Universidade Federal de Minas Gerais
Escola de Engenharia
Departamento de Engenharia Elétrica
ELE077 Otimização Não-Linear

Trabalho Computacional I
Aplicação de Métodos Determinísticos Irrestritos

Daniel Augusto Castro de Paula - 2022060550

Questão 1) a) Devemos escrever a seguinte equação (1) em formato de polinômios

$$R(\mathbf{X}) = \frac{\mathbf{X}^T[K]\mathbf{X}}{\mathbf{X}^T[M]\mathbf{X}} \tag{1}$$

onde

$$[K] = k\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}, \qquad [M] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathbf{X} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} \tag{2}$$

para isso teremos o numerador $X^T[K]X$ e o denominador $X^T[M]X$. Ao realizar a multiplicação das matrizes obtemos:

$numerador = X^TKX = 2X_1^2 - 2X_1X_2 + 2X_2 - 2X_2X_3 + X_3^2$ e
$denominador = X^TMX = X_1^2 + X_2^2 + X_3^2$

No código realizamos essa representação da seguinte forma

```python
# Matrizes K e M
def get_matrizes(k=1):
    K = k * np.array([[2, -1, 0],
                      [-1, 2, -1],
                      [0, -1, 1]])
    M = np.array([[1, 0, 0],
                  [0, 1, 0],
                  [0, 0, 1]])
    return K, M

# Definição da função objetivo R(X)
def R(X):
    K, M = get_matrizes(k=1)
    X = np.array(X).reshape(-1, 1)
    numerator = (X.T @ K @ X)[0, 0]  # X^T K X
    denominator = (X.T @ M @ X)[0, 0]  # X^T M X
    return numerator / denominator
```

b) Para a minimização de R(x) no arquivo questao1.py foi utilizado o método BFGS. O critério de parada utilizado foi que o método deve parar quando a mudança no valor da função objetivo ou no gradiente for menor que menor que 10^{-5} , incluímos também o máximo de 100 iterações. Utilizando o método da biblioteca scipy e passando o gradiente de forma analitica calculado manualmente da função objetivo o resultado obtido foi

Ponto ótimo: [0.60837425 1.09624044 1.3669776]
Valor mínimo de R(X): 0.19806226425150814
Número de Iterações: 4
Número de Avaliações da Função: 5

Sem especificar a função do gradiente analiticamente, ou seja, retirando a função grad_R dos parâmetros da chamada da função do scipy, o resultado foi

Ponto ótimo: [0.60837423 1.09624041 1.36697757]
Valor mínimo de R(X): 0.19806226425144058
Número de Iterações: 4
Número de Avaliações da Função: 20

No arquivo questao1Comparacoes.py executamos os demais métodos com os mesmo critérios de parada, os resultados obtidos foram

Método do Gradiente
Ponto Ótimo: [0.58721634 1.05812709 1.31946017]
Valor Ótimo: 0.19806226419703196
Número de Iterações: 13
Número de Avaliações da Função: 512

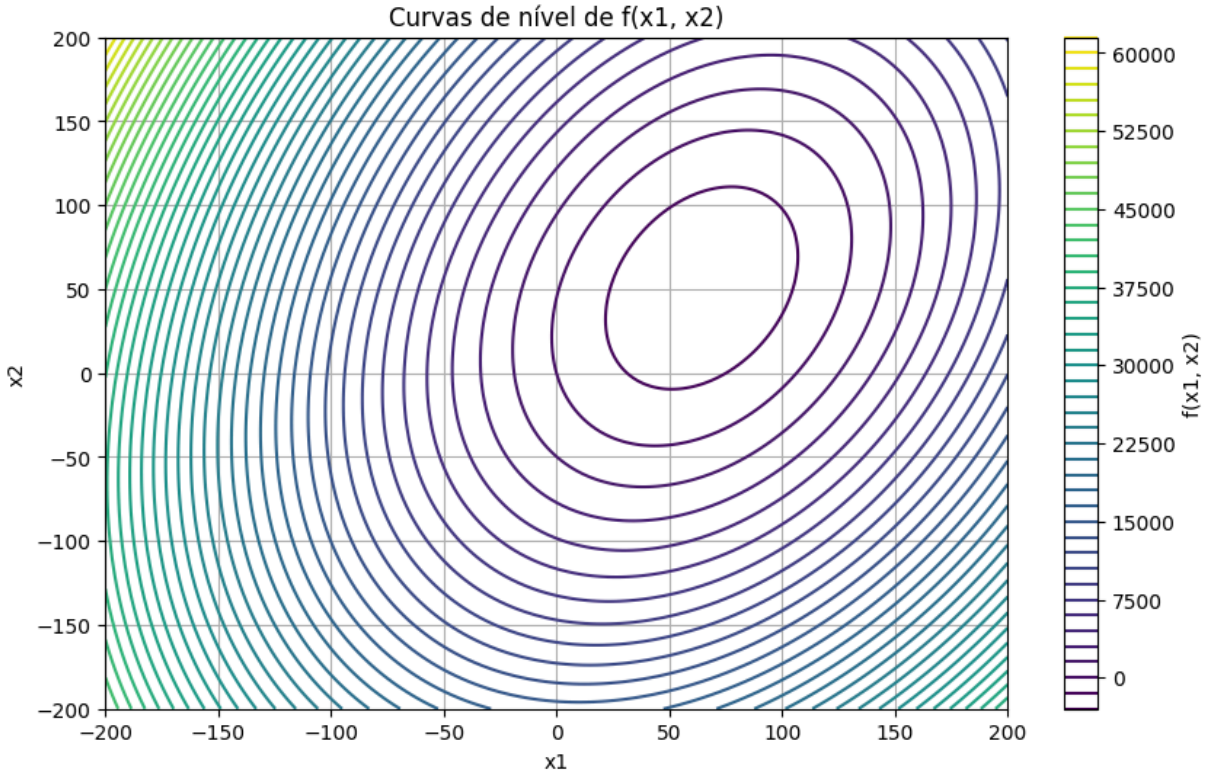
Método: Newton Modificado
Ponto Ótimo: [0.58867465 1.06075506 1.32273992]
Valor Mínimo: 0.19806226419516176
Número de Iterações: 5
Número de Avaliações da Função: 6

Método: Gradiente-Conjugado
Ponto Ótimo: [0.62081068 1.11865088 1.39491309]
Valor Mínimo: 0.19806226432082236
Número de Iterações: 8
Número de Avaliações da Função: 11

Percebe-se que para essas condições o método BFGS se demonstrou o mais eficiente já que vimos que é um método que funciona bem para problemas de dimensão baixa aproximando a Hessiana de forma eficiente e convergindo rapidamente.

C) Ao executar o algoritmo para outros pontos iniciais o valor do ponto ótimo mudou porém o valor mínimo para a função R(x) continuou o mesmo, nesse caso dizemos que ela possui múltiplos mínimos locais que se encontram na região do mínimo global.

2) a) A função $f(x_1, x_2) = 0.6382x_1^2 + 0.3191x_2^2 - 0.2809x_1x_2 - 67.906x_1 - 14.29x_2$ é uma função convexa unimodal que pode ser representada pelas seguintes curvas de nível



Os algoritmos adequados para determinar o mínimo dessa função seriam o método de Newton, método de Quasi-Newton (BFGS), método do gradiente conjugado dentre alguns outros. O melhor para o nosso caso é o método de Newton, já que nossa função é quadrática e vai convergir em 1 iteração.

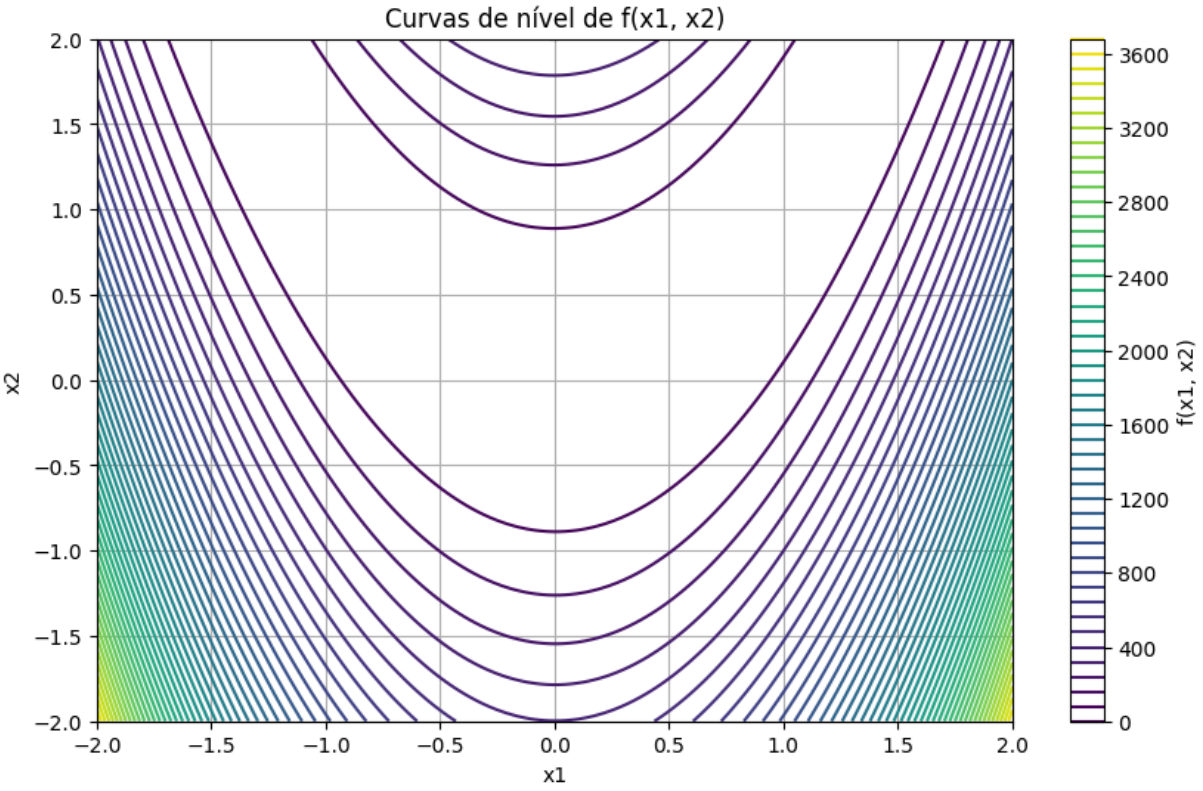
b) Utilizando o método de Newton com o auxílio da hessiana e do gradiente já calculados analiticamente convergimos para o resultado com 1 iteração e avaliação, encontrando o seguinte resultado

Resultado do Método de Newton:
Ponto ótimo: [64.36329702 50.72022898]
Valor mínimo de f(x1, x2): -2547.7230598621304

Fazendo um teste utilizando o BFGS com o critério de parada quando a mudança no valor da função objetivo ou no gradiente for menor que menor que 10^{-6} e máximo de 100 iterações, obtemos o seguinte resultado

Resultado da otimização:
Ponto ótimo: [64.36329702 50.72022898]
Valor mínimo de f(x1, x2): -2547.72305986213
Número de Iterações: 7
Número de Avaliações da Função: 10

3) a) O gráfico de curva de nível da função min f(x) = 100 $(x_2 - x_1^2)^2 + (1-x_1)^2$ é dado pela seguinte imagem

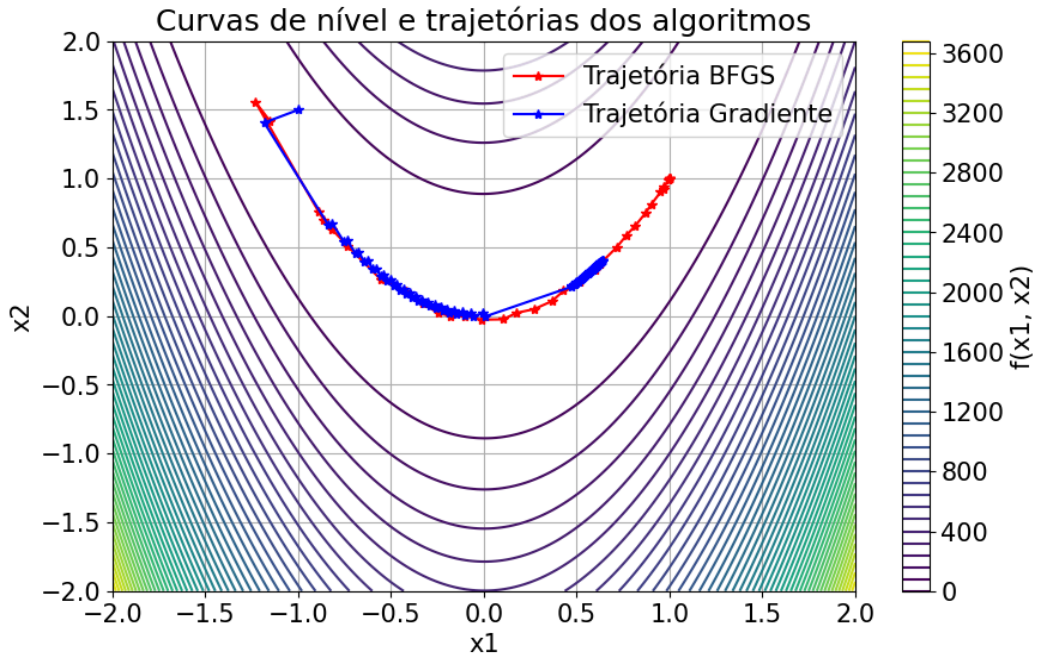


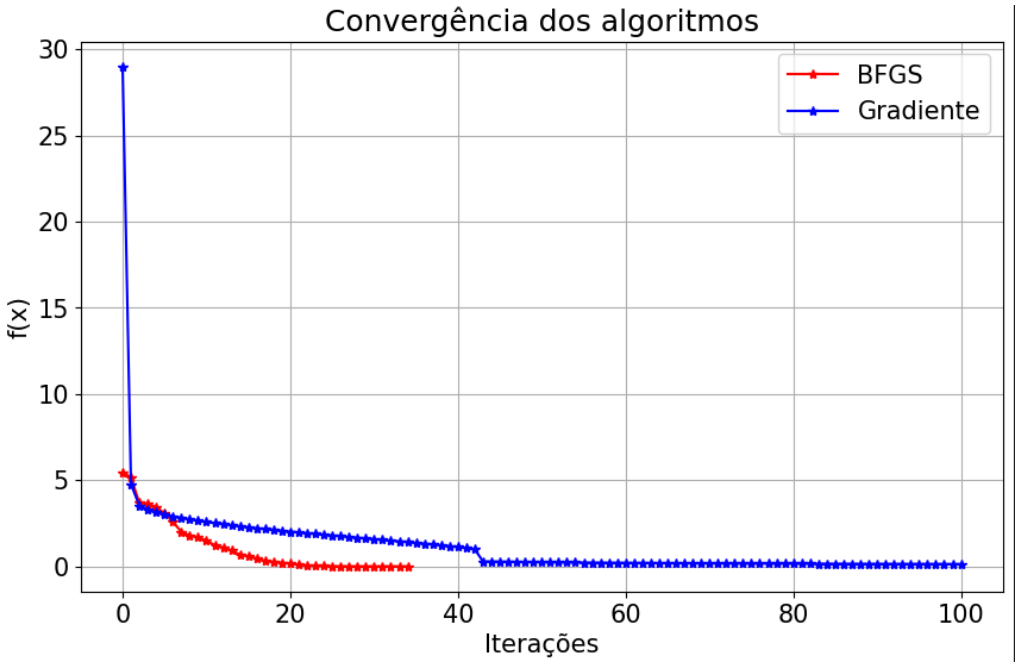
b) Implementamos a otimização dessa função através do método BFGS e do método do gradiente. Para ambos os métodos utilizamos o critério de parada quando a mudança no valor da função objetivo ou no gradiente for menor que menor que 10^{-6} com máximo de 100 iterações. Para o gradiente o valor do passo foi de 10^{-8}. Os resultados obtidos foram

Resultado da otimização com BFGS:
Ponto ótimo: [1. 1.00000001]
Valor mínimo de f(x1, x2): 1.0796552370664605e-17
Número de Iterações: 35
Número de Avaliações da Função: 49

Resultado da otimização com o Método do Gradiente:
Ponto ótimo: [0.63968531 0.40814893]
Valor mínimo de f(x1, x2): 0.12993658448419237
Número de Iterações: 100
Número de Avaliações da Função: 5477

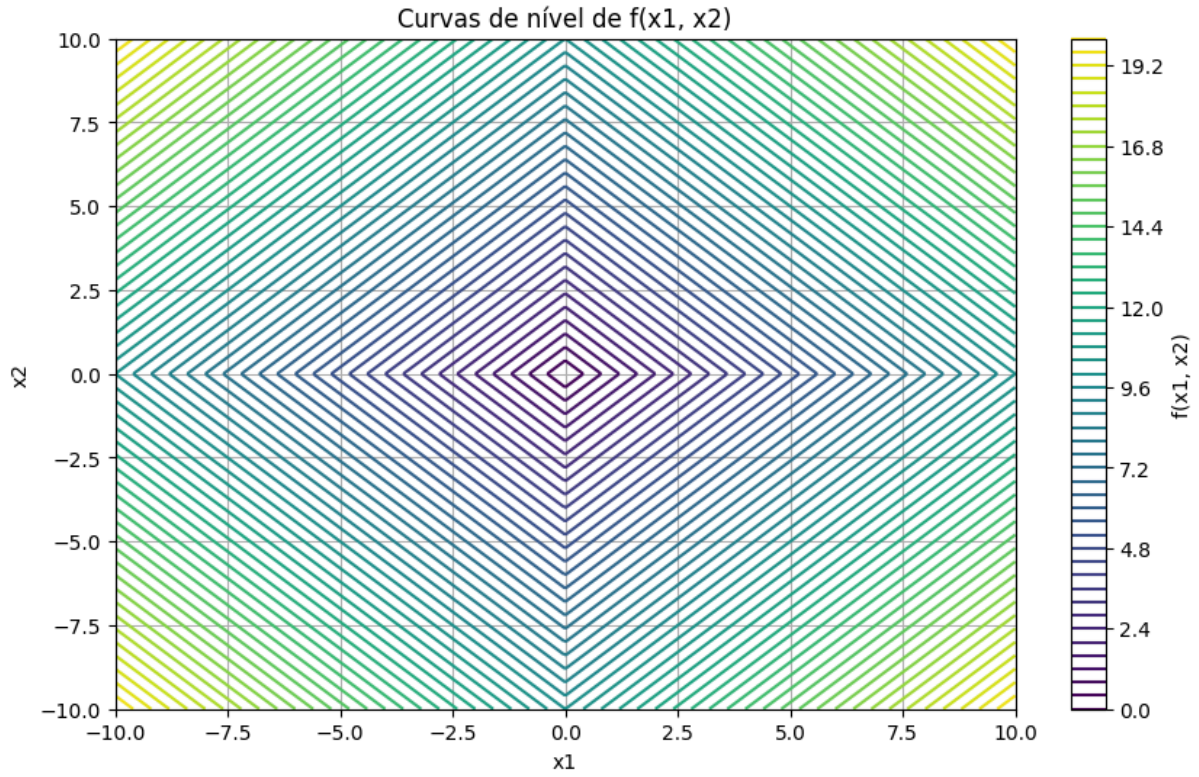
A seguir plotamos 2 gráficos comparativos das trajetórias e convergência desses algoritmos





c) O método BFGS mostrou-se mais eficiente e preciso na otimização da função de Rosenbrock, alcançando o mínimo global com um valor de função objetivo extremamente baixo (apenas 35 iterações e 49 avaliações). Por outro lado, o método do gradiente, apesar de realizar 100 iterações e 5477 avaliações, não conseguiu convergir ao mínimo global utilizando os critérios de parada especificados. Essa diferença se deve à natureza do BFGS, que utiliza aproximações da matriz Hessiana para ajustar automaticamente as direções e tamanhos dos passos, enquanto o gradiente depende de um passo fixo com alpha de 10^{-8} e uma busca unidimensional por seção áurea, por isso ele é mais lento e ineficiente para problemas com vales estreitos e alta curvatura, como o de Rosenbrock. Podemos dizer que ele fica “preso” em certos momentos. Isso evidencia que o BFGS é mais adequado para problemas complexos e não lineares.

4) a) Para a função min f(x) = $|x_1|$ + $|x_2|$ o gráfico das curvas de nível para o intervalo -10 e 10 ficou assim



Por se tratar de uma função não diferenciável (tem quinas) devemos utilizar métodos sem derivadas como por exemplo, Nelder-Mead , Hooke-Jeeves e amostragens aleatórias.

b) Aplicamos o método BFGS e o Nelder-Mead ambos com critério de parada quando a mudança no valor da função objetivo ou no gradiente for menor que menor que 10^{-6} e máximo de 100 iterações. Obtivemos os seguintes resultados

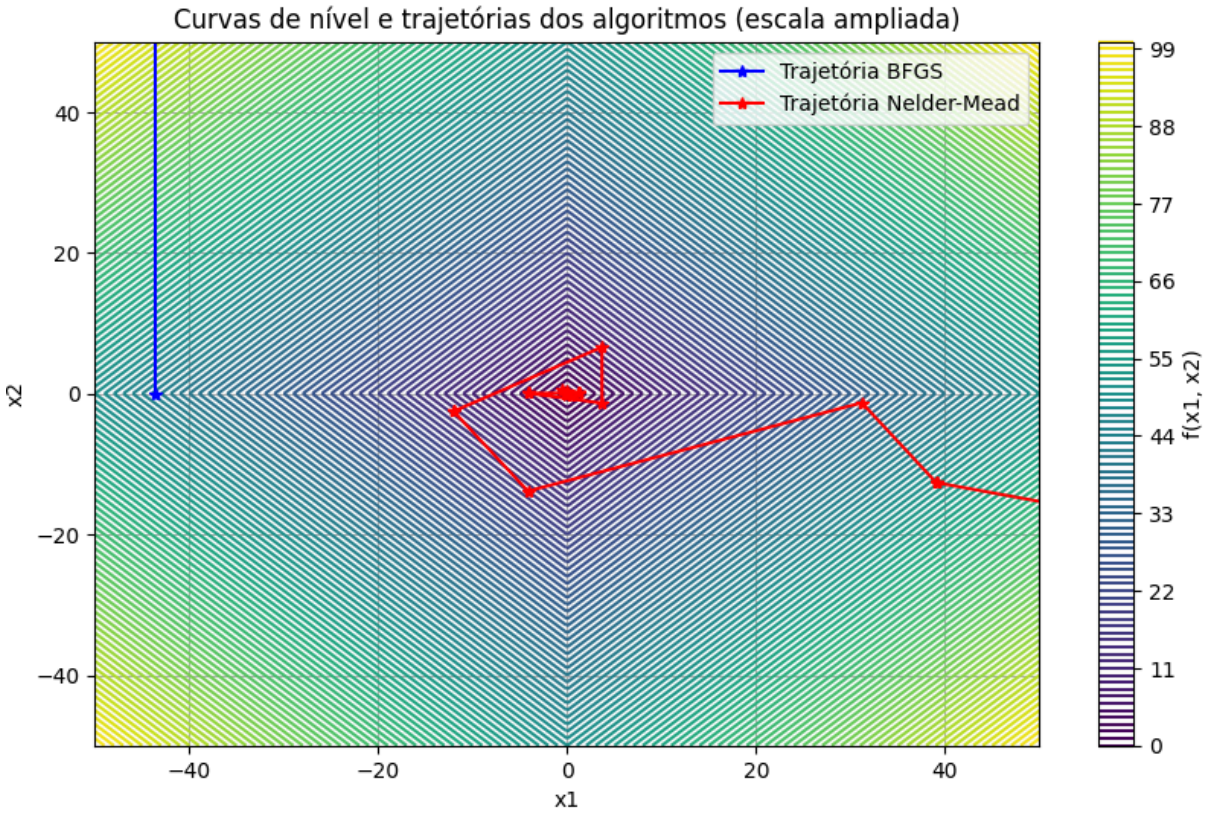
Resultado da otimização com BFGS:
Ponto ótimo: [-4.35346465e+01 -1.58161129e-09]
Valor mínimo de f(x1, x2): 43.534646510003256
Número de Iterações: 2
Número de Avaliações da Função: 345

Resultado da otimização com Nelder-Mead:
Ponto ótimo: [2.71312881e-08 2.88775093e-07]
Valor mínimo de f(x1, x2): 3.1590638139066184e-07
Número de Iterações: 87
Número de Avaliações da Função: 168

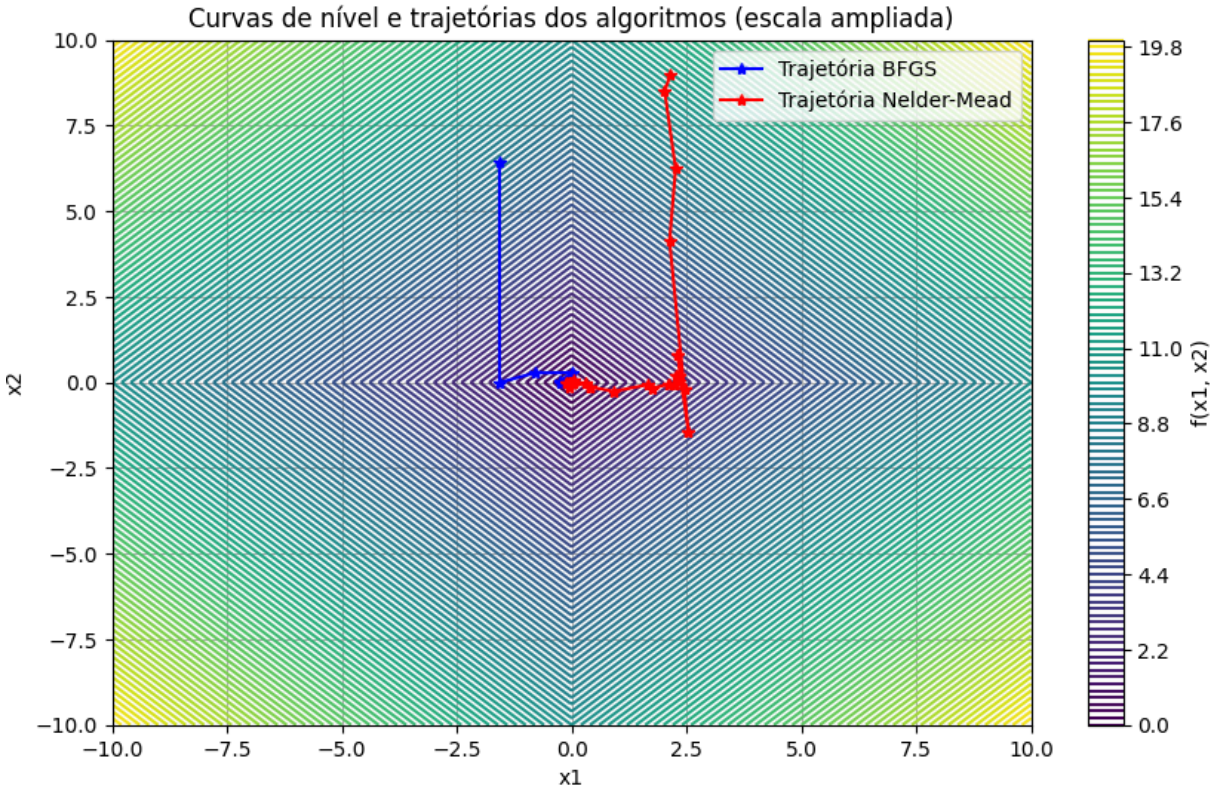
Analisando os resultados da otimização percebemos grandes diferenças entre os métodos BFGS e Nelder-Mead em termos de eficiência e precisão. O método BFGS convergiu rapidamente, alcançando o ponto “ótimo” em apenas 2 iterações No entanto, o número elevado de avaliações da função 345 mostra que o método realizou muitos cálculos internos para avaliar a função e suas derivadas. O método Nelder-Mead, apesar de exigir mais

iterações e menos avaliações da função, encontrou um ponto próximo ao mínimo global real, com um valor mínimo próximo de zero. Esse comportamento reflete a adequação do Nelder-Mead para funções não diferenciáveis, enquanto o BFGS que é mais eficiente para funções diferenciáveis, enfrentou dificuldades devido à natureza da função objetivo, resultando em um mínimo inadequado. Importante destacar que o BFGS só teve algum resultado pois utiliza aproximações numéricas para o gradiente.

Podemos comprovar essa hipótese pelo gráfico da trajetória ampliando um pouco os valores da escala, observamos que o BFGS ficou longe do ponto mínimo global



c) Utilizando o ponto inicial (2,10) para os 2 algoritmos obtivemos os seguintes resultados de trajetória



Nesse novo caso, o método BFGS, mesmo não sendo adequado para funções não diferenciáveis, conseguiu atingir um valor muito próximo do mínimo global. Apesar disso, ele exigiu um número muito alto de avaliações, tendo um custo computacional elevado, concluindo novamente que o Nelder-Mead é mais robusto e confiável para esses casos.