Decodificação iterativa bidimensional utilizando códigos de Hamming e algoritmo BCJR em canais Gaussianos

Igor M. M. Souza, Maria de Lourdes M. G. Alcoforado, Valdemar C. da Rocha Jr.

***Resumo*— Neste artigo é investigado o desempenho de códigos bidimensionais simples, quando decodificados com o algoritmo BCJR em presença de ruído aditivo Gaussiano branco. Com apenas quatro iterações do algoritmo BCJR o desempenho alcançado supera o de outros algoritmos disponíveis na literatura.**

***Palavras-Chave*— Códigos de Hamming, decodificação iterativa, códigos produto.**

***Abstract*— This paper investigates the performance of simple two-dimensional codes, when decoded with the BCJR algorithm in the presence of additive white Gaussian noise. With only four iterations, the BCJR algorithm performance achieved surpasses that of other algorithms available in literature.**

***Keywords*— Hamming codes, iterative decoding, product codes.**

I. INTRODUÇÃO

Um dos grandes desafios da Engenharia de Telecomunicações é recuperar informações que num processo de transmissão ou armazenamento tenham sofrido algum tipo de alteração. Existe a necessidade de garantir a integridade dos dados enviados através de algum tipo de canal, e para tanto, todo sistema de transmissão digital precisa de algum tipo de técnica de correção de erros [1].

Os códigos turbo foram apresentados à comunidade de comunicações em 1993 por Berrou, Glaviex e Thitimajshima [2] e são considerados um dos melhores códigos corretores de erros da atualidade, sendo empregados na terceira geração de sistemas de comunicações móveis e tendo excelente desempenho em baixa relação sinal ruído (SNR).

O objetivo deste artigo é apresentar curvas de desempenho de códigos bidimensionais compostos por códigos de Hamming, relacionando valores das probabilidades de erro com os de relação sinal-ruído. É utilizada decodificação iterativa juntamente com o algoritmo BCJR. Os resultados encontrados neste trabalho são comparados com os encontrados em [3].

O modelo do sistema de comunicação utilizado é o ponto-a-ponto, onde há apenas um remetente enviando uma mensagem para um único destinatário. Para realização das simulações, é considerado que o comportamento do remetente é similar ao de uma fonte de informação binária sem memória, onde os símbolos têm a mesma probabilidade de ocorrência. A modulação BPSK (*binary phase shift key*) é empregada para a transmissão dos dados codificados através de um canal aditivo, afetado por ruído branco Gaussiano (RAGB).

A seção II trata da apresentação de um codificador de bloco bidimensional, que torna possível o uso da decodificação iterativa. Na seção III há a descrição de uma estrutura em treliça, através do uso da matriz de verificação de paridade $\mathbf{H}$ de um determinado código de bloco. A seção IV traz uma breve descrição do algoritmo de decodificação BCJR que minimiza a probabilidade de erro por bit para decodificar a palavra-código recebida. Na seção V é apresentado o algoritmo de decodificação iterativa utilizado para realização das simulações. Na seção VI, são mostrados os resultados das simulações realizadas utilizando códigos de Hamming. Por fim, na seção VII é feita uma avaliação dos resultados das simulações.

II. O CODIFICADOR

Na Figura 1 é ilustrada a estrutura bidimensional do codificador para o código C^* [4], cuja finalidade é permitir posteriormente a decodificação iterativa. O código de bloco C^* é formado tal que cada palavra-código seja um arranjo de n_1 colunas e n_2 linhas, onde cada linha é uma palavra código em C^-, código de bloco sistemático com parâmetros (n_1, k_1), e cada coluna é uma palavra código em $C^|$, código de bloco sistemático com parâmetros (n_2, k_2). Na Figura 1, P^- e $P^|$ representam os bits de paridade de C^- e $C^|$, respectivamente. Os k_1k_2 símbolos binários localizados na quina superior esquerda, representados por $\boldsymbol{u}$, são símbolos de informação. Os $(n_1 - k_1)k_2$ símbolos binários localizados na quina superior direita são calculados a partir das regras de paridade de C^- e os $(n_2-k_2)k_1$ símbolos binários localizados na quina inferior esquerda são calculados a partir das regras de paridade de $C^|$.

Como não está sendo utilizados os bits de paridade das paridades, C^* terá como parâmetros $((n_1 * n_2 - (n_1 - k_1) * (n_2 - k_2), k_1 * k_2)$. Ao não utilizar esses bits obtém-se uma maior taxa de transmissão e um menor número de operações na codificação e decodificação.

III. TRELIÇA PARA CÓDIGOS DE BLOCO LINEARES

Seja C um código de bloco linear binário com parâmetros (n, k). Os vetores correspondentes às mensagens são denotados por $\boldsymbol{u} = \{u_1, u_2, ..., u_k\}$ e as correspondentes palavras

Igor M. M. Souza e Maria de Lourdes M. G. Alcoforado¸ Núcleo de Pesquisa em Telecomunicações, Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Pernambuco, Brasil, E-mails: igor.poli@yahoo.com.br, mlmga@poli.br. Valdemar C. da Rocha Jr., Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, Recife, Pernambuco, Brasil, E-mail: vcr@ufpe.br. Este trabalho foi financiado pelo PIBIC POLI 2010.

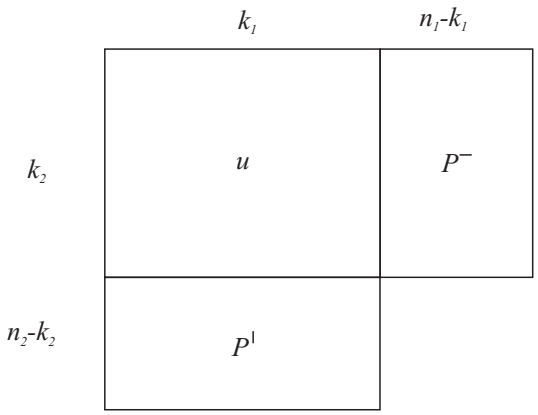

Fig. 1. Estrutura de um código bidimensional C^* com $C^- = (n_1, k_1)$ e $C^| = (n_2, k_2)$ como códigos componentes.

código por $\boldsymbol{v} = \{v_1, v_2, ..., v_n\}$. A treliça correspondente ao código C é definida a partir da matriz de verificação de paridade $\mathbf{H}$ [5]. Seja $\mathbf{h}_i$, $i = \{1, 2, ..., n\}$ os vetores-coluna da matriz $\mathbf{H}$. Os estados $\mathbf{S}_t$ da treliça são definidos de tal forma que:

$$\mathbf{S}_0 = \mathbf{0},$$

$$\mathbf{S}_t = \mathbf{S}_{t-1} + v_t\mathbf{h}_t, t = 1, ..., n. \quad (1)$$

O estado atual $\mathbf{S}_t$ é sempre função do estado anterior $\mathbf{S}_{t-1}$, bem como do bit v_t. Repetindo (1) para todas as palavras código, é possível encontrar a treliça correspondente ao código de bloco, conforme ilustrado no exemplo 2.1.

Exemplo 2.1: A treliça encontrada a partir da matriz de verificação de paridade $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix},$$

representando o código de Hamming $(7, 4)$ [1], está ilustrada na Figura 2, onde cada estado possível é mostrado na base binária.

IV. ALGORITMO BCJR

O algoritmo proposto por Bahl *et al* (BCJR) pode ser aplicado a qualquer código que tenha uma treliça associada [5]. Seja $\boldsymbol{r} = \{r_1, r_2, ..., r_n\}$ a saída do canal, correspondente a palavra código $\boldsymbol{v}$ contaminada com ruído RAGB. A variável aleatória r_t, no instante de tempo t, é definida como $r_t = (2v_t - 1) + q_t$, onde q_t representa amostras de ruídos, independentes, com variância σ^2 e média 0 (zero). O algoritmo BCJR calcula a razão de log-verossimilhança, $\Lambda(v_t)$, associada a cada símbolo v_t da palavra código $\boldsymbol{v}$ como mostrado em (2), em que os termos $P\{v_t = 1|\boldsymbol{r}\}$ e $P\{v_t = 0|\boldsymbol{r}\}$ são as probabilidades *a posteriori* dos bits que compõem a palavra código $\boldsymbol{v}$.

$$\Lambda(v_t) = \log(\frac{P\{v_t = 1|\boldsymbol{r}\}}{P\{v_t = 0|\boldsymbol{r}\}}). \quad (2)$$

Considerando que $\hat{v}_t$ é o valor estimado para v_t, a decisão é feita da seguinte forma: $\Lambda(v_t) \geq 0 : \hat{v}_t = 1$, ou $\Lambda(v_t) < 0 : \hat{v}_t = 0$. O módulo de $\Lambda(v_t)$ representa a informação suave associada com o valor abrupto (0 ou 1) estimado de v_t. Considerando que $\boldsymbol{r}_1^t = \{r_1, r_2, ..., r_t\}$, $\boldsymbol{r}_{t+1}^n = \{r_{t+1}, r_{t+2}, ..., r_n\}$ e introduzindo as funções de probabilidade definidas em [5]:

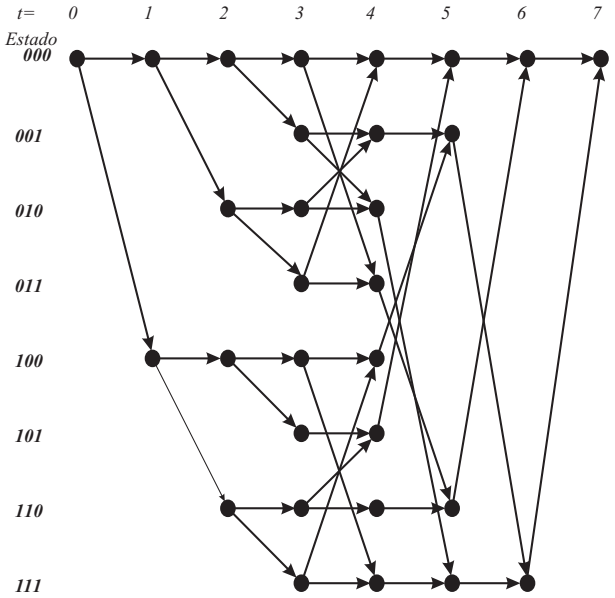

Fig. 2. Treliça obtida a partir da matriz de verificação de paridade do código de Hamming $(7, 4)$.

$$\alpha_t(\mathbf{m}) = \mathrm{p}\{\mathbf{S}_t = \mathbf{m}, \boldsymbol{r}_1^t\}, \quad (3)$$

$$\beta_t(\mathbf{m}) = \mathrm{p}\{\boldsymbol{r}_{t+1}^n|\mathbf{S}_t = \mathbf{m}\}, \quad (4)$$

$$\gamma_i(r_t, \mathbf{m}', \mathbf{m}) = \mathrm{p}\{u_t = i, \mathbf{S}_t = \mathbf{m}, r_t|\mathbf{S}_{t-1} = \mathbf{m}'\}, \quad (5)$$

tem-se que:

$$\Lambda(v_t) = \log \frac{\sum_{\mathbf{m}} \sum_{\mathbf{m}'} \gamma_1(r_t, \mathbf{m}', \mathbf{m})\alpha_{t-1}(\mathbf{m}')\beta_t(\mathbf{m})}{\sum_{\mathbf{m}} \sum_{\mathbf{m}'} \gamma_0(r_t, \mathbf{m}', \mathbf{m})\alpha_{t-1}(\mathbf{m}')\beta_t(\mathbf{m})}. \quad (6)$$

As funções de probabilidade $\alpha_t(\mathbf{m})$ e $\beta_t(\mathbf{m})$ são calculadas de maneira recursiva [5]. As probabilidades $\gamma_i(r_t, \mathbf{m}', \mathbf{m})$ são determinadas a partir das probabilidades de transição do canal contaminado com ruído RAGB e das probabilidades de transição da treliça do codificador [5]. $\alpha_t(\mathbf{m})$ pode ser calculado por meio de:

$$\alpha_t(\mathbf{m}) = \sum_{i=0}^{1} \sum_{\mathbf{m}'=0}^{M-1} \alpha_{t-1}(\mathbf{m}')\gamma_i(r_t, \mathbf{m}', \mathbf{m}). \quad (7)$$

Considerando que a treliça inicializa no estado $S_0 = \mathbf{0}$, as condições de contorno são as seguintes:

$$\alpha_0(\mathbf{0}) = 1 \text{ e } \alpha_0(\mathbf{m}) = 0, \text{ para } \mathbf{m} \neq 0. \quad (8)$$

Similarmente, $\beta_t(\mathbf{m})$ pode ser calculado por meio de:

$$\beta_t(\mathbf{m}) = \sum_{i=0}^{1} \sum_{\mathbf{m}'=0}^{M-1} \gamma_i(r_{t+1}, \mathbf{m}, \mathbf{m}')\beta_{t+1}(\mathbf{m}'). \quad (9)$$

As condições de contorno apropriadas quando o codificador é levado a terminar no estado $\mathbf{0}$ isto é, $S_N = \mathbf{0}$ são:

$$\beta_N(\mathbf{0}) = 1 \text{ e } \beta_N(\mathbf{m}) = 0, \text{ para } \mathbf{m} \neq 0. \quad (10)$$

As probabilidades $\gamma_i(r_t, \mathbf{m}', \mathbf{m})$ podem ser determinadas a partir das probabilidades de transição do canal contaminado com ruído RAGB e das probabilidades de transição da treliça do codificador como visto em (11).

$$\begin{aligned}\gamma_i(r_t, \mathbf{m}', \mathbf{m}) &= \mathrm{P}\{\mathbf{S}_t = \mathbf{m} | \mathbf{S}_{t-1} = \mathbf{m}'\} \\ &\quad \mathrm{p}\{r_t | u_t = i, \mathbf{S}_t = \mathbf{m}, \mathbf{S}_{t-1} = \mathbf{m}'\} \\ &\quad \mathrm{P}\{u_t = i | \mathbf{S}_t = \mathbf{m}, \mathbf{S}_{t-1} = \mathbf{m}'\}. \quad (11)\end{aligned}$$

As probabilidades de transição $\mathrm{P}\{\mathbf{S}_t = \mathbf{m} | \mathbf{S}_{t-1} = \mathbf{m}'\}$ são definidas pelas probabilidades *a priori* dos bits de entrada. Quando os bits de entrada são equiprováveis $\mathrm{P}\{u_t = 1\} = \mathrm{P}\{u_t = 0\} = 1/2$ então $\mathrm{P}\{\mathbf{S}_t = \mathbf{m} | \mathbf{S}_{t-1} = \mathbf{m}'\} = 1/2$. A probabilidade $\mathrm{P}\{r_t | u_t = i, \mathbf{S}_t = \mathbf{m}, \mathbf{S}_{t-1} = \mathbf{m}'\}$ é a probabilidade de transição do canal contaminado com ruído RAGB, podendo ser escrita da seguinte forma:

$$\mathrm{p}\{r_t | u_t = i, \mathbf{S}_t = \mathbf{m}, \mathbf{S}_{t-1} = \mathbf{m}'\} = \mathrm{p}\{r_t | v_t\}. \quad (12)$$

Considerando que o canal é contaminado com ruído RAGB com variância σ^2, tem-se que:

$$\mathrm{p}\{r_t | v_t\} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r_t - v_t)^2}{2\sigma^2}\right), \quad (13)$$

assim,

$$\mathrm{p}\{r_t | v_t = -1\} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r_t + 1)^2}{2\sigma^2}\right),$$

$$\mathrm{p}\{r_t | v_t = +1\} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(r_t - 1)^2}{2\sigma^2}\right).$$

A probabilidade $\mathrm{P}\{u_t = i | \mathbf{S}_t = \mathbf{m}, \mathbf{S}_{t-1} = \mathbf{m}'\}$ é igual a 0 ou 1 uma vez que o codificador convolucional é uma máquina determinística.

O algoritmo BCJR portanto segue da seguinte forma:

As condições iniciais de $\alpha_0(\mathbf{m})$ e $\beta_N(\mathbf{m})$ são vistas em (8) e (10); Quando $\boldsymbol{r}_t$ é recebido, o decodificador calcula $\gamma_i(r_t, \mathbf{m}', \mathbf{m})$ usando (11) e $\alpha_t(\mathbf{m})$ usando (7). Os valores de $\alpha_t(\mathbf{m})$ são armazenados para todo t e $\mathbf{m}$; Depois que a seqüência completa $\boldsymbol{r}_1^N$ é recebida, o decodificador calcula de forma recursiva $\beta_t(\mathbf{m})$ usando (9). O valor de $\beta_t(\mathbf{m})$ calculado pode ser multiplicado pelo $\alpha_t(\mathbf{m})$ e $\gamma_i(r_t, \mathbf{m}', \mathbf{m})$ apropriado para obtenção de (6).

V. Decodificação iterativa

O decodificador turbo utiliza o princípio da decodificação iterativa [6] e consiste de dois decodificadores componentes concatenados em série. Neste trabalho os dois decodificadores empregados usam o algoritmo BCJR. O código de bloco utilizado C^* é formado tal como ilustrado na Figura 1. Não é feito uso dos bits de paridade das paridades, portanto a taxa de transmissão é $R = \frac{k_1 k_2}{n_1 n_2 - (n_1 - k_1)(n_2 - k_2)}$. A estrutura do código produto permite dois passos de decodificação separados, horizontal e vertical, na qual é introduzida a notação $p_t^1(1)$ e $p_t^1(0)$ para representar respectivamente as probabilidades *a priori* $\mathrm{P}\{v_t = 1\}$ e $\mathrm{P}\{v_t = 0\}$, na entrada do primeiro decodificador (decodificação horizontal). Na entrada do segundo decodificador (decodificação vertical) as probabilidades *a priori* são, de forma análoga, representadas por $p_t^2(1)$ e $p_t^2(0)$. A igualdade (6) pode ser reescrita para calcular as razões de Log-Verossimilhança [6] dos decodificadores de C^- e $C^|$, respectivamente como :

$$\Lambda^-(v_t) = \log \frac{p_t^1(1)}{p_t^1(0)} + \frac{2r_t}{\sigma^2} + \Lambda_e^-(v_t). \quad (14)$$

$$\Lambda^|(v_t) = \log \frac{p_t^2(1)}{p_t^2(0)} + \frac{2r_t}{\sigma^2} + \Lambda_e^|(v_t). \quad (15)$$

Os seguintes passos para decodificação iterativa devem ser seguidos:

1) Iniciar a primeira iteração usando o valor zero para a informação a priori, i.e., fazer $\log \frac{p_t^1(1)}{p_t^1(0)} = 0$. Assume-se que as probabilidades dos símbolos emitidos pela fonte de informação são equiprováveis.
2) Decodificar horizontalmente e obter a informação extrínsica horizontal usando (14) como mostrado a seguir:

$$\Lambda_e^-(v_t) = \Lambda^-(v_t) - (\frac{2r_t}{\sigma^2} + \log \frac{p_t^1(1)}{p_t^1(0)}). \quad (16)$$

3) Considerar $\Lambda_e^-(v_t) = \log \frac{p_t^2(1)}{p_t^2(0)}$.
4) Decodificar verticalmente, e usar (15) para obter a informação extrínseca vertical:

$$\Lambda_e^|(v_t) = \Lambda^|(v_t) - (\frac{2r_t}{\sigma^2} + \log \frac{p_t^2(1)}{p_t^2(0)}). \quad (17)$$

5) Considerar $\Lambda_e^|(v_t) = \frac{p_t^1(1)}{p_t^1(0)}$.
6) Se o número de iterações já é suficiente para tomar a decisão, ir para o passo 7, senão ir para o passo 2.
7) As saídas suaves são:

$$\Lambda(v_t) = \frac{2r_t}{\sigma^2} + \Lambda_e^-(v_t) + \Lambda_e^|(v_t). \quad (18)$$

O processo descrito anteriormente está ilustrado na Figura 3, onde **DEC1** e **DEC2** representam a decodificação na horizontal e na vertical, respectivamente. O vetor recebido a ser decodificado é $\boldsymbol{r}$ e o resultado da decodificação é $\hat{\boldsymbol{v}}$, ou seja, a estimativa.

VI. Resultados das simulações

Observou-se nas simulações que apenas quatro iterações, em cada caso, foram suficientes para atingir um desempenho satisfatório. Foram realizadas algumas simulações com quatro iterações cada, sendo utilizados códigos de Hamming para compor o código C^*. A Figura 4 ilustra a simulação utilizando o código de Hamming $(15, 11)$ para compor os códigos C^- e $C^|$. A Figura 5 ilustra a simulação utilizando o código de Hamming $(31, 26)$ para compor os códigos C^- e $C^|$. Na Figura 6 é utilizado o código de Hamming $(63, 57)$ para compor os códigos C^- e $C^|$.

Considerando a probabilidade de erro de 10^{-3} e analisando a Figura 4, há um ganho de cerca de $1dB$ quando compara-se a curva da iteração 1 com a curva da iteração 2. Comparando a iteração 2 com a iteração 3, o ganho é de cerca de $0.2dB$. Em relação a Figura 5, há um ganho de cerca de 0.8dB quando compara-se a curva da iteração 1 com a da iteração 2. Um

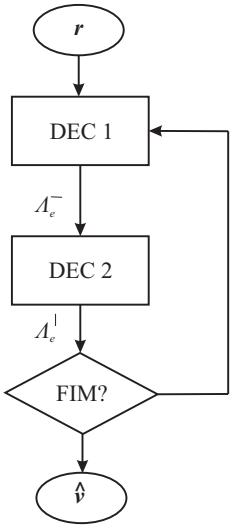

Fig. 3. Etapas da decodificação iterativa.

ganho de 0.2dB da iteração 2 em relação à iteração 3. No caso da Figura 6, há um ganho de 1 db quando compara-se a curva relacionada a iteração 1 com a curva da iteração 3.

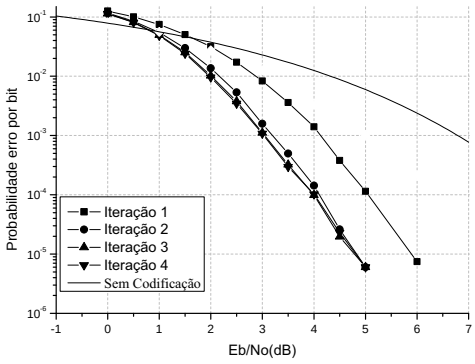

Fig. 4. Desempenho da decodificação iterativa com $C^- = C^| = (15, 11)$.

VII. CONCLUSÕES

Os resultados obtidos neste artigo, por meio de simulação computacional, ilustrados nas Figuras 4, 5 e 6, indicam que em determinadas faixas de E_b/N_0 o desempenho alcançado é significativamente superior àquele encontrado em [3], mesmo sem ter sido feito o uso dos bits de paridade das paridades na decodificação. Considerando o caso específico da Figura 6, para $E_b/N_0 = 4dB$ obtivemos uma probabilidade de erro por bit de quase 10^{-6}, enquanto o valor encontrado em [3] não chega a 10^{-2}. No entanto, a alta complexidade computacional do algoritmo BCJR tem levado ao uso e a busca de vários outros algoritmos de decodificação. Em [3] são apresentados excelentes resultados pelo uso do algoritmo de decodificação introduzido por Pyndiah [7], para isto foi feita uma seleção de parâmetros adaptativos.

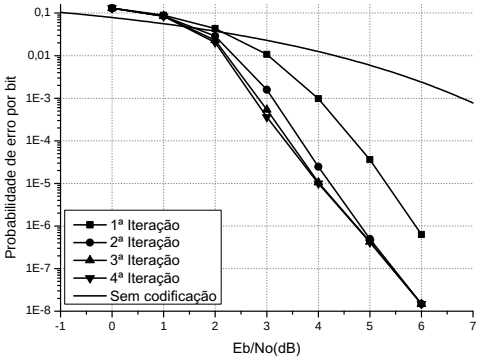

Fig. 5. Desempenho da decodificação iterativa com $C^- = C^| = (31, 26)$.

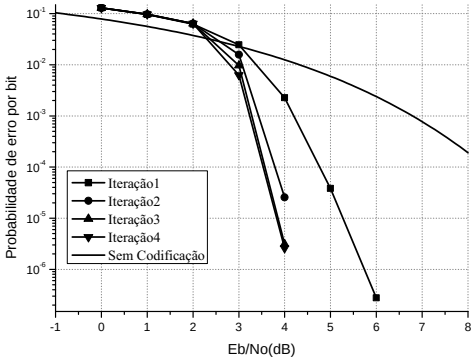

Fig. 6. Desempenho da decodificação iterativa com $C^- = C^| = (63, 57)$.

REFERÊNCIAS

[1] S. Lin, D. Costello Jr., "Error Control Coding: Fundamentals and Applications", Prentice-Hall Inc., 2004.

[2] C. Berrou, A. Glavieux and P. Thitimajshima,"Near Shannon limit, error-correcting coding and decoding: turbo codes ", *IEEE International Conference on Communications (ICC'93)*, vol. 2/3, pp. 1064-1071, 1993.

[3] Y. He, P.C. Ching, "Performance evaluation of adaptive two-dimensional turbo product codes composed of Hamming codes" *Proceedings of the 2007 IEEE International Conference on Integration Technology*, Shenzhen, China, 2007.

[4] P. Elias, "Error-free coding", *IRE Trans. Inform. Theory*, vol. 4, pp. 29-37, 1954.

[5] L. R. Bahl, J. Cocke, F. Jelinek and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate", *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, 1974.

[6] J.Hagenauer, "Iterative Decoding of Binary Block and Convolutional Codes", *IEEE Trans. Inform. Theory*, vol. 42, No 2, pp.429-445, 1996.

[7] R. Pyndiah "Near-optimum decoding of product codes: Block turbo codes", *IEEE trans. Commun.*, vol. 46, pp. 1003-1010, 1998.