MSc Laboratory Experiment MB1 Digital Filter Design

Background

This experiment is concerned with the design and implementation of discrete time digital filters. It will look at a number of different filter design methods and compare the performance of the resultant filters.

To do this experiment you will need to download two toolboxes:

1. Download http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.zip into a temporary location and extract its contents into a folder of your choice, e.g. H:\voicebox. The web page at http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html gives more information about the routines.
2. Download http://www.ee.ic.ac.uk/hp/staff/dmb/courses/msc_lab/mb1_experiment.zip into a temporary location and extract its contents into another folder of your choice, e.g. H:\mb1_experiment. A copy of this labsheet will be saved in the doc subfolder.
3. In MATLAB click on "Set Path" in the "Environment" tab and add the two toolboxes to your path, e.g. H:voicebox and H:\mb1_experiment\matlab.

Test Signal Generation

Run the MATLAB script mb1_demo (appended to this lab sheet). This script creates a test signal and then filters it with a Butterworth filter. In the description below "{n}" refers to line n of the script. Variables and code fragments use Arial font.

The test signal, y, consists is the sum of a clean signal, x, and added noise, v. The clean signal, x, is the sum of sine waves at 80 and 100 Hz; these are specified in {6} as xfa=[80 0.5; 100 -1i] where each row of xfa gives the frequency and complex phasor amplitude of a sine wave component. The added noise, v, consists of two sine wave components at 210 and 280 Hz {7} together with white noise at -8 dB relative to the average power of the clean signal {8}. The function mb1_testsig {10} creates the clean, noise and noisy signals; each signal is of 4 seconds duration {9} and is sampled at fs=4 kHz {5}. The upper plot of Figure 1 shows a 0.1 second portion of the clean signal, x, in blue and the test signal, y, in red.

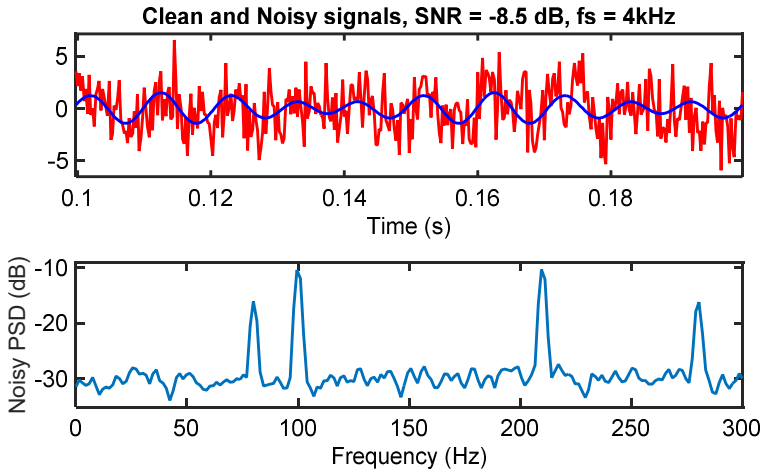

Figure 1: The upper plot shows the clean signal (in blue) and the noisy test signal (in red). The lower plot shows the power spectrum of the test signal.

The lower plot of Figure 1 shows a portion of the 2-sided average power spectral density of the test signal (i.e. energy per Hz) which is plotted by mb1_plotpsd {20}; we only plot the range 0 to 300 Hz although the full spectrum covers ± 2 kHz. The purpose of the filters that you will design in this experiment is to preserve the clean signal components within the range 80 to 100 Hz while attenuating all the other frequencies.

Exercise 1: *(a) Calculate the theoretical average power of the clean, noise and noisy signals and hence the SNR of the test signal. (b) Compare these with theoretical values with the actual average powers of* x, v *and* y *and explain why they are not the same. (c) Why is the noise floor in the lower plot of Figure 1 so much less than the noise power? (d) Calculate the SNR that would be obtained if you filtered the test signal with (i) an ideal lowpass filter with cutoff frequency 100 Hz and (ii) an ideal bandpass filter with pass band 80 to 100 Hz. These represent the best possible SNRs than we can obtain from filtering. In practice the lowpass or bandpass filters will have a stopband attenuation that is less than ∞ dB. (e) For both the above cases, calculate the stop band attenuation that would result in an SNR within 0.1 dB of the ideal target. There is little benefit in designing filters with a stopband attenuation much greater than this.*

Butterworth Filter

The script mb1_demo also designs and evaluates a Butterworth lowpass filter. The filter design parameters are the passband ripple, rp {30}, the stopband attenuation, rs {31} and the frequency range of the transition between passband and stopband, ftr {32}. The MATLAB routine butterord {33} calculates the filter order required, n1, and the filter cutoff frequency, wn1, at which the gain is -3 dB. The routine butter {34} then calculates the numerator and denominator filter coefficient vectors, b1 and a1. Note that the filter design routines in MATLAB normalize frequencies by dividing them by 0.5fs rather than by fs as everyone else in the world does; this is very weird and a frequent source of errors {33}.

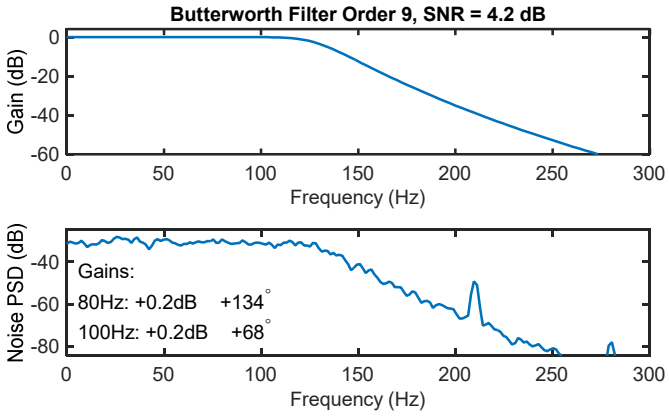

Figure 2: The upper plot shows the magnitude response of a Butterworth filter. The lower plot shows the power spectrum of the filtered test signal as well as showing the gain and phase shift applied to each of the clean signal frequency components.

Figure 2 shows the magnitude response {43} of the resultant filter which, as the title {47} indicates, has an order, n1, of 9. The filter is applied to the test signal, y, {35} and then the routine mb1_snrtone {36} estimates the SNR of the filtered signal, z1. It does this by first estimating the amplitude and phase of each of the test signal's tonal components using cross correlation and then subtracting these tones to leave just the noise components. The lower plot {39} in Figure 2 shows the power spectrum of this residual noise component, v1. The routine mb1_snrtone also estimates the gain and phase shift, e1, that the filter applied to each of the passband tonal components. These values are printed on the power spectrum plot by the somewhat cryptic but useful {40}.

Exercise 2: *(a) What is the unnormalized cutoff frequency corresponding to* wn1*? (b) Explain the values of the power spectrum plot at 50, 150 and 210 Hz. (c) Plot the phase response of the filter (the MATLAB function* unwrap *is helpful) (d) What is the group delay of the filter at low frequencies; how is this related to the pass band phase shifts? (e) How many multiplications per sample would be needed to implement this filter?*

Elliptic Filter

An elliptic filter has a much narrow transition region than a Butterworth for a given filter order which it achieves with multiple z-domain zeros on the unit circle.

Exercise 3: *Use the* ellipord *and* ellip *MATLAB functions to design a suitable elliptic filter. Determine the appropriate order and the resultant SNR. Plot the z-domain poles and zeros of the filter. How do these relate to the filter's magnitude spectrum that you observe and to the noise power spectrum? Can you explain the unwrapped phase spectrum graph? Estimate the number of multiplications per sample needed for this filter. What are the advantages and disadvantages of an Elliptic filter over a Butterworth filter?*

Exercise 4: *Design an elliptic filter of order 14 (using the same* wn *frequency as in Exercise 3) and assess its performance. A more stable implementation of a high-order IIR filter is as a sequence of second order filters (also known as biquads) because lower order filters are less sensitive to coefficient roundoff errors. Use the three output-argument form:* [z,p,k]=ellip(…) *to find the zeros and poles and then use* zp2sos *and* sosfilt *to implement the filter. Compare its performance with the direct implementation.*

Window Filter Design

The previous filters have been IIR which allows rapid transitions in the frequency domain with relatively few coefficients. An alternative is to use an FIR filter which typically requires many more coefficients but can have exactly linear phase (if it is symmetric) and is always stable. There are two common ways to design an FIR filter: the window method gives a closed form solution that is quite close to optimum while the iterative Parks-McLellan algorithm designs a filter that has the minimum worst case deviation from the target magnitude response.

Exercise 5: *Use* kaiserord *and* fir1 *to design a suitable FIR low pass filter and assess its performance. How high an order do you need to match the SNR performance of the elliptic filter? Plot the filter impulse response and the pole-zero diagram. What are the advantages and disadvantages of this filter over the elliptic filter?. How many multiplications per sample are required? How can you take advantage of the symmetry of the filter impulse response?*

Exercise 6: *Use* firpmord *and* firpm *to design an optimal FIR filter and assess its performance.*

Polyphase filter implementation

Since the output of the filter is bandlimited, it is possible to subsample by a factor of K without losing information. In the diagram $H(z)$ is and FIR lowpass filter (from exercise 5 or 6) which is applied twice. In between the signal is downsampled by K and then upsampled by K to restore the original sample rate. Finally, the signal is multiplied by K to restore the original signal level.

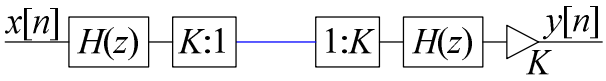

Figure 3: The input signal, $x[n]$, is lowpass filtered by $H(z)$, downsampled by a factor K, upsampled by K, lowpass filtered again and finally multiplied by K.

As covered in lectures the total number of multiplications per sample needed to implement this is $1 + 2(\mathrm{M} + 1)/\mathrm{K}$ where $H(z)$ is of order M.

Exercise 7: *Calculate the appropriate value of K so that the new Nyquist frequency lies close to the centre of the transition band (i.e. 150 Hz). Implement the diagram using the filter from Exercise 6 as H(z). The effect of downsampling and then upsampling is equivalent to setting K-1 out of every K samples to zero and, for a signal* x *of length* N*, can neatly be done in MATLAB with the command* x(mod(0:N-1,K)>0)=0*. Calculate how many multiplications are now needed per sample. Can you explain the amplitude and frequency of all the peaks in the residual noise power spectrum?*

Optional Extensions

See if you can improve the output SNR by, for example, using a bandpass instead of a lowpass filter. How high an SNR can you obtain assuming that the entire signal range between 80 and 100 Hz must be preserved unchanged? See what happens if you have a higher sampling frequency (e.g. 20 kHz).

Mike Brookes
27 September 2016

```
close all; clear all;   % clear all plots and variables
%
% create the test signal
%
fs=4000;               % sample frequency
xfa=[80 0.5; 100 -1i];  % signal frequencies and phasor amplitudes
vfa=[210 1; 280 0.5];   % tonal noise frequencies and phasor amplitudes
snr=-8;                % white noise SNR in dB
nt=round(4*fs);        % 4 seconds worth of samples
[y,t,x,v]=mb1_testsig(xfa,vfa,snr,nt,fs); % y=test signal, t=time axis, x=clean signal, v=noise signal
snr0=mb1_snrtone(y,xfa,fs); % Find the SNR of the noisy signal
%
% plot the signal and its power spectrum
%
fplot=300;             % max frequency to plot
fax=linspace(0,fplot,100); % frequncy axis for magnitude responses
iplot=0.1*fs:0.2*fs;    % samples to plot
figure(1);
subplot(2,1,2);
mb1_plotpsd(y,fplot,fs);  % plot PSD of noisy signal
ylabel('Noisy PSD (dB)');
subplot(2,1,1);
plot(t(iplot),y(iplot),'-r',t(iplot),x(iplot),'-b');     % plot time waveforms
axisenlarge([-1 -1.05]);               % make axes fit the plot
title(sprintf('Clean and Noisy signals, SNR = %.1f dB, fs = %.2gkHz',snr0,fs/1000));
xlabel('Time (s)');
%
% now design a butterworth IIR filter
%
rp=0.1;                % target passband ripple (dB)
rs=35;                 % target stopband attenuation (dB)
ftr=[100 200];         % transition frequency range: 100 to 200 Hz
[n1,wn1]=buttord(2*ftr(1)/fs,2*ftr(2)/fs,rp,rs); % determine order and f0
[b1,a1]=butter(n1,wn1); % design a Butterworth filter
z1=filter(b1,a1,y);     % filter the noisy signal, y
[snr1,ax1,e1,v1]=mb1_snrtone(z1,xfa,fs);  % find the filtered SNR, gain errors and residual noise
figure(2);
subplot(2,1,2);
mb1_plotpsd(v1,fplot,fs);          % plot PSD of residual noise
texthvc(0.02,0.1,['Gains:' sprintf('\n%.0fHz: %+.1fdB \\angle%+.0f^\\circ',[xfa(:,1) e1(:,1) e1(:,2)*180/pi]')],'LBk');
ylabel('Noise PSD (dB)');
subplot(2,1,1);
plot(fax,20*log10(abs(freqz(b1,a1,fax*2*pi/fs)))); % plot the magnitude response
axis([fax(1) fax(end) -60 4]);      % limit the gain range to -60 dB
xlabel('Frequency (Hz)');
ylabel('Gain (dB)');
title(sprintf('Butterworth Filter Order %d, SNR = %.1f dB',n1,snr1));
%
tilefigs([0 0.5 0.8 0.5]);   % display all the figures in the top half of the screen
```