Enhancement of signals

Noise removal with moving averaging and synchronous averaging
Vibration caused by unbalanced shaft and contaminated by electrical noise
10 s record of acceleration signal

nPts = 80000 sampled at 8000.0 Hz and occupying 625 kbytes
Moving average for noise removal
Moving average

• Moving average is used to smooth the data which produces an effect of low pass filtering.
• Points in the signal in the window of certain size are substituted by their average value.
• The window can advance by one point (max overlap) or more, up to the window size (no overlap).
• Moving average is typically applied in conjunction with the calculation of the RMS (or power) for sections of the signal, e.g. reverberation time.
• Moving average can be easily implemented as the filter() function – see Matlab Help for details.
Dr Michael Sek - Synchronous and Moving Average

Moving average

Moving average, window size 100, step every 1 point(s)

Enhanced time average of 80 ensembles: threshold at 5.8G, window size=4096 pts
function [mAver] = movingAverage(g, winSize, advPts)
%M. Sek @ 2017 – moving average
% Usage: [mAver] = movingAverage(g, winSize, advPts)
% mAver – returned vector with the moving average
% g – signal
% winSize – window size
% advPts – advance by this # of points (step size)

nPts = length(g);
% starting indices of each block of data
idx1 = [1:advPts:nPts-winSize+1]';
% last indices of every block of data
idx2 = idx1 + winSize - 1;

nSections = length(idx1);
mAver = zeros(nSections, 1);
for i = 1:nSections
    mAver(i) = mean(g(idx1(i):idx2(i))); 
end
Moving average introduces a phase shift (1)

- Signal g
Moving average introduces a phase shift (2)

- Moving average with window=2, step=1
  \[mAver = \text{movingAverage}(g, \text{window}, \text{step});\]
Moving average introduces a phase shift (3)

- Moving average shifted by the time interval corresponding to ‘(window-1)/2’
Enhanced signal obtained by synchronous averaging

Enhanced time average of 847 ensembles: threshold at 2.4G, window size=4096 pts
A section of original signal compared with the synchronous average

Enhanced time average of 847 ensembles: threshold at 2.4G, window size=4096 pts
Pings buried in noise

- A periodic ping (middle) is buried in the random white noise (top) producing a signal (bottom)
Ping recovery with synchronous averaging

Enhanced time average of 650 ensembles: threshold at 1.8G, window size=4096 pts
Principle of Synchronous Averaging

- Select a trigger threshold
- Find indices of points located just before the threshold level as starting indices (in red).
- Select the size of the window for averaging.
- Extract sections of window length that start at the threshold indices.
- Average the corresponding points of those sections.
Synchronous averaging - trigger starting indices

Assuming that the signal is in vector g:

```matlab
startIdx = find(diff(sign(g-threshold)) == -2);
```
Synchronous averaging procedure

- Find starting points
- Truncate the tail
- Form a matrix for ‘mean’
- Apply ‘mean’ to every row

Not enough points
### Finding starting indices

<table>
<thead>
<tr>
<th>g</th>
<th>(g-thres)</th>
<th>sign</th>
<th>diff</th>
<th>find(diff(sign(g-threshold))== -2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-5.5</td>
<td>-1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>1</td>
<td>-2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>-1.5</td>
<td>-1</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>1.5</td>
<td>1</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>-0.5</td>
<td>-1</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>-3.5</td>
<td>-1</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>-2.5</td>
<td>-1</td>
<td>-2</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>1</td>
<td>-2</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>1.5</td>
<td>1</td>
<td>-2</td>
<td>9</td>
</tr>
</tbody>
</table>

- **Threshold 6.5**
- **Threshold 7**
- **Threshold 8**
The largest index of ‘g’ can be $7 = 10 - 4 + 1$

$$nPts = \text{length}(g);$$

$$\text{lastIdx} = nPts - \text{winSize} + 1$$

$$\text{idxStart} = \ldots$$

$$\text{find}([\text{diff}(\text{sign}(g(1:\text{lastIdx}) - \text{thres})) == -2]);$$

```
\text{truncate}
```

```
\% What if \text{thres} < 0
\% ... == -2*\text{sign}(\text{thres})
```
Dr Michael Sek - Synchronous and Moving Average

Averaging

\[
\begin{array}{c|c|c|c|c}
\hline
\text{Threshold} & 6.5 & 6.5 & 6.5 & 6.5 \\
\hline
\text{idxStart} & 1 & 2 & 7 & 8 \text{ (aver)} \\
\text{g} & 1 & 2 & 5 & 7 \\
\hline
\end{array}
\]

\[n\text{Sections} = \text{length(idxStart)}; \ \% = 3\]

\%initialize a matrix with zeros

\[\text{aver} = \text{zeros}(\text{winSize},1);\]

\[\text{for } i=1:\text{winSize}\]

\[\text{aver}(i) = \text{mean}(\text{g(idxStart + i - 1)});\]

\[\text{end}\]
**Implementation of synchronous averaging as a Matlab function**

```matlab
function [aver,nSections,idxStart] = synchronousAverage(g,thres,winSize)
%M.Sek @2017- Enhanced time processing of a noisy signal in the time domain
%Usage: [aver,nSections,idxStart] = synchronousAverage(g,thres,winSize)
%aver - synchronous average of time signal
%nSections - number of averaged sections
%idxStart - indices of starting points of each section
%g - signal
%thres - threshold or trigger level
%winSize - window size

%Size of g
nPts = length(g);
%The last available starting point is (nPts-winSize+1)
lastIdx = nPts-winSize+1;
%Find indices of points just above the threshold before the signal
%crosses the threshold (negative slope) within the range (1:lastIdx)
idxStart = find(diff(sign(g(1:lastIdx)-thres))== -2*sign(thres));
%Having enough points in the last section has been already taken care of
nSections = length(idxStart); %number of sections
%Initialize with zeros a vector to hold the average
aver = zeros(winSize,1);
for i=1:winSize
    aver(i) = mean(g(idxStart + i - 1));
end
if nargout == 0
    plot(......)
end
```