1D DATA - FILTERING

Introduction

Signal Processing Toolbox™ provides functions and apps that let you design, analyze, and implement a variety of digital FIR and IIR filters, such as lowpass, highpass, and bandstop. Visualize magnitude, phase, group delay, impulse, and step responses. Examine filter poles and zeros. Evaluate filter performance by testing stability and phase linearity. Apply filters to data and remove delays and phase distortion using zero-phase filtering.

Signal Processing Toolbox also provides functions that let you design and analyze analog filters. Perform analog-to-digital filter conversion using discretization methods such as impulse invariance and the bilinear transformation. This example shows how to design, analyze, and apply a digital filter to your data.

Compensating for Constant Filter Delay

Consider a noisy electrocardiogram signal that you want to filter to remove high frequency noise above 75 Hz. You want to apply an FIR lowpass filter and compensate for the filter delay so that the noisy and filtered signals are aligned correctly and can be plotted on top of each other for comparison.

```
Fs = 500;                % sample rate in Hz
N = 500;                 % number of signal samples
rng default;
x = ecg(N)'+0.25*randn(N,1); % noisy waveform
t = (0:N-1)/Fs;          % time vector

% Design a 70th order lowpass FIR filter with cutoff frequency of 75 Hz.

Fnorm = 75/(Fs/2);       % Normalized frequency
df = designfilt('lowpassfir','FilterOrder',70,'CutoffFrequency',Fnorm);

grpdelay(df,2048,Fs)   % plot group delay
D = mean(grpdelay(df)) % filter delay in samples

y = filter(df,[x; zeros(D,1)]); % Append D zeros to the input data
y = y(D+1:end);                  % Shift data to compensate for delay

figure
plot(t,x,t,y,'r','linewidth',1.5);
title('Filtered Waveforms');
xlabel('Time (s)')
legend('Original Noisy Signal','Filtered Signal');
grid on
axis tight
```

Removing Unwanted Spectral Content from a Signal

Filters are commonly used to remove unwanted spectral content from a signal. You can choose from a variety of filters to do this. You choose a lowpass filter when you want to remove high frequency content, or a highpass filter when you want to remove low frequency content. You can also choose a bandpass filter to remove low and high frequency content while leaving an intermediate band of frequencies intact. You choose a bandstop filter when you want to remove frequencies over a given band.

Consider an audio signal that has a power-line hum and white noise. The power-line hum is caused by a 60 Hz tone. White noise is a signal that exists across all the audio bandwidth

You can first remove as much white noise spectral content as possible using a lowpass filter. The passband of the filter should be set to a value that offers a good trade-off between noise reduction and audio degradation due to loss of high frequency content. Applying the lowpass filter before removing the 60 Hz hum is very convenient since you will be able to downsample the band-limited signal. The lower rate signal will allow you to design a sharper and narrower 60 Hz bandstop filter with a smaller filter order.

Design a lowpass filter with passband frequency of 1 kHz, and stopband frequency of 1.4 kHz. Choose a minimum order design.

```
Fs = 44100; % Sample rate
y = audioread('noisymusic.wav');

[P,F] = pwelch(y,ones(8192,1),8192/2,8192,Fs,'power');
```

```
helperFilterIntroductionPlot1(F,P,[60 60],[-9.365 -9.365],...
  {'Original signal power spectrum', '60 Hz Tone'})

Fp = 1e3;    % Passband frequency in Hz
Fst = 1.4e3; % Stopband frequency in Hz
Ap = 1;      % Passband ripple in dB
Ast = 95;    % Stopband attenuation in dB

% Design the filter
df = designfilt('lowpassfir','PassbandFrequency',Fp,...
                'StopbandFrequency',Fst,'PassbandRipple',Ap,...
                'StopbandAttenuation',Ast,'SampleRate',Fs);

% Analyze the filter response
hfvt = fvtool(df,'Fs',Fs,'FrequencyScale','log',...
  'FrequencyRange','Specify freq. vector','FrequencyVector',F);

% Filter the data and compensate for delay
D = mean(grpdelay(df)); % filter delay
ylp = filter(df,[y; zeros(D,1)]);
ylp = ylp(D+1:end);

close(hfvt)

[Plp,Flp] = pwelch(ylp,ones(8192,1),8192/2,8192,Fs,'power');
helperFilterIntroductionPlot1(F,P,Flp,Plp,...
  {'Original signal','Lowpass filtered signal'})
```

Downsample the lowpass filtered signal by a factor of 10 to obtain a sample rate of Fs/10 = 4.41 kHz. Plot the spectrum of the signal before and after downsampling.

```
Fs = Fs/10;
yds = downsample(ylp,10);

[Pds,Fds] = pwelch(yds,ones(8192,1),8192/2,8192,Fs,'power');
helperFilterIntroductionPlot1(F,P,Fds,Pds,...
  {'Signal sampled at 44100 Hz', 'Downsampled signal, Fs = 4410 Hz'})
```

Now remove the 60 Hz tone using an IIR bandstop filter. Let the stopband have a width of 4 Hz centered at 60 Hz. We choose an IIR filter to achieve a sharp frequency notch, small passband ripple, and a relatively low order. Process the data using **filtfilt** to avoid phase distortion.

```
% Design the filter
df = designfilt('bandstopiir','PassbandFrequency1',55,...
                'StopbandFrequency1',58,'StopbandFrequency2',62,...
                'PassbandFrequency2',65,'PassbandRipple1',1,...
                'StopbandAttenuation',60,'PassbandRipple2',1,...
                'SampleRate',Fs,'DesignMethod','ellip');

% Analyze the magnitude response
hfvt = fvtool(df,'Fs',Fs,'FrequencyScale','log',...
  'FrequencyRange','Specify freq. vector','FrequencyVector',Fds(Fds>F(2)));

ybs = filtfilt(df,yds);

yf = interp(ybs,10);
Fs = Fs*10;

[Pfinal,Ffinal] = pwelch(yf,ones(8192,1),8192/2,8192,Fs,'power');
close(hfvt)
helperFilterIntroductionPlot1(F,P,Ffinal,Pfinal,...
  {'Original signal','Final filtered signal'})
```

Listen to the signal before and after processing. As mentioned above, the end result is that you have effectively attenuated the 60 Hz hum and the high frequency noise on the audio file.

```
% Play the original signal
hplayer = audioplayer(y, Fs);
play(hplayer);

% Play the noise-reduced signal
hplayer = audioplayer(yf, Fs);
play(hplayer);
```

Listen to the signal before and after processing. As mentioned above, the end result is that you have effectively attenuated the 60 Hz hum and the high frequency noise on the audio file.

Instruction

1. Read the following explanation above!
2. Please download the demo file and run it!
   You can download the demo file on **https://intip.in/DemoFilter**
3. Make a short report about the simulation results!

After you read the explanation above and run *Demo.m* , please answer the question below!

1. How do you compensate for the delay introduced by a filter?
2. How do you avoid distorting your signal?
3. How do you remove unwanted content from your signal?
4. Describe the function of FIR and IIR filter according to simulation!

Write the answer on your short report!

2D DATA - FILTERING

There are three commonly discussed filters in the frequency domain:

- Lowpass filters, sometimes known as smoothing filters
  A low pass filter is the basis for most smoothing methods. An image is smoothed by decreasing the disparity between pixel values by averaging nearby pixels. Using a low pass filter tends to retain the low frequency information within an image while reducing the high frequency information.
- Highpass filters, sometimes known as sharpening filters
  A high pass filter is the basis for most sharpening methods. An image is sharpened when contrast is enhanced between adjoining areas with little variation in brightness or darkness. A high pass filter tends to retain the high frequency information within an image while reducing the low frequency information. The kernel of the high pass filter is designed to increase the brightness of the center pixel relative to neighboring pixels.
- Notch filters, sometimes known as band-stop filters
  Notch flters are are used to remove repetitive "Spectral" noise from an image are like a narrow highpass filter, but they "notch" out frequencies other than the dc component attenuate a selected frequency (and some of its neighbors) and leave other frequencies of the Fourier transform relatively unchanged. Repetitive noise in an image is sometimes seen as a bright peak somewhere other than the origin. You can suppress such noise effectively by carefully erasing the peaks. One way to do this is to use a notch filter to simply remove that frequency from the picture. This technique is very common in sound signal processing where it is used to remove mechanical or electronic hum, such as the 60Hz hum from AC power. Although it is possible to create notch filters for common noise patterns, in general notch filtering is an ad hoc procedure requiring a human expert to determine what frequencies need to be removed to clean up the signal.

Instruction

1. Please download the demo file!
   You can download the demo file on **https://intip.in/DemoFilter2D**
2. Run the LPF.m, HPF.m, and BSF.m respectively!
3. Make a report about the simulation results!