2.3. Códigos de blocos

- Distância de Hamming e distância mínima
- Matriz geradora e matriz de verificação de paridade
- Grafos de Tanner
- Síndrome e descodificação de máxima verosimilhança
- Matriz-padrão e "coset leaders"
- Limites de Singleton, de Hamming e de Plotkin
- Distribuição de pesos
- Probabilidade de erro não detectado
- Probabilidade de erro não corrigido
- Códigos aumentados e encurtados

Distância de Hamming e distância mínima

Seja um vector de código representado por $\mathbf{X} = (x_1 x_2 \ldots x_n)$.

- Um código é **linear** quando:
  - — inclui o vector nulo
  - — a soma de dois vectores do código é ainda um vector do código.

- **Soma de vectores**: $\mathbf{X} \oplus \mathbf{Y} = (x_1 \oplus y_1 \quad x_2 \oplus y_2 \quad \cdots \quad x_n \oplus y_n) \quad x_i, y_i = 0,1$

- **Peso** do vector X, $w(X)$: Número de elementos não nulos do vector.

  Ex.: Se $X = (101101) \Rightarrow w(X) = 4$

- **Distância de Hamming**, $d(X,Y)$, entre quaisquer dois vectores do código: número de elementos diferentes.

Exemplo: $\left.\begin{array}{l} X = (1\,0\,1\,1\,1\,0\,1) \\ Y = (1\,1\,0\,0\,1\,0\,1) \end{array}\right\} d(X,Y) = 3$

$$Z = X \oplus Y = (0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0)$$

Note-se que $w(Z) = w(X \oplus Y) = 3 = d(X,Y)$

A distância de Hamming entre dois vectores é igual ao peso do seu vector soma.

- **Distância mínima** de um código, $d_{\min}$: é a menor distância de Hamming entre dois vectores válidos do código.

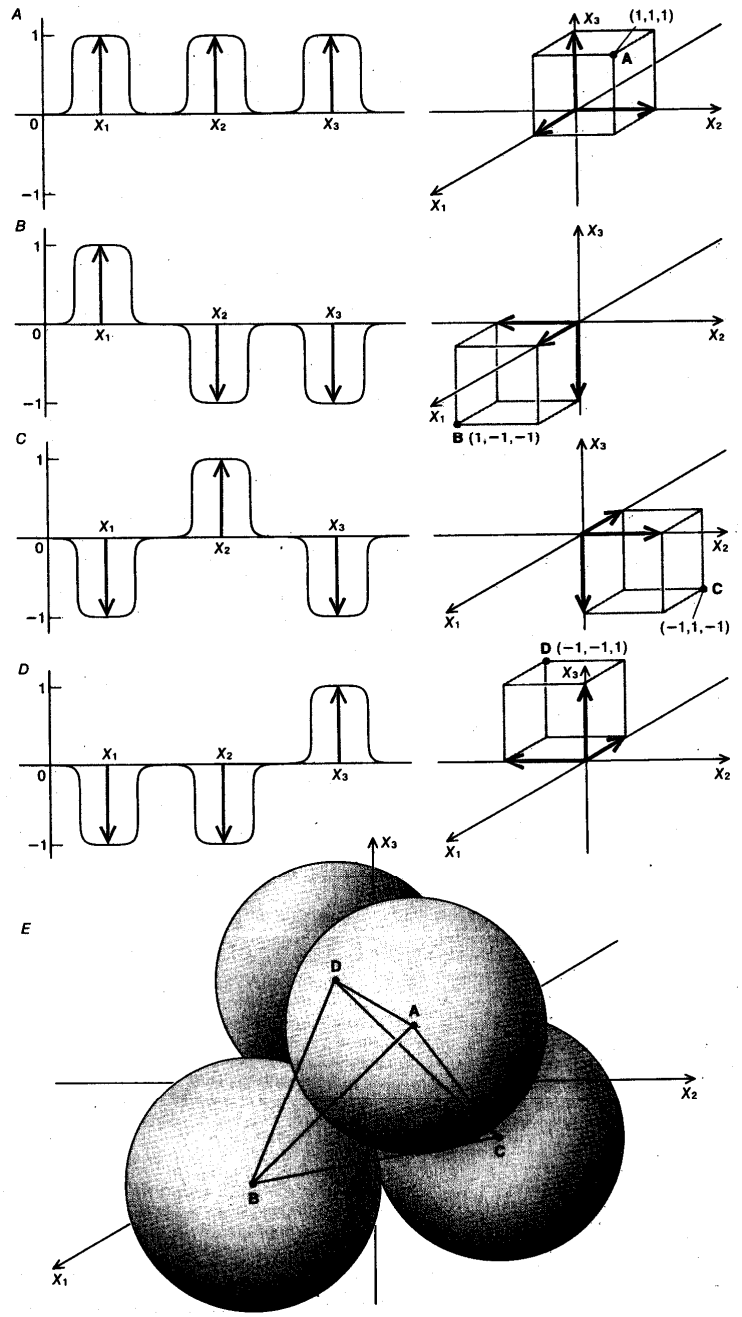
  A distância de Hamming entre X e Y é igual ao peso de um outro vector do código, $X \oplus Y$. Logo, a menor distância de Hamming é igual ao menor peso. Portanto,

$$d_{\min} = [w(X)]_{\min} \qquad X \neq (0\ 0\ \ldots\ 0)$$

De todos os vectores não nulos um apresenta o menor peso. Esse peso é a distância mínima do código.

Códigos, distâncias mínimas e empacotamento de esferas

(In "The Packing of Spheres", N. J. A. Sloane, *Scientific American*, Janeiro de 1984)



DESIGN OF A CODE for the efficient transmission of information is closely related to the sphere-packing problem. The code is to be a finite set of signals called code words that are easily distinguished from one another and do not waste electric power. If each code word is a sequence of, say, three discrete voltage levels, each sequence can be represented as a point in three-dimensional space: the first coordinate of the point is the numerical value of the first voltage level, the second coordinate is the value of the second voltage level and so on (*A–D*). The transmission power required for each voltage pulse is proportional to the square of the voltage, and so the total power needed to transmit one code word is the sum of the squares of the three discrete voltages associated with the code word. The sum is equal to the square of the distance from the origin to the point in three-dimensional space that represents the code word. Thus the problem of minimizing transmission power is equivalent to the problem of placing all the points that represent code words as close to the origin as possible. On the other hand, the need to distinguish code words from one another can be treated as the requirement that the points in space representing the code words not be closer together than some minimum distance *d*. Meeting the two requirements simultaneously is geometrically equivalent to packing hard, nonoverlapping spheres of radius $d/2$ around the origin as densely as possible (*E*).

Capacidade de controlo de erros

- A detecção de erros é sempre possível quando o número de erros de transmissão numa palavra de código é inferior à distância mínima $d_{\min} \Rightarrow$ a palavra errónea não é um vector válido.

- Inversamente, se o número de erros iguala ou excede $d_{\min}$ a palavra errónea pode corresponder a outro vector válido e os erros não podem ser detectados.

  - Se um código detecta até ℓ erros por palavra: $d_{\min} \geq l + 1$

  - Se um código corrige até t erros por palavra: $d_{\min} \geq 2t + 1$

  - Se um código corrige até t erros por palavra e detecta $\ell > t$ erros por palavra: $d_{\min} \geq t + l + 1$

- A distância mínima de um código de blocos (n, k) é limitada superiormente por

$$d_{\min} \leq n - k + 1 \qquad \textit{(Limite de Singleton)}$$

  Com códigos binários a igualdade só se atinge com códigos de repetição ($k = 1$). Infelizmente a sua taxa é muito pequena ($R_c = k/n = 1/n$).

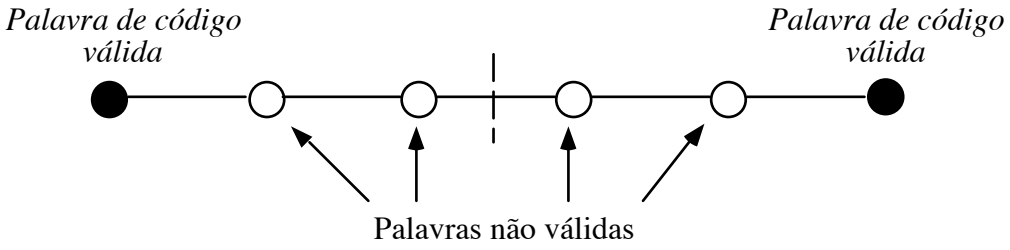
- Um código com $d_{\min} = n - k + 1$ chama-se *código separável pela distância máxima* ("maximum-distance-separable code", ou código MDS).

  Um exemplo é o código de Reed-Solomon, que encontraremos mais adiante.

Capacidade de detectar e corrigir erros

Exemplos:

1) $d_{min} = 5$



Este código consegue detectar até 4 erros por palavra. Consegue corrigir até 2 erros por palavra.

2) Código de repetição tripla:

$0 \rightarrow 000$

$1 \rightarrow 111$ (só tem estas duas palavras de código)

$d_{\min} = 3 \Rightarrow$ Pode detectar $l \leq 3 - 1 = 2$ erros/palavra de 3 bits

Pode corrigir $t \leq \frac{3-1}{2} = 1$ erro/palavra de 3 bits

3) $d_{min} = 7$

É possível:

— corrigir erros *triplos* ($t = 3$)

ou

— corrigir erros *duplos* ($t = 2$) e detectar erros *quádruplos* ($\ell = 4$).

Vale a pena codificar?

Seja: S — Potência do sinal

T_w — duração de uma palavra de k símbolos antes da codificação (= duração da palavra de n símbolos depois da codificação) $\Rightarrow$ Energia/palavra de código: $E_s = ST_w$

Energia recebida / símbolo — $E_b = \frac{ST_w}{k}$ (sem codificação)

Energia recebida / símbolo — $E_{bc} = \frac{ST_w}{n} = \frac{k}{n} E_b$ (com codificação)

- Como $n > k$, a energia por símbolo diminui com o uso de codificação $\Rightarrow$ a probabilidade de erro num símbolo é maior com codificação do que sem codificação.

- No entanto a *redundância* introduzida pelos $n - k$ símbolos de paridade permite corrigir erros, o que pode conduzir a uma melhoria global do desempenho do sistema.

- Uma medida da eficiência da codificação obtém-se comparando a probabilidade de erro numa palavra codificada, P_{enc}, com a probabilidade de erro numa palavra não codificada, P_e.

Seja $p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$ — probabilidade de erro num símbolo, sem codificação.

$p_c = Q\left(\sqrt{2\frac{k}{n}\frac{E_b}{N_0}}\right)$ — probabilidade de erro num símbolo, com codificação.

Sem codificação:

Probabilidade de erro numa palavra é 1 menos a probabilidade de todos os k símbolos da palavra serem recebidos correctamente:

$$P_e = 1 - P(0,k) = 1 - (1-p)^k$$

Com codificação:

Probabilidade de erro numa palavra é a probabilidade de haver $t + 1$ erros, $t + 2$ erros, $t + 3$ erros, etc.:

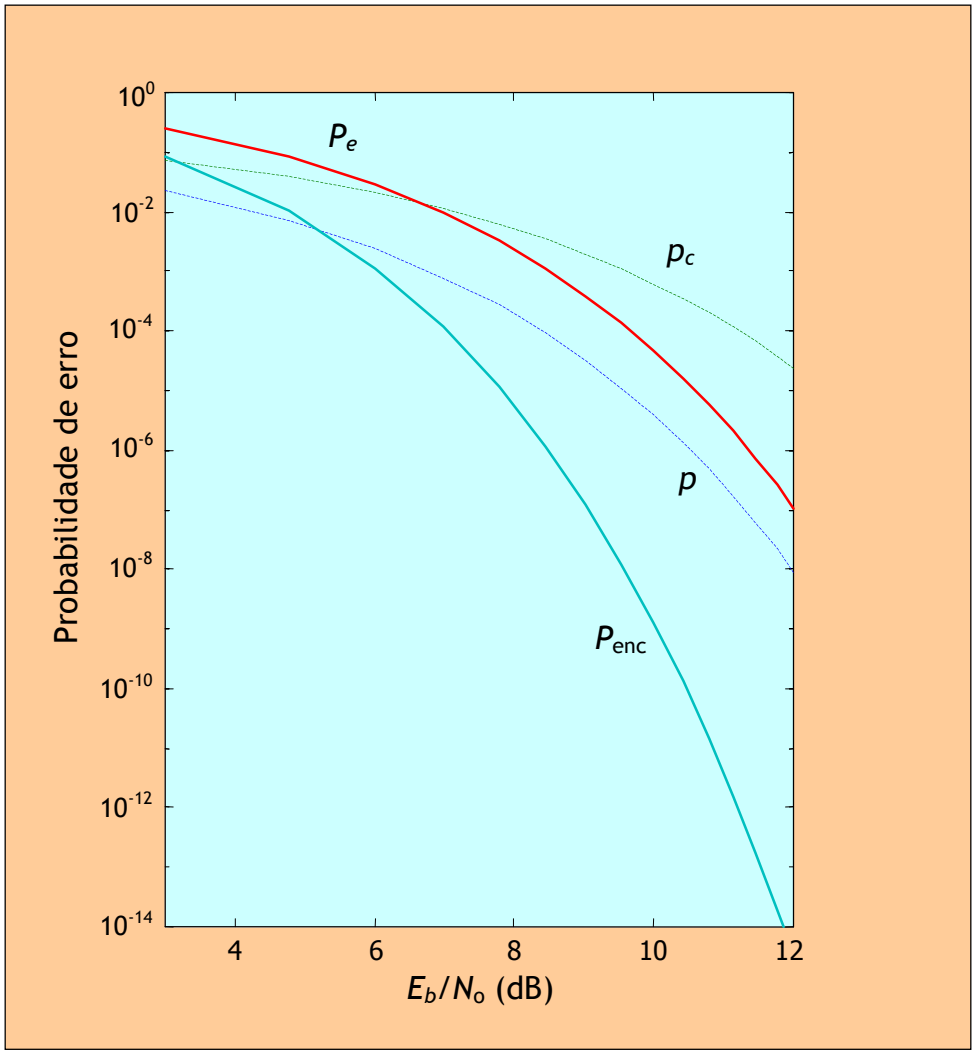
$$P_{enc} = \sum_{i=t+1}^{n} P(i,n) = \sum_{i=t+1}^{n} \binom{n}{i} p_c{}^i (1-p_c)^{n-i}$$

Exemplo com código de Golay (23,12) e transmissão BPSK

$$p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \qquad p_c = Q\left(\sqrt{2\frac{12}{23}\frac{E_b}{N_0}}\right)$$

$$P_e = 1-(1-p)^{12} \qquad t = 3 \qquad P_{enc} = \sum_{i=4}^{23}\binom{23}{i} p_c{}^i (1-p_c)^{23-i}$$

$\frac{N_0}{2}$ — densidade espectral de potência do ruído branco



Códigos de repetição

São códigos $(n, 1)$: a cada bit de informação acrescentam-se n-1 bits iguais.

Taxa	Distância mínima	Erros corrigidos por palavra	Palavra de código
1/3	3	1	0 0 0 1 1 1
1/5	5	2	0 0 0 0 0 1 1 1 1 1
.	.	.	.
.	.	.	.
1/n	n	$\frac{1}{2}(n-1)$	0 0 …0 1 1 …1

Exemplo: Canal binário simétrico e transmissão BPSK

- Probabilidade de erro sem e com codificação:

$$p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \qquad p_c = Q\left(\sqrt{\frac{2E_b}{nN_0}}\right) > p$$

- O código de repetição pode corrigir $t = \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor = \left\lfloor \frac{n-1}{2} \right\rfloor$ erros por palavra de código.

$$\Rightarrow P_{enc} = \sum_{i=t+1}^{n} \binom{n}{i} {p_c}^i (1 - p_c)^{n-i} \qquad (P_e = 1 - (1 - p) = p, \text{ porque } k = 1)$$

Por exemplo, num código de repetição tripla (n = 3) pode corrigir-se 1 erro. Os erros duplos e triplos ocorrem com uma probabilidade

$$P_{enc} = P(2,3) + P(3,3) = \sum_{i=2}^{3} \binom{3}{i} {p_c}^i (1 - p_c)^{3-i} = 3p_c^2 - 2p_c^3$$

Representação matricial dos códigos de blocos

Códigos sistemáticos

Vamos considerar apenas os *códigos sistemáticos*, nos quais:

- os primeiros k símbolos da palavra de código de n bits constituem a sequência de informação $X = (x_1 x_2 \ldots x_k)$, e os últimos n - k símbolos representam os *bits de verificação* ou *bits de paridade*.

$$\mathbf{Y} = (x_1 x_2 \ldots x_k c_1 c_2 \ldots c_{n-k}) = (\mathbf{X} \mid \mathbf{C})$$

$\mathbf{X}$ — Vector de mensagem

$\mathbf{C}$ — Vector de verificação

- Cada palavra de código Y é obtida multiplicando o vector X por uma matriz G, chamada matriz geradora, de dimensões $k \times n$:

$$\mathbf{Y} = \mathbf{XG}$$

Matriz geradora

A matriz geradora de um código sistemático tem a estrutura seguinte:

$$\mathbf{G} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}}_{k \times k} \underbrace{\begin{bmatrix} g_{1,k+1} & \cdots & g_{1,n} \\ g_{2,k+1} & \cdots & g_{2,n} \\ \vdots & & \\ \vdots & & \\ g_{k,k+1} & \cdots & g_{k,n} \end{bmatrix}}_{k \times (n-k)} = \left[\mathbf{I}_k \middle| \mathbf{P} \right]$$

$\mathbf{I}_k$ — Matriz identidade $k \times k$

$\mathbf{P}$ — Submatriz $k \times (n-k)$

Representação matricial dos códigos de blocos

Códigos sistemáticos

Determinação do vector C:

Como $\mathbf{Y} = \mathbf{XG} = (\mathbf{X} \mid \mathbf{C})$ e $\mathbf{G} = (\mathbf{I}_k \mid \mathbf{P})$ $\Rightarrow$ $\mathbf{C} = \mathbf{XP}$

A questão está em determinar a submatriz **P** para obtermos os valores pretendidos de $d_{\min}$ e R_c.

Matriz de verificação de paridade

Relacionada com **G** temos a *matriz de verificação de paridade*, **H**, de dimensões $(n-k)\times n$ e estrutura $\mathbf{H} = \left[\mathbf{P}^T \middle| \mathbf{I}_{n-k}\right]$, tal que $\mathbf{GH}^T = \mathbf{0}$. A sua transposta é

$$\mathbf{H}^T = \left[\frac{\mathbf{P}}{\mathbf{I}_{n-k}}\right] = \begin{bmatrix} g_{1,k+1} & \cdots & g_{1,n} \\ g_{2,k+1} & \cdots & g_{2,n} \\ \vdots & & \\ g_{k,k+1} & \cdots & g_{k,n} \\ 1 & \cdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \cdots & 1 \end{bmatrix}$$

O produto de qualquer palavra de código por $\mathbf{H}^T$ é um vector nulo:

$$\mathbf{YH}^T = \mathbf{X}\underbrace{\mathbf{GH}^T}_{\mathbf{0}} = \mathbf{0}$$

Um exemplo de código de blocos: o código de Hamming

Um *código de Hamming* é um código linear de blocos $(2^{n-k} - 1, k)$ com

- $n - k \geq 3$ bits de paridade
- $n = 2^{n-k} - 1$
- A taxa do código é $R_c = \frac{k}{n} = 1 - \frac{n-k}{2^{n-k}-1}$ $(\approx 1$ se $n - k >> 1)$
- Independentemente do número de bits de paridade a distância mínima é sempre $d_{\min} = 3$.

  Um código de Hamming é um código que permite *corrigir 1 erro* ou *detectar 2 erros*.
- As k linhas da submatriz **P** consistem em todas as palavras de $n - k$ bits com dois ou mais "uns" (por causa de **H**, como verá), em qualquer ordem.

Exemplo:

Seja $n - k = 3 \Rightarrow$ $n = 2^3 - 1 = 7,\ k = 4 \Rightarrow$ código (7, 4)

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}] = [\mathbf{I}_4 \mid \mathbf{P}] = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$ (isto é apenas um exemplo para **P**)

Dada uma mensagem $X = (x_1 x_2 x_3 x_4)$ os bits de verificação obtêm-se de **C** = **XP**, isto é, do sistema de 3 equações

$$\begin{cases} c_1 = x_1 \oplus x_2 \oplus x_3 \oplus 0 \\ c_2 = 0 \oplus x_2 \oplus x_3 \oplus x_4 \\ c_3 = x_1 \oplus x_2 \oplus 0 \oplus x_4 \end{cases}$$ (observe as colunas de **P**…)

Alguns outros códigos de Hamming $((2^{n-k} - 1, k))$**:**

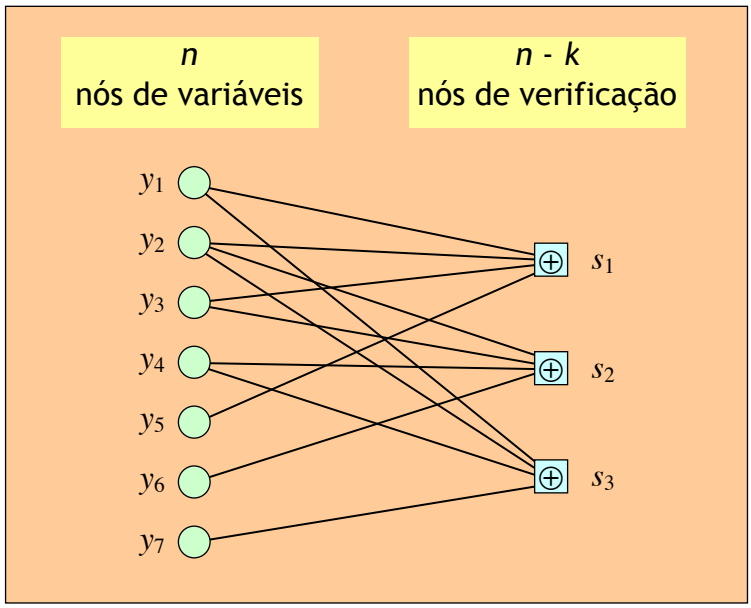
(15,11) (31,26) (63,57)

Grafos de Tanner ou grafos bipartidos

Retomemos as equações de paridade do código (7,4) anterior:

$$\begin{cases} c_1 = x_1 \oplus x_2 \oplus x_3 \\ c_2 = x_2 \oplus x_3 \oplus x_4 \\ c_3 = x_1 \oplus x_2 \oplus x_4 \end{cases} \Rightarrow \begin{cases} x_1 \oplus x_2 \oplus x_3 \oplus c_1 = 0 \\ x_2 \oplus x_3 \oplus x_4 \oplus c_2 = 0 \\ x_1 \oplus x_2 \oplus x_4 \oplus c_3 = 0 \end{cases}$$

A partir delas podemos desenhar o respectivo *grafo de Tanner*, ou *grafo bipartido*.



- A soma (mod 2) dos bits que concorrem num qualquer nó de verificação é nula se a palavra de *n* bits pertencer ao código.

  Exemplo: $y_1 \oplus y_2 \oplus y_3 \oplus y_5 = 0$

- $\mathbf{S} = [s_1 \; s_2 \; s_3]$ é a síndrome correspondente a uma palavra de sete bits recebida $\mathbf{Z} = [z_1 \quad z_2 \quad \ldots \quad z_7]$

  Exemplo: $\mathbf{Z} = [0111011] \quad \Rightarrow \quad \mathbf{S} = [001]$

Nem todos os códigos (2^{n-k}-1, k) são códigos de Hamming!!

Um código de Hamming é um código perfeito corrector de erros simples, com $t = 1$ e $d_{\min} = 3$. Ora… não basta que um código tenha dimensões $(2^{n-k} - 1, k)$ para que seja de Hamming.

Exemplo:

O código (7,4) gerado pela matriz seguinte não é um código de Hamming.

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

É que a submatriz **P** tem linhas iguais e algumas com peso 1. A distância mínima é 2.

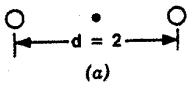
⇓

Este código não corrige sequer todos os padrões com 1 erro!

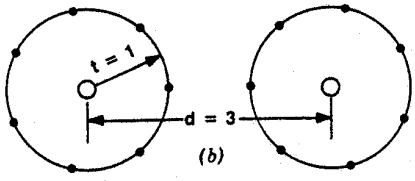
Nos códigos de Hamming todas as linhas de **P** são diferentes e têm sempre mais do que um "1".

Distância mínima e capacidade de correcção

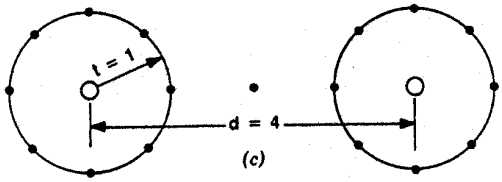
Exemplos com códigos binários



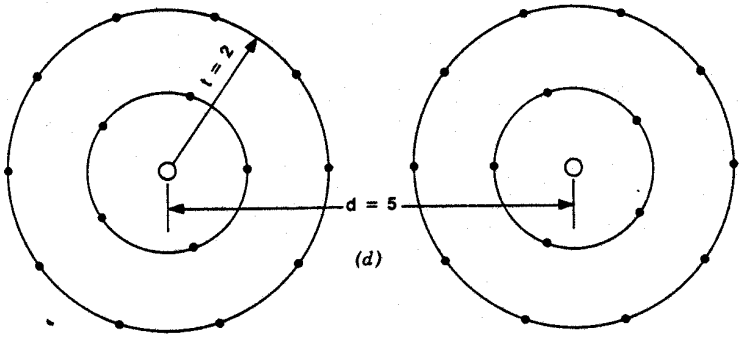
$(n, n-1)$, Paridade par



$(7,4)$, Hamming



$(8,4)$, Hamming aumentado



$(5,1)$, código de repetição

Codificação linear

P.: Como calcular as palavras de um código linear?

R.: Como a soma de duas palavras de um código linear é uma palavra do mesmo código, podemos obter palavras à custa de outras já determinadas.

P.: Quais?

R.: Havendo *k* bits de informação só precisamos de usar *k* palavras de código *linearmente independentes*, isto é, um conjunto de *k* palavras nenhuma das quais pode ser obtida por combinação linear de 2 ou mais palavras do conjunto.

As restantes $2^k - k$ palavras são obtidas das primeiras por adição módulo 2.

Isto quer dizer que não precisamos de calcular 2^k palavras recorrendo à matriz geradora: basta calcular *k*.

P.: Como determinar k palavras linearmente independentes?

R.: Uma maneira fácil é escolher aquelas que só têm um "1" nas primeiras *k* posições, isto é, as mensagens de *k* bits com peso 1.

As operações anteriores equivalem a somar linhas da matriz geradora (ver exemplo seguinte).

Exemplo de codificação

Consideremos o código (7, 4) com a matriz geradora seguinte:

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

- Existem $2^k = 2^4 = 16$ palavras de código correspondentes a outras tantas mensagens de $k = 4$ bits. Nestas 16 mensagens há 4 com peso unitário. As palavras de código correspondentes são:

Mensagens de 4 bits		Palavras de código com 7 bits
(1)	1000	1000 101
(2)	0100	0100 111
(3)	0010	0010 110
(4)	0001	0001 011

- As restantes 12 palavras podem ser obtidas a partir destas. Por exemplo, qual é a palavra de código correspondente à mensagem $X = [0101]$? Basta adicionar a 2ª e a 4ª palavras:

$$\begin{array}{rr} & 0001011 \\ \oplus & 0100111 \\ \hline & 0101100 \end{array}$$

- A operação anterior é equivalente à soma das linhas de ordem 2 e 4 da matriz geradora **G**:

$$\mathbf{Y} = \mathbf{XG} = [0\,1\,0\,1]\mathbf{G} = [0\,1\,0\,1\,1\,0\,0]$$

Como determinar a distância mínima a partir da matriz H?

A transposta da matriz **H** pode ser escrita linha-a-linha como

$$\mathbf{H}^T = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_i \\ \vdots \\ \mathbf{h}_n \end{bmatrix} \qquad (\boldsymbol{h_i} \text{ — vector-linha de ordem } i)$$

Ora já sabemos que se $\boldsymbol{Y}$ for um vector de código então $\mathbf{Y}\mathbf{H}^T = \mathbf{0}$. Suponhamos então que o vector $\boldsymbol{Y}$ tem elementos não-nulos nas posições i, j e k, por exemplo. Será fácil de verificar que as linhas $\boldsymbol{h_i}$, $\boldsymbol{h}_j$ e $\boldsymbol{h}_k$ somadas dão zero.

Exemplo: código (6,3) definido pela matriz $\boldsymbol{G} = \begin{bmatrix} 100110 \\ 010011 \\ 001101 \end{bmatrix}$.

O vector [110101] pertence ao código. Como os bits "1" estão nas 1ª, 2ª, 4ª e 6ª posições, a soma das linhas 1, 2, 4 e 6 de $\boldsymbol{H}^T$ deverá ser nula. De facto, sendo

$$\mathbf{H}^T = \begin{bmatrix} 110 & 011 & 101 & 100 & 010 & 001 \end{bmatrix}^T,$$

a soma das linhas indicadas dá $\boldsymbol{h_1} + \boldsymbol{h_2} + \boldsymbol{h_4} + \boldsymbol{h_6} = 0$:

$$\begin{array}{r} 110 \\ 011 \\ 100 \\ \oplus\, 001 \\ \hline 000 \end{array}$$

Como determinar a distância mínima a partir da matriz H^T?

As considerações anteriores levam-nos a concluir o seguinte:

- Se houver uma palavra de código de peso L então existirão L linhas de $\boldsymbol{H}^T$ que somadas dão zero.
- Como a distância mínima do código é igual ao peso mínimo de todas as palavras de código, então existem pelo menos $d_{\min}$ linhas de $\boldsymbol{H}^T$ que somadas dão zero.

Em conclusão:

> ***O menor número de linhas de H^T cuja soma é nula é igual à distância mínima do código.***

(dito de outra maneira: $d_{\min}$ é igual ao menor número de linhas de $\boldsymbol{H}^T$ linearmente dependentes)

Exemplo: mesmo código (6,3) definido anteriormente:

$$\mathbf{H}^T = \begin{bmatrix} 110 & 011 & 101 & 100 & 010 & 001 \end{bmatrix}^T$$

- não há nenhuma linha nula $\Rightarrow \quad d_{\min} > 1$
- não há linhas iguais $\Rightarrow \quad d_{\min} > 2$
- a soma das três linhas 1, 2 e 3 (por exemplo) é nula

$$\Rightarrow \quad d_{\min} = 3$$

Como fazer a descodificação?

Uma maneira seria comparar a palavra recebida com todas as possíveis 2^k palavras do código.

Mas ... imaginemos que queríamos usar um código de Hamming com $R_c \geq 0{,}8$:

$\Rightarrow$ $n - k \geq 5$ $\Rightarrow$ $n \geq 31$, $k \geq 26$

$\Rightarrow$ Seria preciso guardar $n \times 2^k > 10^9$ bits para comparação!!

Há outros métodos mais práticos, associados à matriz de verificação de paridade **H**. Como $\mathbf{ZH}^T = (00...0)$ se **Z** pertencer ao conjunto das palavras de código, se **Z** não pertencer então

$\mathbf{ZH}^T \neq (00...0)$ ($\Rightarrow$ pelo menos um elemento será não nulo)

A descodificação faz-se multiplicando a sequência recebida $\mathbf{Z} = (z_1 z_2 ... z_n)$ pela transposta da matriz de verificação de paridade **H**, o que dá um vector de *n* - *k* bits:

$\mathbf{S} = \mathbf{ZH}^T$ **S** é a *síndrome* ou *síndroma*

- Uma síndrome não nula indica a presença de erros.
- Uma síndrome nula significa que:
  - ou não houve erros introduzidos na transmissão
  - ou houve erros na transmissão que transformaram a palavra de código enviada numa outra palavra de código válida.
- Se o código tiver uma distância mínima $d_{\min}$ são precisos pelo menos $d_{\min}$ erros para transformar uma palavra de código noutra palavra de código igualmente válida.

Descodificação com síndromes

- A sequência recebida **Z** é a soma em módulo 2 da palavra de código **Y** com um eventual vector binário de erro, **E**:

$$\mathbf{S} = \mathbf{Z}\mathbf{H}^T = (\mathbf{Y} \oplus \mathbf{E})\mathbf{H}^T = \mathbf{Y}\mathbf{H}^T \oplus \mathbf{E}\mathbf{H}^T = \mathbf{E}\mathbf{H}^T$$

- Sendo $\mathbf{Z} = \mathbf{Y} \oplus \mathbf{E}$ então $\mathbf{Y} = \mathbf{Z} \oplus \mathbf{E}$

Exemplo: Se $\mathbf{Y} = [10011]$ e $\mathbf{Z} = [10110]$ $\Rightarrow$ $\mathbf{E} = [00101]$

$\Rightarrow$ $\mathbf{Z} \oplus \mathbf{E} = [10110] \oplus [00101] = [10011] = \mathbf{Y}$

- O vector de erro, **E**, tem n bits $\Rightarrow$ existem 2^n padrões de erro possíveis.
- A síndrome, **S**, tem n-k bits $\Rightarrow$ existem $2^{n-k} < 2^n$ síndromes possíveis.
  $\Rightarrow$ a síndrome não determina univocamente **E**.
- Dos 2^n padrões possíveis de erro apenas podem ser corrigidos $2^{n-k} - 1$ padrões (exclui-se o padrão nulo).
- É conveniente que os $2^{n-k} - 1$ padrões que podem ser corrigidos sejam os padrões de erro *mais prováveis*, isto é, aqueles que apresentem menos erros (por outras palavras, aqueles que tenham um peso mais baixo).

Chama-se a esta técnica a **descodificação de máxima verosimilhança**:

1. A partir dos $2^{n-k} - 1$ padrões de erro mais prováveis calcula-se uma tabela de $2^{n-k} - 1$ síndromes $\mathbf{S} = \mathbf{E}\mathbf{H}^T$ possíveis.

2. Tendo recebido uma palavra **Z** de n bits calcula-se a síndrome respectiva, $\mathbf{S} = \mathbf{Z}\mathbf{H}^T$.

3. Consulta-se a tabela de síndromes para determinar o padrão de erro mais provável, $\hat{\mathbf{E}}$, que corresponde à síndrome calculada.

4. A palavra transmitida mais provável, $\hat{\mathbf{Y}}$, obtém-se adicionando a palavra recebida, **Z**, ao padrão de erro estimado.

Descodificação com síndromes

Se $\mathbf{E} = [e_1 e_2 .. e_j .. e_n]$ e $\mathbf{H}^T = [h_1 h_2 .. h_j .. h_n]^T$ (h_j — vector linha de n - k símbolos)

$$\Rightarrow \mathbf{S} = \mathbf{E}\mathbf{H}^T = [e_1 e_2 .. e_j .. e_n] \begin{bmatrix} h_1 \\ \vdots \\ h_j \\ \vdots \\ h_n \end{bmatrix}$$

Imaginemos que ocorreu um único erro, isto é, **E** é nulo excepto na posição j:

$$\mathbf{E} = [00 \ldots 1 \ldots 0] \quad \Rightarrow \quad \mathbf{S} = [00 \ldots 1 \ldots 0]\, \mathbf{H}^T = h_j$$

Isto significa que, num código corrector de erros simples como o de Hamming:

> **Se a síndrome for igual à linha de ordem *j* da matriz $\mathbf{H}^T$ então houve um erro no *j*-ésimo bit da palavra de código recebida.**

- Para permitir a detecção de erros, as linhas da transposta da matriz de verificação de paridade devem ser distintas (para que não haja ambiguidade) e diferentes de zero* .
  - Número de linhas de $\mathbf{H}^T$: n
  - Número de linhas com n - k bits, distintas e diferentes de zero: $2^{n-k} - 1$

    $\Rightarrow \quad n = 2^{n-k} - 1$ (código de Hamming, precisamente)

* Está a ver porque é que no exemplo de código de Hamming apresentado anteriormente se disse que as linhas de **P** deviam ter dois ou mais uns? É para que todas as linhas de $\mathbf{H}^T$ sejam diferentes.

Exemplo de descodificação de máxima verosimilhança

Consideremos o código de Hamming (7, 4) anterior e seja **G** a sua matriz geradora:

$$\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}] = [\mathbf{I}_4 \mid \mathbf{P}] = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

A transposta da matriz de verificação de paridade vale

$$\mathbf{H}^T = \left[\frac{\mathbf{P}}{\mathbf{I}_{n-k}}\right] = \left[\frac{\mathbf{P}}{\mathbf{I}_3}\right] = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

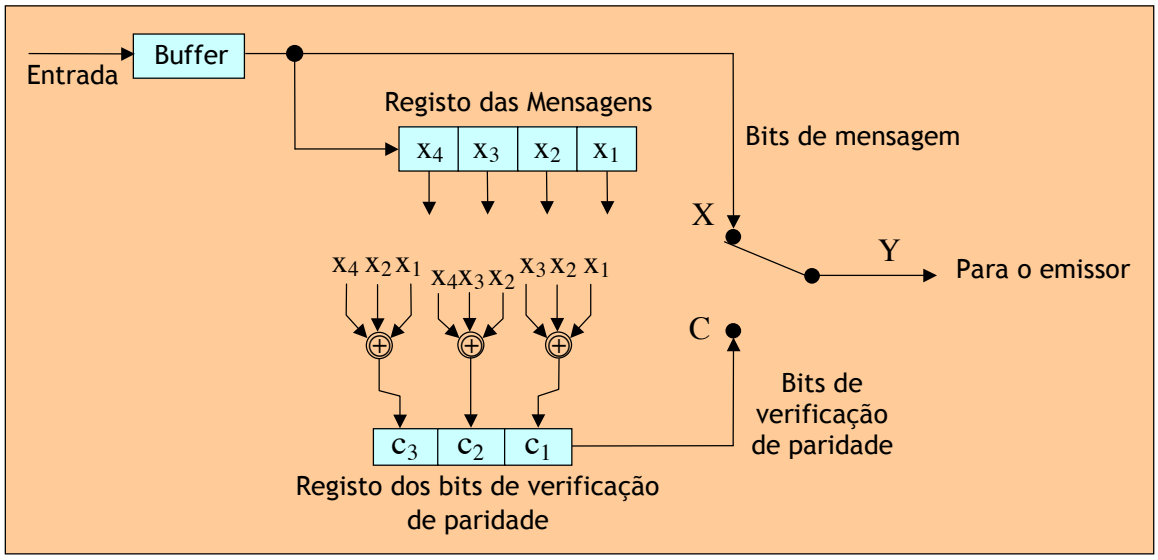
As síndromes são as linhas de $\mathbf{H}^T$ e há $2^{n-k} - 1 = 7$ padrões de erro corrigíveis, que são os 7 vectores de erro com menor peso:

S			Ê						
0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0
0	1	1	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	0	0	0	1

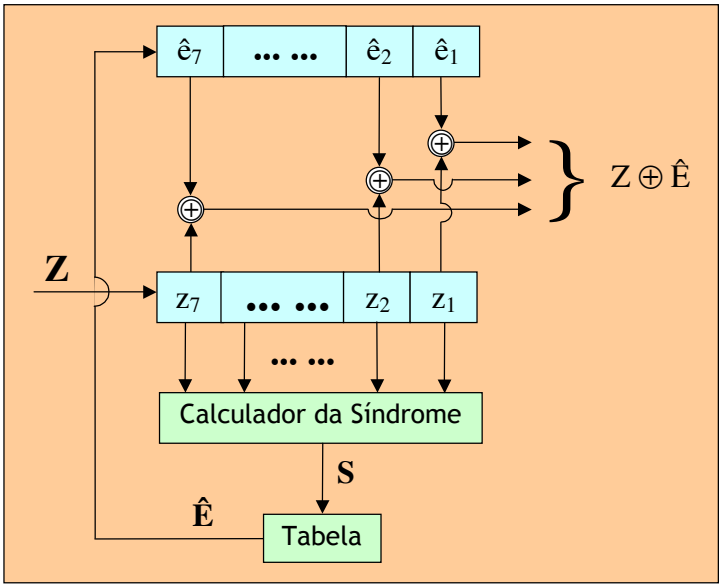
Este código (7,4) permite **detectar erros duplos** (pois $d_{\min} = 3$) mas apenas **corrigir erros simples**.

Exemplo de descodificação de máxima verosimilhança

Codificador do exemplo:



Descodificador de máxima verosimilhança respectivo:



Exemplo de descodificação de máxima verosimilhança

1. *Qual é a palavra de código* $\mathbf{Y}$ *correspondente à mensagem* $\mathbf{X} = [1011]$*?*

R.: $$\mathbf{Y} = \mathbf{XG} = \begin{bmatrix}1011\end{bmatrix} \times \begin{bmatrix}1 & 0 & 0 & 0 & 1 & 0 & 1\\ 0 & 1 & 0 & 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 1 & 0\\ 0 & 0 & 0 & 1 & 0 & 1 & 1\end{bmatrix} = \begin{bmatrix}1011000\end{bmatrix}$$

2. *Qual é a mensagem estimada,* $\hat{X}$ *, se recebermos* $\mathbf{Z} = [1011100]$*?*

R.: A síndrome vale:

$$\mathbf{S} = \mathbf{ZH}^T = \begin{bmatrix}1\oplus 1\oplus 1 & 1\oplus 1 & 1\oplus 1\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\end{bmatrix} \qquad (5^{\text{a}} \text{ linha de } \mathbf{H}^T)$$

Da tabela: $\hat{\mathbf{E}} = [0000\ 100]$.

Portanto, $\hat{\mathbf{Y}} = \mathbf{Z} \oplus \hat{\mathbf{E}} = \begin{bmatrix}1011100\end{bmatrix} \oplus \begin{bmatrix}0000100\end{bmatrix} = \begin{bmatrix}1011000\end{bmatrix} \quad \Rightarrow \quad \hat{\mathbf{X}} = \begin{bmatrix}1011\end{bmatrix}$

3. *É possível corrigir erros duplos com este código?*

R.: Como $d_{\min} = 3$ para este código, não é possível corrigir erros duplos. Se, por exemplo, $\mathbf{Z}$ é tal que $\mathbf{E} = [1000010]$ então

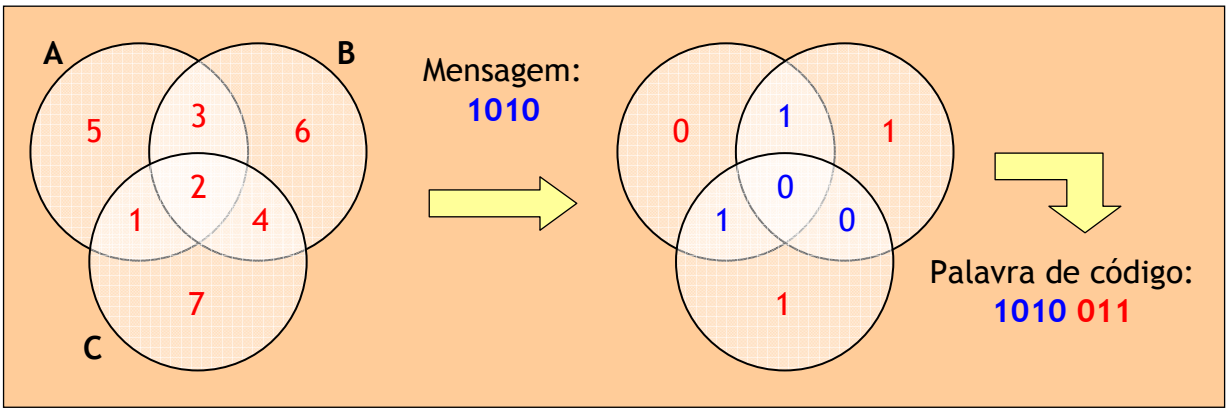
$$\mathbf{S} = \mathbf{ZH}^T = \mathbf{EH}^T = [111] \ \Rightarrow \qquad \hat{\mathbf{E}} = [0100000]$$

O resultado, $\mathbf{Z} \oplus \hat{\mathbf{E}}$, seria uma palavra com 3 erros, na 1ª, 2ª e 6ª posições (dois erros de transmissão + uma correcção errada). Note-se, no entanto, que a síndrome é igual à soma das linhas 1 e 6 de $\mathbf{H}^T$ (reparou certamente que os erros tinham ocorrido nas posições 1 e 6).

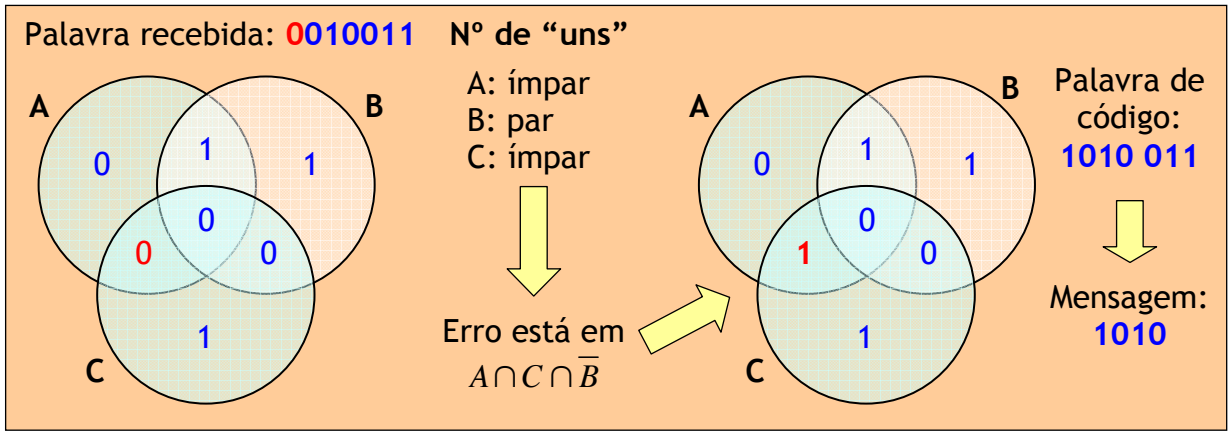
Códigos de Hamming (7,4) e diagramas de Venn

Um código de Hamming (7,4) pode ser caracterizado através de um diagrama de Venn, o que nos permite codificá-lo e descodificá-lo muito facilmente. Como fazer?

- Desenham-se três circunferências que se intersectem, colocando duas em cima e uma por baixo. Assim serão criadas 7 áreas, numeradas de 1 a 7 na figura abaixo*.
- Colocam-se os quatro bits da mensagem nas áreas de intersecção 1, 2, 3 e 4.
- Os três bits de paridade são colocados nas áreas 5, 6 e 7 de modo que em cada um dos círculos exista um número par de "uns" (ou seja, a soma mod 2 dos bits dentro de cada círculo é 0).



Descodificação



** A numeração de 1 a 7 deve respeitar as equações de paridade (ver página seguinte).*

Códigos de Hamming (7,4) e diagramas de Venn

- A numeração dos círculos A, B e C no diagrama de Venn anterior depende do código (7,4) usado, ou seja, deve obedecer às equações de paridade, ou vice-versa, que neste exemplo são

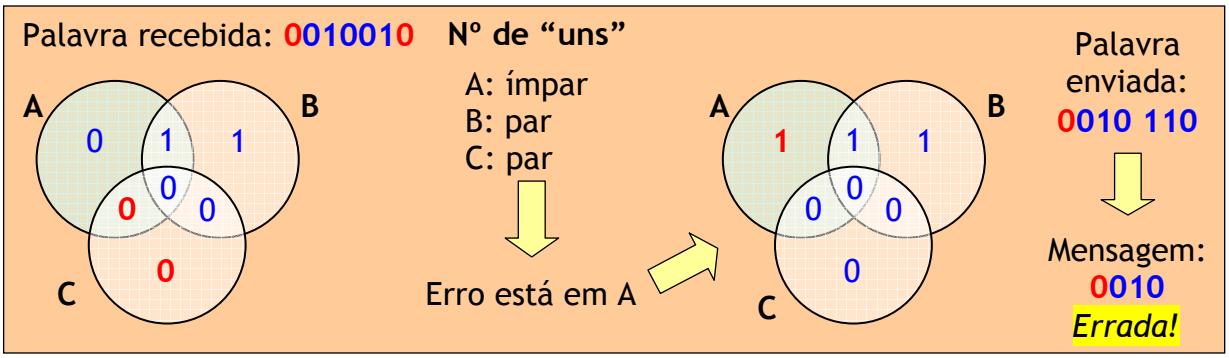
$$\begin{cases} c_1 = x_1 \oplus x_2 \oplus x_3 \\ c_2 = x_2 \oplus x_3 \oplus x_4 \\ c_3 = x_1 \oplus x_2 \oplus x_4 \end{cases}$$

Trata-se do código de Hamming (7,4) do anterior exemplo, em que

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

Repare-se que nas equações de paridade o bit x_2 é comum às três equações pelo que deve ser atribuído à área intersectada pelos três círculos.

- Pode confirmar-se que uma palavra com 2 erros não será correctamente descodificada, pois o código de Hamming é um código corrector de erros simples:



Matriz-padrão de um código de blocos

Sejam $Y_1, Y_2, \ldots Y_{2^k}$ as 2^k palavras de código de *n* bits de um código de blocos (n, k).

Existem 2^n vectores recebidos possíveis. Vamos colocá-los numa matriz de 2^n elementos, chamada *matriz-padrão*, de acordo com as seguintes regras:

1. Na 1ª linha colocamos todos os 2^k vectores de código, começando pelo vector nulo $\mathbf{Y}_1 = [00\ldots0]$.

2. Dos restantes $2^n - 2^k$ énuplos colocamos um énuplo E_2 por baixo de Y_1. A 2ª linha é formada somando E_2 a cada vector Y_i e colocando a soma $E_2 + Y_i$ por baixo de Y_i.

3. Nas restantes linhas procede-se de maneira idêntica.

$$\begin{bmatrix} Y_1 = 0 & Y_2 & Y_3 & \cdots & Y_{2^k} \\ E_2 & Y_2 + E_2 & Y_3 + E_2 & \cdots & Y_{2^k} + E_2 \\ E_3 & Y_2 + E_3 & Y_3 + E_3 & \cdots & Y_{2^k} + E_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ E_{2^{n-k}} & Y_2 + E_{2^{n-k}} & Y_3 + E_{2^{n-k}} & \cdots & Y_{2^k} + E_{2^{n-k}} \end{bmatrix}$$

- Existem 2^k colunas e $\frac{2^n}{2^k} = 2^{n-k}$ linhas.

- Cada linha da matriz-padrão do código chama-se *coset***.**

- Ao primeiro elemento de cada linha (*coset*) chama-se *coset leader*.

- Qualquer elemento de um *coset* pode ser usado como o seu *coset leader*: os elementos do *coset* não são alterados, apenas permutados (verifique para confirmar).

A matriz-padrão e a descodificação

Seja D_j a coluna de ordem j da matriz-padrão:

$$D_j = \{Y_j, Y_j + E_2, Y_j + E_3, \ldots, Y_j + E_{2^{n-k}}\}$$

$\downarrow$ vector de código; $\downarrow$ $E_2, E_3, \ldots$ — *coset leaders*

As colunas $D_1, D_2, \ldots, D_{2^k}$ podem ser usadas para descodificar o código.

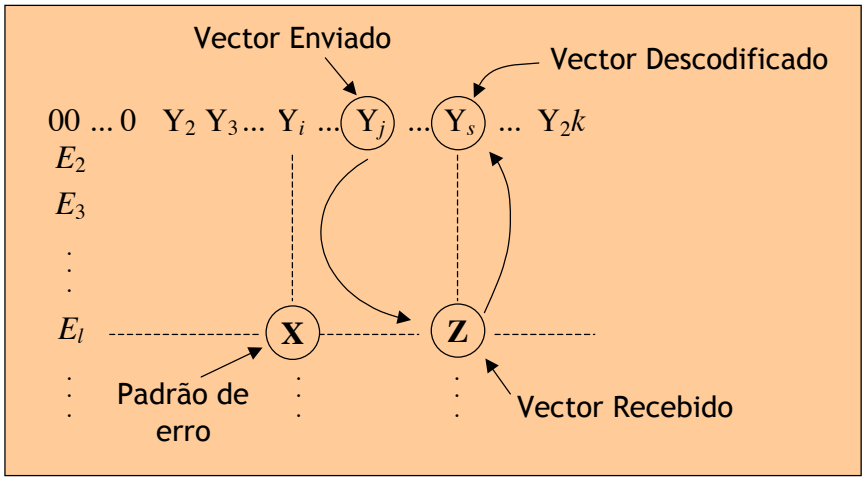
Suponhamos que o vector de código $\mathbf{Y}_j$ é transmitido através de um canal ruidoso. Da expressão de D_j vemos que a palavra recebida, **Z**, está em D_j se o *padrão de erro* causado pelo canal for um *coset leader*. Nesta situação o vector recebido **Z** será correctamente descodificado no vector transmitido $\mathbf{Y}_j$.

E se o padrão de erro não for um *coset leader*? **A descodificação não será correcta.** Vejamos porquê:

Se não é um *coset leader* o padrão de erros **X** está localizado num *coset* mas por baixo de um vector de código não nulo (por exemplo, no *coset* de ordem l e por baixo do vector $Y_i \neq 0$):

$X = Y_i + E_l \rightarrow$ o vector recebido é $Z = Y_j + X = Y_j + (Y_i + E_l) = (Y_j + Y_i) + E_l = Y_s + E_l$

$\Rightarrow$ o vector recebido, **Z**, está na coluna D_s e será descodificado em $\mathbf{Y}_s$, que não é a palavra transmitida.



Matriz-padrão, capacidade de correcção e síndrome

Concluímos que a descodificação estará correcta *se e só se* o padrão de erro causado pelo canal for um *coset leader*.

Como há 2^{n-k} *coset leaders* concluímos ainda que

Capacidade de correcção

Qualquer código de blocos linear (n, k) é capaz de corrigir 2^{n-k} padrões de erro.

Para minimizar a probabilidade de um erro de descodificação escolhem-se como *coset leaders* os padrões de erro mais prováveis (máxima verosimilhança).

Síndrome

Consideremos o *coset* de ordem $j \Rightarrow Y_i + E_j$ é um énuplo desse *coset*. A sua síndrome vale:

$$\mathbf{S} = (\mathbf{Y}_i + \mathbf{E}_j)\mathbf{H}^T = \underbrace{\mathbf{Y}_i\mathbf{H}^T}_{0} + \mathbf{E}_j\mathbf{H}^T = \mathbf{E}_j\mathbf{H}^T$$

Por aqui se vê que

todos os membros da mesma linha têm a mesma síndrome:

$\Rightarrow$ podemos estimar o *coset leader* $\Rightarrow$ podemos estimar o padrão de erro.

Matriz-padrão: um exemplo

Consideremos um código (6, 3) com as seguintes palavras de código:

(000000) (110100) (011010) (101110)

(101001) (011101) (110011) (000111) (Atenção: **Y** = (**C**|**X**)!)

A sua matriz-padrão pode ser escrita como:

000000	110100	011010	101110	101001	011101	110011	000111
000001	110101	011011	101111	101000	011100	110010	000110
000010	110110	011000	101100	101011	011111	110001	000101
000100	110000	011110	101010	101101	011001	110111	000011
001000	111100	010010	100110	100001	010101	111011	001111
010000	100100	001010	111110	111001	001101	100011	010111
100000	010100	111010	001110	001001	111101	010011	100111
010001	100101	001011	111111	111000	001100	100010	010110

↑
Padrões de erro corrigíveis

As palavras de código com menor peso são

(110100), (011010), (101001) e (000111) $\Rightarrow$ $d_{\min} = 3 \Rightarrow t = 1$

Se corrigisse todos os padrões com t ou menos erros e não corrigisse nenhum padrão com mais de t erros diríamos que o código era **perfeito**. Como afinal também corrige um padrão com $t + 1 = 2$ erros (neste caso o padrão 010001, escolhido algo arbitrariamente), diremos que este código não é perfeito.

Capacidade de detecção

Embora um código de blocos garanta que *detecta* todos os padrões de erro com $d_{\min} - 1$ ou menos erros, também consegue *detectar* uma grande fracção de padrões de erro com $d_{\min}$ ou mais erros. Na verdade,

Capacidade de detecção

Com um código (n,k) é possível detectar $2^n - 2^k$ *padrões de erro de comprimento n.*

Porquê?

- Existem $2^n - 1$ padrões de erro não nulos.
- Desses $2^n - 1$ padrões existem $2^k - 1$ padrões de erro que são iguais às $2^k - 1$ palavras de código não nulas (pois elas próprias constituem padrões de erro possíveis).

  Se um destes $2^k - 1$ padrões de erro ocorre transforma a palavra emitida $\mathbf{Y}_i$ numa outra palavra válida $\mathbf{Y}_j$.

  $\Rightarrow$ síndrome nula $\Rightarrow$ descodificação errada

  Logo, **há** $2^k - 1$ **padrões de erro** ***não detectáveis***

- Se o padrão de erro não for idêntico a uma dessas $2^k - 1$ palavras de código o erro é detectável.

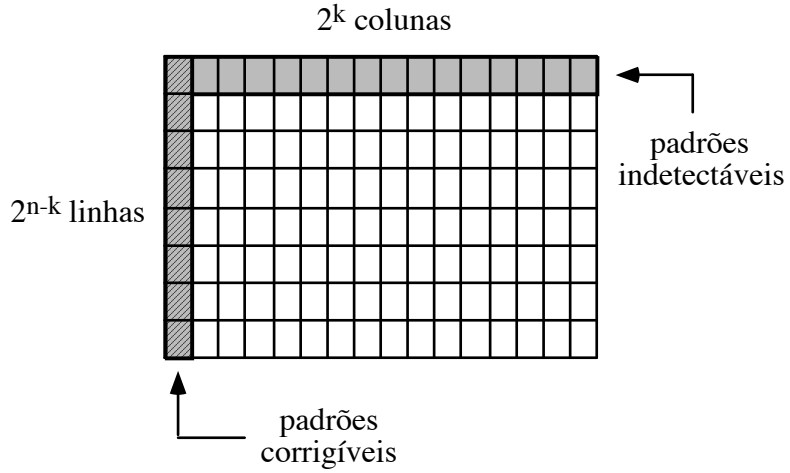
$\Rightarrow$ **Existem** $2^n - 1 - (2^k - 1) = 2^n - 2^k$ **padrões de erro detectáveis.** $c.q.d.$

Para *n* elevado $2^k - 1$ é em geral << 2^n $\Rightarrow$ Apenas uma pequena fracção dos padrões de erro não é detectada.

Matriz-padrão e padrões de erro

- Número de padrões de erro detectáveis: $2^n - 2^k$.

- Número de padrões de erro corrigíveis: 2^{n-k}.



$\Downarrow$

A probabilidade de erro não detectado, P_{end}, é menor que a probabilidade de erro não corrigido, P_{enc}.

Probabilidade de erro não detectado

Seja A_i o número de vectores de código com peso i. Aos números $A_0, A_1, \ldots, A_n$ chama-se *distribuição de pesos* do código. Os valores de A_1 a $A_{\text{dmin-1}}$ são nulos.

Se o código for usado apenas para *detecção* num canal binário simétrico a probabilidade do descodificador não detectar a presença de erros pode ser calculada a partir da distribuição de pesos:

- Um erro não detectado ocorre apenas quando o padrão de erro é idêntico a um vector de código não nulo.

  $\Rightarrow$ Existem A_i padrões de i erros nessas condições.

  $A_1 p(1-p)^{n-1}$ — probabilidade associada aos A_1 padrões com 1 erro.

  $A_2 p^2 (1-p)^{n-2}$ — probabilidade associada aos A_2 padrões com 2 erros.

  …

  $A_i p^i (1-p)^{n-i}$ — probabilidade associada aos A_i padrões com i erros.

- A probabilidade de erro não detectado é dada então por

$$P_{end} = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} \qquad (\text{com} \sum_{i=0}^{n} A_i = 2^k)$$

Exemplo: Código (6,3) usado apenas para detecção. Canal BSC e $p = 10^{-2}$.

Consultando a lista de palavras de código apresentada anteriormente:

$A_o = 1, A_1 = A_2 = 0, A_3 = 4, A_4 = 3, A_5 = A_6 = 0$

$$\Rightarrow P_{end} = \sum_{i=1}^{6} A_i p^i (1-p)^{6-i} = 4p^3(1-p)^3 + 3p^4(1-p)^2 = 3{,}9.10^{-6}$$

Em 1 milhão de palavras transmitidas há em média ≈ 4 palavras erradas não detectadas.

Probabilidade de erro não corrigido

A probabilidade de erro não corrigido é majorada por

$$P_{enc} \le \sum_{i=t+1}^{n} P(i,n) = \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}$$

p — probabilidade de transição do canal

(O máximo atinge-se se o código não corrigir mais que t erros, isto é, se for um *código perfeito*)

Um erro de descodificação ocorre **se e só se** o padrão de erro não for um *coset leader*. Se α_i for o número de *coset leaders* de peso i, então a probabilidade de erro num canal binário simétrico com probabilidade de transição p é

$$P_{enc} = 1 - \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}$$

Em resumo:

Código (n, k):		
	Detecta $2^n - 2^k$ padrões de erro	$\Rightarrow$ *Se* n *grande* $\Rightarrow P_{enc} >> P_{end}$
	Só corrige 2^{n-k} padrões de erro	

Exemplo: Código (6, 3) já apresentado, canal BSC e $p = 10^{-2}$

Na matriz-padrão vê-se que

$\alpha_0 = 1,$
$\alpha_1 = 6,$
$\alpha_2 = 1,$
$\alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = 0,$

logo, a probabilidade de descodificação errada vale

$$P_{enc} = 1 - (1-p)^6 - 6p(1-p)^5 - p^2(1-p)^4 \approx 1{,}37.10^{-3}$$

(compare-se com $P_{end} = 3{,}9.10^{-6}$)

Comprimento de código e distância mínima

A partir de um dado valor de k queremos construir um código com uma dada distância mínima d_{min}. Qual é o menor valor de n necessário? A tabela seguinte dá a resposta.

Menor comprimento n de qualquer código (n, k) conhecido para valores seleccionados de k e d_{min}

k \ d_{min}	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	5	6	8	9	11	12	14	15	17	18	20	21	23	24	26	27	29	30	32	33	35	36	38
3	6	7	—	—	13	14	—	—	20	21	24	25	27	28	31	32	34	35	38	39	41	42	45
4	7	8	—	—	14	15	—	—	—	—	—	—	29	30	—	—	—	—	—	—	44	45	48
5	—	-	12	13	15	16	—	—	—	—	—	—	30	31									
6	—	—	13	14	17	18	22	23	25	26	29	30	31	32									
7	—	—	14	15	18	19	23	24	26	27	31	32	35	36	40	41	43	44	46	47	49	50	55
8	12	13	15	16	19	20	25	26	28	29	33	34	37	38	41	42	44	45	47	48	50	51	57
9	13	14	17	18	20	21	26	27	29	30	34	35	38	39	44	45	47	48	52	53	54	55	58
10	14	15	18	19	21	22	28	29	30	31	36	37	39	40	46	47	48	49	53	54	55	56	59
11	15	16	19	20	22	23	29	30	31	32	37	38	41	42	48	49	50	51	—	—	57	58	60
12	—	—	21	22	23	24	30	31	34	35	40	41	43	44	49	50	51	52	—	—	58	59	61
13	—	—	22	23	26	27	—	—	35	36	41	42	44	45	52	53	54	55	—	—	59	60	65
14	—	—	23	24	27	28	—	—	36	37	42	43	46	47	53	54	55	56	58	59	60	61	66
15	—	—	25	26	29	30	—	—	37	38	43	44	47	48	—	—	57	58	59	60	61	62	69
16	—	—	26	27	30	31	—	—	38	39	47	48	49	50	56	57	58	59	60	61	62	63	70
17	—	—	27	28	32	33	37	38	39	40	48	49	50	51	58	59	59	60	61	62	63	64	71
18	—	—	28	29	33	34	38	39	40	41	49	50	52	53	59	60	61	62	63	64	66	67	75
19	24	25	29	30	34	35	39	40	41	42	50	51	53	54	60	61	63	64	69	70	74	75	80
20	25	26	30	31	35	36	40	41	42	43	51	52	54	55	61	62	65	66	72	73	77	78	81
21	26	27	31	32	37	38	41	42	43	44	52	53	55	56	62	63	66	67	74	75	78	79	84
22	27	28	32	33	38	39	43	44	44	45	53	54	57	58	64	65	67	68	75	76	79	80	85
23	28	29	34	35	39	40	44	45	45	46	54	55	58	59	65	66	68	69	76	77	80	81	86
24	29	30	35	36	40	41	45	46	46	47	55	56	59	60	66	67	69	70	77	78	81	82	93
25	30	31	36	37	41	42	47	48	47	48	58	59	60	61	68	69	70	71	80	81	82	83	96
26	31	32	37	38	42	43	—	—	53	54	59	60	61	62	69	70	71	72	81	82	83	84	97
27	—	—	38	39	43	44	50	51	54	55	60	61	62	63	70	71	72	73	82	83	84	85	98
28	—	—	39	40	44	45	51	52	55	56	61	62	63	64	71	72	78	79	83	84	85	86	99
29	—	—	40	41	45	46	52	53	56	57	62	63	65	66	73	74	79	80	84	85	86	87	101
30	—	—	41	42	46	47	54	55	57	58	63	64	66	67	74	75	80	81	85	86	87	88	104
31	—	—	42	43	47	48	55	56	58	59	65	66	67	68	75	76	81	82	86	87	93	94	105
32	—	—	43	44	48	49	56	57	59	60	66	67	68	69	76	77	82	83	92	93	94	95	106
33	—	—	44	45	49	50	57	58	60	61	67	68	69	70	77	78	83	84	93	94	95	96	107
34	—	—	45	46	50	51	58	59	61	62	68	69	70	71	78	79	85	86	94	95	—	—	108
35	—	—	46	47	51	52	59	60	62	63	69	70	71	72	79	80	86	87	95	96	—	—	109
36	—	—	47	48	52	53	60	61	63	64	70	71	72	73	80	81	87	88	—	—	102	103	111
37	—	—	48	49	53	54	61	62	65	66	72	73	76	77	81	82	88	89	—	—	103	104	112
38	—	—	49	50	54	55	62	63	66	67	73	74	77	78	82	83	89	90	—	—	104	105	113
39	—	—	50	51	55	56	63	64	67	68	76	77	78	79	83	84	90	91	—	—	106	107	114
40	—	—	51	52	56	57	64	65	70	71	77	78	79	80	84	85	91	92	102	103	107	108	115

(In John B. Anderson, "Digital Transmission Engineering", IEEE Press, 1999)

Limites de Singleton e de Hamming

Limite de Singleton

A distância mínima de um código de blocos (n, k) é limitada superiormente por

$$d_{\min} \leq n - k + 1$$

Demonstração: A palavra de código não nula com menor peso tem peso $d_{\min}$ (por definição). Ora existem sempre palavras de código sistemático com **apenas um** símbolo de informação não-nulo + $(n - k)$ símbolos de paridade. Tal palavra de código não pode ter peso maior que $1+(n - k)$. Logo, o peso mínimo do código — isto é, $d_{\min}$ — não pode ser maior do que $1 + (n - k)$. *c.q.d.*

Limite de Hamming

O número de dígitos de verificação de paridade de qualquer código linear binário (n,k) com distância mínima $d_{\min} \geq 2t + 1$ satisfaz o limite de Hamming

$$n - k \geq \log_2 \left[1 + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{t} \right]$$

Demonstração: Sendo $d_{\min} \geq 2t + 1$, então todos os énuplos de peso t ou menos (isto é, padrões de t erros ou menos) podem ser usados como *coset leaders* de uma matriz-padrão. Se o código corrige t ou menos erros por bloco, o conjunto de todos os padrões de t ou menos erros (incluindo o padrão nulo)

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{t}$$

deve ser menor ou igual que o número de *coset leaders*, 2^{n-k}:

$$\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t} \leq 2^{n-k}$$

É o mesmo que $n - k \geq \log_2 \left[1 + \binom{n}{1} + \ldots + \binom{n}{t} \right]$ *c.q.d.*

O *limite de Hamming* estabelece um valor máximo para a capacidade de correcção de erros t do código.

O limite de Hamming e o empacotamento de esferas

Vejamos como obter o limite de Hamming pensando em esferas, ou bolas:

- Existem 2^n palavras de n bits das quais 2^k pertencem a um código (n, k).
- Das que não pertencem ao código $\binom{n}{1}$ estão à distância de Hamming de 1 de qualquer uma das palavras de código.

  Ou seja, este é o número de pontos (ou palavras de n bits) que estão à superfície de uma esfera de raio 1 centrada na palavra de código.

- Existem $\binom{n}{2}$ palavras (pontos) de n bits à distância 2 da palavra de código; esses pontos estão à superfície de uma esfera de raio 2 centrada na palavra de código.
- Quantos pontos ou palavras de n bits estão à superfície e no interior de uma esfera de raio t centrada numa palavra de código (incluindo esta)?

$$1+\binom{n}{1}+\binom{n}{2}+\ldots\binom{n}{t}=\sum_{i=0}^{t}\binom{n}{i}$$

- A distância entre duas esferas de raio t que não se toquem é $d_{\min} \geq 2t+1$.
- Como existem 2^k esferas de raio t centradas nas 2^k palavras de código então em todas essas esferas o número de pontos é $2^k\sum_{i=0}^{t}\binom{n}{i}$, que não pode ser maior que o número total de pontos (dado que as esferas não se interpenetram):

$$2^n \geq 2^k\sum_{i=0}^{t}\binom{n}{i} \quad \Rightarrow \quad 2^{n-k} \geq \sum_{i=0}^{t}\binom{n}{i} \qquad (\textit{limite de Hamming})$$

A igualdade verifica-se nos códigos perfeitos, onde não há nenhuma palavra de n bits a uma distância maior que t de alguma palavra de código.

Limite de Plotkin

Imaginemos que dispomos linha a linha as 2^k palavras de código de um código (n,k). Obteremos então uma matriz de 2^k linhas e n colunas. Em cada uma dessas colunas metade dos bits (isto é, $2^k/2$) é "0" e metade é "1". Logo, o número total de "uns" que existe em todas as palavras de código é

- Número de "uns": $n\frac{2^k}{2} = n.2^{k-1}$

Como existem $2^k - 1$ palavras não nulas então

- Peso médio das palavras de código: $\frac{n.2^{k-1}}{2^k - 1}$

$$d_{min} \leq \text{peso médio}$$

isto é,

$$d_{\min} \leq \frac{n.2^{k-1}}{2^k - 1}$$

Este majorante da distância mínima é o chamado *limite de Plotkin*.

Modificações nos códigos

Códigos aumentados

$$(n,k) \quad \rightarrow \quad (n+1,k)$$

- Acrescenta-se um bit de paridade (para obter paridade par)
- Se o código original tiver distância mínima ímpar a nova distância mínima vem incrementada de 1.

  Porquê? Porque:

  - com palavras de código de peso ímpar o peso é incrementado.
  - com palavras de código de peso par o peso mantém-se.

Exemplos:

- Hamming aumentado

$(15,11), d_{\min} = 3 \quad \rightarrow \quad (16,11), d_{\min} = 4$

$(7,4), d_{\min} = 3 \quad \rightarrow \quad (8,4), d_{\min} = 4$

Peso i	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
A_i (7,4)	1	0	0	7	7	0	0	1	—
A_i (8,4)	1	0	0	0	14	0	0	0	1

$$(7,4) \quad P_{end} = \sum_{i=1}^{7} A_i p^i (1-p)^{7-i} = 7p^3(1-p)^4 + 7p^4(1-p)^3 + p^7 \approx 7.10^{-9}$$

$$(8,4) \quad P_{end} = \sum_{i=1}^{8} A_i p^i (1-p)^{8-i} = 14p^4(1-p)^3 + p^8 \approx 1{,}4.10^{-11} \qquad (!!)$$

- Golay aumentado

$(23,12), d_{\min} = 7 \quad \rightarrow \quad (24,12), d_{\min} = 8$

Modificações nos códigos

Códigos encurtados

$$(n,k) \quad \rightarrow \quad (n-l,k-l)$$

- Nas 2^k palavras do código original alguns (l) bits de informação são forçados a zero e depois removidos.

- o número de bits de informação passa a ser k - l mas o número de bits de paridade é o mesmo.

- Passa a haver 2^{k-l} palavras de código (se l = 1 passa a haver metade das palavras)

- A nova distância mínima é a mesma ou superior (depende dos bits removidos)

- Se se removerem os bits de ordem i ($i = 1,2,\ldots,k$) a nova matriz geradora obtém-se eliminando as linhas e as colunas de ordem i.

Códigos encurtados: um exemplo

Seja $\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ (código de Hamming (7, 4))

Vamos obter o código de Hamming encurtado (6, 3) removendo o terceiro bit de informação. A nova matriz geradora obtém-se da primeira eliminando a 3ª linha e a 3ª coluna:

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

Eis as novas palavras de código:

Palavras do código (7,4)		**Palavras do código (6,3)**
Original	3º bit nulo	Remoção do 3º bit
0 0 0 0 0 0 0	0 0 **0** 0 0 0 0	0 0 0 0 0 0
0 0 0 1 1 1 1	0 0 **0** 1 1 1 1	0 0 1 1 1 1
0 0 1 0 0 1 1	0 0 **0** 0 0 1 1	—
0 0 1 1 1 0 0	0 0 **0** 1 1 0 0	—
0 1 0 0 1 0 1	0 1 **0** 0 1 0 1	0 1 0 1 0 1
0 1 0 1 0 1 0	0 1 **0** 1 0 1 0	0 1 1 0 1 0
0 1 1 0 1 1 0	0 1 **0** 0 1 1 0	—
0 1 1 1 0 0 1	0 1 **0** 1 0 0 1	—
1 0 0 0 1 1 0	1 0 **0** 0 1 1 0	1 0 0 1 1 0
1 0 0 1 0 0 1	1 0 **0** 1 0 0 1	1 0 1 0 0 1
1 0 1 0 1 0 1	1 0 **0** 0 1 0 1	—
1 0 1 1 0 1 0	1 0 **0** 1 0 1 0	—
1 1 0 0 0 1 1	1 1 **0** 0 0 1 1	1 1 0 0 1 1
1 1 0 1 1 0 0	1 1 **0** 1 1 0 0	1 1 1 1 0 0
1 1 1 0 0 0 0	1 1 **0** 0 0 0 0	—
1 1 1 1 1 1 1	1 1 **0** 1 1 1 1	—
	$d_{min} = 3$	$d_{min} = 3$

A distância mínima não se alterou.

Exemplo de aumento da distância mínima com um código encurtado

Matriz geradora de um código (15,11) com distância mínima 3:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & \cdots & 0 & 1 & 0 & 1 & 0 \\ 0 & & \ddots & 0 & 1 & 1 & 0 & 0 \\ \vdots & & & 0 & 1 & 1 & 0 & 1 \\ & & \ddots & & 0 & 0 & 1 & 1 \\ & & & & 0 & 1 & 0 & 1 \\ & & (I_{11}) & & 1 & 0 & 0 & 1 \\ & & & \vdots & 1 & 0 & 1 & 1 \\ 0 & & & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & \cdots & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Se se eliminarem as linhas de **G** que em **P** têm peso par (1, 2, 3, 5, 6, 7 e 11) e também as colunas com o mesmo ordinal obtemos a matriz geradora de um código encurtado (8,4):

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

A transposta da matriz de verificação de paridade deste código encurtado é

$$\mathbf{H'}^T = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Em $\mathbf{H'}^T$ nenhuma linha é nula ($\Rightarrow d_{\min} > 1$), não há duas linhas iguais (logo, nenhum par de linhas somadas dá zero $\Rightarrow d_{\min} > 2$) e cada uma tem peso ímpar (logo, nenhum conjunto de três linhas somadas dá zero $\Rightarrow d_{\min} > 3$). Como há pelo menos um grupo de quatro linhas (1, 2, 6 e 7) que somadas dá zero $\Rightarrow d_{\min} = 4$.