UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE COMUNICAÇÕES

Tese de Mestrado

UMA REDE DE CODIFICAÇÃO/DECODIFICAÇÃO PARA CANAIS NÃO-COERENTES

Autora:

Juliana Ferreira Pedrosa

Orientador:

Prof. Dr. Jaime Portugheis

Área de Concentração:

Telecomunicações e Telemática

Banca Examinadora:

Prof. Dr. Jaime Portugheis (Presidente)	DECOM – FEEC – UNICAMP
Prof. Dr. Ernesto Leite Pinto	IME
Prof. Dr. Reginaldo Palazzo Jr.	DT – FEEC – UNICAMP
Prof. Dr. Walter da Cunha Borelli	DT – FEEC – UNICAMP

Campinas, 07 de dezembro de 2001

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

P343r

Pedrosa, Juliana Ferreira
Uma rede de codificação/decodificação para canais não-coerentes / Juliana Ferreira Pedrosa.--Campinas, SP: [s.n.], 2001.

Orientador: Jaime Portugheis.
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Modulação digital. 2. Códigos de controle. de erros (Teoria da informação). 3. Sistemas de comunicação móvel. 4. Comunicações digitais. 5. Teoria da codificação. I. Portugheis, Jaime. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo

Em virtude dos extraordinários resultados, amplamente citados na literatura, alcançados com o uso da decodificação turbo, novas linhas de pesquisa vêm sendo sugeridas e seguidas, visando aplicar estas contribuições em novos cenários de decodificação. Entre as aplicações existentes, destaca-se o conceito de redes de decodificação, sistemas de recepção que utilizam o que se chamou de princípio turbo ou processamento iterativo. Este trabalho tem, como ponto de partida, uma rede de decodificação proposta para a concatenação serial de um codificador convolucional e um modulador DPSK em *canais coerentes*. A proposta, então, foi a substituição do codificador convolucional por um codificador turbo, criando, assim, um outro cenário de decodificação que permite a utilização e combinação de duas formas diferentes de iteração: uma, devido ao próprio processo iterativo, e outra, inerente à decodificação turbo. Pelo fato de a rede de decodificação proposta se ambientar em *canais não-coerentes,* uma modificação no algoritmo de detecção MAP foi sugerida. Adicionalmente, é também desenvolvida uma descrição detalhada do sistema em estudo. Resultados de simulação indicam que determinadas combinações das duas formas de iteração utilizadas, na rede de decodificação, possuem um compromisso melhor entre desempenho e complexidade de implementação.

Abstract

Due to the extraordinary results, widely mentioned in the literature, attained with turbo decoding, new research lines have been suggested and followed, trying to apply these contributions in new decoding sceneries. Among the existing applications, the concept of decoding networks, reception systems using what is known as turbo principle or iterative processing, stands out. This work begins from a decoding network previously proposed to a serial concatenation of a convolutional coding and a differential phase shift keying (DPSK) modulator used in coherent channels. Therefore, the proposal of this thesis was the replacement of the convolutional coding by the turbo coding. This action creates, in this way, another decoding scenery allowing the use and the combination of two different forms of iterations: one due to the iterative process itself, and another inherent in turbo decoding. As the proposed decoding network is used in non-coherent channels, a modification in the detection MAP (maximum a posteriori) algorithm was suggested. In addition, a detailed description of the system in study is also presented. Simulation results indicates that certain combinations of the two iteration forms, of the decoding network, have a better trade off between performance and implementation complexity.

Agradecimentos

Gostaria, primeiramente, de agradecer a Deus por tudo o que me foi concedido e pela Sua inegável participação em minha vida.

Agradeço, também, à minha família e aos meus amigos que, mesmo distantes, souberam entender o quanto a conclusão deste trabalho significaria para mim e jamais deixaram de me apoiar. Em especial, quero lembrar de pessoas muito importantes cujo apoio, amor e incentivo tornaram-se ainda mais essenciais nesta fase de minha vida: meu pai, Inaldo, minha mãe, Irma, meu irmão, Ricardo e meu noivo, Daniel.

Ainda gostaria de manifestar a minha gratidão pela agradável convivência com o meu orientador, Jaime Portugheis, que sempre se mostrou bastante atencioso e presente, além de Hélder, com quem tive a oportunidade de compartilhar conhecimentos e idéias que se tornaram muito importantes para esta pesquisa.

Aos amigos Virgínia, Joselan e a Daniel, pela generosa acolhida, como também a Raul e Hélder, com os quais tive a oportunidade de conviver nos últimos anos.

A todos os professores que contribuíram para a minha formação profissional.

Aos amigos que conquistei em Campinas, dentre os quais, Andrezza, Flávia e Laís, pela companhia e por tornarem agradável a minha estadia nesta cidade.

Finalmente, agradeço à CAPES o apoio financeiro e a todos os funcionários e alunos da Engenharia Elétrica que, de uma maneira ou outra, me ajudaram durante a elaboração deste trabalho.

Índice

Índice

Capítulo 1

Introdução

Nos últimos tempos, o cenário das comunicações tem sido palco de grandes mudanças. As descobertas se multiplicam e impulsionam, desta forma, o surgimento de novas pesquisas. O resultado desses esforços pode ser observado em diversos segmentos, mas, particularmente, em uma área que vem se desenvolvendo, de maneira contínua, nos últimos anos: a da *Codificação de Canal*. A Codificação de Canal, também chamada de Codificação para Controle de Erros, é um método de adicionar redundância à informação, de maneira que possa ser transmitida em um canal ruidoso, possibilitando, à recepção, a detecção e correção de erros passíveis de terem ocorrido durante a transmissão. O objetivo é, portanto, reduzir o número de erros ocorridos durante o processo de transmissão.

O grande interesse por essa área teve início, praticamente, com a publicação do artigo de *Shannon*, em 1948, intitulado "*A Mathematical Theory of Communication*", onde ele uniu os campos da Teoria da Informação e da Codificação para Controle de Erros. Nesse artigo, *Shannon* introduziu o conceito de *capacidade de canal*, mostrando que, se a taxa, na qual a informação é transmitida, for menor que a capacidade do canal, existe um código de controle de erro que pode proporcionar, arbitrariamente, níveis altos de confiabilidade à informação recebida. Teoricamente, o melhor desempenho possível que qualquer canal pode atingir é chamado de *limite de Shannon*; por conseguinte, um código com desempenho igual a esse limitante é dito ser ideal. Até então, o que se tinha era a afirmação da (real possibilidade de) existência de esquemas de decodificação que atingissem esse limite, porém não se tinha idéia de como construí-los. Os cinqüenta anos seguintes foram marcados pelas tentativas de se atingir esse objetivo.

Esta busca teve um grande êxito em 1993, quando um outro acontecimento marcou a história da Codificação para Controle de Erros, por propor chances reais de atingir a promessa de *Shannon*. Foi a publicação do artigo de *Berrou, Glavieux* e *Thitimajshima*, que introduziu a *Codificação Turbo*, intitulado "*Near Shannon Limit Error Correcting Coding and Decoding: Turbo Codes*".

A Codificação Turbo é um sistema que proporciona confiabilidade extremamente alta à transmissão de dados submetida a baixas relações sinal-ruído. Ela é capaz de propiciar, a um sistema de comunicação, um desempenho muito próximo ao limite de *Shannon*, sendo considerada um passo notável em direção a uma codificação de baixa complexidade e alto ganho ([1], [3], [4], [5]). Esse tipo de codificação é constituído, basicamente, de duas inovações: *codificação com concatenação paralela* e *decodificação iterativa*.

Os codificadores de concatenação paralela (CCP) consistem de dois ou mais codificadores componentes, para códigos convolucionais ou de bloco (sendo os primeiros mais comuns), interligados, entre si, por um *interleaver* (entrelaçador). Considerando, para efeitos ilustrativos, uma concatenação paralela de dois codificadores, ela funciona da seguinte maneira: o bloco de informação u é codificado, utilizando o primeiro codificador componente; em seguida, esse mesmo bloco u é entrelaçado pelo *interleaver* e é codificado pelo segundo codificador componente. A palavra-código resultante é formada, portanto, pela palavra-código obtida do primeiro codificador e pela paridade da palavra-código obtida do segundo multiplexadas. Este resultado é, então, transmitido, pelo canal, até o receptor.

A decodificação iterativa é uma alternativa de decodificação próxima da ótima que proporciona um bom desempenho, enquanto requer, apenas, modestos níveis de complexidade, resultando, assim, em um sistema computacionalmente eficiente. Ela utiliza um decodificador, com decisão suave, para cada codificador componente do CCP. Os decodificadores agem nos dados de maneira a obter estimações do bloco de informação que são, então, utilizados pelo outro decodificador. Essas estimações são trocadas entre os decodificadores componentes, à medida que são utilizadas por eles, funcionando, assim, como níveis de refinamento da informação recebida. Já que esses decodificadores componentes operam com cada saída "não-completamente decodificada" dos outros decodificadores, eles lembram o princípio de um motor turbo. Daí, o nome "Códigos Turbo". É importante ressaltar que é ao mecanismo de decodificação que é dado esse nome e, não, aos códigos em si.

É interessante observar que, mesmo se considerado uma aparição repentina, os códigos turbo são o resultado de uma intuição inteligente, montada em diversos conceitos já disponíveis de antemão [9].

Tendo em vista as promessas de desempenho cada vez melhores e todas as vantagens já conhecidas, é compreensível o fato de a codificação turbo estar sendo considerada um dos

tópicos mais relevantes na comunicação de dados. Ela tem sido lembrada para atuar em, praticamente, todas as aplicações de comunicação e armazenamento de dados, tais como terceira geração da telefonia celular, canais de satélite, acesso à internet banda larga, armazenamento óptico e magnético de dados e transmissão digital de vídeo. Em especial, pode-se destacar as comunicações sem fio, que vêm se tornando cada vez mais populares devido ao baixo preço, ao desenvolvimento dos serviços prestados e à miniaturização dos aparelhos. O uso de codificadores mais poderosos, como o Turbo, vem se solidificando, portanto, como uma alternativa real a uma melhora de desempenho – propiciando, assim, a utilização de novos serviços e permitindo, ainda, um aumento de capacidade – especialmente nos sistemas 3G (Terceira Geração).

Esse breve histórico teve por objetivo descrever o cenário que, inicialmente, motivou esse trabalho, aquele que apresenta os códigos turbo como uma importante inovação, realçando suas inegáveis contribuições.

Impulsionadas pelos notáveis resultados, amplamente citados na literatura, alcançados com o uso da decodificação turbo, novas linhas de pesquisa vêm sendo sugeridas e seguidas, partindo, desse princípio, visando aplicar estas contribuições em novos cenários de decodificação. Entre as aplicações existentes, destaca-se o conceito de redes de decodificação, sistemas de recepção que utilizam o que se chamou de princípio turbo ou processamento iterativo. Vale mencionar o artigo tutorial [11] onde o termo rede de codificação/decodificação foi, pela primeira vez, utilizado.

Entre as redes de decodificação sugeridas, uma delas tornou-se o ponto de partida para essa pesquisa: aquela proposta por *Hoeher* e *Lodge* [6] para a concatenação serial de um codificador convolucional e um modulador diferencial PSK (DPSK) em *canais coerentes*. Aqui, a idéia foi efetuar a substituição do codificador convolucional por um codificador turbo, criando, dessa forma, um cenário de decodificação diferente daquele anterior, que propicia a utilização e combinação de duas formas diferentes de iteração: uma devido ao próprio processo iterativo, e outra inerente à decodificação turbo. Esse trabalho analisa como essas formas de iteração devem ser combinadas, de maneira que possam favorecer o desempenho do sistema. Até onde é de nosso conhecimento, essa é a primeira vez que uma combinação de duas formas diferentes de iteração, na rede de decodificação, é proposta e analisada. Além disso, o fato de se considerar

canais não-coerentes resultou na necessidade de se sugerir uma modificação do algoritmo de detecção MAP (Máximo *a posteriori*), utilizado na rede de decodificação.

Muito recentemente, ocorreu uma explosão de interesse no projeto de sistemas concatenados de modulação diferencial com decodificação iterativa (veja [5] e suas inúmeras referências). Talvez, uma das razões para isso resida no fato que os mesmos tenham demonstrado melhor desempenho, mesmo em canais coerentes, quando comparados a sistemas sem modulação diferencial de mesma complexidade [10]. Algumas modificações do algoritmo de detecção MAP, para uso em canais não-coerentes, foram propostas na literatura; entretanto, acreditamos que a modificação descrita nesse trabalho (capítulo 4) seja distinta dessas propostas, pelo fato da sugestão ter sido feita de maneira independente.

A dissertação está organizada de forma que, no capítulo 2, é feito um estudo dos códigos convolucionais, além de uma descrição do tipo de modulação empregada no sistema proposto. No capítulo 3, por sua vez, é feita uma abordagem, de maneira detalhada, do funcionamento da codificação/decodificação turbo que é o ponto chave do sistema em estudo. Em seguida, o capítulo 4 descreve a rede proposta, através dos blocos constituintes, explicando-a, pormenorizadamente, o que, em geral, não é feito na literatura; logo depois, curvas de desempenho são mostradas, permitindo, portanto, a análise completa da rede de decodificação. Por fim, no capítulo 5, são feitos comentários com algumas sugestões de trabalhos futuros.

Capítulo 2

Conceitos Básicos

Este capítulo faz um estudo dos códigos convolucionais sob vários aspectos: desde a codificação, passando pelas estruturas básicas de representação, até alguns métodos possíveis de serem utilizados na decodificação. Adicionalmente, é feita uma descrição do tipo de modulação utilizada no sistema proposto do capítulo 4.

2.1. Códigos Convolucionais

Os códigos convolucionais foram primeiramente introduzidos por P. Elias, em 1955, como uma alternativa aos códigos de bloco. Nestes últimos, o codificador recebe um bloco de informação de k bits e gera uma palavra código de n bits, tudo se passando, portanto, numa base bloco a bloco. A diferença dos códigos convolucionais, em relação aos de bloco, é que, neles, o codificador contém memória e as suas n saídas, numa dada unidade de tempo, dependem, não apenas das k entradas, naquela mesma unidade de tempo, mas também dos m blocos prévios de entrada. Um codificador convolucional (n, k, m) pode ser implementado, portanto, como um circuito seqüencial de k entradas, n saídas e memória de tamanho m.

A Fig. 2.1 exemplifica um codificador para o código convolucional binário (2, 1, 3). O codificador consiste de um registrador de deslocamento de 3 estágios de *flip-flop* ($m = 3$) com dois adicionadores módulo 2 ($n = 2$), que podem ser implementados como portas lógicas ou-exclusivo, e um multiplexador responsável por tornar serial a saída x formada conjuntamente por $x^{(1)}$ e $x^{(2)}$.

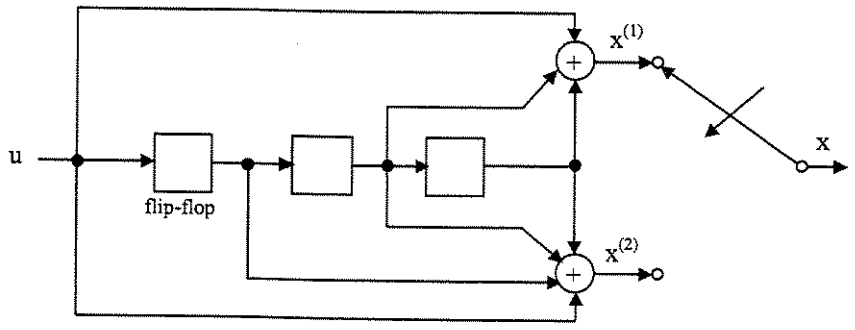

Fig. 2.1 – Exemplo de codificador convolucional binário para código (2, 1, 3).

Algumas definições são utilizadas para caracterizar um codificador convolucional:

Ordem da memória (m): Dado um codificador de k entradas, ele contem k registradores de deslocamento com comprimentos que podem ser diferentes. Sendo K_i o comprimento do i-ésimo registrador, a ordem da memória será definida como,

$$m \triangleq \max_{1 \le i \le k} K_i \tag{2.1}$$

No exemplo da Fig. 2.1, o codificador possui uma entrada (k = 1). Assim, há, apenas, um registrador de comprimento $K_1 = 3$ originando, portanto, uma memória de ordem $m = 3$.

Comprimento de restrição (n_A): Define o número de deslocamentos sob o qual um único bit de informação pode influenciar a saída codificada, ou, em outras palavras, o número máximo de saídas codificadas que podem ser influenciadas por um único bit de informação. Isso acontece porque cada bit de informação requer $m + 1$ unidades de tempo (ou deslocamentos) para entrar e, finalmente, sair do registrador, e, durante cada deslocamento, pode afetar qualquer uma das n saídas codificadas (dependendo das conexões dos registradores de deslocamento)

$$n_A \triangleq n(m+1) \tag{2.2}$$

Taxa do codificador (R): Representa a relação entre os n bits codificados, gerados pelo codificador convolucional, para cada k bits de informação na sua entrada.

$$R = \frac{k}{n} \tag{2.3}$$

Entretanto, para uma seqüência de informação de comprimento finito $k \cdot L$, onde L é o comprimento da seqüência de informação, a palavra código correspondente tem comprimento $n(L + m)$ e, a rigor, a taxa será expressa pela eq. 2.4,

$$R = \frac{kL}{n(L+m)} \tag{2.4}$$

onde as últimas $n \cdot m$ saídas são geradas depois do último bloco de informação, não nulo, ter entrado no codificador. Em outras palavras, uma seqüência de informação é terminada com blocos todos nulos de forma a permitir que a mémoria do codificador possa ser limpada ou "zerada" (para não influenciar a próxima utilização do codificador). A esta seqüência de terminação, formada por m zeros, é dado o nome de *tail* bits.

Observando a eq. 2.4, se $L >> m$, então

$$\frac{L}{(L+m)} \approx 1 \tag{2.5}$$

e as taxas do codificador de bloco e do codificador convolucional tornam-se, aproximadamente, iguais (eq. 2.3). Esta aproximação representa o modo normal de operação para os códigos convolucionais. Por outro lado, se L é um número pequeno, a razão, na eq. 2.4, que é a taxa efetiva de transmissão de informação, se reduziria abaixo da taxa do codificador por uma quantidade fracionária denominada "perda de taxa fracionária", que seria uma espécie de medida de perda de eficiência do código:

$$\frac{k/n - kL/n(L+m)}{k/n} = \frac{m}{L+m} \tag{2.6}$$

Assim, para manter pequena essa perda (próxima do zero), é interessante assumir $L >> m$.

2.1.1. Codificação dos códigos convolucionais

O codificador da Fig. 2.1 será utilizado para ilustrar a teoria dos codificadores convolucionais seguindo dois caminhos: (a) uma descrição no domínio do tempo, e, em seguida, (b) uma outra no domínio da transformada.

(a) Descrição no domínio do tempo

O comportamento, no domínio do tempo, de um código convolucional de taxa $R = 1/n$, pode ser definido em termos de um conjunto de n *respostas ao impulso*. Observando,

portanto, o codificador da Fig. 2.1, que tem taxa $R = ½$, vê-se a necessidade de duas respostas ao impulso para caracterizar seu comportamento. *A* seqüência de informação $u = (u_0, u_1, u_2,...)$ entra no codificador bit a bit. As duas seqüências de saídas codificadas, $x^{(1)} = (x_0^{(1)}, x_1^{(1)}, x_2^{(1)},...)$ e $x^{(2)} = (x_0^{(2)}, x_1^{(2)}, x_2^{(2)},...)$, podem ser obtidas através da execução de convoluções da seqüência de entrada u com a resposta ao impulso associada a cada saída. A determinação das respostas ao impulso é feita observando as duas seqüências de saída do codificador que são produzidas em resposta à seqüência de entrada $u = (1, 0, 0,...)$ ou, mais diretamente, observando as conexões dos registradores de deslocamento relativas a cada uma das saídas. Considerando que o codificador tem uma memória de m unidades de tempo, as respostas ao impulso podem durar, no máximo, $m + 1$ dessas unidades e são escritas como $g^{(1)} = (g_0^{(1)}, g_1^{(1)},..., g_m^{(1)})$ *e* $g^{(2)} = (g_0^{(2)}, g_1^{(2)},..., g_m^{(2)})$. No exemplo da Fig. 2.1, $g^{(1)} = (1\ 0\ 1\ 1)$ e $g^{(2)} = (1\ 1\ 1\ 1)$. As respostas ao impulso $g^{(1)}$ e $g^{(2)}$ são ditas *seqüências geradoras* do código.

As equações de codificação são, então, escritas como:

$$x^{(j)} = u * g^{(j)} \qquad j = 1, 2 \tag{2.7}$$

onde * indica convolução discreta, j é o número de saídas do codificador e todas as operações são módulo 2. A operação de convolução resulta nas saídas codificadas,

$$x_t^{(j)} = \sum_{l=0}^{m} g_l^{(j)} u_{t-l} \qquad \begin{matrix} t = 0, 1, 2,... \\ j = 1, 2 \end{matrix} \tag{2.8}$$

onde $u_{t-l} = 0$ para todo $l > t$.

Após a codificação, as duas seqüências de saída são multiplexadas em uma única seqüência, denominada "palavra-código", que será transmitida no canal:

$$x = (x_0^{(1)} x_0^{(2)}, x_1^{(1)} x_1^{(2)}, x_2^{(1)} x_2^{(2)}, \ldots) \tag{2.9}$$

Uma outra maneira de representar a equação de codificação é através da forma matricial. Se as seqüências geradoras $g^{(1)}$ e $g^{(2)}$ forem entrelaçadas e dispostas numa matriz

$$G = \begin{bmatrix} g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & g_2^{(1)}g_2^{(2)} & \cdots & g_m^{(1)}g_m^{(2)} & & \\ & g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & \cdots & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} & \\ & & g_0^{(1)}g_0^{(2)} & \cdots & g_{m-2}^{(1)}g_{m-2}^{(2)} & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} \\ & & & \ddots & \ddots & & \ddots \end{bmatrix} \tag{2.10}$$

onde as áreas brancas representam zeros, a equação pode ser reescrita, em forma matricial, como:

$$x = uG \tag{2.11}$$

onde todas as operações são módulo 2. G é dita "matriz geradora" do código. Analisando-se a matriz, nota-se que cada linha de G é idêntica à anterior, a menos de um deslocamento de $n = 2$ espaços para a direita, e que G é semi-infinita, correspondendo ao fato de que a seqüência de informação u possui um comprimento arbitrário.

Até então, o texto baseou-se, unicamente, no codificador da Fig. 2.1, que continha, apenas, uma entrada ($k = 1$), para o desenvolvimento das explicações sobre a codificação no domínio do tempo. O número de entradas de um codificador, porém, não necessariamente é igual a um e, assim, pequenas modificações devem ser feitas visando à generalização da equação de codificação. De forma geral, a seqüência de entrada u pode ser vista como

$$u = (u_0^{(1)}\, u_0^{(2)} \ldots u_0^{(k)}, u_1^{(1)}\, u_1^{(2)} \ldots u_1^{(k)}, \ldots) \tag{2.12}$$

onde k é o número de entradas do codificador. O número de seqüências geradoras equivale à quantidade de combinações entre as k entradas e n saídas do codificador convolucional:

$$g_i^{(j)} = (g_{i,0}^{(j)}, g_{i,1}^{(j)}, \ldots, g_{i,m}^{(j)}) \tag{2.13}$$

onde $g_i^{(j)}$ representa a seqüência geradora correspondente à entrada i e à saída j. Dessa forma, a matriz G será formada como

$$G = \begin{bmatrix} g_{1,0}^{(1)} \cdots g_{1,0}^{(n)} & g_{1,1}^{(1)} \cdots g_{1,1}^{(n)} & \cdots & g_{1,m}^{(1)} \cdots g_{1,m}^{(n)} & \\ \vdots & \vdots & \vdots & \vdots & \\ g_{k,0}^{(1)} \cdots g_{k,0}^{(n)} & g_{k,1}^{(1)} \cdots g_{k,1}^{(n)} & \cdots & g_{k,m}^{(1)} \cdots g_{k,m}^{(n)} & \\ & g_{1,0}^{(1)} \cdots g_{1,0}^{(n)} & \cdots & g_{1,m-1}^{(1)} \cdots g_{1,m-1}^{(n)} & g_{1,m}^{(1)} \cdots g_{1,m}^{(n)} \\ & \vdots & \vdots & \vdots & \vdots \\ & g_{k,0}^{(1)} \cdots g_{k,0}^{(n)} & \cdots & g_{k,m-1}^{(1)} \cdots g_{k,m-1}^{(n)} & g_{k,m}^{(1)} \cdots g_{k,m}^{(n)} \\ & \ddots & & \ddots & \ddots \end{bmatrix} \tag{2.14}$$

Na forma generalizada, nota-se que cada conjunto de k linhas é idêntico ao conjunto anterior, a menos de um deslocamento de n espaços para a direita. Para a seqüência de informação u, dada pela eq. 2.12, a palavra código

$$x = (x_0^{(1)}\, x_0^{(2)} \ldots x_0^{(n)}, x_1^{(1)}\, x_1^{(2)} \ldots x_1^{(n)}, \ldots) \tag{2.15}$$

é ainda dada pela eq. 2.11. Portanto, observando que x é uma combinação linear de linhas da matriz geradora G, um código convolucional *(n, k, m)* é dito ser um código linear.

(b) Descrição no domínio da transformada

Nos sistemas lineares, operações envolvendo convoluções, no domínio do tempo, podem ser substituídas por outras mais convenientes, no domínio da transformada, envolvendo multiplicações polinomiais. Isto é possível porque, do estudo da teoria dos filtros lineares, as integrais de convolução, que descrevem a operação de filtragem linear no domínio do tempo, são substituídas pela multiplicação de transformadas de Fourier no domínio da freqüência. Sabendo-se que um codificador convolucional é um sistema linear, pode-se, portanto, simplificar a obtenção das saídas codificadas, aplicando-se transformações apropriadas.

Na representação polinomial de uma seqüência binária, esta é representada pelos coeficientes do polinômio. Assim, dado um codificador convolucional arbitrário (por exemplo, aquele da Fig. 2.1), a resposta ao impulso de cada caminho (referente a cada saída) pode ser substituída por um polinômio cujos coeficientes são representados pelos respectivos elementos da própria resposta ao impulso, ou seja,

$$g^{(j)}(D) = g_0^{(j)} + g_1^{(j)} D + \ldots + g_m^{(j)} D^m \qquad j = 1, 2 \qquad (2.16)$$

onde j representa a j-ésima saída do codificador e $g_0^{(j)}$, $g_1^{(j)}$, ..., $g_m^{(j)}$ são os elementos da resposta ao impulso do caminho j. A variável D pode ser interpretada como um "operador de unidade de atraso" e sua potência, l, define o número de unidades de tempo pelo qual o bit associado, na resposta ao impulso, é atrasado com respeito ao primeiro bit, $g_0^{(j)}$. Os polinômios $g^{(j)}(D)$ são ditos "polinômios geradores" do código. Outra forma de obtê-los é através da observação direta do diagrama do circuito. Já que cada estágio do registrador de deslocamento representa um atraso de uma unidade de tempo, a seqüência de conexões (com um "1" representando uma conexão e um "0", a ausência de conexão) de um registrador para uma dada saída é a seqüência de coeficientes do correspondente polinômio gerador. Para o codificador da Fig. 2.1, as seqüências de conexões do primeiro e segundo registradores, com respeito, respectivamente, à primeira e segunda saídas são, como visto, $g^{(1)} = (1\ 0\ 1\ 1)$ e $g^{(2)} = (1\ 1\ 1\ 1)$. Por conseguinte, os correspondentes

polinômios geradores são $g^{(1)}(D) = 1 + D^2 + D^3$ e $g^{(2)}(D) = 1 + D + D^2 + D^3$. Nota-se que $l = 0, 1, ..., m$; $l = 0$ corresponde ao estágio mais à esquerda e $l = m$, àquele mais à direita.

Logo, a equação de codificação pode ser vista na eq. 2.17, de forma análoga àquela apresentada no sub-item (a):

$$x^{(j)}(D) = u(D)g^{(j)}(D) \qquad j = 1, 2 \tag{2.17}$$

onde $u(D)$ e $x^{(j)}(D)$ correspondem, respectivamente, à seqüência de informação e às seqüências codificadas:

$$u(D) = u_0 + u_1 D + u_2 D^2 + ... \tag{2.18}$$

$$x^{(j)}(D) = x_0^{(j)} + x_1^{(j)} D + x_2^{(j)} D^2 + ... \tag{2.19}$$

Após a multiplexação,

$$x(D) = x^{(1)}(D^2) + Dx^{(2)}(D^2) \tag{2.20}$$

Partindo dessas explicações prévias, onde foi utilizado, como ilustração, o codificador da Fig. 2.1, com $k = 1$ e $n = 2$, pode-se, finalmente, representar a equação de codificação de uma maneira mais geral, seguindo o mesmo raciocínio, para um codificador de k entradas e n saídas. Novamente, partindo do princípio de que o codificador é um sistema linear, e que $u^{(i)}(D)$ representa a i-ésima seqüência de entrada e $x^{(j)}(D)$, a j-ésima seqüência de saída, o polinômio gerador $g_i^{(j)}(D)$ pode ser interpretado como uma função de transferência, relacionando a entrada i e a saída j. Assim, para este sistema, tem-se um total de $k \cdot n$ funções de transferência, que podem ser representadas por uma "matriz função de transferência":

$$G(D) = \begin{bmatrix} g_1^{(1)}(D) & g_1^{(2)}(D) & \dots & g_1^{(n)}(D) \\ g_2^{(1)}(D) & g_2^{(2)}(D) & \dots & g_2^{(n)}(D) \\ \vdots & \vdots & \vdots & \vdots \\ g_k^{(1)}(D) & g_k^{(2)}(D) & \dots & g_k^{(n)}(D) \end{bmatrix} \tag{2.21}$$

Fazendo uso de $G(D)$, a forma generalizada da equação de codificação pode ser expressa por:

$$X(D) = U(D)G(D) \tag{2.22}$$

onde $U(D)$ representa a k-upla seqüência de entrada e $X(D)$ são as n-uplas seqüências codificadas, como na eq. 2.23.

$$U(D) \triangleq [u^{(1)}(D), u^{(2)}(D), ..., u^{(k)}(D)]$$
$$X(D) \triangleq [x^{(1)}(D), x^{(2)}(D), ..., x^{(n)}(D)] \quad (2.23)$$

Após codificadas, as palavras código são multiplexadas para a transmissão no canal:

$$x(D) = x^{(1)}(D^n) + Dx^{(2)}(D^n) + ... + D^{n-1}x^{(n)}(D^n) \quad (2.24)$$

2.1.2. Propriedades estruturais dos codificadores convolucionais

Tradicionalmente, as propriedades estruturais dos codificadores convolucionais são propostas na forma de gráficos e utilizam um dos três diagramas equivalentes: árvore do código, diagrama de estados e treliça. Por uma questão mais didática e para evitar uma repetição entediante de explicações equivalentes, o texto ater-se-á aos dois últimos tipos, por revelarem, mais explicitamente, o fato do codificador convolucional poder ser visto como uma máquina de estados finitos.

(a) Diagrama de estados

Já que o codificador convolucional é um circuito seqüencial, sua operação pode ser descrita por um diagrama de estados. O estado do codificador é definido como sendo o conteúdo de seu registrador de deslocamento, da seguinte maneira: para um código genérico *(n, k, m)* com $k > 1$, o i-ésimo registrador de deslocamento contém K_i bits prévios de informação. Definindo,

$$K \triangleq \sum_{i=1}^{k} K_i \quad (2.25)$$

como sendo a "memória total do codificador", o estado do codificador no instante de tempo t (quando $u_t^{(1)}u_t^{(2)}...u_t^{(k)}$ são as entradas do codificador) é a K-upla binária de entradas

$$(u_{t-1}^{(1)}u_{t-2}^{(1)}...u_{t-K_1}^{(1)} \quad u_{t-1}^{(2)}u_{t-2}^{(2)}...u_{t-K_2}^{(2)} \quad \cdots \quad u_{t-1}^{(k)}u_{t-2}^{(k)}...u_{t-K_k}^{(k)}) \quad (2.26)$$

e existem um total de 2^K diferentes estados possíveis. Para o codificador da Fig. 2.1, por exemplo, $K = K_1 = 3$. Portanto, possui um total de 8 estados, sendo o estado do codificador no instante t simplesmente $(u_{t-1}\ u_{t-2}\ u_{t-3})$.

Cada novo bloco de k entradas causa uma transição para um novo estado. Em conseqüência, há 2^k ramos, deixando cada estado, cada qual correspondendo a um diferente bloco

de entrada. A cada ramo, são associadas as k entradas que causam a transição $(u_t^{(1)}\ u_t^{(2)}\ \ldots\ u_t^{(k)})$ e as n correspondentes saídas $(x_t^{(1)}\ x_t^{(2)}\ \ldots\ x_t^{(n)})$. Aos estados, são associados os símbolos S_0, S_1, ..., $S_{2^{K-1}}$, onde S_i é a representação decimal dos possíveis estados do codificador. A Fig. 2.2 mostra, como um exemplo, o diagrama de estados do codificador da Fig. 2.1.

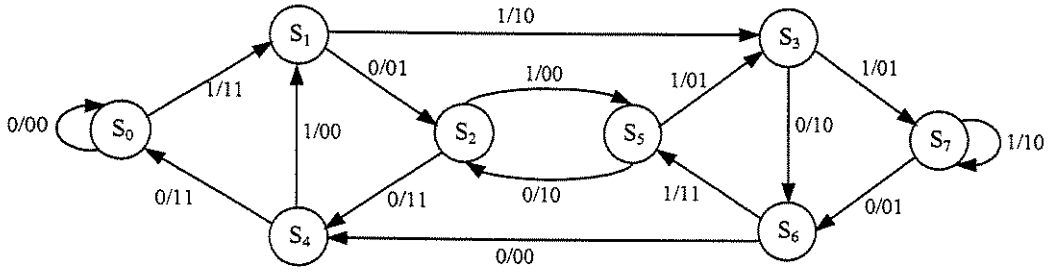

Fig. 2.2 – Diagrama de estados do codificador para o código (2, 1, 3)

Assumindo que o codificador encontra-se, inicialmente, no estado S_0 (estado nulo), a palavra código, correspondente a qualquer seqüência de informação, pode ser obtida seguindo o caminho, no diagrama de estados, determinado por aquela seqüência e observando as saídas associadas aos ramos especificados por aquele caminho. Após o último bloco de informação não nula, é anexada uma seqüência de informação de m zeros responsável por fazer o codificador retornar ao estado nulo.

(b) Treliça

Uma outra forma de representar um codificador convolucional é através de uma treliça. Ela age de maneira a expandir o diagrama de estados no tempo, isto é, representando cada instante de tempo com um diagrama de estados separado, sendo, portanto, apenas um outro meio de visualização dos estados do codificador. Essa estrutura será útil na seção 2.3, que trata da decodificação. A Fig. 2.3 mostra a treliça para um código *(2, 1, 2)* com matriz função de transferência dada por $G(D) = [1 + D + D^2,\ 1 + D^2]$ e seqüência de informação de comprimento arbitrário L.

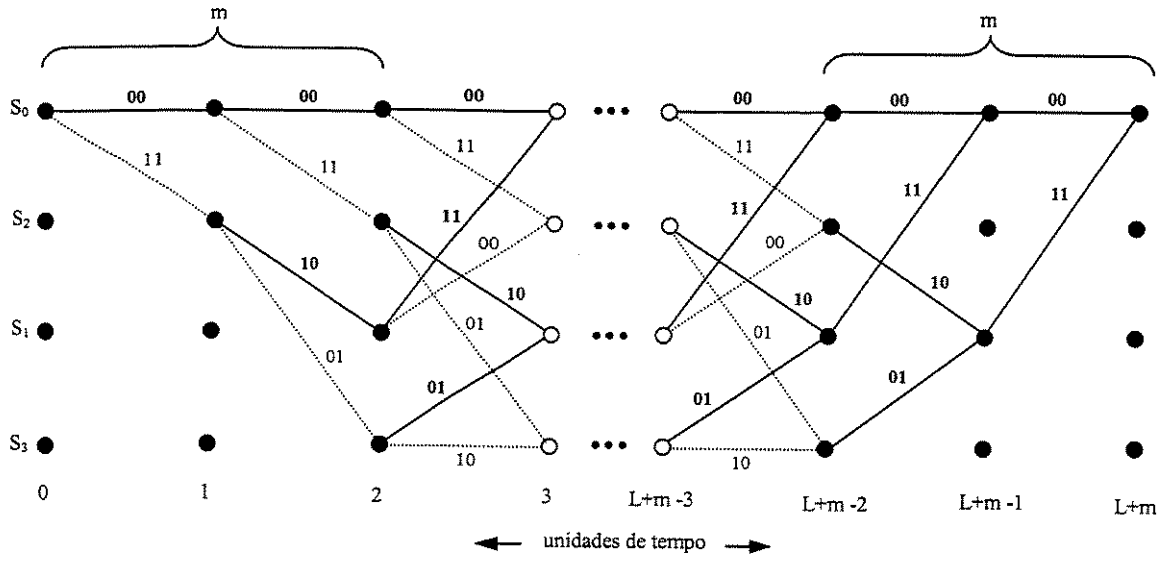

Fig. 2.3 – Treliça para um código (2, 1, 2)

O código do exemplo possui uma ordem de memória $m = 2$ e 1 entrada $(k = 1)$, por isso é representado por um total de 4 estados. Os ramos cheios, na figura, indicam uma entrada "0", enquanto que os tracejados indicam uma entrada "1". A treliça contém $L + m + 1$ unidades de tempo ou níveis. Assumindo que o codificador sempre inicia suas atividades no estado S_0 e, no final da codificação, retorna a ele, os primeiros m níveis correspondem à partida do estado S_0 e os últimos m níveis, ao retorno a este estado, como pode ser visto na Fig. 2.3. Conclui-se, portanto, que nem todos os estados são atingidos nos primeiros e nos últimos m níveis. Entretanto, na porção central da treliça, todos os estados são possíveis de serem atingidos e, em cada unidade de tempo, há uma réplica do diagrama de estados do código. Cada ramo da treliça indica a saída codificada referente à entrada naquela unidade de tempo. No caso da Fig. 2.3, $x_t = (x_t^{(1)}\ x_t^{(2)})$, onde t representa o nível atual. Logo, tal qual visto para o diagrama de estados, dada uma seqüência de entrada, é possível determinar a seqüência de saída codificada, simplesmente observando o caminho, na treliça, descrito por aquela dada seqüência de entrada e lendo as saídas correspondentes a cada ramo do caminho. A única diferença na "leitura" dos dois diagramas é que, no segundo, não há por que se anexar uma seqüência de m zeros, na entrada, como é feito para o diagrama de estados, uma vez que a própria estrutura da treliça já comporta essa necessidade de retorno ao estado nulo S_0.

Uma propriedade interessante dos diagramas de estados é a capacidade de prover uma completa descrição dos pesos de *Hamming* de todas as palavras código não nulas, isto é, fornecer uma função distribuição de pesos para o código. Com este intuito, o diagrama de estados pode ser modificado da seguinte forma: o estado S_0 é dividido em um estado inicial e outro final; o laço, envolvendo o próprio estado, é apagado, e cada ramo é rotulado com um "ganho de ramo" X^i, onde i é o peso dos n bits codificados pertencentes àquele ramo. Cada caminho, conectando o estado inicial ao final, representa uma palavra-código não nula que sai e entra em S_0, apenas, uma vez. Aquelas palavras-código que saem e entram em S_0 mais de uma vez são consideradas uma seqüência de palavras-código curtas. O "ganho de caminho" é o produto dos ganhos de ramo, ao longo de um caminho, e o peso da palavra-código associada é a potência de X resultante no ganho de caminho. A Fig. 2.4 é o diagrama da Fig. 2.2 modificado. Como um exemplo da obtenção dos ganhos de ramo e de caminho, a partir da Fig. 2.2, pode-se observar que o ramo que conecta os estados S_4 e S_0 possui os $n = 2$ bits codificados iguais a "1 1", totalizando um peso de *2*; portanto, na Fig. 2.4, o ganho para aquele ramo será X^2. Observando, na Fig. 2.4, a seqüência de estados S_0 S_1 S_3 S_7 S_6 S_4 S_0 , tem-se que o ganho de caminho será $X^2 \cdot X^1 \cdot X^1 \cdot X^1 \cdot 1 \cdot X^2 = X^7$ e a palavra-código correspondente terá peso 7. O digrama modificado permite, portanto, que o peso de uma determinada palavra-código possa ser encontrado.

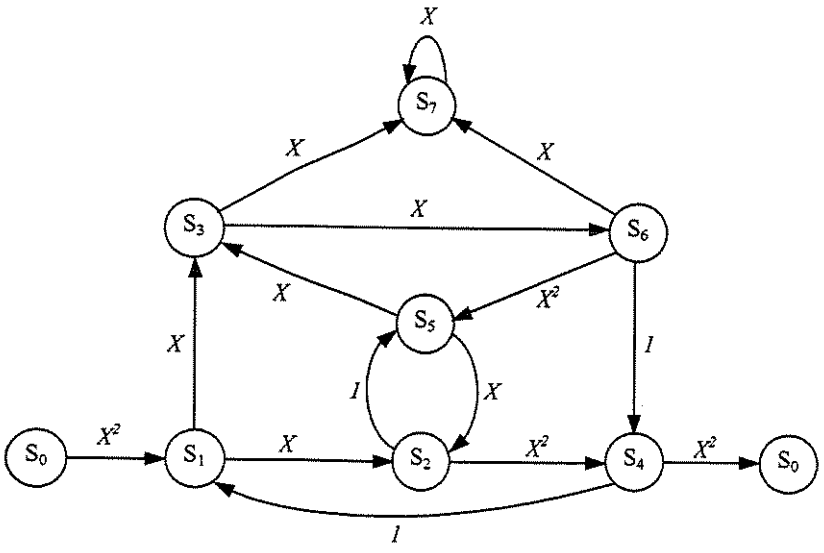

Fig. 2.4 – Diagrama de estados modificado para o código (2, 1, 3)

Partindo-se desses conceitos, pode-se reuni-los, para determinar a função distribuição de pesos do código (eq. 2.27), que permite uma noção geral de todos os pesos possíveis para aquele código. Esta função é determinada, considerando o diagrama de estados modificado como um gráfico de fluxo de sinais e aplicando a fórmula de ganho de Mason (eq. 2.28).

$$T(X) = \sum_i A_i X^i \tag{2.27}$$

onde A_i é o número de palavras-código de peso i. Para a utilização da fórmula de ganho de Mason, são necessárias algumas definições:

caminho direto: é o caminho, ligando o estado inicial ao final que não passa duas vezes por um mesmo estado.

laço: é um caminho fechado, iniciando em qualquer estado e retornando a ele sem passar duas vezes por qualquer outro estado.

laços que não se tocam: é um conjunto de laços onde nenhum estado pertence a mais de um laço no conjunto.

Assim, a fórmula de Mason, utilizada para o cálculo da função distribuição de pesos, pode ser escrita na forma:

$$T(X) = \frac{\sum_i F_i \Delta_i}{\Delta} \tag{2.28}$$

onde F_i é o ganho do i-ésimo caminho direto e Δ é definido como:

$$\Delta = 1 - \sum_i C_i + \sum_{i',j'} C_{i'} C_{j'} - \sum_{i'',j'',l''} C_{i''} C_{j''} C_{l''} + \cdots \tag{2.29}$$

onde C_i é o ganho do i-ésimo laço (obtido de maneira análoga ao ganho de caminho no diagrama de estados modificado); $\{i\}$ é o conjunto de todos os laços; $\{i', j'\}$ é o conjunto de todos os pares de laços que não se tocam; $\{i'', j'', l''\}$ é o conjunto de todos os trios de laços que não se tocam e assim sucessivamente. O segundo termo da eq. 2.29 representa a soma dos ganhos de laço; o terceiro termo é o produto dos ganhos de laço de dois laços que não se tocam, somados sobre todos os pares de laços que não se tocam e assim por diante. Por último, Δ_i é definido, exatamente, como Δ, porém refere-se, apenas, àquela porção do gráfico que não toca o i-ésimo caminho direto, ou seja, todos os estados que compõem o i-ésimo caminho direto, juntamente com todos os ramos conectados a estes estados, são removidos do gráfico no cálculo de Δ_i. A

soma, no numerador da eq. 2.28, é em relação a todos os caminhos diretos. Assim, após *T(X)* ser calculado, segundo a eq. 2.28, terá a forma da eq. 2.27 e a distribuição de pesos do código pode ser obtida pela simples leitura da fórmula.

Informações adicionais, sobre a estrutura geral do código, podem ser obtidas, utilizando o mesmo procedimento anterior, apenas, tomando uma versão "aumentada" do diagrama de estados modificado que rotula cada ramo com mais duas definições, além do ganho de ramo X^i: Y^j, onde j é o peso dos k bits de informação daquele ramo, e Z para indicar o ramo. Logo, a função distribuição de pesos será ampliada para

$$T(X,Y,Z) = \sum_{i,j,l} A_{i,j,l} X^i Y^j Z^l \tag{2.30}$$

onde o coeficiente $A_{i,\,j,\,l}$ descreve o número de palavras-código com peso i, cuja seqüência de informação associada tem peso j e cujo comprimento é de l ramos e essa equação é o objetivo do cálculo do ganho de Mason (eq. 2.28) exatamente como era a eq. 2.27, só que, agora, de uma maneira mais completa.

2.1.3. Propriedades de distância dos códigos convolucionais

As propriedades de distância de um código ou de um codificador determinam a capacidade que eles possuem de detectar e corrigir erros e são úteis na análise do desempenho de várias técnicas de decodificação. Muitos dos parâmetros de distância são, inclusive, úteis na seleção de bons codificadores.

A mais importante medida de distância, para um código convolucional, é a "distância *livre*" mínima, d_{free}, que representa a menor distância entre quaisquer duas palavras do código, definida por

$$d_{free} \triangleq \min\{d(x', x'') : u' \neq u''\} \tag{2.31}$$

onde x' e x'' são as palavras-código correspondentes às seqüências de informação u' e u'', respectivamente, que são assumidas, como tendo comprimentos distintos; para que as palavras-código tenham mesmo comprimento, são adicionados zeros à seqüência mais curta. Já que o código convolucional é linear,

$$d_{free} = \min\{w(x'+x'') : u' \neq u''\} = \min\{w(x) : u \neq 0\} = \min\{w(uG) : u \neq 0\} \tag{2.32}$$

onde x é a palavra código correspondente à seqüência de informação u e w, o seu peso . Como resultado, d_{free}, também pode ser visto como o peso mínimo da palavra código, de qualquer comprimento, produzida por uma seqüência de informação não nula. Em termos do diagrama de estados normal e modificado, pode ser vista, respectivamente, como o peso mínimo de todos os caminhos que saem e entram no estado S_0 e como a menor potência de X em (2.27) ou (2.30). Para o código da Fig. 2.1, por exemplo, pode-se observar o diagrama de estados da Fig. 2.2 e encontrar $d_{free} = 6$. Ainda, por definição, a capacidade máxima de correção de erros de um código ou codificador, t_{free}, é dada por $t_{free} = \lfloor (d_{free} - 1)/2 \rfloor$, onde $\lfloor a \rfloor$ é a parte inteira de a.

Enquanto a distância livre é um parâmetro do código convolucional, outra medida importante é a "função distância de coluna", *FDC*, que é um parâmetro do codificador. Seja

$$[x]_i = (x_0^{(1)} x_0^{(2)} \ldots x_0^{(n)}, x_1^{(1)} x_1^{(2)} \ldots x_1^{(n)}, \ldots, x_i^{(1)} x_i^{(2)} \ldots x_i^{(n)}) \tag{2.33}$$

a i-ésima "truncagem" da palavra código x e

$$[u]_i = (u_0^{(1)} u_0^{(2)} \ldots u_0^{(k)}, u_1^{(1)} u_1^{(2)} \ldots u_1^{(k)}, \ldots, u_i^{(1)} u_i^{(2)} \ldots u_i^{(k)}) \tag{2.34}$$

a i-ésima "truncagem" da seqüência de informação u. A *FDC* de ordem i, d_i, é definida como

$$d_i \triangleq \min\{d([x']_i, [x'']_i) : [u']_0 \neq [u'']_0\} = \min\{w[x]_i : [u]_0 \neq 0\} \tag{2.35}$$

onde x é a palavra-código correspondente à seqüência de informação u. Dessa forma, d_i é a palavra-código de peso mínimo, em relação às primeiras $(i + 1)$ unidades de tempo, cujo bloco de informação inicial é não nulo. Em outras palavras, d_i é o peso da subseqüência codificada de menor peso resultante de uma subseqüência de informação com o primeiro bloco não nulo.

Este parâmetro é útil na determinação da capacidade de correção de erros de um codificador truncado $\lfloor G \rfloor_i$, representado pelas matrizes abaixo, para $i \leq m$ e $i > m$ respectivamente.

$$[G]_i = \begin{bmatrix} G_0 & G_1 & \cdots & G_i \\ & G_0 & \cdots & G_{i-1} \\ & & \ddots & \vdots \\ & & & G_0 \end{bmatrix}$$

$$[G]_i = \begin{bmatrix} G_0 & G_1 & \cdots & G_m & & & & & \\ & G_0 & \cdots & G_{m-1} & G_m & & & & \\ & & \ddots & \vdots & G_{m-1} & \ddots & & & \\ & & & \ddots & \vdots & & \ddots & & \\ & & & & G_1 & \cdots & \cdots & G_m & \\ & & & & G_0 & G_1 & \cdots & G_{m-1} & G_m \\ & & & & & G_0 & \cdots & G_{m-2} & G_{m-1} \\ & & & & & & \ddots & \vdots & \vdots \\ & & & & & & & G_0 & G_1 \\ & & & & & & & & G_0 \end{bmatrix}$$

Se o codificador baseia sua decisão nos $(i + 1)$ blocos, então, qualquer subseqüência $[w]_i$ de comprimento $(i + 1)$ n-uplas que esteja a uma distância $\lfloor (d_i - 1)/2 \rfloor$ ou menos de $[x]_i$ está mais perto deste último que qualquer outra subseqüência do código. Esta distância definida, t_i, é, então, a capacidade de correção de erros para o codificador truncado $\lfloor G \rfloor_i$.

2.1.4. Codificadores convolucionais sistemáticos recursivos

O estudo dos codificadores convolucionais sistemáticos recursivos é de grande importância, uma vez que eles são utilizados como componentes do codificador turbo, responsável pela codificação do sistema proposto, e que será descrito no próximo capítulo. Antes, porém, da introdução dos codificadores convolucionais sistemáticos recursivos, é importante uma discussão inicial sobre uma importante sub-classe dos codificadores convolucionais, que é aquela formada pelos "codificadores sistemáticos". Num codificador sistemático, as primeiras k seqüências de saída são réplicas exatas das k seqüências de entrada,

$$x^{(i)} = u^{(i)} \qquad i = 1, 2, \ldots, k \tag{2.36}$$

e as seqüências geradoras satisfazem,

$$g_i^{(j)} = \begin{cases} 1 & se\ j = i \\ 0 & se\ j \neq i \end{cases}, \qquad i = 1, 2, \ldots, k \tag{2.37}$$

Analisando sob o domínio da transformada, a matriz função de transferência é da forma:

$$G(D) = \begin{bmatrix} 1 & 0 & \ldots & 0 & g_1^{(k+1)}(D) & \ldots & g_1^{(n)}(D) \\ 0 & 1 & \ldots & 0 & g_2^{(k+1)}(D) & \ldots & g_2^{(n)}(D) \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & 1 & g_k^{(k+1)}(D) & \ldots & g_k^{(n)}(D) \end{bmatrix} \tag{2.38}$$

A Fig. 2.5 é um exemplo de um codificador sistemático *(2, 1, 3)* cuja função de transferência é $G(D) = [1 \quad 1+D+D^2+D^3]$.

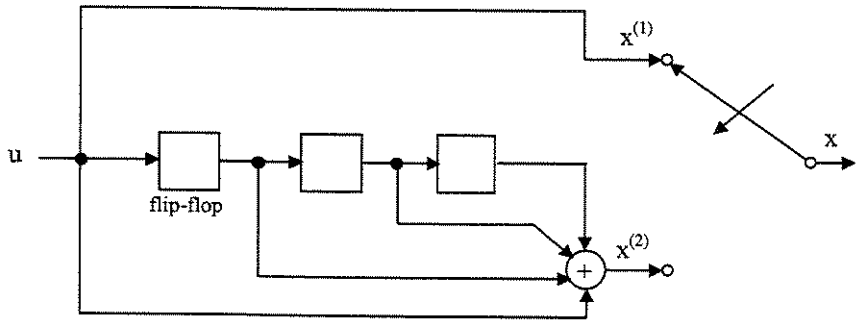

Fig. 2.5 – Exemplo de codificador convolucional sistemático binário para código (2, 1, 3).

Já que as k primeiras seqüências de saída são iguais às seqüências de entrada, elas são chamadas de "seqüências de informação"; as últimas $n - k$ seqüências de saída recebem o nome de "seqüências de paridade" (esta heterogeneidade nas seqüências de saída pode ser vista, claramente, no formato da matriz função de transferência do codificador – eq. 2.38). Nota-se que, enquanto um codificador genérico (n, k, m) necessita da especificação de $k \cdot n$ seqüências geradoras, apenas $k \cdot (n - k)$ seqüências são necessárias para um codificador sistemático. Qualquer outro codificador que não satisfaça as eqs. (2.36) – (2.38) é dito ser "não - sistemático". Os codificadores sistemáticos oferecem algumas vantagens pertinentes sobre os não-sistemáticos:

- A codificação é mais simples do que para os não-sistemáticos, uma vez que menos *hardware* é requerido. Como exemplo, observando o codificador não-sistemático da Fig. 2.1, nota-se que são necessárias duas portas lógicas ou-exclusivo com um total de sete entradas para a obtenção das seqüências de saída; enquanto que o codificador sistemático da Fig. 2.5 requer, apenas, um ou-exclusivo de quatro entradas. Em adição, para um codificador sistemático onde $k < n - k$, é possível projetar um circuito de codificação que requeira um número menor do que os K registradores do circuito normal.
- Na reobtenção da seqüência de informação, após ter sido codificada, esses codificadores não necessitam de circuito de inversão, uma vez que tal seqüência pode ser facilmente obtida através da leitura das saídas sistemáticas $(x)^i = (u)^i$, $i = 1, ..., k$, ao contrário dos não-sistemáticos. Para estes últimos, é necessária uma matriz $G^{-1}(D)$, $n \times k$, tal que

$$G(D)G^{-1}(D) = ID^l \tag{2.39}$$

para algum $l \geq 0$, onde I é a matriz identidade $k \times k$. Por (2.22) e (2.39),

$$X(D)G^{-1}(D) = U(D)G(D)G^{-1}(D) = U(D)D^l \tag{2.40}$$

e a seqüência de informação pode ser reobtida, com um atraso de l unidades de tempo da palavra código, fazendo, agora, $X(D)$ ser a entrada do novo circuito seqüencial de n entradas e k saídas cuja função de transferência é $G^{-1}(D)$. Esta matriz inversa é obtida da seguinte forma: dado um codificador (n, k, m), a matriz $G(D)$ tem uma inversa $G^{-1}(D)$, de atraso l, se e somente se

$$MDC\left[\Delta_i(D) : i = 1,2,\ldots,\binom{n}{k}\right] = D^l \tag{2.41}$$

para algum $l \geq 0$, onde MDC é o máximo divisor comum e $\Delta_i(D)$ são os determinantes das $\binom{n}{k}$ submatrizes distintas $k \times k$ de função de transferência $G(D)$. Se $k = 1$, existe uma matriz inversa se e somente se

$$MDC[g^{(1)}(D), g^{(2)}(D), \ldots, g^{(n)}(D)] = D^l \tag{2.42}$$

para algum $l \geq 0$.

- *Catastroficidade:* O fato de um codificador não apresentar matriz inversa pode causar efeitos indesejáveis. Para fins de análise, um exemplo é considerado. Seja o código $(2, 1, 2)$ com $g^{(1)}(D) = 1+D$ e $g^{(2)}(D) = 1+D^2$. O MDC é calculado tal qual a eq. 2.42 e é encontrado $MDC[1+D, 1+D^2] = 1+D$ não satisfazendo, portanto, a condição imposta. Dessa forma, diz-se que o código não possui matriz inversa. Se considerarmos que a seqüência de informação é dada por $u(D) = 1/1+D = 1+D+D^2+\ldots$, as seqüências de saídas serão da forma $x^{(1)}(D) = 1$ e $x^{(2)}(D) = 1+D$. Isto implica que a palavra código $x(D)$ terá, apenas, três bits não nulos, mesmo se a seqüência de entrada tem um peso infinito. Se esta palavra for transmitida por um canal binário simétrico (BSC), com probabilidade de transição $p < ½$, e os três bits "1" forem mudados para "0" pelo ruído, a seqüência recebida será toda nula. Um decodificador de máxima verossimilhança irá produzir, então, como estimação, uma palavra código toda nula, já que é uma palavra válida do código e concorda, exatamente, com a seqüência recebida. Finalmente, a seqüência de informação estimada será $\hat{u}(D) = 0$, o que implica num número infinito de erros de decodificação provocado por um número finito de erros do canal. Este acontecimento é notadamente indesejável e o codificador é dito ser "catastrófico", por estar sujeito a propagações de erros catastróficos. As eq. 2.41 e 2.42 foram demonstradas por *Massey* e *Sain* serem condições necessárias e suficientes para um codificador ser "não-catastrófico". Eis uma outra vantagem dos codificadores sistemáticos: eles são sempre não-catastróficos. Uma maneira de visualizar, se um

codificador é ou não catastrófico, é através do seu diagrama de estados: ele é catastrófico se e somente se o diagrama contém um outro laço de peso nulo que não seja o laço do próprio estado S_0.

Um codificador convolucional sistemático recursivo (RSC) binário pode ser obtido de um codificador convolucional não-sistemático (NSC). Considerando um codificador convolucional binário de taxa $R = ½$, ordem de memória m e equação de codificação dada pela eq. 2.8, a transformação é feita através da utilização de um laço de realimentação e fazendo com que uma das saídas seja igual à entrada u. Na tentativa de visualizar melhor o problema, é sugerida a observação da Fig. 2.1. Agora, a entrada do registrador de deslocamento não é mais u, e, sim, uma nova variável a. Escolhendo, arbitrariamente, $x^{(1)} = u$ ou $x^{(2)} = u$, a outra saída será definida pela eq. 2.8 substituindo u por a, e esta nova variável é recursivamente calculada, como em [1],

$$a_t = u_t + \sum_{l=1}^{m} g_l^{(j)} a_{t-l} \qquad \begin{matrix} t = 0, 1, 2, \ldots \\ j = 1 \text{ ou } 2 \end{matrix} \tag{2.43}$$

onde $a_{t-l} = 0$ para todo $l > t$ e j é o índice da saída escolhida como não sistemática. A eq. 2.43 pode ser reescrita como

$$u_t = \sum_{l=0}^{m} g_l^{(j)} a_{t-l} \tag{2.44}$$

onde as mesmas condições são consideradas. A Fig. 2.6 representa um codificador RSC obtido do codificador NSC, da Fig. 2.1, para o código *(2, 1, 3)* com $G(D) = [1+D^2+D^3,\ 1+D+D^2+D^3]$ e fazendo $x^{(1)} = u$.

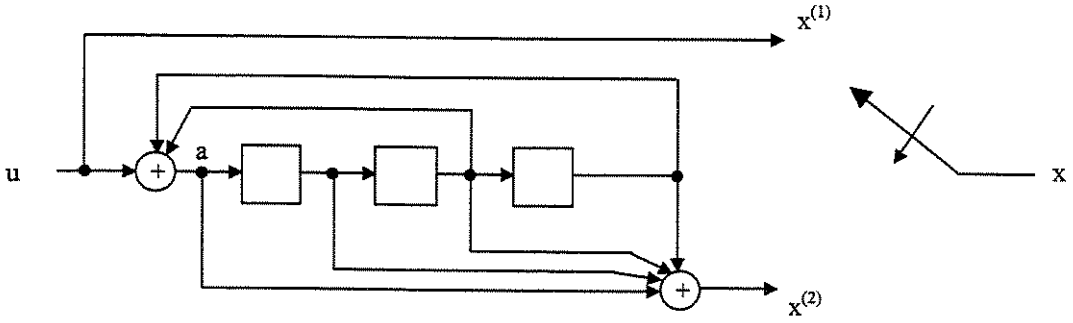

Fig. 2.6 – Codificador convolucional sistemático recursivo

Geralmente, assume-se que o bit de entrada, u_t, onde t é a unidade de tempo, toma valores "0" e "1" com a mesma probabilidade. Em [1], é mostrado que a variável a_t possui as

mesmas propriedades estatísticas que u_t. Assim, as probabilidades de transição de estados $p_t(S_t = m|S_{t-1} = m')$, onde $S_t = m$ e $S_{t-1} = m'$ são, respectivamente, os estados do codificador no tempo t e t - 1, são idênticas para os códigos NSC e RSC equivalentes; e, ainda, os dois códigos apresentam a mesma distância *livre*, d_{free}. Entretanto, para uma mesma seqüência de entrada u, as duas seqüências de saída são diferentes para os dois tipos de código. Para as mesmas seqüências geradoras, pode-se dizer que os códigos RSC não modificam a distribuição de pesos das palavras codificadas, se comparado a um NSC. Eles modificam, apenas, o mapeamento entre as seqüências de entrada e as seqüências de saída codificadas.

O desempenho dos códigos RSC pode ser analisado comparativamente ao desempenho dos códigos NSC, através da determinação dos respectivos espectro de pesos e taxa de erro de bit (BER). O espectro de pesos pode ser obtido da eq. 2.30, fazendo $Y = Z = 1$ e considerando que d_{free} é a menor potência de X, como observado na seção 2.1.3:

$$T(X,Y,Z)\big|_{Y,Z=1} = \sum_{d=d_{free}}^{\infty} \sum_{j=1}^{\infty} \sum_{l=1}^{\infty} A_{d,j,l} X^d = \sum_{d=d_{free}}^{\infty} A_d X^d \tag{2.45}$$

onde $A_d = \sum_j \sum_l A_{d,j,l}$ é o número de caminhos de distância de Hamming d, em relação ao estado nulo, e ainda por

$$\frac{\partial T(X,Y,Z)}{\partial Y}\Big|_{Y,Z=1} = \sum_{d=d_{free}}^{\infty} \sum_{j=1}^{\infty} \sum_{l=1}^{\infty} jA_{d,j,l} X^d = \sum_{d=d_{free}}^{\infty} W_d X^d \tag{2.46}$$

onde $W_d = \sum_j \sum_l jA_{d,j,l}$ é o número total de bits de informação não nulos, ou seja, o peso de Hamming total das seqüências de entrada, utilizados em todos os caminhos de peso d. Por [1], é sabido que um grande número de códigos RSC foram investigados e seus desempenhos comparados com os códigos NSC através da determinação daqueles dois parâmetros acima citados. Os coeficientes A_d são os mesmos para ambos os códigos, porém os coeficientes W_d dos RSC têm uma tendência de crescer, de uma maneira mais lenta, em função de d, do que os mesmos coeficientes para os códigos NSC, qualquer que seja a taxa R e a ordem de memória m. Desta forma, para baixas relações sinal-ruído (SNR), a taxa de erro de bit dos códigos RSC são sempre menores que aquelas para os códigos NSC equivalentes. Em geral, para taxas de código $R \leq 2/3$, os primeiros coeficientes $(W_{dfree}, W_{dfree+1})$ de RSC são maiores que aqueles para os códigos

NSC. Por conseguinte, para SNR altas, o desempenho dos códigos NSC é um pouco melhor do que o obtido para os RSC. Quando $R > 2/3$, é comum encontrar códigos RSC cujo desempenho é melhor do que aquele obtido para os NSC para qualquer SNR.

2.2. Modulação M-DPSK

Ao chegar ao receptor, o sinal recebido possui um deslocamento de fase em relação ao sinal transmitido. Dessa forma, a princípio, o receptor teria que conhecer esta referência de fase, extraindo-a do sinal recebido, através de circuitos de sincronização, para realizar uma boa decodificação. A modulação PSK (*phase shifting keying*) é um exemplo de sistema que necessita de sincronização, já que a informação que se quer transmitir está na fase do sinal. A modulação DPSK (*differential phase shift keying*), por sua vez, é uma técnica que pode ser utilizada, para eliminar essa necessidade de sincronização, uma vez que a fase adicionada, θ, é assumida constante em dois intervalos de símbolo e a informação consiste na diferença de fase transmitida em dois intervalos adjacentes.

A regra de decisão ótima para o receptor não-coerente (que não necessita de circuitos de sincronização) [12] será descrita a seguir.

O sinal transmitido é da forma:

$$s_{i1}(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi f_c t + \overbrace{\frac{(2i_1 - 1)\pi}{M}}^{\phi_1} + \theta\right) \qquad 0 \le t < T \qquad (2.47)$$

$$s_{i2}(t) = \sqrt{\frac{2E}{T}} \cos\left(2\pi f_c t + \overbrace{\frac{(2i_2 - 1)\pi}{M}}^{\phi_2} + \theta\right) \qquad T \le t < 2T \qquad (2.48)$$

onde E é a energia do sinal transmitido $s(t)$ e ϕ é a fase do sinal transmitido, que contém informação e pode assumir uma das M fases PSK:

$$\left\{\frac{(2i-1)\pi}{M}\right\}_{i=1}^{M}$$

A variável aleatória θ é modelada como uniforme no intervalo $[0, 2\pi]$, $p_\theta(\Theta) = 1/2\pi$ para $0 \le \Theta \le 2\pi$, o que representa total ignorância, por parte do receptor, sobre a fase do sinal recebido.

De uma forma geral, por [12], se as mensagens são equiprováveis, ou seja, $p(H_i) = 1/N$ para $i = 1, ..., N$ e H_i equivale à afirmação: $s_i(t)$ foi transmitido, a regra de decisão do receptor é maximizar $p_{r/Hi}(R)$ – eq. 2.49. O cálculo desta probabilidade está relacionado ao conhecimento da probabilidade relativa à variável aleatória θ,

$$p_{r/H_i}(R) = \int_{-\infty}^{\infty} p_{r/H_i,\theta=\Theta}(R)\, p_\theta(\Theta) d\Theta \tag{2.49}$$

visto que, condicionada à hipótese H_i e a uma fase fixa $\theta = \Theta$, $p_{r/Hi,\theta=\Theta}(R)$ é um vetor gaussiano:

$$p_{r/H_i,\theta=\Theta}(R) = \frac{1}{(2\pi\sigma_n^2)^{N/2}} \exp\left\{\frac{-|R - s_i(\Theta)|^2}{2\sigma_n^2}\right\} \tag{2.50}$$

onde $\sigma_n^2 = N_0/2$ e R é um vetor que representa o sinal recebido $r(t)$, saída do filtro casado. Desenvolvendo a eq. 2.50, desprezando os termos que não dependem das hipóteses e substituindo na eq. 2.49, tem-se que o receptor ótimo deve maximizar:

$$D(R, s_i) = \int_{-\infty}^{\infty} \exp\left\{\frac{2R^T s_i(\Theta) - |s_i(\Theta)|^2}{N_0}\right\} p_\theta(\Theta) d\Theta \tag{2.51}$$

Ou, de maneira equivalente,

$$D(r(t), s_i(t)) = \int_{-\infty}^{\infty} \exp\left\{\frac{2}{N_0}\int_0^T r(t) s_i(t,\Theta) dt - \frac{1}{N_0}\int_0^T s_i^2(t,\Theta) dt\right\} p_\theta(\Theta) d\Theta \tag{2.52}$$

Substituindo-se, portanto, $p_\theta(\Theta)$ e sabendo que $\int_0^T s_i^2(t,\Theta) dt = E$ obtém-se:

$$D(r(t), s_i(t)) = e^{\frac{-E}{N_0}} \int_0^{2\pi} \exp\left\{\frac{2}{N_0}\int_0^T r(t)\, s_i(t,\Theta) dt\right\} \frac{1}{2\pi} d\Theta \tag{2.53}$$

Para o caso em estudo:

Regra: O receptor ótimo observa o sinal $r(t)$, representado vetorialmente na forma $(\underline{r}_{i1}, \underline{r}_{i2})$, no intervalo $0 \leq t \leq 2T$ e escolhe as fases ϕ_1 e ϕ_2 como sendo as fases transmitidas, de forma que a probabilidade $p_{\underline{r}i1,\underline{r}i2/\phi1,\phi2}(\underline{R}_{i1}, \underline{R}_{i2})$ seja maximizada, sendo,

$$\underline{r}_{i1} = \underline{s}_{i1} + \underline{n}_{i1}$$
$$\underline{r}_{i2} = \underline{s}_{i2} + \underline{n}_{i2}$$

Como os vetores $\underline{n}_{i1}$ e $\underline{n}_{i2}$ são definidos em intervalos diferentes, são descorrelacionados e, por serem gaussianos, os mesmos são independentes. Assim, substituindo as variáveis na eq. 2.53 e sabendo que os sinais têm a mesma energia (logo, desprezando o termo associado, já que não influenciará na maximização), $p_{\underline{r}i1,\underline{r}i2/\phi1,\phi2}(\underline{R}_{i1},\ \underline{R}_{i2})$ é maximizada, quando a função de verossimilhança $D(\phi_1, \phi_2)$ for maximizada,

$$D(\phi_1,\phi_2) = \int_0^{2\pi} e^{\frac{2}{N_0}\sqrt{\frac{2E}{T}}} \exp\left\{\int_0^T r(t)\cos(2\pi f_c t + \phi_1 + \Theta)dt + \int_T^{2T} r(t)\cos(2\pi f_c t + \phi_2 + \Theta)dt\right\}\frac{1}{2\pi}d\Theta \tag{2.54}$$

Entretanto, observando a equação anterior, percebe-se que ela não pode ser maximizada, dado as duas fases transmitidas ϕ_1 e ϕ_2. Assim, uma conclusão plausível é que a regra *MV* (máxima verossimilhança) para a modulação *M-PSK* não funciona.

Definindo, agora: $\alpha = \phi_1 + \Theta$

$$\Delta\phi = \phi_2 - \phi_1 \qquad \Rightarrow \phi_2 = \Delta\phi + \phi_1$$
$$\phi_2 + \Theta = \Delta\phi + \phi_1 + \Theta = \Delta\phi + \alpha$$

e substituindo na eq. 2.54, tem-se que:

$$D(\Delta\phi) = \int_0^{2\pi} e^{\frac{2}{N_0}\sqrt{\frac{2E}{T}}\left\{\int_0^T r(t)\cos(2\pi f_c t+\alpha)dt + \int_T^{2T} r(t)\cos(2\pi f_c t+\Delta\phi+\alpha)dt\right\}}\frac{1}{2\pi}d\alpha \tag{2.55}$$

Definindo, então:

$$\begin{aligned} X(\Delta\phi) &= \sqrt{\frac{2}{T}}\left\{\int_0^T r(t)\cos 2\pi f_c t\ dt + \int_T^{2T} r(t)\cos(2\pi f_c t + \Delta\phi)\ dt\right\} \\ Y(\Delta\phi) &= \sqrt{\frac{2}{T}}\left\{\int_0^T r(t)\,\mathrm{sen}\, 2\pi f_c t\ dt + \int_T^{2T} r(t)\,\mathrm{sen}(2\pi f_c t + \Delta\phi)\ dt\right\} \end{aligned} \tag{2.56}$$

Obtem-se, portanto:

$$D(\Delta\phi) = \frac{1}{2\pi}\int_0^{2\pi} e^{\frac{2}{N_0}\sqrt{E}\{X(\Delta\phi)\cos\alpha - Y(\Delta\phi)\,\mathrm{sen}\,\alpha\}}d\alpha \tag{2.57}$$

Desenvolvendo o termo entre chaves,

$$X(\Delta\phi)\cos\alpha - Y(\Delta\phi)\,\mathrm{sen}\,\alpha = \mathrm{Re}\{(X(\Delta\phi) - jY(\Delta\phi))(\cos\alpha - j\,\mathrm{sen}\,\alpha)\} =$$
$$= \mathrm{Re}\left\{\sqrt{X^2(\Delta\phi)+Y^2(\Delta\phi)}\ e^{-jtg^{-1}\frac{Y(\Delta\phi)}{X(\Delta\phi)}}\ e^{-j\alpha}\right\} = \sqrt{X^2(\Delta\phi)+Y^2(\Delta\phi)}\cos\left(\alpha + tg^{-1}\frac{Y(\Delta\phi)}{X(\Delta\phi)}\right)$$

e sabendo que

$$I_0(x) = \frac{1}{2\pi}\int_0^{2\pi} \exp\{x\cos(\alpha+\beta)\}d\alpha \tag{2.58}$$

é a função de Bessel modificada de ordem zero e primeira espécie, pode-se, através de uma comparação entre as eq. 2.57 e 2.58, concluir que,

$$D(\Delta\phi) = I_0\left(\frac{2}{N_0}\sqrt{E}\left(X^2(\Delta\phi)+Y^2(\Delta\phi)\right)^{1/2}\right) \tag{2.59}$$

Sendo $I_0(x)$ uma função monotonicamente crescente, a decisão ótima é selecionar o valor de $\Delta\phi$ que maximiza a relação

$$R^2(\Delta\phi) = X^2(\Delta\phi)+Y^2(\Delta\phi) \tag{2.60}$$

Reescrevendo as eqs. 2.56 na forma,

$$\begin{aligned} X(\Delta\phi) &= x_1 + x_2\cos\Delta\phi + y_2\,\mathrm{sen}\,\Delta\phi \\ Y(\Delta\phi) &= -y_1 - y_2\cos\Delta\phi + x_2\,\mathrm{sen}\,\Delta\phi \end{aligned} \tag{2.61}$$

onde:

$$x_1 = \sqrt{\frac{2}{T}}\int_0^T r(t)\cos 2\pi f_c t\,dt \quad ; \quad x_2 = \sqrt{\frac{2}{T}}\int_T^{2T} r(t)\cos 2\pi f_c t\,dt$$

$$y_1 = -\sqrt{\frac{2}{T}}\int_0^T r(t)\,\mathrm{sen}\,2\pi f_c t\,dt \quad ; \quad y_2 = -\sqrt{\frac{2}{T}}\int_T^{2T} r(t)\,\mathrm{sen}\,2\pi f_c t\,dt$$

e substituindo a eq. 2.61 na eq. 2.60, tem-se que:

$$R^2(\Delta\phi) = x_1^2 + y_1^2 + x_2^2 + y_2^2 + 2(x_1x_2 + y_1y_2)\cos\Delta\phi + 2(x_1y_2 - y_1x_2)\,\mathrm{sen}\,\Delta\phi \tag{2.62}$$

Dessa forma, o receptor ótimo seleciona a diferença de fase $\Delta\phi$ tal que maximize a função de verossimilhança (os demais termos da equação anterior foram desprezados, uma vez que não influenciam na maximização):

$$D(\phi) = (x_1x_2 + y_1y_2)\cos\Delta\phi + (x_1y_2 - y_1x_2)\,\mathrm{sen}\,\Delta\phi \tag{2.63}$$

Definindo-se, ainda:

$$\begin{aligned} r_1 &= x_1 + jy_1 = \rho_1 e^{j\beta_1} \\ r_2 &= x_2 + jy_2 = \rho_2 e^{j\beta_2} \end{aligned}$$

E analisando a equação abaixo,

$$r_1^* r_2\, e^{-j\Delta\phi} = \{x_1 x_2 + y_1 y_2 + j(x_1 y_2 - y_1 x_2)\}\{\cos\Delta\phi - j\,\mathrm{sen}\,\Delta\phi\}$$
$$= \rho_1\rho_2\, e^{j((\beta_2-\beta_1)-\Delta\phi)}$$

nota-se, após algumas manipulações na equação acima, que $D(\Delta\phi)$ pode ser reescrito na forma:

$$D(\Delta\phi) = \mathrm{Re}\{r_1^* r_2\, e^{-j\Delta\phi}\} \qquad ou$$
$$= \rho_1\rho_2 \cos[(\beta_2 - \beta_1) - \Delta\phi] \tag{2.64}$$

O fato da fase θ ser constante, em dois intervalos adjacentes, faz com que o sinal $r_1^* r_2$ seja independente de θ. Conclui-se, então, que, definindo-se $\Delta\phi$, a maximização da eq. 2.54 torna-se possível e tem-se a modulação *M-DPSK*. A regra de decisão, para sinais DPSK, pode ser, portanto, formulada como: Dadas as fases β_1 e β_2 dos sinais $\underline{r}_1$ e $\underline{r}_2$ recebidos em dois intervalos sucessivos, o receptor ótimo escolhe a diferença de fase $\Delta\hat{\varphi}$ que maximiza a quantidade $\cos[(\beta_2 - \beta_1) - \Delta\hat{\phi}]$. Observando que a função cos[·] é decrescente, em relação ao eixo *y*, no intervalo [-π/2, π/2], a regra de decisão resulta, simplesmente, em escolher $\Delta\hat{\varphi}$ que minimize a diferença $|(\beta_2 - \beta_1) - \Delta\phi|$. A regra *MV* para *DPSK*, com dois intervalos de observação, resulta, pois, na regra clássica para o demodulador diferencial.

Pode-se decidir, corretamente, se o ângulo do sinal $r_1^* r_2$ pertence ao setor de comprimento angular $2\pi/M$ centrado em $\Delta\phi$. Desse modo, os possíveis valores assumidos por $\Delta\phi = \phi_1 - \phi_2$ são:

$$\Delta\phi = \left\{\frac{(2i-1)\pi}{M}\right\}_{i=1}^{M} - \left\{\frac{(2j-1)\pi}{M}\right\}_{j=1}^{M} = \left\{\frac{(2l-1)\pi}{M}\right\}_{l=1}^{M} \tag{2.65}$$

No caso do 4-*DPSK* ($M = 4$) as regiões de decisão seriam:

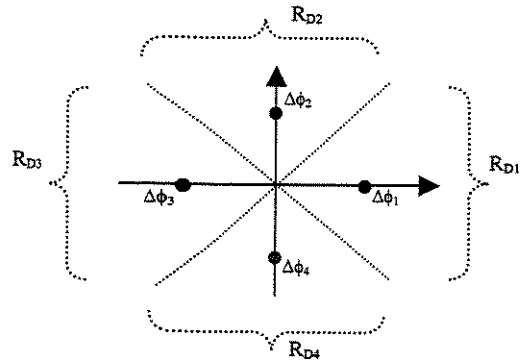

Fig. 2.7 – Regiões de decisão para uma modulação 4-DPSK

Uma outra maneira de representar o sinal transmitido seria a de considerar as fases PSK transmitidas, nas eqs. 2.47 e 2.48, assumindo os M possíveis valores uniformemente distribuídos $\phi_i = 2\pi i/M;\ i = 0, ..., M - 1$. Nesse caso, as fases poderiam tocar os eixos, o que não acontece com a representação anterior. A Fig. 2.8 é um exemplo das duas representações para $M = 4$.

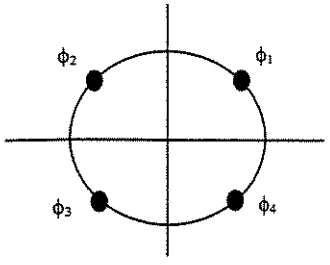

Fig. 2.8 a – Modulação PSK considerando $\phi_i = (2i - 1)\pi/4$; i = 1, ..., 4

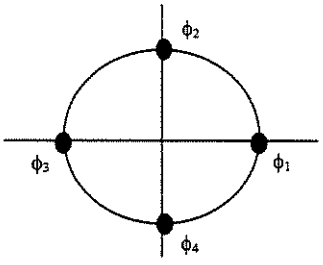

Fig. 2.8 b – Modulação PSK considerando $\phi_i = 2i\pi/4$; i = 0, ..., 3

No caso da segunda representação, a diferença de fase $\Delta\phi = \phi_2 - \phi_1$, estimada pelo receptor, será sempre equivalente a uma das próprias fases já representadas e as regiões de decisão R_{Di} estarão centradas nessas fases. Assim, o mesmo esquema pode ser utilizado, tanto na modulação quanto na demodulação. Observe que isto não acontece com a primeira representação, visto que as possíveis fases PSK estão fora dos eixos e $\Delta\phi = \phi_2 - \phi_1$ poderá assumir valores sobre os eixos (no caso $M = 4$, por exemplo, $\Delta\phi$ sempre estará sobre os eixos) o que requer um outro esquema cujos possíveis valores de $\Delta\phi$ sejam o centro das regiões de decisão R_{Di} (Fig. 2.7 para $M = 4$). No sistema proposto, capítulo 4, é utilizada a segunda representação.

É sabido que, na modulação DPSK, a informação transmitida não está na fase absoluta do sinal que se transmite, mas, sim, na fase relativa ao sinal transmitido no intervalo anterior e que o receptor ótimo faz uma decisão sobre a diferença de fase em dois intervalos adjacentes. Supondo que a fase PSK, no k-ésimo intervalo, seja $\phi_t \in \left\{2i\pi/M\right\}_{i=0}^{M-1}$, um esquema possível de transmissão e recepção [12] seria o da Fig. 2.9.

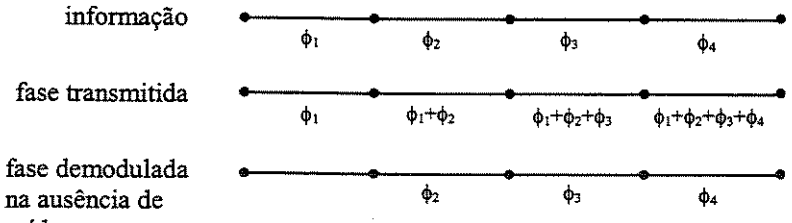

Fig. 2.9 – Esquema de transmissão DPSK

Assim, se a fase transmitida for a soma das fases PSK anteriores, a fase demodulada $\Delta\phi_t$ será a informação ϕ_t que se desejou transmitir. Este é, então, a técnica DPSK. A seguir, é mostrado o esquema de transmissão implementado no capítulo 4, que utiliza modulação 4-DPSK. Primeiramente, os bits de informação são divididos em duas sub-seqüências, I_1 e I_2, as quais são associadas uma das quatro fases PSK de forma arbitrária. Para o sistema proposto, no capítulo 4, a associação é feita segundo a tabela 2.1:

Informação		Fase PSK
I_1	I_2	
0	0	π
0	1	$\pi/2$
1	0	$3\pi/2$
1	1	0

Tab. 2.1 – Associação utilizada entre os bits de informação e as fases 4-PSK

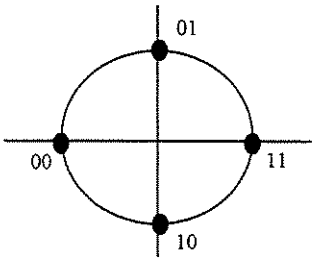

Fig. 2.10 – Esquema de modulação utilizado

Em seguida, é feita a transmissão das fases absolutas, seguindo a idéia da Fig. 2.9, ou seja, a fase em um intervalo é a soma das fases anteriores. Na recepção, os intervalos são analisados, dois a dois, e a diferença de fase $\Delta\phi_t = \phi_t - \phi_{t-1}$, onde t é o intervalo atual, é obtida através da análise das regiões de decisão como sugerido no texto explicativo da Fig. 2.8b.

A Fig. 2.11 é um exemplo utilizando $t = 4$ e fazendo a demodulação sem considerar o ruído do canal:

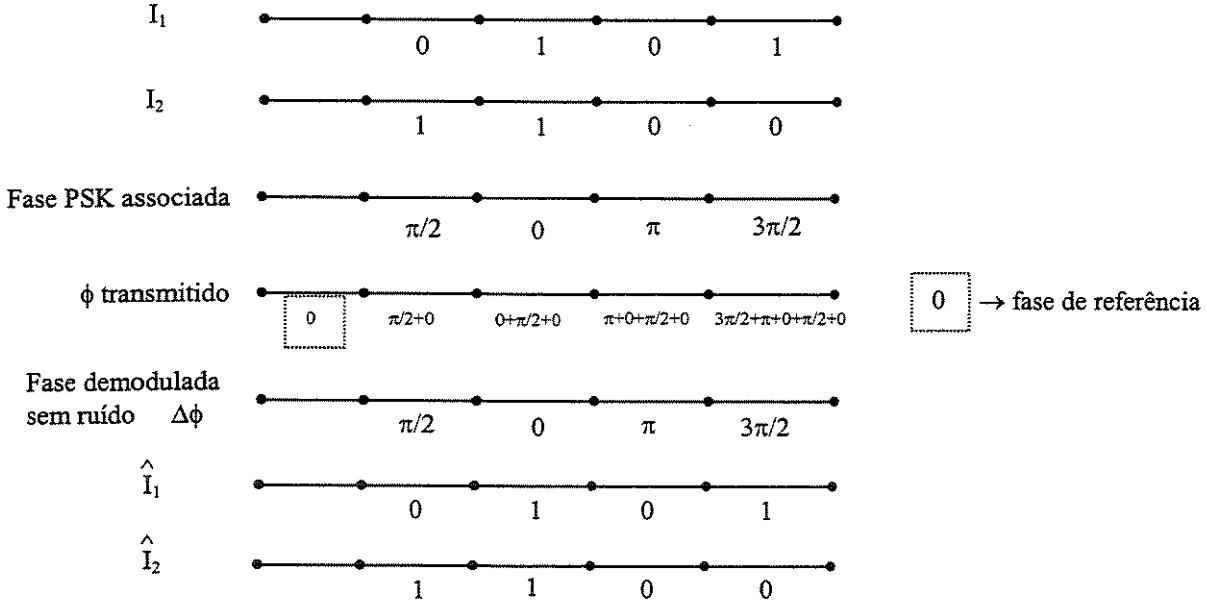

Fig. 2.11 – Transmissão e recepção de um esquema 4-DPSK

Na transmissão, é necessária a utilização de uma fase inicial arbitrária, de forma a servir de referência na demodulação. A associação entre os bits de informação e as fases PSK, em conjunto com a soma das fases absolutas dos intervalos anteriores no atual, é definida, aqui, como codificação diferencial.

2.3. Decodificação MAP versus MV

A decodificação de códigos convolucionais pode ser feita por diversas maneiras. Esta seção ater-se-á a dois esquemas em particular, baseados em dois critérios diferentes: máxima *a posteriori* (MAP) e máxima verossimilhança (MV). Durante a seção, algumas demonstrações são deixadas de lado, uma vez que o intuito é, apenas, o de introduzir os tipos de decodificação mencionados.

No projeto do decodificador, uma maneira de se tomar uma decisão sobre o sinal transmitido é que ela seja feita, em cada intervalo, baseada na observação do sinal recebido r, de tal forma que a probabilidade de uma decisão correta seja maximizada. Dessa maneira, a regra de decisão utilizada é baseada no cálculo de probabilidades *a posteriori,* definidas como

$$\text{P(sinal } s_i \text{ ter sido transmitido} \mid r) \text{ ou } P(s_i \mid r), \qquad \text{onde } i = 1, \ldots, M$$

O critério baseia-se na seleção do sinal correspondente ao máximo do conjunto de probabilidades *a posteriori* $\{P(s_i \mid r)\}$ e é denominado *critério de máximo a posteriori (MAP).* Pode-se demonstrar que este critério maximiza a probabilidade de uma decisão correta e, por conseqüência, minimiza a probabilidade de erro de símbolo ou bit. Um exemplo de um esquema baseado neste critério é o algoritmo BCJR (seção 2.3.2) que é utilizado no receptor do sistema proposto no capítulo 4.

Um outro critério é baseado na função densidade de probabilidade *(fdp)* condicional $p(r \mid s_i)$ que é, usualmente, conhecida como função de verossimilhança. A decisão baseia-se em escolher o máximo de $p(r \mid s_i)$ sobre todos os M sinais e o critério é denominado *critério de máxima verossimilhança (MV).* Demonstra-se que ele minimiza a probabilidade de erro de palavra (ou seqüência). Um esquema que faz uso deste critério é a decodificação de Viterbi, que será descrita na próxima seção.

2.3.1. Decodificação de *Viterbi*

O algoritmo foi proposto por *Viterbi*, em 1967, como uma técnica de decodificação assintoticamente ótima para códigos convolucionais e é um método que minimiza a probabilidade de erro de seqüência.

A operação do algoritmo pode ser melhor explicada com o auxílio da treliça da Fig. 2.3 para o código convolucional C(2, 1, 2) de matriz geradora $G(D) = [1 + D + D^2, 1 + D^2]$. (As variáveis aqui inseridas são as mesmas introduzidas na seção 2.1). Assume-se que a seqüência de informação $\boldsymbol{u} = (u_0, \ldots, u_{L-1})$ de comprimento kL é codificada em uma palavra código $\boldsymbol{x} = (x_0, \ldots, x_{L+m-1})$ de comprimento $N = n(L + m)$, e que a seqüência Q-ária $\boldsymbol{r} = (r_0, \ldots, r_{L+m-1})$ é recebida de um canal discreto sem memória (*DMC*) de entrada binária e saída Q-ária. Alternativamente, estas seqüências podem ser expressas como $\boldsymbol{u} = (u_0, \ldots, u_{kL-1})$, $\boldsymbol{x} = (x_0, \ldots, x_{N-1})$ e $\boldsymbol{r} = (r_0, \ldots, r_{N-1})$, onde, agora, os subscritos representam, simplesmente, a ordem dos símbolos em cada seqüência. O decodificador deve produzir uma estimativa $\hat{x}$ de x baseado na seqüência recebida r. Portanto, um decodificador de máxima verossimilhança (*DMV*) (no caso, o de *Viterbi*), para um *DMC*, escolhe $\hat{x}$ como a palavra código x que minimiza a função *log-likelihood* $\log P(\boldsymbol{r} \mid \boldsymbol{x})$. Assim, para o canal em questão:

$$P(\mathbf{r} \mid \mathbf{x}) = \prod_{i=0}^{L+m-1} P(\mathbf{r}_i \mid \mathbf{x}_i) = \prod_{i=0}^{N-1} P(r_i \mid x_i) \tag{2.66}$$

E, em termos de logaritmo,

$$\log P(\mathbf{r} \mid \mathbf{x}) = \sum_{i=0}^{L+m-1} M(\mathbf{r}_i \mid \mathbf{x}_i) = \sum_{i=0}^{N-1} M(r_i \mid x_i) \tag{2.67}$$

onde $P(r_i \mid x_i)$ é a probabilidade de transição de canal. A função *log-likelihood* $\log P(\boldsymbol{r} \mid \boldsymbol{x})$ é denominada *métrica* associada com o caminho x e é denotada por $M(\boldsymbol{r} \mid \boldsymbol{x})$. Os termos $\log P(\boldsymbol{r}_i \mid \boldsymbol{x}_i)$ são chamados *métricas de ramo* e são denotados por $M(\boldsymbol{r}_i \mid \boldsymbol{x}_i)$; os termos $\log P(r_i \mid x_i)$ são chamados *métricas de bit* e são denotados por $M(r_i \mid x_i)$. Assim, a métrica de caminho $M(\boldsymbol{r} \mid \boldsymbol{x})$ pode ser escrita como:

$$M(\mathbf{r} \mid \mathbf{x}) = \sum_{i=0}^{L+m-1} M(\mathbf{r}_i \mid \mathbf{x}_i) = \sum_{i=0}^{N-1} M(r_i \mid x_i) \tag{2.68}$$

A seqüência recebida, r, pode ser potencialmente errônea, uma vez que é produto de um canal com ruído, e, conseqüentemente, pode não corresponder, de forma exata, a um caminho

na treliça. O algoritmo de *Viterbi* age de forma a calcular a métrica *log-likelihood* para cada caminho em cada nível da treliça. Após calculada, ela é mantida como uma métrica parcial para cada ramo e o processo é repetido, enquanto a treliça é "visitada". Quando aplicado a uma seqüência r obtida de um *DMC*, o algoritmo encontra o caminho, na treliça, com a maior métrica, ou seja, o caminho de máxima verossimilhança. A seqüência recebida é processada de uma maneira iterativa: em cada passo, o algoritmo compara a métrica de todos os caminhos que entram em cada estado e armazena, apenas, o caminho com a maior métrica, chamado sobrevivente, juntamente com a métrica correspondente. Este tipo de decisão é dita *decisão suave*.

O algoritmo pode ser resumido da seguinte maneira (onde t é o nível atual da treliça):

1) Para $t = 1$ até $t = m$ (memória do codificador convolucional), é feito o cálculo da métrica parcial para o único caminho que entra em cada estado. O caminho (sobrevivente) e a métrica associada são armazenados para cada estado. Próximo passo.
2) t é incrementado de um. Tendo atingido a porção central da treliça, todos os estados são possíveis; agora, é efetuado o cálculo da métrica parcial de todos os caminhos que entram em um estado, somando a métrica de ramo que entra naquele estado com a métrica do caminho sobrevivente, conectado ao estado, obtida no nível anterior. Para cada estado, é armazenado o caminho de maior métrica (o sobrevivente) juntamente com a métrica correspondente e os outros caminhos são eliminados. Passo 3.
3) Se $t < L + m$, voltar ao passo anterior. Caso contrário, fim do algoritmo.

Após o processamento do algoritmo, há 2^K sobreviventes, do nível m ao L, um para cada um dos 2^K estados da treliça. Depois do nível L, existem poucos sobreviventes, visto que a treliça se encontra na "área de retorno ao estado nulo" onde poucos estados são atingidos. Finalmente, no nível $L + m$, o último, há, apenas, um estado, o estado nulo, e, conseqüentemente, um sobrevivente. Em [13], é provado que este sobrevivente final é o caminho de máxima verossimilhança.

Do ponto de vista de implementação, é mais conveniente utilizar número inteiros positivos como métricas, ao invés das métricas de bit originais. Com este intuito, a métrica de bit $M(r_i \mid x_i) = \log \mathrm{P}(r_i \mid x_i)$ pode ser modificada para $c_2 [\log P(r_i \mid x_i) + c_1]$, onde c_1 é um número real e c_2 é um número real positivo. Pode ser mostrado que as métricas modificadas podem substituir

as originais, sem afetar o desempenho do algoritmo, apenas, agora, o algoritmo torna-se ligeiramente sub-ótimo.

Como um exemplo do funcionamento do algoritmo, a treliça da Fig. 2.3 é utilizada juntamente com o *DMC* da Fig. 2.12 e a tabela 2.2. Seja a seqüência de informação de comprimento $L = 4$ dada por $\boldsymbol{u} = (1\ 0\ 1\ 1)$. Esta seqüência é codificada pelo codificador da Fig.2.3, resultando na seqüência codificada $\boldsymbol{x} = (1\ 1\ \ 1\ 0\ \ 0\ 0\ \ 0\ 1\ \ 0\ 1\ \ 1\ 1)$ de comprimento $N = n(L + m) = 12$. A seqüência $\boldsymbol{x}$ é transmitida pelo *DMC* e a seqüência quaternária $\boldsymbol{r} = (I_1\ I_2\ \ I_1\ O_2\ \ I_2\ O_1\ \ O_1\ I_2\ \ O_2\ I_1\ \ I_1\ O_1)$ é recebida.

X_i	r_i (números reais)				r_i (números inteiros)			
	O_1	O_2	I_1	I_2	O_1	O_2	I_1	I_2
0	-0.4	-0.52	-1	-0.7	10	8	0	5
1	-1	-0.7	-0.4	-0.52	0	5	10	8

Tab. 2.2 – Métricas *log-likelihood* reais e inteiras para o DMC

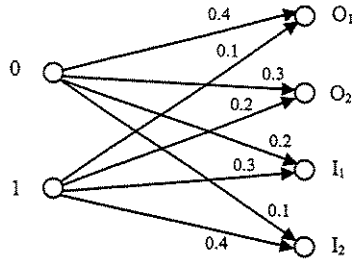

Fig. 2.12 – Canal discreto sem memória de saída quaternária

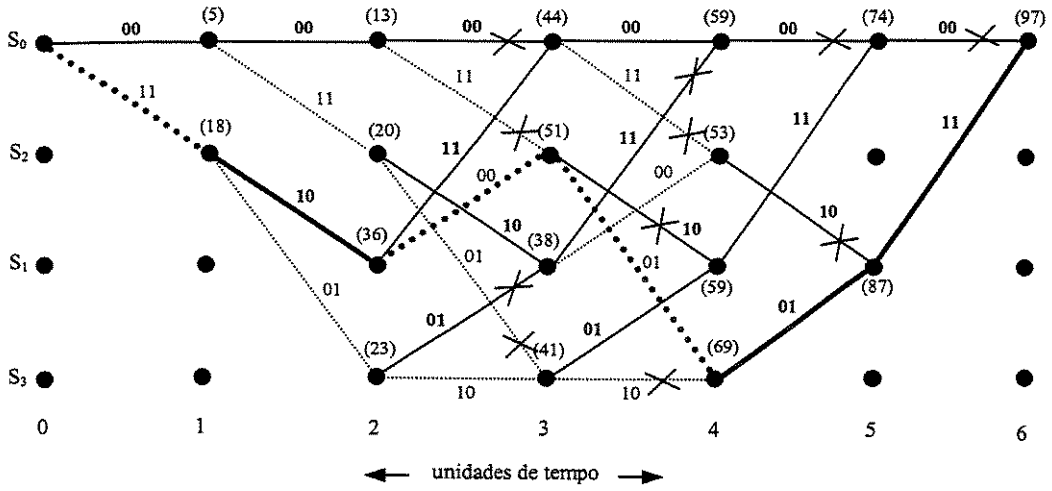

Fig. 2.13 – Treliça para código C (2, 1, 2) com G(D) = [1+D+D^2 1+D^2] exemplificando o algoritmo de Viterbi

Com o auxílio da Fig. 2.13, o algoritmo pode ser visualizado quanto ao acúmulo de métricas e caminhos sobreviventes (que estão em negrito na figura). A seqüência recebida estimada $\hat{x}$ é, então, obtida, $\hat{x} = (1\ 1\ \ 1\ 0\ \ 0\ 0\ \ 0\ 1\ \ 0\ 1\ \ 1\ 1)$, e, por conseqüência, a correspondente estimação da seqüência de informação $\hat{u} = (1\ 0\ 1\ 1)$, que é igual à seqüência original u.

Para um canal binário simétrico (BSC), a métrica *log-likelihood* é o peso de *Hamming* da saída correspondente ao ramo. Agora, o caminho com a menor distância é selecionado, enquanto os outros são descartados. Neste caso, eventualmente, pode ocorrer de nenhum caminho, em um estado, ser descartado, o que indica um empate nos valores das distâncias. Se o sobrevivente final passar por um desses estados, há mais de um caminho de máxima verossimilhança, ou seja, mais de um caminho cuja distância de r é mínima. Em termos de implementação, sempre que um empate ocorrer, um caminho é aleatoriamente escolhido como sobrevivente.

2.3.2. Algoritmo BCJR

O algoritmo BCJR foi proposto por *Bahl, Cocke, Jelinek* e *Raviv*, em 1974, como um método de decodificação ótima, para códigos lineares, que minimiza a probabilidade de erro de símbolo (ou bit). Ele é uma alternativa à decodificação de *Viterbi*, que, como visto, é um método de máxima verossimilhança que minimiza a probabilidade de erro de palavra, contudo, não necessariamente, minimiza a probabilidade de erro de símbolo. Esta seção destina-se a uma breve descrição do algoritmo segundo [2].

Este método está intrinsecamente ligado à utilização de uma fonte Markoviana, portanto, de antemão, é necessário um esquema de transmissão que obedeça à situação da Fig. 2.14, onde o canal exemplificado é um *DMC*.

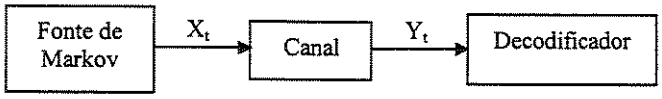

Fig. 2.14 – Diagrama do sistema de transmissão

A fonte é um processo de *Markov* discreto no tempo e de M estados distintos finitos indexados por m, $m = 0, ..., M - 1$. O estado da fonte, no instante de tempo t, é denominado S_t e a saída correspondente, X_t. As transições de estados são dadas pelas probabilidades de transição

$$p_t(m \mid m') = P\{S_t = m \mid S_{t-1} = m'\} \tag{2.69}$$

e as saídas são dadas pelas probabilidades

$$q_t(X \mid m', m) = P\{X_t = X \mid S_{t-1} = m'; S_t = m\} \tag{2.70}$$

onde X pertence a um alfabeto discreto finito. A seqüência de estados da fonte, estendendo-se do instante t ao t' é denotada por $S_t^{t'} = S_t$, S_{t+1} ,..., $S_{t'}$ e a seqüência de saída associada é $X_t^{t'} = X_t$, X_{t+1} ,..., $X_{t'}$.

A fonte de *Markov* pode ser vista como uma treliça, iniciando no estado nulo S_0, produzindo uma seqüência de saída X_1^τ e terminando no estado final S_0. X_1^τ é a entrada do canal discreto sem memória (*DMC*), ruidoso, que produz, como saída, a seqüência $Y_1^\tau = Y_1, Y_2, \ldots, Y_\tau$. As probabilidades de transição do *DMC* são definidas por $R(\cdot \mid \cdot)$ tal que, para todo $1 \le t \le \tau$:

$$P\{Y_1^t \mid X_1^t\} = \prod_{j=1}^{t} R(Y_j \mid X_j) \tag{2.71}$$

O objetivo do algoritmo de decodificação é examinar a seqüência Y_1^τ e estimar as probabilidades *a posteriori* (*APP*) dos estados e transições da fonte. Em outras palavras, o "alvo" do algoritmo é a estimação das probabilidades condicionais:

$$P\{S_t = m \mid Y_1^\tau\} = P\{S_t = m; Y_1^\tau\} \Big/ P\{Y_1^\tau\} \tag{2.72}$$

e

$$P\{S_{t-1} = m'; S_t = m \mid Y_1^\tau\} = P\{S_{t-1} = m'; S_t = m; Y_1^\tau\} \Big/ P\{Y_1^\tau\} \tag{2.73}$$

Associada a cada nó (estado) da treliça está a correspondente *APP* $P\{S_t = m \mid Y_1^\tau\}$ e associada a cada ramo está a correspondente *APP* $P\{S_{t-1} = m';\ S_t = m \mid Y_1^\tau\}$. Assim, o algoritmo observa a seqüência Y_1^τ e calcula as probabilidades mencionadas.

Visando facilitar a descrição do funcionamento do algoritmo, é interessante definir as probabilidades conjuntas:

$$\begin{aligned} \lambda_t(m) &= P\{S_t = m; Y_1^\tau\} \\ \sigma_t(m', m) &= P\{S_{t-1} = m'; S_t = m; Y_1^\tau\} \end{aligned} \tag{2.74}$$

Estas probabilidades são, praticamente, as *APP's* buscadas pelo algoritmo, já que, para uma dada seqüência Y_1^τ , $P\{Y_1^\tau\}$ é uma constante; para obtê-las, basta que dividamos $\lambda_t(m)$ e $\sigma_t(m', m)$ por

essa probabilidade (que é equivalente a $\lambda_t(0)$, calculado do decodificador). A obtenção, passo a passo, das probabilidades da eq. 2.74 é mostrada a seguir.

Definindo as funções probabilidade abaixo,

$$\begin{aligned} \alpha_t(m) &= P\{S_t = m; Y_1^t\} \\ \beta_t(m) &= P\{Y_{t+1}^\tau \mid S_t = m\} \\ \gamma_t(m', m) &= P\{S_t = m; Y_t \mid S_{t-1} = m'\} \end{aligned} \tag{2.75}$$

Pode-se adequar, através de algumas manipulações matemáticas, $\lambda_t(m)$ e $\sigma_t(m', m)$, de forma que possam ser rescritas utilizando as eqs. 2.75.

Para $\lambda_t(m)$, tem-se:

$$\begin{aligned} \lambda_t(m) &= P\{S_t = m; Y_1^t\} \cdot P\{Y_{t+1}^\tau \mid S_t = m; Y_1^t\} \\ &= \alpha_t(m) \cdot P\{Y_{t+1}^\tau \mid S_t = m\} \\ &= \alpha_t(m) \cdot \beta_t(m) \end{aligned} \tag{2.76}$$

Na equação anterior, na passagem do meio, foi utilizada a própria definição de um processo de *Markov* que diz que eventos passados não têm influência no futuro, se o presente é especificado, ou seja, se S_t é conhecido, eventos depois deste instante não são dependentes de Y_1^t.

Similarmente, para $\sigma_t(m', m)$:

$$\begin{aligned} \sigma_t(m', m) &= P\{S_{t-1} = m'; Y_1^{t-1}\} \cdot P\{S_t = m; Y_t \mid S_{t-1} = m'\} \cdot P\{Y_{t+1}^\tau \mid S_t = m\} \\ &= \alpha_{t-1}(m') \cdot \gamma_t(m', m) \cdot \beta_t(m) \end{aligned} \tag{2.77}$$

Para $t = 1, \ldots, \tau$

$$\begin{aligned} \alpha_t(m) &= \sum_{m'=0}^{M-1} P\{S_{t-1} = m'; S_t = m; Y_1^t\} \\ &= \sum_{m'} P\{S_{t-1} = m'; Y_1^{t-1}\} \cdot P\{S_t = m; Y_t \mid S_{t-1} = m'\} \\ &= \sum_{m'} \alpha_{t-1}(m') \cdot \gamma_t(m', m) \end{aligned} \tag{2.78}$$

Novamente, no meio do desenvolvimento da equação, é utilizada a propriedade de que eventos, depois do instante $t-1$, não são influenciados pela seqüência Y_1^{t-1} se S_{t-1} é conhecido. Para $t = 0$, tem-se as condições limiares

$$\alpha_0(0) = 1 \qquad e \qquad \alpha_0(m) = 0, \qquad para \ m \neq 0 \tag{2.79}$$

De maneira análoga, para $t = 1, \ldots, \tau - 1$

$$\begin{aligned}\beta_t(m) &= \sum_{m'=0}^{M-1} P\{S_{t+1} = m'; Y_{t+1}^{\tau} \mid S_t = m\} \\ &= \sum_{m'} P\{S_{t+1} = m'; Y_{t+1} \mid S_t = m\} \cdot P\{Y_{t+2}^{\tau} \mid S_{t+1} = m'\} \\ &= \sum_{m'} \beta_{t+1}(m') \cdot \gamma_{t+1}(m, m')\end{aligned} \tag{2.80}$$

Para $t = \tau$, as condições limite são:

$$\beta_{\tau}(0) = 1 \qquad e \qquad \beta_{\tau}(m) = 0, \qquad para\ m \neq 0 \tag{2.81}$$

O cálculo, para cada instante de tempo, é feito de forma recursiva, como pode ser observado na manipulação das eqs. 2.75, originando as eqs. 2.78 e 2.80.

Por último, o desenvolvimento de $\gamma_t(m', m)$ resulta em:

$$\begin{aligned}\gamma_t(m', m) &= \sum_{X} P\{S_t = m \mid S_{t-1} = m'\} \cdot P\{X_t = X \mid S_{t-1} = m', S_t = m\} \cdot P\{Y_t \mid X\} \\ &= \sum_{X} p_t(m \mid m') \cdot q_t(X \mid m', m) \cdot R(Y_t \mid X)\end{aligned} \tag{2.82}$$

onde o somatório é feito sobre todas as possibilidades de símbolos de saída X.

Tendo-se exposto as equações que compõem o algoritmo, este pode ser resumido nos seguintes passos:

1) $\alpha_0(m)$ e $\beta_t(m)$ são inicializados, segundo as eqs. 2.79 e 2.81, para $m = 0, \ldots, M-1$.
2) À medida que Y_t é recebido, o decodificador calcula $\gamma_t(m', m)$, pela eq. 2.82, e $\alpha_t(m)$, pela eq. 2.78. Os valores de $\alpha_t(m)$ são armazenados para todo t e m.
3) Depois que toda a seqüência Y_1^{τ} tiver sido recebida, $\beta_t(m)$ é calculado recursivamente pela eq. 2.80. Após estes cálculos, $\beta_t(m)$ pode ser multiplicado pelos valores apropriados de $\alpha_t(m)$ e $\gamma_t(m', m)$ de maneira a obter $\lambda_t(m)$ e $\sigma_t(m', m)$ através das eqs. 2.76 e 2.77.

O objetivo da seção tem sido expor algumas técnicas de decodificação para códigos convolucionais. Com este intuito, foi feita, primeiramente, a descrição da decodificação de *Viterbi*. Como o algoritmo BCJR foi desenvolvido, de uma maneira geral, para códigos lineares, o próximo passo é mostrar sua aplicação para este tipo específico de códigos.

Considere um codificador convolucional binário de taxa $R = k/n$. A entrada do codificador no instante t é a seqüência $I_t = (i_t^{(1)}, \ldots, i_t^{(k)})$ e a saída correspondente é a seqüência X_t

$= (x_t^{(1)}, \ldots, x_t^{(n)})$. O codificador pode ser implementado por k registradores de deslocamento, cada um de comprimento m, onde os estados associados são simplesmente os conteúdos desses registradores, ou seja, as m seqüências de entrada mais recentes

$$S_t = (s_t^{(1)}, \ldots, s_t^{(km)}) = (I_t, I_{t-1}, \ldots, I_{t-m+1}) \tag{2.83}$$

Por convenção, o codificador inicia-se no estado nulo S_0. Como visto anteriormente, uma seqüência de informação I_1^T, entrada do codificador, é seguida por m entradas nulas, $I_{T+1}^{\tau} = 0, 0, \ldots, 0$ onde $\tau = T + m$, de forma a levar o codificador a terminar também no estado nulo. A título de exemplo, seja uma treliça para um código de taxa ½ , com $m = 2$ e $\tau = 6$. Geralmente, as seqüências de entrada são ditas serem igualmente prováveis para $t \leq T$. Assim, já que há 2^k possibilidades de transição, na saída de cada estado, as probabilidades de transição da treliça, $p_t(m', m)$, são iguais a 2^{-k} para todas estas transições. Para $t > T$, apenas uma transição é possível na saída de cada estado (região de retorno ao estado nulo), e, portanto, $p_t(m', m) = 1$. As saídas X_t são funções determinísticas das transições; dessa forma, para cada transição, há uma distribuição de probabilidade $q_t(X \mid m', m)$ $0 - 1$ sobre o alfabeto de n-uplas binárias. Se a seqüência de saída é enviada por um canal *DMC* com probabilidades de transição de símbolos dadas por $r(\cdot \mid \cdot)$, as correspondentes probabilidades de transição de bloco são

$$R(Y_t \mid X_t) = \prod_{j=1}^{n} r(y_t^{(j)} \mid x_t^{(j)}) \tag{2.84}$$

onde $Y_t = (y_t^{(1)}, \ldots, y_t^{(n)})$ é o bloco recebido no instante t. A fim de minimizar a probabilidade de erro de bit, os dígitos de entrada $i_t^{(j)}$ mais prováveis são determinados da seqüência recebida Y_1^{τ}. Partindo do princípio que os $\lambda_t(m)$ já foram calculados tal como mostrado, define-se $A_t^{(j)}$ como o conjunto de estados S_t tal que $s_t^{(j)} = 0$. Então, pela eq. 2.83, tem-se que

$$s_t^{(j)} = i_t^{(j)}, \qquad j = 1, \ldots, k$$

o que implica em

$$P\{i_t^{(j)} = 0; Y_1^{\tau}\} = \sum_{S_t \in A_t^{(j)}} \lambda_t(m)$$

Normalizando por $P\{Y_1^{\tau}\} = \lambda_t(0)$, tem-se $P\{i_t^{(j)} = 0 \mid Y_1^{\tau}\}$. A decisão final é tomada da seguinte maneira: decodifica-se $i_t^{(j)} = 0$ se $P\{i_t^{(j)} = 0 \mid Y_1^{\tau}\} \geq 0.5$ (limiar pré-estabelecido); caso contrário,

$i_t^{(j)} = 1$. Se o interesse é determinar a APP dos dígitos de saída codificados, ou seja, $P\{x_t^{(j)} = 0 \mid Y_1^\tau\}$, um desenvolvimento semelhante é seguido. Agora, seja $B_t^{(j)}$ o conjunto de transições $S_{t-1} = m' \rightarrow S_t = m$ tal que o j-ésimo dígito de saída $x_t^{(j)}$, na transição, seja zero. Então,

$$P\{x_t^{(j)} = 0; Y_1^\tau\} = \sum_{(m',m)\in B_t^{(j)}} \sigma_t(m',m)$$

Que pode ser normalizado, analogamente ao procedimento anterior, de forma a obter $P\{x_t^{(j)} = 0 \mid Y_1^\tau\}$.

Uma limitação do algoritmo é que ele requer grande capacidade de armazenamento e complexidade computacional considerável. Para se ter uma idéia, o tamanho dos armazenamentos cresce exponencialmente com km e linearmente com o comprimento dos blocos; ainda, o número de cálculos efetuados na determinação de $\alpha_t(m)$ (ou $\beta_t(m)$), para cada t, é $M \cdot 2^k$ multiplicações e M adições de 2^k números cada. Uma passagem interessante, que visa à diminuição do número de variáveis armazenadas, é que o cálculo de $\gamma_t(m', m)$ é relativamente simples e, por esta razão, para efeitos práticos, é mais fácil (e mais sensato) recalcular $\gamma_t(m', m)$ no passo (3) do que armazená-lo, juntamente com $\alpha_t(m)$, no passo (2).

2.3.2.1. Algoritmo APP (BCJR) para modulação DPSK

O algoritmo BCJR é utilizado, no receptor do sistema proposto (que será detalhado no capítulo 4), nos dois estágios de decodificação: no demodulador e no decodificador. Antes, porém, da simulação completa, o sistema foi simulado, a princípio, sem codificação e, posteriormente, sem modulação, a fim de que fosse possível avaliar o desempenho deste algoritmo em cada bloco do receptor separadamente. Assim, para efeitos comparativos e sob as mesmas condições de simulação, foram empregados o gráfico da probabilidade de erro de bit obtido através de um demodulador 4-DPSK que utiliza regra MAP (de [15]) (método 1) e aquele obtido, através das simulações realizadas, com um demodulador APP, utilizando o algoritmo em questão (método 2). A idéia era avaliar se o uso do segundo método traria algum benefício ao sistema. Analisando, portanto, a Fig. 2.15, que é a representação destes gráficos, pode-se notar que, de fato, há um pequeno ganho (em média 0.5 dB), em relação ao primeiro método. O próximo passo seria o de verificar a vantagem do algoritmo num bloco decodificador.

Esta investigação não é mostrada, mas pode ser encontrada facilmente na literatura ([1], [3] e [4]), onde se percebe, claramente, o bom desempenho deste arranjo de decodificação.

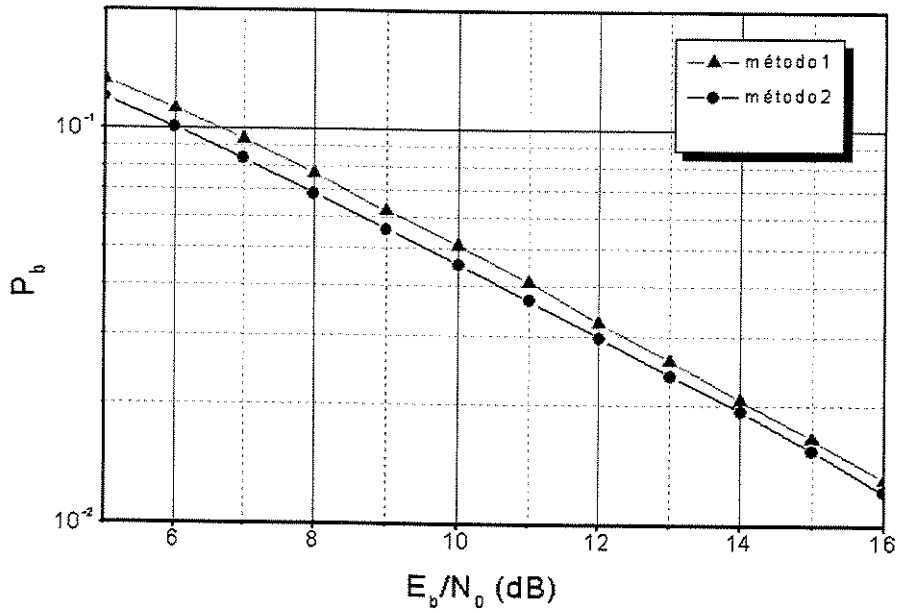

Fig. 2.15 – Gráficos de probabilidade de erro de bit (P_b) verus relação sinal-ruído em dB (E_b/N_0).

Por conseguinte, após a verificação da possibilidade de ganhos individuais, o sistema estava pronto para ser simulado, segundo a proposta central deste trabalho, ou seja, utilizando um processo iterativo que interliga os dois blocos juntamente com as iterações inerentes ao decodificador turbo.

Capítulo 3

Codificação/Decodificação turbo

Este capítulo procura oferecer uma breve descrição do tipo de codificação e decodificação utilizada nas simulações da rede proposta, enfatizando alguns conceitos, tais como: informação extrínseca e interleaver.

3.1. A Codificação turbo

Os codificadores turbo foram introduzidos por *Berrou, Glavieux* e *Thitimajshima* como uma grande inovação no cenário da codificação de baixa complexidade e alto ganho. Eles têm por pretensão alcançar o desempenho de correção de erros próximo ao limite de *Shannon,* através da aplicação de codificadores componentes relativamente simples e grandes *interleavers* (entrelaçadores).

Um codificador turbo é genericamente formado por uma concatenação de dois ou mais codificadores convolucionais sistemáticos recursivos (RSC) separados por um *interleaver*; em outras palavras, eles são construídos, aplicando-se dois ou mais codificadores componentes a diferentes versões entrelaçadas da mesma seqüência de informação. Para a rede proposta no capítulo 4, foi utilizado um codificador turbo de concatenação paralela; por tal razão, este capítulo ater-se-á, exclusivamente, a este tipo de arranjo. A Fig. 3.1 é um exemplo de codificador turbo de taxa $R = ¼$ que é formado pela concatenação paralela de dois RSC's, ambos de taxa $R_1 = ½$ [3]. Apesar dos codificadores, no exemplo, serem idênticos, em geral, eles podem não o ser e uma maneira de atingir a taxa desejada para o turbo é através do puncionamento destes codificadores componentes. O exemplo da Fig. 3.1 é de interesse e foi escolhido para descrever a codificação/decodificação turbo, por ser o esquema de codificação utilizado, na rede proposta, descrito no próximo capítulo.

Na figura, o primeiro codificador convolucional age diretamente sobre a seqüência de informação $u = (u_1, \ldots, u_L)$, produzindo as seqüências codificadas de saída x_{1i} e x_{1p}. O segundo

codificador age sobre uma versão embaralhada dos bits de informação, u', também de comprimento L, produzindo as seqüências codificadas x_{2i} e x_{2p}. O *interleaver* será tratado mais adiante.

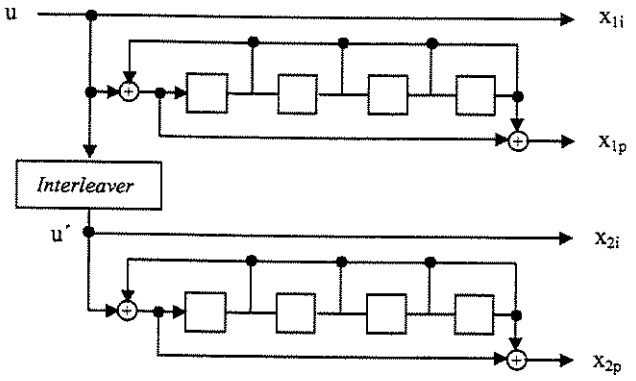

Fig. 3.1 – Esquema de um codificador turbo de taxa $R = 1/4$

Do capítulo anterior, sabe-se que o codificador convolucional necessita de uma seqüência de terminação composta por zeros, de forma que a treliça correspondente possa retornar ao estado nulo. Com este propósito, em [3], é sugerido um método simples capaz de realizar este processo de terminação que leva em consideração a utilização de codificadores recursivos e que, por isso, foi utilizado em nossas simulações. Nele, cada codificador convolucional age, normalmente, nos primeiros L ciclos do relógio, que correspondem à entrada dos L bits de informação. Terminada a seqüência de entrada, o codificador é "desligado" de sua entrada de bits, isto é, a entrada referente aos bits de informação é desconectada do somador, e a entrada proveniente da recursão é conectada em seu lugar pelos m ciclos finais. Isto faz com que, automaticamente, as entradas do somador se tornem iguais, liberando, portanto, zeros para o codificador que permitem o processo de terminação. O método pode ser melhor visualizado através da Fig. 3.2, que representa um dos codificadores convolucionais da Fig. 3.1, onde a posição "A" da chave indica a ação normal do codificador e a posição "B", os m ciclos seguintes.

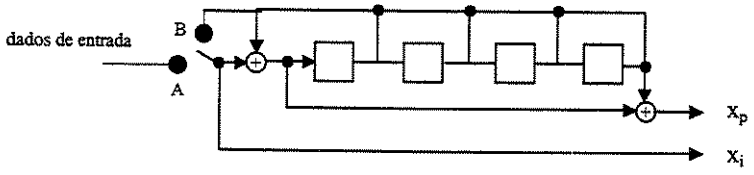

Fig. 3.2 – Exemplificação do método de terminação da treliça

3.1.1. Utilização de RSC´s como forma de melhorar o desempenho do codificador turbo

Como já mencionado, o desempenho de um codificador está ligado a suas propriedades de distância. Analisando, pois, o codificador da Fig. 3.1, observa-se que ele é responsável por dois conjuntos de palavras- código: $x_1 = (x_{1i}, x_{1p})$ e $x_2 = (x_{2i}, x_{2p})$, cujos pesos podem ser calculados. O desafio é, portanto, encontrar os pares de palavras-código, de cada conjunto, induzidos por um particular *interleaver*. Intuitivamente, seria interessante evitar o emparelhamento de palavras-código de baixo peso de um codificador, com palavras-código de baixo peso do outro codificador, de forma a maximizar a distância livre, d_{free}, efetiva do código, que é o objetivo a ser alcançado no projeto do codificador turbo. Alguns destes emparelhamentos podem ser descartados pelo próprio projeto do *interleaver*. Outros, entretanto, podem ser intrínsecos ao codificador e, portanto, de difícil descarte. Por exemplo, se os codificadores são não recursivos, a palavra-código de baixo peso, gerada pela seqüência de entrada $u = (00 \ldots 00100 \ldots 00)$, de peso unitário, sempre aparecerá novamente no segundo codificador, independente da escolha do *interleaver*, o que, fatalmente, tornará menor o valor de d_{free}. Porém, ao utilizar codificadores recursivos, a seqüência de entrada de peso unitário gera uma IIR (resposta de impulso infinito) e, dessa forma, não permite que a palavra código de peso mínimo figure entre as saídas do codificador. O peso da saída codificada é mantido finito, apenas, por causa da terminação da treliça, que força a seqüência codificada a terminar no estado nulo. Estes fatores motivam o uso de codificadores recursivos, onde o ponto chave é a recursividade e não o fato dos codificadores serem sistemáticos. É a propriedade IIR dos códigos RSC que protege contra aquelas codificações de baixo peso que não podem ser remediadas pelo *interleaver*. [3], [4].

3.2. A decodificação turbo

Em geral, para qualquer utilização de codificadores, em que sejam aplicados isoladamente, a ação final do decodificador é a de fornecer uma decisão abrupta sobre os bits (ou símbolos) decodificados. Entretanto, quando um esquema de concatenação é utilizado, como nos codificadores turbo, o algoritmo de decodificação não deve se limitar a passar decisões abruptas

entre os decodificadores componentes do turbo, de maneira a permitir que o sistema possa trabalhar de forma apropriada. No intuito de melhor explorar a informação recebida de cada decodificador, o algoritmo deve promover uma troca de decisões suaves, ao invés de abruptas. Assim, para um sistema com dois codificadores componentes, por exemplo, como é o caso do codificador da Fig. 3.1, o conceito por trás da decodificação turbo é passar decisões suaves da saída de um decodificador para a entrada do outro, repetindo este processo diversas vezes, isto é, fazendo iterações, de forma a produzir melhores decisões [4].

O processo de decodificação é, portanto, baseado num algoritmo iterativo no qual cada decodificador componente utiliza a informação extrínseca (que será tratada na seção 3.2.1) produzida pelo outro decodificador no passo anterior. Este processo iterativo é possível, através do uso de decodificadores que utilizam decisões suaves, ou seja, cujas saídas consistam de estimações suaves da seqüência de informação. A estes propósitos encaixa-se o BCJR, descrito no capítulo anterior: um algoritmo largamente utilizado e que, de fato, foi empregado nas simulações do capítulo 4.

É importante ressaltar que a decodificação turbo é próxima da ótima, apesar do algoritmo BCJR, em cada decodificador, proporcionar uma decodificação ótima. Isto acontece, porque, de acordo com a estrutura do codificador turbo da Fig. 3.1, a regra de decodificação empregada utiliza as duas observações do canal, y_1 e y_2, separadamente, isto é, cada decodificador calcula as probabilidades *a posteriori* (APP's), observando o y referente ao codificador associado. Isto é feito, para evitar a complexidade embutida na regra ótima de observação conjunta.

A Fig. 3.3 mostra o esquema da decodificação turbo, bem como os blocos constituintes associados, para o codificador da Fig. 3.1.

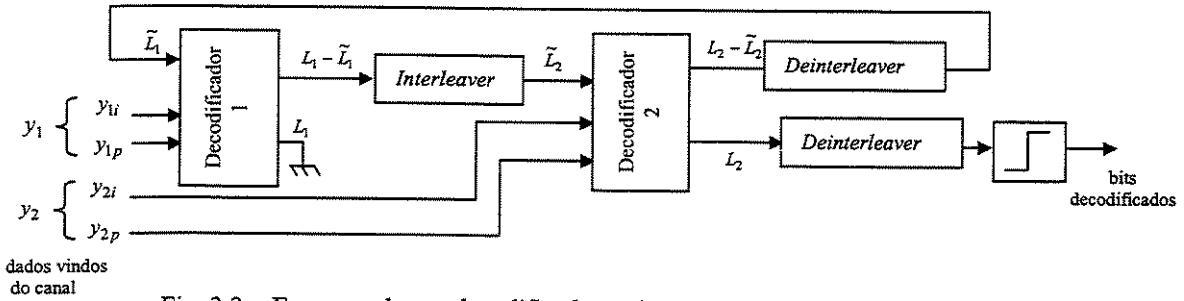

Fig. 3.3 – Esquema de um decodificador turbo para o codificador da Fig. 3.1

A informação proveniente do canal alimenta uma das entradas de cada decodificador. A outra entrada, dita informação *a priori*, é alimentada com os resultados provenientes da saída do outro decodificador e é ela a responsável pela interação entre os dois blocos. Como saída, cada decodificador libera probabilidades *a posteriori* relativas aos bits de informação u que são, por sua vez, convertidas em razões *loglikelihood*, L, (descritas na próxima seção). Estas razões são, então, subtraídas, na saída de cada decodificador, da respectiva entrada *a priori*, dando origem à informação extrínseca, $L - \tilde{L}$, que alimenta a entrada relativa à informação *a priori* do próximo decodificador. O processo se repete por um número pré-estabelecido de iterações. Neste instante, a decodificação é finalizada e o decodificador 2 fornece as probabilidades *a posteriori* dos bits de informação u, que são desentrelaçadas pelo *deinterleaver* interno, e tem-se a decisão final, em relação a um limiar pré-estabelecido.

Nota-se, ainda, a presença dos blocos *interleaver* e *deinterleaver* (ação inversa). Eles se fazem necessários por dois motivos: a informação resultante do decodificador 1, que passa para o decodificador 2, é entrelaçada, pelo *interleaver*, pelo fato de, na codificação, o codificador 2 receber, como entrada, os bits de informação entrelaçados; no caso do decodificador 1, a informação vinda do decodificador 2 tem que ser desentrelaçada, uma vez que o codificador 1 não observa nenhum entrelaçamento na sua entrada, como pode ser constatado na Fig. 3.1.

3.2.1. Informação extrínseca

Na decodificação turbo, o algoritmo *MAP* (no caso o BCJR) fornece, como já foi mencionado, probabilidades *a posteriori* $P(u \mid y_i, \tilde{u}_i)$, $i = 1, 2$, dos bits de informação u, que assumem valores equiprováveis 0 e 1, onde $\tilde{u}_1$ é fornecido pelo decodificador 2 e $\tilde{u}_2$, pelo decodificador 1. As quantidades $\tilde{u}_i$ correspondem a "novas estimações dos dados" e podem ser utilizadas para gerar probabilidades *a priori*, da seqüência de informação, em cada decodificador.

As probabilidades *a posteriori* são, então, convertidas em razões *loglikelihood*, L, representadas como:

$$L(t) = \log \frac{P(u_t = 1 \mid y, \tilde{u})}{P(u_t = 0 \mid y, \tilde{u})} \qquad t = 1, \ldots, L \tag{3.1}$$

onde o sinal de $L(t)$ é uma estimação, $\hat{u}$, de u; a magnitude $|L(t)|$ é a confiabilidade desta estimação[3]; $\boldsymbol{y}$ é a informação recebida do canal e t é a ordem do bit na seqüência de comprimento L (*).

A Fig. 3.4 mostra o esquema de um dos decodificadores MAP da Fig. 3.3, enfatizando a natureza de suas entradas e saídas, bem como a relação entre estas variáveis e as razões *log-likelihood*:

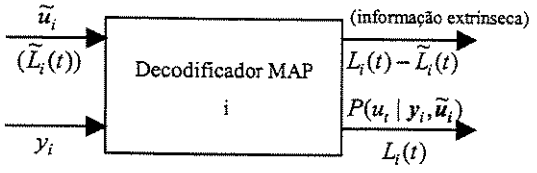

Fig. 3.4 – Esquema do decodificador MAP

Sabe-se que um princípio fundamental da decodificação iterativa é não realimentar um decodificador com informação que tenha sido originada dele próprio. Esta recomendação é, literalmente, seguida no processamento turbo, onde, a cada realimentação de um decodificador para o outro, a saída produzida pelo "realimentador" é subtraída da informação *a priori* introduzida nele (informação esta que provém, justamente, do decodificador que está para ser realimentado). No intuito de esclarecer a citação acima, o decodificador 2 é tomado como exemplo. O esquema da Fig. 3.3 pode ser utilizado, em conjunto com as explicações que se seguem, no intuito de melhorar a visualização das variáveis envolvidas.

Baseado em [3], quando o decodificador 2 gera $P(u_t \mid \tilde{u}_{1,t})$ ou de maneira equivalente $\log(P(u_t = 1 \mid \tilde{u}_{1,t}) / P(u_t = 0 \mid \tilde{u}_{1,t}))$ - que corresponde a uma confiabilidade do bit de informação - para o decodificador 1, esta quantidade não deve incluir a contribuição devido a $\tilde{u}_{2,t}$, uma vez que este valor é fornecido, justamente, por este decodificador e, portanto, já é conhecido (e já foi computado) por ele. Esta adição seria, então, uma redundância ao sistema. Partindo deste princípio, a razão acima deveria ser escrita como:

$$\log \frac{P(u_t = 1 \mid \tilde{u}_{1,t})}{P(u_t = 0 \mid \tilde{u}_{1,t})} = \log \frac{P(u_t = 1 \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})}{P(u_t = 0 \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})} \tag{3.2}$$

(*) É válido ressaltar que, mesmo correndo o risco de confundir o leitor com o uso da mesma notação, para designar variáveis diferentes, decidiu-se por mantê-la nestes termos no intuito de seguir a nomenclatura empregada no capítulo 2 – onde utilizou-se L para representar o comprimento da seqüência de informação – e no capítulo 4 – onde esta mesma letra foi utilizada, na descrição da rede proposta, para indicar as razões *log-likelihood*. No entanto, nota-se que, analisando o contexto, não é difícil discernir sobre qual variável a letra se refere.

Para calcular a eq. 3.2, a probabilidade *a posteriori* obtida no decodificador 2 (de acordo com a Fig. 3.4), é reescrita, através de manipulações, utilizando o Teorema de *Bayes*, como:

$$P(u_t \mid y_2, \tilde{u}_2) \triangleq P(u_t \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})$$
$$= \frac{P(\tilde{u}_{2,t} \mid u_t, y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L}) \cdot P(u_t \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})}{P(\tilde{u}_{2,t} \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})} \tag{3.3}$$

Observando que $\tilde{u}_{2,t}$ foi gerado pelo decodificador 1 e o *interleaver* foi utilizado para entrelaçar a informação que entra no decodificador 2, pode-se concluir que esta quantidade ($\tilde{u}_{2,t}$) depende muito fracamente de y_2 e $\tilde{u}_{2,j}$, para $j \neq t$, o que permite o uso da seguinte aproximação:

$$P(\tilde{u}_{2,t} \mid u_t, y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L}) \approx P(\tilde{u}_{2,t} \mid u_t) = 2 \cdot P(u_t \mid \tilde{u}_{2,t}) P(\tilde{u}_{2,t}) \tag{3.4}$$

onde novamente foi utilizado o Teorema de *Bayes* e também o fato dos bits de informação serem equiprováveis. Substituindo, pois, a eq. 3.4 em 3.3, tem-se que:

$$P(u_t \mid y_2, \tilde{u}_2) = \frac{2 \cdot P(u_t \mid \tilde{u}_{2,t}) P(\tilde{u}_{2,t}) \cdot P(u_t \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})}{P(\tilde{u}_{2,t} \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})}$$

ou

$$P(u_t \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L}) = \frac{P(u_t \mid y_2, \tilde{u}_2) \cdot P(\tilde{u}_{2,t} \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})}{2 \cdot P(u_t \mid \tilde{u}_{2,t}) P(\tilde{u}_{2,t})} \tag{3.5}$$

Ainda, substituindo a eq. 3.5 na eq. 3.2, obtém-se

$$\log \frac{P(u_t = 1 \mid \tilde{u}_{1,t})}{P(u_t = 0 \mid \tilde{u}_{1,t})} = \log \left[\frac{P(u_t = 1 \mid y_2, \tilde{u}_2) \cdot P(\tilde{u}_{2,t} \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})}{2 \cdot P(u_t = 1 \mid \tilde{u}_{2,t}) P(\tilde{u}_{2,t})} \right.$$
$$\left. \cdot \frac{2 \cdot P(u_t = 0 \mid \tilde{u}_{2,t}) P(\tilde{u}_{2,t})}{P(u_t = 0 \mid y_2, \tilde{u}_2) \cdot P(\tilde{u}_{2,t} \mid y_2, \tilde{u}_{2,1}, \ldots, \tilde{u}_{2,t-1}, \tilde{u}_{2,t+1}, \ldots, \tilde{u}_{2,L})} \right]$$
$$= \log \left[\frac{P(u_t = 1 \mid y_2, \tilde{u}_2)}{P(u_t = 0 \mid y_2, \tilde{u}_2)} \cdot \frac{P(u_t = 0 \mid \tilde{u}_{2,t})}{P(u_t = 1 \mid \tilde{u}_{2,t})} \right] \tag{3.6}$$

Definindo, agora,

$$\tilde{L}_i(t) = \log \frac{P(u_t = 1 \mid \tilde{u}_{i,t})}{P(u_t = 0 \mid \tilde{u}_{i,t})}, \qquad i = 1, 2 \tag{3.7}$$

Finalmente, substituindo as eqs. 3.1 e 3.7 na eq. 3.6, chega-se à expressão,

$$\widetilde{L}_1^{(k)}(t) = L_2^{(k)}(t) - \widetilde{L}_2^{(k-1)}(t) \tag{3.8}$$

que representa, exatamente, a saída do decodificador 2, após o *deinterleaver* (Fig. 3.3), na k-ésima iteração (a iteração é representada, aqui, apenas, a título de ilustração). $\widetilde{L}_2(t)$ aparece como pertencente à iteração anterior, porque, analisando a segunda parcela da segunda igualdade da eq. 3.6, nota-se que ele se refere à variável $\widetilde{u}_{2,t}$ que, sabe-se, é fornecida pelo decodificador 1; assim, como a análise atual é em relação à saída do decodificador 2 (e, conseqüentemente, nova entrada no decodificador 1), esta quantidade só pode ter se originado na iteração anterior, quando o decodificador 1 estava em funcionamento. Similarmente, pode-se obter $\widetilde{L}_2^{(k)}(t) = L_1^{(k)}(t) - \widetilde{L}_1^{(k-1)}(t)$ na saída do decodificador 1, após o *interleaver*.

A quantidade representada na eq. 3.8 é conhecida como informação extrínseca e pode ser visualizada nas Figuras 3.3 e 3.4. Basicamente, ela representa uma extração de informação redundante.

A partir dessas definições, as probabilidades *a priori*, na entrada de cada decodificador, podem ser calculadas, através de uma transformação inversa aplicada à eq. 3.7, como

$$P(u_t = 0 \mid \widetilde{u}_{i,t}) = \frac{e^{\widetilde{L}_i(t)}}{1 + e^{\widetilde{L}_i(t)}} = 1 - P(u_t = 1 \mid \widetilde{u}_{i,t}), \qquad i = 1, 2 \tag{3.9}$$

É válido ressaltar que, por definição, a decisão final do decodificador é feita em relação ao bit 0. Por isso, a informação *a priori*, na eq. 3.9, destaca a probabilidade referente ao bit de informação 0.

3.3. *Interleaver*

O *interleaver* é um bloco componente adicional da transmissão e recepção de sinais que age de forma a embaralhar a informação transmitida, visando a aumentar o desempenho final. Ele pode ser encontrado sob várias formas, mas duas delas são mais freqüentes: *interleaver* de bloco e *interleaver* aleatório.

Num cenário de transmissão sem codificação, o ruído do canal pode causar erros de bit aleatórios, na saída do receptor, que são mais ou menos isolados, ou seja, não adjacentes. Por outro lado, quando uma codificação é utilizada, a redundância do código permite ao codificador corrigir os erros, de tal forma que a saída decodificada seja praticamente livre de erros. Entretanto, em algumas aplicações, há a ocorrência de grandes pulsos de ruído de canal, tal como em canais telefônicos com ruído de chaveamento, onde, se as técnicas usuais de codificação forem usadas, surtos de erros poderão ocorrer na saída do decodificador, pelo fato de os surtos de ruído terem duração maior que o "tempo de redundância" do código. Estas situações podem ser melhoradas pelo uso de *interleavers*.

No cenário específico da codificação turbo, o primeiro propósito do *interleaver* aleatório é mapear seqüências de entrada de baixo peso, que geram seqüências de saída de baixo peso no primeiro codificador, em outras seqüências de entrada de baixo peso que gerem, por sua vez, seqüências de saída de peso alto no segundo decodificador[17]. O interesse é, como já mencionado, fazer com que as seqüências codificadas resultantes tenham peso alto. *Interleavers* de bloco convencionais não são muito eficientes, neste sentido, para certas seqüências de entrada; assim, *interleavers* com um mapeamento mais aleatório são de maior interesse, embora permutações geradas, aleatoriamente, possam, algumas vezes, falhar na missão de separar, de maneira suficiente, pares de bit em particular. Em alguns casos, a saída do decodificador MAP para códigos convolucionais simula um surto de erro de canal – eventos de erro de treliça que se traduzem em surtos de erro de bits. Assim, seria desejável que o *interleaver* fosse capaz de quebrar estes surtos, distribuindo, portanto, os erros de maneira aleatória, o que faria o decodificador receber, por conseqüência, probabilidades de entrada com erros "aleatórios" (mais espalhados). Enquanto um *interleaver* de bloco convencional é eficiente em parar tais surtos, aquele do tipo aleatório, quando utilizado em codificadores turbo, tem se mostrado demasiadamente superior em termos de desempenho. Ao passo que um codificador turbo, com um *interleaver* de bloco, tem, em geral, uma distância livre maior do que aquele que utiliza um *interleaver* aleatório, a multiplicidade de eventos de erro, que ocorrem na distância mínima, é tipicamente menor, quando se utiliza estes últimos. Se eventos de erros a grandes distâncias são considerados, a multiplicidade de tais eventos, utilizando *interleaver* aleatório, cresce lentamente com a distância, enquanto que, com o uso de *interleaver* de bloco, este aumento é substancial. A conclusão é que o uso de *interleavers* aleatórios leva a codificadores turbo com, talvez, distâncias

livres muito pequenas, mas com uma faixa de distância relativamente plana [17]. Isto explica, em parte, o desempenho extraordinário dos codificadores turbo em relações sinal-ruído baixas, onde eventos de erro a grandes distâncias contribuem, mais significativamente, para a probabilidade de erro.

Em nossas simulações, é utilizado o *interleaver* aleatório. O método utilizado leva em consideração o conceito de pilha [21]. A construção é feita da seguinte maneira: inicialmente, a pilha é preenchida, de forma que cada posição e o seu elemento associado coincidam. Através de um laço, as posições do *interleaver* vão sendo preenchidas, uma a uma, a partir da posição inicial. A função criada chama "aleatoriamente" um valor, v, que é associado a uma posição da pilha. O elemento correspondente àquela posição passa, agora, a ser o elemento associado à posição atual do *interleaver*. A pilha é, então, reorganizada, a partir de v, de forma que todos os valores chamados pela função correspondam sempre a elementos diferentes no *interleaver*. Na verdade, os valores gerados são pseudo-aleatórios, devido à própria limitação computacional.

3.4. Extensão da decodificação turbo para redes de decodificação ("Turbo DPSK")

Em virtude dos notáveis resultados, amplamente citados na literatura, alcançados com o uso da decodificação turbo, novas linhas de pesquisa vêm sendo sugeridas e seguidas, visando a aplicar estas contribuições em novos cenários de decodificação. Entre as aplicações existentes, destaca-se o conceito de redes de decodificação, sistemas de recepção que utilizam o que se chamou de princípio turbo ou processamento iterativo.

A rede de decodificação da Fig. 3.5 foi proposta por *P. Hoeher* e *J. Lodge* [6], em 1999, como uma estrutura de receptor apropriada para a codificação de sinais DPSK em canais estáticos ou variantes no tempo. Nela, destaca-se a proposta de um demodulador APP DPSK, derivado do algoritmo BCJR – visto no capítulo 2 – mas com algumas modificações, de forma a utilizar predição linear e processamento por sobrevivente, para estimar a resposta do canal. Outro ponto de destaque é a utilização do processamento iterativo que se torna possível graças à capacidade do demodulador em aceitar informação *a priori*, em sua entrada, e produzir saídas de confiabilidade, e à capacidade do decodificador em, também, fornecer informações de

confiabilidade em sua saída.. Assim, ambos os blocos, no receptor, funcionam, de maneira a fornecer informação, um ao outro, caracterizando, portanto, uma iteração tal como na decodificação turbo.

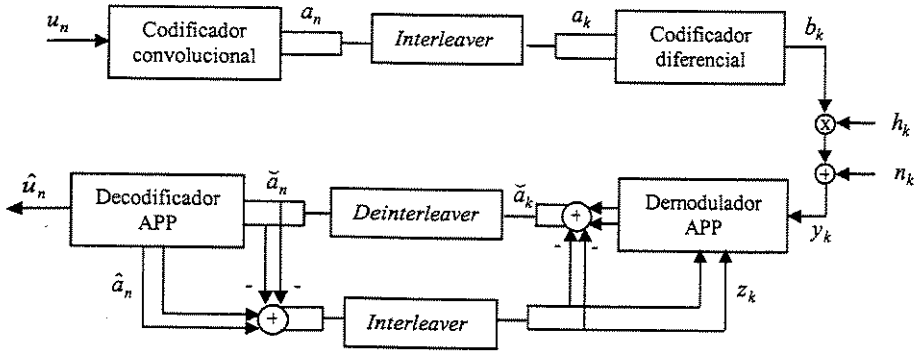

Fig 3.5 – Diagrama de blocos de uma rede de decodificação iterativa para a concatenação serial de um codificador convolucional e um modulador DPSK.

O transmissor é formado pela concatenação serial de um codificador convolucional e um modulador PSK diferencial (DPSK) separados por um *interleaver*. Na recepção, é empregado demodulação APP DPSK e decodificação/filtragem APP num processo iterativo. O funcionamento do receptor é idêntico à decodificação turbo, descrita na seção 3.2, uma vez que o decodificador e o demodulador, aqui presentes, substituem, em uma analogia, os decodificadores constituintes do turbo. Este é o motivo que faz esta interação entre os blocos do receptor ser também conhecida por princípio turbo.

No intuito de unificar as notações, é feita uma associação desta, introduzida em [6], com aquela notação já estabelecida, nesta dissertação, nas seções prévias. Seguindo a Fig. 3.5, tem-se que: u_n representa os bits de informação; a_n, os símbolos codificados; a_k é a versão entrelaçada de a_n, ou, ainda, os símbolos M-ários da entrada do modulador diferencial, onde M representa o número de estados presentes na treliça que, por sua vez, representa o codificador convolucional; b_k representa os símbolos M-ários modulados; h_k são os coeficientes do canal que dependem do modelo utilizado; n_k é uma amostra de um ruído Gaussiano complexo de média zero e variância $N_0/2$; y_k representa a saída do canal; $\breve{a}_n$ (ou na forma entrelaçada, $\breve{a}_k$) denota a entrada do decodificador APP; $\hat{a}_n$ representa as razões *log-likelihood*; z_k é a

informação *a priori* na entrada do demodulador APP e $\hat{u}_n$ é a estimação dos bits de informação. Ainda, $\hat{a}_n - \breve{a}_n$ é a informação extrínseca indicada na seção 3.3 como $L - \tilde{L}$.

Essa rede de decodificação tornou-se particularmente interessante, sendo digna de uma seção neste capítulo, por ser o ponto de partida para este trabalho. Nesta dissertação, procurou-se analisar modificações tais que proporcionassem uma alteração no cenário atual e, conseqüentemente, uma nova visão do receptor.

Capítulo 4

Rede de Codificação/Decodificação Proposta

Este capítulo tem como objetivo descrever o funcionamento da rede proposta, à medida que os blocos que o compõem são introduzidos e detalhados. Em seguida, curvas de desempenho são mostradas, como forma de analisar os possíveis ganhos obtidos com esse arranjo de receptor.

4.1. Considerações iniciais

Como foi dito anteriormente, o sistema estudado é baseado no modelo de rede de decodificação iterativa, em canais coerentes, introduzido por *Hoeher* e *Lodge* [6] (Fig. 3.5). A proposta deste trabalho foi a substituição do codificador convolucional por um codificador turbo, criando, assim, um cenário de decodificação diferente daquele anterior que propicia a combinação de duas formas diferentes de iteração: uma devido ao próprio processo iterativo, e outra inerente à decodificação turbo. O objetivo, portanto, é analisar como estas formas de iteração devem ser combinadas, de maneira que possam favorecer o desempenho do sistema.

O diagrama de blocos completo do sistema de transmissão e recepção é mostrado no final da seção (Fig 4.14). A rede foi desenhada como se fosse formada por duas ramificações, cada qual partindo da saída de um dos codificadores convolucionais. Este arranjo foi, apenas, uma forma encontrada de facilitar a visualização do processo e, até mesmo, a programação e pode ser pensada desta forma, já que os símbolos codificados são gerados de maneira independente em cada codificador. Portanto, tudo se passa como se houvesse, apenas, um ramo de transmissão e recepção, ou seja, um modulador e um demodulador 4-DPSK. Eles trabalham com as informações provenientes de cada codificador convolucional separadamente: cada bloco de informação codificado gerado por um dos codificadores é modulado e demodulado independentemente do bloco gerado pelo outro codificador.

A próxima seção destina-se à descrição da rede proposta. A fim de tornar mais clara tal descrição, ao primeiro contato, é fornecida, abaixo, a lista dos componentes/blocos utilizados:

- Codificador turbo de taxa $R = ¼$;
- *Interleaver* interno (1) ao codificador turbo de comprimento igual a 1000 bits;
- *Interleaver* externo (2) ao codificador turbo, de comprimento igual a 1000 símbolos, onde cada símbolo corresponde a dois bits;
- Modulador 4-DPSK;
- Canal não-coerente com fase aleatória variando lentamente;
- Demodulador APP utilizando algoritmo BCJR;
- Decodificador turbo utilizando algoritmo BCJR.

4.2. Descrição da rede proposta

Esta seção é dividida em duas partes: transmissão e recepção. Em cada uma delas, é abordada a operacionalidade dos blocos que as compõem. É conveniente ressaltar a importância do diagrama de blocos completo, como instrumento de auxílio ao entendimento no decorrer da seção.

4.2.1. Transmissão

4.2.1.1. Codificador turbo

O tamanho do bloco de informação (u) utilizado é de 1000 bits, gerados aleatoriamente. Dois codificadores convolucionais sistemáticos recursivos de taxa ½ e memória $M = 4$, com polinômios geradores $g_a = 21$ e $g_b = 37$ (na representação octal), são concatenados em paralelo, através de um *interleaver* (entrelaçador) interno, formando um codificador turbo de taxa ¼. O entrelaçamento é gerado através de uma permutação pseudo-aleatória (como foi descrito no final da seção 3.3) com comprimento de 1000 bits. Cada um dos codificadores convolucionais gera, como saída, 1000 símbolos de informação codificados e mais quatro símbolos extras, responsáveis por "zerar" a memória do codificador. Na verdade, cada um desses codificadores gera duas seqüências de 1004 bits (x_{wi} e x_{wp}) que são combinados, dois a dois, um

bit proveniente de x_{wi} e outro de x_{wp} (Tabela 4.1), totalizando, por conseguinte, os 1004 símbolos. O codificador turbo é essencialmente aquele mostrado na seção 3.1; para fins de comodidade, ele é redesenhado na Fig. 4.1.

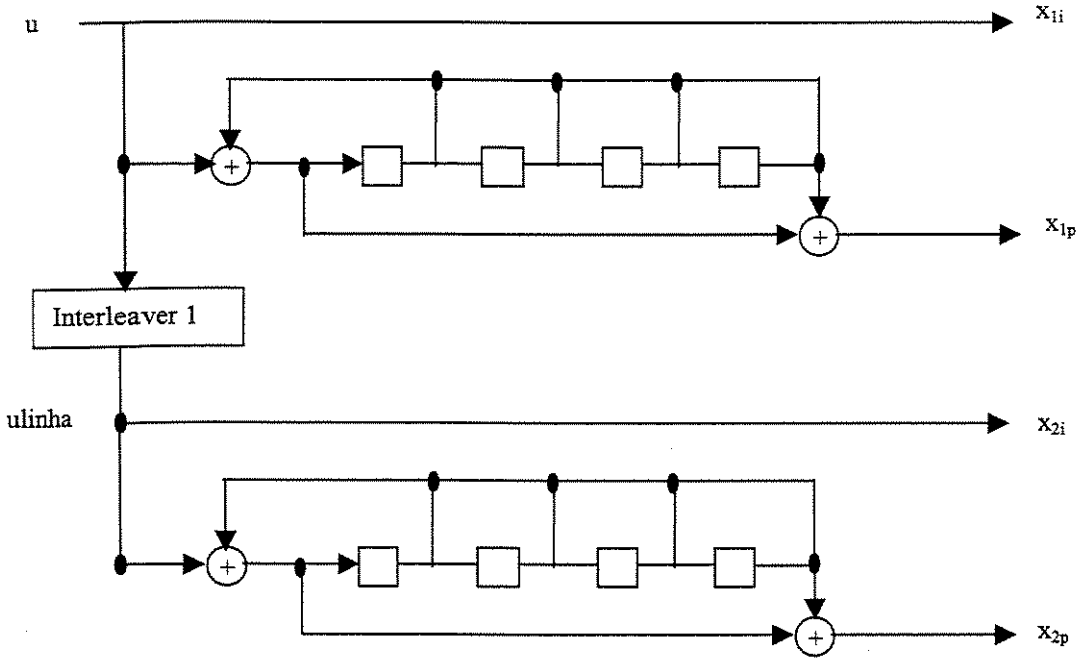

Fig. 4.1 – Codificador turbo

As saídas x_{1i} e x_{1p} formam os símbolos x_1 e as saídas x_{2i} e x_{2p}, os símbolos x_2, seguindo a Tabela 4.1:

bits		**Símbolos**
x_{wi}	x_{wp}	(x_1, x_2)
0	0	0
0	1	1
1	0	2
1	1	3

Tabela 4.1 – Conversão binária/decimal

onde: $w = 1,2$ (representação simbólica da saída de um dos codificadores convolucionais).

4.2.1.2. *Interleaver* externo

O próximo passo foi passar as saídas x_1 e x_2 pelo *interleaver* externo (Fig. 4.2), de comprimento 1000, cujo entrelaçamento também é gerado através de uma permutação pseudo-aleatória , que embaralha os símbolos de informação e não mais os bits (é importante salientar que ele não age sobre os "*tail* símbolos" – os quatro últimos símbolos responsáveis por "zerar" a memória do codificador convolucional – mas, apenas, sobre aqueles em que agirá o decodificador turbo, que são os símbolos que contêm os dados de informação). Uma das razões para a sua aplicação é a tentativa de minimizar efeitos indesejáveis relativos ao desvanecimento.

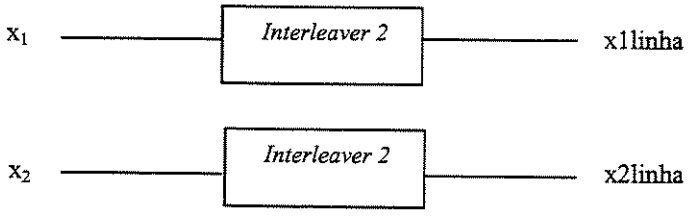

Fig. 4.2 – *Interleaver* externo

4.2.1.3. Modulador 4-DPSK

Seguindo a Fig. 4.14a, deparamos com o modulador 4-DPSK. Antes, porém, foi necessário incluir um conversor de modo a realizar uma transformação bipolar (Fig. 4.3), uma vez que a estrutura de programação do modulador requeria que a notação dos símbolos, a serem modulados, fosse dessa forma. A conversão é mostrada na Tabela 4.2.

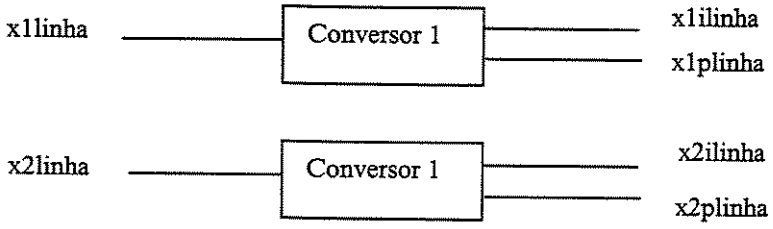

Fig. 4.3 – Conversor unipolar/bipolar

A modulação 4-DPSK foi abordada em detalhes no capítulo 2 (seção 2.2). O que se segue é um breve comentário, no intuito de tornar familiar as variáveis e símbolos empregados.

Basicamente, a modulação diferencial é feita em duas etapas: Em primeiro lugar, os símbolos codificados e embaralhados são associados a uma fase que pode assumir um dos quatro valores uniformemente distribuídos $\beta_m = 2\pi m/4;\ m = 0, 1, 2, 3$ $(0, \pi/2, \pi, 3\pi/2)$ sobre o círculo unitário, como foi mostrado na Fig. 2.8b.

O módulo de cada novo símbolo é mantido constante e igual à energia do símbolo inicial (eq. 4.1). Desta forma, apenas a fase é modulada; daí, este tipo de modulação ser chamada de *phase-shif keying* (PSK).

$$R^2 = E_S \qquad \therefore R = \sqrt{E_S} \tag{4.1}$$

onde *Es* é a energia do símbolo e *R* é o raio do círculo unitário.

Em seguida, é aplicada a codificação diferencial da seguinte maneira: a cada dois símbolos 4-PSK, em intervalos adjacentes, o modulador libera um símbolo, como sendo a diferença das duas fases anteriores, que passa, então, a conter a informação a ser transmitida.

Ainda nesta etapa, foram inseridos mais dois símbolos, além daqueles 1004 provenientes do codificador. O primeiro é um símbolo de referência inerente à própria modulação diferencial, que serve de orientação para o início dos trabalhos do demodulador (vide Fig. 2.11); o segundo é um *tail* símbolo, necessário para "zerar" a treliça do demodulador (o modulador é visto como sendo um codificador de memória $M = 1$). Ambos foram escolhidos como se fossem zero. O primeiro foi uma escolha arbitrária; já, o segundo, foi feito de tal forma que coincidisse com a inicialização do parâmetro *beta* posição *tau* (β_τ) no algoritmo BCJR (descrito no Capítulo 2).

Na Fig.4.4, pode-se observar o esquema do modulador 4-DPSK cujas saídas obedecem à regra PSK diferencial.

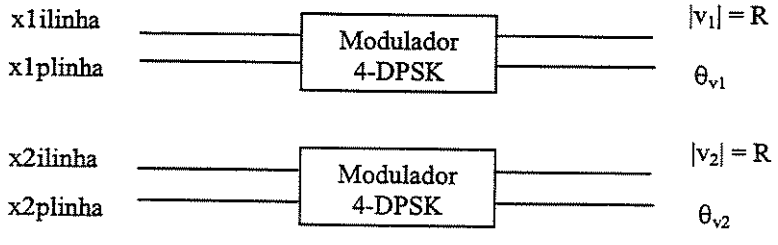

Fig. 4.4– Modulador 4-DPSK

A Tabela 4.2, abaixo, resume as notações e conversões utilizadas, até então,

Bits codificados (notação unipolar)		**Símbolos**	**Bits codificados** (notação bipolar)		**Símbolo 4-DPSK**
x_{wi}	x_{wp}	*xw / xwlinha*	*xwilinha*	*xwplinha*	
0	0	0	-1	-1	π
0	1	1	-1	1	$\pi/2$
1	0	2	1	-1	$3\pi/2$
1	1	3	1	1	0

Tabela 4.2 – Conversões utilizadas

onde: $w = 1, 2$ (representação simbólica da saída de um dos codificadores convolucionais).

4.2.2. Recepção

Como se sabe, no receptor, o sinal recebido possui, em geral, um deslocamento de fase em relação ao sinal transmitido. Esta incerteza, na fase da portadora recebida, pode estar relacionada a um dos seguintes fatores: falta de sincronização de fase nos osciladores utilizados pelos transmissores e receptores para a geração dos sinais de portadora e atraso aleatório no tempo de propagação do sinal entre o transmissor e o receptor.

Uma maneira de contornar tal problema, na recepção, é através da utilização de circuitos de sincronização, responsáveis por extrair a referência de fase do sinal recebido. Receptores, assim projetados, são chamados de coerentes. Todavia, na prática, a obtenção de uma referência de fase precisa é muito difícil; então, uma alternativa é a demodulação do sinal recebido, sem a estimação da fase e, portanto, sem a necessidade dos circuitos sincronizadores. Este tipo de receptor é dito não-coerente.

A idéia de se utilizar a modulação PSK diferencial vem justamente do fato dela não necessitar de sincronização e, assim, permitir o emprego de receptores não-coerentes, que são mais fáceis de lidar, por serem mais simples. A modulação PSK já não seria adequada para estes propósitos, pois a informação que se quer transmitir está na fase do sinal transmitido, o que requer uma boa sincronização e, conseqüentemente, uma maior complexidade de projeto. Logo,

assumindo que a fase aleatória adicionada, na transmissão, é constante em dois intervalos de símbolo, a informação desejada consiste na diferença da fase transmitida em dois intervalos adjacentes, como fora comentado na seção anterior.

Os símbolos recebidos do canal serão, então, processados através de dois estágios: o demodulador diferencial e o decodificador turbo.

Na recepção, ambos os estágios utilizam o algoritmo BCJR (seção 2.3.2), algoritmo de detecção MAP, para a decodificação; apenas, no demodulador, este algoritmo foi modificado, no intuito de permitir uma recepção não-coerente. A idéia de se usar iteração, no sistema, tornou-se possível, graças à natureza das entradas necessárias ao funcionamento do algoritmo (Fig. 4.5): uma, correspondendo à informação vinda do canal, e outra, dita informação *a priori*, responsável, diretamente, pela realimentação do processo. Desse modo, estando o algoritmo presente, tanto em um estágio quanto no outro, os resultados obtidos em um puderam ser utilizados como informação *a priori* no outro e vice-versa, concretizando, assim, o processo iterativo. O sistema utiliza, portanto, duas linhas de iterações: uma externa ao turbo, referente ao processo iterativo entre os estágios demodulador/decodificador, e outra interna, proveniente do próprio conceito de decodificação turbo (seção 3.2).

Fig. 4.5 – Algoritmo BCJR

4.2.2.1. Canal não-coerente

A transmissão dos sinais 4-DPSK se dá através do canal com desvanecimento *Rayleigh*, observado na Fig. 4.6.

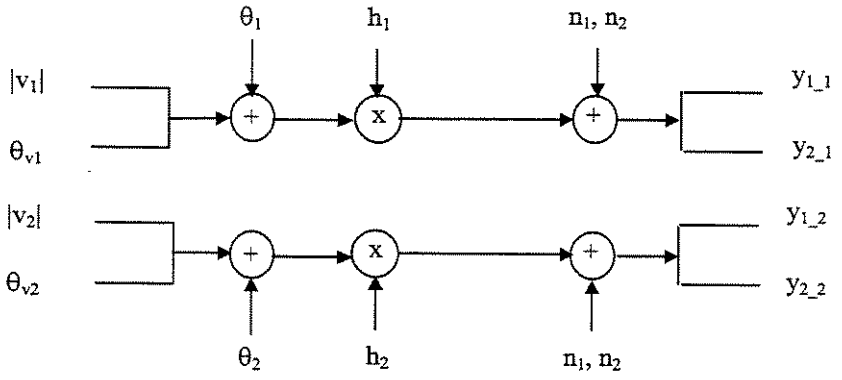

Fig. 4.6 – Canal com desvanecimento Rayleigh com variação lenta de fase

onde: $\begin{cases} h_1 = \sqrt{X_1^{\,2} + X_2^{\,2}} \\ h_2 = \sqrt{X_1^{\,2} + X_2^{\,2}} \end{cases}$

X_1 e X_2 são variáveis Gaussianas independentes de média nula e variância unitária.

O i-ésimo sinal recebido na saída do filtro casado pode ser escrito como:

$$r_i = h_i s_i e^{j\theta} + n_i = y_{1i} + jy_{2i} \tag{4.2}$$

onde h_i é o ruído multiplicativo *Rayleigh*, n_i é uma amostra do ruído gaussiano branco complexo de média nula e variância $N_o/2$, s_i é o sinal transmitido e θ é a fase aleatória introduzida pelo canal que foi assumida como sendo uniformemente distribuída no intervalo $(0, 2\pi)$. As amostras de ruído h_i foram consideradas independentes entre si e de n_i. Considerou-se, também, o conhecimento delas (h_i) no receptor.

Um fato a ser observado é que a modulação 4-DPSK, em sua teoria, necessita de que a fase aleatória permaneça constante durante o intervalo de, pelo menos, dois símbolos. Porém, devido à própria estrutura de treliça usada no demodulador em questão, que trabalha com o conceito de "janela deslizante" (seção 4.2.2.2.1), foi imprescindível a utilização de uma fase aleatória "constante" em todo o bloco de símbolos codificados, uma vez que a demodulação foi feita a cada dois símbolos (janela de dois) com sobreposição de um, como será visto mais adiante. Dessa forma, o canal, cuja fase varia lentamente, foi implementado, na prática, com uma fase aleatoria θ constante em cada bloco de 1006 símbolos provenientes do modulador.

O canal funciona da seguinte maneira (de acordo com a Fig. 4.6): a cada componente do vetor fase, ϕ_v, é somada a fase aleatória arbitrária introduzida pelo canal:

$$\vartheta[i] = \theta + \phi_v[i] \quad (\text{mod } 2\pi) \tag{4.3}$$

Em seguida, é calculado um "módulo parcial", resultante da multiplicação de cada componente do vetor módulo, $|v|$, pelo ruído multiplicativo associado a ele:

$$v[i] = h[i] * |v|[i] \tag{4.4}$$

O próximo passo é converter o símbolo, em notação complexa, formado por esse módulo e fase parciais, para a notação retangular,

$$v[i] * e^{j\vartheta[i]} \Rightarrow x_1[i] + jx_2[i] \tag{4.5}$$

para, então, adicionar o ruído gaussiano branco n_1 e n_2, gerado aleatoriamente:

$$\begin{aligned} y_1[i] &= x_1[i] + n_1 \\ y_2[i] &= x_2[i] + n_2 \end{aligned} \tag{4.6}$$

Em nossas simulações, portanto, foi assumido uma fase aleatória (θ) constante em cada bloco de símbolos com o ruído multiplicativo (h) variando, dentro do bloco, símbolo a símbolo. Uma observação a ser feita é que, em geral, se um dos parâmetros variar lentamente, o outro também terá o mesmo comportamento.

4.2.2.2. Demodulador diferencial

O demodulador diferencial é responsável por processar os símbolos recebidos y_{1i} e y_{2i}, i = 1,2 (referente a cada um dos demoduladores), liberando probabilidades *a posteriori* (*APP's*) associadas a cada um dos símbolos 4-DPSK. As *APP's* são também úteis ao fornecimento de estimações dos símbolos codificados ($\hat{x}_1$ e $\hat{x}_2$) relativas ao (super-) canal de entrada e saída discretas, visto pelo decodificador turbo (seção 4.2.2.4.1). Como informação *a priori* do algoritmo BCJR, que opera no demodulador, são utilizadas quatro entradas, uma para cada símbolo 4-DPSK, alimentadas com dados vindos do decodificador turbo, exceto para a primeira utilização do demodulador, quando a cada uma das entradas é associada a probabilidade ¼ (inicialização).

Abaixo, na Fig. 4.7, o esquema do demodulador:

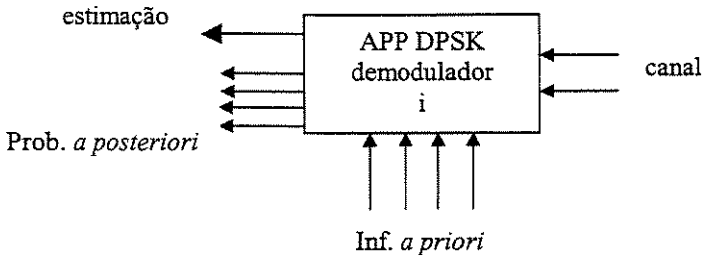

Fig. 4.7 – APP DPSK demodulador

Sabe-se, do capítulo 2, que o algoritmo BCJR destina-se à estimação de probabilidades *a posteriori* (APP) de estados e transições de uma fonte markoviana aplicada a um canal ruidoso. Assim, a utilização desse algoritmo pelo demodulador diferencial (*APP* DPSK demodulador) só seria possível, se o mesmo possuísse tal perfil. Após uma breve análise do modulador DPSK, observou-se que ele poderia ser visto como um codificador de memória $M = 1$ e se comportava como uma máquina de estados finitos de um processo markoviano, o que permitiu, definitivamente, a aplicação do algoritmo, pelo demodulador, bem como o andamento do processo iterativo. A figura seguinte mostra o modulador como uma máquina de quatro estados que coincidem com os símbolos DPSK, e também as transições entre eles.

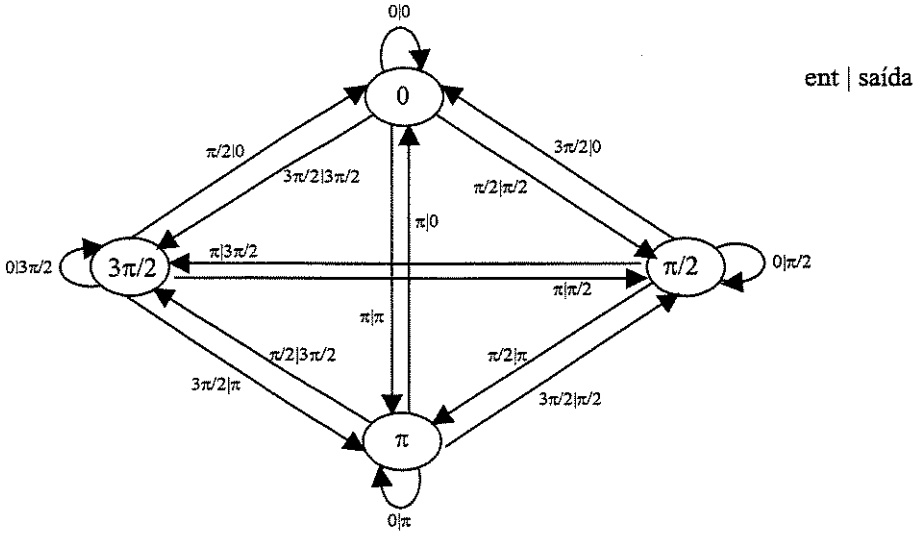

Fig. 4.8 – Representação dos estados e transições do modulador 4-DPSK

Como já foi visto na seção 2.3.2, o algoritmo trabalha, basicamente, com dois tipos de recursões, ao passo que "caminha" pela treliça (obtida através da Fig. 4.8): "para frente" (ou *forward*), α, e "para trás" (ou *backward*), β. A efetuação desses cálculos depende de informações adicionais, como matriz transição de estados, que é obtida da treliça, vetor ou bloco, contendo a informação recebida do canal, probabilidades de transição do canal entre outras. De posse destes resultados, o algoritmo entra em funcionamento. Inicialmente, à medida que o vetor proveniente do canal é recebido, ou seja, para cada estágio t da treliça, o algoritmo calcula a matriz gama e a variável α. Os valores de α são armazenados para todo o t e m (estado da treliça). Em seguida, após o recebimento de todo o bloco ruidoso, as variáveis β são recursivamente calculadas, para que possam ser multiplicadas pelos valores apropriados de gama e α, resultando no σ que, devidamente combinado, dá origem às probabilidades *a posteriori*. Todas as equações associadas a estas variáveis encontram-se na seção 2.3.2.

4.2.2.2.1. Algoritmo BCJR modificado

Em virtude do fato do algoritmo original apresentado não levar em consideração a fase aleatória do canal, θ, foi necessária uma pequena modificação no mesmo, de forma que ele pudesse ser utilizado na recepção de canais não-coerentes. Neste novo cenário, a equação sugerida para o cálculo das probabilidades de transição de canal (utilizadas na obtenção das matrizes gama) não se aplicava ao caso atual e teve que ser substituída por outra encontrada na literatura que trata, justamente, da detecção de símbolos DPSK (eq.9, [8] e/ou eq. 2, [7]):

$$p(r \mid s) = \frac{1}{(2\pi\sigma_0^2)^L} \exp\left[-\frac{1}{2\sigma_0^2}\sum_{l=1}^{L}\left(|r_l|^2 + |s_l|^2\right)\right] \times I_0\left(\frac{1}{\sigma_0^2}\left|\sum_{l=1}^{L} r_l s_l^*\right|\right) \approx I_0\left(\frac{1}{\sigma_0^2}\left|\sum_{l=1}^{L} r_l s_l^*\right|\right) \tag{4.7}$$

onde r e s são, respectivamente, os vetores recebido e transmitido, $\sigma_0^2 = N_0/2$ é a variância do ruído aditivo gaussiano branco (AWGN) e I_0 é a função de *Bessel* modificada de ordem 0 do primeiro tipo.

O termo multiplicativo, na equação anterior, pode ser desconsiderado, já que para o M-PSK, $|s_l|^2$ é constante para todas as fases, resultando, por conseguinte, em informação irrelevante à decodificação.

Como a dedução da fórmula considerava o canal gaussiano, uma modificação foi ainda implementada, de maneira que ela representasse um canal com desvanecimento *Rayleigh*, considerado em nossas simulações:

$$p(r \mid h,s) = \frac{1}{(2\pi\sigma_0^2)^L} \exp\left[-\frac{1}{2\sigma_0^2} \sum_{l=1}^{L} \left(|r_l|^2 + |h_l s_l|^2 \right) \right] \times I_0\left(\frac{1}{\sigma_0^2} \left| \sum_{l=1}^{L} r_l h_l s_l^* \right| \right) \approx I_0\left(\frac{1}{\sigma_0^2} \left| \sum_{l=1}^{L} r_l h_l s_l^* \right| \right) \tag{4.8}$$

onde h é o fator multiplicativo do desvanecimento *Rayleigh*.

Em nossos cálculos, foi utilizado L = 2, que representa a necessidade, na decodificação diferencial, de espera de dois símbolos, tal qual foi feito na codificação. Portanto, as probabilidades de transição de canal passaram a ser calculadas a cada dois símbolos e não mais a cada símbolo apenas, como era feito no desenvolvimento do algoritmo original. Este fato resultou na necessidade de uma pequena modificação no algoritmo (eq. 2.82) justamente nos cálculos das matrizes gama:

$$\begin{aligned} \gamma_t(m',m) &= \sum_{X_1,X_2} P\{X_t = X_2, X_{t-1} = X_1 \mid S_{t-1} = m', S_t = m\} \cdot P\{S_t = m \mid S_{t-1} = m'\} \\ &\quad P\{Y_t, Y_{t-1} \mid X_t = X_2, X_{t-1} = X_1\} \\ &= q_t(X_1, X_2 \mid m', m)\; p_t(m', m)\; R(Y_1, Y_2 \mid X_1, X_2) \end{aligned} \tag{4.9}$$

onde R(·|·) é dado, agora, pela eq. 4.8 com $y_i = r_i$ e $x_i = s_i$, $i = 1, 2$. Isto se reflete no processamento da treliça. Por isso, passou-se a utilizar o conceito de "janela deslizante" de comprimento dois, ou seja, os cálculos não seriam mais efetuados a cada estágio de tempo da treliça e, sim, a cada dois estágios. No próximo cálculo, a janela seria deslocada de um estágio, fazendo com que sempre houvesse sobreposição de um, mantendo, assim, a conformidade com o codificador diferencial do modulador, que exigia que os dois símbolos codificados fossem adjacentes, tornando-os estreitamente ligados entre si, e propiciando uma demodulação correta. A Fig. 4.9 é uma tentativa de visualização deste conceito, onde a variável y representa os símbolos provenientes do canal.

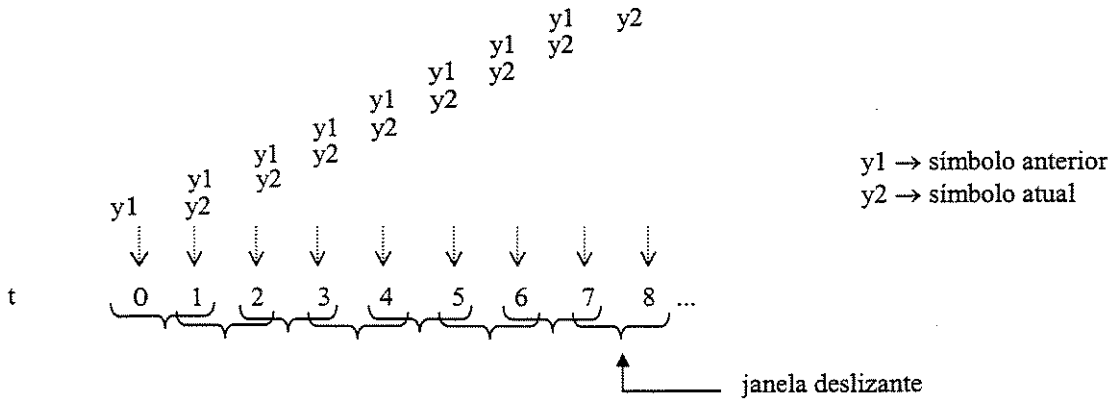

Fig. 4.9 – Representação simbolica da utilização da janela deslizante na treliça

Dessa forma, a cada cálculo de α (ou β), como na seção 2.3.2, o algoritmo continuará se deslocando por um estágio da treliça, apenas terá que observar dois símbolos, o atual e o anterior. No próximo deslocamento, o que era "anterior" passa a ser "atual" e os cálculos em t ainda continuarão dependendo daqueles efetuados em t - 1. O processo continua, até que toda a treliça tenha sido "visitada" pelo algoritmo, tanto em um sentido como no outro, tal qual era feito no processo original.

4.2.2.3. Decodificador turbo

A decodificação turbo é realizada segundo o modelo descrito no capítulo 3. No entanto, devido à mudança no cenário de decodificação – que, agora, compreende a ação conjunta de decodificador e demodulador num processo iterativo – novas variáveis são introduzidas de forma a se adequar ao contexto do receptor em questão. Assim, para uma melhor visualização e entendimento, a Fig. 3.3 é redesenhada na Fig. 4.10.

Os decodificadores componentes, como já foi mencionado, também fazem uso do algoritmo BCJR, mas na sua forma original. Cada um deles utiliza uma treliça binária referente ao codificador convolucional associado, e isto é possível, porque cada codificador também se comporta como uma máquina de estados de um processo de Markov. As treliças são idênticas pelo fato dos codificadores também serem idênticos. A Fig. 4.11 é a representação dos dezesseis estados e transições de um dos codificadores convolucionais do turbo.

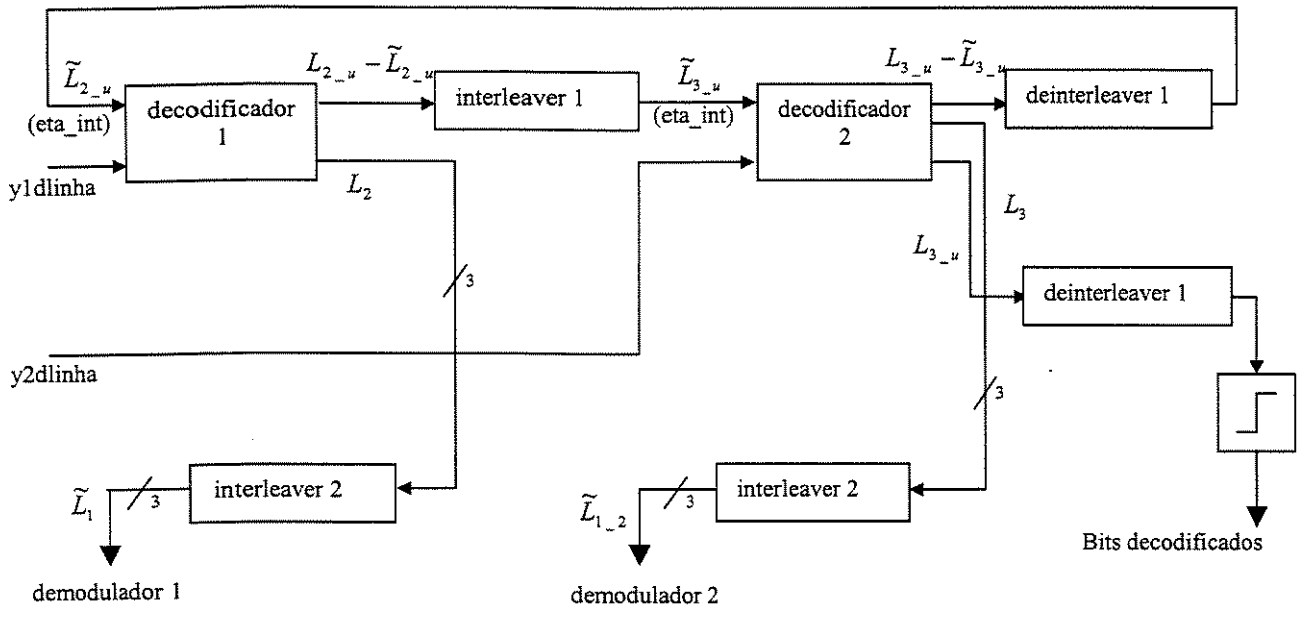

Fig. 4.10 – Decodificador turbo

Sabe-se que o objetivo do decodificador turbo é decidir sobre os bits de informação *u*, o que pressupõe que ele forneça informações a respeito, unicamente, deles. Ocorre que, devido ao processo iterativo, a rede precisa ser realimentada e, por isso, o decodificador envia informação para o demodulador (variáveis $\tilde{L}_1$ e $\tilde{L}_{1\_2}$ na Fig. 4.10) que a recebe por meio da informação *a priori*, cujo conteúdo é atualizado, agindo, assim, de forma a refinar os resultados previamente obtidos (na iteração anterior). Pelo fato, exatamente, de existir esta interação entre os dois estágios, o decodificador precisa produzir como saída, além das *APP's* relativas aos bits de informação, *APP's* associadas aos quatro símbolos DPSK, pois o demodulador apenas lida com símbolos desta natureza.

A solução encontrada para a obtenção de *APP's* de símbolos, a partir de uma decodificação binária, foi fazer uma interpretação diferente sobre uma das informações adicionais necessárias ao funcionamento do algoritmo, à medida que ele "caminha" pela treliça binária do codificador. Reportando-se à seção 2.3.2, ao invés de utilizar os vetores transição de estados relativos aos bits 0 e 1, foi feita uma ligeira modificação no modo de ler estes vetores, para que o algoritmo passasse a trabalhar com vetores transição relativos aos símbolos de saída dos codificadores convolucionais do turbo – 0, 1, 2 e 3 (π, $\pi/2$, $3\pi/2$ e 0, respectivamente) – intrinsecamente associados àqueles bits, como se observa na Tabela 4.2. Para obter as probabilidades relativas aos bits de informação, utiliza-se o fato de que os codificadores são

sistemáticos e, por isso, determinadas combinações de *APP's* dos símbolos originam as *APP's* do bit 0 e do bit 1. Este artifício será mais detalhado na seção 4.2.2.4.2.

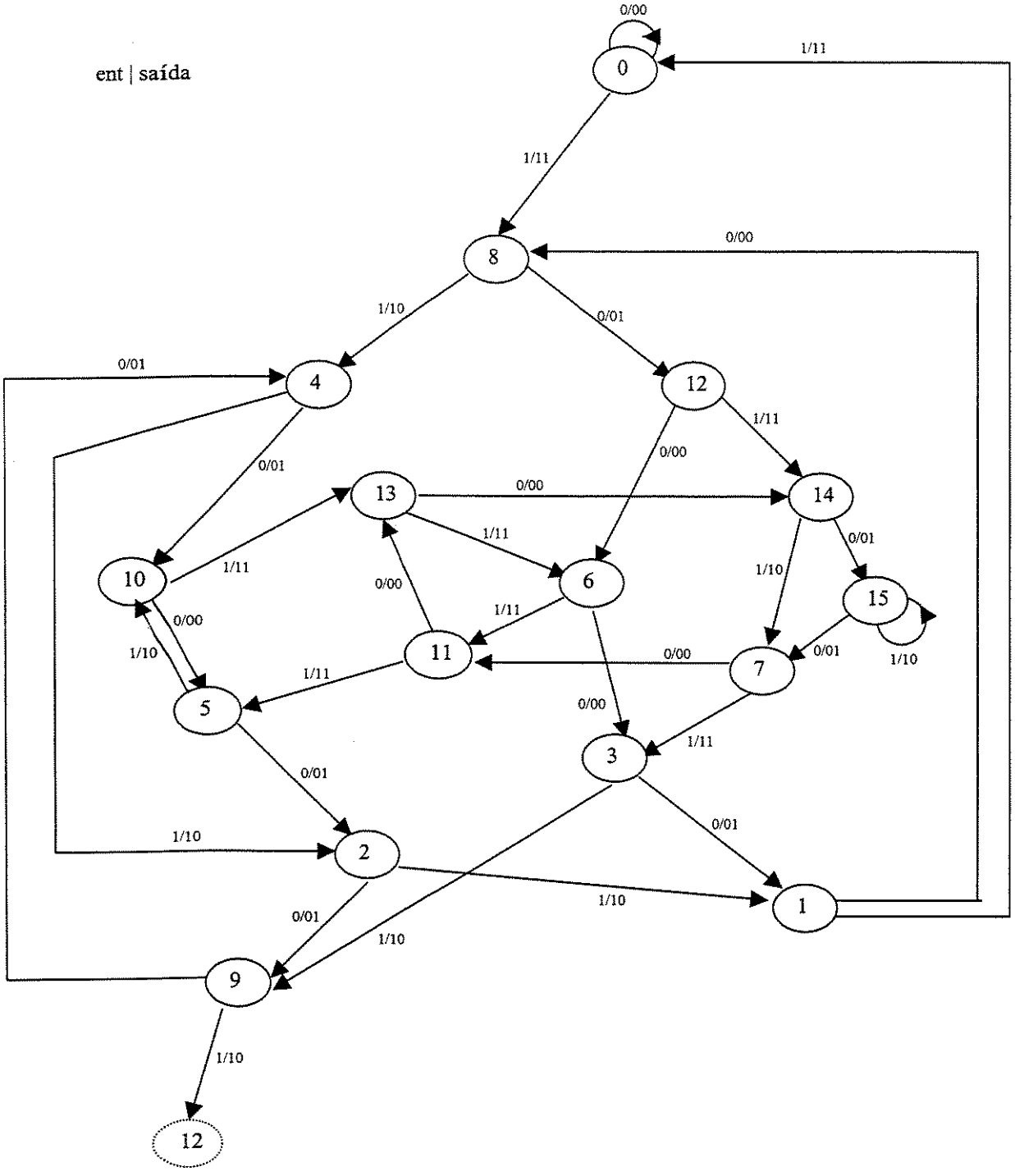

Fig. 4.11 – Representação dos estados e transições de cada codificador convolucional componente do turbo.

4.2.2.4. Descrição da rede de decodificação

Antes de iniciarmos a descrição propriamente dita, é importante frisar que a decodificação turbo utiliza a treliça binária do codificador, ao passo que a demodulação, realizada pelo demodulador APP DPSK, emprega a treliça quaternária do modulador 4-DPSK. Isto significa que, enquanto a decodificação é binária, a demodulação age sobre símbolos. Esta discrepância ocasionou problemas na interface entre os dois estágios, tais como passagem da informação de canal para os decodificadores e passagem da informação *a priori* para o 1° decodificador. Uma proposta para contornar tais dificuldades é exposta nas duas próximas seções.

4.2.2.4.1. O super-canal

As entradas $\hat{x}_1$ e $\hat{x}_2$ dos decodificadores não poderiam ser alimentadas com dados vindos do canal, r, visto que há o bloco demodulador entre eles e não faria sentido algum a utilização desta informação após a demodulação. Assim, uma idéia foi simular um canal visto pelo decodificador turbo, denominado super-canal, mostrado na Fig. 4.12.

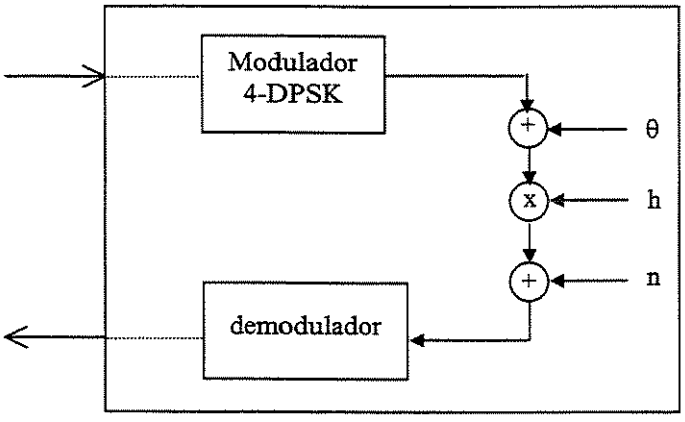

Fig. 4.12 – Canal visto pelo decodificador turbo: super-canal

Antes do sistema entrar em funcionamento, o super-canal foi simulado da seguinte maneira: cada símbolo DPSK foi transmitido pelo super-canal, um certo número de vezes (T), e as probabilidades de transição de símbolo foram estimadas através das suas freqüências relativas.

Em nossas simulações, cada símbolo foi transmitido T = 1000 vezes. A cada transmissão do símbolo, foi observado, na saída do super-canal, o símbolo recebido correspondente. Após findado o número, pré-estabelecido, de transmissões daquele símbolo, foi computada a freqüência relativa de cada símbolo recebido dado o símbolo transmitido. O processo se repetiu para os 4 (quatro) símbolos DPSK (0, $\pi/2$, π, $3\pi/2$).

Ainda, a cada 1000 transmissões de cada símbolo DPSK (ou seja, a cada 4000 transmissões, considerando os quatro símbolos possíveis), a relação sinal-ruído (*SNR*) foi sendo variada e tabelas de probabilidades de transição foram sendo obtidas a partir das freqüências relativas. Em outras palavras, a cada 4000 transmissões (1000 transmissões de cada um dos símbolos 4-DPSK), a *SNR* foi incrementada de forma que cada tabela de probabilidades de transição (sub-tabela representativa daquele valor de *SNR*) pudesse ser vista como uma matriz quadrada onde as linhas representam os símbolos transmitidos e as colunas, os símbolos recebidos. Assim, no final das simulações do super-canal, a tabela armazenada era composta, na verdade, de X sub-tabelas dado que foram considerados X incrementos da *SNR*. Desta forma, quando a rede foi posteriormente simulada, à medida que o algoritmo BCJR, nos decodificadores, requeria informação de canal, a sub-tabela correspondente à *SNR* utilizada era acionada e a probabilidade de transição de símbolos x_i e $\hat{x}_i$ era obtida. Portanto, a entrada do algoritmo BCJR, presente nos decodificadores, relativa a informações de canal, foi substituída por estimações dos símbolos codificados.

4.2.2.4.2. Informação *a priori*

Outro problema encontrado foi como alimentar a entrada relativa à informação *a priori*, no 1° decodificador (chave na Fig. 4.14b), utilizando resultados provenientes do demodulador, uma vez que o primeiro necessita de informações binárias e o segundo apenas fornece informação relativa a símbolos 4-DPSK. A idéia, desta vez, foi fazer uso do fato dos codificadores serem sistemáticos, ou seja, a entrada u equivale à própria saída x_{1i}, e a entrada *ulinha* (u entrelaçada), à saída x_{2i}.

Assim, com o auxílio da Tabela 4.2, a relação direta entre os bits de informação (u) e os símbolos codificados (x_{wi}, x_{wp}) é facilmente percebida:

- Um bit de informação 0 (-1) corresponde ao símbolo codificado π ou $\pi/2$;
- Um bit de informação 1 corresponde ao símbolo codificado 0 ou $3\pi/2$.

Recordando que o decodificador turbo tem como objetivo a decodificação dos bits de informação u, a convenção adotada é que a decisão final será tomada em relação ao bit 0. Portanto, a entrada relativa à informação *a priori* , em ambos os decodificadores do turbo, são assumidas como sendo:

$$eta\_\text{int} = P[u = 0 \mid \tilde{u}] = 1 - P[u = 1 \mid \tilde{u}] \tag{4.10}$$

onde $\tilde{u}$ é a estimação do bit de informação.

O processo iterativo é utilizado, de tal maneira que a informação extraída no receptor possa ser refinada a cada iteração, isto é, a cada passagem pelos blocos demodulador/decodificador, por uma quantidade de vezes previamente estabelecida. Um fato interessante ocorre na primeira iteração externa, quando não há um conhecimento prévio para ser utilizado como informação *a priori* para o demodulador, e em todas as iterações externas, no momento da primeira passagem pelo decodificador turbo, quando são os decodificadores que sofrem esta ausência de conhecimento (esta observação é feita considerando-se a natureza das informações necessárias a cada um dos blocos que, como foi comentado no início da seção, é diferente para o demodulador e o decodificador). Para o primeiro, a solução foi simplesmente estabelecer que, na primeira passagem, todos os símbolos eram equiprováveis (como foi mencionado na seção 4.2.2.2). O segundo exigiu um pouco mais de atenção, porque, após a utilização do demodulador, alguma informação já havia sido extraída na saída desse bloco e deveria ser utilizada para alimentar o 1° decodificador. Porém, essa informação era referente aos símbolos modulados e não aos bits codificados, o que requereu uma espécie de conversão símbolo/bit. A Fig. 4.13 é um "recorte" da Fig. 4.14b justamente na interface entre os dois estágios de decodificação.

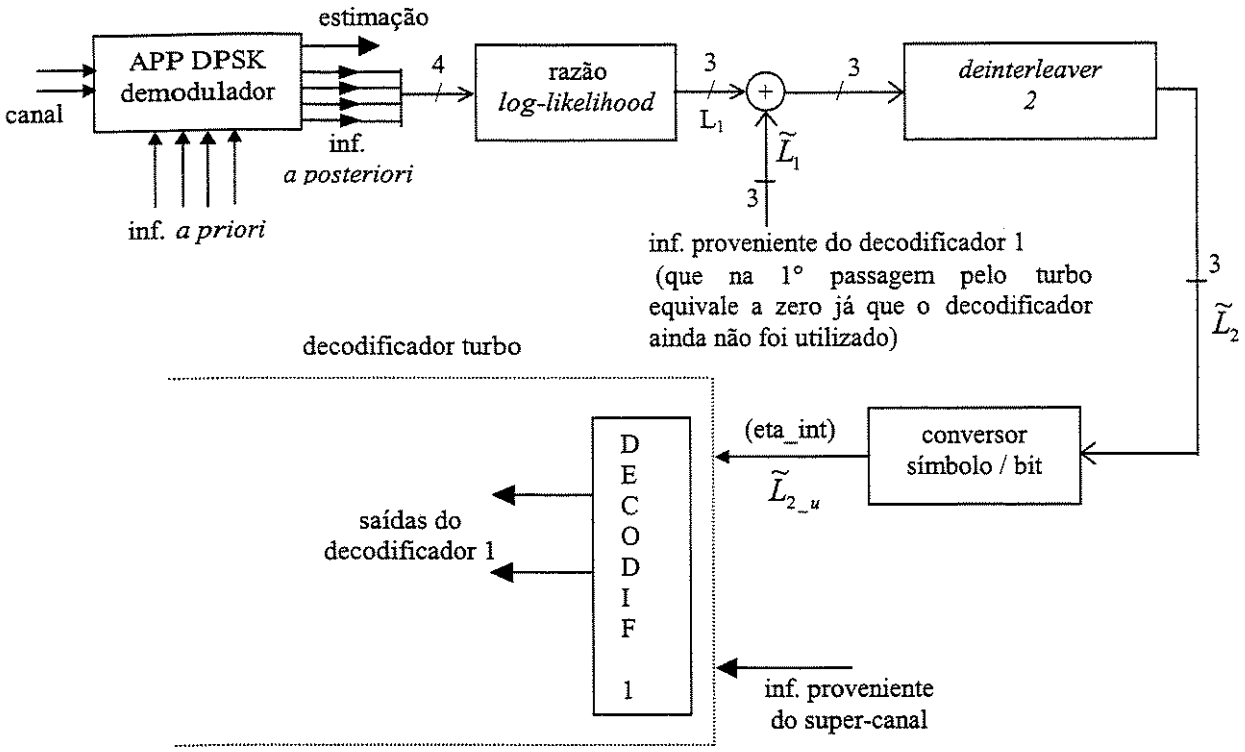

Fig. 4.13 – Interface entre os dois estágios de recepção: "conversão" símbolo/bit

Observando a figura e considerando que ainda não houve iteração demodulador/decodificador turbo: após a primeira passagem pelo demodulador, utilizando a inicialização anteriormente mencionada de probabilidades equiprováveis para as informações *a priori* dos símbolos, as APP's extraídas, através do algoritmo BCJR, no demodulador, são convertidas em razões *log-likelihood*, ditas L_1, adaptadas da eq. 3.1, para se referirem aos símbolos 4-DPSK, como na eq. 4.11. Essas conversões são representadas, na figura, pelo bloco "razão *log-likelihood*".

$$L_{iX} = \log \frac{P[X \mid y]}{P[X_{ref} = 3\pi/2 \mid y]} \tag{4.11}$$

onde X equivale aos símbolos 0, $\pi/2$ ou π, y é a observação do canal, X_{ref} é o símbolo $3\pi/2$ escolhido como referência e $i = 1$, 2 ou 3, dependendo a qual bloco decodificador esta razão se refere. Em seguida, estas razões são desentrelaçadas pelo *deinterleaver 2* (ação necessária, já que estas informações entrarão no decodificador turbo e ele não observa nenhum entrelaçamento

externo nos bits codificados), e dão origem às razões $\tilde{L}_2$ (eq. 4.12) obtidas, utilizando as mesmas definições da equação anterior, em analogia à eq. 3.7,

$$\tilde{L}_{iX} = \log \frac{P[X \mid \tilde{X}]}{P[X_{ref} \mid \tilde{X}]} \tag{4.12}$$

onde $\tilde{X}$ equivale à estimação dos símbolos codificados. Utilizando uma transformação inversa da eq. 4.12, obtém-se:

$$eta_X = P[X \mid \tilde{X}] = e^{\tilde{L}iX} \cdot eta_{X_{ref}} \tag{4.13}$$

onde $eta_{X_{ref}}$ é idêntico a $P[X_{ref} \mid \tilde{X}]$. Empregando, enfim, o artifício mencionado no início desta seção, relativo à particularidade dos códigos sistemáticos, as *APP's* dos símbolos π e $\pi/2$ são devidamente combinadas e tem-se uma fórmula fechada que inicializa a entrada relativa à informação *a priori* do 1° decodificador do turbo durante a primeira passagem, isto é, primeira iteração turbo:

$$eta\_\text{int} = eta_{\pi} + eta_{\pi/2} \tag{4.14}$$

onde *eta_int* refere-se ao bit 0. Essa informação *a priori* é associada, apenas, ao 1° decodificador, que é por onde se tem acesso ao decodificador turbo, como pode ser visto na Fig. 4.14b; a informação *a priori* para o 2° decodificador vem da saída do 1°, durante o processamento turbo.

4.2.2.4.3. Realimentação externa

Sabe-se, do capítulo 3, que um princípio fundamental da decodificação iterativa é não realimentar um decodificador com informação que tenha sido originada dele próprio. Isto se aplica bem à decodificação turbo, porque, mesmo que os dois decodificadores presentes estejam associados a codificadores distintos, eles fornecem informação sobre a mesma variável: os bits de informação u, que são os mesmos na entrada de ambos os codificadores, a menos de um entrelaçamento para o segundo. Entretanto, no caso da realimentação externa demodulador/decodificador, o fato de se trabalhar, simultaneamente, com símbolos e bits gera algumas dificuldades. O problema é que, dentro do decodificador turbo, não há como "passar" informação extrínseca relativa a símbolos, de um decodificador para o outro, uma vez que os

símbolos associados a um decodificador em nada se assemelham àqueles associados ao outro, pois foram gerados independentemente nos dois codificadores convolucionais. Portanto, conclui-se que não há como extrair informação extrínseca de símbolo do segundo decodificador, o que compromete a passagem deste tipo de informação na realimentação externa (saída L_3 na Fig. 4.14b).

Esses aspectos foram testados e, como o previsto, não resultaram em ganho para o sistema, tendo sido, então, descartados. A única exceção ocorre na realimentação do demodulador 1 para o 1° decodificador, pois, neste caso, as saídas relativas aos símbolos, deste demodulador, que são combinadas para originar a informação *a priori* para o dito decodificador, refletem-se mais diretamente na saída desse último e, dessa forma, é gerada uma informação extrínseca que será convertida em informação *a priori,* para ser utilizada no próximo acionamento do decodificador turbo, ou seja, na próxima iteração interna.

4.2.2.4.4. Descrição da rede de decodificação

Visando facilitar as operações realizadas, os decodificadores 1 e 2 e, também, o demodulador fornecem, por fim, como saída, razões *log-likelihood* (*RLL*). Para os bits de informação *u,* as probabilidades *a posteriori* são convertidas como na eq. 3.1, onde a probabilidade do bit 0 é tida como referência. Para os símbolos codificados, nas saídas dos decodificadores e na saída do demodulador, a fórmula para as *RLL's* foi adaptada e resultou na eq. 4.11.

Durante as iterações internas do decodificador turbo, as operações usuais, utilizando variáveis binárias, são utilizadas, ou seja, enquanto o turbo estiver em funcionamento, os decodificadores fornecem, em sua saída, apenas *RLL's* referentes aos bits de informação *u,* pois não há necessidade de se trabalhar com informações sobre símbolos, uma vez que, internamente, a decodificação é binária. Tais informações somente serão fornecidas, quando forem finalizadas as iterações internas e o turbo precisar dar seqüência ao processo iterativo, realimentando o demodulador através das iterações externas.

É importante ressaltar que a realimentação externa obedece à seqüência de origem dos bits e/ou símbolos: o 1° (2°) decodificador está associado ao 1° (2°) codificador

convolucional e, portanto, além de ser responsável por decodificar os bits codificados por este último, age no intuito de refinar as informações referentes aos símbolos diretamente associados a esses bits, ou seja, também se refere ao 1° (2°) demodulador.

Em síntese, a rede de decodificação funciona da seguinte maneira: a informação ruidosa do canal alimenta uma das entradas dos blocos demoduladores. A outra, relativa à informação *a priori*, é alimentada tal qual o sugerido na seção 4.2.2.2, uma vez que ainda não há informação proveniente dos decodificadores. Os blocos demoduladores liberam as *APP's* relativas aos símbolos que são convertidas em *RLL's*. Na interface entre os dois estágios, estas razões são desentrelaçadas, pelo *deinterleaver* externo, originando $\tilde{L}_2$, como na eq. 4.12. Estas novas *RLL's* sofrem uma transformação inversa e dão origem a uma espécie de informação *a priori* de símbolo, eq. 4.13, que são, então, combinadas, como na eq. 4.14, para formar a informação *a priori* de bit para o 1° decodificador. A outra entrada, referente à informação de canal, é alimentada com uma das entradas da tabela do super-canal: as estimações dos símbolos 4-DPSK. O início das iterações internas ocorre neste momento, quando o turbo é acionado. O 1° decodificador, por sua vez, libera a *RLL* de bit, L_{2_u} (eq. 3.1), que é subtraída da *RLL* de bit inicial, $\tilde{L}_{2\_u}$ (eq. 4.15), obtida de *eta_int,* através de uma transformação inversa em uma equação similar à eq. 3.1, apenas utilizando a estimação dos bits de informação, $\tilde{u}$, ao invés da observação do canal *y*,

$$\tilde{L}_{i\_u} = \log\frac{P[u=1\,|\,\tilde{u}]}{P[u=0\,|\,\tilde{u}]} = \log\frac{1-eta\_\text{int}}{eta\_\text{int}} \tag{4.15}$$

formando, portanto, a informação extrínseca que, desentrelaçada pelo *deinterleaver* interno, dá origem a $\tilde{L}_{3\_u}$. Através de uma transformação inversa na eq. 4.15, esta razão dá origem à informação *a priori*, *eta_int*, para o 2° decodificador, tal qual o visto na eq. 4.16.

$$eta\_\text{int} = 1 - \frac{e^{\tilde{L}_{i\_u}}}{1+e^{\tilde{L}_{i\_u}}} \tag{4.16}$$

Após o processamento, o dito decodificador libera a *RLL* L_{3_u}, que, subtraída de $\tilde{L}_{3\_u}$, dá origem a $\tilde{L}_{2\_u}$ formando, novamente, a informação *a priori*, *eta_int*, para o 1° decodificador, de forma a

refinar os valores anteriormente obtidos. Eis, então, a primeira iteração interna. Tendo concluído a quantidade de iterações internas, previamente estabelecida, o decodificador turbo libera as *RLL's* L_2 e L_3, relativas aos símbolos, para os blocos demoduladores 1 e 2, respectivamente, de forma a dar continuidade ao processo iterativo. Estas razões são, dessa forma, entrelaçadas pelo *interleaver* externo e transformadas em informação *a priori*, eta_X, como na eq. 4.13, para os demoduladores. Completou-se, neste momento, uma iteração externa. Este processo é repetido e as informações vão se refinando, à medida que passam de um estágio para o outro, até que seja findado o número previamente estabelecido de iterações externas. Neste instante, a demodulação/decodificação é finalizada e o 2° decodificador do turbo fornece as *APP's* dos bits de informação que são desentrelaçadas pelo *deinterleaver* interno e tem-se a decisão final em relação a um limiar pré-estabelecido. A probabilidade de erro de bit, P_b, é, finalmente, obtida, comparando-se o vetor de entrada u e o vetor decodificado v.

Uma observação a ser feita é que, na interface entre os dois estágios (Fig. 4.13), as *RLL's* L_2 entrelaçadas, $\widetilde{L}_1$, não contribuem para formar a informação extrínseca, $L_1 - \widetilde{L}_1$, uma vez que ainda não há informação vinda do turbo, já que, neste momento, ele ainda não foi utilizado. Esta informação só será relevante, a partir da segunda iteração externa.

A análise do desempenho da rede é fornecida na próxima seção.

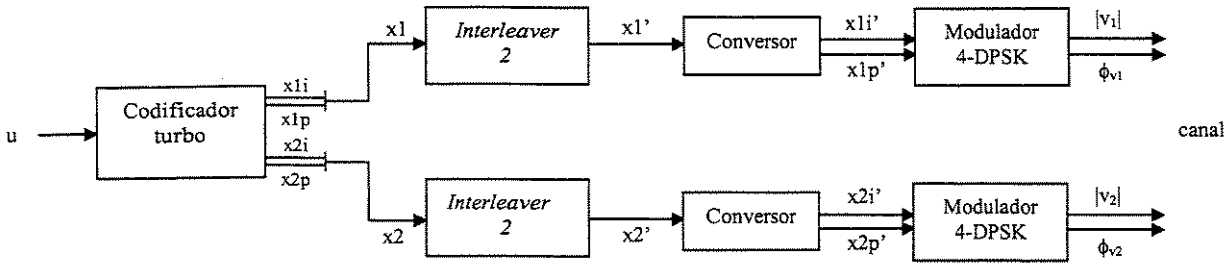

Fig. 4.14a - Arranjo do transmissor

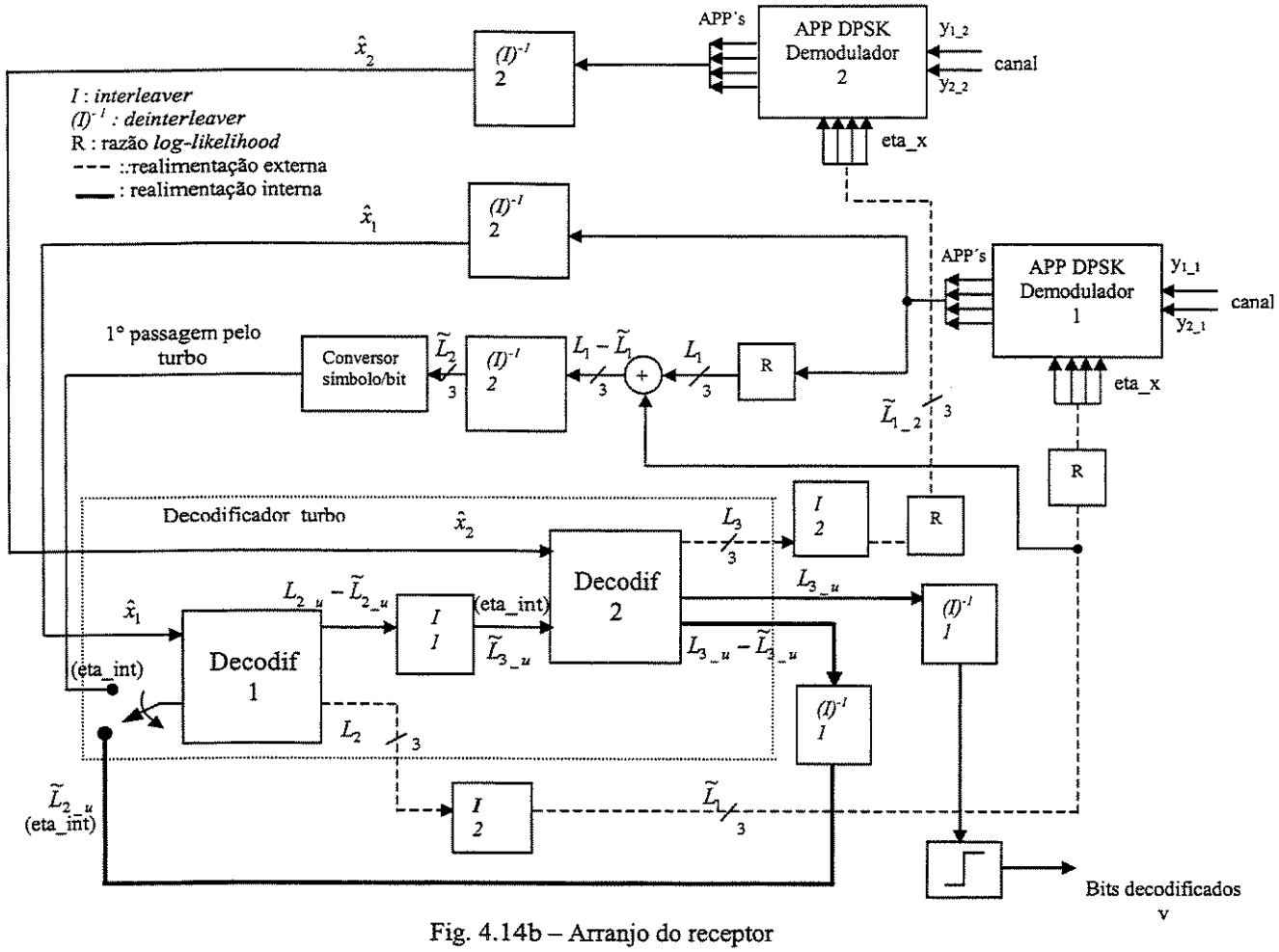

Fig. 4.14b – Arranjo do receptor

4.3. Resultados de simulação

A rede de codificação/decodificação proposta foi simulada, exaustivamente, visando à busca das melhores combinações entre as iterações externa e interna. Nas simulações, foi utilizada, para a programação, a linguagem *C*. Em nossos cálculos, foram consideradas palavras de comprimento de 1000 bits e cada ponto, de cada curva obtida, foi gerado pela transmissão e recepção de 3000 palavras (blocos de informação), totalizando, então, $3 \cdot 10^6$ pontos para cada relação sinal-ruído. Esta dimensão (10^6), no número de pontos gerados, foi escolhida de tal forma que tivéssemos uma confiabilidade nas estimativas da probabilidade de erro de bit (P_b) até 10^{-5}. À título de observação, vale ressaltar que o ruído multiplicativo (h) foi admitido conhecido pelo receptor. As figuras seguintes mostram resultados de simulação para a probabilidade de erro de bit (P_b) versus E_b/N_0. Nas figuras, *sExI* representa a simulação com E iterações externas e I iterações internas, onde uma iteração significa uma passagem pelo demodulador ou decodificador turbo.

Na Fig. 4.15, o desempenho da modulação 4-DPSK (sistema não-codificado), com demodulação MAP – algoritmo BCJR – foi também incluído e demonstra ser ligeiramente superior ao desempenho com demodulação diferencial convencional (não mostrado na figura). A comparação entre essas curvas pode ser visualizada na Fig. 2.15. Para $E = 1$, quatro iterações internas ($I = 4$) são suficientes para atingir o melhor desempenho. Mantendo a mesma complexidade de decodificação, foi testada a combinação $E = I = 2$ e observou-se uma degradação de desempenho de aproximadamente 0,6 dB para $P_b \approx 10^{-4}$. Várias outras curvas foram obtidas combinando diversos valores de E e I mas, dentre elas, a combinação $E = 2$, $I = 4$ mostrou-se ser a mais eficiente. Com estes valores foi alcançado um pequeno ganho adicional, relativo a $E = 1$, $I = 4$.

A Fig. 4.16 mostra um exemplo do efeito da realimentação externa, no desempenho da rede, para $I = 2$. Comparando-se os dois gráficos, nota-se que um aumento de E não contribui de maneira significativa para o desempenho do sistema, sendo, portanto, a realimentação interna o ponto de maior importância. Como exemplo, podemos citar as curvas *s*1x1, *s*1x2 e *s*1x3, da Fig. 4.15, e *s*1x2, *s*2x2, *s*3x2, da Fig. 4.16. Nitidamente, observa-se que o ganho de desempenho obtido, na seqüência das curvas, na primeira figura, é bem mais substancial que aquele alcançado

no segundo grupo. Este fato pode ter relação com a dificuldade de passagem da informação extrínseca para o demodulador, como foi descrito na seção 4.2.2.4.3.

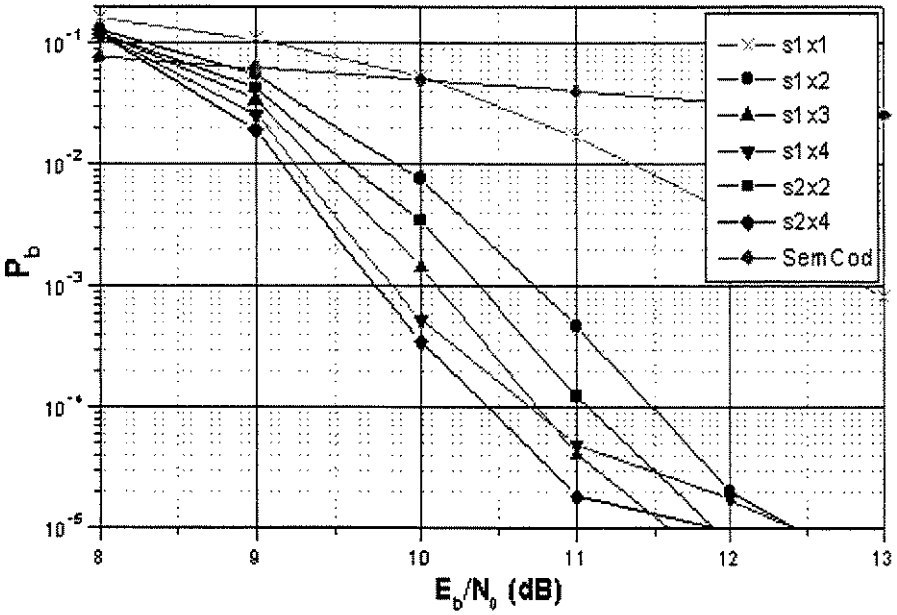

Fig. 4.15 – Curvas de desempenho para várias combinações de iterações externas e internas

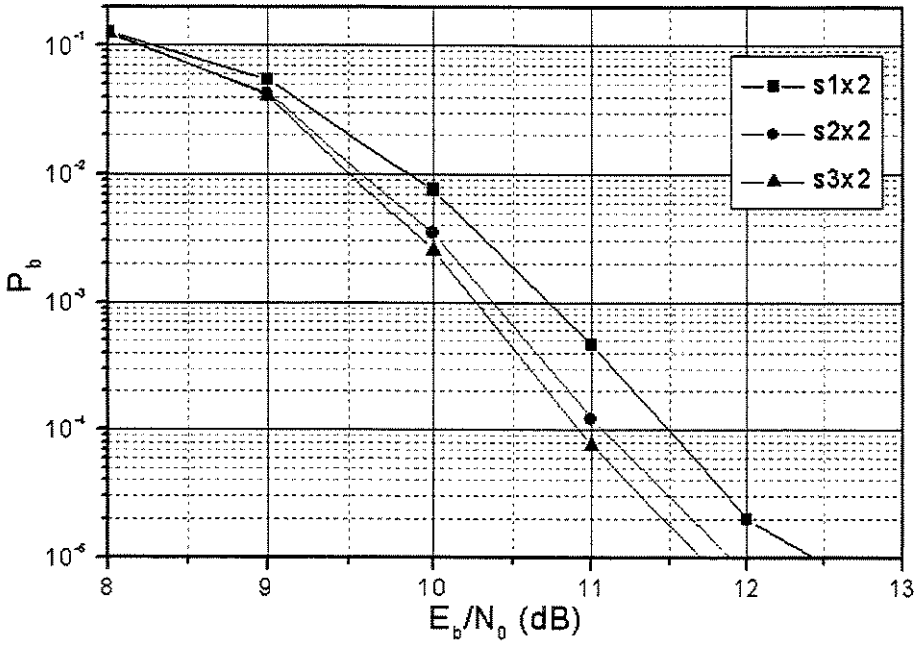

Fig. 4.16 – Curvas de desempenho enfatizando a realimentação externa

Capítulo 5

Conclusão

Tendo em vista o grande interesse recente em projetos de sistemas concatenados de modulação diferencial com decodificação iterativa, este trabalho teve como objetivo aliar o desempenho, já comprovado, da codificação/decodificação turbo ao cenário das redes de decodificação. Com este fim, foi apresentada uma solução heurística que permitiu não apenas inserir a codificação turbo no contexto da rede proposta, mas também solucionar problemas práticos originários desta ação, tais como aqueles inerentes à interface entre os dois estágios de decodificação (como foi apresentado no Capítulo 4). É válido mencionar que, sendo o esquema da rede de codificação/decodificação proposta resultado de um exaustivo processo de tentativas e erros, ele foi a solução escolhida, por se mostrar mais adequado, uma vez que propiciou melhores resultados.

Considerando um mesmo modelo de canal – não-coerente e com desvanecimento *Rayleigh* – e complexidade de decodificação similar, não foi possível comparar os valores de E_b/N_0 mínimos, atingidos pela rede de decodificação proposta, com resultados da literatura, para um sistema com mesma eficiência espectral. Tal comparação poderia ser realizada em trabalhos futuros.

Resultados de simulação mostraram que, considerando a arquitetura da rede proposta, a decodificação turbo, representada pela realimentação interna, contribui, de maneira mais significativa, para o desempenho final. A realimentação externa, por sua vez, demonstrou colaborar, de maneira aquém daquela esperada, provavelmente devido à dificuldade encontrada na passagem da informação extrínseca de símbolo. É muito provável que modificações na passagem deste tipo de informação, do decodificador para o demodulador, levem a uma substancial melhora, no desempenho da rede, para um número total fixo de iterações. Estudos vêm sendo desenvolvidos neste sentido.

Referências

[1] C. Berrou e A. Glavoux, "Near Optimum Error Correcting Coding and Decoding: Turbo Codes", IEEE Transactions on Communications, vol. 44, No. 10, pp. 1261-1271, Outubro 1996.

[2] L. R. Bahl, J. Cocke, F. Jelinek e J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", IEEE Transactions on Information Theory, pp. 284-287, Março 1974.

[3] D. Divsalar e F. Pollara, "Turbo Codes for Deep-Space Communications", TDA Progress Report 42-120, JPL, Fevereiro 1995.

[4] B. Sklar, "A Primer on Turbo Codes Concepts", IEEE Communications Magazine, pp. 94-102, Dezembro 1997.

[5] G. Covalope, G. Ferrari e R. Raheli, "Noncoherent Iterative (Turbo) Decoding", IEEE Transactions on Communications , vol. 48, No. 9, pp. 1488-1498, Setembro 2000.

[6] P. Hoeher e J. Lodge, ""Turbo DPSK": Iterative Differential PSK Demodulation and Channel Decoding", IEEE Transactions on Communications, vol. 47, No. 6, pp. 837-843, Junho 1999.

[7] M. Peleg e S. Shamai, "Iterative decoding of coded and interleaved noncoherent multiple symbol detected DPSK", Electronics Letters, vol. 33, No. 12, pp. 1018-1020, Junho 1997.

[8] D. Divsalar e M. Simon, "Multiple-Symbol Differential Detection of MPSK", IEEE Transactions on Communications, vol. 38, No. 3, pp. 300-308, Março 1990.

[9] S. Benedetto, G. Montorsi, "Unveiling Turbo Codes: Some Results on Parallel Concatenated Coding Schemes", IEEE Transactions on Information Theory, vol. 42, No. 2, pp. 409-428, Março 1996.

[10] I. Marsland e P. Mathiopoulos, "On the Peformance of Iterative Noncoherent Detection of Coded M-PSK Signals", IEEE Transactions on Communications , vol. 48, No. 4, pp. 588-596, Abril 2000.

[11] S. Benedetto, G. Montorsi, D. Divsalar e F. Pollara, "Soft-Input Soft-Output Modules for the Construction and Distributed Iterative Decoding of Code Networks", European Transactions on Telecommunications, vol. 9, No. 2, Março-Abril 1998.

[12] Cecílio Pimentel, Notas de aula do curso de graduação "Comunicação digital", UFPE.

[13] Shu Lin, D. Costello, "Error Control Coding – Fundamentals and Applications", Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1983.

[14] A. Dholakia, "Introduction to Convolutional Codes with Applications", Kluwer Academic Publishers, 1994.

[15] J. Proakis, "Digital Communications", McGraw-Hill International Editions, 1995.

[16] C. Heegard, S. Wicker, "Turbo Coding", Kluwer Academic Publishers.

[17] D. N. Rowitch, "Convolutional and Turbo Coded Multicarrier Direct Sequence CDMA, and Applications of Turbo Codes to Hybrid ARQ Communication Systems", PhD Dissertation, University of California, San Diego, 1998.

[18] Channel Science, http://www.channelscience.com/turbo_training.htm.

[19] Project Turbo Code, http://personal.ie.cuhk.edu.hk/~chankm6/TurboCode/.

[20] Research and Development: Communications/Turbo Coding, http://www.xenotran.com/turbo_tech.html.

[21] H. Freire, Comunicação pessoal.

[22] J. F. Pedrosa, J. Portugheis, "Uma Rede de Codificação/Decodificação para Canais Não-coerentes", Simpósio Brasileiro de Telecomunicações - SBrT, Setembro 2001.