ELT 576

Atividade Prática: filtragem digital

Especialização em Inteligência artificial e Computacional

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 frequência de um sinal. Por exemplo, um filtro pode ser especificado para remover sinais indesejados (ruídos) acima de uma certa frequência 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).

2 Roteiro

2.1 Tutorial 1: Recuperação de sinal de interesse em meio a ruído

Os filtros digitais podem ser usados para extrair informação útil em meio de ruído. Isso também pode ser visto como uma forma de realçar uma informação - no caso espectral - em detrimento de outras. Vamos inicialmente criar uma senóide e vamos chama-la de "sinal de interesse".

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

```
close all, clc;;
pkg load signal;
fs = 2048;
t = 0:1/fs:5;'
s = sin(2*pi*262*t);
```

O sinal de interesse, contido na variável `s`, possui amplitude 1 e frequência 262 Hz. Note que a frequência de amostragem está de acordo com o teorema da amostragem. Dê uma olhada nos instantes iniciais desse sinal

```
figure;
stem(t,s);
axis([0 0.04 -30 30]);
title('Sinal de interesse');
xlabel('Tempo (s)')
```

Normalmente os sinais de interesse são coletados imersos em ruído. Vamos simular essa situação adicionando um ruído do tipo gaussiano ao sinal de interesse. Note a amplitude bem mais elevada da mistura sinal+ruído quando comparada apenas ao sinal.

```
ruido = load('ruido.txt');
x = s + ruido;
figure;
stem(t,x);
```

```
axis([0 0.04 -30 30]);
title('Sinal + ruído');
xlabel('Tempo (s)')
```

Vejamos o espectro do sinal+ruído. Vamos usar o comando "pwelch", o qual executa internamente o algoritmo da trasnformada de Fourier e tem várias opções (- dê um `help pwelch` e confira). Note que o espectro do sinal de interesse está acompanhado de ruídos em todas as faixas de frequências possíveis nesse caso.

```
nfft = 1024;
overlap = .5;
[spectra,freq] = pwelch(x,nfft,overlap,nfft,fs,'half','db');
figure;plot(freq,spectra)
title('Espectro do sinal + ruído');
xlabel('Frequência (Hz)')
```

Nos interessa aqui extrair apenas a informação contida na frequência do sinal de interesse. Assim, projetaremos um filtro digital passa-faixa, com frequência de centro igual à frequência do sinal de interesse. Neste exemplo, projetaremos um filtro do tipo butterworth de quarta ordem. Note que as frequências de projeto são ponderadas por metade do valor da frequência de amostragem.

```
f1 = 260;
f2 = 264;
n = 4;
[B, A] = butter (n, [f1/(fs/2) f2/(fs/2)]);
```

Projetar um filtro digital significa obter os conjuntos de parâmetros A e B vistos no código acima. De posse desses parâmetros do filtro podemos verificar a sua resposta em frequência. Podemos verificar que o filtro projetado atenua todas as frequência em torno de 262 Hz. Quanto mais longe uma frequência estiver da frequência de centro, mais ela será atenuada.

```
figure; freqz(B,A,nfft,fs);
```

Depois que confirmamos que o filtro projetado está estável e de acordo com as especificações, vamos filtrar o sinal+ruído para extrair apenas o sinal de interesse.

```
y = filtfilt(B,A,x);
```

Plotemos agora uma amostra do sinal filtrado. Pode-se notar que o sinal de interesse foi recuperado do meio ruidoso.

```
figure;plot(t,y);
axis([0 0.04 -30 30]);
title('Sinal de interesse recuperado');
xlabel('Tempo (s)')
```

2.2 Tutorial 2: Eliminação de informação espectral indesejada

Em diversas situações na análise de sinais e sistemas precisamos eliminar ruídos de nossos bancos de dados. Caso esses ruídos não sejam eliminados ou pelo menos minimizados, as aplicações posteriores - como ajuste de modelos de inteligência artificial - podem ficar comprometidas ou até inviabilizadas. Os filtros digitais podem ser usados para eliminar informações indesejadas de dados reais. Veremos aqui um exemplo tutorial de eliminação de ruído

de microfonia de sinais de áudio.

Inicialmente, vamos carregar uma amostra de sinal de áudio contaminada por microfonia e ouvi-la. A microfonia aparece como esse zumbido constante ao longo de toda a amostra.

```
clear all, close all, clc;
pkg load signal
[Y, FS] = audioread ('bird2fil.wav');
soundsc(Y,FS)
```

Vamos dar uma olhada no espectro do áudio a ser limpo. Pode-se ver nitidamente uma amplitude exagerada - quando comparada com o resto do espectro - em 5000 Hz. Essa é a "assinatura"espectral da microfonia.

```
nfft = FS/2;
overlap = .5;
[spectra,freq] = pwelch(Y,nfft,overlap,nfft,FS,'half','db');
figure;plot(freq,spectra)
title('Espectro do sinal + ruído');
xlabel('Frequência (Hz)')
```

A partir daqui temos dois caminhos para limpar o áudio: uma filtragem passa-baixas abaixo de 5000 Hz ou um filtro rejeita-faixa sintonizado em 5000 Hz. Vamos começar pelo mais simples, que é o filtro passa-baixas. O código abaixo projeta um filtro passa-baixas do tipo butterworth de quarta ordem e com frequência de corte em 4000 Hz. Confira a resposta em frequência.

```
fbaixa = 4000;
n = 4;
[B,A] = butter(n,fbaixa/(FS/2),'low');
figure; freqz(B,A,nfft,FS);
```

Vamos agora filtrar, ouvir e olhar o espectro do sinal filtrado pelo passa-baixas. A experiência sonora agora não tem microfonia significativa, porém é um pouco mais "abafada". Isso ocorre porque o passa-baixas retirou, além da microfonia em 5000 Hz, praticamente toda a informação de alta frequência acima de 4000 Hz. As informações de alta frequência são relacionadas com esse "ajuste fino"ou "brilho"da música.

```
Ypb = filtfilt(B,A,Y);
[spectra,freq] = pwelch(Ypb,nfft,overlap,nfft,FS,'half','db');
figure;plot(freq,spectra)
soundsc(Ypb,FS)
```

Como o ruído que queremos eliminar - microfonia em 5000 Hz - possui um espectro muito estreito, então podemos obter resultados mais seletivos usando um filtro rejeita-faixa. Este tipo de filtro também é chamado de filtro *notch*. Segue abaixo o projeto de um filtro digital rejeita-faixa, do tipo butterworth, de quarta ordem e sintonizado em torno de 5000 Hz. A resposta em frequência confirma a característica de seletividade em frequência desse filtro.

```
f1 = 4930;
f2 = 5070;
[B, A] = butter (n, [f1/(FS/2) f2/(FS/2)],'stop');
figure; freqz(B,A,nfft,FS);
```

Filtremos então o áudio com esse novo filtro. Vamos verificar o espectro obtido e ouvir o resultado da filtragem. A microfonia foi minimizada sem a perda dos detalhes da música original. Uma desvantagem desse tipo de filtro é que ele demanda um pouco mais de tempo para sintonizar e possui mais coeficientes - o que o torna mais lento, computacionalmente falando.

```
Ynotch = filtfilt(B,A,Y);
[spectra,freq] = pwelch(Ynotch,nfft,overlap,nfft,FS,'half','db');
figure;plot(freq,spectra)
soundsc(Ynotch,FS)
```