Ackley).
3. Dla funkcji Rosenbrocka algorytm często kończy w lokalnym minimum zamiast w optymalnym punkcie (1, 1).
4. Dobre wartości początkowe przyspieszają zbieżność, ale nie eliminują problemu utknięcia w minimum lokalnym.