Codificação de Canal

Codificação de Canal

- **<u>Codificação de Canal</u>**
  - Permite a detecção e correção de erros introduzidos pelos canal
  - <u>Tipos de Códigos</u>
    - Códigos de bloco
    - Códigos convolucionais
    - Modulação codificada (TCM - BCM)
    - Códigos turbo
    - Códigos espaçotemporais

Códigos de Bloco

- **<u>Códigos de Bloco</u>**
  - <u>Palavra código</u> = <u>informação</u> + <u>redundância</u>
  - Os códigos de bloco são especificados como códigos (n,k)
  - k = número de bits de informação
  - n = número de bits da palavra código
  - n-k = número de bits de redundância

Códigos de Bloco

- **Detecção de Erros**
  - Quando a palavra código recebida é inválida

- **Erros Não Detectáveis**
  - Quando a palavra código transmitida se transforma numa outra palavra código

Códigos de Bloco

- **<u>Ações para erros detectados</u>**
  - Solicitar a retransmissão da palavra
    - Automatic Repeat Request (ARQ)
  - Marcar a palavra como sendo incorreta e mandar adiante
    - Muting
  - Tentar corrigir os erros da palavra recebida
    - Forward Error Correction (FEC)

Códigos de Bloco

- **<u>Propriedades dos Códigos de Bloco</u>**
  - <u>Distância Mínima do Código</u>
  - Para códigos binários a distância mínima também é denominada de

    Distância de Hamming
  - A habilidade de correção de erros de um código de bloco é uma função da distância mínima do código

Códigos de Bloco

- **Estrutura de Codificação**
  - Matriz Geradora

$$G = \left[P \,|\, I_k \right] \qquad k \times n$$

$P = matriz\ \ de\ coeficientes$

$I = matriz\ \ identidade$

  - Processo de Codificação

$$v = m \cdot G$$

m = palavra de informação

v = palavra código

Códigos de Bloco

- **<u>Matriz de Verificação de Paridade</u>**

$$G = \left[P \mid I_k\right] \qquad k \times n$$

$$H = \left[I_{n-k} \mid P^T\right] \qquad (n-k) \times n$$

A matriz H é construída de tal maneira que:

$$v \cdot H^T = 0$$

Matriz Geradora (G) e Matriz de Paridade (H) – Exemplo C(7,4)

$$\boldsymbol{G} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{G} = \left[\underbrace{\begin{matrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{matrix}}_{\text{submatrix } \boldsymbol{P}} \quad \underbrace{\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}}_{\text{submatrix } \boldsymbol{I}}\right]$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Palavras-código, C(7,4)

Message				Codewords						
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	1
0	0	1	0	1	1	1	0	0	1	0
0	0	1	1	0	1	0	0	0	1	1
0	1	0	0	0	1	1	0	1	0	0
0	1	0	1	1	1	0	0	1	0	1
0	1	1	0	1	0	0	0	1	1	0
0	1	1	1	0	0	1	0	1	1	1
1	0	0	0	1	1	0	1	0	0	0
1	0	0	1	0	1	1	1	0	0	1
1	0	1	0	0	0	1	1	0	1	0
1	0	1	1	1	0	0	1	0	1	1
1	1	0	0	1	0	1	1	1	0	0
1	1	0	1	0	0	0	1	1	0	1
1	1	1	0	0	1	0	1	1	1	0
1	1	1	1	1	1	1	1	1	1	1

Códigos de Bloco

- **<u>Síndrome e Detecção de Erros</u>**

$$v = \text{palavra} \quad \text{código}$$

$$e = \text{padrão de erro (vetor erro)}$$

$$r = v + e$$

$$r \cdot H^T = v \cdot H^T + e \cdot H^T$$

$$s = e \cdot H^T = \text{síndrome}$$

Cálculo da síndrome – C(7,4)

Vetor recebido

$$\boldsymbol{r} = (r_0, r_1, r_2, r_3, r_4, r_5, r_6)$$

$$\boldsymbol{S} = (s_0, s_1, s_2) = (r_0, r_1, r_2, r_3, r_4, r_5, r_6) \circ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

$$s_0 = r_0 \oplus r_3 \oplus r_5 \oplus r_6$$

$$s_1 = r_1 \oplus r_3 \oplus r_4 \oplus r_5$$

$$s_2 = r_2 \oplus r_4 \oplus r_5 \oplus r_6$$

Exemplo para determinar os padrões de erro - S=(0 0 1)

$$0 = e_0 \oplus e_3 \oplus e_5 \oplus e_6$$
$$0 = e_1 \oplus e_3 \oplus e_4 \oplus e_5$$
$$1 = e_2 \oplus e_4 \oplus e_5 \oplus e_6$$

e_0	e_1	e_2	e_3	e_4	e_5	e_6
0	0	1	0	0	0	0
1	1	1	1	0	0	0
1	0	0	0	0	0	1
0	1	0	1	0	0	1
0	0	0	1	0	1	0
1	1	0	0	0	1	0
0	1	1	0	0	1	1
1	0	1	1	0	1	1
0	1	0	0	1	0	0
1	0	0	1	1	0	0
1	1	1	0	1	0	1
0	0	1	1	1	0	1
1	0	1	0	1	1	0
0	1	1	1	1	1	0
1	1	0	1	1	1	1
0	0	0	0	1	1	1

Padrões de erro – C(7,4)

Error patterns							Syndromes		
1	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	1	0
0	0	0	0	1	0	0	0	1	1
0	0	0	0	0	1	0	1	1	1
0	0	0	0	0	0	1	1	0	1

Exemplo

Considere que a sequência

$$r = (1\ 0\ 1\ 0\ 0\ 1\ 1)$$

é recebida por um receptor. Determine se houve erro na transmissão e caso tenha havido, determine qual a informação que foi tranmsitida.

Códigos de Bloco

- **Capacidade de Detecção de Erros**

$$\text{número de erros detetáveis} \leq \left(d_{\min} - 1\right)$$

- **Capacidade de Correção de Erros**

$$t \leq \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

  - O código corrige todos os padrões de erros com *t* ou menos erros

Códigos de Bloco

- **<u>Famílias de Códigos de Blocos</u>**
  - Códigos de Hamming
  - Códigos Cíclicos
  - Códigos de Hadamard
  - Códigos de Golay
  - Códigos BCH
  - Códigos Reed-Solomon
  - Código Reed-Muller

Exercício

Considerando a matriz geradora de um código de bloco, dada a seguir, determine qual o vetor síndrome para a sequência ‘0 1 1 0 1 0 0’ e qual a taxa deste código. Sabendo que sua distância mínima é 3, calcule quantos erros este código pode corrigir.

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Códigos de Bloco Lineares

Help Matlab: encode

ENCODE Block encoder.

CODE = ENCODE(MSG, N, K, METHOD, OPT) encodes MSG using an error-control

coding technique. For information about the parameters and about using a

specific technique, type one of these commands at the MATLAB prompt:

```
FOR DETAILS, TYPE       CODING TECHNIQUE
  encode hamming          % Hamming
  encode linear           % Linear block
  encode cyclic           % Cyclic
```

Códigos de Bloco Lineares

Help Matlab: encode linear

ENCODE Encodes a message using linear block code method.
CODE = ENCODE(MSG, N, K, METHOD, GEN), METHOD = 'linear', encodes the binary message in MSG using the linear block code method. The codeword length is N and the message length is K. The format of MSG can be either a vector or K-column matrix. The generator matrix GEN is a K-by-N matrix. Linear block code is a generic code. For example, You can use HAMMGEN function to generate a generator matrix for Hamming code.

CODE = ENCODE(MSG, N, K, METHOD, GEN), METHOD = 'linear/decimal', specifies that the input data in CODE is decimal integers. This function converts the decimal integer into M bits binary before processing the encode computation, where M is the smallest integer such that N <= 2^M-1.

[CODE, ADDED] = ENCODE(...) outputs the number of columns added to the input variable MSG in order to make the MSG fit for encoding.

Códigos de Hamming

Parâmetros

Length	$n = 2^m - 1$
Number of message bits	$k = 2^m - m - 1$
Number of parity check bits	$n - k = m$
Error-correction capability	$t = 1, (d_{\min} = 3)$

Exemplo de um código de Hamming: C(7,4)

$$n = 2^3 - 1 = 7$$

$$k = 2^3 - 3 - 1 = 4$$

$$n - k = m = 3$$

$$t = 1(d_{\min} = 3)$$

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Códigos de Hamming

Help Matlab: encode hamming

ENCODE Encodes a message using Hamming code method.

CODE = ENCODE(MSG, N, K, METHOD), METHOD = 'hamming', encodes the binary codeword in CODE using the Hamming code method. The codeword length is N and the message length is K. The format of MSG can be either a vector or K-column matrix. Hamming code is a single error-correction code. Its codeword length is N = 2^M-1. Its message length is N-M.

CODE = ENCODE(MSG, N, K, METHOD, PRIM_POLY), METHOD = 'hamming', specifies the primitive polynomial used in the Hamming encode. PRIM_POLY is a degree N polynomial defined in GF(2).

CODE = ENCODE(MSG, N, K, METHOD...), METHOD = 'hamming/decimal', specifies that the input data in MSG is decimal integers. This function converts the decimal integer into M bits binary before processing the encode computation, where M is the smallest integer such that N <= 2^M-1.

[CODE, ADDED] = ENCODE(...) outputs the number of columns added to the input variable MSG in order to make the MSG fit for encoding.

Códigos de Hamming

help Matlab: hammgen

HAMMGEN Produce parity-check and generator matrices for Hamming code.
H = HAMMGEN(M) produces the parity-check matrix H for a given integer M, M >= 3. The code length of a Hamming code is N=2^M-1. The message length is K = 2^M - M - 1. The parity-check matrix is an M-by-N matrix.

H = HAMMGEN(M, P) produces the parity-check matrix using a given GF(2) primitive polynomial P.

[H, G] = HAMMGEN(...) produces the parity-check matrix H as well as the generator matrix G. The generator matrix is a K-by-N matrix.

[H, G, N, K] = HAMMGEN(...) produces the codeword length N and the message length K.

Note: The parameter M must be an integer greater than or equal to 3.
Hamming code is a single-error-correction code.

Códigos Cíclicos

Help Matlab: encode cyclic

ENCODE encodes a message using cyclic code method.

CODE = ENCODE(MSG, N, K, METHOD, GENPOLY), METHOD = 'cyclic', encodes binary message in MSG using the cyclic code method. The codeword length is N and the message length is K. The format for MSG can be either a vector or K-column matrix. GENPOLY is a degree N-K cyclic polynomial. You can use function CYCLPOLY to produce the cyclic polynomial.

CODE = ENCODE(MSG, N, K, METHOD...), METHOD = 'cyclic/decimal', specifies that the input data in MSG is decimal integers. This function converts the decimal integer into M bits binary before processing the encode computation, where M is the smallest integer such that N <= 2^M-1.

[CODE, ADDED] = ENCODE(...) outputs the number of columns added to the input variable MSG in order to make the MSG fit for encoding.

Códigos Cíclicos

Help Matlab: cyclpoly

```
CYCLPOLY Produce generator polynomials for a cyclic code.
   POL = CYCLPOLY(N, K) finds one cyclic code generator polynomial for a given
   codeword length N and message length K.  POL represents the polynomial
   by listing its coefficients in order of ascending exponents.

   POL = CYCLPOLY(N, K, OPT) finds cyclic code generator polynomial(s) for
   a given code word length N and message length K. The flag OPT
   means:
   OPT = 'min'  find one generator polynomial with the smallest possible weight.
   OPT = 'max'  find one generator polynomial with the greatest possible weight.
   OPT = 'all'  find all generator polynomials for the given codeword length
                and message length.
   OPT = L      find all generator polynomials with weight L.

   If OPT = 'all' or L, and more than one generator polynomial satisfies
   the constraints, then each row of POL represents a different polynomial.
   If no generator polynomial satisfies the constraints, then POL is empty.

   A divisor of X^N-1 generates a cyclic code of codeword length N.
```

Códigos Cíclicos

Help Matlab: cyclgen

CYCLGEN Produce parity-check and generator matrices for cyclic code.

H = CYCLGEN(N, P) produces the parity-check matrix for a given codeword length N and generator polynomial P. The vector P gives the binary coefficients of the generator polynomial in order of ascending powers. A polynomial can generate a cyclic code if and only if it is a factor of X^N-1. The message length of the code is K = N - M, where M is the degree of P. The parity-check matrix is an M-by-N matrix.

H = CYCLGEN(N, P, OPT) produces the parity-check matrix based on the instruction given in OPT. When OPT = 'nonsys', the function produces a nonsystematic cyclic parity-check matrix; OPT = 'system', the function produces a systematic cyclic parity-check matrix. This option is the default.

[H, G] = CYCLGEN(...) produces the parity-check matrix H as well as the generator matrix G. The generator matrix is a K-by-N matrix, where
K = N - M;

[H, G, K] = CYCLGEN(...) produces the message length K.

Códigos Reed-Solomon

help Matlab: rsenc

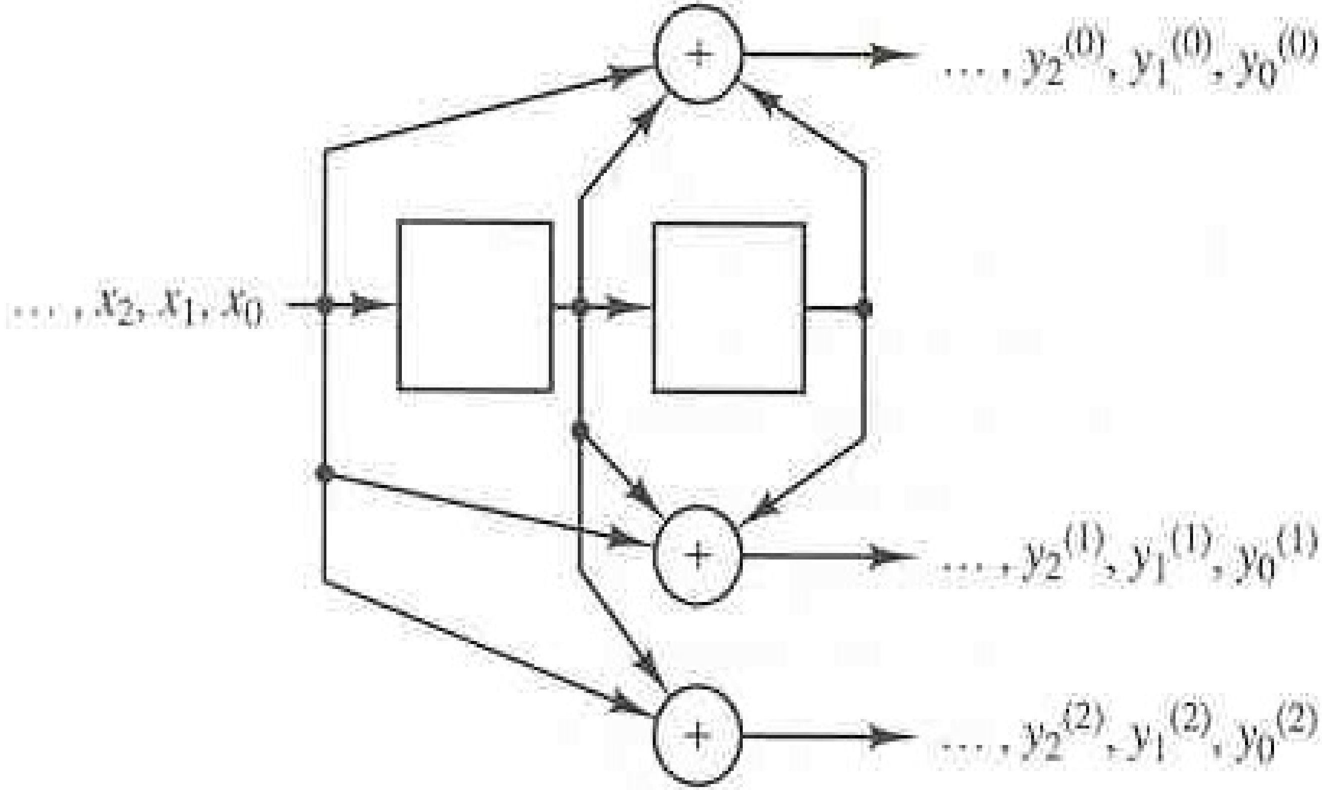
RSENC Reed-Solomon encoder.
CODE = RSENC(MSG,N,K) encodes the message in MSG using an (N,K) Reed-Solomon encoder with the narrow-sense generator polynomial. MSG is a Galois array of symbols over GF(2^m). Each K-element row of MSG represents a message word, where the leftmost symbol is the most significant symbol. If N is smaller than 2^m-1, then RSENC uses a shortened Reed-Solomon code. Parity symbols are at the end of each word in the output Galois array code.

CODE = RSENC(MSG,N,K,GENPOLY) is the same as the syntax above, except that a nonempty value of GENPOLY specifies the generator polynomial for the code. In this case, GENPOLY is a Galois row vector that lists the coefficients, in order of descending powers, of the generator polynomial. The generator polynomial must have degree N-K. To use the default narrow-sense generator polynomial, set GENPOLY to [].

CODE = RSENC(...,PARITYPOS) specifies whether RSENC appends or prepends the parity symbols to the input message to form code. The string PARITYPOS can be either 'end' or 'beginning'. The default is 'end'.

Códigos Convolucionais

- **<u>Exemplo – Codificador:</u>** R=1/3 ν=2 (3,1,2) = (n,k,ν), onde ν é comprimento de restrição



Códigos Convolucionais

- **Exemplo** - R=1/3 ν=2 (3,1,2)

$$x = (1011) \qquad g^0 = (111) \quad g^1 = (111)$$

$$g^2 = (110)$$

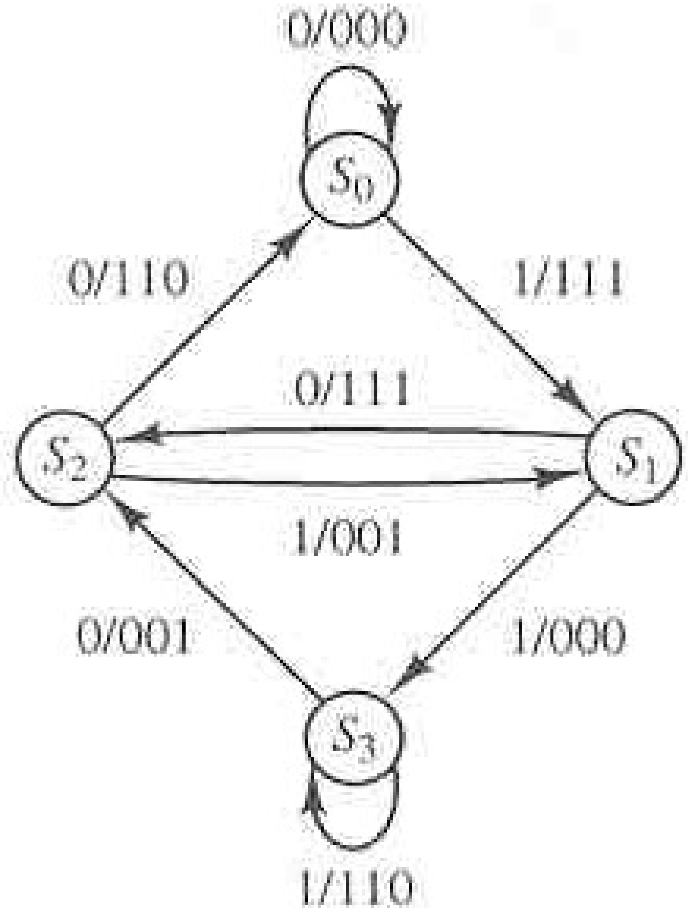
$$y^i = x * g^i \qquad y^0 = (1100) \quad y^1 = (1100)$$

$$y^2 = (1110)$$

$$y = (111, 111, 001, 000)$$

Códigos Convolucionais

- **Exemplo - Diagrama de Estados**



$$S_0 = (00)$$

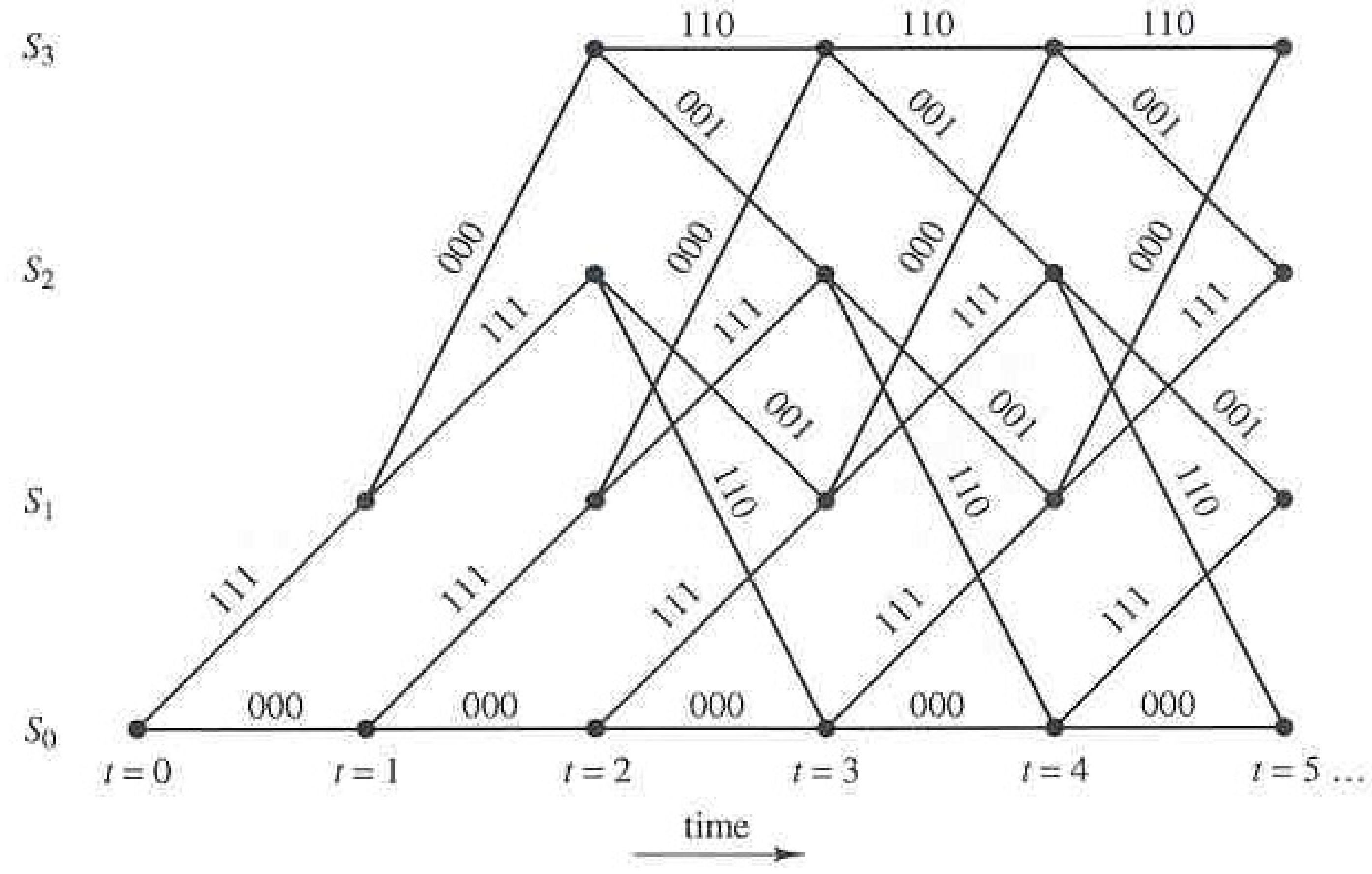
$$S_1 = (10)$$

$$S_2 = (01)$$

$$S_3 = (11)$$

Códigos Convolucionais

- **Exemplo - Treliça**



Códigos Convolucionais

- **<u>Representação Polinomial</u>**

$$x(D) = 1 + D^2 + D^3$$

$$g^0(D) = g^1(D) = 1 + D + D^2 \qquad g^2(D) = 1 + D$$

$$G(D) = \left[g^0(D)\; g^1(D)\; g^2(D)\right]$$

$$y(D) = x(D) \cdot G(D)$$

- Código Sistemático: $G(D) = \left[I\;\; P(D)\right]$

Códigos Convolucionais

- **Distância Livre**

  O desempenho de um código convolucional é função da distância livre: d_{free}

  A distância livre é definida como a mínima distância de Hamming entre duas palavras código.

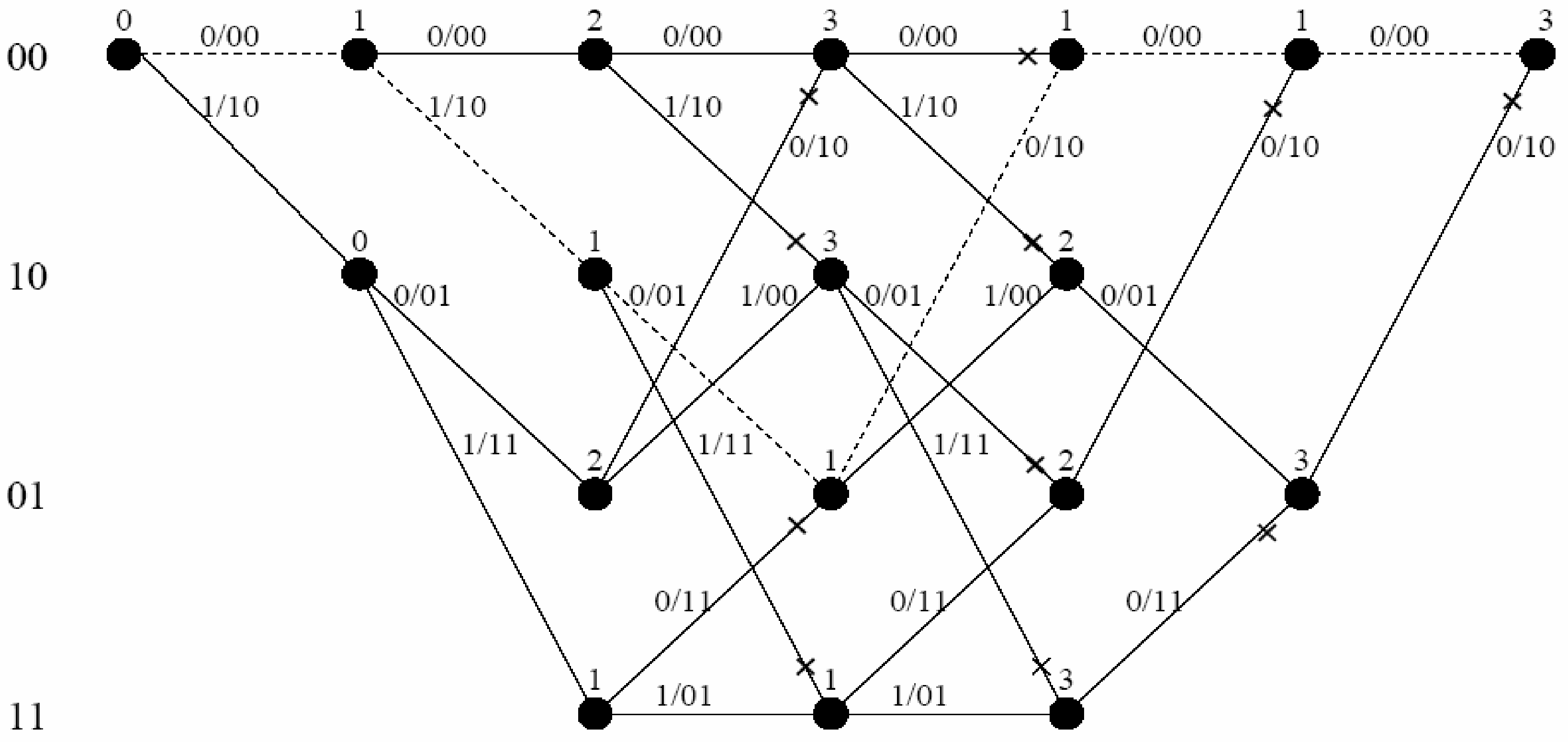
  Um segundo parâmetro importante é o espectro de distâncias (multiplicidades)

Códigos Convolucionais

- **Decodificação**
  - O método mais utilizado é a aplicação do “Algoritmo de Viterbi”
  - O Algoritmo de Viterbi busca pela seqüência mais provável através da treliça do código.

  - Outras opções: BCJR, SOVA, Fano

Algoritmo de Viterbi



$$r = (1\ 0,\ 1\ 0,\ 0\ 1,\ 1\ 0,\ 0\ 0,\ 1\ 1).$$

Exercício:

Um sistema de comunicação digital utiliza um codificador convoluvional. Os geradores polinomiais do código utilizado são: g1=(0 1 1) e g2=(1 0 1). Com esta informação responda os seguintes itens:

- Desenhe o codificador que está representado pelos polinômios acima;
- Determine o comprimento de restrição e a taxa deste código.
- Quantos estados este codificador pode assumir?
- Seria possível melhorar o desempenho do sistema utilizando um código com comprimento de restrição maior?

Códigos Convoluvionais

Help Matlab: convenc

CONVENC Convolutionally encode binary data.

CODE = CONVENC(MSG,TRELLIS) encodes the binary vector MSG using the convolutional encoder defined by the MATLAB structure TRELLIS. See POLY2TRELLIS and ISTRELLIS for a valid TRELLIS structure. The encoder starts at the all-zeros state. Each symbol in MSG consists of log2(TRELLIS.numInputSymbols) bits. MSG may contain one or more symbols. CODE is a vector in the same orientation as MSG, and each of its symbols consists of log2(TRELLIS.numOutputSymbols) bits.

CODE = CONVENC(MSG, TRELLIS, PUNCPAT) is the same as the syntax above, except that it specifies a puncture pattern (PUNCPAT) to allow higher rate encoding. PUNCPAT must be a vector of 1's and 0's where the 0's indicate the punctured bits. PUNCPAT must have a length of at least log2(TRELLIS.numOutputSymbols) bits.

CODE = CONVENC(MSG,TRELLIS,...,INIT_STATE) is the same as the syntaxes above, except that the encoder registers start at a state specified by INIT_STATE. INIT_STATE is an integer between 0 and TRELLIS.numStates - 1 and must be the last input parameter. To use the default value for INIT_STATE, specify it as 0 or [].

[CODE FINAL_STATE] = CONVENC(...) returns the final state FINAL_STATE of the encoder after processing the input message.

Outros Esquemas de Codificação

- **Outros Esquemas de Codificação**

  - TCM
  - Códigos Turbo (Convolucional e de Bloco)
  - LDPC
  - Códigos Espaçotemporais
  - etc...