Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica e de Computação

Projeto e Análise de Receptores Iterativos através de Funções EXIT

Dissertação de Mestrado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica. Área de concentração: **Engenharia de Telecomunicações e Telemática**.

Autor: Rafael de Sousa Marinho
Orientador: Jaime Portugheis

Campinas, SP
2007

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

M338p

Marinho, Rafael de Sousa
Projeto e análise de receptores iterativos através de funções EXIT / Rafael de Sousa Marinho. – Campinas, SP: [s.n.], 2007.

Orientador: Jaime Portugheis.
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.
1. Teoria da Codificação. 2. Códigos de controle de erros (Teoria da Informação). I. Portugheis, Jaime. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Título em Inglês:	Design and analysis of iterative receivers using EXIT charts
Palavras-chave em Inglês:	Coding theory, Error-correcting codes (Information theory)
Área de concentração:	Engenharia de Telecomunicações e Telemática
Titulação:	Mestre em Engenharia Elétrica
Banca Examinadora:	Daniel Carvalho da Cunha, Celso de Almeida, Renato da Rocha Lopes
Data da defesa:	19/12/2007
Programa de Pós-Graduação:	Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidato: Rafael de Sousa Marinho

Data da Defesa: 19 de dezembro de 2007

Título da Tese: "Projeto e Análise de Receptores Iterativos Através de Funções EXIT"

Prof. Dr. Jaime Portugheis (Presidente): _____

Prof. Dr. Daniel Carvalho da Cunha: _____

Prof. Dr. Celso de Almeida: _____

Prof. Dr. Renato da Rocha Lopes: _____

Agradecimentos

Agradeço primeiramente a Deus, que me ajudou e assistiu nessa caminhada e a meus pais que tanto me auxiliaram e que me deram a chance de ser quem sou.

De Belém, agradeço a todos os meus amigos que participaram de momentos importantes durante minha vida. Agradeço a UFPA e a todos os professores que também fizeram parte de minha escalada acadêmica.

De Campinas, agradeço aos amigos novos e antigos, aos que conheci e aos que já conhecia. Agradeço a todos os paraenses erradicados em Campinas. Agradeço a meu orientador Jaime Portugheis e aos membros da banca.

Agradeço também a CAPES e a UNICAMP que me deram suporte e me acolheram para o desenvolvimento desse trabalho.

Resumo

Durante a última década foram desenvolvidas novas ferramentas para o projeto e análise da decodificação iterativa de códigos concatenados. Duas destas ferramentas são: o algoritmo de evolução de densidades de probabilidade (conhecido em inglês como *density evolution*) e as funções EXIT (do inglês, *Extrinsic Information Transfer Functions*). As curvas EXIT foram primeiramente desenvolvidas para códigos concatenados paralelos (PCC, do inglês, *Parallel Concatenated Codes*). Posteriormente, estas curvas foram utilizadas para analisar a decodificação da classe de códigos conhecida como LDPC (do inglês, *Low-Density Parity-Check Codes*). Como se sabe, o desempenho dos códigos LDPC pode se aproximar bastante da capacidade de canal.

Neste trabalho utilizamos as funções EXIT para analisar um sistema de transmissão digital composto pela concatenação serial de um codificador LDPC, um modulador M-APSK e um codificador diferencial. O canal considerado é o AWGN não-coerente de bloco. O processo de decodificação/demodulação é descrito através de um algoritmo iterativo baseado em um grafo-fator. Da análise concluímos que códigos LDPC regulares possuem um bom desempenho, mas ainda estão afastados da capacidade de canal.

Palavras-chave: codificação de canal; códigos LDPC; receptores iterativos; funções EXIT.

Abstract

Late 1990s, new tools for evalution and design of iterative decoding of concatenated codes were developed. Two of these tools are: the density evolution algorithm and the EXIT charts. The EXIT charts were first developed for Parallel Concatenated Codes (PCC) and then used for evaluating the decoding process of Low-Density Parity-Check codes (LDPC Codes). As already known, the LDPC codes perform close to channel capacity limits.

In this work we use the EXIT Charts to evaluate a digital transmission system composed by the concatenation of a LDPC encoder, a M-APSK modulator and a diferencial encoder. The channel considered is an AWGN block noncoherent channel and the decoding/demodulating process is described through an iterative algorithm based on a factor graph. The analisys of this system we conclude that LDPC codes have good performance, but are still far from channel capacity.

Keywords: channel coding; LDPC codes; iterative receivers; EXIT Charts.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A humanidade sempre sentiu a necessidade de se comunicar, repassar notícias, compartilhar conhecimento. Devido a esta necessidade, diversas formas de transmitir informação foram criadas. A invenção do rádio, por exemplo, diminuiu as distâncias geográficas de tal forma que instantaneamente já era possível saber o que acontecia do outro lado do mundo.

Com a comunicação digital, a distância que separa geograficamente a informação das pessoas se tornou ainda menor. A partir disso, diversos dispositivos foram adaptados com a finalidade de oferecer maior confiabilidade e variedade de serviços e equipamentos. A comunicação sem fio, por exemplo, tende em alguns anos a substituir completamente os meios de transmissão com fio, convergindo diversos serviços, como TV digital e vídeo conferência, em um único aparelho. Entretanto, para isso é necessário uma forte confiabilidade na transmissão de dados, além de ferramentas que ajudem na criação dos sistemas de comunicação.

No final da década de quarenta Claude E. Shannon, em um trabalho pioneiro [1], demonstrou conceitos importantes para a comunicação digital. Neste trabalho, ele afirma que existe um limite máximo de transmissão que relaciona a potência transmitida, a largura de faixa e a taxa máxima com que a informação pode ser enviada confiavelmente por um canal. Esse limite ficou conhecido como *capacidade de canal*. Além disso, Shannon afirma que existe uma maneira confiável de transmitir a informação utilizando *códigos de correção de erros*. Nasce então uma nova área de estudo conhecida como *teoria da informação*.

Neste trabalho, mostramos uma maneira de analisar os códigos corretores de erro de uma família específica, conhecida como códigos LDPC (do inglês, *Low-Density Parity-Check codes*). Essa família de códigos foi criada por Gallager em 1963 em sua tese de doutorado [2]

e permaneceu pouco estudada durante mais ou menos trinta anos, quando na década de noventa estudos feitos por MacKay *et al.* [3] e [4], mostraram que essa família de códigos pode chegar perto do limite determinado por Shannon.

O intuito desse trabalho é esclarecer aspectos importantes do projeto de sistemas de transmissão que utilizam códigos LDPC, através de uma nova ferramenta criada por Stephan ten Brink em [5] e [6] para análise desses sistemas.

Na intenção de elucidar a proposta desse trabalho, o dividimos da seguinte forma: No Capítulo 2 mostramos um apanhado de técnicas necessárias para o entendimento de sistemas de comunicação e de codificação/decodificação, bem como alguns conceitos necessários de teoria de informação. No Capítulo 3 , introduzimos os grafos-fatores [7] e o algoritmo soma-produto, ferramentas indispensáveis na decodificação de códigos LDPC. No Capítulo 4, é introduzido um estudo estrutural sobre as curvas EXIT, onde comparamos as características das curvas EXIT de códigos LDPC e de códigos concatenados paralelamente, além de mostrarmos algumas características dessas curvas. Já no Capítulo 5, uma análise utilizando curvas EXIT sobre o sistema proposto em [8] é feita. Ainda nesse capítulo, uma forma generalizada é apresentada para análise de códigos em conjunto à modulação, utilizando códigos EXIT. No Capítulo 6, mostramos os aspectos mais importantes desse trabalho e algumas idéias para trabalhos futuros.

Capítulo 2

Conceitos Básicos

O processo de comunicação consiste, em grosso modo, em transmitir informação de um ponto a outro. A informação a ser transmitida pode ser digital ou analógica e isso depende de sua fonte. A Figura 2.1 mostra um diagrama de blocos e alguns elementos de um sistema de comunicação digital. A fonte de sinal pode ser digital, no caso de uma foto digitalizada; ou analógica, como um sinal de áudio, nesse caso, o sinal de áudio precisa ser digitalizado de alguma forma. Após esse sinal passar por um codificador de fonte ele terá uma nova representação digital, onde se retira informação desnecessária para sua representação. A esse processo dá-se o nome de *codificação de fonte*.

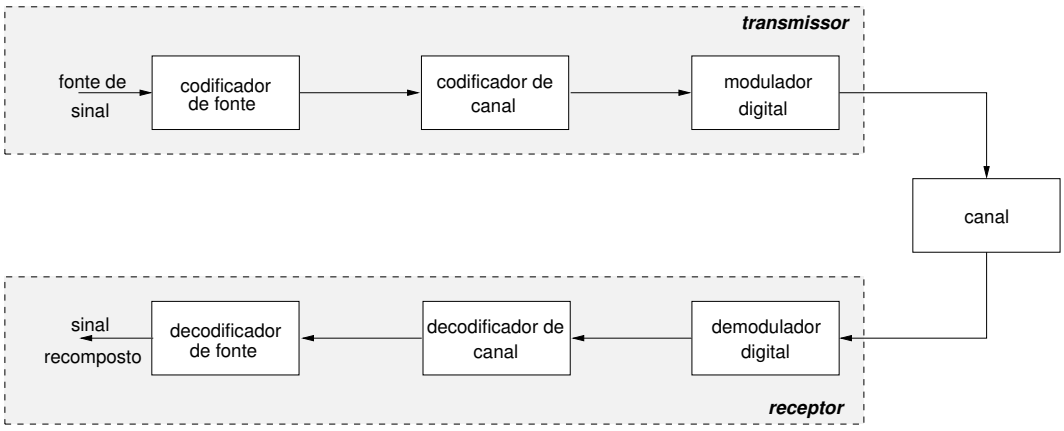


Figura 2.1: Diagrama de blocos de um canal de comunicação.

No próximo passo, o sinal que sai do primeiro bloco é novamente codificado, entretanto, agora são adicionados bits de redundância, que servem para controle de erros durante a transmissão do sinal no canal. Esses bits de redundância são inseridos de maneira controlada

para que possam, depois, serem utilizados para correção de erros.

Na saída do codificador de canal, o sinal precisa de uma nova representação para satisfazer as limitações físicas de canal que pertence ao sistema de comunicação. Portanto, o sinal ainda passa pelo bloco de modulação, que serve para mapear sequências de bits codificadas em formas de ondas que serão então transmitidas pelo canal.

O quadrado cinza inferior da Figura 2.1 mostra o receptor, onde se executam tarefas inversas às do transmissor, até que o sinal seja recomposto. No caso de o sinal na entrada do transmissor não ser um sinal digital, algum tipo de transdutor deve ser inserido na saída do sistema receptor para que o sinal seja iteiramente recomposto.

A diferença entre o processo de codificação de fonte e de canal é que a codificação de fonte retira bits de redundância desnecessários para descrever a informação, por exemplo, a voz é um sinal que têm componentes em várias faixas de freqüência, mas nem todas essas faixas são perceptíveis pelo ouvido humano, portanto retirar essas faixas de freqüência pode reduzir a representação do sinal de voz como um sinal digital. Já no caso da codificação de canal, bits para controle de erro são adicionados, dessa forma é possível que erros sejam encontrados e possívelmente corrigidos pelo receptor na saída do canal.

2.1 Modulação e canais ruidosos

Os sinais enviados ao canal pelo transmissor são formas de onda definidas pelo modulador em intervalos de tempo T. A esses sinais, na representação de espaço de sinais, dá-se o nome de *símbolos*. A cada símbolo é atribuido um mapeamento de bits, e dependendo do número de símbolos que existem na modulação a taxa do sistema de comunicação digital pode modificar-se.

Um mapeamento de bits em símbolos é mostrado na equação 2.1, os bits 0 e 1 são mapeados para 1 e -1,

$$a_i = 2c_i - 1, \tag{2.1}$$

onde c_i é o bit codificado e a_i é o valor de amplitude da modulação, no caso BPSK (do inglês, *Binary Phase Shift-Keying*). A Figura 2.2 mostra a *constelação* referente a essa modulação. Os símbolos da modulação são expressos por $s_i(t) = a_i \cos(2\pi f_c t)$ por exemplo, onde f_c é a

freqüência da onda portadora.

O principal problema que deve ser combatido em sistemas de comunicação é o ruído inserido no sinal pelo canal. De fato, a maioria dos canais de transmissão são ruidosos de alguma forma, por exemplo, canais de comunicação com fio, canais de comunicação sem fio e mesmo canais de comunicação ópticos.

Figura 2.2: Constelação da modulação BPSK.

O ruído mais comum estudado em sistemas digitais de comunicação é o ruído chamado de *ruído aditivo branco gaussiano*. Sua densidade espectral de potência têm componentes em todas as faixas de freqüência, e como seu nome diz, ele interfere de forma aditiva no símbolo a_k enviado, isto é,

$$r_k = a_k + n_k, \tag{2.2}$$

em que r_k é o símbolo na saída do canal e n_k é o ruido gaussiano adicionado ao símbolo a_k.

2.2 Teoria da informação

A teoria da informação investiga as limitações teóricas dos sistemas de comunicações, ela lida com os aspectos mais fundamentais desses sistemas e dá subsídios para construção de codificadores e decodificadores para modelos de fontes e canais reais.

A modelagem matemática de um sistema de comunicação fornece as estruturas necessárias para o projeto de sistemas reais, projeto esse que é feito para prever os problemas estruturais que podem existir, e combatê-los.

2.2.1 Modelos matemáticos de canais

Em todos os sistemas de comunicação existe um canal, que é o meio pelo qual a informação trafega entre o transmissor e o receptor. A modelagem de canais é importante pois cria-se um modelo referente às características físicas do canal, sem que seja preciso a implementação física para avaliação do sistema estudado.

Nessa seção trataremos de alguns modelos que demonstram a simplicidade na descrição de canais reais.

Canal sem memória

Um canal dito *sem memória* é um modelo de canal em que cada símbolo na entrada do receptor depende, estatisticamente, apenas do símbolo de entrada no canal. Matematicamente ele é descrito como

$$p(\mathbf{y}|\mathbf{x}) = \prod_{k=1}^{J} p(y_k|x_k), \tag{2.3}$$

ou seja, dada uma sequência de entrada $\mathbf{x} = (x_1, x_1, \ldots, x_J)$, a sequência de saída $\mathbf{y} = (y_1, y_1, \ldots, y_J)$, representada na Figura 2.3, é definida através da função de probabilidade dada pela equação 2.3. Um exemplo de modelo de canal sem memória é o canal DMC (do inglês, *Discrete Memoryless Channel*), que se caracteriza pelos alfabetos de entrada e saída do canal serem finitos.

Canal binário simétrico

Talvez o exemplo mais importante de canais DMC seja o canal BSC (do inglês, *Binary Symmetric Channel*). Neste canal, os símbolos de entrada e saída pertencem a um alfabeto binário, e os bits, quando passam por ele, se trocam com probabilidade p. ou seja, dado

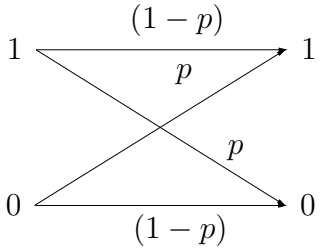


Figura 2.3: Modelo de canal BSC.

um bit de entrada do canal igual a 1, a saída do canal tem probabilidade $(1 - p)$ do bit não ter sido trocado pelo canal, e probabilidade p do bit ter sofrido troca, representado na Figura 2.3. O mesmo acontece com o bit 0 na entrada do canal. Ou seja, $p(y_k = 1|x_k = 0) = p(y_k = 0|x_k = 1) = p$.

Canal aditivo Gaussiano

Esse canal já foi mostrado através da equação 2.2. Sua probabilidade de transição é definida pela da densidade de probabilidade de uma variável aleatória gaussiana. Ou seja, dado um símbolo de entrada a_k no canal gaussiano, a saída é um número $r \in \mathbb{R}$, descrito por uma função gaussiana de média zero e variância σ_n^2 do ruído inserido pelo canal. Isto é

$$p(r_k|a_k) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(r_k - a_k)^2}{2\sigma_n^2}\right). \tag{2.4}$$

O canal aditivo gaussiano é um bom modelo para transmissão via satélite, no entanto para transmissões terrestres ele é comumente utilizado junto com multicaminhos, desvanecimento, bloqueio terrestre, e outros tipos de interferências previstas para esse tipo de comunicação.

2.2.2 Informação mútua

Em termos gerais, a informação mútua é uma medida da quantidade de informação que o conhecimento de uma variável aleatória nos traz em relação à outra. Essa medida deriva de conceitos importantes como *entropia*, *entropia condicional* e *entropia relativa* [9].

Primeiramente definimos entropia, que é uma medida de incerteza de uma variável aleatória. Considere X uma variável aleatória discreta de alfabeto $\mathcal{X}$ e função massa de probabilidade $p(x) = P(X = x),\ x \in \mathcal{X}$. Define-se x e y como duas variáveis aleatórias e suas respectivas funções massa de probabilidade $p(x)$ e $p(y)$. A entropia de x é definida como o valor esperado de $\log[1/p(x)]$, onde $p(x)$ é a probabilidade do evento x acontecer. Portanto,

$$H(X) = E\left\{\log\frac{1}{p(X)}\right\} = -\sum_{x\in\mathcal{X}} p(x)\log p(x). \tag{2.5}$$

E a entropia conjunta de duas variáveis aleatórias é

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y), \tag{2.6}$$

que é definida como

$$H(X,Y) = E\left\{\log\frac{1}{p(X,Y)}\right\}. \tag{2.7}$$

O cálculo da entropia condicional é definida como o valor esperado da entropia das variáveis condicionadas multiplicadas pela função de distribuição de probabilidade da variável

aleatória x, isto é,

$$H(X|Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x|y) \tag{2.8}$$

$$= E\left\{\log\frac{1}{p(X|Y)}\right\}. \tag{2.9}$$

A medida de entropia nos dá a quantidade de informação necessária para descrever uma variável aleatória, como ela é uma medida de informação, define-se sua unidade como dependente da base do log; portanto, no caso do log ter base dois, diz-se que a unidade da entropia é medida em bits.

Por sua vez, o conceito de entropia relativa é uma medida de distância entre duas distribuições. Por definição essa distância é

$$D(p||q) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}. \tag{2.10}$$

O que, finalmente, nos leva ao conceito de *informação mútua* $I(X,Y)$, que é a entropia relativa entre uma distribuição conjunta e o produto de distribuições $p(x_i)p(y_i)$. Por definição:

$$I(X,Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}. \tag{2.11}$$

No caso de a distribuição de probabilidade $p(y)$ ser contínua, então a infomação mútua $I(X,Y)$ é:

$$I(X,Y) = \sum_{x\in\mathcal{X}}\int p(x,y)\log\frac{p(x,y)}{p(x)p(y)}dy. \tag{2.12}$$

A equação 2.12 nos mostra o cálculo da informação mútua no caso específico em que a entrada x do canal é uma variável discreta, mas graças ao ruído inserido no sinal pelo canal, a saída pertence aos números reais, levando a uma distribuição de probabilidade contínua na saída do canal.

Além disso, a informação mútua entre duas variáveis aleatórias é medida através das relações

$$H(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{2.13}$$

Em outras palavras, a informação mútua nos diz a quantidade de informação que uma variável aleatória y têm em relação a outra x.

A capacidade de canal, que diz qual é o limite máximo de informação em bits possível de se transmitir por uso do canal, é medida através da máxima informação mútua.

2.2.3 Codificação de canal

Quando bits trafegam através de um canal ruidoso existe a possibilidade deles serem trocados durante o tráfego. Ou seja, o que é 0 passa a ser 1 ou o que é 1 passa a ser 0. Isso acontece dependendo da regra de decisão do demodulador, que pode interpretar um símbolo ruidoso como um outro símbolo. Com o intuito de recuperar essas trocas de símbolos provenientes do ruído do canal, alguns *bits de redundância* são adicionados aos bits de informação. Esses bits são adicionados de maneira regular, controlando a quantidade de trocas de bits que podem ser recuperados. Além disso, com a inserção desses bits existe a possibilidade de se corrigir os erros introduzidos pelo canal.

Ao processo de se adicionar bits de redundância aos bits de informação é dado o nome de *codificação de canal*. Quanto mais bits de redundância são adicionados à informação, maior capacidade de correção terá o receptor.

Códigos de bloco e convolucionais

Uma divisão bastante simplificada pode ser feita do conceito geral de codificação de canal utilizando dois tipos de códigos, os de bloco e os convolucionais. Além de os códigos convolucionais serem relativamente mais fáceis de se decodificar que os códigos de bloco, eles ficaram muito famosos graças aos códigos conhecidos como códigos turbo, que chegam muito próximos à capacidade de canal. Apesar disso, os códigos de bloco ainda são bastante utilizados e diversas pesquisas recentes demonstram a possibilidade de se chegar muito próximo à capacidade de canal, utilizando uma família específica de códigos de bloco, como mostrado em [3].

Estruturalmente se diz que a diferença mais básica entre esses dois tipos de códigos é que o código convolucional tem memória, enquanto o código de bloco não. Durante o processo de codificação dos códigos convolucionais cada bit na saída do codificador depende de bits anteriores. Esse processo pode ser interpretado como uma máquina de estados de memória finita, em que cada bit c_k na saída do codificador depende de m bits $\mathbf{x}_i$ na entrada do

decodificador, onde m é o tamanho da memória da máquina de estados referente ao código.

Códigos de bloco lineares

Define-se um código de bloco $C(n,k)$ linear, como o conjunto de seqüências resultantes de uma função linear que mapeia um vetor de k bits de informação $\mathbf{u} = [u_1, u_2, \ldots, u_k]$ em um outro bloco de n bits, que é a palavra codificada com os bits de paridade[1], onde $k \leq n$. Esse mapeamento é feito através de uma combinação linear de k vetores $\mathbf{g}$ de base linearmente independentes de tamanho n, gerando a palavra codificada $\mathbf{c} = [c_1, c_2, \ldots, c_n]$, isto é,

$$c_n = \sum_{i=1}^{k} u_i \cdot g_{i,n}. \tag{2.14}$$

No caso de códigos binários a operação de soma e multiplicação[2] são executadas em módulo-2. As leis cumulativa, associativa e distributiva são satisfeitas pelas operações módulo-2. Portanto a equação 2.14 é representada em forma binária por

$$c_n = u_1 \cdot g_{1,n} \oplus u_2 \cdot g_{2,n} \oplus \cdots \oplus u_k \cdot g_{k,n}. \tag{2.15}$$

Arranjando os vetores base em uma matriz $\mathbf{G}$ de tamanho $k \times n$, pode-se representar uma palavra código pela equação

$$\mathbf{c} = [u_1, u_2, \ldots, u_k] \begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k,1} & g_{k,2} & \cdots & g_{k,n} \end{bmatrix} = \mathbf{uG}; \tag{2.16}$$

em que $\mathbf{u}$ é um vetor de bits de informação e $\mathbf{G}$ é chamada de matriz geradora do código $C(n,k)$.

O código de bloco $C(n,k)$ é chamado de *código sistemático* quando $\mathbf{G} = [\mathbf{I}_k; \mathbf{P}^T]$, em que $\mathbf{I}_k$ é a matriz identidade de ordem k e $\mathbf{P}_{k\times(n-k)}$ é a matriz de paridade, a qual seus elementos determinam os bits de paridade obtidos a partir dos bits de informação. Os códigos de bloco que não apresentam essa construção recebem a denominação de *códigos não-sistemáticos*.

[1] De agora em diante chamaremos os bits de redundância de *bits de verificação de paridade*, ou simplesmente de *bits de paridade*.

[2] Denotados pelos operadores $\otimes$ e $\oplus$, que representam multiplicação e soma respectivamente. Em todo caso omitimos o operador de multiplicação, uma vez que a operação de multiplicação fica implícita.

Uma das razões para a utilização de códigos sistemáticos é a implementação do codificador. Ao recuperar a palavra-código, o decodificador simplesmente descarta os bits de paridade.

Para um código de bloco $C(n,k)$ definido por uma matriz geradora $\mathbf{G}$, existe uma matriz $\mathbf{H}_{m\times n}$, em que $m = n - k$, tal que

$$\mathbf{G}\mathbf{H}^T = 0. \tag{2.17}$$

A matriz $\mathbf{H}$ é chamada de *matriz de verificação de paridade*, e recebe este nome porque todas as palavras códigos $\mathbf{c}$ por ela multiplicadas têm como resultado um vetor nulo de tamanho m. Dessa forma, para saber se a palavra código $\mathbf{c}$ é uma palavra válida, verificamos:

$$\mathbf{c}\mathbf{H}^T = \mathbf{0}. \tag{2.18}$$

No caso de o código $C(n,k)$ ser um código sistemático, a matriz $\mathbf{H}$ é definida como $\mathbf{H} = [\mathbf{P}^T; \mathbf{I}_m]$.

É necessário enfatizar que a equação 2.18 não é uma equação de decodificação, ela somente informa se a palavra código pertence ao código. Isso é importante durante o processo de recepção, pois após o receptor fazer a estimação do bit r_k corrompido pelo ruído do canal a palavra código pode ser classificada como uma palavra pertencente ao código ou não. No caso de receptores iterativos, a equação 2.18 é utilizada como critério de parada das iterações, como será visto no Capítulo 3.

Exemplo:

A matriz geradora de um código $C(6,3)$ é mostrada pela matriz geradora $\mathbf{G}$ da equação 2.19. Para este código, existem 8 vetores de informação, uma vez que $2^k = 2^3 = 8$, e portanto, oito palavras código.

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{2.19}$$

As linhas da matriz $\mathbf{G}$ são os vetores linearmente independentes $\mathbf{g}_1$, $\mathbf{g}_2$ e $\mathbf{g}_3$ que formam todas as palavras código do código C.

Seja, por exemplo, o vetor de informação $\mathbf{u} = [0\ \ 1\ \ 1]$. Utilizando a

equação 2.16, encontramos a palavra codificada, isto é

$$\mathbf{c} = \mathbf{uG} = (0\ \ 1\ \ 1) \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = 0 \cdot \mathbf{g}_1 \oplus 1 \cdot \mathbf{g}_2 \oplus 1 \cdot \mathbf{g}_3 = [0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0]. \quad (2.20)$$

Como o código C é um código sistemático, podemos encontrar a matriz de verificação de paridade do código, isto é

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (2.21)$$

2.3 Regras de decisão

Seja $\mathbf{r}$ a seqüência tomada na saida do canal de comunicação. $P(\mathbf{c}|\mathbf{r})$ é chamada de probabilidade *a posteriori* e representa a probabilidade de $\mathbf{c}$ ser o modelo (nesse caso, a sequência de bits) que melhor representa a sequência recebida $\mathbf{r}$. Ela é dada por:

$$P(\mathbf{c}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{c})P(\mathbf{c})}{P(\mathbf{r})}; \quad (2.22)$$

Onde as probabilidades $P(\mathbf{c})$ e $P(\mathbf{r})$ são as probabilidades *a priori* e evidência respectivamente.

Diretamente da equação 2.22 deriva-se a regra de decisão MAP (do inglês, *Maximum a posteriori*), que consiste em maximizar a probabilidade *a posteriori*. Esta regra é definida como

$$\mathbf{c}^{MAP} = \arg\max_{\mathbf{c}} P(\mathbf{c}|\mathbf{r}). \quad (2.23)$$

Note que a supressão de $P(\mathbf{r})$ na equação 2.22 não altera a regra de decisão MAP, isso porque, para efeitos numéricos, esse é apenas um parâmetro de normalização. Caso as probabilidades $P(\mathbf{c})$ para os valores do alfabeto de $\mathbf{c}$ sejam igualmente prováveis, então pode-se suprimir $P(\mathbf{c})$ da equação 2.22 também, uma vez que o efeito de classificação permanece o mesmo, levando à uma forma simplificada do classificador. Ou seja, com a supressão da probabilidade *a priori* e da evidência, simplificamos a regra de decisão a tal ponto que precisa-se apenas da verossimilhança.

Isto nos leva à outra regra de decisão chamada de ML (Máxima Verossimilhança, do inglês, *Maximum Likelihood*).

$$\mathbf{c}^{ML} = \arg\max_{\mathbf{c}} P(\mathbf{r}|\mathbf{c}). \tag{2.24}$$

Naturalmente a regra MAP oferece melhor confiabilidade de decisão, isso se dá pelo fato de não existir garantia de que as probabilidades $P(\mathbf{c})$ sejam idênticas, entretanto, em aplicações práticas em que dificilmente é possível obter as probabilidades $P(\mathbf{c})$, MV é utilizada mesmo não sendo uma regra ótima.

Um dos algoritmos mais conhecidos que utiliza a regra de decisão MAP é o algoritmo BCJR [10]. Este algoritmo é utilizado em várias áreas de aprendizagem computacional, e, inclusive em problemas de decodificação. Para a regra MV, o algoritmo de Viterbi [11] é um dos algoritmos mais conhecidos, esse algoritmo é uma simplificação do algoritmo BCJR.

No Capítulo 3, será mostrado um algoritmo que pode chegar muito próximo da regra MAP para alguns tipos de códigos. Esse algoritmo é conhecido como algoritmo de passagem de mensagem, ou, em alguns casos, como algoritmo soma-produto.

Capítulo 3

Grafos-fatores e algoritmo soma-produto

Códigos LDPC, códigos concatenados, e qualquer outro tipo de código podem de alguma forma serem definidos por grafos-fatores. Esses grafos não somente descrevem o código, como também estruturam o algoritmo soma-produto (SP) para o processo de decodificação, o que pode ser reconhecido como uma das características mais importantes dos grafos-fatores. Utilizando o algoritmo SP no processo de decodificação é possível chegar a probabilidades de erro muito pequenas, caso esse algoritmo seja utilizado iterativamente. É claro que esse desempenho depende da estrutura do código utilizado no sistema de comunicação, entretanto, em muitos casos a decodificação iterativa baseada em grafos-fatores e algoritmo soma-produto alcança valores próximos ao limite de Shannon.

Os grafos-fatores mostram uma maneira fácil de visualizar as limitações do sistema que definem o código. Essas limitações estão diretamente ligadas ao algoritmo de decodificação referentes, por exemplo, à complexidade ou mais qualitativamente ao desempenho do decodificador iterativo.

É possível descrever um sistema de transmissão digital inteiro através de um grafo-fator, desde a fonte de bits, passando pelo modulador, canal, entrelaçador, todos conectados ao grafo do código, o que leva à idéia de um receptor unificado que apresenta funções como estimação de canal, equalização, sincronização, entre outras.

3.1 Grafos-fatores

Grafos-fatores são uma generalização dos grafos de Tanner [12], eles são capazes de descrever os códigos LDPC, facilitando o processo de decodificação pelo algoritmo SP. Estes grafos são chamados de grafos-fatores, pois com eles é possível fatorar as funções marginais de probabilidade para cada bit da palavra código.

Para que os grafos-fatores possam ser definidos, precisamos primeiramente introduzir o conceito de verificação marginal. Assim, considere uma função $g(x_1, x_2, \ldots, x_n)$ com n argumentos, e que para cada $i = 1, 2, \ldots, n$ existe um número finito de símbolos possíveis chamado domínio (ou alfabeto), definidos por A_i. Então o domíno da função $g(x_1, x_2, \ldots, x_n)$ é

$$S = A_1 \times A_2 \times \cdots \times A_n; \tag{3.1}$$

e o contradomínio $R \in \Re$. O domínio S é chamado de espaço de configuração, e cada elemento de S é uma configuração particular de variáveis.

Supondo que a soma em R seja bem definida, então, associadas com cada função $g(x_1, x_2, \ldots, x_n)$ existem n funções marginais $g_i(x_i)$; e para cada $a \in A_i$, o valor de $g_i(a)$ é obtido somando o valor de $g(x_1, x_2, \ldots, x_n)$ em todas as configurações de variável em que $x_i = a$.

$$g_i(x_i) := \sum_{\sim\{x_i\}} g(x_1, x_2, \ldots, x_n); \tag{3.2}$$

essa notação é chamada de não-soma[1] [7].

Suponha agora que a função $g(x_1, x_2, \ldots, x_n)$ pode ser fatorada em várias funções locais, cada uma tendo um subconjunto de $x_1, x_2, \ldots, x_n$ como argumento:

$$g(x_1, x_2, \ldots, x_n) = \prod_{j \in J} f_j(X_j); \tag{3.3}$$

onde J é um conjunto de índices, X_j é um subconjunto de $\{x_1, \ldots, x_n\}$ e $f_j(X_j)$ é uma função com os elementos de X_j como argumentos.

[1] por exemplo, considere h uma função de três variáveis x_1, x_2, x_3, então, a não-soma é definida como:

$$\sum_{\sim\{x_2\}} h(x_1, x_2, x_3) := \sum_{x_1 \in A_1} \sum_{x_3 \in A_3} h(x_1, x_2, x_3)$$

Define-se então um grafo fator como um modelo gráfico biparticionado[2] (Figura 3.1) capaz de representar a fatorização de funções marginais de forma simples, através de nós de verificação, que representam as funções $f_j(X_j)$ e nós de variável, que representam as variáveis x_i, conectados quando a variável x_i é argumento da função $f_j(X_j)$ [7].

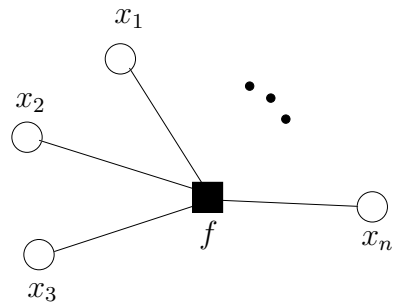


Figura 3.1: Grafo-fator da função $f(x_1, x_2, \ldots, x_n)$.

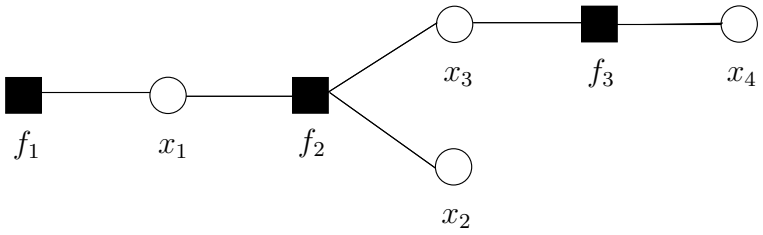


Figura 3.2: Grafo fator da função $g(x_1, x_2, x_3, x_4) = f_1(x_1)f_2(x_1, x_2, x_3)f_3(x_3, x_4)$.

Seja, por exemplo, a função global $g(x_1, x_2, x_3, x_4)$ que pode ser fatorada no produto de funções:

$$g(x_1, x_2, x_3, x_4) = f_1(x_1)f_2(x_1, x_2, x_3)f_3(x_3, x_4) \tag{3.4}$$

Onde deseja-se calcular as funções marginais $g_k(x_k), k \in 1, 2, 3, 4$. A Figura 3.2 representa o grafo-fator da equação 3.4, e através dela podemos calcular as funções marginais $g_k(x_k)$. Considere, por exemplo, que queremos obter a função marginal $g_1(x_1)$, pelo grafo da

[2]O termo biparticionado deve-se ao fato de o grafo-fator possuir apenas dois tipos de nós, os de variável e os de verificação.

Figura 3.2, sabemos que o nó de variável está ligado aos nós de função f_1, f_2 e f_3, ou seja:

$$g_1(x_1) = \sum_{x_2,x_3,x_4} g(x_1, x_2, x_3) \tag{3.5}$$

$$= \sum_{x_2,x_3,x_4} f_1(x_1) f_2(x_1, x_2, x_3) f_3(x_3, x_4) \tag{3.6}$$

$$= \sum_{\sim\{x_1\}} f_1(x_1) f_2(x_1, x_2, x_3) f_3(x_3, x_4). \tag{3.7}$$

E para calcular as outras i funções marginais procedemos da mesma forma.

É importante ressaltar que em grafos biparticionados a conexão entre os nós se realiza somente entre nós de tipos diferentes. Ou seja, em grafos-fatores os nós de verificação não se conectam com nós de verificação, e nós de variável não se conectam com nós de variável.

3.1.1 Cálculo das funções marginais

Existem dois tipos de mensagens que precisam ser calculadas: as que saem dos nós de variável e chegam aos nós de verificação, denotadas por $\mu_{x\to f}(x)$, e as que saem dos nós de verificação e chegam aos nós de variável, denotadas por $\mu_{f\to x}(x)$. A mensagem que trafega por uma conexão entre um nó de variável x e um nó de verificação f, é uma função de argumento x. Portanto, a mensagem não assume um valor categórico, e sim um conjunto de valores que dependem das variáveis e interligações do grafo-fator relacionado.

A execução do algoritmo é iniciada pelos nós de extremidade, ou seja, pelas extremidades do grafo fator, e calculadas passo-a-passo. Cada cálculo referente a um passo do algoritmo é executado atômicamente, o que significa que cada nó espera as mensagens provenientes dos outros nós conectados a ele para calcular a mensagem de saída. As mensagens que saem dos nós de extremidade são definidas pelas equações:

$$\mu_{x\to f}(x) = 1 \tag{3.8}$$

$$\mu_{f\to x}(x) = f(x) \tag{3.9}$$

O cálculo das mensagens é feito da seguinte forma: a mensagem que sai de um nó de variável x e segue a um nó de verificação f conectado a este nó de variável, é o produto das mensagens que chegam de outros nós de verificação conectados ao nó de variável menos a

mensagem que chega do nó de verificação f, que é o nó para qual a mensagem de saída será enviada. Já para os nós de verificação, o cálculo da mensagem que sai desses nós e vai até os nós de variável é feito em dois passos.

- Primeiro se calcula o produto da função que o nó representa pelas mensagens recebidas, exceto a do nó de destino;
- Depois, o produto gerado é submetido ao operador não-soma, para a obtenção da mensagem que sai do nó[3].

Como exemplo a função marginal do grafo-fator da Figura 3.2 é calculado de acordo com os passos descritos na Figura 3.3.

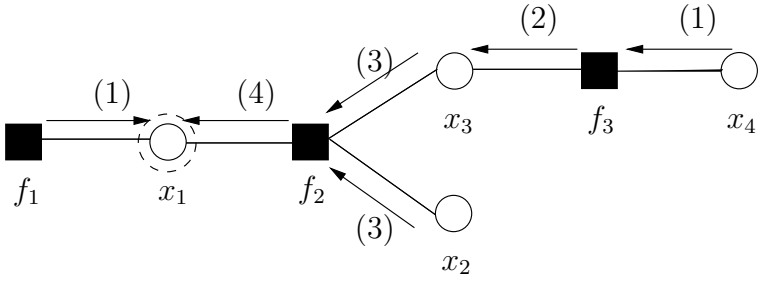


Figura 3.3: Iterações para o cálculo da função marginal $g_1(x_1)$ no grafo da função $g(x_1, x_2, x_3, x_4)$.

- **Passo (1)**
  - ▷ $\mu_{f_1 \to x_1}(x_1) = f_1(x_1)$
  - ▷ $\mu_{x_4 \to f_3}(x_4) = 1$
- **Passo (2)**
  - ▷ $\mu_{f_3 \to x_2}(x_2) = \sum_{\sim\{x_1\}} \mu_{x_4 \to f_3}(x_4)\, f_1(x_3, x_4)$
- **Passo (3)**
  - ▷ $\mu_{x_3 \to f_2}(x_3) = (1)\, \mu_{f_3 \to x_3}(x_3)$

[3]A não-soma é definida pelo sentido da mensagem, o nó que não será somado é o nó que receberá a mensagem.

▷ $\mu_{x_2 \to f_2}(x_2) = 1$

- **Passo (4)**

▷ $\mu_{f_2 \to x_1}(x_1) = \sum_{\sim\{x_1\}} f_1(x_1, x_2, x_3)\, \mu_{x_2 \to f_2}(x_2)\, \mu_{x_3 \to f_2}(x_3)$

- **Finalização**

▷ $g_1(x_1) = \mu_{f_1 \to x_1}(x)\, \mu_{f_2 \to x_1}(x)$

As outras funções são calculadas da mesma forma, cada função $g_i(x_i)$ por vez, entretanto calcular as funções uma a uma é pouco eficiente, uma vez que mensagens podem ser reutilizadas no cálculo de funções $g(x)$ diferentes. Portanto, executa-se um processo onde é possível calcular todas as funções marginais simultaneamente.

O cálculo das mensagens de saída em cada nó, envolve todos os nós conectados a ele menos o nó que será o destino da mensagem enviada. Tomamos como exemplo a Figura 3.4, a mensagem $\mu_{x \to f}(x)$, que sai do nó de variável x e destina-se ao nó de verificação f, envolve todos os nós de verificação ligados a x menos o f. O mesmo acontece ao nó de verificação f na mensagem $\mu_{f \to x}(x)$ enviada a x.

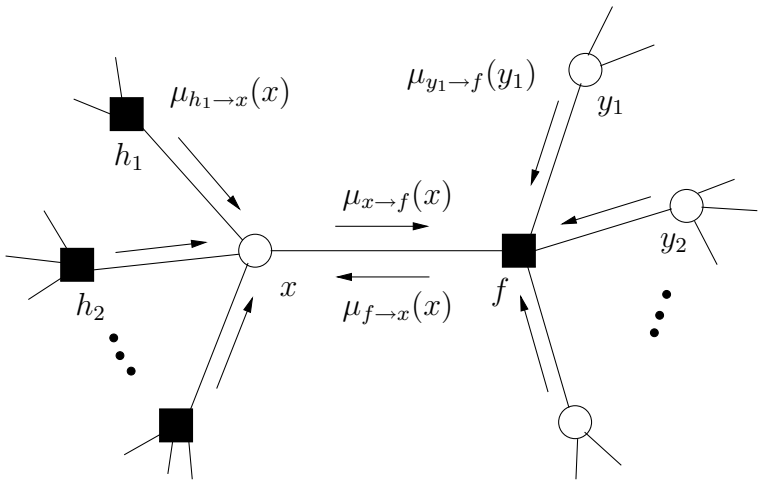


Figura 3.4: Caminho das mensagens ao percorrerem o grafo. Generalização do algoritmo soma-produto.

Dessa forma, define-se a notação $y \in n(f) \backslash x$, que indica que a operação do cálculo das mensagens é feita em todos os nós y ligados ao nó f exceto o nó x. As equações 3.10 e 3.11

utilizam a notação que seleciona qual dos nós está envolvido na operação e dizem respeito às mensagens que serão enviadas aos nós de verificação e aos nós de variável respectivamente:

$$\mu_{x\to f}\{x\} = \prod_{y\in n(f)\setminus x} \mu_{h\to f}\{y\}; \tag{3.10}$$

$$\mu_{f\to x}\{x\} = \sum_{\sim\{x\}} f(x) \left(\prod_{y\in n(f)\setminus x} \mu_{y\to f}\{y\} \right). \tag{3.11}$$

A utilização das equações 3.10 e 3.11 é de suma importância quando se deseja trabalhar com grafos-fatores muito grandes, uma vez que calcular as funções marginais uma a uma se transforma numa tarefa muito exaustiva. Na verdade, sistemas descritos por grafos-fatores em geral são sistemas complexos que envolvem um número de nós muito grande, como exemplos desses sistemas citamos códigos de bloco, Cadeias de Markov, Modelos de Markov Escondidos, e treliças de códigos convolucionais.

3.2 Códigos LDPC

Seu nome é uma sigla do que no inglês é conhecido como *Low-Density Parity-Check Codes*. Esta família de códigos corretores de erros são definidas pela quantidade de 1's em sua matriz de paridade, e foi proposta na década de sessenta por Gallager em sua tese de doutorado [2]. Na época, com a tecnologia que existia, não era possível a criação de sistemas de comunicação que utilizassem esses tipos de códigos, e apesar de esses códigos alcançarem probabilidade de erro muito baixa, eles permaneceram pouco estudados, até que em 1981 Tanner estudou uma representação para códigos de bloco através de grafos [12]. Mas foi graças ao trabalho de MacKay *et al* [3] [13], que mostrou as vantagens de códigos lineares de bloco baseados em matrizes de paridade esparsas, que os códigos LDPC tomaram notoriedade, quase trinta anos depois de sua descoberta.

Somente no início desse século é que as primeiras aplicações que utilizam esta família de códigos começaram a aparecer. Padrões novos, e importantes, começaram a utilizar este código, exemplos são: WiMAX, e DVB-S2 (segunda geração de transmissão de vídeo digital via satélite).

3.2.1 Propriedades

Um código LDPC é um código linear de bloco, definido por uma matriz de paridade **H** com baixa densidades de 1's. Existem várias maneiras de se construir um código LDPC [14] [15] [16], entretanto, pode-se dividir os códigos LDPC em duas classes que dizem respeito à quantidade de 1's em suas linhas e colunas. Essas classes são chamadas de *regulares* e *irregulares*.

Os códigos regulares são códigos que tem o mesmo grau, número de 1's, em todas as colunas e em todas as linhas, e não necessariamente as colunas têm o mesmo grau das linhas. O grau das colunas é medido por w_c e o grau das linhas por w_r.

A Equação 3.12 mostra um exemplo de uma matriz esparsa de paridade para um código LDPC regular[4] $\mathbf{C}(3,4)$.

$$\mathbf{H} = \left(\begin{array}{cccccccccccccccccccc}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&0&0\\
\hline
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0\\
\hline
1&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0\\
0&1&0&0&0&0&1&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0\\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&1&0&0&0\\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&1&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0
\end{array}\right) \tag{3.12}$$

Para códigos regulares a taxa $R = k/n$ é relacionada aos parâmetros w_c e w_r através da fórmula $R = 1 - w_c/w_r$. Isso é facilmente demonstrado já que $m \cdot w_r = n \cdot w_c$, onde m e n são as dimenções da matriz de paridade **H** do código.

Os códigos irregulares, como é de se esperar, não mantêm a regularidade de 1's em suas linhas e colunas, ou seja, os parâmetros w_c e w_r não são medidos como nos códigos regulares. Ao invés disso, são definidos *polinômios de distribuição de grau* que definem os graus das linhas e colunas da matriz de paridade.

Os polinômios $\lambda(x)$ e $\rho(x)$ representam os graus das linhas e colunas respectivamente [16], e são definidos através das equações 3.13 e 3.14, como segue.

Seja $\lambda(x)$ o polinômio de distribuição de grau para as linhas da matriz, então

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1} \tag{3.13}$$

[4] A notação em negrito representa os graus das linhas e colunas da matriz de paridade respectivamente, utilizaremos essa notação para descrever os códigos LDPC, ao contrário da notação em itálico que descreve os tamanhos em bits das linhas e colunas da matriz geradora do código.

O parâmetro λ_d representa a fração de colunas da matriz $\mathbf{H}$ de grau d, e d_v representa o grau máximo entre as colunas. Para os nós de verificação utiliza-se a mesma abordagem, isto é

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1}, \tag{3.14}$$

onde d_c é o grau máximo entre as colunas. A taxa de códigos LDPC irregulares é dada pela equação

$$R_c = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}. \tag{3.15}$$

Os códigos LDPC irregulares alcançam probabilidades de erro muito menores que os códigos LDPC regulares, além disso, eles ficam muito próximos à capacidade de canal imposta por Shannon [3] [4]. É correto afirmar ainda, que esses códigos são uma generalização dos códigos LDPC regulares, onde os graus dos nós de variável são os mesmos para todos os nós de variável, assim como para os nós de verificação.

Deve-se notar que códigos LDPC irregulares, apresentam uma quantidade menor de ciclos do que nos códigos LDPC regulares, o que leva a um melhor desempenho do algoritmo SP.

3.3 Grafo-fator de código LDPC e algoritmo SP

O algoritmo de decodificação (algoritmo SP) de códigos LDPC é baseado no grafo-fator deste código. É necessário então que se tenha a matriz de paridade $\mathbf{H}$ para definição das mensagens que são repassadas. O grafo-fator de um código qualquer com matriz de paridade $\mathbf{H}$ é obtido como segue.

As colunas da matriz $\mathbf{H}$ do código LDPC $C(n,k)$ representam os nós de variável, enquanto as linhas da mesma matriz representam os nós de verificação. Portanto, tem-se que o grafo-fator relacionado ao código $C(n,k)$ terá k nós de verificação e $n-k$ nós de variável. As conexões entre os nós de verificação e os nós de variável no grafo-fator são caracterizadas pelos 1's que existem na matriz $\mathbf{H}$ de verificação, ou seja, um 1 presente na posição $a_{i,j}$ da matriz, onde $i = 1, 2, \ldots, n-k$ e $j = 1, 2, \ldots, k$, indica que no grafo fator existirá uma conecção entre o i-ésimo nó de variável com o j-ésimo nó de verificação.

Exemplo:

Considere um código LDPC $\mathbf{C}(2,4)$ de taxa $r_c = 0,5$, cujo a matriz de verificação é dada por:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{3.16}$$

Seu grafo-fator é:

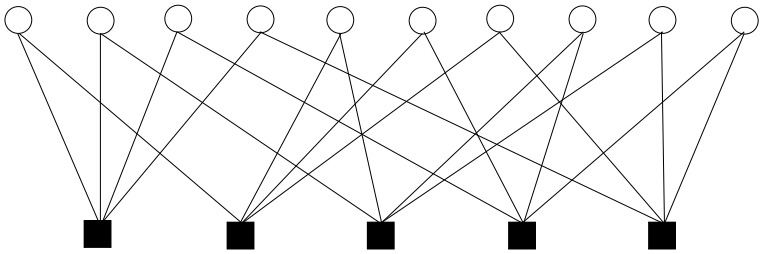


Figura 3.5: Grafo-fator do código definido pela matriz de paridade da equação 3.16

O grafo-fator de um código de bloco é um tipo de grafo particular pois ele apresenta ciclos. Ciclos em um grafo-fator indicam que o cálculo das mensagens será feito iterativamente, e por isso, o valor calculado das funções marginais não alcançará o valor exato. A Figura 3.6 mostra um exemplo de um grafo com ciclo.

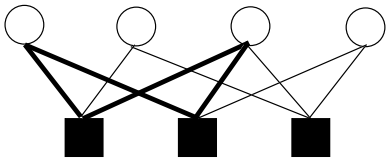


Figura 3.6: Exemplo de grafo de um código $C(3,4)$, as conexões em negrito indicam um ciclo.

Como o grafo-fator é um grafo biparticionado, as equações de decodificação para códigos de bloco são bem definidas (equações 3.10 e 3.11). Na sessão seguinte mostraremos as equações de mensagem entre os nós de verificação e os nós de variável. Entretanto, apesar das equações serem bem definidas, os ciclos nos grafos-fatores dos códigos não calculam o

valor exato da função marginal. Por isso, quanto menos ciclos o código tiver, um desempenho mais próximo do ótimo será alcançado. Essa observação indica empiricamente que códigos LDPC irregulares têm desempenho superior do que códigos LDPC regulares.

3.4 O problema da decodificação e o algoritmo SP em canal gaussiano

Dado um código LDPC definido por uma matriz $\mathbf{H}_{m\times n}$, tem-se que a saída $\mathbf{r} = [r_1, r_2, \ldots r_n]$ de um canal gaussiano é relacionada com a palavra código transmitida por:

$$\mathbf{r} = 2\mathbf{c} - 1 + \mathbf{n}, \tag{3.17}$$

onde os componentes do vetor $\mathbf{n}$ são variáveis aleatórias gaussianas de média zero e variância σ^2.

O problema da decodificação se divide em duas etapas:

1. O detector que minimiza a a probabilidade de erro do n-ésimo bit pode ser um LLR-MAP, que maximiza a probabilidade *a posteriori* do valor LLR do bit recebido:

$$\begin{aligned} \mathrm{LLR}(\hat{c}_n) &= \ln \frac{p(c_n = 1|\mathbf{r})}{p(c_n = 0|\mathbf{r})} && (3.18)\\ &= \ln \frac{p(c_n = 1|r_n, \{r_{i\neq n}\})}{p(c_n = 0|r_n, \{r_{i\neq n}\})}, && (3.19) \end{aligned}$$

2. e então decide entre $\hat{c} = 1$, caso $\mathrm{LLR}(\hat{c}) > 0$, ou $\hat{c} = 1$, no caso contrário.

Entretanto, um detalhamento da primeira etapa deve ser feito. Utilizando a regra de Bayes no numerador e no denominador, podemos reescrever a equação 3.19 de uma maneira simplificada, isto é:

$$\mathrm{LLR}(\hat{c}_n) = \ln \frac{p(r_n|c_n = 1)}{p(r_n|c_n = 0)} + \ln \frac{p(c_n = 1|\{r_{i\neq n}\})}{p(c_n = 0|\{r_{i\neq n}\})}. \tag{3.20}$$

Para simplificar ainda mais, reescrevemos $p(r_n|c_n = 1)/p(r_n|c_n = 0)$, como $2/\sigma^2\, r_n$ uma vez que o canal é AWGN. Chegamos então a equação:

$$\mathrm{LLR}(\hat{c}_n) = \underbrace{\frac{2}{\sigma^2} r_n}_{\text{intrínseco}} + \underbrace{\ln \frac{p(c_n = 1|\{r_{i\neq n}\})}{p(c_n = 0|\{r_{i\neq n}\})}}_{\text{extrínseco}}, \tag{3.21}$$

O primeiro termo representa a contribuição da n-ésima observação do canal, e é chamado de *informação intrínseca*, enquanto o segundo termo representa a contribuição das outras observações, e é chamado de *informação extrínseca*.

3.4.1 O algoritmo SP

O algoritmo soma-produto para códigos LDPC binários é determinado pelos conjuntos de índices $\mathcal{M}_n = \{m : H_{m,n} = 1\}$ e $\mathcal{N}_m = \{n : H_{m,n} = 1\}$, onde $H_{m,n} = 1$, são todos os índices m e n na matriz de verificação de paridade que são iguais a 1. Fazendo $u_{m,n}^{(l)}$ serem as mensagens que vão dos nós de verificação de índice m aos nós de variável de índice n durante a l-ésima iteração, obtemos o algoritmo como no quadro abaixo [17]:

1. inicialização -
   - $u_{m,n}^{(0)} = 0,$ para todos $m \in \{1, \ldots, M\}$ e $n \in \mathcal{N}_m$;
   - $\lambda_n^{(0)} = \frac{2}{\sigma^2}\, r_n,$ para todos $n \in \{1, \ldots, N\}$.
2. iterações - conta em l até o máximo de iterações l_{max}
   - *- atualização dos nós de verificação -*
     para todos $m \in \{1, \ldots, M\}$ e $n \in \mathcal{N}_m$:

$$u_{m,n}^{(l)} = -2\tanh^{-1}\left(\prod_{i \in \mathcal{N}_m - n} \tanh\left(\frac{-\lambda_i^{(l-1)} + u_{m,i}^{(l-1)}}{2}\right)\right); \quad (3.22)$$

   - *- atualização dos nós de variável -*
     para todos $n \in \{1, \ldots, N\}$:

$$\lambda_n^{(l)} = \frac{2}{\sigma^2}\, r_n + \sum_{m \in \mathcal{M}_n} u_{m,n}^{(l)}. \quad (3.23)$$

Na primeira fase, a de inicialização, as variáveis u e λ são inicializadas. Lambda é inicializado com os bits que saem do canal $\mathbf{r} = [r_1, r_1, \ldots, r_n]$ enquanto u é uma variável que auxilia no cálculo das mensagens parciais até que o algoritmo alcance alguma condição de parada.

A segunda fase, que é a fase iterativa, se divide em dois passos: o de atualização dos nós de verificação de paridade e o de atualização dos nós de variável. As iterações continuam

ocorrendo até que se alcance uma das condições de parada, que são:

1. O número máximo l_{max} de iterações foi alcançado;

2. Uma palavra código válida foi encontrada.

Caso não exista uma das duas condições de parada o sistema pode ficar na fase de iteração indefinidamente, sem encontrar uma palavra código válida. As equações 3.22 e 3.23 calculam as mensagens que chegarão aos nós de variável e aos nós de verificação. As Figuras 3.7 e 3.8 mostram o comportamento dos passos de *atualização dos nós de verificação* e *atualização dos nós de variável*.

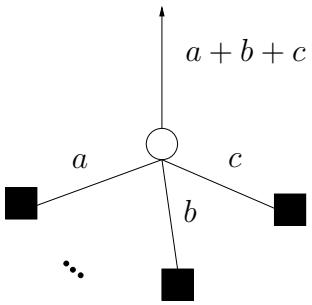


Figura 3.7: Exemplo de mensagem que sai de um nó de variável

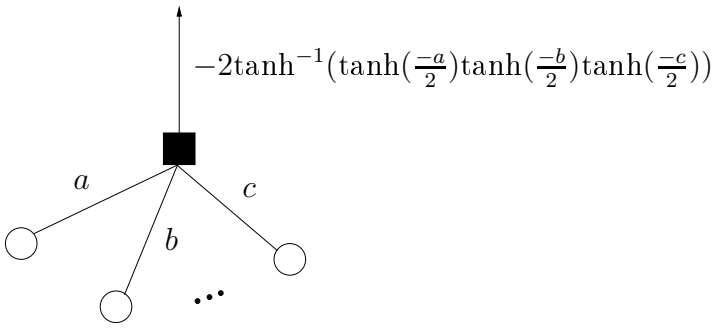


Figura 3.8: Exemplo de mensagem que sai de um nó de verificação

As mensagens a, b e c que chegam ao nó de variável são somadas, como mostrado na Figura 3.7. As mensagens que saem dos nós de verificação, entretanto, são processadas de acordo com a equação 3.22, onde o resultado obtido é a tangente hiperbólica do produto da tangente das mensagens que chegam ao nó.

A utilização do operador $tanh$ e $tanh^{-1}$, vem diretamente da equação 3.11 com a utilização de valores LLR para o cálculo das funções marginais [17]. Existe ainda uma aproximação da equação 3.11 que pode ser utilizada, como visto em [18] e [19].

Capítulo 4

Funções EXIT

As funções EXIT são uma proposta recente para examinar a transferência da informação extrínseca durante a decodificação em receptores iterativos. Essas curvas foram primeiramente propostas em [5] e depois extendidas para o uso em códigos paralelos concatenados (PCC, do inglês *Parallel Concatenated Codes*) [6]. A motivação para a utilização dessas curvas no projeto de códigos LDPC surgiu pois estudos para PCC mostraram bons resultados para análise de códigos, de forma rápida.

O nome EXIT vem do inglês, ***EX****trinsic* ***I****nformation* ***T****ransfer Function*. As curvas são calculadas utilizando as informações mútuas entre os bits nas saídas dos decodificadores e os bits transmitidos, que são conhecidos, monitorando o comportamento do decodificador.

4.1 Informação intríseca e extrínseca de códigos PCC e LDPC

Considere a Figura 4.1, que representa um modelo em blocos de um decodificador PCC. No caso do código PCC ser sistemático, os bits na saída do canal são representados por p_1 e p_2, que são, respectivamente, os bits de paridade para os decodificadores A e B, e i, que são os bits de informação. Para cada iteração, o primeiro decodificador (Dec. A) recebe a informação *intrínseca*, as observações do canal Z_1, e então repassa a informação como E_1 que é a informação *extrínseca* do decodificador A, mas que passa a ser a informação *a priori* para o segundo decodificador (Dec. B). O processo se repete continuamente por várias iterações, até que se obtenha uma estimação válida da palavra código recebida, essa estimação pode

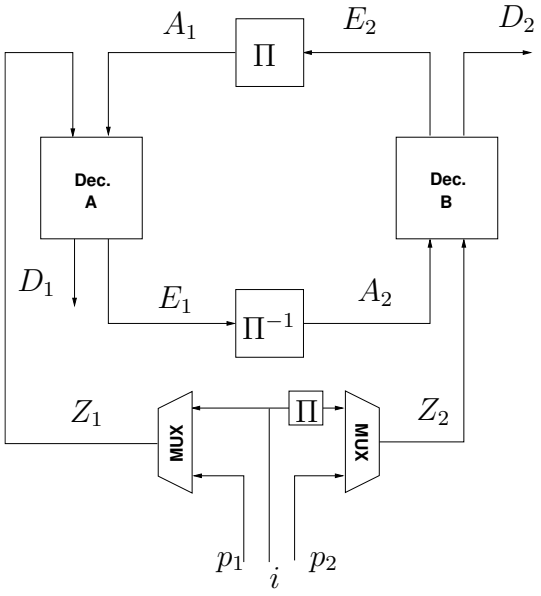


Figura 4.1: Diagrama de blocos do decodificador PCC.

ser obtida através de uma das duas variáveis D_1 e D_2.

Nos códigos PCC, cada curva EXIT é referente a um decodificador [5]. No caso desses decodificadores serem iguais, ou em outras palavras os códigos concatenados serem iguais, as curvas EXIT para esses decodificadores também serão iguais.

Em códigos LDPC, as curvas são diferentes das curvas para PCC. A decodificação desses códigos é feita através de um grafo fator biparticionado, e por definição, cada nó do grafo de decodificação realiza operações bem definidas. Diferente das curvas dos PCC, as curvas EXIT dos códigos LDPC sempre serão diferentes, pois os nós de variável e de verificação de paridade têm características distintas.

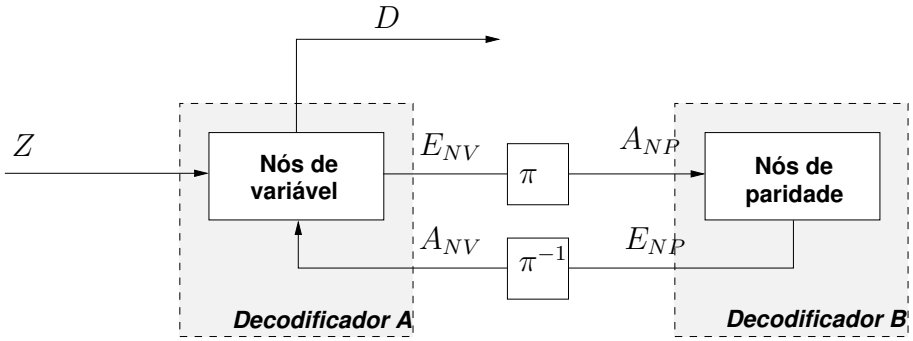


Figura 4.2: Diagrama de blocos do decodificador LDPC.

A Figura 4.2 mostra os decodificadores para códigos LDPC. Em comparação com o decodificador do código PCC, o decodificador A representa os nós de variável e cada nó de variável está ligado a w_r nós de verificação, enquanto o decodificador B representa os nós de verificação e cada nó está ligado a w_c nós de variável.

4.2 Cálculo da informação mútua

Em [20] alguns esclarecimentos são dados a respeito de porque utilizar informação mútua para as curvas EXIT. Duas informações mútuas são calculadas: uma entre os sinais de entrada dos decodificadores e os bits codificados, e outra entra os sinais de saída dos decodificadores e, também, os bits codificados. Para isso, é preciso obter os sinais da saída dos decodificadores de acordo com a saída do canal. As mensagens de saída do canal são probabilidades de transição dos bits, e seus valores são calculados na forma de LLR. Considere r a saída de um canal gaussiano, onde n é distribuida com média $\mu = 0$ e variância σ_n^2. As probabilidades de transição do canal são dadas por:

$$p(r|x = \pm 1) = \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left[-\frac{(r-x)^2}{2\sigma_n^2}\right]. \tag{4.1}$$

A entrada dos decodificadores é uma LLR, obtida com as probabilidades da equação 4.1:

$$L_Z = \mathrm{LLR}(r) = \ln\left[\frac{p(r|x=+1)}{p(r|x=-1)}\right] = \frac{2}{\sigma_n^2} r = \frac{2}{\sigma_n^2}(x+n) \tag{4.2}$$

que agora passa a ser uma nova variável aleatória gaussiana, chamada Z, onde:

$$Z = \mu_Z \cdot x + n_Z, \tag{4.3}$$

de média

$$\mu_Z = \frac{2}{\sigma_n^2} \tag{4.4}$$

e n_Z sendo uma variável aleatória gaussiana de média zero e variância

$$\sigma_Z^2 = \frac{4}{\sigma_n^2}. \tag{4.5}$$

A média e a variância de Z são relacionadas por

$$\mu_Z = \frac{\sigma_Z^2}{2}. \tag{4.6}$$

Uma vez definido o sinal de entrada do decodificador como uma variável aleatória gaussiana Z, tem-se que definir agora, a distribuição de probabilidades na saída dos decodificadores A e B para que o cálculo da informação mútua possa ser efetuado. Como será visto mais a frente, considerar a variável aleatória na saída dos nós, tanto os de variável como os de verificação, como uma variável aleatória gaussiana é condição suficiente. Portanto, supondo A como uma LLR de distribuição gaussiana, assim como no caso da equação 4.3, a média deve obedecer a equação 5.28. Assim, a função densidade de probabilidade condicional para as mensagens que entram no decodificador A é

$$p_A(\xi|X=x) = \frac{1}{\sqrt{2\pi}\sigma_A} \exp - \frac{(\xi - (\sigma_A^2/2)x)^2}{2\sigma_A^2}. \tag{4.7}$$

Para o medir o conteúdo de informação do conhecimento *a priori*, a informação mútua [9]

$$I_A = I(X;A) = \frac{1}{2} \sum_{x=\pm 1} \int_{-\infty}^{\infty} p_A(\xi|X=x) \log_2 \left[\frac{2p_A(\xi|X=x)}{p_A(\xi|X=-1) + p_A(\xi|X=1)} \right] d\xi \tag{4.8}$$

entre os bits sistemáticos X transmitidos e o sinal A é utilizada, e $0 \leq I_A \leq 1$.

Substituindo a equação 4.7 na equação 4.8, encontramos a informação mútua calculada em função de σ_A:

$$I_A = J(\sigma_A) = 1 - \int_{-\infty}^{\infty} \frac{e^{-\left(\xi - \sigma_A^2/2\right)^2 / 2\sigma_A^2}}{\sqrt{2\pi\sigma_A^2}} \cdot \log_2[1 + e^{-\xi}] d\xi, \tag{4.9}$$

e, por definição:

$$\lim_{\sigma \to 0} J(\sigma_A) = 0, \qquad \lim_{\sigma \to \infty} J(\sigma_A) = 1; \qquad \text{para } \sigma_A > 0.$$

Além disso, a função $J(\sigma_A)$ é reversível, e sua inversa é expressa por:

$$\sigma_A = J^{-1}(I_A). \tag{4.10}$$

As curvas das funções $J(\sigma_A)$ e $J^{-1}(I_A)$ podem ser calculadas através de simulação monte carlo (medidas de histograma). Em [6] e [21] é feita uma aproximação numérica das curvas $J(\sigma_A)$ e $J^{-1}(I_A)$ (vide anexo A), tornando possível substituir o processo de simulação da decodificação.

Não se pode afirmar, entretanto, que a distribuição de probabilidade da variável extrínseca dos decodificadores é uma variável aleatória normal. Para códigos LDPC afirmar que os nós

de variável apresentam uma variável aleatória normal em sua saída é plausível, mas para os nós de verificação as mensagens E não são variáveis aleatórias normais e sua distribuição de probabilidade deve ser obtida por medidas de histograma.

Para quantificar a informação extrínsica, a informação mútua também é utilizada. Através da equação 4.11, é obtido a informação mútua entre as variáveis E, que não são variáveis aleatórias normais, e X.

$$I_E = I(X;E) = \frac{1}{2}\sum_{x=\pm 1}\int_{-\infty}^{\infty} p_E(\xi|X=x)\log_2\left[\frac{2p_E(\xi|X=x)}{p_E(\xi|X=-1)+p_E(\xi|X=1)}\right]d\xi. \quad (4.11)$$

Em outras palavras, a informação intríseca I_A nos diz o quanto de informação a variável aleatória A contém do sinal enviado X, analogamente, a informação extrínsseca I_E nos diz o quanto de informação a variável aleatória E contém sobre o sinal enviado X.

4.2.1 Funções EXIT

A Figura 4.3 mostra um exemplo de uma curva EXIT para um código de taxa $R = 0,5$ em um canal gaussiano de $E_b/N_0 = 3$ dB. A curva superior mostra a informação mútua das

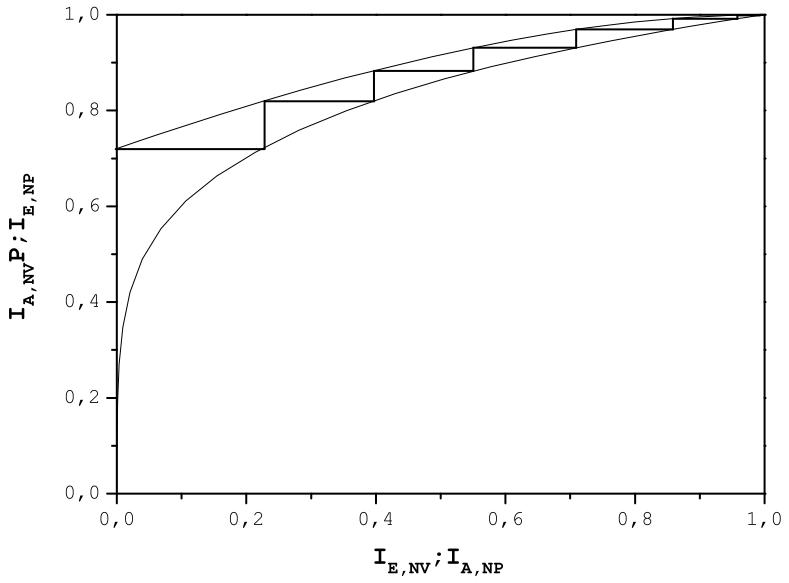


Figura 4.3: Curvas EXIT de um código de taxa $r = 0,5$ e $\frac{E_b}{N_0} = 3$ dB.

variáveis de saída dos nós de variável, $I_{E_{NV}}$, como função da informação mútua da entrada desses mesmos nós $I_{A_{NV}}$, enquanto a inferior mostra a informação mútua do sinal de entrada dos nós de verificação, $I_{A_{NP}}$, como função da informação mútua da saída desses mesmos nós, $I_{E_{NP}}$.

A trajetória da decodificação também é mostrada na Figura 4.3 representada pelas retas em negrito. Como na decodificação iterativa as mensagens passam pelos nós de variável e de verificação em cada iteração, a trajetória de decodificação vai dos nós de variável aos nós de verificação, e novamente retornam aos nós de variável, esse processo se repete em cada iteração. As funções EXIT são, portanto, funções recursivas, onde em cada iteração a informação mútua calculada é função da informação mútua calculada no outro decodificador, isto é,

$$I^{l}_{E_{NV}} = T(I^{l-1}_{A_{NV}} \rightarrow I^{l-1}_{E_{NP}} = T(I^{l-1}_{A_{NP}} \rightarrow I^{l-1}_{E_{NV}} = T(I^{l-2}_{A_{NV}} = \ldots))), \tag{4.12}$$

onde l é o número de iterações do decodificador.

Com o auxílio das funções EXIT, é possível prever o comportamento do decodificador durante o processo de decodificação, essas funções mostram a trajetória da decodificação iterativa através da informação mútua. Em outras palavras, elas mostram o quanto as variáveis intrínsecas e extrínsecas têm de informação sobre o bit transmitido x no decorrer do processo de decodificação, além disso, a trajetória da decodificação varia sobre o nível de ruído inserido pelo canal.

4.3 Funções EXIT dos nós de variável

As curvas dos dois decodificadores são calculadas de maneiras distintas, portanto, para cada decodificador é necessário que se obtenha a distribuição de probabilidade em sua saída.

Para o cálculo das curvas EXIT dos nós de variável tem-se que, da definição do algoritmo soma-produto, a saída dos nós de variável é a soma das mensagens que chegam até o nó, menos a mensagem do nó de destino. Supondo que um nó de variável tem grau w_r, então existem w_r+1 mensagens de entrada (as w_r mensagens dos nós de verificação conectados, mais uma que vem do canal) e a operação realizada pelo algoritmo soma-produto na decodificação

dos nós de variável faz a saída do decodificador A ser

$$E_{NV_i} = Z_i + \sum_{j \neq i} A_{NV_j}, \tag{4.13}$$

onde A_{NV_j} são as w_r variáveis aleatórias que vêm dos nós de verificação ligados ao nó i de variável, Z_i é o valor-L dos bits na saída do canal gaussiano e i e j são os índices dos nós de variável e verificação, respectivamente. Note que para cada E_{NV_i}, apenas w_r mensagens são repassadas ($w_r - 1$ dos nós e uma do canal).

Através de medidas de histograma encontra-se a distribuição de probabilidade da saída dos nós de variável e se executa o cálculo da informação mútua. As medidas de histogramas devem ser tomadas para vários valores de σ_Z na entrada do decodificador, calculando a informação mútua até que se encontre a função EXIT dos nós de variável.

Entretanto, podemos evitar os cálculo de histograma utilizando uma aproximação da função $J(\cdot)$. Para os nós de verificação, vemos que a função extrínseca $I_{E_{NV}}(\sigma)$ é função de $I_{A_{NV}}$, w_r, E_b/N_0, e R, e, portanto, calcula-se o valor da informação mútua como $J(\sigma_Z)$ na saída do decodificador:

$$I_{E_{NV}}\left(I_{A_{NV}}, w_r, \frac{E_b}{N_0}, R\right) = J\left(\sqrt{(w_r - 1)[J^{-1}(I_{A_{NV}})]^2 + \sigma_Z^2}\right). \tag{4.14}$$

Pelo teorema central do limite tem-se que a variável aleatória gerada pela soma de um número grande de variáveis aleatórias se aproxima de uma variável aleatória normal. A variância da variável normal gerada é a soma de todas as variâncias envolvidas na soma. Por isso, no argumento da equação 4.14 as variâncias são somadas.

No caso de códigos LDPC a distribuição de A_{NV_i} é gaussiana para um número grande de iterações, como mostrado em [22] e [23]. A Figura 4.4 mostra o comportamento da informação mútua $I_{A_{NV}}(\sigma)$. Não é preciso uma variação muito grande de σ_A para obter $I_{A_{NV}}(\sigma) = 1$. De fato, um valor de σ_A próximo a 7,2 é suficiente.

A distribuição de E_{NV_i} pode ser aproximada por uma gaussiana (Figura 4.7) ou mesmo utilizar as medidas de histograma. A Figura 4.5 foi obtida através de aproximações por Gaussianas para diferentes valores de w_r. No entanto utilizar as aproximações das funções $J(\cdot)$ e $J^{-1}(\cdot)$ mostradas em [21] é uma maneira mais fácil, rápida e bem confiável de conseguir

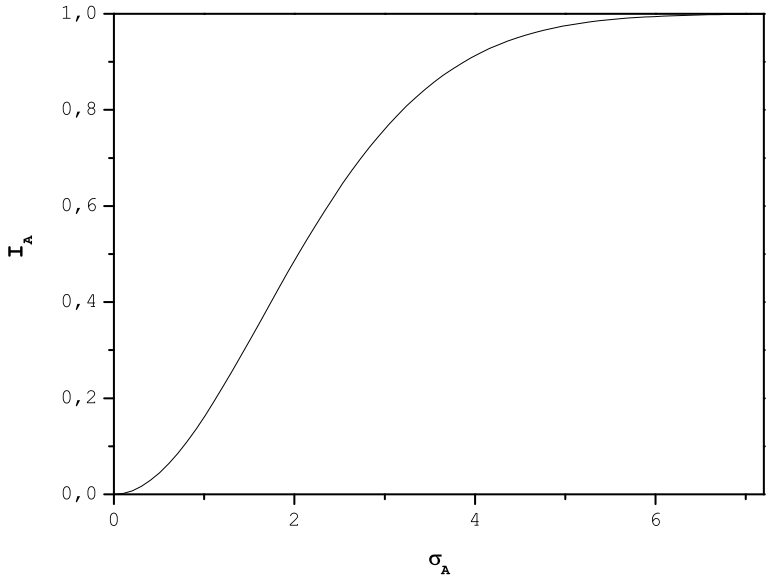


Figura 4.4: Gráfico da informação intrínseca $I_{A_{NV}}$ variando-se σ_A na entrada do decodificador.

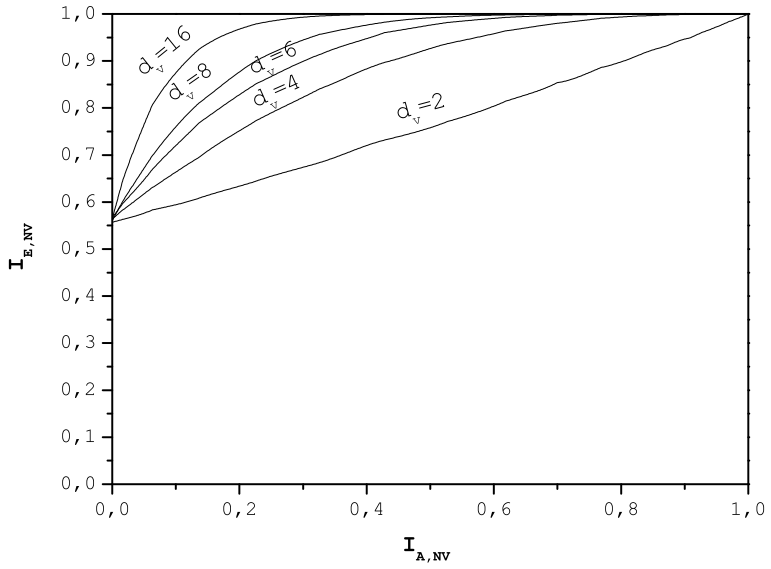


Figura 4.5: Gráfico da informação extrínseca na saída dos nós de variável utilizando aproximação gaussiana e variando a quantidade de ligações nos nós de variável.

as curvas EXIT, como mostrado na Figura 4.6, onde são plotadas as três formas de obteção das curvas.

No caso de se utilizar a distribuição do sinal na saída dos nós, o cálculo de $p_E(\xi|X = x)$ é feito através de medidas de histograma, Figura 4.7. Como, os resultados são muito próximos e é preferível trabalhar com as aproximações.

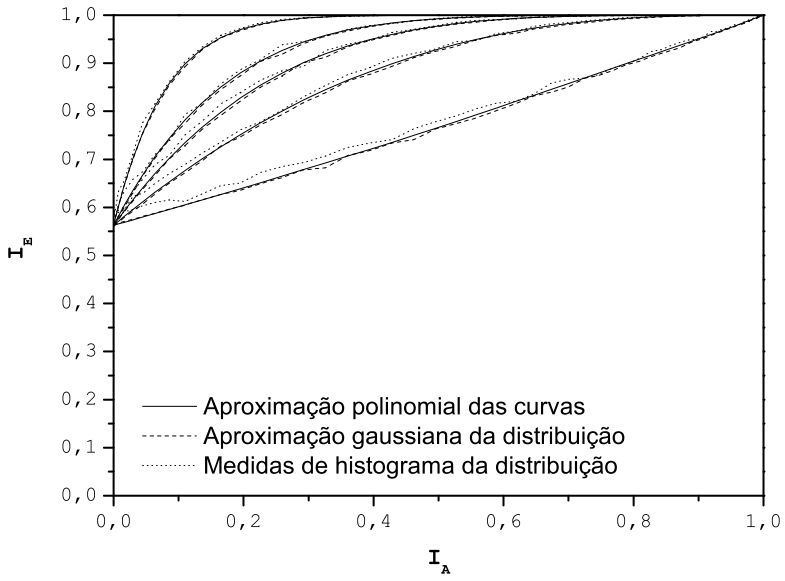


Figura 4.6: Resultado comparativo das curvas de informação extrínseca.

4.4 Curvas EXIT dos nós de verificação

Assim como os nós de variável, os nós de verificação também são considerados como um único decodificador no processo de obtenção das curvas EXIT. As mensagens que passam pelos nós de verificação são diferentes das mensagens que passam pelos nós de variável e a regra que rege essas mensagens é mostrada pela equação

$$E_{NP_i} = \ln \frac{1 - \prod_{j \neq i} \frac{1-e^{A_{NP_j}}}{1+e^{A_{NP_j}}}}{1 + \prod_{j \neq i} \frac{1-e^{A_{NP_j}}}{1+e^{A_{NP_j}}}}. \tag{4.15}$$

Novamente A_{NP_j} pode ser aproximada como a saída de um canal gaussiano, isso por que essas mensagens são as mesmas mensagens que saem do decodificador A (Figura 4.2), em que a saída é uma soma de variáveis aleatórias gaussianas. Apesar das mensagems E_{NP} de saída

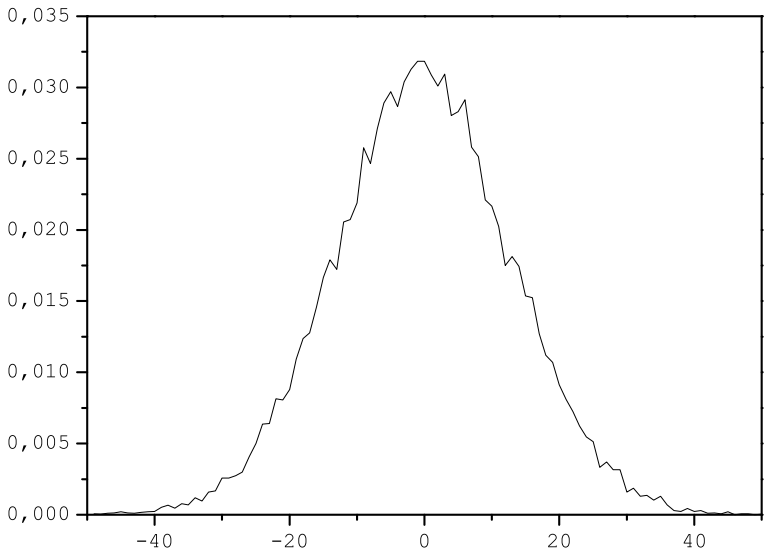


Figura 4.7: Histograma da saída dos nós de variável.

dos nós de verificação não se parecerem com variáveis aleatórias gaussianas, como mostrado na Figura 4.8, a utilização da função gaussiana para descrever essa variável aleatória pode ser utilizada como aproximação [24] [22] e [23].

Com as equações 4.13 e 4.15, pode-se analisar o comportamento dos nós do grafo-fator do código LDPC como duas caixas onde não se tem preocupação sobre as operações realizadas em seu interior. Após a obtenção das distribuições de probabilidade de entrada e de saída de cada nó, efetua-se o cálculo das informações mútuas na entrada (*intrínsecas*) e na saída (*extrínsecas*) de ambos os tipos de nós.

Portanto, uma vez que tenhamos as distribuições de saída dos nós de verificação, ainda é preciso calcular a informação mútua na saída desse decodificador, obtendo $I_{E_{NP}}$. Assim conseguimos calcular as curvas EXIT do segundo decodificador.

Como, apesar da distribuição de mensagens na saída dos nós de verificação não ser uma gaussiana, utiliza-se uma aproximação gaussiana da variável aleatória E_{NP} de saída dos nós de verificação, podemos utilizar novamente a função $J(\cdot)$ mostrada no Anexo A. Existe uma propriedade de dualidade para o canal BEC (do inglês, *Binary Erase Channel*), mostrada

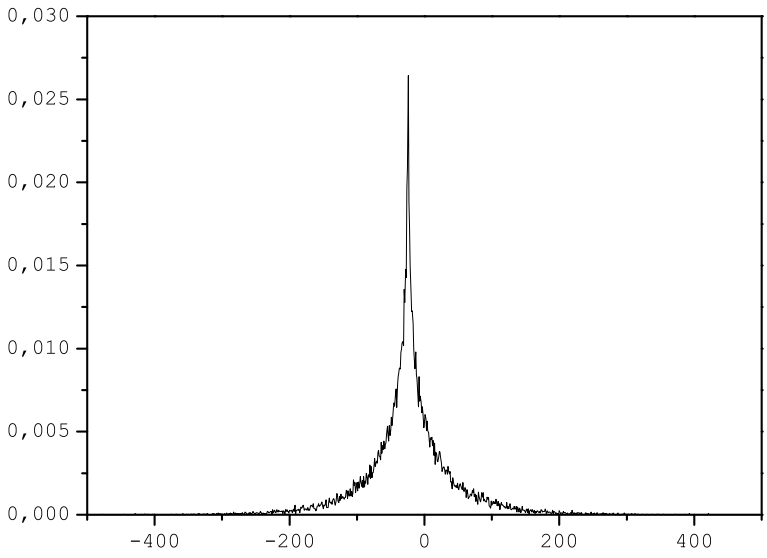


Figura 4.8: Histograma da saída dos nós de verificação de paridade.

em [21], que diz que a curva I_E para um código de grau w_c pode ser função da curva $I_{E_{REP}}$ de um código de repetição de tamanho w_c (ou seja, taxa $1/w_c$), isto é,

$$I_E(I_A, w_c) = 1 - I_{E,REP}(1 - I_a, w_c) \tag{4.16}$$

A curva $I_{E,NP}$ é então obtida utilizando-se as aproximações

$$\begin{aligned} I_{E_{NP}}(I_A, w_c) &\approx 1 - I_{E_{REP}}(1 - I_a, w_c) && (4.17) \\ &= 1 - J\left(\sqrt{w_c - 1} \cdot J^{-1}(1 - I_A)\right) && (4.18) \end{aligned}$$

e

$$I_{A_NP} \approx 1 - J\left(\frac{J^{-1}(1 - I_{E_{NP}})}{\sqrt{w_c - 1}}\right). \tag{4.19}$$

Essa aproximação é bastante confiável no que diz respeito aos resultados, e dado a facilidade de se obter $I_{E_{NP}}$, é mais utilizada. A Figura 4.9 mostra as curvas EXIT para o decodificador referente aos nós de verificação, variando o grau dos nós. Note que quanto maior o grau dos nós as funções apresentam maior curvatura.

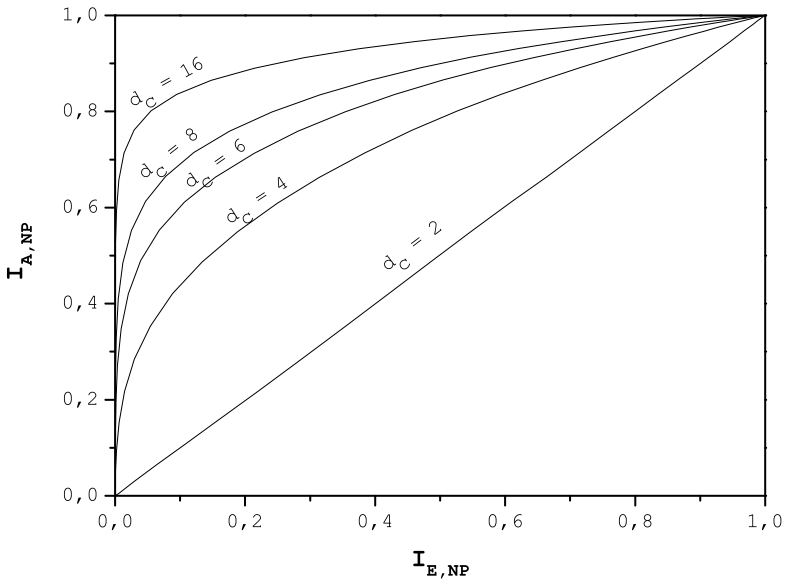


Figura 4.9: Curvas EXIT para o decodificador referente aos nós de verificação com variação do grau dos nós em $w_c = 2, 4, 6, 8$ e 16.

4.5 Curvas EXIT para códigos LDPC irregulares

Para os códigos irregulares as curvas são obtidas das mesmas formas que para códigos regulares no caso de se obter as distribuições de saída, tanto dos nós de variável quanto dos nós de verificação. Entretanto as ligações feitas entre os nós será ditada de acordo com a matriz de paridade **H** do código e os polinômios de grau do código. Novamente utilizamos as equações 4.14 e 4.18, substituindo w_c e w_r por d_c e d_v respectivamente, obtendo as equações

$$I_{E_{NV}}\left(I_{A_{NV}}, d_v, \frac{E_b}{N_0}, R\right) = J\left(\sqrt{(d_v - 1)[J^{-1}(I_{A_{NV}})]^2 + \sigma_Z^2}\right) \tag{4.20}$$

e

$$I_{E_{NP}}(I_A, d_c) = 1 - J\left(\sqrt{d_c - 1} \cdot J^{-1}(1 - I_A)\right) \tag{4.21}$$

e, assim como para os códigos regulares, o mapeamento da função $I_{A_{NP}}$ é feito utilizando a equação:

$$I_{A_{NP}} \approx 1 - J\left(\frac{J^{-1}(1 - I_{E_{NP}})}{\sqrt{d_c - 1}}\right). \tag{4.22}$$

4.6 Projeto de Códigos LDPC utilizando Curvas EXIT

As funções EXIT fornecem informações muito importantes para o projeto de códigos sem que seja preciso simular um sistema de comunicação para obter as curvas de probabilidade de erro. Entretanto, dois pontos importantes precisam ser abordados quanto ao projeto de códigos utilizando curvas EXIT: o ponto-de-queda e a complexidade dos códigos LDPC. O ponto-de-queda é o valor E_b/N_0 onde acontece a queda brusca na probabilidade de erro de bit. Este ponto também é conhecido como *water-fall*. Nas curvas EXIT esse ponto é o valor de E_b/N_0 exatamente no limiar onde as curvas se encontram (Figura 4.10).

O ponto-de-queda é obtido variando o valor de E_b/N_0. Assim as curvas para o decodificador A, que é o decodificador referente aos nós de variável e onde chega o sinal que sai do canal, sobem ou descem no gráfico como mostra a Figura 4.11. A relação sinal ruído influencia apenas nas curvas EXIT dos nós de variável, pois a relação é argumento da equação de informação mútua para os nós de variável (equação 4.20).O ponto-de-queda pode variar também de acordo com a complexidade ou com a taxa do código. Por exemplo, aumentar o peso w_r dos nós de variável faz com que a curva EXIT para esses nós suba mais rapidamente (Figura 4.5), e possibilitando diminuir a razão sinal ruido.

Por esse motivo as curvas EXIT dos nós de verificação sempre saem de zero, já que no momento em que o receptor recebe a saída do canal para iniciar o processo de decodificação a informação que os nós de verificação têm sobre o sinal é nenhuma.

Diz-se que um código LDPC é mais ou menos complexo quando nos referimos a quantidade de 1's presentes em sua matriz de verificação de paridade, ou seja, quanto mais esparsa for a matriz de verificação de paridade, menos complexa será sua decodificação.

Além da complexidade de informação, a esparsidade da matriz interfere no ponto-de-queda do código. Quanto mais esparsa for a matriz, mais baixo será o ponto-de-queda. O que fortalece a afirmação de que quanto mais esparsa for a matriz, melhor será o código. A Figura 4.12 mostra a variação dos valores de queda para códigos LDPC regulares de taxa $r = 0,5$, variando-se apenas os valores de w_r e w_c, ou seja, a complexidade do código.

As funções EXIT não são ferramentas de criação de código, entretanto elas ajudam a entender o comportamento da decodificação iterativa e fornece a informação de que existe

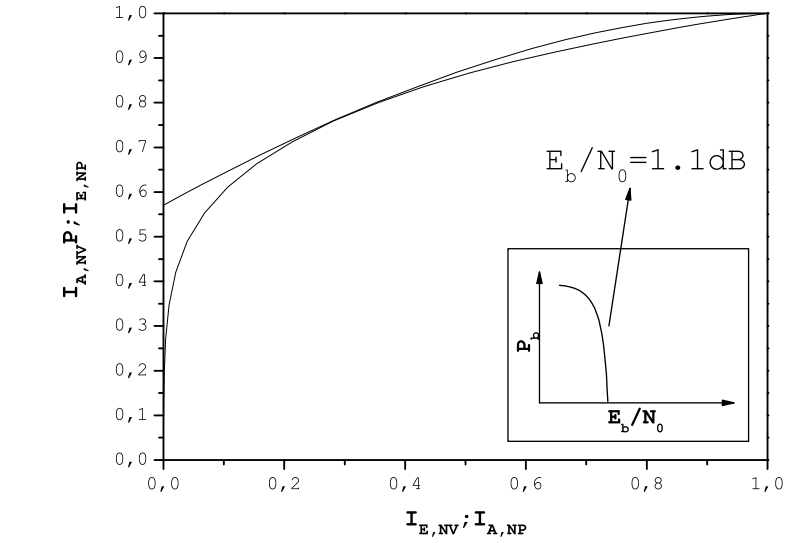


Figura 4.10: Exemplo de obtenção do ponto-de-queda utilizando as curvas EXIT.

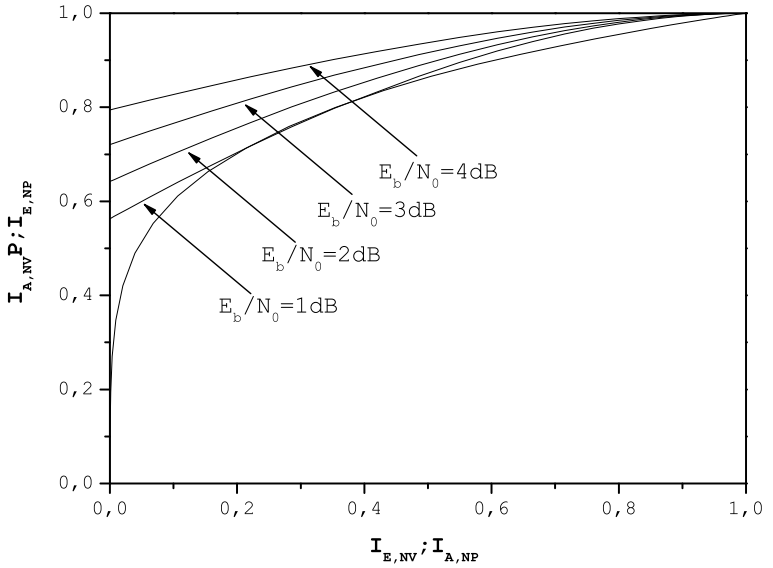


Figura 4.11: Variação das curvas EXIT pela relação sinal/ruído.

um código com os parâmetros w_r e w_c de tamanho infinito que consegue chegar a esses limiares de decodificação. Para códigos de tamanhos muito longos ela é uma ferramenta que ajuda muito a escolha de valores dos parâmetros dos códigos a serem utilizados, já que nos

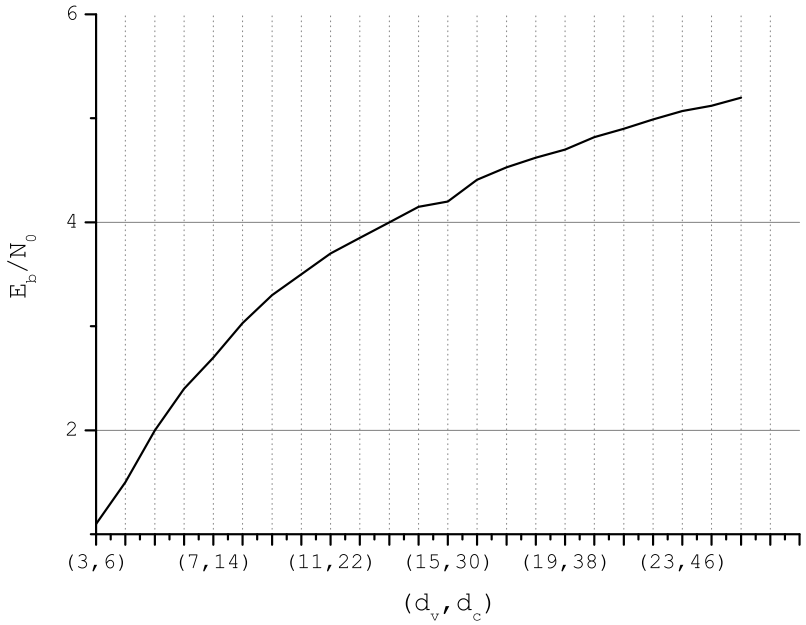


Figura 4.12: Medidas de pontos-de-queda para códigos de taxa r=0,5 e de complexidades diferentes.

fornece o limiar de convergência do algoritmo sem que seja preciso simular o sistema.

Para o caso dos códigos LDPC irregulares, as curvas EXIT auxiliam ainda no processo do projeto de códigos de uma maneira efetiva, no que diz respeito a distribuição de pesos ρ_d e λ_d dos graus das linhas e das colunas nas equações 3.13 e 3.14. Esse é um processo de otimização, onde se deve encontrar a melhor distribuição de pesos de acordo com as curvas EXIT consideradas ótimas.

Capítulo 5

Funções EXIT para Modulação codificada

Existem diversas técnicas para se combinar codificação e modulação. Uma destas técnicas consiste na concatenação serial de um codificador binário, um entrelaçador de bits e um modulador q-ário ($q > 2$) [25] [26]. Esta técnica é conhecida como BICM (do inglês, Bit Interleaved Coded Modulation). Recentemente, muita atenção foi dada a BICM onde o código binário é um código LDPC [21] [27]. Vários aspectos explicam esta atenção. Como já citado anteriormente os códigos LDPC permitem uma decodificação iterativa eficiente. Além desta qualidade, quando combinados com modulação, a decodificação iterativa pode ser aplicada de maneira conjunta com a demodulação. E também, dada a natureza esparsa e aleatória das suas matrizes de paridade, os códigos LDPC não precisam do entrelaçador para se obter um bom desempenho (baixas probabilidades de erro).

Neste capítulo utilizaremos funções EXIT para analisar um algoritmo iterativo para demodulação e decodificação conjuntas. Desta análise, será possível encontrar os parâmetros de um código LDPC que melhor se adequam ao canal e ao esquema de codificação combinado com modulação .

5.1 Curvas EXIT para receptores iterativos

A Figura 5.1 mostra um modelo em blocos do transmissor. Após os bits de fonte $\{x_k\}$ serem codificados gerando bits $\{c_i\}$, eles são então mapeados em formas de onda na constelação, gerando $\{s_j\}$ e são enviados ao canal.

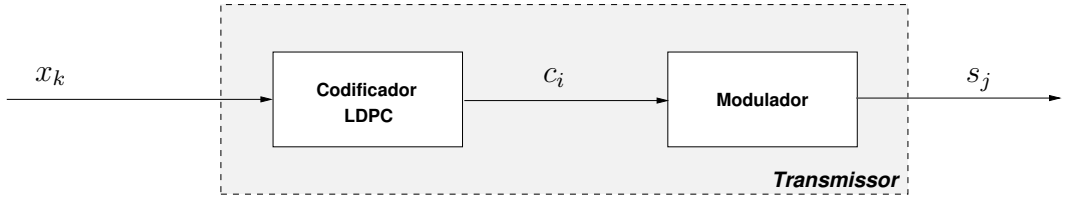


Figura 5.1: Diagrama em blocos do transmissor.

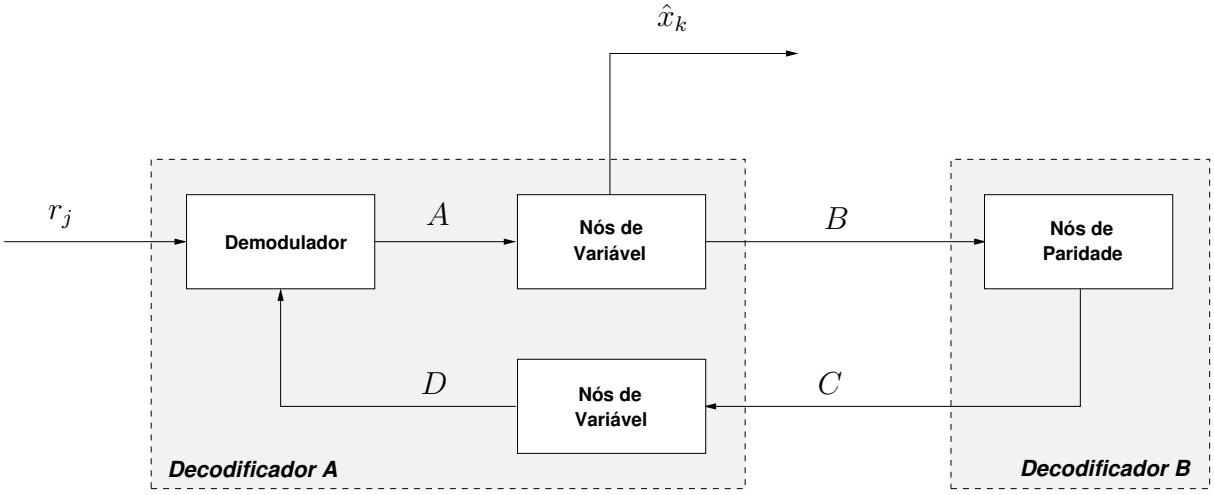


Figura 5.2: Diagrama em blocos do receptor.

No canal, os símbolos $\{s_j\}$ sofrem interferência e ruído, e chegam ao receptor na forma $\{r_j\}$. A Figura 5.2 mostra o diagrama de blocos do receptor. Os símbolos $\{r_j\}$ chegam ao bloco demodulador e são demodulados. Nesse ponto, eles são mensagens que descrevem uma probabilidade do bit de informação ser reconhecido como 0 ou 1.

Depois de demodulados, os sinais são passados aos outros blocos de acordo com a Figura 5.2. As letras A, B, C e D se referem às mensagens que saem de cada bloco. Os blocos *nós de variável* e *nós de paridade* se referem aos blocos do grafo-fator relativos ao código LDPC. Os retângulos cinzas, chamados de Decodificadores A e B, são os retângulos aos quais as curvas EXIT se referem, as mensagens B e C são as mensagens que se referem as informações mútuas das curvas.

Antes de traçar as curvas EXIT para esse decodificador é preciso entender as passagens de mensagens entre os blocos decodificadores da Figura 5.2. O algoritmo de passagem de mensagens segue algumas regras referentes a cada bloco e de acordo com o seguinte processo de decodificação:

- O primeiro passo na decodificação é referente ao bloco demodulador. Os símbolos r_j chegam ao demodulador e com as mensagems a priori D obtém as mensagens A, que são as mensagens que entram no próximo bloco, o de nós de variável;
- As mensagens que chegam aos nós de variável são as mensagens que saem do bloco demodulador A. Após isso, os nós de variável repassam as mensagens B ao bloco de verificação de paridade. Em uma situação de decodificação real, é aqui que os bits passam para a estimação da palavra código;
- Ao bloco de verificação de paridade chegam d_v mensagens, e após as operações serem executadas sobre essas mensagens, elas são novamente passadas ao bloco de nós de variável. Essas são as mensagens C;
- Os nós de variável então recebem as mensagens C, operam sobre essas mensagens e repassam novas mensagens D ao bloco demodulador. Na Figura 5.2, é feita essa distinção entre os dois nós de variável pois as mensagens que são passadas são bem diferentes. No bloco inferior de nós de variável, as d_c mensagens que vêm dos nós de paridade são somadas e passadas ao demodulador, enquanto no superior as mensagens seguem a regra da Equação 4.13.

As informações mútuas na entrada e na saída do decodificador B, I_B e I_C, respectivamente, são calculadas da mesma forma que no Capítulo 4, pois o decodificador B continua sendo os nós de paridade. Entretanto, para se obter a curva EXIT do decodificador A é preciso uma série de operações diferentes das do Capítulo 4.

No decodificador B as mensagens de entrada podem ser consideradas como variáveis aleatórias gaussianas, já que elas são referentes à saída do decodificador A e podem, portanto, ser aproximadas por gaussianas.

5.1.1 Mensagens do demodulador

É necessário que um tratamento especial seja dado a essas mensagens, porque dessa forma pode-se generalizar os gráficos EXIT para qualquer outra modulação combinada aos nós de variável. Ou seja, com essa abordagem o demodulador pode ser um grafo fator

de um demodulador M-APSK, que é o caso apresentado nesse trabalho, ou qualquer outro demodulador. As mensagens D são a soma das mensagens provenientes dos nós de verificação de paridade ligados a cada nó de variável do bloco de nós de variável, portanto, através de uma pequena modificação na Equação 4.13, mostramos que a mensagem D é

$$D = \sum_{j=1}^{d_c} C_j. \tag{5.1}$$

Novamente as mensagens C que saem do decodificador B (que são os nós de verificação de paridade) são aproximadas por variáveis aleatórias gaussianas, podemos portanto, utilizar novamente a aproximação $J(\cdot)$ para o cálculo das curvas EXIT [27]. Isto é,

$$I_D = J\left(\sqrt{i} \cdot J^{-1}(I_C)\right). \tag{5.2}$$

Para cada sistema de comunicação, é possível que exista um tipo de modulador/demodulador, e, consequentemente, as mensagens D são variáveis aleatórias em que suas distribuições de probabilidade tem características que dependem do demodulador, e, por isso, dependem também do tipo de modulação empregada. Portanto, apesar de em [21] ser feito uma aproximação para as funções EXIT para tipos específicos de demoduladores, nesse trabalho preferimos obter as distribuições de probabilidade na saída do demodulador através de simulação. Já que dessa forma, é possível generalizar para diferentes demoduladores.

As distribuições de probabilidade na saída do demodulador foram obtidas considerando o fato de as mensagens D assumirem comportamento de uma variável aleatória gaussiana, o que é esperado, já que, de acordo com a Equação 5.1, a mensagem D é uma soma de variáveis aleatórias gaussianas. Portanto, para facilitar, é dito que a variância de D na entrada do decodificador é medida como função J inversa, ou seja: $\sigma = J^{-1}(I_D)$.

5.1.2 Curvas EXIT para o decodificador A

As curvas EXIT são funções das informações mútuas variando σ no sinal de informação intrínseca no bloco decodificador e simulando as distribuições de probabilidade das mensagens nas iterações do decodificador. No caso do decodificador A não é diferente, as mensagens

D são enviadas ao demodulador e a distribuição de probabilidade da mensagem A é obtida através de medidas de histograma. Após isso, calcula-se a informação mútua (Equação 4.11) para cada valor de σ da variável de entrada D.

Dizemos que B é uma variável aleatória gaussiana, assim como a variável aleatória A. Portanto, é preciso simular apenas o demodulador, para que se obtenha as distribuições de probabilidade na sua saída [21]. Com as distribuições de probabilidade da saída do demodulador, podemos calcular a informação mútua $I_A(\sigma_{I_D})$, já que I_D é conhecido (equação 5.2). As funções EXIT dos nós de variável já são conhecidas, portanto, através de um mapeamento conseguimos as curvas EXIT do deodulador A, e a informação mútua I_B é calculada por:

$$I_B\left(I_C, d_v, \frac{E_b}{N_0}, R\right) = J\left(\sqrt{(d_v-1)[J^{-1}(I_C)]^2 + [J^{-1}(I_A)]^2}\right). \tag{5.3}$$

Note que a Equação 5.3 é obtida substituindo σ_Z por $J^{-1}(I_A)$ em 4.16, como em [27]. A função EXIT do decodificador A é aproximada pela função $J(\cdot)$ porque as mensagens B ainda saem dos nós de variável, e para números muito grandes de amostras, a aproximação gaussiana é razoável.

5.1.3 Curvas EXIT para o decodificador B

Nas curvas EXIT para códigos LDPC, considera-se os nós de variável e de paridade como decodificadores, como mostrado na Figura 5.2. O fato de inserirmos o demodulador na análise das curvas EXIT, interfere apenas nas curvas dos nós de variável, já que é onde o demodulador está diretamente ligado.

Portanto, as operações realizadas pelo decodificador B são as mesmas operações regidas pela Equação 4.15. Dessa forma, podemos obter as curvas EXIT utilizando as equações 4.18 e 4.19, reescritas como

$$I_C(I_A, d_c) \approx 1 - J\left(\sqrt{d_c - 1}\cdot J^{-1}(1-I_A)\right) \tag{5.4}$$

e

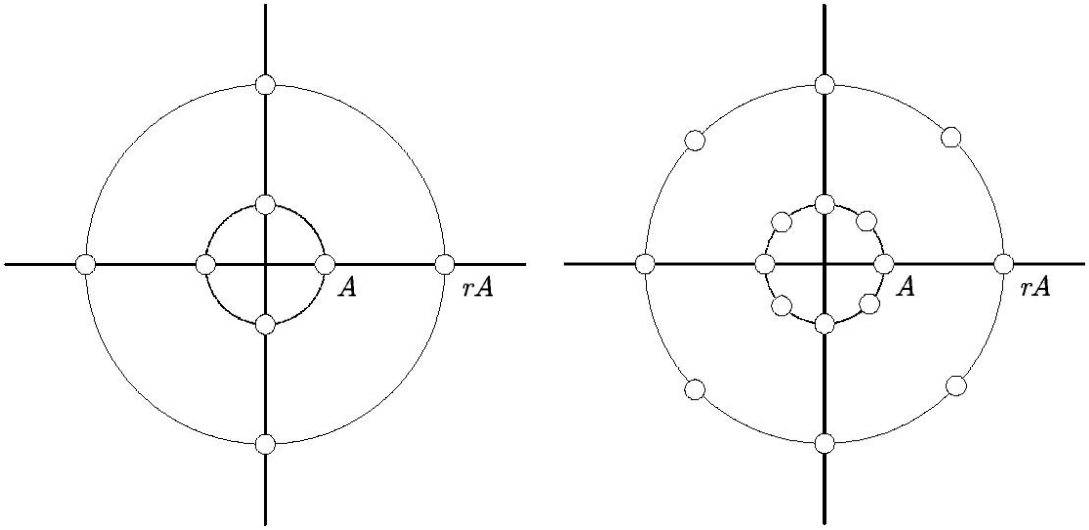
$$I_B \approx 1 - J\left(\frac{J^{-1}(1-I_C)}{\sqrt{d_c-1}}\right), \tag{5.5}$$

onde I_C é a informação mútua da variável intrínseca e I_B a informação mútua da variável

extrínsseca. Como mostrado no Capítulo 4, a aproximação gaussiana das distribuições dos sinais de entrada e saída para esses nós é possível, e portanto utiliza-se a função $J(\cdot)$.

5.2 Canal M-APSK/AWGN não-coerente de bloco

Para esse estudo de caso, foi utilizada a modulação 8-APSK, que é uma modulação M-APSK de oito símbolos. A modulação M-APSK é uma denominação formada por constelações APSK compostas de N anéis de amplitudes diferentes, e cada amplitude contendo P valores de fase alinhados. Os raios dos anéis diferem por um fator constante r denominado *fator de raio*. As constelações M-APSK são caracterizadas pelas constantes N e P,



(a) 8-APSK. (b) 16-APSK.

Figura 5.3: Diagramas das constelações APSK com dois níveis de amplitude A e rA.

onde $M = NP$, portanto, no nosso exemplo, a constelação 8-APSK tem $N = 2$ e $P = 4$ (Figura 5.3a). A constelação 16-APSK da Figura 5.3b, que tem $N = 2$ e $P = 8$.

A configuração 8-APSK da Figura 5.3a foi escolhida para esse trabalho pois é a mesma utilizada em [8], portanto para efeitos de comparação essa é a escolha natural.

5.2.1 Modelo do Canal e Modulador

O canal M-APSK/AWGN não coerente de bloco é definido através da equação 5.6, e recebe o nome M-APSK pois os símbolos s_l da constelação são símbolos APSK definidos

anteriormente. A saída $\mathbf{r} = (r_1, \ldots, r_L)$ é da forma:

$$r_l = s_l e^{j\theta} + n_l, \qquad l = 1, \ldots, L, \tag{5.6}$$

onde L é o tamanho do bloco, s_l são os símbolos da saída do modulador e n_l é um ruído gaussiano i.i.d. Uma fase aleatória θ é introduzida pelo canal, e é representada por uma variável aleatória contínua com distribuição uniforme entre $[0, 2\pi)$. A fase é aleatória, mas constante em cada bloco de L símbolos enviados. O transmissor desse sistema é mostrado na Figura 5.4 através de diagrama de blocos.

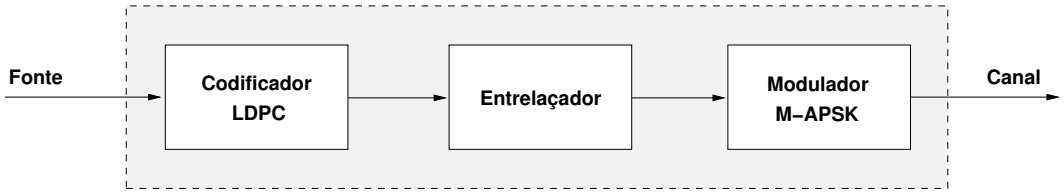
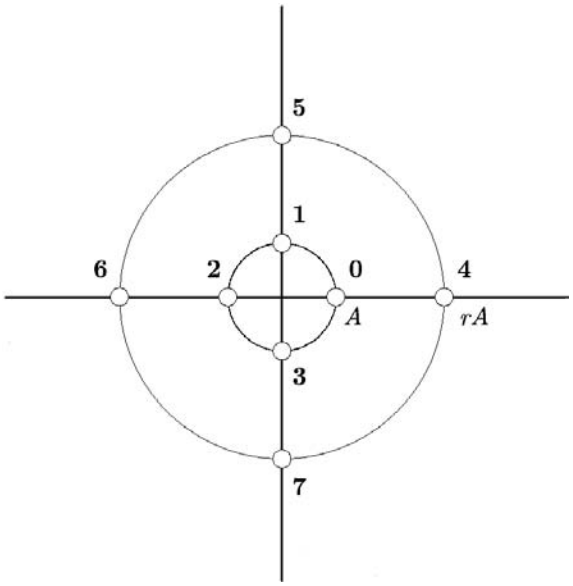
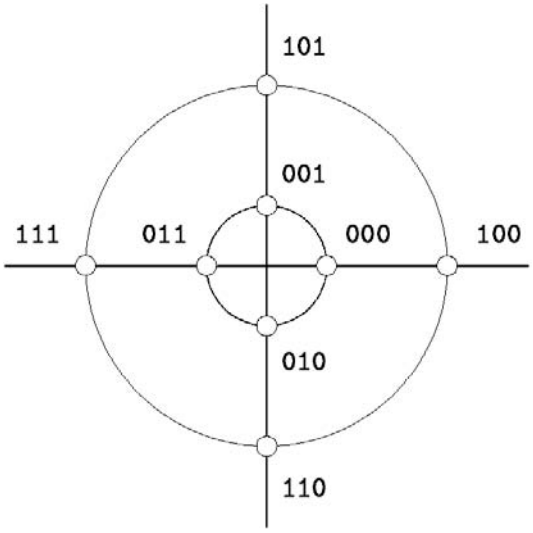


Figura 5.4: Diagrama de bloco do transmissor M-APSK.



(a) Rotulamento de símbolos para a constelação M-APSK.



(b) Mapeamento de símbolos para a constelação M-APSK.

Figura 5.5

O bloco modulador recebe a saída de bits do bloco codificador, onde acontece a codificação LDPC dos bits de informação, e inicia o mapeamento de bits em símbolos da constelação.

Como mostrado na Figura 5.5b, cada conjunto de 3 bits é mapeado em um símbolo s_l da constelação. O bloco entrelaçador da Figura 5.4 não influencia na análise através das funções EXIT, isto porque, apesar do entrelaçador melhorar o desempenho em sistemas codificados, ele apenas modifica a matriz de verificação de paridade do código utilizado no sistema.

Na Figura 5.5b é mostrado como é feito o mapeamento do conjunto de bits em símbolos da modulação. O anel interno da modulação se refere às sequências que têm 0 como o bit mais significativo (b_1), ao passo que as sequências de bits que iniciam com 1 fazem parte do anel externo. A fase também é mapeada de acordo com as sequências, os dois bits seguintes ao bit mais significativo (b_2 e b_3) indicam qual é a fase do símbolo no qual a sequência completa deve ser mapeada, de acordo com a tabela abaixo:

b_2b_3	ϕ
00	0
01	$\pi/2$
11	π
10	$3\pi/2$

A função de mapeamento dos bits codificados em símbolos da modulação, rotulados na Figura 5.5a, é descrita através da tabela:

$b_1b_2b_3$	s
000	0
001	1
011	2
010	3
100	4
101	5
111	6
110	7

Além disso, utilizamos uma modulação diferencial, assim como em [8], isso implica dizer que os símbolos enviados não são exatamente os símbolos mapeados, ao invés disso, é enviado um símbolo s'_i, onde:

$$s'_i = s_{i-1} \oplus_M s'_{i-1}, \tag{5.7}$$

na qual s'_{i-1} é o símbolo diferencial anterior, s_{i-1} é o símbolo proveniente do mapeador M-APSK e o operador $\oplus_M$ representa a soma módulo M.

Exemplo:

Digamos que o símbolo $s'_{i-1} = 001$ é o símbolo que foi enviado anteriormente pelo canal e agora o modulador recebe o símbolo $s_{i-1} = 010$. Seguindo a operação descrita pela equação 5.7 o símbolo enviado pelo canal será o símbolo $s'_i = 100$, pois

$$100 = 010 \oplus_M 001. \tag{5.8}$$

No nosso caso de estudo em particular, cada bloco de tamanho L (que é o tamanho do bloco do canal) tem um símbolo de referência. Além disso, cada bloco é independente do bloco anterior, ou seja, o primeiro símbolo de um bloco não depende do ultimo símbolo do bloco anterior, e cada bloco tem um símbolo de referência. O símbolo de referencia utilizado é o símbolo rotulado como 000, e é o primeiro símbolo de cada bloco enviado de tamanho L.

Em outras palavras, o transmissor codifica os bits de informação através de um codificador LDPC, após isto, os bits codificados passam por um processo de entrelaçamento. O processo de entrelaçamento modifica a matriz geradora do código diminuindo a probabilidade de erro do sistema [8]. Depois do entrelaçamento dos bits codificados, o transmissor faz o mapeamento de sequências de bits codificados em sinais que são símbolos da modulação que são enviados pelo canal.

5.2.2 Receptor para o canal M-APSK/AWGN não coerente de bloco e suas mensagens

O grafo fator do receptor iterativo por sua vez, é mostrado na Figura 5.6 e é descrito em detalhes em [8]. Na Figura 5.6 fica clara a divisão do bloco demodulador do bloco decodificador. O bloco demodulador é mostrado em separado graças ao caso particular em que estamos trabalhando.

Além disso, dividimos ainda o bloco demodulador pois a análise utilizando funções EXIT é feita independente do código que será utilizado efetivamente no sistema. Portanto é razoável utilizar somente o que chamamos de *bloco demodulador* (Figura 5.7) para uma análise em camadas. Esses blocos são blocos de tamanho L, e no caso da simulação de um sistema de comunicação dessa natureza, a matriz de paridade do código LDPC deve ser de tamanho $n_c = N_b\, m_b\, [L-1]$, onde N_b é a quantidade de blocos demoduladores para cada palavra de

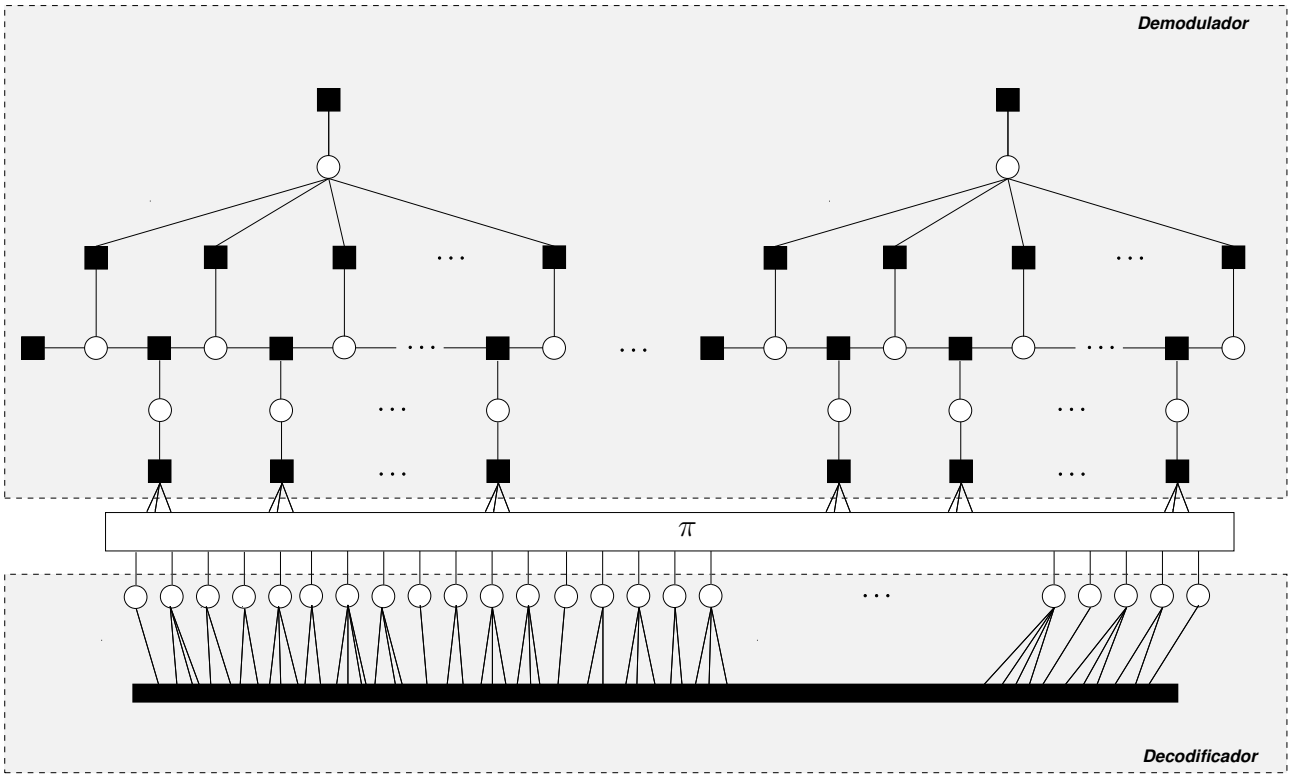


Figura 5.6: Divisão em blocos do receptor, diferenciando o bloco demodulador do bloco decodificador.

tamanho n_c, e m_b é o número de bits necessários para o mapeamento da modulação. Por exemplo no caso de a modulação ter 8 símbolos, $m_b = 3$ pois $2^3 = 8$.

Portanto considera-se o demodulador apenas um único bloco de demodulação ao invés dos N_b blocos necessários para formar a palavra código de tamanho n_c. A Figura 5.7 mostra o grafo de apenas um bloco de demodulação.

As mensagens repassadas entre os nós são definidas como:

▷ **Nós θ:**

As mensagens $\mu_{\theta_k \to T_i}(\theta_k)$ são estimações da fase aleatória do canal no bloco k. Essas mensagens são calculadas para cada símbolo r_i recebido:

$$\mu_{\theta_k \to T_i}(\theta_k) = \mu_{\Pi_k \to \theta_k}(\theta_k) \cdot \prod_{\substack{j=(k-1)L+1 \\ j \neq i}}^{kL} \mu_{T_j \to \theta_k}(\theta_k); \tag{5.9}$$

▷ **Nós T_i:**

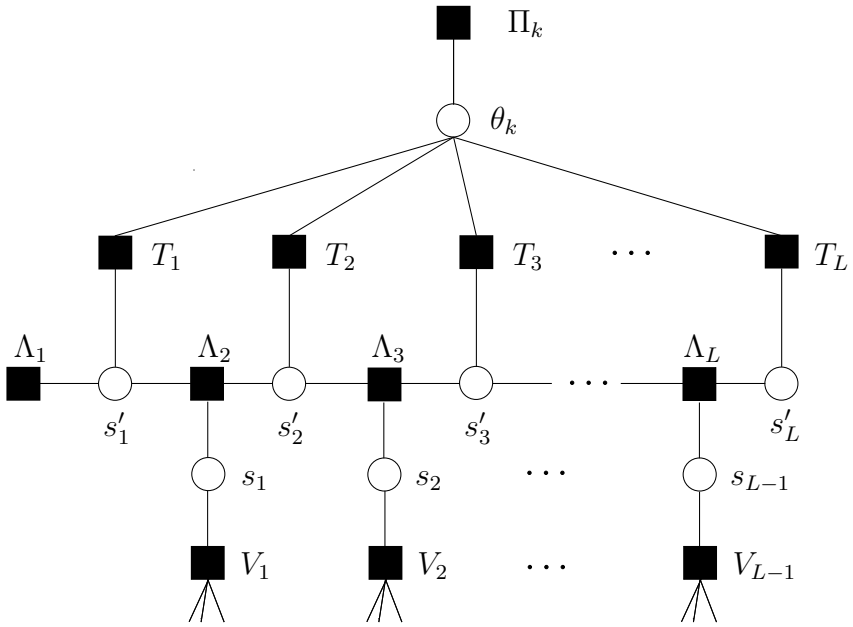


Figura 5.7: Bloco demodulador para o canal M-APSK/AWGN não coerente de bloco. O índice L indica o tamanho do bloco de memória do canal.

$$\mu_{T_i\to\theta_k}(\theta_k) = \sum_{s'_i} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{||r_i - s'_i\exp(j\theta_k)||^2}{2\sigma^2}\right)\cdot \mu_{s'_i\leftarrow T_i}(s'_i); \quad (5.10)$$

$$\mu_{T_i\to s'_i}(s'_i) = \sum_{z=1}^{Z} a_z \exp\left(-\frac{||r_i - s'_i\exp(j\hat{\theta}_{kz})||^2}{2\sigma^2}\right); \quad (5.11)$$

onde $a_z = \mu_{\theta_k\to T_i}(\theta_k = \hat{\theta}_{kz})$ que é a mensagem $\mu_{\theta_k\to T_i}(\theta_k)$ quantizada em Z níveis. Essa mensagem é quantizada apenas para efeito de computação. A mensagem original

$$\mu_{T_i\to\theta_k}(\theta_k) = \int_0^{2\pi} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{||r_i - s'_i\exp(j\hat{\theta}_k)||^2}{2\sigma^2}\right)\cdot \mu_{\theta_k\to T_i}(\theta_k)d\theta_k \quad (5.12)$$

envolve um operador de integração, o que eleva o número de operações realizadas no receptor e, ceonsequentemente a velocidade na decodificação. Como a quantização dessa mensagem reduz substancialmente o número de operações realizadas pelo computador, preferimos utilizá-la.

▷ **Nós Π_k:**

$$\mu_{\Pi_k \to \theta_k}(\theta_k) = \frac{1}{Z}; \tag{5.13}$$

Essas mensagens são importantes para efeito de normalização da aproximação da mensagem $\mu_{T_i \to s'_i}(s'_i)$, que é uma versão quantizada da mensagem original calculada através de uma integral.

▷ **Nós s'_i e Λ_i:**

Esses nós merecem atenção no demodulador, pois eles fazem a decodificação do codificador diferencial inserido no transmissor, que é feita em duas etapas através do algoritmo *forward-backward*.

Parte *forward*:

$$\mu_{s'_i \to \Lambda_{i+1}}(s'_i) = \mu_{\Lambda_i \to s'_i}(s'_i) \cdot \mu_{T_i \to s'_i}(s'_i) \tag{5.14}$$

$$\mu_{\Lambda_i \to s'_i}(s'_i) = \sum_{\sim\{s'_i\}} I\{s'_i = s_{i-1} \oplus_M s'_{i-1}\} \cdot \mu_{s'_{i-1} \to \Lambda_i}(s'_{i-1}) \cdot \mu_{s_{i-1} \to \Lambda_i}(s_{i-1}) \tag{5.15}$$

Parte *backward*:

$$\mu_{s'_i \to T_i}(s'_i) = \mu_{\Lambda_{i+1} \to s'_i}(s'_i) \cdot \mu_{\Lambda_i \to s'_i}(s'_i) \tag{5.16}$$

$$\mu_{s'_i \to \Lambda_i}(s'_i) = \mu_{T_i \to s'_i}(s'_i) \cdot \mu_{\Lambda_{i+1} \to s'_i}(s'_i) \tag{5.17}$$

Chegando finalmente às mensagens

$$\mu_{\Lambda_i \to s'_{i-1}}(s'_{i-1}) = \sum_{\sim\{s'_{i-1}\}} I\{s'_i = s_{i-1} \oplus_M s'_{i-1}\} \cdot \mu_{s'_i \to \Lambda_i}(s'_i) \cdot \mu_{s_{i-1} \to \Lambda_i}(s_{i-1}) \tag{5.18}$$

$$\mu_{\Lambda_i \to s_{i-1}}(s_{i-1}) = \sum_{\sim\{s_{i-1}\}} I\{s'_i = s_{i-1} \oplus_M s'_{i-1}\} \cdot \mu_{s'_{i-1} \to \Lambda_i}(s'_{i-1}) \cdot \mu_{s'_i \to \Lambda_i}(s'_i) \tag{5.19}$$

Especificamente, tem-se que no inicio de cada bloco as mensagens $\mu_{\Lambda_i \to s'_{i-1}}(s'_{i-1})$ são:

$$\mu_{\Lambda_i \to s'_{i-1}}(s'_{i-1}) = \begin{cases} 1, & s'_i = 0 \\ 0, & s'_i = 1, 2, \ldots M-1 \end{cases} \tag{5.20}$$

já que é conhecido que o primeiro símbolo de cada bloco k é o símbolo zero. A função $I\{f(x) = y\}$ é chamada de função indicadora, essa função é definida como:

$$I(x) = \begin{cases} 1; & f(x) = y; \\ 0; & \textit{caso contrário.} \end{cases} \tag{5.21}$$

Ou seja, resulta em 1 caso a o peração realizada seja verdadeira, e em 0 caso a operação seja falsa.

▷ **Nós s_i:**

Os nós s_i são nós unitários, ou seja, esses nós apenas repassam as mensagens que chegam neles :

$$\mu_{\Lambda_i \to s_{i-1}}(s'_i) = \mu_{s_{i-1} \to V_{i-1}}(s_i); \tag{5.22}$$

$$\mu_{s_i \to \Lambda_{i+1}}(s_i) = \mu_{V_i \to s_i}(s_i). \tag{5.23}$$

▷ **Nós V_i:**

As mensagens na saída dos nós V_i são as probabilidades de o bit transmitido ser 0 ou 1. Como existe um mapeamento entre os bits na saída do codificador e o modulador, as m mensagens (estimativas de bits) na saída desse nó dependem da posição de cada bit nos sinais da modulação, o que justifica a utilização da função indicadora nessas mensagens

$$\mu_{V_i \to s_i}(s_i) = \sum_{\sim\{s_i\}} I\{s_i = V(b_i)\} \prod_{n=1}^{m} \mu_{b_i^n \to V_i}(b_i^n); \tag{5.24}$$

$$\mu_{V_i \to b_i^\alpha}(b_i^\alpha) = \sum_{\sim\{b_i^\alpha\}} I\{s_i = V(b_i)\} \cdot \mu_{s_i \to V_i}(S_i) \prod_{\substack{n=1 \\ n \neq \alpha}}^{m} \mu_{b_i^n \to V_i}(b_i^n). \tag{5.25}$$

▷ **Nós b_i^α:**

Os nós b_i^α são os nós de variável do código LDPC. As mensagens que são tratadas nesses nós, assim como nos de verificação de paridade, são estimativas das LLR, e portanto quando passadas ao demodulador devem ser passadas no domínio das probabilidades. Isso é feito através das formulas definidas abaixo:

$$\mu_{b_i^\alpha \to V_i}(b_i^\alpha) = \frac{\exp(LLR(b_i^\alpha))}{1 + \exp(LLR(b_i^\alpha))}; \tag{5.26}$$

$$\mu_{b_i^\alpha \to V_i}(b_i^\alpha) = \frac{1}{1 + \exp(LLR(b_i^\alpha))}. \tag{5.27}$$

As variáveis D são as mensagens que saem dos nós de variável e entram no bloco demodulador ($\mu_{b_i^\alpha \to V_i}(b_i^\alpha)$, e $\mu_{b_i^\alpha \to V_i}(b_i^\alpha)$) e as variáveis A são as que saem do demodulador e chegam ao bloco de nós de variável ($\mu_{V_i \to b_i^\alpha}(b_i^\alpha)$).

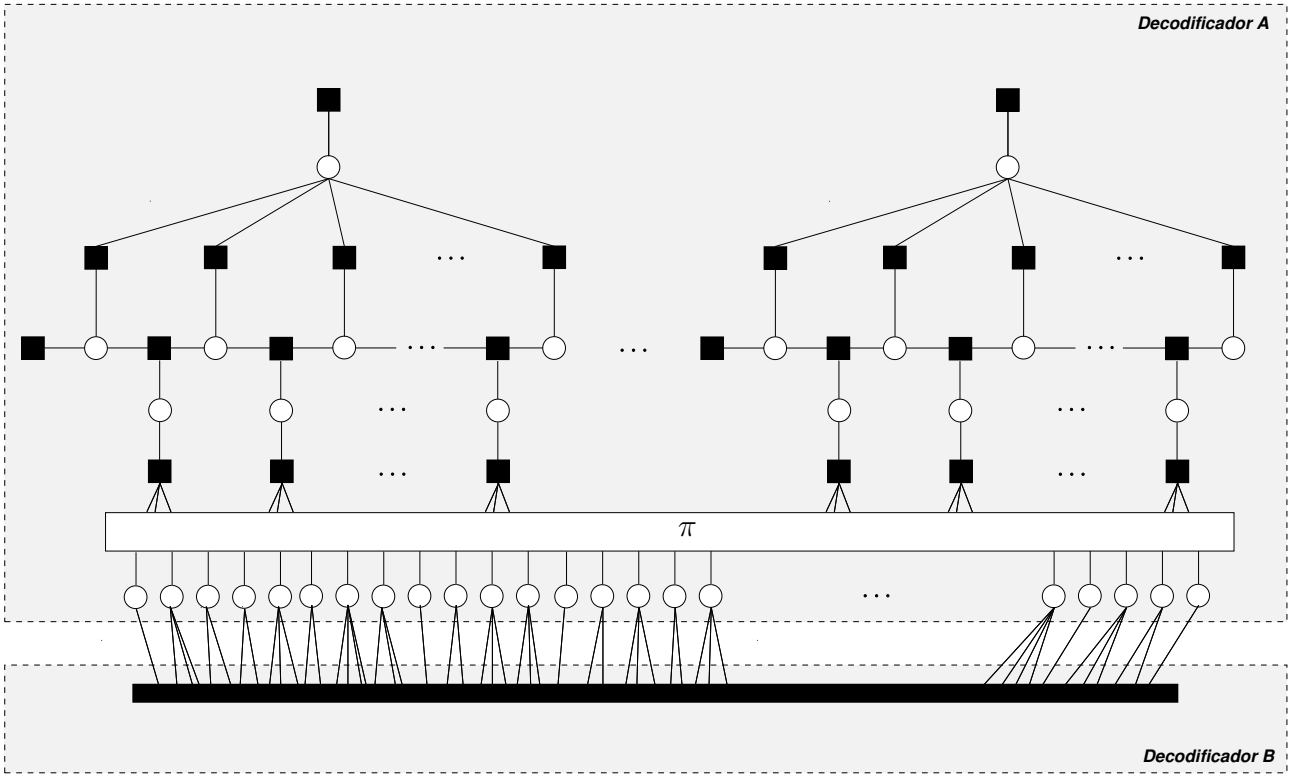


Figura 5.8: Decodificadores em relação às curvas EXIT. A divisão em dois blocos se refere à divisão em decodificadores A e B da Figura 5.2. O decodificador A é formado parte pelos blocos demoduladores e parte pelos nós de variável.

Após essas mensagens serem definidas, temos os instrumentos necessários para obter as curvas EXIT para o demodulador desse sistema. Como mostrado na Figura 5.8, dividimos o receptor em dois blocos decodificadores.

As mensagens A e D do decodificador, necessárias para o cálculo da informação mútua, são as mensagens que saem dos nós de variável e as que saem dos nós de verificação de paridade, respectivamente. As mensagens que saem dos nós de variável b_i^α e vão aos nós de paridade são estimativas de probabilidades no domínio de LLR, e são repassadas seguindo as mensagens A que saem do bloco demodulador.

A Figura 5.9 mostra as variáveis aleatórias A, B, C e D como as mensagens que passam

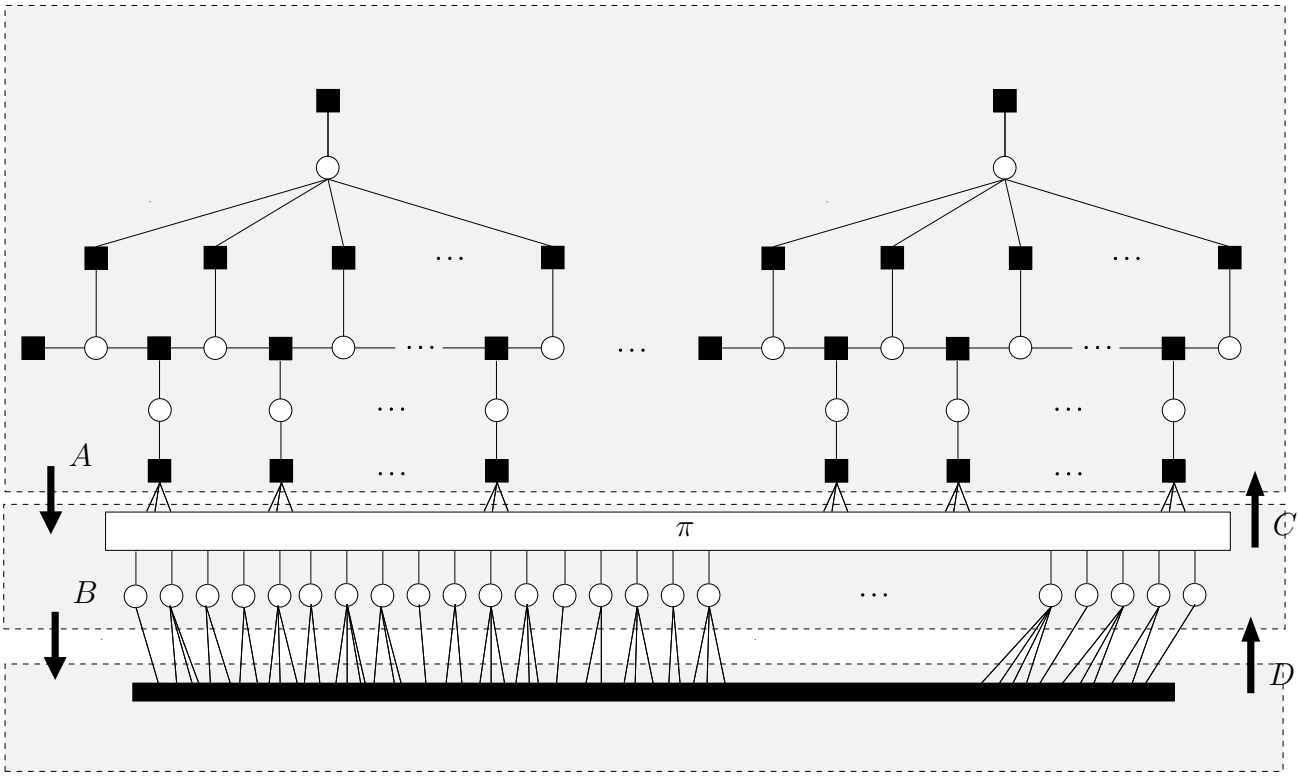


Figura 5.9: Divisão entre blocos demodulador, nós de variável e nós de verificação de paridade. As setas indicam os sentidos das variáveis aleatórias A, B, C e D.

entre os blocos do grafo fator do demodulador. A mensagem A que vai do bloco demodulador aos nós de variável, a mensagem B, que vai dos nós de variável aos nós de verificação e as mensagens C e D, que são as mensagens que fazem o caminho inverso.

5.3 Resultados

As curvas EXIT das figuras 5.10 e 5.11, mostram os pontos-de-queda para um decodificador do canal M-APSK/AWGN para $L = 9$ e $L = 29$ respectivamente. Para $L = 9$ o ponto-de-queda do código chega em 4.7dB e para $L = 29$ chega em 2.5dB, como mostrado em [8] a capacidade de canal para o canal M-APSK/AWGN de bloco $L = 9$ e $L = 29$ são, respectivamente, $C \approx 1.7$dB e $C \approx 1.4$dB, isso nos indica que as curvas EXIT podem servir para calcular o ponto de queda de um sistema de comunicação codificado de forma aproximada.

O comportamento de aproximação da capacidade aumentando o tamanho dos códigos

LDPC, foi demonstrado em [8] através de curvas de probabilidade de erros de bits. Entretanto, aumentar o tamanho do código acarreta em outros problemas, como, por exemplo, o incremento da espera do decodificador.

Para o sistema considerado nesses resultados utilizamos um código LDPC regular de taxa $r_c = 0,5$, modulação 8-APSK, $r = 2,45$. Os resultados foram concentrados apenas na variação do tamanho L do bloco de ruído de fase. Dessa forma podemos prever o comportamento da decodificação para canais de memórias de tamanhos que variam.

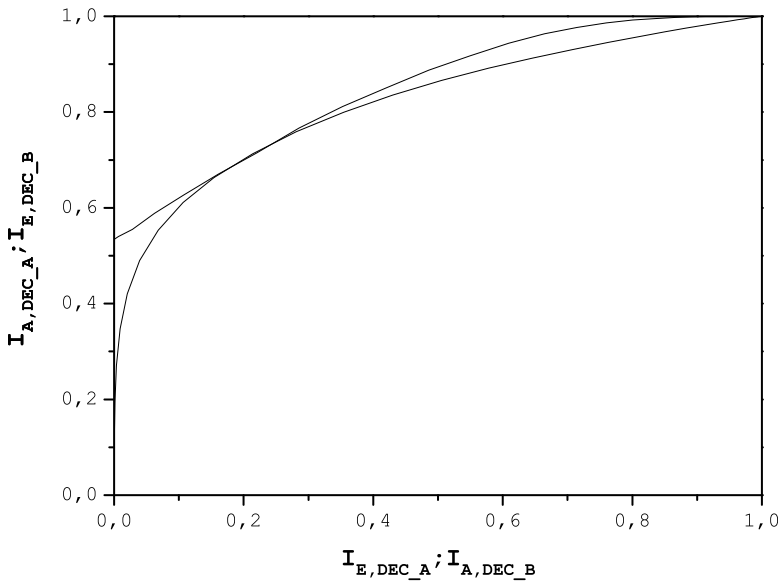


Figura 5.10: Curva EXIT para canal M-APSK/AWGN não coerente de bloco $L = 9$ em seu ponto-de-queda. C(3,6) e $r_c = 0,5$.

As curvas EXIT para esse sistema descrito, foram obtidas utilizando o mesmo cronograma utilizado em [8]. Entretanto, as mensagens iniciais $\mu_{s'_i \to T_i}(s'_i)$ são calculadas primeiramente através do algoritmo *forward-backward*, que faz a decodificação diferencial no demodulador. E então, após a estimação de fase, as mensagens são novamente calculadas e repassadas aos nós de variável. Essas mensagens precisam ser calculadas, pois as variações nas estatísticas das variáveis aleatórias dos sinais no decodificador, acontecem como resultado das iterações do algorítmo de decodificação. Portanto, as mensagens também variam. Caso a mensagem

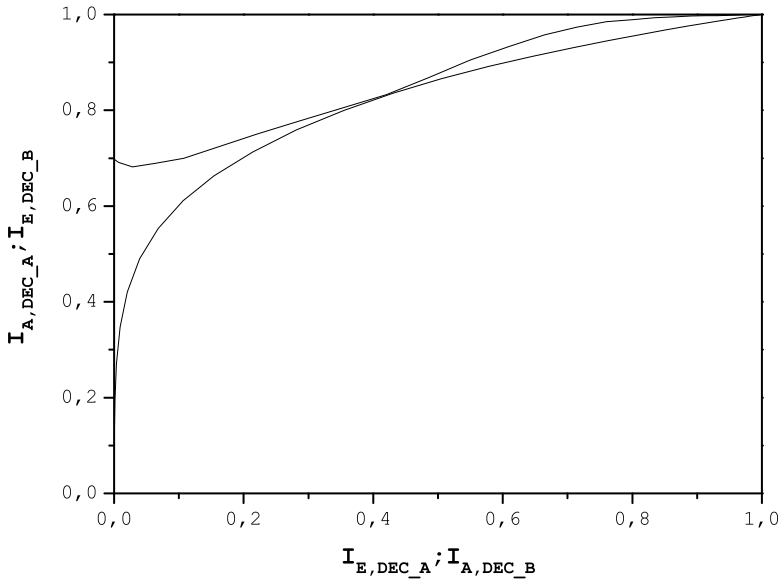


Figura 5.11: Curva EXIT para canal M-APSK/AWGN não coerente de bloco $L = 29$ em seu ponto-de-queda. C(3,6) e $r_c = 0,5$.

seja inicializada com valores regulares, o cronograma funcionaria como se a cada iteração do algorítmo de decodificação essa mensagem fosse reinicializada.

O cronograma para o bloco demodulador funciona então da seguinte forma:

- Primeiramente, sabemos que as mensagens $\mu_{s'_i \to T_i}(s'_i)$ devem ser calculadas, então são gerados valores igualmente prováveis para as os s sinais possíveis da modulação que serão as mensagens $\mu_{T_i \to s'_i}(s'_i)$. Depois são gerados valores aleatórios para as mensagens $\mu_{b_i^\alpha \to V_i}(b_i^\alpha)$, sabemos que essas mensagens são variáveis aleatórias gaussianas D de variância igual a todos os valores relacionados ao mapeamento da função $J^{-1}(I_D)$, e cada valor de σ correspondendos a um passo no processo de decodificação, e média relacionada a equação

$$\mu_D = \frac{\sigma_D^2}{2}. \tag{5.28}$$

  Assim podemos calcular as mensagens $\mu_{s_{i-1} \to \Lambda_i}(s_{i-1})$, necessárias na próxima etapa.

- Depois disso o passo *forward-backward* no processo de decodificação é executado, ob-

tendo as mensagens $\mu_{s'_i \to T_i}(s'_i)$, necessárias para o passo de estimação de fase. O processo de estimação de fase é descrito pelas mensagens $\mu_{\theta_k \to T_i}(\theta_k)$ e dependem do valor Z de amostragem de fase no receptor.

- Após isso, novamente as mensagens $\mu_{b_i^\alpha \to V_i}(b_i^\alpha)$ são calculadas aleatoriamente com mesmos valores de variância para executar o passo *forward-backward* do algoritmo, dessa vez como uma fotografia instantanêa do decodificador no momento em que a informação a priori é utilizada pelo demodulador para fazer a estimação de fase.

- Agora as mensagens $\mu_{V_i \to b_i^\alpha}(b_i^\alpha)$ são calculadas. Essas mensagens, são os valores da variável aleatória A que saem do demodulador e que são utilizadas para calcular a informação mútua I_A.

Esse cronograma é repetido tantas vezes quanto forem os valores de σ_D na entrada no demodulador necessários para fazer o mapeamento da função EXIT do demodulador.

Os tamanhos de L, são referentes ao tamanho do bloco de símbolos s_i enviados pelo canal. Portanto, no caso de $L = 10$, por exemplo, significa que 30 bits foram enviados através do canal, já para $m_c = 3$ temos $3 \cdot 10 = 30$. Os sinais decodificados desses 30 bits, no entanto, têm apenas 27 bits de informação, já que um símbolo é o símbolo de referência.

As figura 5.12, 5.13 e 5.14, deixam claro que quanto maior o valor de L, mais baixo é o ponto-de-queda. Esse comportamento é explicado pois quando $L \to \infty$ o canal se transforma em um canal AWGN não coerente de memória infinita. No entanto, para valores de L muito grandes as curvas EXIT mostram que o decodificador não melhora tanto o sistema, como evidenciado pelas figuras 5.13 e 5.14 onde aumentando o valor de $L = 20$ para $L = 30$ o ganho de 0.3 dB não é tão grande quando comparado ao ganho de $L = 10$ para $L = 20$ que é de 1.5 dB no ponto-de-queda.

Os valores de ponto-de-queda para os sistemas com $L = 10$, $L = 20$ e $L = 30$, são 4.1 dB, 2.6 dB e 2.3 dB respectivamente, o que nos leva a crer que mesmo para canais com memória finita com L muito grande, o ponto-de-queda não modifica em escala linear.

É importante ressaltar que os exemplos mostrados utilizam apenas códigos LDPC regulares, o que concorda com [27], é mostrado que a utilização de códigos LDPC irregulares

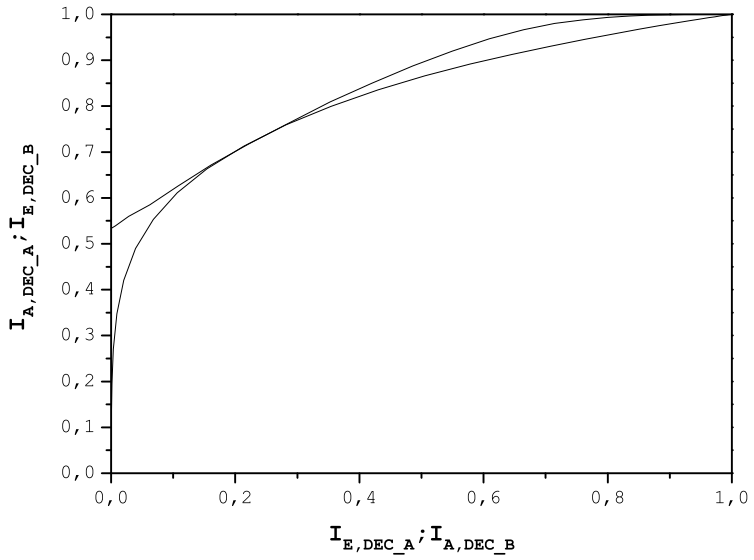


Figura 5.12: Curva EXIT para canal M-APSK/AWGN não coerente de bloco $L = 10$ em seu ponto-de-queda. C(3,6) e $r_c = 0,5$.

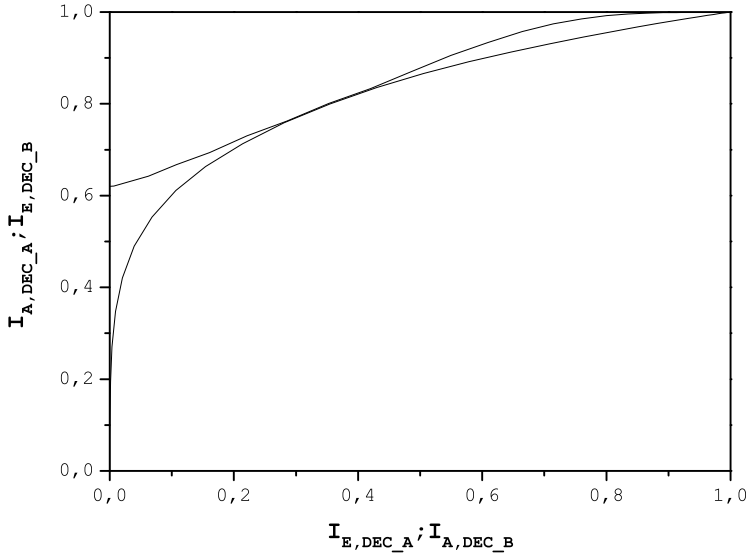


Figura 5.13: Curva EXIT para canal M-APSK/AWGN não coerente de bloco $L = 20$ em seu ponto-de-queda. C(3,6) e $r_c = 0,5$.

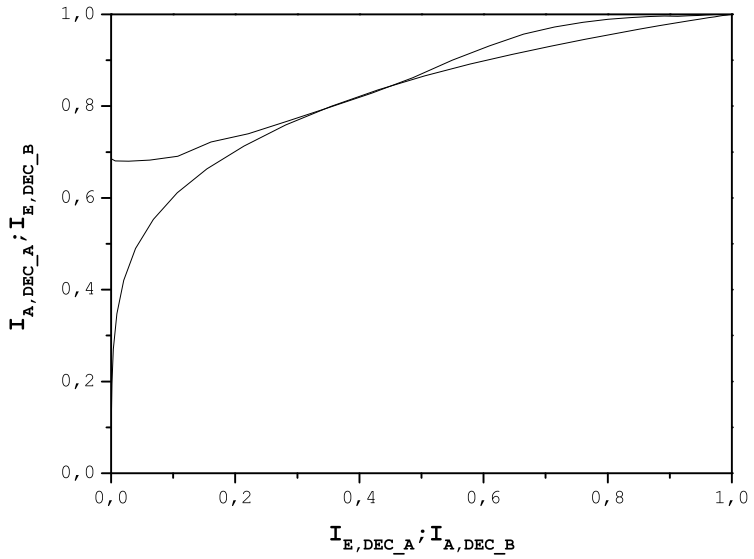


Figura 5.14: Curva EXIT para canal M-APSK/AWGN não coerente de bloco $L = 30$ em seu ponto-de-queda. C(3,6) e $r_c = 0,5$.

podem melhorar o desempenho do sistema, podendo chegar a um ponto-de-queda mais baixo que o obtido para os códigos regulares.

O Projeto desse sistema através de curvas EXIT demonstra a capacidade de análise que as funções EXIT podem oferecer. Nesse caso, as funções EXIT nos dizem que existe um código LDPC, que pode chegar próximo da capacidade calculada em [8], para vários valores de bloco L do canal M-APSK/AWGN, assim como para diversos parâmetros da modulação, como constante de raio e amostragem de fase. No entanto elas não nos dizem qual é esse código, que de forma geral são códigos muito grandes, com palavras código maiores que 50 mil bits. Esses códigos conseguem chegar a uma baixa probabilidade de erro com uma relação sinal-ruído baixa.

Capítulo 6

Conclusões

Neste trabalho, apresentamos aspectos fundamentais sobre a descrição de receptores iterativos utilizando grafos-fatores com decodificação de códigos conhecidos como LDPC. Através dessas descrições, foi possível analisar o processo de decodificação, utilizando uma ferramenta relativamente nova na literatura, as funções EXIT. Essas funções conseguem descrever o comportamento do receptor iterativo de um sistema de comunicação sem que seja preciso simulá-lo completamente.

Através de exemplos, mostramos claramente como se obter as curvas EXIT de um sistema conjunto de demodulador e decodificador. Com esse estudo é possível generalizar o projeto de sistemas utilizando Funções EXIT para qualquer combinação demodulador/decodificador. Isso porque a obtenção das curvas EXIT do processo de decodificação LDPC é baseada em expressões analíticas. Com essas expressões, é possível obter as distribuições de probabilidade das mensagens. Essas distribuições podem, no entanto, ser aproximadas por distribuições normais de probabilidade.

Apresentamos um estudo de caso utilizando o mesmo sistema de transmissão mostrado em [8]. Gráficos mostram que as curvas EXIT proporcionam as informações necessárias para a escolha de códigos LDPC, capazes de alcançar baixa probabilidade de erro de bits. Além da escolha dos códigos, é possível dizer que, como as funções EXIT são feitas sobre um bloco onde também está presente o demodulador, a escolha da modulação pode ser feita em conjunto à do código utilizando as curvas EXIT.

Também foi mostrado que densidade da matriz de paridade para códigos LDPC exerce influência no processo de decodificação. Melhor dizendo, quanto mais esparsa for a matriz

de paridade, espera-se que o sistema codificado alcance a mesma probabilidade de erro para um valor de E_b/N_0 mais baixo. As características de esparsidade da matriz de paridade, também influenciam na complexidade do processo de decodificação. Sendo assim, ao passo que a matriz se torna mais esparsa, menos operações serão realizadas pelo algorítmo soma produto.

O ponto-de-queda do sistema proposto em [8], pode ser ainda mais baixo, como mostram os resultados. Para isso, esperamos que a utilização de códigos mais longos e/ou códigos irregulares seja suficiente para reduzir ainda mais o ponto-de-queda do sistema estudado.

Referências Bibliográficas

[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[2] R. Gallager, "Low-Density Parity-Check Codes," *Tese de doutorado*, 1963.

[3] D. MacKay and R. Neal, "Good Codes Based on Very Sparse Matrices," in *IMA: IMA Conference on Cryptography and Coding, LNCS lately (earlier: Cryptography and Coding II, Edited by Chris Mitchell, Clarendon Press, 1992)*, 1995.

[4] D. Mackay and R. Neal, "Near Shannon Limit Performance of Low-Density Parity Check Codes," *IEEE Transactions on Information Theory*, vol. 32, no. 18, pp. 1645–1646, 1996.

[5] S. ten Brink, "Convergence of Iterative Decoding," *Electronics Letters*, vol. 35, pp. 806–808, 1999.

[6] S. ten Brink, "Convergence Behavior of Iteratively Decoded Parallel Concatenated Codes," *IEEE Transactions on Communications*, vol. 49, pp. 1727–1737, 2001.

[7] B. Frey, F. Kschischang, H. Loeliger, and N. Wiberg, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[8] D. Cunha, "Canal M-APSK Não-Coerente de Bloco: Capacidade e Proposta de Codificação para Receptores Iterativos," *Tese de doutorado, FEEC, UNICAMP*, 2006.

[9] T. M. Cover and J. A. Thomas, E*lements of* I*nformation* T*heory*. John Wiley & Sons, Inc., 1991.

[10] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, vol. vol. IT-20(2), pp. 284–287, March 1974.

[11] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, pp. 260– 269, Apr 1967.

[12] M. Tanner, "A Recursive Aproach to Low Complexity Codes," *IEEE Transactions on Information Theory*, vol. 27, no. 5, pp. 533–547, 1981.

[13] D. MacKay, "Good Error Correcting Codes Based on Very Sparce Matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 599–618, 1999.

[14] W. E. Ryan, "An Introduction to LDPC Codes," tech. rep., Department of Electrical and Computer Engineering, The University of Arizona, Aug. 19, 2003.

[15] M. Yang, W. E. Ryan, and Y. Li, "Design of Efficiently Encodable Moderate-Length High-Rate Irregular LDPC Codes," *IEEE Transactions on Communications*, vol. 52, no. 4, Apr 2004.

[16] Urbanke, T. Richardson, and A. Shokrollahi, "Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 619–637, 2001.

[17] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, D*igital* C*ommunication*. Kluwer Academic Publishers, 2004.

[18] J.Hagenauer, E.Offer, and L.Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Trans. on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar 1996.

[19] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient Implementations of the Sum-Product Algorithm for Decoding LDPC Codes," *In Proceedings of IEEE*

Global Conference on Telecommunications (GLOBECOM'01), vol. 2, pp. 1036–1036E, San Antonio, Texas, EUA, Nov 2001.

[20] S. ten Brink, A. Ashikhmin, and G. Kramer, "Extrinsic Information Transfer Functions: A Model and Two Properties," *IEEE Transactions on Information Theory*, vol. 50, pp. 2657–2673, 2004.

[21] S. ten Brink, A. Ashikhmin, and G. Kramer, "Design of Low-Density Parity-Check Codes for Modulation and Detection," *IEEE Transactions on Communications*, vol. 52, pp. 670–678, April 2004.

[22] S. Y. Chung, R. Urbanke, and T. J. Richardson, "Gaussian Approximation for Sum-Product Decoding of Low-Density Parity-Check Codes," *ISIT*, p. 318, 2000.

[23] S. Y. Chung, R. Urbanke, and T. J. Richardson, "Analysis of Sum-Product Decoding of Low-Density Parity-Check Codes Using a Gaussian Approximation," *IEEE Transactions on Information Theory*, vol. 47, pp. 657–670, 2001.

[24] F. R. Kschischang and M. Ardakani, "A More Accurate One-Dimensional Analysis and Design of Irregular LDPC Codes," *IEEE Transactions on Communications*, vol. 52, pp. 2106–2114, 2004.

[25] E. Zehavi, "8-PSK Trellis Codes for a Rayleigh Channel," *IEEE Transactions Communications*, vol. 40, no. 5, pp. 873–884, May 1992.

[26] E. Biglieri, G. Caire, and G. Taricco, "Bit-Interleaved Coded Modulation," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 927–946, May 1998.

[27] M. Franceschini, G. Ferrari, R. Raheli, and A. Curtoni, "Serial Concatenation of LDPC Codes and Differential Modulations," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 1758–, Sep, 2005.

Apêndice A

Informação mútua de uma V.A. gaussiana.

Nesse anexo mostramos uma aproximação polinômial feita em [6] e [21] sobre o cálculo da informação mútua entre duas variáveis aleatórias gaussianas.

Considere $Y = X + N$, onde $P(X = \pm 1) = 1/2$ e N gaussiano de média zero e σ_n^2. O valor LLR do canal de entrada é função de y, e escrito como $L_{canal}(y)$. Note que $L_{canal}(Y)$ condicionado em $X = \pm 1$ é gaussiano com média $\mu_{canal} = \pm 2/\sigma_{canal}^2$ e variância $\sigma_{canal}^2 = 4/\sigma_n^2$. Também tem-se que

$$\mu_{canal} = \frac{\pm \sigma_{canal}^2}{2}.$$

Fazendo $J(\sigma_{canal})$ a informação mútua $I(X; L_{canal}(Y))$. tem-se que:

$$\begin{aligned} J(\sigma_{canal}) &= H(X) - H(X|L_{canal}(X)) && \text{(A.1)} \\ &= 1 - \int_{-\infty}^{\infty} \frac{e^{-\left(\xi - \sigma_{canal}^2/2\right)^2/2\sigma_{canal}^2}}{\sqrt{2\pi\sigma_{canal}^2}} \cdot \log_2[1 + e^{-\xi}] d\xi && \text{(A.2)} \end{aligned}$$

onde $H(X)$ é a entropia de X e $H(X|L_{canal}(Y))$ é a entropia condicionada a L_{canal}. Note que $I(X; L_{canal}(Y))$ é o mesmo que $I(X; Y)$. A capacidade do canal $X = Y + N$ é, portanto $J(\sigma_{canal} = J(2/\sigma_n)$.

Para implementação, a curva $J(\cdot)$ foi dividida em duas partes correspondentes ao intervalo $0 \leq \sigma \leq \sigma^*$ e $\sigma < \sigma^*$ onde $\sigma^* = 1.6363$. Para o primeiro intervalo foi utilizada uma aproximação polinomial, enquanto para o segundo uma aproximação exponencial. O método de busca utilizado para a aproximação da função é conhecido como algoritmo Marquadrat-

Levenberg, chegando a:

$$J(\sigma) \approx \begin{cases} a_{J,1}\sigma^3 + b_{J,1}\sigma^2 + c_{J,1}\sigma & ,0 \leq \sigma \leq \sigma^* \\ 1 - e^{a_{J,2}\sigma^3 + b_{J,2}\sigma^2 + c_{J,2}\sigma + d_{J,2}} & ,\sigma^* < \sigma < 10 \\ 1 & ,\sigma \geq 10 \end{cases} \tag{A.3}$$

onde

$$\begin{array}{llll} a_{J,1} = -0.0421061 & b_{J,1} = 0.209252 & c_{J,1} = -0.00640081 & \\ a_{J,2} = 0.00181491 & b_{J,2} = -0.142675 & c_{J,2} = -0.0822054 & d_{J,2} = 0.0549608 \end{array}$$

Para a inversa da função $J(\cdot)$ divide-se a curva em dois intervalos em $I^* = 0.3646$. A função $J^{-1}(\cdot)$ fica:

$$J^{-1}(I) \approx \begin{cases} a_{\sigma,1}I^2 + b_{\sigma,1}I + c_{\sigma,1}\sqrt{I} & ,0 \leq I \leq I^* \\ -a_{\sigma,2}ln[b_{\sigma,2}(1-I)] - c_{\sigma,2}I & ,I^* < I < 1 \end{cases} \tag{A.4}$$

onde

$$\begin{array}{lll} a_{\sigma,1} = 1.09542 & b_{\sigma,1} = 0.214217 & c_{\sigma,1} = 2.33727 \\ a_{\sigma,2} = 0.706692 & b_{\sigma,2} = 0.386013 & c_{\sigma,1} = -1.75017 \end{array}$$