AE3 - Códigos de Blocos

Jessica de Souza, Luísa Machado
Engenharia de Telecomunicações, Instituto Federal de Santa Catarina
<jessica.souzajds@gmail.com, luisamachado@gmail.com>

05 de novembro de 2018

1. Introdução

O objetivo desta atividade é reforçar os conhecimentos adquiridos sobre Códigos Corretores de Erro (CCE). Os códigos corretores de erro são usados para recuperar possíveis erros na informação recebida e adicionar redundância à informação. Neste trabalho os códigos estudados foram os códigos de blocos de Hamming e de Golay.

A execução desta atividade consistiu em realizar quatro experimentos: o primeiro consistiu na realização de um pré-laboratório com o objetivo de fortalecer os conceitos abordados em aula e auxiliar nas atividades propostas (este pré-laboratório não será discutido neste relatório); o segundo consistiu em implementar um decodificador HDD (*Hard-Decision Decoding*), o qual utiliza a tabela da síndrome/LUT (*Look up Table*) do código para decodificar uma informação. Já no terceiro experimento, foi desenvolvido um decodificador SDD (*Soft-Decision Decoding*), a decodificação neste caso é realizada encontrando a mínima distância Euclidiana ou a máxima correlação. Por fim, o último experimento utiliza os dois decodificadores feitos anteriormente para simular o desempenho de BER x Eb/N0 do código de Hamming(7, 4) e do código de Golay(23, 12) e compará-los com valores teóricos.

2. Metodologia

Foi utilizado o *software MATLAB* na realização das simulações descritas neste relatório. Todos os experimentos foram executados na ordem descrita na introdução, com exceção do pré-laboratório o qual não houve necessidade. Nas seções a seguir, são detalhados os procedimentos realizados em cada um deles.

Os códigos de Hamming e Golay são denominados CCE's lineares e perfeitos, pois corrigem no máximo os padrões de erro respeitando a equação 2^(n-k). Ou seja, o código de Hamming corrige um padrão de erro de peso 0 e 7 padrões de erro de

peso 1. Já o código de Golay corrige 1 padrão de erro de peso 0, 23 padrões de erro de peso 1, 253 padrões de erro de peso 2 e 1771 padrões de erro de peso 3. Ambos os códigos não corrigem nada mais além destes pesos. Além disso, os dois códigos são considerados sistemáticos, pois os bits de informação aparecem intactos nas palavras-código.

2.1. Experimento 1

No primeiro experimento, foi implementada uma função que decodifica um código de bloco linear genérico utilizando a tabela de síndrome/LUT, este tipo de decodificação é chamada *Hard-Decision Decoding* (HDD). A Figura 1 mostra um diagrama de decodificação, onde as partes em vermelho claro são as que utilizamos.

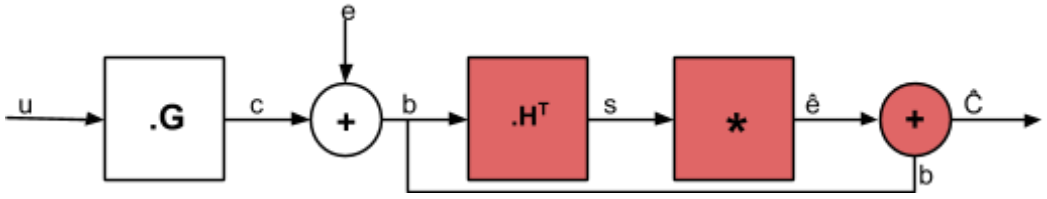

Figura 1 - Diagrama do decodificador via síndrome.

Segundo a Figura 1, no decodificador HDD a palavra recebida, representada por **b**, é multiplicada pela Matriz de Verificação de Paridade transposta, $\mathbf{H}^T$. O resultado desta multiplicação será correspondente a um valor da tabela Síndrome, **s**. A partir disto, é necessário encontrar o **s** na tabela Síndrome o valor correspondente do padrão-de-erro, **ê**. Com isto, basta somar **b** ao **ê** encontrando a palavra-código estimada, **ĉ**. Como dito anteriormente, foi utilizado o código de Hamming(7,4) para esta atividade.

A função dessa atividade é o "decodeHDD", ele recebe como entradas uma sequência de bits recebidos do canal e uma estrutura representando o código. Na saída temos uma sequência estimada dos bits de informação. A Figura 2 mostra o código utilizado para a implementação do decodificador HDD.

```
function u_hat = decodeHDD(b, args)
    % Argumentos:
    % b = sequência binaria de tamanho 7
    % args = struct obtida via função fun_ham(n,k)

    % Transpõe a matriz H
    H_t = args.h.';

    for ii = 1:size(b, 1)
        % Multiplica pela sequência binária
        s = b(ii, 1:end) * H_t;
        S = mod(s, 2);

        % Encontra a posição correspondente na LUT
        LUT = args.lut;
        pos = find(ismember(LUT.s, S, 'rows'));
        e_hat = LUT.e_hat(pos, :);

        % Faz o XOR do valor encontrado com a sequência
        c_hat(ii, 1:args.n) = xor(e_hat, b(ii, 1:end));
    end

    % Obtém apenas os primeiros dígitos correspondentes à informação
    u_hat = c_hat(1:end, 1:args.k);
end
```

Figura 2 - Implementação do decodificador HDD.

2.2. Experimento 2

No segundo experimento também foi elaborada uma função que decodifica um código de bloco linear genérico, porém esse realiza a decodificação calculando a mínima distância Euclidiana da mensagem recebida com todas as palavras-código possíveis, utilizando o método chamado de *Soft-Decision Decoding* (SDD). A decodificação SDD melhora o processo de decisão, pois a comparação é feita com a informação recebida e as palavras-código modulada reduzindo a probabilidade de erro. O algoritmo de Viterbi implementa um decodificador SDD.

Para realização desta função assume-se sinalização polar. As entradas recebidas nessa função são: uma sequência de símbolos recebida do canal e uma estrutura que representa o código. Foi utilizado como prova o código de Hamming(7,4) para implementação deste experimento. Na saída desta função tem-se uma sequência estimada dos bits de informação. O código é transformado para polar, obtendo valores de -1 e 1, assim como a informação recebida pelo decodificador. Logo após isto, a sequência de símbolos é repetida com o mesmo

tamanho do código, para que se possa calcular a distância euclidiana e assim, achar a mínima distância que será a palavra-código estimada. Por fim, apenas os primeiros 4 dígitos são retornados como a mensagem estimada. A Figura 3 contém o código utilizado para a implementação do decodificador SDD.

```
function u_hat = decodeSDD(r, code)
    % Argumentos:
    % r = Sequência de símbolos recebidas
    % code = Representação do código

    % Transformando o código para polar
    c = 2 * code.c - 1;

    for ii = 1:size(r, 1)
        % Repetindo a sequência ao longo de todo o código
        rr = repmat(r(ii, 1:end), size(c, 1), 1);

        % Calcula a distância Euclidiana
        distE = sum((c - rr).^2, 2);
        [valor_min indice] = min(distE);

        % Usa o de menor índica
        c_hat(ii, 1:code.n) = code.c(indice, :);
    end

    % Obtém apenas os primeiros dígitos correspondentes à informação
    u_hat = c_hat(1:end, 1:code.k);

end
```

Figura 3 - Implementação do decodificador SDD.

2.3. Experimento 3

O terceiro experimento consistiu em realizar a simulação teórica e prática de decodificação via HDD e SDD, usando os códigos de Golay(23,12) e de Hamming(7,4), e apresentar as probabilidades de erro de bit versus a variação de Eb/N0 de cada caso. Foram utilizados os decodificadores desenvolvidos nas questões anteriores (Figuras 2 e 3). A simulação de Hamming e Golay foram realizadas de forma separadas. Assumiu-se o uso de um canal AWGN e sinalização polar. Foram transmitidas 100.000 palavras-códigos, criadas de forma aleatória para este experimento. Além disso, o valor de Eb/N0 variou de −1 a 7 dB, com passo de 1 dB.

O cálculo da probabilidade de erro de palavra-código, Pc, teórica foi realizada por meio das equações a seguir. A partir disso, para HDD foram obtidos os limitantes superior e inferior da probabilidade de erro de bit conforme mostrado

mais adiante. Para SDD foi obtido apenas o limitante superior para os valores teóricos.

Para HDD:

$$Pc = 1 - \sum_{w=0}^{n} \alpha_w \cdot p^w \cdot (1-p)^{n-w}$$

onde:

$$p = Q\left(\sqrt{\frac{2 \cdot R \cdot Eb}{N_0}}\right)$$

Abaixo temos relação de Pb com a fórmula de Pc apresentada anteriormente.

$$\frac{Pc}{k} \leq Pb \leq Pc$$

Para SDD:

$$Pc \leq \sum_{w=1}^{n} A_w \cdot Q\left(\sqrt{\frac{2 \cdot w \cdot R \cdot Eb}{N_0}}\right)$$

A partir da análise da fórmula do cálculo da probabilidade de erro da palavra-código do decodificador SDD, pode-se perceber que a relação de Pb com o Pc, diferentemente do HDD, tem apenas o limitante superior, como apresentada a seguir.

$$Pb \leq Pc$$

3. Resultados e Discussão

3.1. Experimento 1

O primeiro experimento apenas implementou o decodificador SDD, onde a palavra recebida foi uma sequência pré-escolhida pela equipe. A sequência escolhida para o recebimento foi **b** = [1 0 1 1 0 0 0]. Para a codificação de b foi utilizado Hamming(7,4), onde utilizamos sua LUT para encontrar a palavra-código correspondente à mesma.

A sequência **b** foi multiplicada pela matriz **H** transposta, de forma que se obtenha uma síndrome. É verificado na LUT o valor correspondente à síndrome encontrada, assim obtendo o padrão de erro estimado (denominado por **ê**). Então, é realizada a operação *xor* entre o **ê** e a palavra **b** recebida, para que assim se obtenha a palavra-código estimada (denominado por **ĉ**). Por fim, são utilizados apenas os

quatro primeiros dígitos de **ĉ** para se obter a mensagem estimada (denominado por **û**). O resultado deste experimento é **û** = [1 0 1 1].

3.2. Experimento 2

No segundo experimento foi implementado o decodificador SDD. Também foi utilizada a palavra recebida escolhida no experimento 1, **b** = [1 0 1 1 0 0 0]. Em seguida, **b** foi codificado, utilizando Hamming(7,4). Após isso, foi necessário modular o sinal, neste experimento a modulação foi realizada multiplicando o sinal por 2 e subtraindo por 1, (b * 2) - 1. Por último, a informação passa pelo decodificador.

No decodificador, as palavras-código, **c**, são moduladas e a informação recebida é repetida de modo a ter uma matriz do tamanho da matriz de palavras-código. Com isso, calcula-se a distância Euclidiana ponto a ponto entre as matrizes, verificando, em seguida, qual é a linha com o menor valor calculado. A partir desse índice é encontrada a linha correspondente na matriz de palavras-código, obtendo assim a palavra-código estimada, **ĉ**. Os primeiros **k** bits (no caso desse experimento **k** = 4) de **ĉ** representam a mensagem estimada, **û**. Por fim, o resultado obtido neste experimento é **û** = [1 0 1 1].

3.3. Experimento 3

O último experimento foi dividido em duas partes. Nesse experimento foi simulado a transmissão de uma informação com 100.000 mensagens **u**, sendo na primeira parte as mensagens codificadas utilizando Hamming(7,4), enquanto na segunda parte codificadas com Golay(23,12).

Todos os procedimentos das duas partes desse experimento são iguais, a única diferença são os codificadores utilizados. O experimento começa gerando, codificando e modulando a informação. Após isso, a informação modulada é transmitida por um canal AWGN. A informação recebida passa pelos dois decodificadores, sendo passado pelo HDD a informação recebida após passar pelo demodulador, enquanto pelo SDD não é necessário demodular a informação recebida.

Por fim, é realizada a comparação da informação original com a informação recebida e decodificada, de modo a obter a probabilidade de erro de bit, Pb,

enquanto varia-se o Eb/N0, a fim de comparar aos valores práticos com os calculados a partir das fórmulas teóricas.

Na Figura 4, estão apresentados os limitantes superior e inferior teóricos e a probabilidade de erro de bit prática do decodificador HDD para o código de Hamming(7,4). Analisando o gráfico, pode-se notar que o valor de Pb prático ficou dentro dos limitantes teóricos, confirmando o bom funcionamento do decodificador programado.

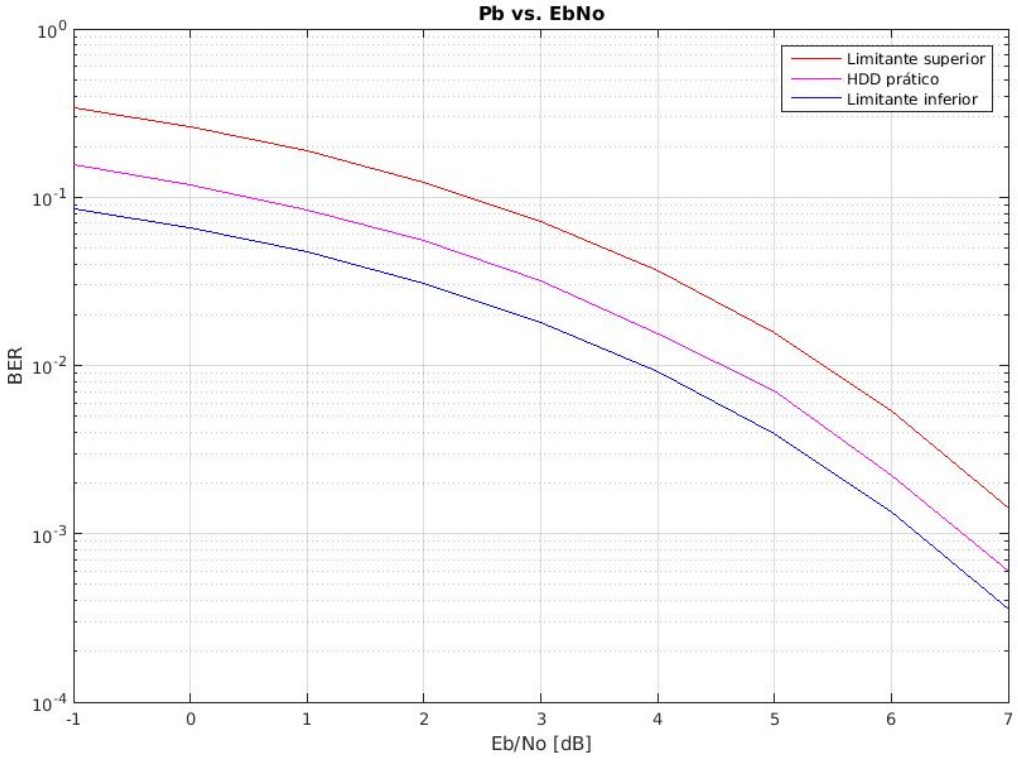

Figura 4 - Pb vs Eb/N0 para HDD usando o código de Hamming(7,4).

A Figura 5 mostra o limitante superior teórico e a probabilidade de erro de bit prática do decodificador SDD para o código de Hamming(7,4). A partir do gráfico, conclui-se que o valor de Pb prático está abaixo do limitante superior teórico, o qual confirma que o decodificador implementado funciona conforme o esperado.

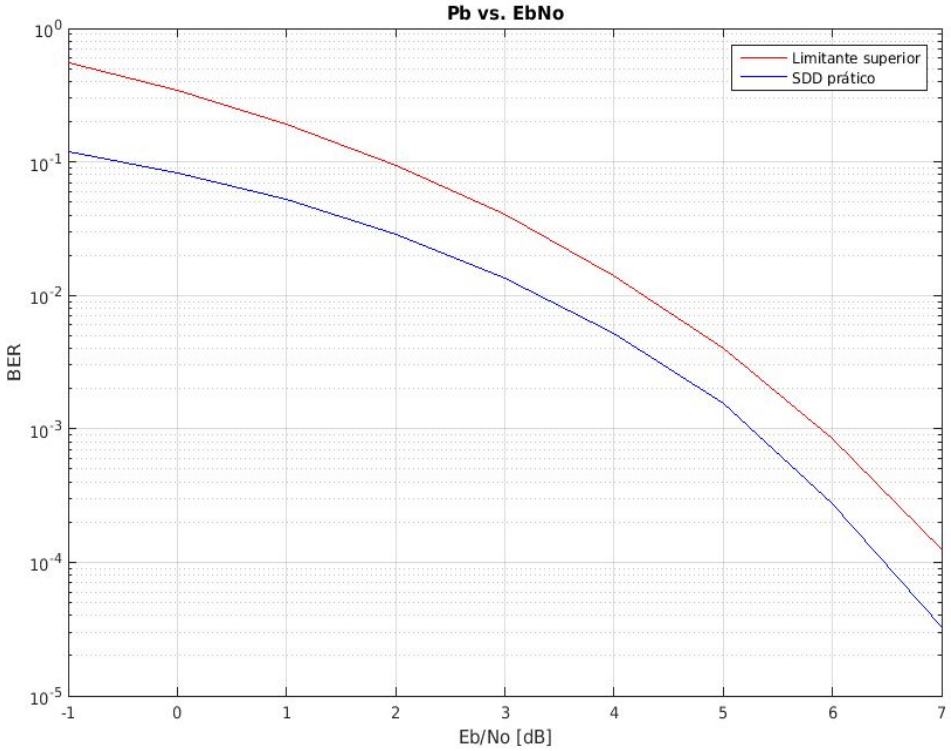

Figura 5 - Pb vs Eb/N0 para SDD usando o código de Hamming(7,4).

A Figura 6 faz um comparativo entre as probabilidades de erro de bit dos decodificadores HDD e SDD para o código de Hamming(7,4) e do não-codificado. Para a Pb do não-codificado foi calculado o valor teórico utilizando a fórmula a seguir:

$$Pb = Q\left(\sqrt{\frac{2 \cdot Eb}{N_0}}\right)$$

Conclui-se a partir da análise do gráfico da Figura 6, que o decodificador SDD tem o melhor desempenho de erro de bit conforme aumenta o Eb/N0. Nota-se também que a Pb do decodificador HDD e do não-codificado são próximas, sendo o não-codificado um pouco melhor, e conforme aumenta o Eb/N0, o HDD se supera o não-codificado, se tornando um pouco melhor.

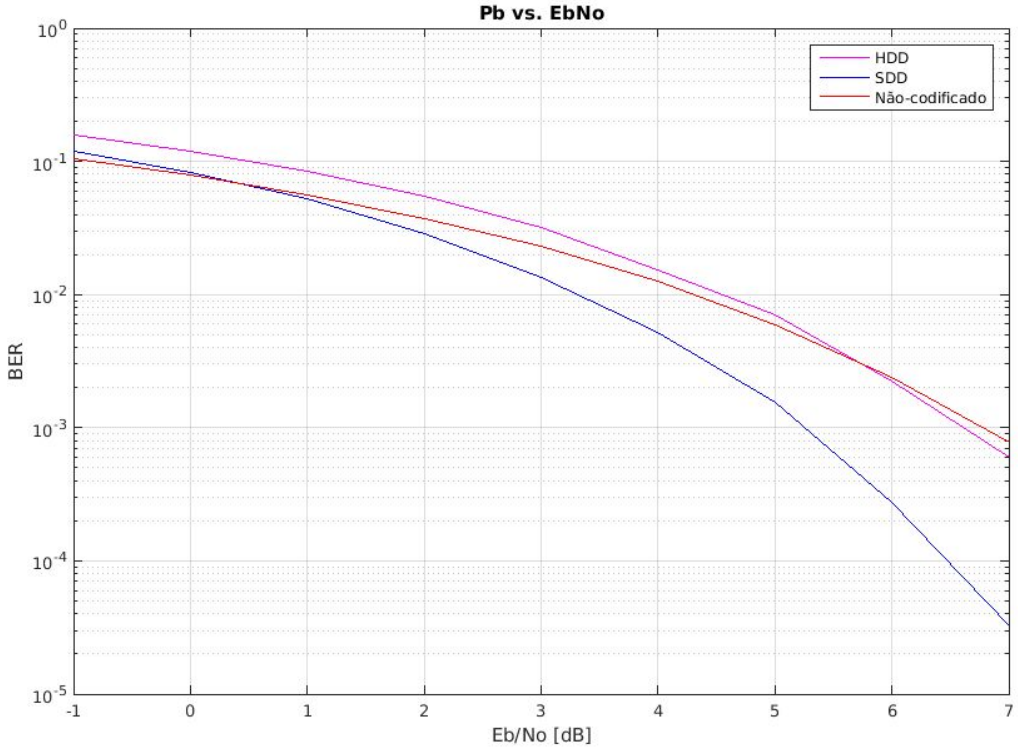

Figura 6 - Comparação do Pb vs Eb/N0 do HDD e SDD simulado e não-codificado teórico.

A Figura 7 apresenta os limitantes superior e inferior teóricos e a probabilidade de erro de bit prática do decodificador HDD para o código de Golay(23,12). Com base no resultado do gráfico, constata-se que a Pb prática respeitou os limitantes teóricos, provando que funcionamento do decodificador programado ocorreu como o esperado.

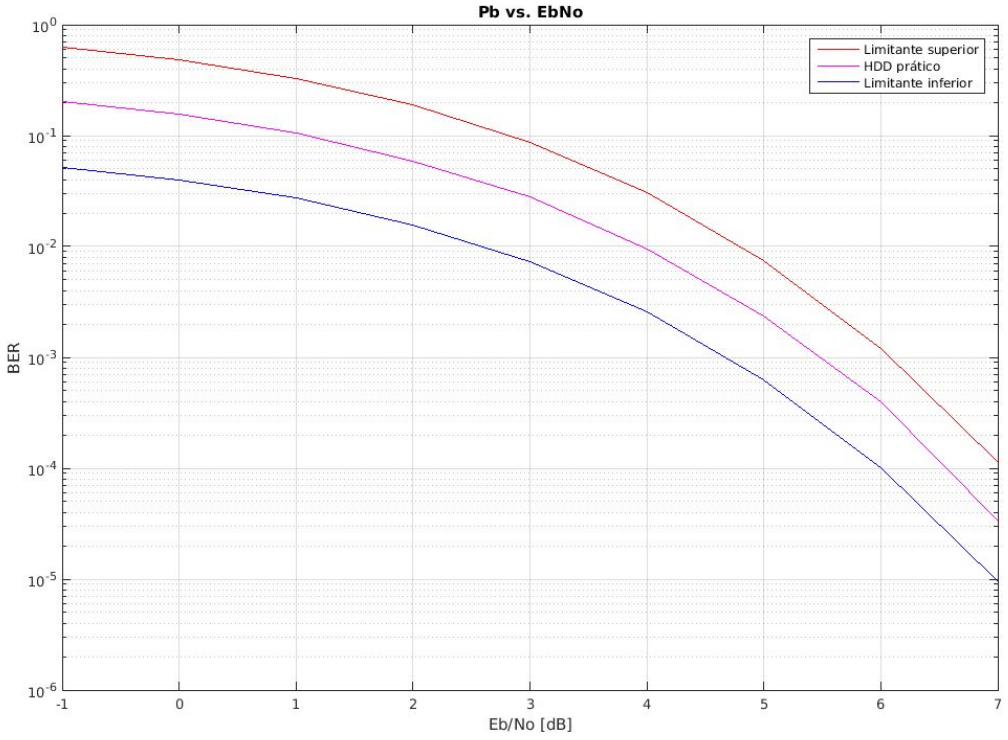

Figura 7 - Pb vs Eb/N0 para HDD usando o código de Golay(23,12).

Na Figura 8, pode-se observar o limitante superior teórico e a probabilidade de erro de bit prática do decodificador SDD para o código de Golay(23,12). Segundo o gráfico, observa-se o valor da Pb prática abaixo do limitante superior, confirmando que o decodificador implementado funciona conforme o esperado, porém a partir do valor de Eb/N0 = 5, nota-se que não há mais valores de Pb, o que sugere que não houveram erros de bit a partir deste ponto.

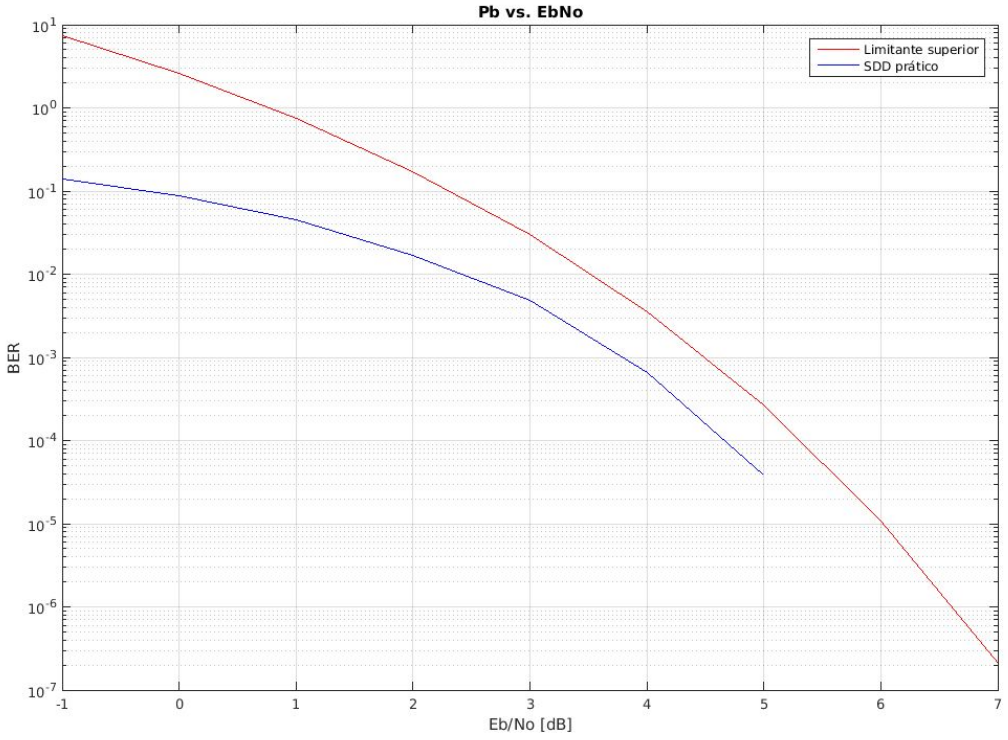

Figura 8 - Pb vs Eb/N0 para SDD usando o código de Golay(23,12).

A Figura 9 compara as probabilidades de erro de bit dos decodificadores HDD e SDD para o código de Golay(23,12) e do não-codificado. Para a Pb do não-codificado foi calculado o valor teórico utilizando a mesma fórmula apresentada na descrição da Figura 6.

Com base na análise do gráfico da Figura 9, nota-se que o decodificador SDD, assim como no código de Hamming, tem melhor desempenho de erro de bit conforme aumenta o Eb/N0. Conclui-se também que todas as Pb analisadas próximas no início do gráfico e conforme aumenta o Eb/N0 elas se distanciam, assim observamos que além do decodificador SDD ser o melhor, a Pb do decodificador HDD é melhor que o resultado teórico do não-codificado, provando que mesmo não sendo o melhor decodificador, o fato de se utilizar o código de Golay torna a comunicação mais robusta a erros de bit.

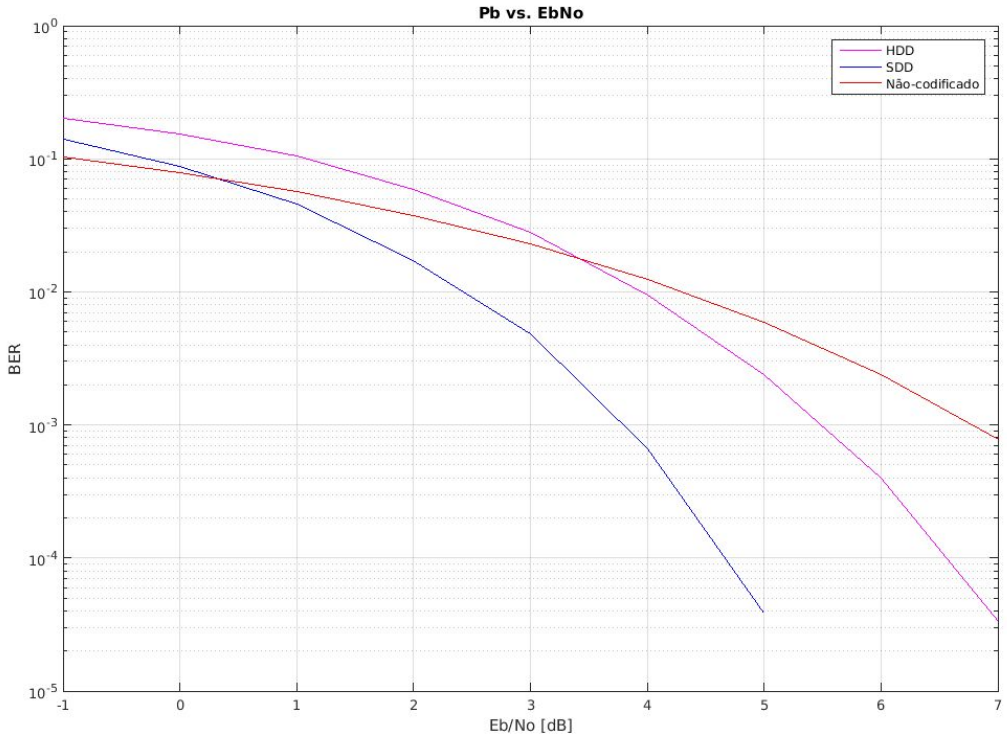

Figura 9 - Comparação do Pb vs Eb/N0 do HDD e SDD simulado e não-codificado teórico.

Analisando as últimas imagens de cada codificação, Figuras 6 e 9, obtemos a tabela abaixo com os valores de Eb/N0 quando Pb = 10^{-3}. O valor encontrado para o não-codificado é independente dos códigos, por isso é igual em ambas as simulações. Com base nesses resultados observa-se a diferença de desempenho entre os códigos e os decodificadores. Em ambos os códigos, o decodificador SDD se mostrou mais eficaz, porém no código de Golay a diferença entre os resultados é muito maior. Comparando todos os valores da tabela, percebe-se que o código de Golay mesmo se usado com o decodificador HDD é o mais eficiente, pois necessita menor potência de transmissão para enviar a informação com a mesma probabilidade de erro de bit.

	HDD	**SDD**	**Não-codificado**
Simulação de Hamming	6,5 dB	5,25 dB	6,75 dB
Simulação de Golay	5,5 dB	3,75 dB	6,75 dB

4. Conclusão

Esta atividade buscou aprofundar os conhecimentos referentes aos Códigos Corretores de Erro, de modo a explorar especificamente a decodificação e desempenho dos Códigos de Blocos. Todos os experimentos obtiveram os resultados esperados, comprovando assim a eficácia do decodificador SDD como visto na teoria.

5. Referências

[1] GOLDSMITH, A. Wireless Communications. Stanford University. 2004.
[2] HAYKIN, S. Sistemas de Comunicação - Analógicos e Digitais. Bookman. 4 ed.