INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE PERNAMBUCO

CAMPUS JABOATÃO DOS GUARARAPES

Pós-Graduação em Gestão e Qualidade em Tecnologia da Informação e Comunicação

RODRIGO FERREIRA DA SILVA

COMPARAÇÃO DE ALGORITMOS PARA DETECÇÃO DE ERROS NA TRANSMISSÃO

Jaboatão dos Guararapes

2021

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

RODRIGO FERREIRA DA SILVA

COMPARAÇÃO DE ALGORITMOS PARA DETECÇÃO DE ERROS NA TRANSMISSÃO

Trabalho de Conclusão de Curso apresentado como requisito final para obtenção do diploma de Pós-Graduação Lato Sensu em Gestão e Qualidade em Tecnologia da Informação e Comunicação do Instituto Federal de Ciência e Tecnologia de Pernambuco, campus Jaboatão dos Guararapes, sob orientação o professor doutor Sérgio Torres de Santana e co-orientação do mestre Emerson Ferreira da Silva.

Jaboatão dos Guararapes

2021

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

FICHA CATALOGRÁFICA

S586c Silva, Rodrigo Ferreira da.

Comparação de algoritmos para detecção de erros na transmissão / Rodrigo Ferreira da Silva; orientador Prof. Dr. Sérgio Torres de Santana; coorientação Ms. Emerson Ferreira da Silva. Jaboatão dos Guararapes: IFPE, 2021.

61 f.; il.

Trabalho de Conclusão de Curso (Especialização em Gestão e Qualidade em Tecnologia da Informação e Comunicação) – IFPE - Campus Jaboatão dos Guararapes.

. Inclui Referências.

1. Tecnologia da Informação e Comunicação 2. Sistema de transmissão de dados. 3. Algoritmos computacionais. 4. Erro I. Santana, Sérgio Torres. II. IFPE. III. Título.

CDD 005.1068

Catalogação na fonte elaborada pela Bibliotecário Thiago Leite Amaro da Silva - CRB4/ 2.021

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Instituto Federal de Educação, Ciência e Tecnologia de Pernambuco
Campus Jaboatão dos Guararapes
Divisão de Pesquisa e Extensão e Pós-graduação

ATA DE REALIZAÇÃO DE BANCA

No dia **22** de **março** de **2021** as **20h** na sala **on-line** do IFPE Campus Jaboatão dos Guararapes, compareceram à banca de defesa do Trabalho de Conclusão de Curso da Especialização *lato sensu* em **Gestão e Qualidade em Tecnologia da Informação e Comunicação,** do(a) aluno(a) **Rodrigo Ferreira da Silva** que defendeu o trabalho intitulado **Comparação de Algoritmos para Detecção de Erros na Transmissão**, os(as) professores(as) que compõem a banca descrita abaixo, e concederam a nota 8,3 sendo o(a) aluno(a) considerado(a) APROVADO de acordo com a composição das notas estabelecida pela banca avaliadora.

COMPOSIÇÃO DA BANCA		
	NOTA	**ASSINATURA**
Prof. Sérgio Torres de Santana (presidente da banca)	9	Documento assinado digitalmente gov.br Sergio Torres de Santana Data: 30/03/2021 20:32:07-0300 CPF: 963.333.674-00
Prof. Luciano de Souza Cabral (1 avaliador)	8	Documento assinado digitalmente gov.br Luciano de Souza Cabral Data: 31/03/2021 08:58:13-0300
Prof. Elmano Ramalho Cavalcanti (2 avaliador)	8	SERPRO Assinado digitalmente por: ELMANO RAMALHO CAVALCANTI Sua autenticidade pode ser confirmada no endereço : <http://www.serpro.gov.br/assinador-digital>
NOTA FINAL	8,3	

DocuSigned by:
6DB129DB17A849A...

Rodrigo Ferreira da Silva

Nilson Cândido de Oliveira Júnior
Coordenador do Curso de Pós-Graduação em Gestão e Qualidade em TIC
SIAPE: 1829625

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Agradecimentos

Agradeço primeiramente a Deus, por me guiar, passar confiança e motivação necessárias para que eu possa alcançar meus objetivos.

A minha mãe Maria José Ferreira da Silva, e meu pai José Ferreira da Silva, que sempre foram juntos, meu maior alicerce, responsáveis por toda estrutura e apoio necessários em cada passo de minha jornada.

Agradeço a meu irmão Emerson Ferreira da Silva, pelo apoio na elaboração das simulações e co-orientação neste trabalho, como também por todo esforço e empenho no que diz respeito a mim, pois se sou o que sou, grande parte é reflexo da convivência com ele.

Ao meu orientador Sérgio Torres de Santana, por toda orientação e suporte transmitidos para elaboração deste trabalho.

Por fim, agradeço a toda minha família e amigos que contribuíram de certa forma para realização deste trabalho.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Dedico este trabalho a minha família.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

“Mesmo que a vida pareça difícil, há sempre algo que você pode fazer para ter sucesso nela.”

(Stephen Hawking)

Resumo

Com intuito de atenuar erros causados por ruídos ou interferências em um sistema de transmissão de dados, são aplicadas diversas técnicas. Um método bastante eficaz para detecção de erros em transmissões digitais, o Algoritmo de Viterbi, agregado aos conceitos de codificadores convolucionais foram e ainda são bastante utilizados em sistemas de comunicações. Este trabalho procura, através de vários estudos científicos, trazer maiores esclarecimentos sobre o desempenho dos codificadores convolucionais, bem como mostrar seu funcionamento através de simulações, com o objetivo de ilustrar a grande melhoria que se tem ao inserir um codificador deste tipo em um sistema de transmissão. Através das técnicas aplicadas, foi possível reduzir a taxa de erros em até 10 vezes na taxa de erros no sistema de transmissão com canal com ruído aditivo gaussiano branco.

Palavras-Chave: Viterbi, palavra-código, codificadores.

Abstract

In order to mitigate errors caused by noise or interference in a data transmission system, several techniques are applied. A very effective method for detecting errors in digital transmissions, the Viterbi algorithm, together the concepts of convolutional encoders were and still are quite used in communication systems. This work seeks, through various scientific studies, bringing further explanation on the performance of convolutional encoders, as well as show its operation through simulations, with the purpose of illustrate the great improvement that has to insert an encoder like this in a transmission system. Through the applied techniques, it was possible to reduce the error rate by up to 10 times in the error rate in the transmission system with Additive White Gaussian Noise channel.

Keywords: Viterbi, codeword, encoders.

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Sumário

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Índice de Figuras

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Índice de Gráficos

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Índice de Tabelas

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Lista de Símbolos e Siglas

AWGN – *Additive White Gaussian Noise*
BER – *Bit Error Rate*
BPSK – *Binary Phase-Shift Keying*
BSC – *Binary Symmetric Channel*
DMC – *Discrete Memoryless Channel*
FIR – *Finite Impulse Response*
FSSM – *Finite State Sequential Machine*
GF – *Galois Fields*
RSC – *Recursive Systematic Convolutional*
SNR – *Signal-to-Noise Ratio*
SOVA – *Soft Output Viterbi Algorithm*

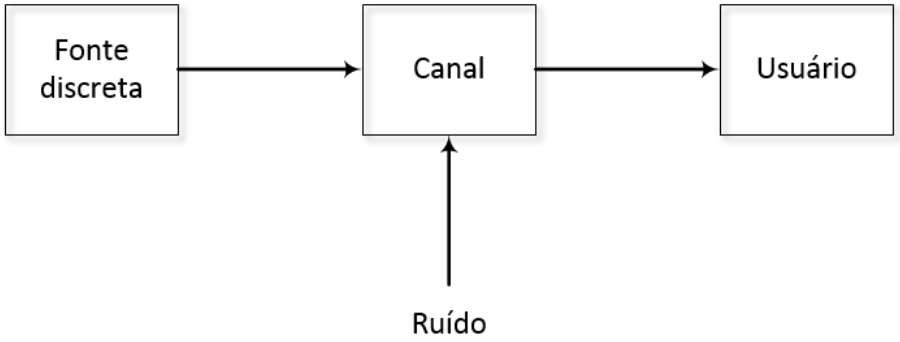
1. Introdução

O estudo dos códigos corretores de erros é um campo de pesquisa muito explorado na atualidade, nas mais diversas áreas, tais como: matemática, computação, engenharias, estatísticas entre outras (HAYKIN, 2004).

Em um sistema de transmissão de dados, em casos reais, às vezes podem ocorrer problemas como interferências eletromagnéticas, ou até mesmo erros causados por ruídos que fazem com que a mensagem recebida não seja a igual à mensagem enviada. O foco destes estudos é desenvolver métodos que permitam detectar e corrigir estes erros (MOON, 2005).

Na Figura 1, segue um esquema de transmissão, onde o mesmo sofre inserção de ruído.

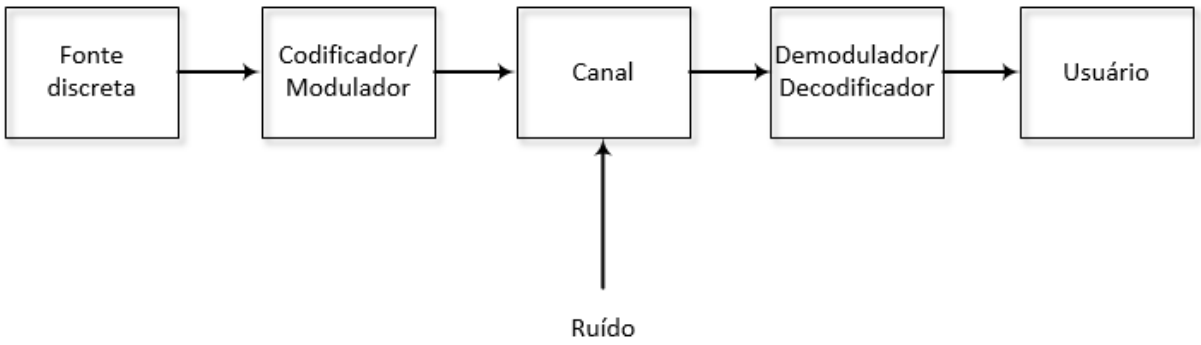
Figura 1 – Esquema de transmissão sem codificação.



Fonte: Elaborada pelo autor (2020).

No sistema da Figura 1, o receptor irá receber a mensagem enviada somada ao ruído que afetou o canal. A Figura 2, mostra o esquema de uma transmissão com a inserção de codificador e decodificador antes e depois do canal, respectivamente. Dessa forma, o codificador irá manipular a mensagem enviada, de tal modo que, após a passagem desta pelo canal ruidoso, o decodificador seja capaz de detectar e corrigir erros ocasionados pelo ruído, retornando uma mensagem estimada mais próxima da mensagem enviada e reduzindo bastante o número de erros da mensagem que chega ao receptor.

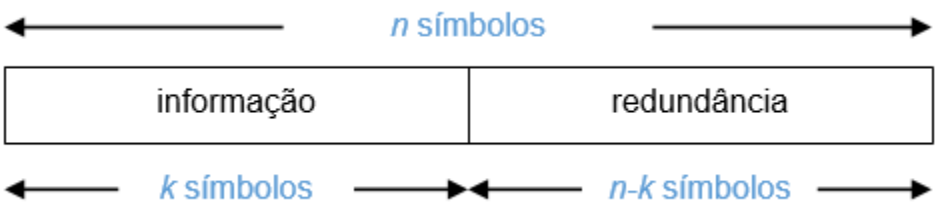
Figura 2 – Esquema de transmissão com codificação.



Fonte: Elaborada pelo autor (2020).

Os códigos de correção de erro basicamente partem do mesmo princípio: Adiciona-se redundância à informação, com o objetivo de corrigir eventuais erros no processo de armazenamento e/ou transmissão. Símbolos redundantes são adicionados aos símbolos de informação, fazendo com que se tenha uma sequência codificada, também conhecida como *palavra-código*. A Figura 3 ilustra uma palavra-código obtida por codificação de um código de bloco. Esta codificação é dita como sistemática, ou seja, que nas primeiras posições (*k)* da palavra código, aparecem os símbolos de informação. Nos demais símbolos (*n-k)* da palavra-código estão algumas funções do símbolo de informação, e uma delas é fornecer redundância que pode ser utilizada para correção e/ou detecção de erros (MORELOS-ZARAGOZA, 2002).

Figura 3 – Estrutura de uma palavra-código.



Fonte: Elaborada pelo autor (2020).

Dependendo da forma que a redundância é adicionada a informação, os códigos de correção de erro se dividem em 2 pilares: Códigos de Bloco e Códigos Convolucionais. E cada um desses 2 tipos citados tem suas aplicações específicas.

Alguns estudos (HAYKIN, 2004; MOON, 2005) mostravam a preferência pelo uso dos códigos convolucionais devido à capacidade de decisão suave do Algoritmo de Viterbi, pois houve uma crença durante muitos anos que os códigos de bloco não teriam tanta eficácia em decodificações de decisões suaves. Porém, estudos posteriores ajudaram a erradicar essa crença, mostrando que os códigos de bloco são também bastante eficazes (MORELOS-ZARAGOZA, 2002).

Os códigos de bloco processam a informação bloco a bloco, ou seja, cada bloco de informação independe dos outros blocos. Logo, códigos de bloco são denominados como códigos sem memória.

Em contrapartida, os códigos convolucionais não dependem apenas da informação de entrada, como também da saída dos estados anteriores.

1.1. Objetivos e Metas

Os objetivos e metas deste trabalho compreendem a apresentação de uma análise da utilização de códigos corretores de erros, mediante a utilização de códigos convolucionais, com o intuito de gerar um sistema menos susceptível a erros.

Cientes da complexidade em que consiste na eliminação dos erros em um determinado sistema de comunicação, espera-se que grande parte dos erros sejam detectados e corrigidos, garantindo uma alta confiabilidade no sistema (PROAKIS, SALEHI & BAUCH, 2006).

1.2. Metodologia e Estratégia de Ação

A metodologia empregada neste trabalho é baseada nos estudos de códigos convolucionais, com foco no algoritmo de Viterbi, analisando alguns possíveis erros que possam vir a ocorrer através de uma simulação.

A fase inicial, uma das mais importantes, consiste em um estudo sobre os códigos convolucionais, haja vista que este conhecimento possibilita estimar as necessidades e carências do projeto, bem como especificar quais resultados esperar.

O passo seguinte é a simulação de um sistema de transmissão com a inserção de ruído, fazendo uso de codificação no sistema de transmissão, para que chegue ao receptor uma mensagem mais próxima da mensagem enviada, ou seja, com menos erros. São utilizados 2 codificadores para efeitos de comparação: codificador RSC na primeira simulação, e um codificador FIR para a segunda.

1.3. Organização do Trabalho

Este trabalho está organizado em 7 capítulos. No capítulo 2 estão reunidos conceitos básicos necessários para o entendimento de codificadores.

O capítulo 3 apresenta informações sobre códigos convolucionais e suas representações.

O capítulo 4 faz uma abordagem sobre o Algoritmo de Viterbi, mostrando seu funcionamento, e métricas utilizadas para que o receptor obtenha informação com a menor taxa de erro possível. Já no capitulo 5 é realizada uma explanação sobre o algoritmo de Viterbi de decisão suave, mostrando suas vantagens e desvantagens em relação ao algoritmo de Viterbi.

O capítulo 6 é reservado para expor simulações e seus respectivos resultados baseados nos conhecimentos descritos nos capítulos anteriores. E o capítulo 7 é destinado para a conclusão e propostas de trabalhos/estudos posteriores.

2. Conceitos Básicos

2.1. O canal de comunicação

Todo processo de comunicação tem seu início numa ponta chamada “transmissor” e termina na outra ponta chamada “receptor”. Entre essas 2 pontas do processo de comunicação deve haver um meio que será encarregado de levar a informação desde sua origem até seu destino, e este meio é chamado de “canal” (HAYKIN, 2004).

O canal pode ser baseado em propagação guiada, que são meios como cabos coaxiais, fibras óticas ou cabos de pares trançados. Também existe a utilização do ar como canal de comunicação, como por exemplo, na telefonia celular e nas transmissões via Satélite (MORELOS-ZARAGOZA, 2002).

2.1.1. Canais Discretos sem Memória

Um *canal discreto sem memória* (DMC, do inglês *Discrete Memoryless Channel*) é representado por um modelo estatístico com entrada $\mathcal{X}$ e saída $\mathcal{Y}$, que são variáveis aleatórias, onde o $\mathcal{Y}$ representa uma versão ruidosa da entrada. A cada instante de tempo, é introduzido no canal um símbolo de entrada $\mathcal{X}$, que faz parte do um determinado alfabeto $\mathcal{X}$. Para cada símbolo inserido na entrada, é ocasionada uma resposta com um símbolo de saída $\mathcal{Y}$ que é parte integrante de um alfabeto $\mathcal{Y}$. Um canal é dito como *discreto*, quando estes dois alfabetos $\mathcal{X}$ e $\mathcal{Y}$, possuem tamanhos finitos. Um canal é denominado “sem memória” quando a resposta atual na saída $\mathcal{Y}$ depende unicamente da entrada neste mesmo instante tempo, e não de instantes de tempo anteriores (HAYKIN, 2004; MORELOS-ZARAGOZA, 2002).

Um fator relevante para o estudo desses alfabetos é que estes dois alfabetos, de entrada e saída, não precisam ser de mesmo comprimento, podendo ser descrito da seguinte maneira:

$$\mathcal{X} = \{x_0, x_1, \dots, x_{J-1}\}, \tag{2.1}$$

$$y = \{y_0, y_1, \dots, y_{K-1}\}. \tag{2.2}$$

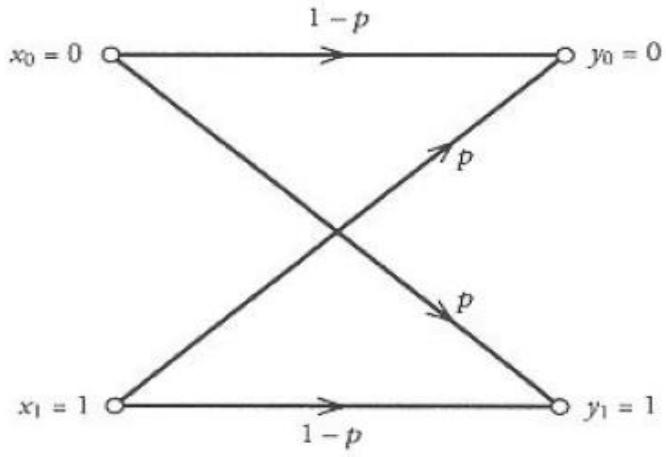
Em que J e K são termos independentes entre si.

2.1.2. Canal Binário Simétrico

O canal binário simétrico (BSC, do inglês *Binary Symmetric Channel*) é um caso particular do DMC, sendo $J = K = 2$. Este tipo de canal é composto por dois símbolos de entrada $(x_0 = 0, x_1 = 1)$ e dois símbolos de saída $(y_0 = 0, y_1 = 1)$. O canal é dito ser simétrico se a probabilidade de ser recebido um número 0 se um 1 for enviado for a mesma de ser recebido um número 1 se um 0 for enviado. Essa probabilidade de inversão de bit é indicada por *p* (HAYKIN, 2004; MORELOS-ZARAGOZA, 2002).

A Figura 4 mostra um esquema de um BSC e suas probabilidades de transições.

Figura 4 – Canal Binário Simétrico.



Fonte: Elaborada pelo autor (2020).

2.1.3. Canal AWGN

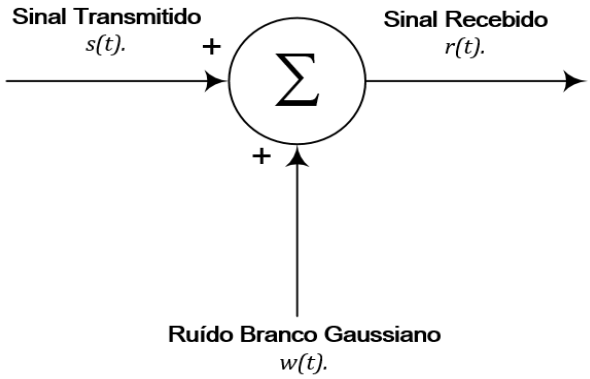
Outro tipo de canal é o canal com ruído branco gaussiano aditivo (AWGN, do inglês *Additive White Gaussian Noise*). Este canal tem como característica a adição de um ruído gaussiano branco ao sinal transmitido, chamado de $w(t)$. O canal AWGN

é representado analiticamente em função da entrada $s(t)$ e da saída $r(t)$ (PROAKIS et al., 2006; SKLAR, 1993), logo:

$$r(t) = s(t) + w(t). \tag{2.3}$$

O ruído é aditivo porque é somado ao sinal de informação, é dito ser branco porque existe em todas as componentes de frequência e, por último, é gaussiano porque a distribuição de amplitude é gaussiana. A Figura 5 ilustra o modelo analítico.

Figura 5 – Inserção do ruído AWGN.



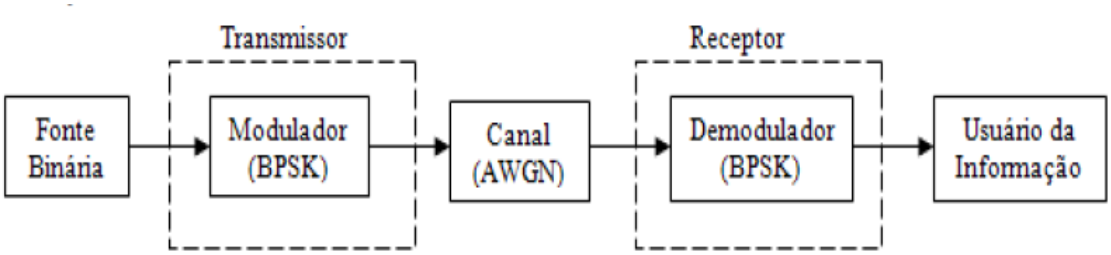
Fonte: Elaborada pelo autor (2020).

2.2. Modulação / Demodulação BPSK

Qualquer processo de comunicação, independente do canal utilizado, está sujeito a influência de ruídos. Portanto, com o objetivo de garantir a integridade da informação que passa pelo canal, o transmissor altera a sequência enviada de modo mais adequado para passar pelo canal ruidoso. Este processo é chamado de modulação. Ao chegar no receptor, o sinal é reconstruído com base no sinal modulado e degradado pelo canal ruidoso. Este processo é chamado de demodulação que, em outras palavras, significa retirar a modulação criada no transmissor (PROAKIS et al., 2006; SKLAR, 1993).

A Figura 6 ilustra em diagrama de blocos do processo de modulação e demodulação.

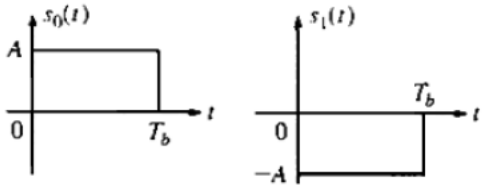
Figura 6 – Processo de modulação/demodulação BPSK.



Fonte: Elaborada pelo autor (2020).

Note que a modulação utilizada no esquema da Figura 6 é a Binária por Deslocamento de Fase (BPSK, do inglês *Binary Phase-Shift Keying*). O BPSK utiliza uma técnica de transmissão de sinais antipodais, ou seja, são transmitidos dois sinais com formas de onda opostas, representados por duas ondas defasadas de 180 graus uma da outra (PROAKIS et al., 2006; SKLAR, 1993), de acordo com a Figura 7.

Figura 7 – Sinais antipodais.



Fonte: Elaborada pelo autor (2020).

De acordo com a Figura 7, $s_0(t) = s(t)$ e $s_1(t) = -s(t)$ são responsáveis por transmitir a informação binária em um sistema com modulação BPSK, onde S(t) representa uma forma de onda qualquer com energia $\sqrt{E_b}$, que representa a energia do sinal em cada bit transmitido (PROAKIS et al., 2006; SKLAR, 1993). Logo, a representação dos bits 0 e 1 é feita com base na forma de onda $s_0(t)$ e $s_1(t)$, respectivamente, de tal maneira:

$$s_0(t) = \sqrt{E_b}, \tag{2.4}$$

$$s_1(t) = -\sqrt{E_b}. \tag{2.5}$$

Logo, com base nessa representação, a saída de um canal AWGN fica na forma:

$$r(t) = \pm\sqrt{E_b} + w(t). \tag{2.6}$$

A demodulação é realizada com um receptor de correlação ou um filtro casado em série com um detector, que tem função realizar a máxima verossimilhança, que é a técnica utilizada para decidir se o sinal transmitido foi $s_0(t)$ ou $s_1(t)$.

No caso de os bits enviados serem equiprováveis, o detector demodula avaliando o valor de $r(t)$. Se $r(t) > 0$, o sinal é demodulado como $s(t)$ (0 binário), e no caso de $r(t) < 0$, o sinal é demodulado como $-s(t)$ (1 binário).

2.3. Relação Sinal Ruído

A Relação Sinal Ruído (SNR, do inglês *Signal-to-Noise Ratio*) é um parâmetro de desempenho de um sistema de comunicação que relaciona a Potência do Sinal (P_S) com a Potência do Ruído (P_N) presente neste sistema (PROAKIS et al., 2006; SKLAR, 1993). A relação é feita da seguinte forma:

$$SNR = \frac{Ps}{P_N}. \tag{2.7}$$

Significa que quanto maior a SNR, menor é a influência do ruído no sistema de transmissão, portanto, deve-se alcançar o menor P_N possível para que se tenha uma boa SNR.

Geralmente o SNR é analisado em decibéis (dB), e a conversão para dB é realizada com a seguinte igualdade:

$$SNR_{(dB)} = 10 \log (SNR). \quad (2.8)$$

2.4. Taxa de Erro de Bit

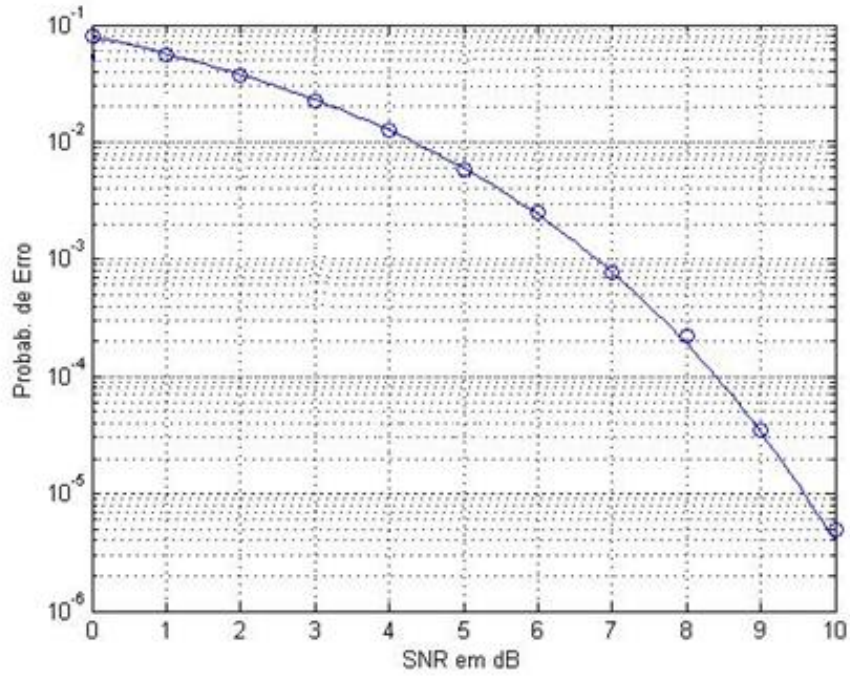
A Taxa de Erro de Bit (BER, do inglês *Bit Error Rate*), é um parâmetro utilizado para avaliar a quantidade de bits que foram afetados por erros em um total de bits enviados, dada pela seguinte igualdade (PROAKIS et al., 2006; SKLAR, 1993):

$$P_e = \frac{[erfc(\sqrt{SNR})]}{2}. \quad (2.9)$$

Onde $erfc$ representa a função erro complementar.

A BER, também chamada de probabilidade de erro, tem relação com a SNR, conforme ilustrado no Gráfico 1.

Gráfico 1 – Curva BER x SNR.



Fonte: Elaborada pelo autor (2020).

O Gráfico 1 mostra uma curva de desempenho de uma mensagem transmitida em um canal AWGN. Essa curva é a representação de uma transmissão sem codificação, que será utilizada como referência nos comparativos com a curva com codificação neste trabalho.

Observando o Gráfico 1, é possível visualizar que quanto maior o nível de SNR (menor nível de ruídos e distorções no sistema), menor será a quantidade de bits com erro (BER), que é uma característica desse tipo de curva.

2.5. Circuitos Sequenciais Lineares

Uma parte muito importante de codificadores convolucionais são os circuitos sequenciais lineares (HAYKIN, 2004). Estes circuitos são constituídos usando unidades básicas de memória, ou atrasos (*delay*) combinados com somadores e

multiplicadores que operam sobre campos de Galois (GF, do inglês *Galois Fields*) (MOREIRA & FARREL, 2006).

Os GF(q) são definidos para todos os números primos q e suas respectivas potências. Um GF(q) bastante utilizado em codificadores é o GF(q) dado em um campo binário. Este representa um caso particular em que q = 2. Em uma sequência de n componentes $(a_0, a_1, \ldots, a_{n-1})$, também chamada de vetor de n-componentes, cada um destes é parte de um GF(2) e os valores possíveis para cada elemento da sequência serão 0 e 1 (MOREIRA & FARREL, 2006).

Estes circuitos sequenciais lineares são chamados também de Máquina Sequêncial de Estados Finitos (FSSM, do inglês *Finite State Sequential Machine*) (TANNER, 1981). A quantidade de atrasos ou unidades de memória define o nível de memória e a capacidade de correção de erro de um código convolucional Cconv (n,k,m), onde:

n: número de saídas do circuito;

k: número de entradas do circuito;

m: número de elementos de memória do codificador;

Em que todos os elementos de memória juntos recebem o nome de registrador de deslocamento, onde M é comprimento de restrição de um código convolucional, expresso em bits, que representa o número de deslocamentos necessários para que um bit de mensagem entre e saia do registrador de deslocamento ($M = m + 1$) (HAYKIN, 2004).

Cada estado da FSSM é atribuído à unidade de memória correspondente. Estas máquinas podem ter como variáveis bits, ou um vetor de bits estendido como um elemento de campo, sobre os quais a FSSM é definida (AMMAR, HONARY, KOU, XU & LIN, 2004; ARNONE, GAYOSO, GONZALEZ & CASTIÑEIRA, 2005; BRINK, KRAMER & ASHIKHMIN, 2004; HAGENHAUER, OFFER & PAPKE, 1996; KOU, LIN & FOSSORIER, 2001; NAYAGAM, 2002; TANG, XU, KOU, LIN & ABDEL-GHAFFAR,

2004; TANNER, 1981). Geralmente há uma representação binária dos elementos que utilizam a forma de um vetor de componentes retirados de GF(2), nestas estruturas algébricas.

Basicamente, um codificador convolucional, é criado a partir do uso de FSSMs, que, para uma certa entrada de dados, resulta em uma dada sequência. Este conjunto de sequência constituem os Códigos Convolucionais (Cconv).

3. Códigos Convolucionais

Em codificação de bloco, a entrada do codificador é representada por uma mensagem de k bits, sendo gerada uma palavra-código de n bits. Como o próprio nome diz, as palavras-código são formadas bloco a bloco. Portanto, deve ser tomado o cuidado necessário para que seja analisado o bloco de mensagem inteiro para que a palavra-código relacionada seja gerada, e isso requer armazenamento de dados. Em contrapartida, pode haver determinados casos em que os bits de mensagem cheguem serialmente, e não em blocos, tornando, desta forma, inviável o uso de codificadores de bloco. Para sistema como esses, é dado o nome de codificadores convolucionais. Nesse caso, utilizar codificadores convolucionais pode ser a estratégia mais apropriada (HAYKIN, 2004).

Em relação a codificadores de bloco, os codificadores convolucionais muitas vezes são preferidos na prática, pois estes fornecem um excelente desempenho de codificação / decodificação. Além disso, os códigos convolucionais estão entre os mais antigos para os quais foram desenvolvidos eficazes algoritmos de decodificação fazendo uso de decoficação suave (MORELOS-ZARAGOZA, 2002).

Considerando que os códigos de bloco utilizam blocos distintos de k símbolos, para então produzir blocos de n símbolos que dependem apenas k símbolos de entrada, os códigos convolucionais são freqüentemente vistos como códigos de fluxo, na medida em que muitas vezes operam em fluxos contínuos de símbolos não divididos em blocos de mensagens discretas. Um codificador convolucional pode ser visto como um conjunto de filtros digitais (MORELOS-ZARAGOZA, 2002).

Com um breve comparativo realizado entre codificadores de bloco e codificadores convolucionais, este trabalho agora será focado no desenvolvimento e explanação de codificadores convolucionais e o algoritmo de Viterbi.

3.1. Histórico

Os códigos convolucionais, foram introduzidos por Elias (ELIAS, 1954) e vêm sendo utilizado em inúmeras aplicações (MORELOS-ZARAGOZA, 2002), tais como:

- Sistemas de comunicação sem fio (*Wireless*);
- Sistemas de Comunicação Terreste e via Satélite;
- Sistemas de *Broadcast*.

Ainda por Elias (ELIAS, 1954) foi mostrado que pode ser introduzida uma redundância em uma cadeia de dados através de uma mudança linear no registrador.

O primeiro algoritmo para decodificação de códigos convolucionais foram descritos em 1961 por Wozencraft and Reiffen, sendo o primeiro da classe de “algoritmos sequenciais” que demonstrou bastante velocidade (WOZENCRAFT & JACOBS, 1965).

Foram então apresentados, em 1963 por Massey (MASSEY, 1963), algoritmos para decodificação tanto para os códigos convolucionais, como para códigos de bloco. Estes algoritmos foram otimizados em códigos de bloco de várias classes, porém, quando utilizados em códigos convolucionais, na maioria das vezes não tiveram melhores desempenho que os algoritmos sequenciais.

Fano (FANO, 1963) e Jelinik (JELINEK, 1969) traçaram novos algoritmos sequenciais modificados em 1963 e 1969, melhorando a performance do algoritmo de Wozencraft-Reiffen.

Logo em seguida, em 1967, surgiu Viterbi com a terceira aproximação para decodificação de códigos convolucionais.

Dois anos depois, foi mostrado por Omura (OKUMURA, 1979) que o algoritmo de Viterbi poderia ser a solução para o problema de encontrar o caminho de peso mínimo.

Fourney mostrou então, em 1973, que o algoritmo de decodificação de máxima verossimilhança para códigos convolucionais é então o já apresentado algoritmo de Viterbi.

O método de decodificação mais utilizado no momento para códigos convolucionais é o algoritmo de Viterbi (MORELOS-ZARAGOZA, 2002), que também tem outras aplicações, tais como: a detecção da sequência de máxima verossimilhança e sinalização de resposta parcial, no qual podem ser utilizadas em casos de equalização e gravação magnética, respectivamente.

3.2. Taxa de Codificação

Os códigos convolucionais binários possuem uma taxa de codificação expressa em bits por símbolo, que também pode ser visto como uma FSSM, formada por um registrador de deslocamento de m etapas, com n saídas e um multiplexador responsável por serializar as saídas do somador (HAYKIN, 2004; MOON, 2005; MOREIRA & FARREL, 2006; MORELOS-ZARAGOZA, 2002; WICKER, 1995).

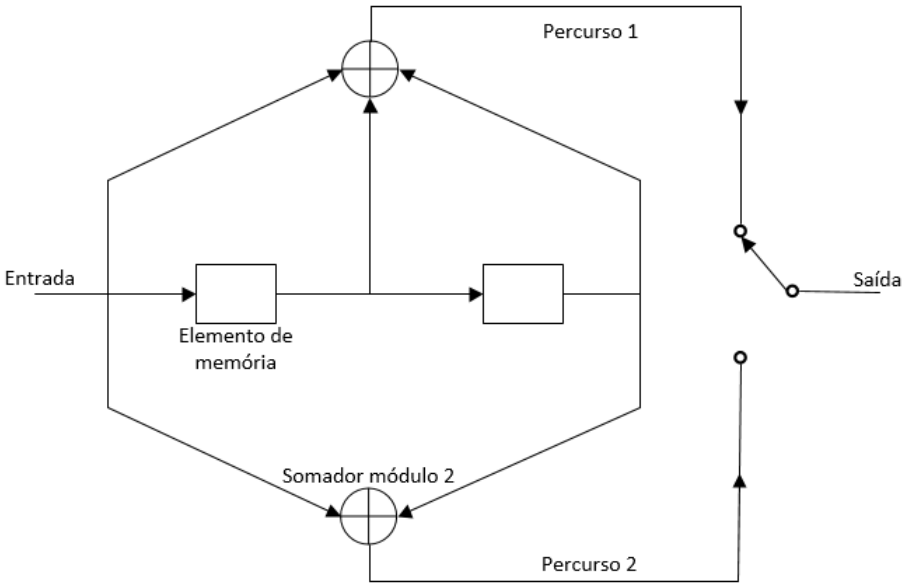
Uma sequência de saída codificada de tamanho $n(L+m)$ é gerada a partir de uma sequência de mensagem de L bits (HAYKIN, 2004), portanto, a taxa de codificação para esta sequência, é dada por:

$$R = \frac{kL}{n(L+m)} \quad bits/símbolo. \tag{3.1}$$

Como geralmente, o L é muito maior que m, logo, tem-se que:

$$R \cong \frac{k}{n} \; bits/símbolo. \tag{3.2}$$

Figura 8 – Codificador convolucional com M=3 e taxa de codificação 1/2.



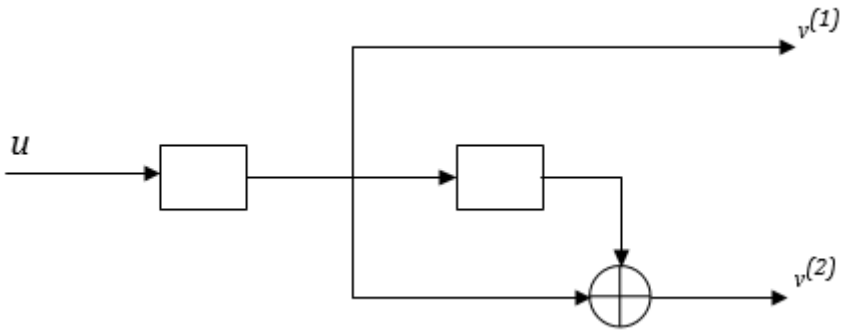
Fonte: Elaborada pelo autor (2020).

A Figura 8 mostra um codificador convolucional com n=2 e k=1. O mesmo possui uma sequência de entrada de apenas 1 bit por vez. Tem-se que a taxa de codificação para este exemplo é de ½. Este é um código não-sistemático, pois o uso deste é preferível quando se trata de códigos convolucionais. Diferentemente aos códigos de bloco, que por sua vez é preferível o uso de códigos sistemáticos (HAYKIN, 2004; LIN & COSTELLO, 1983; MORELOS-ZARAGOZA, 2002).

3.3. Codificadores não recursivos

Também conhecidos como codificadores de resposta finita ao impulso (FIR, do inglês *Finite Impulse Response*), são codificadores onde o circuito não possui nenhuma derivação de saída do registrador de deslocamento de volta na entrada (HAYKIN, 2004; LIN & COSTELLO, 1983; MOON, 2005; MORELOS-ZARAGOZA, 2002). Segue um codificador FIR na Figura 9.

Figura 9 – Codificador FIR.



Fonte: Elaborada pelo autor (2020).

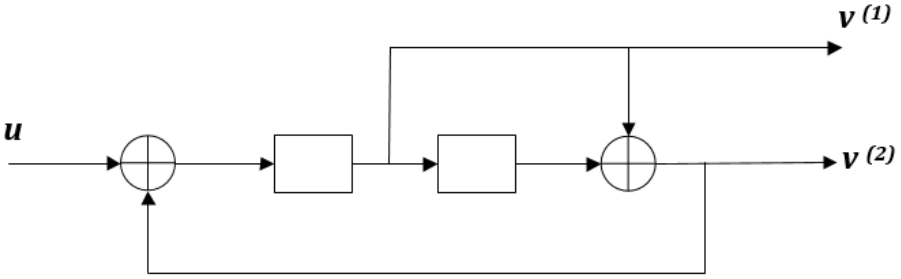
Esses codificadores são usados quando não se deseja que os estados do codificador em instantes de tempos passados interfiram nos estados atuais (HAYKIN, 2004; LIN & COSTELLO, 1983).

3.4. Codificadores recursivos

Um codificador convolucional recursivo sistemático (RSC, do inglês *Recursive Systematic Convolutional*) é caracterizado pela realimentação de pelo menos uma das derivações de saída do registrador de deslocamento de volta na entrada (HAYKIN, 2004; LIN & COSTELLO, 1983; MOON, 2005; MORELOS-ZARAGOZA, 2002).

Os codificadores RSC também são conhecidos como codificadores de resposta ao impulso infinita. Segue um codificador RSC na Figura 10.

Figura 10 – Codificador RSC.



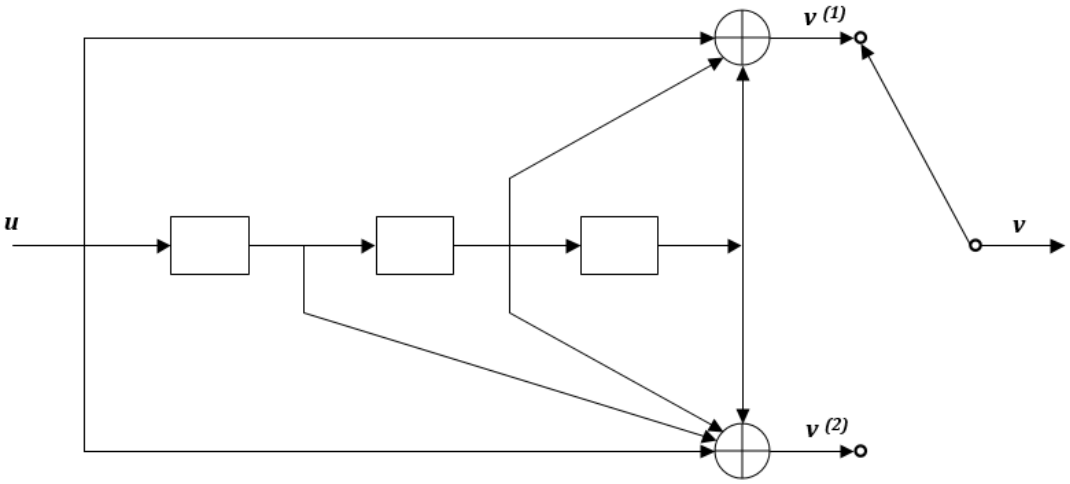
Fonte: Elaborada pelo autor (2020).

Um RSC é utilizado quando se deseja que o estado atual do registrador de deslocamento dependa de estados passados (HAYKIN, 2004; LIN & COSTELLO, 1983).

3.5. Codificação

A sequência de entrada $\boldsymbol{u}$, representada por $\boldsymbol{u} = (u_o, u_1, u_2, \dots)$ entra no codificador k bits por vez. Pode-se obter as saídas do codificador $\boldsymbol{v^{(1)}} = (v_o^{(1)}, v_1^{(1)}, v_2^{(1)}, \dots)$ e $\boldsymbol{v^{(2)}} = (v_o^{(2)}, v_1^{(2)}, v_2^{(2)}, \dots)$ com a convolução da entrada $\boldsymbol{u}$ com as duas “respostas ao impulso” do codificador (LIN & COSTELLO, 1983). Essas respostas podem ser obtidas fazendo $\boldsymbol{u}$ = (1 0 0 0....), fazendo com que este bit 1 chegue ao final do codificador, sempre analisando suas saídas $\boldsymbol{v^{(1)}}$ e $\boldsymbol{v^{(2)}}$ em cada instante de tempo. O resultado desta análise, que será realizada M vezes, são as respostas ao impulso, que são representadas como $\boldsymbol{g^{(1)}} = (g_o^{(1)}, g_1^{(1)}, g_2^{(1)}, \dots, g_m^{(1)})$ e $\boldsymbol{g^{(2)}} = (g_o^{(2)}, g_1^{(2)}, g_2^{(2)}, \dots, g_m^{(2)})$ (WICKER, 1995).

Figura 11 – Codificador com 3 elementos de memória.



Fonte: Elaborada pelo autor (2020).

A Figura 11 ilustra um codificador com M=4 e n=2. Analisando a resposta ao impulso deste codificador, esta é dada por:

$$\begin{cases} \boldsymbol{g}^{(1)} = (1\ 0\ 1\ 1), \\ \boldsymbol{g}^{(2)} = (1\ 1\ 1\ 1). \end{cases} \tag{3.3}$$

Estas expressões também são comumente chamadas de sequências geradoras.

De posse das sequências geradoras, pode-se obter as saídas $\boldsymbol{v}^{(1)}$ e $\boldsymbol{v}^{(2)}$ do codificador. Estas equações $\boldsymbol{v}^{(n)}$ podem ser descritas pela convolução de $\boldsymbol{u}$ e $\boldsymbol{g}^{(n)}$ da seguinte maneira:

$$\boldsymbol{v}^{(n)} = \boldsymbol{u} * \boldsymbol{g}^{(n)} \tag{3.4}$$

Desenvolvendo (3.3), tem-se que:

$$\boldsymbol{v_l^{(j)}} = \sum_{l=0}^{m} u_{l-i} g_i^{(j)} = u_l g_0^{(j)} + u_{l-1} g_1^{(j)} + \cdots + u_{l-m} g_m^{(j)}. \quad (3.5)$$

$$\text{Onde } \begin{cases} u_{l-i} = 0, \ l < i, \\ j = 1,2. \end{cases}$$

Logo, têm-se as seguintes expressões:

$$\boldsymbol{v_l^{(1)}} = u_l + u_{l-2} + u_{l-3}, \quad (3.6)$$

$$\boldsymbol{v_l^{(2)}} = u_l + u_{l-1} + u_{l-2} + u_{l-3}. \quad (3.7)$$

Depois de encontradas as expressões $\boldsymbol{v_l^{(1)}}$ e $\boldsymbol{v_l^{(2)}}$, as duas são concatenadas em uma única expressão, formando uma sequência única, chamada de palavra-código (PROAKIS et al., 2006), para que possam ser transmitidas através do canal. A palavra-código é dada por:

$$\boldsymbol{v} = \left(v_0^{(1)} v_0^{(2)}, v_1^{(1)} v_1^{(2)}, v_2^{(1)} v_2^{(2)}, \ldots\right). \quad (3.8)$$

A equação de codificação também pode ser expressa na forma de matricial como:

$$\boldsymbol{v} = \boldsymbol{u} G. \quad (3.9)$$

Na forma matricial, G também é chamada de Matriz Geradora do código. Percebe-se que G é uma matriz semi-infinita, explicado pelo fato que a sequência de informação tem tamanho arbitrário, ou seja, se $\boldsymbol{u}$ tem tamanho L, G terá L linhas e $2(m + L)$ colunas. Consequentemente, $\boldsymbol{v}$ terá o tamanho igual ao número de colunas da matriz G, que neste caso seria $2(m + L)$ (LIN & COSTELLO, 1983; MOREIRA & FARREL, 2006).

Intercalando as sequências geradoras, tem-se a matriz G, que é dada por:

$$G = \begin{bmatrix} g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & g_2^{(1)}g_2^{(2)} & \cdots & g_m^{(1)}g_m^{(2)} & & \\ & g_0^{(1)}g_0^{(2)} & g_1^{(1)}g_1^{(2)} & \cdots & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} & \\ & & g_0^{(1)}g_0^{(2)} & \cdots & g_{m-2}^{(1)}g_{m-2}^{(2)} & g_{m-1}^{(1)}g_{m-1}^{(2)} & g_m^{(1)}g_m^{(2)} \\ & & \ddots & & & & \ddots \end{bmatrix} \quad (3.10)$$

Ou ainda representada, no caso geral por:

$$G = \begin{bmatrix} G_0 & G_1 & G_2 & \cdots & G_m & & \\ & G_0 & G_1 & \cdots & G_{m-1} & G_m & \\ & & G_0 & \cdots & G_{m-2} & G_{m-1} & G_m \\ & & \ddots & & & & \ddots \end{bmatrix} \quad (3.11)$$

Outra forma de representar estes sistemas envolvendo convolução é por expressões polinomiais, desde que o codificador utilizado seja um sistema linear (LIN & COSTELLO, 1983). Cada umas das sequências $\boldsymbol{v^{(1)}}$ e $\boldsymbol{v^{(2)}}$ podem ser reescritas na forma polinomial, e alternando a operação de convolução, pela operação de multiplicação. Na forma polinomial, a sequência é representada por:

$$v^{(1)}(D) = u(D)g^{(1)}(D), \quad (3.12)$$

$$v^{(2)}(D) = u(D)g^{(2)}(D), \quad (3.13)$$

em que,

$$u(D) = u_0 + u_1 D + u_2 D^2 + \cdots, \quad (3.14)$$

que, por sua vez

$$v^{(1)}(D) = v_0^{(1)} + v_1^{(1)} D + v_2^{(1)} D^2 + \cdots, \quad (3.15)$$

$$v^{(2)}(D) = v_0^{(2)} + v_1^{(2)} D + v_2^{(2)} D^2 + \cdots. \quad (3.16)$$

Também é possível representar as sequências geradoras na forma polinomial, da seguinte maneira:

$$g^{(1)}(D) = g_0^{(1)} + g_1^{(1)}D + \cdots + g_m^{(1)}D^m \ , \tag{3.17}$$

$$g^{(2)}(D) = g_0^{(2)} + g_1^{(2)}D + \cdots + g_m^{(2)}D^m \ . \tag{3.18}$$

Neste formato, estas sequências são chamadas de “sequências geradoras polinomiais”.

Por fim, será realizada a multiplexação, para obtermos a palavra-código e poder transmiti-la pelo canal:

$$v(D) = v^{(1)}(D^2) + Dv^{(2)}(D^2). \tag{3.19}$$

3.6. Diagrama de Estados

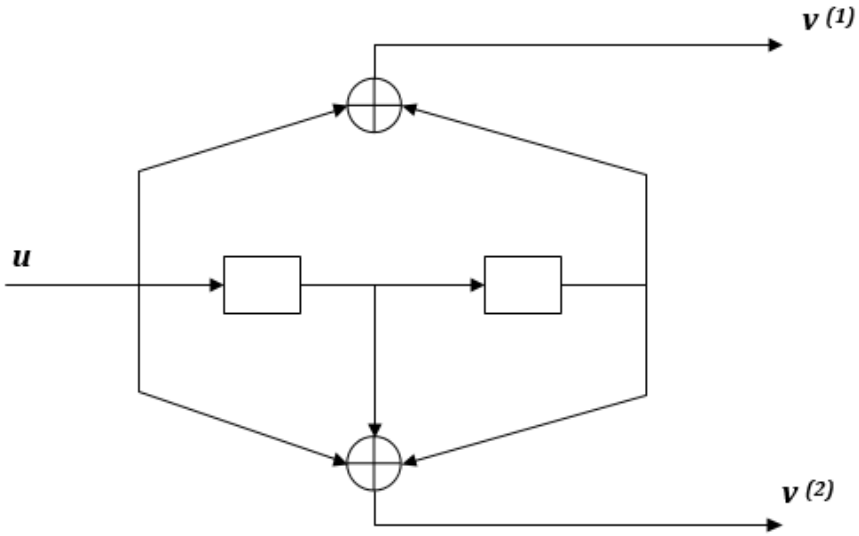
Para que seja possível descrever um diagrama de estados, é necessário que o codificador seja um circuito sequencial, ou seja, o estado do codificador deve mudar junto com o conteúdo de cada registrador da FSSM (HAYKIN, 2004; MORELOS-ZARAGOZA, 2002).

Quando as entradas do codificador são $\boldsymbol{u} = (u^{(1)}, u^{(2)}, \ldots, u^{(k)})$, para um codificador de (n, k, m) com k>1, existe um total de 2^k estados, em que o i-ésimo registrador da FSSM contém k_i bits de informação anteriores. Ainda para essa situação, a memória total do decodificar é representada por $\sum_{i=1}^{k} k_i$ (LIN & COSTELLO, 1983).

Em outras palavras, o conteúdo presente nos registradores da FSSM em um dado instante de tempo representa o estado no qual a FSSM se encontra. O próximo estado é determinado pela inserção do próximo bit da sequência de entrada (serial) na FSSM, e pelo deslocamento para a direita dos bits alocados em cada registrador, formando um novo arranjo de bits nos registradores, ou seja, um novo estado.

Cada estado depende do estado anterior dos registradores e da entrada no instante de tempo atual. A FSSM da Figura 12 serve de exemplo para a construção do diagrama de estado da Figura 13.

Figura 12 – Codificador convolucional.



Fonte: Elaborada pelo autor (2020).

A Tabela 1 mostra o que acontece com cada estado analisado, permutando a entrada nos valores 0 e 1. A coluna do meio mostra Entrada / Saída $\boldsymbol{v}^{(1)}$, Saída $\boldsymbol{v}^{(2)}$ (E/SS), de tal forma que no estado 00 do registrador, para uma entrada $\boldsymbol{u}$ igual a 0, o próximo estado dessa FSSM será novamente o estado 00, e assim por diante.

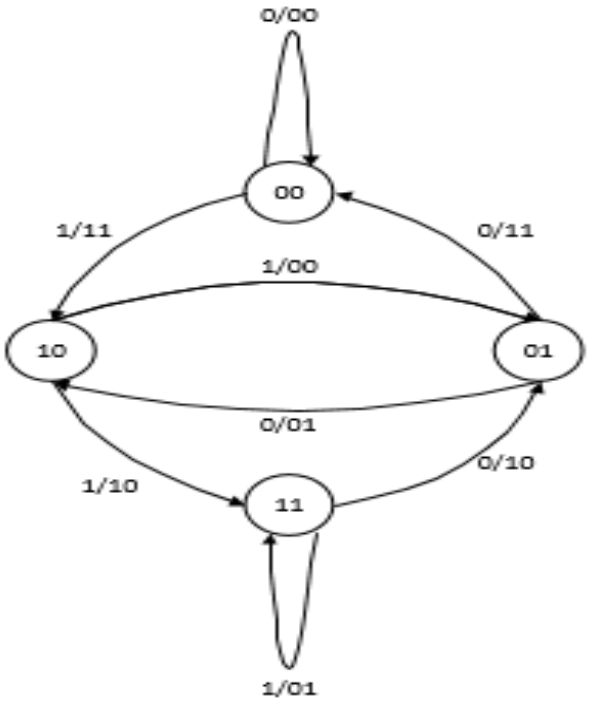
Tabela 1 – Tabela de Estados.

ESTADO ATUAL	E/SS	PRÓXIMO ESTADO
00	0/00	00
00	1/11	10
01	0/11	00

01	1/00	10
10	0/01	01
10	1/10	11
11	0/10	01
11	1/01	11

De posse destas informações, basta criar os estados, que normalmente são utilizados “círculos” para representá-los. Logo em seguida, basta usar as informações da Tabela 1 e interligar os estados, sempre observando o próximo estado conforme a entrada inserida, ilustrado na Figura 13.

Figura 13 – Elaboração do diagrama de estados.



Fonte: Elaborada pelo autor (2020).

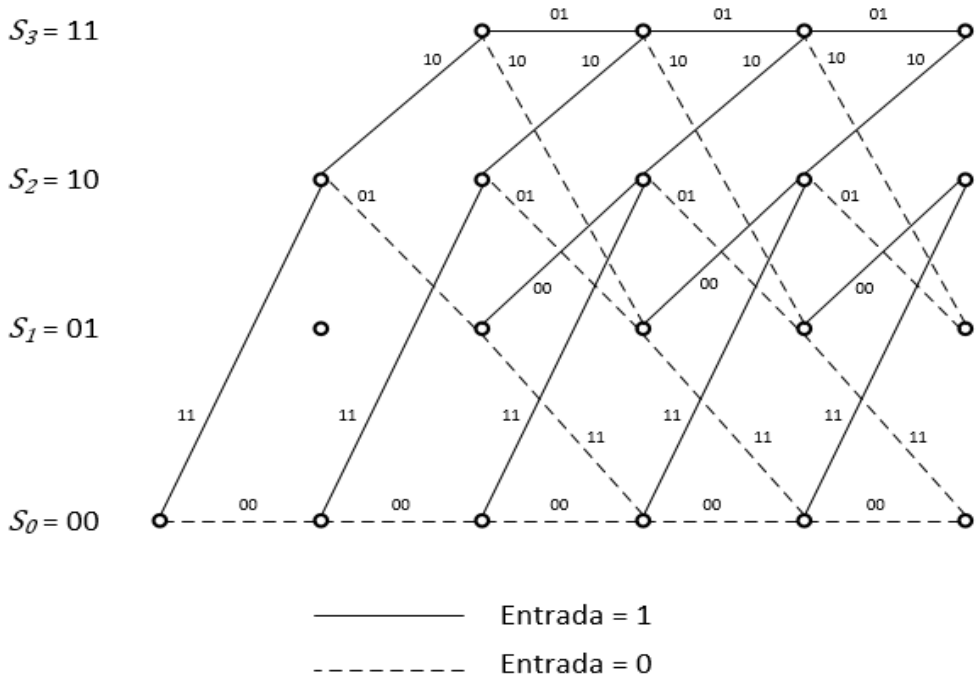
Vale salientar que se faz necessário a notação E/SS no diagrama de estados, mostrada na Tabela 1, em cada interligação de estados, para identificar as entradas e saídas para cada caso analisado em particular.

3.7. Diagrama de Treliça

A fim de obter uma estrutura dinâmica, e facilitar a visualização de cada mudança de estado, os codificadores convolucionais podem ser representados através de mudanças de instantes de tempo, na forma de diagrama de treliça (SKLAR, 1993; LIN & COSTELLO, 1983; BERROU, GLAVIEUX, & THITIMAJSHIMA, 1993; VITERBI, & OMURA, 1979). Este diagrama é constituído com base no diagrama de estados do codificador no instante atual $(t = i)$ utilizando ramos para interligar os próximos estados em instantes de tempo posteriores $(t = i + 1)$. Os ramos de interligação dos estados do diagrama de treliça são rotulados com as saídas da FSSM de acordo com a mudança de estados prevista no diagrama de estados (MOREIRA & FARREL, 2006).

No codificador da Figura 12, para a construção do diagrama de treliça, considera-se o estado inicial do codificador como S_0, e partir daí, o diagrama de estados ilustrado Figura 13 será usado, ou ainda se preferível, a Tabela 1 que originou o diagrama de estados. O ponto de partida será o estado S_0, com base no diagrama de estados do codificador, verifica-se o próximo estado, criando um ramo de ligação ao próximo estado $(t = i + 1)$, de maneira análoga a construção do diagrama de estados, sempre lembrando de representar este ramo com as saídas visualizadas no diagrama de estados (MOREIRA & FARREL, 2006). À medida que novos estados forem tocados por ramos de ligação, os mesmo servirão de estados atuais, para que sejam analisados seus próximos estados, e também interligados com ramos, conforme o diagrama de estados (MOREIRA & FARREL, 2006). Segue na Figura 14 o diagrama de treliça do codificador ilustrado na Figura 12.

Figura 14 – Elaboração do diagrama de treliça.



Fonte: Elaborada pelo autor (2020).

4. Algoritmo de Viterbi

O algoritmo de Viterbi foi proposto por Andrew Viterbi (VITERBI, 1967). Em 1973 foi constatado que este algoritmo calcula a sequência de máxima verossimilhança dos bits recebidos, e representa o caminho mais curto através do diagrama de treliça (HAYKIN, 2004; MORELOS-ZARAGOZA, 2002; WICKER, 1995).

Admita uma sequência de informação $\boldsymbol{u} = (u_0, u_1, \ldots u_{L-1},)$ de comprimento kL. Ao passar pelo codificador, esta sequência se torna na palavra-código $\boldsymbol{v} = (v_0, v_1, \ldots v_{L+m-1},)$ com comprimento $N = n(L + m)$. Porém, a mensagem recebida na saída de um canal discreto sem memória (DMC) da FSSM é $\boldsymbol{r} = (r_0, r_1, \ldots r_{L+m-1},)$. Outra forma de escrever estas mesmas sequências é: $\boldsymbol{u} = (u_0, u_1, \ldots u_{kL-1},)$, $\boldsymbol{v} = (v_0, v_1, \ldots v_{N-1},)$ e $\boldsymbol{r} = (r_0, r_1, \ldots r_{N-1},)$ (LIN & COSTELLO, 1983). O decodificador deve estimar uma sequência para a palavra-código $\hat{\boldsymbol{v}}$ de acordo com sequência recebida $\boldsymbol{r}$. Então, o decodificador de máxima verossimilhança determina $\hat{\boldsymbol{v}}$ como sendo a palavra-código $\boldsymbol{v}$, maximizando a função de log-verossimilhança $\log P(\boldsymbol{r}|\boldsymbol{v})$.

Para um DMC, tem-se:

$$P(\boldsymbol{r}|\boldsymbol{v}) = \prod_{i=0}^{L+m-1} P(r_i|v_i) = \prod_{i=0}^{N-1} P(r_i|v_i). \tag{4.1}$$

Logo,

$$log\, P(\boldsymbol{r}|\boldsymbol{v}) = \sum_{i=0}^{L+m-1} logP(r_i|v_i) = \sum_{i=0}^{N-1} logP(r_i|v_i). \tag{4.2}$$

Onde:

$P(r_i|v_i)$ representa uma probabilidade de transição de canal.

A função de log-verossimilhança $log\ P(\boldsymbol{r}|\boldsymbol{v})$ é também chamada de métrica, em relação ao caminho $\boldsymbol{v}$, e pode ser expressa como $\mathrm{M}(\boldsymbol{r}|\boldsymbol{v})$. Os termos $log\ P(\boldsymbol{r_i}|\boldsymbol{v_i})$ representam as métricas de cada ramo, e podem ser expressas por $\mathrm{M}(\boldsymbol{r_i}|\boldsymbol{v_i})$. Já no caso dos termos $log\ P(r_i|v_i)$, são chamados de métrica dos bits, ou também expressos por $\mathrm{M}(r_i|v_i)$ (HAYKIN, 2004; PROAKIS et al., 2006). Logo, de maneira análoga, a métrica do caminho pode ser expressa por:

$$\mathrm{M}(\boldsymbol{r}|\boldsymbol{v}) = \sum_{i=0}^{L+m-1} \mathrm{M}(\boldsymbol{r_i}|\boldsymbol{v_i}) = \sum_{i=0}^{N-1} \mathrm{M}(r_i|v_i)\,. \tag{4.3}$$

Em um caso particular, analisando a métrica dos primeiros *j* ramos, conclui-se que:

$$\mathrm{M}([\boldsymbol{r}|\boldsymbol{v}]_j) = \sum_{i=0}^{j-1} \mathrm{M}(\boldsymbol{r_i}|\boldsymbol{v_i}). \tag{4.4}$$

O algoritmo de Viterbi, quando aplicado a um DMC de sequência recebida $\boldsymbol{r}$, encontra o caminho através do conceito de maior métrica pelo diagrama de treliça, ou seja, caminho de máxima verossimilhança. O algoritmo processa $\boldsymbol{r}$ através de iterações. Em cada iteração, o sistema compara o valor referente a métrica de cada um dos caminhos, e armazena apenas a informação do caminho que contém a maior métrica, que junto com sua métrica, é chamado de *sobrevivente* (HAYKIN, 2004; LIN & COSTELLO, 1983; MORELOS-ZARAGOZA, 2002). Segue abaixo o passo-a-passo realizado pelo algoritmo:

1º Passo: O tempo inicial é $j = m$, e é realizado o cálculo das métricas parciais de cada caminho que chega em cada estado. Então é aplicada pelo algoritmo a métrica *sobrevivente* em cada estado.

2º Passo: O valor de tempo j sofre incrementos de 1 unidade, e novamente é realizado o cálculo das métricas parciais de cada caminho que chega em cada estado. Logo após a métrica do *sobrevivente* é novamente utilizada, fazendo com que todos os caminhos (exceto o *sobrevivente*) sejam descartados.

3º Passo: Se $j > L + m$, repete-se o 2º passo, caso contrário, é o final do algoritmo.

Haverá 2^k sobreviventes, um para cada um dos 2^k estados. Após um instante de tempo L, haverá menos sobreviventes, pois haverá menos estados, e com isso o codificador retornará ao seu estado inicial.

Em se tratando de probabilidades, é mais palpável que esses números sejam inteiros positivos. Portanto, em termos de praticidade, podem-se representar essas métricas reais como métricas quem utilizem números inteiros positivos, tomando os devidos cuidados para que isto não afete o resultado ou desempenho do algoritmo.

A métrica de bit $M(r_i|v_i) = logP(r_i|v_i)$ pode então ser expressa por $c_2[logP(r_i|v_i) + c_1]$, onde c_1 pode ser qualquer número e c_2 pode ser qualquer número real positivo, ou seja, um caminho $\boldsymbol{v}$ capaz de maximizar (HAYKIN, 2004; LIN & COSTELLO, 1983; MOON, 2005):

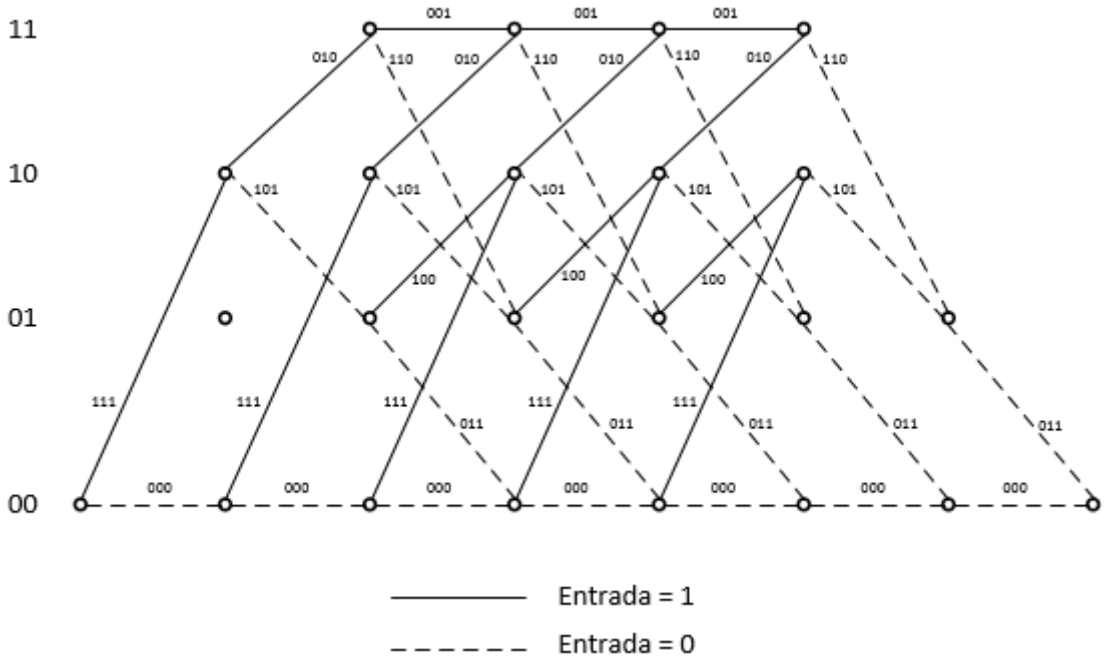
$$\mathrm{M}(\boldsymbol{r}|\boldsymbol{v}) = \sum_{i=0}^{N-1} M(r_\mathrm{i}|v_\mathrm{i}) = \sum_{i=0}^{N-1} \log P\,(r_i|v_i)\ . \tag{4.5}$$

Funcionará de maneira análoga, maximizando também, uma maneira específica para um canal DMC é:

$$\sum_{i=0}^{N-1} c_2\{\log[P(r_i|v_i) + c_1]\}. \tag{4.6}$$

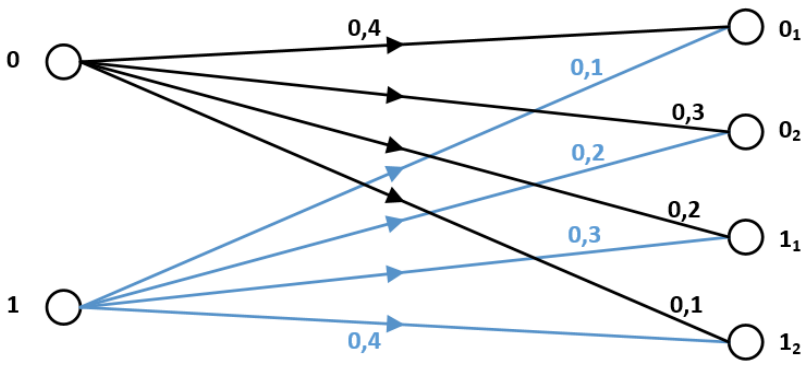
Considere o exemplo de canal DMC, com o diagrama de treliça da Figura 15 e a distribuição de probabilidades da Figura 16.

Figura 15 – Diagrama de treliça para um Canal DMC.



Fonte: Elaborada pelo autor (2020).

Figura 16 – Distribuição de probabilidades.



Fonte: Elaborada pelo autor (2020).

Admitindo que a sequência recebida seja:

$$\boldsymbol{r} = (1_1 1_2 0_1, 1_1 1_1 0_2, 1_1 1_1 0_1, 1_1 1_1 1_1, 0_1 1_2 0_1, 1_2 0_2 1_1, 1_2 0_1 1_1). \qquad (4.7)$$

A Tabela 2 ilustra o esquema referente ao exemplo da Figura 16.

Tabela 2 – Tabela das distribuições de probabilidades dadas.

v_i \ r_i	0_1	0_2	1_2	1_1
0	-0,4	-0,52	-0,7	-1
1	-1	-0,7	-0,52	-0,4

Escolhendo $c_1 = 2$ e $c_2 = 100$, e aproximando todas essas probabilidade para números inteiros foi gerada a Tabela 3.

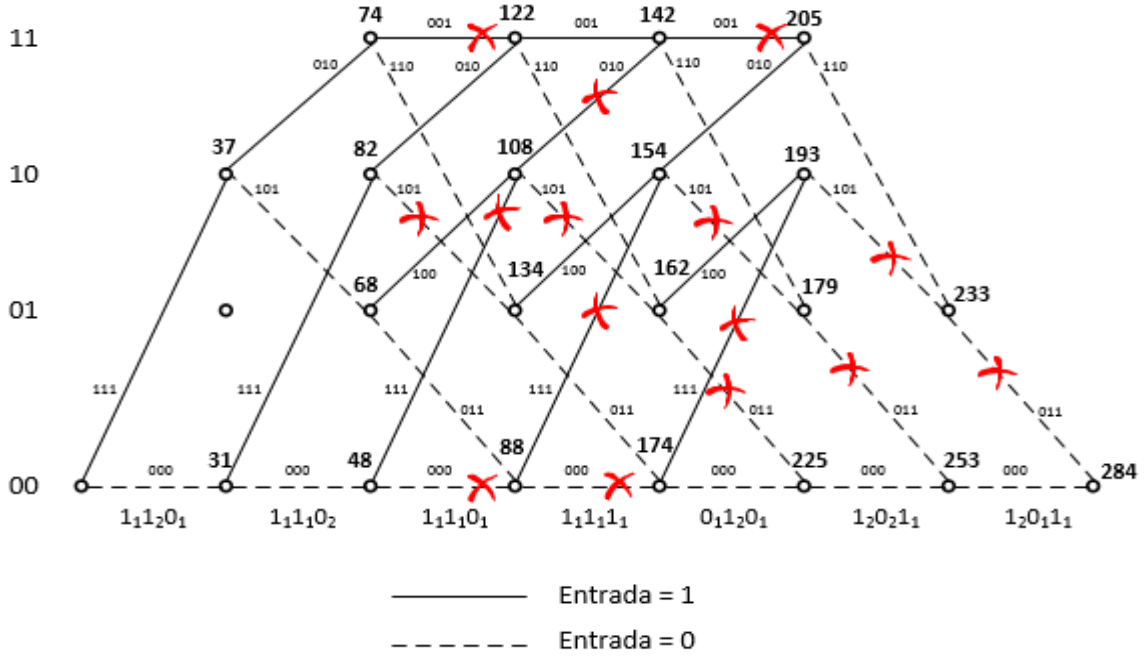
Tabela 3 – Tabela das distribuições de probabilidades modificadas.

v_i \ r_i	0_1	0_2	1_2	1_1
0	20	17	11	0
1	0	11	17	20

Os números presentes nas Tabelas 2 e 3 mostram a métrica de sobrevivente para cada estado. A eliminação dos caminhos com a métrica de menor probabilidade será realizada no diagrama de treliça.

A Figura 17 representa o diagrama de treliça e as respectivas métricas de cada estado.

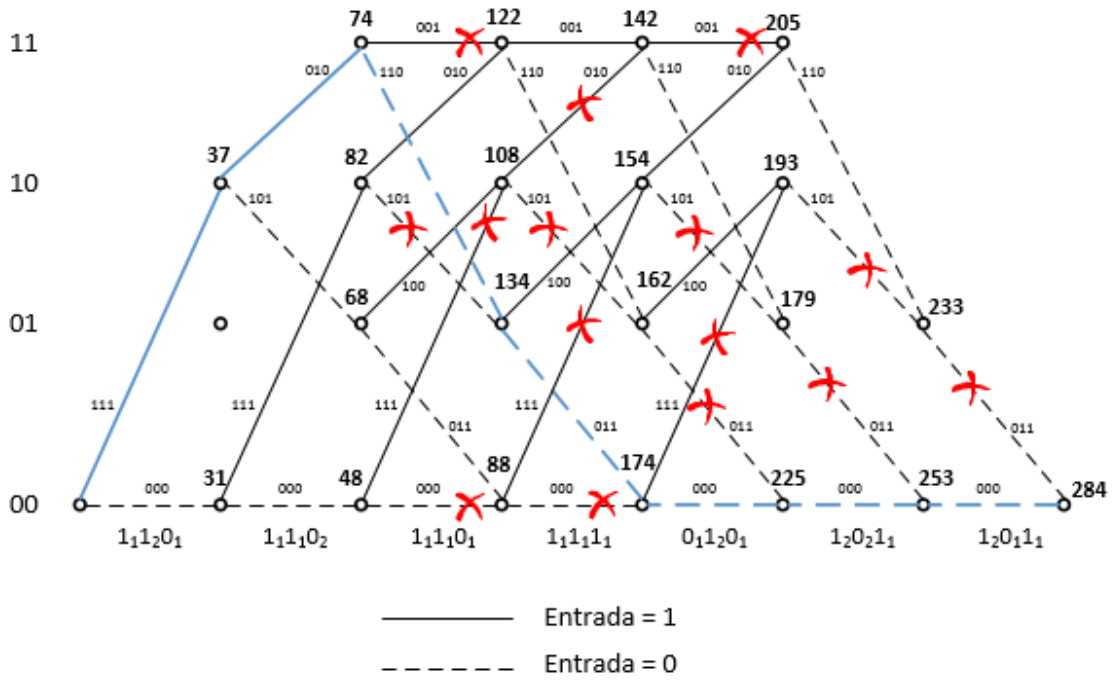
Figura 17 – Desenvolvimento do diagrama de treliça de um DMC.



Fonte: Elaborada pelo autor (2020).

Note que é utilizada a maior métrica para determinação do caminho sobrevivente, conforme Figura 18.

Figura 18 – Diagrama de treliça final para um DMC.



Fonte: Elaborada pelo autor (2020).

Observe que o caminho em azul foi determinado pelas maiores métricas (probabilidades) em cada estado, formando assim um caminho único. Este caminho foi estabelecido após a análise em cada instante de tempo e, de acordo com a métrica calculada em cada nó, o caminho de menor métrica é desprezado. Por fim, é analisado o caminho partindo do final do diagrama de treliça, onde só terá um único caminho para que, com isso, seja possível estimar o sobrevivente final.

$$\hat{\boldsymbol{v}} = (111, 010, 110, 011, 000, 000, 000).$$

No caso particular do BSC com probabilidade de transição $p < 1/2$, como a sequência recebida $\boldsymbol{r}$ é binária, tem-se que:

$$logP(\boldsymbol{r}|\boldsymbol{v}) = d(\boldsymbol{r}, \boldsymbol{v}) log \frac{p}{1-p} + N log(1-p). \qquad (4.8)$$

Em que:

$d(\boldsymbol{r}|\boldsymbol{v})$ representa a distância de Hamming entre $\boldsymbol{r}$ e $\boldsymbol{v}$.

Sendo $log\frac{p}{1-p} < 0$, desde que $Nlog(1-p)$ (PROAKIS et al., 2006) seja constante para todo $\boldsymbol{v}$, a determinação da máxima verossimilhança determina $\boldsymbol{v}$ como palavra-código $\boldsymbol{\hat{v}}$, minimizando a distância de Hamming para um canal BSC:

$$d(\boldsymbol{r},\boldsymbol{v}) = \sum_{i=0}^{L+m-1} d(\boldsymbol{r_i},\boldsymbol{v_i}) = \sum_{i=0}^{N-1} d(r_i,v_i). \qquad (4.9)$$

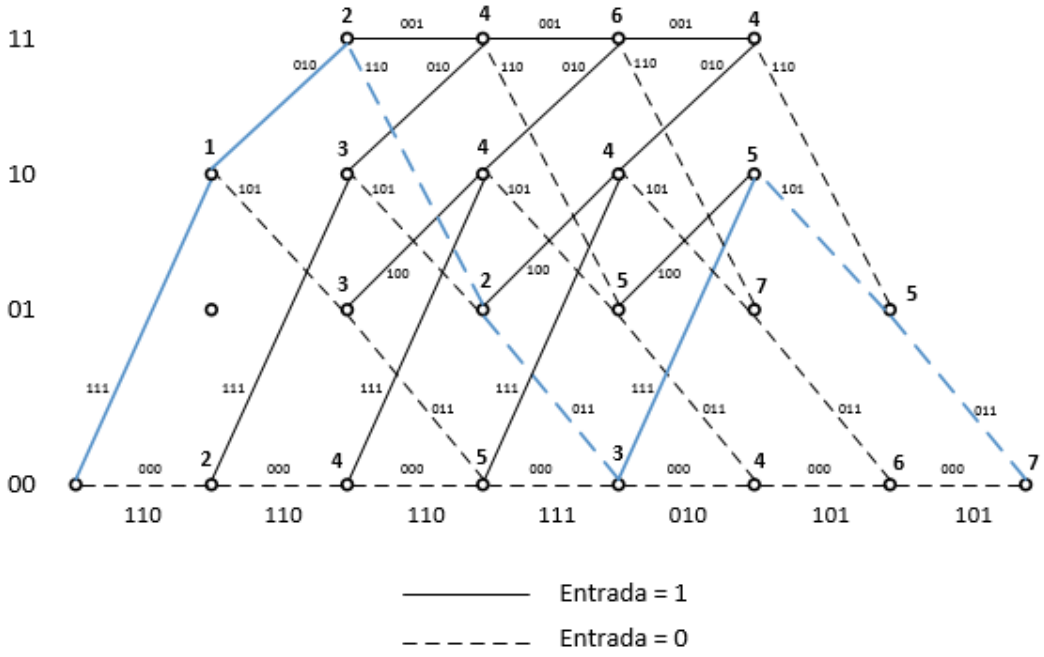
Basicamente, o algoritmo de Viterbi para um canal BSC difere no aspecto de que, ao invés de ser utilizada a função de log-verossimilhança como métrica, é utilizada a distância de Hamming, desta vez analisada pela menor métrica (HAYKIN, 2004; LIN & COSTELLO, 1983; MOON, 2005; MOREIRA & FARREL, 2006; MORELOS-ZARAGOZA, 2002; WICKER, 1995), sendo escolhida a menor métrica em cada nó, em relação ao mesmo intervalo de tempo.

Supondo a seguinte sequência recebida:

$$\boldsymbol{r} = (110, 110, 110, 111, 010, 101, 101).$$

Considere o mesmo diagrama de treliça da Figura 15 como exemplo, porém com a utilização do método da distância de Hamming, conforme Figura 19.

Figura 19 – Diagrama de treliça para um canal BSC.



Fonte: Elaborada pelo autor (2020).

Onde o caminho em azul mostra o sobrevivente final, tendo como resultado:

$$\hat{\boldsymbol{v}} = (111, 010, 110, 011, 111, 101, 011).$$

5. SOVA

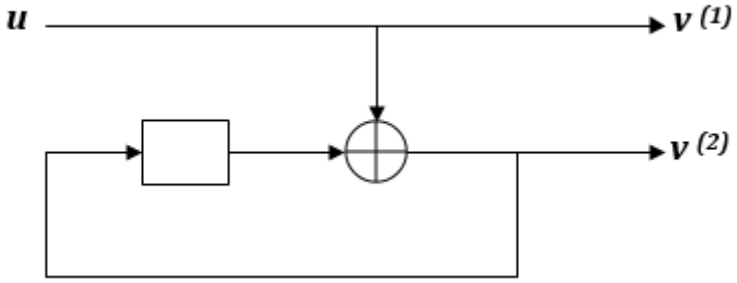
Há uma outra métrica que pode ser utilizada no Algoritmo de Viterbi, que é capaz de tornar a mensagem estimada ainda mais próxima da mensagem enviada, quando comparada com a métrica explicada anteriormente utilizando o código de Hamming. Essa métrica é chamada de distância Euclidiana quadrática, que é utilizada no Algoritmo de Viterbi de Decisão Suave (SOVA, do inglês *Soft Output Viterbi Algorithm*). (HAGENAUER & HOEBER, 1989; LI & VUCETIC, 1995; LI, VUCETIC & SATO, 1995; VITERBI, 1967; VUCETIC, 1997; VUCETIC & YUAN, 2000).

Considere o codificador da Figura 20, que ilustra um codificador RSC. Os procedimentos para criação do diagrama de estados e diagrama de treliça utilizam a mesma metodologia explicada no capítulo 4, conforme Figura 21 e Figura 22 respectivamente.

Suponha que para este circuito, a sequência recebida seja:

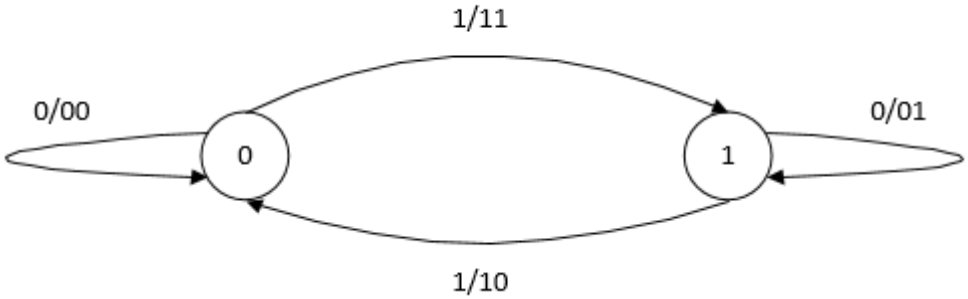
$$\boldsymbol{r}_1^4 = [(1\ 1), (-1\ 1)\ , (0{,}8\ -1), (-1\ -1)].$$

Figura 20 – Codificador RSC.



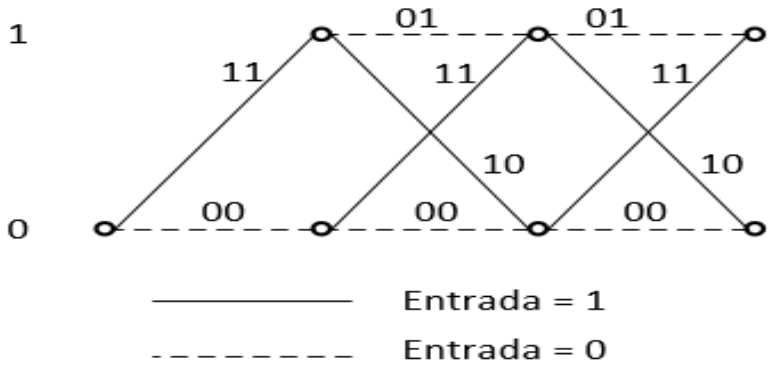
Fonte: Elaborada pelo autor (2020).

Figura 21 – Diagrama de estados de um Cconv (2,1,1) RSC.



Fonte: Elaborada pelo autor (2020).

Figura 22 – Diagrama de treliça de um Cconv (2,1,1) RSC.



Fonte: Elaborada pelo autor (2020).

Com os diagramas de estado e treliça desenvolvidos, é possível calcular os valores das métricas em cada ramo individualmente para o SOVA, com a seguinte expressão (VUCETIC & YUAN, 2000):

$$v_t^X = \sum_{i=0}^{n-1} (r_{t,i} - x_{t,i})^2. \qquad (5.1)$$

Onde:

$r_{t,i}$: sequência recebida

$\boldsymbol{x}_{t,i}$: sequência modulada no diagrama de treliça. Quando o $\boldsymbol{x}_{t,i}$ for 0, será modulado para -1 no cálculo das métricas.

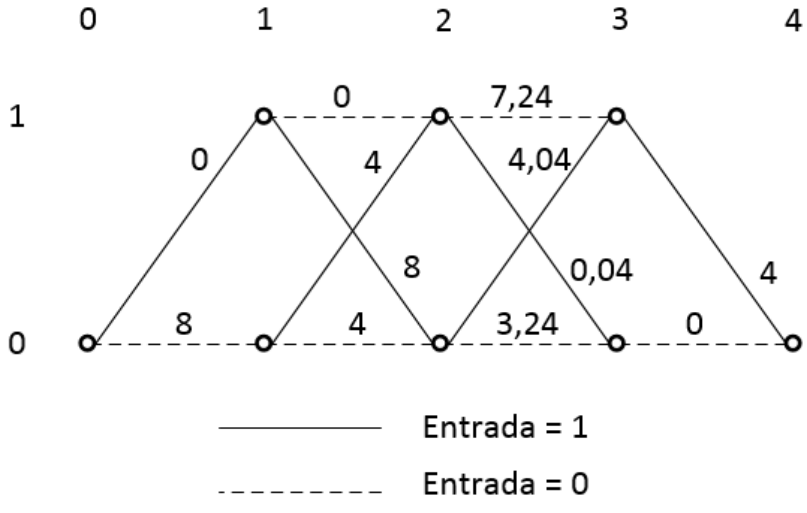
Neste exemplo, com base no $\boldsymbol{r}$ dado e sabendo as saídas de cada ramo de acordo com a Figura 22, podem ser calculadas as métricas do caminho de ida do algoritmo. Tem-se que os resultados das métricas dos dois primeiros ramos são:

$$[1-(1)]^2+[1-(1)]^2=0$$

$$[1-(-1)]^2+[1-(-1)]^2=8$$

Onde os termos entre parênteses representam a saída de cada ramo de acordo com a entrada dada (lembrando que as saídas em 0 serão interpretadas pelo algoritmo como -1). Esse procedimento será repetido em cada um dos ramos e, após a realização deste, obtêm-se todas as métricas de cada ramo do diagrama de treliça, conforme ilustrado na Figura 23.

Figura 23 – Métricas dos ramos.

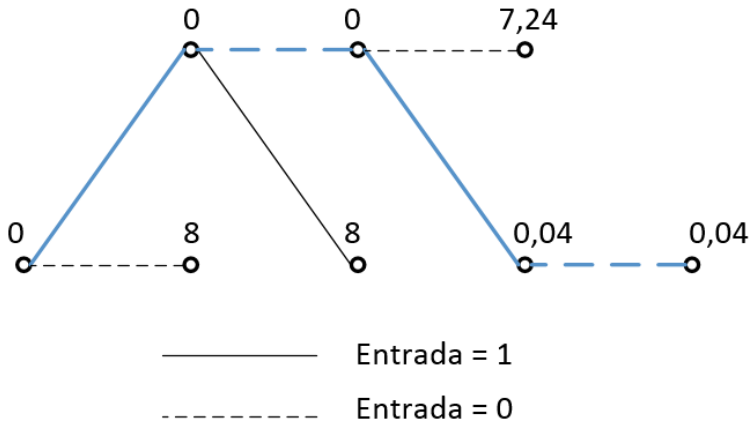


Fonte: Elaborada pelo autor (2020).

Após o desenvolvimento do diagrama da Figura 23, é necessário verificar qual a maior métrica em cada instante de tempo e eliminar o caminho que dá continuidade

a este nó com maior métrica, fazendo prevalecer apenas os caminhos com menores métricas, que estão em azul conforme Figura 24.

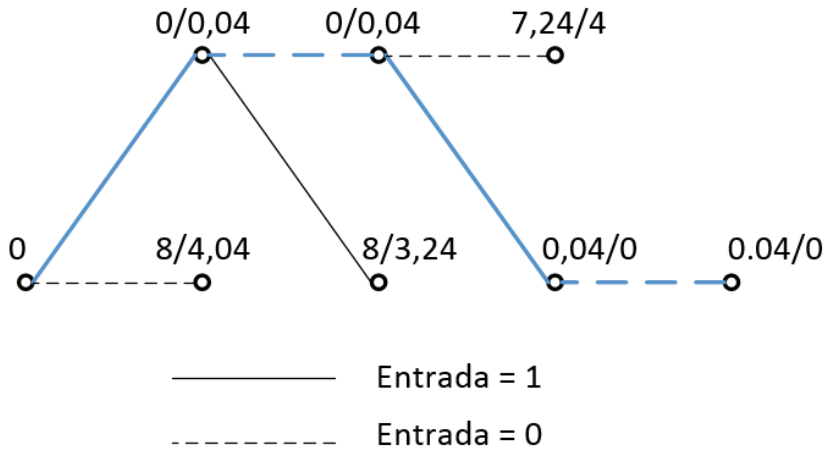
Figura 24 – Caminho de ida.



Fonte: Elaborada pelo autor (2020).

O SOVA calcula também as métricas para o caminho de volta. Essas métricas são calculadas com o mesmo procedimento do caminho de ida. Após o cálculo das métricas do caminho reverso, o diagrama de treliça ficará de acordo com a Figura 25, onde cada nó tem a representação das 2 métricas expressas no formato "*métrica do caminho de ida / métrica do caminho de volta*".

Figura 25 – Caminho de volta.



Fonte: Elaborada pelo autor (2020).

Na etapa final do algoritmo, é necessário calcular a relação de log-verossimilhança, que é dada por:

$$\Lambda(u_t) = \mu_t^0 - \mu_t^1. \tag{5.2}$$

Em que:

μ_t^0: representa a métrica no instante de tempo t, devido a entrada de um bit 0;

μ_t^1: representa a métrica no instante de tempo t, devido a entrada de um bit 1;

O algoritmo estima o bit da seguinte maneira:

$$u_t = \begin{cases} 1 & se\ \Lambda(u_t) \geq 0, \\ 0 & caso\ contrário. \end{cases}$$

Como exemplo, o cálculo de tempo no instante de tempo 1 é realizado da seguinte maneira:

$\mu_1^0 = 8 + 4{,}04 = 12{,}04$

$\mu_1^1 = 0 + 0{,}04 = 0{,}04$

$\Lambda(u_1) = 12{,}04 - 0{,}04 = 12$

Como o valor de $\Lambda(u_1)$ foi um número ≥ 0, o bit estimado é 1.

Já para o instante de tempo 2, tem-se:

$\mu_2^0 = 0 + 0{,}04 = 0{,}04$

$\mu_2^1 = 8 + 3{,}24 = 11{,}24$

$\Lambda(u_2) = 0{,}04 - 11{,}24 = -11{,}2$

Como o valor de $\Lambda(u_2)$ não foi um número ≥ 0, o bit estimado é 0.

Por analogia, tem-se que:

$\Lambda(u_3) = 11{,}2$

$\Lambda(u_4) = -11{,}2$

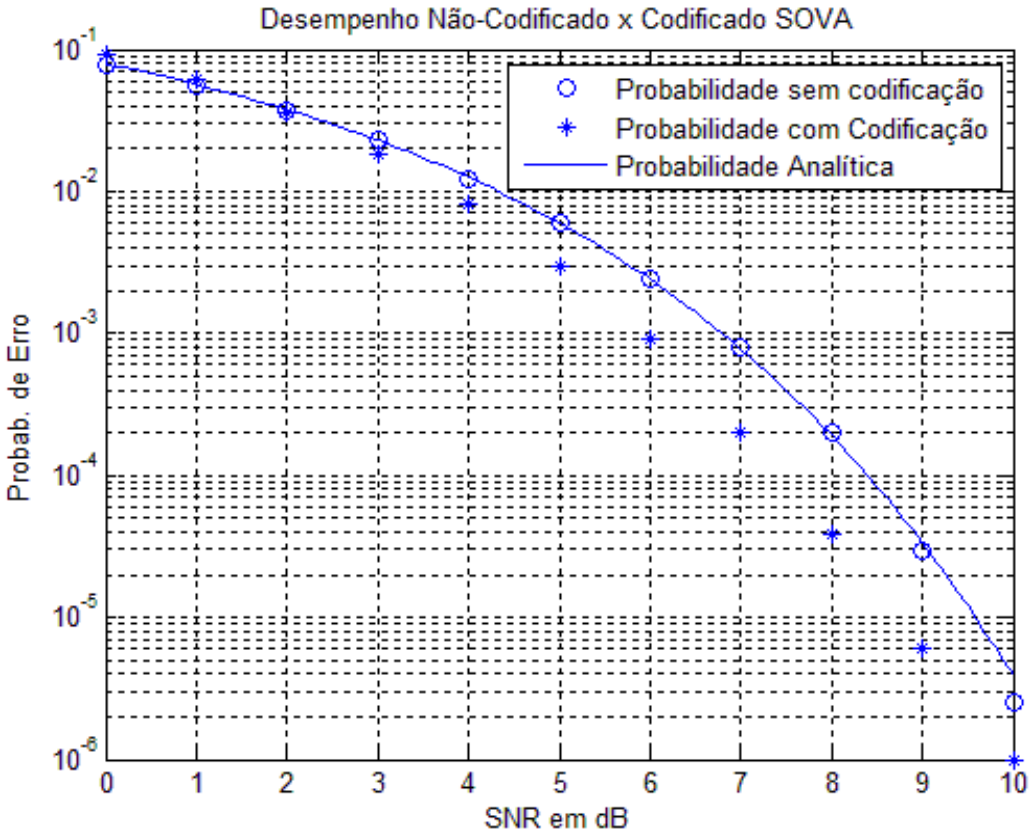
O caminho de ida do SOVA possui complexidade computacional muito similar ao algoritmo de Viterbi, no entanto, considerando apenas o caminho de volta (recursivo), geralmente a complexidade computacional é menor quando comparado ao algoritmo de Viterbi. A capacidade computacional total (caminha de ida + caminho de volta) do SOVA é sempre superior ao algoritmo de Viterbi, porém no máximo 2 vezes a complexidade do algoritmo de Viterbi. Geralmente para entradas binárias, a complexidade do SOVA fica em torno de 1,5 vezes em relação a complexidade do algoritmo de Viterbi (VUCETIC & YUAN, 2000).

6. Resultados

Este capítulo é reservado para mostrar resultados de simulações da teoria abordada durante todo o trabalho. Para ambas as simulações, foi utilizado o software Scilab.

Na primeira simulação são utilizados canal AWGN e modulação BPSK. O objetivo principal da simulação é fazer o comparativo de como se comporta o sistema de comunicação sem codificação e com codificação. O algoritmo escolhido para o sistema com codificação foi o SOVA, e para este caso é utilizado para simulação o circuito da Figura 20, que é um código convolucional RSC Cconv (2,1,1). O resultado obtido é mostrado no Gráfico 2.

Gráfico 2 – Análise de desempenho de um Cconv (2,1,1).

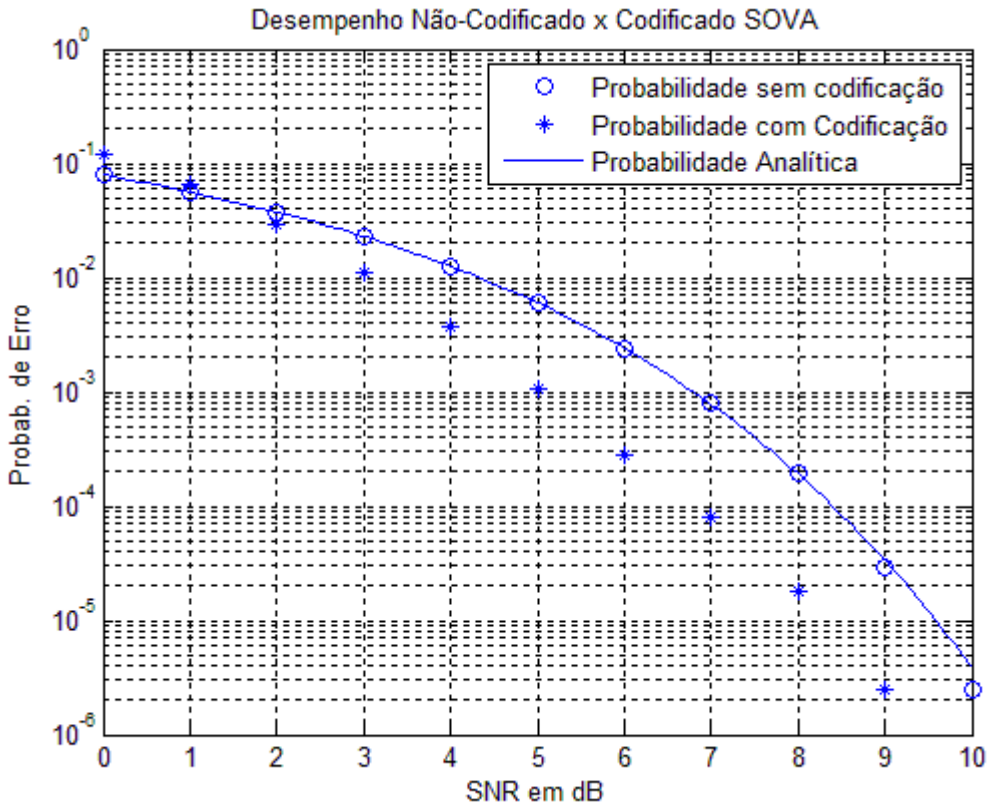


Fonte: Elaborada pelo autor (2020).

O Gráfico 2 representa uma análise de desempenho relacionando BER em função de SNR. Analisando este gráfico e escolhendo um ponto ao acaso, como por exemplo o SNR = 8dB, nota-se que para este valor de SNR, o sistema sem codificação apresenta BER=$1x10^{-4}$ contra BER=$3x10^{-5}$ em um sistema com codificação, o que significa uma redução cerca de 3 vezes na taxa de erros.

A fim de comprovar a eficácia dos codificadores, o Gráfico 3 mostra o resultado de outra simulação também com o SOVA, e desta vez utilizando um codificador FIR. Para a simulação foi escolhido o codificador da Figura 11, que é um código convolucional não recursivo Cconv (2,1,3).

Gráfico 3 – Análise de desempenho de um Cconv (2,1,3).



Fonte: Elaborada pelo autor (2020).

Analisando o Gráfico 3, é possível perceber este codificador se comportou melhor que o codificador do Gráfico 2, de tal modo que escolhendo o mesmo ponto (SNR = 8dB) que foi escolhido na análise do Gráfico 2, o valor correspondente para este SNR é BER=$0{,}8 \times 10^{-5}$ utilizando este outro codificador. Logo, pode-se dizer que este codificador mostrou um resultado com melhor eficácia na correção de erros neste mesmo sistema de transmissão, reduzindo cerca de 10 vezes a taxa de erros em relação a um cenário sem codificação.

Porém, este teste deve ser realizado para cada tipo de situação, pois o comportamento do codificador varia conforme o sistema de transmissão no qual ele é aplicado, podendo ser melhor em alguns casos, e pior em outros.

Contudo, é possível perceber, após essas duas análises, que a utilização de codificadores em um sistema de transmissão reduz a BER de forma a melhorar a confiabilidade da mensagem transmitida, comprovando os estudos abordados no decorrer deste trabalho.

7. Conclusão e Trabalhos Futuros

O uso de codificadores em um sistema de comunicação é imprescindível para que se tenha confiabilidade na mensagem recebida. É necessário conhecer as características deste sistema para que o codificador seja dimensionado, sempre buscando melhor eficiência e qualidade no sinal recebido.

Este trabalho teve como principal objetivo utilizar um dos diversos tipos existentes de codificadores, que são os codificadores convolucionais e, através de um estudo do algoritmo de Viterbi e do SOVA, mostrou uma diferença significativa de desempenho entre um sistema de comunicação com e sem codificação. As análises gráficas ajudam a evidenciar esta diferença.

Com base nesses resultados, estudos sobre Códigos Turbo, Códigos de Bloco, Códigos Cíclicos, Códigos Reed-Muller e Códigos Reed-Solomon permitirão comparações úteis na eficiência da detecção e correção de erros.

Referências Bibliográficas

HAYKIN, S. **Sistemas de Comunicação: Analógicos e Digitais,** 4 ed. São Paulo: Bookman, 2004.

MOON, T. K. **Error Correction Coding: Mathematical Methods and Algorithms.** New Jersey, USA: WILEY, 2005.

MORELOS-ZARAGOZA, R. H. **The Art of Error Correcting Coding. Chichester**, England: John Wiley & Sons, 2002.

PROAKIS, J. G.; SALEHI, M.; BAUCH, G. **Contemporary Communication Systems using Matlab and Simulink**. 2ª ed. New Jersey, USA: Bookware Company, 2006.

SKLAR, B. **Digital Communications, Fundamentals and Applications.** New Jersey,USA: Prentice Hall, 1993.

MOREIRA, J. C.; FARREL, P. G. **Essentials of Error-Control Coding.** West Sussex, England: WILEY, 2006.

TANNER, L. M. **A recursive approach to low complexity codes.** *IEEE Trans. Inf. Theory*, vol. 27, no. 5, pp. 533–547, 1981.

TANG, H.; XU, J.; KOU, Y.; LIN, S. and Abdel-Ghaffar, K.; **On algebraic construction of Gallager and circulant low-density parity-check codes**. *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1269–1279, June 2004.

KOU, Y.; LIN, S.; FOSSORIER, M. **Low-density parity-check codes based on finite geometries: A rediscovery and new results.** *IEEE Trans. Inf. Theory*, vol. 47, pp. 2711–2736, November 2001.

AMMAR, B.; HONARY, B.; KOU, Y.; XU, J.; LIN, S. **Construction of low-density parity check codes based on balanced incomplete block designs.** *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1257–1269, June 2004.

ARNONE, L.; GAYOSO, C.; GONZALEZ, C.; CASTIÑEIRA, J. **A LDPC logarithmic decoder implementation.** *Proc. VIII International Symposium on Communications*

Theory and Applications, St. Martin's College, Ambleside, United Kingdom, pp. 356–361, July 2005.

NAYAGAM, A. **LDPC toolkit for Matlab**, disponível em http://arun-10.tripod.com/ldpc/generate.html, Maio 2002.

BRINK, S. T., KRAMER, G.; ASHIKHMIN, A. **Design of low-density parity-check codes for modulation and detection.** *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 670–678, April 2004.

HAGENHAUER, J.; OFFER, E.; PAPKE, L. **Iterative decoding of binary block and convolutional codes.** *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, March 1996.

LIN, S.; COSTELLO, D. **Error Control Coding: Fundamentals and Applications.** New Jersey, USA: Prentice-Hall, 1983.

ELIAS, P.; **Error-Free Coding.** IRE Trans., vol. PGIT-4, pp. 29-37, 1954.

WOZENCRAFT, J. M.; JACOBS, I. M. **Principles of Communication Engineering.** New York: John Wiley & Sons, 1965.

MASSEY, J. L. **Threshold Decoding.** Cambridge, MA: MIT Press, 1963

FANO, R. M. **A Heuristic Discussion of Probabilistic Decoding.** *IEEE Transactions on Information Theory*, IT-9, pp. 64-74, April 1963.

JELINEK, F. **A Fast Sequential Decoding Algorithm Using a Stack.** *IBM Journal of Research and Development*, Vol. 13, pp. 675-685, November 1969.

OKUMURA, J. K. **On the Viterbi Decoding Algorithm.** *IEEE Transaction on Information Theory*, Vol. IT-15, pp. 177-179, January, 1979.

BERROU, C.; GLAVIEUX, A; THITIMAJSHIMA, P. **Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes.** *Proc. I993 IEEE Int. Con$ Coinm. (ICC'93)*, pp. 1064-1070, Geneve, Switzerland, May 1993.

WICKER, S. B.; **Error Control Systems for Digital Communication and Storage.** Prentice Hall, 1995.

VITERBI, A. J.; OMURA, J. K. **Principles of Digital Communication and Coding.** New York, USA: McGraw-Hill, 1979.

VITERBI, A. J. **Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm.** *IEEE Trans. Information Theory*, vol. 13, pp. 260-269, Apr. 1967.

VUCETIC, B. **Iterative Decoding Algorithms.** *Invited paper, The International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'97*, Sept. 1-4, Helsink, Finland, 1997.

HAGENAUER, J.; HOEBER, P. **A Viterbi algorithm with soft-decision outputs and its applications.** *Proc. Globecom'89*, pp.1680-1686, Nov. 1989.

LI, Y.; VUCETIC, B. **A Generalized MLSE Algorithm.** *INNSP'95*, China, December 10-13, 1995.

LI, Y.; VUCETIC, B.; SATO, Y. **Optimun soft-output detection for channels with inter-symbol interference.** *IEEE Trans. Inform. Theory*, Vol-41, No. 3, May 1995, pp. 704-713.

VUCETIC, B.; YUAN, J. **Turbo Codes: Principles and Applications.** Massachussetts, USA: Kluwer Academic Publishers, 2000.