Noções Sobre Decodificação Turbo: Um Estudo de Caso para os Códigos SPC-TPC

Dayan Adionel Guimarães

Departamento de Telecomunicações – Inatel. E-mail: dayan@inatel.br

Resumo - ***Neste tutorial são apresentados os principais conceitos sobre o processo de decodificação turbo, com enfoque na decodificação turbo de códigos de bloco. Os códigos produto de paridade simples foram adotados como estudo de caso. A decodificação dos códigos componentes é realizada por meio de um algoritmo MAP símbolo-a-símbolo. Alguns resultados de simulação com modulação BPSK com detecção coerente em canal AWGN e com desvanecimento plano do tipo Rayleigh são fornecidos e discutidos.***

Palavras-chave - Decodificação Turbo, Códigos Produto de Paridade Simples, Códigos Turbo de Bloco.

Abstract - ***In this tutorial, the main concepts associated to the turbo decoding process are presented, with emphasis on turbo decoding of block codes. Single-Parity Check Turbo Product Codes (SPC-TPC) are considered as a case study. A symbol-by-symbol MAP algorithm is used for decoding the SPC-TPC component codes. Some simulation results using coherent BPSK modulation on the AWGN and flat Rayleigh fading channels are also presented and discussed.***

Key words - Turbo Decoding, Single-Parity Check Product Codes, Block Turbo Codes.

I - INTRODUÇÃO

EM 1966 G. DAVID FORNEY JR. propôs uma técnica de codificação de canal implementada através da combinação de códigos componentes. Com essa técnica era possível obter códigos resultantes de maior comprimento e com maior capacidade de correção de erros que aqueles proporcionados pelos códigos componentes individualmente. Ainda assim, permitia-se que o processo de decodificação, dividido em etapas associadas a cada código componente, se tornasse menos complexo que aquele que seria necessário para decodificar um único código equivalente de comprimento igual ao do código resultante. Ao código resultante dessa implementação foi dado o nome de código concatenado [1].

Em 1989, J. Hagenauer e P. Hoeher propuserem uma variante do algoritmo de decodificação de Viterbi. Com ela eram produzidas decisões suaves e através destas uma decisão abrupta poderia ser tomada. Ou então, e principalmente, em um esquema com códigos concatenados uma métrica associada à confiabilidade dessa decisão poderia ser utilizada como entrada suave do processo de decodificação conseguinte. A essa variante do algoritmo de Viterbi foi dado o nome de SOVA (*Soft-Output Viterbi Algorithm*) [4].

Alguns anos depois, em 1993, os *códigos turbo* foram inventados por C. Berrou, A. Glavieux e P. Thitimajshima [2] na *Ecole Nationale Supérieuri dês Télécommunications de Bretagne*, França, tendo como forte motivação as idéias de [4] e incorporando a essência da construção proposta em [1]. Este novo esquema de codificação de canal corresponde à concatenação paralela de códigos convolucionais recursivos e sistemáticos, decodificados iterativamente por um algoritmo baseado no algoritmo MAP (*maximum a posteriori*) símbolo-a-símbolo BCJR[1] [3].

Embora a invenção de 1993 se referisse apenas àquela forma de codificação e decodificação, de forma genérica, atualmente, pode-se classificar como *código turbo* todo esquema de codificação de canal que utilize: 1) processos de decodificação iterativa e 2) a concatenação de códigos componentes separados por entrelaçadores temporais. Assim, percebe-se que o termo "turbo" está diretamente associado à decodificação iterativa e não necessariamente à forma de implementação da codificação, embora os códigos componentes utilizados em [2] tenham sido também propostos pela primeira vez naquela publicação. Conforme afirmam Claude Berrou e Alain Glavieux [7]:

> "*Os códigos turbo são resultado de uma construção empírica e trabalhosa de um esquema completo de codificação e decodificação, utilizando blocos existentes que nunca haviam sido colocados juntos daquela forma antes*".

Os códigos turbo utilizam algoritmos do tipo SISO (*Soft-Input*, *Soft-Output*) no processo de decodificação iterativa. Nestes algoritmos, informações sobre a confiabilidade ou qualidade da decodificação de um dos códigos componentes (*soft output*) alimentam o processo de decodificação de outro código componente, na forma de entrada suave (*soft input*). Dessa forma, a cada iteração tem-se maior confiabilidade na estimação do bit, palavra ou seqüência transmitida, dependendo da forma específica de implementação da decodificação. A obtenção dessa confiabilidade, denominada de informação extrínseca em [2], incorpora um princípio já identificado por Gallager em [5] e, em paralelo com as investigações de C. Berrou *et al*.,

[1] A sigla BCJR foi adotada devido aos sobrenomes dos inventores do algoritmo: L. R. BAHL; J. COCKE; F. JELINEK e J. RAVIV.

por Lodge *et al*. em [6]. Entretanto, todos estes trabalhos ocorreram de forma independente [7].

Hoje, pouco mais de dez anos após a invenção dos códigos turbo, as pesquisas sobre o tema se encontram em um estágio significativamente avançado e ramificaram-se, dando surgimento ao chamado *processamento turbo* no qual, de forma genérica, para a realização de um determinado processo há troca de informação entre sub-processos componentes que cooperam entre si de forma iterativa. Dentre as várias técnicas nas quais o processamento turbo pode ser aplicado destacam-se a *equalização*, a *estimação de canal*, a *codificação de fonte e canal conjunta*, a *detecção multi-usuário* e o *cancelamento de interferências*, os *sistemas MIMO* (*Multiple Input*, *Multiple Output*) e a *codificação espaço-temporal*, apenas para citar alguns exemplos.

O processamento turbo está sendo interpretado como uma das mais promissoras técnicas para a melhoria de desempenho em sistemas de comunicação. Por essa razão, a invenção dos códigos turbo está sendo considerada como o segundo grande marco do desenvolvimento científico das comunicações, desde o desenvolvimento da teoria matemática da comunicação [18] no final da década de 40. Qualquer sistema com realimentação, conforme desafiou Simon Haykin na apresentação de [19], a partir de agora deve ser interpretado não simplesmente como um sistema onde há realimentação de sinais, mas sim onde há *realimentação de informação*, como acontece no processamento turbo. De fato, durante as investigações que culminaram na invenção dos códigos turbo, C. Berrou *et al*. foram inspirados por uma idéia de amplificação de informação a partir de uma estrutura com realimentação [7]. Nesta estrutura pode-se interpretar como condição de convergência e estabilidade do processamento turbo a adequada configuração do "ponto de operação" desse amplificador.

As demais seções deste tutorial estão organizadas da seguinte maneira: na Seção II são apresentados alguns conceitos iniciais genéricos sobre os códigos turbo. Na Seção III faz-se uma revisão sobre os códigos produto de paridade simples, com destaque para as regras de implementação de um código multidimensional. A Seção IV aborda o processo de decodificação turbo de códigos produto multidimensionais e a Seção V trata especificamente da decodificação turbo de códigos produto com componentes de paridade simples. Na Seção VI é fornecido um exemplo de decodificação turbo de um código produto bidimensional com componentes de paridade simples. Na Seção VII são tecidos alguns comentários sobre a influência do aumento de dimensões e do processo de entrelaçamento temporal no desempenho de códigos produto de paridade simples. A Seção VIII apresenta e tece comentários sobre alguns resultados de desempenho de códigos produto de paridade simples nos canais AWGN e com desvanecimento Rayleigh plano. Por fim, no Apêndice trata-se de uma rotina, elaborada na plataforma Mathcad 2001*i*, para codificação e decodificação turbo dos códigos em questão.

Muitas das conclusões e interpretações aqui registradas devem parecer, num primeiro momento, não corresponder a um estudo introdutório simplificado sobre códigos turbo. E de fato não correspondem! Optou-se por elaborar um tutorial mais abrangente e menos básico, evitando-se repetir a estrutura de excelentes trabalhos básicos já publicados, mas complementando-os de certa forma. Por esta razão, recomenda-se que o estudo deste texto seja realizado em etapas: primeiramente deve-se buscar entender os conceitos básicos associados aos princípios da decodificação turbo, por exemplo utilizando as referências [10] e [22], principalmente os exemplos lá fornecidos. Posteriormente recomenda-se um aprofundamento gradativo deste tutorial, consultado as demais referências citadas ao seu final, sempre que necessário. A rotina computacional citada no Apêndice pode ser de particular utilidade para auxiliar o aprendizado.

II - CONCEITOS INICIAIS

BASICAMENTE TÊM-SE duas famílias de códigos turbo: uma baseada na concatenação de códigos convolucionais (CTC, *Convolutional Turbo Codes*) [2][11] e outra baseada na concatenação de códigos de bloco (BTC, *Block Turbo Codes*) [12]. Quando da invenção dos códigos turbo, C. Berrou *et al*. trataram dos códigos turbo convolucionais [2]. Mais recentemente, grande interesse tem sido demonstrado por implementações de processos de decodificação iterativa baseados em códigos de bloco [8][10][12]-[16]. Os principais objetivos dessas implementações se referem à possibilidade de redução na complexidade e aumento na velocidade de decodificação e também à possibilidade de aumento de desempenho em relação aos códigos turbo convolucionais, para códigos de taxas altas [10][12][14] e blocos relativamente curtos. Para taxas baixas, os CTCs tendem a apresentar melhor desempenho. Outra vantagem aparente dos códigos turbo de bloco se refere à possibilidade de redução do valor do "patamar de saturação da taxa de erro de bit" ou "joelho" (do Inglês *error floor*) que é percebido em curvas de taxa de erro de bit *versus* relação sinal-ruído média por bit, para esses códigos. Esse joelho corresponde a um comportamento similar a uma saturação na taxa de erro de bit, fazendo com que esta seja reduzida apenas de forma marginal, mesmo com significativos aumentos na relação E_b/N_0. O fenômeno ocorre principalmente devido à presença de um número significativo de palavras-código de baixo peso, posto que em valores mais altos de E_b/N_0 a taxa de erro de bit passa a ser governada predominantemente por estas palavras-código [15]. O joelho pode ser reduzido através de melhorias nos processos de entrelaçamento temporal entre os códigos componentes [17].

Destaca-se uma interpretação sobre o processo de entrelaçamento temporal supracitado: este processo, localizado entre códigos concatenados, tem um propósito distinto daquele entrelaçamento que tipicamente é realizado após a codificação de canal. Neste último caso objetiva-se a

transformação de um comportamento de memória[2] do canal num comportamento sem memória, do ponto de vista do decodificador de canal localizado no receptor. No que diz respeito ao entrelaçamento temporal realizado entre os códigos componentes da concatenação, o principal objetivo é fazer com que as informações trocadas no processo de decodificação iterativa sobre um determinado símbolo ou palavra-código sejam tão descorrelacionadas quanto possível, o que trará, iteração a iteração, melhorias mais significativas no desempenho do código turbo. De fato, há grande influência da forma com que é realizado o entrelaçamento temporal no desempenho do processo de decodificação iterativa, especificamente, e no desempenho do código turbo, genericamente.

Surpreendentes resultados de desempenho de códigos turbo foram reportados na literatura: em [2][3], publicação que marca a invenção desses códigos, a 10^{-5} de taxa de erro de bit pôde-se operar a uma E_b/N_0 distante apenas 0,5 dB do limite de Shannon em canal AWGN, para modulação BPSK ($\cong$ 0,19 dB). Desempenhos resultando na operação a uma E_b/N_0 distante 0,2 dB e 0,35 dB da capacidade do canal AWGN foram reportados em [8] e [7], respectivamente, também para modulação BPSK. Mais recentemente, potentes códigos Turbo e LDPC (*Low Density Parity-Check*) [23][24] chegam a operar a menos de 0,1 dB do limite de Shannon.

Apesar dos ganhos de codificação significativos que podem ser obtidos com os códigos turbo, um dos grandes obstáculos a ser ainda transposto se refere à simplificação dos algoritmos de decodificação: aqueles considerados ótimos segundo o critério MAP símbolo-a-símbolo, via de regra são complexos, demandando altas velocidades de processamento para que certas aplicações onde há comunicação em tempo real, por exemplo, possam ser viabilizadas. Várias pesquisas têm sido encaminhadas no sentido de desenvolver novas formas de algoritmos ótimos ou de desenvolver algoritmos sub-ótimos, ou ainda no sentido de modificar algoritmos ótimos, tornando-os sub-ótimos, porém com menor grau de complexidade.

Os códigos turbo permitem ainda que, a um moderado grau de complexidade, seja possível operar com valores de E_b/N_0 abaixo da *taxa de corte* do canal [9]. Até então se sabia que esta era uma tarefa possível, mas de alta complexidade de realização [10], pois a taxa de corte era considerada como a "capacidade prática" do canal.

III - CÓDIGOS PRODUTO DE PARIDADE SIMPLES

NESTE TEXTO foram adotados como foco de estudo os códigos turbo implementados com códigos de bloco, tendo por certo que essa escolha é suficiente à apresentação dos principais fundamentos sobre a decodificação turbo. Dentre as várias configurações que podem ser obtidas combinando-se diferentes códigos componentes de diferentes taxas, diferentes regras de entrelaçamento temporal e diferentes esquemas de decodificação iterativa, a família dos códigos produto [20] foi escolhida. Dentre os códigos produto, abordagens teóricas e exemplos são aqui apresentados para os códigos produto de paridade simples (SPC-PC, *Single-Parity Check Product Codes*), para os quais a decodificação dos códigos componentes é efetuada pelo algoritmo MAP símbolo-a-símbolo proposto em [15].

Os códigos produto podem ser interpretados como pertencentes à família dos códigos de arranjo [21], nos quais o processo de codificação pode ser associado a uma construção geométrica composta por um arranjo de códigos componentes. Além de grande simplicidade de implementação, os códigos produto apresentam ampla flexibilidade em termos de adequação da taxa do código resultante e do comprimento do bloco.

Um código produto de dimensão D pode ser definido a partir do comprimento do bloco de informação em cada dimensão $\{k_1, k_2, ..., k_i, ...k_D\}$. Podem ser utilizados códigos componentes sistemáticos ou não-sistemáticos (n_i, k_i, $d_{\min i}$). O código produto resultante possui palavras-código de comprimento

$$\nu = \prod_{i=1}^{D} n_i \tag{1}$$

Se $r_i = k_i/n_i$ é a taxa do código componente na i-ésima dimensão, a taxa do código produto será

$$r = \prod_{i=1}^{D} r_i \tag{2}$$

e a distância mínima será

$$\delta_{\min} = \prod_{i=1}^{D} d_{\min_i} \tag{3}$$

Se os códigos componentes são de paridade simples e idênticos em todas as dimensões, ou seja, códigos caracterizados por $(n, k, d_{\min}) = (n, n-1, 2)$, o código produto $(\nu, \kappa, \delta_{\min})$, denotado por $(n, k, d_{\min})^D$, terá então:

$$\begin{aligned} \nu &= n^D \\ \kappa &= k^D = (n-1)^D \\ \delta_{\min} &= 2^D \end{aligned} \tag{4}$$

A Figura 1 mostra a estrutura geométrica de um código produto bidimensional (2D). Além dos bits de paridade gerados pelo processo de codificação em cada dimensão, o código produto possui os bits de paridade gerados a partir da paridade das paridades (*check on*

[2] Um exemplo típico de canal com memória é o canal rádio-móvel, onde podem ocorrer erros de símbolo em rajada. O exemplo clássico de canal sem memória é o canal AWGN, no qual os erros são descorrelacionados, permitindo melhor desempenho do processo de correção de erros realizado pelo esquema de codificação de canal.

[3] O resultado de simulação apresentado em [2] foi obtido com o uso de codificadores convolucionais recursivos e sistemáticos, um extenso bloco de entrelaçamento temporal pseudo-aleatório entre os codificadores (65.536 bits), 18 iterações e alguns ajustes no algoritmo BCJR.

checks). Neste caso pode-se verificar que se trata da concatenação serial dos códigos componentes.

Certos autores ainda definem uma outra categoria de códigos produto: o código produto incompleto [10][22], correspondente à concatenação paralela dos códigos componentes. Um código produto incompleto não possui a paridade das paridades e, apesar de ter implementação ligeiramente mais simples, possui desempenho inferior, assintoticamente. Isto se deve às principais razões: a distância mínima do código incompleto é menor que a do código completo [15] e o número de palavras-código de mais baixo peso é maior no código incompleto [25].

As propriedades descritas pelas expressões (1) - (4) não se aplicam aos códigos incompletos. Sendo assim, deste ponto em diante no texto o termo *código produto* refere-se ao *código completo*, a menos que esteja explícito o contrário.

A concatenação serial presente na estrutura *D*-dimensional de um código produto permite que sejam também utilizados códigos componentes não-sistemáticos [26]. Tanto a taxa de codificação final quanto a distância mínima de Hamming e também o comprimento do bloco do código serão iguais àqueles obtidos com códigos componentes na sua forma sistemática.

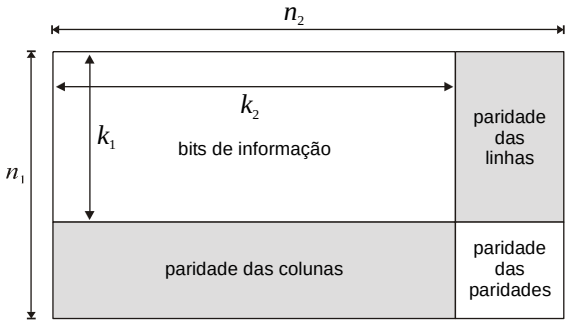

***Figura 1** - Estrutura do código produto bidimensional completo.*

Cogita-se que a concatenação paralela de códigos de bloco seja mais adequada naquelas situações nas quais se deseja a operação do sistema codificado a baixos valores de E_b/N_0, porém podendo suportar relativamente altos valores de taxa de erro de bit (BER, *Bit Error Rate*), acima do "joelho" da curva BER *versus* E_b/N_0. É aparente que, na concatenação serial de códigos de bloco, têm-se maiores valores de E_b/N_0 a altos valores de taxa de erro de bit, mas o decréscimo da BER com o aumento de E_b/N_0 é consideravelmente mais abrupto, possibilitando transmissões praticamente "livres de erro" a valores de E_b/N_0 acima de um determinado limiar. Em [27, p. 925] e [28, p. 623] têm-se exemplos deste comportamento para códigos convolucionais concatenados. Uma comparação entre resultados apresentados em [15], [25] e [26, Capítulo 3] permite a identificação de exemplos similares para códigos de bloco concatenados. A Figura 2 ilustra o comportamento descrito neste parágrafo.

Quanto aos processos de entrelaçamento temporal utilizados na formação do código produto, podem-se utilizar simples arranjos retangulares em bloco. O vetor contendo os bits a serem entrelaçados alimenta o bloco de entrelaçamento pelas linhas e o vetor entrelaçado é lido pelas colunas, alimentando o codificador seguinte *k* a *k* bits. O número de linhas e o número de colunas do bloco de entrelaçamento entre as dimensões (códigos concatenados) *i* e *i* + 1, números esses governados pela própria estrutura geométrica dos códigos produto, são dados por [26, p. 114]

$$(N_l \times N_c)^{i,i+1} = (n^i k^{D-i-1} \times n) \tag{5}$$

onde *n* é o comprimento da palavra-código do código componente, *k* é o comprimento da palavra de informação do código componente, *D* é o número de dimensões do código produto (número de códigos concatenados) e $i = 0, 1, \ldots D-1$ é o índice de cada dimensão. A Figura 3 ilustra a seqüência de codificação de um código produto tridimensional (3D) com códigos componentes de paridade simples (3,2,2), denotado por $(3,2,2)^3$. Como os códigos de paridade simples são sistemáticos, o cubo contendo os k^3 bits de informação, após a formação do código produto 3D, fica localizado na parte da frente, superior e esquerda do arranjo de n^3 bits, com os bits na mesma disposição daqueles do arranjo inicial de k^3 bits, conforme ilustra a Figura 4.

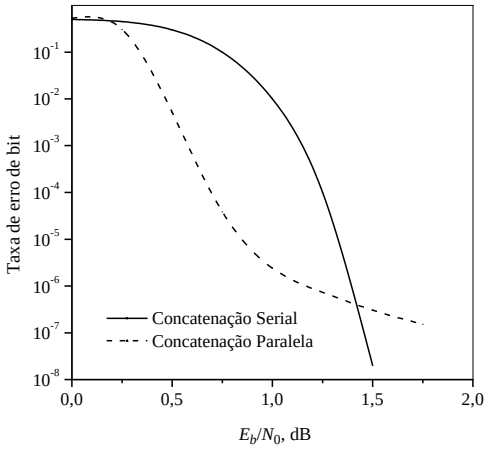

***Figura 2** - Comportamento típico do desempenho de códigos de bloco concatenados em série e em paralelo.*

Tomando novamente como referência o exemplo ilustrado pela Figura 3, as etapas de formação do arranjo correspondente a uma palavra-código do código produto de paridade simples 3D com componentes (3,2,2) são apresentadas no Algoritmo 1.

Uma propriedade relevante dos códigos produto, principalmente do ponto de vista da decodificação, corresponde ao fato de que *todas as palavras em um determinado sentido da estrutura de dimensão D são palavras-código dos códigos componentes* [26]. Por exemplo, na estrutura 3D mostrada na Figura 3, o bloco de $n^D = 27$ bits é formado por $n^{D-1} = 9$ palavras-código de $n = 3$ bits dispostas no sentido de suas três faces, totalizando $Dn^{D-1} = 27$ palavras-código do código componente (3,2,2). Objetivando melhor ilustrar este processo, admita que **R** corresponda ao arranjo tridimensional de bits recebidos,

conforme mostra a Figura 5. Em todos os sentidos indicados pelas setas da Figura 5 têm-se $n^{D-1} = 9$ palavras-código de n bits dos códigos componentes. Mais adiante essa propriedade será explorada no contexto da decodificação turbo de códigos produto.

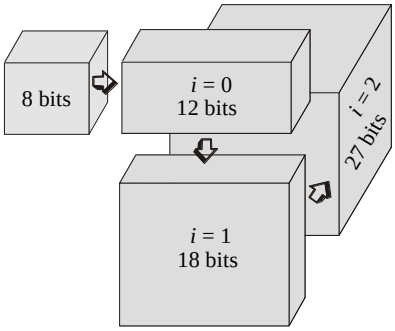

Figura 3 - *Estrutura do código produto 3D, para códigos componentes (3,2,2).*

IV - DECODIFICAÇÃO TURBO DE CÓDIGOS PRODUTO

PARA A CONCATENAÇÃO SERIAL de dois ou mais códigos de bloco separados por entrelaçadores temporais, formando a estrutura de um código produto, o processo de decodificação iterativa pode valer-se de algoritmos derivados de algoritmos de decodificação de códigos produto, desde que estes algoritmos possam operar com estradas suaves e forneçam decisões suaves como saída. A Figura 6 apresenta o diagrama de um decodificador com entrada e saída suaves (SISO) usado na composição de um decodificador turbo, para códigos produto com componentes sistemáticos ou não-sistemáticos.

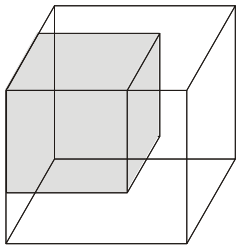

Figura 4 - *Disposição dos bits de informação (bloco com preenchimento sólido) na estrutura do código produto 3D, para códigos componentes sistemáticos quaisquer.*

A *entrada suave* destacada na Figura 6 está diretamente associada à saída do detector no receptor. Esta saída contém a chamada *informação de estado de canal*[4] obtida para todos os bits codificados, mais a informação *a priori* somente dos bits de informação (para códigos componentes sistemáticos) ou para todos os bits codificados (para códigos componentes sistemáticos ou não-sistemáticos). Tais valores são operados tipicamente no domínio logarítmico. As variáveis apresentadas na Figura 6 serão definidas e utilizadas mais adiante.

Algoritmo 1 *- Formação de códigos produto de paridade simples: exemplo para o código (3,2,2)*[3].

1. Forme o vetor de entrada composto de $k^D = 2^3 = 8$ bits de informação;
2. na dimensão $i = 0$, codifique k a $k = 2$ a 2, os $k^D = 2^3 = 8$ bits de entrada, gerando um vetor com $n^{i+1}k^{D-i-1} = 3^{0+1}2^{3-0-1} = 12$ bits;
3. faça o entrelaçamento temporal nesse vetor de 12 bits, entre as dimensões 0 e 1, alimentando pelas linhas um arranjo do tipo linha-coluna com $(N_l \times N_c)^{i,\, i+1} = (n^i k^{D-i-1} \times n) = (3^0 2^{3-0-1} \times 3) = (4 \times 3)$ bits; a leitura dos bits entrelaçados é feita pelas colunas desse arranjo;
4. na dimensão $i = 1$, codifique, 2 a 2, os 12 bits anteriormente entrelaçados, gerando um vetor com $n^{i+1}k^{D-i-1} = 3^{1+1}2^{3-1-1} = 18$ bits;
5. faça o entrelaçamento temporal no vetor de 18 bits, entre as dimensões 1 e 2, alimentando pelas linhas um arranjo do tipo linha-coluna com $(N_l \times N_c)^{i,\, i+1} = (n^i k^{D-i-1} \times n) = (3^1 2^{3-1-1} \times 3) = (6 \times 3)$ bits; a leitura dos bits entrelaçados é feita pelas colunas desse arranjo;
6. na dimensão $i = 2$, finalmente codifique, 2 a 2, os 18 bits anteriormente entrelaçados, gerando um vetor com $n^{i+1}k^{D-i-1} = 3^{2+1}2^{3-2-1} = 27$ bits.

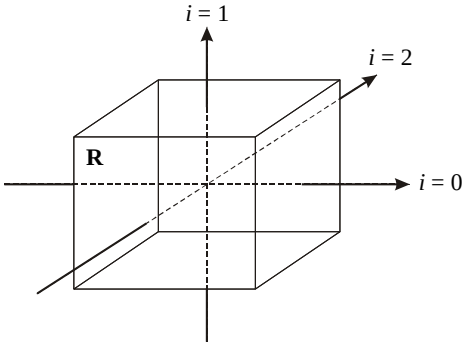

Figura 5 - *Ilustração dos sentidos de decodificação elementar em cada dimensão de um código produto 3D.*

De forma a melhor ilustrar a composição da entrada suave mostrada na Figura 6, admita um canal de comunicação com desvanecimento plano do tipo Rayleigh. Após o sinal transmitido passar por este canal pode-se determinar a entrada suave do decodificador SISO através da *razão de verossimilhança* - que em escala logarítmica pode ser chamada de *razão de verossimilhança logarítmica* [10, p. 430] [28, p. 32] do símbolo $d \in \{\pm\sqrt{E}\}$, condicionada à saída x do filtro casado de recepção e à amplitude g do desvanecimento (*ganho do canal*):

[4] No presente contexto, por *informação de estado de canal* entende-se uma métrica associada ao valor real obtido na saída de um filtro casado ou correlator (ou banco de filtros casados ou de correlatores), no momento ótimo que seria utilizado para decisão abrupta sobre o correspondente símbolo recebido.

$$L(d \mid x,g) = \ln\left[\frac{\Pr(d=+\sqrt{E} \mid x,g)}{\Pr(d=-\sqrt{E} \mid x,g)}\right] \quad (6)$$

Pelo *Teorema de Bayes* pode-se re-escrever (6) como:

$$L(d \mid x,g) = \ln\left[\frac{p(x,g \mid d=+\sqrt{E})}{p(x,g \mid d=-\sqrt{E})}\frac{\Pr(d=+\sqrt{E})}{\Pr(d=-\sqrt{E})}\right] \quad (7)$$

Como o canal com desvanecimento Rayleigh plano pode ser considerado como um canal condicionalmente gaussiano (condicionado à magnitude do desvanecimento), tem-se:

$$L(d \mid x,g) = \ln\left\{\frac{\exp\left[-\frac{\left(x-g\sqrt{E}\right)^2}{N_0}\right]}{\exp\left[-\frac{\left(x+g\sqrt{E}\right)^2}{N_0}\right]}\right\} + \ln\left[\frac{\Pr(d=+\sqrt{E})}{\Pr(d=-\sqrt{E})}\right] \quad (8)$$

$$= L_c x + L(d) = 4g\frac{\sqrt{E}}{N_0}x + L(d) = 2g\frac{\sqrt{E}}{\sigma^2}x + L(d)$$

onde $\sigma^2 = N_0/2$ é a variância da componente de ruído gaussiano na variável de decisão, $N_0/2$ é densidade espectral de potência bilateral desse ruído e a variável $L_c = 4g\sqrt{E}/N_0$ pode ser interpretada como a *confiabilidade do símbolo* [10, p. 430] [28, p. 62]. Ainda com relação à expressão (8), E é a energia média por símbolo e $L(d)$ corresponde aos valores *a priori* do símbolo d. Para o canal sem desvanecimento (AWGN puro) basta fazer $g = 1$ em (8).

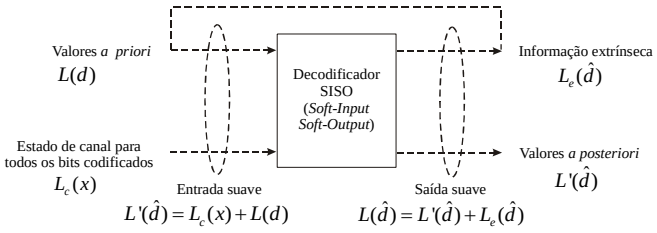

***Figura 6** - Ilustração do processo de decodificação turbo (adaptado de [10] e [22])*

No processo de decodificação iterativa, os valores *a priori* não são conhecidos antes da primeira iteração completa e normalmente são considerados nulos nessa etapa (probabilidades *a priori* iguais a 1/2). A *saída suave* do decodificador SISO é composta pelos valores *a posteriori* dos bits de informação (para códigos componentes sistemáticos) ou de todos os bits codificados (para códigos componentes sistemáticos ou não-sistemáticos), mais a *informação extrínseca* obtida pelo processo de decodificação. Esse valor de informação extrínseca é realimentado à entrada do decodificador SISO como o valor da verossimilhança logarítmica *a priori* para a próxima iteração. Este procedimento faz com que os novos valores das métricas de entrada possam produzir valores de métricas de saída mais confiáveis, iteração a iteração.

A informação extrínseca pode ser interpretada como a quantidade de informação adicionada à entrada suave do decodificador para formar o valor correspondente à saída suave [30]. Ela pode também ser interpretada como a quantidade de informação obtida pelo processo de decodificação [22], porém independente dos valores antes deste processo (na saída do detector) [11]. A informação extrínseca pode ainda ser definida como a diferença entre a métrica calculada na saída do estágio de decodificação (saída suave) e a informação intrínseca representada por uma métrica realimentada à entrada do estágio de decodificação (entrada suave); *é a informação adicional obtida através da exploração das dependências que existem entre os bits de informação e os bits codificados* [31], *segundo cada regra específica de codificação*.

A razão de verossimilhança logarítmica para um determinado símbolo na entrada do decodificador SISO, deste ponto em diante representada por LLR (*Log-Likelihood Ratio*), pode ser expressa de forma simplificada por

$$L(d \mid x,g) = L(x \mid d,g) + L(d) \quad (9)$$

A métrica $L(x \mid d,g)$ é a LLR das medidas do canal x, dadas as condições de transmissão de um 0 ou de um 1 ($d = \pm\sqrt{E}$) e dado o ganho do canal g. $L(d)$ é a LLR *a priori* do símbolo d. Para o canal AWGN pode-se simplificar ainda mais a notação usada em (9), tal que se obtenha [22]:

$$L'(\hat{d}) = L_c(x) + L(d) \quad (10)$$

onde $L'(\hat{d})$ é a LLR estimada de um determinado símbolo d, apresentada à entrada do decodificador SISO, e $L_c(x)$ é a LLR da medida do estado de canal, correspondente ao símbolo d e obtida a partir da saída do detector.

Pode-se mostrar [11] que a LLR na saída do decodificador SISO (saída suave) pode ser expressa por:

$$L(\hat{d}) = L'(\hat{d}) + L_e(\hat{d}) \quad (11)$$

onde $L'(\hat{d})$ é a LLR de um símbolo (valor *a posteriori*) e $L_e(\hat{d})$ é definida como a LLR extrínseca ou *informação extrínseca* desse símbolo.

Combinando (10) e (11) pode-se escrever a LLR obtida através do processo de decodificação SISO como:

$$L(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d}) \quad (12)$$

Esta LLR contém a informação necessária para decodificação abrupta do símbolo d, dada pela polaridade de $L(\hat{d})$, bem como a informação de confiabilidade dessa decisão, dada pela magnitude de $L(\hat{d})$. Relembrando, $L_c(x)$ representa a LLR obtida a partir do valor de saída do

detector, $L(d)$ é a LLR *a priori* do símbolo d e $L_e(\hat{d})$ é a informação extrínseca gerada pelo, e dependente do, processo de decodificação.

Na Figura 6 apresentou-se o diagrama de um decodificador SISO, onde agora pode-se observar como as variáveis supramencionadas são operadas. A saída do detector provê a LLR que contém a informação de canal obtida para todos os símbolos codificados, mais a informação *a priori* para: 1) os símbolos de informação [10] [22] ou 2) todos os símbolos codificados [15][26]. Como mencionado anteriormente, no processo de decodificação iterativa estes valores *a priori* não são conhecidos antes da primeira iteração completa e normalmente são considerados nulos (probabilidades *a priori* iguais).

A saída do decodificador SISO é composta pelos valores *a posteriori* dos símbolos de informação, mais a informação extrínseca obtida pelo processo de decodificação. Esse valor de informação extrínseca é realimentado à entrada do decodificador SISO como o valor da LLR *a priori* do símbolo em questão para a próxima iteração. Este procedimento faz com que os novos valores de LLR de entrada possam produzir valores de LLR de saída mais confiáveis, iteração a iteração.

Como mostrado mais adiante com o auxílio de um exemplo, no processo de decodificação turbo de códigos produto de paridade simples a magnitude da informação extrínseca de um símbolo em particular é *aproximadamente* igual à menor das magnitudes das LLR's dos outros símbolos em uma palavra-código, e o sinal dessa informação extrínseca é igual ao próprio sinal dessa LLR, se a decisão abrupta levar a uma palavra-código válida (se for verificada a paridade), e tem seu sinal invertido se esta paridade não é verificada. Esta operação faz com que, a cada iteração, os valores das LLR's de saída do decodificador SISO sejam reduzidos ou fortalecidos, dependendo da verificação ou não da paridade. A quantidade de redução ou fortalecimento dependerá do menor valor de LLR dos demais bits envolvidos [30][25]. Esse menor valor pode ser interpretado como sendo a menor confiabilidade obtida no resultado de verificação da paridade.

Uma outra observação a ser mencionada se refere à convergência do processo iterativo na decodificação de códigos produto de paridade simples. Com um número de iterações igual à dimensão do código, praticamente tem-se LLR's de saída já adequadas à decisão abrupta sobre os símbolos transmitidos. A partir daí percebe-se, a cada iteração, um "reforço" nas LLR's de saída do decodificador SISO, numa tendência média de confirmar a decisão que já poderia ser tomada logo após a iteração de número D. Nenhuma melhoria estaria sendo obtida em termos da decisão final a partir desse ponto. Esta observação pode ser constatada no exemplo de decodificação turbo apresentado na Seção VI.

Entretanto, há alguma chance da convergência acontecer antes do final das D iterações. Este fato indica a necessidade de se definir um critério de parada no processo iterativo de tal sorte que seja reduzido o tempo total de decodificação e se possa aumentar a vazão média de dados em um sistema de comunicação real. Em [10] são abordados alguns desses critérios e em [30, p. 43, item 4.7] é sugerido um método bastante simples de verificação de convergência no processo iterativo e de interrupção desse processo caso a convergência seja verificada. A idéia em [30] é adequada à decodificação iterativa de códigos produto com paridade simples.

Vários algoritmos utilizados para decodificação turbo de códigos de bloco podem ser encontrados na literatura, e também outros algoritmos não específicos para decodificação turbo, mas que podem ser utilizados como base na implementação da decodificação turbo. Esta útima opção é possível somente se tais algoritmos operarem com entradas suaves e permitirem que decisões suaves possam ser obtidas em suas saídas. Do rol de algoritmos propostos até o momento, praticamente todos são baseados em variações e/ou derivações dos algoritmos Chase [32], BCJR [3], Kaneko [33] ou SOVA [4]. Dentre os mais citados como referência tem-se o algoritmo de Pyndiah [12] e os algoritmos sub-ótimos classificados como Log-MAP [34].

V - DECODIFICAÇÃO TURBO DE CÓDIGOS PRODUTO DE PARIDADE SIMPLES

OS ALGORITMOS DE DECODIFICAÇÃO turbo ótimos e sub-ótimos, no domínio logarítmico das verossimilhanças, são conhecidos como algoritmos Log-MAP [34]. Em [10] e [22], uma versão sub-ótima de um algoritmo Log-MAP é utilizada em exemplos simples de decodificação turbo de códigos de paridade simples concatenados em paralelo (aqui denominados códigos produto incompletos). Na seção seguinte, o algoritmo Log-MAP apresentado em [15][5] é utilizado num exemplo de decodificação turbo de um código produto 2D completo, na decodificação dos códigos componentes de paridade simples. Tal algoritmo foi implementado em Mathcad e a rotina resultante pode ser acessada via *web*, conforme orientações constantes do Apêndice.

Como citado, a razão de verossimilhança logarítmica, LLR, obtida por meio do processo de decodificação SISO (saída suave) pode ser expressa pela equação (12), repetida aqui por conveniência:

$$L(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d}) \qquad (13)$$

onde, recordando: $L_c(x)$ representa a LLR obtida a partir da saída do detector; $L(d)$ é a LLR *a priori* do símbolo d e $L_e(\hat{d})$ é a informação extrínseca gerada em função da estrutura do código, pelo processo de decodificação SISO.

Como ilustração desse processo de decodificação, seja um código produto de paridade simples bidimensional. O algoritmo de decodificação opera de acordo com a seqüência mostrada no Algoritmo 2. Essa seqüência pode

[5] Embora o algoritmo de [15] seja ótimo do ponto de vista da decodificação dos códigos componentes do código produto, a decodificação turbo do código produto resultante é sub-ótima. Entretanto, para valores elevados de relação sinal-ruído, o desempenho tende a ser muito próximo da decodificação ótima [15, p. 65].

facilmente ser identificada na rotina mencionada no Apêndice e também no exemplo de decodificação apresentado na Seção VI. Ressalta-se que, para um código de dimensão *D*, a atualização da informação *a priori* a cada dimensão é feita através da soma das informações extrínsecas calculadas para as *outras* dimensões até aquele momento [15].

Para realizar os cálculos do processo iterativo de decodificação, há que se utilizar um conjunto de ferramentas apropriadas ao problema. A esse conjunto foi dado o nome de álgebra no domínio logarítmico das verossimilhanças (do Inglês *log-likelihood algebra*) [11][10][22]. Os principais resultados dessa álgebra são resumidamente apresentados em seguida.

Pode-se mostrar que, para variáveis estatisticamente independentes e denotadas por d_j, a razão de verossimilhança logarítmica obtida da operação de soma (módulo 2) entre dois bits é definida por [10][22][6]

$$\begin{aligned} L(d_1) \boxplus L(d_2) &= L(d_1 \oplus d_2) = \ln\left[\frac{e^{L(d_1)} + e^{L(d_2)}}{1 + e^{L(d_1)} e^{L(d_2)}}\right] \\ &\approx \operatorname{sign}\left[L(d_1)\right] \times \operatorname{sign}\left[L(d_2)\right] \times \min\left[|L(d_1)|, |L(d_2)|\right] \end{aligned} \tag{14}$$

com as regras adicionais:

$$\begin{aligned} &L(d) \boxplus \infty = L(d), \\ &L(d) \boxplus -\infty = -L(d) \text{ e} \\ &L(d) \boxplus 0 = 0 \end{aligned} \tag{15}$$

A expressão (14) pode ser generalizada, levando a [10]

$$\begin{aligned} \sum_{j=1}^{J} \boxplus L(d_j) &= \ln\left[\frac{1 + \prod_{j=1}^{J} \tanh\left(L(d_j)/2\right)}{1 - \prod_{j=1}^{J} \tanh\left(L(d_j)/2\right)}\right] \\ &= 2\operatorname{arctanh}\left(\prod_{j=1}^{J} \tanh\left(L(d_j)/2\right)\right) \end{aligned} \tag{16}$$

Esta expressão também pode sofrer uma aproximação, assim como em (14), levando a

$$\begin{aligned} \sum_{j=1}^{J} \boxplus L(d_j) &= L\left(\sum_{j=1}^{J} \oplus d_j\right) \\ &\approx \left(\prod_{j=1}^{J} \operatorname{sign}[L(d_j)]\right) \min_{j=1\ldots J} \left|L(d_j)\right| \end{aligned} \tag{17}$$

O valor de $L_c(x)$, em canal AWGN, conforme (8), pode ser determinado por

$$L_c(x) = 4\frac{\sqrt{E}}{N_0} x \tag{18}$$

onde, recordando, $N_0/2$ é densidade espectral de potência bilateral do ruído gaussiano presente nas amostras da saída do filtro casado do receptor no instante de decisão, $\sqrt{E}$ é o valor médio dessas amostras e x é o valor real da amostra obtida.

Algoritmo 2 *- Etapas do processo de decodificação turbo de códigos produto.*

1. Faça todas as verossimilhanças logarítmicas (LLR's) *a priori* iguais a zero (probabilidades *a priori* iguais a 0,5), $L(d) = 0$;
2. conhecida a lógica de codificação, que estabelece a relação de dependência entre os bits de informação e a paridade, obtenha as informações extrínsecas na dimensão horizontal, $L_{eh}(\hat{d})$;
3. faça as novas LLR's *a priori* iguais às informações extrínsecas calculadas no passo anterior, $L(d) = L_{eh}(\hat{d})$;
4. obtenha as informações extrínsecas na dimensão vertical, $L_{ev}(\hat{d})$, levando em conta as novas LLR's *a priori* atualizadas no passo 3;
5. faça as LLR's *a priori* iguais às informações extrínsecas calculadas no passo anterior, $L(d) = L_{ev}(\hat{d})$;
6. havendo mais iterações vá ao passo 2, levando em conta os últimos valores de LLR's *a priori* atualizados. Não havendo mais iterações, vá ao passo 7;
7. a decisão suave será $L(\hat{d}) = L_c(x) + L_{eh}(d) + L_{ev}(d)$.

O agrupamento dos bits codificados de uma palavra-código de um código produto em um arranjo 2D, de forma que cada dimensão contenha um conjunto de palavras-código do código componente, é tarefa bastante simples. Tal arranjo permite facilmente a identificação da posição dos bits que formam cada palavra-código. Para um código 3D essa tarefa ainda é simples, pois mais uma vez uma construção geométrica pode ser associada à formação do arranjo final de bits codificados e, dessa forma, pode auxiliar na identificação das posições dos bits codificados que formam palavras-código dos códigos componentes. Entretanto, para $D > 3$ essa identificação não mais pode ser auxiliada por uma figura geométrica. De forma a solucionar esse problema, considere como exemplo um código $(3,2,2)^3$. Se os índices dos $n^D = 27$ bits codificados forem dispostos em um arranjo com $D = 3$ linhas e $n^D = 27$ colunas, seguindo a regra apresentada no Algoritmo 3, ter-se-á o arranjo mostrado na Figura 7. Todos os conjuntos de $n = 3$ bits consecutivos agrupados conforme os índices de cada

[6] A expressão (14) apresenta ligeiras diferenças nas publicações [10] e [22]. Se o mapeamento dos bits nos símbolos codificados for $\{0, 1\} \rightarrow \{-1, +1\}$ (elemento nulo = 0), deve ser utilizado um fator multiplicador $(-1)^{k-1}$ na expressão (14), sendo k o número de bits de informação do código componente ($k_1 = k_2 = k$). Se o mapeamento for $\{0, 1\} \rightarrow \{+1, -1\}$ (elemento nulo = 1), não é necessário o fator multiplicador.

linha desse arranjo são palavras-código dos códigos componentes. Por exemplo, tomando como referência a primeira linha da matriz mostrada na Figura 7, os bits de índice 0, 1 e 2 formam uma palavra-código do código (3,2,2), assim como os bits de índice 3, 4 e 5, e assim por diante. Tomando agora como referência a segunda linha dessa matriz, os bits de índice 0, 3 e 6 formam uma palavra-código do código (3,2,2) assim como os bits de índice 9, 12 e 15, e assim por diante. Por fim, tomando como referência a terceira linha da matriz mostrada na Figura 7, os bits de índice 0, 9 e 18 formam uma palavra-código do código componente, bem como os bits de índice 1, 10 e 19, e assim sucessivamente.

Algoritmo 3 *- Geração de arranjos similares ao mostrado na Figura 7.*

1. Crie um vetor **v** com n^D elementos tal que $v_j = j$, $j = 0, 1, ..., n^D - 1$;
2. faça i variar de 0 a $D - 1$;
3. para cada valor de i, faça w variar de 0 a $n^D - 2$;
4. calcule os elementos matriz **A** de ordem $D \times n^D$ através de: $A_{i,w} = v_{\left(n^i w\right) \bmod \left(n^D - 1\right)}$;
5. os elementos restantes valem: $A_{i,n^D-1} = n^D - 1$;
6. a matriz **A** é o arranjo que se deseja.

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 & 26 \\ 0 & 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 & 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 & 25 & 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 & 26 \\ 0 & 9 & 18 & 1 & 10 & 19 & 2 & 11 & 20 & 3 & 12 & 21 & 4 & 13 & 22 & 5 & 14 & 23 & 6 & 15 & 24 & 7 & 16 & 25 & 8 & 17 & 26 \end{pmatrix}$$

Figura 7 - *Arranjo de bits codificados utilizado no processo de decodificação de um código produto $(3,2,2)^3$.*

A regra geral de formação do arranjo a ser utilizado na decodificação de códigos produto para D, k e n quaisquer também é descrita através do Algoritmo 3, admitindo-se apenas que os códigos componentes em todas as dimensões possuem os mesmos valores de n e k. Esta última consideração, apesar de aparentemente restringir as possibilidades de escolha da taxa do código e de seu comprimento, é justificada admitindo-se que é consideravelmente mais simples implementar códigos componentes idênticos. Ainda assim tem-se grande flexibilidade na adequação de taxa e de comprimento do código, o que, caso necessário, pode sofrer ajustes adicionais através de adequadas técnicas de puncionamento (do Inglês, *puncturing*).

VI - EXEMPLO DE DECODIFICAÇÃO TURBO DE UM CÓDIGO PRODUTO SPC 2D

OBJETIVANDO PERMITIR a fixação dos conceitos sobre decodificação turbo abordados até aqui, nesta seção apresenta-se um exemplo de decodificação iterativa de um código produto bidimensional (2D) formado com componentes de paridade simples (4,3,2). O código resultante é o código produto $(n, k, d_{\min})^D = (4,3,2)^2$. Este exemplo busca complementar, de certa forma, aqueles apresentados em [10] e [22]. Nestas referências foi utilizada uma estrutura correspondente à concatenação paralela de códigos componentes de paridade simples (3,2,2), formando o que neste texto foi denominado de código produto incompleto. Aqui foi utilizada a concatenação serial dos códigos componentes (4,3,2), formando um código produto $(4,3,2)^2$ completo.

Em [10] e [22], somente aos bits de informação foram associadas a informação extrínseca e a informação a priori. Apenas a informação de estado de canal foi levada em conta para todos os bits de uma palavra-código. Neste texto, todos estes valores foram operados para todos os bits da palavra-código, assim como em [15], o que eleva significativamente o desempenho do código turbo.

Quando utilizada a álgebra no domínio logarítmico das verossimilhanças adotou-se aqui, assim como em [10], [15] e [22], a forma simplificada dada pela expressão (17), em vez da forma exata obtida através de (16). Deve-se ressaltar, entretanto, que o uso de (16) leva a resultados significativamente melhores, em termos de taxa de erro de bit final, em comparação com aqueles obtidos com o uso de (17). Esta constatação foi verificada com o auxílio de simulação e pode ser comprovada utilizando-se a rotina comentada no Apêndice. A modulação utilizada foi a BPSK, simulada através de sua equivalente antipodal em banda-base.

Admita que uma palavra correspondente a $k^D = 3^2 = 9$ bits de informação [0 0 1 0 1 0 1 0 0] seja organizada em um arranjo bidimensional. Depois de calculadas as paridades nos sentidos horizontal e vertical desse arranjo obtém-se a palavra-código do código produto $(4,3,2)^2$, conforme ilustrado pela Figura 8(*a*). Nesta figura e em outras similares que seguem, a parte mais clara corresponde ao arranjo dos bits de informação e a parte mais escura corresponde às paridades. Os índices dos bits associados a esses arranjos estão mostrados na Figura 9.

A seqüência de transmissão dos bits de uma palavra-código foi adotada como sendo: b_0, b_3, b_6, b_{v0}, b_1, b_4, b_7, b_{v1}, b_2, b_5, b_8, b_{v2}, b_{h0}, b_{h1}, b_{h2} e b_{vh}, embora possa ser diferente.

0	0	1	1
0	1	0	1
1	0	0	1
1	1	1	1

(*a*)

0,36	-0,23	1,10	-0,07
-0,03	0,04	-1,25	-1,08
-0,83	-0,66	-0,004	-0,13
0,52	0,02	-0,99	0,28

(*b*)

8,38	3,93	2,63	-6,21
5,43	-5,36	-3,77	-13,75
-11,93	0,68	5,62	-6,63
-1,70	-5,46	-13,1	-3,52

(*c*)

Figura 8 *- Arranjos referentes ao código $(4,3,2)^2$: (a) uma palavra-código; (b) amostras de ruído na saída do detector ótimo e (c) razão de verossimilhança logarítmica (LLR) de canal para todos os bits codificados.*

Na simulação realizada para compor o exemplo desta seção foi utilizado um valor de E_b/N_0 igual a 4 dB. A energia média por bit de informação, E_b, foi feita unitária e

a variância das amostras do ruído na saída do correlator ou do filtro casado, aqui discriminadas por η, foi calculada através de: $N_0/2 = E_b/(2\times E_b/N_0)$. Para o caso, $N_0/2 = 1/(2\times 10^{4/10}) \cong 0{,}199$ watts. Tais amostras podem ser visualizadas na Figura 8(*b*).

Os símbolos transmitidos têm valor $\in \{\pm\sqrt{E}\}$, onde E é a energia média por bit codificado (que é igual à energia média por símbolo transmitido), com valor $E = r\times E_b = 9/16 \cong 0{,}563$ Joules. O mapeamento utilizado na sinalização antipodal foi: bit $0 \Rightarrow +\sqrt{E}$ e bit $1 \Rightarrow -\sqrt{E}$.

Na Figura 8(*c*) têm-se os valores das LLR's obtidas a partir da saída do detector, $L_c(x)$, calculados de acordo com (18). Estes valores servirão como entrada suave do decodificador turbo na etapa de inicialização, conforme (10), posto que nesse momento faz-se as LLR's *a priori* iguais a 0. Como exemplo, o valor 8,38 na Figura 8(*c*) (correspondente ao bit b_0) foi obtido através das operações:

$$4\frac{\sqrt{E}}{N_0}x = 4\frac{\sqrt{E}}{2\sigma^2}\left(\sqrt{E}+\eta\right) = 4\frac{\sqrt{0{,}563}}{2\times 0{,}199}\left(\sqrt{0{,}563}+0{,}36\right) \cong 8{,}38.$$

Um outro exemplo pode ser dado para permitir um entendimento melhor sobre o processo: o valor −3,52 na Figura 8(*c*), correspondente ao bit de paridade b_{vh}, foi obtido através das operações:

$$4\frac{\sqrt{E}}{N_0}x = 4\frac{\sqrt{E}}{2\sigma^2}\left(-\sqrt{E}+\eta\right) = 4\frac{\sqrt{0{,}563}}{2\times 0{,}199}\left(-\sqrt{0{,}563}+0{,}28\right) \cong -3{,}52.$$

Os resultados de cálculo da informação extrínseca, obtidos da decodificação da primeira dimensão (dimensão vertical) do código produto, na primeira iteração, são mostrados na Figura 10(*a*). Estes resultados foram obtidos operando-se os valores de acordo com as dependências existentes entre os bits codificados, estas governadas pela regra de codificação de paridade simples, e utilizando-se as ferramentas de álgebra das verossimilhanças apresentadas na seção anterior.

b_0	b_1	b_2	b_{h0}
b_3	b_4	b_5	b_{h1}
b_6	b_7	b_8	b_{h2}
b_{v0}	b_{v1}	b_{v2}	b_{vh}

Figura 9 *- Possível distribuição dos bits numa palavra-código de um código produto (4,3,2)*[2].

De acordo com (12), a saída suave para um determinado bit é formada pela soma da LLR obtida a partir da saída do detector, $L_c(x)$, da informação *a priori* do correspondente bit, $L(d)$ e da informação extrínseca $L_e(\hat{d})$. Esta informação extrínseca pode ser calculada através de (16) ou (17), conforme se queira um cálculo exato ou aproximado, respectivamente. Por exemplo, na decodificação na dimensão vertical, a informação extrínseca para o bit b_0 pode ser obtida operando-se as somas das LLR's $L_c(x) + L(d)$ dos bits b_3, b_6 e b_{v0}, aqui denominadas Lb_3, Lb_6 e Lb_{v0}, respectivamente. Como ainda não há estimativa de $L(d)$ para tais bits nessa etapa, admite-se seu valor igual a zero. Então, usando a aproximação dada em (17), a informação extrínseca para o bit b_0 na dimensão vertical é calculada como segue:

$$\begin{aligned} L_{ev}(b_0) &= \left[\text{sign}(Lb_3)\times\text{sign}(Lb_6)\times\text{sign}(Lb_{\text{v0}})\right]\min\left[|Lb_3|,|Lb_6|,|Lb_{\text{v0}}|\right] \\ &= \left[\text{sign}(5{,}43)\times\text{sign}(-11{,}93)\times\text{sign}(-1{,}70)\right] \\ &\times\min\left[|5{,}43|,|-11{,}93|,|-1{,}70|\right] \\ &= 1{,}70 \end{aligned}$$

Usando um procedimento análogo, a informação extrínseca para o bit b_3 na dimensão vertical é calculada da seguinte forma:

$$\begin{aligned} L_{ev}(b_3) &= \left[\text{sign}(Lb_0)\times\text{sign}(Lb_6)\times\text{sign}(Lb_{\text{v0}})\right]\min\left[|Lb_0|,|Lb_6|,|Lb_{\text{v0}}|\right] \\ &= \left[\text{sign}(8{,}38)\times\text{sign}(-11{,}93)\times\text{sign}(-1{,}70)\right] \\ &\times\min\left[|8{,}38|,|-11{,}93|,|-1{,}70|\right] \\ &= 1{,}70 \end{aligned}$$

Para o bit b_6 tem-se:

$$\begin{aligned} L_{ev}(b_6) &= \left[\text{sign}(Lb_0)\times\text{sign}(Lb_3)\times\text{sign}(Lb_{\text{v0}})\right]\min\left[|Lb_0|,|Lb_3|,|Lb_{\text{v0}}|\right] \\ &= \left[\text{sign}(8{,}38)\times\text{sign}(5{,}43)\times\text{sign}(-1{,}70)\right] \\ &\times\min\left[|8{,}38|,|5{,}43|,|-1{,}70|\right] \\ &= -1{,}70 \end{aligned}$$

E para o bit b_{v0} tem-se:

$$\begin{aligned} L_{ev}(b_{\text{v0}}) &= \left[\text{sign}(Lb_0)\times\text{sign}(Lb_3)\times\text{sign}(Lb_6)\right]\min\left[|Lb_0|,|Lb_3|,|Lb_6|\right] \\ &= \left[\text{sign}(8{,}38)\times\text{sign}(5{,}43)\times\text{sign}(-11{,}93)\right] \\ &\times\min\left[|8{,}38|,|5{,}43|,|-11{,}93|\right] \\ &= -5{,}43 \end{aligned}$$

Segue-se o mesmo procedimento para os demais bits na dimensão vertical, o que permite obter todos valores mostrados na Figura 10(*a*). Observa-se que, se fosse tomada uma decisão abrupta a partir da informação extrínseca calculada na dimensão vertical, haveria erros nos bits b_2 e b_5.

Agora todos os valores de informação extrínseca fornecidos na Figura 10(*a*) servirão como LLR's *a priori* para a decodificação na próxima, a dimensão horizontal. Então, utilizando novamente a aproximação dada em (17), a informação extrínseca para o bit b_0 na dimensão horizontal é calculada da seguinte maneira:

$$\begin{aligned} L_{eh}(b_0) &= \left[\text{sign}(Lb_1)\times\text{sign}(Lb_2)\times\text{sign}(Lb_{\text{h0}})\right]\min\left[|Lb_1|,|Lb_2|,|Lb_{\text{h0}}|\right] \\ &= \left[\text{sign}(3{,}93{+}0{,}68)\times\text{sign}(2{,}63{+}3{,}77)\times\text{sign}(-6{,}21-3{,}52)\right] \\ &\times\min\left[|3{,}93{+}0{,}68|,|2{,}63{+}3{,}77|,|-6{,}21-3{,}52|\right] \\ &= -4{,}61 \end{aligned}$$

Para o bit b_1 tem-se:

$$\begin{aligned} L_{eh}(b_1) &= \left[\text{sign}(Lb_0)\times\text{sign}(Lb_2)\times\text{sign}(Lb_{h0})\right]\min\left[|Lb_0|,|Lb_2|,|Lb_{h0}|\right] \\ &= \left[\text{sign}(8{,}38+1{,}70)\times\text{sign}(2{,}63+3{,}77)\times\text{sign}(-6{,}21-3{,}52)\right] \\ &\quad\times\min\left[|8{,}38+1{,}70|,|2{,}63+3{,}77|,|-6{,}21-3{,}52|\right] \\ &= -6{,}40 \end{aligned}$$

1,70	0,68	3,77	-3,52
1,70	-0,68	-2,63	-3,52
-1,70	3,93	2,63	-3,52
-5,43	-0,68	-2,63	-6,21

(*a*)

-4,61	-6,40	-4,61	4,61
-6,04	6,40	6,04	6,04
-4,61	8,25	4,61	-4,61
-6,14	-7,13	-6,14	-6,14

(*b*)

5,47	-1,79	1,79	-5,13
1,09	0,36	-0,36	-11,24
-18,24	12,86	12,86	-14,76
-13,27	-13,27	-21,88	-15,8

(*c*)

Figura 10 *- Resultados da primeira iteração: (a) informação extrínseca obtida da decodificação vertical; (b) informação extrínseca obtida da decodificação horizontal e (c) razão de verossimilhança logarítmica (LLR) total.*

Procedendo-se de forma análoga obtêm-se os demais valores dados na Figura 10(*b*). Observe agora que, se fosse tomada uma decisão abrupta a partir da informação extrínseca calculada na dimensão horizontal, haveria erros nos bits b_0, b_1, b_{h0}, b_3, b_4, e b_{h1}. Entretanto, somando as LLR's do canal (Figura 8(c)) com as informações extrínsecas calculadas nas dimensões vertical e horizontal (Figura 10(*a*) e Figura 10(*b*)), de acordo com o Algoritmo 2, passo 7, obtém-se o arranjo mostrado na Figura 10(*c*). Uma decisão abrupta a partir desse arranjo levaria a erros nos bits b_1, b_2, b_4 e b_5. Observe também que praticamente todos os bits correspondentes às decisões corretas têm suas LLR's totais com magnitudes mais elevadas, o que pode ser interpretado como um aumento na confiabilidade das decisões já ao final da primeira iteração, resultado do processo de decodificação turbo.

Uma iteração está completa depois da decodificação nas dimensões vertical e horizontal. Em seguida inicia-se a segunda iteração com uma nova decodificação na dimensão vertical, utilizando os valores de informação extrínseca calculados na dimensão horizontal da iteração anterior como LLR's *a priori*. Por exemplo, na segunda iteração a informação extrínseca para o bit b_0 na dimensão vertical é calculada através das seguintes operações:

$$\begin{aligned} L_{ev}(b_0) &= \left[\text{sign}(Lb_3)\times\text{sign}(Lb_6)\times\text{sign}(Lb_{v0})\right]\min\left[|Lb_3|,|Lb_6|,|Lb_{v0}|\right] \\ &= \left[\text{sign}(5{,}43-6{,}04)\times\text{sign}(-11{,}93-4{,}61)\times\text{sign}(-1{,}70-6{,}14)\right] \\ &\quad\times\min\left[|5{,}43-6{,}04|,|-11{,}93-4{,}61|,|-1{,}70-6{,}14|\right] \\ &= -0{,}61 \end{aligned}$$

Para o bit b_3 tem-se:

$$\begin{aligned} L_{ev}(b_3) &= \left[\text{sign}(Lb_0)\times\text{sign}(Lb_6)\times\text{sign}(Lb_{v0})\right]\min\left[|Lb_0|,|Lb_6|,|Lb_{v0}|\right] \\ &= \left[\text{sign}(8{,}38-4{,}61)\times\text{sign}(-11{,}93-4{,}61)\times\text{sign}(-1{,}70-6{,}14)\right] \\ &\quad\times\min\left[|8{,}38-4{,}61|,|-11{,}93-4{,}61|,|-1{,}70-6{,}14|\right] \\ &= 3{,}77 \end{aligned}$$

Procedendo-se de forma similar obtêm-se os demais valores apresentados na Figura 11(*a*). Se fosse tomada uma decisão abrupta a partir da informação extrínseca calculada na dimensão vertical da segunda iteração, haveria erros nos bits b_0, b_1, b_4, b_6 e b_{v0}.

Todos os valores de informação extrínseca mostrados na Figura 11(*a*) agora servirão como LLR's *a priori* para a decodificação suave da dimensão horizontal na segunda iteração. Nesta etapa a informação extrínseca para o bit b_0 é calculada da seguinte forma:

$$\begin{aligned} L_{eh}(b_0) &= \left[\text{sign}(Lb_1)\times\text{sign}(Lb_2)\times\text{sign}(Lb_{h0})\right]\min\left[|Lb_1|,|Lb_2|,|Lb_{h0}|\right] \\ &= \left[\text{sign}(3{,}93-1{,}03)\times\text{sign}(2{,}63-2{,}27)\times\text{sign}(-6{,}21-7{,}71)\right] \\ &\quad\times\min\left[|3{,}93-1{,}03|,|2{,}63-2{,}27|,|-6{,}21-7{,}71|\right] \\ &= -0{,}36 \end{aligned}$$

Para o bit b_1 tem-se:

$$\begin{aligned} L_{eh}(b_1) &= \left[\text{sign}(Lb_0)\times\text{sign}(Lb_2)\times\text{sign}(Lb_{h0})\right]\min\left[|Lb_0|,|Lb_2|,|Lb_{h0}|\right] \\ &= \left[\text{sign}(8{,}38-0{,}61)\times\text{sign}(2{,}63-2{,}27)\times\text{sign}(-6{,}21-7{,}71)\right] \\ &\quad\times\min\left[|8{,}38-0{,}61|,|2{,}63-2{,}27|,|-6{,}21-7{,}71|\right] \\ &= -0{,}36 \end{aligned}$$

Operando de forma similar para todos os bits da palavra-código obtêm-se os demais valores fornecidos na Figura 11(*b*). Tomando-se uma decisão abrupta a partir da informação extrínseca calculada na dimensão horizontal da segunda iteração, erros nos bits b_0, b_1, b_{h0}, b_3, b_4 e b_{h1} ainda permaneceriam. Somando as LLR's do canal (Figura 8(c)) com as informações extrínsecas calculadas nas dimensões vertical e horizontal da segunda iteração (Figura 11(*a*) e Figura 11(*b*)) obtém-se o arranjo mostrado na Figura 11(*c*). A decisão abrupta tomada a partir desse arranjo *não mais levaria a erros* e o processo iterativo poderia ser interrompido.

-0,61	-1,03	-2,27	-7,71
3,77	2,47	1,97	-1,61
0,61	1,03	1,97	-1,61
0,61	-1,03	-1,97	-1,61

(*a*)

-0,36	-0,36	-2,90	0,36
-1,79	1,79	2,90	1,79
-1,71	7,59	1,71	-1,71
-5,13	-1,09	-1,09	-1,09

(*b*)

7,41	2,54	-2,54	-13,57
7,41	-1,11	1,11	-13,57
-13,03	9,30	9,30	-9,94
-6,22	-7,58	-16,17	-6,22

(*c*)

Figura 11 *- Resultados da segunda iteração: (a) informação extrínseca obtida da decodificação vertical; (b) informação extrínseca obtida da decodificação horizontal e (c) razão de verossimilhança logarítmica (LLR) total.*

Observe mais uma vez, agora na Figura 11(*c*), que praticamente todos os bits têm suas LLR's totais com magnitudes mais elevadas em relação à Figura 10(*c*), demonstrando mais uma vez um aumento na confiabilidade das decisões, como conseqüência do processo de decodificação turbo.

Realizando-se seis iterações, os arranjos correspondentes à informação extrínseca obtida na

decodificação das dimensões vertical e horizontal, bem como aquele correspondente à LLR total devem ser aqueles mostrados na Figura 12. Observe que as decisões seriam tomadas corretamente a partir de qualquer desses arranjos. Observe também que, embora a decisão não tenha sido afetada da segunda iteração para a sexta iteração, mais uma vez as magnitudes das LLR's totais tiveram seus valores aumentados em relação àqueles apresentados na Figura 11(*c*). Então, todas as decisões têm, agora, confiabilidades mais elevadas.

4,83	4,76	-8,07	-11,76
7,77	-3,33	7,77	-5,61
-4,83	3,33	7,77	-5,61
-4,83	-3,33	-7,77	-5,61

(*a*)

5,44	5,44	-8,69	-5,44
4,00	-4,00	8,69	-4,00
-4,00	12,24	4,00	-4,00
-8,79	-6,52	-6,52	-6,52

(*b*)

18,64	14,13	-14,13	-23,41
17,21	-12,69	12,69	-23,37
-20,77	16,24	17,39	-16,24
-15,32	-15,32	-27,40	-15,66

(*c*)

Figura 12 *- Resultados da sexta iteração: (a) informação extrínseca obtida da decodificação vertical; (b) informação extrínseca obtida da decodificação horizontal e (c) razão de verossimilhança logarítmica (LLR) total.*

VII - COMENTÁRIOS GERAIS SOBRE CÓDIGOS PRODUTO SPC MULTIDIMENSIONAIS

OS CONCEITOS E RESULTADOS abordados neste texto, complementados com aqueles citados em [10], [15] e [22], podem ser estendidos para códigos produto de paridade simples com mais de duas dimensões, incluindo a paridade das paridades no processo de codificação/decodificação e obtendo-se um certo tipo de informação extrínseca sobre essa paridade. Assim, a informação *a priori* e a extrínseca passam a estar associadas a todos os bits, não somente aos bits de informação como em [10] e [22].

Considerando-se que um algoritmo apropriado, com desempenho ótimo ou pouco distante do ótimo esteja sendo utilizado, com o aumento da dimensão do código produto de paridade simples os retornos em termos de desempenho são sempre menores entre cada aumento de dimensão. Este fato pode ser constatado através dos resultados reportados em [15] e de outros apresentados mais adiante neste texto.

Os ganhos de desempenho em função do aumento da dimensionalidade ocorrem principalmente devido ao aumento da distância mínima do código e da aleatoriedade atribuída ao código resultante. Adicionalmente, a cada iteração em um esquema de decodificação turbo de um código produto multidimensional, percebem-se maiores retornos em desempenho em relação a esquemas bidimensionais. Este fato é devido principalmente à redução na correlação entre as probabilidades envolvidas nos cálculos das LLR's a cada iteração. Melhores desempenhos também podem ser obtidos às custas do aumento do tamanho dos blocos para um código de dimensão *D* [10]. Esse fato é mais claramente percebido quando os códigos componentes possuem, individualmente, maior capacidade de correção de erros.

Como pôde ser verificado através de simulações realizadas pelo autor, para códigos produto de paridade simples bidimensionais o aumento no tamanho dos blocos não trouxe ganhos significativos em termos de desempenho. Há, inclusive, uma redução de desempenho com o uso de blocos de tamanho muito elevado, o que é justificado pela predominância da influência do número de palavras código de baixo peso nestes casos. Para baixos valores de relação sinal-ruído percebe-se também uma redução no desempenho quando é aumentado o tamanho dos blocos para esses códigos bidimensionais.

Uma outra constatação se refere a uma melhoria (não muito significativa, contudo) no desempenho do processo de correção de erros para códigos produto SPC completos, de dimensão maior que 2, quando é incluído um processo de entrelaçamento temporal aleatório (*random interleaving*) entre as etapas de codificação em cada dimensão[7] [15]. Para duas dimensões é percebida uma redução nesse desempenho, em vez de uma melhoria. Um resultado adicional se refere à possibilidade de redução do patamar de saturação da taxa de erro de bit. Contudo, o entrelaçamento temporal aleatório, apesar de reduzir o número de palavras código de baixo peso reduz, também, a distância mínima do código [15].

Deve-se lembrar que, diferentemente do que acontece com os códigos de bloco turbo com concatenação paralela, o processo de entrelaçamento temporal é crucial à melhoria do desempenho dos códigos turbo convolucionais e dos códigos de bloco turbo com concatenação serial. Para estes últimos, tanto a profundidade quanto a "aleatoriedade" do processo de entrelaçamento temporal tem relação direta com o desempenho do processo de decodificação turbo [29][35][27]. Para os códigos de bloco com concatenação paralela, a profundidade do entrelaçamento temporal pode se restringir ao comprimento do bloco de mensagem ou, no pior caso, do bloco completo (informação mais paridade), sem melhorias de desempenho significativas a partir daí.

Em casos onde a dimensionalidade do código produto de paridade simples deva ser aumentada em demasia, acarretando em um excessivo aumento no tamanho dos blocos, alternativamente pode-se implementar o código com códigos componentes mais eficientes, por exemplo, códigos de Hamming. Como citado no início deste tutorial, surpreendentes resultados utilizando códigos componentes de Hamming operando com blocos de tamanho não muito elevado (1023,1013) foram reportados em [8].

Como comentário adicional acerca dos possíveis aumentos de desempenho dos códigos produto com o aumento da sua dimensão e/ou com o uso de códigos componentes de melhor capacidade de correção de erros, enfatiza-se que, embora a distância mínima seja um parâmetro relevante no dimensionamento de um esquema de codificação de canal, argumentos citados em [36]

[7] O código SPC-PC utilizado em [15] é um código completo, apresentando baixos valores de saturação na taxa de erro de bit (*error floor*) e um decréscimo mais abrupto da taxa de erro de bit (BER) com o aumento da relação E_b/N_0.

contribuem para que se conclua que o principal critério de melhoria de desempenho seja atribuído às propriedades de aleatoriedade dos códigos. Vários exemplos são citados em [36] mostrando resultados atrativos com códigos de distância mínima relativamente baixa, mas com distribuições de pesos das palavras-código que se assemelham à distribuição de um código aleatório. Em [37] são antecipados alguns comentários também nesse sentido, porém objetivando determinar qual a distância de Hamming mínima realmente necessária a um determinado desempenho alvo de um esquema de codificação de canal.

Também merecem ser comentadas algumas questões relacionadas ao tempo que o decodificador turbo precisa para fornecer sua melhor estimativa da palavra-código ou palavra de informação. Duas medidas de tempo podem ser aqui definidas: a *latência* e o *atraso*. O atraso é função da taxa de transmissão e do tamanho do bloco codificado, pois de forma independente do tipo de decodificação, há que se esperar pela recepção de todos os símbolos associados a uma palavra-código para que a estimação dessa palavra-código se inicie[8]. Já a latência está associada ao tempo de processamento necessário para que, depois de recebida uma palavra-código completa, o primeiro bit estimado válido de informação esteja disponível na saída do decodificador. Este tempo depende do número de iterações no decodificador turbo e também da complexidade do algoritmo de decodificação. Para os códigos produto, o número de dimensões também tem influência na latência de forma diretamente proporcional. Nesse sentido, um dos grandes desafios impostos ao desenvolvimento de novos algoritmos de decodificação turbo está relacionado com o objetivo de redução da latência de decodificação. Os códigos produto de paridade simples apresentam relativa vantagem nesse aspecto, posto que sua decodificação turbo pode ser significativamente mais simples que aquelas desenvolvidas para outros códigos ou mesmo para códigos produto com outros códigos componentes. Alguns exemplos neste sentido podem ser obtidos em [12] e [14].

VIII - Alguns Resultados de Simulação em Canal AWGN e Rayleigh Plano

Nesta seção são fornecidos alguns resultados de simulação para avaliação do desempenho de códigos produto de paridade simples com decodificação turbo, SPC-TPC (*Single-Parity Check Turbo Product Code*), em canal AWGN e em canal com desvanecimento plano do tipo Rayleigh. O código utilizado como teste é o código *D*-dimensional $(8,7,7)^D$, para *D* igual a 2, 3, 4 e 5. O algoritmo de decodificação dos códigos componentes é o algoritmo MAP símbolo-a-símbolo apresentado neste texto, o mesmo utilizado em [15]. Os resultados de simulação foram obtidos com a rotina comentada no Apêndice, para modulação BPSK com detecção coerente.

[8] Esta afirmação é válida para códigos de bloco e para outros esquemas de codificação de canal que operem com blocos de símbolos codificados, por exemplo, quando se utiliza um código convolucional com terminação na sua treliça de forma que ao final da seqüência de informação de comprimento definido o estado do codificador retorne ao estado nulo.

A Figura 13 apresenta resultados de simulação para os códigos $(8,7,7)^D$ no canal AWGN. O desempenho da sinalização BPSK sem codificação de canal também é fornecido como referência para comparação. Como esperado, o desempenho do código produto de paridade simples completo é significativamente melhorado com o aumento da dimensão do código, com aumentos de ganho de codificação aproximadamente iguais de um código com dimensão *D* para outro com dimensão *D* + 1.

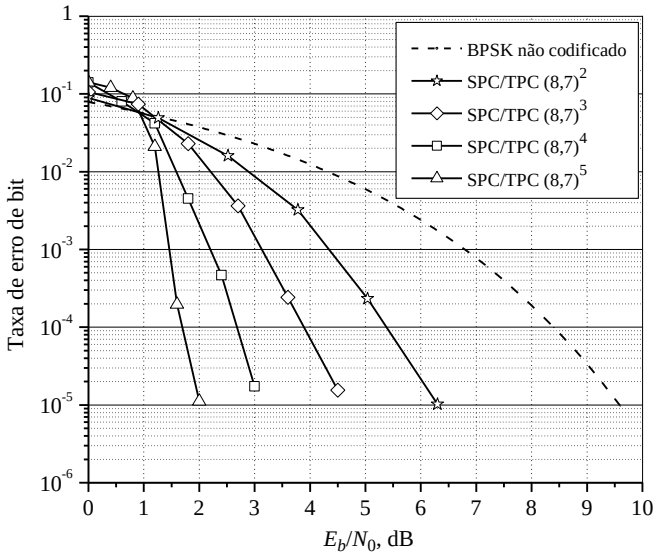

Figura 13 *- Resultados de simulação para um código produto de paridade simples* $(8,7,7)^D$ *com decodificação turbo em canal AWGN e modulação BPSK, para D = 2, 3, 4 e 5.*

Observa-se ainda, na Figura 13, um comportamento similar àquele descrito através da Figura 2, no que se refere à taxa de decréscimo da probabilidade de erro de bit com o aumento da relação E_b/N_0.

Para o código de cinco dimensões tem-se um ganho de codificação de pouco mais de 7 dB @ 10^{-5} de taxa de erro de bit. Para este código, cuja taxa é aproximadamente 0,513, o mínimo valor de E_b/N_0 para comunicação livre de erros, correspondente ao *Limite de Shannon*, é de cerca de 0,19 dB em canal AWGN com modulação BPSK [26]. Dessa forma, esse código apresenta um desempenho que dista cerca de 1,8 dB da capacidade do canal, @ 10^{-5} de taxa de erro de bit, um resultado bastante atrativo, dada a simplicidade de codificação e de decodificação.

A Figura 14 apresenta resultados de simulação para os códigos $(8,7,7)^D$, também para *D* igual a 2, 3, 4 e 5, em canal com desvanecimento Rayleigh plano. O desempenho da sinalização BPSK sem codificação de canal é fornecido para comparação. Tais resultados foram também obtidos utilizando-se a rotina comentada no Apêndice, com o uso do conhecimento da informação de estado de canal pelo receptor. Foi observado em outras simulações um decréscimo de desempenho de cerca de 1 dB quando a informação de estado de canal não é utilizada no processo de decodificação turbo.

Para o canal com desvanecimentos Rayleigh pode-se perceber, pelos resultados mostrados na Figura 14, que os ganhos de codificação são significativamente mais elevados que aqueles obtidos no canal AWGN. Por exemplo, para o código $(8,7,7)^5$ tem-se um ganho de codificação de cerca de 40 dB @ 10^{-5} de taxa de erro de bit. Para este código, o *Limite de Shannon* é de cerca de 1,6 dB para o canal Rayleigh com modulação BPSK [26]. Então, esse código apresenta um desempenho distante cerca de 2,5 dB do Limite de Shannon, @ 10^{-5} de taxa de erro de bit, um resultado também bastante atrativo para um esquema de codificação de canal de baixa complexidade como o SPC-TPC.

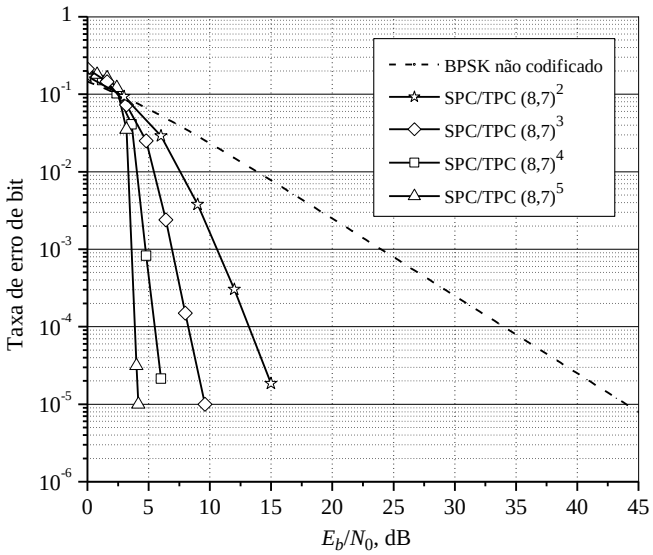

***Figura 14** - Resultados de simulação para um código produto de paridade simples $(8,7,7)^D$ com decodificação turbo no canal com desvanecimentos Rayleigh planos e modulação BPSK, para D = 2, 3, 4 e 5.*

Tanto na Figura 13 quanto na Figura 14 não se observa o fenômeno de saturação na taxa de erro de bit, o *error floor*. De fato, como antecipado no início deste tutorial, esse fenômeno tende a ocorrer em valores mais baixos de taxa de erro de bit para o tipo de código em questão, em comparação com os códigos turbo convolucionais e com os códigos produto incompletos (concatenação paralela). Por limitações computacionais o *error floor* não foi atingindo para nenhum dos casos simulados. Para valores elevados de relação sinal-ruído, o *error floor* normalmente coincide com o Limitante de União (ou com outro limitante equivalente) para o desempenho do código analisado. Em [15] podem ser verificados exemplos que ratificam esta afirmativa: por exemplo, de acordo com o Limitante de União, o desempenho do código $(8,7,7)^5$ deverá a apresentar um modesto comportamento de saturação da taxa de erro de bit abaixo de 10^{-10}. A partir deste valor o desempenho deverá seguir o que prevê esse limitante, numa taxa de decréscimo mais suave da BER em função da relação E_b/N_0 [15, p. 65].

A empresa *Comtech AHA Corporation* fabrica soluções baseadas em códigos produto e códigos LDPC, com vazões da ordem de 30 a 300 Mbit/s para os Códigos Produto e 30 a 100 Mbit/s para os códigos LDPC. Mais detalhes sobre esses produtos podem ser obtidos em *http://www.comtechaha.com*.

AGRADECIMENTO

O AUTOR AGRADECE imensamente aos revisores pelas críticas construtivas e sugestões que em muito colaboraram para a melhoria da qualidade deste trabalho.

APÊNDICE

POR MEIO DO LINK a seguir tem-se acesso a uma rotina elaborada na plataforma *Mathcad* para análise do desempenho de códigos produto de paridade simples com decodificação turbo:

http://cict.inatel.br/nova2/docentes/dayan/publications/Inatel_09/

Uma versão em *pdf* dessa rotina é também fornecida por meio desse link, permitindo que ela possa ser "traduzida" para um outro aplicativo, caso necessário. Nessa rotina podem ser escolhidos: os códigos componentes do código produto (que são idênticos em todas as dimensões), o número de dimensões do código resultante, a faixa de valores de E_b/N_0, o número de pontos no gráfico de E_b/N_0 *versus* taxa de erro de bit, o número de erros mínimo simulado a cada valor de E_b/N_0, o número de iterações e o canal de comunicação (AWGN ou Rayleigh plano). Pode-se ainda optar por utilizar ou não utilizar a informação de estado de canal no processo de decodificação turbo.

A modulação utilizada é a BPSK em sua forma equivalente em banda base e com detecção coerente. O desempenho teórico dessa modulação é também traçado no gráfico resultante como forma de comparação, de acordo com o canal escolhido.

Para utilizar a rotina fornecida por meio do link supramencionado, deve-se executar diretamente o arquivo com extensão *mcd* no *Mathcad*, versão 2001*i* ou superior. Para versões inferiores à 2001*i*, basta realizar uma cópia *ipsis litteris* do conteúdo do arquivo com extensão *pdf* na área de trabalho do *Mathcad*. Pode-se também traduzir o conteúdo do arquivo com extensão *pdf* para um outro aplicativo, por exemplo, para o *Matlab*. Este processo de tradução é bastante simples, dado o aspecto "amigável" da interface com o usuário do *Mathcad*.

Na rotina em questão percebe-se que o cálculo da informação extrínseca está sendo realizado de duas formas: um cálculo exato conforme a expressão (16), quando este cálculo não apresenta erro computacional, e um cálculo aproximado de acordo com a expressão (17), quando esse erro ocorre. Tal erro tem maior chance de ocorrência quando os valores das LLR's operadas são muito elevados, por conta do cálculo numérico efetuado pelo *Mathcad* envolvendo a tangente hiperbólica presente em (16). Essa possibilidade de erro computacional também foi constatada por D. M. Rankin em [15] e a alternativa lá utilizada para evitar tais erros foi o truncamento nos valores das LLR's

operadas. Ambas as alternativas foram testadas para a composição deste trabalho e levaram aos mesmos resultados.

REFERÊNCIAS

[1] FORNEY Jr, G. D. **Concatenated Codes**, Ph.D. Thesis, Cambridge, Massachusetts Institute of Technology - MIT, USA, 1966.

[2] BERROU, Claude, A. GLAVIEUX and P. THITIMAJSHIMA. **Near Shannon limit error-correcting coding and decoding: Turbo-Codes**, in Proceedings of the 1993 Communication Conference, ICC'93, Geneva, Switzerland, pp. 1064-1070, May 1993.

[3] BAHL, L. R., J. COCKE, F. JELINEK and J. RAVIV. **Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate.** IEEE Transactions on Information Theory, pp. 284-287. March 1974.

[4] HAGENAUER, J and P. HOEHER. **A Viterbi Algorithm With Soft-Decision Outputs and its Applications**. Proceedings of Globecom '89, Dallas, Texas, pp. 47.11-47.17, November 1989.

[5] GALLAGER, R. G. **Low-Density Parity-Check Codes**. Cambridge, MIT Press, 1963.

[6] LODGE, J, R. YOUNG, P. HOEHER and J. HAGENAUER**. Separable MAP 'filters' for the decoding of product and concatenated codes**. Proceedings of ICC'93, Geneva, pp. 1740-1745, May 1993.

[7] BERROU, Claude, A. GLAVIEUX. **Reflections on the Prize paper: "Near optimum error correcting coding and decoding: Turbo codes"**, IEEE IT Society Newsletter, Vol. 48, N° 2, June 1998.

[8] NICKL, H., J. HAGENAUER and F. BURKERT. **Approaching Shannon's Capacity Limit by 0.2 dB Using Simple Hamming Codes**. IEEE Communications Letters, Vol. 1 5, pp. 130-132, September 1997.

[9] PROAKIS, J. G. **Digital Communications**. 3rd Edition - McGraw Hill. New York, 1995.

[10] HAGENAUER, J., E. OFFER and L PAPKE. **Iterative Decoding of Binary Block and Convolutional Codes**. IEEE Transactions on Information Theory, pp. 429-445, Vol. 42 N° 2, March 1996.

[11] BERROU, Claude and A. GLAVIEUX. **Near Optimum Error Correcting Coding And Decoding: Turbo-Codes.** IEEE Transactions on Communications, pp. 1261-1271, Vol. 44, n° 10. October 1996.

[12] PYNDIAH, Ramesh M. **Near-Optimum Decoding of Product Codes: Block Turbo Codes.** IEEE Transactions on Communication, pp. 1003-1010, Vol. 46, n° 8. August 1998.

[13] HUNT, Andrew, S. CROZIER and D. FALCONER. **Hyper-codes: High-performance Low-Complexity Error-Correcting Codes**. Proceedings of the 19th Biennial Symposium on Communications, pp. 263-267, Kingston, Canada, June 1998.

[14] DAVE, Sameep, J. KIM and S. C. KWATRA. **An Efficient Decoding Algorithm for Block Turbo Codes**. IEEE Transactions on Communications, pp. 41-46, Vol. 49, n° 1, January 2001.

[15] RANKIN, David M., **Single Parity Check Product Codes and Iterative Decoding**. Ph.D. Thesis, University of Canterbury, Christchurch, New Zealand, May/2001. *Veja também*: RANKIN, D. M. and GULLIVER, T. A., **Single Parity Check Product Codes**, IEEE Transactions on Communication, vol. 49, no. 8, pp. 1354-1362, August 2001.

[16] GUIMARÃES, D. A. and J. PORTUGHEIS. **A Class of Product Codes and Its Iterative (Turbo) decoding**, in: Proceedings of the 3rd International Symposium on Turbo Codes & Related Topics, pp. 431-434: Brest, France, September 1-5, 2003.

[17] BARBULESCU, S. A. and S. S. PIETROBON. **Turbo Codes: a Tutorial on a New Class of Powerful Error Correcting Coding Schemes. Part I: Code Structures and Interleaver Design**. Institute for Telecommunications Research, University of South Australia, October 1998.

[18] SHANNON, C. E., **A Mathematical Theory of Communication**, reprinted with corrections from The Bell System Technical Journal, Vol. 27, July, October 1948.

[19] HAYKIN, S. and M. SELLATHURAI. **Turbo-BLAST with Multi-loop Feedback Receiver**, in: Proceedings of the 3rd International Symposium on Turbo Codes & Related Topics, pp. 195-202: Brest, France, September 1-5, 2003.

[20] ELIAS, P. **Error-free Coding**. IRE Transactions on Information Theory, PGIT–4, pp. 29-37, September 1954.

[21] HONARY, Bahram. **Trellis Decoding of Block Codes**. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1997.

[22] SKLAR, B., **A Primer on Turbo Code Concepts**. IEEE Communication Magazine, pp. 94-101, December 1997.

[23] GALLAGER, R. G., **Low density parity check code**s. IRE Trans. Info. Theory, IT-8:21, 28, Jan 1962. Veja também: R. G. Gallager. Low Density Parity Check Codes. Number 21 in Research monograph series. MIT Press, Cambridge, Mass., 1963.

[24] MACKAY, D. J. C. and R.M. NEAL, **Near Shannon limit performance of low density parity check codes**, IEE Electronics Letters, vol. 32, no. 18, pp. 1645-1655, 29th Aug. 1996.

[25] GUIMARÃES, D. A., **Decodificação Turbo de Códigos Produto de Paridade Simples**, Revista Telecomunicações, Vol. 5, No. 1, pp. 11-28: Inatel. Santa Rita do Sapucaí, MG, Junho, 2002.

[26] GUIMARÃES, D. A. **Uma Classe de Códigos Produto e sua Decodificação Turbo Aplicada em um Sistema CDMA Multiportadora**. Tese de Doutorado: Universidade Estadual de Campinas - Unicamp. Campinas, SP, June 2003.

[27] BENEDETTO, Sergio, D. DIVSALAR, G. MONTORSI and F. POLLARA. **Serial Concatenation of Interleaved Codes: Performance Analysis, Design and Iterative Decoding.** IEEE Transactions on Information Theory, Vol. 44, n° 3, pp. 909-926, May/1998.

[28] BENEDETTO, Sergio and E. BIGLIERI. **Principles of Digital Transmission With Wireless Applications**. Kluwer Academic / Plenum Publishers, New York, 1999.

[29] BARBULESCU, S. A. **Iterative Decoding of Turbo Codes and Other Concatenated Codes**. Ph.D. Thesis, Faculty of Engineering, University of South Australia, February 1996.

[30] HUNT, Andrew, **Hyper-codes: High-performance Low-Complexity Error-Correcting Codes**. M. Sc. Thesis, Faculty of Engineering, Ottawa-Carleton Institute of Electrical Engineering, Carleton University, Ottawa, Ontario, Canada, May 1998.

[31] HAYKIN, S. **Communication Systems.** 4rd Edition - John Wiley and Sons, Inc. New York, USA, 2001.

[32] CHASE, David. **A Class of Algorithms for Decoding Block Codes With Channel Measurement Information.** IEEE Transactions on Information Theory, pp. 170-182, Vol. IT-18, n° 1. January 1972.

[33] KANEKO, T., T. NISHIJIMA, H. INAZUMI, S. HIRASAWA. **An Efficient Maximum-Likelihood-Decoding Algorithm for Linear Block Codes with Algebraic Decoder**. IEEE

Transactions on Information Theory, pp. 320-327, Vol. 40, N° 2, March 1994.

[34] ROBERTSON, P., E. VILLEBRUN and P. HOEHER. **A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain**. In Proceedings of the 1995 International Conference on Communications, ICC '95, Seattle, pp. 1009-1013, Vol. 2, 1995.

[35] BENEDETTO, Sergio, G. MONTORSI. **Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes.** IEEE Transactions on Information Theory, Vol 42, n° 2, pp. 409-428. March 1996.

[36] YUE, D. W. and E. H. YANG. **Asymptotically Gaussian Weight Distribuition and Performance of Multicomponent Turbo Block Codes and Product Codes**. IEEE Transactions on Communications, v. 52, n.5, p. 728-736, May 2004.

[37] BERROU, C., E. MAURY and H. GONZALEZ. **Which Minimum Hamming Distance Do We Really Need?**, in: Proceedings of the 3rd International Symposium on Turbo Codes & Related Topics, pp. 141-148: Brest, France, September 1-5, 2003.

SOBRE O AUTOR

Dayan Adionel Guimarães nasceu em Carrancas, MG, em 01 de março de 1969. Possui os títulos: *Técnico em Eletrônica* (ETE "FMC", 1987), *Engenheiro Eletricista* (Inatel, 1994), *Especialista em Engenharia de Comunicação de Dados* (Inatel, 2003), *Especialista em Administração* com ênfase em Gerência de RH (FAI, 1996), *Mestre em Engenharia Elétrica* (Unicamp, 1998) e *Doutor em Engenharia Elétrica* (Unicamp, 2003). De 1988 a 1993 desenvolveu sensores e equipamentos para instrumentação industrial e controle e também foi Supervisor de Produção e Supervisor de Engenharia de Produtos na SENSE Sensores e Instrumentos. Desde de janeiro de 1995 é Professor do Inatel onde, por oito anos, foi responsável pela estrutura que dá apoio às atividades de ensino prático nas áreas de Telecomunicações, Eletrônica e Eletrotécnica. É membro do corpo editorial da revista *Telecomunicações* do Inatel. Suas pesquisas incluem aspectos gerais sobre transmissão digital e sistemas de comunicação móvel, especificamente sistemas CDMA Multiportadora e esquemas de codificação para canais com desvanecimento, especificamente códigos turbo de bloco.