5
Simulações

Neste capítulo é realizada uma breve descrição do simulador turbo implementado. Alguns fatores que influenciam o desempenho dos códigos turbo são analisados, e esta influência é ilustrada através resultados obtidos por simulações. Os parâmetros dos códigos turbo e entrelaçadores utilizados na simulações foram obtidos de [20].

Este capítulo está organizado da seguinte forma: na Seção 5.1, é apresentado o simulador turbo e suas funcionalidades. Na Seção 5.2, é discutido um pouco sobre o excelente desempenho desses códigos. Na seção 5.3, é ilustrado o desempenho dos algoritmos de decodificação, discutidos ao longo deste trabalho. Na Seção 5.4, são discutidos critérios de parada. Nas seções 5.5 e 5.6, são analisados como tipos de entrelaçadores e técnicas de puncionamento influenciam no desempenho dos códigos turbo. Por fim, na Seção 5.8, é mostrado e discutido o desempenho de um codificador turbo quando este é associado a um modulador ASK-4.

5.1
Simulador Turbo

Para implementação do simulador turbo desenvolvido, escolhemos a linguagem de programação[1] C++, devido a dois motivos: a rapidez da linguagem e o fato da linguagem ser orientada a objeto.

Uma linguagem orientada objeto é indicada para aplicações cuja estrutura é composta por blocos(objetos) que comunicam-se entre si — exatamente como ocorre em um sistema de comunicação, ou em menor escala na codificação e decodificação turbo — permitindo uma implementação mais eficiente para estes casos.

Este tipo de linguagem fundamenta-se em dois elementos: classes e objetos. As classes são compostas por membros que podem ser variáveis

[1]Mais informações sobre a linguagem C++ podem ser encontradas em www.cplusplus.com/doc/tutorial/.

e/ou funções, enquanto que os objetos são vistos como instâncias das classes. Abaixo são discriminadas as classes implementadas no simulador turbo desenvolvido:

`Classe RSConv:` realiza a codificação convolucional recursiva.

`Classe Inter:` realiza as funções de entrelaçamento e desentrelaçamento. Dispõe de três tipos de entrelaçadores: entrelaçador de bloco, entrelaçador pseudo-aleatório e entrelaçador S-aleatório.

`Classe Turbo Enc:` realiza a codificação turbo e utiliza membros das classes `RSConv` e `Inter`.

`Classe Punc:` realiza o puncionamento das seqüências de saída da classe `Turbo Enc`, permitindo obter codificadores turbo de diferentes taxas.

`Classe Modem BPSK:` realiza a modulação BPSK, e calcula as probabilidades condicionais do canal, fornecidas para classe `BCJR`.

`Classe Modem ASK-4:` realiza a modulação ASK-4, e calcula as probabilidades condicionais do canal, fornecidas para classe `BCJR`.

`Classe AWGN:` simula um canal AWGN.

`Classe BCJR:` realiza a decodificação convolucional, permitindo a utilização de três algoritmos: MAP(BCJR), Log-MAP e Max-Log-MAP.

`Classe Turbo Dec:` realiza a decodificação turbo iterativa e utiliza membros das classes `BCJR` e `Inter`.

Na Fig. 5.1, está ilustrado o diagrama de blocos do simulador turbo em função das classes implementadas.

5.2 Desempenho de Códigos Turbo

Os códigos turbo, como foi abordado anteriormente, apresentam um desempenho que se encontra a poucos dB's do limite teórico proposto por Shannon[2]. Por exemplo, o código turbo $\mathcal{C} = (21, 37, 4096)$ cujo desempenho

[2]Neste trabalho, comparamos códigos com o limite de Shannon para um canal AWGN cuja entrada do canal assume valores reais quaisquer, no entanto, vale a pena ressaltar que uma comparação mais justa seria obtida, comparando-se esses códigos ao limite de Shannon para um canal gaussiano entrada binária.

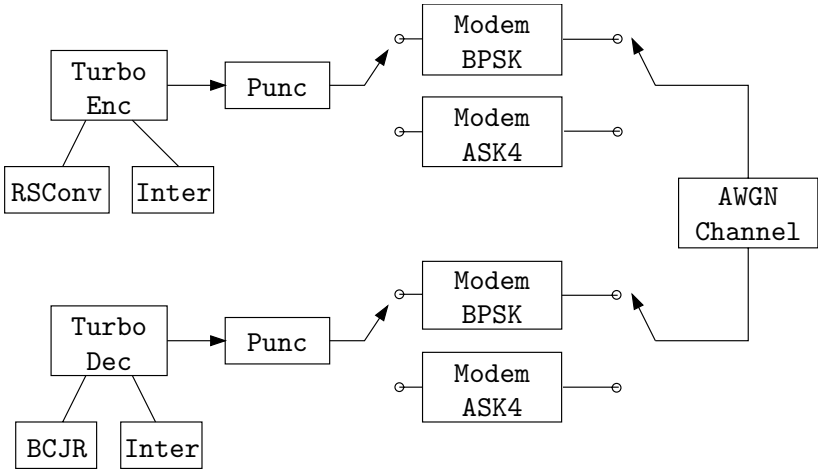

Figura 5.1: Diagrama de blocos para o simulador turbo.

está ilustrado na Fig. 5.2, encontra-se a cerca de 1.05 dB do limite(-0.55 dB) de Shannon para taxa $R = 1/3$ e probabilidade de erro $P(e_b) = 10^{-5}$.

Os parâmetros utilizados na simulação de desempenho deste código foram: $R = 1/3$, algoritmo de decodificação MAP, entrelaçador pseudo-aleatório, canal AWGN com modulação BPSK.

Na Fig. 5.2, pode-se perceber inicialmente que para altas razões de $E_b/\mathcal{N}_0$, aumentar o número de iterações implica em uma melhoria considerável no desempenho do código, entretanto a partir de um certo número de iterações, quando a decodificação se aproxima do desempenho assintótico estabelecido para o código, realizar novas iterações resultará em pouca melhoria no desempenho. No caso do código turbo $\mathcal{C} = (21, 37, 4096)$, isto ocorre para iteração $l = 18$ [20].

Este comportamento típico, presente nas curvas de desempenho de códigos turbo, é justificado pelo fato de que a informação trocada entre os decodificadores SISO do decodificador turbo torna-se mais correlatada a cada nova iteração.

Outro parâmetro que influencia de maneira crítica o desempenho dos códigos turbo, particularmente para baixas razões de $E_b/\mathcal{N}_0$, é o tamanho do entrelaçador utilizado, conforme ilustrado na Fig. 5.3. Como o entrelaçador é o elemento responsável por descorrelatar as entradas dos decodificadores SISO que compõem o decodificador turbo, aumentar seu tamanho implica em torná-las mais descorrelatadas, em relação ao ruído à entrada do decodificador turbo, aumentando o ganho de informação dos decodificadores no processo de decodificação iterativo e melhorando assim o desempenho do código.

Na Fig. 5.3, pode ser observada a influência do comprimento N do entrelaçador no desempenho do código turbo $\mathcal{C} = (21, 37, N)$. Os

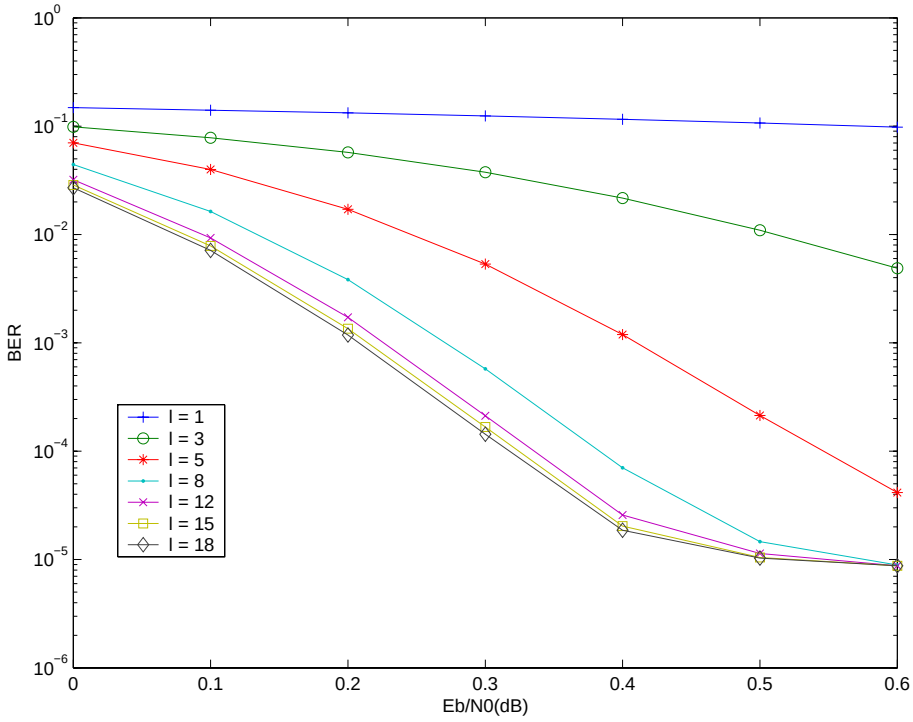

Figura 5.2: Influência do número de iterações no desempenho do código turbo $\mathcal{C} = (21, 37, 4096)$. $R = 1/3$, MAP, BPSK, AWGN.

parâmetros utilizados na simulação do desempenho deste código foram: $R = 1/3$, algoritmo de decodificação MAP, entrelaçador pseudo-aleatório, canal AWGN com modulação BPSK. Foi escolhida a iteração $l = 8$ para $N = 420$, 1024, e 2048, pois para estes comprimentos não há melhoria significativa de desempenho para um número superior de iterações, enquanto que para os comprimentos $N = 4098$ e 8182, a iteração escolhida foi $l = 18$, sob a mesma justificativa [20].

Análise de Patamar de Erro

Uma característica importante da curva de desempenho de códigos turbo, conforme ilustrado na Fig. 5.4, é que esta apresenta duas regiões distintas: uma na qual para uma pequena variação na razão $E_b/\mathcal{N}_0$, há uma grande variação na taxa de erro de bit(BER), denominada de *região de queda*(*cliff*); e outra em que ocorre a situação oposta, ou seja, para uma grande variação na razão $E_b/\mathcal{N}_0$, há uma pequena variação na BER, região denominada de *patamar* (*floor*). Ao valor da probabilidade de erro que caracteriza a região de patamar, denomina-se *nível de patamar de erro*(*error floor*).

Compreender os códigos turbo e seu excelente desempenho, consiste

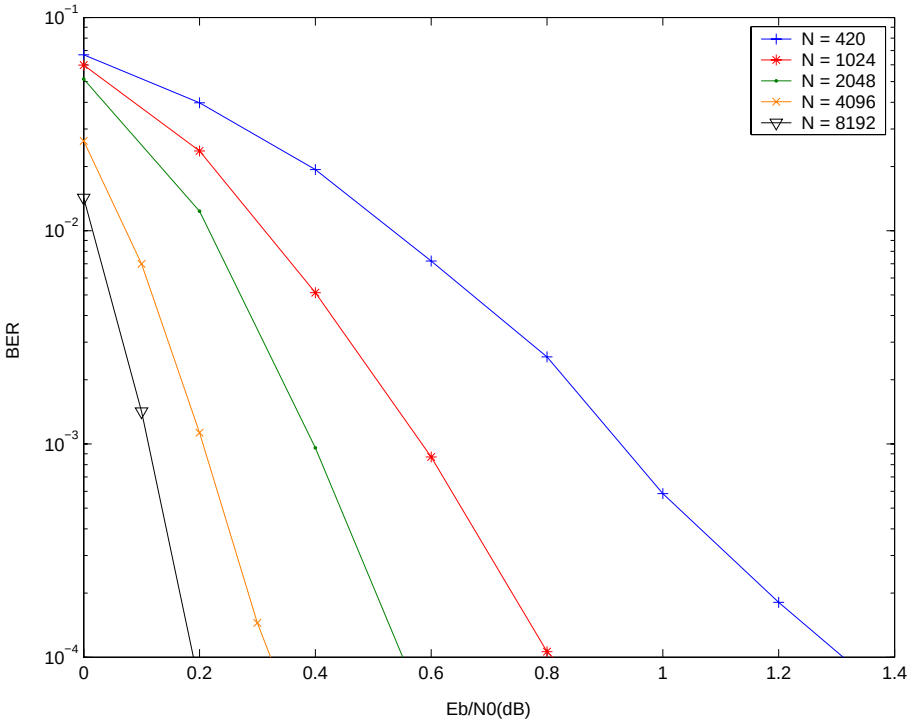

Figura 5.3: Influência de N no desempenho do código turbo $\mathcal{C} = (21, 37, N)$. $R = 1/3$, $l = 8$ para $N = 420, 1024, 2048$, $l = 18$ para $N = 4096, 8192$, MAP, BPSK, AWGN.

em compreender quais fatores influenciam e determinam as duas regiões, principalmente a região de patamar.

Foi proposto em [9], que o desempenho assintótico para códigos turbo, para moderadas e altas razões sinal-ruído, ou seja, para região onde o *error floor* ocorre, é dado por

$$P(e_b) \approx \frac{\eta_{\text{free}} \tilde{w}_{\text{free}}}{N} \, Q\left(\sqrt{d_{\text{free}} \frac{2RE_b}{\mathcal{N}_0}}\right) \tag{5-1}$$

sendo $P(e_b)$, a probabilidade de erro de bit, η_{free}, a multiplicidade das palavras códigos com a distância livre d_{free}, e $\tilde{w}_{\text{free}}$, o peso médio das mensagens de entrada que provocam a ocorrência destas palavras. A razão η_{free}/N é denominada de *multiplicidade efetiva*.

Note que a inclinação da assíntota da Eq. (5-1) é determinada por d_{free}(quanto menor d_{free}, menor a inclinação) e deste modo conclui-se que a presença do patamar de erro nas curvas de desempenho dos códigos turbo está relacionada ao fato destes códigos possuírem distâncias livres relativamente baixas.

Ainda em relação à Eq. (5-1), pode-se dizer que a multipicidade efetiva η_{free}/N é responsável por determinar a posição do ponto de intersecção entre

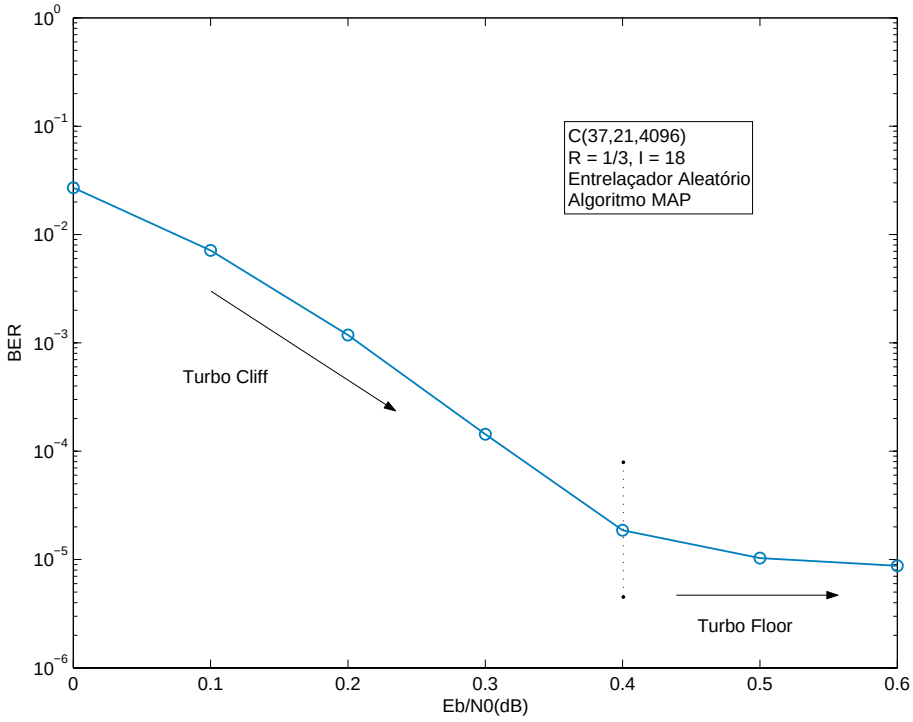

Figura 5.4: Curva de desempenho típica para um código turbo.

a região de *cliff* e a região de *floor*. Neste caso, diminuir a multiplicidade efetiva implica em posicionar este ponto para baixas probabilidades de erro, fazendo com que o patamar de erro surja para valores mais baixos de BER(altas razões $E_b/\mathcal{N}_0$).

Para um algoritmo de decodificação fixo, tem-se então que otimizar o desempenho de um código turbo na região do *floor*, é otimizar o par $(\eta_{\text{free}}/N, d_{\text{free}})$, diminuindo η_{free}/N e aumentando d_{free}, onde η_{free} determina a posição de início e d_{free} a inclinação do patamar de erro, assíntota da Eq. (5-1). Isto é obtido através de um bom projeto para os códigos componentes e do entrelaçador utilizados.

5.3 Algoritmos de Decodificação

Como foi abordado nas seções 4.3 e 4.3.1, o algoritmo MAP(BCJR) proposto em [4] para decodificação de códigos turbo consiste em uma técnica subótima com excelente desempenho, no entanto de alta complexidade. Foram propostas outras versões do algoritmo MAP no domínio logarítmico de menor complexidade, os algoritmos Log-MAP e Max-Log-MAP. Entre estes, o algoritmo Max-Log-MAP possui menor complexidade e pior desempenho, enquanto que o Log-MAP possui uma complexidade intermediária

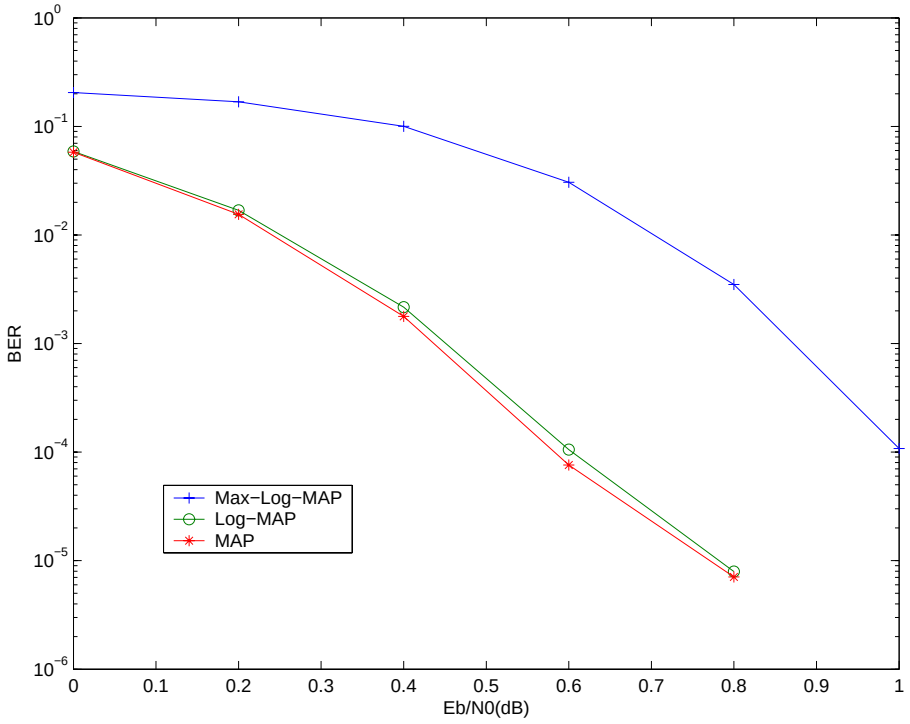

Figura 5.5: Desempenho do código turbo $\mathcal{C} = (17, 15, 2048)$ para os algoritmos MAP, Log-MAP e Max-Log-MAP. $R = 1/3$, $l = 8$, $S = 20$, BPSK, AWGN.

em relação aos outros dois, e o mesmo desempenho que o algoritmo MAP, como pode ser observado na Fig. 5.5.

Na simulação de desempenho código turbo $\mathcal{C} = (15, 17, 2048)$ para cada algoritmo de decodificação, foram utilizados os seguintes parâmetros: $R = 1/3$, $l = 8$, entrelaçador S-aleatório com $S = 20$, canal AWGN com modulação BPSK. Na Fig. 5.5 está ilustrada apenas o resultado da oitava iteração para cada algoritmo.

A Tabela 5.1 discrimina o número de operações realizadas pelos algoritmos, por seção da treliça, por iteração, para um codificador convolucional (n, k, K).

Determinar quanto um algoritmo é mais complexo que outro, a partir da Tabela 5.1, depende do custo que cada operação possui em determinada implementação. Em *software*, não pudemos observar diferença significativa na velocidade de execução de cada algoritmo, o que nos motivou a optar pelo uso do algoritmo MAP em nossas simulações. Acreditamos que na implementação em *hardware*, a diferença entre a velocidade dos algoritmos seja bem mais significativa.

Outra limitação que encontramos na implementação dos algoritmos em *software*, foi a rápida perda de precisão numérica que ocorria nos cálculos das mensagens, ou métricas, α e β, para o algoritmo MAP. Os cálculos

Tabela 5.1: Tabela com as complexidades dos algoritmos MAP, Log-MAP e Max-Log-MAP para um codificador convolucional (n, k, K).

Operações	MAP	Log-MAP	Max-Log-MAP
Adição	$2 \cdot 2^k \cdot 2^{K-1} + 6$	$6 \cdot 2^k \cdot 2^{K-1} + 6$	$4 \cdot 2^k \cdot 2^{K-1} + 8$
Multiplicação	$5 \cdot 2^k \cdot 2^{K-1} + 8$	$2^k \cdot 2^{K-1}$	$2 \cdot 2^k \cdot 2^{K-1}$
Operador $\max(\cdot)$		$4 \cdot 2^{K-1} - 2$	$4 \cdot 2^{K-1} - 2$
Consultas		$4 \cdot 2^{K-1} - 2$	
Exponenciação	$2 \cdot 2^k \cdot 2^{K-1}$		

dessas quantidades envolvem a soma e produtos de números pequenos, e a solução para estabilizar numericamente esse algoritmo é obtida utilizando as versões normalizadas α', β' de α e β, de modo que [31]

$$\sum_{\sigma_t} \alpha'(\sigma_t) = 1 \qquad (5\text{-}2)$$
$$\sum_{\sigma_{t-1}} \beta'(\sigma_{t-1}) = 1$$

onde

$$\alpha'(\sigma_t) = A_t \cdot \sum_{e \in E_t(\sigma_t)} \alpha'(e)\gamma(e) \qquad (5\text{-}3)$$
$$\beta'(\sigma_{t-1}) = B_t \cdot \sum_{e \in E_t(\sigma_{t-1})} \beta'(e)\gamma(e)$$

com

$$A_t = \frac{1}{\sum_{\sigma_t} \alpha'(\sigma_t)} \qquad (5\text{-}4)$$
$$B_t = \frac{1}{\sum_{\sigma_{t-1}} \beta'(\sigma_{t-1})}.$$

5.4 Critérios de Parada

A decodificação turbo, como dito anteriormente, é realizada iterativamente, e em [4] o critério de parada consiste em finalizar o algoritmo após um número fixo de iterações. Entretanto, esse critério de parada mostra-se ineficiente, quando a partir de uma iteração l, novas iterações não produzem melhoria no desempenho.

Outros critérios de parada capazes que identifique quando iterações

adicionais provocam pouca ou nenhuma melhoria podem ser vantajosos. A decodificação turbo é de alta complexidade, utilizar um critério de parada eficiente, evita iterações desnecessárias e otimiza o processo sob o ponto de vista computacional.

Com este fim, em [23], foi mostrado que há cinco padrões típicos de blocos na saída do decodificador turbo.

Padrão 1**:** Blocos sem erros com convergência rápida. Neste caso, o decodificador estima a mensagem corretamente em poucas iterações.

Padrão 2**:** Blocos sem erros com convergência lenta. Para tais blocos, o decodificador estima corretamente a mensagem transmitida, mas utiliza um número excessivo de iterações para tal.

Padrão 3**:** Blocos com poucos erros O decodificador estima uma mensagem incorreta, mas que está próxima da mensagem transmitida.

Padrão 4**:** Blocos com erros oscilantes. O decodificador não se decide, apresentando uma estimativa que oscila ao longo das iterações.

Padrão 5**:** Blocos com muitos erros. A decodificação inicia e finaliza com muitos erros, ao longo das iterações.

A Fig. 5.6 ilustra o número de bits errados para a iteração $l = 18$ em um bloco representativo de cada padrão. Os parâmetros utilizados nessa simulação foram: código turbo $\mathcal{C} = (21, 37, 256)$, $R = 1/3$, algoritmo de decodificação MAP, entrelaçador aleatório, canal AWGN com modulação BPSK, $E_b/\mathcal{N}_0 = 0.5$ dB, 100 blocos transmitidos.

A partir da existência desses cinco padrões, decidiu-se realizar novamente a simulação descrita acima, transmitindo 50000 blocos, e discriminar o número de ocorrência de cada padrão através de critérios específicos para essa realização, os resultados obtidos estão ilustrados na Fig. 5.7.

Note que o Padrão 1 — bloco sem erros com convergência rápida — é o mais freqüente, motivando a utilização de um critério de parada.

Vários critérios propostos na literatura [24], são listados a seguir. O critério correspondente ao caso na qual a decodificação é finalizada após um número fixo de iterações pré-definido é denominado, neste trabalho, de *NIF*(Número Fixo de Iterações).

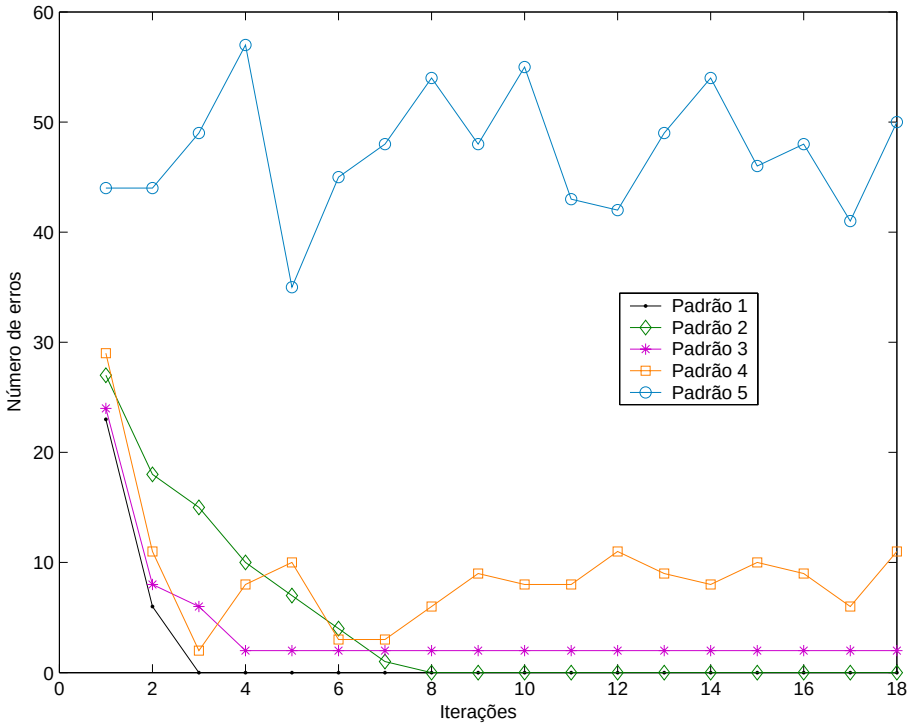

Figura 5.6: Padrões típicos de blocos na decodificação turbo. Código turbo $(21, 37, 256)$, $R = 1/3$, $l = 18$, MAP, BPSK, AWGN.

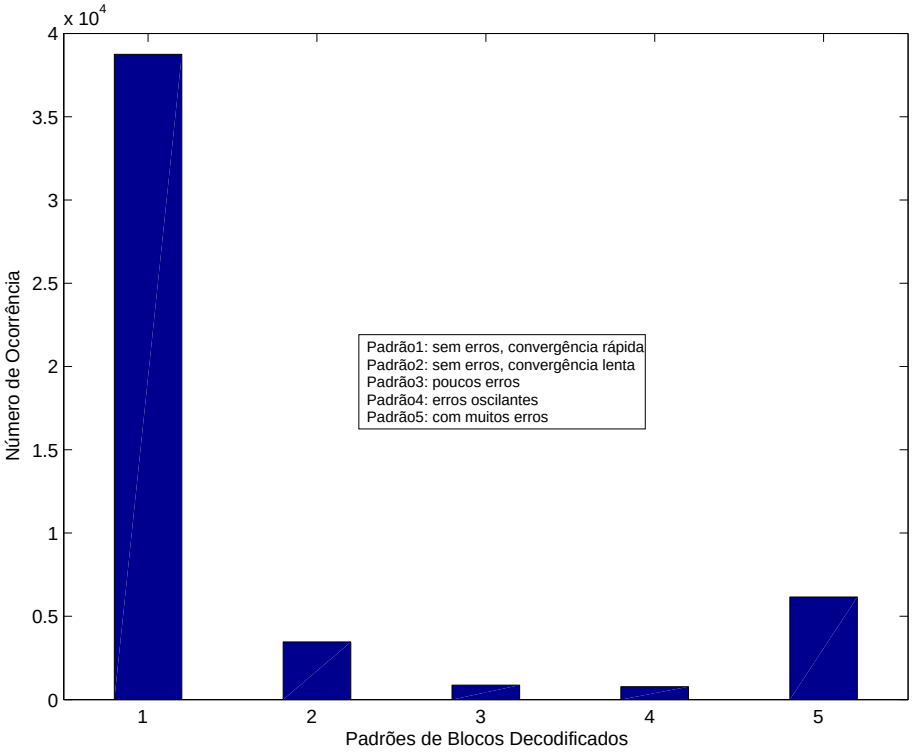

Figura 5.7: Freqüência de ocorrência dos padrões de blocos decodificados. Código turbo $(21, 37, 256)$, $R = 1/3$, $l = 18$, MAP, BPSK, AWGN.

1) **CRC(*Cyclic Redundancy Check*) [16]:** Esse critério consiste em um esquema onde um bloco $\boldsymbol{c}_p$ com $\mathcal{N}_c$ bits, provenientes de um codificador CRC detetor de erros, são anexados ao final de cada bloco mensagem antes de serem processados pelo codificador turbo. O decodificador turbo ao final de cada iteração entrega ao decodificador do CRC uma estimativa do bloco mensagem e o vetor de paridade $\boldsymbol{c}_p$ é utilizado para checar se há erros. As iterações são encerradas assim que o decodificador CRC não detete mais erro.
2) **CE(*Cross Entropy*) [10]:** Depois de cada iteração l, o critério CE calcula uma aproximação $T(l)$ da entropia cruzada entre os logaritmos das razões verossimilhança(LLR) dos decodificadores de cada componente de codificação. A decodificação é interrompida quando $T(l) < (10^{-2} \sim 10^{-4})T(1)$.
3) **SCR(*Sign Change Ratio*) [17]:** Neste critério calcula-se $C(l)$, o número de mudança de sinais entre as LLR das informações extrínsecas das iterações $l-1$ e l, e a decodificação é terminada quando $C(l) \leq qN$, onde q é uma constante usualmente escolhida entre $0.005 \leq q \leq 0.03$, e N é o tamanho do entrelaçador empregado.
4) **HDA(*Hard Decision Aid*) [17]:** Neste critério o algoritmo pára se as estimativas da mensagem(*hard decision*) transmitida na iteração $l-1$ e l são iguais.

No simulador turbo, implementamos o critério HDA, pois possui baixa complexidade, proporciona uma boa economia no tempo das simulações e uma perda de desempenho desprezível.

As Fig. 5.8 e Fig. 5.9 onde estão ilustradas, respectivamente, o número médio de iterações por bloco para cada $E_b/\mathcal{N}_0$ com o uso do HDA, permite verificar que a perda de desempenho, quando se compara o critério HDA com o critério NIF.

Os parâmetros das simulações foram: código turbo $\mathcal{C} = (21, 37, 4096)$, $R = 1/3$, algoritmo de decodificação MAP, entrelaçador S-aleatório com $S = 29$, canal AWGN com modulação BPSK. Sendo $l = 18$ para NIF1 e $l = 6$ para NIF2.

Note que em relação ao critério NIF1, o critério HDA apresentou um nível de complexidade inferior (convergência mais rápida) e um desempenho equivalente. Em relação ao critério NIF2, para a região em torno da razão $E_b/\mathcal{N}_0 = 0.4$ dB, ou seja $P(e_b) < 10^{-4}$, o critério HDA apresentou uma convergência média equivalente e um desempenho superior, conforme pode ser observado nas Fig. 5.8 e Fig. 5.9.

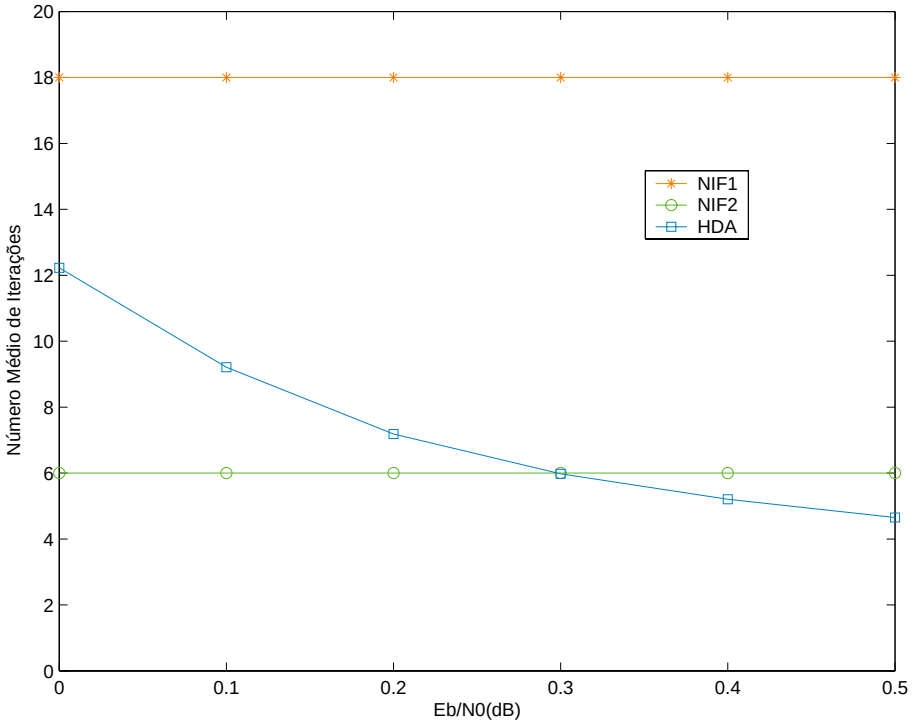

Figura 5.8: Número médio de iterações por bloco para as simulações para o código turbo $\mathcal{C} = (21, 37, 4096)$. $R = 1/3$, MAP, BPSK, AWGN.

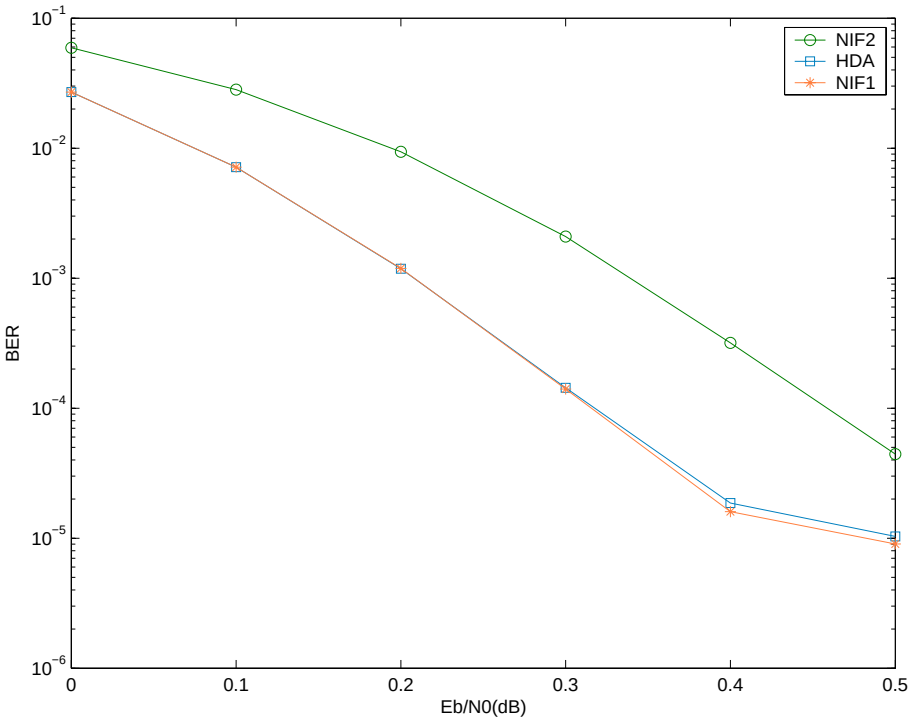

Figura 5.9: Curva de desempenho para o código turbo $\mathcal{C} = (21, 37, 4096)$, $l = 18$ para NIF1 e $l = 6$ para NIF2. $R = 1/3$, MAP, BPSK, AWGN.

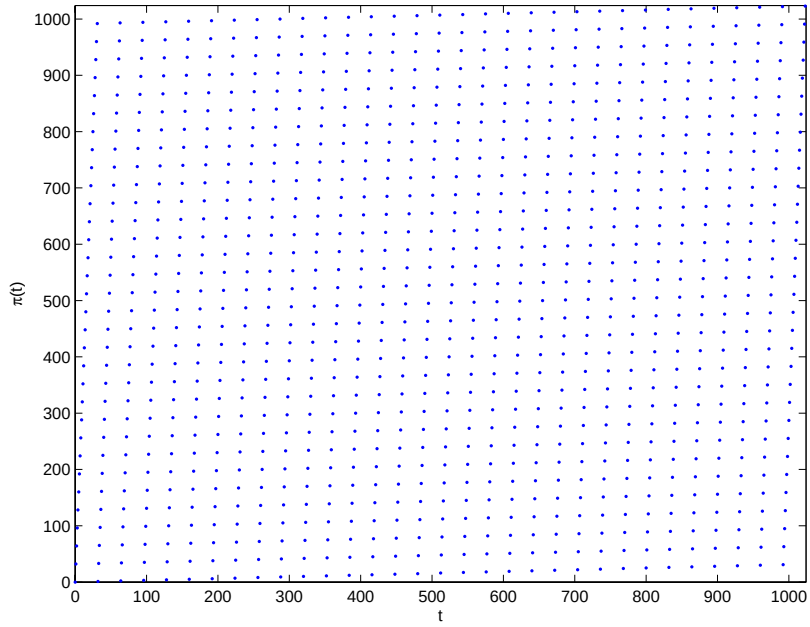

Figura 5.10: Entrelaçador de bloco, 32×32.

5.5 Entrelaçador

Nesta seção é avaliado o desempenho dos códigos turbo associado aos entrelaçadores descritos na Seção 2.3.1. Primeiramente, para se compreender como cada entrelaçador influencia o desempenho de um determinado código turbo, sua estrutura deve ser analisada.

Isto pode ser realizado através da observação do padrão de entrelaçamento, que consiste em um gráfico que plota a posição t do bit antes de ser entrelaçado *versus* a posição $\pi(t)$ do bit depois de entrelaçado[3]. Nas Fig. 5.10, 5.11 e 5.12 estão ilustrados os padrões de entrelaçamento para os entrelaçadores de bloco, pseudo-aleatório e S-aleatório.

Note que o padrão de entrelaçamento correspondente ao entrelaçador de bloco é bastante regular e apresenta um bom fator de dispersão. Já o padrão do entrelaçador aleatório, opostamente, é bastante irregular, visto que os bits são entrelaçados aleatoriamente, e apresenta um fator de dispersão baixo para alguns bits. Como uma alternativa intermediária entre os dois entrelaçadores, encontra-se o entrelaçador S-aleatório, que possui um padrão de entrelaçamento irregular, mas que no entanto está associado a um parâmetro S responsável por estabelecer um fator de dispersão mínimo entre os bits entrelaçados.

Em [9], observou-se que um entrelaçador de bloco quando utilizado

[3]Para notação vide Seção 2.3.1.

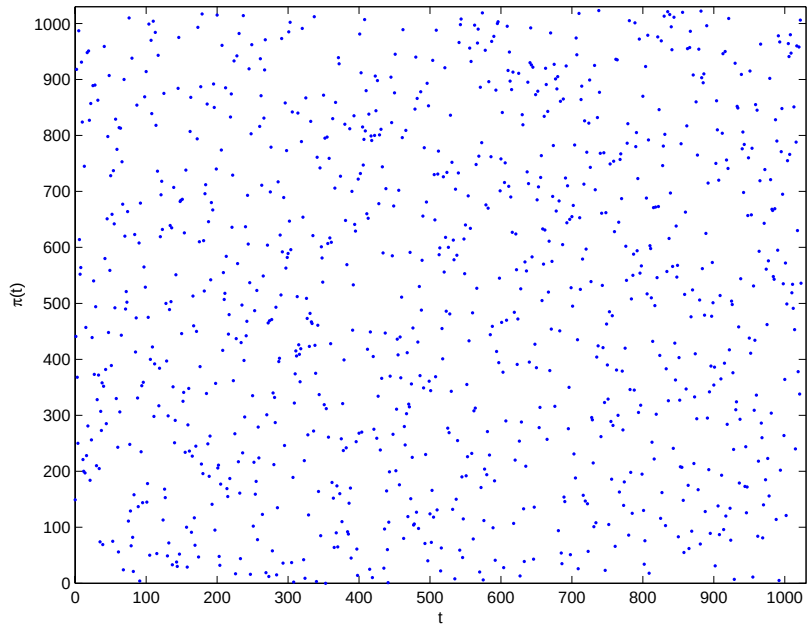

Figura 5.11: Entrelaçador aleatório.

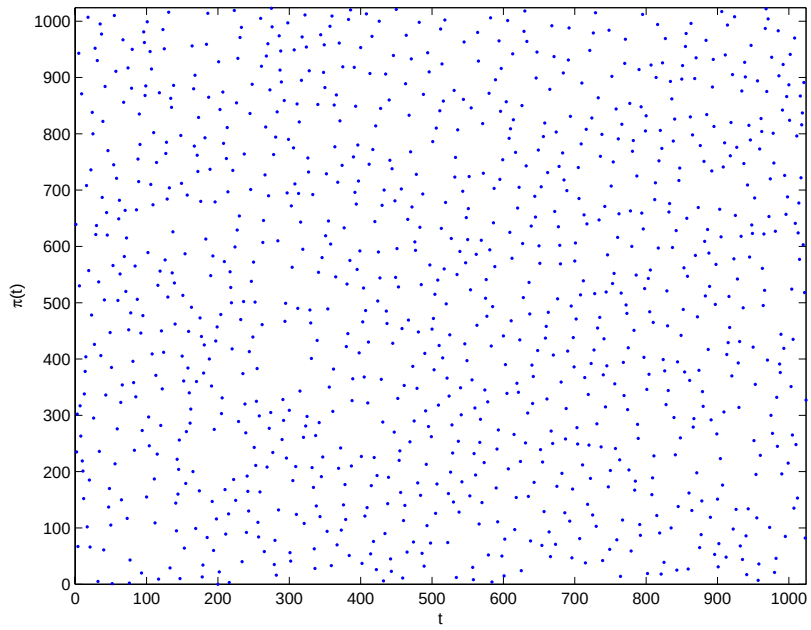

Figura 5.12: Entrelaçador S-aleatório, $S = 15$.

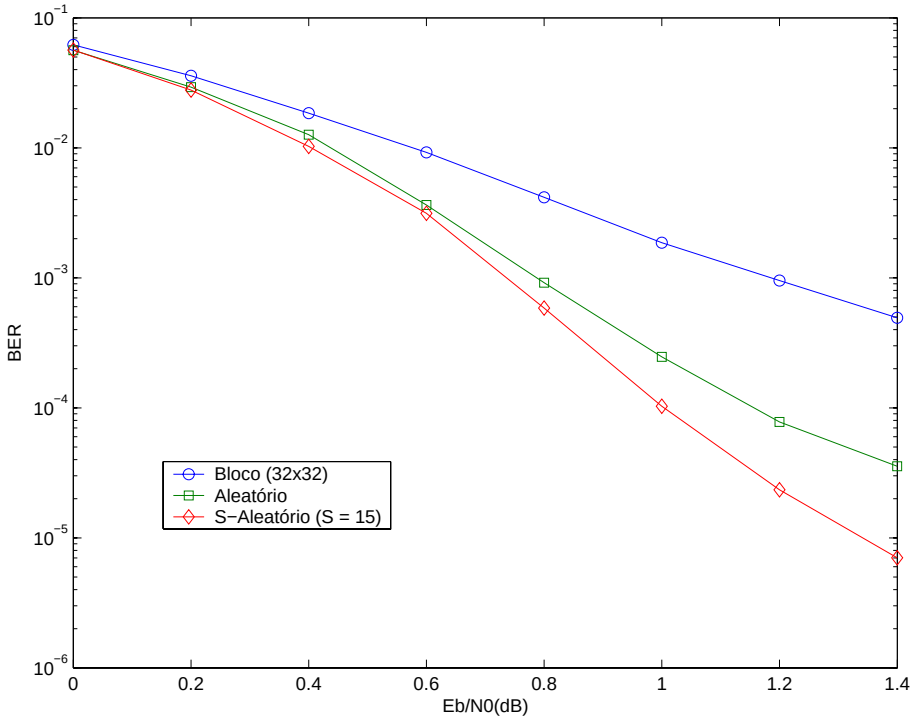

Figura 5.13: Desempenho do código turbo $\mathcal{C} = (17, 15, 2048)$ com os entrelaçadores: de bloco, pseudo-aleatório e S-aleatório. $R = 1/3$, $l = 8$, $S = 20$, MAP, HDA, BPSK, AWGN.

na codificação turbo, resulta em um desempenho inferior se comparado a entrelaçadores com padrões irregulares, como é o caso dos entrelaçadores pseudo-aleatório e S-aleatório. Isto ocorre porque o entrelaçador de bloco conduz a um código com multiplicidade η_{free} de palavras códigos com distância mínima d_{free}, maior do que a de um código turbo que utiliza entrelaçadores com padrões irregulares, conforme pode ser visto na Fig. 5.13.

Os parâmetro utilizados nesta simulação foram: código turbo $\mathcal{C} = (17, 15, 2048)$, $R = 1/3$, iteração $l = 8$, com HDA, algoritmo de decodificação MAP, para o entrelaçador S-aleatório $S = 20$, canal AWGN com modulação BPSK.

Um entrelaçador pode ser otimizado de modo a diminuir a multiplicidade η_{free}, melhorando o perfil de peso das palavras códigos do código turbo $\mathcal{C}$ e conseqüentemente seu desempenho. Isto é realizado quando o entrelaçador Π mapeia o bloco de entrada $\boldsymbol{m}$ do primeiro codificador componente $\mathcal{C}_1$ — que resultou em um bloco codificado $\boldsymbol{c}^{(1)}$ com baixo peso — em um bloco de entrada $\tilde{\boldsymbol{m}}$ para o segundo codificador componente $\mathcal{C}_2$ que gera um bloco codificado $\boldsymbol{c}^{(2)}$ com alto peso, garantindo assim uma palavra código de "bom" peso.

No caso do entrelaçador de bloco, muitas vezes, devido a sua regulari-

dade, o bloco de entrada $\boldsymbol{m}$ é entrelaçado nele mesmo, resultando em uma palavra código de baixo peso, apresentando uma multiplicidade η_{free} grande. O mesmo não ocorre para padrões de entrelaçamento irregulares.

Na Fig. 5.13, ainda podemos observar que o entrelaçador S-aleatório apresenta melhor desempenho se comparado ao entrelaçador pseudo-aleatório, isto é justificado pelo fato de o entrelaçador S-aleatório possuir um fator de dispersão melhor. A necessidade do entrelaçador possuir um bom fator de dispersão está relacionada à utilização de codificadores convolucionais recursivos como códigos componentes.

Para codificadores convolucionais recursivos, um bloco de entrada $\boldsymbol{m}$ de peso 2 corresponde a um percurso na treliça do código que inicia e finaliza no estado zero, onde o primeiro "1" é responsável por deixar o estado zero e o segundo "1" por retornar ao mesmo estado.

Denotando w_{per} o peso da seqüência de paridade correspondente ao percurso, tem-se que o peso da palavra código neste caso é igual a $i \cdot w_{\text{per}} + t$, onde i é um inteiro e t é o peso da paridade obtido nas transições ao se divergir e retornar ao estado zero provocadas pelos "1's". O peso mínimo w_{min}, ou distância mínima do código, ocorre para $i = 1$ [29].

Na codificação turbo, quando o bloco de entrada possui peso 2, deseja-se que se o codificador $\mathcal{C}_1$ produz uma seqüência de paridade com peso w_{min}, o mesmo não ocorra para $\mathcal{C}_2$. Isto pode ser evitado, projetando-se um entrelaçador, de modo que se os dois "1's" de $\boldsymbol{m}$, separados por uma distância menor que um número D de zeros, ao se entrelaçar $\boldsymbol{m}$, em $\tilde{\boldsymbol{m}}$ estejam separados por uma distância maior que $S > D$, onde S corresponde a um fator mínimo de dispersão. Este entrelaçador é um entrelaçador S-aleatório e se for utilizado, garante uma η_{free} menor para palavras códigos de baixo peso se comparado a multiplicidade de um entrelaçador pseudo-aleatório.

De maneira qualitativa, conclui-se que o entrelaçador indicado para codificação turbo, deve apresentar uma estrutura irregular e um bom fator de dispersão.

5.6 Puncionamento

Para aplicações com limitação em banda torna-se necessário o uso de modulações[4] e códigos eficientes em relação à banda disponível. Este objetivo pode ser alcançado utilizando-se códigos com altas taxas e modulações

[4]Sobre modulação, vide Seção 5.8.

com constelações com muitos pontos. O aumento da taxa de codificadores turbo, como dito anteriormente, pode ser realizado com o uso de técnicas de puncionamento.

As técnicas para obtenção de padrões(matrizes) de puncionamento para codificadores turbo descritos na literatura, restringem-se a métodos heurísticos baseados em simulações, ou algoritmos que realizam buscas exaustivas para encontrar o trio $(\mathcal{C}, \Pi, \boldsymbol{P})$, com intuito de otimizar os pesos mínimos das palavras códigos e suas multiplicidades de ocorrência [18].

Em [19], foi observado que a escolha de um padrão de puncionamento para um codificador turbo deve levar em conta o algoritmo de entrelaçamento empregado. Também foi comprovado através de simulações que para um entrelaçador pseudo-aleatório, os bits puncionados em uma seqüência de paridade são independentes dos bits puncionados na outra; e a escolha dos bits a serem puncionados em cada seqüência, não influencia no desempenho do código.

Foi desenvolvida em [27], uma técnica modificada para obtenção da distribuição de peso das palavras códigos pertencentes a codificadores turbo puncionados, e assim obter limites teóricos para probabilidade de erro.

Em [18] [19] [27], constatou-se que puncionar os bits correspondentes a seqüência sistemática, consistia em uma drástica perda de desempenho do código. Um guia para escolha de um bom padrão de puncionamento, proposto em [27], pode ser resumido nas seguintes recomendações:

1) Deve-se evitar o puncionamento dos bits sistemáticos;
2) O puncionamento deve ser aplicado uniformemente entre as seqüências de paridade $\boldsymbol{c}^{(1)}$ e $\boldsymbol{c}^{(2)}$;
3) Os bits puncionados devem estar o mais espalhado possível, e matrizes de puncionamento obtidas através de deslocamentos; cíclicos das colunas tem essencialmente o mesmo desempenho.

Note que 3) conflita com o que foi observado em [19], caso o entrelaçador utilizado seja aleatório. Nenhuma justificativa plausível foi apresentada em [19], sendo as conclusões obtidas baseadas apenas em resultados obtidos através de simulações.

Para o caso em que se convenciona não puncionar os bits sistemáticos, as taxas para codificadores puncionados são dadas por $R = k/(k+1)$, $k = 1, 2, \ldots, 16$, significando que em cada bloco de tamanho $2k$ bits, apenas $2k + 2$ bits de paridade são transmitidos.

Utilizando o simulador turbo, foram realizadas simulações de desempenho de códigos turbo $\mathcal{C} = (5, 7, 1200)$ com taxas 1/2, 2/3 e 3/4. Os se-

guintes parâmetros foram utilizados: iteração $l = 8$, com HDA, algoritmo de decodificação MAP, entrelaçador S-aleatório com $S = 15$, canal AWGN com modulação BPSK.

Como não foram encontrados na literatura padrões de puncionamento para codificadores turbo com entrelaçadores S-aleatório, a fim de se avaliar o desempenho de codificadores turbo de altas taxas quando este entrelaçador é utilizado, os padrões de puncionamento utilizados nas simulações foram escolhidos de maneira aleatória. Sendo

$$\boldsymbol{P} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{5-5}$$

o padrão utilizado para taxa $R = 1/2$,

$$\boldsymbol{P} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5-6}$$

o padrão para taxa $R = 2/3$, e

$$\boldsymbol{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{5-7}$$

o padrão utilizado para taxa 3/4.

Na Fig. 5.14, são mostradas as curvas de desempenho para códigos turbo com taxas 1/3, 1/2, 2/3 e 3/4. O limitante de Shannon para cada uma das taxa pode ser encontrado na Fig. 1.1.

Note que o padrão escolhido para $R = 3/4$ não apresentou um bom desempenho, estando a cerca de -7.8 dB do limitante de Shannon, como pode ser visto na Fig. 5.14. Isso ilustra o fato de que uma escolha sem critérios de $\boldsymbol{P}$ não é indicada para entrelaçadores S-aleatórios, sendo necessário realizar um estudo do perfil das palavras códigos pertencentes ao codificador turbo utilizado, para eleger o melhor $\boldsymbol{P}$.

5.7 Código Turbo Encurtado

Na decodificação turbo há 5 padrões típicos para os blocos na saída de um decodificador turbo, vide Seção 5.4, a fim de tentar minimizar a ocorrência dos padrões 3, 4 e 5, diminuindo dessa forma o número

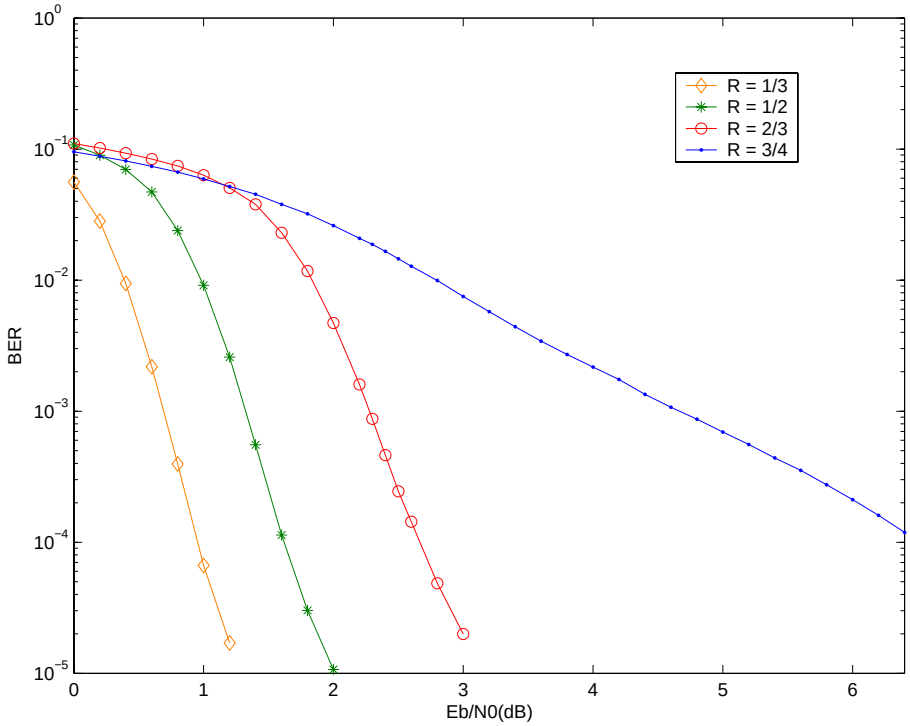

Figura 5.14: Comparação de desempenho do código turbo $\mathcal{C} = (5, 7, 1200)$ para as taxas 1/3, 1/2, 2/3 e 3/4. $l = 8$, $S = 15$, MAP, HDA, BPSK, AWGN.

de iterações necessárias para que o decodificador convirja à mensagem transmitida, resolvemos implementar um esquema, na qual são inseridos, com período T, bits conhecidos no bloco de entrada $\boldsymbol{m}$, como ilustrado na Fig. 5.15.

Neste esquema, a seqüência correspondente aos bits inseridos não é transmitida, pois o esquema foi implementado, de modo que esta seja conhecida pelo decodificador, sendo redundante sua transmissão. O novo bloco de entrada $\boldsymbol{m}_s$ é entregue ao codificador componente $\mathcal{C}_1$, enquanto que ao codificador componente $\mathcal{C}_2$ é entregue a versão permutada do mesmo, $\tilde{\boldsymbol{m}}_s$, na saída do codificador turbo encurtado obtém-se

$$\boldsymbol{c} = \left(\begin{array}{cccccccc} m_0 c_0^{(1)} c_0^{(2)} & \cdots & m_{N-1} c_{N-1}^{(1)} c_{N-1}^{(2)} & c_N^{(1)} c_N^{(2)} & \cdots & c_{N_s}^{(1)} c_{N_s}^{(2)} \end{array} \right). \qquad (5\text{-}8)$$

A taxa para este código é $R_s = N/(2N_s + N)$, onde N_s é o tamanho do bloco $\boldsymbol{m}_s$. Como $N_s > N$, o código turbo resultante $\mathcal{C}_s$ tem taxa $R_s < R$, onde $R = 1/3$, corresponde à taxa de um código $\mathcal{C}$ original cujo bloco de entrada também é $\boldsymbol{m}$, e não há inserção de bits. Neste caso, diz-se que $\mathcal{C}_s$ é um código encurtado. Ao contrário da técnica de puncionamento, a técnica de encurtamento permite obter códigos de taxas menores a partir de um código-mãe.

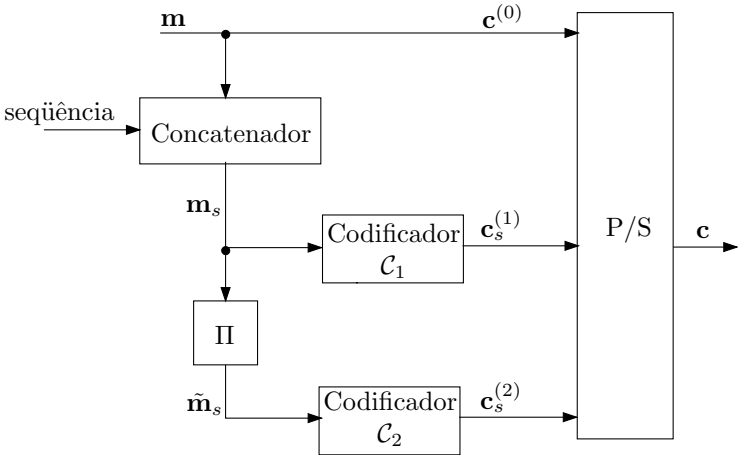

Figura 5.15: Diagrama de bloco de um codificador turbo encurtado.

A fim de se avaliar como a utilização de bits conhecidos afetam o desempenho e a convergência — número médio de iterações utilizados na decodificação — deste esquema, dois códigos turbo encurtados $\mathcal{C}_{1s}$ e $\mathcal{C}_{2s}$ foram comparados a um código-mãe $\mathcal{C}$, onde em $\mathcal{C}_{1s}$, os bits conhecidos eram todos 1, e em $\mathcal{C}_{2s}$, eram gerados aleatoriamente. Para compará-los, ambos os códigos foram puncionados de modo que $R_{1s} = R_{2s} = R = 1/3$. Nas Fig. 5.16 e Fig. 5.17 encontram-se os resultados obtidos.

Os parâmetros utilizados nas simulações foram: código turbo $\mathcal{C} = (5, 7, 1200)$, iteração $l = 8$, algoritmo de decodificação MAP, com HDA, entrelaçador S-aleatório $S = 15$, canal AWGN com modulador BPSK. Para $\mathcal{C}_{1s}$ e $\mathcal{C}_{2s}$, $T = 5$, sendo a matriz de puncionamento utilizada

$$\boldsymbol{P} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}. \tag{5-9}$$

Note que a diferença de desempenho entre as seqüências utilizadas foi insignificativa, entretanto os resultados obtidos para os códigos encurtados foram inferiores aos obtidos pelo código-mãe. Esperávamos que os resultados obtidos para os três códigos fossem equivalentes, ou melhores para os códigos encurtados. A possível justificativa para o desempenho inferior dos códigos encurtados em relação ao código-mãe, seria utilização de um padrão de puncionamento ruim para estes códigos.

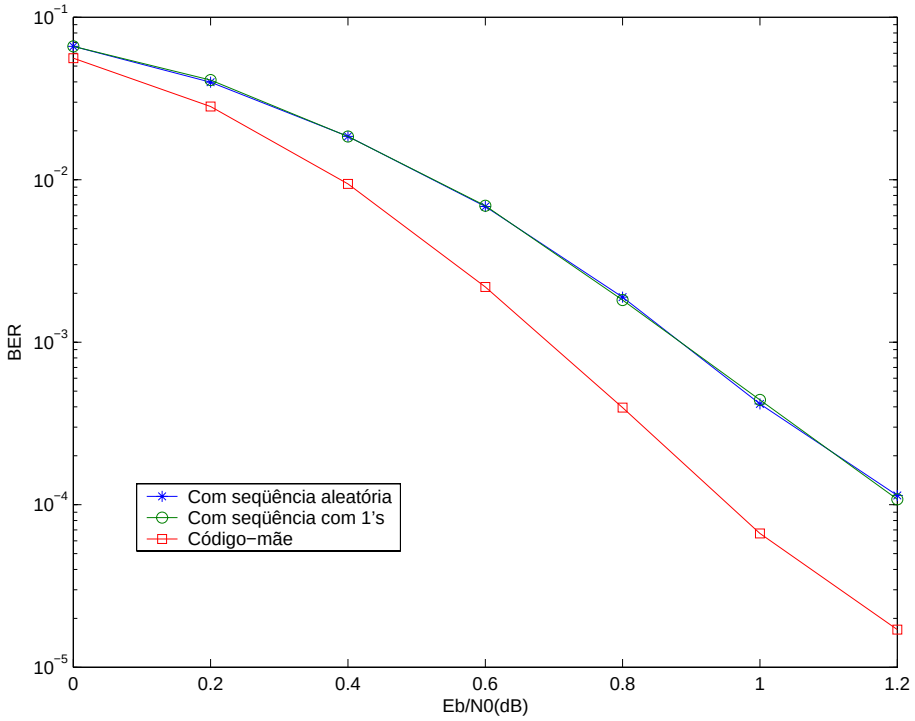

Figura 5.16: Desempenho do código turbo $\mathcal{C} = (5, 7, 1200)$ comparado aos códigos turbo encurtados $\mathcal{C}_{1s}$ e $\mathcal{C}_{2s}$ com $T = 5$. $R = 1/3$, $l = 8$, $S = 15$, MAP, HDA, BPSK, AWGN.

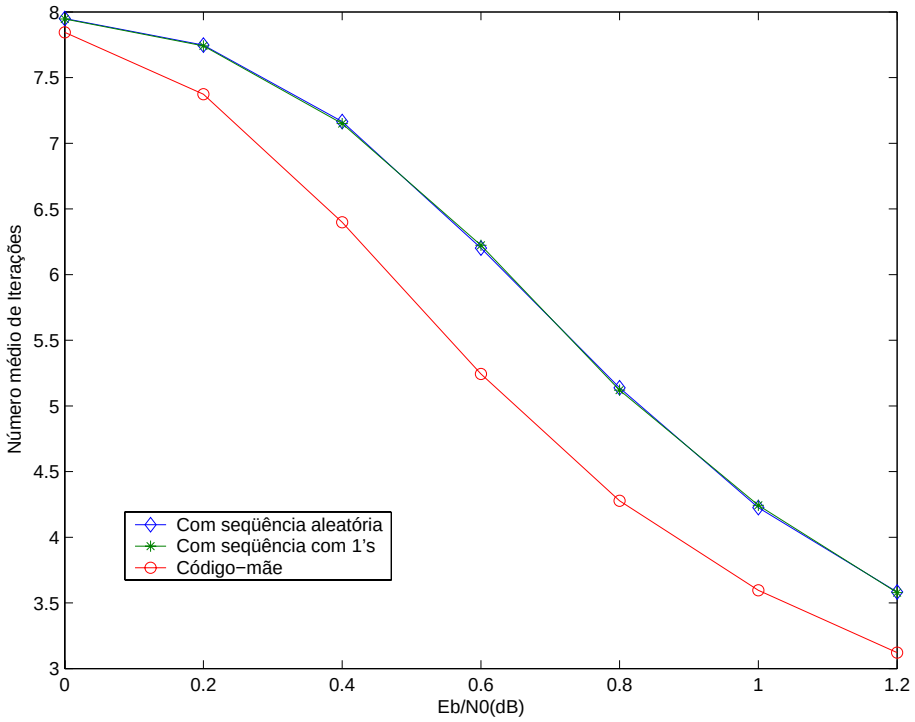

Figura 5.17: Número médio de iterações para o código turbo $\mathcal{C} = (5, 7, 1200)$ comparado aos códigos turbo encurtados $\mathcal{C}_{1s}$ e $\mathcal{C}_{2s}$ com $T = 5$. $R = 1/3$, $l = 8$, $S = 15$, MAP, HDA, BPSK, AWGN.

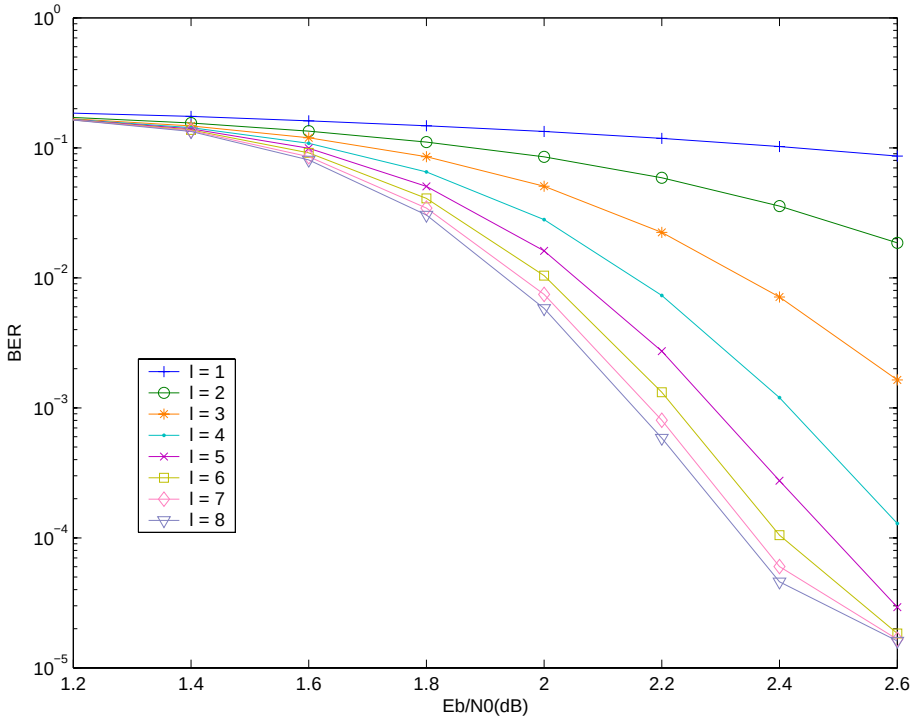

Figura 5.18: Desempenho do código turbo $\mathcal{C} = (21, 37, 2048)$ associado à modulação ASK-4. $R = 1/3$, $L = 8$, MAP, HDA, AWGN.

5.8 Turbo e Outras Modulações

Nesta seção, são mostrados resultados de simulações para um codificador turbo associado a um modulador ASK-4 cuja implementação foi realizada utilizando o desenvolvimento da Seção 4.4.1. Nessa implementação optou-se por utilizar um entrelaçador de bits aleatório para $\Pi_{\mathcal{M}}$, a fim de melhorar o desempenho do algoritmo de decodificação turbo.

Na Fig. 5.18, está ilustrado o desempenho para o código turbo $\mathcal{C} = (21, 37, 2048)$ associado à modulação ASK-4 com parâmetros: $R = 1/3$, $L = 8$, algoritmo de decodificação MAP, entrelaçador aleatório, canal AWGN, com critério de parada HDA.

A simulação do desempenho do código turbo $\mathcal{C} = (21, 37, 4096)$ associado à modulação ASK-4 com os parâmetros: $R = 1/3$, $L = 8$, algoritmo de decodificação MAP, entrelaçador S-aleatório com $S = 29$, canal AWGN, com critério de parada HDA, tem seus resultados ilustrados na Fig. 5.19.

Além do excelente desempenho se comparado com o esquema não codificado, a utilização de turbo associado a modulações mais eficientes quanto ao uso da banda disponível, foi mostrado em [5] como uma alternativa de simples de implementação e de melhor desempenho se comparada ao es-

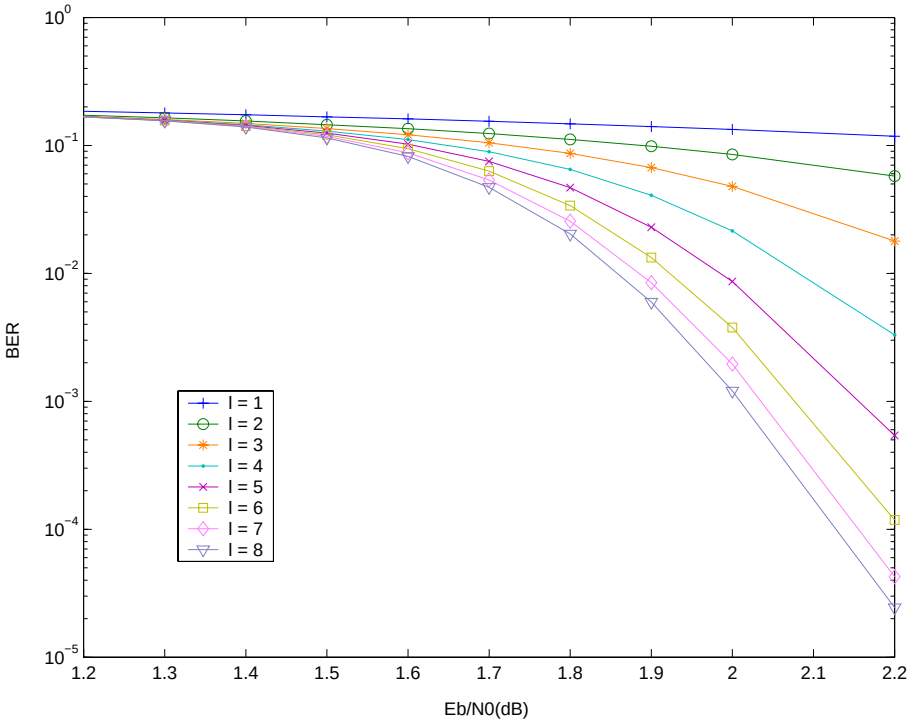

Figura 5.19: Desempenho do código turbo $\mathcal{C} = (21, 37, 4096)$ associado à modulação ASK-4. $R = 1/3$, $L = 8$, $S = 29$, MAP, HDA, AWGN.

quema convencional de codificação TCM (*Trellis Coded Modulation*).