Inflection Area Ratio

Table of Contents

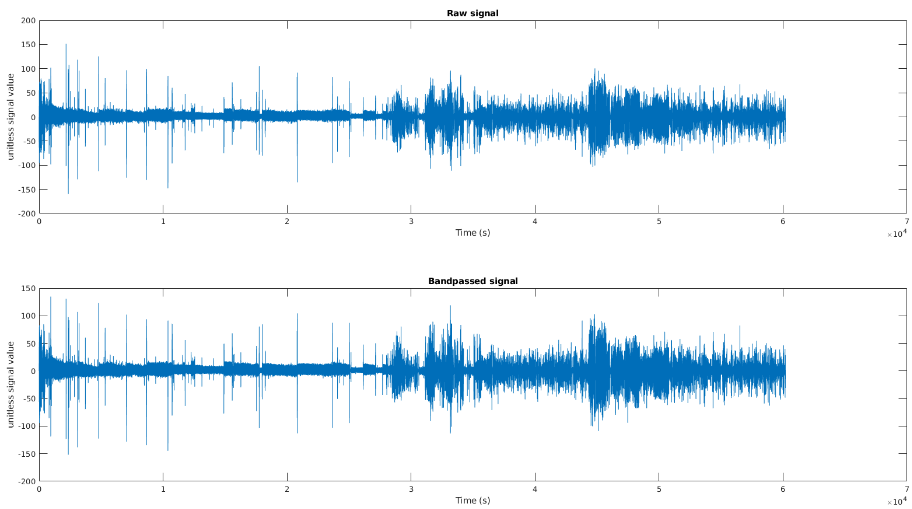
The inflection area ratio is a useful term that's related to vascular distensibility. The hard part is determining at which point in time to calculate the area under the curve. This program calculates the inflection area ratio of a given signal.

Visualize the signal, and pick a "clean" area

The signal is very noisy. A clean region will manually be picked in order to check the efficacy of the base algorithm. A filtering algorithm can later be created to ether not calculate inflection area in noisy signal regions or to filter out poor inflection area values.

```
clear all
close all
clc

raw = csvread('PPG_1.csv');     % raw data.
Fs = raw(2);                    %sampling frequency is the second
 element
raw = raw(3:end);               %data begins on third element
time = (1:length(raw)) / Fs;

% Raw signal plot
figure('units','normalized','outerposition',[0 0 1 1])
subplot(2,1,1)
plot(time,raw)
title('Raw signal')
xlabel('Time (s)')
ylabel('unitless signal value')

%Bandpass Filter plot
subplot(2,1,2)
h  = fdesign.bandpass('N,F3dB1,F3dB2', 12, 0.05, 28, Fs); %bandpass
 0.1 Hz to 28 Hz
bpass = design(h, 'butter');
BPraw = filter(bpass,raw);
plot(time,BPraw);
title('Bandpassed signal')
xlabel('Time (s)')
ylabel('unitless signal value')

%output averages to demonstrate working of highpass filter.
rawav = mean(raw)
rawbp = mean(BPraw)
```

```
rawav =

   -0.0100


rawbp =

   2.5604e-05
```

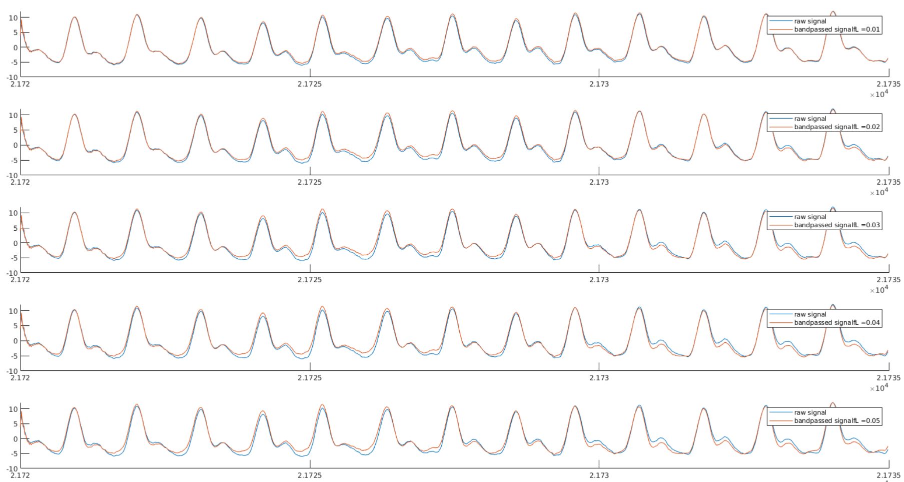


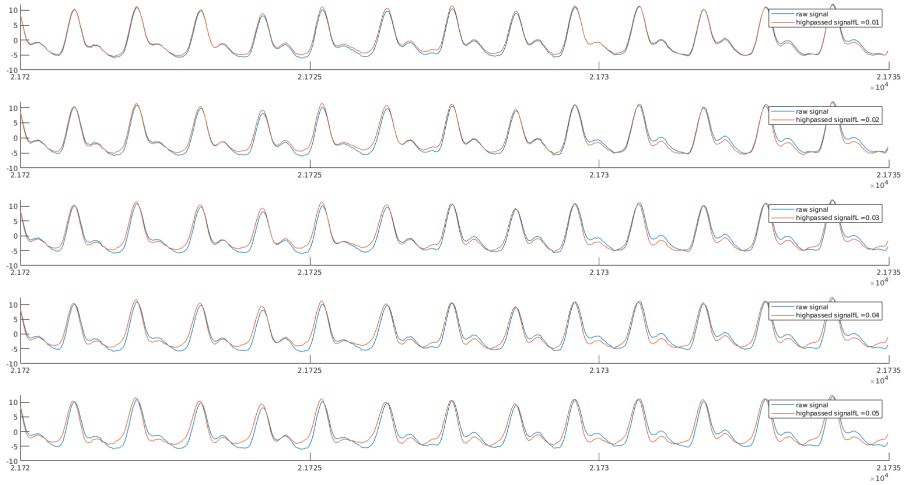
Bandpass Filter

First I want to examine the effect of the bandpass filter. I'll iterate through a couple of filter coeficcients to see the effect of each on the signal. I'll pick a clean region of the signal to do this in. To examine the effecct of other regions, just alter the "cindex" value.

```
%cindex = (21721*Fs:21728*Fs-1);   %index for clean signal
cindex = (21720*Fs:21735*Fs-1);
ctime = cindex / Fs;              %clean time array
craw = raw(cindex); %clean raw signal

%Plot bandpass filter for different cutoff values
figure('units','normalized','outerposition',[0 0 1 1])
for i = 1:5
    low_coeff = 0.01 * i; %iterate through low pass cutoff values
    h  = fdesign.bandpass('N,F3dB1,F3dB2', 12, low_coeff, 28, Fs);
  %create filter "h"
    bpass = design(h, 'butter');
  %design butterworth filter based on h
    cBP_raw = filter(bpass,raw(cindex));
  %filter raw data

    subplot(5,1,i)  %plot bandpass filtered signal at different
 lowpass coefficients
```

```
    hold on
    plot(ctime,craw)
    plot(ctime,cBP_raw)
    legend('raw signal', strcat('bandpassed signal    ','fL = ',
 num2str(low_coeff)))
    hold off
end

%Plot highpass filter at different values. Same logic as before.
figure('units','normalized','outerposition',[0 0 1 1])
for i = 1:5
    low_coeff = 0.01 * i; %low pass cutoff.
    h  = fdesign.highpass('N,F3dB', 12, low_coeff, Fs);
    hpass = design(h, 'butter');
    cHP_raw = filter(hpass,raw(cindex));

    subplot(5,1,i)
    hold on
    plot(ctime,craw)
    plot(ctime,cHP_raw)
    legend('raw signal', strcat('highpassed signal    ','fL = ',
 num2str(low_coeff)))
    hold off
end
```

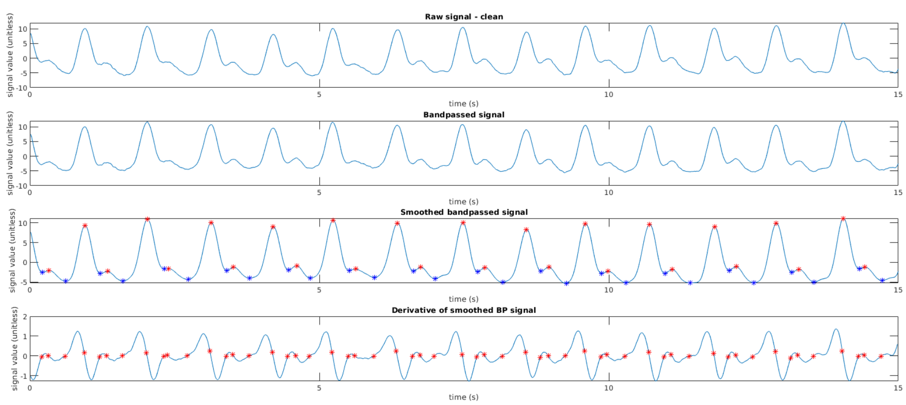




Take a moving average to smooth jitters, plot derivative, find minima and maxima

```
figure('units','normalized','outerposition',[0 0 1 1])
smooth = movmean(BPraw,8);                %8 sample moving average
sig = BPraw(cindex);
smoothsig = smooth(cindex);               %smooth signal in specified
 clean range
smoothtime = (1:length(smoothsig))/Fs;   %time vector for smoothed
 signal

%Plot OG signal
subplot(5,1,1)
plot(smoothtime,raw(cindex))
title('Raw signal - clean')
xlabel('time (s)')
ylabel('signal value (unitless)')

%plot BP signal
subplot(5,1,2)
plot(smoothtime,sig)
title('Bandpassed signal')
xlabel('time (s)')
ylabel('signal value (unitless)')

%plot smoothed signal. Find minima and maxima
subplot(5,1,3)
plot(smoothtime,smoothsig)
title('Smoothed bandpassed signal')
xlabel('time (s)')
ylabel('signal value (unitless)')
hold on    %plot peaks
[maxs maxlocs] = findpeaks(smoothsig);    %find local maxima
```

```
plot(smoothtime(maxlocs),smoothsig(maxlocs),'*r')
[mins minlocs] = findpeaks(smoothsig*-1);
mins = mins*-1;
plot(smoothtime(minlocs),smoothsig(minlocs),'b*')
hold off

%plot derivative and zero crossings (should correlate with peaks found
%previously)
subplot(5,1,4)
d1 = diff(smooth(cindex));
N = 1:length(d1);  %sample index for diff singal
t1 = N/Fs;        %time for diff signal
plot(N/Fs,d1)
title('Derivative of smoothed BP signal')
xlabel('time (s)')
ylabel('signal value (unitless)')
hold on %plot when it crosses zero
zs = Zero(cindex(2:end),d1);
plot(t1(zs),d1(zs),'*r')
hold off
```



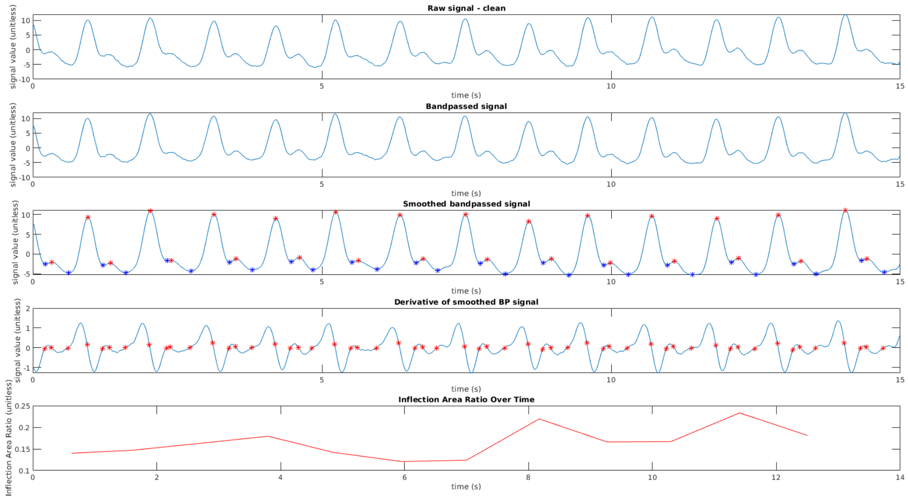
Determine Inflection Area Ratio

I've found the locations where mins and maxes occur. I have an array of mins and an array of maxes. I can iterate through this array to determine the indeces that will determine important signal parameters, such as inflection area ration. To determine the inflection area ratio, I need the area under the curve of the diastolic portion and the systolic portion. I have the variables "mins", "maxs", "minlocs", and "maxlocs" which represent the minima, maxima, minima index locations, and maxima index locations. I can use these four arrays to create logic that should determine systolic and diastolic region on a clean (important!!! dealing with a bad signal will come later) signal. I'll start the code at a complete minimum value.

```
%find total minima.
mincount = 0;  %initialize count variable for mins
for i = 2:length(mins)-1
    if mins(i) < mins(i-1)
```

```
        mincount = mincount+1;
        tmins(mincount) = mins(i) ;         %total mins array
        tminlocs(mincount) = minlocs(i);    %total min location array
    end
end

maxcount = 0;   %initialize count variable for maxs
%find total maxima and their index locations.
for i = 2:length(maxs)-1
    if maxs(i) > maxs(i-1)
        maxcount = maxcount + 1;
        tmaxs(maxcount) = maxs(i);
        tmaxlocs(maxcount) = maxlocs(i);
    end
end

%make a for loop that iterates from one minima to the next
for i = 1:length(tminlocs)-1
    sdur = tminlocs(i):tminlocs(i+1);             %array with segment
 duration indeces i.e. [343 344 345 ... 410] contains 1 ppg waveform
    p = sort([tminlocs(i) tminlocs(i+1)]);       %sort array to
 determine baseline
    siggy = smoothsig(sdur) + p(1);              %add minimum signal
 value
    [mx mxloc] = findpeaks(smoothsig(sdur));      %find maxima within
 this signal
    [mn mnloc] = findpeaks(smoothsig(sdur)*-1);   %find minima within
 this signal
    mnloc = mnloc+sdur(1);                       %minloc is relative
 to a start index of 1, fix this by adding start index of signal

    if length(mx) == 2 && length(mn)==1   %EXCLUSION CRITERIAthere
 are two maxima within signal as expected. Add other conditional logic
 HERE!!!!. These are "EXCLUSION CRITERIA"
    systlocs = tminlocs(i):mnloc;         %systolic segment index
    diastlocs = mnloc:tminlocs(i+1);      %diastolic segment index
    offset = -smoothsig(sdur(1));         %offset to make all areas
 under curve positive
    systsig = smoothsig(systlocs)+offset;        %systolic segment
 signal
    diastsig = smoothsig(diastlocs)+offset;      %diastolic segment
 signal
    A1(i) = trapz(systsig);               %systolic area
    A2(i) = trapz(diastsig);              %diastolic area
    ifarea(sdur(1)) = A2(i)/A1(i);        %inflection area ratio
    else
        ifarea(sdur(1)) = 0;
    end
end

%plot inflection area
subplot(5,1,5)
ifarea(ifarea>2)=0; %remove outliers
ifarea(ifarea<0)=0;
```

```
it = (1:length(ifarea))./Fs;    %time array
f = find(ifarea);               %location indeces of non-zero values
plot(it(f),ifarea(f),'r')
title('Inflection Area Ratio Over Time')
xlabel('time (s)')
ylabel('Inflection Area Ratio (unitless)')

function fourier_plot(Fs,signal)    %plot the fourier of the signal.

Y = fft(signal);
L = length(signal);

%math to plot single sided amplitude spectrum
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;

%plot
plot(f,20*log10(P1))
title('Single-Sided Amplitude Spectrum of X(t)')
xlabel('Frequency (Hz)')
ylabel('Amplitude (db)')
%axis([0.5 30 -inf inf])
end

%Finds zeros of a sample vector. x is sample vector, y is signal.
function z = Zero(x,y)
z = [];  %zeros
for i = 1:length(y)-1
    if (y(i) > 0 && y(i+1) < 0) || (y(i) < 0 && y(i+1) > 0) || (y(i)
 == 0)
        z = [z,i];  %add index position
    end
end
end
```



Published with MATLAB® R2018a