DÉCIMA AULA PRÁTICA – FILTROS DIGITAIS

G. P. Calais, M. V. R. Campos

UFV, Viçosa, Brasil

E-mails: gabriel.calais@ufv.br, marciovonrondow96@gmail.com

Resumo: O presente relatório trata de uma aula prática sobre filtros digitais no *software MATLAB*. O objetivo da aula foi aprender sobre os filtros e suas finalidades. Para tanto, foram mostradas algumas aplicações dos mesmos, além da proposição de exercícios. Com os exercícios foi possível o aprendizado sobre o assunto.

Palavras-chave: *MATLAB*, filtragem, filtros, digitais, áudio.

Introdução

Sistemas lineares invariantes no tempo são comumente chamados de filtros e o processo de geração de uma saída y[n] a partir de uma entrada x[n] é chamado filtragem. Esse processo tem como objetivo realizar uma alteração do espectro de frequência de um sinal. Um filtro simples pode, por exemplo, ser especificado para remover sinais indesejados (ruídos) acima de uma certa frequência de corte.

A décima aula prática da disciplina de Processamento Digital de Sinais consistiu na apresentação e aplicação de conceitos sobre filtros digitais no *software MATLAB.*

Na prática em questão, o *software* foi usado para a compreensão das aplicações dos filtros citados, além de suas aplicações análise de sinais de áudio e de imagem, através de orientações e utilizando o comando *help*.

Materiais e métodos

Foi utilizado o *software MATLAB* com a finalidade de aprender sobre as aplicações dos filtros digitais no processamento de sinais de áudio.

Alguns dos comandos mais importantes utilizados na aula prática constam na tabela 1.

Tabela 1: Alguns comandos da aula prática.

Comando	Função
subplot	Divide uma figura em vários gráficos.
audioread	Carrega um arquivo de som para a *workspace.*
sound	Reproduz um arquivo de som da *workspace.*
fft	Realiza a transformada rápida de Fourier.
spectrogram	Calcula o espectrograma de um sinal.
plot	Plota um gráfico bidimensional em uma figura.
surf	Plota um gráfico tridimensional em uma figura.
buttord	Define os parâmetros necessários para a criação do filtro Butterworth.
butter	Cria um filtro Butterworth.
freqz	Plota a resposta em frequência de um filtro.

A filtragem é um processo que tem o objetivo de alterar o espectro de frequência de um sinal. O filtro digital é usado para separar componentes de frequências desejados dentro de um sinal. Há diversos filtros digitais implementados pelo *software* MATLAB: *fir1*, *filtfilt*, *cheby2*, *cheb1ord*, *filter*, *butter*, *ellys*, *cheb2ord*, *kaiserord*, *cheby1*, *buttord*, *ellipord*. Cada uma dessas funções possui suas características de filtragem.

A função *fir1* aplica o filtro FIR (resposta ao impulso finita) linear e estável. Ela é baseada no método de janelamento "Hamming" (padrão). Essa função é aplicável em filtros passa-baixas, passa-altas, passa-faixa e rejeita-faixa.

A função *kaiserord* estima parâmetros otimizados para a função *fir1* de forma a atender uma série de especificações na filtragem. Sua implementação no *software* MATLAB é feita da seguinte forma:

$$[n, w_n, beta, ftype] = kaiserord(f, a, dev) \quad (1)$$

A função retorna um filtro de ordem n, bordas da banda de frequência normalizadas w_n e um fator beta que especifica o janelamento a ser usado na função *fir1*. A variável f indica as bordas da banda, a a amplitude da banda e dev o desvio máximo permitido dado em vetor.

A função *filter* aplica uma filtragem de um sinal de entrada usando uma função de transferência racional. A função de transferência é dada por vetores no domínio Z, definida da seguinte forma:

$$Y(z) = \frac{b(1)+b(2)z^{-1}+\cdots+b(n_b+1)z^{-n_b}}{1+a(2)z^{-1}+\cdots+a(n_a+1)z^{-n_a}} X(z) \quad (2)$$

Onde os termos a, b são coeficientes implementados na função e $X(z)$ é a transformada Z do sinal de entrada $x(t)$.

A característica do filtro aplicado é dada pela função *fir1* com resposta de fase original. Por esse motivo não é aplicável em fases não lineares como IIR.

A função *filtfilt* é implementada na presença de filtro IIR (resposta ao impulso infinita). Esse filtro tem uma característica não linear, por isso é necessário o uso da função *filtfilt,* a qual permite realizar filtragens não causais além de apresentar fase zero, o que elimina distorções na própria fase do filtro IIR.

O filtro *Butterworth* é conhecido devido a sua alta taxa de inclinação, sua função de transferência é especificada a seguir:

$$H_n(jw) = \frac{1}{\sqrt{1 + (w/w_c)^{2n}}} \quad (3)$$

Onde n é a ordem do filtro, w a frequência angular em radianos por segundo, w_c é a frequência de corte.

Nesse filtro, quanto maior a ordem, maior é a taxa de inclinação a partir de sua frequência de corte.

Esse filtro é implementado no *software* MATLAB da seguinte forma:

$$[b, a] = butter(n, w_n, ftype) \quad (4)$$

Onde n é a ordem do filtro, w_n é a frequência de corte dada em forma vetorial indicada de acordo com o tipo de filtro a ser aplicado pelo $ftype$, passa-baixa, passa-alta, passa-faixa ou rejeita-faixa.

A função *buttord* definida a seguir:

$$[n, W_n] = buttord(W_p, W_s, R_p, R_s, 's') \quad (5)$$

A função retorna a ordem n mais baixa do filtro *Butterworth*, onde W_p e W_s são respectivamente as frequências de canto da banda passante e a banda de parada normalizadas entre 0 e 1, onde 1 representa π radianos por amostra. R_p é o *ripple* da banda passante, R_s a atenuação da banda de parada e $'s'$ especifica o tipo de filtro.

O filtro *Chebyshev* tipo 1 possui um aumento na atenuação (*roll-off*) e u maior *ripple* na banda passante que os filtros *Butterworth*. Sua função de transferência é indicada a seguir:

$$H_n(jw) = \frac{1}{\sqrt{1 + \epsilon^2 T_n^2(\frac{W}{W_o})}} \quad (6)$$

Onde n é a ordem do filtro, ϵ é o parâmetro de ondulação da banda de passagem, w a frequência angular em radianos por segundo, w_o a frequência de corte e $T_n(\frac{w}{w_o})$ é um polinomial de *Chebyshev.*

Esse filtro é implementado no *software* MATLAB da seguinte forma:

$$[b, a] = cheby1(n, R_p, w_p, ftype) \quad (7)$$

Onde n é a ordem do filtro, R_p a oscilação pico a pico da banda passante, w_p frequência de borda da banda passante e $ftype$ a característica do filtro.

A função *cheb1ord* definida a seguir:

$$[n, W_p] = cheb1ord(W_p, W_s, R_p, R_s, 's') \quad (8)$$

A função retorna a ordem n mais baixa do filtro *Chebyshev* tipo 1, onde W_p é a frequência de canto da banda passante ou frequência de corte, W_s é a frequência de borda de banda de parada, R_p é o *ripple* da banda passante, R_s a atenuação da banda de parada, as duas são dadas em decibéis. Por fim, 's' especifica o tipo de filtro.

O filtro *chebyshev* tipo 2 é um filtro incomum, pois não apresenta um *roll off* acentuado e requer mais componentes. A função *cheby2* é implementada da seguinte forma:

$$[a, b] = cheby2(n, R_s, W_s) \quad (9)$$

Onde n é a ordem do filtro, R_s é a atenuação da banda de parada expressa em decibéis e W_s é a frequência de borda de banda de parada.

A função *cheb2ord* definida a seguir:

$$[n, W_p] = cheb2ord(W_p, W_s, R_p, R_s, 's') \quad (10)$$

A função retorna a ordem n mais baixa do filtro *Chebyshev* tipo 2, onde W_pé a frequência de canto da banda passante ou frequência de corte, W_s é a frequência de borda de banda de parada, R_p é o *ripple* da banda passante, R_s a atenuação da banda de parada, as duas são dadas em decibéis. Por fim, 's' especifica o tipo de filtro.

Já o filtro elíptico é conhecido por apresentar *ripples* na banda passante e na banda rejeitada com o erro minimizado. Seja sua função de transferência a seguir:

$$H_n(jw) = \frac{1}{\sqrt{1 + (\epsilon R_n(w))^2}} \quad (11)$$

Onde n é a ordem do filtro, ϵ é o parâmetro de ondulação da banda de passagem e $R_n(w)$ é a função racional de *Chebyshev*.

Esse filtro é implementado no *software* MATLAB da seguinte forma:

$$[b, a] = ellip(n, R_p, R_s, w_p, ftype) \quad (12)$$

Onde n é a ordem do filtro, R_p é a ondulação de pico a pico da banda passante, R_s a atenuação da banda de parada, w_p é a frequência de borda de banda passante, $ftype$ a característica do filtro.

A função *ellipord* definida a seguir:

$$[n, W_p] = ellipord(W_p, W_s, R_p, R_s, 's') \quad (13)$$

A função retorna a ordem n mais baixa do filtro elíptico, onde W_p e W_s são respectivamente as frequências de canto da banda passante e a banda de parada, R_p é o *ripple* da banda passante, R_s a atenuação da banda de parada e $'s'$ especifica o tipo de filtro.

A partir das descrições dos filtros, analisou-se os resultados obtidos dos exercícios propostos. Criou-se uma senoide contaminada por ruído e em seguida foram aplicados a ela filtros digitais a fim de observar o efeito da resposta gerada, modificando os parâmetros dos filtros e a sua característica.

Além disso, foram processados dois arquivos de áudio contaminados por ruídos. Nesse exercício foi feito uma limpeza no áudio aplicando os filtros digitais e fazendo as devidas análises dos resultados.

Resultados

Após os testes feitos implementou-se o código de programação com suas devidas finalidades. Nas figuras constam os resultados da resolução dos problemas propostos.

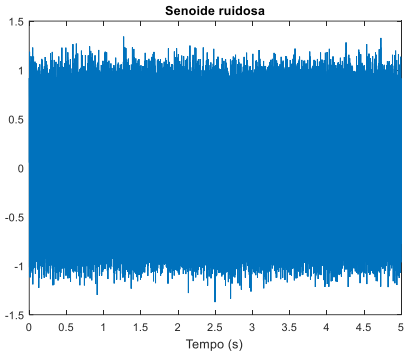

Figura 1: Senoide com ruído.

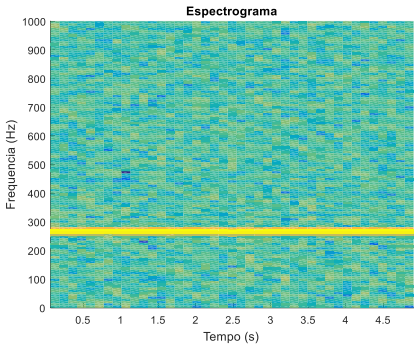

Figura 2: Espectrograma da senoide com ruído.

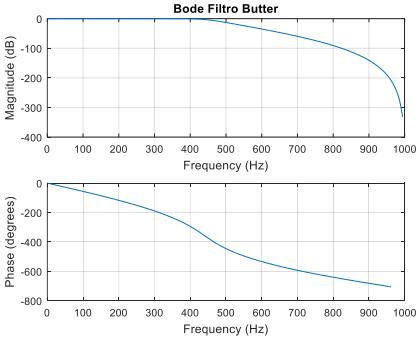

Figura 3: Diagrama de Bode do filtro Butterworth aplicado na senoide ruidosa.

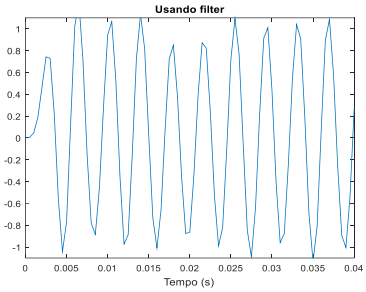

Figura 4: Sinal filtrado usando o comando *filter.*

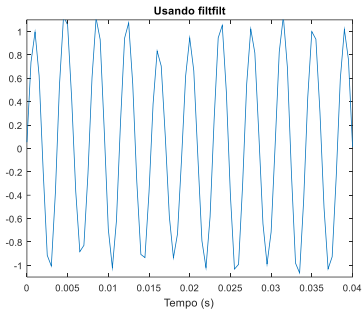

Figura 5: Sinal filtrado usando o comando *filtfilt.*

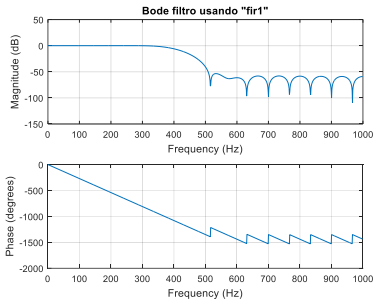

Figura 6: Diagrama de Bode do filtro implementado com a função *fir1*.

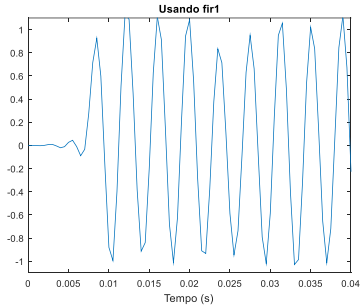

Figura 7: Sinal filtrado com o filtro *fir1*.

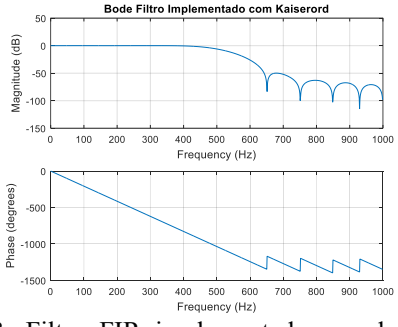

Figura 8: Filtro FIR implementado usando a função *kaiserord.*

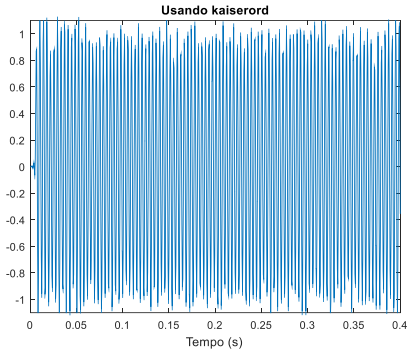

Figura 9: Sinal filtrado usando o filtro *kaiserord*.

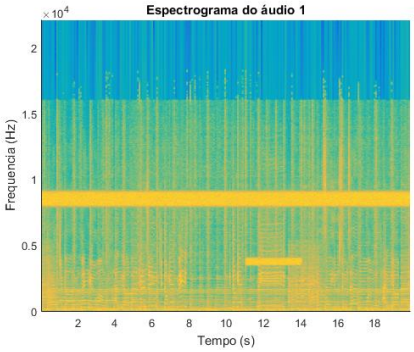

Figura 10: Espectrograma do áudio 1.

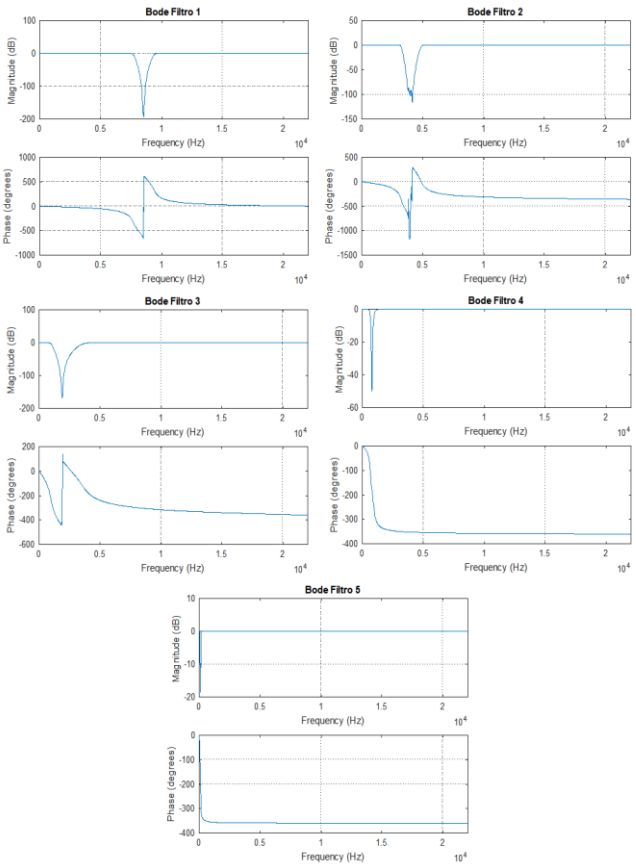

Figura 11: Filtros utilizados para filtrar o áudio 1.

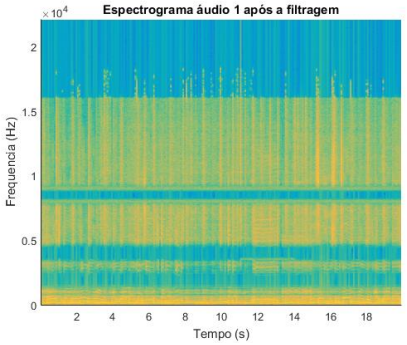

Figura 12: Espectrograma do áudio 1 após a filtragem.

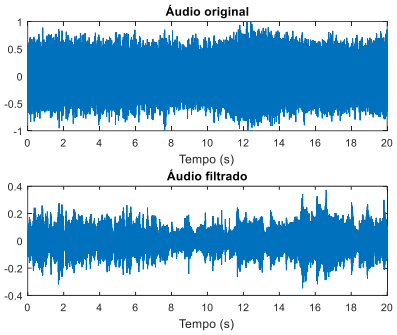

Figura 13: Áudio 1 original e após a filtragem, respectivamente.

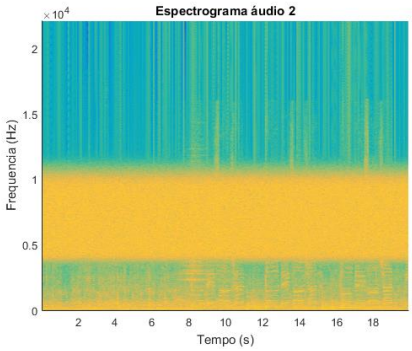

Figura 14: Espectrograma do áudio 2.

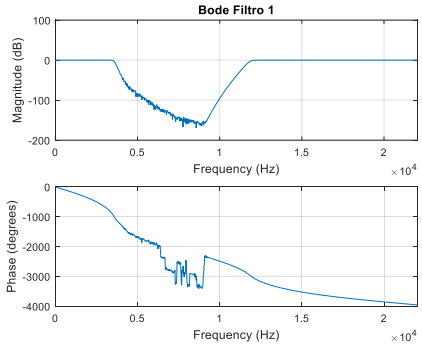

Figura 15: Primeiro filtro utilizado para filtrar o áudio 2.

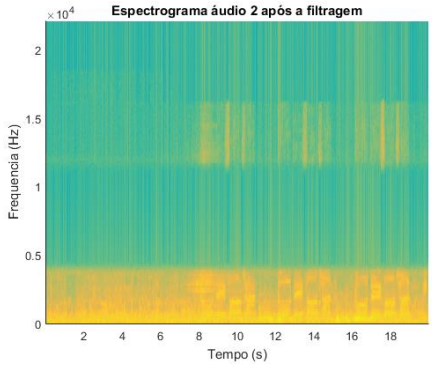

Figura 16: Espectrograma do áudio 2 após a filtragem.

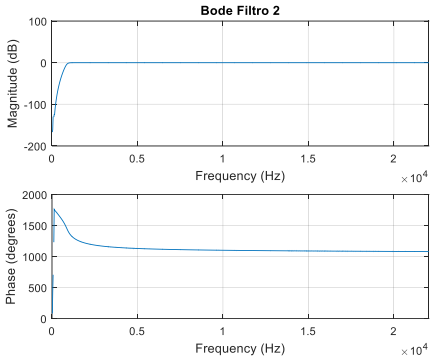

Figura 17: Segundo filtro utilizado para filtrar o áudio 2.

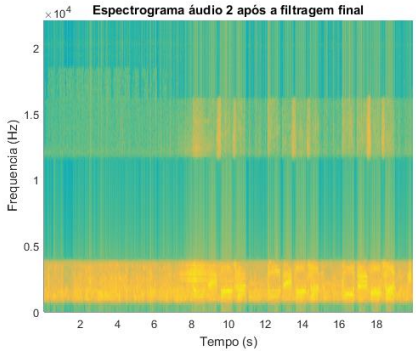

Figura 18: Espectrograma do áudio 2 após a filtragem final.

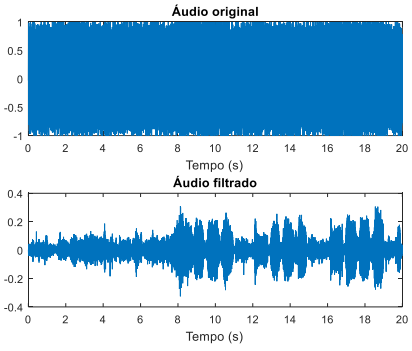

Figura 19: Áudio 2 original e após a filtragem, respectivamente.

Discussão

No primeiro exercício realizou-se a filtragem de um sinal senoidal com ruído utilizando-se, a partir da análise de seu espectrograma, um filtro passa-baixas com frequência de corte de 400 Hz, como pode ser observado nas figuras 1, 2 e 3. Para tanto foram usadas as funções *buttord* e *butter* para criar um filtro Butterworth e os comandos *filter* e *filtfilt* para realizar a filtragem, o que consta nas figuras 4 e 5.

Seguidamente, foi feita novamente a implementação do filtro anteriormente citado, mas utilizando as funções *kaiserord* e *fir1*. Com a utilização do comando *kaiserord* foi possível observar uma otimização da ordem do FIR equivalente ao IIR anteriormente usado, como se pode verificar nas figuras 7 e 9. É possível observar na figura 6 que o filtro FIR apresenta algumas oscilações tanto na sua magnitude quanto em sua fase e, com o uso da função *kaiserord* essas oscilações diminuem, como consta na figura 8.

No exercício seguinte carregou-se dois arquivos de áudio contendo duas músicas com diferentes ruídos. Esse fato foi analisado pelos espectrogramas dos áudios, nos quais interpretou-se as componentes de frequência de maior magnitude como sendo responsáveis pelos ruídos.

Com relação ao primeiro áudio, como é possível observar na figura 10, foram identificados 5 diferentes ruídos, logo, optou-se pela utilização de 5 filtros rejeita-faixa. As frequências de corte foram determinadas a partir da análise do espectrograma e foi possível obter um resultado satisfatório, ou seja, a remoção do ruído foi feita sem danificar a informação que desejava ser preservada, a música. Na figura 12 é possível notar as frequências atenuadas e na figura 13 pode-se verificar as diferenças entre os sinais.

Já para a filtragem do segundo arquivo de áudio, foram encontradas algumas dificuldades. Como se pode verificar na figura 14, o ruído de frequências entre aproximadamente 4000 e 11000 Hz não coincide com as frequências da parte principal da música. Assim, optou-se pela utilização do filtro mostrado na figura 15.

Após a filtragem, observando o espectrograma que consta na figura 16 e ouvindo o áudio, foi possível notar a presença de um ruído presente durante todo o arquivo. O ruído em questão se trata do som de várias pessoas conversando, som este que possui uma faixa de frequências abrangente e que, neste caso, se sobrepõe às frequências mais importantes da música. Assim, não foi possível efetuar a filtragem sem que a informação desejada fosse danificada ou perdida.

Dessa forma optou-se por atenuar uma pequena parte do ruído em questão utilizando-se o filtro mostrado na figura 17, de forma que a filtragem não causou interferência significativa na qualidade do arquivo. Logo, o ruído foi atenuado, porém parte da música também foi, como se pode verificar no espectrograma da figura 18. Na figura 19 é possível ver as diferenças entres os áudios antes e após a filtragem.

Conclusão

Dado o exposto, a prática em questão focou em filtros digitais como ferramentas para filtragem de sinais a partir de seus espectrogramas. De acordo com os resultados obtidos e discutidos, foi possível observar as

aplicações dos filtros digitais como ferramenta de filtragem de sinais. Comparados aos filtros analógicos, os filtros digitais mostram-se mais vantajosos pois não dependem de características de componentes, consegue-se facilmente implementar filtros de ordem elevada, são programáveis via *software,* além de possuírem uma banda larga com transições agudas. Assim, pode-se concluir que este recurso matemático pode ser amplamente utilizado na análise do comportamento de diferentes tipos de sinais, como os de som.

Referências

[1] LATHI, B.P. Sinais e Sistemas Lineares. 2ª edição. Porto Alegre, Bookman, 2007.

[2] Mathworks. Disponível em: <https://www.mathworks.com/help/matlab/>. Acessado em 10/11/2019