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

Técnicas de Redução de Estados Aplicadas à Equalização Turbo

Autor:

Miguel David Cosac

Orientador:

Prof. Dr. Jaime Portugheis

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Banca Examinadora:

Prof. Dr. Jaime Portugheis (Presidente)	FEEC/UNICAMP
Prof. Dr. César Kyn D'Ávila	CEDET
Prof. Dr. Amauri Lopes	FEEC/UNICAMP
Prof. Dr. Dalton Soares Arantes	FEEC/UNICAMP

Campinas, fevereiro de 2004.

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

Cosac, Miguel David

C82t Técnicas de redução de estados aplicadas à equalização turbo / Miguel David Cosac. – Campinas, SP: [s.n.], 2004.

Orientador: Jaime Portugheis.

Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Comunicações digitais. 2. Códigos de controle de erros (Teoria da informação). I. Portugheis, Jaime. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Este trabalho considera a utilização de um sistema de comunicação digital através de um canal que introduz interferência intersimbólica (IIS). No receptor, tanto o canal quanto o codificador de canal são modelados através de diagramas de treliça e um algoritmo iterativo (turbo) é utilizado para detecção da informação. Inicialmente, verifica-se que o desempenho do algoritmo, para razões sinal-ruído altas, é praticamente o mesmo do sistema utilizado num canal sem IIS. Posteriormente, com intuito de diminuir a complexidade de implementação do receptor, é analisado o uso de técnicas de redução de estados da treliça do canal. As técnicas utilizadas, que consistem no truncamento dos coeficientes do canal, na estimação dos coeficientes do canal e no algoritmo M, reduzem consideravelmente os cálculos de implementação do algoritmo. Os resultados da análise mostraram que o algoritmo M possui o melhor compromisso entre desempenho e complexidade de implementação.

Abstract

The use of a digital communication system through an intersymbol interference (ISI) channel is considered. At the receiver, both the channel and the channel encoder are modelled as trellis diagrams and an iterative (turbo) algorithm is used for information detection. For high sinal to noise ratio, the algorithm performance is verified to be almost the same of a channel without ISI. Afterwards, some reduced-state techniques are analyzed, as a way of reducing the receiver implementation complexity. These techniques consist of the channel coefficients truncation, the channel coefficients estimation and the M Algorithm, resulting in a significant reduction in the receiver implementation computations. The results indicate that the M algorithm has a better trade-off between performance and complexity.

À memória
de meu pai.

Agradecimentos

Agradeço

À minha família, pelo amor, compreensão, incentivo e dedicação.

Ao prof. Jaime Portugheis, pela dedicação, confiança e valiosos ensinamentos.

Aos professores César Kyn D'Ávila, Amauri Lopes e Dalton Soares Arantes que gentilmente se dispuseram a contribuir com este trabalho.

Ao grande amigo Daniel Carvalho da Cunha pela enorme ajuda, sobretudo nos momentos mais difíceis e pelas dicas no decorrer deste trabalho.

Aos funcionários, professores e amigos da FEEC, pelo apoio indispensável.

Ao Conselho de Aperfeiçoamento do Pessoal do Ensino Superior - CAPES, pelo apoio financeiro.

E a todos aqueles que, direta ou indiretamente, contribuíram para a realização deste trabalho.

Sumário

Lista de Figuras

Lista de Tabelas

Abreviaturas

AR:	Autoregressivo
ARMA:	Autoregressivo de Média Móvel
AWGN:	Additive White Gaussian Noise
BCJR:	Bahl, Cocke, Jelinek and Raviv algorithm
BER:	Bit Error Rate
BPSK:	Binary Phase Shift Keying
CC:	Codificador Convolucional
CCE:	Código Corretor de Erro
FIR:	Finite Impulse Response
GF:	Galois Field
IIR:	Infinite Impulse Response
IIS:	Interferência Intersimbólica
MA:	Média Móvel
MAP:	Maximum a Posteriori
ML:	Maximum Likelihood
MLSD:	Maximum Likelihood Sequence Detector
MSE:	Mean Square Error
PAM:	Pulse Amplitude Modulation
PDS:	Power Density Spectrum
RSC:	Recursive Systematic Code
RSR:	Razão Sinal-Ruído
SISO:	Soft-Input Soft-Output
VA:	Variável Aleatória
WMF:	Whitening Matched Filter

1

Introdução

1.1 Histórico

Em 1948, o matemático e engenheiro eletricista Claude Shannon publicou um artigo intitulado "*A Mathematical Theory of Communication*" [1], marcando o início da Teoria de Informação. Neste artigo, Shannon provou que, se a taxa na qual a informação é transmitida for menor que a capacidade do canal, existem códigos corretores de erro (CCE) capazes de fornecer níveis arbitrariamente altos de confiabilidade na saída do receptor. Esse resultado é conhecido como Teorema da Codificação de Canal. No entanto, mesmo sabendo-se na época da existência de tais códigos, não se fazia idéia de como construí-los. Desde então, grandes esforços vêm sendo

realizados pela comunidade científica para a obtenção de esquemas de codificação e decodificação cujo desempenho se aproxime do limite calculado por Shannon.

Um obstáculo ao uso de códigos que atinjam níveis altos de confiabilidade é o compromisso que se deve ter entre desempenho e complexidade. Geralmente, a utilização de códigos mais longos e complexos leva a taxas menores de erro de bit, BER (*Bit Error Rate*). Entretanto, isso implica no aumento exponencial da complexidade do sistema, limitando as aplicações a sistemas a baixas taxas de transmissão.

Uma maneira de contornar esse problema é a utilização de códigos concatenados. O conceito de concatenação foi introduzido por David Forney, em 1965 [2]. Forney aplicou a concatenação serial utilizando dois codificadores, um Reed-Solomon, externo, e um convolucional, interno. Na decodificação, há um decodificador para cada código, separadamente. A vantagem da utilização desse método é permitir que a probabilidade de erro decaia exponencialmente, tendo-se um aumento na complexidade de decodificação que é uma potência pequena do comprimento do bloco transmitido.

Em 1993, foi introduzido por Berrou, Glavieux e Thitimajshima [3] o conceito de concatenação paralela. A esse esquema de codificação e decodificação utilizando códigos concatenados paralelamente e decodificação iterativa dá-se o nome de codificação turbo. Esse sistema atinge um desempenho bem próximo da capacidade de canal, a poucos décimos de dB do limite de Shannon. De lá para cá, a codificação turbo vem sendo aplicada nos mais diversos tipos de sistemas de transmissão e recepção, demonstrando resultados promissores.

Recentemente, utilizando-se do mesmo princípio turbo, agora aplicado a codificadores convolucionais serialmente concatenados, Benedetto e outros conseguiram atingir um desempenho equiparado à codificação turbo original e, em alguns casos, até superior [4]. A concatenação serial pode ser aplicada a dois ou mais codificadores de canal separados por entrelaçadores ou a codificadores de canal concatenados com o canal com interferência intersimbólica (IIS) [5]. Nesse segundo caso,

a decodificação iterativa aplicada à detecção da informação é chamada de "equalização turbo", por analogia aos códigos turbo. Em sistemas de comunicações, a equalização é utilizada para compensar a distorção do sinal, causada pelo canal ou meio de transmissão. A equalização turbo foi primeiro utilizada em [6].

Um critério comum de desempenho é a BER. O receptor ótimo, que minimiza a BER, é um detector *maximum a posteriori* (MAP) ou de máxima verossimilhança (ML). A diferença entre eles é que enquanto um detector ML considera que todos os símbolos do alfabeto de entrada são eqüiprováveis, o detector MAP tem conhecimento das probabilidades de ocorrência de tais símbolos. Esse receptor ótimo não remove a IIS, mas tenta encontrar as entradas do canal mais prováveis dados os símbolos de saída corrompidos pelo ruído, o que equivale a minimizar a BER.

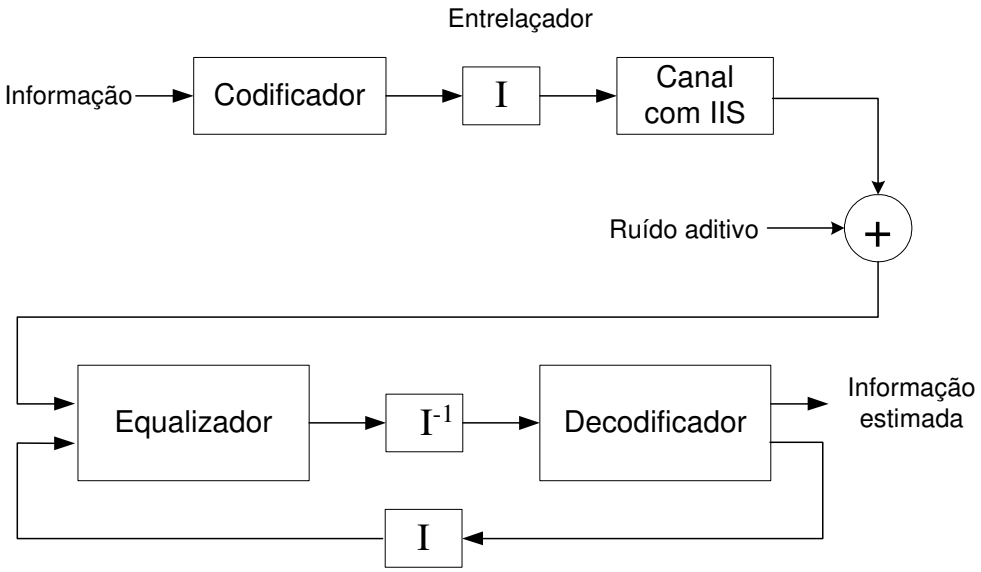

Figura 1.1: Sistema de transmissão utilizando decodificação turbo em canais com IIS.

Vários sistemas de comunicações na presença de IIS utilizam codificadores convolucionais e equalizadores com decisão suave, como mostrado na Figura 1.1. Um dos maiores desafios nesses sistemas está em diminuir a complexidade computacional

em ambos os blocos, equalizador e decodificador.

Neste trabalho, é estudada a utilização da decodificação turbo aplicada a codificadores convolucionais concatenados serialmente, um dos quais representa a codificação de canal propriamente dita e, o outro, o canal com IIS. É utilizado um esquema de equalização e decodificação sub-ótimo que apresenta elevado gasto computacional. Posteriormente, são testadas técnicas de redução de estados para a treliça do bloco de equalização e comparados os resultados, analisando-se o compromisso complexidade-desempenho dos sistemas.

O objetivo principal do trabalho é avaliar, tanto em termos de desempenho quanto complexidade, as técnicas propostas de redução de estados. Sem dúvida, a diminuição de complexidade proposta nos três métodos faz com que o desempenho seja afetado, mas não se sabe quanto se perde com isso, nem mesmo se as técnicas são viáveis, já que, pelo menos para as duas primeiras, há um descasamento entre os possíveis símbolos na saída do canal e os símbolos interpretados pelo receptor.

É importante frisar que não é interesse deste trabalho projetar códigos que levem ao melhor desempenho do sistema. Deseja-se, na verdade, verificar o desempenho desse sistema com o uso de um código simples, de fácil implementação. Entende-se que as técnicas propostas podem ser implementadas independentemente da complexidade do código utilizado.

1.2 Organização

O Capítulo 2 apresenta os conceitos básicos utilizados no decorrer do trabalho, tais como: codificadores convolucionais, concatenação de códigos, entrelaçadores, canal AWGN, modelo discreto de canal AWGN com IIS e cálculo da distância mínima de evento de erro para canais com IIS.

No Capítulo 3, é feita uma descrição do sistema implementado. Trata-se de um sistema de decodificação iterativa aplicada a um codificador convolucional con-

catenado serialmente com um canal na presença de IIS. O sistema de decodificação iterativa utilizado é detalhado. São apresentados, nesse capítulo, os resultados da simulação desse sistema, que atinge um desempenho muito próximo ao do canal sem IIS para razões sinal-ruído (RSR) altas.

O Capítulo 4 descreve três métodos de redução de estados: truncamento dos coeficientes do canal, estimação dos coeficientes do canal e algoritmo M. Os resultados das simulações de cada um desses métodos são apresentados e, por fim, uma comparação entre os métodos é feita. Nesse capítulo é realizada uma análise acerca da complexidade de cada método, comparando-se a quantidade de cálculos necessárias para cada símbolo recebido por iteração com a do sistema implementado no caítulo anterior. Há ainda uma comparação desses métodos com métodos já propostos de redução de complexidade para equalização linear.

No Capítulo 5, são apresentadas as conclusões obtidas, baseadas nos resultados das simulações e as perspectivas para trabalhos futuros.

1.3 Contribuições

Como principais contribuições desse trabalho pode-se citar:

- Verificação do desempenho de um sistema de equalização turbo utilizando-se um equalizador MAP baseado no BCJR modificado [5];
- Utilização de métodos de redução de estados para equalização turbo diferentes dos demais métodos encontrados na literatura;
- Análise do desempenho e da complexidade dos métodos estudados e comparação com outros métodos de redução de complexidade já propostos.

2

Conceitos Básicos

Neste Capítulo serão expostos alguns conceitos básicos de teoria de informação e codificação necessários ao entendimento do trabalho.

2.1 Códigos Convolucionais

Num codificador convolucional, as n saídas de cada instante dependem não somente das k entradas nesse mesmo instante, mas também de m grupos de k entradas anteriores [7]. Dessa forma, o codificador convolucional é caracterizado pela tripla n, k, m, sendo m a ordem da memória do codificador. De maneira geral, um codificador possui k registradores de deslocamento, um para cada seqüência de entrada,

os quais não possuem, necessariamente, o mesmo número de elementos de memória. Se K_j é o número de elementos de memória do j-ésimo registrador, então a ordem da memória do codificador, m, é

$$m \triangleq \max_{1 \leq j \leq k} K_j. \tag{2.1}$$

Um parâmetro importante é o comprimento de restrição (ou *constraint length*), n_A, que pode ser interpretado como sendo o máximo número de símbolos na saída que podem ser afetados por um único símbolo de entrada. Como cada símbolo permanece no codificador por, no máximo, m+1 deslocamentos, tem-se

$$n_A \triangleq n(m+1). \tag{2.2}$$

Para cada símbolo de entrada são gerados n símbolos de saída e, portanto, o código possui uma taxa $R = k/n$.

Códigos convolucionais são códigos lineares, ou seja, as 2^k palavras-código formam um sub-espaço vetorial k-dimensional do espaço vetorial de todas as n-uplas sobre o corpo GF(2) (*Galois Field*) [7].

Para seqüências de entrada do codificador de tamanho kL, a palavra-código possui comprimento $n(L+m)$, supondo-se que a seqüência de informação seja terminada com mk zeros. Assim, a taxa do bloco resultante é $\frac{kL}{n(L+m)}$. A perda em taxa, normalizada pela taxa original, é dada por

$$\frac{k/n - kL/n(L+m)}{k/n} = \frac{m}{L+m}. \tag{2.3}$$

Uma forma de representar as operações dos codificadores convolucionais é através do uso de polinômios. Seja, por exemplo, um codificador convolucional binário $(2, 1, m)$ com seqüência de entrada

$$u = (u_0, u_1, u_2, ...)$$

e saídas

$$v^{(j)} = (v_0^{(j)}, v_1^{(j)}, v_2^{(j)}, ...),\ j = 1, 2.$$

Constrói-se, então, os polinômios

$$u(D) = u_0 + u_1 D + u_2 D^2 + ...$$

e

$$v^{(j)}(D) = v_0^{(j)} + v_1^{(j)} D + v_2^{(j)} D^2 + ...,\ j = 1, 2.$$

A palavra-código pode ser escrita através do polinômio $v(D)$,

$$v(D) = v^{(1)}(D^2) + Dv^{(2)}(D^2),$$

sendo

$$v^{(j)}(D) = u(D)g^{(j)}(D),$$

e

$$g^{(j)}(D) = g_0^{(j)} + g_1^{(j)} D + g_2^{(j)} D^2 + ... + g_m^{(j)} D^m$$

que representam os polinômios geradores.

De uma forma mais compacta, define-se os vetores

$$\mathbf{U}(D) \triangleq [u^{(1)}(D), u^{(2)}(D), ...u^{(k)}(D)]$$

$$\mathbf{V}(D) \triangleq [v^{(1)}(D), v^{(2)}(D), ...v^{(n)}(D)]$$

e a matriz

$$\mathbf{G}(D) = \begin{bmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \cdots & g_1^{(n)}(D) \\ \vdots & & & \vdots \\ g_k^{(1)}(D) & g_k^{(2)}(D) & \cdots & g_k^{(n)}(D) \end{bmatrix}.$$

Pode-se, então, escrever

$$\mathbf{V}(D) = \mathbf{U}(D)\mathbf{G}(D).$$

Sendo assim, $v(D)$ é obtida através de $\mathbf{V}(D)$ como

$$v(D) = v^{(1)}(D^n) + Dv^{(2)}(D^n) + ... + D^{(n-1)}v^{(n)}(D^n).$$

Dada a matriz geradora do código, constrói-se o codificador convolucional. Seja, por exemplo, uma matriz geradora dada por:

$$\mathbf{G}(D) = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1+D^2 & 1+D+D^2 \end{bmatrix}.$$

O número de linhas da matriz representa as k entradas e o número de colunas as n saídas. O maior grau dentre os polinômios $g^{(j)}(D)$ representa o número de registradores de memória, m.

Assim, tal matriz gera um código convolucional com taxa de bloco resultante igual a $\frac{2L}{3L+6}$ e cujo codificador é dado pela Figura 2.1.

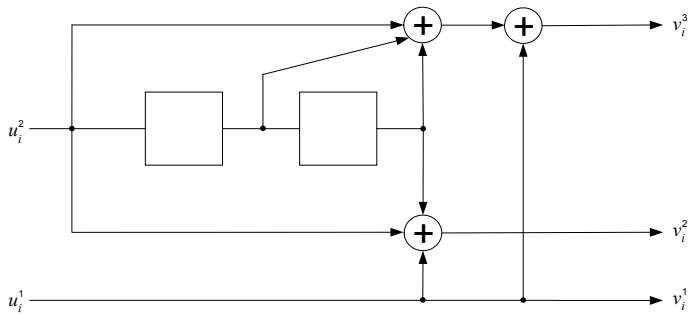

Figura 2.1: Codificador convolucional (3,2,2).

Uma representação bastante usual para codificadores convolucionais é através de uma máquina de estados finitos. O número de estados é definido por 2^K, onde K é a memória total, $K = \sum_{j=1}^{k} K_j$. Em geral, os estados são definidos pelo conteúdo das memórias do codificador. A Figura 2.2 mostra o exemplo de um codificador convolucional cuja matriz geradora é dada por $G = [101, 111]$. No diagrama, cada nó representa um estado, formado por dois elementos de memória. Os ramos são as transições de um estado para outro. Cada símbolo de informação na entrada do codificador, u_i no instante i, gera dois símbolos de saída, $v_i^{(1)}$ e $v_i^{(2)}$, determinados pela entrada atual, u_i, e pelas $m = 2$ entradas anteriores, u_{i-1} e u_{i-2}.

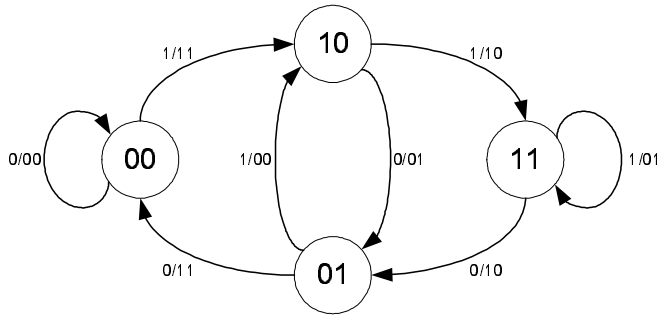

Figura 2.2: Diagrama de estados para G=[101, 111].

Outra representação utilizada para codificadores convolucionais é na forma de uma treliça, que nada mais é que uma extensão do diagrama de estados no domínio temporal. A Figura 2.3 mostra o diagrama de treliça de um código (3,1,2) com $G(D) = [1 + D, 1 + D^2, 1 + D + D^2]$ e $L = 5$. Se a seqüência de informação possui comprimento $L = 5$, as palavras-código, formadas por todas as seqüências que partem do estado nulo e voltam ao estado nulo após $L + m$ instantes, possuem comprimento $N = n(L + m) = 21$. Ao todo, existem $2^{kL} = 32$ palavras-código.

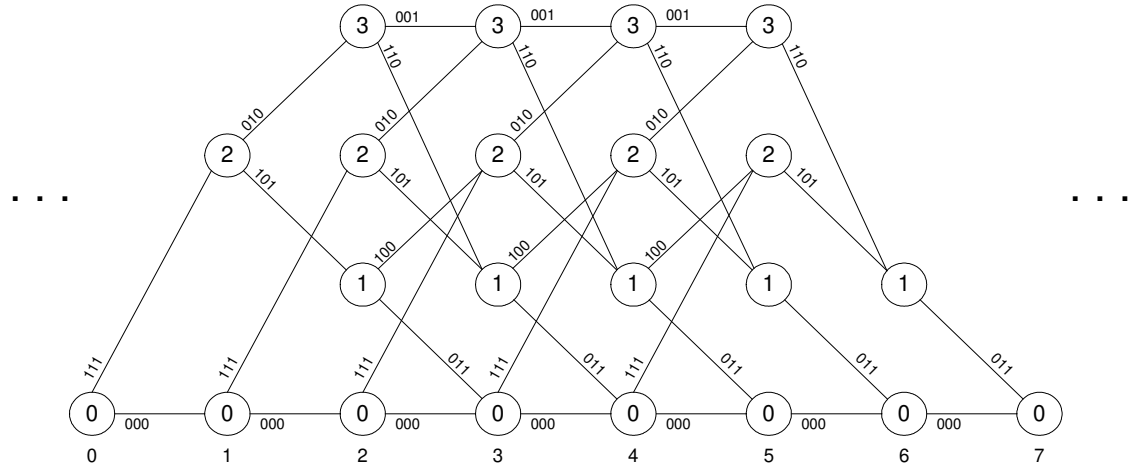

Figura 2.3: Exemplo de treliça para um código convolucional (3,1,2).

Os códigos convolucionais podem ser FIR (*Finite Impulse Response*) ou IIR

(*Infinite Impulse Response*), sistemáticos ou não-sistemáticos [8].

Um codificador é do tipo FIR ou não-recursivo se a saída pode ser calculada como combinação linear da entrada atual mais um número finito de entradas passadas. Nos codificadores IIR ou recursivos a saída depende não somente das entradas atuais e anteriores, mas também das saídas anteriores. Sendo assim, a saída de um codificador IIR pode facilmente depender de um número infinito de entradas anteriores.

Num código sistemático a entrada ou uma das entradas do codificador compõe sempre uma das saídas. Caso isso não ocorra, o código é dito não-sistemático. A Figura 2.4 mostra exemplos de codificadores convolucionais. É válido notar que para os mesmos parâmetros de um código convolucional, como taxa, n, k e m, pode haver diferentes tipos de codificador.

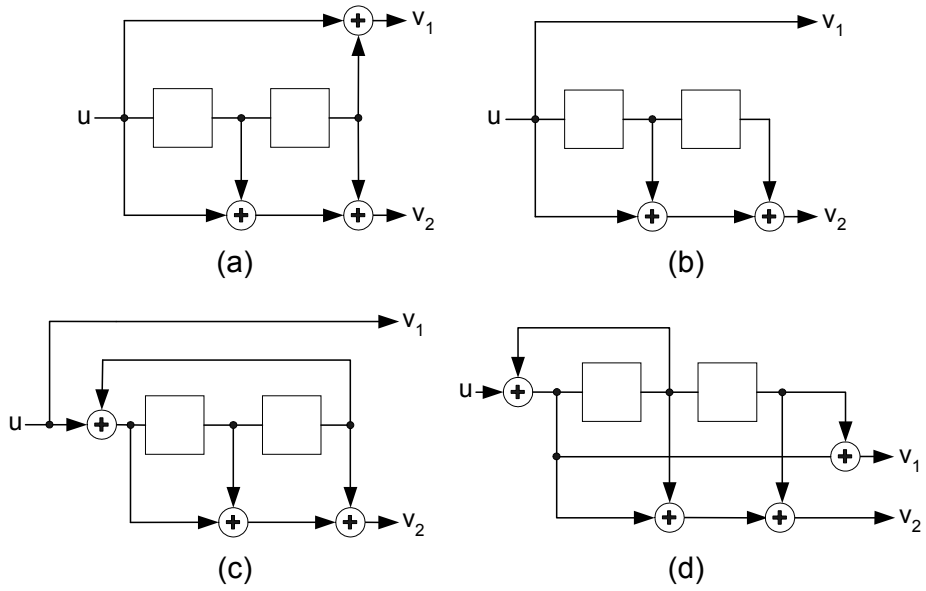

Figura 2.4: Codificadores convolucionais (2,1,2): (a)FIR, não-sistemático; (b)FIR, sistemático; (c)IIR, sistemático; (d)IIR, não-sistemático.

Todo código convolucional tem um codificador FIR minimal e um codificador sistemático minimal. Um codificador minimal é aquele cujo tamanho da memória total, K, é o menor possível.

Um parâmetro importante para se determinar o desempenho de um código convolucional é a distância mínima, $d_{\min}$, desse código. Por definição, para códigos lineares, $d_{\min}$ é a menor distância de Hamming entre quaisquer duas palavras-código, ou ainda, o peso da palavra-código com menor peso de Hamming de uma seqüência que não seja composta somente de zeros [7]

$$\begin{aligned} d_{\min} &\stackrel{\Delta}{=} \min\{d(v^1, v^2) : u^1 \neq u^2\} \\ &= \min\{w(v^1 + v^2) : u^1 \neq u^2\} \\ &= \min\{w(v) : u \neq 0\} \\ &= \min\{w(uG) : u \neq 0\}. \end{aligned} \tag{2.4}$$

Uma forma de se obter $d_{\min}$ é gerando-se a função de transferência do código, o que é feito por meio de grafos e da regra de Mason, como mostrado em [7]. Entretanto, $d_{\min}$ pode ser facilmente visualizado observando-se a treliça do código. Basta verificar o menor peso de Hamming da saída para uma seqüência que sai e retorna ao estado nulo.

2.1.1 Probabilidade de Erro de Bit

O algoritmo de Viterbi é ótimo no sentido de minimizar a probabilidade de erro de palavra-código, isto é, a probabilidade de se enviar uma seqüência do código e decidir por outra seqüência não pertencente ao código. Dada uma seqüência de entrada toda nula, tem-se, quando não há erro, uma seqüência de saída toda nula. Se, num instante de tempo, i, o caminho todo nulo não for escolhido pelo algoritmo, diz-se que ocorreu um primeiro evento de erro. A probabilidade de primeiro evento de erro, $P_i(E, i)$, independe do instante i, e é limitada por:

$$P_i(E, i) = P_f(E) < \sum_{d=d_{\min}}^{\infty} A_d P_d. \tag{2.5}$$

O parâmetro A_d representa a distribuição de pesos do código e P_d, por sua vez, é a probabilidade de erro par a par, isto é, a probabilidade de se enviar uma palavra-código e decidir por outra com distância de Hamming, d, quando somente as duas palavras são comparadas.

Tendo-se a função geradora, T(X), como descrito em [7], pode-se obter o limitante superior para a probabilidade de erro de bit do algoritmo de Viterbi. Como a função geradora é dada por $T(X) = \sum_{d=d_{\min}}^{\infty} A_d X^d$, tem-se para um canal com modulação BPSK e saída não quantizada

$$P_f(E) < T(X)\,\big|_{X=e^{-RE_b/N_0}}\ . \tag{2.6}$$

A probabilidade de erro de bit, P_b, é dada pela razão entre o número de bits de informação em erro e o número total de bits de informação enviados. Seja B_d o número total de bits de informação não nulos nos caminhos de peso d da palavra-código. Então

$$P_b < \frac{1}{k} \sum_{d=d_{\min}}^{\infty} B_d P_d.$$

Definindo-se a função geradora

$$T(X,Y) = \sum_{d=d_{\min}}^{\infty} \sum_{b=1}^{\infty} A_{d,b} X^d Y^b,$$

tem-se

$$\left.\frac{\partial T(X,Y)}{\partial Y}\right|_{Y=1} = \sum_{d=d_{\min}}^{\infty} \sum_{b=1}^{\infty} b A_{d,b} X^d = \sum_{d=d_{\min}}^{\infty} B_d X^d$$

pois $B_d = \sum_{b=1}^{\infty} b A_{d,b}$ [7].

Pode-se mostrar que a probabilidade média de eventos de erro, $P(E)$, é limitada pela probabilidade de primeiro evento de erro. Assim, o limitante torna-se

$$P(E) \le P_f(E) < T(X)\,\big|_{X=e^{-RE_b/N_0}} \tag{2.7}$$

e

$$P_b(E) < \frac{1}{k}\ \left.\frac{\partial T(X,Y)}{\partial Y}\right|_{X=e^{-\frac{RE_b}{N_0}},Y=1}, \tag{2.8}$$

onde E_b é a energia média por bit transmitido e N_0 é a densidade espectral de potência unilateral do ruído.

O limitante é dominado pelo primeiro termo, de forma que pode-se aproximar

$$P_b(E) \approx \frac{1}{k}\left(e^{-\frac{RE_b}{N_0}}\right)^{d_{\min}}. \tag{2.9}$$

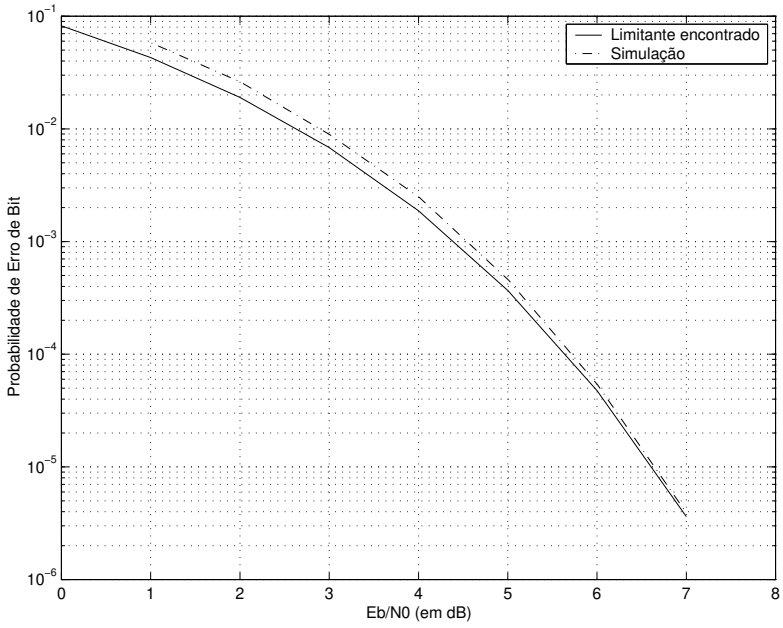

Figura 2.5: Comparação do limitante da Eq. 2.9 com a simulação para um código convolucional com matriz geradora $G = [101, 111]$.

Esse limitante foi usado para se verificar a validade da simulação do código convolucional para o canal sem interferência intersimbólica (IIS). A Figura 2.5 mostra o limitante encontrado para o código com matriz geradora $G = [101, 111]$ e o resultado da simulação desse código transmitido num canal sem IIS. Como pode ser observado, os resultados obtidos estão muito próximos do limitante.

2.2 Códigos Concatenados

Como foi dito, a concatenação de códigos permite que se alie dois ou mais códigos curtos, ditos códigos componentes, de forma que a decodificação possa ser feita separadamente. Tal técnica permite que sejam utilizadas estratégias de decodificação mais simples do que se fosse utilizado apenas um código mais longo.

Existem dois tipos de concatenação: concatenação serial e concatenação paralela, que serão descritos a seguir. Há ainda a possibilidade de mesclar ambos os tipos de concatenação, desde que essa estrutura seja respeitada na decodificação.

2.2.1 Concatenação serial

Um sistema de controle de erro com concatenação serial consiste em dois ou mais conjuntos distintos de codificadores e decodificadores. Comumente, entre os codificadores/decodificadores é colocado um bloco entrelaçador/desentrelaçador ($\mathrm{I}/\mathrm{I}^{-1}$). Esse bloco é detalhado na Seção 2.3.

A Figura 2.6 mostra um exemplo de sistema concatenado serialmente, com dois blocos de codificador/decodificador. Um bloco de informação de comprimento k é primeiramente codificado pelo codificador externo, C_1, resultando numa palavra-código de comprimento n_1. Este, por sua vez, é codificado pelo codificador interno, C_2, resultando numa palavra-código de comprimento n_2. Assim, a decodificação é realizada em dois estágios, ocorrendo primeiro a decodificação de C_2 seguida pela decodificação de C_1. A complexidade de se realizar dois estágios de decodificação é bem menor do que utilizando-se apenas um estágio que tenha a mesma eficiência dos dois estágios.

2.2.2 Concatenação paralela

O conceito de concatenação paralela foi introduzido por Berrou e outros [3]. Utilizando-se dois codificadores convolucionais componentes, paralelamente concate-

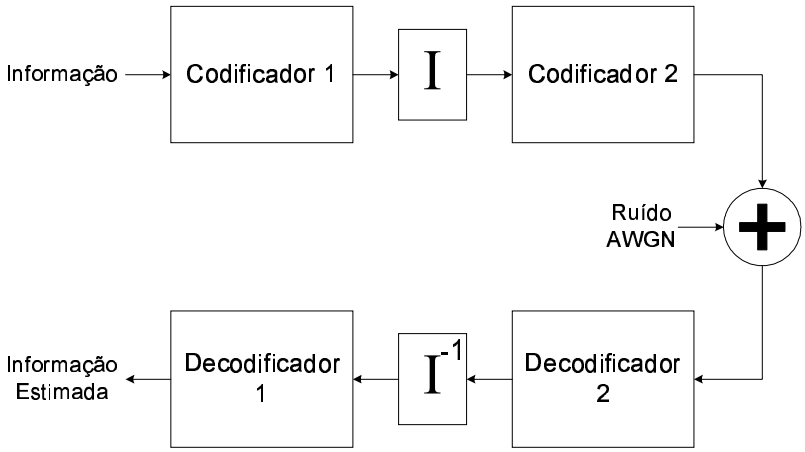

Figura 2.6: Sistema concatenado serialmente.

nados, e os respectivos decodificadores, foi desenvolvida uma nova família de códigos, os chamados *turbo codes*, ou códigos turbo. O uso de códigos sistemáticos permite a construção de codificadores concatenados como na Figura 2.7.

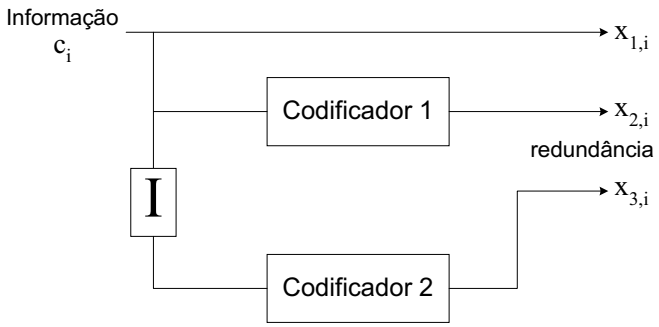

Figura 2.7: Sistema concatenado paralelamente.

Cada símbolo a ser transmitido, c_i, para cada instante i, constitui a primeira das três saídas do sistema concatenado paralelamente da Figura 2.7. As demais saídas são dadas pelos símbolos de saída de dois codificadores RSC, C_1 e C_2, separados por um entrelaçador. Como para cada símbolo de entrada, c_i, é gerada a tripla $(c_i, x_{1,i}, x_{2,i})$, a taxa, neste caso, é de $R = 1/3$.

2.3 Entrelaçadores

Entrelaçadores (ou *interleavers*) são dispositivos que tomam símbolos de um alfabeto fixo, $\mathcal{A}$, na entrada e produzem símbolos de saída idênticos, mas numa seqüência temporal diferente. Tal técnica é geralmente utilizada em canais onde ocorrem erros em rajada (*burst errors*) como em canais de comunicação sem fio (*wirelless communications channels*) e também em esquemas de codificação concatenada, nos quais o primeiro estágio de decodificação gera erros em rajada. Uma aplicação mais recente do entrelaçador, apresentada em [3], é em sistemas com decodificação turbo.

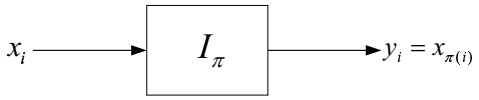

Figura 2.8: Entrelaçador.

Em geral, a ação de um entrelaçador é descrita como mostrado na Figura 2.8, ou seja, a saída no instante i, $y_i \in \mathcal{A}$, é a $\pi(i)$-ésima entrada, $x_{\pi(i)}$. Quando trabalha-se com uma seqüência de entrada, utiliza-se a notação $\mathbf{y} = I_\pi(\mathbf{x})$. O desentrelaçador (*de-interleaver*), I_π^{-1}, recebe as saídas do entrelaçador e coloca os símbolos novamente na ordem original.

Para que um entrelaçador seja realizável, é necessário que ele seja periódico, com período T finito. Da mesma forma, o respectivo desentrelaçador é também um entrelaçador com período T.

A seguir, os tipos mais comuns de entrelaçadores serão apresentados, com ênfase para os entrelaçadores aleatórios, que apresentam um bom desempenho e fácil implementação [8].

2.3.1 Entrelaçadores de Bloco

Nos entrelaçadores de bloco, os dados de entrada são escritos ao longo das linhas de um conjunto de elementos de memória configurados como uma matriz e lidos ao longo das colunas dessa matriz.

Há quatro variações básicas de entrelaçadores de bloco, como mostra a Figura 2.9. As variações se dão no modo como as colunas da matriz são lidas, de cima para baixo (CB) ou de baixo para cima (BC) e como as linhas da matriz são lidas, da esquerda para a direita (ED) ou da direita para a esquerda (DE). As seqüências de permutação geradas num vetor de 16 posições dispostos na forma matricial, como na Figura 2.9 são:

ED/CB: (0 4 8 12 1 5 9 13 2 6 10 14 3 7 11 15)
ED/BC: (12 8 4 0 13 9 5 1 14 10 6 2 15 11 7 3)
DE/CB: (3 7 11 15 2 6 10 14 1 5 9 13 0 4 8 12)
DE/BC: (15 11 7 3 14 10 6 2 13 9 5 1 12 8 4 0)

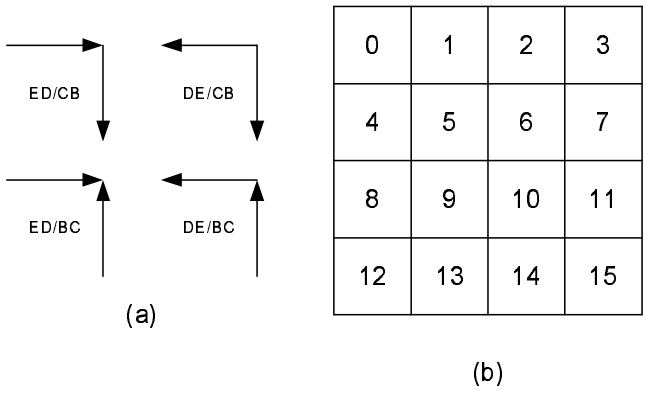

Figura 2.9: Entrelaçador de Bloco Clássico.
(a) Modos de operação; (b) Bloco 4 x 4.

2.3.2 Entrelaçadores Convolucionais

Um entrelaçador convolucional (ou multiplexado ou ainda de registrador de deslocamento) de período T multiplexa a seqüência de entrada em T subseqüências, introduzindo, para cada uma delas, um atraso e demultiplexa as T subseqüências resultantes. Assim, tal entrelaçador pode ser implementado com o uso de um conjunto de T registradores de deslocamento, onde o comprimento do j-ésimo registrador, m_j, determina o atraso. A Figura 2.10 mostra a permutação resultante de um entrelaçador convolucional com período $T = 3$, onde j é o índice de entrada e $\pi(j)$ o índice de saída.

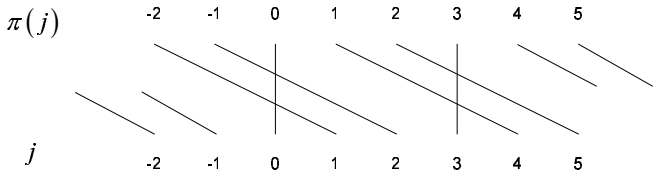

Figura 2.10: Entrelaçador Convolucional, com T=3.

2.3.3 Entrelaçador de Berrou-Glavieux

Trata-se de um entrelaçador não-uniforme utilizado por Berrou e Glavieux em [9] que diminui os padrões de peso baixo do código, associados ao entrelaçador de bloco. O entrelaçador de Berrou-Glavieux é completamente definido pelo número de colunas $C = 2^m$, pelo número de linhas $L = 2^n$, onde m e n são inteiros, e por um conjunto de números primos, $p(l),\ l = 1, ..., 8$. A partir desses valores, segue-se o algoritmo descrito a seguir.

Para cada $0 \leq j \leq C.L = T$, faça:

$$\pi(j) = c(j) + Mr(j)$$

em que:

$$r(j) = \mathrm{mod}(p(l+1)(c_0+1) - 1, N), \tag{2.10}$$

$$c(j) = \mathrm{mod}((M/2+1)(r_0 + c_0), M), \tag{2.11}$$

$$r_0 = \mathrm{mod}(j, M), \tag{2.12}$$

$$c_0 = (j - r_0)/M, \tag{2.13}$$

$$l = \mathrm{mod}((r_0 + c_0), 8). \tag{2.14}$$

As Equações 2.12 e 2.13 realizam a transposição da matriz, como ocorre nos entrelaçadores de bloco, a Eq. 2.14 seleciona um número primo do conjunto $p(l)$ de acordo com a posição do bit (linha e coluna) na matriz já transposta e a Eq. 2.10 escolhe uma nova coluna baseada no número primo selecionado. Por fim, a Eq. 2.11 encontra uma nova coluna de forma que os bits de colunas adjacentes quando passarem pelo entrelaçador sejam espaçados de $(M/2+1)$ colunas após passarem pelo entrelaçador.

2.3.4 Entrelaçadores aleatórios

Os entrelaçadores aleatórios nada mais são que entrelaçadores de bloco que proporcionam a permutação de uma seqüência de comprimento T em outra, escolhida aleatoriamente dentre as $T!$ possibilidades.

Os entrelaçadores aleatórios, no projeto de códigos turbo, têm como principal função o mapeamento de seqüências de entrada de peso baixo em seqüêcias de saída de peso alto. Para tal, a seqüência de saída do primeiro codificador componente, com peso baixo, após ser entrelaçada e passar pelo segundo codificador componente, terá peso maior, como mostra a Figura 2.11.

Para essa tarefa, os entrelaçadores de bloco convencionais podem não funcionar bem para todas as seqüências de entrada. De fato, até mesmo os entrelaçadores aleatórios podem falhar na tarefa de manter distantes na saída alguns pares de bits que estavam próximos na entrada. Por exemplo, sejam os bits de informação x_1 e x_2 que, após permutados, assumam as posições $\tilde{x}_{500}$ e $\tilde{x}_{498}$. Esta proximidade pode

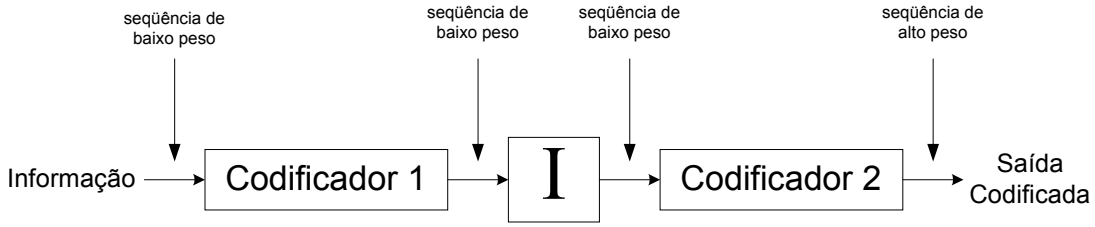

Figura 2.11: Alteração do peso das seqüências pelo entrelaçador aleatório num sistema concatenado serialmente.

causar degradação de desempenho no decodificador iterativo. A saída do decodificador MAP para códigos convolucionais, de certo modo, imita o comportamento de um canal com erro em rajada, ou seja, eventos de erro na treliça se traduzem em rajadas de erro de bits. Dessa forma, é desejável que o entrelaçador desfaça qualquer seqüência de erros em rajada, distribuindo-os aleatoriamente na seqüência permutada.

Mesmo em casos onde um entrelaçador de bloco convencional é capaz de eliminar as rajadas de erros, foi mostrado que códigos turbo usando entrelaçadores aleatórios têm um desempenho melhor [10]. Isso ocorre pelo fenômeno de *spectral thinning*, que, nos códigos turbo, trata-se de aumentar a multiplicidade de eventos de erros nas distâncias mínimas. Enquanto um código turbo com um entrelaçador de bloco tem, em geral, uma distância livre maior que outro com um entrelaçador aleatório, a multiplicidade de eventos de erro à distância mínima é tipicamente muito menor no segundo caso. A multiplicidade de eventos de erro utilizando um entrelaçador aleatório tende a crescer lentamente com o aumento da distância enquanto que, num entrelaçador de bloco, ela cresce significativamente.

Para blocos de informação maiores, os entrelaçadores aleatórios conseguem satisfazer ambos os requisitos, relativos à distribuição de erros em rajadas e *spectral thinning* do código resultante. Entretanto, para blocos menores, a probabilidade de se atingir o primeiro requisito diminui. Vários métodos foram propostos no sentido

de impor um espalhamento mínimo no entrelaçador. Um método bastante utilizado é o "*S-Random interleaver*", ou entrelaçador semi-aleatório [11].

O entrelaçador *S-Random* é construído gerando-se inteiros aleatórios j, no intervalo $1 \leq j \leq T$, sem reposição. Cada valor de j é comparado com os S inteiros previamente selecionados. Se o valor do inteiro atual estiver distante pelo menos $\pm$S dos S valores anteriores é adicionado, caso contrário, descartado. O processo continua até que todos os T inteiros tenham sido selecionados. Como o tempo de procura aumenta com o aumento de S, foi observado que escolhendo-se $S < \sqrt{T/2}$ leva à solução num tempo razoável [11]. A tabela 2.1 mostra alguns exemplos de valores do parâmetro S, dado o comprimento do bloco.

Comprimento do Bloco (T)	Distância (S)
256	11
512	15
1024	19
4096	31

Tabela 2.1: Parâmetros do entrelaçador *S-Random*

O entrelaçador S-Random é, na verdade, um entrelaçador pseudo-aleatório haja visto que ele restringe as seqüências de permutação a um espalhamento mínimo entre as posições. Mas não é difícil perceber que um entrelaçador *S-Random* com $S = 1$ é um entrelaçador aleatório puro. Para fins de obtenção de resultados mais gerais, o entrelaçador utilizado nas simulações deste trabalho é o entrelaçador aleatório puro (ou $S = 1$) em que, para cada bloco transmitido, de tamanho T, gera-se uma seqüência aleatória para permutação, também de comprimento T, sem qualquer restrição de posição. Surge daí, naturalmente, que este entrelaçador é um limitante inferior de desempenho para os demais valores de S.

2.4 Canal AWGN

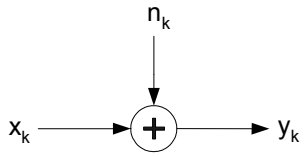

Figura 2.12: Modelo do canal AWGN.

O modelo de canal AWGN (do inglês *Additive White Gaussian Noise*), mostrado na Figura 2.12 possui entrada contínua e ruído gaussiano aditivo branco. Na prática, quando se usa modulação digital, assume-se valores discretos, que são os símbolos x_k da constelação gerada pelo modulador. Estes símbolos, ao serem transmitidos por um canal AWGN, assumirão infinitos valores devido à adição de amostras de ruído, n_k. Essas amostras são variáveis aleatórias gaussianas de média zero e variância σ^2. Para cada componente x_k do vetor transmitido, tem-se uma componente y_k do vetor recebido, com distribuição gaussiana de média x_k e variância σ^2. As probabilidades condicionais são dadas por

$$p(y_k|x_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left[\frac{-(y_k-x_k)^2}{2\sigma^2}\right]}, k = 1, 2, ..., L \tag{2.15}$$

em que L é o comprimento do bloco transmitido.

As variáveis gaussianas η_k são estatisticamente independentes, uma vez que não há correlação entre elas. Portanto, para o bloco de tamanho L, tem-se

$$p(y_1, y_1, ...y_L|x_1, x_2, ...x_L) = \prod_{k=1}^{L} p(y_k|x_k). \tag{2.16}$$

2.5 Capacidade de Canal

A capacidade de canal é definida como a quantidade máxima de informação que um canal é capaz de transmitir tendo-se uma probabilidade de erro tão pequena

quanto se deseje. Para canais AWGN limitados em faixa, a fórmula para a capacidade de canal é [1]

$$\mathbb{C} = W \log_2 \left(1 + \frac{E_s}{N_0}\right) \tag{2.17}$$

em que W é a largura de faixa do canal (em Hertz) e E_s é a energia média do sinal para um intervalo de duração T segundos. A eficiência espectral de um sinal modulado, η, é o número médio de bits de informação transmitido no intervalo T, por largura de faixa ocupada, expresso em bits por segundo por Hertz. Dessa forma, η pode ser escrito na forma $\eta = \frac{C}{W}$.

A energia média do sinal, E_S pode ser escrita como $E_b\eta$, sendo E_b a energia média gasta por bit de informação.

Fazendo-se as devidas substituições em 2.17, tem-se

$$\frac{E_b}{N_0} = \frac{2^\eta - 1}{\eta} \tag{2.18}$$

Para uma taxa $R = 1/2$ e eficiência espectral $\eta = 1/2$ bit por uso do canal tem-se

$$\frac{E_b}{N_0} = \frac{2^{0,5} - 1}{0,5} = 0,828 \tag{2.19}$$

equivalente a $-0,81$dB.

No caso de largura de faixa infinita, $\frac{\mathbb{C}}{W} \to 0$ e tem-se que

$$\frac{E_b}{N_0} = \lim_{\eta \to 0} \frac{2^\eta - 1}{\eta} = \ln 2$$

ou

$$\frac{E_b}{N_0} = -1,6 \text{ dB}.$$

Esse valor representa o limitante inferior de energia por bit transmitido, E_b, para que haja comunicação confiável num canal AWGN.

2.6 Modelo Discreto do Canal AWGN com IIS

Seja o sinal recebido

$$y(t) = \sum_{k=1}^{K} S_k h(t - kT) + e(t),$$

onde $h(t)$ é o formato do pulso recebido, que não necessariamente obedece ao critério de Nyquist, S_k são os símbolos na saída do transmissor e $e(t)$ é um ruído aditivo.

Em canais limitados em faixa, resultando em IIS, convém obter-se um modelo equivalente em tempo discreto para um sistema em tempo contínuo (analógico).

Aplicando-se o critério da distância mínima [12], o receptor escolhe a seqüência de símbolos que satisfaz

$$\max_{\{S_k, 1 \leq k \leq K\}} \left[2\mathrm{Re} \left\{ \sum_{k=1}^{K} z_k S_k^* \right\} - \sum_{k=1}^{K} \sum_{m=1}^{K} S_k S_m^* \rho_h(m-k) \right], \tag{2.20}$$

onde $\rho_h(k)$ é a função de autocorrelação do pulso, definida por

$$\rho_h(k) = \int_{-\infty}^{\infty} h(t) h^*(t - kT) dt. \tag{2.21}$$

As amostras z_k são as saídas amostradas do filtro casado com $h(t)$, dadas por

$$z_k = \int_{-\infty}^{\infty} y(t) h^*(t - kT) dt. \tag{2.22}$$

Esse filtro em conjunto com o amostrador é conhecido como filtro casado amostrado, como mostra a Figura 2.13.

Uma característica importante desse receptor de distância mínima é que o sinal contínuo no tempo $y(t)$ é transformado em um sinal recebido em tempo discreto z_k, cuja taxa de amostragem coincide com a taxa de símbolo. A representação em tempo discreto do sinal recebido é então processada para decidir pela seqüência completa de símbolos $\{S_k, 1 \leq k \leq K\}$.

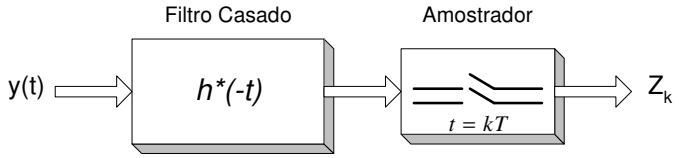

Figura 2.13: "Front end receiver" formado por um filtro casado seguido de um amostrador.

Esse resultado pode ser estendido para canais com ruído colorido. Neste caso, a saída do demodulador é primeiro "branqueada" pelo filtro $\frac{1}{\sqrt{S_N[j(\omega+\omega_c)]}}$, em que $S_N\left[j(\omega+\omega_c)\right]$ é a densidade espectral de potência (PSD) do ruído. De forma equivalente, o filtro casado pode ser substituído por um filtro casado com $h(t)$ normalizado por $S_N\left[j(\omega+\omega_c)\right]$.

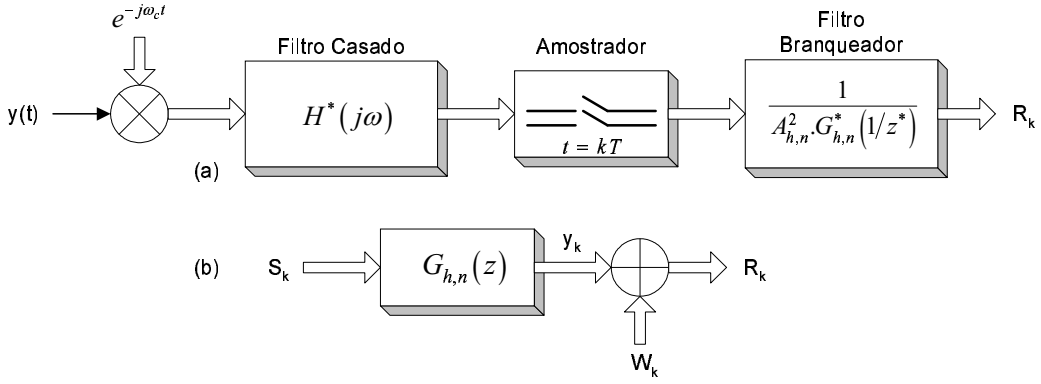

Figura 2.14: Filtro casado branqueador (WMF) para PAM e ruído colorido. (a) "Front end" completo do receptor; (b) modelo equivalente em tempo discreto.

Como resultado, tem-se o "front end receiver" mostrado na Figura 2.14(a).

As amostras de ruído na saída do amostrador não são brancas, tendo densidade espectral de potência

$$S_{h,n}(e^{j\omega T}) = \frac{1}{T}\sum_{m=-\infty}^{\infty}\left|H\left[j\left(\omega + m\frac{2\pi}{T}\right)\right]\right|^2. \tag{2.23}$$

Pode-se mostrar [12] que se pode obter amostras de ruído branco utilizando-se o filtro branqueador mostrado na Figura 2.14(a), onde o termo $G_h, n(z)$ é obtido da fatoração do espectro da Equação 2.23, isto é,

$$S_{h,n}(z) = A_{h,n}^2 G_{h,n}(z) G_{h,n}^*(z) \tag{2.24}$$

onde $A_{h,n}^2$ é uma constante positiva e $G_{h,n}(z)$ é uma função de transferência mônica de fase mínima.

Na Figura 2.14(a), um filtro branqueador de fase máxima é adicionado à saída do filtro amostrado casado, levando a um processo de ruído de saída, W_k, que é branco, Gaussiano e circularmente simétrico, com variância $1/A_{h,n}^2$. A Figura 2.14(b) mostra o modelo de canal equivalente em tempo discreto.

Pode-se também mostrar [12] que a saída do "front end" descrito na Figura 2.14(a) é uma estatística suficiente, ou seja, qualquer que seja a regra de decisão implementada (inclusive uma ótima), não há a necessidade de se obter outras observações além desta saída.

2.7 Cálculo da Distância Mínima para um Canal com IIS

Um parâmetro relacionado ao desempenho de qualquer algoritmo de detecção para um canal com IIS é a distância mínima entre possíveis seqüências na saída do canal [12].

Seja um símbolo de erro ε_k dado por $\varepsilon_k = S_k - \tilde{S}_k$, onde S_k e $\tilde{S}_k$ são possíveis símbolos. A distância mínima é dada por [12]

$$d_{\min}^2 = \min_{\{\varepsilon_k, 1 \le k \le K\}} \sum_{m=1}^{\infty} \left| \sum_{k=1}^{K} \varepsilon_k g_{h,m-k} \right|^2 \tag{2.25}$$

onde $g_{h,k}$ é a transformada inversa de $G_{h,n}$ e a minimização ocorre ao longo de todas

as seqüências não nulas de erros de símbolos, isto é, ao menos um dos símbolos de erro deve ser diferente de zero.

No caso em que $G_h(z)$ é um filtro FIR, a Eq. 2.25 pode ser reformulada de maneira que o algoritmo de Viterbi possa encontrar a solução. Assumindo-se $g_{h,k} = 0$ para $k > M$, pode-se inverter o somatório, obtendo-se

$$d^2_{\min} = \min_{\{\varepsilon_k, 1 \leq k \leq K\}} \sum_{m=1}^{K+M} \left| \sum_{i=0}^{M} g_{h,i} \varepsilon_{m-i} \right|^2 . \tag{2.26}$$

Dessa forma, o somatório em m torna-se finito, embora deseje-se valores altos de K. Além disso, a minimização passa agora a ser uma minimização das métricas dos caminhos da treliça. Definindo-se os estados

$$\psi_k = [\varepsilon_{k-1}, \varepsilon_{k-2}, ..., \varepsilon_{k-M}]$$

e um diagrama de treliça para a progressão dos estados e nomeando-se cada ramo dessa treliça com a respectiva métrica $\left| \sum_{i=0}^{M} g_{h,i} \varepsilon_{k-i} \right|^2$, a solução passa a ser calcular a métrica do caminho não nulo de menor métrica.

Esta nova maneira de se tratar o problema tem como vantagem o fato de que a métrica do ramo é uma função de cada ramo isoladamente e não de pares de ramos (caminho correto e caminho em erro), o que reduz consideravelmente o número de alternativas a serem consideradas. Em contrapartida, o alfabeto de símbolos de erros é composto por todos os inteiros no intervalo $[-(Q-1), (Q-1)]$ ou $2Q-1$ valores distintos, onde Q é a cardinalidade do alfabeto.

3

Algoritmo de Equalização Turbo

3.1 Introdução

O Capítulo anterior expôs alguns conceitos da Teoria de Informação e Codificação aplicados em sistemas de comunicações digitais. Neste Capítulo, será apresentado o sistema de transmissão e recepção implementado, com ênfase na decodificação iterativa do código convolucional utilizado e do canal com IIS.

Considera-se que o receptor faz detecção coerente, conhece os coeficientes do canal e o intervalo de símbolo T. Dessa forma, o canal pode ser aproximado por um modelo em banda base discreto no tempo, como mostra a Figura 3.1, onde o filtro do transmissor, o canal e o filtro casado branqueador do receptor podem ser

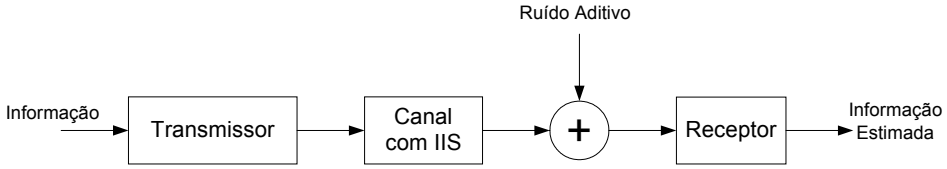

Figura 3.1: Representação de um sistema de transmissão.

representados por um filtro linear com resposta impulsiva finita

$$h[i] = \sum_{k=0}^{M} h_{k,i}\delta\left[i-k\right],$$

onde i é o índice de tempo.

Assume-se que o canal utilizado é invariante no tempo, levando a coeficientes constantes

$$h_{k,i} = h_k, \ \forall \text{ i.}$$

Num caso mais geral os coeficientes são desconhecidos e variantes em relação a i. Há então a necessidade de utilização de algoritmos adaptativos para se obter estimativas dos coeficientes do canal.

Além disso, os coeficientes da resposta ao impulso do canal e as amostras de ruído W_k são reais. A resposta ao impulso do canal é um sinal de energia e satisfaz

$$E_h = \sum_{k=0}^{M} h_k h_k^* = 1.$$

O modelo de canal utilizado foi extraído de [13], e é dado por

$$h_c[i] = 0,227\delta[i] + 0,46\delta[i-1] + 0,688\delta[i-2] + 0,46\delta[i-3] + 0,227\delta[i-4]. \quad (3.1)$$

A Figura 3.2 mostra a densidade espectral de potência (PSD, do inglês *Power Spectrum Density*) deste canal. Percebe-se, pela Equação 3.1, que os símbolos transmitidos num dado instante são bastante influenciados pelos símbolos transmitidos nos instantes anteriores, alterando os níveis de amplitude do sinal reconhecidos pelo receptor e dificultando a detecção correta desses símbolos.

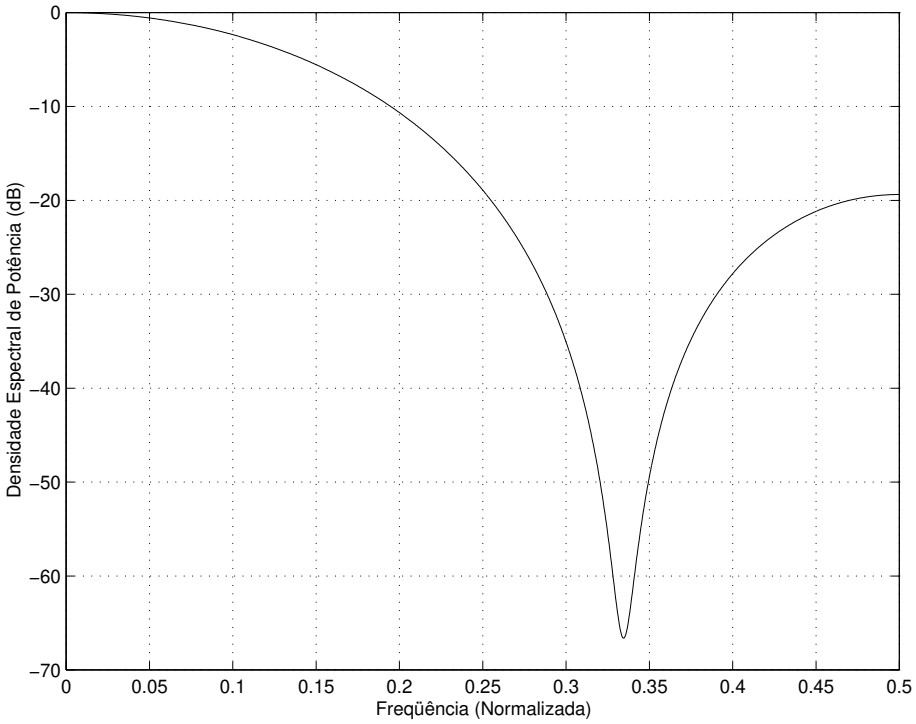

Figura 3.2: Densidade espectral de potência do canal dado pela Eq. 3.1.

O objetivo de sua escolha foi justamente testar o desempenho da aplicação do algoritmo SISO (*Soft-Input Soft-Output*), apresentado em [14], a canais com forte IIS. Comparações do desempenho de algoritmos de recepção aplicados a este canal podem ser vistas em [13] e [15].

3.2 Sistema Implementado

O sistema implementado está representado na Figura 3.3. Os módulos decodificadores SISO serão detalhados na Seção 3.3.1.

A fonte gera símbolos u_k, $k = 1, 2, ..., K$, sendo K o comprimento do bloco de informação. Os símbolos u_k pertencem a um alfabeto binário $\mathcal{U} = \{0, 1\}$, com igual probabilidade de ocorrência. Esses símbolos (ou bits) são codificados por um codificador convolucional não-sistemático, não-recursivo, com matriz geradora

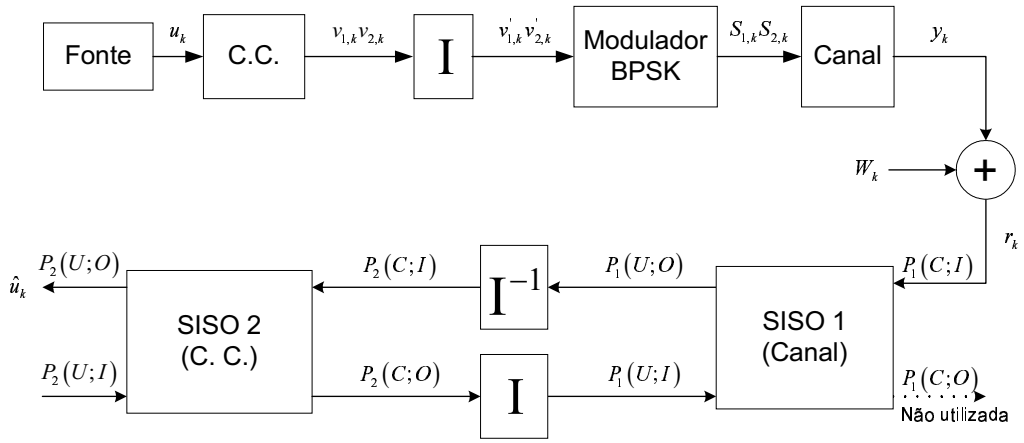

Figura 3.3: Sistema de transmissão/recepção utilizando concatenação serial e decodificação iterativa.

$G = [101, 111]$. Sabe-se que para valores grandes de E_b/N_0, o desempenho de um codificador convolucional não-sistemático é melhor que o de um sistemático de mesma memória e para valores pequenos de E_b/N_0, geralmente ocorre o oposto [16]. A utilização de codificadores recursivos sistemáticos leva a um melhor desempenho para quaisquer valores de E_b/N_0 [16]. Apesar disso, foge do escopo deste trabalho o projeto de codificadores que levem ao melhor desempenho. Deseja-se apenas verificar o desempenho de um código simples num canal com IIS. Para tal tarefa, o codificador convolucional utilizado é adequado e de simples implementação.

Os símbolos $v_{i,k}$, $i = 1, 2$ e $k = 1, 2, ..., K$, são agora entrelaçados, bit a bit, por um entrelaçador aleatório de comprimento $2K$. O fato de se ter um entrelaçador com mesmo comprimento do bloco codificado facilita sua implementação e permite que cada bloco seja decodificado no momento em que chega, sem a necessidade de se aguardar os demais. O entrelaçador gera uma seqüência totalmente aleatória para cada bloco codificado de comprimento $L = 2K$ transmitido. A matriz de permutação gerada na transmissão é conhecida pelo receptor, onde é realizado o desentrelaçamento do bloco.

Os bits codificados e entrelaçados são, então, modulados por um modulador BPSK (*Binary Phase Shift Keying*) que gera símbolos S_k da constelação $\mathcal{A} = \{-A, +A\}$.

Os símbolos da saída do modulador, S_k, passam pelo canal, com equação recursiva dada por 3.1. Aos símbolos gerados são adicionadas amostras W_k de ruído AWGN com média nula e variância $\sigma^2 = N_0/2$.

Para o cálculo de E_b/N_0 considera-se σ^2 e E_h de valores unitários. Dessa forma, E_b/N_0 é dada por

$$\frac{E_b}{N_0} \triangleq \frac{\mathrm{E}\left[|y_k|^2\right] E_h}{RN_0} = \frac{A^2 E_h}{2\sigma^2 R}. \tag{3.2}$$

Como a taxa do código utilizado é $R = 1/2$ e a energia E_h é unitária, tem-se que

$$\frac{E_b}{N_0} = A^2.$$

Para obter-se os diferentes valores de E_b/N_0 basta variar a amplitude do sinal $\{-A, +A\}$, referente à energia média gasta por bit, E_b, enquanto a variância do ruído σ^2 permanece constante.

3.3 Decodificador

O decodificador deve produzir estimativas, $\hat{u}_k$, do bit de informação transmitido, u_k, baseado na seqüência recebida, $\mathbf{r} = (r_1, r_2, ..., r_{nK})$. Como a cada bit de informação u_k na entrada do codificador convolucional tem-se uma palavra-código v_k de comprimento n, o bit estimado $\hat{u}_k = u_k$ se e somente se $\hat{v}_k = v_k$. Caso contrário, há erro na decodificação.

Um decodificador que minimiza a probabilidade de erro, $P(E)$, chamado decodificador ótimo, deve então minimizar $P(\hat{v}_k \neq v_k|\mathbf{r})$, ou seja, minimizar o número de erros de decodificação dada a seqüência recebida $\mathbf{r}$, para toda seqüência $\mathbf{r}$. Como $P(v_k|\mathbf{r}) = \frac{P(\mathbf{r}|v_k)P(v_k)}{P(\mathbf{r})}$ e $P(\mathbf{r})$ é independente da regra de decodificação, minimizar

$P(\hat{v}_k \neq v_k|\mathbf{r})$ é equivalente a maximizar $P(\mathbf{r}|v_k)P(v_k)$. Um decodificador que realiza a maximização de $P(\mathbf{r}|v_k)$ é chamado de decodificador de máxima verossimilhança (ML). Um caso particular ocorre quando todas as palavras-código são eqüiprováveis. Nesse caso, basta ao decodificador maximizar $P(\mathbf{r}|v_k)$ e o decodificador ML é ótimo. Esse decodificador minimiza a probabilidade de erro de bit. Um exemplo de decodificador assim é encontrado em [17], onde Bahl e outros propuseram um decodificador ótimo, o algoritmo BCJR, que minimiza a probabilidade de erro de bit para códigos lineares.

Um exemplo de decodificador ML é o algoritmo de Viterbi, que minimiza a probabilidade de erro de seqüência para códigos convolucionais. Este algoritmo, entretanto, não necessariamente minimiza a probabilidade de erro de símbolo (ou bit) [13].

O decodificador utilizado neste trabalho foi proposto em [14]. Trata-se de um módulo de decodificação SISO, que atualiza as distribuições de probabilidade dos bits de informação ou símbolos codificados baseado na informação do canal de transmissão. É uma modificação do algoritmo original, BCJR, apresentando melhorias significativas. Dentre elas, pode-se citar: decodificação contínua de códigos concatenados sem a necessidade de terminação das treliças dos códigos constituintes; aplicação em códigos de bloco ou convolucionais, sistemáticos ou não sistemáticos, para símbolos M-*ários* e não só binários; em esquemas concatenados, é capaz de lidar com códigos com taxas superiores a um. Este decodificador pode ser utilizado em várias aplicações, como decodificação MAP símbolo a símbolo de um único código ou ainda na decodificação de dois ou mais códigos concatenados, separados por entrelaçadores, como em esquemas de decodificação iterativa.

3.3.1 O algoritmo SISO

No que segue, letra maiúscula indica variável aleatória (VA) e letra minúscula a realização da VA.

O módulo SISO é um dispositivo com duas entradas e duas saídas conforme mostra a Figura 3.4. Ele recebe como entrada as seqüências de distribuição de probabilidade $P(u;I)$ e $P(c;I)$, gerando na saída as seqüências $P(u;O)$ e $P(c;O)$, calculadas de acordo com as respectivas entradas e com o conhecimento da seção da treliça do código.

Figura 3.4: Módulo SISO.

As distribuições de probabilidade de entrada, $\mathbf{P}(u;I)$ e $\mathbf{P}(c;I)$ indicam, respectivamente, as probabilidades $\mathbf{P}(u) = [P_k(u)]$ para os bits de informação e $\mathbf{P}(c) = [P_k(c)]$ para os bits codificados, sendo $P_k(u) \triangleq \mathrm{P}(U_k = u)$ e $P_k(c) \triangleq \mathrm{P}(C_k = c)$.

A treliça de um código convolucional invariante no tempo é caracterizada por uma única seção, que descreve a transição entre estados da treliça nos instantes $k-1$ e k. A Figura 3.5 representa uma seção de treliça, em que $E^i(r)$ e $E^f(r)$ indicam estado inicial e estado final do ramo r, respectivamente.

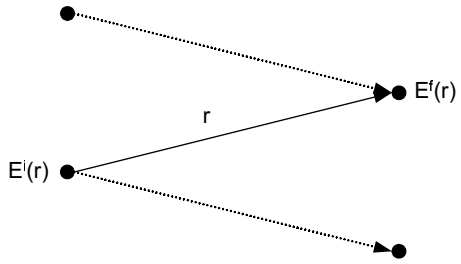

Figura 3.5: Seção da treliça do codificador.

A primeira parte do algoritmo realiza o cálculo das quantidades A_k e B_k, obtidas de forma similar a α e β no algoritmo BCJR, através de recursão direta e inversa,

respectivamente:

$$A_k(e) = \sum_{r:e^f(r)=e} A_{k-1}\left[e^i(r)\right] P_k\left[u(r);I\right] P_k\left[c(r);I\right],\ k = 1,2,...,n-1 \tag{3.3}$$

e

$$B_k(e) = \sum_{r:e^i(r)=e} B_{k+1}\left[e^f(r)\right] P_{k+1}\left[u(r);I\right] P_{k+1}\left[c(r);I\right],\ k = n-1,...,2,1. \tag{3.4}$$

Antes de se calcular 3.3 e 3.4, deve-se atribuir valores iniciais

$$A_0(e) = \begin{cases} 1,\ e = E_0 \\ 0,\ \text{caso contrário} \end{cases}$$

e

$$B_n(e) = \begin{cases} 1,\ e = E_n \\ 0,\ \text{caso contrário} \end{cases}.$$

Essa inicialização de A e B garante que a treliça, para um bloco, iniciará e terminará no estado todo nulo.

Tendo-se A_k e B_k, calcula-se as distribuições de probabilidades

$$\tilde{P}_k(u;O) = \tilde{H}_u \sum_{r:u(r)=u} A_{k-1}\left[e^i(r)\right] P_k\left[u(r);I\right] P_k\left[c(r);I\right] B_k\left[e^f(r)\right] \tag{3.5}$$

e

$$\tilde{P}_k(c;O) = \tilde{H}_c \sum_{r:c(r)=c} A_{k-1}\left[e^i(r)\right] P_k\left[u(r);I\right] P_k\left[c(r);I\right] B_k\left[e^f(r)\right]. \tag{3.6}$$

Os valores $\tilde{H}_u$ e $\tilde{H}_c$ são constantes de normalização que, a cada seção da treliça, fazem

$$\tilde{H}_u \rightarrow \sum_u \tilde{P}_k(u;O) = 1$$

e

$$\tilde{H}_c \rightarrow \sum_c \tilde{P}_k(c;O) = 1.$$

Os valores de $P_k\left[c(r);I\right]$ e $P_k\left[u(r);I\right]$ são constantes em relação às variáveis dos somatórios nas Eq. 3.5 e 3.6, respectivamente. Define-se, então, os valores

$$P_k(u;O) \triangleq H_u \frac{\tilde{P}_k(u;O)}{P_k(u;I)}$$

e

$$P_k(c;O) \triangleq H_c \frac{\tilde{P}_k(c;O)}{P_k(c;I)}$$

e as respectivas constantes de normalização

$$H_u \rightarrow \sum_u P_k(u;O) = 1$$

e

$$H_c \rightarrow \sum_c P_k(c;O) = 1,$$

que podem ser obtidas de

$$P_k(u;O) = H_u \tilde{H}_u \sum_{r:u(r)=u} A_{k-1}\left[e^i(r)\right] P_k\left[c(r);I\right] B_k\left[e^f(r)\right] \tag{3.7}$$

e

$$P_k(c;O) = H_c \tilde{H}_c \sum_{r:c(r)=c} A_{k-1}\left[e^i(r)\right] P_k\left[u(r);I\right] B_k\left[e^f(r)\right]. \tag{3.8}$$

As distribuições de probabilidade de saída $P_k(u;O)$ e $P_k(c;O)$ são versões atualizadas das distribuições de probabilidade de entrada, $P_k(u;I)$ e $P_k(c;I)$, respectivamente. Na literatura de "decodificação turbo", tais valores são chamados de *informação extrínseca* e representam a contribuição do algoritmo SISO às distribuições de probabilidade *a priori*, $P_k(u;I)$ e $P_k(c;I)$. A utilização das Eq. 3.7 e 3.8 simplifica a complexidade dos blocos em esquemas iterativos para decodificação de códigos concatenados, se comparada às Eq. 3.5 e 3.6.

Para esquemas concatenados nos quais a saída está ligada ao canal ao invés de estar ligada a outro codificador, a Eq. 3.8 não precisa ser calculada [14], como pode ser visto na Figura 3.3.

Como cada símbolo codificado C_k é formado por n_0 bits $C_k^j,\ j = 1, 2, ..., n_0$ com realização $c^j \in \{0, 1\}$, à seqüência de símbolos codificados associa-se a seqüência

de distribuições de probabilidade *a priori* $P(c;I) = P_k(c;I) = \prod_{j=1}^{n_0} P_k(c^j;I)$. As distribuições de probabilidade dos símbolos codificados podem ser representadas como produto das distribuições marginais dos bits, desde que seja utilizado um entrelaçador de bit ao invés de um entrelaçador de símbolo no esquema de decodificação iterativa para códigos concatenados.

No sistema considerado na Figura 3.3, o primeiro módulo SISO está relacionado ao canal, que funciona como um codificador em treliça de taxa unitária. A entrada codificada depende, no SISO 1, das seqüências recebidas, r_k, referentes aos símbolos codificados pela equação recursiva do canal adicionados de amostras de ruído W_k. Dessa forma, para se calcular $P_k\left[c(r);I\right]$, deve-se considerar a distribuição condicional da Eq. 2.15. Adaptando para o sistema proposto, tem-se

$$P_k(c;I) \propto \mathrm{P}\left[r_k|Y_k = y\right] = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left[\frac{-(r_k-y_k)^2}{2\sigma^2}\right]}. \tag{3.9}$$

Para a primeira iteração, o SISO 1 recebe como entrada valores eqüiprováveis. A partir da segunda iteração, as distribuições $P_k\left[u(r);I\right]$ de entrada no SISO 1 são geradas pelo SISO 2. As distribuições de saída $P_k\left[c(r);O\right]$ não são calculadas nesse módulo, como foi dito anteriormente.

Para o segundo SISO, referente ao codificador convolucional, a entrada $P_k\left[c(r);I\right]$ é dada pela saída $P_k\left[u(r);O\right]$, após passar pelo desentrelaçador. Esse módulo recebe como entrada $P_k\left[u(r);I\right]$ uma distribuição de probabilidades eqüiprovável, dada a característica da fonte de informação.

Enquanto o SISO 1 encontra as distribuições de probabilidade *a posteriori* (APP) baseado nas informações do canal, o SISO 2 atua como decodificador MAP, decidindo pela informação que tenha maior probabilidade de ocorrência, dado que recebeu do SISO 1 as distribuições de probabilidade atualizadas. No SISO 2, saída $P_k\left[u(r);O\right]$ é utilizada pela regra MAP para estimar o bit mais provável e a saída $P_k\left[c(r);O\right]$ realimenta o módulo SISO 1 com a informação extrínseca processada no SISO 1.

Esse processo se repete a cada iteração. Em termos gerais, quanto maior o

número de iterações, maior o ganho de codificação. Em contrapartida, maior o tempo de processamento exigido. Um comportamento típico de sistemas com decodificação iterativa é que o ganho de codificação diminui gradativamente de iteração para iteração, até atingir um ponto de saturação. Deve-se, assim, observar qual o número médio de iterações necessárias para que o sistema sature.

É imprescindível que, para cada bloco recebido, seja realizada a terminação da treliça. Isso se faz adicionando-se M zeros à seqüência recebida, garantindo-se que a treliça retornará ao estado inicial.

3.4 Simulações

3.4.1 Metodologia

Todas as simulações realizadas foram feitas utilizando-se o ambiente MATLAB. Para tal, foram desenvolvidos códigos-fonte específicos e não foram utilizadas as rotinas já existentes.

O critério para a obtenção das BER's inerentes a cada sistema foi o de no mínimo se transmitir dez vezes o valor do inverso do erro encontrado. Ou seja, para uma BER de 10^{-4}, deve-se transmitir no mínimo 10^5 bits. Dessa forma, garante-se um erro médio satisfatório e dificulta a ocorrência de resultados muito destoantes da realidade.

3.4.2 Resultados

Inicialmente, um teste foi realizado utilizando-se um canal com IIS e de memória curta. O canal escolhido foi retirado de [12] e é dado por $h_{teste}[i] = \delta[i] + 0,5\delta[i-1]$. O bloco de informação utilizado foi de 100 bits e, conseqüentemente, o entrelaçador tinha comprimento de 200 bits.

Como pode ser observado na Figura 3.6, o sistema de decodificação iterativo

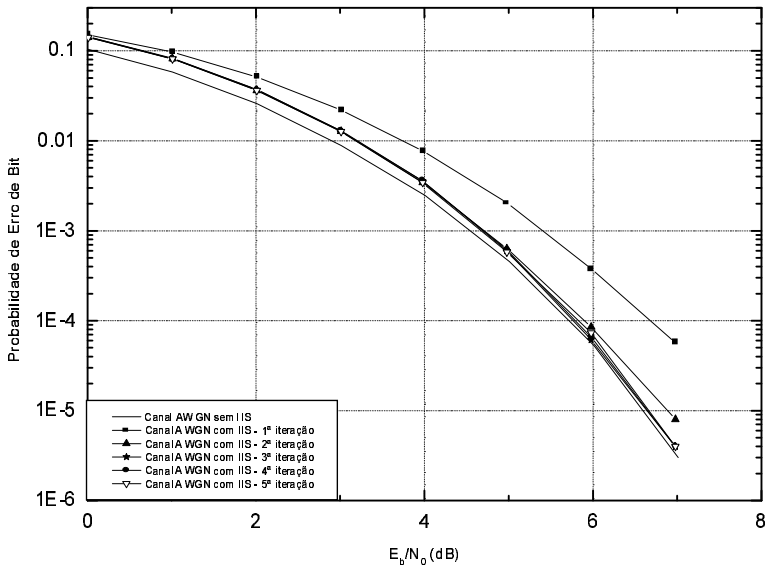

Figura 3.6: Probabilidade de Erro de Bit *versus* E_b/N_0 para o canal $h_{teste}[i] = \delta[i] + 0,5\delta[i-1]$, com bloco de informação de 100 bits.

para o canal testado aproxima-se rapidamente de um canal AWGN sem IIS, com apenas três iterações.

Em seguida, foi realizado um teste variando-se o comprimento do entrelaçador de bit a fim de se chegar a um valor que fornecesse um bom desempenho em termos de probabilidade de erro de bit. Assim, fixando-se a relação E_b/N_0 em 6 dB e variando-se o comprimento do bloco, para uma, duas, três e quatro iterações, obteve-se a curva da Figura 3.7.

Observa-se, na Figura 3.7, que o comprimento do bloco influencia o desempenho da probabilidade de erro de bit. Para blocos de comprimento maior, principalmente para a quarta iteração, a diferença é considerável.

O desempenho para blocos de informação com comprimento superiores a 16000 bits (entrelaçador de 32000 bits) se mostrou praticamente o mesmo que para blocos de 16000 bits.

Como a complexidade e, conseqüentemente, o tempo de processamento aumenta

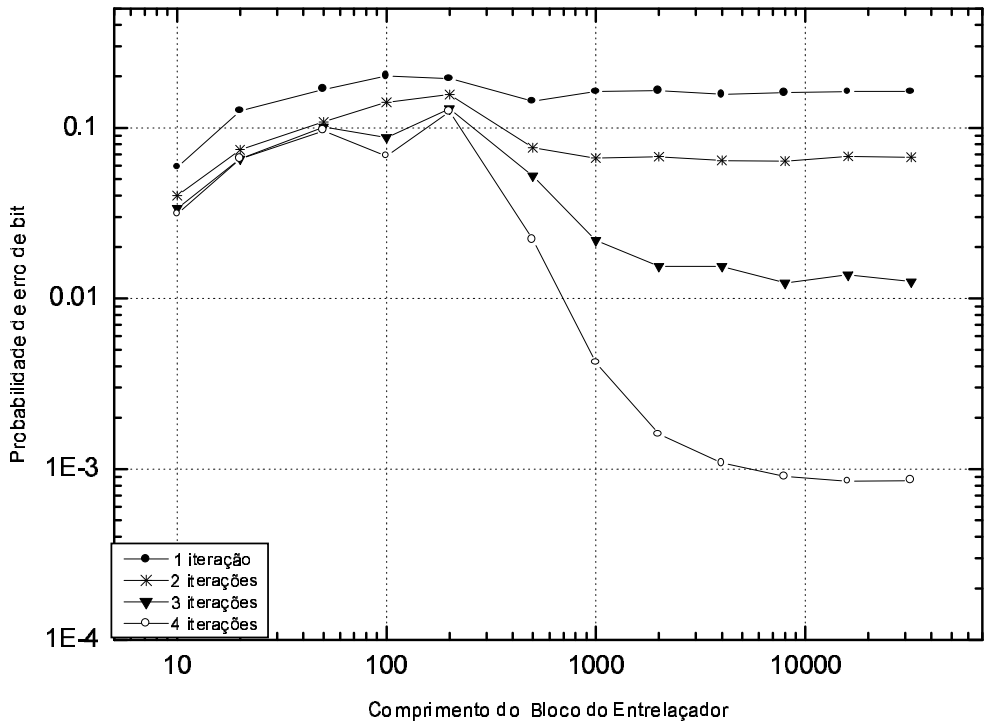

Figura 3.7: Probabilidade de Erro de Bit *versus* Comprimento do bloco do entrelaçador para uma relação $\frac{E_b}{N_0} = 6$ dB.

exponencialmente com o aumento do comprimento do bloco, optou-se por blocos de informação de 16000 bits para as simulações deste e do próximo Capítulo.

A Figura 3.8 mostra os resultados da simulação realizada para o bloco de informação de 16000 bits. Observa-se que para 14 iterações, o desempenho do canal com IIS se aproxima do desempenho de um canal sem IIS.

Para fins de comparação, foi realizada também a simulação para blocos de informação de 1000 bits. O resultado é mostrado na Figura 3.9.

Como pode ser observado, enquanto o sistema com bloco de informação de 16000 bits se aproxima da curva de um canal AWGN sem IIS em 5 dB, o sistema com bloco de 1000 se aproxima da canal AWGN sem IIS somente em 6 dB. Para o bloco maior são necessárias quatorze iterações para atingir o ponto de saturação e para o menor,

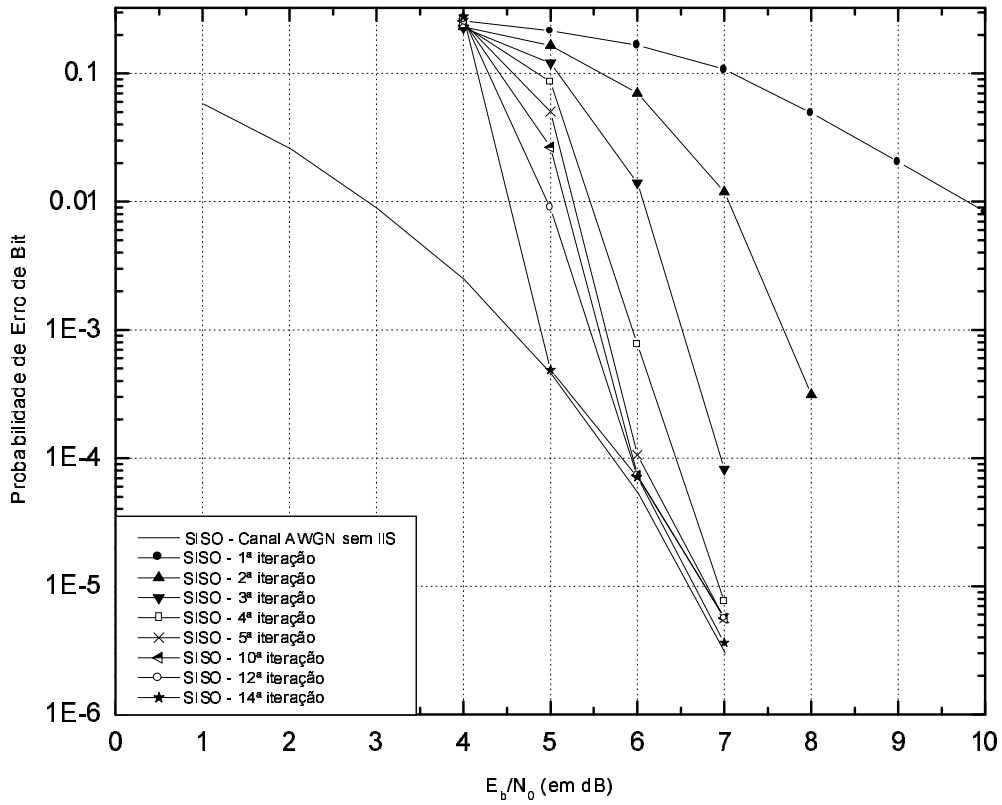

Figura 3.8: Probabilidade de Erro de Bit *versus* E_b/N_0 para o canal da Eq. 3.1 e bloco de informação de 16000 bits.

sete iterações.

Embora o sistema implementado tenha conseguido convergir para um desempenho bastante satisfatório, a complexidade de implementação, refletindo em gasto excessivo com tempo de processamento, faz com que tal sistema, em alguns casos, tenha sua aplicação restrita a blocos menores. No próximo capítulo, serão testados três métodos de redução de estados na treliça calculada pelo receptor. Trata-se de alternativas para se diminuir a complexidade do receptor, tendo como conseqüência a perda no desempenho, mas permitindo ao sistema maior flexibilidade para se escolher um receptor que melhor atenda às necessidades reais.

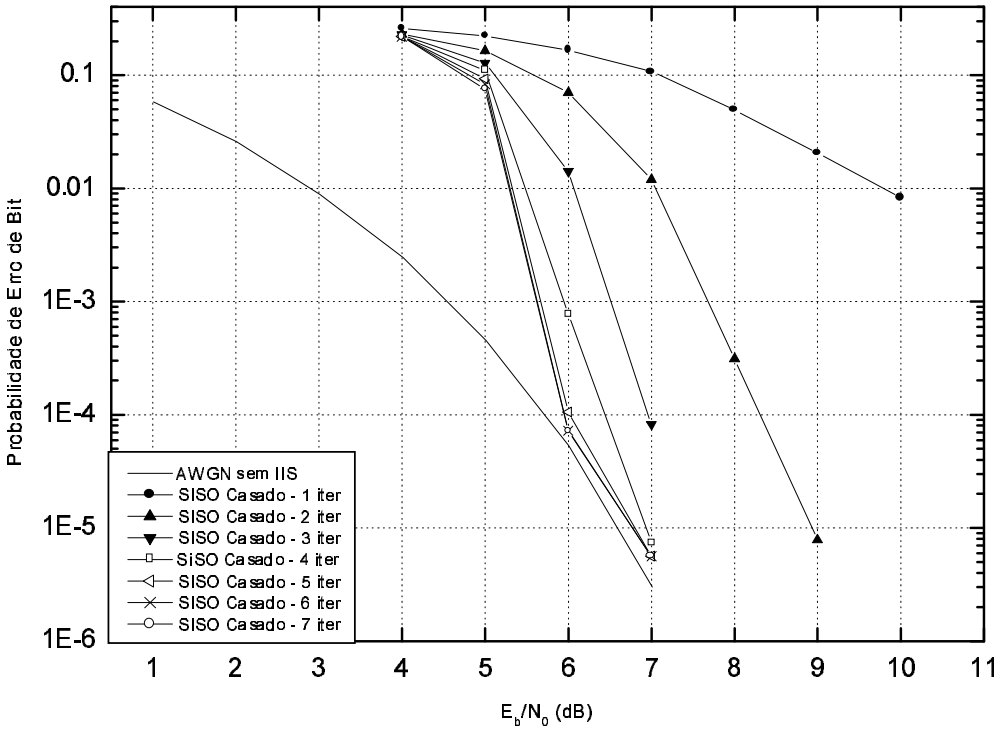

Figura 3.9: Probabilidade de Erro de Bit *versus* E_b/N_0 para o canal da Eq. 3.1 e bloco de informação de 1000 bits.

4

Técnicas de Redução de Estados

4.1 Introdução

O sistema apresentado no capítulo anterior se mostrou eficiente quando aplicado a um canal AWGN com a presença de IIS, se comparado ao canal AWGN sem IIS. Entretanto, em termos de complexidade, a quantidade de esforço computacional exigido na recepção é bastante grande e, em alguns casos, limita as taxas de transmissão ao tempo de processamento dos blocos recebidos.

Neste capítulo serão abordados três métodos de redução de estados da treliça dos decodificadores diretamente relacionados ao canal em questão. Embora menos eficientes que o algoritmo SISO apresentado no Capítulo 3, esses métodos são alter-

nativas que reduzem a complexidade na decodificação. Essas alternativas se fazem necessárias em alguns casos onde o tempo de processamento na decodificação é o fator que restringe as aplicações do sistema a ser desenvolvido.

Na descrição do algoritmo MAP do capítulo anterior, considerou-se um casamento perfeito entre os filtros de transmissão e recepção. Neste capítulo, para os métodos de redução de estados baseados no truncamento dos coeficientes do canal e na estimação espectral será forçado um descasamento do filtro de recepção e avaliado o compromisso entre desempenho e complexidade de implementação. No terceiro método, o algoritmo M, não há descasamento dos filtros de transmissão e recepção e sim um uma redução nos cálculos dos estados da treliça.

4.2 Truncamento de coeficientes do canal

O método de truncamento consiste tão somente em desconsiderar, no receptor, a influência das amostras de símbolos geradas em função de um ou mais *taps* de memória do canal. Seja o canal considerado no capítulo anterior:

$$h_c[i] = 0,227\delta[i] + 0,46\delta[i-1] + 0,688\delta[i-2] + 0,46\delta[i-3] + 0,227\delta[i-4]. \quad (4.1)$$

Como este canal tem quatro *taps* de memória (instantes anteriores) mais um instante atual, para um tipo de modulação BPSK, tem-se, ao todo, $2^4 = 16$ estados na treliça do decodificador. Cada elemento de memória desconsiderado na recepção reduz pela metade o número de estados na treliça. Assim, retirando-se uma, duas ou três memórias tem-se, no decodificador uma treliça de oito, quatro e dois estados, respectivamente.

A Figura 4.1 mostra a densidade espectral de potência do canal estudado, $h_c[i]$, como também as respostas ao impulso dos canais truncados, $h_{c3}[i]$, $h_{c2}[i]$ e $h_{c1}[i]$, referentes a retirada de uma, duas e três memórias, respectivamente.

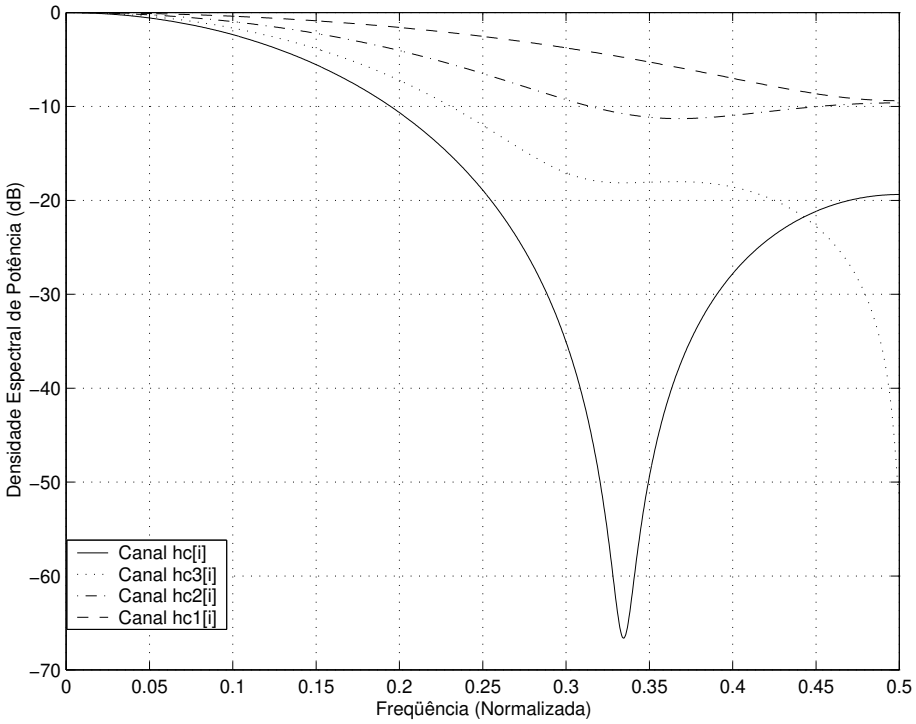

Figura 4.1: Densidade espectral de potência do canal.

Como pode ser observado pela Figura 4.1, a retirada de um ou mais *taps* de memória altera bastante a função densidade espectral de potência. A Tabela 4.1 mostra o erro médio quadrático das PSD dos canais $h_{c3}[i]$, $h_{c2}[i]$ e $h_{c1}[i]$ em relação à PSD do canal estudado. O erro relativo ao canal $h_{c3}[i]$ é da ordem de $0,10065$.

Canal	Erro
$h_{c1}[i]$	2,27631
$h_{c2}[i]$	0,78866
$h_{c3}[i]$	0,10065

Tabela 4.1: Erro das densidades espectrais de potência dos canais truncados em relação à do canal estudado.

Essa alteração vai refletir numa diferença considerável entre os símbolos do alfa-

beto transmitido e os símbolos do novo alfabeto reconhecido pelo receptor descasado. Essa diferença, para os casos onde são retirados dois e três elementos de memória, degrada o desempenho de tal forma que o receptor passa a decidir por símbolos errados em quase 50 por cento das vezes, independentemente do aumento de E_b/N_0, tornando-se, assim, impraticável a sua aplicação.

Como os símbolos gerados pelo transmissor, na modulação BPSK não se alteram, já que tanto o transmissor quanto o canal permanecem os mesmos, o cálculo da relação E_b/N_0 também não se altera, sendo dado pela Eq. 3.2.

O receptor tem a tarefa de decodificar os símbolos transmitidos, com ruído AWGN e IIS com memória $M = 4$ e um alfabeto diferente do transmitido, já que transmissor, canal e receptor estão descasados. Para tal tarefa foi desenvolvido o receptor baseado nos níveis de decisão do canal truncado com 8 estados. A intenção era verificar o efeito turbo do algoritmo iterativo em tal situação.

4.2.1 Resultados

O Gráfico 4.2 mostra os resultados da simulação do sistema implementado para um receptor descasado com 8 estados, $h_{c3}[i]$. No mesmo gráfico estão também as curvas para o canal AWGN puro e para o canal com IIS casado, obtidos no Capítulo anterior. Nota-se que, apesar do descasamento e do erro inserido pelo próprio receptor, há um ganho razoável de codificação da primeira à décima iteração, denotando que o princípio turbo é verificado nesse sistema.

4.3 Estimação de Canal

4.3.1 Modelagem Paramétrica

A estimação espectral, nesse contexto, é um procedimento com três passos. Primeiro, escolhe-se um modelo adequado. Após, os parâmetros do modelo as-

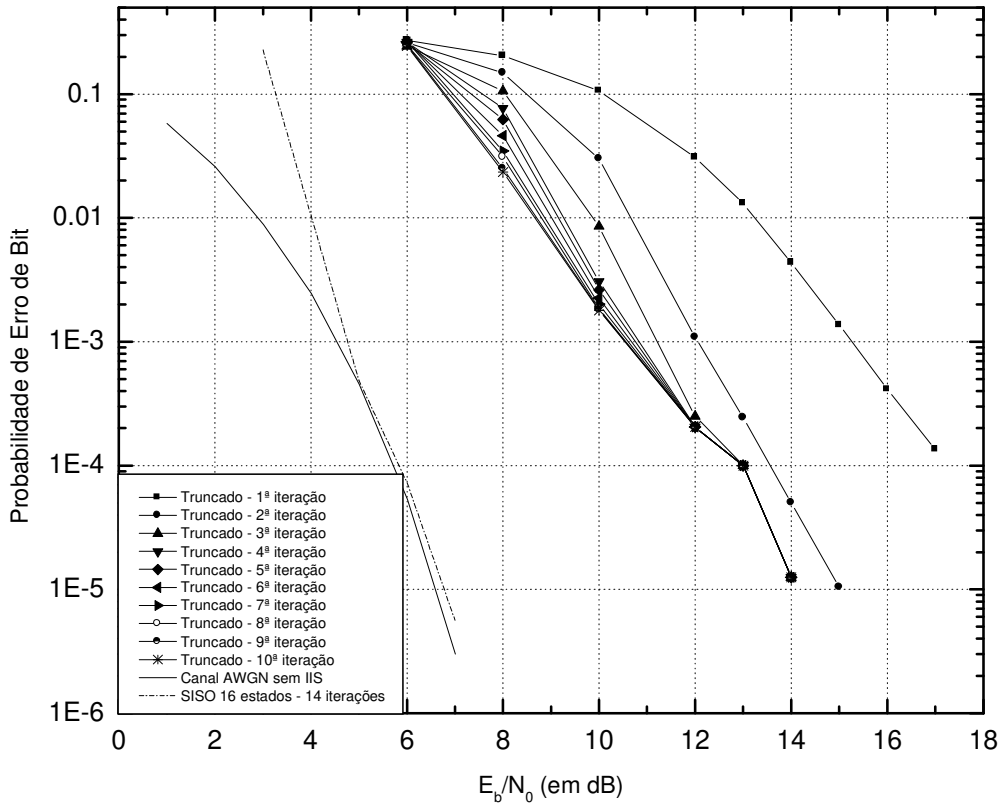

Figura 4.2: P_b x E_b/N_0 utilizando o algoritmo truncado para 8 estados.

sumido são estimados, utilizando-se as amostras disponíveis dos dados. Por último, obtém-se o espectro estimado, substituindo-se os parâmetros do modelo estimado na densidade espectral de potência teórica da qual se obteve os parâmetros.

Os modelos adotados são séries temporais ou funções de transferência racionais, podendo ser autoregressivo (AR), de média móvel (MA) ou autoregressivo de média móvel (ARMA). Na prática, muitos processos aleatórios podem ser aproximados por um modelo de série temporal ou função de transferência racional [18]. Nesse modelo, uma seqüência piloto de entrada, u[n], e a seqüência de saída, x[n], estão relacionadas pela equação linear

$$x[n] = -\sum_{k=1}^{p} a[k]x[n-k] + \sum_{k=0}^{q} b[k]u[n-k]. \tag{4.2}$$

Este modelo, mais geral, é chamado de modelo ARMA e é mostrado na Figura 4.3.

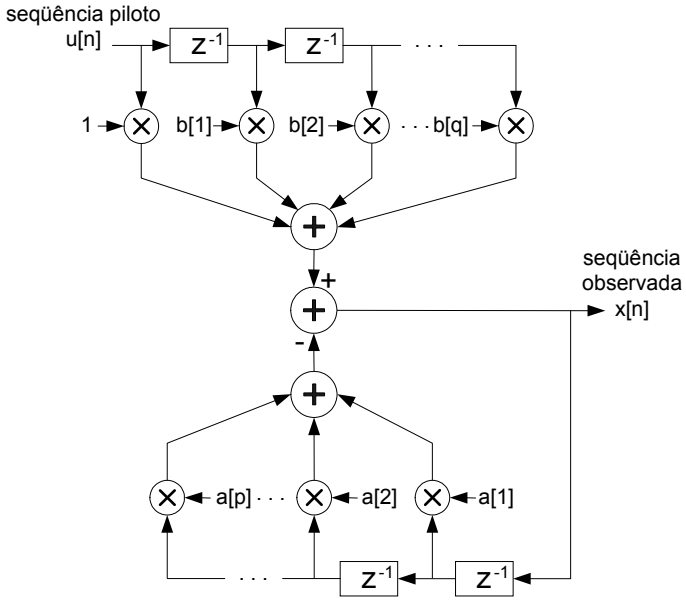

Figura 4.3: Modelo autoregressivo de média móvel de um processo aleatório.

A função de transferência do sistema entre a entrada $u[n]$ e a saída $x[n]$ para o processo ARMA da Equação 4.2 é a função racional dada por

$$H(z) = \frac{B(z)}{A(z)} \tag{4.3}$$

$$\text{em que } \begin{cases} A(z) = \text{transformada z do ramo AR} = \sum_{k=0}^{p} a[k]z^{-k} \\ B(z) = \text{transformada z do ramo MA} = \sum_{k=0}^{q} b[k]z^{-k} \end{cases}.$$

Assume-se que *A(z)* tem todos os seus zeros dentro do círculo unitário do plano *z*. Isso garante que H(z) é um filtro causal e estável. Sem essa assertiva, $x[n]$ em

4.2 não seria uma descrição válida de um processo aleatório estacionário no sentido amplo [18]. É sabido, [12], que a transformada Z da função de autocorrelação da saída de um filtro, $P_{xx}(z)$, está relacionada com a do filtro de entrada, $P_{uu}(z)$, pela equação

$$P_{xx}(z) = H(z)H^*(1/z^*)P_{uu}(z) = \frac{B(z)B^*(1/z^*)}{A(z)A^*(1/z^*)}P_{uu}(z). \tag{4.4}$$

Se calculada ao longo da circunferência unitária, $z = \exp(j2\pi f)$ para $-\frac{1}{2} \leqslant f \leqslant \frac{1}{2}$ a Eq. 4.4 torna-se, então, a PSD $P_{xx}(f)$. Freqüentemente o processo piloto é interpretado como sendo uma seqüência de ruído branco de média nula e variância σ^2. Assim, a PSD da saída de um processo ARMA é dada por

$$P_{ARMA}(f) = P_{xx}(z) = \sigma^2 \left| \frac{B(\exp(j2\pi f))}{A(\exp(j2\pi f))} \right|^2. \tag{4.5}$$

Com isso, especificar os parâmetros $a[k]$ (chamados de coeficientes autoregressivos), $b[k]$ (coeficientes de média móvel) e σ^2 é equivalente a especificar a PSD do processo $x[n]$. Sem perda de generalidade, pode-se assumir que $a[0] = 1$ e $b[0] = 1$, pois qualquer eventual ganho que haja no filtro pode ser incorporado a σ^2.

Se todos os coeficientes $b[k]$, exceto $b[0] = 1$, forem zero no modelo ARMA, então

$$x[n] = -\sum_{k=1}^{p} a[k]x[n-k] + u[n] \tag{4.6}$$

e o processo passa a ser estritamente autoregressivo (AR) de ordem p. O termo autoregressivo surge do fato de que a seqüência $x[n]$ é uma regressão linear dela mesma, com $u[n]$ representando o erro. Neste modelo, o valor atual de $x[n]$ é expresso como um somatório ponderado dos valores anteriores mais um termo correspondendo ao ruído. A PSD de um processo AR é

$$P_{AR}(f) = \frac{\sigma^2}{|A(\exp(j2\pi f))|^2}. \tag{4.7}$$

A Figura 4.4 representa esse modelo, denotado processo AR(p).

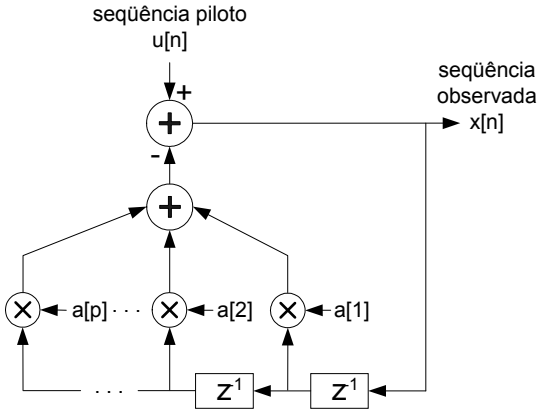

Figura 4.4: Modelo autoregressivo de um processo aleatório.

Por outro lado, se todos os coeficientes $a[k]$, exceto $a[0] = 1$, são nulos nos parâmetros do modelo ARMA, então

$$x[n] = \sum_{k=0}^{q} b[k]u[n-k] \tag{4.8}$$

e o processo é estritamente de média móvel (MA), de ordem q.

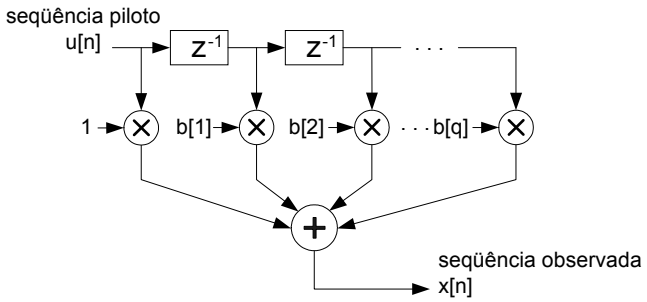

Figura 4.5: Modelo de média móvel de um processo aleatório.

A Figura 4.5 representa tal modelo, denotado por MA(q) e cuja PSD é dada por

$$P_{MA}(f) = \sigma^2 \left| B(\exp(j2\pi f)) \right|^2 . \tag{4.9}$$

Pela Figura 4.5, observa-se que, adicionando-se à seqüencia $x[n]$, amostras n_k de um ruído AWGN, tem-se, então, um modelo equivalente em tempo discreto de um canal com IIS e ruído AWGN.

4.3.2 Processo MA

O processo adotado neste trabalho é o processo MA, pela semelhança entre as Eqs. 4.8 e 4.1 e, conseqüentemente, pela possibilidade de representá-lo na forma de um diagrama de treliça invariante no tempo. A função de autocorrelação de um processo MA(1) é dada por [18]

$$r_{xx}[k] = \begin{cases} \sigma^2(1 + b^2[1]) \text{ para } k = 0 \\ \sigma^2 b[1] \text{ para } k = 1 \\ 0 \text{ para } k \geqslant 2 \end{cases} \tag{4.10}$$

com a PSD correspondente

$$P_{MA}(f) = \sigma^2 \left|1 + b[1] \exp(-j2\pi f)\right|^2 . \tag{4.11}$$

Para um processo MA(2), as relações são

$$r_{xx}[k] = \begin{cases} \sigma^2(1 + b^2[1] + b^2[2]) \text{ para } k = 0 \\ \sigma^2(b[1] + b[1]b[2]) \text{ para } k = 1 \\ \sigma^2(b[2] \text{ para } k = 2 \\ 0 \text{ para } k \geqslant 3 \end{cases} \tag{4.12}$$

e

$$P_{MA}(f) = \sigma^2 \left|1 + b[1] \exp(-j2\pi f) + b[2] \exp(-j4\pi f)\right|^2 . \tag{4.13}$$

Generalizando, a PSD de um MA(q) é

$$P_{MA}(f) = \sigma^2 \left| \sum_{k=0}^{q} b[k] \exp(j2\pi fk) \right|^2 . \tag{4.14}$$

A técnica de redução de estados aplicada nesse trabalho trata de aproximar a PSD do canal utilizado (Eq. 4.1) por uma PSD de um processo MA(q), $1 \leqslant q \leqslant 3$. A restrição no valor de q se dá pelo fato de que se for aplicado ao processo MA uma ordem $q = 4$, não haverá, então, redução no número de estados interpretados pelo receptor, uma vez que o número de elementos de memória do canal é $m = 4$ para o exemplo considerado. Assim, tal qual no método da Seção 4.2, serão obtidos os coeficientes para 2, 4 e 8 estados.

4.3.3 Erro Quadrático Médio

O critério utilizado para estimação dos coeficientes das PSDs com estados reduzidos na treliça do receptor foi o critério de Erro Quadrático Médio (MSE, do inglês *Mean Square Error*). Nesse critério, os coeficientes $b[k]$ do equalizador são ajustados de forma a minimizar o valor quadrático médio do erro

$$\varepsilon = (S(f) - P_{MA}(f))^2 \tag{4.15}$$

em que $S(f)$ é a PSD do canal da Eq. 4.1. Como se trata de valores discretos, deve-se utilizar amostras do erro.

Substituindo-se a Eq. 4.14 em 4.15, tem-se

$$\varepsilon = \sum_{l=1}^{L} \left(S(f_l) - \sigma^2 \left| \sum_{k=0}^{q} b[k] \exp(j2\pi f_l k) \right|^2 \right)^2 . \tag{4.16}$$

Encontrar os coeficientes $b[i]$, $1 \leqslant i \leqslant 3$, e σ^2 que minimizem o erro, ε, é calcular a solução para o sistema

$$\begin{cases} \frac{\partial \varepsilon}{\partial b[i]} = \frac{\partial}{\partial b[i]} \left[\sum_{l=1}^{L} \left(S(f_l) - \sigma^2 \left| \sum_{k=0}^{q} b[k] \exp(j2\pi f_l k) \right|^2 \right)^2 \right] = 0, 1 \leq i \leq 3 \\ \frac{\partial \varepsilon}{\partial \sigma^2} = \frac{\partial}{\partial \sigma^2} \left[\sum_{l=1}^{L} \left(S(f_l) - \sigma^2 \left| \sum_{k=0}^{q} b[k] \exp(j2\pi f_l k) \right|^2 \right)^2 \right] = 0 \end{cases} . \tag{4.17}$$

Na Tabela 4.2 constam os valores obtidos dos coeficientes $b[n]$ para $q = 1$, 2 e 3, encontrados como solução do sistema da Eq. 4.17.

q	b[1]	b[2]	b[3]	σ^2	Erro Médio Quadrático
1	1	-	-	0,614399	0.798357
2	1,29762	1,6	-	0,223906	0,193058
3	1,23886	1,23886	1	0,205577	0,008145

Tabela 4.2: Parâmetros do processo MA para q = 1, 2 e 3.

4.3.4 Resultados

Com os valores da Tabela 4.2, pode-se calcular as novas métricas a serem ajustadas no receptor. Como no método da Seção 4.2, os resultados obtidos para as aproximações referentes a $q = 1$ e $q = 2$, referentes a dois e quatro elementos de memória, respectivamente, não foram bons.

Para a aproximação do espectro pelo processo MA(3), cujos coeficientes obtidos constam na Tabela 4.2, os resultados foram melhores. O Gráfico 4.6 mostra os resultados das simulações desse sistema. Pode ser observado nesse gráfico que o sistema implementado, apesar de melhorar o desempenho através das iterações, converge rapidamente para o ponto de saturação, que se encontra próximo a 5×10^{-4}, ou seja, um erro a cada dois mil bits transmitidos.

A estimação espectral dos coeficientes do canal, embora tenha alcançado um

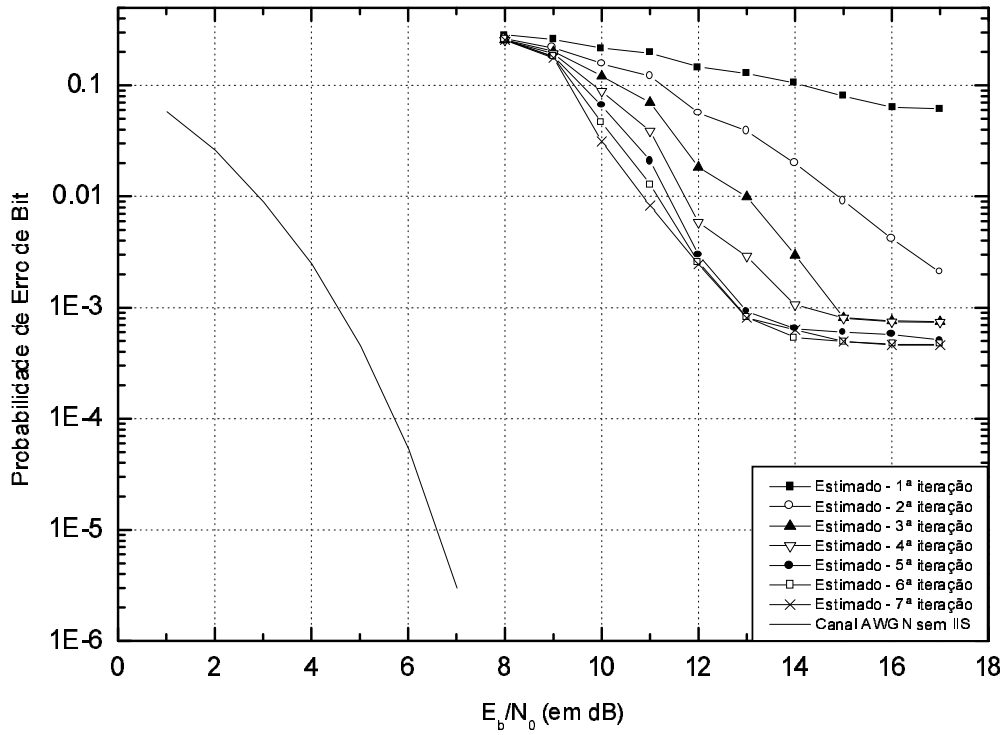

Figura 4.6: P_b x E_b/N_0 utilizando a técnica de estimação espectral para 8 estados.

erro relativo cerca de dez vezes menor que o erro relativo do sistema truncado, para oito estados, obteve um desempenho pior. O Gráfico 4.7 mostra que a PSD estimada, embora esteja bem próxima à real até $-10dB$, diverge bastante a partir desse ponto, elevando o erro relativo entre as duas curvas. Em geral, os processos AR e ARMA apresentam desempenho melhor que o processo MA em espectros com picos acentuados e vales profundos [18]. Dessa forma, para que um processo MA atinja uma aproximação razoável, é necessário elevar a ordem q desse processo, acarretando num aumento do número de coeficientes e, conseqüentemente, estados na treliça do decodificador. Apesar disso, a característica variante no tempo das treliças que representam os processos AR e ARMA (o que não ocorre no processo MA), fez com que a escolha por eles fosse descartada.

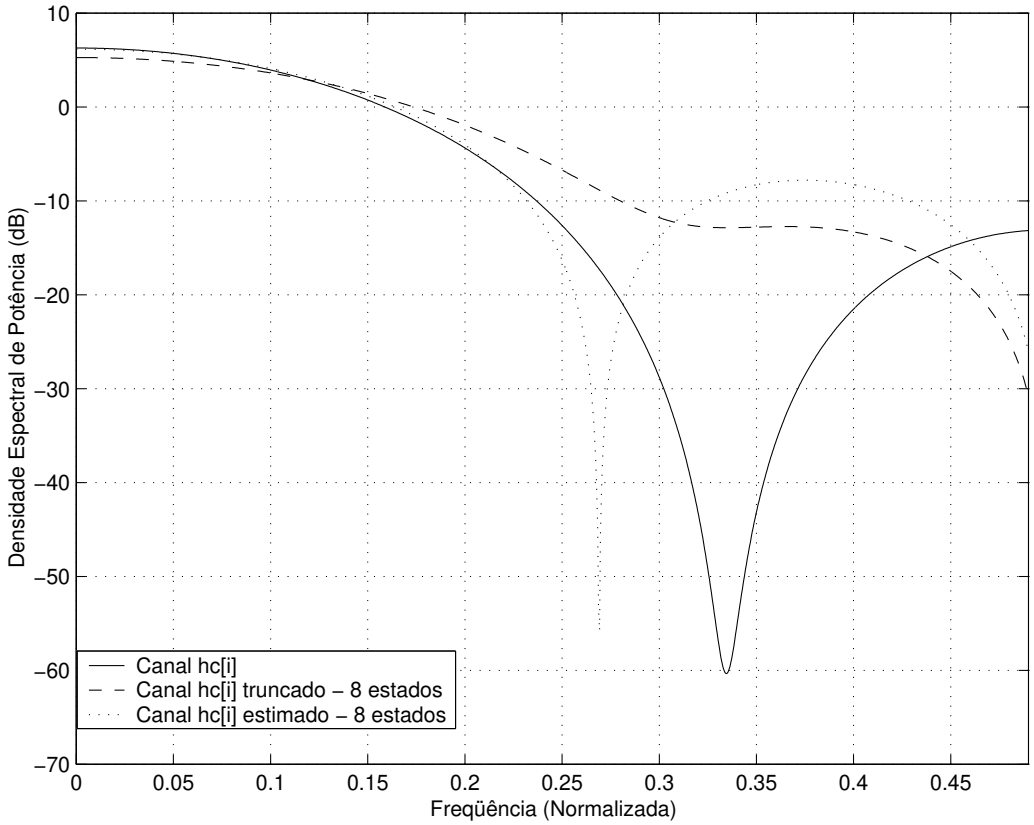

Figura 4.7: Comparação entre as PSD real, estimada de ordem 3 e truncada com 3 elementos de memória.

4.4 Algoritmo M

Em [19], duas variantes do algoritmo BCJR original foram aplicadas, no sentido de redução de cálculos na decodificação de códigos concatenados paralela e serialmente. Essas variações, chamadas de algoritmo M e algoritmo T, são algoritmos de busca reduzida que eliminam o cálculo de probabilidades de transição de estados na treliça, de acordo com um determinado critério. Dessa forma, os ramos na treliça referentes à essas transições são eliminados.

Esses algoritmos consideram o fato de que algumas componentes dos vetores α e β do BCJR ou A_k e B_k do SISO utilizado possuem valores muito pequenos

se comparadas às componentes mais significativas. Pode-se assim, removê-los dos cálculos sem perda expressiva em desempenho.

O algoritmo T determina um valor limiar de probabilidade abaixo do qual todos os valores de A_k e B_k são descartados, no instante k. Isso é feito em cada seção da treliça e os valores descartados nos conjuntos A_{k-1} e B_{k+1} não são considerados no cálculo de A_k e B_k, respectivamente. Para que sejam descartadas apenas componentes menos significativas, é necessário fazer, a cada seção da treliça, a normalização dos vetores A_k e B_k.

O algoritmo M mantém os M estados com maior probabilidade em cada seção da treliça. Assim, no cálculo de A_k e B_k, são considerados apenas os M maiores valores de A_{k-1} e B_{k+1}. Dessa forma, embora para o caso binário apenas M ramos sejam considerados para o cálculo de cada distribuição de probabilidade de saída, os valores de A_k e B_k são calculados para todos os estados e somente depois de comparados, descarta-se os $2^m - M$ com menores valores. Dessa forma, o algoritmo M efetua uma quantidade de cálculos superior à dos outros métodos estudados, uma vez que estes últimos somente calculam os parâmetros da treliça para o número de estados utilizados. Entretanto, o volume de cálculos necessários para executar as rotinas inerentes aos algoritmos de busca reduzida é bem menor que o executado pelo algoritmo com todos os estados e reduz consideravelmente o tempo de processamento gasto na recepção.

Em sistemas concatenados serialmente, o algoritmo T se mostrou mais eficiente que o algoritmo M [19]. Entretanto, como o algoritmo T estabelece um limiar para A_k e B_k e não se sabe quantos estados estão acima desse limiar, não há como controlar o número de estados em cada seção da treliça. Já o algoritmo M determina o número de estados sobreviventes, o que facilita a obtenção de uma estimativa da quantidade de cálculos realizados na decodificação. Assim, foi escolhida a aplicação do algoritmo M para fins de comparação com os métodos utilizados anteriormente, para o mesmo número de estados.

4.4.1 Resultados

Como o algoritmo M controla efetivamente a quantidade de estados sobreviventes, pode-se obter o desempenho desse sistema para quaisquer quantidade de estados na treliça do equalizador. É intuitivo que quanto maior o número de estados que se retira, pior será o desempenho. Entretanto, o número médio de estados sobreviventes, em geral, tende para valores baixos à medida que se aumenta E_b/N_0 e a quantidade de iterações [20]. Dessa forma, a retirada de estados da treliça com baixas probabilidades de ocorrência não obrigatoriamente diminui o desempenho do sistema. Com o objetivo de se avaliar o quanto se perde em desempenho com a

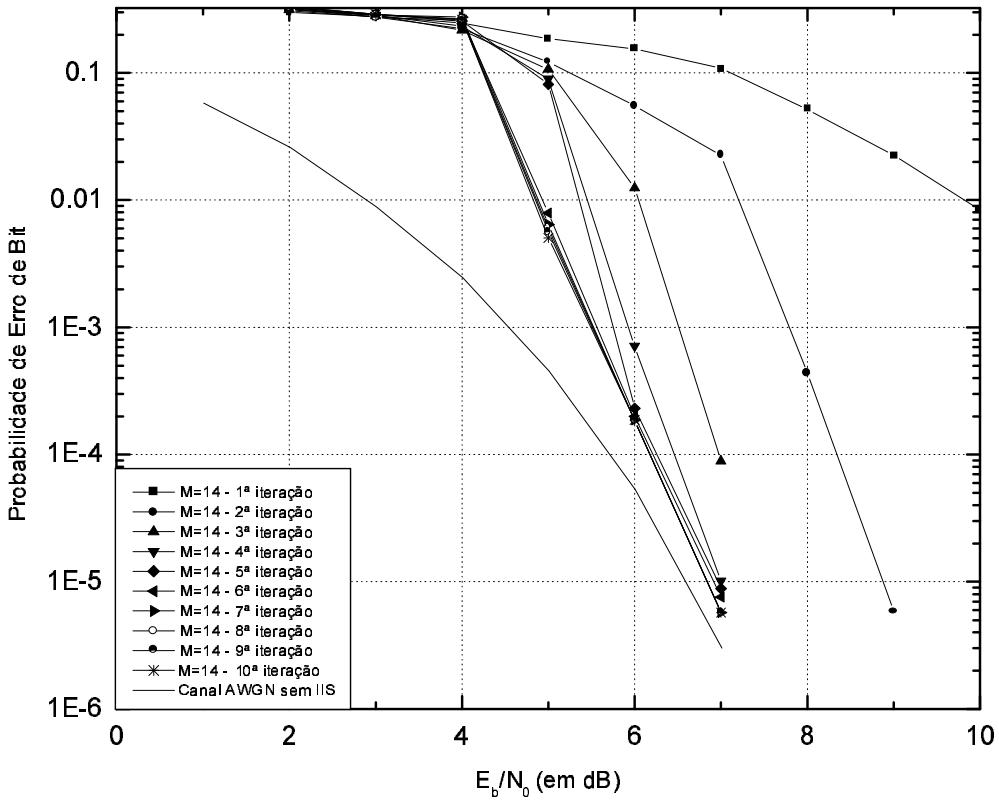

Figura 4.8: P_b x E_b/N_0 utilizando o algoritmo M para M=14 estados.

retirada de estados, foram realizadas várias simulações utilizando-se o algoritmo M, para valores de M iguais a 4, 8, 10, 12 e 14.

A Figura 4.8 mostra os resultados da simulação do sistema aplicando-se o algoritmo M para 14 estados. Como pode ser observado, este resultado está próximo do resultado obtido para os 16 estados (Figura 3.8), embora a curva tenha se afastado um pouco da curva de um canal sem IIS, para $\frac{E_b}{N_0} \simeq 5$ dB. Para valores de RSR maiores que 6 dB, o sistema com 14 estados tende a se aproximar bastante do de 16 estados. Observa-se ainda que o ponto de saturação do sistema está em 6 iterações.

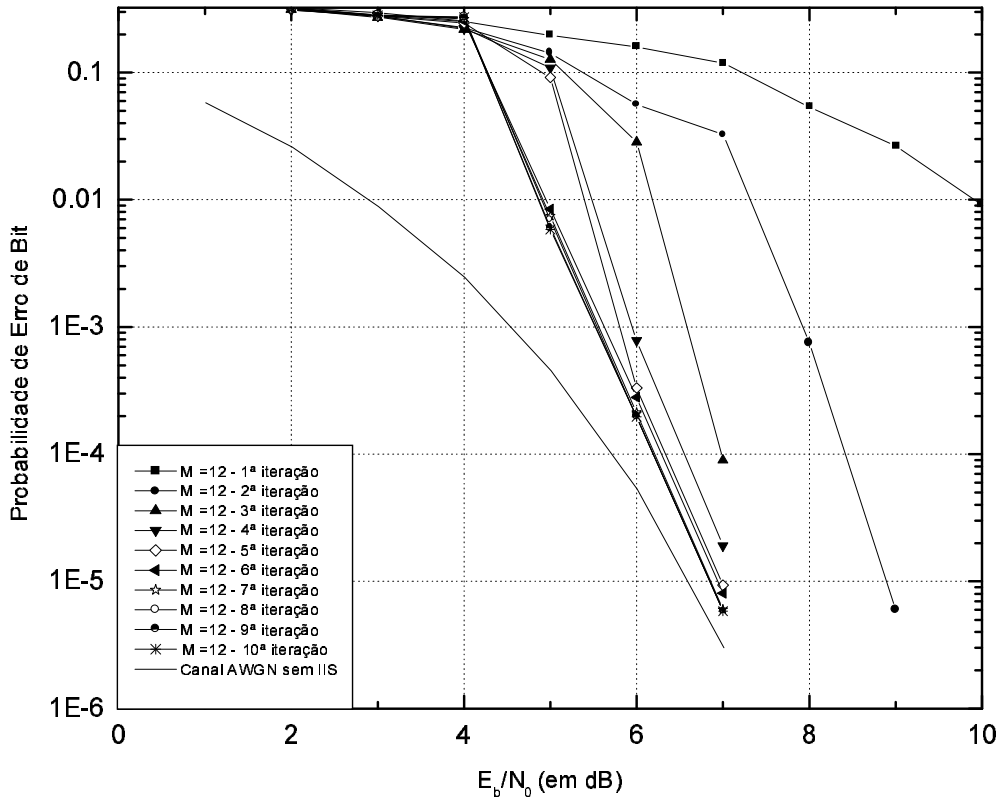

Figura 4.9: P_b x E_b/N_0 utilizando o algoritmo M para M=12 estados.

O resultado da simulação para $M = 12$ estados é mostrado na Figura 4.9. Embora tenha havido perda em desempenho do sistema com 16 estados para o de

14 estados, comparando-se os sistemas para 14 e 12 estados, percebe-se que esses sistemas apresentam curvas de desempenho muito próximas, com uma variação pequena na taxa de erro. Isso reforça a idéia de que a retirada de alguns estados da treliça não necessariamente prejudica o desempenho do sistema.

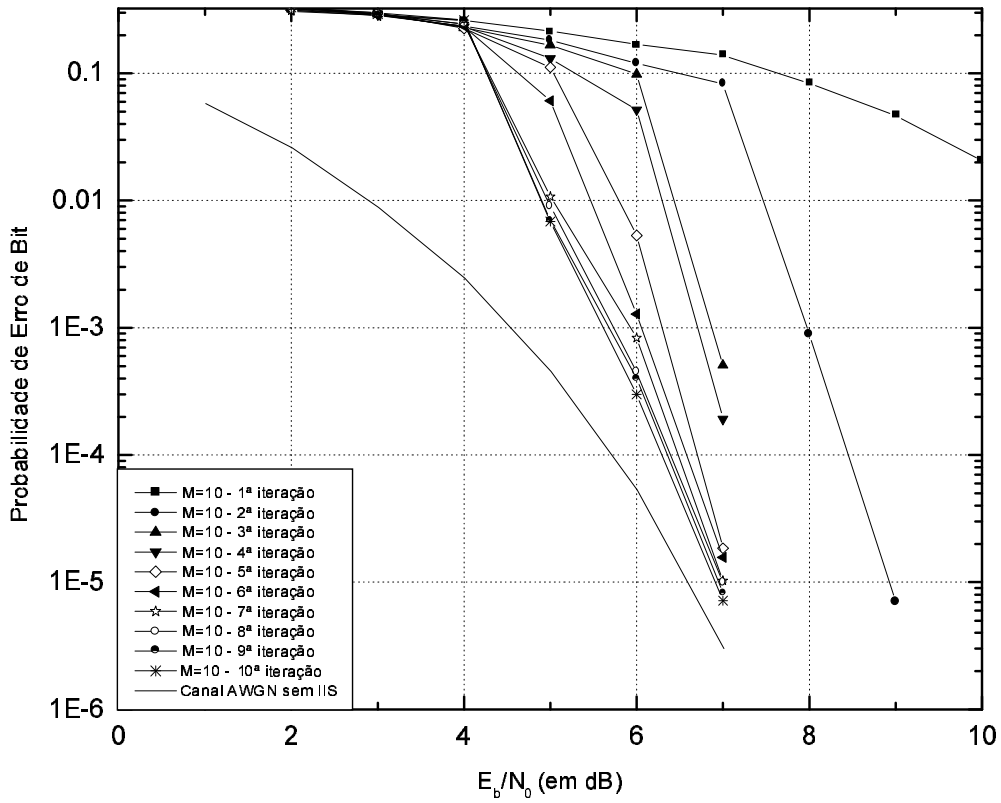

Figura 4.10: P_b x E_b/N_0 utilizando o algoritmo M para M=10 estados.

A Figura 4.10 mostra o resultado da simulação do sistema para $M = 10$ estados. Observa-se, nesse caso, um espalhamento entre as taxas de erro de bit para as várias iterações. Enquanto que para as primeiras iterações o sistema tem seu desempenho prejudicado, ainda que bem pouco, para as últimas iterações, o desempenho se aproxima do sistema com 12 estados. Isso confirma a idéia de que o número de estados sobreviventes tende a um valor baixo, à medida em que se aumenta o número

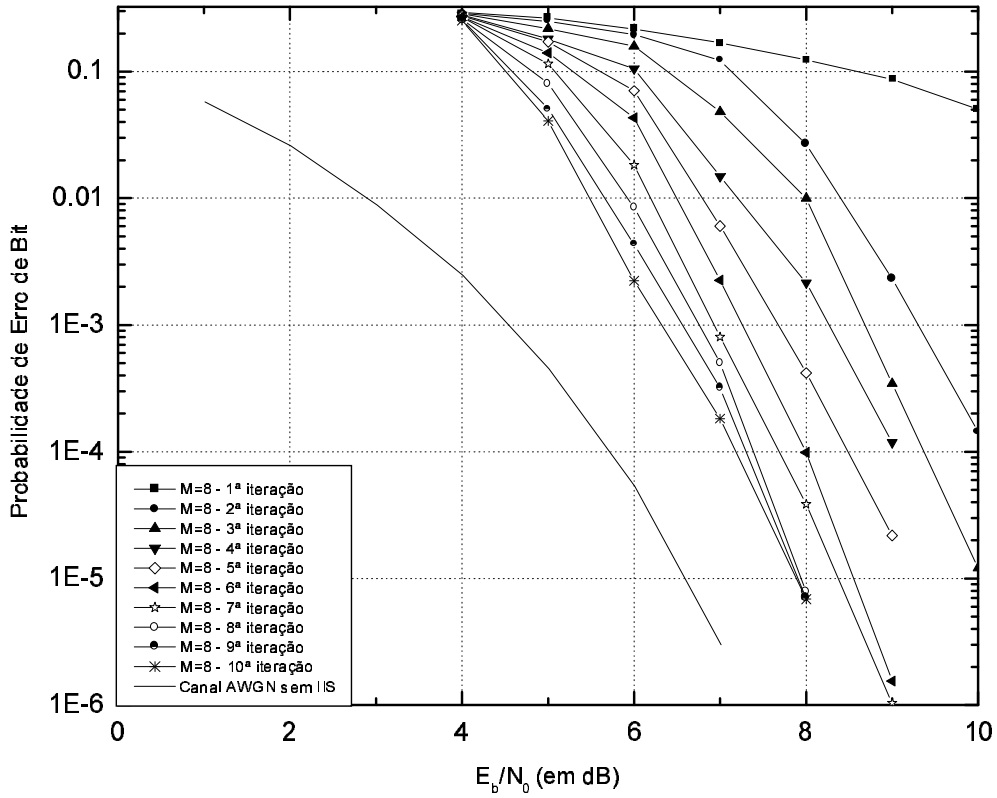

Figura 4.11: P_b x E_b/N_0 utilizando o algoritmo M para M=8 estados.

de iterações. Outro fato relevante é que quanto mais estados se retira, mais iterações são necessárias até atingir o ponto de saturação.

O resultado para a simulação do sistema tendo-se $M = 8$ estados é mostrado na Figura 4.11. Há uma perda considerável desse sistema para o de 10 estados. Para uma BER de 10^{-4} e 10 iterações, por exemplo, a diferença é de quase 1 dB. Para 5 iterações, essa diferença sobe para mais de $1,5$ dB. Em relação ao sistema com todos os estados (fig. 3.8), para essa mesma BER, a diferença chega a aproximadamente $1,2$dB.

Foi realizada ainda a simulação do sistema para $M = 4$, a fim de se verificar se há uma perda considerável quando se mantém apenas quatro estados sobreviventes.

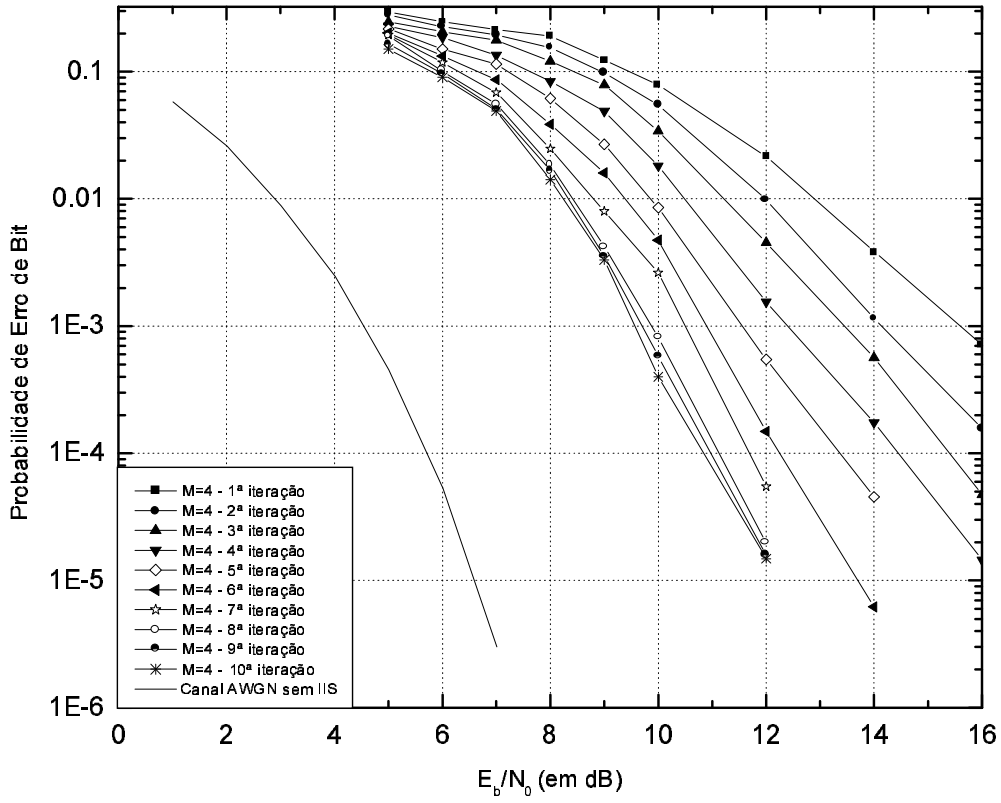

Figura 4.12: P_b x E_b/N_0 utilizando o algoritmo M para M=4 estados.

A Figura 4.12 mostra o resultado obtido e pode-se observar que o desempenho desse sitema é bastante pobre se comparado ao de 16 estados. Para uma BER de 10^{-4} e 10 iterações, há uma perda de mais de 3 dB. Para 5 iterações, a diferença é de aproximadamente 5 dB. Entretanto, comparando-se o desempenho desse sistema com os dos sistemas truncado e estimado com oito estados (figs. 4.2 e 4.6), percebe-se que ainda assim o sistema com quatro estados tem desempenho superior. A diferença em relação ao sistema truncado, para uma BER de 10^{-4} e sete iterações, é de aproximadamente um dB, enquanto que o sistema estimado nem sequer atinge esse patamar de erro, saturando para uma BER de 5×10^{-4}.

4.5 Comparação dos Resultados

Tendo-se completado as simulações dos três métodos de redução de estados, pode-se então comparar as respectivas curvas de desempenho para o mesmo número de estados.

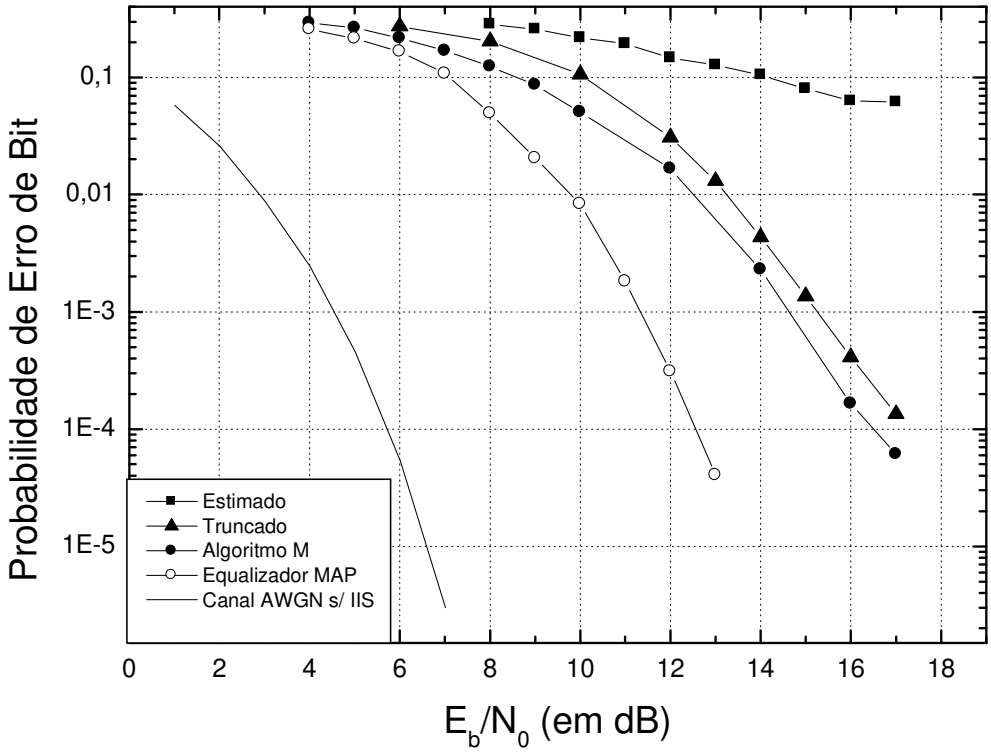

Figura 4.13: Comparação entre os métodos de redução de estados para uma iteração e oito estados.

A Figura 4.13 mostra a comparação do desempenho dos três métodos estudados para uma iteração. O método de estimação dos coeficientes do canal obteve um desempenho bastante ruim, saturando perto de uma BER de $0,06$. Os outros dois sistemas, truncamento dos coeficientes e algoritmo M, tiveram desempenhos melhores e bem próximos um do outro. Para uma BER de 10^{-4}, o método de trun-

camento necessita de uma RSR de pouco mais que 17 dB, quase um dB a mais que o algoritmo M. Este, por sua vez, para a mesma BER, está aproximadamente 4 dB pior que o equalizador MAP com todos os estados.

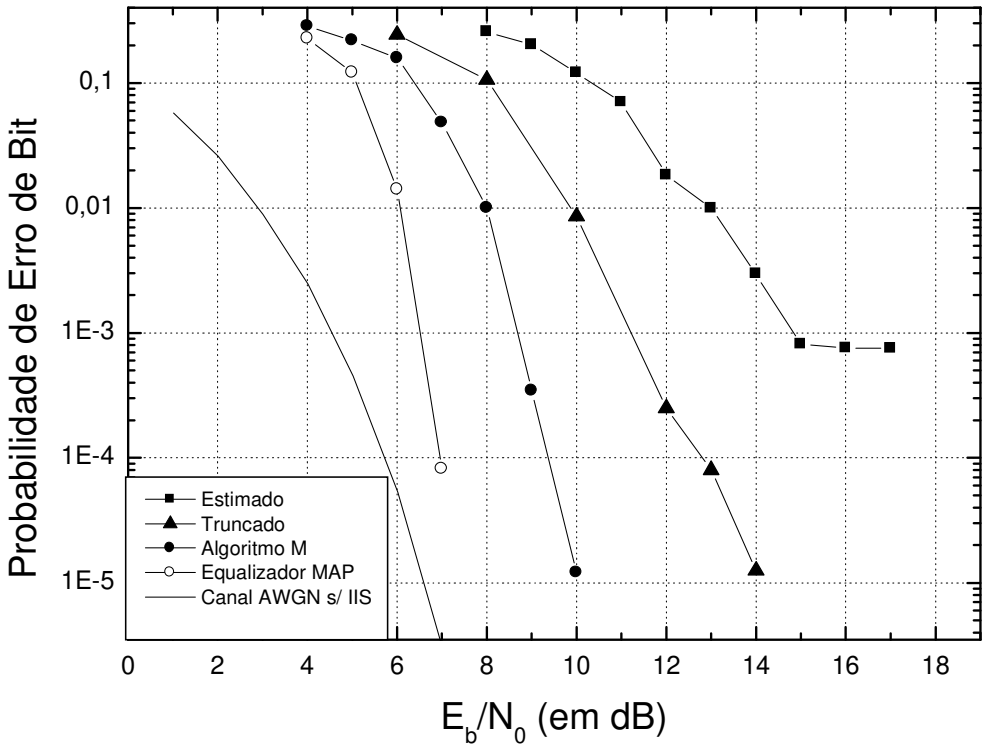

Figura 4.14: Comparação entre os métodos de redução de estados para três iterações e oito estados.

Para três iterações, a comparação é feita na Figura 4.14. Como pode ser observado, novamente o algoritmo M obteve o melhor desempenho, aproximadamente $9,5$ dB para uma BER de 10^{-4}. Se comparado ao sistema do capítulo 3, a redução de estados desse método acarreta num acréscimo de $2,5$ dB, para o mesmo número de iterações. O desempenho do truncamento dos coeficientes está por volta de $3,5$ dB pior que o do algoritmo M. O método da estimação dos coeficientes não consegue, novamente, atingir o patamar de BER analisado.

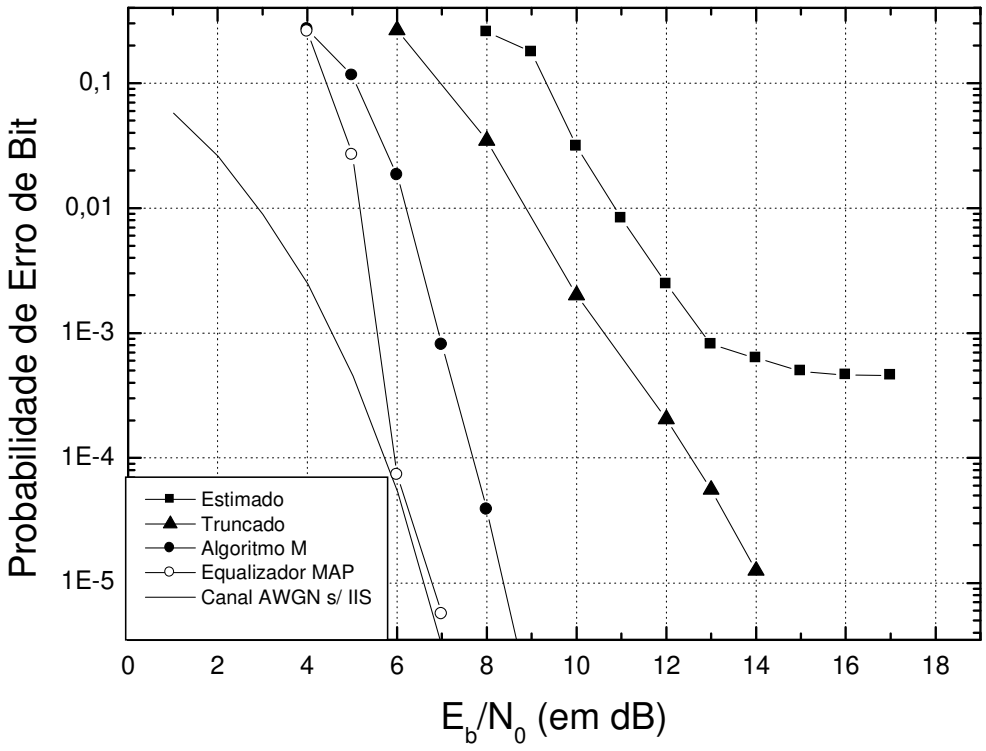

Figura 4.15: Comparação entre os métodos de redução de estados para sete iterações e oito estados.

O desempenho dos métodos estudados, para sete iterações, é mostrado na Figura 4.15. Há um ganho razoável de codificação para todos os três métodos, da primeira para a sétima iteração, denotando que o processo iterativo de equalização efetivamente melhora o desempenho do canal mesmo utilizando-se métodos aproximados de redução de estados. Pode-se observar, nessa figura, que o algoritmo M atinge uma diferença de desempenho de apenas 2dB de um canal livre de IIS, para uma BER de 10^{-4}. O desempenho do método de truncamento, bem pior, dista de 5 dB do desempenho do algoritmo M.

4.6 Complexidade

Um aspecto a ser levado em consideração, quando se trata de algoritmos de equalização SISO, sobretudo aqueles que utilizam uma regra de decisão MAP ou ML, é o esforço computacional exigido no processamento dos blocos recebidos.

Definindo-se a complexidade computacional como a quantidade de operações de adição e multiplicação realizadas por símbolo recebido por iteração, pode-se chegar a uma comparação razoável entre os diversos métodos de equalização utilizados. Trata-se, na verdade, de uma aproximação para o volume de cálculos realizados. Dessa forma, a curva de desempenho em relação à complexidade de cada método pode ser analisada. Essa análise oferece dados suficientes para que a escolha da técnica a ser aplicada, por parte do projetista do sistema, seja a mais adequada.

O equalizador MAP utilizado, derivado do BCJR [5] e utilizado no Capítulo 3, realiza o cálculos das grandezas A_k, B_k e $P_k(U;O)$ (Equações 3.3, 3.4 e 3.7). A distribuição de probabilidades $P_k(C;O)$ (Eq. 3.8), como foi dito anteriormente, não é calculada nesse decodificador. Assim, para cada seção da treliça, pelas Equações 3.3, 3.4 e 3.7, há um total de $3.2^m.2^M$ multiplicações e $3.2^{M-1}(2^m-1)$ adições, onde 2^m é o tamanho do alfabeto da constelação de sinais e M é o comprimento da resposta ao impulso do canal.

Para as técnicas de redução de estados utilizadas neste capítulo, também pode-se fazer uma estimativa da quantidade de cálculos necessários. No caso das duas primeiras técnicas apresentadas, truncamento dos coeficientes de canal e estimação dos coeficientes do canal, há apenas uma redução no número de estados em relação ao algoritmo implementado no Capítulo 3, ou seja, diminuição no valor de M. A quantidade de operações é dada pelas mesmas equações.

A terceira técnica utilizada, o algoritmo M, ao contrário das demais, realiza uma redução probabilística de estados da treliça do canal. Dessa forma, não há como se prever exatamente a posição dos A_k e B_k. Como o número de ramos a serem eliminados é uma função da posição de A_k e de B_k, também não se pode prever

quantos ramos da treliça serão eliminados.

A Figura 4.16 mostra dois exemplos onde há variação na posição dos A_k e B_k eliminados, para M=2, ou seja, apenas dois estados são mantidos, dos quatro iniciais.

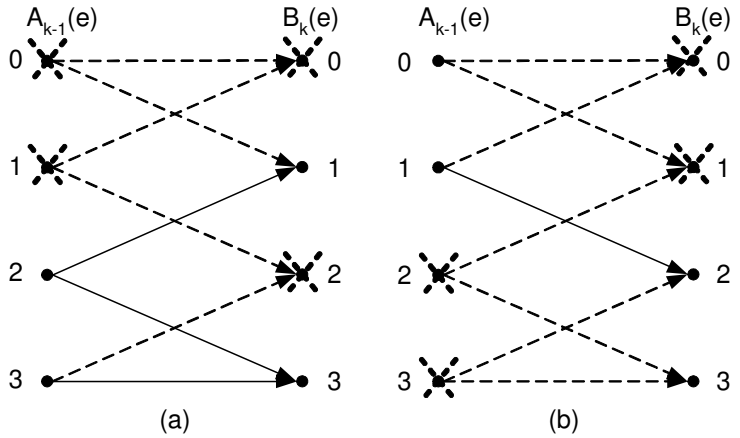

Figura 4.16: Exemplo de diferença na quantidade de ramos da treliça considerados no cálculo de $P_k(U;O)$ para o algoritmo M.

Na Figura 4.16(a), como os valores de $A_{k-1}(0)$, $A_{k-1}(1)$, $B_k(0)$ e $B_k(2)$ são zerados, no cálculo de $P_k(U;O)$ são considerados três ramos (linhas contínuas). Os ramos pontilhados são eliminados. Na Figura 4.16(b), zerando-se $A_{k-1}(2)$, $A_{k-1}(3)$, $B_k(0)$ e $B_k(1)$, apenas um ramo é considerado em $P_k(U;O)$, todos os demais são iguais a zero.

Os cálculos de A_k e B_k são recursões direta e inversa, respectivamente. Assim, a eliminação de um dado A_{k-1} e um dado B_{k+1} influenciará no número de operações exigidas no cálculo de A_k e B_k, uma vez que o número de ramos pode se alterar.

Em termos práticos, como não se pode prever quais posições de A_k e B_k serão eliminadas e quais serão mantidas, o que se busca é estimar o volume de cálculos para A_k, B_k e $P_k(U;O)$ no algoritmo M.

Uma aproximação pode ser obtida fazendo-se a média (μ) dos ramos sobreviventes dentre todas as combinações possíveis de posições de A_k e B_k. No caso específico deste trabalho, considera-se 16 estados ao todo e $M = 8$. Assim, tanto

A_k quanto B_k possui, ao todo, $\frac{16!}{8!(16-8)!}$ possíveis combinações. Para tal sistema, o número médio de ramos sobreviventes obtido foi de 10, ou 31,25% do total de ramos.

A Tabela 4.3 mostra o volume de operações necessárias para os diversos métodos estudados.

Método	Adições	Multiplicações
Equalizador MAP	$3.2^{M-1}(2^m - 1)$	$3.2^m.2^M$
Truncamento	$3.2^{M-1}(2^m - 1)$	$3.2^m.2^M$
Estimação	$3.2^{M-1}(2^m - 1)$	$3.2^m.2^M$
Algoritmo M	$3.2^m\frac{\mu}{2}$	$2^m\left(2^M + 2\mu\right)$

Tabela 4.3: Volume de operações de adição e multiplicação por símbolo recebido por iteração requeridas usando os diversos métodos apresentados.

Para o caso em que o número de estados é reduzido de 16 para 8 estados, a redução no volume de operações em relação ao equalizador MAP é mostrada na Tabela 4.4.

Método	Adições	Multiplicações	Total	Redução (%)
Equalizador MAP	48	192	240	-
Truncamento	24	96	120	50
Estimação	24	96	120	50
Algoritmo M	30	104	134	44

Tabela 4.4: Comparação entre o volume de operações de adição e multiplicação por símbolo recebido por iteração requeridas usando os diversos métodos apresentados, para 8 estados.

Percebe-se que há uma redução de complexidade de aproximadamente 50% dos métodos de truncamento e estimação dos coeficientes do canal em relação ao equalizador MAP. Essa redução, para o Algoritmo M, é de 44%. Há, portanto, uma

coerência quanto às variáveis envolvidas, complexidade e desempenho, para os métodos estudados. Ou seja, o Algoritmo M obteve melhor desempenho que as demais técnicas mas, em contrapartida, também possui maior complexidade de implementação.

4.7 Comparação com Métodos de Equalização Linear

Outras tentativas de se diminuir a complexidade do equalizador MAP podem ser encontradas na literatura de equalização turbo. Em [15], equalização linear é utilizada para simplificar o processo de detecção iterativo. Neste artigo, a implementação de um código convolucional recursivo sistemático (RSC), com mesma matriz geradora, num sistema parecido e blocos de informação de 32.000 bits é considerada.

A Figura 13 de [15] mostra o desempenho dos métodos baseados em equalização linear.

Para uma BER de 10^{-4} e com 15 iterações, há uma diferença de aproximadamente 0,5 dB entre o sistema de equalização turbo de [15] e o desenvolvido neste trabalho (Fig. 3.8) . Isso mostra que, mesmo com um bloco de informação igual a metade do usado em [15] e sem a utilização de um código RSC, o resultado obtido foi satisfatório.

Comparando-se os resultados obtidos em [15] para aproximações de equalizadores lineares com os resultados alcançados pelos métodos estudados neste trabalho, observa-se que o desempenho do algoritmo M e do equalizador linear MMSE estão muito próximos, para uma BER de 10^{-4} e com três iterações. Da mesma forma, o método de truncamento e o equalizador linear MMSE aproximado também tem desempenhos similares, para a mesma BER. O método de estimação, mesmo não tendo uma melhoria satisfatória, apresenta desempenho superior ao equalizador MMSE

DFE.

Em geral, os métodos de equalização linear tendem a apresentar uma complexidade menor que os derivados de um equalizador MAP. Mas para se ter uma comparação mais ampla é necessário analisar o comportamento de todos esses métodos em diversos canais, com respostas impulsivas diferentes.

5

Conclusão

Este trabalho considerou um estudo sobre a utilização de um sistema de comunicação digital através de um canal introduzindo IIS e a aplicação da equalização turbo para eliminar as distorções causadas pelo canal.

Em um primeiro instante, verificou-se que o módulo de decodificação desenvolvido em [14] e em [5] foi capaz de eliminar a IIS desse canal utilizando, para isso, comprimentos de blocos finitos (16000 bits) e número de iterações também finito. O resultado das simulações se mostrou bem próximo ao obtido em [15], para um esquema de equalização turbo similar, utilizando codificadores RSC e comprimento de bloco de informação de 32000 bits. Foi verificado ainda que o desempenho desse mesmo sistema, para comprimento de bloco de 32000 bits de informação, se mostrou

praticamente idêntico ao do bloco de 16000 bits. Esse resultado leva a crer que, embora o tamanho do bloco influencie no desempenho do sistema, há um ponto de saturação inerente a cada sistema. Como a complexidade tende a aumentar com o aumento do comprimento do bloco, o uso de blocos de 16000 bits se mostrou mais adequado para esse sistema.

Posteriormente, dada a complexidade de implementação do bloco de decodificação do canal com IIS, buscou-se analisar o desempenho de três métodos de redução dos estados da treliça do canal. Dois desses métodos, o truncamento dos coeficientes do canal e a estimação dos coeficientes do canal, apresentam uma abordagem determinística, diferentemente do outro método, o algoritmo M, cuja abordagem é probabilística.

O método do truncamento dos coeficientes do canal atingiu um desempenho intermediário, entre os demais métodos. Essa característica deve variar de acordo com o canal escolhido. Quanto maior for a influência do(s) elemento(s) de memória retirado(s) sobre o símbolo, maior será a diferença de níveis entre o que é transmitido e o que é interpretado pelo receptor. Além disso, quanto maior for o comprimento da memória do canal, maior será a vantagem de se utilizar este método já que a retirada de uma memória representa, para um alfabeto binário, a redução, pela metade, do número de estados.

A estimação dos coeficientes do canal apresentou um desempenho pobre. Uma justificativa pode ser a própria escolha do modelo. A PSD desse canal apresenta um pico negativo estreito. Como mostrado em [18], processos MA de ordem pequena não são adequados para aproximar este tipo de comportamento espectral. Neste caso, seria melhor usar um modelo ARMA ou AR [18]. Entretanto, a escolha do processo MA se deve à necessidade de representação do modelo na forma de uma treliça invariante no tempo. Outra justificativa pode estar na falta de informação sobre a fase da resposta impulsiva, já que somente a amplitude desta é considerada na aproximação da PSD do canal.

Embora o volume de cálculos do algoritmo M seja ligeiramente maior que o dos demais métodos, ele sempre apresentou o melhor compromisso entre desempenho e complexidade. Com isso conclui-se que métodos probabilísticos de redução de estados, em geral, possuem um melhor compromisso.

Em [15], diferentes métodos de equalização linear foram considerados para a simplificação do equalizador turbo. Resultados de desempenho foram obtidos para um bloco de informação de 32.000 bits utilizando um codificador convolucional sistemático e recursivo. Para uma taxa de erro igual a 10^{-4} e três iterações, o desempenho do algoritmo M aproxima-se ao do equalizador linear MMSE, o método de truncamento equivale ao do equalizador linear MMSE aproximado e o método de estimação de coeficientes supera o do equalizador MMSE DFE.

Como proposta para trabalhos futuros, pode-se citar a necessidade do estudo de alternativas mais adequadas para se obter aproximações dos coeficientes do canal que melhorem o desempenho do método de estimação dos coeficientes. Recentemente, outras abordagens de redução probabilística de estados foram apresentadas em [20][21]. Uma comparação entre o método proposto em [21], o algoritmo M e o algoritmo híbrido de [20] poderá ser realizada, a fim de se avaliar qual desses métodos apresenta o melhor compromisso entre complexidade e desempenho.

Referências Bibliográficas

[1] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3-4):379–423,623–656, October 1948.

[2] G. D. Forney Jr. *Concatenated Codes*. Tese de Doutorado, M.I.T., Cambridge,MA, March 1966.

[3] C.Berrou, A.Glavieux, and P.Thitimajshima. Near Shannon limit error-correcting coding and decoding:Turbo-codes. In *IEEE International Conference on Communications (ICC'93)*, volume 2, pages 1064–1070, Geneva, Switerzland, May 1993.

[4] G. Montorsi S. Benedetto, D. Divsalar and F. Pollara. Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding. *IEEE Transactions on Informatin Theory*, 44(3):909–926, May 1998.

[5] G. Montorsi S. Benedetto, D. Divsalar and F. Pollara. Soft-Input Soft-Output Modules for the Construction and Distributed Iterative Decoding of Code Networks. *European Transactions on Telecommunications Special Issue*, 9(2):165–172, March-April 1998.

[6] A. Picart P. Didier C. Douillard, M. Jéséquel and A. Glavieux. Iterative Correction of Intersymbol Interference: Turbo-Equalization. *European Transactions on Telecommunications*, 6(5):507–511, September-October 1995.

[7] S. Lin and D. Costello. *Error Control Coding: Fundamentals and Applications*. Prentice-Hall, 1ª edition, 1983.

[8] C. Heegard and S. Wicker. *Turbo Coding*. Kluwer Academic, 1ª edition, 1999.

[9] C. Berrou and A. Glavieux. Near Optimum Error Correcting Coding and Decoding: Turbo- Codes. *IEEE Transactions on Communications*, 44(10):1261–1271, October 1996.

[10] J. Seghers L. C. Perez and Jr. D. J. Costello. A distance spectrum interpretation of turbo codes. *IEEE Transactions on Informatin Theory*, 42(6):1698–1709, November 1996.

[11] D. Divsalar and F. Polara. Turbo Codes for PCS Applications. In *IEEE International Conference on Communications (ICC'95)*, pages 54–59, Seattle, WA - USA, June 1995.

[12] E. A. Lee and D. G. Messerschmitt. *Digital Communication*. Kluwer Academic, 3ª edition, 1994.

[13] J. G. Proakis. *Digital Communications*. McGraw-Hill, 3ª edition, 1995.

[14] G. Montorsi S. Benedetto, D. Divsalar and F. Pollara. A Soft-Input Soft Output APP Module for Iterative Decoding of Concatenated Codes. *IEEE Communications Letters*, 1(1):22–24, January 1997.

[15] R. Koetter M. Tüchler and A. C. Singer. Turbo Equalization: Principles and New Results. *IEEE Transactions on Communications*, 50(5):754–767, Mayewewewe 2002.

[16] Bernard Sklar. *Digital Communications: Fundamentals and Applications*. Prentice Hall, 2ª edition, 2001.

[17] F. Jelinek L. R. Bahl, J. Cocke and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory*, (1):284–287, march 1974.

[18] Steven M. Kay. *Modern Spectral Estimation: Theory and Application*. Prentice Hall, 1ª edition, 1988.

[19] Volker Franz and John B. Anderson. Concatenated Decoding with a Reduced-Search BCJR Algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):186–195, February 1998.

[20] H. Freire e J. Portugheis. Algoritmo H-BCJR de Busca Reduzida para Código Turbo. In *XVIII Simpósio Brasileiro de Telecomunicações*, Gramado, RS - Brasil, Setembro 2000.

[21] G. Ferrari G Colavolpe and R. Raheli. Reduced-State BCJR-Type Algorithms. *IEEE Journal on Selected Areas in Communications*, 19(5):848–859, May 2001.