UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

Wagner Spinato Chittó

3º TRABALHO PRÁTICO

Quantização

Santa Maria, RS
05/09/2025

Objetivo:

Simulação dos processos de amostragem e quantização. Observação dos efeitos dos níveis de quantização, faixa de excursão do sinal, razão entre sinal e ruído de quantização (SQNR).

Fundamentação:

- Arquivo de simulação “amostra_quant.m”:

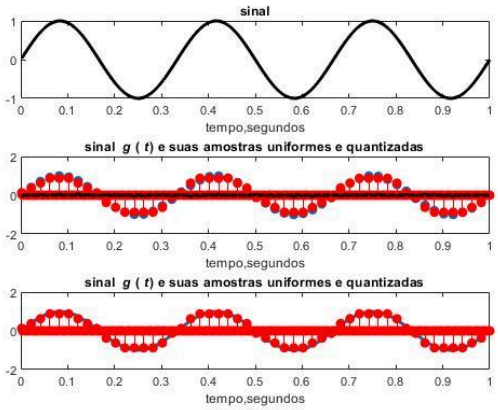
```
% amostra_quant.m
function [s_out,sq_out]=amostra_quant(sig_in,td,ts, amp, L)
    if(rem(ts/td,1)==0)
      nfac=round(ts/td);
      s_out=downsample(sig_in,nfac);
      sq_out=quantizacao(s_out, amp, L);
      s_out=upsample(s_out,nfac);
      sq_out=upsample(sq_out,nfac);
    else
      warning('Erro! ts/td nao eh um inteiro!');
      s_out=[];sq_out=[];
    end
end

function x=quantizacao (sinal,amp,n)
% quantizacao
Delta=2*amp/n; % quantizacao uniforme
x = sinal + amp; % somar nivel CC igual a amp
q = floor(x/Delta); % dividir em intervalos iguais a D
x = Delta/2 + Delta*q - amp; % quantizar e remover nivel CC
end

% (uniquan.m)
function [q_out,Delta,SQNR]=uniquan(sig_in,L)
% L - numero de niveis de quantizacao uniforme
% sig_in - vetor para sinal de entrada
sig_pmax = max(sig_in); % pico positivo
sig_nmax = min(sig_in); % pico negativo
Delta=(sig_pmax - sig_nmax)/L; % intervalo de quantizacao
q_level = sig_nmax+Delta/2:Delta:sig_pmax-Delta/2; % define Q niveis
L_sig = length(sig_in); % comprimento do sinal
sigp = (sig_in-sig_nmax)/Delta+1/2; % converte a faixa de 1/2 a L+1/2
qindex=round(sigp); % arredonda a 1,2,...,L niveis
qindex=min(qindex,L); % elimina L+1, se houver
q_out=q_level(index); % usa vetor index para gerar saida
SQNR=20*log10(norm(sig_in)/norm(sig_in-q_out)); % valor da SQNR
end
```

- Arquivo de simulação “conversao_ad.m”:

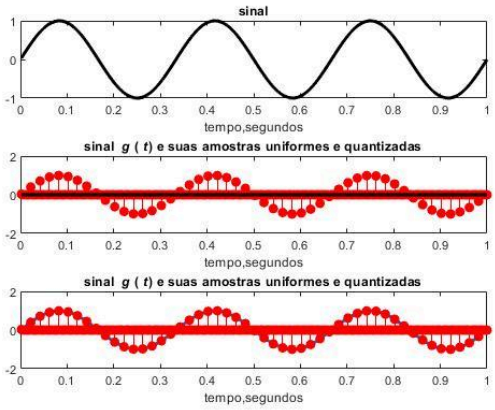
```
clear all;close all;
  % Exemplo de amostragem e quantizacao
td=0.002; % intervalo entre os pontos do sinal "analogico"
t=td:td:1; % intervalo de 1 segundo
xsig=sin(2*3*pi*t); % seno de 3Hz;
Lsig=length(xsig);
ts=0.02; % taxa de amostragem 50 Hz
fator=ts/td;
figure(1);
subplot(311); sfig1a=plot(t,xsig,'k');
set(sfig1a,'LineWidth',2);
xlabel('tempo,segundos');
title('sinal');
amp = 1;
%amp = max(xsig);
bits = 3;  % numero de bits da conversao analogico/digital
n=2^bits; % numero de niveis
  % envia o sinal a um amostrador e quantizador
[s_out,sq_out] = amostra_quant(xsig,td,ts,amp,n);
subplot(312);
stem(t,s_out,'filled');
hold on;
stem(t,sq_out,'r', 'filled');
  % ruido de quantizacao
sr = s_out - sq_out; % erro
emq = sr*sr'/length(sr); % erro medio quadratico
SQNR=20*log10(norm(s_out)/norm(sr)); % valor da SQNR
sfig1b = plot(t,sr,'k');
set(sfig1b,'LineWidth',2);
axis([0 1 -2 2]);
xlabel('tempo,segundos');
title('sinal {\it g} ({\it t}) e suas amostras uniformes e quantizadas');
  % diferenca entre sinal e ruido
subplot(313);
sfig1c=plot(t,xsig);
hold on;
stem(t,sq_out,'r','filled');
set(sfig1c,'LineWidth',2);
axis([0 1 -2 2]);
xlabel('tempo,segundos');
title('sinal {\it g} ({\it t}) e suas amostras uniformes e quantizadas');
```

Procedimento:

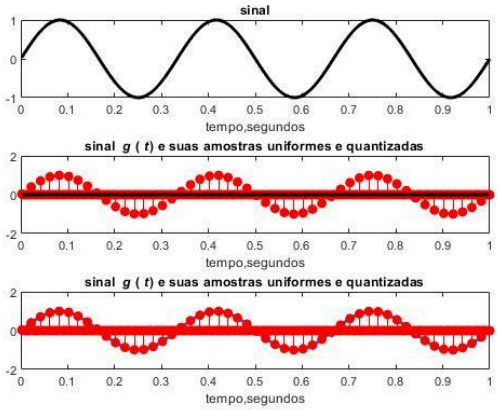
2 - Gráficos para algumas amostras de bits:



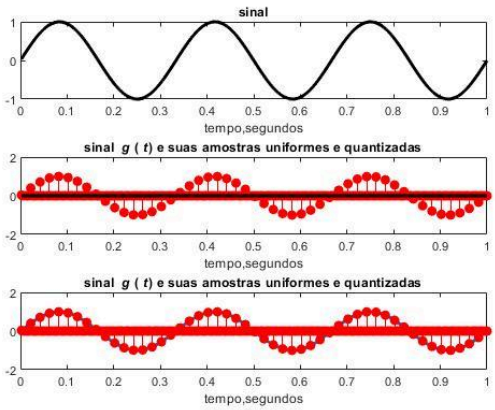
3 bits



8 bits



13 bits



16 bits

- Código para interpolação linear:

```
clear all; close all;

% parâmetros de tempo e sinal
td = 0.002;              % intervalo entre pontos do sinal
t = td:td:1;             % duração
xsig = sin(2*pi*3*t);    % seno de 3 Hz
amp = 1;                 % amplitude de referência
ts = 0.02;               % intervalo de amostragem - 50 Hz

% variação do número de bits
bits_vec = 3:16;
SQNR_vec = zeros(size(bits_vec));
for k = 1:length(bits_vec)
    bits = bits_vec(k);
    n = 2^bits; % níveis de quantização
    % amostragem + quantização
    [s_out, sq_out] = amostra_quant(xsig, td, ts, amp, n);
    % erro de quantização
    sr = s_out - sq_out;
    SQNR_vec(k) = 20*log10(norm(s_out)/norm(sr));
end

% ajuste linear SQNR = a*bits + b
p = polyfit(bits_vec, SQNR_vec, 1);
a = p(1); b = p(2);

% gráfico
figure;
plot(bits_vec, SQNR_vec, 'ro', 'MarkerFaceColor','r');
hold on;
plot(bits_vec, polyval(p, bits_vec), 'b-','LineWidth',1.5);
xlabel('Número de bits');
ylabel('SQNR (dB)');
title('SQNR em função do número de bits');
grid on;
legend('Simulação','Ajuste linear');

fprintf('Reta ajustada: SQNR ≈ %.3f*bits + %.3f dB\n', a, b);
```

2.1 - Obentção dos valores para uma função *SQNR(bits) = b + a.bits*

Através do código abaixo obteve-se os seguintes valores para os bits:

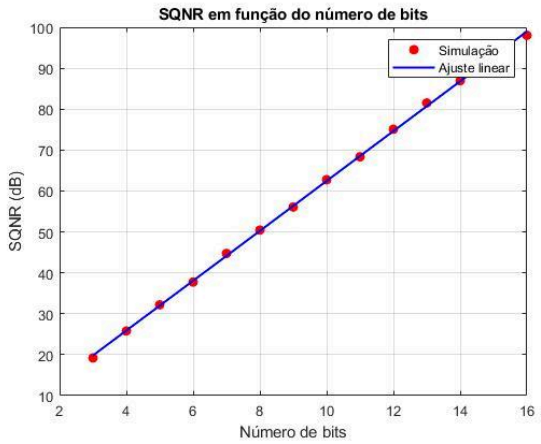
bits	SQNR_dB_pred
3	10.055
4	16.139
5	22.222
6	28.305
7	34.389
8	40.472
9	46.555
10	52.639
11	58.722
12	64.805
13	70.889
14	76.972
15	83.055
16	89.139

Amplitude = 1

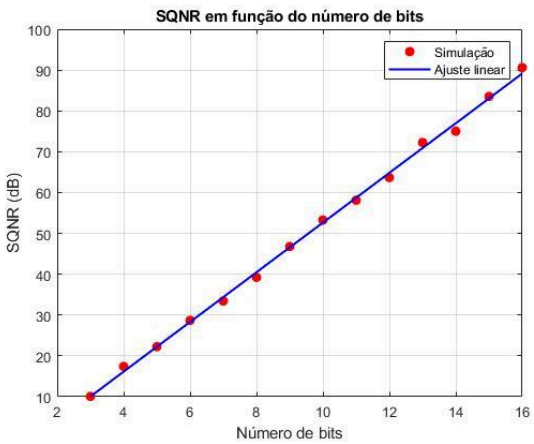
bits	SQNR_dB_pred
3	19.799
4	25.891
5	31.983
6	38.075
7	44.167
8	50.259
9	56.351
10	62.443
11	68.535
12	74.627
13	80.719
14	86.811
15	92.903
16	98.995

Amplitude = 3

- Gráficos da interpolação linear:



Amplitude = 1



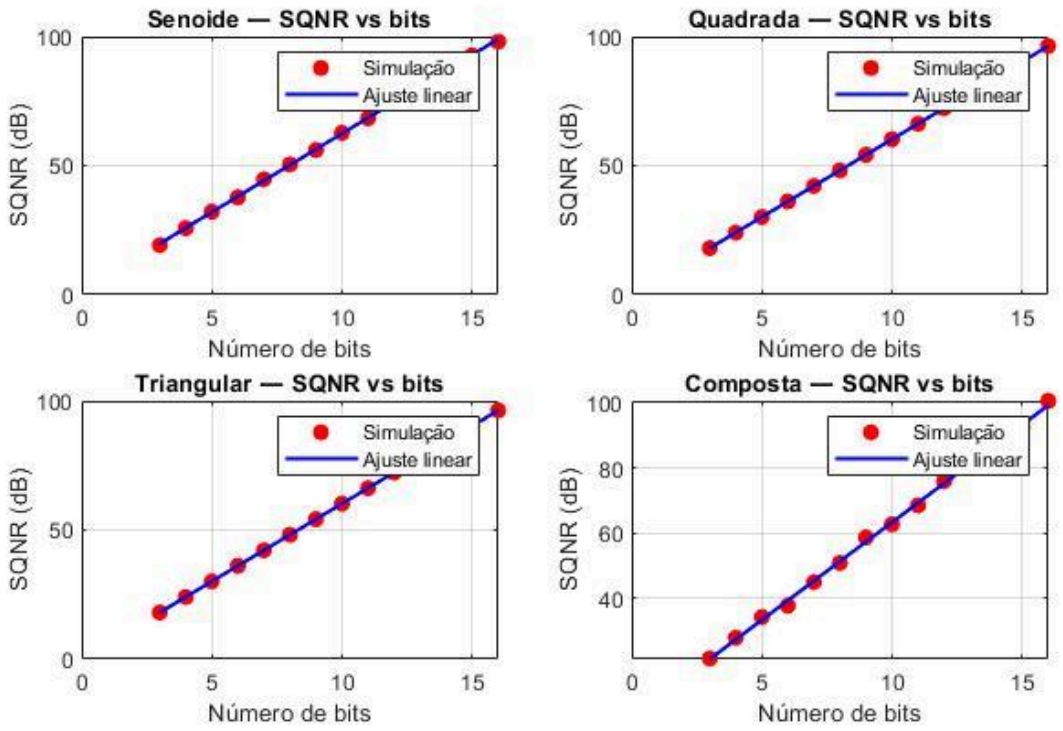
Amplitude = 3

3.1 - Programa para obtenção dos gráficos senoidal, quadrada, triangular e composta:

```
td = 0.002;
t = td:td:1;
amp = 1;
ts = 0.02;
bits_vec = 3:16;

% Definição dos sinais
xsig1 = sin(2*pi*3*t);                          % senoide
xsig2 = square(2*pi*3*t);                       % quadrada
xsig3 = sawtooth(2*pi*3*t, 0.5);                % triangular
xsig4 = sin(2*pi*3*t) + 0.5*sin(2*pi*7*t);      % composta

signals = {xsig1, xsig2, xsig3, xsig4};
labels  = {'Senoide','Quadrada','Triangular','Composta'};

figure;
for s = 1:length(signals)
    xsig = signals{s};
    SQNR_vec = zeros(size(bits_vec));

    for k = 1:length(bits_vec)
        bits = bits_vec(k);
        n = 2^bits;

        [s_out, sq_out] = amostra_quant(xsig, td, ts, amp, n);
        sr = s_out - sq_out;
        SQNR_vec(k) = 20*log10(norm(s_out)/norm(sr));
    end

    % ajuste linear
    p = polyfit(bits_vec, SQNR_vec, 1);
    a = p(1); b = p(2);

    % gráfico
    subplot(2,2,s)
    plot(bits_vec, SQNR_vec, 'ro','MarkerFaceColor','r'); hold on;
    plot(bits_vec, polyval(p,bits_vec),'b-','LineWidth',1.5);
    xlabel('Número de bits'); ylabel('SQNR (dB)');
    title([labels{s} ' – SQNR vs bits']);
    grid on;
    legend('Simulação','Ajuste linear');
```

```
    % imprimir equação da reta
    fprintf('\n%s: SQNR ≈ %.3f*bits + %.3f dB\n', labels{s}, a, b);

    % tabela bits vs SQNR previsto
    SQNR_hat = b + a*bits_vec;
    T = table(bits_vec.', SQNR_vec.', SQNR_hat.', ...
              'VariableNames', {'bits','SQNR_sim','SQNR_pred'});
    disp(T)
end
```

3.1 Gráficos:



Conclusão:

Obteve-se o valor para a interpolação linear:

6.092*bits + 1.522 dB

Em acordo com a teoria, obteve-se um valor próximo à 6.02 para **a**; e **b** próximo à 1.76;

Bibliografia:

- Algoritmo amostra_quant.m: https://ead06.proj.ufsm.br/pluginfile.php/5461154/mod_resource/content/1/amostra_quant.m
- Algoritmo conversao_ad.m: https://ead06.proj.ufsm.br/pluginfile.php/5461153/mod_resource/content/1/conversao_ad.m