# EA614 - Análise de Sinais

## Atividade 1 - Sistemas LIT e Convolução

Iran Seixas Lopes Neto - RA: 244827
Letícia Lopes Mendes da Silva - RA: 184423

---

# Parte Teórica

## Item (a)

Para obter a resposta do canal $h[n]$ ao impulso, podemos resolver a operação de convolução:

$$x[n] = s[n] * h[n] = h[n] * s[n] = \sum_{k=-\infty}^{\infty} h[k]s[n-k]$$

Expandindo o somatório, temos:

$$x[n] = \ldots + h[0]s[n] + h[1]s[n-1] + h[2]s[n-2] + \ldots$$

Como, do enunciado, $x[n] = 1 \cdot s[n] + 0.7 \cdot s[n-1] - 0.04 \cdot s[n-2]$, comparando os dois lados da equação, temos que:

$$\begin{cases} h[0] = 1 \\ h[1] = 0.7 \\ h[2] = -0.04 \end{cases}$$

Logo, a resposta $h[n]$ ao impulso é dada por:

$$h[n] = \begin{cases} 1, & n = 0 \\ 0.7, & n = 1 \\ -0.04, & n = 2 \\ 0, & \text{caso contrário} \end{cases}$$

## Item (b)

Para obter a resposta combinada canal-equalizador, dada por $z[n] = h[n] * w[n]$, podemos resolver a operação de convolução:

$$y[n] = w[n] * x[n] = w[n] * (s[n] * h[n])$$

Pela propriedade associativa:

$$y[n] = (h[n] * w[n]) * s[n] = z[n] * s[n] = \sum_{k=-\infty}^{\infty} z[k]s[n-k]$$

Na situação ideal, $y[n] = s[n]$:

$$y[n] = \ldots + z[0]s[n] + \ldots = s[n]$$

Comparando os dois lados da equação, temos que $z[0] = 1$ e $z[k] = 0, \forall k \in \mathbb{Z}^*$. Logo, a resposta combinada $z[n]$ é dada por:

$$z[n] = \begin{cases} 1, & n = 0 \\ 0, & \text{caso contrário} \end{cases}$$

É interessante observar que $z[n] = \delta[n]$, a sequência de impulso unitário. Este resultado era esperado, já que, das propriedades de convolução, um sinal convoluído pelo impulso unitário resulta neste mesmo sinal como saída ($s[n] * \delta[n] = s[n]$), o que coincide com a hipótese que adotamos para uma equalização perfeita: $y[n] = s[n]$.

# Parte Computacional

Para todos os itens a seguir, utilizamos códigos em Pyhton, disponíveis no jupyter notebook em anexo, com as bibliotecas importadas e os arrays abaixo:

```
import numpy as np
import matplotlib.pyplot as plt

w1 = np.array([1, -0.7, pow(0.7, 2), -pow(0.7, 3), pow(0.7, 4), -pow(0.7,
                                        5), pow(0.7, 6)])
w2 = np.array([1, -0.2, 0.4, 0.9, -0.3, -0.9, -0.8])
h = np.array([1, 0.7, 0.04])
```

## Item (c)

```
z1 = np.convolve(h, w1) # z1[n] = h[n] * w1[n]
z2 = np.convolve(h, w2) # z2[n] = h[n] * w2[n]

print("z1 =", z1)
print("z2 =", z2)
```

Com a função "np.convolve()", fizemos a convolução $z[n] = h[n] * w[n]$, obtendo as seguintes respostas combinadas:

$$z_1 = \begin{bmatrix} 1 & 0 & 0.04 & -0.028 & 0.0196 & -0.01372 & 0.009604 & 0.0756315 & 0.00470596 \end{bmatrix}$$

$$z_2 = \begin{bmatrix} 1 & 0.5 & 0.3 & 1.172 & 0.346 & -1.074 & -1.442 & -0.596 & -0.032 \end{bmatrix}$$

Comparando os resultados $z_1[n]$ e $z_2[n]$ com o sinal $z[n]$, verificamos que a resposta combinada $z_1[n]$ é o sinal que mais se aproxima do canal-equalizador perfeito, devido à maior similaridade dos coeficientes com o resultado encontrado no item (b). Logo, o melhor filtro equalizador é o $w_1[n]$, pois ele consegue reverter as distorções causadas pelo canal de forma mais eficiente, de modo que o sinal no receptor seja mais próximo ao sinal recebido se comparado ao uso do filtro $w_2[n]$.

## Item (d)

Utilizando o seguinte comando em Python:

```
alphabet = np.array([1+1j,1-1j,-1+1j,-1-1j])
s = np.random.choice(alphabet,(500,))

x = np.convolve(s, h)    # x[n] = s[n] * h[n]

plt.scatter(x=np.real(s),y=np.imag(s), label='$s[n]$')
plt.scatter(x=np.real(x),y=np.imag(x), label='$x[n]$')
plt.xlabel('$Re$')
plt.ylabel('$Im$')
plt.legend()
plt.show()
```

geramos a sequência aleatória $s[n]$ e, então, fizemos a transmissão desse sinal pelo canal $h[n]$ por meio da convolução $x[n] = s[n] * h[n]$. Exibindo os valores dos símbolos de $s[n]$ e $x[n]$ no plano complexo, obtivemos o gráfico:

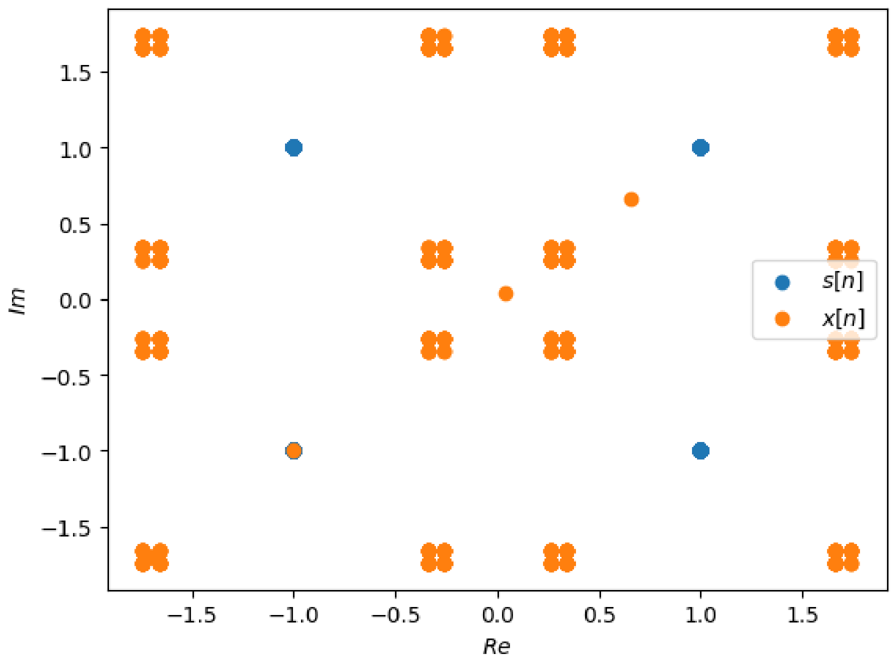


Figura 1: Gráfico com os valores dos sinais de entrada $s[n]$, em azul, e saída $x[n]$ da transmissão, em laranja.

Nesta imagem, é possível observar que o sinal $x[n]$ se dispersa consideravelmente em comparação à entrada do sistema $s[n]$. Isso mostra como a distorção causada pela resposta ao impulso $h[n]$ afeta o sinal transmitido, sendo interessante, então, aplicar um filtro equalizador na transmissão do sinal para compensar a distorção. Nota-se também que $h[n]$ tem apenas três coeficientes, e, ao sofrer convolução com o sinal $s[n]$, que assume somente quatro valores possíveis, cria um sinal com certa padronização e variação uniforme.

## Item (e)

Primeiro, geramos um ruído $\eta[n]$, uma variável aleatória Gaussiana complexa, e adicionamos ao sinal $s[n]$, criando o sinal $r[n]$. Depois, filtramos o novo sinal utilizando os filtros $w_1[n]$ e $w_2[n]$, fazendo a convolução entre $r[n]$ e cada equalizador, gerando como saídas $y_1[n]$ e $y_2[n]$.

```
eta=0.02*np.random.randn(x.size,)+0.02*1j*np.random.randn(x.size,)
r=x+eta

y1 = np.convolve(r, w1) # y1[n] = r[n] * w1[n]
y2 = np.convolve(r, w2) # y2[n] = r[n] * w2[n]

plt.scatter(x=np.real(y1),y=np.imag(y1), c='tab:red', label='$y_1[n]$')
plt.scatter(x=np.real(s),y=np.imag(s), c='b', label='$s[n$]')
plt.xlabel('$Re$')
plt.ylabel('$Im$')
plt.legend()
plt.show()

plt.scatter(x=np.real(y2),y=np.imag(y2), c='tab:red', label='$y_2[n]$')
plt.scatter(x=np.real(s),y=np.imag(s), c='b', label='$s[n]$')
plt.xlabel('$Re$')
plt.ylabel('$Im$')
plt.legend()
plt.show()
```

Então, exibimos as saídas no plano complexo com o sinal de entrada $s[n]$ nos gráficos abaixo:

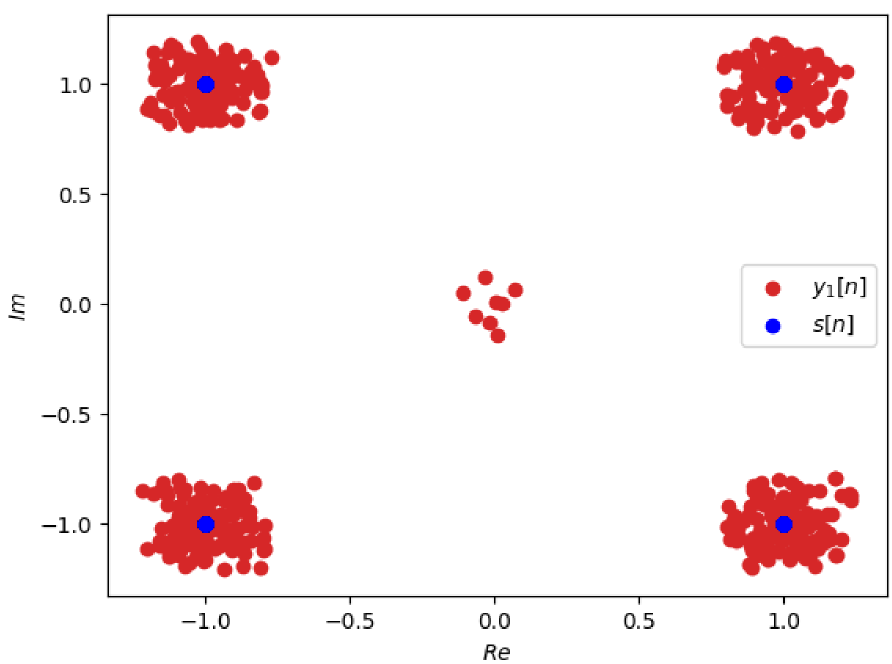


Figura 2: Gráfico com os valores dos sinais de entrada $s[n]$, em azul, e saída $y_1[n]$ ao final do processo de equalização, em vermelho.

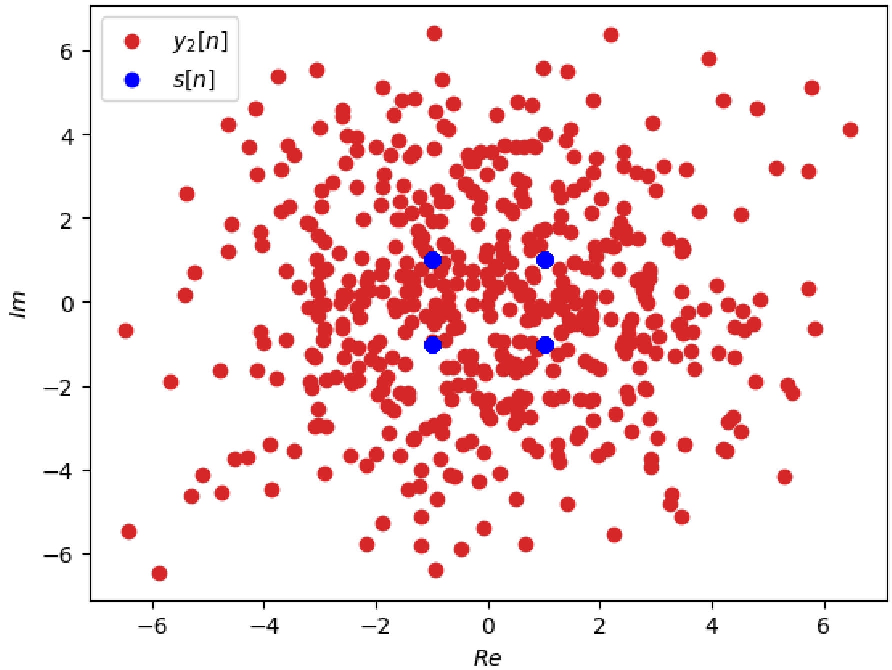


Figura 3: Gráfico com os valores dos sinais de entrada $s[n]$, em azul, e saída $y_2[n]$ ao final do processo de equalização, em vermelho.

Analisando as imagens, podemos confirmar as afirmações feitas sobre os equalizadores no item (c): os pontos da saída $y_1[n]$, filtrados por $w_1[n]$, parecem se aproximar bastante dos pontos da entrada $s[n]$, com apenas poucos erros nos pontos próximos à origem; enquanto os pontos de $y_2[n]$, filtrados por $w_2[n]$, parecem estar muito mais distorcidos em comparação aos pontos da entrada, como podemos ver pela distribuição dos pontos vermelhos e pela escala do gráfico.
Logo, a saída $y_1[n]$ está mais próxima do sinal original $s[n]$, mostrando que $w_1[n]$ é o melhor filtro equalizador.