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

initial_x = 0 # Initialize x
initial_y = 1 # Initialize y
learning_rate = 0.01 # Learning rate
num_iterations = 10000 # Number of iterations

def gradient_descent_custom(x_start, y_start, custom_func, custom_gradient, alpha, iterations):
    x_val = x_start
    y_val = y_start

    for i in range(iterations):
        gradient_x, gradient_y = custom_gradient(x_val, y_val)

        x_val = x_val - alpha * gradient_x
        y_val = y_val - alpha * gradient_y

    return x_val, y_val

def custom_function(x, y):
    return 1 - np.exp(-x**2 - (y-2)**2) - 2*np.exp(-x**2 - (y+2)**2)

def custom_gradient_function(x, y):
    grad_x = 2 * x * np.exp(-x**2 - (y-2)**2) + 4 * x * np.exp(-x**2 - (y+2)**2)
    grad_y = 2 * (y-2) * np.exp(-x**2 - (y-2)**2) + 4 * (y+2) * np.exp(-x**2 - (y+2)**2)

    return grad_x, grad_y

optimal_result = gradient_descent_custom(initial_x, initial_y, custom_function, custom_gradient_function, learning_rate, num_iterations)
print("Optimal values:", optimal_result)

# Plotting the function
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

X = np.linspace(-5, 5, 100)
Y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(X, Y)
z = custom_function(x, y)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
plt.title("Question 4C Graph")
ax.plot_surface(x, y, z, cmap='viridis', edgecolor='none')

plt.show()
```

```
Optimal values: (0.0, 1.9999990997123154)
```