Uso do algoritmo BCJR e decodificação turbo em sistemas com codificação de bloco

Igor Menezes Marinho de Souza e Maria de Lourdes Melo Guedes Alcoforado

***Resumo*— Neste artigo decodificação iterativa usando algotitmo BCJR e códigos produto, sobre o canal contaminado com ruído aditivo Gaussiano branco é investigada. Não é feito uso dos bits de paridade das paridades. São feitas simulações computacionais e geradas curvas de desempenho, relacionando valores das probabilidades de erro com os de relação sinal-ruído.**

***Palavras-Chave*— Decodificação iterativa, algoritmo BCJR, códigos de bloco.**

***Abstract*— In this paper iterative decoding using BCJR algorithm and product codes on the additive white Gaussian noise channel is investigated. It is not used the check on check bits. The simulation results are presented and the curves obtained relates the bit error probability and signal to noise ratio.**

***Keywords*— Iterative decoding, BCJR algorithm, block codes.**

I. INTRODUÇÃO

Nos últimos anos muitas estruturas de decodificação iterativa usando códigos de bloco têm sido projetadas [1][2], sendo bastante utilizados códigos produto pelas suas propriedades atrativas para implementações práticas e também por estarem em uso em inúmeros padrões de telecomunicações. Os códigos produto foram introduzidos na literatura por Elias [3] e são construídos pela concatenação em série de dois ou mais códigos de bloco lineares. Apresentam a vantagem de a partir de códigos mais simples, gerarem códigos mais robustos com maior comprimento.

O objetivo deste artigo é o estudo e a implementação de sistemas codificados para simulação de códigos produto, sem o uso dos bits gerados devido à codificação dos bits de paridade, isto é, paridade das paridades [4], e decodificadores em presença de ruído aditivo Gaussiano branco (RAGB). A seção II traz uma breve revisão sobre a técnica de geração de treliça para códigos de bloco lineares necessária ao algoritmo de decodificação BCJR. Na seção III é vista a decodificação iterativa para uso em códigos de bloco lineares. A seção IV ilustra os resultados das simulações através de curvas de desempenho geradas e finalmente na seção V a conclusão e comentários finais.

II. BCJR E TRELIÇA

O algoritmo proposto por Bahl *et al* (BCJR) pode ser aplicado a qualquer código que tenha uma treliça associada [5]. Seja $C(n,k)$ um código de bloco linear binário. Os vetores correspondentes às mensagens são denotados por $\vec{u} = \{u_1, u_2, ..., u_k\}$ e as correspondentes palavras código por $\vec{v} = \{v_1, v_2, ..., v_n\}$. A treliça correspondente ao código C é definida a partir da matriz de verificação de paridade **H**. Seja $\vec{h_i}$, $i = \{1, 2, ..., n\}$ os vetores-coluna da matriz **H**. Os estados da treliça são definidos como S_t, de tal forma que:

$$S_0 = \vec{0},$$
$$S_t = S_{t-1} + v_t\vec{h_t}, t = 1, ..., n. \quad (1)$$

O estado atual S_t é sempre função do estado anterior S_{t-1}, bem como do bit v_t. Repetindo (1) para todas as palavras código, é possível encontrar a treliça mínima [6] correspondente ao código de bloco.

Seja $\vec{r} = \{r_1, r_2, ..., r_n\}$ a saída do canal com modulação BPSK(*binary phase shift key*), correspondente a palavra código $\vec{v}$ 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 $\vec{v}$ como mostrado em (2), em que os termos $P\{v_t = 1|\vec{r}\}$ e $P\{v_t = 0|\vec{r}\}$ são as probabilidades *a posteriori* dos bits que compõem a palavra código $\vec{v}$.

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

Considerando que $\hat{v}_t$ é o valor estimado para v_t, a decisão será 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]:

$$\alpha_t(m) = \mathrm{p}\{S_t = m, \boldsymbol{r}_1^t\}, \quad (3)$$
$$\beta_t(m) = \mathrm{p}\{\boldsymbol{r}_{t+1}^n | S_t = m\}, \quad (4)$$
$$\gamma_i(r_t, m', m) = \mathrm{p}\{u_t = i, S_t = m, \boldsymbol{r}_t | S_{t-1} = m'\}, \quad (5)$$

tem-se que:

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

As funções de probabilidade $\alpha_t(m)$ e $\beta_t(m)$ são calculadas de maneira recursiva [5]. As probabilidades $\gamma_1(r_t, m', m)$ são determinadas a partir das probabilidades de transição do canal contaminado com ruído RAGB e das probabilidades de transição da treliça do codificador [5].

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. Este trabalho foi financiado pelo PIBIC POLI 2010.

III. DECODIFICAÇÃO ITERATIVA

Para ser efetuada a decodificação iterativa[7], o código de bloco utilizado C^* é formado tal que cada palavra-código é um arranjo de n_1 colunas e n_2 linhas, onde cada linha corresponde a uma palavra código em C^-, código de bloco sistemático com parâmetros (n_1, k_1), e cada coluna corresponde a uma palavra código em $C^|$, código de bloco sistemático com parâmetros (n_2, k_2). Não é feito uso dos bits de paridade das paridades, portanto a taxa de transmissão é $R = \frac{k_1 k_2}{n_1 n_2 - (n_1 - k_1)(n_2 - k_2)}$. O não uso desses bits leva a uma perda de desempenho na decodificação. Por outro lado tem-se uma maior taxa de transmissão e um menor número de operações na decodificação. 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 [4] dos decodificadores de C^- e $C^|$, respectivamente como :

$$\Lambda^-(v_t) = \log \frac{p_t^1(1)}{p_t^1(0)} + \frac{2r_t}{\sigma^2} + \Lambda_e^-(v_t). \quad (7)$$

$$\Lambda^|(v_t) = \log \frac{p_t^2(1)}{p_t^2(0)} + \frac{2r_t}{\sigma^2} + \Lambda_e^|(v_t). \quad (8)$$

Os seguintes passos para decodificação iterativa devem ser seguidos:

1) Inicializar a informação *a priori* para a primeira iteração $\log \frac{p_t^1(1)}{p_t^1(0)} = 0$. Assume-se que as probabilidades dos símbolos emitidos pela fonte de informação são equiprováveis.
2) Decodificar horizontalmente e obter a informação extrínsica horizontal usando (7) como mostrado abaixo:

$$\Lambda_e^-(v_t) \quad = \quad \Lambda^-(v_t) - (\frac{2r_t}{\sigma^2} + \log \frac{p_t^1(1)}{p_t^1(0)}). \quad (9)$$

3) Considerar $\Lambda_e^-(v_t) = \log \frac{p_t^2(1)}{p_t^2(0)}$.
4) Decodificar verticalmente, e usar (8) para obter a informação extrínseca vertical:

$$\Lambda_e^|(v_t) \quad = \quad \Lambda^|(v_t) - (\frac{2r_t}{\sigma^2} + \log \frac{p_t^2(1)}{p_t^2(0)}). \quad (10)$$

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:

$$\Lambda(v_t) \quad = \quad \frac{2r_t}{\sigma^2} + \Lambda_e^-(v_t) + \Lambda_e^|(v_t). \quad (11)$$

IV. RESULTADOS DAS SIMULAÇÕES

As Figuras 1 e 2 ilustram gráficos contendo curvas de desempenho que relacionam probabilidade de erro por bit *versus* relação sinal ruído (Pe x E_b/N_0). Para geração da Figura 1, o código C^* é composto tal que $C^- = C^| = C(31, 26)$, códigos de Hamming, sendo $R = 0,703$. Em relação a Figura 2, o código C^* é composto tal que $C^- = C^| = C(24, 12)$, códigos de Golay extendidos, sendo $R = 0,333$. Em ambos os casos são realizadas 4 iterações.

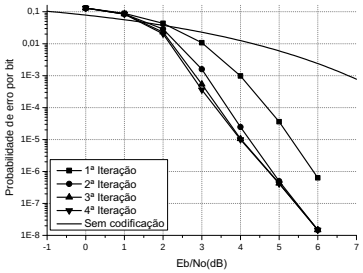

Fig. 1. Desempenho da decodificação iterativa com $C^- = C^| = C(31, 26)$.

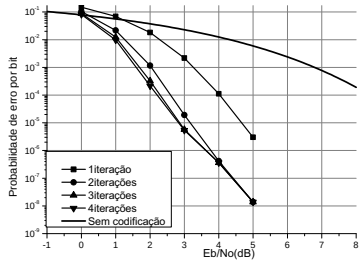

Fig. 2. Desempenho da decodificação iterativa com $C^- = C^| = C(24, 12)$.

V. CONCLUSÕES

Mesmo sem o uso dos bits de paridade das paridades, é possível notar um ganho de aproximadamente 2,5dB para probabilidade de erro por bit de 10^{-3}, quando compara-se o sistema codificado (1 iteração) com o sistema sem codificação, em ambos os casos simulados. As curvas obtidas para decodificação iterativa, Figuras 1 e 2, mostram uma melhoria de desempenho a cada nova iteração. A partir da terceira iteração a probabilidade de erro por bit passa a não melhorar significativamente. Quando compara-se as curvas relacionadas a primeira e segunda iteração para uma probabilidade de erro por bit de 10^{-4} é possível perceber um ganho de aproximadamente 1dB na Figura 1 e de 1,5dB na Figura 2.

REFERÊNCIAS

[1] R. Pyndiah "Near-optimum decoding of product codes: Block turbo codes", *IEEE trans. Commun.*, vol. 46, pp. 1003-1010, 1998.
[2] R. Pyndiah, P. Adde, R. Zhou, Block Turbo Codes: Ten years later, IEE Seminar on Sparse Graph Codes, October 2004.
[3] P. Elias, "Error-free coding", *IRE Trans. Inform. Theory*, vol. 4, pp. 29-37, 1954.
[4] J.Hagenauer, "Iterative Decoding of Binary Block and Convolutional Codes", *IEEE Trans. Inform. Theory*, vol. 42, No 2, pp.429-445, 1996.
[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] Robert J. McEliece, "On the BCJR trellis for linear block codes", *IEEE Trans. Inform. Theory*, vol. 42, pp. 1072-1092, 1996.
[7] 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.