

# Tarefa 4

Leonardo de Andrade Santos

Audric Costacurta dos Santos

A) Utilizando o software Matlab, traçar o gráfico da curva teórica de desempenho da modulação BPSK (2-PSK). O eixo x deve ser a relação sinal-ruído ( $E_b/N_0$ ) em escala de dB, na faixa de -2 dB a 15 dB. O eixo y deve ser a probabilidade de erro de símbolo (bit) na faixa de  $10^{-8}$

Para isso foi desenvolvido o seguinte trecho de código ilustrado pela Figura 1 a seguir:

```
1 function Curva_teorica_BPSK()
2 SNR_dB = -2:0.5:15;
3 SNR = 10.^(SNR_dB/10);
4 Pb = (1/2)*erfc(sqrt(SNR));
5
6 semilogy(SNR_dB, Pb);
7 xlabel('Relação sinal-ruído (SNR) em dB');
8 ylabel('Probabilidade de erro de símbolo (Pb)');
9 title('Curva teórica de desempenho da modulação BPSK');
10
```

Figura 1: Curva teórica da simulação BPSK

O resultado da simulação esta ilustrado pela Figura 2 a seguir:

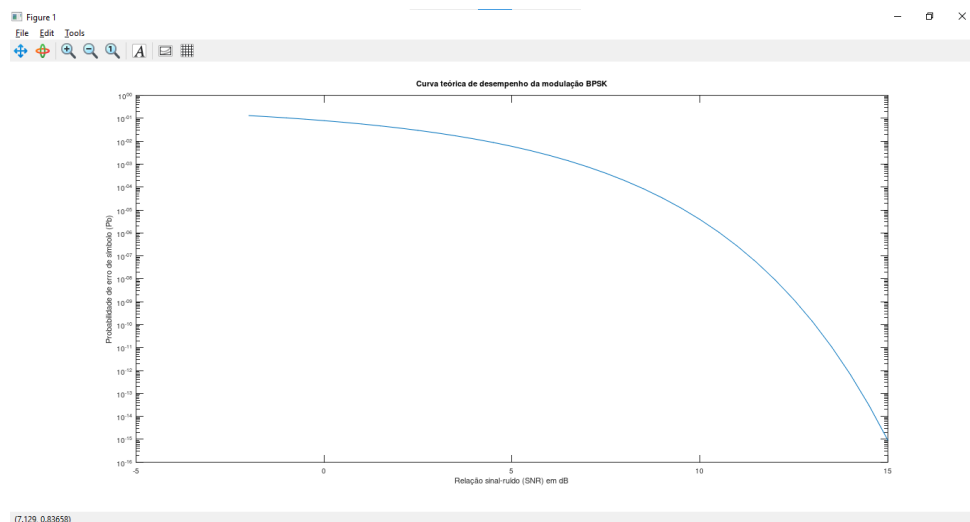
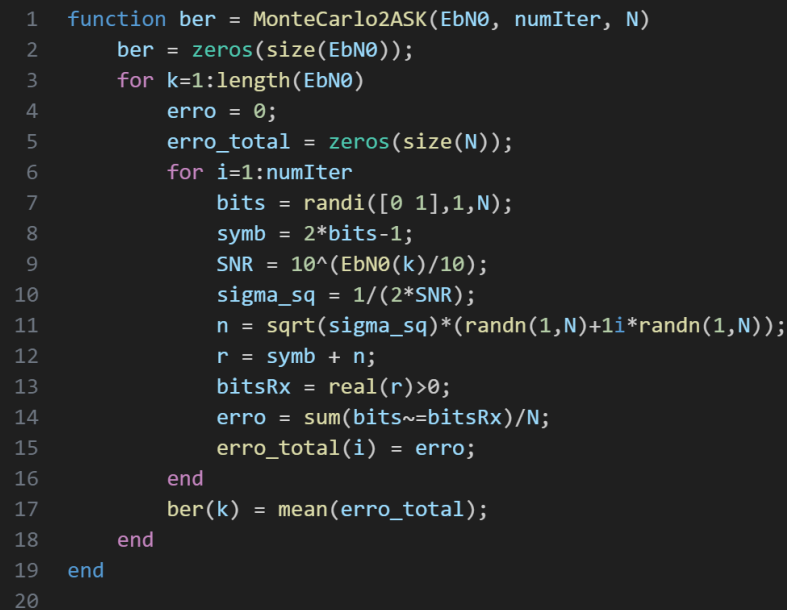


Figura 2: Curva teórica da simulada BPSK

**b) Simular uma modulação BPSK (2-PSK) usando a técnica de Monte Carlo para estimar a probabilidade de erro de bit (BER) para as seguintes relações sinal-ruído ( $E_b/N_0$ ): 2 dB, 5 dB, 8 dB e 10 dB.**

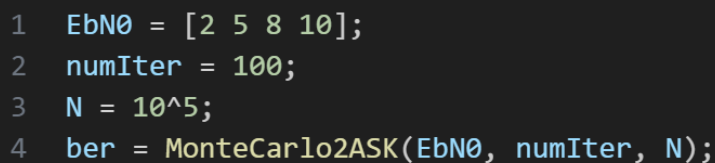
Foi desenvolvido o seguinte código ilustrado pelas Figura 3 e Figura 4:



```
1 function ber = MonteCarlo2ASK(EbN0, numIter, N)
2     ber = zeros(size(EbN0));
3     for k=1:length(EbN0)
4         erro = 0;
5         erro_total = zeros(size(N));
6         for i=1:numIter
7             bits = randi([0 1],1,N);
8             symb = 2*bits-1;
9             SNR = 10^(EbN0(k)/10);
10            sigma_sq = 1/(2*SNR);
11            n = sqrt(sigma_sq)*(randn(1,N)+1i*randn(1,N));
12            r = symb + n;
13            bitsRx = real(r)>0;
14            erro = sum(bits~=bitsRx)/N;
15            erro_total(i) = erro;
16        end
17        ber(k) = mean(erro_total);
18    end
19 end
20
```

Figura 3: Código da simulação MonteCarlor BPSK

Em seguida foi feito a simulação utilizando os seguintes parâmetros:



```
1 EbN0 = [2 5 8 10];
2 numIter = 100;
3 N = 10^5;
4 ber = MonteCarlo2ASK(EbN0, numIter, N);
```

Figura 4: Parâmetros da simulação BPSK

**c) Plotar os pontos simulados do item b) juntamente com a curva teórica de desempenho da modulação do item a).**

Realizando simulação chegou-se no seguinte resultado:

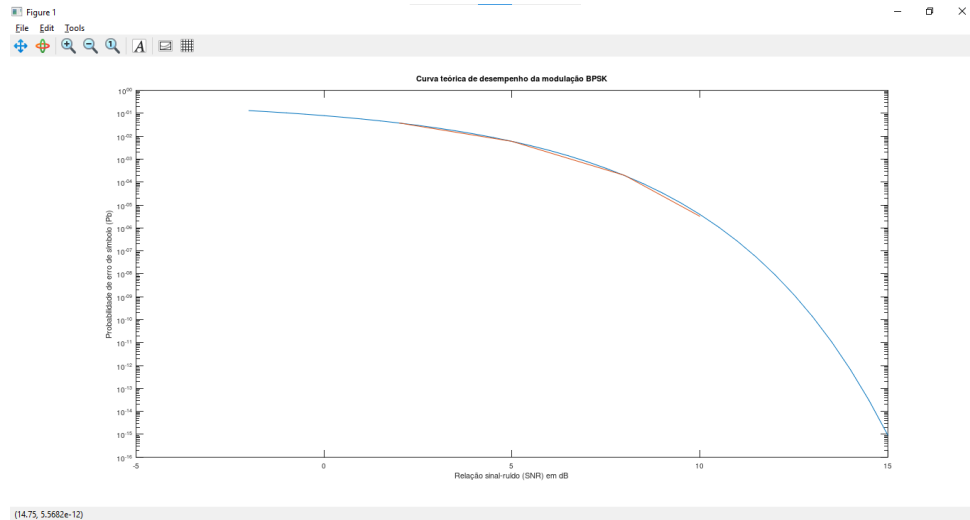


Figura 5: Resultado da simulação MonteCarlor BPSK

Analisando o gráfico é possível ver que os resultados simulados e calculados da curva teórica são bem semelhantes, ou seja, estão bem próximos do esperado.

**d) Simular uma modulação 8-ASK (polar) usando a técnica de Monte Carlo para estimar a probabilidade de erro de símbolo (SER) para as seguintes relações sinal-ruído ( $E_b/N_0$ ): 0 dB, 5 dB, 10 dB e 15 dB**

Para iniciar essa etapa foi feito o cálculo da curva teórica do 8ASK, cujo o código está ilustrado pela Figura 6 a seguir:

```
1 function Curva_teorica_8ASK()
2 SNR_dB = -2:0.5:15;
3 SNR = 10.^(SNR_dB/10);
4 Pb = 7/8 * erfc(sqrt(3/(8^2-1)*SNR));
5 semilogy(SNR_dB, Pb);
6 xlabel('Relação sinal-ruído (Eb/N0) em dB');
7 ylabel('Probabilidade de erro de símbolo (Pb)');
8 title('Curva teórica de desempenho da modulação 8ASK');
9 end
10
```

Figura 6: Código da simulação 8 ASK

Em seguida foi feito o código da simulação do de MonteCarlo do 8ASK, cujo o cpodigo esta disponibilizado pela Figura 7 a seguir:

```
1 function ser = MonteCarlo8ASK(EbN0, numIter, N)
2     ser = zeros(size(EbN0));
3     const = [-7 -5 -3 -1 1 3 5 7];
4     Es = mean(const.^2);
5     symb = const/sqrt(Es);
6     for k=1:length(EbN0)
7         erro = 0;
8         SNR = 10^(EbN0(k)/10);
9         sigma_sq = 1 / (2*SNR);
10        sigma = sqrt(sigma_sq);
11        for i=1:numIter
12            ind = randi(8);
13            simbolo_constelacao = symb(ind);
14            noise = randn * sigma;
15            simbolo_noise = simbolo_constelacao + noise;
16            [v,ind2] = min(abs(symb - simbolo_noise));
17            simbolo_recebido = symb(ind2);
18            if (simbolo_recebido ~= simbolo_constelacao)
19                erro = erro + 1;
20            end
21        end
22        ser(k) = erro/numIter;
23    end
24 end
25
26
```

Figura 7: Código da simulação 8 ASK

### e) Plotar os pontos simulados juntamente com a curva teórica de desempenho

Em seguida foi feito a simulação utilizando os seguintes parâmetros;

```

1 function ser = MonteCarlo8ASK(EbN0, numIter, N)
2     ser = zeros(size(EbN0));
3     const = [-7 -5 -3 -1 1 3 5 7];
4     Es = mean(const.^2);
5     symb = const/sqrt(Es);
6     for k=1:length(EbN0)
7         erro = 0;
8         SNR = 10^(EbN0(k)/10);
9         sigma_sq = 1 / (2*SNR);
10        sigma = sqrt(sigma_sq);
11        for i=1:numIter
12            ind = randi(8);
13            simbolo_constelacao = symb(ind);
14            noise = randn * sigma;
15            simbolo_noise = simbolo_constelacao + noise;
16            [v,ind2] = min(abs(symb - simbolo_noise));
17            simbolo_recebido = symb(ind2);
18            if (simbolo_recebido ~= simbolo_constelacao)
19                erro = erro + 1;
20            end
21        end
22        ser(k) = erro/numIter;
23    end
24 end
25
26

```

Figura 8: Código da simulação 8ASK

Em seguida foi feito plotagem das curvas, as quais estão ilustradas pela Figura 9, a seguir:

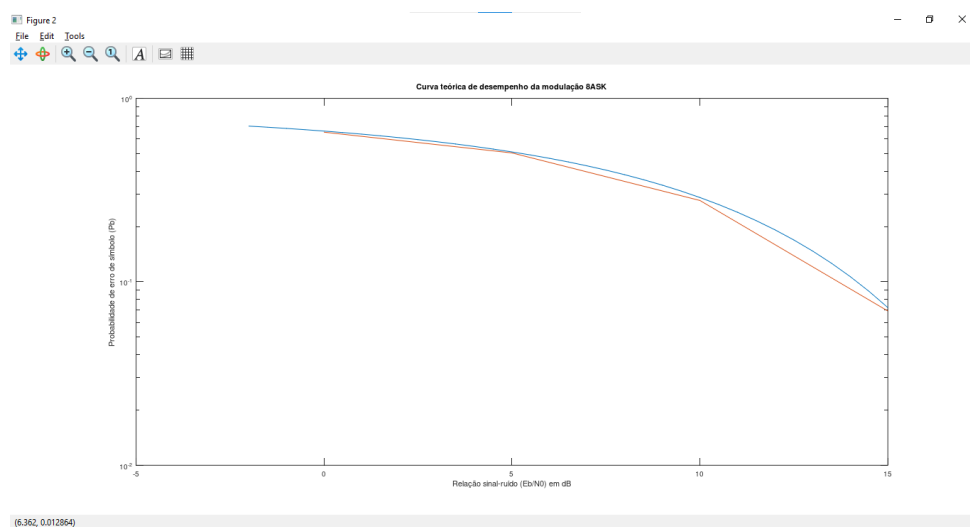


Figura 9: Resultado da simulação 8 ASK

Conforme o esperado as curvas ficaram bem próximas.

**f) Simular uma modulação 16-QAM usando a técnica de Monte Carlo para estimar a probabilidade de erro de símbolo (SER) para as seguintes relações sinal-ruído ( $E_b/N_0$ ): 0 dB, 5 dB, 10 dB e 12 dB.**

Para iniciar a simulação de uma modulação 16-QAM, em primeiro lugar é preciso entender que a mesma possui uma constelação em 2D, o que é diferente de todas as outras modulações simuladas até agora.

Sendo assim, para simular esse comportamento 2D, foram definidos dois vetores onde cada um indica os “eixos” X e Y desse tipo de modulação. Em sequência, a constelação como um todo foi normalizada e foram gerados 100000 valores aleatórios entre 1 e 4.

Apartir disso, foram calculados os valores de desvio e padrão e variancia, os quais foram usados para a geração de ruído, que ao ser somado com a constelação normalizadas em função dos valores sorteados, geram o sinal recebido.

Para a conferência da quantidade de erros foi feita uma comparação entre os valores da modulação QAM, sem ruído, com os valores encontrados através da função `quantalph`. No caso desses valores serem diferentes, isso indica que foi obtido um erro, e isso é contabilizado através da variável `erro`. Vale ressaltar que isso tudo é feito para os dois eixos da constelação.

Em sequência disso é calculada a taxa de erros de bits, com base na quantidade de erros e na quantidade de interações, que no caso foi 100000.

**g) Plotar os pontos simulados juntamente com a curva teórica de desempenho.**

Rodando o código chegou-se no seguinte resultado ilustrado pela

`cod16` Com os todos os passos descritos no item f, torna-se possível plotar uma curva simulada da modulação 16-QAM. Para plotar a curva teórica, foi elaborado o seguinte código.

```

1
2 %***** Curva teórica *****
3
4 SNRdB = [0:0.1:15];
5 SNR = 10.^(SNRdB./10);
6
7 M = 16;
8 Prob = ((1-1/sqrt(M))*erfc(sqrt(3/(2*(M-1)).*SNR)));
9
10 Ps = 1.-(1.-Prob).^2;
11
12 function y=quantalph(x,alphabet)
13 % function y=quantalph(x,alphabet)
14 % quantize the input signal x to the alphabet
15 % using nearest neighbor method
16 % input x - vector to be quantized
17 % alphabet - vector of discrete values that y can take on
18 % sorted in ascending order
19 % output y - quantized vector
20 [r c] = size(alphabet); if c>r, alphabet=alphabet'; end
21 [r c] = size(x); if c>r, x=x'; end
22 alpha=alphabet(:,ones(size(x)))';
23 dist=(x(:,ones(size(alphabet))))-alpha).^2;
24 [v,i]=min(dist,[],2);
25 y=alphabet(i);
26 end
27
28 %***** Curva simulada *****
29
30 EB_N0_dB=[0 5 10 12];
31
32 Const_X = [-3 -1 1 3];
33 Const_Y = [-3 -1 1 3];
34
35 Es = mean(Const_X.^2 + Const_Y.^2);
36 Const_X_norm = Const_X./sqrt(Es);
37 Const_Y_norm = Const_Y./sqrt(Es);
38
39 Const_X_data = randi(4,100000,1);
40 Const_Y_data = randi(4,100000,1);
41
42 erro = 0;
43
44 for snr=1:1:4
45     EB_N0(snr) = 10*(EB_N0_dB(snr)/10);
46     desvioPad = 1/(2*EB_N0(snr));
47     variancia = sqrt(desvioPad);
48
49     for i=1:1:100000
50         Const_X_qam(i) = Const_X_norm(Const_X_data(i));
51         Const_Y_qam(i) = Const_Y_norm(Const_Y_data(i));
52         Ruído_X(i) = randn*variancia;
53         Ruído_Y(i) = randn*variancia;
54         X_rx(i) = Const_X_qam(i)+Ruído_X(i);
55         Y_rx(i) = Const_Y_qam(i)+Ruído_Y(i);
56     endfor
57
58     X_rx_quant = quantalph(X_rx, Const_X_norm);
59     Y_rx_quant = quantalph(Y_rx, Const_Y_norm);
60
61     for k=1:1:100000
62         if(Const_X_qam(k) ~= X_rx_quant(k)) || (Const_Y_qam(k) ~= Y_rx_quant(k))
63             erro = erro+1;
64         end
65     endfor
66
67     BER(snr) = erro/100000;
68     erro = 0;
69 endfor
70
71 figure(1);
72 semilogy(SNRdB,Ps);
73 hold on;
74 semilogy(EB_N0_dB, BER, 'r+');
75 title("Desempenho 16-QAM");
76 xlabel("SNR[dB]");
77 ylabel("BER");
78 legend("teórica","simulada");
79
80 figure;
81 scatter(X_rx(:,1:1:1000),Y_rx(1,1:1:1000),'.');
82 title("Constelação recebida");
83

```

Figura 10: Código da simulação 16 QAM

Sendo assim, foi possível gerar a figura demonstrada abaixo, onde a curva em azul indica o resultado teórico. Enquanto que os pontos em vermelho mostram o resultado simulado. Isso é indicativo favorável de que a simulação elaborada pela equipe está correta, além de ser uma forma de visualizar como a “prática” converge para o mesmo ponto da teórica, lógico que isso considerando as condições impostas por essa simulação. A Figura 11 abaixo mostra as curvas teórica e simulada.

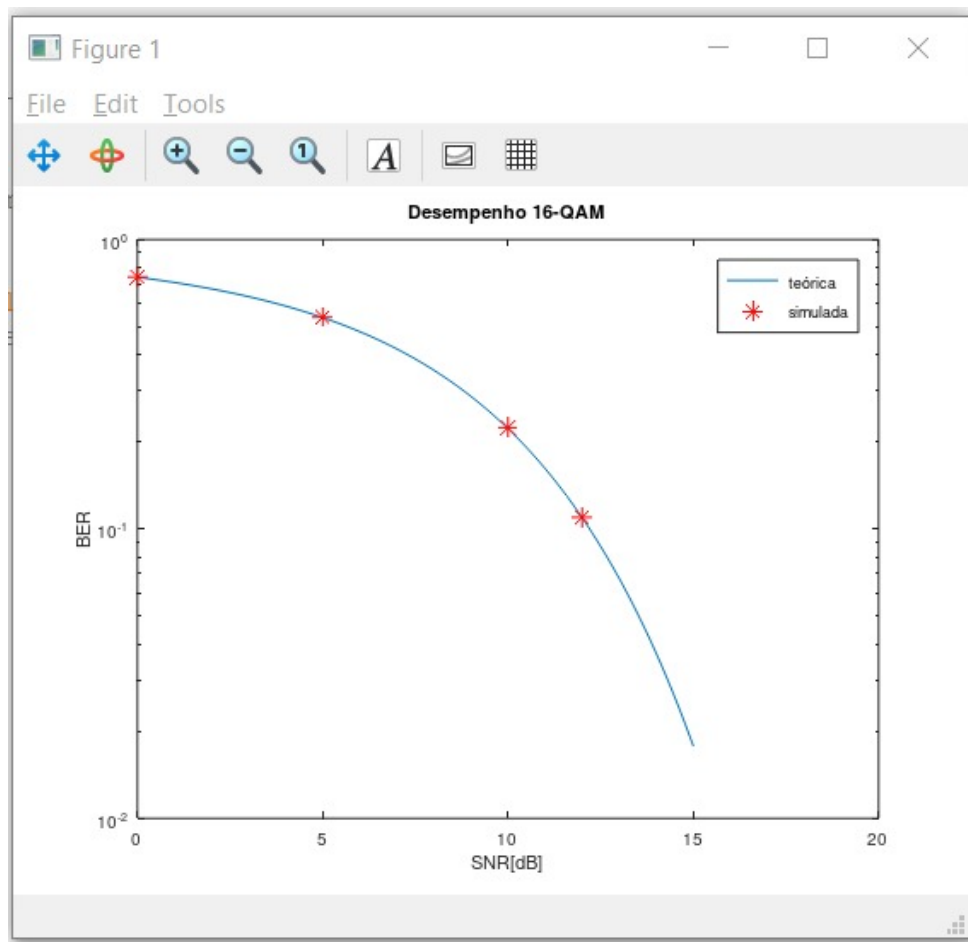


Figura 11: Resultado da simulação 8 ASK

**h) Adaptar o código de simulação do item e) para visualizar os pontos recebidos na constelação de sinais do 16-QAM, investigando o efeito da variação da SNR.**



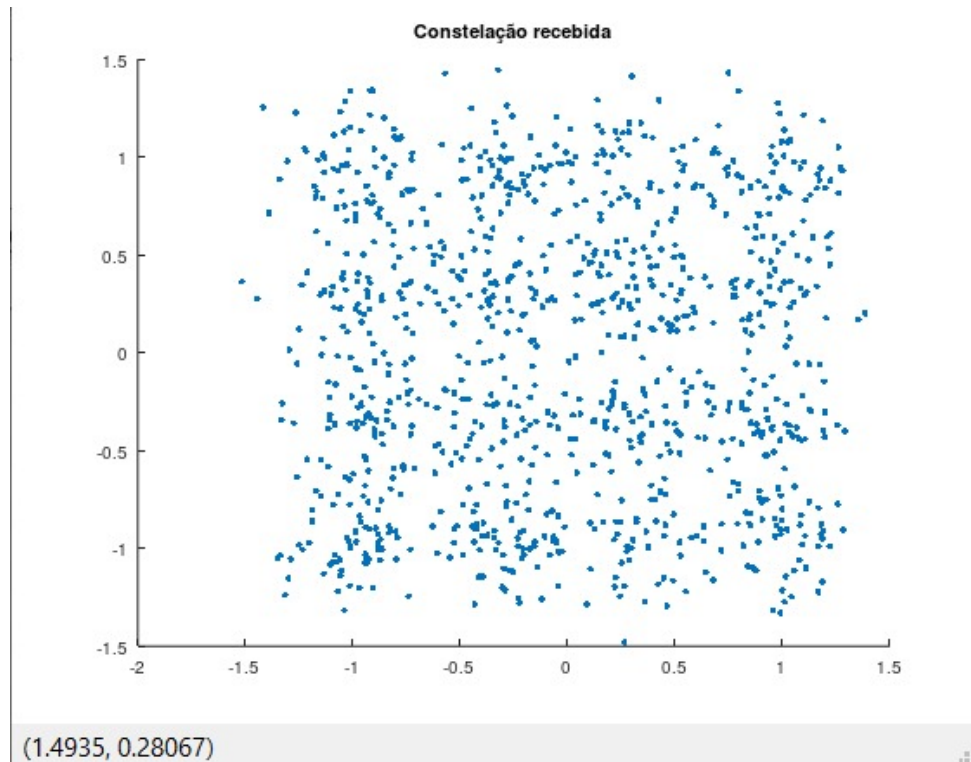


Figura 12: Constelações recebidas

Para plotar os pontos recebidos na constelação, foi usado os valores definidos nos vetores  $X_{rx}$  e  $Y_{rx}$ . O que gerou o resultado demonstrado na figura X. E como pode ser visualizado, a concentração dos pontos é semelhante ao que se espera de uma modulação do tipo 16-QAM, ou seja, 4 pontos por quadrante. Entranto, nota-se que ocorre uma grande variação na posição dos pontos, por isso os pontos não ficam concentrados exatamente no mesmo ponto. Isso se deve a variação de SNR, a qual é introduzida pela inclusão de ruído na simulação.