Capítulo 9

Códigos para Control de Erros

Este capítulo trata o problema da codificação da informação para a transmissão, designada de codificação do canal, cuja finalidade é a de permitir controlar os erros de transmissão em sistemas de telecomunicações não-fiáveis ou ruidosos. Segundo o Teorema de Shannon da codificação do canal, existe um código para a informação que possibilita a sua transmissão desde a fonte até ao destino com um número de erros arbitráriamente pequeno.

Neste capítulo estabelecem-se as bases matemáticas que permitem a construção de alguns desses códigos e abordam-se os métodos mais utilizados quer na detecção quer na correcção de erros, conjuntamente designados por métodos de *control* de erros. Considera-se apenas o caso da transmissão digital binária.

9.1 Tipos de erros

Há fundamentalmente dois tipos de ruído que afectam as comunicações digitais: o ruído branco e *gaussiano* e o ruído *impulsivo*. E para enfrentar cada um destes tipos de ruído são utilizados códigos com estruturas distintas.

Os erros de transmissão causados pelo ruído branco e *gaussiano* são tais que a ocorrência de erro num determinado dígito (binário) não afecta os dígitos subsequentes, isto é, as ocorrências de erros são estatísticamente independentes. Neste caso o canal discreto pode ser modelado pelo canal binário simétrico. Os erros de transmissão devidos ao ruído branco e gaussiano são designados de erros aleatórios.

A presença de ruído *impulsivo* é caracterizada por longos intervalos de tempo em que os dígitos não são corrompidos, intercalados por *mólhos*[1] de dígitos corrompidos. O ruído deste tipo, normalmante de grande energia, quando ocorre, afecta um conjunto (mólho) de dígitos seguidos, ou seja, as ocorrências de erros não são estatísticamente independentes.

As técnicas de control de erros destinadas a lidar com o primeiro tipo de erros utilizam os designados *códigos de correcção de erros aleatórios*[2] e com o segundo, *códigos de correcção de erros aos mólhos*[3]. Neste capítulo abordaremos apenas os códigos de correcção de erros aleatórios cuja base matemática é semelhante à dos destinados à correcção de erros aos mólhos visto que nos interessa apenas compreender os fundamentos desta teoria.

9.2 Tipos de códigos

Os códigos para control de erros são básicamente de dois tipos: *códigos de bloco* e *códigos convolucionais*.

Nos códigos de bloco, cada conjunto de k dígitos de informação é acompanhado de $n-k$ de dígitos redundantes, chamados dígitos de *verificação de paridade*, calculados a partir dos dígitos de informação e formando um *bloco* de tamanho fixo, de n dígitos, designado por *palavra de código*.

Nos códigos convolucionais não há dígitos de verificação distintos dos dígitos de informação. A sequência dos dígitos de informação dá origem a uma outra sequência de dígitos dependentes entre si de uma forma também calculada de certa maneira (por convolução).

Embora seja possivel tratar matemáticamente ambos os tipos de códigos numa teoria unificada, esta seria desnecessáriamente complexa para este curso dificilmente permitindo uma visualização imediata das realizões práticas. Neste capítulo abordaremos apenas os códigos de bloco visto serem os mais utilizados nos protocolos de comunicação de dados.

[1] *bursts* na terminologia inglesa.
[2] *random error correcting codes*
[3] *burst error correcting codes*

9.3 Códigos Lineares de Bloco

Os códigos de bloco mais usuais são os lineares. Cada bloco de k dígitos da mensagem (dígitos de informação) são codificados num bloco de $n > k$ dígitos pela adição de $n - k$ dígitos de verificação calculados a partir dos k dígitos da mensagem. A figura 9.1 ilustra a entrada e a saída do codificador do canal. Cada bloco de n dígitos à saida do codificador é uma

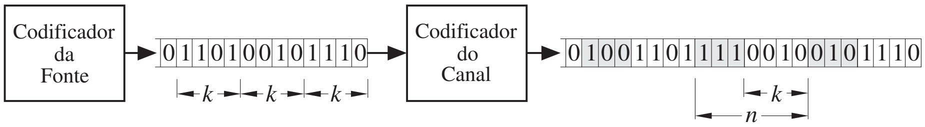

Figura 9.1: Codificador de bloco do canal

palavra de código. Os códigos nos quais os dígitos da mensagem aparecem destacados num sub-bloco conexo da palavra de código são chamados códigos *sistemáticos* ou *separáveis*. Se cada uma das 2^k diferentes palavras de código fôr obtida por combinação *linear* de k palavras de código independentes então o código é chamado de *código linear de bloco*.

Um bloco de mensagem (de dígitos de informação, a que chamaremos dígitos de *dados*) será um *tuplo* $D = (d_0 \ d_1 \ d_2 \ \ldots \ d_{k-1})$ com $d_j \in \{0, 1\}$. Haverá portanto 2^k blocos de mensagem distintos. Cada bloco de mensagem é transformado numa palavra de código representada pelo *tuplo* $C = (c_0 \ c_1 \ c_2 \ \ldots \ c_{n-1})$ com $c_j \in \{0, 1\}$ e haverá portanto apenas 2^k palavras de código distintas, uma para cada bloco de mensagem. As restantes $2^n - 2^k$ palavras que se podem formar com os n dígitos binários não fazem parte do dicionário do código e, se porventura forem recebidas no destino, denunciam a ocorrência de erro.

Estes códigos são designados de *códigos*-(n, k) ou, abreviadamente, $C(n, k)$. O seu *rendimento*, ρ, é igual a

$$\rho = \frac{k}{n}$$

pois aumentam o número de dígitos originalmente produzidos pelo codificador da fonte de um factor igual a $\frac{n}{k}$.

Num código sistemático linear de bloco os primeiros k dígitos do código são os dígitos de informação, isto é, $c_j = d_j$ para $j = 0, 2, \ldots, k-1$ e os últimos $n-k$ dígitos são os de verificação.

9.3.1 Distância de Hamming

Definição 9.1 *Distância de Hamming entre duas palavras de um código de bloco, $d(C_i, C_j)$, é o número de posições em que as duas palavras, C_i e C_j, diferem.*

A distância de Hamming entre as palavras (1 1 1 0 0) e (1 0 1 0 1), por exêmplo, é igual a 2. Elas diferem apenas na segunda e na quinta posições. Duas palavras idênticas estão à distância zero uma da outra e duas palavras distintas estarão a uma distância igual ou superior a uma unidade. O conceito de distância de Hamming é passível de interpretação geométrica semelhante à da distância Euclideana (usual) entre dois pontos do espaço físico. Com efeito, pode estabelecer-se uma correspondência biunívoca entre as 2^n palavras distintas de n dígitos binários e os 2^n vértices de um hipercubo unitário no espaço de n dimensões como se mostra na figura 9.2 para vários valores de n. Cada palavra de n dígitos define a coor-

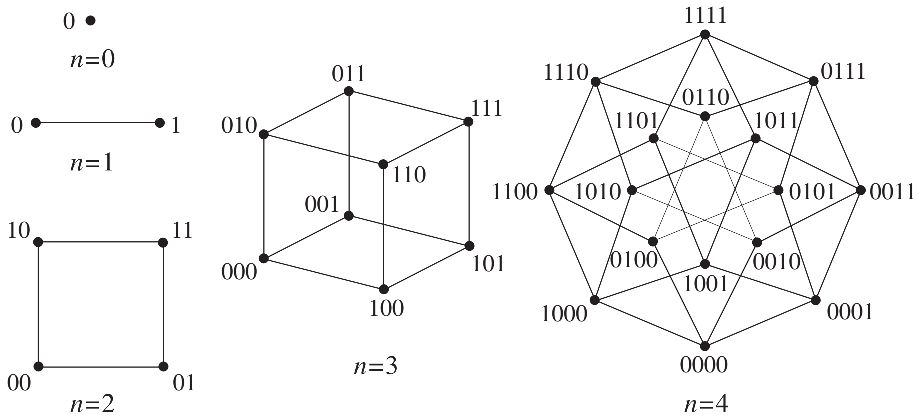

Figura 9.2: Palavras de código como coordenadas dos vértices do hipercubo unitário a n dimensões

denada de um vértice do cubo a n dimensões. Dois vértices adjacentes, isto é, situados nos extremos de uma mesma aresta de comprimento igual a 1 unidade, estão distanciados de uma unidade e as suas coordenadas diferem apenas numa única posição dimensional. Se nos movermos de um vértice para outro, ao longo de d arestas, então a distância entre esses dois vértices será igual a d desde que não haja um número menor de arestas que liguem esses dois vértices.

Definição 9.2 *Distância mínima de um código de bloco, d_{min}, é a menor das distâncias de Hamming entre quaisquer duas palavras desse código.*

Qualquer código com um número constante de dígitos será um sub-conjunto das 2^n palavras disponíveis e poderá ser representado mapeando-o no cubo unitário a n dimensões. As propriedades métricas das palavras de um código determinam a sua capacidade de control de erros.

9.3.2 Capacidade de control de erros dos códigos de bloco

Se se construir um código de tal forma que cada palavra esteja a uma distância igual ou superior a 2 de qualquer outra palavra de código então pode sempre detectar-se um único erro em qualquer palavra transmitida. Com efeito, todos os casos em que ocorra um único erro darão lugar a uma nova palavra à distância 1 da palavra correcta original e à distância *pelo menos* 1 de alguma outra palavra de código. Assim, a palavra corrompida será diferente (e poderá ser identificada) de qualquer palavra válida. Não será possível corrigir o erro dado que não é possível localizar a posição do dígito errado visto que a mesma palavra pode ter resultado da corrupção de várias palavras de código (válidas) em diferentes posições dos dígitos. Da mesma forma, se ocorrerem dois erros na mesma palavra o resultado pode dar origem a uma palavra válida e a corrupção não será detectada. Torna-se evidente que para detectar a ocorrência de até e erros numa palavra de código a distância mínima do código deve ser pelo menos igual a $e + 1$. De facto, e erros simultâneos darão origem a uma nova palavra à distância e da palavra original, mas também pelo menos à distância 1 de qualquer outra palavra de código.

Considere-se agora um código com distância mínima igual a 3. Um único erro em qualquer posição de uma palavra de código produzirá uma nova palavra à distância 1 da palavra original e pelo menos à distância 2 de qualquer outra palavra de código, portanto, mais perto daquela do que destas.

Se se adoptar o princípio da máxima verosimilhança[4], é mais provável que

[4]Num bloco de n dígitos, a probabilidade de ocorrerem apenas i erros é maior do

a palavra original fosse a que está à distância 1, na qual terá ocorrido um único erro, do que qualquer outra à distância 2, em que teriam occorido dois erros.

Ao tomar esta decisão de correcção diz-se que o descodificador do canal corrigiu um erro simples. Contudo, pode ter-se cometido um erro de descodificação caso tenham realmente ocorrido dois erros na palavra. Nesta situação a palavra corrompida estará à distância 1 de uma outra palavra de código e à distância 2 da palavra original.

Assim, a probabilidade de erro por dígito de informação entregue ao destino, P_E, será menor do que a probabilidade de erro por dígito no canal, P_e, porque se elimina, pelo menos, a possibilidade de erros simples por palavra. Na secção 9.5 abordar-se-á esta questão para o caso dos sistemas que utilizam o mecanismo da correcção de erros progressiva.

Em conclusão, se se desejar corrigir até e erros por bloco (palavra de código) deve assegurar-se que o código utilizado possui uma distância mínima igual a $2e+1$ pois, deste modo, e erros darão lugar a uma palavra à distância e da palavra correcta original e a pelo menos $e+1$ de qualquer outra palavra válida (de código).

Seja d_{min} a distância mínima de um código,

Para detectar até e_d erros:	$d_{min} = e_d + 1$
Para corrigir até e_c erros:	$d_{min} = 2e_c + 1$

Existe sempre um compromisso entre a capacidade de correcção de erros de um código e a sua capacidade de detecção de erros dado que um código que corrige e_c erros pode ser empregue, alternativamente, como um código detector de $e_d = 2e_c$ erros.

Definição 9.3 *Peso de uma palavra C_i de um código de bloco, $p(C_i)$, é o número de dígitos 1 que a palavra C_i contém.*

Definição 9.4 *Peso mínimo de um código de bloco, $[p(C_i)]_{min}$ é o peso da palavra de menor peso desse código, exceptuando a palavra de peso zero.*

Como consequência da linearidade pode determinar-se a distância mínima de um código conhecido o seu peso mínimo:

que a probabilidade de ocorrerem $i+1$ erros $(0 \leq i \leq n-1)$, i.é., $P(0 \text{ erros}) \gg P(1 \text{ erro}) \gg P(2 \text{ erros}) \gg \ldots \gg P(n \text{ erros})$ quando $P(1 \text{ erro}) = P_e \ll 1$.

Teorema 9.1 — Distância mínima
A distância mínima de um código de bloco é igual ao seu peso mínimo.

Demonstração
Seja $p(C_i)$ o peso da palavra C_i. A distância de Hamming entre quaisquer duas palavras do código, $C_i = (c_{0i}\ c_{1i}\ \dots\ c_{n-1\,i})$ e $C_j = (c_{0j}\ c_{1j}\ \dots\ c_{n-1\,j})$, será então

$$d(C_i, C_j) = p(C_i + C_j)$$

dado que $c_{ki} + c_{kj} = 1$ sempre que $c_{ki} \neq c_{kj}$. Portanto, a distância entre C_i e C_j é igual ao peso de uma outra palavra de código $C_z = C_i + C_j$. Mas se $C_j = (0\,0\ \dots\ 0)$ então $C_i + C_j = C_i$ e por conseguinte

$$d_{min} = [p(C_i)]_{min} \quad \text{com} \quad C_i \neq (0\,0\ \dots\ 0)$$

c.q.d.

9.3.3 Códigos de Hamming

Para cada valor de $n-k$ existe um código binário com $n-k$ dígitos de verificação e comprimento da palavra de código

$$\boxed{n = 2^{n-k} - 1}$$

A cardinalidade do dicionário é 2^k e o código é capaz de corrigir um erro simples em qualquer posição da palavra recebida. O rendimento ou taxa do código é

$$\rho = \frac{k}{n} = \frac{2^{n-k} - 1 - (n-k)}{2^{n-k} - 1} = 1 - \frac{n-k}{2^{n-k} - 1}$$

Um código que obedeça a estas condições é designado de código de Hamming.

Os códigos de Hamming são códigos correctores de erros simples ou detectores de erros duplos de elevado rendimento. A ocorrência de dois ou mais erros não pode, em geral ser corrigida e a ocorrência de três ou mais erros não pode, em geral ser detectada.

9.4 Códigos Cíclicos Binários

Os *códigos cíclicos* constituem uma sub-classe dos códigos lineares de bloco que é atractiva porque tanto a codificação, como a detecção ou a correcção dos erros no descodificador, podem realizar-se fácilmente com registos de deslocamento realimentados lineares[5] devido à simplicidade da estrutura matemática que possuem. Estes códigos são representados por *polinómios* no *corpo finito* binário.

O corpo binário é o corpo finito constituido por um conjunto de dois elementos, por exêmplo, o conjunto $\mathcal{A} = \{0, 1\}$, onde estão definidas as operações de adição, $+$, e multiplicação, $\cdot$, obedecendo às seguintes regras:

$+$	0	1
0	0	1
1	1	0

$\cdot$	0	1
0	0	0
1	0	1

em que o elemento 0 é a identidade aditiva (o elemento zero do corpo) e o elemento 1 é a identidade multiplicativa (o elemento unidade do corpo) e onde se pode verificar que cada elemento é o seu próprio inverso aditivo (simétrico), isto é, $y = -y$, $\forall y \in \mathcal{A}$, e cada elemento diferente de zero é o seu próprio inverso multiplicativo, ou seja, $y = y^{-1}$, $\forall y \in \mathcal{A}_{-\{0\}}$. Por outras palavras, a multiplicação é a multiplicação usual e a adição é a adição usual tomada módulo-2, ou seja, é a operação *ou-exclusivo*.

9.4.1 Estrutura algébrica dos códigos cíclicos

Definição 9.5 *Um <u>código</u> linear de bloco $C(n, k)$ é <u>cíclico</u> se possuir a seguinte propriedade:*

> *Se o tuplo $C = (c_0, c_1, c_2, \ldots, c_{n-1})$ fôr uma palavra de código então o tuplo $C^{(1)} = (c_{n-1}, c_0, c_1, \ldots, c_{n-2})$ obtido por deslocação cíclica direita de uma posição de C também é uma palavra de código.*

Desta definição resulta que

$$C^{(i)} = (c_{n-i}, c_{n-i+1}, \ldots, c_{n-1}, c_0, c_1, \ldots, c_{n-i-1})$$

[5] *linear feedback shift registers* ou *lfsr*

também é uma palavra de código. Esta propriedade dos códigos cíclicos permite considerar os elementos de cada palavra de código como coeficientes de um polinómio de grau $n-1$ numa variável real x, isto é,

$$C \rightarrow C(x) = c_0 + c_1 x + c_2 x^2 + \ldots + c_{n-1} x^{n-1}$$

em que os coeficientes $c_j \in \{0, 1\}$ do polinómio e as operações $+$ e $\cdot$, são, respectivamente, os elementos e as operações do corpo binário.

O polinómio de código $C^{(i)}(x)$ correspondente à palavra de código $C^{(i)}$ é

$$C^{(i)}(x) = c_{n-i} + c_{n-i+1} x + \ldots + c_{n-1} x^{i-1} + c_0 x^i + c_1 x^{i+1} + \ldots + c_{n-i-1} x^{n-1}$$

e pode mostrar-se que o polinómio $C^{(i)}(x)$ é o resto da divisão de $x^i\, C(x)$ por $x^n + 1$, isto é,

$$x^i\, C(x) = q(x)\,(x^n + 1) + C^{(i)}(x)$$

(Resolver o problema 9.1).

Enunciemos e demonstremos agora um importante teorema que nos permite gerar códigos cíclicos.

Teorema 9.2 — Geração de códigos cíclicos
Seja $g(x)$ um polinómio de grau $n-k$, que divide o polinómio $x^n + 1$. Então $g(x)$ gera um código cíclico (n, k) no qual o polinómio de código $C(x)$ correspondente ao polinómio de informação (de dados) $D(x) = d_0 + d_1 x + d_2 x^2 + \ldots + d_{k-1} x^{k-1}$ é gerado por

$$C(x) \;=\; D(x) \cdot g(x) \tag{9.1}$$

Demonstração
Considerem-se os k polinómios $g(x)$, $x\, g(x)$, $x^2\, g(x)$, …, $x^{k-1}\, g(x)$ de grau menor ou igual a $n-1$. Qualquer combinação linear destes polinómios é um polinómio de grau menor ou igual a $n-1$ e é múltiplo de $g(x)$. Em particular, a seguinte combinação linear está nessas condições

$$\begin{aligned} C(x) &= d_0\, g(x) + d_1\, x\, g(x) + d_2\, x^2\, g(x) + \ldots + d_{k-1}\, x^{k-1}\, g(x) \\ &= D(x) \cdot g(x) \end{aligned}$$

Existe um total de 2^k polinómios $C(x)$ distintos correspondendo aos 2^k diferentes tuplos $D = (d_0, d_1, d_2, \ldots, d_{k-1})$ e os polinómios de código $C(x)$

correspondentes a esses 2^k polinómios constituem um código linear (n, k). Para mostrar que o código resultante é cíclico, considere-se que $C(x) = c_0 + c_1x + c_2x^2 + \ldots + c_{n-1}x^{n-1}$ é um polinómio deste código. O produto de x por $C(x)$ dá

$$\begin{aligned} x \cdot C(x) &= c_0x + c_1x^2 + c_2x^3 + \ldots + c_{n-2}x^{n-1} + c_{n-1}x^n \\ &= c_{n-1}(x^n + 1) + (c_{n-1} + c_0x + c_1x^2 + \ldots + c_{n-2}x^{n-1}) \\ &= c_{n-1}(x^n + 1) + C^{(1)}(x) \end{aligned}$$

onde se fez $c_{n-1} + c_{n-1} = 0$. $C^{(1)}(x)$ obtem-se por deslocação cíclica direita de $C(x)$. Dado que $x \cdot C(x)$ e $x^n + 1$ são ambos divisiveis por $g(x)$, $C^{(1)}(x)$ também é divisivel por $g(x)$. Assim, sendo $C^{(1)}(x)$ um múltiplo de $g(x)$, pode ser expresso como uma combinação linear dos polinómios $g(x)$, $x\,g(x)$, $x^2\,g(x)$, …, $x^{k-1}\,g(x)$, o que significa que $C^{(1)}(x)$ é também um polinómio de código. Em conclusão, pela definição de um código cíclico, o código linear gerado por $g(x)$, $x\,g(x)$, …, $x^{k-1}\,g(x)$ é um código cíclico (n, k). c.q.d..

O polinómio $g(x)$ é chamado *polinómio gerador* do código cíclico. Pode mostrar-se que $g(x)$ é um polinómio irredutível no corpo binário, isto é, um polinómio primo (não factorizável) pelo que será sempre $g_{n-k-1} = g_0 = 1$.

9.4.2 Geração de códigos cíclicos sistemáticos

Dado o polinómio gerador $g(x)$ de um código cíclico, as respectivas palavras de código podem ser construidas de uma forma sistemática separando os dígitos de verificação de paridade dos dígitos da mensagem da seguinte maneira

$$C = (\underbrace{r_0, r_1, r_2, \ldots, r_{n-k-1}}_{\substack{n-k \text{ dígitos} \\ \text{de verificação} \\ \text{de paridade}}}, \underbrace{d_0, d_1, d_2, \ldots, d_{k-1}}_{\substack{k \text{ dígitos} \\ \text{da mensagem}}}) \tag{9.2}$$

onde $r(x) = r_0 + r_1x + r_2x^2 + \ldots + r_{n-k-1}x^{n-k-1}$ é o *polinómio de verificação de paridade* correspondente ao polinómio da mensagem (a que chamaremos polinómio dos dados), $D(x) = d_0 + d_1x + d_2x^2 + \ldots + d_{k-1}x^{k-1}$. É simples verificar que o polinómio $r(x)$ é o resto da divisão de $x^{n-k}D(x)$ por $g(x)$, ou seja

$$x^{n-k}D(x) = q(x) \cdot g(x) + r(x) \tag{9.3}$$

onde $q(x)$ e $r(x)$ são, respectivamente, o quociente e o resto daquela divisão e que, por conseguinte, o polinómio de código pode ser escrito sob a forma

$$C(x) = r(x) + x^{n-k}D(x) \tag{9.4}$$

que constitui a representação polinomial do código cíclico sistemático.

Exêmplo 9.1 *Seja $g(x) = 1 + x + x^3$ o polinómio gerador de um código cíclico $(7, 4)$. Determinar as 16 palavras deste código, das seguintes maneiras:*
a)calculando os polinómios de código através de $C(x) = D(x) \cdot g(x)$
b)calculando os polinómios de código na forma sistemática

Resolução

a) Considere-se uma sequência de dados qualquer, por exêmplo $D = (d_0, d_1, d_2, d_3) = (1\,0\,1\,0)$, a que corresponde o polinómio $D(x) = 1 + x^2$. O correspondente polinómio de código é

$$\begin{aligned} C(x) &= D(x) \cdot g(x) \\ &= (1 + x^2) \cdot (1 + x + x^3) = 1 + x + x^3 + x^2 + x^3 + x^5 \\ &= 1 + x + x^2 + x^5 \end{aligned}$$

dado que $x^3 + x^3 = (1 + 1) \cdot x^3 = 0 \cdot x^3 = 0$. Portanto a palavra de código é $C = (1\,1\,1\,0\,0\,1\,0)$. Podem obter-se outras palavras do código por deslocação cíclica desta. A segunda coluna da tabela 9.1 lista o código completo assim calculado.

b) Na forma sistemática os três primeiros dígitos são os de verificação e os últimos quatro são os da mensagem. Os dígitos de verificação são os coeficientes do polinómio $r(x)$ que é o resto da divisão de $x^{n-k}D(x)$ por $g(x)$, isto é,

$$\frac{x^{n-k}D(x)}{g(x)} = q(x) + \frac{r(x)}{g(x)}$$

Considere-se uma sequência qualquer de mensagem, por exêmplo $D = (1\,1\,1\,0)$, a que corresponde $D(x) = 1 + x^2 + x^3$. Como $n - k = 7 - 4 = 3$, tem-se $x^3D(x) = x^3 + x^4 + x^5$ e executando a divisão polinomial:

$$\begin{array}{lll|l} x^5+x^4+x^3 & & & x^3+x+1 \\ \cline{4-4} \underline{x^5+\quad\; x^3+x^2} & & & x^2+x \\ 0+x^4+0+x^2 & & & \\ \quad\;\underline{x^4+\quad\; x^2+x} & & & \\ \quad\; 0+\quad\;\; 0+x & = r(x) & & \end{array}$$

donde resulta $q(x) = x^2 + x$ *e* $r(x) = x$ *pelo que* $C = (\underbrace{0\,1\,0}_{r(x)}\,\underbrace{1\,1\,1\,0}_{D(x)})$. *De igual forma, podem obter-se outras palavras do código por rotação cíclica a partir desta. A terceira coluna da tabela 9.1 lista o código sistemático completo assim calculado e na quarta coluna inidicam-se os pesos de cada palavra deste código.*

Tabela 9.1: Código cíclico (7,4) gerado por $g(x) = 1 + x + x^3$

Informação $D(x)$				Código criptográfico $C(x) = D(x) \cdot g(x)$							Código sistemático $C(x) = r(x) + x^{n-k} D(x)$							Peso $p(C_i)$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	1	0	1	1	0	1	0	0	0	1	3
0	0	1	0	0	0	1	1	0	1	0	1	1	1	0	0	1	0	4
0	0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1	3
0	1	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	3
0	1	0	1	0	1	1	1	0	0	1	1	1	0	0	1	0	1	4
0	1	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1	0	3
0	1	1	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	4
1	0	0	0	1	1	0	1	0	0	0	1	1	0	1	0	0	0	3
1	0	0	1	1	1	0	0	1	0	1	0	1	1	1	0	0	1	4
1	0	1	0	1	1	1	0	0	1	0	0	0	1	1	0	1	0	3
1	0	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1	1	4
1	1	0	0	1	0	1	1	1	0	0	1	0	1	1	1	0	0	4
1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	1	0	1	3
1	1	1	0	1	0	0	0	1	1	0	0	1	0	1	1	1	0	4
1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	7

Quatro observações são devidas relativamente aos códigos deste exêmplo e que também se aplicam à generalidade dos casos:

(1) O conjunto das palavras de código é o mesmo em ambas as codificações embora possam não corresponder às mesmas palavras da mensagem.

(2) O *ciclo* obtido pelas sucessivas deslocações cíclicas a partir de uma palavra de código pode não gerar *todas* as palavras de código. Nestes

casos haverá que repetir o procedimento a partir de uma palavra de mensagem cujo código não esteja contido num ciclo já gerado.

(3) No código gerado através da equação 9.1 as posições dos dígitos de verificação e de informação não são localizáveis constituindo o que se designa por *código criptográfico*.

(4) O peso mínimo dos códigos é 3 sendo a sua distância mínima também $d_{min} = 3$. Portanto, o código (7,4) é corrector de erros simples ($e_c = 1$) ou detector de erros duplos ($e_d = 2$). Trata-se de um código de Hamming pois verifica a relação $n = 2^{n-k} - 1$.

9.4.3 Codificação com registos de deslocamento

A maior vantagem que os códigos cíclicos oferecem é a simplicidade de realização prática dos codificadores e dos descodificadores bem como do control de erros.

As operações de codificação descritas pelas equações 9.1 e 9.4 envolvem respectivamente a multiplicação e a divisão de um polinómio pelo polinómio gerador $g(x)$. Em particular, no segundo caso, o polinómio de verificação de paridade, $r(x)$, é o resto daquela divisão. Para realizar a divisão utilizam-se *registos de deslocamento com realimentação*.

Considere-se a divisão de um polinómio *dividendo* qualquer $a(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$ de grau n pelo polinómio fixo *divisor* $b(x) = b_0 + b_1x + b_2x^2 + \ldots + b_mx^m$ de grau m. O resultado será um polinómio *quociente*, $q(x)$, de grau $n - m$ e possivelmente um polinómio *resto*, $r(x)$, de grau menor que m

$$\frac{a(x)}{b(x)} = q(x) + \frac{r(x)}{b(x)}$$

O primeiro termo de $q(x)$ terá a forma $q_{n-m}x^{n-m}$ e pode deduzir-se q_{n-m} pelo facto de se ter de fazer $(q_{n-m}x^{n-m}) \cdot (b_mx^m)$ igual ao inverso aditivo do termo de maior ordem de $a(x)$, ou seja, a_nx^n. Assim, $q_{n-m} = a_nb_m^{-1}$. Na divisão executada manualmente continuar-se-ia, somando $a_nb_m^{-1}x^{n-m} \cdot b(x)$ a $a(x)$ eliminando-se o termo a_nx^n (e possivelmente vários outros). Repetir-se-ia o procedimento deduzindo o termo seguinte de $q(x)$ que seria $(a_{n-1} - a_nb_{m-1}b_m^{-1}) \cdot b_m^{-1}x^{n-m-1}$, ou seja,

$$\begin{array}{l|l} a_nx^n + a_{n-1}x^{n-1} + \ldots + a_1x + a_0 \quad & b_mx^m + b_{m-1}x^{m-1} + \ldots + b_1x + b_0 \\ \cline{2-2} & a_nb_m^{-1}x^{n-m} + (a_{n-1} - a_nb_{m-1}b_m^{-1})x^{n-m-1} + \ldots \end{array}$$

$$\begin{array}{l}
\underline{a_n x^n + a_n b_{m-1} b_m^{-1} x^{n-1} + \ldots + a_n b_1 b_m^{-1} x^{n-m+1} + a_n b_0 b_m^{-1} x^{n-m}} \\
(a_{n-1} - a_n b_{m-1} b_m^{-1}) x^{n-1} + \ldots \\
\underline{(a_{n-1} - a_n b_{m-1} b_m^{-1}) x^{n-1} + (a_{n-1} - a_n b_{m-1} b_m^{-1}) b_m^{-1} b_{m-1} x^{n-2} + \ldots} \\
(a_{n-2} - (a_n b_{m-2} + a_{n-1} b_{m-1}) b_m^{-1} + a_n b_{m-1}^2 b_m^{-2}) + \ldots \\
\qquad\qquad\qquad\qquad\text{etc.}
\end{array}$$

para cada novo coeficiente do quociente q_i o polinómio $q_i b(x)$ é subtraido do dividendo e o procedimento repetido até não se encontrar novos termos para o quociente. Neste ponto restarão ainda alguns termos no dividendo que constituem o polinómio resto final, $r(x)$.

O circuito representado na figura 9.3 mecaniza esta operação de divisão polinomial. Nessa figura, as caixas quadradas representam circuitos multi-estáveis tipo-D interligados por adicionadores constituindo um registo de

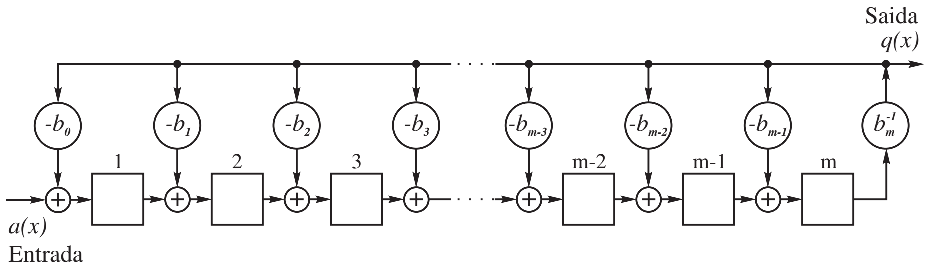

Figura 9.3: Circuito de divisão de polinómios

deslocamento. Os circulos circunscrevendo os valores dos coeficientes do polinómio divisor são multiplicadores por escalar na álgebra considerada. A saída do registo de deslocamanto, após multiplicação pela constante b_m^{-1} e, em cada estágio, pelo valor dum coeficiente do dividendo *realimenta* o registo nesse estágio adicionando-se ao valor da saída do estágio anterior.

No caso binário, que é o que interessa considerar, a adição é efectuada módulo-2, o multiplicador por escalar é uma ligação directa fechada se $b_i = 1$ ou uma ligação aberta se $b_i = 0$ e os multi-estáveis são bi-estáveis (*flip-flops*) tipo-D. A cada ocorrência do pulso de relógio de temporização (que não está representado) dos flip-flops, os valores à entrada destes transitam para as respectivas saidas. O registo é inicialmente colocado a zero. A saida $q(x)$ permanece a zero durante os primeiros m deslocamentos, ao fim dos quais o coeficiente da maior potência do dividendo a_n atinge o último estágio do registo. Aparece então o primeiro coeficiente da saida não-zero, ou seja, $a_n b_m^{-1}$ (notar que no caso binário $b_i^{-1} \equiv b_i$). As ligações de realimentação asseguram que, para cada coeficiente do quociente, q_i, o polinómio $q_i b(x)$ seja subtraido do dividendo que é mantido no registo sob a forma de restos parciais. Após um total de $n+1$ deslocamentos, todo o dividendo entrou no registo, todo o quociente já apareceu à saida e o polinómio resto final encontra-se no registo.

Deve notar-se que a divisão só começa quando o coeficiente da maior potência do dividendo chega ao último flip-flop, ou seja, apenas passados $n-k$ pulsos do relógio de temporização do circuito. Assim sendo e para que a divisão possa começar imediatamente, o polinómio dividendo deve entrar directamente somado à saída do último flip-flop cujo valor é 0 visto que o registo é inicializado a (00...0) antes do início da divisão.

Pode verificar-se que o mesmo circuito, alimentado à saída como se mostra na figura 9.4, efectua a divisão polinomial. Mas, para formar uma palavra de código sistemático, não é necessário o quociente mas sim o resto da divisão, portanto a saída útil deste circuito é o próprio dividendo seguido do resto final que ficou no registo.

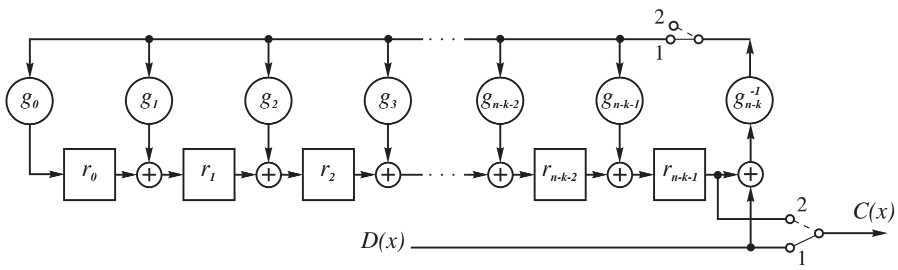

Figura 9.4: Codificador para um código sistemático (n, k)

Esta configuração é pois a mais apropriada para realizar a codificação sistemática pela qual são transmitidos em primeiro lugar os k dígitos da mensagem, seguidos dos $n-k$ dígitos do resto descarregados do registo, por deslocamento, após efectuada a divisão. Durante os k primeiros pulsos do relógio, os comutadores estão na posição 1, e durante os restantes $n-k$ pulsos na posição 2.

Exêmplo 9.2 *Definir um codificador para o código cíclico binário* $(7,4)$ *gerado por* $g(x) = 1+x+x^3$ *e verificar a sua operação utilizando a palavra de mensagem* $(0\,1\,0\,1)$.

Resolução

O codificador para este código está representado na figura 9.5 pois tem-se

$n-k=7-4=3$ *e* $g_0=1$, $g_1=1$, $g_2=0$ *e* $g_3=1$. *Os passos da operação*

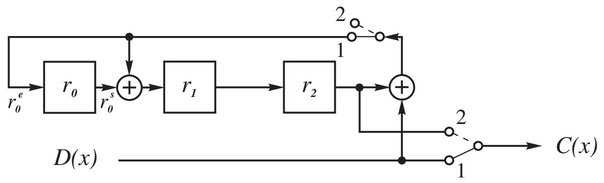

Figura 9.5: Codificador para o código cíclico sistemático $(7,4)$ gerado por $g(x)=1+x+x^3$

de codificação estão descriminados na tabela 9.2 e portanto a palavra de código de $(0\,1\,0\,1)$ *é* $(1\,1\,0\,0\,1\,0\,1)$.

Tabela 9.2: Operação de codificação da mensagem $(0\,1\,0\,1)$

bit de entrada	entrada nos registos				saída dos registos			
$D(x)$	r_0^e	r_1^e	r_2^e		r_0^s	r_1^s	r_2^s	
—	0	0	0		0	0	0	
1	1	1	0	$\rightarrow$	1	1	0	1º deslocamento
0	0	1	1	$\rightarrow$	0	1	1	2º deslocamento
1	0	0	1	$\rightarrow$	0	0	1	3º deslocamento
0	1	1	0	$\rightarrow$	1	1	0	4º deslocamento

9.4.4 O síndroma: detecção e correcção de erros

Quando uma palavra de código C é transmitida através do canal, ela será eventualmente corrompida, chegando ao receptor uma palavra qualquer R. A partir desta o descodificador deverá determinar qual a palavra de código efectivamente transmitida. Para isso divide $R(x)$ pelo polinómio gerador $g(x)$ obtendo um resto $S(x)$ que é designado por *síndroma* de $R(x)$. Se o síndroma fôr zero a palavra recebida pertence ao código e portanto toma-a como correcta. Caso contrário decide que houve erro e poderá ou não tentar corrigi-la. Essa divisão é representada pela equação

$$R(x) = P(x) \cdot g(x) + S(x) \tag{9.5}$$

em que $P(x)$ é o polinómio quociente. O polinómio síndroma $S(x)$ é de grau $n-k-1$ ou inferior. Seja E a sequência de erro que ocorreu no canal. E pode ser representado por um polinómio $E(x)$, desconhecido, de grau $n-1$ dando origem a $R(x)$ da seguinte forma:

$$R(x) = C(x) + E(x) \tag{9.6}$$

Substituindo este valor na equação 9.5 tem-se

$$C(x) + E(x) = P(x) \cdot g(x) + S(x) \tag{9.7}$$

e dado que $C(x) = D(x) \cdot g(x)$ tem-se

$$E(x) = [P(x) + D(x)] \cdot g(x) + S(x) \tag{9.8}$$

donde se verifica que o resto da divisão da sequência de erro pelo polinómio gerador é igual ao síndroma de $R(x)$. Por outras palavras, o síndroma do erro é igual ao síndroma da palavra recebida. Daqui se conclui que o síndroma contém informação àcerca do padrão do erro e pode ser utilizado para corrigir R.

No código (7,4) dos exêmplos anteriores o síndroma é um polinómio de grau $n-k=2$, isto é, tem 3 dígitos, que é o número de dígitos suficiente para *apontar* para *uma* das sete possíveis posições do erro podendo portanto corrigi-lo. Caso tenham ocorrido dois ou mais erros o reduzido número de dígitos do síndroma neste código (7,4) não permite apontá-los a ambos. Apenas indicará que eles ocorreram.

Para a descodificação, é então efectuada uma divisão utilizando-se um circuito semelhante ao da figura 9.3 como se mostra na figura 9.6. Neste circuito existe um registo de deslocamento (*buffer*) de entrada com n flip-flops onde dá entrada a palavra recebida $R(x)$. Simultâneamente, é executada a divisão de $R(x)$ pelo polinómio gerador $g(x)$ no registo realimentado abaixo daquele, que conterá o resto logo que $R(x)$ esteja todo no *buffer*. A soma das posições em erro é $S(x) = r(x)$. Se ocorreu um erro simples então $S(x) = s_0 + s_1 x + ... + s_{n-k-1} x^{n-k+1}$ indica a posição desse erro.

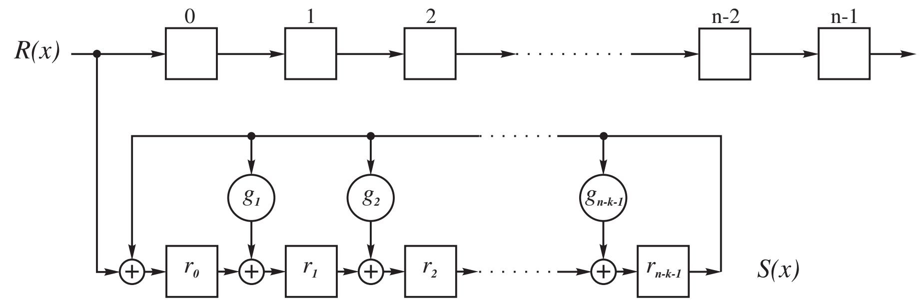

Figura 9.6: Divisão de $R(x)$ por $g(x)$ no descodificador

Caso o descodificador seja utilizado para detecção de erros, bastará calcular o ou-inclusivo dos valores dos registos do circuito divisor para se sinalizar a ocorrência de erro.

Caso seja utilizado para a correcção, numa solução elegante, adiar-se-á a correcção para a altura em que a palavra recebida saia do *buffer* de entrada para ser entregue ao destino. Assim, se o síndroma indica um erro na posição 2^i dever-se-á deixar sair do *buffer* $n-1-i$ dígitos de $R(x)$ e depois corrigir o dígito i quando ele aparecer à saída.

De modo a libertar o registo de divisão para a palavra R que venha logo a seguir, transfere-se $S(x)$ para um circuito divisor idêntico — o registo corrector —, temporizado síncronamente com os restantes, e leva-se o registo anterior a zero. A figura 9.7 ilustra a lógica de um circuito corrector de erros. Pode verificar-se que a função de correcção para o caso de erros simples é, como mostra a figura:

$$f(s_0, s_1, \ldots, s_{n-k-1}) = \overline{\overline{s_0} \vee s_1 \vee s_2 \vee \ldots \vee s_{n-k-1}} \qquad (9.9)$$

9.4.5 Códigos BCH

A concepção óptima de códigos para control de erros envolve a procura de códigos com o menor comprimento possível de bloco, n, para um dado comprimento k do bloco de informação e para uma determinada distância mínima do código, d_{min}. Ou, para um dado comprimento do bloco n e valor do rendimento k/n, descobrir aqueles que possuem a maior distância mínima d_{min}. Isto é, normalmente deseja-se encontrar códigos com a maior capacidade de detecção ou de correcção de erros possível.

Os códigos BCH (Bose-Chaudhuri-Hocquenghem) constituem a sub-classe

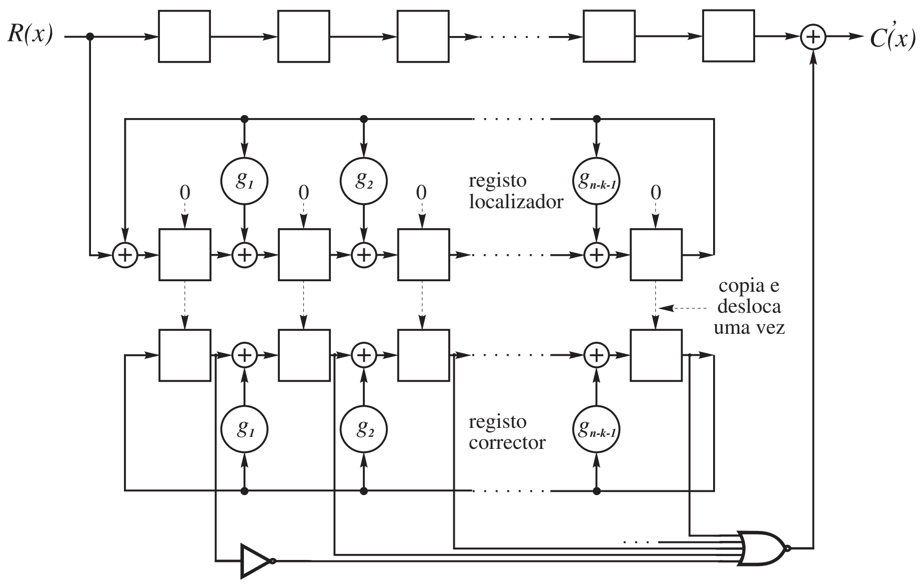

Figura 9.7: Circuito corrector de erros simples

mais poderosa dos códigos cíclicos correctores de erros. A sua análise detalhada, porém, exige tratamentos com álgebras em corpos finitos o que sai fora do âmbito deste curso. Enunciam-se, contudo, algumas propriedades dos códigos BCH que ilustram a sua potencialidade.

Assim, para quaisquer números inteiros positivos m e e_c (com $e_c < 2^{m-1}$) existe um código BCH com os seguintes parâmetros:

comprimento do bloco:	$n = 2^m - 1$
nº de dígitos de verificação:	$n - k \leq m\, e_c$
distância mínima:	$d_{min} \geq 2e_c + 1$

em que e_c é o número de erros que o código é capaz de corrigir.

Segundo o teorema 9.2, qualquer factor de x^n+1 com grau $n-k$ pode servir de polinómio gerador de um código cíclico (n, k) mas não gera necessáriamente um bom código. A tabela 9.3 lista os polinómios geradores de alguns códigos cíclicos que revelaram possuir as melhores características para serem usados como códigos correctores em sistemas de transmissão genéricos.

Tabela 9.3: Alguns bons códigos cíclicos

Tipo	n	k	ρ	d_{min}	$g(x)$
códigos de	7	4	0.57	3	x^3+x+1
Hamming	15	11	0.73	3	x^4+x+1
	31	26	0.84	3	x^5+x^2+1
códigos	15	7	0.46	5	$x^8+x^7+x^6+x^4+1$
BCH	31	21	0.68	5	$x^{10}+x^9+x^8+x^6+x^5+x^3+1$
	63	45	0.71	7	$x^{18}+x^{17}+x^{16}+x^{15}+x^9+x^7+$
					$+x^6+x^3+x^2+x+1$
código	23	12	0.52	7	$x^{11}+x^9+x^7+x^6+x^5+x+1$
Golay					

9.5 Técnicas de Correcção de Erros

9.5.1 Correcção Progressiva e ARQ

Quando um código para control de erros é utilizado directamente como corrector, o processo designa-se por *correcção de erros progressiva — Forward Error Correction (FEC)*. É o único mecanismo utilizável quando só existe um canal num só sentido (comunicação simplex) e a retransmissão é impossível ou quando esta é impraticável. Como exêmplos desta situação podem citar-se as transmissões por difusão nas quais existem múltiplos destinatários e as sondas espaciais que, pelo facto do tempo de propagação ser muito grande, utilizam essencialmente canais simplex.

Uma outra técnica, que é designada por *correcção por pedido automático de repetição — Automatic Repeat reQuest (ARQ)*, exige um canal em sentido oposto — comunicação duplex —, que permita ao descodificador do canal pedir ao codificador a retransmissão de palavras (blocos) sempre que *detecte* a existência de erros. Nesta situação o código é utilizado apenas como detector de erros e a correcção processa-se por repetição.

A técnica de correcção progressiva é raramente utilizada nos sistemas de transmissão de dados visto funcionarem normalmente em duplex possibilitando a retransmissão e proporcionando reduções muito maiores da probabilidade global de erro com rendimentos substancialmente melhores. Ela é utilizada noutras aplicações, por exêmplo, nas memórias de computadores ou nas gravações digitais magnéticas ou ópticas.

Como termo de comparação com as técnicas ARQ, consideraremos neste capítulo apenas o caso da correcção progressiva.

As técnicas ARQ serão abordadas num próximo capítulo no contexto dos protocolos de comunicação de dados.

9.5.2 Probabilidade de Erro na Correcção Progressiva

Um código corrector é caracterizado por corrigir até e_c erros por palavra. Ocasionalmente corrigirá mais do que e_c erros. Portanto, uma palavra de n dígitos será incorrectamente descodificada, ou seja, estará normalmente errada quando ocorrerem $e_c + 1$ ou mais erros. A probabilidade, P_{pe}, de uma palavra errada será pois

$$P_{pe} \leq P(e_c + 1, n) + P(e_c + 2, n) + \ldots + P(n, n) = \sum_{i=e_c+1}^{n} P(i, n)$$

em que $P(i, n)$ designa a probabilidade de existirem exactamente i erros numa palavra de n dígitos cujo valor, como se sabe, é dado pela distribuição binomial:

$$P(i, n) = C_i^n \; P_e^i \, (1 - P_e)^{n-i}$$

onde P_e é a probabilidade de erro por dígito no canal. Se se tiver $P_e \ll 1$, o que de facto se verifica nos sistemas reais, então $P(i + 1, n) \ll P(i, n)$, sendo aceitável tomar-se a aproximação

$$P_{pe} \approx P(e_c + 1, n) \approx C_{e_c+1}^n \; P_e^{e_c+1} \tag{9.10}$$

o que significa que uma palavra errada, típicamente, possuirá e_c+1 dígitos errados dos quais, em média,

$$\frac{k}{n}(e_c + 1)$$

são dígitos de informação e os restantes erros são nos dígitos de verificação.

Quando são transmitidas $N \gg 1$ palavras as quais contêm um total de kN dígitos de informação, espera-se receber NP_{pe} palavras erradas ou seja $\frac{k}{n}(e_c + 1)NP_{pe}$ dígitos de informação errados. Portanto, a probabilidade de se entregar ao destino um dígito de informação errado, isto é, a probabilidade de erro por bit[6], P_E, quando é utilizada a correcção de erros progressiva é

$$P_E = \frac{\frac{k}{n}(e_c+1)NP_{pe}}{kN} = \frac{e_c+1}{n}\,P_{pe}$$

ou seja,

$$\boxed{P_E = C_{e_c}^{n-1}\;P_e^{e_c+1}} \tag{9.11}$$

É importante notar que este resultado está de acordo com o Teorema de Shannon da codificação do canal, segundo o qual, uma melhoria (diminuição) da probabilidade de erro ($P_E < P_e$) e portanto menor equivocação e maior transferência de informação se faz à custa duma diminuição da velocidade de transmissão (débito) de informação, neste caso de um factor igual a $\rho = \frac{k}{n}$. O débito da fonte, r_s, e o débito no canal, r_c, estão pois relacionados por

$$\boxed{\rho = \frac{k}{n} = \frac{r_s}{r_c}} \tag{9.12}$$

9.6 Problemas

9.1 – Seja $C(x)$ um polinómio de grau $n-1$ com coeficientes num corpo binário e $C^{(i)}(x)$ o polinómio resultante da rotação cíclica direita dee grau $n-1$ com coeficientes num corpo binário e $C^{(i)}(x)$ o polinómio resultante da rotação cíclica direita de i posições dos coeficientes de $C(x)$. Mostrar que o polinómio $C^{(i)}(x)$ é o resto da divisão de $x^i\,C(x)$ por x^n+1, isto é, que

$$x^i\,C(x) = q(x)\,(x^n+1) + C^{(i)}(x)$$

onde $q(x)$ é o polinómio quociente da divisão.

9.2 – Um código linear cíclico $(15, 5)$ tem como polinómio gerador:

$$g(x) = 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10}$$

a) Esquematizar um codificador e um calculador de síndroma para este código.

[6] passaremos a designar o dígito binário por *bit* sempre que não haja confusão com a unidade de medida da informação

b) Determinar o polinómio do código, quer sistemático quer criptográfico, correspondente ao polinómio de mensagem $D(x) = 1 + x^2 + x^4$.

c) O polinómio $R(x) = 1+x^4+x^6+x^8+x^{14}$ pertence ao dicionário do código? Em caso negativo determine o respectivo síndroma.

9.3 – O polinómio gerador de um código cíclico $(15, 7)$ é

$$g(x) = 1 + x^4 + x^6 + x^7 + x^8$$

a) Determinar a palavra do código sistemático para o polinómio de mensagem $D(x) = x^2 + x^3 + x^4$.

b) Supôr que o primeiro e último dígitos do polinómio de código $C(x)$ para $D(x)$ sofreu um erro na transmissão. Determinar o síndroma de $R(x)$.

9.4 – Construir um código para correcção de erros simples com um comprimento de bloco da mensagem igual a 11 e mostrar, através de um exêmplo, que ele pode efectivamente corrigir erros simples.

9.5 – Num código de repetição tripla, por cada dígito binário produzido pela fonte são enviados três. Por cada 0 é enviado 000 e por cada 1 é enviado 111.

a) Mostre que este código é corrector de erros simples.

b) Determine um polinómio gerador do código.

c) Determine uma expressão para a probabilidade de erro por bit em função da probabilidade, P_e, de erro no canal.

d) Compare este código com o de Hamming (7,4).

9.6 – Mostre que um polinómio gerador de um código cíclico binário (n, k) tem sempre $g_{n-k} = g_0 = 1$ e que portanto só existem 2^{n-k-1} polinómios candidatos a geradores desse código.

9.7 – O código de Hamming $(7, 4)$ pode ser extendido de modo a formar um código $(8, 4)$ por adição de um quarto dígito de verificação escolhido de modo a tornar *par* a paridade de todas as palavras de código. Aplique este processo de extensão aos códigos da tabela 9.1 e mostre que $d_{min} = 4$. Determine o respectivo polinómio gerador.

fim do capítulo 9