Digital Filter Design Using Optimization

MATS VIBERG

Chalmers University of Technology

Department of Signals and Systems

SE-412 96 Göteborg, Sweden

Email: viberg@chalmers.se

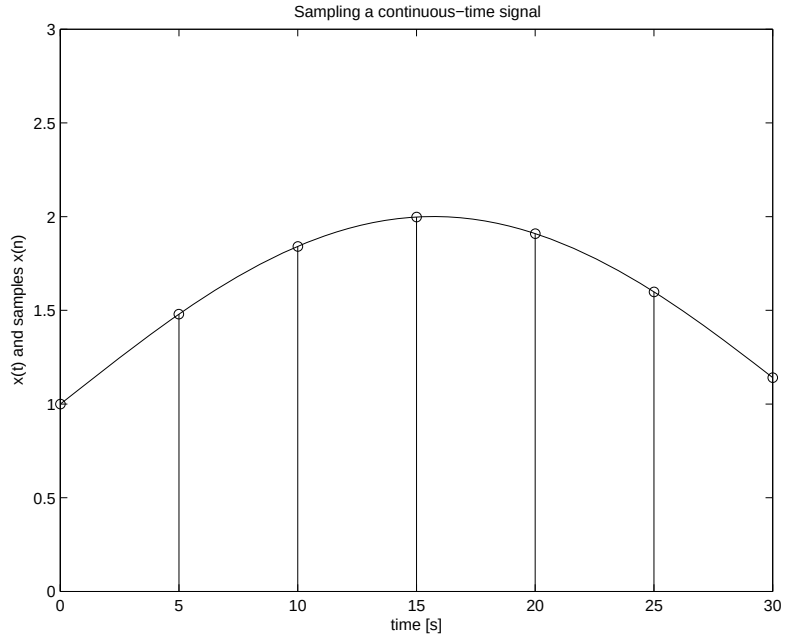
Mini-project for the Applied Optimization Course, Spring 2008.

Outline

- Introduction to Digital Filters
  - Sampling and digital signals
  - Digital filters
  - Frequency response
- Digital Filter Design
  - Specifications
  - Linear Phase FIR Filters
  - Least-Squares Design
  - Minimax Design
- Optimization Tasks

Sampling

EX analog signal $x(t)$ sampled with period $T = 5$ s:



Digital signal $x(n),\ n = \ldots, -1, 0, 1, 2, \ldots$, represents $x(t)$ for $t = \ldots, -T, 0, T, 2T, \ldots$.

T sampling time [s], $f_{samp} = 1/T$ sampling frequency [Hz].

Digital Filters

Analog filter = continuous-time LTI-system: differential equation

Digital filter = discrete-time LTI-system: difference equation

First-order filter:

$$y(n) = -a_1 y(n-1) + b_0 x(n) + b_1 x(n-1)$$

Implemented in a computer, DSP, FPGA or digital ASIC

Time-Domain Characterization

First-order difference equation:

$$y(n) + a_1 y(n-1) = b_0 x(n) + b_1 x(n-1)$$

Impulse response $h(n)$ = filter output when $x(n) = \delta(n)$. I/O relation:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k)$$

- Causal filter if $h(n) = 0,\ n < 0$.
- Finite Impulse Response (FIR) if $h(n) = 0,\ n > M$ ($a_1 = 0$)
- Infinite Impulse Response (IIR) otherwise

Discrete-Time Fourier Transform

Frequency content of digital signals? Need ≥ 2 samples per period $\Rightarrow$ can only represent frequencies $\leq f_{samp}/2$ = *Nyquist frequency*.

DTFT (Discrete-Time Fourier Transform):

$$X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega Tn}, \quad \omega = 2\pi f \ [\text{rad/s}]$$

Equal to continuous-time FT if $x(t)$ contains no frequencies above Nyquist. Then: $x(t)$ can be perfectly reconstructed from samples $x(n)$!

If $x(t)$ has frequency content above $f_{samp}/2 \Rightarrow$ *aliasing* (folding distortion).

Frequency Response

Frequency domain: DTFT$\{x(n-k)\} = e^{-j\omega Tk}X(e^{j\omega}) \quad \Rightarrow$

$$Y(e^{j\omega T}) + a_1 e^{-j\omega T} Y(e^{j\omega T}) = b_0 X(e^{j\omega T}) + b_1 e^{-j\omega T} X(e^{j\omega T})$$

which implies

$$Y(e^{j\omega T}) = \frac{b_0 + b_1 e^{-j\omega T}}{1 + a_1 e^{-j\omega T}} X(e^{j\omega T}) = H(e^{j\omega T}) X(e^{j\omega T})$$

$H(e^{j\omega T}) = Y(e^{j\omega T})/X(e^{j\omega T}) = H(e^{j\Omega})$ is the *frequency response* of the filter!

Note: $H(e^{j\Omega})$ is the DTFT of the impulse response $h(n)$!

ω frequency in rad/s, $\Omega = \omega T$ normalized frequency in rad/sample.

Stability

A critical feature of a filter: **stability**!

A filter is BIBO-stable (Bounded Input Bounded Output) if $|x(n)| < C_x$ implies $|y(n)| < C_y$. A filter is BIBO-stable if

$$\sum_{k=-\infty}^{\infty} |h(k)| < \infty$$

The transfer function of a general (finite-dimensional) system is

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} = \frac{B(z)}{A(z)}$$

so $H(e^{j\Omega}) = H(z)|_{z=e^{j\Omega}}$. The **poles**, $p_i,\ i = 1, \ldots, N$ of an Nth order system are the zeros of $A(z)$, and the system is BIBO-stable if $|p_i| < 1\ \forall i$.

Frequency Response, cont'd

Filter response to a single frequency:

$$x(n) = \sin(\Omega n)$$

Stationary output (assuming stable filter):

$$y(n) = |H(e^{j\Omega})| \sin(\Omega n + \mathrm{Arg}\{H(e^{j\Omega})\})$$

The frequency response can obviously be interpreted as a complex frequency-dependent gain!

- Amplitude characteristics: $|H(e^{j\Omega})|$
- Phase characteristics: $\mathrm{Arg}\{H(e^{j\Omega})\}$

Choice of filter coefficients determines which frequencies to pass and which ones to block.

Digital Filters, cont'd

The spectrum $|X(e^{j\Omega})|^2$ is the energy of $x(n)$ per Hz.

The spectrum of $y(n)$ is related to that of $x(n)$ by

$$|Y(e^{j\Omega})|^2 = |H(e^{j\Omega})|^2 \, |X(e^{j\Omega})|^2$$

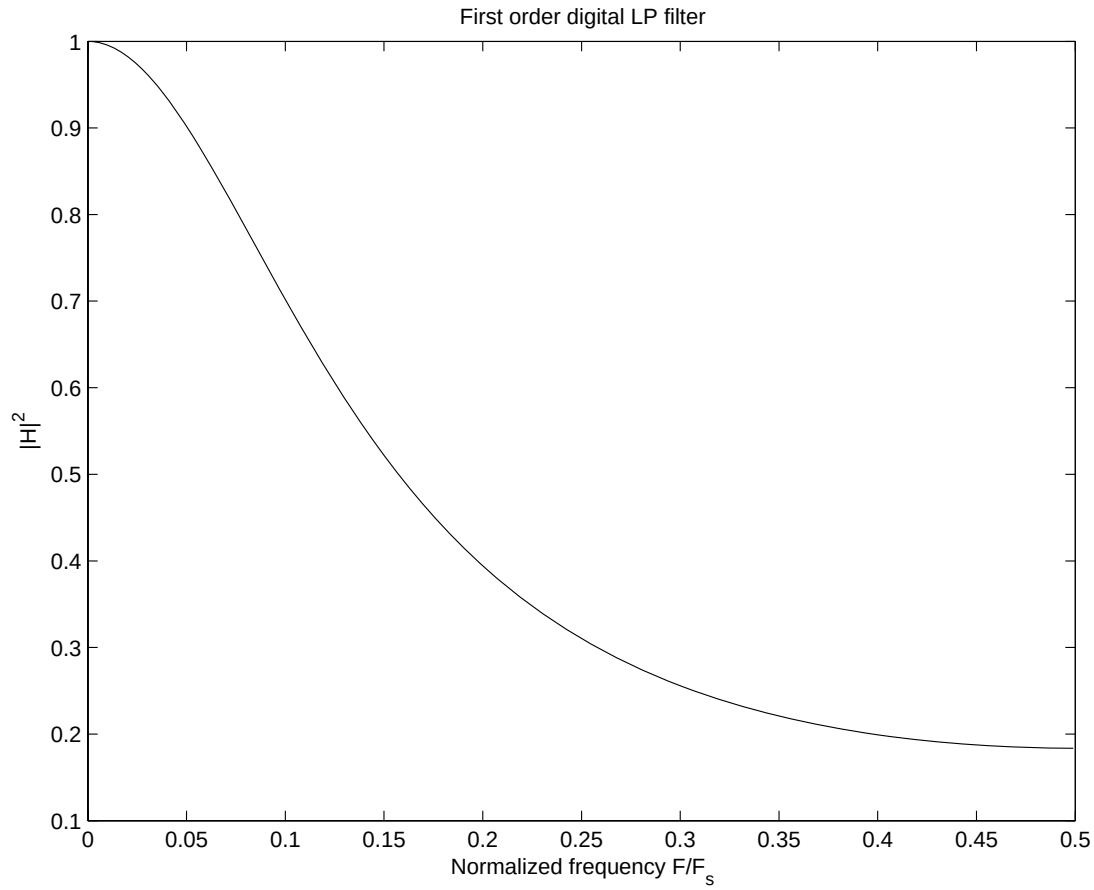
The filter changes the frequency content of a signal - we can suppress noise and interference!

Filter design = choice of coefficients $\{a_k\}_{k=1}^{N}$ and $\{b_k\}_{k=1}^{M}$ so that $|H(e^{j\Omega})|$ (and $\mathrm{Arg}\{H(e^{j\omega})\}$) meets specifications.

Filter design tools in Matlab: sptool, fdatool (graphic), fir1, firls, firpm, butter, cheby1, cheby2, etc.

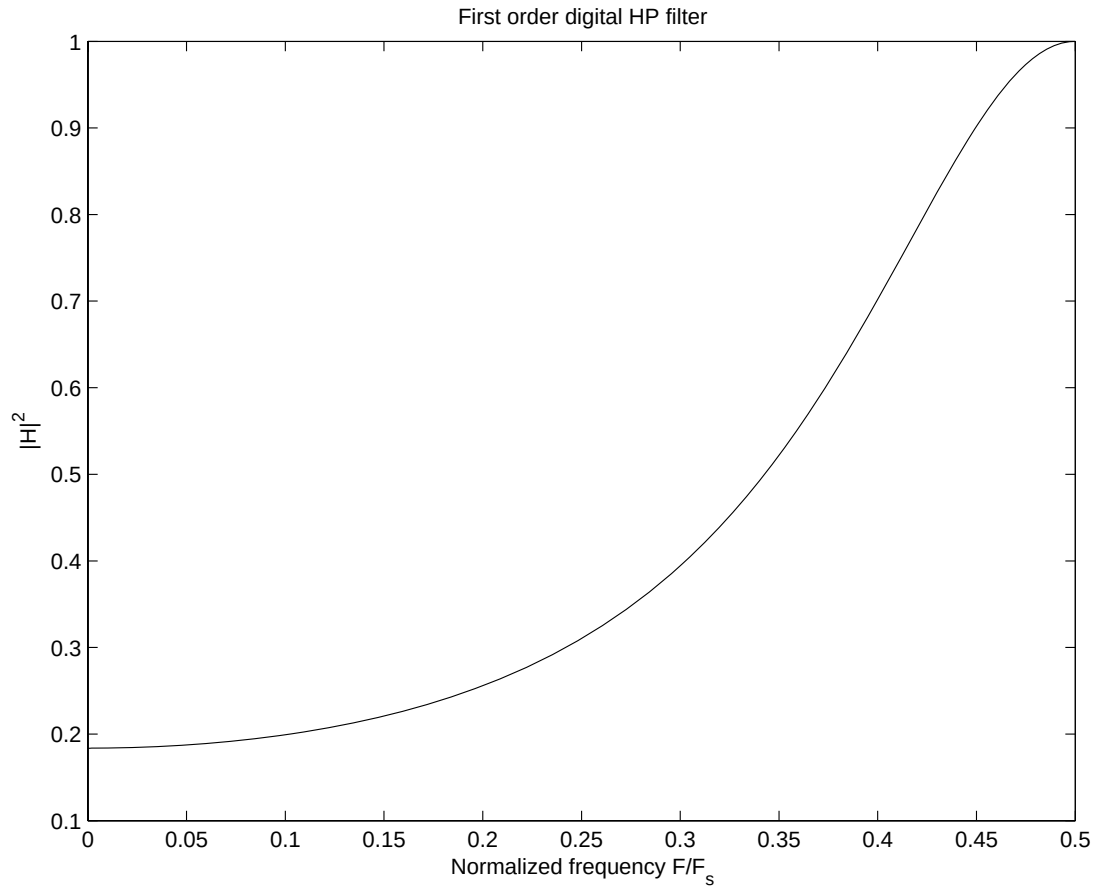
Digital Filters, cont'd

EX LP-filter $y(n) = 0.4y(n-1) + 0.6x(n)$



Digital Filters, cont'd

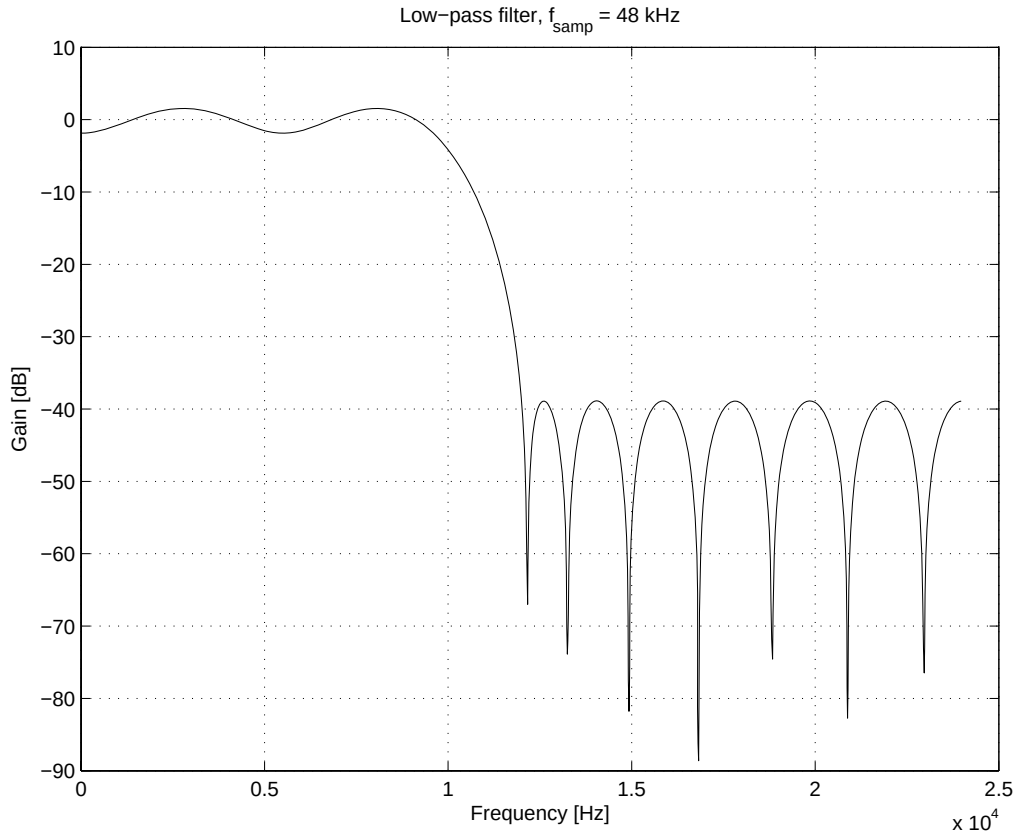
EX HP-filter $y(n) = -0.4y(n-1) + 0.6x(n-1)$



Digital Filters, summary

- Sampling $x(n) \sim x(t)\,,\ t = nT$
- DTFT $X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega Tn}$
- Digital filter (1st order): $y(n) + a_1 y(n-1) = b_0 x(n) + b_1 x(n-1)$
- Frequency domain: $Y(e^{j\Omega}) = H(e^{j\Omega})X(e^{j\Omega})\,,\ H(e^{j\Omega}) = \frac{b_0 + b_1 e^{-j\Omega}}{1 + a_1 e^{-j\Omega}}$
- Amplitude characteristics: $|H(e^{j\Omega})|$
- Phase characteristics: $\mathrm{Arg}\{H(e^{j\Omega})\}$
- A digital filter is stable if all poles are inside the unit circle

Filter Specifications (LP Filter)



Passband: $|H(e^{j2\pi fT}) - 1| \leq \epsilon_1$ for $0 \leq f \leq f_p$

Stopband: $|H(e^{j2\pi fT})| < \epsilon_2$, $f_s \leq f \leq f_{samp}/2$ (Hz)

FIR vs IIR Filters

FIR or IIR filter? A causal FIR filter with $M+1$ "taps" has:

$$H(e^{j\Omega}) = b_0 + b_1 e^{-j\Omega} + \cdots + b_M e^{-jM\Omega}$$

so $h(n) = b_n,\ n = 0, 1, \ldots, M$, and $h(n) = 0$ for $n < 0$ or $n > M$.

Pros and cons of FIR filters vs IIR filters:

+ $H(e^{j\Omega})$ and $h(n)$ are linear functions of b_i (but non-linear in a_i)
+ FIR filters are always stable (all poles are in the origin)
+ FIR filters have linear phase if $b_i = \pm b_{M-i}$ (symmetry or anti-symmetry)
- FIR filters need in general more coefficients to meet given specifications on $|H(e^{j\Omega})|$

In this project we focus on FIR filter design!

Linear Phase FIR Filters

Assume M is odd and $b_i = b_{M-i}$ (symmetry). Then:

$$H(e^{j\Omega}) = b_0(1+e^{-jM\Omega}) + b_1(e^{-j\Omega}+e^{-j(M-1)\Omega}) + \cdots = e^{-jM\Omega/2}A(\Omega)$$

where

$$A(\Omega) = 2b_0 \cos\frac{M\Omega}{2} + 2b_1 \cos\frac{(M-2)\Omega}{2} + \cdots + 2b_{(M-1)/2}\cos\frac{\Omega}{2}$$

is a real function! Thus, the phase $\mathrm{Arg}\{H(e^{j\Omega})\} = -(M/2)\Omega$ is linear in Ω - corresponds to a time-delay by $M/2$ samples and no phase distortion!

Similar calculations when $b_i = -b_{M-i}$ and/or M even - still linear phase, but real function $A(\Omega)$ slightly different.

Least-Squares FIR Filter Design

An FIR filter with (anti-)symmetric coefficients has "perfect" (linear) phase. We need only worry about the amplitude!

Natural and simple approach: specify desired response $A(\Omega, \mathbf{b})$ at a discrete set of frequencies $\{\Omega_k\}_{k=1}^{K}$

$$A(\Omega_k, \mathbf{b}) = A_d(\Omega_k),\ k = 1, 2, \ldots, K$$

where $A_d(\Omega_k)$ is the desired response and

$$A(\Omega, \mathbf{b}) = 2\left(b_0 \cos\frac{M\Omega}{2} + b_1 \cos\frac{(M-2)\Omega}{2} + \cdots + b_{(M-1)/2} \cos\frac{\Omega}{2}\right)$$

(for the symmetric case with M odd). The dependency on the $(M+1)/2$ filter parameters $\mathbf{b} = [b_0, \ldots, b_{(M-1)/2}]^T$ has been stressed.

LS FIR Filter Design, cont'd

Weighted Least-Squares filter design:

$$\mathbf{b}_{LS} = \arg\min_{\mathbf{b}} \sum_{k_1}^{K} W_k |A_d(\Omega_k) - A(\Omega_k, \mathbf{b})|^2$$

where W_k is a set of weights, used to emphasize certain frequencies or frequency bands.

Linear LS-problem $\Rightarrow$ explicit solution!

To solve this in, e.g. Matlab, express the magnitude response as

$$A(\Omega, \mathbf{b}) = \boldsymbol{\phi}^T(\Omega)\mathbf{b}$$

where

$$\boldsymbol{\phi}^T(\Omega) = 2[\cos\frac{M\Omega}{2}, \cos\frac{(M-2)\Omega}{2}, \ldots, \cos\frac{\Omega}{2}]$$

LS FIR Filter Design, cont'd

The LS problem can now be put in matrix-vector form as

$$\mathbf{b}_{LS} = \arg\min_{\mathbf{b}} \|\mathbf{a}_d - \mathbf{\Phi b}\|^2_{\mathbf{W}}$$

where

$$\begin{aligned} \mathbf{a}_d &= [A_d(\Omega_1), \ldots, A_d(\Omega_K)]^T \quad (K \times 1) \\ \mathbf{\Phi} &= [\boldsymbol{\phi}(\Omega_1), \ldots, \boldsymbol{\phi}(\Omega_K)]^T \quad (K \times (M+1)/2) \end{aligned}$$

and $\mathbf{W}$ is a diagonal matrix with diagonal elements W_k.

The weighted norm is:

$$\|\mathbf{x}\|^2_{\mathbf{W}} = \mathbf{x}^T\mathbf{W}\mathbf{x}$$

LS FIR Filter Design, cont'd

The normal equations are

$$\mathbf{\Phi}^T \mathbf{W} \mathbf{\Phi}\, \mathbf{b}_{LS} = \mathbf{\Phi}^T \mathbf{W} \mathbf{a}_d$$

and the LS-solution is

$$\mathbf{b}_{LS} = (\mathbf{\Phi}^T \mathbf{W} \mathbf{\Phi})^{-1} \mathbf{\Phi}^T \mathbf{W} \mathbf{a}_d$$

Never type this in Matlab, but use:

$$\mathbf{b}_{LS} = (\mathbf{W}^{1/2} \mathbf{\Phi}) \backslash (\mathbf{W}^{1/2} \mathbf{a}_d)$$

where $\mathbf{W}^{1/2}$ has $\sqrt{W_k}$ at the diagonal (generally a matrix square-root).

This is a numerically more stable implementation, using QR decomposition!

Minimax FIR Filter Design

LS approach simple, but the optimal (according to specifications) is a **minimax** design! Given desired response $A_d(\Omega)$, the weighted error is

$$E(\Omega, \mathbf{b}) = W(\Omega)\left(A_d(\Omega) - A(\Omega, \mathbf{b})\right)$$

where $W(\Omega)$ controls passband ripple vs stopband damping.

Minimax (Chebyshev) filter design:

$$\mathbf{b}_{PM} = \arg\min_{\mathbf{b}} \max_{\Omega \in \mathcal{O}} |E(\Omega, \mathbf{b})|$$

where $\mathcal{O}$ is the set of frequencies where specifications exist, normally the passband and stopband. "PM" stands for Parks and McClellan.

Minimax FIR Filter Design: Equiripple Property

"Easy" to see that the error $|E(\Omega, \mathbf{b}_{PM})|$ has equi-ripple - the maximum $|E(\Omega_i, \mathbf{b}_{PM})| = \delta$ is "generically" attained at $(M+3)/2$ frequencies Ω_i, with alternating sign of $E(\Omega)$. This is the "alternation theorem".

If the Ω_is were known, we could use this insight to find $\mathbf{b}_{PM}$ by solving a linear equation system ($(M+3)/2$ equations for $(M+3)/2$ unknowns, including δ). Parks and McClellan's algorithm iterates between solving for $\mathbf{b}$ and δ, and updating the Ω_is by finding all local maxima in an efficient way. In Matlab, this is called `firpm`. Your goal is to implement your own algorithm, and compare with `firpm`!

Exercises: FIR Filter Design

The tasks consist of implementing Matlab functions for LS (with weighting) and minimax FIR filter design. The functions should then be compared with the existing ones in Matlab. As a test example we use the following lowpass specifications (with $f_{samp} = 1$ Hz):

- Passband: $0 \leq f \leq 0.25$, corresponding to $0 \leq \Omega \leq \pi/2$
- Passband ripple: $20\log_{10}(1/\epsilon_1) < 3$ dB
- Transition band (don't care region): $0.25 < f < 0.3$
- Stopband: $0.3 \leq f \leq 0.5$
- Damping: $20\log_{10}(1/\epsilon_2) > 40$ dB

1. Warm-up. The simple averaging filter

$$y(n) = \frac{1}{M+1} \sum_{k=0}^{M} x(n-k)$$

corresponds to an FIR filter with $b_k = 1/(M+1),\ k = 0, \ldots, M$. Compute and display the corresponding frequency response, using the Matlab commands:

```
» M = 10; %%% Filter order
» h = ones(M+1,1)/(M+1); %%% Impulse response
» [H,w] = freqz(h,1,1024); %%% Frequency response
» plot(w/2/pi,20*log10(abs(H))); %%% Magnitude plot in dB
» xlabel('Frequency [normalized]')
» ylabel('Magnitude [dB]')
```

Hopefully, you will see that this is a lowpass filter with bandwidth approximately $1/M$ and stopband damping -13 dB (which does not improve with increasing M!).

2. Least-squares design. Write a Matlab function that implements the weighted least-squares digital FIR filter design. The input should be:
   - M, the filter order (corresponding to $M+1$ filter taps)
   - F, a vector of frequency samples
   - A, a vector of desired response values (of equal length as F)
   - W, a vector of weights to be used in the LS solution

The output of the function is the filter coefficients $\mathbf{b}_{LS}$. Hint: the following commands are useful for setting up the matrix $\mathbf{\Phi}$, avoiding for-loops that are slow in Matlab:

```
» MM = (M:-2:1)/2;
» F = [linspace(0,Fp,K/2) linspace(Fs,0.5,K/2)];
» Phi = 2*cos(2*pi*F'*MM);
```

3. Run your LS design for increasing filter orders M and different choices of passband and stopband weights W. Use the following set of commands for the first run:

```
» M = 11; %%% Filter order
» K = 100; %%% Number of frequency samples
» Fp = 0.25; Fs = 0.3; %%% Passband and stopband edge
» F = [linspace(0,Fp,K/2) linspace(Fs,0.5,K/2)]; %%% Frequency
  samples
» A = [ones(1,K/2) zeros(1,K/2)]; %%% Desired response
» Wpass = 1; Wstop = 1; %%% Passband and stopband weights
» W = [Wpass*ones(1,K/2) Wstop*ones(1,K/2)]; %%%
» bLS = LSdesign(M,F,A,W); %%% Name your function "LSdesign.m"
» [H,w] = freqz(bLS,1,1024); %%% Frequency response
» plot(w/2/pi,20*log10(abs(H))); %%% Check if the specifications
  are met
» xlabel('Frequency [normalized]')
```

```
» ylabel('Magnitude [dB]')
```

Spend some time to select the input to the filter so you get something that almost meets the specifications of the test problem. Compare your results with that of Matlab's implementation: `firls`. Note that the syntax is slightly different for `firls`. You should use the following command:

```
» bFIRLS = firls(M,2*[0 0.25 0.3 0.5],[1 1 0 0],[Wpass Wstop]);
```

4. Parks-McClellan design. Write a Matlab function that implements the Parks-McClellan digital FIR filter design for the low-pass case. The input should be:
   - M, the filter order (corresponding to $M+1$ filter taps).
   - F, a vector of frequencies used to separate the frequency bands, i.e. $F = [0, Fp, Fs, 0.5]$ (assume a lowpass filter!).
   - $A = [1, 1, 0, 0]$, a vector used to define that there are two bands, where the first is a passband and the second a stopband (only this case needs to be implemented).
   - $W = [W_{pass}, W_{stop}]$, specifies the weights for the passband and stopband respectively.

   The output is the filter coefficients $\mathbf{b}_{PM}$, obtained by solving the minimax optimization problem:

$$\mathbf{b}_{PM} = \arg\min_{\mathbf{b}} \max_{\Omega \in \mathcal{O}} |E(\Omega, \mathbf{b})|$$

   where $E(\Omega, \mathbf{b})$ is the same weighted error as used in the LS-design:

$$E(\Omega, \mathbf{b}) = W(\Omega)\left(A_d(\Omega) - A(\Omega, \mathbf{b})\right) .$$

   The simplest way to solve the optimization problem is to use Matlab's `fminimax` routine. See `help`

`fminimax` for details. If you are really interested, you could also try to implement your own code based on the iterative procedure described above. Contact `viberg@chalmers.se` in that case. Regardless which implementation you have used, you should verify that your resulting filter is indeed equi-ripple!

5. Spend some time to select the input to the filter (M and the weights) so that the resulting amplitude response meets the specifications of the test problem. Compare your results with that of Matlab's implementation `firpm`. Use:

```
» bFIRPM = firpm(M,2*[0 0.25 0.3 0.5],[1 1 0 0],[Wpass Wstop]);
```

(there is also a command `firpmord` that estimates the necessary filter order to satisfy the given specifications).