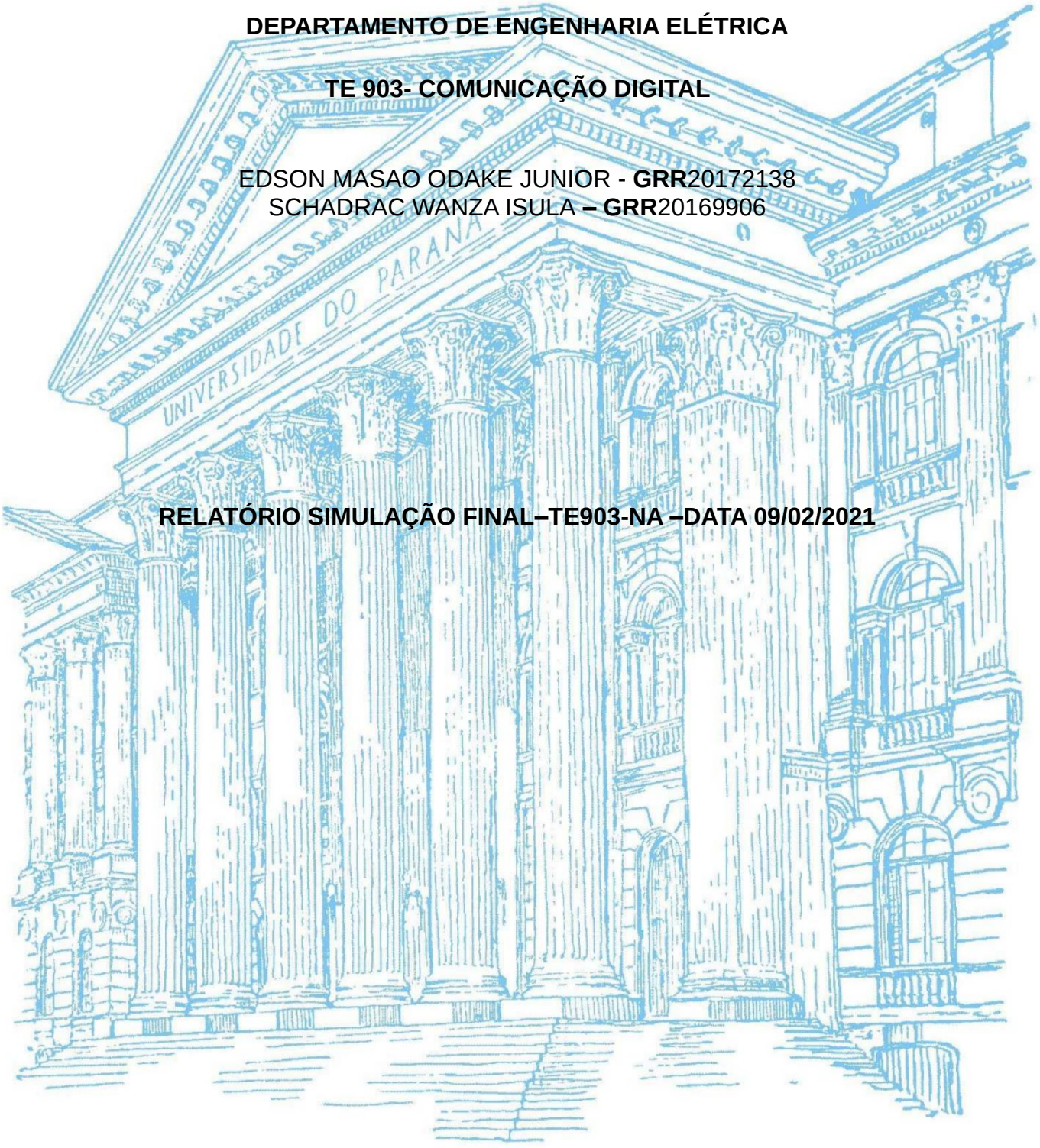
UNIVERSIDADE FEDERAL DO PARANÁ

SETOR DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

TE 903- COMUNICAÇÃO DIGITAL

EDSON MASAO ODAKE JUNIOR - **GRR**20172138
SCHADRAC WANZA ISULA **– GRR**20169906



RELATÓRIO SIMULAÇÃO FINAL–TE903-NA –DATA 09/02/2021

CURITIBA

2021

EDSON MASAO ODAKE JUNIOR - **GRR**20172138
Schadrac Wanza Isula – **GRR**20169906

RELATÓRIO SIMULAÇÃO FINAL– DATA 09/02/2021

Relatório acadêmico apresentado à a disciplina de comunicação digital, do curso de graduação em Engenharia Elétrica, da Universidade Federal do Paraná (UFPR).
Orientador: Prof. Evelio Martin Garcia Fernandez

CURITIBA

2021

SUMÁRIO

1 INTRODUÇÃO

Nessa última simulação será feita a transmissão de um sinal, no caso uma imagem, considerando todas as etapas de transmissão. Primeiramente iremos passa-lo por um codificador de canal, depois faremos a modulação M-QAM, com M = 16. Em seguida será adicionado o problema de um canal distorcido e um ruído gaussiano. Então utilizaremos um equalizador para que os efeitos do canal distorcido sejam amenizados. Caso não seja utilizado o equalizador, a imagem recebida acaba tendo uma taxa de erro de bits muito grande, mesmo que o equalizador também amplifique o ruído a sua presença ainda é muito necessária. Para terminar o sinal será demodulado e decodificado no receptor.

Esse trabalho final tem como finalidade a implementação do codificador de canal.

O sinal de ruído e interferências são introduzidos no sistema a partir do canal de comunicação. Isso leva a uma taxa de erro de bits maior encontrada no lado do receptor. A codificação de canal tem como intuito reduzir o valor de SNR necessário para se manter uma comunicação ideal.

Esse processo acorre através da adição de bits “redundantes” de modo que possa ocorrer a detecção de bits que foram recebidos de forma errada. Portanto uma quantidade **k** de bits em um bloco terá **n** bits depois da codificação representando a mesma mensagem. Ao se usar desse método existe também um aumento da largura de banda necessária para transmissão.

A codificação do canal ocorre no transmissor antes que ocorra a modulação, e a decodificação no receptor logo depois da demodulação. Essa transformação ocorre de forma digital. Na simulação utilizaremos a ferramenta *bertool* do matlab para que possamos encontrar o valor de **n** e **k** cujo resultado leve a um ganho de pelo menos 2 dB.

2 BERTOOL - Matlab

Primeiramente, para que possamos fazer a simulação no MATLAB, vamos usar o bertool para encontrar valores de **n** e **k que** resultem um ganho de no mínimo 2,12 dB em relação ao canal não codificado considerando uma BER = 10e-6. A figura 1 mostra o setup no ambiente bertool e a figura 2 mostra o resultado obtido. Em que a curva azul escura é o BER sem codificação e o azul claro é o BER do sinal codificado.

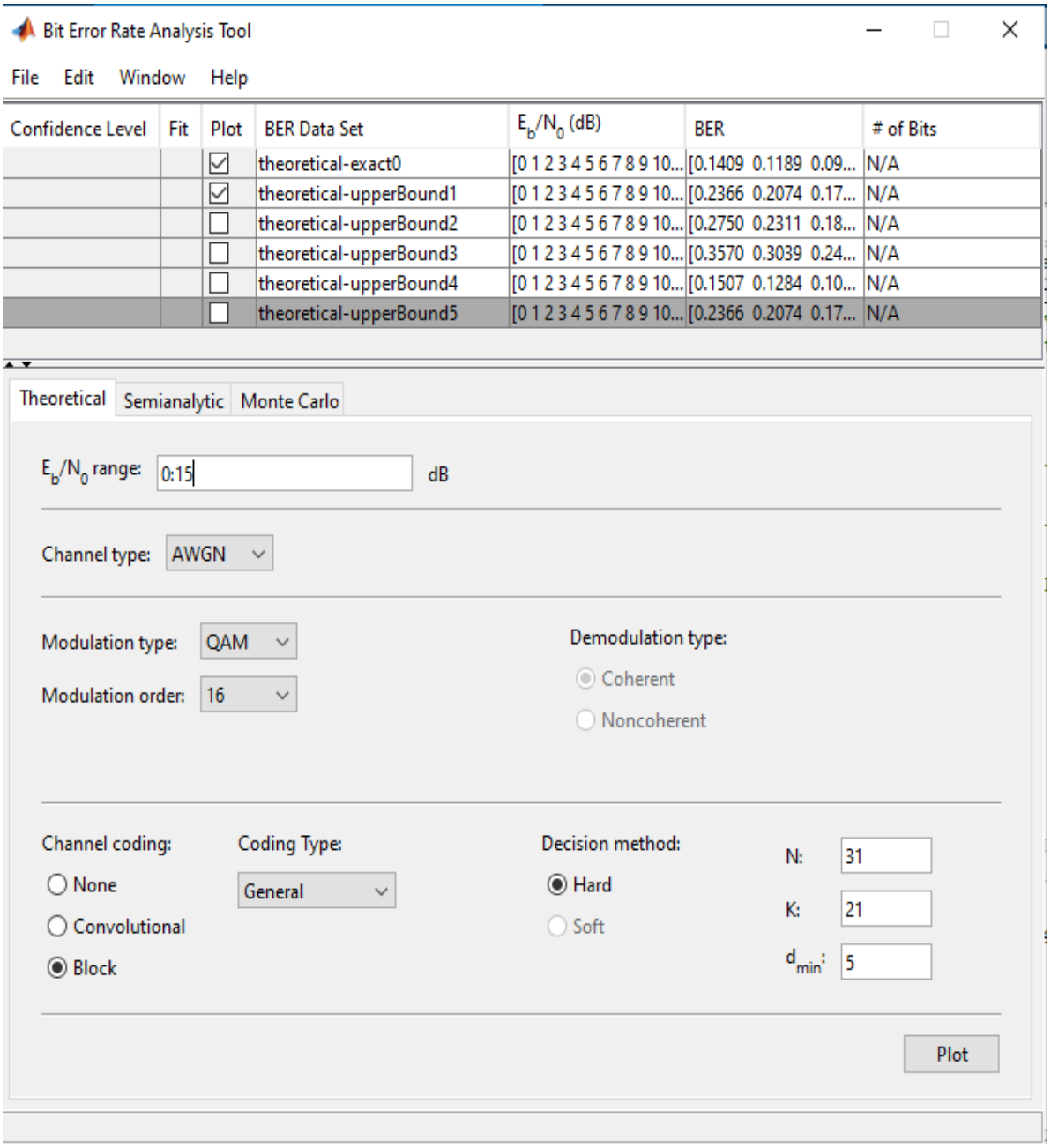


Figura 1: Bertool matlab

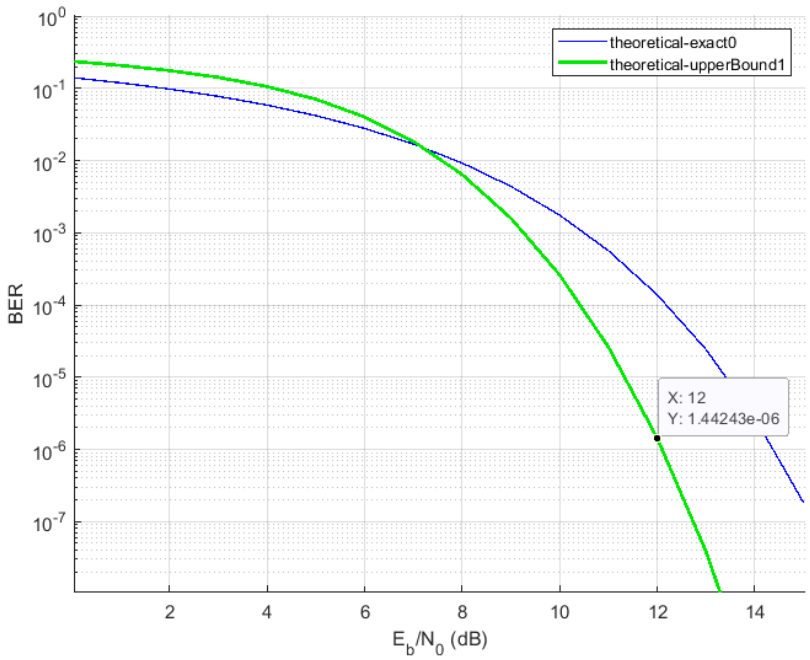


Figura 2: Curvas de Desempenhos de Erro

Pela figura 2 a cima, podemos ver as curvas de desempenhos de erro (BER) sem e com código. Foi pedido fazer escolha de maneira que a diferencia (analiticamente) do ganho entre a curva sem e com código no ponto 10e-6 seja de 2 db. Observamos que o código melhor para tal resultado é N=31, K=21 e dmin= 5.

3 Procedimento para Cálculo de BER

Para obter a expressão analítica de BER foi usado as equações abaixo, com os valores de código de correção, ou seja, N=31, K=21, dmin=5.

$$\frac{E_s}{N_o} = (\log_2(M))(\frac{k}{n})\frac{E_b}{n_o};$$

$$P_e = erfc(\sqrt{(\frac{E_s}{N_0})}\sin(\frac{\pi}{M}));$$

$$P_c = \frac{P_e}{\log_2(M)};$$

$$BER = \frac{1}{n}\sum_{j=t+1}^{n} j\,(\binom{n}{j})P_c^{\,j}(1-P_c)^{n-j};$$

$$t = \frac{dmin-1}{2}$$

Os cálculos foram feitos no matlab e o foi gerado um gráfico mostrando o valor de BER em relação a variação Eb/N0. Isso é mostrado na figura 3.

O script utilizado para gerar o gráfico foi o código abaixo.

```
clear; clc; close all;

n=31;
k=21;
t=2;
M=16;
ebn0=[0:1:15];

% n=63;
% k=51;
% t=2;
% ebn0=20.89;
% M=8;

esn0=log2(M)*ebn0*k/n;

pe=erfc(sqrt(esn0)*sin(pi/M));

pc=pe/log(M);
BER=0;
for j=t+1:1:n;
```

```
BER=BER+j*(factorial(n)/(factorial(j)*factorial(n-j))).*pc.^j.*(1-pc).^(n-j)/n;
end

plot(ebn0,2*BER);
semilogy(ebn0,BER,'LineWidth',2)
xlabel("Eb/n0");
ylabel("BER");
axis([0 15 1e-3 1]);
```

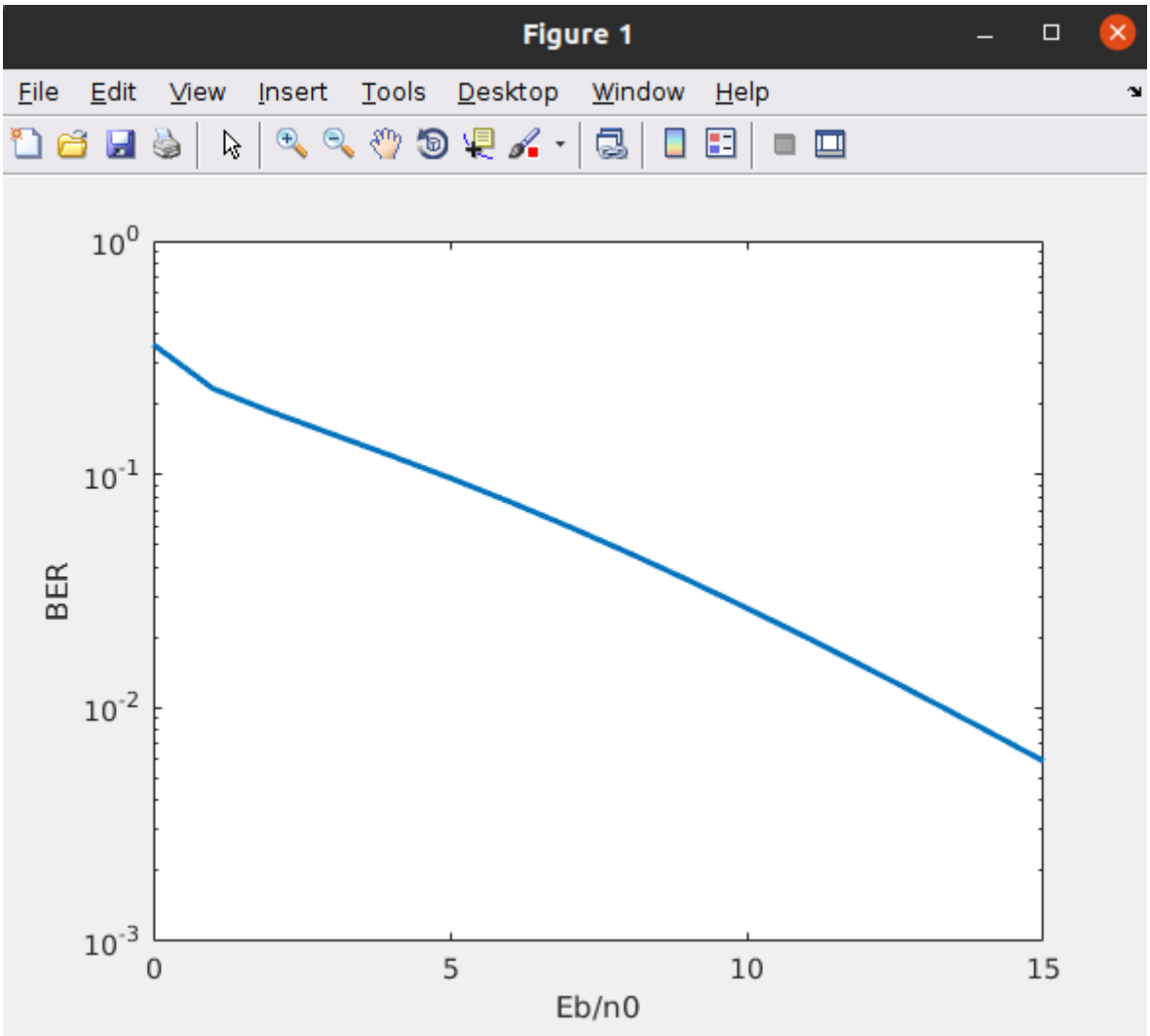


Figura 3: BER calculado

4 Curvas

Nessa parte é necessário plotar as curvas de desempenhos de erro (BER vs. Eb/No), no mesmo gráfico plotar 3 curvas, uma para o sistema sem código de correção de erro (simulado), uma para sistema com código (analítico) e enfim sistema com codificação de canal simulado.

Com tentativa e erro, com falta de conhecimento aprofundada no matlab, e com a demora na compilação, infelizmente não conseguimos chegar no resultado esperado nessa parte de simulação. Tentamos resolver, mas não deu certo e não terá figura das curvas.

5 Conclusão

Como é mostrado na figura 2, teoricamente, a utilização de codificação de canal se mostra bem mais efetivo que o canal não codificado. Foi possível ver um ganho de 2.12 dB do canal codificado em relação ao não codificado. É claro que a codificação do canal também apresenta alguns pontos positivo. Por exemplo, com a codificação do canal, vem junto um aumento na banda para transmitir a mensagem.

Os gráficos encontrados pelos diferentes métodos computacionais podem apresentar resultados distintos, porém semelhantes. No caso o valor encontrado pela simulação e o valor do BER calculado mostram muita diferença em seu resultado. Isso nós levamos a acreditar que pode ter ocorrido um erro no código que gerou o gráfico.

6 Código da Simulação (script)

```
% --------------------------------------------------------------------------
% Simulacao do Equivalente em banda Base de um Sistema 16-QAM
% Transmissao atraves de um Canal Seletivo em Frequencia
% Equalizacao ZF
% --------------------------------------------------------------------------

clear; clc; close all;

% --------------------------------------------------------------------------
% Parametros
% --------------------------------------------------------------------------

Fs=44100;           % Frequencia de Amostragem do Sinal Continuo (Hz)
Ts=1/Fs;            % Periodo de Amostragem (s)

Fc=8000;            % Frequencia da Portadora (Hz)

oversampling=10;    % Fator Fs/R

R=Fs/oversampling;  % Taxa de Transmissao em simbolos/s (baud rate)
T=1/R;              % Periodo de Simbolo (s)

del=25;             % Resposta do filtro formatador se estende por (2*del) per?odos de simbolo
                    % Numero de amostras do filtro formatador: N=2*(del*oversampling)+1

rolloff=0.5;          % Fator de rolloff dos filtros Tx e Rx
N= 31;
K= 21;
t=2;
palavras = 1e5;

% --------------------------------------------------------------------------
% Geracao de bits aleatorios e mapeamento 4-PAM
%--------------------------------------------------------------------------
```

```
% Geracao de bits aleatorios e mapeamento 4-PAM
%------------------------------------------------------------------------

bits_msg = round(rand(palavras,K));   %geracao de um vetor de bits aleatorios
                                      %cada linha --> bloco de informa��o de k bits
bits_msg_gf = gf(bits_msg);
msg_coded = bchenc(bits_msg_gf,N,K);
bit_symbols = reshape(msg_coded,palavras*N/2,2);
bit_symbols=double(bit_symbols.x);
symbols=bi2de(bit_symbols);            %dibits a ser transmitidos

a=[-3 -1 1 3];
symbols=symbols+1;
pam=a(symbols);

 pam_I=pam(1:2:length(pam));    %Simbolso em fase e quadratura
 pam_Q=pam(2:2:length(pam));


% ------------------------------------------------------------------------
% Filtro de Tx+Rx
% Formatacao de Pulso - Tipo Raiz Quadrada do Cosseno Levantado no Tx e Rx
% ------------------------------------------------------------------------
filtro=rcosfir(rolloff,del,oversampling,1,'sqrt');

%Formatar e transmitir os simbolos PAM

sinal_tx_I=upsample(pam_I,oversampling);           % Realiza Upsampling
sinal_tx_Q=upsample(pam_Q,oversampling);
sinal_tx_filtrado_I=conv(sinal_tx_I,filtro);       % Sinal Filtrado de Transmissao
sinal_tx_filtrado_Q=conv(sinal_tx_Q,filtro);

sinal_QAM=sinal_tx_filtrado_I+1j*sinal_tx_filtrado_Q;


% ------------------------------------------------------------------------
% Canal com multipercurso (ISI channel)
% ------------------------------------------------------------------------

ISI_channel = [0.19+.56j .45-1.28j -.14-.53j -.19+.23j .33+.51j]; % Exemplo de canal complexo

% ------------------------------------------------------------------------
% Equalizador de ZF baseado em LS
% ------------------------------------------------------------------------
L1=1; L2=3; % O tap m�ximo do canal
N1=5; N2=10; % O comprimento do equalizador � N1 + N2 + 1
%N1=2; N2=5; % O comprimento do equalizador � N1 + N2 + 1
P=convmtx(ISI_channel.',N1+N2+1); % Matriz de convolu��o
u_ZF=zeros(N1+N2+L1+L2+1,1); % u vector (zero forcing)
u_ZF(L1+N1+1)=1; % coloca o um no lugar correspondiente
c_LS=((P'*P)\(P'))*u_ZF; % equalizador LS taps com pseudoinverso

figure(1);

%Plotando a funcao de transferencia do canal, do equalizador, e do canal
%equalizado

[H,F]=freqz(ISI_channel,1,2048,'whole',Fs); % Fun��o de transferencia do canal
gain=20*log10(fftshift(abs(H)));
plot((F-Fs/2)/1000,gain,'LineWidth',2);
hold;
[H_LS,F]=freqz(c_LS,1,2048,'whole',Fs); % Funcao de transferencia do equalizsdor
gain=20*log10(fftshift(abs(H_LS)));
plot((F-Fs/2)/1000,gain,'r','LineWidth',2);
H_tot=H.*H_LS;                          % Funcao de transferencia conjunta canal-equalizador
gain=20*log10(fftshift(abs(H_tot)));
plot((F-Fs/2)/1000,gain,'k','LineWidth',2); grid;
axis([-Fs/(2*1000) Fs/(2*1000) -20 15]);
xlabel('Frequ�ncia (kHz)');
ylabel('Ganho (dB)');
legend('Resposta do Canal','Resposta do Equalizador','Resposta Total');
```

```
ebn0=[0:1:15];

esn0=log2(16)*ebn0*K/N;

pe=erfc(sqrt(esn0)*sin(pi/16));

pc=pe/log(16);
BER_2=0;
for j=t+1:1:N;
    BER_2=BER_2+j*(factorial(N)/(factorial(j)*factorial(N-j))).*pc.^j.*(1-pc).^(N-j)/N;
end


sinal_rx_ISI=filter(ISI_channel,1,sinal_QAM);

ebn0=[0:1:15];
M=4;      % Duas modulações 4-PAM em quadratura

for k=1:1:length(ebn0);

sinal_rx_ISI_ruido = awgn(sinal_rx_ISI,(ebn0(k)+10*log10(log2(M))-10*log10(oversampling/2)),'measured');

%sinal_rx_ISI_ruido = sinal_rx_ISI;

%sinal_rx_ISI_ruido = sinal_rx_ISI_ruido(L1+1:end); % Descarte o atraso, se o canal for anti-causal

sinal_equalizado=filter(c_LS,1,sinal_rx_ISI_ruido);   % Sinal equalizado
sinal_equalizado=sinal_equalizado(L1+N1+1:end);       % Descarte transitorio do equalizador


% Receptor (Filtro Casado)
% -------------------------------------------------------------------------

sinal_rx_casado_I=conv(real(sinal_equalizado),filtro);       %Filtro casado
sinal_rx_casado_Q=conv(imag(sinal_equalizado),filtro);


pam_rx_I=downsample(sinal_rx_casado_I,oversampling);
pam_rx_Q=downsample(sinal_rx_casado_Q,oversampling);
pam_rx_I=pam_rx_I(del*2+1:length(pam_rx_I)-del*2);
pam_rx_Q=pam_rx_Q(del*2+1:length(pam_rx_Q)-del*2);

%Estimacao dos simbolos PAM recebidos (fase e quadratura)

alphabet=[-3 -1 1 3];
symbols_rx_quant_I=quantalph(pam_rx_I,alphabet);
symbols_rx_quant_Q=quantalph(pam_rx_Q,alphabet);

a=[0 1 2 3];
symbols_rx_I=(symbols_rx_quant_I+3)/2+1;
symbols_rx_Q=(symbols_rx_quant_Q+3)/2+1;
symbols_rx=zeros(1,length(pam));
symbols_rx(1:2:length(pam))=symbols_rx_I;
symbols_rx(2:2:length(pam))=symbols_rx_Q;
symbols_rx=a(symbols_rx);

%symbols_rx=double(symbols_rx.x);

bit_symbols_rx=de2bi(symbols_rx);
bits_msg_rx_1=reshape(bit_symbols_rx,length(bit_symbols_rx)*2/N,N);

%bits_msg_rx=double(bits_msg_rx.x);

bits_msg_rx=gf(bits_msg_rx_1);
bits_msg_rx=bchdec(bits_msg_rx,N,K);
```

```
bits_msg_rx=gf(bits_msg_rx_1);
bits_msg_rx=bchdec(bits_msg_rx,N,K);

[num_erros(k),BER(k)]=symerr(bits_msg_rx,bits_msg)
[num_erros_1(k),BER_1(k)]=symerr(bits_msg_rx_1,bits_msg)
end



x=sinal_rx_casado_I+j*sinal_rx_casado_Q;

scatterplot(pam_rx_I+j*pam_rx_Q)

figure(5);
semilogy(ebn0,BER_1,'LineWidth',2)
grid on
hold on
semilogy(ebn0,BER,'LineWidth',2)
semilogy(ebn0,BER_2,'LineWidth',2)
xlabel('E_b/N_0 (dB)');
ylabel('BER');
title('Taxa de Erro de Bits');
axis([0 15 1e-6 1]);
```

```
Current plot held


Error using symerr (line 76)
Only string and numeric arguments are accepted.

Error in ber_equivalenteBB_16QAM_multipath_equalizado_fonte_aleatoria (line 173)
[num_erros(k),BER(k)]=symerr(bits_msg_rx,bits_msg)
```