Universidade Federal de Alagoas

Instituto de Computação

Engenharia de Computação

Disciplina: Processamento Digital de Sinais

Professor: Thiago D. Cordeiro

Aluno: Hugo Tallys Martins Oliveira

Atividade 01 - AB2

Importando os pacotes necessários:

```
begin
    using Random   , Distributions
    using Plots
    using SciPy   : signal, fft
    using LaTeXStrings
end
```

Problema 1

Considere as especificações do filtro, em tempo discreto, dadas a seguir:

$$0.89125 \leq |H(e^{j\omega})| \leq 1 \ , \ 0 \leq \omega \leq 0.2\pi$$

$$|H(e^{j\omega})| \leq 0.17783 \ , \ 0.3\pi \leq \omega \leq \pi$$

Utilizando o método da *Transformação Bilinear*, projete um filtro em tempo contínuo (FTC) para posteriormente encontrar a função de transferência de um filtro em tempo discreto (FTD) que esteja de acordo com as especificações dadas. Apresente os diagramas de magnitude para o FTC e para o FTD e aplique sinais na entrada dos filtros para comprovar que as funções de transferência possuem desempenho adequado.

Solucão

No projeto de filtro utilizando a transformação bilinear pré-deformamos as frequências em tempo discreto para tempo contínuo. Seja $|H_c(j\Omega)|$ a resposta em magnitude do filtro de tempo contínuo, então teremos que:

$$0.89125 \leq |H_c(j\Omega)| \leq 1\ ,\ 0 \leq \Omega \leq \frac{2}{T_d} tg(0.2\pi/2)$$

$$|H_c(j\Omega)| \leq 0.17783\ ,\ \frac{2}{T_d} tg(0.3\pi/2) \leq \Omega \leq \infty$$

Onde, por coveniência, exolhemos $T_d = 1$ (os efeitos de T_d se anulam durante as transformações). Além disso, já que um filtro Butterworth de tempo contínuo tem uma resposta em magnitude monotônica, podemos requerer de forma equivalente que

$$|H_c(j2tg(0.1\pi))| \geq 0.89125 \ \text{ e } \ |H_c(j2tg(0,15\pi))| \leq 0.17783$$

A forma da função de magnitude ao quadrado para o filtro Butterworth é:

$$|Hc(j\Omega)|^2 = \frac{1}{1 + (\Omega/\Omega_c)^{2N}}$$

Calculando N e Ω_c com o sinal de igualdade nas equações anteriores, obtemos:

$$1 + \left(\frac{2tg(0.1\pi)}{\Omega_c}\right)^{2N} = \left(\frac{1}{0.892}\right)^2$$

$$1 + \left(\frac{2tg(0.15\pi)}{\Omega_c}\right)^{2N} = \left(\frac{1}{0.178}\right)^2$$

Resolvendo para N, encontramos:

$$N = \frac{log[((\frac{1}{0.178})^2 - 1)/((\frac{1}{0.892})^2 - 1)]}{2log[tg(0.15\pi)/tg(0.1\pi)]} = 5.305$$

Como é preciso que N seja um inteiro, escolhemos $N = 6$. Substituindo este valor de N, obtemos $\Omega_c = 0.766$.

Instanciando o *filtro de Butterworth* a tempo cointínuo:

```
Hc = PyObject TransferFunctionContinuous(
     array([0.21835861]),
     array([1.        , 2.99823376, 4.49470285, 4.27177467, 2.70660219,
            1.08720734, 0.21835861]),
     dt: None
     )
```

```
Hc = signal.lti(signal.butter(6, 0.776, analog=true)...)
```

O que nos dá uma função de transferência em tempo contínuo aproximadamente dada por:

$$H_c(s) = \frac{0.2183}{s^6 + 2.99s^5 + 4.49s^4 + 4.27s^3 + 2.70s^2 + 1.08s + 0.21}$$

Em seguida utilizamos a seguinte relação da transformação bilinear para converter o filtro analógico para digital:

$$s \leftarrow \frac{2}{T_d}\frac{z-1}{z+1}$$

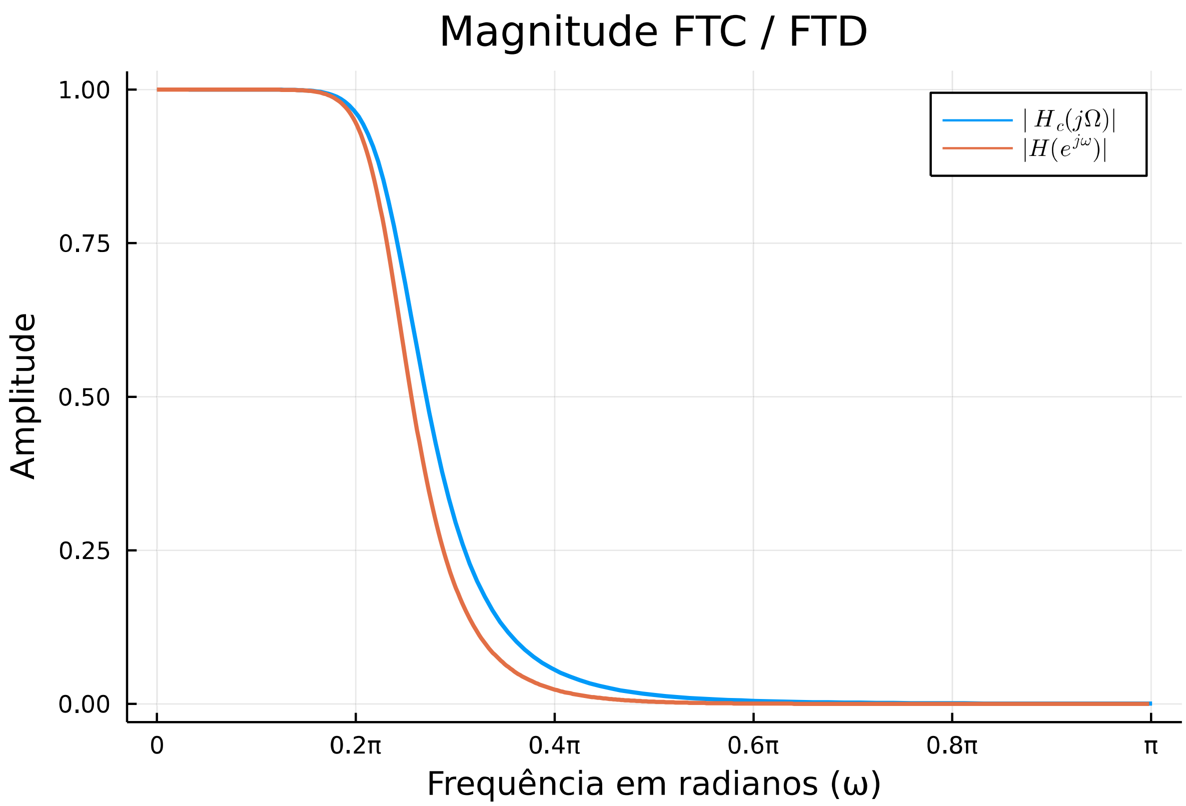
```
H = PyObject TransferFunctionContinuous(
    array([0.00078194, 0.00469164, 0.01172911, 0.01563882, 0.01172911,
           0.00469164, 0.00078194]),
    array([ 1.        , -3.15107305,  4.54285065, -3.69299805,  1.7635206 ,
           -0.46478167,  0.05252573]),
    dt: None
    )
```

```
H = signal.lti(signal.bilinear(Hc.num, Hc.den, 1.)...)
```

O que nos dá (aproximadamente) a seguinte função de transferência para o filtro digital:

$$H(z) = \frac{0.0007378(1+z^{-1})^6}{(1-1.2z^{-1}+0.7z^{-2})(1-1.0z^{-1}+0.35z^{-2})(1-0.90z^{-1}+0.21z^{-2})}$$

Plotando a magnitude da resposta para os filtros contínuo e discreto:




```
begin
    wz, hz = signal.freqz(H.num, H.den)
    ws, hs = signal.freqs(Hc.num, Hc.den)

    cutIdx = findfirst(x -> x > π, ws)
    ws, hs = ws[1:cutIdx], hs[1:cutIdx]

    plot(ws, abs.(hs), label=L"|H_c(jΩ)|",  linewidth=2)
    plot!(wz, abs.(hz), label=L"|H(e^{jω})|", linewidth=2)
    plot!(
        xticks=(
            collect(pi .* (0:0.2:1)),
            ["0", "0.2π", "0.4π", "0.6π", "0.8π", "π"]
        ),
        xlabel="Frequência em radianos (ω)",
        ylabel="Amplitude",
        title="Magnitude FTC / FTD"
    )
end
```

Para validar o projeto dos filtros, vamos aplicar um sinal de entrada dado por:

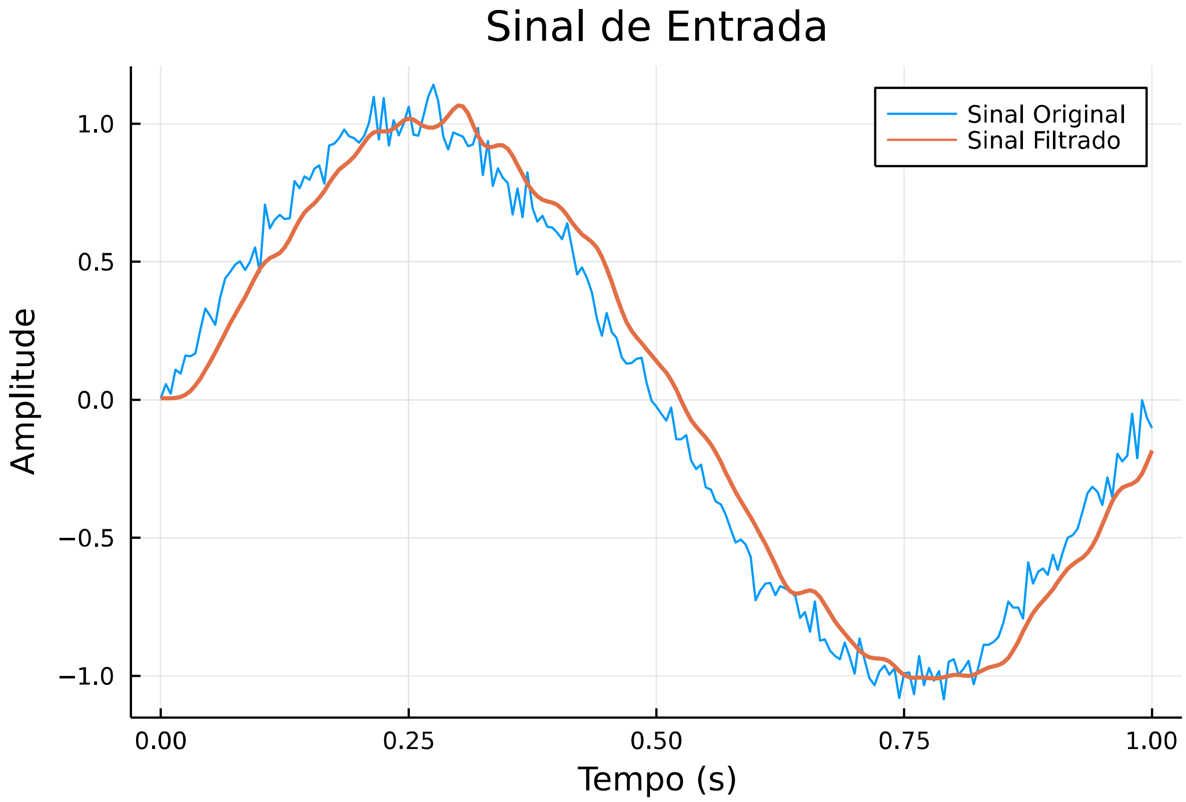
$$y(t) = sin(2\pi t) + 0.05 \cdot n(t)$$

Contaminado por um ruído de alta frequência $n \sim \mathcal{N}(0, 1)$. Note que se estamos amostrando nosso sinal a tempo contínuo numa frequência $\Omega_s = 1/T_s = 1/0.005$, entao a frequência de corte será:

$$\Omega_c = 2\Omega_s tg(0.1\pi) \approx 130Hz$$

```
Ωc = 129.96787849316252
```

```
Ωc = (2/0.005)*tan(0.1π)
```




```julia
begin
    t = 0:0.005:1
    y = sin.(2π*t)
    s = y + 0.05*rand(Normal(), size(y))

    zi = signal.lfilter_zi(H.num, H.den)
    z, _ = signal.lfilter(H.num, H.den, s, zi=zi*s[1])
    plot(t, s, label="Sinal Original")
    plot!(t, z, label="Sinal Filtrado", linewidth=2)
    plot!(
        xlabel="Tempo (s)",
        ylabel="Amplitude",
        title="Sinal de Entrada"
    )
end
```

Problema 2

Considere o filtro passa-baixa *Chebyshev do tipo I* dado a seguir:

$$H_{lp}(Z) = \frac{0.001836(1+Z^{-1})^4}{(1-1.5548Z^{-1}+0.6493Z^{-2})\,(1-1.4996Z^{-1}+0.8482Z^{-2})}$$

Este filtro foi projetado para atender as seguintes especificações de desempenho:

$$0.89125 \leq |H_{lp}(e^{j\theta})| \leq 1\ ,\ 0 \leq \theta \leq 0.2\pi$$

$$|H_{lp}(e^{j\theta})| \leq 0.17783\ ,\ 0.3\pi \leq \theta \leq \pi$$

Utilizando a técnica de transformação de frequências para filtros do tipo passa-baixa, transforme o filtro descrito em um filtro passa-alta com frequência de corte $\omega_p = 0.6\pi$. Apresente os diagramas de magnitude para os dois filtros e aplique sinais na entrada de ambos para comprovar que as funções de transferência possuem desempenho adequado.

Solução

Para transformar esse filtro em um filtro passa-altas com frequência de corte da faixa de passagem $\omega_p = 0.6\pi$, utilizaremos a seguinte transformação:

$$Z^{-1} = -\frac{z^{-1}+\alpha}{1+\alpha z^{-1}}$$

onde $\alpha = -cos([0.2\pi + 0.6\pi]/2)/cos([0.2\pi - 0.6\pi]/2) = -0.38197$, de modo que:

$$H(z) = H_{lp}([(z^{-1} - 0.38197)/(1 - 0.38197z^{-1})])$$

Isto é:

$$H(z) = \frac{0.02426(1-z^{-1})^4}{(1+1.0416z^{-1}+0.4019z^{-2})(1+0.5661z^{-1}+0.7657z^{-2})}$$

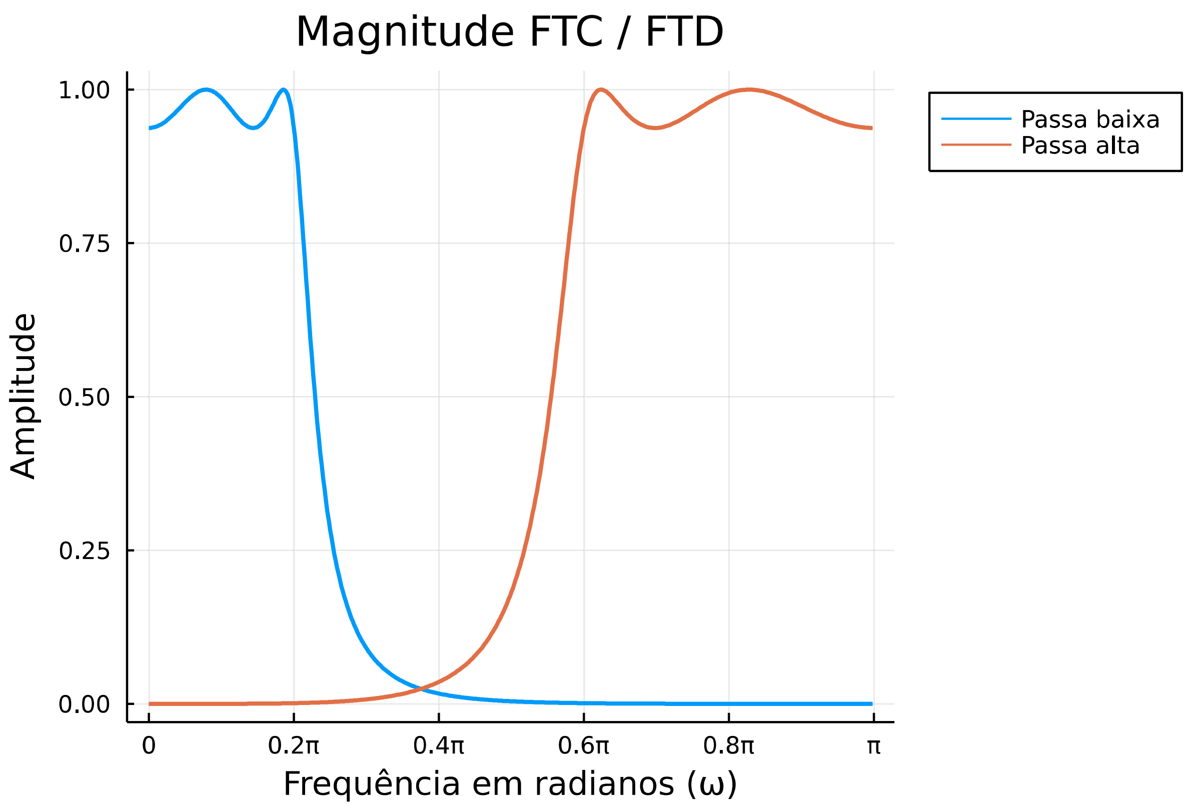
Instânciando o filtro passa baixa de ordem 4, com o desvio 0.56 dB e frequência de corte desejada 0.2π (normalizada):

```
Hlp = signal.lti(signal.cheby1(4, 0.56, 0.2)...);
```

Para criar o filtro passa-alta, basta especificar a nova frequência de corte e tipo `"highpass"`:

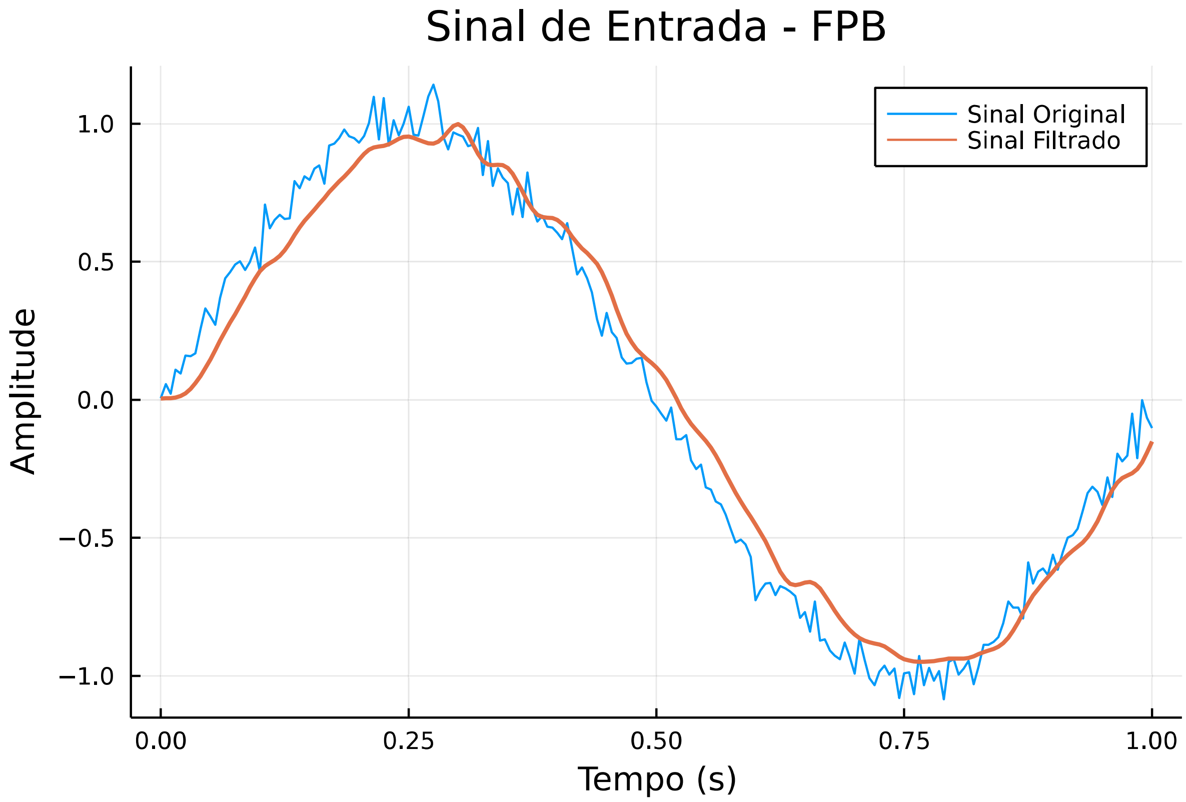
```
Hhp = signal.lti(signal.cheby1(4, 0.56, 0.6, "highpass")...);
```

Plotando os diagramas de magintude:



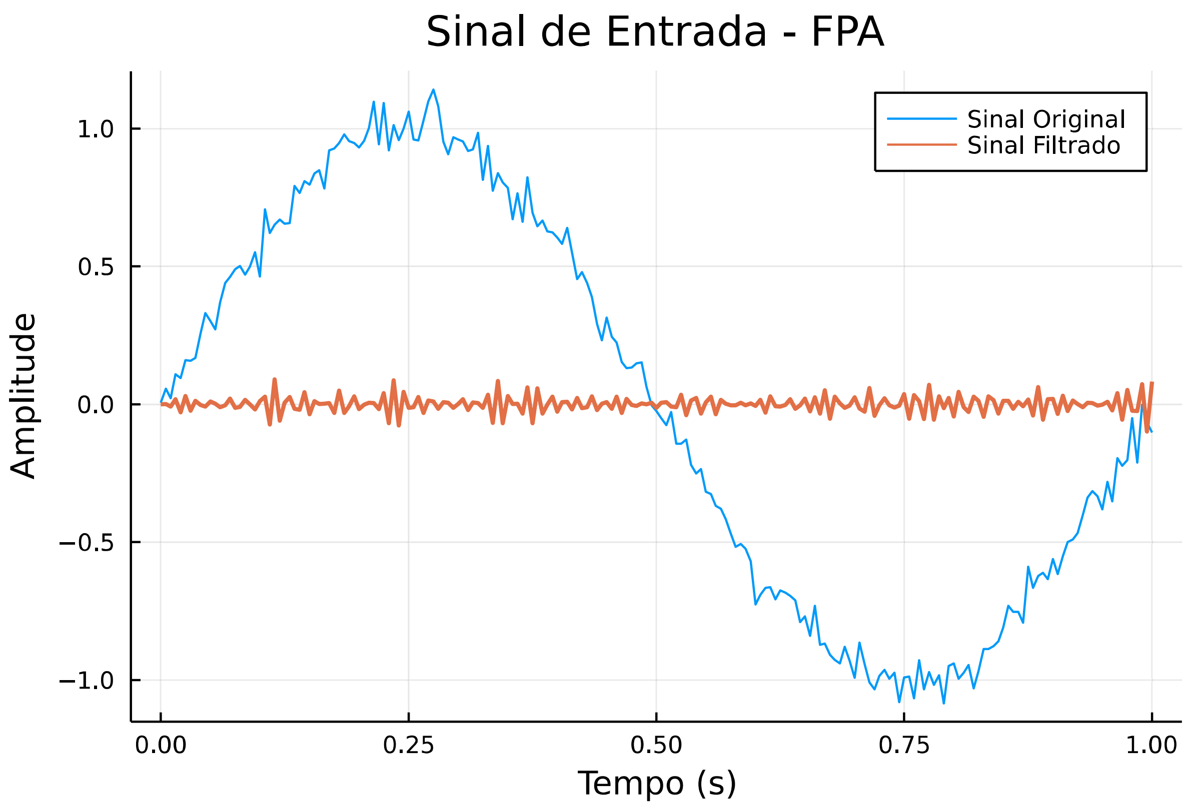

```
begin
    wzLp, hzLp = signal.freqz(Hlp.num, Hlp.den)
    wzHp, hzHp = signal.freqz(Hhp.num, Hhp.den)

    plot(wzLp, abs.(hzLp), label="Passa baixa", linewidth=2)
    plot!(wzHp, abs.(hzHp), label="Passa alta", linewidth=2)
    plot!(
        xticks=(
            collect(pi .* (0:0.2:1)),
            ["0", "0.2π", "0.4π", "0.6π", "0.8π", "π"]
        ),
        xlabel="Frequência em radianos (ω)",
        ylabel="Amplitude",
        title="Magnitude FTC / FTD",
        legend=:outertopright
    )
end
```

Aplicando o filtro passa-baixa ao sinal de entrada, obtemos um resultado similar ao anterior:




```
begin
    ziLp = signal.lfilter_zi(Hlp.num, Hlp.den)
    zLp, _ = signal.lfilter(Hlp.num, Hlp.den, s, zi=ziLp*s[1])
    plot(t, s, label="Sinal Original")
    plot!(t, zLp, label="Sinal Filtrado", linewidth=2)
    plot!(
        xlabel="Tempo (s)",
        ylabel="Amplitude",
        title="Sinal de Entrada - FPB"
    )
end
```

Aplicando o filtro passa-alta no sinal de entrada, obtemos a filtragem apenas do sinal base, deixando passar o ruído.




```
begin
    ziHp = signal.lfilter_zi(Hhp.num, Hhp.den)
    zHp, _ = signal.lfilter(Hhp.num, Hhp.den, s, zi=ziHp*s[1])
    plot(t, s, label="Sinal Original")
    plot!(t, zHp, label="Sinal Filtrado", linewidth=2)
    plot!(
        xlabel="Tempo (s)",
        ylabel="Amplitude",
        title="Sinal de Entrada - FPA"
    )
end
```

Problema 3

Com o uso das fórmulas utilizadas no projeto de filtros pelo método da janela de *Kaiser*, projete um filtro passa-baixa FIR para atender as seguintes especificações de desempenho:

$$\omega_p = 0.4\pi$$

$$\omega_s = 0.6\pi$$

$$\delta_1 = 10^{-2}$$

$$\delta_2 = 10^{-3}$$

Apresente os diagramas de magnitude para este filtro.

Solução

De acordo com as especificações dadas, para o projeto por janela, o filtro resultante terá o mesmo erro de pico δ na faixa de passagem e na faixa de rejeição. Como os filtros projetados pelo método de janelamento têm inerentemente $\delta_1 = \delta_2$, precisamos definir $\delta = 0.001$. Em seguida, encontramos a frequência de corte do filtro passa-baixas, definindo que:

$$\omega_c = \frac{\omega_p + \omega_s}{2} = 0.5\pi$$

devido à simetria da aproximação na descontinuidade de $H_d(e^{j\omega})$.

Para determinar os parâmetros da *janela de Kaiser*, primeiro calculamos $\omega = \omega_s - \omega_p = 0.2\pi$ e $A = -20log_{10}\delta = 60$, a largura da região de transição e o desvio de magintude desejado em dB respectivamente.

Substituímos essas duas quantidades nas equações 7.75 e 7.76 **[1]** para obter os valores requeridos de β e M, calculados empiricamente o que nos dá:

$$\beta = 5.653\ ,\ M = 37$$

Por fim, calculamos a resposta ao impulso do filtro utilizando as equações 7.71 e 7.72 **[1]**, obtendo:

$$h[n] = \frac{sen(\omega_c(n-\alpha))}{\pi(n-\alpha)} \frac{I_0[\beta(1-[(n-\alpha)/\alpha]^2)^{1/2}]}{I_0(\beta)}\ ,\ 0 \leq m \leq M$$

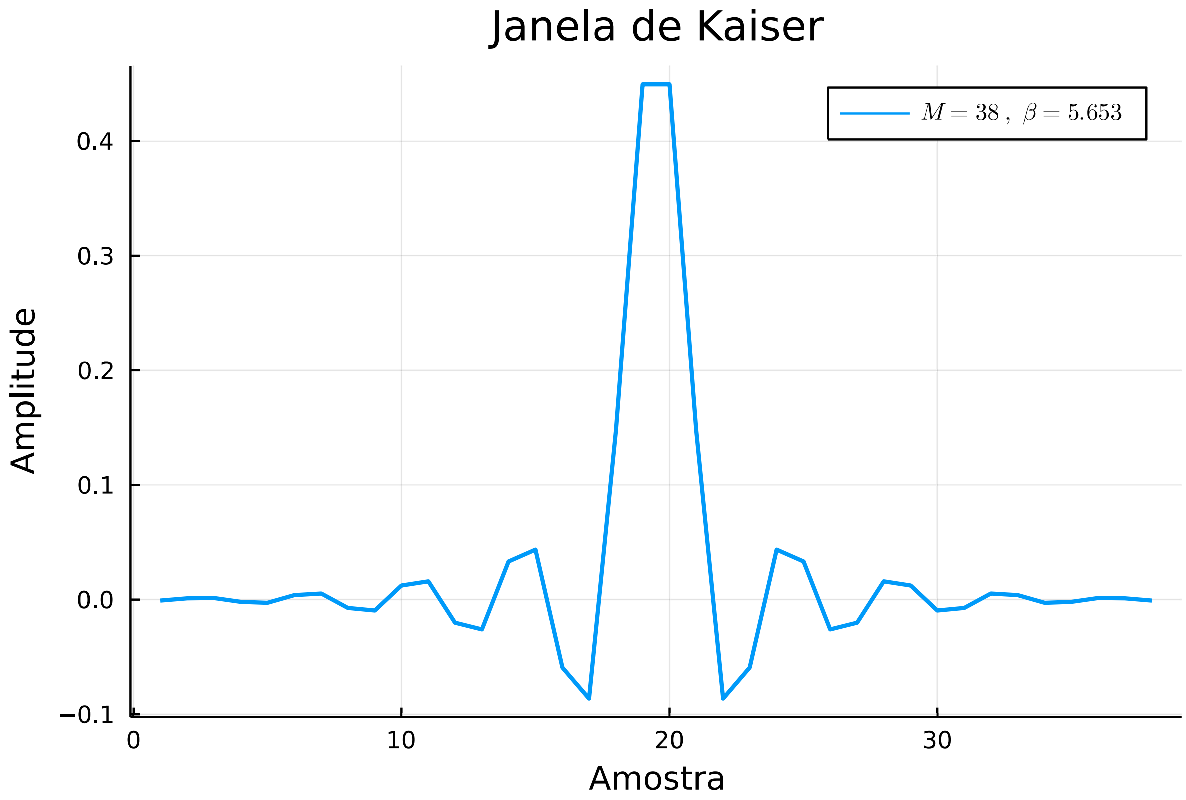
com $h[n] = 0$ caso contrário.

```
(38, 5.65326)
```

```
begin
    ω_p, ω_s = 0.4π, 0.6π
    δ = 0.001
    A = Int(round(-20log(10, δ)))
    M, β = signal.kaiserord(A, (ω_s - ω_p)/π)
end
```

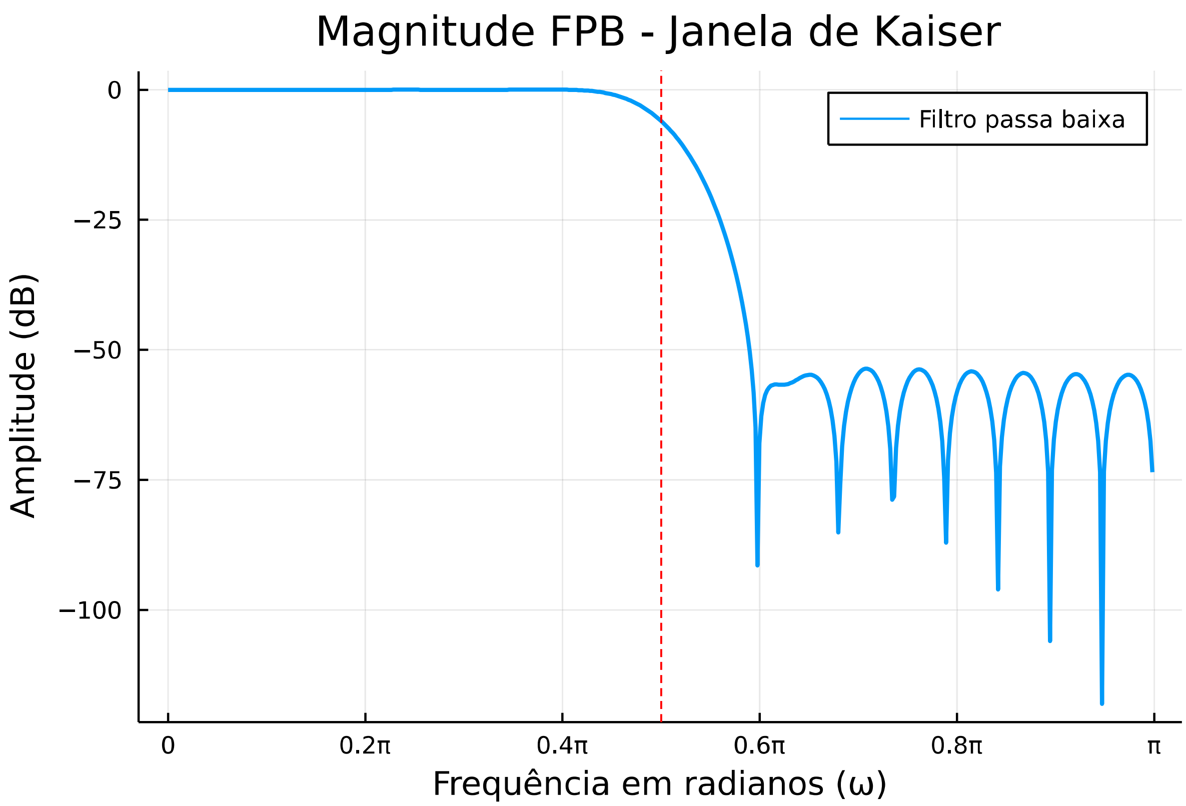
```
begin
    ω_c = (ω_s + ω_p)/2π
    kaiser = signal.firwin(M, ω_c)
end;
```

Plotando a curva da *janela de Kaiser* encontrada:




```
plot(
    kaiser, title="Janela de Kaiser",
    label=L"$M=38 \ , \ \beta=5.653$",
    xlabel="Amostra", ylabel="Amplitude", linewidth=2
)
```

Plotando o diagrama de magintude:




```
begin
    fK, rK = signal.freqz(kaiser)
    plot(fK, 20*log.(10, abs.(rK)), label="Filtro passa baixa", linewidth=2)
    plot!(
        xticks=(
            collect(pi .* (0:0.2:1)),
            ["0", "0.2π", "0.4π", "0.6π", "0.8π", "π"]
        ),
        xlabel="Frequência em radianos (ω)",
        ylabel="Amplitude (dB)",
        title="Magnitude FPB - Janela de Kaiser"
    )
    plot!(
        [0.5π], seriestype="vline",
        style=:dash, color=:red,label=false
    )
end
```

Referências

[1] Oppenheim, Alan V. Processamento em tempo discreto de sinais, 3. ed. – São Paulo: Pearson Education do Brasil, 2012. (páginas 303 - 323)

[2] Documentação SciPy-Signal (https://docs.scipy.org/doc/scipy/reference/signal.html?highlight=signal#module-scipy.signal)