EA614 - Análise de Sinais

Exercício de Fixação de Conceitos (EFC) 1 – Sistemas LIT e Convolução
Turma B – 2º semestre de 2024

Prof: Levy Boccato   Email: `lboccato@unicamp.br`

Introdução

Neste exercício, vamos estudar alguns aspectos básicos de um problema de grande relevância na área de comunicações, conhecido como equalização de canais, tendo como base os conceitos de convolução e sistemas lineares e invariantes com o tempo (LIT).

Visão Geral do Problema

Considere que um transmissor envia uma sequência de símbolos pertencentes a algum alfabeto finito, aqui representada por $s[n]$, através de um canal de comunicações (atmosfera, fibra ótica, par trançado, etc), modelado por um sistema LIT com resposta ao impulso $h[n]$, conforme mostra a Figura 1.

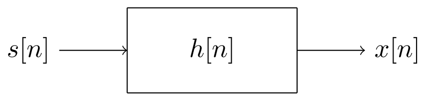

Figura 1: Transmissão através do canal $h[n]$.

Devido às características físicas do canal, o sinal que chega ao receptor corresponde a uma versão distorcida do sinal original por conta de vários efeitos, entre os quais destacamos o fenômeno conhecido como interferência intersimbólica.

Assim sendo, o objetivo é projetar um filtro no receptor, denominado equalizador (também modelado como um sistema LIT, com resposta ao impulso $w[n]$), capaz de compensar as distorções observadas, como mostra a Figura 2.

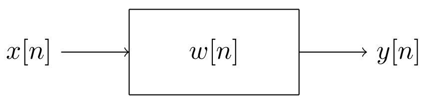

Figura 2: Uso de um equalizador $w[n]$ para processar o sinal recebido.

Parte Teórica

Vamos considerar um cenário específico neste exercício, no qual ao transmitirmos o sinal $s[n]$ através do canal, recebemos sua versão distorcida $x[n]$, conforme a seguinte relação:

$$\boxed{x[n] = s[n] + 0{,}7s[n-1] - 0{,}04s[n-2].} \tag{1}$$

(a) A partir da Equação (1), determine a resposta ao impulso do canal $h[n]$.

Combinando os diagramas mostrados nas Figuras 1 e 2, o processo de equalização pode ser representado pela estrutura exibida na Figura 3.

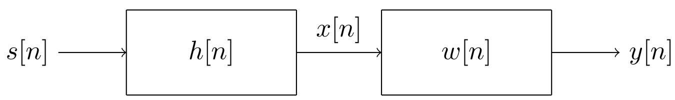

Figura 3: Processo de equalização.

Na situação ideal em que conseguimos equalizar completamente o canal e não há ruído, a saída do equalizador se torna o próprio sinal de entrada, ou seja:

$$\boxed{y[n] = s[n].} \tag{2}$$

(b) Considerando a situação de equalização perfeita, determine a resposta combinada canal-equalizador.
Dica: note que o canal $h[n]$ e o equalizador $w[n]$ são dois sistemas LIT em série (cascata).

Parte Computacional

(c) Vamos considerar agora dois filtros candidatos a equalizador, cujos coeficientes das respectivas respostas ao impulso são mostrados a seguir:

$$\mathbf{w}_1 = \left[\begin{array}{ccccccc} 1 & -0{,}7 & 0{,}7^2 & -0{,}7^3 & 0{,}7^4 & -0{,}7^5 & 0{,}7^6 \end{array}\right] \tag{3}$$

$$\mathbf{w}_2 = \left[\begin{array}{ccccccc} 1 & -0{,}2 & 0{,}4 & 0{,}9 & -0{,}3 & -0{,}9 & -0{,}8 \end{array}\right] \tag{4}$$

Apresente, então, a resposta combinada para cada um dos filtros usados e discuta a qualidade de cada filtro tendo em vista o objetivo desejado na tarefa de equalização. Utilize o comando `conv` no Matlab/Octave para calcular as convoluções discretas. Em Python, isto pode ser feito através do comando `np.convolve` (é preciso antes carregar o pacote numpy através do comando `import numpy as np`).

(d) Vamos, agora, simular uma transmissão e analisar novamente o papel dos filtros equalizadores. Para isso, crie uma sequência $s[n]$ com 500 símbolos pertencentes à modulação 4-QAM, isto é, com valores equiprováveis no alfabeto complexo $\{1+j, 1-j, -1+j, -1-j\}$. Em Matlab, isto pode ser feito através dos seguintes comandos:

```
alphabet = [1+1j 1-1j -1+1j -1-1j];
s = randsrc(1,500,alphabet);
```

Obs.: no Octave, é preciso antes carregar o pacote de comunicações, através do comando `pkg load communications`.

Em Python, isto pode ser feito através dos seguintes comandos:

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

Faça, então, a transmissão desse sinal pelo canal $h[n]$, cujo resultado é o vetor `x`, que contém as amostras do sinal recebido ($x[n]$). Exiba, então, no plano complexo os valores dos símbolos transmitidos (`s`) e dos símbolos observados na saída do canal (`x`). Em Matlab, isso pode ser feito com o comando `plot(real(s),imag(s),'.')`; já em Python, esse tipo de gráfico está implementado na biblioteca Matplotlib.pyplot e basta fazer `import matplotlib.pyplot as plt` para, depois, usar o comando `plt.scatter(x=np.real(s),y=np.imag(s))`.

(e) Adicione um pouco de ruído a $x[n]$, gerando o sinal recebido

$$r[n] = x[n] + \eta[n],$$

onde $\eta[n]$ é uma variável aleatória Gaussiana complexa com $\mathrm{Re}\{\eta[n]\} \sim N(0, 0.02^2)$ e $\mathrm{Im}\{\eta[n]\} \sim N(0, 0.02^2)$, com $\mathrm{Re}\{\eta[n]\} \perp\!\!\!\perp \mathrm{Im}\{\eta[n]\}$ (i.e., são estatisticamente independentes). Utilize os comandos indicados na Tabela 1 para obter $r[n]$.

Filtre, então, o sinal recebido com os equalizadores $w_1[n]$ e $w_2[n]$ (cujos coeficientes foram apresentados no item (c)), obtendo as saídas $y_1[n]$ e $y_2[n]$, respectivamente. Faça dois gráficos (*i.e.* duas figuras diferentes no Matlab ou em Python), detalhados a seguir:

- Gráfico 1: em uma mesma figura, plote o sinal de entrada $s[n]$ em azul e a saída $y_1[n]$ em vermelho.
- Gráfico 2: em uma mesma figura, plote o sinal de entrada $s[n]$ em azul e a saída $y_2[n]$ em vermelho.

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

Tabela 1: Trecho de código para gerar o sinal recebido com ruído.

Com base nestes dois gráficos, qual das saídas obtidas está mais próxima do sinal original $s[n]$?
Os seguintes comandos no Matlab podem ser empregados para a geração dos gráficos:

`figure()` – abre uma nova figura no Matlab
`stem()` – usado para plotar gráficos de valores discretos
`hold on` – comando do Matlab usado para plotar mais de um gráfico na mesma figura
`xlabel()` – atribui um nome ao eixo x
`ylabel()` – atribui um nome ao eixo y
`title()` – título do gráfico.

Em Python, os comandos são os mesmos que os do Matlab, sendo necessário colocar `plt.` no início.