UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES - DECOM

CODIFICAÇÃO DE CANAL ALTERNATIVA PARA O SISTEMA MÓVEL TDMA

Por
CARLOS HENRIQUE RODRIGUES DE OLIVEIRA

Tese submetida à Faculdade de Engenharia Elétrica da UNICAMP como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.

Banca Examinadora :
Prof. Dr. Renato Baldini Filho (Orientador) - FEEC/UNICAMP
Prof. Dr. Carlos Eduardo Câmara - CPqD/USF
Prof. Dr. Jaime Portugheis - FEEC/UNICAMP
Prof. Dr. Lee Luan Ling - FEEC/UNICAMP

Campinas, Outubro de 2000.

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

OL4c

Oliveira, Carlos Henrique Rodrigues de
Codificação de canal alternativa para o sistema móvel TDMA / Carlos Henrique Rodrigues de Oliveira.-- Campinas, SP: [s.n.], 2000.

Orientador: Renato Baldini Filho
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Sistemas de comunicação móvel. 2. Telefonia celular. 3. Códigos de controle de erros (Teoria da informação). I. Baldini Filho, Renato. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Este trabalho apresenta propostas alternativas de codificação de canal para o sistema móvel TDMA/IS-136. As comparações destes sistemas de codificação de bloco com o contido na recomendação IS-136 são feitas através de curvas de desempenho (BER x SNR) obtidas por simulação computacional. Os sistemas são analisados em canais AWGN com e sem desvanecimento Rayleigh. A decodificação por treliça de códigos de bloco é realizada com decisão abrupta e suave.

Abstract

This work presents alternative channel encoding schemes to the TDMA/IS-136 mobile system. The comparisons of these block code systems with that contained in the IS-136 Recommendation have been made by computer simulation in terms of the Bit Error Rate (BER) as a function of the Signal-to-Noise Ratio (SNR). The system performance is evaluated in AWGN channel with and without Rayleigh fading. The trellis decoding of block codes is made with hard and soft-decision.

Agradecimentos

- A Deus, por mais esta caminhada de sucesso em minha vida. Sua presença me confortou e me deu força nas horas difíceis deste trabalho.
- A minha mãe, pela força e pelo carinho que me deu para superar todos os obstáculos e poder realizar mais esta etapa em minha vida.
- A minha namorada Sônia, pelo seu amor, carinho e compreensão nos momentos de ausência em que tive que me dedicar mais ao trabalho.
- Ao Prof. Dr. Renato Baldini Filho, pela dedicação e paciência desprendidas ao meu trabalho nestes últimos dois anos.
- Aos Profs. Drs. Carlos Eduardo Câmara (CPqD / USF), Jaime Portugheis (FEEC / UNICAMP) e Lee Luan Ling (FEEC / UNICAMP) membros da Banca Examinadora.
- Aos amigos de Pós-Graduação, Luiz Carlos, Magno, Evélio e Rivael que contribuíram muito para a realização deste trabalho.
- Aos colegas e funcionários da FEEC, pela amizade e companheirismo, que tornaram agradável minha estadia em Campinas.
- Ao CNPq, pelo apoio financeiro através da bolsa de estudo.

Conteúdo

Lista de Figuras

Glossário

AWGN -*Additive White Gaussian Noise* (Ruído Gaussiano Branco Aditivo)

BCH -Bose, Chaudhri e Hocquenghem

CDMA -*Code Division Multiple Access* (Acesso Múltiplo por Divisão de Código)

CELP -*Code Excited Linear Predictive Coding* (Codificação Preditiva Linear com Código de Excitação)

CRC -*Cyclic Redundancy Code* (Código de Redundância Cíclica)

D-AMPS - *Digital Advanced Mobile Phone System* (Sistema Avançado de Telefone Móvel Digital)

DCCH -*Digital Control Channel* (Canal de Controle Digital)

DTC -*Digital Traffic Channel* (Canal de Tráfego Digital)

FACCH -*Fast Associated Control Channel* (Canal de Controle Associado Rápido)

RAC -*Row and Column* (Linha e Coluna)

SACCH -*Slow Associated Control Channel* (Canal de Controle Associado Lento)

SPC -*Single Parity Check* (Verificação de Paridade Simples)

TDMA -*Time Division Multiple Access* (Acesso Múltiplo por Divisão de Tempo)

VSELP -*Vector-Sum Excited Linear Predictive Coding* (Codificação Preditiva Linear de Soma de Vetores de Excitação

Capítulo 1

Introdução

As comunicações sem fio (*Wireless Communications)* vêm atravessando em menos de duas décadas uma acelerada evolução de três gerações [Victor95], motivada em parte por uma vertiginosa demanda de mobilidade e portabilidade nas comunicações, a qual não foi prevista em seu início. Por outro lado, tem-se a revolução digital pela qual os sistemas de telecomunicações vêm atravessando, motivo este que agora se esteja investigando e desenvolvendo a terceira geração destes sistemas. Os sistemas de comunicações sem fio usam os sinais de Rádio Freqüência (RF) e se propagam dentro de canais definidos internacionalmente para o espectro de RF.

A primeira geração dos sistemas de comunicação sem fio foi concebida na década de 70 e baseada em tecnologias analógicas. Sistemas típicos desta geração são os sistemas de telefonia celular analógicos, os sistemas de telefonia *cordless* analógicos bastante usados para comunicações *indoor*, os sistemas *paging* analógicos ou sistemas de busca de pessoas (voz e alfanuméricos), os sistemas de comunicação por satélite geoestacionários com transmissão analógica, entre outros.

A segunda geração dos sistemas de comunicações sem fio se inicia com emergentes tecnologias digitais de acesso múltiplo como o TDMA (*Time Division Multiple Access)* Acesso Múltiplo por Divisão de Tempo e o CDMA (*Code Division Multiple Access)* Acesso Múltiplo por Divisão de Código. Nesta geração são resolvidos e melhorados alguns aspectos não previstos nos sistemas anteriores, como a capacidade de usuários, qualidade e custo dos

serviços.

Desde o surgimento, no início da década de 90, do primeiro sistema móvel comercial baseado na tecnologia TDMA no mercado, a demanda por esquemas e serviços que se utilizam desta técnica de acesso múltiplo tem crescido cerca de 60% ao ano. Predições indicaram mais de 300 milhões de celulares e serviços de comunicação pessoal no mundo todo já neste ano 2000, sendo que grande parte com capacidade de transmissão digital [Harte98].

A norma IS-54 conhecida também conhecida como D-AMPS (*Digital Advanced Mobile Phone System)* registrada na EIA/TIA (*Electronic Industry Association / Telecommunications Industry Association)* foi o primeiro padrão de telefonia celular digital americano, operando no mesmo espectro usado pelos antigos sistemas AMPS (*Advanced Mobile Phone System)* com os mesmos 30 kHz de banda do canal AMPS. Para facilitar a evolução dos sistemas digitais, um fator importante na mudança de sistemas AMPS para sistemas TDMA é o aproveitamento da estrutura já montada e amplamente utilizada, por isso a IS-54 definiu o conceito *dual mode*, que significa operar em ambos os sistemas analógico e digital.

A IS-136 é uma norma de telefonia celular digital baseada em TDMA e na realidade, é uma continuação da norma IS-54 na sua versão IS-54B. A IS-136 incorpora um canal de controle digital, o que permite à operadora da rede uma grande eficiência na operação do tráfego de serviço e a introdução de melhores características, como maior capacidade para transmissão de voz e dados, *roaming* automático internacional, privacidade, maior qualidade de transmissão de voz, além de aumentar significativamente a quantidade de usuários no sistema. Atualmente mais de 36 países usam o sistema celular padrão IS-136 [Harte98].

Na terceira geração dos sistemas sem fio, as comunicações serão pessoais, móveis e universais. Importantes investigações e desenvolvimentos no mundo estão destinados a integrar todos os serviços da segunda geração e cobrir uma ampla gama de serviços *broadband (*voz, dados, vídeo, multimídia) de forma coerente e compatível com a atual tecnologia, complementando as redes de telecomunicações fixas. Nos Estados Unidos os serviços de terceira geração que estão emergindo são chamados PCS (*Personal Communications Systems),* si-

milarmente na Europa estão desenvolvendo o UMTS (*Universal Mobile Telecommunications Systems)* e de outro lado a ITU (*International Telecommunication Union)* vem propondo o IMT-2000 (*International Mobile Communications at year 2000).* Para todos estes sistemas estão sendo previstas implementações comercias no início do século XXI.

Esquemas digitais de comunicações móveis baseados na tecnologia TDMA, utilizam um método de divisão temporal de um canal de voz analógico, com largura de faixa de 30 kHz, por vários usuários (geralmente de 3 a 6 ligações simultâneas). Apesar do aumento da eficiência de utilização de canal do sistema TDMA sobre o sistema AMPS, na concepção deste esquema digital os sinais de controle e redundâncias inseridas na codificação de controle de erro podem ser otimizados. Sendo assim, pode-se oferecer uma melhoria na qualidade do sinal de voz recebida pelo usuário através de uma proteção extra aos bits de maior relevância na saída do codificador de voz do sistema móvel TDMA. Esta possibilidade de melhoria da qualidade do sinal de voz motivou esta pesquisa.

No **Capítulo 2**, foram apresentados conceitos gerais e definições de codificação em bloco e dos arranjos de códigos.

No **Capítulo 3**, foi introduzido o sistema TDMA, suas características gerais, enfocando o canal de tráfego digital [IS-136.2] e a codificação de canal.

No **Capítulo 4**, foram apresentadas as propostas alternativas descrevendo os novos esquemas de codificação de canal e os resultados das simulações para o canal $AWGN$ (*Additive White Gaussian Noise*) Ruído Gaussiano Branco Aditivo com e sem desvanecimento Rayleigh.

O **Capítulo 5** trata das conclusões, comentários sobre os resultados obtidos e sugestões para trabalhos futuros.

Ao fim desta tese temos um apêndice e a bibliografia utilizada.

Capítulo 2

Conceitos e Definições

2.1 Codificação de Bloco

Um código de bloco consiste de um conjunto de vetores de comprimento fixo, onde cada vetor no código é chamado de palavra código e o seu comprimento, determinado pelo número de elementos do vetor, é denotado por n. Os elementos de uma palavra código são selecionados de um alfabeto de q-elementos. Quando o alfabeto do código consiste de 2 elementos, 0 e 1, o código é dito binário. Quando q é uma potência de 2, ou seja, $q = 2^b$ onde b é um inteiro positivo, cada elemento q-ário tem uma representação equivalente de b bits, e assim, um código não binário de comprimento N, pode ser mapeado num código binário de comprimento de bloco $n = bN$.

Considere um código binário com $M = 2^k$ palavras código $(k < n)$. Pode-se associar um mapeamento um-para-um do conjunto de seqüências binárias de comprimento k no conjunto de palavras código. Os k bits são denominados bits de informação e o código resultante é referenciado como um código (n, k), com taxa de k bits de informação por n bits codificados, isto é, $R_c = \frac{k}{n}$.

Um importante parâmetro de um código de bloco (n, k) é a sua distância de Hamming mínima, $d_{\min}$. Sejam duas palavras código C_i e C_j, a distância de Hamming d_{ij} entre elas é o número de elementos ou posições em que elas diferem. O menor valor de $d_{ij}, i \neq j$ é a distância mínima do código.

Além de caracterizar os códigos como sendo binários e não binários, pode-se também

classificar os códigos como lineares e não lineares. A maioria dos códigos de blocos pertence à classe dos códigos lineares, ou seja, códigos que são subespaços vetoriais do espaço consistindo de todas a n-uplas q-árias [Mac Williams98].

No processo de codificação de um código de bloco linear (n, k) é importante adotar a convenção de que as palavras código são representadas como vetores linha. Desta forma, considerando $x_{m1}, x_{m2}, \ldots, x_{mk}$ como os k bits de informação, cada palavra código na entrada do codificador pode ser representada da seguinte forma

$$X_m = [x_{m1} \quad x_{m2} \quad \ldots \quad x_{mk}] \tag{2.1}$$

e a representação desta palavra código na saída do codificador é expressa como

$$C_m = [c_{m1} \quad c_{m2} \quad \ldots \quad c_{mn}] \tag{2.2}$$

Esta operação de codificação realizada, por exemplo, em um codificador de bloco binário pode ser representada pelo conjunto de equações da forma

$$c_{mj} = x_{m1}g_{1j} + x_{m2}g_{2j} + \ldots + x_{mk}g_{kj} \qquad j = 1, 2, \ldots, n. \tag{2.3}$$

onde $g_{ij} = 0$ ou 1. Estas equações lineares acima podem ser representadas também em uma forma matricial como

$$C_m = X_m G \tag{2.4}$$

onde G é chamada de matriz geradora do código. A matriz geradora pode ser escrita na forma sistemática

$$G = [I_k \mid P] = \left[\begin{array}{ccccc|cccc} 1 & 0 & 0 & \cdots & 0 & p_{11} & p_{12} & \cdots & p_{1n-k} \\ 0 & 1 & 0 & \cdots & 0 & p_{21} & p_{22} & \cdots & p_{2n-k} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{k1} & p_{k2} & \cdots & p_{kn-k} \end{array}\right] \tag{2.5}$$

onde I_k é a matriz identidade de dimensão k que simplesmente reproduz o vetor mensagem

X_m e P é uma matriz $k \times (n-k)$ que gera os $n-k$ bits de redundância ou bits de paridade do código.

Um dos mais importantes subconjuntos dos códigos lineares são os códigos cíclicos, estes códigos são caracterizados por apresentar a seguinte propriedade :

Se $C = [c_{n-1}c_{n-2}\ldots c_1c_0]$ é uma palavra código de um código cíclico, então

$$[c_{n-2}c_{n-3}\ldots c_0c_1] \tag{2.6}$$

é obtida do deslocamento cíclico dos elementos de C, e é também uma palavra código. Assim, todos os deslocamentos cíclicos de C são palavras código.

Uma das conseqüências desta propriedade cíclica é a possibilidade de implementar códigos de bloco longos com um grande número de palavras código num sistema de comunicação.

Tratando-se de códigos cíclicos, é conveniente associar a cada palavra código $C = [c_{n-1}c_{n-2}\ldots c_1c_0]$ um polinômio $C(p)$ de grau $\leq n-1$, definido como

$$C(p) = c_{n-1}p^{n-1} + c_{n-2}p^{n-2} + \ldots + c_1p + c_o \tag{2.7}$$

onde para um código binário, os coeficientes do polinômio podem ser 0 ou 1. Então, se $C(p)$ representa uma palavra código de um código cíclico, $p^iC(p) \mod(p^n+1)$ é também uma palavra código do código cíclico. Assim podemos escrever

$$p^iC(p) = Q(p)(p^n+1) + C_i(p) \tag{2.8}$$

onde o polinômio resto $C_i(p)$ representa uma palavra código do código cíclico e $Q(p)$ é o quociente. Pode-se gerar uma palavra código usando um polinômio gerador denotado por $g(p)$ de grau $n-k$ tendo a seguinte forma :

$$g(p) = p^{n-k} + g_{n-k-1}p^{n-k-1} + \ldots + g_1p + x_0 \tag{2.9}$$

Definindo os k bits de informação $[x_{k-1}x_{k-2}\ldots x_1x_0]$ como um polinômio mensagem $X(p) = x_{k-1}p^{k-1} + x_{k-2}p^{k-2} + \ldots + x_1p + x_0$, a multiplicação destes dois polinômios produz

um outro polinômio de grau menor ou igual a $n-1$, que representa uma palavra código.

Uma grande classe dos códigos cíclicos que incluem ambos alfabetos binários e não binários são os chamados códigos BCH (Bose,Chaudhuri e Hocquenghem). Estes códigos podem ser construídos com os parâmetros :

$$\text{Comprimento do bloco :} \quad n = 2^m - 1$$

$$\text{Número de dígitos de verificação de paridade :} \quad n - k \leq mt$$

$$\text{Distância mímina :} \quad d_{\min} \geq 2t + 1$$

para quaisquer inteiros positivos $m\,(m \geq 3)$ e $t(t < 2^{m-1})$. A classe binária destes códigos disponibiliza uma grande quantidade de comprimentos e taxas.

2.2 Arranjos de Códigos

2.2.1 Técnica de Codificação em Arranjo (*Array*)

Arranjos de códigos foram introduzidos por [Elias54] e têm sido propostos para muitas aplicações em controle de erros aleatórios em canais sem memória e em controle de erros em surtos em canais com memória [Farrell86]. Arranjos de códigos são mais freqüentes na forma binária mas também, eles podem ter símbolos no campo finito de Galois de q elementos, $GF(q)$.

A essência de um arranjo de códigos é que a combinação dos bits é baseada em uma contrução geométrica. Os códigos componentes são agrupados em duas ou mais dimensões (ou direções), e tem uma estrutura simples e baixa complexidade de implementação [Markarian97]. Os arranjos de códigos, conhecidos também como códigos produto ou códigos RAC (*Row and Column*) Linha e Coluna [Farrell92], são códigos de verificação de paridade nas linhas e nas colunas, têm distância de Hamming $d_{\min} = 4$ e parâmetros $(n_1 n_2, k_1 k_2, d_{\min})$, onde $(n_1, k_1, 2)$ e $(n_2, k_2, 2)$ são códigos componentes de linha e coluna, respectivamente. O esquema de codificação do código RAC consiste em localizar os bits de informação em k_1 linhas e k_2 colunas.

Então a operação de SPC *(Single Parity Check)* Verificação de Paridade Simples é feita nas linhas e nas colunas para obter um código com $n_1 = k_1 + 1$ linhas e $n_2 = k_2 + 1$ colunas. A estrutura de um código RAC é dada como segue

$$C_{RAC} = \begin{matrix} x_{11} & x_{12} & \dots & x_{1j} & \dots & x_{1k_2} & p_{1n_2} \\ x_{21} & x_{22} & \dots & x_{2j} & \dots & x_{2k_2} & p_{2n_2} \\ . & . & . & . & . & . & . \\ x_{k_1 1} & x_{k_1 2} & \dots & x_{k_1 j} & \dots & x_{k_1 k2} & p_{k_1 n_2} \\ p_{n_1 1} & p_{n_1 2} & \dots & p_{n_1 j} & \dots & p_{n_1 k_2} & p_{n_1 n_2} \end{matrix} \tag{2.10}$$

onde $x_{ij}, (i = 1, 2, ..., k_1;\ j = 1, 2, ..., k_2)$ representam os bits de informação respeitando suas posições na seqüência e $p_{ij}, (i = 1, 2, ..., n_1;\ j = n_2)$ representam os bits de verificação de paridade de linha e de coluna. O número de linhas e de colunas neste esquema de codificação é arbitrário, assim o código pode ser quadrado ou retangular no formato geométrico. O seguinte código $(3, 2)(3, 2)$ pode ser dado como um exemplo de um código RAC quadrado

$$C = \begin{matrix} x_1 & x_2 & p_1 \\ x_3 & x_4 & p_2 \\ p_3 & p_4 & p_5 \end{matrix} \tag{2.11}$$

enquanto como um exemplo de um código RAC retangular, o código$(4, 3)(6, 5)$ pode ser dado por :

$$C = \begin{matrix} x_1 & x_2 & x_3 & x_4 & x_5 & p_1 \\ x_6 & x_7 & x_8 & x_9 & x_{10} & p_2 \\ x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & p_3 \\ p_5 & p_6 & p_7 & p_8 & p_9 & p_4 \end{matrix} \tag{2.12}$$

A matriz geradora de um código RAC é obtida do produto de Kronecker [Slepian92] das matrizes geradoras dos códigos componentes de verificação de paridade simples. Dado as matrizes A e B, o produto de Kronecker [A,B] é definido como o resultado da multiplicação de cada elemento de A pela matriz B.

Seja G^r a matriz geradora de um código componente SPC de linha (n_1, k_1) com o seguinte formato :

$$G^r = \begin{bmatrix} g_1^r \\ g_2^r \\ . \\ . \\ . \\ g_{k_1}^r \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & ... & 0 & 1 \\ 0 & 1 & 0 & ... & 0 & 1 \\ . & . & . & ... & 0 & 1 \\ . & . & . & ... & 0 & 1 \\ . & . & . & ... & 0 & 1 \\ 0 & 0 & 0 & ... & 1 & 1 \end{bmatrix} \tag{2.13}$$

Então G^c é a matriz geradora de um código componente SPC de coluna (n_2, k_2) com o seguinte formato :

$$G^c = \begin{bmatrix} g_1^c \\ g_2^c \\ . \\ . \\ . \\ g_{k_2}^c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & ... & 0 & 1 \\ 0 & 1 & 0 & ... & 0 & 1 \\ . & . & . & ... & 0 & 1 \\ . & . & . & ... & 0 & 1 \\ . & . & . & ... & 0 & 1 \\ 0 & 0 & 0 & ... & 1 & 1 \end{bmatrix} \tag{2.14}$$

Então a matriz geradora G para o código RAC pode ser construída pelo produto de Kronecker entre G^r e G^c. A matriz G tem $k_1 \times k_2$ linhas e $n_1 \times n_2$ colunas, e sua estrutura é dada como segue :

$$G = \begin{bmatrix} g_1 \\ . \\ . \\ . \\ g_{k_1} \\ g_{k_1+1} \\ . \\ . \\ . \\ g_{k_1 k_2} \end{bmatrix} = \begin{bmatrix} g_{11}^r g_1^c & g_{12}^r g_1^c & g_{13}^r g_1^c & \cdots & g_{1n_1}^r g_1^c \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ g_{11}^r g_{k_1}^c & g_{12}^r g_{k_1}^c & g_{13}^r g_{k_1}^c & \cdots & g_{1n_1}^r g_{k_1}^c \\ g_{21}^r g_1^c & g_{22}^r g_1^c & g_{23}^r g_1^c & \cdots & g_{2n_1}^r g_1^c \\ . & . & . & ... & . \\ . & . & . & ... & . \\ . & . & . & ... & . \\ g_{k_1 1}^r g_{k_1}^c & g_{k_1 2}^r g_{k_1}^c & g_{k_1 3}^r g_{k_1}^c & \cdots & g_{k_1 n_1}^r g_{k_1}^c \end{bmatrix} \tag{2.15}$$

Por exemplo, se escolhesse o código SPC (3,2,2) para ambos os códigos componentes de linha e de coluna, a matriz geradora correspondente será definida como segue :

$$G^r = G^c = \begin{bmatrix} g_1^r \\ g_2^r \end{bmatrix} = \begin{bmatrix} g_1^c \\ g_2^c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{2.16}$$

Então, a matriz geradora para o arranjo de códigos (3,2)(3,2) será dada por :

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{2.17}$$

Os parâmetros n, k e $d_{\min}$ de um código RAC são os produtos dos correspondentes parâmetros dos códigos SPC, ou seja :

$$n = n_1 \times n_2 \tag{2.18}$$

$$k = k_1 \times k_2$$

$$d_{\min} = d_1 \times d_2 = 2 \times 2 = 4$$

Assim, códigos RAC podem corrigir um erro e detectar até três erros. Para certas aplicações o dígito correspondente à verificação de paridade dos bits de verificação de paridade, conforme a figura 2.1, poderá ser suprimido reduzindo a distância mínima de 4 para 3, mas a correção de um erro ainda é possível, pois $\left\lfloor t = \frac{(d_{\min}-1)}{2} \right\rfloor$ [Lin83], onde t é o número de erros possíveis de serem corrigidos.

2.2.2 GACs (*Generalised Array Codes*) Arranjo de Códigos Generalizados

Para um dado tamanho n da palavra codificada e distância mínima, $d_{\min}$, arranjo de códigos podem não alcançar a máxima taxa de codificação, $(\frac{k}{n})$, possível com código linear. Por exemplo, se $n = 16$ e $d_{\min} = 4$, então é possível obter um código de bloco binário com $k = 11$

bits de informação	verificação nas linhas
verificação nas colunas	verificação nas verificações

Figura 2.1: Formação geral do arranjo de códigos.

dígitos de informação [Mac Williams98]. Entretanto a técnica descrita anteriormente para arranjo de códigos não permite a construção de um código com estes parâmetros. Uma nova técnica, que permite a construção de um arranjo de códigos com os mesmos parâmetros n e $d_{\min}$, mas com um incremento do número de bits de informação k, foi proposta [Farrell93]. A técnica chamada GACs (*Generalised Array Codes)* Arranjo de Códigos Generalizados, apresenta uma construção de baixa complexidade de diferentes códigos de bloco (e.g. Hamming, Golay, Reed-Muller (RM), BCH, etc.) [Honary93] [Markarian93] [Darnell93].

Um arranjo de códigos generalizados é um código em que os códigos componentes de linha e de coluna podem ter diferentes números de bits de informação e de verificação de paridade. O comprimento deste código é $n = n_1 \times n_2$, onde n_1 e n_2 são número de colunas e linhas, respectivamente, e o número total de bits de informação é dado por $k = \sum_{i=1}^{n_2} k_l$, onde k_l é o número de bits de informação da l-ésima linha.

A construção de um código GAC (n_0, k_0, d_0) segue os seguintes passos :

1. Projete um código produto binário básico C_1 como mostra na figura 2.2a, com $n = n_1 \times n_2$ $(n = n_0)$ onde n_1 é a quantidade de colunas e n_2 é a quantidade de linhas de C_1 a partir do código componente de linha $R_1 = (n_1, k_1, d_1)$ e do código componente de coluna SPC (n_2, k_2, d_2), onde k_1 é a quantidade de bits de informação por linha de C_1 e k_2 é a quantidade de bits de informação por coluna de C_1 e $d_1 = \lfloor \frac{d_0}{2} \rfloor$, onde o operador $\lfloor x \rfloor$ representa o maior inteiro menor que x. Se n_0 for um número primo faça $n = n_0 + 1$.

2. Projete um código produto binário adicional C_2 como mostra na figura 2.2b, onde P é um arranjo binário $k_1 \times n_2$ onde todos os elementos são bits de paridade. A submatriz

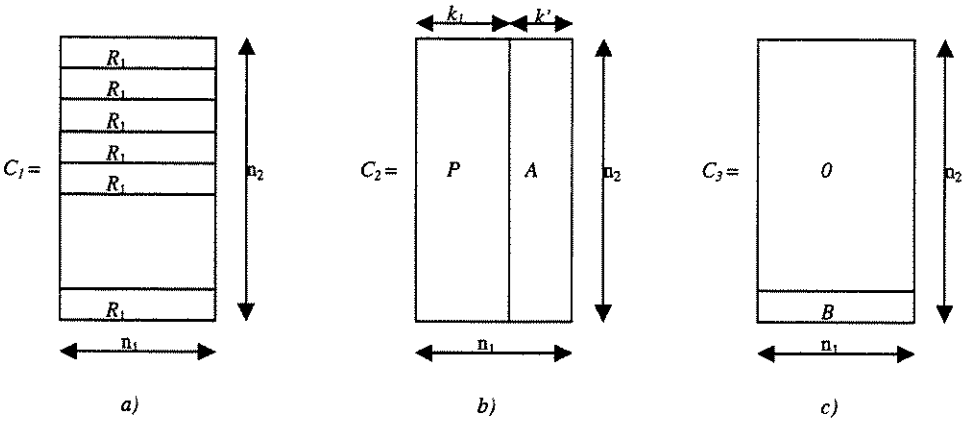

Figura 2.2: Construção do Arranjo de Códigos Generalizados (GAC).

A binária $(n_1 - k_1) \times n_2$ possui em sua primeira linha $k' \leq n_1 - k_1$ dígitos de informação e todos os códigos coluna são de repetição $(n_2, 1, n_2)$.

3. Se um dígito de informação adicional for necessário, projete um terceiro código produto binário adicional C_3 como mostra na figura 2.2c, onde O é a matriz $[(n_2 - 1) \times n_1]$ toda nula e B é um código de linha de repetição $(n_1, 1, n_1)$ com k_0 dígitos de informação.

4. Adicione, soma módulo-2, os três códigos C_1, C_2 e C_3 :

$$C = C_1 \oplus C_2 \oplus C_3 \tag{2.19}$$

5. Se $n = n_0 + 1$, apague o bit localizado na linha n_2 e coluna n_1.

Os códigos C_1, C_2 e C_3 são códigos de produto lineares ou não lineares e a construção do código total C resultará em um código não sistemático com os seguintes parâmetros : (n_0, k_0, d_0). Serão analizados como exemplo alguns casos de GACs de interesse ao trabalho.

Construção dos GACs (8,4,4) e (7,4,3)

O GAC (8,4,4) é equivalente ao código Reed-Muller RM (1,3) com os seguintes parâmetros: $r = 1$ e $m = 3$ onde r é uma ordem do código e $n = 2^m$ é o comprimento do código [Lin83]. Seja $X = (x_1, x_2, x_3, x_4)$ o vetor dos bits de informação de entrada. O procedimento para a construção do código é como segue [Honary93] :

1. Projete o arranjo do código básico (8,3,4) com $(n_1 = 2, n_2 = 4)$, C_1, com o código componente de linha de repetição $R_1 = (n_1, k_1, d_1) = (2, 1, 2)$ e o código componente de coluna SPC, $(n_2, k_2, d_2) = (4, 3, 2)$ como

$$C_1 = \begin{matrix} x_1 & p_1 \\ x_2 & p_2 \\ x_3 & p_3 \\ p_4 & p_4 \end{matrix} \tag{2.20}$$

onde $x_i, i = 1, 2, 3$ representam os bits de informação, $p_j, j = 1, ...4$ representam os bits de verificação de paridade e

$$\begin{aligned} p_i &= x_i, i = 1, 2, 3 \\ p_4 &= x_1 \oplus x_2 \oplus x_3 \end{aligned}$$

2. Projete um código adicional C_2 com a seguinte estrutura

$$C_2 = \begin{matrix} 0 & x_4 \\ 0 & x_4 \\ 0 & x_4 \\ 0 & x_4 \end{matrix} \tag{2.21}$$

onde x_4 é um bit de informação.

3. Desde que todos os bits de informação foram utilizados, não é necessário projetar o terceiro arranjo de códigos adicional, C_3.

4. Some módulo-2, os dois códigos C_1 e C_2 e leia a palavra codificada linha por linha :

$$C = C_1 \oplus C_2 = \begin{matrix} x_1 & (x_4 \oplus p_1) \\ x_2 & (x_4 \oplus p_2) \\ x_3 & (x_4 \oplus p_3) \\ p_4 & (x_4 \oplus p_4) \end{matrix} = [x_1, (x_4 \oplus p_1), x_2, (x_4 \oplus p_2), x_3, (x_4 \oplus p_3), p_4, (x_4 \oplus p_4)] \tag{2.22}$$

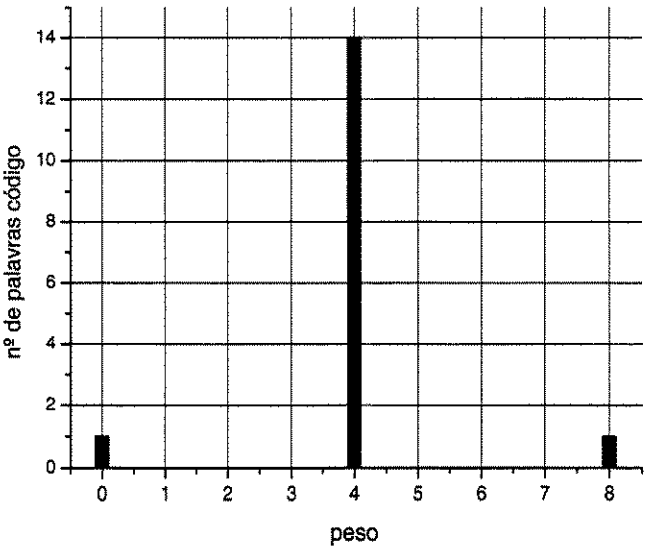

Figura 2.3: Espectro de distribuição de pesos do GAC (8,4,4).

5. Como $n_0 = n$, não é necessário apagar o bit de verificação de paridade que está localizado na quarta linha e segunda coluna da seqüência.

A matriz geradora a partir do código construído C é dada por :

$$GAC_{(8,4,4)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{2.23}$$

Este código é não sistemático com os seguintes parâmetros : $(n_0 = 8, k_0 = 4, d_0 = 4)$. O espectro de distribuição de pesos deste código é mostrado na figura 2.3, que é o mesmo espectro de distribuição de pesos do correspondente código RM (1,3).

O código (7,4,3) pode ser obtido apagando o bit de verificação de paridade localizado na segunda coluna e na quarta linha do código GAC (8,4,4). O novo GAC (7,4,3) apresenta a

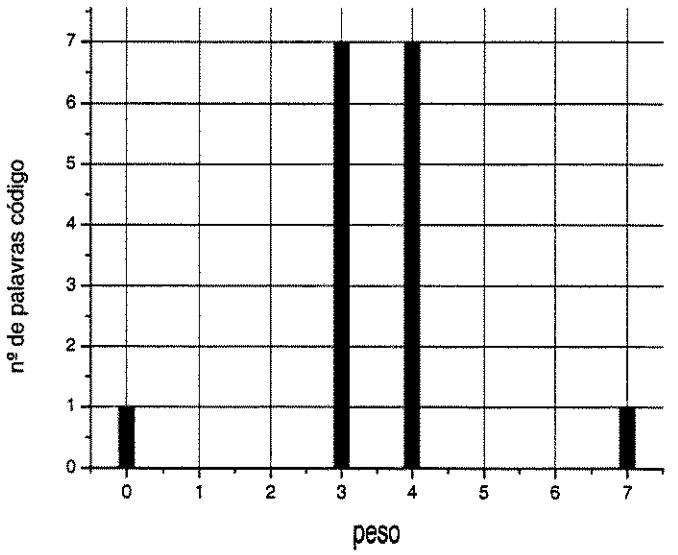

Figura 2.4: Espectro de distribuição de pesos do GAC (7,4,3).

seguinte matriz geradora :

$$GAC_{(7,4,3)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \tag{2.24}$$

O espectro de distribuição de pesos é mostrado na figura 2.4. Note que este espectro de pesos é o mesmo do código de Hamming [Lin83] equivalente.

Construção dos GACs (24,12,8) e (23,12,7)

O GAC (24,12,8) é equivalente ao código de Golay (24,12,8) estendido. O procedimento para construção do código é como segue [Markarian97] :

a) Projete o código produto básico (24,8,8) ($n = 8 \times 3, n_1 = 8, n_2 = 3$), C_1, usando o código componente de linha $R_1 = (n_1, k_1, d_1) = (8, 4, 4)$ descrito anteriormente e o

código componente de coluna SPC $(n_2, k_2, d_2) = (3, 2, 2)$ como

$$C_1 = \begin{matrix} x_1 & x_4 \oplus p_1 & x_2 & x_4 \oplus p_2 & x_3 & x_4 \oplus p_3 & p_4 & x_4 \oplus p_4 \\ x_5 & x_8 \oplus p_5 & x_6 & x_8 \oplus p_6 & x_7 & x_8 \oplus p_7 & p_8 & x_8 \oplus p_8 \\ p_9 & p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} \end{matrix} \tag{2.25}$$

onde $x_i, i = 1, 2, ..., 8$ representam os bits de informação, $p_j, j = 1, 2, ..., 16$ representam os bits de verificação de paridade e :

$$p_i = x_i, i = 1, 2, 3, 5, 6, 7$$

$$p_4 = x_1 \oplus x_2 \oplus x_3$$

$$p_8 = x_5 \oplus x_6 \oplus x_7$$

$$p_9 = x_1 \oplus x_5$$

$$p_{10} = x_4 \oplus p_1 \oplus x_8 \oplus p_5$$

$$p_{11} = x_2 \oplus x_6$$

$$p_{12} = x_4 \oplus p_2 \oplus x_8 \oplus p_6$$

$$p_{13} = x_3 \oplus x_7$$

$$p_{14} = x_4 \oplus p_3 \oplus x_8 \oplus p_7$$

$$p_{15} = p_4 \oplus p_8$$

$$p_{16} = x_4 \oplus p_4 \oplus x_8 \oplus p_8$$

b) Projete os dois códigos produto adicionais, C_2 e C_3, com as seguintes estruturas respectivamente

$$C_2 = \begin{matrix} c_1 & x_9 & c_2 & x_{10} & c_3 & c_4 & x_{11} & c_5 \\ c_1 & x_9 & c_2 & x_{10} & c_3 & c_4 & x_{11} & c_5 \\ c_1 & x_9 & c_2 & x_{10} & c_3 & c_4 & x_{11} & c_5 \end{matrix} \tag{2.26}$$

$$C_3 = \begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_{12} & x_{12} & x_{12} & x_{12} & x_{12} & x_{12} & x_{12} & x_{12} \end{matrix} \tag{2.27}$$

onde : $x_i, i = 9, ..., 12$ representam os bits de informação, $c_j, j = 1, 2, ...5$ representam os bits de verificação de paridade e o subcódigo SPC linha (8,3,4) de C_2 é dado na tabela 2.1.

Vetor de Informação (x_9, x_{10}, x_{11})	Vetor codificado $(c_1, x_9, c_2, x_{10}, c_3, c_4, x_{11}, c_5)$
000	00000000
001	10000111
010	10011100
011	10110010
100	11100100
101	11001010
110	11010001
111	01010110

Tabela 2.1: Tabela de codificação do subcódigo SPC de linha (8,3,4)

c) Faça a soma módulo-2 dos três códigos, C_1, C_2 e C_3 obtendo :

$$C = \begin{matrix} x_1 \oplus c_1 & x_4 \oplus p_1 \oplus x_9 & x_2 \oplus c_2 & x_4 \oplus p_2 \oplus x_{10}... \\ x_5 \oplus c_1 & x_8 \oplus p_5 \oplus x_9 & x_6 \oplus c_2 & x_8 \oplus p_6 \oplus x_{10}... \\ p_9 \oplus c_1 \oplus x_{12} & p_{10} \oplus x_9 \oplus x_{12} & p_{11} \oplus c_2 \oplus x_{12} & p_{12} \oplus x_{10} \oplus x_{12}... \end{matrix}$$

$$\begin{matrix} ...x_3 \oplus c_3 & x_4 \oplus p_3 \oplus c_4 & p_4 \oplus x_{11} & x_4 \oplus p_4 \oplus c_5 \\ ...x_7 \oplus c_3 & x_8 \oplus p_7 \oplus c_4 & p_8 \oplus x_{11} & x_8 \oplus p_8 \oplus c_5 \\ ...p_{13} \oplus c_3 \oplus x_{12} & p_{14} \oplus c_4 \oplus x_{12} & p_{15} \oplus x_{11} \oplus x_{12} & p_{16} \oplus c_5 \oplus x_{12} \end{matrix} \quad (2.28)$$

E leia a palavra codificada linha por linha :

$$C = [(x_1 \oplus c_1), (x_4 \oplus p_1 \oplus x_9), ..., (x_4 \oplus p_4 \oplus c_5), ..., (p_{16} \oplus c_5 \oplus x_{12})] \quad (2.29)$$

O espectro de distribuição de pesos do GAC (24,12,8) construído é mostrado na figura 2.5. Este é o mesmo espectro de distribuição de pesos do código de Golay (24,12,8) estendido gerado a partir de sua matriz geradora dada por Forney [Forney88b] apresentada na figura 2.6. Isto segue de [Pless68] que o GAC (24,12,8) construído é o código de Golay (24,12,8) estendido.

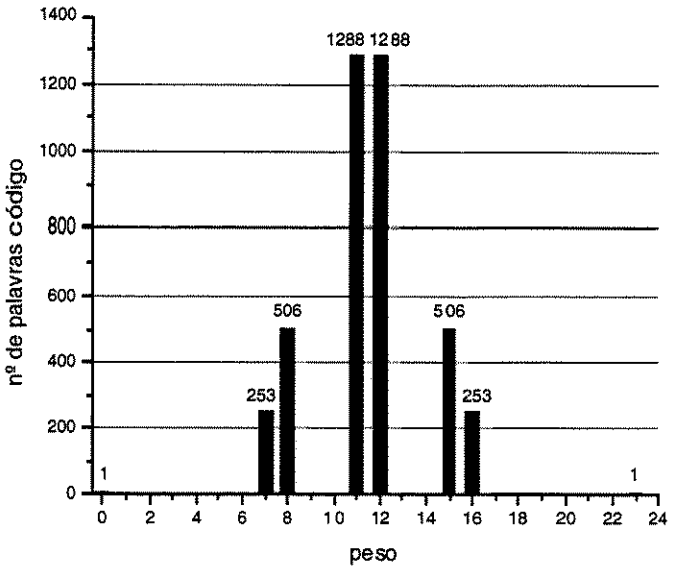

Figura 2.5: Espectro de distribuição de pesos do GAC (24,12,8).

O GAC (23,12,7) pode ser obtido do GAC (24,12,8) simplesmente apagando o bit de verificação de paridade encontrado na terceira linha e na oitava coluna. Assim, a estrutura do código fica :

$$C = \begin{array}{cccc} x_1 \oplus c_1 & x_4 \oplus p_1 \oplus x_9 & x_2 \oplus c_2 & x_4 \oplus p_2 \oplus x_{10}... \\ x_5 \oplus c_1 & x_8 \oplus p_5 \oplus x_9 & x_6 \oplus c_2 & x_8 \oplus p_6 \oplus x_{10}... \\ p_9 \oplus c_1 \oplus x_{12} & p_{10} \oplus x_9 \oplus x_{12} & p_{11} \oplus c_2 \oplus x_{12} & p_{12} \oplus x_{10} \oplus x_{12}... \\ ...x_3 \oplus c_3 & x_4 \oplus p_3 \oplus c_4 & p_4 \oplus x_{11} & x_4 \oplus p_4 \oplus c_5 \\ ...x_7 \oplus c_3 & x_8 \oplus p_7 \oplus c_4 & p_8 \oplus x_{11} & x_8 \oplus p_8 \oplus c_5 \\ ...p_{13} \oplus c_3 \oplus x_{12} & p_{14} \oplus c_4 \oplus x_{12} & p_{15} \oplus x_{11} \oplus x_{12} & \end{array} \quad (2.30)$$

$$G = \begin{bmatrix}
11111111 & 00000000 & 00000000 \\
00000000 & 11111111 & 00000000 \\
00000000 & 00000000 & 11111111 \\
11110000 & 11110000 & 00000000 \\
00000000 & 11110000 & 11110000 \\
11001100 & 11001100 & 00000000 \\
00000000 & 11001100 & 11001100 \\
10101010 & 10101010 & 00000000 \\
00000000 & 10101010 & 10101010 \\
01111000 & 01111000 & 01111000 \\
10011100 & 10011100 & 10011100 \\
01010110 & 01010110 & 01010110
\end{bmatrix}$$

Figura 2.6: Matriz geradora do código de Golay (24,12,8) estendido.

O espectro de distribuição de pesos do GAC (23,12,7) construído é mostrado na figura 2.7. Este é o mesmo espectro de distribuição de peso do código de Golay (23,12,7) [Markarian97].

2.2.3 Procedimento de Construção de Treliças para Arranjo de Códigos

O procedimento de construção de treliças para os arranjos de códigos que utilizam matrizes de verificação de paridade de códigos componentes de linha e coluna, foi introduzido por Wolf [Wolf78]. Embora esta técnica forneça treliças mínimas para todos os códigos produto, o procedimento de construção das treliças é complexo e requer uma grande quantidade de cálculos. Todos os ramos das treliças são rotulados somente pelos símbolos (ou bits) codificados. Assim, para reconstruir os dados de informação original devem ser realizadas operações adicionais.

O uso de matrizes geradoras dos códigos componentes de linha e de coluna simplifica o procedimento de construção da treliça, mantendo a complexidade da mesma inalterada. O diagrama de treliça mínima de um arranjo de códigos com símbolos de $GF(q)$,onde ambos os códigos componentes de linha e coluna são de verificação de paridade simples, consistirá

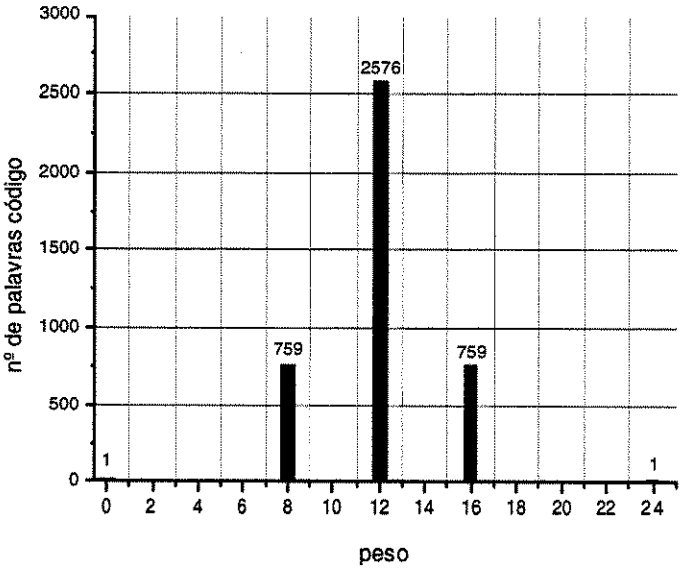

Figura 2.7: Espectro de distribuição de pesos do GAC (23,12,7).

de

$$N_s^{\min} = q^{\min[n_1,n_2]-1} \tag{2.31}$$

estados, onde $\min[n_1, n_2]$ é o menor dos dois valores entre n_1 e n_2, n_1 é o número de colunas e n_2 é o número de linhas do código produto básico C_1.

Como o arranjo de códigos pode ter formato quadrado ou retangular, o procedimento da construção da treliça é como segue [Kaya93] :

1. Selecione o menor valor entre n_1 e n_2 (por exemplo n_1).
2. Determine o número N_s de estados na treliça e o número N_c de vértices (ou segmentos de colunas) como

$$N_s = q^{(n_1-1)} \tag{2.32}$$

$$N_c = n_2 + 1 \tag{2.33}$$

3. Identifique cada estado por um vetor $S^p(A) = S^p(a_1, a_2, ..., a_{k_1})$ de comprimento k_1 bits,

onde $a_j \in GF(q)$, $j = 1, 2, ..., k_1$ e p é a profundidade de estados $(0 \leq p \leq N_c - 1)$.

4. Marque o estado inicial e o estado final da treliça respectivamente como :

$$S^0(0_1, 0_2, \cdots, 0_{k_1}) \ \ e \ \ S^{N_c-1}(0_1, 0_2, \cdots, 0_{k_1})$$

5. Rotule cada ramo da treliça de profundidade p com dois valores particulares : a seqüência de informação de entrada $m_A^p(B) = (a_1, a_2, ..., a_{k_1})$ e o vetor de saída $U_A^p(B) = (a_1, a_2, ..., a_{k_1}, a_{n_1})$ resultante de $m_A^p(B)$, onde $a_{n_1} = a_1 \oplus a_2 \oplus \cdots \oplus a_{k_1}$. Nas transições do estado antecessor $S^p(A)$ com profundidade p para o estado sucessor $S^{p+1}(B)$ com profundidade $p + 1$, o vetor m_A^p com k_1 bits é obtido da seguinte maneira

$$m_A^p = S^p(A) \oplus S^{p+1}(B) \tag{2.34}$$

onde A e B são todos os possíveis vetores com k_1 símbolos que representam a profundidade p atual e a profundidade $p + 1$ seguinte. O procedimento de construção da treliça mostra que cada profundidade da treliça corresponde a uma linha do arranjo de códigos, e é rotulada de acordo com o código linha correspondente.

Como a última linha do código contém somente símbolos de verificação de paridade, os ramos da última profundidade da treliça são rotulados apenas com $U_A^{N_c-1}$.

Existem $2^{(n_1-1)(n_2-1)}$ caminhos distintos através do diagrama de treliça e cada caminho corresponde a uma única palavra código.

Definições para Construção das Treliças

O procedimento de construção da treliça descrito acima fornece uma estrutura de treliça mínima para arranjo de códigos e apresenta uma complexidade de decodificação mais baixa que outros códigos corretores de erros simples em uma faixa similar de taxa de codificação [Kaya93].

Uma treliça é uma forma compacta de representar todas as q^k palavras código de um código linear. Um caminho do estado inicial ao final corresponde a uma palavra código. Assim, existe um total de $N_p = q^k$ caminhos possíveis na treliça.

É importante também ressaltar que podem existir mais de uma treliça para representar um mesmo código de bloco.

Uma treliça T é mínima para um código de bloco C, se para qualquer outra treliça T' de C, $V_i \leq V_i'$ para todo i [Muder88]. V_i e V_i' são os vértices ou colunas $V_0, V_1, \cdots, V_{N_c-1}$ das treliças. Para encontrar a treliça mínima, divide-se as palavras código em duas partes : um passado e um futuro com relação a algum indexador de tempo i [Forney88a] [Muder88]. O passado, $P_{i-}(c)$, de uma palavra código $\mathbf{c} = (c_1, c_2, \cdots, c_n)$ indexado no tempo i é a i-upla $(c_1, c_2, \cdots, c_i)$; o futuro, $P_{i+}(c)$, de $\mathbf{c}$ é a $(n-i)$-upla $(c_{i+1}, c_{i+2}, \cdots, c_n)$. Quando $i = 0$, o passado de $\mathbf{c}$ é nulo, enquanto o futuro de $\mathbf{c}$ é a palavra código $\mathbf{c}$. Similarmente, quando $i = n$, o futuro de $\mathbf{c}$ é nulo, enquanto o passado de $\mathbf{c}$ é a palavra código $\mathbf{c}$. Para cada valor do indexador de tempo i, a palavra código $\mathbf{c}$ é a concatenação do passado com o futuro para aquele valor i do indexador de tempo. Em uma treliça mínima, se duas palavras código distintas passarem através do mesmo estado no índice i, então a continuação do passado de uma com o futuro da outra é também uma palavra válida do código.

Um estado é um ponto de conexão entre passados e futuros de palavras código, e o problema para encontrar uma treliça mínima para o código é equivalente ao problema de minimização do número de estados em um valor do indexador i de tempo [Forney88b] [Muder88]. Para um dado código C de bloco $(n, k, d_{\min})$, o número de estados na treliça mínima $T_{\min}$ pode ser obtida como segue [Forney88b] [Muder88] :

Sejam $C_p = (n_p, k_p, d_{\min})$ e $C_f = (n_f, k_f, d_{\min})$ os códigos componentes passado e futuro para um ponto de projeção $1 \leq i \leq n$, tal que, $n = n_p + n_f$. Então o número mínimo de estados para a i-ésima coluna da treliça pode ser definido como

$$N_i^{\min} \geq \frac{q^k}{q^{k_p} \times q^{k_f}} \qquad i = 1, 2, \cdots, n \tag{2.35}$$

onde k, k_p e k_f são as dimensões de C, C_p e C_f, respectivamente.

Exemplo 1 *Encontrar o número de estados na treliça mínima do GAC* $(n = 16, k = 5, d_{\min} = 8)$.

Seguindo o procedimento apresentado anteriormente, escolhe-se um ponto de quebra no meio do código : $i = 8$. Neste caso ambos C_p e C_f devem ser definidos como códigos com $n = 8$ e $d_{\min} = 8$. Assim estes códigos são dados como

$$C_p = C_f = (8, 1, 8)$$

e representam códigos de repetição com $(k_p = k_f = 1)$. Então, o número de estados no meio da treliça mínima $(i = 8)$ é dado por :

$$N_{i=8}^{\min} = \frac{2^5}{2^1 \times 2^1} = 2^3$$

Entretanto, se fosse escolhido o ponto de quebra $i = 7$, os códigos componentes passado e futuro deveriam ser definidos por

$$C_p = (7, 0, 8) \qquad C_f = (9, 1, 8)$$

com $k_p = 0, k_f = 1$ e

$$N_{i=7}^{\min} = \frac{2^5}{2^0 \times 2^1} = 2^4$$

Portanto, para minimizar o número de estados no diagrama de treliça do GAC (16,5,8), a treliça deve ser construída com vértices que correspondam ao ponto de quebra $i = 8$.

Decodificação por Treliça de Arranjo de Códigos.

Uma vez que o diagrama de treliça do arranjo de códigos foi construído, a decodificação de Viterbi por decisão abrupta ou suave pode ser empregada para obter a Decodificação por Máxima Verossimilhança (*Maximum Likelihood Decoding)* [Markarian97]. Neste caso, o procedimento do algoritmo de Viterbi para códigos convolucionais é ligeiramente modificado para acomodar diferenças entre a estrutura da treliça para arranjo de códigos.

Como cada linha de um arranjo de códigos é representada por uma seção da treliça, a seqüência de símbolos (ou bits) recebida é dividida em n_2 sub-seqüências, cada uma correspondendo a uma seção da treliça, como mostra a figura 2.8.

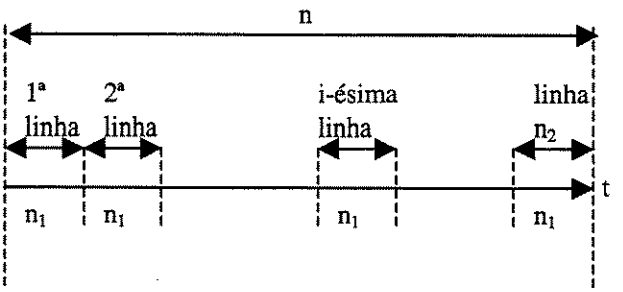

Figura 2.8: Divisão da seqüência recebida.

Em cada estado a distância (Euclidiana para decisão suave ou de Hamming para decisão abrupta) entre a sub-seqüência recebida e todos os possíveis ramos rotulados é calculada. O valor calculado para cada ramo é denominado de métrica do ramo. O cálculo da métrica do ramo começa em $p = 0$. Para a próxima profundidade, o ramo com distância mínima entre todos os possíveis ramos que chegam no estado é selecionado e seu valor, denominado métrica do caminho, é armazenado. O ramo selecionado é um caminho sobrevivente para aquele estado particular, e que pode ser chamado de caminho sobrevivente temporário. Os outros ramos que entram neste estado em questão são eliminados.

Como um exemplo, uma seção de uma treliça é dada na figura 2.9 para demonstrar o cálculo da métrica do ramo, a métrica do caminho e a eliminação do caminho não válido (o ramo não válido é marcado com um sinal x). Para cada coluna e estado da treliça, o processo de cálculo da métrica do ramo, da métrica do caminho e a determinação do caminho sobrevivente temporário é mostrado na figura 2.9.

O procedimento é realizado até a profundidade $p = N_c - 2$. Da profundidade $p = N_c - 2$ todos os ramos terminam no estado final todo zero. Selecionando o caminho com distância mínima entre todos os caminhos sobreviventes temporários, permanecerá somente um caminho até este ponto. Então, todos os outros caminhos sobreviventes temporários serão eliminados e o caminho sobrevivente é determinado. Finalmente, os símbolos da palavra código são obtidos pela leitura dos símbolos do lado direito do rótulo dos ramos do caminho sobrevivente $[U_A^p(B)]$, enquanto que o vetor de informação de entrada pode ser obtido pela

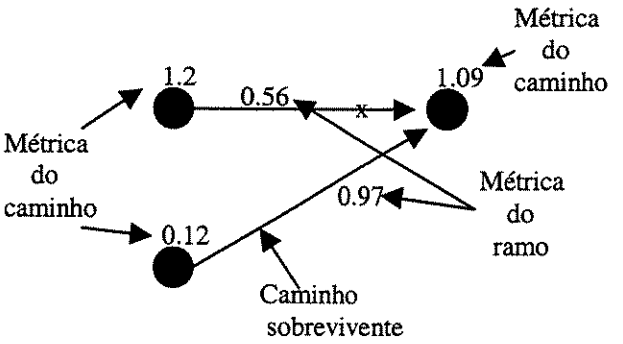

Figura 2.9: Cálculo das métricas e caminho sobrevivente.

leitura dos símbolos do lado esquerdo do rótulo dos ramos do caminho sobrevivente $[m_A^p(B)]$.

Decodificação por Treliça de Arranjo de Códigos Generalizados (*GACs*)

O procedimento de construção de treliça para arranjo de códigos, descrito no ítem 2.2.3, pode ser adotado para a construção de treliça para GACs. Entretanto, a treliça de um arranjo de códigos necessita ser modificada para acomodar novas palavras código adicionais. Similarmente aos arranjos de códigos, o diagrama de treliça de um GAC começa de um estado inicial e termina no estado final. A l-ésima linha do GAC definirá o rotulamento dos ramos na l-ésima seção da treliça, uma vez que cada linha do código é representada por uma seção da treliça e qualquer caminho do estado inicial até o estado final corresponderá a uma palavra código admissível do GAC. O procedimento para a construção da treliça é como segue [Farrell93] [Darnell93] :

1. Calcule o número de colunas, N_c, e o número de estados, N_s, como

$$N_c = n_2 + 1 \tag{2.36}$$

$$N_s = 2^{\max k_l} \tag{2.37}$$

$$l = 1, 2, ..., n_2 \tag{2.38}$$

onde k_l é o número de símbolos de informação na l-ésima linha do GAC.

2. Identifique cada estado de profundidade p pelo vetor de $\max\{k_l\}$ símbolos

$$S_p(A) = S_p(a_1, a_2, \cdots, a_{\max\{k_l\}}) \tag{2.39}$$

onde $a_j = 0, 1$ e $j = 1, 2, ..., \max\{k_l\}$.

3. Os estados inicial e final devem ser rotulados como $S_0(00...0)$ e $S_{n_1}(00...0)$, respectivamente.

4. Os ramos na seção p da treliça devem ser rotulados como X_p/C_p, onde X_p é um vetor de informação de k_l símbolos para o l-ésimo código linha e C_p é a palavra código referente ao l-ésimo código linha

$$C_p = X_p^1 G_p^1 \oplus C_p^2 \tag{2.40}$$

onde G_p^1 é a matriz geradora do l-ésimo código linha do código produto básico, C_1; X_p^1 é um vetor de símbolos de informação do l-ésimo código linha do código produto básico, C_1 e C_p^2 é a palavra código do l-ésimo código linha do primeiro código produto adicional, C_2.

O vetor C_p pode ser obtido de uma forma mais simples considerando-o apenas como a palavra código, gerada a partir de X_p, referente a cada uma das l linhas do GAC .

5. Existem 2^{k_l} ramos partindo de cada estado, $S_p(A)$, na profundidade p ($p < N_c$) e cada ramo é conectado com o estado $S_{p+1}(A)$ na profundidade $p+1$, como é definido a seguir :

$$S_{p+1}(A) = S_p(A) \oplus X_p \tag{2.41}$$

6. Se um segundo código adicional, C_3, for usado para a construção do GAC, na última seção da treliça $p = N_c$, todos os estados anteriores deve ser conectados ao estado final, $S_{n_1}(00...0)$, com dois ramos paralelos (o rótulo do segundo ramo é o complemento do

rótulo do ramo original). Existem

$$N_0 = \prod_{l=1}^{n_2} 2^{k_l} \tag{2.42}$$

caminhos distintos através deste diagrama de treliça e cada caminho corresponde a uma palavra código pertencente ao código.

Com a técnica descrita acima pode-se projetar uma treliça mínima para um dado GAC. Será apresentado um exemplo para os GACs (8,4,4) e (7,4,3).

Exemplo 2 *Construção do diagrama de treliça dos GACs (8,4,4) e (7,4,3).*

Seguindo a técnica introduzida por [Forney88b] [Muder88], pode-se estimar a quantidade de estados na treliça mínima usando a equação 2.35 :

- GAC (8,4,4)

  para $i = 4 \Rightarrow C_p = C_f = (4, 1, 4)$ com $k_p = k_f = 1$ e

$$N_{i=4}^{\min} = \frac{2^4}{2^1 \times 2^1} = 4 \tag{2.43}$$

- GAC (7,4,3)

  para $i = 3 \Rightarrow C_p = (3, 1, 3)$ e $C_f = (4, 1, 3)$ com $k_p = k_f = 1$ e $N_{i=3}^{\min} = \frac{2^4}{2^1 \times 2^1} = 4$

Após verificado a quantidade de estados mínimos da treliça, calcula-se a quantidade de colunas e de estados da treliça como segue :

Da construção do GAC (8,4,4), vem que $C_1(8, 3, 4)$. Isto implica que o código componente de linha $R_1 = (n_1 = 2,\ k_1 = 1,\ d_1 = 2)$ e que o código componente de coluna SPC $= (n_2 = 4,\ k_2 = 3,\ d_2 = 2)$. Então, a quantidade de colunas, $N_c = n_2 + 1 = 4 + 1 = 5$. A quantidade de estados, $N_s = 2^{\max k_l} = 2^2 = 4$, é a mesma calculada em 2.43.

Cada estado das $p(0, 1, ..., n_2 = 4)$ colunas é identificado com um vetor de dois bits, $S_p(a_1, a_2)$; nas colunas $p = 0$ e $p = 4$ a treliça terá somente um estado nomeados $S_0(00)$ e $S_4(00)$, respectivamente. Os ramos da treliça são rotulados como X_p/C_p. Seguindo este

procedimento, o diagrama de treliça para o GAC (8,4,4) é apresentado na figura 2.10 e é similar ao dado por [Forney88b] para o código de Reed-Muller RM (1,3).

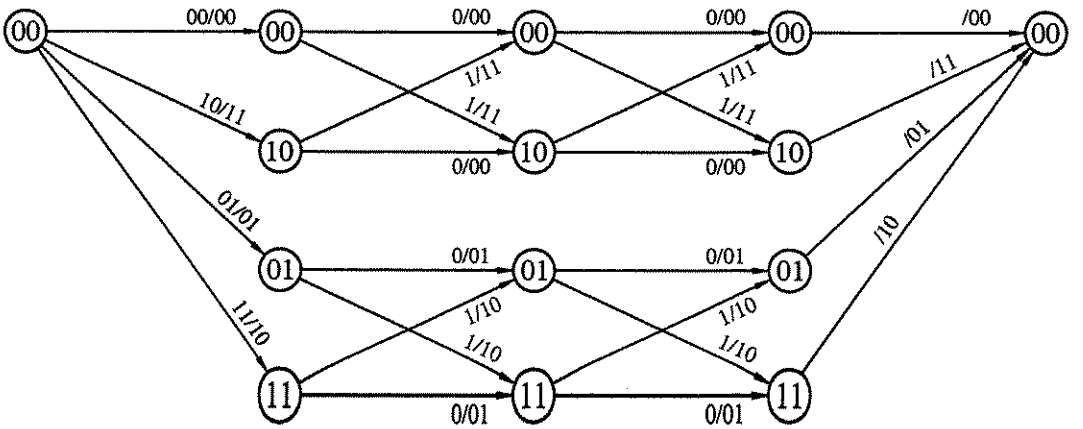

Figura 2.10: Estrutura da treliça para o GAC (8,4,4).

O diagrama de treliça do GAC (7,4,3) é similar ao diagrama de treliça do GAC (8,4,4) e difere apenas na quantidade de dígitos usados para o rótulo dos ramos da última seção da treliça, como mostra a Figura 2.11.

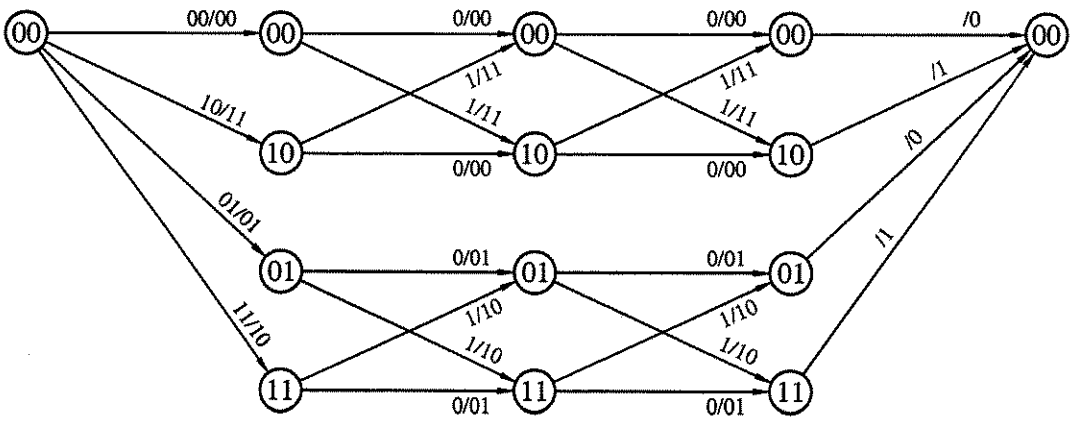

Figura 2.11: Estrutura da treliça para o GAC (7,4,3).

Construção do Diagrama de Treliça dos GACs (24,12,8) e (23,12,7)

Da construção do GAC (24,12,8) vem que $C_1 = (24, 8, 8)$. Isto implica que o código componente de linha $R_1 = (n_1 = 8,\ k_1 = 4,\ d_1 = 4)$ e que o código componente de coluna SPC $= (n_2 = 3,\ k_2 = 2,\ d_2 = 2)$. Então, $N_c = n_2 + 1 = 3 + 1 = 4$ e $N_s = 2^{\max k_l} = 2^7 = 128$.

Cada estado das $p(0, 1, ..., n_2 = 3)$ colunas é identificado com um vetor de sete bits, $S_p(a_1, a_2, a_3, a_4, a_5, a_6, a_7)$; nas colunas $p = 0$ e $p = 3$ a treliça terá somente um estado nomeado $S_0(0000000)$ e $S_3(0000000)$, respectivamente. Os ramos da treliça são rotulados como X_p/C_p. Como houve a necessidade de um segundo código adicional C_3, visto em 2.27, todos os ramos que chegam no último estado, S_3, são ramos paralelos rotulados complementarmente. Seguindo este procedimento, o diagrama de treliça para o GAC (24,12,8) é apresentado na figura 2.12.

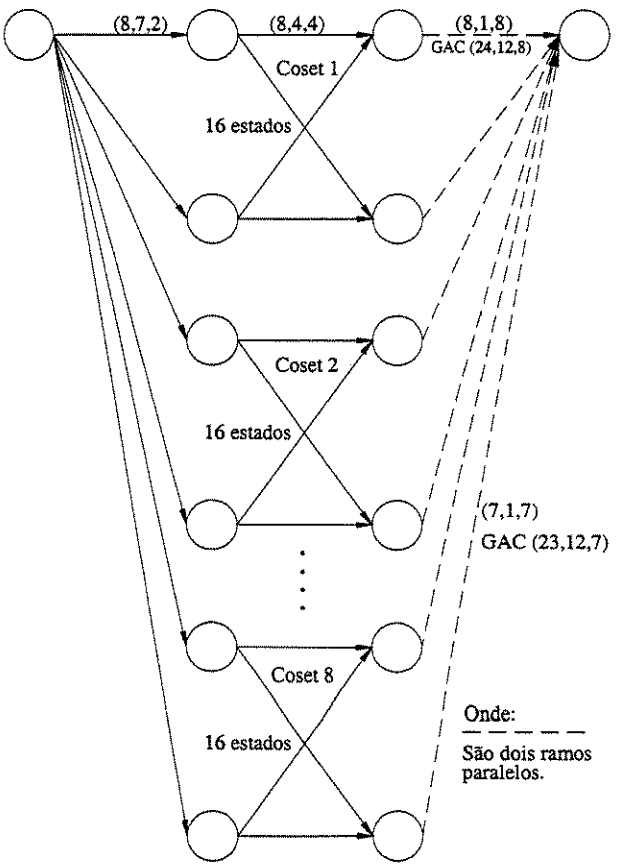

Figura 2.12: Estrutura da treliça para os GACs (24,12,8) e (23,12,7).

Capítulo 3

Sistema de Transmissão da Recomendação IS-136 TDMA

3.1 Introdução

TDMA para telefonia móvel digital é uma técnica de múltiplo acesso onde todos os usuários são associados a uma mesma portadora em diferentes intervalos de tempo (*time slots*) com duração de 40 ms. Cada canal de 30 kHz pode ser compartilhado por três ou seis usuários. Isto é possível devido a divisão do canal em freqüência e no tempo como pode ser visto na figura 3.1, de forma que cada usuário tenha o uso total do canal por um terço do tempo. Desta forma, a capacidade da rede TDMA é pelo menos triplicada em relação ao sistema celular analógico equivalente.

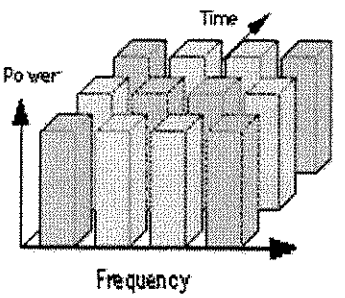

Figura 3.1: Divisão dos canais no Sistema TDMA.

A crescente utilização deste esquema devido à expansão do mercado de telecomunicações, trouxe a necessidade de uma padronização que permitisse a interconexão e a compatibili-

dade dos diversos sistemas. A evolução da especificação de um sistema celular digital parte do padrão EIA-533, AMPS, seguido da IS-$54B$, onde um sistema TDMA básico foi estabelecido, e termina com a introdução de um canal de controle digital, $DCCH$ (*Digital Control Channel*) formando a especificação IS-136. A figura 3.2 mostra esta evolução.

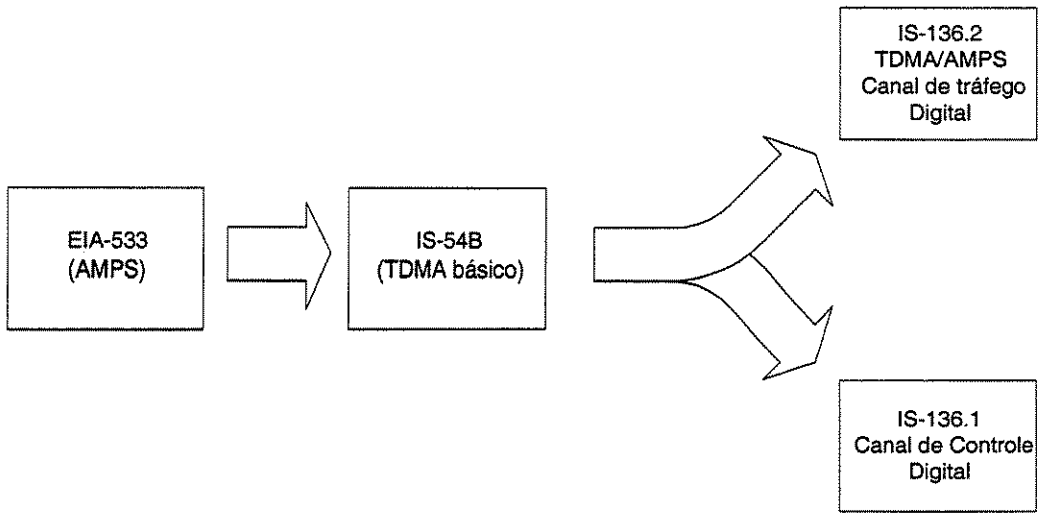

Figura 3.2: Evolução da especificação IS-136.

O padrão IS-136 apresenta em sua documentação uma grande quantidade de especificações técnicas de forma a garantir a compatibilidade entre sistemas de comunicações móveis baseados em TDMA, na faixa de 800 MHz, e serviços de comunicação pessoal, PCS (*Personal Communications Services*), na faixa de 1900 MHz. São estabelecidas normas que propiciam, ao provedor, a prestação de serviços de voz e dados garantindo que a estação móvel tenha acesso a qualquer sistema celular desenvolvido de acordo com essas especificações. Esta padronização não estabelece performance para equipamentos e procedimentos de medição, ou seja, a qualidade do serviço é papel do provedor.

3.2 Fundamentos de um Sistema Celular Móvel IS-136

A principal evolução do padrão IS-136 de Múltiplo Acesso por Divisão Temporal em relação ao padrão anterior IS-$54B$ (padrão ANSI TIA/EIA 627) é a introdução de um canal de controle digital, o $DCCH$. O canal de controle digital é uma coleção de canais

lógicos transmitidos por canais de rádio usando modulação $\frac{\pi}{4}DQPSK$ e tem a finalidade de transmitir informações de controle e mensagens de dados curtas entre as estações base e móvel. Esta modificação faz da especificação IS-136 uma poderosa tecnologia de comunicação pessoal capaz de prover uma grande quantidade de serviços operando tanto na freqüência de 800 MHz quanto na de 1.900 MHz.

As principais vantagens da introdução do $DCCH$ são [Harte98] :

- Maior duração da bateria através do processo chamado *sleep mode;*
- Suporte para múltiplos *vocoders* (Codificadores de Voz), podendo acompanhar a evolução tecnológica;
- Possibilidade de transmissão de dados e aplicativos de e para o aparelho celular móvel (identificadores de chamada, por exemplo);
- Uma hierarquia macrocélula-microcélula provendo um suporte para operação microcelular;
- Habilidade para rapidamente incorporar serviços avançados satisfazendo as necessidades dos consumidores.

Outra característica primária da especificação IS-136 é a coexistência com os sistemas AMPS existentes. Os canais de rádio IS-136 mantêm a mesma largura de faixa de 30 kHz do sistema AMPS e ambos os serviços são oferecidos em um mesmo sistema e células. Para conseguir tal compatibilidade muitas características da primeira geração TDMA (IS-54B) foram mantidas na IS-136, tais como, a estrutura do *slot*, a modulação, processamento de uma ligação e a utilização da mesma codificação de canal tanto para o $DCCH$ como para o canal de tráfego digital DTC (*Digital Traffic Channel*). Isso faz com o que a migração AMPS→TDMA seja feita de forma gradativa sem que prejuízos sejam causados aos usuários do sistema analógico.

3.3 Canal de Tráfego Digital

O canal de tráfego digital DTC é utilizado para transportar informação do usuário ou sinalização da estação móvel para a estação base e vice-versa, correspondendo ao canal reverso e ao canal direto, respectivamente. A figura 3.3 ilustra a utilização do canal de tráfego IS-136.

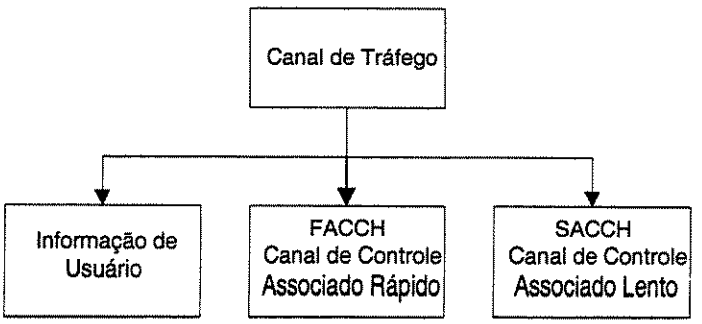

Figura 3.3: Estrutura do Canal de Tráfego Digital (DTC).

Cada bloco da estrutura será detalhado nos próximas subseções. O fundamento de um sistema TDMA consiste na utilização dos canais de rádio de 30 kHz, de forma compartilhada para que vários usuários possam acessar os serviços na mesma faixa de freqüência por multiplexagem temporal.

No sistema IS-136, os canais digitais são *duplex* em freqüência, ou seja, o processo de transmissão e recepção é realizado em freqüências distintas, diminuindo os efeitos de interferência. Além disso, estas freqüências de transmissão e recepção são divididas em intervalos de tempo (*time slots*) caracterizando uma operação *duplex* temporal TDD (*Time Division Duplex*).

No sistema celular, os canais direto e reverso usados em uma ligação telefônica são separados por 45 MHz, sendo que a faixa de $869 - 894$ MHz é dedicada à transmissão base $\rightarrow$ móvel e a faixa de $824 - 849$ MHz para a transmissão móvel $\rightarrow$ base. Para determinar a freqüência central f, em MHz, dado o conhecimento do número do canal N, pode-se utilizar

as fórmulas (3.1) e (3.2).

$$\text{Canal Reverso} : \begin{cases} 0.03N + 825 \quad [MHz] & 1 \leq N \leq 799 \\ 0.03\,(N - 1023) + 825\,[MHz] & 990 \leq N \leq 1023 \end{cases} \tag{3.1}$$

$$\text{Canal Direto} : \begin{cases} 0.03N + 870 \quad [MHz] & 1 \leq N \leq 799 \\ 0.03\,(N - 1023) + 870\,[MHz] & 990 \leq N \leq 1023 \end{cases} \tag{3.2}$$

Além da separação em freqüência dos canais direto e reverso, existe também uma separação temporal entre os sinais transmitidos e recebidos de modo a evitar que ocorra interferência entre os mesmos. Esta separação, denominada de deslocamento no tempo Δt (*time offset*), simplifica o projeto de transmissores e receptores e tem a duração de um quadro mais 45 períodos de símbolo. A figura 3.4 ilustra este procedimento.

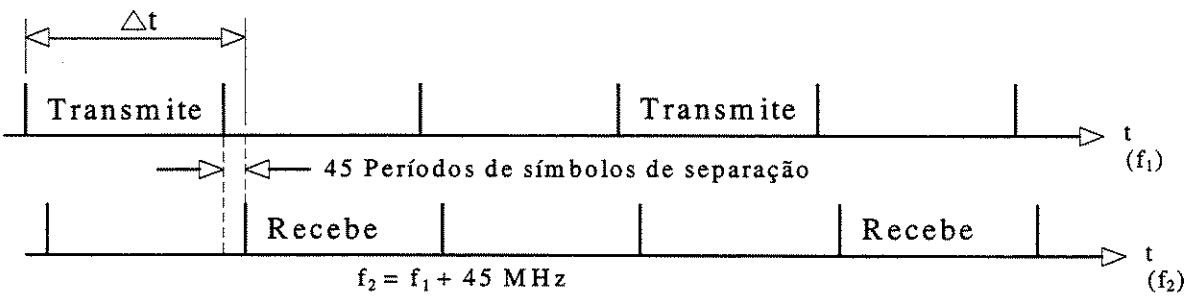

Figura 3.4: Referência de *time offset*.

A utilização dos canais IS-136 pode ser feita de duas formas denominadas **taxa plena** e **meia taxa.**

3.3.1 Taxa Plena

A utilização do quadro TDMA em taxa plena faz com que até 3 usuários por canal tenham acesso simultâneo ao sistema, cada um transmitindo em $\frac{1}{3}$ do quadro, de forma que o usuário A use os *slots* 1 e 4, o usuário B use os *slots* 2 e 5 e o usuário C use os *slots* 3 e

6, mostrando como os canais de rádio-freqüência TDMA são divididos no tempo, figura 3.5. Essa configuração, para a transmissão em taxa plena, proporciona uma taxa de dados bruta de 13 *kbps* [Harte98].

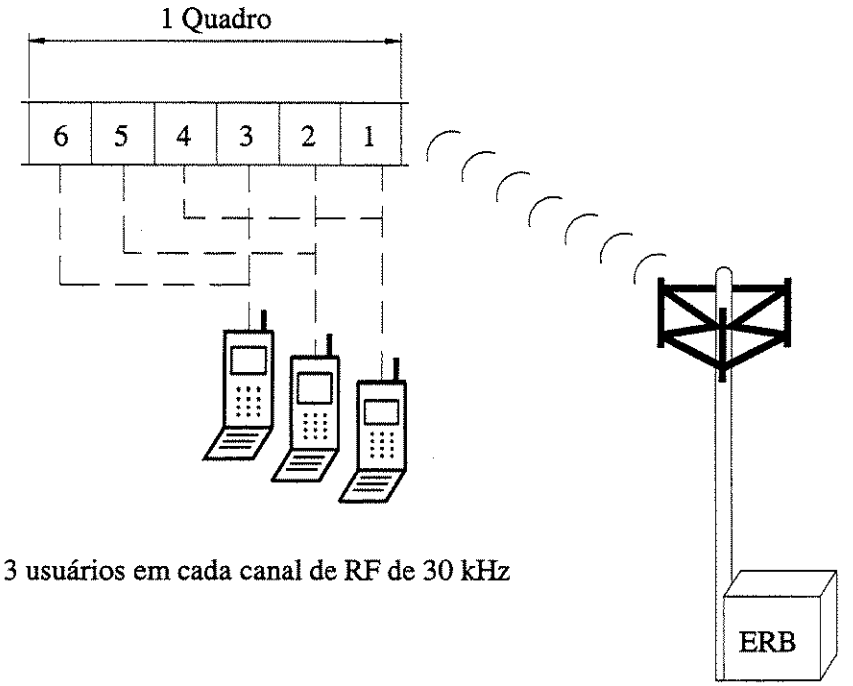

Figura 3.5: Quadro TDMA em taxa plena.

3.3.2 Meia Taxa

Neste caso a capacidade dos canais de *RF* é duplicada com a utilização de um *time slot* por usuário conforme ilustra figura 3.6 [Harte98]. Assim, até 6 usuários podem compartilhar um canal de *RF* na transmissão em meia taxa.

3.4 Estrutura de Quadro TDMA

Cada um dos canais digitais *IS*-136 são divididos em quadros de duração igual a 40 *ms* de comprimento igual a 1944 bits ou 972 símbolos da modulação $\frac{\pi}{4}DQPSK$ (*Differentialy Quadrature Phase Shift Keying*). Estes quadros são subdivididos em 6 *time slots* de 6.67 *ms* cada, composto de 324 bits ou 162 símbolos de acordo com a figura 3.7. A taxa de transmissão

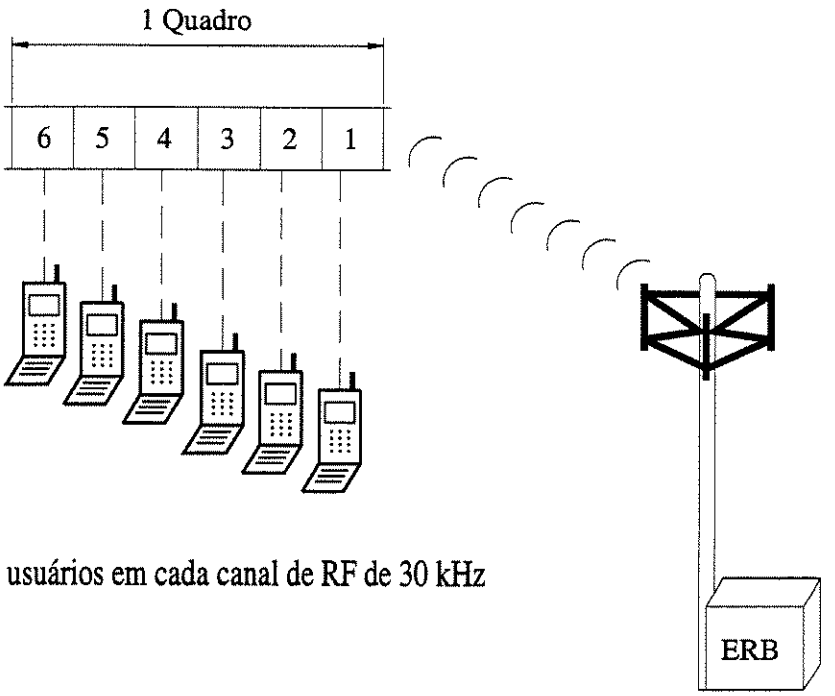

Figura 3.6: Quadro TDMA em meia taxa.

de quadro é de 25 quadros por segundo.

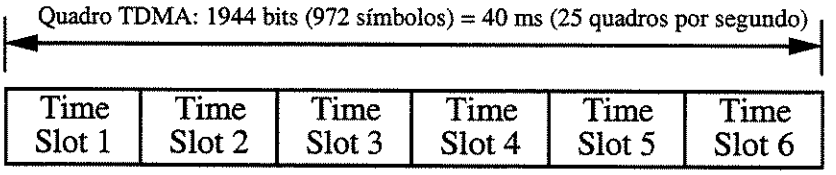

Figura 3.7: Estrutura de Quadro TDMA.

Esses *time slots* são utilizados pelo móvel de modo que os mesmos possam estar ativos (transmissão ou recepção) ou inativos (ociosos). A recomendação *IS*-136 descreve várias estruturas de *time slots* que podem ser utilizadas para a transmissão ou recepção de voz ou dados para o móvel.

3.4.1 Formato do *Slot* de Dados Direto

O *slot* de dados direto é usado para transferir bits de dados ou voz da estação base para o telefone móvel. Este *slot* contém 324 bits sendo 260 dedicados ao assinante. Os *slots*

temporais são divididos em bits de acordo com a figura 3.8 e numerados de 1 à 324, sendo o primeiro bit transmitido do campo **SYNC** corresponde à posição de bit 1 e o último bit transmitido do campo **CDL** é o 324.

SYNC	SACCH	DATA	CDVCC	DATA	RSVD	CDL
28	12	130	12	130	1	11

SYNC : Sincronização e Treinamento

SACCH : Canal de Controle Associado Lento

CDVCC : Código de Cores de Verificação Digital Codificado

DATA : Informação do Usuário

CDL : Localizador de Canal de Controle Digital Codificado

RSVD : Reservado

Figura 3.8: Formato de Slot da Estação Base para a Estação Móvel.

Os 28 bits iniciais de **SYNC** são utilizados para sincronização, treinamento do equalizador e identificação do *slot* temporal. A palavra de sincronismo apresenta boas propriedades de autocorrelação para facilitar a sincronização e treinamento.

O campo seguinte de 12 bits é utilizado para informação de controle **SACCH** (*Slow Associated Control Channel*) que são transmitidos a uma taxa de 300 bps. É importante notar que na medida em que temos um campo reservado para as mensagens **SACCH**, o campo destinado à fala (DATA) não é alterado.

O campo **DATA** de 260 bits (dois campos de 130 bits) carrega a informação de voz do usuário ou sinalização de controle a altas taxas **FACCH** (*Fast Associated Control Channel*) Canal de Controle Associado Rápido que é transmitido a uma taxa de 3.25 *kbps*.

CDVCC é um campo constituído de 12 bits que comporta 255 valores distintos de Códigos de Cores de Verificação Digital Codificado. O mesmo **CDVCC** pode ser utilizado para

todas as transmissões da base e do móvel na mesma célula ou setor. A função deste código é similar à função do *SAT (Supervisory Audio Tone)* para o sistema *AMPS,* onde cada célula é referenciada por um único identificador. O Código de Cores de Verificação Digital (**DVCC**) é uma palavra de 8 bits que é codificada por um código de *Hamming* $(15, 11)$ encurtado para $(12, 8)$ para formar o **CDVCC.**

Todo *slot* de canal de tráfego direto da IS-136 tem um Localizador de Canal de Controle Codificado (**CDL**). Este campo de 12 bits fornece informação que pode ser utilizada pela estação móvel para ajudar na localização de um Canal de Controle Digital ($DCCH$) durante a busca inicial. Um valor **DL** decodificado corretamente indica que um canal de controle digital pode ser encontrado no número de canal de RF na faixa $[8DL + 1, 8DL + 8]$, desde que o número de canal de *RF,* N, seja válido. Um dos bits do campo **CDL** é ajustado para 0 no processo de codificação e não é transmitido como parte do mesmo.

3.4.2 Formato do *Slot* de Dados Reverso

Este *slot* é utilizado para a transferência de dados ou voz da estação móvel para a base tendo dois possíveis formatos : Normal e Encurtado.

Slot Normal

Poucas são as diferenças entre este formato de *time slot* reverso normal e o direto. Basicamente, existe a adição dos campos de tempo de guarda (**G**) e de rampa (**R**), além de um preâmbulo (**PREAM**). A figura 3.9 apresenta o formato de *time slot* reverso normal.

O campo **G** tem duração de 125 μs, ou seja, correspondente a 3 símbolos (6 bits) e protege o sistema de possíveis problemas de sobreposição de *time slots* adjacentes transmitidos por diferentes estações móveis. Durante este tempo, o transmissor da estação móvel está na condição desligado.

O campo **R** representa a rampa de subida de potência com duração de 3 símbolos (6 bits). Durante o tempo de rampa, o transmissor é ativado, progressivamente, para evitar possíveis interferências no canal reverso.

G	R	PREAM	SYNC	DATA	SACCH	CDVCC	DATA
6	6	16	28	122	12	12	122

G : Tempo de Guarda

R : Tempo de Rampa

PREAM : Preâmbulo

Figura 3.9: Formato de Slot da Estação Móvel para a Estação Base.

O campo **PREAM** composto de 16 bits (8 símbolos) permite que a estação base realize o controle automático de ganho (**AGC**). O **PREAM** é usado para a sincronização de símbolo e treinamento.

Surto Encurtado

Esta estrutura de *slot* é utilizada quando o aparelho móvel está operando em células de grande área de cobertura ou logo após uma operação de *hand-off* entre células de diâmetros diferentes. Nestes casos, o móvel envia surtos encurtados até que se adquira um sincronismo com o sistema, evitando dessa forma uma superposição de surtos devido à atrasos de propagação dos sinais de rádio no sentido móvel→base. Esta estrutura está apresentada na Figura 3.10.

É importante notar a presença de vários campos de sincronização facilitando o alinhamento temporal do *slot*. Este *slot* temporal não carrega dados de usuário (fala).

3.4.3 Formato de *Slot* com Dados FACCH

Em certas situações, como processos de *hand-off*, mensagens urgentes de controle se fazem necessárias. Nesses casos, informações de controle são enviadas nos campos **DATA** (260 bits) que eram dedicados aos dados de fala. Esta estrutura denominada Canal de Controle Associado Rápido (**FACCH**) é identificada pela utilização de um diferente código de correção de erro.

G_1	R	Sync	D	Sync	D	V	Sync	D	W	Sync	D	X	Sync	D	Y	Sync	G_2
6	6	28	12	28	12	4	28	12	8	28	12	12	28	12	16	28	22

G_1 : Tempo de guarda de 6 bits
R : Tempo de rampa de 6 bits
Sync : Palavra de sincronismo de 28 bits
D : CDVCC de 12 bits
V : 0000
W : 00000000
X : 000000000000
Y : 0000000000000000
G_2 : Tempo de guarda adicional de 22 bits

Figura 3.10: Formato de um *slot* de surto encurtado.

Ao ser recebido o *slot,* o campo DATA é sempre decodificado como sendo sinal de fala. Quando uma sinalização **FACCH** ocorre, o *CRC* (*Cyclic Redundancy Code*) Código de Redundância Cíclica falhará fazendo com que o decodificador identifique que se trata de uma sinalização de controle. Este procedimento faz com que não seja necessária a adição extra de bits para a identificação de conteúdo do *slot.*

O *slot* de dados **FACCH** é idêntico ao *slot* de fala, apenas a informação contida no campo **DATA** é adequada a cada situação, não atrapalhando a conversação porque é imperceptível ao usuário.

3.5 Transmissão

3.5.1 Processamento do Sinal de Voz

O primeiro passo no processamento do sinal de voz analógico de um sistema móvel digital é a sua conversão em um sinal digital. Este processo se inicia na captação do sinal de voz (fonte analógica) por um microfone. Este sinal elétrico de áudio gerado pelo microfone é muito complexo tendo componentes de altas e baixas freqüências, muitas delas são imperceptíveis

ao ouvido humano e que, portanto, não precisam ser transmitidas. Assim, uma filtragem deste sinal é feita eliminando-se as freqüências abaixo de 300 Hz e acima de 3400 Hz. Este sinal limitado em freqüência é então amostrado à uma taxa de 8000 amostras/segundo e, em seguida, é quantizado e codificado em palavras de 8 bits. O sinal de fala digital tem, conseqüentemente, uma taxa de transmissão de 64 *kbps*.

Para um melhor aproveitamento espectral, este sinal de 64 *kbps* passa por um processo de compressão. Na especificação IS-136, um codificador de fala é utilizado para realizar esta tarefa. Esta codificação resulta em um sinal que preserva todas as características fundamentais da voz humana sem perda na sua qualidade e inteligibilidade. Na recepção, um decodificador é utilizado para reconstruir o sinal de fala original.

Neste processo de compressão, o codificador de fala do esquema IS-136 analisa o sinal digital de 64 *kbps* e o caracteriza por *pitch* (período), volume e outros parâmetros. Após esta caracterização do sinal de fala, um algoritmo de codificação busca em tabelas de código (*codebooks)* a seqüência binária que mais precisamente representa o sinal de entrada. Todo este processamento, ilustrado na figura 3.11, resulta em uma compressão 8 : 1. Logo, a taxa do sinal original de 64 *kbps* é comprimida para uma taxa de 7950 *bps*.

O algoritmo de codificação de fala é um membro da classe de *codecs* de fala conhecido como $CELP$ (*Code Excited Linear Predictive Coding)* Codificação Preditiva Linear com Código de Excitação. Esta técnica faz uso de *codebooks* para quantizar vetorialmente o sinal de excitação. O algoritmo adotado na especificação IS-136 é uma variação do $CELP$, denominado de $VSELP$ (*Vector-Sum Excited Linear Predictive Coding*) Codificação Preditiva Linear de Soma de Vetores de Excitação, que utiliza um *codebook* com uma estrutura pré-definida de tal forma que os cálculos necessários para que o processo de busca do código apropriado seja significativamente reduzido.

Para que a voz seja sintetizada no decodificador de fala, diversos parâmetros devem ser transmitidos ou mesmo pré-definidos para o decodificador. O decodificador de fala utiliza dois *codebooks* de excitação $VSELP$, cada qual com seu respectivo ganho. Estas excitações

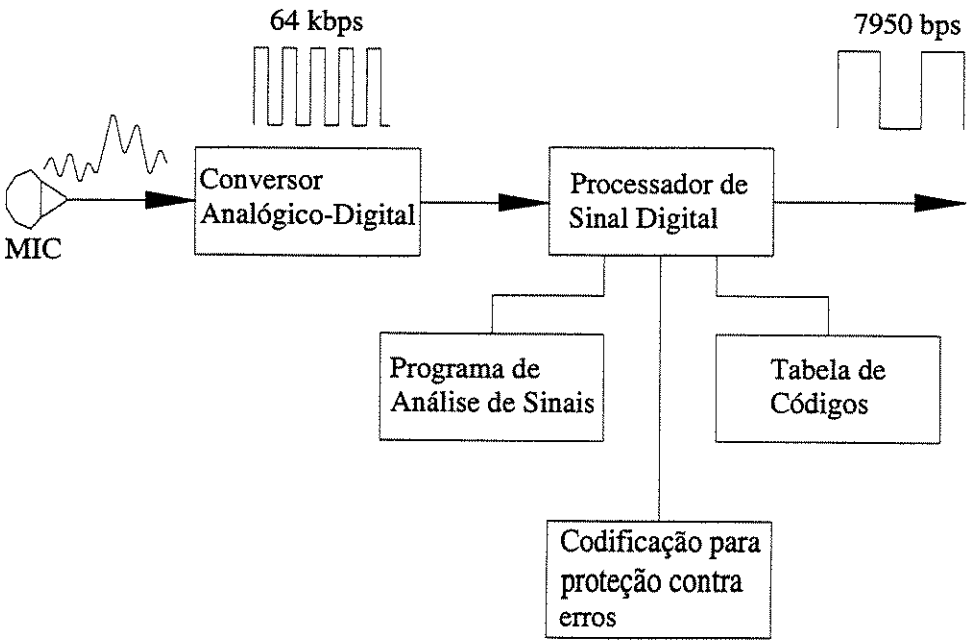

Figura 3.11: Codificação de fala.

são somadas com a saída $b_L(n)$ do filtro de longo prazo, resultando em uma excitação de *codebook* combinada, ex(n). A figura 3.12 ilustra o processo de decodificação.

Uma vez que a taxa de dados na saída do *codec* de fala é 7950 *bps*, teremos 159 *bits* por quadro de fala já que o mesmo tem duração de 20 *ms*. Estes 159 *bits* são distribuídos de acordo com a tabela 3.1, onde :

- Coeficientes do filtro, α_i's, são coeficientes de predição do filtro de curto prazo. O filtro de curto prazo é equivalente ao filtro de síntese visto na figura 3.12. Os parâmetros de

Descrição	bits/quadro
Coeficientes do filtro, α_i's	38
Energia de quadro, $R(0)$	5
Lag (retardo), L	28
Palavras código, I e H	56
Ganhos, β, γ_1 e γ_2	32

Tabela 3.1: Distribuição dos bits no quadro TDMA.

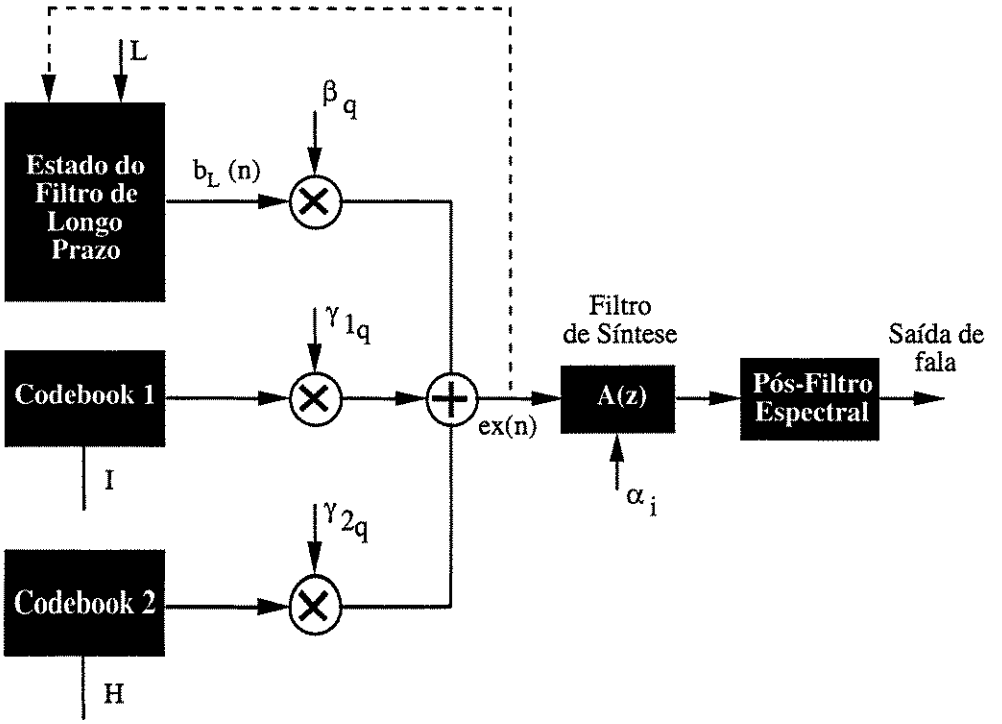

Figura 3.12: Diagrama de blocos do decodificador.

predição de curto prazo são calculados da fala de entrada. A função de transferência $A(z)$ do filtro de curto prazo é dada por

$$A(z) = \frac{1}{1 - \sum_{i=1}^{N_p} \alpha_i z^{-i}} \tag{3.3}$$

onde N_p é a ordem do preditor de curto prazo $= 10$.

- Energia de quadro, $R(0)$. O valor de energia é calculado e codificado uma vez por quadro. Este valor de energia, $R(0)$, reflete a potência média do sinal da fala na entrada, dentro de 20 ms de intervalo que é centrado em relação ao meio do 4°subquadro. $R(0)$ pode ser calculada por

$$R(0) = \frac{\phi(0,0) + \phi(N_p, N_p)}{2(N_A - N_p)} = \frac{\phi(0,0) + \phi(10,10)}{320} \tag{3.4}$$

onde : $\phi(i,k)$ é a matriz covariância (autocorrelação) da fala de entrada

$N_A = N_F + N_p = 170$ amostras

N_F é o comprimento do quadro (20 ms) $= 160$ amostras

$R(0)$ pode ser convertida para dB em relação a seu valor máximo assumido, isto é :

$$R_{dB} = 10 \log_{10} \left[\frac{R(0)}{R_{\max}} \right] \tag{3.5}$$

R_{dB} é quantizada em 32 níveis. Um valor zero para R0 corresponde a uma energia de 0 $[dB]$, $[R(0) = 0]$. Este código pode ser usado para um decodificador totalmente em silêncio. Os restantes 31 valores de quantização de R_{dB} variam de um mínimo de -64 (correspondente a um código de 1 para R0) a um máximo de -4 (correspondente a um código de 31 para R0). O tamanho do passo do quantizador é 2 dB.

- Processamento de Subquadro : O quadro de fala de 20 ms é subdividido em 4 subquadros de 5 ms. Para cada subquadro, o codec de fala deve determinar e codificar o atraso de predição de longo prazo L, as duas palavras códigos I e H e os ganhos β, γ_1 e γ_2. O retardo L é sempre usado como o atraso para as primeiras L amostras do subquadro.

3.5.2 Codificação de Canal

O controle de erro de canal para os bits gerados no codificador de fala do sistema IS-136 emprega três mecanismos : Codificação CRC, Codificação Convolucional e Entrelaçamento.

Estes três mecanismos são usados para combater os erros que por ventura possam ocorrer devido ao ruído $AWGN$ presente no canal, além do desvanecimento do sinal transmitido que chega ao receptor por múltiplos percursos.

O primeiro passo no processo de codificação consiste na separação dos 159 bits de informação do quadro de fala em duas classes de bits : classe 1 formada por 77 bits e classe 2 por 82 bits. Os 12 bits perceptualmente mais significativos do conjunto de bits classe 1 são aplicados à um Codificador de Redundância Cíclica (CRC) onde são adicionados 7 bits de redundância com o propósito de detecção de erro. Assim, os 84 bits resultantes na classe 1 entram num codificador convolucional de taxa $\frac{1}{2}$.

Os bits classe 2 são transmitidos sem proteção contra erros inseridos no canal porque são menos significativos para o codificador de voz uma vez que os erros que por ventura venham

a ocorrer nesses bits são considerados irrelevantes à percepção humana.

A figura 3.13 ilustra todo o processo de proteção contra erros no sistema IS-136.

Todos os bits de saída do codificador convolucional mais os bits de classe 2 podem ser criptografados (se desejado) antes de serem enviados para o entrelaçador.

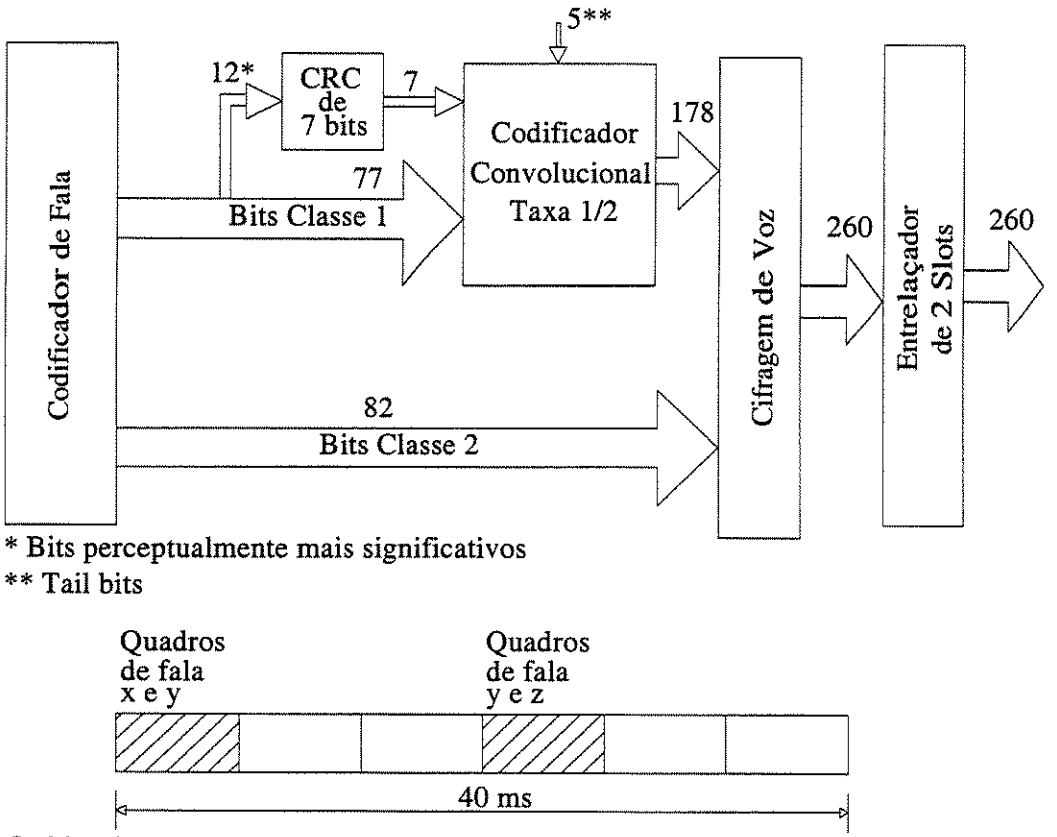

Figura 3.13: Estrutura de proteção desigual contra erros de canal da IS-136.

Código de Redundância Cíclica (CRC)

O código de redundância cíclica acrescenta 7 bits aos 12 bits mais significativos da classe 1 caracterizando um código de bloco $(n, k) = (19, 12)$. O polinômio gerador do CRC para o esquema IS-136 é dado por :

$$g_{CRC}(X) = 1 + X + X^2 + X^4 + X^5 + X^7 \tag{3.6}$$

Representando os doze bits na entrada do CRC na forma polinomial teremos :

$$a(X) = a_{11}X^{11} + a_{10}X^{10} + a_9X^9 + \cdots + a_0X^0 \tag{3.7}$$

Dessa forma o polinômio de paridade $b(X)$ é dado pelo resto da divisão de $a(X)$ deslocado de sete posições à esquerda, pelo polinômio gerador

$$a(X).X^7 = q(X).\left(1 + X + X^2 + X^4 + X^5 + X^7\right) + b(X)$$

onde $q(X)$ é o quociente da divisão. No processo de codificação apenas o polinômio $b(X)$ é relevante, sendo o polinômio $q(X)$ descartado.

A forma geral do polinômio de paridade $b(X)$ é, portanto :

$$b(X) = b_6X^6 + b_5X^5 + \cdots + b_0X^0$$

Exemplo 3 *Sejam os* 12 *bits classe* 1 *mais significativos do quadro de fala dados por :*

$$\{a_{11}, a_{10}, \cdots, a_0\} = \{0,0,1,1,0,0,1,1,0,0,0,1\}$$

Na representação polinomial :

$$a(X) = X^9 + X^8 + X^5 + X^4 + 1$$

Portanto

$$\underbrace{\left(X^9 + X^8 + X^5 + X^4 + 1\right)}_{a(X)}.X^7 + b(X) = q(X).g_{CRC}(X)$$

onde

$$a(X).X^7 = X^{16} + X^{15} + X^{12} + X^{11} + X^7 \tag{3.8}$$

O resto da divisão de (3.8) por (3.6) será :

$$b(X) = X^6 + X + 1$$

Logo os 7 bits adicionais de paridade gerados pelo CRC são $\{1,0,0,0,0,1,1\}$.

Codificação Convolucional

Da figura 3.13 observa-se que existem 89 bits na entrada do codificador convolucional, 84 proveniente do codificador de fala + CRC e 5 bits adicionais denominados *tail* bits. Como o codificador convolucional possui 5 unidades de memória, os *tail* bits (todos zeros) são necessários para que o decodificador de Viterbi na recepção possa decodificar corretamente os últimos bits de informação. Antes de ser realizada a codificação desses bits, os mesmos são colocados em um arranjo de forma que os 77 bits classe 1 do codificador de fala sejam posicionados de 4 até 80. As posições que vão de 0 até 3 e 81 até 83 são reservadas para os bits de redundância inseridos pelo CRC (7 bits). Nos lugares remanecentes, de 84 até 88, são alocados os *tail* bits. A figura 3.14 mostra o arranjo de bits que entram no codificador convolucional ordenadamente da posição 0 até 88.

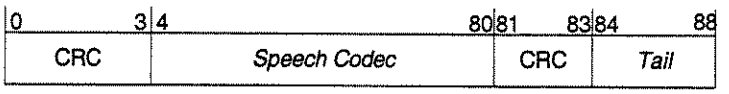

Figura 3.14: Arranjo de bits na entrada do codificador convolucional.

O codificador convolucional adotado na recomendação IS-136 [IS-136.2] é de taxa igual a $\frac{1}{2}$ e memória igual a 5. Sendo assim, este código possui $2^5 = 32$ estados.

Os polinômios geradores deste código covolucional são :

$$g_0(D) = 1 + D + D^3 + D^5 \tag{3.9}$$

$$g_1(D) = 1 + D^2 + D^3 + D^4 + D^5 \tag{3.10}$$

O bits de saída do codificador convolucional são gerados alternadamente pelos polinômios $g_0(D)$ e $g_1(D)$. A partir dos polinômios geradores obtém-se o diagrama do codificador ilustrado na figura 3.15.

Da referência [Cos92], verifica-se que a distância livre (d_{free}) deste código é 8 tendo portanto uma capacidade de correção de $\left\lfloor \frac{d_{free}-1}{2} \right\rfloor = \left\lfloor \frac{8-1}{2} \right\rfloor = 3$ erros, onde o operador $\lfloor x \rfloor$ representa o maior inteiro menor que x.

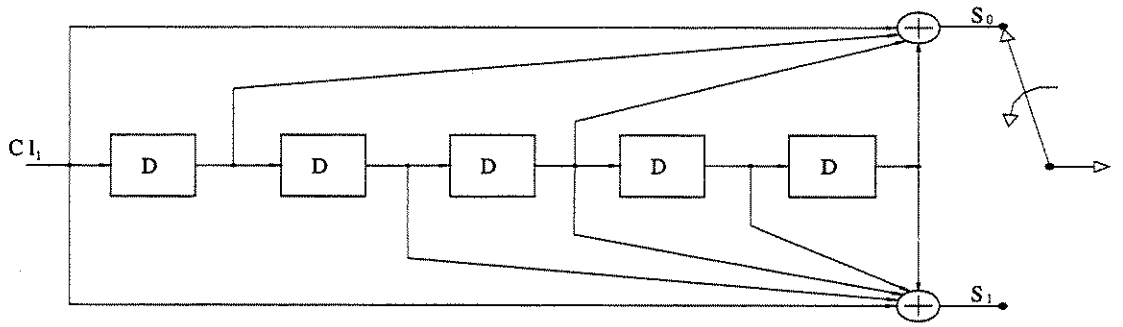

Figura 3.15: Codificador convolucional de taxa $\frac{1}{2}$ e $m = 5$.

Entrelaçador

A maioria dos códigos desenvolvidos para transmissão digital apresentam melhor desempenho quando os erros inseridos no canal de comunicação são estatisticamente independentes. Este é o caso do canal $AWGN$. Entretanto, para um sistema de comunicação sem fio, o canal é caracterizado por múltiplos percursos e desvanecimento do sinal transmitido. O desvanecimento devido à propagação multipercurso variante com o tempo faz com que o nível do sinal caia abaixo do nível de ruído podendo provocar uma seqüência longa de erros. Este surto (*burst*) de erros torna ineficaz a codificação convencional para canais com erros correlacionados.

Uma maneira eficiente para descorrelacionar os erros em canais multipercurso é entrelaçar os dados codificados transformando este canal com erros independentes. Desta forma, pode-se utilizar um codificador convencional neste tipo de canal multipercurso.

Na recomendação IS-136 os dados de fala codificados são entrelaçados em dois *slots* de tempo junto com os dados de fala codificados dos quadros adjacentes, ou seja, cada *slot* de tempo contém informação de dois quadros do *codec* de fala. Os 260 bits que formam o quadro de fala codificado são arranjados em uma matriz de dimensão (26×10) da seguinte forma :

$$\begin{bmatrix} 0x & 26x & 52x & 78x & 104x & 130x & 156x & 182x & 208x & 234x \\ 1y & 27y & 53y & 79y & 105y & 131y & 157y & 183y & 209y & 135y \\ 2x & 28x & 54x & 80x & 106x & 132x & 158x & 184x & 210x & 136x \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 12x & 38x & 64x & 90x & 116x & 142x & 168x & 194x & 220x & 258x \\ 13y & 39y & 65y & 91y & 117y & 143y & 169y & 195y & 221y & 259y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 24x & 50x & 76x & 102x & 128x & 154x & 180x & 206x & 232x & 258x \\ 25y & 51y & 77y & 103y & 129y & 155y & 181y & 207y & 233y & 259y \end{bmatrix} \quad (3.11)$$

Os dados entram coluna-a-coluna e são transmitidos linha-a-linha. Na matriz anterior, x faz referência ao quadro de fala anterior enquanto que y ao quadro atual.

Os dados são colocados na matriz de modo a haver um entrelaçamento entre os bits classe 2 do *codec* de fala e os bits classe 1 codificados convolucionalmente. Os bits classe 2 são seqüencialmente colocados na matriz nas seguintes posições :

0, 26, 52, 78

93 a 129

130, 156, 182, 208

223 a 259

As posições remanescentes são ocupadas, seqüencialmente, pelos bits classe 1.

3.5.3 Modulação

O padrão IS-136 utiliza-se de um esquema de modulação por fase onde os bits de informação (voz ou dados) são convertidos em deslocamentos de fase do sinal de rádio freqüência (RF). Essa modulação, conhecida como deslocada $\frac{\pi}{4}$ e codificada diferencialmente, $\frac{\pi}{4}DQPSK$ (*Differentialy Quadrature Phase Shift Keying*), foi empregada pela [IS-136.2] para manter a eficiência espectral e otimizar a seção de amplificação de RF [Harte98]. Os símbolos são transmitidos nas mudanças de fase ao invés de valores absolutos de fases. Desde que diferentes padrões de bits na entrada do modulador provocam específicos deslocamentos de fase

nos sinais transmitidos, esses bits podem ser recuperados bastando para isso que os sinais RF recebidos sejam amostrados, para transições de fase e amplitude, em específicos períodos de tempo. Nota-se, do diagrama de fase da modulação $\frac{\pi}{4}DQPSK$, na figura 3.16, que apenas quatro possíveis deslocamentos de fase são permitidos $\{+\frac{\pi}{4}, -\frac{\pi}{4}, +\frac{3\pi}{4}, -\frac{3\pi}{4}\}$, a partir de uma dada fase. O demodulador procura através da observação do sinal recebido no período atual e anterior, estimar o padrão de bits transmitido. Observa-se que um conhecimento absoluto da fase dos sinais recebidos não se faz necessário já que a informação está contida na diferença de fase entre os sinais atual e anterior.

O período de transição entre os pontos da constelação é de 41.15 μs resultado em uma taxa de símbolo de $\frac{1}{41.15\mu s} \approx 24300\ \frac{símbolos}{s}$. Como cada símbolo corresponde à 2 bits, a taxa de dados na entrada do modulador é 48.6 *kbps*.

É importante destacar que os sinais da constelação são mapeados usando código *Gray* entre os *di-bits*. A figura 3.16 mostra que o diagrama de modulação pode ser dividido em dois subconjuntos formados pelos sinais ímpares $\{1, 3, 5, 7\}$ sendo representados pelo símbolo $\circ$ e os sinais pares $\{0, 2, 4, 6\}$ representados pelo símbolo $\bullet$.

O processo de modulação se inicia com a conversão série-paralelo da seqüência de bits de dados b_m em duas outras X_k e Y_k de acordo com a figura 3.17. Nessa conversão, os bits b_m são numerados a partir das posições $1, 2, ..., 6$. Assim os bits das posições ímpares são associados à seqüência X_k e os bits das posições pares à seqüência Y_k. Por exemplo, para $b_m = \{0, 1, 0, 1, 0, 1\}$, teríamos $X_k = \{0, 0, 0\}$ e $Y_k = \{1, 1, 1\}$.

Os dados digitais X_k e Y_k são convertidos em I_k e Q_k de acordo com as seguintes equações [IS-136.2]

$$I_k = I_{k-1} \cos\left[\Delta\Phi\left(X_k, Y_k\right)\right] - Q_{k-1} \sin\left[\Delta\Phi\left(X_k, Y_k\right)\right] \tag{3.12}$$

e

$$Q_k = I_{k-1} \sin\left[\Delta\Phi\left(X_k, Y_k\right)\right] + Q_{k-1} \cos\left[\Delta\Phi\left(X_k, Y_k\right)\right] \tag{3.13}$$

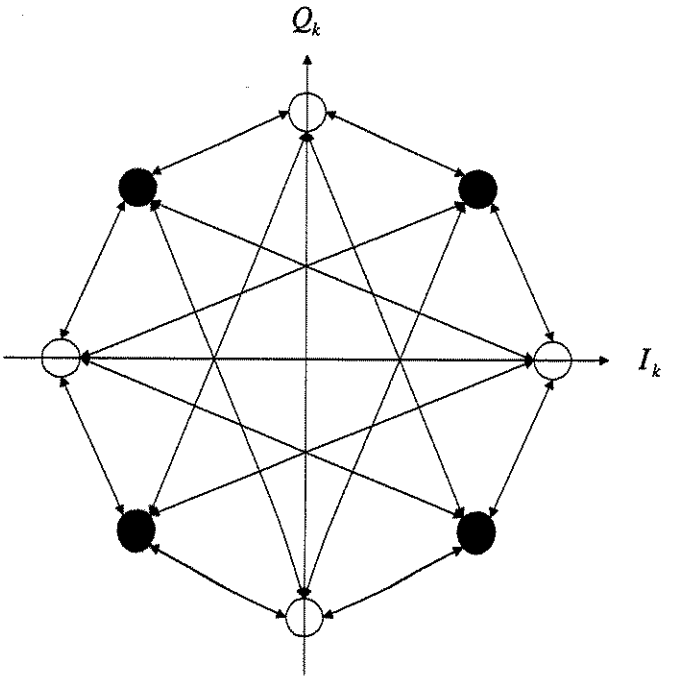

Figura 3.16: Constelação $\frac{\pi}{4}$ $DQPSK$.

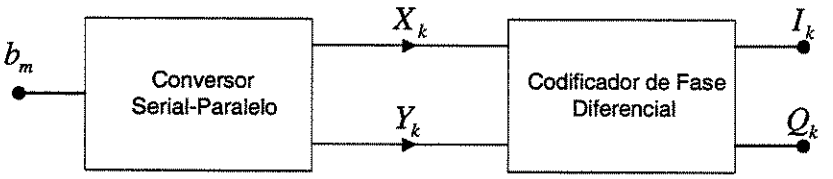

Figura 3.17: Conversor serial-paralelo e codificador diferencial.

onde I_{k-1} e Q_{k-1} são os sinais gerados no período anterior. O parâmetro de mudança de fase $\Delta\Phi$ é determinado de acordo com a tabela 3.2

É importante notar que os sinais I_k e Q_k na saída do codificador de fase diferencial podem assumir um dos cinco valores do conjunto $\left\{0, \pm 1, \pm\frac{1}{\sqrt{2}}\right\}$, resultando na constelação da figura 3.16.

Como exemplo, suponha que os sinais I_{k-1} e Q_{k-1} sejam 1 e 0, respectivamente e que na entrada do codificador diferencial da figura 3.17 se tenha $X_k = 1$ e $Y_k = 0$. Dessa forma, de acordo com a tabela anterior, tem-se um deslocamento de fase de $\Delta\Phi = -\frac{\pi}{4}$. Aplicando este resultado nas equações (3.12) e (3.13) tem-se na saída do codificador de fase diferencial os

X_k	Y_k	$\Delta\Phi\ [rad/s]$
1	1	$-3\pi/4$
0	1	$3\pi/4$
0	0	$\pi/4$
1	0	$-\pi/4$

Tabela 3.2: Determinação do deslocamento de fase da modulação *DQPSK*.

sinais $(I_k, Q_k) = \left(-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right)$.

Os sinais gerados pelo codificador diferencial são aplicados a filtros em banda básica passa-baixa que possuem resposta de fase linear e função de transferência dada pela raiz quadrada do cosseno levantado, isto é,

$$|H(f)| = \begin{cases} 1 & 0 \leq f \leq \frac{(1-\alpha)}{2T} \\ \sqrt{\frac{1}{2}\left\{1 - \sin\left[\frac{\pi(2fT-1)}{2\alpha}\right]\right\}} & \frac{(1-\alpha)}{2T} \leq f \leq \frac{(1+\alpha)}{2T} \\ 0 & f > \frac{(1+\alpha)}{2T} \end{cases} \tag{3.14}$$

onde T é o período de símbolo, f é a freqüência e α é o fator de *roll-off* que determina a largura da banda de transmissão. Para o sistema *IS-136*, α é fixado em 0.35.

Para criar o sinal modulado $\frac{\pi}{4}DQPSK$, dois sinais de amplitude modulada (AM) defasados de 90° são combinados dando origem ao sinal $S(t)$ de acordo com a figura 3.18.

O sinal $S(t)$ é dado por [IS-136.2] :

$$S(t) = \sum_n g(t - nT)\cos(\Phi_n)\cos(\omega_c t) - \sum_n g(t - nT)\sin(\Phi_n)\sin(\omega_c t) \tag{3.15}$$

onde $g(t)$ é a função formatadora do pulso, ω_c é a freqüência da portadora em radianos e Φ_n é a fase absoluta correspondente ao n-ésimo intervalo de símbolo. Esta fase resulta da codificação diferencial que é dada por [IS-136.2] :

$$\Phi_n = \Phi_{n-1} + \Delta\Phi_n \tag{3.16}$$

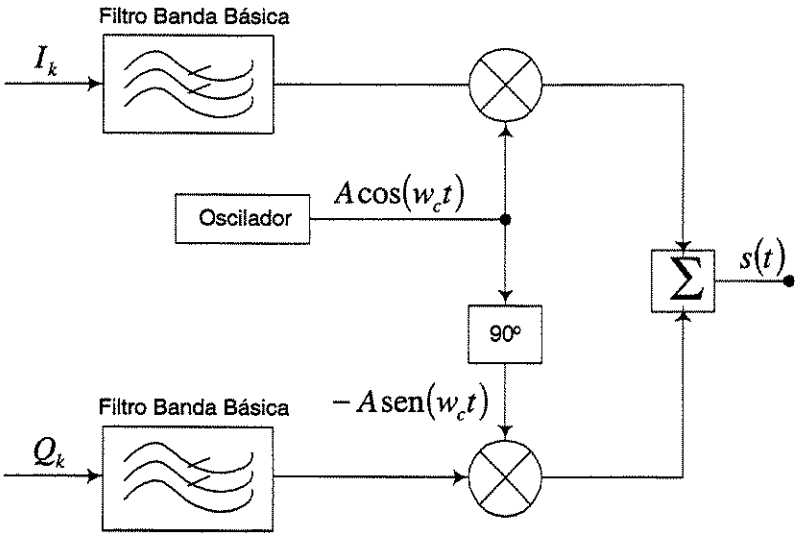

Figura 3.18: Geração do sinal transmitido.

3.6 Recepção

A figura 3.19 ilustra o processo de recepção de acordo com a norma IS-136. A descrição de cada estágio será feita nas subseções seguintes.

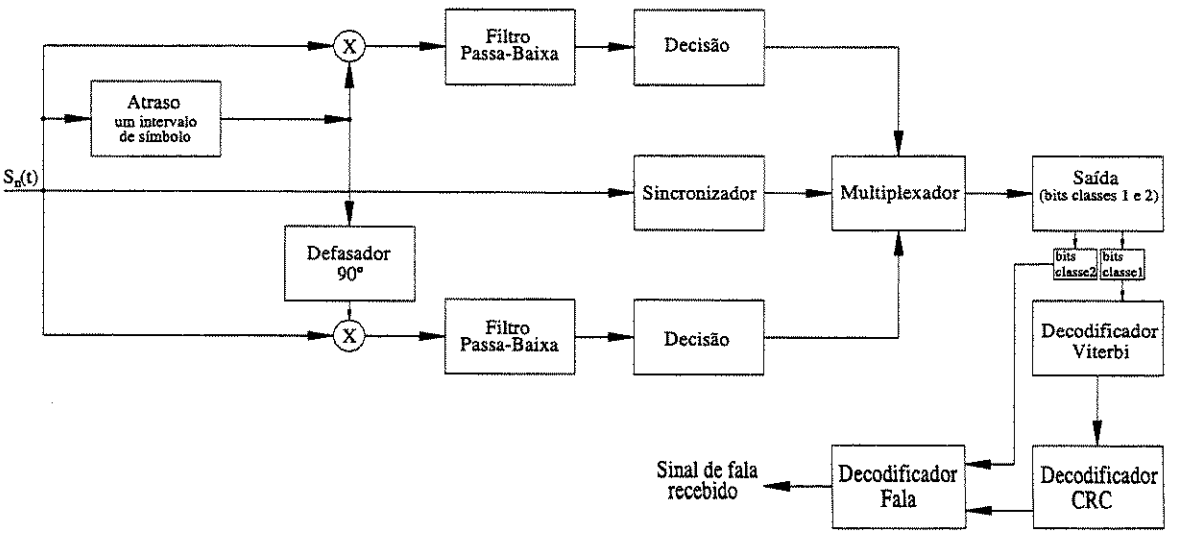

Figura 3.19: Processo de recepção da estação móvel

3.6.1 Demodulação $\frac{\pi}{4}DQPSK$

Considere um sinal recebido $S_n(t) = A\cos(2\pi f_c t + \phi_n)$ e sua versão atrasada $S_{n_{-1}}(t) = A\cos(2\pi f_c t + \phi_{n_{-1}})$. Realizando a multiplicação $S_n(t).S_{n_{-1}}(t)$ e submetendo este resultado ao integrador $\frac{1}{T}\int_0^T dt$ (filtro passa-baixa) da figura 3.20, uma componente do sinal proporcional a $\cos(\phi_n - \phi_{n_{-1}})$ é obtida. Como, da equação 3.16, $\Phi_n - \Phi_{n-1} = \Delta\Phi_n$ e a mudança de fase $\Delta\Phi_n$ assume os valores dados na tabela 3.2, então $\cos\Delta\Phi_n = \pm\frac{1}{\sqrt{2}}$. Portanto, a componente em fase da informação é extraída do sinal da variável de decisão. Do mesmo modo, a componente em quadratura pode ser obtida multiplicando $S_n(t)$ e sua versão atrasada $S_{n_{-1}}(t)$ com deslocamento de fase de $\frac{\pi}{2}$ rad. O sinal resultante após o filtro passa-baixa é proporcional a $\sin(\phi_n - \phi_{n_{-1}})$, assumindo valores $\pm\frac{1}{\sqrt{2}}$. A figura 3.20 mostra o diagrama de bloco do demodulador.

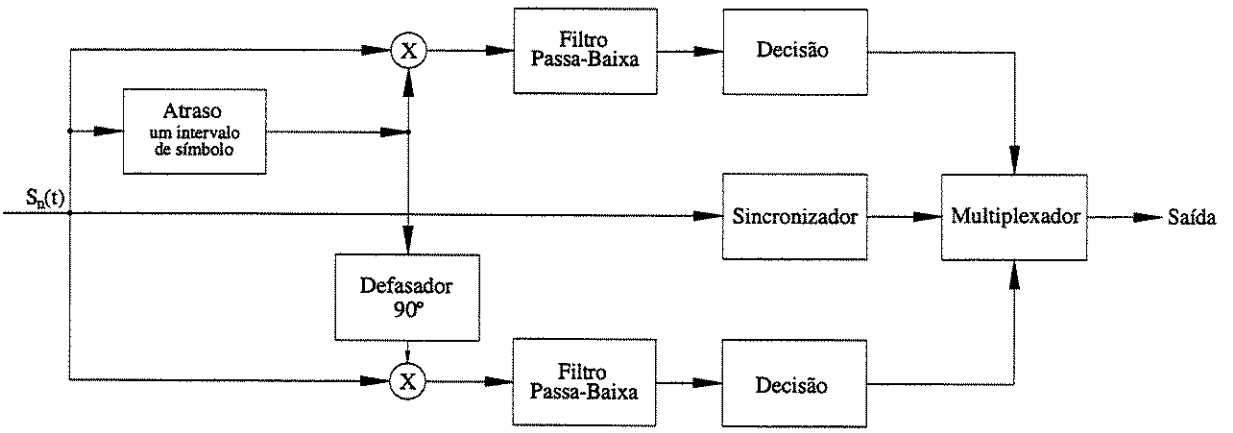

Figura 3.20: Demodulador $\frac{\pi}{4}$ $DQPSK$.

3.6.2 De-entrelaçamento

Após a demodulação, os símbolos detectados são de-entrelaçados. Sabe-se, previamente, que cada *slot* de tempo contém a informação entrelaçada da parte de dois quadros do codificador de fala.

Diferentemente do entrelaçador, o quadro x é o quadro do *codec* de fala presente e o quadro y será o próximo quadro. Os dados recebidos da demodulação são alocados linha-a-linha em

um arranjo 26 × 10 de de-entrelaçamento como é mostrado a seguir :

$$\begin{bmatrix} 0x & 26x & 52x & 78x & 104x & 130x & 156x & 182x & 208x & 234x \\ 1y & 27y & 53y & 79y & 105y & 131y & 157y & 183y & 209y & 135y \\ 2x & 28x & 54x & 80x & 106x & 132x & 158x & 184x & 210x & 136x \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 12x & 38x & 64x & 90x & 116x & 142x & 168x & 194x & 220x & 258x \\ 13y & 39y & 65y & 91y & 117y & 143y & 169y & 195y & 221y & 259y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 24x & 50x & 76x & 102x & 128x & 154x & 180x & 206x & 232x & 258x \\ 25y & 51y & 77y & 103y & 129y & 155y & 181y & 207y & 233y & 259y \end{bmatrix} \quad (3.17)$$

Os dados são então lidos coluna-a-coluna e uma vez que todos os dados do quadro x (atual) estarão disponíveis poderemos proceder com a decodificação do mesmo.

3.6.3 Decodificação Convolucional

Depois do de-entrelaçamento, os dados de um quadro inteiro do *codec de* fala estão agora disponíveis. Os dados do de-entrelaçamento que foram codificados convolucionalmente agora devem ser decodificados usando o algoritmo de Viterbi recomendado na [IS-136.2].

3.6.4 Código de Redundância Cíclica (CRC)

Após a decodificação dos bits classe 1, os 12 bits mais significativos são verificados através dos 7 bits de redundância introduzidos pelo codificador CRC no transmissor. Para realizar a detecção de erro, um segundo polinômio CRC de teste de paridade $[b'(X)]$ é gerado a partir dos 12 bits mais significativos e comparado ao polinômio CRC recebido $[b'_{rec}(X)]$.

O segundo polinômio é gerado utilizando-se a informação recebida e o polinômio gerador CRC :

$$g_{CRC}(X) = 1 + X + X^2 + X^4 + X^5 + X^7 \quad (3.18)$$

Os 12 bits mais significativos decodificados do quadro formam o polinômio de entrada do

decodificador CRC :

$$a'(X) = a'_{11}X^{11} + a'_{10}X^{10} + \cdots + a'_1X + a'_0 \tag{3.19}$$

onde a'_i são os bits classe 1 decodificados de parâmetros LPC1 a LPC5 (detalhado na subseção a seguir) e R0 (tabela 3.1).

O polinômio recebido $a'(X)$ é deslocado para esquerda de sete posições e dividido pelo polinômio gerador $g_{CRC}(X)$ obtendo o polinômio de verificação de paridade $b'(X)$ (resto da divisão) :

$$a'(X).X^7 = q'(X).g_{CRC}(X) + b'(X)$$

O polinômio de paridade recebido $b_{rec}(X)$ é extraído dos bits adicionais classe 1 decodificados no estágio anterior e representado pela seguinte expressão

$$b'_{rec}(X) = b'_{r6}X^6 + b'_{r5}X^5 + \cdots + b'_{r1}X + b'_{r0}$$

onde b'_{r_i} são os bits de paridade do parâmetro CRC recebidos.

O polinômio recebido $b'_{rec}(X)$ é então comparado bit-a-bit ao polinômio $b'(X)$ gerado no receptor. Em caso de diferença, um ou mais erros foram detectados nos 12 bits perceptualmente mais significativos do quadro de fala.

3.6.5 Mascaramento de Quadro Ruim

Um erro nos bits classe 1 perceptualmente mais significativos podem ocorrer devido ao canal de comunicação ou pelo envio da mensagem de controle **FACCH** no lugar dos dados de fala. Uma vez que dados corrompidos por ruído pode causar uma severa degradação na qualidade do sinal de fala gerado no receptor, uma estratégia de mascaramento de quadro ruim é empregada.

Neste sistema, uma máquina de 7 estados é utilizada, onde uma mudança de estado corresponde a uma falha na decodificação do CRC. A contagem de estados indica quantos quadros consecutivos tiveram falha de paridade do CRC. Por exemplo, quando a máquina

Parâmetros apresentados	Total de bits da palavra código	Bits de classe 1	Bits de classe 2	Bits perceptualmente mais significativos
$R(0)$	5	4	1	3
$LPC1$	6	4	2	3
$LPC2$	5	3	2	2
$LPC3$	5	3	2	2
$LPC4$	4	2	2	1
$LPC5$	4	1	3	1

Tabela 3.3: Determinação da quantidade de bits de R(0) e dos coeficientes de predição

está no estado 5 concluímos que houve erro de recepção nos 12 bits perceptualmente mais significativos em 5 quadros consecutivos recebidos (incluindo o atual). A única exceção ocorre no estado 6, no qual um número indefinido de quadros consecutivos corrompidos podem ter sido detectados.

Na ocorrência de uma verificação de paridade correta, a máquina retorna para o estado inicial (estado 0), com excessão do estado 6 que requer duas decodificações corretas para retornar ao estado 0. Isto fornece uma proteção adicional durante intervalos prolongados de condições de canal ruins que poderiam indicar, erradamente, dados válidos de fala. Assim, os procedimentos tomados pelo decodificador de fala em cada estado são :

Estado 0 : Não é detectado erro. Os dados de fala recebidos são utilizados normalmente. O sistema permanece no estado 0 a menos que seja detectado um erro pelo *CRC*.

Estado 1 : Um erro foi detectado nos bits perceptualmente mais significativos. Os parâmetros para $R(0)$ e os bits $LPCs$ são trocados pelos valores do último quadro que estava no estado 0. Os demais bits não são modificados. Tanto $R(0)$ que representa a energia de quadro como visto na tabela 3.1 quanto os $LPCs$ que são os coeficientes de predição linear do filtro de síntese do codec de fala estão representados na tabela 3.3.

Estado 2 : É tomado procedimento similar do estado 1.

Estado 3 : Do mesmo modo que nos estados 1 e 2, uma repetição de quadro é feita. No entanto, o parâmetro $R(0)$ é atenuado em 4 dB.

Estado 4 : Repetição do quadro anterior com nova atenuação de 4 dB do parâmetro $R(0)$.

Estado 5 : Mesmo procedimento que o estado 4.

Estado 6 : O quadro é repetido, mas desta vez, $R(0)$ é zerado tornando completamente muda a saída de fala. Alternativamente, um ruído de conforto pode ser inserido no lugar do sinal de fala.

3.6.6 Decodificação de Fala

O decodificador de fala toma os dados a 7950 *bps* do decodificador de canal e gera o sinal recebido. O decodificador de fala deve operar corretamente em conjunto com o *codec* de fala descrito anteriormente.

Capítulo 4

Codificação de Canal Utilizando Conjunto de Códigos de Bloco

Este capítulo apresenta uma alternativa para o processo de codificação de canal contido na Recomendação IS-136 TDMA [IS-136.2].

Na IS-136 [IS-136.2], a codificação de canal utiliza um código convolucional. Nos processos alternativos, o código convolucional é substituído por um conjunto de códigos de Golay estendidos ou por uma combinação de um código gerado a partir de um código BCH e códigos de Golay. A motivação da substituição do código convolucional por código de Golay é devido a menor quantidade de bits de redundância (168 contra 178 porque não necessita de *tail* bits), ser mais adequado para decodificar porque não necessita truncar e porque pode-se usar 10 bits de redundância a mais para dar proteção aos bits perceptualmente mais significativos da saída do codificador de fala.

Os desempenhos dos processos alternativos, em termos de taxa de erro de bit pela relação sinal-ruído, são comparados com o código convolucional. A utilização de conjuntos de códigos de bloco pode fornecer proteção maior para os bits mais relevantes da saída do codificador de fala.

4.1 Conjunto de Códigos de Golay Estendidos Aplicados como GACs

A figura 4.1 apresenta a estrutura de proteção desigual contra erros de canal descrita na [IS-136.2] que contém um codificador com código convolucional de taxa $\frac{1}{2}$ com 5 memórias, $2^5 = 32$ estados, distância livre ($d_{free} = 8$) e uma capacidade de correção de $\left\lfloor \frac{d_{free}-1}{2} \right\rfloor = \left\lfloor \frac{8-1}{2} \right\rfloor = 3$ erros.

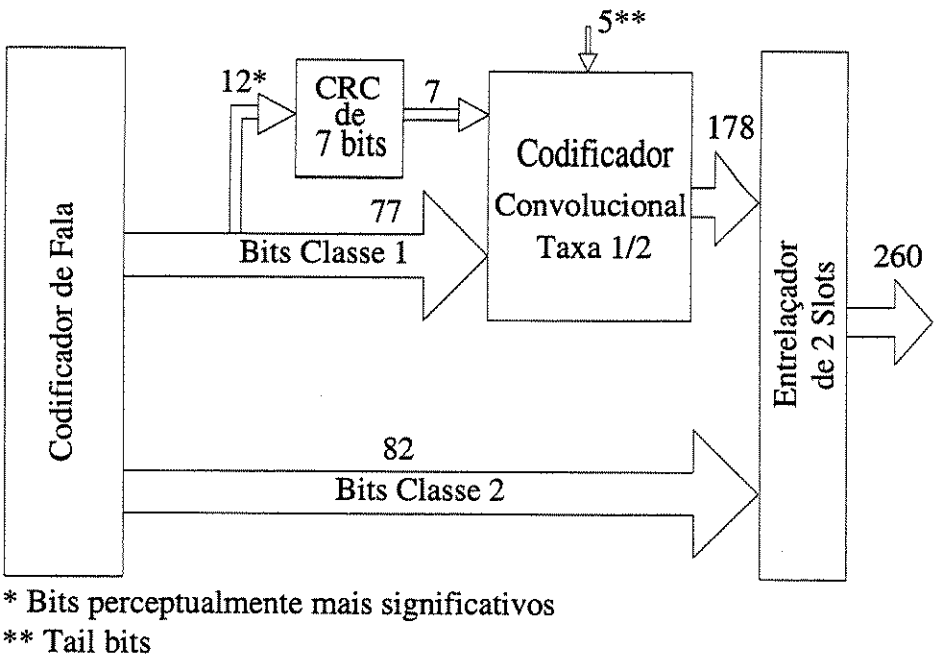

Figura 4.1: Diagrama em blocos da estrutura de proteção desigual contra erros no canal da *IS*-136.2.

A figura 4.2 apresenta a estrutura de proteção contra erros de canal contendo um conjunto de sete codificadores em paralelo com código de Golay (24,12,8) estendido. O conjunto de sete codificadores em paralelo com código de Golay (24,12,8) estendido é apresentado na figura 4.3.

Tanto para cada codificador com código de Golay (24,12,8) estendido como para o conjunto de sete codificadores, a taxa ($R = \frac{k}{n}$) é igual a $\frac{1}{2}$, que é a mesma do codificador convolucional [IS-136.2], o que garante uma comparação justa em termos de desempenho de sistema.

As curvas apresentadas são comparativas entre o atual sistema TDMA [IS-136.2] com

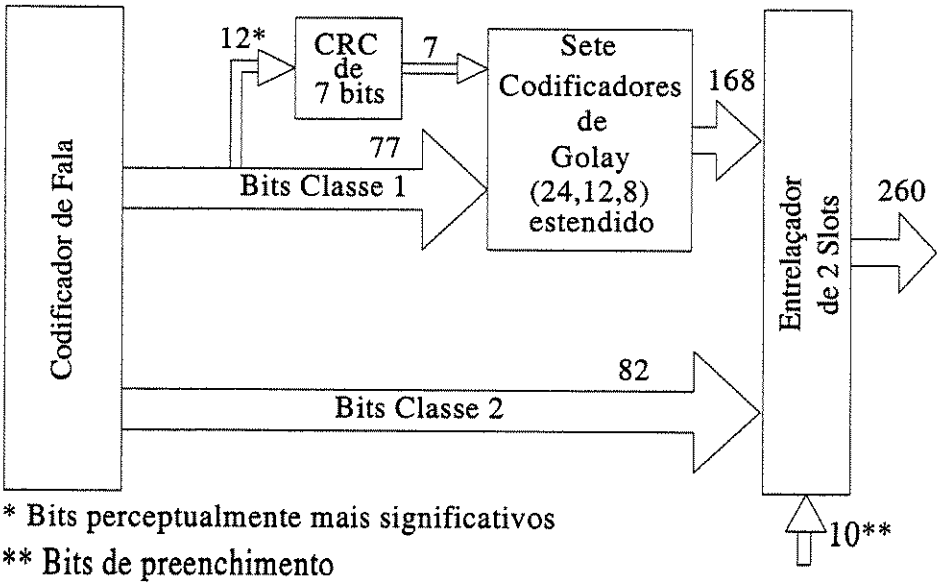

Figura 4.2: Diagrama em blocos modificado da estrutura de proteção desigual contra erros no canal.

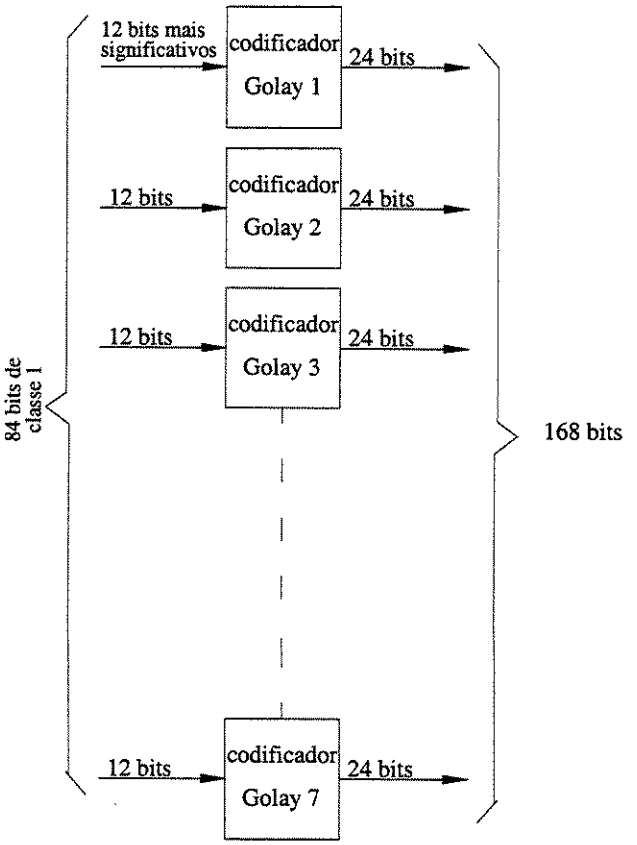

Figura 4.3: Conjunto de sete codificadores em paralelo com código de Golay (24,12,8) estendido.

o codificador usando o código convolucional e o modelo proposto com o conjunto de sete codificadores em paralelo usando o código de Golay (24,12,8) estendido, ambos em canal modelado como mostrado no diagrama em blocos da figura 4.4. A densidade espectral bilateral de potência de ruído $AWGN$ é $\frac{N_0}{2}$.

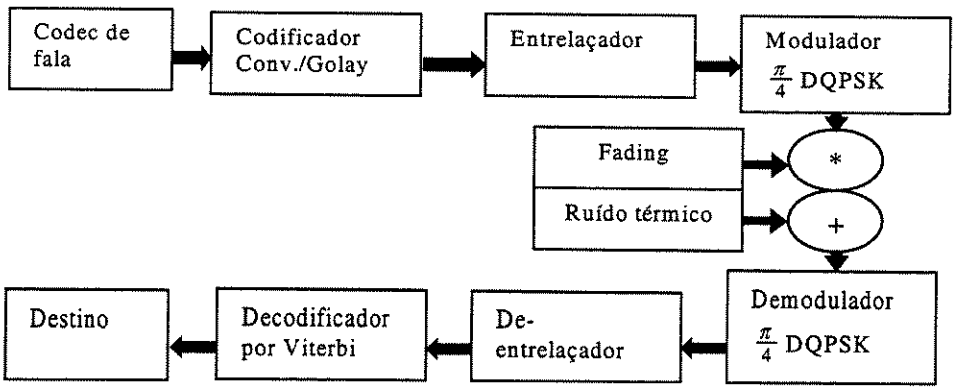

Figura 4.4: Diagrama em blocos para criação do programa de simulação.

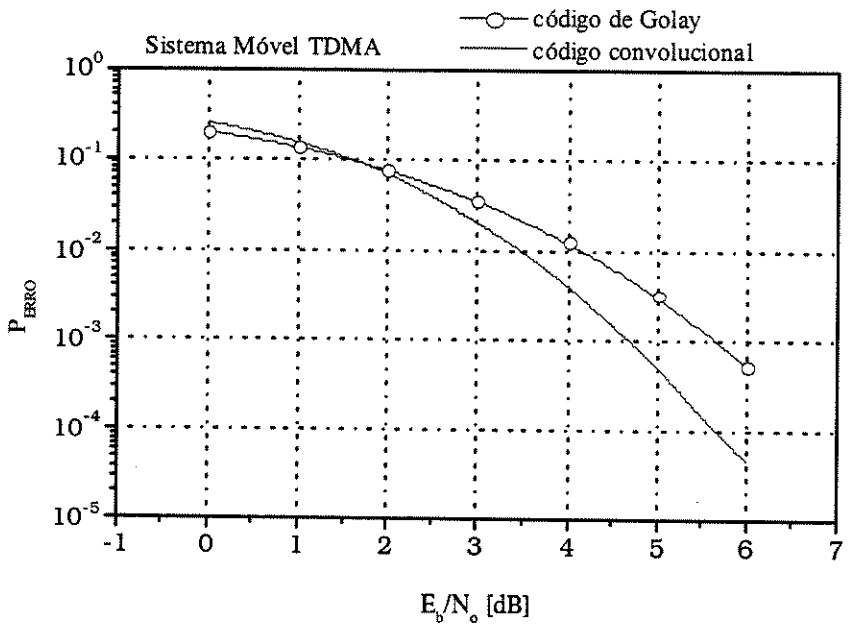

Figura 4.5: Curva de probabilidade de erro em canal AWGN com decisão abrupta.

A figura 4.5 apresenta os resultados obtidos com simulações, onde ambos sistemas foram

decodificados por Viterbi com decisão abrupta calculando a distância de Hamming mínima, $d_{\min}$, entre o vetor recebido R_p e o conjunto de vetores C_p que rotulam os ramos da treliça, como vistos no capítulo 2.

Observando a figura 4.5, verifica-se que o sistema que utiliza o conjunto de codificadores de Golay apresenta um desempenho 1.0 dB pior que o codificador convolucional para a probabilidade de erro, $P_{erro} = 10^{-3}$. Entretanto o codificador convolucional utiliza 10 bits de redundância a mais, devido aos *tail* bits, que ainda podem ser usados no modelo proposto para uma melhor proteção aos bits perceptualmente mais significativos da saída do codec.

A figura 4.6 apresenta os resultados obtidos com simulações, onde ambos sistemas foram decodificados por Viterbi com decisão suave calculando a mínima distância Euclidiana quadrática entre o sinal recebido Y_k e o conjunto de sinais S_k dos rótulos dos ramos da treliça.

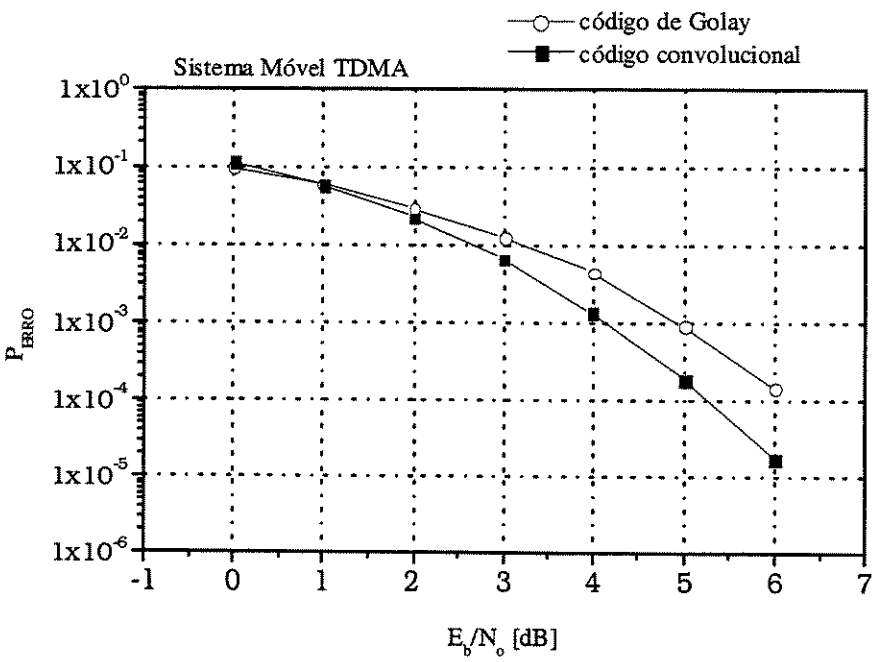

Figura 4.6: Curva de probabilidade de erro em canal AWGN com decisão suave.

A distância Euclidiana quadrática é calculada como

$$d^2 = \sum_{k=0}^{K} |Y_k - S_k|^2 \tag{4.1}$$

com K sinais de informação da seqüência de entrada.

Observando a figura 4.6, verifica-se que o sistema que utiliza o conjunto de codificadores de Golay apresenta um desempenho 1.0 dB pior que o codificador convolucional para $P_{erro} = 10^{-3}$. Entretanto, como já mencionado, o codificador convolucional utiliza 10 bits de redundância a mais, devido aos *tail* bits, que ainda podem ser usados no modelo proposto para uma melhor proteção aos bits perceptualmente mais significativos da saída do codec.

Analizando as figuras 4.5 e 4.6 observa-se a melhoria de desempenho quando o sistema é decodificado por decisão suave, em relação à decisão abrupta, comprovando-se assim os resultados simulados com valores reais, uma vez que o ganho assintótico em decodificação por decisão suave, na prática, é até 2 dB melhor que o ganho assintótico em decodificação por decisão abrupta.

As curvas apresentadas na figura 4.7 são comparativas entre o sistema TDMA [IS-136.2] atual com o codificador usando código convolucional e o modelo proposto com o banco de sete codificadores em paralelo usando o código de Golay (24,12,8) estendido, ambos em canal modelado como sendo $AWGN$ com densidade espectral bilateral de potência de ruído $\frac{N_0}{2}$ e com desvanecimento Rayleigh.

Observando a figura 4.7, verifica-se que na presença de desvanecimento, o conjunto de codificadores de Golay apresentou um desempenho 2.0 dB pior que o codificador convolucional para $P_{erro} = 10^{-3}$.

A diferença de desempenho entre o conjunto de codificadores com código de Golay (24,12,8) estendido e o codificador convolucional, nas curvas anteriores com e sem desvanecimento, é devida ao número de palavras código "vizinhas" com $d_{\min}$ serem maior no código de Golay (24,12,8) estendido que o número de palavras código "vizinhas" com d_{free} do código convolucional. O número de palavras código "vizinhas" com $d_{\min}$ no código de Golay (24,12,8)

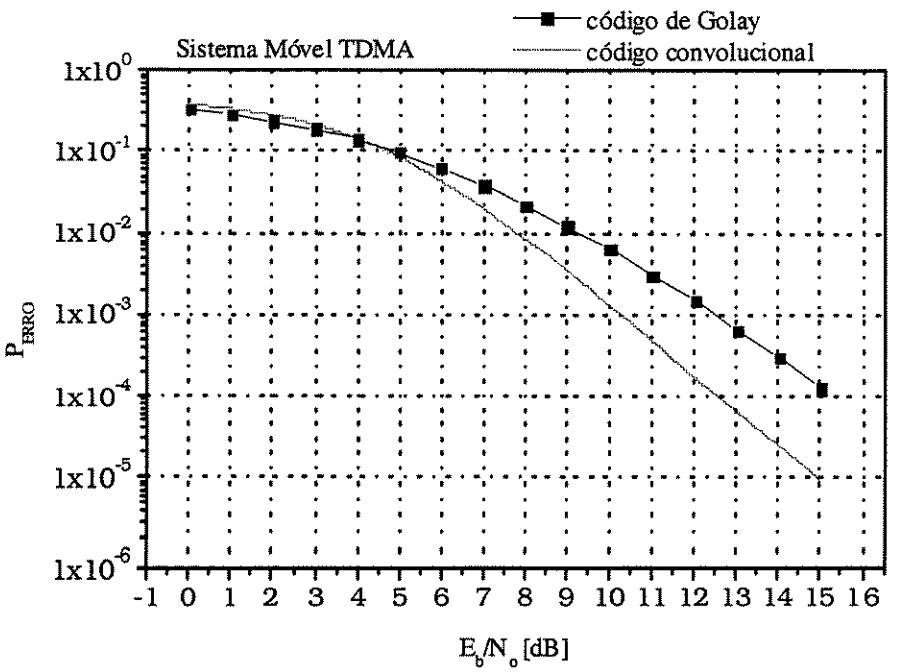

Figura 4.7: Curva de probabilidade de erro em canal AWGN com desvanecimento.

estendido podem ser vistas no diagrama de espectro de pesos da figura 2.5 e é igual a 759 palavras código com $d_{\min} = 8$, enquanto que o número de palavras código "vizinhas" do código convolucional é igual a 73 palavras código com $d_{free} = 8$.

O número de palavras código com distância igual a d_{free} de um código convolucional é chamado de Ad_{free} [Lin83] e é o número de caminhos que saem do estado inicial zero da treliça e retornam a outro estado zero com o valor da métrica acumulada igual a d_{free} do código convolucional. Como o código convolucional da IS-136.2 está sendo decodificado como código de bloco devido aos 5 *tail* bits que fazem a treliça convergir para o estado zero, levou-se agora em consideração não só os caminhos que partem do estado inicial zero, mas também todos os outros caminhos que partem dos outros estados zero da treliça que apresentam métrica acumulada igual a $d_{free} = 8$. O valor de Ad_{free} encontrado por simulação foi igual a 73 como mencionado anteriormente.

Como os números de palavras código "vizinhas" para os dois códigos em questão são

fatores multiplicativos na fórmula para o cálculo matemático da P_{erro}, explica-se o motivo pelo qual o codificador convolucional apresentou melhor desempenho que o conjunto de codificadores com código de Golay (24,12,8) estendido .

4.2 Esquema de Codificação com 6 Códigos de Golay e 1 BCH Puncionado

Nesta segunda etapa foi feita uma procura nos códigos conhecidos para encontrar aquele que pudesse apresentar uma maior potencialidade de correção em relação ao código de Golay. A partir da pesquisa nos códigos BCH primitivos binários em [Lin83], encontrou-se o código BCH (63,36,11). Seu polinômio gerador na forma octal é :

1 033 500 423

Expandindo este polinômio em binário, obtém-se

001 000 011 011 101 000 000 100 010 011

com o coeficiente de maior ordem sendo o da esquerda.

O polinômio gerador é então dado por :

$$g(x) = x^{27} + x^{22} + x^{21} + x^{19} + x^{18} + x^{17} + x^{15} + x^8 + x^4 + x + 1$$

A figura 4.8 apresenta a matriz (36 × 63) geradora do código BCH (63,36,11) na forma sistemática.

Utilizando a técnica de "puncionamento" de códigos descrita em [Clark81], reduziu-se o código BCH (63,36,11) em um código (39,12,11) eliminando as primeiras 24 linhas e as primeiras 24 colunas da matriz 4.8. Finalmente adicionando-se uma coluna de paridade par, obteve-se a matriz (12 × 40) geradora do novo código (40,12,12) mostrada na figura 4.9.

Códigos de Golay (23,12,7) são utilizados para complementar a codificação fazendo com que o número de bits codificados não ultrapasse os 178 bits da saída do codificador com código convolucional.

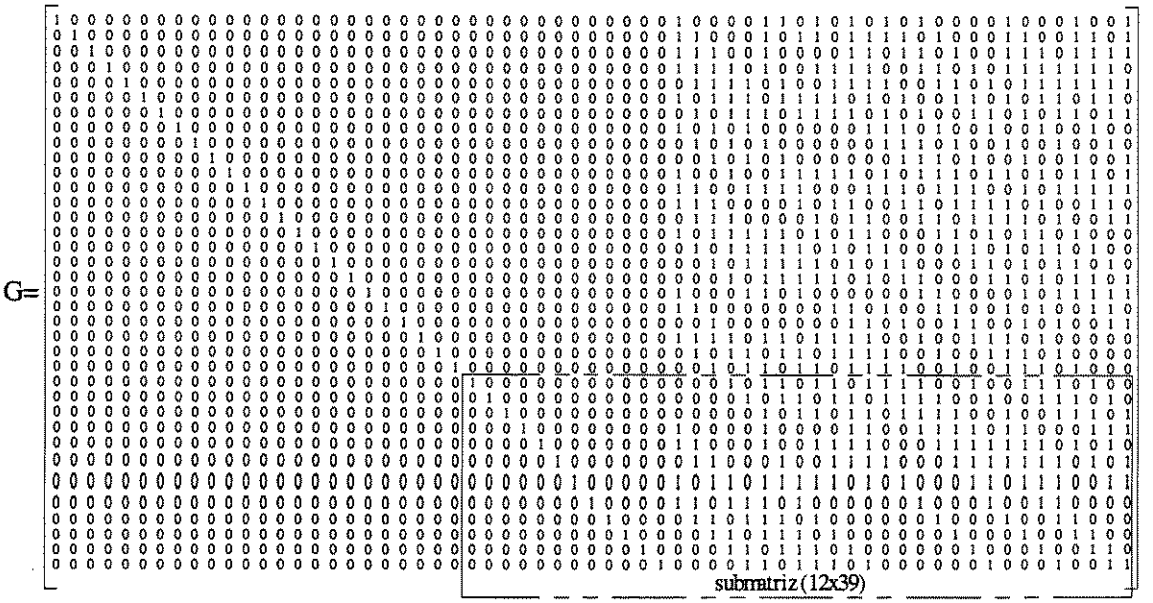

Figura 4.8: Geração do código (40,12,12) a partir da matriz geradora do código BCH (63,36,11).

$$G = \begin{bmatrix}
1000000000\ 0000010110\ 1101111001\ 0011101001 \\
0100000000\ 0000001011\ 0110111100\ 1001110101 \\
0010000000\ 0000000101\ 1011011110\ 0100111011 \\
0001000000\ 0010000100\ 0011001111\ 0110001110 \\
0000100000\ 0011000100\ 1111000111\ 1111010101 \\
0000010000\ 0001100010\ 0111100011\ 1111101011 \\
0000001000\ 0010110111\ 1101010001\ 1011100110 \\
0000000100\ 0011011101\ 0000001000\ 1001100001 \\
0000000010\ 0001101110\ 1000000100\ 0100110001 \\
0000000001\ 0000110111\ 0100000010\ 0010011001 \\
0000000000\ 1000011011\ 1010000001\ 0001001101 \\
0000000000\ 0100001101\ 1101000000\ 1000100111
\end{bmatrix}$$

Figura 4.9: Matriz geradora do código (40,12,12).

A figura 4.10 mostra o processo de codificação. A figura 4.11 mostra o espectro de distribuição de pesos do novo código (40,12,12) com 49 palavras código com $d_{\min} = 12$.

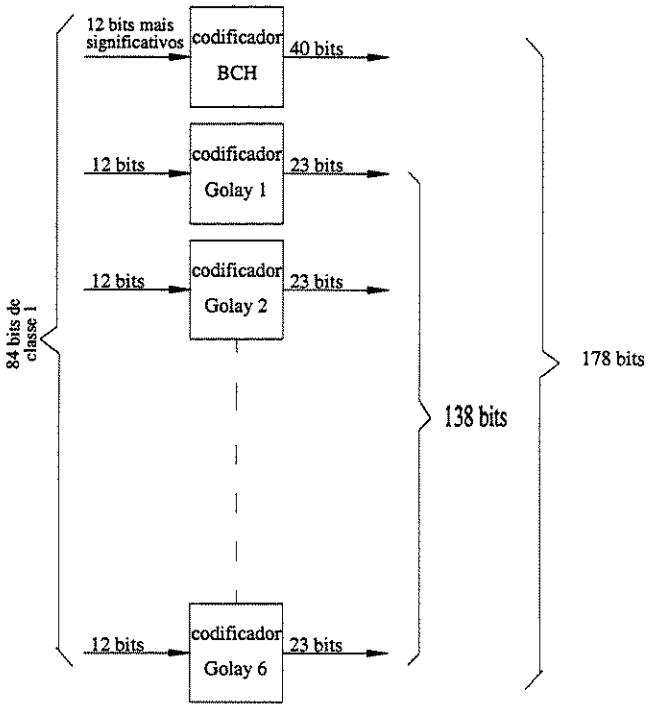

Figura 4.10: Diagrama em blocos do banco de seis codificadores com código de Golay (23,12,8) em paralelo com o codificador com o código (40,12,12).

A figura 4.12 apresenta o desempenho do novo banco com seis codificadores com código de Golay (23,12,7) em paralelo com o novo código (40,12,12) comparativamente com o banco de sete codificadores em paralelo com código de Golay (24,12,8) estendido anterior.

Verifica-se pela figura 4.12 que o modelo formado pelo conjunto de seis codificadores com código de Golay (23,12,7) em paralelo com o código (40,12,12) com taxa de codificação, $R_c = \frac{k}{n} = \frac{84}{178} = 0.472$, resulta em um desempenho 0,25 dB pior que o conjunto de sete codificadores com código de Golay (24,12,8) estendido com taxa de codificação, $R_c = 0.5$, devido a $d_{\min}$ do código de Golay (23,12,7) ser menor que a $d_{\min}$ do código de Golay (24,12,8) estendido.

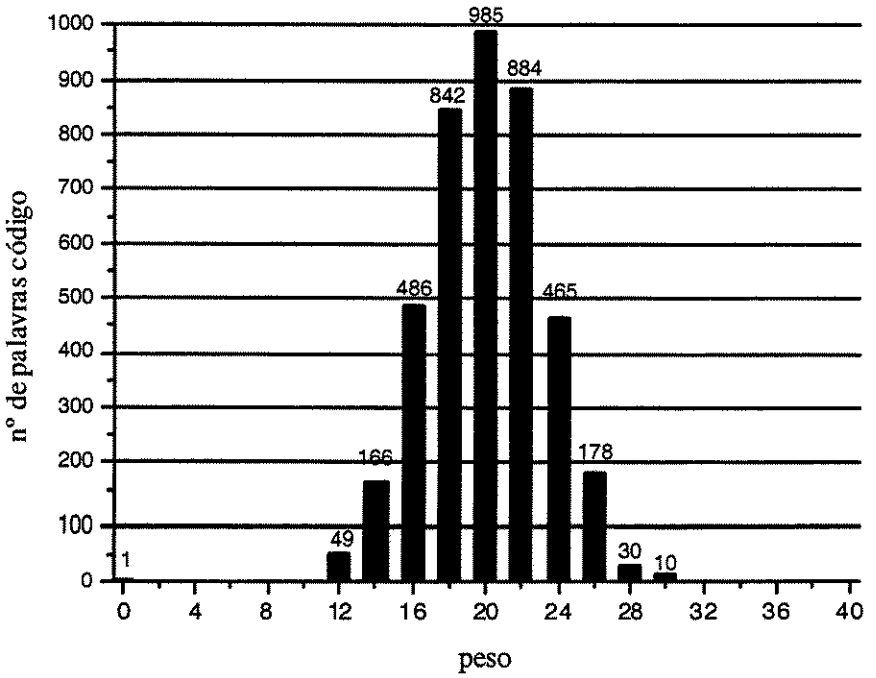

Figura 4.11: Espectro de distribuição de pesos do código (40,12,12).

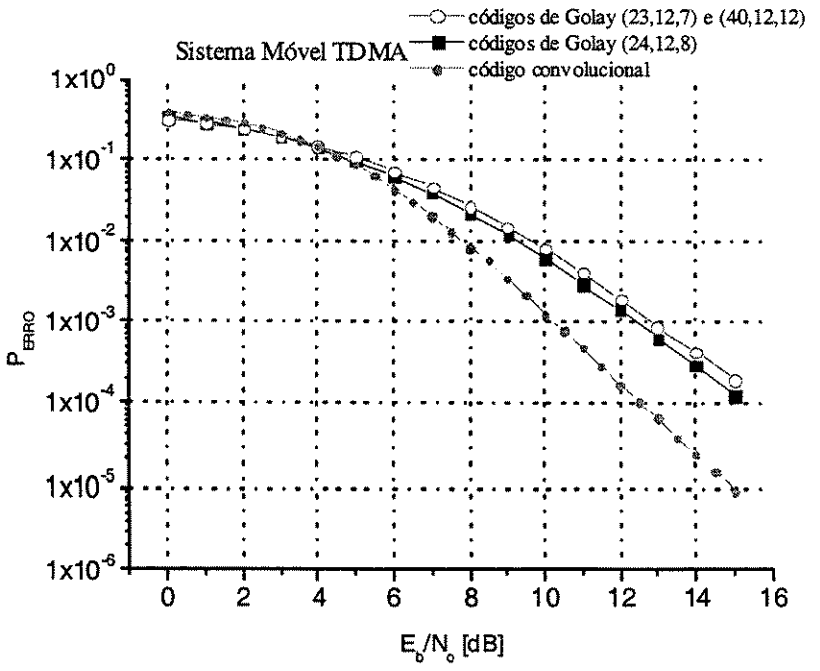

Figura 4.12: Curva de probabilidade de erro de bit em canal AWGN com desvanecimento.

Capítulo 5

Conclusões

A codificação de canal, descrita na Recomendação IS-136 TDMA [IS-136.2], faz uso de um codificador convolucional binário de taxa $\frac{1}{2}$ para proteger os bits de classe 1 provenientes do codificador de fala. Este trabalho apresentou um esquema alternativo de codificação de bloco que realizasse a mesma função do codificador convolucional com os seguintes objetivos :

Uma comparação dos esquemas, o atual com código convolucional com o proposto com códigos de Golay (24,12,8) estendido, em termos de taxa de erro de bit através de simulações em canal $AWGN$ com e sem desvanecimento Rayleigh. Em seguida foi feita uma busca de um código mais poderoso que pudesse oferecer maior proteção aos doze bits mais significativos do conjunto de bits de classe 1 da saída do codificador de fala.

A partir dos resultados das curvas apresentadas no capítulo 4, chegou-se às seguintes conclusões :

- O sistema proposto que utiliza um conjunto de sete codificadores em paralelo usando o código de Golay (24,12,8) estendido apresentou um desempenho 1.0 dB abaixo do sistema com codificador convolucional para $P_{erro} = 10^{-3}$ em canal $AWGN$ tanto com decisão abrupta quanto com decisão suave. Apesar do conjunto de codificadores com código de Golay ter capacidade de correção de (7 palavras código de 24 bits $\times$ 3 erros por palavra) 21 erros em 168 bits codificados de classe 1, o número de palavras código vizinhas mais próximas é muito elevado (759 palavras código). Isto explica o pior desempenho.

- O sistema proposto que utiliza um conjunto de sete codificadores em paralelo usando o código de Golay (24,12,8) estendido e o sistema que utiliza um conjunto de seis codificadores com código de Golay (23,12,7) em paralelo com um codificador com código (40,12,12) apresentaram um desempenho 2.0 dB abaixo do codificador convolucional para $P_{erro} = 10^{-3}$ em canal AWGN com desvanecimento, decodificados por decisão abrupta. Novamente o número de palavras código vizinhas mais próximas prevalece na obtenção da taxa de erro de bit. Entretanto, como o codificador convolucional possui cinco unidades de memória e utiliza cinco *tail* bits (todos zeros) para que o decodificador de Viterbi na recepção possa decodificar corretamente os últimos bits de informação, o que não ocorre com códigos de bloco por não necessitar de memória, resultou na sobra de dez bits que foram usados apenas como preenchimento na matriz de entrelaçamento no modelo proposto.

  A possibilidade de usar estes bits que "sobraram" não apenas como preenchimento e sim para proteção extra aos doze bits mais significativos do conjunto de bits de classe 1 da saída do codificador de fala, proporciona melhoria na qualidade do sinal de voz recebida pelo usuário e justifica a perda de desempenho dos sistemas propostos em relação ao atual com codificador convolucional da [IS-136.2].

- As curvas apresentadas na figura 4.12 na seção 4.2, comparativas com a curva do sistema TDMA, apresentaram uma diferença entre elas de 0,25 dB em função do parâmetro $\exp(-\frac{E_b}{N_0}.R_c.d_{\min})$ no cálculo matemático da P_{erro}. Para uma dada $\frac{E_b}{N_0}$, o parâmetro fica menor com código de Golay (24,12,8) estendido que com o código de Golay (23,12,7).

Ressalta-se que a decodificação do código (40,12,12) foi simulada a partir do algoritmo de Berlekamp-Massey apresentado em [Lin83] convertido em linguagem C++ por Robert Morelos-Zaragoza. Seu código fonte pode ser encontrado no site : http ://imailab-www.iis.u-tokyo.ac.jp/~robert/codes.html.

Este código foi adaptado e é parte integrante do programa completo para decodificação do sistema que utiliza o conjunto de seis codificadores com código de Golay (23,12,7) em

paralelo com o codificador com código (40,12,12).

5.1 Sugestões para Trabalhos Futuros

Análise de desempenho do codificador de canal proposto para o sistema móvel descrito na Recomendação IS-136 TDMA [IS-136.2] fazendo uso de outros algoritmos de decodificação (p.ex. : algoritmo de Chase).

Análise de desempenho do codificador de canal do sistema móvel descrito na recomendação IS-136.2 fazendo uso de *turbo codes* no lugar do código convolucional.

Uso de esquemas de modulação codificada.

Uso de esquemas de que utilizam códigos não binários (p.ex. : Reed-Solomon).

Apêndice A

Decodificação do Sistema Móvel TDMA

A.1 Codificador de Canal com Código de Golay

Será apresentado o código fonte desenvolvido para a simulação da decodificação do sistema móvel TDMA a partir do digrama em blocos da figura 4.4 no Capítulo 4.

```
#include <iostream.h>
#include <math.h>
#include <time.h>
#include <fstream.h>
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
#include <iomanip.h>
#include <math.h>
#include "matriz_4096x12.h"
#define Pi 3.14159265359
#define diretorio_saida "Golay_Gauss.txt"
struct inf{
double men[2];
}mensagem,r,Gauss,oldmen,rest,Ray;
struct Coder{
int M[24];unsigned int mcoded[4096][8];unsigned int m_coded[8];int estado;
}bit_code,output,rcod,mctest;
struct Block{
unsigned int CL[260];
}trans,recept;
struct Cl1{
unsigned int bit[168];
}Class1;
struct Decoder{
int Classe1[21][12];unsigned int Estatual;int Metrica;
}Treliss[4096],Orgn;
unsigned long Gera_Semente_fon(void);
void Gera_semente(void);
unsigned int fonte(void);
unsigned int Dmin(struct inf rec, struct inf old);
struct Coder Codificador(int []);
struct Block Interleaver(struct Cl1 C1);
struct inf Mapeamento(struct inf Old, unsigned int cod);
struct inf AWGN(double va);
struct Cl1 Dinterleav(struct Block S);
```

```
struct inf Rayleigh(double var);
void Trellis(int session, struct Coder mref);
int fpos,codigo,codest,ntr,pwindow,L,lower,low,z_inicial,z_final;
double pRay,dp,Limiar,SNR,Perro,erro;
unsigned int sem;
void main (){
int infobit[84];int bitent[12];int z,fpos,p;sem=Gera_Semente_fon();
Gera_semente();ofstream outfile(diretorio_saida,ios : :out);SNR=0;
pRay = (double) 1/sqrt(2);
do{dp=(double)sqrt(1.0/(2.0*pow(10.0,(double) SNR/10.0)));erro=0.0;ntr=0;
if(SNR<=3.0)Limiar=100000;else if(SNR<=5.0)Limiar = 10000;
else if(SNR<=6.0)Limiar = 5000;else if(SNR<=7.0)Limiar = 2000;
else if(SNR<=8.0)Limiar = 250;
do{
oldmen.men[0]=1.0;oldmen.men[1]=0.0;p=0;fpos=0;
for(int t=0; t<7; t++){for (int i=0; i<12; i++){
bitent[i]= fonte();infobit[p]=bitent[i];p++;}
bit_code=Codificador(bitent);
for (int w=0; w<24; w++){Class1.bit[fpos]=bit_code.M[w];fpos++;}}
trans = Interleaver(Class1);
for(int aux=0; aux<259; aux+=2){
codigo = trans.CL[aux]*2 + trans.CL[aux+1];mensagem = Mapeamento(oldmen,codigo);
Gauss = AWGN(dp);Ray = Rayleigh(pRay);
for(int w=0; w<2;w++){
3Ray.men[w]=1.0;r.men[w]=Ray.men[w]*mensagem.men[w]+Gauss.men[w];}
codest = Dmin(r,oldmen);recept.CL[aux+1]= codest&1;recept.CL[aux]=codest>>1;}
Class1 = Dinterleav(recept);pwindow=0;z=0;z_inicial=0;z_final=3;
for(int aux=0; aux<161; aux+=8){
rcod.m_coded[0]=Class1.bit[aux];rcod.m_coded[1]=Class1.bit[aux+1];
rcod.m_coded[2]=Class1.bit[aux+2];rcod.m_coded[3]=Class1.bit[aux+3];
rcod.m_coded[4]=Class1.bit[aux+4];rcod.m_coded[5]=Class1.bit[aux+5];
rcod.m_coded[6]=Class1.bit[aux+6];rcod.m_coded[7]=Class1.bit[aux+7];
Trellis(pwindow,rcod);pwindow++;
if((pwindow==3)||(pwindow==6)||(pwindow==9)||(pwindow==12)||
(pwindow==15)||(pwindow==18)||(pwindow==21)){
for(int posw=z_inicial; posw<z_final; posw++){
if((posw==0)||(posw==3)||(posw==6)||(posw==9)||
(posw==12)||(posw==15)||(posw==18)){for(int i=0; i<7; i++){
if(Treliss[lower].Classe1[posw][i] !=infobit[z])erro++;z++;}}
else if((posw==1)||(posw==4)||(posw==7)||(posw==10)
||(posw==13)||(posw==16)||(posw==19)){
for(int i=7; i<11; i++){
if(Treliss[lower].Classe1[posw][i] !=infobit[z])erro++;z++;}}
else if((posw==2)||(posw==5)||(posw==8)||(posw==11)
||(posw==14)||(posw==17)||(posw==20)){
for(int i=11; i<12; i++){
if(Treliss[lower].Classe1[posw][i] !=infobit[z])erro++;z++;}}}
z_inicial+=3;z_final+=3;}}ntr+=84;
}while(erro<Limiar);Perro =(double) (erro / ntr);
outfile <<SNR<<" "<<Perro<< endl;SNR+=1.0;
}while((Perro !=0)&&(SNR<6.5));outfile.close();}
void Gera_semente(void){time_t t;srand((unsigned) time(&t));}
unsigned long Gera_Semente_fon(void){
unsigned long semente;srand((unsigned) time(NULL));
semente=abs(random(10000))+524288;return(semente);}
unsigned int fonte(void){
register long int masc1,masc2,saida;masc1=sem&0x1000000;masc2=sem&0x0000001;
saida=((masc1>>6)^masc2)&1;sem=sem>>1;
if (saida==1)sem=sem|0x10000000;return(saida);}
struct Coder Codificador(int b[]){
int sym,p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p44,p88;
int x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12;
int c1,c2,c3,c4,c5,j,k,q;
x1=b[0];x2=b[1];x3=b[2];x4=b[3];p1=x1^x4;p2=x2^x4;
```

```
p3=x3^x4;p4=x1^x2^x3;p44=p4^x4;
x5=b[4];x6=b[5];x7=b[6];x8=b[7];p5=x5^x8;p6=x6^x8;
p7=x7^x8;p8=x5^x6^x7;p88=p8^x8;
p9=x1^x5;p10=p1^p5;p11=x2^x6;p12=p2^p6;p13=x3^x7;
p14=p3^p7;p15=p4^p8;p16=p44^p88;
x9=b[8]; x10=b[9]; x11=b[10]; x12=b[11];sym = x11 + 2*x10 + 4*x9;
if (sym==0){c1=0;c2=0;c3=0;c4=0;c5=0;}
else if (sym==1){c1=1;c2=0;c3=0;c4=1;c5=1;}
else if (sym==2){c1=1;c2=0;c3=1;c4=1;c5=0;}
else if (sym==3){c1=1;c2=1;c3=0;c4=0;c5=0;}
else if (sym==4){c1=1;c2=1;c3=0;c4=1;c5=0;}
else if (sym==5){c1=1;c2=0;c3=1;c4=0;c5=0;}
else if (sym==6){c1=1;c2=0;c3=0;c4=0;c5=1;}
else if (sym==7){c1=0;c2=0;c3=0;c4=1;c5=0;}
int C[3][8]=
{{x1^c1,p1^x9,x2^c2,p2^x10,x3^c3,p3^c4,p4^x11,p44^c5},
{x5^c1,p5^x9,x6^c2,p6^x10,x7^c3,p7^c4,p8^x11,p88^c5},
{p9^c1^x12,p10^x9^x12,p11^c2^x12,p12^x10^x12,p13^c3^x12,p14^c4^x12,
p15^x11^x12,p16^c5^x12}};q=0;
for(j=0; j<3; j++){
for(k=0; k<8; k++){output.M[q]=C[j][k];q++;}}
return(output);}
struct Block Interleaver(struct Cll C1){
unsigned int Matriz[26][10];struct Block Transmit;int row,colm,q3,pos;q3=0;pos=0;
for(colm=0;colm<10;colm++)
for(row=0;row<26;row++){
if((pos%26==0)||(pos>=93)&&(pos<=129)||(pos>=223)&&(pos<=259)||
(pos>=1)&&(pos<=5)||(pos>=131)&&(pos<=135))
Matriz[row][colm]=0;
else{Matriz[row][colm]=C1.bit[q3];q3++;}pos++;}q3=0;
for(row=0;row<26;row++)
for(colm=0;colm<10;colm++){Transmit.CL[q3]=Matriz[row][colm];q3++;}
return(Transmit);}
struct inf AWGN(double va){
long double U,V,X;struct inf sai;int fd;
do{U=(double)rand()/RAND_MAX;}while(U==1);
V=(double)(rand())/RAND_MAX;X=va*(sqrt(-2*(log(1-U))));
for(fd=0; fd<2; fd++){
if (fd==0)sai.men[0]=(double)X*cos(2*Pi*V);
else sai.men[1]=(double)X*sin(2*Pi*V);}return(sai);}
struct inf Mapeamento(struct inf Old, unsigned int cod){
struct inf Map;double Fase;
if(cod%2==0){cod=(cod>>1)&1;
if(cod==1)Fase = -Pi/4;
else Fase = Pi/4;}
else{cod=(cod>>1)&1;
if(cod==1)Fase = -3*Pi/4;
else Fase = 3*Pi/4;}
Map.men[0] = Old.men[0]*cos(Fase) - Old.men[1]*sin(Fase);
Map.men[1] = Old.men[1]*cos(Fase) + Old.men[0]*sin(Fase);
return(Map);}
unsigned int Dmin(struct inf rec, struct inf old){
double D,Dm;unsigned int cd,cdaux;int Di;struct inf E,Est;Dm = 0.0;cdaux = 0;
Est = Mapeamento(old,cdaux);
for(Di=0;Di<2;Di++)
Dm = pow(Est.men[Di]-rec.men[Di],2.0) + Dm;
for(cd=1;cd<4;cd++){D = 0.0;E = Mapeamento(old,cd);
for(Di=0;Di<2;Di++)D = pow(E.men[Di]-rec.men[Di],2.0) + D;
if(D<Dm){Dm = D;Est = E;cdaux = cd;}}oldmen = Est;return(cdaux);}
struct Cll Dinterleav(struct Block S){unsigned int Mat[26][10];
struct Cll Received;int row,colm,q3,pos;q3=0;
for(row=0;row<26;row++)
for(colm=0;colm<10;colm++){Mat[row][colm] = S.CL[q3];q3++;}q3=0;pos = 0;
for(colm=0;colm<10;colm++)
```

```
for(row=0;row<26;row++){
if(!((pos%26==0)||(pos>=93)&&(pos<=129)||(pos>=223)&&(pos<=259)||
(pos>=1)&&(pos<=5)||(pos>=131)&&(pos<=135))){
Received.bit[q3] = Mat[row][colm];q3++;}pos++;}return(Received);}
struct inf Rayleigh(double var){struct inf Rayl;long double uni;int Rcont;
for(Rcont=0;Rcont<2;Rcont++){
do{uni=(double)rand()/RAND_MAX;}while(uni==1);
Rayl.men[Rcont] =(double) (var*sqrt(-2*log(1-uni)));}return(Rayl);}
void Trellis(int session, struct Coder mref){
int x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12;
int sym,sim,p1,p2,p3,p4,p44,p5,p6,p7,p8,p88,p9,p10,p11,p12,p13,p14,p15,p16;
int c1,c2,c3,c4,c5;
if((session==0)||(session==3)||(session==6)||(session==9)||(session==12)
||(session==15)||(session==18)){int depti,best,vari;best=0;vari=0;depti=0;
for(int i=0; i<4096; i++){Treliss[i].Estatual=0;Treliss[i].Metrica=0;
if(depti<32)best=vari+0;
else if(depti<64)best=vari+8;
else if(depti<96)best=vari+16;
else if(depti<128)best=vari+24;
else if(depti<160)best=vari+32;
else if(depti<192)best=vari+40;
else if(depti<224)best=vari+48;
else if(depti<256)best=vari+56;
else if(depti<288)best=vari+64;
else if(depti<320)best=vari+72;
else if(depti<352)best=vari+80;
else if(depti<384)best=vari+88;
else if(depti<416)best=vari+96;
else if(depti<448)best=vari+104;
else if(depti<480)best=vari+112;
else if(depti<512)best=vari+120;
mctest.estado=Treliss[i].Estatual;
x1=bj[i][0]; x2=bj[i][1]; x3=bj[i][2]; x4=bj[i][3];
x9=bj[i][8]; x10=bj[i][9]; x11=bj[i][10];
p1=x1^x4; p2=x2^x4; p3=x3^x4; p4=x1^x2^x3; p44=x4^p4;
sym=x11+ 2*x10+ 4*x9;
if (sym==0){c1=0;c2=0;c3=0;c4=0;c5=0;}
else if (sym==1){c1=1;c2=0;c3=0;c4=1;c5=1;}
else if (sym==2){c1=1;c2=0;c3=1;c4=1;c5=0;}
else if (sym==3){c1=1;c2=1;c3=0;c4=0;c5=0;}
else if (sym==4){c1=1;c2=1;c3=0;c4=1;c5=0;}
else if (sym==5){c1=1;c2=0;c3=1;c4=0;c5=0;}
else if (sym==6){c1=1;c2=0;c3=0;c4=0;c5=1;}
else if (sym==7){c1=0;c2=0;c3=0;c4=1;c5=0;}
mctest.mcoded[i][0]=x1^c1;mctest.mcoded[i][1]=p1^x9;mctest.mcoded[i][2]=x2^c2;
mctest.mcoded[i][3]=p2^x10;mctest.mcoded[i][4]=x3^c3;mctest.mcoded[i][5]=p3^c4;
mctest.mcoded[i][6]=p4^x11;mctest.mcoded[i][7]=p44^c5;
output.estado=mctest.estado ^best;Treliss[i].Estatual=output.estado;
if(mctest.mcoded[i][0] != mref.m_coded[0])Treliss[i].Metrica++;
if(mctest.mcoded[i][1] != mref.m_coded[1])Treliss[i].Metrica++;
if(mctest.mcoded[i][2] != mref.m_coded[2])Treliss[i].Metrica++;
if(mctest.mcoded[i][3] != mref.m_coded[3])Treliss[i].Metrica++;
if(mctest.mcoded[i][4] != mref.m_coded[4])Treliss[i].Metrica++;
if(mctest.mcoded[i][5] != mref.m_coded[5])Treliss[i].Metrica++;
if(mctest.mcoded[i][6] != mref.m_coded[6])Treliss[i].Metrica++;
if(mctest.mcoded[i][7] != mref.m_coded[7])Treliss[i].Metrica++;
for(int j=0; j<7; j++){Treliss[i].Classe1[session][j]=0;
Treliss[i].Classe1[session][j]=bj[i][j];}depti++;
if(depti==512){vari++;depti=0;}}}
else if((session==1)||(session==4)||(session==7)||(session==10)||
(session==13)||(session==16)||(session==19)){
int bite[16][4]={{0,0,0,0},{0,0,0,1},{0,0,1,0},{0,0,1,1},{0,1,0,0},{0,1,0,1},
{0,1,1,0},{0,1,1,1},{1,0,0,0},{1,0,0,1},{1,0,1,0},{1,0,1,1},{1,1,0,0},{1,1,0,1},
{1,1,1,0},{1,1,1,1}};
```

```
int a,b,c,sym[16],bitent,dept ;dept=0 ;a=0 ;b=1 ;c=0 ;
for(int j=0 ; j<16 ; j++)
sym[j]=bite[j][3]+ 2*bite[j][2] + 4*bite[j][1] + 8*bite[j][0] ;
for(int i=0 ; i<4096 ; i++){
if((c/2)==b){a++ ;b++ ;}c++ ;
if(dept<32)bitent=sym[a]<<3 ;
else if(dept<64)bitent=(sym[a]^1)<<3 ;
else if(dept<96)bitent=(sym[a]^2)<<3 ;
else if(dept<128)bitent=(sym[a]^3)<<3 ;
else if(dept<160)bitent=(sym[a]^4)<<3 ;
else if(dept<192)bitent=(sym[a]^5)<<3 ;
else if(dept<224)bitent=(sym[a]^6)<<3 ;
else if(dept<256)bitent=(sym[a]^7)<<3 ;
else if(dept<288)bitent=(sym[a]^8)<<3 ;
else if(dept<320)bitent=(sym[a]^9)<<3 ;
else if(dept<352)bitent=(sym[a]^10)<<3 ;
else if(dept<384)bitent=(sym[a]^11)<<3 ;
else if(dept<416)bitent=(sym[a]^12)<<3 ;
else if(dept<448)bitent=(sym[a]^13)<<3 ;
else if(dept<480)bitent=(sym[a]^14)<<3 ;
else if(dept<512)mctest.estado=Treliss[i].Estatual ;
x5=bj[i][4] ; x6=bj[i][5] ; x7=bj[i][6] ;x8=bj[i][7] ;x9=bj[i][8] ; x10=bj[i][9] ;
x11=bj[i][10] ;
p5=x5^x8 ; p6=x6^x8 ; p7=x7^x8 ; p8=x5^x6^x7 ; p88=x8^p8 ;sim=x11+ 2*x10+ 4*x9 ;
if (sim==0){c1=0 ;c2=0 ;c3=0 ;c4=0 ;c5=0 ;}
else if (sim==1){c1=1 ;c2=0 ;c3=0 ;c4=1 ;c5=1 ;}
else if (sim==2){c1=1 ;c2=0 ;c3=1 ;c4=1 ;c5=0 ;}
else if (sim==3){c1=1 ;c2=1 ;c3=0 ;c4=0 ;c5=0 ;}
else if (sim==4){c1=1 ;c2=1 ;c3=0 ;c4=1 ;c5=0 ;}
else if (sim==5){c1=1 ;c2=0 ;c3=1 ;c4=0 ;c5=0 ;}
else if (sim==6){c1=1 ;c2=0 ;c3=0 ;c4=0 ;c5=1 ;}
else if (sim==7){c1=0 ;c2=0 ;c3=0 ;c4=1 ;c5=0 ;}
mctest.mcoded[i][0]=x5^c1 ;mctest.mcoded[i][1]=p5^x9 ;mctest.mcoded[i][2]=x6^c2 ;
mctest.mcoded[i][3]=p6^x10 ;mctest.mcoded[i][4]=x7^c3 ;mctest.mcoded[i][5]=p7^c4 ;
mctest.mcoded[i][6]=p8^x11 ;mctest.mcoded[i][7]=p88^c5 ;
output.estado=mctest.estado ^bitent ;
Treliss[i].Estatual=output.estado ;
if(mctest.mcoded[i][0] != mref.m_coded[0])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][1] != mref.m_coded[1])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][2] != mref.m_coded[2])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][3] != mref.m_coded[3])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][4] != mref.m_coded[4])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][5] != mref.m_coded[5])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][6] != mref.m_coded[6])Treliss[i].Metrica++ ;
if(mctest.mcoded[i][7] != mref.m_coded[7])Treliss[i].Metrica++ ;
for(int j=7 ; j<11 ; j++){Treliss[i].Classe1[session][j]=0 ;
Treliss[i].Classe1[session][j]=bj[i][j] ;}dept++ ;
if(dept==512)dept=0 ;
if((dept==0)||(dept==32)||(dept==64)||(dept==96)||(dept==128)||
(dept==160)||(dept==192)||(dept==224)||(dept==256)||(dept==288)||(dept==320)||
(dept==352)||(dept==384)||(dept==416)||(dept==448)||(dept==480)){
a=0 ;b=1 ;c=0 ;}}}
else if((session==2)||(session==5)||(session==8)||(session==11)||
(session==14)||(session==17)||(session==20)){
int start=0 ;int end=2 ;int inicio=0 ;int fim=2 ;int vi=0 ;int vf=2 ;
for(int k=0 ; k<8 ; k++){
for(int j=0 ; j<256 ; j++){
for(int i=vi ; i<vf ; i++){
mctest.estado=Treliss[i].Estatual ;
x1=bj[i][0] ; x2=bj[i][1] ; x3=bj[i][2] ; x4=bj[i][3] ;
x5=bj[i][4] ; x6=bj[i][5] ; x7=bj[i][6] ; x8=bj[i][7] ;
x9=bj[i][8] ; x10=bj[i][9] ; x11=bj[i][10] ; x12=bj[i][11] ;
p1=x1^x4 ; p2=x2^x4 ; p3=x3^x4 ; p4=x1^x2^x3 ; p44=x4^p4 ;
p5=x5^x8 ; p6=x6^x8 ; p7=x7^x8 ; p8=x5^x6^x7 ; p88=x8^p8 ;
```

```
p9=x1^x5; p10=p1^p5; p11=x2^x6; p12=p2^p6;
p13=x3^x7; p14=p3^p7; p15=p4^p8; p16=p44^p88;
sym=x11+ 2*x10+ 4*x9;
if (sym==0){c1=0;c2=0;c3=0;c4=0;c5=0;}
else if (sym==1){c1=1;c2=0;c3=0;c4=1;c5=1;}
else if (sym==2){c1=1;c2=0;c3=1;c4=1;c5=0;}
else if (sym==3){c1=1;c2=1;c3=0;c4=0;c5=0;}
else if (sym==4){c1=1;c2=1;c3=0;c4=1;c5=0;}
else if (sym==5){c1=1;c2=0;c3=1;c4=0;c5=0;}
else if (sym==6){c1=1;c2=0;c3=0;c4=0;c5=1;}
else if (sym==7){c1=0;c2=0;c3=0;c4=1;c5=0;}
mctest.mcoded[i][0]=x12^p9^c1;mctest.mcoded[i][1]=x12^p10^x9;
mctest.mcoded[i][2]=x12^p11^c2;mctest.mcoded[i][3]=x12^p12^x10;
mctest.mcoded[i][4]=x12^p13^c3;mctest.mcoded[i][5]=x12^p14^c4;
mctest.mcoded[i][6]=x12^p15^x11;mctest.mcoded[i][7]=x12^p16^c5;
output.estado=mctest.estado ^mctest.estado;Treliss[i].Estatual=output.estado;
if(mctest.mcoded[i][0] != mref.m_coded[0])Treliss[i].Metrica++;
if(mctest.mcoded[i][1] != mref.m_coded[1])Treliss[i].Metrica++;
if(mctest.mcoded[i][2] != mref.m_coded[2])Treliss[i].Metrica++;
if(mctest.mcoded[i][3] != mref.m_coded[3])Treliss[i].Metrica++;
if(mctest.mcoded[i][4] != mref.m_coded[4])Treliss[i].Metrica++;
if(mctest.mcoded[i][5] != mref.m_coded[5])Treliss[i].Metrica++;
if(mctest.mcoded[i][6] != mref.m_coded[6])Treliss[i].Metrica++;
if(mctest.mcoded[i][7] != mref.m_coded[7])Treliss[i].Metrica++;
for(int j=11; j<12; j++){Treliss[i].Classe1[session][j]=0;
Treliss[i].Classe1[session][j]=bj[i][11];}}vi+=32;vf+=32;
if(vi>(inicio+480)){inicio+=2;fim+=2;vi=inicio;vf=fim;}}
start+=512;end+=512;inicio=start;fim=end;vi=inicio;vf=fim;}lower=0;
for(int x=1; x<4096; x++)
if(Treliss[x].Metrica < Treliss[lower].Metrica)lower=x;}}
```

A.2 Geração das Variáveis Aleatórias Rayleigh e Gaussiana a partir da Variável Aleatória Uniforme

A função de densidade de probabilidade, pdf (*probability density function*) de uma variável aleatória Rayleigh é dada por :

$$p(r) = \frac{r}{\sigma^2} \exp(-\frac{r^2}{2\sigma^2}), r \geq 0 \tag{A.1}$$

Sua função de distribuição de probabilidade, fdp (*probability distribution function*) é dada por :

$$P(R) = prob(r \leq R) = \int_0^R p(r)dr = \int_0^R \frac{r}{\sigma^2} \exp(-\frac{r^2}{2\sigma^2})dr \tag{A.2}$$

Aplicando o método de substituição da variável e fazendo $u = -\frac{r^2}{2\sigma^2}$, implica que $\frac{du}{dr} = -\frac{2r}{2\sigma^2} = -\frac{r}{\sigma^2}$ e portanto $dr = -\frac{\sigma^2 du}{r}$. Os novos limites da integral são : se $r = 0 \Rightarrow u = -\frac{0^2}{2\sigma^2} = 0$ e se $r = R \Rightarrow u = -\frac{R^2}{2\sigma^2}$. Substituindo dr e os limites inferior e superior da integral

em A.2, tem-se :

$$P(R) = \int_0^{-\frac{R^2}{2\sigma^2}} \frac{r}{\sigma^2} \exp(u)(-\frac{\sigma^2 du}{r}) = \int_0^{-\frac{R^2}{2\sigma^2}} -\exp(u)du = 1 - \exp(-\frac{R^2}{2\sigma^2}) \quad (A.3)$$

$$1 - P(R) = \exp(-\frac{R^2}{2\sigma^2}) \quad (A.4)$$

Aplicando ln nos dois membros da expressão A.4, tem-se :

$$\ln[1 - P(R)] = -\frac{R^2}{2\sigma^2} \quad (A.5)$$

Aplicando o método da inversão de domínio, visto em [Papoulis91], no eixo y do gráfico A.1 da função de distribuição de probabilidade da variável aletória Rayleigh vê-se uma distribuição uniforme entre 0 e 1. Assim pode-se chamar $P(R)$ de U, onde U é uma variável aleatória Uniforme. Aplicando esta definição em A.5 tem-se :

$$\ln(1 - U) = -\frac{R^2}{2\sigma^2} \Rightarrow R^2 = -2\sigma^2.\ln(1 - U) \Rightarrow R = \sqrt{-2\sigma^2.\ln(1 - U)} \quad (A.6)$$

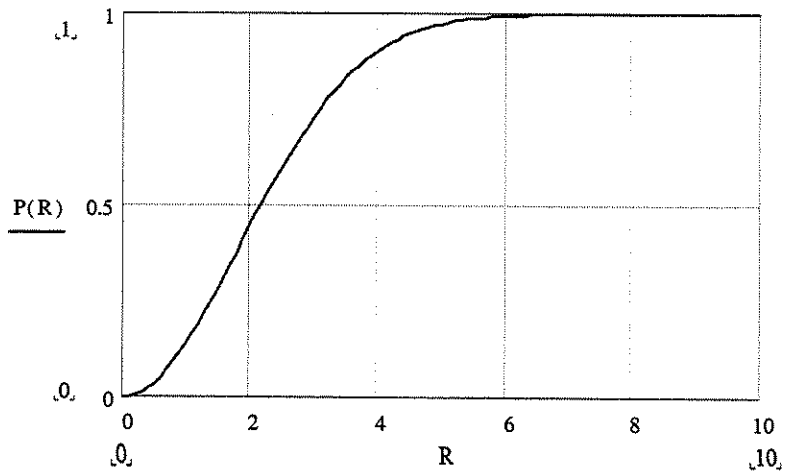

Figura A.1: Função de distribuição de probabilidade da variável aleatória Rayleigh.

E finalmente tem-se a variável aletória Rayleigh em função da variável aletória Uniforme :

$$R = \sigma\sqrt{-2.\ln(1 - U)} \quad (A.7)$$

De [Pro95]

$$R^2 = G_x^2 + G_y^2 \tag{A.8}$$

que é obtida pelo Teorema de Pitágoras para Triângulos Retângulos como pode ser visto na figura A.2, onde G_x é a variável aleatória Gaussiana em fase e G_y é a variável aleatória Gaussiana em quadratura, uma vez que a modulação do sistema TDMA é a $\frac{\pi}{4}DQPSK$.

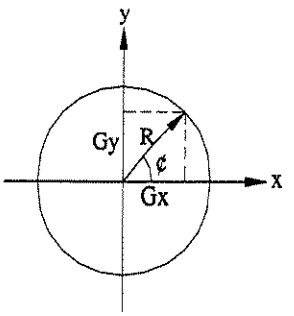

Figura A.2: Representação das variáveis aleatórias Rayleigh e Gaussiana.

E portanto, pelo Teorema de Pitágoras

$$G_x = R\cos(2\pi.U_1) \tag{A.9}$$

$$G_y = R\sin(2\pi.U_2) \tag{A.10}$$

onde U_1 e U_2 são variáveis aleatórias Uniformes geradas em instantes diferentes.

A.3 Ajuste da Variância do Ruído Gaussiano

Para uma modulação usando n bits por símbolo de modulação :

$$\frac{Es}{N_0} = \frac{n.Eb}{N_0} \tag{A.11}$$

Para um sistema com modulação usando n bits por símbolo de modulação e codificado a uma taxa Rc :

$$\frac{Es}{N_0} = n.\frac{Eb}{N_0}.Rc \tag{A.12}$$

No canal AWGN, o sinal é corrompido pelo ruído aditivo $n(t)$, que tem densidade espectral bilateral de potência $\frac{N_0}{2}[\frac{Watts}{Hz}]$. A variância σ^2 deste ruído é igual a $\frac{N_0}{2}$. Normalizando a energia por símbolo $Es = 1$, então $\frac{Es}{N_0} = \frac{1}{2\sigma^2}$. Substituindo este valor em A.12 :

$$\frac{1}{2\sigma^2} = n.\frac{Eb}{N_0}.Rc \tag{A.13}$$

Portanto :

$$2\sigma^2 = \frac{1}{n.\frac{Eb}{N_0}.Rc} \Rightarrow \sigma^2 = \frac{1}{2.n.\frac{Eb}{N_0}.Rc} \Rightarrow \sigma = \frac{1}{\sqrt{2.n.\frac{Eb}{N_0}.Rc}} \tag{A.14}$$

Para poder atribuir valores a $\frac{Eb}{N_0}$ em dB :

$$\sigma = \frac{1}{\sqrt{\frac{1}{2.n.Rc.10^{(\frac{\frac{Eb}{N_0}}{10})}}}} \tag{A.15}$$

A.4 SBrT 2000

A seguir é apresentado o artigo selecionado para apresentação no XVIII Simpósio Brasileiro de Telecomunicações, 3 a 6 de Setembro de 2000, Gramado-RS, Brasil.

As figuras citadas se encontram no corpo desta dissertação.

CODIFICAÇÃO DE CANAL ALTERNATIVA PARA O SISTEMA MÓVEL TDMA

Carlos Henrique R. Oliveira e Renato Baldini Filho

Departamento de Comunicações - FEEC - Unicamp

Caixa Postal 6101 - CEP 13083-970 - Campinas - SP

carloshe@decom.fee.unicamp.br, baldini@decom.fee.unicamp.br

RESUMO

A codificação de canal, descrita na Recomendação TDMA IS-136, faz uso de um codificador convolucional binário de taxa $\frac{1}{2}$ para proteger os bits de Classe 1 provenientes do

codificador de fala. Este trabalho analisa um esquema alternativo de codificação de bloco que realiza a mesma função do codificador convolucional.

1. INTRODUÇÃO

O controle de erro de canal para os dados do codec de fala visto na figura 3.13, emprega 3 mecanismos para combater os erros de canal. O primeiro utiliza um código convolucional de taxa $\frac{1}{2}$ para proteger os bits mais vulneráveis na seqüência de dados do codificador de fala (codec). O segundo entrelaça os dados transmitidos para cada quadro do codec sobre dois slots de tempo para mitigar os efeitos do desvanecimento Rayleigh. O terceiro emprega o uso de uma verificação de redundância cíclica sobre os bits perceptualmente mais significativos da saída do codec. No receptor, estes bits de redundância cíclica são utilizados para verificar se os bits perceptualmente mais significativos foram recebidos apropriadamente. O código convolucional é substituído por um conjunto de códigos de bloco com o intuito de oferecer uma proteção adicional aos bits perceptualmente mais significativos responsáveis pela qualidade do sinal de voz recebida pelo usuário.

2. CODIFICAÇÃO DE CANAL

2.1 Classes de Dados de Fala

O processo de correção de erro separa a informação de quadro de fala de 159 bits codificada em duas classes de bits :

· Classe 1. Composta de 77 bits e representa a porção da seqüência de dados de fala na qual a codificação convolucional é aplicada. Para oferecer uma proteção extra aos 12 bits perceptualmente mais significativos dos bits da Classe 1 um CRC de 7 bits é usado para propósito de detecção de erro e é calculado tomando os 12 bits perceptualmente mais significativos para cada quadro.

· Classe 2. Composta de 82 bits que são transmitidos sem qualquer proteção contra erro.

2.2 Verificação de Redundância Cíclica

Os 7 bits de verificação de redundância cíclica (CRC) são calculados sobre os 12 bits perceptualmente mais significativos do quadro. O polinômio gerador é dado por :

gCRC(X) = X7 + X5 + X4 + X2 + X + 1 (1)

2.3 Codificação Convolucional

A entrada do codificador convolucional é uma seqüência de 89 bits :

· 77 bits pertencem a Classe 1 do codec de fala.

· 7 bits pertencem a saída do codificador CRC.

· 5 bits zeros (tail bits) são adicionados para a decodificação por algoritmo de Viterbi.

O código convolucional utilizado é de taxa (R) igual a $\frac{1}{2}$e memória (m) igual a 5. Existem portanto 2m = 25 = 32 estados neste código. Os polinômios geradores do código são :

g0(D) = 1 + D + D3 + D5 (2)

g1(D) = 1 + D2 + D3 + D4 + D5 (3)

A saída do codificador convolucional da figura 3.15 alterna entre estes 2 polinômios começando por g0(D) em cada slot de tempo.

3.3 Entrelaçamento

Antes de ser transmitido, os dados de fala codificados são entrelaçados em dois slots de tempo com os dados de fala de quadros adjacentes. Ou seja, cada slot de tempo contém informação de 2 quadros do codec de fala. Os dados de fala são entrelaçados segundo a matriz 3.11.

Os dados são escritos linha a linha e são lidos (transmitidos) coluna a coluna. Os dois quadros de fala são referenciados por x e y, onde x é o quadro de fala anterior e y é o quadro mais recente. Os dados são colocados no arranjo de entrelaçamento de modo a haver uma mistura entre os bits de Classe 2 do codec de fala com os bits da Classe 1 codificados convolucionalmente. Os bits da Classe 2 são seqüencialmente colocados no arranjo e ocupam as seguintes localizações :

· 0, 26, 52, 78

· 93 a 129

· 130, 156, 182, 208

· 223 a 259

3. MODELO PROPOSTO

De um modo geral, um código de bloco binário (n, k, dmin) transforma k bits de informação em n bits, tendo portanto uma taxa de codificação R = k/n. A distância mínima dmin de um código é a menor distância de Hamming entre duas palavras código válidas. A substituição do codificador convolucional por um banco de sete codificadores em paralelo usando código de bloco de Golay (24,12,8) estendido na versão GAC (Código Array Generalizado) [2] mantendo a mesma taxa total do código convolucional, pode ser vista na figura 4.2. A escolha do uso de código de bloco se justifica pela não necessidade do uso dos cinco tail bits responsáveis em zerar a máquina geradora de estados do código convolucional, de forma que se possa aproveitá-los para proporcionar uma proteção extra aos doze bits perceptualmente mais significativos da saída do codificador de fala.

4. DESEMPENHO DO SISTEMA

As curvas apresentadas na figura 4.7 são comparativas entre o sistema TDMA [1] atual com código convolucional e o modelo proposto com código de Golay, ambos em canal AWGN (Additive White Gaussian Noise) com densidade espectral bilateral de potência de ruído $\frac{N_0}{2}$ modelado como Gaussiano e desvanecimento Rayleigh [3] por múltiplos percursos a partir do diagrama em blocos da figura 4.4.

O método de modulação é o deslocado $\frac{\pi}{4}$ e codificado diferencialmente - $\frac{\pi}{4}DQPSK$ (*Differentialy Quadrature Phase Shift Keying*) [5].

O código de Gray é usado no mapeamento dos símbolos a serem transmitidos. A cada mudança de fase são associados dois bits.

Note que há rotação de $\frac{\pi}{4}$ da constelação básica QPSK para símbolos denotados por bolas brancas, na figura 3.16, em relação aos símbolos denotados por bolas pretas da outra constelação.

A informação é codificada diferencialmente, os símbolos são transmitidos como mudanças de fase ao invés de fases absolutas.

Uma seqüência binária b_m é convertida em 2 seqüências binárias Xk e Yk de acordo com

a figura 3.17. Estas duas sequências são então codificadas em Ik e Qk de acordo com as equações :

Ik = Ik-1cos[$\Delta\Phi\,(X_k, Y_k)$] - Qk-1sen[$\Delta\Phi\,(X_k, Y_k)$] (4)

Qk = Ik-1sen[$\Delta\Phi\,(X_k, Y_k)$] + Qk-1cos[$\Delta\Phi\,(X_k, Y_k)$] (5)

onde Ik-1 e Qk-1 são amplitudes do pulso anterior. A mudança de fase $\Delta\Phi$ é determinada de acordo com a tabela 3.2.

Ik e Qk estão compreendidos entre$\{0, \pm 1, \pm\}$.

O demodulador recupera o sinal recebido do canal calculando a distância mínima [7] entre o sinal recebido e o conjunto de sinais transmitidos. O de-entrelaçador utiliza método inverso ao entrelaçador descrito e o decodificador foi implementado usando diagrama de treliça pelo algoritmo de Viterbi com decisão suave [2,4] em ambos os casos, para o sistema atual usando código convolucional e para o modelo proposto.

4.1 Resultados

A figura 4.7 apresenta os resultados gráficos comparativos obtidos na simulação do desempenho de sistema (curva de probabilidade de erro de bit pela relação sinal - ruído dada por Eb/N0) do sistema atual com código convolucional e do sistema proposto com código de Golay, ambos simulados a partir da figura 4.4.

Note que o sistema que utiliza o conjunto de codificadores de Golay apresenta um desempenho 2.5 dB pior que codificador convolucional para

Perro $=10^{-4}$. Entretanto é possível utilizar os bits de preenchimento da figura 4.2 para uma melhor proteção aos bits perceptualmente mais significativos da saída do codec.

5. CONCLUSÕES

A possibilidade de usar os cinco tail bits da codificação convolucional para uma proteção extra aos bits mais significativos do codificador de fala e com isso oferecer uma melhoria na qualidade de voz recebida pelo usuário (destino) justifica a pequena perda de desempenho do sistema proposto em relação ao atual. Com a substituição do banco de sete codificadores com código de Golay estendido (24,12,8) por outro banco de seis codificadores com código

de Golay (23,12,7) e um código BCH (63,36,11) "puncionado" [6] para (40,12,12), a taxa de codificação continua sendo igual ao a do código convolucional. Como este código (40,12,12) resulta em uma potencialidade de corrigir todos os padrões de até cinco erros e alguns de seis erros, ele oferecerá uma proteção maior aos bits perceptualmente mais significativos a ele encaminhados resultando finalmente na melhoria da qualidade do sinal de voz recebida pelo usuário cumprindo assim o objetivo do modelo proposto.

6. REFERÊNCIAS

[1] TIA/EIA Interim Standard IS-136.2 : 800 MHz Cellular - Radio Interface - Mobile Station Compatibility - Traffic Channels and FSK Control Channel, December 1994.

[2] Honary, B., Markarian G., Trellis Decoding of Block Codes, 1997.

[3] Proakis, John G., Digital Communications, 1995.

[4] Shu, L. & Costello Jr., D. J., Error Control Coding Fundamentals and Applications, Prentice-Hall, 1983.

[5] Yacoub, Michel Daoud., Foundations of Mobile Radio Engineerring, 1993.

[6] Clark, George C., Cain J.B., Error-Correction for Digital Communications, 1981.

[7] Lee, Edward A., Messerschmitt, David G., Digital Communications-Second Edition-1994.

Bibliografia

[Clark81] Clark, George C. e Cain J.B. *Error-Correction for Digital Communications*,1981.

[Cos92] Wu, J. e Costello D. J. "New Multilevel Codes Over GF(q)", *IEEE Transactions on Information Theory,* vol. 38, no 3, pp. 933-939, maio 1992.

[Darnell93] Honary, B. ; Markarian, G. e Darnell, M. "Low complexity trellis decoding of linear block codes", *Proceeding of IEEE International Symposium on Information Theory ISIT"94*, Trondheim, Norway, p. 341, jun. 1993.

[Elias54] Elias, P. "Error Free Coding", *IRE Transactions on Information Theory,* vol. 4, 1954, pp. 29-37.

[Farrell86] Blaum, M. ; Farrell, P.G. e van Tilborg, H.C.A. "A class of burst-error-correcting array codes", *IEEE Transactions on Information Theory,* vol. 32, no 6, pp. 836-839, nov. 1986.

[Farrell92] Farrell, P.G. "A survey of array error control codes", *European Transactions on Telecommunications and Related Techniques (ETT)*, vol. 3, no 5, pp. 17-30, set. 1992.

[Farrell93] Honary, B. ; Markarian, G. e Farrell, P.G. "Generalised array codes and their trellis structure", *Electronics Letters*, vol. 29, no 6, pp. 541-542, mar. 1993.

[Forney88a] Forney, G.D. "Coset codes-Part 1 : Introduction and geometrical classification", *IEEE Transactions on Information Theory,* vol. 34, no 5, pp.

1123-1151, 1988.

[Forney88b] Forney, G.D. "Coset codes-Part 2 : Binary lattices and related codes", *IEEE Transactions on Information Theory,* vol. 34, no 5, pp. 1152-1187, set. 1988.

[Harte98] Harte, L. ; Smith, A. D. e Jacobs, C. A., *IS*-136 *TDMA Technology, Economics and Services,* Artech House Publishers, 1998.

[Honary93] Honary, B. e Markarian, G. "Low complexity trellis decoding of Hamming codes", *Electronics Letters*, vol. 29, no 12, pp. 1114-1116, jun. 1993.

[IS-136.2] TIA/EIA Interim Standard IS-136.1 : 800 MHz TDMA Cellular - Radio Interface - Mobile Station Compatibility - Traffic Channels and FSK Control Channel, dez. 1994.

[Kaya93] Honary, B. ; Markarian, G.S. ; Kaya, L. e Darnell, M. "Maximum likelihood decoding of array codes with trellis structure", *IEE Proceeding- I,* vol. 140, no 5, pp. 340-345, out. 1993.

[Lin83] Lin, S. e Costello, D. *Error Control Coding : Fundamentals and Applications,* Prentice-Hall, Englewood Cliffs, NY, 1983.

[Mac Williams98] MacWilliams, F. J. e Sloane, N.J.A. *The Theory of Error-Correcting Codes.* Nova Iorque : North Holland, cap.10, pp. 294-301, 1978.

[Markarian93] Honary, B. e Markarian, G. "New simple encoder and trellis decoder for Golay codes", *Electronics Letters*, vol. 29, no 25, pp. 2170-2171, dez. 1993.

[Markarian97] Markarian, G. e Honary, B. *Trellis Decoding Of Block Codes.* Dordrecht : Kluwer Academic Publishers, 1997.

[McEliece94] McEliece, R.J. "The Viterbi decoding complexity of linear block codes", *Proceedings of IEEE International Symposium on Information Theory,* Trondheim, Norway, p.341, 1994.

[McEliece96] McEliece, R.J."On the BCJR trellis for linear bolck codes", *IEEE Transactions on Information Theory,* vol. 42, no 4, pp. 1072-1092, 1996.

[Muder88] Muder D.J. "Minimal trellises for block codes", *IEEE Transactions on Information Theory,* vol. 34, no 5, pp. 1049-1053, 1992 IN Markarian, G. e Honary B. *Trellis Decoding Of Block Codes.* Dordrecht : Kluwer Academic Publishers, 1997.

[Papoulis91] Papoulis, Athanasios *Probability, random variables, and stochastic processes.* New York : McGraw-Hill, 1991.

[Pless68] Pless, V. "On the uniqueness of Golay codes", *Journal of Combinatorial Theory,* vol. 5, pp. 215-228, 1968 IN Markarian, G. e Honary B. *Trellis Decoding Of Block Codes.* Dordrecht : Kluwer Academic Publishers, 1997 IN Markarian, G. e Honary B. *Trellis Decoding Of Block Codes.* Dordrecht : Kluwer Academic Publishers, 1997.

[Pro95] Proakis, J.G. *Digital Communications.* Nova Iorque : McGraw-Hill, 3°ed., 1995.

[Slepian92] Slepian, D. "Some further theory of group codes", *Bell Systems Technical Journal,* vol. 39, pp. 1219-1252, set. 1960.

[Stein66] Schwartz, M.; Bennett, W. R. e Stein, S. *Communications Systems and Techniques.* Nova Iorque : McGraw-Hill, pp. 395-403, 1966.

[Victor95] LI, O.K. Victor, Personal Communication Systems. *Procedings Of The IEEE.* set. 1995.

[Wolf78] Wolf, J.K. "Efficient maximum likeliwoood decoding of linear block codes using a trellis", *IEEE Transactions on Information Theory,* vol. 24, pp. 76-80, 1988.