3 Windowing

Speech is non-stationary signal where properties change quite rapidly over time. This is fully natural and nice thing but makes the use of DFT or autocorrelation as a such impossible. For most phonemes the properties of the speech remain invariant for a short period of time ($\approx$ 5-100 ms). Thus for a short window of time, traditional signal processing methods can be applied relatively successfully.

Most of speech processing in fact is done in this way: by taking short windows (overlapping possibly) and processing them.

The short window of signal like this is called *frame*. In implementational view the windowing corresponds to what is understoods in filter design as window-method: a long signal (of speech for instance or ideal impulse response) is multiplied with a window function of finite length, giving finite length weighted (usually) version of the original signal. Illustration is in figure 1.

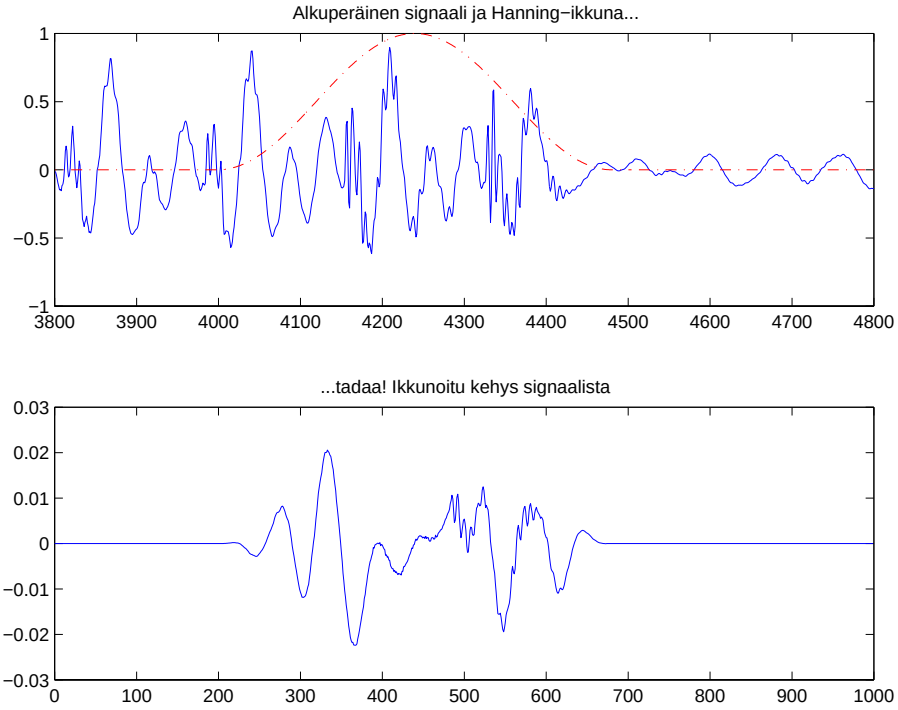

Figure 1: The original signal and its windowed version below.

In speech processing the shape of the window function is not that crucial but usually some soft window like Hanning, Hamming, triangle, half parallerogram, not with right angles.

The reason is same as in filter design, sideband lobes as substantially smaller than in a rectangular window. (figure 2). Moreover in LPC-analysis (to be studied later on) the signal is presumed to be 0 outside the window, hence the rectangular window produces abrupt change in the signal, which usually distorts the analysis.

We must keep in mind that in speech processing (unlike in filter design for instance) the methods are seldom well grounded mathematically. Usually the goal is to implement a system working as well as possible in some application. The criteria of the application may be buried under complex human-related criteria/preferences hindering precise mathematical modeling. For instance the quality of coded speech, intelligibility of synthesized speech or pleasantness of enchanced speech.

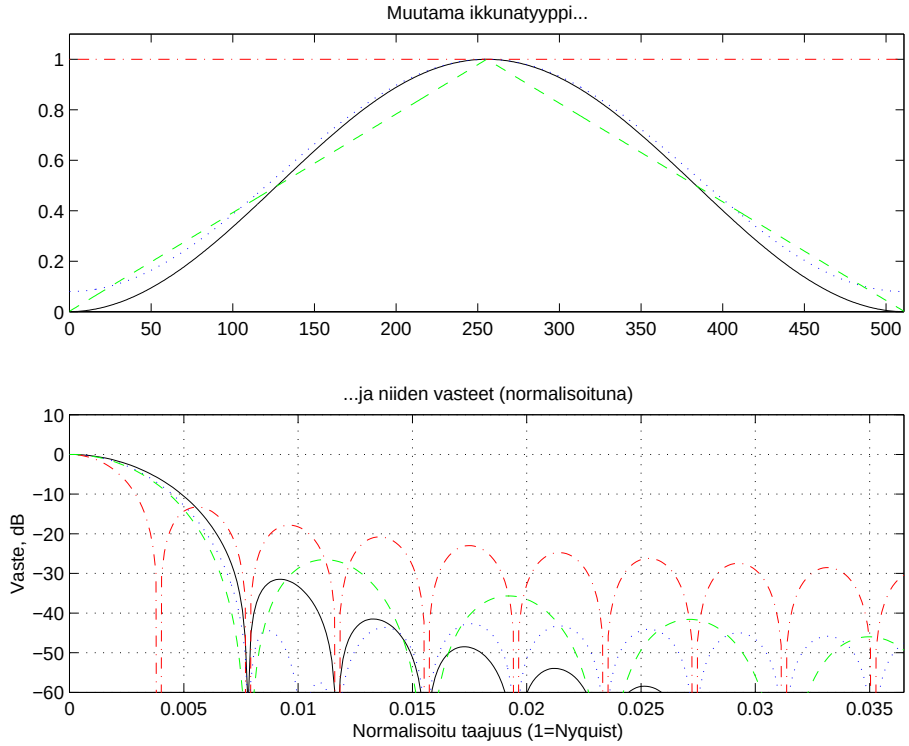

Figure 2: A bunch of windows and their amplitude responses.

Based on this, the windowing should be confronted with certain degree of freedom being prepared to change the window function when necessary.

Example: in speech coding the samples will be pursued to be represented exactly as they are when the rectangular window is used. On the other hand when the LPC-parameters are analyzed, a soft non-symmetric window is used in order to minimize the delay. In speech recognition the windows are usually overlapping 10 ms windows, which are analyzed in order to make hypothesis of the current phoneme. The hypotheses are combined over several frames and finally the decision is made to maximize the joint probability.

If one desires to shape the speech (not only analyze), it is a good idea to use overlapping windows summing approximately to 1. For instance: the coarsest coding system in the world, where in each frame the 10 highest samples of DFT are quantized and others are set to 0. The inverse transform then restores the original signal.

Let's implement this, because it brings up nicely different aspects related to windowing, analyzation and synthesis.

The Matlab-source script can be found in

`http://www.cs.tut.fi/sgn/arg/8003051/FFT_coding_en.m`

and below.

```
function syn = FFT_coding_en(ind,N);
% syn = FFT_coding_en(ind,N);
%
```

```
% This is the English version of FFT_koodaus_fi.m. The operation of
% FFT_coding_en.m is identical to that of FFT_koodaus_fi.m.
%
% Windowing demo: the speech is processed in
% windows of length 60 ms and round edges (if ind == 0), or in
% windows of length 30 ms and a rectangular shape (if ind == 1).
% FFT is calculated and all but N+1 biggest-amplitude frequency bins are
% set to zero. Then, the output speech ('syn') is synthesized using IFFT.
% The speech data used by this function is yhdeksän16.mat.


load yhdeksän16.mat
x = x(:); % force as column vector
fs = 16000; % sampling frequency

if ( ind == 0),
   awinlen = round( fs*0.06); % length of the analysis window, 60 ms
   % the window function is almost abritrarily chosen:
   % we make a window that is flat in the middle and has soft edges
   temp = hanning( fs*0.01); % the soft edges are here
   awinfun = [temp(1:length(temp)/2); ones(awinlen-length(temp),1); ...
           temp(length(temp)/2+1:end)];
   swinlen = round(awinlen/2);  % length of the synthesis window is
                                % half of the analysis window
   swinfun = hanning( swinlen); % window function for synthesis
   nforward = swinlen/2; % how many frequency bins there are between
                         % successive frames
end
if ( ind == 1),
   awinlen = round( fs*0.03); % length of the analysis window, 30 ms
   awinfun = boxcar( awinlen); % now a rectangular window shape is chosen
   swinlen = round(awinlen); % now the length of the synthesis window =
                             % length of the analysis window
   swinfun = boxcar( swinlen); % window function for synthesis
   nforward = swinlen; % how many frequency bins there are between
                       % successive frames
end

figure
plot(awinfun,'--');
hold on
plot(1+awinlen/2-swinlen/2+(0:swinlen-1), swinfun, 'r-.');
title('analysis window (blue) and synthesis window (red)')
axis([0 1000 0 1.4])
print -depsc a_and_s_window.eps
```

```
fftind = 1:floor(awinlen/2)+1; % indices of half of FFT vector
n = 1+ceil(awinlen/2); % the middle sample of the frame
syn = zeros( size( x));
flag = 0; % flag for doing some decisions related to drawing
while ( n+ceil(awinlen/2) <= length(x))
   awinind = n-ceil(awinlen/2)+(0:awinlen-1); % indices of the analysis
                                              % window in current frame
   frame = x( awinind).*awinfun; % frame
   Frame = fft(frame); % FFT of the frame

   % we search (N+1)th largest absolute value
   [val,sind] = sort( abs( Frame( fftind)));
   valN = val( end-N);
   Frame( abs(Frame) < valN)=0; % set to zero all but N+1 highest-amplitude
                                % frequency bins
   iframe =  real(ifft(Frame)); % inverse FFT, the real part is taken
                                % to avoid precision problems

   swinind = n - swinlen/2 + (0:swinlen-1);
   % synthesis windowing
   swin = iframe( 1+ awinlen/2 - swinlen/2 + (0:swinlen-1)).*swinfun;
   syn( swinind) = syn(swinind) + swin; % overlap-add

   if ((n > 3000) & ~flag) % draw a pitcure of the frame
      figure
      plot( x(awinind),'k'); % this one can hardly be seen in the picture
                             % if the rectangular window is used
      hold on
      plot( frame,'b-.');
      plot( awinlen/2-swinlen/2+(0:swinlen-1), swin, 'r-.');
      title('original speech (black), analysis window (blue) and synthesis wi
      print -depsc frame.eps

      flag = 1;
   end
   n = n + nforward;
end

figure
plot(x,'k');
hold on
plot(syn,'r-.');
title('original speech (black) and modified speech (red)');
print -depsc result.eps
```

The idea is the following: the speech is first windowed for FFT analysis with relatively long (60 ms) and quite rectangular window. The purpose is to get some idea of the frequency content in the window, but so that the windowing will not excessively affect the shape of the frame. As a (cautionary) example, the function has also been written to contain a possibility of using a rectangular window of length 30 ms, but in practice it is not advisable to use this option since it considerably degrades the speech quality.

Next the FFT of the analysis frame is calculated and all small-energy frequencies are set to zero. The inverse FFT is then applied to this partially zero spectrum to get the time domain signal. The next thing to do is a new windowing system (in case the 60 ms window was chosen): we window only the central part of the frame and edges will be faded. We use Hanning-window that has the nice property of summing to 1 when the time difference between successive windows is half of the length of the window (exactly 1 when the length is odd integer). The method of using overlapping short-time signals and forming the reconstruction by summing partially overlapping frames is called *overlap-add*-method.

If, as an experiment, the 30 ms rectangular window is chosen, the successive frames do not overlap at all, but instead, the synthesis consists of catenating the time domain frames produced by the inverse FFT as such. In this case, discontinuities will occur at the frame boundaries, which makes the output speech sound very unnatural.

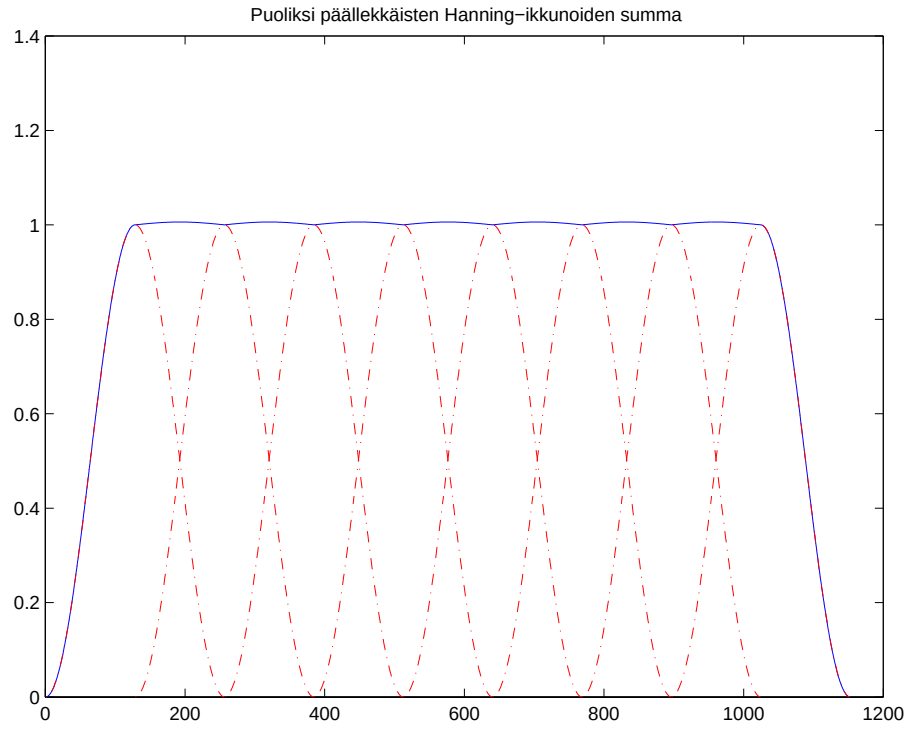

Figure 3: The sum of Hanning-windows with time difference half of the length is almost 1, which makes them attractive in synthesis windowing.

When the 60 ms window is used (i.e., ind = 0), and the value of 9 is chosen for N (i.e., 10 largest-amplitude frequency bins of FFT are preserved), the function will produce the figures shown in Figures 4-6.

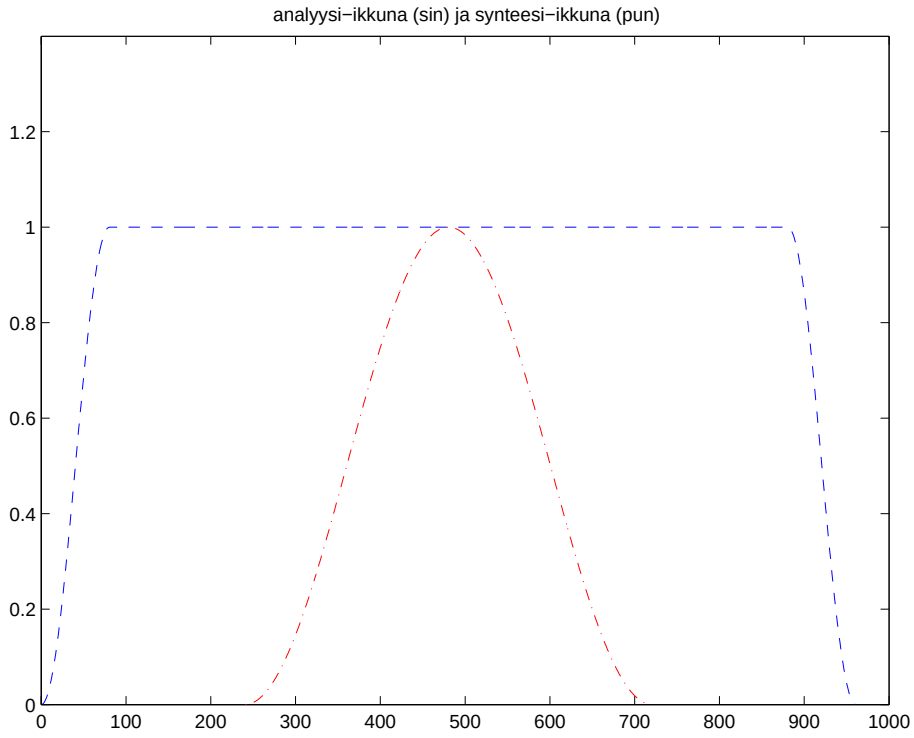

Figure 4: Analysis and synthesis windows used in the function.

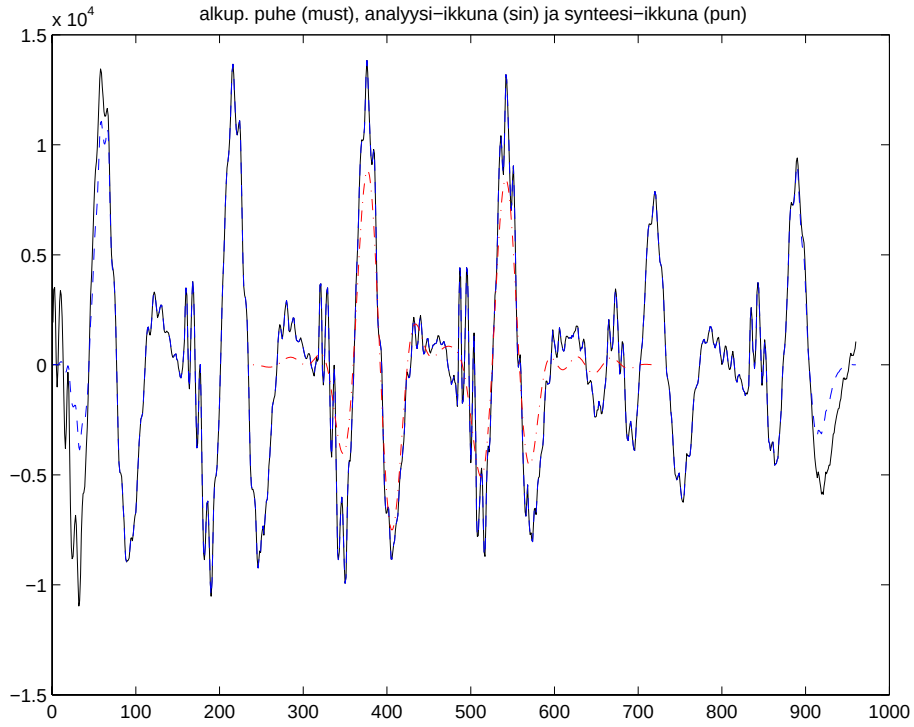

Figure 5: Original speech (black), analysis window(blue), synthesis window(red).

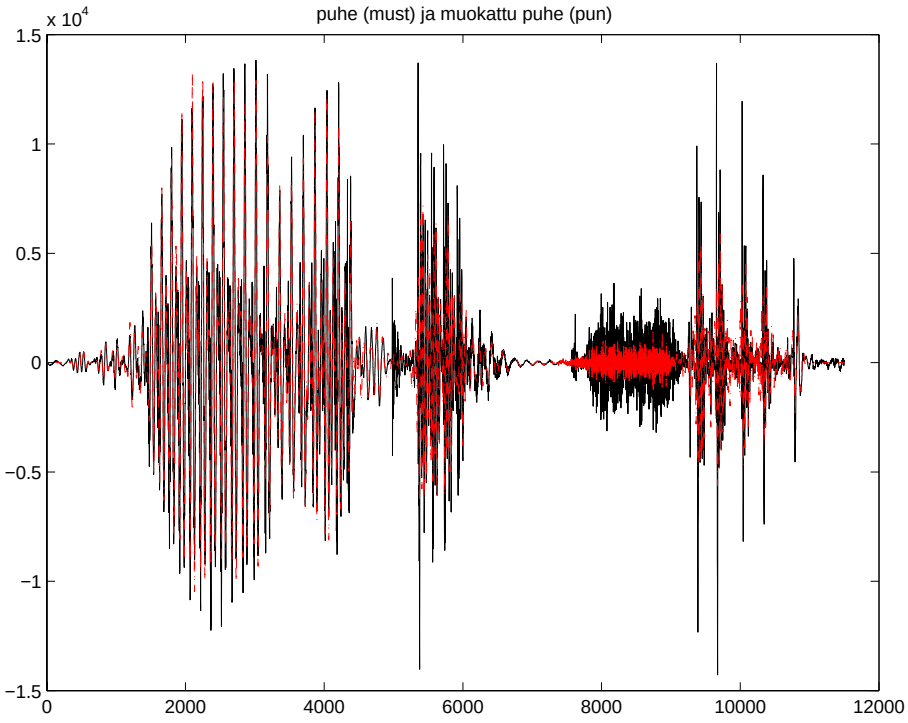

Figure 6: Original speech(black) and coded speech (red).