Bibliotécas usadas: Plots, DSP, FFTW, WAV, SampledSignals, FixedPointNumbers,Statistcs

Experiência 1

Gabriel Tavares 10773801

Guilhemre Reis 10773700

Alexandre Alcoforado 9381426

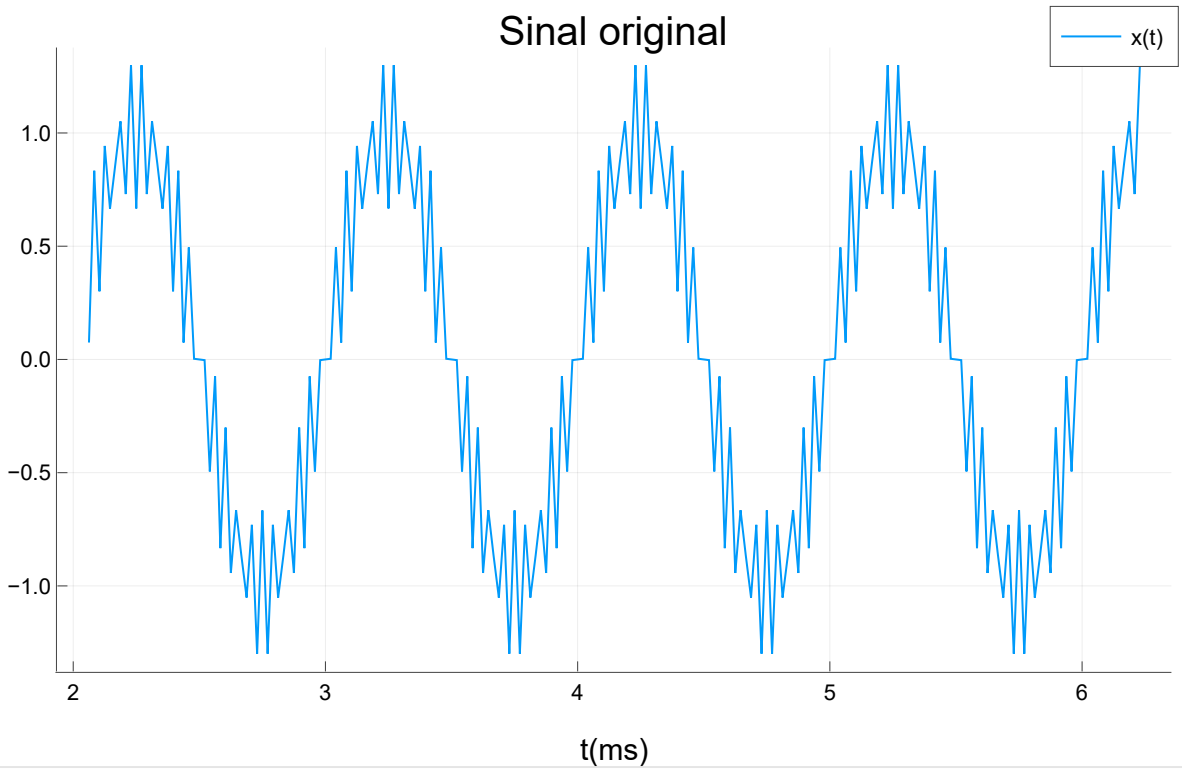
Redução de taxa de amostragem

Sinal enunciado

Usaremos o sinal $x(t) = sin(1000.2\pi t) - 1/3sin(21000.2\pi t)$ para realizar o experimento. Inicialmente a frequência de amostragem de 48kHz

```
24000.0
```

```
begin
    fa1 = 48e3 #48kHz
    fa2 = 24e3 #24kHz
end
```



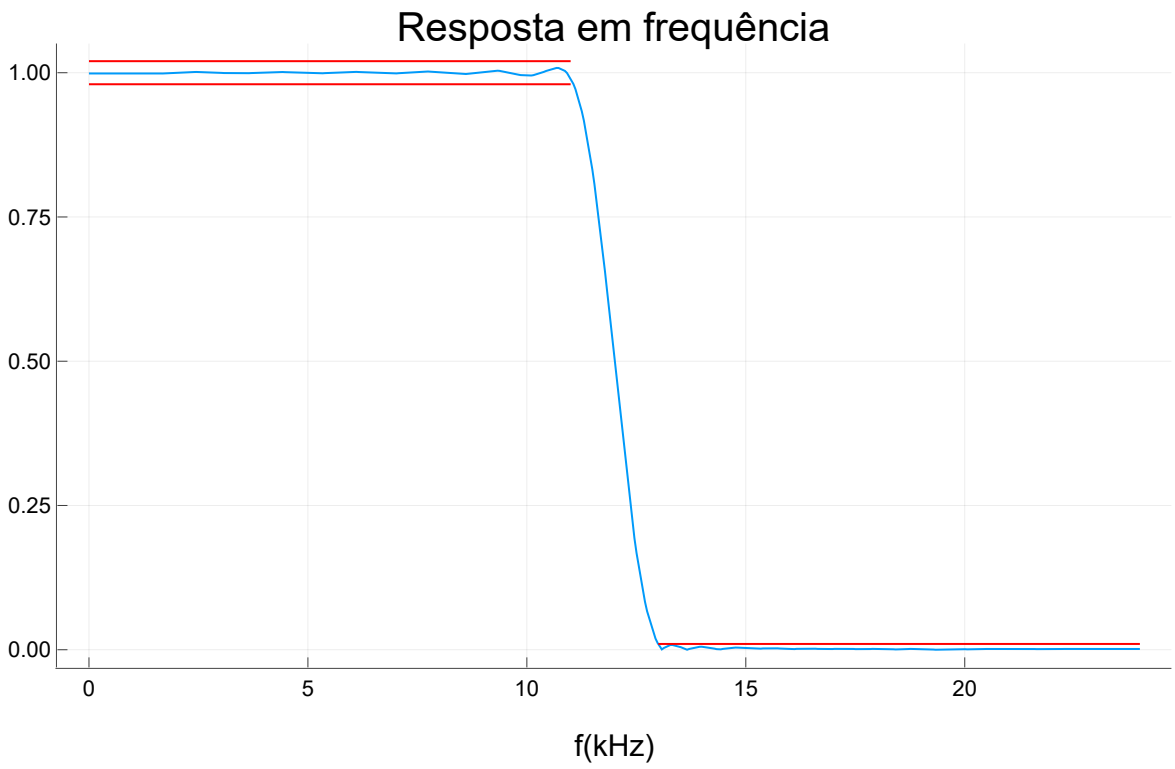

```
begin
    range_plot = 100:300
    plot_range = range_plot
    t = 0:1/fa1:0.05
    t_ms = t*1000
    x_48 = sin.(1e3*2π * t) - 1/3*sin.(21e3*2π * t)
    plot(t[range_plot]*1000,x_48[range_plot], label = "x(t)")
    plot!(title = "Sinal original", xlabel = "t(ms)")
end
```

1) Projeto de filtro

Projeto de filtro com frequência de passagem de 11kHz, frequência de rejeição de 13kHz e tolerância de 0.01

```
[-0.0017444, 0.0, 0.00274191, 0.0, -0.00401654, 0.0, 0.00562212, 0.0, -0.00762953, 0.0, (
```

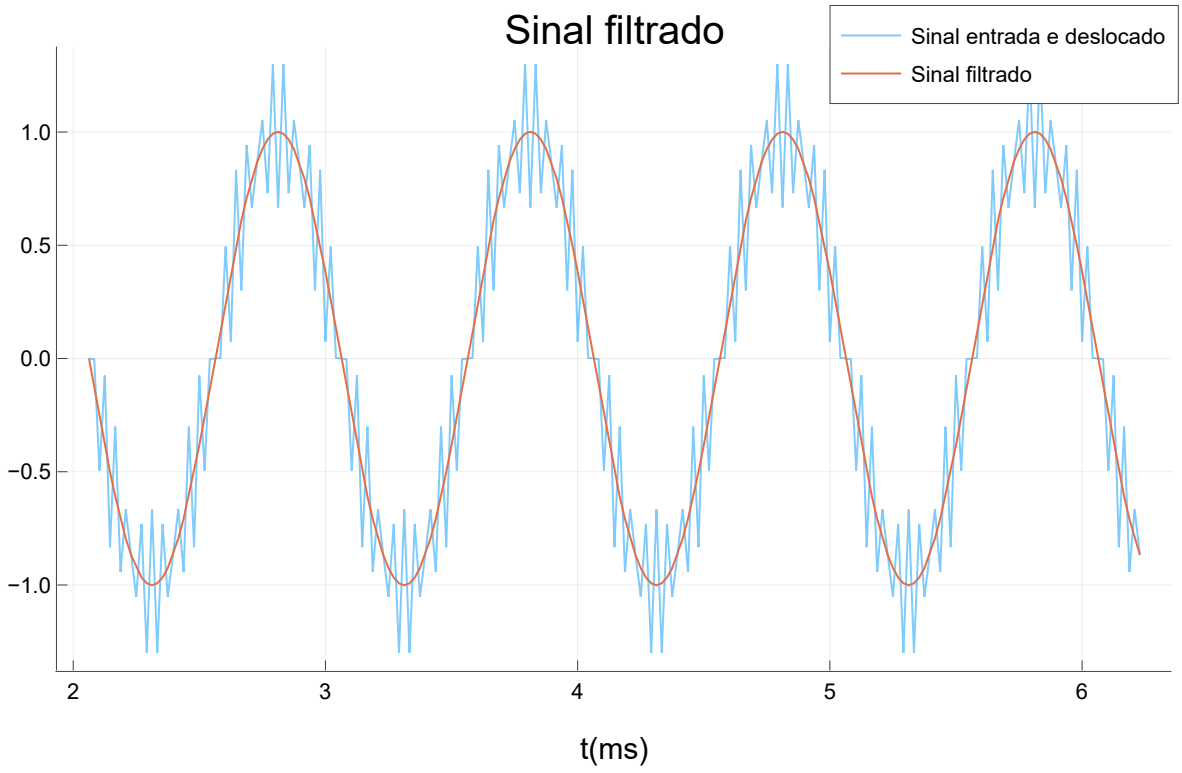
```
begin
    δp = 0.02
    δr = 0.01

    fp = 11e3
    fr = 13e3

    ωp = fp/fa1 *2π
    ωr = fr/fa1 * 2π

    pb_12k_kaiser = kaiser_filter_lowpass(δp, δr, ωp, ωr)
end
```



2) Filtragem do sinal

```
x_filtrado =
  [0.0, -5.17223e-6, -0.000862644, -0.000122218, -9.77365e-5, -0.000331743, -0.00134598, -
```

```
x_filtrado = filt(pb_12k_kaiser,x_48)
```

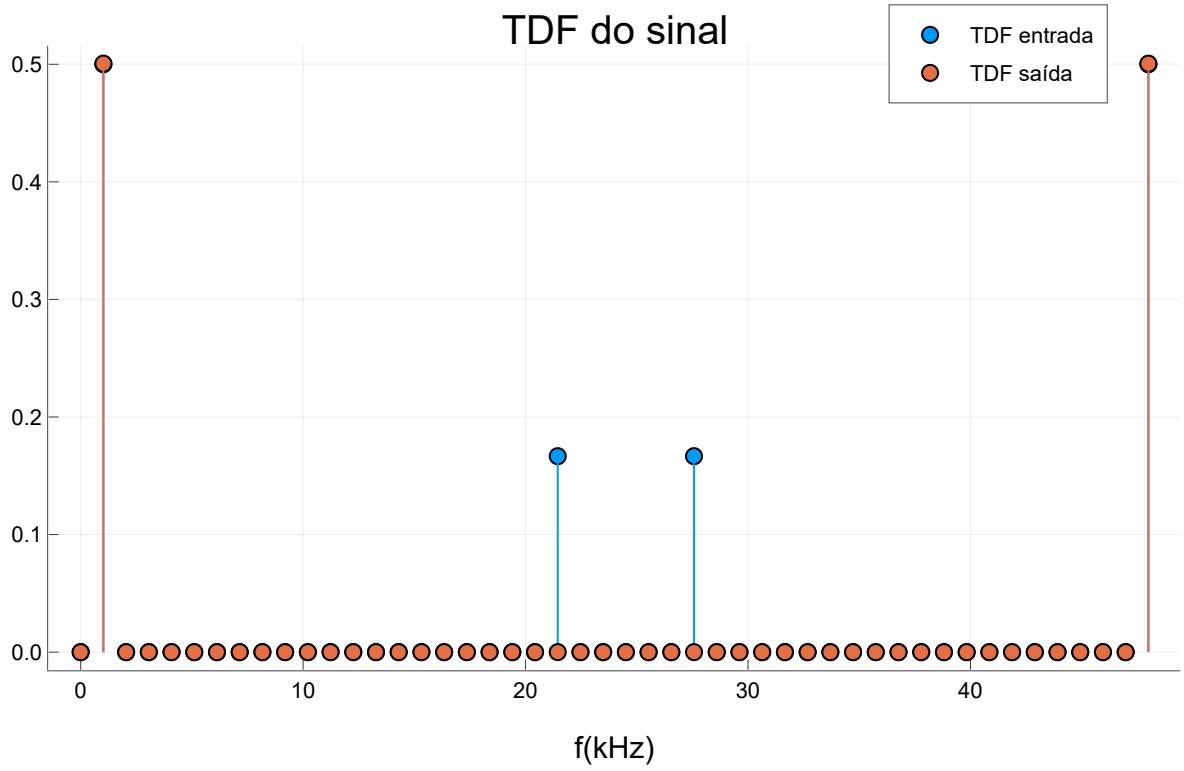


3) Calculando espectros e respostas do filtro

Nesta secção iremos comparar o ganho do filtro do sinal em coparação com a resposta em frequência, e a razão entre as TDFs dos sinais de entrada e saída

```
0.0:1.0212765957446808:48.0
```

```
begin
    X =  fft(x_48[1:48])
    X_saida = fft(x_filtrado[2*length(pb_12k_kaiser):2*length(pb_12k_kaiser)+48-1])

    fk = range(0,48,length = 48)
end
```




```
0.00016883144036456844
```

```
begin
    ω_1kHz = 1e3*2π/fa1
    ω_21kHz = 21e3*2π/fa1
    indice_1k_ω = findall(x->isapprox(x,ω_1kHz,atol = 0.003),ω)
    indice_21k_ω = findall(x->isapprox(x,ω_21kHz,atol = 0.003),ω)

    PB_1k_filtro = abs.(PB_12k_kaiser[indice_1k_ω])[]
    PB_21k_filtro = abs.(PB_12k_kaiser[indice_21k_ω])[]

    indice_1k_fk = findall(x-> isapprox(x,1,atol = 0.1),fk)
    indice_21k_fk = findall(x-> isapprox(x,21,atol = 0.5),fk)

    PB_1k_fft = abs(X_saida[indice_1k_fk][])/abs(X[indice_1k_fk][])
    PB_21k_fft =abs(X_saida[indice_21k_fk][])/abs(X[indice_21k_fk][])
end
```

Tabela de ganhos calculados pela razão entre a fft na entrada e saída do sinal; e ganho da resposta em frequência do sinal.

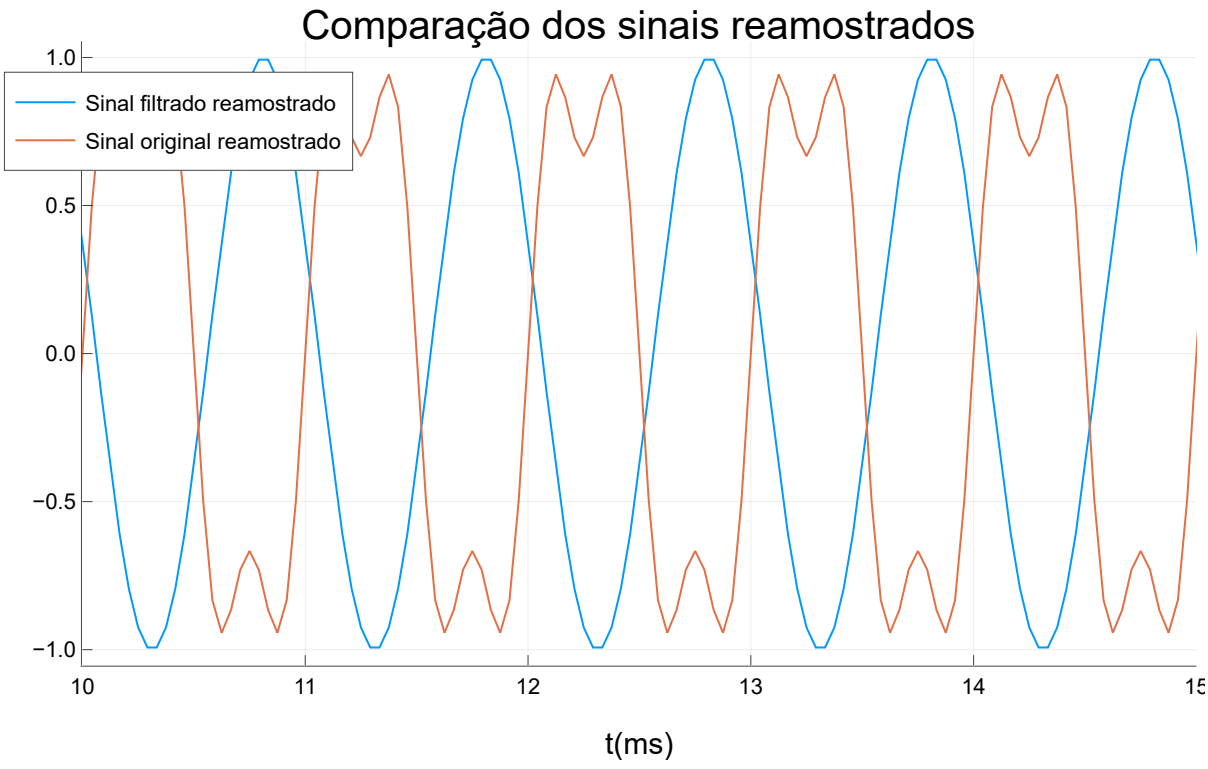
Frequência (kHz)	Ganho Teórico ($H(e^{j\omega})$)	Ganho Calculado (FFT)
1	1.0011	1.0011
21	0.0001	0.0002

4) Redução da taxa de amostragem

Nesta seção iremos reduzir a taxa de amostragem e comparar a redução do sinal filtrado e não filtrado

```
[0.0, -0.000862644, -9.77365e-5, -0.00134598, 0.000507544, -0.00177393, 0.00131696, -0.0
```

```
begin
    M = 2
    x_reamostrado = reduz_amostragem(x_filtrado,M)
end
```



Aumento da Taxa de Amostragem

Nesta etapa iremos fazer uma reamostragem de um sinal que foi amostrado a 44kHz para uma taxa L vezes maior.

Inicialmente usaremos o mesmo sinal $x(t) = sin(1000.2\pi t) - 1/3sin(21000.2\pi t)$ amostrado a 48kHz e reamostraremos ele a 144kHz.

$$144kHz = 48kHz * L = 48kHz * 3$$

```
144000
```

```
begin
    L = 3
    fa_48 = 48_000
    fa_144 = fa_48*L
end
```

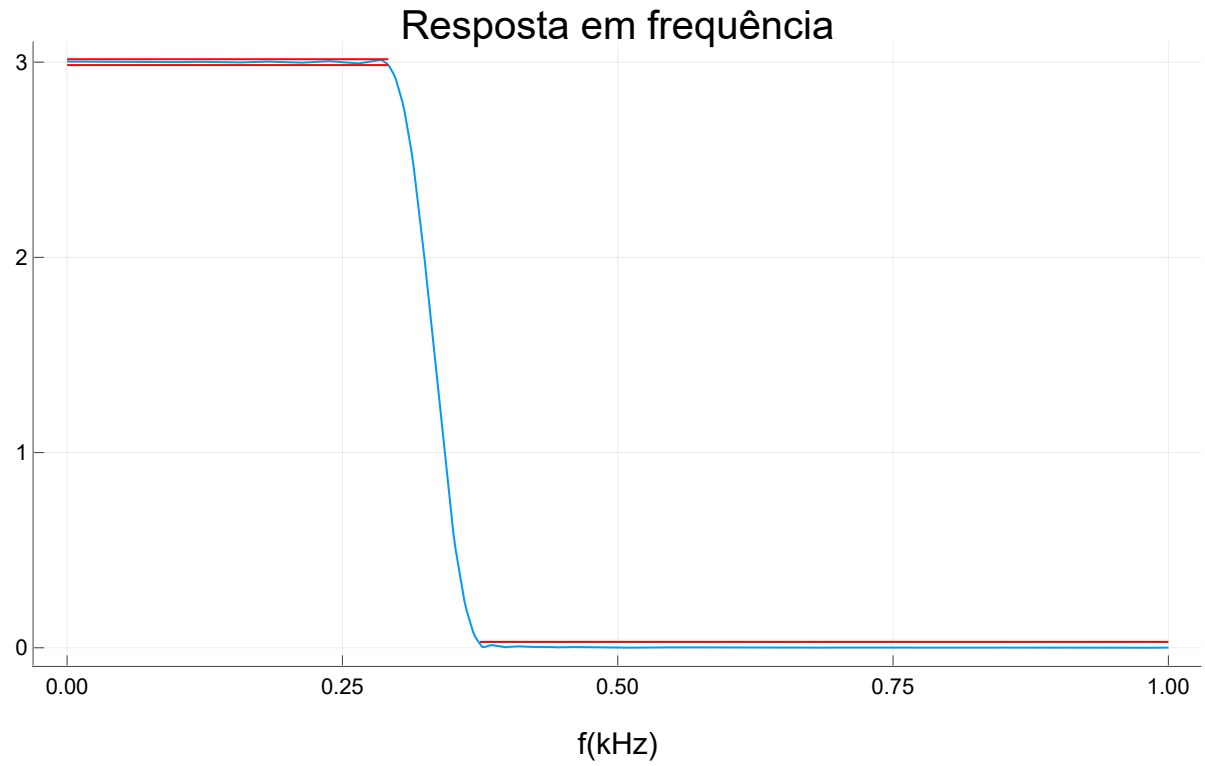
1) Filtro Passa Baixas

Primeiramente é necessário criar um filtro que servirá como interpolador ideal. O interpolador terá a frequência de corte de $\omega_c = \pi/3$ e ganho de passagem de 3.

Os parâmetros da másara do filtro estão descritos na célula a baixo

```
[0.00211369, 0.00278069, 0.0, -0.00443062, -0.00543242, 0.0, 0.0078425, 0.00927483, 0.0,
```

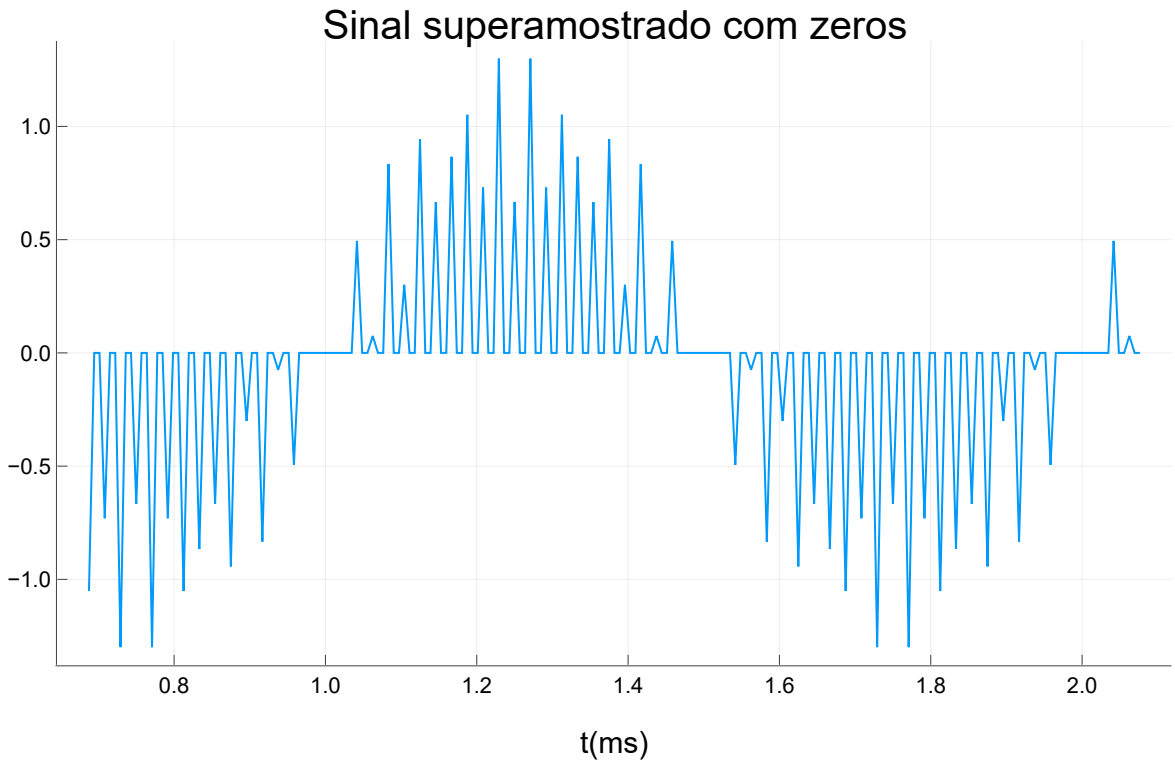
```
begin
    ωp3 =2π*21_000/fa_144
    ωr3 = 2π/L - 2π*21_000/fa_144
    δp3 = 0.005
    δr3 = 0.01
    pb_interpolador = kaiser_filter_lowpass(δp3,δr3,ωp3,ωr3)*3
end
```



2) Criação do sinal super amostrado

O sinal superamostrado será criado colocando 0's entre as amostras e depois interpolando este sinal com um filtro (sinc).

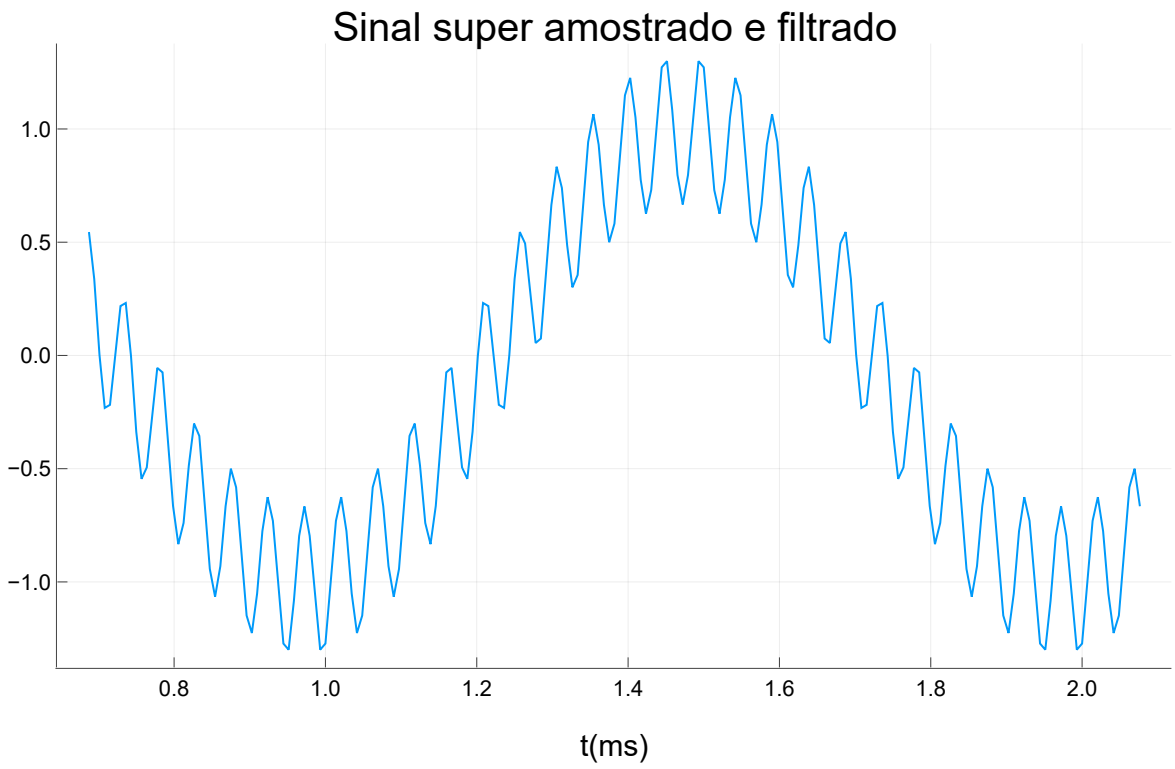
```
begin
    t_144 = range(0,t[end],step = 1/fa_144)
    t_144_ms = t_144.*1000
    y = zeros(length(x_48)*3)
    for i  in 1:length(x_48)
        y[1+(i-1)*3] = x_48[i]
    end
end
```



Interpolação/Filtragem do sinal amostrado com zeros

```
[-1.52656e-16, -1.94289e-16, -1.38778e-16, 6.26719e-6, 8.24488e-6, 2.7864e-16, 0.0010321
```

```
begin
    x_144 = filt(pb_interpolador,y)
end
```



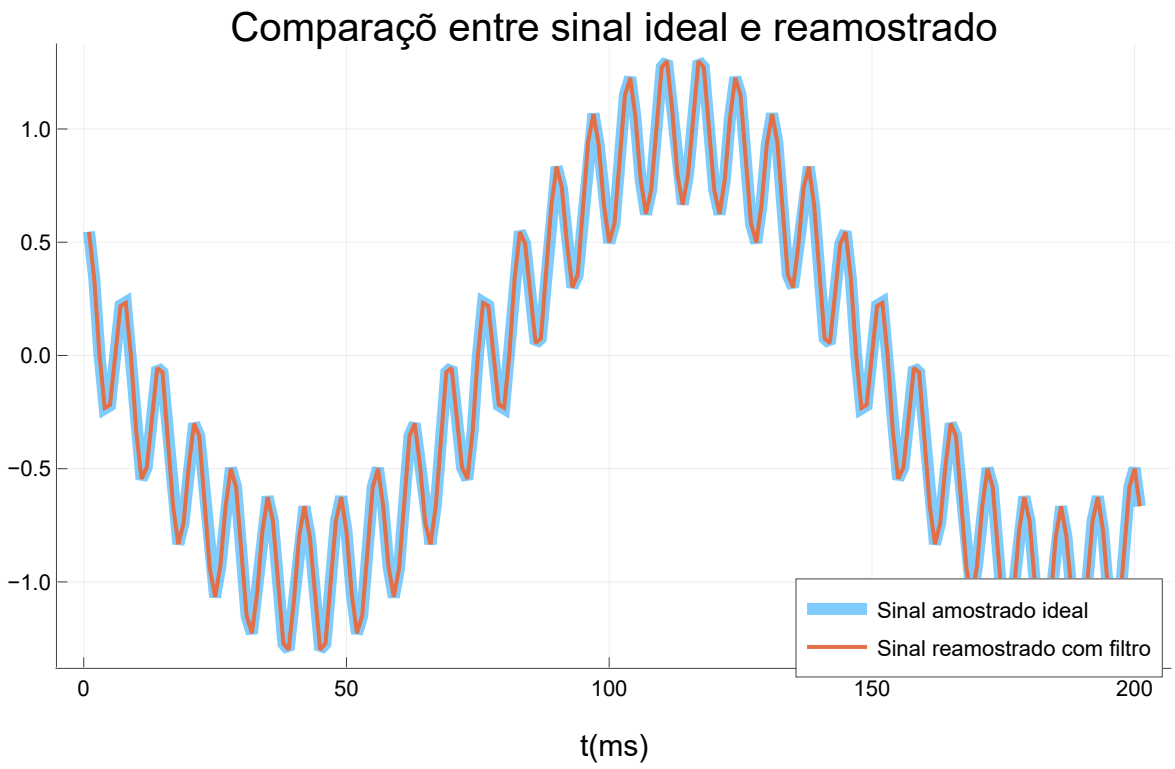
3) Comparação do sinal ideal com o reamostrado

Para verificar a qualidade dessa super amostragem, iremos amostrar o sinal original a 144kHz e comparar os resultados.

Obs: iremos atrasar o sinal ideal para compensar o efeito da filtragem do sinal

```
[0.0, -0.220832, -0.23482, 0.00296505, 0.340315, 0.546921, 0.494521, 0.257197, 0.053345,
```

```
begin
    x_144_ideal = sin.(1e3*2π * t_144) - 1/3*sin.(21e3*2π * t_144)
end
```

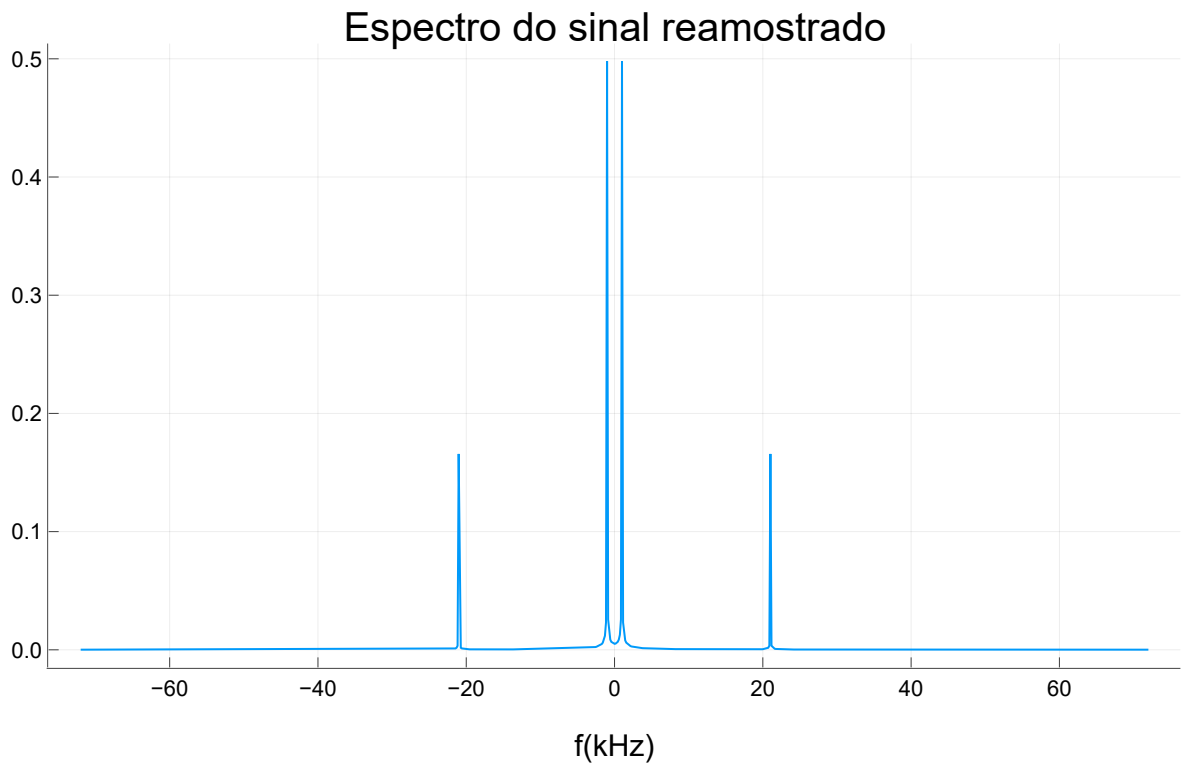


É possível ver que a reamostragem e interpolação com o filtro Passa-Baixas gerá um sinal muito parecido com o sinal originl amostrado a 144kHz

4)Espectro do sinal reamostrado

```
[-0.00508893+0.0im, -0.00519961+0.000675053im, -0.00556162+0.00144145im, -0.00628773+0.(
```

```
begin
    X_144 = fft(x_144[200:1200])/length(x_144[200:1200])
end
```



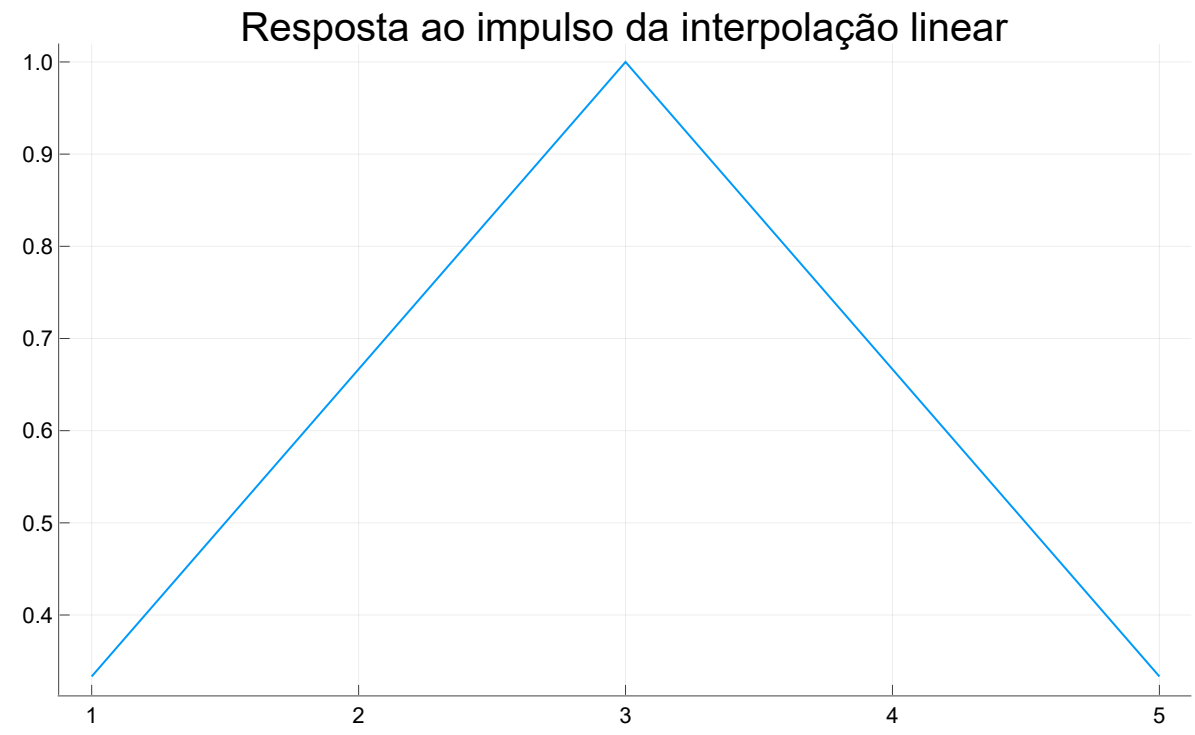
Aumento da taxa de amostragem usando iterpolação linear

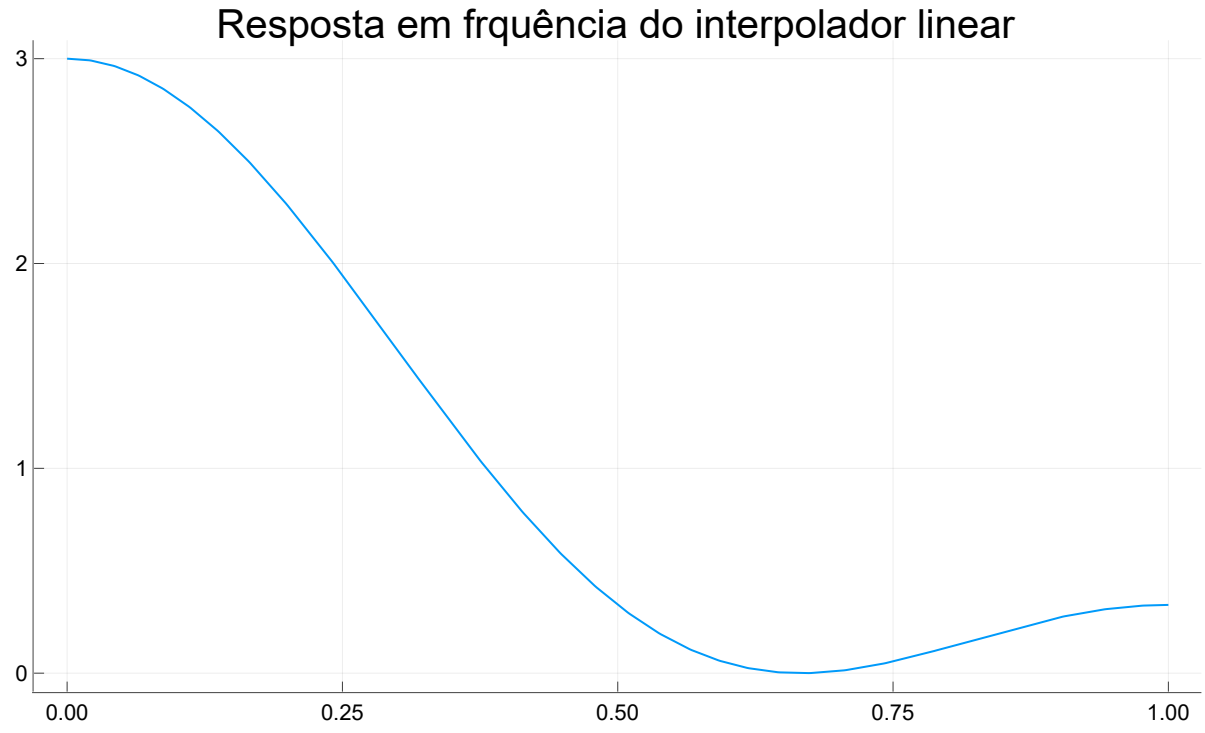
Nesta etapa iremos comparar a interpolação com o filtro Passa-Baixas e uma interpolação linear simples

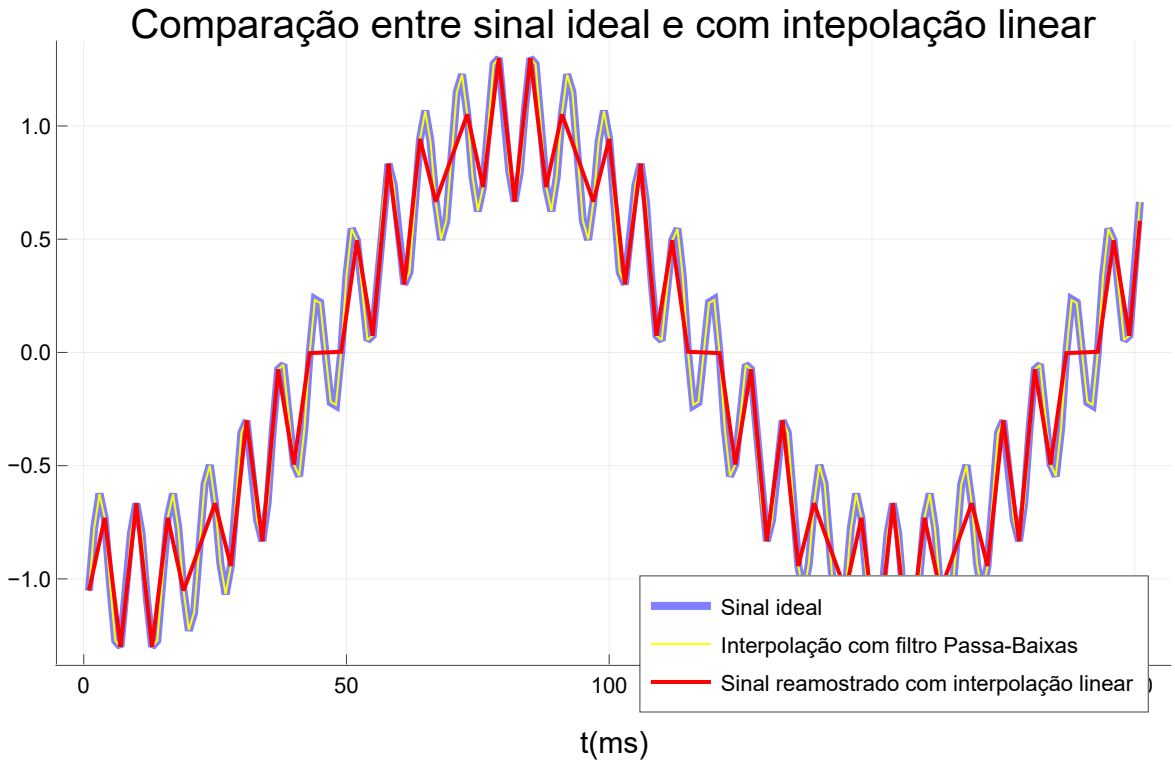
Para fazer a interpolação linear usaremos um filtro triangular como interpolador para poder analisar sua resposta em frequência e netender a origem da distorção do sinal

```
[3.0+0.0im, 2.99968-0.0377727im, 2.99873-0.0755334im, 2.99715-0.11327im, 2.99493-0.1509
```

```
begin
    filt_interp_linear = [1/3,2/3,1,2/3,1/3]
    y_linear = filt(filt_interp_linear,y)

    FILT_interp_linear = freqz(filt_interp_linear,ω)
end
```


Resposta ao impulso da interpolação linear
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1
2
3
4
5


Resposta em frquência do interpolador linear
3
2
1
0
0.00
0.25
0.50
0.75
1.00



Observa-se que o sinal com interpolação linear tem diferenças consideráveis em relação ao sinal original. Portanto conclui-se que a interpolação com o filtro Passa-Baixas tem um resultado muito melhor.

Conversão A/D com sobreamostragem

Nesta etapa iremos fazer a comparação entre a uma conversão analógico digital simples (com taxa de conversã de 40kHz) e uma conersão com sobre amostragem (com fator de M=19).

```
0.0:0.0013157894736842105:2000.0
```

```
begin
    #parâmetros do exercício
    Ω0 = 3_000
    ΔΩ = 3_000
    Ω1 = 2π*750


    B = 5 #bits


    f_40 = 40_000
    fa_40 = 40_000
    fa_5 = f_40*M5 #760kHz
    t_40 = 0:1/fa_40:2
    t_40_ms = t_40*1000
    t_5 = 0:1/fa_5:2
    t_5_ms = t_5*1000
end
```

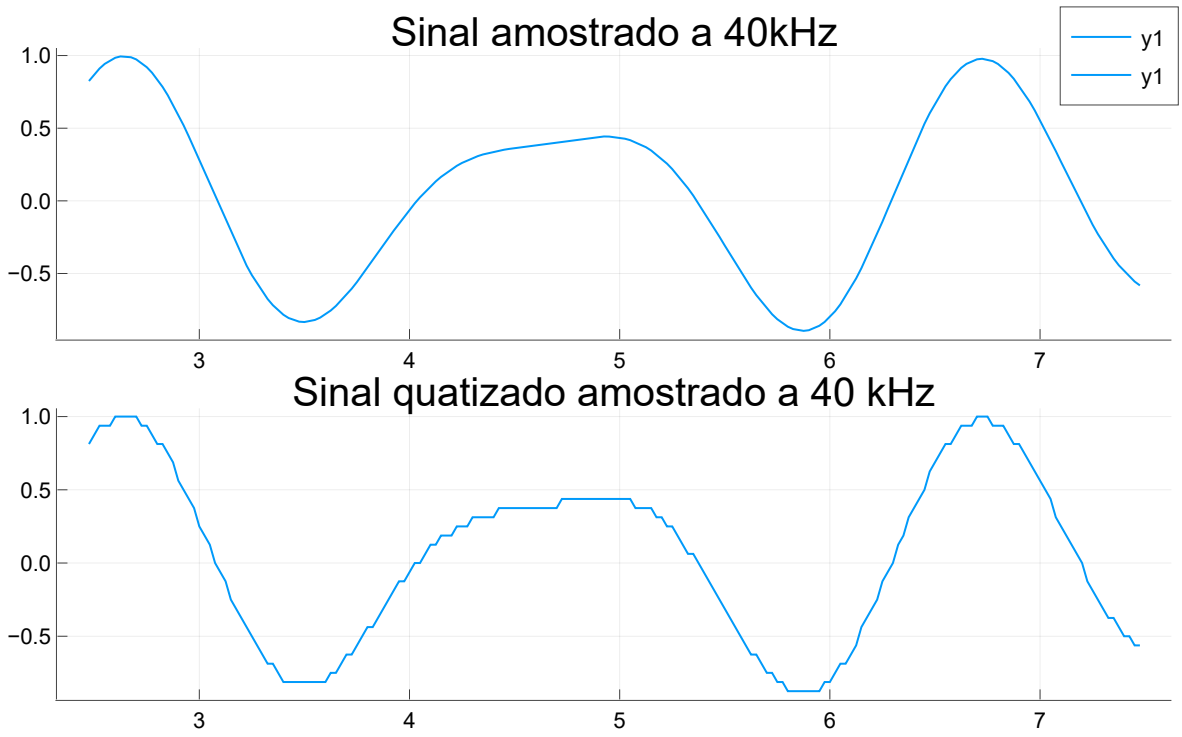
Convesão simples

Aqui teremos uma conversão direta de um sinal analógico para um sinal digital de 5 bits a 40kHz

Sinal a 40kHz

```
[0.31, 0.38, 0.38, 0.44, 0.5, 0.5, 0.56, 0.56, 0.56, 0.56, 0.56, 0.62, 0.62, 0.56, 0.56, (
```

```
begin
    s_40_simples = 0.7sin.((Ω0 .+ 0.5*ΔΩ*t_40).*t_40) + 0.3cos.(Ω1*t_40)
    sq_40_simples = Fixed{Int16,B-1}.(s_40_simples)
end
```




```
begin
    p_s_40 = plot(t_40_ms[range_plot], s_40_simples[range_plot])
    plot!(title="Sinal amostrado a 40kHz")

    p_sq_40 = plot(t_40_ms[range_plot], sq_40_simples[range_plot])
    plot!(title = "Sinal quatizado amostrado a 40 kHz")
    plot(p_s_40,p_sq_40,layout=(2,1))
end
```

Som original

0:00 / 0:02

Som quantizado

0:00 / 0:02

Vemos que há um ruído na quantização do sinal, que será modelado como um ruído uniforma para efeito de cálculos teóricos.

Relação Sinal Ruído

Aqui iremos modelar o ruído observado como uma variável aleatória uniforme para ter uma noção teórico do ruído esperado.

```
29.52052404364425
```

```
begin
    S_40_teo = 0.7^2.0/2 + 0.3^2.0/2
    N_40_teo = 2^(-2.0*B)/3
    σ5 = N_40_teo

    ϵ_simples = s_40_simples-sq_40_simples

    S_40_real = mean(s_40_simples.^2)
    N_40_real = mean(ϵ_simples.^2)

    SN_teo_simples = pow2db(S_40_teo/N_40_teo)
    SN_real_simples = pow2db(S_40_real/N_40_real)
end
```

Temos uma relação **SNR = 29.521dB** na conversão simples a 40kHz de 5 bits.

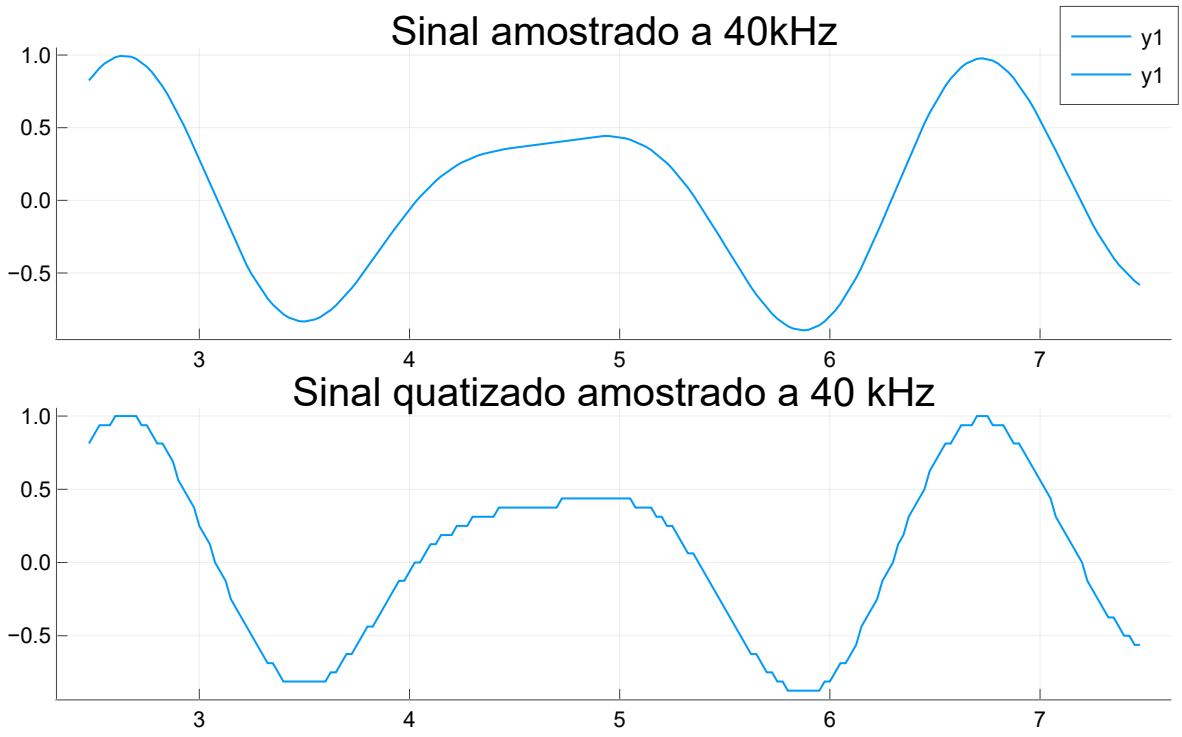
Conversão com sobre amostragem

Aqui usaremos um arranjo de convesão com sobreamostragem, onde o sinal é amostrado a uma taxa alta, depois filtrado e reamostrado à taxa desejada.

Criação do sinal

```
[0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31, 0.31
```

```
begin
    s_5 = 0.7sin.((Ω0 .+ 0.5*ΔΩ*t_5).*t_5) + 0.3cos.(Ω1*t_5)
    sq_5 = Fixed{Int16,B-1}.(s_5)
end
```

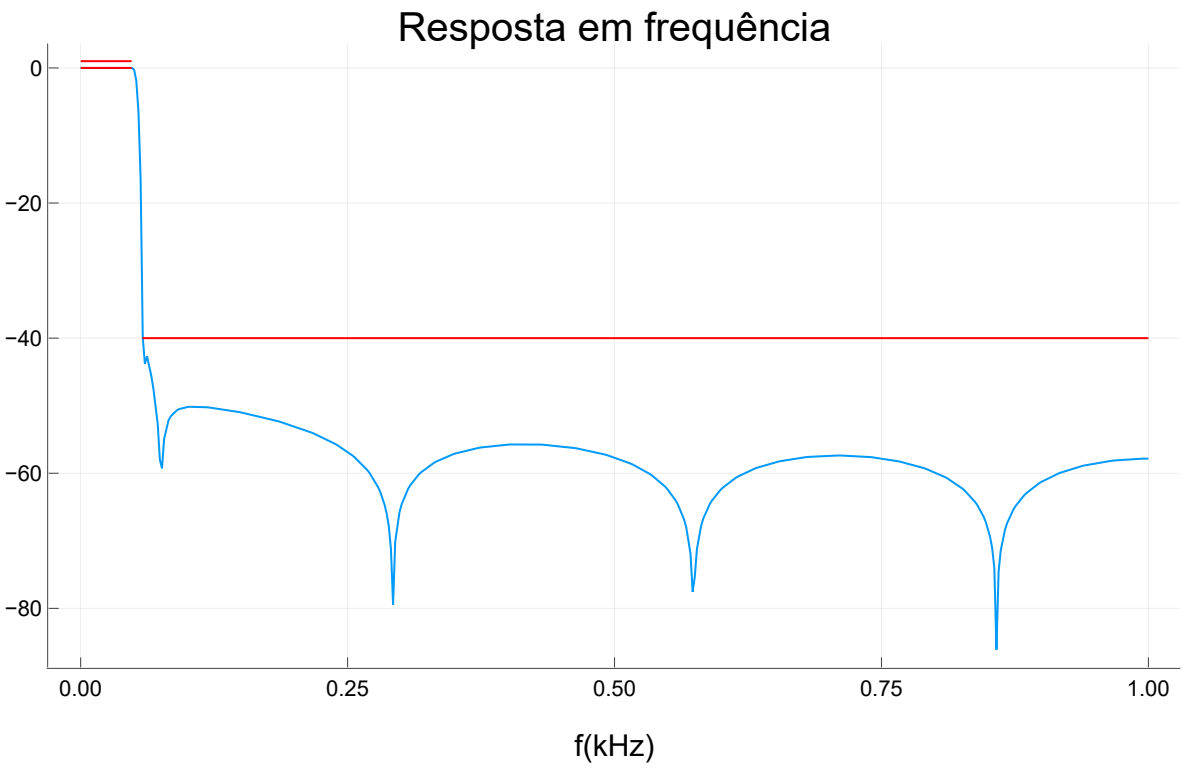



```
begin
    p_s_5 = plot(t_5_ms[range_plot], s_5[range_plot])
    plot!(title="Sinal amostrado a 40kHz")

    p_sq_5 = plot(t_5_ms[range_plot], sq_5[range_plot])
    plot!(title = "Sinal quatizado amostrado a 40 kHz")
    plot(p_s_40,p_sq_40,layout=(2,1))
end
```

Filtro Passa-Baixas

```
1005
```

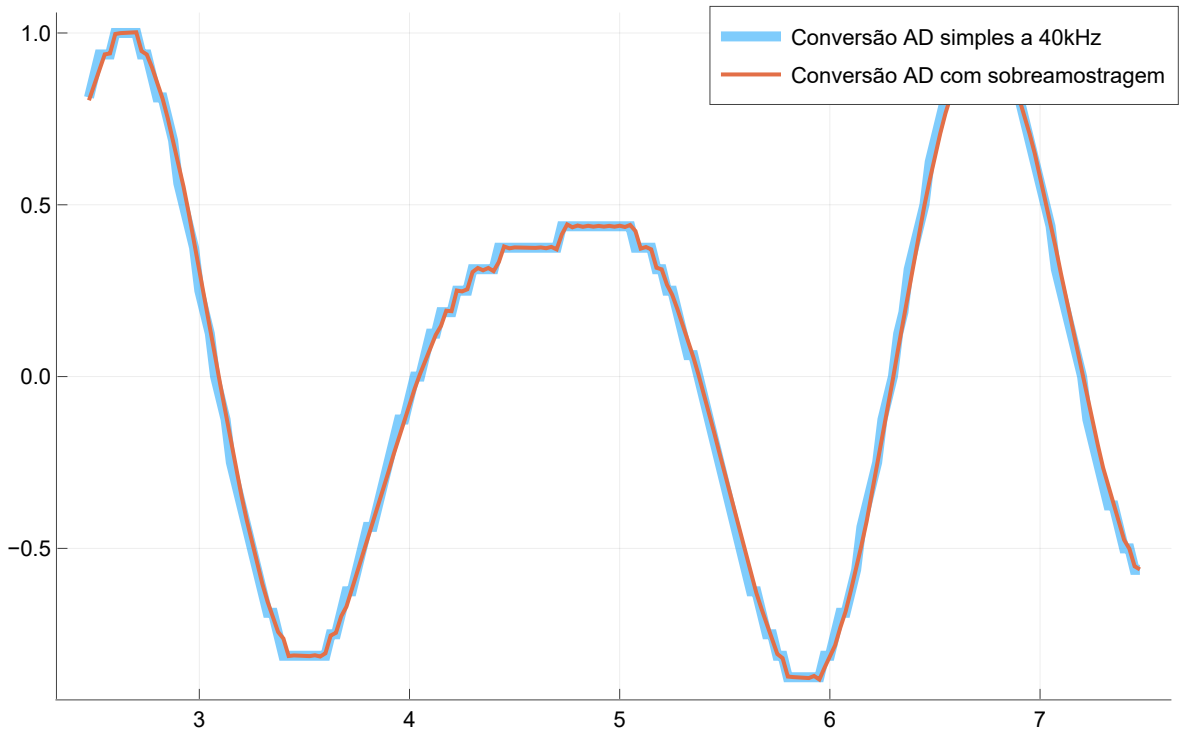
```
begin
    δp5 = 0.0001
    δr5 = 0.0001
    ωp5 = 6/40
    Δω5 = π/100
    ωr5 = ωp5 + Δω5

    M5 = Int16(ceil(π/(ωp5 + Δω5/2) ))

    pb5 = kaiser_filter_lowpass(δp5, δr5, ωp5, ωr5)
    PB5 = freqz(pb5, ω)
    length(pb5)
end
```



Filtragem e reamostragem

```
[5.07265e-7, -4.01307e-6, 3.64794e-6, -9.60691e-6, 1.09331e-5, -2.05221e-5, 2.52733e-5,
```

```
begin
    sq_5_filtrado = filt(pb5, Float64.(sq_5))
    s_5_filtrado = filt(pb5, Float64.(s_5))
    sq_5_40 = reduz_amostragem(sq_5_filtrado,M5)
    s_5_40 = reduz_amostragem(s_5_filtrado,M5)
end
```



Relação Sinal-Ruído

```
37.20166470940386
```

```
begin
    ϵ = s_5_40-sq_5_40

    S_real = mean(s_5_40.^2)
    N_real = mean(ϵ.^2)

    SN_real = pow2db(S_real/N_real)
end
```

Temos agora uma relação sinal ruído de **SNR = 37.202dB** usando a conversão com sobreamostragem

Número de bits relativo

```
6.2699256019725995
```

```
begin
    B_rel =-log2(3*N_real)/2
end
```

O número de bits relativo a essa conversão é de **B' = 6.27**

Anexos

Funções usadas nessa experiência

```
DSP.freqz(filt_fir::Vector, ω) = DSP.freqz(PolynomialRatio(filt_fir,[1]),ω)
```

```
Main.workspace83.kaiser_filter_lowpass
```

```
"""
    kaiser_filter_lowpass{δp, δr, ωp, ωr}
Retorna os coeficientes de um filtro FIR passa baixas de ganho 1 definido com os
parâmetros:

`δp` : Atenuação linear da banda passante em escala linear

`δr` : Atenuação linear da banda de rejeição em escala linear

`ωp` : Frequência passante normalizada (0-π)

`ωr` : Frequência de rejeição normalizada (0-π)
"""
function kaiser_filter_lowpass(δp, δr, ωp, ωr)
    #retorna um filtro passa baixas apenas
    Δω = ωr - ωp
    A= -20log10(min(δp, δr))
    Nk_aux = ceil(Int,(A - 8) / (2.285 * Δω) + 1)
    Nk = ifelse(iseven(Nk_aux), Nk_aux+1, Nk_aux)
    β = kaiserbeta(δp, δr, Δω)
    nk = 0:Nk-1
    kaiser_window = kaiser(Nk,β/π)

    Lk = (Nk -1)÷2
    ωc = (ωr+ωp)/2
    h = ωc/π * sinc.(ωc/π .* (nk.-Lk))
    hk = h.*kaiser_window
    return hk
end
```

```
kaiserbeta (generic function with 1 method)
```

```
function kaiserbeta(δp, δr, Δω)
     δ = min(δp, δr)
     A = -20log10(δ)
     if A < 21
        return 0.0
     elseif A ≤ 50
        return 0.5842(A-21)^0.4 + 0.07886(A-21)
     else
        return 0.1102(A-8.7)
     end
end

```

```
reduz_amostragem (generic function with 1 method)
```

```
function reduz_amostragem(sinal, M)
    #reduz a taxa de amostragem do sinal para a taxa M
    saida = sinal[1:M:end]
end
```