Codigo em python

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# Configuração para evitar problemas com interface gráfica

import matplotlib

matplotlib.use('TkAgg') # ou 'Qt5Agg' se TkAgg não funcionar

# --- RESOLUÇÃO MANUAL DAS DUAS PRIMEIRAS ITERAÇÕES ---

def resolucao\_manual():

print("=" \* 60)

print("RESOLUÇÃO MANUAL - DUAS PRIMEIRAS ITERAÇÕES")

print("=" \* 60)

# Função f(x,y) = xy \* exp(-x² - y²)

def f(x, y):

return x \* y \* np.exp(-x\*\*2 - y\*\*2)

# Gradiente: ∇f = [∂f/∂x, ∂f/∂y]

# ∂f/∂x = y \* exp(-x²-y²) \* (1-2x²)

# ∂f/∂y = x \* exp(-x²-y²) \* (1-2y²)

def grad\_f(x, y):

exp\_term = np.exp(-x\*\*2 - y\*\*2)

df\_dx = y \* exp\_term \* (1 - 2\*x\*\*2)

df\_dy = x \* exp\_term \* (1 - 2\*y\*\*2)

return np.array([df\_dx, df\_dy])

# Dados iniciais

x0, y0 = 0.3, 1.2

alpha = 0.1

print(f"Ponto inicial: x₀ = {x0}, y₀ = {y0}")

print(f"Tamanho do passo: α = {alpha}")

print(f"f(x₀, y₀) = {f(x0, y0):.6f}")

# ITERAÇÃO 1

print("\n--- ITERAÇÃO 1 ---")

grad\_0 = grad\_f(x0, y0)

print(f"∇f(x₀, y₀) = [{grad\_0[0]:.6f}, {grad\_0[1]:.6f}]")

x1 = x0 - alpha \* grad\_0[0]

y1 = y0 - alpha \* grad\_0[1]

print(f"x₁ = x₀ - α \* ∂f/∂x = {x0} - {alpha} \* {grad\_0[0]:.6f} = {x1:.6f}")

print(f"y₁ = y₀ - α \* ∂f/∂y = {y0} - {alpha} \* {grad\_0[1]:.6f} = {y1:.6f}")

print(f"f(x₁, y₁) = {f(x1, y1):.6f}")

# ITERAÇÃO 2

print("\n--- ITERAÇÃO 2 ---")

grad\_1 = grad\_f(x1, y1)

print(f"∇f(x₁, y₁) = [{grad\_1[0]:.6f}, {grad\_1[1]:.6f}]")

x2 = x1 - alpha \* grad\_1[0]

y2 = y1 - alpha \* grad\_1[1]

print(f"x₂ = x₁ - α \* ∂f/∂x = {x1:.6f} - {alpha} \* {grad\_1[0]:.6f} = {x2:.6f}")

print(f"y₂ = y₁ - α \* ∂f/∂y = {y1:.6f} - {alpha} \* {grad\_1[1]:.6f} = {y2:.6f}")

print(f"f(x₂, y₂) = {f(x2, y2):.6f}")

return (x0, y0), (x1, y1), (x2, y2)

# --- Definição das Funções ---

def f(x, y):

"""Função objetivo: f(x,y) = xy \* exp(-x² - y²)"""

return x \* y \* np.exp(-x\*\*2 - y\*\*2)

def grad\_f(x, y):

"""Gradiente da função f"""

exp\_term = np.exp(-x\*\*2 - y\*\*2)

df\_dx = y \* exp\_term \* (1 - 2\*x\*\*2)

df\_dy = x \* exp\_term \* (1 - 2\*y\*\*2)

return np.array([df\_dx, df\_dy])

# Executa a resolução manual

pontos\_manuais = resolucao\_manual()

# --- Configurações Iniciais ---

ponto\_inicial = np.array([0.3, 1.2])

lambda\_passo = 0.1

tolerancia = 0.00001

# --- Item (a): Plotar a função e o ponto inicial ---

print("\n" + "=" \* 60)

print("ITEM (a): PLOTANDO FUNÇÃO E PONTO INICIAL")

print("=" \* 60)

x\_grid = np.linspace(-2, 2, 100)

y\_grid = np.linspace(-2, 2, 100)

X, Y = np.meshgrid(x\_grid, y\_grid)

Z = f(X, Y)

fig = plt.figure(figsize=(12, 8))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X, Y, Z, cmap='viridis', alpha=0.7)

ax.scatter(ponto\_inicial[0], ponto\_inicial[1], f(ponto\_inicial[0], ponto\_inicial[1]),

color='red', s=100, label='Ponto Inicial (x₀)')

ax.set\_title('Superfície da Função f(x,y) = xy·exp(-x²-y²) e Ponto Inicial')

ax.set\_xlabel('x')

ax.set\_ylabel('y')

ax.set\_zlabel('f(x,y)')

ax.legend()

plt.savefig('grafico\_a.png', dpi=300, bbox\_inches='tight')

plt.show()

# --- Item (b): Plotar o vetor gradiente na direção do mínimo ---

print("\nITEM (b): PLOTANDO VETOR GRADIENTE")

print("=" \* 60)

gradiente\_inicial = grad\_f(ponto\_inicial[0], ponto\_inicial[1])

direcao\_descida = -gradiente\_inicial

print(f"Gradiente no ponto inicial: {gradiente\_inicial}")

print(f"Direção de descida (-∇f): {direcao\_descida}")

fig = plt.figure(figsize=(12, 8))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X, Y, Z, cmap='viridis', alpha=0.7)

ax.scatter(ponto\_inicial[0], ponto\_inicial[1], f(ponto\_inicial[0], ponto\_inicial[1]),

color='red', s=100, label='Ponto Inicial')

# Plota o vetor gradiente

ax.quiver(ponto\_inicial[0], ponto\_inicial[1], f(ponto\_inicial[0], ponto\_inicial[1]),

direcao\_descida[0], direcao\_descida[1], 0,

length=0.5, normalize=True, color='black', arrow\_length\_ratio=0.1,

label='Direção de Descida (-∇f)', linewidth=3)

ax.set\_title('Vetor Gradiente Apontando na Direção de Descida')

ax.set\_xlabel('x')

ax.set\_ylabel('y')

ax.set\_zlabel('f(x,y)')

ax.legend()

plt.savefig('grafico\_b.png', dpi=300, bbox\_inches='tight')

plt.show()

# --- Item (c): Rotina em Python para encontrar o mínimo ---

print("\nITEM (c): EXECUTANDO ALGORITMO COMPLETO")

print("=" \* 60)

ponto\_atual = np.copy(ponto\_inicial)

iteracao = 0

caminho = [ponto\_atual.copy()]

historico\_norma\_grad = []

historico\_funcao = []

print(f"Ponto inicial: [{ponto\_atual[0]:.3f}, {ponto\_atual[1]:.3f}]")

print(f"Critério de parada: Norma do Gradiente < {tolerancia}")

print("\nIniciando otimização...")

# Laço de iteração com o novo critério de parada

while True:

# Calcula o gradiente no ponto atual

gradiente = grad\_f(ponto\_atual[0], ponto\_atual[1])

# Calcula a norma do gradiente (critério de parada)

norma\_grad = np.linalg.norm(gradiente)

# Armazena histórico da iteração atual (antes de dar o passo)

historico\_norma\_grad.append(norma\_grad)

historico\_funcao.append(f(ponto\_atual[0], ponto\_atual[1]))

# Verifica a condição de parada

if norma\_grad < tolerancia:

break

# Passo do gradiente descendente para encontrar o novo ponto

ponto\_atual = ponto\_atual - lambda\_passo \* gradiente

# Atualiza o caminho e o contador de iterações

caminho.append(ponto\_atual.copy())

iteracao += 1

# Imprime o progresso periodicamente

if iteracao <= 10 or iteracao % 20 == 0:

print(f"Iteração {iteracao:3d}: Ponto = [{ponto\_atual[0]:8.5f}, {ponto\_atual[1]:8.5f}], "

f"f = {f(ponto\_atual[0], ponto\_atual[1]):10.6f}, Norma Grad. = {norma\_grad:.2e}")

ponto\_minimo = ponto\_atual

norma\_grad\_final = norma\_grad # Salva a última norma calculada

print(f"\nAlgoritmo convergiu após {iteracao} iterações!")

print(f"Ponto de mínimo: [{ponto\_minimo[0]:.6f}, {ponto\_minimo[1]:.6f}]")

print(f"Valor mínimo da função: {f(ponto\_minimo[0], ponto\_minimo[1]):.8f}")

print(f"Norma final do gradiente: {norma\_grad\_final:.2e}")

# --- Item (d): Plotar resultado final com caminho ---

print("\nITEM (d): PLOTANDO RESULTADO FINAL")

print("=" \* 60)

caminho = np.array(caminho)

fig = plt.figure(figsize=(15, 10))

# Subplot 1: Gráfico 3D com caminho

ax1 = fig.add\_subplot(221, projection='3d')

ax1.plot\_surface(X, Y, Z, cmap='viridis', alpha=0.6)

ax1.plot(caminho[:, 0], caminho[:, 1], f(caminho[:, 0], caminho[:, 1]),

'r-o', markersize=2, linewidth=2, label='Caminho da Descida')

ax1.scatter(ponto\_inicial[0], ponto\_inicial[1], f(ponto\_inicial[0], ponto\_inicial[1]),

color='green', s=100, label='Início', zorder=5)

ax1.scatter(ponto\_minimo[0], ponto\_minimo[1], f(ponto\_minimo[0], ponto\_minimo[1]),

color='red', s=150, label='Mínimo', zorder=5)

ax1.set\_title('Função com Caminho de Otimização')

ax1.set\_xlabel('x')

ax1.set\_ylabel('y')

ax1.set\_zlabel('f(x,y)')

ax1.legend()

# Subplot 2: Vista superior (contorno)

ax2 = fig.add\_subplot(222)

contour = ax2.contour(X, Y, Z, levels=20, cmap='viridis')

ax2.clabel(contour, inline=True, fontsize=8)

ax2.plot(caminho[:, 0], caminho[:, 1], 'r-o', markersize=3, linewidth=2,

label='Caminho de Otimização')

ax2.scatter(ponto\_inicial[0], ponto\_inicial[1], color='green', s=100, label='Início', zorder=5)

ax2.scatter(ponto\_minimo[0], ponto\_minimo[1], color='red', s=100, label='Mínimo', zorder=5)

ax2.set\_title('Vista Superior - Curvas de Nível')

ax2.set\_xlabel('x')

ax2.set\_ylabel('y')

ax2.legend()

ax2.grid(True, alpha=0.3)

# Subplot 3: Convergência da norma do gradiente

ax3 = fig.add\_subplot(223)

ax3.semilogy(range(len(historico\_norma\_grad)), historico\_norma\_grad, 'b-o', markersize=2)

ax3.set\_title('Convergência da Norma do Gradiente')

ax3.set\_xlabel('Iteração')

ax3.set\_ylabel('Norma do Gradiente (escala log)')

ax3.grid(True, which="both", ls="-", alpha=0.3)

# Subplot 4: Evolução do valor da função

ax4 = fig.add\_subplot(224)

ax4.plot(range(len(historico\_funcao)), historico\_funcao, 'g-o', markersize=2)

ax4.axhline(y=f(ponto\_minimo[0], ponto\_minimo[1]), color='r', linestyle='--',

label=f'Valor mínimo = {f(ponto\_minimo[0], ponto\_minimo[1]):.6f}')

ax4.set\_title('Evolução do Valor da Função')

ax4.set\_xlabel('Iteração')

ax4.set\_ylabel('f(x,y)')

ax4.legend()

ax4.grid(True, alpha=0.3)

plt.tight\_layout()

plt.savefig('resultado\_final\_norma\_grad.png', dpi=300, bbox\_inches='tight')

plt.show()

print("\n" + "=" \* 60)

print("RESUMO DOS RESULTADOS")

print("=" \* 60)

print(f"✓ Função: f(x,y) = xy·exp(-x²-y²)")

print(f"✓ Ponto inicial: (0.3, 1.2)")

print(f"✓ Tamanho do passo: α = 0.1")

print(f"✓ Critério de parada: Norma do Gradiente < {tolerancia}")

print(f"✓ Número de iterações: {iteracao}")

print(f"✓ Ponto de mínimo: ({ponto\_minimo[0]:.6f}, {ponto\_minimo[1]:.6f})")

print(f"✓ Valor mínimo: {f(ponto\_minimo[0], ponto\_minimo[1]):.8f}")

print(f"✓ Gráficos salvos como: grafico\_a.png, grafico\_b.png, resultado\_final\_norma\_grad.png")

print("=" \* 60)

Resultado do codigo  
  
============================================================

RESOLUÇÃO MANUAL - DUAS PRIMEIRAS ITERAÇÕES

============================================================

Ponto inicial: x₀ = 0.3, y₀ = 1.2

Tamanho do passo: α = 0.1

f(x₀, y₀) = 0.077953

--- ITERAÇÃO 1 ---

∇f(x₀, y₀) = [0.213071, -0.122126]

x₁ = x₀ - α \* ∂f/∂x = 0.3 - 0.1 \* 0.213071 = 0.278693

y₁ = y₀ - α \* ∂f/∂y = 1.2 - 0.1 \* -0.122126 = 1.212213

f(x₁, y₁) = 0.071911

--- ITERAÇÃO 2 ---

∇f(x₁, y₁) = [0.217947, -0.115021]

x₂ = x₁ - α \* ∂f/∂x = 0.278693 - 0.1 \* 0.217947 = 0.256898

y₂ = y₁ - α \* ∂f/∂y = 1.212213 - 0.1 \* -0.115021 = 1.223715

f(x₂, y₂) = 0.065831

============================================================

ITEM (a): PLOTANDO FUNÇÃO E PONTO INICIAL

============================================================

ITEM (b): PLOTANDO VETOR GRADIENTE

============================================================

Gradiente no ponto inicial: [ 0.2130711 -0.12212612]

Direção de descida (-∇f): [-0.2130711 0.12212612]

ITEM (c): EXECUTANDO ALGORITMO COMPLETO

============================================================

Ponto inicial: [0.300, 1.200]

Critério de parada: Norma do Gradiente < 1e-05

Iniciando otimização...

Iteração 1: Ponto = [ 0.27869, 1.21221], f = 0.071911, Norma Grad. = 2.46e-01

Iteração 2: Ponto = [ 0.25690, 1.22371], f = 0.065831, Norma Grad. = 2.46e-01

Iteração 3: Ponto = [ 0.23466, 1.23445], f = 0.059729, Norma Grad. = 2.47e-01

Iteração 4: Ponto = [ 0.21200, 1.24435], f = 0.053617, Norma Grad. = 2.47e-01

Iteração 5: Ponto = [ 0.18899, 1.25339], f = 0.047505, Norma Grad. = 2.47e-01

Iteração 6: Ponto = [ 0.16565, 1.26151], f = 0.041402, Norma Grad. = 2.47e-01

Iteração 7: Ponto = [ 0.14202, 1.26867], f = 0.035314, Norma Grad. = 2.47e-01

Iteração 8: Ponto = [ 0.11816, 1.27485], f = 0.029244, Norma Grad. = 2.46e-01

Iteração 9: Ponto = [ 0.09410, 1.28001], f = 0.023196, Norma Grad. = 2.46e-01

Iteração 10: Ponto = [ 0.06989, 1.28414], f = 0.017170, Norma Grad. = 2.46e-01

Iteração 20: Ponto = [-0.17179, 1.26690], f = -0.042450, Norma Grad. = 2.44e-01

Iteração 40: Ponto = [-0.53874, 1.00252], f = -0.147890, Norma Grad. = 1.95e-01

Iteração 60: Ponto = [-0.66981, 0.79099], f = -0.180948, Norma Grad. = 6.83e-02

Iteração 80: Ponto = [-0.69913, 0.72636], f = -0.183781, Norma Grad. = 1.63e-02

Iteração 100: Ponto = [-0.70538, 0.71133], f = -0.183932, Norma Grad. = 3.61e-03

Iteração 120: Ponto = [-0.70673, 0.70803], f = -0.183939, Norma Grad. = 7.87e-04

Iteração 140: Ponto = [-0.70703, 0.70731], f = -0.183940, Norma Grad. = 1.71e-04

Iteração 160: Ponto = [-0.70709, 0.70715], f = -0.183940, Norma Grad. = 3.71e-05

Algoritmo convergiu após 177 iterações!

Ponto de mínimo: [-0.707102, 0.707119]

Valor mínimo da função: -0.18393972

Norma final do gradiente: 9.36e-06

ITEM (d): PLOTANDO RESULTADO FINAL

============================================================

============================================================

RESUMO DOS RESULTADOS

============================================================

✓ Função: f(x,y) = xy·exp(-x²-y²)

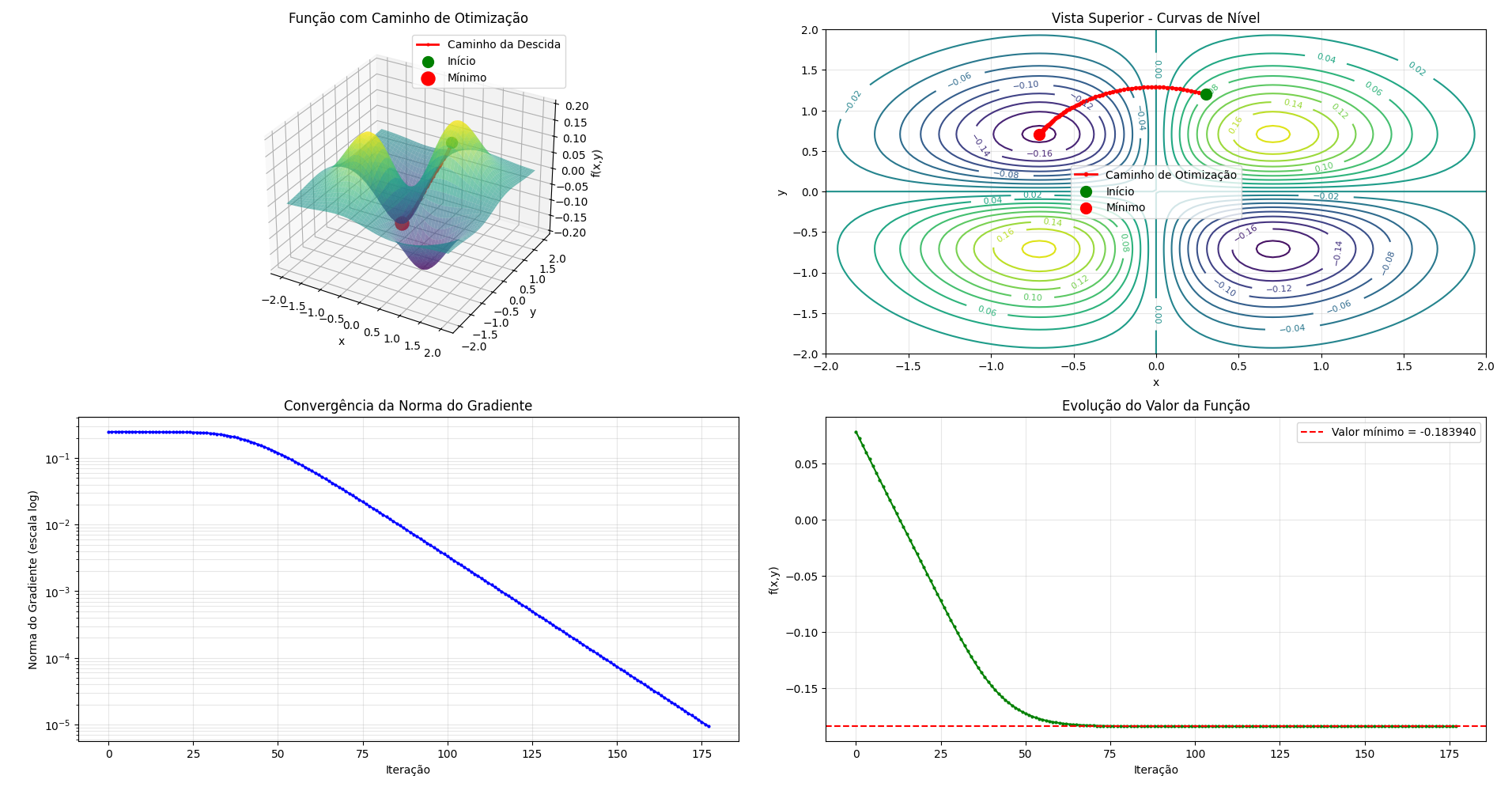
✓ Ponto inicial: (0.3, 1.2)

✓ Tamanho do passo: α = 0.1

✓ Critério de parada: Norma do Gradiente < 1e-05

✓ Número de iterações: 177

✓ Ponto de mínimo: (-0.707102, 0.707119)

✓ Valor mínimo: -0.18393972