

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia



Docentes:

José Nascimento

André Lourenço

Discentes:

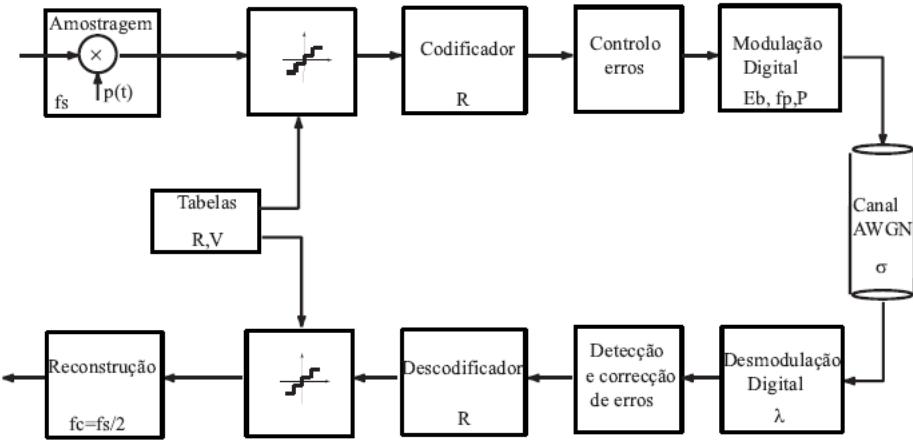
Ana Rodrigues nº40517

Cláudia Rodrigues nº41026

Sara Machás nº40567

Data: 14-11-2014

Esquema de blocos funcionais



1. Construa uma função que dado um array de N bits, para cada 4 bits gera 7 bits, sendo 3 bits resultado do código Hamming H(7,4).

```

#1)
def Hamming(sinal):
    Adicionar = (4-len(sinal)%4)%4
    sinal = np.hstack((sinal,np.ones(Adicionar)))
    sinalSaida = np.zeros((len(sinal)/4)*7)
    G = np.array([[1,0,0,0,1,1,1],[0,1,0,0,1,1,0],[0,0,1,0,1,0,1],[0,0,0,1,0,1,1]])
    for i in np.arange(len(sinal)/4):
        C=np.dot((sinal[i*4:((i+1)*4)]),G)%2 #dot = multiplica matrizes, e o resto
        sinalSaida[i*7:(i+1)*7] =C
    return sinalSaida
  
```

2. Construa uma função que dado um array M bits, para cada 7 detete e corrige possíveis erros, sendo o resultado 4 bits de mensagem.

```
#2)
def detectarCorrigirErros(SinalIn):
    sinal = np.array(SinalIn)
    sinalsaida = np.zeros((len(sinal)/7)*4)
    Ht = np.array([[1,1,1],[1,1,0],[1,0,1],[0,1,1],[1,0,0],[0,1,0],[0,0,1]])
    for i in np.arange(len(sinal)/7):
        Valor=np.dot((sinal[i*7:((i+1)*7)]),Ht)%2
        if (sum(Valor)!=0):
            index = -int(str(int(100* Valor[0]+10*Valor[1]+Valor[2])),2)
            #normalmente no código de Hamming ficam 3 bits do sinal, 1 redundante, 1 de S.
            #como na nossa implementação temos os 4 bits de sinal juntos e depois os 3 restando
            if index== -3:
                index = -4
            elif index == -4:
                index = -3
            valorCorrecto = (sinal[i*7:((i+1)*7)][index]+1)%2
            sinal[i*7:((i+1)*7)][index] = valorCorrecto
            sinalsaida[i*4:(i+1)*4]=sinal[i*7:i*7 + 4]
    return np.array(sinalsaida, dtype=int)
```

3. Com as duas funções anteriores e admitindo que os restantes blocos do emissor, canal e receptor possam ser simulados pela expressão “ $y = 1 * np.logical_xor(x, np.random.binomial(1, BER, len(x)))$ ”, onde BER contém o valor pretendido para o bit error rate. Meça SNR na recepção, o BER antes e após correção de erros, para diferentes valores de BER.

```
#3
def inserirErro(sinal,BER):
    y = 1*np.logical_xor(sinal, np.random.binomial(1,BER, len(sinal)))
    return y

def stringToArrayBin(string): #entra em string e sai em array de bits
    Array = np.ones(len(string))
    for a in np.arange(len(string)):
        Array[a] = int(string[a])
    return Array

if __name__=="__main__":
    fs7=8000
    t7= np.arange(fs7*1.)/fs7
    y7 = 1000*np.cos(2.0*np.pi*1008*t7)

    R=3
    valorMax=npamax(abs(y7))
    #Obter as tabelas
    vD, vQ = get_tabela(R, valorMax)
    #Quantificar o sinal com as tabelas
    sinalQuantificado, indiceQ = quantificacao(y7, vQ, vD)
    #Converter os indices do sinal quantificado para dar p converter para binario
    valorBinarioCodificado = arrayToBinario(indiceQ, R)

    #AQUI INSEREM-SE AS NOVAS FUNCIONALIDADES
    ArrayBins = stringToArrayBin (valorBinarioCodificado)

    sinalHamming = Hamming(ArrayBins)

    sinalErro = inserirErro(sinalHamming, 0.1)

    sinalCorrigido = detectarCorrigirErros(sinalErro)

    String = ArrayToString(sinalCorrigido)
    #Voltar a reconverter os indices em binario para valores decimais
    arrayDescodificado = binarioToArray(String,R)
    #Volta a obter-se as amplitudes do sinal, apartir dos indices e da tabelas de valores de quantificacao
    SinalFinal = DescodificacaoDoSinal(vQ, arrayDescodificado)

    SNR = SNRqdBPratica(y7, SinalFinal)
    print SNR
```

4. Admita o bloco com a mensagem [1,0,1,0]

- Através do polinómio gerador $x^3 + x + 1$ gere a mensagem com os bits de controlo errados.

Resolução:

$$\begin{array}{r}
 1x^6 \ 0x^5 \ 1x^4 \ 0x^3 \ 0x^2 \ 0x \ 0 \\
 x^6 \quad -x^4 \quad -x^3 \\
 \hline
 x^3 \quad +1
 \end{array}$$

$$\begin{array}{r}
 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \\
 \quad -x^3 \quad -0 \quad -x \quad -1 \\
 \hline
 \textcolor{blue}{0} \quad \textcolor{blue}{0} \quad \textcolor{blue}{1} \quad \textcolor{blue}{1}
 \end{array}$$

- Transmite-se 1 0 1 0 **0 1 1**

- Admita que recebe a mensagem [1, 1, 0, 1, 1, 0, 1]. O que pode concluir?

Resolução:

$$\begin{array}{r}
 1x^6 \ 1x^5 \ 0x^4 \ 1x^3 \ 1x^2 \ 0x \ 1 \\
 -x^6 \ -0 \ -x^4 \ -x^3 \ 0 \ 0 \ 0 \\
 \hline
 x^3 \quad +x^2 \quad +x \quad +1
 \end{array}$$

$$\begin{array}{r}
 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \\
 \quad -x^5 \quad 0 \quad -x^3 \quad -x^2 \quad 0 \quad 0 \\
 \hline
 \textcolor{blue}{0} \quad \textcolor{blue}{1} \quad \textcolor{blue}{1} \quad 0 \quad 0 \quad 1
 \end{array}$$

$$\begin{array}{r}
 \textcolor{blue}{-x^4} \quad 0 \quad -x^2 \quad -x \quad 0 \\
 \hline
 0 \quad 1 \quad 1 \quad 1 \quad 1
 \end{array}$$

$$\begin{array}{r}
 -x^3 \quad 0 \quad -x \quad 1 \\
 \hline
 0 \quad \textcolor{blue}{1} \quad 0 \quad 0
 \end{array}$$

Houve erro uma vez que o valor é diferente de zero.

- Admitindo que o sistema transmissão funciona a 10Mbit/s com um BER inicial de 10^{-3} , qual o BER após a aplicação do controle de erros.

Resolução:

$$\text{BER} = 10^{-6}; H(7,4) \text{ onde } n = 7$$

Então:

$$\text{BER}'s = \frac{3}{7} \left(\frac{7*6}{2} * (10^{-3})^2 \right) = 9 * 10^{-6}$$

O BER diminui logo o número de bits errados é menor.

- Nas condições da alínea anterior, qual o tempo médio entre dois bits errados (com e sem deteção de erros).

Resolução:

$$B_t = 10 \text{kbits} = 10 * 10^6 \text{ Bits}$$

$$\text{BER} = 10^{-3}$$

$$\text{BER}'s = 9 * 10^{-6}$$

Sem deteção de erro:

$$10 * 10^6 \text{ bits} \quad \text{---} \quad 1\text{s}$$

$$10^{-3} \text{ bits} \quad \text{---} \quad x1$$

$$x1 = 1 * 10^{-10} \text{ segundos entre bits errados}$$

Com deteção de erros:

$$10 * 10^6 \text{ bits} \quad \text{---} \quad 1\text{s}$$

$$9 * 10^{-3} \text{ bits} \quad \text{---} \quad x2$$

$$x2 = 9 * 10^{-10} \text{ segundos entre bits errados}$$