31606 Signals and Linear Systems in discrete time

Hands on 9
Group 1

Søren Krogh Andersen, Carsten Lau Nielsen
David Scott Winterrose
s123369, s123161, s123821

November 13, 2014

Danmarks Tekniske Universitet

||||1 How finite should it be?

1.1) Like a hot knife through a Butterworth We are asked to design a filter meeting the following requirements:

- Passband from 0.2 to 0.3
- Stopbands from 0 to 0.1 ad 0.4 to 1
- Rippels in passband no greater than 2dB
- Stopband attenuation min. 100dB

While the headline does imply that a butterworth filter could be used, this does not allow us to utilize that rippels in both the stop and passband are accepted.

We use the *Matlab* filter toolbox to design our filer:

```
% Construct an FDESIGN object and call its ELLIP method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2);
Hd = design(h, 'ellip', 'MatchExactly', match);
```

This gives us a 12^{th} order filter, compared to that a buttersworth filter would have to be of order 24 to meet the same specifications.

Pole-Zero plot of the filter can be seen in fig. 1, frequency response (and specification boundaries) can be found in fig. 2 and impulse response in fig. 3

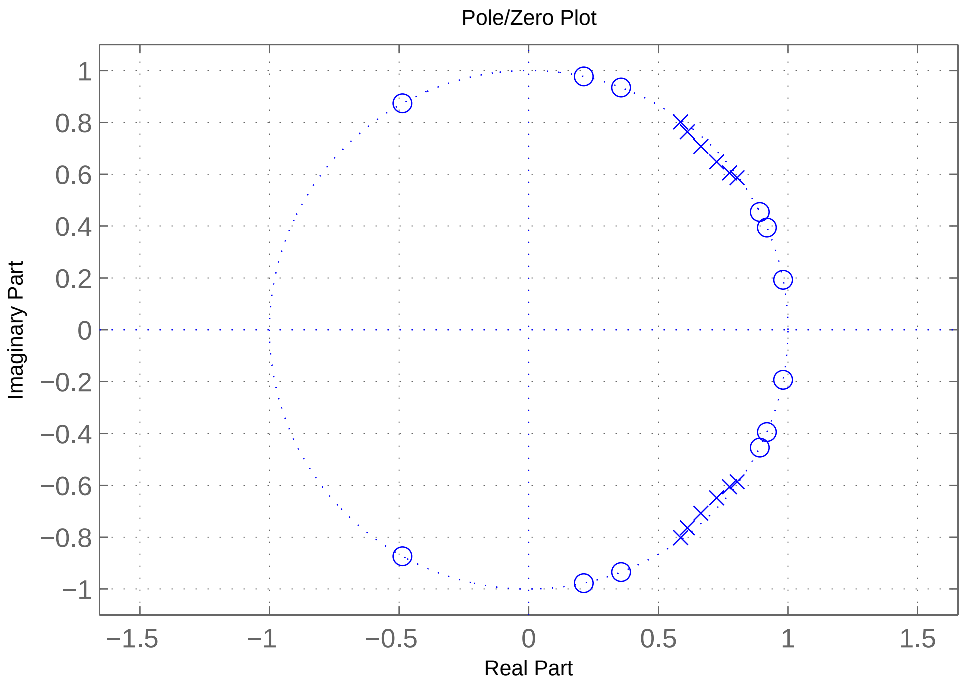

Figure 1: Pole-zero map of a 12^{th} order eliptic bandpass filter with $-3dB$ cutoff frequencies at $0.3004\pi rad/sample$ and $1.997\pi rad/sample$

We find that point i in the impulse response where the maximum amplitude is below a value of 10% of the maximum amplitude of the whole signal so that:

$$x[0:\text{inf}] * 10\% > x[i:\text{inf}]$$

This is done with the *Matlab* code:

```
%make list of max of remaining response:
maxrest = uncut_ir;
for i = 1:length(maxrest)
    maxrest(i) = max( abs( uncut_ir(i:end) ) );
    if maxrest(i) < maxval*sig
        break; %stop here, i is now the index, where all samples [i:inf[ < max
    end
end
```

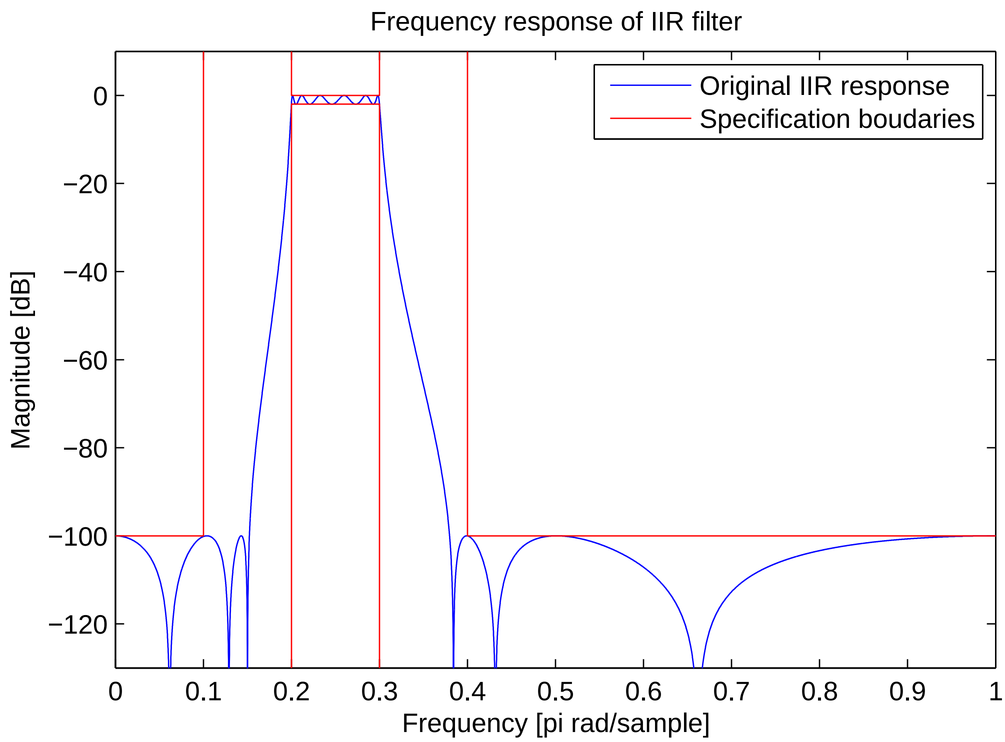

Figure 2: Frequency response of a 12^{th} order eliptic bandpass filter with $-3dB$ cutoff frequencies at $0.3004\pi rad/sample$ and $1.997\pi rad/sample$

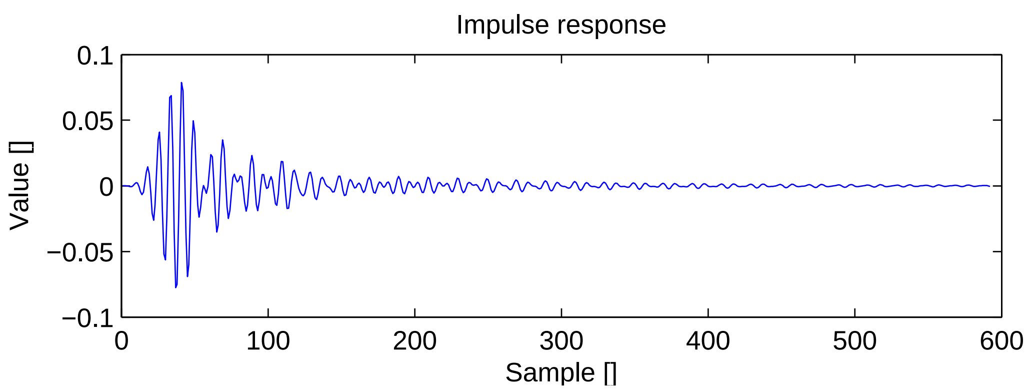

Figure 3: Impulse response of a 12^{th} order eliptic bandpass filter with $-3dB$ cutoff frequencies at $0.3004\pi rad/sample$ and $1.997\pi rad/sample$

This gives an "effective" length of 134 samples. (Had we chosen a buttersworth filter instead this would have been 146 samples).

We now use a rectangular window to cut the impulseresponse at 100%, 75%, 60%, 40% and 10% of the original. The frequency response of the filter with these impulseresponses as kernes are shown in fig. 4

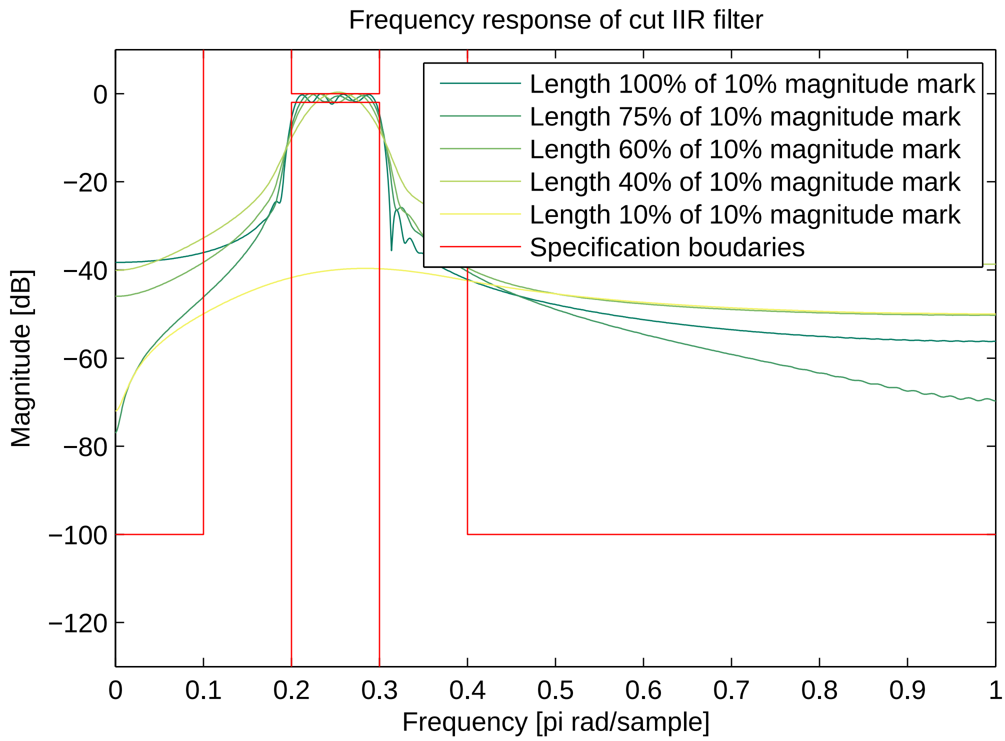

Figure 4: Frequency response of an FIR approximation of an IIR filter using a rectangular window, where the impulse response has been cut at different lengths. Notice how the ripple in the passband stays almost constant at $2dB$, while the attenuation in the stopband is drastically reduced compared to the original IIR filter.

It is quite clear that neither of the filters meet the requirements. As the impulse length is shortend there, surprisingly, is not that big of a difference in the filter, until somewhere between 40% and 10% of the length.

Using a hann window, however, we can manipulate the filter respons to get much closer to the requirements - all without changing the length of the filter. See fig. 5. When applying the window it is important to notice, that the impulse response is already causal and $x[i] = 0$ for $i < 0$, so we only have to apply the half of the window on the right of the magnutude axis.

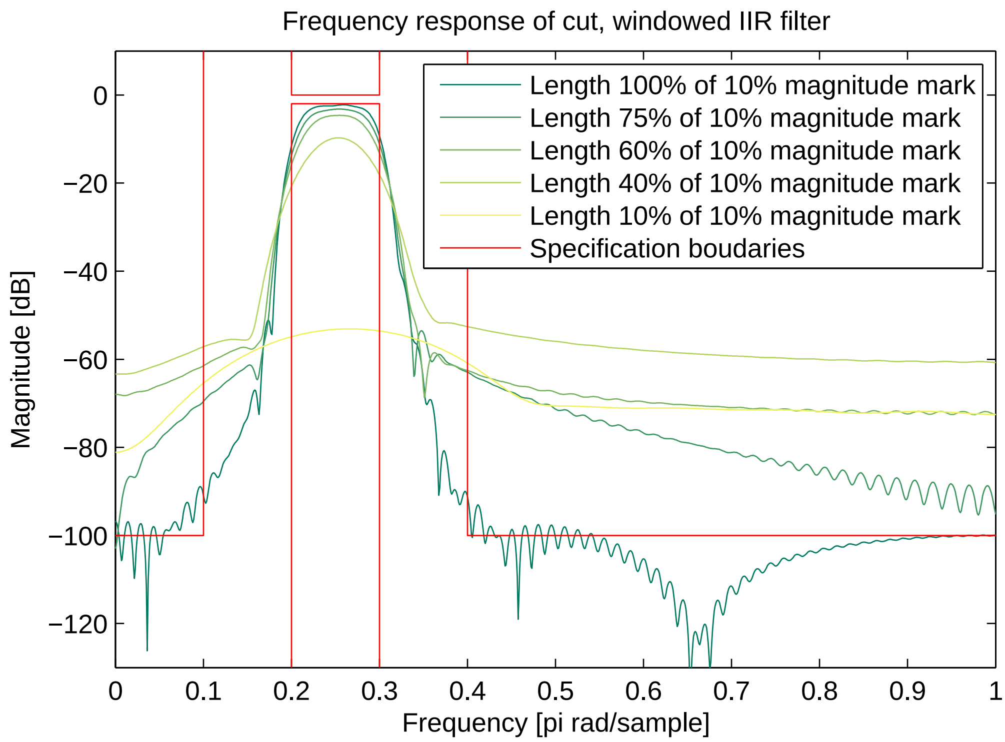

Figure 5: Frequency response of an FIR approximation of an IIR filter using a hann window, where the impulse response has been cut at different lengths. Notice how the passband ripple has been smeared out by convolution in the frequency domain, this has also made the corners of the filter rounder, resulting in the cutoff frequencies to move inside the desired passband. On the other hand stopband attenuation has ben increased by a huge factor.

1.2) Shorter and shorter We are tasked with creating a filter for which the specification has been partly destroyed by a coffee addict. Luckily, the surviving information is sufficient to create the original information.

The filter should pass all frequencies between 0 and 5kHz with a gain on 0dB, this means the filter is a lowpass filter. As we can see from the sketched frequency response, 5kHz corresponds to one sixth of the sampling frequency since a normalised frequency of 1 corresponds to the nyquist frequency. The sampling frequency is 30kHz. Additionally the impulse response of the filter should be 601 samples long, meaning our filter is of the 600th order as it is a FIR filter where the length of the impulse response is $N+1$ where N is the order of the filter.

The filter can be constructed in Matlab using the frequency domian equivalence method. This means we construct a perfect lowpass filter in the frequency domain, and sample it at the frequency bins given by our signal length. Our signal is 601 samples long, which at 30kHz yields a period of 0.02 seconds, meaning that bins in the frequency domain are separated by 50Hz. After sampling our perfect lowpass filter, we transform back to the time domain and delay by $\frac{N-1}{2}$ samples, obtaining the impulse response shown in figure 6 which is, not surprisingly, a time-shifted sinc function.

Since we want to please our boss, we decide to try minimizing the resources required to implement the filter. We do this by reducing the length of the impulse response and applying an appropriate window to reduce high-frequency components. We choose to use the hamming filter, applied to impulse responses of varying lengths shown in figure **??**. In the end we conclude that the filter with an impulse response length of 201 samples is actually better when windowed than the original impulse response in terms of rolloff without sacrificing any significant gain in the passband.

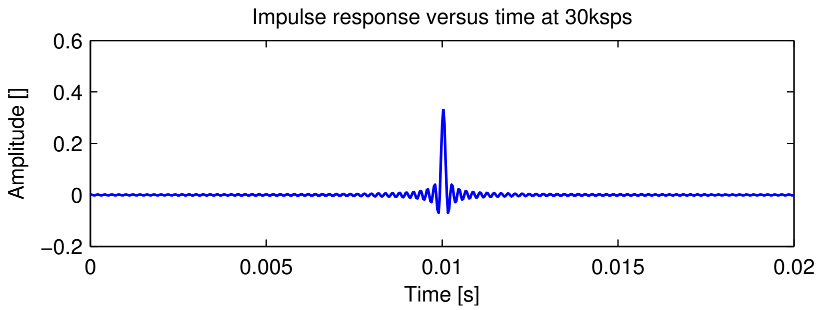

Figure 6: Impulse response of the 600th order FIR filter created by sampling a perfect lowpass filter in the frequency domain and transforming back to the time domain. It is a time-shifted sampled sinc function.

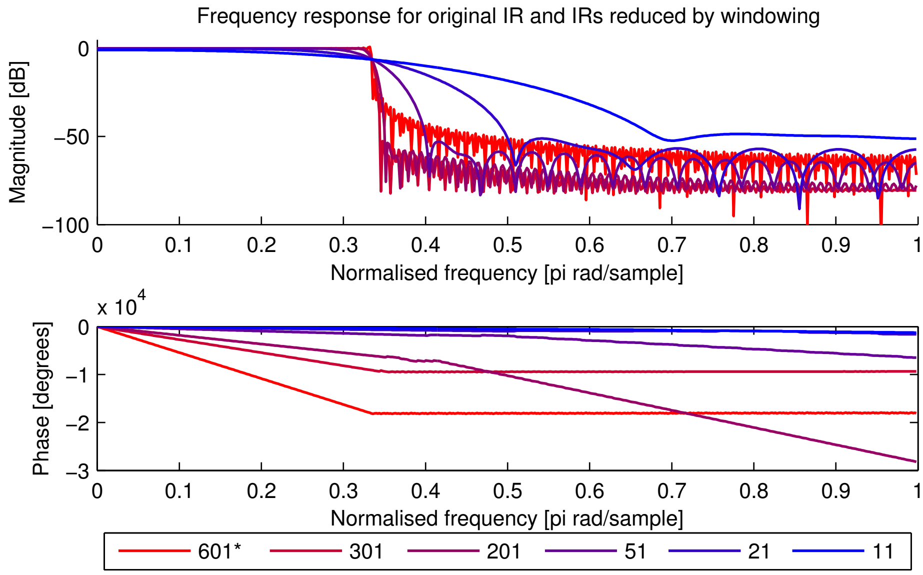

Figure 7: Frequency reponse of the *original impulse response as well as the impulse responses reduced in length by windowing with a hamming window. Legend shows the length of the impulse response in samples. It is clear that the windowed impulse responses have better maximum attenuation all the way to down to 11 samples, however the rolloff is worse for impulse responses shorter than 300 samples. Beyond 51 samples the filter behaviour is completely unsatisfactory for our purposes. 201 would be the best compromise between performance and cost. Note that the phase is linear until the cutoff frequency after which the sawtooth pattern is caused by the inability to represent small changes with floating point numbers at those magnitudes.

||||2 From bass/treble to an equalizer

The task is to implement a tunable FIR equalizer with five equally spaced (linear scale) bands. Using the frequency domain equivalence method the FIR filter coefficients can be determined by defining the magnitude response, transforming to the time-domain, windowing and translating to obtain a causal response. A function that constructs the filter coefficients was implemented where filter order and window type are parameters. The code is shown in Figure 8.

```
function [h] = equalizer5band(G1, G2, G3, G4, G5, n, W);
%INPUT: Band gains: G1, G2, G3, G4, G5. Approximate length: n. Window: W.
%OUTPUT: Impulse response/FIR filter coefficients: h.
m = round(n/10);
H = [G1*ones(1, m) , G2*ones(1, m) , G3*ones(1, m) , G4*ones(1, m) , ...
     G5*ones(1, m) , G5*ones(1, m) , G4*ones(1, m) , G3*ones(1, m) , ...
     G2*ones(1, m) , G1*ones(1, m)];
h = ifftshift(ifft(H, 'symmetric'));
L = length(h);
if W == 0 %rectangualr window (technically unnecessary code)
   h = h; %do nothing
elseif W == 1
   h = h.*hanning(L)';
elseif W == 2
   h =h.*hamming(L)';
end
end %eof
```

Figure 8: The `equalizer5band` function used to generate the FIR filter coefficients.

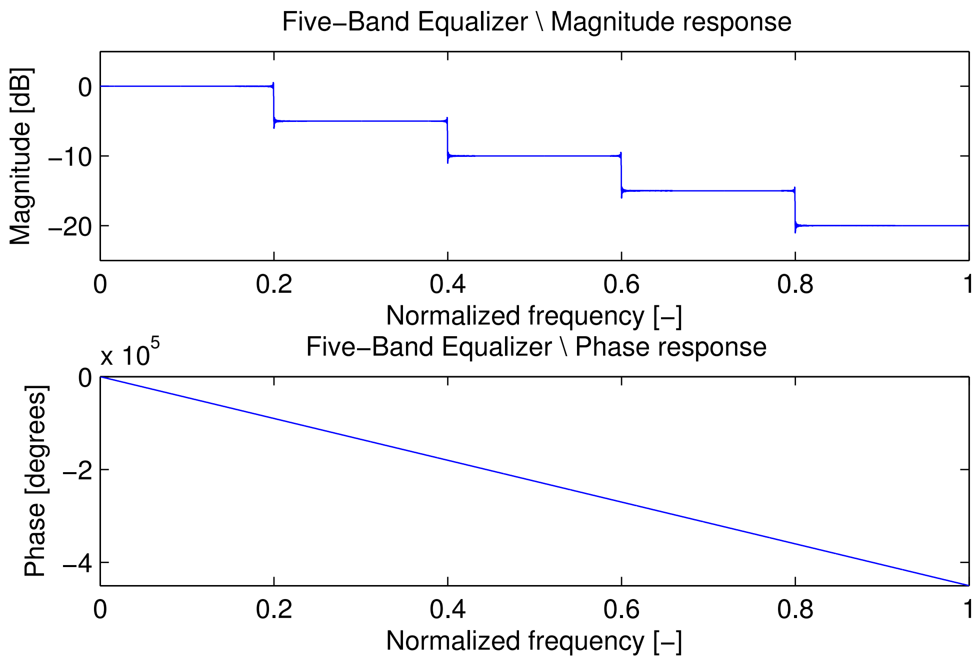

Figure 9: Magnitude and phase response of the transfer function of a five-band FIR equalizer with band gains decreasing in steps of 5 dB from the lowest to the highest normalized frequency. Ringing at the edges is caused by the use of a rectangular window in this case. See Figure 8. The filter order in this case is 4999 (IR length 5000).

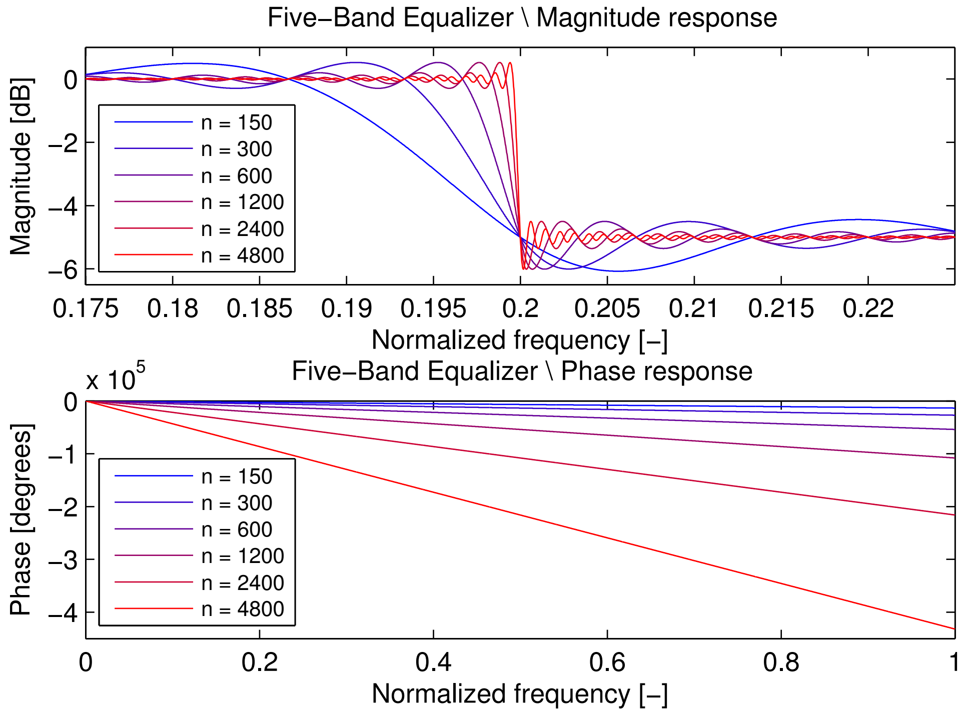

Figure 10: Equalizer frequency response (rectangular window function) for several IR lengths n. Ringing caused by the discontinuous rectangular window is clear and increases with the order. The phase response is linear, since the filter type is FIR and had zero-phase before translation.

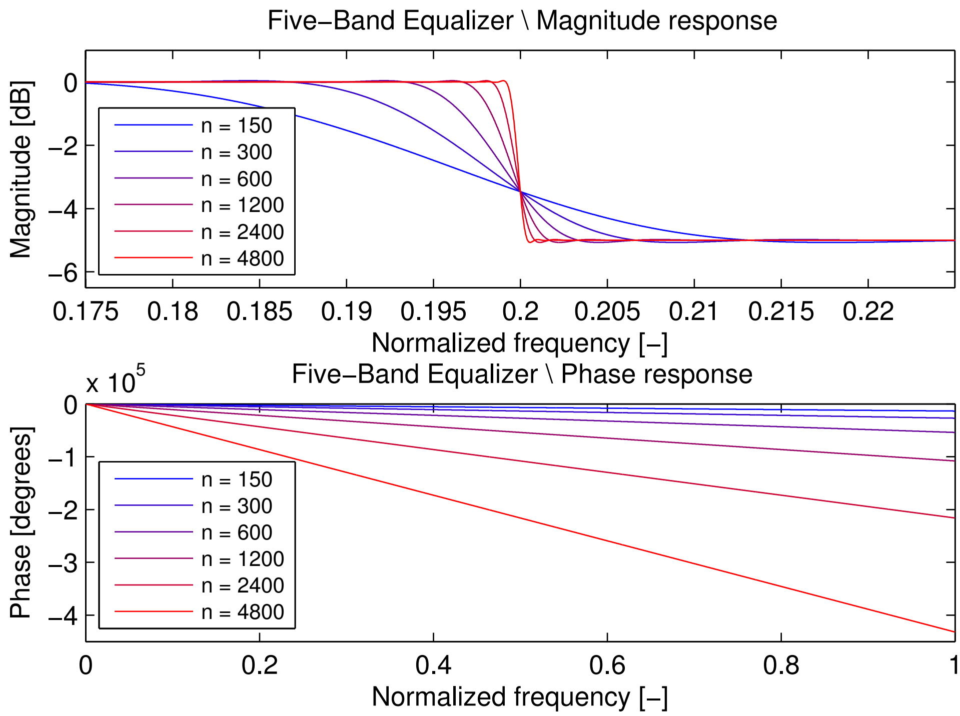

Figure 11: Equalizer frequency response (using Hann window function) for several IR lengths n. Compare with Figure 10 to see the clear improvement in smoothness of the magnitude response. Note also that the slope of the phase response is proportional to the IR length (also Figure 10). This is also as expected since a time translation by τ corresponds to a phase shift proportional to τ.

Higher order results in a greater phase slope and therefore more initial delay (besides the cost), but the approximation is much better. The relations between order, phase and approximation are illustrated in Figure 10 (rectangular window function) and Figure 11 (Hann window function). A filter order of 499 (corresponding to IR length $n = 500$) may be chosen for practical purposes (e.g. equalizing music), since an order of this size gives a somehat reasonable approximation without too many samples (less processing time, less memory usage and less delay as benefits). The magnitude response in Figure 11 show that this choice is middle ground.

Zero-padding the IR increases samples in the time domain and hence in the frequency domain. The spectral resolution increases with the number of samples. By comparing the resulting plots this effect was observed in the magnitude response (but not in the phase response however). Choosing a smoother window gives a less pronounced effect (since there is less ringing).

The result of processing white noise using the equalizer settings in Figure 9 is shown in Figure 12. As expected, a stepwise descent corresponding to the magnitude response can be clearly seen.

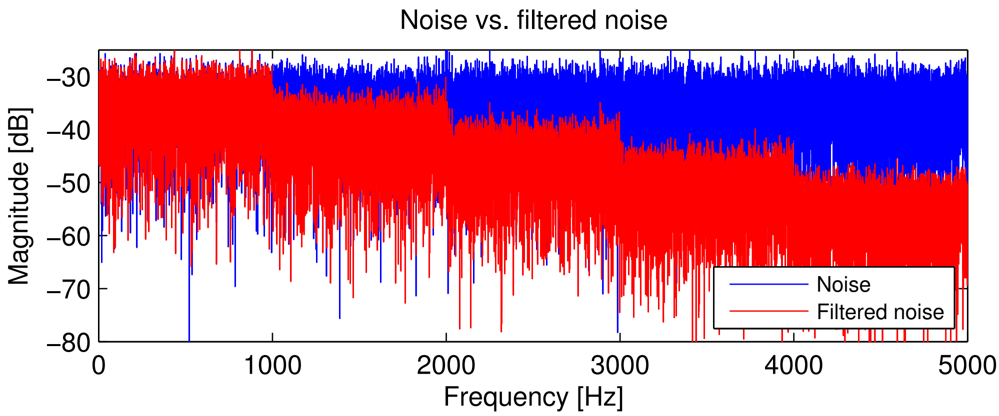

Figure 12: White noise signal filtered using the frequency response from Figure 9 (order 4999). The expected descent by 5 dB for each step is clearly visible in the output spectrum.

The result of processing `piano.wav` using the equalizer settings in Figure 9 is shown in Figure 13. A stepwise descent is no longer clear, but attenuation appears to increase steadily with frequency.

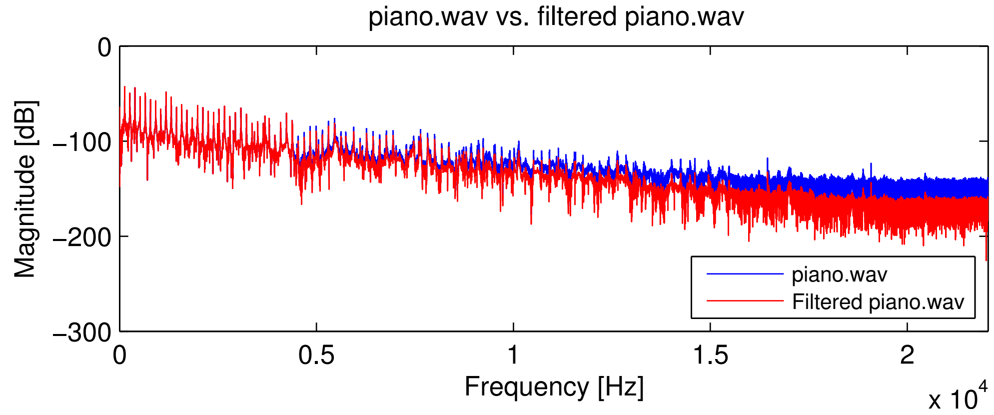

Figure 13: Sound `piano.wav` filtered using the frequency response from Figure 9 (order 4999). Attenuation increases steadily in accordance with the magnitude response of the applied filter.

||||3 The grandparents and DSP

3.1) Granma rolling with Maadlab My bored grandmother fires up *Matlab* and generates a linear sweep from $50Hz$ to $5kHz$ with a sampling frequency of $5kHz$.

1,a What I expect to hear Assuming *Matlab* and my soundcard likes eachother and can upsample and filter away any frequencies above half the sampling frequency, I expect to hear a cosine sweep from $50Hz$ to $2500Hz$ and then (with opposite phase) back again from $2500Hz$ to $0Hz$. This sweep has been illustrated in fig. 14.

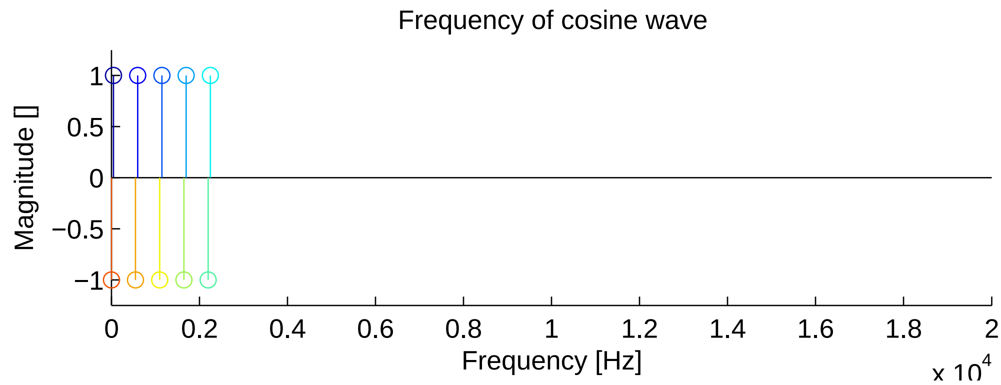

Figure 14: Linear sweep from $50Hz$ to $5kHz$ with a sampling frequency of $5kHz$ an brickwall filter at $2.5kHz$. Each color represents the frequency response at a different time, starting with blue and mooving to red.

Now I know that is probably not the case (by listening this is quite easy to check). Neither *Matlab* or my soundcard has a brickwall lowpass filter at $2500Hz$, so what I hear is actually multipe sweeps moving up and down. The audible range of the first 25% of the sweep is illustrated in fig 15. Or with a zero-order hold *DAC (digital to analog converter)* : 16.

1,a What my parrents expected was of course to actually hear a sweep from $50Hz$ to $5kHz$, because they do not know about the nyquist theorem and frequency folding. If I was to explain this to them, I'd probably tell them it's like that effect when you photograph a checkered shirt and get those wavy patterns instead.

1,a Explanation The reason for this frequency folding is sampling in the time domain, which leads to preodicity in the frequency domain. This leads artefacts if care is not taken to ensure that all frequencies stay inside their "period" in the frequency spectrum, but it also leads to artefacts if one is not careful when outputting the signal through ex. a *DAC*.

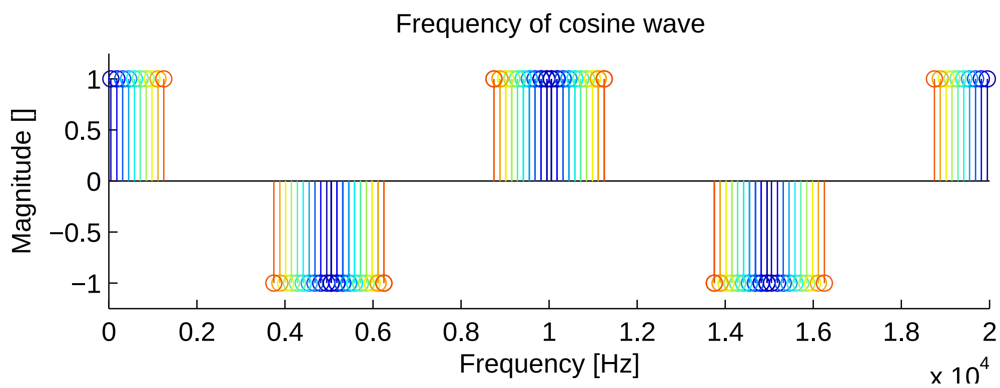

Figure 15: Linear sweep from $50Hz$ to $25\% \cdot 5kHz$ with a sampling frequency of $5kHz$. Each color represents the frequency response at a different time, starting with blue and mooving to red.

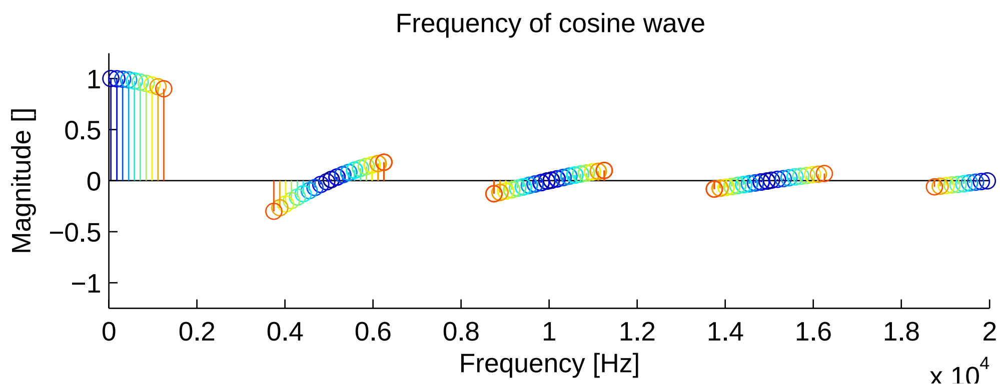

Figure 16: Linear sweep from $50Hz$ to $25\% \cdot 5kHz$ with a sampling frequency of $5kHz$ run through a zero-order hold DAC. Each color represents the frequency response at a different time, starting with blue and mooving to red.

||||1 Source code for tasks

../src/carsten/ha9.m

```
function Hd = butt
%ELLIPT Returns a discrete-time filter object.

%
% MATLAB Code
% Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox 6.18.
%
% Generated on: 09-Nov-2014 14:03:04
%

% Elliptic Bandpass filter designed using FDESIGN.BANDPASS.

% All frequency values are normalized to 1.

Fstop1 = 0.1;     % First Stopband Frequency
Fpass1 = 0.2;     % First Passband Frequency
Fpass2 = 0.3;     % Second Passband Frequency
Fstop2 = 0.4;     % Second Stopband Frequency
Astop1 = 100;     % First Stopband Attenuation (dB)
Apass  = 2;       % Passband Ripple (dB)
Astop2 = 100;     % Second Stopband Attenuation (dB)
match  = 'pass';  % Band to match exactly

% Construct an FDESIGN object and call its ELLIP method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2);
Hd = design(h, 'butter', 'MatchExactly', match);

% [EOF]
```

../src/soren/butt.m

```
function Hd = ellipt
%ELLIPT Returns a discrete-time filter object.

%
% MATLAB Code
% Generated by MATLAB(R) 8.0 and the Signal Processing Toolbox 6.18.
%
% Generated on: 09-Nov-2014 14:03:04
%

% Elliptic Bandpass filter designed using FDESIGN.BANDPASS.

% All frequency values are normalized to 1.

Fstop1 = 0.1;     % First Stopband Frequency
Fpass1 = 0.2;     % First Passband Frequency
Fpass2 = 0.3;     % Second Passband Frequency
Fstop2 = 0.4;     % Second Stopband Frequency
Astop1 = 100;     % First Stopband Attenuation (dB)
Apass  = 2;       % Passband Ripple (dB)
Astop2 = 100;     % Second Stopband Attenuation (dB)
match  = 'both';  % Band to match exactly

% Construct an FDESIGN object and call its ELLIP method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2);
Hd = design(h, 'ellip', 'MatchExactly', match);

% [EOF]
```

../src/soren/ellipt.m

```
clear all
close all

% set colormap
col = summer( 5 )';
col = col*0.95;
%% 1 How finite should it be

%% 1.1 butter
%actually we use an elliptic filter since ripple in both stop and passband
%is accepted

uncut = ellipt(); % genereate filter object from matlab toolbox
uncut_ir = impz( uncut );


%do some plots:
zplane( uncut );
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 15])
saveas(gcf,'./pics/1-1-uncut-zplane.eps','psc2')

% frequency response
[h,w] =  freqz( uncut );
plot( w/pi, mag2db(abs(h)) )
hold on
% plot bounds
plot( [0,0.1,0.1], [-100,-100,10], 'r' )
plot( [0.2,0.2,0.3,0.3], [-130,-2,-2,-130], 'r' )
plot( [0.2,0.2,0.3,0.3], [10,0,0,10], 'r' )
plot( [0.4,0.4,1], [10,-100,-100], 'r' )
legend('Original IIR response',...
    'Specification boudaries');
xlabel( 'Frequency [pi rad/sample]' )
ylabel( 'Magnitude [dB]' )
title( 'Frequency response of IIR filter' )
axis( [0,1,-130,10] )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 10])
saveas(gcf,'./pics/1-1-uncut-fresponse.eps','psc2')


%impulse response
m = impz( uncut );
figure; plot( m(1:end/3 ) )

xlabel( 'Sample []' )
ylabel( 'Value []' )
title( 'Impulse response' )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 5])
saveas(gcf,'./pics/1-1-uncut-impresponse.eps','psc2')
axis tight



%% Lets start cutting:

%get maximum value:
maxval = max( abs( uncut_ir ) );

%significance %
sig = 0.1;
```

```
%make list of max of remaining response:
maxrest = uncut_ir;
for i = 1:length(maxrest)
    maxrest(i) = max( abs( uncut_ir(i:end) ) );
    if maxrest(i) < maxval*sig
        break; %stop here, i is now the index, where all samples [i:inf[ < max
    end
end

%cut
cut_ir = uncut_ir(1:i);

%extra cuts:
cut_ir_75 = uncut_ir( 1:(i*0.75) );
cut_ir_60 = uncut_ir( 1:(i*0.60) );
cut_ir_40 = uncut_ir( 1:(i*0.40) );
cut_ir_10 = uncut_ir( 1:(i*0.10) );

%equalize length for easy plotting
n = 2048;
cut_ir = [ cut_ir; zeros( n - length( cut_ir ), 1 ) ];
cut_ir_75 = [ cut_ir_75; zeros( n - length( cut_ir_75 ), 1 ) ];
cut_ir_60 = [ cut_ir_60; zeros( n - length( cut_ir_60 ), 1 ) ];
cut_ir_40 = [ cut_ir_40; zeros( n - length( cut_ir_40 ), 1 ) ];
cut_ir_10 = [ cut_ir_10; zeros( n - length( cut_ir_10 ), 1 ) ];

% make plots
cut_ir_h = mag2db( abs( fft( cut_ir ) ) );
cut_ir_h = cut_ir_h(1:end/2);
cut_ir_75_h = mag2db( abs( fft( cut_ir_75 ) ) );
cut_ir_75_h = cut_ir_75_h(1:end/2);
cut_ir_60_h = mag2db( abs( fft( cut_ir_60 ) ) );
cut_ir_60_h = cut_ir_60_h(1:end/2);
cut_ir_40_h = mag2db( abs( fft( cut_ir_40 ) ) );
cut_ir_40_h = cut_ir_40_h(1:end/2);
cut_ir_10_h = mag2db( abs( fft( cut_ir_10 ) ) );
cut_ir_10_h = cut_ir_10_h(1:end/2);


figure;
plot( linspace(0,1,length(cut_ir_h)), cut_ir_h, 'Color', col(:,1) );
hold on
plot( linspace(0,1,length(cut_ir_h)), cut_ir_75_h, 'Color', col(:,2) );
plot( linspace(0,1,length(cut_ir_h)), cut_ir_60_h, 'Color', col(:,3) );
plot( linspace(0,1,length(cut_ir_h)), cut_ir_40_h, 'Color', col(:,4) );
plot( linspace(0,1,length(cut_ir_h)), cut_ir_10_h, 'Color', col(:,5) );
xlabel( 'Frequency [pi rad/sample]' )
ylabel( 'Magnitude [dB]' )
title( 'Frequency response of cut IIR filter' )
plot( [0,0.1,0.1], [-100,-100,10], 'r' )
plot( [0.2,0.2,0.3,0.3], [-130,-2,-2,-130], 'r' )
plot( [0.2,0.2,0.3,0.3], [10,0,0,10], 'r' )
plot( [0.4,0.4,1], [10,-100,-100], 'r' )
legend('Length 100% of 10% magnitude mark',...
    'Length 75% of 10% magnitude mark',...
    'Length 60% of 10% magnitude mark',...
    'Length 40% of 10% magnitude mark',...
    'Length 10% of 10% magnitude mark',...
    'Specification boudaries' );
axis( [0,1,-130,10] )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 10])
saveas(gcf,'./pics/1-1-cut-fresponse.eps','psc2')
%its baad

%% with windows
```

```

%make windows
window = hann( i*2 );
window = window( end/2+1:end );
window_75 = hamming( i*2*0.75 );
window_75 = window_75( end/2+1:end );
window_60 = hamming( i*2*0.60 );
window_60 = window_60( end/2+1:end );
window_40 = hamming( i*2*0.40 );
window_40 = window_40( end/2+1:end );
window_10 = hamming( i*2*0.10 );
window_10 = window_10( end/2+1:end );

%recut
cut_ir = uncut_ir(1:i);
cut_ir_75 = uncut_ir( 1:(i*0.75) );
cut_ir_60 = uncut_ir( 1:(i*0.60) );
cut_ir_40 = uncut_ir( 1:(i*0.40) );
cut_ir_10 = uncut_ir( 1:(i*0.10) );

%zero pad
cut_ir = [ cut_ir .* window; zeros( n - length( cut_ir ), 1 ) ];
cut_ir_75 = [ cut_ir_75 .* window_75; zeros( n - length( cut_ir_75 ), 1 ) ];
cut_ir_60 = [ cut_ir_60 .* window_60; zeros( n - length( cut_ir_60 ), 1 ) ];
cut_ir_40 = [ cut_ir_40 .* window_40; zeros( n - length( cut_ir_40 ), 1 ) ];
cut_ir_10 = [ cut_ir_10 .* window_10; zeros( n - length( cut_ir_10 ), 1 ) ];

% make plots
cut_ir_h = mag2db( abs( fft( cut_ir ) ) );
cut_ir_h = cut_ir_h(1:end/2);
cut_ir_75_h = mag2db( abs( fft( cut_ir_75 ) ) );
cut_ir_75_h = cut_ir_75_h(1:end/2);
cut_ir_60_h = mag2db( abs( fft( cut_ir_60 ) ) );
cut_ir_60_h = cut_ir_60_h(1:end/2);
cut_ir_40_h = mag2db( abs( fft( cut_ir_40 ) ) );
cut_ir_40_h = cut_ir_40_h(1:end/2);
cut_ir_10_h = mag2db( abs( fft( cut_ir_10 ) ) );
cut_ir_10_h = cut_ir_10_h(1:end/2);

figure;
plot( linspace(0,1,length(cut_ir_h)), cut_ir_h, 'Color', col(:,1) );
hold on
plot( linspace(0,1,length(cut_ir_h)), cut_ir_75_h, 'Color', col(:,2) );
plot( linspace(0,1,length(cut_ir_h)), cut_ir_60_h, 'Color', col(:,3) );
plot( linspace(0,1,length(cut_ir_h)), cut_ir_40_h, 'Color', col(:,4) );
plot( linspace(0,1,length(cut_ir_h)), cut_ir_10_h, 'Color', col(:,5) );
xlabel( 'Frequency [pi rad/sample]' )
ylabel( 'Magnitude [dB]' )
title( 'Frequency response of cut, windowed IIR filter' )
plot( [0,0.1,0.1], [-100,-100,10], 'r' )
plot( [0.2,0.2,0.3,0.3], [-130,-2,-2,-130], 'r' )
plot( [0.2,0.2,0.3,0.3], [10,0,0,10], 'r' )
plot( [0.4,0.4,1], [10,-100,-100], 'r' )
legend('Length 100% of 10% magnitude mark',...
    'Length 75% of 10% magnitude mark',...
    'Length 60% of 10% magnitude mark',...
    'Length 40% of 10% magnitude mark',...
    'Length 10% of 10% magnitude mark',...
    'Specification boudaries');
axis( [0,1,-130,10] )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 10])
saveas(gcf,'./pics/1-1-cut-window-fresponse.eps','psc2')
```

../src/soren/ho9_ex1.m

```
clear all
```

```
close all

plotN = 10;
col = jet( plotN )';
col = col*0.95;


%% Grandma's soo MAADLAB, yeah!

%% What parents expects:
%parameters
Fs = 44100;
T = 4;

%generate stuff
t = 0:1/Fs:T;
s = chirp( t, 50, T, 5000 );

%this is what she expets to hear:
%sound( s*0.1, 44100 ); %Turn down for what? Eh ok.
%or as a plot:
f = linspace( 50, 5000, plotN );
m = ones(1,length(f));

figure; hold on;
for i = 1:length(f)
    stem( f(i), m(i), 'color', col(:,i) );
end
axis( [0, 20000, -1.25, 1.25] )

%% What you expect, because that PC was damn expensive
%  so that soundcard better perform reeeal nice
%parameters
Fs = 5000;
T = 4;

%generate stuff
t = 0:1/Fs:T;
s = chirp( t, 50, T, 5000 );

%this is what we hear:
%sound( s*0.1, 5000 ); %Ew

%or as a plot (first second)
f = linspace( 50, 5000, plotN );
m = ones(1,length(f));
f = [f, linspace( Fs-50, Fs-5000, plotN )];
m = [m,-m];

figure; hold on;
for i = 1:length(f)
    if f(i) <= 2500
    stem( f(i), m(i), 'color', col(:,mod(i-1,plotN)+1) );
    end
end
axis( [0, 20000, -1.25, 1.25] )
xlabel( 'Frequency [Hz]' )
ylabel( 'Magnitude []' )
title( 'Frequency of cosine wave' )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 5])
saveas(gcf,'./pics/3-1-sweep-fold-brick.eps','psc2')

%% What w'all  hear: ('caus that shit was not made for an Fs of 5000Hz)
%parameters
Fs = 5000;
T = 4;
```

```
%generate stuff
t = 0:1/Fs:T;
s = chirp( t, 50, T, 5000 );

%this is what we hear:
%sound( s*0.1, 5000 ); %Ew

%or as a plot (first second)
f = linspace( 50, 5000*0.25, plotN );
m = ones(1,length(f));
f = [f, linspace( Fs-50, Fs-5000*0.25, plotN )];
m = [m,-m];

f = [f,f+Fs];
m = [m,-m];
f = [f,f+Fs];
m = [m,-m];
f = [f,f+Fs];
m = [m,-m];

figure; hold on;
for i = 1:length(f)
    stem( f(i), m(i), 'color', col(:,mod(i-1,plotN)+1) );
end
axis( [0, 20000, -1.25, 1.25] )
xlabel( 'Frequency [Hz]' )
ylabel( 'Magnitude []' )
title( 'Frequency of cosine wave' )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 5])
saveas(gcf,'./pics/3-1-sweep-fold-nofilt.eps','psc2')

m = m.*sinc(f/5000); % zero order hold DAC

figure; hold on;
for i = 1:length(f)
    stem( f(i), m(i), 'color', col(:,mod(i-1,plotN)+1) );
end
axis( [0, 20000, -1.25, 1.25] )
xlabel( 'Frequency [Hz]' )
ylabel( 'Magnitude []' )
title( 'Frequency of cosine wave' )
set(gca,'Fontsize',10)
set(gcf,'paperunits','centimeters','Paperposition',[0 0 15 5])
saveas(gcf,'./pics/3-1-sweep-fold-zero.eps','psc2')
```

../src/soren/ho9_ex3.m

```
%% 2 From bass/treble to an equalizer

%% 2.1 Five knobs to turn

n = 5000;
G1 = db2mag(0);
G2 = db2mag(-5);
G3 = db2mag(-10);
G4 = db2mag(-15);
G5 = db2mag(-20);

h = equalizer5band(G1, G2, G3, G4, G5, n, 0);

[H, f] = freqz(h, 1, 2^16);
f=f/pi;

figure
subplot(2,1,1)
```

```
plot(f, mag2db(abs(H)));
xlim([0 1])
ylim([-25 5])
title('Five-Band Equalizer \\ Magnitude response')
xlabel('Normalized frequency [-]')
ylabel('Magnitude [dB]')
subplot(2,1,2)
plot(f, (180/pi)*unwrap(angle(H)));
xlim([0 1])
ylim([-4.5*10^5 0])
title('Five-Band Equalizer \\ Phase response')
xlabel('Normalized frequency [-]')
ylabel('Phase [degrees]')

%%

G1 = db2mag(0);
G2 = db2mag(-5);
G3 = db2mag(-10);
G4 = db2mag(-15);
G5 = db2mag(-20);

h_h150 = equalizer5band(G1, G2, G3, G4, G5, 150, 1);
h_h300 = equalizer5band(G1, G2, G3, G4, G5, 300, 1);
h_h600 = equalizer5band(G1, G2, G3, G4, G5, 600, 1);
h_h1200 = equalizer5band(G1, G2, G3, G4, G5, 1200, 1);
h_h2400 = equalizer5band(G1, G2, G3, G4, G5, 2400, 1);
h_h4800 = equalizer5band(G1, G2, G3, G4, G5, 4800, 1);

[H_h150, f_h150] = freqz(h_h150, 1, 2^16);
f_h150=f_h150/pi;
[H_h300, f_h300] = freqz(h_h300, 1, 2^16);
f_h300=f_h300/pi;
[H_h600, f_h600] = freqz(h_h600, 1, 2^16);
f_h600=f_h600/pi;
[H_h1200, f_h1200] = freqz(h_h1200, 1, 2^16);
f_h1200=f_h1200/pi;
[H_h2400, f_h2400] = freqz(h_h2400, 1, 2^16);
f_h2400=f_h2400/pi;
[H_h4800, f_h4800] = freqz(h_h4800, 1, 2^16);
f_h4800=f_h4800/pi;

figure
subplot(2,1,1)
plot(f_h150, mag2db(abs(H_h150)), 'Color', [0.00 0 1.00]);
hold on
plot(f_h300, mag2db(abs(H_h300)), 'Color', [0.20 0 0.80]);
hold on
plot(f_h600, mag2db(abs(H_h600)), 'Color', [0.40 0 0.60]);
hold on
plot(f_h1200, mag2db(abs(H_h1200)), 'Color', [0.60 0 0.40]);
hold on
plot(f_h2400, mag2db(abs(H_h2400)), 'Color', [0.80 0 0.20]);
hold on
plot(f_h4800, mag2db(abs(H_h4800)), 'Color', [1.00 0 0.00]);
hold on
xlim([0.175 0.225])
ylim([-6.5 1])
title('Five-Band Equalizer \\ Magnitude response')
xlabel('Normalized frequency [-]')
ylabel('Magnitude [dB]')
p = legend('n = 150','n = 300','n = 600','n = 1200','n = 2400','n = 4800', 3);
set(p,'FontSize', 8);
subplot(2,1,2)
plot(f_h150, (180/pi)*unwrap(angle(H_h150)), 'Color', [0.00 0 1.00]);
hold on
plot(f_h300, (180/pi)*unwrap(angle(H_h300)), 'Color', [0.20 0 0.80]);
```

```
hold on
plot(f_h600, (180/pi)*unwrap(angle(H_h600)), 'Color', [0.40 0 0.60]);
hold on
plot(f_h1200, (180/pi)*unwrap(angle(H_h1200)), 'Color', [0.60 0 0.40]);
hold on
plot(f_h2400, (180/pi)*unwrap(angle(H_h2400)), 'Color', [0.80 0 0.20]);
hold on
plot(f_h4800, (180/pi)*unwrap(angle(H_h4800)), 'Color', [1.00 0 0.00]);
hold on
xlim([0 1])
ylim([-4.5*10^5 0])
title('Five-Band Equalizer \\ Phase response')
xlabel('Normalized frequency [-]')
ylabel('Phase [degrees]')
p = legend('n = 150','n = 300','n = 600','n = 1200','n = 2400','n = 4800', 3);
set(p,'FontSize', 8);

%%

L = length(h);

h_2  = [h, zeros(1, L)]; n_2 = 2*n;
h_10 = [h, zeros(1, 9*L)]; n_10 = 10*n;

f_2 = -1:2/n_2:1-2/n_2;
H_2 = fftshift(fft(h_2));

f_10 = -1:2/n_10:1-2/n_10;
H_10 = fftshift(fft(h_10));

figure
subplot(2,1,1)
plot(f_2, mag2db(abs(H_2)));
xlim([0.195 0.205])
ylim([-10 5])
title('Five-Band Equalizer \\ Magnitude response')
xlabel('Normalized frequency [-]')
ylabel('Magnitude [dB]')
subplot(2,1,2)
plot(f_2, (180/pi)*unwrap(angle(H_2)));
xlim([0 1])
ylim([-9*10^5 -4.5*10^5])
title('Five-Band Equalizer \\ Phase response')
xlabel('Normalized frequency [-]')
ylabel('Amplitude [-]')

figure
subplot(2,1,1)
plot(f_10, mag2db(abs(H_10)));
xlim([0.195 0.205])
ylim([-10 5])
title('Five-Band Equalizer \\ Magnitude response')
xlabel('Normalized frequency [-]')
ylabel('Magnitude [dB]')
subplot(2,1,2)
plot(f_10, (180/pi)*unwrap(angle(H_10)));
xlim([0 1])
ylim([-9*10^5 -4.5*10^5])
title('Five-Band Equalizer \\ Phase response')
xlabel('Normalized frequency [-]')
ylabel('Amplitude [-]')

%%

figure
subplot(2,1,1)
plot(f_2, mag2db(abs(H_2)));
```

```
xlim([0.195 0.205])
ylim([-10 5])
title('Five-Band Equalizer x2 length zero padding \\ Magnitude response')
xlabel('Normalized frequency [-]')
ylabel('Magnitude [dB]')
subplot(2,1,2)
plot(f_10, mag2db(abs(H_10)));
xlim([0.195 0.205])
ylim([-10 5])
title('Five-Band Equalizer x10 length zero padding \\ Magnitude response')
xlabel('Normalized frequency [-]')
ylabel('Magnitude [dB]')

%%

%parameter setup
T = 5;
fs = 10000;
t = 0:1/fs:T-1/fs;
f = -fs/2:1/T:fs/2-1/T;

%generate noise
r = randn(fs*T, 1);
x = r/0.99*max(r);

%filter and transform
y = filter(h, 1, x);
X = fftshift(fft(x))/length(x);
Y = fftshift(fft(y))/length(y);

%plot noise vs. filtered noise
figure
plot(f, mag2db(abs(X)), 'b')
hold on
plot(f, mag2db(abs(Y)), 'r')
hold on
xlim([0 fs/2])
ylim([-80 -25])
title('Noise vs. filtered noise')
xlabel('Frequency [Hz]')
ylabel('Magnitude [dB]')
p = legend('Noise','Filtered noise', 4);
set(p,'FontSize', 8);

%sound(x, fs)
%sound(y, fs)

%%

%data setup
[s, fs] = audioread(['C:\Users\Aztar\Documents\MATLAB' filesep 'piano.wav']);
T = length(s)/fs;
t = 0:1/fs:T-1/fs;
f = -fs/2:fs/(length(s)-1):fs/2;

%filter and transform
z = filter(h, 1, s);
S = fftshift(fft(s))/length(s);
Z = fftshift(fft(z))/length(z);

figure
plot(f, mag2db(abs(S)), 'b');
hold on
plot(f, mag2db(abs(Z)), 'r');
hold on
xlim([0 fs/2])
title('piano.wav vs. filtered piano.wav')
```

```
xlabel('Frequency [Hz]')
ylabel('Magnitude [dB]')
p = legend('piano.wav','Filtered piano.wav', 4);
set(p,'FontSize', 8);
```

../src/david/david9.m

```
function [h] = equalizer5band(G1, G2, G3, G4, G5, n, W);
%INPUT: Five band gains: G1, G2, G3, G4, G5. Approximate length: n. Window type: W.
%OUTPUT: Impulse response/FIR filter coefficients: h.
m = round(n/10);
H = [G1*ones(1, m) , G2*ones(1, m) , ...
     G3*ones(1, m) , G4*ones(1, m) , ...
     G5*ones(1, m) , G5*ones(1, m) , ...
     G4*ones(1, m) , G3*ones(1, m) , ...
     G2*ones(1, m) , G1*ones(1, m)];
h = ifftshift(ifft(H, 'symmetric'));
L = length(h);
if W == 0 %really unnecessary
   h = h; %do nothing
elseif W == 1
   h = h.*hanning(L)';
elseif W == 2
   h =h.*hamming(L)';
end
end %eof
```

../src/david/equalizer5band.m

||||2 Time spent and link to repository

Time spent

- 10h Søren
- 10h Carsten
- 20h David

Repository all sourcecode (including report) is available at: `https://github.com/skrogh/31606-handson9`