UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

---

DISSERTAÇÃO

apresentada à UTFPR

para obtenção do grau de

MESTRE EM CIÊNCIAS

por

GILBERTO TITERICZ JUNIOR

Uma Avaliação de Diferentes Métodos Para Transmissão de Fontes Não-Uniformes Usando Códigos Turbo

---

Banca Examinadora:

Presidente e Orientador:

Prof. Dr. Richard Demo Souza	UTFPR

Examinadores:

Prof. Dr. Evelio Martín García Fernández	UFPR
Prof. Dr. Marcelo Eduardo Pellenz	PUC-PR

Curitiba, 14 de Março de 2008.

GILBERTO TITERICZ JUNIOR

Uma Avaliação de Diferentes Métodos Para Transmissão de Fontes Não-Uniformes Usando Códigos Turbo

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná, como requisito parcial para a obtenção do grau de "Mestre em Ciências".
Área de Concentração: Telemática.
Orientador: Prof. Dr. Richard Demo Souza.

Curitiba
2008

Agradecimentos

Agradeço a Deus pela saúde e inspiração para realizar este trabalho.

Agradeço aos meus familiares pelo apoio.

Agradeço aos amigos e colegas que sempre me incentivaram.

Agradeço ao Professor Dr. Richard Demo Souza, pelos seus conselhos, paciência e amizade.

E agradeço especialmente minha esposa Nayara, pelo seu companheirismo, dedicação, incentivo e compreensão durante todo o tempo.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ARQ	Do inglês, Automatic Request Repeat, Pedido automático de Repetição
AWGN	Do inglês, Additive White Gaussian Noise, Ruído branco aditivo Gaussiano
BCJR	Bahl, Cocke, Jeinek e Raviv, Iniciais dos nomes dos inventores do algoritmo
BER	Do inglês, Bit Error Rate, Taxa de erro de bit
CDMA	Do inglês, Code Division Multiple Access, Acesso múltiplo por divisão de código
FER	Do inglês, Frame Error Rate, Taxa de erro de quadro
FEC	Do inglês, Forward Error Correction, Correção adiante de erros
GF	Do inglês, Galois Field, Campo de Galois
LLR	Do inglês, Log Likelihood Ratio, Taxa do logaritmo da verossimilhança
MAP	Do inglês, Maximum a Posteriori, Máximo a-posteriori
NSC	Do inglês, Non Systematic Coder, Codificador não-sistemático
NSTC	Do inglês, Non Systematic Turbo Code, Código turbo não-sistemático
OPTA	Do inglês, Optimal Performance Theoretically Archievable, Performance ótima teoricamente alcançável
PAM	Do inglês, Pulse Amplitude Modulation, Modulação por amplitude de pulso
QLI	Do inglês, Quick-look-in
RCC	Do inglês, Recursive Convolutional Coder, Codificador convolucional recursivo
RNSC	Do inglês, Recursive Non-Systematic Coder, Codificador recursivo não-sistemático
RSC	Do inglês, Recursive Systematic Coder, Codificador recursivo sistemático
SC	Do inglês, Systematic Code, Código sistemático
SCCD	Do inglês, Source-Controlled Channel Decoding, Decodificação controlada pela fonte
SNR	Do inglês, Signal-to-Noise Ratio, Relação sinal-ruído
SOVA	Do inglês, Soft-Output Viterbi Algorithm, Algoritmo de saída suave de Viterbi
STC	Do inglês, Systematic Turbo Code, Código turbo sistemático
TC	Do inglês, Turbo Code, Código Turbo
TMS	Tamanho Médio dos Símbolos
UMTS	Do inglês, Universal Mobile Telecommunication System, Sistema universal de telecomunicações móvel

Resumo

Os códigos turbo foram primeiramente propostos na codificação de fontes totalmente uniformes. Porém fontes não-uniformes são comumente encontradas no mundo real como na voz, textos e imagens. Devido a este fato muitos trabalhos recentes têm estudado codificadores turbo aplicados à codificação de fontes com redundância.

Neste trabalho avaliaremos três diferentes métodos de codificação e decodificação turbo para fontes não-uniformes, em canais submetidos a ruído AWGN e desvanecimento Rayleigh. Um dos métodos é baseado no modelo clássico de codificação separada de fonte e de canal com base no principio da separação de Shannon. Os outros dois métodos são baseados na decodificação de canal controlada pela fonte (SCCD), onde os dados não são compactados antes da transmissão e a redundância é explorada no receptor.

Os resultados obtidos através de simulações apresentaram desempenhos distintos entre os métodos. Enquanto o método da codificação separada de fonte e canal apresentou um bom desempenho para blocos grandes na região de queda, em blocos pequenos mostrou um alto patamar de erro. Já os métodos de decodificação controlada pela fonte apresentaram um patamar de erro inferior ao método da codificação separada de fonte e canal, entretanto seu desempenho na região de queda foi ligeiramente inferior ao primeiro método.

Abstract

The turbo codes had been first proposed in the codification of uniform sources. However non-uniforms sources are very commom and can be found in real world in form of voice, texts and images. Due to this fact there are many recent works studing turbo coders when used to code redundant sources.

In this work we evaluate three different methods of turbo coding and decoding for non-uniforms sources, submitted to AWGN and Rayleigh fading channels. One of the methods is based on the classic model of separated coding of source and channel based on Shannon separation principle. The others two methods are based on the source controlled channel decoding (SCCD), where the data is not compressed before the transmission and the redundancy is explored in the receiver.

The results obtained through simulations presented different performances between the methods. While the method of separate source and channel coding presented a good performance in the waterfall region for large blocks, when it is used with small blocks it showed a high error floor. Already the methods of source controlled channel decoding presented a better error floor if compared with the separate source and channel codifing method, however its performance in the waterfall region was slightly inferior to the first method.

1 Introdução

1.1 Motivações

Os códigos turbo apresentados em 1993 por C. Berrou, A. Glavieux e P. Thitimajashima em [1], representam uma das mais importantes descobertas para a área da teoria da codificação. O trabalho original de Berrou et al. demonstrou uma excelente performance destes códigos para fontes uniformes, transmitidas por canais submetidos a ruído branco aditivo Gaussiano (AWGN). A performance apresentada foi de apenas 0,7dB de distância do limite máximo de Shannon para um código de taxa $\frac{1}{2}$, para uma taxa de erro de bit (BER) de 10^{-5}. Devido a esta excelente performance a comunidade científica ficou em dúvida da validade dos resultados e o reconhecimento aos seus criadores veio somente após a comprovação dos resultados por outros pesquisadores.

A maioria dos trabalhos sobre codificação e controle de erros assume que a entrada do codificador de canal é uniforme, ou seja, apresenta uma seqüência binária e sem memória com distribuição uniforme de bits. Entretanto é muito natural no mundo real encontrar fontes com algum tipo de redundância, como a fala, sons, textos e imagens, onde esta redundância aparece na forma de não-uniformidade dos símbolos da fonte. Nesta linha, trabalhos recentes começaram a estudar códigos turbo aplicados na transmissão de fontes não-uniformes como em [2], [3], [4] e [5].

De acordo com Shannon em [6], o esquema ótimo de codificação seria separar o processo em duas partes: a primeira seria a compressão da fonte até o seu limite teórico dado pela entropia e a segunda seria obter um código de canal que alcançasse a capacidade do canal. Entretanto este esquema é ótimo se não levarmos em conta a complexidade, pois na prática existem várias limitações, como o tamanho variável do bloco codificado e o resíduo de redundância presente mesmo após a codificação, pois para obter uma distribuição perfeitamente uniforme seria necessário uma fonte com tamanho tendendo ao infinito. Em contrapartida, Shannon também cita em [6], que se uma fonte possui redundância e nada é feito para eliminá-la, então esta redundância ajudará a combater o ruído. Muitos trabalhos recentes como em [4], [5], [7] e [8] têm

utilizado a redundância presente na fonte através do método da *decodificação de canal controlada pela fonte* (SCCD). Neste método a redundância presente na fonte serve para auxiliar o decodificador de canal e conseqüentemente melhorar a performance do sistema.

Nesta linha, em [3] é mostrado uma busca de códigos sistemáticos para transmissão de fontes não-uniformes. Porém é observado em [9] que códigos não-sistemáticos têm uma grande vantagem sobre os sistemáticos na transmissão de fontes não-uniformes. Isto se deve ao fato de que os códigos sistemáticos resultam numa seqüência codificada que não é uniformemente distribuída para fontes com redundância. Já com códigos não-sistemáticos é possível obter uma seqüência uniformemente distribuída, e portanto, que se aproxima da capacidade de canal, mesmo para fontes altamente não-uniformes.

A falta de uma comparação justa entre os métodos de tratamento de fontes não-uniformes usando códigos turbo levou à elaboração deste trabalho. O objetivo do estudo é obter uma comparação entre três métodos diferentes submetidos a duas distribuições distintas de fontes não-uniformes. O primeiro método utiliza o processo de codificações de fonte e canal separados, com código Huffman aplicado à fonte e código turbo sistemático ao canal. O segundo utilizando SCCD e codificador turbo não-sistemático. E por fim um método que utiliza SCCD com codificador turbo não-sistemático e alocação desigual de energia no canal.

1.2 Estrutura da Dissertação

O restante desta dissertação encontra-se organizada da seguinte forma:

- Capítulo 2 - Sistemas de Comunicação.

  Neste capítulo é apresentada uma visão geral sobre sistemas de comunicações digitais. Mostra também os modelos de canais AWGN e Rayleigh, bem como o cálculo da capacidade de cada modelo de canal e o seu limite de Shannom. Explica o que é a entropia da fonte e como pode ocorrer a codificação de fonte e de canal.

- Capítulo 3 - Códigos Turbo.

  Neste capítulo é apresentado o conceito geral dos Códigos Turbo, tais como os codificadores sistemáticos, o entrelaçador, o puncionamento e o decodificador turbo. Mostra também o conceito de códigos turbo não-sistemáticos e, por fim, explica um algoritmo muito utilizado para decodificação turbo.

- Capítulo 4 - Codificação Turbo para Fontes Não-Uniformes.

Neste capítulo são apresentados os três métodos avaliados neste trabalho para codificação de fontes não-uniformes. O primeiro baseia-se na codificação separada da fonte e do canal, o segundo na decodificação controlada pela fonte usando códigos turbo não-sistemáticos e o último na decodificação controlada pela fonte usando códigos não-sistemáticos e alocação desigual de energia.

- Capítulo 5 - Simulações.

  Neste capítulo são apresentados os detalhes de cada código turbo estudado, como a composição dos codificadores constituintes, a taxa utilizada, o tamanho do bloco da fonte, a distribuição da fonte e alguns outros detalhes de cada método. Por fim são mostrados os resultados gráficos das simulações com cada método, e uma avaliação geral sobre o desempenho individual.

- Capítulo 6- Discussão e Conclusões Gerais.

  Neste capítulo revisamos os principais resultados deste trabalho, bem como propomos alguns trabalhos futuros.

2 *Sistemas de Comunicação*

2.1 Modelo do Sistema de Comunicação

A Figura 1 ilustra alguns elementos básicos de um sistema de comunicação digital em banda básica conforme definido em [10]. A Fonte de Informação produz as mensagens que serão posteriormente convertidas em uma seqüência de dígitos binários. Esta Fonte de Informação é dita uniforme se as mensagens produzidas por ela não possuem redundância, ou seja, é representada pela menor quantidade de símbolos possível. Fontes não-uniformes são aquelas onde existe a predominância de um símbolo sobre o outro. Este tipo de fonte é muito comum na natureza e mesmo no mundo digital ela pode estar presente de diversas formas, como por exemplo, em imagens ou sons, onde quase sempre existe muita redundância. O Codificador de Fonte é o dispositivo que converte efetivamente a fonte em uma seqüência digital binária. Este Codificador de Fonte pode executar também a compressão dos dados. Esta seqüência binária digital é então passada para o Codificador de Canal, que por sua vez tem como função inserir redundância na seqüência binária de forma controlada, a fim de que o receptor possa usar esta informação para sobrepujar os efeitos do ruído e da interferência causados pelo canal. A Taxa r do código é definida como k/n, onde n representa o tamanho da seqüência após a inserção de redundância e k o tamanho antes da inserção.

Após a codificação de canal a seqüência de bits será convertida pelo Mapeador de Canal ou Modulador Digital. Este bloco serve de interface com o canal de comunicação. O propósito dele é mapear a seqüência de informação digital em uma forma de onda para ser entregue ao canal. Na modulação binária o modulador digital mapeia cada bit em uma forma de onda. Entretanto o modulador poderia mapear M seqüências de bits, possuindo $M = 2^m$ formas de ondas, uma forma de onda para cada uma das 2^m possibilidades de seqüência de bits. Isto é chamado modulação M-ária, quando M>2. Se M=2 é chamada de modulação binária. Uma vez mapeada a seqüência de bits é transmitida via o canal de comunicação. Este canal é o meio físico que é usado pela transmissão. Na comunicação sem fio o canal pode ser a atmosfera ou o espaço livre. Na comunicação via telefone é usada uma variedade de meios físicos como:

Figura 1: Modelo Básico de um Sistema de Comunicação.

par de fios de cobre, fibra ótica e rádio. Entretanto independente do meio físico escolhido este canal se caracteriza por estar constantemente submetido a diversos tipos de ruídos e distorções aleatórias.

O bloco receptor encarregado de realizar o primeiro processamento no sinal que chega do canal é o Demapeador de Canal ou Demodulador Digital. Nesta etapa o sinal corrompido pelas interferências do meio é reduzido a uma seqüência de números que representam uma estimativa dos símbolos transmitidos, sejam binários ou M-ários. Esta seqüência então é passada pelo Decodificador de Canal, que por sua vez tenta reconstituir a seqüência original usando o conhecimento que ele tem sobre o codificador usado e a redundância inserida na seqüência de dados recebida. O passo final é o Decodificador de Fonte que tenta reconstruir o sinal original dada a seqüência apresentada pelo Decodificador de Canal e o conhecimento do método usado para a Codificação de Fonte.

2.2 Modelos Matemáticos Para os Canais de Comunicação

Nos modelos de sistemas de comunicação é importante definir as características do meio de transmissão. Neste trabalho serão usados dois modelos de canais físicos que serão mostrados nos próximos capítulos, o canal submetido ao ruído branco aditivo *Gaussiano* e o canal submetido ao desvanecimento *Rayleigh*.

2.2.1 Canal AWGN

O Ruído Branco Aditivo Gaussiano (AWGN) é o padrão de ruído mais utilizado em simulações por apresentar resultados muito próximos à solução exata [11]. Ele ganhou este nome devido ao fato de ter banda infinita e possuir uma distribuição Gaussiana. Conforme [12], o ruído AWGN é aditivo e o dito canal AWGN pode ser representado pela equação:

$$y = x + n \,, \tag{2.1}$$

onde x representa o sinal da saída do transmissor, n o ruído e y o sinal corrompido pelo ruído que chega na entrada do receptor. Seu diagrama em blocos é representado pela Figura 2. Este ruído é caracterizado, principalmente, pelo ruído térmico presente em circuitos de receptores eletrônicos. O canal AWGN pode representar bem um canal de comunicação via satélite ou um canal com fio sem seletividade em freqüência.

Figura 2: Modelo do Canal AWGN.

2.2.2 Canal Rayleigh

O Canal Rayleigh é usado principalmente em modelos onde deseja-se o efeito de desvanecimento, conforme introduzido em [12]. Este efeito deve-se principalmente ao fato do meio apresentar multi-percursos para o sinal. Estes multi-percursos fazem com que a soma vetorial dos sinais que chegam ao receptor possa apresentar uma interferência construtiva ou destrutiva, causando assim uma alteração na potência do sinal. Esta alteração chama-se desvanecimento e apresenta-se conforme uma distribuição Rayleigh [10]. O canal com desvanecimento ou canal Rayleigh, é definido como o produto do sinal transmitido por um coeficiente complexo de desvanecimento, conforme a seguinte equação:

$$y = h \cdot x + n \,, \tag{2.2}$$

onde y é o sinal corrompido pelo canal, n o ruído aditivo Gaussiano e h a componente multiplicativa do canal. A Figura 3 ilustra este canal.

O ruído AWGN também faz parte do modelo e é adicionado após o efeito de desvanecimento.

Figura 3: Modelo do Canal Rayleigh.

2.3 Entropia da Fonte

No início dos estudos da teoria da informação [6], Shannon precisava de um método para medir a quantidade de informação em um conjunto de símbolos ou mensagens e, a partir da definição de entropia usada na física ele definiu a entropia na teoria da informação como sendo a média de informação contida em um conjunto de símbolos.

Considerando uma fonte S com uma seqüência de símbolos finitos representados por $s_1, s_2, ...s_n$, onde cada símbolo possui uma probabilidade $p(s_x)$ de ocorrer, a quantidade de informação própria representada por cada símbolo é definida como [6]:

$$I(s_x) = \log_2 \frac{1}{p(s_x)} \ . \qquad (2.3)\text{n}$$

Traçando esta função obtemos o gráfico da Figura 4. Observando o máximo da função podemos verificar que a entropia é máxima somente quando a fonte é uniforme, ou seja, p_0 = p_1 = 0,5.

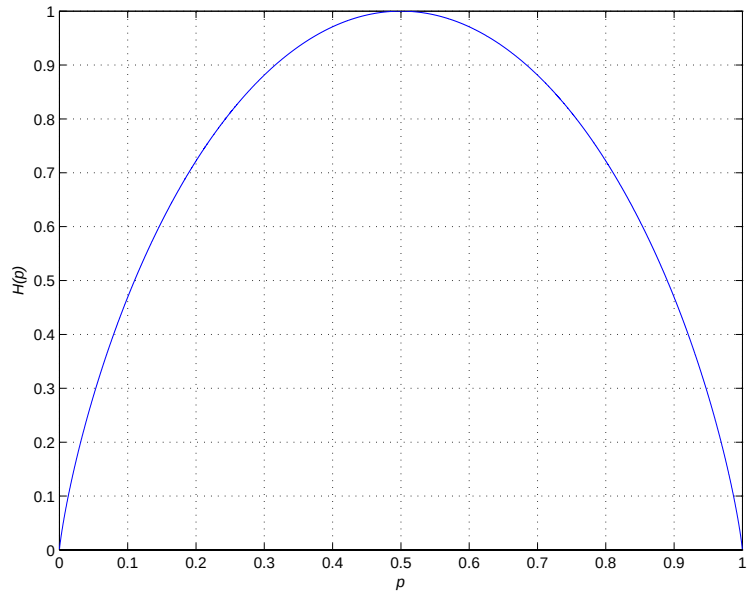

Figura 4: Entropia em função da distribuição de uma fonte binária.

A entropia condicional de um canal é a incerteza que resta sobre a entrada de um canal *X* após a observação da sua saída *Y*. Ela é calculada através da equação:

$$H(X|Y=y) = p(x|y) \cdot \log_2 \frac{1}{p(x|y)},$$

onde $p(x|y)$ é uma probabilidade condicional do canal, que nada mais é que a probabilidade de certeza de que a saída de um canal seja *Y*, dado que sua entrada foi *X*, ou seja, é a probabilidade de transição $p(X|Y)$ do canal.

2.4 Capacidade do Canal

A Capacidade de Canal (C) é definida como a máxima quantidade de informação que pode ser transmitida por um canal específico [6]. Para que a comunicação por este canal possa ser estabelecida com uma taxa de erros tão pequena quanto se queira, a taxa r [bits/s] da informação a ser transmitida pelo canal deve ser menor que a capacidade C [bits/s] do canal.

A capacidade de um canal sem memória com entrada discreta é definida como sendo a

máxima informação mútua entre a entrada X e a saída Y do canal, tal que:

$$C = max_{p(x)} I(X;Y)$$

e a maximização deve ser feita para todos as distribuições de entrada $p(x)$. A informação mútua do canal $I(X;Y)$, conforme [12], é calculada com base na entropia condicional do canal e representa a incerteza da entrada do canal que é resolvida observando-se a sua saída. A informação mútua é definida como:

$$I(X;Y) = H(X) - H(X|Y)$$

Como já foi mostrado em [9], quando o canal é simétrico a capacidade alcançável para um canal AWGN é dada pela equação:

$$C_{AWGN} = 1 - \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} . \log_2(1 + e^{\frac{2y}{\sigma^2}}) . e^{-\frac{(y+1)^2}{2\sigma^2}} . dy \, , \tag{2.5}$$

e quando o canal é Rayleigh a capacidade de canal é dada por:

$$C_{Ray} = 1 - \int_{0}^{\infty} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} . \log_2(1 + e^{\frac{2ay}{\sigma^2}}) . 2ae^{-a^2} . e^{-\frac{(y+a)^2}{2\sigma^2}} . dy . da \, , \tag{2.6}$$

onde σ^2 representa a variância da densidade espectral de potência do ruído, dada por $\sigma^2 = \frac{N_0}{2}$, e a representa a constante de amplitude do canal submetido ao desvanecimento Rayleigh.

Quando uma fonte não-uniforme, onde $p_0 \neq \frac{1}{2}$, é transmitida por um canal, não é possível explorar a capacidade máxima deste canal. A capacidade alcançável para este tipo de fonte em um canal AWGN fica dependente da sua distribuição e conseqüentemente a equação de capacidade fica, conforme visto em [9], da seguinte forma:

$$\begin{aligned} C_{AWGN}^{bias} = & -p_0 \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \log_2(p_0 + p_1 e^{\frac{2y}{\sigma^2}}) e^{-\frac{(y+1)^2}{2\sigma^2}} dy \\ & -p_1 \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \log_2(p1 + p_0 e^{\frac{-2y}{\sigma^2}}) e^{-\frac{(y-1)^2}{2\sigma^2}} dy \end{aligned} \tag{2.7}$$

e analogamente para o canal Rayleigh:

$$\begin{aligned} C_{Ray}^{bias} = & -p_0 \int_{0}^{\infty} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \log_2(p_0 + p_1 e^{\frac{2ay}{\sigma^2}}) 2ae^{-a^2} . e^{-\frac{(y+a)^2}{2\sigma^2}} dyda \\ & -p_1 \int_{0}^{\infty} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \log_2(p_1 + p_0 e^{\frac{-2ay}{\sigma^2}}) 2ae^{-a^2} . e^{-\frac{(y-a)^2}{2\sigma^2}} dyda \end{aligned} \tag{2.8}$$

Estas equações comprovam que quando se transmite uma fonte não-uniforme diretamente por uma canal simétrico, é impossível alcançar a capacidade máxima deste canal. A Tabela 1

ilustra a capacidade de um canal AWGN para diferentes valores de $\frac{E_s}{N_0}$, onde E_s é a energia de símbolo de canal. O valor de $\frac{E_s}{N_0}$ em dB é dito como a relação sinal-ruído(SNR). Nesta tabela é fácil notar que para a fonte não-uniforme onde $p_0 = 0,9$ a capacidade de canal é sempre menor do que no caso da fonte uniforme. O mesmo fenômeno acontece para canais Rayleigh como pode ser visto na Tabela 2. Isto comprova que só é possível obter o melhor aproveitamento da capacidade destes canais quando são transmitidas fontes uniformes por eles.

Tabela 1: Capacidade de Canal para um Canal AWGN.

SNR(dB)	$C_{AWGN}(p_0 = p_1 = 0,5)$	$C_{AWGN}^{bias}(p_0 = 0,9)$
-4	0,415	0,172
-2	0,564	0,242
0	0,721	0,320
+2	0,860	0,392
+4	0,951	0,442

Tabela 2: Capacidade de Canal para um Canal Rayleigh.

SNR(dB)	$C_{Ray}(p_0 = p_1 = 0,5)$	$C_{Ray}^{bias}(p_0 = 0,9)$
-4	0,348	0,147
-2	0,454	0,197
0	0,566	0,250
+2	0,671	0,302
+4	0,763	0,348

2.5 Limite Teórico Máximo de Transmissão

Baseado nos teoremas de Shannon em [6], é possível traçar um limite teórico máximo de transmissão (OPTA, do inglês *Optimal Performance Theoretically Archievable*) que nos mostra o valor mínimo de $\frac{E_b}{N_0}$ no qual, teoricamente, pode haver uma transmissão sem erros por um canal específico. Este limite é calculado baseado na capacidade do canal C, taxa de transmissão r e em função da taxa de distorção R, conforme proposto por Shannon. Portanto, dado uma fonte sem memória e um canal sem memória com capacidade C, a fonte pode ser transmitida via o canal a uma taxa de transmissão $r = \frac{r_c}{r_s}$ [símbolos da fonte/símbolos do canal], onde r_c representa a taxa do código do canal e r_s a taxa do código da fonte, e reproduzida no receptor com uma taxa de erro P_{BER}, apenas se a seguinte condição for aceita:

$$r.R(P_{BER}) < C\,, \tag{2.9}$$

onde $R(P_{BER})$ é a taxa de distorção em função da probabilidade de erro. Para uma fonte binária não-uniforme e com distribuição p_0, temos que P_{BER} é a taxa de erro de bit (BER), então conforme [6], $R(P_{BER})$ torna-se:

$$R(P_{BER}) = \begin{cases} H(p_0) - H(P_{BER}), & 0 \leq P_{BER} \leq min\{p_0, 1-p_0\} \\ 0 & P_{BER} > min\{p_0, 1-p_0\} \end{cases}, \tag{2.10}$$

onde $H(x)$ é a função de entropia.

Substituindo em (2.9) o valor definido de r, dada a distribuição da fonte e calculando o valor de $R(P_{BER})$ dado um valor de taxa de erro de referência (ex.$P_{BER} = 10^{-5}$), é possível calcular o valor ótimo de C. Dado este valor de C é possível usar as devidas equações de capacidade de canal para calcular o valor de $\frac{E_b}{N_0}$ que corresponde ao OPTA do sistema. Devido à complexidade das integrações, estes cálculos são realizados normalmente por integração e aproximação numérica.

Alguns valores do OPTA para $P_{BER} = 10^{-5}$ podem ser vistos nas Tabelas 3 e 4, para os canais AWGN e Rayleigh respectivamente.

Tabela 3: OPTA para canais AWGN em $P_{BER} = 10^{-5}$.

	OPTA (dB)	
p	$r = \frac{1}{2}$	$r = \frac{1}{3}$
0,700	-0,620	-1,189
0,800	-1,810	-2,244
0,83079	-2,347	-2,899
0,899063	-4,103	-4,394
0,900	-4,140	-4,399

Tabela 4: OPTA para canais Rayleigh em $P_{BER} = 10^{-5}$.

	OPTA (dB)	
p	$r = \frac{1}{2}$	$r = \frac{1}{3}$
0,700	0,760	-0,339
0,800	-0,730	-1,559
0,83079	-1,382	-2,304
0,899063	-3,433	-3,964
0,900	-3,470	-3,994

2.6 Codificação de Fonte

A idéia de codificação de fonte veio da teoria de Shannon, onde ele demonstrou que, a princípio, sempre é possível representar a informação gerada por uma fonte a uma taxa igual a da sua entropia. Isto inspirou o estudo de métodos que codifiquem a fonte a fim de se obter uma saída com a taxa próxima a da sua entropia. Prova deste estudo é o código desenvolvido por David A. Huffman em 1952 que utiliza-se da redundância da fonte para gerar um código de comprimento ótimo.

2.6.1 Código de Huffman

O código de Huffman é bastante utilizado por ser uma forma de compressão sem perda de dados. Conforme ele foi proposto no artigo original de Huffman [13] de 1952, ele é um método que utiliza um algoritmo de compressão que explora propriedades da entropia para gerar códigos de tamanho variável, ou seja, aproveita a estatística do conjunto de dados a ser compactado para determinar os códigos de cada símbolo. As probabilidades de ocorrência de cada símbolo são computadas em todos os possíveis valores na fonte e esses são então ordenados. O menor código é associado aos símbolos com maior probabilidade de ocorrer, e assim por diante. Visto que o código Huffman possui tamanho variável, a associação de códigos é feita pela relação a menor probabilidade recursivamente até que uma árvore binária seja gerada com uma raiz de duas probabilidades. Este código pode então ser empregado na codificação de fontes.

O algoritmo que constrói o código Huffman é bastante simples conforme pode ser comprovado em [14]. Para atribuir aos símbolos mais freqüentes as codificações menores e, como geralmente é usado código binário, constrói-se uma árvore binária a partir dos símbolos de menor valor de probabilidade, onde os nós representam a soma da freqüência de ocorrência de todos os símbolos da sub-árvore correspondente, e cujas folhas representam cada um dos símbolos da mensagem a ser codificada e sua freqüência. A codificação de cada símbolo será a representação do caminho da raiz até o símbolo. Então, o algoritmo de construção do código Huffman pode ser especificado da seguinte forma:

1. O alfabeto de entrada deve ser organizado em uma coluna em ordem decrescente de freqüência com sua probabilidade ou o seu número de ocorrência no texto.

2. Definir um novo nó unindo dois ramos cuja soma das probabilidades seja a menor possível.

3. Calcular a probabilidade deste novo nó como a soma das probabilidades dos grupos unidos (nós).

4. Para um dos nós filho usado para definir o novo nó atribuir o bit 0 e para o outro o bit 1.

5. Se restar mais de um grupo (sub-árvores) ir para o passo 2, senão seguir para o passo 6.

6. Obter o código de cada símbolo seguindo a seqüência de bits atribuída aos grupos.

Como o processo constitui uma estrutura de árvore binária, o código de cada elemento será a seqüência de bits da raiz até a folha que corresponde ao símbolo ou caractere k. O símbolo de maior freqüência possui o menor comprimento que é exatamente a meta da compressão estatística. Assim acontece com os demais símbolos, em ordem crescente do número de bits. Portanto, o tamanho médio dos símbolos (*TMS*) no código Huffman para um alfabeto de n símbolos é dado pela somatória dos produtos do tamanho de cada símbolo $t(k)$ pela sua probabilidade $p(k)$, conforme a equação:

$$TMS = \sum_{k=1}^{n} t(k).p(k) \ . \tag{2.11}$$

Um exemplo de compressão usando Huffman pode ser dado considerando uma *string* de texto onde aparecem somente os caracteres do conjunto $K = \{a, b, c, d\}$. Cada símbolo ou caractere apresenta uma probabilidade de aparecimento diferente na $string$ de texto, conforme a Tabela 5 abaixo:

Tabela 5: Probabilidade de cada símbolo.

Símbolo	$\boldsymbol{p(k)}$
a	40%
b	35%
c	20%
d	5%

Usando o algoritmo de Huffman é criada a árvore binária da Figura 5. O lado esquerdo da figura representa a raiz da árvore e apresenta todas as possibilidades de símbolos do conjunto. Cada nó da raiz possui a somatória das probabilidades dos nós a esquerda até chegar ao último nó da direita onde a probabilidade é sempre 1. Percorrendo os ramos da raiz da direita para a esquerda de modo que chegue até o símbolo desejado é formada a palavra codificada agregando o símbolo 0 ou 1 de cada ramo. Fazendo isso para cada símbolo é então construída a tabela de codificação conforme a Tabela 6.

Figura 5: Árvore de Huffman para o exemplo.

Tabela 6: Código Huffman para o exemplo.

Caracter	**Código**
a	0
b	10
c	110
d	111

Uma vez construída a tabela a codificação Huffman, a codificação consiste apenas de substituir os caracteres do alfabeto pelos símbolos binários correspondentes da tabela. Para o exemplo dado o *TMS* será $0,4 \cdot 1 + 0,35 \cdot 2 + 0,20 \cdot 3 + 0,05 \cdot 3 = 1,85$ bits/símbolo.

2.7 Codificação de Canal

Em qualquer tipo de transmissão, o canal físico corrompe o sinal transmitido devido ao ruído inerente do circuito eletrônico ou distorções do meio físico. Isto pode induzir erros no sinal, e assim, a mensagem originalmente transmitida não pode ser totalmente reconstruída no receptor. Para prevenir isso, o codificador de canal pode fazer a adição controlada de redundância para que a mesma possa ser explorada no decodificador de canal, a fim de corrigir possíveis erros. Por outro lado o receptor pode, também, mandar uma mensagem ao transmissor pedindo para que ele repita a mensagem original. Ambas as técnicas ajudam no controle de erros.

As técnicas de correção de erros podem ser classificadas em dois grupos: *Forward Error Correction* (FEC), onde se utilizam códigos corretores de erro para fazer a correção no receptor e *Automatic Repeat reQuest* (ARQ), que na ocorrência de um erro, detectado por um código detector de erros, emite um pedido de retransmissão da mensagem ou de parte dela. Caso não houvessem os códigos corretores de erro, o número de pedidos de retransmissão seria muito

alto, fazendo com que a taxa de transmissão do sistema caia demais.

Classicamente os códigos corretores de erro podem ser divididos em dois grandes grupos: códigos de bloco e códigos convolucionais, sendo que apenas este último será estudado neste trabalho. Dentro de cada grupo existe uma vastidão de tipos de códigos para as mais diversas situações.

2.7.0.1 Códigos Convolucionais

Os códigos convolucionais foram introduzidos em 1955 por Elias em [15], como uma alternativa para os códigos de bloco, mas somente em 1967 Viterbi em [16] propôs um algoritmo de decodificação que era simples o suficiente para ser implementado na época e acabou sendo muito utilizado, tornando-se famoso a partir da década de 70.

Um código convolucional é um tipo de código corretor de erro em que cada conjunto de k símbolos de entrada é transformado em um conjunto de n símbolos de saída, onde essa transformação é função dos últimos m símbolos na entrada do codificador, e a razão $\frac{k}{n}$ é conhecida como taxa do código r_c. A variável m indica também a quantidade de memória do codificador. A representação dos codificadores convolucionais é dada por $\mathsf{C}(n, k, m)$.

Eles recebem o nome de codificador convolucional porque executam uma operação de convolução entre o vetor de entrada $\overline{u}$ e vetores geradores do código $\overline{g}$. Existe um vetor gerador $\overline{g}$ para cada saída do código. Portanto, dado uma seqüência de bits de entrada $\overline{u}$ = (u_0, u_1, ...), as seqüências geradoras $\overline{g}^{(i)}$ tal que $0 \leq i \leq n$ e as seqüências de saída $\overline{c}^{(i)}$ tal que $0 \leq i \leq n$, a operação que define as saídas é $\overline{c}^{(i)} = \overline{u}^{(i)} * \overline{g}^{(i)}$ para cada saída do codificador, onde $*$ representa o operador de convolução.

Na prática o processo de codificação é feito pela combinação linear, definida em GF(2), das saídas de registradores de deslocamento de m estágios. Os códigos convolucionais também permitem o uso do método de puncionamento, vide [17], que consiste em excluir periodicamente algumas saídas do codificador, a fim de aumentar sua taxa.

Existem inúmeras formas de representar um código convolucional. Veremos então algumas representações possíveis do código $\mathsf{C}(2, 1, 2)$, definido pelos vetores geradores $\overline{g}^{(0)}$ = [1 0 1] e $\overline{g}^{(1)}$ = [1 1 1]. Sua representação mais famosa é a máquina de estados finitos. Esta representação apresenta uma definição de estado atual e uma saída entre os estados. O estado atual da máquina depende do conteúdo das memórias e a saída do conteúdo dos somadores, portanto a máquina de estados que represente este codificador pode ser ilustrada como a da Figura 6, onde $\overline{u}$ representa os $bits$ de entrada, $\overline{c}_0$ e $\overline{c}_1$ os $bits$ de saída e o estado atual da máquina está contido dentro de

cada circunferência.

Figura 6: Máquina de estados do código $\mathbb{C}(2,1,2)$.

Outra representação mais prática pode ser feita utilizando uma analogia a um circuito digital, onde existem m registradores de deslocamento e n somadores módulo-2 interligados formando um codificador convolucional equivalente, como o da Figura 7. Esta representação mostra também a possibilidade de puncionamento entre as saídas $\overline{c}_0$ e $\overline{c}_1$ através de uma chave seletora c_p na saída do circuito.

Figura 7: Circuito equivalente do código $\mathbb{C}(2,1,2)$.

A representação do mesmo código $\mathbb{C}(2,1,2)$ pode ser feita matematicamente utilizando uma matriz de convolução do tipo,

$$G = \begin{bmatrix} G_0 & G_1 & ... & G_m & 0 & ... & 0 \\ 0 & G_0 & G_1 & ... & G_m & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & G_0 & G_1 & ... & G_{m-1} & G_m \end{bmatrix},$$

onde G_i são matrizes k x n cujos elementos especificam se os registradores de deslocamento estão conectados (1) ou não estão conectados (0) aos somadores módulo-2 que definem as saídas. Para o código $\mathbb{C}(2,1,2)$ as sub-matrizes são $G_0 = [11], G_1 = [01]$ e $G_2 = [11]$.

Por fim outra representação dos códigos convolucionais é a da treliça, que é utilizada no decodificador de Viterbi e pode ser vista na Figura 8. Esta representação mostra todos os estados possíveis do código, os próximos estados e as respectivas saídas do codificador para cada transição de estado.

Figura 8: Código $\mathbb{C}(2, 1, 2)$ em forma de treliça.

Se o codificador usa a própria fonte $\overline{u}$ como uma de suas saídas codificadas ele é chamado de Codificador Sistemático (SC). Caso ele use apenas a paridade gerada pelos somadores módulo-2 ele é chamado de Codificador Não-Sistemático (NSC). Um codificador convolucional onde parte da paridade gerada é inserida novamente na entrada do codificador, é chamado de Codificador Recursivo, como pode ser visto na Figura 9. Se um codificador recursivo usar a informação da fonte como uma de suas saídas ele é então chamado de Codificador Recursivo Sistemático (RSC) e se ele usar apenas a paridade como saída então ele é chamado de Codificador Recursivo Não-Sistemático (RNSC).

Figura 9: Codificador Convolucional Recursivo.

Uma das propriedades que determina o desempenho dos códigos convolucionais é a distância livre (d_{free}). Esta distância corresponde ao peso da palavra de menor peso gerada pelo codificador com base em uma palavra de entrada que não seja somente de zeros.

Dentre os vários métodos de decodificação dos códigos convolucionais o mais usado até hoje é o algoritmo de Viterbi. Ele usa uma treliça com todos os estados possíveis do código e aliada a um algoritmo de decodificação por máxima verossimilhança, obtém uma performance ótima na decodificação (em termos de erro de palavras).

2.7.0.2 Códigos Turbo

Em 1993, Berrou, Glaviex e Thitimajshima em [1] introduziram uma nova técnica, baseada numa associação em paralelo de dois códigos convolucionais. Sua característica fundamental é a decodificação iterativa, onde cada um dos estágios decodificadores alimenta o outro num processo de refinamento da informação. Essa nova classe de códigos, que foi denominada Códigos Turbo (o uso da palavra turbo tem origem na característica de realimentação da informação obtida em um estágio no outro, semelhante à realimentação do turbo em motores a combustão), atingiu ganhos de codificação próximos ao limite estabelecido por Shannon, ao custo de uma maior complexidade computacional na decodificação. Esse desempenho desencadeou em uma série de pesquisas na área e levou à adoção dos códigos turbo como padrão para o sistema de terceira geração de telefonia móvel (UMTS na Europa e Japão e CDMA-2000 nos EUA) e, devido ao seu alto ganho de codificação, os Códigos Turbo são vistos como fortes candidatos a serem adotados como padrão das futuras gerações de redes sem fio.

Códigos Turbo serão melhor explicados no Capítulo 3, porém aqui será apresentada uma visão geral. O codificador turbo original é construído através da concatenação paralela de dois codificadores convolucionais recursivos sistemáticos. Estes dois codificadores, usam a mesma seqüência de bits $\overline{u}$, mas em ordem diferente.

A concatenação dos códigos turbo originais é dita paralela, pois, ambos codificadores têm como entrada a seqüência de informação. Os codificadores trabalham, desse modo, de forma independente. A independência na codificação é fundamental no processo iterativo da decodificação. O decodificador turbo proposto por Berrou em [1] é um decodificador iterativo baseado na troca de informação entre dois estágios de decodificação.

2.8 Comentários Finais

Este capítulo apresentou o modelo do sistema de comunicação estudado. Mostrou a influência que os canais AWGN e Rayleigh trazem ao sistema, bem como a capacidade do canal e o limite de transmissão máximo teórico proposto por Shannon.

Apresentou o código Huffman que é o modelo de codificação de fonte utilizada neste trabalho, bem como discutiu a codificação de canal utilizando códigos convolucionais e códigos turbo. Mostrou que no processo de codificação, o codificador convolucional usa códigos convolucionais para codificar a fonte e um detetor de seqüência de máxima verossimilhança (Viterbi) para proceder a decodificação. Já o código turbo utiliza a concatenação de códigos convolucionais e um entrelaçador para construir um codificador, e na decodificação, apresenta uma malha complexa composta por blocos decodificadores e desentrelaçadores e uma realimentação, que é a chave da eficiência do código Turbo.

3 *Códigos Turbo*

3.1 Introdução

Em 1993 um grupo de pesquisadores apresentou uma nova classe de códigos corretores de erros em conjunto com uma técnica de decodificação iterativa. Estes códigos foram chamados de *Códigos Turbo* (TC) e mostraram ser capazes de alcançar uma performance distante apenas 0,7dB do limite de capacidade de Shannon, como mostrado no trabalho pioneiro de Berrou, Glaviex e Thitimajshima em [1]. Desde a sua introdução, os códigos turbo foram usados para aplicações limitadas em potência como comunicações no espaço profundo e via satélite, bem como para aplicações onde existe limitação por interferência, como o caso da terceira geração de celulares.

O codificador turbo consiste em dois ou mais códigos convolucionais em paralelo com suas saídas separadas por entrelaçadores pseudo-aleatórios. O decodificador turbo consiste de dois decodificadores de máxima a posteriori (MAP) conectados em série via um entrelaçador e com um laço de realimentação da saída do segundo para a entrada do primeiro decodificador. Este capítulo irá descrever a estrutura dos codificadores e decodificadores usados em códigos turbo, bem como o algoritmo de decodificação. Todos os esquemas de codificação neste capítulo assumirão uma modulação binária em fase (BPSK).

3.2 Codificadores Turbo Sistemáticos

A Figura 10 mostra a estrutura padrão do codificador turbo introduzida por Berrou em [1]. A variável u_k representa a entrada do codificador e as variáveis c_k as saídas. Este codificador turbo consiste na concatenação de dois ou mais codificadores convolucionais recursivos e sistemáticos (RSC) representados por ENC1 e ENC2 e interligados por um entrelaçador de comprimento *N* bits. Cada um destes codificadores representa um codificador constituinte do código turbo. O polinômio que define cada codificador constituinte pode ser igual ou diferente em cada um deles. Quando os codificadores constituintes forem diferentes, então o codificador

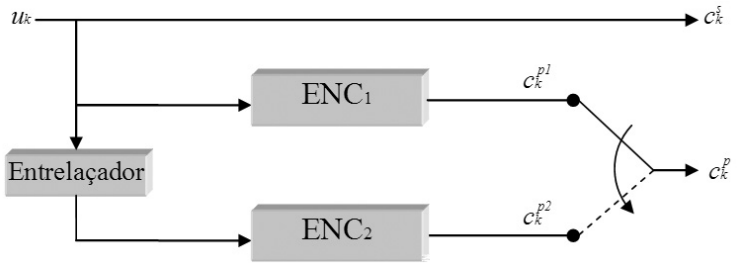

Figura 10: Codificador Turbo Sistemático Original.

turbo é chamado de assimétrico. Se for usado mais de dois codificadores constituintes, então, cada novo codificador receberá um entrelaçador distinto em sua entrada. A concatenação é dita paralela pois a mesma seqüência de entrada, proveniente da fonte, é aplicada em cada codificador constituinte.

O entrelaçador tem como função básica permutar os bits da seqüência de entrada a fim de que sejam geradas versões diferentes desta mesma seqüência. Isso é feito para que cada codificador constituinte receba o mesmo bloco de informações proveniente da fonte, porém cada um com uma seqüência distinta, gerando paridades diferentes entre si para o mesmo bloco de informação.

O codificador pode ainda aumentar a taxa de transmissão puncionando as saídas de paridade dos codificadores. Para isso existe um mecanismo que através de um padrão pré-determinado de puncionamento, escolhe qual saída de paridade será usada em cada intervalo de tempo. Como pode ser visto na Figura 10, se forem usadas as saídas c_k^s, c_k^{p1} e c_k^{p2}, é obtido um codificador turbo com taxa $\frac{1}{3}$. E se forem puncionadas as saídas c_k^{p1} e c_k^{p2} com um padrão pré-definido de alternância entre as saídas é possível obter uma taxa de $\frac{1}{2}$.

O processo de codificação é relativamente simples, se comparado com o processo de decodificação. Os códigos turbo trabalham com blocos de informações do mesmo comprimento do seu entrelaçador. Esta característica se deve ao fato de que o entrelaçador só pode aplicar as *N* permutações sobre o bloco após recebê-lo completamente. Como a codificação é sistemática o bloco de informações provenientes da fonte, representado pela seqüência $u_k = (u_0, ..., u_{N-1})$, corresponde diretamente à saída c_k^s do codificador, portanto:

$$u_k = c_k^s = (c_0^s, c_1^s, ..., c_{N-1}^s)$$

O primeiro codificador constituinte receberá u_k em sua entrada e gera a primeira seqüência de paridade:

$$c_k^{p1} = (c_0^{p1}, c_1^{p1}, ..., c_{N-1}^{p1})$$

O segundo codificador constituinte possui o entrelaçador em sua entrada, portanto ele receberá a versão entrelaçada de u_k, representada por $u_k^{'}$ e gerará a segunda seqüência de paridade:

$$c_k^{p2} = (c_0^{p2}, c_1^{p2}, ..., c_{N-1}^{p2})$$

Para este codificador a palavra código gerada é representada pela concatenação das três saídas c_k^s, c_k^{p1} e c_k^{p2} e é representada por:

$$c = (c_0^s, c_0^{p1}, c_0^{p2}, c_1^s, c_1^{p1}, c_1^{p2}, ..., c_{N-1}^s, c_{N-1}^{p1}, c_{N-1}^{p2})$$

Neste caso a saída c possuí taxa $\frac{1}{3}$, porém esta taxa pode ser aumentada para $\frac{1}{2}$ através do puncionamento das saídas de paridade c_k^{p1} e c_k^{p2}. Um padrão muito comum de puncionamento consiste na alternância dos bits de paridade conforme pode ser observado abaixo:

$$c = (c_0^s, c_0^{p1}, c_1^s, c_1^{p2}, c_2^s, c_2^{p1}, ..., c_{N-2}^s, c_{N-2}^{p1}, c_{N-1}^s, c_{N-1}^{p2})$$

Uma característica importante dos codificadores constituintes é que todos começam a codificação no estado zero e, após a codificação do bloco, precisam acrescentar quantos bits forem precisos para que o codificador atinja outro estado pré-definido. Normalmente este estado é o zero. Estes bits extras de terminação auxiliam na decodificação e são chamados de *tail bits* ou bits de cauda. O número de bits no bloco reservados para os *tail bits* são no máximo m bits, onde m representa a memória do codificador convolucional.

3.3 Entrelaçador

O entrelaçador (*Interleaver*) é uma das chaves do alto desempenho dos códigos turbo. A permutação dos bits entregues ao segundo codificador constituinte garante que, na média, o peso das palavras código geradas nunca seja muito baixo. Isso ocorre pelo fato de que é pouco provável que os dois codificadores apresentem palavras-código com peso baixo para a mesma seqüência de entrada, pois pelo menos um deles possui um entrelaçador na entrada. Essa melhora no peso médio das palavras-código é chamada de ganho do entrelaçador.

Outra finalidade do entrelaçador é de descorrelacionar os bits em seqüência. Isso garante que se um bit for muito prejudicado por interferências do meio de transmissão, existe outro com maior confiabilidade para auxiliar na decodificação, aumentando assim a chance de correção de erros. Essa característica é muita usada também para evitar erros provenientes de ruído em rajada, ou seja, que provoquem erros em seqüência.

O projeto do entrelaçador é muito importante na construção de um bom codificador turbo. Desde sua criação até hoje muitos pesquisadores vem propondo diversas maneiras de construir entrelaçadores como em [18], [19], [20], [21] e [22]. Dentre os tipos mais comumente usados destacam-se os entrelaçadores: de Bloco, Pseudo Aleatório e os do Tipo-S.

3.3.1 Entrelaçador de Bloco

Este entrelaçador tem a característica de ser formado por uma matriz pré-definida onde a seqüência de entrada u_k é escrita numa matriz na ordem de suas linhas e lida na ordem de suas colunas. Como exemplo temos a seqüência $u_k = (u_0, u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, u_{10}, u_{11})$, usamos uma matriz 3x4 como base e escrevemos u_k em suas linhas de forma que:

$$X = \begin{bmatrix} u_0 & u_1 & u_2 \\ u_3 & u_4 & u_5 \\ u_6 & u_7 & u_8 \\ u_9 & u_{10} & u_{11} \end{bmatrix}.$$

Baseado na matriz obtida a seqüência entrelaçada pode ser gerada concatenando os valores da matriz X em forma de colunas. Desta forma temos a seqüência entrelaçada de u_k representada por $u_k^{'} = (u_0, u_3, u_6, u_9, u_1, u_4, u_7, u_{10}, u_2, u_5, u_8, u_{11})$.

3.3.2 Entrelaçador Pseudo Aleatório

Este tipo de entrelaçador gera permutações aleatórias no bloco de informação, de forma que todos os *N* bits do bloco de informação sejam permutados de lugar com outro bit. As permutações ocorrem baseadas num gerador de números pseudo-aleatórios, daí o seu nome.

3.3.3 Entrelaçador Tipo-S

Este tipo de entrelaçador é baseado no entrelaçador pseudo-aleatório, porém com uma pequena modificação proposta em [23]. A letra *S* representa uma distância mínima (*Spread factor* em inglês) entre os índices entrelaçados, ou seja, índices próximos na entrada do entrelaçador devem estar espaçados de uma distância mínima *S* na saída do entrelaçador.

O algoritmo de construção deste entrelaçador é extremamente simples. Ele simplesmente seleciona aleatoriamente um índice como próximo membro da permutação e compara com os *S* índices anteriormente escolhidos. Caso o valor absoluto da diferença deste índice e os *S*

anteriores for menor que *S*, então o valor é temporariamente descartado e o processo é repetido até completar os *N* bits do entrelaçador.

3.4 Decodificador Turbo

A decodificação turbo é implementada através da combinação de um algoritmo de decodificação de *maximum a posteriori* (MAP), com um algoritmo de decodificação iterativa. A principal vantagem desse esquema é a simplicidade da decodificação, aliada a elevados ganhos de codificação. Caso fosse utilizado apenas o algoritmo de máxima verossimilhança seria preciso comparar 2^N seqüências possíveis com a seqüência recebida, o que computacionalmente seria inviável para tamanhos de blocos muito grandes.

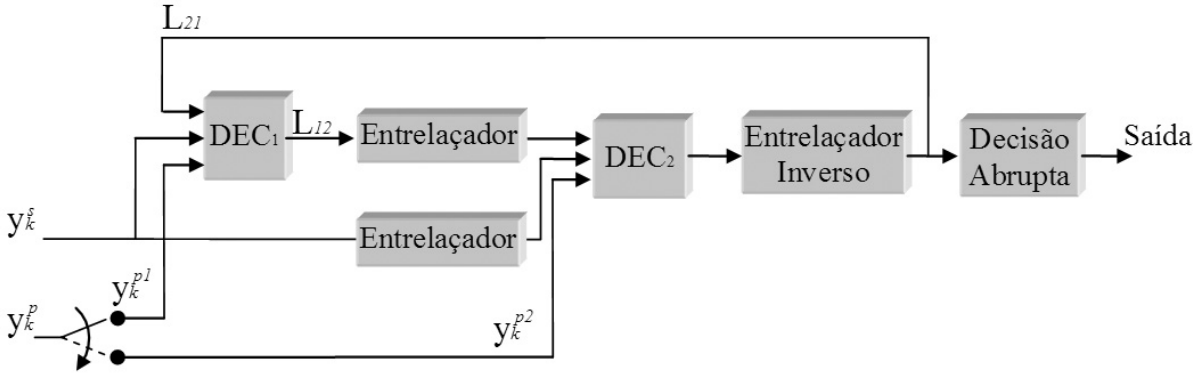

Figura 11: Decodificador Turbo.

O decodificador turbo original proposto por Berrou em [1] é mostrado na Figura 11. Nesta figura, DEC_1 representa o decodificador que recebe os dados referentes ao primeiro codificador constituinte (ENC_1) e DEC_2 representa o decodificador 2 que receberá os dados do segundo codificador (ENC_2). O bloco recebido terá *N* bits enumerados através da variável k, variando de 0 a *N*-1. Cada um dos decodificadores é um decodificador MAP que será explicado adiante. DEC_1 recebe como entrada os valores da sequência sistemática y_k^s e os valores da sequência de paridade y_k^{p1}, provenientes de ENC_1. Sua saída L_{12} é uma sequência suave de valores que representam uma prévia da estimativa dos bits transmitidos. Os dados desta saída são chamados de extrínsecos. L_{12} é então entrelaçado e mandado ao segundo decodificador. Portanto, DEC_2 terá em suas entradas a versão entrelaçada da seqüência sistemática y_k^s, a seqüência de paridades y_k^{p2} e L_{12} do primeiro decodificador entrelaçado. Com estas informações DEC_2 provê uma sequência de saída que depois de desentrelaçada torna-se a saída extrínseca L_{21}. Esta informação extrínseca L_{21} é então usada como uma nova entrada no primeiro decodificador DEC_1, formando-se assim um elo de realimentação. Todo este procedimento é repetido de uma

forma iterativa e esta característica contribui muito no desempenho dos códigos turbo. A saída do decodificador é dada na saída suave de DEC_2 aplicando-se uma regra de decisão abrupta. Normalmente esta saída é computada após algumas iterações do decodificador.

O algoritmo usado pelos decodificadores DEC_1 e DEC_2 é um algoritmo de entrada e saída suave (SISO) e de acordo com o trabalho original de *Berrou et al* em [1], é o algoritmo *Maximum A Posteriori* (MAP) modificado de *Bahl* e conhecido como algoritmo BCJR [24] devido ao nome de seus desenvolvedores Bahl, Cocke, Jelinek e Raviv. O algoritmo MAP é ótimo no sentido de minimizar a probabilidade de erro de bit e é o mais utilizado em esquemas de decodificação turbo, pois o algoritmo de Viterbi minimiza apenas a probabilidade de erro de palavras. Como os dois decodificadores DEC_1 e DEC_2 devem ser capazes de produzir decisões suaves para as etapas seguintes de decodificação, essas decisões suaves são expressas por meio do logaritmo da razão de verossimilhança ou *Log-Likelihood Ratio* (LLR) do inglês, definido da seguinte maneira para uma modulação BPSK:

$$L(u_k) = ln\frac{P(u_k = +1|y_k)}{P(u_k = -1|y_k)}\,, \tag{3.1}$$

onde u_k é o bit da mensagem no instante de tempo k e y_k é o bit da seqüência recebida no instante k. O sinal de $L(u_k)$ representa uma estimativa de u_k e sua magnitude representa a confiabilidade desta estimativa conforme sugerido em [8]. Desta forma as saídas de DEC_1 e DEC_2 são dadas em LLR.

O algoritmo MAP é melhor detalhado em [1] e [24], aqui é apresentado apenas um resumo. Em [8] e [1] é mostrado que a partir da equação (3.1), a LLR produzida na saída de cada decodicador pode ser reescrita como a soma de três termos: um proveniente do canal (L_c), outro vindo do conhecimento *a priori* das probabilidades de ocorrência dos bits (L_a) e por fim um correspondente à informação extrínseca gerada na decodificação (L_e). Uma vez calculada esta LLR, é possível obter as informações extrínsecas simplesmente subtraindo de $L(u_k)$ os termos correspondentes ao canal (L_c) e à probabilidade *a priori* (L_a). Quando usado na decodificação turbo, o algoritmo BCJR estima as probabilidades *a posteriori* dos bits de mensagem, isto é, $P(u_k = +1|y_k)$ e $P(u_k = -1|y_k)$ para a modulação BPSK, e calcula o logaritmo da razão de verossimilhança de acordo com a equação (3.1), que pode ser reescrita como:

$$L(u_k) = ln\frac{P(u_k = +1|y_k)}{P(u_k = -1|y_k)} = ln\frac{\sum_{(s',s)u_k=+1} p(s', s, y_k)}{\sum_{(s',s)u_k=-1} p(s', s, y_k)}\,, \tag{3.2}$$

onde s' e s representam os estados da treliça no instante de tempo k anterior e atual respectivamente. O somatório no numerador é realizado para todas as transições existentes entre os estados s' e s quando o bit de informação u_k é $+1$ e no denominador para quando o bit de

informação é -1. O termo y_k representa o sinal corrompido recebido do canal. Na descrição do algoritmo BCJR em [24] é mostrado que a probabilidade conjunta $p(s', s, y)$ pode ser escrita como o produto de três termos independentes, conforme a equação abaixo:

$$p(s', s, y) = p(s', y_{t<k}) \cdot p(s, y_{k\text{st}}$$

reversa (*backward recursion*) e é calculado do final da treliça até o instante de tempo $k+1$. Ja o termo γ representa a transição entre os termos α_k e β_{k+1} e depende mais especificamente da saída atual do canal.

Como proposto por Berrou em [1] o codificador inicia e termina sempre no estado pré-definido zero. Dado esta premissa α e β são iniciados com um valor padrão, antes de começar o algoritmo. Portanto, as condições iniciais de α e β são:

$$\alpha_0(s) = \begin{cases} 1, & \text{se } s = 0 \\ 0, & \text{se } s \neq 0 \end{cases}, \tag{3.10}$$

$$\beta_N(s) = \begin{cases} 1, & \text{se } s = 0 \\ 0, & \text{se } s \neq 0 \end{cases}, \tag{3.11}$$

Conforme [1] a variável γ é determinada pelas saída atual do canal, das probabilidades de transição do canal e da treliça codificadora, portanto aplicando a regra de Bayes em (3.6), temos:

$$\gamma_k(s', s) = p(s, s') \cdot p(y_k|s', s),$$

onde $p(s, s')$ representa a probabilidade *a priori* conhecida de transição de estados do codificador e $p(y_k|s', s)$ é a probabilidade de transição de um canal Gaussiano discreto e sem memória que pode ser escrito como:

$$p(y_k|s', s) = \prod_{k=0}^{N-1} \frac{1}{\sigma_k \cdot \sqrt{2\pi}} \cdot e^{-\frac{E_b \cdot r}{2 \cdot \sigma_k^2}(y_k - a \cdot c_k^p)^2},$$

onde σ_k^2 representa a variância do ruído estimado no instante de tempo k, c_k^p o símbolo gerado pelo codificador e y_k o símbolo recebido na entrada do decodificador no instante de tempo k, E_b representa a energia de bit, r a taxa de transmissão e a representa a amplitude do desvanecimento do canal (a = 1 para um canal AWGN).

Finalmente $L(u_k)$ do decodificador pode ser obtido pela seguinte equação:

$$L(u_k) = ln \frac{\sum_{(s',s)u_k=+1} \alpha_k(s') \cdot \gamma_1(y_k^p|1, s', s) \cdot \beta_{k+1}(s)}{\sum_{(s',s)u_k=-1} \alpha_k(s') \cdot \gamma_0(y_k^p|0, s', s) \cdot \beta_{k+1}(s)}, \tag{3.12}$$

onde o numerador é calculado para todas as transições de estado onde $u_k = +1$ e o denominador onde $u_k = -1$.

Em [8], Hagenauer mostra que o $L(u_k)$ do decodificador é composto de três termos: um dependente do canal, outro dependente da informação *a priori* e por fim a informação extrínseca.

Com base nesta premissa é possível escrever $L(u_k)$ como:

$$L(u_k) = L_c(y_k^s|u_k) + L_a(u_k) + L_e(u_k), \tag{3.13}$$

onde $L_c(y_k^s|u_k) = \frac{4 \cdot E_b \cdot y_k^s}{N_0}$ é chamado de valor do canal, $L_a(u_k)$ é a informação *a priori* proveniente da saída do outro decodificador constituinte e $L_e(u_k)$ é a informação extrínseca que será enviada para o outro decodificador constituinte. Note que em (3.13) a informação extrínseca $L_e(u_k)$ é obtida simplesmente subtraindo do $L(u_k)$ calculado os valores da informação de canal e *a priori*.

3.5 Codificadores Turbo Não-Sistemáticos

Na maior parte da literatura sobre códigos turbo os codificadores são sistemáticos. Recentemente foi proposto por Costello *et al* em [25] e posteriormente estudado por diversos autores, como em [26], [27] e [28], uma nova classe de códigos turbo chamados de Códigos Turbo Não-Sistemáticos (NSTC). Em [28] Massey *et al* construiram um código turbo assimétrico que superou o codificador de Berrou $\mathbb{C}(37_8,21_8)$ [1] em 0,2dB para um tamanho de bloco de 4096 bits e uma BER próxima a 10^{-5}.

Os NSTCs tem a característica de apresentar apenas saídas de paridade e conseqüentemente um distribuição assintoticamente uniforme do bloco codificado. A Figura 12 apresenta um codificador RNSC padrão. Sua estrutura é equivalente ao do codificador turbo sistemático, exceto pelo fato de que no lugar do bit sistemático, é incluso o codificador constituinte ENC_0. A entrada é representada pelo vetor u_k e como o codificador possui somente saídas de paridade, elas são representadas pelas seqüências codificadas: $c_k^{p0}, c_k^{p1}, c_k^{p2}$ e c_k^{p3}. Na Figura 12 o NSTC é representado por dois codificadores constituintes com duas saídas de paridade cada um, porém é possível acrescentar mais codificadores constituintes em paralelo para obter códigos com taxas ainda menores. A regra do puncionamento é válida também para os codificadores turbo não-sistemáticos.

O decodificador RNSC possui uma diferença em relação ao decodificador RSC na sua equação $L(u_k)$. Ela é representada apenas por dois termos, conforme mostrado em [9]:

$$L(u_k) = L_a(u_k) + L_e(u_k), \tag{3.14}$$

onde aparecem apenas os componentes de informação *a priori* (L_a) e da informação extrínseca (L_e). O termo da transição de canal (L_c) não faz mais parte da equação, pois o decodificador usa apenas entradas de paridade no processo de decodificação.

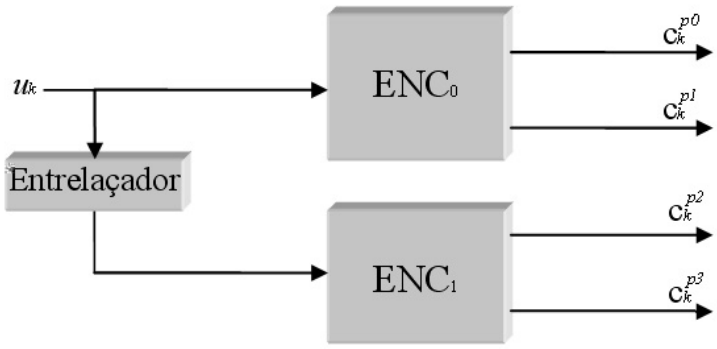

Figura 12: Codificador Recursivo Não-Sistemático(RNSC).

3.6 Algoritmo Max-Log-MAP

Para estimar os estados ou saídas de um processo, o algoritmo MAP símbolo a símbolo é dito ótimo, conforme [29]. Porém esse algoritmo, mesmo na sua forma recursiva, apresenta dificuldades de implementação, pois além de usar funções não-lineares, apresenta um alto número de adições e multiplicações. Para contornar este problema vários métodos simplificados foram propostos como o *Soft-Output Viterbi Algorithm* (SOVA) [30], o algoritmo *Log-MAP* [31] e o algoritmo *Max-Log-MAP* [32]. Estes três algoritmos sub-ótimos são comparados em [32] e é mostrado que para uma BER de 10^{-4}, existe uma diferença de ganho de 1dB entre o SOVA e o MAP e uma diferença de apenas 0,5dB entre o Max-Log-MAP e o MAP. Portanto em [32] é proposto o algoritmo Max-Log-MAP como o melhor compromisso entre complexidade e desempenho por apresentar um desempenho próximo ao MAP com uma complexidade computacional muito inferior.

Para prevenir um número alto de operações complexas e problemas numéricos, o algoritmo Max-Log-MAP irá computar e usar apenas os logaritmos das funções α, β e γ. É então aplicado o logaritmo na função γ de (3.6), representada por $\tilde{\gamma}$, que poderá então ser reescrita como:

$$\tilde{\gamma} = \ln[\gamma((y_k^s, y_k^p), s', s)] = \frac{2y_k^s x_k^s(i)}{N_0} + \frac{2y_k^p x_k^p(i, s', s)}{N_0} + \ln(p(s', s)) + K \ ,$$

onde $i = 0, 1$ e representa a fonte binária. A constante K pode ser ignorada pois ela irá se anular quando forem calculados $\ln \alpha_k(s)$ e $\ln \beta_k(s)$.

Para os cálculos subseqüentes a seguinte simplificação é considerada:

$$\ln(e^{\delta_1} + ... + e^{\delta_N}) \approx max(\delta_k), k \in \{1..N\} \ ,$$

Definimos então $\tilde{\alpha}_k(s) = \ln \alpha_k(s)$ e $\tilde{\beta}_k(s) = \ln \beta_k(s)$. Usando algumas simplificações

de [32] é possível escrever:

$$\tilde{\alpha}_k(s_k) \approx max_{(s_{k-1},i)}[\tilde{\gamma}((y_k^s, y_k^p), s_{k-1}, s_k) + \tilde{\alpha}_{k-1}(s_{k-1})] \\ -max_{(s_{k-1},s_k,i)}[\tilde{\gamma}((y_k^s, y_k^p), s_{k-1}, s_k) + \tilde{\alpha}_{k-1}(s_{k-1})]\,, \tag{3.15}$$

e

$$\tilde{\beta}_k(s_k) \approx max_{(s_{k+1},i)}[\tilde{\gamma}((y_{k+1}^s, y_{k+1}^p), s_k, s_{k+1}) + \tilde{\beta}_{k+1}(s_{k+1})] \\ -max_{(s_k,s_{k+1},i)}[\tilde{\gamma}((y_{k+1}^s, y_{k+1}^p), s_k, s_{k+1}) + \tilde{\alpha}_k(s_k)]\,, \tag{3.16}$$

Portanto a equação aproximada do LLR de cada bit é dada por:

$$L(u_k) \approx max_{(s_k,s_{k-1})}[\tilde{\gamma}_1(((y_k^s, y_k^p), s_{k-1}, s_k) + \tilde{\alpha}_{k-1}(s_{k-1}) + \tilde{\beta}_k(s_k)] \\ -max_{(s_k,s_{k-1})}[\tilde{\gamma}_0(((y_k^s, y_k^p), s_{k-1}, s_k) + \tilde{\alpha}_{k-1}(s_{k-1}) + \tilde{\beta}_k(s_k)]\,, \tag{3.17}$$

Como conseqüência das aproximações feitas em [33], o Max-Log-MAP considera a cada instante apenas dois caminhos da treliça do canal: o melhor para um bit transmitido x_k = +1 e o melhor para x_k = -1, sendo que um deles é o caminho de máxima verossimilhança. Desta maneira, as decisões abruptas produzidas pelo Max-Log-MAP são idênticas às produzidas pelo algoritmo de Viterbi [8]. Já o MAP e o Log-MAP levam em consideração todos os caminhos da treliça para calcular as LLR dos bits transmitidos e suas decisões podem diferir daquelas produzidas pelo algoritmo de Viterbi.

Portanto, usando a equação (3.17) é possível reduzir a complexidade do cálculo do LLR do decodificador MAP, pois somente operações de soma e logaritmos serão necessárias. Isso contribui muito na construção de decodificadores e diminui consideravelmente o tempo de simulação, com um custo pequeno de degradação de performance do código.

3.7 Desempenho do Código Turbo

O que mais chamou a atenção quando os códigos turbo foram apresentados pela primeira vez foi sua performance. Devido à forma como ele é construído, foi possível alcançar um desempenho nunca antes visto, cerca de 0,7dB do limite de Shannon para uma BER de 10^{-5}. Desde então houve muitos estudos para tentar compreender todas as características dos códigos turbo, tais como em [25], [27], [3], [29], [34] e [35].

Em princípio, Berrou mostrou que o número de iterações tem grande influência no desempenho do código turbo. Porém existem inúmeros fatores que contribuem com a performance

dos códigos turbo. A Figura 13 mostra a influência do número de iterações na performance do código turbo. Nesta simulação o código turbo usado foi o código sistemático $\mathbb{C}(37_8,21_8)$, com um tamanho de bloco de 10000 bits, taxa $r = \frac{1}{2}$, entrelaçador pseudo-aleatório e com um canal submetido a um ruído AWGN. Observando este gráfico nota-se que existe um ganho altíssimo de desempenho quando aumentamos de uma para seis iterações, porém de seis para dezoito iterações o ganho de performance é muito menor. Isso deve-se ao fato de que a cada nova iteração o decodificador tende a corrigir os bits recebidos com uma maior confiabilidade, porém existe um limite onde o decodificador satura e não consegue mais corrigir os bits mesmo aumentando o número de iterações.

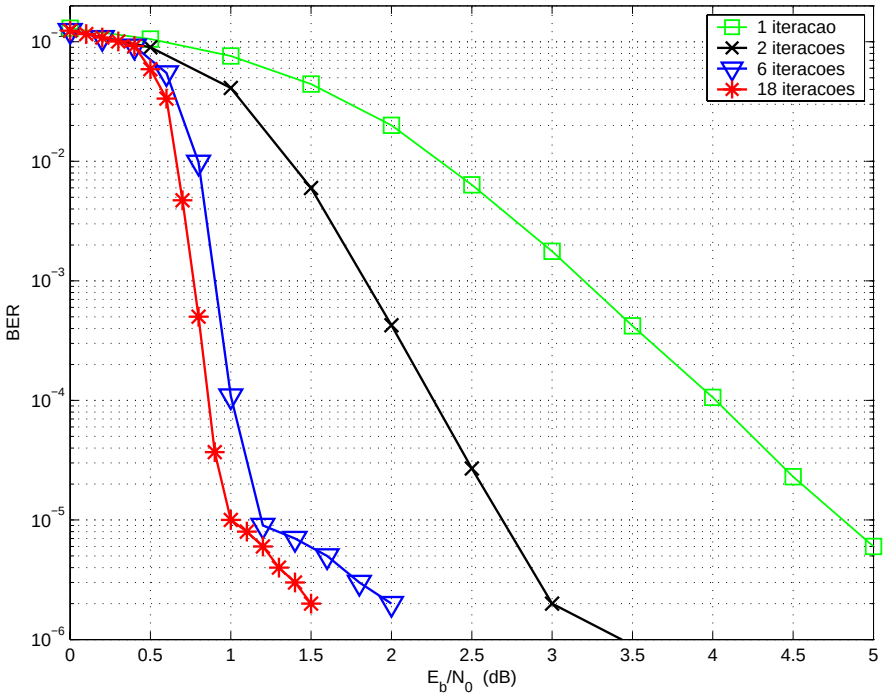

Figura 13: Número de Iterações x BER x SNR(dB).

Outra característica que contribui com o desempenho do código turbo é o tamanho do bloco de dados processado. Quanto maior o tamanho deste bloco mais próximo ao limite de Shannon, que neste caso é 0dB, o código estará. A Figura 14 ilustra a performance do mesmo código do parágrafo anterior submetido a blocos de dados com tamanhos de 100, 1000, 10000 e 100000 bits. A figura deixa claro que o tamanho do bloco também tem grande influência no desempenho e quanto maior o tamanho do bloco mais ele se aproxima do limite de Shannon.

Observando a curva de desempenho dos códigos turbo é possível traçar três regiões distintas: a região de alta BER, a região de queda (*waterfall*) e a região do patamar de erro. A região de alta BER compreende baixos valores de $\frac{E_b}{N_0}$ e é caracterizada por um valor de BER$\approx$0,1. A região de alta BER aparece na figura 13 na curva de 18 iterações, até aproximadamente 0,5dB.

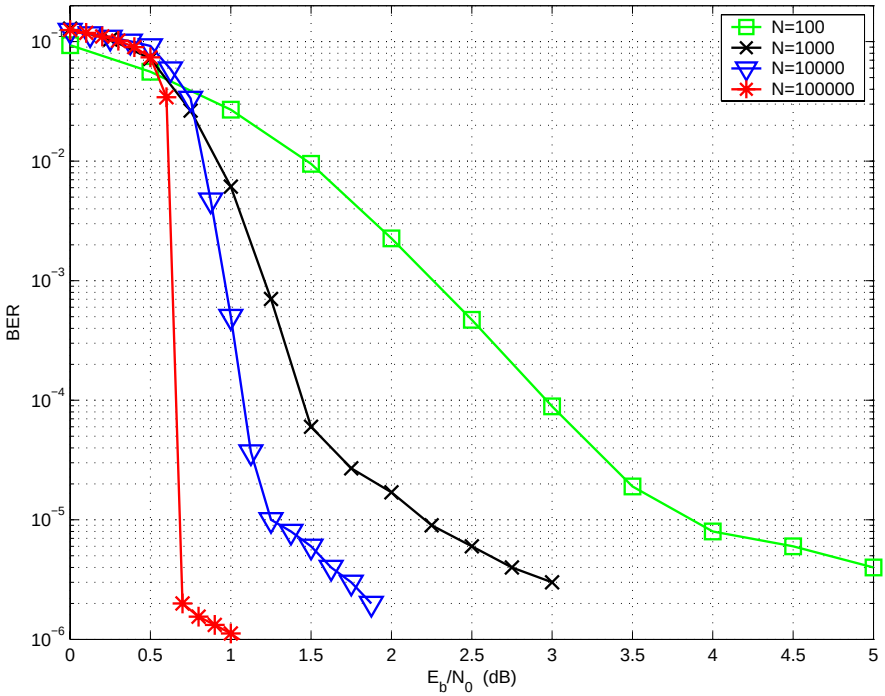

Figura 14: Tamanho do Bloco(N) x BER x E_b/N_0(dB).

Nesta região, mesmo aumentado o número de iterações a BER não diminui. A região de queda pode ser bem observada na Figura 14, na curva de *N*=100000 bits. Nesta curva existe um momento perto de 0,5dB onde o gráfico cai vertiginosamente, assemelhando-se a uma queda d'água, daí o nome em inglês: *waterfall*. Ela é vista como uma região de transição entre a região de alto BER e a região do patamar de erro. Também nesta curva, seguindo o eixo $\frac{E_b}{N_0}$, logo após a região de queda existe uma mudança na tendência de desempenho da curva próximo de 0,7dB onde a queda começa a se tornar mais amena. Este efeito ocorre deste ponto para frente e é chamado de região do patamar de erro. Ela tem a característica de ter uma baixa BER, porém mesmo aumentando o $\frac{E_b}{N_0}$ nesta região, a melhora de performance é muito inferior à melhora na região de queda. Em [36] é mostrado que o fenômeno do patamar de erro ocorre devido à baixa distância livre (d_{free}) dos codificadores convolucionais do código turbo.

Os codificadores convolucionais constituintes do código turbo também podem influenciar no desempenho do sistema. Dependendo da distribuição da fonte, pode existir um código que apresente um desempenho melhor para esta distribuição. Em [3] é mostrado como usar códigos turbo sistemáticos para aumentar a performance na transmissão de fontes não-uniformes e tirar proveito da informação a-priori da distribuição da fonte no decodificador. Já em [9] é mostrado como encontrar bons códigos não-sistemáticos para fontes não-uniformes, além de apresentar simulações onde os códigos turbo não-sistemáticos apresentam uma performance superior aos sistemáticos para alguns casos de fontes não-uniformes.

3.8 Comentários Finais

Este capítulo apresentou o funcionamento dos códigos turbo tanto no transmissor como no receptor, além de mostrar a decodificação através do algoritmo BCJR e o algoritmo aproximado Max-Log-MAP. Por fim, apresentou algumas de suas características principais através do gráfico de desempenho para diferentes configurações do código.

O capítulo seguinte descreverá três configurações de códigos turbo adaptados para transmissão de fontes não-uniformes, para que através de simulações possa haver uma comparação justa entre eles.

4 *Codificação Turbo para Fontes Não-Uniformes.*

4.1 Introdução

Este capítulo tem como objetivo apresentar os três métodos de codificação para fontes não-uniformes estudados: o método da codificação separada de fonte e canal com código turbo sistemático, o método da decodificação controlada pela fonte usando codificador turbo não-sistemático e o método da decodificação controlada pela fonte usando codificador turbo não-sistemático e alocação desigual de energia.

4.2 Método 1 - Codificação Separada de Fonte e Canal

O primeiro método é baseado no modelo clássico de um sistema de transmissão. Sua característica principal é a codificação independente da fonte e do canal e foi estudado, em conjunto de códigos turbo, por Alajaji em [9]. A Figura 15 ilustra o diagrama em blocos deste sistema. Dada uma fonte não-uniforme $u_k = \{u_0, u_1, ..., u_{N-1}\}$ com distribuição p_0 e $p_1 = 1 - p_0$ e comprimento N, submetida a um codificador Huffman com taxa de compressão r_s, é gerada a seqüência de saída v_k de comprimento médio $N \cdot r_s$. Esta seqüência é então aplicada a um codificador turbo sistemático de taxa r_c, gerando a sequência c_k de taxa $\frac{r_s}{r_c}$ e comprimento médio:

$$N_c = \frac{N \cdot r_s}{r_c} \tag{4.1}$$

O bloco c_k é então mapeado em BPSK (+1 e -1) e submetido ao canal. Na chegada do receptor é então aplicado um decodificador turbo sistemático que estima o valor de v_k e o aplica no decodificador Huffman, que através da tabela de compressão poderá decodificar o bloco u_k original.

Em um caso onde é usado diretamente um codificador sistemático na fonte sem compressão, parte da distribuição da fonte aparece no bloco codificado através dos bits sistemáticos do

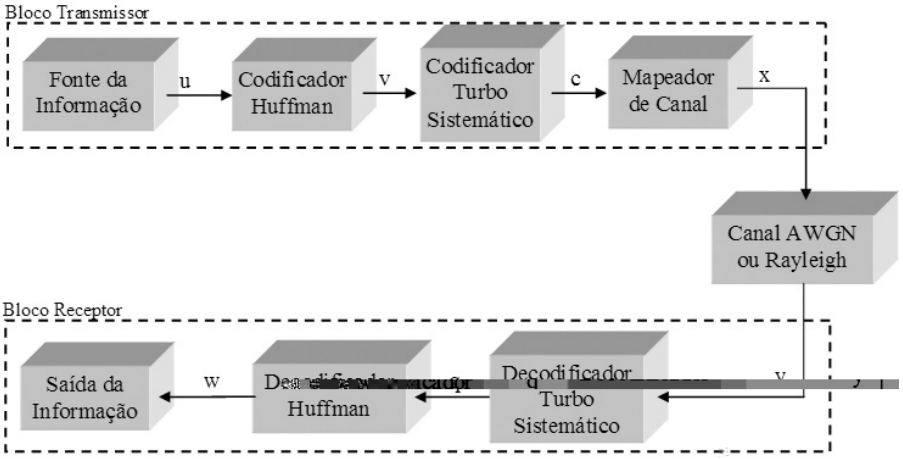

Figura 15: Método 1 - Codificação Separada de Fonte e Canal.

codificador, fazendo com que a distribuição do bloco seja não-uniforme. A grande vantagem de codificar previamente a fonte com o código Huffman, é a possibilidade de utilizar um código turbo com taxa menor e conseqüentemente mais poderoso. Além disso o codificador de fonte garante um bloco assintoticamente uniforme, ou seja, com uma distribuição $p_0 \approx 0.5$ após codificado, podendo assim aproveitar melhor a capacidade do canal como visto na seção 2.4, pois tanto os bits sistemáticos quanto os de paridade apresentaram uma distribuição assintoticamente uniforme. Uma desvantagem deste método é a possibilidade de ocorrer uma avalanche de erros no decodificador Huffman devido ao fato de que se um bit for decodificado com a palavra código errada, o resultado dos bits subseqüentes podem ser alterados.

Uma característica deste método é que o bloco codificado não possui tamanho fixo, isto ocorre devido ao fato de que o codificador Huffman é um codificador de palavras de tamanho variável. Outra característica é que a taxa de codificação é dependente da distribuição da fonte, ou seja, para atingir uma determinada taxa de canal é necessário uma certa distribuição da fonte que aliada à taxa do codificador Huffman e à taxa do código turbo, constituirá a taxa de canal. Neste método é necessário também que tanto o codificador quanto o decodificador conheçam a tabela Huffman de compressão. Simulações em [9] demonstraram uma boa performance deste método sobre fontes não-uniformes, daí o interesse em estudá-lo.

4.3 Método 2 - Decodificação Controlada pela Fonte usando Códigos Turbo Não-Sistemáticos

Este método, também conhecido como decodificação controlada pela fonte, ou do inglês, *Source-Controlled Channel Decoding* (SCCD) como mostrado em [8], consiste em usar um có-

digo turbo não-sistemático para a codificação conjunta fonte-canal e aproveitar a informação *a priori* da distribuição da fonte no receptor. A vantagem de usar um codificador não-sistemático é que, ao contrário do sistemático, é possível obter uma saída codificada praticamente uniforme mesmo para fontes altamente não-uniformes, conforme visto em [9]. Esta uniformidade gerada garante que a capacidade do canal seja melhor explorada, tanto para canais com ruído AWGN quanto para canais com desvanecimento Rayleigh, conforme mostrado em [6].

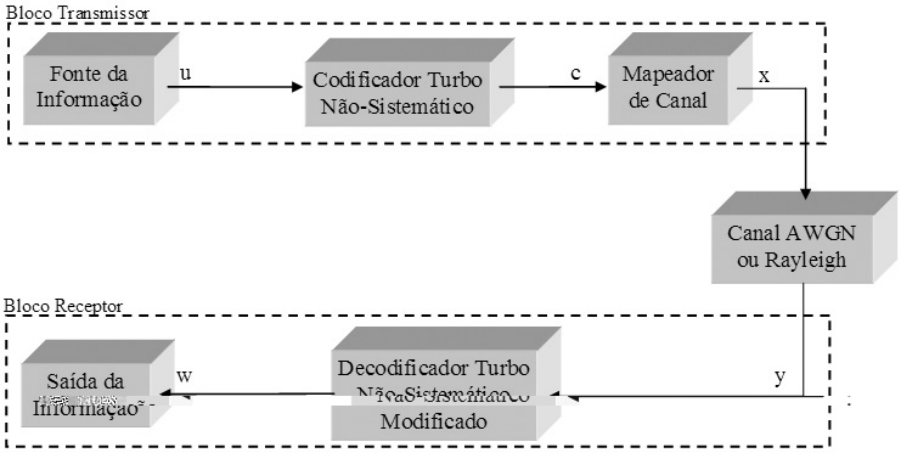

Figura 16: Método 2 - Decodificação Controlada pela Fonte usando Código Turbo Não-Sistemático.

O diagrama em blocos deste método é mostrado na Figura 16. A seqüência não-uniforme da fonte $u_k = \{u_0, u_1, ..., u_{N-1}\}$ com distribuição p_0 e $p_1 = 1 - p_0$ e comprimento N, é codificada diretamente a uma taxa r_c através do codificador turbo não-sistemático. O decodificador, uma vez que recebe o sinal corrompido pelo canal precisa estimar a seqüência u_k. Para isso é aplicada uma pequena modificação na equação do decodificador turbo como proposto por Alajaji em [9]. A idéia é considerar a distribuição p_0 *a priori* da fonte, para obter uma performance otimizada no processo de decodificação. Esta modificação é relativamente simples e consiste em alterar a equação do logaritmo da máxima verossimilhança $L(u_k)$ do decodificador turbo, a fim de acrescentar a informação *a priori* da fonte nas suas equações. A modificação utiliza a informação *a priori* da fonte na forma $\log(\frac{1-p_0}{p_0})$, no termo $L_a(u_k)$ durante a primeira iteração e nas iterações seguintes utilizar $L_e(u_k) + \log(\frac{1-p_0}{p_0})$ como a nova informação extrínseca que será trocada entre os codificadores constituintes. Esta pequena alteração no decodificador garante um desempenho superior do código, como comprovado por Alajaji em [9], e proposto no seu conceito em [8].

Ao contrário do primeiro método, este trabalha com um comprimento de bloco fixo de tamanho $\frac{N}{r_c}$ e, um pré-requisito para seu funcionamento adequado é o conhecimento *a priori* da distribuição da fonte pelo decodificador. Em [9], Alajaji mostra como procurar por um

código não-sistemático que apresente um melhor desempenho para uma dada distribuição da fonte. Para fazer esta pesquisa ele utiliza algumas propriedades assintóticas dos codificadores recursivos que reduzem o número de possíveis candidatos.

Simulações em [9] demonstraram também uma boa performance deste método sobre fontes não-uniformes.

4.4 Método 3 - Decodificação Controlada pela Fonte usando Códigos Turbo com propriedade QLI e Alocação Desigual de Energia

Este método foi primeiramente proposto em [4] e, é composto por um codificador turbo não-sistemático com uma propriedade especial chamada *quick-look-in* (QLI). Baseado na informação *a priori* da fonte, usa o método da alocação desigual de energia proposta em [4], para ponderar os símbolos a serem transmitidos pelo canal. A alocação desigual de energia tem um efeito equivalente ao da compressão da fonte e todo o processo apresenta duas etapas de codificação e apenas uma de decodificação, como pode ser visto na Figura 17.

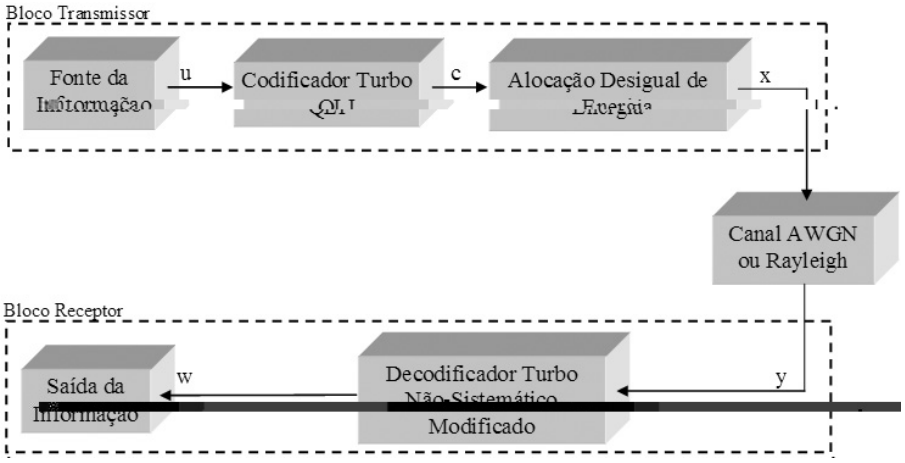

Figura 17: Método 3 - Decodificação Controlada pela Fonte usando Código Turbo com propriedade QLI e Alocação Desigual de Energia..

Como mostrado por Costello em [28], a característica básica do codificador não-sistemático com a propriedade QLI é que a soma das paridades, gera uma versão do bit sistemático.

A estrutura do codificador QLI é mostrada na Figura 18. Ele é composto por dois codificadores constituíntes ENC_0 e ENC_1 e é formado pelos polinômios, em base octal, na forma $\mathbf{C}(g^0, g^1, g^2, g^3, g^4)$. Sendo que ENC_0 possui duas saídas de paridade c_k^{p0} e c_k^{p1}, e é formado pelos polinômios numeradores g^1 e g^2 e o polinômio denominador comum g^0. O segundo codificador

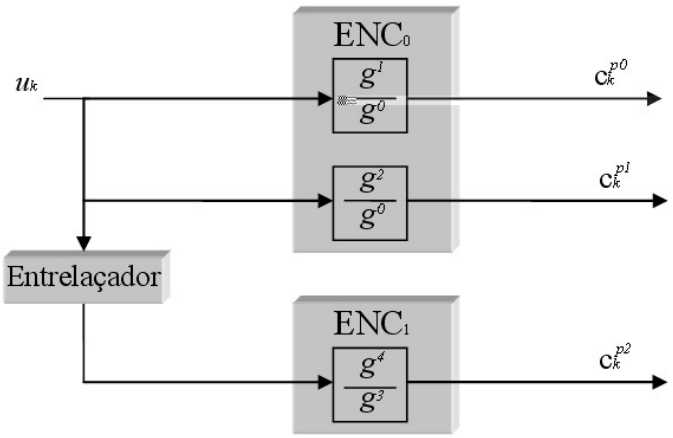

Figura 18: Diagrama em Blocos do Codificador QLI

constituinte possui apenas a saída c_k^{p2} e é formado pelo polinômio numerador g^4 e denominador g^3. O que define a característica QLI é uma dependência entre os polinômios geradores, tal que:

$$g^0 = g^1 + g^2 \,, \tag{4.2}$$

onde a adição é realizada módulo 2. Esta característica dos polinômios contribui para que os bits da fonte (u_k) possam ser estimados no receptor apenas somando as paridades c_k^{p0} e c_k^{p1} de *ENC*$_0$. Esta propriedade é facilmente comprovada somando os polinômios que geram as paridades c_k^{p0} e c_k^{p1} e substituindo a Equação (4.2), tal que:

$$u \cdot (\frac{g^1}{g^0} + \frac{g^2}{g^0}) = u \cdot (\frac{g^1}{g^1 + g^2} + \frac{g^2}{g^1 + g^2}) \equiv u \tag{4.3}$$

A estimativa dos bits da fonte permite ao decodificador uma previsão da distribuição da fonte, que é usada no processo de decodificação para melhorar a performance. Caso seja necessário aumentar a taxa de transmissão, é possível usar o puncionamento das saídas.

A alocação desigual de energia é aplicada no mapeamento do sinal que será transmitido e baseia-se na alocação de mais energia para transmitir os bits menos freqüentes e menos energia para transmitir os bits mais freqüentes. Esta alocação de energia tem mais sentido quando usado diretamente nos bits da fonte que apresentam uma distribuição não-uniforme, pois nesse caso é possível ponderar a energia de cada bit, como explicado em [5]. Porém, em [4] é mostrado que os bits de paridade podem também ser submetidos a alocação desigual de energia, desde que um parâmetro empírico de performance seja obtido. Na saída do codificador QLI, a alocação de energia dos bits não-sistemáticos é dada assumindo que a energia média por símbolo é igual a 1, portanto a alocação de energia para os bits 0 e 1 é representada conforme demonstrado em [4]

por:

$$E_{bit}^{c} = \begin{cases} \rho \cdot \frac{p_0}{p_1}, & \text{se bit = 1} \\ \frac{1-(\rho \cdot p_0)}{p_0}, & \text{se bit = 0} \end{cases}, \tag{4.4}$$

onde ρ é escolhido de modo a apresentar o melhor desempenho do código. O artigo [4] mostra como obter o valor ideal de ρ para cada distribuição de fonte. O mapeamento da energia depende dos bits sistemáticos e dos bits de paridades correspondentes. O bit sistemático define a amplitude do bit de saída utilizando a Equação (4.4) e o bit de paridade define o sinal deste bit. Por exemplo, dado uma fonte não-uniforme com distribuição p_0=0,1, representada por u_k ={ 0, 0, 1, 1}, através de um codificador convolucional gera uma seqüência de paridade representada por c_k^p={1, 0, 1, 0}. Usando a equação (4.4) com ρ=0,7, obtemos os valores de amplitude E_0^c=0,078 e E_1^c=9,30. Como queremos que a energia média por símbolo seja unitária, o canal recebe a raiz quadrada destes valores. Com base nestes valores de alocação de energia, E_0^c é mapeado para todo bit 0 em u_k e E_1^c para todo bit 1 em u_k. O bit de paridade define o valor do sinal de E_0^c e E_1^c, de tal forma que se c_k^p é 0, então o bit é representada por um sinal negativo e caso c_k^p seja igual a 1, o bit terá sinal positivo. Com base nestas premissas a alocação desigual de energia para o exemplo é dada por x_k = {+0,28, -0,28, +3,05, -3,05}.

A vantagem de usar este método é que a alocação desigual de energia simula o efeito da compressão da fonte, enquanto o codificador não-sistemático gera uma sequência codificada assintoticamente uniforme a fim de explorar melhor a capacidade do canal. Este método possui tamanho de bloco fixo e o decodificador não necessita ter conhecimento prévio da distribuição da fonte.

Simulações em [4] demonstraram uma boa performance para este método sobre fontes não-uniformes. O desempenho superou inclusive o mesmo código utilizando alocação uniforme de energia.

5 *Simulações.*

5.1 Introdução

Este capítulo mostra as simulações realizadas utilizando os três métodos de codificação apresentados. Primeiramente são mostradas as configurações e parâmetros utilizados em cada método. Em seguida são mostrados os resultados das simulações e por fim alguns comentários e conclusões.

5.2 Preparação das Simulações

Os três métodos estudados serão comparados através de simulações computacionais. O objetivo das simulações é traçar o gráfico de desempenho de cada método através da taxa de erro de bit versus a razão sinal-ruído (SNR) do canal. Os modelos de canais utilizados nas simulações serão de dois tipos: canal sobre influência do ruído branco aditivo Gaussiano (AWGN) e canal sobre influência do desvanecimento Rayleigh rápido. É considerado que nenhum decodificador simulado possui conhecimento prévio do canal. O método de simulação utilizado será o de Monte Carlo, que consiste basicamente na geração aleatória e repetitiva de entradas para um sistema e na captura e análise das suas saídas. Este método é usado quando é impossível ou impraticável de se calcular uma resposta determinística exata para determinado algoritmo. O método de Monte Carlo é repetido até que se obtenha um resultado com um valor mínimo de confiança.

Como o desempenho do código turbo depende do tamanho do bloco da fonte, serão testados 3 tamanhos distintos de blocos: 1,2k bits, 12k bits e 120k bits. Os blocos de 1,2k bits representam sistemas com tamanho de bloco pequeno, como o caso de alguns protocolos de celular, que podem variar em torno de 1000 bits por pacote. Já o tamanho de bloco de 12k bits representa protocolos de tamanho intermediários, como os protocolos de Ethernet da família IEEE 802.3. Os blocos de 120k bits representam blocos grandes e apresentaram uma performance muito mais elevada se comparados com tamanhos menores de blocos.

O número de iterações de decodificação dos códigos turbo influenciam muito em seu desempenho como já visto em [1]. Neste trabalho, para assegurar que os decodificadores turbo apresentem seus desempenhos máximos, serão utilizadas 20 iterações de decodificação em todos os métodos.

Todas as simulações serão realizadas utilizando o entrelaçador do tipo-S, conforme explicado na seção 3.3.3, com S=10 para garantir uma boa distribuição e distância entre os bits.

Como o método de Monte Carlo precisa de um grande número de rodadas para obter uma boa estimativa da resposta, o critério de parada para as simulações foi feita conforme a Tabela 7. Este critério garante que serão simulados aproximadamente 10^8 bits para cada tamanho de bloco.

Tabela 7: Critério de parada das simulações.

Tamanho do bloco	**Número de blocos simulados**
1,2k bits	84.000
12k bits	8.400
120k bits	840

Serão estudadas duas distribuições de fonte distintas nomeadas de fonte A e a fonte B, com distribuições p_0^A=0,83079 e p_0^B=0,899063, respectivamente. Estas distribuições foram escolhidas devido a uma dependência observada no método 1, como será visto a seguir na preparação deste método.

Para que a comparação entre os três métodos seja justa, o ambiente de simulação, bem como algumas características deles precisam ser pré-definidas anteriormente. Estas configurações de cada método serão explicadas a seguir, porém a idéia básica é fazer com que cada método codifique as duas fontes de modo que entregue o bloco codificado ao canal com uma taxa $r = \frac{1}{2}$ e energia média de bit igual a 1. Os gráficos serão traçados até a taxa de erro de bit de 10^{-6}.

As simulações foram realizadas num microcomputador padrão PC x86, rodando o sistema operacional Windows XP SP2. A configuração do sistema é a seguinte:

- Placa mãe MSI K8N Neo Platinum Edition;
- Processador Athlon 64 3,2GHz com 128kB L1 e 1MB L2;
- 1GB DDR400;
- HD Serial Ata Seagate 320GB;

Tabela 8: Tabela Huffman de Conversão.

Binário(Input)	**Codificação Huffman**
0000	0
0001	1110
0010	1111
0011	110011
0100	100
0101	110100
0110	110101
0111	110010111
1000	101
1001	110110
1010	110111
1011	11001000
1100	11000
1101	11001001
1110	11001010
1111	110010110

O compilador utilizado é o GCC versão 3.4.2-20040916-1 e as compilações não possuem nenhum parâmetro de otimização do código.

5.2.1 Preparação do Método 1

A distribuição das duas fontes estudadas, p_0^A=0,83079 e p_0^B=0,899063, tem origem de uma característica intrínseca deste primeiro método e está ligada à taxa de compressão do código Huffman. A Tabela 8 mostra o código Huffman utilizado, este código é de ordem 4, ou seja, cada 4 bits da fonte representa um símbolo para o codificador. Esta tabela apresenta um tamanho médio de símbolo para a fonte A de 2,667 bits/símbolo e para a fonte B de 2 bits/símbolo. Esta tabela de codificação é fixa e conhecida tanto no transmissor quanto no receptor. A codificação de fonte garante uma taxa média de compressão de r_s^A=$\frac{2}{3}$ para a fonte A e uma taxa média de compressão de r_s^B=$\frac{1}{2}$ para a fonte B. A fonte A, após o codificador Huffman, possibilita o uso de um codificador turbo com taxa r_c=$\frac{1}{3}$, mantendo a saída com uma taxa média de canal de $\frac{r_c}{r_s}$ = $\frac{1}{2}$. Já para a fonte p_0^B, após o codificador Huffman, podemos usar um codificador turbo com taxa r_c=$\frac{1}{4}$, mantendo a saída com a mesma taxa média de canal de $\frac{r_c}{r_s}$ = $\frac{1}{2}$.

Para a fonte A foi utilizado o código sistemático $\mathsf{C}(35_8,23_8)$ da Figura 19, pois este código apresenta um bom desempenho na taxa $\frac{1}{3}$ para fontes uniformes, como estudado em [9]. Para a fonte B, foi usado o código $\mathsf{C}(23_8, 33_8, 37_8, 25_8)$, de taxa $1/4$, mostrado na Figura 20, por apresentar também um bom desempenho na taxa de $1/4$ para fontes uniformes, conforme estudado

previamente em [37]. Um resumo dos parâmetros utilizados no método 1 estão na Tabela 9.

Figura 19: Codificador turbo sistemático $\mathsf{C}(35_8,23_8)$ com $r_c = \frac{1}{3}$ para a fonte com p_0^A=0,83079.

Figura 20: Codificador turbo sistemático $\mathsf{C}(23_8,\ 33_8,\ 37_8,\ 25_8)$ com $r_c = \frac{1}{4}$ para a fonte com p_0^B=0,899063.

Tabela 9: Configuração do Método 1.

	Fonte A	**Fonte B**
Codificador Huffman	$r_s = \frac{3}{2}$	$r_s = \frac{2}{1}$
Código de Canal	$\mathsf{C}(35_8,23_8)$, $r_c = \frac{1}{3}$	$\mathsf{C}(23_8, 33_8, 37_8, 25_8)$, $r_c = \frac{1}{4}$
Taxa no Canal	$r = \frac{1}{2}$	$r = \frac{1}{2}$
Mapeador de Canal	+1 e -1	+1 e -1

Tabela 10: Configuração do Método 2.

	Fonte A	**Fonte B**
Codificador Huffman	$r_s = 1$	$r_s = 1$
Código de Canal	$\mathsf{C}(35_8, 23_8, 25_8)$, $r_c = \frac{1}{2}$	$\mathsf{C}(31_8, 23_8, 27_8)$, $r_c = \frac{1}{2}$
Taxa no Canal	$r = \frac{1}{2}$	$r = \frac{1}{2}$
Mapeador de Canal	+1 e -1	+1 e -1

5.2.2 Preparação do Método 2

A escolha dos codificadores constituintes é fundamental neste método, pois existe um codificador não-sistemático ideal para cada distribuição da fonte. Em [9] é mostrado um método de encontrar um bom codificador para cada distribuição de fonte. Para a fonte A com distribuição de p_0^A=0,83079, foi escolhido o codificador não-sistemático com equação $\mathsf{C}(35_8, 23_8, 25_8)$ e taxa r_c=$\frac{1}{2}$ como pode ser visto na Figura 21. Para a fonte B com distribuição p_0^B=0,899063, foi usado o codificador $\mathsf{C}(31_8, 23_8, 27_8)$ também de taxa r_c=$\frac{1}{2}$ da Figura 22. Ambos os códigos foram mostrados em [9] e apresentam também uma boa performance para cada distribuição de fonte. Um resumo dos parâmetros utilizados no método 2 estão na Tabela 10.

5.2.3 Preparação do Método 3

Este método em especial usará o mesmo codificador não-sistemático com a propriedade QLI para as duas distribuições de fonte. Em [38] é estudado o desempenho deste codificador $\mathsf{C}(73_8, 46_8, 35_8, 03_8, 15_8)$, que apresentou uma boa performance na região do patamar de erro. Uma diferença deste codificador, se comparado com os codificadores dos outros dois métodos, é que ele possui uma memória igual a 5 para um codificador constituinte e uma memória igual a 3 para o segundo codificador constituinte, enquanto os outros dois métodos possuem memória igual a 4 para ambos os codificadores constituintes. Portanto, este decodificador possui um tempo de decodificação superior aos outros dois métodos devido ao tamanho de sua treliça ser maior. O diagrama deste codificador pode ser visto na Figura 23. As saídas c_{k1} e c_{k2} são puncionadas alternadamente, a fim de obtermos uma taxa de $r_c = \frac{1}{2}$.

Figura 21: Codificador turbo não-sistemático $\mathsf{C}(35_8, 23_8, 25_8)$ com $r_c = \frac{1}{2}$ para a fonte p_0^A=0,83079.

Figura 22: Codificador turbo não-sistemático $\mathsf{C}(31_8, 23_8, 27_8)$ com $r_c = \frac{1}{2}$ para a fonte p_0^B=0,899063.

Para realizar a alocação desigual de energia para a fonte p_0^A=0,83079, usamos a equação (4.4) e substituímos a variável ρ pela constante 0,65, pois em [4] é mostrado que este valor maximiza o desempenho do código para fontes com distribuição próxima de 0,8. Já para fonte p_0^B=0,899063, o valor de ρ que maximiza o seu resultado, e será usado, é de aproximadamente 0,70. Com isso é possível mapear os níveis de alocação de energia para os dois modelos de fonte. Para a fonte A com distribuição p_0^A=0,83079 e p_1^A=0,16921, utilizando a equação (4.4) com ρ=0,65, obtemos os valores de alocação de energia conforme a Tabela 12. Já para a fonte B com distribuição p_0^B=0,899063 e com ρ=0,70, obtemos os valores de alocação de energia

Figura 23: Codificador turbo não-sistemático com propriedade QLI, $\mathsf{C}(73_8, 46_8, 35_8, 03_8, 15_8)$ com $r_c = \frac{1}{2}$ para as fontes A e B.

Tabela 11: Configuração do Método 3.

	Fonte A	**Fonte B**
Codificador Huffman	$r_s = 1$	$r_s = 1$
Código de Canal	$\mathsf{C}(73_8, 46_8, 35_8, 03_8, 15_8)$, $r_c = \frac{1}{2}$	$\mathsf{C}(73_8, 46_8, 35_8, 03_8, 15_8)$, $r_c = \frac{1}{2}$
Taxa no Canal	$r = \frac{1}{2}$	$r = \frac{1}{2}$
Mapeador de Canal	-0,744, +0,744, -1,786 e +1,786	-0,642, +0,642, -2,497 e +2,497

para os bits codificados como mostrado na Tabela 13. Um resumo dos parâmetros utilizados no método 3 estão na Tabela 11.

5.3 Resultado das Simulações

O gráficos das simulações foram separados em duas seções. A primeira seção compreende os gráficos das simulações para a fonte A sobre os dois modelos de canais e para os três tamanhos de blocos propostos, e são mostrados nas Figuras 24 à 29. A segunda seção apresenta os mesmos gráficos, porém para a fonte B e podem ser visto nas Figuras 30 à 35.

A primeira seção possui seis gráficos distintos para a fonte A. As Figuras 24 e 25 são as

Tabela 12: Tabela de alocação desigual de energia para os bits codificados da fonte p_0^A=0,83079, com ρ=0,65.

Bit de Entrada (Fonte)	Bit Codificado	Alocação Desigual de Energia
0	0	-0,744
0	1	+0,744
1	0	-1,786
1	1	+1,786

Tabela 13: Tabela de alocação desigual de energia para os bits codificados da fonte p_0^B=0,899063, com ρ=0,70.

Bit de Entrada (Fonte)	Bit Codificado	Alocação Desigual de Energia
0	0	-0,642
0	1	+0,642
1	0	-2,497
1	1	+2,497

simulações com tamanho de bloco N=1,2k bits para os dois modelos de canais. As Figuras 26 e 27 correspondem às simulações com tamanho de bloco N=12k bits e as Figuras 28 e 29 com bloco de tamanho N=120k bits.

A segunda seção também possui seis gráficos distintos, porém destina-se à apresentação das simulações da fonte B. As Figuras 30 e 31 representam as simulações com tamanho de bloco N=1,2k bits. Já as Figuras 32 e 33 são simulações com tamanho de bloco N=12k bits e por fim as Figuras 34 e 35 com bloco de N=120k bits.

5.3.1 Análise dos Resultados

Os três métodos analisados possuem características distintas de construção, porém apresentaram um desempenho bem similar no decorrer das simulações com pequenas variações.

Como já eram esperadas, as simulações sobre o canal influenciado pelo ruído AWGN tiveram uma performance superior, se comparados com as simulações sobre o canal com desvanecimento Rayleigh.

Nas simulações da fonte A, os métodos 1 e 3 apresentaram um desempenho ligeiramente superior ao método 2, principalmente na região da queda. Contudo o método 1 apresentou um patamar de erro muito elevado, na faixa de 10^{-5}, para os blocos de 1,2k e 12k bits. Este é um resultado razoável, pois o método 1 possui tendência de apresentar um resíduo de erro alto devido ao seu modelo de construção baseado na codificação separada de fonte e canal. Como

Tabela 14: Distância para o limite de Shannon em BER=10^{-5}, Canal AWGN, $p_0 = 0,83079$, OPTA=$-2,35dB$.

Bloco(bit)	Método 1	Método 2	Método 3
1,2k	1,84 dB	2,10 dB	1,79 dB
12k	1,10 dB	1,16 dB	0,94 dB
120k	0,68 dB	0,89 dB	0,67 dB

Tabela 15: Distância para o limite de Shannon em BER=10^{-5}, Canal sobre desvanecimento Rayleigh, $p_0 = 0,83079$, OPTA=$-1,38dB$.

Bloco	Método 1	Método 2	Método 3
1,2k	2,20 dB	2,68 dB	2,27 dB
12k	1,12 dB	1,40 dB	1,17 dB
120k	0,69 dB	1,01 dB	0,79 dB

os códigos Huffman são códigos seqüenciais e a decodificação turbo não discrimina as posições dos erros, se houver um erro no início do bloco que está sendo decodificado por Huffman, então este erro poderá se propagar pela decodificação até o final do bloco, ocasionando assim uma avalanche de erros e conseqüentemente aumentando muito a taxa de erro do bloco.

Nas simulações da fonte B, o método 1 aparece sempre superior aos outros dois métodos na região de queda, porém, idêntico ao que ocorreu para a fonte A, este método apresentou o patamar de erro na faixa de 10^{-5} também. Nos métodos 2 e 3, este patamar de erro é menor que 10^{-6}, portanto não aparece nos gráficos. Outra característica importante apresentada nos gráficos da fonte B, é que para o canal AWGN o método 2 apresentou o pior desempenho na região de queda, já para o canal com desvanecimento Rayleigh, o método 3 foi quem apresentou o pior desempenho na região da queda.

Para comparar o desempenho dos códigos usados a uma taxa de erro de 10^{-5}, foi calculado a distância de cada curva para o limite máximo de Shannom (OPTA) do canal, considerando um código de taxa $\frac{1}{2}$. Para a fonte A as tabelas 14 e 15 mostram estas distâncias para o canal AWGN e Rayleigh respectivamente, já para a fonte B as tabelas correspondentes são a 16 e a 17.

Considerando os OPTAs obtidos com a fonte A, fica claro que o método 2 obteve uma performance inferior aos outros dois métodos, pois apresentou a maior distância ao OPTA para os três tamanhos de bloco, chegando a apresentar uma diferença em relação aos outros métodos de 0,48dB. Observando apenas os gráficos sobre o canal AWGN, fica claro que o método 3 obteve uma performance ligeiramente superior aos outros 2 métodos nos três casos, aparecendo

Tabela 16: Distância para o limite de Shannon em BER=10^{-5}, Canal AWGN, $p_0 = 0,899063$, OPTA=$-4,10dB$.

Bloco	Método 1	Método 2	Método 3
1,2k	2,48 dB	2,42 dB	2,43 dB
12k	1,35 dB	1,46 dB	1,38 dB
120k	0,95 dB	1,13 dB	1,21 dB

Tabela 17: Distância para o limite de Shannon em BER=10^{-5}, Canal sobre desvanecimento Rayleigh, $p_0 = 0,899063$, OPTA=$-3,43dB$.

Bloco	Método 1	Método 2	Método 3
1,2k	3,56 dB	2,69 dB	3,34 dB
12k	1,47 dB	1,52 dB	1,79 dB
120k	1,00 dB	1,18 dB	1,36 dB

inclusive, a uma distância de apenas 0,67dB ao limite máximo de Shannon para blocos de 120k bits. O mesmo não pode ser dito quando se trata do canal submetido ao desvanecimento Rayleigh. Para estas simulações o método 1 obteve a melhor performance, tanto na região de queda, quanto na BER de 10^{-5}. Porém o método 1 possui uma desvantagem que não aparece nos outros dois métodos, ele possui o patamar de erro muito elevado. Este método está a apenas 0,69dB do limite de Shannon para o canal com desvanecimento Rayleigh a uma BER de 10^{-5}, como pode ser visto na Figura 29.

Considerando as tabelas de performance 16 e 17 da fonte B, é possível verificar que o método 2 obteve a melhor performance para blocos pequenos, já para os blocos médios e grandes a melhor performance ficou com o método 1, chegando a apenas 0,95dB do limite máximo de Shannon, para o canal AWGN e bloco de 120k bits.

Comparando o nível de dificuldade de implementação o mais simples é sem dúvida o método 2, pois sua implementação é apenas uma modificação na equação do decodificador turbo usado. Já a implementação do método 1 exige maior processamento dos blocos codificador e decodificador, pois apresenta um estágio de codificação de fonte a mais em cada um. Independente disso o código Huffman é de fácil implementação e não exige muito processamento, pois toda a codificação pode ser realizada em apenas uma passada e não utiliza operações matemáticas como soma ou multiplicação. A maior desvantagem na implementação prática deste método é o fato do bloco codificado ter tamanho variável, o que na prática pode gerar dificuldades, pois o decodificador precisa ter conhecimento prévio do tamanho do bloco transmitido para o correto processamento. Já a implementação do código turbo do método 3 é muito similar

ao do método 2, porém este método possui a característica de apresentar uma constelação de símbolos com amplitudes desbalanceados, e conseqüentemente tem grande dependência de um circuito eletrônico rápido o suficiente e capaz de gerar com precisão estes valores.

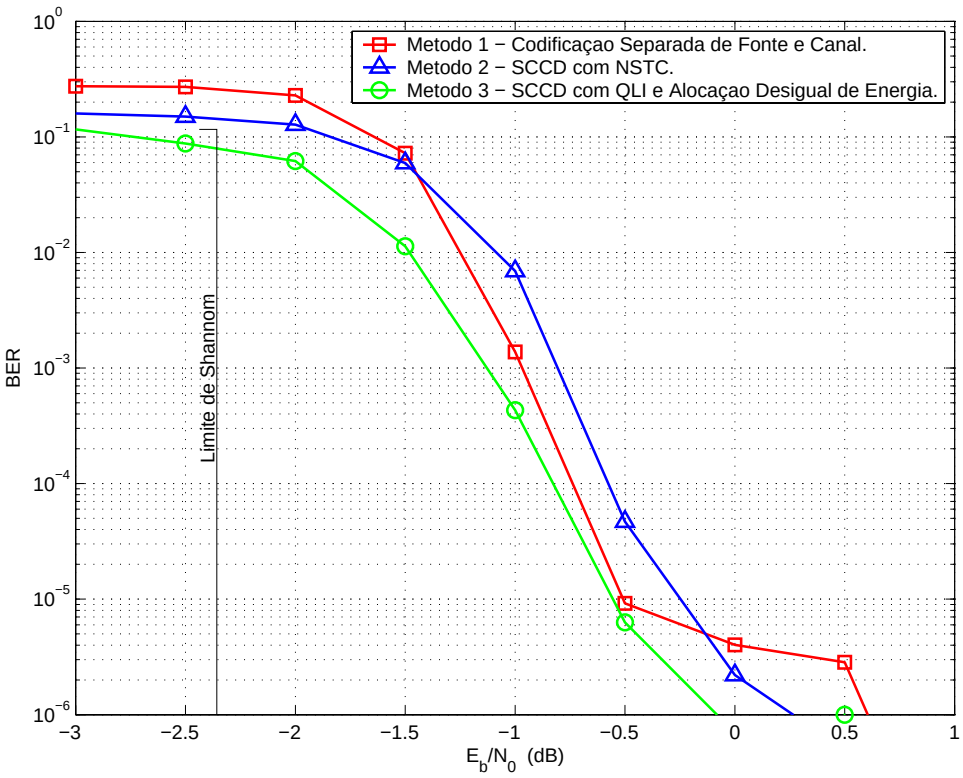

Figura 24: Canal AWGN, p_0^A=0,83079 e N=1,2k bits.

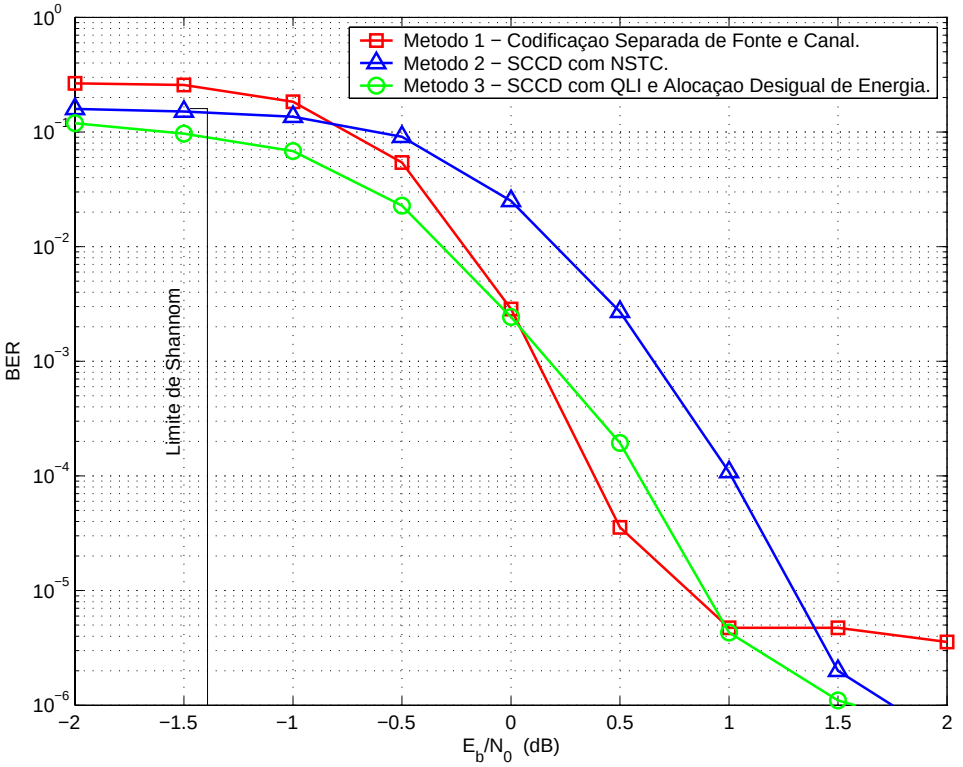

Figura 25: Canal Rayleigh, p_0^A=0,83079 e N=1,2k bits.

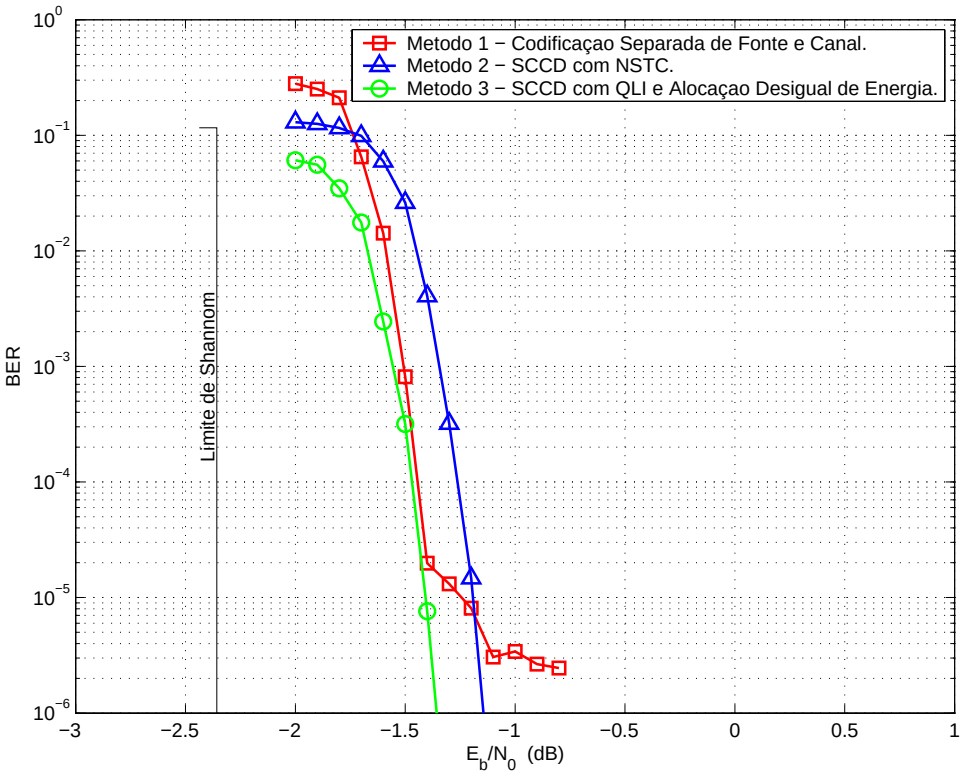

Figura 26: Canal AWGN, p_0^A=0,83079 e N=12k bits.

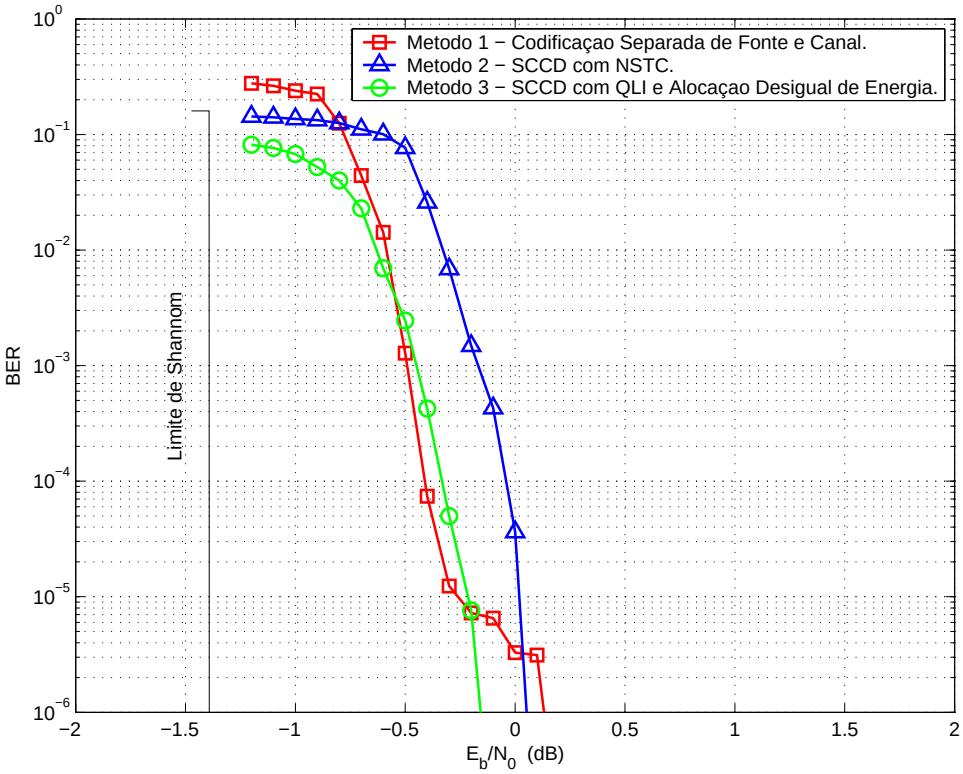

Figura 27: Canal Rayleigh, p_0^A=0,83079 e N=12k bits.

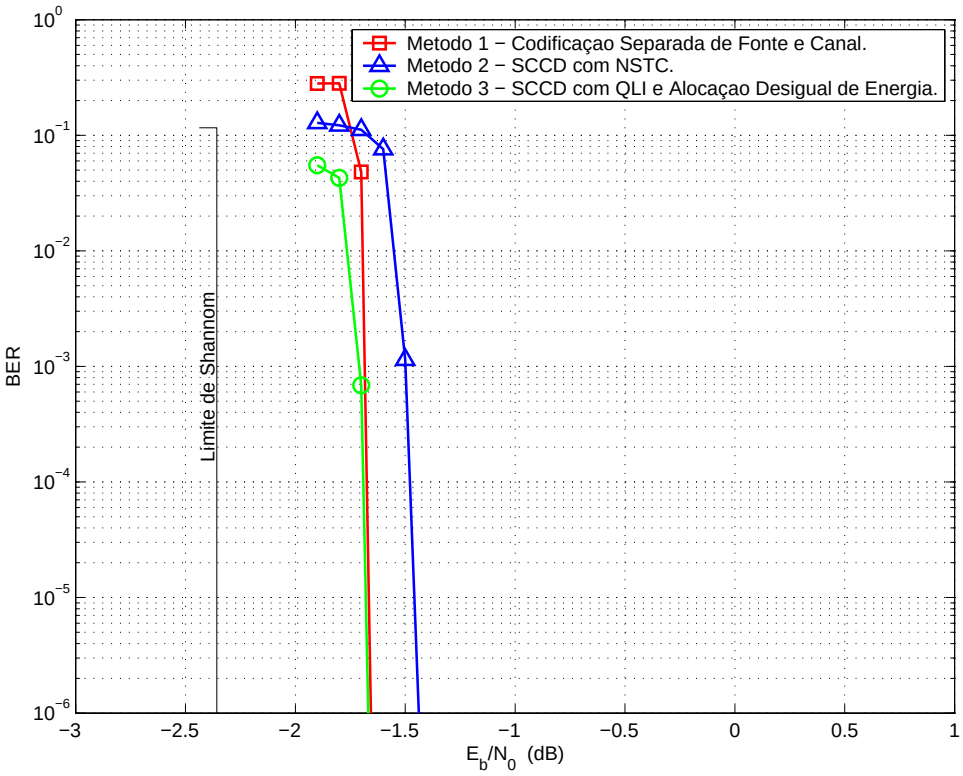

Figura 28: Canal AWGN, p_0^A=0,83079 e N=120k bits.

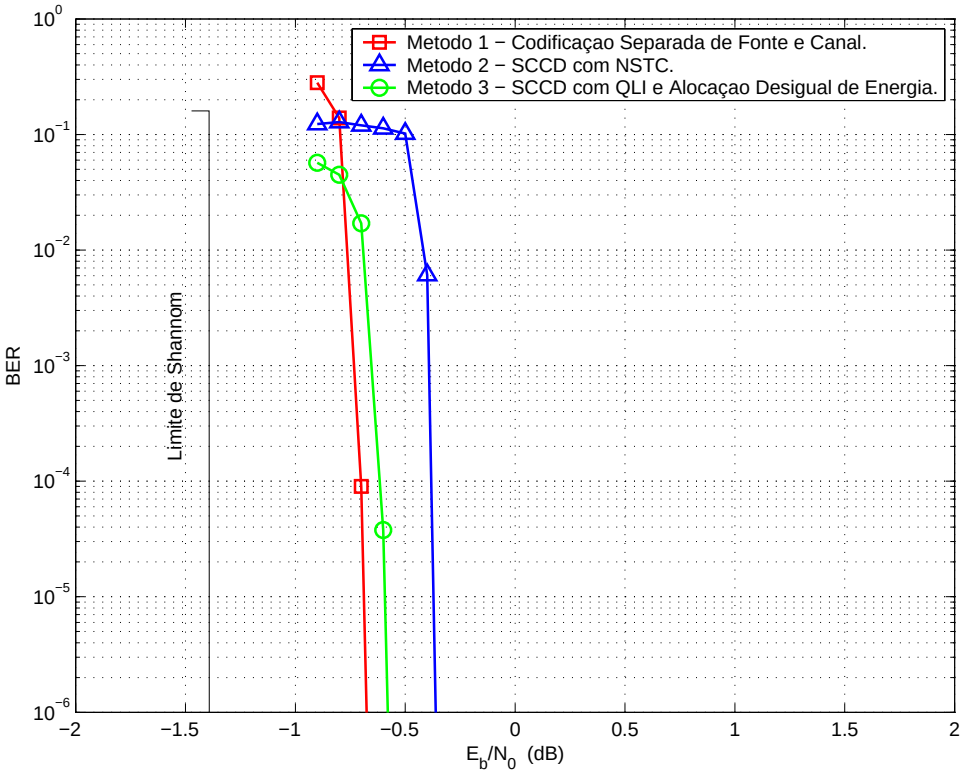

Figura 29: Canal Rayleigh, p_0^A=0,83079 e N=120k bits.

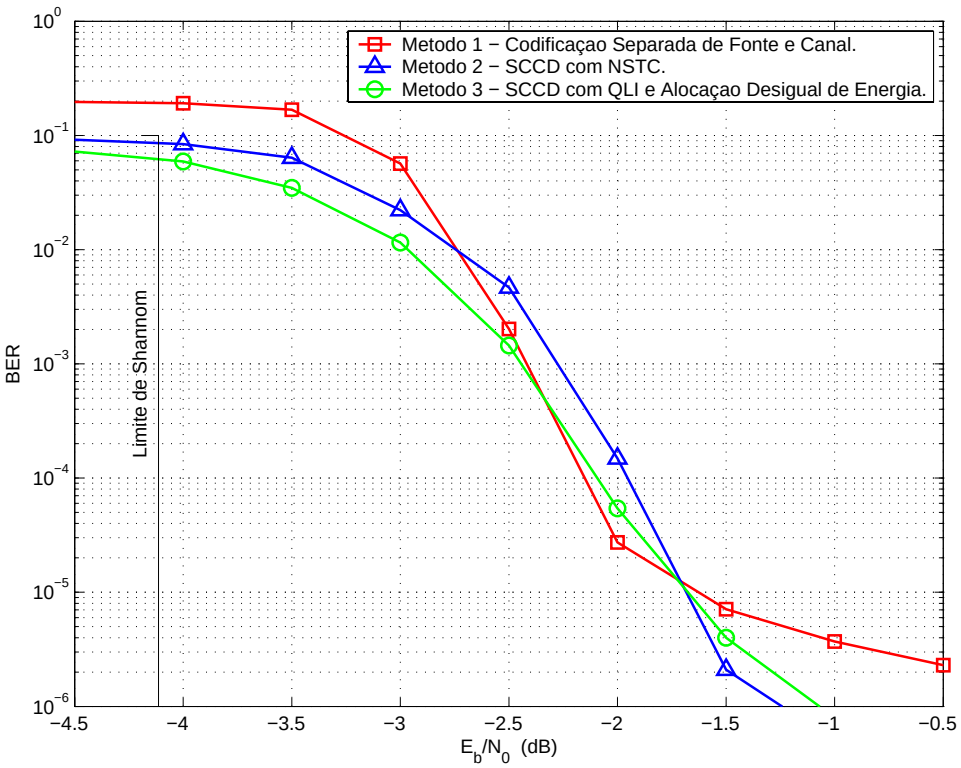

Figura 30: Canal AWGN, p_0^B=0,899063 e N=1,2k bits.

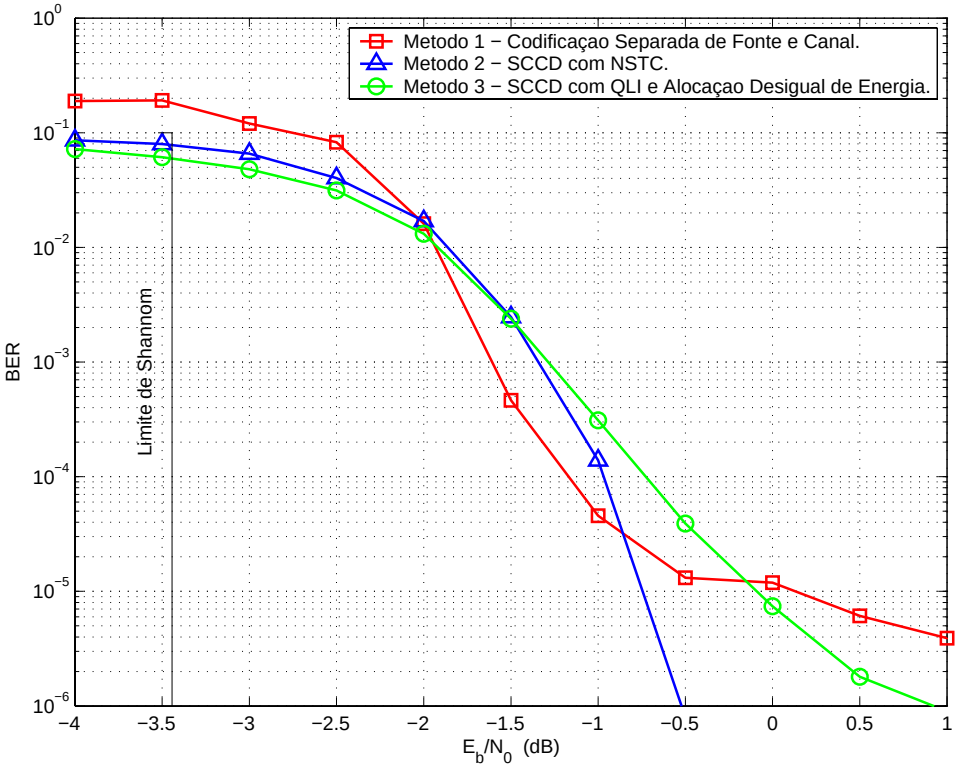

Figura 31: Canal Rayleigh, p_0^B=0,899063 e N=1,2k bits.

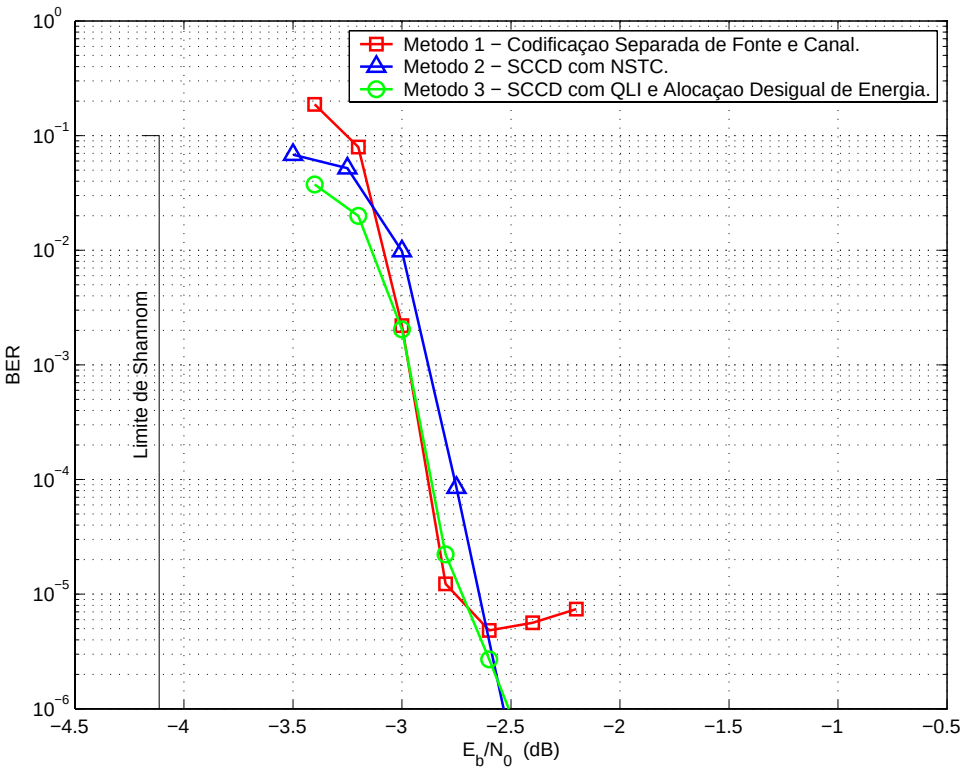

Figura 32: Canal AWGN, p_0^B=0,899063 e N=12k bits.

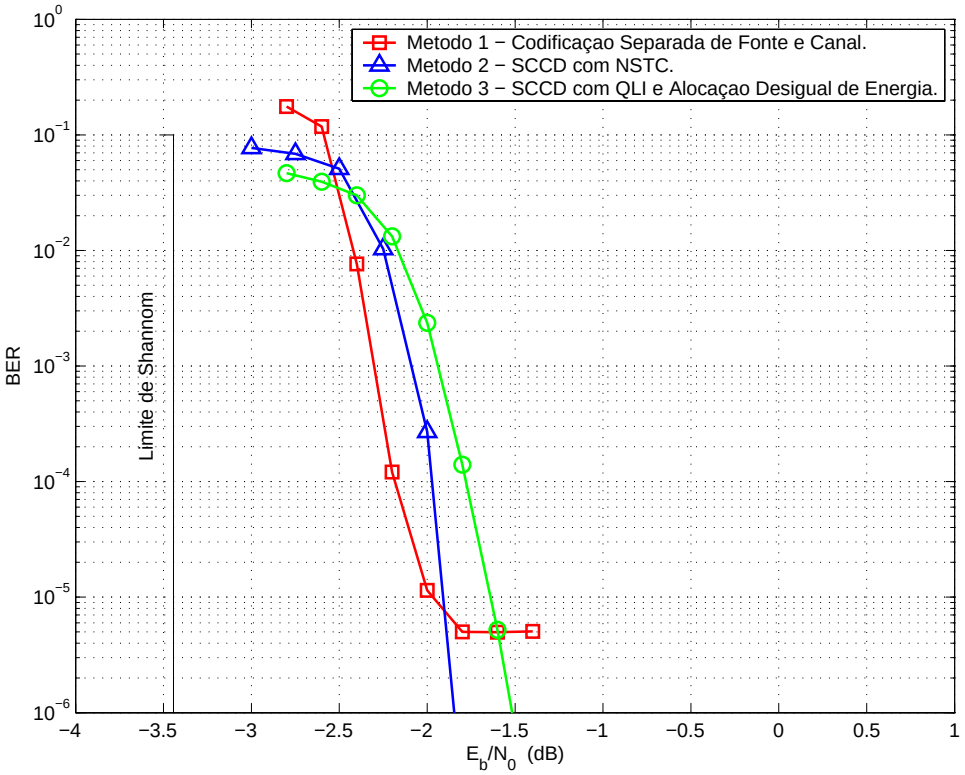

Figura 33: Canal Rayleigh, p_0^B=0,899063 e N=12k bits.

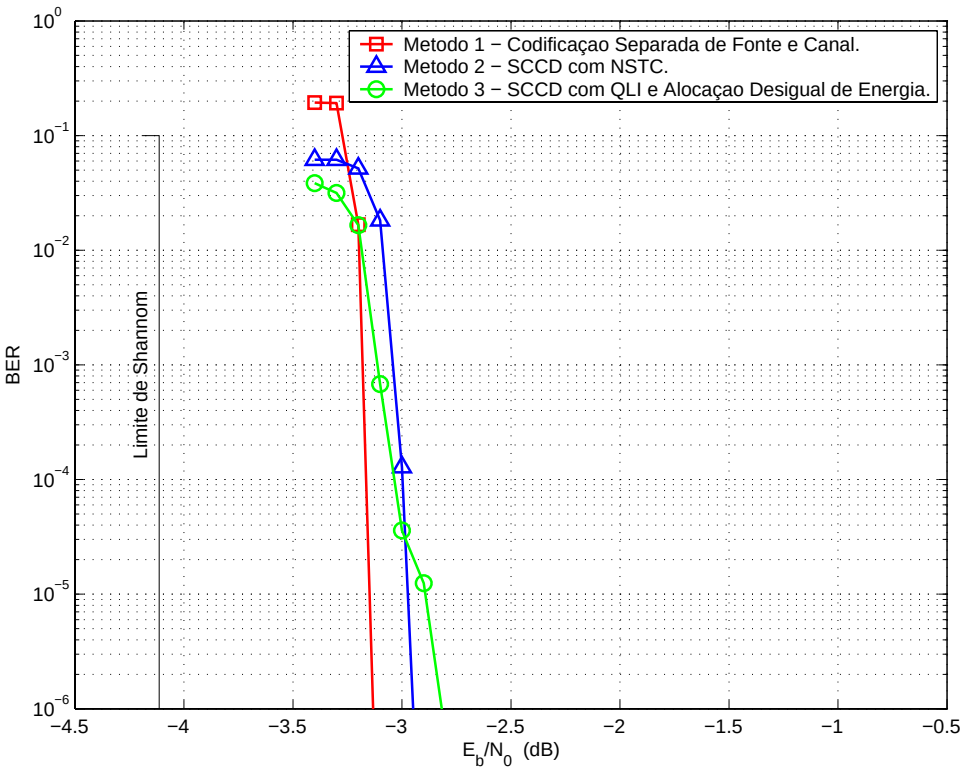

Figura 34: Canal AWGN, p_0^B=0,899063 e N=120k bits.

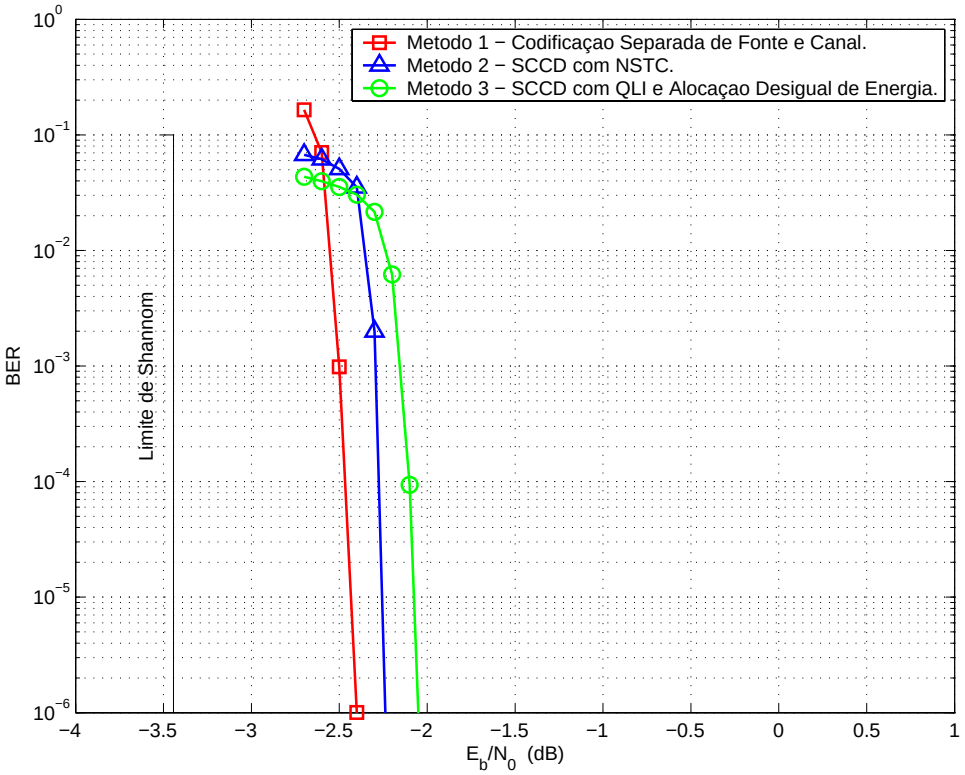

Figura 35: Canal Rayleigh, p_0^B=0,899063 e N=120k bits.

6 Conclusões

Este capítulo apresenta uma visão geral da dissertação e mostra possíveis trabalhos futuros de investigação na área.

6.1 Discussão e Conclusões

Nesta dissertação foram estudados três métodos de transmissão usando códigos turbo para fontes não-uniformes e sem memória, através de canais submetidos ao ruído AWGN e ao desvanecimento Rayleigh.

O Capítulo 2 apresentou uma visão geral dos sistemas de comunicação, modelos de fonte, modelos de canal, capacidade de canal, limite de Shannon, codificação de fonte e codificação de canal.

O Capítulo 3 explicou o funcionamento de um código turbo, desde o codificador até o decodificador. Apresentou o algoritmo simplificado de decodificação turbo chamado Max-Log-MAP e finalizou apresentando um gráfico do desempenho dos códigos turbo.

O Capítulo 4 introduziu os três métodos usados para transmissão de fontes não-uniformes estudados:

1. Método da codificação separada de fonte e canal;
2. Método da decodificação controlada pela fonte;
3. Método da decodificação controlada pela fonte usando código turbo QLI e com alocação desigual de energia.

O Capítulo 5 introduziu o modelo de simulação utilizado, e na seqüência demonstrou as característica específicas de cada método para que as comparações entre eles fossem justas. Por fim, apresentou em forma de gráficos os resultados das simulações.

Todas as simulações foram realizadas ao longo de aproximadamente 1 ano e as respostas de cada ponto simulado foram armazenadas em arquivos no formato texto para possíveis verificações futuras. Este estudo gerou também um artigo [39], apresentado na Conferência Internacional de Comunicações (ICC) de 2007, ocorrida em Glasgow, Escócia.

Os três métodos estudados apresentaram uma boa performance para as duas fontes estudadas, com pouca variação de desempenho entre eles. Com isso a escolha de qual método usar depende apenas das especificações de cada projeto.

O método 1 mostrou-se ótimo na região de queda, porém apresentou um alto patamar de erro, na faixa de 10^{-5}. Este método tem como ponto fraco o tamanho do bloco variável, o que na prática pode ser de difícil implementação. Além disso, sua complexidade computacional aumenta ligeiramente visto que apresenta um codificador e decodificador Huffman e os outros dois métodos não.

O Método 2 apesar de não apresentar uma boa performance na região de queda, apresentou um patamar de erro abaixo de 10^{-6} para todos os tamanhos de blocos simulados. A grande vantagem deste método é que o codificador turbo não precisa ser alterado para seu funcionamento, apenas uma simples mudança no decodificador turbo incluindo a informação *a priori* já é suficiente para alcançar seu desempenho.

O Método 3 apresentou uma performance ligeiramente similar à do método 1 na região da queda, porém apresentou o patamar de erro abaixo de 10^{-6} em todas as simulações. As únicas simulações onde este método não apresentou um bom desempenho foram para a fonte B com o canal sobre desvanecimento Rayleigh, onde ficou atrás dos dois outros métodos, porém ainda assim com uma boa performance.

6.2 Trabalhos Futuros

Este trabalho estudou apenas duas distribuições de fontes não-uniformes. Uma possível extensão deste trabalho, seria o estudo de codificadores para fontes com alto grau de não-uniformidade, por exemplo, acima de p_0=0,99. Este nível de redundância possibilitaria o uso de um codificador Huffman que compactasse a fonte a uma taxa muito pequena, possibilitando assim o uso de um código turbo de taxa muito baixa.

A obtenção de códigos turbo não-sistemáticos, conforme o método 2, para altos graus de não-uniformidade da fonte é um grande desafio que também poderia ser melhor estudado.

Observou-se no método 1 que após a codificação Huffman, ainda existe um pequeno resíduo

de redundância no bloco compactado. Este resíduo poderia ser utilizado como informação *a priori* no decodificador turbo de um trabalho futuro.

Na tentativa de diminuir o efeito da avalanche de erros que o decodificador Huffman pode provocar, outros tipos de codificadores de fonte também poderiam ser estudados para o método 1, inclusive codificadores de fonte com taxa fixa, para evitar o problema de blocos com tamanho variável.

Uma grande desvantagem do método 1 foi o alto patamar de erro, principalmente em blocos pequenos e médios. Em [40] e [41] são mostrados projetos de entrelaçadores que melhoram a performance dos códigos principalmente na região do patamar de erros. Um trabalho futuro seria a procura e o estudo destes entrelaçadores, visando melhorar a performance e conseqüentemente diminuir o patamar de erros observado neste método.

Referências Bibliográficas

[1] C. Berrou, A. Glavieux, P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding Turbo Codes- Proc. Int. Conf. Communications, p. 1064-1070, Mai. 1993.

[2] G. C. Zhu, F. Alajaji, "Design of turbo codes for nonequiprobable memoryless source", 39th Allerton Conf. Communication, Monticello, IL, p. 1253-1262, Out. 2001.

[3] G. C. Zhu, F. Alajaji, "Turbo codes for non uniform memoryless sources over noisy channels,"IEEE Commun. Letters, vol. 6, p. 64-66, Fev. 2002.

[4] R. D. Souza, G. I. Shamir, J. Garcia-Frias e Kai Xie, "Non-systematic turbo coding with unequal energy allocation for nonuniform memoryless sources,"Proc. IEEE ISIT, Set. 2005.

[5] F. Cabarcas, R. D. Souza e J. Garcia-Frias, "Turbo Coding of Strongly Non-Uniform Memoryless Sources With Unequal Energy Allocation and PAM Signaling,"IEEE Transactions on Signal Processing, vol. 54, no. 5, p. 1942-1946, Mai. 2006.

[6] C. E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Jornal, vol.27, p. 379-423, 623-656, Out. 1948.

[7] F. Cabarcas, R. D. Souza, J. Garcia-Frias, "Source-controlled turbo coding of non-uniform memoryless sources based on unequal energy allocation", IEEE International Symposium on Information Theory, Chicago, IL, p. 166, Jun 2004.

[8] J. Hagenauer, "Source-controlled channel decoding,"IEEE Trans. On Commun., vol. 43, no. 9, pp. 2449-2457, Set. 1995.

[9] G. C. Zhu, F. Alajaji, J. Bajcsy, and P. Mitran, "Transmission of nonuniform memoryless sources via nonsystematic turbo codes,"IEEE Trans. Commun., vol. 52, n. 8, p. 1344-1354, Ago. 2004.

[10] J. G. Proakis, *"Digital Comunications"*, Terceira Edição, McGraw-Hill, p. 767-768, 1995.

[11] P. Elias, "Error-Free Coding", IEEE Trans. Inform. Theory, vol. IT-4, p. 29-37, Set. 1954.

[12] S. Haykin, "Sistemas de Comunicação Analógicos e Digitais", Quarta edição, Bookman, 2004.

[13] D. A. Huffman, "A method for the construction of minimum-redundancy codes", Proceedings of the I.R.E., p. 1098-1102, Set. 1952

[14] M. Nelson e J. Gailly: *"The Data Compression Book"*, M&T Books, 1995.

[15] P. Elias, "Coding for noisy channels", IRE Conv. Rec., 4ª Parte, p. 37-47, 1955.

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

[17] J. B. Cain, G. C. Clark e J. M. Geist, "Punctured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding", IEEE Transactions on Information Theory, p. 97-100, Jan. 1979.

[18] A. S. Barbulescu e S. S. Pietrobon, "Interleaver design for turbo codes", Electron. Lett., vol. 30, pp. 2107-2108, Dez. 1994.

[19] F. Daneshgaran e M. Mondin, "Design of interleavers for turbo codes: iterative interleaver growth algorithms of polynomial complexity", IEEE Trans. Inform. Theory, vol. 45, pp. 1845-1859, Set. 1999.

[20] W. Feng, J. Yuan e B. Vucetic, "A code-matched interleaver design for turbo codes", IEEE Trans. Commun., vol. 50, pp. 926-937, Jun. 2002.

[21] M. Ferrari, F. Scalise e S. Bellini, "Prunable s-random interleavers", in IEEE Conference on Communications, vol. 3, pp. 1711-1715, Abr. 2002.

[22] C. Fragouli e R. Wesel, "Semi-random interleaver design criteria", in Global Telecommunications Conference, pp. 2352-2356, 1999.

[23] S. Dolinar e D. Divsalar, "Weight distributions for turbo codes using random and nonrandom permutations", JPL TDA Progress Report, v42. p. 56-65, Ago. 1995.

[24] L. R. Bahl, J. Cocke, F. Jelinek e J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", na IEEE Trans. Information Theory, vol. 20, n.2, p. 284-287, Mar. 1974.

[25] D. J. Costello, Jr., P. C. Massey, O. M. Collins e O. Y. Takeshita, "Some reflections on the mythology of turbo codes", em Proc. 3rd ITG Conf. Source-Channel Coding, Munique, Alemanha, pp. 157-160, Jan. 2000.

[26] O. M. Collins, O. Y. Takeshita e D. J. Costello, Jr., "Iterative decoding of nonsystematic turbo codes", em Proc. Int. Symp. Information Theory, Sorrento, Italia, p.172, Jun. 2000.

[27] D. J. Costello, Jr., H. A. Cabral e O. Y. Takeshita, "Some thoughts on the equivalence of systematic and nonsystematic convolutional encoders", em Codes, Graphs and Systems, R. E. Blahut e R. Koetter, Editora. Norwell, 2002.

[28] P. C. Massey r D. J. Costello, Jr., "Turbo codes with recursive nonsystematic quick-look-in constituent codes", em Proc. Int. Symp. Information Theory, Washington, DC, p. 141, Jun 2001.

[29] P. Robertson, E. Villebrun e P. Hoeber, "Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain", Institute for Communications Technology, Germany Aerospace Research Establishment (DLR). IEEE 0-7803-2486-2, p. 1009-1013, 1995

[30] J. Hagenauer e P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications". Em Proc GLOBECOM'89, p. 1680-1686, 1989.

[31] P. Robertson e P. Hoeher. "Optimum and sub-optimum maximum a posteriori algorithms suitable for Turbo decoding", Europ. Tran. On Tele. and Related Areas, p. 119-125, 1997.

[32] W. Kock e A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying intersymbol interference", em Proc. GLOBECOM ’90, p. 1679-1684, Dez. 1990.

[33] V. Franz e J. B. Anderson, "Concatenated Decoding with a Reduced Search BCJR Algorithm", IEEE Journal on Selected Areas on Communications, vol. 16, no. 2, p. 186 - 195, Fev. 1998.

[34] A. Banerjee, P. Vatta, F. Scanavino e D. J. Costello JR., "Nonsystematic turbo codes,"IEEE Trans. Commun., vol. 53, no. 11, p. 1841-1849, Nov. 2005.

[35] J. Hagenauer, "The Turbo Principle: Tutorial Introduction and State of the Art", em Proceedings of the International Symposium on Turbo Codes and Related Topics., Brest, França, p. 1 - 11, Set. 1997.

[36] C. B. Schlegel e L. C. Pérez. "Trellis and Turbo Coding", IEEE Press Series on Digital and Mobile Communication, p. 292-293, 2004.

[37] D. Divsalar e F. Pollara, "On the Design of Turbo Codes", The JPL TDA Progress Report 42-123, p. 99-121, Nov. 1995.

[38] G. I. Shamir e Kai Xie, "Design of Non-Systematic Turbo Codes for Universal Source Controlled Channel Decoding", In the Proc. of IEEE ISOC ITW2005, p. 200-204, 2005.

[39] G. Titericz, R. D. Souza, J. G. Frias e G. Shamir. "Comparing Different Transmission Strategies Using Turbo Codes for Nonuniform Memoryless Sources". Proceedings of the 2007 IEEE International Conference on Communications - ICC, v. 1. p. 2722-2727, Jun. 2007.

[40] S. Crozier, J. Lodge, P. Guinand e A. Hunt, "Performance of Turbo-Codes with Relative Prime and Golden Interleaving Strategies", Proceedings of the 6th International Mobile Satellite Conference (IMSC ’99), Ottawa, Ontario, Canada, p. 268-275, Jun. 1999.

[41] F. Chan, "Matched Interleavers for Turbo Codes With Short Frames", The Seventh Canadian Workshop on Information Theory (CWIT), Jun. 2001.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo