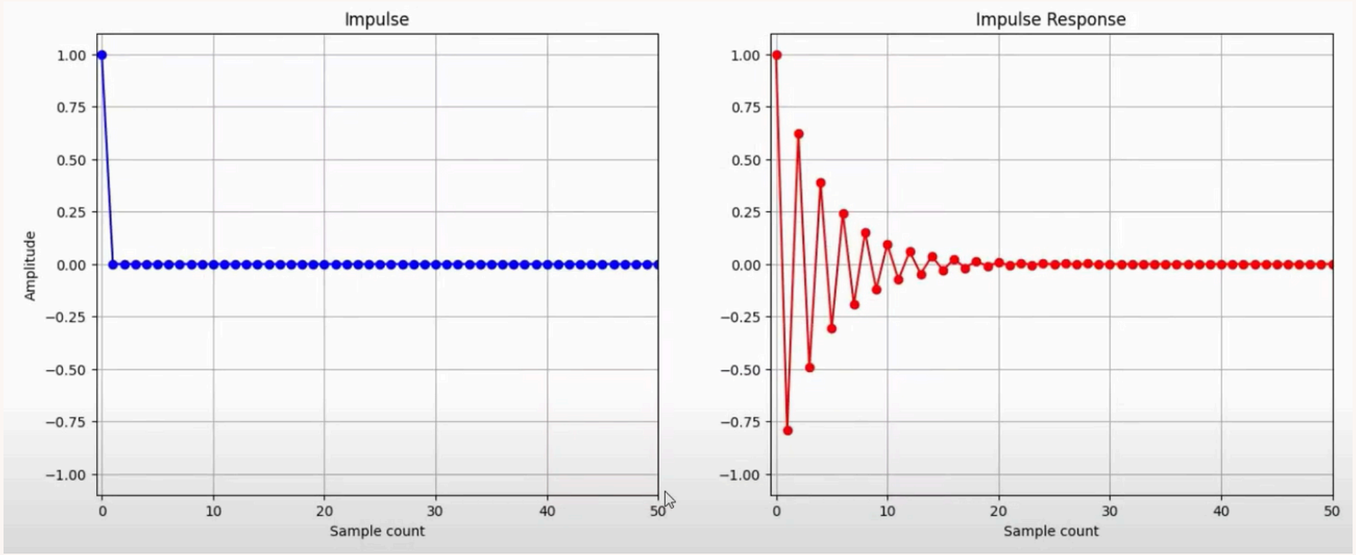
FILTRAGEM IIR DE FASE ZERO

IE550 - Processamento Digital de Sinais

FILTRAGEM IIR

- Resposta infinita ao impulso
- Recebe feedback
- Tem maior confiabilidade
- Melhor desempenho
- São mais flexíveis

$$y(n) = \sum_{i=0}^{q} b_i \, x(n-i) - \sum_{i=1}^{p} a_i \, y(n-i), \quad n \geq 0.$$



POR QUE FASE ZERO?

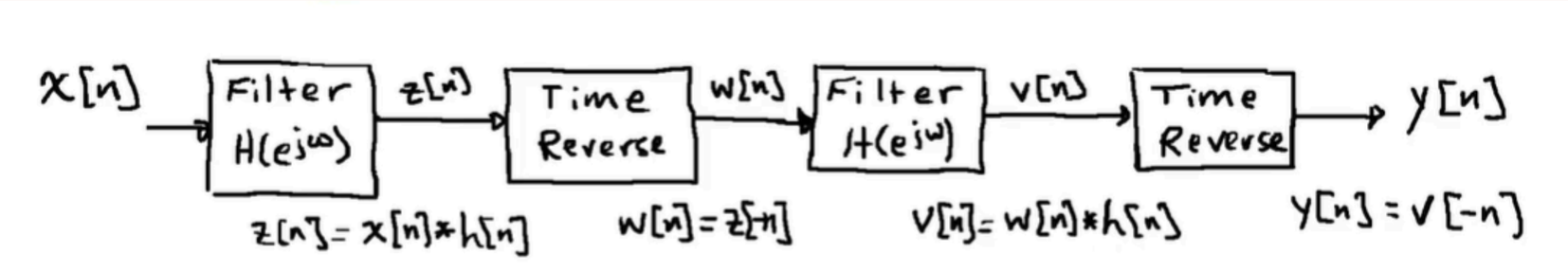
Filtros IIR em geral não têm fase linear, causando distorção de fase (alteração nas relações temporais entre componentes do sinal).

Resulta em um filtro não-causal ou seja é possível realizar a filtragem apenas quando todos os dados estão disponíveis

Com a fase zero pode-se garantir:

- Não haja distorção de fase (atrasos não lineares entre frequências)
- A estrutura temporal do sinal seja mantida

Forward-backward



PROJETO

1. Implementar filtros IIR não causais (anticausais) para alcançar fase zero, utilizando reversão no tempo e filtragem forward-backward, garantindo preservação temporal no processamento dos sinais

2. Filtro forward-backward

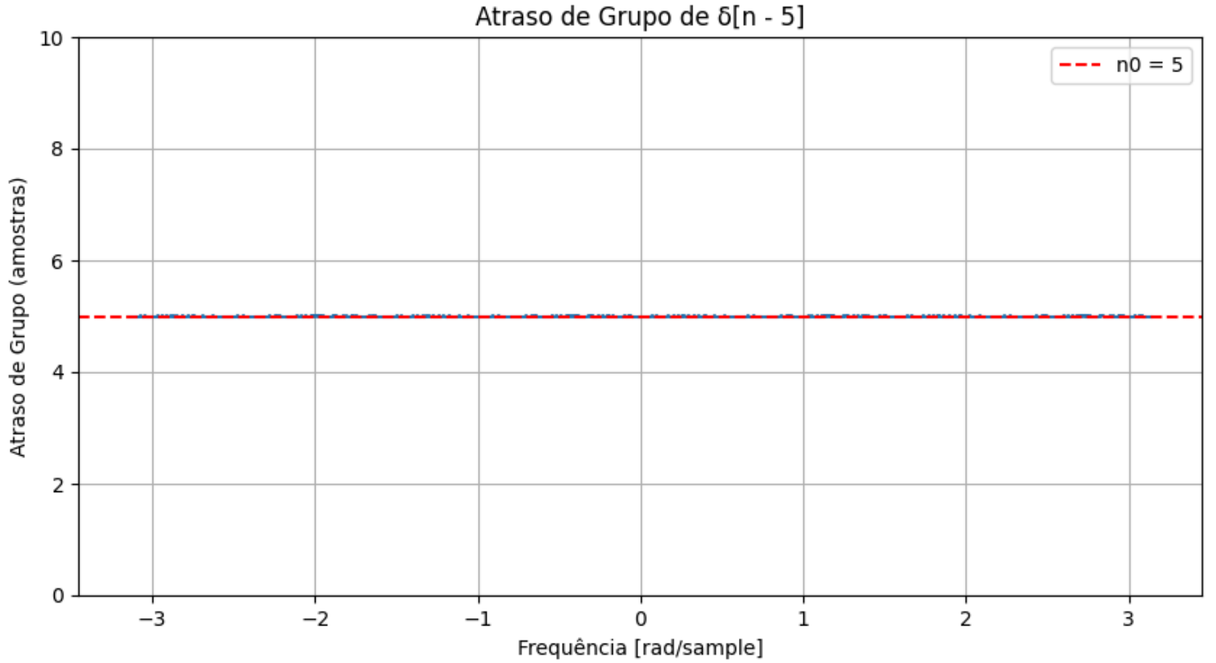
EXERCÍCIO 1.1

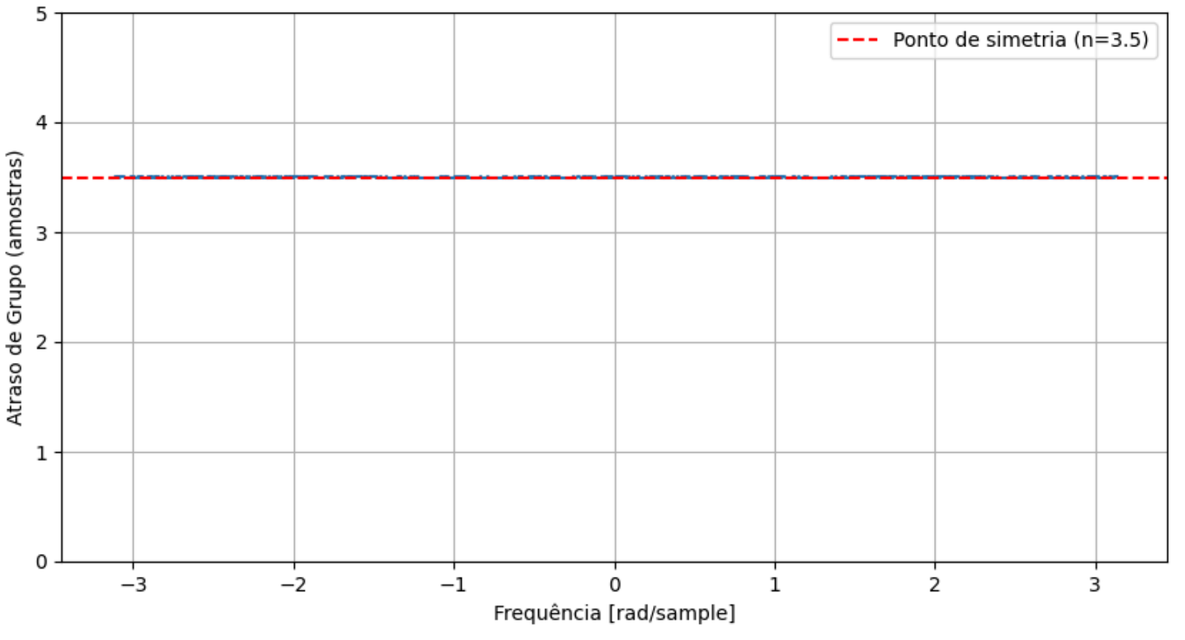
Cálculo de atraso de grupo

- Usar a função gdel do livro para calcular o atraso de grupo dos sinais:
  - Impulso deslocado
  - Sinal simétrico [1 2 3 4 4 3 2 1]

Mede o atraso temporal de cada componente de frequência. Para sinais simétricos, ele indica o centro de energia no tempo

```python
def gdel(x, n, Lfft):
    """
    Calcula o atraso de grupo de x[n].
    x: Sinal
    n: Vetor de índices de tempo
    Lfft: Tamanho da FFT
    """
    X = np.fft.fft(x, Lfft)
    dXdw = np.fft.fft(n * x, Lfft)  # Transformada de n*x[n]
    gd = np.fft.fftshift(np.real(dXdw / X))  # Atraso de grupo
    w = (2 * np.pi / Lfft) * np.arange(Lfft) - np.pi  # Frequências normalizadas
    return gd, w
```


Atraso de Grupo de δ[n - 5]
n0 = 5
Atraso de Grupo (amostras)
Frequência [rad/sample]
0
2
4
6
8
10
−3
−2
−1
0
1
2
3


Ponto de simetria (n=3.5)
Atraso de Grupo (amostras)
Frequência [rad/sample]
0
1
2
3
4
5
−3
−2
−1
0
1
2
3

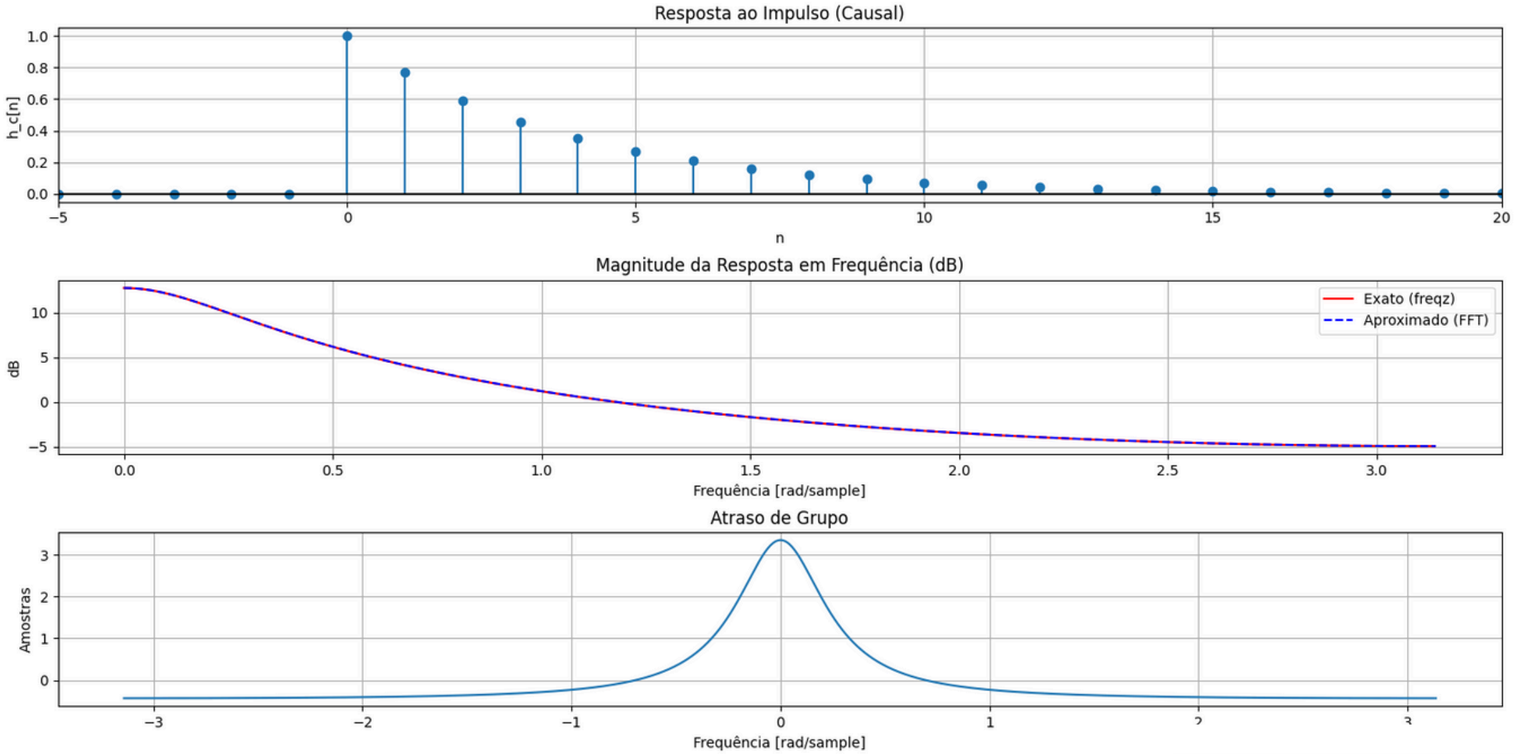
EXERCÍCIO 1.2

Sistema causal de primeira ordem

- Plotar resposta ao impulso, magnitude e atraso de grupo
- Comparar resultados usando freqz (exato) vs. FFT (aproximado)
- Comparar resultados para |z| = 0.77 e |z| = 0.95

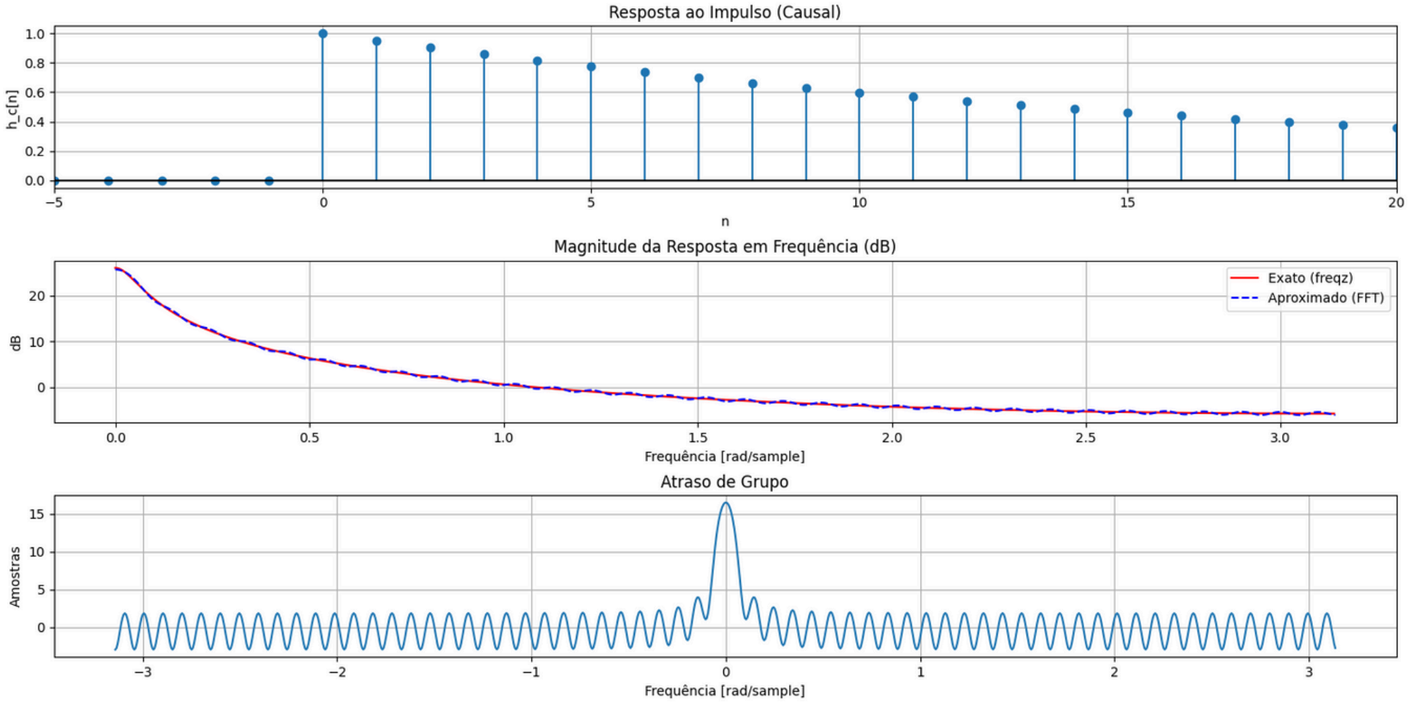
$$H_c(z) = \frac{1}{1 - 0.77z^{-1}} \qquad \text{ROC} = \{z : |z| > 0.77\}$$

```python
# 1. Resposta ao impulso
impulse = (nn == 0).astype(float)
hc = signal.lfilter(b, a, impulse)

# 2. Resposta em frequência
w, H_exato = signal.freqz(b, a, worN=1024, whole=False)

# 3. Resposta em frequência
H_approx = np.fft.fft(hc, 1024)[:512]
w_approx = np.linspace(0, np.pi, 512)
```



|z| = 0.77

|z| = 0.95

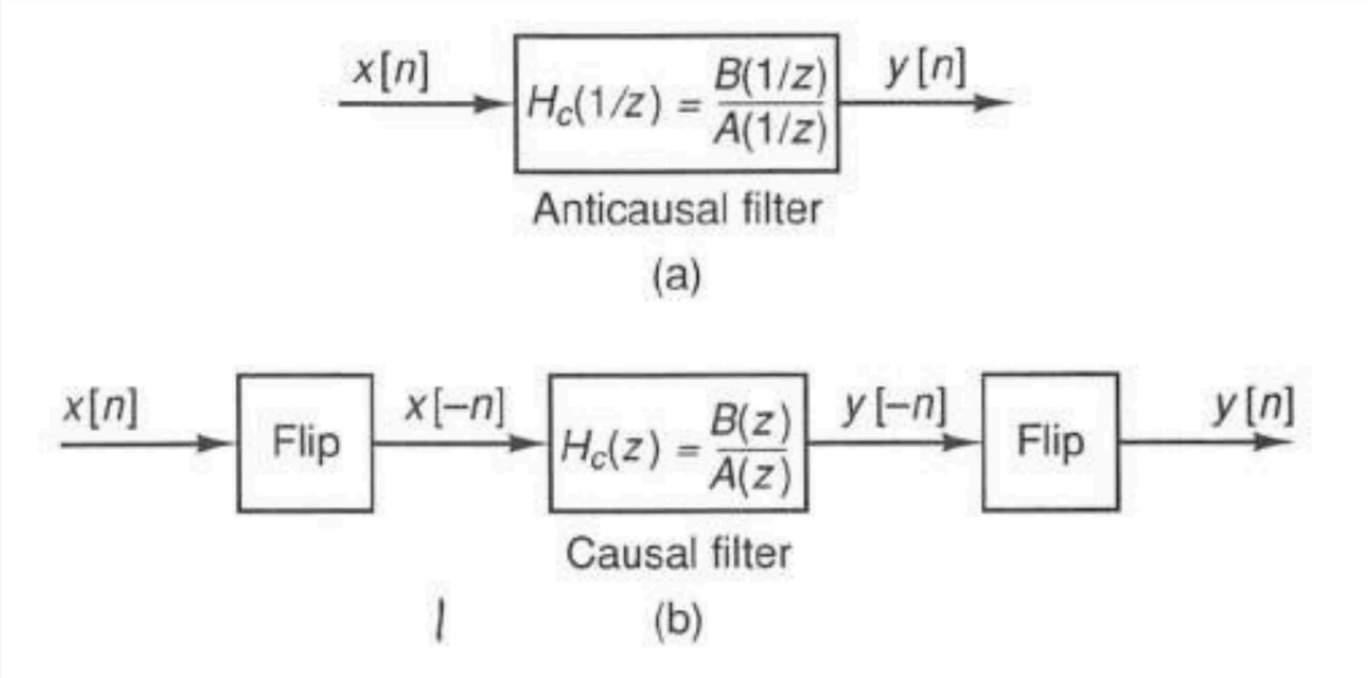


Polos próximos a |z| = 1 aumentam a seletividade, mas introduzem distorções de fase e instabilidades numéricas no cálculo do atraso de grupo.

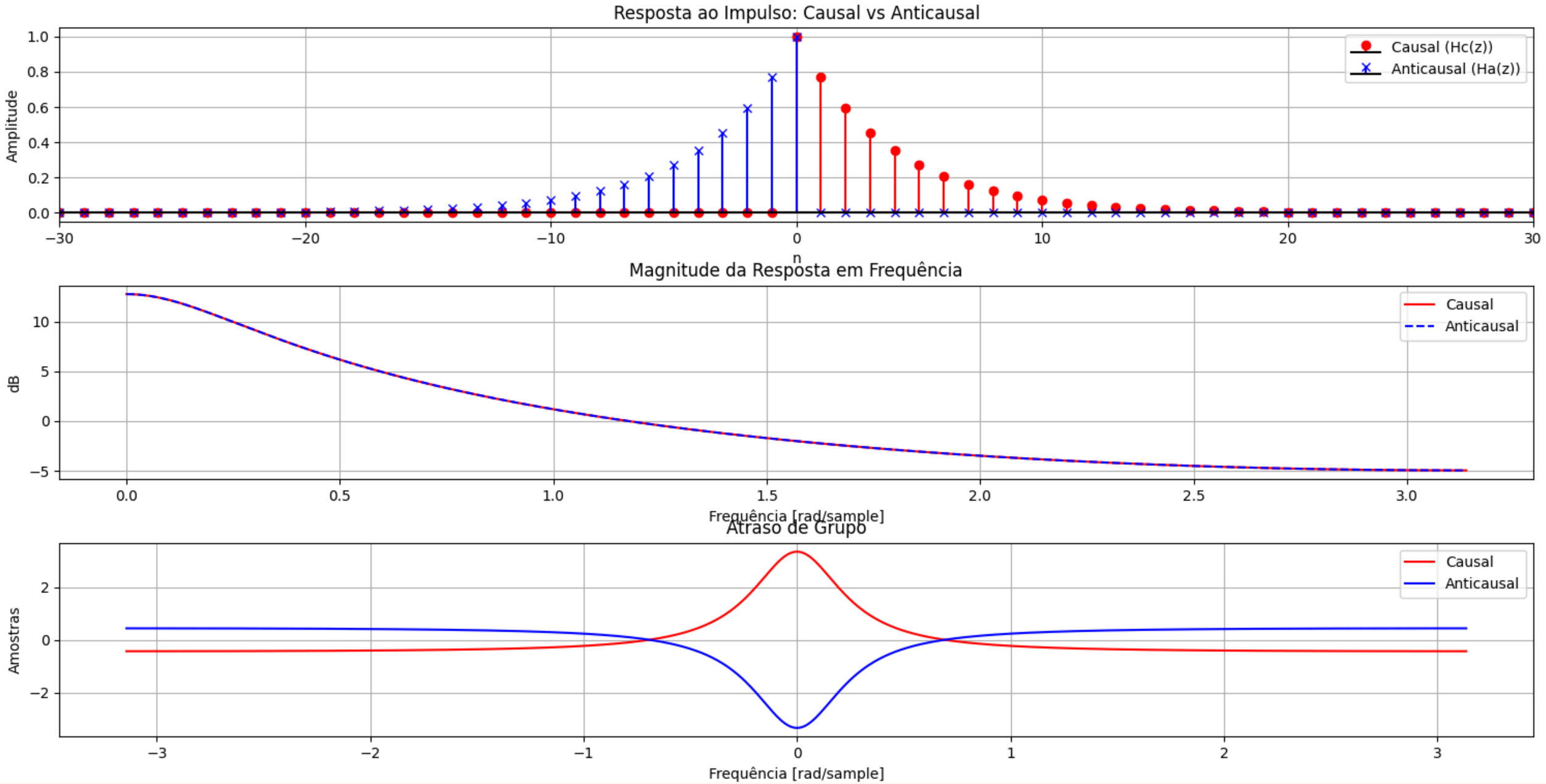
EXERCÍCIO 1.3

Sistema Anticausal de Primeira Ordem

- Usar reversão no tempo + filtragem causal
- Comparar com o sistema causal



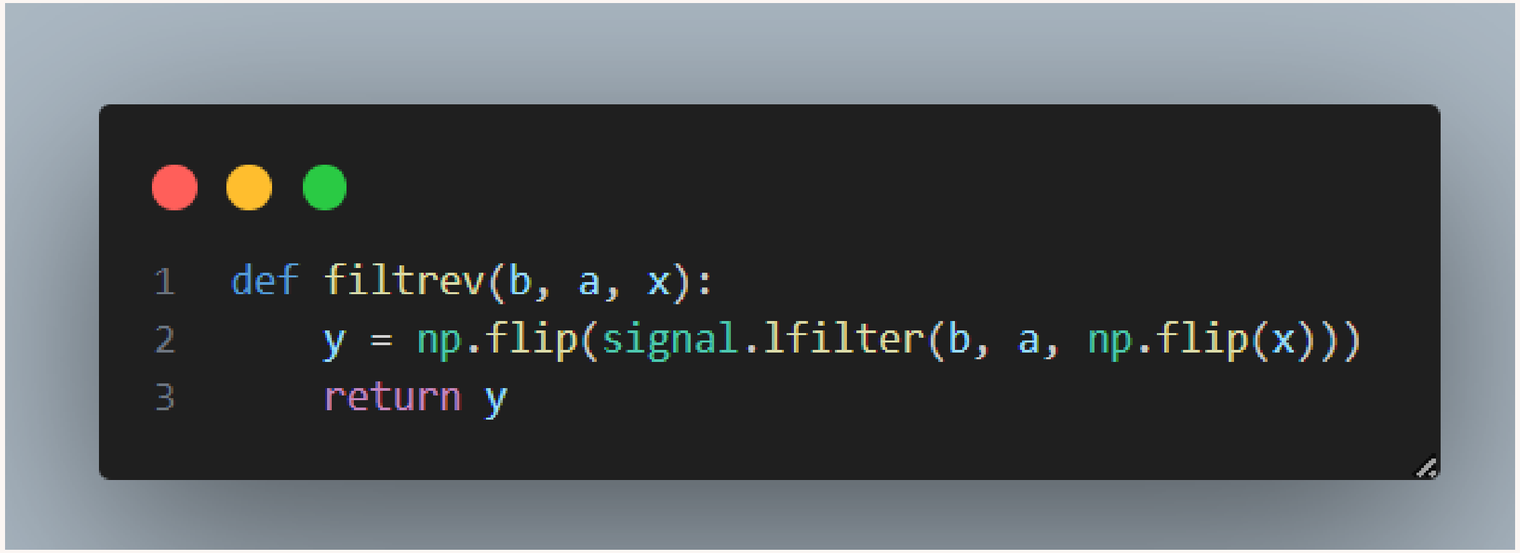

```
# Implementação do filtro anticausal (3 passos):
# 1. Reverte o sinal de entrada: x[-n]
# 2. Filtra causalmente com Hc(z)
# 3. Reverte a saída: y[n] = y_causal[-n]
ha = np.flip(signal.lfilter(b, a_anticausal, np.flip(impulse)))
```


Resposta ao Impulso: Causal vs Anticausal
Causal (Hc(z))
Anticausal (Ha(z))
Amplitude
n
Magnitude da Resposta em Frequência
Causal
Anticausal
dB
Frequência [rad/sample]
Atraso de Grupo
Causal
Anticausal
Amostras
Frequência [rad/sample]

EXERCÍCIO 1.4

Função filtrev

- Criar uma função que implemente filtragem anticausal



Obrigado!

Filtragem IIR de fase zero
Processamento Digital de Sinais - IE550