05.01.2024
Țapordei Maia
Mathematical Analysis
Homework 10

Q: Ex 1

Find the tangent plane to the unit sphere $x^2+y^2+z^2=1$ at an arbitrary point.

$f(x,y,z) = x^2+y^2+z^2-1=0$

$\nabla f(x,y,z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) = (2x, 2y, 2z)$

We take points $x_0, y_0, z_0 \Rightarrow \nabla f(x_0,y_0,z_0) = (2x_0, 2y_0, 2z_0)$

Equation of tangent plane: $2x_0(x-x_0)+2y_0(y-y_0)+2z_0(z-z_0)=0 \Rightarrow$

$\Rightarrow 2x_0x+2y_0y+2z_0z = 2x_0^2+2y_0^2+2z_0^2 \Rightarrow$

$\Rightarrow 2x_0x+2y_0y+2z_0z = 2 \Rightarrow x_0x+y_0y+z_0z=1$

Q: Ex 2

Let $x, y, f: \mathbb{R}^2 \to \mathbb{R}$ and $f(x,y) = f(x(u,v), y(u,v))$. Prove that $\frac{\partial f}{\partial u} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial u}$ and $\frac{\partial f}{\partial v} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial v}$

$f(x,y)$ depends on u,v indirectly through $x(u,v)$ and $y(u,v)$.

When u changes, f changes both through x and y:

$x \Rightarrow \frac{\partial f}{\partial x}\frac{\partial x}{\partial u}$, $y \Rightarrow \frac{\partial f}{\partial y}\frac{\partial f}{\partial u} \Rightarrow$

$\Rightarrow \frac{\partial f}{\partial u} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial u}$

When v changes, f changes both through x and y:

$x \Rightarrow \frac{\partial f}{\partial x}\frac{\partial x}{\partial v}$, $y = \frac{\partial f}{\partial y}\frac{\partial y}{\partial v} \Rightarrow$

$\Rightarrow \frac{\partial f}{\partial v} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial v}$

Q: Ex 3

Consider the quadratic function $f: \mathbb{R}^2 \to \mathbb{R}$, $f(x,y) = 1/2 \cdot (x^2+by^2)$ with $b>0$ and the following gradient descent method for finding its minimum $(x_{k+1}, y_{k+1}) = (x_k, y_k) - s_k \nabla f(x_k, y_k)$, with step size $s_k > 0$ (also called learning rate in the machine learning literature)

a) Find the step size s_k using exact line search, such that it minimizes the function $\varphi(s_k) = f(x_{k+1}, y_{k+1}) = f((x_k, y_k) - s_k \nabla f(x_k, y_k))$.

$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (x, by)$

At point $(x_k, y_k) \Rightarrow \nabla f(x_k, y_k) = (x_k, by_k)$

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

$\Rightarrow x_{k+1} = (1-s_k)x_k$, $y_{k+1} = (1-s_k b)y_k$

$\varphi(s_k) = f(x_{k+1}, y_{k+1}) = 1/2 \cdot (x_{k+1}^2 + by_{k+1}^2) = 1/2 \cdot ((1-s_k)^2 x_k^2 + b(1-s_k b)^2 \cdot y_k^2)$.

$\varphi'(s_k) = \frac{\partial}{\partial s_k}\left[\frac{1}{2}((1-s_k)^2x_k^2 + b(1-s_k b)^2 y_k^2\right] = -(1-s_k)x_k^2 - b^2(1-s_k b)y_k^2$

$\varphi'(s_k) = 0$

$(1-s_k)x_k^2 + b^2(1-s_k b)y_k^2 = 0 \Rightarrow x_k^2 - s_k x_k^2 + b^2 y_k^2 - b^3 s_k y_k^2 = 0 \Rightarrow$

$\Rightarrow s_k(x_k^2 + b^3 y_k^2) = x_k^2 + b^2 y_k^2 \Rightarrow s_k = \frac{x_k^2 + b^2 y_k^2}{x_k^2 + b^3 y_k^2}$

b) What do you notice as b gets smaller?

For $b=1 \Rightarrow$ The contours are circular, and the gradient descent path is straight toward the minimum.

As b decreases $\Rightarrow$ the contours become elongated ellipses. Gradient descent steps "zig-zag" more and take longer to converge. This behaviour occurs because the function has a condition number that worsen as $b \to 0$.

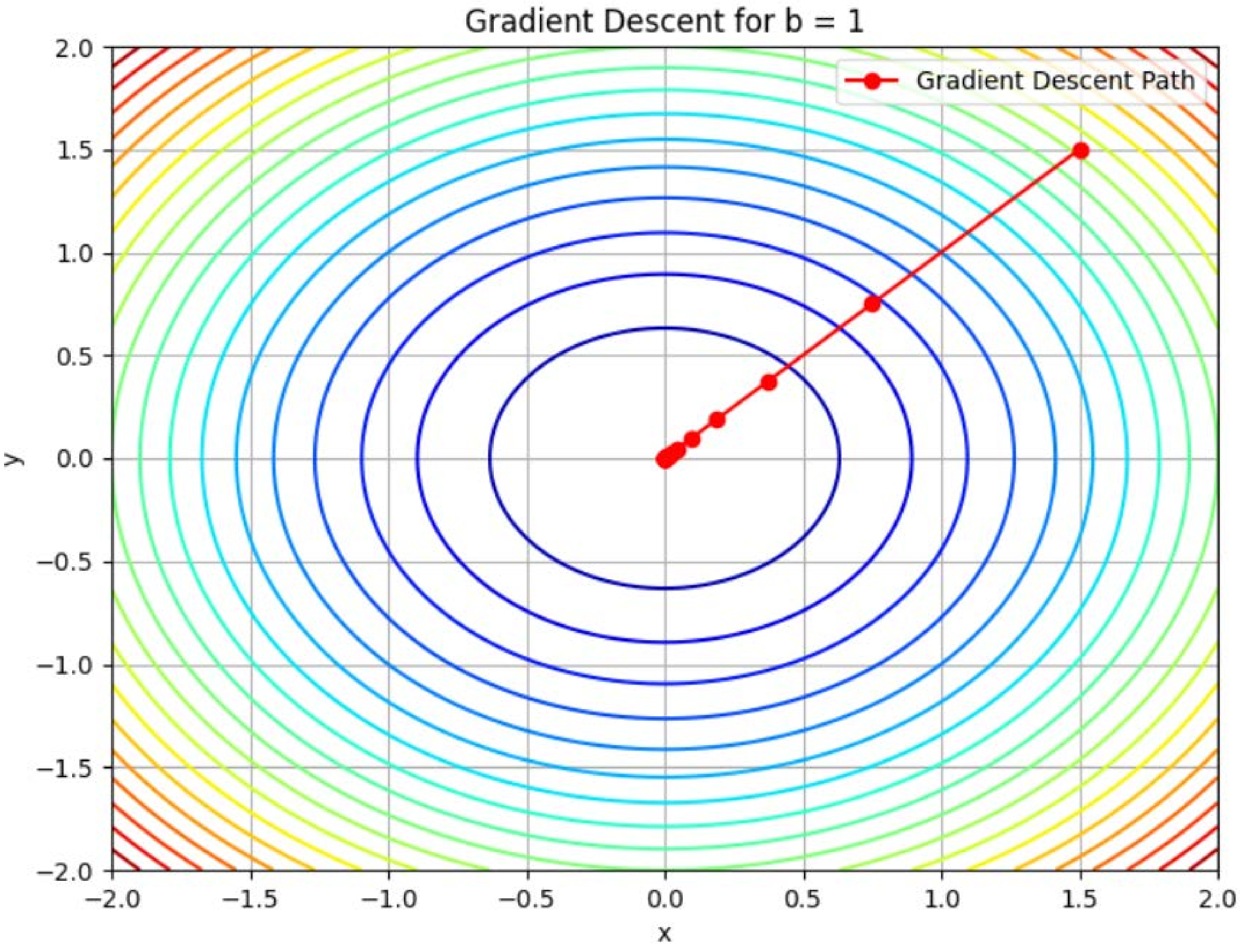
Scanned with CamScanner
Scanat cu CamScanner

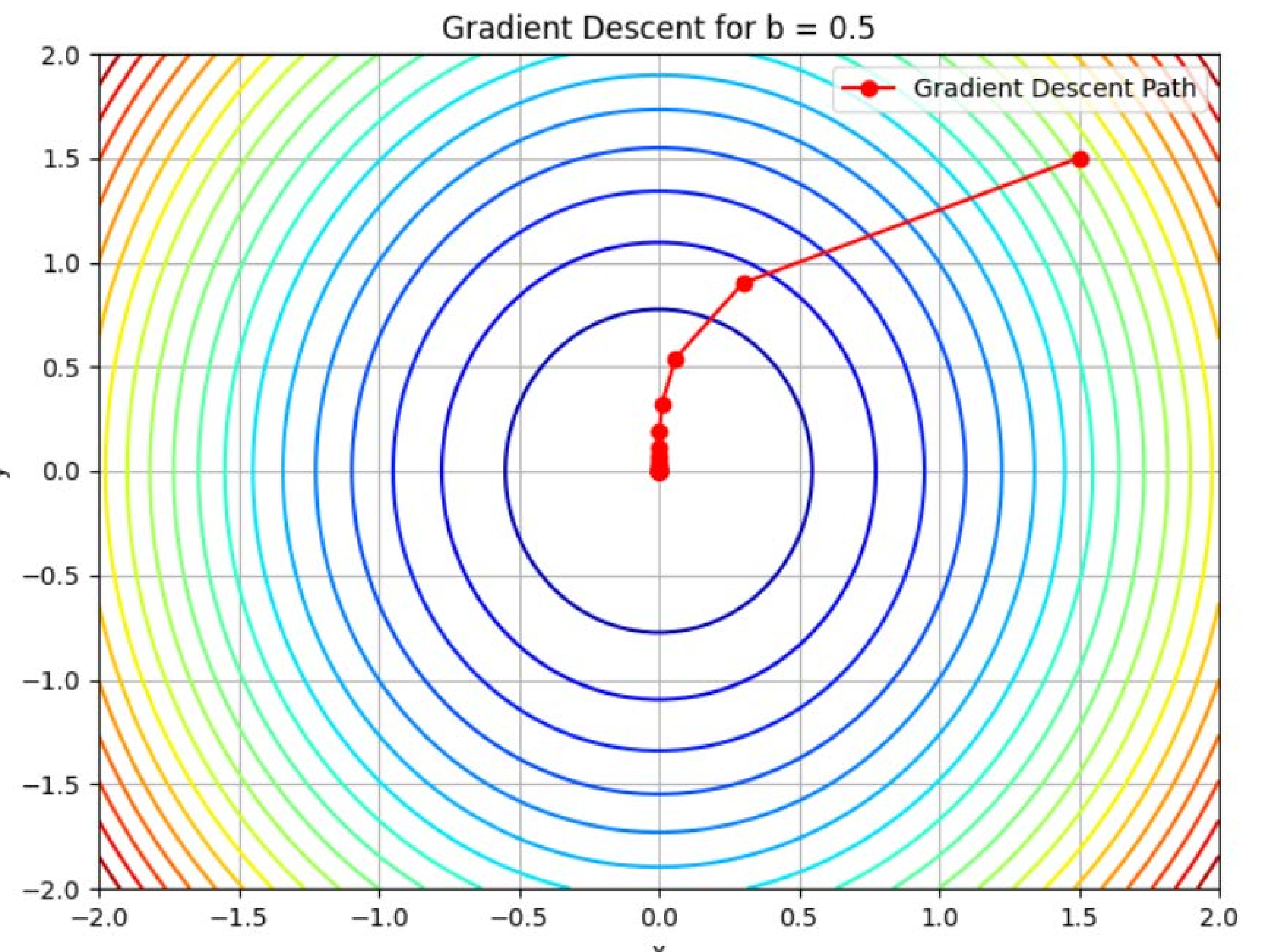
```python
import numpy as np
import matplotlib.pyplot as plt


# Function definition and gradient
def f(x, y, b):
    return 0.5 * (x ** 2 + b * y ** 2)


def grad_f(x, y, b):
    return np.array([x, b * y])


# Gradient descent function
def gradient_descent(x0, y0, b, num_steps=50):
    x, y = x0, y0
    trajectory = [(x, y)]  # To store the path of descent
    step_size = 1 / (1 + b ** 2)  # Optimal step size derived earlier

    for _ in range(num_steps):
        grad = grad_f(x, y, b)
        x -= step_size * grad[0]
        y -= step_size * grad[1]
        trajectory.append((x, y))
    return np.array(trajectory)
```

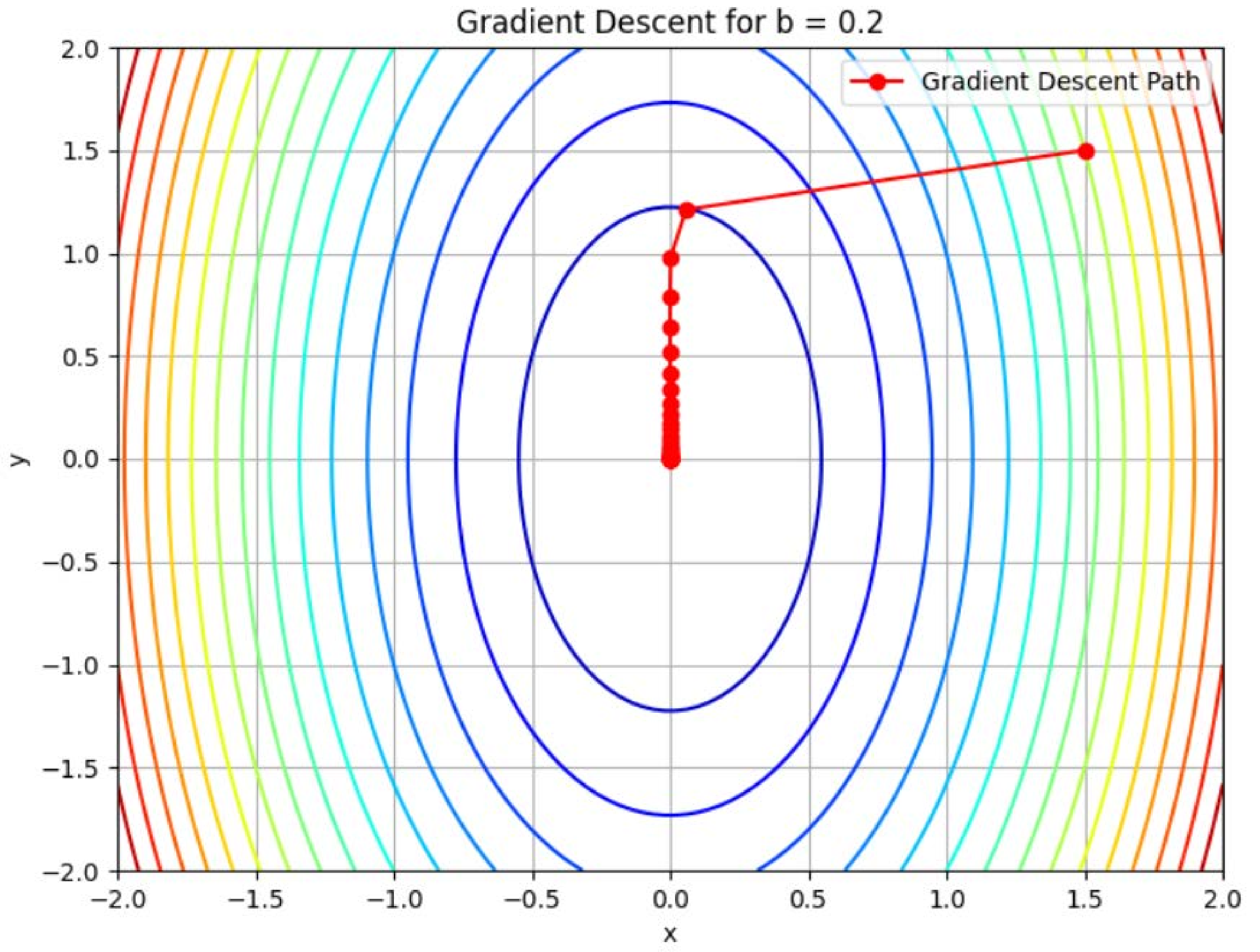
```python
# Plotting function
def plot_contours_and_descent(b_values, num_steps=50):
    x = np.linspace(-2, 2, 100)
    y = np.linspace(-2, 2, 100)
    X, Y = np.meshgrid(x, y)

    for b in b_values:
        Z = f(X, Y, b)  # Compute contours

        # Initial point
        x0, y0 = 1.5, 1.5

        # Perform gradient descent
        trajectory = gradient_descent(x0, y0, b, num_steps)

        # Plotting
        plt.figure(figsize=(8, 6))
        plt.contour(X, Y, Z, levels=20, cmap='jet')
        plt.plot(trajectory[:, 0], trajectory[:, 1], 'o-', color='red', label='Gradient Descent Path')
        plt.title(f"Gradient Descent for b = {b}")
        plt.xlabel('x')
        plt.ylabel('y')
        plt.legend()
        plt.grid()
        plt.show()
```

```python
# Main function
if __name__ == "__main__":
    b_values = [1, 0.5, 0.2, 0.1]  # b = 1, 1/2, 1/5, 1/10
    plot_contours_and_descent(b_values)
```

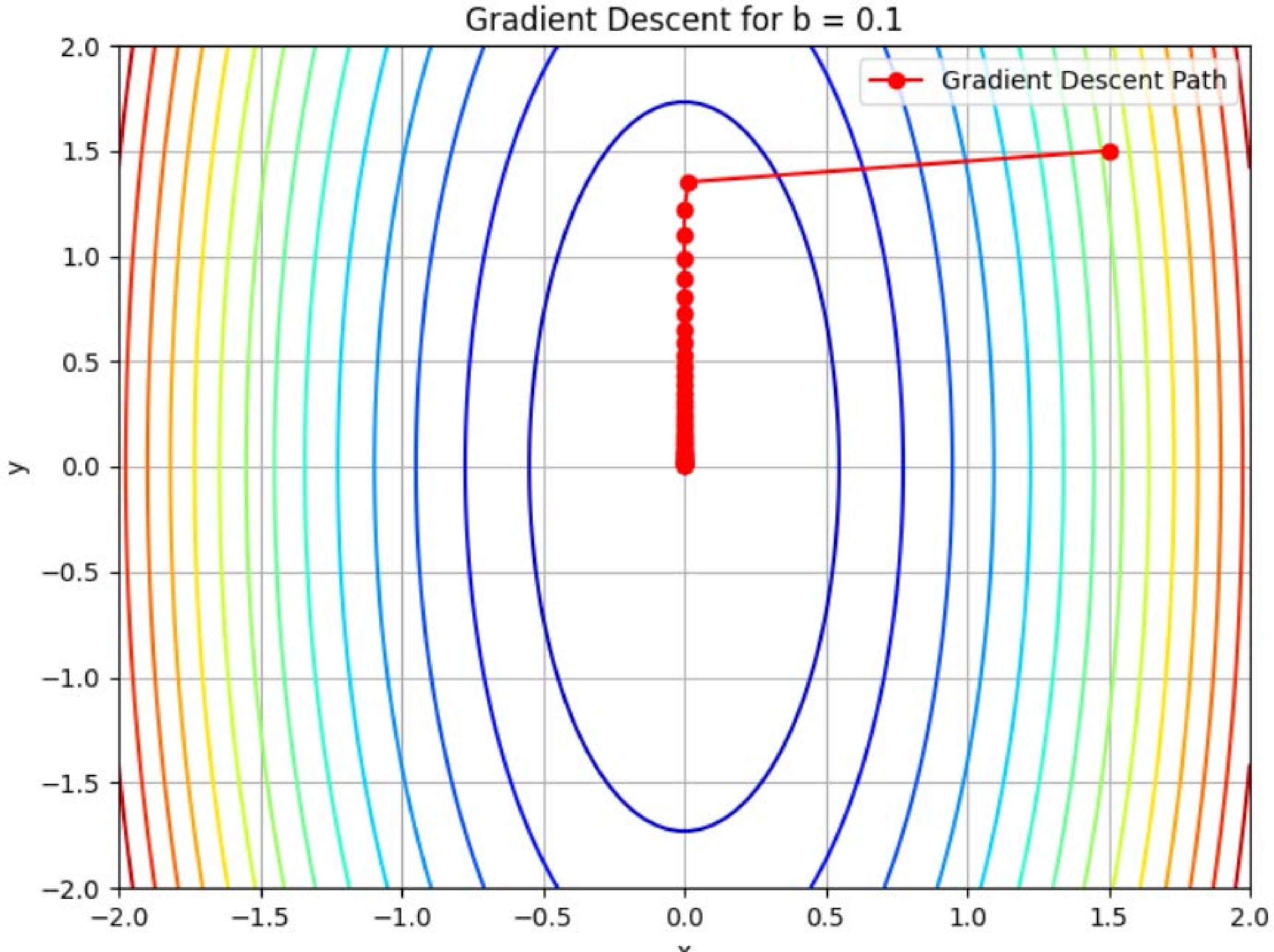
Clo


Gradient Descent for b = 1
Gradient Descent Path
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


Gradient Descent for b = 0.5
Gradient Descent Path
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
x



Gradient Descent for b = 0.2
Gradient Descent Path
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x



Gradient Descent for b = 0.1
Gradient Descent Path
y
x
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0