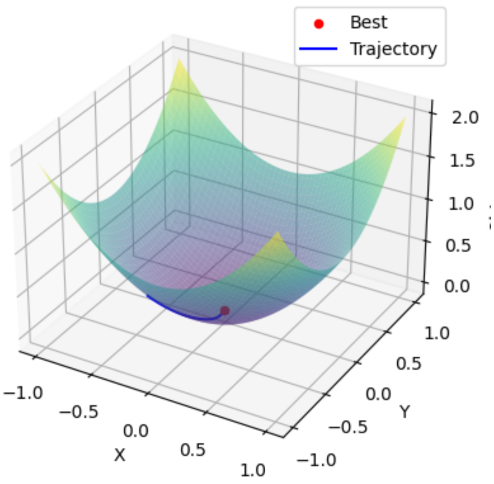
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def objective(x, y):
  return x ** 2.0 + y ** 2.0


def derivative(x, y):
  return np.array([2.0 * x, 2.0 * y])
```

Gradient descent algorithm with Adam

```
def adam(objective, derivative, bounds, n_iter,
    alpha, beta1, beta2, eps=1e-8):
  x = bounds[:, 0] + np.random.rand(len(bounds))\
  * (bounds[:, 1] - bounds[:, 0])
  scores = []
  trajectory = []

  m = np.zeros(bounds.shape[0])
  v = np.zeros(bounds.shape[0])

  for t in range(n_iter):
    g = derivative(x[0], x[1])

    for i in range(x.shape[0]):
      m[i] = beta1 * m[i] + (1.0 - beta1) * g[i]
      v[i] = beta2 * v[i] + (1.0 - beta2) * g[i] ** 2
      mhat = m[i] / (1.0 - beta1 ** (t + 1))
      vhat = v[i] / (1.0 - beta2 ** (t + 1))
      x[i] = x[i] - alpha * mhat / (np.sqrt(vhat) + eps)

    score = objective(x[0], x[1])
    scores.append(score)
    trajectory.append(x.copy())

  return x, scores, trajectory
```

```
bounds = np.array([[-1.0, 1.0], [-1.0, 1.0]])

n_iter = 60

alpha = 0.02

beta1 = 0.8

beta2 = 0.999

best, scores, trajectory = adam(objective, derivative,
                                bounds, n_iter, alpha,
                                beta1, beta2)

x = np.linspace(bounds[0, 0], bounds[0, 1], 100)
y = np.linspace(bounds[1, 0], bounds[1, 1], 100)
X, Y = np.meshgrid(x, y)
Z = objective(X, Y)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis', alpha=0.5)
ax.scatter(best[0], best[1], objective(best[0], best[1]),
    color='red', label='Best')
ax.plot([point[0] for point in trajectory],
    [point[1] for point in trajectory], scores,
    color='blue', label='Trajectory')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Objective')
ax.legend()

plt.show()
```


Best
Trajectory
2.0
1.5
1.0
0.5
0.0
−1.0
−0.5
0.0
X
0.5
1.0
1.0
0.5
0.0
Y
−0.5
−1.0