UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FREDERICK ALVARES IMGÄRTCHEN

AVALIAÇÃO DE SISTEMA DE COMUNICAÇÃO DIGITAL SEM FIO UTILIZANDO CODIFICAÇÃO CONVOLUCIONAL ADAPTATIVA

PORTO ALEGRE
2020

FREDERICK ALVARES IMGÄRTCHEN

AVALIAÇÃO DE SISTEMA DE COMUNICAÇÃO DIGITAL SEM FIO UTILIZANDO CODIFICAÇÃO CONVOLUCIONAL ADAPTATIVA

Projeto de diplomação apresentado ao programa de graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como requisito parcial à obtenção do grau de bacharel.

Orientador: Prof. Dr. IVAN MÜLLER

PORTO ALEGRE

2020

FREDERICK ALVARES IMGÄRTCHEN

AVALIAÇÃO DE SISTEMA DE COMUNICAÇÃO DIGITAL SEM FIO UTILIZANDO CODIFICAÇÃO CONVOLUCIONAL ADAPTATIVA

Projeto de diplomação apresentado ao programa de graduação em Engenharia Elétrica da Universidade Federal do Rio Grande do Sul, como requisito parcial à obtenção do grau de bacharel.

Aprovado em: Porto Alegre, 01 de Dezembro de 2020.

_____

Prof. Dr Ivan Müller – UFRGS
Orientador

_____

Prof. Me. Max Feldman – UFRGS
Examinador

_____

Prof. Dr. Giovani Bulla – UFRGS

Examinador

Dedico o presente trabalho aos meus pais, Misia e Harry, pelo apoio constante e esforço em garantir minha educação.

AGRADECIMENTOS

O aprendizado é um bem perene, um patrimônio que não se perde, uma vez adquirido, fica guardado conosco. Tive o privilégio de ter várias pessoas que me ensinaram grandes lições, as quais foram fundamentais para a realização deste trabalho. Portanto, vou utilizar este espaço para agradecer essas pessoas. Primeiramente agradeço aos meu pais, Misia e Harry, que sempre estiveram ao meu lado, me motivaram e me apoiaram para que eu pudesse focar nos meus estudos e na conclusão deste trabalho. Agradeço à minha irmã, Caroline, pelas opiniões compartilhadas sobre o texto. Agradeço aos meus avós e tia-avó pelos ensinamentos e apoio constante.

Agradeço ao meu orientador, Prof. Ivan Müller, por aceitar conduzir meu trabalho de pesquisa, pelo tempo dedicado e a atenção disponibilizada. Sem a sua orientação, o caminho até a conclusão seria mais longo e árduo.

Agradeço ao Prof. Marcelo Lubaszewski pela excelente condução da cadeira de Projeto de Diplomação.

Agradeço aos professores do curso de Engenharia Elétrica da Universidade Federal do Rio Grande do Sul pelos conhecimentos compartilhados.

Agradeço aos profissionais da empresa Android por me proporcionarem o estágio obrigatório e complementarem minhas habilidades técnicas.

Por fim, agradeço aos meus amigos, colegas e a todas as pessoas que acompanharam e participaram da minha trajetória acadêmica.

"*O que sabemos é uma gota; o que ignoramos é um oceano. Mas o que seria o oceano se não infinitas gotas?*"

Isaac Newton

RESUMO

A codificação convolucional é uma técnica para propiciar correção de erro, e o seu uso em sistemas de comunicação digital permite reduzir a taxa de erro de bit (BER). Neste trabalho, essa redução é demonstrada através de simulações no ambiente MATLAB, que comprovam a eficácia do sistema. A codificação convolucional consiste em inserir bits de paridade no sinal da informação e esses bits são utilizados pelo decodificador para corrigir possíveis erros advindos das imperfeições do canal de comunicação. Como consequência negativa, a taxa de dados efetivos diminui, ocasionando em uma menor eficiência espectral. Por outro lado, a utilização de correção de erro em avanço aumenta a robustez do enlace, proporcionando maior qualidade de serviço. Foi proposto no presente trabalho o uso da técnica de puncionamento para reduzir a quantidade de bits de paridade conforme a relação sinal/ruído (SNR) do canal. O objetivo é demonstrar que se pode variar a taxa do código conforme o SNR do canal. Desta forma é apresentado o *trade off* entre taxa de bits errados e uso eficiente da banda do canal. Em suma, canais com alta relação sinal ruído podem operar com altas taxas de código, enquanto canais com baixos valores de SNR necessitam de baixas taxas de código, e, nesse sentido, a adaptação dos códigos convolucionais empregados é eficiente. Os resultados do trabalho permitem concluir tal proposição através das simulações, onde é demonstrado que o uso de um codificador convolucional adaptativo, com variação automática da taxa do código mantém baixas taxas de bits errados enquanto reduz o uso da banda do canal. É possível constatar também, que codificadores com padrões de puncionamento diferentes porém com mesma taxa de código, possuem desempenhos diferentes.

Palavras-chave: Códigos convolucionais; Puncionamento; BER; SNR; MATLAB.

ABSTRACT

The Convolutional codification is a technique to provide Error-correcting, and your usage in Digital Communication Systems allows to reduce the Bit Error Rate (BER). In this Paper this reduction is demonstrated with simulations in MATLAB Software, that proves the system efficiency. The Convolutional codification consists in inserting parity bits in the Information signal, those bits are used by the Decoder to correct possible errors that are originated from Communication Channel imperfections. As a downside consequence, the Data Rate of the signal lowers causing lower spectral efficiency. On the other hand, the use of Error-correcting in advance increases the link robustness, providing better quality service. The usage of punctioning techniques is proposed in the present Paper to reduce the quantity of parity bits according to the Channel Signal to Noise Ratio (SNR). The goal is to demonstrate that it is possible to change the Code Rate according to the Channel SNR. Thus, it is presented the trade off between Bit Error Rate and efficient Channel band Usage. Channels with high Signal to Noise Ratio can operate with high Code Rates, as Channels with low values of SNR needs low Code Rates and, in that sense, the adaptation of the Convolutional Codes employed is efficient. The work results allow to conclude such proposition through the simulations, where it is demonstrated that the usage of an adaptive Convolutional Encoder with automatic variation of Code Rate; keeps low bit error rates as reduces the usage of Channel band. It is also possible to conclude that Encoders with different Puncturing patterns but identical Code Rates has different performances.

Keywords: Convolutional Codes; Puncturing; BER; SNR; MATLAB.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

O uso de comunicação digital sem fio é nova realidade. Diariamente, as pessoas utilizam seus computadores e celulares para acessar dados e se comunicar, portanto, sistemas rápidos e robustos para transmissão e recepção de dados são necessários. Ruídos aleatórios, interferências e coexistência são algumas das diferentes fontes de erro na transmissão e recepção de dados. Para o usuário é imperceptível que eventualmente estes erros ocorram, porém, sem o uso de técnicas de detecção e correção de erros, a transmissão robusta de dados seria impossível (JIANG, YUAN; 2010).

A codificação de canal é uma técnica amplamente utilizada para melhorar a taxa de erro de canais de comunicação digitais, e o início desse método deu-se em meados do século 20 com o artigo *A Mathematical Theory of Communication* publicado por Shannon (SÍLVIO A. ABRANTES; 2003), a partir desse trabalho diversas variações surgiram como a codificação de blocos, convolucional, Reed- Solomon e códigos Turbo.

O principio básico da codificação de canal é incluir redundâncias (bits de paridade) na mensagem original de forma a reduzir a probabilidade de erro no receptor do canal, o fato de adicionar bits de paridade reduz a taxa de dados do código e pode acarretar em um maior consumo da banda do canal, portanto, o emprego das técnicas de codificação de canal devem ser analisadas utilizando o conceito de *trade off* entre redução de erro e consumo de banda.

A codificação convolucional, técnica a qual será dada ênfase nesse trabalho, foi fundamentada em 1955 por Peter Elias (SÍLVIO A. ABRANTES; 2010). Essa técnica de correção de erros consiste em multiplicar a mensagem que deve ser enviada pela resposta ao impulso do codificador. Os componentes básicos presentes em um codificador convolucional são registradores e portas XOR. Esses registradores são memórias, eles guardam valores passados dos bits de informação da entrada do codificador. Desta forma, diferentemente da codificação por blocos, a codificação convolucional possui memória.

A técnica de puncionamento permite uma abordagem mais prática na utilização da codificação convolucional, pois permite alterar a taxa de dados do código através da supressão de bits de paridade. A utilização deste método em certas condições de ruído do canal pode ser vantajosa. Será feito o uso desta técnica no presente trabalho juntamente com a codificação convolucional.

O presente trabalho tem por objetivo demonstrar que o uso de codificação convolucional reduz o erro na transmissão da informação e apresenta um papel fundamental em canais com alto índice de ruído. Em um segundo momento é proposto o uso da técnica de puncionamento

juntamente com a codificação convolucional. O intuito dessa segunda proposta é demonstrar que para determinados valores de ruído no canal é possível aumentar a taxa do código e manter uma baixa incidência de erro de bits, e esse aumento da taxa de código é possível devido ao uso da técnica de puncionamento. O trabalho foi feito através de simulações no Simulink, ambiente de simulação por diagrama de blocos do *software* Matlab da empresa *MathWorks Inc.* A simulação consiste basicamente na transmissão de um sinal em um canal de comunicação digital AWGN (*additive white Gaussian noise*) e na comparação da taxa de erro de bits (*Bit Error Rate*) para as diferentes situações, sem codificação, com codificação e com codificação e puncionamento. Notar que o termo "sem codificação" aqui utilizado refere-se a não utilização de codificação redundante para mitigar os efeitos do canal e, por consequência, aumentar a robustez do enlace.

O trabalho está estruturado em seis capítulos, os quais contemplam uma revisão literária, que consiste na teoria necessária para a compreensão do texto, presente no capítulo 2, a explanação dos métodos e materiais utilizados para a simulação no capítulo 3, o desenvolvimento da proposta com todos os detalhes de simulação no capítulo 4, a exposição dos resultados obtidos e a interpretação dos mesmos no capítulo 5 e, por fim, a conclusão do trabalho e considerações para pesquisas futuras sobre o tópico no capítulo 6.

2 REVISÃO DA LITERATURA

Na comunicação digital de sinais existem alguns elementos básicos, entre eles, a fonte (local onde é originada a informação), o canal de comunicação (meio onde está sendo transmitido o sinal), e o receptor (local onde se destina a informação). A fonte é comumente modelada como um bloco de geração de informação pseudoaleatória, o canal de comunicação é comumente categorizado nos tipos DMC (*Discrete Memoryless Channel*) e AWGN. Essas categorizações são utilizadas para modelar o ruído presente no canal, e esses ruídos podem ser de natureza determinística e/ou probabilística e adicionam erro ao sinal recebido pelo receptor. Com o advento dos sistemas de comunicação digital, tornou-se necessário utilizar técnicas de codificação de canal. Esse tipo de codificação é feito antes da transmissão do sinal e é utilizado para diminuir o erro entre a mensagem recebida no receptor e a mensagem enviada pela fonte. Para minimizar o erro entre a informação da fonte e do receptor, são utilizadas as codificações de canal e essas codificações adicionam sequências determinadas para cada informação a ser enviada pelo canal. Os dois tipos mais utilizados são a codificação de bloco e a codificação convolucional. A codificação é, em suma, uma redundância incluída na mensagem para amenizar os efeitos de ruídos e interferências no canal utilizado. De forma geral, a codificação de canal pode ser dividida em três tipos: de bloco, convolucional e turbo. No presente trabalho, será dada ênfase no segundo tipo uma vez que a codificação convolucional é amplamente utilizada em sistemas de telecomunicação.

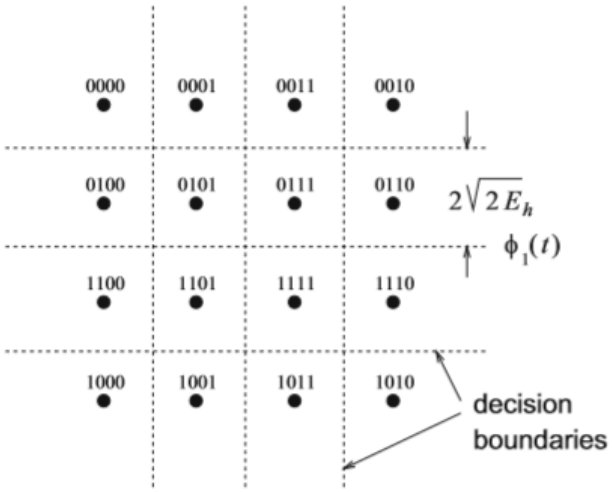
2.1 CANAL AWGN

O meio de transmissão do sinal deve ser considerado ao modelar um sistema de comunicação, e isso é necessário visto que o meio não é ideal e, portanto, interfere no sinal enviado pela fonte podendo ocasionar em erros na recepção. Para prever esses efeitos foram definidos modelos de canais, um desses modelos é o AWGN. Este tipo de canal considera a adição de ruído branco gaussiano no sinal de entrada, portanto, a saída é a mensagem da fonte somada ao ruído, o que gera erros de comunicação semelhantes ao que acontece na prática. Para este modelo, o ruído é branco e, portanto, independente no tempo e a sua densidade de probabilidade tem uma distribuição gaussiana, possuindo média zero.

2.2 MODULAÇÃO QAM

Em um sistema de comunicação digital o processo de modulação é a conversão de bits em sinais elétricos de banda passante, ou seja, o processo de modulação é a conversão da informação digital para a analógica utilizando uma portadora. A modulação QAM (*Quadrature Amplitude Modulation*) consiste numa combinação entre a modulação ASK (*Amplitude shift Keying*) e PSK (*Phase Shift Keying*). Uma vantagem da modulação QAM em comparação com as demais é que ela permite a transmissão de um número maior de bits para a mesma portadora. Na prática isso melhora a taxa de dados no canal, porém pode aumentar o erro em caso de sinais ruidosos. Na Figura 1 é possível observar a constelação para uma modulação 16-QAM, onde as combinações possíveis de fase e amplitude da portadora permitem a representação de 4 bits simultaneamente. Desta forma existem 16 símbolos diferentes que podem ser transmitidos a cada período.

Figura 1 – Constelação 16QAM



Fonte: Principles of Mobile Communication, 4th ed. – Gordon L. Stüber

2.3 CÓDIGOS CONVOLUCIONAIS

Os códigos convolucionais foram fundamentados em 1955 por Peter Elias como uma alternativa à codificação por blocos (SÍLVIO A. ABRANTES; 2010). A codificação convolucional baseia-se na multiplicação dos *k* bits do sinal de entrada m por K-1 valores dentro de memórias e essas memórias são registradores, normalmente flip-flops que fazem parte do codificador convolucional implementado em hardware. A quantidade de registradores define

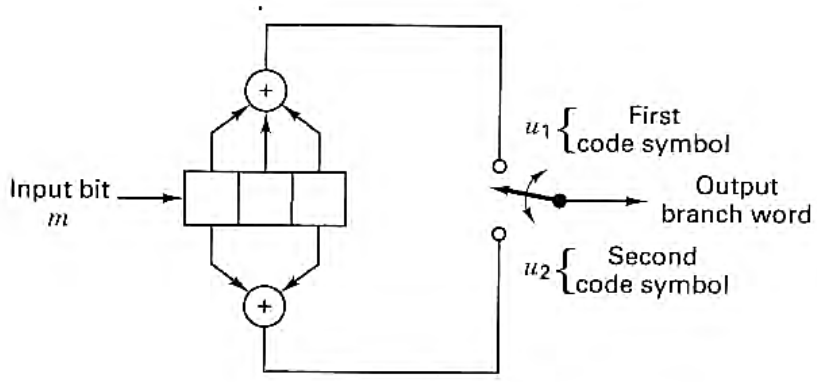
um parâmetro do encoder chamado comprimento de restrição (K), e dentro destes registradores ficam os últimos K valores do sinal de entrada. O nome vem da operação de convolução, que é a multiplicação de um sinal de entrada por uma resposta ao impulso, portanto, a codificação convolucional é a multiplicação do sinal de entrada pela resposta ao impulso do codificador, semelhante a um filtro de resposta finita ao impulso (FIR), onde o sinal codificado não depende apenas do sinal de entrada, mas também dos valores nos registradores.

Um codificador convolucional pode ser descrito por três números inteiros (*k*, *n*, *K*), onde *k* é o número de bits na entrada do codificador, *n* é o número de bits na saída e *K* é a quantidade de registradores. Será utilizado como base para a elaboração do assunto, o codificador convolucional da Figura 2, chamado de codificador de taxa ½. A taxa em um codificador convolucional é definida pela Equação 1, onde é possível observar que este codificador possui *k* igual a 1, *n* igual a 2 e *K* igual a 3, ou seja, a entrada é constituída de 1 bit, a saída é constituída de 2 bits e o codificador possui 3 registradores.

$$\boldsymbol{DR = k/n} \tag{1}$$

Na Equação 1, DR denota *Data Rate*, que é a taxa do codificador, sendo assim é um valor que indica a quantidade de redundância inserida na mensagem. Quanto menor a taxa de informação (*Data Rate*) maior a quantidade de bits de paridade na mensagem.

Figura 2 – Codificador convolucional de taxa 1/2



Fonte: Digital Communications. Fundamentals and Applications – Bernard Sklar

Pode-se observar na Figura 2 dois módulos de adição (portas XOR) e setas saindo dos registradores e entrando nas portas XOR, sendo essa, a representação das conexões do codificador e definem a resposta ao impulso do mesmo. Essas conexões podem ser expressas

através de n vetores, um para cada porta XOR, e a dimensão desses vetores é igual a *K* (número de registradores). As conexões do codificador da Figura 2 podem ser representadas, portanto por dois vetores de conexão $\mathbf{v_1}$ (representado as conexões da porta XOR superior) e $\boldsymbol{v_2}$ (representado as conexões da porta XOR inferior) cada um com dimensão *K*=3, ambos expressos nas Equações 2 e 3 respectivamente. Quando houver uma seta ligando o registrador à porta XOR o valor da posição do vetor será "1", quando não houver ligação o valor será "0".

$$\boldsymbol{v_1} = \mathbf{1\ 1\ 1} \tag{2}$$

$$\boldsymbol{v_2} = \mathbf{1\ 0\ 1} \tag{3}$$

Se uma mensagem $\boldsymbol{m}$ = 1 0 1 for inserida no codificador da Figura 2 a saída esperada será a mensagem codificada $\boldsymbol{g}$ = 11 10 00 10 11 esse resultado é decorrente do processo de convolução do sinal de entrada (a mensagem) com a função de transferência do codificador.

2.3.1 **Resposta ao impulso**

Ao colocar um impulso na entrada de um decodificador, pode-se encontrar a função de transferência do mesmo. Um impulso nada mais é do que um sinal com apenas um bit "1" que se movimenta através dos registradores do codificador. Considerando, portanto, que um impulso seja colocado na entrada do codificador da Figura 2, o resultado encontrado será o apresentado no Quadro 1.

Quadro 1 – Cálculo da resposta ao impulso

		Saída do codificador	
Instante de tempo	**Conteúdo dos registradores**	**g1**	**g2**
t1	1 0 0	1	1
t2	0 1 0	1	0
t3	0 0 1	1	1

Fonte: Próprio Autor

Pode-se observar na Equação 4 o resultado da resposta ao impulso do codificador, também denominada função de transferência.

$$\boldsymbol{g = 11\ 10\ 11} \tag{4}$$

Agora, para uma entrada qualquer **m**, pode-se encontrar a saída através da superposição dos impulsos do sinal de entrada deslocados, conforme seu instante de tempo *t*. Este processo pode ser visualizado no Quadro 2, considerando o sinal de entrada ***m*** = 101.

Quadro 2 – Cálculo da saída do codificador

Entrada	Saída				
1	1 1	1 0	1 1		
0		0 0	0 0	0 0	
1			1 1	1 0	1 1
Adição XOR:	1 1	1 0	0 0	1 0	1 1

Fonte: Próprio Autor

O resultado, portanto, é consistente com o encontrado anteriormente e demonstra que a codificação convolucional é uma operação linear.

2.3.2 **Representação polinomial**

Os códigos convolucionais podem ser representados por polinômios, essa representação é semelhante aos vetores de conexão, explicados anteriormente. Cada somador XOR será representado por um polinômio de ordem *K*-1 ou menos (onde *K* é o comprimento de restrição do codificador), o coeficiente de cada termo será um ou zero, dependendo se há conexão entre o registrador e o somador XOR. Quando houver conexão, o coeficiente terá valor igual a 1, quando não houver, igual a 0. Para o codificador da Figura 2 a representação polinomial para os somadores XOR superior e inferior pode ser encontrada nas Equações 5 e 6 respectivamente.

$$\boldsymbol{v_1(X) = 1 + X + X^2} \tag{5}$$

$$\boldsymbol{v_2(X) = 1 + 0X + X^2} \tag{6}$$

A mensagem codificada pode ser encontrada através do cálculo representado na Equação 7.

$$\boldsymbol{U(X) = m(X)v_1(X)\ entrelaçada\ com\ m(X)v_2(X)} \tag{7}$$

Para um sinal de entrada **m** = 101 a representação polinomial do sinal de entrada e saída está explicitada nas Equações 8 e 9 respectivamente.

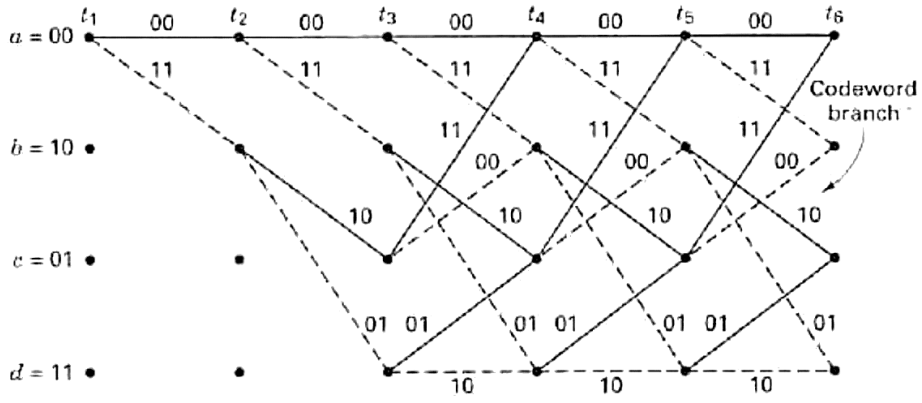
$$\boldsymbol{m}(\mathbf{X}) = \mathbf{1} + \mathbf{0X} + \mathbf{X^2} \tag{8}$$

$$\boldsymbol{U}(\mathbf{X}) = (\mathbf{1},\mathbf{1}) + (\mathbf{1},\mathbf{0})\mathbf{X} + (\mathbf{0},\mathbf{0})\mathbf{X^2} + (\mathbf{1},\mathbf{0})\boldsymbol{X^3} + (\mathbf{1},\mathbf{1})\boldsymbol{X^4} \tag{9}$$

2.3.3 **Diagrama de treliças**

Codificadores convolucionais são máquinas de estado finitas e, portanto, podem ser representadas por um diagrama de transição de estados. O método mais utilizado para representação de códigos convolucionais é em termos do diagrama de treliças (PROAKIS, JOHN G., 1996). Essa representação consiste em plotar os 2^{K-1} estados (representados por pontos) para cada instante de tempo *t* com ramos interligando os pontos que correspondem às transições entre os estados. Ramos representados por linhas sólidas indicam saída geradas por bits de entrada iguais a 0 e linhas pontilhadas indicam saídas geradas por bits de entrada iguais a 1. A Figura 3 ilustra o diagrama de treliça do codificador convolucional da Figura 2.

Figura 3 – Diagrama de treliça do codificador de taxa 1/2



Fonte: Digital Communications. Fundamentals and Applications – Bernard Sklar

Note que o diagrama de treliças sempre inicia do estado 00 e percorre a treliça até voltar para o estado 00 e que quanto maior for o comprimento de restrição do codificador, maior será o número de estados, essas duas variáveis possuem uma relação exponencial e acarretam numa maior complexidade do diagrama.

2.3.4 **Decodificação de códigos convolucionais e o algoritmo de Viterbi**

A decodificação de um código convolucional consiste em obter o sinal de entrada do codificador a partir da saída do mesmo, ou seja, realizar o processo inverso. Existem vários métodos para realizar tal decodificação, porém o Algoritmo de Viterbi é o método mais utilizado. Este algoritmo percorre o diagrama de treliças de forma a achar o caminho mais provável de gerar a sequência recebida, e esta forma de decodificação utiliza o conceito de probabilidade máxima. Neste trabalho serão vistas as decodificações *Hard-decision* em canais Gaussianos, onde o caminho mais provável é aquele no qual a sequência recebida possui a menor distância Euclidiana do código $U(X)$. A grande vantagem do algoritmo de Viterbi é que, conforme ocorrem as iterações, ele ignora os caminhos que não são mais candidatos a menor distância métrica, e esse comportamento reduz a complexidade da decodificação.

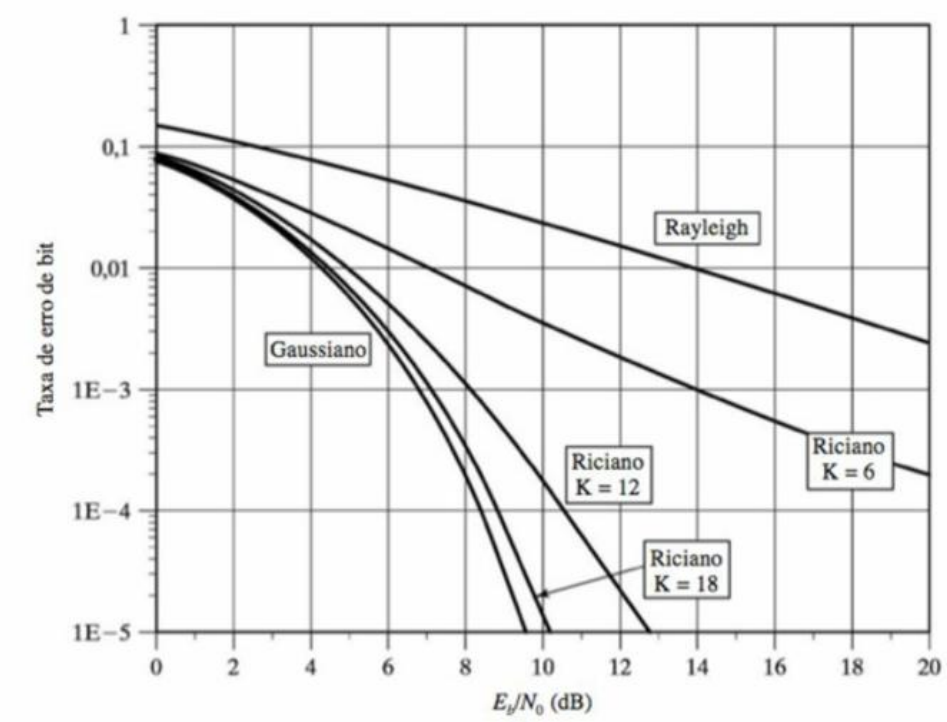
2.4 BER – *BIT ERROR RATE*

Existem diversas formas de se quantificar a qualidade de um sistema de comunicação digital. Uma das formas mais utilizadas é o BER (*Bit Error Rate*). Este parâmetro é definido como a razão entre a quantidade de bits com erro e o número de bits enviados. A expressão matemática pode ser encontrada na Equação 10.

$$\boldsymbol{BER = \frac{Nº\ de\ bits\ com\ erro}{Nº\ total\ de\ bits}} \tag{10}$$

Quanto menor for o BER, melhor é a qualidade do sistema de comunicação, e este parâmetro é comumente analisado juntamente com a variação do SNR (*Signal to Noise Ratio*) e utilizado para medir a efetividade de modulações e codificações de canais. Na Figura 4 pode-se observar gráfico gerado para uma aplicação usual deste parâmetro.

Figura 4 – Gráfico de relação BER vs SNR para diferentes tipos de canal.



Fonte: Michael, H.S.; Sistemas Modernos de Comunicações Wireless (2008, p.180)

Neste caso o BER está sendo utilizado para analisar o desempenho de diferentes tipos de canais conforme varia o SNR. Três tipos de canais foram analisados, o AWGN, Rician (com diferentes valores de *K*) e Rayleigh. Os últimos dois canais são modelos de multi-caminhos e, portanto, um erro maior é esperado e, de fato, isto ocorre como observado no gráfico da Figura 4.

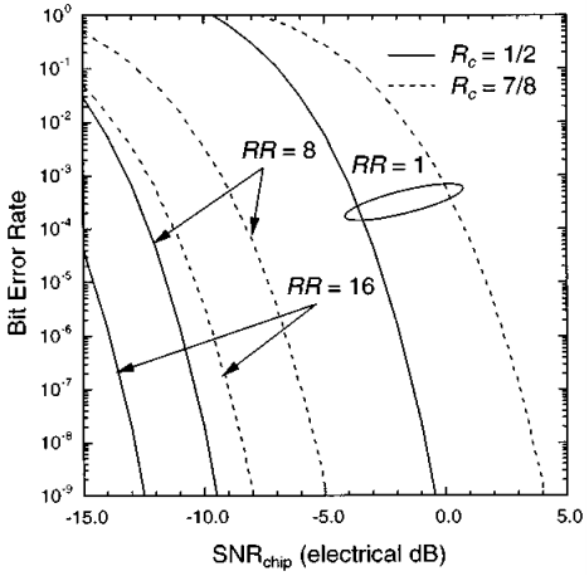
2.5 TRABALHOS RELACIONADOS

Esta seção apresenta trabalhos relacionados ao uso da codificação convolucional utilizada para diminuir o BER. Desta forma, pelo melhor esforço de pesquisa, estão aqui destacados trabalhos importantes sobre o tema e os resultados obtidos para o problema em questão.

No trabalho de Michihito Matsuo. *et al*. (2000) é apresentado o uso de codificação convolucional com puncionamento em um sistema de comunicação infravermelho, por tanto, sem fio. O autor propõe variar a taxa de bits enviados dependendo da condição do canal com o intuito de diminuir o BER. Essa variação foi feita para duas taxas de código (1/2 e 7/8), o resultado pode ser observado no gráfico da Figura 5. A partir do estudo pôde-se concluir que a variação na taxa de bits enviados viabiliza o funcionamento do sistema mesmo para valores de

SNR baixos tais como -7,2 dB. Outra abordagem para atingir viabilidade na comunicação trata da variação da taxa de puncionamento (conforme o presente trabalho) visto que a variação da taxa de bits enviados afeta a taxa de código. O mesmo efeito poderia ser alcançado com a variação do número de bits de paridade.

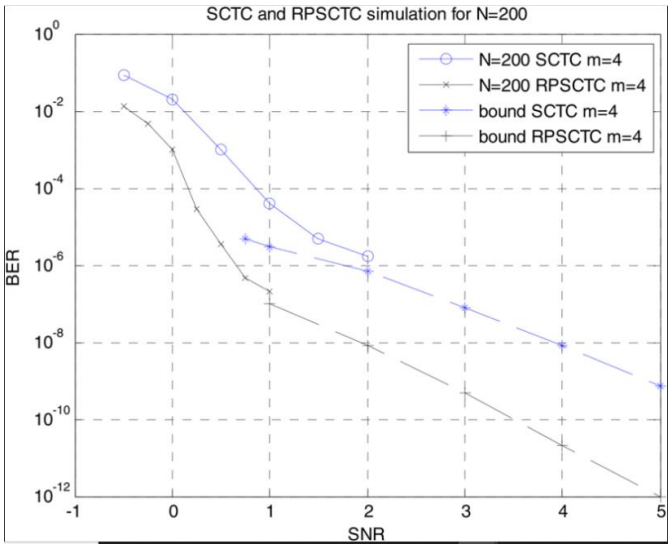
Figura 5 – BER vs SNR para diferentes taxa de bits enviados e diferentes taxas de código



Fonte: Matsuo, Michihito. *et al* (2000, p.57)

No trabalho de Pillay Narushan *et al*. (2007) é apresentado um estudo que simula código turbo em um canal AWGN, onde, para aprimorar o sistema, foi proposto o uso de puncionamento. Foi utilizado o parâmetro BER para avaliar o desempenho do sistema. Na Figura 6 pode-se observar o gráfico de BER versus SNR, e o resultado corrobora com a hipótese levantada no trabalho. A variação na taxa do código através do puncionamento não foi explorada no trabalho, ponto que poderia ser abordado como uma continuação a ser proposta.

Figura 6 – Relação BER vs SNR para um sistema com codificação turbo e puncionamento



Fonte: Pillay Narushan *et al*., (2007, p.4)

No trabalho de Sari Lydia (2014) é apresentado um estudo sobre o desempenho de códigos convolucionais puncionados em canais AWGN e Rayleigh. O artigo demonstra que codificadores com a mesma taxa de código, porém diferentes padrões de puncionamento, apresentam desempenhos diferentes. Sob essa perspectiva define-se que para obter codificadores com boa relação BER versus SNR devem-se utilizar padrões de puncionamento já conhecidos e amplamente validados. Também se conclui que devem ser feitas mais pesquisas e definidas bases de dados maiores para encontrar a correlação entre bom desempenho e padrão de puncionamento. Este estudo complementa o presente trabalho, visto que apresenta uma abordagem para definição de padrões de puncionamento.

3 MÉTODOS E MATERIAIS

Este capítulo tem como objetivo explanar e descrever a metodologia seguida. As ferramentas utilizadas para a execução do trabalho estão detalhadas nesta seção, onde serão identificados os softwares utilizados e detalhados os blocos usados na simulação. Os parâmetros envolvidos também são detalhados.

3.1 SOFTWARE DE SIMULAÇÃO

A simulação da presente proposta foi feita utilizando o software Matlab R2020a da empresa *MathWorks Inc.* com a licença de estudante. O Matlab é um software de simulação computacional projetado para análises iterativas e desenhar processos com uma linguagem de programação que expressa matrizes e vetores de forma direta (MathWorks.com, 2020). O programa possui diversas extensões, denominadas *Toolboxes*, que permitem ao usuário ter acesso a bibliotecas com funções pré-definidas. Para a realização do trabalho foi utilizado a *Toolbox* do *Simulink* e *Communications System*. O *Simulink* é um ambiente de simulação por diagrama de blocos que permite simular canais de comunicação de forma intuitiva sem a utilização de códigos. O *Communication System* é um *Toolbox* específico para simulação de sistemas de comunicação, ele contém funções para simular modulação de sinais, modelos de canais de comunicação, codificação de canal, entre outros. A especificação do software e hardware utilizado para simular o sistema é apresentada no Quadro 3.

Quadro 3 – Especificação do Software e Hardware utilizado na simulação

Software	
Versão	MatLab R2020a
Toolboxes utilizadas	- Simulink - Communication System
Tipo licença	Estudante
Sistema Operacional	Windows 10
Hardware	
Processador	Intel Core i5
Memória RAM	8GB DDR4

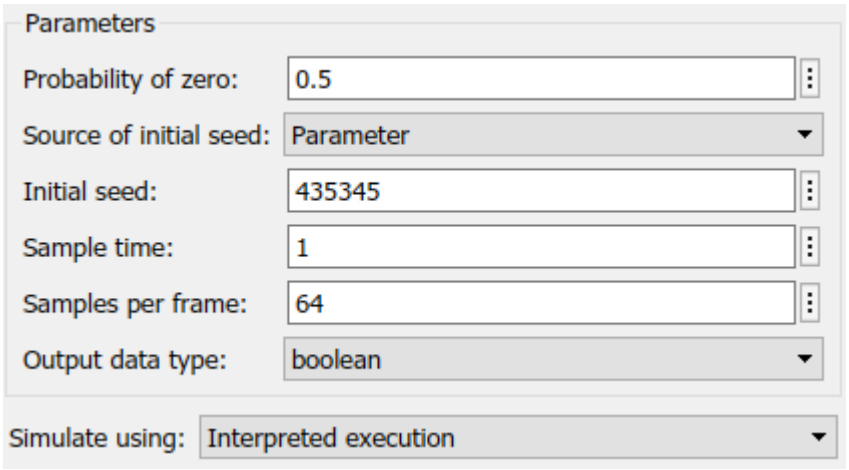
Fonte: o Autor

3.2 GERADOR DE BERNOULLI

O primeiro bloco utilizado é o gerador de Bernoulli, bloco do Simulink que permite gerar código binário pseudoaleatório utilizando uma distribuição de Bernoulli. A finalidade deste bloco para o sistema é simular uma fonte de informação em um canal digital.

Os parâmetros de configuração deste bloco podem ser encontrados na Figura 7. A configuração utilizada define a probabilidade do número gerado ser zero, o tipo de dado gerado pelo bloco é booleano e a taxa de amostras por quadro de dados.

Figura 7 – Parâmetros do bloco gerador de Bernoulli



Fonte: Simulink

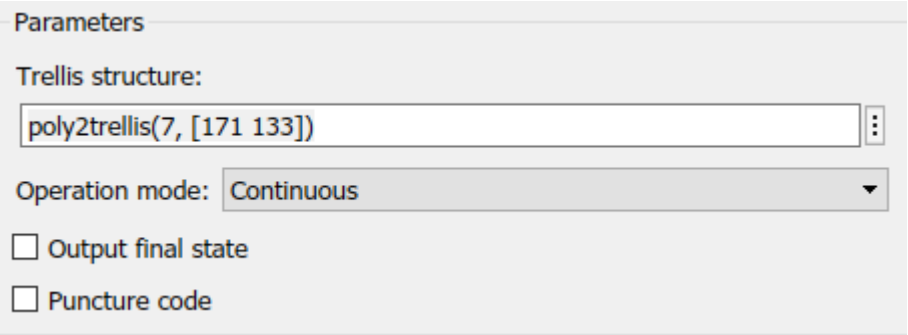
Este bloco foi escolhido para a simulação por gerar um sinal aleatório. Outro bloco, o "*Random integer generator*" também poderia ser utilizado sem grandes alterações no resto do sistema.

3.3 CODIFICADOR CONVOLUCIONAL

O codificador convolucional é um bloco que codifica a informação convolucionalmente. Ele recebe uma sequência de bits na sua entrada na forma de um vetor e na sua saída fornece um vetor com a sequência recebida codificada.

A parametrização utilizada neste bloco pode ser encontrada na Figura 8, onde o funcionamento do codificador é definido por uma estrutura de treliça e isso é feito através da função *polytrellis* do MatLab.

Figura 8 – Parâmetros do bloco de codificação convolucional



Fonte: Simulink

O primeiro parâmetro dessa função é o comprimento de restrição (*K*) do codificador e o segundo é o polinômio gerador. Este polinômio (conforme visto no capítulo anterior) representa as ligações entre os registradores do codificador e os somadores XOR. Isto é feito no Simulink escrevendo entre colchetes os coeficientes do polinômio em sistema octal. Como se pode observar na Figura 8, o comprimento de restrição definido é igual a sete e os polinômios geradores são conforme a Equação 11 e Equação 12.

$$\boldsymbol{v_1(X) = 1 + 0X + 0X^2 + X^3 + X^4 + X^5 + X^6 + 0X^7} \quad (11)$$

$$\boldsymbol{v_2(X) = 1 + X + 0X^2 + X^3 + X^4 + 0X^5 + X^6 + 0X^7} \quad (12)$$

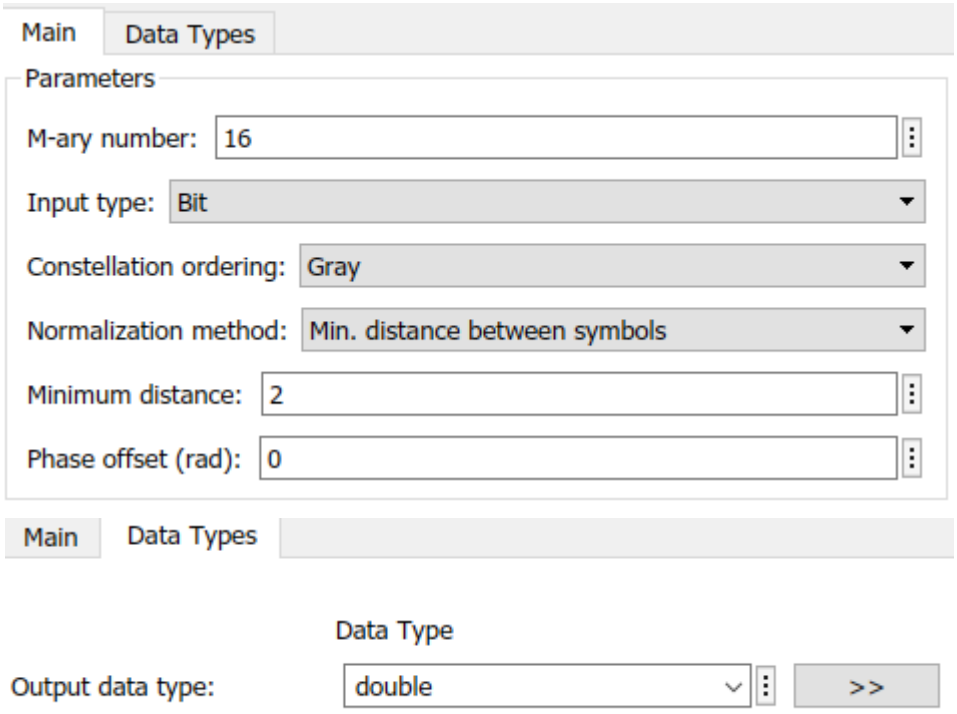
Os coeficientes do vetor $\boldsymbol{v_1}$ em octonário é 171 e o valor dos coeficientes do vetor $\boldsymbol{v_2}$ em octonário é 133. Outros dois parâmetros do bloco são o modo de operação e o código puncionado. O modo de operação foi definido como contínuo, isso significa que à medida que os bits entram no codificador eles são decodificados, não é feito um *buffer* de bits que são posteriormente truncados, e o puncionamento foi habilitado na segunda etapa da simulação. Quando este parâmetro é habilitado ele abre um campo para definir o vetor de puncionamento. Este vetor representa os coeficientes que serão multiplicados pela saída do codificador. Quando o coeficiente é zero significa que aquele bit de saída será suprimido.

3.4 MODULADOR QAM

O modulador QAM realiza a modulação do sinal de entrada utilizando o método de modulação retangular em quadratura e amplitude. No presente trabalho este bloco tem a função de modular o sinal a ser enviado, que foi codificado convolucionalmente.

Os parâmetros deste bloco podem ser observados na Figura 9. O primeiro parâmetro, *M-ary number* define o tamanho da constelação do modulador, o próximo parâmetro *Input Type* define o tipo de dado na entrada, *Constellation ordering* indica como o bloco irá distribuir as *words* binárias com os pontos da constelação, o *Normalization method* define como será a escala da constelação do canal, neste caso foi mantida a configuração padrão do bloco. O parâmetro *Minimum distance* define a distância entre dois pontos mais próximos da constelação. O *Phase offset* define a rotação da constelação do sinal em radianos. O tipo do dado da saída desse bloco foi configurado para *double*.

Figura 9 – Parâmetros do bloco de modulação QAM



Fonte: Simulink

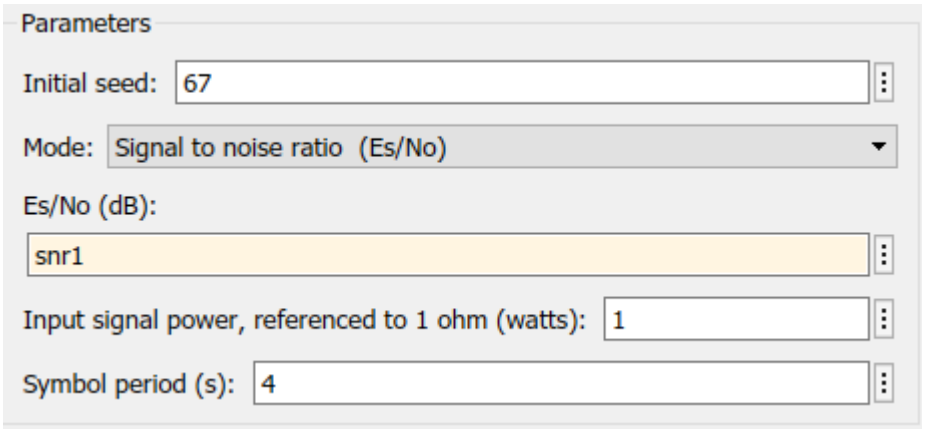
A técnica de modulação em quadratura e amplitude é muito utilizada em sistemas de comunicação digital, por este motivo foi escolhida como técnica de modulação no presente trabalho.

3.5 CANAL AWGN

O canal AWGN é um bloco que simula um canal gaussiano com ruído branco aditivo. No presente trabalho este bloco tem a função de adicionar um ruído gaussiano branco aditivo ao sinal.

Os parâmetros deste bloco podem ser visualizados na Figura 10. O primeiro parâmetro, *Initial Seed* define como será a geração do ruído, neste caso foi utilizada a parametrização padrão do bloco, e, desta forma a geração do ruído apresenta resultados constantes a cada nova simulação que for feita no sistema. O segundo parâmetro, *Mode*, define qual o cálculo utilizado para a relação sinal ruído, e, neste caso foi utilizado o modo E_S/N_0 que utiliza a energia do símbolo da informação como dividendo. O terceiro parâmetro define o valor da relação sinal ruído. Para as simulações esse parâmetro deve variar, portanto é definido como uma variável *snr*. O objetivo é identificar o desempenho dos diferentes sistemas para os diferentes valores deste parâmetro. O quarto parâmetro, *Input signal power, referenced to 1 ohm (watts)*, define a potência média quadrática dos símbolos de entrada, e, neste caso foi mantida a potência padrão de um watt, já que alterações no valor deste parâmetro não ocasionam em alterações significativas no resultado final da simulação do sistema. O quinto parâmetro, *Symbol period*, define o período do símbolo do sinal de entrada. O valor deste parâmetro é definido em segundos e deve ser considerado sem a codificação do canal.

Figura 10 – Parâmetros do bloco Canal AWGN



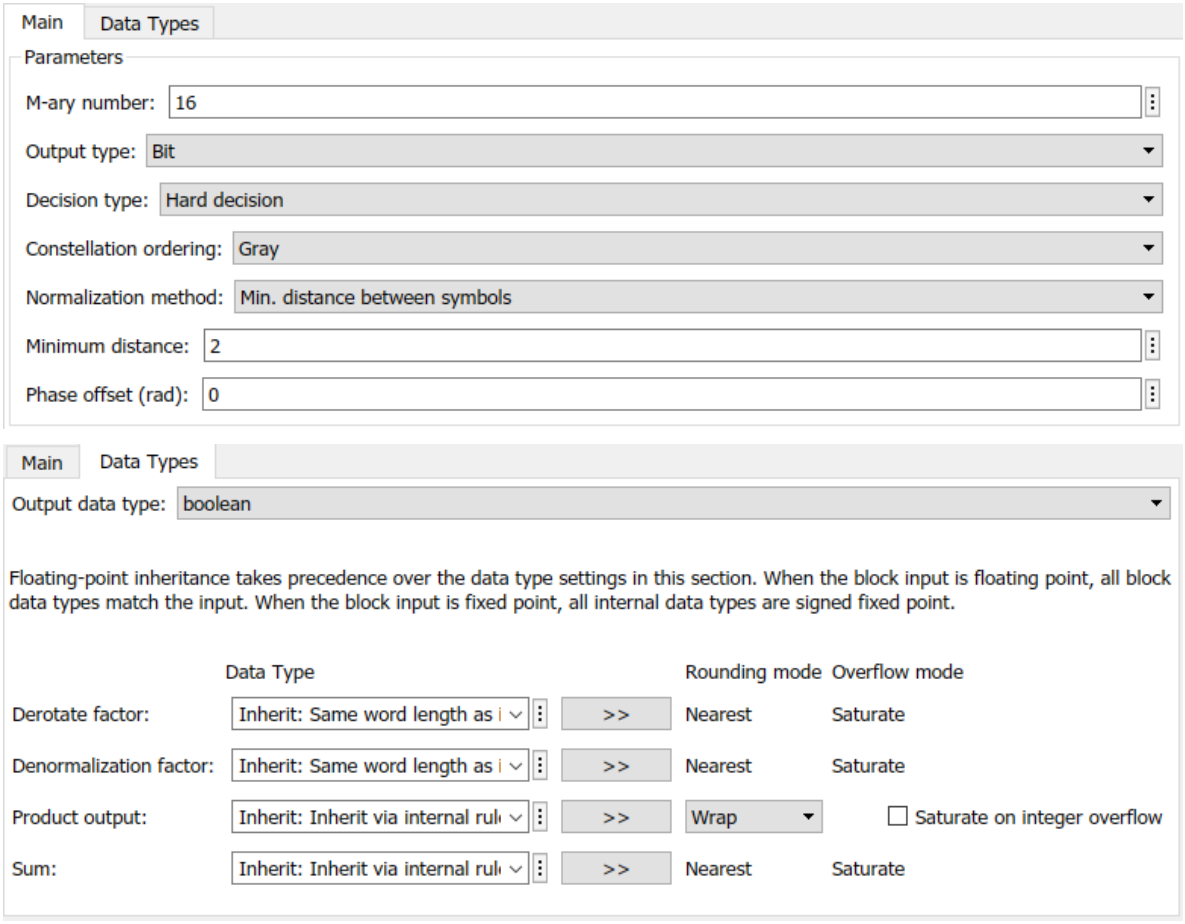
Fonte: Simulink

A escolha de um canal AWGN para a simulação do sistema é motivada por ser um modelo muito utilizado em estudos de canais digitais.

3.6 DEMODULADOR QAM

O demodulador QAM é um bloco utilizado para a demodulação de sinais que foram modulados por um modulador QAM. O bloco tem a função de demodular o sinal codificado enviado pelo canal gaussiano e os seus parâmetros podem ser encontrados na Figura 11. O primeiro parâmetro, *M-ary number* define o tamanho da constelação do demodulador, o próximo parâmetro *Output Type* define o tipo de dado na saída do demodulador. O parâmetro *Decision type* define como será a quantificação do sinal de saída, e, caso seja escolhido o tipo *Hard decision* valores maiores do que 0,5 serão considerados como bit 1 e valores menores do que 0,5 serão quantificados como 0. Caso seja escolhido o tipo *Soft decision* haverá mais níveis de quantificação (no presente trabalho foi utilizado o *Hard decision*). O parâmetro *Constellation ordering* indica como o bloco irá distribuir as *words* binárias com os pontos da constelação, o *Normalization method* define como será a escala da constelação do canal, neste caso foi mantida a configuração padrão do bloco. O parâmetro *Minimum distance* define a distância entre dois pontos mais próximos da constelação. O *Phase offset* define a rotação da constelação do sinal em radianos. O tipo do sinal da saída desse bloco foi configurado para *boolean*. Os outros parâmetros de tipo de dados como *Derotate factor*, *Denormalization factor*, *Product output* e *Sum* definem o tipo de dado que será utilizado para essas informações do bloco, neste caso foi mantida a configuração padrão que é a *Inherit* sendo assim o tipo de dado é definido conforme as variáveis utilizadas na entrada e saída do bloco.

Figura 11 – Parâmetros do bloco Demodulador QAM



Fonte: Simulink

Outros tipos de configurações para o modulador e demodulador podem ser utilizadas. Um fator crítico para o correto funcionamento do sistema é que a configuração do modulador esteja compatível com o demoduador.

3.7 DECODIFICADOR DE VITERBI

O decodificador de Viterbi é um bloco utilizado para decodificar códigos convolucionais através do algoritmo de mesmo nome. No presente trabalho este bloco tem a função de decodificar o sinal enviado pela fonte, ou seja, a sua entrada recebe o sinal codificado que passou pelo canal AWGN e a sua saída apresenta o sinal decodificado. Em suma, em um sistema físico, este bloco estaria no receptor.

Os parâmetros deste bloco podem ser encontrados na Figura 12. O primeiro parâmetro define a estrutura da treliça, conforme já explicado no subcapítulo do bloco de codificação convolucional. Para que a decodificação ocorra corretamente, a estrutura de treliça configurada

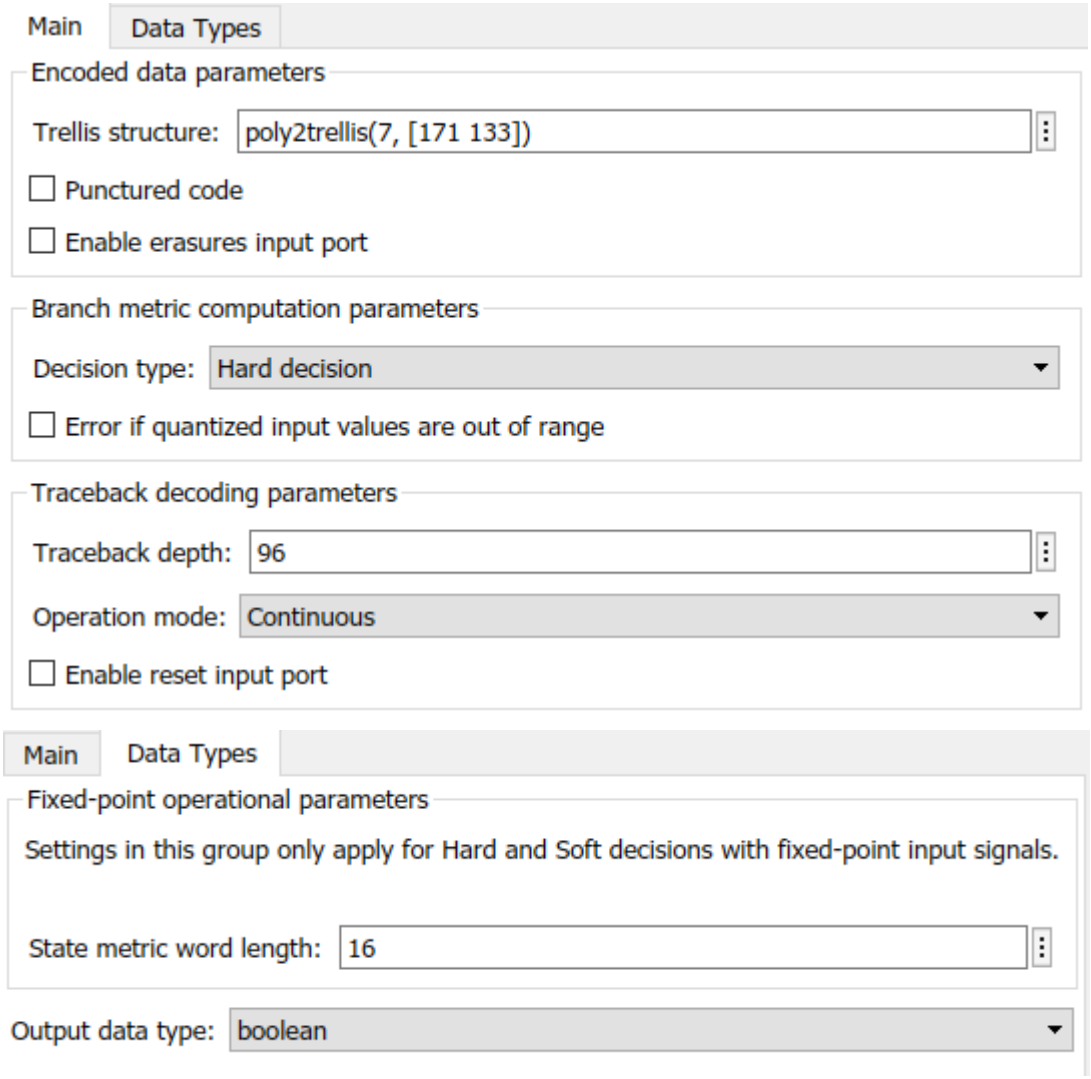
neste bloco deve corresponder com a estrutura definida no bloco de codificação. O segundo parâmetro, *Punctured code*, define se o código recebido foi puncionado ou não, caso esteja habilitado, significa que houve puncionamento. Este parâmetro foi habilitado em um dos experimentos, a ser descrito no capítulo seguinte. O terceiro parâmetro, *Enable merasures input port*, quando acionado, habilita uma porta na entrada do bloco que permite zerar a estrutura de treliça e voltar para o estado inicial. O parâmetro *Decision type* define como foi quantizado o sinal demodulado, conforme foi explanado no subcapítulo do bloco demodulador, o tipo de decisão do canal utilizado para a simulação foi *Hard decision*. O próximo parâmetro, *Error if quantized input values are out of range* se habilitado, emite um erro durante a simulação, caso os valores da entrada estiverem fora da amplitude (conforme definido no tipo de decisão). O parâmetro *Traceback depth*, define o número de ramos necessários na treliça para reconstruir um sinal codificado. Em geral o valor utilizado é calculado conforme a Equação 13 e a taxa de dados do código é calculada conforme a Equação 14.

$$\boldsymbol{Traceback\ Depth} = \mathbf{3} \times \frac{\boldsymbol{Comprimento\ de\ restri}\text{ção}\boldsymbol{o} - \mathbf{1}}{\mathbf{1} - \boldsymbol{taxa\ de\ dados\ do\ c}\text{ó}\boldsymbol{digo}} \tag{13}$$

$$\boldsymbol{Taxa\ do\ c}\text{ó}\boldsymbol{digo} = \frac{\boldsymbol{k}}{\boldsymbol{n}} \times \frac{\boldsymbol{comprimento\ do\ vetor\ de\ punc.}}{\boldsymbol{soma\ do\ vetor\ de\ punc.}} \tag{14}$$

Este parâmetro está relacionado com o atraso na decodificação, que é a quantidade de símbolos nulos que precedem o primeiro símbolo codificado (MOISION. B., 2008). O próximo parâmetro, *Operation mode*, define como será a transição de cada quadro de informação que entra no decodificador. No modo contínuo, o fluxo de informações pela treliça durante toda a simulação é ininterrupto, neste caso o valor configurado no *Traceback depth* será igual ao atraso no decodificador. O tipo de dado na saída *Output data type* foi configurado como *boolean* para corresponder com o tipo de dado definido na saída do bloco *Bernoulli generator*.

Figura 12 – Parâmetros do bloco Decodificador de Viterbi



Fonte: Simulink

O algoritmo de Viterbi foi escolhido como técnica de demodulação no presente trabalho, pois é o método mais utilizado para decodificação convolucional por ser um método de máxima probabilidade e por otimizar o tempo de computação de decodificação em sistemas físicos (PROAKIS, JOHN G., 1996)

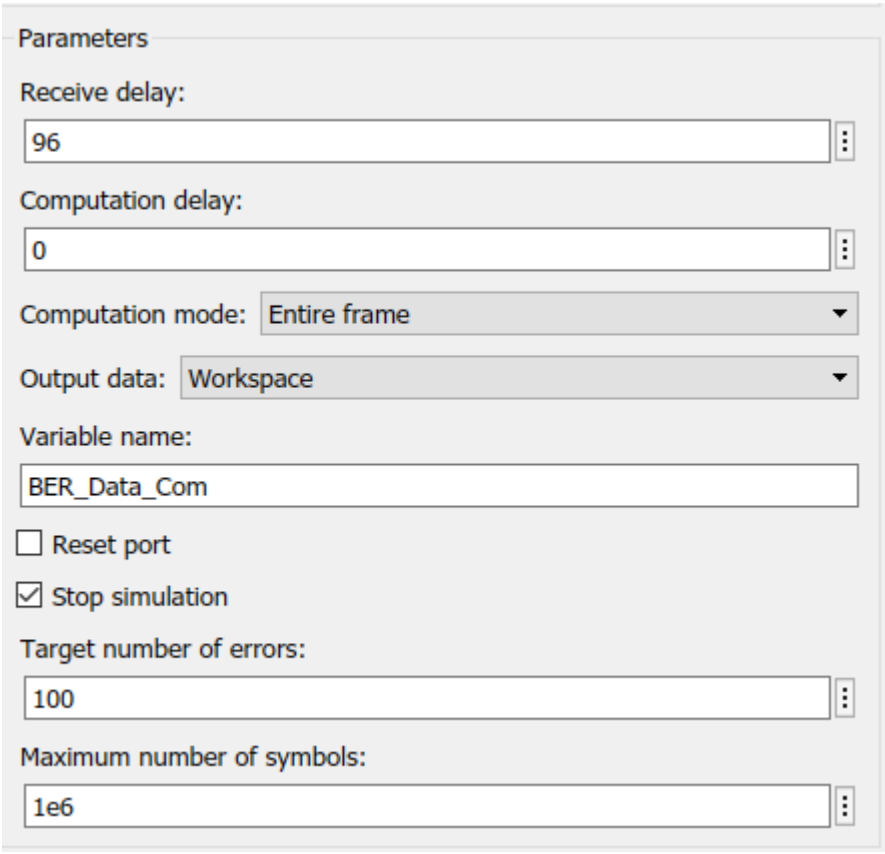
3.8 BER – *BIT ERROR RATE*

O bloco BER é utilizado para calcular a taxa de bits errados entre dois sinais, utilizado para verificar a diferença entre o sinal enviado pela fonte e o sinal recebido no receptor.

Os parâmetros deste bloco podem ser encontrados na Figura 13. O primeiro parâmetro, *Receive delay*, define a quantidade de atraso que existe entre o sinal enviado e o recebido, o

valor configurado é incluído como atraso no sinal transmitido antes de começar a comparação entre os sinais, para que ambos estejam sincronizados. Este parâmetro deve ser igual ao atraso na decodificação ou *Traceback depth*. O segundo parâmetro, *Computation delay*, define o número de amostras que o bloco irá desconsiderar no início da comparação, neste caso esse valor foi configurado como nulo. O parâmetro *Computation mode* define se o bloco deve considerar todo o *frame* do sinal na entrada do bloco ou apenas uma parte. No presente trabalho este parâmetro é utilizado com a configuração *Entire frame*, que considera toda a informação na entrada do bloco. O parâmetro *Output data* define para onde será enviada a saída deste bloco, onde existem duas opções *Workspace* e *Port*, ambas as configurações foram utilizadas no experimento. A opção *Workspace* permite que o resultado seja enviado para a área de trabalho do MATLAB, e a opção *Port* permite que os resultados sejam visualizados diretamente no ambiente de simulação do Simulink. Quando a opção *Workspace* é utilizada, o parâmetro *Variable Name* é habilitado e este define o nome da variável que irá receber a informação da saída do bloco. O parâmetro *Reset port* habilita uma porta no bloco que, quando acionada, zera as informações da saída do bloco. No presente trabalho este parâmetro não é habilitado. O parâmetro *Stop Simulation* quando habilitado define premissas para o fim da simulação do sistema, e essas premissas são definidas pelos parâmetros *Target number of errors* e *Maximum number of symbols* e quando um desses parâmetros é alcançado, a simulação é interrompida. O primeiro parâmetro define o número máximo de bits errados e o segundo parâmetro define o número máximo de símbolos computados pelo bloco.

Figura 13 – Parâmetros do bloco BER



Fonte: Simulink

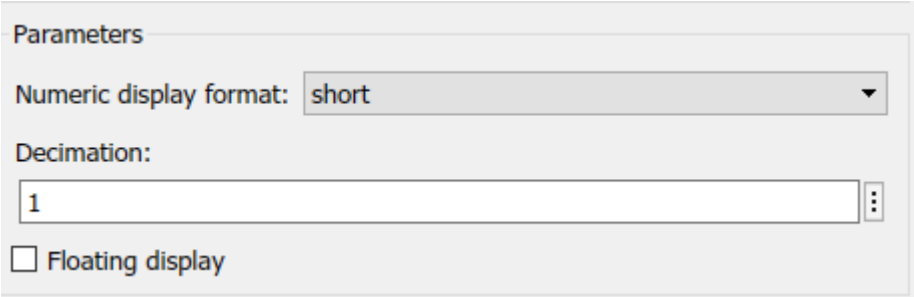
O bloco BER foi escolhido como método de análise para a qualidade da codificação do sinal, escolha baseada na frequente utilização dessa técnica em trabalhos acadêmicos e sistemas físicos, como por exemplo, para avaliação da qualidade de enlaces de TV a cabo.

3.9 DISPLAY

O bloco display é utilizado para visualizar no ambiente de simulação do Simulink, o resultado de um determinado bloco. No presente trabalho este bloco tem a função de auxiliar na visualização dos resultados das simulações realizadas manualmente no Simulink. Ao utilizar um código de automatização, a ser explicado em detalhes no próximo capítulo, quando os resultados são enviados para o espaço de trabalho do MATLAB, o bloco Display não é utilizado. Os parâmetros deste bloco podem ser encontrados na Figura 14. O primeiro parâmetro *Numeric display format* define o tipo de variável dos números na saída, neste caso foi definido como *short*. O segundo parâmetro, *Decimation*, define o período de tempo no qual o dado aparece, e, no caso deste trabalho, foi mantida a configuração padrão do bloco “1” que define

que o dado será atualizado a cada passo de tempo da simulação. O parâmetro *Floating display* não foi habilitado.

Figura 14 – Parâmetros do bloco Display



Fonte: Simulink

4 DESENVOLVIMENTO DA PROPOSTA

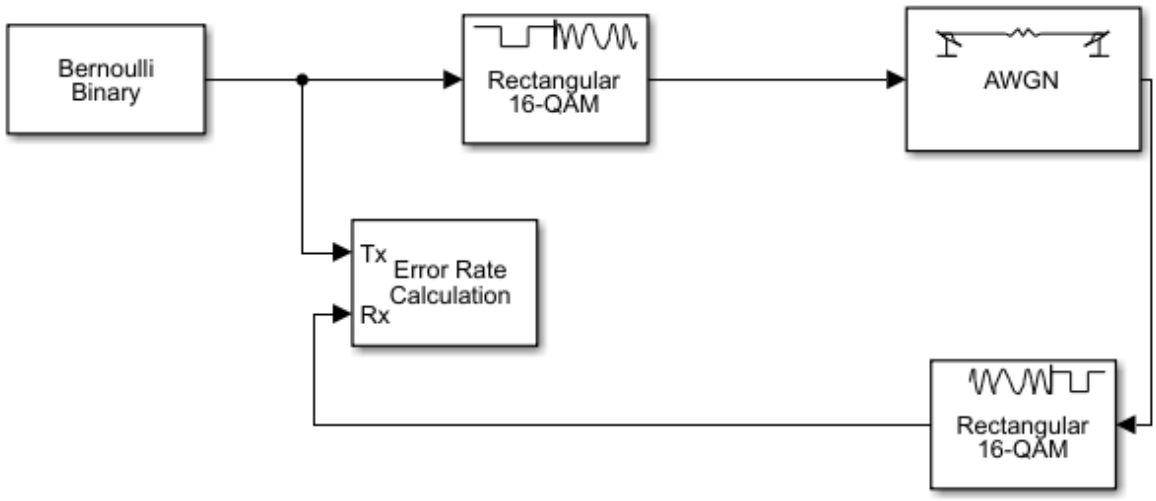
O presente capítulo tem como objetivo explanar a solução proposta, assim como descrever as premissas consideradas e os sistemas simulados.

A proposta consiste em simular a codificação convolucional em um canal de comunicação AWGN com modulação da banda base em 16-QAM. Um sinal aleatório é produzido, codificado convolucionalmente, modulado com a técnica 16-QAM e enviado em um canal AWGN. Após, é demodulado, decodificado e, por fim, o sinal resultante é comparado com o enviado através do cálculo de BER. A simulação é feita para diversas situações de SNR. Em uma segunda fase é utilizada a técnica de puncionamento para verificar a relação entre a taxa de dados e o BER para diferentes situações de SNR. O objetivo é demonstrar que para canais com baixa relação sinal ruído é vantajoso utilizar a técnica de puncionamento para suprimir bits de paridade e assim, consumir menos banda do canal. Aqui será descrita a metodologia utilizada na realização da simulação, assim como uma explanação das considerações e ferramentas utilizadas nas simulações.

4.1 SISTEMA SEM CODIFICAÇÃO

O primeiro sistema a ser simulado não tem a presença de codificação de canal, são utilizados apenas os blocos de geração da informação, modulação, demodulação, bloco do canal AWGN e o BER. O sinal enviado é uma sequência binária de 64 bits por segundo, e esse sinal é modulado com a técnica 16-QAM, passa por um canal AWGN, é demodulado e por fim, o sinal recebido é comparado com o sinal enviado. Essa comparação é feita bit a bit e cada bit errado é contabilizado para o indicador BER. A simulação é interrompida caso o bloco BER contabilize cem erros ou caso ultrapasse mais de 10^6 símbolos contabilizados. A simulação é repetida para diversos valores de SNR. O SNR é variado de 0 dB até 10 dB com passos de 0,1dB, e para cada valor de SNR o BER é verificado. O diagrama de blocos do sistema simulado pode ser encontrado na Figura 15.

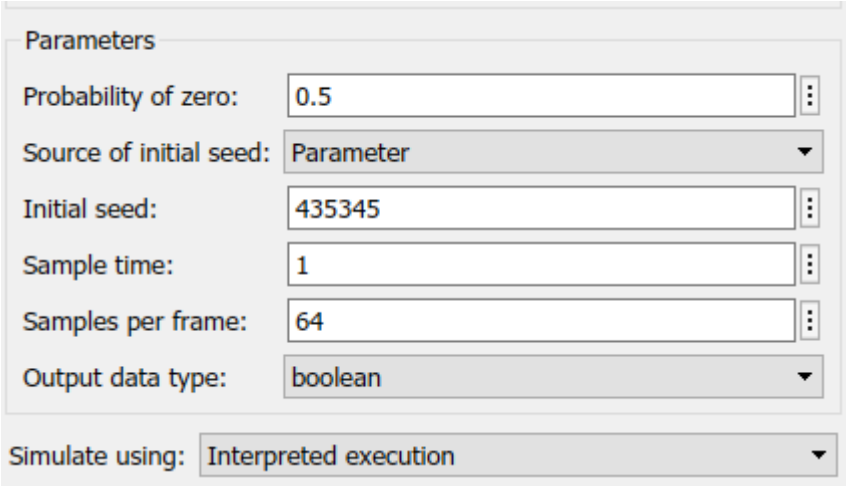
Figura 15 – Diagrama de blocos do sistema sem codificação convolucional



Fonte: Simulink

Os parâmetros do bloco *Bernoulli binary* configurados para este sistema podem ser encontrados na Figura 16. Nesta configuração o sinal gerado pelo bloco é de 64 bits por segundo, com probabilidade de zero igual a 0,5. É importante ressaltar que o número de bits por amostra deve ser um múltiplo de 16, caso contrário ocorrerá um erro ao modular o sinal com uma constelação de 16 níveis.

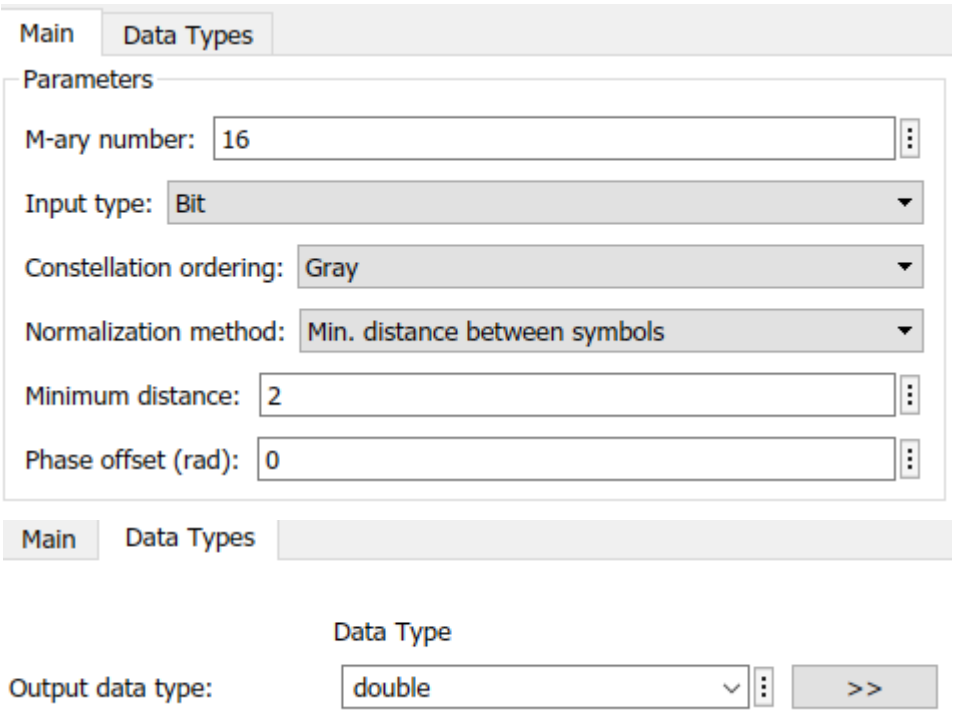
Figura 16 – Parâmetros do bloco Bernoulli Binary para o sistema sem codificação



Fonte: Simulink

Os parâmetros do bloco *Rectangular 16-QAM* podem ser encontrados na Figura 17. Esta configuração permite que o sinal digital seja modulado para analógico, e, após a modulação, o sinal passa pelo canal AWGN.

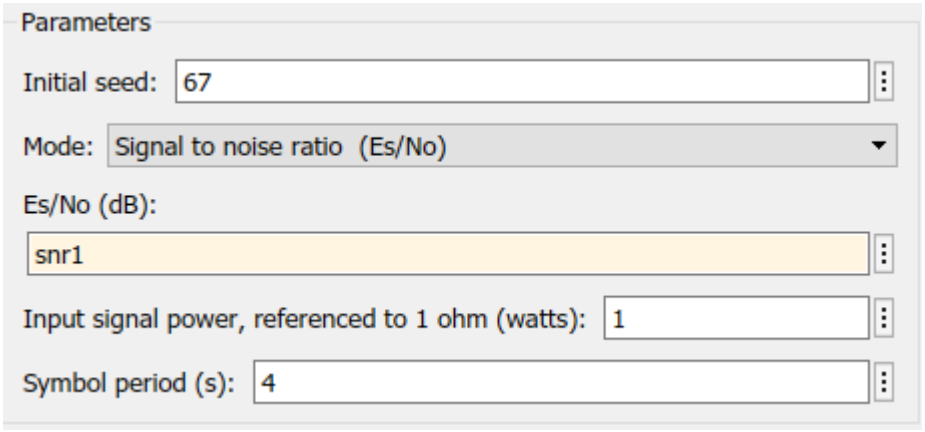
Figura 17 – Parâmetros do bloco Modulador 16-QAM para o sistema sem codificação.



Fonte: Simulink

Os parâmetros do bloco *AWGN Channel* podem ser observados na Figura 18. Ao parâmetro Es/No foi atribuída a variável *snr1*. Esta variável é um vetor de 100 posições com início em 0 até 10 com passos de 0,1, portanto, a relação sinal ruído do canal irá variar conforme a variável, iniciando em 0dB até chegar em 10dB. O parâmetro *Symbol period* será de 4 segundos, este é o período de duração do símbolo modulado, visto que, para cada período de tempo, o modulador QAM produz 16 símbolos, totalizando 64 em 4 períodos de tempo.

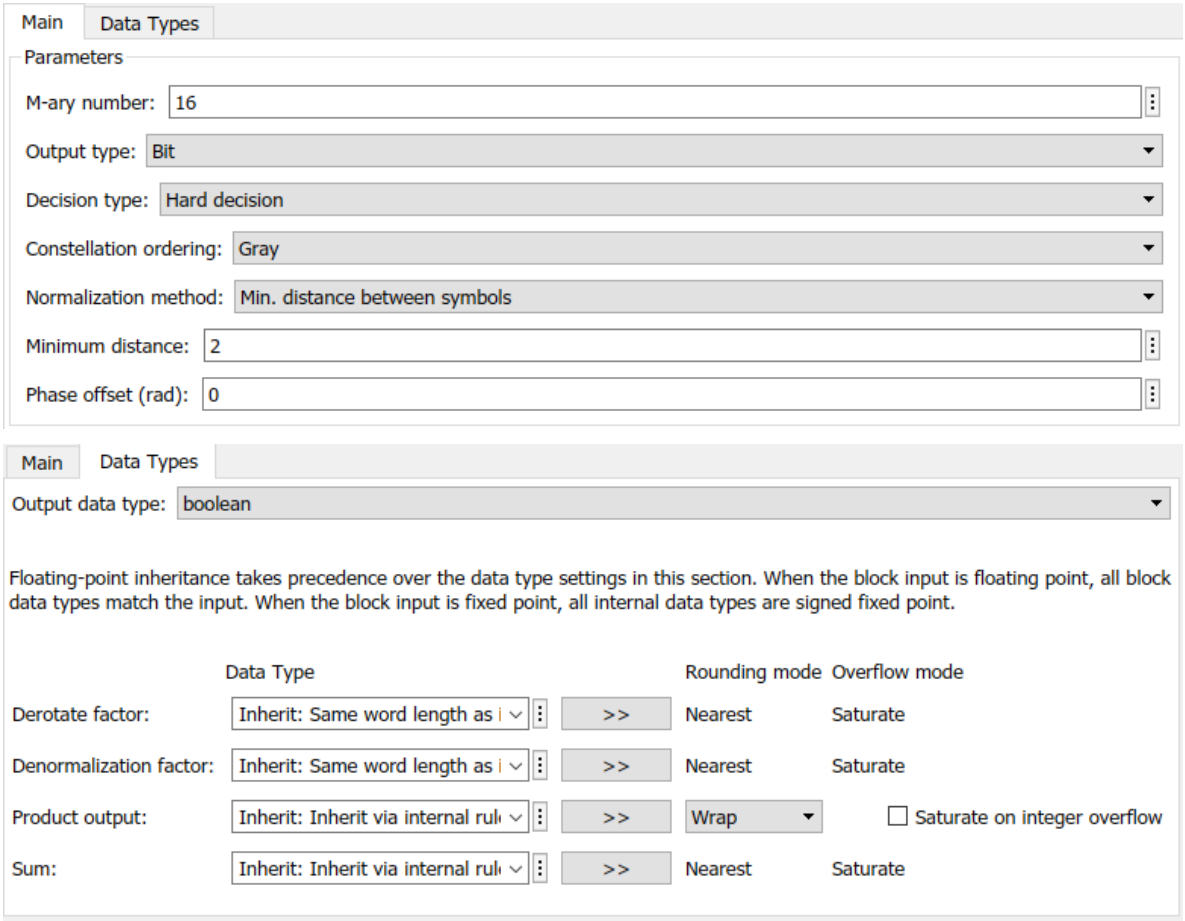
Figura 18 – Parâmetros do bloco canal AWGN para o sistema sem codificação.



Fonte: Simulink

Os parâmetros do bloco de demodulação 16-QAM do sistema sem codificação podem ser encontrados na Figura 19. Os parâmetros de demodulação devem coincidir com a configuração da modulação.

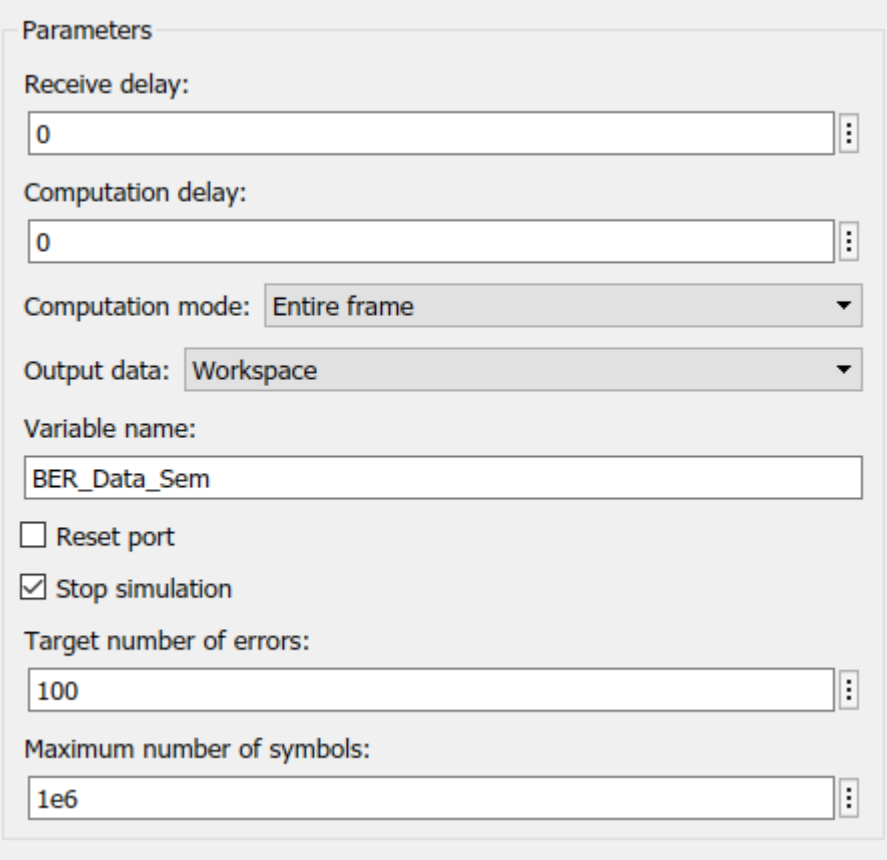
Figura 19 – Parâmetros do bloco de demodulação 16-QAM para o sistema sem codificação.



Fonte: Simulink

O último bloco desse sistema *Error Rate Calculation* irá contabilizar os erros entre o sinal enviado e o recebido, os parâmetros deste bloco para esse sistema podem ser encontrados na Figura 20.

Figura 20 – Parâmetros do bloco BER para o sistema sem codificação



Fonte: Simulink

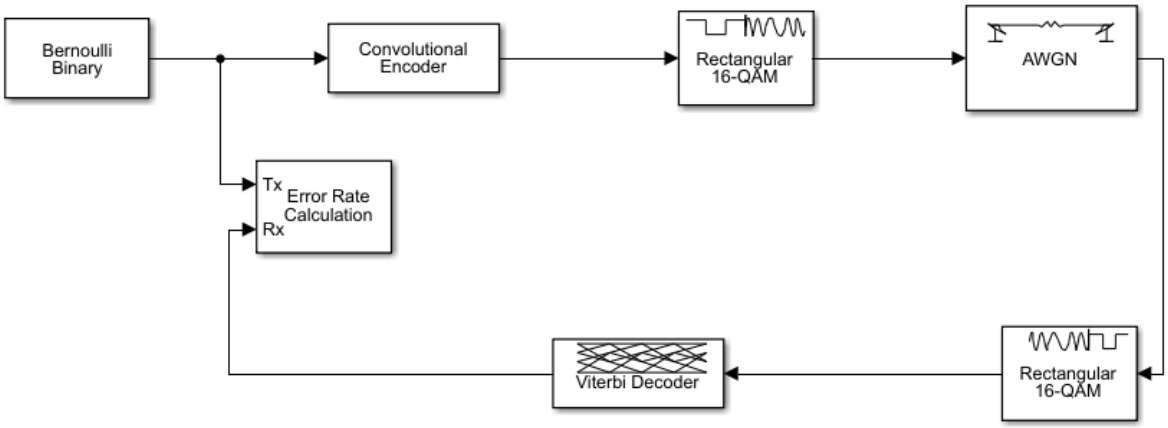
Para o sistema sem codificação não há atraso entre o sinal enviado e o recebido, portanto o parâmetro *Receive delay* é nulo, e os resultados deste bloco devem ser armazenados para que possam ser plotados, por isso o parâmetro *Output data* é configurado para *Workspace* e uma variável será atribuída à saída, sendo esta variável denominada *Ber_Data_Sem*. O tempo de simulação no ambiente do Simulink é ajustado para "infinito", com interrupção ocorrendo quando o bloco BER contabiliza 100 erros ou analisar mais do que 10^6 símbolos.

Para o sistema sem codificação o objetivo é mapear e plotar o resultado do BER (que é a variável *Ber_Data_Sem*) versus a relação sinal ruído (variável *snr1*), os resultados deste sistema serão comparados com os sistemas com codificação e puncionamento para demonstrar que para altos valores de relação sinal ruído o sistema sem codificação apresenta desempenho inferior ao sistemas com codificação.

4.2 SISTEMA COM CODIFICAÇÃO CONVOLUCIONAL

O diagrama de blocos do sistema com codificação covolucional pode ser encontrado na Figura 21. Neste sistema foram adicionados os blocos de codificação convolucional e decodificação de Viterbi.

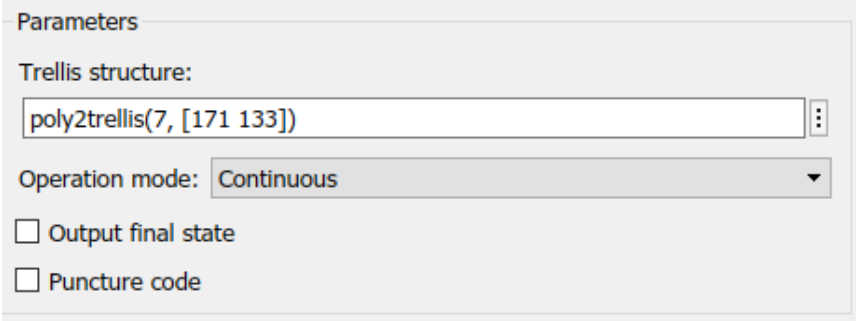
Figura 21 – Diagrama de blocos do sistema com codificação convolucional



Fonte: Simulink

Os parâmetros do bloco *Bernoulli binary* podem ser encontrados na Figura 16. Este bloco foi parametrizado para que o sinal do sistema sem codificação seja idêntico ao do sistema com codificação, dessa forma a comparação fica mais fidedigna, visto que a sequência de bits enviados é a mesma. Os parâmetros do bloco de codificação convolucional podem ser encontrados na Figura 22.

Figura 22 – Parâmetros do bloco codificador convolucional para o sistema com codificação.



Fonte: Simulink

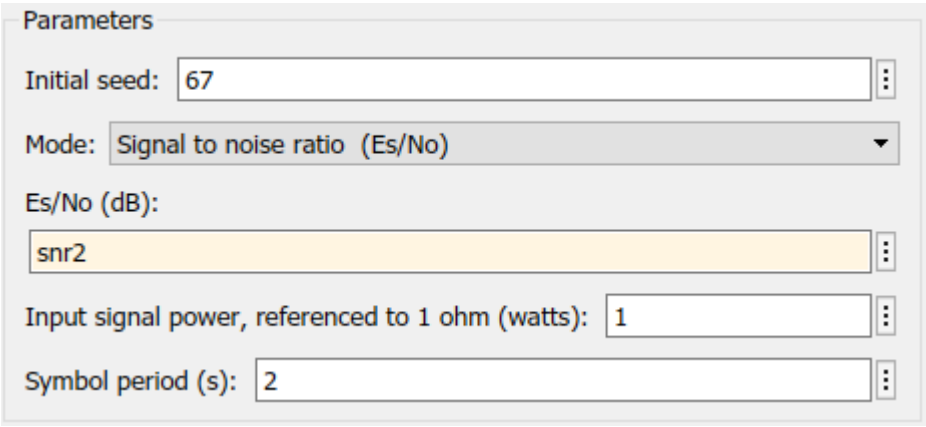
Conforme mencionado anteriormente, a função *poly2trellis* define a estrutura do codificador. Portanto, para este sistema, o codificador convolucional terá os seguintes

parâmetros: o comprimento de restrição (*K*) do codificador é 7, o número de bits na entrada é 64 e o número de bits na saída é 128. Os coeficientes dos polinômios geradores, em octonário são 171 e 133, convertendo para binário, os valores ficam 01111001 e 01011011. Neste sistema não é utilizado o puncionamento, então não haverá supressão de bits de paridade.

O bloco de modulação 16-QAM é configurado com os mesmos parâmetros utilizados no sistema sem codificação, estes parâmetros podem ser encontrados na Figura 17. Um ponto crítico na modulação do sinal para este sistema é que o sinal após a codificação deve ser um múltiplo de 16, caso contrário ocorrerá um erro na simulação.

Os parâmetros do bloco canal AWGN podem ser encontrados na Figura 23, ao parâmetro Es/No foi atribuída a variável *snr2*. Esta variável é um vetor de 100 posições com início em 0 até 10 com passos de 0,1, portanto, a relação sinal ruído do canal irá variar conforme a variável, iniciando em 0dB até chegar em 10dB. O parâmetro *Symbol period* é de 2 segundos, este é o período mínimo necessário para enviar o símbolo de informação do canal.

Figura 23 – Parâmetros do bloco Canal AWGN para o sistema com codificação



Fonte: Simulink

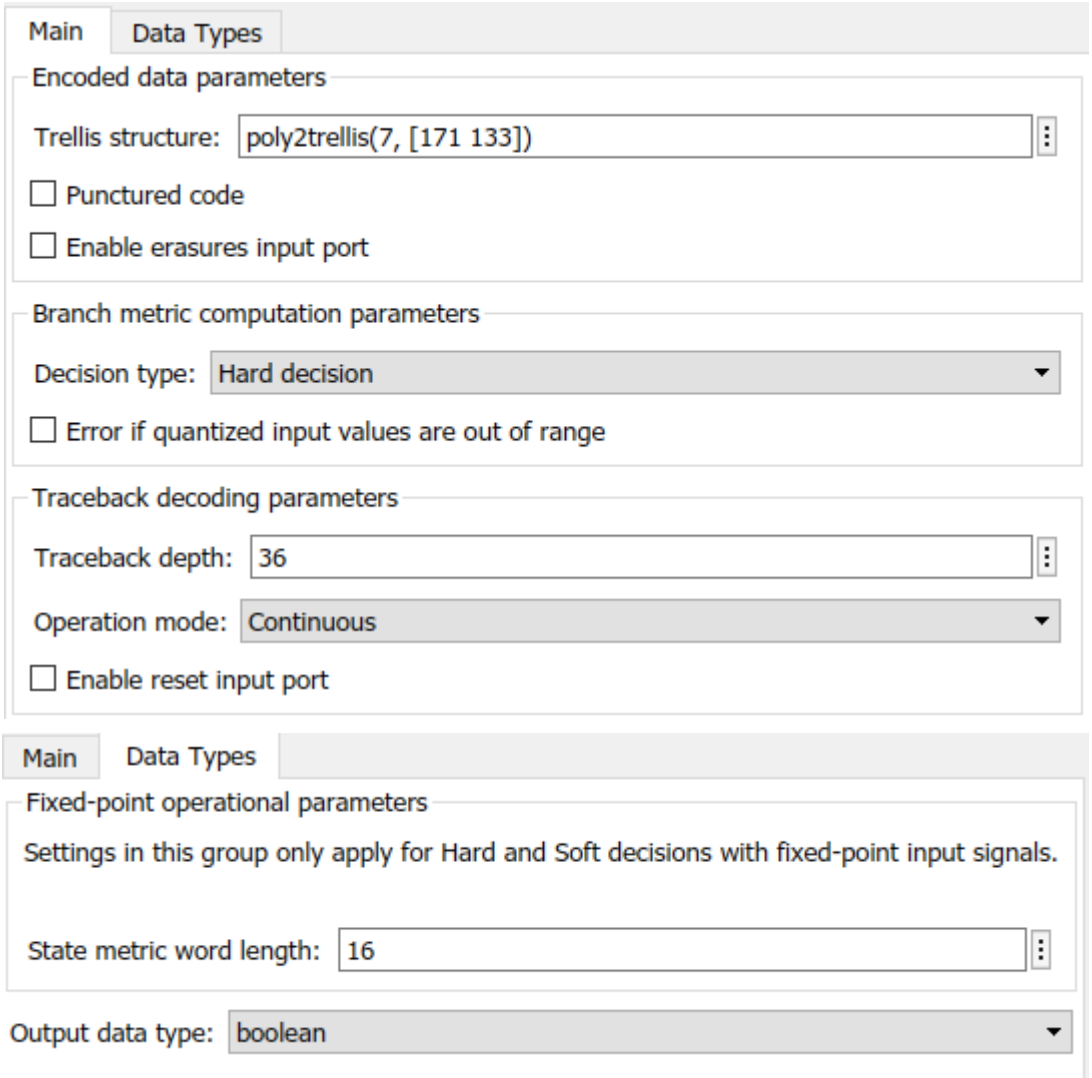
Os parâmetros do bloco de demodulação 16-QAM podem ser encontrados na Figura 19. A configuração deste bloco é idêntica à configuração do sistema sem codificação, não há diferença na demodulação do sinal codificado ou não codificado.

No sistema com codificação é adicionado o decodificador de Viterbi e este bloco faz a decodificação do sinal cujos parâmetros deste bloco podem ser encontrados na Figura 24. O parâmetro *Trellis structure* é configurado com a mesma estrutura do codificador convolucional, neste caso, o decodificador espera receber um sinal que passou por um codificador com comprimento de restrição (*K*) igual a 7 e taxa de dados de ½. Para este sistema não há

puncionamento, portanto não é habilitado o parâmetro *Punctured Code*. O parâmetro *Traceback depth* é definido conforme a Equação 13 e o seu cálculo pode ser encontrado na Equação 15.

$$\boldsymbol{Traceback\ Depth = 3 \times \frac{7-1}{1-0,5} = 3 \times \frac{6}{0,5} = 36} \tag{15}$$

Figura 24 – Parâmetros do bloco decodificador de Viterbi para o sistema com codificação

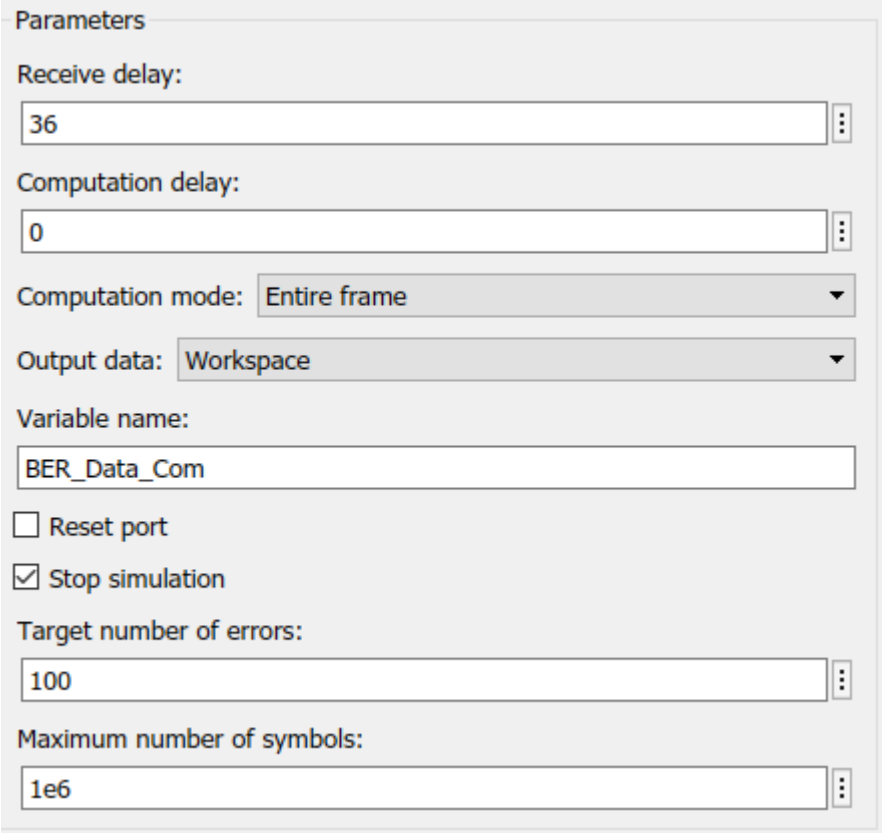


Fonte: Simulink

O próximo, e último, bloco deste sistema é o BER, cujos parâmetros deste bloco podem ser encontrados na Figura 25. O parâmetro *Receive delay* é configurado com o valor do *Traceback depth* do decodificador de Viterbi. Os resultados deste bloco devem ser armazenados para que possam ser plotados, por isso o parâmetro *Output data* será configurado para *Workspace* e uma variável será atribuída à saída, esta variável será denominada como

Ber_Data_Com. O tempo de simulação no ambiente do Simulink é definido como infinito, a interrupção ocorre quando o bloco BER contabilizar 100 erros ou analisar mais do que 10^6 símbolos.

Figura 25 – Parâmetros do bloco BER para o sistema com codificação



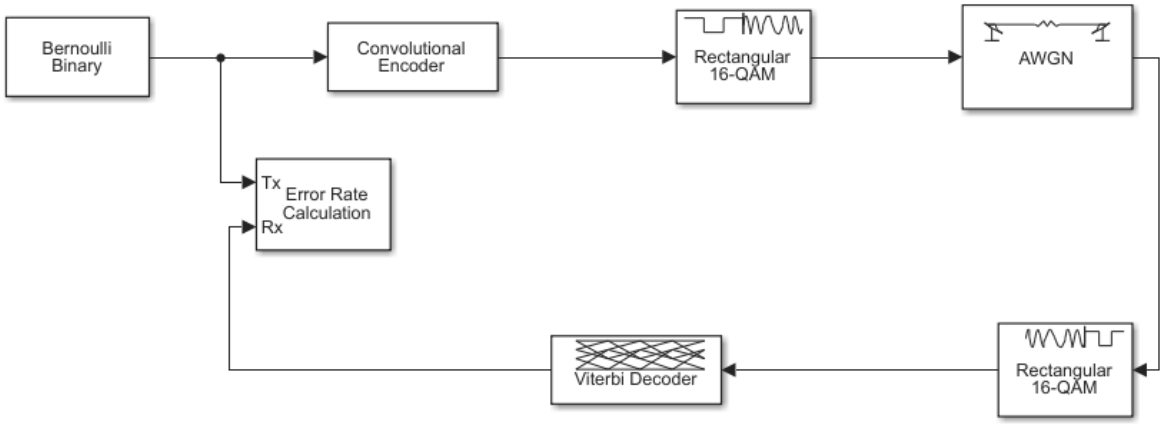
Fonte: Simulink

Para este sistema de comunicação digital com codificação convolucional espera-se obter um desempenho melhor do que o sistema sem codificação de canal. Os resultados obtidos no bloco BER são plotados para cada valor de relação sinal ruído simulado. O resultados são apresentados e discutidos no próximo capítulo.

4.3 SISTEMA COM CODIFICAÇÃO CONVOLUCIONAL E PUNCIONAMENTO

Neste sistema foi habilitado o puncionamento do codificador convolucional e do decodificador de Viterbi. O diagrama de blocos pode ser visto na Figura 26, e este sistema possui os mesmos blocos do que o sistema anterior, no qual havia codificação convolucional, porém sem puncionamento. A grande diferença neste caso está na configuração dos blocos, principalmente do codificador convolucional e do decodificador de Viterbi.

Figura 26 – Diagrama de blocos do sistema com codificação e puncionamento

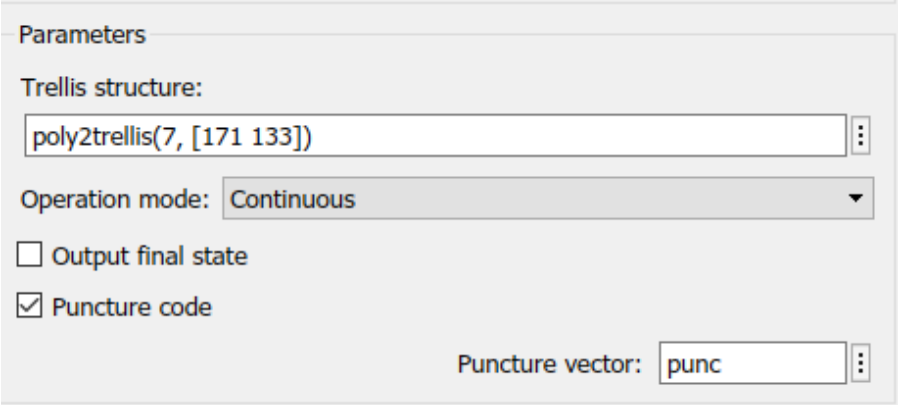


Fonte: Próprio Autor

Os parâmetros do bloco *Bernoulli binary* permanecem os mesmos dos sistemas anteriores e podem ser encontrados na Figura 16, e, conforme mencionado anteriormente, não há diferenciação do sinal enviado da fonte entre os sistemas.

Os parâmetros do bloco de codificação convolucional podem ser encontrados na Figura 27, onde a principal diferença na configuração está na habilitação da função *Puncture Code*.

Figura 27 – Parâmetros do bloco de codificação convolucional para o sistema com puncionamento.



Fonte: Simulink

É possível configurar o vetor de puncionamento através do parâmetro *Puncture vector*, onde este parâmetro é definido com a variável *punc*. São simulados três possíveis vetores, conforme as Equações 16, 17 e 18, onde o vetor que apresentar o melhor desempenho em função da SNR será utilizado para fins de comparação com os sistemas anteriores.

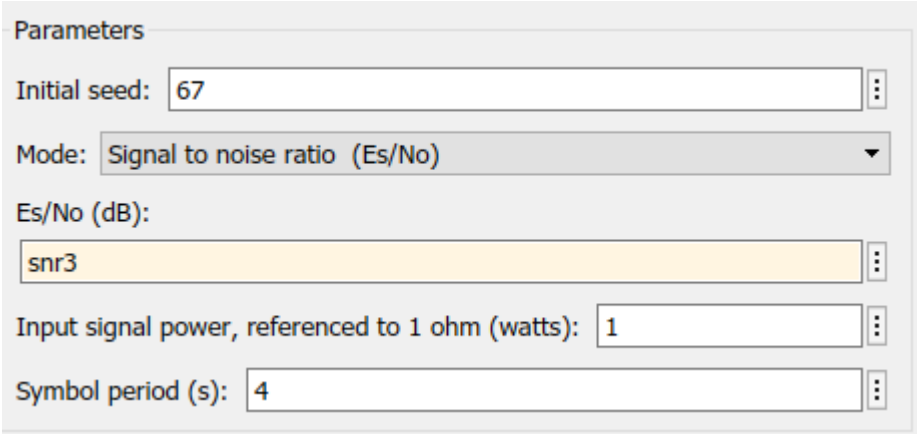
$$\boldsymbol{punc} = [\mathbf{0\ 1\ 0\ 1\ 0\ 1\ 1\ 1}] \tag{16}$$

$$\boldsymbol{punc} = [\mathbf{1\ 1\ 0\ 1\ 0\ 1\ 0\ 1}] \tag{17}$$

$$\boldsymbol{punc} = [\mathbf{1\ 0\ 1\ 0\ 1\ 0\ 1\ 1}] \tag{18}$$

Os parâmetros do bloco de modulação QAM permanecem iguais aos dos sistemas anteriores e estes podem ser encontrados na Figura 17. Os parâmetros do bloco canal AWGN podem ser encontrados na Figura 28, onde, ao parâmetro Es/No foi atribuída a variável *snr3*. Esta variável será um vetor de 100 posições com início em 0 até 10 com passos de 0,1, portanto, a relação sinal ruído do canal irá variar conforme a variável, iniciando em 0dB até chegar em 10dB. O parâmetro *Symbol period* é de 4 segundos sendo este o período mínimo necessário para enviar o símbolo de informação do canal.

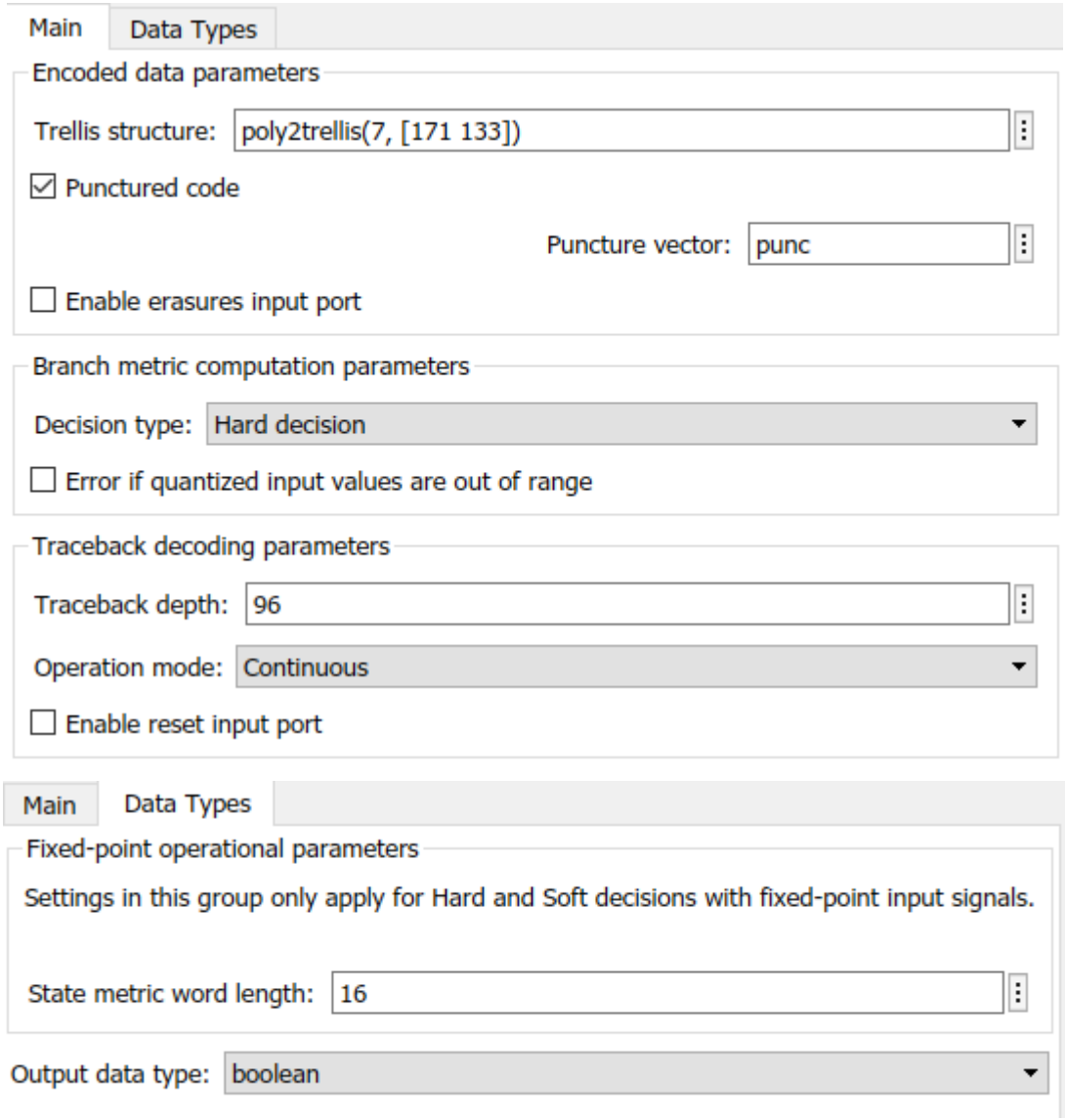
Figura 28 – Parâmetros do bloco canal AWGN para o sistema com puncionamento



Fonte: Simulink

Os parâmetros do bloco de demodulação QAM permanecem iguais aos dos sistemas anteriores e podem ser encontrados na Figura 19. Os parâmetros do bloco decodificador de Viterbi podem ser encontrados na Figura 29, as principais alterações na configuração deste bloco com relação ao sistema anterior são a habilitação do puncionamento, a atribuição da variável *punc* ao parâmetro *Puncture vector* e a alteração do *Traceback depth*.

Figura 29 – Parâmetros do decodificador de Viterbi para o sistema com puncionamento



Fonte: Simulink

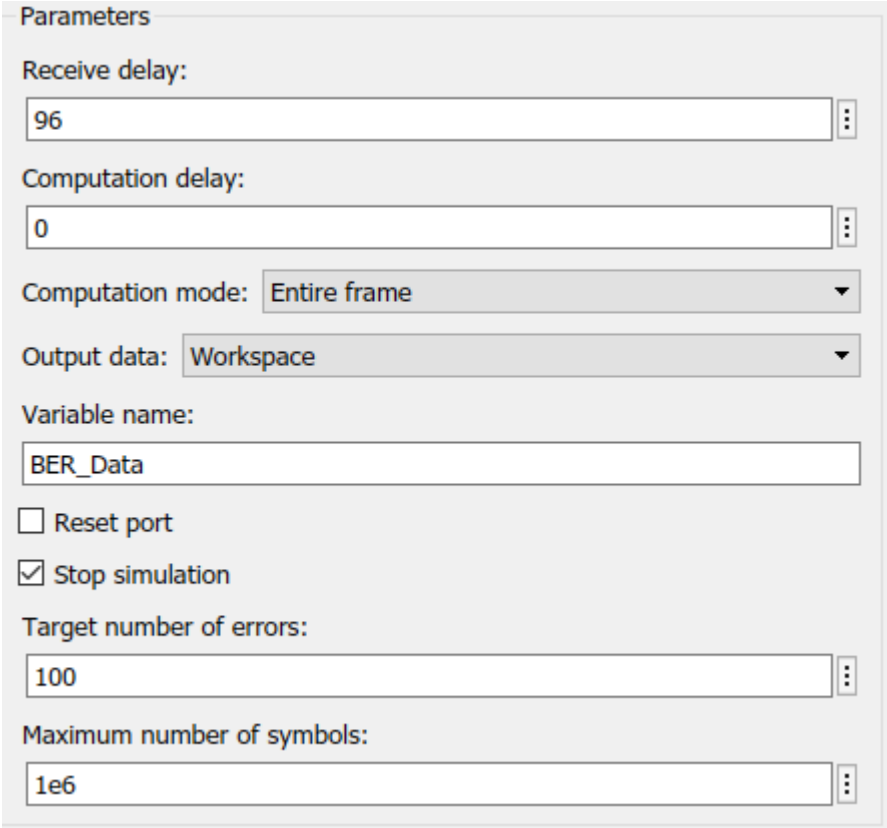
Conforme visto na Equação 13 e Equação 14, o *Traceback depth* depende da taxa de dados do código, que neste caso será de 4/5, desta forma o seu valor será calculado conforme a Equação 19. Este é o valor mínimo necessário para o *Traceback Depth*, porém no sistema será adicionado mais 6 segundos.

$$\boldsymbol{Traceback\ Depth} = \mathbf{3} \times \frac{\mathbf{7-1}}{\mathbf{1-4/5}} = \mathbf{3} \times \frac{\mathbf{6}}{\mathbf{0,2}} = \mathbf{90} \tag{19}$$

O último bloco deste sistema é o BER, e os parâmetros deste bloco podem ser encontrados na Figura 30. O parâmetro *Receive delay* é configurado com o valor do *Traceback depth* do decodificador de Viterbi. Os resultados deste bloco devem ser armazenados para que

possam ser plotados, por isso o parâmetro *Output data* é configurado para *Workspace* e uma variável será atribuída à saída e esta variável será denominada como *Ber_Data*. O tempo de simulação no ambiente do Simulink é definido como infinito, e a interrupção ocorrerá quando o bloco BER contabilizar 100 erros ou analisar mais do que 10^6 símbolos.

Figura 30 – Parâmetros do bloco BER para o sistema com puncionamento



Fonte: Simulink

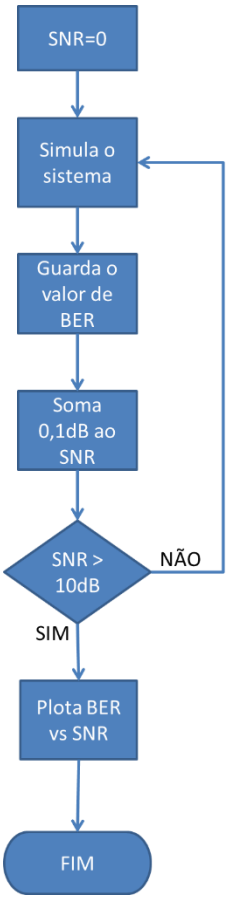
O que se espera deste sistema é atender um meio termo entre o sistema sem codificação e o sistema com codificação, e a expectativa é obter uma taxa de erro intermediária, porém com um uso da banda do canal menor, visto que a taxa de dados do canal deste sistema é maior.

4.4 CÓDIGO DE SIMULAÇÃO COM VARIAÇÃO AUTOMÁTICA DO SNR

Foi desenvolvido no MATLAB uma rotina para automatizar a simulação dos diagramas de bloco dos sistemas apresentados nos subcapítulos anteriores. Na Figura 31 pode ser observado um fluxograma resumindo a função desse código. O objetivo é repetir a simulação do sistema para os diferentes valores de relação sinal ruído e, para cada situação, armazenar o BER. Por

fim é gerado um gráfico com os valores de BER versus SNR. O código pode ser encontrado no APÊNDICE A.

Figura 31 – Fluxograma do código de automatização da simulação



Fonte: Próprio autor

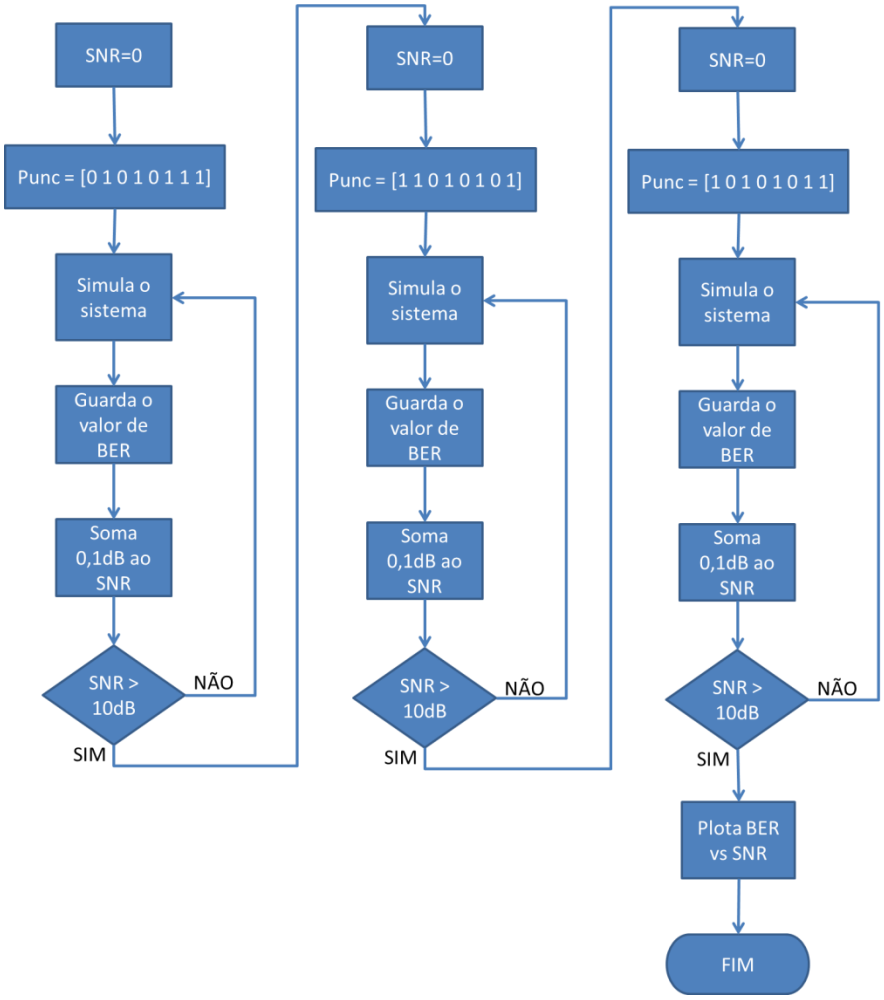
A rotina utiliza as funções *load_system* e *simOut* para acessar o sistema no Simulink e simular o sistema com os parâmetros atualizados, quando, após a simulação, os valores de BER são armazenados em um vetor e cada posição do vetor é dedicada para uma iteração da simulação. Este código foi utilizado com todas as abordagens apresentadas anteriormente.

4.5 CÓDIGO DE SIMULAÇÃO COM VARIAÇÃO AUTOMÁTICA DO PADRÃO DE PUNCIONAMENTO

Na Figura 32 pode ser encontrado um fluxograma que representa o código desenvolvido para encontrar o vetor de puncionamento com o melhor desempenho. Esta rotina repete a simulação do sistema com puncionamento para os três diferentes vetores definidos pelas

Equações 15, 16 e 17. Esses padrões são simulados para uma variação de SNR de 0 dbB até 10dB com passos de 0,1dB. Por fim, os valores de BER versus SNR são plotados sobrepostos no mesmo gráfico para poder comparar o padrão que obteve melhor resultado. O código desta rotina pode ser encontrado no APÊNDICE B.

Figura 32 – Fluxograma do código para encontrar o vetor de puncionamento com melhor desempenho



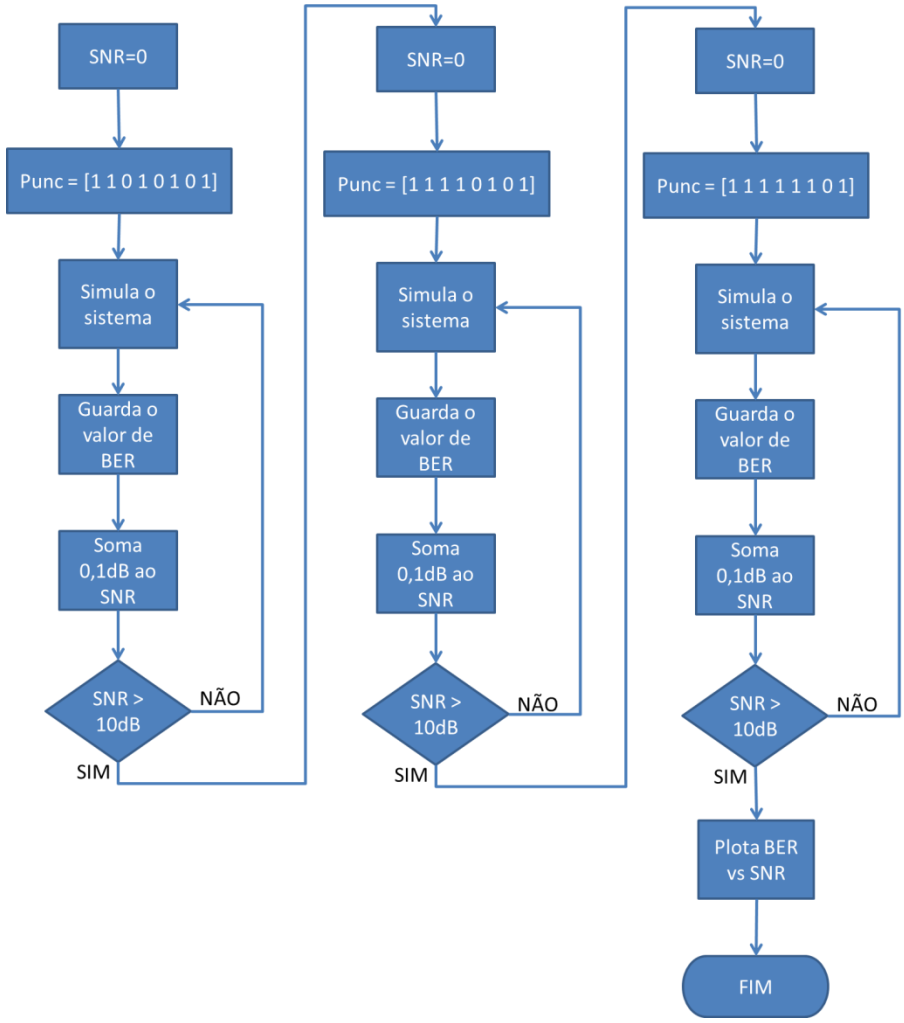
Fonte: Próprio autor.

Este código altera o padrão de puncionamento alterando o valor da variável *punc*, sendo essa atribuída ao parâmetro de vetor de puncionamento do bloco de codificação convolucional e decodificador de Viterbi.

4.6 CÓDIGO DE VARIAÇÃO DA TAXA DE CÓDIGO

Com base no código de variação do padrão de puncionamento, foi desenvolvido um código para avaliar o desempenho do sistema com variações na taxa do código. O fluxograma que representa essa rotina pode ser encontrado na Figura 33.

Figura 33 – Fluxograma do código de variação da taxa de código



Fonte: Próprio Autor

Foram utilizadas três taxas de código e os vetores de puncionamento correspondentes para cada taxa podem ser encontrados no Quadro 4. Os padrões destacados no Quadro 4 são simulados para uma variação de SNR de 0 dB até 10dB com passos de 0,1dB. Por fim os valores de BER versus SNR são plotados sobrepostos no mesmo gráfico para poder comparar o desempenho do sistema para diferentes taxas de código. O código desta rotina pode ser encontrado no APÊNDICE C.

Quadro 4 – Taxas de código simuladas e os respectivos vetores de puncionamento.

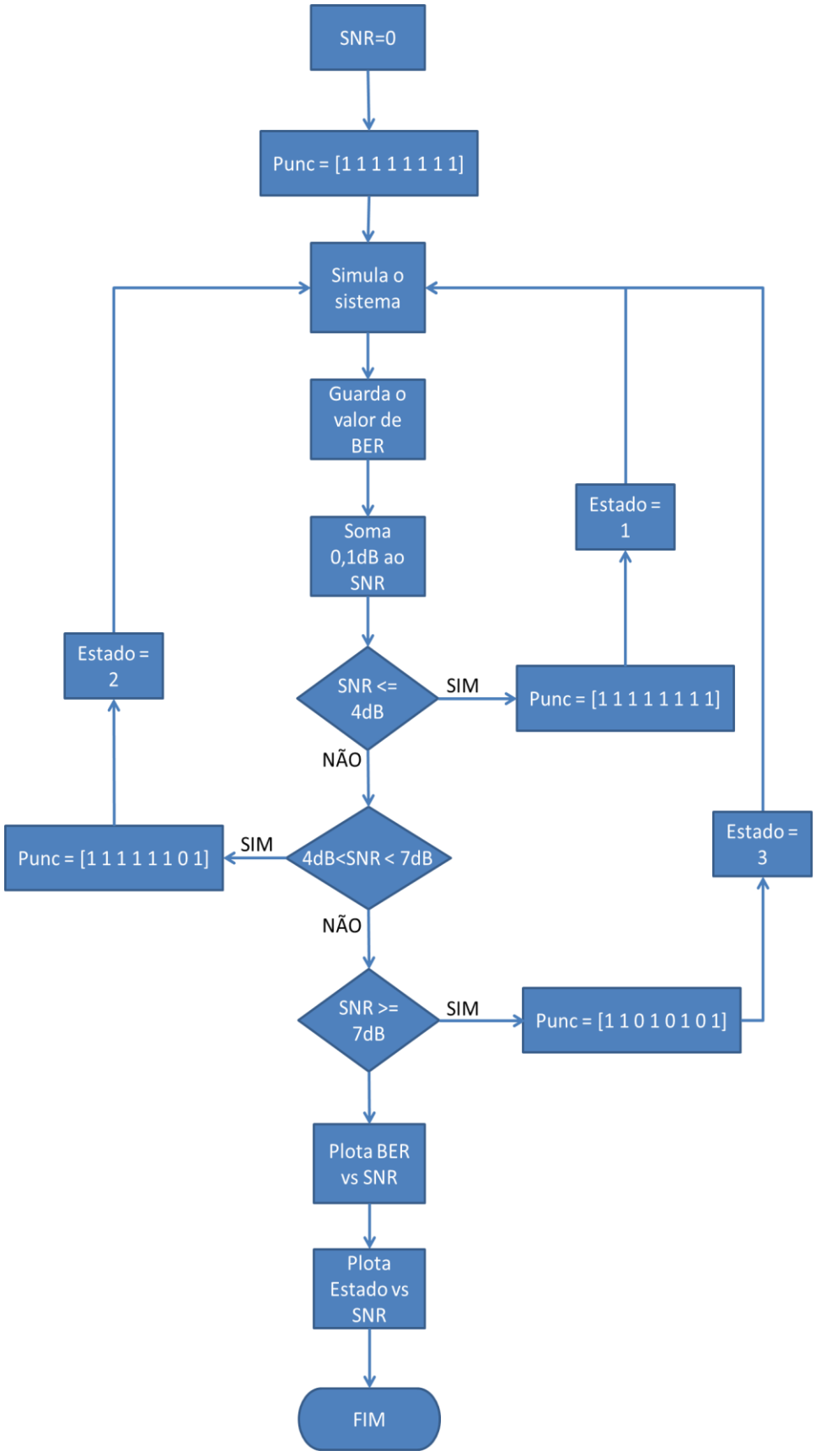
Taxa do código	Vetor de puncionamento
4/5	1 1 0 1 0 1 0 1
2/3	1 1 1 1 0 1 0 1
4/7	1 1 1 1 1 1 0 1

Fonte: Próprio autor

4.7 VARIAÇÃO AUTOMÁTICA DA TAXA DE CÓDIGO CONFORME O SNR

Foi desenvolvida uma rotina de variação automática da taxa do código conforme o SNR do canal. Na Figura 34 pode-se observar o fluxograma que representa essa rotina. O objetivo desse código é variar a taxa do código conforme o nível da relação sinal ruído do canal, onde a variação na taxa do código é atingida mudando o padrão do vetor de puncionamento, ou seja, mudando a quantidade de bits de paridade na codificação. Para SNRs menores que 4dB (alto nível de ruído) é utilizado o codificador sem puncionamento com taxa de código 1/2. Para SNRs entre 4dB e 7dB é utilizado o codificador com puncionamento com taxa de código 4/7, ou seja, supressão de 1 bit de paridade. E para relações sinal ruído maiores do que 7dB é utilizado o codificador com puncionamento e taxa de código 4/5 (supressão de 3 bits). A amplitude de SNR simulada é de 0dB até 10dB com passos de 0,1dB. Esse sistema pode ser utilizado para diferentes taxas de código, conforme a necessidade do sistema e com mais estágios de funcionamento (mais valores de taxa de código). A rotina foi parametrizada para manter o BER abaixo de 10^{-3} em toda a amplitude de SNR estudada, porém este limiar também pode ser alterado conforme a necessidade do sistema de comunicação. Neste caso foi definida uma variação linear crescente do SNR, porém, a utilização de uma sequência aleatória para o SNR (mais próxima da realidade) não alteraria a funcionalidade do código. O código desta rotina pode ser encontrado no APÊNDICE D.

Figura 34 – Fluxograma do código variação automática da taxa do código conforme o SNR



Fonte: Próprio Autor

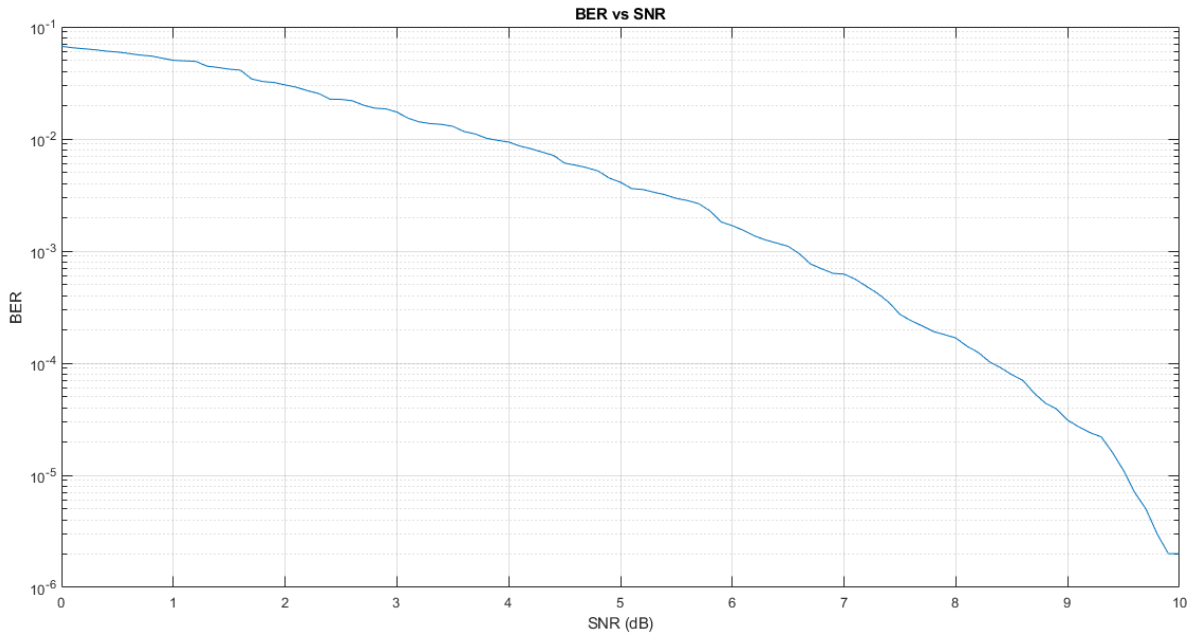
5 RESULTADOS OBTIDOS

O presente capítulo tem por objetivo mostrar os resultados obtidos nas simulações, assim como interpretar esses resultados. Os dados obtidos são explorados na forma de gráficos analisados a seguir.

5.1 SIMULAÇÃO SEM CODIFICAÇÃO

Os resultados da simulação do sistema sem codificação podem ser encontrados no gráfico da Figura 35. O gráfico apresenta no eixo *y* a taxa de erro de bit (BER) em escala logarítmica (para facilitar a visualização) e no eixo *x* a relação sinal ruído. Como se pode observar, o sistema apresenta uma taxa de erro elevada até um valor de SNR de 7dB, e conclui-se que em um sistema de comunicação digital real, esses resultados seriam inaceitáveis, visto que uma comunicação assertiva não pode ser garantida com um desempenho como esse. Uma taxa de erros (BER) aceitável para um sistema de comunicação digital com canal AWGN (por exemplo) é, na maioria dos casos, de cerca de 10^{-3}. Pode-se observar que esse sistema atende este requisito apenas para níveis de SNR próximos a 7dB.

Figura 35 – Gráfico de BER vs SNR do sistema sem codificação.

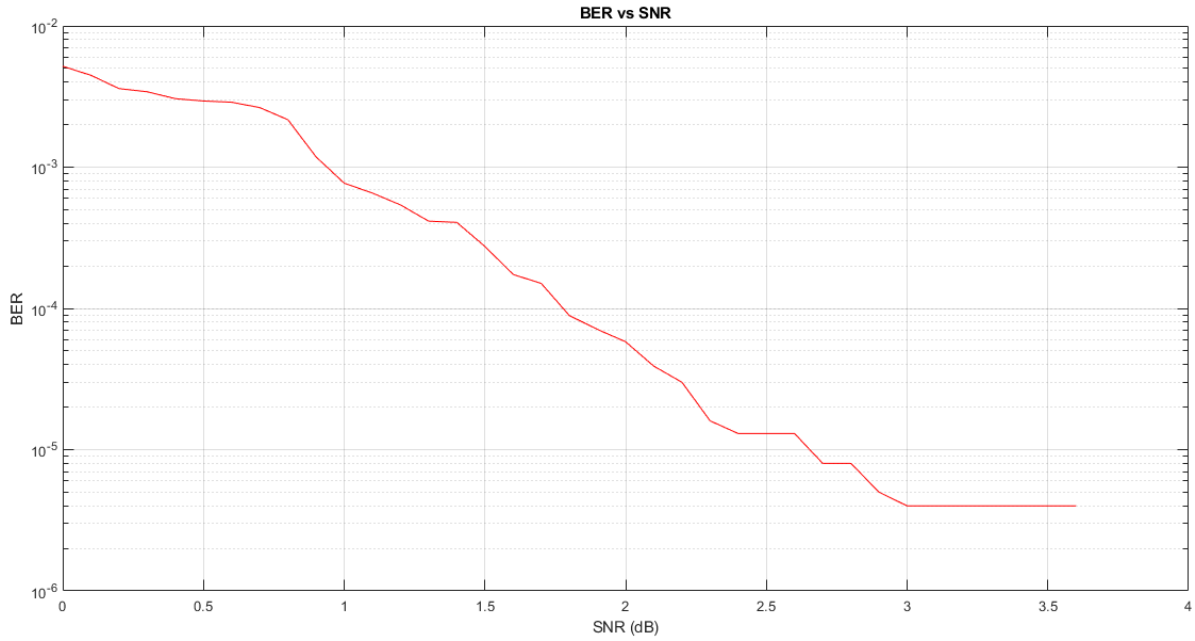


Fonte: Próprio Autor

5.2 SIMULAÇÃO COM CODIFICAÇÃO

Os resultados da simulação do sistema com codificação podem ser encontrados no gráfico da Figura 36. O gráfico da figura apresenta no eixo *y* a taxa de erro (BER) em escala logarítmica (para facilitar a visualização) e no eixo *x* a relação sinal ruído. Como se pode observar, o limite superior do eixo *x* é em 4dB, porém a simulação foi feita até 10dB, e isto foi feito pois, para valores de SNR maiores do que 4dB, a taxa e erros do sistema é nula. Desta forma fica evidente que este sistema possui uma taxa de erros menor do que o sistema anterior em toda a faixa de SNR simulada, e estes resultados permitem concluir que a codificação convolucional diminui a taxa de erros do sistema de comunicação.

Figura 36 – Gráfico BER vs NSR para o sistema com codificação



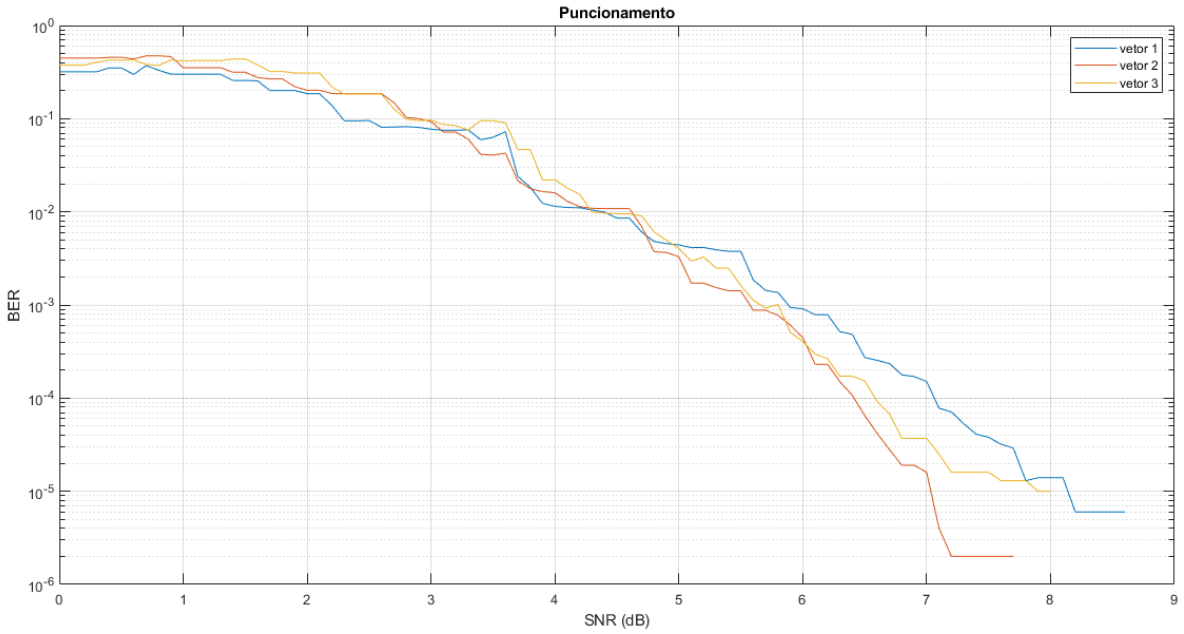
Fonte: Próprio Autor

5.3 SIMULAÇÃO COM PUNCIONAMENTO

O sistema com codificação convolucional e puncionamento foi simulado para três diferentes possíveis vetores de puncionamento, e esses vetores podem ser encontrados nas Equações 16, 17 e 18. Na Figura 37 podem-se observar os resultados da simulação para os três vetores, onde no eixo *y* consta a taxa de erro (BER) em escala logarítmica (para facilitar a visualização) e no eixo *x* a relação sinal ruído. Observando o gráfico é possível concluir que a sequência da Equação 17 (denominado na legenda do gráfico como “vetor 2”) gera uma

codificação mais robusta, visto que possui a taxa de bits errados média mais baixa na faixa de SNR analisada.

Figura 37 – Gráfico BER vs SNR do sistema com três padrões de puncionamento e taxa de código constante

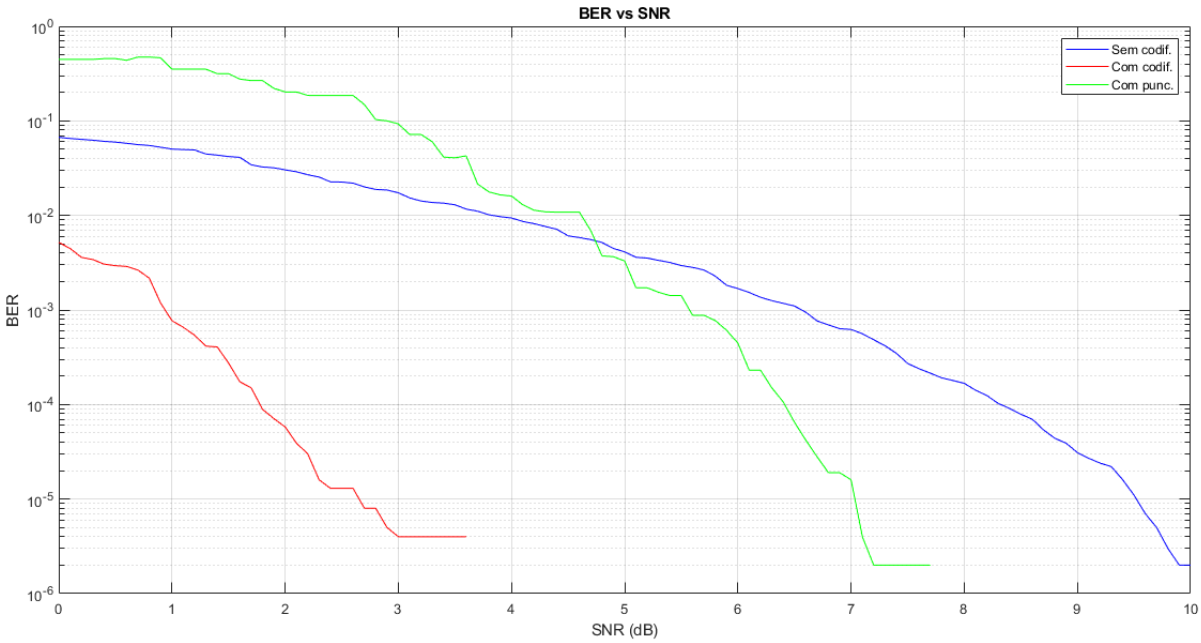


Fonte: Próprio Autor

Para SNRs acima de 4dB este sistema apresenta um resultado melhor do que o sistema sem codificação e resultados semelhantes ao do sistema com codificação pura para SNRs acima de 7dB. Isso permite concluir que para canais com pouco ruído é desnecessário utilizar uma codificação convolucional pura, visto que essa configuração reduz a taxa de dados e, por conseguinte requer um maior uso da banda do canal para manter a taxa de dados original (redução de eficiência espectral).

Na Figura 38 pode-se observar o resultado dos três sistemas sobrepostos, sendo que o gráfico relaciona o BER versus SNR: quanto mais distante a curva do eixo *x*, maior é o erro e, por conseguinte, pior é o desempenho. Isto posto, é possível concluir que para obter um melhor desempenho em termos de BER, deve-se utilizar a codificação sem puncionamento, porém, com relação a um melhor desempenho de BER versus banda do canal, o melhor é uma combinação de todos os três sistemas.

Figura 38 – Gráfico com os três sistemas sobrepostos

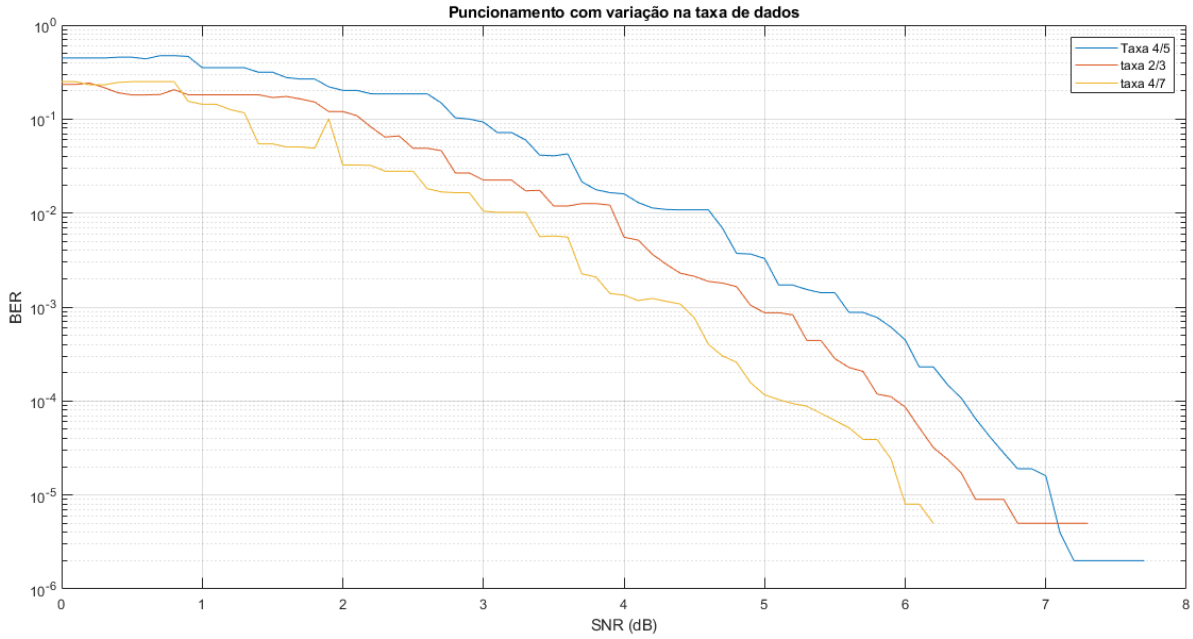


Fonte: Próprio Autor

5.4 SIMULAÇÃO COM DIFERENTES TAXAS DE CÓDIGO

No gráfico da Figura 39 pode-se observar os resultados do sistema com 3 diferentes taxas de código, onde essas três taxas são obtidas através do puncionamento. No eixo *x* do gráfico consta a taxa de erro de bit, no eixo *y* consta a relação sinal ruído (SNR) do canal. Este gráfico é o resultado obtido ao compilar o código representado pelo fluxograma da Figura 33.

Figura 39 – Gráfico BER vs SNR do sistema com diferentes taxas de código



Fonte: Prórpio Autor

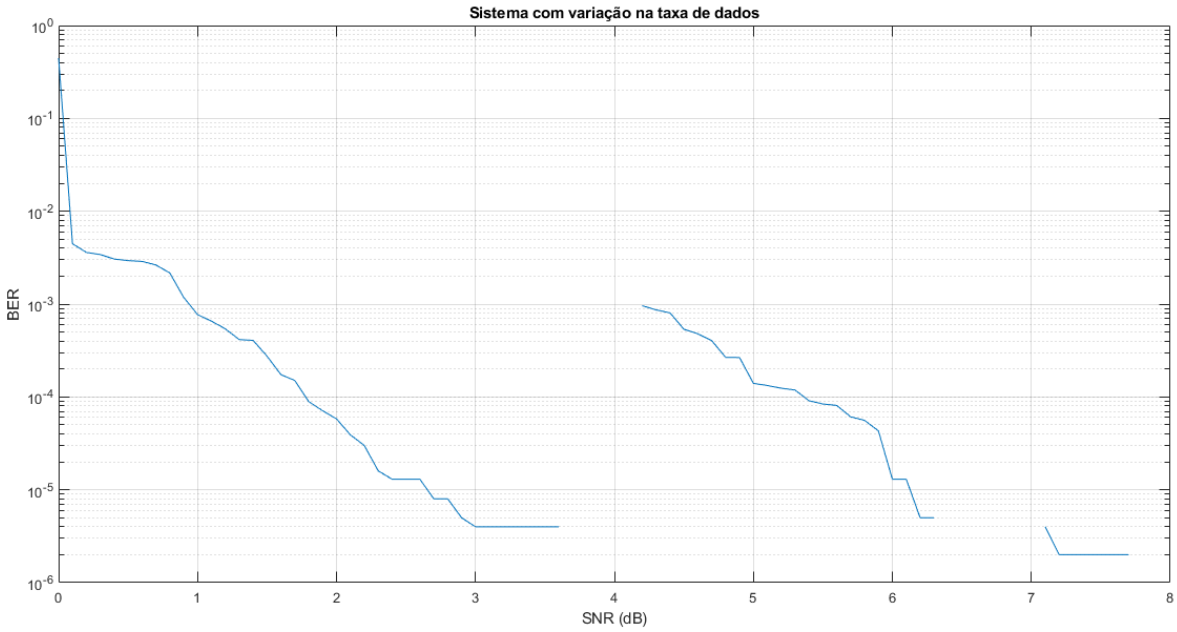
Pode-se concluir que o sistema com menor valor de BER para todo o espectro de SNR é o com taxa de código 4/7, o segundo melhor desempenho é o sistema com taxa de código 2/3 e o sistema com pior desempenho dentre os três é o com taxa de código 4/5. Estes resultados fazem sentido, visto que, quanto menor a taxa de código, melhor deve ser o desempenho do sistema, isto é consequência da quantidade de bits de paridade usada na codificação.

O código elaborado para variar a taxa de código pode ser utilizado para analisar padrões de puncionamento, e este recurso permite que padrões com melhor desempenho sejam escolhidos para a utilização prática do sistema. Atualmente utiliza-se bancos de dados para a escolha de padrões de puncionamento com bom desempenho. O recurso de poder simular automaticamente inúmeros padrões de puncionamento permite que esse banco de dados seja ampliado.

5.5 SISTEMA COM VARIAÇÃO AUTOMÁTICA DA TAXA DE CÓDIGO

Os resultados da simulação do sistema com variação automática da taxa de código podem ser encontrados no gráfico da Figura 40, onde no eixo *y* consta a taxa de bits errados em escala logarítmica, no eixo *x* consta o SNR do canal em dB.

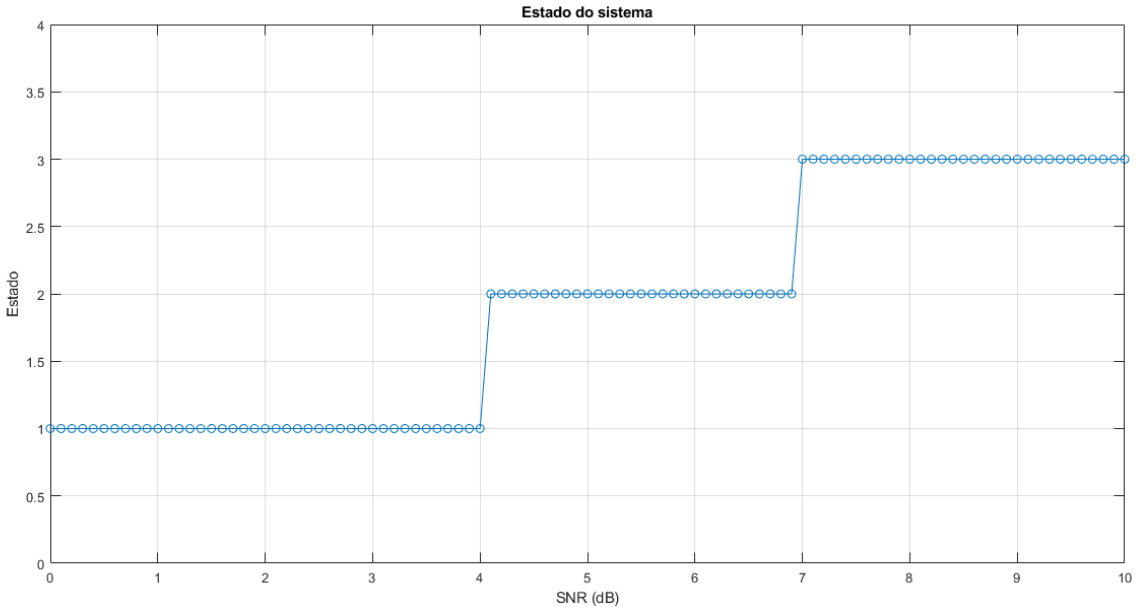
Figura 40 – Gráfico BER vs SNR do sistema com variação automática da taxa de código



Fonte: Próprio Autor

Pode-se observar para o sistema híbrido que para SNRs maiores do que 1dB, a taxa de bits errados não supera o valor de 10^{-3} e os períodos nos quais não aparece a curva, o BER é nulo. Portanto, o desempenho do sistema em função do BER é comparável ao sistema com codificação convolucional pura. Na Figura 41 pode-se observar o estado do sistema em função da relação sinal ruído do canal, e através deste gráfico pode-se identificar a taxa do código em função do SNR e qual o vetor de puncionamento está sendo utilizado. O Quadro 5 resume a relação entre estado, taxa do código e vetor de puncionamento.

Figura 41 – Estado do sistema conforme a variação do SNR



Fonte: Próprio Autor

Quadro 5 – Relação entre estado, taca do código e vetor de puncionamento

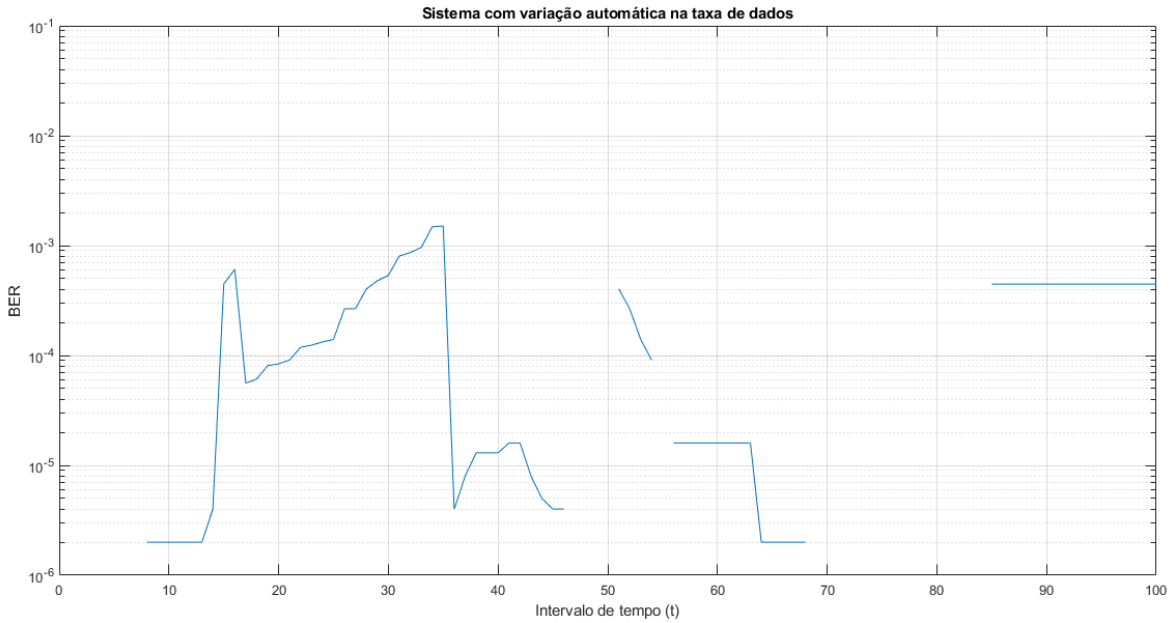
Estado	Taxa do código	Vetor de puncionamento
1	1/2	1 1 1 1 1 1 1 1
2	4/7	1 1 1 1 1 1 0 1
3	4/5	1 1 0 1 0 1 0 1

Fonte: Próprio Autor

Pode-se concluir pela Figura 41 que o estado mais oneroso em função do uso da banda do canal ocorre apenas em 40% da faixa de SNR analisada, portanto, este sistema apresenta uma vantagem no uso da banda do canal. Isto corrobora com a afirmação de que o sistema com variação automática da taxa de dados apresenta os melhores resultados entre todos os sistemas analisados. Neste caso foi utilizada uma faixa linearmente crescente e determinada do SNR, porém, caso o SNR variasse aleatoriamente, não haveria alteração no desempenho do sistema. Na Figura 42 pode-se observar o resultado do código para uma variação aleatória do SNR. No eixo *y* em escala logarítmica consta a taxa de bits errados, no eixo *x* consta o instante de tempo no qual decorre a comunicação. Pelo gráfico pode-se concluir que para variações aleatórias do SNR, o código mantém o BER abaixo de 10^{-3}. Este caso é apresentado, pois está mais próximo de um sistema de comunicação real, onde a variação do SNR é aleatória. O principal a ser observado neste gráfico é que o sistema de variação automática procura deixar o BER abaixo

de 10^{-3}. As variações bruscas que ocorrem no BER são justamente os instantes de tempo que o código está variando a taxa de código. Portanto, o sistema está otimizando a taxa de código conforme o SNR, porém sempre mantendo o BER abaixo de um valor definido, neste caso foi escolhido 10^{-3}.

Figura 42 – Resultado do código para uma variação aleatória do SNR



Fonte: Próprio Autor

6 CONCLUSÕES

Os resultados obtidos permitem concluir que a codificação convolucional reduz o BER do sistema quando comparado com o sistema sem codificação, também se pode concluir que o uso de padrões de puncionamento aumenta a taxa do código do sistema nas custas do desempenho do codificador. Portanto, o uso do puncionamento é vantajoso para canais com relação sinal ruído acima de 4dB. O uso do canal sem codificação apresenta resultados satisfatórios apenas para relações sinal ruído elevadas, que na prática não são frequentes, portanto, pode se concluir que o uso de códigos corretores de erro é fundamental para viabilizar a eficácia de sistemas de comunicação digital, e nesses casos, a codificação convolucional é um desses códigos.

Pode-se constatar através das simulações do sistema com puncionamento que existe diferença no desempenho do codificador mesmo que a taxa de código seja idêntica. Pode-se concluir que o vetor 2 obteve o melhor resultado, porém isto só pôde ser observado através da simulação. Isto ressalta que os padrões de puncionamento não apresentam uma relação matemática direta com a probabilidade de erro do codificador.

Isto posto, o código de variação do padrão de puncionamento pode ser utilizado em trabalhos futuros. Esta rotina é uma maneira de analisar inúmeros padrões de puncionamento e, assim, identificar padrões com bom desempenho. Essa análise automática permite aumentar o banco de dados de padrões de puncionamento. Uma sugestão de trabalho futuro consiste em portar o presente trabalho para a plataforma GNURadio e, posteriormente, para alguma plataforma de rádio definido por software, como o NI USRP.

O sistema com variação automática da taxa de código apresentou o melhor desempenho geral entre todos os sistemas simulados, isto porque apresentou uma taxa de bits errados abaixo de 10^{-3} para toda a faixa de SNR analisada e, também, reduziu o uso da banda do canal. Essa aplicação permite que os códigos corretores de erro sejam otimizados conforme a necessidade. Para um sistema de comunicação digital real, o uso eficiente de recursos é fundamental, este trabalho vai de encontro com este conceito, portanto a sua aplicação caracteriza um ganho prático.

Com este trabalho foi desenvolvido um framework para trabalhos futuros. O código de automatização de variação da taxa do código pode ser utilizado para sistemas QAMs com constelações maiores, por exemplo, 256 QAM e com outros modelos de canais, principalmente os multi-caminhos, como *Rician* e *Rayleigh*. O uso de um codificador com comprimento de restrição maior também poderia ser utilizado, a rotina desenvolvida permite o uso de

codificadores convolucionais com alta taxa de código e, este fato, poderia ser abordado em trabalhos futuros.

REFERÊNCIAS

[1] PROAKIS, John G.; SALEHI, Masoud. **Contemporary Communication Systems using MATLAB**. Boston. PWS Publishing Company. 1996. ISBN 0-534-93804-3.

[2] STÜBER, Gordon L. **Principles of Mobile Communication**. Atlanta. Springer. 2017. ISBN 978-3-319-55614-7.

[3] SKLAR, Bernard. **Digital Communications: Fundamentals and Applications**. Nova Jersey. Prentice Hall. 2001.

[4] SIMON, Michael, Haykin; **Sistemas Modernos de Comunicações Wireless**. Porto Alegre. Bookman. 2008. ISBN 978-85-7780-155-8.

[5] MOISION, Bruce. A Truncation Depth Rule of Thumb for Convolutional Codes. In Information Theory and Applications Workshop (January 27 2008-February 1 2008, San Diego, California), 555-557. New York: IEEE, 2008.

[6] ABRANTES, Silvio. A. **Código Correctores de Erros em Comunicações Digitais**. Porto. FEUP Edições. 2010. ISBN 978-972-752-127-2.

[7] JIANG, Yuan. **A Practical Guide to Error-Control Coding Using MATLAB**. Norwood. Artech House. 2010. ISBN 13: 978-1-60807-088-6.

[8] MATSUO, Michihito. *et al*. Rate-Adaptive Indoor Infrared Wireless Communication Systems Using Punctured Convolutional Codes and Adaptive PPM. **Electronics and Communications in Japan**, Japão, Part 1, vol. 84, No. 8, p31-39, ago, 2001. Disponível em: http://eds.a.ebscohost.com/eds/pdfviewer/pdfviewer?vid=7&sid=bc1048cd-d860-4278-9416-5c21b7285278%40sessionmgr4006. Acesso em: 23 out. 2020.

[9] NARUSHAN, Pillay. Et al. Repeat-Punctured Superorthogonal Convolutional Turbo Codes on AWGN and Flat Rayleigh Fading Channels. **South African Journal of Science**. África do Sul, vol. 106, No. 11/12, p48-58, nov. 2010. Disponível em:

http://eds.a.ebscohost.com/eds/pdfviewer/pdfviewer?vid=17&sid=bc1048cd-d860-4278-9416-5c21b7285278%40sessionmgr4006. Acesso em: 23 out. 2020.

[10] SARI, Lydia. RCPC codes performance analysis using non-punctured equivalent convolutional codes. **2014 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)**, Jakarta, 2014, pp. 21-24, nov. 2014. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7050519. Acesso em: 23 out. 2020.

APÊNDICE A – CÓDIGO DE VARIAÇÃO AUTOMÁTICA DO SNR

```
Snr = zeros(1,100);
Ber_Sem = zeros(1,100);
Ber_Com = zeros(1,100);
Ber_Pun = zeros(1,100);
i = 0;
for m=0:0.1:10
    snr1 = m;
    i = i+1;
    load_system("fredana.slx");
    simOut = sim("fredana.slx");
    Ber_Sem(1,i) = BER_Data_Sem(1,1);
    Snr(1,i) = snr1;
end


i=0;
for m=0:0.1:10
    snr2 = m;
    i = i+1;
    load_system("fredana.slx");
    simOut = sim("fredana.slx");
    Ber_Com(1,i) = BER_Data_Com(1,1);
    Snr(1,i) = snr2;
end




i=0;
for m=0:0.1:10
    snr3 = m;
    i = i+1;
    load_system("fredana.slx");
    simOut = sim("fredana.slx");
    Ber_Pun(1,i) = BER_Data(1,1);
    Snr(1,i) = snr3;
end


semilogy(Snr, Ber_Sem,'b',Snr,Ber_Com,'r',Snr,Ber_Pun,'g');
title('BER vs SNR');
xlabel('SNR (dB)');
ylabel('BER');
legend('Sem codif.','Com codif.','Com punc.')
grid on;
```

APÊNDICE B – CÓDIGO DE VARIAÇÃO DO PADRÃO DE PUNCIONAMENTO

```
Snr = zeros(1,100);
Ber_Pun = zeros(1,100);
Ber_Sem = zeros(1,100);
Ber_Com = zeros(1,100);
i = 0;
for m=0:0.1:10
    punc = [0;1;0;1;0;1;1;1];
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_Sem(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
end


i=0;
for m=0:0.1:10
    punc = [1;1;0;1;0;1;0;1];
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_Com(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
 end




i=0;
for m=0:0.1:10
    punc = [1;0;1;0;1;0;1;1];
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_Pun(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
end

semilogy(Snr,Ber_Sem, Snr,Ber_Com, Snr,Ber_Pun);
title('Puncionamento');
xlabel('SNR (dB)');
ylabel('BER');
legend('vetor 1','vetor 2','vetor 3');
grid on;
```

APÊNDICE C – CÓDIGO DE VARIAÇÃO DA TAXA DO CÓDIGO

```
Snr = zeros(1,100);
Ber_Pun = zeros(1,100);
Ber_Sem = zeros(1,100);
Ber_Com = zeros(1,100);
i = 0;
for m=0:0.1:10
    punc = [1;1;0;1;0;1;0;1];
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_Sem(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
end


i=0;
for m=0:0.1:10
    punc = [1;1;1;1;0;1;0;1];
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_Com(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
 end




i=0;
for m=0:0.1:10
    punc = [1;1;1;1;1;1;0;1];
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_Pun(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
end


semilogy(Snr,Ber_Sem, Snr,Ber_Com, Snr,Ber_Pun);
title('Puncionamento com variação na taxa de dados');
xlabel('SNR (dB)');
ylabel('BER');
legend('Taxa 4/5','taxa 2/3','taxa 4/7');
grid on;
```

APÊNDICE D – CÓDIGO DE VARIAÇÃO AUTOMÁTICA DA TAXA DO CÓDIGO CONFORME O SNR DO CANAL

```
Snr = zeros(1,100);
t = zeros(1,100);
estado = zeros(1,100);
Ber_final = zeros(1,100);
punc = [1;1;0;1;0;1;0;1];
periodo = 4;
i = 0;
for m=0:0.1:10
    variando = m;
    i = i+1;
    load_system("punvariando.slx");
    simOut = sim("punvariando.slx");
    Ber_final(1,i) = Ber_Data(1,1);
    Snr(1,i) = variando;
    t(1,i) = m;

    if (variando <= 4)
        punc = [1;1;1;1;1;1;1;1];
        periodo = 2;
        retorno = 36;
        estado(1,i) = 1;
    end

    if ((variando > 4) & (variando < 7))
        punc = [1;1;1;1;1;1;0;1];
        periodo = 4;
        retorno = 43;
        estado(1,i) = 2;
    end

    if (variando >= 7)
        punc = [1;1;0;1;0;1;0;1];
        periodo = 4;
        retorno = 96;
        estado(1,i) = 3;
    end

end


semilogy(Snr,Ber_final);
title('Sistema com variação automática na taxa de dados');
xlabel('SNR (dB)');
ylabel('BER');
grid on;


plot(Snr,estado,'-o');
axis([0 10 0 4]);
```

```
title('Estado do sistema');
xlabel('SNR (dB)');
ylabel('Estado');
grid on;
```