Relatório – Ex1

1. Enunciado

(1): Design an FIR filter ($q[n]$), subtype I, of order $M = 8$ to cut-off frequencies within the range 2000 Hz $\sim$ 4000 Hz, allowing for all the others to pass through. Assume that the input signal to be filtered ($x[n]$) was sampled at 22050 samples per second. Normalize the filters' coefficients in such a way that the filter presents a gain of 0dB in the pass-band. Lastly, write down the difference equation to filter an input signal $x[n]$ by using $q[n]$.

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
```

```python
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

```
Coeficientes Normalizados do Filtro q[n]:
[ 0.1181  0.1139  0.0202 -0.0997  0.6951 -0.0997  0.0202  0.1139  0.1181]

Equação de Diferenças:
y[n] = +0.1181x[n-0] +0.1139x[n-1] +0.0202x[n-2] -0.0997x[n-3] +0.6951x[n-4] -0.0997x[n-5] +0.0202x[n-6] +0.1139x[n-7] +0.1181x[n-8]
```

Relatório – Ex2

1. Enunciado

(2): Design an FIR filter with the following specifications:

- $0.98 \leqslant |H(e^{j\omega})| \leqslant 1.02$, in the range $0 \leqslant \omega \leqslant 0.1\pi$
- $|H(e^{j\omega})| \leqslant 0.06$, in the range $0.4\pi \leqslant \omega \leqslant \pi$

Then, normalize the windowed filter, write down the difference equation to implement the filter you have just designed in a computer-based application, and depict the corresponding block diagram.

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

2. Código Python M = 20

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import freqz

M = 20
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_barlett = np.bartlett(M + 1)

h = h_ideal * w_barlett

h_norm = h / np.sum(h)

print("--- Prova Final: Exercício (2) - Solução Final Corrigida ---")
print(f"Filtro projetado com Janela de Barlett e M={M}")

print("\nCoeficientes Normalizados do Filtro h[n]:")
print(np.round(h_norm, 8).tolist())

print("\nEquação de Diferenças:")
equation = "y[n] = "
is_first_term = True
```

```
for i, coeff in enumerate(h_norm):
    if abs(coeff) > 1e-9:
        if is_first_term:
            equation += f"{coeff:.8f}*x[n-{i}]"
            is_first_term = False
        else:
            equation += f" {coeff:+.8f}*x[n-{i}]"
print(equation.replace("+ -", "- "))


w, H = freqz(h_norm, 1, worN=2048)

plt.figure(figsize=(12, 8))
plt.plot(w / np.pi, 20 * np.log10(np.abs(H)))
plt.title('Resposta de Frequência do Filtro Projetado - Exercício (2)')
plt.xlabel('Frequência Normalizada (x π rad/amostra)')
plt.ylabel('Magnitude (dB)')
plt.grid(True)
plt.ylim(-60, 5)
plt.show()
```

3. Resultado

```
Coeficientes Normalizados do Filtro h[n]:
h[00] = +0.00000000
h[01] = +0.00064612
h[02] = +0.00198944
h[03] = -0.00000000
h[04] = -0.00941931
h[05] = -0.02280137
h[06] = -0.02576826
h[07] = +0.00000000
h[08] = +0.06221706
h[09] = +0.14684795
h[10] = +0.22101055
h[11] = +0.25055563
h[12] = +0.22101055
h[13] = +0.14684795
h[14] = +0.06221706
h[15] = +0.00000000
h[16] = -0.02576826
h[17] = -0.02280137
h[18] = -0.00941931
h[19] = -0.00000000
h[20] = +0.00198944
h[21] = +0.00064612
h[22] = +0.00000000
```

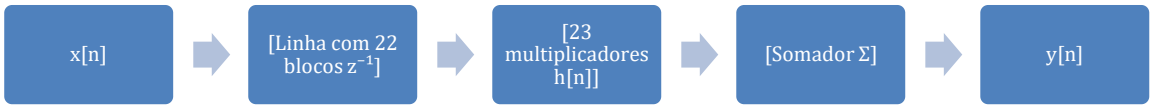
```
Equação de Diferenças:
y[n] =
  0.00064612 * x[n-1]
  +0.00198944 * x[n-2]
  -0.00941931 * x[n-4]
  -0.02280137 * x[n-5]
  -0.02576826 * x[n-6]
  +0.06221706 * x[n-8]
  +0.14684795 * x[n-9]
  +0.22101055 * x[n-10]
  +0.25055563 * x[n-11]
  +0.22101055 * x[n-12]
  +0.14684795 * x[n-13]
  +0.06221706 * x[n-14]
  -0.02576826 * x[n-16]
  -0.02280137 * x[n-17]
  -0.00941931 * x[n-18]
  +0.00198944 * x[n-20]
  +0.00064612 * x[n-21]
```

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] = 0.00064612*x[n-1] +0.00198944*x[n-2] -0.00941931*x[n-4] -0.02280137*x[n-5] -0.02576826*x[n-6] +0.06221706*x[n-8] +0.14684795*x[n-9] +0.22101055*x[n-10] +0.25055563*x[n-11] +0.22101055*x[n-12] +0.14684795*x[n-13] +0.06221706*x[n-14] -0.02576826*x[n-16] -0.02280137*x[n-17] -0.00941931*x[n-18] +0.00198944*x[n-20] +0.00064612*x[n-21]



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

Relatório – Ex3

1. Enunciado

(3): Explain, in a short paragraph, the concept of window. What is it used for? Why is it important?

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

(4): Find the transfer function $H[z]$ whose poles are at $z = -\frac{1}{4}$ and $z = \frac{1}{3}$, in addition to one zero at $z = \frac{1}{5}$. Write down the corresponding difference equation. Is the transfer function stable and causal?

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

```
--- Prova Final: Exercício (4) ---

Polos: [-0.25, 0.3333333333333333]
Zeros: [0.2]

Função de Transferência H(z):
Numerador (em z^-1): [0.0, 1.0, -0.2]
Denominador (em z^-1): [1.0, -0.0833, -0.0833]

Equação de Diferenças:
y[n] = 0.0833*y[n-1] + 0.0833*y[n-2] + 1.0000*x[n-1] -0.2000*x[n-2]

Análise de Estabilidade e Causalidade:
Magnitudes dos polos: [0.25, 0.3333333333333333]
Estável e Causal? Sim, todos os polos estão dentro do círculo unitário.
```

$$y[n] = \frac{1}{12}y[n-1] + \frac{1}{12}y[n-2] + x[n-1] - \frac{1}{5}x[n-2]$$

Relatório – Ex5

1. Enunciado

(5): Consider the hypothetical speech signal segment $s[n] = \{-1, 2, -3, 3, 2, 1, -1, -1, -4, 5, 5, 4\}$, sampled at 16000 samples per second. Assume that a sliding rectangular window $w[n]$ traverses it in order to extract features for inclusion in the feature vector $f[n]$, covering 0.125ms at each placement, with 50% overlap between consecutive windows. What is the length of $f[n]$? What are the values in $f[n]$, considering the ordinary entropy, calculated with the log basis $\beta = 10$, as being the feature used?

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

```
--- Prova Final: Exercício (5) ---

1. Comprimento do vetor de características f[n]: 11

2. Valores de f[n] para Entropia (base 10):
[0.301, 0.301, 0.301, 0.301, 0.301, 0.301, 0.0, 0.301, 0.301, 0.0, 0.301]
```

Relatório – Ex6

1. Enunciado

(6): Explain, in a short paragraph, the concept of Bark scale. Why is it important?

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

(7): Estimate the 4rd order LPC coefficients $\{a_1, a_2, a_3, a_4\}$ for the signal $y[n] = \{2, 5, 2, 3, 5, 8, 4, 8, 10, 6\}$.

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

```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

```
--- Prova Final: Exercício (7) ---

Coeficientes LPC de 4ª ordem:
a_1 = 0.6975
a_2 = 0.9516
a_3 = 0.1444
a_4 = -0.2161
```

Relatório – Ex8

1. Enunciado

(8): Explain the differences between, the advantages, and disadvantages of handcrafted features in comparison with those learned.

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)

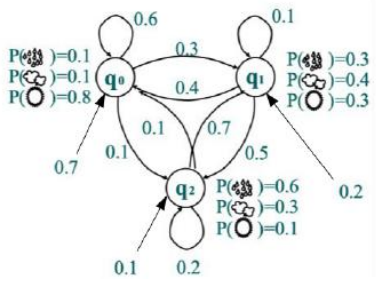
Diferença: São características extraídas automaticamente por um modelo de aprendizado de máquina, tipicamente uma rede neural profunda (como autoencoders), diretamente a partir dos dados brutos. O próprio modelo decide quais representações são mais úteis.

- 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

(9): Answer the first two HMM-related questions on page 103 for the weather-related model below, considering the observation sequence $O = \{rainy, sunny\}$.



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)
```

```python
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

```
--- Prova Final: Exercício (9) ---

1. Probabilidade P(O|λ) (Forward Algorithm):
0.0832

2. Melhor Caminho de Estados (Viterbi Algorithm):
Caminho: ['q0', 'q0']
Probabilidade do melhor caminho: 0.0336
```

8,32% e 3,36%

Relatório – Ex10

1. Enunciado

(10): Given the template model for classes C_A and C_B, i.e., $\{0.2, 1, 0.7\}$ and $\{0.3, 0.9, 0.8\}$, respectively, find the best matching class for the testing signal $t[n] = \{0.2, 0.9, 0.9\}$ by using the ordinary Euclidian distance metric. What would you do to find that match in case the testing signal and the template models have different dimensions?

2. Código Python

```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

```
--- Prova Final: Exercício (10) ---

Parte 1: Classificação por Distância Euclidiana
Distância para a Classe A: 0.2236
Distância para a Classe B: 0.1414
Melhor correspondência: Classe B
```