

# Lista 00

MEC 2403 - Otimização e Algoritmos para Engenharia Mecânica

**Pedro Henrique Cardoso Paulo**

pedrorjpaulo.phcp@gmail.com

Professor: Ivan Menezes



Departamento de Engenharia Mecânica  
PUC-RJ Pontifícia Universidade Católica do Rio de Janeiro  
março, 2023

# 1 Questão 01

## 1.1 Enunciado

Calcular o gradiente e a matriz Hessiana da função  $f(x_1, x_2, x_3)$ , dada pela equação eq. (1)

$$f(x_1, x_2, x_3) = 3x_1^3x_2^2x_3 - 6x_1x_3^4\log x_2 + x_1^{-1}x_2^3 - x_1^2\sqrt{x_2} \quad (1)$$

onde  $\log x$  representa o logaritmo neperiano.

## 1.2 Solução

A expressão para o gradiente ( $\nabla f$ ) e para a Hessiana ( $\mathbf{H}$ ) são apresentadas, respectivamente, nas equações eq. (2) e eq. (3)

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} \quad (2)$$

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 x_2} & \frac{\partial^2 f}{\partial x_1 x_3} \\ \frac{\partial^2 f}{\partial x_1 x_2} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 x_3} \\ \frac{\partial^2 f}{\partial x_1 x_3} & \frac{\partial^2 f}{\partial x_2 x_3} & \frac{\partial^2 f}{\partial x_3^2} \end{bmatrix} \quad (3)$$

Vale ressaltar que a formulação da Hessiana já faz uso do da igualdade matemática de derivadas segundas expressa na equação eq. (4)

$$\frac{\partial^2 f}{\partial x_i x_j} = \frac{\partial^2 f}{\partial x_j x_i} \quad (4)$$

Dessa forma, o problema da determinação do gradiente e da hessiana se resume ao problema do cálculo das derivadas de primeira e segunda ordem da função  $f(x_1, x_2, x_3)$ . Começando pelas derivadas de primeira ordem para determinação do gradiente, estas são expressas pelas equações eqs. (5)–(7)

$$\frac{\partial f}{\partial x_1} = 9x_1^2x_2^2x_3 - 6x_3^4\log x_2 - x_1^{-2}x_2^3 - 2x_1\sqrt{x_2} \quad (5)$$

$$\frac{\partial f}{\partial x_2} = 6x_1^3x_2x_3 - 6x_1x_2^{-1}x_3^4 + 3x_1^{-1}x_2^2 - \frac{x_1^2}{2\sqrt{x_2}} \quad (6)$$

$$\frac{\partial f}{\partial x_3} = 3x_1^3x_2^2 - 24x_1x_3^3\log x_2 \quad (7)$$

Já as derivadas segundas, necessárias para a determinação da Hessiana, são dadas pelas equações eqs. (8)–(13)

$$\frac{\partial^2 f}{\partial x_1^2} = 18x_1x_2^2x_3 + 2x_1^{-3}x_2^3 - 2\sqrt{x_2} \quad (8)$$

$$\frac{\partial^2 f}{\partial x_2^2} = 6x_1^3x_3 + 6x_1x_2^{-2}x_3^4 + 6x_1^{-1}x_2 + \frac{x_1^2}{4x_2\sqrt{x_2}} \quad (9)$$

$$\frac{\partial^2 f}{\partial x_3^2} = -72x_1x_3^2\log x_2 \quad (10)$$

$$\frac{\partial^2 f}{\partial x_1 x_2} = 18x_1^2x_2x_3 - 6x_2^{-1}x_3^4 - 3x_1^{-2}x_2^2 - \frac{x_1}{\sqrt{x_2}} \quad (11)$$

$$\frac{\partial^2 f}{\partial x_1 x_3} = 9x_1^2x_2^2 - 24x_3^3\log x_2 \quad (12)$$

$$\frac{\partial^2 f}{\partial x_2 x_3} = 6x_1^3x_2 - 24x_1x_2^{-1}x_3^3 \quad (13)$$

## 2 Questão 02

### 2.1 Enunciado

Classificar a matriz abaixo quanto à sua positividade:

$$\mathbf{A} = \begin{bmatrix} 3 & -2 & 1 \\ -2 & 4 & -3 \\ 1 & -3 & 2 \end{bmatrix}$$

### 2.2 Solução

A determinação da positividade da matriz será feita por meio de seus autodeterminantes. Caso os 3 autodeterminantes da matriz sejam positivos, sabe-se que ela será positiva definida (ou semi-definida caso algum seja zero). Caso eles alternem entre valores positivos e negativos, sabe-se que ela será negativa definida. Caso contrário, nada pode-se afirmar. Calculando o primeiro:

$$| 3 | = 3$$

Calculando o segundo:

$$\begin{vmatrix} 3 & -2 \\ -2 & 4 \end{vmatrix} = 8$$

Calculando o terceiro:

$$\begin{vmatrix} 3 & -2 & 1 \\ -2 & 4 & -3 \\ 1 & -3 & 2 \end{vmatrix} = -3$$

Como nenhum dos padrões descritos foi respeitado, a matriz não é nem positiva definida e nem negativa definida, ou seja, a matriz apresenta autovalores positivos e negativos.

## 3 Questão 03

### 3.1 Enunciado

Determinar a expansão em série de Taylor, em torno do ponto  $x = 1$ , da função  $f(x) = e^{2x}$ . Em seguida, usando o MATLAB ou Python, plotar os gráficos da função  $f$  e de suas respectivas aproximações (polinômios) de ordens 0, 1, 2 e 3.

### 3.2 Solução

Para a função em questão, suas derivadas são calculadas de acordo com a regra de derivação da função exponencial. Assim temos que a derivada  $n$ -ésima da função  $f(x)$  criada é dada pela equação eq. (14)

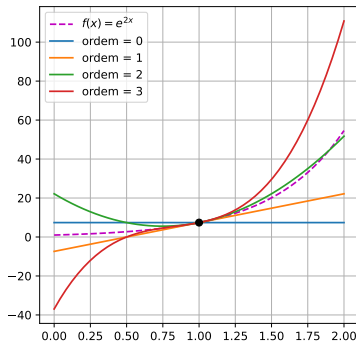
$$\frac{d^n f}{dx^n} = 2^n e^{2x} \quad (14)$$

Pela definição da série de Taylor, para uma expansão ao redor de  $x = 1$ , temos que a expansão da função seria a dada pela equação eq. (15)

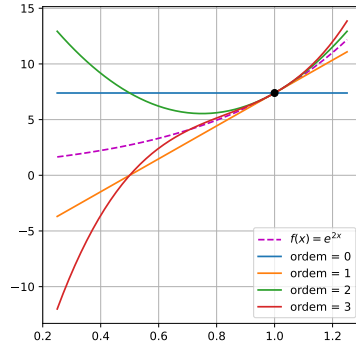
$$f(x) \cong \sum_{n=0}^{\infty} 2^n e^2 (x-1)^n \quad (15)$$

O primeiro passo é implementar em Python as funções `f` e `taylor_f` que representam, respectivamente, a função  $f$  definida na questão e seu polinômio de Taylor de ordem arbitrária.

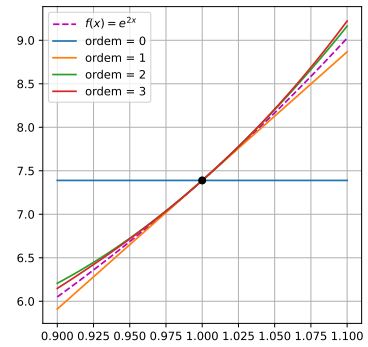
```
def f(x):  
    return np.exp(2*x)  
  
def taylor_f(x, order):  
    f_approx = 0  
    for n in range(order+1):  
        f_approx += (x-1)**n * (2**n) * np.exp(2)  
    return f_approx
```



(a)  $x \in [0, 2]$



(b)  $x \in [0.25, 1.25]$



(c)  $x \in [0.9, 1.1]$

**Figura 1:** Função  $f(x) = e^{2x}$  e seus polinômios de Taylor

As funções são implementadas com funções do módulo `numpy` de modo a garantir a vetorização. Uma vez tendo as funções implementadas, podemos plotar gráficos das aproximações desejadas para vizinhanças do ponto  $x = 1$  usando o seguinte código:

```
orders = [0, 1, 2, 3] # Ordens da aproximacao
points = 50 # Pontos de x considerados
x_min = 0 # Minimo x
x_max = 2 # Maximo x
x_values = np.linspace(x_min, x_max, points)
fig, ax = plt.subplots(1,1, figsize=(5, 5))
f_real = f(x_values)
ax.plot(x_values, f_real, 'm--', label='$f(x) = e^{\{2x\}}$')
for n in orders:
    f_approx = taylor_f(x_values, n)
    ax.plot(x_values, f_approx, label=f'ordem = {n}')

ax.plot(1, f(1), 'ko')
ax.legend()
ax.grid()
```

A fig. 1 apresenta alguns plots com diferentes intervalos de  $x$  para a análise da aproximação dos polinômios de Taylor. É possível ver que, para ordens maiores, a aproximação tende a ficar melhor na vizinhança do ponto expandido, mas isso não necessariamente é verdade conforme nos afastamos dele.

## 4 Questão 04

### 4.1 Enunciado

Seja a função  $f(x_1, x_2)$  dada pela equação eq. (16)

$$f(x_1, x_2) = x_1^3 + 2x_1x_2^2 - x_2^3 - 20x_1 \quad (16)$$

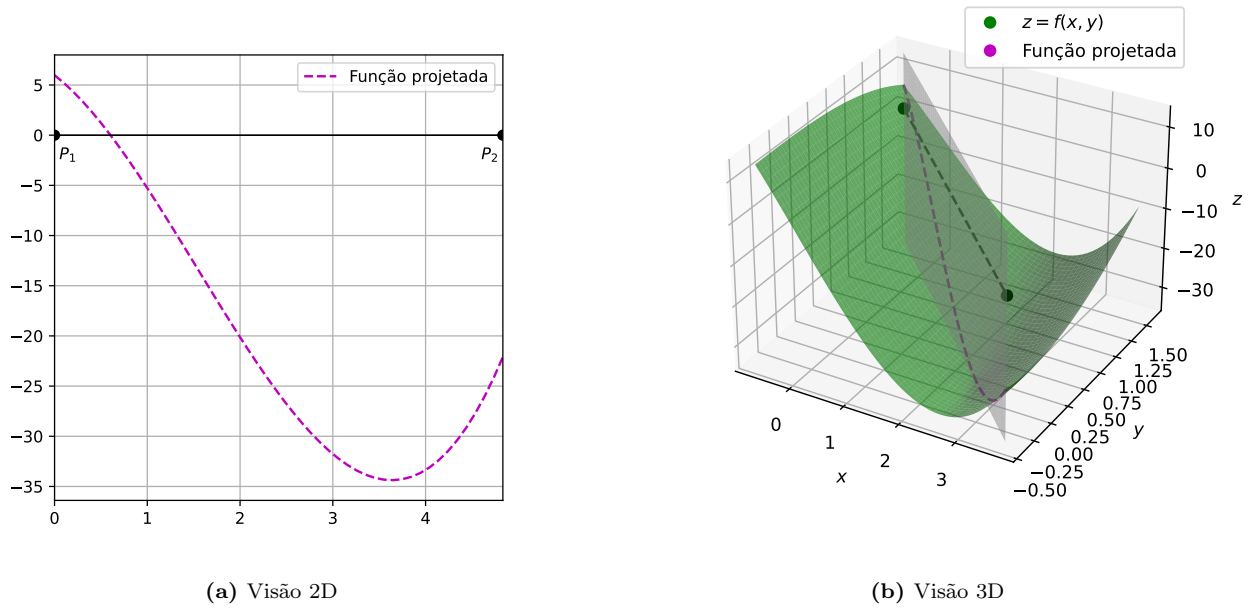
Desenhar o gráfico da curva contida em um plano Cartesiano, cujo eixo das abscissas corresponde à reta que passa pelos pontos  $P_1 = (-0.7, 1.6)$  e  $P_2 = (3.7, -0.4)$ , a origem desse eixo é no ponto  $P_1$  e o eixo das ordenadas corresponde aos valores da função  $f$ . Utilizar o MATLAB ou Python e considerar apenas o trecho entre os pontos  $P_1$  e  $P_2$ .

### 4.2 Solução

De forma similar ao feito na Questão 03 (seção 3), nossa solução começa por meio da implementação da função Python que representa a equação que desejamos estudar. Essa função receberá um vetor 1D que tem as coordenadas do ponto no plano  $xy$ .

```
def f2(P):
    return P[0]**3 + 2*P[0]*(P[1]**2) - P[1]**3 - 20*P[0]
```

A determinação da equação da reta que passa pelos pontos  $P_1$  e  $P_2$ , por sua vez, será feita vetorialmente por meio da equação paramétrica eq. (17)



**Figura 2:** Plot da projeção da função no plano da reta

$$\mathbf{P} = \mathbf{P}_1 + \alpha \hat{\mathbf{v}}, \quad \alpha \in [0, \|\mathbf{v}\|] \quad (17)$$

onde o vetor  $\mathbf{v} = \mathbf{P}_2 - \mathbf{P}_1$ . É importante notar que, no espaço 3D, essa reta determina, na verdade, um plano paralelo ao eixo vertical, de modo que o objetivo da questão é determinar a interseção entre esse plano e a superfície definida na equação eq. (16). Em Python, o plot dessa interseção pode ser feito por meio do seguinte código:

```
P1 = np.array([-0.7, 1.6])
P2 = np.array([3.7, -0.4])
v = P2 - P1
v_norm = np.linalg.norm(v)
v_hat = v / v_norm

points = 50
x_values = np.linspace(0, v_norm, points)
y_values = f2(P1.reshape(-1,1) + v_hat.reshape(-1,1)*x_values.reshape(1,-1))
fig, ax = plt.subplots(1,1, figsize=(5, 5))
ax.plot(x_values, y_values, 'm--', label='Funcao projetada')
ax.hlines(0, 0, v_norm, 'k', linewidth=1.0)
ax.set_xlim(0, v_norm)
ax.plot(0, 0, 'ko')
ax.text(0.05, -2.5, '$P_1$')
ax.plot(v_norm, 0, 'ko')
ax.text(4.6, -2.5, '$P_2$')
ax.legend()
ax.grid()
```

A visualização 2D dessa interseção, junto com uma representação 3D do problema é apresentada na fig. 2.