6,0

PSI-3431 Processamento Estatístico de Sinais

Professor: Vítor H. Nascimento

Igor Costa D'Oliveira NUSP 11391446

Wesley Freitas Bernardino NUSP 11808411

Experiência 2

1. Projete usando o método dos mínimos quadrados os filtros a seguir:

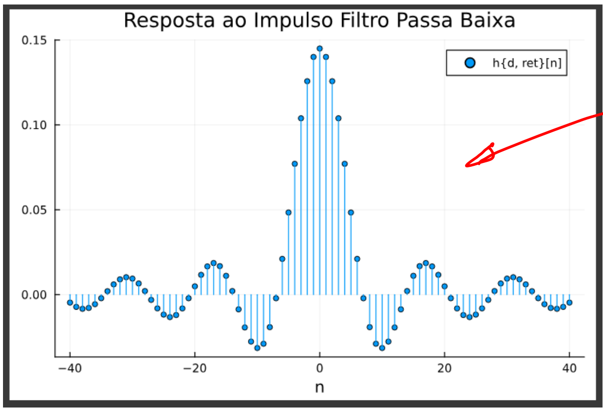
(a) Um filtro com N = 81 coeficientes que aproxime a resposta ideal

$$H_d(e^{jw}) = \begin{cases} 1, & |w| < 29\pi/200 \\ 0, & 29\pi/200 \leq |w| \leq \pi \end{cases}$$

o comprimento é N.
L é o atraso.

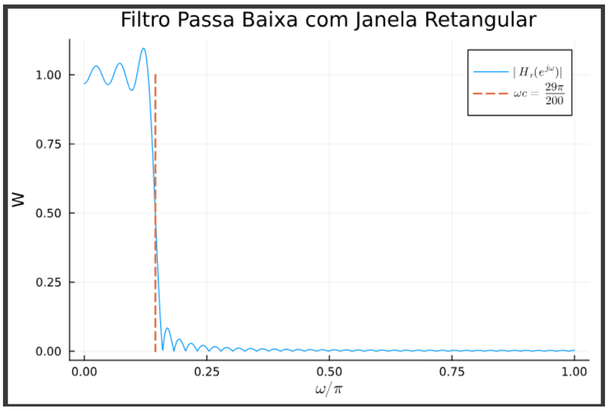
Resp: No método dos mínimos quadrados temos que o comprimento total da janela deve valor L = (N-1)/2 com fase linear generalizada. Também temos que este filtro é um filtro passa baixa de acordo com a sua resposta. Assim utilizaremos a função h = hd = (wc/π) * sinc.((wc/π)*n).

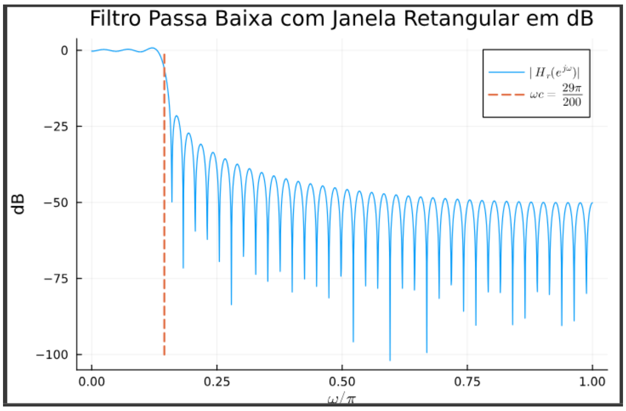
Abaixo temos a resposta ao impulso gerado a partir da função descrita acima para o filtro passa baixa.



este é o filtro para atraso (não-causal, portanto).

Assim, foi gerada a resposta a frequência do filtro, no qual os dois gráficos abaixo ilustram os módulos em W e em dB do filtro. É possível concluir que o filtro foi dimensionado corretamente visto que está atenuando a entrada justamente quando w for maior que wc.



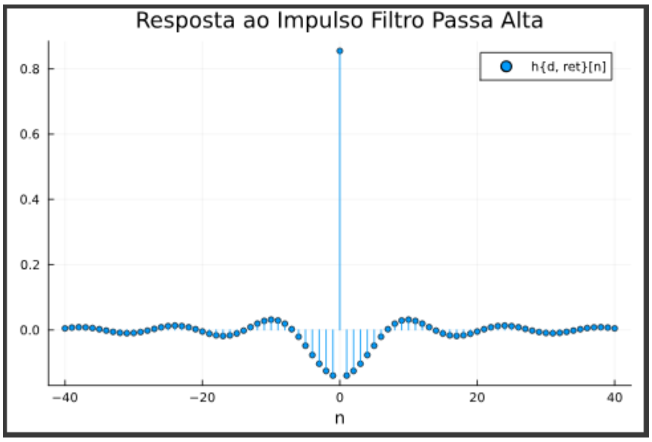


(b) Um filtro com N = 81 coeficientes que aproxime a resposta ideal

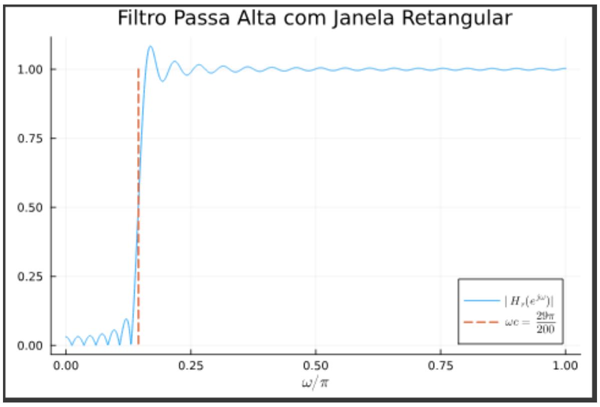
$$H_d(e^{jw}) = \begin{cases} 0, & |w| < 29\pi/200 \\ 1, & 29\pi/200 \leq |w| \leq \pi \end{cases}$$

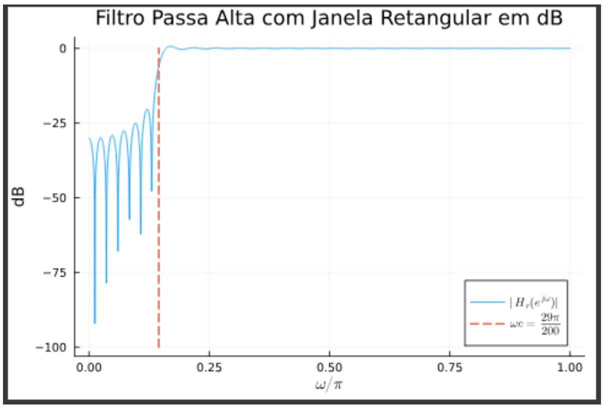
Resp: Agora temos que projetar um filtro passa alta, no qual utilizaremos a função resposta ao impulso igual a h = δ[n] - (wc/π) * sinc.((wc/π)*n). As passagens deste item são idênticas ao anterior, só mudou h. Por que?

Abaixo temos a resposta ao impulso gerado a partir da função descrita acima para o filtro passa alta.



Assim, foi gerada a resposta a frequência do filtro, no qual os dois gráficos abaixo ilustram os módulos em W e em dB do filtro. É possível concluir que o filtro foi dimensionado corretamente visto que está atenuando a entrada justamente quando w for maior que wc.





2. Escreva em Matlab, Julia ou Python um programa para implementar o filtro

$$y[n] = h[0]x[n] + h[1]x[n-1] + \ldots h[N-1]x[n-N+1].$$

O programa deve satisfazer as seguintes condições:

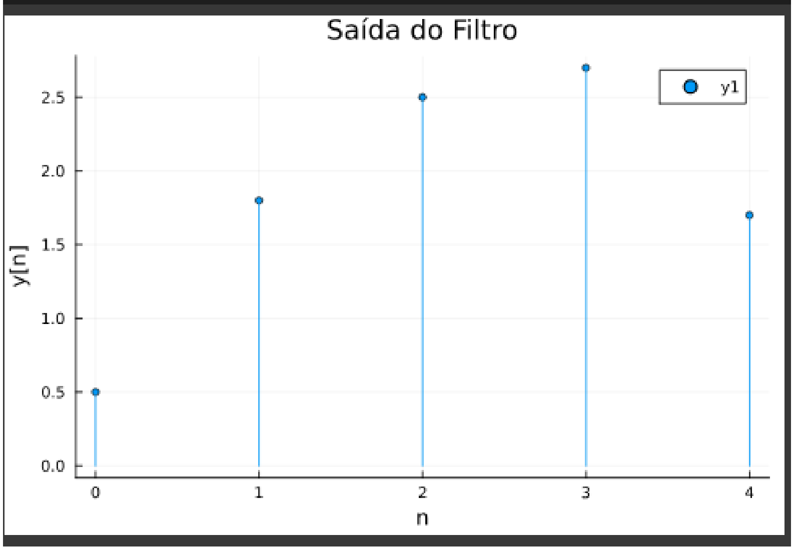
(a) Deve ser criada uma função. As entradas da função devem ser K amostras do sinal x[n] para n = 0 . . . K − 1 e os coeficientes do filtro, h[n] para n = 0 . . . N − 1. (b) A função deve ser escrita usando apenas funções básicas, como laços for, comandos if-then-else, etc. Não é permitido usar funções prontas como conv, fft, filt ou filter. (c) O filtro deve gerar a saída assumindo que as entradas para n < 0 são nulas, e deve calcular a saída para os instantes de 0 a K − 1.

Resp: Foi desenvolvida a função filtro_FIR(x, h) com as entradas pedidas:

```
function filtro_FIR(x, h)
    K = length(x)
    N = length(h)
    y = zeros(K)

    for n = 0:K-1
        for k = 0:N-1
            if n-k >= 0
                y[n+1] += h[k+1] * x[n-k+1] # Aplica a convolução
            end
        end
    end

    return y
end
```

Nela é usada a propriedade da soma de convolução entre os pontos da entrada e a resposta ao impulso do filtro. Por fim, testamos a função com valores de entrada e coeficientes do filtro aleatórios e obtivemos a saída conforme a imagem abaixo. Realizando a convolução manualmente é possível concluir que o filtro foi implementado corretamente.

```
x = [1, 3, 3, 3, 1]  # Amostras do sinal de entrada
h = [0.5, 0.3, 0.1]  # Coeficientes do filtro

y = filtro_FIR(x, h)  # Chama a função filtro_FIR
n = 0:length(y)-1

plot(n, y, line = :stem, marker = (:circle, 3), xlabel = "n",
     ylabel = "y[n]", title = "Saída do Filtro")
```



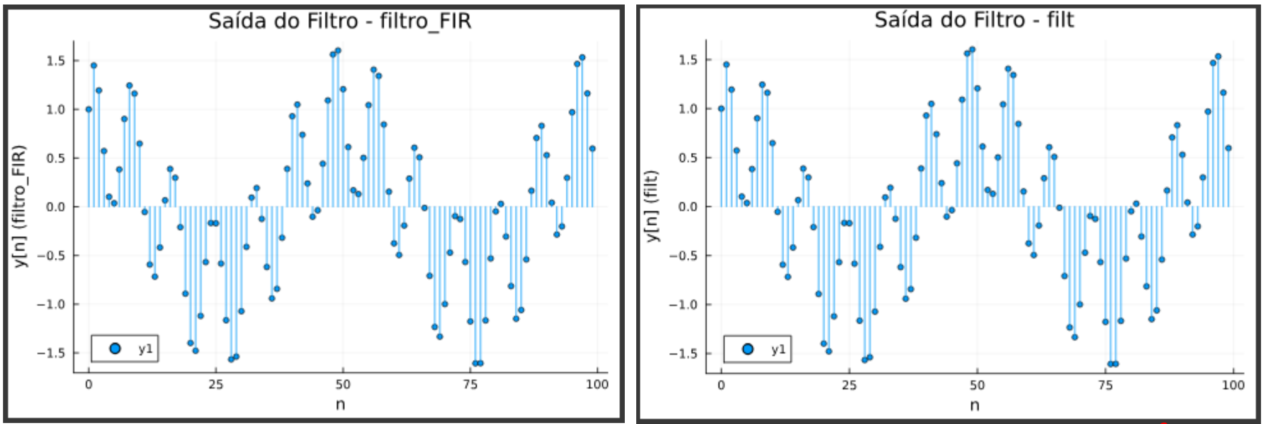
3. Teste o seu programa e os seus filtros para o sinal de entrada x[n] = cos(πn/25) + cos(πn/4).

(a) Compare a saída do seu programa com a do comando filter (Matlab) ou filt (Julia) e obtenha a saída de cada um dos filtros para o sinal acima.

Resp: Foi necessário criar a entrada x[n] = cos(πn/25) + cos(πn/4) e utilizamos 100 amostras para isso: → Com apenas 100 amostras, você só vai ver o transitório do filtro. O regime permanente para um filtro FIR começa na amostra N.

```
x = [cos(π*n/25) + cos(π*n/4) for n in 0:99]  # Amostras do sinal de entrada

y_custom = filtro_FIR(x, h)  # Saída do filtro usando o programa implementado

# Plot da saída do filtro usando o programa implementado
n = 0:length(y_custom)-1
plot(n, y_custom, line = :stem, marker = (:circle, 3), xlabel = "n",
     ylabel = "y[n] (filtro_FIR)", title = "Saída do Filtro - filtro_FIR")
```

Foram obtidas as saídas dos filtros utilizando a função do item 2 e a função filt (Julia). De acordo com os gráficos gerados abaixo é possível notar que as saídas foram exatamente iguais nos dois filtros, logo pode-se concluir que o filtro projetado no item 2 está correto.

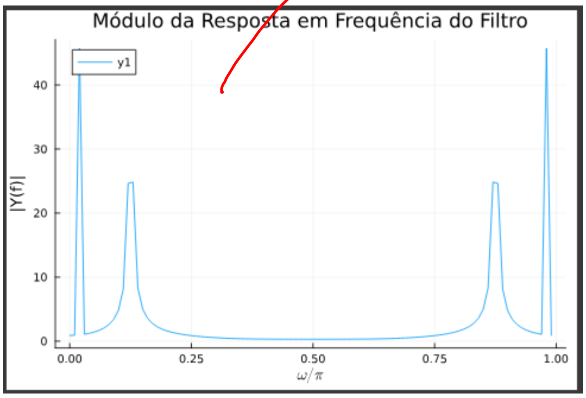


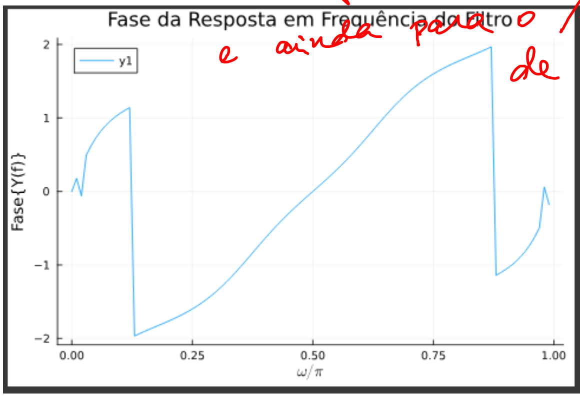
↳ Vocês passaram o sinal pelo filtro de 3 coeficientes que vocês usaram como teste, não pelo passa-baixas.

(b) Desenhe a resposta em frequência (módulo e fase) do filtro. Usando os sinais x1[n] = cos(πn/25) e x2[n] = cos(πn/4), compare a amplitude observada do sinal de saída dos filtros com as respostas em frequência calculadas para as duas frequências.

Resp: Para encontrar a resposta em frequência do filtro foi utilizado a função fft, logo obtivemos o gráfico abaixo:

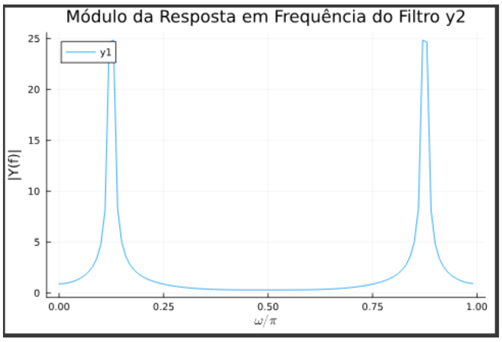
Da maneira como vocês usaram a FFT não funciona. Vocês calcularam a transformada da saída, e ainda para o / pelo "filtro" de 3 coeficientes





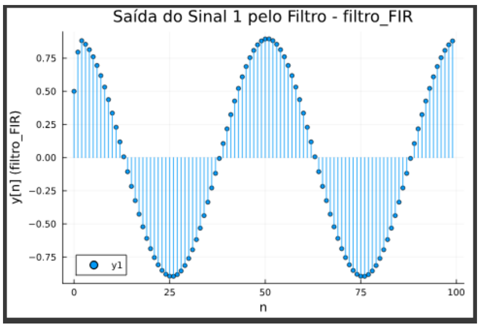
Nela é possível observar que possui duas componentes, uma em cada frequência dos sinais cossenoidais que compõem a entrada. Para confirmar isto nós calculamos a resposta em frequência dos sinais 1 e 2 separadamente e os seus gráficos se encontram abaixo:

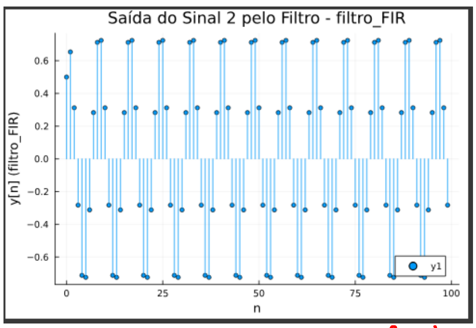
Sinal não tem resposta em frequência, tem transformada.



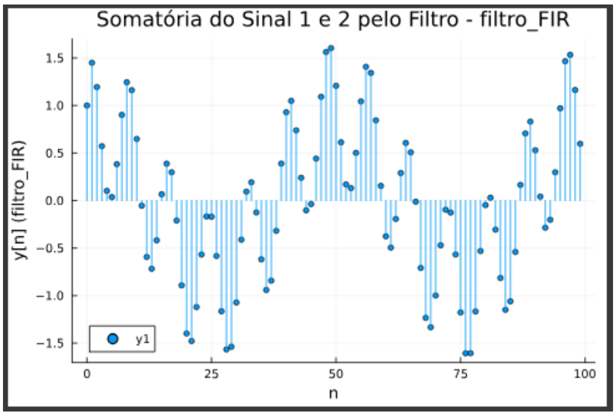
Por último, comparamos as amplitudes observadas dos sinais 1 e 2 com a resposta em frequência encontrada no item (a) e percebemos que a soma das amplitudes dos dois sinais resultava exatamente na saída do Y[n], logo os sinais são independentes. Abaixo é possível visualizar os gráficos com os módulos das amplitudes dos sinais 1, 2 e a sua somatória.

Você está misturando coisas. Os sinais aqui são determinísticos





Repare que o filtro não está filtrando nada.



4. Suponha que você quer fazer um filtro passa-baixas e um passa-altas para separar os dois cossenos do exercício anterior (quer dizer, você quer manter um e eliminar o outro). Projete os filtros usando janelas de Kaiser, supondo que: (a) O erro no ganho da banda-passante deve ser menor ou igual a 0,005; (b) o cosseno que será eliminado teve ser atenuado por pelo menos 0,001.

Determine os valores de ωp e ωr adequados. Projete os filtros, mostre as respostas em frequência obtidas, e experimente passar o sinal x[n] pelos filtros para comparar as saídas com os sinais desejados. Confira se o sinal observado na saída confere com a resposta em frequência teórica.

Resp: Temos que**:**

const ωp_pb = π/25 # Frequência de passagem para o filtro passa-baixas

const ωr_pb = π/4 # Frequência de rejeição para o filtro passa-baixas

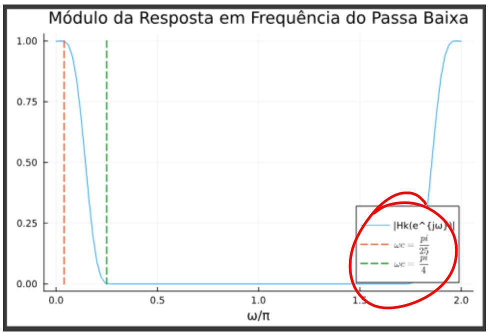
const ωp_pa = π/4 # Frequência de passagem para o filtro passa-altas

const ωr_pa = π/25 # Frequência de rejeição para o filtro passa-altas

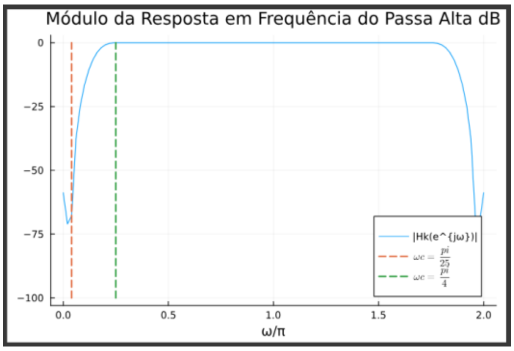
Utilizamos a função filtrokaiser(ωp, ωr, dp, dr) feita em aula.

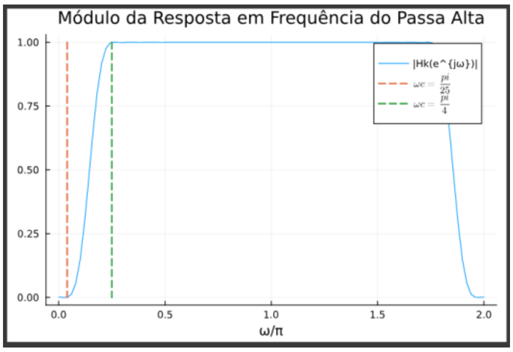
respostas em frequência

Temos abaixo o gráfico dos módulos das funções de transferência dos filtros passa baixa e passa alta. É possível perceber que eles funcionaram visto que estão eliminando as frequências de corte.



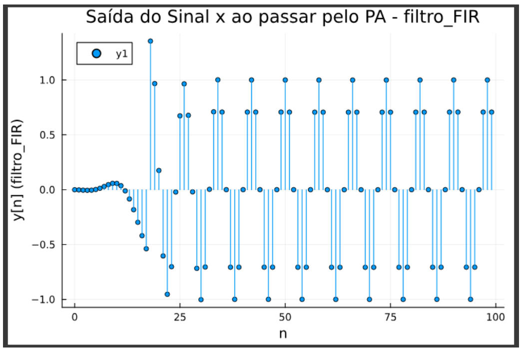
ωp e ωr, não ωc

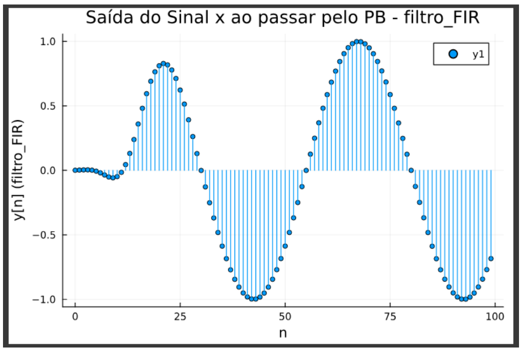




Conferir se o filtro obedece às especificações

Finalmente, passamos o sinal de entrada x[n] pelos filtros e o sinal resultante foi como o esperado, eles resultaram nas respostas em frequência obtidas no item 3(b).



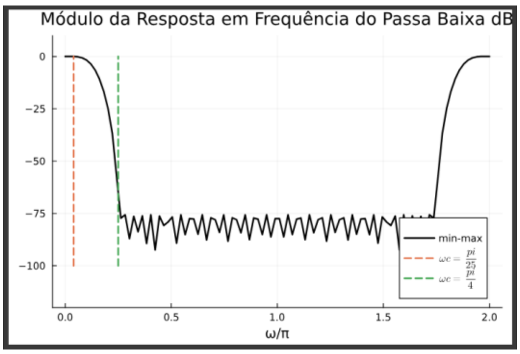


5. Repita o item anterior usando o método min-max de projeto (algoritmo de ParksMcClellan).

Por último, foi utilizado o código feito em aula para min-max, ele está ilustrado abaixo:

```

Nmm = ceil(Int, (-10log10(dp*dr)-13) / (2.324*wd)) + 2 # A fórmula básica soma  # estou somando 2 para # sejam atendidas

hmm = remez(Nmm-1, [(0, wp_pb/pi) => (1, 1), (wr_pb/pi, 1) => (0, dp/dr)]; Hz = 2)
hpbmmf = PolynomialRatio(hmm, [1])
Hpbmm = freqresp(hpbmmf, ω)

plot(ω/pi, amp2db.(abs.(Hpbmm)), label = "min-max",
      xlabel = "ω/π", ylims = (-120, 10),
      lw = 2, color = :black, title = "Módulo da Resposta em Frequência do Passa Baixa dB")
plot!([pi/25,pi/25]/π, [-100,0], lw = 2, label = L"ωc = \frac{pi}{25}", linestyle= :dash)
plot!([pi/4,pi/4]/π, [-100,0], lw = 2, label = L"ωc = \frac{pi}{4}", linestyle= :dash)
```

É possível notar com o módulo da resposta em frequência do filtro passa baixas que o filtro está funcionando e foi capaz de atenuar as frequências superiores a frequência de corte.



→ Conferir as especificações

E o passa-altas?