INSTITUTO FEDERAL
Santa Catarina
Câmpus São José

Avaliação 3

Codigos convolucionais

Disciplina: COM029008 - SISTEMAS DE COMUNICAÇÃO II (2025 .2 - T01)
Professor: Roberto Wanderley da Nóbrega
Aluno: Wagner Santos

Novembro de 2025

Sumário

1. Questão 1

Considere o código convolucional com matrizes geradoras dadas por

$$G0 = [111], G1 = [110], G2 = [011]. \tag{1}$$

1.1. (a) Determine a taxa e a ordem de memória do código convolucional.

O código utiliza um bit de entrada por vez ($k = 1$) e produz três bits de saída ($n = 3$). Portanto, a taxa é:

$R = \frac{k}{n} = \frac{1}{3}$

Os polinômios possuem três coeficientes cada, o que implica dependência de u_t, u_{t-1} e u_{t-2}. Sendo assim, no pior caso o atraso será de 2 bits. Portanto, a ordem é:

$m = 2$

1.2. (b) Esboce o diagrama de blocos do codificador.

Tem-se:

$$\begin{aligned} v_t &= u_t G_0 + u_{t-1} G_1 + u_{t-2} G_2 \\ &= u_t.[111] + u_{t-1}.[110] + u_{t-2}.[011] \\ &= [u_t u_t u_t] + [u_{t-1} u_{t-1} 0] + [0 u_{t-2} u_{t-2}] \\ &= [u_t + u_{t-1}, u_t + u_{t-1} + u_{t-2}, u_t + u_{t-2}] \end{aligned} \tag{2}$$

Portando:

$$\begin{aligned} v_t^0 &= u_t + u_{t-1} \\ v_t^1 &= u_t + u_{t-1} + u_{t-2} \\ v_t^2 &= u_t + u_{t-2} \end{aligned} \tag{3}$$

Diagrama de blocos correspondente:

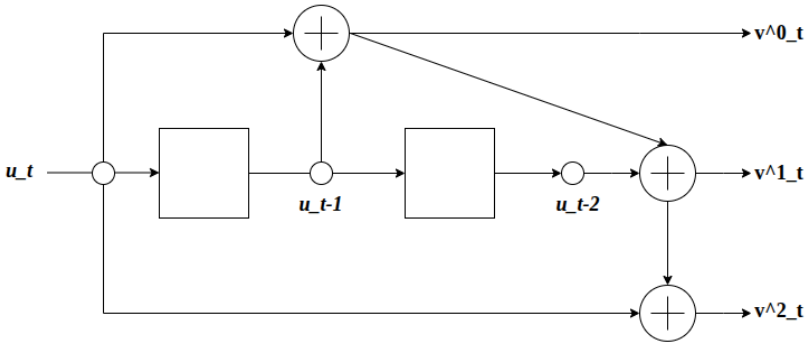


Figure 1: Elaborada pelo autor

1.3. (c) Esboce o diagrama de estados do código.

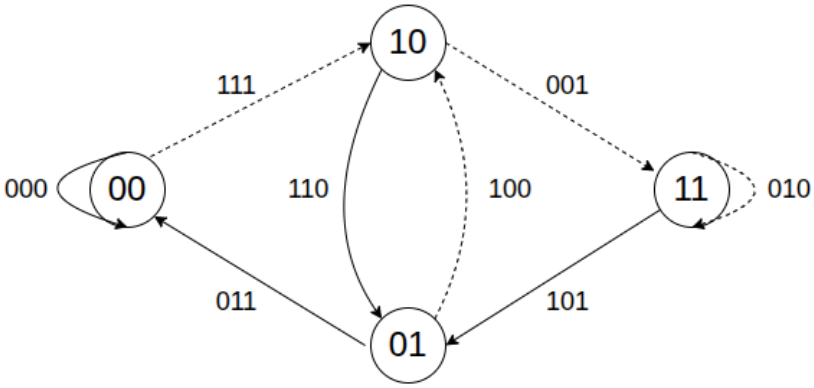


Figure 2: Elaborada pelo autor

1.4. (d) Determine os parâmetros (n', k') do código de bloco resultante, com $h = 4$ blocos e terminação no estado zero.

A dimensão do código de bloco resultante é:

$$k' = hk = 4.1 = 4 \tag{4}$$

e o comprimento é:

$$n' = (h + m)n = (4 + 2).3 = 18 \tag{5}$$

1.5. (e) Codifique a mensagem 0101. Insira a cauda apropriada.

Mensagem original:

$$u = 0101 \tag{6}$$

Adicionar 2 bits da cauda = 00

$$u = 010100 \tag{7}$$

Codificação de cada bit, usando:

$$\begin{aligned} v_t^0 &= u_t + u_{t-1} \\ v_t^1 &= u_t + u_{t-1} + u_{t-2} \\ v_t^2 &= u_t + u_{t-2} \end{aligned} \tag{8}$$

Cálculo das saídas:

t	Estado atual	u_t	Tipo de linha	Ramo usado	Próx. estado	Saída
0	00	0	cheia	$00 \to 00$	00	000
1	00	1	tracejada	$00 \to 10$	10	111
2	10	0	cheia	$10 \to 01$	01	110
3	01	1	tracejada	$01 \to 10$	10	100
4	10	0	cheia	$10 \to 01$	01	110
5	01	0	cheia	$01 \to 00$	00	011

Sequência codificada:

$$000\ 111\ 110\ 100\ 110\ 011 \tag{9}$$

1.6. (f) Decodifique a palavra recebida 010 111 110 101 110 011 utilizando o algoritmo de Viterbi.

A treliça correspondente ao diagrama de estados é mostrada abaixo:

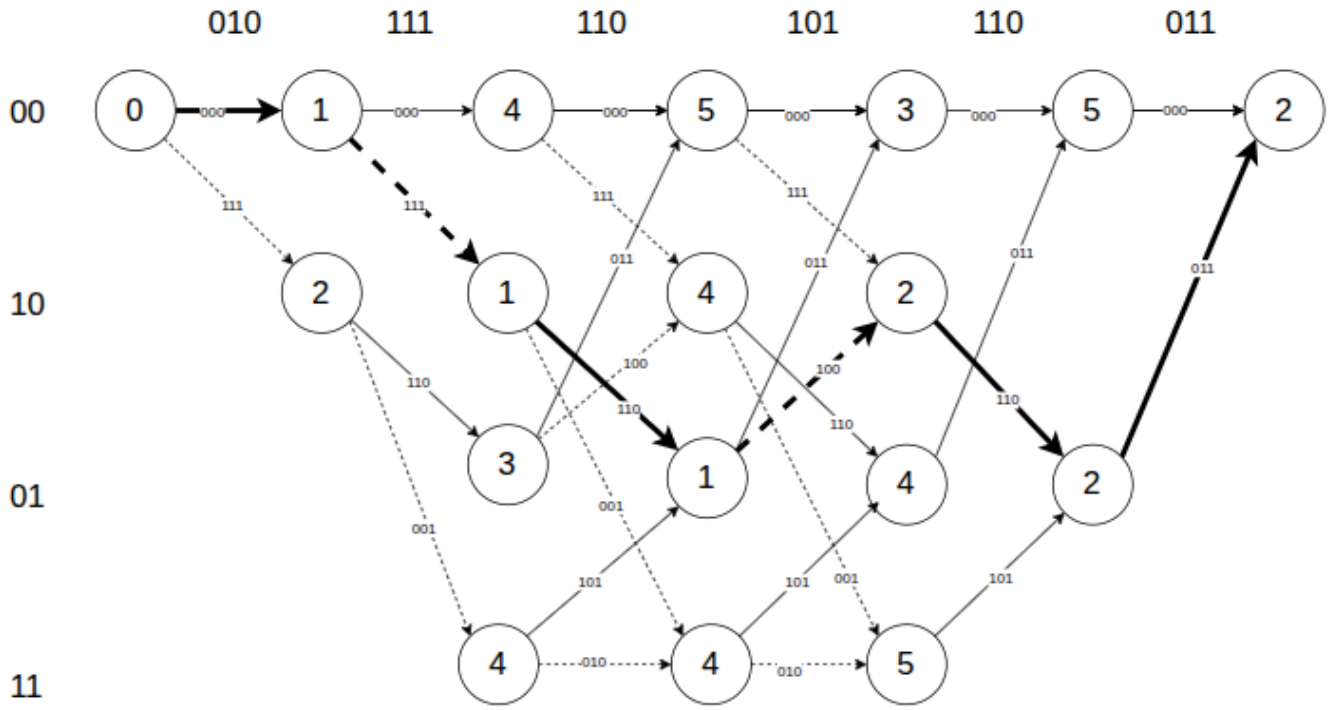


Figure 3: Elaborada pelo autor

$$\hat{v} = 000 \quad 111 \quad 110 \quad 100 \quad 110 \quad 011 \quad (2 \text{ erros}) \tag{10}$$

$$\hat{u}_{\text{cauda}} = 010100 \tag{11}$$

$$\hat{u} = 010100 \tag{12}$$

1.7. (g) Determine a função de transferência de peso do código.

Peso de Hamming na saída de cada ramo:

$$\begin{aligned} w(000) = 0, w(111) = 3, w(110) = 2, w(001) = 1, \\ w(011) = 2, w(100) = 1, w(101) = 2, w(010) = 2 \end{aligned} \tag{13}$$

O diagrama de Mason:

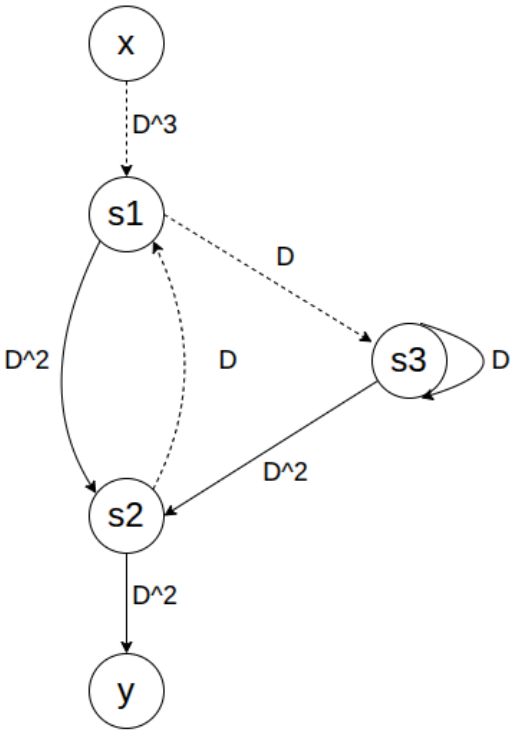


Figure 4: Elaborada pelo autor

As equações de estado são:

$$\begin{aligned} s_1 &= D^3x + Ds2 \\ s_2 &= D^2s_1 + D^2s_3 \\ s_3 &= Ds_1 + Ds_3 \\ y &= D^2s_2 \end{aligned} \tag{14}$$

Resolvendo o sistema:

s_3 :

$$s_3 = Ds_1 + Ds_3 \rightarrow s_3 - Ds_3 = Ds_1 \rightarrow s_3(1-D) = Ds_1 \rightarrow s_3 = \left(\frac{D}{1-D}\right)s_1 \tag{15}$$

s_2 :

$$s_2 = D^2 s_1 + D^2 s_3 = D^2 s_1 + D^2 . \left(\frac{D}{1-D} \right) s_1 = \left(D^2 + \frac{D^3}{1-D} \right) s_1$$

Mesmo denominador:

$$D^2 + \left(\frac{D^3}{1+D} \right) = \left(D^2 \frac{1-D}{1} - D \right) + (\frac{D^3}{1-D} = \frac{D^2 - D^3 + D^3}{1-D} = \frac{D^2}{1-D} \quad (16)$$

Logo:

$$s_2 = \left(\frac{D^2}{1-D} \right) s^1$$

s_1 :

$$\begin{aligned} s_1 &= D^3 x + D^2 s_2 = D^3 x + D . \left(\frac{D^2}{1-D} \right) s_1 \\ &= D^3 x + \left(\frac{D^3}{1-D} \right) s_1 = \left(\frac{D^3 (1-D)}{1-D-D^3} \right) x \end{aligned} \quad (17)$$

Finalmente $\frac{y}{x}$:

$$y = D^2 s_2 = D^2 . \left(\frac{D^2}{1-D} \right) s_1 = \left(\frac{D^4}{1-D} \right) s_1$$

Substituindo s_1 :

$$y = \left(\frac{D^4}{1-D} \right) . \left(\frac{D^3 (1-D)}{1-D-D^3} \right) x = \left(\frac{D^7}{1-D-D^3} \right) x \quad (18)$$

Logo:

$$T(D) = \frac{y}{x} = \left(\frac{D^7}{1-D-D^3} \right)$$

Escrevendo $T(D)$ como série de potências em D:

$$T(D) = D^7 + D^8 + D^9 + 2D^{10} + 3D^{11} + 4D^{12} + ... \quad (19)$$

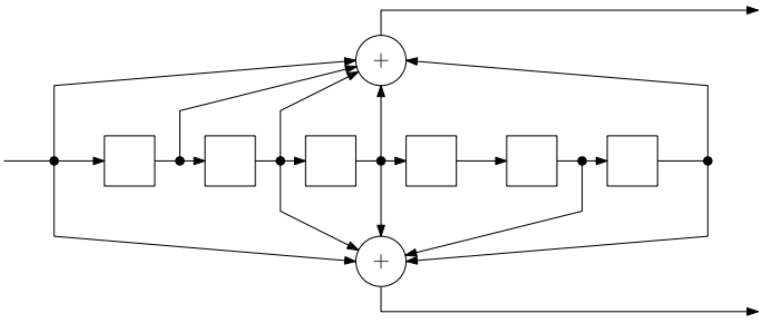
Assim, conclui-se que:

- a menor potência de D é 7, logo $d_{\text{free}} = 7$;
- há 1 sequência-código de peso 7;
- há 1 sequência-código de peso 8;

- há 1 sequência-código de peso 9;
- há 2 sequências-código de peso 10;
- etc.

2. Questão 2

Escreva um programa que simule a probabilidade de erro de bit de um sistema de comunicação que utiliza o código convolucional mostrado na figura abaixo com decodificação via algoritmo de Viterbi no canal BSC(p). Considere a transmissão de 1000 quadros, cada qual contendo h = 200 blocos de informação e p variando de 0 a $\frac{1}{2}$.



2.1. Código Python

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

# ----------------------------
# Parâmetros da simulação
# ----------------------------
h = 200                # bits de informação por quadro
num_frames = 1000      # número de quadros transmitidos
p_values = np.linspace(0.0, 0.5, 11)  # p = 0, 0.05, ..., 0.5

# ----------------------------
# Código convolucional da figura:
# rate 1/2, K = 7, geradores (171,133)_8
# ----------------------------
convcode = komm.ConvolutionalCode([[0o171, 0o133]])
# equivalentes a [[0b1111001, 0b1011011]]

# Constrói código convolucional TERMINADO com h bits de informação
code = komm.TerminatedConvolutionalCode(
    convolutional_code=convcode,
    num_blocks=h,
    mode="zero-termination",
)
```

```
print("Dimensão (bits de informação por quadro):", code.dimension)
print("Comprimento (bits codificados por quadro):", code.length)

# Decodificador Viterbi para esse código em bloco
decoder = komm.ViterbiDecoder(code)

# ----------------------------
# Canal BSC(p)
# ----------------------------
def bsc(bits, p):
    """
    bits: array de 0/1
    p: probabilidade de inversão
    retorna: bits passados pelo BSC(p)
    """
    noise = np.random.rand(bits.size) < p  # ruído Bernoulli(p)
    return (bits ^ noise.astype(int))      # XOR (soma módulo 2)

# ----------------------------
# Loop de simulação
# ----------------------------
ber_results = []  # lista para guardar BER para cada p

for p in p_values:
    bit_errors = 0
    total_bits = h * num_frames  # bits de informação transmitidos

    for _ in range(num_frames):
        # 1) Gera bloco de informação aleatório
        u = np.random.randint(0, 2, size=h)

        # 2) Codifica com o código convolucional terminado
        x = code.encode(u)  # tamanho = code.length

        # 3) Passa pelo canal BSC(p)
        y = bsc(x, p)

        # 4) Decodifica com Viterbi
        u_hat = decoder.decode(y)

        # 5) Conta erros de bit na mensagem de informação
        bit_errors += np.sum(u != u_hat)

    ber = bit_errors / total_bits
    ber_results.append(ber)
    print(f"p = {p:.2f} -> BER estimado = {ber:.5f}")


plt.semilogy(p_values, ber_results, marker="o")
plt.grid(True, which="both")
plt.xlabel("p (probabilidade de inversão no BSC)")
plt.ylabel("Probabilidade de erro de bit (BER)")
```

```
plt.title("Simulação BER – código convolucional (171,133)_8 com Viterbi")
plt.show()
```

2.2. Resultados

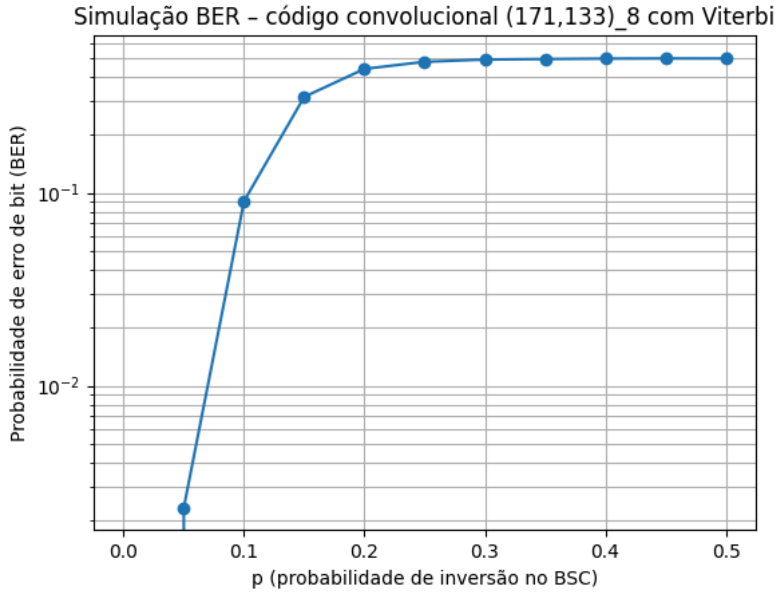


Figure 6: Plot

```
Dimensão (bits de informação por quadro): 200
Comprimento (bits codificados por quadro): 412
p = 0.00 -> BER estimado = 0.00000
p = 0.05 -> BER estimado = 0.00233
p = 0.10 -> BER estimado = 0.09067
p = 0.15 -> BER estimado = 0.31472
p = 0.20 -> BER estimado = 0.43950
p = 0.25 -> BER estimado = 0.47846
p = 0.30 -> BER estimado = 0.49136
p = 0.35 -> BER estimado = 0.49427
p = 0.40 -> BER estimado = 0.49692
p = 0.45 -> BER estimado = 0.49791
p = 0.50 -> BER estimado = 0.49766
```

Figure 7: Saída terminal