ELT 355 - Laboratório de sinais e sistemas

Aula Prática 12: Filtros Digitais

Departamento de Engenharia Elétrica
Centro de Ciência Exatas e Tecnológicas, Universidade Federal de Viçosa

1 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. O objetivo da filtragem é realizar uma alteração do espectro de freqüência de um sinal. Por exemplo, um filtro simples pode ser especificado para remover sinais indesejados (ruídos) acima de uma certa freqüência de corte. Um projeto mais sofisticado pode partir da especificação dos níveis de oscilação permitidos na banda de passagem (Rp, em decibéis) ou da largura da banda de transição (Wp - Ws, em Hertz). Já uma abordagem mais precisa pode ainda exigir que os objetivos sejam alcançados com filtro de ordem mínima, fase linear etc.

1.1 Filtros FIR

Filtros digitais com resposta ao impulso finita (FIR ou *all-zeros*) são obtidos truncando-se a resposta ao impulso do filtro (ideal) pretendido. Filtros FIR possuem, dentre outras, as seguintes vantagens:

- Fase linear.
- Sempre são estáveis.
- Podem ser implementados facilmente em hardware.

O comando `fir1` implementa filtros FIR passa-alta, passa-baixa, passa-faixa etc, baseado no método do janelamento. O janelamento Hamming é usado como *default*, mas o comando aceita outros tipos de janelas. Adicionalmente, a função `kaiserord` estima parâmetros otimizados para a função `fir1` de forma a atender uma série de especificações de filtragem. A filtragem propriamente dita é realizada com o comando `filter`.

1.2 Filtros IIR

A principal vantagem de filtros digitais com resposta ao impulso infinita (IIR) é o fato de se alcançarem as especificações de projeto com ordem bem menor que os filtro FIR. Embora estes filtros apresentem fase não-linear, o processamento digital permite que o processo de filtragem seja realizado *off-line*, o que torna possível a realização de um filtragem não-causal e que apresente fase zero (comando `filtfilt`) e, portanto, elimina as distorções de fase do filtro IIR.

Os filtros IIR clássicos - Butterworth, Chebyshev, Elíptico - estão implementados no matlab (comandos `butter, cheby1, cheby2, ellip`), onde os coeficientes são obtidos a partir das especificações de cada filtro. A função `filter` filtra um dado vetor com o filtro especificado pelas funções anteriores porém com a resposta de fase original do filtro (é recomendável nesse caso usar o comando `filtfilt` ao invés do `filter` para eliminar a distorção em função de fase não-linear do filtro IIR). É ainda possível otimizar os filtros em função das características de resposta em freqüência desejadas. Nesse sentido, os comandos `buttord, cheb1ord, cheb2ord, ellipord` são responsáveis pela obtenção da ordem e dos parâmetros adicionais dos filtros otimizados.

2 Roteiro

2.1 Exercício 1: Filtro digital passa-baixa 400 Hz

Inicialmente é necessário normalizar as freqüências em função da freqüência de Nyquist, a qual é metade da freqüência de amostragem. Todas as funções da *toolbox* de processamento de sinais do Matlab trabalham com freqüências normalizadas, pelo fato de não necessitarem, assim, de um parâmetro de entrada extra que seria a freqüência de amostragem. A freqüência normalizada está sempre no intervalo entre 0 e 1. Por exemplo, com 1000 Hz de freqüência de amostragem, 300 Hz é $300/500 = 0,6$.

Criemos um senóide ruidosa:

⇒*Acesse o* `help` *das funções e comente todas as linhas de código!*

```
fs = 2000;
t = 0:1/fs:5;
s = sin(2*pi*262.62*t);
n = 0.1*randn(size(s));
sn = s + n;
```

Obtendo os parâmetros ótimos para um filtragem Butterworth passa-baixa que atenue, pelo menos, 35 dB na banda de corte e que atenue, no máximo, 1 dB na banda de passagem além de ter banda de transição entre 400 a 600 Hz:

```
[N, Wn] = buttord(400/(fs/2), 600/(fs/2), 1, 35)
[B,A] = butter(N,Wn,'low')
```

Vejamos a resposta em freqüência deste filtro (note a fase não-linear).

```
freqz(B,A,1024,fs)
```

Filtrando com `filter`

```
y = filter(B,A,sn);
figure;plot(t,y);axis([0 0.04 -1.1 1.1]);title('Usando filter')
soundsc(y,fs)
```

Filtrando com `filtfilt`

```
y = filtfilt(B,A,sn);
figure;plot(t,y);axis([0 0.04 -1.1 1.1]);title('Usando filtfilt')
soundsc(y,fs)
```

Descreva as diferenças percebidas entre os arquivos filtrados, caso existam.

Alternativamente, pode-se implementar um filtro equivalente através de uma abordagem FIR. Por exemplo, para um passa-baixa FIR de ordem 20 e corte em 400 Hz:

```
B = fir1(30,400/1000)
freqz(B,1,1024,2000)
y = filter(B,A,sn);
figure;plot(t,y); axis([0 0.04 -1.1 1.1])
soundsc(y,fs)
```

Tente agora otimizar a ordem do FIR equivalente ao IIR acima. Use, por exemplo, a função `kaiserord`.

2.2 Limpeza de áudio

Acesse o PVAnet e projete filtros para limpar os arquivos 1.wav e 7.wav. Não é permitido usar informações dos áudios originais na solução. Comente as dificuldades na filtragem de cada um dos sinais, as soluções encontradas e a qualidade sonora dos arquivos processados.