Universidade Federal da Bahia - Escola Politécnica - UFBA
ENGC63 - T01 - PROCESSAMENTO DIGITAL DE SINAIS
Alunos(a): Ana Clara Batista, André Paiva Conrado Rodrigues
Trabalho sobre projeto de filtros - Primeiro Projeto - Relatório

1 - Introdução

Dentre os projetos expostos, escolhemos o segundo projeto que consiste em projetar um filtro passa-baixas Butterworth de sexta ordem, um filtro discreto passa-baixa com frequência de corte (-3dB) pré-determinado usando o método de transformação bilinear. Neste trabalho, escolhemos trabalhar com o software livre Octave que utiliza GNU Octave, uma linguagem computacional desenvolvida para computação matemática que possui uma interface em linha de comando para a solução de problemas numéricos.

O áudio escolhido para ser tratado foi um corte do cover de uma música da banda Alt-J produzido por um dos integrantes da equipe, André Paiva, na forma mono.

2- Método de transformação bilinear

A técnica de transformação bilinear usa uma transformação algébrica entre as variáveis s e z, que mapeia todo o eixo j no plano s em uma volta na circunferência unitária no plano z. O uso dessa técnica é restrito a situações em que a deformação não linear correspondente do eixo das frequências é aceitável. Para o uso da transformação linear, aplicado a um projeto de tempo contínuo, as frequências precisam ser pré-deformadas de acordo com a Equação 7.26 [1], explícitas na Figura 1, para que seja mapeado as frequências contínuas de volta para o tempo discreto corretamente. Este cálculo foi realizado diretamente no Octave de acordo com a Figura 2. A frequência de amostragem (fs) é lida a partir do arquivo de áudio utilizado (que no nosso caso é 44,1 kHz).

$$\Omega = \frac{2}{T_d} \operatorname{tg}(\omega/2), \qquad (7.26)$$

ou

$$\omega = 2 \operatorname{arctg}(\Omega T_d/2). \qquad (7.27)$$

Figura 1 - Equações utilizadas para projeto do filtro

```
n = 6; %número de órdens
wc = 2*pi/3; %frequência digital
Td = 1/fs; %Período de amostragem
wn = 2*tan(wc/2)/Td; %frequência analógica transformada para uso na Bilinear
```

Figura 2 - Variáveis utilizadas no Octave

3 - Filtro em forma direta

Primeiramente, implementamos o filtro digital em forma direta por meio da transformação bilinear, utilizando a função bilinear( ) para obter o numerador e o denominador da função de transferência do filtro digital. Utilizando as funções freqz( ) e freqz_plot( ), conseguimos plotar o diagrama de resposta em frequência do filtro implementado, que se mostrou um filtro passa-baixa, como visto na Figura 3.

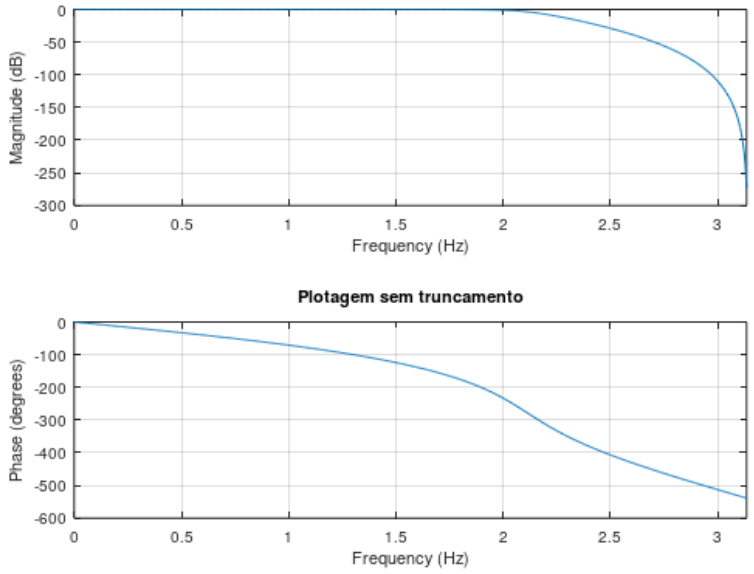

Figura 3 - Resposta em frequência do filtro em forma direta pré-truncamento de coeficientes

Logo depois, fizemos o truncamento dos coeficientes do filtro pós-transformação bilinear, e presenciamos distorções bastante significativas na resposta em frequência do filtro, como é possível notar na Figura 4.

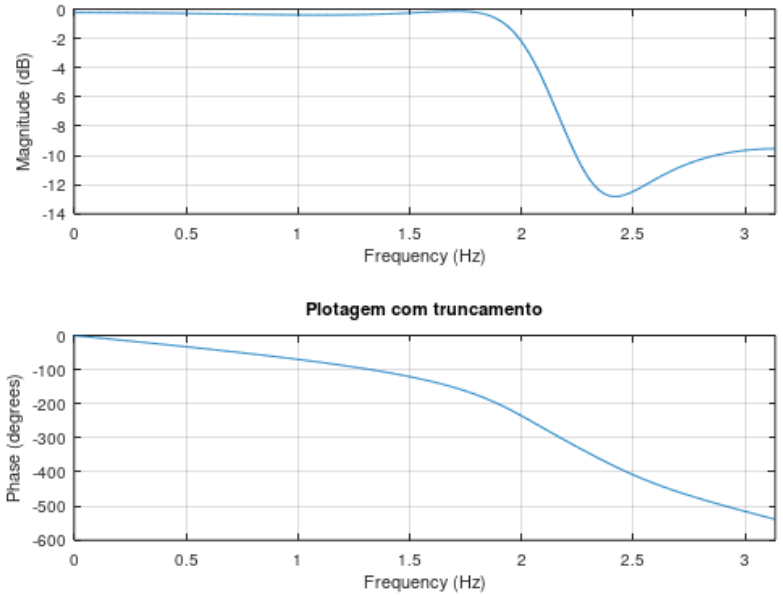

Figura 4 - Resposta em frequência do filtro em forma direta pós-truncamento de coeficientes

Fizemos também a plotagem do espectrograma do sinal de áudio antes e depois da aplicação do filtro (ressaltando que o filtro sem truncamento de coeficientes foi aplicado). Comparando os espectrogramas, vemos facilmente que as frequências mais altas foram atenuadas, como esperado.

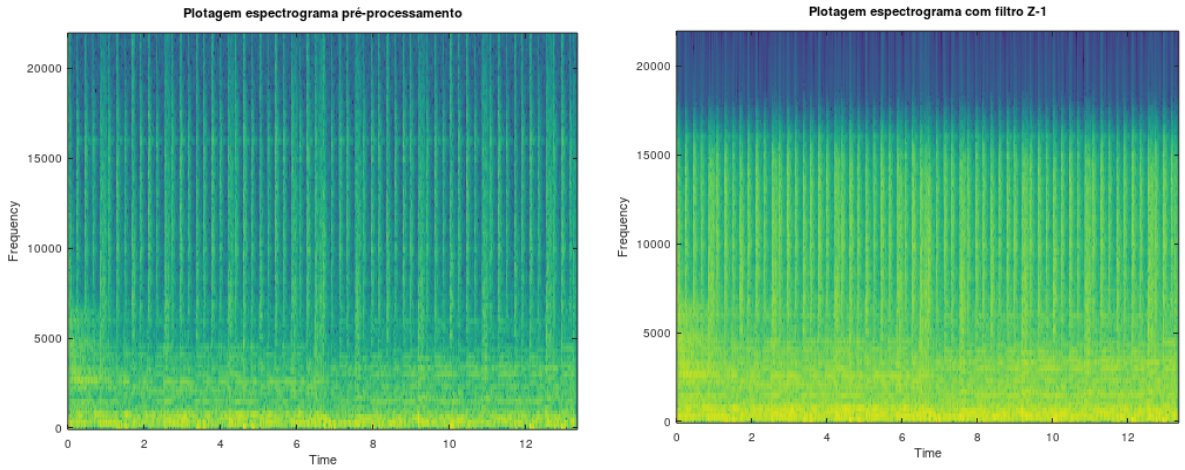

Figura 5 - Espectrogramas do áudio antes e depois da aplicação do filtro passa-baixa

4 - Filtro em forma cascata

Implementamos o filtro digital em forma direta por meio de transformação bilinear, e então convertemos a implementação em forma direta para implementação com filtros de segunda ordem em cascata utilizando a função tf2sos( ). Fizemos o truncamento dos coeficientes dos filtros em cascata e percebemos que a distorção na resposta em frequência não foi tão acentuada como no truncamento dos coeficientes da implementação em forma direta. A comparação entre as respostas em frequência pré e pós-truncamento são vistas na Figura 6.

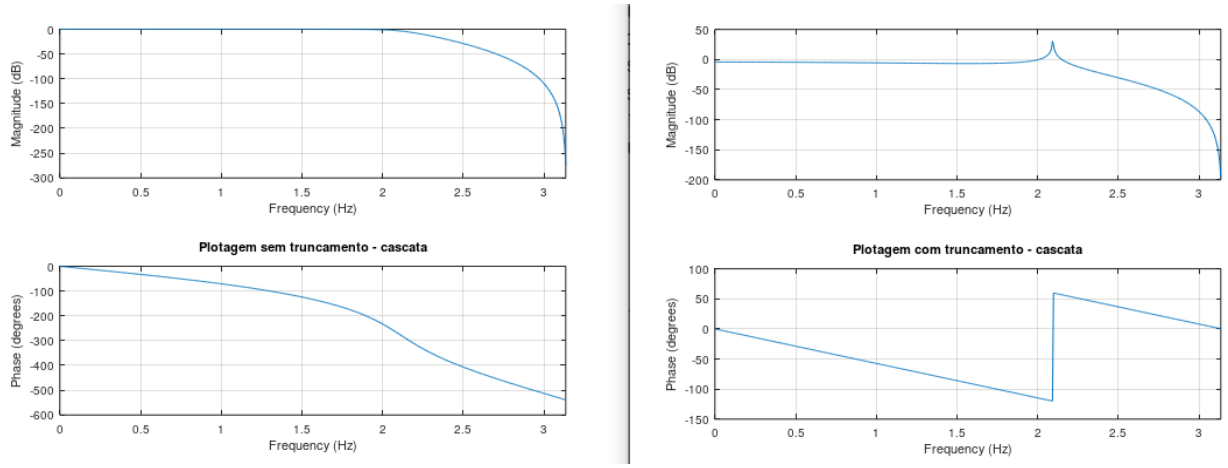

Figura 6 - Resposta em frequência do filtro em forma em cascata pré e pós-truncamento

5 - Transformações na forma direta

Por fim, foi requisitada a manipulação da resposta em frequência do filtro da forma direta em 3 modos: $Z^{-1} = -z^{-1}$, $Z^{-1} = z^{-2}$, $Z^{-1} = -z^{-2}$. Fizemos as manipulações, plotamos a curva da resposta em frequência e utilizamos estes novos filtros para processar o sinal de áudio, gerando um espectrograma e um arquivo de áudio wav para cada filtro.

Primeiramente foi feita a transformação$Z^{-1} = -z^{-1}$, por meio da qual foi gerado um filtro passa-alta. Filtramos o áudio com este novo filtro e encontramos exatamente o esperado: um novo áudio onde as frequências baixas foram cortadas.

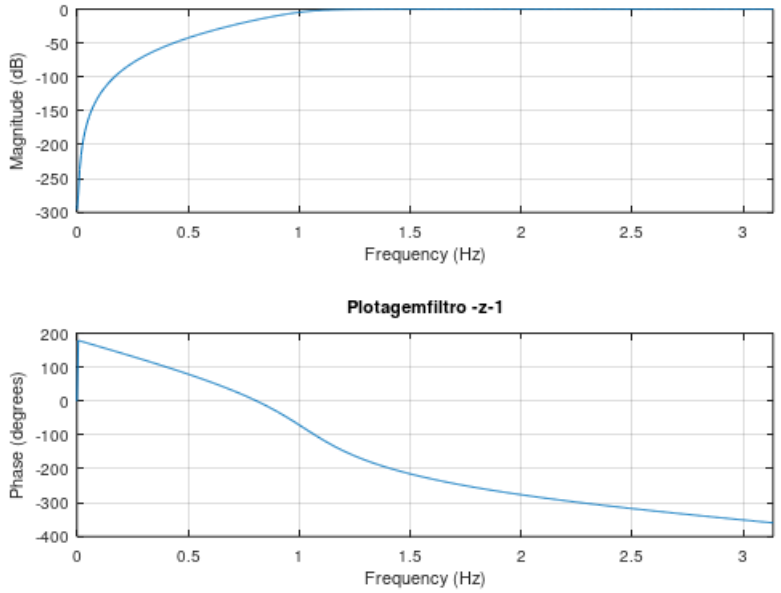

Figura 7 - Resposta em frequência do filtro com a primeira transformação

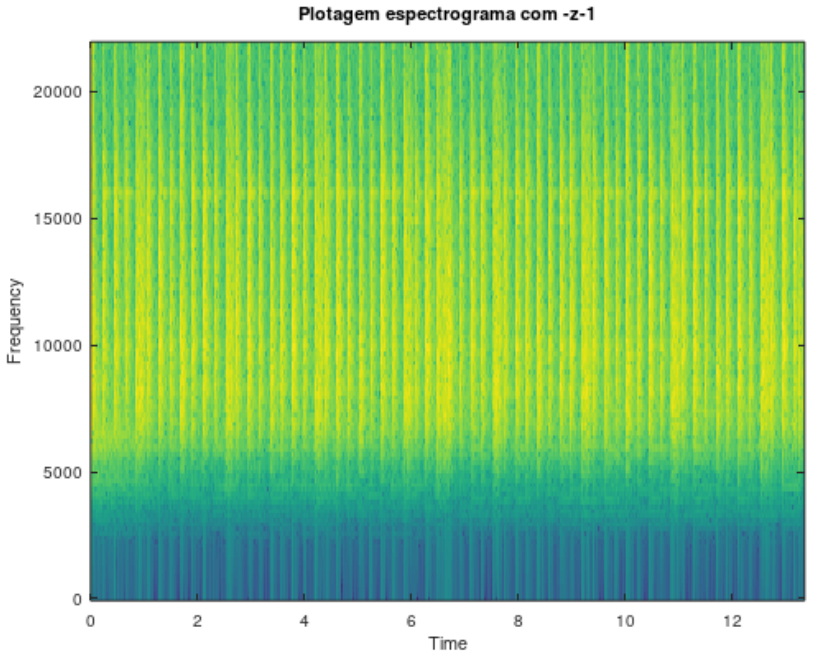

Figura 8 - Espectrograma do áudio após aplicação do filtro passa-alta

Para a segunda transformação foi realizado o translado $Z^{-1} = z^{-2}$, com o qual foi encontrado um filtro rejeita-faixa. Filtramos o áudio através deste novo segundo filtro e encontramos novamente o esperado: um novo áudio onde as frequências medianas foram cortadas.

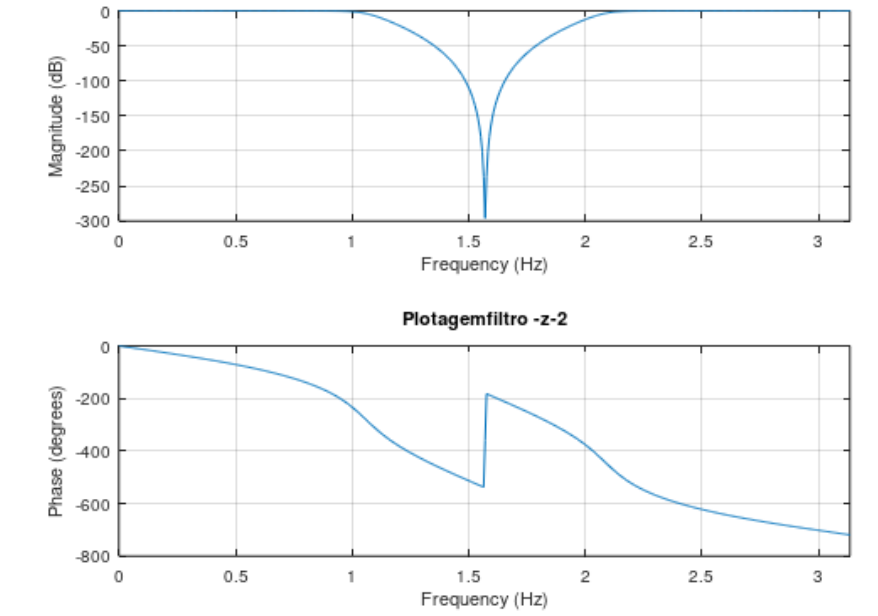

Figura 9 - Resposta em frequência do filtro com a segunda transformação

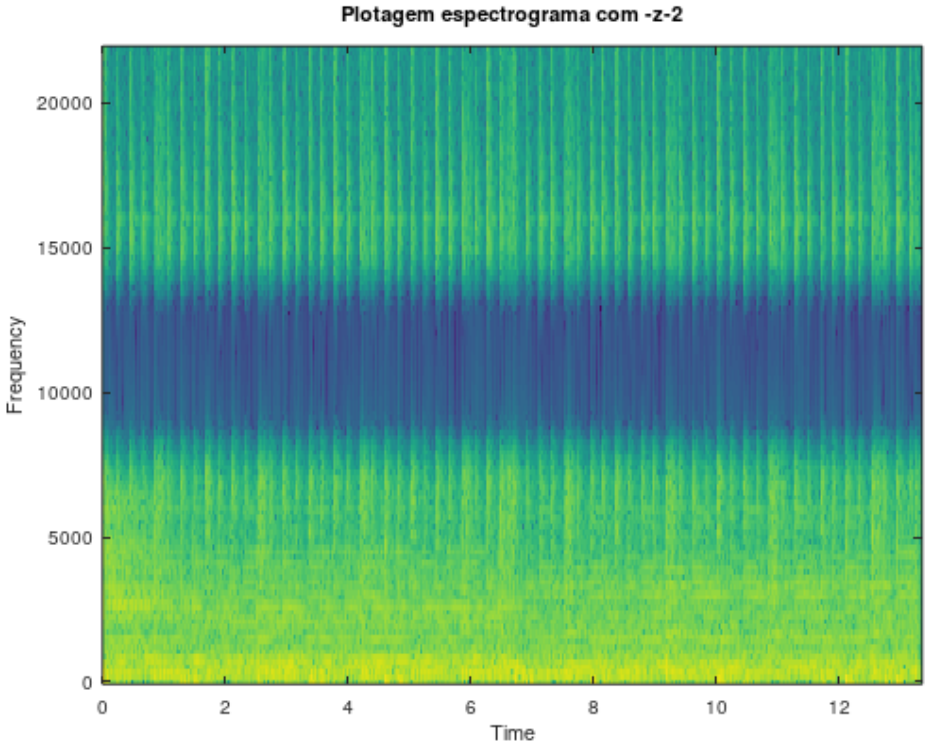

Figura 10 - Espectrograma do áudio após aplicação do filtro rejeita-faixa

Para a terceira transformação foi realizado o translado $Z^{-1} = -z^{-2}$, na qual foi encontrado um filtro passa-faixa. Filtramos o áudio através deste terceiro filtro e presenciamos as frequências baixas e altas sendo atenuadas, como esperado.

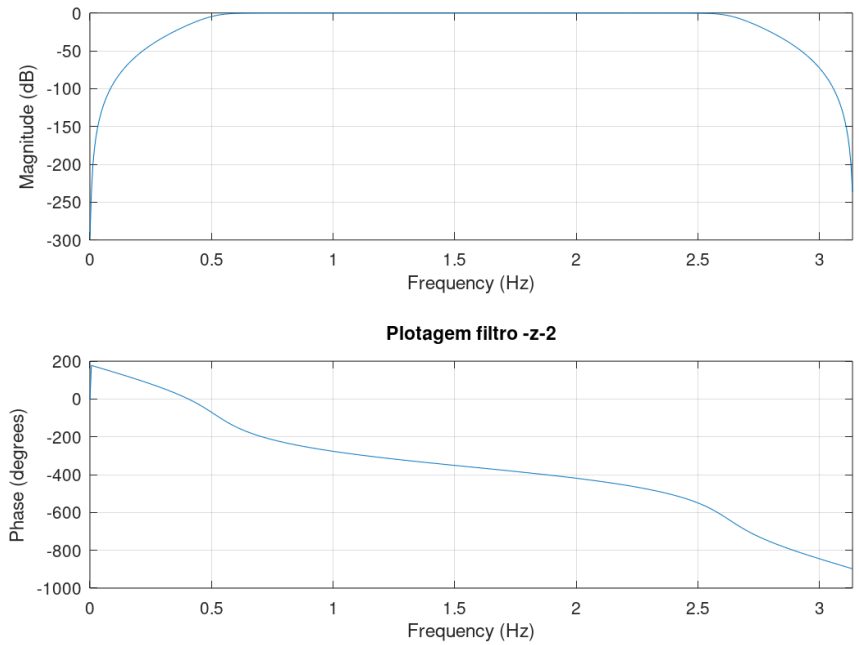

Figura 11 - Resposta em frequência do filtro com a terceira transformação

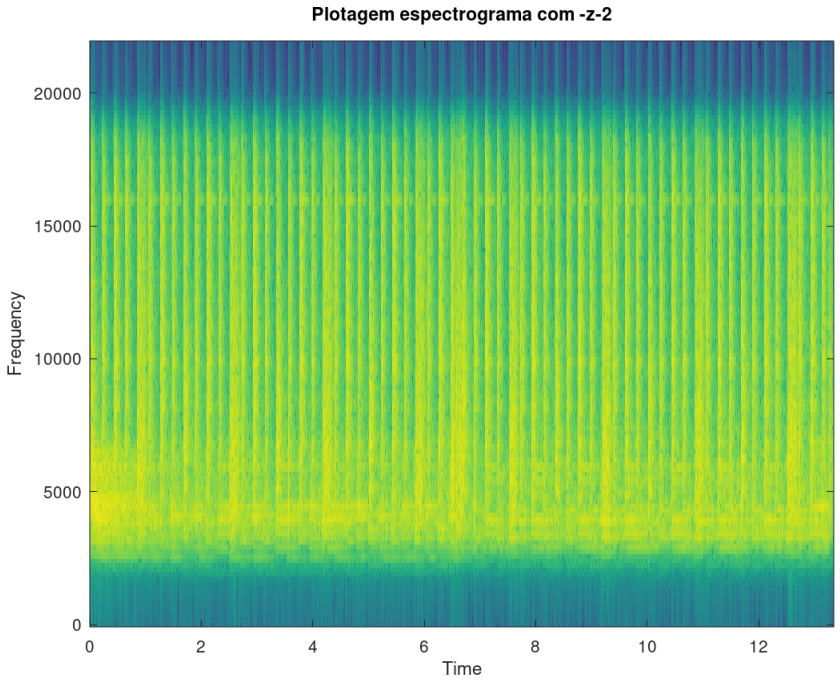

Figura 12 - Espectrograma do áudio após aplicação do filtro passa-faixa

6 - Conclusão

Por fim, conseguimos completar o trabalho de forma favorável, construindo cada filtro e observando seus impactos em cima do áudio utilizado. Todas as observações foram realizadas e o aprendizado sobre a ferramenta Octave e o projeto de filtros foi favorável e bem proveitoso para todos os integrantes da equipe.

Referências

[1] Alan V. Oppenheim, Ronald W. Schafer, John R. Buck: Discrete-Time Signal Processing, Prentice Hall
[2] https://octave.sourceforge.io/ - Documentação do Octave, acessado em Junho/2021