# Projeto 3 de Cálculo 2 Honors
# Diferenças finitas

Prof. Lucas Pedroso

$2^{\underline{o}}$ semestre de 2024

**Última atualização:** 09/11 (confira no final do arquivo a lista com as modificações desde a publicação)

Este arquivo trata do achievement de diferenças finitas para o Projeto 3.

## 1 Motivação teórica

Esta seção explica de onde vêm as fórmulas que usaremos. Por simplicidade de notação, considere que a variável de trabalho é $x = [x_1, x_2, ..., x_n]^T$, Considere $e_i$ como a $i-$ésima coluna da matriz identidade $n \times n$, ou seja, $e_1 = [1, 0, 0, ..., 0]^T$, $e_2 = [0, 1, 0, ..., 0]^T$ e assim por diante. Por exemplo, suponha que $n = 4$ (ou seja, o problema estará em $\mathbb{R}^4$). Só pra deixar clara uma notação que usaremos adiante, temos que se $h$ é uma constante

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad x + he_1 = \begin{bmatrix} x_1 + h \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad x + he_2 = \begin{bmatrix} x_1 \\ x_2 + h \\ x_3 \\ x_4 \end{bmatrix},$$

$$x + he_3 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 + h \\ x_4 \end{bmatrix}, \quad x + he_4 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 + h \end{bmatrix}.$$

### 1.1 Aproximação para as derivadas de primeira ordem

Essa vem diretamente do que vimos na aula para uma variável. Na ocasião, tínhamos a fórmula de diferenças centradas

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}.$$

Assim, as fórmulas para as derivadas de primeira ordem são

$$\frac{\partial f}{\partial x_j} \approx \frac{f(x+he_j) - f(x+he_j)}{2h}, \quad j = 1, ..., n. \tag{1}$$

Note que pra derivarmos em relação a $x_j$, aplicamos a mesma fórmula de uma variável, porém fazendo $x_j$ assumir $x_j + h$ e $x_j - h$ e mantendo fixas todas as outras variáveis. Por exemplo, se $n = 3$, temos

$$\frac{\partial f}{\partial x_1} \approx \frac{f(x+he_1) - f(x-he_1)}{2h} = \frac{f(x_1+h, x_2, x_3) - f(x_1-h, x_2, x_3)}{2h}.$$

O análogo vale para $\dfrac{\partial f}{\partial x_2}$ e $\dfrac{\partial f}{\partial x_3}$.

## 1.2 Aproximação para as derivadas de segunda ordem

Nesse caso, há fórmulas para derivadas puras e para derivadas mistas.

### 1.2.1 Derivadas puras

Para calcular $f_{x_j x_j}$, tenhamos em mente que computar derivada segunda implica derivar duas vezes. Derivando $f_{x_j}$ em relação a $x_j$ por diferenças finitas avançadas, temos que

$$f_{x_j x_j}(x) \approx \frac{f_{x_j}(x+he_j) - f_{x_j}(x)}{h}.$$

Agora, aproximando $f_{x_j}(x+he_j)$ e $f_{x_j}(x)$ por diferenças atrasadas nos dá

$$f_{x_j x_j}(x) \approx \frac{\dfrac{f(x+he_j) - f(x)}{h} - \dfrac{f(x) - f(x-he_j)}{h}}{h}.$$

Com isso, obtemos a expressão

$$\frac{\partial^2 f}{\partial x_j^2} \approx \frac{f(x+he_j) - 2f(x) + f(x-he_j)}{h^2}. \tag{2}$$

### 1.2.2 Derivadas mistas

Para calcular $f_{x_i x_j}$ com $i \neq j$, vamos aplicar diferenças finitas centradas pra derivar em relação a $x_j$ a função $f_{x_i}$, ou seja

$$f_{x_i x_j} \approx \frac{f_{x_i}(x+he_j) - f_{x_i}(x-he_j)}{2h}.$$

Aplicar diferenças centradas para aproximar $f_{x_i}(x+he_j)$ e $f_{x_i}(x-he_j)$ nos dá

$$f_{x_i x_j} \approx \frac{\dfrac{f(x+he_j+he_i) - f(x+he_j-he_i)}{2h} - \dfrac{f(x-he_j+he_i) - f(x-he_j-he_i)}{2h}}{2h},$$

de onde obtemos

$$\frac{\partial^2 f}{\partial x_i \partial x_j} \approx \frac{f(x+he_i+he_j)-f(x+he_i-he_j)-f(x-he_i+he_j)+f(x-he_i-he_j)}{4h^2}. \tag{3}$$

# 2 Implementação

Pede-se para implementar a função `fin_diff` que aproxima gradientes e derivadas de uma função em um ponto. As fórmulas a serem usadas são (1), (2) e (3).

## 2.1 Primeira linha da função

```
def fin_diff(f,x,degree,h):
```

sendo

- `f:` a função a ser derivada. Mais detalhes sobre ela abaixo;
- `x:` o numpy array com o ponto no qual se deseja calcular as derivadas;
- `degree:` o grau da derivada: 1 para calcular o vetor gradiente (derivadas primeiras) e 2 para calcular a matriz Hessiana (derivadas segundas);
- `h:` o passo a ser usado nas diferenças finitas.

Sobre a função `f:` deve ser uma função cuja entrada seja um numpy array e a saída seja o valor da função naquele ponto. Um exemplo de função em $\mathbb{R}^2$ : se a função desejada for $f(x_1, x_2) = -x_1^4 + 2x_1^2 - x_1 + x_1x_2 - x_2^2$, podemos implementar

```
def f(x):
    return -x[0]**4+2*x[0]**2-x[0]+x[0]*x[1]-x[1]**2
```

Para calcular a função em $x = [1, 3]^T$, fazemos

```
x = np.array([1,3])
print(f(x))
-6
```

## 2.2 Saída

- Se `degree == 1,` a saída deve ser um numpy array de $n$ componentes com o vetor gradiente aproximado.
- Se `degree == 2,` a saída deve ser um numpy array $n \times n$ com a matriz Hessiana aproximada.

# 3 Exemplo

Considere a função

```
def f(x):
     return np.sin(x[0]*x[1])*np.cos(x[1]**2)
```

Ao se executar

```
g = fin_diff(f,np.array([-1,2]),1,1e-5)
print(f"∇f(x) = g")
H = fin_diff(f,np.array([-1,2]),2,1e-5)
print(f"∇²f(x) = H")
```

obter-se-á o seguinte output

```
∇f(x) = [ 0.54402345 -3.02464597]
∇²f(x) = [[-2.37742492 -1.05880305]
[-1.05880305 -8.96085095]]
```

Para a função

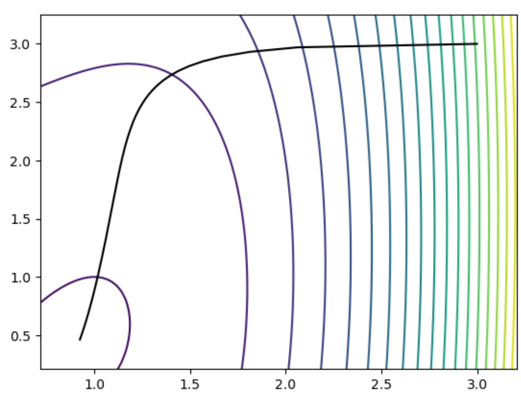
```
def f(x):
    return x[0]**4-2*x[0]**2+x[0]-x[0]*x[1]+x[1]**2
```

sem implementar o gradiente (pois será computado por diferenças finitas), ao se executar

```
x,k = gd(f,np.array([3,3]),grad,alpha = 1e-2,eps = 1e-8,fd = True,
         plot = True)
print(f"x = x")
print(f"k = k")
```

os seguinte output será obtido (ou semelhante)

```
x = [0.92442503 0.46221252]
k = 1109
```



## 4 Detalhes e observações

- No cálculo da Hessiana, note que pelo fato da mesma ser quadrada será necessário calcular apenas metade dos termos mistos;

- há outras fórmulas de diferenças finitas para a Hessiana. Essas são as centradas. É possível aplicar por exemplo a fórmula de derivadas mistas pra calcular as puras, aparecendo assim termos como $f(x + 2he_j)$;

- implementar isso de maneira eficiente é um problema interessante. Por exemplo, se $f(x)$ foi calculado uma vez, não precisaria ser calculado de novo (já que aparece em várias fórmulas). Algumas implementações só guardam metade da Hessiana (pois a outra metade tem os mesmos valores e assim economiza bastante espaço de armazenamento), mas daí todas as contas que vão usar a Hessiana precisam ser modificadas de acordo. Não vou cobrar esses pontos, mas é bom que saibam que pra ficar realmente eficiente o desafio é bem maior.

- quando forem fazer testes, lembrem-se do Projeto 2 que $h$ precisa ser pequeno, mas não pode ser muito pequeno, caso contrário o erro poderá ser grande.

## 5 Updates neste arquivo

- 08/11 v.2 - foi retirada a exigência de que os vetores sejam coluna. Façam com numpy arrays comuns.

- 09/11 - foi acrescentado um exemplo de uso tanto da função `fin_diff` como da função `gd` com `fd = True`.