Processamento Digital de Sinais
Tutorial 12 - Filtros de Resposta ao Impulso Finita (FIR)

https://github.com/fchirono/AulasDSP

Objetivo de Aprendizagem

Ao final desta sessão, você será capaz de:

- Projetar e aplicar filtros FIR simples.

Sumário

1 Removendo uma Perturbação de um Sinal de Áudio

Objetivo: Remover um ruído tonal irritante em 4000 Hz de um sinal de música

Tarefas:

1. Leia o arquivo `BetterDaysAheadT.wav` e obtenha seu sinal e frequência de amostragem. Observe que este é um arquivo estéreo: você pode escolher trabalhar com o canal esquerdo (geralmente o primeiro canal) ou com o canal direito, mas não é necessário trabalhar com ambos.
2. Confirme, usando uma DFT, a frequência do tom; sugerimos usar um comprimento de DFT longo (da ordem de 2^{15} pontos) para melhor visualização.
3. Projete um filtro FIR passa-baixas que remova este tom, aplique-o ao sinal de música e ouça os resultados. Experimente com diferentes parâmetros de filtro, e justifique sua escolha final para os parâmetros.
4. Projete um filtro passa-altas que também remova este tom e aplique-o ao sinal de música original. Justifique a escolha final dos parâmetros do filtro.
5. Calcule a razão dos espectros de amplitude dos sinais de saída sobre o espectro de amplitude do sinal de entrada, e confirme que seus filtros alcançaram as respostas em frequência desejadas.
6. Some os sinais passa-baixas e passa-altas e verifique o resultado.
7. Usando a razão dos espectros de saída/entrada descrito no item 5 e usando ruído branco como sinal de entrada, demonstre que a combinação de ambos os filtros alcança um efeito de filtro rejeita-faixa.

2 Simulando um Eco Simples

Tarefas:

1. A partir de princípios básicos (usando análise matemática e, quando necessário, Python) gere a Resposta ao Impulso correspondente a um sinal direto em $t = 0$ s e um eco após $\Delta T = 0.0455$ s.

2. Qual é o efeito de um eco no domínio da frequência? Calcule a resposta em frequência do eco e plote sua magnitude; veja se o resultado corresponde ao que você esperava. (Dica: cuidado com a resolução em frequência que você precisará para observar o efeito!)

3. Aplique esta Resposta ao Impulso à amostra de música, e analisando os espectros de entrada e saída das primeiras `N_DFT` amostras, confirme que você obtém a resposta em frequência esperada.

4. Variando o atraso, avalie a mudança na amostra de áudio. Neste contexto, encoraja-se a leitura sobre filtros tipo "*comb*" (filtro em pente)!

3 OPCIONAL: Reflexão Simples de uma Parede

Calcule a resposta ao impulso para a configuração representada na Figura 1. Assuma uma lei de decaimento da pressão irradiada pela fonte em função da distância d do centro da fonte, dada por $A(d) = P_1/d$. A parede é considerada perfeitamente rígida, e tanto a fonte quanto o microfone estão alinhados em uma linha perpendicular à parede. Considere os seguintes parâmetros e quantidades:

- $f_s = 44100$ Hz;
- $c_0 = 343$ m/s;
- $R = 0.1$ m;
- a pressão equivalente na superfície da fonte é $P_1 = 0.1$ Pa·m;
- Distância do centro da fonte até a parede: 4 metros;
- Distância do microfone até a parede: 1.8 metros.

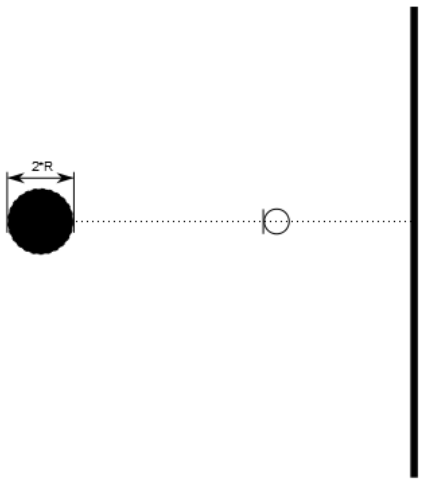

Figura 1: Fonte esférica e microfone em frente a parede rígida.

4 OPCIONAL: Cálculo de Resposta ao Impulso e Resposta em Frequência de Filtro

Calcule as Respostas ao Impulso e em Frequência dos seguintes filtros FIR:

1. Filtro de média móvel:
$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k],$$
com a soma sobre $M = 3$, $M = 5$ e $M = 10$ pontos. Compare os resultados dos diferentes filtros.

2. Filtro de diferença:
$$y[n] = x[n] - x[n-1].$$

Compare seus resultados com aqueles da função `scipy.signal.freqz` (veja Apêndice).

5 Apêndice: Projeto de filtro FIR em Python

Para projeto de filtro FIR, usaremos a biblioteca `scipy.signal`. O vetor `b` dos coeficientes do filtro pode ser obtido via método da janela usando o seguinte código:

```
import numpy as np
from scipy import signal

# obter os coeficientes do filtro
b_lowpass = signal.firwin(N_taps, wn, pass_zero=True)
b_highpass = signal.firwin(N_taps, wn, pass_zero=False)

# criar um vetor de coeficientes 'a' com o mesmo comprimento que 'b'
a_lowpass = np.zeros(b_lowpass.shape)
a_highpass = np.zeros(b_highpass.shape)

# apenas o elemento zero de 'a' eh unitario
a_lowpass[0] = 1.
a_highpass[0] = 1.
```

onde

- `b_lowpass`, `b_highpass` são os vetores contendo os coeficientes do numerador da função de transferência dos filtros em potências ascendentes de z^{-1} (ou seja, o numerador é escrito como $b[0] + b[1] \cdot z^{-1} + b[2] \cdot z^{-2} + \ldots$);
- `a_lowpass`, `a_highpass` são os vetores contendo os coeficientes do denominador da função de transferência, também em potências ascendentes de z^{-1}. Embora filtros FIR tenham denominador unitário, o SciPy tem melhor desempenho quando tanto numerador quanto denominador têm o mesmo número de coeficientes, daí o vetor de zeros com um 1 como a primeira amostra;
- `N_taps` é o número de elementos no numerador;
- `wn` é a frequência de corte do filtro normalizada para *metade* da frequência de amostragem;
- e o parâmetro `pass_zero` determina se a frequência zero está incluída na banda de passagem ou não. Por exemplo, um filtro passa-baixas incluiria a frequência zero em sua banda de passagem, enquanto um filtro passa-altas excluiria a frequência zero de sua banda de passagem.

Uma vez que temos o vetor de coeficientes do filtro, podemos obter a resposta em frequência do filtro em `N` pontos sobre o vetor de frequência normalizada `w` $\in [0, \pi)$ usando a função `scipy.signal.freqz`:

```
# calcular a resp em freq dos filtros em 'N' pontos sobre as frequencias 'w'
w, H_lowpass = signal.freqz(b_lowpass, a_lowpass, N)
```

Finalmente, um sinal `x` pode ser processado com um filtro definido por seus coeficientes de função de transferência (`b`, `a`) usando a função `scipy.signal.lfilter`:

```
# filtrar sinal 'x' com coeficientes de filtro 'b' e 'a'
x_filtered_lp = signal.lfilter(b_lowpass, a_lowpass, x)
```

As funcionalidades descritas neste Apêndice serão suficientes para completar o Tutorial; no entanto, essas funções têm muitos outros parâmetros opcionais que não cobrimos, o que as torna muito flexíveis. Por favor, leia a documentação para as funções acima mencionadas no Guia de Referência de Processamento de Sinais do SciPy[1] e certifique-se de se familiarizar com suas implementações.

[1] URL: http://docs.scipy.org/doc/scipy/reference/signal.html