MODULAÇÃO CODIFICADA E COMPRESSÃO DE SINAIS

José Fernando Leite de Oliveira

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Gelson Vieira Mendonça, Ph.D.

Prof. Eduardo Antônio Barros da Silva, Ph.D.

Prof. Sérgio Lima Netto, Ph.D.

Prof. Abraham Alcaim, Ph.D.

Prof. Weiler Alves Finamore, Ph.D.

Prof. Ernesto Leite Pinto, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

DEZEMBRO DE 2003

OLIVEIRA, JOSÉ FERNANDO LEITE DE

Modulação Codificada e Compressão de Sinais [Rio de Janeiro] 2003

XIV, 125 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia Elétrica, 2003)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. *Turbo* Quantização
2. Codificação por Treliças
3. Modulação Codificada
4. Quantização Codificada
5. Códigos *Turbo*

I. COPPE/UFRJ II. Título (série)

# Agradecimentos

Os meus agradecimentos são para todos aqueles que contribuíram direta ou indiretamente para a elaboração desta tese e, especialmente, para minha família que sempre me apoiou.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

MODULAÇÃO CODIFICADA E COMPRESSÃO DE SINAIS

José Fernando Leite de Oliveira

Dezembro/2003

Orientadores: Gelson Vieira Mendonça
Eduardo Antônio Barros da Silva

Programa: Engenharia Elétrica

A modulação codificada por treliças (TCM – *Trellis-Coded Modulation*) aumentou expressivamente o desempenho de sistemas de transmissão de dados. A quantização codificada por treliças (TCQ – *Trellis-Coded Quantisation*), cujo desempenho em termos de taxa × distorção é excelente para diversas fontes, foi proposta com base na TCM. Recentemente, a turbo modulação codificada por treliças (TTCM – *Turbo TCM*) aumentou significativamente o desempenho da TCM. Neste trabalho, um esquema de turbo quantização baseado na TTCM é proposto. A principal motivação para tal proposta é o excelente desempenho da TCQ. A turbo quantização foi desenvolvida e as simulações comprovaram que a Turbo Quantização Codificada por Treliças (TTCQ – *Turbo TCQ*) produz resultados consideravelmente melhores que os da TCQ, apesar de problemas de convergência e da necessidade de se utilizar um elevado número de iterações para obter tais resultados. Além disto, o desempenho conjunto da TTCQ e TTCM, em esquemas de codificação conjunta fonte e canal, foi também avaliado e é melhor que o da TCQ e TCM para relações sinal-ruído de canal relativamente altas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

# CODED MODULATION AND SIGNAL COMPRESSION

José Fernando Leite de Oliveira

December/2003

Advisors: Gelson Vieira Mendonça
Eduardo Antônio Barros da Silva

Department: Electrical Engineering

Trellis-coded modulation (TCM) has expressively improved the performance of data transmission systems. Trellis-coded quantisation (TCQ), whose performance in terms of rate $\times$ distortion is excellent for several sources, was proposed based on TCM. Recently, turbo trellis-coded modulation (TTCM) has significantly improved the performance of TCM. This work proposes a turbo quantisation scheme based on TTCM. The main motivation for such proposal is the excellent performance of TCQ. Turbo quantisation was developed and simulations have shown that Turbo Trellis-Coded Quantisation (TTCQ) performs considerably better than TCQ, in spite of convergence problems and the need of a large number of iterations in order to obtain such performance. In adition, the joint performance of TTCQ and TTCM, for joint source/channel schemes, was also evaluated and it is better than that of TCQ and TCM for relative high channel signal to noise ratios.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Desde que SHANNON [1] apresentou seu teorema da codificação de canal, que pode ser considerado, sem exagero, um dos pilares da teoria da informação, muito esforço foi despendido para tentar realizar o que este teorema estipula: é possível construir bons códigos para um canal ruidoso desde que a taxa de transmissão não exceda a capacidade do canal. Neste contexto, a expressão "bom código" significa obter probabilidade de erro tão pequena quanto se queira para uma dada potência de transmissão. Um código de canal pode ser considerado melhor que um outro se a probabilidade de erro de transmissão é menor para a mesma potência de transmissão. É importante também dar atenção a fatores como complexidade de decodificação do código que está relacionada com sua estrutura.

Códigos não estruturados, em geral, não possuem algoritmos de decodificação realizáveis. Então, basicamente, uma das ocupações da teoria da informação é construir códigos de canal capazes de reduzir a potência de transmissão para uma dada probabilidade de erro até o limite imposto pelo teorema de Shannon. Para se ter uma noção da importância desta tarefa, cada vez que se reduz a potência de transmissão em 20% (1 dB) estima-se que dezenas de milhões de dólares são poupados em setores como o das transmissões espaciais de longa distância [2].

Em 1993, BERROU *et al.* [3] introduziram os códigos turbo cuja principal característica é realizar taxas de transmissão muito próximas da capacidade teórica do canal, iniciando, com isto, o que pode ser o fim do problema de se encontrar os bons códigos prometidos pelo teorema de Shannon.

Um outro problema da teoria da informação é o da representação de uma

variável aleatória, que pode assumir qualquer valor dentro do conjunto dos números reais, utilizando um número finito de *bits* por amostra e produzindo a menor distorção média possível entre as amostras e suas representações. Este problema dá origem a teoria taxa × distorção que pode ser vista como dual da teoria de codificação de canal. Isto porque um bom codificador de canal pode ser transformado num bom código taxa × distorção [4].

UNGERBOECK [5] propôs um esquema de codificação de canal, a modulação codificada por treliças (TCM – *Trellis-Coded Modulation*), cuja principal característica é aumentar a eficiência espectral em relação aos esquemas de modulação multinível não codificados, juntando codificação e modulação num mesmo bloco. A modulação codificada por treliças foi utilizada por MARCELLIN [6] como quantizador, um código taxa × distorção que ficou conhecido como TCQ (*Trellis-Coded Quantisation*), obtendo resultados comparáveis, por vezes superiores, aos de esquemas de quantização até então disponíveis. Isto é um bom exemplo de como um codificador de canal pode ser utilizado para resolver um problema de taxa × distorção.

Em 1995, ROBERTSON *et al.* [7] fizeram uma fusão dos códigos turbo com os códigos de UNGERBOECK [5], produzindo um esquema de codificação de canal similar aos códigos turbo e com a eficiência espectral dos códigos de UNGERBOECK [5]: a turbo modulação codificada por treliças (TTCM – *Turbo Trellis-Coded Modulation*). Visto que os códigos de UNGERBOECK [5] puderam ser utilizados diretamente como códigos taxa × distorção, por que não tentar utilizar os códigos de ROBERTSON *et al.* [7] para produzir um novo código taxa × distorção?

Como os codificadores de ROBERTSON *et al.* [7] são mais eficientes que os de UNGERBOECK [5], pelo menos para valores de relação sinal-ruído suficientemente altos, espera-se que os quantizadores derivados daqueles códigos sejam melhores que os de MARCELLIN [6], ou seja, produzam uma distorção média menor. E esta é, basicamente, a motivação deste trabalho, ou seja, desenvolver a turbo quantização codificada por treliças (TTCQ – *Turbo TCQ*).

Em 1991, FISCHER *et al.* [8] propuseram um esquema de codificação conjunta de fonte e canal com TCQ e TCM. A idéia é codificar uma fonte sem memória utilizando a TCQ que por sua vez gera diretamente os símbolos de canal a serem

transmitidos. No receptor, o decodificador TCM gera diretamente os símbolos de reconstrução da fonte. Este esquema de codificação conjunta de fonte e canal também mostrou-se muito eficiente em relação a outros até então disponíveis [8]. Como a TTCM e a TTCQ são esquemas baseados na TCM e na TCQ, decidiu-se também avaliar o desempenho conjunto da TTCQ e TTCM por meio de simulações e compará-lo ao da TCQ e TCM.

## 1.1 Organização da Tese

O capítulo 1, que corresponde a esta introdução, apresenta a motivação deste trabalho. No capítulo 2, são introduzidos conceitos básicos relacionados aos códigos turbo e a sua estrutura de decodificação iterativa que emprega o algoritmo BCJR-MAP[1] (*Maximum A Posteriori*) [9]. Além disto, outros algoritmos utilizados na decodificação dos códigos turbo como BCJR-LogMAP e BCJR-MaxLogMAP [10] são também descritos. Os conceitos e algoritmos apresentados neste capítulo são de fundamental importância para se compreender a TTCM de ROBERTSON *et al.* [7] que também emprega o princípio turbo.

O capítulo 3 apresenta, sucintamente, a modulação e a quantização codificadas por treliças, bem como um breve paralelo entre a teoria de codificação de canal e a teoria taxa $\times$ distorção. Os conceitos relativos à TCM também são necessários para compreender a estrutura da TTCM e sua operação. Além disto, este capítulo mostra como a TCM, que é um esquema de codificação de canal, foi "transformada" na TCQ, que é um esquema codificação de fonte, explorando a dualidade entre as teorias de codificação de canal e taxa $\times$ distorção [4]. E, como a TTCM é derivada da TCM, este procedimento servirá como base para desenvolver a TTCQ.

No capítulo 4, é feita uma descrição das principais características da turbo modulação codificada por treliças, destacando-se o processo iterativo de decodificação com o algoritmo BCJR-MAP para fontes não binárias [10, 11]. Neste capítulo, é desenvolvida a TTCQ, um novo esquema de quantização para fontes sem memória derivado da TTCM, seguindo passos semelhantes aos de MARCELLIN [6]. Ficará

[1] As letras B, C, J e R correspondem às iniciais dos nomes dos autores deste algoritimo, ou seja, *Bahl*, *Cocke*, *Jelinek* e *Raviv*.

evidente neste capítulo que a transformação da TTCM num código taxa × distorção é mais complexa que a da TCM na TCQ. Ainda neste capítulo, os resultados obtidos com este novo quantizador são relatados.

No capítulo 5, a TTCQ e a TTCM são utilizadas na codificação conjunta de fonte e canal e os resultados comparados com aqueles obtidos pelo emprego conjunto de TCQ e TCM [8, 12].

Finalmente, o capítulo 6 contém as conclusões gerais a respeito dos resultados obtidos nos capítulos 4 e 5, e propõe alguns trabalhos futuros.

# Capítulo 2

# Códigos Turbo

Em 1948, SHANNON [1] publicou um dos artigos mais importantes para a teoria da informação. Ele estabeleceu os limites teóricos para a taxa máxima de transmissão de informação num canal ruidoso no Teorema da Capacidade do Canal. Este teorema assegura a existência de códigos capazes de atingir a taxa máxima de transmissão para blocos de código suficientemente longos. Este teorema, porém, não mostra como construir tais códigos e por cerca de 50 anos não se encontrou códigos capazes de transmitir perto da taxa estabelecida por SHANNON [1]. Então, em 1993, BERROU *et al.* [3] apresentam os primeiros códigos que realizam este feito: os Códigos Turbo.

## 2.1 Codificação e Capacidade do Canal

De acordo com DIVSALAR *et al.* [13], antes do surgimento dos códigos turbo, os teóricos da codificação insistiam em atacar o problema do projeto de bons codificadores desenvolvendo códigos extremamente estruturados, os quais permitem a construção de decodificadores práticos, apesar de a teoria desenvolvida por SHANNON [1] sugerir que códigos escolhidos "aleatoriamente" teriam uma taxa de transmissão próxima da capacidade do canal se os tamanhos dos blocos fossem suficientemente longos, conforme o teorema 2.1.

De fato, a busca por decodificadores práticos para códigos "quase" aleatórios para tamanhos de blocos longos não era considerada seriamente [13]. Esta tendência pode ser justificada levando em conta que um conjunto de palavras código gerado

aleatoriamente com uma distribuição apropriada exigiria, utilizando o esquema mais simples, uma tabela com um tamanho que cresceria exponencialmente com o tamanho do bloco, tornando inviável sua implementação prática [4]. E, talvez, por estes motivos a reação inicial dos teóricos da codificação tenha sido de profundo ceticismo diante do desempenho tão surpreendente dos códigos turbo, anunciado por BERROU *et al.* [3], que utilizavam códigos componentes relativamente simples e permutadores.

**Teorema 2.1 (Capacidade do Canal [4]).** *Todas as taxas abaixo da capacidade $C$ podem ser alcançadas. Especificamente, para todo $\epsilon > 0$ e $R < C$, existe uma seqüência de códigos $(2^{NR}, N)$, ou seja, contendo $2^{NR}$ palavras código de $R$ bits por símbolo de tamanho $N$, com máxima probabilidade de erro $P_e^{(N)} < \epsilon$. Por outro lado, qualquer seqüência de códigos $(2^{NR}, N)$ com $P_e^{(N)} < \epsilon$ deve ter $R \leq C$.*

Portanto, até o surgimento dos códigos turbo, os códigos estruturados que permitiam codificação e decodificação relativamente simples não possuíam uma taxa assintótica próxima da capacidade do canal. Os códigos turbo produzem palavras código estruturadas e com características de um código gerado aleatoriamente. Isto permite que se tenha uma taxa assintótica de transmissão próxima da capacidade do canal e um decodificador realizável na prática [3].

## 2.2 O Turbo Codificador

O turbo codificador da forma descrita por BERROU *et al.* [3] consiste da concatenação de dois codificadores convolucionais em paralelo. Apesar da nomenclatura, somente o decodificador, descrito na seção 2.4, utiliza o princípio turbo. O diagrama de blocos deste tipo de codificador é mostrado na figura 2.1. Trata-se de um código sistemático produzido com o auxílio de dois codificadores convolucionais recursivos [14], um permutador (*interleaver* [2, 10]) e, opcionalmente, pode-se utilizar um perfurador para obter taxas diferentes da taxa básica de 1/3. Desta forma, cada *bit* da mensagem $u_{k-1}$ produz três *bits* de saída $v_{k-1}^{S} v_{k-1}^{1P} v_{k-1}^{2P}$ e três símbolos de canal $x_{k-1}^{S} x_{k-1}^{1P} x_{k-1}^{2P}$, no caso da taxa 1/3, para $k = 1, 2, \ldots, N$, onde $N$ é a dimensão do vetor de entrada $\mathbf{u} = [u_0 u_1 \ldots u_{N-1}]^T$, sendo a letra $T$ indicação de transposição. Para produzir um código com uma taxa 1/2, por exemplo, são desprezados

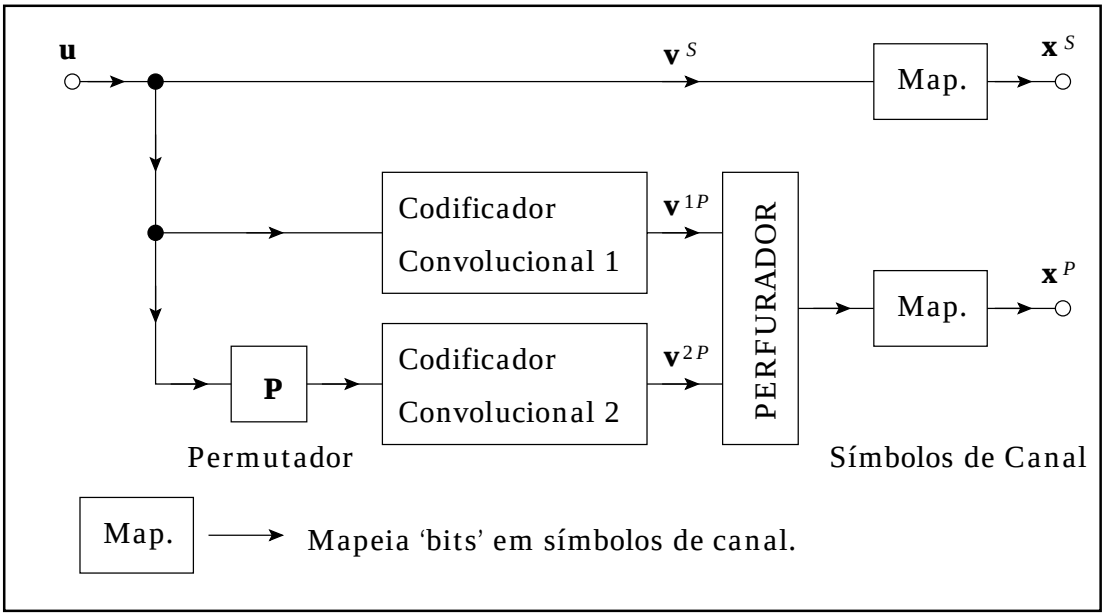


Figura 2.1: O turbo codificador.

alternadamente os *bits* de paridade $v_{k-1}^{1P}$ e $v_{k-1}^{2P}$.

Os códigos turbo, apesar de serem constituídos por codificadores convolucionais, podem ser vistos como códigos de bloco já que a codificação se processa de $N$ em $N$ *bits*, condição imposta pelo tamanho do permutador. Normalmente, não se adota $N$ menor que 1000. Além dos próprios códigos convolucionais, o tamanho do bloco e o projeto do permutador são fatores determinantes da eficiência dos códigos turbo [3].

## 2.3 O Algoritmo BCJR-MAP

Antes do surgimento dos códigos turbo, já havia interesse em estratégias subótimas de decodificação para códigos concatenados envolvendo, usualmente, dois decodificadores operando conjunta e iterativamente. Um desses algoritmos era o BCJR, publicado por Bahl, Cocke, Jelinek e Raviv [9] de cujas iniciais advém o nome do algoritmo. Foi este algoritmo que BERROU *et al.* [3] adotaram na decodificação iterativa dos códigos turbo.

O vetor de símbolos $\mathbf{x} = [x_0 x_1 \ldots x_{N-1}]^T$ é transmitido através de um canal e, em geral, é corrompido por ruído aditivo $\mathbf{e} = [e_0 e_1 \ldots e_{N-1}]^T$, resultando no vetor $\mathbf{y} = [y_0 y_1 \ldots y_{N-1}]^T$, sendo $\mathbf{y} = \mathbf{x} + \mathbf{e}$, que é decodificado para recuperar a

mensagem $\mathbf{u}$. Então, o decodificador BCJR-MAP símbolo a símbolo decide que o símbolo $u_{k-1} = 1$ foi transmitido se

$$\Pr\{U_{k-1} = 1|\mathbf{y}\} > \Pr\{U_{k-1} = 0|\mathbf{y}\}. \tag{2.1}$$

Caso contrário, decide que $u_{k-1} = 0$ foi transmitido. Utilizando o logaritmo da razão das probabilidades *a posteriori* cuja expressão é

$$L_{k-1}(1) = \ln\left[\frac{\Pr\{U_{k-1} = 1|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}}\right] \tag{2.2}$$

tem-se que

$$u_{k-1} = \begin{cases} 1, L_{k-1}(1) > 0 \\ 0, L_{k-1}(1) < 0 \end{cases} \tag{2.3}$$

para $k = 1, 2, \ldots, N$. Utilizando o fato de os codificadores serem convolucionais e, portanto, representáveis na forma de treliças [14], pode-se escrever a equação 2.2 na sua forma equivalente

$$L_{k-1}(1) = \ln\left[\frac{\sum\limits_{(s',s)\in\mathcal{B}^1_{k-1}} \Pr\{S_{k-1} = s', S_k = s|\mathbf{y}\}}{\sum\limits_{(s',s)\in\mathcal{B}^0_{k-1}} \Pr\{S_{k-1} = s', S_k = s|\mathbf{y}\}}\right], \tag{2.4}$$

onde $\mathcal{B}^1_{k-1}$ é o conjunto de todas as transições de estado $(s', s)$ produzidas por uma entrada $u_{k-1} = 1$, $\mathcal{B}^0_{k-1}$ é o conjunto correspondente de transições $(s', s)$ produzidas por uma entrada $u_{k-1} = 0$ e $s', s \in \{0, 1, \ldots, N_e - 1\}$, sendo $N_e$ o número de estados da treliça do código convolucional. Observando que

$$\Pr\{S_{k-1} = s', S_k = s|\mathbf{y}\} = \frac{\Pr\{S_{k-1} = s', S_k = s, \mathbf{y}\}}{\Pr\{\mathbf{y}\}} \tag{2.5}$$

então, definindo $\mathbf{y}_i^j = (y_i, y_{i+1}, \ldots, y_{j-1}, y_j)$ para $i < j$, tem-se que

$$\begin{aligned}
\Pr\{S_{k-1} = s', S_k = s, \mathbf{y}\} &= \Pr\{S_{k-1} = s', S_k = s, \mathbf{y}_0^{k-2}, y_{k-1}, \mathbf{y}_k^{N-1}\} \\
&\overset{(a)}{=} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \times \\
&\quad \times \Pr\{S_k = s, y_{k-1}, \mathbf{y}_k^{N-1}|S_{k-1} = s', \mathbf{y}_0^{k-2}\} \\
&\overset{(b)}{=} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \times \\
&\quad \times \Pr\{S_k = s, y_{k-1}|S_{k-1} = s', \mathbf{y}_0^{k-2}\} \times \\
&\quad \times \Pr\{\mathbf{y}_k^{N-1}|S_{k-1} = s', S_k = s, y_{k-1}, \mathbf{y}_0^{k-2}\} \\
&\overset{(c)}{=} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \times \\
&\quad \times \Pr\{S_k = s, y_{k-1}|S_{k-1} = s'\} \times \\
&\quad \times \Pr\{\mathbf{y}_k^{N-1}|S_k = s\}.
\end{aligned} \tag{2.6}$$

Os passos (a) e (b) são obtidos pela aplicação da regra de Bayes. O passo (c) é o resultado da observação de que os códigos convolucionais geram um processo de Markov e, portanto, o conhecimento do estado $s_{k-1}$ remove a influência da seqüência $\mathbf{y}_0^{k-2}$, recebida antes do "tempo" $k-1$, sobre o estado $s_k$ e sobre $y_{k-1}$. Além disto, o conhecimento do estado $s_k$ remove a influência de $s_{k-1}$, $y_{k-1}$ e $\mathbf{y}_0^{k-2}$ sobre $\mathbf{y}_k^{N-1}$. A figura 2.2 mostra como os índices são utilizados na treliça. Dividindo a equação 2.6 por

$$\Pr\{\mathbf{y}\} = \Pr\{\mathbf{y}_0^{k-2}\}\Pr\{y_{k-1}|\mathbf{y}_0^{k-2}\}\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}, \tag{2.7}$$

resulta em

$$\begin{aligned}\Pr\{S_{k-1}=s', S_k=s|\mathbf{y}\} &= \frac{\Pr\{S_{k-1}=s', \mathbf{y}_0^{k-2}\}}{\Pr\{\mathbf{y}_0^{k-2}\}} \times \\ &\times \frac{\Pr\{S_k=s, y_{k-1}|S_{k-1}=s'\}}{\Pr\{y_{k-1}|\mathbf{y}_0^{k-2}\}} \times \\ &\times \frac{\Pr\{\mathbf{y}_k^{N-1}|S_k=s\}}{\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}}. \end{aligned} \tag{2.8}$$

Defina

$$\alpha_k(s) = \frac{\Pr\{S_k=s, \mathbf{y}_0^{k-1}\}}{\Pr\{\mathbf{y}_0^{k-1}\}}, \tag{2.9a}$$

$$\gamma_k(s',s) = \Pr\{S_k=s, y_{k-1}|S_{k-1}=s'\} \text{ e} \tag{2.9b}$$

$$\beta_k(s) = \frac{\Pr\{\mathbf{y}_k^{N-1}|S_k=s\}}{\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}}. \tag{2.9c}$$

Substituindo-se estas expressões na equação 2.8, obtém-se

$$\Pr\{S_{k-1}=s', S_k=s|\mathbf{y}\} = \frac{1}{\Pr\{y_{k-1}|\mathbf{y}_0^{k-2}\}}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s). \tag{2.10}$$

Substituindo esta equação em 2.4, resulta em

$$L_{k-1}(1) = \ln\left[\frac{\sum_{(s',s)\in\mathcal{B}_{k-1}^1}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}{\sum_{(s',s)\in\mathcal{B}_{k-1}^0}\alpha_{k-1}(s')\gamma_k(s',s)\beta_k(s)}\right]. \tag{2.11}$$

Os coeficientes $\alpha_k(s)$ e $\beta_k(s)$ podem ser calculados recursivamente notando-se

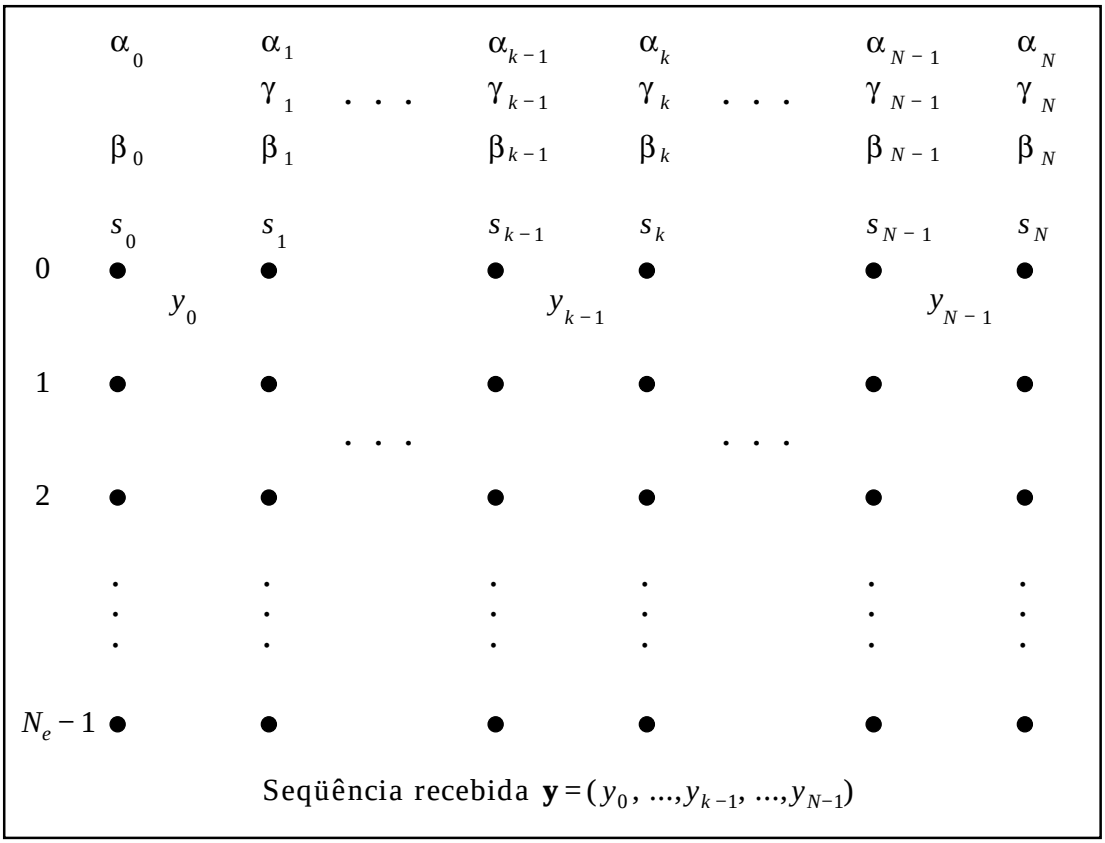


Figura 2.2: O algoritmo BCJR aplicado a uma treliça com $N_e$ estados.

que

$$
\begin{aligned}
\alpha_k(s) &= \frac{\Pr\{S_k = s, \mathbf{y}_0^{k-1}\}}{\Pr\{\mathbf{y}_0^{k-1}\}} \\
&= \frac{\sum_{s'} \Pr\{S_{k-1} = s', S_k = s, \mathbf{y}_0^{k-1}\}}{\sum_{s}\sum_{s'} \Pr\{S_{k-1} = s', S_k = s, \mathbf{y}_0^{k-1}\}} \\
&= \frac{\sum_{s'} \Pr\{S_{k-1} = s', S_k = s, \mathbf{y}_0^{k-2}, y_{k-1}\}}{\sum_{s}\sum_{s'} \Pr\{S_{k-1} = s', S_k = s, \mathbf{y}_0^{k-2}, y_{k-1}\}} \\
&= \frac{\sum_{s'} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \Pr\{S_k = s, y_{k-1} | S_{k-1} = s', \mathbf{y}_0^{k-2}\}}{\sum_{s}\sum_{s'} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \Pr\{S_k = s, y_{k-1} | S_{k-1} = s', \mathbf{y}_0^{k-2}\}} \\
&\overset{(a)}{=} \frac{\sum_{s'} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \Pr\{S_k = s, y_{k-1} | S_{k-1} = s'\}}{\sum_{s}\sum_{s'} \Pr\{S_{k-1} = s', \mathbf{y}_0^{k-2}\} \Pr\{S_k = s, y_{k-1} | S_{k-1} = s'\}} \frac{\Pr\{\mathbf{y}_0^{k-2}\}}{\Pr\{\mathbf{y}_0^{k-2}\}} \\
&= \frac{\sum_{s'} \alpha_{k-1}(s')\gamma_k(s', s)}{\sum_{s}\sum_{s'} \alpha_{k-1}(s')\gamma_k(s', s)},
\end{aligned}
\tag{2.12}
$$

onde a razão do passo (a) é a mesma da do passo (c) na equação 2.6, e que

$$
\begin{aligned}
\beta_{k-1}(s') &= \frac{\Pr\{\mathbf{y}_{k-1}^{N-1}|S_{k-1}=s'\}}{\Pr\{\mathbf{y}_{k-1}^{N-1}|\mathbf{y}_0^{k-2}\}} \\
&= \frac{\sum_s \Pr\{S_k=s,\mathbf{y}_{k-1}^{N-1}|S_{k-1}=s'\}}{\Pr\{\mathbf{y}_{k-1}^{N-1}|\mathbf{y}_0^{k-2}\}}\frac{\Pr\{\mathbf{y}_0^{k-2}\}}{\Pr\{\mathbf{y}_0^{k-2}\}} \\
&= \frac{\sum_s \Pr\{S_k=s,y_{k-1},\mathbf{y}_k^{N-1}|S_{k-1}=s'\}}{\Pr\{\mathbf{y}_0^{k-1}\}}\frac{\Pr\{\mathbf{y}_0^{k-2}\}}{\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}} \\
&= \frac{\sum_s \Pr\{S_k=s,y_{k-1}|S_{k-1}=s'\}\frac{\Pr\{\mathbf{y}_k^{N-1}|S_{k-1}=s',S_k=s,y_{k-1}\}}{\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}}}{\sum_s\sum_{s'}\frac{\Pr\{S_{k-1}=s',S_k=s,\mathbf{y}_0^{k-2},y_{k-1}\}}{\Pr\{\mathbf{y}_0^{k-2}\}}} \\
&\overset{(a)}{=} \frac{\sum_s \Pr\{S_k=s,y_{k-1}|S_{k-1}=s'\}\frac{\Pr\{\mathbf{y}_k^{N-1}|S_k=s\}}{\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}}}{\sum_s\sum_{s'}\frac{\Pr\{S_{k-1}=s',\mathbf{y}_0^{k-2}\}}{\Pr\{\mathbf{y}_0^{k-2}\}}\Pr\{S_k=s,y_{k-1}|S_{k-1}=s',\mathbf{y}_0^{k-2}\}} \\
&\overset{(b)}{=} \frac{\sum_s \Pr\{S_k=s,y_{k-1}|S_{k-1}=s'\}\frac{\Pr\{\mathbf{y}_k^{N-1}|S_k=s\}}{\Pr\{\mathbf{y}_k^{N-1}|\mathbf{y}_0^{k-1}\}}}{\sum_s\sum_{s'}\frac{\Pr\{S_{k-1}=s',\mathbf{y}_0^{k-2}\}}{\Pr\{\mathbf{y}_0^{k-2}\}}\Pr\{S_k=s,y_{k-1}|S_{k-1}=s'\}} \\
&= \frac{\sum_s \gamma_k(s',s)\beta_k(s)}{\sum_s\sum_{s'}\alpha_{k-1}(s')\gamma_k(s',s)},
\end{aligned}
\tag{2.13}
$$

onde em (a) o conhecimento do estado $s_k$ remove a influência dos estados e da seqüência anteriores ao índice $k$ sobre a seqüência futura $\mathbf{y}_k^{N-1}$ e em (b) a razão é a mesma do passo (c) na equação 2.6, para $k = 1, 2, \ldots, N$. Utilizando um procedimento semelhante para obter a expressão de $\alpha_1(s)$, pode-se mostrar que $\alpha_0(s) = \Pr\{S_0 = s\}$ e como, em geral, o codificador convolucional é colocado no estado $s_0 = 0$ no início da codificação de cada seqüência de entrada, tem-se que $\alpha_0(s) = 1$ para $s = 0$ e $\alpha_0(s) = 0$ para $s \neq 0$. Se ao final da codificação de cada seqüência de entrada o codificador é forçado a retornar ao estado zero ($s_N = 0$), adota-se $\beta_N(s) = 1$ para $s = 0$ e $\beta_N(s) = 0$ para $s \neq 0$, caso contrário, adota-se $\beta_N(s) = 1/N_e$, $\forall s \in 0, 1, \ldots, N_e - 1$.

Resta, então, mostrar como calcular $\gamma_k(s', s)$. Assume-se para efeito de exemplificação que o canal seja afetado por um ruído com distribuição gaussiana. Assume-se também que a transmissão é feita utilizando-se BPSK (*Binary Phase Shift Keying*) de forma que o *bit* 0 é mapeado no símbolo de canal $-1$ e o *bit* 1 é mapeado no símbolo de canal $+1$. Note que, para cada evento $u_{k-1}$, há um evento

$x_{k-1} = (x_{k-1}^S, x_{k-1}^P)$ correspondente e que $y_{k-1} = (y_{k-1}^S, y_{k-1}^P)$ é o sinal $x_{k-1}$ corrompido pelo ruído. Note, também, que $Y_{k-1}^S$ e $Y_{k-1}^P$ são variáveis aleatórias gaussianas não correlacionadas cuja média depende do valor transmitido de $x_{k-1}^S$ e $x_{k-1}^P$, respectivamente. Isto posto, defina

$$L_{k-1}^a(1) = \ln\left[\frac{\Pr\{U_{k-1}=1\}}{\Pr\{U_{k-1}=0\}}\right] = \ln\left[\frac{\Pr\{X_{k-1}^S=+1\}}{\Pr\{X_{k-1}^S=-1\}}\right]. \tag{2.14}$$

Assim sendo, após alguma manipulação, tem-se que

$$\Pr\{U_{k-1}=1\} = \Pr\{X_{k-1}^S=+1\} = \frac{e^{L_{k-1}^a(1)}}{1+e^{L_{k-1}^a(1)}} \tag{2.15}$$

e

$$\Pr\{U_{k-1}=0\} = \Pr\{X_{k-1}^S=-1\} = \frac{1}{1+e^{L_{k-1}^a(1)}}. \tag{2.16}$$

As equações 2.15 e 2.16 podem ser escritas, mais sinteticamente, como

$$\begin{aligned}\Pr\{X_{k-1}^S = x_{k-1}^S\} &= \frac{e^{L_{k-1}^a(1)/2}}{1+e^{L_{k-1}^a(1)}} e^{x_{k-1}^S L_{k-1}^a(1)/2} \\ &= A_{k-1} e^{x_{k-1}^S L_{k-1}^a(1)/2}.\end{aligned} \tag{2.17}$$

Desta forma, partindo-se da definição de $\gamma_k(s',s)$, se $(s',s) \in \mathcal{B}_{k-1}^1$ então

$$\begin{aligned}\gamma_k(s',s) &= \Pr\{S_k=s, y_{k-1}|S_{k-1}=s'\} \\ &= \Pr\{S_k=s|S_{k-1}=s'\}\Pr\{y_{k-1}|S_{k-1}=s', S_k=s\} \\ &= \Pr\{X_{k-1}^S=+1\}\Pr\{y_{k-1}|X_{k-1}=x_{k-1}\} \\ &= \Pr\{X_{k-1}^S=+1\}\times \\ &\quad \times\Pr\{y_{k-1}^S|X_{k-1}^S=+1\}\times \\ &\quad \times\Pr\{y_{k-1}^P|X_{k-1}^P=x_{k-1}^P(s',s)\}.\end{aligned} \tag{2.18}$$

Visto que

$$\Pr\{y_{k-1}^S|X_{k-1}^S=+1\} \propto e^{-\frac{\left(y_{k-1}^S-1\right)^2}{2\sigma^2}} \tag{2.19}$$

e que

$$\Pr\{y_{k-1}^P|X_{k-1}^P=x_{k-1}^P(s',s)\} \propto e^{-\frac{\left(y_{k-1}^P-x_{k-1}^P(s',s)\right)^2}{2\sigma^2}} \tag{2.20}$$

então a equação 2.18 pode se reescrita como

$$\begin{aligned}\gamma_k(s',s) &= \underbrace{B_{k-1}e^{-\frac{\left(y_{k-1}^S\right)^2+1}{2\sigma^2}}e^{-\frac{\left(y_{k-1}^P\right)^2+\left(x_{k-1}^P(s',s)\right)^2}{2\sigma^2}}}_{C_{k-1}}e^{\frac{1}{2}L_{k-1}^a(1)}e^{\frac{y_{k-1}^S}{\sigma^2}}e^{\frac{y_{k-1}^P x_{k-1}^P(s',s)}{\sigma^2}} \\ &= C_{k-1}e^{\frac{1}{2}L_{k-1}^a(1)}e^{\frac{y_{k-1}^S}{\sigma^2}}e^{\frac{y_{k-1}^P x_{k-1}^P(s',s)}{\sigma^2}}. \qquad (2.21)\end{aligned}$$

Analogamente, se $(s',s) \in \mathcal{B}_{k-1}^0$, tem-se que

$$\begin{aligned}\gamma_k(s',s) &= \Pr\{X_{k-1}^S = -1\} \times \\ &\quad \times \Pr\{y_{k-1}^S | X_{k-1}^S = -1\} \times \\ &\quad \times \Pr\{y_{k-1}^P | X_{k-1}^P = x_{k-1}^P(s',s)\} \qquad (2.22)\end{aligned}$$

e que

$$\gamma_k(s',s) = C_{k-1}e^{-\frac{1}{2}L_{k-1}^a(1)}e^{-\frac{y_{k-1}^S}{\sigma^2}}e^{\frac{y_{k-1}^P x_{k-1}^P(s',s)}{\sigma^2}}. \qquad (2.23)$$

Então, definindo gama extrínseco como

$$\gamma_k^e(s',s) = e^{\frac{y_{k-1}^P x_{k-1}^P(s',s)}{\sigma^2}} \qquad (2.24)$$

e substituindo as equações 2.21 e 2.23 adequadamente na equação 2.11 obtém-se, finalmente,

$$L_{k-1}(1) = L_{k-1}^a(1) + \frac{2}{\sigma^2}y_{k-1}^S + \ln\left[\frac{\sum_{(s',s)\in\mathcal{B}_{k-1}^1}\alpha_{k-1}(s')\gamma_k^e(s',s)\beta_k(s)}{\sum_{(s',s)\in\mathcal{B}_{k-1}^0}\alpha_{k-1}(s')\gamma_k^e(s',s)\beta_k(s)}\right]. \qquad (2.25)$$

O valor $2/\sigma^2$, que multiplica $y_{k-1}^S$, é por vezes chamado de valor de canal e designado por $L_c$. Da equação 2.25, conclui-se que o logaritmo da razão de verossimilhança é igual à soma de um valor *a priori*, $L_{k-1}^a(1)$, de um valor de canal, $L_c y_{k-1}^S$, e da informação extrínseca, $L_{k-1}^e(1)$, que corresponde ao terceiro termo da equação 2.25. A informação extrínseca é a informação realmente nova gerada a respeito da mensagem transmitida $\mathbf{x}$ (equivalentemente $\mathbf{u}$) a partir da mensagem recebida $\mathbf{y}$. Definindo $\mathbf{L} = [L_0(1)L_1(1)\ldots L_{N-1}(1)]^T$, $\mathbf{L}^a = [L_0^a(1)L_1^a(1)\ldots L_{N-1}^a(1)]^T$ e $\mathbf{L}^e = [L_0^e(1)L_1^e(1)\ldots L_{N-1}^e(1)]^T$, a equação 2.25 pode ser expressa na forma vetorial como

$$\mathbf{L} = \mathbf{L}^a + L_c\mathbf{y}^S + \mathbf{L}^e. \qquad (2.26)$$

## 2.4 O Turbo Decodificador

O turbo decodificador é constituído por dois decodificadores BCJR-MAP. Eles recebem a informação sistemática, $\mathbf{y}^S$, e de paridade, $\mathbf{y}^P$, provavelmente corrompidas por ruído, e produzem estimativas contínuas da mensagem transmitida $\mathbf{u}$, ou seja, $\mathbf{L}_1$ e $\mathbf{P}^{-1}\mathbf{L}_2$, conforme mostrado na figura 2.3. A partir destas estimativas obtém-se a informação extrínseca, $\mathbf{L}_{12}^e$ e $\mathbf{L}_{21}^e$, que é reinjetada nos decodificadores como informação *a priori*, $\mathbf{L}_1^a = \mathbf{P}^{-1}\mathbf{L}_{21}^e$ e $\mathbf{L}_2^a = \mathbf{P}\mathbf{L}_{12}^e$, como indicado nesta mesma figura, para produzir novas e melhores estimativas. Isto é feito iterativamente, com um número de iterações oscilando normalmente entre 10 e 20. Em geral, o processo iterativo pára quando o número máximo de iterações é atingido. Matematicamente, dado o número de iterações $N_{it}$, tem-se que

$$\mathbf{L}_{12}^{e(i)} = \mathbf{L}_1^{(i)} - \mathbf{P}^{-1}\mathbf{L}_{21}^{e(i-1)} - L_c\mathbf{y}^S \tag{2.27}$$

e

$$\mathbf{L}_{21}^{e(i)} = \mathbf{L}_2^{(i)} - \mathbf{P}\mathbf{L}_{12}^{e(i)} - L_c\mathbf{P}\mathbf{y}^S \tag{2.28}$$

para cada iteração $i = 1, 2, \ldots, N_{it}$, sendo que $\mathbf{L}_{21}^{e(0)} = \mathbf{0}$. Substituindo-se a equação 2.27 em 2.28, obtém-se

$$\mathbf{L}_{21}^{e(i)} = \mathbf{L}_2^{(i)} - \mathbf{P}\mathbf{L}_1^{(i)} + \mathbf{L}_{21}^{e(i-1)} \tag{2.29}$$

e, portanto,

$$\mathbf{L}_{21}^{e(i)} = \Delta\mathbf{L}_{21}^{e(i)} + \mathbf{L}_{21}^{e(i-1)} \tag{2.30}$$

A mensagem final é obtida através do mapeamento $\hat{u}_{k-1} = h([\mathbf{P}^{-1}\mathbf{L}_2^{(N_{it})}]_{k-1})$, onde

$$h(x) = \begin{cases} 1, x > 0 \\ 0, x < 0. \end{cases} \tag{2.31}$$

As equações 2.29 e 2.30 mostram que a nova informação extrínseca obtida pelo decodificador 2 é a diferença entre as estimativas correntes do decodificador 2 e do decodificador 1 somada à informação extrínseca anterior gerada pelo decodificador 2, de modo que $\Delta\mathbf{L}_{21}^{e(i)} = \mathbf{L}_2^{(i)} - \mathbf{P}\mathbf{L}_1^{(i)} = \mathbf{0}$ implica que a convergência foi atingida na $i$-ésima iteração.

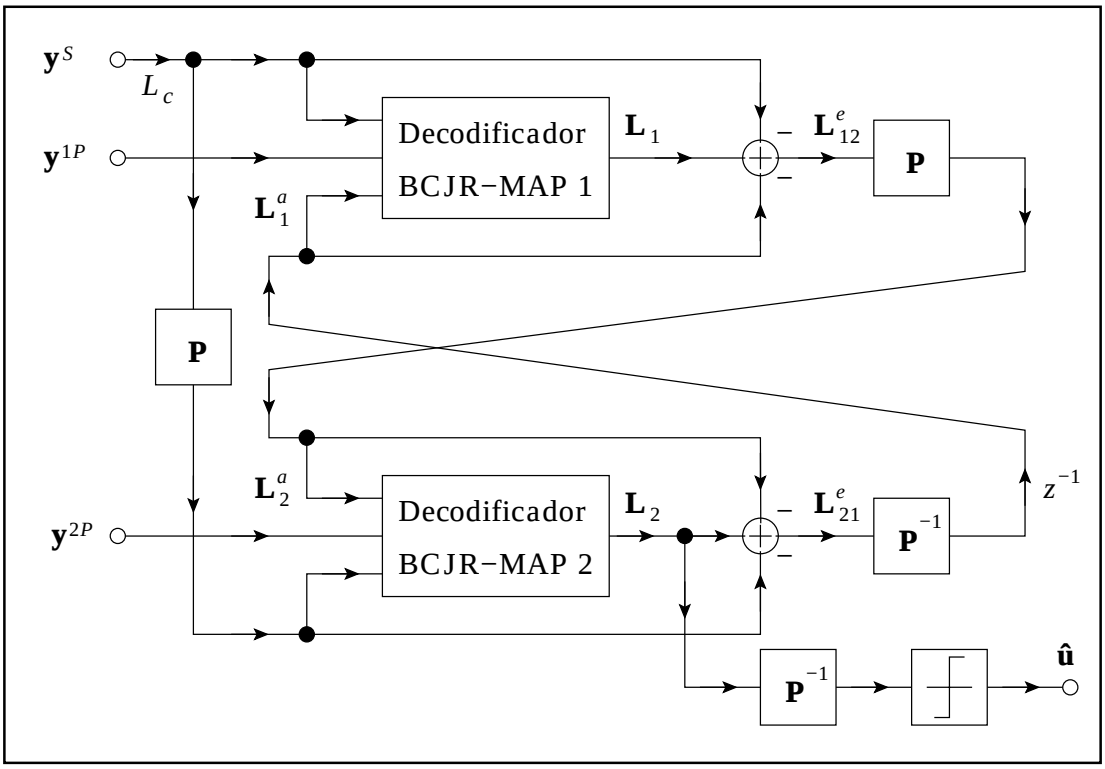


Figura 2.3: O turbo decodificador.

**Definição 2.1.** Seja $\mathbf{y}$ o vetor corrompido por ruído a ser processado pelo turbo decodificador. Um par de matrizes $(\mathbf{L}_1, \mathbf{L}_2)$ para o qual $\mathbf{L}_2 - \mathbf{P}\mathbf{L}_1 = \mathbf{0}$ é chamado de *ponto fixo do código turbo* para $\mathbf{y}$.

Na prática, devido a erros de natureza numérica inevitáveis que ocorrem quando o ruído de canal é relativamente baixo, se utiliza o algoritmo BCJR-LogMAP, que trabalha com os valores $\tilde{\alpha}_k(s) = \ln(\alpha_k(s))$, $\tilde{\gamma}_k(s', s) = \ln(\gamma_k(s', s))$ e $\tilde{\beta}_k(s) = \ln(\beta_k(s))$. Desta forma, as equações 2.11, 2.12 e 2.13 são substituídas por

$$L_{k-1}(1) = \ln\left[\frac{\sum\limits_{(s',s)\in\mathcal{B}_{k-1}^1} e^{\tilde{\alpha}_{k-1}(s')+\tilde{\gamma}_k(s',s)+\tilde{\beta}_k(s)}}{\sum\limits_{(s',s)\in\mathcal{B}_{k-1}^0} e^{\tilde{\alpha}_{k-1}(s')+\tilde{\gamma}_k(s',s)+\tilde{\beta}_k(s)}}\right], \tag{2.32}$$

$$\tilde{\alpha}_k(s) = \ln\left[\frac{\sum\limits_{s'} e^{\tilde{\alpha}_{k-1}(s')+\tilde{\gamma}_k(s',s)}}{\sum\limits_{s}\sum\limits_{s'} e^{\tilde{\alpha}_{k-1}(s')+\tilde{\gamma}_k(s',s)}}\right] \tag{2.33}$$

e

$$\tilde{\beta}_{k-1}(s') = \ln\left[\frac{\sum\limits_{s} e^{\tilde{\gamma}_k(s',s)+\tilde{\beta}_k(s)}}{\sum\limits_{s}\sum\limits_{s'} e^{\tilde{\alpha}_{k-1}(s')+\tilde{\gamma}_k(s',s)}}\right], \tag{2.34}$$

respectivamente. Utilizando o fato de $\ln(e^a + e^b) = \max\{a, b\} + \ln(1 + e^{-|a-b|})$ pode-se calcular exata e recursivamente a expressão $\ln(e^{b_0} + e^{b_1} + \ldots + e^{b_n})$, como mostrado na seção B.1 do apêndice B. E, tabelando a função $\ln(1 + e^{-|x|})$, evita-se completamente a avaliação de exponenciais e logaritmos [10].

O algoritmo BCJR-LogMAP pode ser simplificado desprezando-se o termo $\ln(1 + e^{-|a-b|})$ da equação $\ln(e^a + e^b) = \max\{a, b\} + \ln(1 + e^{-|a-b|})$ de tal forma que $\ln(e^{b_0} + e^{b_1} + \ldots + e^{b_n}) \simeq \max\{b_0, b_1, \ldots, b_n\}$. Utilizando esta aproximação,

$$\tilde{\alpha}_k(s) = \frac{\max\{\tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) : s' = 0, 1 \ldots, N_e - 1\}}{\max\{\tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) : s, s' = 0, 1 \ldots, N_e - 1\}} \tag{2.35}$$

e

$$\tilde{\beta}_{k-1}(s') = \frac{\max\{\tilde{\gamma}_k(s', s) + \tilde{\beta}_k(s) : s' = 0, 1 \ldots, N_e - 1\}}{\max\{\tilde{\alpha}_{k-1}(s') + \tilde{\gamma}_k(s', s) : s, s' = 0, 1 \ldots, N_e - 1\}}. \tag{2.36}$$

Este algoritmo se chama BCJR-MaxLogMAP. As operações que envolvem agora o cálculo de $\tilde{\alpha}_k(s)$ e $\tilde{\beta}_{k-1}(s')$ são as mesmas do algoritmo de Viterbi (adição, comparação e seleção) [10, 15]. Portanto, o algoritmo BCJR-MaxLogMAP executa duas recursões do algoritmo de Viterbi, uma para frente e outra para trás [10]. O algoritmo BCJR-MaxLogMAP tem desempenho idêntico ao do algoritmo de Viterbi [10] em termos de taxa de erro de *bits*.

Algoritmos de decodificação baseados em treliças como o BCJR são, como foi visto na seção anterior, métodos recursivos para a estimação da seqüência de estados de um processo de Markov discreto no tempo com um número finito de estados [10]. O algoritmo de Viterbi (HOVA – *Hard Ouput Viterbi Algorithm*) é também um decodificador baseado em treliças. Ele é do tipo SIHO (*Soft Input – Hard Output*) com uma recursão para frente, envolvendo decisões contínuas (*soft decisions*). A recursão termina com uma decisão de limiar (*hard decision*) através da qual um caminho particular da treliça é selecionado dentre muitos outros. Este algoritmo é um estimador de seqüências de máxima verossimilhança, pois maximiza a função de verossimilhança para a seqüência inteira e não para cada *bit* ou símbolo. Em sistemas concatenados, com mútiplos estágios de processamento de sinal, o desempenho geral do decodificador (receptor) é melhorado se os estágios mais internos produzem estimativas de saída contínuas (*soft output*) [10]. O algoritmo de Viterbi pode ser modificado para gerar este tipo de informação de saída. Esta modificação é conhecida como SOVA (*Soft Output Viterbi Algorithm* [10]). O BCJR-MaxLogMAP

pode ser considerado como uma implementação do SOVA. Além das estimativas dos símbolos de máxima verossimilhança, o SOVA produz uma medida de confiança para cada símbolo recebido. Porém, tais estimativas são subótimas [10]. Sendo SISO (*Soft Input – Soft Output*), o SOVA pode ser utilizado também na decodificação dos códigos turbo.

O algoritmo BCJR-MAP também é do tipo SISO e executa duas recursões na treliça do código convolucional, sendo uma para frente e outra para trás, ambas envolvendo decisões contínuas (*soft decisions*). Este algoritmo é um decodificador MAP que minimiza a probabilidade de erro dos *bits* ou símbolos estimando probabilidades *a posteriori* de cada *bit* ou símbolo em uma seqüência recebida. Para reconstruir a seqüência original, as saídas contínuas produzidas pelo algoritmo BCJR passam por uma decisão de limiar (*hard decision*). Este algoritmo gera estimativas de confiança ótimas na forma de probabildades *a posteriori* dos *bits* ou símbolos. A recursão para trás do algoritmo BCJR é um fator responsável pela sua maior complexidade em relação ao algoritmo de Viterbi (HOVA). Entretanto, a taxa de erro de *bits* ou símbolos do BCJR é sempre menor ou igual à do HOVA.

HEEGARD *et al.* [2] mostram que SOVA e BCJR podem ser considerados como casos particulares do algoritmo generalizado de Viterbi (GVA – *Generalized Viterbi Algorithm*). Usando o GVA, mostram que a simples substituição dos somatórios nas equações 2.11, 2.12 e 2.13 pela operação $\max\{\cdot\}$ "transforma" o algoritmo BCJR no SOVA.

## 2.5 Códigos Turbo Múltiplos

Como foi exposto nas seções anteriores, a turbo codificação é feita por meio da concatenação paralela de dois codificadores convolucionais e um permutador. A decodificação, por sua vez, é feita por meio de dois decodificadores BCJR-MAP ou SOVA que operam em modo serial, ou seja, cada um deles é ativado seqüencialmente, de modo que a informação extrínseca produzida por um é utilizada como informação *a priori* pelo outro num processo iterativo.

Os códigos considerados nesta seção consistem da concatenação paralela de três ou mais codificadores convolucionais, precedidos por permutadores pseudo-

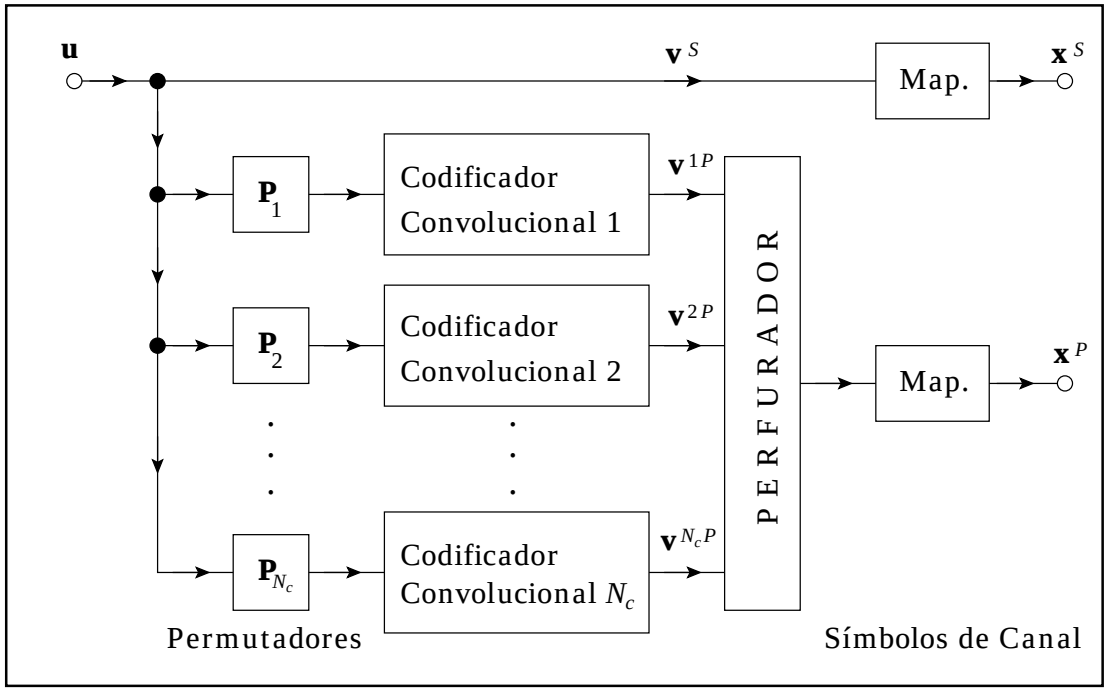


Figura 2.4: O codificador de um código turbo de multiplicidade $N_c$.

aleatórios, como é descrito por DIVSALAR *et al.* [13, 16, 17, 18]. Estes códigos são denominados códigos turbo múltiplos ou MTC (*Multiple Turbo Codes*).

Na figura 2.4, é mostrado o diagrama de blocos de um codificador turbo de multiplicidade $N_c$. $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{N_c}$ são permutadores distintos e $\mathbf{P}_1$ é, normalmente, o permutador identidade, ou seja, nenhuma permutação é feita. Em geral, os códigos componentes não precisam ser iguais nem possuir a mesma taxa. No caso ilustrado na figura, tem-se um MTC de taxa $1/(N_c + 1)$. Várias outras taxas podem ser obtidas pela conveniente perfuração dos *bits* de $\mathbf{v}^{1P}, \mathbf{v}^{2P}, \ldots, \mathbf{v}^{N_cP}$ e até mesmo de $\mathbf{v}^S$ se o decodificador permanecer funcional [13]. Além disto, um MTC é capaz de reduzir a probabilidade de certas seqüências críticas de entrada serem permutadas numa seqüência da mesma forma por todos os permutadores. Quando este fato ocorre a eficiência de um dado código turbo é prejudicada [13].

Como os MTC são constituídos por três ou mais codificadores convolucionais, a decodificação utilizando os algoritmos BCJR-MAP ou SOVA pode ser feita de vários modos. Alguns destes modos podem ser vistos na figura 2.5 para o caso $N_c = 3$. A figura 2.5a corresponde à decodificação serial usual dos códigos turbo: primeiro o decodificador número 1 é ativado, depois o número 2, em seguida o número 3 e assim sucessivamente. A figura 2.5b corresponde a um esquema misto de decodifi-

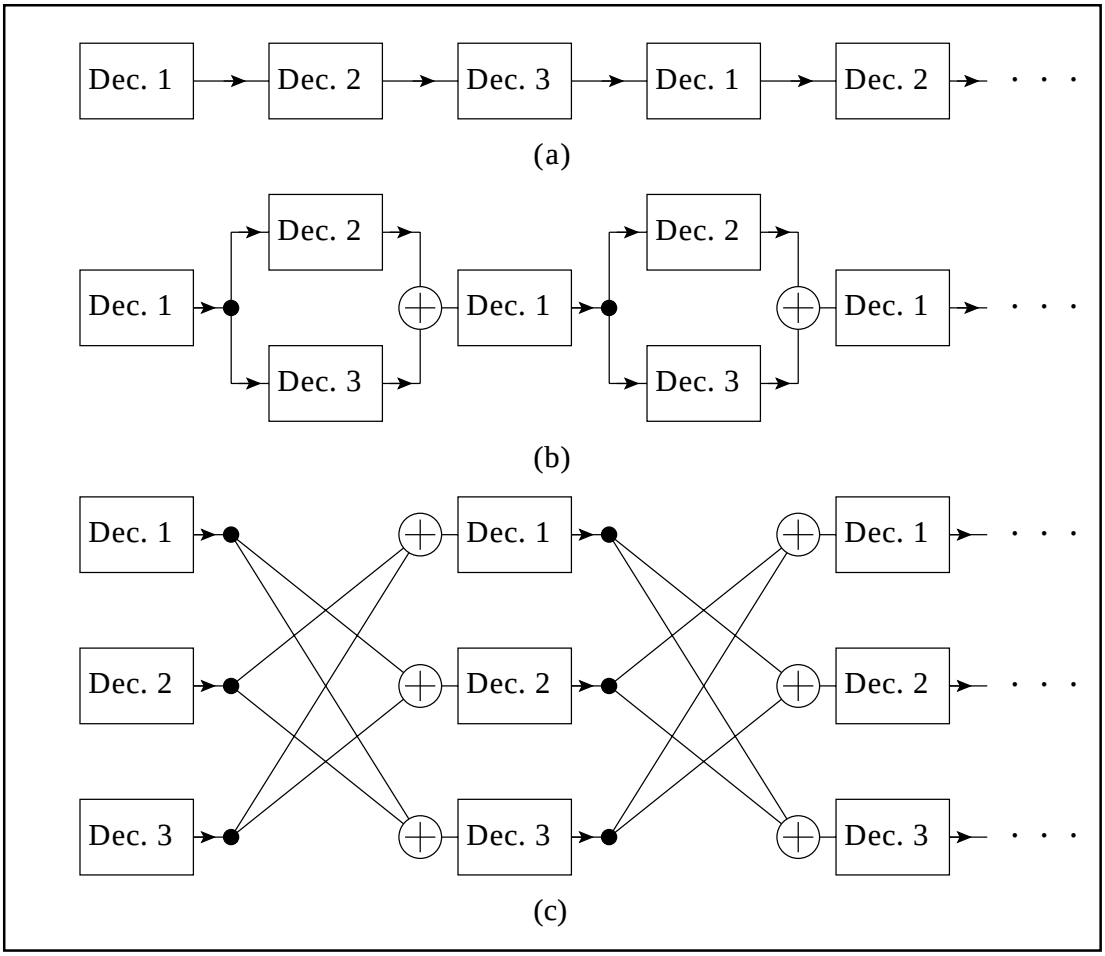


Figura 2.5: Algumas estruturas para a decodificação de códigos turbo de multiplicidade $N_c = 3$. O modo serial é mostrado em (a), o modo mestre-escravo, em (b) e o modo paralelo, em (c).

cação serial e paralela chamado mestre-escravo [13]. Por fim, na figura 2.5c, têm-se todos os decodificadores operando em paralelo, sendo esta última configuração a que produz os melhores resultados para os MTC [13] de taxa $1/(N_c + 1)$. Como esta estrutura é paralela, todos os decodificadores podem processar informação simultaneamente, o que é uma vantagem sobre a decodificação serial ordinária em termos de tempo de decodificação. A figura 2.6 mostra um decodificador completo para $N_c = 3$, utilizando realimentação, que é equivalente ao da figura 2.5c.

Os códigos turbo para $N_c = 2$ também podem ser decodificados utilizando o modo paralelo, mas o desempenho é similar ao do modo serial [13]. Na verdade, a forma paralela para $N_c = 2$ é uma composição de duas formas seriais disjuntas como é mostrado na figura 2.7.

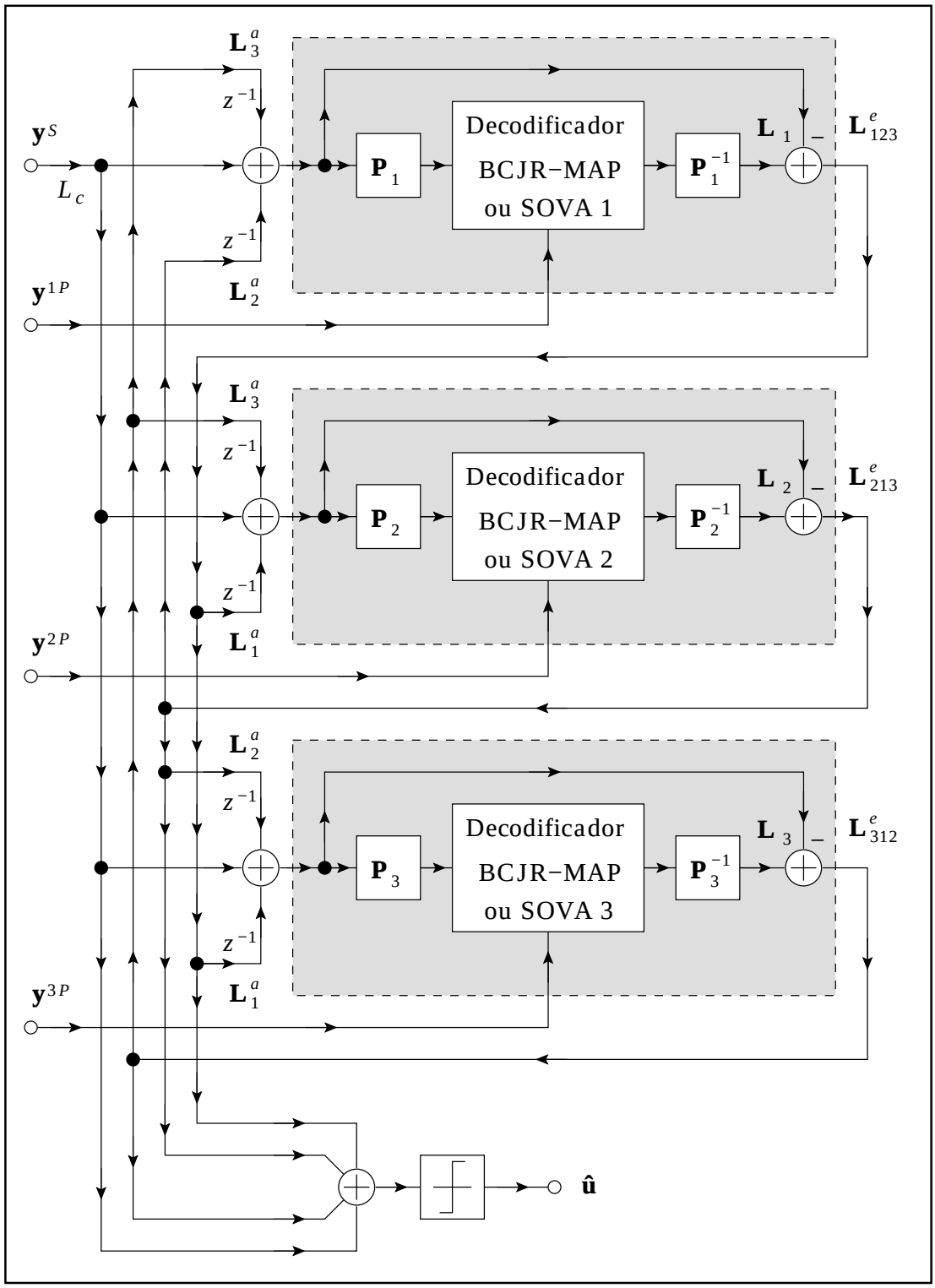


Figura 2.6: Modo paralelo de decodificação para códigos turbo de multiplicidade $N_c = 3$.

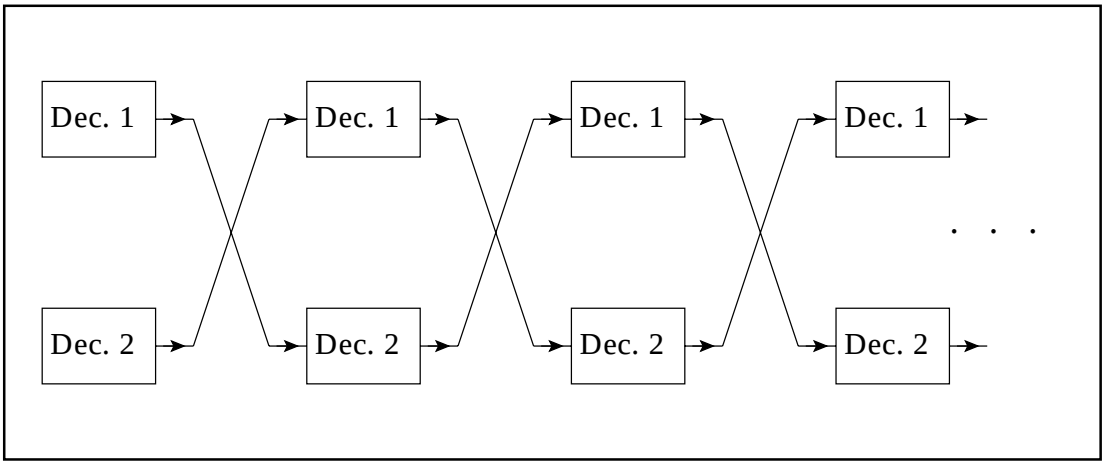


Figura 2.7: Modo paralelo de decodificação para códigos turbo de multiplicidade $N_c = 2$.

# Capítulo 3

# Modulação e Quantização Codificadas por Treliças

Um bom codificador de canal gera seqüências de símbolos de canal que estão afastadas de tal forma no espaço que permitem ao decodificador escolher, após corrupção da informação pelo ruído, a seqüência que está mais próxima daquela recebida, minimizando a probabilidade de ocorrer um erro de decisão entre quaisquer duas delas. Um bom quantizador gera seqüências afastadas de tal forma que a distorção média entre a seqüência original e a quantizada seja mínima. Neste capítulo, mostra-se, utilizando os exemplos da modulação e da quantização codificadas por treliças, que um bom codificador de canal pode ser utilizado como um bom quantizador, explorando a dualidade entre a teoria da codificação de canal e a teoria taxa $\times$ distorção.

## 3.1 Modulação Codificada por Treliças

A modulação codificada por treliças, ou TCM, é uma técnica que combina codificação e modulação para transmissão digital num canal de banda limitada [19]. A principal vantagem deste tipo de modulação é obter ganhos de codificação significativos em relação aos esquemas de modulação multinível não codificados sem a necessidade de se expandir a banda de transmissão.

A modulação codificada por treliças se utiliza da modulação multinível associada a um codificador com um número finito de estados, cuja saída seleciona

os sinais de modulação, gerando seqüências de sinais já codificados. Os ganhos de codificação significativos são obtidos pela expansão do conjunto de sinais modulados, provendo redundância para o código, e pelo projeto do código e das funções de mapeamento dos sinais, maximizando diretamente a distância euclidiana mínima entre as seqüências do sinal codificado. Isto permite a construção de códigos modulados cuja distância livre excede, significativamente, a distância livre entre os sinais modulados de um esquema não codificado, mantendo-se a mesma taxa, largura de banda e potência de sinal [19].

### 3.1.1 O Codificador

Na figura 3.1a, tem-se o diagrama de blocos do codificador da TCM. Ele é composto por um codificador convolucional sitemático recursivo, seguido por um mapeador que seleciona o sinal modulado a ser transmitido pelo canal. O codificador convolucional produz um *bit* de paridade para cada $R$ *bits* de entrada, gerando localmente um código de taxa $R/(R+1)$. No caso particular em que se emprega um modulador ASK (*Amplitude Shift Keying*), somente um dos $R$ *bits* de cada símbolo do vetor $\mathbf{u} = [u_0 u_1 \dots u_{N-1}]^T$ será codificado. Ou seja, sendo $u_k = (u_k^{R-1}, \dots, u_k^1, u_k^0)$ um dos símbolos do vetor de entrada $\mathbf{u}$, somente o *bit* $u_k^0$ será codificado. Na figura 3.1b, é mostrado um diagrama deste tipo de codificador convolucional.

A relação de entrada e saída de um codificador convolucional pode ser representada por meio de matrizes polinômiais através do operador atraso $D$ (transformada $D$) [14]. Com esta notação, o codificador da figura 3.1b pode ser expresso matematicamente como

$$\begin{bmatrix} V^R(D) \\ V^{R-1}(D) \\ \vdots \\ V^1(D) \\ V^0(D) \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & \frac{g^1(D)}{g^0(D)} \end{bmatrix} \begin{bmatrix} U^{R-1}(D) \\ U^{R-2}(D) \\ \vdots \\ U^0(D) \end{bmatrix}, \tag{3.1}$$

onde o polinômio

$$g^0(D) = g_{N_0}^0 D^{N_0} + g_{N_0-1}^0 D^{N_0-1} + \cdots + g_1^0 D + g_0^0 \tag{3.2}$$

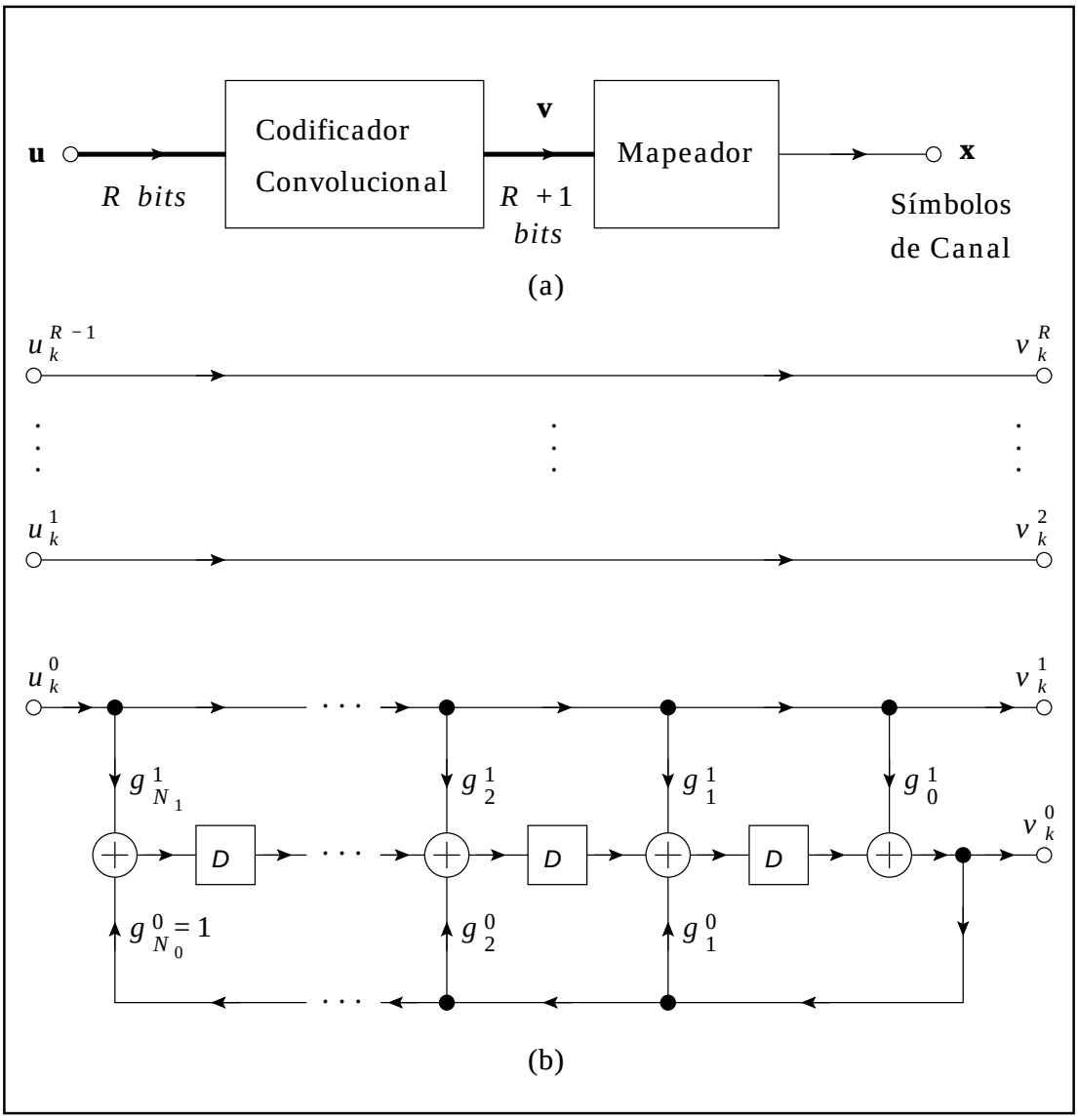


Figura 3.1: Codificação com TCM. Diagrama de blocos em (a). Codificador convolucional em (b).

corresponde à parte recursiva do código, sendo $g^0_{N_0} = g^0_0 = 1$, e o polinômio

$$g^1(D) = g^1_{N_1} D^{N_1} + g^1_{N_1-1} D^{N_1-1} + \cdots + g^1_1 D + g^1_0 \tag{3.3}$$

corresponde à parte não recursiva. Os coeficientes dos polinômios pertencem ao conjunto $\mathcal{G}_2 = \{0, 1\}$ e o número de estados do código é dado por $N_e = 2^{\max\{N_0, N_1\}}$. Estes polinômios são referenciados de forma mais sucinta agrupando seus coeficientes binários de três em três da direita para a esquerda e escrevendo o resultado em octal. Desta forma, supondo que $g^0(D) = D^4 + D + 1$ e que $g^1(D) = D^2$ este código será

referenciado como

$$\begin{aligned}\left(g^0(D), g^1(D)\right) &= (010\ 011_2, 100_2) \\ &= (23, 4)_8. \end{aligned} \tag{3.4}$$

Em geral, omite-se a base e escreve-se $(23, 4)$.

A partir do vetor de entrada $\mathbf{u}$, o codificador convolucional produz o vetor de índices $\mathbf{v} = [v_0 v_1 \dots v_{N-1}]^T$ com $R+1$ *bits* por símbolo, onde $v_k \in \mathcal{V}$ para $k = 0, 1, \dots, N-1$, sendo $\mathcal{V} = \{0, 1, \dots, M-1\}$ com $M = 2^{R+1}$. Este vetor é utilizado pelo mapeador para gerar o vetor $\mathbf{x} = [x_0 x_1 \dots x_{N-1}]^T$ de símbolos de canal, onde $x_k \in \mathcal{R}$ para $k = 0, 1, \dots, N-1$, sendo $\mathcal{R} = \{r_0, r_1, \dots, r_{M-1}\}$ um conjunto com $M$ números reais. De acordo com esta notação, $x_k = r_{v_k}$ para $k = 0, 1, \dots, N-1$. O conjunto $\mathcal{R}$ é particionado em quatro subconjuntos

$$\mathcal{D}_0 = \{r_{4j+0} : j = 0, 1, \dots, M/4-1\}, \tag{3.5a}$$

$$\mathcal{D}_1 = \{r_{4j+1} : j = 0, 1, \dots, M/4-1\}, \tag{3.5b}$$

$$\mathcal{D}_2 = \{r_{4j+2} : j = 0, 1, \dots, M/4-1\} \text{ e} \tag{3.5c}$$

$$\mathcal{D}_3 = \{r_{4j+3} : j = 0, 1, \dots, M/4-1\}, \tag{3.5d}$$

de modo a maximizar a distância mínima entre os elementos de $\mathcal{A}_0 = \mathcal{D}_0 \cup \mathcal{D}_2$ e entre os elementos de $\mathcal{A}_1 = \mathcal{D}_1 \cup \mathcal{D}_3$, como mostrado mais adiante. Dos $R+1$ *bits* de um símbolo $v_k = (v_k^R, v_k^{R-1}, \dots, v_k^1, v_k^0)$ do vetor $\mathbf{v}$, o grupo formado pelos *bits* $v_k^1 v_k^0$ é utilizado para selecionar os subconjuntos e o grupo formado pelos *bits* $v_k^R v_k^{R-1} \dots v_k^2$ seleciona um dos elementos do subconjunto.

Na figura 3.2, é mostrado um exemplo de codificação com o código $(7, 2)$ de 4 estados com eficiência espectral de 2 *bits*/s/Hz, ou seja, cada símbolo de canal representa 2 *bits* da mensagem $\mathbf{u}$. A figura 3.2a mostra o vetor de entrada $\mathbf{u}$ = (00, 01, 11, 10, 00, 11) sendo codificado como $\mathbf{v}$ = (00:0, 01:1, 11:0, 10:1, 00:1, 11:1), por meio do código convolucional da figura 3.2b, produzindo o vetor $\mathbf{x} = (r_{v_0=0}, r_{v_1=3}, r_{v_2=6}, r_{v_3=5}, r_{v_4=1}, r_{v_5=7})$ através do mapeamento mostrado na figura 3.2d. Em $\mathbf{v}$, o *bit* de paridade está à direita do sinal de dois pontos. Note que, para cada símbolo da mensagem $\mathbf{u}$ a ser codificado, só é permitido escolher entre os valores no conjunto $\mathcal{A}_0$ ou entre os valores no conjunto $\mathcal{A}_1$, conforme a treliça deste

código mostrada na figura 3.2c[1]. No caso deste exemplo, tem-se que

$$\mathcal{D}_0 = \left\{ r_0 = -\frac{7}{8}A, r_4 = +\frac{1}{8}A \right\}, \tag{3.6a}$$

$$\mathcal{D}_1 = \left\{ r_1 = -\frac{5}{8}A, r_5 = +\frac{3}{8}A \right\}, \tag{3.6b}$$

$$\mathcal{D}_2 = \left\{ r_2 = -\frac{3}{8}A, r_6 = +\frac{5}{8}A \right\} \text{e} \tag{3.6c}$$

$$\mathcal{D}_3 = \left\{ r_3 = -\frac{1}{8}A, r_7 = +\frac{7}{8}A \right\}, \tag{3.6d}$$

sendo $A$ é um fator de escala para ajustar a potência transmitida. Conseqüentemente, a distância mínima também foi maximizada entre elementos de $\mathcal{A}_0$ e entre elementos de $\mathcal{A}_1$.

### 3.1.2 O Decodificador

Os sinais modulados representados pelo vetor $\mathbf{x}$ na figura 3.2a são transmitidos por meio de um canal e, em geral, assume-se que sejam corrompidos por ruído branco gaussiano aditivo (AWGN – *Aditive White Gaussian Noise*) $\mathbf{e} = [e_0 e_1 \dots e_{N-1}]^T$, gerando um vetor $\mathbf{y} = [y_0 y_1 \dots y_{N-1}]^T$, sendo $\mathbf{y} = \mathbf{x} + \mathbf{e}$, que deve ser decodificado para se obter uma estimativa $\hat{\mathbf{u}} = [\hat{u}_0 \hat{u}_1 \dots \hat{u}_{N-1}]^T$ da mensagem transmitida $\mathbf{u}$. No caso da TCM, utiliza-se como algoritmo de decodificação o HOVA, conforme figura 3.3. Este algoritmo encontra a seqüência de símbolos de canal $\hat{\mathbf{x}} = [\hat{x}_0 \hat{x}_1 \dots \hat{x}_{N-1}]^T$[2] que está mais próxima, segundo a métrica euclidiana, da seqüência de canal corrompida por ruído $\mathbf{y}$, ou seja, a seqüência $\hat{\mathbf{x}}$ que minimiza

$$d_E(\mathbf{y}, \hat{\mathbf{x}}) = \sqrt{\sum_{k=0}^{N-1} (y_k - \hat{x}_k)^2}. \tag{3.7}$$

O HOVA retorna, então, o vetor $\hat{\mathbf{u}}$ correspondente ao vetor $\hat{\mathbf{x}}$ determinado.

## 3.2 Quantização e Teoria Taxa × Distorção

Visto que um número real, em geral, requer um número infinito de *bits* para ser representado, a utilização de um número finito de *bits* para representar uma variável aleatória, cujas amostras assumem valores arbitrários no conjunto dos números

[1] As transições paralelas causadas pelo *bit* $v_k^2$ não foram representadas.

[2] Note que $\hat{x}_k \in \mathcal{R}$ com $\hat{x}_k = r_{\hat{v}_k}$, para $k = 0, 1, \dots, N-1$.

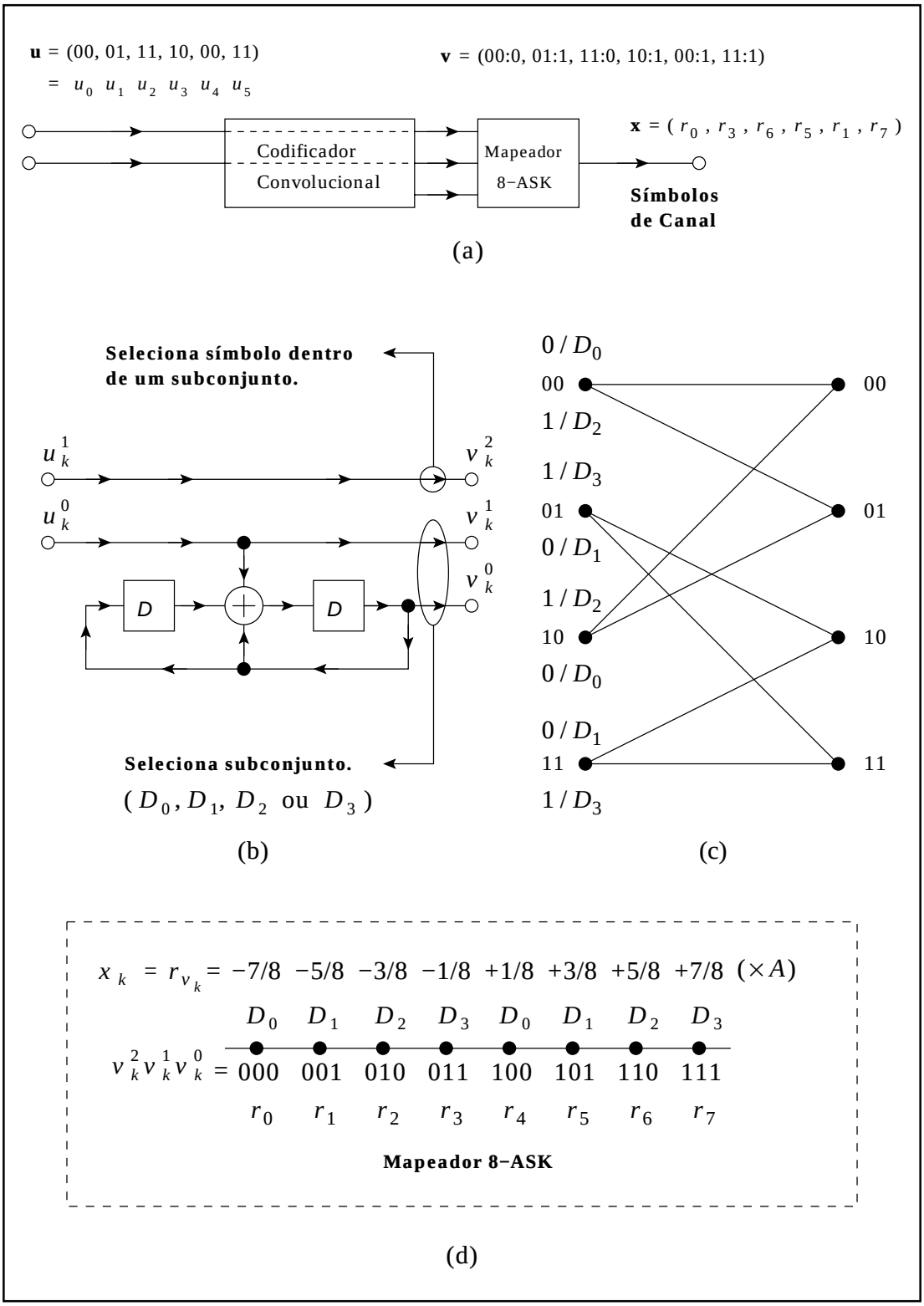


Figura 3.2: Exemplo de codificação com TCM. Processo de codificação em (a), codificador convolucional em (b), treliça correspondente em (c) e mapeador em (d).

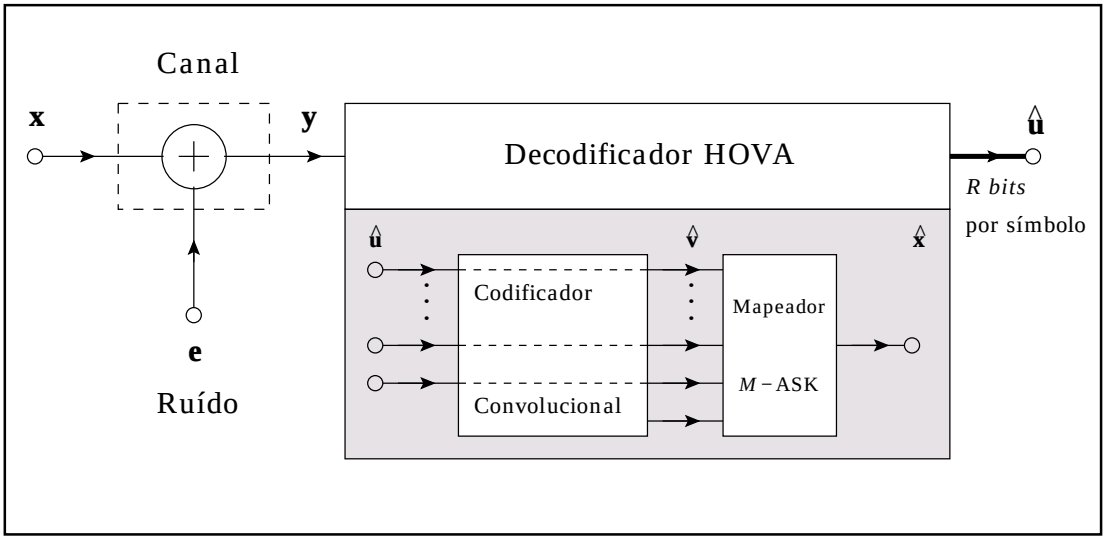


Figura 3.3: Decodificação da TCM. O algoritmo de Viterbi encontra a seqüência de símbolos de canal $\hat{\mathbf{x}}$ que está mais próxima, segundo a métrica euclidiana, da seqüência de canal corrompida por ruído $\mathbf{y}$ e retorna o vetor $\hat{\mathbf{u}}$ correspondente.

reais, produz inevitavelmente um erro entre o valor e sua representação. Por outro lado, dado um número finito de *bits* para representar uma dada variável aleatória, qual o menor erro médio que pode ser obtido? A teoria taxa × distorção tem como objetivo responder este tipo de pergunta. Ou seja, dada a distribuição da fonte e uma medida de distorção (erro) a teoria taxa × distorção fornece as ferramentas para calcular a distorção mínima esperada para uma determinada taxa, medida em *bits* por amostra, ou a taxa mínima necessária para gerar uma dada distorção média.

Seja o problema de se representar uma variável aleatória $X$, que segue uma determinada distribuição, com $R$ *bits*. Desta forma, a representação $Y$ de $X$ pode assumir um total de $2^R$ valores. Então, é necessário encontrar os valores ótimos para compor o alfabeto de reconstrução $\mathcal{Y}$ e a região de reconstrução correspondente a cada um destes valores. É desta forma que o quantizador de Lloyd-Max é construído [20]. Mas, pode-se ver o problema da quantização de outra forma. Ao invés de se quantizar uma única variável aleatória por vez, pode-se quantizar um conjunto de $N$ variáveis aleatórias independentes e identicamente distribuídas (iid) que seguem uma dada distribuição. Agora, o problema é representar todas estas variáveis aleatórias fazendo uso de $NR$ *bits*. Como a fonte é iid, é natural presumir-se que cada símbolo produzido pela fonte deva ser tratado independentemente. Desta forma, a

quantização por amostra também seria a solução ótima para este problema. Porém, os resultados da teoria taxa $\times$ distorção invalidam esta hipótese, como será apresentado a seguir.

### 3.2.1 O Teorema Taxa $\times$ Distorção

**Definição 3.1.** A *função de distorção* ou *medida de distorção* é um mapeamento $d : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$ onde $\mathcal{X}$ é o alfabeto da fonte, $\mathcal{Y}$ é o alfabeto de reprodução e $\mathbb{R}_+$ é o conjunto dos números reais não negativos. A função $d(x, y)$ mede o custo de se representar o símbolo $x$ usando o símbolo $y$. Uma *função de distorção limitada* é aquela para a qual

$$d_{max} \triangleq \max\{d(x, y) : (x, y) \in \mathcal{X} \times \mathcal{Y}\} < \infty. \tag{3.8}$$

A *distorção entre duas seqüências* $\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})$ e $\mathbf{y} = (y_0, y_1, \ldots, y_{N-1})$ é definida por

$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{k=0}^{N-1} d(x_k, y_k). \tag{3.9}$$

**Definição 3.2.** Um *código taxa $\times$ distorção* $(2^{NR}, N)$ é composto por uma *função codificadora*

$$f : \mathcal{X}^N \to 1, 2, \ldots, 2^{NR} \tag{3.10}$$

e por uma *função decodificadora*

$$g : 1, 2, \ldots, 2^{NR} \to \mathcal{Y}^N. \tag{3.11}$$

A distorção associada ao código $(2^{NR}, N)$ é dada por $D = E[d(\boldsymbol{X}, g(f(\boldsymbol{X})))]$, onde o valor esperado é calculado em relação à distribuição da variável aleatória $X$,

$$D = \sum_{\mathbf{x}} p(\mathbf{x}) d(\mathbf{x}, g(f(\mathbf{x}))). \tag{3.12}$$

**Definição 3.3.** A *função informação taxa $\times$ distorção*, $R_I(D)$, para uma fonte $X$ com medida de distorção $d(x, y)$ é definida por

$$R_I(D) = \min \left\{ I(X;Y) : \forall p(y|x) \text{ tal que } \sum_{(x,y)} p(x)p(y|x)d(x, y) \leq D \right\}, \tag{3.13}$$

onde a minimização se dá sobre todas as distribuições condicionais $p(y|x)$ para as quais $p(x, y)$ satisfaz a restrição da distorção esperada.

**Teorema 3.1 (Taxa $\times$ Distorção [4]).** *A função taxa $\times$ distorção para uma fonte iid $X$ com distribuição $p(x)$ e função de distorção limitada $d(x, y)$ é igual a função informação taxa $\times$ distorção associada. Portanto, $R(D) = R_I(D)$ é a taxa mínima realizável para uma dada distorção $D$.*

É possível mostrar formalmente [4] que a função taxa $\times$ distorção para uma fonte gaussiana $\mathcal{N}(0, \sigma^2)$, usando como função distorção o erro quadrático, $d(x_k, y_k) = (x_k - y_k)^2$, é

$$R(D) = \begin{cases} \frac{1}{2}\log_2\left(\frac{\sigma^2}{D}\right), & 0 \leq D \leq \sigma^2 \\ 0, & D > \sigma^2. \end{cases} \tag{3.14}$$

A equação 3.14 pode ser reescrita como $D(R) = \sigma^2 2^{-2R}$. Então, para $R = 1$ a menor distorção esperada para a métrica erro quadrático é $D(1) = 0,25\sigma^2$. O quantizador de Lloyd-Max para a mesma taxa e função de distorção produz uma distorção média aproximadamente igual a $0,36\sigma^2$ [20], ou seja, cerca de 45% maior. Vê-se, então, que considerando vários problemas de taxa $\times$ distorção em conjunto produz-se uma distorção resultante menor.

### 3.2.2 Dualidade entre Codificação de Canal e Taxa $\times$ Distorção

#### Codificação de Canal para Ruído Gaussiano

Seja $Y_k = X_k + Z_k$ onde as variáveis aleatórias $Z_k$ são iid de acordo com uma distribuição normal $\mathcal{N}(0, P_Z)$ e seja $P_X$ a restrição da potência por símbolo da palavra código transmitida, ou seja, $\frac{1}{N}\Sigma_{k=0}^{N-1} x_k^2 \leq P_X$. Isto implica que a seqüência transmitida $\boldsymbol{X} = [X_0 X_1 \ldots X_{N-1}]^T$ está contida numa esfera de raio $\sqrt{NP_X}$ em $\mathbb{R}^N$. O problema da codificação de canal é equivalente ao de encontrar um conjunto de $2^{NR}$ seqüências dentro desta esfera de modo que a probabilidade de se cometer um erro de decisão entre quaisquer duas delas seja pequena. Ou seja, as esferas de ruído de raio $\sqrt{NP_Z}$ ao redor de cada seqüência $\boldsymbol{X}$ são quase disjuntas. Isto corresponde a preencher a esfera de raio $\sqrt{N(P_X + P_Z)}$ com as esferas de raio $\sqrt{NP_Z}$. Pode-se estimar que o maior número de esferas que podem ser inseridas seja a razão entre os seus respectivos volumes. Então, o número de palavras código, $N_{pc}$, que pode ser

transmitido eficientemente é

$$N_{pc} = \frac{\left[\sqrt{N(P_X + P_Z)}\right]^N}{\left[\sqrt{NP_Z}\right]^N} = \left(\frac{P_X + P_Z}{P_Z}\right)^{N/2} = \left(1 + \frac{P_X}{P_Z}\right)^{N/2}. \tag{3.15}$$

O teorema da codificação de canal mostra que é possível fazer isto de forma eficiente para $N$ grande o suficiente. Ou seja, é possível encontrar, aproximadamente,

$$2^{NC} = \left(1 + \frac{P_X}{P_Z}\right)^{N/2} \tag{3.16}$$

palavras código de modo que as esferas de ruído ao redor delas sejam disjuntas [4].

### Taxa × Distorção para Fonte Gaussiana

Seja uma fonte $\boldsymbol{Y} = [Y_0 Y_1 \dots Y_{N-1}]^T$ onde as variáveis aleatórias $Y_k$ são iid de acordo com uma distribuição gaussiana $\mathcal{N}(0, P_Y)$ com $P_Y = \sigma^2$. Seja $\boldsymbol{W} = [W_0 W_1 \dots W_{N-1}]^T$ um código taxa × distorção $(2^{NR}, N)$ para esta fonte, produzindo um vetor distorção $\boldsymbol{Z} = [Z_0 Z_1 \dots Z_{N-1}]^T$ e uma distorção $D = P_Z$, de modo que $\boldsymbol{Y} = \boldsymbol{W} + \boldsymbol{Z}$. Então, um código taxa × distorção $(2^{NR}, N)$ para esta fonte com distorção $D$ corresponde a um conjunto de $2^{NR}$ seqüências em $\mathbb{R}^N$ de modo que a maioria das seqüências da fonte de comprimento $N$, que estão na esfera de raio $\sqrt{NP_Y}$, estejam a uma distância $\sqrt{NP_Z}$ de alguma palavra código $\mathbf{w}$. Utilizando novamente o argumento do empacotamento das esferas, tem-se que o número mínimo de palavras código necessário é

$$2^{NR(P_Z)} = \left(\frac{P_Y}{P_Z}\right)^{N/2}. \tag{3.17}$$

O teorema taxa × distorção assegura que este mínimo é assintoticamente alcançável, ou seja, existe um conjunto de esferas de raio $\sqrt{NP_Z}$ que cobre o espaço exceto por um conjunto de probabilidade arbitrariamente pequena [4]. Além disso, assumindo que $\boldsymbol{W}$ e $\boldsymbol{Z}$ não são correlacionadas, obtém-se

$$2^{NR(P_Z)} = \left(1 + \frac{P_W}{P_Z}\right)^{N/2}. \tag{3.18}$$

Desta forma, um bom código utilizado para transmissão num canal pode ser transformado num bom código para taxa × distorção. Em ambos os casos a idéia essencial é preencher o espaço de seqüências da fonte. Na codificação de canal, deseja-se encontrar o maior conjunto de palavras código $\mathbf{x}$ com a maior distância

mínima entre elas. Já em taxa × distorção deseja-se encontrar o menor conjunto de palavras código **w** que represente (preencha) o espaço inteiro. Tendo-se um conjunto de palavras código que satisfaça o limite de empacotamento por esferas para um propósito, automaticamente, ele satisfará o outro [4].

No caso gaussiano, escolher as palavras código com distribuição gaussiana e com variância apropriada é assintoticamente ótimo tanto para codificação de canal quanto para taxa × distorção [4].

Embora COVER *et al.* [4] utilizem a expressão *empacotamento das esferas* tanto para a codificação quanto para a taxa × distorção, é na codificação de canal que se procura resolver o problema do *empacotamento*, ou seja, distribuir $N_{pc}$ pontos numa esfera no $\mathbb{R}^N$ de modo que se *maximize a distância mínima* entre eles. Já em taxa × distorção procura-se resolver o problema da *cobertura*, isto é, como se deve distribuir $N_{pc}$ pontos em uma esfera no $\mathbb{R}^N$ de modo que se *minimize a distância máxima* de qualquer ponto da esfera ao mais próximo dos $N_{pc}$ pontos. Em geral, para uma esfera no $\mathbb{R}^N$, onde $N > 0$ é um inteiro qualquer, estes dois problemas não possuem a mesma solução, ou seja, o melhor empacotamento não corresponderá, necessariamente, à melhor cobertura [21].

## 3.3 Quantização Codificada por Treliças

A quantização codificada por treliças, ou TCQ, se utiliza das idéias de UNGERBOECK [5, 19, 22], relativas à partição da constelação de sinais do esquema de modulação expandido, para obter um desempenho superior aos esquemas de quantização como os de Lloyd-Max. Na modulação, o HOVA é utilizado para encontrar a seqüência de símbolos de canal mais próxima da seqüência obtida na saída do canal que está corrompida por ruído. Visto que cada seqüência define um código de fonte então, o conjunto de todas as seqüências de símbolos de canal permitidas e o decodificador de Viterbi da modulação codificada por treliças podem ser usados como um código de fonte e um codificador de fonte, respectivamente. Ou seja, dada uma seqüência gerada por uma fonte, o algoritimo de Viterbi encontra a seqüência de símbolos, correspondente aos níveis de reconstrução do quantizador, que minimiza o erro médio quadrático. O desempenho deste tipo de quantizador é

uma conseqüência da cobertura mais eficiente das esferas, mencionada na seção 3.2, em relação ao esquema não codificado representado pelo quantizador de Lloyd-Max. A seguir, então, descreve-se a quantização codificada por treliças como foi proposta por MARCELLIN [6].

### 3.3.1 O Codificador

Na figura 3.4, é mostrado um quantizador codificado por treliças de taxa $R = 2$ *bits* de 4 estados que também utiliza o código convolucional $(7, 2)$. No caso da quantização, o algoritmo de Viterbi é empregado para encontrar o subconjunto que contém o valor mais próximo do valor emitido pela fonte mas restrito às condições impostas pela treliça do código convolucional. De acordo com a figura 3.4a, a fonte emite um vetor $\mathbf{a} = [a_0 a_1 \dots a_{N-1}]^T$, onde $a_k \in \mathcal{A}$ com $\mathcal{A} \subset \mathbb{R}$, que é codificado pelo HOVA, produzindo o vetor $\mathbf{u} = f(\mathbf{a})$ com $R = 2$ *bits* por símbolo. Intrinsecamente, o HOVA procura por um vetor $\mathbf{w} = [w_0 w_1 \dots w_{N-1}]^T$ de símbolos de reconstrução, onde $w_k \in \mathcal{Q}$ para $k = 0, 1, \dots, N-1$, sendo $\mathcal{Q} = \{q_0, q_1, \dots, q_{M-1}\}$ um conjunto com $M$ números reais, que minimize $d_E(\mathbf{a}, \mathbf{w})$ e retorna o $\mathbf{u}$ que produz este $\mathbf{w} = g(\mathbf{u})$. O conjunto $\mathcal{Q}$ também é particionado em quatro subconjuntos

$$\mathcal{D}_0 = \{q_{4j+0} : j = 0, 1, \dots, M/4 - 1\}, \tag{3.19a}$$

$$\mathcal{D}_1 = \{q_{4j+1} : j = 0, 1, \dots, M/4 - 1\}, \tag{3.19b}$$

$$\mathcal{D}_2 = \{q_{4j+2} : j = 0, 1, \dots, M/4 - 1\} \text{ e} \tag{3.19c}$$

$$\mathcal{D}_3 = \{q_{4j+3} : j = 0, 1, \dots, M/4 - 1\}, \tag{3.19d}$$

de tal forma que, para cada valor emitido pela fonte, só é permitido escolher entre os valores no conjunto $\mathcal{A}_0 = \mathcal{D}_0 \cup \mathcal{D}_2$ ou entre os valores no conjunto $\mathcal{A}_1 = \mathcal{D}_1 \cup \mathcal{D}_3$. No caso do exemplo mostrado na figura 3.4, os conjuntos são

$$\mathcal{D}_0 = \left\{q_0 = -\frac{7}{8}A, q_4 = +\frac{1}{8}A\right\}, \tag{3.20a}$$

$$\mathcal{D}_1 = \left\{q_1 = -\frac{5}{8}A, q_5 = +\frac{3}{8}A\right\}, \tag{3.20b}$$

$$\mathcal{D}_2 = \left\{q_2 = -\frac{3}{8}A, q_6 = +\frac{5}{8}A\right\} \text{ e} \tag{3.20c}$$

$$\mathcal{D}_3 = \left\{q_3 = -\frac{1}{8}A, q_7 = +\frac{7}{8}A\right\}. \tag{3.20d}$$

Por exemplo, assumindo que o HOVA inicie a codificação no estado zero, a primeira

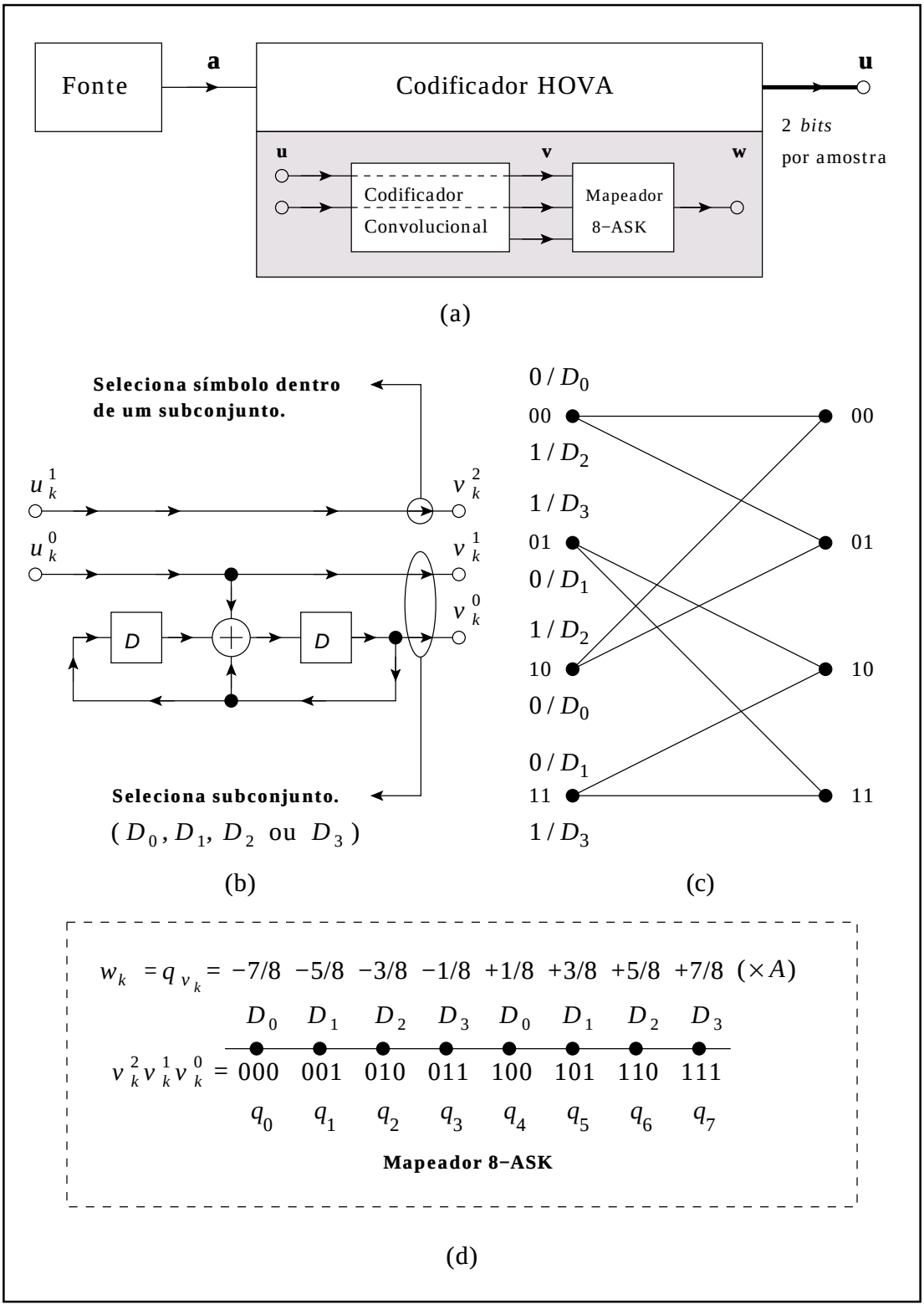


Figura 3.4: Um quantizador codificado por treliças de 2 *bits* e 4 estados em (a) e as estruturas intrínsecas do código convolucional (b), de sua treliça (c) e do mapeador (d) utilizadas pelo algoritmo de Viterbi (HOVA).

amostra só pode ser quantizada utilizando um dos valores do conjunto

$$\mathcal{A}_0 = \left\{ q_0 = -\frac{7}{8}A, q_2 = -\frac{3}{8}A, q_4 = +\frac{1}{8}A, q_6 = +\frac{5}{8}A \right\}. \tag{3.21}$$

Apesar de a TCQ codificar os índices contidos no vetor $\mathbf{v}$ com os mesmos códigos da TCM, os conjuntos $\mathcal{Q}$ e $\mathcal{R}$ não são iguais em geral, refletindo a natureza distinta dos problemas de cobertura e empacotamento.

### 3.3.2 O Decodificador

A decodificação é um processo simples, bastando utilizar como entrada do codificador convolucional da figura 3.4b os símbolos produzidos pelo processo de quantização. O vetor $\mathbf{u}$ é decodificado pelo código convolucional, gerando o vetor de índices $\mathbf{v}$. Este vetor é utilizado pelo mapeador do quantizador para produzir o vetor $\mathbf{w}$ de símbolos de reconstrução de acordo com a relação $w_k = q_{v_k}$.

Os níveis de reconstrução mostrados na figura 3.4d são mais adequados para uma fonte com distribuição uniforme no intervalo $[-A, A]$. Para uma fonte com uma distribuição diferente, como gaussiana ou laplaciana, pode-se utilizar os níveis de reconstrução do quantizador de Lloyd-Max de taxa $R+1$ correspondente. MARCELLIN [6], além de utilizar esta estratégia, otimiza os valores de reconstrução para cada tipo de fonte, taxa $R$ e número de estados, obtendo com isto um ganho extra de desempenho. Na seção A.1 do apêndice A, descreve-se um algoritmo com este propósito.

# Capítulo 4

# Turbo Quantização

Seguindo basicamente o que foi apresentado no capítulo 3, procura-se empregar um bom codificador de canal, a turbo modulação codificada por treliças proposta por ROBERTSON *et al.* [7], como um codificador de fontes sem memória com a intenção de produzir um bom quantizador cujo desempenho possa superar o da quantização codificada por treliças de MARCELLIN [6], sendo este o objetivo do presente capítulo. Primeiramente, descreve-se o codificador de ROBERTSON *et al.* [7], em seguida, a sua utilização como quantizador e, finalmente, são apresentados os resultados obtidos.

## 4.1 Turbo Modulação Codificada por Treliças

Uma importante característica dos códigos turbo é a utilização simples de códigos convolucionais sistemáticos e recursivos num esquema de concatenação paralela, sendo que a utilização de um permutador pseudo-aleatório assegura uma baixa probabilidade de erro [11]. Porém, o mais importante é o fato de que os códigos turbo podem ser decodificados iterativamente e com bom desempenho. Por outro lado, a TCM obtém ganhos de codificação significativos em relação aos esquemas de codificação multinível não codificados sem a necessidade de se expandir a banda de transmissão, sendo um esquema de alta eficiência espectral.

A turbo modulação codificada por treliças, ou TTCM, procura reunir num único esquema de codificação as características da TCM e dos códigos turbo. Basicamente, ROBERTSON *et al.* [11] substituem os dois codificadores convolucionais

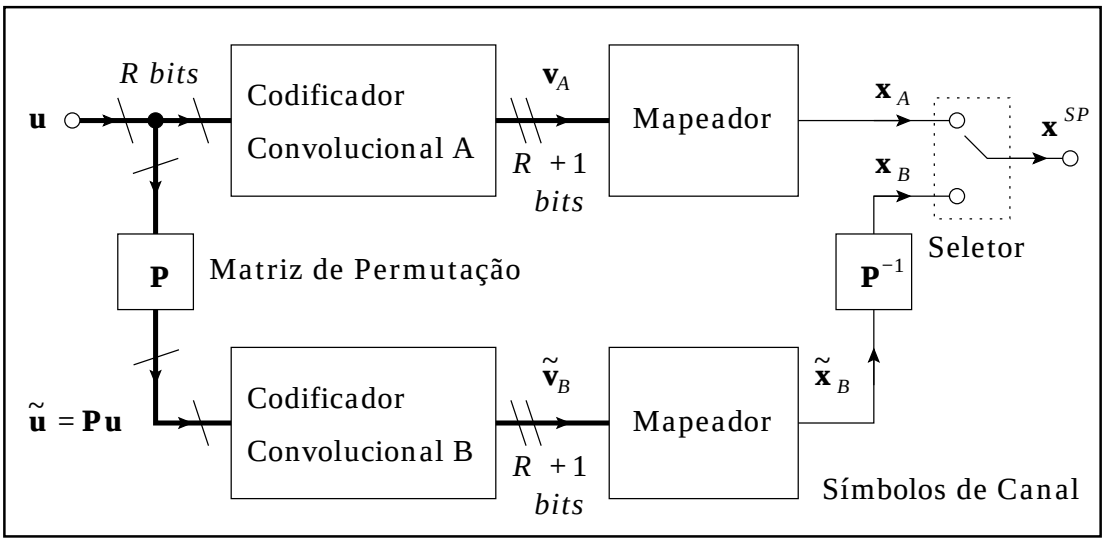


Figura 4.1: O codificador da turbo modulação codificada por treliças.

dos códigos turbo por dois codificadores idênticos àqueles utilizados na TCM.

### 4.1.1 O Codificador

Na figura 4.1, tem-se o diagrama de blocos do codificador da TTCM. Ele é composto por dois codificadores convolucionais sistemáticos recursivos seguidos por mapeadores que selecionam o sinal modulado a ser transmitido pelo canal. Cada codificador convolucional produz um *bit* de paridade para cada grupo de $R$ *bits* de entrada, gerando localmente um código de taxa $R/(R+1)$, e quando se emprega um modulador ASK somente um dos $R$ *bits* de cada símbolo do vetor $\mathbf{u}$ é codificado, da mesma forma como é feito na TCM. Em relação aos codificadores dos códigos turbo, há algumas diferenças dentre as quais se pode destacar:

1. o emprego da permutação por grupo de *bits*, onde cada grupo contém $R$ *bits*. Os *bits* dentro de cada grupo não são permutados, sendo cada um destes grupos chamado de símbolo de canal;

2. a maior complexidade da operação de perfuração da informação de paridade para obter a eficiência espectral desejada;

3. a necessidade de restrições específicas na estrutura dos permutadores.

Com o auxílio da figura 4.2, onde é apresentado um exemplo de codificação empregando TTCM, será possível compreender melhor estas diferenças. Na figura

4.2a, tem-se um codificador com eficiência espectral de 2 *bits*/s/Hz, ou seja, cada símbolo de canal representa 2 *bits* da mensagem $\mathbf{u}$. A figura 4.2b ilustra como o conjunto de símbolos de canal é particionado nos subconjuntos $\mathcal{D}_0 = \{r_0, r_4\}$, $\mathcal{D}_1 = \{r_1, r_5\}$, $\mathcal{D}_2 = \{r_2, r_6\}$ e $\mathcal{D}_3 = \{r_3, r_7\}$. A figura 4.2b também ilustra como os *bits* produzidos pelos codificadores convolucionais selecionam o subconjunto e os elementos em cada subconjunto. Na verdade, este é o mesmo mapeamento adotado na TCM-ASK apresentado no capítulo 3.

Primeiramente, a seqüência de entrada $\mathbf{u} = (00, 01, 11, 10, 00, 11)$ é injetada no codificador convolucional $A$, produzindo a saída $\mathbf{v}_A = (00{:}0, 01{:}1, 11{:}0, 10{:}1, 00{:}1, 11{:}1)$. A mesma seqüência $\mathbf{u}$ é permutada, obtendo-se $\tilde{\mathbf{u}} = \mathbf{P}\mathbf{u} = (11, 11, 00, 01, 00, 10)$. Observe que o permutador do exemplo foi construído de tal forma que somente são permutadas posições pares com pares e ímpares com ímpares. Isto será justificado a seguir. A seqüência $\tilde{\mathbf{u}}$ é injetada no codificador convolucional $B$, produzindo a saída $\tilde{\mathbf{v}}_B = (11{:}1, 11{:}0, 00{:}1, 01{:}0, 00{:}1, 10{:}1)$. Os símbolos desta seqüência são permutados, por meio da permutação inversa, obtendo-se $\mathbf{v}_B = \mathbf{P}^{-1}\tilde{\mathbf{v}}_B = (00{:}1, 01{:}0, 11{:}1, 10{:}1, 00{:}1, 11{:}0)$. Visto que o permutador mapeia posições pares em pares e ímpares em ímpares e que os codificadores convolucionais atuam em grupos de $R = 2$ *bits* por vez, a seqüência de *bits* sistemáticos de $\mathbf{v}_A$ e $\mathbf{v}_B$ são iguais. Desta forma, a perfuração dos *bits* de paridade é feita selecionando-se alternadamente as componentes dos vetores de símbolos de canal $\mathbf{x}_A$ e $\mathbf{x}_B$, ou seja, $x_k = x_{A,k}$ para $k$ par e $x_k = x_{B,k}$ para $k$ ímpar. Conseqüentemente, a TTCM utiliza a mesma banda da TCM. O sobrescrito $SP$ do vetor $\mathbf{x}^{SP}$ tem por finalidade ressaltar que cada um de seus símbolos transmite simultaneamente *bits* sistemáticos e de paridade.

### 4.1.2 O Decodificador

O decodificador da turbo modulação codificada por treliças é muito semelhante àquele utilizado na decodificação dos códigos turbo. Entretanto, há uma diferença em relação ao tipo de informação que os decodificadores trocam entre si e no modo como a primeira iteração deve ser iniciada. Outra característica, que é particular a este processo de decodificação, é o fato de cada um dos decodificadores receber alternadamente o símbolo corrompido por ruído do codificador correspondente e o símbolo corrompido pelo ruído do outro codificador. O símbolo que corresponde ao

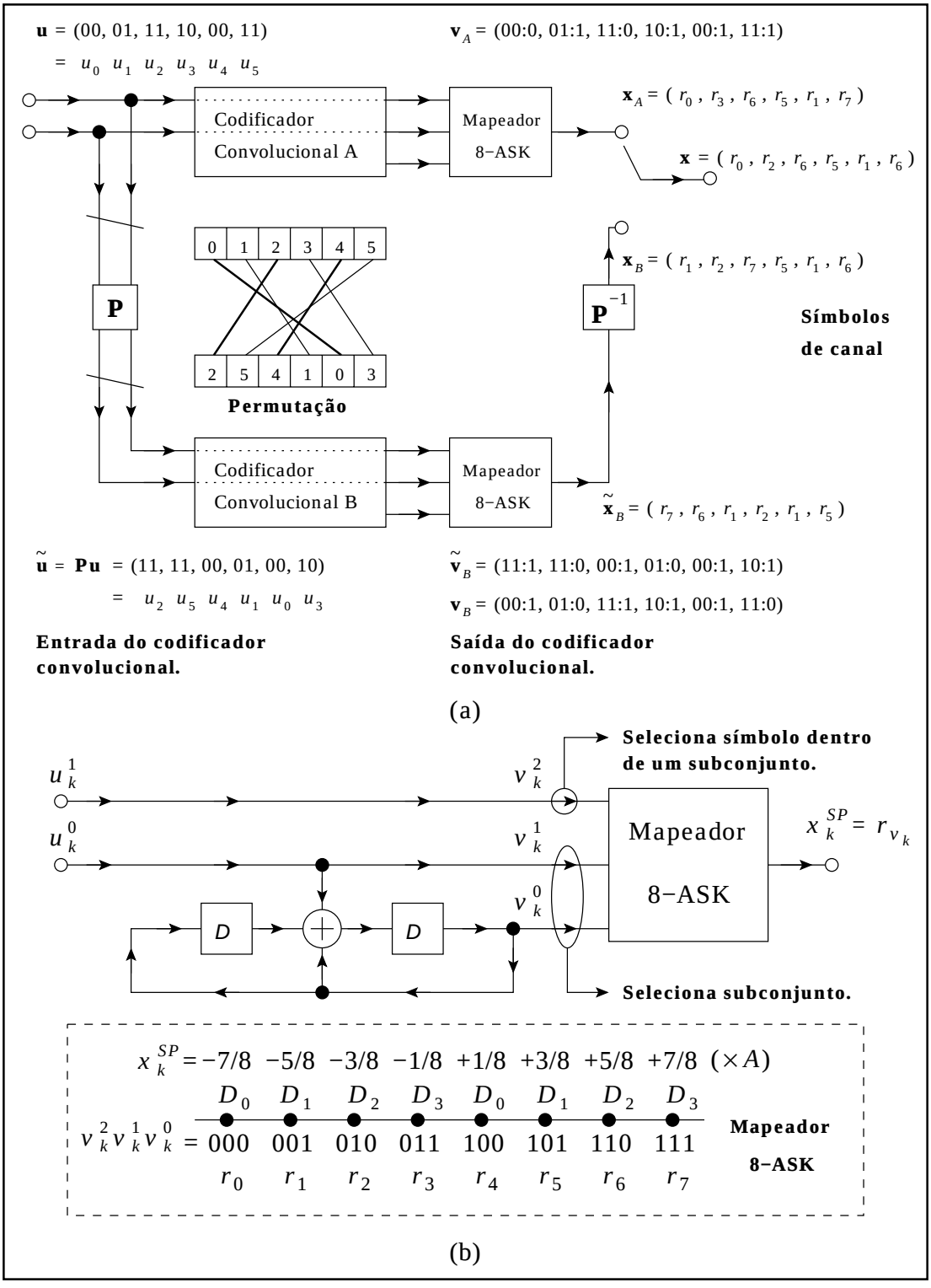


Figura 4.2: Exemplo de turbo modulação codificada por treliças. O processo de codificação é mostrado em (a) e detalhes do codificador convolucional e do mapeador são mostrados em (b).

outro codificador é ignorado e o decodificador corrente utiliza somente a informação *a priori* fornecida pelo outro decodificador que não ignorou este símbolo. E é por isto que a iniciação do processo de decodificação da turbo modulação é um pouco diferente daquele dos códigos turbo. Na primeira parte da primeira iteração, a informação *a priori* do símbolo ignorado deve ser estimada. Esta estimativa é feita utilizando-se o valor de $\Pr\{U_{k-1} = i|y_{k-1}\}$ [11], ou seja,

$$\begin{aligned}
\Pr\{U_{k-1} = i|y_{k-1}\} &= \frac{\Pr\{U_{k-1} = i\}}{\Pr\{y_{k-1}\}} \Pr\{y_{k-1}|U_{k-1} = i\} \\
&= C_{k-1} \sum_{j=0}^{1} \Pr\{y_{k-1}, V_{k-1}^0 = j|U_{k-1} = i\} \\
&= C_{k-1} \sum_{j=0}^{1} \big[\Pr\{y_{k-1}|U_{k-1} = i, V_{k-1}^0 = j\} \times \\
&\quad \times \Pr\{V_{k-1}^0 = j|U_{k-1} = i\}\big] \\
&= \frac{C_{k-1}}{2} \sum_{j=0}^{1} \Pr\{y_{k-1}|U_{k-1} = i, V_{k-1}^0 = j\} \\
&= \frac{C_{k-1}}{2} \sum_{j=0}^{1} \Pr\{y_{k-1}|[V_{k-1}]_1^R = i, V_{k-1}^0 = j\}, \qquad (4.1)
\end{aligned}$$

onde $v_{k-1}^0$ é o *bit* de paridade correspondente aos $R$ *bits* do símbolo de entrada $[u_{k-1}]_0^{R-1} = u_{k-1}^{R-1} \ldots u_{k-1}^1 u_{k-1}^0$, que são representados na forma decimal pelo inteiro $i \in \{0, 1, \ldots, 2^R - 1\}$, $v_{k-1}^{r+1} = u_{k-1}^r$ para $r = 0, 1, \ldots, R-1$ e assumiu-se que $\Pr\{V_{k-1}^0 = j|U_{k-1} = i\} = 1/2$. Note que, $[v_{k-1}]_0^R = v_{k-1}^R \ldots v_{k-1}^1 v_{k-1}^0$ é o índice que seleciona o sinal modulado a ser transmitido no canal. Desta forma, tem-se que

$$\frac{\Pr\{U_{k-1} = i|y_{k-1}\}}{\Pr\{U_{k-1} = 0|y_{k-1}\}} = \frac{\sum_{j=0}^{1} \Pr\{y_{k-1}|[V_{k-1}]_1^R = i, V_{k-1}^0 = j\}}{\sum_{j=0}^{1} \Pr\{y_{k-1}|[V_{k-1}]_1^R = 0, V_{k-1}^0 = j\}}, \qquad (4.2)$$

onde assumiu-se, como de costume, que $\Pr\{U_{k-1} = i\} = 1/2^R$ para $i = 0, 1, \ldots, 2^R - 1$.

No capítulo 2, foi mostrado que a saída de cada decodificador componente dos códigos turbo pode ser dividida em três partes:

1. a informação *a priori*,

2. a informação sistemática

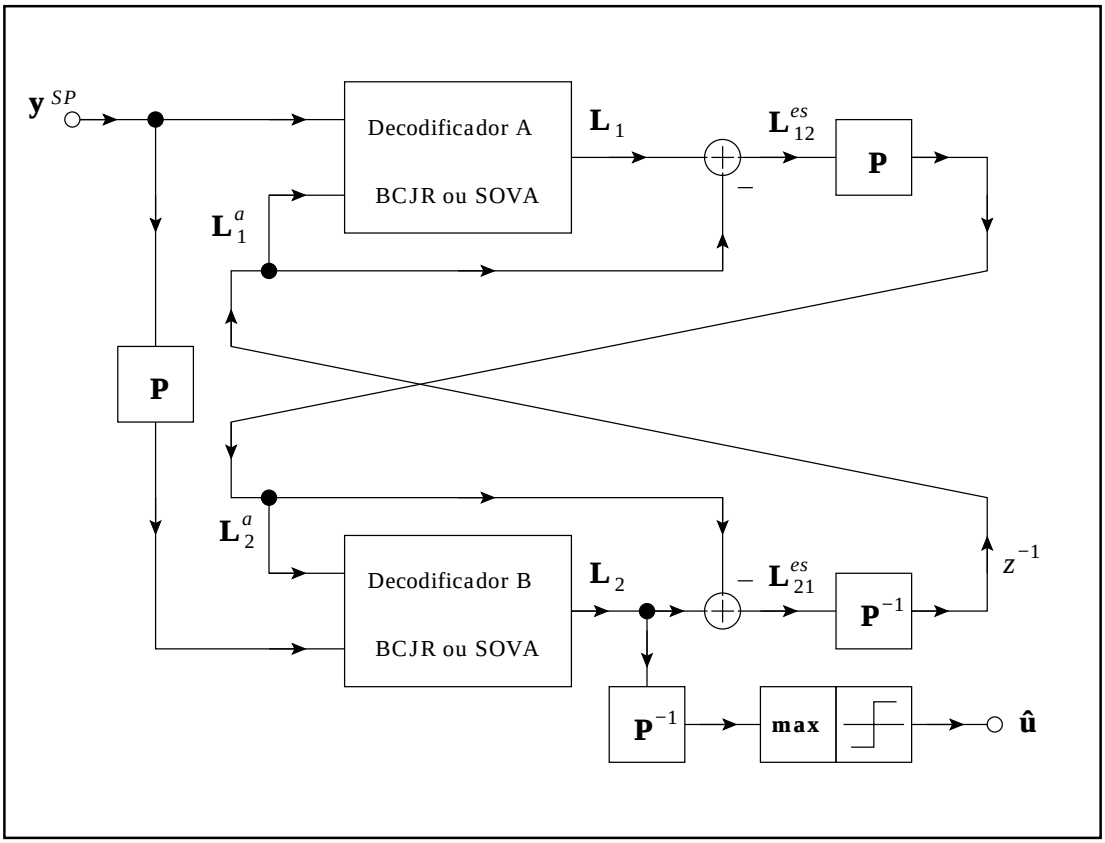


Figura 4.3: O decodificador da turbo modulação codificada por treliças.

3. e a informação extrínseca

que são afetadas por ruídos independentes. Na turbo modulação, ver figura 4.3, a componente sistemática não pode ser separada da componente extrínseca, pois o mesmo símbolo de canal transporta a informação sistemática e de paridade em contraste com os códigos turbo. Na figura 4.3, estas componentes de informação mista correspondem às matrizes $\mathbf{L}_{12}^{es}$ e $\mathbf{L}_{21}^{es}$, onde o sobrescrito *es* significa informação extrínseca e sistemática não separáveis. Desta forma, no esquema da turbo modulação, a saída de cada decodificador pode ser dividida em duas partes:

1. a informação *a priori*

2. e a informação extrínseca e sistemática.

### 4.1.3 O Algoritmo BCJR Aplicado à Turbo Modulação

Visto que o codificador da turbo modulação, em geral, recebe como entrada qualquer um dos símbolos pertencentes ao conjunto $\mathcal{U} = \{0, 1, \ldots, 2^R - 1\}$, o deco-

dificador BCJR-MAP primeiramente encontra o símbolo $i \in \mathcal{U}^*$, onde $\mathcal{U}^* = \mathcal{U} - \{0\}$, para o qual

$$\frac{\Pr\{U_{k-1} = i|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}} = \max\left\{\frac{\Pr\{U_{k-1} = j|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}} : j \in \mathcal{U}^*\right\}. \tag{4.3}$$

Isto feito, para $k = 1, 2, \ldots, N$, decide que o símbolo $u_{k-1} = i$ foi transmitido se

$$\Pr\{U_{k-1} = i|\mathbf{y}\} > \Pr\{U_{k-1} = 0|\mathbf{y}\}. \tag{4.4}$$

Caso contrário, decide que o símbolo $u_{k-1} = 0$ foi transmitido. Generalizando a expressão do logaritmo da razão das probabilidades *a posteriori* tem-se que

$$L_{k-1}(i) = \ln\left[\frac{\Pr\{U_{k-1} = i|\mathbf{y}\}}{\Pr\{U_{k-1} = 0|\mathbf{y}\}}\right] \tag{4.5}$$

e, desta forma,

$$u_{k-1} = \begin{cases} i, & L_{k-1}(i) > 0 \\ 0, & L_{k-1}(i) < 0, \end{cases} \tag{4.6}$$

onde $i \in \mathcal{U}^*$ é o símbolo para o qual $L_{k-1}(i) = \max\{L_{k-1}(j) : j \in \mathcal{U}^*\}$. Seguindo os mesmos passos da seção 2.3, conclui-se que a equação 4.5 pode ser reescrita como

$$L_{k-1}(i) = \ln\left[\frac{\sum\limits_{(s',s)\in\mathcal{B}_{k-1}^{i}} \Pr\{s_{k-1} = s', s_k = s|\mathbf{y}\}}{\sum\limits_{(s',s)\in\mathcal{B}_{k-1}^{0}} \Pr\{s_{k-1} = s', s_k = s|\mathbf{y}\}}\right], \tag{4.7}$$

onde $\mathcal{B}_{k-1}^{i}$ é o conjunto de todas as transições de estado $(s', s)$ produzidas por um símbolo de entrada $u_{k-1} = i$ e $\mathcal{B}_{k-1}^{0}$ é o conjunto correspondente de transições $(s', s)$ produzidas por um símbolo de entrada $u_{k-1} = 0$. Desta forma, pode-se mostrar que

$$L_{k-1}(i) = \ln\left[\frac{\sum\limits_{(s',s)\in\mathcal{B}_{k-1}^{i}} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}{\sum\limits_{(s',s)\in\mathcal{B}_{k-1}^{0}} \alpha_{k-1}(s')\gamma_k(s', s)\beta_k(s)}\right], \tag{4.8}$$

onde $\alpha_k(s)$, $\gamma_k(s', s)$ e $\beta_k(s)$ são definidos como na seção 2.3 e $\alpha_k(s)$ e $\beta_k(s)$ podem ser calculados recursivamente como demonstrado nesta mesma seção.

Para o cálculo de $\gamma_k(s', s)$ assume-se que o canal é afetado por um ruído com distribuição gaussiana e que a transmissão é feita utilizando $M$-ASK. Conseqüentemente, cada símbolo de entrada $u_{k-1}$ é mapeado num símbolo de canal

$x_{k-1}^{SP} \in \mathcal{R} = \{r_0, r_1, \ldots, r_{M-1}\}$, onde $M = 2^{R+1}$, e $y_{k-1}^{SP}$ é o sinal $x_{k-1}^{SP}$ corrompido pelo ruído. Defina

$$L_{k-1}^{a}(i) = \ln\left[\frac{\Pr\{U_{k-1} = i\}}{\Pr\{U_{k-1} = 0\}}\right]. \tag{4.9}$$

Então, visto que $\Pr\{U_{k-1} = 0\} = 1 - \sum_{j=1}^{M/2-1} \Pr\{U_{k-1} = j\}$, obtêm-se

$$\begin{aligned} \Pr\{U_{k-1} = i\} &= \Pr\{U_{k-1} = 0\} e^{L_{k-1}^{a}(i)} \\ &= \frac{e^{L_{k-1}^{a}(i)}}{1 + \sum_{j=1}^{M/2-1} e^{L_{k-1}^{a}(j)}} \end{aligned} \tag{4.10}$$

e

$$\Pr\{U_{k-1} = 0\} = \frac{1}{1 + \sum_{j=1}^{M/2-1} e^{L_{k-1}^{a}(j)}}. \tag{4.11}$$

Portanto, partindo-se da definição de $\gamma_k(s', s)$, se $(s', s) \in \mathcal{B}_{k-1}^{i}$ então

$$\begin{aligned} \gamma_k(s', s) &= \Pr\{S_k = s, y_{k-1} | S_{k-1} = s'\} \\ &= \Pr\{S_k = s | S_{k-1} = s'\} \Pr\{y_{k-1} | S_{k-1} = s', S_k = s\} \\ &= \Pr\{U_{k-1} = i\} \Pr\{y_{k-1} | X_{k-1} = x_{k-1}\} \\ &= \Pr\{U_{k-1} = i\} \Pr\{y_{k-1}^{SP} | X_{k-1}^{SP} = x_{k-1}^{SP}(s', s)\}. \end{aligned} \tag{4.12}$$

Como o canal é gaussiano,

$$\Pr\{y_{k-1}^{SP} | X_{k-1}^{SP} = x_{k-1}^{SP}(s', s)\} \propto e^{-\frac{\left(y_{k-1}^{SP} - x_{k-1}^{SP}(s', s)\right)^2}{2\sigma^2}} \tag{4.13}$$

e, assim sendo, a equação 4.12 pode ser reescrita como

$$\begin{aligned} \gamma_k(s', s) &= \frac{e^{L_{k-1}^{a}(i)}}{1 + \sum_{j=1}^{M/2-1} e^{L_{k-1}^{a}(j)}} e^{-\frac{\left(y_{k-1}^{SP}\right)^2}{2\sigma^2}} e^{\frac{x_{k-1}^{SP}(s', s)\left(y_{k-1}^{SP} - \frac{1}{2} x_{k-1}^{SP}(s', s)\right)}{\sigma^2}} \\ &= C_{k-1} e^{L_{k-1}^{a}(i)} e^{\frac{x_{k-1}^{SP}(s', s)\left(y_{k-1}^{SP} - \frac{1}{2} x_{k-1}^{SP}(s', s)\right)}{\sigma^2}}. \end{aligned} \tag{4.14}$$

Procedendo de forma analoga, se $(s', s) \in \mathcal{B}_{k-1}^{0}$ então

$$\gamma_k(s', s) = \Pr\{U_{k-1} = 0\} \Pr\{y_{k-1}^{SP} | X_{k-1}^{SP} = x_{k-1}^{SP}(s', s)\} \tag{4.15}$$

e, por conseqüência,

$$\gamma_k(s', s) = C_{k-1} e^{\frac{x_{k-1}^{SP}(s',s)\left(y_{k-1}^{SP} - \frac{1}{2}x_{k-1}^{SP}(s',s)\right)}{\sigma^2}}. \tag{4.16}$$

Definindo gama extrínseco como

$$\gamma_k^{es}(s', s) = e^{\frac{x_{k-1}^{SP}(s',s)\left(y_{k-1}^{SP} - \frac{1}{2}x_{k-1}^{SP}(s',s)\right)}{\sigma^2}} \tag{4.17}$$

e substituindo adequadamente as equações 4.14 e 4.16 na equação 4.8, obtém-se a forma final

$$L_{k-1}(i) = L_{k-1}^{a}(i) + \ln\left[\frac{\sum_{(s',s)\in\mathcal{B}_{k-1}^{i}} \alpha_{k-1}(s')\gamma_k^{es}(s', s)\beta_k(s)}{\sum_{(s',s)\in\mathcal{B}_{k-1}^{0}} \alpha_{k-1}(s')\gamma_k^{es}(s', s)\beta_k(s)}\right]. \tag{4.18}$$

Ou seja, $L_{k-1}(i)$ é a soma da informação *a priori*, $L_{k-1}^{a}(i)$, e da informação extrínseca e sistemática não separáveis.

## 4.2 Turbo Quantização Codificada por Treliças

A utilização da turbo modulação como código taxa $\times$ distorção é feita de forma direta como no caso da quantização codificada por treliças de MARCELLIN [6]. Na figura 4.4, é mostrado o diagrama de blocos do quantizador proposto. A fonte discreta no tempo produz um sinal $\mathbf{a}$ que é visto pelo decodificador da turbo modulação como se fosse a saída de um canal ruidoso. Este decodificador, que está atuando como codificador do ponto de vista da quantização, procura encontrar a seqüência de *bits* $\mathbf{u} = f(\mathbf{a})$ que melhor aproxima a seqüência $\mathbf{a}$. A seqüência de *bits* $\mathbf{u}$ aplicada ao codificador da turbo modulação, que atua como decodificador para a quantização, produz uma saída $\mathbf{w} = g(\mathbf{u}) = g(f(\mathbf{a}))$ que é a representação de $\mathbf{a}$, conforme a notação utilizada no capítulo 3. Contudo, o processo de codificação da TTCQ pode ser executado por meio de diversos algoritmos como os que serão descritos nas subseções a seguir.

### 4.2.1 Algoritmo Serial

O algoritmo básico de codificação da TTCQ é o próprio algoritmo de decodificação empregado na TTCM. Trata-se de um algoritmo serial pois os codificadores

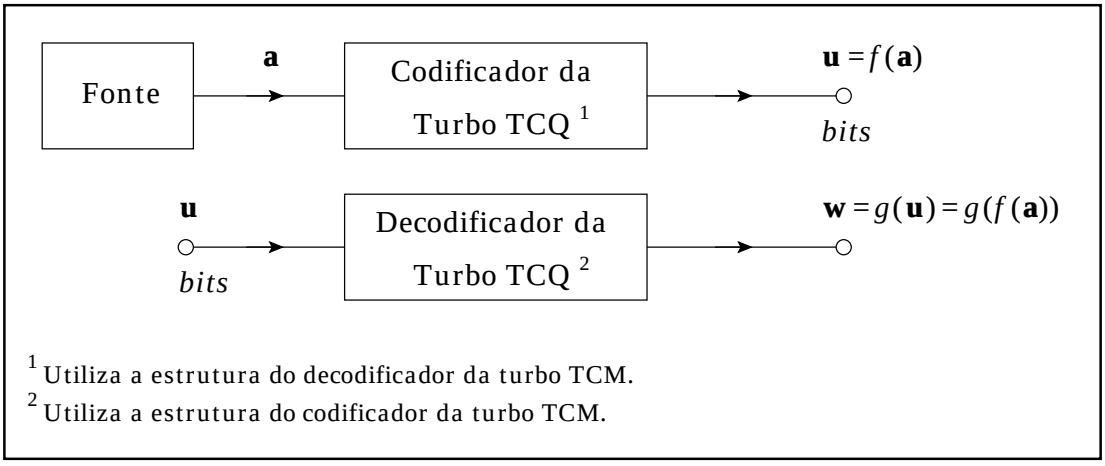


Figura 4.4: Turbo quantização codificada por treliças.

componentes são ativados seqüencialmente ($A \to B \to A \to B \to \cdots$). Nenhuma mudança é feita em relação ao algoritmo empregado na TTCM exceto pelo mapeador que, como já foi visto, produz um vetor $\mathbf{w} \in \mathcal{Q}^N$, mais adequado para um código taxa $\times$ distorção, ao invés do vetor $\mathbf{x} \in \mathcal{R}^N$, mais adequado para um código de canal. Desta forma, dado o número de iterações, $N_{it}$, o algoritmo serial básico de codificação para a TTCQ consiste nos seguintes passos:

1. $\mathbf{L}_1^{a(0)} = \mathbf{L}$; // Usando a estimativa da equação 4.2.
   $i = 1$;

2. // Codificando o vetor $\mathbf{a}$ gerado pela fonte.
   // Codificador $A$.
   $\mathbf{L}_{12}^{es(i)} = \mathbf{L}_1^{(i)} - \mathbf{L}_1^{a(i-1)}$;
   $\mathbf{L}_2^{a(i)} = \mathbf{P}\mathbf{L}_{12}^{es(i)}$;
   // Codificador $B$.
   $\mathbf{L}_{21}^{es(i)} = \mathbf{L}_2^{(i)} - \mathbf{L}_2^{a(i)}$;
   $\mathbf{L}_1^{a(i)} = \mathbf{P}^{-1}\mathbf{L}_{21}^{es(i)}$;

3. Se ( $i == N_{it}$ )
   {
   // O $\max\{\cdot\}$ efetua a operação indicada na equação 4.3.
   // A função $h$ efetua a decisão de limiar (*hard decision*).
   $\mathbf{u} = h(\max\{\mathbf{P}^{-1}\mathbf{L}_2^{(N_{it})}\})$;
   Parar;

}

4. $i = i + 1$;

   Ir para 2;

Como no caso da TTCM, $\mathbf{P}$ é um permutador pseudo-aleatório, são utilizados blocos de tamanho relativamente grande ($N = 1000$) e adota-se um pequeno número de iterações ($N_{it} = 4$). A idéia é verificar se esta configuração é capaz de produzir resultados melhores que os da TCQ.

### 4.2.2 Algoritmo Serial Refinado

Com o intuito de melhorar o desempenho do codificador serial apresentado na subseção anterior, CHAPPELIER *et al.* [23] propuseram um algoritmo que conta o número de iterações seguidas, $n_{\mathbf{u}}$, para as quais o vetor $\mathbf{u}^{(i)}$ se mantém inalterado, onde $\mathbf{u}^{(i)} = h(\max\{\mathbf{P}^{-1}\mathbf{L}_2^{(i)}\})$ para $i = 1, 2, \ldots, N_{it}$. Isto é, se $\mathbf{u}^{(i)} = \mathbf{u}^{(i+1)} = \cdots = \mathbf{u}^{(N_{it})}$ e $i$ é a menor iteração para a qual isto ocorre então $n_{\mathbf{u}} = N_{it} - i$. Assim sendo, ao final das $N_{it}$ iterações, CHAPPELIER *et al.* [23] checavam se $\mathbf{u}^{(i)}$ tinha se mantido inalterado por pelo menos $N_{\mathbf{u}} = 50$ iterações[1]. Assumiam que o algoritmo havia convergido em caso afirmativo e $\mathbf{u} = \mathbf{u}^{(N_{it})}$. Caso contrário, $\mathbf{u} = \mathbf{u}_{min}$, onde o vetor $\mathbf{u}_{min}$ é igual ao vetor $\mathbf{u}^{(i)}$ que produziu a menor distorção no decorrer das $N_{it}$ iterações.

Neste trabalho, foram feitas algumas alterações no algoritmo do parágrafo anterior, resultando no que se chamou de *algoritmo serial refinado*. A primeira mudança faz com que o novo algoritmo sempre retorne $\mathbf{u}_{min}$ independentemente do valor de $n_{\mathbf{u}}$. Desta forma, não se está assumindo que $n_{\mathbf{u}} \geq N_{\mathbf{u}}$ implica $\mathbf{u}_{min} = \mathbf{u}^{(N_{it})}$. A segunda mudança introduz um critério de parada baseado no elemento de maior valor absoluto da matriz $\Delta\mathbf{L}_{21}^{es(i)}$, denominado por $\Delta_{max}^{es} = \max(\text{abs}(\Delta\mathbf{L}_{21}^{es(i)}))$. Dado $\epsilon \geq 0$, se $\Delta_{max}^{es} \leq \epsilon$ e $n_{\mathbf{u}} \geq N_{\mathbf{u}}$ ou se $\Delta_{max}^{es}$ é igual a 0 (zero) ou se o número máximo de iterações foi atingido, então o algoritmo é interrompido e retorna $\mathbf{u}_{min}$, pois considera-se que um ponto fixo do codificador da TTCQ foi encontrado para o vetor

[1] Como será visto na seção de resultados, uma das condições para a TTCQ gerar distorções menores que a TCQ é utilizar um elevado número de iterações (1000 ou mais).

de fonte $\mathbf{a}$ que está sendo quantizado[2]. O objetivo destas mudanças é diminuir a distorção e o tempo de processamento, respectivamente. Adicionalmente, armazena-se $i_{min}$, a iteração correspondente a $\mathbf{u}_{min}$, para estimar seu valor médio ($\mu_{I_{min}}$) e compará-lo com $N_{it}$. Armazena-se também $i_{max}$, o número de iterações executado, para poder estimar seu valor médio ($\mu_{I_{max}}$). O valor estimado de $\mu_{I_{max}}$ servirá para comprovar a redução do tempo de processamento. Então, dados $N_{it}$, $N_{\mathbf{u}}$ e $\epsilon \geq 0$, os passos do algoritmo serial refinado são:

1. $n_{\mathbf{u}} = 0$;
   $i_{min} = 1$;
   $D_{min} = \infty$;
   $\mathbf{u}^{(0)} = \mathbf{0}$;
   $\mathbf{L}_1^{a(0)} = \mathbf{L}$;
   $i = 1$;

2. // Codificando o vetor $\mathbf{a}$ gerado pela fonte.
   // Codificador $A$.
   $\mathbf{L}_{12}^{es(i)} = \mathbf{L}_1^{(i)} - \mathbf{L}_1^{a(i-1)}$;
   $\mathbf{L}_2^{a(i)} = \mathbf{P}\mathbf{L}_{12}^{es(i)}$;

   // Codificador $B$.
   $\mathbf{L}_{21}^{es(i)} = \mathbf{L}_2^{(i)} - \mathbf{L}_2^{a(i)}$;
   $\mathbf{L}_1^{a(i)} = \mathbf{P}^{-1}\mathbf{L}_{21}^{es(i)}$;

3. $\mathbf{u}^{(i)} = h(\max\{\mathbf{P}^{-1}\mathbf{L}_2^{(i)}\})$;

4. Se ( $\mathbf{u}^{(i)} == \mathbf{u}^{(i-1)}$ )
   {
   Se ( $i > 1$ ) Ir para 6;
   }
   Senão
   {
   $n_{\mathbf{u}} = 0$;
   }

[2]Na verdade, um ponto fixo é encontrado quando $\Delta_{max}^{es} = 0$ já que $\Delta_{max}^{es} = 0 \Rightarrow \Delta\mathbf{L}_{21}^{es(i)} = \mathbf{L}_2 - \mathbf{P}\mathbf{L}_1 = \mathbf{0}$ (Ver definição 2.1).

5. $\mathbf{w}^{(i)} = g(\mathbf{u}^{(i)})$; // Decodificar $\mathbf{u}^{(i)}$, produzindo $\mathbf{w}^{(i)}$.
   $D = \frac{1}{N} \sum_{k=0}^{N-1} \left[ a_k - w_k^{(i)} \right]^2$; // Calculando a distorção entre $\mathbf{a}$ e $\mathbf{w}^{(i)}$.

6. Se ( $n_{\mathbf{u}} \neq 0$ )
   {
   Ir para 8;
   }

7. Se ( $D < D_{min}$ )
   {
   $i_{min} = i$;
   $D_{min} = D$;
   $\mathbf{u}_{min} = \mathbf{u}^{(i)}$;
   }

8. $n_{\mathbf{u}} = n_{\mathbf{u}} + 1$;

9. $\Delta \mathbf{L}_{21}^{es(i)} = \mathbf{L}_{21}^{es(i)} - \mathbf{L}_{21}^{es(i-1)} = \mathbf{L}_{21}^{es(i)} - \mathbf{PL}_{1}^{a(i-1)}$;
   $\Delta_{max}^{es} = \max(\text{abs}(\Delta \mathbf{L}_{21}^{es(i)}))$;

10. Se ((( $n_{\mathbf{u}} \geq N_{\mathbf{u}}$ ) e ( $\Delta_{max}^{es} \leq \epsilon$ )) ou ( $\Delta_{max}^{es} == 0$ ) ou ( $i == N_{it}$ ))
    {
    $\mathbf{u} = \mathbf{u}_{min}$;
    Parar;
    }

11. $i = i + 1$;
    Ir para 2;

### 4.2.3 Algoritmo Paralelo

Visto que a convergência do código turbo é bastante dependente de uma boa iniciação do processo iterativo, CHAPPELIER *et al.* [23] propuseram também quantizar o vetor de entrada $\mathbf{a}$ de formas diferentes através da alteração dos rótulos da treliça do código convolucional, como mostrado na figura 4.5, gerando de

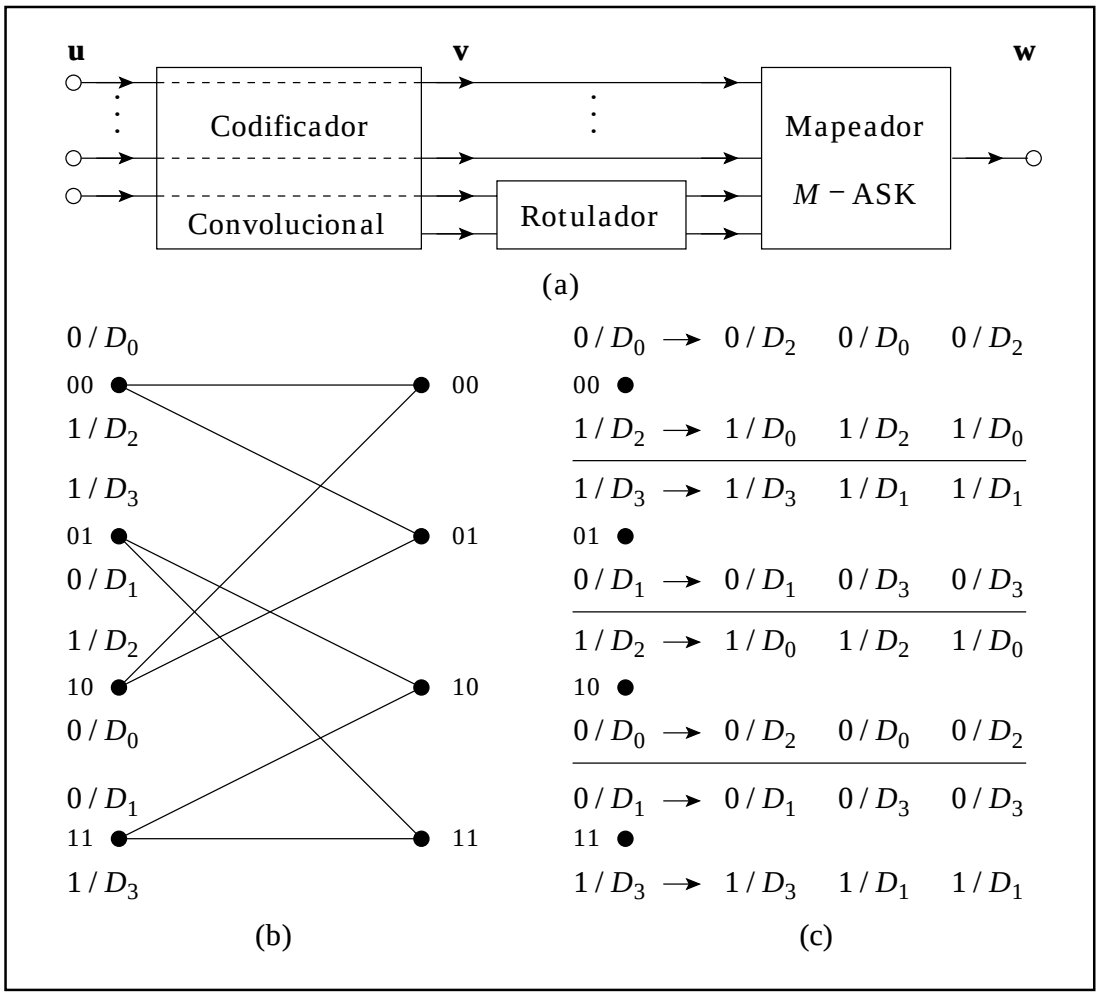


Figura 4.5: Modos diferentes de rotular uma treliça da TTCQ. Em (a), o rotulador altera os *bits* $v_k^1 v_k^0$ de $\mathbf{v}$ para rotular de forma diferente os ramos da treliça em (b) de acordo com o mapeamento indicado em (c).

duas a quatro formas distintas de quantizar o mesmo vetor de entrada. Desta forma, são executadas de duas a quatro TTCQs, cada uma delas utilizando uma das configurações de rótulos da figura 4.5c, e aquela que produzir a menor distorção é selecionada. Uma vantagem deste método é reduzir o impacto da escolha do estado inicial na quantização do início da seqüência. Com isto, CHAPPELIER *et al.* [23] conseguiram diminuir um pouco mais a distorção média em relação ao algoritmo serial descrito no primeiro parágrafo da subseção anterior. Uma desvantagem é a necessidade de adicionar um ou dois *bits* para informar qual disposição de rótulos foi utilizada.

Entretanto, há uma forma mais elegante de se gerar formas distintas de quantizar o vetor de entrada que evita a necessidade de *bits* extras de informação: *o*

*algoritmo paralelo*. Foram utilizadas aqui as idéias sobre codificação paralela apresentadas na seção 2.5. Visto que cada codificador da TTCQ processa metades disjuntas das amostras produzidas pela fonte, é possível codificá-las utilizando as seqüências $A \rightarrow B \rightarrow A \rightarrow B \rightarrow \cdots$ e $B \rightarrow A \rightarrow B \rightarrow A \rightarrow \cdots$ simultaneamente, ou seja, a cada iteração $i$ os codificadores $A$ e $B$ podem operar em paralelo.

Como se pode ver na figura 4.6, cada seqüência de codificação deve ser iniciada de forma distinta, já que o codificador $A$ processa as amostras de índice par do vetor $\mathbf{a}$ e o codificador $B$, as de índice ímpar permutadas. Com isto, para cada iteração $i$, $\mathbf{u}_A^{(i)}$ não é necessariamente igual a $\mathbf{u}_B^{(i)}$. O objetivo é obter uma distorção média menor que a do algoritmo serial refinado, selecionando a melhor seqüência de codificação para cada vetor de entrada $\mathbf{a}$. Note que este algoritmo pode ser facilmente combinado com o algoritmo de CHAPPELIER *et al.* [23], que muda os rótulos da treliça, sem adicionar mais *bits* extras de informação. Então, dados $N_{it}$, $N_{\mathbf{u}}$ e $\epsilon \geq 0$, o algoritmo paralelo é constituído pelos seguintes passos:

1. $n_{\mathbf{u}} = n_{\mathbf{u}_A} = n_{\mathbf{u}_B} = 0$;
   $i_{min} = 1$;
   $D_{min} = \infty$;
   $\mathbf{u}_A^{(0)} = \mathbf{u}_B^{(0)} = \mathbf{0}$;
   $\mathbf{L}_1^{a(0)} = \mathbf{L}_A$;
   $\mathbf{L}_2^{a(0)} = \mathbf{L}_B$;
   $i = 1$;

2. // Codificar $\mathbf{a}$ com o codificador $A$.
   $\mathbf{L}_{12}^{es(i)} = \mathbf{L}_1^{(i)} - \mathbf{L}_1^{a(i-1)}$;
   $\mathbf{u}_A^{(i)} = h(\max\{\mathbf{L}_1^{(i)}\})$;

   // Codificar $\mathbf{a}$ com o codificador $B$.
   $\mathbf{L}_{21}^{es(i)} = \mathbf{L}_2^{(i)} - \mathbf{L}_2^{a(i-1)}$;
   $\mathbf{u}_B^{(i)} = h(\max\{\mathbf{P}^{-1}\mathbf{L}_2^{(i)}\})$;

3. $\mathbf{L}_1^{a(i)} = \mathbf{P}^{-1}\mathbf{L}_{21}^{es(i)}$;
   $\mathbf{L}_2^{a(i)} = \mathbf{P}\mathbf{L}_{12}^{es(i)}$;

4. Se ( $\mathbf{u}_A^{(i)} == \mathbf{u}_A^{(i-1)}$ )
   {

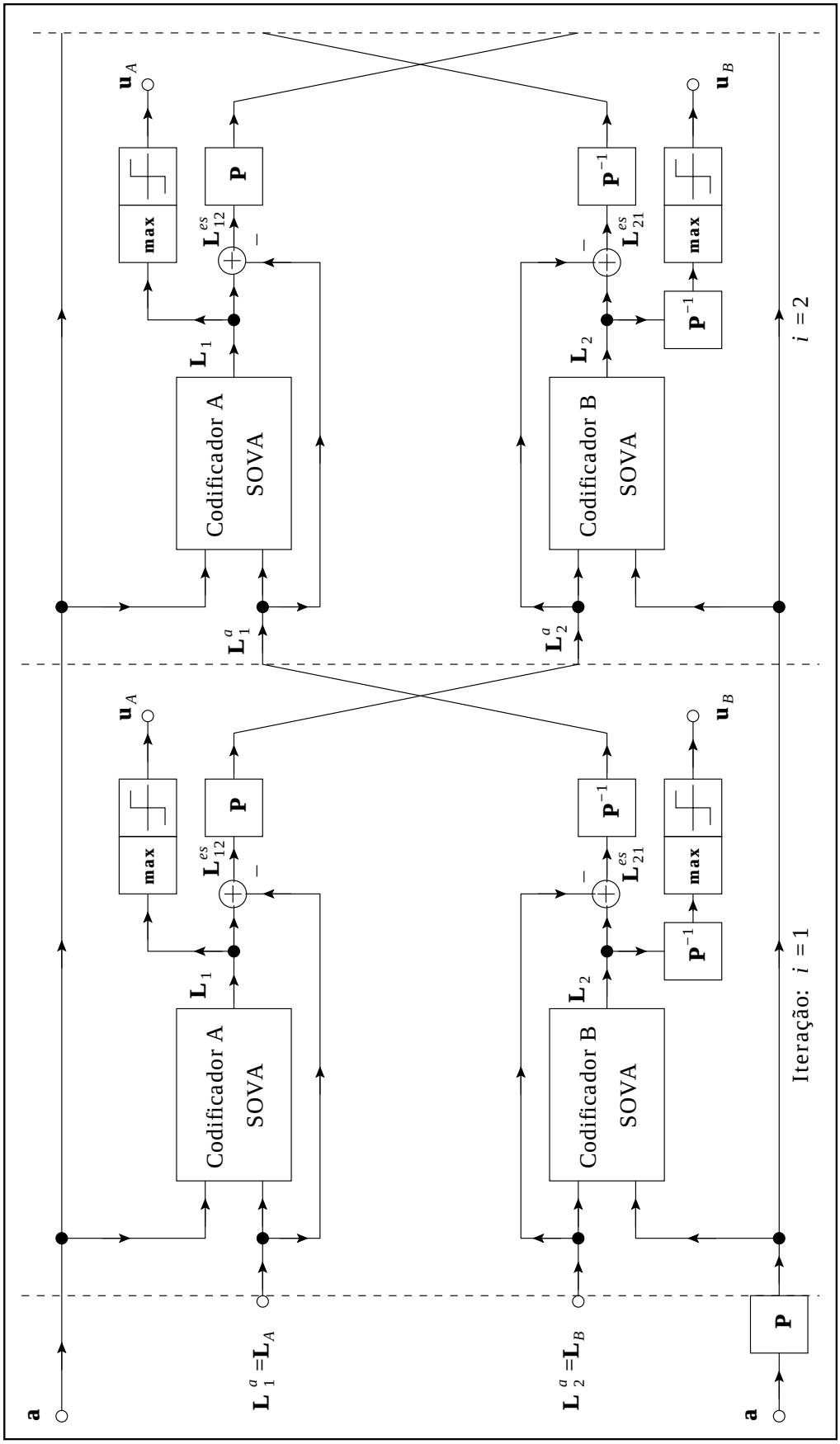


Figura 4.6: Turbo TCQ em modo paralelo.

Se ( $i > 1$ ) Ir para 6;

}

Senão

{

$n_{\mathbf{u}_A} = 0$;

}

5. $\mathbf{w}_A^{(i)} = g(\mathbf{u}_A^{(i)})$;
   $D_A = \frac{1}{N} \sum_{k=0}^{N-1} \left[ a_k - w_{A,k}^{(i)} \right]^2$;

6. Se ( $\mathbf{u}_B^{(i)} == \mathbf{u}_B^{(i-1)}$ )

   {

   Se ( $i > 1$ ) Ir para 8;

   }

   Senão

   {

   $n_{\mathbf{u}_B} = 0$;

   }

7. $\mathbf{w}_B^{(i)} = g(\mathbf{u}_B^{(i)})$;
   $D_B = \frac{1}{N} \sum_{k=0}^{N-1} \left[ a_k - w_{B,k}^{(i)} \right]^2$;

8. Se (( $n_{\mathbf{u}_A} \neq 0$ ) e ( $n_{\mathbf{u}_B} \neq 0$ ))

   {

   Ir para 10;

   }

9. Se ( $D_A \leq D_B$ )

   {

   $cod = 0$;

   Se ( $D_A < D_{min}$ )

   {

   $i_{min} = i$;

   $D_{min} = D_A$;

   $\mathbf{u}_{min} = \mathbf{u}_A^{(i)}$;

}

}

Senão

{

$cod = 1$;

Se ( $D_B < D_{min}$ )

{

$i_{min} = i$;

$D_{min} = D_B$;

$\mathbf{u}_{min} = \mathbf{u}_B^{(i)}$;

}

}

10. $n_{\mathbf{u}_A} = n_{\mathbf{u}_A} + 1$;

$n_{\mathbf{u}_B} = n_{\mathbf{u}_B} + 1$;

11. Se ( $cod == 0$ )

{

$n_{\mathbf{u}} = n_{\mathbf{u}_A}$;

$\Delta\mathbf{L}_{12}^{es(i)} = \mathbf{L}_{12}^{es(i)} - \mathbf{P}^{-1}\mathbf{L}_2^{a(i-1)}$;

$\Delta_{max}^{es} = \max(\text{abs}(\Delta\mathbf{L}_{12}^{es(i)}))$;

}

Senão

{

$n_{\mathbf{u}} = n_{\mathbf{u}_B}$;

$\Delta\mathbf{L}_{21}^{es(i)} = \mathbf{L}_{21}^{es(i)} - \mathbf{P}\mathbf{L}_1^{a(i-1)}$;

$\Delta_{max}^{es} = \max(\text{abs}(\Delta\mathbf{L}_{21}^{es(i)}))$;

}

12. Se ((( $n_{\mathbf{u}} \geq N_{\mathbf{u}}$ ) e ( $\Delta_{max}^{es} \leq \epsilon$ )) ou ( $\Delta_{max}^{es} == 0$ ) ou ( $i == N_{it}$ ))

{

$\mathbf{u} = \mathbf{u}_{min}$;

Parar;

}

13. $i = i + 1$;

    Ir para 2;

### 4.2.4 Resumo dos Algoritmos

A seguir são destacadas as principais características dos algoritmos apresentados nas subseções anteriores. A função deste resumo é tornar mais claras as diferenças entre estes algoritmos e, se for o caso, destacar também as contribuições deste trabalho.

#### Algoritmo Serial

- Executa $N_{it}$ iterações e retorna $\mathbf{u} = h(\max\{\mathbf{P}^{-1}\mathbf{L}_2^{(N_{it})}\})$. É o mesmo algoritmo utilizado no decodificador da TTCM.

#### Algoritmo Serial de Chappelier

- O algoritmo de CHAPPELIER *et al.* [23] conta o número de iterações seguidas ($n_{\mathbf{u}}$) durante as quais o vetor $\mathbf{u}^{(i)}$ se mantém inalterado. Armazena $\mathbf{u}_{min}$, o vetor que produziu a menor distorção no decorrer das $N_{it}$ iterações.

- Executa $N_{it}$ iterações.

- Se $n_{\mathbf{u}} \geq 50$, retorna $\mathbf{u} = \mathbf{u}^{(N_{it})}$. Caso contrário, retorna $\mathbf{u} = \mathbf{u}_{min}$.

#### Algoritmo Serial Refinado

- Conta o número de iterações seguidas ($n_{\mathbf{u}}$) durante as quais o vetor $\mathbf{u}^{(i)}$ se mantém inalterado. Armazena $\mathbf{u}_{min}$, o vetor que produziu a menor distorção no decorrer das iterações executadas (neste algoritmo, o número de iterações executadas $i_{max}$ pode ser menor que $N_{it}$).

- Calcula $\Delta_{max}^{es} = \max(\text{abs}(\Delta\mathbf{L}_{21}^{es(i)}))$ para cada iteração.

- Se $n_{\mathbf{u}} \geq 50$ e $\Delta_{max}^{es} \leq \epsilon$ ou se $\Delta_{max}^{es}$ é igual a 0 (zero) ou se o número máximo de iterações foi atingido o algoritmo pará o processo iterativo, pois considera ter encontrado um ponto fixo do processo de codificação para o vetor de fonte $\mathbf{a}$ dado, e retorna $\mathbf{u} = \mathbf{u}_{min}$. Caso contrário, executa mais uma iteração.

Com este critério de parada baseado no valor de $\Delta_{max}^{es} = \max(\text{abs}(\Delta\mathbf{L}_{21}^{es(i)}))$, o número médio de iterações executado ($\mu_{I_{max}}$) poderá ser menor que $N_{it}$ sem reduzir o valor da distorção média obtida. Isto não foi considerado por CHAPPELIER *et al.* [23]. O benfício deste critério de parada é, portanto, possibilitar a redução o tempo de codificação da TTCQ e foi proposto neste trabalho.

Quando o critério de parada é satisfeito sempre é retornado $\mathbf{u} = \mathbf{u}_{min}$. De forma distinta do que é feito por CHAPPELIER *et al.* [23], não se assume que $n_{\mathbf{u}} \geq N_{\mathbf{u}} = 50$ implica $\mathbf{u}_{min} = \mathbf{u}^{(i_{max} \leq N_{it})}$. O benfício desta modificação, também introduzida neste trabalho, é possibilitar a redução da distorção média da TTCQ.

**Algoritmo Paralelo**

- Executa dois processos distintos de codificação serial refinada (seqüências de codificação $A \to B \to A \to B \to \cdots$ e $B \to A \to B \to A \to \cdots$) em paralelo e retorna $\mathbf{u}_{min}$ do processo que produzir a menor distorção.

  O algoritmo de codificação paralelo para TTCQ é uma proposta original deste trabalho, visando à redução do tempo de codificação e da distorção média produzidos pelo algoritmo serial refinado.

## 4.3 Resultados

### 4.3.1 Resultados do Algoritmo Serial

Nesta subseção, serão apresentados os resultados obtidos pela TTCQ na quantização de fonte gaussiana $\mathcal{N}(0,1)$ sem memória com $R$ *bits* para $R = 1, 2, 3, 4$. Na tabela 4.1, são mostrados os resultados obtidos pela TCQ de MARCELLIN [6] utilizando os níveis de reconstrução do quantizador de Lloyd-Max de taxa $R + 1$ [20]. Na tabela 4.2, são apresentados os resultados da TCQ de MARCELLIN [6] utilizando níveis de reconstrução otimizados[3]. Estas duas tabelas serão utilizadas para comparação com os resultados obtidos pela TTCQ. Cada valor experimental

[3]Ver algoritmo do apêndice A, seção 1.

Tabela 4.1: Resultados da TCQ-HOVA utilizando fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| **Estados** | **1** | **2** | **3** | **4** |
| **4** | 4,65 ± 0,03 | 10,19 ± 0,03 | 15,83 ± 0,04 | 21,61 ± 0,05 |
| **8** | 4,79 ± 0,03 | 10,31 ± 0,03 | 15,93 ± 0,04 | 21,72 ± 0,05 |
| **16** | 4,87 ± 0,03 | 10,35 ± 0,03 | 15,99 ± 0,03 | 21,79 ± 0,05 |
| **32** | 4,94 ± 0,03 | 10,41 ± 0,03 | 16,07 ± 0,04 | 21,86 ± 0,05 |
| **64** | 5,00 ± 0,03 | 10,49 ± 0,03 | 16,12 ± 0,03 | 21,91 ± 0,05 |
| **128** | 5,05 ± 0,03 | 10,54 ± 0,03 | 16,18 ± 0,04 | 21,96 ± 0,05 |
| **256** | 5,09 ± 0,03 | 10,58 ± 0,03 | 16,21 ± 0,04 | 22,00 ± 0,05 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

listado nas tabelas referentes à TCQ e à TTCQ é o resultado do cálculo da média oriunda da simulação de $N_s = 100$ seqüências de comprimento $N$ igual a 1000 de realizações independentes de um gerador de distribuição gaussiana. O intervalo de confiança[4] de 95% foi calculado e incluído nestas tabelas também. Os valores de relação sinal-ruído (SNR – *Signal to Noise Ratio*) contidos nestas tabelas estão listados em dB ($10\log_{10}(1/D)$). As últimas duas linhas destas tabelas correspondem aos resultados obtidos pelo quantizador de Lloyd-Max [20] e o limite superior teórico que pode ser obtido para uma fonte gaussiana.

Nas tabelas 4.3 e 4.4, são mostrados os resultados obtidos pela TTCQ-BCJR utilizando uma iteração e quatro iterações, respectivamente. Foram utilizados os códigos convolucionais de UNGERBOECK [22] e os níveis de reconstrução do quantizador de Lloyd-Max de taxa $R+1$. Claramente, o desempenho do quantizador proposto é bastante inferior até mesmo ao de Lloyd-Max e, independente do número de iterações utilizado. Verificou-se, posteriormente, que o algoritmo BCJR quando utilizado na TCQ também não produz resultados satisfatórios. Então, decidiu-se

[4]Ver apêndice B, seção 2.

Tabela 4.2: Resultados da TCQ-HOVA utilizando fonte gaussiana sem memória e pontos de reconstrução otimizados. Os valores de SNR estão listados em dB.

| **Número de Estados** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **4** | 5,03 ± 0,05 | 10,56 ± 0,05 | 16,18 ± 0,07 | 21,95 ± 0,09 |
| **8** | 5,22 ± 0,04 | 10,69 ± 0,05 | 16,33 ± 0,07 | 22,06 ± 0,09 |
| **16** | 5,29 ± 0,05 | 10,77 ± 0,05 | 16,39 ± 0,07 | 22,13 ± 0,09 |
| **32** | 5,35 ± 0,04 | 10,84 ± 0,05 | 16,46 ± 0,07 | 22,16 ± 0,09 |
| **64** | 5,44 ± 0,04 | 10,92 ± 0,05 | 16,53 ± 0,07 | 22,28 ± 0,09 |
| **128** | 5,51 ± 0,04 | 10,96 ± 0,05 | 16,58 ± 0,07 | 22,35 ± 0,09 |
| **256** | 5,54 ± 0,04 | 11,01 ± 0,05 | 16,63 ± 0,07 | 22,40 ± 0,10 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

utilizar na turbo quantização o SOVA que funciona com a TCQ e pode ser empregado num esquema que utiliza o princípio turbo. Os resultados apresentados na tabela 4.5 evidenciam uma melhora significativa de desempenho em relação àqueles das tabelas 4.3 e 4.4, mas ainda inferiores aos resultados da TCQ e aos do quantizador de Lloyd-Max. Comparando as tabelas 4.5 e 4.6, conclui-se que a utilização de mais de uma iteração foi prejudicial para a TTCQ-SOVA neste caso.

Continuando com as experiências com a TTCQ-SOVA, verificou-se que os códigos convolucionais de UNGERBOECK [22] não produziam os melhores resultados quando utilizados no codificador de canal empregando turbo modulação (TTCM-SOVA), e conjeturou-se que isto poderia estar causando os péssimos resultados obtidos. Através de uma procura por códigos convolucionais de quatro estados melhores, concluiu-se experimentalmente que o código $(7, 2)$ produz melhores resultados que o código $(5, 2)$ de UNGERBOECK [22] quando utilizado na estrutura de decodificação do turbo modulador. A diferença de desempenho entre estes dois códigos é mostrada na figura 4.7, onde são comparados os seus gráficos de taxa de erro de *bits* (BER – *Bit Error Rate*) em função da SNR de canal. Então, com mais esta mudança, foram obtidos os resultados da tabela 4.7. São um pouco melhores que os

Tabela 4.3: Resultados da TTCQ-BCJR utilizando uma iteração, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de Estados** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **4** | 1,65 ± 0,04 | 6,67 ± 0,04 | 12,04 ± 0,05 | 17,68 ± 0,06 |
| **8** | 1,73 ± 0,04 | 6,67 ± 0,04 | 12,06 ± 0,05 | 17,74 ± 0,06 |
| **16** | 1,73 ± 0,04 | 6,68 ± 0,04 | 12,08 ± 0,04 | 17,71 ± 0,05 |
| **32** | 1,71 ± 0,04 | 6,68 ± 0,04 | 12,03 ± 0,05 | 17,70 ± 0,06 |
| **64** | 1,70 ± 0,04 | 6,70 ± 0,04 | 12,05 ± 0,05 | 17,72 ± 0,06 |
| **128** | 1,73 ± 0,03 | 6,68 ± 0,04 | 12,06 ± 0,05 | 17,74 ± 0,06 |
| **256** | 1,70 ± 0,04 | 6,67 ± 0,04 | 12,06 ± 0,05 | 17,67 ± 0,07 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.4: Resultados da TTCQ-BCJR utilizando quatro iterações, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de Estados** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **4** | 1,59 ± 0,04 | 6,68 ± 0,04 | 12,04 ± 0,05 | 17,68 ± 0,06 |
| **8** | 1,71 ± 0,04 | 6,66 ± 0,04 | 12,06 ± 0,05 | 17,74 ± 0,06 |
| **16** | 1,71 ± 0,04 | 6,68 ± 0,04 | 12,08 ± 0,04 | 17,71 ± 0,05 |
| **32** | 1,70 ± 0,03 | 6,68 ± 0,04 | 12,03 ± 0,05 | 17,70 ± 0,06 |
| **64** | 1,72 ± 0,04 | 6,70 ± 0,04 | 12,05 ± 0,05 | 17,72 ± 0,06 |
| **128** | 1,73 ± 0,03 | 6,68 ± 0,04 | 12,06 ± 0,05 | 17,74 ± 0,06 |
| **256** | 1,72 ± 0,04 | 6,67 ± 0,04 | 12,06 ± 0,05 | 17,67 ± 0,07 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.5: Resultados da TTCQ-SOVA utilizando uma iteração, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| **Estados** | **1** | **2** | **3** | **4** |
| **4** | 2,79 ± 0,04 | 7,84 ± 0,04 | 13,31 ± 0,05 | 19,04 ± 0,06 |
| **8** | 3,03 ± 0,04 | 8,06 ± 0,04 | 13,53 ± 0,05 | 19,26 ± 0,07 |
| **16** | 3,00 ± 0,04 | 8,06 ± 0,04 | 13,49 ± 0,05 | 19,25 ± 0,07 |
| **32** | 3,15 ± 0,04 | 8,13 ± 0,04 | 13,62 ± 0,06 | 19,32 ± 0,07 |
| **64** | 3,10 ± 0,04 | 8,15 ± 0,04 | 13,61 ± 0,05 | 19,31 ± 0,07 |
| **128** | 3,13 ± 0,04 | 8,15 ± 0,04 | 13,63 ± 0,05 | 19,38 ± 0,07 |
| **256** | 3,18 ± 0,04 | 8,17 ± 0,04 | 13,62 ± 0,06 | 19,31 ± 0,07 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.6: Resultados da TTCQ-SOVA utilizando quatro iterações, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| **Estados** | **1** | **2** | **3** | **4** |
| **4** | 2,22 ± 0,06 | 7,26 ± 0,06 | 12,76 ± 0,06 | 18,40 ± 0,10 |
| **8** | 2,46 ± 0,04 | 7,47 ± 0,05 | 12,97 ± 0,05 | 18,70 ± 0,07 |
| **16** | 2,46 ± 0,04 | 7,50 ± 0,05 | 13,00 ± 0,05 | 18,80 ± 0,06 |
| **32** | 2,55 ± 0,04 | 7,55 ± 0,04 | 13,05 ± 0,05 | 18,84 ± 0,07 |
| **64** | 2,55 ± 0,05 | 7,57 ± 0,05 | 13,10 ± 0,05 | 18,82 ± 0,07 |
| **128** | 2,57 ± 0,04 | 7,60 ± 0,05 | 13,12 ± 0,05 | 18,91 ± 0,07 |
| **256** | 2,59 ± 0,04 | 7,61 ± 0,05 | 13,13 ± 0,06 | 18,86 ± 0,06 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.7: Resultados da TTCQ-SOVA utilizando fonte gaussiana sem memória, os pontos de reconstrução de Lloyd-Max de taxa $R + 1$ e o código convolucional $(7, 2)$. Os valores de SNR estão listados em dB.

| **Número de Iterações** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **1** | 3,00 ± 0,04 | 8,02 ± 0,04 | 13,49 ± 0,05 | 19,21 ± 0,07 |
| **4** | 2,41 ± 0,05 | 7,43 ± 0,05 | 12,92 ± 0,05 | 18,65 ± 0,06 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

resultados correspondentes das tabelas 4.5 e 4.6, mas ainda muito ruins. Nota-se, também, que utilizar mais de uma iteração produz resultados ainda piores. Desta forma, a conjetura não se mostrou válida como era esperado.

Efetuando uma análise de convergência, RICHARDSON [24] prova que os códigos turbo sempre possuem pontos fixos, fornecendo as condições teóricas suficientes para a unicidade dos mesmos. Ele comenta que a unicidade do ponto fixo deve, provavelmente, ocorrer com regularidade. Entretanto, ele diz que um problema ainda não resolvido é o fator limitante do desempenho dos códigos turbo, isto é, baixa relação sinal-ruído. A medida que o desempenho do decodificador turbo se degrada nesta região de baixa relação sinal-ruído, a estabilidade do ponto fixo diminui. Desta forma, a quebra no desempenho dos códigos turbo é, provavelmente, uma falha na convergência. Embora o decodificador tenha pontos fixos estáveis, o algoritmo pode iniciar muito longe da região de convergência e não conseguir atingi-la mesmo após várias iterações. Outro fator que também poderia afetar o desempenho do turbo decodificador seria a existência de pontos fixos múltiplos. No caso da multiplicidade de pontos fixos, haveria a possibilidade de poder distingui-los e encontrar aquele que produzisse o melhor desempenho. Então, duas hipóteses podem ser formuladas:

1. o algoritmo convergiu em uma única iteração, mas para um ponto fixo que não produz o melhor desempenho;

2. o algoritmo de decodificação não convergiu.

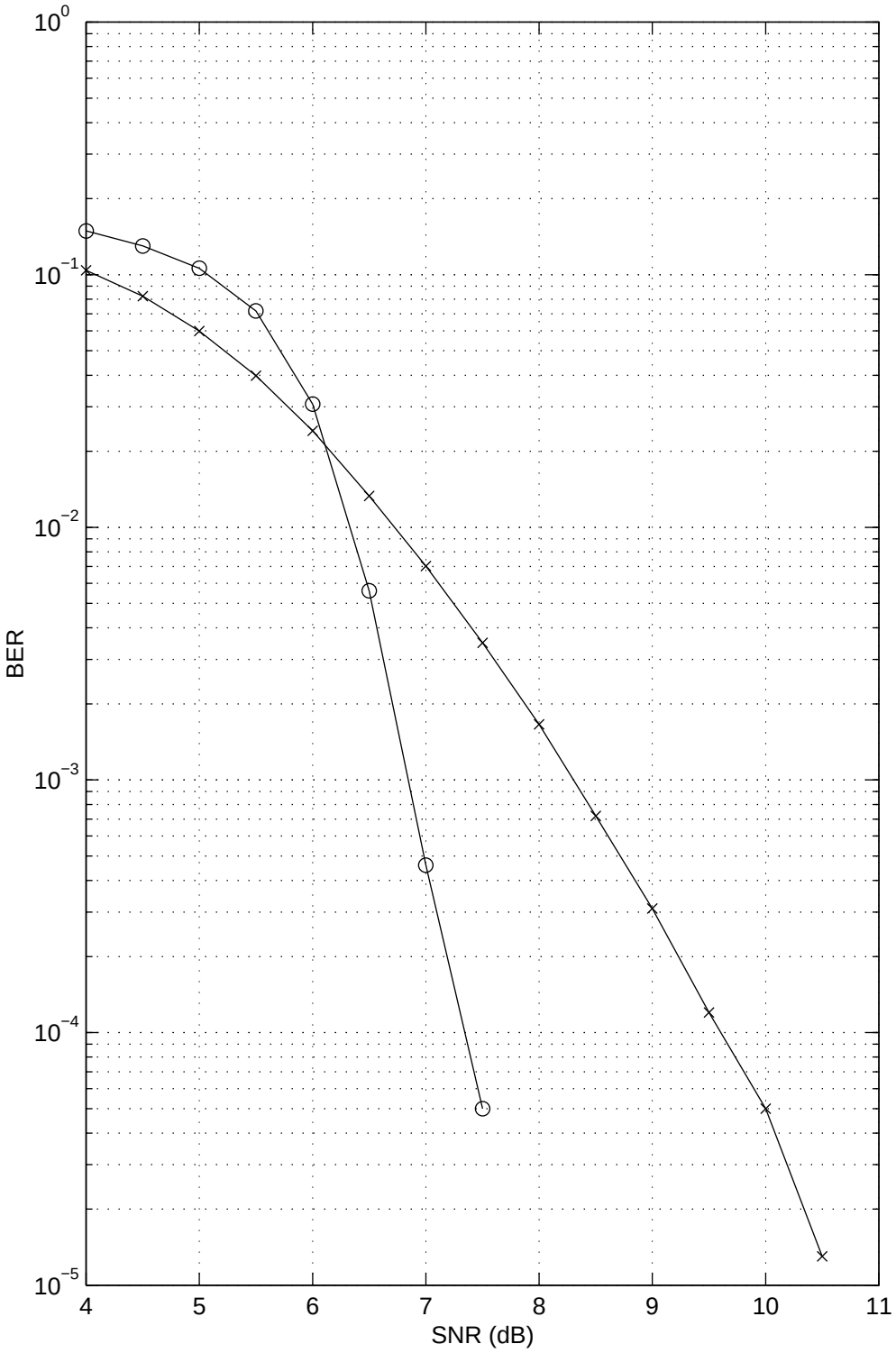


Figura 4.7: TTCM-SOVA com $R = 1$ *bit* por símbolo. Comparação entre os códigos convolucionais ($\circ$) $(7, 2)$ e ($\times$) $(5, 2)$.

Testes experimentais mostram que o algoritmo não está convergindo, pois quando isto ocorre, ver figura 4.3, $\mathbf{L}_2^{(i)} \approx \mathbf{PL}_1^{(i)}$ e $\mathbf{L}_{21}^{es\,(i)} \approx \mathbf{L}_{21}^{es\,(i-1)}$, para alguma iteração $i = 1, 2, \ldots$, visto que $\mathbf{L}_{21}^{es\,(i)} = \mathbf{L}_2^{(i)} - \mathbf{PL}_1^{(i)} + \mathbf{L}_{21}^{es\,(i-1)}$, e isto não é verificado, mesmo utilizando centenas de iterações. De fato, ocorreu divergência em praticamente todos os testes efetuados quando se utilizou mais de uma iteração, já que resultados se tornaram ainda piores. Observando-se a figura 4.7, nota-se que a TTCM-SOVA para o código $(7, 2)$ possui uma região relativamente plana, onde a BER se altera pouco com o aumento da SNR de canal, e uma região onde a BER se altera bruscamente com o aumento da SNR de canal. É nesta última região que o algoritimo iterativo da TTCM-SOVA converge. Infelizmente, esta região está além da SNR teórica máxima que um quantizador para fonte gaussiana pode atingir. Portanto, por um problema de convergência, que é inerente ao algoritmo de decodificação dos códigos turbo, o melhor código para TTCM, ou seja, para codificação de canal, não pode ser utilizado como código taxa $\times$ distorção. Porém, conjetura-se que se, de alguma forma, o problema de convergência pudesse ser resolvido ou atenuado, o código $(7, 2)$ poderia ser utilizado na TTCQ e produzir resultados melhores.

Contudo, analisando o desempenho da TCQ, da TTCQ, da TCM e da TTCM nota-se que a TCQ está resolvendo melhor o problema da cobertura que a TTCQ. Por outro lado, a TTCM resolve melhor o problema do empacotamento que a TCM. Visto que os problemas de cobertura e empacotamento não possuem, em geral, a mesma solução para uma dada dimensão $N$, pode-se conjeturar que os códigos convolucionais que produzem o melhor desempenho para a TTCM não serão, em princípio, aqueles que produzirão o melhor desempenho para a TTCQ. Embora tanto a TCM quanto a TCQ utilizem os mesmos códigos convolucionais, a TCM não corresponde ao melhor codificador de canal. No gráfico da figura 4.8, pode-se ver que a TTCM é claramente mais eficiente na codificação de canal que a TCM. Desta forma, uma busca por outros códigos para a TTCQ pode acarretar uma melhora nos resultados até então obtidos.

Realizando, então, uma busca por novos códigos foi constatado experimentalmente que os melhores códigos convolucionais para TTCQ são os mostrados na tabela 4.8. Utilizando os novos códigos e os níveis de reconstrução de Lloyd-Max foram obtidos os resultados das tabelas 4.9 e 4.10. Agora os resultados já superam

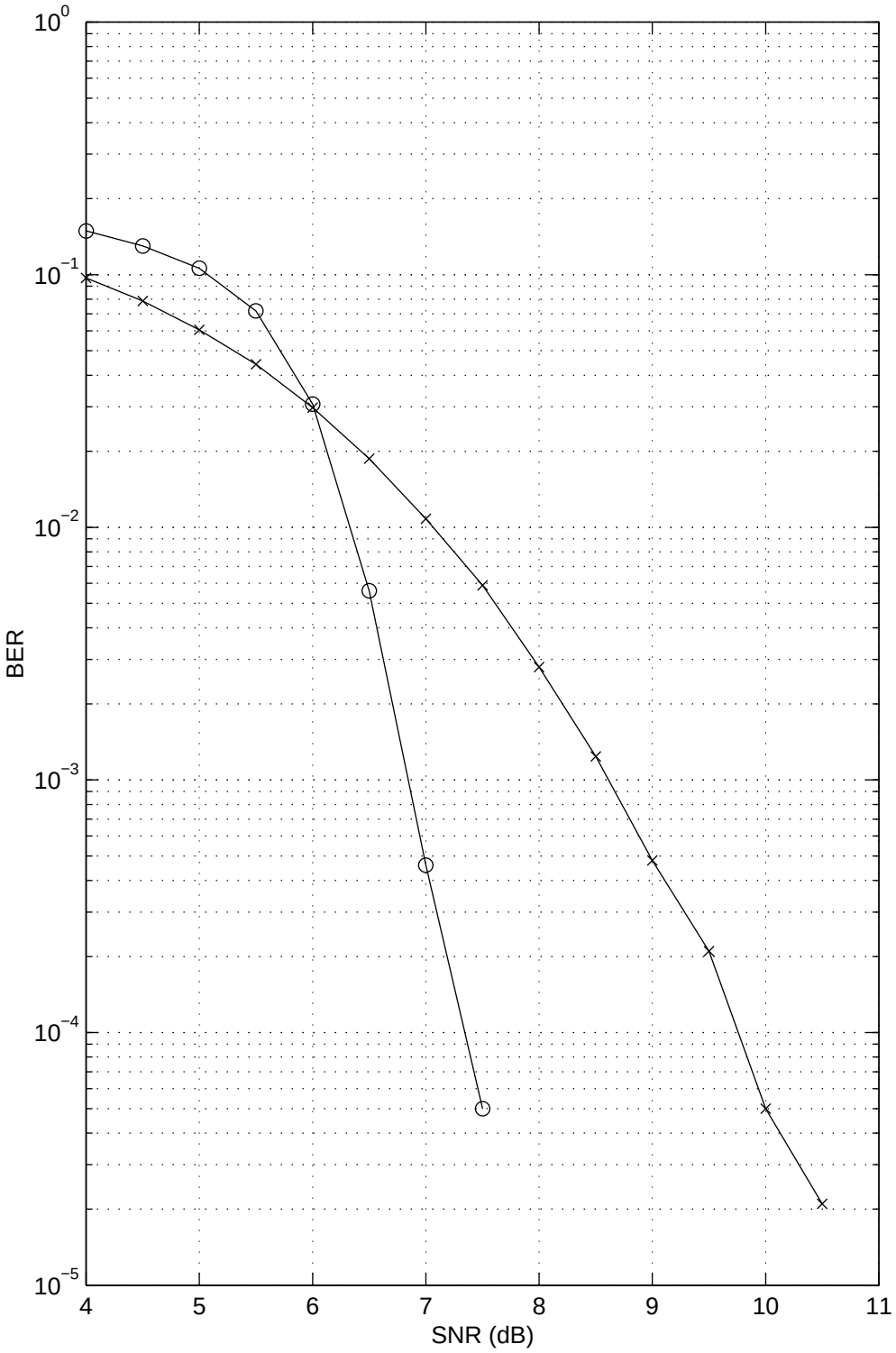


Figura 4.8: Comparação entre (×) TCM com o código $(5, 2)$ e (∘) TTCM-SOVA com o código $(7, 2)$ para 1 *bit* por símbolo.

Tabela 4.8: Códigos utilizados na TCQ e os códigos que produzem os melhores resultados para TTCQ-SOVA para fonte gaussiana sem memória com os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Códigos em octal.

| **Número de Estados** | **Códigos para TCQ** | **Códigos para Turbo TCQ** |
|---|---|---|
| **4** | (005, 002) | (005, 004) |
| **8** | (013, 004) | (017, 014) |
| **16** | (023, 004) | (025, 004) |
| **32** | (045, 010) | (077, 014) |
| **64** | (103, 024) | (125, 004) |
| **128** | (235, 126) | (377, 014) |
| **256** | (515, 362) | (405, 124) |

os de Lloyd-Max, exceto pelos dois primeiros valores da coluna de 1 *bit* por amostra. Como se pode verificar pela tabela 4.10, a utilização de mais de uma iteração não afeta os resultados. De fato, constatou-se que o algoritmo da TTCQ-SOVA se estabiliza após uma iteração com estes novos códigos. Curiosamente, observa-se que os resultados das linhas de 16, 64 e 256 estados da tabela 4.9 são semelhantes aos obtidos para TCQ na tabela 4.1 nas linhas de 4, 8 e 16 estados. Ou seja, parece que a TTCQ-SOVA necessita do quadrado do número de estados para produzir um desempenho semelhante ao da TCQ. Nas tabelas 4.11 e 4.12, tem-se o resultado da aplicação dos novos códigos na TTCQ-BCJR. Neste caso, os resultados são muito ruins.

Porém, ainda é possível otimizar os níveis de reconstrução utilizados na TTCQ-SOVA. A tabela 4.13 foi obtida efetuando-se mais este passo. Comparando com a tabela 4.2, onde se encontram os resultados da TCQ com níveis de reconstrução também otimizados, constata-se, novamente, que a TTCQ-SOVA necessita do quadrado do número de estados para produzir um desempenho correspondente ao da TCQ. Na tabela 4.14, são apresentados os resultados da TTCQ-BCJR, utilizando os melhores códigos da TTCQ-SOVA e níveis de reconstrução otimizados. Pela tabela 4.14, vê-se claramente que a TTCQ-BCJR se degenera num quantizador de

Tabela 4.9: Resultados da TTCQ-SOVA utilizando uma iteração, códigos para TTCQ da tabela 4.8, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de Estados** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **4** | 4,06 ± 0,03 | 9,51 ± 0,03 | 15,13 ± 0,03 | 20,93 ± 0,04 |
| **8** | 4,06 ± 0,03 | 9,51 ± 0,03 | 15,13 ± 0,03 | 20,93 ± 0,04 |
| **16** | 4,67 ± 0,03 | 10,18 ± 0,03 | 15,82 ± 0,03 | 21,61 ± 0,05 |
| **32** | 4,67 ± 0,03 | 10,18 ± 0,03 | 15,82 ± 0,03 | 21,61 ± 0,05 |
| **64** | 4,79 ± 0,03 | 10,30 ± 0,03 | 15,93 ± 0,03 | 21,71 ± 0,05 |
| **128** | 4,79 ± 0,03 | 10,30 ± 0,03 | 15,93 ± 0,03 | 21,71 ± 0,05 |
| **256** | 4,86 ± 0,03 | 10,36 ± 0,03 | 15,99 ± 0,03 | 21,79 ± 0,05 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.10: Resultados da TTCQ-SOVA utilizando quatro iterações, códigos para TTCQ da tabela 4.8, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de Estados** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **4** | 4,06 ± 0,03 | 9,51 ± 0,03 | 15,13 ± 0,03 | 20,93 ± 0,04 |
| **8** | 4,06 ± 0,03 | 9,51 ± 0,03 | 15,13 ± 0,03 | 20,93 ± 0,04 |
| **16** | 4,67 ± 0,03 | 10,18 ± 0,03 | 15,82 ± 0,03 | 21,61 ± 0,05 |
| **32** | 4,67 ± 0,03 | 10,18 ± 0,03 | 15,82 ± 0,03 | 21,61 ± 0,05 |
| **64** | 4,79 ± 0,03 | 10,30 ± 0,03 | 15,93 ± 0,03 | 21,71 ± 0,05 |
| **128** | 4,79 ± 0,03 | 10,30 ± 0,03 | 15,93 ± 0,03 | 21,71 ± 0,05 |
| **256** | 4,86 ± 0,03 | 10,36 ± 0,03 | 15,99 ± 0,03 | 21,79 ± 0,05 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.11: Resultados da TTCQ-BCJR utilizando uma iteração, códigos para TTCQ da tabela 4.8, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| **Estados** | **1** | **2** | **3** | **4** |
| **4** | 1,38 ± 0,05 | 6,67 ± 0,04 | 12,07 ± 0,05 | 17,74 ± 0,06 |
| **8** | 1,38 ± 0,05 | 6,67 ± 0,04 | 12,07 ± 0,05 | 17,74 ± 0,06 |
| **16** | 1,66 ± 0,04 | 6,69 ± 0,04 | 12,08 ± 0,04 | 17,72 ± 0,06 |
| **32** | 1,66 ± 0,04 | 6,69 ± 0,04 | 12,08 ± 0,04 | 17,72 ± 0,06 |
| **64** | 1,68 ± 0,04 | 6,67 ± 0,04 | 12,09 ± 0,05 | 17,71 ± 0,06 |
| **128** | 1,68 ± 0,04 | 6,67 ± 0,04 | 12,09 ± 0,05 | 17,71 ± 0,06 |
| **256** | 1,72 ± 0,04 | 6,68 ± 0,04 | 12,03 ± 0,05 | 17,73 ± 0,05 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.12: Resultados da TTCQ-BCJR utilizando quatro iterações, códigos para TTCQ da tabela 4.8, fonte gaussiana sem memória e os pontos de reconstrução de Lloyd-Max de taxa $R+1$. Os valores de SNR estão listados em dB.

| **Número de** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| **Estados** | **1** | **2** | **3** | **4** |
| **4** | 1,38 ± 0,05 | 6,67 ± 0,04 | 12,07 ± 0,05 | 17,74 ± 0,06 |
| **8** | 1,38 ± 0,05 | 6,67 ± 0,04 | 12,07 ± 0,05 | 17,74 ± 0,06 |
| **16** | 1,66 ± 0,04 | 6,69 ± 0,04 | 12,08 ± 0,04 | 17,72 ± 0,06 |
| **32** | 1,66 ± 0,04 | 6,69 ± 0,04 | 12,08 ± 0,04 | 17,72 ± 0,06 |
| **64** | 1,68 ± 0,04 | 6,67 ± 0,04 | 12,09 ± 0,05 | 17,71 ± 0,06 |
| **128** | 1,68 ± 0,04 | 6,67 ± 0,04 | 12,09 ± 0,05 | 17,71 ± 0,06 |
| **256** | 1,72 ± 0,04 | 6,68 ± 0,04 | 12,03 ± 0,05 | 17,73 ± 0,05 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Tabela 4.13: Resultados da TTCQ-SOVA utilizando uma iteração, códigos para TTCQ da tabela 4.8, fonte gaussiana sem memória e pontos de reconstrução otimizados. Os valores de SNR estão listados em dB.

| **Número de Estados** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| | **1** | **2** | **3** | **4** |
| **4** | 4,71 ± 0,05 | 9,99 ± 0,06 | 15,63 ± 0,07 | 21,40 ± 0,09 |
| **8** | 4,71 ± 0,05 | 9,99 ± 0,06 | 15,63 ± 0,07 | 21,40 ± 0,09 |
| **16** | 5,08 ± 0,05 | 10,55 ± 0,05 | 16,18 ± 0,07 | 21,94 ± 0,09 |
| **32** | 5,08 ± 0,05 | 10,55 ± 0,05 | 16,18 ± 0,07 | 21,94 ± 0,09 |
| **64** | 5,19 ± 0,05 | 10,69 ± 0,05 | 16,31 ± 0,06 | 22,07 ± 0,09 |
| **128** | 5,19 ± 0,05 | 10,69 ± 0,05 | 16,31 ± 0,06 | 22,07 ± 0,09 |
| **256** | 5,29 ± 0,04 | 10,79 ± 0,05 | 16,41 ± 0,07 | 22,14 ± 0,09 |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

Lloyd-Max.

Observando as tabelas 4.9 e 4.13, nota-se que os resultados para 4 e 8, 16 e 32 e, 64 e 128 estados são extremamente semelhantes. Analisando o caso de 4 e 8 estados, verificou-se que o melhor código de 8 estados, $(17, 14)$, contém o melhor código de 4 estados, $(5, 4)$, como é mostrado na figura 4.9. Na figura 4.9a, tem-se a treliça do código $(5, 4)$ e na figura 4.9b, a treliça do $(17, 14)$. A figura 4.9c mostra que a treliça do código $(17, 14)$ pode ser separada em duas outras disjuntas. A treliça da esquerda na figura 4.9c é idêntica à da figura 4.9a. Como o algoritmo da TTCQ é iniciado no estado zero os resultados obtidos por estes dois códigos devem ser necessariamente iguais. Especula-se que para os outros casos este fato também se verifique. Desta forma, a TTCQ, na prática, disporia somente dos códigos para 4, 16, 64 e 256 estados. Conjetura-se que os melhores códigos de 512 estados produzirão resultados semelhantes aos melhores códigos de 256 estados, seguindo o padrão observado.

Por fim, nas figuras de 4.10 a 4.13, é feita uma comparação entre os códigos de quatro estados $(5, 2)$, $(5, 4)$ e $(7, 2)$, para 1, 2, 3 e 4 *bits* por símbolo, quando

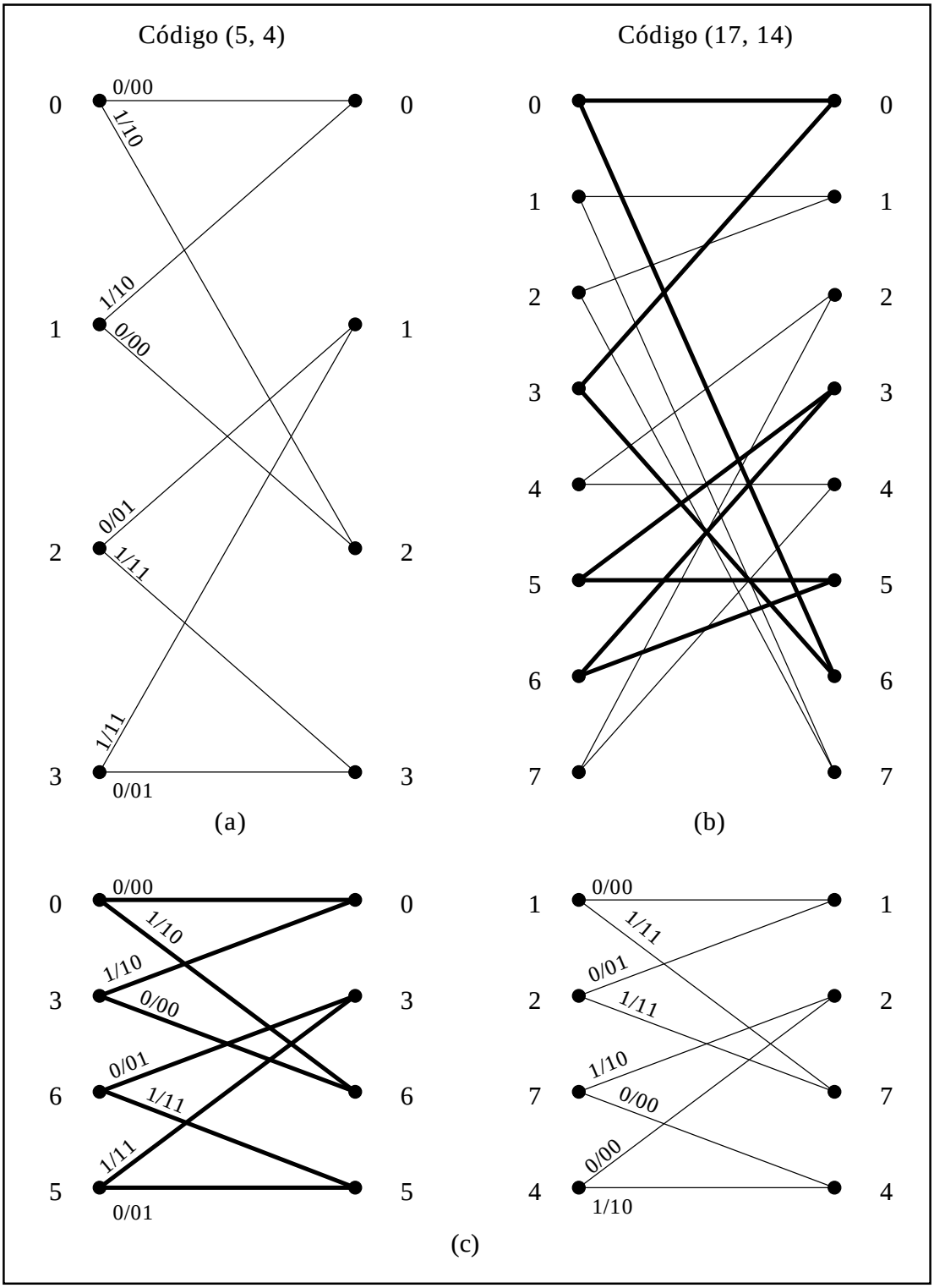


Figura 4.9: Os códigos $(5, 4)$ em (a) e $(17, 14)$ em (b) produzem os mesmos resultados quando utilizados na TTCQ porque a treliça do $(17, 14)$ pode ser dividida como mostrado em (c). A treliça do lado esquerdo em (c) que contém o estado 0 (zero) é idêndica à do código $(5, 4)$.

Tabela 4.14: Resultados da TTCQ-BCJR utilizando uma iteração, códigos para TTCQ da tabela 4.8, fonte gaussiana sem memória e pontos de reconstrução otimizados. Os valores de SNR estão listados em dB.

| **Número de** | **Taxa (*bits*)** | | | |
|---|---|---|---|---|
| **Estados** | **1** | **2** | **3** | **4** |
| **4** | 4,43 ± 0,05 | 9,33 ± 0,06 | 14,59 ± 0,08 | 20,10 ± 0,10 |
| **8** | 4,43 ± 0,05 | 9,33 ± 0,06 | 14,59 ± 0,08 | 20,10 ± 0,10 |
| **16** | 4,43 ± 0,05 | 9,33 ± 0,06 | 14,61 ± 0,07 | 20,00 ± 0,10 |
| **32** | 4,43 ± 0,05 | 9,33 ± 0,06 | 14,61 ± 0,07 | 20,00 ± 0,10 |
| **64** | 4,43 ± 0,05 | 9,33 ± 0,06 | 14,61 ± 0,07 | 20,00 ± 0,10 |
| **128** | 4,43 ± 0,05 | 9,33 ± 0,06 | 14,61 ± 0,07 | 20,00 ± 0,10 |
| **256** | 4,43 ± 0,05 | **,** ± *,** | **,** ± *,** | **,** ± *,** |
| **Lloyd-Max** | 4,40 | 9,30 | 14,62 | 20,22 |
| **Limite** | 6,02 | 12,04 | 18,06 | 24,08 |

utilizados na codificação de canal empregando TTCM. O código $(5, 2)$ produz os melhores resultados para TCM e TCQ, o $(5, 4)$ produz os melhores resultados para TTCQ e o $(7, 2)$ faz o mesmo pela TTCM. Como se pode observar, para uma BER menor que $10^{-2}$, região na qual a TTCM começa a se destacar na codificação de canal, o melhor código convolucional para TTCQ gera os piores resultados para a TTCM. Por outro lado, na faixa de SNR que é permitida para um quantizador para fonte gaussiana, o melhor código para TTCQ também produz os melhores resultados (ainda que modestos como na figura 4.10) como codificador de canal.

Portanto, aparentemente, os melhores códigos convolucionais para codificação de canal com TTCM e para taxa × distorção com TTCQ são completamente distintos, ao contrário do que ocorre com TCM e TCQ, quando se utiliza a TTCQ com a configuração específica desta subseção, isto é, mesmo número de iterações, tamanho de bloco e permutador da TTCM. Entretanto, alterando esta configuração e empregando os algoritmos serial refinado e paralelo propostos neste trabalho, derivados do algoritmo serial de CHAPPELIER *et al.* [23], será possível modificar esta situação e fazer com que a TTCQ supere a TCQ em termos de SNR.

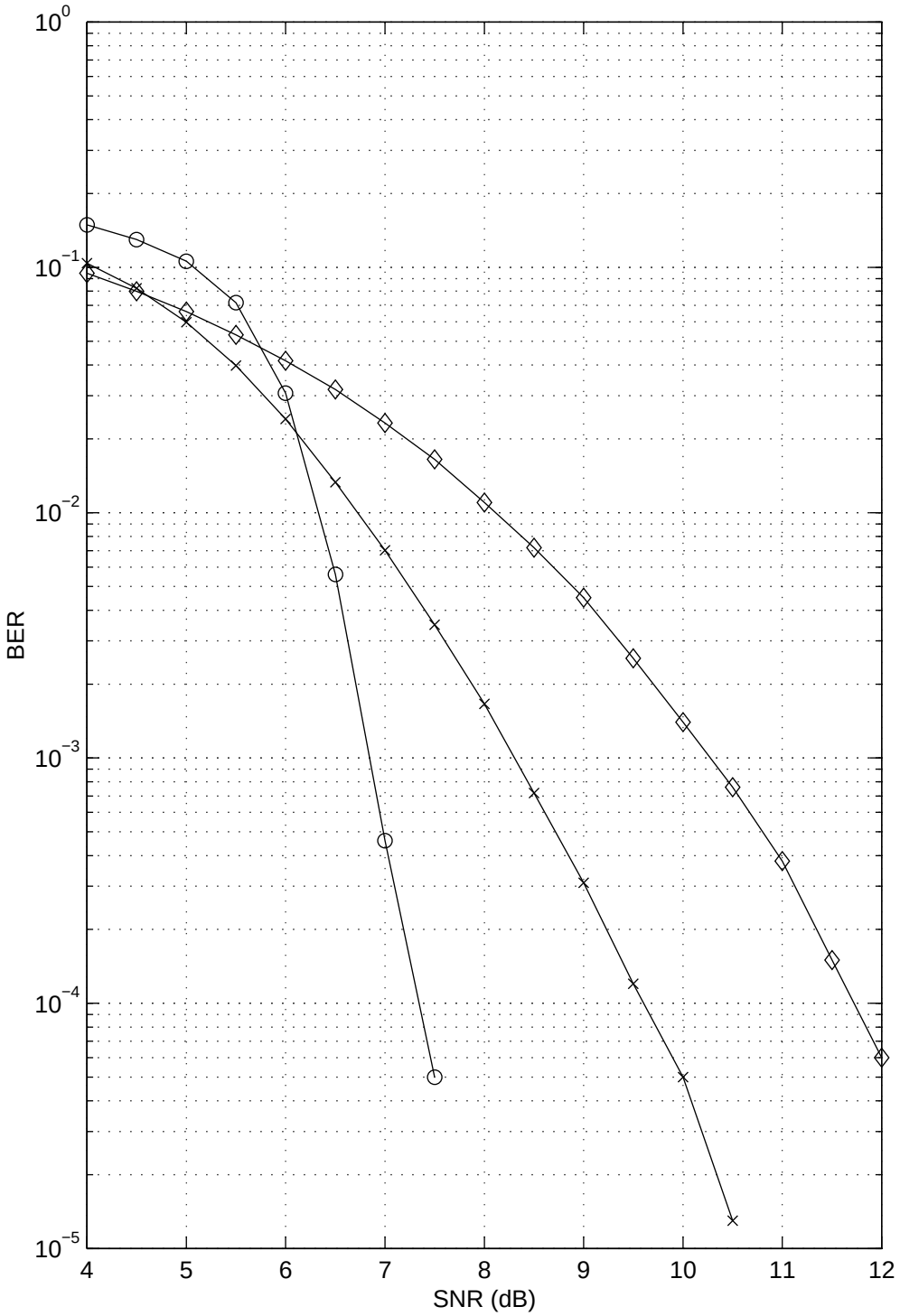


Figura 4.10: Comparação entre os códigos ($\times$) $(5, 2)$, ($\diamond$) $(5, 4)$ e ($\circ$) $(7, 2)$ quando utilizados na TTCM para 1 *bit* por símbolo.

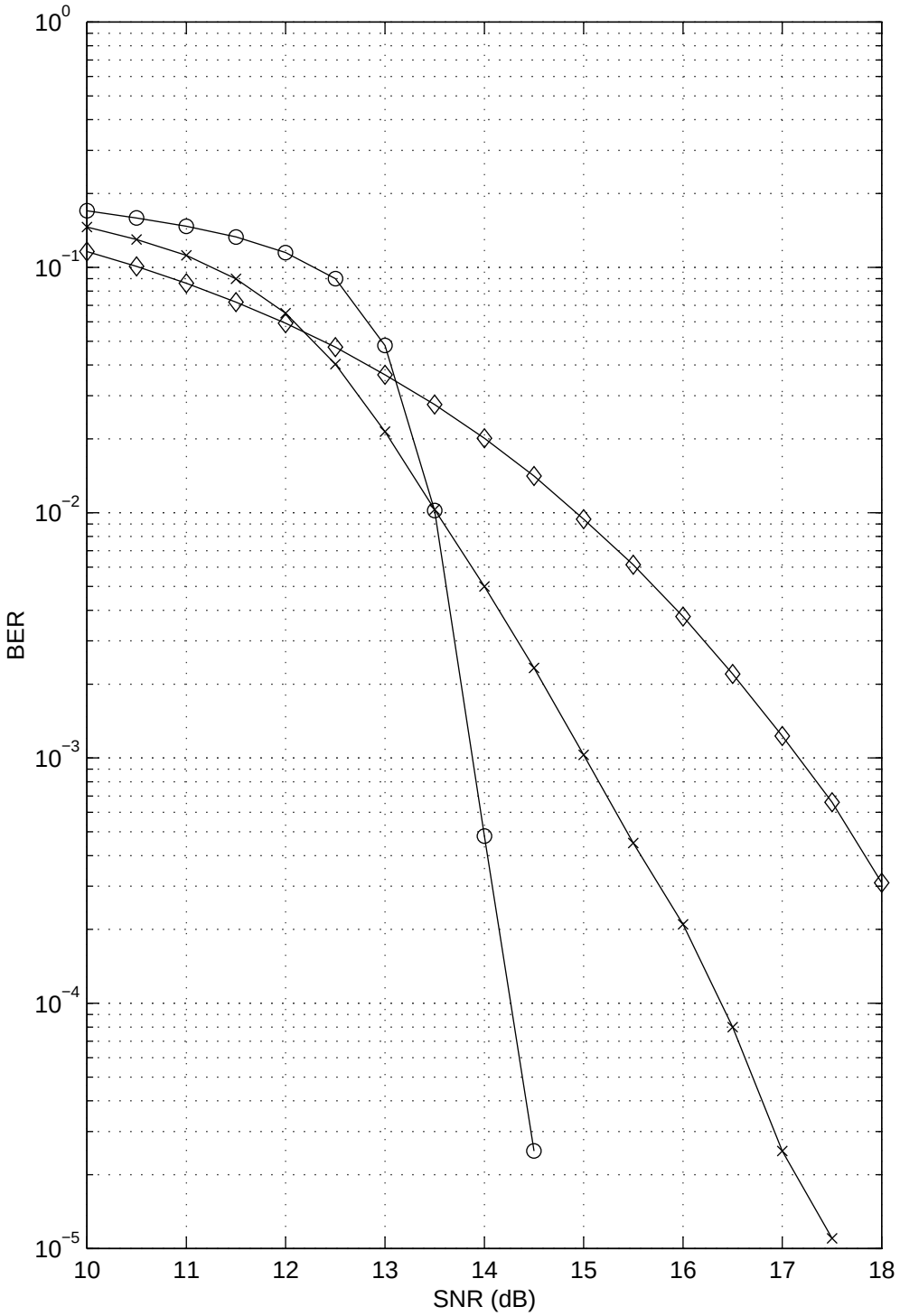


Figura 4.11: Comparação entre os códigos ($\times$) $(5,2)$, ($\diamond$) $(5,4)$ e ($\circ$) $(7,2)$ quando utilizados na TTCM para 2 *bits* por símbolo.

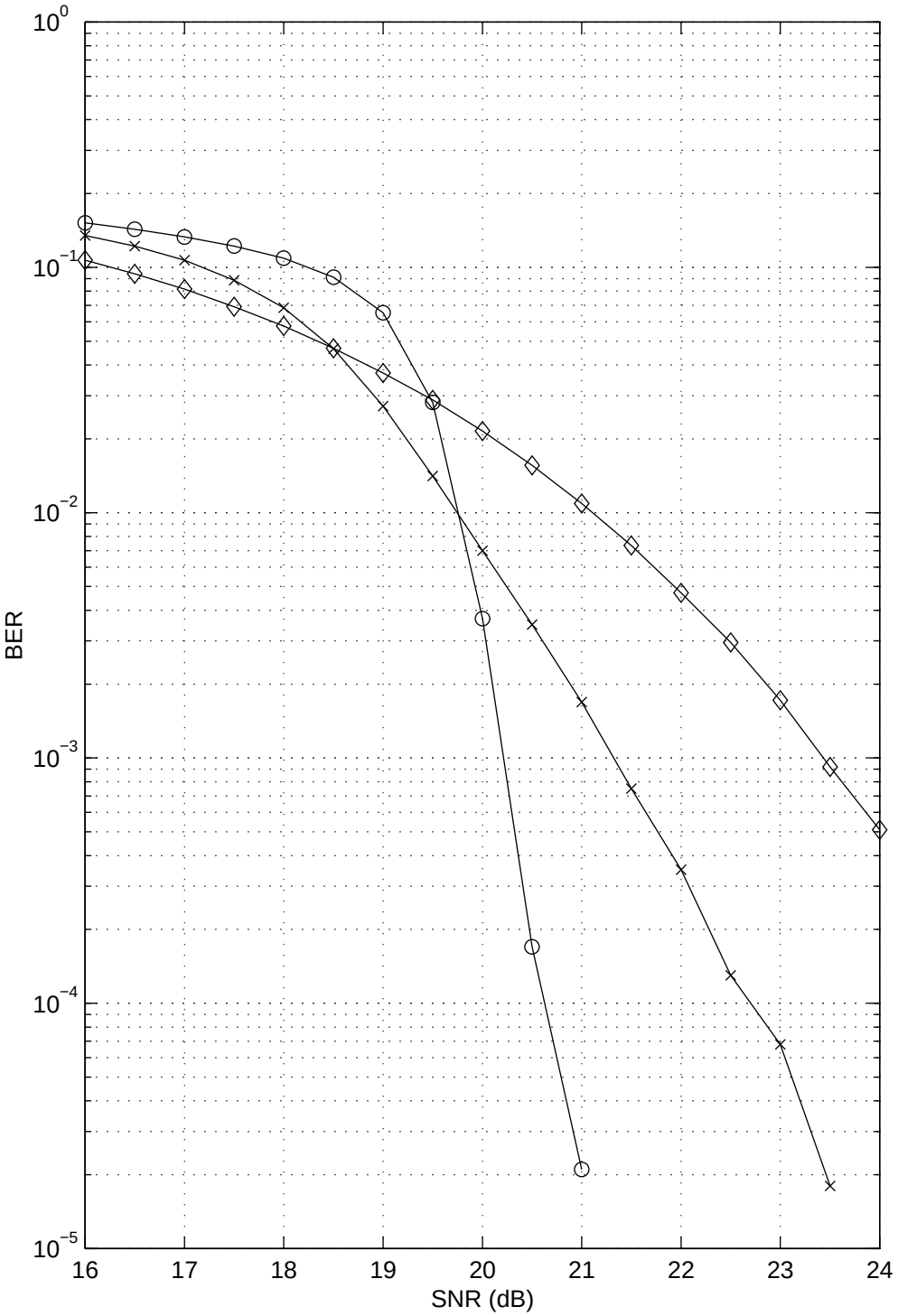


Figura 4.12: Comparação entre os códigos ($\times$) $(5,2)$, ($\diamond$) $(5,4)$ e ($\circ$) $(7,2)$ quando utilizados na TTCM para 3 *bits* por símbolo.

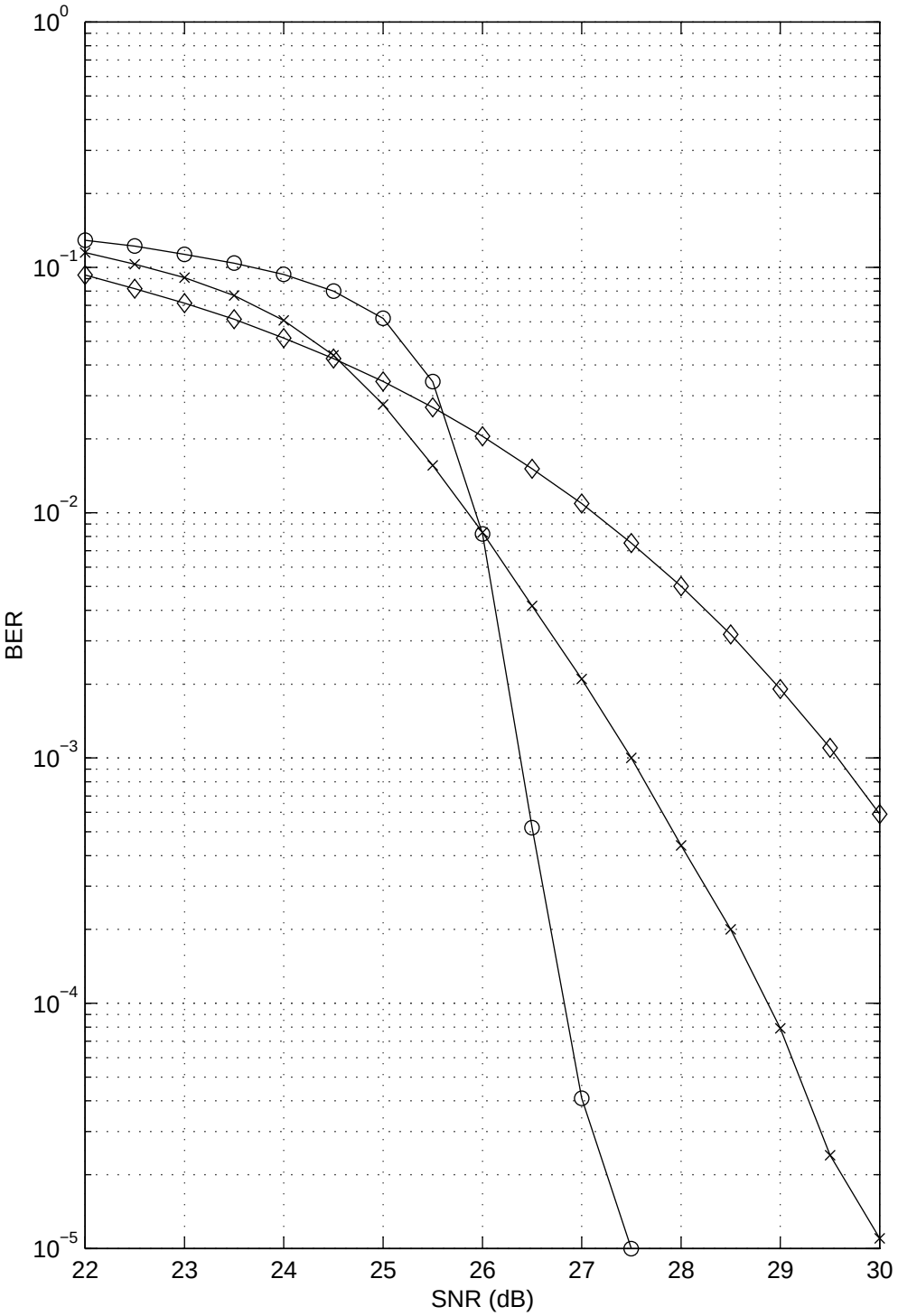


Figura 4.13: Comparação entre os códigos ($\times$) $(5, 2)$, ($\diamond$) $(5, 4)$ e ($\circ$) $(7, 2)$ quando utilizados na TTCM para 4 *bits* por símbolo.

### 4.3.2 Resultados do Algoritmo Serial Refinado

Os resultados anteriores demonstraram experimentalmente que a utilização direta da estrutura da TTCM na TTCQ não produziu um quantizador mais eficiente que a TCQ. Então, como próximo passo, será empregado o algoritmo apresentado na subseção 4.2.2, na tentativa de obter melhores resultados.

Como bem observado por CHAPPELIER *et al.* [23], a principal diferença entre TTCQ e TTCM reside nas distribuições dos vetores aleatótios $\boldsymbol{Y}$ (canal ruidoso) e $\boldsymbol{A}$ (fonte a ser quantizada) sobre os quais o algoritmo iterativo atua. No caso dos códigos de canal, assume-se geralmente que o codificador emite uma palavra código $\mathbf{x}$ à qual é adicionado ruído gaussiano branco $\mathbf{e}$, produzindo o vetor $\mathbf{y} = \mathbf{x} + \mathbf{e}$. Logo, a distribuição de $\boldsymbol{Y}$ pode ser representada como uma soma de distribuições gaussianas multidimensionais centradas em cada palavra código $\mathbf{x}$. Portanto, a probabilidade de um $\mathbf{y}$ estar muito afastado de uma palavra código $\mathbf{x}$ tende a zero com o aumento da dimensão de $\mathbf{y}$. Visto que os comprimentos típicos das seqüências são grandes, esta probabilidade pode ser considerada aproximadamente zero [23].

Contudo, no caso da quantização (TTCQ), assumindo uma distribuição uniforme, gaussiana ou laplaciana das componentes $a_k$ do vetor $\mathbf{a}$, esta propriedade não se verifica. Ou seja, não se pode assumir que o vetor $\mathbf{a}$ está inicialmente próximo de uma palavra código $\mathbf{w}$ da TTCQ. Isto acarreta problemas de convergência pois as métricas não favorecerão um caminho particular (na treliça) em detrimento de outros, ao contrário do caso dos códigos de canal (TTCM) [23].

Assim sendo, ao contrário do que ocorre com a TTCM, utilizar blocos de dimensão elevada prejudica a convergência da TTCQ. Além de reduzir a dimensão do vetor $\mathbf{a}$, CHAPPELIER *et al.* [23] constatam que empregar um permutador $S$-aleatório (*S-random interleaver* [2, 10]) torna a TTCQ mais eficiente. Porém, para que a TTCQ possa superar de forma significativa a TCQ em termos de taxa $\times$ distorção, um número muito elevado de iterações deve ser adotado.

No trabalho de CHAPPELIER *et al.* [23], é assumido que o algoritmo de quantização convergiu se, após $N_{it}$ iterações, $n_{\mathbf{u}} \geq N_{\mathbf{u}} = 50$, utilizando a notação da subseção 4.2.2. Caso não haja convergência, o vetor $\mathbf{u}_{min}$ que gerou a menor distorção é utilizado. O número de iterações utilizado por seqüência simulada ($i_{max}$) é sempre igual a $N_{it}$, já que não há um critério de parada para interromper o algorit-

mo numa iteração $i < N_{it}$. Neste trabalho, foi adicionado um critério de parada e $i_{min}$ é armazenado para cada seqüência simulada. O objetivo do critério de parada é reduzir o número de iterações executadas em média durante as simulações ($\mu_{I_{max}}$). Armazenar $i_{min}$ tem por objetivo estimar seu valor médio $\mu_{I_{min}}$ para compará-lo com o valor de $\mu_{I_{max}}$ e $N_{it}$.

Primeiramente, será feita uma breve comparação com os resultados obtidos pelo algoritmo serial de CHAPPELIER *et al.* [23]. Uma fonte uniforme de média zero e variância unitária é codificada com TTCQ com taxas de 1, 2, 6 e 8 *bits* por amostra. Os blocos são de tamanho $N = 50$ e $N = 64$, com um número de seqüências simuladas $N_s = 156000/N$[5]. O código convolucional de 4 estados $(7, 2)$ e os níveis de reconstrução do quantizador de Lloyd-Max de taxa $R+1$ são novamente utilizados. Na subseção anterior, conjeturou-se que se a TTCQ convergisse com o código $(7, 2)$ ele produziria melhores resultados e, de fato, isto foi constatado para este código de 4 estados.

Com esta configuração, foram obtidos os resultados da tabela 4.15, onde as últimas três linhas correspondem aos resultados de CHAPPELIER *et al.* [23] para 2, 6 e 8 *bits* por amostra, respectivamente. Na coluna $\epsilon$, o traço (–) significa que o critério de parada foi desativado, ou seja, o valor de $\Delta^{es}_{max}$ não é utilizado para interromper o processo iterativo. Os valores de $\epsilon$ utilizados no algoritmo serial refinado são 0 (zero) e $10^{-3}$. O valor $\epsilon = 0$ permite verificar se pontos fixos da TTCQ são encontrados. Já o valor $\epsilon = 10^{-3}$ foi escolhido arbitrariamente. Como se pode ver nesta tabela, para as taxas testadas de 1, 2, 6 e 8 *bits* por amostra, a adoção do critério de parada reduziu o número médio de iterações ($\mu_{I_{max}}$) executadas e manteve o valor da SNR inalterado. Para $\epsilon = 0$, isto era esperado porque, quando se encontra um ponto fixo, não há necessidade de se executar mais nenhuma iteração já que o valor de $\mathbf{u}_{min}$ não se altera mais. Para $\epsilon = 10^{-3}$, foi apenas uma coincidência o valor de SNR não ter se alterado, já que este valor foi escolhido arbitrariamente. Note também que, apesar de se estar utilizando um número elevado de iterações, a iteração mínima $i_{min}$ correspondente a $\mathbf{u}_{min}$ foi, em média ($\mu_{I_{min}}$), bem menor que $N_{it}$ nos casos testados, ou seja, de 15 a 30% do valor de $N_{it}$. Logo, a determinação

[5]No trabalho de CHAPPELIER *et al.* [23] $N_s = 10000$, o que produz um intervalo de confiança mais preciso. Entretanto, ele não é incluído como parte de seus resultados.

Tabela 4.15: Comparação entre os algoritmos seriais para TTCQ com e sem critério de parada. TTCQ com código convolucional $(7, 2)$ e pontos de reconstrução de Lloyd-Max the taxa $R + 1$. Fonte uniforme de média zero e variância unitária.

| $\epsilon$ | $N_s$ | $N$ | $N_{it}$ | $R$ $(bits)$ | SNR (dB) | $\mu_{I_{min}}$ | $\mu_{I_{max}}$ |
|---|---|---|---|---|---|---|---|
| $10^{-3}$ | 2437 | 64 | 10000 | 1 | 5,96 ± 0,02 | 1600 ± 300 | 2900 ± 300 |
| 0 | 2437 | 64 | 10000 | 1 | 5,96 ± 0,02 | 1600 ± 300 | 2900 ± 300 |
| – | 2437 | 64 | 10000 | 1 | 5,96 ± 0,02 | 1600 ± 300 | 10000 |
| $10^{-3}$ | 2437 | 64 | 10000 | 2 | 12,62 ± 0,01 | 1300 ± 200 | 4000 ± 300 |
| 0 | 2437 | 64 | 10000 | 2 | 12,62 ± 0,01 | 1300 ± 200 | 4000 ± 300 |
| – | 2437 | 64 | 10000 | 2 | 12,62 ± 0,01 | 1300 ± 200 | 10000 |
| $10^{-3}$ | 2437 | 64 | 10000 | 6 | 37,19 ± 0,01 | 1600 ± 200 | 6200 ± 400 |
| 0 | 2437 | 64 | 10000 | 6 | 37,19 ± 0,01 | 1600 ± 200 | 10000 |
| – | 2437 | 64 | 10000 | 6 | 37,19 ± 0,01 | 1600 ± 200 | 10000 |
| $10^{-3}$ | 3120 | 50 | 1000 | 8 | 49,16 ± 0,01 | 310 ± 30 | 460 ± 30 |
| 0 | 3120 | 50 | 1000 | 8 | 49,16 ± 0,01 | 310 ± 30 | 1000 |
| – | 3120 | 50 | 1000 | 8 | 49,16 ± 0,01 | 310 ± 30 | 1000 |
| Ref. [23] | 10000 | 64 | 10000 | 2 | 12,64 | – | 10000 |
| Ref. [23] | 10000 | 64 | 10000 | 6 | 37,20 | – | 10000 |
| Ref. [23] | 10000 | 50 | 1000 | 8 | 49,16 | – | 1000 |

do valor de $\mathbf{u}_{min}$ ocorre bem antes da iteração $N_{it}$. Comparando-se $\mu_{I_{min}}$ com $\mu_{I_{max}}$ para os valores de $\epsilon$ tais que $\mu_{I_{max}} < N_{it}$ observa-se que a determinação do valor de $\mathbf{u}_{min}$ também ocorre bem antes do critério de parada ser satisfeito. Contudo, os valores de $\epsilon$ adotados não foram selecionados de modo a minimizar a diferença $\mu_{I_{max}} - \mu_{I_{min}}$. Portanto, encontrar valores de $\epsilon$ que minimizem esta diferença é um procedimento que deve ser efetuado porque minimiza o tempo de codificação (médio) da TTCQ.

Comparando os resultados das linhas 4, 5 e 6 da tabela 4.15 para 2 *bits* por amostra com o respectivo resultado de CHAPPELIER *et al.* [23] na linha 13 da mesma tabela nota-se uma diferença de 0,02 dB. Esta diferença pode ser causada

pelos permutadores utilizados. Embora ambos sejam $S$-aleatórios não há como garantir que sejam iguais e é possível que para uma dada seqüência de simulações um permutador seja mais adequado que um outro. Note que os resultados são idênticos para 8 *bits* por amostra em termos de SNR.

O algoritmo serial refinado não assume que $n_{\mathbf{u}} \geq N_{\mathbf{u}}$ implica $\mathbf{u}_{min} = \mathbf{u}^{(i_{max})}$ como o algoritmo serial de CHAPPELIER *et al.* [23] (neste caso, $i_{max}$ é sempre igual a $N_{it}$) e esperava-se que esta mudança pudesse reduzir a distorção média produzida pela TTCQ. Contudo, com os resultados da tabela 4.15, não foi possível constatar nenhuma redução da distorção média produzida pelo algoritmo serial refinado em relação ao algoritmo serial de CHAPPELIER *et al.* [23]. É possível que o número de simulações para as quais esta modificação é efetiva não seja suficiente para alterar o valor da distorção média. Entretanto, esta modificação não aumenta a complexidade em relação ao algoritmo serial de CHAPPELIER *et al.* [23], já que $\mathbf{u}_{min}$ deve ser armazendo em ambos os algoritmos, e pode ser mantida uma vez que tem o potencial de reduzir a distorção média da TTCQ.

Em seguida, serão mostrados resultados obtidos para a TTCQ em função do tamanho do bloco (dimensão) $N$. Será utilizado agora $R = 1$ *bit* por amostra e o número de seqüências simuladas varia de acordo com a relação $N_s = 156000/N$, onde $N = 8j$ para $j = 1, 2, \ldots, 39$. As figuras 4.14, 4.15 e 4.16 correspondem às fontes uniforme, gaussiana e laplaciana, respectivamente. A curva correspondente da TCQ foi incluída para comparação. Em todos os casos, há uma região em que a TTCQ supera a TCQ significativamente. Como já havia sido mencionado, o aumento de $N$ afeta a convergência da TTCQ reduzindo seu desempenho.

As figuras 4.17 e 4.18 comparam a TTCQ e TCQ com níveis de reconstrução otimizados para as fontes uniforme e gaussiana, respectivamente. No caso da fonte uniforme, a otimização dos níveis de reconstrução da TTCQ aparentemente eliminou a queda de desempenho com o aumento da dimensão $N$, além de produzir um ganho em relação à TCQ da ordem de 0,15 dB. Entretanto, uma simulação com tamanho de bloco $N = 1000$ produz uma SNR de 6,25 $\pm$ 0,02 dB, indicando que o problema continua ocorrendo. No caso da fonte gaussiana, apesar do ganho expressivo de cerca de 0,3 dB em relação à TCQ para as dimensões entre 64 e 160, a degradação com o aumento da dimensão é bem mais acentuada que no caso da fonte uniforme.

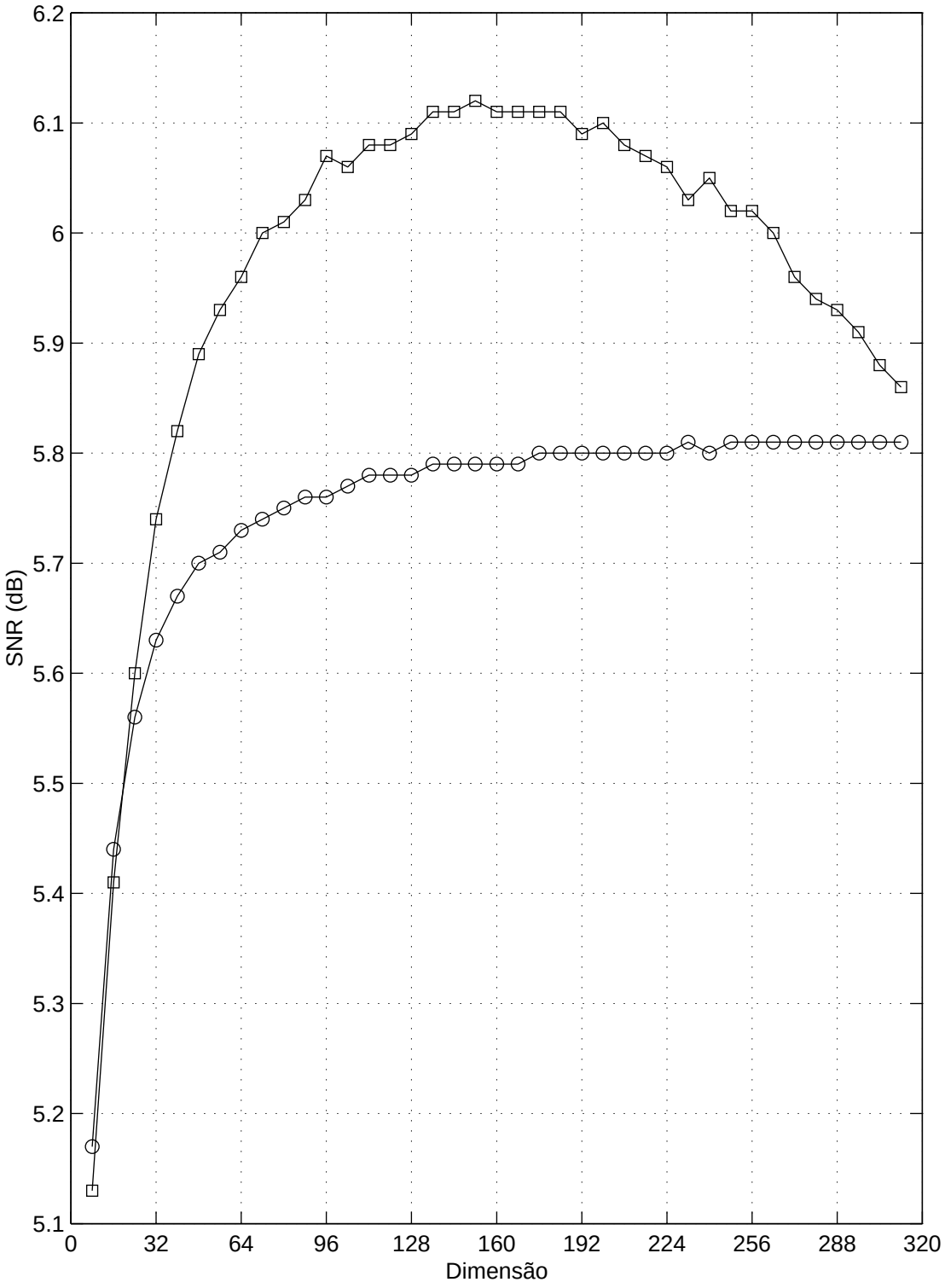


Figura 4.14: Comparando (∘) TCQ e (□) TTCQ modo serial em função da dimensão. Fonte uniforme sem memória de média zero e variância unitária.

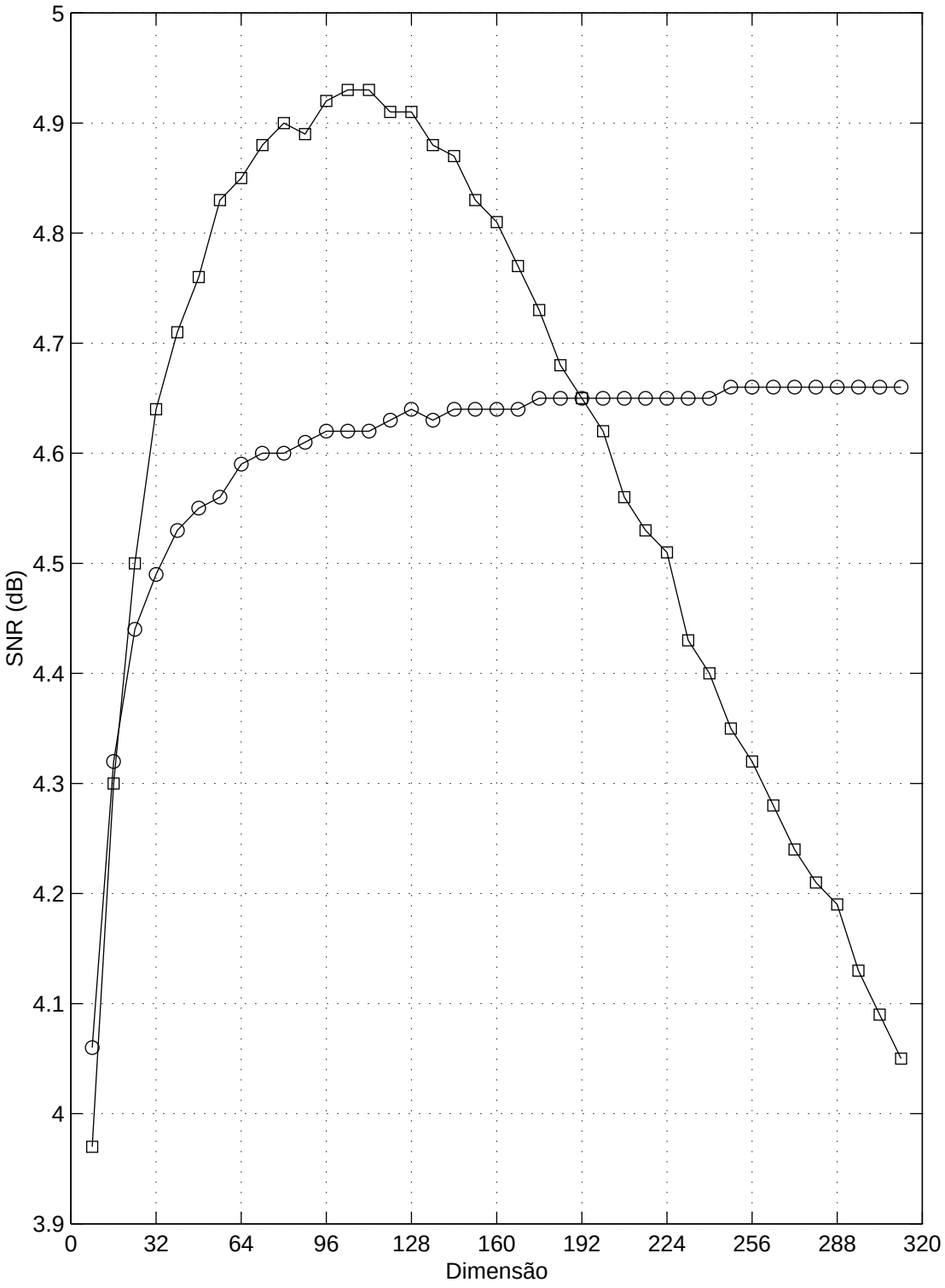


Figura 4.15: Comparando (○) TCQ e (□) TTCQ modo serial em função da dimensão. Fonte gaussiana sem memória de média zero e variância unitária.

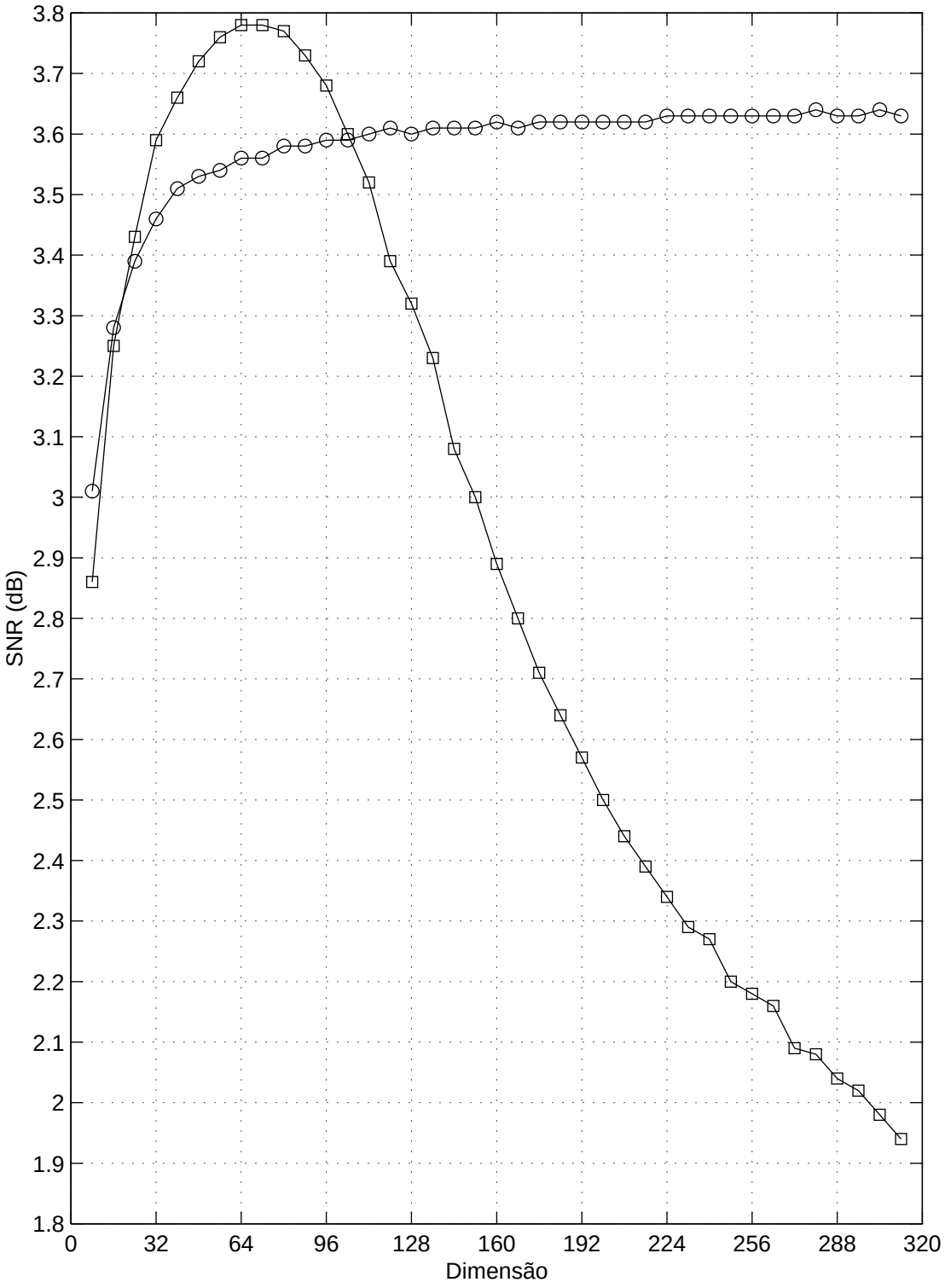


Figura 4.16: Comparando (○) TCQ e (□) TTCQ modo serial em função da dimensão. Fonte laplaciana sem memória de média zero e variância unitária.

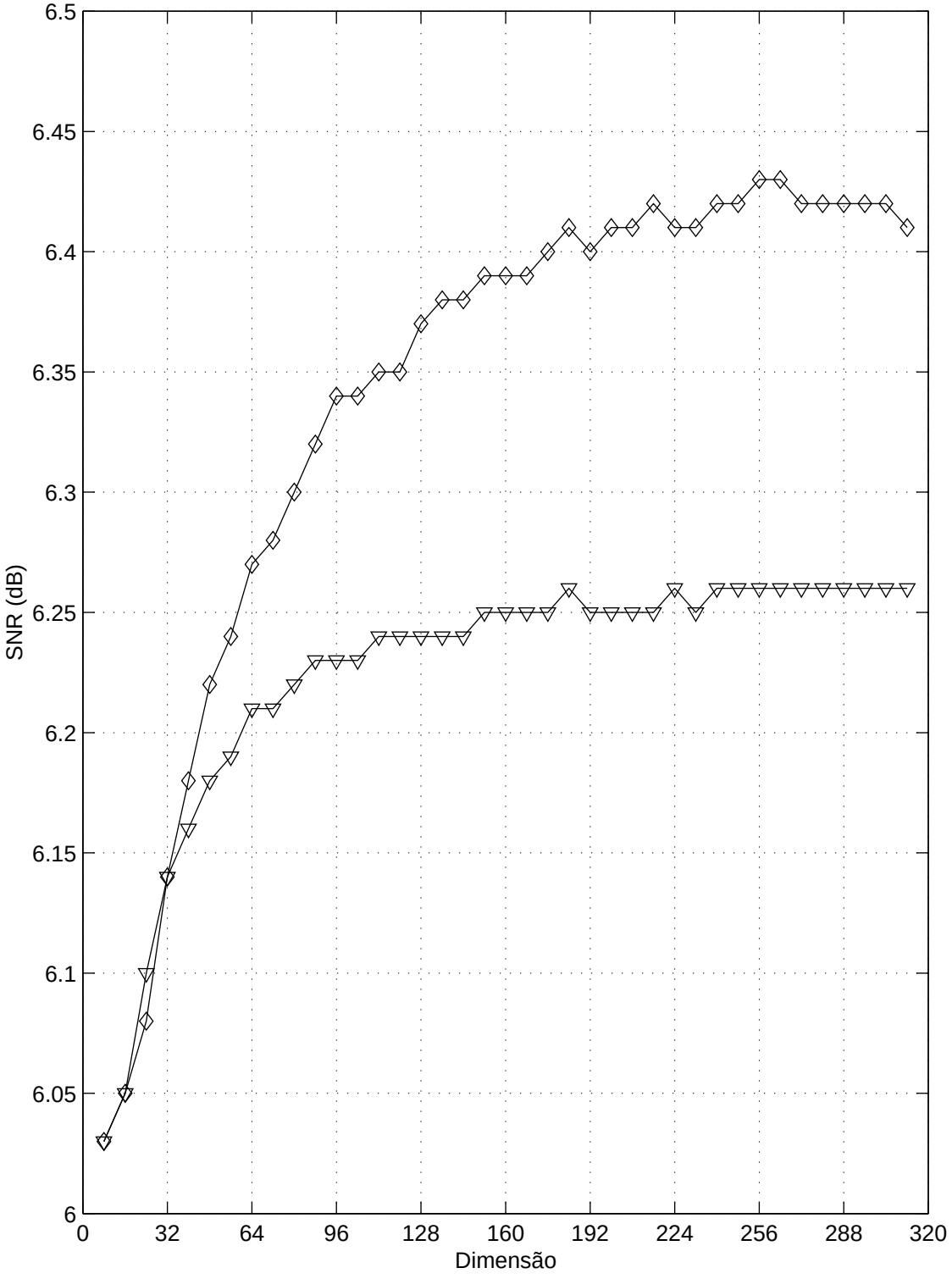


Figura 4.17: Comparando ($\triangledown$) TCQ otimizada e ($\diamond$) TTCQ otimizada modo serial em função da dimensão. Fonte uniforme sem memória de média zero e variância unitária.

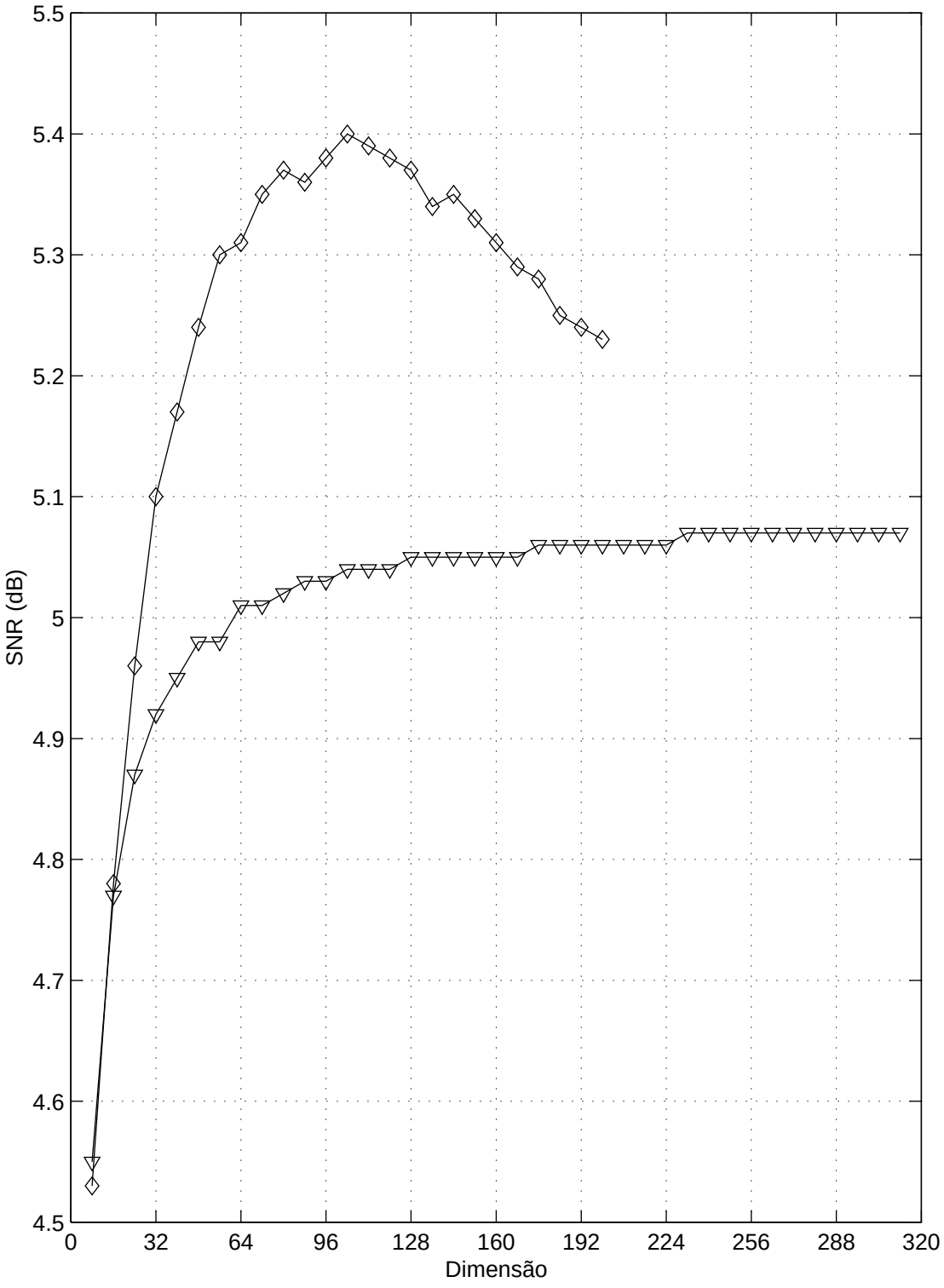


Figura 4.18: Comparando ($\triangledown$) TCQ otimizada e ($\diamond$) TTCQ otimizada modo serial em função da dimensão. Fonte gaussiana sem memória de média zero e variância unitária.

### 4.3.3 Resultados do Algoritmo Paralelo

Nesta subseção, comparam-se os algoritmos de codificação serial refinado e paralelo para TTCQ. Será utilizado novamente $R = 1$ *bit* por amostra e o número de seqüências simuladas varia de acordo com a relação $N_s = 156000/N$, onde $N = 8j$ para $j = 1, 2, \ldots, 39$. Na figura 4.19, os resultados dos algoritmos serial refinado e paralelo, com e sem otimização dos níveis de reconstrução da TTCQ, são colocados em um mesmo gráfico para a fonte uniforme. Nota-se que, mesmo com a otimização dos níveis de reconstrução, o algorimo paralelo é superior ao serial refinado em termos de taxa $\times$ distorção. A figura 4.20 contém resultados para a fonte gaussiana. Desta vez, somente TTCQ otimizada com os algoritmos serial refinado e paralelo estão disponíveis para comparação. Mais uma vez, o algoritmo paralelo supera o serial refinado. A figura 4.21, contém resultados para a fonte laplaciana. Nesta figura, o que se pode observar é o ganho de aproximadamente 0,4 dB que a TTCQ otimizada modo paralelo obtém em relação à TCQ otimizada.

Por fim, a tabela 4.16 faz uma comparação entre o tempo de processamento dos algoritmos serial refinado e paralelo para TTCQ com níveis de reconstrução otimizados e fonte uniforme. Esta comparação assume que o algoritmo paralelo foi implementado de tal forma que os codificadores SOVA $A$ e $B$ processem e produzam informação simultaneamente. Desta forma, sendo $T$ o tempo de processamento utilizado por cada um dos codificadores ($A$ ou $B$) cada iteração no modo serial utiliza um tempo de processamento $t_s = 2T$ enquanto que no modo paralelo utiliza $t_p = T$. Os valores de tempo listados na tabela são o resultado da multiplicação do valor estimado de $\mu_{I_{min}}$ por $t_s$ ou $t_p$. Isto posto, nota-se que o modo paralelo consumiria, em média, de 80 a 90% do tempo de processamento do modo serial para encontrar $\mathbf{u}_{min}$.

## 4.4 Resumo

Neste capítulo, um novo esquema de quantização foi proposto, a turbo quantização codificada por treliças (TTCQ – *Turbo TCQ*), derivado do esquema de codificação de canal de ROBERTSON *et al.* [7], a turbo modulação codificada por treliças (TTCM – *Turbo TCM*). Porém, a implementação da TTCQ feita diretamente a par-

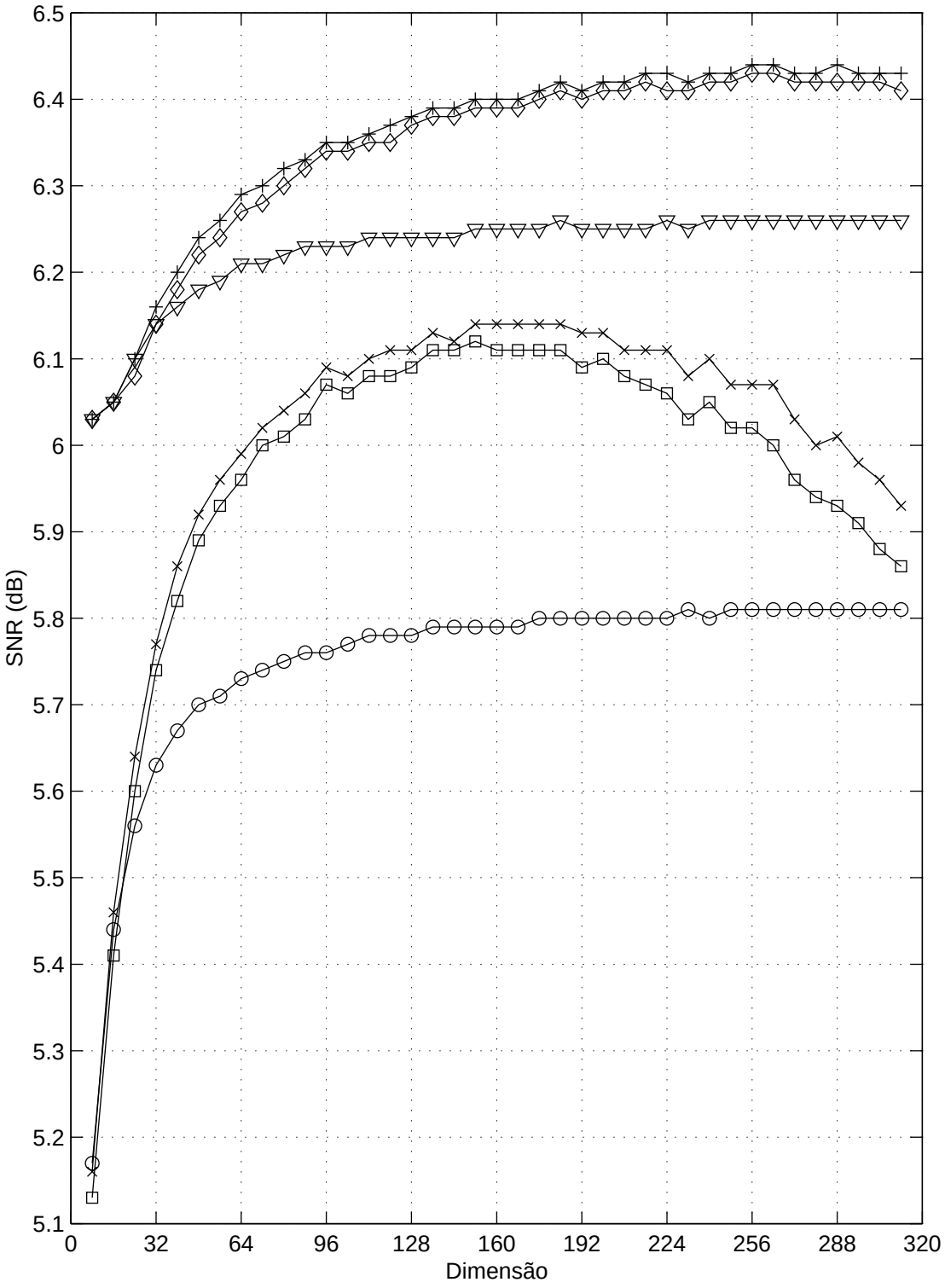


Figura 4.19: Comparando (∘) TCQ, (□) TTCQ modo serial, (×) TTCQ modo paralelo, (▽) TCQ otimizada, (⋄) TTCQ otimizada modo serial e (+) TTCQ otimizada modo paralelo. Fonte uniforme sem memória de média zero e variância unitária.

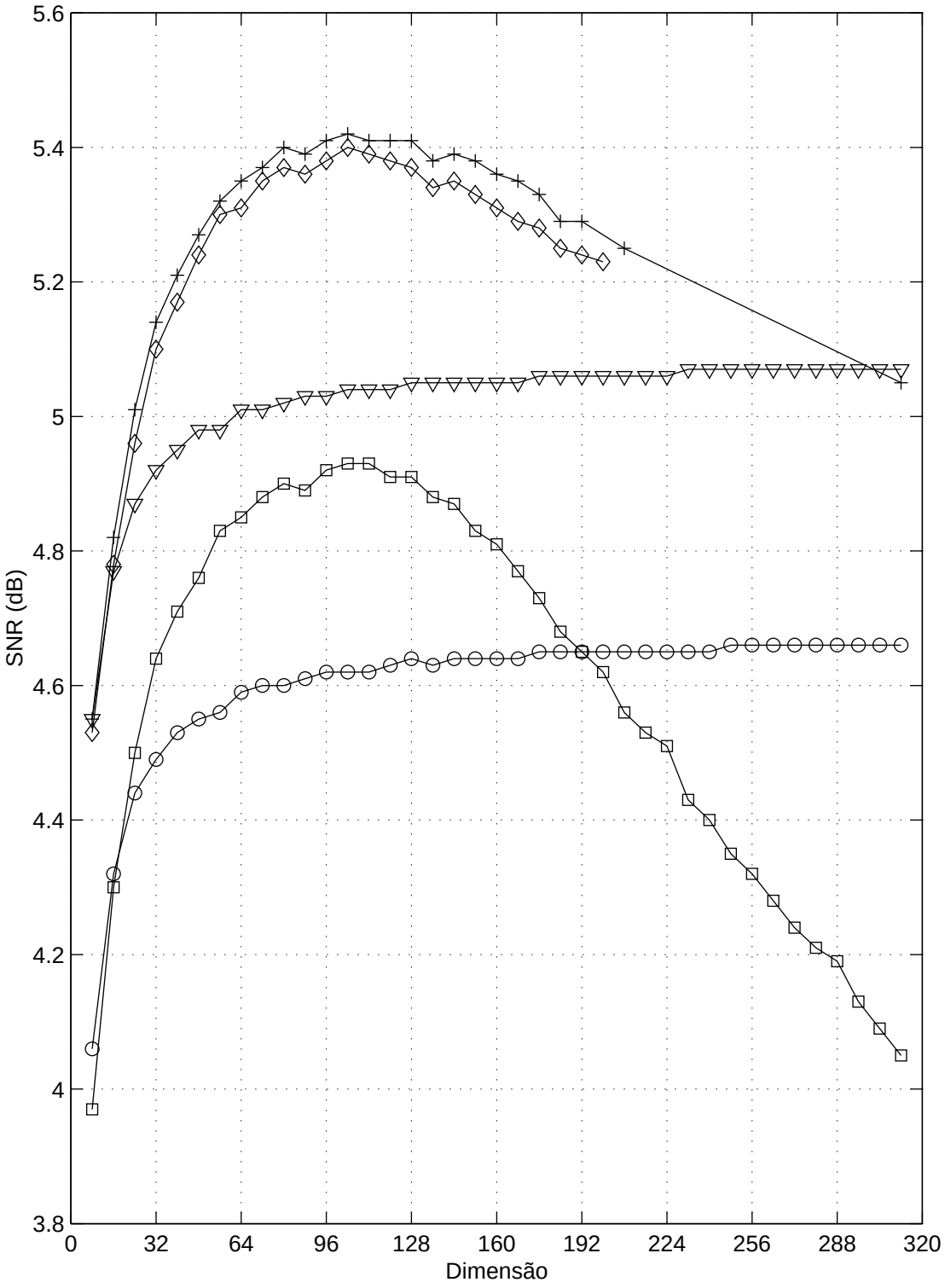


Figura 4.20: Comparando (∘) TCQ, (□) TTCQ modo serial, (▽) TCQ otimizada, (⋄) TTCQ otimizada modo serial e (+) TTCQ otimizada modo paralelo. Fonte gaussiana sem memória de média zero e variância unitária.

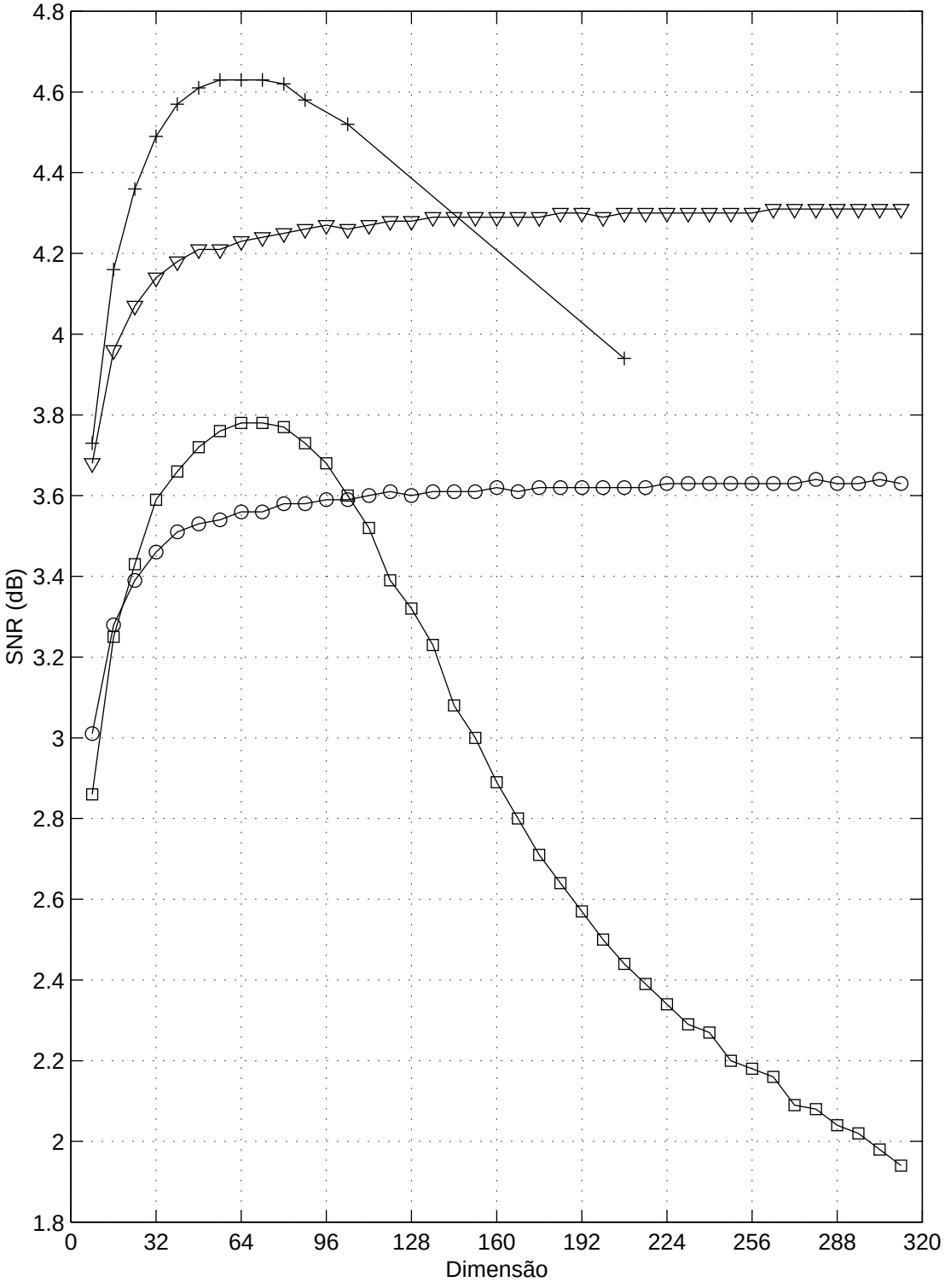


Figura 4.21: Comparando (○) TCQ, (□) TTCQ modo serial, (▽) TCQ otimizada e (+) TTCQ otimizada modo paralelo. Fonte laplaciana sem memória de média zero e variância unitária.

Tabela 4.16: Comparando o tempo de processamento dos algoritmos serial refinado e paralelo para TTCQ com níveis de reconstrução otimizados. $T$ é o tempo de processamento utilizado por cada um dos codificadores componentes do codificador TTCQ. Fonte uniforme de média zero e variância unitária.

| **Dimensão** ($N$) | **Serial** ($T_{ser}(N)$) | **Paralelo** ($T_{par}(N)$) | $T_{par}(N)/T_{ser}(N)$ |
|---|---|---|---|
| 160 | $6000T$ | $5200T$ | 0,87 |
| 168 | $6400T$ | $5100T$ | 0,80 |
| 176 | $6600T$ | $5300T$ | 0,80 |
| 184 | $6400T$ | $5600T$ | 0,88 |
| 192 | $7000T$ | $5700T$ | 0,81 |
| 200 | $7400T$ | $5800T$ | 0,78 |
| 208 | $7200T$ | $6000T$ | 0,83 |

tir da TTCM, seguindo passos semelhantes aos de MARCELLIN [6] que desenvolveu a TCQ a partir da TCM, ou seja, utilizando simplesmente a estrutura do codificador e do decodificador da TTCM, com a mesma permutação pseudo-aleatória, com o mesmo número de iterações e com o mesmo tamanho de bloco, como quantizador não produziu resultados satisfatórios. Neste caso, constatou-se que a TTCQ necessitava do quadrado do número de estados para produzir resultados semelhantes aos da TCQ.

No entanto, em 2003, CHAPPELIER *et al.* [23], investigando também a possibilidade de usar a estrutura da TTCM como quantizador, observaram que era possível ultrapassar significativamente o desempenho da TCQ quando se empregava um permutador $S$-aleatório (*S-random interleaver*), um número de iterações milhares de vezes maior que aquele adotado na TTCM e blocos de tamanho médio. Com base nestas observações, foram propostos dois novos algoritmos de codificação no presente trabalho que melhoraram ainda mais o desempenho da TTCQ em termos de tempo de codificação e da distorção média produzida.

O primeiro algoritmo de codificação para TTCQ proposto foi o serial refinado, descrito na subseção 4.2.2. Este algoritmo é o resultado do refinamento do algoritmo serial de CHAPPELIER *et al.* [23] e introduz um critério de parada que avalia se o

ponto fixo da TTCQ foi encontrado. Com isto, foi possível reduzir significativamente o número de iterações executadas durante o processo de codificação da TTCQ e, conseqüentemente, o tempo de codificação do algoritmo serial refinado se tornou bem menor que o do algoritmo serial de CHAPPELIER *et al.* [23] para os casos testados. Acredita-se que esta redução no tempo de codificação possa ser ainda maior se o valor do parâmetro $\epsilon$, que controla o critério de parada do algoritmo serial refinado, for determinado experimentalmente de modo a minimizar o tempo de codificação sem aumentar o valor da distorção média da TTCQ.

O segundo algoritmo de codificação para TTCQ proposto foi o paralelo, descrito na subseção 4.2.3. Ele permite executar duas instâncias distintas do algoritmo serial refinado em paralelo, explorando a estrutura do codificador iterativo da TTCQ. A instância que produz a menor distorção média é selecionada e, com isto, é possível obter uma distorção média que é menor que a do algoritmo serial refinado e, por conseqüência, menor que a do algoritmo serial de CHAPPELIER *et al.* [23].

Adicionalmente, foi realizada a otimização dos níveis de reconstrução da TTCQ, de modo semelhante ao que MARCELLIN [6] fez com a TCQ, e foram feitas simulações com as fontes gaussiana e laplaciana, visto que no trabalho de CHAPPELIER *et al.* [23] somente a TTCQ não otimizada com fonte uniforme havia sido testada. Constatou-se novamente que a TTCQ supera significativamente a TCQ. Todavia, a corrente limitação do tamanho do bloco e, principalmente, o ainda elevado número de iterações, acarretando um tempo de processamento muito maior que o da TCQ, certamente representam um problema para várias aplicações.

# Capítulo 5

# Turbo Modulação e Codificação Conjunta de Fonte e Canal

Neste capítulo, um sistema de codificação conjunta fonte e canal é construído com TTCQ e TTCM de modo semelhante ao que FISCHER *et al.* [8] fizeram empregando TCQ e TCM. Separadamente, TTCM e TTCQ superam a TCM e a TCQ, respectivamente. Porém, no caso da TTCM, é preciso garantir uma determinada relação sinal-ruído de canal e utilizar tamanhos de bloco relativamente grandes ($N \geq 1000$) para ter um desempenho superior ao da TCM. No caso da TTCQ, é necessário utilizar blocos de tamanho médio (entre 60 e 300) para ultrapassar o desempenho da TCQ. Blocos deste tamanho reduzem a eficiência da TTCM mas, apesar disto, o desempenho conjunto da TTCQ e TTCM é melhor que o da TCQ e TCM quando se assegura uma determinada relação sinal-ruído de canal, como será descrito na seção de resultados.

## 5.1 Codificação Conjunta de Fonte e Canal com TCQ e TCM

Normalmente, o problema de codificação conjunta de fonte e canal é abordado de duas formas distintas [8]. Na primeira, assume-se um modelo digital de canal, em geral, um canal binário simétrico (BSC – *Binary Symmetric Channel*) e o código da fonte é projetado para minimizar a distorção extra introduzida pelo canal. Na segunda, permite-se que a seleção dos símbolos do modulador e o mapeamento dos

símbolos do codificador da fonte nos símbolos do modulador estejam sob o controle do projetista do sistema. Esta última abordagem foi utilizada na codificação conjunta com TCQ e TCM, sendo também utilizada na codificação conjunta com TTCQ e TTCM. O objetivo, neste caso, é "casar" as treliças utilizadas na codificação da fonte e do canal [8].

O sistema de codificação conjunta com TCQ e TCM é estruturado de tal forma que o quadrado da distância entre seqüências de canal é comparável ao quadrado do erro de quantização. Esta formulação garante que eventos prováveis de erro de canal acarretem uma pequena distorção extra na quantização [8].

O esquema básico de codificação conjunta com TCQ e TCM é mostrado na figura 5.1, onde conforme notação do capítulo 3:

1. $\mathbf{a} = [a_0 a_1 \ldots a_{N-1}]^T$ é o vetor de amostras produzidas pela fonte e $a_k \in \mathcal{A}$ com $\mathcal{A} \subset \mathbb{R}$;

2. $\mathbf{u} = [u_0 u_1 \ldots u_{N-1}]^T$ é o vetor de símbolos correspondente às amostras da fonte quantizadas com $R$ *bits* por amostra e $u_k \in \mathcal{U} = \{0, 1, \ldots, M/2 - 1\}$ para $k = 0, 1, \ldots, N-1$, onde $M = 2^{R+1}$;

3. $\mathbf{v} = [v_0 v_1 \ldots v_{N-1}]^T$ é o vetor correspondente aos índices que selecionam os símbolos de canal para a TCM ou os símbolos do quantizador para a TCQ e $v_k \in \mathcal{V} = \{0, 1, \ldots, M-1\}$ para $k = 0, 1, \ldots, N-1$;

4. $\mathbf{w} = [w_0 w_1 \ldots w_{N-1}]^T$ é o vetor correspondente aos símbolos do quantizador e $w_k \in \mathcal{Q} = \{q_0, q_1, \ldots, q_{M-1}\}$, sendo que $w_k = q_{v_k}$ para $k = 0, 1, \ldots, N-1$. $\mathcal{Q}$ é o conjunto dos níveis de reconstrução do quantizador;

5. $\mathbf{x} = [x_0 x_1 \ldots x_{N-1}]^T$ é o vetor correspondente aos símbolos de canal da TCM e $x_k \in \mathcal{R} = \{r_0, r_1, \ldots, r_{M-1}\}$, sendo que $x_k = r_{v_k}$ para $k = 0, 1, \ldots, N-1$. $\mathcal{R}$ é o conjunto das amplitudes transmitidas através do canal;

6. $\mathbf{e} = [e_0 e_1 \ldots e_{N-1}]^T$ é vetor correspondente ao ruído introduzido pelo canal;

7. $\mathbf{y} = [y_0 y_1 \ldots y_{N-1}]^T$ é o vetor correspondente aos símbolos de canal recebidos e $y_k = x_k + e_k = r_{v_k} + e_k$ para $k = 0, 1, \ldots, N-1$;

8. $\hat{\mathbf{u}} = [\hat{u}_0 \hat{u}_1 \ldots \hat{u}_{N-1}]^T$ é o vetor correspondente à estimativa de $\mathbf{u}$;

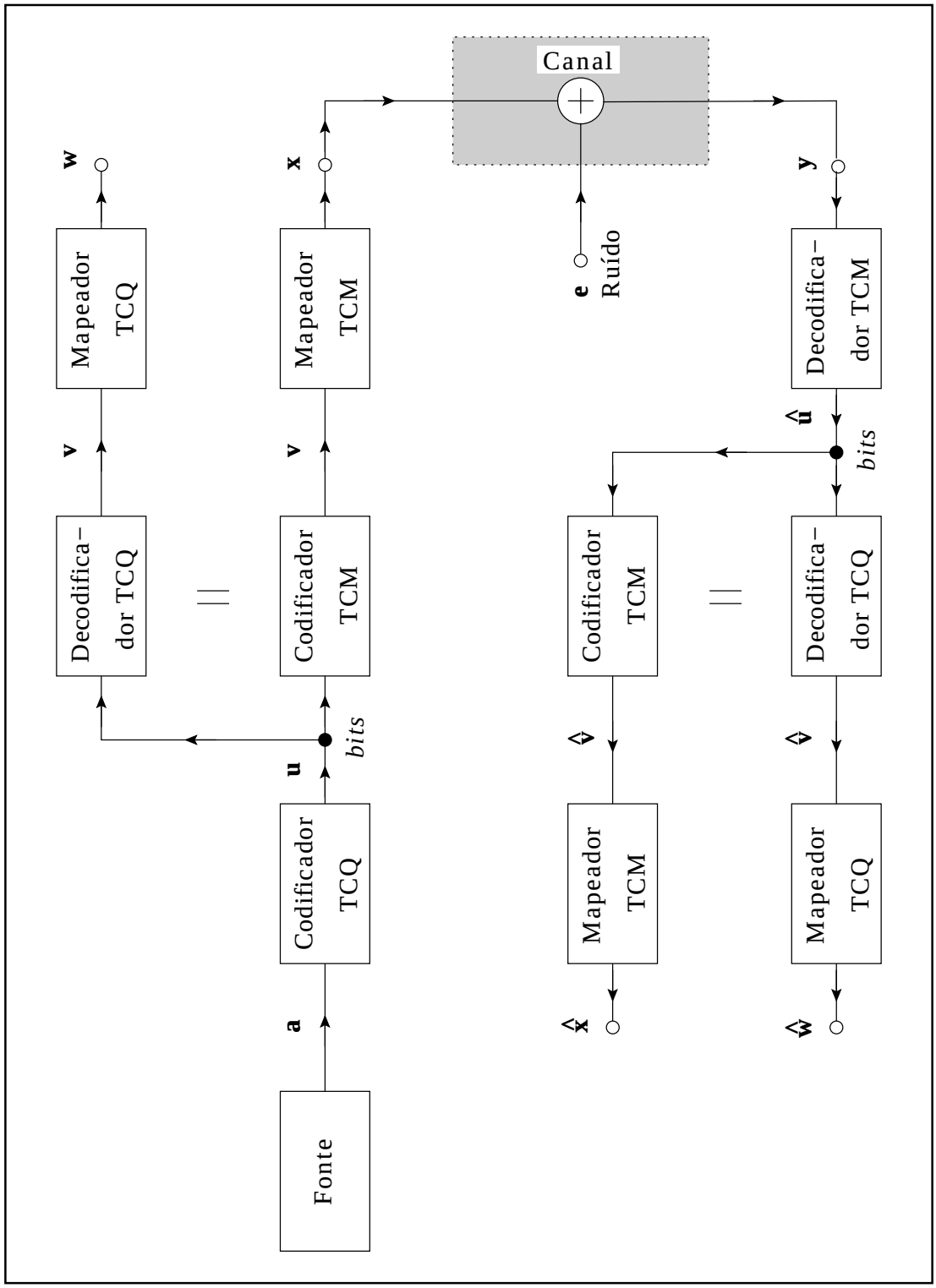


Figura 5.1: Diagrama da codificação conjunta com TCQ e TCM.

9. $\hat{\mathbf{v}} = [\hat{v}_0 \hat{v}_1 \ldots \hat{v}_{N-1}]^T$ é o vetor correspondente à estimativa de $\mathbf{v}$;

10. $\hat{\mathbf{x}} = [\hat{x}_0 \hat{x}_1 \ldots \hat{x}_{N-1}]^T$ é o vetor correspondente à estimativa de $\mathbf{x}$, sendo que $\hat{x}_k = r_{\hat{v}_k}$ para $k = 0, 1, \ldots, N-1$;

11. $\hat{\mathbf{w}} = [\hat{w}_0 \hat{w}_1 \ldots \hat{w}_{N-1}]^T$ é o vetor correspondente à estimativa de $\mathbf{w}$, sendo que $\hat{w}_k = q_{\hat{v}_k}$ para $k = 0, 1, \ldots, N-1$;

A fonte discreta no tempo gera o vetor $\mathbf{a}$ que é codificado pela TCQ, isto é, quantizado, produzindo o vetor de símbolos $\mathbf{u}$ com $R$ *bits* por símbolo. O vetor $\mathbf{u}$ é codificado pela TCM, um processo idêntico ao da decodificação da TCQ, gerando o vetor de índices $\mathbf{v}$ com $R+1$ *bits* por símbolo que, por sua vez, é utilizado pelo mapeador da TCM para selecionar o vetor $\mathbf{x}$ de símbolos de canal a ser transmitido. Como já foi mencionado, em geral, os mapeadores da TCQ e da TCM são distintos, ou seja, $\mathcal{Q} \neq \mathcal{R}$. O vetor $\mathbf{x}$ é corrompido pelo vetor $\mathbf{e}$ de ruído branco gaussiano aditivo, dando origem ao vetor $\mathbf{y} = \mathbf{x} + \mathbf{e}$ que é injetado no decodificador TCM, produzindo uma estimativa $\hat{\mathbf{u}}$ do vetor $\mathbf{u}$. O vetor $\hat{\mathbf{u}}$ passa pelo decodificador TCQ, gerando uma estimativa $\hat{\mathbf{v}}$ do vetor $\mathbf{v}$. Finalmente, o mapeador TCQ produz $\hat{\mathbf{w}}$ a partir de $\hat{\mathbf{v}}$.

Num esquema mais eficiente de codificação conjunta com TCQ e TCM o codificador TCQ é capaz de gerar diretamente o vetor $\mathbf{x}$ e o decodificador TCM é capaz de gerar diretamente o vetor $\hat{\mathbf{w}}$. A figura 5.1 apenas destaca detalhes da estrutura de codificação conjunta com TCQ e TCM, introduzindo uma notação que é util na elaboração de algoritmos de otimização para codificação conjunta de fonte e canal, como os que são apresentados no apêndice A, seções 2 e 3. Do ponto de vista da simulação, a maior parte do tempo de processamento é gasta no codificador TCQ e no decodificador TCM que são essencialmente idênticos neste caso, pois ambos empregam o algoritmo de Viterbi e os mesmos códigos convolucionais.

### 5.1.1 Otimização Conjunta de Fonte e Canal

Em WANG *et al.* [12], um algoritmo de otimização conjunta de fonte e canal é proposto para melhorar o desempenho da codificação conjunta com TCQ e TCM que havia sido introduzida por FISCHER *et al.* [8]. Em FISCHER *et al.* [8], a TCQ e a TCM eram otimizadas separadamente, ou seja, minimizava-se ao valor médio do

erro médio quadrático da TCQ, por meio de um algoritmo como o apresentado na seção A.1 do apêndice A, sem considerar a distorção extra introduzida pelos possíveis erros de canal. Como conseqüência, uma variação relativamente pequena da relação sinal-ruído de canal (CSNR – *Channel Signal to Noise Ratio*), em determinadas faixas de valores, causava uma variação relativamente grande no valor da relação sinal-ruído do quantizador (QSNR – *Quantiser Signal to Noise Ratio*), cujo cálculo é feito utilizando os vetores $\mathbf{a}$ e $\hat{\mathbf{w}}$. Em WANG *et al.* [12], os níveis de reconstrução, pertencentes ao conjunto $\mathcal{Q}$, e os símbolos de canal, pertencentes ao conjunto $\mathcal{R}$, são ajustados conjuntamente através de um algoritmo como o apresentado na seção A.3 do apêndice A. Através desta otimização conjunta é possível atenuar o problema descrito anteriormente e, para taxas maiores que 1 *bit* por amostra-símbolo ($R \geq 2$), esta atenuação é bastante significativa [12].

Entretanto, um algoritmo como o da seção A.3 exige um considerável tempo de processamento, pois é necessária uma nova simulação conjunta com TCQ e TCM toda vez que os símbolos de canal são ajustados para que se possa avaliar a função objetivo, isto é, estimar o valor médio do erro médio quadrático entre os vetores $\mathbf{a}$ e $\hat{\mathbf{w}}$. Então, com o intuito de reduzir este tempo de processamento, propôs-se um algoritmo de otimização parcialmente conjunta. Parcialmente, pois ele leva em consideração a distorção extra introduzida pelos erros de canal mas não ajusta os símbolos de canal em $\mathcal{R}$. O algoritmo, descrito na seção A.2, ajusta os níveis de reconstrução em $\mathcal{Q}$, minimizando diretamente o valor médio do erro médio quadrático entre as seqüências da fonte $\mathbf{a}$ e as correspondentes estimativas das seqüências quantizadas $\hat{\mathbf{w}}$.

## 5.2 Codificação Conjunta de Fonte e Canal com Turbo TCQ e Turbo TCM

O esquema de codificação conjunta que emprega turbo TCQ e turbo TCM é, na verdade, semelhante ao mostrado na figura 5.1, bastando substituir TCQ por TTCQ e TCM por TTCM. Entretanto, o codificador TTCQ utiliza o SOVA em vez do HOVA comumente adotado na TCQ e o decodificafor TTCM pode fazer uso tanto do SOVA quanto do algoritmo BCJR. De fato, tanto o SOVA quanto o algoritmo

BCJR serão testados nas simulações deste capítulo referentes à codificação conjunta com TTCQ e TTCM. Para simplificar a notação utilizada, a codificação conjunta de fonte e canal com TCQ e TCM será designada simplesmente por TCQ & TCM. Analogamente, a codificação conjunta de fonte e canal com TTCQ e TTCM será designada por TTCQ & TTCM. Se a TTCM utilizar o SOVA será empregada a notação TTCQ & TTCM-SOVA. Caso a TTCM use o algoritmo BCJR será empregada a notação TTCQ & TTCM-BCJR.

## 5.3 Resultados

### 5.3.1 Algoritmos de Otimização Conjunta Fonte e Canal

Nesta subseção, o algoritmo de otimização parcialmente conjunta (seção A.2) e o algoritmo de WANG *et al.* [12] (representado aqui pelo algoritmo da seção A.3) são comparados. Para efetuar esta comparação, será utilizada a TCQ & TCM com o código $(13, 4)$ de oito estados. A fonte é gaussiana de média zero e variância unitária. As taxas testadas serão de 1 *bit* por amostra-símbolo e 2 *bits* por amostra-símbolo. Esta configuração específica é utilizada para que possa ser feita uma comparação com os resultados disponíveis em WANG *et al.* [12]. Cada um dos pontos nos gráficos das três figuras mencionadas a seguir é o resultado do cálculo da média oriunda da simulação de $N_s = 100$ seqüências de tamanho $N = 1000$ de realizações independentes de um gerador de distribuição gaussiana. No algoritmo de otimização parcialmente conjunta, cada iteração realiza uma simulação deste tipo. Já no algoritmo da seção A.3, cada vez que a função objetivo é avaliada uma simulação como esta é executada.

Na figura 5.2, são mostrados os resultados para a taxa de 1 *bit* por amostra-símbolo, relativos à comparação dos algoritmos das seções A.2 e A.3 para otimização conjunta da TCQ & TCM. Neste caso, é possível observar, pelo gráfico, que o algoritmo da seção A.3 é um pouco melhor que o da seção A.2 na faixa de CSNR entre 2,0 e 5,0 dB. Na faixa de CSNR de 5,0 a 6,5 dB, o algoritmo da seção A.2 produz resultados de QSNR ligeiramente inferiores (de $-0{,}03$ a $-0{,}02$ dB) aos da TCQ & TCM otimizada separadamente, enquanto que o algoritmo da seção A.3 produz resultados de QSNR ligeiramente superiores (de $+0{,}01$ a $+0{,}03$ dB) aos

da TCQ & TCM otimizada separadamente. Acima de 7,0 dB de CSNR nenhum dos algoritmos se destaca. Nesta região, ocorrem poucos erros de canal e a QSNR "satura" no valor da distorção média da TCQ otimizada.

Na figura 5.3, são mostrados os resultados para a taxa de 2 *bits* por amostra-símbolo, ainda relativos à comparação dos algoritmos das seções A.2 e A.3 para otimização conjunta da TCQ & TCM. Agora é possível observar claramente que o algoritmo da seção A.3 é capaz de produzir resultados bem melhores que os do algoritmo da seção A.2 na região onde a QSNR varia de forma mais abrupta com a CSNR (entre 10,0 e 12,0 dB de CSNR), chegando a diferenças de QSNR da ordem de 1 dB. Também se vê claramente que, na região entre 11,0 e 13,5 dB de CSNR, o algoritmo da seção A.2 não consegue superar a QSNR da TCQ & TCM otimizada separadamente. Na faixa onde a variação não é tão abrupta (entre 2,0 e 7,0 dB de CSNR), ambos os algoritmos têm desempenho semelhante. Portanto, o algoritmo da seção A.3 parece ser mais eficiente na região de transição abrupta, onde os erros de canal são ainda relativamente pouco freqüentes. Nesta região, ajustar os símbolos de canal em $\mathcal{R}$ permite "controlar" os erros de canal de tal forma que a distorção extra introduzida pelos mesmos na quantização pode ser minimizada. Não se pode fazer isto minimizando somente a distorção entre $\mathbf{a}$ e $\hat{\mathbf{w}}$ sem ajustar os símbolos de canal como é feito no algoritmo da seção A.2. Por outro lado, na região onde os erros de canal são relativamente freqüentes (entre 2,0 e 7,0 dB de CSNR), não é mais possível "controlar" os erros de canal pelo ajuste dos símbolos de canal e o algoritmo da seção A.3 produz resultados semelhantes aos do algoritmo da seção A.2.

Na tabela 5.1, compara-se o número de simulações executadas pelos algoritmos das seções A.2 e A.3 na obtenção dos pontos da figura 5.2. Na tabela 5.2, faz-se uma comparação semelhante com relação aos pontos da figura 5.3. Em média, o algoritmo da seção A.2 executou 8,3% do número de simulações do algoritmo da seção A.3 no caso da figura 5.2 e 10% no caso da figura 5.3. Evidentemente, isto comprova que houve uma redução considerável no tempo de processamento, proporcionada pelo algoritmo da seção A.2, como era desejado. Todavia, o que se ganha em termos de tempo de processamento com o algoritmo da seção A.2 não compensa a redução de QSNR resultante, principalmente no caso da figura 5.3.

Na figura 5.4, os resultados obtidos pelo algoritmo da seção A.3 são compa-

Tabela 5.1: Comparando o número de simulações executadas pelo algoritmo de otimização parcialmente conjunta da seção A.2 ($N_{SE_A}$) e pelo algoritmo de otimização conjunta da seção A.3 ($N_{SE_B}$) na obtenção dos pontos do gráfico da figura 5.2.

| **CSNR (dB)** | **Seção A.2** ($N_{SE_A}$) | **Seção A.3** ($N_{SE_B}$) | $100\left(\frac{N_{SE_A}}{N_{SE_B}}\right)$ |
|---|---|---|---|
| 2,0 | 7 | 161 | 4,3 |
| 2,5 | 8 | 105 | 6,6 |
| 3,0 | 6 | 137 | 4,4 |
| 3,5 | 6 | 133 | 4,5 |
| 4,0 | 6 | 133 | 4,5 |
| 4,5 | 9 | 94 | 9,6 |
| 5,0 | 10 | 68 | 14,7 |
| 5,5 | 13 | 75 | 17,3 |
| 6,0 | 9 | 51 | 17,6 |
| 6,5 | 11 | 62 | 17,7 |
| **Média** | 8,5 | 101,9 | 8,3 |

rados com os de WANG *et al.* [12]. A figura mostra que o algoritmo da seção A.3 produziu resultados notadamente melhores que os relatados por WANG *et al.* [12]. Isto pode ser uma conseqüência da utilização de uma seqüência de treinamento maior ($N_sN = 100000$ amostras por simulação contra 10000 amostras por simulação de WANG *et al.* [12]). Mas acredita-se que a utilização de vários pares de parâmetros $\delta_D$ e $\theta$ (ver seção A.3) seja a principal razão destes resultados mais expressivos. Esta estratégia permite encontrar o par $(\delta_D, \theta)$ que torna o algoritmo de otimização da seção A.3 mais eficaz. Conseqüentemente, o ponto de mínimo encontrado é melhor.

### 5.3.2 Codificação Conjunta com Turbo TCQ e Turbo TCM

Prosseguindo, serão comparados os resultados da TCQ & TCM com os da TTCQ & TTCM para as fontes uniforme e gaussiana de média zero e variância unitária. Nas simulações efetuadas, tanto a TCQ & TCM quanto a TTCQ & TTCM utilizaram o código convolucional de 4 estados $(7, 2)$ e taxa $R = 1$ *bit* por amostra-

Tabela 5.2: Comparando o número de simulações executadas pelo algoritmo de otimização parcialmente conjunta da seção A.2 ($N_{SE_A}$) e pelo algoritmo de otimização conjunta da seção A.3 ($N_{SE_B}$) na obtenção dos pontos do gráfico da figura 5.3.

| **CSNR (dB)** | **Seção A.2** ($N_{SE_A}$) | **Seção A.2** ($N_{SE_B}$) | $100\left(\frac{N_{SE_A}}{N_{SE_B}}\right)$ |
|---|---|---|---|
| 2,0 | 8 | 222 | 3,6 |
| 2,5 | 19 | 306 | 6,2 |
| 3,0 | 51 | 187 | 27,2 |
| 3,5 | 11 | 193 | 5,7 |
| 4,0 | 24 | 191 | 12,6 |
| 4,5 | 10 | 173 | 5,8 |
| 5,0 | 7 | 187 | 3,7 |
| 5,5 | 9 | 161 | 5,6 |
| 6,0 | 9 | 183 | 4,9 |
| 6,5 | 9 | 197 | 4,6 |
| 7,0 | 10 | 173 | 5,8 |
| 7,5 | 9 | 227 | 4,0 |
| 8,0 | 15 | 242 | 6,2 |
| 8,5 | 51 | 204 | 25,0 |
| 9,0 | 51 | 245 | 20,8 |
| 9,5 | 9 | 239 | 3,8 |
| 10,0 | 11 | 181 | 6,1 |
| 10,5 | 10 | 57 | 17,5 |
| 11,0 | 12 | 60 | 20,0 |
| 11,5 | 18 | 67 | 26,9 |
| 12,0 | 15 | 73 | 20,5 |
| 12,5 | 17 | 75 | 22,7 |
| **Média** | 17,5 | 174,7 | 10,0 |

símbolo. A TTCQ executa $N_{it} = 10000$ iterações para cada seqüência produzida pela fonte e a TTCM executa $N_{it} = 4$ iterações. Devido ao elevado número de iterações exigido pela TTCQ, somente a otimização realizada separadamente foi testada. O algoritmo utilizado na TTCQ é o serial refinado proposto na seção 4.2.2. Cada ponto dos gráficos nas figuras que comparam a TCQ & TCM com a TTCQ & TTCM é o resultado do cálculo da média oriunda da simulação de $N_s = 156000/N$ seqüências de tamanho $N$ de realizações independentes de um gerador de distribuição uniforme ou gaussiana, onde o valor de $N$ pode ser 104, 256 ou 1000.

Na figura 5.5, é comparado o desempenho da TCQ & TCM com o da TTCQ & TTCM-BCJR/SOVA para fonte uniforme. A TCQ & TCM emprega blocos de tamanho $N = 1000$ e a TTCQ & TTCM, blocos de tamanho $N = 256$. Como se pode ver pela figura, a TTCQ & TTCM-BCJR tem um desempenho melhor que o da TTCQ & TTCM-SOVA em cerca de 0,3 dB de QSNR na faixa de CSNR entre 5,5 e 6,5 dB. Isto permite aumentar a faixa em que a TTCQ & TTCM supera a TCQ & TCM em cerca de 0,3 dB em termos de CSNR. Para uma CSNR menor que 6,5 dB, os erros de canal cometidos pela TTCM são maiores que os da TCM como mostra a figura 5.6. Nesta figura, pode-se ver que a diferença entre a taxa de erro de *bits* é mais acentuada na faixa entre 3,5 e 5,5 dB de CSNR. Na figura 5.5, nota-se uma depressão acentuada da curva da TTCQ & TTCM em relação à curva da TCQ & TCM na mesma faixa. Abaixo de 3,5 dB, a diferença da taxa de erro de *bits* entre TCM e TTCM volta a diminuir (ver figura 5.6), o que se reflete no gráfico da TTCQ & TTCM (ver figura 5.5) cuja diferença de QSNR em relação à TCQ & TCM também se reduz.

Finalmente, na figura 5.7, compara-se o desempenho da TCQ & TCM com o da TTCQ & TTCM-BCJR/SOVA para fonte gaussiana. A TCQ & TCM emprega novamente blocos de tamanho $N = 1000$ e a TTCQ & TTCM, blocos de tamanho $N = 104$. Nota-se que acima de 6,5 dB de CSNR a TTCQ & TTCM produz resultados melhores que os da TCQ & TCM. Entretanto, abaixo deste valor de CSNR, a situação se inverte de forma semelhante ao que ocorreu para a fonte uniforme. A explicação para este comportamento é também semelhante àquela dada para a fonte uniforme no parágrafo anterior.

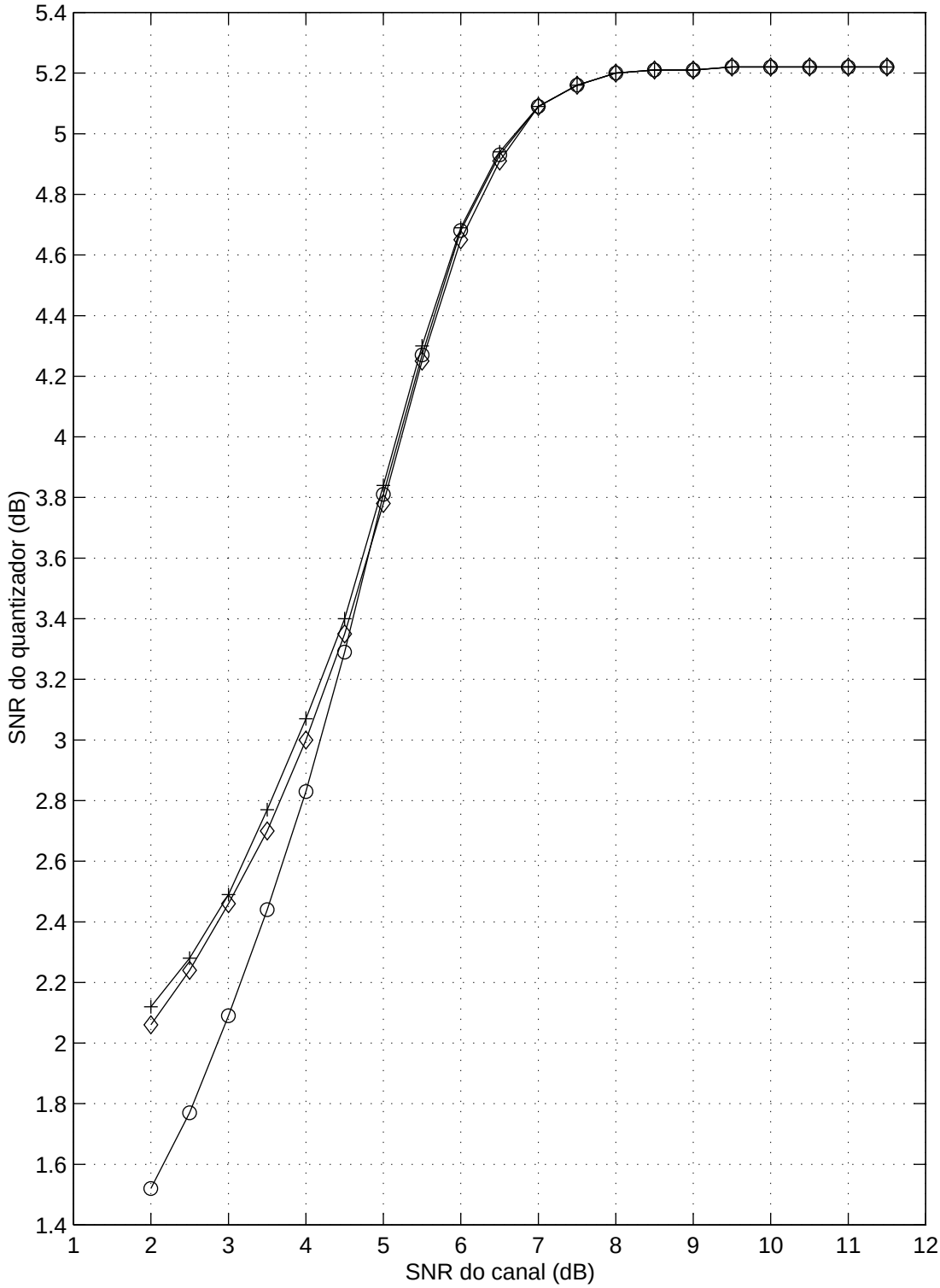


Figura 5.2: Algoritmos de otimização para TCQ & TCM. ($\circ$) Desconsiderando a distorção extra introduzida pelos erros de canal. ($\diamond$) Considerando parcialmente os erros de canal. ($+$) Considerando integralmente os erros de canal. Código $(13, 4)$ com 1 *bit* por amostra-símbolo e fonte gaussiana sem memória de média zero e variância unitária.

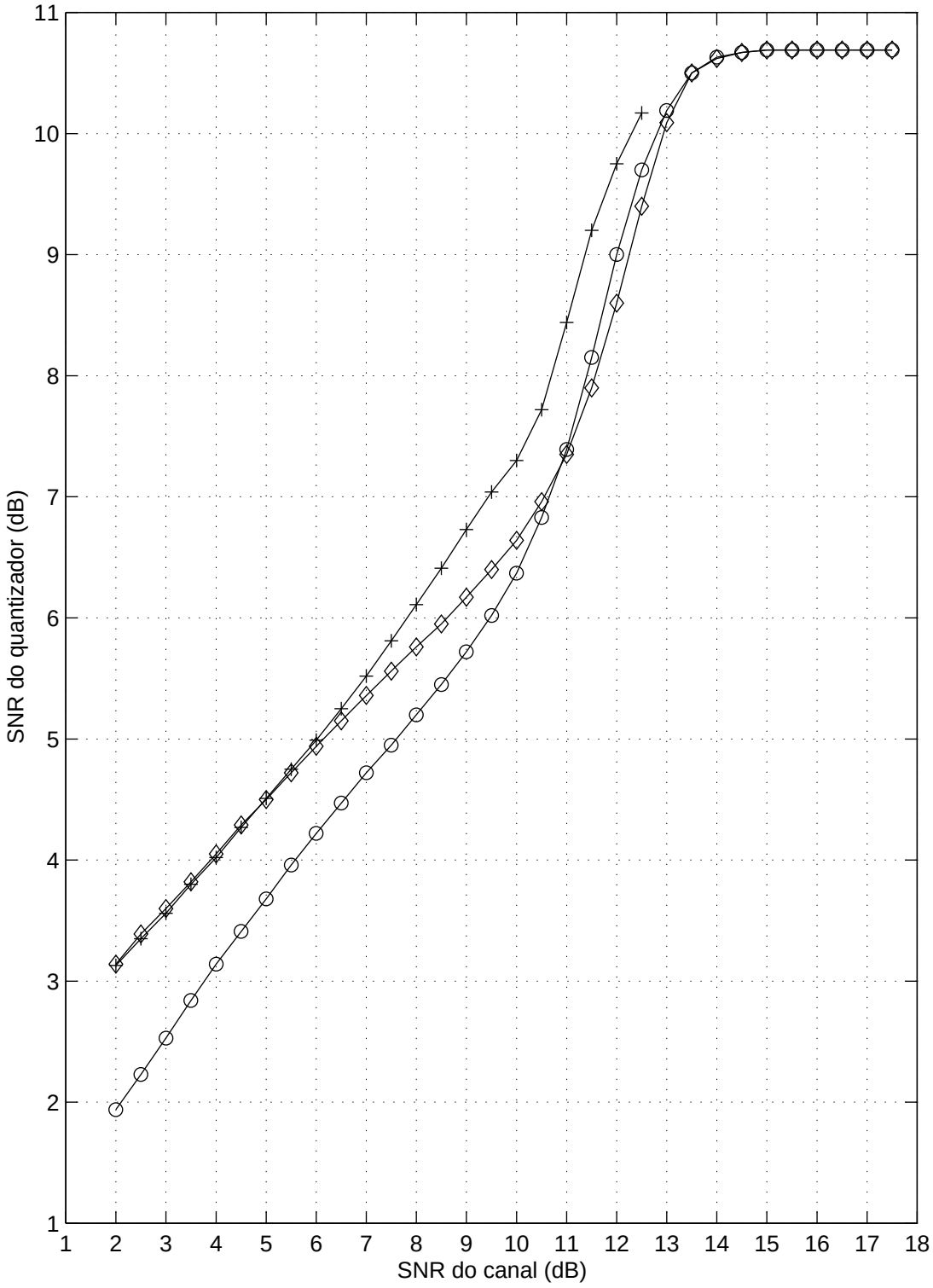


Figura 5.3: Algoritmos de otimização para TCQ & TCM. ($\circ$) Desconsiderando a distorção extra introduzida pelos erros de canal. ($\diamond$) Considerando parcialmente os erros de canal. ($+$) Considerando integralmente os erros de canal. Código $(13, 4)$ com 2 *bits* por amostra-símbolo e fonte gaussiana sem memória de média zero e variância unitária.

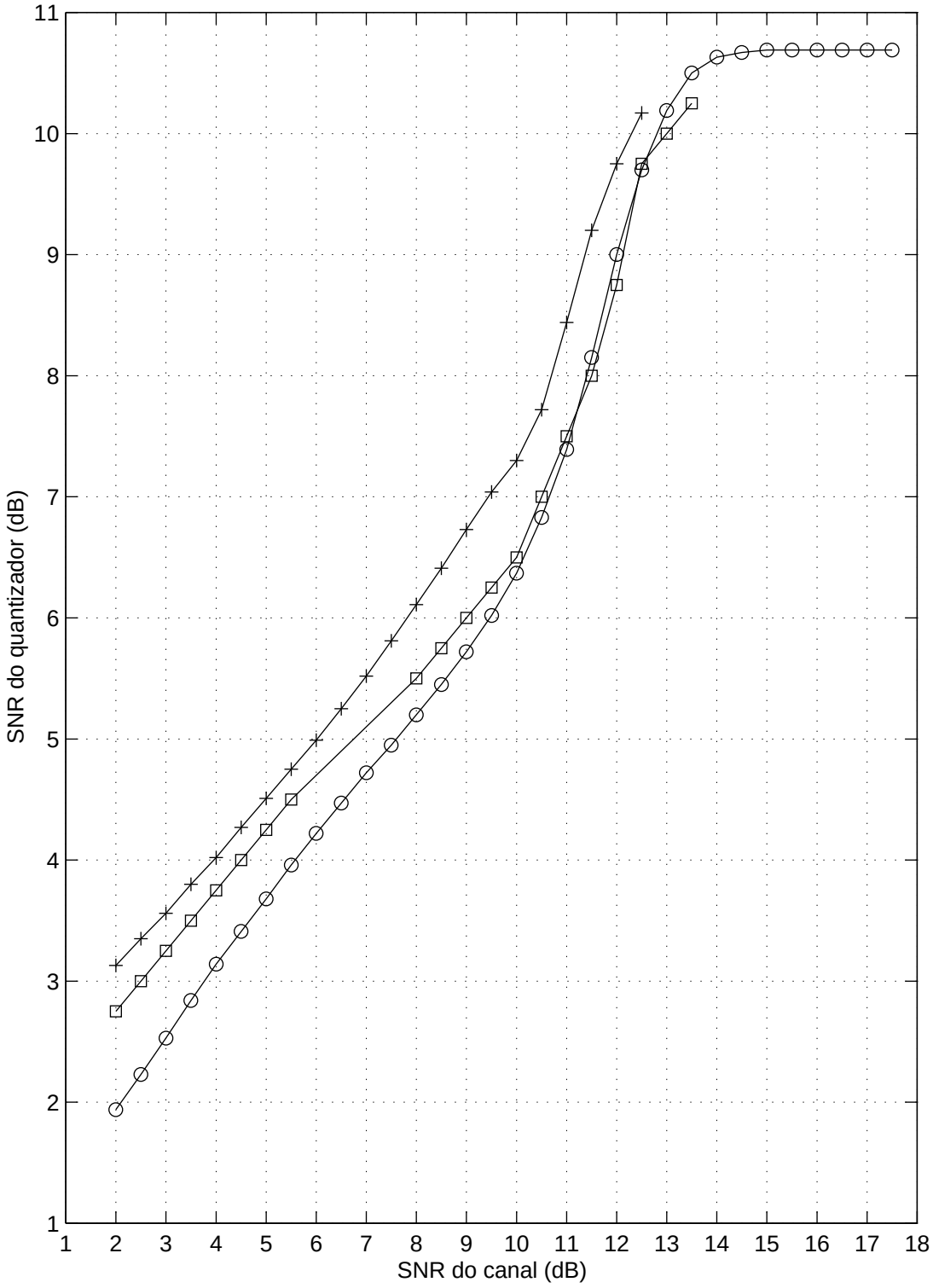


Figura 5.4: Algoritmos de otimização para TCQ & TCM. (○) Desconsiderando a distorção extra introduzida pelos erros de canal. Considerando integralmente os erros de canal: (□) resultados de Wang e (+) resultados obtidos neste trabalho. Código $(13, 4)$ com 2 *bits* por amostra-símbolo e fonte gaussiana sem memória de média zero e variância unitária.

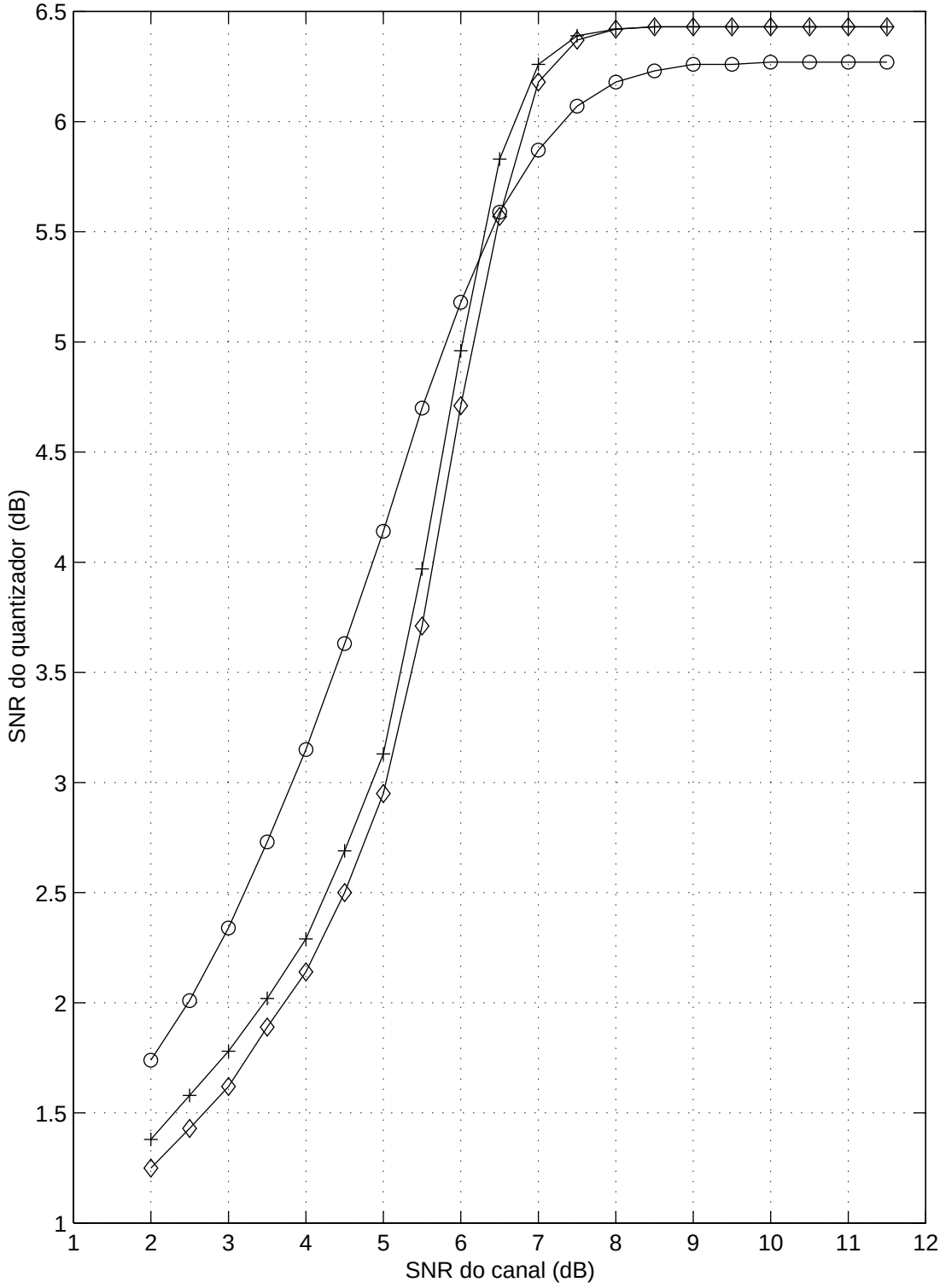


Figura 5.5: Comparando TCQ & TCM, TTCQ & TTCM-SOVA e TTCQ & TTCM-BCJR. ($\circ$) TCQ & TCM com bloco de tamanho 1000, ($\diamond$) TTCQ & TTCM-SOVA e ($+$) TTCQ & TTCM-BCJR, ambas com bloco de tamanho 256. Fonte uniforme sem memória de média zero e variância unitária.

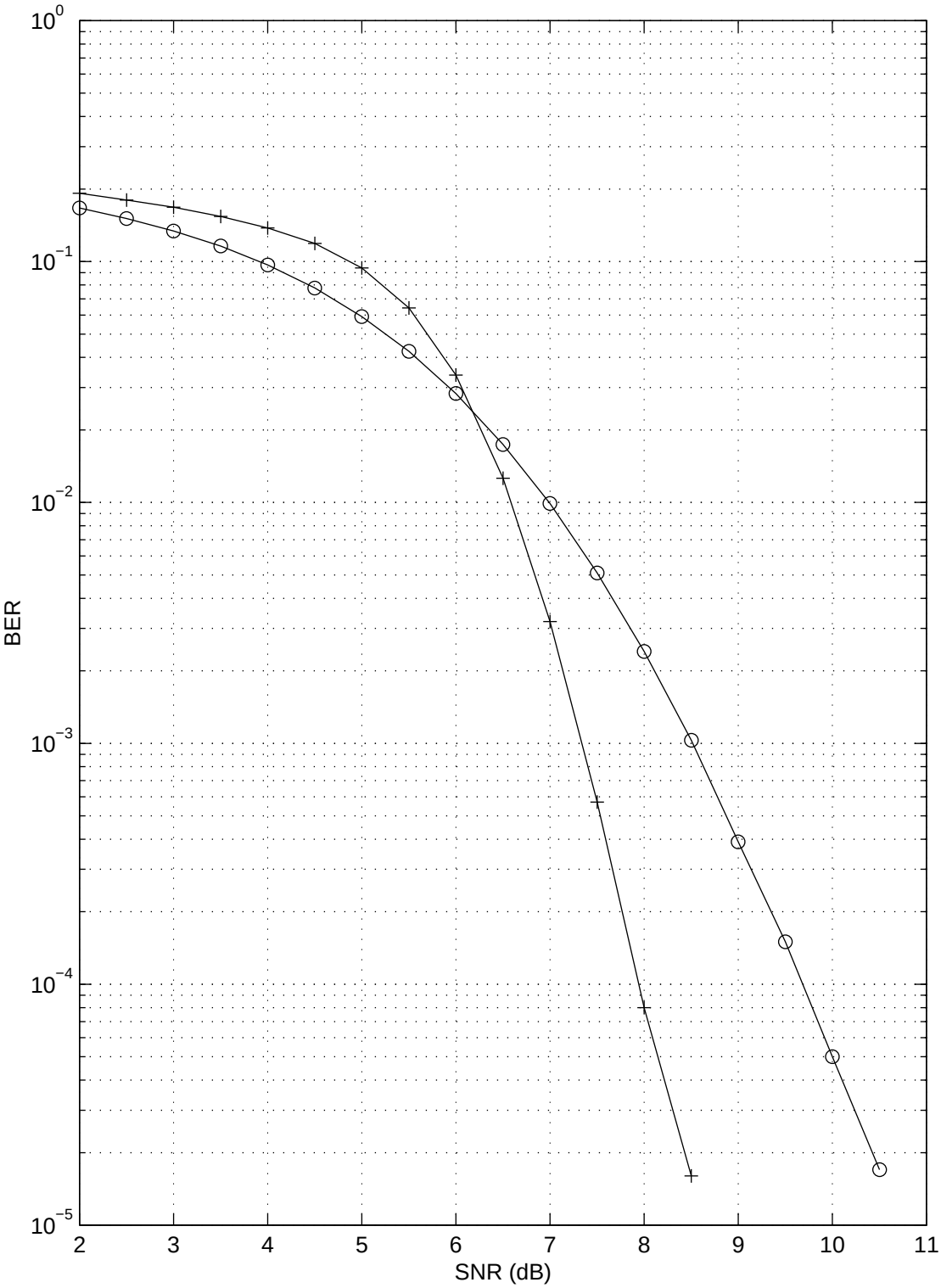


Figura 5.6: ($\circ$) TCM-HOVA, código convolucional $(7, 2)$ e $N = 1000$. ($+$) TTCM-SOVA, 4 iterações, código convolucional $(7, 2)$ e $N = 256$.

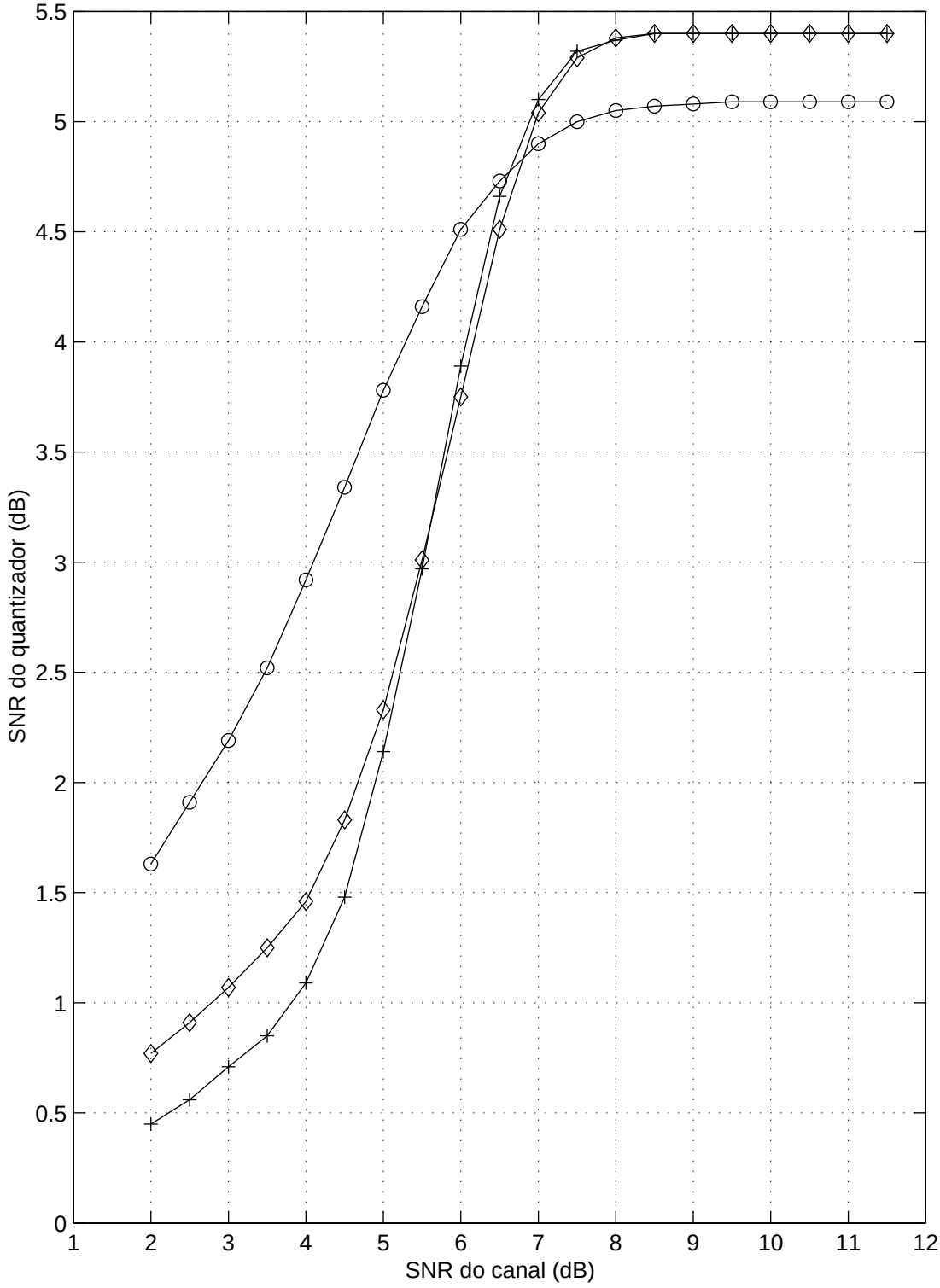


Figura 5.7: Comparando TCQ & TCM, TTCQ & TTCM-SOVA e TTCQ & TTCM-BCJR. ($\circ$) TCQ & TCM com bloco de tamanho 1000, ($\diamond$) TTCQ & TTCM-SOVA e ($+$) TTCQ & TTCM-BCJR, ambas com bloco de tamanho 104. Fonte gaussiana sem memória de média zero e variância unitária.

## 5.4 Resumo

Neste capítulo, um algoritmo de otimização parcialmente conjunta (seção A.2) para codificação conjunta de fonte e canal com TCQ e TCM (ou com TTCQ e TTCM) foi proposto com o objetivo de reduzir o tempo de processamento do algoritmo otimização conjunta de WANG *et al.* [12]. Apesar do tempo de processamento ter sido reduzido consideravelmente, o desempenho do algoritmo de otimização parcialmente conjunta se degrada sensivelmente para certas faixas de relação sinal-ruído de canal e o que se ganha em termos de tempo de processamento não compensa a grande redução da relação sinal-ruído do quantizador nestas faixas.

O desempenho conjunto da TTCQ e TTCM na codificação conjunta de fonte e canal também foi avaliado neste capítulo. Uma comparação com o desempenho conjunto da TCQ e TCM mostrou que aparentemente há uma faixa de relação sinal-ruído de canal relativamente alta para a qual a TTCQ & TTCM é notadamente superior à TCQ & TCM. Na faixa restante, a TCQ & TCM é claramente melhor. Isto ocorre porque a TTCM tem uma taxa de erro de *bits* maior que a da TCM nesta última região, causando esta inversão de desempenho. Mesmo assim, se o elevado tempo de codificação da TTCQ for reduzido futuramente, a TTCQ & TTCM será capaz de competir com a TCQ & TCM, ao menos na faixa de relação sinal-ruído de canal relativamente alta.

# Capítulo 6

# Conclusões

Um novo esquema de quantização foi proposto, a turbo quantização codificada por treliças (TTCQ – *Turbo TCQ*), derivado do esquema de codificação de canal de ROBERTSON *et al.* [7], a turbo modulação codificada por treliças (TTCM – *Turbo TCM*). Porém, a implementação da TTCQ feita diretamente a partir da TTCM, seguindo passos semelhantes aos de MARCELLIN [6] que desenvolveu a TCQ a partir da TCM, ou seja, utilizando simplesmente a estrutura do codificador e do decodificador da TTCM, com a mesma permutação pseudo-aleatória, com o mesmo número de iterações e com o mesmo tamanho de bloco, como quantizador não produziu resultados satisfatórios. Neste caso, constatou-se que a TTCQ necessitava do quadrado do número de estados para produzir resultados semelhantes aos da TCQ.

No entanto, em 2003, CHAPPELIER *et al.* [23], investigando também a possibilidade de usar a estrutura da TTCM como quantizador, observaram que era possível ultrapassar significativamente o desempenho da TCQ quando se empregava um permutador $S$-aleatório ($S$-*random interleaver*), um número de iterações milhares de vezes maior que aquele adotado na TTCM e blocos de tamanho médio. Com base nestas observações, foram propostos dois novos algoritmos de codificação no presente trabalho que melhoraram ainda mais o desempenho da TTCQ em termos de tempo de codificação e da distorção média produzida.

O primeiro algoritmo de codificação para TTCQ proposto foi o serial refinado, descrito na subseção 4.2.2. Este algoritmo é o resultado do refinamento do algoritmo serial de CHAPPELIER *et al.* [23] e introduz um critério de parada que avalia se o

ponto fixo da TTCQ foi encontrado. Com isto, foi possível reduzir significativamente o número de iterações executadas durante o processo de codificação da TTCQ e, conseqüentemente, o tempo de codificação do algoritmo serial refinado se tornou bem menor que o do algoritmo serial de CHAPPELIER *et al.* [23] para os casos testados. Acredita-se que esta redução no tempo de codificação possa ser ainda maior se o valor do parâmetro $\epsilon$, que controla o critério de parada do algoritmo serial refinado, for determinado experimentalmente de modo a minimizar o tempo de codificação sem aumentar o valor da distorção média da TTCQ.

O segundo algoritmo de codificação para TTCQ proposto foi o paralelo, descrito na subseção 4.2.3. Ele permite executar duas instâncias distintas do algoritmo serial refinado em paralelo, explorando a estrutura do codificador iterativo da TTCQ. A instância que produz a menor distorção média é selecionada e, com isto, é possível obter uma distorção média que é menor que a do algoritmo serial refinado e, por conseqüência, menor que a do algoritmo serial de CHAPPELIER *et al.* [23].

Adicionalmente, foi realizada a otimização dos níveis de reconstrução da TTCQ, de modo semelhante ao que MARCELLIN [6] fez com a TCQ, e foram feitas simulações com as fontes gaussiana e laplaciana, visto que no trabalho de CHAPPELIER *et al.* [23] somente a TTCQ não otimizada com fonte uniforme havia sido testada. Constatou-se novamente que a TTCQ supera significativamente a TCQ. Todavia, a corrente limitação do tamanho do bloco e, principalmente, o ainda elevado número de iterações, acarretando um tempo de processamento muito maior que o da TCQ, certamente representam um problema para várias aplicações.

Um algoritmo de otimização parcialmente conjunta (ver seção A.2) para codificação conjunta de fonte e canal com TCQ e TCM (ou com TTCQ e TTCM) também foi proposto neste trabalho com o objetivo de reduzir o tempo de processamento do algoritmo otimização conjunta de WANG *et al.* [12]. Apesar do tempo de processamento ter sido reduzido consideravelmente, o desempenho do algoritmo de otimização parcialmente conjunta se degrada sensivelmente para certas faixas de relação sinal-ruído de canal e o que se ganha em termos de tempo de processamento não compensa a grande redução da relação sinal-ruído do quantizador nestas faixas.

O desempenho conjunto da TTCQ e TTCM na codificação conjunta de fonte e canal também foi avaliado neste trabalho. Uma comparação com o desempenho

conjunto da TCQ e TCM mostrou que aparentemente há uma faixa de relação sinal-ruído de canal relativamente alta para a qual a TTCQ & TTCM é notadamente superior à TCQ & TCM. Na faixa restante, a TCQ & TCM é claramente melhor. Isto ocorre porque a TTCM tem uma taxa de erro de *bits* maior que a da TCM nesta última região, causando esta inversão de desempenho. Mesmo assim, se o elevado tempo de codificação da TTCQ for reduzido futuramente, a TTCQ & TTCM será capaz de competir com a TCQ & TCM, ao menos na faixa de relação sinal-ruído de canal relativamente alta.

## 6.1 Propostas para Trabalhos Futuros

A primeira sugestão seria combinar o algoritmo de CHAPPELIER *et al.* [23], que muda os rótulos da treliça utilizada pela TTCQ, com o algoritmo paralelo para TTCQ proposto neste trabalho, como descrito na seção 4.2.3. Como separadamente ambos produzem resultados melhores que o algoritmo serial da seção 4.2.2 em termos de distorção média, seria interessante verificar o desempenho conjunto.

Um problema que pode afetar o desempenho da TTCQ é a escolha do ponto inicial dado ao codificador na forma da informação *a priori* $\mathbf{L}_1^{a(0)}$ que é utilizada nos algoritmos da seção 4.2. Neste trabalho, foi utilizada a estimativa de ROBERTSON *et al.* [11] da subseção 4.1.2 para a seleção de $\mathbf{L}_1^{a(0)}$. Uma possível alternativa seria a utilização de $N_{\mathbf{L}}$ condições inicias $\mathbf{L}_1^{a(0)}$ selecionadas aleatoriamente em torno da produzida pela estimativa da subseção 4.1.2. O objetivo é fazer $N_{\mathbf{L}}$ TTCQs e selecionar aquela que produzir a menor distorção. Seria interessante incluir entre os $\mathbf{L}_1^{a(0)}$ gerados aleatoriamente aquele gerado pela estimativa de seção 4.1.2 para verificar com que freqüência esta escolha é melhor que a aleatória.

Com relação à codificação conjunta de fonte e canal, seria interessante testar a combinação da TTCQ com a TCM e compará-la com a da TCQ com a TCM e com a da TTCQ com a TTCM. Visto que a TCM supera a TTCM quando se utiliza uma relação sinal-ruído de canal relativamente baixa, é possível que a combinação da TTCQ com a TCM possa produzir resultados melhores que os da TCQ com a TCM em todas as faixas de relação sinal-ruído.

# Referências Bibliográficas


[1] SHANNON, C. E., "A Mathematical Theory of Communication", *Bell System Technical Journal*, v. 27, pp. 379–423 e 623–656, 1948.

[2] HEEGARD, C., WICKER, S. B., *Turbo Coding*. 1 ed. Norwell, MA, USA, Kluwer Academic Publishers, 1999.

[3] BERROU, C., GLAVIEUX, A., THITIMAJSHIMA, P., "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes". In: *IEEE ICC 1993 - IEEE Iternational Conference on Comunication, 1993*, pp. 1064–1070, Geneva, Switzerland, May 1993.

[4] COVER, T. M., THOMAS, J. A., *Elements of Information Theory*. 1 ed. New York, USA, John Wiley & Sons, Inc., 1991.

[5] UNGERBOECK, G., "Channel Coding with Multilevel/Phase Signals", *IEEE Transactions on Information Theory*, v. IT-28, n. 1, pp. 55–67, January 1982.

[6] MARCELLIN, M. W., *Trellis Coded Quantization: An Efficient Technique for Data Compression*. Ph.D. dissertation, Texas A&M University, Texas, USA, 1987.

[7] ROBERTSON, P., WÖRZ, T., "Coded Modulatiom Scheme Employing Turbo Codes", *Electronics Letters*, v. 31, n. 18, pp. 1546–1547, August 1995.

[8] FISCHER, T. R., MARCELLIN, M. W., "Joint Trellis Coded Quantization/Modulation", *IEEE Transactions on Communications*, v. 39, n. 2, pp. 172–176, February 1991.

[9] BAHL, L. R., COCKE, J., JELINEK, F., *et al.*, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", *IEEE Transactions on Information Theory*, v. IT-20, n. 2, pp. 248–287, March 1974.

[10] VUCETIC, B., YUAN, J., *Turbo Codes: Principles and Applications*. 1 ed. Norwell, MA, USA, Kluwer Academic Publishers, 2001.

[11] ROBERTSON, P., WÖRZ, T., "Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes", *IEEE Journal on Selected Areas in Communications*, v. 16, n. 2, pp. 206–218, February 1998.

[12] WANG, M., FISCHER, T. R., "Trellis-Coded Quantization Designed for Noisy Channels", *IEEE Transactions on Information Theory*, v. 40, n. 6, pp. 1792–1802, November 1994.

[13] DIVSALAR, D., POLLARA, F., *Multiple Turbo Codes for Deep-Space Communications*, TDA - Telecommunications and Data Acquisition Progress Report 42-121, California Institute of Technology, Jet Propulsion Laboratory, Passadena, California 91109, USA, May 1995. pp. 66-77.

[14] LEE, L. H. C., *Convolutional Coding: Fundamentals and Applications*. 1 ed. Norwood, MA, USA, Artech House, Inc., 1997.

[15] WICKER, S. B., *Error Control Systems for Digital Communication and Storage*. 1 ed. Upper Sadle River, New Jersey, USA, Prentice-Hall, 1995.

[16] DIVSALAR, D., POLLARA, F., *On Design of Turbo Codes*, TDA - Telecommunications and Data Acquisition Progress Report 42-123, California Institute of Technology, Jet Propulsion Laboratory, Passadena, California 91109, USA, November 1995. pp. 99-121.

[17] DIVSALAR, D., POLLARA, F., "Turbo Codes for PCS Applications". *IEEE ICC 1995 - IEEE International Conference on Communication, 1995, Gateway to Globalization*, Seattle, 18-22 June 1995.

[18] DIVSALAR, D., POLLARA, F., "Multiple Turbo Codes". *MILCOM 1995 - Military Communications Conference, 1995. Conference Record, IEEE.*, 6 November 1995.

[19] UNGERBOECK, G., "Trellis-Coded Modulation with Redundant Signal Sets: Part I - Introduction", *IEEE Communications Magazine*, v. 25, n. 2, pp. 5–11, February 1987.

[20] JAIN, A. K., *Fundamentals of Digital Image Processing*. 1 ed. Englewood Cliffs, NJ, USA, Prentice-Hall, Inc., 1989.

[21] CONWAY, J. H., SLOANE, N. J. A., *Sphere Packings, Lattices and Groups*. New York, USA, Springer Verlag, 1988.

[22] UNGERBOECK, G., "Trellis-Coded Modulation with Redundant Signal Sets: Part II - State of Art", *IEEE Communications Magazine*, v. 25, n. 2, pp. 12–21, February 1987.

[23] CHAPPELIER, V., GUILLEMOT, C., MARINKOVIĊ, S., "Turbo Trellis-Coded Quantization". *3rd International Symposium on Turbo Codes & Related Topics*, Brest, France, 1-5 September 2003.

[24] RICHARDSON, T., "The Geometry of Turbo-Decoding Dynamics", *IEEE Transactions on Information Theory*, v. 46, n. 1, pp. 9–23, January 2000.

[25] HIMMELBLAU, D. M., *Applied Nonlinear Programming*. 1 ed. New York, USA, McGraw-Hill, Inc., 1972.

[26] LU, W.-S., ANTONIOU, A., *Two-Dimensional Digital Filters*. 1 ed. New York, NY, USA, Marcel Dekker, Inc., 1992.

[27] PAPOULIS, A., *Probability, Random Variables and Stochastic Processes*. 3 ed. Cingapura, MaGraw-Hill, Inc., 1991.

# Apêndice A

# Algoritmos

Este apêndice contém os algoritmos utilizados na obtenção dos resultados dos capítulos 4 e 5. Eles são baseados nas informações pouco detalhadas encontradas em MARCELLIN [6], para a TCQ, e em WANG *et al.* [12], para a codificação conjunta de fonte e canal com TCQ e TCM. Desta forma, os algoritmos aqui apresentados são também uma contribuição desta tese. O algoritmo de busca linear utilizado é o DSC Powell [25].

## A.1 Otimização da TCQ ou da TTCQ

Note que são feitas $N_s$ simulações com vetores de tamanho $N$ para calcular o intervalo de confiança da distorção média produzida pela TCQ ou TTCQ. Nesta seção, estes vetores foram concatenados, gerando um vetor de tamanho $N_t = N_s N$.

Sejam $R$ e $M = 2^{R+1}$ o número de *bits* por amostra e o número de níves de quantização da TCQ (ou TTCQ), respectivamente. Deseja-se minimizar o erro médio quadrático entre o vetor de símbolos da fonte $\mathbf{a} = [a_0 a_1 \dots a_{N_t-1}]^T$, onde $a_k \in \mathcal{A}$, e o vetor de símbolos produzidos pelo quantizador $\mathbf{w} = [w_0 w_1 \dots w_{N_t-1}]^T$, onde $w_k \in \mathcal{Q} = \{q_0, q_1, \dots, q_{M-1}\}$ para $k = 0, 1, \dots, N_t - 1$. Defina $\mathcal{A}_m = \{a_k : v_k = m, k \in \{0, 1, \dots, N_t - 1\}\}$. Então, deseja-se minimizar

$$\begin{aligned} \phi &= \frac{1}{N_t} \sum_{k=0}^{N_t-1} (a_k - w_k)^2 \\ &= \frac{1}{N_t} \sum_{k=0}^{N_t-1} (a_k - q_{v_k})^2 \\ &= \phi(\mathbf{q}). \end{aligned} \tag{A.1}$$

Assim sendo,

$$\begin{aligned}\phi(\mathbf{q}) &= \frac{1}{N_t}\sum_{m=0}^{M-1}\sum_{a\in\mathcal{A}_m}(a-q_m)^2 \\ &= \frac{1}{N_t}\sum_{m=0}^{M-1}\left[\sum_{a\in\mathcal{A}_m}a^2 - 2\sum_{a\in\mathcal{A}_m}aq_m + |\mathcal{A}_m|\,q_m^2\right] \\ &= \frac{1}{N_t}\sum_{m=0}^{M-1}\left[C_m - 2B_mq_m + A_mq_m^2\right], \end{aligned} \tag{A.2}$$

onde $|\mathcal{A}_m|$ corresponde ao número de elementos em $\mathcal{A}_m$. E, portanto, tem-se que

$$\nabla\phi(\mathbf{q}) = 2\begin{bmatrix} A_0q_0 - B_0 \\ A_1q_1 - B_1 \\ \vdots \\ A_{M-1}q_{M-1} - B_{M-1} \end{bmatrix}. \tag{A.3}$$

Logo, dados $\mathbf{q}$ inicial, $\epsilon_\phi > 0$, $\epsilon_\mathbf{q} > 0$, $\epsilon_\mathbf{g} > 0$ e o número de iterações $N_{it}$, executam-se os seguintes passos para a otimização da TCQ ou TTCQ:

1. Simular TCQ (ou TTCQ), obtendo os coeficientes $A_m$, $B_m$ e $C_m$ de $\phi(\cdot)$;

2. $\mathbf{S} = \mathbf{I}_M$;
   $f_s = f_\mathbf{q} = \phi(\mathbf{q})$;
   $\mathbf{g_q} = \nabla\phi^(\mathbf{q})$;

3. $i = 0$;

4. $\mathbf{d} = \mathbf{S}\mathbf{g_q}$;
   Busca linear; // Encontrar $\alpha$ que minimize $\phi(\mathbf{q} + \alpha\mathbf{d})$.
   // A busca linear produz os seguintes valores.
   $\mathbf{p} = \mathbf{q} + \alpha\mathbf{d}$;
   $f_\mathbf{p} = \phi(\mathbf{p})$;

5. Se busca linear falhou parar;

6. Se ( $f_\mathbf{p} \geq f_s$ )
   {
   $\mathbf{S} = \mathbf{I}_M$;
   $\mathbf{d} = \mathbf{S}\mathbf{g_q}$;

Busca linear;

$\mathbf{p} = \mathbf{q} + \alpha\mathbf{d}$;

$f_{\mathbf{p}} = \phi(\mathbf{p})$;

Se busca linear falhou parar;

}

7. $f_s = f_{\mathbf{p}}$;
   $f_{\mathbf{q}} = f_{\mathbf{p}}$;
   $\mathbf{q}_s = \mathbf{q}$;
   $\mathbf{q} = \mathbf{p}$;
   $\Delta\mathbf{q} = \mathbf{q} - \mathbf{q}_s$;

8. Checar convergência usando as tolerâncias $\epsilon_\phi$ (função), $\epsilon_{\mathbf{q}}$ (vetor) e $\epsilon_{\mathbf{g}}$ (gradiente);

9. Se convergiu, parar;

10. Simular TCQ (ou TTCQ), obtendo os coeficientes $A_m$, $B_m$ e $C_m$ de $\phi(\cdot)$;

11. Ordenar $\mathbf{g}_{\mathbf{q}}$ e $\Delta\mathbf{q}$ de acordo com a ordenação de $\mathbf{q}$;

12. $\mathbf{q}_s = \mathbf{q}$;
    $\mathbf{g}_s = \mathbf{g}_{\mathbf{q}}$;
    $f_{\mathbf{q}} = \phi(\mathbf{q})$;
    $\mathbf{g}_{\mathbf{q}} = \nabla\phi(\mathbf{q})$;
    $\Delta\mathbf{g} = \mathbf{g}_s - \mathbf{g}_{\mathbf{q}}$;

13. // Fórmula de Broyden, Fletcher, Goldfarb e Shanno para calcular a matriz
    // de correção $\mathbf{C}$ de $\mathbf{S}$ [26].
    $\mathbf{C} = \left[1 + \frac{\Delta\mathbf{g}^T\mathbf{S}\Delta\mathbf{g}}{\Delta\mathbf{g}^T\Delta\mathbf{q}}\right]\frac{\Delta\mathbf{q}\Delta\mathbf{q}^T}{\Delta\mathbf{g}^T\Delta\mathbf{q}} - \left[\frac{\Delta\mathbf{q}\Delta\mathbf{g}^T\mathbf{S}+\mathbf{S}\Delta\mathbf{g}\Delta\mathbf{q}^T}{\Delta\mathbf{g}^T\Delta\mathbf{q}}\right]$;

14. $\mathbf{S} = \mathbf{S} + \mathbf{C}$;

15. Se ( $i == N_{it}$ ) parar;

16. $i = i + 1$;
    Ir para 4;

Este algoritmo pode ser simplificado quando a fonte possui uma distribuição simétrica, como no caso das fontes gaussiana, laplaciana e uniforme, assumindo também simetria para o vetor $\mathbf{q}$. Com isto pode-se reduzir à metade o número de variáveis [6].

## A.2 Otimização Parcialmente Conjunta para TCQ & TCM ou para TTCQ & TTCM

Sejam $R$ e $M = 2^{R+1}$ o número de *bits* por amostra e o número de níves de quantização da TCQ (ou TTCQ), respectivamente. Deseja-se minimizar o erro médio quadrático entre o vetor de símbolos da fonte $\mathbf{a} = [a_0 a_1 \dots a_{N_t-1}]^T$, onde $a_k \in \mathcal{A}$, e o vetor de símbolos produzidos pelo quantizador $\hat{\mathbf{w}} = [\hat{w}_0 \hat{w}_1 \dots \hat{w}_{N_t-1}]^T$, onde $\hat{w}_k \in \mathcal{Q} = \{q_0, q_1, \dots, q_{M-1}\}$ para $k = 0, 1, \dots, N_t - 1$. Defina $\hat{\mathcal{A}}_m = \{a_k : \hat{v}_k = m, k \in \{0, 1, \dots, N_t - 1\}\}$. Então, deseja-se minimizar

$$\begin{aligned} \phi &= \frac{1}{N_t} \sum_{k=0}^{N_t-1} (a_k - \hat{w}_k)^2 \\ &= \frac{1}{N_t} \sum_{k=0}^{N_t-1} (a_k - q_{\hat{v}_k})^2 \\ &= \phi(\mathbf{q}). \end{aligned} \tag{A.4}$$

Assim sendo,

$$\begin{aligned} \phi(\mathbf{q}) &= \frac{1}{N_t} \sum_{m=0}^{M-1} \sum_{a \in \hat{\mathcal{A}}_m} (a - q_m)^2 \\ &= \frac{1}{N_t} \sum_{m=0}^{M-1} \left[ \sum_{a \in \hat{\mathcal{A}}_m} a^2 - 2 \sum_{a \in \hat{\mathcal{A}}_m} a q_m + \left| \hat{\mathcal{A}}_m \right| q_m^2 \right] \\ &= \frac{1}{N_t} \sum_{m=0}^{M-1} \left[ \hat{C}_m - 2 \hat{B}_m q_m + \hat{A}_m q_m^2 \right], \end{aligned} \tag{A.5}$$

onde $\left| \hat{\mathcal{A}}_m \right|$ corresponde ao número de elementos em $\hat{\mathcal{A}}_m$. E, portanto, tem-se que

$$\nabla \phi(\mathbf{q}) = 2 \begin{bmatrix} \hat{A}_0 q_0 - \hat{B}_0 \\ \hat{A}_1 q_1 - \hat{B}_1 \\ \vdots \\ \hat{A}_{M-1} q_{M-1} - \hat{B}_{M-1} \end{bmatrix}. \tag{A.6}$$

Os passos de otimização são os mesmos do algoritmo anterior, bastando substituir os coeficientes $A_m^{(i)}$, $B_m^{(i)}$ e $C_m^{(i)}$ pelos coeficientes $\hat{A}_m^{(i)}$, $\hat{B}_m^{(i)}$ e $\hat{C}_m^{(i)}$ para $m = 0, 1, \ldots, M-1$.

## A.3 Otimização Conjunta para TCQ & TCM ou para TTCQ & TTCM

Sejam $R$ e $M = 2^{R+1}$ o número de *bits* por amostra e o número de níves de quantização da TCQ (ou TTCQ), respectivamente. Deseja-se minimizar o erro médio quadrático entre o vetor de símbolos da fonte $\mathbf{a} = [a_0 a_1 \ldots a_{N_t-1}]^T$, onde $a_k \in \mathcal{A}$, e o vetor de símbolos produzidos pelo quantizador $\mathbf{w} = [w_0 w_1 \ldots w_{N_t-1}]^T$, onde $w_k \in \mathcal{Q} = \{q_0, q_1, \ldots, q_{M-1}\}$ para $k = 0, 1, \ldots, N_t - 1$. Defina $\mathcal{A}_m = \{a_k : v_k = m, k \in \{0, 1, \ldots, N_t - 1\}\}$. Então, deseja-se minimizar

$$\begin{aligned} \phi &= \frac{1}{N_t} \sum_{k=0}^{N_t-1} d\left(a_k, w_k\right) \\ &= \frac{1}{N_t} \sum_{k=0}^{N_t-1} d\left(a_k, q_{v_k}\right) \\ &= \frac{1}{N_t} \sum_{m=0}^{M-1} \sum_{a \in \mathcal{A}_m} d\left(a, q_m\right). \end{aligned} \tag{A.7}$$

Definindo a função de distorção $d(a, q_m) = \Sigma_{n=0}^{M-1} p_{nm}(a - q_n)^2$ como WANG *et al.* [12], onde $p_{nm} = \Pr\{\hat{X} = r_n \text{ recebido} | X = r_m \text{ enviado}\}$, obtém-se

$$\phi = \frac{1}{N_t} \sum_{m=0}^{M-1} \sum_{a \in \mathcal{A}_m} \sum_{n=0}^{M-1} p_{nm} \left(a - q_n\right)^2 = \phi(\mathbf{q}, \mathbf{r}), \tag{A.8}$$

onde $\mathbf{q} = [q_0 q_1 \ldots q_{M-1}]^T$ e $\mathbf{r} = [r_0 r_1 \ldots r_{M-1}]^T$. Então, desenvolvendo-se a equação A.8,

$$\begin{aligned} \phi(\mathbf{q}, \mathbf{r}) &= \frac{1}{N_t} \sum_{m=0}^{M-1} \sum_{a \in \mathcal{A}_m} \sum_{n=0}^{M-1} p_{nm} \left[a^2 - 2aq_n + q_n^2\right] \\ &= \frac{1}{N_t} \sum_{m=0}^{M-1} \sum_{a \in \mathcal{A}_m} \left[a^2 - 2a \sum_{n=0}^{M-1} p_{nm} q_n + \sum_{n=0}^{M-1} p_{nm} q_n^2\right] \\ &= \frac{1}{N_t} \sum_{m=0}^{M-1} \left[\sum_{a \in \mathcal{A}_m} a^2 - 2 \sum_{a \in \mathcal{A}_m} a \sum_{n=0}^{M-1} p_{nm} q_n + |\mathcal{A}_m| \sum_{n=0}^{M-1} p_{nm} q_n^2\right] \\ &= \frac{1}{N_t} \sum_{m=0}^{M-1} \left[C_m - 2B_m \sum_{n=0}^{M-1} p_{nm} q_n + A_m \sum_{n=0}^{M-1} p_{nm} q_n^2\right], \end{aligned} \tag{A.9}$$

onde $|\mathcal{A}_m|$ corresponde ao número de elementos em $\mathcal{A}_m$. Mantendo-se $\mathbf{r}$ fixo, o vetor $\mathbf{q}$ que minimiza $\phi(\mathbf{q}, \mathbf{r})$ é dado por

$$q_n = \frac{\sum_{m=0}^{M-1} p_{nm} B_m}{\sum_{m=0}^{M-1} p_{nm} A_m}, n = 0, 1, \ldots, M-1. \tag{A.10}$$

Visto que as variáveis $r_n$ para $n = 0, 1, \ldots, M-1$ são ocultas, dados $\mathbf{q}$ inicial, $\mathbf{r}$ inicial, $\epsilon_\phi > 0$, $\epsilon_\mathbf{r} > 0$ e o número de iterações $N_{it}$, a minimização em relação a $\mathbf{r}$ faz uso da seguinte adaptação do algoritmo de Powell de busca direta que não usa derivadas [25]:

1. $\theta = 0,5$; $\delta_D = 1,0$;
   $\delta_I = \delta_D$;
   $f_a = f_b = f_c = 0,0$;
   $\mathbf{S} = \mathbf{I}_M$; // $\mathbf{S} = [\mathbf{s}_0 \mathbf{s}_1 \ldots \mathbf{s}_{M-1}]$.

2. $iter = 0$;
   $f_\mathbf{r} = \phi(\mathbf{q}, \mathbf{r})$[1];

3. $\mathbf{r}_a = \mathbf{r}$;
   $f_a = f_\mathbf{r}$;

4. $\Delta f_{pos} = 0$;
   $\Delta f_{max} = 0,0$;
   $restauraQ = 1$;

5. Para ( $k = 0$; $k < M-1$; $k++$ )
   {
      $\mathbf{d} = \mathbf{s}_k \delta_D$;
      Busca linear;
      $\mathbf{t} = \mathbf{r} + \alpha \mathbf{d}$;
      $f_\mathbf{t} = \phi(\mathbf{t})$;
      Se ( busca falhou ) ir para 6;
      $\Delta f = f_\mathbf{r} - f_\mathbf{t}$;

[1]Note que é necessário simular TCQ & TCM (ou TTCQ & TTCM), obtendo $p_{nm}$ e os coeficientes $A_m$, $B_m$ e $C_m$, toda vez que se for avaliar $\phi(\mathbf{q}, \mathbf{r})$.

Se ( $\Delta f > \Delta f_{max}$ )

{

$\Delta f_{pos} = k$;

$\Delta f_{max} = \Delta f$;

}

$f_{\mathbf{r}} = f_{\mathbf{t}}$;

$\mathbf{r} = \mathbf{t}$;

$\mathbf{r}_s = \mathbf{r}$;

Se ( $1 == restauraQ$ )

{

restauraQ $= 0$;

}

}

6. Se ( busca falhou )

{

Se ( $1 == restauraQ$ )

{

$\mathbf{q} = \mathbf{q}_s$;

}

$\mathbf{r} = \mathbf{r}_s$;

Parar;

}

7. $f_b = f_{\mathbf{r}}$;

$\mathbf{r}_b = \mathbf{r}$;

$\mathbf{r}_c = 2\mathbf{r}_b - \mathbf{r}_a$;

$f_c = \phi(\mathbf{q}, \mathbf{r}_c)$;

$\Delta f = f_c - f_a$;

8. Se ( $\Delta f < 0$ )

{

$\Delta f = (f_a - f_b - \Delta f_{max})/\Delta f$;

$\Delta f = (f_a - 2f_b + f_c)/\Delta f^2$;

Se ( $\Delta f < \Delta f_{max}$ )

{

Se ( $\Delta f_{pos} < M - 1$ )

{

Para ( $i = \Delta f_{pos}; i < M - 1; i++$ )

{

$\mathbf{s}_i = \mathbf{s}_{i+1}$;

}

}

$\mathbf{s}_{M-1} = \mathbf{r} - \mathbf{r}_a$;

$\Delta f = 1/|\mathbf{s}_{M-1}|$;

$\mathbf{s}_{M-1} = \Delta f \mathbf{s}_{M-1}$;

$\mathbf{d} = \mathbf{s}_{M-1}\delta_D$;

Busca linear;

$\mathbf{t} = \mathbf{r} + \alpha\mathbf{d}$;

$f_{\mathbf{t}} = \phi(\mathbf{t})$;

Se ( busca falhou )

{

$\mathbf{r} = \mathbf{r}_s$;

Parar;

}

$f_{\mathbf{r}} = f_{\mathbf{t}}$;

$\mathbf{r} = \mathbf{t}$;

$\mathbf{r}_s = \mathbf{r}$;

}

}

9. $iter = iter + 1$

10. Checar convergência comparando $f_a$, $f_{\mathbf{t}}$ e $\mathbf{r}_a$, $\mathbf{t}$ por meio das tolerâncias $\epsilon_\phi$ (função) e $\epsilon_{\mathbf{r}}$ (vetor);
    Se ( convergiu ) parar;

11. $\mathbf{q}_s = \mathbf{q}$;

Atualizar $\mathbf{q}$ usando a equação A.6;

12. $\delta_D = \theta\sqrt{|f_a - f_\mathbf{r}|}$;

13. Se ( $f_a < f_\mathbf{r}$ ) $\delta_D = -\delta_D$;
    Se ( $\delta_I < \delta_D$ ) $\delta_D = \delta_I$;

14. Se ( $iter == N_{it}$ ) parar;

15. Ir para 4;

Os valores de $\theta$ e $\delta_D$ devem ser ajustados para evitar que o algoritmo termine prematuramente. No presente trabalho, $\theta$ varia entre 0,4 e 0,9 com passo de 0,1 e $\delta_D$ varia entre 0,8 e 1,0 com passo de 0,01. Escolhe-se, então, a minimização que produz o melhor resultado. As faixas de valores, para $\theta$ e $\delta_D$, e os respectivos passos foram obtidos de forma experimental.

# Apêndice B

# Fórmulas

Este apêndice contém algumas fórmulas e teoremas simples que são utilizados nas simulações efetuadas neste trabalho.

## B.1 Logaritmo da Soma de Exponenciais

O logaritmo da soma de exponenciais é utilizado na implementação do algoritmo BCJR-LogMAP. Os teoremas a seguir são úteis no cálculo destas expressões pois evitam problemas numéricos que ocorrem com freqüência na avaliação da soma de exponenciais.

**Teorema B.1 (Duas Exponenciais).** *Sejam $a, b \in \mathbb{R}$, onde $\mathbb{R}$ é o conjunto dos números reais. Então,*

$$\ln\left(e^a + e^b\right) = \max(a,b) + \ln\left(1 + e^{-|a-b|}\right), \tag{B.1}$$

*onde* $\max(a,b)$ *retorna o maior entre os números* $a$ *e* $b$.

*Demonstração.* O caso $a = b$ é trivial. Se $a > b$ então $b - a = -|b-a| = -|a-b|$ e

$$\begin{aligned} \ln\left(e^a + e^b\right) &= \ln\left[e^a\left(1 + e^{b-a}\right)\right] \\ &= a + \ln\left(1 + e^{-|a-b|}\right). \end{aligned} \tag{B.2}$$

Se $a < b$ então $a - b = -|a-b|$ e

$$\begin{aligned} \ln\left(e^a + e^b\right) &= \ln\left[e^b\left(1 + e^{a-b}\right)\right] \\ &= b + \ln\left(1 + e^{-|a-b|}\right), \end{aligned} \tag{B.3}$$

e o teorema está provado. □

**Teorema B.2 (Três ou Mais Exponenciais).** *Sejam $b_0, b_1, \ldots, b_n$ números reais, onde $n > 1$. Calcule*

$$\begin{aligned} a_0 &= \ln\left(e^{b_0} + e^{b_1}\right) \\ &= \max(b_0, b_1) + \ln\left(1 + e^{-|b_0 - b_1|}\right). \end{aligned} \tag{B.4}$$

*Calcule a seguinte seqüência de valores*

$$\begin{aligned} a_i &= \ln\left(e^{a_{i-1}} + e^{b_{i+1}}\right) \\ &= \max(a_{i-1}, b_{i+1}) + \ln\left(1 + e^{-|a_{i-1} - b_{i+1}|}\right), \end{aligned} \tag{B.5}$$

*para $i = 1, 2, \ldots, n-1$. Então,*

$$a_{n-1} = \ln\left[\sum_{i=0}^{n} e^{b_i}\right]. \tag{B.6}$$

*Demonstração.* Prova por indução. Tem-se que

$$\begin{aligned} a_1 &= \ln\left(e^{a_0} + e^{b_2}\right) \\ &= \ln\left(e^{b_0} + e^{b_1} + e^{b_2}\right) \\ &= \max(a_0, b_2) + \ln\left(1 + e^{-|a_0 - b_2|}\right) \end{aligned} \tag{B.7a}$$

$$\begin{aligned} a_2 &= \ln\left(e^{a_1} + e^{b_3}\right) \\ &= \ln\left(e^{b_0} + e^{b_1} + e^{b_2} + e^{b_3}\right) \\ &= \max(a_1, b_3) + \ln\left(1 + e^{-|a_1 - b_3|}\right) \\ &\vdots \\ a_i &= \ln\left(e^{a_{i-1}} + e^{b_{i+1}}\right) \\ &= \ln\left[\sum_{j=0}^{i-1} e^{b_j} + e^{b_{i+1}}\right] \\ &= \max(a_{i-1}, b_{i+1}) + \ln\left(1 + e^{-|a_{i-1} - b_{i+1}|}\right) \\ &\vdots \\ a_{n-1} &= \ln\left(e^{a_{n-2}} + e^{b_n}\right) \\ &= \ln\left[\sum_{j=0}^{n-1} e^{b_j} + e^{b_n}\right] \\ &= \max(a_{n-2}, b_n) + \ln\left(1 + e^{-|a_{n-2} - b_n|}\right) \\ &= \ln\left[\sum_{i=0}^{n} e^{b_i}\right] \end{aligned} \tag{B.7b, B.7c, B.7d}$$

e o teorema está provado. □

## B.2 Intervalo de Confiança

**Definição B.1.** Uma *estimativa de intervalo* de um parâmetro $x$ é um intervalo $(a, b)$, onde $a = g_a(\mathbf{x})$ e $b = g_b(\mathbf{x})$, sendo $\mathbf{x} = [x_0 x_1 \dots x_{N_s-1}]^T$ o vetor de observação com $N_s$ amostras. O intervalo aleatório correspondente $(A, B)$ é o *intervalo estimador* de $x$. Diz-se que $(a, b)$ é um *intervalo de confiança* de $\gamma$ de $x$ se

$$\Pr\{A < x < B\} = \gamma. \tag{B.8}$$

A constante $\gamma$ é chamada de *coeficiente de confiança* da estimativa e $\delta = 1 - \gamma$ é o *nível de confiança*. Desta forma, $\gamma$ é uma medida da confiança de que o parâmetro $x$ pertence ao intervalo $(a, b)$. Ou seja, em $100\gamma$ por cento dos casos isto ocorre [27].

### Estimativa da Média

Deseja-se estimar a média $\mu_X$ de uma variável aleatória $X$, usando como estimativa o valor

$$m_X = \frac{1}{N_s} \sum_{i=0}^{N_s-1} x_i. \tag{B.9}$$

Para encontrar uma estimativa de intervalo seria preciso determinar a distribuição da variável aleatória $M_X$ o que, geralmente, é um problema difícil [27]. Para simplificar o problema é comum assumir que $M_X$ é normal. Se $X$ for normal, isto é verdade. Se $N_s$ for suficientemente grande, isto é aproximadamente verdade.

Se a variância $\sigma_X^2$ de $X$ for conhecida e assumindo uma distribuição normal para $X$, então o estimador $M_X$ de $\mu_X$ será a distribuição $\mathcal{N}(\mu_X, \sigma_X/\sqrt{N_s})$. Sendo $z_u$ o percentil[1] $u$ da densidade normal padrão $\mathcal{N}(0, 1)$, tem-se que

$$\Pr\left\{\mu_X - z_{1-\delta/2}\frac{\sigma_X}{\sqrt{N_s}} < M_X < \mu_X + z_{1-\delta/2}\frac{\sigma_X}{\sqrt{N_s}}\right\} = G(z_{1-\delta/2}) - G(-z_{1-\delta/2}). \tag{B.10}$$

Visto que $z_u = -z_{1-u}$ e que $G(-z_{1-u}) = G(z_u) = u$, então

$$\Pr\left\{M_X - z_{1-\delta/2}\frac{\sigma_X}{\sqrt{N_s}} < \mu_X < M_X + z_{1-\delta/2}\frac{\sigma_X}{\sqrt{N_s}}\right\} = 1 - \delta = \gamma. \tag{B.11}$$

[1] Ou seja, $u = \Pr\{Z \leq z_u\} = G(z_u)$, onde $G(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-z^2/2} dz$.

Ou seja, pode-se afirmar, com um coeficiente de confiança $\gamma$, que $\mu_X$ está no intervalo $m_X \pm z_{1-\delta/2}\sigma_X/\sqrt{N_s}$. Por conseguinte, para determinar um intervalo de confiança para $\mu_X$, seguem-se os seguintes passos:

1. selecionar o número $\gamma = 1 - \delta$;
2. encontrar $z_u$ para $u = 1 - \delta/2$
3. formar o intervalo $m_X \pm z_{1-\delta/2}\sigma_X/\sqrt{N_s}$.

Caso a distribuição de $M_X$ não seja conhecida sempre é possível se recorrer à expressão [27]

$$\Pr\left\{M_X - \frac{\sigma_X}{\sqrt{N_s\delta}} < \mu_X < M_X + \frac{\sigma_X}{\sqrt{N_s\delta}}\right\} > 1 - \delta = \gamma. \tag{B.12}$$

Se a variância de $X$ não é conhecida os métodos anteriores não podem ser utilizados. Neste caso, para estimar $\mu_X$, é preciso calcular a variância da amostra por meio de

$$s_X^2 = \frac{1}{N_s - 1}\sum_{i=0}^{N_s-1}\left(x_i - m_X\right)^2, \tag{B.13}$$

que é uma estimativa não polarizada (não tendenciosa) que tende para $\sigma_X^2$ quando $N_s \to \infty$ [27]. Portanto, se $N_s$ for suficientemente grande pode-se assumir $s_X^2 = \sigma_X^2$ e utilizar o intervalo de confiança aproximado

$$m_X - z_{1-\delta/2}\frac{s_X}{\sqrt{N_s}} < \mu_X < m_X + z_{1-\delta/2}\frac{s_X}{\sqrt{N_s}}. \tag{B.14}$$

Assumindo que $X$ seja normal, é possivel encontrar um intervalo de confiança exato, pois neste caso a razão

$$\frac{M_X - \mu_X}{S_X/\sqrt{N_s}} \tag{B.15}$$

possui uma distribuição Student-t com $N_s - 1$ graus de liberdade [27]. Sendo $t_u$ seu percentil $u$, tem-se que

$$\Pr\left\{-t_u < \frac{M_X - \mu_X}{S_X/\sqrt{N_s}} < t_u\right\} = 2u - 1 = \gamma, \tag{B.16}$$

levando ao intervalo

$$m_X - t_{1-\delta/2}\frac{s_X}{\sqrt{N_s}} < \mu_X < m_X + t_{1-\delta/2}\frac{s_X}{\sqrt{N_s}}. \tag{B.17}$$

Se $N_s < 30$ é necessário consultar uma tabela para obter $t_u(N_s)$. Se $N_s \geq 30$, $t_u(N_s) \simeq z_u\sqrt{N_s/(N_s-2)}$.

## Intervalo de Confiança para o Valor Médio do Erro Médio Quadrático

Neste trabalho, deseja-se calcular o intervalo de confiança com $\gamma = 0,95$ para o valor médio da distorção entre os $N_s$ vetores observados $\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_{N_s-1}$ e os $N_s$ vetores $\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_{N_s-1}$, todos pertencentes ao $\mathbb{R}^N$, de modo que

$$x_i = D_i = \frac{1}{N} \sum_{k=0}^{N-1} \left[a_{i,k} - w_{i,k}\right]^2 . \tag{B.18}$$

Como $N$ é um valor grande pode-se assumir que a variável aleatória $X$ possui uma distribuição normal (Teorema do Limite Central [27]). Sendo $N_s \geq 100$, $t_u(N_s) \simeq z_u \sqrt{N_s/(N_s-2)}$ e, portanto, substituindo esta expressão na equação B.17, obtém-se

$$m_X - z_{1-\delta/2} \frac{s_X}{\sqrt{N_s - 2}} < \mu_X < m_X + z_{1-\delta/2} \frac{s_X}{\sqrt{N_s - 2}}. \tag{B.19}$$

Então, $\gamma = 0,95$ implica $u = 1 - \delta/2 = 0,975$ e $z_{0,975} = 1,967$. Logo,

$$m_X - 1,967 \frac{s_X}{\sqrt{N_s - 2}} < \mu_X < m_X + 1,967 \frac{s_X}{\sqrt{N_s - 2}}. \tag{B.20}$$