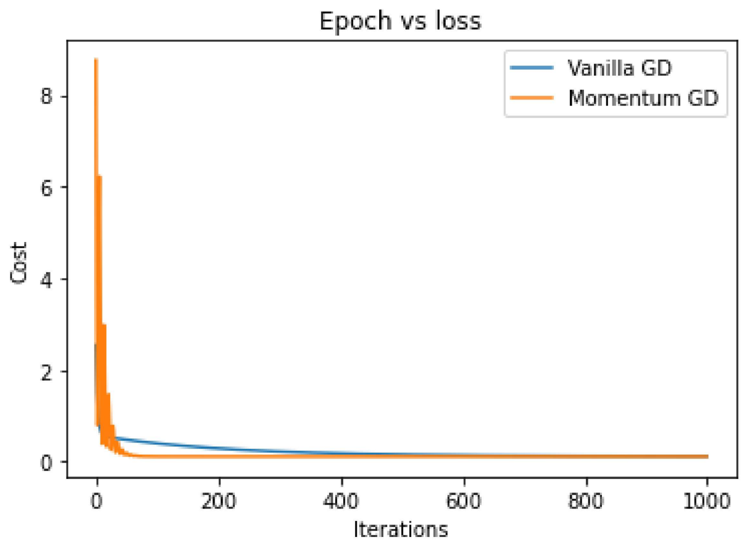
In [34]:
```
import numpy as np
from matplotlib import pyplot as plt

# Define the input data
x = np.c_[np.ones(3), np.array([1, 3.5, 6])]
y = np.array([4, 5.5, 9]).reshape((-1, 1))

# Define the learning rate, number of iterations, and momentum factor
learning_rate = 0.01
num_iterations = 1000
momentum_factor = 0.9

# Initialize the parameters
theta_vanilla = np.random.rand(2, 1)
velocity_vanilla = np.zeros_like(theta_vanilla)
theta_momentum = np.random.rand(2, 1)
velocity_momentum = np.zeros_like(theta_momentum)

# Define the cost function
def cost_function(theta, x, y):
    m = len(y)
    predictions = x.dot(theta)
    cost = (1/(2*m)) * np.sum(np.square(predictions-y))
    return cost

# Define the gradient function
def gradient_function(theta, x, y):
    m = len(y)
    predictions = x.dot(theta)
    gradient = (1/m) * x.T.dot(predictions-y)
    return gradient

# Initialize lists to store costs
vanilla_costs = []
momentum_costs = []

# Vanilla gradient descent
for i in range(num_iterations):
    gradient = gradient_function(theta_vanilla, x, y)
    theta_vanilla = theta_vanilla - learning_rate * gradient

    # Calculate and store the cost
    cost = cost_function(theta_vanilla, x, y)

    vanilla_costs.append(cost)

# Momentum-based gradient descent
for i in range(num_iterations):
    gradient = gradient_function(theta_momentum, x, y)
    velocity_momentum = momentum_factor * velocity_momentum - learning_rate *
    theta_momentum = theta_momentum + velocity_momentum

    # Calculate and store the cost
    cost = cost_function(theta_momentum, x, y)
    momentum_costs.append(cost)

# Plot the cost function over iterations
```

```
plt.plot(range(num_iterations), vanilla_costs, label='Vanilla GD')
plt.plot(range(num_iterations), momentum_costs, label='Momentum GD')
plt.xlabel('Iterations')
plt.ylabel('Cost')
plt.title('Epoch vs loss')
plt.legend()
plt.show()
```



In [30]:

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

# Define the input data
x = np.array([1, 3.5, 6]).reshape((-1, 1))
y = np.array([4, 5.5, 9]).reshape((-1, 1))

# Define the learning rate, number of iterations, and momentum factor
learning_rate = 0.01
num_iterations = 1000
momentum_factor = 0.9

# Initialize the parameters
theta = np.random.rand(1, 2)
velocity = np.zeros_like(theta)

# Define the cost function
def cost_function(theta, x, y):
    m = len(y)
    predictions = x.dot(theta)
    cost = (1/(2*m)) * np.sum(np.square(predictions-y))
    return cost

# Define the gradient function
def gradient_function(theta, x, y):
    m = len(y)
    predictions = x.dot(theta)
    gradient = (1/m) * x.T.dot(predictions-y)
    return gradient

# Vanilla gradient descent
for i in range(num_iterations):
    gradient = gradient_function(theta, x, y)
    theta = theta - learning_rate * gradient

    # Print the cost function every 100 iterations
    if i % 100 == 0:
        cost = cost_function(theta, x, y)
        print(f"Vanilla GD - Iteration {i}: Cost={cost}")

# Momentum-based gradient descent
for i in range(num_iterations):

    gradient = gradient_function(theta, x, y)
    velocity = momentum_factor * velocity - learning_rate * gradient
    theta = theta + velocity

    # Print the cost function every 100 iterations
    if i % 100 == 0:
        cost = cost_function(theta, x, y)
        print(f"Momentum GD - Iteration {i}: Cost={cost}")
```

```
Vanilla GD - Iteration 0: Cost=12.067316670781546
Vanilla GD - Iteration 100: Cost=2.0270727580372276
Vanilla GD - Iteration 200: Cost=2.0270727580372245
Vanilla GD - Iteration 300: Cost=2.0270727580372245
Vanilla GD - Iteration 400: Cost=2.0270727580372245
Vanilla GD - Iteration 500: Cost=2.0270727580372245
Vanilla GD - Iteration 600: Cost=2.0270727580372245
Vanilla GD - Iteration 700: Cost=2.0270727580372245
Vanilla GD - Iteration 800: Cost=2.0270727580372245
Vanilla GD - Iteration 900: Cost=2.0270727580372245
Momentum GD - Iteration 0: Cost=2.0270727580372245
Momentum GD - Iteration 100: Cost=2.0270727580372254
Momentum GD - Iteration 200: Cost=2.0270727580372245
Momentum GD - Iteration 300: Cost=2.0270727580372245
Momentum GD - Iteration 400: Cost=2.0270727580372245
Momentum GD - Iteration 500: Cost=2.0270727580372245
Momentum GD - Iteration 600: Cost=2.0270727580372245
Momentum GD - Iteration 700: Cost=2.0270727580372245
Momentum GD - Iteration 800: Cost=2.0270727580372254
Momentum GD - Iteration 900: Cost=2.0270727580372245
```

In [ ]: