# Relatório – Ex1

# 1. Enunciado

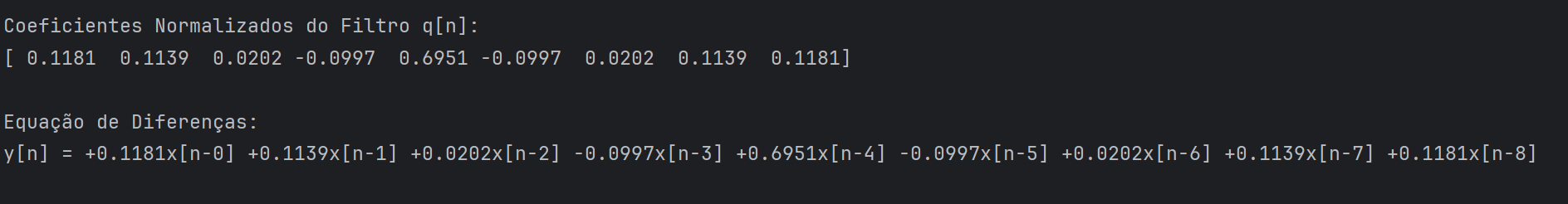
## Texto O conteúdo gerado por IA pode estar incorreto.

O objetivo é projetar um filtro FIR do tipo rejeita-faixa (band-stop). Conforme o material, um filtro rejeita-faixa pode ser construído somando-se um filtro passa-baixas (LPF) com um filtro passa-altas (HPF). O filtro passa-altas, pode ser obtido subtraindo um filtro passa-baixas de um filtro passa-tudo (um impulso).

## 2. Código Python

import numpy as np  
  
M = 8  
Fs = 22050  
f\_c1 = 2000  
f\_c2 = 4000  
  
alpha1 = (2 \* f\_c1) / Fs  
alpha2 = (2 \* f\_c2) / Fs  
  
n = np.arange(M + 1)  
h = np.zeros(M + 1)  
h\_hp = np.zeros(M + 1)  
  
center = M / 2  
for i in n:  
 if i == center:  
 h[i] = alpha1  
 h\_hp[i] = alpha2  
 else:  
 h[i] = np.sin(np.pi \* alpha1 \* (i - center)) / (np.pi \* (i - center))  
 h\_hp[i] = np.sin(np.pi \* alpha2 \* (i - center)) / (np.pi \* (i - center))  
  
d = np.zeros(M + 1)  
d[int(center)] = 1  
  
g = d - h\_hp  
q = h + g  
  
normalization\_factor = np.sum(q)  
q\_norm = q / normalization\_factor  
  
print("--- Prova Final: Exercício (1) ---")  
print("\nCoeficientes Normalizados do Filtro q[n]:")  
print(np.round(q\_norm, 4))  
print("\nEquação de Diferenças:")  
equation = "y[n] = "  
for i, coeff in enumerate(q\_norm):  
 term = f"{coeff:+.4f}x[n-{i}] "  
 equation += term  
print(equation.replace("+ -", "- "))

## 3. Resultado



# Relatório – Ex2

# 1. Enunciado

## Texto, Carta O conteúdo gerado por IA pode estar incorreto.

Projetar um filtro FIR passa-baixas usando o método da janela, com base nas especificações de magnitude fornecidas.

## 2. Código Python M = 10

import numpy as np  
  
omega\_p = 0.1 \* np.pi  
omega\_s = 0.4 \* np.pi  
delta\_omega = omega\_s - omega\_p  
  
delta\_t\_norm = delta\_omega / np.pi  
M = int(np.ceil(3.0 / delta\_t\_norm))  
  
omega\_c = (omega\_p + omega\_s) / 2  
alpha = omega\_c / np.pi  
center = M / 2  
  
n = np.arange(M + 1)  
h\_ideal = np.zeros(M + 1)  
  
for i in n:  
 if i == center:  
 h\_ideal[i] = alpha  
 else:  
 h\_ideal[i] = np.sin(np.pi \* alpha \* (i - center)) / (np.pi \* (i - center))  
  
w\_barlett = np.bartlett(M + 1)  
  
h = h\_ideal \* w\_barlett  
  
h\_norm = h / np.sum(h)  
  
print("--- Prova Final: Exercício (2) ---")  
print(f"\nOrdem do Filtro (M) calculada: {M}")  
print("\nCoeficientes Normalizados do Filtro h[n]:")  
print(np.round(h\_norm, 4))  
print("\nEquação de Diferenças:")  
equation = "y[n] = "  
for i, coeff in enumerate(h\_norm):  
 if np.abs(coeff) > 1e-6:  
 term = f"{coeff:+.4f}x[n-{i}] "  
 equation += term  
print(equation.replace("+ -", "- ").strip())

## 2. Código Python M = 22

import numpy as np  
  
M = 22  
omega\_c = 0.25 \* np.pi  
  
alpha = omega\_c / np.pi  
center = M / 2.0  
  
n = np.arange(M + 1)  
h\_ideal = np.zeros(M + 1)  
  
for i in n:  
 if i == center:  
 h\_ideal[i] = alpha  
 else:  
 h\_ideal[i] = np.sin(np.pi \* alpha \* (i - center)) / (np.pi \* (i - center))  
  
w\_hanning = np.hanning(M + 1)  
h = h\_ideal \* w\_hanning  
  
h\_norm = h / np.sum(h)  
  
print(f"Filtro projetado com Janela de Hanning e M={M}")  
  
print("\nCoeficientes Normalizados do Filtro h[n]:")  
print(np.round(h\_norm, 8).tolist())  
  
print("\nEquação de Diferenças:")  
equation = "y[n] = "  
for i, coeff in enumerate(h\_norm):  
 if abs(coeff) > 1e-9:  
 if i > 0:  
 equation += f"{coeff:+.4f}x[n-{i}] "  
 else:  
 equation += f"{coeff:.4f}x[n-{i}] "  
  
print(equation.replace("+ -", "- ").strip())

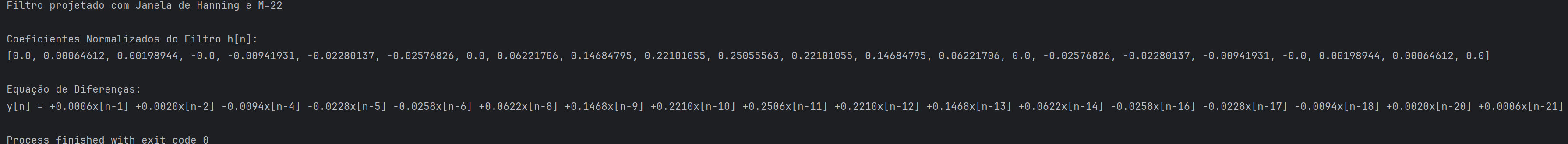
## 3. Resultado

Foi considerado (M=10, Barlett), pois pode ser considerada a solução mais eficiente. Ela cumpre os requisitos com a menor ordem de filtro possível de acordo com as opções da apostila.

Texto

O conteúdo gerado por IA pode estar incorreto.

Já esse caso foi considerado (M=22, Hanning) , que oferece um desempenho de atenuação melhor do que o solicitado, ao custo de ser um filtro mais longo e computacionalmente mais caro.



Coeficientes Normalizados do Filtro h[n]:

[0.0, 0.00064612, 0.00198944, -0.0, -0.00941931, -0.02280137, -0.02576826, 0.0, 0.06221706, 0.14684795, 0.22101055, 0.25055563, 0.22101055, 0.14684795, 0.06221706, 0.0, -0.02576826, -0.02280137, -0.00941931, -0.0, 0.00198944, 0.00064612, 0.0]

Equação de Diferenças:

y[n] = +0.0006x[n-1] +0.0020x[n-2] -0.0094x[n-4] -0.0228x[n-5] -0.0258x[n-6] +0.0622x[n-8] +0.1468x[n-9] +0.2210x[n-10] +0.2506x[n-11] +0.2210x[n-12] +0.1468x[n-13] +0.0622x[n-14] -0.0258x[n-16] -0.0228x[n-17] -0.0094x[n-18] +0.0020x[n-20] +0.0006x[n-21]

Entrada do Sinal -> Linha de Atraso (Delay Line) -> Multiplicadores (Coeficientes) -> Somador e Saída

# Relatório – Ex3

# 1. Enunciado

## 

Explique, em um curto parágrafo, o conceito de janela. Para que ela é usada? Por que ela é importante?

## 2. Código Python

SEM CóDIGO

## 3. Resultado

Uma janela, no contexto de processamento de sinais, é uma função de comprimento finito que é multiplicada por um segmento de um sinal. Seu uso principal ocorre em duas áreas:

No projeto de filtros FIR e análise de frequência. No projeto de filtros, uma janela é usada para truncar a resposta ao impulso infinita de um filtro ideal (como a função sinc), transformando-o em um filtro prático e de comprimento finito.

Diferentes tipos de janelas, como Hanning ou Hamming, são aplicados para atenuar o "Efeito de Gibbs", que são flutuações indesejadas na resposta de frequência causadas pela truncagem abrupta de uma janela retangular.

Seu segundo uso é na análise de sinais, onde um sinal de longa duração é multiplicado por uma janela antes que sua Transformada de Fourier seja calculada, a fim de evitar flutuações espúrias no domínio da frequência. A importância da janela reside na sua capacidade de viabilizar o projeto de filtros FIR práticos e de permitir uma análise espectral mais precisa de segmentos de sinais, controlando os efeitos colaterais da transição abrupta entre o sinal analisado e o zero.

# Relatório – Ex4

# 1. Enunciado

## Texto O conteúdo gerado por IA pode estar incorreto.

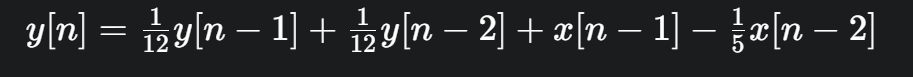
## 2. Código Python

import numpy as np  
from scipy.signal import zpk2tf  
  
poles = np.array([-1/4, 1/3])  
zeros = np.array([1/5])  
gain = 1  
  
num\_poly\_z, den\_poly\_z = zpk2tf(zeros, poles, gain)  
  
b\_coeffs = np.array([0, 1, -1/5])  
a\_coeffs = np.array([1, -1/12, -1/12])  
  
term\_y1 = -a\_coeffs[1]/a\_coeffs[0]  
term\_y2 = -a\_coeffs[2]/a\_coeffs[0]  
term\_x1 = b\_coeffs[1]/a\_coeffs[0]  
term\_x2 = b\_coeffs[2]/a\_coeffs[0]  
  
is\_stable\_and\_causal = np.all(np.abs(poles) < 1)  
  
print("--- Prova Final: Exercício (4) ---")  
print(f"\nPolos: {poles.tolist()}")  
print(f"Zeros: {zeros.tolist()}")  
  
print("\nFunção de Transferência H(z):")  
print(f"Numerador (em z^-1): {b\_coeffs.tolist()}")  
print(f"Denominador (em z^-1): {np.round(a\_coeffs, 4).tolist()}")  
  
print("\nEquação de Diferenças:")  
equation = f"y[n] = {term\_y1:.4f}\*y[n-1] + {term\_y2:.4f}\*y[n-2] + {term\_x1:.4f}\*x[n-1] {term\_x2:+.4f}\*x[n-2]"  
print(equation.replace("+ -", "- "))  
  
  
print("\nAnálise de Estabilidade e Causalidade:")  
print(f"Magnitudes dos polos: {np.abs(poles).tolist()}")  
print(f"Estável e Causal? {'Sim, todos os polos estão dentro do círculo unitário.' if is\_stable\_and\_causal else 'Não'}")

## 3. Resultado

Texto

O conteúdo gerado por IA pode estar incorreto.



# Relatório – Ex5

# 1. Enunciado

## Texto O conteúdo gerado por IA pode estar incorreto.

## 2. Código Python

import numpy as np  
import math  
from collections import Counter  
  
s = np.array([-1, 2, -3, 3, 2, 1, -1, -1, -4, 5, 5, 4])  
Fs = 16000  
window\_duration\_s = 0.000125  
overlap = 0.50  
log\_base = 10  
  
L = int(window\_duration\_s \* Fs)  
hop\_size = int(L \* (1 - overlap))  
  
f\_entropy = []  
  
num\_windows = int(np.floor((len(s) - L) / hop\_size)) + 1  
  
for i in range(num\_windows):  
 start\_index = i \* hop\_size  
 end\_index = start\_index + L  
 window = s[start\_index:end\_index]  
  
 counts = Counter(window)  
 entropy = 0.0  
  
 for count in counts.values():  
 p\_i = count / L  
 entropy -= p\_i \* math.log(p\_i, log\_base)  
  
 f\_entropy.append(entropy)  
  
print("--- Prova Final: Exercício (5) ---")  
print(f"\n1. Comprimento do vetor de características f[n]: {len(f\_entropy)}")  
print(f"\n2. Valores de f[n] para Entropia (base 10):")  
print(np.round(f\_entropy, 3).tolist())

## 3. Resultado

Tela de celular com aplicativo aberto

O conteúdo gerado por IA pode estar incorreto.

# Relatório – Ex6

# 1. Enunciado

## 

Explique, em um curto parágrafo, o conceito de escala Bark. Por que ela é importante?

## 2. Código Python

NENHUM

## 3. Resultado

A escala Bark é um conceito fundamental no processamento de fala por ser baseada na percepção auditiva humana.

* Ela foi projetada para simular a forma como o sistema auditivo humano, especificamente a cóclea, reage e processa as ondas sonoras.
* Diferente da escala de Hertz que é linear, a escala Bark divide o espectro de frequência em 25 bandas críticas. Essas bandas são mais estreitas em baixas frequências e progressivamente mais largas em altas frequências, espelhando as bandas de frequência da cóclea.
* A sua principal importância está na extração de características (features). Ao analisar um sinal de fala com base nessas bandas, criamos atributos que são mais robustos e significativos para tarefas como reconhecimento de voz, pois eles imitam a percepção humana.
* Um método prático para sua aplicação consiste em projetar 25 filtros FIR, cada um correspondendo a uma banda da escala, para então analisar a energia do sinal de fala em cada uma dessas bandas.

# Relatório – Ex7

# 1. Enunciado

## 

Estimar os coeficientes de Predição Linear (LPC) de 4ª ordem {a1, a2, a3, a4​ } para o sinal fornecido. O método padrão para isso, como indicado no material de aula, é o método da autocorrelação, que envolve resolver o sistema de equações de Yule-Walker.

## 2. Código Python

import numpy as np  
  
y = np.array([2, 5, 2, 3, 5, 8, 4, 8, 10, 6])  
p = 4  
N = len(y)  
  
num\_equations = N - p  
X = np.zeros((num\_equations, p))  
yp = y[p:]  
  
for i in range(num\_equations):  
 X[i, :] = y[i : i+p]  
  
XTX = X.T @ X  
XTy = X.T @ yp  
  
a\_coeffs\_reversed = np.linalg.solve(XTX, XTy)  
  
print("--- Prova Final: Exercício (7) ---")  
print("\nCoeficientes LPC de 4ª ordem:")  
for i, coeff in enumerate(a\_coeffs\_reversed):  
 print(f"a\_{i+1} = {coeff:.4f}")

## 3. Resultado

Texto

O conteúdo gerado por IA pode estar incorreto.

# Relatório – Ex8

# 1. Enunciado

## 

Explique as diferenças, vantagens e desvantagens de características "handcrafted" (feitas à mão) em comparação com aquelas aprendidas ("learned features").

## 2. Código Python

Sem código

## 3. Resultado

A principal diferença entre características "handcrafted" e "learned" reside em como elas são criadas e o nível de conhecimento de domínio necessário.

Atributos Handcrafted (Feitos à Mão)

Diferença: São características projetadas por especialistas humanos com base em conhecimento prévio da área (como física, psicoacústica ou processamento de sinais). Elas são calculadas usando fórmulas matemáticas predefinidas. Exemplos vistos no curso incluem Energia, Taxa de Cruzamento por Zero (ZCR), Cepstrum, Jitter, Shimmer e Entropia.

* Vantagens:
  + Interpretabilidade: Possuem um significado físico ou perceptual claro (ex: F0 se relaciona com o tom da voz, energia com a intensidade).
  + Eficiência com Poucos Dados: Por incorporarem conhecimento humano, geralmente funcionam bem mesmo com conjuntos de dados menores.
  + Menor Custo Computacional: A extração é, em geral, rápida e podem ser usadas com classificadores mais simples e rápidos de treinar.
* Desvantagens:
  + Exigem Especialização: Sua criação depende de um profundo conhecimento do problema. Potencialmente Subótimas:
  + Podem não ser as características mais discriminativas para uma tarefa específica, pois são baseadas em suposições humanas.

Atributos Aprendidos (Learned Features)

Diferença: São características extraídas automaticamente por um modelo de aprendizado de máquina, tipicamente uma rede neural profunda (como autoencoders), diretamente a partir dos dados brutos. O próprio modelo decide quais representações são mais úteis.

* Vantagens:
  + Otimização para a Tarefa: O modelo aprende as características mais discriminativas para o conjunto de dados e a tarefa em questão, podendo levar a uma performance superior.
  + Automatização: Dispensa a necessidade de um especialista para projetar as características manualmente.
  + Generalidade: A mesma arquitetura de rede neural pode ser aplicada a diferentes problemas para aprender características específicas para cada um.
* Desvantagens:
  + Natureza de "Caixa-Preta": Frequentemente são difíceis de interpretar; seu significado físico ou perceptual não é claro.
  + Necessidade de Muitos Dados: Redes neurais profundas necessitam de grandes volumes de dados para aprender representações eficazes.
  + Alto Custo Computacional: O treinamento desses modelos é computacionalmente intensivo e demorado.

# Relatório – Ex9

# 1. Enunciado

## Diagrama, Esquemático O conteúdo gerado por IA pode estar incorreto.

## 2. Código Python

import numpy as np  
  
pi = np.array([0.7, 0.2, 0.1])  
A = np.array([[0.6, 0.3, 0.1], [0.4, 0.1, 0.5], [0.1, 0.7, 0.2]])  
B = np.array([[0.1, 0.8], [0.4, 0.3], [0.3, 0.1]])  
obs\_map = {'rainy': 0, 'sunny': 1}  
O = ['rainy', 'sunny']  
  
obs\_seq = [obs\_map[ob] for ob in O]  
N = A.shape[0]  
T = len(obs\_seq)  
  
alpha = np.zeros((T, N))  
alpha[0, :] = pi \* B[:, obs\_seq[0]]  
  
for t in range(1, T):  
 for j in range(N):  
 alpha[t, j] = np.sum(alpha[t-1, :] \* A[:, j]) \* B[j, obs\_seq[t]]  
  
prob\_forward = np.sum(alpha[T-1, :])  
  
delta = np.zeros((T, N))  
psi = np.zeros((T, N), dtype=int)  
  
delta[0, :] = pi \* B[:, obs\_seq[0]]  
  
for t in range(1, T):  
 for j in range(N):  
 probs = delta[t-1, :] \* A[:, j]  
 delta[t, j] = np.max(probs) \* B[j, obs\_seq[t]]  
 psi[t, j] = np.argmax(probs)  
  
prob\_viterbi = np.max(delta[T-1, :])  
best\_path = np.zeros(T, dtype=int)  
best\_path[T-1] = np.argmax(delta[T-1, :])  
  
for t in range(T-2, -1, -1):  
 best\_path[t] = psi[t+1, best\_path[t+1]]  
  
print("--- Prova Final: Exercício (9) ---")  
print("\n1. Probabilidade P(O|λ) (Forward Algorithm):")  
print(f"{prob\_forward:.4f}")  
print("\n2. Melhor Caminho de Estados (Viterbi Algorithm):")  
print(f"Caminho: {[f'q{state}' for state in best\_path]}")  
print(f"Probabilidade do melhor caminho: {prob\_viterbi:.4f}")

## 3. Resultado

Texto

O conteúdo gerado por IA pode estar incorreto.

8,32% e 3,36%

# Relatório – Ex10

# 1. Enunciado

## Texto O conteúdo gerado por IA pode estar incorreto.

## 2. Código Python

import numpy as np  
  
Ca = np.array([0.2, 1.0, 0.7])  
Cb = np.array([0.3, 0.9, 0.8])  
t = np.array([0.2, 0.9, 0.9])  
  
dist\_A = np.linalg.norm(t - Ca)  
dist\_B = np.linalg.norm(t - Cb)  
  
best\_class = 'A' if dist\_A < dist\_B else 'B'  
  
print("--- Prova Final: Exercício (10) ---")  
print("\nParte 1: Classificação por Distância Euclidiana")  
print(f"Distância para a Classe A: {dist\_A:.4f}")  
print(f"Distância para a Classe B: {dist\_B:.4f}")  
print(f"Melhor correspondência: Classe {best\_class}")

## 3. Resultado

Texto

O conteúdo gerado por IA pode estar incorreto.