2.4. Códigos cíclicos

- Polinómio gerador
- Como gerar códigos cíclicos
- Síndrome
- Divisão polinomial
- Equivalência entre a representação matricial de códigos de blocos e a representação polinomial de códigos cíclicos
- Exemplos de códigos: Hamming, BCH, Golay, Reed-Solomon e CRC

Códigos cíclicos

- Os códigos cíclicos são os códigos de blocos lineares mais úteis e populares.

Porque é que este tipo de código é importante?	**Como se codifica?**	**Como se calcula a síndrome?**
Porque a codificação e a descodificação se podem fazer facilmente com registos de deslocamento e alguma lógica adicional	Com registos de deslocamento de k andares ou de n-k andares	Com registos de deslocamento de k andares ou de n-k andares

- Os códigos cíclicos são "manuseáveis" algebricamente, com polinómios, sem necessidade de recorrer a matrizes e vectores.

- Num código cíclico um deslocamento de uma palavra de código conduz a uma outra palavra de código:

  Se $[y_n\, y_{n-1} \ldots y_0]$ for uma palavra de código $\Rightarrow$ $[y_{n-1}\, y_{n-2} \ldots y_0\, y_n]$ também é uma palavra de código.

- Mas… só n-1 palavras de código são geradas por deslocamento cíclico.

  $\Rightarrow$ para gerar por deslocamento todo o conjunto de 2^k palavras de código temos de considerar várias palavras.

Códigos cíclicos

Associemos um polinómio de grau $\leq n-1$ a cada palavra de código:

Palavra de código	**Polinómio**
$Y = (y_{n-1} y_{n-2} \cdots y_1 y_0)$	$Y(x) = y_{n-1}x^{n-1} + y_{n-2}x^{n-2} + \ldots + y_1 x + y_0$

Como formar nova palavra por deslocamento cíclico?

- Formemos o polinómio $xY(x)$:

$$xY(x) = y_{n-1}x^n + y_{n-2}x^{n-1} + \ldots + y_1 x^2 + y_0 x$$

Não é palavra de código porque, se $y_{n-1} = 1$, o polinómio tem grau n.

- Vamos dividir $xY(x)$ por $x^n + 1$:

$$\frac{xY(x)}{x^n+1} = Y_{n-1} + \frac{Y_1(x)}{x^n+1} \quad (\text{com } Y_1(x) = y_{n-2}x^{n-1} + y_{n-3}x^{n-2} + \ldots + y_0 x + y_{n-1})$$

$Y_1(x)$ é um polinómio que representa a nova palavra de código

$Y_1 = (y_{n-2} y_{n-3} \cdots y_0 y_{n-1})$ (deslocamento de Y de uma posição).

- Como $Y_1(x)$ é o resto da divisão de $xY(x)$ por $x^n + 1$ podemos escrever

$$Y_1(x) = x\ Y(x) \mod (x^n + 1)$$

Generalizando:

Se $Y(x)$ representa uma palavra de código então $x^i\ Y(x) \mod (x^n + 1)$ é também palavra de código.

$$x^i Y(x) = Q(x)(x^n + 1) + Y_i(x)$$

$Q(x)$: Quociente

$Y_i(x)$: Resto (corresponde a uma palavra de código deslocada)

Como gerar códigos cíclicos?

Vamos ver duas maneiras de calcular as palavras de código a partir da mensagem, uma para códigos não sistemáticos e outra para códigos sistemáticos.

- Seja $M(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \ldots + m_1 x + m_0$ um polinómio de grau $\leq k$-1 representando uma mensagem de k bits.

- Seja $g(x)$ um polinómio de grau $n - k$ que seja divisor de $x^n + 1$.

  A este polinómio vamos chamar *polinómio gerador* do código.

- Existem 2^k polinómios $\{X_i(x)\} \Rightarrow$ há 2^k palavras de código possíveis, a partir de um dado $g(x)$.

Códigos não sistemáticos:

- O produto $M(x)g(x)$ tem grau $\leq n-1$ e pode representar uma palavra de código.

- Representemos as 2^k palavras de código por

$$Y_m(x) = M_m(x)\, g(x) \qquad m = 1, 2, \ldots, 2^k$$

Códigos sistemáticos:

Seja $C(x) = c_{q-1}\, x^{q-1} + \ldots + c_1\, x + c_0$ o polinómio de teste de paridade (em que $q = n-k$). Cada palavra de código é obtida percorrendo a seguinte sequência de operações:

1. Calcula-se $x^{n-k}M(x)$.

2. Calcula-se $C(x) = x^{n-k}M(x) \pmod{g(x)}$.

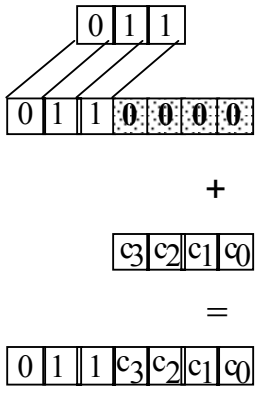
3. Determina-se a palavra de código calculando $Y(x) = x^{n-k}\, M(x) + C(x)$

Se $Y(x)$ representa uma palavra de código então $Y(x)$ será múltiplo de $g(x)$.

No exemplo seguinte vamos ver a razão destas operações.

Exemplo de codificação

Código cíclico (7,3) sistemático



$= M(x) = x+1$ mensagem original

$= x^4 M(x) = x^5 + x^4$ mensagem deslocada

$= C(x)$ paridade

$= Y(x)$ palavra de código

A figura mostra que $Y(x) = x^4 M(x) + C(x)$.

Quanto vale *C*(*x*)?

R.: $Y(x)$ é múltiplo de $g(x)$, por definição $\Rightarrow$ $Y(x) = x^4 M(x) + C(x) = Q(x)g(x)$

$$\Rightarrow \quad \frac{x^4 M(x)}{g(x)} + \frac{C(x)}{g(x)} = Q(x) \quad \Rightarrow \quad \frac{x^4 M(x)}{g(x)} = Q(x) + \frac{C(x)}{g(x)}$$

$C(x) = c_3 x^3 + c_2 x^2 + c_1 x + c_0$ representa o resto da divisão de $x^4 M(x)$ por $g(x)$.

Se $g(x) = x^4 + x^2 + x + 1 \Rightarrow C(x) = x^3 + 1$

(ou $c_3 = 1, \quad c_2 = 0, \quad c_1 = 0, \quad c_0 = 1.$)

Resumindo… **para calcular uma palavra de código de um código cíclico sistemático (*n*, *k*) devemos**:

1.	Factorizar $x^n + 1$ e escolher um factor com grau $n-k$. Será esse o polinómio gerador $g(x)$.
2.	Calcular $x^{n-k} M(x)$.
3.	Dividir $x^{n-k} M(x)$ por $g(x)$ e achar o resto, $C(x)$.
4.	Determinar $Y(x) = x^{n-k} M(x) + C(x)$.

A factorização de $x^n + 1$ é apresentada no Anexo 5.

Polinómios usados em códigos cíclicos

Polinómios		grau
Polinómio gerador	*g(x)*	*n-k*
Polinómio de verificação de paridade	*h(x)*	*k*
Polinómio de informação	*M(x)*	$\leq k-1$
Polinómio de palavra de código	*Y(x)*	$\leq n-1$
Polinómio de erro	*e(x)*	$\leq n-1$
Polinómio recebido	*Z(x)*	$\leq n-1$
Polinómio da síndrome	*S(x)*	$\leq n-k-1$

Os polinómios *g(x)* e *h(x)* estão relacionados através de $x^n + 1 = g(x)\,h(x)$.

Síndrome:

Dado um vector recebido **Z** a síndrome calcula-se de *S(x) = Z(x)* mod *g(x)*. Se *Z(x)* for um polinómio de código válido então *g(x)* será um factor de *Z(x)* e $\frac{Z(x)}{g(x)}$ terá um resto nulo. Se o resto não for nulo é porque houve erros na transmissão.

$$S(x) = Z(x) \bmod g(x) = \left[Y(x) + e(x) \right] \bmod g(x) =$$
$$= \underbrace{Y(x) \bmod g(x)}_{0} + e(x) \bmod g(x) = e(x) \bmod g(x)$$

Para cada polinómio *e(x)* corrigível calcula-se e tabula-se *S(x)* como se mostra na seguinte *tabela de síndromes*:

e(x)	*S(x)*
1	$1 \bmod g(x)$
x	$x \bmod g(x)$
x^2	$x^2 \bmod g(x)$
...	...
$1+x$	$(1+x) \bmod g(x)$
$1+x^2$	$(1+x^2) \bmod g(x)$
...	...

Códigos cíclicos — um exemplo

Exemplo: código (7,4)

Vamos determinar a palavra de código correspondente à mensagem $M=[1010] \Rightarrow M(x)=x^3+0+x+0$.

Por definição o polinómio gerador é factor de x^n+1. Ora o polinómio x^7+1 tem os seguintes factores:

$$x^7+1=(x+1)(x^3+x^2+1)(x^3+x+1)$$

Se quisermos gerar um código cíclico (7, 4) o polinómio gerador tem grau 7-4=3 e pode ser

$$g_1(x)=x^3+x^2+1 \qquad \text{ou} \qquad g_2(x)=x^3+x+1$$

Vamos escolher $g_2(x)$, por exemplo.

- Cálculo do polinómio de teste de paridade, $C(x)$:

  1. $x^{n-k}M(x)=x^3M(x)=x^6+x^4$
  2. Dividindo $x^{n-k}M(x)$ por $g(x)$:

$$\frac{x^6+x^4}{x^3+x+1}=x^3+1+\frac{x+1}{x^3+x+1}$$

  Portanto, $C(x)=x+1$.

- Palavra de código: $Y(x)=x^3M(x)+C(x)=x^6+x^4+x+1$

$$\Rightarrow Y=[1010011]$$

Códigos cíclicos — um exemplo (cont.)

O código $(7,4)$ precedente pode ser gerado por um registo de deslocamento de 3 andares. O polinómio gerador é $g(x) = x^3 + g_2 x^2 + g_1 x + 1 = x^3 + x + 1$.

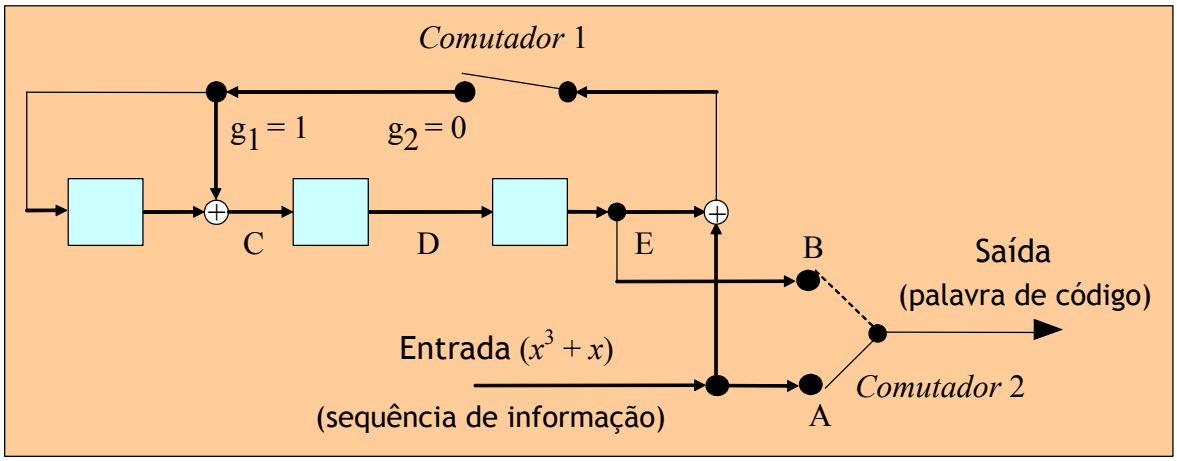
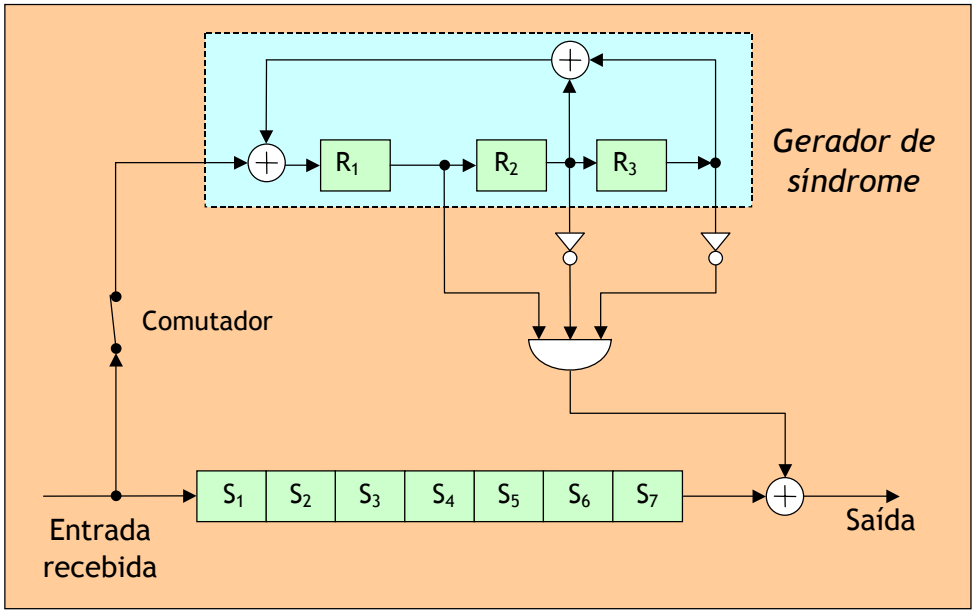


TABELA DE CONTEÚDOS DO CODIFICADOR (7, 4)

Deslocamento	Entrada	Comutador 1	Comutador 2	C	D	E	Saída
1	1	Fechado	A	1	0	0	1
2	0	Fechado	A	1	1	0	0
3	1	Fechado	A	0	1	1	1
4	0	Fechado	A	1	0	1	0
5	--	Aberto	B	1	1	0	→ 0
6	--	Aberto	B	0	1	1	→ 1
7	--	Aberto	B	0	0	1	→ 1

Durante os primeiros 4 deslocamentos o comutador 1 está fechado e o comutador 2 está na posição A. Só após esses 4 deslocamentos ($k = 4$) é que os 3 bits de paridade são acrescentados.

Código cíclico (7, 4) — Descodificação



Suponhamos que a palavra enviada foi [1010011] e que a palavra recebida foi [1 0 1 1 0 1 1].

TABELA DE CONTEÚDOS DO DESCODIFICADOR (7,4)

Deslocam.	Entrada	Comut.	R_1	R_2	R_3	AND	S_1	S_2	S_3	S_4	S_5	S_6	S_7	Saída
1	1	Fech.	1	0	0	--	1	--	--	--	--	--	--	--
2	0	"	0	1	0	--	0	1	--	--	--	--	--	--
3	1	"	0	0	1	--	1	0	1	--	--	--	--	--
4	1	"	0	0	0	--	1	1	0	1	--	--	--	--
5	0	"	0	0	0	--	0	1	1	0	1	--	--	--
6	1	"	1	0	0	--	1	0	1	1	0	1	--	--
7	1	"	1	1	0	--	1	1	0	1	1	0	1	--
8	--	Aberta	1	1	1	0	--	1	1	0	1	1	0	1
9	--	"	0	1	1	0	--	--	1	1	0	1	1	0
10	--	"	0	0	1	0	--	--	--	1	1	0	1	1
11	--	"	1	0	0	1	--	--	--	--	1	1	0	0
12	--	"	0	1	0	0	--	--	--	--	--	1	1	0
13	--	"	1	0	1	0	--	--	--	--	--	--	1	1
14	--	"	1	1	0	0	--	--	--	--	--	--	--	1

No 11º deslocamento a saída do AND é 1 (e é a única ocasião em que isso sucede), a qual vai inverter ($1 \oplus 1 = 0$) o 4º símbolo recebido, que estava em erro.

Calculando as matrizes **G** e **H** verificar-se-ia que **S** = [0 11], a que corresponderia um vector de erro com erro na 4ª posição. Logo, o rectângulo tracejado desempenha as funções de um gerador de síndrome.

Circuito para dividir polinómios

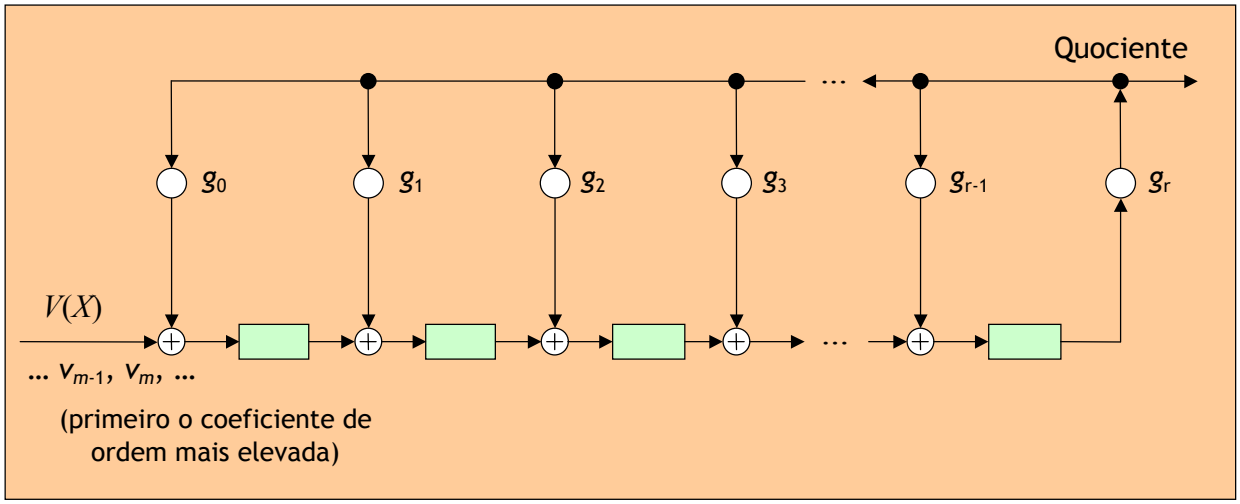
Dados dois polinómios $V(X)$ e $g(X)$:

$$V(X) = v_m X^m + v_{m-1} X^{m-1} + ... + v_1 X + v_0$$

$$g(X) = g_r X^r + ... + g_1 X + g_0 \, (m \geq r)$$

o circuito seguinte realiza a **divisão polinomial** de $V(X)$ por $g(X)$, determinando o **quociente** e o **resto**:

$$\frac{V(X)}{g(X)} = q(X) + \frac{r(X)}{g(X)}$$



Após m+1 deslocamentos no registo de deslocamento o ***quociente*** *foi apresentado em série na saída* e o ***resto*** *reside no* registo de deslocamento.

Divisão polinomial

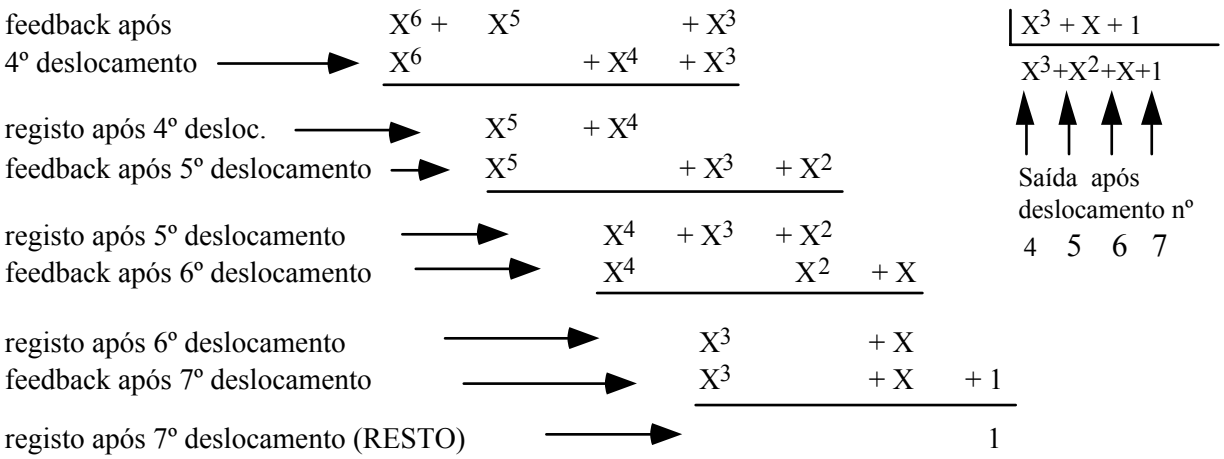
Exemplo:

Use um circuito divisor idêntico ao apresentado para dividir $V(X) = X^3 + X^5 + X^6$ (**V** = 1 1 0 1 0 0 0) por $g(X) = X^3 + X + 1$. Determine o quociente e o resto. Compare a implementação em circuito com os passos da divisão polinomial feita à mão.

R.: Queremos determinar o quociente e o resto:

$$\frac{X^6 + X^5 + X^3}{X^3 + X + 1} = q(X) + \frac{r(X)}{X^3 + X + 1}$$

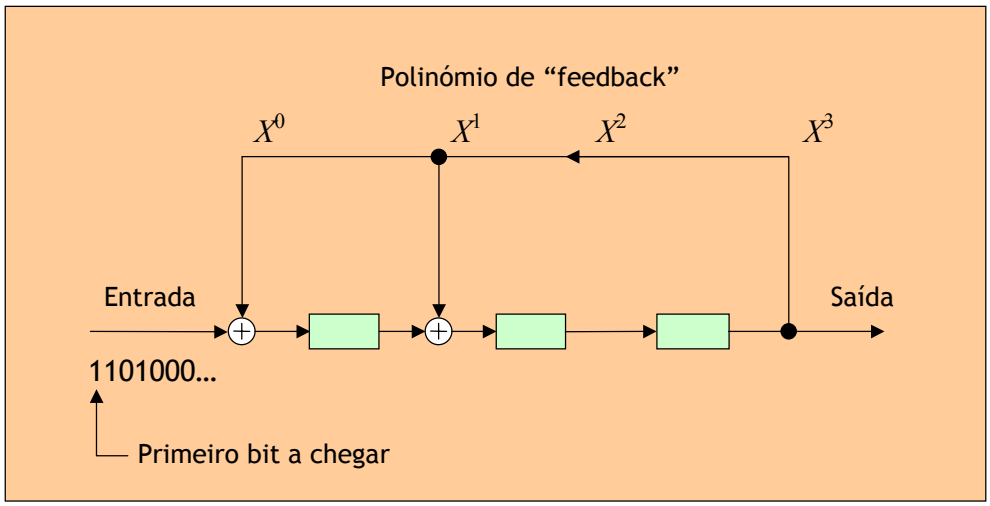
À mão:



Portanto $q(X) = X^3 + X^2 + X + 1$ e *r(X) = 1*.

Divisão polinomial com o circuito

O circuito divisor é este:



Fila de entrada	Deslocamento nº	Conteúdo registo	Saída
0 0 0 1 0 1 1	0	0 0 0	-
0 0 0 1 0 1	1	1 0 0	0
0 0 0 1 0	2	1 1 0	0
0 0 0 1	3	0 1 1	0
0 0 0	4	0 1 1	1
0 0	5	1 1 1	1
0	6	1 0 1	1
--	7	1 0 0	1

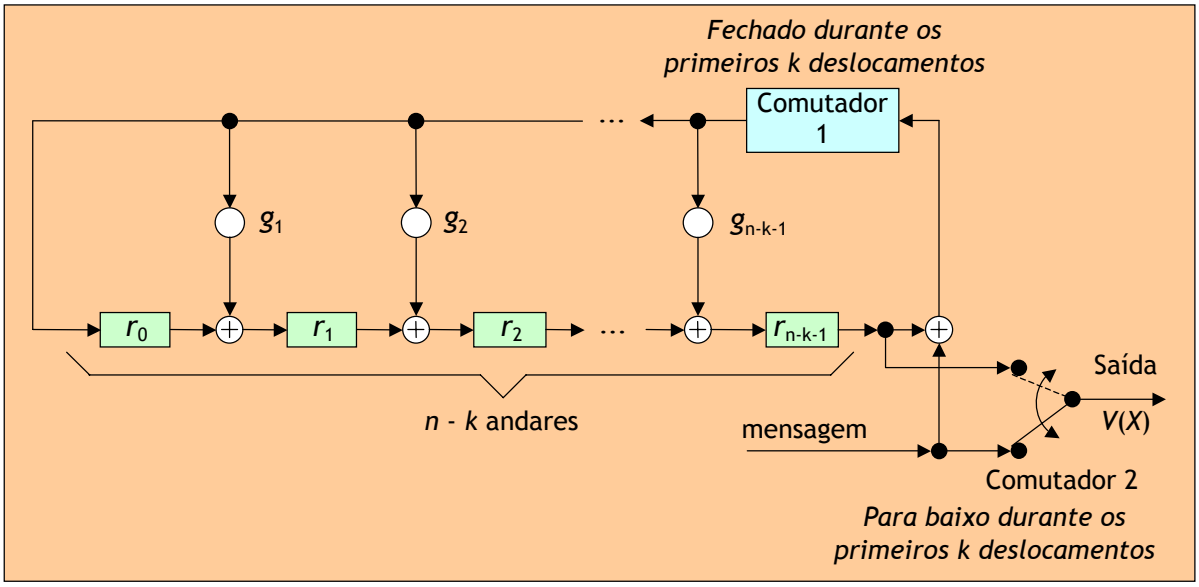
Coeficientes do quociente: 1111 $\rightarrow$ $q(X) = X^3 + X^2 + X + 1$

Coeficientes do resto: 100 $\rightarrow$ $r(X) = 1$

isto é, $$\frac{X^6 + X^5 + X^3}{X^3 + X + 1} = X^3 + X^2 + X + 1 + \frac{1}{X^3 + X + 1}$$

Codificação sistemática com um registo de deslocamento de *n* - *k* andares

Forma Geral:



Exemplo: Codificar a mensagem **M** = [1 1 0 1] num vector de código (7, 4) usando o polinómio gerador $g(x) = x^3 + x + 1$.

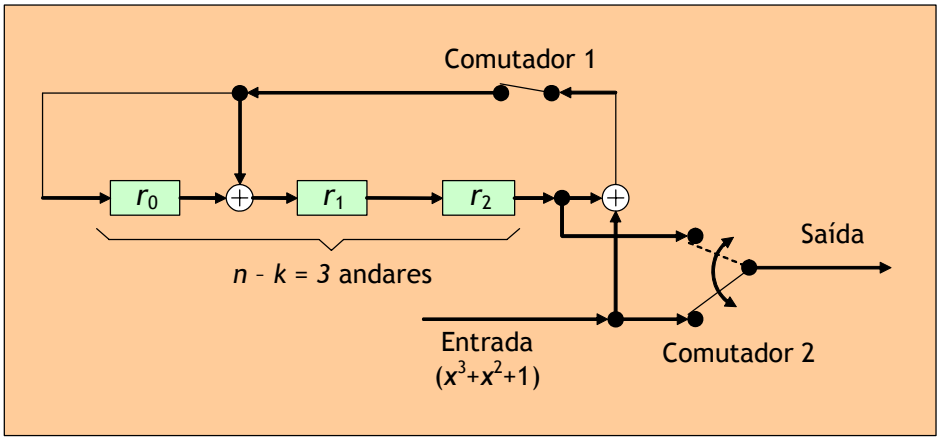
R.: $M = [1101] \Rightarrow M(x) = x^3 + x^2 + 1$

$x^{n-k} M(x) = x^3 M(x) = x^6 + x^5 + x^3$

$x^{n-k} M(x) = q(x)g(x) + r(x)$ ← Código sistemático

Paridade $\rightarrow r(x) = x^6 + x^5 + x^3 \mod(x^3 + x + 1)$

Circuito:

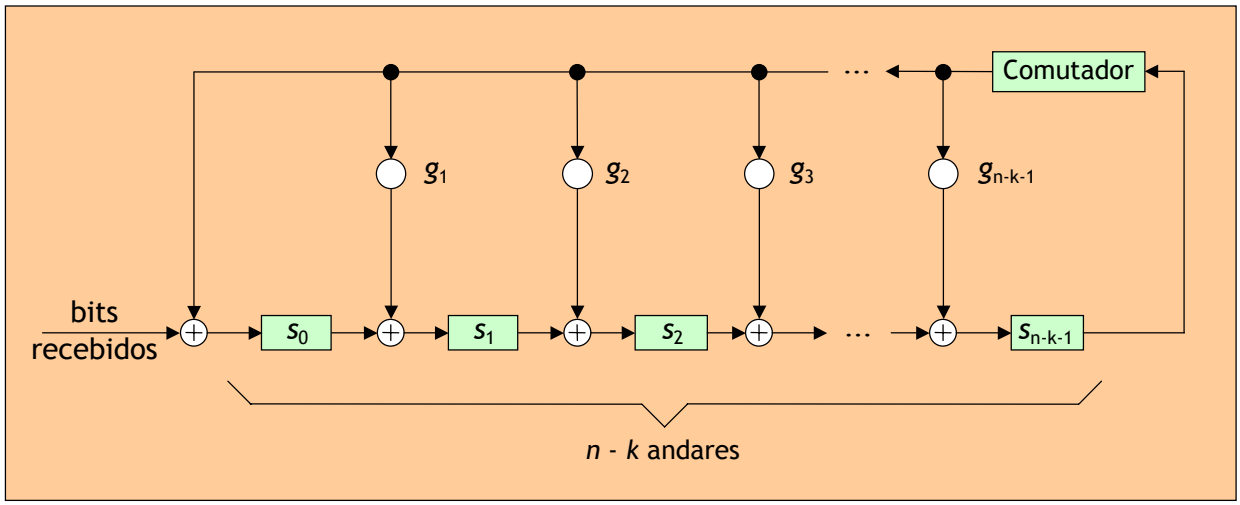


Vector de código de saída: [1101 001] $(x^6+x^5+x^3+1)$

Descodificação com um registo de deslocamento de *n* - *k* andares

O cálculo da síndrome na descodificação de códigos cíclicos (*n*,*k*) realiza-se com um circuito de *n*-*k* andares análogo ao que foi usado na codificação:

Calculador de síndromes



A síndrome reside no registo (isto é, é o seu conteúdo) no fim do deslocamento dos bits da palavra recebida à esquerda.

Equivalência entre a representação matricial de códigos de blocos e a representação polinomial de códigos cíclicos

Seja $g(x)$ o polinómio gerador de um código cíclico (n, k).

- A linha de ordem i da matriz geradora equivalente, **G**, de um código de blocos sistemático vem dada por

$$x^{n-i} + R_i(x) \qquad i = 1, 2, ..., k \qquad (\text{com } R_i(x) = x^{n-i} \bmod g(x))$$

isto é, $\dfrac{x^{n-i}}{g(x)} = Q_i(x) + \dfrac{R_i(x)}{g(x)}$ ou $x^{n-i} = Q_i(x)\,g(x) + R_i(x) \quad i = 1,2,...,k$

$R_i(x)$ representa o resto da divisão de x^{n-i} pelo polinómio gerador $g(x)$.

Nota: a última linha de G corresponde sempre ao polinómio gerador.

- Se estivermos interessados apenas na submatriz **P**, as suas k linhas vêm dadas por:

$$R_i(x) = x^{n-i} \bmod g(x) \qquad i = 1,2,...,k$$

Exemplo: Seja o código (7,3) definido pelo polinómio $g(x) = x^4 + x^2 + x + 1$.

- 1ª linha de **P**: $x^{7-1} \bmod g(x) = x^6 \bmod g(x) = x^3 + x + 1 \quad \Rightarrow \quad [1011]$
- 2ª linha de **P**: $x^{7-2} \bmod g(x) = x^5 \bmod g(x) = x^3 + x^2 + x \quad \Rightarrow \quad [1110]$
- 3ª linha de **P**: $x^{7-3} \bmod g(x) = x^4 \bmod g(x) = x^2 + x + 1 \quad \Rightarrow \quad [0111]$

Logo, $\boldsymbol{G} = \begin{bmatrix} 1001011 \\ 0101110 \\ 0010111 \end{bmatrix} \leftarrow g(x)$

Exemplos de códigos de blocos

Código de Hamming

São os códigos de blocos mais simples.

- Estrutura: $(n,k) = (2^m - 1, 2^m - 1 - m)$ $\quad m = 2, 3, \ldots$
- Distância mínima = 3
- Corrigem *todos* os erros simples (t=1) $\Rightarrow$ é um código *perfeito*.
- Detectam todas as combinações de 2 ou menos erros dentro de um bloco ($l = 2$).

Código de Golay Aumentado (24, 12)

- É um dos códigos de blocos mais úteis.
- O código é formado acrescentando um bit de paridade ao código perfeito (23,12) (código de Golay). Este bit extra aumenta a distância mínima $d_{\min}$ de 7 para 8.
- O código corrige todos os erros *triplos* e alguns (mas não todos) erros quádruplos.
- A taxa do código é 1/2 pelo que é mais fácil de implementar (relativamente a "relógios") que o código de Golay original.
- Os códigos de Golay aumentados são consideravelmente mais poderosos que os códigos de Hamming. Em contrapartida:
  - O descodificador é mais complexo.
  - A taxa de código é mais baixa.
  - Há uma maior expansão de largura de banda.

Exemplos de códigos de blocos: o código BCH

B — Bose **C** — Chaudhuri **H** — Hocquenghem

São dos mais importantes códigos cíclicos e são uma generalização dos códigos de Hamming, permitindo correcções de erros múltiplos.

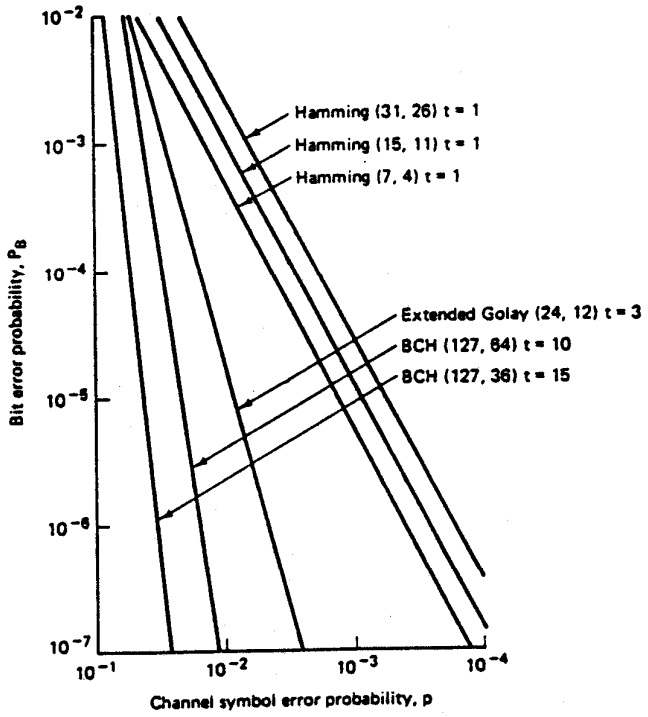
Para um dado comprimento de bloco, n, podem definir-se códigos com uma gama alargada de taxas e capacidades de correcção de erros:

$$\left.\begin{array}{c}\text{Sendo } t \text{ o nº de erros corrigíveis/palavra}\\ \text{e } m \text{ um inteiro arbitrário}\end{array}\right\} \Rightarrow \left|\begin{array}{ll} n = 2^m - 1 & m > 2 \\ n - k \le mt & \end{array}\right.$$

TABLE 5.2 Generators of Primitive BCH Codes (in octal)

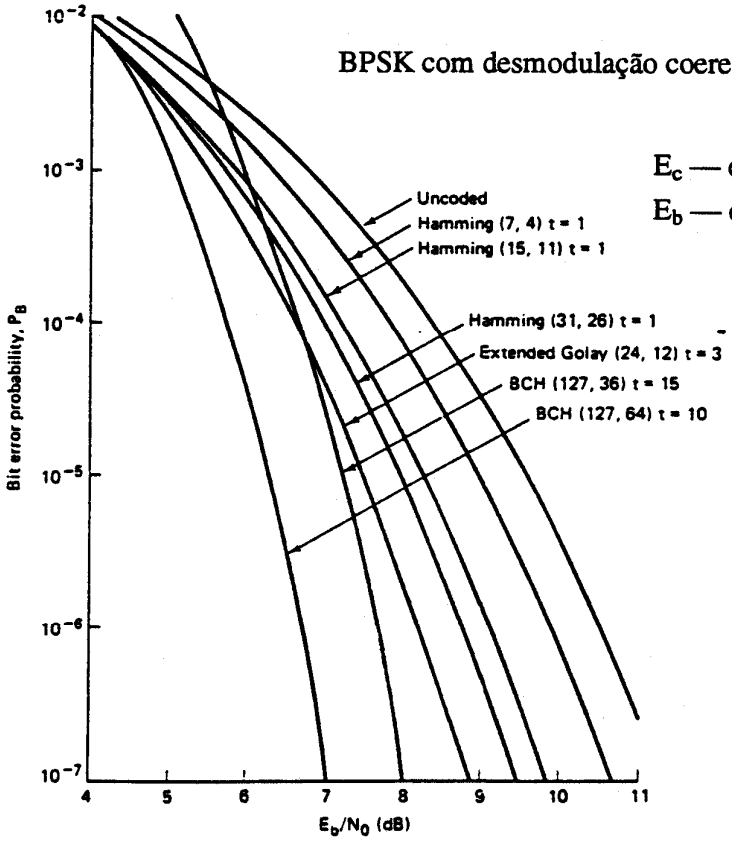
n	k	t	$g(x)$	n	k	t	$g(x)$
7	4	1	13 (→ 001011 → $\alpha^3+\alpha+1$)	255	171	11	154162142123423560770616306 37
15	11	1	23 (→ 010011 → $\alpha^4+\alpha+1$)		163	12	75004155100756025515747245146 01
	7	2	721		155	13	37575130054076650157225064646776 33
	5	3	2467 etc.		147	14	16421301735371655253041653054410117 11
31	26	1	45		139	15	46140173206017556157072273024745356744 5
	21	2	3551		131	18	21571333147151015126125027744214202 41
	16	3	107657				65471
	11	5	5423325		123	19	12061405224206600371721032651614122 62
	6	7	313365047				72506267
63	57	1	103		115	21	60526665572100247263636404600276352 55
	51	2	12471				6313472737
	45	3	1701317		107	22	22205772322066256312417300235347420 17
	39	4	166623567				6574750154441
	36	5	1033500423		99	23	10656667253473174222741416201574332 25
	30	6	157464165547				241107642236431
	24	7	17323260404441		91	25	67502650303274441727236317247325110 75
	18	10	1363026512351725				550762720724344561
	16	11	6331141367235453		87	26	11013676341474323643523163430717204 62
	10	13	472622305527250155				06722545273311721317
	7	15	523104554350327173 7		79	27	66700035637657500020270344207366174 62
127	120	1	211				10153267117665413423 55
	113	2	41567		71	29	24024710520644321515554172112331163 20
	106	3	11554743				54442503625576432217060 35
	99	4	3447023271		63	30	10754475055163544325315217357707003 66
	92	5	624730022327				61117264552676136567025433 01
	85	6	130704476322273		55	31	73154252035011001330152753060320543 25
	78	7	26230002166130115				41432675501055704442603547361 7
	71	9	6255010713253127753		47	42	25335420170626465630330413774062331 75
	64	10	1206534025570773100045				12333414544604500506602455254317 3
	57	11	335265252505705053517721		45	43	15202056055234161131101346376423701 56
	50	13	54446512523314012421501421				36700244707623730332021570250515 41
	43	14	17721772213651227521220574343		37	45	51363302550670074141774472454375304 20
	36	15	31460746665220750447645747217 35				73570617432343234764435473403044 003
	29	21	40311446136767060366753014117615 5		29	47	30257155366730714655270640123613771 15
	22	23	12337607040472252243544562663764704 3				34224232420117411406025475741040356
	15	27	22057042445604554770523013762217604353				5037
	8	31	70472640527510306514762242715677331302 17		21	55	12562152570603326560017731536076121 03
255	247	1	435				22734140565307454252115312161446651
	239	2	267543				3473725
	231	3	156720665		13	59	46417320050525645444265737142500660 04
	223	4	75626641375				33067744547656140317467721357026134
	215	5	23157564726421				460500547
	207	6	16176560567636227		9	63	15726025217472463201031043253551346 1
	199	7	7633031270420722341				41623672120440745451127661155477055
	191	8	26634701761153337145 67				61677516057
	187	9	52755313540001322236351				
	179	10	22624710717340432416300455				

Comparação entre códigos de blocos



CANAL BINÁRIO SIMÉTRICO

Bit error probability versus channel symbol error probability for several block codes.



BPSK com desmodulação coerente $p = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) = Q\left(\sqrt{\frac{2kE_b}{nN_0}}\right)$

E_c — energia de símbolo codificado

E_b — energia de bit de informação

P_B versus E_b/N_0 for coherently demodulated BPSK over a Gaussian channel for several block codes.

Exemplos de códigos de blocos: Cyclic Redundancy Check Codes (CRC)

Os códigos cíclicos usados para detecção de erros são habitualmente designados por CRCs. Os mais usados são caracterizados pelos polinómios geradores da tabela seguinte.

Polinómios geradores dos principais códigos CRC

Código	*Polinómio gerador*
CRC-12	$x^{12}+x^{11}+x^3+x^2+x+1$
CRC-16	$x^{16}+x^{15}+x^2+1$
CRC-32 (Ethernet)	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$
CRC-CCITT (norma X25)	$x^{16}+x^{12}+x^5+1$

CRC-16, por exemplo, significa que o número de bits de paridade é 16.

- Uma palavra com erro é detectada se, dividindo o seu polinómio pelo polinómio gerador, o resto (*checksum*) não for zero.

- Nem todos os códigos CRC são cíclicos. Para o serem o seu polinómio gerador tem de ser um factor de x^n+1 dependendo, portanto, do comprimento n.

- Definição: uma rajada de erros ("CRC Error Burst") de comprimento B numa palavra recebida de n bits é uma sequência contígua de B bits em que o primeiro e o último bits e qualquer número de bits intermédios são recebidos com erro.

Códigos detectores de erros (CRC)

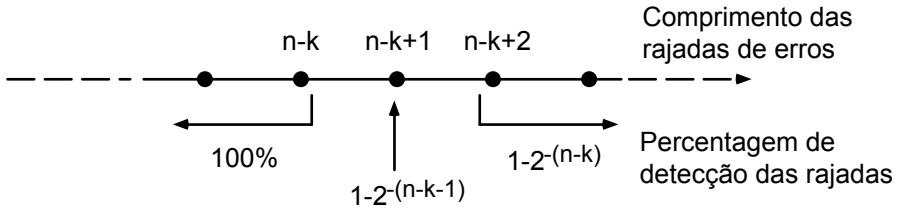
Os códigos (n,k) binários são capazes de detectar os seguintes padrões de erro:

- *Todas as rajadas de erros CRC de comprimento $n-k$ ou menor;*
- *Uma fracção, $1-2^{-(n-k-1)}$, de rajadas de comprimento $n-k+1$;*

  (isto é, apenas não detecta a fracção $\frac{1}{2^{n-k-1}}$ destas rajadas).
- *Uma fracção, $1-2^{-(n-k)}$, de rajadas de comprimento maior que $n-k+1$;*

  (isto é, apenas não detecta a fracção $\frac{1}{2^{n-k}}$ destas rajadas).
- *Todas as combinações de $d_{\min}-1$, ou menos, erros, como sempre;*
- *Todos os padrões de erro com um número ímpar de erros se o polinómio gerador do código, $g(x)$, tiver um número par de coeficientes não nulos.*

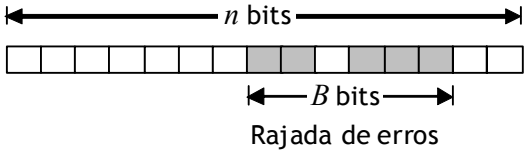
Percentagem de rajadas de erros detectadas com códigos CRC



Tipo de erros	CRC-12	CRC-16 CRC-CCITT	CRC-32
Rajadas de comprimento $\leq n-k$	100%	100%	100%
Rajadas de comprimento $n-k+1$	99,951%	99,997%	99,99999995%
Rajadas de comprimento $>n-k+1$	99,976%	99,998%	99,99999998%

Códigos detectores de erros (CRC)

Uma rajada de erros de *B* bits é representada pelo polinómio $e(x) = x^i e_B(x)$, em que $e_B(x) = x^{B-1} + \ldots + 1$ e x^i localiza a rajada no padrão de erros de *n* bits. Em $e_B(x)$ alguns coeficientes podem ser nulos, como na figura seguinte.



Qualquer código cíclico (n, k) pode detectar todas as rajadas de erros de comprimento igual ou inferior a n – k.

Demonstração:

A síndrome vale

$$S(x) = e(x) \bmod g(x) = x^i e_B(x) \bmod g(x).$$

Ora nem x^i nem $e_B(x)$ são múltiplos de $g(x)$ se $B \le n-k$.

- no primeiro caso porque o termo constante de $g(x)$ é não-nulo;

$$g(x) = x^{n-k} + \ldots + 1 \qquad \Rightarrow \qquad x^i \bmod g(x) \neq 0$$

- no segundo caso porque $e_B(x)$ tem grau $B-1$, logo, não pertence ao código.

> Não esquecer que num código cíclico só há um polinómio de código de menor grau, que é o polinómio gerador, de grau $n-k$. Ou seja, não há nenhum polinómio de código de grau inferior a $n-k$.

Portanto, $S(x) = x^i e_B(x) \bmod g(x) \neq 0$ e os erros são detectados.

Exemplo:

Seja $g(x) = x^4 + x^2 + x + 1$ e rajada $e(x) = x^2(x^3 + x^2 + 1) = x^5 + x^4 + x^2$:

$$e(x) \bmod g(x) = x^3 + x^2 + 1 \neq 0$$

Exemplos de códigos de blocos: o código de Reed-Solomon (RS)

- O código de Reed-Solomon é um código não-binário que constitui uma sub-classe dos códigos BCH.

- Os códigos RS possuem a maior distância mínima possível (n - k + 1) entre códigos lineares com os mesmos comprimentos de blocos de entrada (k) e saída (n) do codificador.

- São muito úteis na correcção de "burst errors" (erros agrupados).

- Existem técnicas eficientes de descodificação de códigos RS.

- Um código RS com correcção de t símbolos tem os seguintes parâmetros:

    - Comprimento do bloco: $n = 2^m$-1 símbolos

    - Tamanho da mensagem: k símbolos

    - Número de símbolos de paridade: n-$k = 2t$ símbolos

    - Distância mínima: $d_{\min} = 2t$+1 símbolos

    - A distância mínima é igual ao número de símbolos de paridade + 1
      (o limite de Singleton é atingido)

    - t é igual a metade dos símbolos de paridade.

- O codificador agrupa os bits de entrada em grupos de km bits, isto é, em grupos de k símbolos, em que cada símbolo é composto por m bits.

- A cada k símbolos o codificador acrescenta n-k símbolos de paridade.

- Os códigos RS são por vezes usados num modo *concatenado*. Por exemplo:
    - O código interior é um código convolucional
    - O código exterior é um código Reed-Solomon

- Nos discos compactos (CD) são usados códigos RS concatenados.

Um exemplo com o código de Reed-Solomon

Um código RS corrector de erros simples usa símbolos com bytes de 2 bits, isto é, $t = 1$ e $m = 2$. Se os quatro símbolos possíveis forem designados por 0, 1, 2 e 3, podemos escrever a sua representação binária como

0 : 00
1 : 01
2 : 10
3 : 11

Este código tem os seguintes parâmetros:

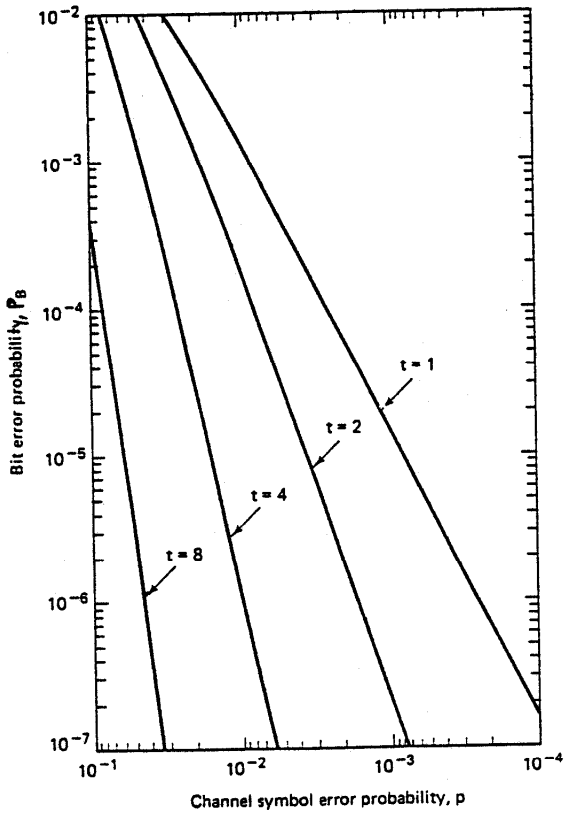
$n = 2^2 - 1 = 3$ bytes = 6 bits

$n - k = 2t = 2$ bytes = 4 bits

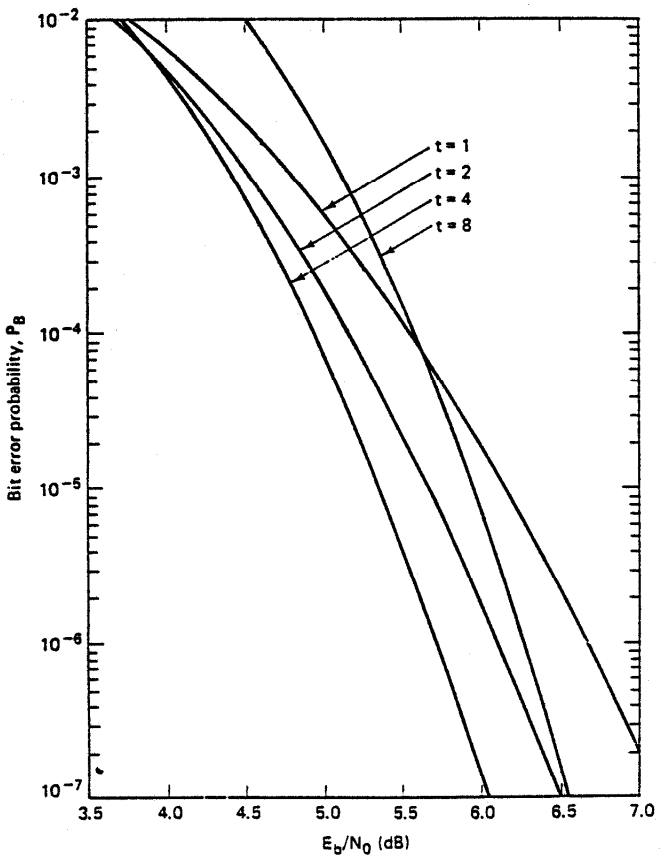
Taxa do código: $R = \frac{k}{n} = \frac{1}{3}$

Curvas de probabilidade de erro com o código de Reed-Solomon

Probabilidade de bit errado para um código RS com $n = 31$ (trinta e um símbolos de 5 bits por bloco de código):



P_B versus p for 32-ary orthogonal signaling and $n = 31$, t-error-correcting Reed–Solomon coding. (Reprinted with permission from *Data Communications, Networks, and Systems*, ed. Thomas C. Bartee, Howard W. Sams Company, Indianapolis, Ind., 1985, p. 311. Originally published in J. P. Odenwalder, *Error Control Coding Handbook*, M/A-COM LINKABIT, Inc., San Diego, Calif., July 15, 1976, p. 91.)

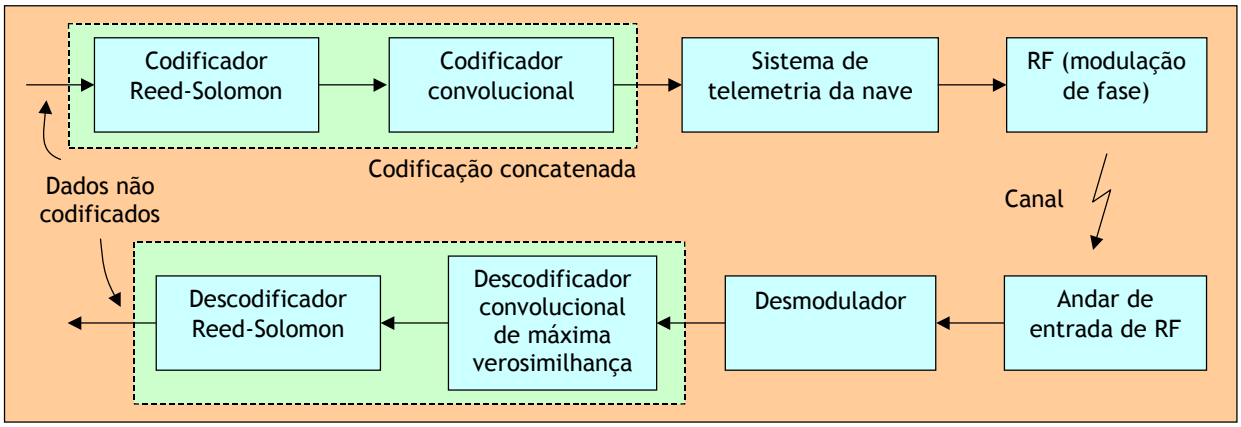


Bit error probability versus E_b/N_0 performance of several $n = 31$, t-error-correcting Reed–Solomon coding systems with 32-ary MFSK modulation over an AWGN channel. (Reprinted with permission from *Data Communications, Networks, and Systems*, ed. Thomas C. Bartee, Howard W. Sams Company, Indianapolis, Ind., 1985, p. 312. Originally published in J. P. Odenwalder, *Error Control Coding Handbook*, M/A-COM LINKABIT, Inc., San Diego, Calif., July 15, 1976, p. 92.)

Anexo 1

Aplicação de codificação de canal a transmissões espaciais

O conceito de codificação concatenada usado pela NASA

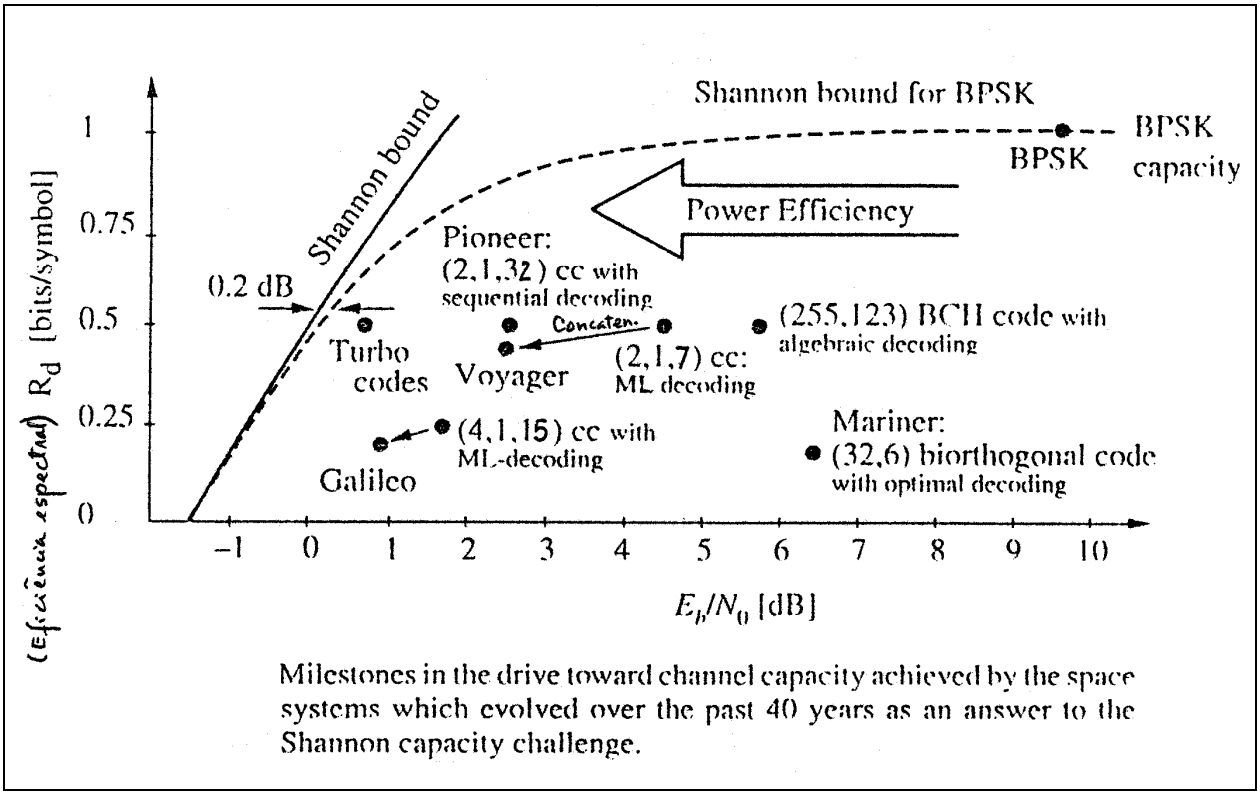


- O código interior (código convolucional de taxa 1/2 e comprimento de restrição 7) é o último a ser usado na codificação, na nave espacial, e o primeiro a ser usado na descodificação em Terra.
    - algoritmo usado: algoritmo de Viterbi com decisões brandas.
- O codificador exterior é um codificador Reed-Solomon que alimenta o codificador convolucional da nave.

(Adaptado de *IEEE Communications Magazine*, Setembro de 1990)

Anexo 2

Evolução dos códigos e seu uso em sondas espaciais



Milestones in the drive toward channel capacity achieved by the space systems which evolved over the past 40 years as an answer to the Shannon capacity challenge.

- BPSK, detecção coerente e sem codificação: $P_b = 10^{-5}$ @ $E_b/N_0 = 9{,}6$ dB
- BPSK, detecção coerente e com codificação:

Sonda	Código	E_b/N_0 para $P_b = 10^{-5}$
Mariner e Viking	Biortogonal (Reed-Muller) (32,6)	6,4 dB
Pioneer 10 (1972, Júpiter) Pioneer 11 (1973, Saturno)	Convolucional (2,1,32) não sistemático, descodificação sequencial	2,5 dB
Voyager (1977)	Convolucional (2,1,7), descodificação de Viterbi, "soft decision" Com concatenação, RS(255,223)	4,5 dB ↓ 2,5 dB
Galileo (1989, Júpiter)	Convolucional (4,1,15), BVD Com concatenação, RS(255,223)	1,75 dB ↓ ≈1 dB
	Turbo-códigos, descodificação iterativa	0,7 dB

Fonte: C. Schlegel, "Trellis Coding", IEEE Press, 1997.

Anexo 3

Voyager: algumas especificações e características

- Emissores - 20 W
- Frequência "downlink" - Banda X (8,5 GHz)
- Antenas - 3,66 m (48,2 dB de ganho)
- Distância a Neptuno = $4,42 \cdot 10^9$ km
- Taxa de transmissão = 21600 bit/s
- Código de canal no "downlink" - Código convolucional, comprimento de restrição 7, taxa 1/2 (norma da NASA)

  Probabilidade de erro requerida $\leq 5.10^{-3}$ (para $\frac{E_b}{N_0} = 2,34\ dB$, "the lowest anywhere ever").

- Também foi usado: Código de Golay (24,12), como código exterior concatenado, para $P_e = 10^{-5}$.
- Após encontro com Urano (1986) foi usado o código concatenado:

  - Exterior: Código Reed-Solomon (255,223)
  - Interior: Código Convolucional 1/2, 7 (como em cima)
  - Profundidade de entrelaçamento: 4 ("a deep-space first")

- Na altura do lançamento (1977) não havia ainda descodificador RS. A nave transportou o codificador mas o descodificador só foi construído mais tarde, a tempo do encontro com Urano.

  Conseguiu-se com este esquema concatenado melhorado:

$$P_e = 10^{-6}, \text{ com } \frac{E_b}{N_0} = 2,43\ dB$$
$$\downarrow$$

  Está a 4,02 dB do limite do Shannon (-1,59 dB), o que, no momento, era "the closest anywhere for this low error probability".

(In *IEEE Communications Magazine*, Setembro de 1990)

Anexo 4

Características dos códigos mais conhecidos

Hamming

Comprimento do bloco: $n = 2^m - 1$

Nº de bits da mensagem: $k = 2^m - m - 1$

Nº de bits de paridade: $n - k = m$

Distância mínima: $d_{\min} = 3$

Nº de erros corrigíveis: $t = 1$ (código perfeito)

CRC

CRC-12 $g(x) = x^{12} + x^{11} + x^3 + x^2 + x + 1$
$k = 6$

CRC-16 $g(x) = x^{16} + x^{15} + x^2 + 1$
$k = 8$

CRC-CCITT $g(x) = x^{16} + x^{12} + x^5 + 1$
$k = 8$

Golay

Comprimento do bloco: $n = 23$

Nº de bits da mensagem: $k = 12$

Nº de bits de paridade: $n - k = 11$

Distância mínima: $d_{\min} = 7$

Nº de erros corrigíveis: $t = 3$ (código perfeito)

$g_1(x) = x^{11} + x^{10} + x^6 + x^5 + x^4 + x^2 + 1$

ou

$g_2(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$

$x^{23} + 1 = (x+1)\, g_1(x)\, g_2(x)$

Golay aumentado
(não é cíclico)

Comprimento do bloco: $n = 24$

Nº de bits da mensagem: $k = 12$

Nº de bits de paridade: $n - k = 12$

Distância mínima: $d_{\min} = 8$

Nº de erros corrigíveis: $t = 3$

BCH

Comprimento do bloco: $n = 2^m - 1$ $\quad m \geq 3$

Nº de bits da mensagem: $k \geq n - mt$

Distância mínima: $d_{\min} \geq 2t + 1$

Nº de erros corrigíveis: $t < \dfrac{2^m - 1}{2}$

Códigos de comprimento máximo (códigos PN)

Comprimento do bloco: $n = 2^m - 1$ $\quad m \geq 3$

Nº de bits da mensagem: $k = m$

Distância mínima: $d_{\min} = 2^{m-1}$

$$g(x) = \frac{x^n + 1}{h(x)}$$

h(x) — polinómio primitivo de grau *m*

Reed-Solomon

Código BCH não binário

Comprimento do bloco: $n = 2^m - 1$ símbolos

Nº de bits/símbolo: m

Nº de símbolos da mensagem: k símbolos

Nº de símbolos de paridade: $n - k = 2t$ símbolos

Distância mínima: $d_{\min} = 2t + 1 = n - k + 1$ símbolos

(*maximum-distance separable code*, ou MDS)

Nº de símbolos corrigíveis: $t = \dfrac{n-k}{2}$ símbolos

Anexo 5: Factorização de $x^n + 1$

A Tabela seguinte apresenta os factores de $x^n + 1$, para $n \leq 63$ e $n = 127$.

Notas:

- Na Tabela só estão valores ímpares de *n* porque em Álgebra binária $x^{2m} + 1 = \left(x^m + 1\right)^2$.
- Não há entradas para $n = 3,5,11,13,19,29,37,53,59,61$ porque para estes valores a factorização de $x^n + 1$ é simplesmente

$$x^n + 1 = (x+1)\left(x^{n-1} + x^{n-2} + \ldots + x + 1\right)$$

- Os factores são dados em octal, com o termo de maior grau à esquerda ao converter em binário.

Exemplo: Qual é a factorização de $x^9 + 1$? Da Tabela vemos que

$$x^9 + 1 \Rightarrow 3.7.111 \Rightarrow \underbrace{011}_{x+1} \ \underbrace{111}_{x^2+x+1} \ \underbrace{001\ 001\ 001}_{x^6+x^3+1}$$

$$\Rightarrow x^9 + 1 = (x+1)(x^2 + x + 1)(x^6 + x^3 + 1)$$

n	Factores de $x^n + 1$
7	3.13.15.
9	3.7.111.
15	3.7.23.31.37.
17	3.471.727.
21	3.7.13.15.127.165.
23	3.5343.6165.
25	3.37.4102041.
27	3.7.111.1001001.
31	3.45.51.57.67.73.75.
33	3.7.2251.3043.3777.
35	3.13.15.37.13627.16475.
39	3.7.13617.17075.17777.
41	3.5747175.6647133.
43	3.47771.52225.64213.
45	3.7.23.31.37.111.10011.11001.
47	3.43073357.75667061.
49	3.13.15.10000201.10040001.
51	3.7.433.471.637.661.727.763.
55	3.37.3777.5551347.7164555.
57	3.7.1341035.1735357.1777777.
63	3.7.13.15.103.111.127.133.141.147.155.163.165.
127	3.203.211.217.221.235.247.253.271.277.301.313.323. 325.345.357.361.367.375.

Adaptado de "Digital Transmission Theory", S. Benedetto, E. Biglieri e V. Castellani, Prentice-Hall, 1987.

Anexo 6

Distribuição de pesos

- Código de **Hamming** (n,k) binário

O número de palavras de código de peso i é o coeficiente de z^i no *polinómio enumerador de pesos*

$$A(z) = \frac{1}{n+1}\left[(1+z)^n + n(1+z)^{\frac{n-1}{2}}(1-z)^{\frac{n+1}{2}}\right]$$

- Código de **Hamming** binário **aumentado**

Acrescenta-se um bit de paridade a cada palavra de código do código de Hamming *normal* $\Rightarrow d_{\min} = 4$, $n = 2^m$, $m \geq 2$, $k = n - m + 1$.

O *polinómio enumerador de pesos* é

$$A(z) = \frac{1}{2n}\left[(1+z)^n + (1-z)^n + 2(n-1)(1-z^2)^{\frac{n}{2}}\right]$$

- Códigos de **Golay**

	(23,12) $d_{\min} = 7$	**(24,12)** $d_{\min} = 8$
Peso	Nº de palavras de código	
0	1	1
7	253	0
8	506	759
11	1288	0
12	1288	2576
15	506	0
16	253	759
23	1	0
24	0	1

- Código de **Reed-Solomon**

O número de palavras de código de peso i é dado por

$$A_i = \binom{n}{i}\sum_{j=0}^{i-2t-1}(-1)^j\binom{i}{j}\left[(n+1)^{i-2t-j} - 1\right]$$

Anexo 7 Fórmulas

ARQ

Taxa de transferência: $R'_c = \frac{k}{n\overline{m}}$

GBN: $R'_c = \frac{k}{n}\frac{1-p_R}{1-p_R+Np_R} \le \frac{k}{n}\frac{1-p_R}{1-p_R+\frac{2t_d r}{k}p_R}$

SR: $R'_c = \frac{k}{n}(1-p_R)$

SW: $R'_c = \frac{k}{n}\frac{1-p_R}{1+\frac{D}{T_w}} \le \frac{k}{n}\frac{1-p_R}{1+\frac{2t_d r}{k}}$

FEC

- Relação entre a distância mínima e o número de erros corrigíveis por palavra de código: $d_{\min} \ge 2t+1$
- Relação entre a distância mínima e o número de erros detectáveis por palavra de código: $d_{\min} \ge l+1$
- Limite de Singleton: $d_{\min} \le n-k+1$
- Limite de Plotkin: $d_{\min} \le n.2^{k-1}/(2^k-1)$
- Limite de Hamming: $2^{n-k} \ge \sum_{i=0}^{t}\binom{n}{i}$
- Nº de padrões de erro detectáveis num código de blocos (n, k): $2^n - 2^k$
- Nº de padrões de erro corrigíveis num código de blocos (n, k): 2^{n-k}
- Probabilidade de i erros numa palavra de n bits: $P(i,n) = \binom{n}{i}p^i(1-p)^{n-i} \approx \binom{n}{i}p^i \quad p << 1$
- Probabilidade de erro numa palavra de k bits não codificada: $P_e = 1-P(0,k)$
- Majorante da probabilidade de erro numa palavra codificada: $P_{enc} \le \sum_{i=t+1}^{n} P(i,n) = \sum_{i=t+1}^{n}\binom{n}{i}p^i(1-p)^{n-i}$
- Probabilidade de erro numa palavra codificada: $P_{enc} = 1-\sum_{i=0}^{n}\alpha_i p^i(1-p)^{n-i}$
- Probabilidade de erro numa palavra codificada (código perfeito): $P_{enc} = \sum_{i=t+1}^{n} P(i,n) = \sum_{i=t+1}^{n}\binom{n}{i}p^i(1-p)^{n-i}$
- Probabilidade de erro não detectado: $P_{end} = \sum_{i=1}^{n} A_i p^i(1-p)^{n-i}$

Significado dos parâmetros

$\overline{m}$	–	número médio de palavras transmitidas por palavra aceite, em ARQ.	n	–	tamanho, em bits, da palavra de código.
A_i	–	distribuição de pesos	p	–	probabilidade de erro de transição num canal binário simétrico (BSC)
α_i	–	número de *coset leaders* de peso i.	p_R	–	probabilidade de retransmissão de palavras em ARQ
D	–	intervalo de tempo entre duas palavras enviadas (*round trip delay*), em ARQ SW	r	–	taxa de transmissão da mensagem, em bits/s
k	–	tamanho, em bits, da palavra de mensagem.	t	–	número de erros corrigíveis por palavra.
l	–	número de erros detectáveis por palavra de código.	t_d	–	tempo de propagação do emissor ao receptor, em ARQ.
N	–	atraso em símbolos no método Go-Back-N.	T_w	–	duração de cada palavra em ARQ SW