SP.718 Special Topics at Edgerton Center: D-Lab Health: Medical Technologies for the Developing World
Spring 2009

Notes on Matlab processing of vital signs data

D-lab Health
Feb 2009

Brian Tracey

Data Acquisition

- Plug an audio cable from the output jack on the heart sounds monitor to the laptop ‘mic in’
- Use the software tool of your choice to make a recording
  - Audacity is a good choice: other options are Mac ‘SimpleSound’ or Windows Sound Recorder
- Save the recording as a .wav file
- Repeat this process but transmit the heart sounds over the baby monitor (to add noise)
- Some things to try or think about:
  - Experiment with holding your breath while recording, so lung sounds will not be present. Does this make a big difference?
  - Does background noise in the room make a big difference?
  - If you do deep breathing (1 breath / 10 sec) while recording, can you still hear the heart sounds?

Loading / saving wav files in Matlab

- The ‘wavread’ command will load a .wav file
  - for example, for file ‘hbeat.wav’:
    [hbData,fSamp]=wavread(‘hbeat’);
  - this command returns the heart sounds data in a vector ‘hbData’, and the sampling rate used by the recording in ‘fSamp’
- If your software gave 2 channels (stereo recording), throw one away as the stethoscope is mono
- You can save and load your Matlab workspace using ‘save’ and ‘load’ commands
- After processing the data, you may want to save the output waveform into another .wav file
  - For example, to save the vector ‘hbFilt’ into a file ‘filteredHeartbeat.wav’:
    wavwrite(hbFilt, fSamp, ‘filteredHeartbeat’);

Plotting and playing back sound in Matlab

- Based on your sampling rate, set up a vector ‘t’ of times that correspond to each sample
- Then, you can plot the data: plot(t,hbData)
  - plot(hbData) will plot the data without a time axis
- You can play the sounds using the ‘sound’ or ‘soundsc’ commands:

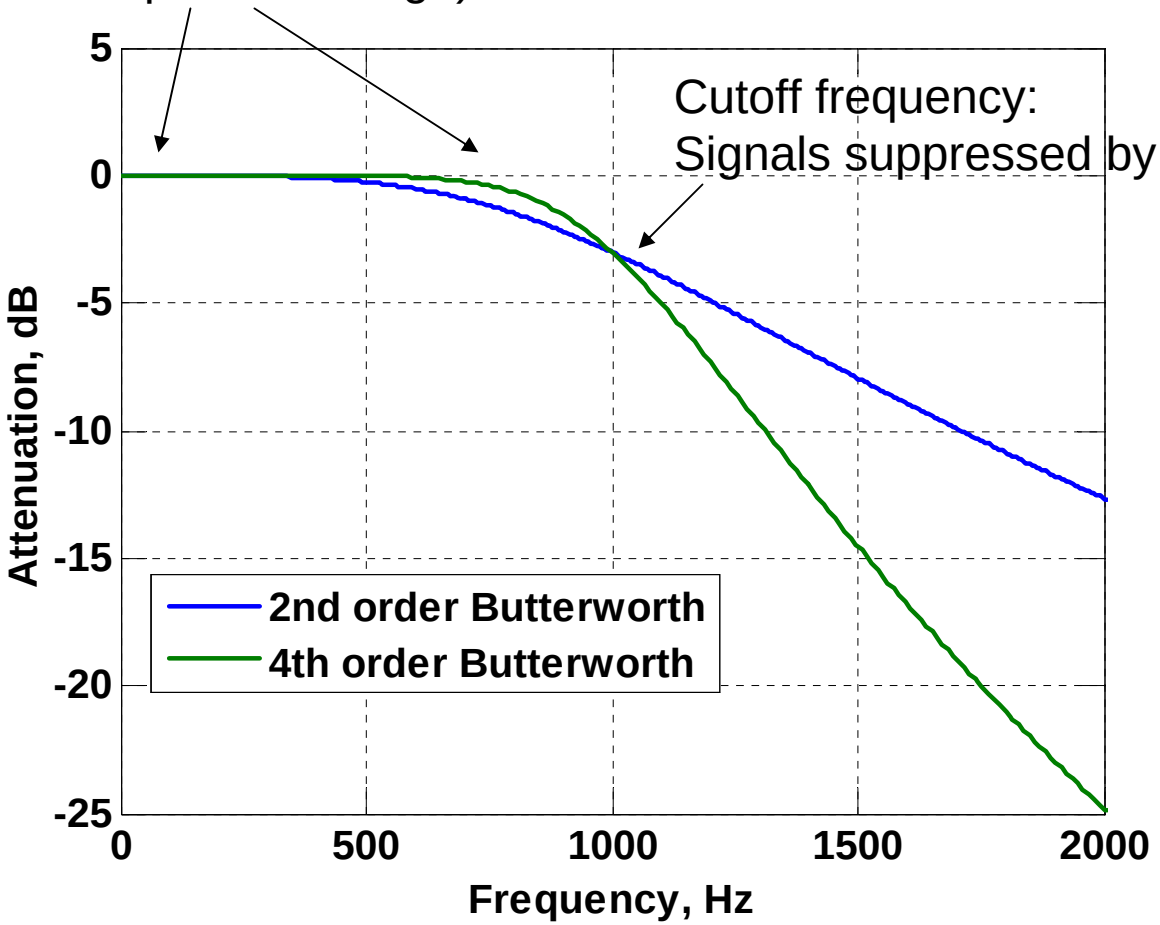
```
sound(hbData, fSamp); % plays whole recording
sound(hbData(1:fSamp*5), fSamp); % plays first 5 sec
```

Filtering the data

- Heart sounds are low frequency, while noise may be higher-frequency – which suggests we try filtering
- Butterworth filters are a common choice for biomedical applications
  - Parameters are filter cutoff frequencies and filter order (higher order -> more suppression); see next page
- For convenience, functions ‘LPfilterWrapper.m’ and ‘BPfilterWrapper.m’ are included at end of this PPT
  - Example call for a 500 Hz lowpass filter is:
    filteredHB = LPfilterWrapper(hbData,fSamp,500);

Example of Butterworth filter parameters

Passband: < 3 dB suppression
(0 dB = no amplitude change)



Higher-order filter has more suppression of high frequencies

- Main lab involves using filtering to clean up signals
- For some additional ideas, see following slides -

Additional challenge #1: Downsampling the data

- During sampling, the analog signals are sampled at a frequency = 2x the highest frequency of interest (http://www.dspguide.com/ch3/2.htm for why)
- The sound recorder on your laptop samples at a rate which is appropriate for music, but heart sounds are lower in frequency
- Question: how much can you downsample the acquired data to reduce the file size, without losing information? How could this help in a telemedicine application?

Additional challenge 1, con't

- You need to filter out any high-frequency noise before reducing the sampling rate, or it will distort your signal
  - Option 1: you can use the filtered data from before, then just discard alternate samples

    ```
    hbNew = hbFilt(1:2:end);  % discards every 2nd sample
    fsNew = fSamp/2;
    ```

  - Option 2: use the Matlab command 'resample', which internally applies filtering

    ```
    hbNew = resample(hbData,1,3);  % reduces rate by 1/3
    fsNew = fSamp/3
    ```

- Check your work by plotting and replaying the new signals*. Do you hear/see significant differences in the signal?
- You can save the output as a .wav file to see the file size reduction*

* Remember to use the new sampling rate in any function calls

Some more ideas

- Heart-rate: can you devise an algorithm for estimating heart-rate from the signals you've acquired?
- Background noise: did conversations or other noises in the room cause problems for your recording?
  - Do you think that may be a problem in a health clinic?
  - If so, do any of your filter approaches help with the problem? Are there other solutions (mechanical, DSP) that you can imagine?

Useful matlab codes

Low-pass filter wrapper code

```
function y = LPfilterWrapper(data, fSamp, fCutoff, nOrder)
% function y = LPfilterWrapper(data, fSamp, fCutoff, nOrder)
% does LOW-PASS filtering of an input signal 'data' using a Butterworth
% filter
% Inputs:
%      data - vector of input data
%      fSamp - sampling rate of input data, Hz
%      fCutoff - desired lowpass cutoff frequency, Hz
%      nOrder (optional) - filter order. If not specified, defaults to '2'

% set default filter order if needed
if nargin ==3,
   nOrder = 2;
end

% normalize cutoff frequency by sample rate
Wn = fCutoff/(fSamp/2);

% get filter coefficients
[b,a]=butter(nOrder,Wn,'low');

% filter the input data
y = filter(b,a,data);

return
```

Band-pass filter wrapper code

```
function y = BPfilterWrapper(data, fSamp, fCutoffLow, fCutoffHi, nOrder)
% function y = BPfilterWrapper(data, fSamp, fCutoffLow, fCutoffHi,, nOrder)
% does BAND-PASS filtering of an input signal 'data' using a Butterworth
% filter; pass band is from
% Inputs:
%      data - vector of input data
%      fSamp - sampling rate of input data, Hz
%      fCutoffLow - desired lowpass cutoff frequency, Hz
%      fCutoffHi - desired hipass cutoff frequency, Hz
%      nOrder (optional) - filter order. If not specified, defaults to '2'

% set default filter order if needed
if nargin ==3,
   nOrder = 2;
end

% normalize cutoff frequenciesby sample rate
W1 = fCutoffLow/(fSamp/2);
W2 = fCutoffHi/(fSamp/2);

% get filter coefficients
[b,a]=butter(nOrder,[W1 W2]); % two cutoff frequencies means bandpass

% filter the input data
y = filter(b,a,data);

return
```