```python
#Matematyka Konkretna

#Laboratorium 7

#Paweł Wawrzuta https://github.com/PawelWawrzuta/MK-Lab7

#Wariant 9

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Modified function to minimize
def funkcja(x, y):
    return np.sin((x + 3*y)**2)

# Modified gradient descent function
def gradient_descent(learning_rate, iterations):
    x = np.random.uniform(1, 3)
    y = np.random.uniform(1, 3)

    history = []

    for _ in range(iterations):
        # Numerical gradients
        df_dx = (funkcja(x + 1e-6, y) - funkcja(x, y)) / 1e-6
        df_dy = (funkcja(x, y + 1e-6) - funkcja(x, y)) / 1e-6

        x = x - learning_rate * df_dx
        y = y - learning_rate * df_dy
        history.append([x, y, funkcja(x, y)])

    return np.array(history)

# Visualization of the function
x_vals = np.linspace(1, 3, 100)
y_vals = np.linspace(1, 3, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = funkcja(X, Y)

# 3D plot initialization
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis', alpha=0.8, edgecolor='k')

# Initial point
ax.scatter(1, 1, funkcja(1, 1), color='red', marker='o', s=100,
label='Start')

# Optimization
learning_rate = 0.01
```

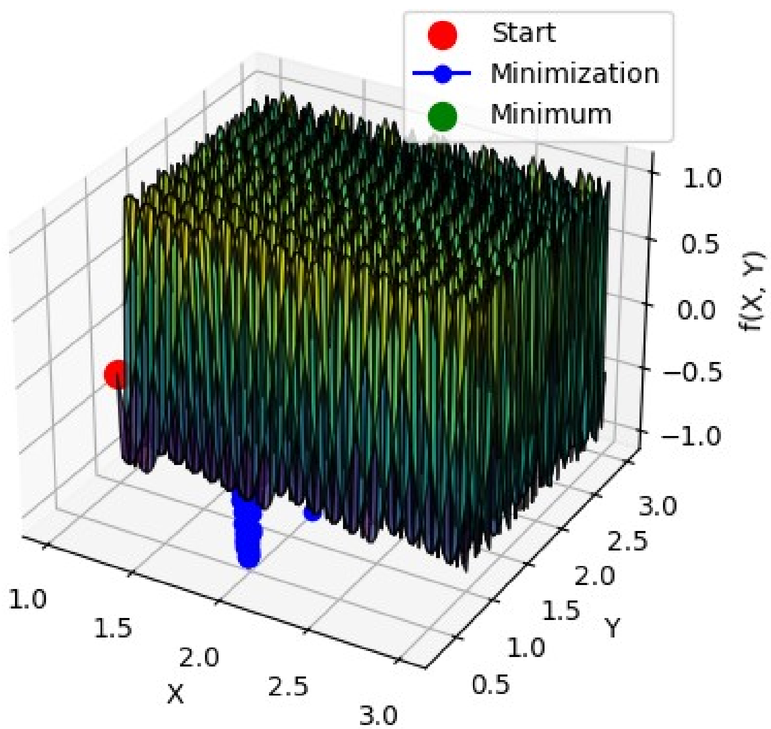
```python
iterations = 100
history = gradient_descent(learning_rate, iterations)

# Trajectory
ax.plot(history[:, 0], history[:, 1], history[:, 2], color='blue',
marker='o', label='Minimization')

# Final minimum point
ax.scatter(history[-1, 0], history[-1, 1], history[-1, 2],
color='green', marker='o', s=100, label='Minimum')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('f(X, Y)')
ax.legend()

plt.show()
```




```
```