Universidade de Brasília

Departamento de Engenharia Elétrica

Tópicos em Engenharia - Processamento de Sinais Biomédicos

Lista de Exercícios 1

Autores:

Caio Luiz Candeias Flôres 190134283

Felipe Carneiro da Motta 200017616

João Pedro Daher Aranha 190109742

Brasília

24 de Junho de 2022

Conteúdo

1 Exercícios

1.1 Exercício 1.1:

A lowpass filter is desired with a cutoff frequency of 10 Hz. This filter should attenuate a 100-Hz signal by a factor of at least 78. What should be the order of this filter?

É sabido que a frequência de corte é determinada quando o ganho do filtro decai aproximadamente 3 dB em relação ao ganho de banda passante. Para o caso do filtro passa-baixa com frequência de corte de 10 Hz e com atenuação de pelo menos 78, é necessário que tenha, pelo menos, 4 polos. Tendo em vista que um polo atenua o sinal em 20 dB/década, com 4 polos ocorre uma atenuação em 80 dB/década do sinal, o que é mais que o mínimo de atenuação necessária.

1.2 Exercício 1.5:

Generate a discrete 2-Hz sine wave using MATLAB as in Example 1.2. Use sample intervals of 0.05, 0.01, and 0.001 s and again use enough points to make the sine wave 1-s long, that is, the total time, TT, should be 1 s. Plot with lines connecting the points and on the same plot, the individual points superimposed on the curves as in Example 1.2. Plot the sine waves using the three sample intervals separately, but use subplot to keep the three plots together. Note that even the plot with very few points looks sinusoidal. Chapter 3 discusses the minimum number of points required to accurately represent a signal.

O código no MATLAB:

```
close all; clear all; clc;

intervals = [0.05,0.01,0.001]; % intervalos
num_plots = length(intervals); % qtde de plots
TT = 1; % tempo total
f = 2; % frequencia do sinal senoidal
for i = 1:num_plots
  subplot(num_plots, 1, i);
  ts = intervals(i); % periodo de amostragem
  t = 0:ts:TT; % vetor de tempo
```

```
11    x = sin(2*pi*f.*t);
12    plot(t, x); % plotagem da linha
13    hold on;
14    plot(t, x, '.k'); % plotagem dos pontos
15    xlabel('Tempo (s)');
16    ylabel(sprintf('Amplitude do sinal'));
17    legend('Linha', 'Pontos', 'Location', 'best'); %
          melhor localização
18      title(sprintf('Ts = %s', string(ts)));
19  end
20
21  sgtitle('Sinal senoidal de 2 Hz com diferentes Ts');
22  saveas(gcf,'semmlow_1_5.png');
```

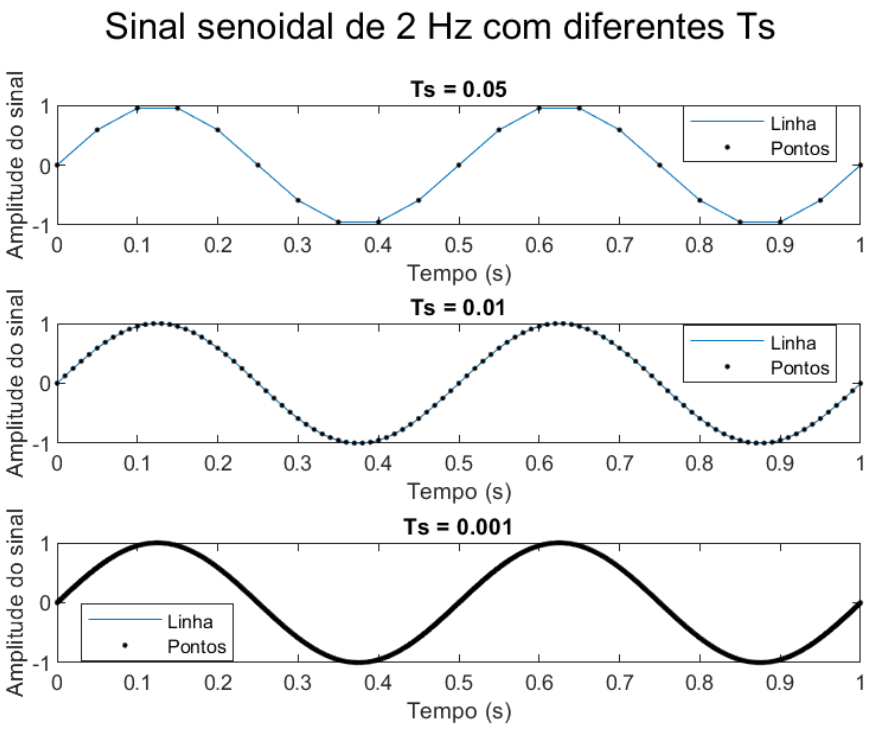

Figura 1: Gráfico do Exercício 1.5

1.3 Exercício 1.6:

Write a MATLAB problem to test Equation 1.6 through simulation. Generate a 4-Hz, 1000-point sine wave as in Example 1.4 assuming a sample interval of Ts = 0.002. Use quantization.m to digitize it using 4-, 8-, 12-, and 16-bit ADC. Then, as in Example 1.4, subtract the original signal from the quantized signal to find the error signal. The amplitude of the error signal should be equal to the quantization level, q in Equation 1.6. Use MATLAB's max and min functions to find this amplitude and compare it with the value predicted by Equation 1.6. Put this code in a for-loop and repeat the evaluations for the four different bit levels requested. Be sure to display the results to at least four decimal places to make an accurate comparison. [Hint: Most of the code needed for this problem will be similar to that in Example 1.4.]

O código no MATLAB:

```
clc; clear all; close all;

% a rotina "quantization.m" deve ser carregada

f1 = 4; % frequencia da senoide: 4 Hz
N = 1e3; % numero de pontos do vetor de tempo
ts = 0.002; % periodo de amostragem

t1 = (0:N-1)*ts; % vetor de tempo de 1e3 pontos;

% definindo numero de bits
% quantizacao para 4, 8, 12 e 16 bits
for bits=4:4:16
    signal_in = sin(2*pi*f1*t1); % senoide de entrada
    signal_out = quantization(signal_in, bits); %
       quantizando
    noise_signal = signal_out - signal_in;

    % calculo do erro experimental/amplitude do sinal
       de erro
    q_exp = max(noise_signal) - min(noise_signal);

    % calculo do nivel de quantizacao
    q = 1/(2^bits-1);
```

```
    out = sprintf('%2d bits: Valor experimental de q =
        %5e | Valor teórico de q = %5e', bits, q_exp, q)
        ;

    % saida formatada
    disp(out)
end
```

```
 4 bits: Valor experimental de q = 6.120598e-02 | Valor teórico de q = 6.666667e-02
 8 bits: Valor experimental de q = 3.825425e-03 | Valor teórico de q = 3.921569e-03
12 bits: Valor experimental de q = 2.370130e-04 | Valor teórico de q = 2.442002e-04
16 bits: Valor experimental de q = 1.511926e-05 | Valor teórico de q = 1.525902e-05
```

Figura 2: Resultado do código do Exercício 1.6

Por meio da figura 2, pode ser visto que há uma pequena diferença entre os valores de q (nível de quantização) e q_exp (valor da amplitude do sinal de erro). Essa diferença é justificada pelo fato de que o uso de um número determinado de bits (no caso, 4, 8, 12 e 16 bits) seleciona uma quantidade finita de amostras no tempo, o que faz com que quanto maior seja a quantidade de bits, mais próximo o valor de q_exp fica de q. Em um cenário ideal, o valor de q_exp seria, no máximo, igual ao valor de q.

$$\lim_{n\to\infty} q_{exp}(n) = q$$

1.4 Exercício 1.7:

Extend the code in Example 1.4 to include quantization levels involving 4, 8, 10, and 12 bits. Compare the theoretical and simulated noise for these four different quantization levels. Present the output in a format with sufficient resolution to illustrate the small differences between the simulated and theoretical noise.

Repetindo o código do exercício 1.6 com a devida alteração do vetor de iteração do loop for para $n_{bits} = [4, 8, 10, 12]$, obteve-se o seguinte resultado:

```
clc; clear all; close all;

% a rotina "quantization.m" deve ser carregada

```

```
f1 = 4; % frequencia da senoide: 4 Hz
N = 1e3; % numero de pontos do vetor de tempo
ts = 0.002; % periodo de amostragem

t1 = (0:N-1)*ts; % vetor de tempo de 1e3 pontos;

n_bits = [4, 8, 10, 12];
% quantizacao para 4, 8, 10 e 12 bits
for bits=n_bits
    signal_in = sin(2*pi*f1*t1); % senoide de entrada
    signal_out = quantization(signal_in, bits); %
       quantizando
    noise_signal = signal_out - signal_in;

    % calculo do erro experimental/amplitude do sinal
       de erro
    q_exp = max(noise_signal) - min(noise_signal);

    % calculo do nivel de quantizacao
    q = 1/(2^bits-1);
    out = sprintf('%2d bits: Valor experimental de q =
       %5e | Valor teórico de q = %5e', bits, q_exp, q)
       ;

    % saida formatada
    disp(out)
end
```

```
 4 bits: Valor experimental de q = 6.120598e-02 | Valor teórico de q = 6.666667e-02
 8 bits: Valor experimental de q = 3.825425e-03 | Valor teórico de q = 3.921569e-03
10 bits: Valor experimental de q = 9.390508e-04 | Valor teórico de q = 9.775171e-04
12 bits: Valor experimental de q = 2.370130e-04 | Valor teórico de q = 2.442002e-04
```

Figura 3: Resultado do código do Exercício 1.7

1.5 Exercício 1.8:

Write a MATLAB program to generate 1 s of a 5-Hz sine wave in a 1000-point array. (Use Equation 1.4 to determine the equivalent Ts.) Plot the sine wave. Simulate the sampling of this waveform at 7 Hz by plotting a point (such as an "*") at intervals of Ts = 1/fs = 1/7 s. (Use a for-loop to generate

the sample time, Ts, insert it into the equation for the 5-Hz sine wave, and plot the resulting point.) Aliasing predicts that these sampled points should fall on a 2-Hz (7–5 Hz) sine wave. Can you find a 2-Hz sine wave that includes all seven points? [Hint: The sine wave could be phase shifted by 180°.]

A onda senoidal de frequência de 2 Hz a qual inclui todos os 7 pontos foi a segiunte:

```
clear all;
close all;

f=5; %5Hz
Tt= 1;%1 segundo

%Representação 1000 pontos:
N=1000; %N=Tt/Ts
Ts1 = Tt/N;
t1=(0:N-1)*Ts1;
sig1=sin(2*pi*f*t1);

%Representação subamostrada
fs2=7; %Amostragem 7Hz
Ts2=1/fs2;
t2=0:Ts2:Tt;
sig2=sin(2*pi*f*t2);

%Plotando:
plot(t1,sig1,'b');grid;
xlabel('Tempo (s)');
ylabel('Amplitude do sinal')
hold on;
plot(t2,sig2,'r*');

%Sinal "recuperado"
t3=linspace(0,Tt);
f2=fs2-f;
sig3=-sin(2*pi*f2*t3);
plot(t3,sig3,'g');

legend('sinal original','sinal subamostrado','sinal "
```

```
    recuperado"');
title('Efeito de Aliasing');
```

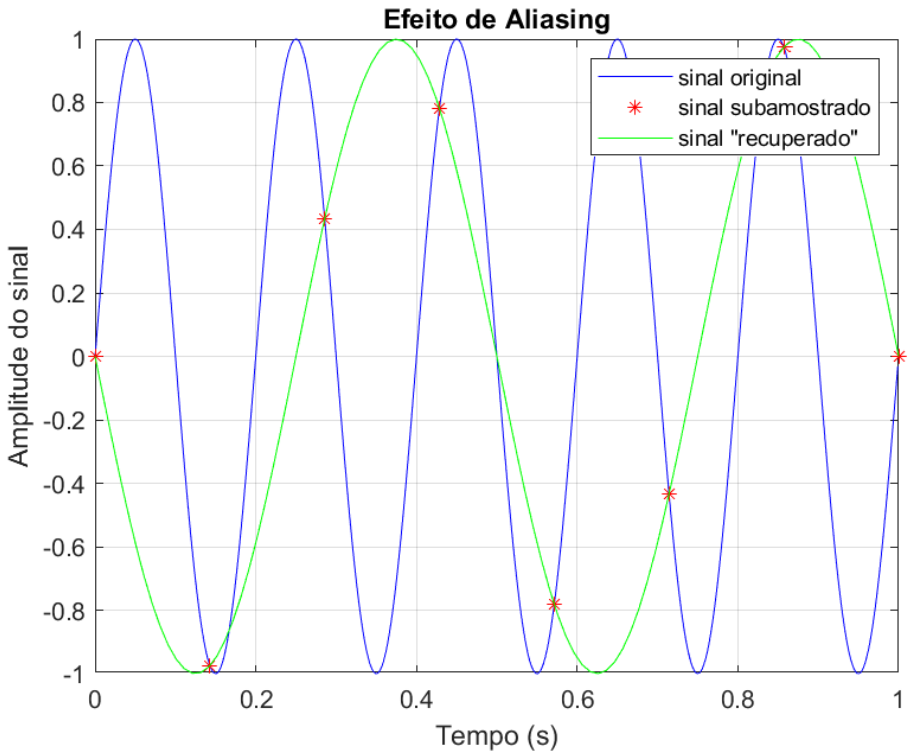

Figura 4: O resultado do código do Exercício 1.8

1.6 Exercício 1.9:

Repeat Problem 1.8 using a simulated sample interval of 9 Hz ($Ts = \frac{1}{fs} = \frac{1}{9}$ s) and find the appropriate sine wave produced by aliasing.

Apenas alterando a variável fs2 para 9 Hz e realizando as devidas alterações no código do problema 1.8, obteve-se o seguinte resultado:

```
clear all;
close all;

f=5; %5Hz
Tt= 1;%1 segundo
```

```
6
7  %Representação 1000 pontos:
8  N=1000; %N=Tt/Ts
9  Ts1 = Tt/N;
10 t1=(0:N-1)*Ts1;
11 sig1=sin(2*pi*f*t1);
12
13 %Representação subamostrada
14 fs2=9; %Amostragem 7Hz
15 Ts2=1/fs2;
16 t2=0:Ts2:Tt;
17 sig2=sin(2*pi*f*t2);
18
19 %Plotando:
20 plot(t1,sig1,'b');grid;
21 xlabel('Tempo (s)');
22 ylabel('Amplitude do sinal')
23 hold on;
24 plot(t2,sig2,'r*');
25
26 %Sinal "recuperado"
27 t3=linspace(0,Tt);
28 f2=fs2-f;
29 sig3=-sin(2*pi*f2*t3);
30 plot(t3,sig3,'g');
31
32 legend('sinal original','sinal subamostrado','sinal "
     recuperado"');
33 title('Efeito de Aliasing');
```

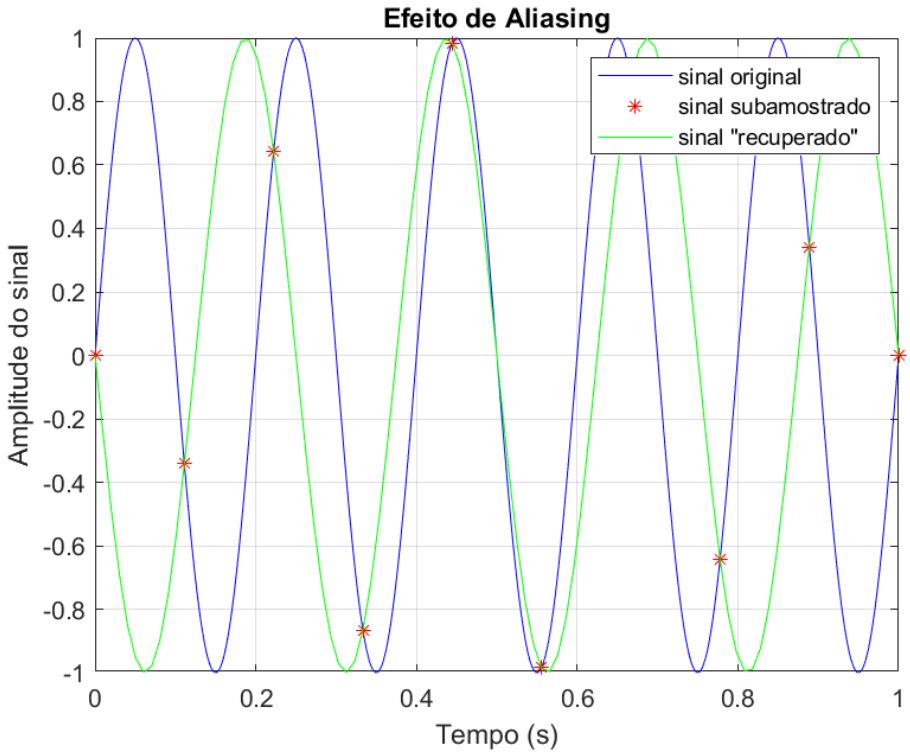

Figura 5: O resultado do código do Exercício 1.9