UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

Wagner Spinato Chittó

5º TRABALHO PRÁTICO

Codificações de Linha, Modulações Digitais e PSDs
Densidades Espectrais de Potência

Santa Maria, RS
26/09/2025

Objetivo:

Simulação das codificações de linha (banda base), modulações digitais (passa-faixa) e estimação de PSDs. Observação dos efeitos das codificações e formatos de pulsos na PSD do sinal transmitido.

Fundamentação:

1.1 - Arquivo de simulação “cod_linha_PSD.m”

```
clear all; close all; clc;
pkg load signal

Ts = 16; % pontos por simbolo
Tb = Ts; % tempo de duracao do bit
Nb=10000; % numero de bits
B=rand(1,Nb)>0.5; % gera Nb bits aleatoriamente
T = 1:50*Tb; % tempo usado para mostrar grafico

% pulsos
% pulso NRZ
pulso_nrz = ones(1, Ts);

% pulso RZ
pulso_rz = ones(1, Ts/2);
pulso_rz(Ts/2+1:Ts) =  zeros(1, Ts/2);

pulso = pulso_nrz;
%pulso = pulso_rz;

%Codifi��o polar NRZ
NRZ = [];
for b=B
    if (b==0)
        simbolo = -1;
    else
        simbolo = 1;
    end

    NRZ = [NRZ simbolo*pulso];

end

figure;
subplot(611);
p=plot(T,NRZ(T));
```

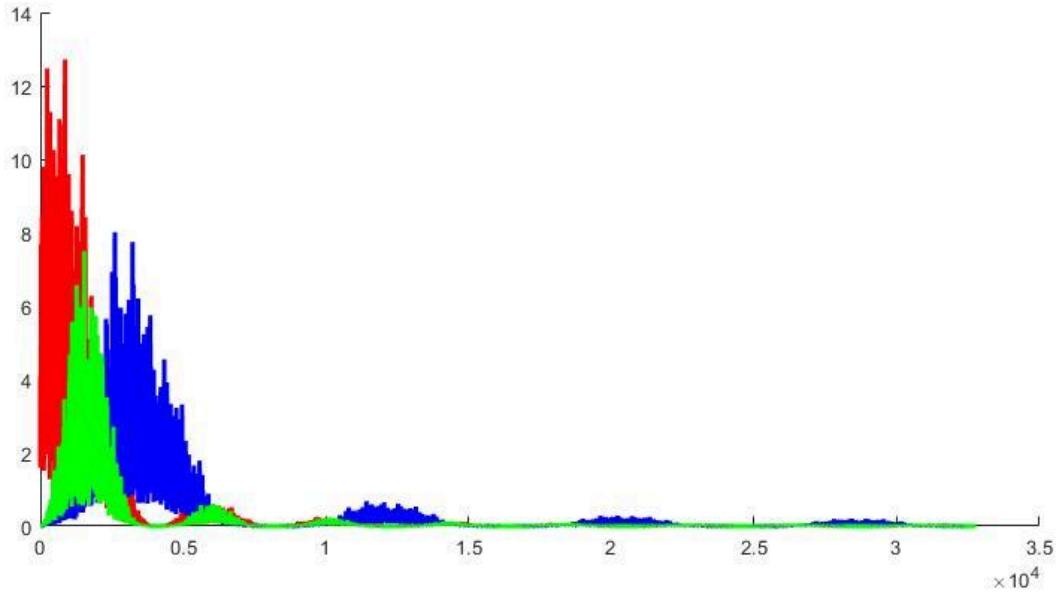
```
title('NRZ')
axis([0 length(T) -2 2]);
set(p,'Color','black','LineWidth',2.5)

% codigo NRZI
NRZI = [];

simbolo=-1;
for b = B
    if(b==1)
        simbolo = simbolo*(-1);
    end
    NRZI = [NRZI simbolo*pulso];
end

%grafico
subplot(612);
p=plot(T,NRZI(T));
title('NRZI');
axis([0 length(T) -2 2]);
set(p,'Color','black','LineWidth',2.5)

%% Bipolar AMI
AMI = [];

aux = 1;
for b=B
    if (b == 1)
        if (aux == 1)
            simbolo = 1;
            aux = 0;
        else
            simbolo = -1;
            aux = 1;
        end
    else
         simbolo = 0;
    end
    AMI = [AMI simbolo*pulso];
end
```

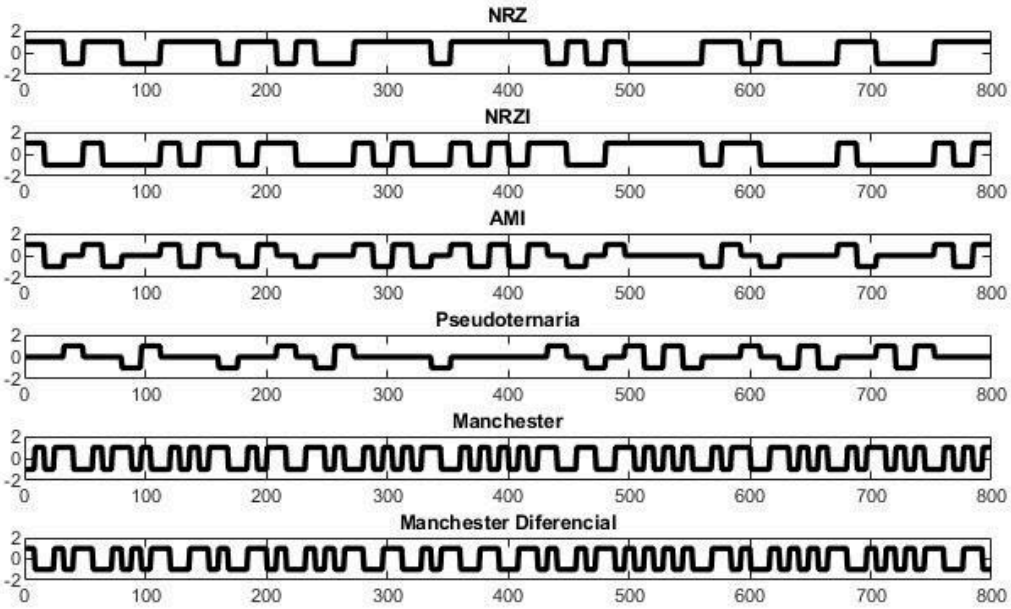
```
%grafico
subplot(613);
p=plot(T,AMI(T));
title('AMI')
axis([0 length(T) -2 2]);
set(p,'Color','black','LineWidth',2.5)

%PSEUDOTERNARIA
PSEUD = [];

aux = 1;
for b=B
    if (b == 0)
        if (aux == 1)
            simbolo = 1;
            aux = 0;
        else
            simbolo = -1;
            aux = 1;
        end
    else
         simbolo = 0;
    end

    PSEUD = [PSEUD simbolo*pulso];

end

subplot(614);
p=plot(T,PSEUD(T));
title('Pseudoternaria')
axis([0 length(T) -2 2]);
set(p,'Color','black','LineWidth',2.5)

%%MANCHESTER
MANCH=[];

m=1;
for b=B
    if (b==0)
```

```
        simbolo=1;
    else
        simbolo=-1;
    end;

    MANCH = [MANCH simbolo*pulso(1:Ts/2) (-1)*simbolo*pulso(Ts/2+1:Ts)];

end;

subplot(615);
p=plot(T,MANCH(T));
title('Manchester')
axis([0 length(T) -2 2]);
set(p,'Color','black','LineWidth',2.5)

%
%%MANCHESTER DIFERENCIAL

DIFFMANCH=[];
aux=-1;
for b=B
    if (b==0)
        simbolo=aux;
    elseif (b==1 && aux==-1)
        simbolo=1;
        aux=1;
    else
        simbolo=-1;
        aux=-1;
    end;

    DIFFMANCH = [DIFFMANCH simbolo*pulso(1:Ts/2)
(-1)*simbolo*pulso(Ts/2+1:Ts)];
end;

subplot(616);
p=plot(T,DIFFMANCH(T));
title('Manchester Diferencial')
axis([0 length(T) -2 2]);
set(p,'Color','black','LineWidth',2.5)
```

```
% Calculo de PSDs
% PSD usando metodo de Welch (pode ser usado tambem o periodogram)
figure;
hold on;

%Hpsd=psd(spectrum.welch,NRZ); % matlab
[Hpsd,f] = pwelch(NRZ); %octave
handle1=plot(Hpsd);
set(handle1,'LineWidth',2,'Color','r');

%Hpsd=psd(spectrum.welch,MANCH); % matlab
[Hpsd,f] = pwelch(MANCH); %octave
handle1=plot(Hpsd);
set(handle1,'LineWidth',2,'Color','b')

%Hpsd=psd(spectrum.welch,AMI); % matlab
[Hpsd,f] = pwelch(AMI); ; %octave
handle1=plot(Hpsd);
set(handle1,'LineWidth',2,'Color','g')
```

1.2 - outputs



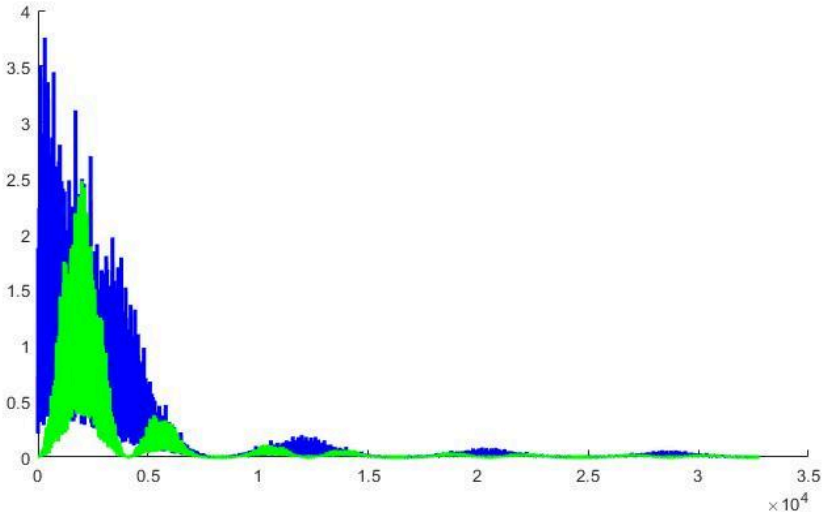

NRZ
NRZI
AMI
Pseudoternaria
Manchester
Manchester Diferencial
2
0
-2
0
100
200
300
400
500
600
700
800

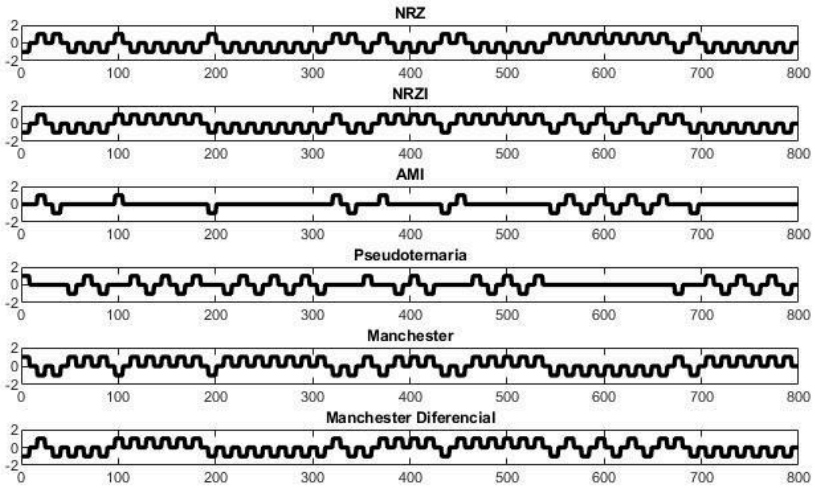
Procedimento:

2.1 - Modificação da simulação para utilizar o formato de pulso com retorno ao zero (pulso RZ) para a codificação polar:

```
pulso = pulso_rz;
```

2.2 - outputs:





2.3 - Conclusão

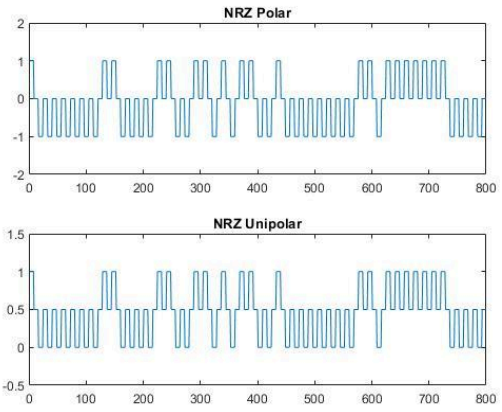
A simulação com pulso RZ mostra que o sinal apresenta maior número de transições, facilitando a recuperação de relógio, porém à custa de maior largura de banda em comparação ao pulso NRZ.

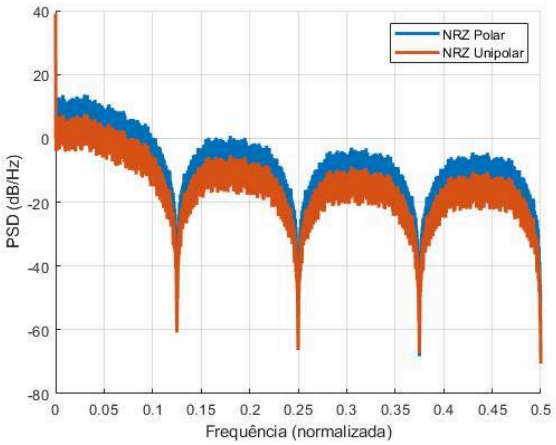
3.1 - Código para comparação do sinal polar vs unipolar:

```
NRZ_unipolar = (NRZ + 1) / 2;    % escala linear: -1..+1 -> 0..1

figure;
subplot(211);
plot(T, NRZ(T));
title('NRZ Polar');
axis([0 length(T) -2 2]);

subplot(212);
plot(T, NRZ_unipolar(T));
title('NRZ Unipolar');
axis([0 length(T) -0.5 1.5]);

figure; hold on;
[Px_polar,f] = pwelch(NRZ,[],[],[],1);            % sinal polar
[Px_unipol, ~] = pwelch(NRZ_unipolar,[],[],[],1);% sinal unipolar

plot(f,10*log10(Px_polar),'LineWidth',2);
plot(f,10*log10(Px_unipol),'LineWidth',2);
legend('NRZ Polar','NRZ Unipolar');
xlabel('Frequência (normalizada)');
ylabel('PSD (dB/Hz)');
grid on;
```

(código gerado com auxílio de IA)

3.2 - Outputs da comparação:





3.3 - Observações:

NRZ unipolar tem média positiva o que introduz forte componente em f=0 (linha DC).
NRZ polar tem média ≈0 e não exibe essa componente DC.
Fora o pico DC, as formas são semelhantes porque a variação temporal das transições é parecida, porém a presença do termo médio desloca energia para baixas frequências no unipolar. NRZ unipolar tem potência média diferente.

4.1 - Código para as codificações B8ZS e HDB3 e o cálculo das respectivas PSDs.

```
clear all; close all; clc;

%% Parâmetros
Ts = 16;              % samples por símbolo
Nb = 20000;           % número de bits (grande para estimativa PSD)
B  = rand(1,Nb) > 0.5;  % bits aleatórios
pulso = ones(1,Ts);  % pulso retangular NRZ

%% --- Funções auxiliares (inline) ---
% Converte sequência de símbolos (-1,0,1) para sinal amostrado usando
pulso
shape = @(symb) kron(symb, pulso);

%% --- Gera AMI (Alternate Mark Inversion) ---
AMI_sym = zeros(1,Nb);
last = -1;
for k=1:Nb
    if B(k)==1
        last = -last;        % alterna polaridade a cada '1'
        AMI_sym(k) = last;
    else
        AMI_sym(k) = 0;
    end
end
AMI = shape(AMI_sym);

%% --- Gera Pseudoternária (zeros -> +/- , ones -> 0) ---
PSEUD_sym = zeros(1,Nb);
alt = 1;
for k=1:Nb
    if B(k)==0
```

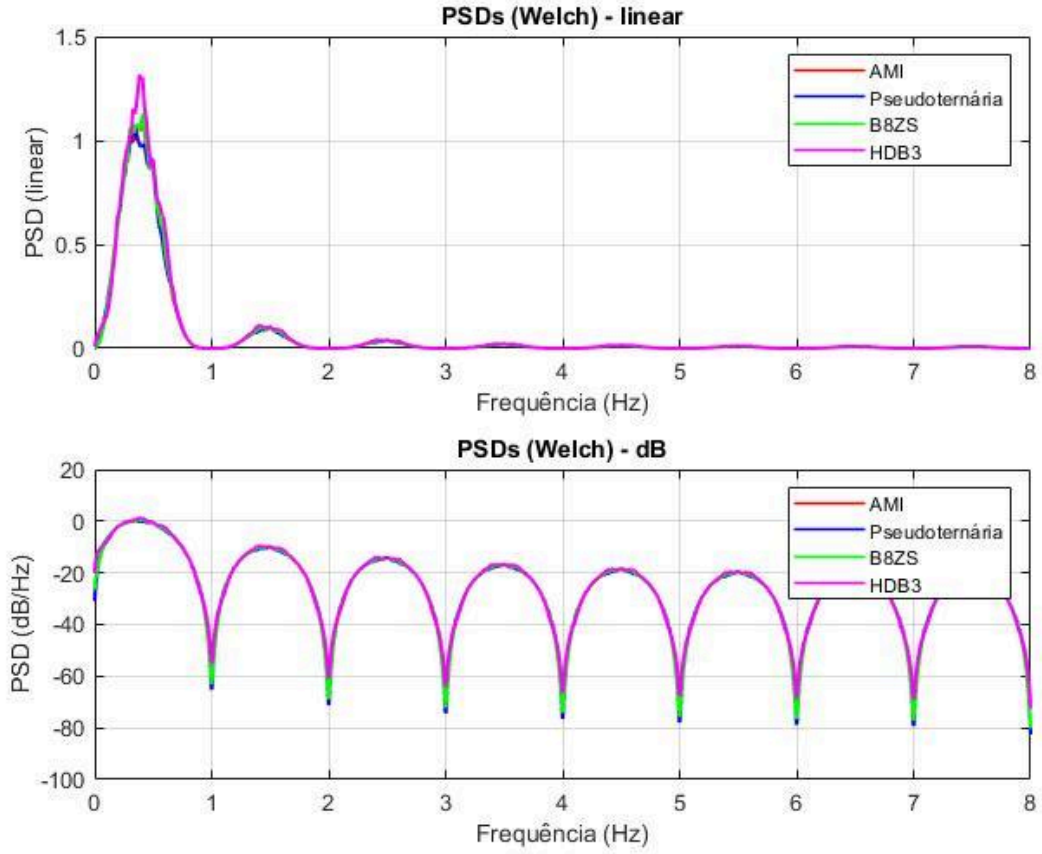
```
        PSEUD_sym(k) = alt;
        alt = -alt;
    else
        PSEUD_sym(k) = 0;
    end
end
PSEUD = shape(PSEUD_sym);

%% --- Gera B8ZS (substitui 8 zeros por 000VB0VB) ---
% trabalha sobre bits e produz símbolos AMI-modificados
B8ZS_sym = [];
last = -1; % polaridade do ultimo pulso transmitido (assuma -1
inicialmente)
k = 1;
while k <= Nb
    if k+7 <= Nb && all(B(k:k+7)==0)   % encontro de 8 zeros
        % V = violação = mesma polaridade do último pulso não-zero
        if last == 0, V = 1; else V = last; end
        Bp = -V; % B = bipolar pulse com polaridade oposta ao V
        % padrão 000 V B 0 V B
        block = [0 0 0 V Bp 0 V Bp];
        B8ZS_sym = [B8ZS_sym block];
        last = Bp;           % última polaridade transmitida após
substituição
        k = k + 8;
    else
        if B(k)==1
            last = -last;
            B8ZS_sym = [B8ZS_sym last];
        else
            B8ZS_sym = [B8ZS_sym 0];
        end
        k = k + 1;
    end
end
B8ZS = shape(B8ZS_sym);

%% --- Gera HDB3 (substitui 4 zeros por 000V ou B00V conforme paridade)
---
HDB3_sym = [];
```

```
last = -1;            % polaridade do último pulso não-zero (assuma -1)
count_nonzero = 0;    % número de pulsos (+/-) desde a última substituição
k = 1;
while k <= Nb
    if k+3 <= Nb && all(B(k:k+3)==0)  % encontro de 4 zeros
        if mod(count_nonzero,2) == 0
            % usar 000V : V tem mesma polaridade que last
            if last==0, V = 1; else V = last; end
            block = [0 0 0 V];
            % atualiza contagem: apenas V é não-zero
            count_nonzero = count_nonzero + 1;
            last = V;
        else
            % usar B00V : B e V têm polaridade oposta ao last
            Bp = -last;
            V = Bp;    % B e V mesma polaridade
            block = [Bp 0 0 V];
            % atualiza contagem: Bp e V são dois pulsos -> +2
            count_nonzero = count_nonzero + 2;
            last = V;
        end
        HDB3_sym = [HDB3_sym block];
        k = k + 4;
    else
        if B(k)==1
            last = -last;
            HDB3_sym = [HDB3_sym last];
            count_nonzero = count_nonzero + 1;
        else
            HDB3_sym = [HDB3_sym 0];
        end
        k = k + 1;
    end
end
HDB3 = shape(HDB3_sym);

%% --- Ajuste AMI/Pseudoternária originais gerados acima (já OK) ---

%% --- Para comparação: padronizar comprimentos (usar mínimo) ---
L = min([length(AMI), length(PSEUD), length(B8ZS), length(HDB3)]);
```

```
AMI = AMI(1:L);
PSEUD = PSEUD(1:L);
B8ZS = B8ZS(1:L);
HDB3 = HDB3(1:L);
%% --- Estimativa de PSD (Welch) ---
Fs = Ts;                    % frequência de amostragem arbitrária
(samples/símbolo)
nfft = 4096;
window = 1024;
noverlap = 512;

[Px_AMI, f]   = pwelch(AMI,  window, noverlap, nfft, Fs);
[Px_PSEUD, ~] = pwelch(PSEUD,window, noverlap, nfft, Fs);
[Px_B8ZS, ~]  = pwelch(B8ZS, window, noverlap, nfft, Fs);
[Px_HDB3, ~]  = pwelch(HDB3, window, noverlap, nfft, Fs);

%% --- Plots PSD (linear e dB) ---
figure('Position',[100 100 900 600]);
subplot(2,1,1);
plot(f, Px_AMI, 'r','LineWidth',1.2); hold on;
plot(f, Px_PSEUD, 'b','LineWidth',1.2);
plot(f, Px_B8ZS, 'g','LineWidth',1.2);
plot(f, Px_HDB3, 'm','LineWidth',1.2);
legend('AMI','Pseudoternária','B8ZS','HDB3');
xlabel('Frequência (Hz)'); ylabel('PSD (linear)');
title('PSDs (Welch) - linear');
grid on;

subplot(2,1,2);
plot(f, 10*log10(Px_AMI+eps), 'r','LineWidth',1.2); hold on;
plot(f, 10*log10(Px_PSEUD+eps),'b','LineWidth',1.2);
plot(f, 10*log10(Px_B8ZS+eps), 'g','LineWidth',1.2);
plot(f, 10*log10(Px_HDB3+eps), 'm','LineWidth',1.2);
legend('AMI','Pseudoternária','B8ZS','HDB3');
xlabel('Frequência (Hz)'); ylabel('PSD (dB/Hz)');
title('PSDs (Welch) - dB');
grid on;
```

(código gerado com auxílio de IA)

4.2 - output:

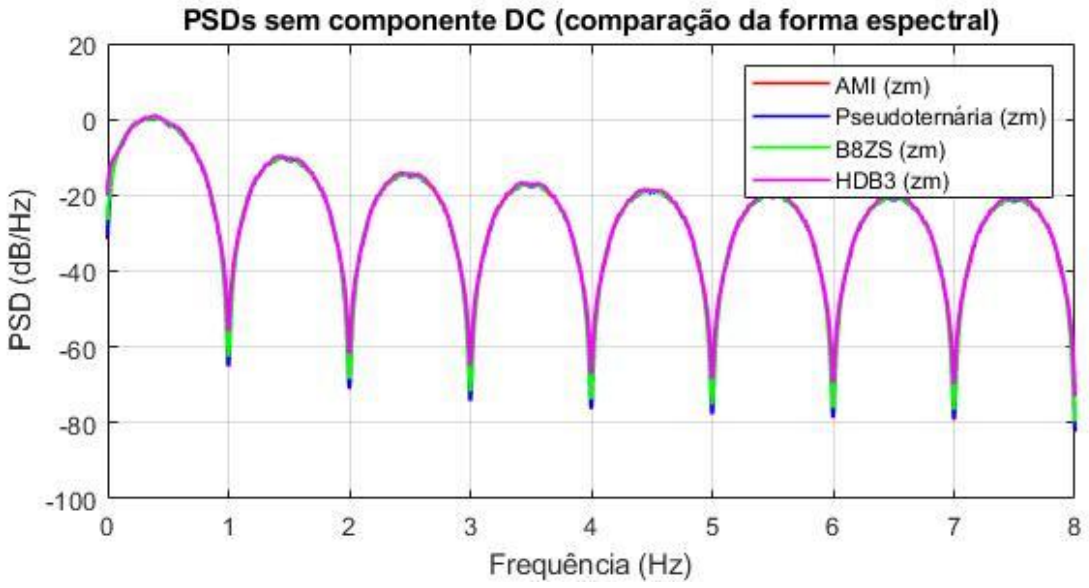


4.3 - Código para a comparação

```
%% PSDs sem componente DC (removendo média) para comparar forma espectral
x_AMI_z  = AMI  - mean(AMI);
x_PSE_z  = PSEUD- mean(PSEUD);
x_B8ZS_z = B8ZS - mean(B8ZS);
x_HDB3_z = HDB3 - mean(HDB3);

[PxA_z, ~] = pwelch(x_AMI_z,  window, noverlap, nfft, Fs);
[PxP_z, ~] = pwelch(x_PSE_z,  window, noverlap, nfft, Fs);
```

```
[PxB_z, ~] = pwelch(x_B8ZS_z, window, noverlap, nfft, Fs);
[PxH_z, ~] = pwelch(x_HDB3_z, window, noverlap, nfft, Fs);

figure;
plot(f,10*log10(PxA_z+eps),'r','LineWidth',1.2); hold on;
plot(f,10*log10(PxP_z+eps),'b','LineWidth',1.2);
plot(f,10*log10(PxB_z+eps),'g','LineWidth',1.2);
plot(f,10*log10(PxH_z+eps),'m','LineWidth',1.2);
legend('AMI (zm)','Pseudoternária (zm)','B8ZS (zm)','HDB3 (zm)');
xlabel('Frequência (Hz)'); ylabel('PSD (dB/Hz)');
title('PSDs sem componente DC (comparação da forma espectral)');
grid on;
```

4.4 - output:



4.5 - Observações:

B8ZS:

Introduz padrões deliberados (violations) em blocos de 8 zeros. Estes impulsos artificiais adicionam energia de alta frequência localmente porque inserem transições rápidas. Na PSD aparece maior conteúdo em bandas médias-altas comparado com AMI puro. B8ZS preserva média zero global (nenhuma DC significativa) porque as substituições são balanceadas.

HDB3:

Substituições a cada 4 zeros adicionam pulses B/V de forma a manter balanceamento de polaridade. Isso cria mais transições do que AMI puro e desloca energia para frequências mais altas, mas de modo menos "agressivo" que B8ZS (B8ZS é mais raro porém com bloco maior). HDB3 também mantém média próxima de zero, reduzindo pico DC.

Conclusão:

A análise prática das PSDs esteve em conformidade com a teoria, evidenciando que cada técnica de codificação possui vantagens e limitações em termos de eficiência espectral e robustez para transmissão de dados digitais.

Bibliografia:

- Slides da disciplina
- https://web.stanford.edu/class/ee179/lectures/notes14.pdf
- https://www.univasf.edu.br/~edmar.nascimento/pcom/pcom_aula18_21.pdf
- Material sobre cálculo de PSD de códigos de linha - prof. Edmar Nascimento