TD 1: Gradient Descent Algorithm

Gradient Descent is an **iterative optimization algorithm** used to minimize a differentiable function.

Given a function:

$$f(x)$$

The update rule is:

$$x_{k+1} = x_k - \alpha \nabla f(x_k)$$

Where:

- x_k : current point
- $\alpha > 0$: learning rate (step size)
- $\nabla f(x_k)$: gradient at point x_k

- **Algorithm (Pseudo-Code)**

```
Input: f(x), learning rate α, initial point x0, tolerance ε
Output: Approximate minimizer x*

1. Set k = 0
2. Repeat:
       Compute gradient gk = ∇f(xk)
       Update xk+1 = xk - α gk
       k = k + 1
   Until ||∇f(xk)|| < ε
3. Return xk
```

Exercice 1 :

Minimize the function:

$$f(x) = x^2$$

Using Gradient Descent with:

- Initial point x0=4
- Learning rate α=0.1
- Perform 3 iterations.

Exercice 2 :

Minimize:

$$f(x, y) = x^2 + y^2$$

Given:

- (x0,y0)=(2,1)
- α=0.1
- Perform 2 iterations.

Exercise 3: Linear Regression with Gradient Descent

We consider the dataset:

x	y
1	2
2	3
3	5

We want to fit a linear model:

$$\hat{y} = wx + b$$

Using the Mean Squared Error:

$$J(w,b) = \frac{1}{2n}\sum_{i=1}^{n}(wx_i + b - y_i)^2$$

Tasks

1. Compute the gradients $\frac{\partial J}{\partial w}$, $\frac{\partial J}{\partial b}$
2. Perform **one iteration** of Gradient Descent
   - Initial values: $w_0 = 0, b_0 = 0$
   - Learning rate: $\alpha = 0.1$

Exercise 4: One Neuron (Forward + Back propagation) with Gradient Descent

Consider a single neuron:

$$z = w_1x_1 + w_2x_2 + b$$

$$\hat{y} = \sigma(z)$$

where the sigmoid function is:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Loss function (Binary Cross Entropy):

$$J = -[y\log(\hat{y}) + (1-y)\log(1-\hat{y})]$$

Given:

- $x_1 = 1$
- $x_2 = 2$
- $y = 1$
- $w_1 = 0.2$
- $w_2 = -0.4$
- $b = 0.1$
- $\alpha = 0.1$

Questions:

- Perform forward propagation
- Compute gradients using backpropagation
- Update parameters