FEEC
Faculdade de Engenharia Elétrica e de Computação
UNICAMP

DOUTORADO

CÓDIGOS DE LINHA A PARTIR DE CÓDIGOS CORRETORES DE ERRO CONCATENADOS

Tese apresentada ao Departamento de Comunicações da Faculdade de Engenharia Elétrica e de Computação da UNICAMP como parte dos requisitos exigidos para a obtenção do título de Doutor em Engenharia Elétrica.

AUTOR: **Evelio Martín García Fernández**

ORIENTADOR: **Prof. Dr. Renato Baldini Filho**

COMISSÃO JULGADORA

Prof. Dr. Bartolomeu Ferreira Uchoa Filho
Prof. Dr. Rege Romeu Scarabucci
Prof. Dr. Helio Waldman
Prof. Dr. Jaime Portugheis

Abril 2001

RESUMO

Este trabalho aborda a obtenção de códigos de linha a partir de códigos corretores de erro binários concatenados construídos através da combinação de códigos externos de Reed-Solomon com códigos internos que possuem boas propriedades de codificação de linha. Os códigos internos foram obtidos a partir da modificação de códigos corretores de erro (bloco e convolucional) conhecidos com o objetivo de se obter seqüências codificadas em que o número de símbolos iguais consecutivos ("runlength") é limitado. Esta modificação é feita sem alterar a taxa de transmissão de dados nem a capacidade de correção de erro dos códigos originais. Simulações feitas para o canal Gaussiano levaram a resultados que concordam com a performance dos códigos originais utilizados. Através da concatenação é possível a obtenção de códigos de linha corretores de erro praticamente de qualquer comprimento e capacidade de correção de erro. Também foram feitas transformações nas seqüências codificadas geradas por estes códigos visando suprimir seu conteúdo espectral em baixas freqüências. Simulações realizadas nos permitiram comprovar que os códigos projetados podem ser transformados em códigos "dc-free" com uma pequena diminuição da taxa de transmissão de dados.

ABSTRACT

This work deals with the construction of line codes based on concatenated binary error control codes, which are a combination of outer Reed-Solomon codes with inner codes that possess good properties of line coding. The inner codes were obtained by modification of known error correcting codes (block and convolutional) in which the number of consecutive like-symbols (runlength) is limited in the coded sequence. This modification is made maintaining the data transmission rate and the error correcting capacity of the original codes. Simulations results for the Gaussian channel have shown that the performance of the line codes agrees with the performance of the original codes. Through the code concatenation it is possible to obtain error control line codes of practically any length and error correcting capacity. Transformations have also been made in the coded sequences in order to suppress the spectral content at low frequencies. Computer simulations allowed us to check that the designed codes can be made dc-free codes with a slight decrease in the data transmission rate.

O Projeto de Pesquisa desenvolvido durante quatro anos, e que culminou com a elaboração desta Tese foi, durante o primeiro ano de execução, financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e, durante os últimos três anos, pela Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

AGRADECIMENTOS

Pessoas, profissionais em particular e instituições, ajudaram a transformar a realidade de um projeto de pesquisa no estudo apresentado no decorrer das páginas aqui contidas. Assim, manifestamos a todos os nossos agradecimentos e, de modo especial,

ao professor Renato Baldini Filho, pela sua colaboração e orientação para a realização deste trabalho, aconselhando e sugerindo-me idéias que foram muito importantes;

aos professores da Faculdade de Engenharia Elétrica e de Computação da UNICAMP, especialmente do Departamento de Comunicações, por sua contribuição para a culminação deste trabalho;

ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela bolsa de estudo concedida durante o primeiro ano de execução do projeto de Tese;

à Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP), pelo financiamento concedido para a execução do projeto de pesquisa;

à Rosa Lydia Teixeira Corrêa pela paciência, companheirismo, amizade e compreensão com que acompanhou essa trajetória;

à minha família, pelo carinho e atenção que dispensaram a mim durante toda minha vida e especialmente durante os últimos seis anos em que tenho estado longe dela.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

O objetivo fundamental deste trabalho é a obtenção de códigos para a correção de erros em sistemas de comunicações digitais com restrições no número de símbolos consecutivos iguais ("runlength") transmitidos. Exemplos destes sistemas são os sistemas de gravação magnética, sistemas de armazenamento de informação e sistemas de transmissão digital em banda básica, entre outros. Em todos estes sistemas, seqüências longas de símbolos iguais podem provocar um mal funcionamento dos circuitos digitais encarregados de garantir o sincronismo de bit ou da equalização adaptativa. No caso particular dos sistemas de gravação magnética, restrições adicionais são impostas para eliminar o conteúdo de corrente contínua

(nível DC) da seqüência codificada, uma vez que estes sistemas não respondem a sinais de baixa freqüência.

Nos sistemas convencionais, a codificação de canal tradicionalmente é feita em duas etapas: (a) codificação para controle de erros e, (b) codificação para atender às restrições do canal. A codificação para controle de erros é realizada através da adição sistemática de símbolos redundantes à seqüência de informação a ser transmitida. Estes símbolos redundantes são utilizados posteriormente pelo decodificador no receptor para detectar e/ou corrigir alguns dos erros que possam estar presentes na mensagem recebida. A principal exigência neste processo consiste em alcançar a proteção necessária contra os inevitáveis erros de transmissão de informação, sem ter que se pagar um preço muito alto pela redução da taxa de transmissão decorrente da adição de símbolos redundantes à seqüência de informação.

A codificação para atender às restrições do canal é realizada através de códigos conhecidos como códigos de gravação ou códigos de modulação. Estes códigos transformam a seqüência de saída do código corretor de erro numa forma de onda adequada às restrições do canal. Em sistemas de transmissão adequados para cabos elétricos e ópticos, estes códigos são chamados de códigos de linha [Immink, 1999].

A conformação em cascata destes processos de codificação traz consigo duas desvantagens fundamentais: (a) os códigos de modulação (ou de linha) geralmente provocam uma propagação dos erros de transmissão e, (b) estes códigos introduzem uma redundância adicional na seqüência de símbolos transmitidos que reduz a taxa de transmissão sem ter utilidade para detecção e/ou correção de erros. Ambas as desvantagens impõem exigências adicionais ao código corretor de erro. Códigos construídos desta forma podem ser encontrados em [Lee, P., 1989] e [Abdel, 1995].

Uma alternativa consiste em realizar os dois processos de codificação conjuntamente utilizando-se códigos corretores de erro com boas propriedades de codificação de linha, ou seja, códigos que além de detectar e/ou corrigir erros possam entregar na sua saída, seqüências codificadas com "runlength" limitado. Estes códigos podem ser obtidos a partir da modificação de códigos corretores de erro conhecidos. Desta forma, pode-se obter códigos em que a quantidade máxima de símbolos consecutivos iguais enviados pelo canal seja limitada a um determinado valor (garantindo assim a recuperação do sincronismo no receptor), e com a capacidade de correção de erros dos códigos originais.

Exemplos da obtenção de códigos com "runlength" limitado a partir de códigos corretores de erro conhecidos podem ser encontrados em [Popplewell, 1992] e [Honary, 1997]. Em ambos os casos, os códigos obtidos são de comprimento pequeno e, portanto, com capacidade de correção de erro pequena.

Nos capítulos seguintes serão apresentados procedimentos para a transformação de códigos corretores de erro conhecidos em códigos de linha. No Capítulo 2, serão descritos algoritmos de modificação de códigos de bloco corretores de erro visando à obtenção de seqüências codificadas com limitantes inferiores para o número consecutivo de símbolos iguais contidos nelas. A forma de se encontrar esses limitantes também será descrita.

Os códigos obtidos desta forma serão utilizados no Capítulo 3 conjuntamente com códigos não binários de Reed-Solomon na construção de esquemas de codificação concatenada. Desse modo, será possível a obtenção de códigos corretores de erro longos e com capacidade de correção de erro maior sem serem alterados os limitantes inferiores para o valor máximo de "runlength".

Procedimentos equivalentes para a obtenção de códigos de linha a partir de códigos convolucionais são apresentados no Capítulo 4.

Em todos os casos, os códigos construídos serão apresentados de um ponto de vista eminentemente prático, ou seja, a construção de códigos não será abordada no sentido de demonstrar teoricamente a existência de códigos com determinadas características sem ter em conta a possibilidade real de implementação prática dos algoritmos de codificação e decodificação para eles.

No Capítulo 5, as seqüências codificadas obtidas a partir dos códigos construídos nos capítulos anteriores são transformadas em seqüências cujas curvas de densidade espectral de potência apresentam redução do conteúdo espectral em baixas freqüências.

Por último, no Capítulo 6, são realizadas considerações finais sobre o trabalho bem como algumas sugestões para trabalhos futuros.

CAPÍTULO 2

CÓDIGOS DE BLOCO CORRETORES DE ERRO COM "RUNLENGTH" LIMITADO

2.1 INTRODUÇÃO

Os códigos com "runlength" limitado convencionais encontrados na literatura especializada geralmente são construídos a partir de seqüências conhecidas como seqüências (d, k), as quais apresentam as seguintes características [Lee, 1989]:

1) dois "1's" estão separados por ao menos d "0's" consecutivos e,

2) o comprimento de qualquer seqüência de "0's" é, no máximo, igual a k $(0 \leq d < k < \infty)$.

O valor do parâmetro *d* tem um papel importante no controle da interferência intersimbólica. O valor do parâmetro *k* por sua vez, determina o máximo "runlength" nas seqüências de símbolos produzidas por estes códigos e tem um papel importante na recuperação do sincronismo de relógio no receptor.

Neste capítulo serão apresentados algoritmos de transformação de códigos de bloco corretores de erro em códigos com "runlength" limitado. As seqüências de símbolos codificados geradas por estes códigos são do tipo (0, *k*)

2.2 RELAÇÃO ENTRE CÓDIGOS DE BLOCO CORRETORES DE ERRO E CÓDIGOS DE LINHA

Em geral, os códigos de bloco corretores de erro lineares não possuem boas propriedades de codificação de linha, pois: a) eles têm uma disparidade acumulada ilimitada (desbalanceamento acumulado entre "uns" e "zeros" enviados através do canal) e, portanto, não produzem a eliminação da componente de corrente contínua (nível DC) no espectro de potência e b) eles têm um "runlength" ilimitado, o que pode levar a problemas de sincronismo no receptor.

Todos os códigos corretores de bloco de erro binários lineares e transparentes têm "runlength" ilimitado devido à inclusão, no conjunto das palavras-código, das palavras toda "um" e toda "zero". Define-se um código transparente como sendo um código em que o inverso aditivo de cada palavra-código é também uma palavra-código. Assim, para produzir um código de bloco corretor de erro com "runlength" limitado é necessário encontrar um conjunto de palavras-código que não contenha as palavras formadas por todos os símbolos iguais a zero ou a um, mas que mantenha as propriedades de distância do código original. Isto

pode ser feito, formando classes laterais (ou "cosets") do código original mediante a adição a todas as palavras-código, de uma palavra de n bits não pertencente ao código, que será chamada de vetor modificador. A linearidade do código assegura que a distância mínima deste será preservada e, portanto, a capacidade de correção de erros. Isto significa que a seqüência de informação pode ser codificada de acordo com o código original, mas antes da transmissão, um vetor modificador (líder de "coset") é somado a cada palavra-código fazendo com que a seqüência de símbolos binários enviada pelo canal tenha um "runlength" limitado a um determinado valor. No receptor é feita a operação inversa, portanto, antes da decodificação, da palavra recebida é extraído o vetor modificador e em seguida esta é decodificada segundo o código corretor de erro original, como mostrado na Figura 2.1.

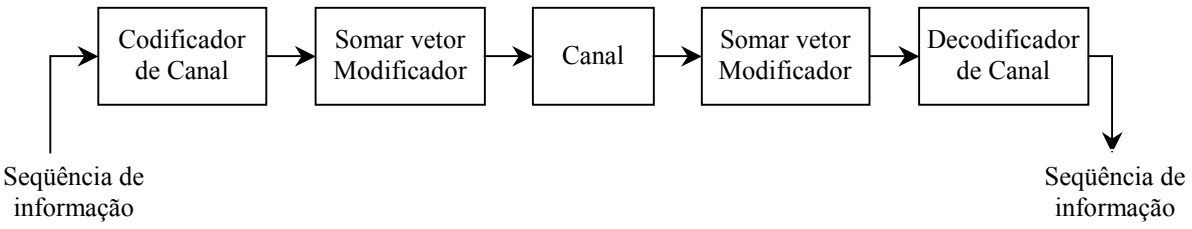

Figura 2.1: Diagrama de Blocos de um Sistema que Utiliza Códigos Corretores de Erro com "Runlength" Limitado.

2.3 MODIFICAÇÃO DE CÓDIGOS DE BLOCO LINEARES TRANSPARENTES

Existem 2^{n-k} classes laterais diferentes associadas a um código de bloco (n, k) corretor de erro linear transparente sendo que uma delas é o conjunto das palavras do código corretor de erro original. Portanto, existem $2^{n-k}-1$ classes laterais que não contém as

palavras toda "um" e toda "zero" e conseqüentemente podem produzir um código com "runlength" limitado. O problema é: qual a classe lateral que proporciona o mínimo "runlength"? Isto pode ser resolvido em duas etapas: primeiro, determinar o mínimo "runlength" possível de se alcançar com uma determinada classe lateral e segundo, encontrar um vetor modificador para formar a classe lateral que satisfaz o mínimo "runlength" previamente determinado. Popplewell e O'Reilly [Popplewell, 1992] propuseram um método para se alcançar este objetivo e que será descrito a seguir:

Dado um conjunto de palavras-código que não contém as palavras formadas totalmente por "uns" e por "zeros", tem-se que o máximo "runlength" (RL_{MAX}) é definido como o máximo número de 1/0s consecutivos em uma palavra-código (R_{MID}), ou o máximo número de 1/0s no início de uma palavra-código (R_{START}) mais o máximo número de 1/0s ao final de uma palavra-código (R_{END}), tomando-se entre as duas possibilidades o maior valor.

Portanto,

$$RL_{MAX} = \text{MAX}\{R_{END} + R_{START}\ ,\ R_{MID}\} \tag{2.1}$$

Dado um código de bloco corretor de erro (n, k) com matriz geradora G utilizando os três teoremas seguintes, pode-se determinar os valores mínimos atingíveis por R_{END} , R_{START} e R_{MID} , que serão chamados de MinR_{END}, MinR_{START} e MinR_{MID} respectivamente, para alguma classe lateral do código corretor de erros, e portanto definir um limitante inferior para o máximo "runlength" (MinRL_{MAX}) do código.

Teorema 2.1:

MinR_{START} = o número de colunas linearmente independentes (LI) consecutivas no início da matriz G. [Popplewell, 1992]

Teorema 2.2:

MinR_{END} = o número de colunas linearmente independentes consecutivas no final da matriz *G*. [Popplewell, 1992]

Teorema 2.3:

MinR_{MID} = o máximo número de colunas linearmente independentes consecutivas na matriz *G*. [Popplewell, 1992]

A conseqüência destes teoremas é que o ordenamento das colunas da matriz geradora *G* pode afetar significativamente o limitante MinRL_{MAX} e como o reordenamento das colunas de *G* não afeta as características de correção de erro do código original, podem ser realizadas permutações de colunas desta matriz *G* para fazer com que MinR_{START}, MinR_{MID} e MinR_{END} sejam o menor possível e assim produzir um limitante inferior para o "runlength" do código.

Os teoremas possibilitam, na maioria dos casos, encontrar classes laterais que cumpram com os limitantes MinR_{END}, MinR_{START} e MinR_{MID} e, portanto, determinar MinRL_{MAX}, o limitante inferior de RL_{MAX} usando a equação 2.1. Entretanto, não é possível saber diretamente se existe uma classe lateral que possa atingir esses limitantes nem como selecionar um vetor modificador adequado para produzir essa classe lateral.

Popplewell e O'Reilly [Popplewell, 1992] também propuseram um método para encontrar a classe lateral que melhor se aproxima do limitante MinRL_{MAX}, o qual pode-se resumir da seguinte forma:

a) Dada a matriz *G'* do código de bloco (*n*, *k*) modificado, formar a matriz *H'* de verificação de paridade do código.

b) Reordenar a matriz *H'* realizando operações elementares sob as linhas para produzir uma matriz $\widetilde{H}$ que satisfaz as seguintes condições:

(i) $\widetilde{H}$ tem ao menos uma linha tal que,

$$\begin{gathered}\widetilde{h}_{z,i} = 0 \text{ para todo } i = \text{Min}R_{START} + 2, \\ \text{Min}R_{START} + 3, \\ \vdots \\ n\end{gathered} \tag{2.2}$$

(ii) $\widetilde{H}$ tem ao menos uma linha tal que,

$$\begin{gathered}\widetilde{h}_{z,i} = 0 \text{ para todo } i = 1, 2, ..., p, \\ \text{Min}RL_{MAX} + p + 2, \\ \text{Min}RL_{MAX} + p + 3, \\ \vdots \\ n\end{gathered} \tag{2.3}$$

para todo $p = 1, 2, ..., n - \text{Min}RL_{MAX} - 2$.

(iii) $\widetilde{H}$ tem pelo menos uma linha tal que,

$$\widetilde{h}_{z,i} = 0 \text{ para todo } i = 1, 2, ..., \; n - \text{Min}R_{END} - 1 \tag{2.4}$$

A matriz de verificação de paridade pode ser colocada desta forma usando-se um procedimento que requer um máximo de $(n-k)(n-k-1) \approx (n-k)^2$ operações sobre as linhas. Portanto, para um determinado código, a complexidade do procedimento aumenta com o quadrado do número de bits de verificação de paridade.

Note-se que se Min$RL_{MAX} > n - 2$ então a condição (ii) não se aplica.

c) Se $\widetilde{H}$ não satisfaz todas as condições, então não existe uma classe lateral capaz de atingir o limitante RL_{MAX} e portanto, um ou mais de um dos limitantes para MinR_{END}, MinR_{START} ou MinR_{MID} devem ser aumentados até que todas as condições em (*b*) sejam satisfeitas.

d) Quando $\widetilde{H}$ satisfaz todas as condições em (*b*) então forma-se uma nova matriz *S* a qual é construída a partir de todas as linhas de $\widetilde{H}$ que satisfazem ao menos uma das condições (i), (ii) e (iii).

e) Forma-se a seguinte equação matricial:

$$\mathbf{m}\, S^t = \mathbf{1} \tag{2.5}$$

f) Qualquer solução desta equação produzirá um vetor modificador de *n* bits, **m**, que vai formar uma classe lateral do código original que satisfaz os limitantes predeterminados para RL_{MAX}.

2.3.1 EXEMPLO DA UTILIZAÇÃO DO PROCEDIMENTO ANTERIOR PARA A MODIFICAÇÃO DO CÓDIGO BCH (15, 5, 7)

O polinômio gerador deste código é dado por:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \tag{2.6}$$

Formando a matriz geradora, tem-se que:

$$G=\begin{bmatrix}1&1&1&0&1&1&0&0&1&0&1&0&0&0&0\\0&1&1&1&0&1&1&0&0&1&0&1&0&0&0\\0&0&1&1&1&0&1&1&0&0&1&0&1&0&0\\0&0&0&1&1&1&0&1&1&0&0&1&0&1&0\\0&0&0&0&1&1&1&0&1&1&0&0&1&0&1\end{bmatrix} \tag{2.7}$$

Trocando a coluna 4 com a 11 e a coluna 8 com a 12 obtém-se a seguinte matriz geradora modificada:

$$G'=\begin{bmatrix}1&1&1&1&1&1&0&0&1&0&0&0&0&0&0\\0&1&1&0&0&1&1&1&0&1&1&0&0&0&0\\0&0&1&1&1&0&1&0&0&0&1&1&1&0&0\\0&0&0&0&1&1&0&1&1&0&1&1&0&1&0\\0&0&0&0&1&1&1&0&1&1&0&0&1&0&1\end{bmatrix} \tag{2.8}$$

Usando os teoremas 2.1, 2.2 e 2.3 pode-se encontrar,

$\mathrm{Min}R_{END}=3$

$\mathrm{Min}R_{START}=3$

$\mathrm{Min}R_{MID}=5$

Assim,

$$RL_{MAX}=\mathrm{MAX}\{3+3,5\}=6 \tag{2.9}$$

Para encontrar o vetor modificador que satisfaça esse limitante tem-se que:

a) Formar a matriz de verificação de paridade do código gerado por *G'*

$$H' = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{2.10}$$

b) As condições para $\tilde{H}$ são:

(i) Uma linha com os últimos 11 elementos iguais a zero.

(ii) Uma linha com o primeiro e os últimos 7 elementos iguais a zero, uma linha com os primeiros 2 e os últimos 6 elementos iguais a zero,

.
.
.

uma linha com os primeiros 7 e o último elemento iguais a zero.

(iii) Uma linha com os primeiros 11 elementos iguais a zero.

Realizando operações elementares sobre as linhas de *H'*, obtém-se:

$$\widetilde{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{2.11}$$

c) A linha 1 satisfaz a condição (i), as linhas 2-7 satisfazem a condição (ii) e a linha 8 satisfaz a condição (iii). As linhas 9 e 10 são redundantes.

d) Formar a matriz *S*,

$$S = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.12}$$

e) Agora, resolvendo a equação $\mathbf{m}\,S^t = \mathbf{1}$, pode-se obter como uma possível solução,

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

A Tabela 2.1 mostra uma lista das palavras-código modificadas junto às palavras do código BCH (15, 5, 7) original. As maiores seqüências de símbolos iguais consecutivos estão realçadas em negrito. Note-se que não é possível encontrar mais do que RL_{MAX} = 6 uns ou

zeros consecutivos como previsto da equação (2.1). Existem muitas outras soluções possíveis para a equação $\mathbf{m}\,S^t = \mathbf{1}$, sendo que cada uma delas produzirá uma classe lateral com RL_{MAX} = 6.

Tabela 2.1: Palavras-Código do Código BCH (15, 5, 7) Original e Modificado.

Palavras da Fonte	Palavras-Código	Palavras-Código Modificadas
00000	000000000000000	100001001010001
00001	000011101100101	100010100110100
00010	000011011011010	100010010001011
00011	000000110111111	100001111101110
00100	001110100011100	101111101001101
00101	001101001111001	101100000101**000**
00110	001101111000110	101100110010**111**
00111	001110010100011	101111011110010
01000	011001110110000	**111**000111100001
01001	011010011010101	**111**011010000100
01010	011010101101010	**111**011100111011
01011	011001000001111	111000001011110
01100	010111010101100	110110**111111**01
01101	010100111001001	110101110011**000**
01110	010100001110110	110101000100**111**
01111	010111100010011	110110101000010
10000	111111001000000	011110000010001
10001	111100100100101	011101101110100
10010	111100010011010	011101011001011
10011	111111111111111	011110110101110
10100	110001101011100	010000100001101
10101	110010000111001	010011001101**000**
10110	110010110000110	010011111010**111**
10111	110001011100011	010000010110010
11000	100110111110000	**000**111110100001
11001	100101010010101	**000**100011000100
11010	100101100101010	**000**100101111011
11011	100110001001111	**000**111000011110
11100	101000011101100	001001010111101
11101	101011110001001	001010111011**000**
11110	101011000110110	001010001100**111**
11111	101000101010011	001001**000000**10

O procedimento é fundamentado na identificação de vetores modificadores que têm correspondência com limitantes ótimos para o "runlength" do código. Essas idéias são aplicáveis de forma geral a todos os códigos lineares transparentes ao mesmo tempo em que o procedimento torna-se complexo na medida que se aumenta o comprimento do código.

O aspecto mais importante a assinalar é que uma vez que o vetor modificador é identificado, a sua implementação praticamente não aumenta a complexidade com respeito a um código corretor de erros convencional.

Alguns dos resultados obtidos em [Popplewell, 1992] são mostrados na Tabela 2.2.

Tabela 2.2: Códigos Corretores de Erro com "Runlength" Limitado.

Código Corretor de Erros	MinRL_{MAX}
BCH (15, 5, 7)	6
BCH (15, 7, 5)	7
BCH (15, 11, 3)	17
BCH (31, 16, 7)	16
BCH (31, 21, 5)	32
BCH (31, 26, 3)	38
Golay (23, 12, 7)	16

Do exemplo mostrado a partir do código BCH (15, 5, 7), deve-se notar que se tomou como ponto de partida a matriz geradora original na forma não sistemática. Isto aumenta a complexidade na hora de ser gerado o código modificado com respeito a esquemas que utilizam codificação sistemática.

2.4 ALGORITMO PARA MODIFICAR CÓDIGOS DE BLOCO TRANSPARENTES SISTEMÁTICOS

Na secção anterior foi apresentado um método para modificar códigos de bloco transparentes a partir da matriz geradora na forma não sistemática. Nesse método, as colunas da matriz geradora são reordenadas de acordo com os teoremas 2.1, 2.2 e 2.3, sendo que a permutação ótima é alcançada mediante a escolha de dois subconjuntos de colunas linearmente dependentes S_1 e S_2 do conjunto S (o conjunto formado por todas as colunas de G) tal que, $N(S_1)+N(S_2)$ e $N(S_1 \cap S_2)$ são minimizados, onde $N(T)$ é o número de elementos no conjunto T. Assim, as primeiras $N(S_1)$ colunas de G' são o subconjunto S_1 e as últimas $N(S_2)$ colunas são S_2 ou vice-versa. As colunas restantes são os vetores pertencentes ao subconjunto $S-(S_1 \cup S_2)$.

Nesta secção será apresentado um algoritmo mais simples para modificar códigos de bloco transparentes a partir da matriz geradora na forma sistemática que atingem o limitante do "runlength" definido pelos teoremas 2.1, 2.2 e 2.3. Este algoritmo foi desenvolvido como parte do trabalho realizado no curso de Mestrado [Fernández, 1997-1]. A seguir será apresentado um resumo do algoritmo propriamente dito devido à importância deste na construção dos códigos concatenados que serão apresentados no seguinte capítulo.

O algoritmo para modificar códigos de bloco cíclicos transparentes sistemáticos de tal forma que eles se tornem adequados para alcançar o limitante do "runlength" é dado abaixo:

a) A partir do polinômio gerador $g(x)$ do código, formar a matriz geradora G na forma sistemática.

b) Encontrar um subconjunto S_1 de colunas linearmente dependentes, tal que $N(S_1)$ seja mínimo.

Para se encontrar este subconjunto, é selecionada dentre as $n-k$ colunas de verificação de paridade da matriz *G*, a coluna (ou menor combinação de colunas) com menor quantidade de uns. Sejam:

t = vetor soma das colunas selecionadas,

nu = número de uns em **t**,

nc = número de colunas contidas nessa combinação.

Então, para que as colunas contidas em S_1 sejam linearmente dependentes, tem-se que:

$$N(S_1) = nu + nc \tag{2.13}$$

sendo que as *nu* colunas contidas neste subconjunto são selecionadas dentre as *k* colunas da submatriz de identidade de *G*, especificamente aquelas em que a posição dos uns coincide com a posição dos uns do vetor **t**.

c) Colocar o subconjunto S_1 no início da matriz geradora modificada *G*'.

Desta forma e tendo em conta o teorema 2.1 tem-se que:

$$\mathrm{Min}R_{START} = N(S_1) - 1 = nu + nc - 1 \tag{2.14}$$

d) Procurar nas colunas ainda não utilizadas das $n-k$ colunas de verificação de paridade de *G*, se existe alguma em que a posição dos uns não coincida com a posição dos uns do vetor **t** no passo *b*. A quantidade de uns dessa coluna deve ser a menor possível.

Dependendo da existência ou não de alguma coluna que satisfaça a condição anterior, dois casos são possíveis de ocorrer:

Caso 1: Existe uma coluna como descrita no passo *d*. Neste caso $N(S_1 \cap S_2) = 0$, e o método continua da seguinte forma:

e) Colocar a coluna selecionada como sendo a última coluna da matriz geradora modificada *G'*. Seja,

mnu = número de uns da coluna anterior

f) Colocar a esquerda da coluna anterior as *mnu* colunas dentre as colunas da submatriz de identidade de *G* ainda não utilizadas de modo que a posição dos uns coincida com a posição dos uns da última coluna de *G'*. Assim,

$$N(S_2) = mnu + 1 \tag{2.15}$$

Desta forma e tendo em conta o teorema 2.2, tem-se que:

$$MinR_{END} = N(S_2) - 1 = mnu \tag{2.16}$$

g) Completar a matriz *G'* com as restantes colunas de *G*.

Caso 2: Não existe uma coluna como descrita no passo *d*. Neste caso $N(S_1 \cap S_2) \neq 0$ e portanto, os dois subconjuntos S_1 e S_2 vão se sobrepor. O método continua buscando minimizar esta interseção:

e) Colocar na continuação do subconjunto S_1 as colunas ainda não utilizadas das $n-k$ colunas de verificação de paridade de *G*. Seja **q**, o vetor soma dessas colunas.

Ao fazer esta operação deve-se ter em conta que a ordem das *nu* colunas que estão contidas em S_1 não pode ser qualquer uma. As últimas colunas à direita de S_1 serão aquelas em que a posição dos uns coincide com posições de uns do vetor **q** e do vetor **t** simultaneamente.

Seja r = número de uns do vetor **q** cujas posições coincidem com posições de uns do vetor **t**. Então, existirão r colunas fazendo parte tanto de S_1 como de S_2.

f) Colocar à direita das colunas do passo *e* aquelas colunas da submatriz de identidade de G em que a posição dos uns coincida com a posição dos uns restantes de **q**.

g) Completar à direita com as restantes colunas de G.

Desta forma,

$$N(S_2) = n - N(S_1) + r \tag{2.17}$$

Assim, tendo em conta o teorema 2.3 tem-se que:

$$\text{Min}R_{END} = N(S_2) - 1 = n - N(S_1) + r - 1 = n - \text{Min}R_{START} + r - 2 \tag{2.18}$$

No caso particular dos códigos de Hamming todas as $n-k$ colunas de verificação de paridade têm igual número de uns e a combinação que tem a menor quantidade de uns é a formada pela soma de todas elas. Então, ao executar o passo *b*, deve colocar-se ao início da matriz G' qualquer combinação de $n-k-1$ colunas de verificação de paridade e continuar como descrito no algoritmo anterior.

Uma vez modificada a matriz geradora e encontrados os limitantes $\text{Min}R_{START}$ e $\text{Min}R_{END}$, pode-se encontrar o limitante inferior do "runlength" do código a partir da equação (2.1).

Por último, a partir da matriz G', procura-se o líder de classe lateral (ou vetor **m**) que somado a todas as palavras-código geradas por G', produz um código que atinge os limitantes encontrados previamente.

O algoritmo pode ser mais bem compreendido através do seguinte exemplo.

2.4.1 EXEMPLO DE MODIFICAÇÃO DE CÓDIGOS EM QUE OS SUBCONJUNTOS S_1 E S_2 NÃO SE SOBREPÕEM

Para tratar o caso em que os subconjuntos S_1 e S_2 não se sobrepõem, se aplicará o caso 1 do algoritmo anterior para modificar o código de Golay (23, 12, 7) que tem o polinômio gerador $g(x) = x^{11} + x^9 + x^7 + x^6 + x^5 + x + 1$.

a) A matriz geradora na forma sistemática é dada por:

$$G = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1\\
0&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&1&0&0&1&0&1&0&1\\
0&0&0&1&0&0&0&0&0&0&0&0&1&1&0&0&0&1&1&1&0&1&1\\
0&0&0&0&1&0&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&1&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&0&0&1&1&0&1&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&1
\end{bmatrix} \tag{2.19}$$

b) Fazendo todas as combinações lineares das colunas de verificação de paridade da matriz anterior, tem-se que a combinação com menor número de uns é:

$$\mathbf{t} = c_{13} + c_{15} + c_{16} + c_{18} + c_{19} + c_{20} + c_{21} \tag{2.20}$$

Portanto a matriz geradora modificada *G*' começa a ser montada com estas colunas da seguinte forma:

$$G' = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2.21}$$

onde,

$$\mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = c_8 \tag{2.22}$$

$nu = 1$

$nc = 7$

Portanto, da equação 2.13 tem-se que:

$$N(S_1) = nu + nc = 8 \tag{2.23}$$

c) Assim, as oito primeiras colunas de *G*' serão:

$$G' = \begin{bmatrix} 1&1&0&1&1&0&0&0 \\ 1&1&1&0&0&1&0&0 \\ 1&0&1&0&1&0&1&0 \\ 1&0&0&1&1&1&0&0 \\ 1&0&0&1&0&1&1&0 \\ 0&1&0&1&1&0&1&0 \\ 0&1&1&0&1&1&0&0 \\ 1&1&1&1&1&1&1&1 \\ 0&0&1&0&1&1&1&0 \\ 0&1&0&1&0&1&1&0 \\ 1&1&1&0&0&0&1&0 \\ 0&0&1&1&0&0&0&0 \end{bmatrix} \tag{2.24}$$

Da equação (2.14) tem-se que:

$$\text{Min}R_{START} = N(S_1) - 1 = 7 \tag{2.25}$$

d) A coluna c_{23} não contém o dígito binário "1" na oitava posição (posição do um do vetor **t**), e não existe nenhuma outra coluna, ainda não utilizada, com menor quantidade de uns, que satisfaça esta condição, portanto este é um exemplo do caso 1, onde $N(S_1 \cap S_2) = 0$.

e) A coluna c_{23} é colocada como sendo a última coluna da matriz *G*':

$$G' = \begin{bmatrix} 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0 & 1 \\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0 & 1 \\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 & 1 \\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 & 1 \\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0 & 0 \\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0 & 0 \\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 & 1 \\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 & 0 \\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 & 0 \\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0 & 1 \\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0 & 0 \\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{2.26}$$

Note que:

$mnu = 7$

f) Agora serão colocadas à esquerda da última coluna de G', $mnu = 7$ colunas da submatriz de identidade de G, levando-se em conta que a posição dos uns dessas colunas devem coincidir com a posição dos uns da última coluna de G'. Esta matriz fica da seguinte forma:

$$G' = \begin{bmatrix} 1\ 1\ 0\ 1\ 1\ 0\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 \\ 1\ 1\ 1\ 0\ 0\ 1\ 0\ 0 & 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1 \\ 1\ 0\ 1\ 0\ 1\ 0\ 1\ 0 & 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1 \\ 1\ 0\ 0\ 1\ 1\ 1\ 0\ 0 & 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1 \\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ 0 & 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1 \\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0 & 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1 \\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0 & 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0 & 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \end{bmatrix} \tag{2.27}$$

Da matriz anterior e tendo em conta a equação (2.15), tem-se que:

$$N(S_2) = mnu + 1 = 8 \tag{2.28}$$

e da equação (2.16),

$$\text{Min}R_{END} = N(S_2) - 1 = 7 \tag{2.29}$$

g) A seguir, a matriz *G*' é completada com as colunas restantes não utilizadas de *G*:

$$G' = \begin{bmatrix}
1&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1\\
1&1&1&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&1\\
1&0&1&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1\\
1&0&0&1&1&1&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&1\\
1&0&0&1&0&1&1&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&1&0&1&1&0&1&0&0&1&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&1&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&1&1&0&0&0&1&0&1&1&1&0&0&0&0&0&0&0&0\\
0&1&0&1&0&1&1&0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&1\\
1&1&1&0&0&0&1&0&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1
\end{bmatrix} \tag{2.30}$$

h) Por último, da equação (2.1) tem-se que:

$$\text{Min}RL_{MAX} = \text{MAX}\{7+7, 12\} = 14 \tag{2.31}$$

Através de busca computacional é simples achar um líder de classe lateral **m** que produz um código com os limitantes anteriores, sendo um deles:

$$\mathbf{m} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \tag{2.32}$$

O algoritmo possibilita a obtenção do vetor modificador **m** com a menor quantidade possível de uns. Isto faz com que o aumento da complexidade do codificador seja mínimo toda vez que o número de uns no vetor **m** determina o número de somadores módulo-2 que serão incorporados ao codificador.

2.5 CAPACIDADE DE CORREÇÃO E/OU DETECÇÃO DE ERROS DOS CÓDIGOS DE BLOCO COM "RUNLENGTH" LIMITADO.

Uma vez encontrado o líder de classe lateral adequado para produzir o código com "runlength" limitado, a codificação da seqüência de informação pode ser feita da maneira convencional, mas antes da transmissão das palavras-código, adiciona-se o vetor **m** a cada uma delas, garantindo-se que a seqüência de dados enviada pelo canal esteja limitada no "runlength". No receptor é feita a operação inversa e, portanto, antes da correção de erros, a palavra recebida é modificada para ser entregue ao decodificador. A forma de eliminar o efeito do vetor modificador na recepção dependerá do tipo de decisão utilizada na decodificação (decisão suave ou decisão abrupta) como será visto no próximo capítulo.

Sejam

$c(x)$: polinômio que representa a palavra-código,

$u(x)$: polinômio de grau menor do que k, que representa a seqüência de informação,

$m(x)$: polinômio que representa o vetor modificador,

então, as palavras-código geradas pelos códigos de bloco com "runlength" limitado podem ser representadas como:

$$c(x) = u(x)g(x) + m(x), \tag{2.33}$$

Devido à linearidade dos códigos originais, as propriedades de correção e/ou detecção de erros destes códigos vão ser preservadas. Se $c(x)$ é transmitido e $r(x)=c(x)+e(x)$ é recebido, então, supondo que a decodificação seja feita com decisão abrupta, no receptor o primeiro passo feito é a subtração de $m(x)$ do polinômio recebido $r(x)$. Como resultado é obtido o polinômio $u(x)g(x)+e(x)$ e a correção de erros pode ser feita com o decodificador do código original.

2.6 RESULTADOS OBTIDOS

Na Tabela 2.3 são mostrados os resultados obtidos na modificação de alguns códigos cíclicos transparentes conhecidos.

Os limitantes inferiores de MinR_{START}, MinR_{END} e MinRL_{MAX} são os valores mínimos possíveis de serem encontrados para esses códigos.

As posições dos uns do vetor modificador representam as posições que ocupam os dígitos binários "1" no vetor modificador **m** de n bits, onde $\mathrm{m}=m_1m_2...m_n$. Por exemplo, no caso particular do código BCH (15, 5, 7) as posições representadas por (4, 10, 12) significam que o vetor **m** tem uns nas posições m_4, m_{10}, e m_{12} ou seja, $\mathbf{m}=\begin{bmatrix}0&0&0&1&0&0&0&0&0&1&0&1&0&0&0\end{bmatrix}$.

Tabela 2.3: Códigos Cíclicos Transparentes Modificados.

Código	MinR_{START}	MinR_{END}	MinRL_{MAX}	Posições dos uns do Vetor Modificador
BCH (15, 5, 7)	3	3	6	(4, 10, 12)
BCH (15, 7, 5)	3	3	7	(1, 9, 15)
BCH (15, 11, 3)	7	10	17	(5)
BCH (31, 16, 7)	7	7	16	(8, 22, 24)
BCH (31, 21, 5)	11	20	31	(10, 13, 22)
BCH (31, 26, 3)	15	22	37	(10)
GOLAY (23, 12, 7)	7	7	14	(8, 16)
(15, 7, 3)	3	3	7	(2, 7, 13)
(15, 9, 3)	5	9	14	(6)
(15, 9, 4)	5	10	15	(5)
(21, 12, 5)	7	7	14	(6, 14)

O algoritmo descrito é mais simples do que o apresentado em [Popplewell, 1992], pois é baseado no reordenamento da matriz geradora do código na forma sistemática sem ter que utilizar a matriz de verificação de paridade. Por isso, a sua implementação é fácil de ser feita através de programas computacionais.

2.7 ALGORITMO PARA A TRANSFORMAÇÃO DE EXPANSÕES BINÁRIAS DE CÓDIGOS DE REED-SOLOMON

Da Tabela 2.3 pode-se apreciar que na medida em que o comprimento do código aumenta, os valores de MinRL_{MAX} também vão aumentando e atingindo valores inadequados para aplicações práticas. Daí pode-se concluir que a utilização de códigos de bloco modificados visando a obtenção de valores práticos para o número de símbolos consecutivos

iguais na seqüência codificada fica restringida a códigos de pequeno comprimento e, portanto, com capacidade de correção de erro limitada.

O algoritmo apresentado na Secção 2.4 foi utilizado em códigos de Reed-Solomon como códigos originais. Especificamente, foram utilizadas expansões binárias destes códigos, já que os códigos de Reed-Solomon propriamente não são binários [Fernández, 1998].

Os códigos de Reed-Solomon são códigos não binários cujos símbolos pertencem a um corpo de Galois GF(q), onde q é uma potência de um número primo. Um código de Reed-Solomon com capacidade de correção de t erros é definido pelos seguintes parâmetros:

$$\begin{aligned} n &= q-1 \\ n-k &= 2t \\ d_{\min} &= 2t+1 \end{aligned} \tag{2.34}$$

O polinômio gerador deste código está dado por:

$$g(x) = \left(x-\alpha^{m_0}\right)\left(x-\alpha^{m_0+1}\right)\cdots\left(x-\alpha^{m_0+d_{\min}-2}\right) \tag{2.35}$$

onde α é um elemento primitivo de GF(q) e m_0 um inteiro qualquer.

Tomando $q = 2^m$, onde m é um inteiro, os símbolos do código podem ser representados por m-uplas binárias. Desta forma, os códigos de Reed-Solomon podem ser expandidos em códigos lineares binários com os seguintes parâmetros [Lin, 1983]:

$$\begin{aligned} n &= m\left(2^m-1\right) \\ n-k &= 2mt \end{aligned} \tag{2.36}$$

Estes códigos são capazes de corrigir qualquer padrão de erro que afete *t* ou menos símbolos de *m* bits, portanto, eles são adequados para a correção de erros em surtos.

Os códigos construídos desta forma são lineares e, em muitos casos, transparentes, portanto eles têm "runlength" ilimitado devido à inclusão neles das palavras toda zeros e toda uns.

2.7.1 MODIFICAÇÃO DE UMA EXPANSÃO BINÁRIA DO CÓDIGO DE REED-SOLOMON (7, 5, 3)

Sejam $\mathrm{GF}(2^3)$ um corpo de Galois construído a partir do polinômio primitivo $1+x+x^3$ com coeficientes em $\mathrm{GF}(2)$ e α^2 um elemento primitivo deste corpo (o conjugado da raiz α do polinômio anterior em $\mathrm{GF}(2^3)$). Como α^2 é também uma raiz deste polinômio, então:

$$1+\alpha^2+\alpha^6=0 \quad \text{ou} \quad \alpha^2=1+\alpha^6 \tag{2.37}$$

Assim, os elementos do corpo podem ser gerados a partir do elemento primitivo α^2 e da base $1, \alpha, \alpha^6$ da seguinte forma:

$0 \qquad 0$

$1 \qquad 1$

$\alpha \qquad \alpha$

$\alpha^6 \qquad \alpha^6$

$\alpha^2 \qquad 1+\alpha^6$

$\alpha^3 \qquad \alpha^2\alpha = (1+\alpha^6)\alpha = \alpha+\alpha^7 = 1+\alpha$

$\alpha^5 \qquad \alpha^2\alpha^3 = (1+\alpha^6)(1+\alpha) = 1+\alpha+\alpha^6+\alpha^7 = \alpha+\alpha^6$

$\alpha^4 \qquad \alpha^2\alpha^2 = (1+\alpha^6)(1+\alpha^6) = 1+\alpha^6+\alpha^6+\alpha^{12} = 1+\alpha^5 = 1+\alpha+\alpha^6$

Portanto, o mapeamento dos símbolos sobre $\mathrm{GF}(2^3)$ em sua representação binária é:

$0 \qquad 000$

$1 \qquad 100$

$\alpha \qquad 010$

$\alpha^2 \qquad 101$

$\alpha^3 \qquad 110$

$\alpha^4 \qquad 111$

$\alpha^5 \qquad 011$

$\alpha^6 \qquad 001$

Considere agora o código RS (7, 5, 3) sobre $\mathrm{GF}(2^3)$ com o polinômio gerador,

$$g_1(x) = (x+\alpha^5)(x+\alpha^6) = \alpha^4 + \alpha x + x^2 \tag{2.38}$$

As expansões binárias das palavras deste código constituem as palavras do código cíclico (21, 15, 3) gerado pelo polinômio,

$$g_2(y) = 1 + y + y^2 + y^4 + y^6 \tag{2.39}$$

O próprio $g_1(x)$ é expandido no vetor

$$111,\ 010,\ 100,\ 000,\ 000,\ 000,\ 000 \tag{2.40}$$

que é exatamente $g_2(y)$. Mais ainda, $\alpha g_1(x)$ é mapeado em $y g_2(x), \alpha^2 g_1(x)$ em $y^2 g_2(y)$ e assim por diante. Este é o único caso conhecido em que a expansão binária de um código de Reed Solomon forma um código cíclico [MacWilliams, 1977].

O código gerado por $g_2(y)$ é um código cíclico transparente, portanto pode ser transformado num código com "runlength" limitado. A matriz geradora deste código na forma sistemática é:

$$G = \begin{bmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1\\
0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&1&1&1\\
0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&1
\end{bmatrix} \tag{2.41}$$

Usando o algoritmo de modificação, a matriz modificada para a geração de um código que apresente um mínimo valor do "runlength" é dada por:

$$\mathrm{G}' = \begin{bmatrix}
1&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&1&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&1&0&0\\
1&0&1&0&0&0&0&0&1&1&1&0&1&0&0&0&0&0&0&0&0\\
1&1&1&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&1&0&0\\
0&0&1&0&1&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0\\
0&1&1&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&0&0&0&0&1&0&0&0&0&0\\
1&0&1&0&0&0&0&0&1&1&0&0&0&0&0&0&0&1&0&0&0\\
0&1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0
\end{bmatrix} \tag{2.42}$$

Note da matriz $\mathbf{G}'$ que $\mathrm{Min}R_{START} = 7$ e $\mathrm{Min}R_{END} = 15$, portanto, $\mathrm{Min}RL_{MAX} = 22$. Um líder de "coset" apropriado a partir de $\mathbf{G}'$ para atingir este limitante é:

$$\mathbf{m} = \begin{bmatrix}0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\end{bmatrix} \tag{2.43}$$

2.8 GENERALIZAÇÃO DO ALGORITMO

É possível obter expansões binárias de códigos de Reed-Solomon que não sejam códigos cíclicos, mas sim lineares e transparentes por meio da seleção adequada da base a ser utilizada na expansão.

Usando a base polinomial $\left\{1, \alpha, \alpha^2, \ldots, \alpha^{m-1}\right\}$ para expandir um código de Reed-Solomon, pode ser obtida uma base sistemática para a expansão binária com o procedimento a seguir:

a) Formar um bloco de *km* palavras de informação formadas por vetores de *k* símbolos de $\mathrm{GF}(2^m)$. O bloco é formado da seguinte maneira:

$$
\begin{array}{cccccccccc}
1, & 0, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
\alpha, & 0, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
. & . & \ldots & . & . & . & . & . & . & . \\
. & . & \ldots & . & . & . & . & . & . & . \\
. & . & \ldots & . & . & . & . & . & . & . \\
\alpha^{m-1}, & 0, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 1, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & \alpha, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
. & . & \ldots & . & . & . & . & . & . & . \\
. & . & \ldots & . & . & . & . & . & . & . \\
. & . & \ldots & . & . & . & . & . & . & . \\
0, & \alpha^{m-1}, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\
0, & 0, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & 1 \\
0, & 0, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & \alpha \\
. & . & \ldots & . & . & . & . & . & . & . \\
. & . & \ldots & . & . & . & . & . & . & . \\
. & . & \ldots & . & . & . & . & . & . & . \\
0, & 0, & \ldots & 0, & 0, & 0, & 0, & 0, & 0, & \alpha^{m-1}
\end{array}
$$

b) Codificar as palavras de cada linha do bloco anterior utilizando um codificador RS sistemático.

c) Representar os símbolos das palavras-código obtidas em forma binária. Os vetores obtidos desta forma constituem uma matriz geradora na forma sistemática para um código binário linear.

d) Obter o código modificado.

Por exemplo, considere-se o código RS (3, 2, 2) sobre $\mathrm{GF}(2^2)$. Fazendo a codificação sistemática dos vetores,

$$
\begin{array}{cc}
1 & 0 \\
\alpha & 0 \\
0 & 1 \\
0 & \alpha
\end{array}
$$

a representação binária das palavras obtidas formam a seguinte matriz geradora para o código binário (6, 4, 2):

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{2.44}$$

Note da matriz **G** que o número de "1's" das duas colunas de verificação de paridade é ímpar, portanto, o código é transparente.

A matriz modificada é dada por:

$$\mathbf{G}' = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{2.45}$$

Somando (modulo-2) o vetor $\mathbf{m} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ a todas as palavras-código geradas pela matriz **G′**, os limitantes obtidos são: $\text{Min}R_{START} = 3$, $\text{Min}R_{END} = 3$ e $\text{Min}R_{MID} = 4$. Portanto, o número máximo de 1's (ou 0's) consecutivos em qualquer seqüência codificada não será maior do que 6.

Com esta aproximação, os códigos obtidos a partir de expansões binárias de códigos de Reed-Solomon podem ser usados para corrigir todos os surtos binários de cumprimento até $m(t-1)+1$ bits [Lin, 1983]. Se os surtos têm cumprimento menor, estes códigos ainda podem corrigir outros erros aleatórios. Desta forma podem ser obtidos códigos com alta capacidade de correção de erro e capazes de combater erros em surtos, os quais estão sempre presentes nas classes de canais sob análise. Não obstante, nas simulações efetuadas foi constatado que as restrições de "runlength" atingem valores altos demais na medida em que são utilizados

códigos de maior comprimento. Por exemplo, para uma expansão binária do código RS (15, 9, 3), o limitante inferior do "runlength" obtido foi $\text{Min}RL_{MAX} = 67$.

2.9 CONCLUSÃO

Neste capítulo foram apresentados algoritmos para transformar códigos de bloco corretores de erro conhecidos em códigos com "runlength" limitado. Estes algoritmos conseguem determinar os melhores valores das restrições de "runlength" para um código determinado com um aumento mínimo da complexidade na codificação e decodificação. Isto permite a construção de esquemas de codificação em que só um código realiza as funções de controle de erro e limitação do "runlength". Sem dúvida, para que as restrições de "runlength" tenham valores práticos, estes códigos devem ser de pequeno comprimento. Isto limita grandemente a capacidade de correção de erros possível de se alcançar.

Uma forma de resolver este problema consiste na utilização de códigos concatenados utilizando códigos corretores de erro com "runlength" limitado como códigos internos. A maior parte dos esquemas utilizados na atualidade em canais com restrições utiliza códigos internos só para satisfazer as restrições do canal, ficando a correção de erros sob responsabilidade do código externo. Isto faz com que o código externo tenha que ser capaz de lidar ainda com os erros introduzidos pelo próprio decodificador interno. Mais ainda, os códigos internos tradicionalmente usados utilizam inserção de bits para manter o valor do "runlength" dentro dos limites quando se justapõem duas palavras-código, o que reduz a eficiência do código. Com a utilização de códigos corretores de erro com "runlength" limitado

como códigos internos pode-se aumentar a eficiência na codificação respeito aos esquemas convencionais.

No próximo capítulo será vista a construção de esquemas de codificação concatenados para este tipo de canais bem como a obtenção de algoritmos de decodificação eficientes para os códigos propostos.

CAPÍTULO 3

CÓDIGOS DE BLOCO CONCATENADOS COM "RUNLENGTH" LIMITADO

3.1 INTRODUÇÃO

No capítulo anterior, algoritmos de modificação foram utilizados para a obtenção de seqüências codificadas com "runlength" limitado a partir de códigos de bloco corretores de erro conhecidos e de expansões binárias de códigos de Reed-Solomon. Em ambos os casos, os melhores códigos obtidos têm comprimento relativamente pequeno, pois, para códigos

maiores, os limitantes inferiores obtidos para o máximo "runlength" (RL_{MAX}), são muitos altos para aplicações práticas.

A partir daí, neste capítulo será proposta a obtenção de códigos com restrições teoricamente de qualquer comprimento e capacidade de correção de erro a partir de esquemas de codificação concatenados.

3.2 CÓDIGOS DE BLOCO CONCATENADOS

Um código concatenado é formado pela combinação de dois códigos distintos visando a obtenção de um código de comprimento maior. Usualmente um dos códigos é não binário e o outro é binário. Os dois códigos são concatenados como mostrado na Figura 3.1:

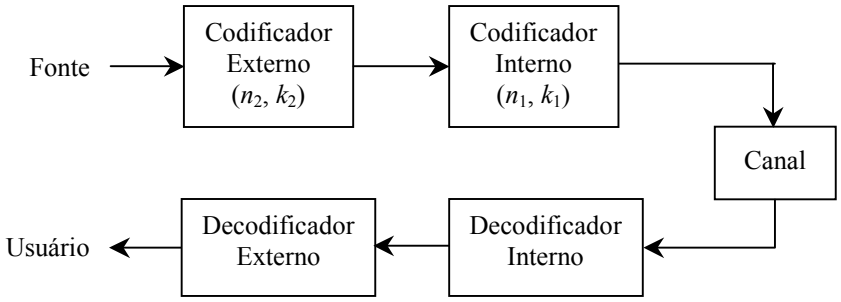

Figura 3.1: Diagrama de Blocos de um Sistema de Codificação Concatenado

O código externo é constituído pelo código não binário (n_2, k_2) e o código interno pelo código binário (n_1, k_1). As palavras-código são formadas subdividindo blocos de k_1k_2 bits de informação em k_2 grupos, chamados de símbolos, onde cada símbolo está composto por k_1 bits. Estes k_2 símbolos são codificados pelo codificador externo obtendo-se na sua saída n_2

símbolos de k_1 bits. A partir daí, o codificador interno codifica cada símbolo de k_1 bits em blocos de n_1 bits. Desta forma, no final do processo é obtido um código de bloco concatenado tendo um comprimento de bloco de $n_1 n_2$ bits e contendo $k_1 k_2$ bits de informação.

A distância mínima do código concatenado é dada por $d_{1\min} d_{2\min}$ onde $d_{1\min}$ é a distância mínima do código interno e $d_{2\min}$ é a distância mínima do código externo. Mais ainda, a taxa do código concatenado é dada por $k_1 k_2 / n_1 n_2$, sendo igual ao produto das taxas individuais de cada um dos códigos do esquema.

A decodificação com decisão abrupta de um código concatenado é feita separadamente por meio dos decodificadores dos códigos interno e externo. O decodificador interno aceita a decisão abrupta feita pelo receptor sobre cada bloco de n_1 bits, correspondente a uma palavra do código interno, e faz a decisão sobre os k_1 bits de informação baseada numa estratégia de decodificação de máxima verossimilhança (distância mínima). Estes k_1 bits conformam um símbolo de uma palavra-código do código externo. Quando um bloco de n_2 símbolos de k_1 bits é recebido do decodificador interno, o decodificador externo faz a decisão sobre os k_2 símbolos de k_1 bits baseada também numa estratégia de decodificação de máxima verossimilhança.

A decodificação com decisão suave também é possível de se fazer com um código concatenado. Geralmente, a decodificação com decisão suave é realizada no código interno, se este é selecionado de tal forma que seu número de palavras-código seja pequeno, ou seja, se 2^{k_1} não é muito grande. O código externo geralmente é decodificado através de decodificação com decisão abrupta especialmente se o comprimento do código e o número de palavras-código são grandes.

A concatenação de códigos permite obter, a partir de códigos relativamente pequenos, sistemas de codificação com os quais pode-se obter probabilidades de erro muito baixas e grandes ganhos de codificação reduzindo-se a complexidade da decodificação [Kasami, 1997].

3.3 ESQUEMA DE CODIFICAÇÃO CONCATENADA COM "RUNLENGTH" LIMITADO

Tomando como base os códigos apresentados no Capítulo 2, nesta secção serão construídos códigos concatenados baseados em um código não binário (especificamente um código de Reed-Solomon) como código externo e um código com "runlength" limitado como código interno. O esquema de codificação proposto está mostrado na Figura 3.2:

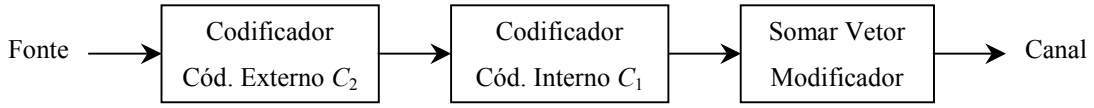

Figura 3.2: Esquema de Codificação Concatenado

O código interno C_1 é um código equivalente de um código binário, linear e transparente (n_1, k_1) obtido segundo o algoritmo apresentado no Capítulo 2. O código externo C_2 é um código não binário (n_2, k_2) de Reed-Solomon com símbolos em $\mathrm{GF}\left(2^{k_1}\right)$ os quais são representados por seqüências binárias de k_1 bits. O processo de codificação é feito em três etapas. Primeiramente, os bits de informação na saída da fonte são divididos em k_2 símbolos de k_1 bits cada. Esses k_2 símbolos são codificados por um codificador de Reed-Solomon, obtendo-se uma palavra de n_2 símbolos do código externo C_2. Na próxima etapa, cada símbolo

de k_1 bits das palavras do código C_2 é codificado numa palavra do código C_1. Finalmente, um vetor modificador é somado a cada palavra-código de C_1 para se obter uma seqüência de n_2 palavras-código modificadas de C_1 com "runlength" limitado. Assim, o código resultante é um código binário (n_1n_2, k_1k_2).

Antes de ser transmitida, a seqüência codificada deve ser mapeada em um tipo de sinal adequado ao canal de comunicações. Neste caso, tratando-se de canais com restrições, é conveniente tentar eliminar o conteúdo de corrente contínua no espectro da seqüência codificada. Assim, a seqüência de bits de saída do codificador interno foi mapeada num sinal antipodal de banda básica fazendo corresponder ao bit "zero" o sinal $+1$ e ao bit "um" o sinal -1. Isto se consegue através da operação $y = 1 - 2x,$ para $x \in \{0,1\}$.

Como os códigos externos foram escolhidos como códigos de Reed-Solomon devido a que estes possuem uma estrutura algébrica bem definida além de atingirem o limitante de Singleton com a igualdade ($n - k = d_{\min} - 1$), ou seja, atingem o maior valor possível de distância entre palavras-código [MacWilliams, 1977]. Ademais, existem vários algoritmos de codificação e decodificação eficientes para estes códigos como, por exemplo, o algoritmo de Berlekamp-Massey e o algoritmo Euclidiano. Devido ao fato destes códigos serem não binários, os mesmos proporcionam uma capacidade de correção de erros em surto significativa. A principal limitação dos códigos de Reed-Solomon reside na dificuldade existente para se encontrar algoritmos eficientes de decodificação com decisão suave para eles devido principalmente à incompatibilidade entre a estrutura algébrica dos corpos finitos, sob os quais estes códigos são construídos, e os valores reais do sinal na saída do canal na recepção.

O código com "runlength" limitado é obtido a partir do algoritmo apresentado no Capítulo 2. O valor máximo do "runlength" deste código é obtido modificando a matriz geradora sistemática de um código corretor de erro binário linear e transparente conhecido e adicionando um líder de "coset" apropriado a todas as palavras-código geradas pela matriz geradora modificada. A modificação é feita através de permutações de colunas da matriz geradora na forma sistemática do código C_1 original para se obter um limitante inferior para a restrição *k*. Devido à linearidade do código original, a distância de Hamming mínima, e, portanto, a capacidade de correção de erro deste código é preservada. Teoricamente qualquer código de bloco linear e transparente pode ser transformado em um código com "runlength" limitado [Popplewell, 1992] mas, valores práticos de MinRL_{MAX} oscilam entre 6 e 14 [Immink, 1994]. É por isso que os códigos internos utilizados neste trabalho são de pequeno comprimento. A Tabela 3.1 mostra alguns dos códigos que podem ser utilizados como códigos internos. Nela, o vetor modificador está representado pela posição dos seus "1's", da esquerda para a direita.

Tabela 3.1: Códigos com Runlength Limitado

Códigos	Taxa	MinRL_{MAX}	Vetor Modificador
BCH (7, 4, 3)	0,571	7	(6)
BCH (15, 5, 5)	0,333	6	(10, 12, 14)
BCH (15, 7, 5)	0,467	7	(1, 9, 15)
Golay (23, 12, 7)	0,522	14	(18, 16)
RM (8, 4, 4)	0,5	6	(1, 5)
RM (16, 5, 8)	0,312	6	(1, 7, 13)

A partir destes códigos e suas possíveis combinações com códigos de Reed-Solomon, pode-se obter um grande número de códigos concatenados com valor de MinRL_{MAX} iguais aos correspondentes valores listados na tabela para os códigos internos. Dentre essas combinações foram selecionadas para simulação aquelas que garantem uma capacidade de correção de erro adequada, sem que a taxa do código diminua muito. Na Tabela 3.2 são mostrados os códigos concatenados construídos.

Tabela 3.2: Códigos Concatenados

Código Externo	Código Interno	Código Concatenado	Taxa	MinRL_{MAX}
RS (15, 11, 5)	BCH (7, 4, 3)	(105, 44, 15)	0,42	7
RS (15, 9, 7)	BCH (7, 4, 3)	(105, 36, 21)	0,34	7
RS (31, 29, 3)	BCH (15, 5, 7)	(465, 145, 21)	0,31	6
RS (31, 27, 5)	BCH (15, 5, 7)	(465, 135, 35)	0,29	6
RS (31, 25, 7)	BCH (15, 5, 7)	(465, 125, 49)	0,26	6
RS (127, 109, 19)	BCH (15, 7, 5)	(1905, 763, 95)	0,4	7
RS (127, 107, 21)	BCH (15, 7, 5)	(1905, 749, 105)	0,393	7
RS (15, 11, 5)	RM (8, 4, 4)	(120, 44, 20)	0,367	6
RS (15, 9, 7)	RM (8, 4, 4)	(120, 36, 28)	0,3	6
RS (31, 29, 3)	RM (16, 5, 8)	(496, 145, 24)	0,29	6
RS (31, 25, 7)	RM (16, 5, 8)	(496, 125, 56)	0,25	6
RS (15, 11, 5)	Golay (23, 12, 7)	(345, 132, 35)	0,38	14
RS (15, 9, 7)	Golay (23, 12, 7)	(345, 108, 49)	0,31	14

3.3.1 CODIFICAÇÃO DO CÓDIGO EXTERNO

Os códigos externos utilizados nos códigos concatenados listados na Tabela 3.2 em todos os casos são códigos de Reed-Solomon primitivos, codificados de forma sistemática a partir do polinômio gerador correspondente. As características principais destes códigos são:

- **Código RS (15, 11, 5)**
  - Símbolos sobre $\mathrm{GF}(2^4)$ construído a partir do elemento primitivo α e do polinômio primitivo $p(x) = 1 + x + x^4$.
  - Polinômio gerador: $g(x) = \alpha^{10} + \alpha^3 x + \alpha^6 x^2 + \alpha^{13} x^3 + x^4$.

- **Código RS (15, 9, 7)**
  - Símbolos sobre $\mathrm{GF}(2^4)$ construído a partir do elemento primitivo α e do polinômio primitivo $p(x) = 1 + x + x^4$.
  - Polinômio gerador: $g(x) = \alpha^6 + \alpha^9 x + \alpha^6 x^2 + \alpha^4 x^3 + \alpha^{14} x^4 + \alpha^{10} x^5 + x^6$.

- **Código RS (31, 29, 3)**
  - Símbolos sobre $\mathrm{GF}(2^5)$ construído a partir do elemento primitivo α e do polinômio primitivo $1 + x^2 + x^5$.
  - Polinômio gerador: $g(x) = \alpha^3 + \alpha^{19} x \ + x^2$

- **Código RS (31, 27, 5)**
  - Símbolos sobre $\mathrm{GF}(2^5)$ construído a partir do elemento primitivo α e do polinômio primitivo $1+x^2+x^5$.
  - Polinômio gerador: $g(x)=\alpha^{10}+\alpha^{29}x+\alpha^{19}x^2+\alpha^{24}x^3+x^4$

- **Código RS (31, 25, 7)**
  - Símbolos sobre $\mathrm{GF}(2^5)$ construído a partir do elemento primitivo α e do polinômio primitivo $1+x^2+x^5$.
  - Polinômio gerador: $g(x)=\alpha^{21}+\alpha^{24}x+\alpha^{16}x^2+\alpha^{24}x^3+\alpha^{9}x^4+\alpha^{10}x^5+x^6$

- **Código RS (127, 109, 19)**
  - Símbolos sobre $\mathrm{GF}(2^7)$ construído a partir do elemento primitivo α e do polinômio primitivo $1+x^3+x^7$.
  - Polinômio gerador:

$$g(x)=\alpha^{44}+\alpha^{83}x+\alpha^{73}x^2+\alpha^{38}x^3+\alpha^{89}x^4+\alpha^{126}x^5+\alpha^{63}x^6+\alpha^{76}x^7+\alpha^{105}x^8+\alpha^{88}x^9+\alpha^{86}x^{10}+\alpha^{38}x^{11}+\alpha^{6}x^{12}+\alpha^{50}x^{13}+\alpha^{121}x^{14}+\alpha^{51}x^{15}+\alpha^{67}x^{16}+\alpha^{58}x^{17}+x^{18}$$

- **Código RS (127, 107, 21)**
  - Símbolos sobre $\mathrm{GF}(2^7)$ construído a partir do elemento primitivo α e do polinômio primitivo $1+x^3+x^7$.
  - Polinômio gerador:

$$g(x) = \alpha^{83} + \alpha^{106}x + \alpha^{4}x^{2} + \alpha^{67}x^{3} + \alpha^{122}x^{4} + \alpha^{12}x^{5} + \alpha^{46}x^{6} + \alpha^{35}x^{7} + \alpha^{119}x^{8} + \alpha^{91}x^{9} + \alpha^{77}x^{10} + \alpha^{70}x^{11} + \alpha^{77}x^{12} + \alpha^{99}x^{13} + \alpha^{89}x^{14} + \alpha^{34}x^{15} + \alpha^{123}x^{16} + \alpha^{47}x^{17} + \alpha^{90}x^{18} + \alpha^{44}x^{19} + x^{20}$$

3.3.2 CODIFICAÇÃO DO CÓDIGO INTERNO

Os códigos internos do tipo BCH mostrados na Tabela 3.2 e utilizados nas simulações foram codificados a partir das matrizes geradoras obtidas em [Fernández, 1997-1]. Estas matrizes produzem códigos equivalentes a códigos BCH sistemáticos primitivos. As características principais destes códigos são:

- **Código BCH (7, 4, 3)**

  - Matriz geradora:

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \tag{3.1}$$

  - Vetor modificador: $\mathbf{m} = [0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0]$

- **Código BCH (15, 5, 7)**

  - Matriz geradora:

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

- Vetor modificador: $\mathbf{m} = [0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0]$

- **Código BCH (15, 7, 5)**

  - Matriz geradora:

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

  - Vetor modificador: $\mathbf{m} = [1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1]$

No caso particular de códigos de Reed-Müller, devido às suas propriedades geométricas, as suas matrizes geradoras não precisam ser modificadas para cumprir com o teorema enunciado por Popplewell et. al. [Poplewell, 1992].

- **Código RM (8, 4, 4)**

  - Matriz geradora:

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.4}$$

  - Vetor modificador: $\mathbf{m} = [1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0]$

- **Código RM (16, 5, 8)**

  - Matriz Geradora

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.5}$$

  - Vetor modificador: $\mathbf{m} = [1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0]$

Em todos os casos o vetor modificador foi selecionado como sendo o líder de "coset" com menor quantidade de "uns" que garante que a seqüência de bits enviada pelo canal atinja o valor de MinRL_{MAX} indicado na Tabela 3.2. Isto faz com que o aumento na complexidade do codificador seja o menor possível.

3.4 ALGORITMOS DE DECODIFICAÇÃO EFICIENTES

O processo de decodificação dos códigos concatenados pode ser realizado de duas formas: utilizando decisão abrupta em ambos os códigos ou realizando decisão abrupta no código externo e decisão suave no código interno. Foram feitas simulações das duas formas de decodificação visando obter o algoritmo de decodificação mais eficiente possível. De forma geral, o processo de decodificação também é realizado em três etapas como mostrado na Figura 3.3.

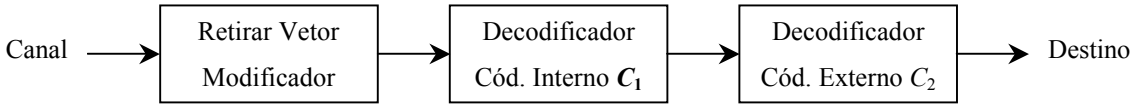

Figura 3.3: Decodificação dos Códigos Concatenados

Primeiramente, o vetor modificador é retirado de cada palavra modificada do código C_1. A forma de se fazer esta operação varia de acordo com o tipo de decisão feita no receptor sobre o sinal na saída do canal, como será visto mais adiante. Na etapa seguinte, são decodificadas n_2 palavras do código C_1 obtendo-se uma seqüência de n_2k_1 bits que corresponde ao comprimento (em bits) de uma palavra do código externo. Esta decodificação é feita de tal forma que, se o número de bits errados recebidos ultrapassa a capacidade de correção de erro do código interno, o valor das posições correspondentes aos k_1 bits, de informação é transferido inalteradamente ao decodificador do código externo. Finalmente, esta seqüência é decodificada pelo decodificador do código externo.

3.4.1 DECODIFICAÇÃO DO CÓDIGO INTERNO

Como foi dito anteriormente, no receptor pode ser feita decisão abrupta ou suave sobre o sinal de saída do canal. Dependendo do tipo de decisão feita, o algoritmo de decodificação do código interno será diferente. Ambos os casos foram objeto de simulação, como será visto a seguir.

3.4.1.1 Decodificação com Decisão Abrupta

Neste caso, a quantização do sinal de saída do canal é feita com um bit de precisão, ou seja, se o sinal recebido é menor do que 0 volts, a decisão é feita pelo bit 1 e se é maior do que 0 volts, a decisão é feita pelo bit 0. Em seguida, o vetor modificador é somado (módulo 2) a cada palavra-código recebida. A partir daí, algoritmos de decodificação algébricos podem ser usados para a decodificação do código interno.

Quando da utilização de códigos BCH como códigos internos, a decodificação foi feita através do algoritmo de Berlekamp-Massey para códigos BCH primitivos sistemáticos. Isto é possível de se fazer pois os códigos modificados são equivalentes a códigos BCH sistemáticos, portanto só existirão diferenças na posição dos bits de informação nas palavras-código recebidas, com respeito à posição que teriam se a codificação tivesse sido feita de forma sistemática. Para isso, é necessário armazenar a posição dos bits de informação $\mathbf{u} = \{u_1, u_2, ..., u_{k_1}\}$ nas palavras codificadas **c**, onde $\mathbf{c} = \{c_1, c_2, ..., c_{n_1}\}$. Os parâmetros necessários para a execução do algoritmo de decodificação são colocados a seguir para cada um dos códigos BCH utilizados como códigos internos:

- **Código BCH (7, 4, 3)**
  - Posição dos bits de informação: **u** = {c_6, c_7, c_3, c_4}
  - Polinômio Gerador: $g(x) = 1 + x + x^3$

- **Código BCH (15, 5, 7)**
  - Posição dos bits de informação: **u** = {c_{14}, c_4, c_{13}, c_5, c_{12}}

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

- **Código BCH (15, 7, 5)**
  - Posição dos bits de informação: **u** = $\{c_5, c_6, c_{14}, c_4, c_{13}, c_{12}, c_7\}$
  - Polinômio Gerador: $g(x) = 1 + x^4 + x^6 + x^7 + x^8$

No caso do código de Reed-Müller (8, 4, 4) foi feita decodificação por síndrome, tendo em conta que este é um código de comprimento pequeno. Para isso, foi utilizada a seguinte matriz de verificação de paridade:

$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Note que para este código $G = H$, pois este é um código auto dual. Como este código pode corrigir $2^{n-k} = 16$ padrões de erro, foi elaborada a seguinte tabela de decodificação, onde estão contemplados os oito padrões de erro correspondentes a erros em uma posição mais sete padrões de erros correspondentes a erros em duas posições além do padrão todo "zeros".

Tabela 3.3: Tabela de Decodificação do Código RM (8, 4, 4)

Síndromes	Padrões de Erro
0000	00000000
1000	10000000
1001	01000000
1010	00100000
1011	00010000
1100	00001000
1101	00000100
1110	00000010
1111	00000001
0001	11000000
0010	10100000
0011	10010000
0100	10001000
0101	10000100
0110	10000010
0111	10000001

A Figura 3.4 mostra o desempenho destes códigos considerando o canal afetado por ruído branco Gaussiano, através das curvas de probabilidade de erro de bit em função da relação sinal-ruído (nesta relação foi considerada a energia de bits de informação transmitidos).

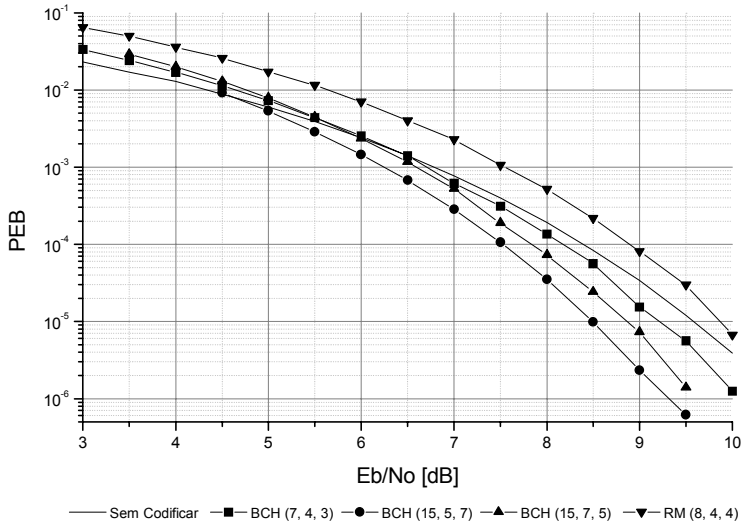

Figura 3.4: Curvas de Desempenho dos Códigos Internos com Decisão Abrupta

Na figura anterior foi incluída a curva do canal gaussiano não codificado como referência para comparação.

Pode-se notar neste caso que, devido ao pequeno comprimento e, portanto, à pequena capacidade de correção de erro do código RM (8, 4, 4) junto ao uso de decodificação por síndrome para este código, a curva correspondente a ele vai cortar a curva do canal gaussiano não codificado para valores de relação sinal-ruído acima de 10 dB. Para valores pequenos de relação sinal-ruído, os ganhos de codificação apresentados pelos códigos internos com a utilização de decodificação com decisão abrupta, são pequenos.

A decodificação do código externo de Reed-Solomon foi feita através do algoritmo de Berlekamp-Massey como descrito em [Lin, 1983]. Este é um dos algoritmos mais eficientes para este tipo de código.

As figuras a seguir mostram o desempenho dos códigos concatenados construídos utilizando decodificação com decisão abrupta:

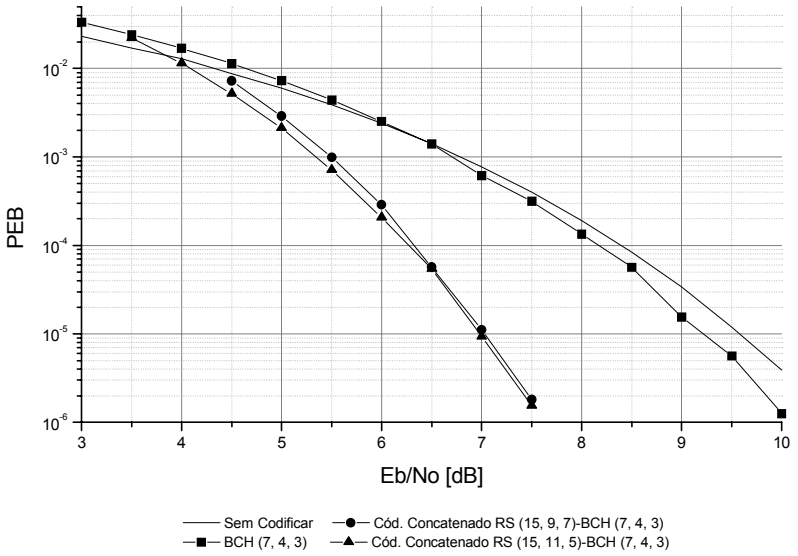

Figura 3.5: Códigos Concatenados usando o Código BCH (7, 4, 3) como Código Interno

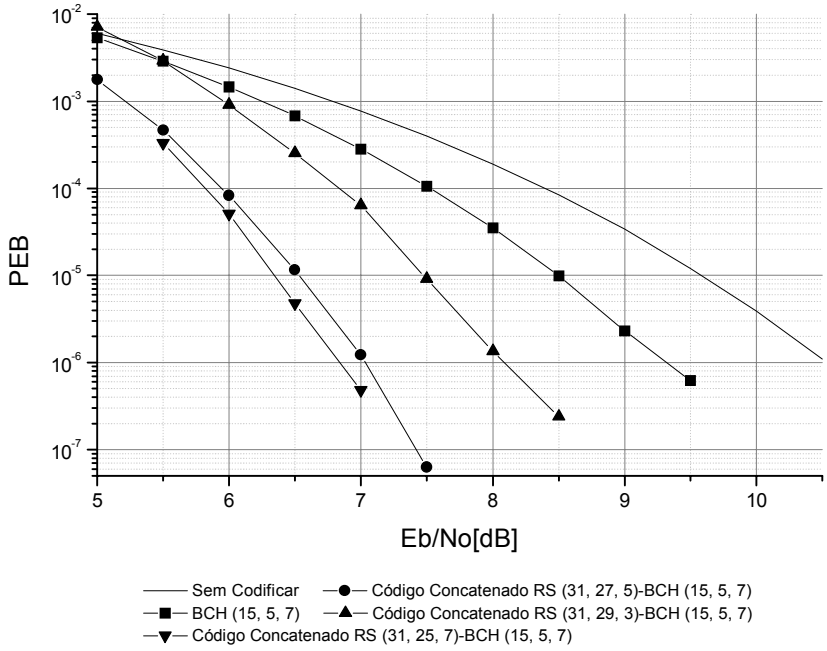

Figura 3.6: Códigos Concatenados usando o Código BCH (15, 5, 7) como Código Interno

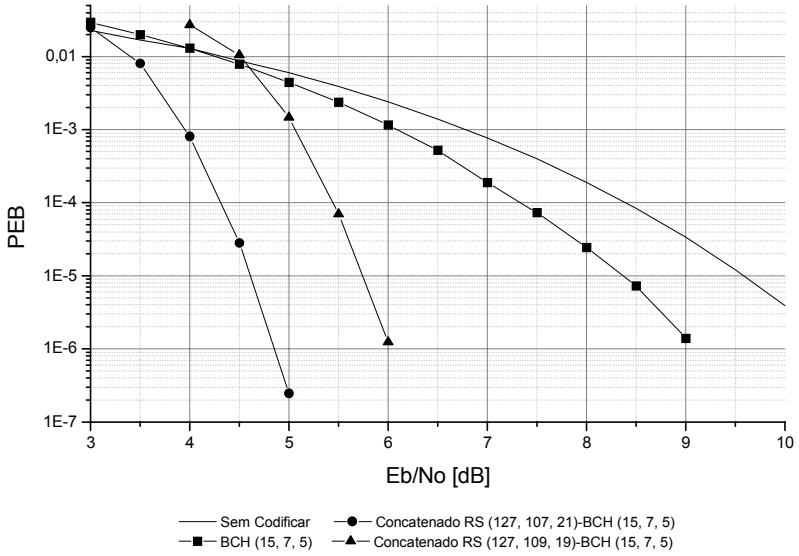

Figura 3.7: Códigos Concatenados usando o Código BCH (15, 7, 5) como Código Interno

A partir das três figuras anteriores pode-se notar que o ganho de codificação alcançado com a utilização de códigos concatenados é considerável em relação ao ganho de codificação que pode ser conseguido com a utilização dos códigos internos de forma isolada.

3.4.1.2 Decodificação com Decisão Suave

Uns dos principais resultados que tem incentivado o uso e a popularidade de códigos de bloco (em particular códigos BCH e de Reed-Solomon) em aplicações práticas, são os métodos algébricos de decodificação desenvolvidos para eles, em particular, o algoritmo de Berlekamp-Massey. Com o uso deste algoritmo, estes códigos podem ser decodificados com rapidez usando-se aritmética de corpos finitos que pode ser facilmente implementada em circuitos de grande escala de integração. Porém, estes algoritmos de decodificação só realizam

correção de erro e a informação suave utilizada pelo código C_1 interno não pode ser aproveitada. Assim, é muito difícil obter performances de máxima verossimilhança na decodificação [Schlegel, 1997].

Os códigos de bloco podem ser representados através de treliças e, portanto, os algoritmos existentes para a decodificação de códigos de treliça podem ser aplicados na decodificação destes códigos. O vetor modificador para atingir o valor de MinRL_{MAX} pode ser incorporado à treliça invertendo o valor dos bits correspondentes nos rótulos dos ramos da treliça. Assim sendo, decisão suave sobre o sinal de saída do canal pode ser feita, o que traz como conseqüência um ganho de codificação adicional. A representação de códigos de bloco por treliças é apropriada para a aplicação de algoritmos de busca de máxima verossimilhança, como por exemplo, o algoritmo de Viterbi.

Nas simulações realizadas, a quantização do sinal de saída do canal foi feita com precisão infinita (mais exatamente, através de números em ponto flutuante). Devido ao pequeno número de dígitos de verificação de paridade dos códigos internos utilizados no esquema de codificação concatenado, as treliças que os representam têm um número pequeno de estados. Assim, decidiu-se utilizar o algoritmo de Viterbi para realizar a decodificação de máxima verossimilhança destes códigos. Para isto, foram feitas algumas modificações na implementação deste algoritmo para códigos convolucionais, com vistas a adaptá-lo às treliças dos códigos de bloco. As especificações na decodificação por decisão suave para cada um dos códigos internos simulados estão colocadas a seguir:

- **Código RM (8, 4, 4)**

Este código foi representado por sua treliça mínima de acordo com a abordagem de Forney [Forney, 1988]. Para facilitar a construção da treliça, a matriz geradora do código deve ser transformada na forma de matriz geradora orientada a treliça (MGOT), a qual deve cumprir as seguintes condições [Lin, 1998]:

1) O primeiro "1" de cada linha está em uma coluna anterior à correspondente ao primeiro "1" de qualquer linha abaixo dela.
2) O último "1" de cada linha está em colunas diferentes.

Esta matriz pode ser obtida através de operações elementares sob as linhas da matriz geradora do código original.

- Matriz orientada a treliça:

$$G_{MGOT} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.7}$$

Note que, de acordo com a teoria exposta no Capítulo 2, esta matriz garante $\mathrm{Min}RL_{MAX} = 6$, podendo ser utilizado o mesmo vetor modificador que no caso da decisão abrupta.

- Treliça:

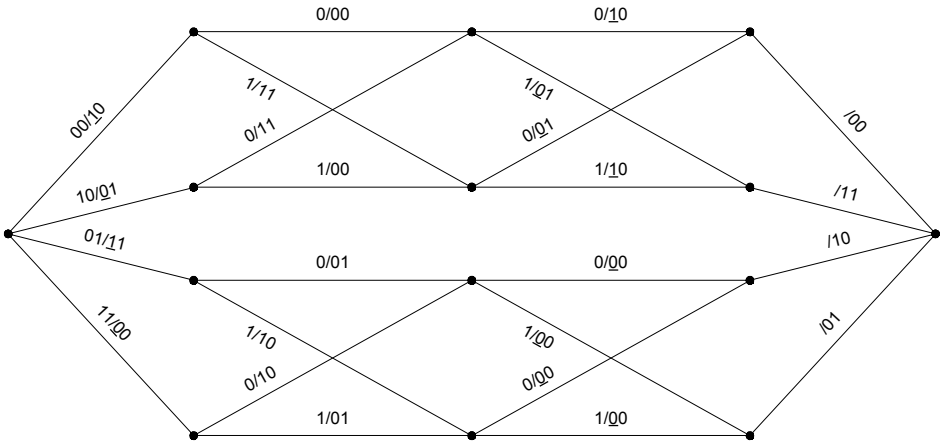

Figura 3.8: Treliça do Código RM (8, 4, 4)

Os rótulos dos ramos da treliça indicam a combinação: bits de entrada/bits de saída em cada instante de tempo de codificação. Os bits de saída sublinhados indicam a posição do "1" do vetor modificador e, portanto, esses bits estão invertidos com respeito à treliça original.

A Figura 3.9 mostra de forma comparativa o desempenho do código anterior para os dois tipos de decisão na decodificação:

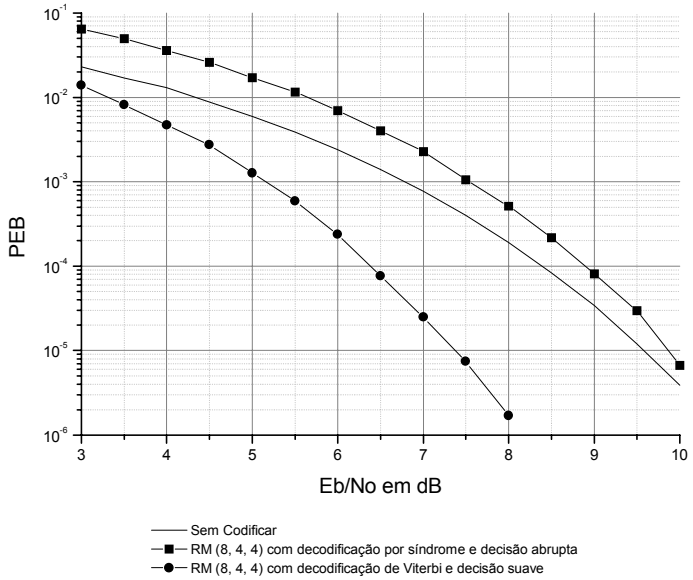

Figura 3.9: Desempenho do Código RM (8, 4, 4)

Pode-se notar o ganho de codificação de aproximadamente 2 dB alcançado com a utilização de decisão suave em relação ao ganho conseguido com a utilização de decisão abrupta na decodificação.

- **Código BCH (7, 4, 3)**

A representação por treliça deste código foi feita a partir da construção do código por meio de arranjos generalizados [Honary, 1997]. Segundo esta construção, a matriz geradora do código equivalente ao BCH (7, 4, 3) é a seguinte:

$$G = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{3.8}$$

Pode-se mostrar que esta matriz garante que MinRL_{MAX} = 7 com o mesmo vetor modificador usado no caso de decisão abrupta. A partir desta matriz, a treliça do código é:

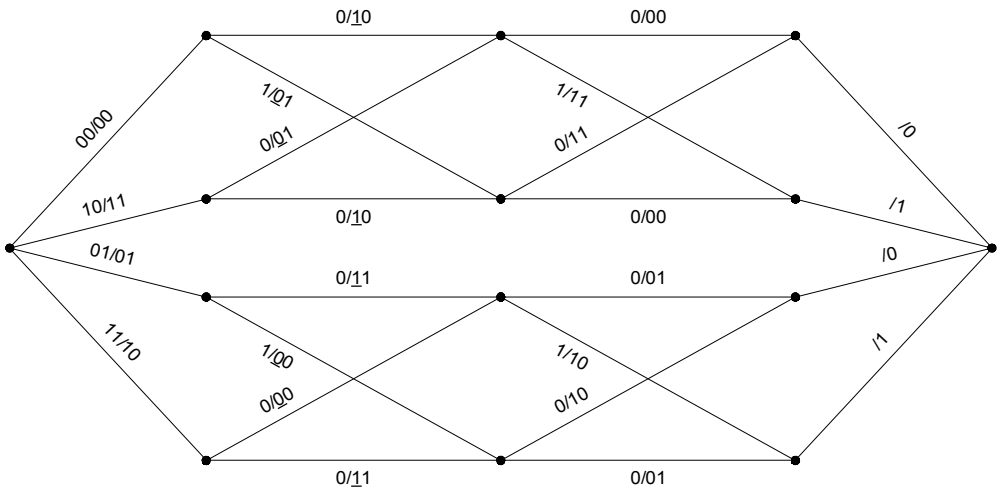

Figura 3.10: Treliça do Código BCH (7, 4, 3)

A Figura 3.11 mostra as curvas de desempenho deste código.

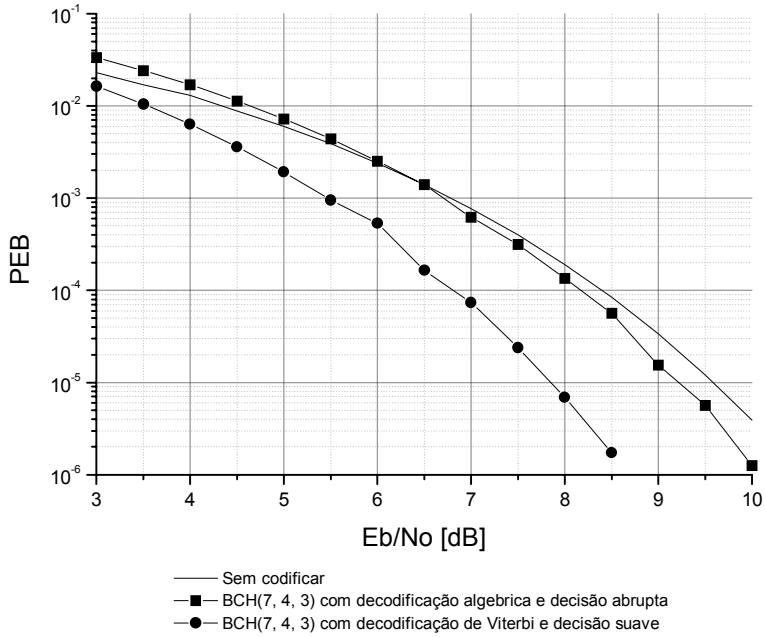

Figura 3.11: Desempenho do Código BCH (7, 4, 3)

- **Código RM (16, 8, 5)**

A representação por treliça deste código também foi feita a partir da construção do código por meio de arranjos generalizados, sendo a matriz geradora a mesma apresentada na Seção 3.1.2. Deve-se destacar que tanto nesta matriz como na do código anterior $(\mathrm{BCH}(7, 4, 3))$, foram feitas permutações de linhas com respeito às matrizes apresentadas em [Honary, 1997]. Isto se deve ao fato que as matrizes apresentadas em [Honary, 1997] não estão na forma orientada a treliça. É possível comprovar que os códigos produzidos pelas treliças apresentadas nessa referência, realmente são códigos de permutação em relação aos construídos a partir das matrizes geradoras ali apresentadas. Isto pode levar à introdução de

erros na decodificação se a matriz geradora for utilizada na codificação. Em nosso trabalho, este problema foi resolvido através das permutações de linhas como mencionado anteriormente.

A treliça do código RM (16, 5, 8) é portanto a seguinte:

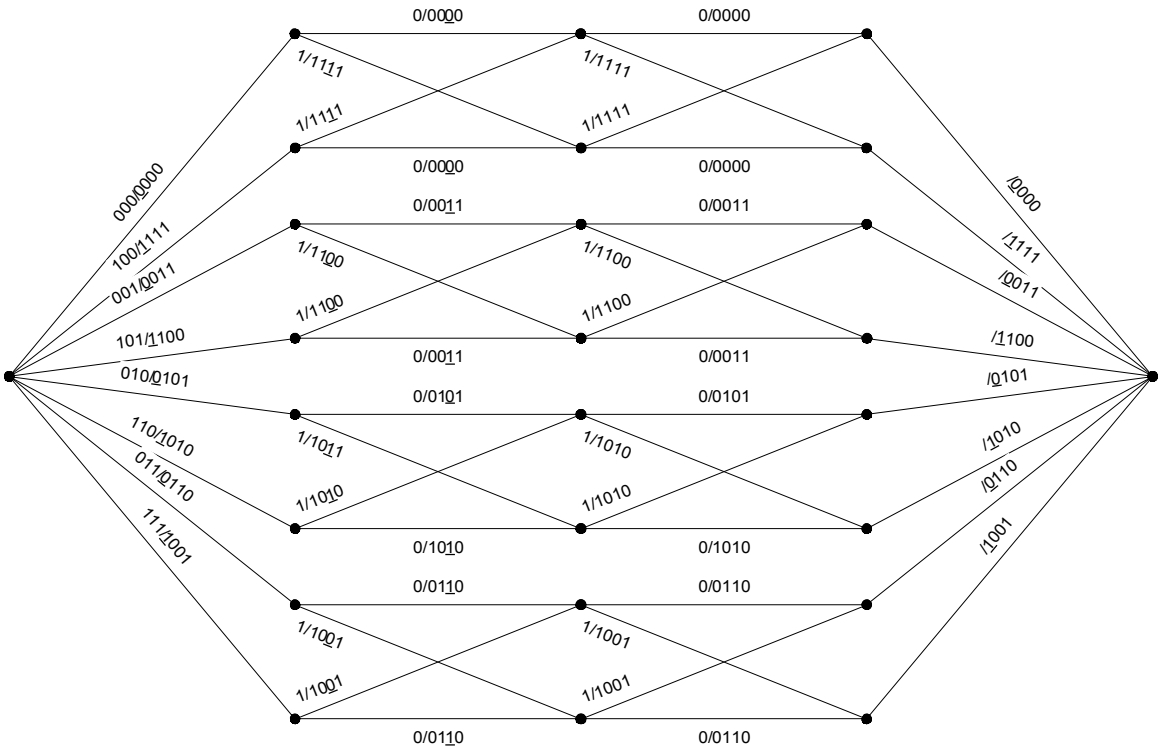

Figura 3.12: Treliça do Código RM (16, 5, 8)

A Figura 3.13 mostra o desempenho da decodificação deste código com decisão suave comparado com o desempenho da decodificação do código BCH (15, 5, 7) com decisão abrupta.

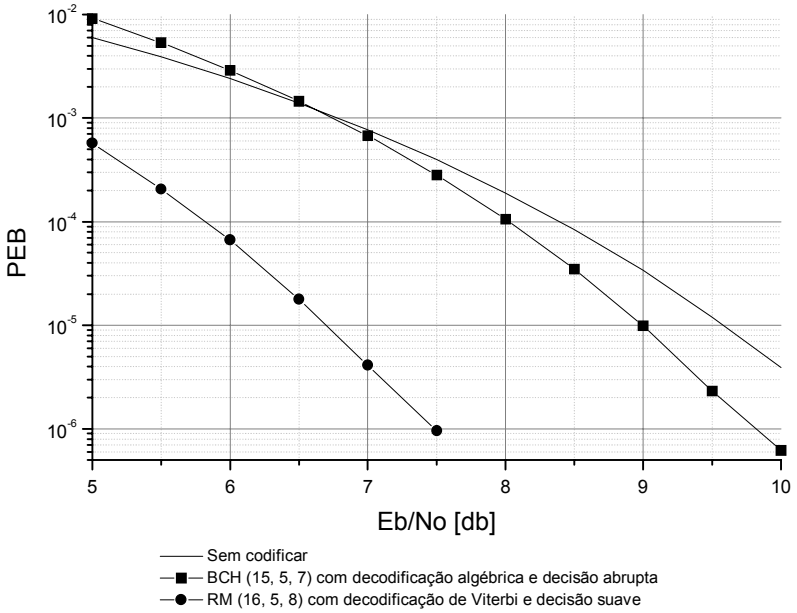

Figura 3.13: Desempenho do Código RM (16, 5, 8)

Note que para uma probabilidade de erro de bit de 10^{-6}, a curva correspondente à decodificação com decisão suave apresenta um ganho de codificação superior a 2 dB em relação à decodificação com decisão abrupta.

As figuras a seguir mostram as curvas de desempenho de vários códigos concatenados utilizando decisão suave na decodificação do código interno.

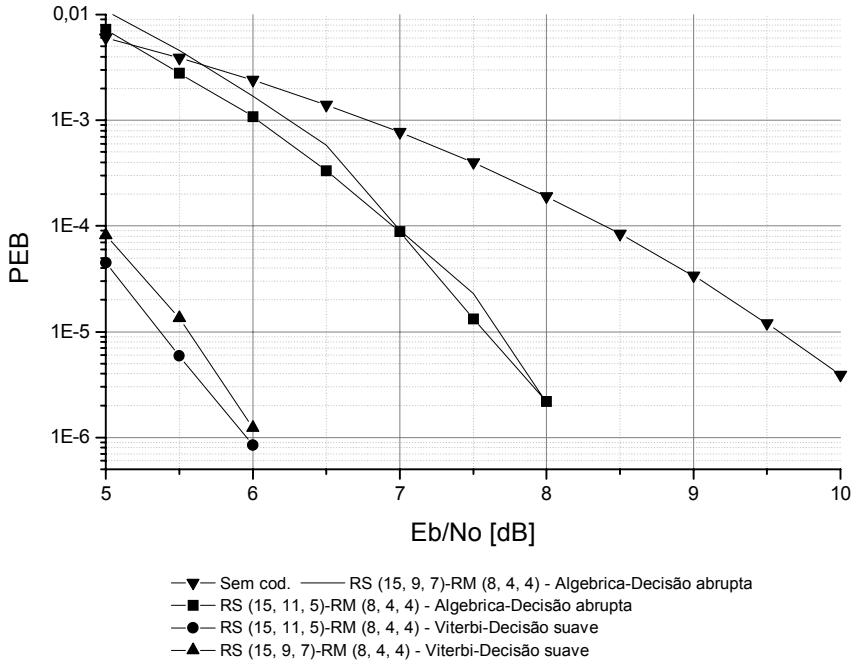

Figura 3.14: Desempenho de Códigos Concatenados Utilizando RM (8, 4, 4) como Código Interno

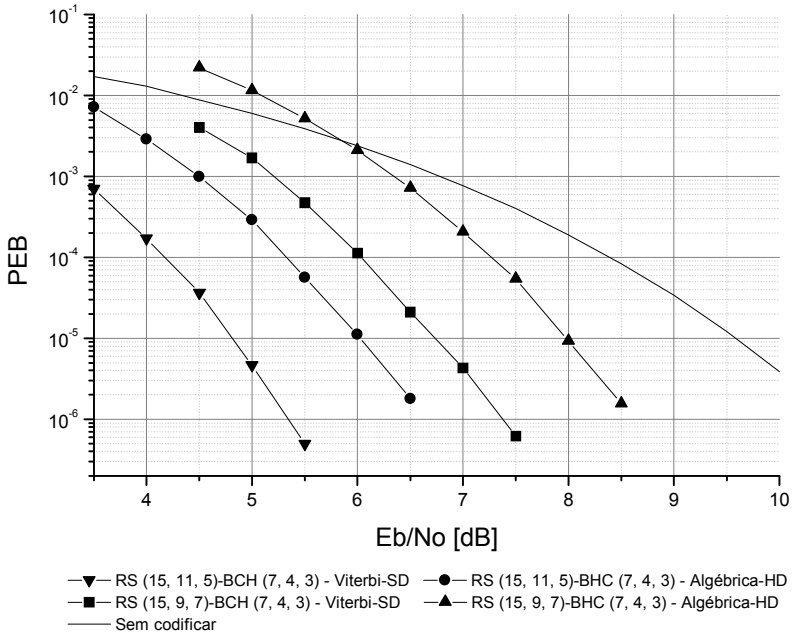

Figura 3.15: Desempenho de Códigos Concatenados Utilizando BCH (7, 4, 3) como Código Interno

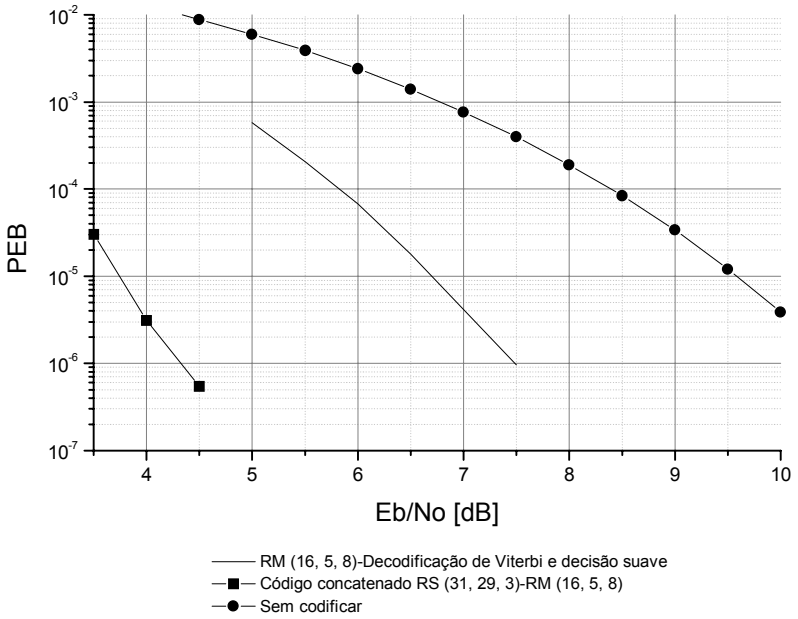

Figura 3.16: Desempenho do Código Concatenado RS (31, 29, 3)-RM (16, 5, 8)

Nas Figuras 3.14 e 3.15 pode-se notar que a partir de valores de probabilidade de erro de bit da ordem de 10^{-6}, as curvas correspondentes aos códigos que utilizam decisão suave apresentam um ganho de codificação acima de 2 dB em relação aos códigos que utilizam decisão abrupta na decodificação.

3.5 ENTRELAÇAMENTO

Os esquemas concatenados apresentados permitem a obtenção de códigos corretores de erro de grande comprimento com as mesmas restrições de "runlength" apresentadas pelos códigos internos de pequeno comprimento. A capacidade de correção de erro do código interno pode ser utilizada para a correção de erros aleatórios. Surtos de erro afetando alguns

símbolos podem ser corrigidos pelo código externo. Através do entrelaçamento dos símbolos do código externo pode-se aumentar o comprimento do surto que pode ser corrigido. Desta forma, os processos de codificação e decodificação apresentam algumas modificações com respeito aos processos descritos anteriormente, como será visto a seguir.

O código externo de Reed-Solomon C_2, com símbolos em $\mathrm{GF}(2^m)$ é entrelaçado a uma profundidade λ onde $\lambda = k_1/m$. Na codificação, primeiramente, uma palavra de informação de k_2 símbolos sobre $\mathrm{GF}(2^m)$ $(k_2 \times m \text{ bits})$, é codificada numa palavra de $n_2 = 2^m - 1$ símbolos do código externo C_2. Esta palavra é armazenada como uma coluna de comprimento n_2 em uma memória na forma de arranjo. Este processo se repete e após serem obtidas λ palavras do código externo C_2, terá se formado um arranjo de $(2^m - 1) \times \lambda$ símbolos sobre $\mathrm{GF}(2^m)$. Cada linha do arranjo é formada por λ símbolos $(\lambda \times m \text{ bits} = k_1 \text{ bits})$. No passo seguinte, cada linha do arranjo é codificada em uma palavra do código C_1. Finalmente, o vetor modificador é somado a cada palavra-código de C_1 para se obter uma seqüência de n_2 palavras-código modificadas de C_1 com "runlength" limitado.

Na decodificação, primeiramente, o vetor modificador é removido de cada seqüência de n_1 bits recebidos e, ao mesmo tempo, um decodificador convencional do código C_1 original é utilizado para produzir as palavras de informação correspondentes de comprimento k_1 bits $(\lambda \text{ símbolos sobre } \mathrm{GF}(2^m))$, os quais são armazenados como linhas de um arranjo. Este processo se repete até se obter um arranjo de n_2 linhas da forma descrita anteriormente. Finalmente, cada uma das λ colunas do arranjo são decodificadas pelo decodificador RS para se obter a estimativa da mensagem original.

O código RS externo tem uma capacidade de correção de t_2 erros de símbolos. Se a síndrome de uma coluna do arranjo corresponde a um padrão de t_2 ou menos símbolos errados na recepção, então a correção de erro é realizada com sucesso. Um surto de λ símbolos de comprimento, não importando onde comece, afetará não mais do que uma posição em cada coluna do arranjo. Portanto, o efeito do entrelaçamento faz com que o código RS externo adquira uma capacidade de correção de erros em surto de comprimento até $t_2 \times \lambda$ símbolos.

Como exemplo do esquema anterior, considere-se a concatenação do código de Reed-Solomon (15, 11, 5) sobre $\mathrm{GF}(2^4)$ entrelaçado a uma profundidade de $\lambda = 3$ como código externo e o código modificado de Golay (23, 12, 7) como código interno. Primeiramente, uma palavra de informação de $k_2 = 11$ símbolos sobre $\mathrm{GF}(2^4)$ é codificada numa palavra de $n_2 = 15$ símbolos pelo codificador do código externo C_2. Para isto, é utilizado o polinômio gerador $g(x) = x^4 + \alpha^{13} x^3 + \alpha^6 x^2 + \alpha^3 x + \alpha^{10}$ com símbolos sobre o corpo de Galois $\mathrm{GF}(2^4)$, obtido a partir do polinômio primitivo $p(x) = 1 + x + x^4$. Esta palavra é armazenada na forma de coluna num arranjo de memória. Após serem obtidas $\lambda = 3$ palavras desta forma, ficará armazenado na memória um arranjo de 15×3 vetores sobre $\mathrm{GF}(2^4)$.

A seguir, cada linha do arranjo (12 bits) é codificada numa palavra-código pelo código de Golay (23, 12, 7) usando-se a matriz geradora da equação (2.30), que foi obtida no exemplo da Secção 2.4.1 do Capítulo 2 e que por conveniência é reproduzida abaixo:

$$G = \begin{bmatrix}
1&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&1\\
1&1&1&0&0&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0&1&0&1\\
1&0&1&0&1&0&1&0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&1\\
1&0&0&1&1&1&0&0&0&0&0&0&1&0&1&0&0&0&1&0&0&0&1\\
1&0&0&1&0&1&1&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&1&0&1&1&0&1&0&0&1&0&0&1&0&1&0&0&0&0&0&0&0&0\\
0&1&1&0&1&1&0&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&1\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&1&1&0&0&0&1&0&1&1&1&0&0&0&0&0&0&0&0\\
0&1&0&1&0&1&1&0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&1\\
1&1&1&0&0&0&1&0&0&0&0&1&0&1&1&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&1
\end{bmatrix} \tag{3.9}$$

Finalmente, o vetor modificador **m**=[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] é adicionado a todas as palavras do código de Golay modificado para formar uma seqüência codificada com no máximo 14 "uns" ou "zeros" consecutivos.

Na Figura 3.17 são apresentadas as curvas de desempenho do código de Golay (23, 12, 7) e dos códigos concatenados entrelaçados que o utilizam como código interno. A decodificação do código de Golay foi feita utilizando decisão abrupta e decodificação por síndrome, tendo em conta que este é um código perfeito de comprimento moderado e, portanto, a complexidade da tabela de decodificação não é muito grande.

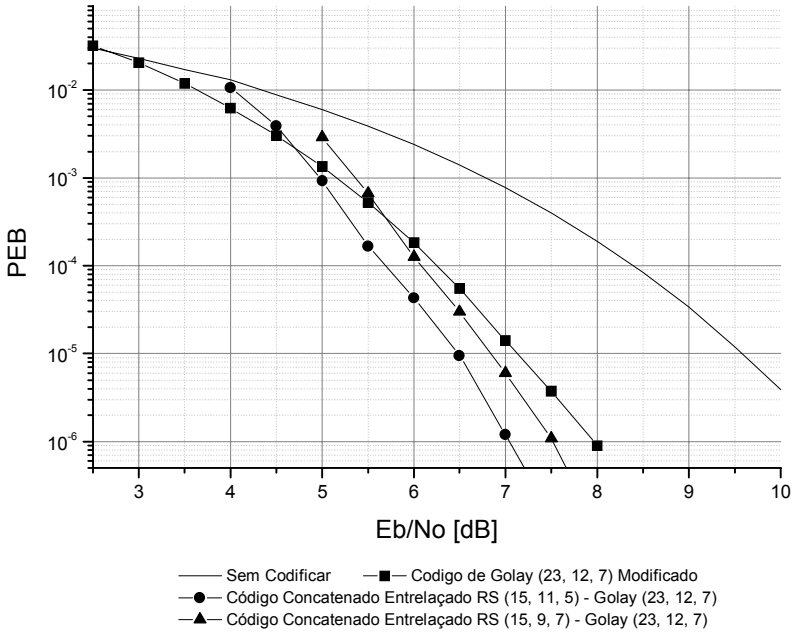

Figura 3.17: Códigos Concatenados Entrelaçados Utilizando o Código de Golay (23, 12, 7) como Código Interno.

A capacidade de correção de erro do código de Reed-Solomon (15, 11, 5) é de $t_2 = 2$ símbolos. Com o nível de entrelaçamento de $\lambda = 3$, este código é capaz de corrigir erros em surto de até 6 símbolos (ou 24) bits de comprimento. A capacidade de correção de erro do código de Golay (23, 12, 7) é de $t_1 = 3$ bits. Esta capacidade de correção pode ser utilizada para a correção de erros aleatórios. Assim, o código concatenado resultante combina propriedades de correção de erros em surtos e aleatórios.

3.6 CONCLUSÃO

Através da concatenação de códigos de bloco, se dispõe de um conjunto de códigos concatenados apropriados para canais com restrições no número de símbolos consecutivos

iguais a serem transmitidos por estes canais. Bons valores para estas restrições são garantidos pelos códigos internos utilizados nos esquemas de codificação concatenados. Os códigos externos utilizados em todos os casos foram códigos de Reed-Solomon, os quais são universalmente usados em esquemas de codificação concatenados, devido às suas boas estruturas algébricas que foram descritas anteriormente neste capítulo. Como códigos internos foram selecionados códigos BCH, RM e o código de Golay (23, 12, 7), os quais, para comprimento pequeno e/ou moderado das palavras-código, formam parte das melhores classes de códigos conhecidas em termos dos seus parâmetros (n, k, $d_{\min}$). A representação por treliças destes códigos junto com o uso do algoritmo de Viterbi para realizar decodificação de máxima verossimilhança constitui o método de decodificação mais eficiente para estes códigos [Forney, 1998]. A partir das simulações realizadas pode-se comprovar que o ganho de codificação obtido encontra-se dentro dos valores teóricos de ganho de codificação nominais para estes códigos [Forney, 1998]. Esquemas de codificação concatenados similares não foram encontrados na bibliografia pesquisada.

O algoritmo de Viterbi tradicionalmente utilizado para decodificar códigos convolucionais foi modificado para ser usado na decodificação dos códigos de bloco internos do sistema de codificação concatenada levando em conta que as treliças destes códigos são finitas ao contrario das treliças semi-infinitas dos códigos convolucionais. Devido ao pequeno comprimento dos códigos de bloco utilizados, o algoritmo de Viterbi não aporta resultados tão bons se comparados com o caso dos códigos convolucionais na decodificação, toda vez que o comprimento das seqüências a serem decodificadas fica restrito ao comprimento do código interno. Não obstante, das simulações realizadas se pode observar um ganho de aproximadamente 2 dB em relação à decodificação com decisão abrupta.

No próximo capítulo será apresentada a obtenção de algoritmos de modificação de códigos convolucionais em códigos com "runlength" limitado seguindo a mesma estratégia de utilizá-los como códigos internos de esquemas de codificação concatenados.

CAPÍTULO 4

CÓDIGOS CONVOLUCIONAIS COM "RUNLENGTH" LIMITADO

4.1 INTRODUÇÃO

No capítulo anterior foi descrito como obter códigos com restrições de "runlength" para praticamente qualquer comprimento e capacidade de correção de erro a partir de esquemas de codificação concatenada usando um código não binário (especificamente um código de Reed-Solomon) em conjunto com um código de bloco com "runlength" limitado. Utilizando a mesma estratégia, neste capítulo será considerada a obtenção de códigos

corretores de erro com "runlength" limitado a partir da concatenação de códigos de Reed-Solomon com códigos convolucionais modificados.

A decodificação destes códigos também pode ser feita de diferentes formas. Tendo em conta que os códigos convolucionais que serão utilizados como códigos internos têm pequeno comprimento de restrição, o algoritmo de Viterbi proporciona um meio eficiente de decodificação utilizando tanto decisão abrupta como decisão suave na saída do canal. No restante deste capítulo será apresentada uma forma de se obter estes códigos. Os resultados obtidos nas simulações realizadas na decodificação com decisão suave dos mesmos, também serão mostrados.

4.2 "COSETS" DE CÓDIGOS CONVOLUCIONAIS COM "RUNLENGTH" LIMITADO

Vários autores têm abordado a inconveniência da utilização de códigos binários lineares (na sua forma original) em sistemas de comunicações digitais e de armazenamento de informação em que o sincronismo de relógio no receptor é obtido a partir das transições na seqüência de símbolos recebida. Baumert *et al.* [Baumert, 1979] estudaram o sincronismo de bit em sistemas que utilizam códigos convolucionais. Eles inverteram cada segundo símbolo na seqüência codificada para evitar as seqüências toda zeros e toda uns. Este método supõe que os códigos convolucionais utilizados não contêm a palavra-código formada por uma seqüência semi-infinita de símbolos alternados.

Considere-se as palavras-código de um código convolucional binário C com parâmetros $(n, n-r), r \geq 1,$ como sendo seqüências semi-infinitas de n-uplas binárias e seja $\boldsymbol{A}$

uma seqüência semi-infinita de *n*-uplas binárias diferente de qualquer palavra-código em um número muito grande de posições. Um "coset" de *C* é obtido somando a seqüência ***A*** (chamada de líder de "coset") a todas as palavras-código de *C*. O método descrito em [Baumert, 1979] é equivalente a determinar um "coset" de um código convolucional cujo líder de "coset" ***A*** seja uma seqüência semi-infinita de zeros e uns alternados.

Utilizando um procedimento similar, Calderbank *et al.* [Calderbank, 1986] determinaram o valor de $\text{Min}RL_{MAX}$ para alguns "cosets" de códigos convolucionais com parâmetros $(n, n-1)$. Wolf e Ungerboeck [Wolf, 1986] mostraram que qualquer código convolucional $(n, n-1)$ tem um "coset" com $\text{Min}RL_{MAX} = n\nu + 2n - 2 - s$ onde ν é o comprimento de restrição e $s \in \{0, ..., n-1\}$. Estes resultados indicaram que "cosets" de códigos convolucionais com parâmetros $(n, n-1)$ apresentam valores grandes de "runlength".

Tentando obter "cosets" com valores pequenos de $\text{Min}RL_{MAX}$ (que são os valores desejáveis para lidar com o sincronismo de bit), Hole [Hole, 1995] obteve limitantes gerais de "runlength". Estes limitantes são mais simples e precisos do que os obtidos em [Baumert, 1979] devido a que não é necessário que o líder de "coset" seja igual à seqüência de uns e zeros alternados. Foi provado [Hole, 1995] que qualquer "coset" de um código convolucional binário *C* com parâmetros $(n, n-r)$ tem $\text{Min}RL_{MAX} \geq \nu_{\min}$ onde $\nu_{\min}$ é o valor do grau da linha de menor grau de uma matriz mínima de verificação de paridade na forma polinomial, ***H***(*D*), de *C*. A partir deste limitante pode-se dizer que qualquer "coset" de um código convolucional $(n, n-1)$ tem $\text{Min}RL_{MAX} \geq \nu$. Portanto, qualquer "coset" de um código convolucional $(n, n-1)$ com alta taxa e/ou alto comprimento de restrição tem um valor grande de $\text{Min}RL_{MAX}$.

Hole [Hole, 1995] também mostrou que existem "cosets" de códigos com parâmetros $(n, n-r)$ com valores pequenos de $\text{Min}RL_{MAX}$ para qualquer comprimento de restrição, para $r \geq 2$. Posteriormente, Hole e Ytrehus [Hole, 1999] mostraram como selecionar o "coset" com o menor valor possível de $\text{Min}RL_{MAX}$ para este tipo de código.

Farkaš e Šechny [Farkaš, 1996], propuseram um método para a limitação do "runlength" de códigos binários lineares de bloco e convolucionais que pode ser aplicado sob a condição de que as matrizes geradoras dos códigos originais cumpram determinadas condições. Posteriormente, em [Šechny, 1999], os autores apresentaram algumas modificações ao método anterior que possibilitaram diminuir o valor de $\text{Min}RL_{MAX}$ obtido previamente em [Farkaš, 1996] para "cosets" de códigos convolucionais.

Neste capítulo utilizaremos os códigos convolucionais obtidos em [Šechny, 1999] devido à forma simples em que a modificação é feita para obter valores de $\text{Min}RL_{MAX}$ que coincidem com os obtidos em [Hole, 1999]. Algumas modificações foram feitas com relação ao trabalho de Šechny e Farkaš que melhoram o desempenho do processo de codificação e decodificação, como será visto mais adiante neste capítulo.

4.3 OBTENÇÃO DE CÓDIGOS CONVOLUCIONAIS COM "RUNLENGTH" LIMITADO

A seguir será apresentado o procedimento para modificar códigos convolucionais conhecidos em códigos com "runlength" limitado como em [Šechny, 1999].

Considere um código convolucional binário. k_0 bits entram no codificador, que por sua vez calcula n_0 bits de saída a partir dos k_0 bits presentes na entrada e dos (m k_0) bits anteriores,

onde m é um inteiro positivo que representa a memória do codificador. Este código é denominado em [Blahut, 1983] código convolucional (n, k) sobre GF(2) com taxa $R = k_0 / n_0$, onde

$$k = (m+1)k_0, \qquad n = (m+1)n_0 \tag{4.1}$$

Também é mostrado em [Blahut, 1983] que a matriz geradora semi-infinita **G** deste código é formada por submatrizes $\mathbf{S}_i$, $i = 1, 2, 3, \ldots$, com dimensões $(i \cdot k_0) \times n_0$ se $i \le m$ e $k \times n_0$ se $i > m$ como mostrado na Figura 4.1. As submatrizes $\mathbf{S}_i$, $i \le m$, são compostas pelas $i \cdot k_0$ linhas inferiores de $\mathbf{S}_{m+1}$. Todas as $\mathbf{S}_i$, para $i > m$, são idênticas. Estas submatrizes são também mostradas na Figura 4.1 para o caso particular de um código convolucional com R=1/3 e m=3.

$$i: \quad \begin{matrix} 1 & & & 2 & & & 3 & & & 4 & & & 5 & & & 6 & & \cdots \end{matrix}$$

$$\mathbf{G} = \left[\begin{array}{ccc:ccc:ccc:ccc:ccc:ccc:ccc:c} 1&1&1&0&1&1&1&0&1&1&1&1&0&0&0&0&0&0& \\ 0&0&0&1&1&1&0&1&1&1&0&1&1&1&1&0&0&0& \\ 0&0&0&0&0&0&1&1&1&0&1&1&1&0&1&1&1&1& \\ 0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&1&0&1& \\ 0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&0&1&1&\cdots \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1& \\ & & & & & & & & & & & & & & & & & & \end{array}\right]$$

$$\mathbf{S}_1 = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{S}_2 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{S}_3 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\mathbf{S}_4 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Figura 4.1: Matriz Geradora e Submatrizes do Código Convolucional com R=1/3 e m=3.

A seqüência de bits na saída do codificador pode ser vista como uma seqüência de n_0-uplas onde a i-ésima n_0-upla é uma combinação linear de linhas em $\mathbf{S}_i$ determinada pela posição dos uns na seqüência de entrada. As 2^k combinações de linhas diferentes em cada $\mathbf{S}_i$, $i > m$, podem produzir no máximo 2^{n_0} n_0-uplas distintas. Geralmente, porém, as n_0-uplas possíveis no i-ésimo segmento, $i \leq m$, formam um subconjunto do conjunto de n_0-uplas que pode acontecer em cada l-ésimo segmento, $l > m$. Portanto, a análise será centrada a partir da $(m+1)$-ésima n_0-upla da seqüência codificada sendo desconsiderada a primeira parte da mesma.

Considere agora que a submatriz $\mathbf{S}_{i+1}$, $i > m$, esteja ativada por k_0 bits com respeito à submatriz $\mathbf{S}_i$. Neste caso pode-se ver que os bits da $(2n_0)$-upla formada pelos dois segmentos consecutivos i e $(i+1)$, $i > m$, são determinados por um segmento de $(k + k_0)$ bits na seqüência de entrada. Em outras palavras, se $i > (m+1)$, cada i-ésima n_0-upla pode ser seguida por no máximo 2^{k_0} n_0-uplas diferentes geradas por $\mathbf{S}_{i+1}$ e precedidas por no máximo 2^{k_0} n_0-uplas diferentes geradas por $\mathbf{S}_{i-1}$. Portanto, se o código satisfaz a condição de que o número total de $(2n_0)$-uplas possíveis em cada palavra-código é menor que o número total de todas as $(2n_0)$-uplas (ver equação (4.2)), pode-se assegurar que cada n_0-upla em cada palavra-código pode ser seguida só por algumas e não por todas 2^{n_0} n_0-uplas possíveis,

$$2^{(k+k_0)} < 2^{(2 \cdot n_0)}, \qquad \text{ou seja,} \quad k < 2 \cdot n_0 - k_0 \tag{4.2}$$

Por exemplo, o código da Figura 4.1 satisfaz a equação (4.2). Na Figura 4.3 (a) são mostradss para cada n_0-upla todas as n_0-uplas que podem vir na continuação dela.

Se um código obedece a equação (4.2), existe a possibilidade de que ele tenha restrições de "runlength" do tipo $(\max_0, \max_1)$, onde a seqüência de zeros consecutivos mais longa que é subseqüência de qualquer seqüência codificada tem comprimento $\max_0$ e a seqüência mais longa de uns consecutivos tem comprimento igual a $\max_1$ Neste caso, como os códigos convolucionais não são transparentes, os valores $\max_0$ e $\max_1$ podem não coincidir, portanto, optou-se por utilizar a notação $(\max_0, \max_1)$ ao invés de $\mathrm{Min}RL_{MAX}$. Estas restrições de "runlength" podem ser conseguidas através de determinadas modificações no código original. Primeiramente, através de permutações das colunas de cada submatriz $\mathbf{S}_i$ podem-se diminuir os valores $\max_0$ e $\max_1$ mudando o conjunto de possíveis sucessores. Estas permutações são equivalentes a mudar a ordem dos *n* bits de saída. Em segundo lugar, para evitar a situação em que uma n_0-upla toda zeros seja seguida por outra n_0-upla toda zeros (e da mesma forma para as n_0-uplas toda uns), podem ser invertidos determinados bits em cada segmento da seqüência codificada por meio de um vetor modificador semi-infinito e periódico.

Por exemplo, fazendo estas modificações no código apresentado na Figura 4.1, a matriz geradora resultante e o vetor modificador para o código convolucional modificado são mostrados na Figura 4.2. Na Figura 4.3 (b) são mostradas as n_0-uplas e todas suas possíveis sucessoras na seqüência de palavras-código do código modificado da Figura 4.2. Pode-se comprovar que este código satisfaz $(\max_0, \max_1) = (5, 5)$.

$$i: \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad \ldots$$

$$\mathbf{G}=\left[\begin{array}{ccc:ccc:ccc:ccc:ccc:ccc:ccc}
1&1&1&1&1&0&0&1&1&1&1&1&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&0&0&1&1&1&1&1&0&0&0&0\\
0&0&0&0&0&0&1&1&1&1&1&0&0&1&1&1&1&1&1\\
0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
&&&&&&&&&&&&&&&&&&
\end{array}\right]\cdots \qquad \mathbf{S}_i=\begin{bmatrix}1&1&1\\0&1&1\\1&1&0\\1&1&1\end{bmatrix} \; i>4$$

$$\mathbf{m}=(100\ 100\ 100\ 100\ 100\ 100\ \ldots)=(\bar{\mu},\bar{\mu},\ldots)$$

Figura 4.2: Matriz Geradora e Vetor Modificador do Código Modificado.

3-upla de ramo	Possíveis sucessores
000	001, 111, 000, 100
001	010, 110, 001, 101
010	011, 111, 000, 100
011	010, 110, 001, 101
100	010, 110, 001, 101
101	011, 111, 000, 100
110	010, 110, 001, 101
111	011, 111, 000, 100

(a)

3-upla de ramo	Possíveis sucessores
000	010, 011, 100, 101
001	000, 001, 110, 111
010	000, 001, 110, 111
011	010, 011, 100, 101
100	010, 011, 100, 101
101	000, 001, 110, 111
110	000, 001, 110, 111
111	010, 011, 100, 101

(b)

Figura 4.3: Possíveis Sucessoras para cada 3-upla no (a) Código Original da Figura 4.1 e (b) Código Modificado da Figura 4.2.

Estas modificações são realizadas utilizando-se um entrelaçador interno de ramo após o qual a nova ordem das colunas em $\mathbf{S}_i$ é (1 2 0) e a continuação, somando o vetor modificador $\mathbf{m} = (\overline{\mu}, \overline{\mu}, \ldots)$, onde $\overline{\mu}$ representa o modificador interno de cada ramo da treliça do código. O diagrama em blocos do processo de modificação de códigos convolucionais é mostrado na Figura 4.4.

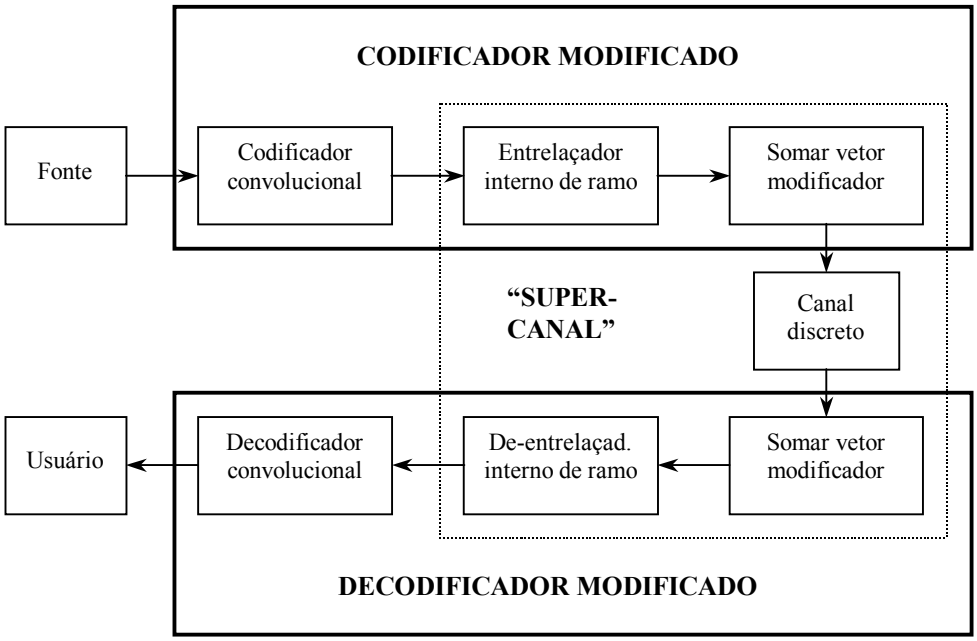

Figura 4.4: Diagrama em Blocos do Esquema de Codificação Modificado proposto em [Šechny, 1999].

Para entender porque a capacidade de correção de erro do código original permanece inalterada após as modificações é necessário analisar o processo de modificação. Este processo, ilustrado na Figura 4.4, é uma operação em cascata que é realizada em blocos separados depois do codificador convolucional e antes do decodificador.

A operação de inversão de bits é realizada simplesmente através da soma modulo-2 de um vetor modificador que periodicamente inverte os bits selecionados previamente. Como o reordenamento das saídas do codificador está confinado a um ramo na treliça do código e todos os ramos em qualquer nível da treliça são reordenados e modificados pelo vetor modificador da mesma forma, estas modificações não alteram as propriedades de distância do código convolucional original. Por outro lado, a inversão de bits antes do decodificador elimina o efeito da inversão realizada após a codificação e o de-entrelaçador elimina o efeito do entrelaçador. Em outras palavras, o código convolucional original opera sobre um "supercanal" e as operações de modificação são transparentes para o codificador e o decodificador.

Os melhores códigos obtidos em termos de capacidade de correção de erro e propriedades de "runlength" (valores mínimos de $(\max_0, \max_1)$), são mostrados na Tabela 4.1:

Tabela 4.1: Códigos Convolucionais com "Runlength" Limitado

Códigos	Taxa (R)	d_{free}	Re-ordenamento das colunas em S_i	Vetor Modificador $\bar{\mu}$	$(\max_0, \max_1)$
Código 1	1/3	8	0,1,2	010	(3,3)
Código 2	1/3	10	1,2,0	100	(5,5)
Código 3	1/4	20	0,1,2,3	1000	(4,4)
Código 4	1/3	6	2,0,1	100	(4,5)
Código 5	1/3	6	2,0,1	001	(5,4)

4.4 CONCATENAÇÃO

Os códigos apresentados na Tabela 4.1 têm comprimento de restrição pequeno e, portanto, sua capacidade de correção de erro é pequena. Assim, o uso destes códigos de forma isolada não garante um ganho de codificação elevado com respeito aos sistemas não codificados. A aplicação do algoritmo de modificação em códigos de comprimento de restrição maior levaria à obtenção de valores muito altos para $(\max_0, \max_1)$. Um aumento considerável no ganho de codificação mantendo-se inalterados os valores das restrições de "runlength" pode ser obtido através da utilização destes códigos como códigos internos em sistemas de codificação concatenada junto a códigos corretores de erro externos de comprimento maior.

Para isto se propõe um esquema de codificação concatenada similar ao apresentado na Figura 3.1 do capítulo anterior, sendo que neste caso, o código interno C_1 é um código convolucional (n_1, k_1) modificado para se obter o valor de $\text{Min}RL_{MAX}$ desejado. O código externo C_2 é um código não binário (n_2, k_2) de Reed-Solomon com símbolos em $\text{GF}(2^m)$, os quais são representados por seqüências binárias de m bits.

No processo de codificação, primeiramente os bits de informação na saída da fonte são divididos em k_2 símbolos de m bits cada. Esses k_2 símbolos são codificados por um codificador de Reed-Solomon, obtendo-se uma palavra de n_2 símbolos do código externo C_2. Na próxima etapa, a seqüência codificada de $n_2 \times m$ bits é utilizada como entrada ao codificador do código convolucional C_1 (k_1 bits por vez). Finalmente, um vetor modificador é somado a cada seqüência de n_1 bits do código C_1 para se obter uma seqüência de n_2 palavras-código

modificadas de C_1 com "runlength" limitado. Assim, o código resultante é um código binário (n_1n_2, k_1k_2).

A decodificação dos códigos internos pode ser realizada utilizando-se diferentes algoritmos de decodificação para códigos convolucionais. Entre estes algoritmos encontram-se os algoritmos de decodificação seqüencial, de decodificação por lógica majoritária e o algoritmo de Viterbi. A decodificação seqüencial foi o primeiro algoritmo para a decodificação de códigos convolucionais e foi introduzido por Wozencraft em 1957 [Wozencraft, 1961]. Outras versões deste algoritmo, particularmente o algoritmo de Fano e o algoritmo de "stack" foram desenvolvidos por Fano [Fano, 1963] e por Zigangirov e Jelenik [Jelenik, 1969]. O algoritmo de decodificação por lógica majoritária é um método de decodificação algébrica baseado no algoritmo de decodificação por limiar que foi generalizado por Massey [Massey, 1963] a partir do algoritmo desenvolvido por Reed para a decodificação de códigos de Reed-Muller [Reed, 1954]. Em 1967, Viterbi [Viterbi, 1967] introduziu um algoritmo probabilistico não seqüencial para a decodificação de códigos covolucionais que ficou conhecido como o algoritmo de Viterbi. Posteriormente, Forney [Forney, 1974] demonstrou que o algoritmo de Viterbi é de fato um algoritmo de decodificação de máxima verossimilhança para códigos convolucionais.

A complexidade na decodificação utilizando-se o algoritmo de Viterbi aumenta exponencialmente em relação ao comprimento de restrição do código convolucional. Desta forma, e tendo em conta o pequeno comprimento dos códigos utilizados neste trabalho, decidiu-se utilizar este algoritmo na decodificação dos códigos internos do esquema de codificação concatenada.

A decodificação dos códigos RS externos é feita utilizando-se o algoritmo de Berlekamp-Massey para a decodificação com decisão abrupta. No capítulo anterior foi colocado que a principal limitação dos códigos de Reed-Solomon reside na dificuldade existente para encontrar algoritmos eficientes de decodificação com decisão suave para eles, devido principalmente à incompatibilidade entre a estrutura algébrica dos corpos finitos, sob os quais estes códigos são construídos, e os valores reais do sinal na saída do canal na recepção.

Por outro lado, para códigos convolucionais, a decisão suave na decodificação utilizando o algoritmo de Viterbi é incorporada facilmente e de forma natural, obtendo-se um incremento no ganho de codificação acima de 2dB com respeito à decodificação com decisão abrupta em um canal Gaussiano. Entretanto, os códigos convolucionais apresentam seus próprios problemas, por exemplo, a dificuldade para serem implementados com altas taxas de codificação e a tendência que estes códigos têm de gerar surtos de erros na saída do decodificador na medida em que o nível de ruído na entrada do mesmo aumenta.

Um bom desempenho pode ser obtido através da combinação de códigos de Reed-Solomon com códigos convolucionais em esquemas de codificação concatenados. O código convolucional (com decodificação de Viterbi e decisão suave) é utilizado para "limpar" o canal para o código de Reed-Solomon, que por sua vez, corrige os surtos de erros provenientes do decodificador de Viterbi. Assim, através da escolha apropriada dos códigos, pode-se obter uma probabilidade de erro que diminui exponencialmente em proporção ao comprimento total do código a qualquer taxa inferior à capacidade do canal. Enquanto isso, a complexidade da decodificação é dominada pela complexidade do decodificador algébrico do código de Reed-Solomon, a qual cresce na proporção do aumento do comprimento do código [Reed, 1999].

A partir dos códigos apresentados na Tabela 4.1 e de suas possíveis combinações com códigos de Reed-Solomon, pode-se obter um grande número de códigos concatenados com valores de $(\max_0, \max_1)$ iguais aos listados na Tabela 4.1 para os códigos convolucionais internos. Dentre essas combinações foram selecionadas para simulação aquelas que garantem uma capacidade de correção de erro adequada sem que a taxa do código diminua muito. Na Tabela 4.2 são mostrados os códigos concatenados construídos.

Tabela 4.2: Códigos Concatenados

Código Externo	Código Interno	Código Concatenado	Taxa	$(\max_0, \max_1)$
RS (255, 223, 33)	Código 1	(765, 223)	0,29	(3,3)
RS (127, 109, 19)	Código 1	(381, 109)	0,286	(3,3)
RS (255, 223, 33)	Código 2	(765, 223)	0,29	(5,5)
RS (127, 109, 19)	Código 2	(381, 109)	0,286	(5,5)
RS (255, 223, 33)	Código 3	(1020, 223)	0,22	(4,4)
RS (127, 109, 19)	Código 3	(508, 109)	0,215	(4,4)
RS (255, 223, 33)	Código 4	(765, 223)	0,29	(4,5)
RS (127, 109, 19)	Código 4	(381, 109)	0,286	(4,5)
RS (255, 223, 33)	Código 5	(765, 223)	0,29	(5,4)
RS (127, 109, 19)	Código 5	(381, 109)	0,286	(5,4)

4.4.1 CODIFICAÇÃO DO CÓDIGO EXTERNO

Os códigos externos utilizados nos códigos concatenados listados na Tabela 4.2 são códigos de Reed-Solomon primitivos sistemáticos. As características principais destes códigos são:

- **Código RS (127, 109, 19)**

  - Símbolos sobre $\mathrm{GF}(2^7)$ construído a partir do elemento primitivo α e do polinômio primitivo $1+x^3+x^7$.

  - Polinômio gerador:

$$g(x)=\alpha^{44}+\alpha^{83}x+\alpha^{73}x^2+\alpha^{38}x^3+\alpha^{89}x^4+\alpha^{126}x^5+\alpha^{63}x^6+\alpha^{76}x^7+\alpha^{105}x^8+\alpha^{88}x^9+\alpha^{86}x^{10}+\alpha^{38}x^{11}+\alpha^{6}x^{12}+\alpha^{50}x^{13}+\alpha^{121}x^{14}+\alpha^{51}x^{15}+\alpha^{67}x^{16}+\alpha^{58}x^{17}+x^{18}$$

- **Código RS (255, 223, 33)**

  - Símbolos sobre $\mathbf{GF}(2^8)$ construído a partir do elemento primitivo α e do polinômio primitivo $1+x^2+x^3+x^4+x^8$.

  - Polinômio gerador:

$$g(x)=\alpha^{18}+\alpha^{251}x+\alpha^{215}x^2+\alpha^{28}x^3+\alpha^{80}x^4+\alpha^{107}x^5+\alpha^{248}x^6+\alpha^{53}x^7+\alpha^{84}x^8+\alpha^{194}x^9+\alpha^{91}x^{10}+\alpha^{59}x^{11}+\alpha^{176}x^{12}+\alpha^{99}x^{13}+\alpha^{203}x^{14}+\alpha^{137}x^{15}+\alpha^{43}x^{16}+\alpha^{104}x^{17}+\alpha^{137}x^{18}+x^{19}+\alpha^{44}x^{20}+\alpha^{149}x^{21}+\alpha^{148}x^{22}+\alpha^{218}x^{23}+\alpha^{75}x^{24}\alpha^{11}x^{25}+\alpha^{173}x^{26}+\alpha^{254}x^{27}+\alpha^{194}x^{28}+\alpha^{109}x^{29}+\alpha^{8}x^{30}+\alpha^{11}x^{31}+x^{32}$$

4.4.2 CODIFICAÇÃO DO CÓDIGO INTERNO

A codificação dos códigos convolucionais internos foi feita utilizando máquinas de estados finitos. Em [Šechny, 1999] a mudança na ordem das colunas da submatriz $\mathbf{S}_i$ e a soma do vetor modificador foram feitas em blocos independentes em cascata com o codificador convolucional original. Estas operações podem ser incorporadas no próprio codificador. Na Figura 4.5 é mostrado o codificador do código convolucional do exemplo da Figura 4.1:

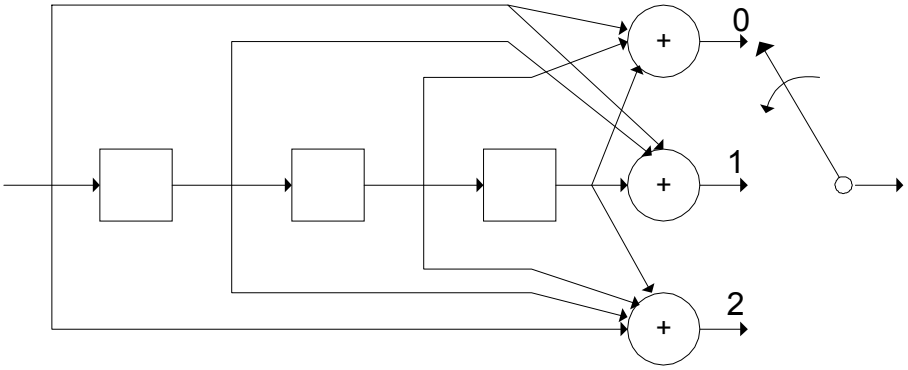

Figura 4.5: Codificador Convolucional do Exemplo da Figura 4.1.

As modificações feitas neste código implicam que a nova ordem das colunas na submatriz $\mathbf{S}_i$ é (1 2 0). Isto pode ser realizado intercambiando apropriadamente as saídas do codificador. Por outro lado, a soma do vetor modificador $\mathbf{m} = (\overline{\mu}, \overline{\mu}, \ldots)$, onde $\overline{\mu} = (1 \;\; 0 \;\; 0)$, pode ser realizada diretamente no somador correspondente do codificador como mostrado na Figura 4.6:

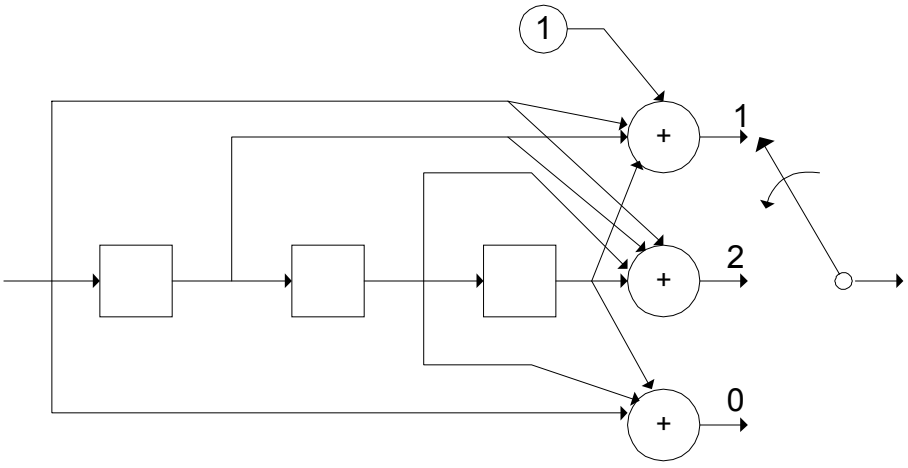

Figura 4.6: Codificador Convolucional Modificado.

As características principais destes códigos convolucionais utilizados como códigos internos são:

- **Código 1**

  - $R = 1/3, \quad m = 2, \quad d_{free} = 8$.

  - Polinômios geradores em octal:

  (5, 7, 7)

  - Matriz geradora modificada:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \\ & & & & & & & \vdots & & & & & & & & \end{bmatrix} \quad \mathbf{S}_i = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.3}$$

  - Vetor modificador: $\mathbf{m} = [0 \quad 1 \quad 0, \quad 0 \quad 1 \quad 0, \quad \ldots]$

- **Código 2**

  - $R = 1/3, \quad m = 3, \quad d_{free} = 10$

  - Polinômios geradores em octal:

  (64, 54, 74)

  - Matriz geradora modificada:

$$\mathbf{G}=\begin{bmatrix} 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ \cdots \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 1 \\ \multicolumn{24}{c}{\vdots} \end{bmatrix} \mathbf{S}_i=\begin{bmatrix} 1\ 1\ 1 \\ 1\ 1\ 0 \\ 0\ 1\ 1 \\ 1\ 0\ 1 \end{bmatrix} \qquad (4.4)$$

- Vetor modificador: $\mathbf{m}=[1\quad 0\quad 0,\quad 1\quad 0\quad 0,\quad \ldots]$

- **Código 3**

  - $R=1/4,\quad m=6,\quad d_{free}=20$

  - Polinômios geradores em octal:

  (564, 564, 714, 634)

  - Matriz geradora modificada:

$$\mathbf{G}=\begin{bmatrix} 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 0\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 1\ 1\ 1\ 1 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ \cdots \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1 \\ \multicolumn{24}{c}{\vdots} \end{bmatrix} \mathbf{S}_i=\begin{bmatrix} 1\ 1\ 1\ 1 \\ 0\ 0\ 1\ 1 \\ 1\ 1\ 0\ 1 \\ 1\ 1\ 0\ 0 \\ 1\ 1\ 1\ 0 \\ 0\ 0\ 1\ 1 \\ 1\ 1\ 1\ 1 \end{bmatrix} \qquad (4.5)$$

  - Vetor modificador: $\mathbf{m}=[1\quad 0\quad 0\quad 0,\quad 1\quad 0\quad 0\quad 0,\quad \ldots]$

- **Código 4**

  - $R=1/3,\quad m=2,\quad d_{free}=6$

- Polinômios geradores em octal:

(1, 3, 5)

- Matriz geradora modificada:

$$\mathbf{G}=\begin{bmatrix} 1&0&1&1&0&0&1&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&1&1&0&0&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&1&1&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&1&0&0&1&1&1\ \cdots \\ & & & & & & & & & \vdots & & & & & & & & \end{bmatrix} \quad \mathbf{S}_i=\begin{bmatrix} 1&1&1 \\ 1&0&0 \\ 1&0&1 \end{bmatrix} \tag{4.6}$$

- Vetor modificador: $\mathbf{m}=[1 \quad 0 \quad 0,\quad 1 \quad 0 \quad 0,\quad \ldots]$

- **Código 5**

- $R=1/3, \quad m=2, \quad d_{free}=6$

- Polinômios geradores em octal:

(1, 3, 5)

- Matriz geradora modificada:

$$\mathbf{G}=\begin{bmatrix} 1&0&1&1&0&0&1&1&1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&1&0&1&1&0&0&1&1&1&0&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&1&1&0&0&1&1&1&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0&1&1&0&0&1&1&1\ \cdots \\ & & & & & & & & & \vdots & & & & & & & & \end{bmatrix} \quad \mathbf{S}_i=\begin{bmatrix} 1&1&1 \\ 1&0&0 \\ 1&0&1 \end{bmatrix} \tag{4.7}$$

- Vetor modificador: $\mathbf{m}=[0 \quad 0 \quad 1,\quad 0 \quad 0 \quad 1,\quad \ldots]$

4.5 ELABORAÇÃO DE ALGORITMOS DE DECODIFICAÇÃO EFICIENTES

O processo de decodificação dos códigos concatenados foi feito utilizando decisão abrupta no código externo e decisão suave no código interno. De forma geral, o processo de decodificação é realizado em três etapas. Primeiramente, o vetor modificador é retirado de cada n_0-upla modificada do código C_1. Na etapa seguinte, é decodificada uma seqüência de $(n_0 \times m \times n_2)$ bits de saída do canal pelo decodificador do código C_1 obtendo-se uma seqüência de $(k_0 \times m \times n_2)$ bits que corresponde ao comprimento (em bits) de uma palavra do código externo levando-se em conta que para todos os códigos internos utilizados, $k_0 = 1$. Finalmente, esta seqüência é decodificada pelo decodificador do código externo.

4.5.1 DECODIFICAÇÃO DO CÓDIGO INTERNO

A decodificação do código interno foi feita utilizando o algoritmo de Viterbi com decisão suave. Para isto se trabalhou com uma profundidade na treliça de $(m+1)\times 5$. Valores maiores de profundidade da treliça aumentam o atraso no processo de decodificação sem incrementar significativamente a performance da decodificação [Flemming, 1999].

O vetor modificador para atingir os valores mínimos de $(\max_0, \max_1)$ pode ser incorporado à treliça invertendo o valor dos bits nos rótulos dos ramos da treliça. Da mesma forma, intercambiando apropriadamente estes rótulos, podem ser incorporadas à treliça as modificações feitas na ordem das colunas da submatriz $\boldsymbol{S}_i$. Assim, simplifica-se o processo de decodificação com respeito ao apresentado em [Šechny, 1999]. Por exemplo, nas figuras a seguir são mostradas as treliças para a decodificação do código 1 e do código 1 modificado:

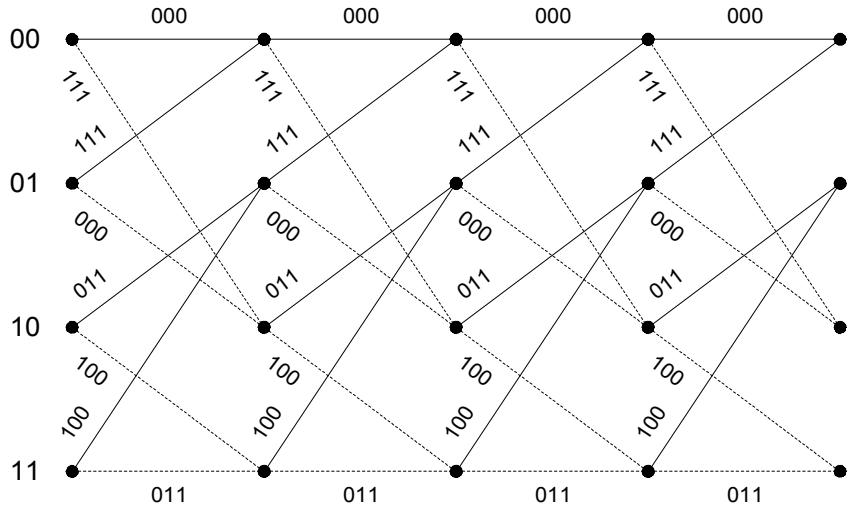

Figura 4.7: Treliça do Código 1

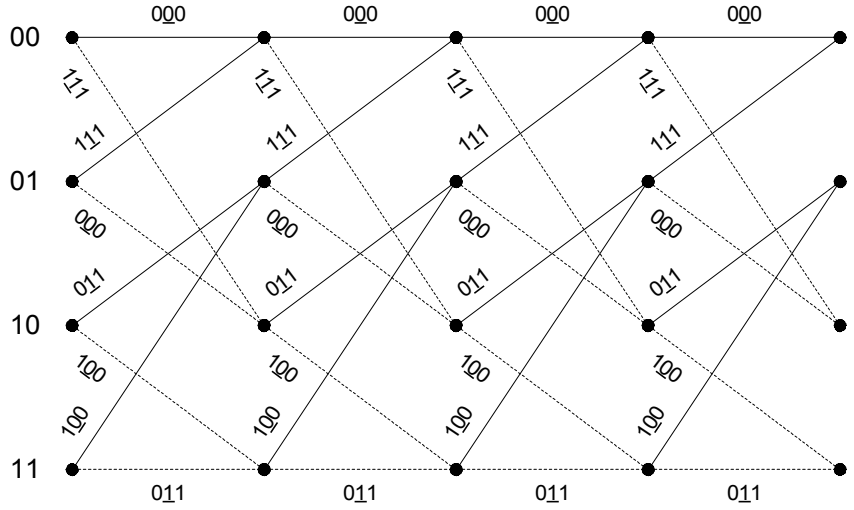

Figura 4.8: Treliça do Código 1 Modificado

Na Figura 4.8, os bits sublinhados representam as posições dos uns do vetor modificador e, portanto estão invertidos em relação aos valores da Figura 4.7. Neste caso particular, as posições dos bits em cada ramo das treliças são iguais em ambas as figuras, pois,

da Tabela 4.1, pode-se ver que, para este código, a ordem das colunas da submatriz $\boldsymbol{S}_i$ permanece inalterada após a modificação.

As simulações foram realizadas considerando um canal com ruído branco gaussiano e a quantização do sinal de saída do canal foi feita com precisão infinita (mais exatamente através de números em ponto flutuante). Na Figura 4.9 é apresentado o desempenho dos códigos convolucionais utilizados como códigos internos:

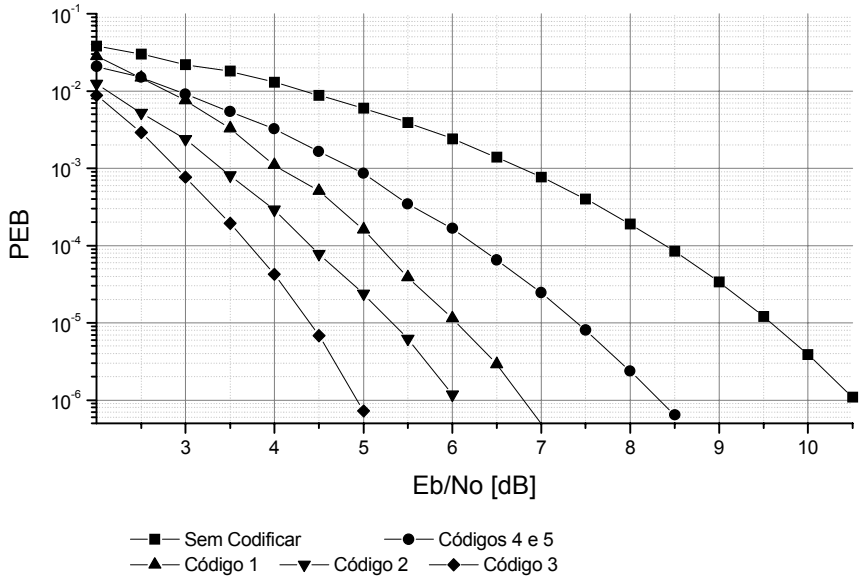

Figura 4.9. Desempenho dos Códigos Internos.

Pode-se comprovar que estas curvas coincidem com as curvas de desempenho dos códigos originais correspondentes [Proakis, 1995].

4.5.2 DECODIFICAÇÃO DO CÓDIGO EXTERNO

A decodificação do código externo de Reed-Solomon foi feita através do algoritmo de Berlekamp-Massey como descrito em [Lin, 1983]. As figuras a seguir mostram o desempenho dos códigos concatenados construídos:

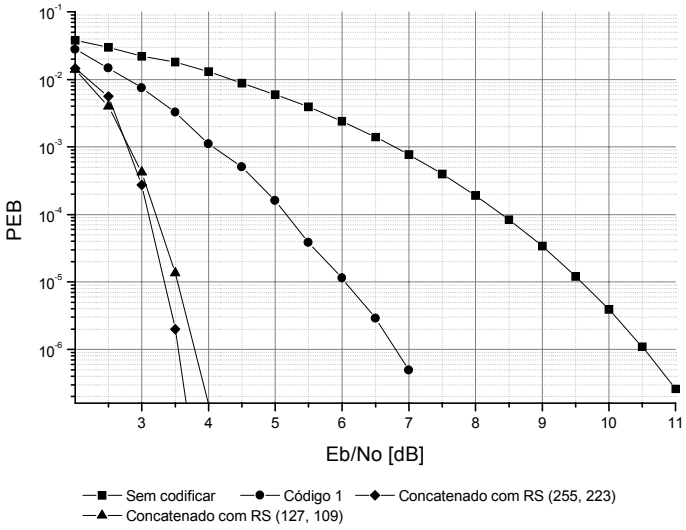

Figura 4.10: Códigos Concatenados usando o Código 1 como Código Interno

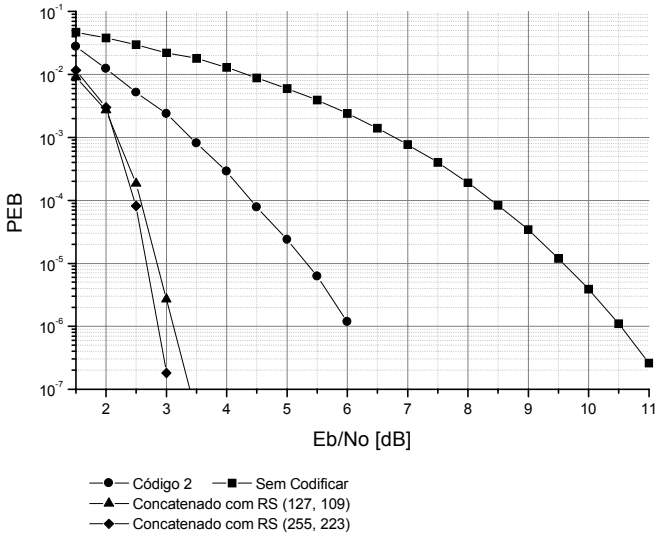

Figura 4.11: Códigos Concatenados usando o Código 2 como Código Interno

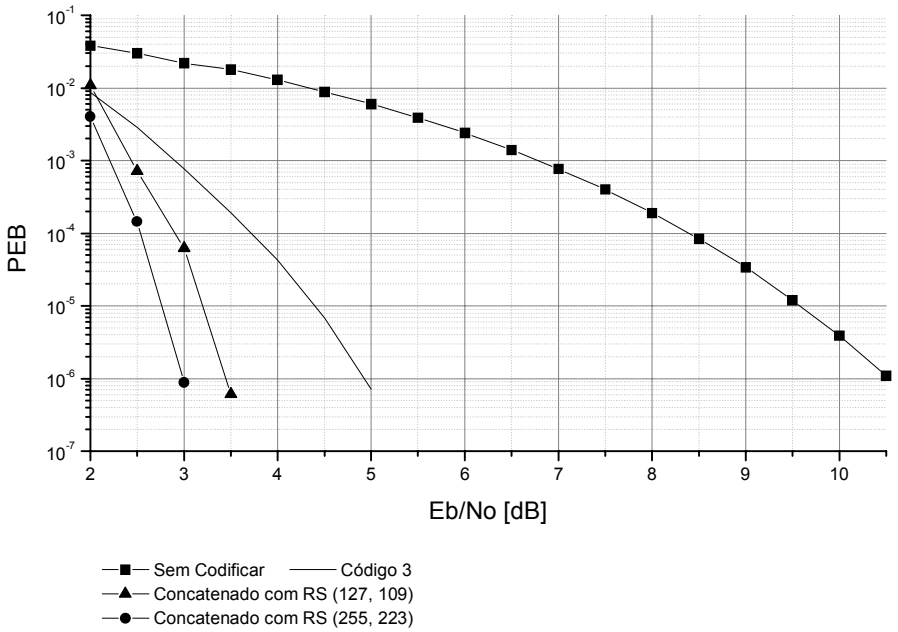

Figura 4.12: Códigos Concatenados usando o Código 3 como Código Interno

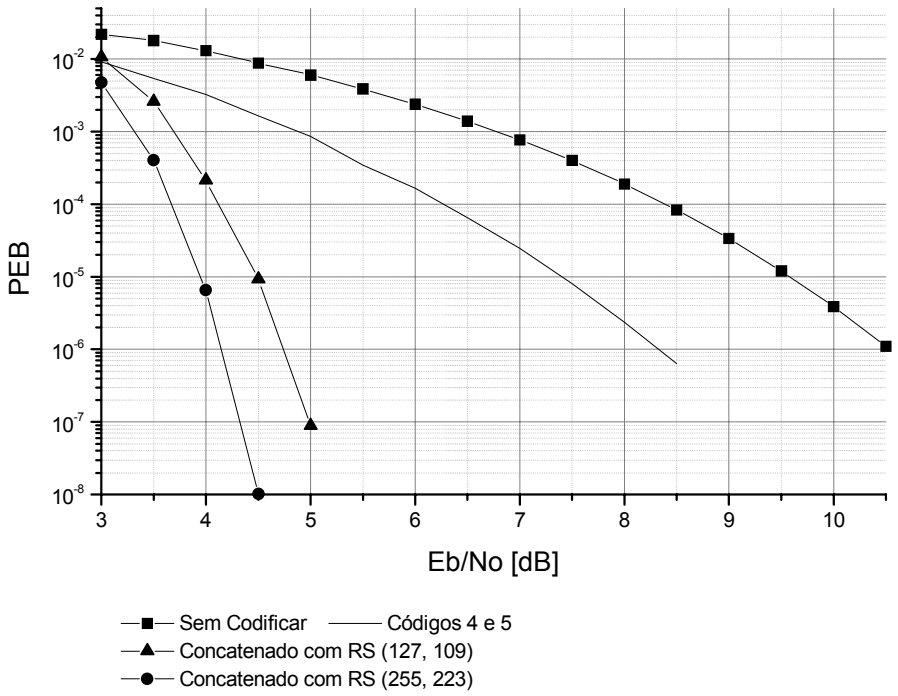

Figura 4.13: Códigos Concatenados usando os Códigos 4 e 5 como Código Interno.

Das figuras anteriores pode-se ver que a probabilidade de erro de bit da ordem de 10^{-5} é atingida com a utilização dos códigos concatenados obtendo-se ganhos de codificação acima de 2dB em relação aos códigos convolucionais internos utilizados em cada esquema. Mais ainda, o ganho de codificação em relação ao canal AWGN sem codificação e considerável. Por exemplo, com a concatenação do código 3 e o código RS (255, 223) se consegue uma probabilidade de erro de bit da ordem de 10^{-5} para uma relação sinal-ruído E_b/N_o de aproximadamente 2,75 dB. Sem a utilização de códigos, seria necessário uma E_b/N_o de 9,6 dB para atingir a mesma probabilidade de erro, obtendo-se neste caso um ganho de codificação de 6,85 dB.

4.6 CONCLUSÃO

Os esquemas concatenados apresentados permitem a obtenção de códigos corretores de erro de grande comprimento com as mesmas restrições de "runlength" apresentadas pelos códigos convolucionais de pequeno comprimento de restrição utilizados como códigos internos. Devido ao fato de que a maioria dos canais com restrições de "runlength" apresentam ruídos em surtos, a utilização de códigos convolucionais em esquemas concatenados torna-se uma opção interessante.

Na decodificação de códigos convolucionais, eventos de erro acontecem toda vez que a palavra-código recebida está mais perto (em termos de distância Euclidiana) de uma palavra-código incorreta do que da palavra-código transmitida. Note-se que as palavras-código de um código convolucional podem ser vistas como caminhos na treliça que representa o código. Assim, quaisquer duas palavras-código vão diferir em um ou mais ramos na treliça a partir do

estado todo zero da mesma. Tendo em conta que essa diferença pode atingir vários ramos consecutivos na treliça, o padrão de erro que provoca uma decisão incorreta pode ser um surto de erros. Portanto, um código convolucional pode ser capaz de corrigir um grande número de erros bem espaçados, mas por outro lado, não é capaz de lidar com erros em surto.

Os códigos externos utilizados neste trabalho operam com símbolos de 7 e 8 bits respectivamente e, portanto, poderão lidar com grande parte dos surtos de erro existentes na saída dos códigos convolucionais internos.

Assim, o trabalho realizado permite dispor de um conjunto de códigos concatenados apropriados para canais com restrições no número de símbolos consecutivos iguais a serem transmitidos, sendo estas restrições garantidas pelos códigos internos utilizados. Os códigos externos utilizados em todos os casos foram códigos de Reed-Solomon. Como códigos internos foram selecionados os códigos convolucionais apresentados em [Šechny, 1999] aos quais foram feitas algumas modificações que simplificam os processos de codificação e decodificação. Para o caso de uma implementação prática destes códigos, estas modificações são válidas desde que seja possível ter acesso ao interior dos codificadores e decodificadores. Por exemplo, no caso de aplicações em que estas funções sejam realizadas por circuitos integrados comerciais, é preferível utilizar o esquema mostrado na Figura 4.4.

Entre os códigos convolucionais internos utilizados encontram-se os melhores códigos convolucionais de taxa 1/3 e 1/4.

CAPÍTULO 5

PROPRIEDADES ESPECTRAIS DOS CÓDIGOS COM "RUNLENGTH" LIMITADO

5.1 INTRODUÇÃO

A maior parte dos sistemas de comunicações que precisam da utilização de códigos com "runlength" limitado, também requer que as seqüências binárias produzidas por estes códigos tenham seu espectro anulado na freqüência zero. Códigos com anulação do espectro na freqüência zero são conhecidos como códigos "dc-free" ou códigos "dc-balanceados".

Desde o início dos sistemas de comunicações digitais através de cabos, os códigos "dc-balanceados" têm sido utilizados para conter os efeitos da ausência de resposta em baixa

freqüência destes sistemas devido aos componentes de acoplamento, transformadores de isolamento, etc. Em sistemas de gravação magnética e óptica, os códigos "dc-balanceados" são utilizados para reduzir a interação entre os dados gravados no disco e os servossistemas utilizados na gravação. Por exemplo, distúrbios de baixa freqüência provocados por marcas de dedos nos discos, podem causar erros de leitura se o sinal cai abaixo do nível de decisão [Immink, 1999]. Erros deste tipo são evitados com a utilização de filtragem passa alto, a qual só pode ser feita se a seqüência codificada não possuir conteúdo de baixa freqüência, ou seja, se a seqüência for "dc-balanceada".

A conversão de seqüências binárias arbitrárias em seqüências "dc-balanceadas" necessariamente leva a uma diminuição na taxa de transmissão de dados. Neste capítulo será abordada a obtenção de códigos "dc-balanceados" a partir dos códigos corretores de erro com "runlength" limitado construídos nos capítulos anteriores, minimizando a perda na taxa na codificação.

5.2 PROPRIEDADES DOS CÓDIGOS "DC-BALANCEADOS"

A soma digital corrida, RDS ("running digital sum"), tem um papel importante na análise e síntese de códigos que apresentam redução do conteúdo espectral em baixa freqüência. Seja

$$\{x_i\} = \{\ldots, x_{i-1}, x_0, \ldots x_i, \ldots\}, \quad x_i \in \{-1, 1\}$$

uma seqüência binária. A soma digital corrida z_i é definida como [Immink, 1999]:

$$z_i = \sum_{j=-\infty}^{i} x_j = z_{i-1} + x_i \tag{5.1}$$

Se z_i é limitada, pode-se provar [Immink, 1999] que a densidade espectral de potência da seqüência $\{x_i\}$ é anulada na freqüência zero. Pierobon [Pierobon, 1984] mostrou que a densidade espectral de potência de uma seqüência $\{x_i\}$ é anulada na freqüência zero, se e só se, o codificador é um codificador de soma digital corrida finita. Desta forma, a supressão das componentes de baixa freqüência pode ser conseguida através da limitação da soma digital corrida da seqüência codificada.

Esquemas de codificação práticos projetados para se obter supressão das componentes de baixa freqüência geralmente são baseados em códigos de bloco. Neles, os bits de entrada do codificador são agrupados em blocos de *m* bits que posteriormente serão transformados em blocos de *n* bits utilizando-se tabelas de conversão. Entre estes esquemas, os mais utilizados na atualidade são os códigos de disparidade zero, códigos de baixa disparidade e códigos com bit de polaridade [Immink, 1999].

A disparidade de uma palavra-código binária é definida como a diferença entre o número de uns e o número de zeros contidos nela. Assim, as palavras-código '000110' e '100111' tem disparidade -2 e $+2$ respectivamente. Palavras-código com igual número de uns e de zeros são conhecidas como palavras-código com disparidade zero. Desta forma, códigos "dc-balanceados" podem ser construídos a partir de palavras-código com disparidade zero que tenham uma correspondência um-a-um com as palavras da fonte de informação.

Seja $\boldsymbol{x}$ uma palavra-código de n bits, n par, formada por símbolos bipolares x_i, $1 \le i \le n$, $x_i \in \{-1, 1\}$. A disparidade $d(\boldsymbol{x})$ da palavra-código é a soma dos símbolos contidos nela, ou

$$d(\boldsymbol{x}) = \sum_{i=1}^{n} x_i \tag{5.2}$$

Se todas as palavras-código tiver igual número de uns e de zeros, então $d(\boldsymbol{x}) = 0$. O número de palavras-código com disparidade zero, N_0, e comprimento n (n par) pode ser calculado através do coeficiente binomial:

$$N_0 = \binom{n}{n/2} = \frac{n!}{\frac{n}{2}!\frac{n}{2}!} \tag{5.3}$$

A taxa do código, R, é dada por

$$R = \frac{1}{n} \log_2 N_0 \tag{5.4}$$

O procedimento de construção de códigos com disparidade zero pode ser aplicado na obtenção de códigos de baixa disparidade onde a correspondência entre as palavras-código e as palavras da fonte não é necessariamente um para um. Apenas as palavras-código com disparidade zero guardam uma correspondência um para um com palavras da fonte. As demais palavras-código são alocadas em duplas de polaridade invertida fazendo com que um determinado número de palavras da fonte possa ser representado de duas formas diferentes. O decodificador interpreta estas duas formas de representação da mesma maneira. Durante a

transmissão, a escolha da palavra-código a ser transmitida é feita de forma a fazer com que a disparidade acumulada, ou a soma digital corrida, da seqüência codificada após a transmissão de cada palavra-código, esteja o mais perto possível do valor zero. A soma digital corrida para uma seqüência binária é definida como a soma acumulada de uns e zeros (um zero é considerado como -1) contados desde o início da transmissão.

Neste tipo de código, a tabela de conversão é composta por dois conjuntos de palavras-código ou páginas chamados de S_+ e S_-, respectivamente. O conjunto S_+ é formado por palavras-código com disparidade zero e com disparidade positiva e o conjunto S_- por palavras-código com disparidade zero e negativa. O conjunto S_+ contém $(K+1)$ subconjuntos chamados de $S_0, S_1, \ldots, S_K$, $(K \le n/2)$. Os elementos do subconjunto S_j são todas as possíveis palavras-código com disparidade $2j$, $0 \le j \le K$. As palavras-código do conjunto S_- podem ser formadas através da inversão bit a bit das palavras-código do conjunto S_+ e vice-versa. A cardinalidade N_j do subconjunto S_j é dada por,

$$N_j = \binom{n}{n/2+j} \tag{5.5}$$

O número total *M* de palavras-código possíveis no subconjunto S_+ (e, por simetria, no subconjunto S_-), é dado por:

$$M = |S_+| = |S_-| = \sum_{j=0}^{K} N_j \tag{5.6}$$

A taxa do código é dada por,

$$R = \frac{1}{n}\log_2 M \tag{5.7}$$

Mais detalhes, tanto dos códigos de disparidade zero como dos códigos de baixa disparidade, podem ser encontrados em [Cattermole, 1983] e [Griffiths, 1969].

Os códigos com bit de polaridade foram projetados por Bowers [Bowers, 1960] e Carter [Carter, 1965], que propuseram um método de construção de códigos "dc-balanceados" ligeiramente diferente dos métodos anteriores que não utiliza tabelas de conversão nos processos de codificação e decodificação. Eles propuseram um código onde aos $(n-1)$ símbolos da fonte é adicionado um símbolo chamado de bit de polaridade. Este bit inicialmente tem o valor '1'. O codificador tem a opção de transmitir a palavra de *n* bits inalteradamente ou de inverter todos os símbolos. A escolha da opção a ser utilizada pelo codificador é feita de tal forma que a disparidade acumulada esteja o mais perto possível do valor zero. Se a disparidade acumulada no início da transmissão de uma palavra-código e a disparidade da palavra a ser transmitida têm o mesmo sinal, então todos os bits da palavra-código (incluindo o bit de polaridade) são invertidos antes de serem transmitidos, caso contrário, a palavra-código é transmitida inalterada. Se a disparidade da palavra-código a ser transmitida é zero, esta é invertida com probabilidade ½. O bit de polaridade é utilizado pelo decodificador para determinar se a palavra-código transmitida foi invertida ou não. A taxa dos códigos com bit de polaridade é dada por:

$$R = 1 - \frac{1}{n} \tag{5.8}$$

5.3 TRANSFORMAÇÃO DE CÓDIGOS DE BLOCO CORRETORES DE ERRO EM CÓDIGOS DC-BALANCEADOS

As características espectrais dos códigos de bloco concatenados construídos no Capítulo 2 são determinadas pelos códigos internos. Estes códigos são códigos de bloco com "runlength" limitado e é fácil comprovar que as seqüências codificadas obtidas a partir deles não têm um valor limitado de soma digital corrida, portanto, a densidade espectral de potência destes códigos não é nula na freqüência zero. Mais ainda, o conteúdo espectral destes códigos na região de baixas freqüências é relativamente alto. Por exemplo, na Figura 5.1 é mostrada a densidade espectral de potência (PSD-"power spectral density") do código BCH (15, 5, 7) modificado utilizando-se uma representação antipodal dos símbolos do sinal codificado.

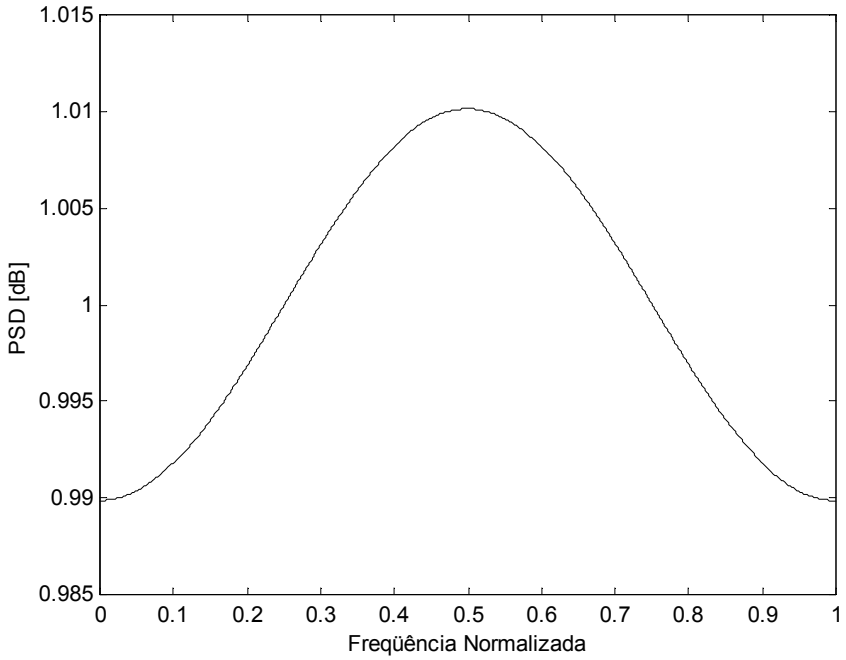

Figura 5.1: Densidade Espectral de Potência do Código BCH (15, 5, 7) Modificado com Representação Antipodal da Seqüência Codificada

Os códigos de bloco com "runlength" limitado foram obtidos a partir da soma de um vetor modificador às palavras-código de códigos de blocos corretores de erros lineares e transparentes. Assim, as palavras-código, tanto dos códigos originais como dos códigos modificados, podem ser agrupadas em pares de palavras-código de polaridade oposta mantendo-se a correspondência um para um entre palavras da fonte e palavras-código. Isto pode ser garantido devido ao fato destes códigos serem transparentes. Portanto, tendo em conta a teoria exposta na seção anterior, é possível a obtenção de códigos "dc-balanceados" a partir dos códigos de bloco com "runlength" limitado através da utilização do método de codificação com bit de polaridade.

Na Tabela 5.1 são mostrados os códigos "dc-balanceados" construídos a partir dos códigos de bloco concatenados. De acordo com a equação (5.8), pode-se ver que a redução na taxa, após a inclusão do bit de polaridade, é praticamente insignificante. Por outro lado, devido à inclusão do bit de polaridade, o valor de MinRL_{MAX} é incrementado em apenas uma unidade em relação ao valor apresentado pelos códigos concatenados originais. Desta forma, os códigos "dc-balanceados" obtidos mantêm boas propriedades quanto à limitação de "runlength".

Tabela 5.1: Códigos "DC-Balanceados" Construídos a partir de Códigos Concatenados com "Runlength" Limitado

Código Concatenado com "Runlength" Limitado	Taxa	MinRL_{MAX}	Código Concatenado "DC-Balanceado"	Taxa	MinRL_{MAX}
(105, 44, 15)	0,42	7	(112, 44, 15)	0,39	8
(105, 36, 21)	0,34	7	(112, 36, 21)	0,32	8
(465, 145, 21)	0,31	6	(496, 145, 21)	0,29	7
(465, 135, 35)	0,29	6	(496, 135, 35)	0,27	7
(465, 125, 49)	0,26	6	(496, 125, 49)	0,25	7
(1905, 763, 95)	0,4	7	(2032, 763, 95)	0,38	8
(1905, 749, 105)	0,393	7	(2032, 749, 105)	0,37	8
(120, 44, 20)	0,367	6	(135, 44, 20)	0,33	7
(120, 36, 28)	0,3	6	(135, 36, 28)	0,27	7
(496, 145, 24)	0,29	6	(527, 145, 24)	0,28	7
(496, 125, 56)	0,25	6	(527, 125, 56)	0,24	7
(345, 132, 35)	0,38	14	(360, 132, 35)	0,37	15
(345, 108, 49)	0,31	14	(360, 108, 49)	0,30	15

5.4 DENSIDADE ESPECTRAL DE POTÊNCIA DOS CÓDIGOS DE BLOCO "DC-BALANCEADOS"

A análise espectral de seqüências codificadas em blocos é feita levando em conta o fato de que as propriedades estatísticas destas seqüências são, em geral, ciclo-estacionárias, ou seja, variam periodicamente com um período igual ao comprimento do bloco codificado. Seja $\mathbf{x}_j = (x_{j,1}, \ldots, x_{j,n})$ a j-ésima palavra-código transmitida cujos símbolos $x_{i,j}$, $1 \le i \le n$ são

transmitidos na forma serial. Assim, a equação geral de uma seqüência codificada em blocos é dada por:

$$X(t) = \sum_{j=-\infty}^{\infty} \sum_{i=1}^{n} x_{j,i}\, s[t - (jn + i - 1)] \tag{5.9}$$

onde $s(t)$ representa o pulso utilizado para a transmissão da seqüência.

Os códigos tratados neste trabalho são códigos de bloco sem memória, ou seja, são códigos em que existe uma correspondência um-a-um entre as palavras na entrada do codificador e as palavras codificadas.

Suponha, para um código de bloco, um conjunto X de M palavras-código e que o u-ésimo elemento do conjunto de palavras-código seja $\boldsymbol{x}_u = \left(x_1^{(u)}, \ldots, x_n^{(u)}\right),\ 0 \le u \le M-1$. As palavras-código são escolhidas aleatoriamente a partir do conjunto X formando-se uma seqüência infinita que é transmitida em forma serial. Immink [Immink, 1999] mostrou que a densidade espectral de potência desta seqüência é dada por:

$$H(\omega) = H_c(\omega) + H_d(\omega) \sum 2\pi\delta(\omega - 2\pi\, k/n) \tag{5.10}$$

onde $H_c(\omega)$ e $H_d(\omega)$ representam as componentes contínua e discreta da densidade espectral de potência, dadas por [Immink, 1999]:

$$H_c(\omega) = \frac{1}{n}\left\{\mu_0 + 2\sum_{k=1}^{n-1} \mu_k \cos k\omega\right\} \tag{5.11}$$

e

$$H_d(\omega) = \frac{1}{n^2}\left\{\sum_{i=1}^{n} \nu_i^2 + 2\sum_{k=1}^{n-1}\sum_{i=1}^{n-k} \nu_i \nu_{i+k} \cos k\varpi\right\}, \tag{5.12}$$

onde

$$\nu_k = \frac{1}{M}\sum_{u=0}^{M-1} x_k^{(u)}, \quad 1 \le k \le n \tag{5.13}$$

e

$$\mu_k = \frac{1}{M}\sum_{i=1}^{n-k}\sum_{u=0}^{M-1}\left(x_i^{(u)} - \nu_i\right)\left(x_{i+k}^{(u)} - \nu_{i+k}\right), \quad 0 \le k \le n-1 \tag{5.14}$$

Da equação (5.12) pode-se verificar que a componente discreta do espectro $H_d(\omega)$ é igual a zero se $\nu_k = 0,\ k = 1, \ldots, n-1$, ou seja, a densidade espectral de potência de um código de bloco sem memória não contém linhas espectrais se para todo i, $1 \le i \le n$, o número de palavras-código $\boldsymbol{x}_u \in X$, onde $x_i^{(u)} = 1$ é igual ao o número de palavras-código em que $x_i^{(u)} = -1$, o que equivale a dizer que a soma de todas as palavras-código de X dá como resultado o vetor todo zero.

Devido aos códigos de bloco utilizados neste trabalho serem transparentes, a condição descrita no parágrafo anterior é sempre satisfeita. Isto garante que os códigos "dc-balanceados" mostrados na Tabela 5.1 apresentem uma densidade espectral de potência livre de componentes discretas. A presença de componentes discretas significa um gasto desnecessário de potência na transmissão da seqüência codificada.

Considerando $H_d(\omega) = 0$, a densidade espectral de potência de um código de bloco sem memória pode ser expressa de uma forma mais simplificada. A transformada de Fourier $X^{(u)}(\omega)$ da palavra-código $\boldsymbol{x}_u$ é definida por:

$$X^{(u)}(\omega) = \sum_{i=1}^{n} x_i^{(u)} e^{-j\omega}, \tag{5.15}$$

onde $j = \sqrt{-1}$.

Considerando novamente que as palavras-código são transmitidas serialmente de forma aleatória, a densidade espectral de potência $H(\omega)$ é dada por [Immink, 1999]:

$$H(\omega) = \frac{1}{Mn} \sum_{u=0}^{M-1} \left| X^{(u)}(\omega) \right|^2 \tag{5.16}$$

Da equação (5.16) pode-se ver que a densidade espectral de potência da seqüência codificada é a média das densidades espectrais de potência das palavras do código.

5.5 RESULTADOS DAS SIMULAÇÕES REALIZADAS

Tendo em conta o significado da equação (5.16), foi simulada a densidade espectral de potência dos códigos "dc-balanceados" utilizando-se o método das médias de peridiogramas de Welch [Oppenheim, 1989].

Neste método, a seqüência de símbolos codificados é dividida em segmentos de *n* amostras com uma janela de comprimento *L* aplicada a cada um desses segmentos. A seguir, é calculada a estimativa da densidade espectral de potência de cada segmento e logo após a

média sob todos os segmentos. As figuras a seguir mostram a região de baixas freqüências das curvas de densidade espectral de potência obtidas a partir das simulações realizadas utilizando os códigos "dc-balanceados" da Tabela 5.1.

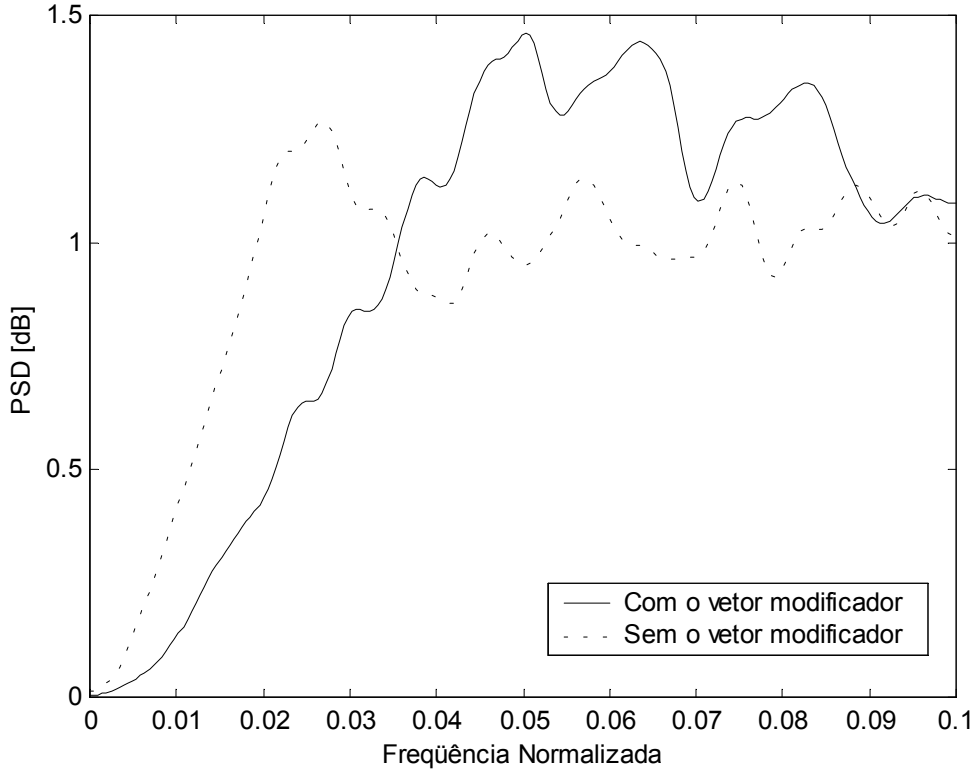

Figura 5.2: Densidade Espectral de Potência dos Códigos de Comprimento 112.

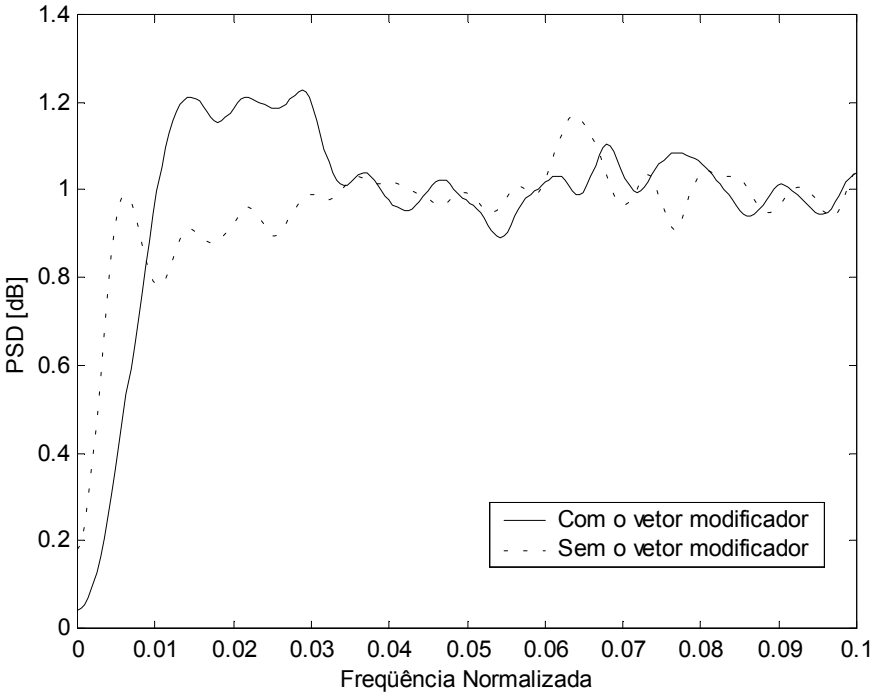

Figura 5.3: Densidade Espectral de Potência dos Códigos de Comprimento 496.

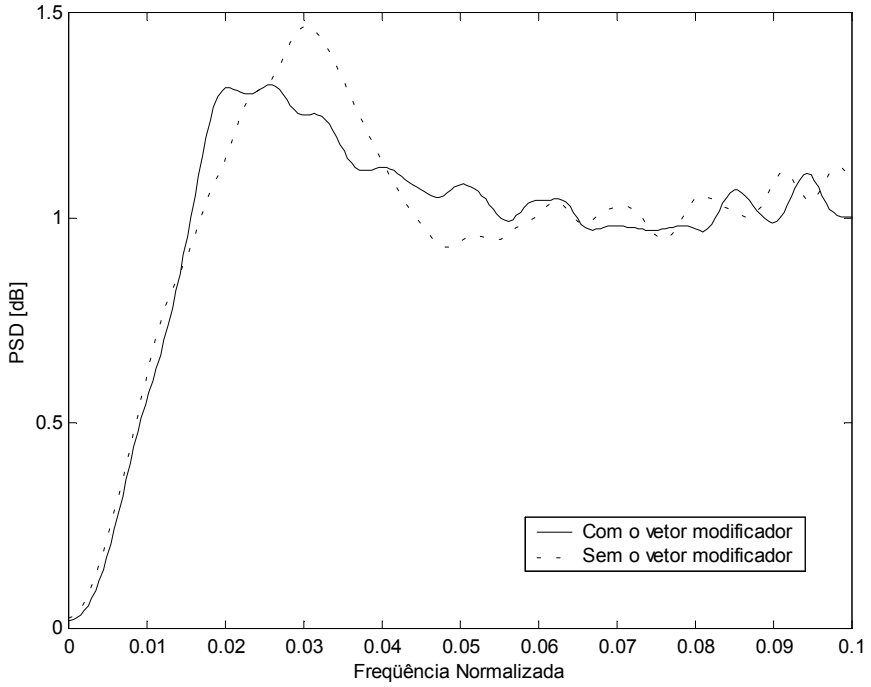

Figura 5.4: Densidade Espectral de Potência dos Códigos de Comprimento 2032.

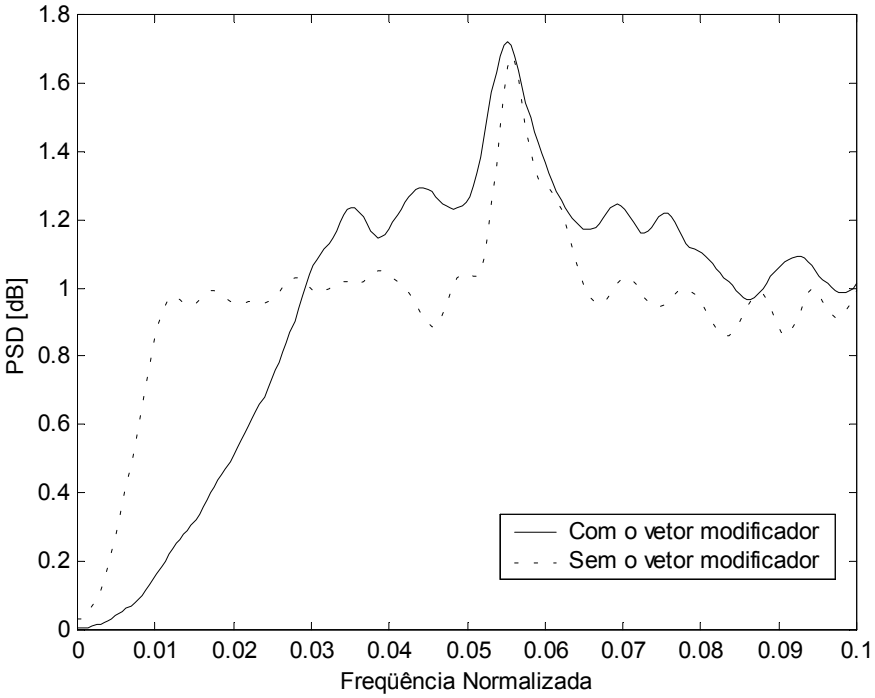

Figura 5.5: Densidade Espectral de Potência dos Códigos de Comprimento 135.

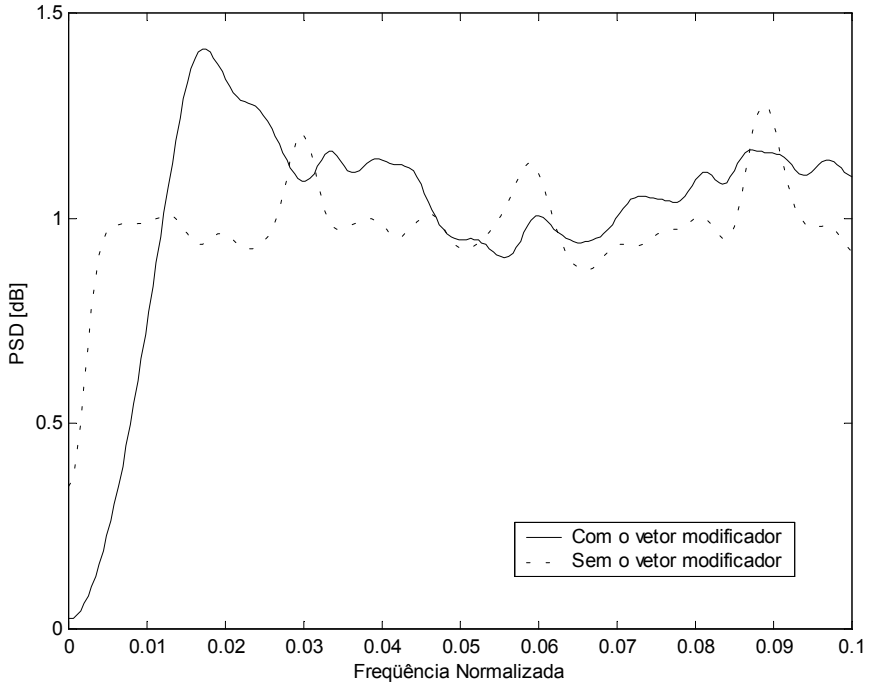

Figura 5.6: Densidade Espectral de Potência dos Códigos de Comprimento 527.

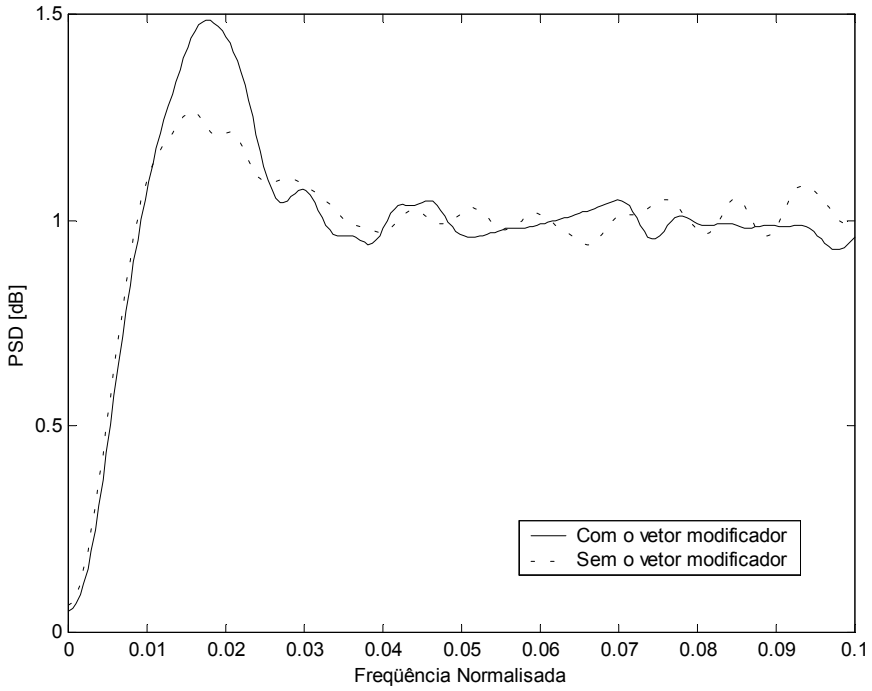

Figura 5.7: Densidade Espectral de Potência dos Códigos de Comprimento 360.

A partir das figuras anteriores pode-se observar que a utilização do vetor modificador faz com que a densidade espectral de potência seja praticamente zero na freqüência zero. Mais ainda, com a utilização do vetor modificador, o conteúdo espectral na região de freqüências baixas é menor para a maior parte dos códigos simulados. Esta diferença pode ser explicada pelo fato de que a composição dos símbolos das seqüências codificadas é determinada pelos códigos internos utilizados nos esquemas de codificação concatenada. Uma vez que estes códigos são códigos corretores de erro, algumas seqüências binárias de comprimento *n* não são possíveis de acontecer. Esta diferença é mais acentuada nos códigos de comprimento 135 da Tabela 5.1, os quais utilizam como código interno o código RM (16, 5, 8). As figuras a seguir mostram o espectro de pesos do código RM (16, 5, 8) original e modificado, respectivamente.

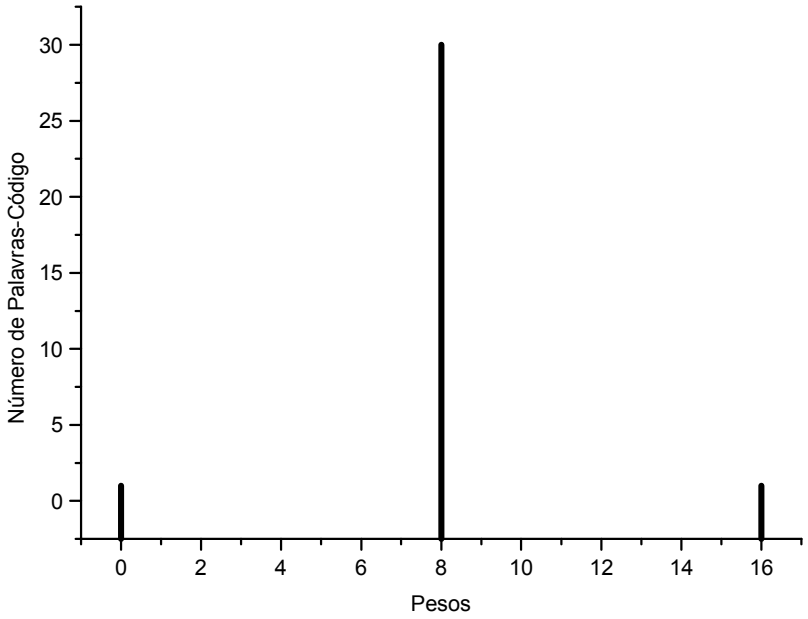

Figura 5.8: Espectro de Pesos do Código RM (16, 5, 8) Original.

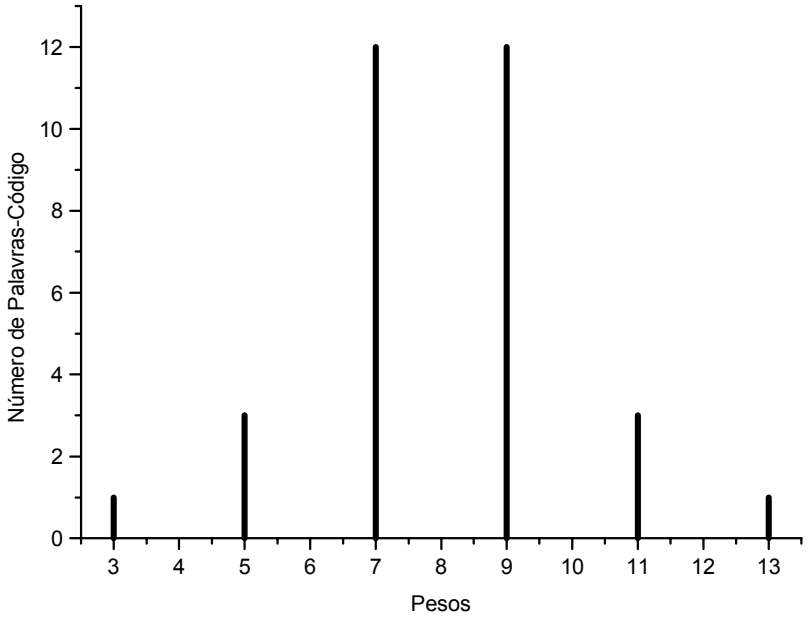

Figura 5.9: Espectro de Pesos do Código RM (16, 5, 8) Modificado.

A Figura 5.9 mostra que as seqüências produzidas pelo código modificado são de certa forma mais "aleatórias" do que as produzidas pelo código original, onde só acontecem seqüências com oito uns além das seqüências toda zeros e toda uns. Isto sem dúvidas altera a forma das curvas de densidade espectral de potência.

5.6 CÓDIGOS CONVOLUCIONAIS CONCATENADOS "DC-BALANCEADOS"

Os códigos convolucionais com "runlength" limitado apresentados no Capítulo 4 não são transparentes e, portanto, o procedimento aplicado neste capítulo para a obtenção de códigos de bloco "dc-balanceados" não procede para estes códigos.

Códigos "dc-balanceados" a partir dos códigos convolucionais com "runlength" limitado podem ser obtidos utilizando uma representação dos símbolos codificados do tipo AMI ("Alternate Mark Inversion"), onde os símbolos binários zeros são representados como 0 e os símbolos binários uns são representados alternadamente como $+1$ e -1. Desta forma, a seqüência codificada é "dc-free" ao custo de uma perda na eficiência da transmissão devido à utilização de três níveis diferentes para a representação da seqüência binária.

As figuras a seguir mostram as curvas de densidade espectral de potência dos códigos apresentados na Tabela 4.2 do Capítulo 4. Estas curvas foram obtidas através de simulações da densidade espectral de potência dos códigos convolucionais concatenados a partir da representação dos símbolos das seqüências codificadas por sinais do tipo AMI e do tipo BPNRZ. No BPNRZ ("Bipolar Non Return to Zero") o símbolo binário zero é representado pelo símbolo -1 e o símbolo binário um é representado pelo símbolo $+1$.

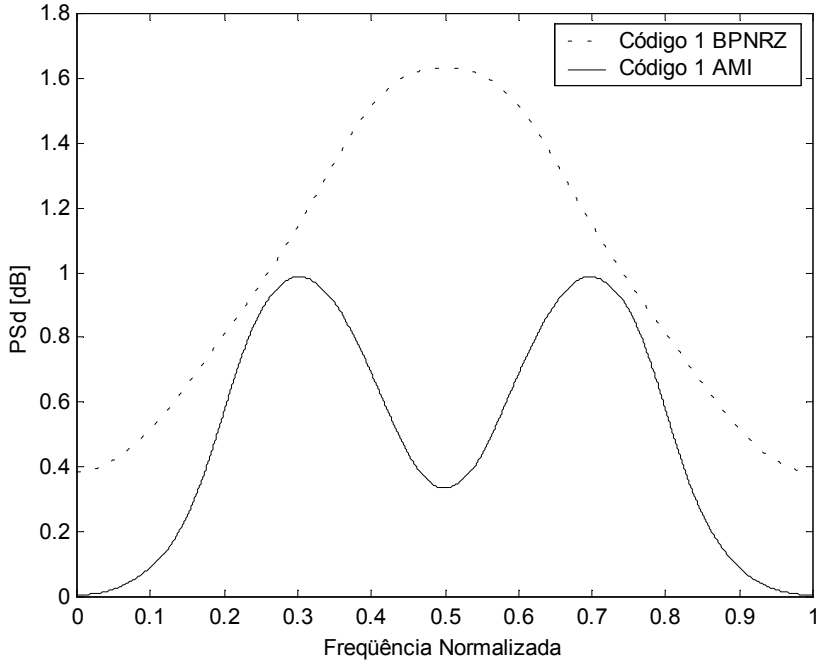

Figura 5.10: Densidade Espectral de Potência dos Códigos Convolucionais Concatenados que Utilizam o Código 1 como Código Interno

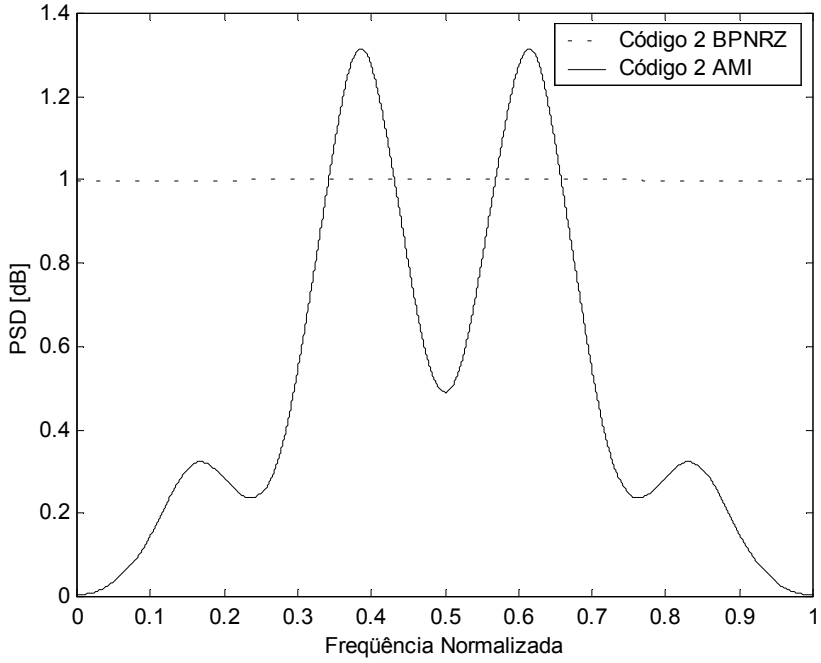

Figura 5.11: Densidade Espectral de Potência dos Códigos Convolucionais Concatenados que Utilizam o Código 2 como Código Interno

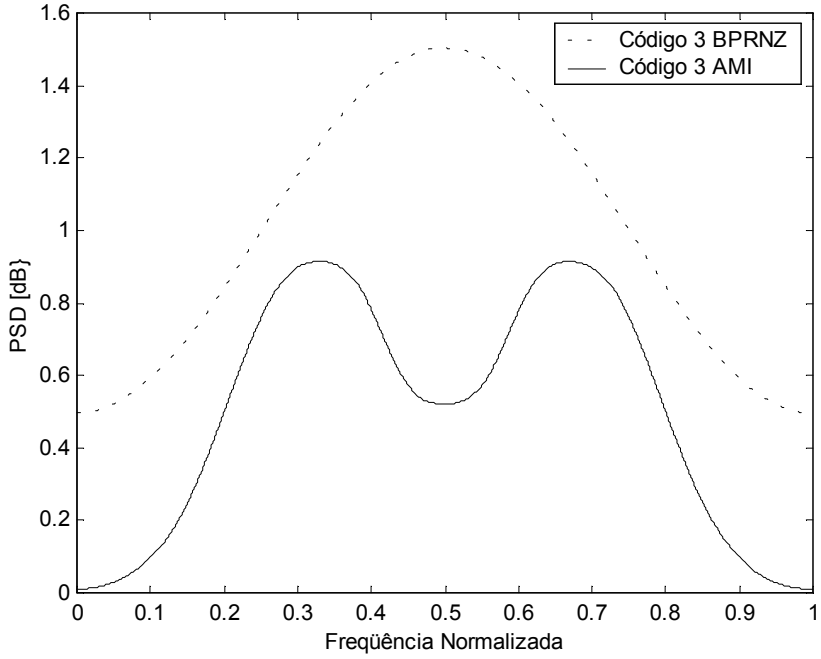

Figura 5.12: Densidade Espectral de Potência dos Códigos Convolucionais Concatenados que Utilizam o Código 3 como Código Interno

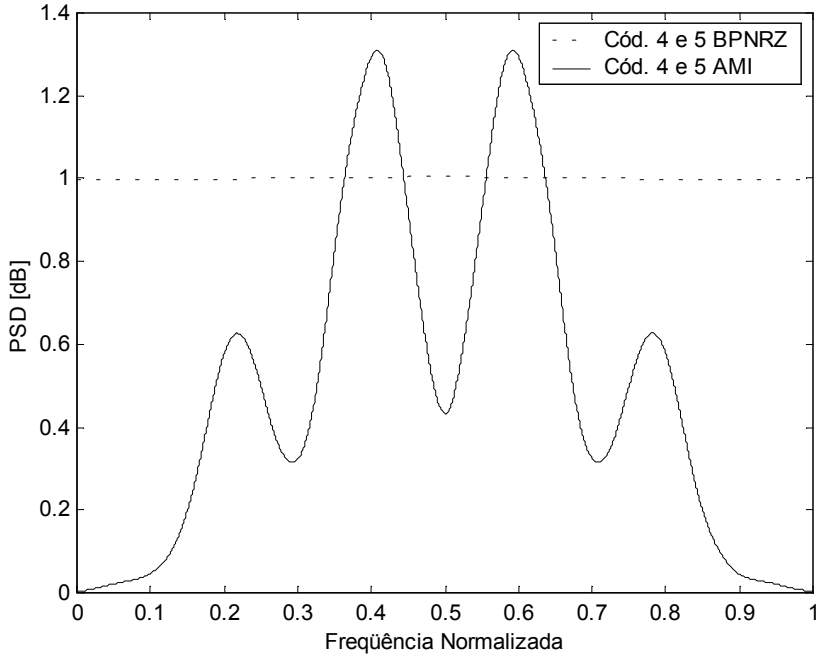

Figura 5.13: Densidade Espectral de Potência dos Códigos Convolucionais Concatenados que Utilizam os Códigos 4 e 5 como Código Interno

Das figuras anteriores pode-se notar que quando da utilização de sinais do tipo BPNRZ, em alguns casos se consegue uma supressão significativa do conteúdo espectral na região das freqüências baixas. Esta supressão é mais acentuada quando da utilização do código 1 no esquema de codificação concatenada. Com a representação das seqüências codificadas através de sinais do tipo AMI, a densidade espectral de potência é anulada na freqüência zero em todos os casos.

5.7 CONCLUSÃO

Neste capítulo foi mostrado como é possível obter códigos "dc-balanceados" a partir dos códigos corretores de erro com "runlength" limitado. Desta forma, estes códigos, além de apresentar restrições no número máximo de símbolos iguais consecutivos nas seqüências codificadas, agora apresentam também restrições no conteúdo espectral na região de freqüências baixas para a densidade espectral de potência destas seqüências. Os sistemas de comunicações que utilizam códigos com restrições, geralmente precisam de ambos os tipos de restrições descritas, isto é, "runlength" limitado e densidade espectral de potência nula para baixas freqüências.

Para os códigos construídos, estas restrições atingem bons valores a um custo baixo, quanto à taxa de transmissão de dados, e mantém inalteradas as propriedades de correção de erro dos códigos originais utilizados.

Devido ao fato de os códigos de bloco com "runlength" limitado serem transparentes, os resultados obtidos para estes códigos no que se refere às propriedades espectrais são superiores aos obtidos para códigos convolucionais pois, o nulo espectral quando da utilização

de códigos convolucionais foi conseguido através da utilização de um sinal de três níveis do tipo AMI, o que reduz o desempenho do código.

CAPÍTULO 6

CONSIDERAÇÕES FINAIS

A partir dos resultados obtidos nesta tese, dispõe-se de um conjunto de códigos corretores de erro, que apresentam determinadas propriedades de codificação de linha, que os tornam apropriados para serem utilizados em sistemas de comunicações digitais onde as seqüências de símbolos transmitidas necessitam cumprir com determinadas restrições. Especificamente, as seqüências de símbolos de saída dos códigos construídos neste trabalho têm um valor limitado do máximo "runlength" e, ao mesmo tempo, suas curvas de densidade espectral de potência têm baixo conteúdo na região de baixas freqüências, chegando a ser nulo na freqüência zero, dependendo do tipo de sinal utilizado para enviar as seqüências

codificadas pelo canal de comunicações. Estas propriedades são úteis em sistemas de comunicações digitais de banda básica e sistemas de gravação magnética e óptica.

Os códigos foram construídos através da concatenação de códigos não binários de Reed-Solomon com códigos corretores de erro binários (de bloco e convolucionais) com "runlength" limitado. Os códigos internos, por sua vez, foram obtidos a partir de códigos de bloco e códigos convolucionais curtos modificados. Especificamente, foram utilizados "cosets" de códigos de bloco e convolucionais escolhidos de forma tal que as seqüências de símbolos de saída destes "cosets" apresentaram um valor limitado do "runlength" máximo. A escolha do vetor modificador (ou líder de "coset") foi feita fazendo com que a quantidade de uns seja a menor possível. Isto faz com que o aumento na complexidade nos processos de codificação e decodificação seja mínimo.

Desta forma, foi possível obter códigos corretores de erro com "runlength" limitado de comprimento e capacidade de correção de erro elevada sem necessidade de utilizar inserção de bits nas seqüências codificadas e, portanto, mantendo inalterada a taxa de codificação em relação aos códigos originais utilizados. A capacidade de correção de erro dos códigos obtidos é bem maior do que a apresentada por outros códigos com "runlength" limitado encontrados na literatura. A obtenção de códigos corretores de erro concatenados com "runlength" limitado e com o código interno tendo capacidade de correção de erro não é encontrada na literatura especializada, o que caracteriza uma contribuição original deste trabalho.

Com a utilização de códigos internos com capacidade de correção de erro, a propagação de erros geralmente inerente à utilização de códigos de linha convencionais é bem menor e pode ser minimizada ou mesmo eliminada pelo código de Reed-Solomon externo.

Devido à utilização de códigos não binários de Reed-Solomon nos esquemas de codificação concatenada, os códigos projetados podem corrigir tanto erros aleatórios como surtos de erros. O tamanho dos surtos de erros corrigíveis pode ser aumentado através do entrelaçamento dos símbolos dos códigos de Reed-Solomon. Este tipo de erro é comum em sistemas que se utilizam de discos magnéticos ou ópticos.

Simulações realizadas com a utilização destes códigos no canal Gaussiano levaram a obtenção de resultados que concordam com a performance dos códigos originais utilizados. Os algoritmos de codificação e decodificação utilizados nas simulações foram elaborados em linguagem C.

Por último, procedimentos para transformar os códigos com "runlength" limitado em códigos "dc-free" foram elaborados. No caso da utilização de códigos de bloco internos, devido ao fato destes códigos serem transparentes, foi possível transformar as seqüências codificadas antipodais em seqüências "dc-free" com a simples utilização de um bit adicional entre palavras do código interno com a função de manter limitado o valor da soma digital corrida durante o processo de transmissão. A inserção deste bit levou a uma pequena diminuição da taxa de transmissão de dados e a um incremento de uma unidade do "runlength" máximo. Sendo N o valor absoluto máximo alcançado pela soma digital corrida durante o processo de transmissão, foi possível comprovar que $\mathrm{Min}RL_{MAX} \leq N-2$ para todos os códigos utilizados. Este resultado é similar aos resultados apresentados por códigos com "runlength" limitado sem capacidade de correção de erro [Immink, 1999]. Entendemos que a transformação de códigos de bloco corretores de erro em códigos "dc-free" utilizando o mesmo conjunto de palavras do código original e, portanto, com uma redução mínima da taxa

de transmissão de dados é também um procedimento novo em relação aos procedimentos convencionais para a obtenção destas classes de códigos.

Quando da utilização de códigos convolucionais internos, através da representação das seqüências codificadas por sinais do tipo AMI, também foi possível a obtenção de seqüências "dc-free". Neste caso a eficiência da transmissão diminui devido à utilização de um sinal de três níveis.

Estas propriedades espectrais foram simuladas com a utilização de periodiogramas, dando como resultado a constatação de que as curvas de densidade espectral de potência dos esquemas que utilizam códigos de bloco modificados são anuladas na freqüência zero e apresentam uma redução considerável do conteúdo espectral na região de baixas freqüências em relação às curvas dos códigos originais. Estas simulações foram feitas utilizando o programa MATLAB.

Futuros trabalhos nesta área podem ser desenvolvidos visando à obtenção de códigos que apresentem valores limitados tanto do "runlength" máximo como do "runlength" mínimo. Estes códigos teriam a propriedade adicional de serem adequados para canais de comunicações com altos níveis de interferência intersimbólica.

Também pode ser objeto de pesquisa a transformação de códigos de bloco corretores de erro de grande comprimento em códigos com "runlength" limitado através de algoritmos de modificação que visem à limitação do número de símbolos consecutivos iguais em pequenas secções das palavras-código de grande comprimento. Desta forma, os processos de codificação e decodificação poderiam ser mais simples que no caso da utilização de códigos concatenados.

A utilização de entrelaçamento nos esquemas de codificação concatenada que utilizam códigos convolucionais com "runlength" limitado, visando o aumento da capacidade de correção de surtos de erros destes sistemas, é um outro aspecto de interesse que pode ser abordado em trabalhos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

[Abdel, 1995] K. A. S. Abdel-Ghaffar, M. Blaum and J. H. Weber, "Analysis of coding schemes for modulation and error control", IEEE Transactions on Information Theory, vol. 41, No. 6, pp. 1955-1968, November 1995.

[Abdel, 1991] K. A. S. Abdel-Ghaffar and J. H. Weber, "Bounds and constructions for runlength-limited error-control block codes", IEEE Transactions on Information Theory, vol. 37, No. 3, May 1991.

[Baumert, 1979] L. D. Baumert, R. J. McEliece and H. C. A. van Tilborg, "Symbol synchronization in convolutionally coded systems", IEEE Transactions on Information Theory, vol. IT-25, pp. 362-365, May 1979.

[Blahut, 1983] R. E. Blahut, "Theory and Practice of Error Control Codes", Reading, MA: Addison-Wesley, 1983.

[Blaum, 1991] M. Blaum, "Combining ECC with modulation: performance comparisons", IEEE Transactions on Information Theory, vol. 37, No. 3, pp. 945-949, May 1991.

[Bowers, 1960] F. K. Bewers, US Patent 2,957,947, 1960.

[Braun, 1996] V. Braun and A. J. E. M. Janssen, "On the low-frequency suppression performance of DC-free runlength-limited modulation codes", IEEE Transactions on Consumer Electronics, vol. 42, No. 4, pp. 939-945, November 1996.

[Calderbank, 1986] A. R. Calderbank, C. Heegard and T. Lee, "Binary convolutional codes with application to magnetic recording", IEEE Transactions on Information Theory, vol. IT-32, No. 6, pp. 797-815, November 1986.

[Calmet, 1985] J. Calmet, "Algebraic algorithms in GF(q)", Discrete Mathematics, vol. 56, No. 2-3, pp. 101-109, 1985.

[Cariolaro, 1974] G. L. Cariolaro and G. P. Tronca, "Spectra of block coded digital signals", IEEE Transactions on Communications, vol. COM-22, No. 10, pp. 1555-1563, October 1974.

[Cariolaro, 1983] G. L. Cariolaro, G.L. Pierobon and G. P. Tronca, "Analysis of codes and spectra calculations", International Journal of Electronics, vol. 55, No. 1, 1983.

[Carter, 1965] R. O. Carter, "Low-disparity binary coding system", Electronics Letters, vol. 1, pp. 65-68, May 1965.

[Cattermole, 1983] K. W. Cattermole, "Principles of digital line coding", International Journal of Electronics, vol. 55, No. 1, pp. 3-33, July 1983.

[Cideciyan, 1997] R. D. Cideciyan, "Concatenated Reed-Solomon/convolutional coding for data transmission in CDMA-based cellular systems", IEEE Transactions on Communications, vol. 45, No. 10, pp. 1291-1303, October 1997.

[Costello, 1998] D. J. Costello, J. Hagenauer, H. Imai, and S. B. Wicker, "Applications of error-control coding", IEEE Transactions on Information Theory, vol. 44, No. 6, pp. 2531-2595, October 1998.

[Fano, 1963] R. M. Fano, "A heuristic discussion of probabilistic decoding", IEEE Transaction on Information Theory, vol. IT-9, pp. 64-74, April 1963.

[Farkaš, 1996] P. Farkaš and H. Weinrichter, "Transcontrol Codes with Run-Length Limitation", AEÜ Int. J. Electron. Commun., Vol. 50, No. 6, pp. 353-356, 1996.

[Farkaš, 1999] P. Farkaš, W. Pusch, M. Taferner and H. Weinrichter, "Turbo-Codes with Run Length Constraints", AEÜ Int. J. Electron. Commun., vol. 53, No. 3, pp. 161-166, 1999.

[Fernández, 1997-1 E. M. G. Fernández, "Códigos corretores de erro com boas propriedades de codificação de linha", Dissertação de Mestrado, Unicamp, Brasil, 1997.

[Fernández, 1997-2] E. M. G. Fernández and R. Baldini F., "A method to find runlength limited block error control codes", Proceedings of the International Symposium on Information Theory, p. 220, Ulm, Germany, June 29 – July 4, 1997.

[Fernández, 1998] E.M.G. Fernández and R Baldini F., "Runlength limited error control codes from binary expansions of Reed-Solomon codes". Proceedings of the 1998 IEEE-SBT International Telecommunication Symposium, p. 138, São Paulo, Brazil, 1998.

[Fernández, 1999] E.M.G. Fernández e R. Baldini F., "Códigos concatenados entrelaçados para canais com runlength limitado". Anais do XVII Simpósio Brasileiro de Telecomunicações, vol. 2, pp. 539-541, Vila Velha, Brasil, 1999.

[Fernández, 2000] E.M.G. Fernández and R. Baldini F., "Contatenated runlength limited error control codes with soft-decision decoding". Proceedings of the 2000 IEEE International Symposium on Information Theory, p. 259, Sorrento, Italy, June 2000.

[Fleming, 1999] C. Fleming, “A tutorial on convolutional coding with Viterbi decoding”, Technical Report of Spectrum Applications at http://home.netcom.com/~chip.f /Viterbi.html, 1999.

[Forney, 1974] G. D. Forney, “Convolutional codes II: maximum likelihood decoding”, Information and Control, vol. 25, pp. 222-266, July 1974.

[Forney, 1988] G. D. Forney, "Coset codes – part II: binary lattices and related codes", IEEE Transactions on Information Theory, vol. 34, No. 5, pp. 1152-1187, September 1988.

[Forney, 1998] G. D. Forney and G. Ungerboeck, "Modulation and coding for linear gaussian channels", IEEE Transactions on Information Theory, vol. 44, No. 6, pp. 2384-2415, October 1998.

[Gallopoulos, 1989] A. Gallopoulos, C. Heegard and P. H. Siegel, “The power spectrum of run-length-limited codes”, IEEE Transactions on Communications, vol. COM-37, September 1989.

[Griffiths, 1969] J. M. Griffiths, “Binary code suitable for line transmission”, Electronics Letters, vol. 5, pp. 79-81, 1969.

[Helberg, 1992] A. S. J. Helberg and H. C. Ferreira, “Some new runlength constrained binary modulation codes with error-correcting capabilities”, Electronics Letters, vol. 28, No. 2, pp. 137-139, January 1992.

[Ho 1996] K. Ho and C. Leung, “On the undetected error probability of binary expansions of Reed-Solomon Codes”, IEEE Transactions on Information Theory, vol. 42, No. 4, pp. 1271 – 1274, July 1996.

[Hoeve, 1982] H. Hoeve, J. Timmermans and L. B. Vries, "Error correction and concealment in the Compact Disc system", Philips Technical Review, vol. 40, No. 6, pp. 166-172, 1982.

[Hole, 1995] K. J. Hole, "Cosets of convolutional codes with short maximum zero-run lengths", IEEE Transactions on Information Theory, vol. 41, No. 4, pp. 1145-1150, July 1995.

[Hole, 1999] K. J. Hole and ∅. Ytrehus, "Cosets of convolutional codes with least possible maximum zero and one-run lengths", IEEE Transactions on Information Theory, vol. 44, No. 1, pp. 423-431, January 1999.

[Honary, 1993] B. Honary and G. Markarian, "Low complexity trellis decoding of Hamming codes", Electronics Letters, vol. 29, No. 12, pp. 1114-1116, June 1993.

[Honary, 1997] B. Honary and G. Markarian, "Trellis Decoding of Block Codes: A Practical Approach", Kluwer Academic Publisher, 1997.

[Immink, 1990] K. A. S. Immink, "Runlength-limited sequences", Proceedings of the IEEE, vol. 78, No. 11, pp. 1744-1759, November 1990.

[Immink, 1994] K. A. S. Immink, "Reed-Solomon codes and the Compact Disc" in Reed-Solomon Codes and their Applications, pp. 41-59, IEEE Press, 1994.

[Immink, 1998] K. A. S. Immink, P. H. Siegel and J. K. Wolf, "Codes for digital recorders", IEEE Transactions on Information Theory, vol. 44, No. 6, pp. 2260-2299, October 1998.

[Immink, 1999] K. A. S. Immink, "Codes for Mass Data Storage Systems" Shannon Foundation Publishers, The Netherlands, 1999.

[Jelenik, 1969] "A fast sequential decoding algorithm using a stack", IBM J. Research and Development, vol. 13, pp. 675-685, November 1969.

[Kasami, 1997] T. Kasami, "On bit-error probability of a concatenated coding scheme", IEEE Transactions on Communications, vol. 45, No. 5, pp. 536-543, May 1997.

[Kernighan, 1978] B. W. Kernighan and D. M. Ritchie, "The C Programming Language", Bell Laboratories, 1978.

[Lee, 1989] P. Lee and J. K. Wolf, "A general error-correcting code construction for run-length limited binary channels", IEEE Transactions on Information Theory, vol. 35, No. 6, pp. 1330-1335, November 1989.

[Lin, 1983] S. Lin and D. J. Costello Jr., "Error Control Coding: Fundamentals and Applications", Prentice Hall, Inc. Englewood Cliffs, New Jersey, 1983.

[Lin, 1998] S. Lin, T. Kasami, T. Fujiwara and M. Fossorier, "Trellises and Trellis-based Decoding Algorithms for Linear Block Codes", Kluwer Academic Publisher, 1998.

[Markarian, 1994] G. Markarian and B. Honary, "Trellis decoding technique for block RLL/ECC", IEE Proceeding-Communications, vol. 141, No. 5, 297-302, October 1994.

[MacWilliams, 1970] F. J. MacWilliams, "On binary cyclic codes which are also cyclic codes over GF$\left(2^s\right)$", SIAM Journal on Applied Mathematics, vol. 19, No. 1, pp. 75-95, July 1970.

[MacWilliams, 1977] F. J. MacWilliams and N. J. A. Sloane, "The Theory of Error-Correcting Codes", North Holland, 1977.

[Massey, 1963] J. L. Massey, "Threshold Decoding", MIT Press, 1963.

[Oppenheim, 1989] A. V. Oppenheim and R. W. Schafer, "Discrete-Time Signal Processing", Englewood Cliffs, N. J.: Prentice-Hall, 1989.

[O'Reilly, 1990] J. J. O'Reilly and A. Popplewell, "A further note on DC-free coset codes", IEEE Transactions on Information Theory, vol. 36, No. 3, pp. 675-676, May 1990.

[Patapoutian, 1992] A. Patapoutian and P. V. Kumar, "The (d, k) subcode of a linear block code", IEEE Transactions on Information Theory, vol. 38, No. 4, pp. 1375-1382, July 1992.

[Peterson, 1972] W. W. Peterson and E. J. Weldon Jr., "Error Correcting Codes", Second Edition, MIT Press, 1972.

[Pierobon, 1884] G. L. Pierobon, "Codes for Zero Spectral Density at Zero Frequency", IEEE Transaction on Information Theory, vol. IT-30, No. 2, pp. 435-439, March, 1984.

[Popplewell, 1990] A. Popplewell and J. J. O'Reilly, "Spectral characterization and performance evaluation for a new class of error control line codes", IEE Proceedings-Communications, vol. 137, No. 4, pp. 242-246, August 1990.

[Popplewell, 1992] A. Popplewell and J. J. O'Reilly, "Runlength limited binary error control codes", IEE Proceedings-1, vol. 139, No. 3, pp. 349-355, June 1992.

[Popplewell, 1993] A. Popplewell and J. J. O'Reilly, "New class of runlength-limited error-control codes with minimum distance 4", IEE Proceedings-1, vol. 140, No. 2, pp. 104-108, April 1993.

[Popplewell, 1994] A. Popplewell and J. J. O'Reilly, "Algorithms suitable for low complexity implementation of a class of runlength-limited error control codes", IEE Proceedings-Communications, vol. 141, No. 3, pp. 111-117, June 1994.

[Popplewell, 1995] A. Popplewell and J. J. O'Reilly, "A simple strategy for constructing a class of DC-free error-correcting codes with minimum distance 4", IEEE Transactions on Information Theory, vol. 41, No. 4, pp. 1134-1137, July 1995.

[Proakis, 1995] J. G. Proakis, "Digital Communications", Third Edition, Mc Graw-Hill, 1995.

[Reed, 1954] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme", IRE Transactions, vol. IT-4, pp. 38-49, September 1954.

[Reed, 1999] I. Reed and X. Chen, "Error-Control Coding for Data Networks", Kluwer Academic Publishers, 1999.

[Retter, 1997] C. E. Retter, "The average binary expansion of a Reed-Solomon code is good for error detection", Proceedings of the IEEE International Symposium on Information Theory, Ulm, Germany, June 29 - July 4, 1997.

[Roberts, 1996] J. D. Roberts, A. R., D. M. Jones and D. Burke, "Analysis of error-correction constraints in an optical disk", Applied Optics, vol. 35, No. 20, pp. 3915-3924, July 1996.

[Schildt, 1997] H. Schildt, "Borland C++: The Complete Reference", Osborne McGraw-Hill, 1997.

[Schlegel, 1997] C. Schelegel, "Trellis Coding", IEEE Press, 1997.

[Šechny, 1999] M. Šechny and P. Farkaš, "Some new runlength-limited convolutional codes", IEEE Transactions on Communications, vol. 47, No. 7, pp. 962-966, July 1999.

[Siegel, 1985] P. H. Siegel, "Recording codes for digital magnetic storage", IEEE Transactions on Magnetics, vol. MAG-21, No. 5, September 1985.

[Verdú, 1998] S. Verdú, "Fifty years of Shannon theory", IEEE Transactions on Information Theory, vol. 44, No. 6, pp. 2057-2078, October 1998.

[Viterbi, 1967] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", IEEE Transactions on Information Theory, vol. IT-13, pp. 260-269, April 1967.

[Wicker, 1992] S. B. Wicker, "Reed-Solomon error control coding for Rayleigh fading channels with feedback", IEEE Transactions on Vehicular Technology, vol. 41, No. 2, pp. 124-133, May 1992.

[Wolf, 1986] J. K. Wolf and G. Ungerboeck, "Trellis coding for partial-response channels", IEEE Trans. Commun. Theory, vol. COM-34, pp. 765-773, August 1986.

[Wood, 1986] R. W. Wood, "Magnetic recording systems", Proceedings of the IEEE, vol. 74, pp. 1557-1569, November 1986.

[Wozencraft, 1961] M. Wozencraft and B. Reiffen, "Sequential Decoding", MIT Press, 1961.

[Ytrehus, 1991] ∅. Ytrehus, "Runlength-limited codes for mixed-error channels", IEEE Transactions on Information Theory, vol. 37, No. 6, pp. 1577-1585, November 1991.

[Zehavi, 1988] E. Zehavi and J. K. Wolf, "On runlength codes", IEEE Transactions on Information Theory, vol. 34, No. 1, pp. 45-54, January 1988.