Homework-seminar 10

Jitareanu Eduard-David, 914

```python
# importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt


# defining the quadratic function
1 usage
def f(x, y, b):
    # with the formula:
    return 0.5 * (x ** 2 + b * y ** 2)


# defining the gradient of the quadrtic function
1 usage
def gradient(x, y, b):
    # calculating the gradient vector
    return np.array([x, b * y])
```

```python
# performing gradient descent
1 usage
def gradient_descent(b, x0, y0, learning_rate, num_iterations):
    # initializing lists to store the values during iterationsS
    x_values, y_values = [x0], [y0]

    # loop through the iterations
    for _ in range(num_iterations):
        # calculating the gradient
        grad = gradient(x_values[-1], y_values[-1], b)

        # updating x and y based on the learning rate and gradient
        x_values.append(x_values[-1] - learning_rate * grad[0])
        y_values.append(y_values[-1] - learning_rate * grad[1])

    # returning the result as numpy arrays
    return np.array(x_values), np.array(y_values)
```

```python
def plot_contour(b):
    # creating a meshgrid for the contour plot
    x = np.linspace(-5,  stop: 5,  num: 100)
    y = np.linspace(-5,  stop: 5,  num: 100)
    X, Y = np.meshgrid( *xi: x, y)

    # calculating the function values for the contour plot
    Z = f(X, Y, b)

    # plotting the contours
    plt.contour( *args: X, Y, Z, levels=20)
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title(f'Contour plot for b = {b}')
    plt.grid(True)
```

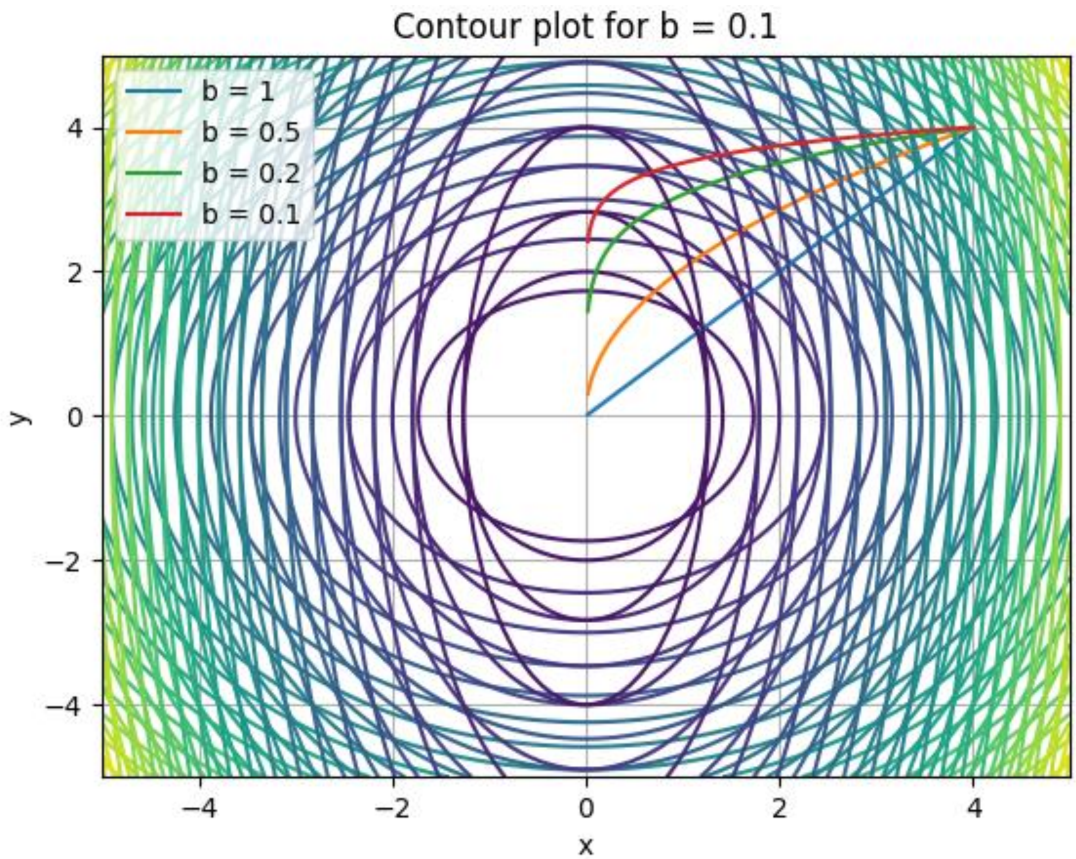
```python
# Setting parameters for the gradient descent
learning_rate = 0.1
num_iterations = 50

# Let's try different values of b
for b_value in [1, 0.5, 0.2, 0.1]:
    # Running the gradient descent
    x_values, y_values = gradient_descent(b_value,  x0: 4,  y0: 4, learning_rate, num_iterations)

    # Plotting the results
    plot_contour(b_value)
    plt.plot( *args: x_values, y_values, label=f'b = {b_value}')

# Adding a legend and showing the plot
plt.legend()
plt.show()
```

Results:



Jătăreanu Eduard
David, 914.

Homework - 10.

1. Find the tangent plane to the unit sphere $x^2+y^2+z^2=1$ at an arbitrary point (x_0, y_0, z_0).

$$\left.\begin{aligned} \frac{\partial f}{\partial x} &= 2x \\ \frac{\partial f}{\partial y} &= 2y \\ \frac{\partial f}{\partial z} &= 2z \end{aligned}\right\} \Rightarrow \left.\begin{aligned} &\nabla f = \langle 2x, 2y, 2z \rangle \\ &(x_0, y_0, z_0) \text{-arbitrary point} \end{aligned}\right\} \Rightarrow \nabla f(x_0, y_0, z_0) = \langle 2x_0, 2y_0, 2z_0 \rangle$$

The equation of the tangent plane:

$$2x_0(x - x_0) + 2y_0(y - y_0) + 2z_0(z - z_0) = 0 \Rightarrow$$

$$\Rightarrow 2x_0 x - 2x_0^2 + 2y_0 y - 2y_0^2 + 2z_0 z - 2z_0^2 = 0 \Rightarrow$$

$$\left.\begin{aligned} &\Rightarrow 2x_0 x + 2y_0 y + 2z z_0 = 2(x_0^2 + y_0^2 + z_0^2) \\ &\text{but } (x_0^2 + y_0^2 + z_0^2) = 1 \end{aligned}\right\} \Rightarrow$$

$\Rightarrow 2x_0 x + 2y_0 y + 2z_0 z = 2$ - the equation of the tangent plane to the unit sphere at an arbitrary point (x_0, y_0, z_0).

2a). $f: \mathbb{R}^2 \to \mathbb{R}$, $f(x,y) = \frac{1}{2}(x^2 + by^2)$, $b > 0$

$(x_{k+1}, y_{k+1}) = (x_k, y_k) - \alpha_k \nabla f(x_k, y_k)$.

~~$\phi(\alpha_k) = f(x_{k+1}, y_{k+1}) = f(x_k, y_k) - \alpha_k \nabla f(x_k, y_k)$~~

$\phi(\alpha_k) = f(x_{k+1}, y_{k+1}) = f((x_k, y_k) - \alpha_k \nabla f(x_k, y_k)) \Rightarrow$

$\Rightarrow \phi'(\alpha_k) = \nabla f(x_{k+1}, y_{k+1}) \cdot (-\nabla f(x_k, y_k))$.

$(x_{k+1}, y_{k+1}) = (x_k, y_k) - \alpha_k \nabla f(x_k, y_k) \Rightarrow$

$$\Rightarrow \begin{cases} x_{k+1} = x_k - \alpha_k \frac{\partial f}{\partial x}(x_k, y_k) = x_k - \alpha_k x_k \\ y_{k+1} = y_k - \alpha_k \frac{\partial f}{\partial y}(x_k, y_k) = y_k - \alpha_k b y_k \end{cases}$$

$\phi'(\alpha_k) = x_{k+1} \cdot (-x_k) + b\, y_{k+1} (-by_k) \Rightarrow$

$\Rightarrow \phi'(\alpha_k) = (x_k - \alpha_k x_k) \cdot (-x_k) + b(y_k - \alpha_k b y_k) \cdot (-by_k) \Rightarrow$

$$\left.\begin{aligned} \Rightarrow \phi'(\alpha_k) &= -x_k^2 + \alpha_k x_k^2 - b^2 y_k^2 + \alpha_k (b^3 y_k^2) \\ \phi'(\alpha_k) &= 0 \end{aligned}\right\} \Rightarrow$$

$\Rightarrow \alpha_k (x_k^2 + b^3 y_k^2) = x_k^2 + b^2 y_k^2 \Rightarrow$

$\Rightarrow \alpha_k = \dfrac{x_k^2 + b^2 y_k^2}{x_k^2 + b^3 y_k^2}$ - the optimal step size for exact line search.