Lab Assignment 4

.. Optimization Functions

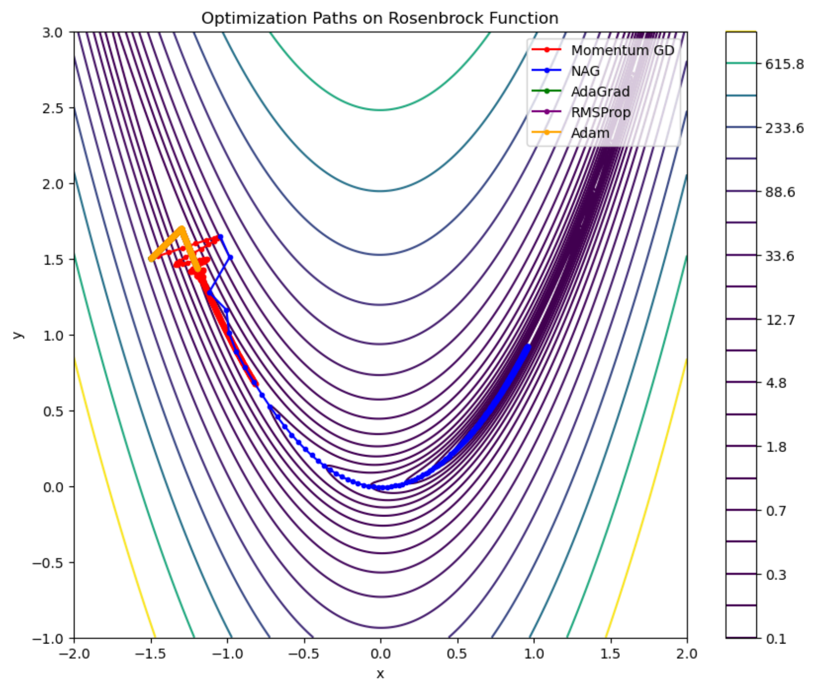
(a) Consider the Rosenbrock function and perform the following optimization: Momentumbased **Gradient Descent (Momentum GD), Nesterov Accelerated Gradient Descent (NAG), AdaGrad, RMSProp, Adam** with following parameter:

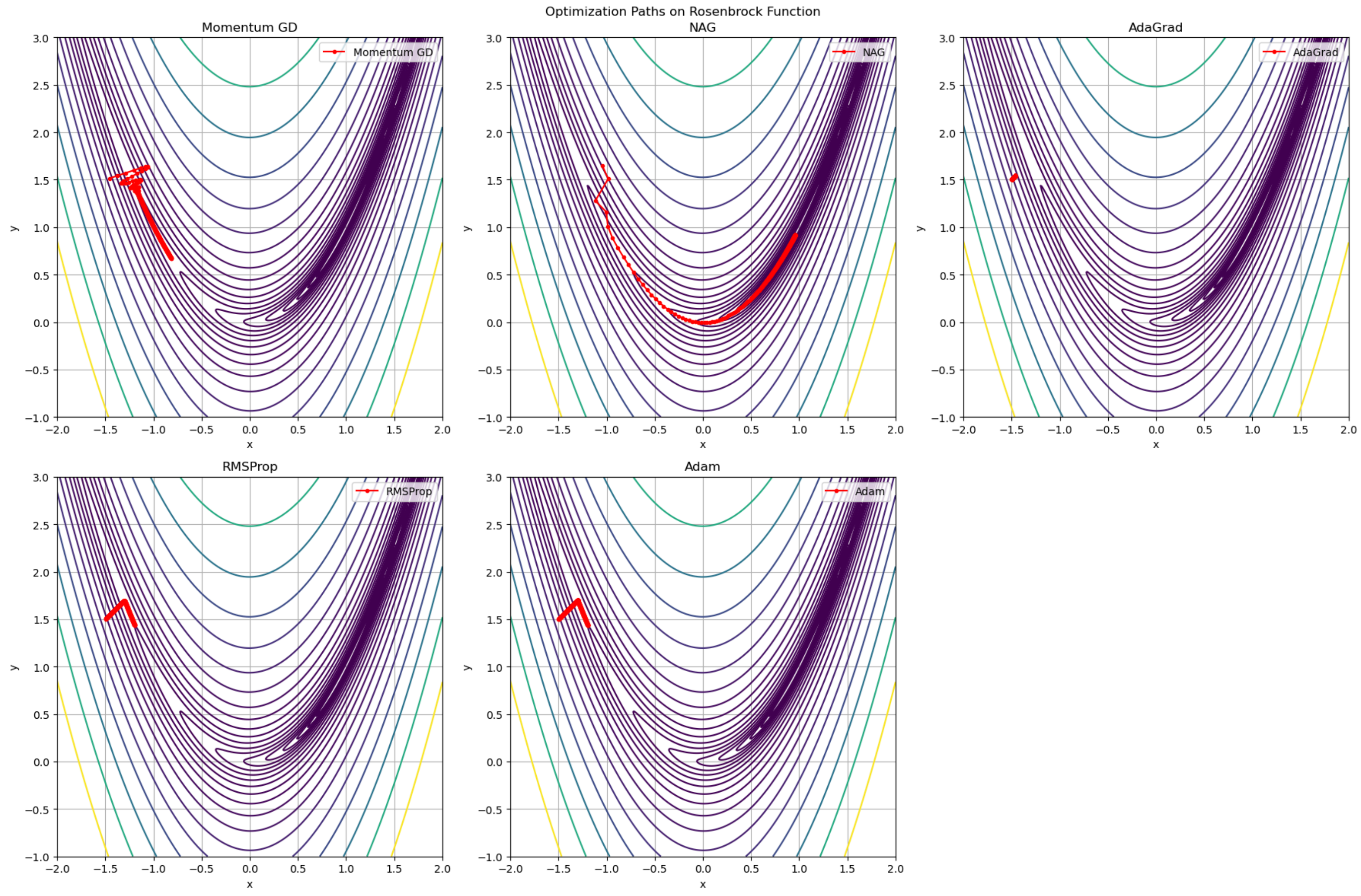
- learning rate = 0.001
- beta = 0.9
- iterations = 500
- start point = (-1.5, 1.5)

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

# Define the Rosenbrock function
def rosenbrock(x, y):
  return (1 - x)**2 + 100 * (y - x**2)**2
def rosenbrock_grad(x, y):
  df_dx = -2 * (1 - x) - 400 * x * (y - x**2)
  df_dy = 200 * (y - x**2)
  return np.array([df_dx, df_dy])
def optimize(method, x, y, alpha, beta, iterations):
  v = np.zeros(2)
  G = np.zeros(2)
  m = np.zeros(2)
  path = []

  for t in range(1, iterations + 1):
    grad = rosenbrock_grad(x, y)

    if method == "Momentum GD":
      v = beta * v + (1 - beta) * grad
      x, y = np.array([x, y]) - alpha * v

    elif method == "NAG":
      grad = rosenbrock_grad(x - beta * v[0], y - beta * v[1])
      v = beta * v + alpha * grad
      x, y = np.array([x, y]) - v

    elif method == "AdaGrad":
      G += grad**2
      x, y = np.array([x, y]) - alpha * grad / (np.sqrt(G) + 1e-8)

    elif method == "RMSProp":
      G = beta * G + (1 - beta) * grad**2
      x, y = np.array([x, y]) - alpha * grad / (np.sqrt(G) + 1e-8)

    elif method == "Adam":
      m = beta * m + (1 - beta) * grad
```

```
            v = beta * v + (1 - beta) * grad**2
            m_hat = m / (1 - beta**t)
            v_hat = v / (1 - beta**t)
            x, y = np.array([x, y]) - alpha * m_hat / (np.sqrt(v_hat) + 1e-8)

        path.append((x, y))

    return path

x0, y0 = -1.5, 1.5
alpha = 0.001
beta = 0.9
iterations = 500
methods = ["Momentum GD", "NAG", "AdaGrad", "RMSProp", "Adam"]
paths = {method: optimize(method, x0, y0, alpha, beta, iterations) for method in methods}

# Create a grid for contour plot
x_vals = np.linspace(-2, 2, 400)
y_vals = np.linspace(-1, 3, 400)
X, Y = np.meshgrid(x_vals, y_vals)
Z = rosenbrock(X, Y)

# Plot optimization paths separately
fig, axes = plt.subplots(2, 3, figsize=(18, 12))
axes = axes.flatten()

for i, method in enumerate(methods):
    ax = axes[i]
    ax.contour(X, Y, Z, levels=np.logspace(-1, 3, 20), cmap='viridis')

    path = np.array(paths[method])
    ax.plot(path[:, 0], path[:, 1], label=method, marker='o', markersize=3, color='red')

    ax.set_title(method)
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.legend()
    ax.grid(True)

# Remove extra subplot
fig.delaxes(axes[-1])

plt.suptitle('Optimization Paths on Rosenbrock Function')
plt.tight_layout()
plt.show()
```

```
Momentum GD: -0.8146775638661041 0.6717642926839041
NAG: 0.9590474567976796 0.9196040653397444
AdaGrad: -1.4587682336803645 1.5414195159824142
RMSProp: -1.1952631828184124 1.436179411032135
Adam: -1.1936506938736722 1.4338007212063029
GD final value: [-0.8675442   0.76071108]
SGD final value: [-0.8763019  0.7721509]
Mini-batch SGD final value: [-0.86608056  0.75880034]
```





Thank You Sir