## T-61.3010 Digital Signal Processing and Filtering

(v. 1.0, 30.1.2009), Matlab #2 (3.2., 4.2., 5.2.2009)

**Registration** in WebOodi. Bring your own **headphones** if you have. The assistant will guide you through the exercises, but you may go on your own speed. Feel free to ask the assistant, if you have troubles. You can also consult `http://www.cis.hut.fi/Opinnot/T-61.3010/how_to_start_with_matlab.shtml` or `kuinka_aloitan_matlabin.shtml`.

Getting started: In **Windows** just click `Programs - Matlab`. Write down the code into separate files in your working directory (e.g. `Z:\DSP\`) for future use. Set the "Current Directory" in Matlab to point to the working directory (or type `cd <workdir>`).

The problems marked with [**Pxx**] are from the course exercise material (Spring 2009).

**In the end** of this session **you should know**: (a) to be aware of aliasing effect, (b) how to generate sinusoidal signals, (c) how to load and analyze audio signals in time, frequency and time-frequency domain, (d) an example of a simple moving average (MA) filter.

1. [**M2057**] Run the code below in order to plot a cosine signal with $A = 2.5$, $f = 400$ Hz, $\theta = \pi/6$.

```
%% Synthesize a cosine signal
fT = 16000;                       % sampling frequency 16 kHz
f  = 400;                         % frequency 400 Hz
t  = [0 : 1/fT : 0.1];            % time axis, 0.1 seconds
y  = 2.5 * cos(2*pi*f*t + pi/6);  % cosine
%% Plot the signal
figure;                           % open a NEW window
plot(t, y);                       % plot to the current active window
grid on;
title('My cosine');
xlabel('time (s)');
ylabel('amplitude');
axis([0.015 0.033 -2.7 2.8]);       % manual zoom: [xmin xmax ymin ymax]
%% Listen to the signal
soundsc(y, fT);                   % or 'wavwrite(y, ...)' into a file
%% Save the figure into a file
print('-dpng','myCosine.png');    % also saveas(), etc...
```

   **Task:** Plot the cosine of the previous example and a new cosine ($A = 2.5$, $f = 16400$ Hz, $\theta = \pi/6$) with black circles in the same axis (`help plot`, `hold on`). The interesting result is that the circles with $f = 16400$ Hz seem to lie on the curve of a continuous-time cosine with $f = 400$ Hz. How do you explain the result, see Figure 1? Compute the normalized angular frequency $\omega_i = 2\pi(f_i/f_T)$ for both cases.

2. [**M2021**] Download the file `kiisseli.wav`. Analyze the audio sample `kiisseli.wav` in time and frequency domain with the following code.

```
%% Read and listen to the signal
[x, fs, nbits] = wavread('kiisseli.wav');   % download the file!
M   = length(x);
```



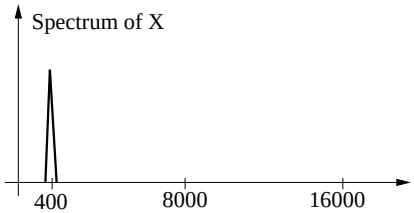



Figure 1: Problem 1: Spectrum $|X(e^{j\omega})|$.

```
soundsc(x, fs);
%% Plot the signal waveform in time domain
t   = [ 0 : M-1 ] / fs;                     % time-axis
figure(1); clf;           % open/activate Figure No 1, clean it
plot(t, x);
grid on;
xlabel('time (sec)');
title('/kiisseli/');
%% frequency domain
xF  = fft(x);             % Discrete Fourier transform of signal x, 0..2pi
mag = 20*log10(abs(xF));  % in decibels; in linear scale: mag = abs(xF)
w   = fs * [0 : (M-1)]/M; % frequency axis
figure(2); clf;
plot(w, mag);             % spectrum
grid on;
xlabel('frequency (Hz)');
ylabel('dB');
title('DFT of /kiisseli/');
%% zoom only frequencies from 0 to half of the sampling frequency!
axis([0 fs/2 min(mag) max(mag)]);
%% Time-frequency domain: short-time Fourier-transform STFT / spectrogram
figure(3); clf;
spectrogram(x, 128, 64, 128, fs, 'yaxis');   % spectrogram
title('Spectrogram of /kiisseli/');
colorbar;                    % adds a colorbar: color <=> value
%% If you need to print a grayscale document, then change palette
colormap(gray);              % probably better if grayscale printing
colorbar;                    % adds a colorbar: gray value <=> value
```

   **Task:** What is the sampling frequency of the audio file? Find a "quasi-periodic" part of the signal, e.g. /i/, and read the value of fundamental period in Figure 1. The spectrum in Figure 2 seems to be symmetric before zooming, why so? Why is this spectrum not so relevant in the speech analysis? What can you see in the spectrogram, Figure 3? You can also run `specgramdemo(x, fs)` from command line.

3. [**M2056**] A normal procedure to get "a big picture" is to take averages inside a certain time window. The simpliest average is to add two adjacent values (window length 2) and divide the result by two (Moving Average 2, MA-2):

$$y[n] = 0.5 \cdot (x[n] + x[n-1])$$

It can be shown that the corresponding impulse response and frequency response are

$$\begin{aligned} h[n] &= 0.5 \cdot (\delta[n] + \delta[n-1]) \\ H(e^{j\omega}) &= \sum_k h[k] e^{-jk\omega} = 0.5 \cdot (1 + e^{-j\omega}) \end{aligned}$$

**Task:** Draw a flow/block diagram of the filter. Fill in the missing line in the Matlab function `ma2.m` which implements the MA-2 filter, see the code below. Apply your filter `ma2` to the signal `x`, e.g. `kiisseli.wav`.

How does the output look like compared to the input? Listen to the signal before and after filtering. What can be said about frequency-domain properties of MA-2, in other words, draw $|H(e^{j\omega})|$ in range $[0 \ldots \pi]$. Is the filter lowpass / highpass / bandpass / bandstop / allpass?

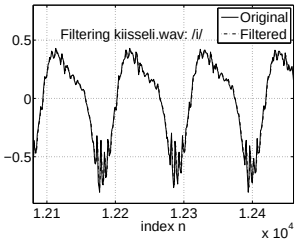
A function file `ma2.m`:

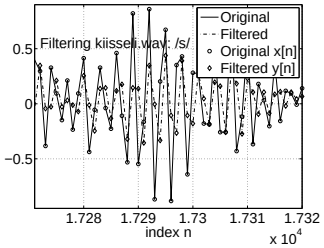
```
function y = ma2(x)
% MA2 computes two-point averaging filter, ``moving average''
% Usage:  [y] = ma2(x);

y    = zeros(size(x));          % initialize to zeros
y(1) = x(1);                    % value y(1)
for k = [2 : length(x)]         % one possible implementation
  ...                           % FILL THIS LINE
end;                            %   note: index values start from 1
```

A script file `ma2analysis.m` for the analysis:

```
%% Read or create a vector x
[x, fs] = wavread('kiisseli.wav');
%% Analysis of MA2 filter
n  = [ 1 : length(x) ];
y  = ma2(x);                    % calls function ma2 with x
%% Signals in time-domain
figure(41); clf;
plot(n, x, 'b', n, y, 'k-.');
xlabel('time indices n');
grid on;
legend({'Original','Filtered'});
%% Filter analysis, see [P4] and/or Matlab #1: H(w) = 2 - exp(-j*w)
w  = [0 : pi/256 : pi];
H  = 0.5 * (1 + exp(-j*w));     % frequency response
r  = abs(H);                    % amplitude response
figure(42); clf;
plot(w, r);
xlabel('norm. angular frequency \omega'); ylabel('|H(e^{j \omega})|');
grid on;
%% Listen to original:
soundsc(x, fs)
%% Listen to filtered:
soundsc(y, fs)
%% Listen to difference:
soundsc(x-y, fs)
```






**Task:** Create a longer MA filter in order to get smoother result.