MODELLING AND SIMULATIONS

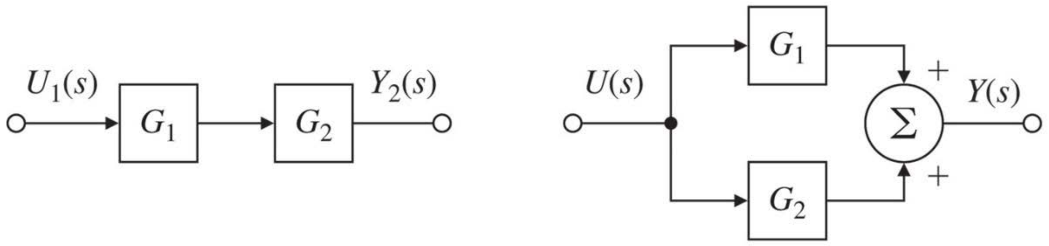
TRANSFER FUNCTIONS SUMUP

AALBORG UNIVERSITY
DENMARK

Transfer Function and Block Diagrams

The function $H(s)$ is the transfer gain from $U(s)\ to\ Y(s)$ or the (input to output), thus:

$$\frac{Y(s)}{U(s)} = H(s)$$



$$\frac{Y_2(s)}{U_1(s)} = G_2 G_1$$

(a)

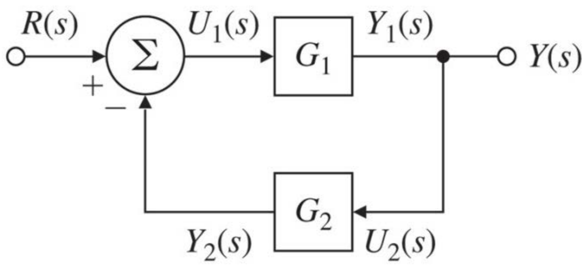
$$\frac{Y(s)}{U(s)} = G_2 + G_1$$

(b)

Transfer Function and Block Diagrams

The closed loop transfer function :

$$U_1(s) = R(s) - Y_2(s)$$
$$Y_2(s) = G_2(s)G_1(s)U_1(s)$$
$$Y_1(s) = G_1(s)U_1(s)$$



$$\frac{Y(s)}{R(s)} = \frac{G_1}{1 + G_2G_1}$$

(c)

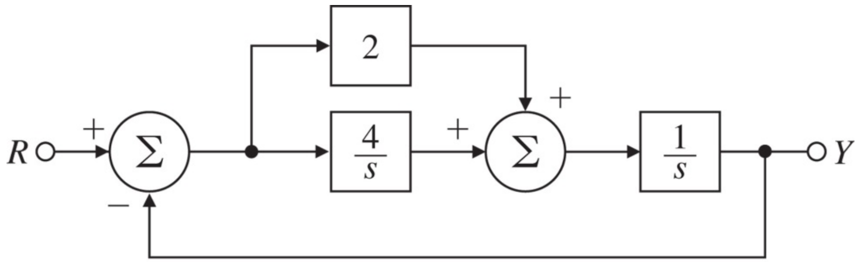
Where

$$Y_1(s) = \frac{G_1(s)}{1 + G_1(s)G_2(s)} R(s)$$

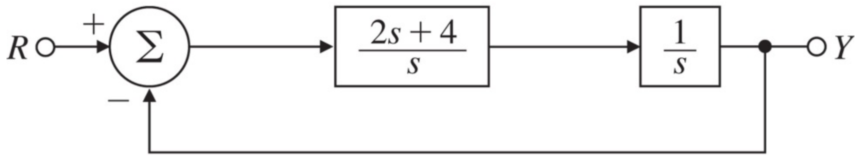
Which is reffered to as the negative feedback, where positive feedback is:

$$Y_1(s) = \frac{G_1(s)}{1 - G_1(s)G_2(s)} R(s)$$

Example



(a)



(b)

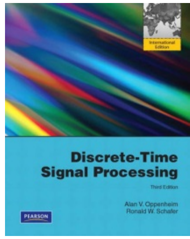
MODELLING AND SIMULATIONS

FILTERING

AALBORG UNIVERSITY
DENMARK

Intro

- Useful book
  - **Discrete-time Signal Processing: International Version** Paperback – International Edition, 2010

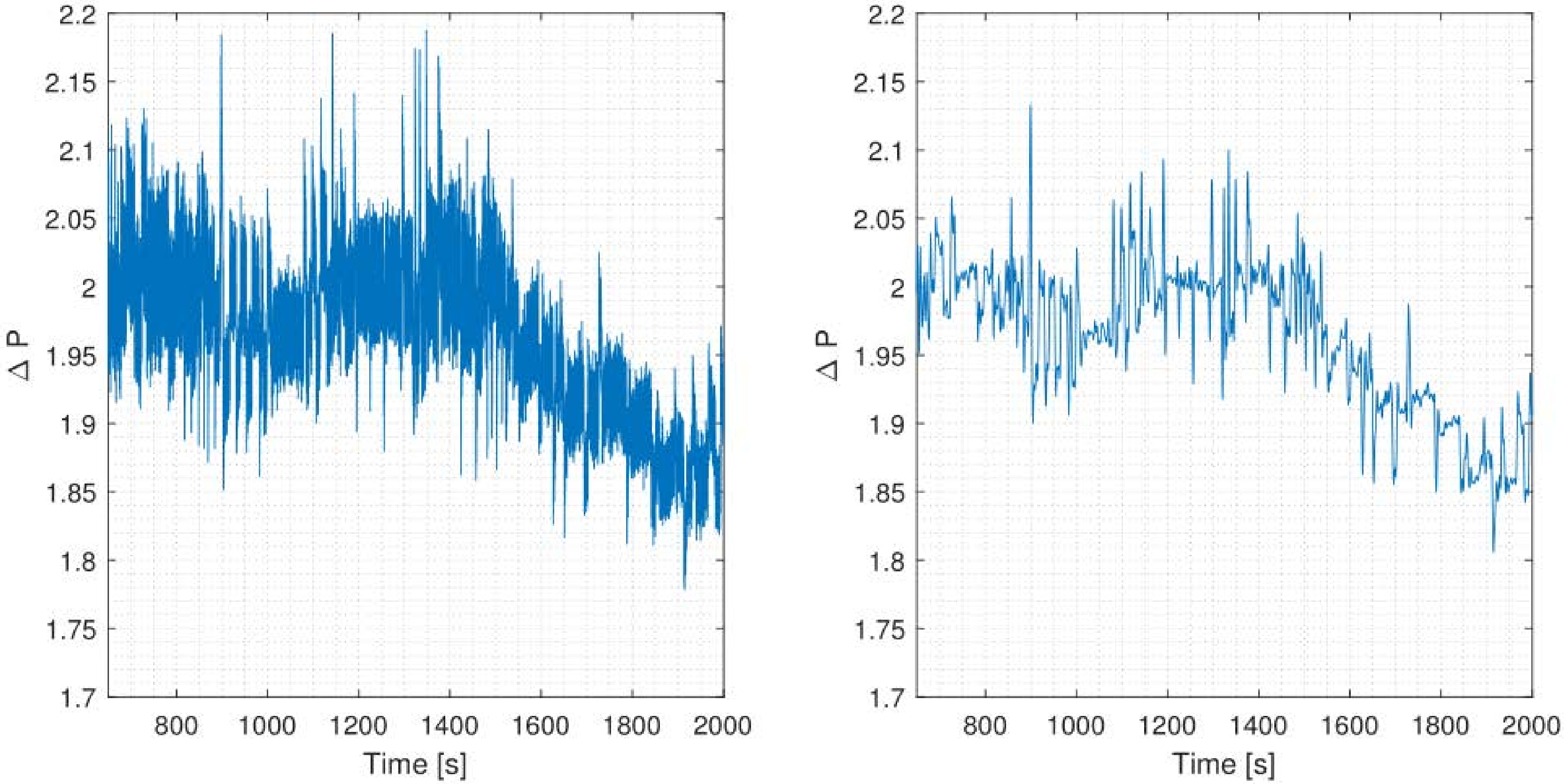


Topics of the lecture: Noise and Filtering

- Noise in the data can obstruct the information
- Some noise can be removed by filtering
  - A filter attenuates specific frequencies
- In order to do so:
  - The Noise frequency needs to be obtained
  - Through frequency analysis of the signal
  - Fourier analysis
  - Bode plot
- Filter design
  - Different types of filters
    - Pass bands
    - FIR/IIR
    - Butterworth…
- Practical session

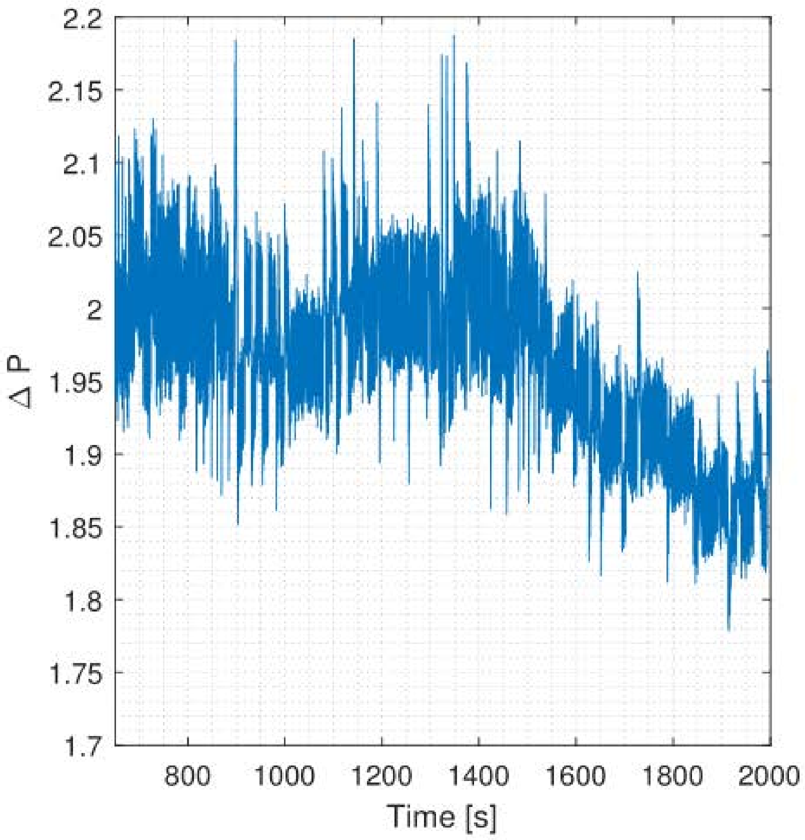
Filtering and Noise

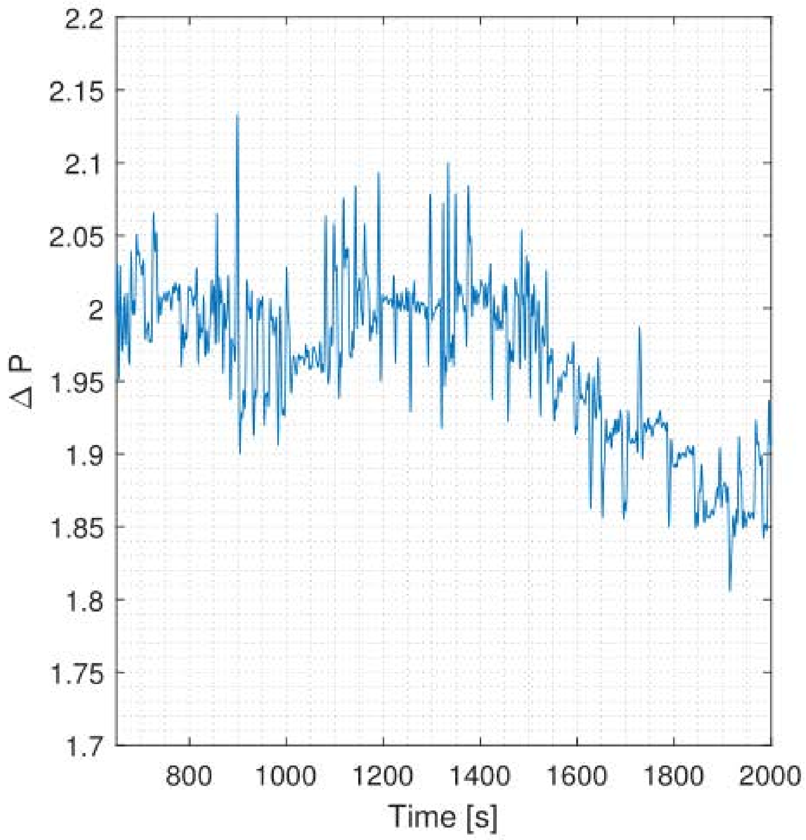
- Noise can obstruct the relevant data
- Noise can be caused by various things:
    - Poor performing sensors, vibrations, disturbances, signal noise…



Filtering and Noise

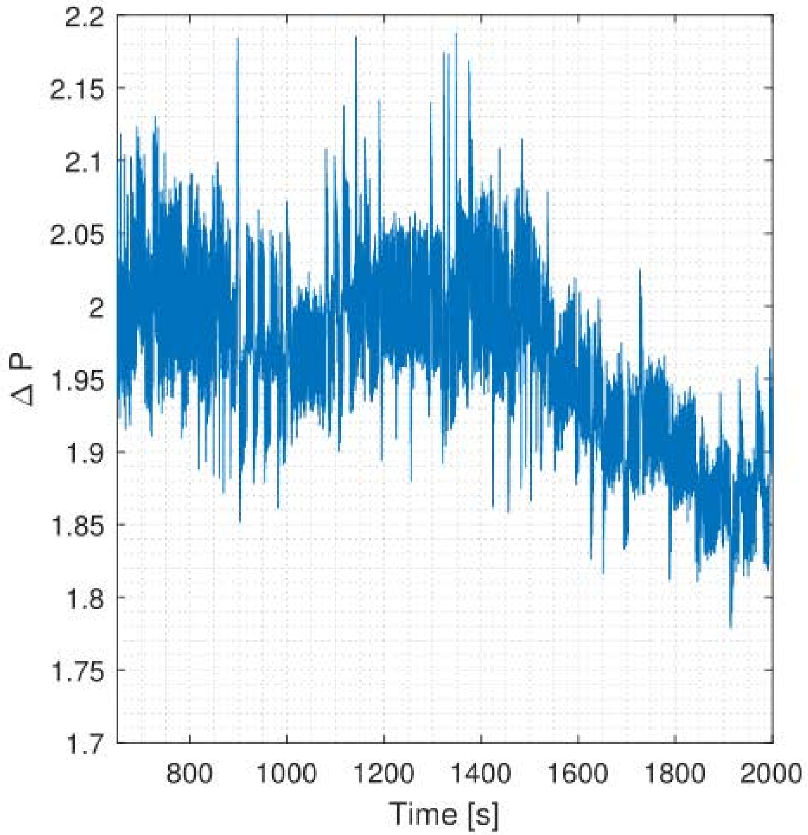
- The following signal is a delta pressure measurement sampled at 100 Hz
- Noise comes from the pressure transmitters sensitivity and the ΔP calculation

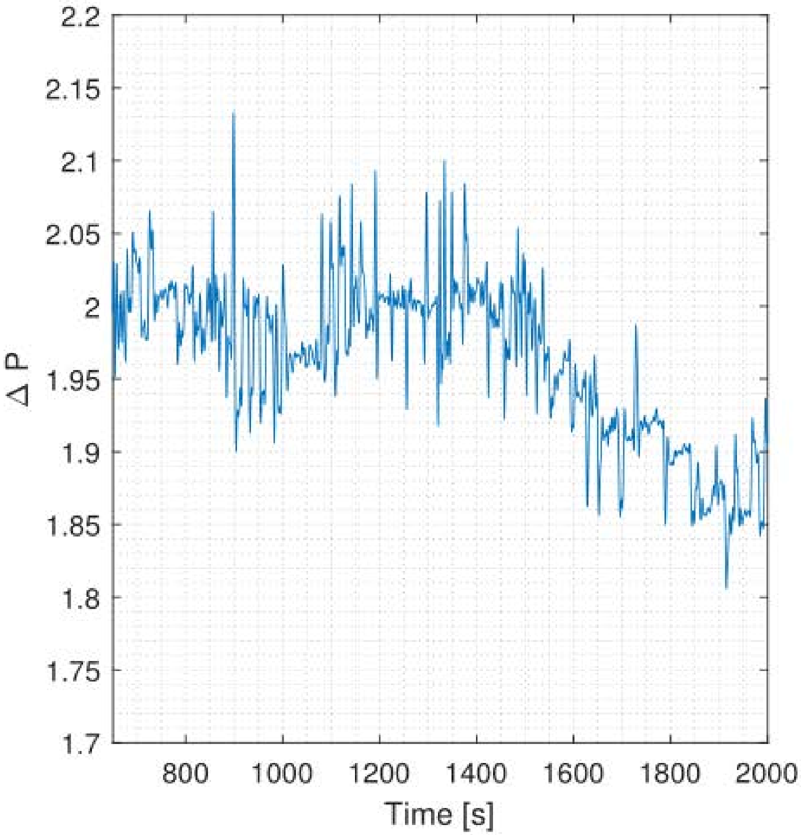




Filtering and Noise

- The signal is filtered using a low pass filter
- Signals application after filtering
  - Model identification
  - Feedback control
  - Plotting





Types of Filters

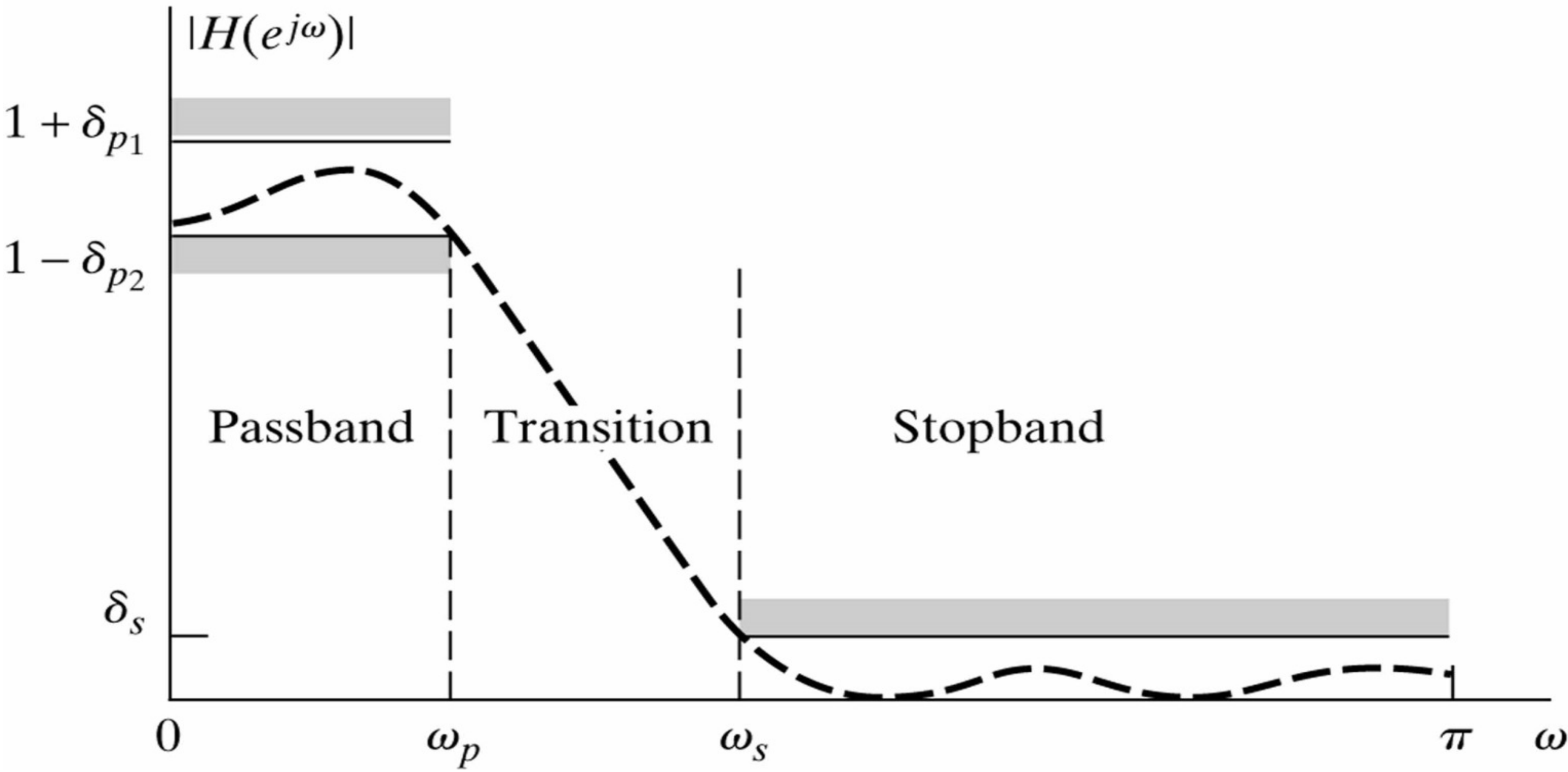
AALBORG UNIVERSITY
DENMARK

Low-pass filter tolerance scheme

Passband = Unity gain

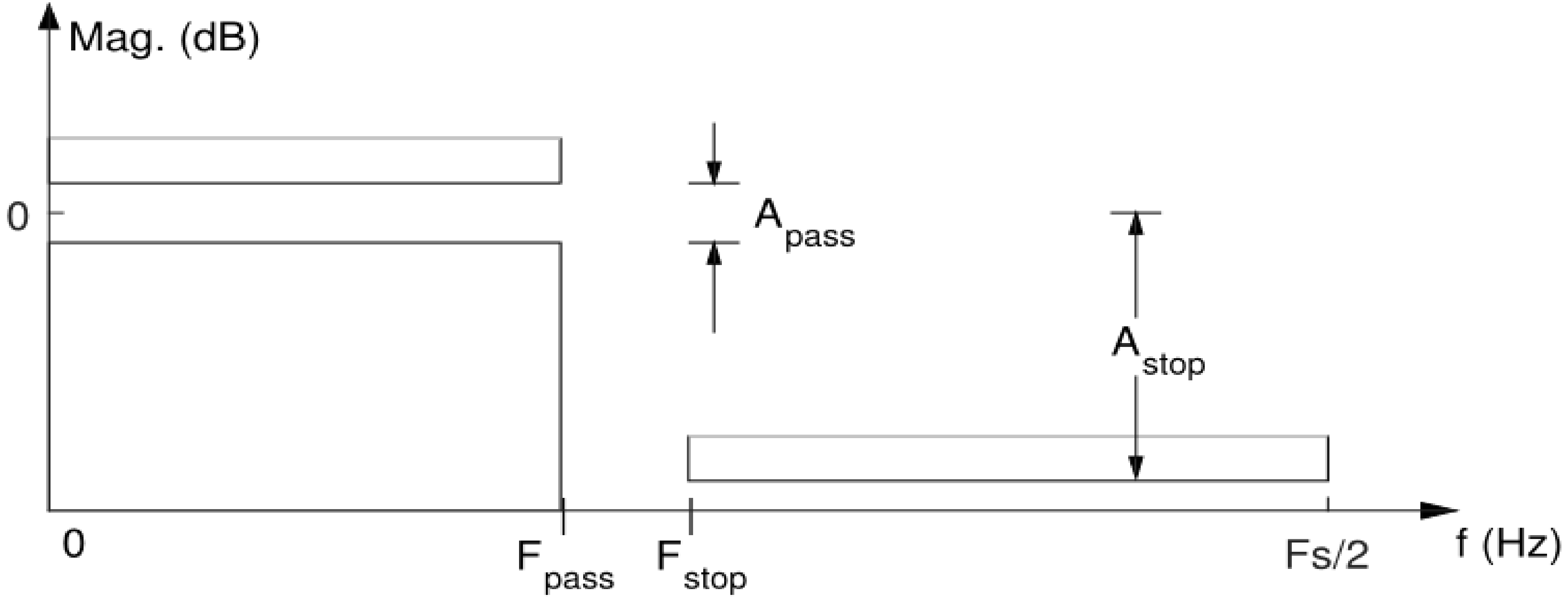
Stopband = 0 gain

Transition is unconstrained



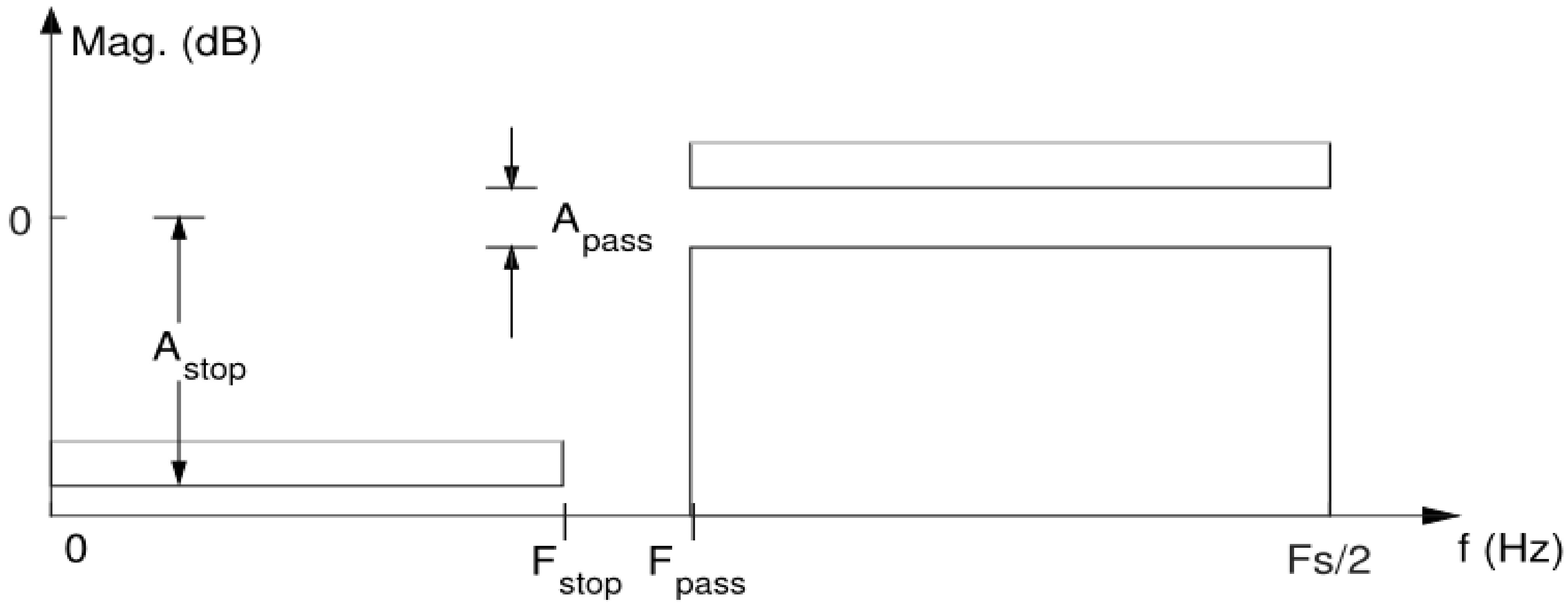
Filter types

- Low-pass filter
  - Used to filter away high frequency response which is not relevant for model design or parameter estimation
  - Common type of filter in control design, to remove noise



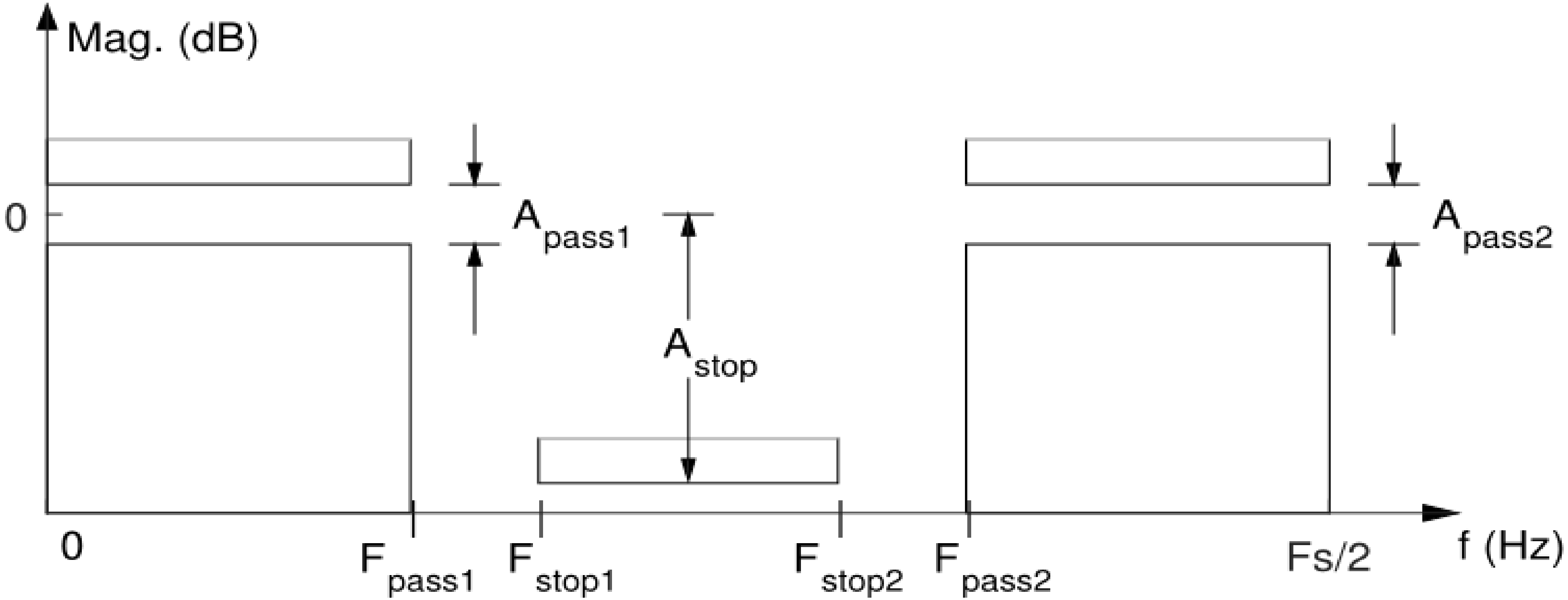
Filter types

- High-pass filter
  - Could be used in audio systems, to filter high frequencies from woofers



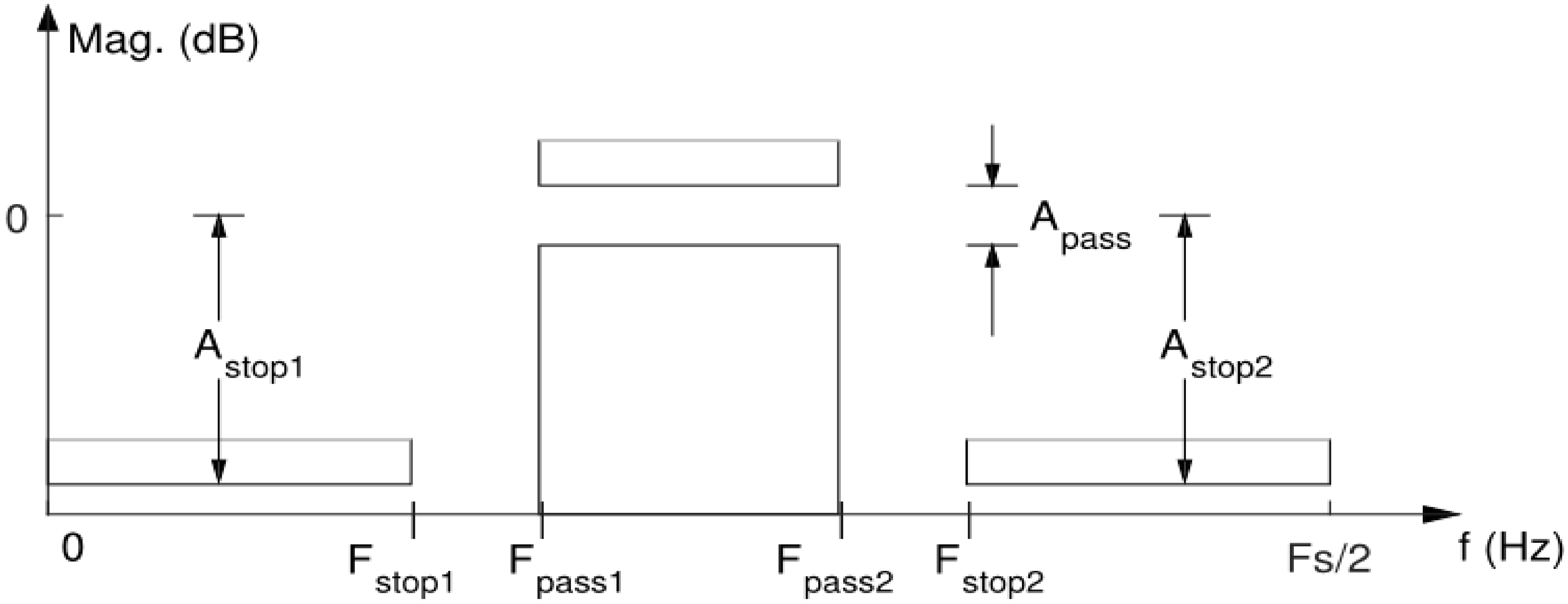
Filter types

- Band-stop filter
  - Could be used to reject 50Hz or 60Hz signal from the power grid



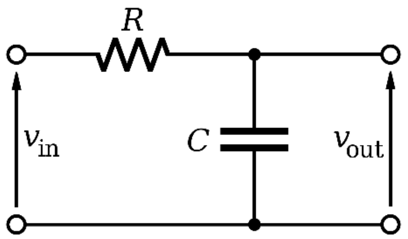
Filter types

- Band-pass filter
  - Used for datalink, where a frequency is used as a carry signal

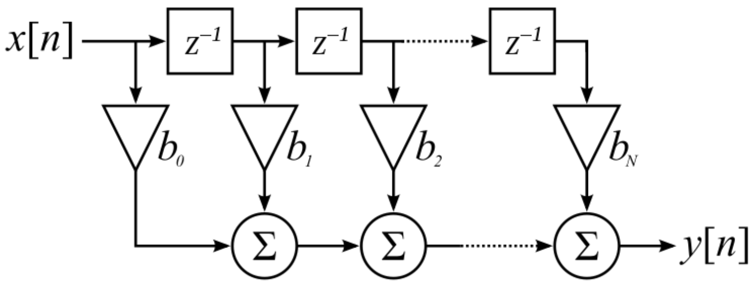


Analog and Digital Filters

- Analog
  - Lowpass
  - $\omega_c = \frac{1}{R \cdot c}$



- Digital/software
  - $y[n] = x[n] \cdot b_0 + x[n-1] \cdot b_1 + x[n-2] \cdot b_2 \ldots + x[n-N] \cdot b_N$
  - $y[n] = y[n-1] \ldots$
  - $y[n] = y[n+2] \ldots$ can not be used "online"



FIR and IIR filters

- A **finite impulse response (FIR)** filter is a filter whose impulse response (or response to any finite length input) is of *finite* duration, because it settles to zero in finite time.

- A **infinite impulse response (IIR)** filters, which may have internal feedback and may continue to respond indefinitely

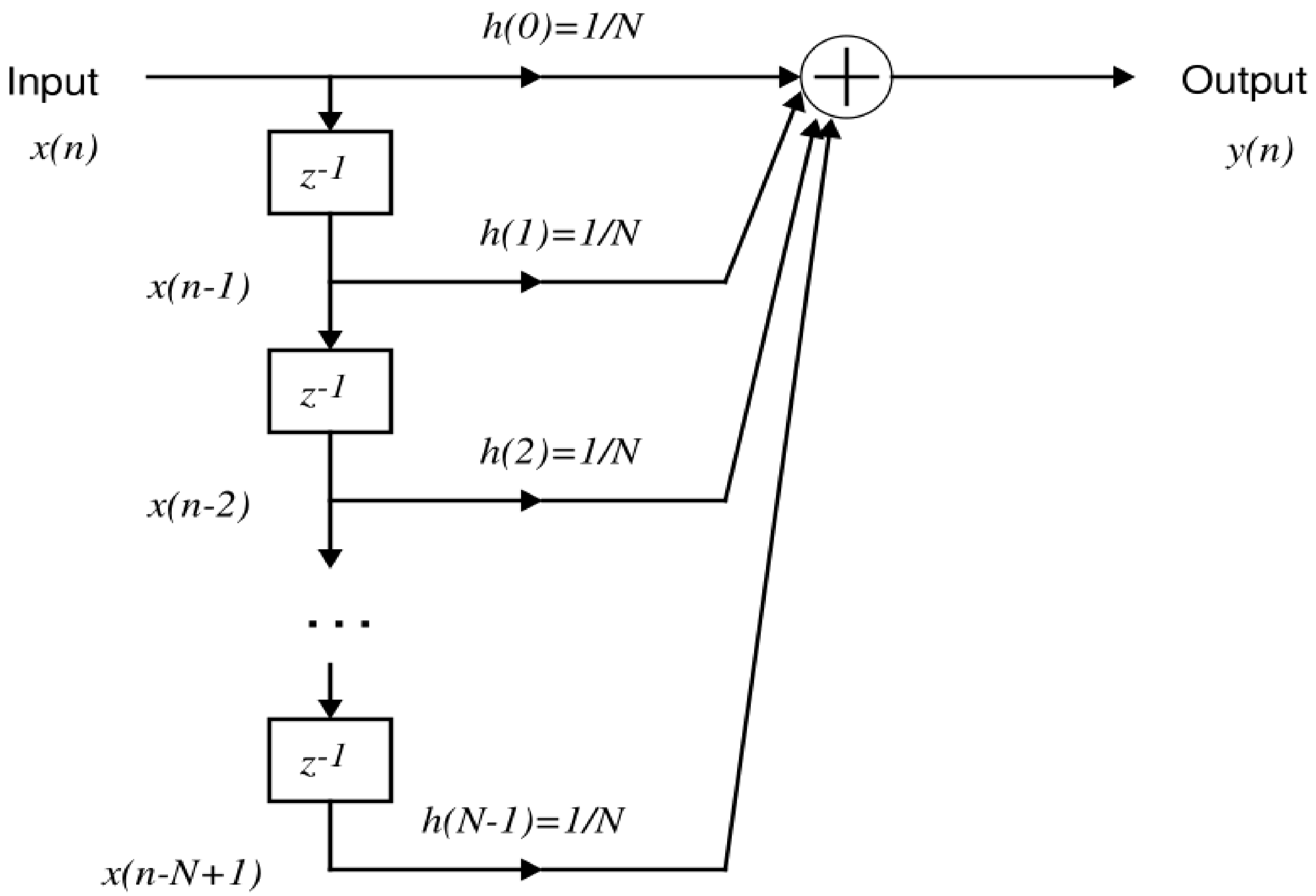
- Advantage/disadvantage – The internal feedback in the IIR filter, reduces the order of the filter compared to a FIR filter, with the same design specs. The FIR filter have a constant group delay, rather the IIR filter have a filter delay dependent on the frequency of the signal.

FIR filters

AALBORG UNIVERSITY
DENMARK

FIR filters

- FIR example



AALBORG UNIVERSITY
DENMARK

FIR filter Design

- Frequency response of a Nth-order casual FIR filter

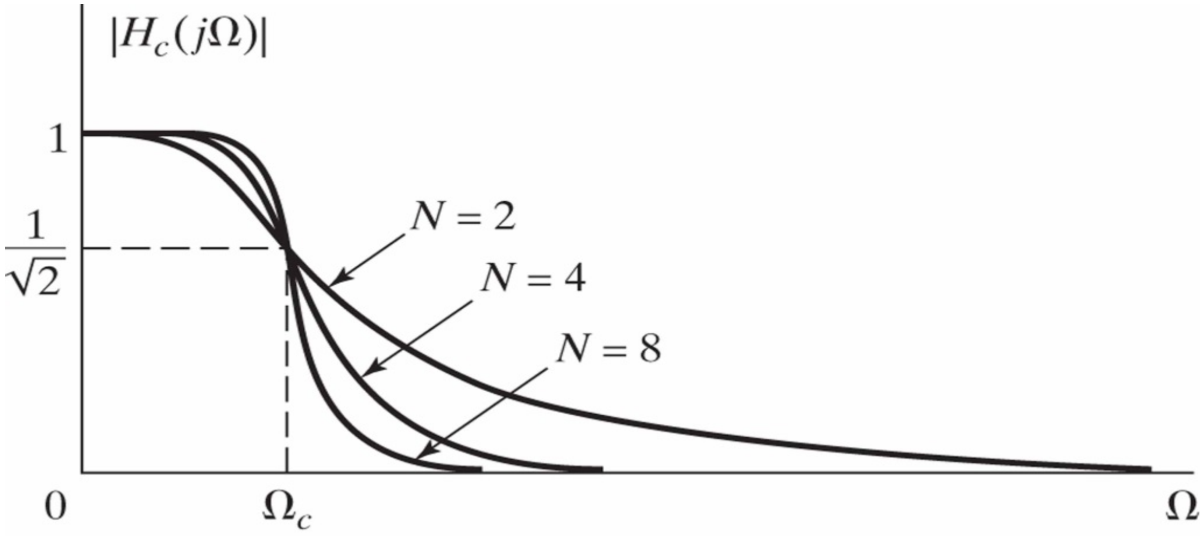
$$H(e^{j\omega}) = \sum_{n=0}^{N} h[n]e^{-jn\omega}$$

IIR filters

AALBORG UNIVERSITY
DENMARK

Butterworth filter

$$|H_c(j\omega)|^2 = \frac{1}{1+\left(\frac{j\Omega}{j\Omega_c}\right)^{2N}}$$



- The higher the frequency Ω the lower the gain and vice versa
  - Inherently it is a low pass filter
- The higher the N the sharper the cutoff becomes *(see figure)*
- For $N \Rightarrow \infty$ the gain becomes a rectangular function, i.e. **ideal filter**

AALBORG UNIVERSITY
DENMARK

Chebyshev Filters – Type 1

It has a **pass-band equiripple** and varies **monotonically** in the **stop-band** *(type I)*

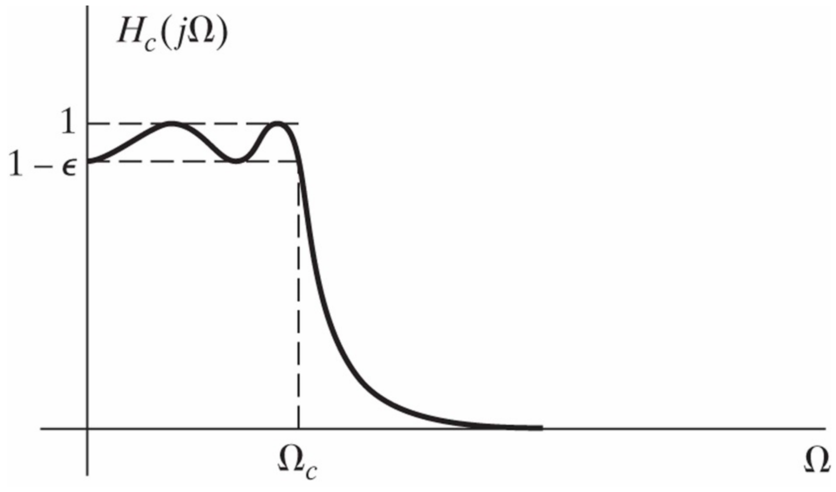
Stop-band equiripple and **monotonic** in the **pass-band** *(type II)*

Magnitude-square function of **Type 1**:

$$|H_c(j\Omega)|^2 = \frac{1}{1 + \varepsilon^2 V_N^2\left(\frac{\Omega}{\Omega_c}\right)}$$

Based on the Chebyshev polynomial $V_N(x)$,

$$V_N(x) = \cos(N \cos^{-1} x)$$



AALBORG UNIVERSITY
DENMARK

Elliptic filters

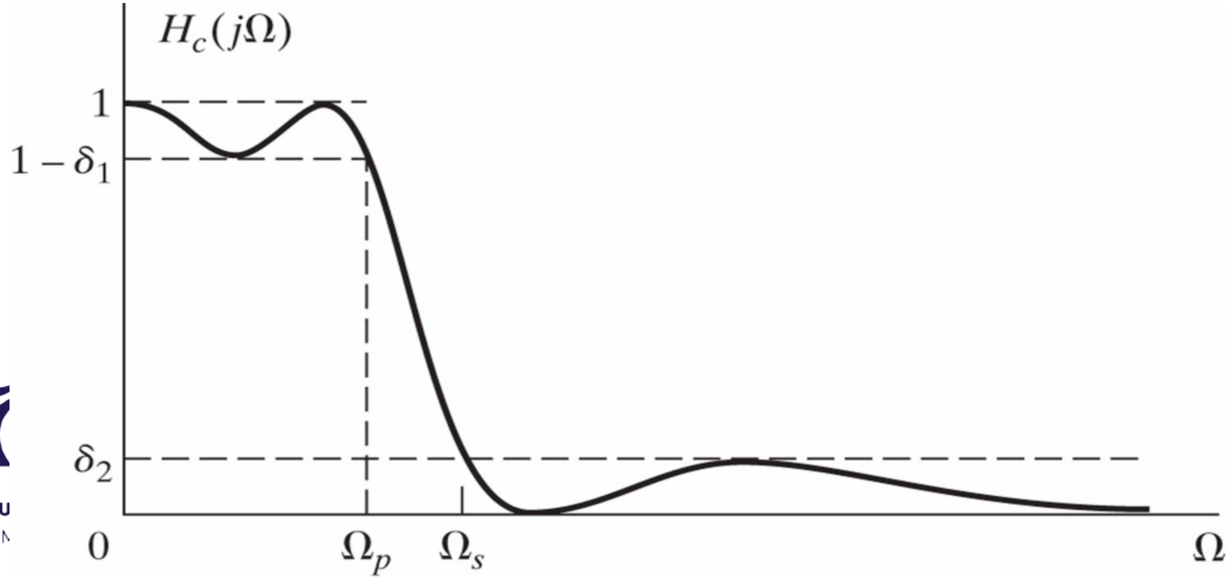
- Equiripple in the passband and the stopband
  - Preserved in DT using bilinear transformation
- The best that can be achieved for a given filter order N

i.e. for given values Ω_p, δ_1 and δ_2 the transition band $\left(\Omega_s - \Omega_p\right)$ is as small as possible

The Magnitude square function of a Elliptic filter is:

$$|H_c(j\Omega)|^2 = \frac{1}{1 + \varepsilon^2 U_N^2(\Omega)}$$

Where $U_N(\Omega)$ is the Jacobian elliptic function



FIR vs. IIR short guide

	FIR	IIR
Computation vs. performance	- Often more computation for the same magnitude response as IIR	+ Less computation for a given magnitude response
Phase	+ Can have exactly linear phase + Other phase responses possible	- Nonlinear phase leads to phase distortion of signal (distortion of "waveshape")
Stability	+ Guaranteed stability	- Must verify stability of final design; no guarantee of stability
Effect of limited number of bits for coefficients and math	+ Noise and errors are generally lower than for IIR	- More sensitive to quantization of coefficients and noise from rounding off calculations
Use of analog filter as model	- No direct conversion from an analog design to FIR (indirect methods exist)	+ Several methods for converting analog designs (this is the most common "pencil and paper" design method)
Arbitrary filter specifications	+ Possible even when no analog equivalent is possible	- (Much) more difficult to produce arbitrary designs
Implementation	+ Straightforward implementation; most DSP hardware supports directly and efficiently	- Large filters tend to be broken down into smaller stages; a bit more complicated than FIR - For optimum performance, careful design of filter stages is necessary
Design	- Optimum designs require computer programs (no closed-form solutions) + Some design methods are fairly simple	+ Well-known design processes, can be done manually

Aliasing

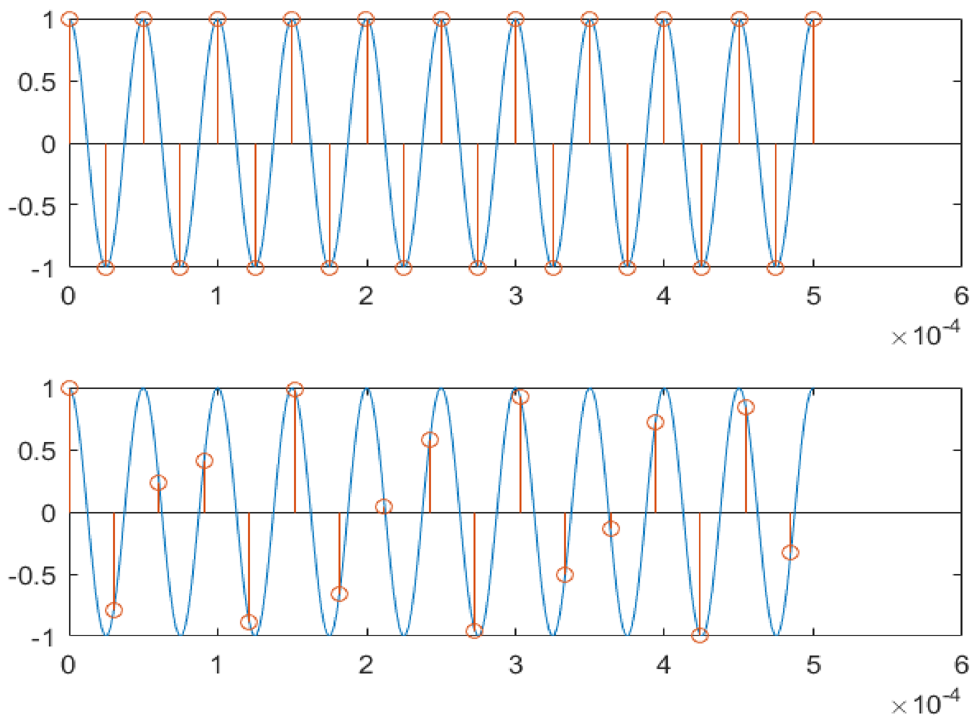
Signals above $\frac{F_S}{2}$ are represented in DT as being in the range $0\ to\ \frac{F_S}{2}$

AALBORG UNIVERSITY
DENMARK

Aliasing

Example: 20 kHz sine signal

Sampled at 40 kHz and 33 kHz respectively



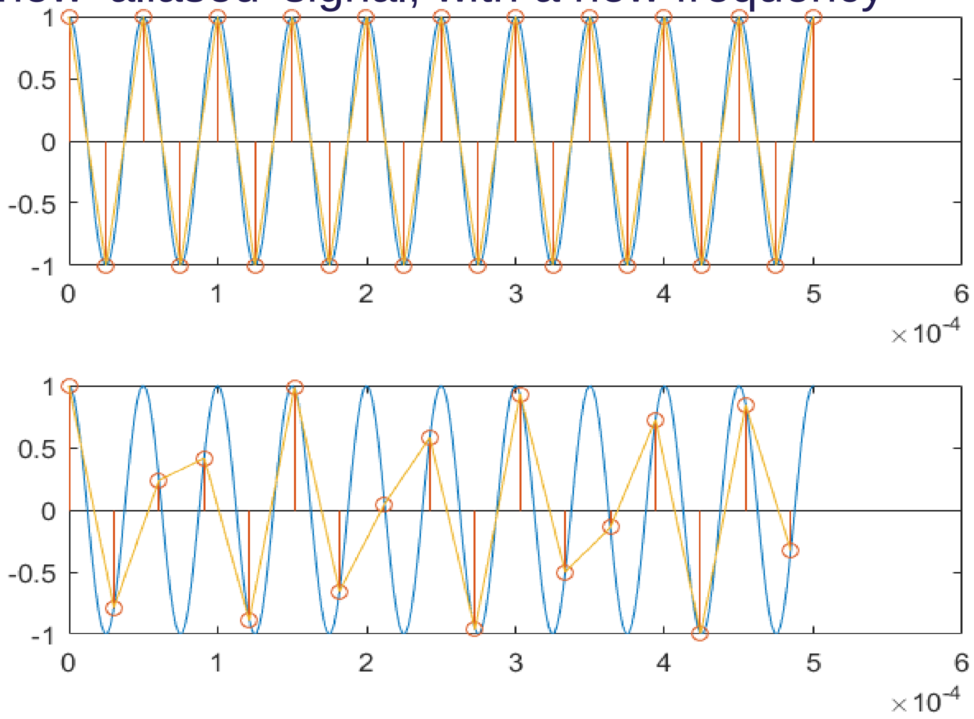
Aliasing

Sampling_of_sine.m

Example: 20 kHz sine signal

Sampled at 40 kHz and 33 kHz

At 33kHz aliasing produces a new ‘aliased’ signal, with a new frequency



Nyquist Frequency

$\frac{F_s}{2}$, half of the sampling rate

Nyquist rate = the frequency above which we avoid aliasing

Thus we wish sampling frequency F_s, to be at least ***two times the signal frequency to avoid aliasing***

As was shown in the previous Matlab example.

Nyquist Frequency

$\frac{F_s}{2}$, half of the sampling rate, ‘Nyquist frequency or Folding rate’
Nyquist rate = the frequency above which we avoid aliasing

Example:
Calculate the aliased frequency F_a

$$F_a = F - F_s \left\lfloor \frac{F + \left(\frac{F_s}{2}\right)}{F_s} \right\rfloor$$

Wrt. the example:
F = 20kHz, Fs = 33kHz

$$-13kHz = 20 - 33 \left\lfloor \frac{20 + \left(\frac{33}{2}\right)}{33} \right\rfloor$$

Choice of Filter

- **Cut off frequency**
  - System Bandwidth
- **Order of filter**
  - More noise reduction and smaller transition range both results in higher order
  - Online/offline? - implementation in Software/Matlab/Simulink
    - If online, the processing capabilities must be able to process the data in real time.
    - If offline, there are less restriction to the processer.
      - Note - If the filter is implemented as an analog filter, there are no such restrictions

FOURIER TRANSFORMS

Representation of sequences using Fourier transform

- Fourier transform is a generalization of the frequency response
  - For describing both signals and systems
- Mapping of signals into another ‘domain’
- Convolution operation is mapped to multiplication

Computation of the DFT

The DFT is **comparing the sequence $\boldsymbol{x(n)}$** and to the series of **sampled sinusoidal waves** to compare for **similarity**.

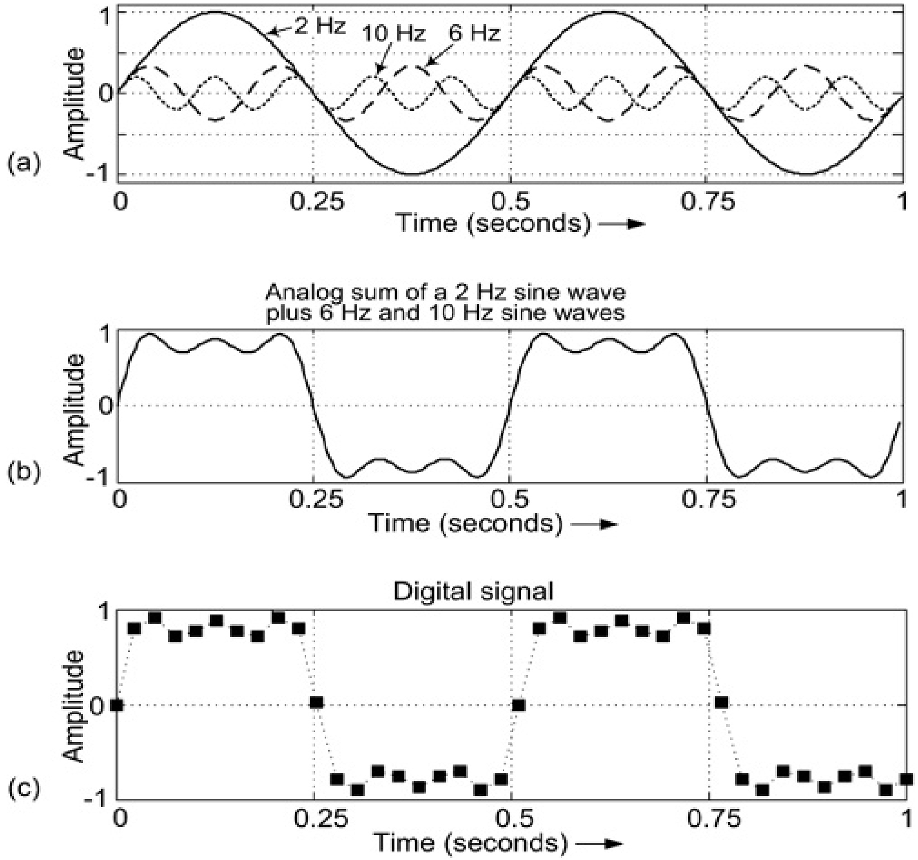
The DFT equation can be computed using the following equation:

$$X(k) = \sum_{n=0}^{N-1} x[n](W_N)^{kn}, \qquad k = 0, 1, \dots, N-1$$

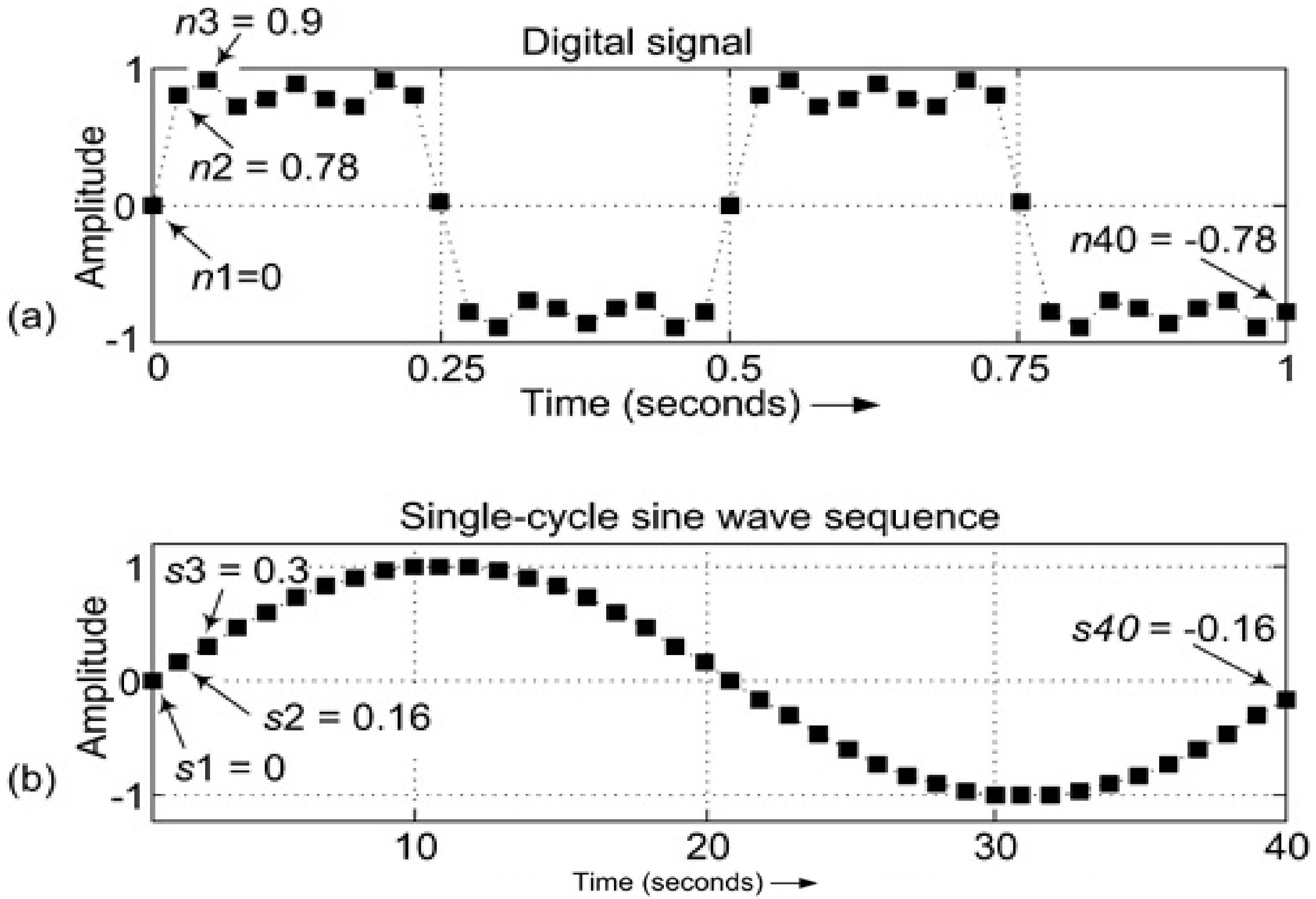
Where the **complex sinusoidal** $[n](W_N)^{kn}$ has the following form:

$$(W_N)^{kn} = \left(\cos\left(\frac{2\pi \mathrm{kn}}{N}\right) - j \cdot sin\left(\frac{2\pi kn}{N}\right)\right), \qquad k = 0, 1, \dots, N-1$$

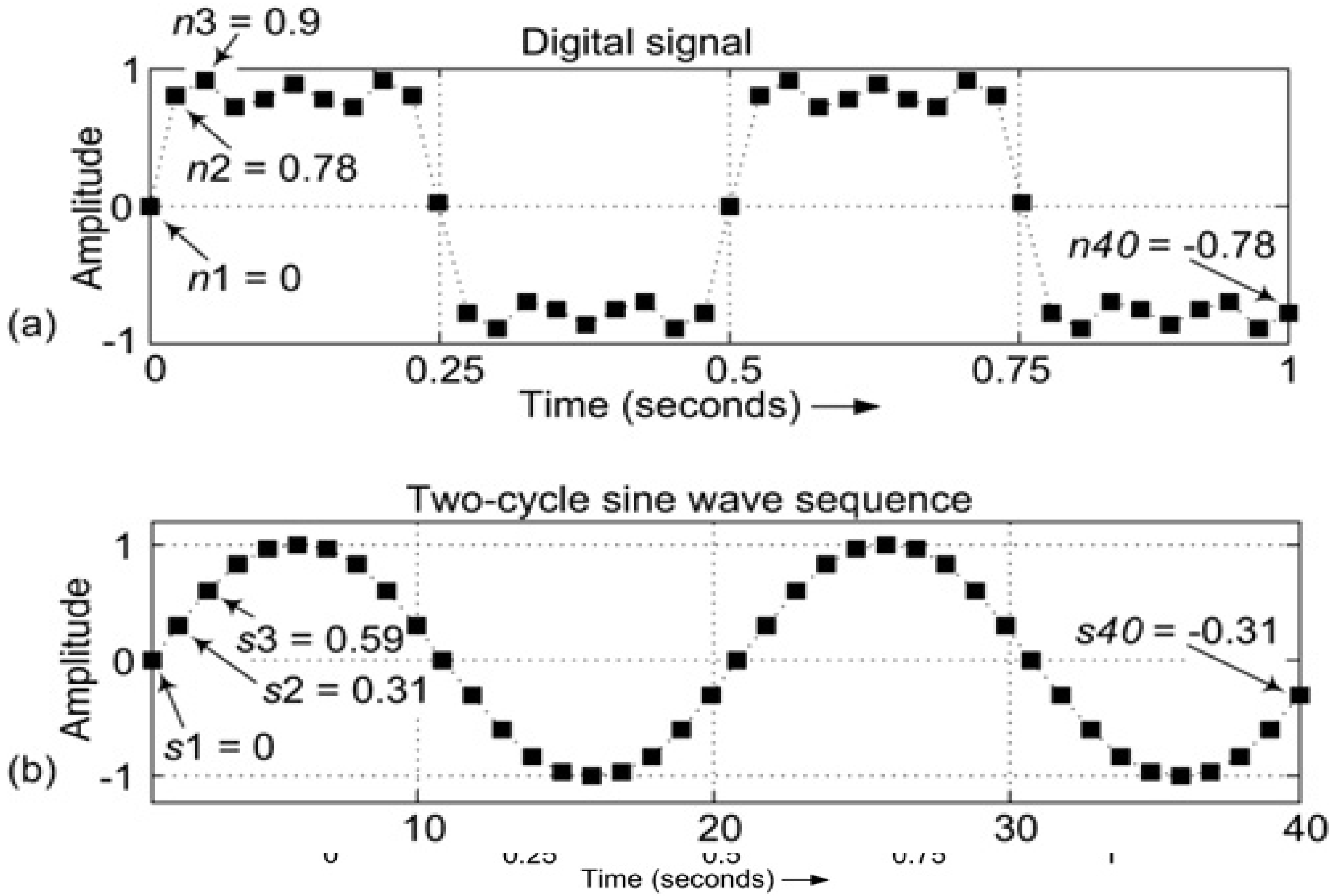
DFT Example – Part 1



DFT Example – Part 2

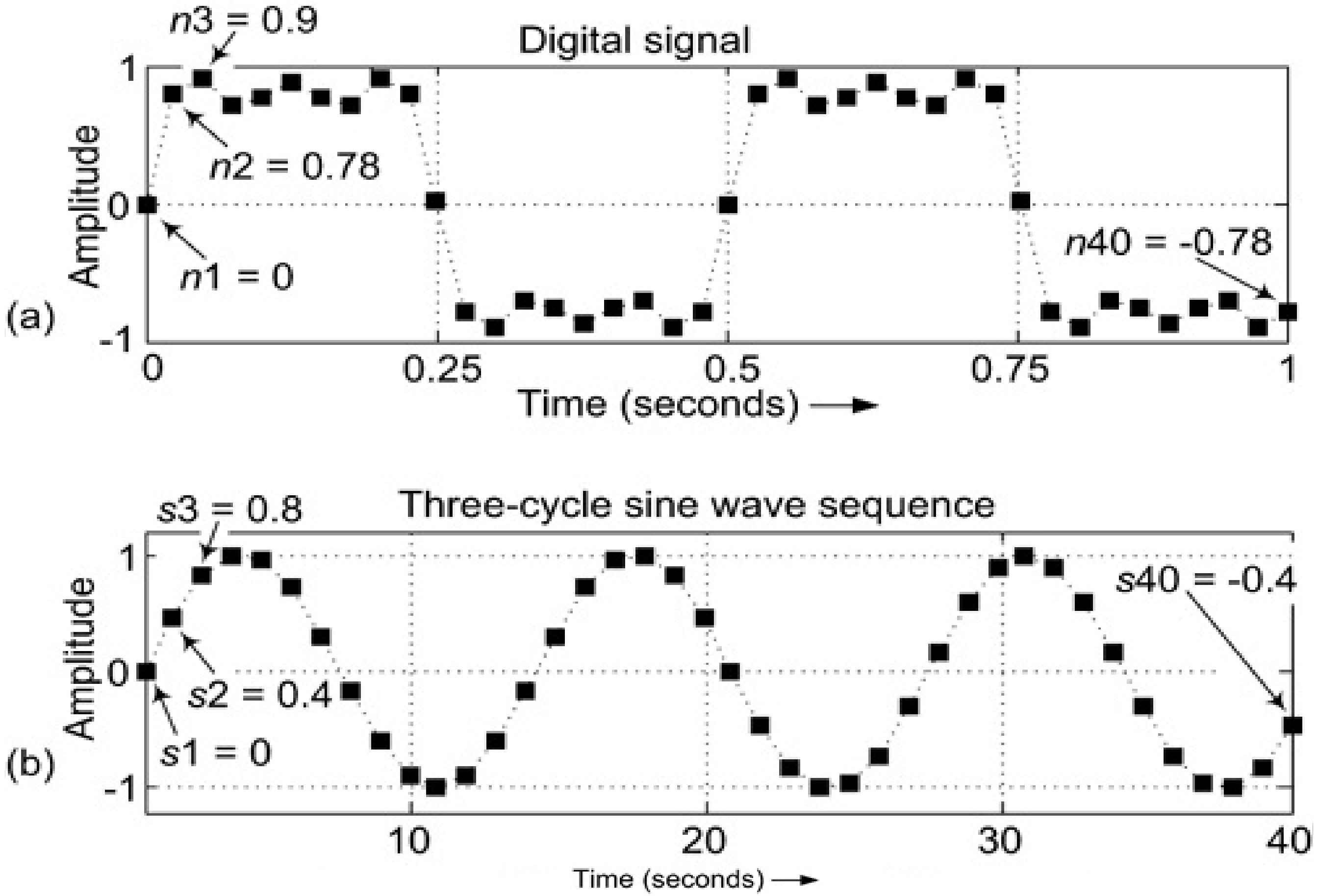


DFT Example – Part 2



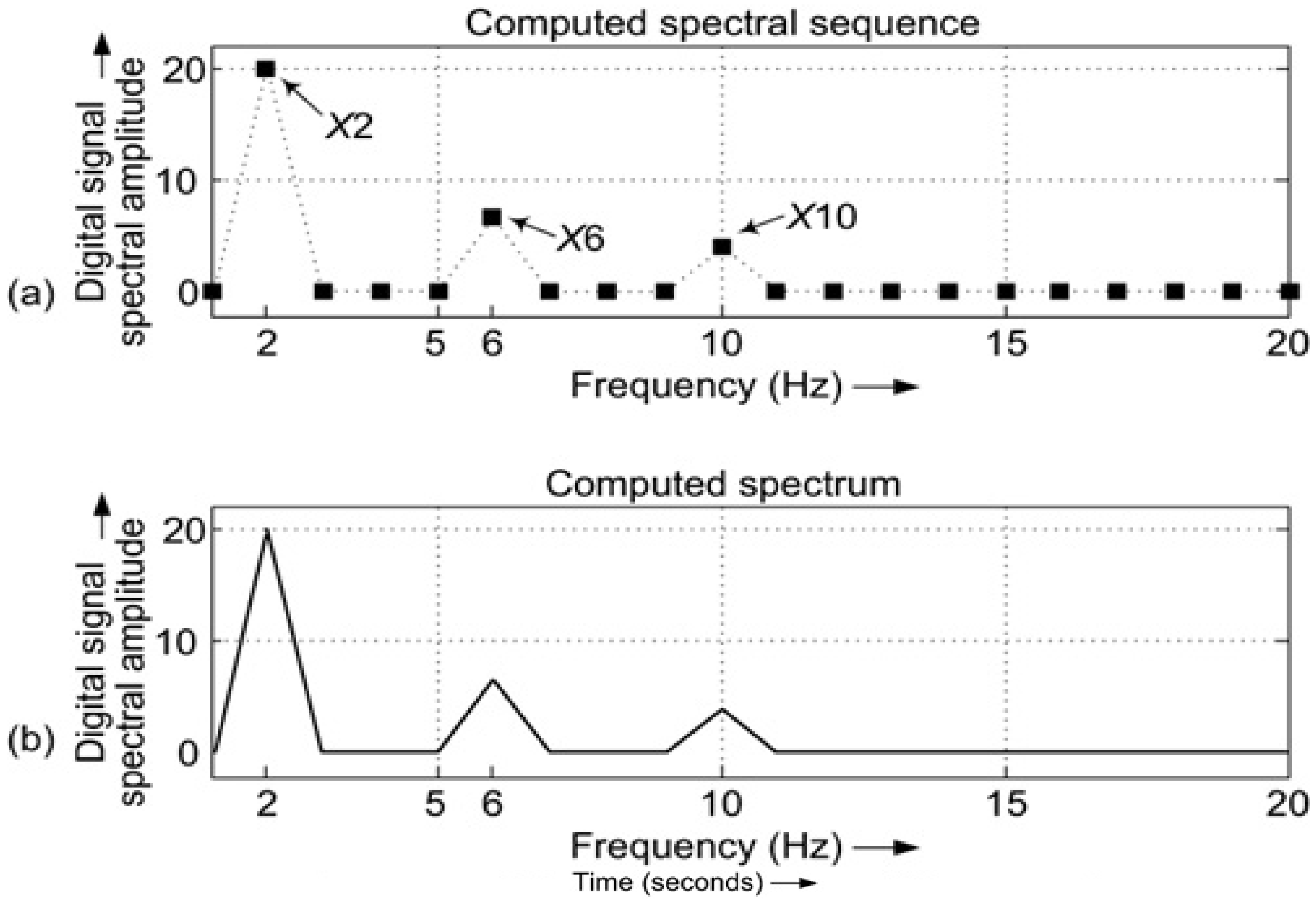
AALBORG UNIVERSITY
DENMARK

DFT Example – Part 2



AALBORG UNIVERSITY
DENMARK

DFT Example – Part 2



AALBORG UNIVERSITY
DENMARK

DFT - FFT

DFT is computationally heavy

Example:

1s interval of a sample, with $f_s = 8000Hz$ takes 128million addition operations and 256 million multiplication operations to compute the spectral $X1, X2, X3,$ ...

FFT, 1960s

Same example takes ≈ 100000 addition operations and 200000 multiplication operations, ca. $\frac{1}{1000}$ reduction

SIGNAL ANALYSIS USING SPTOOL

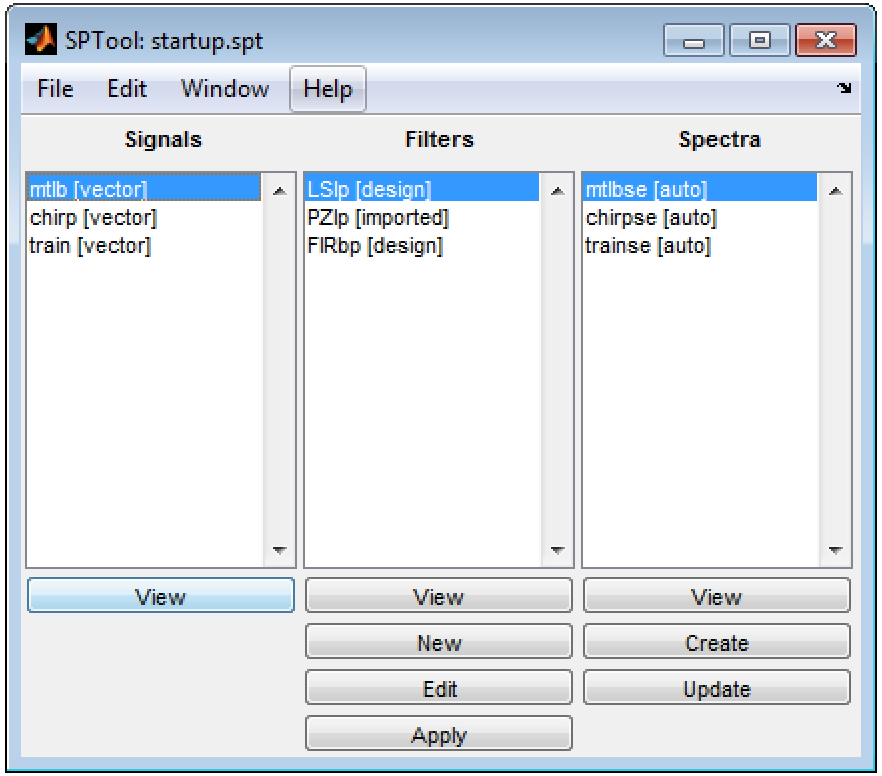
AALBORG UNIVERSITY
DENMARK

Practical session: Open interactive digital signal processing tool

- Tool for signal processing in Matlab

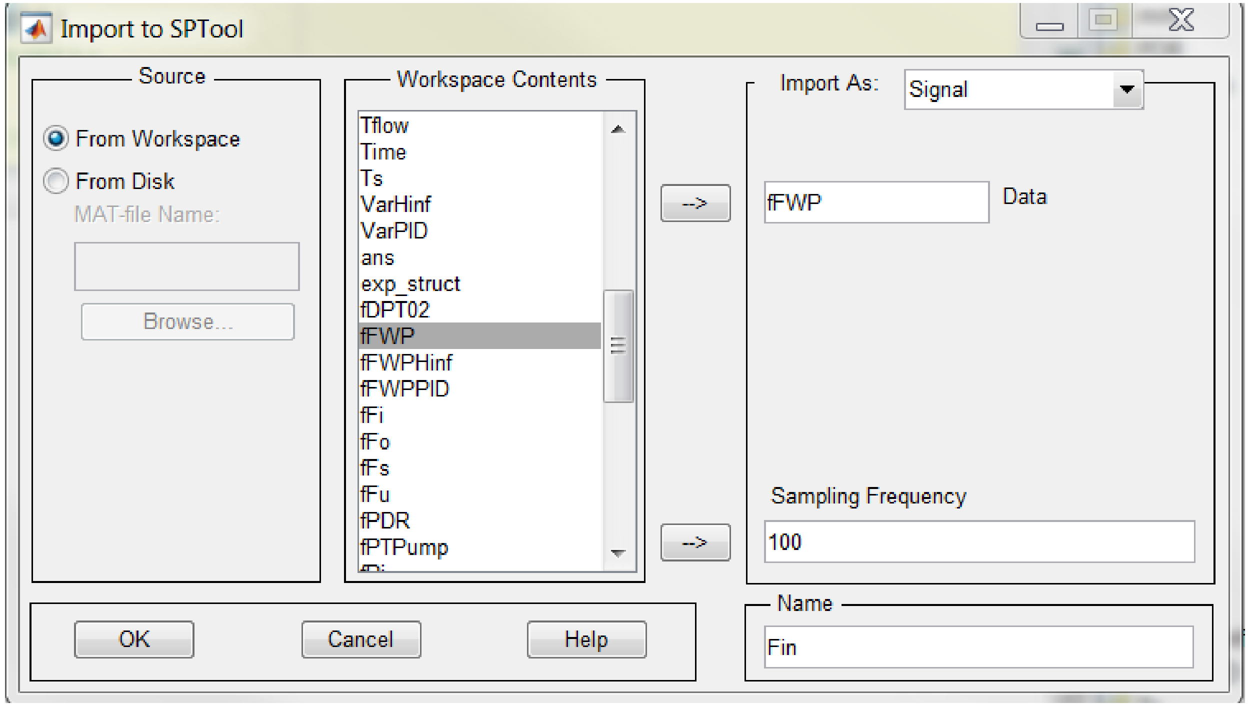
`sptool`

- SPTool, is a suite with four tools:
  - Signal Browser
  - Filter Design and Analysis Tool
  - FVTool
  - Spectrum Viewer.



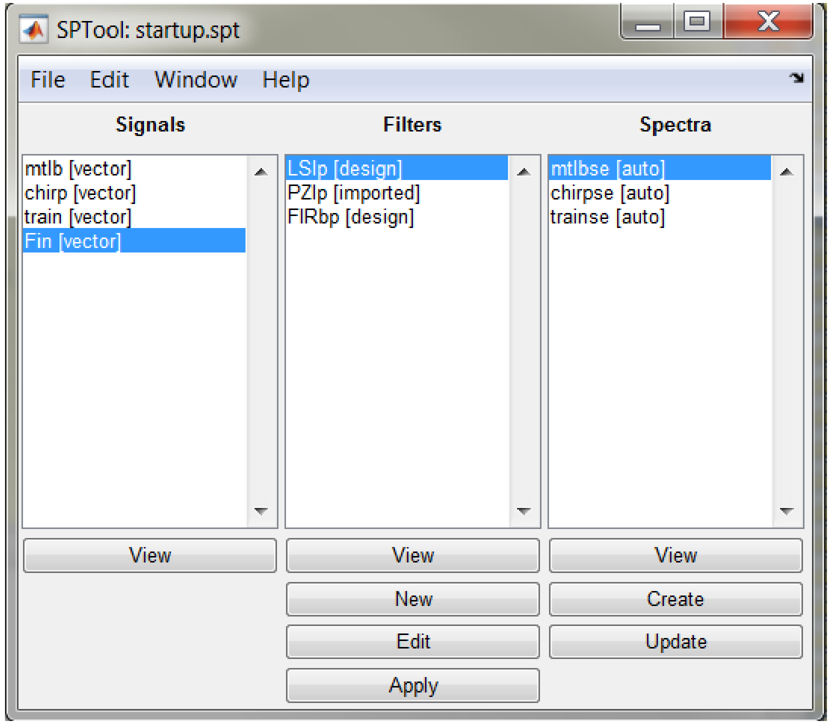
Practical session: introduction to SPTOOL

- Import signals



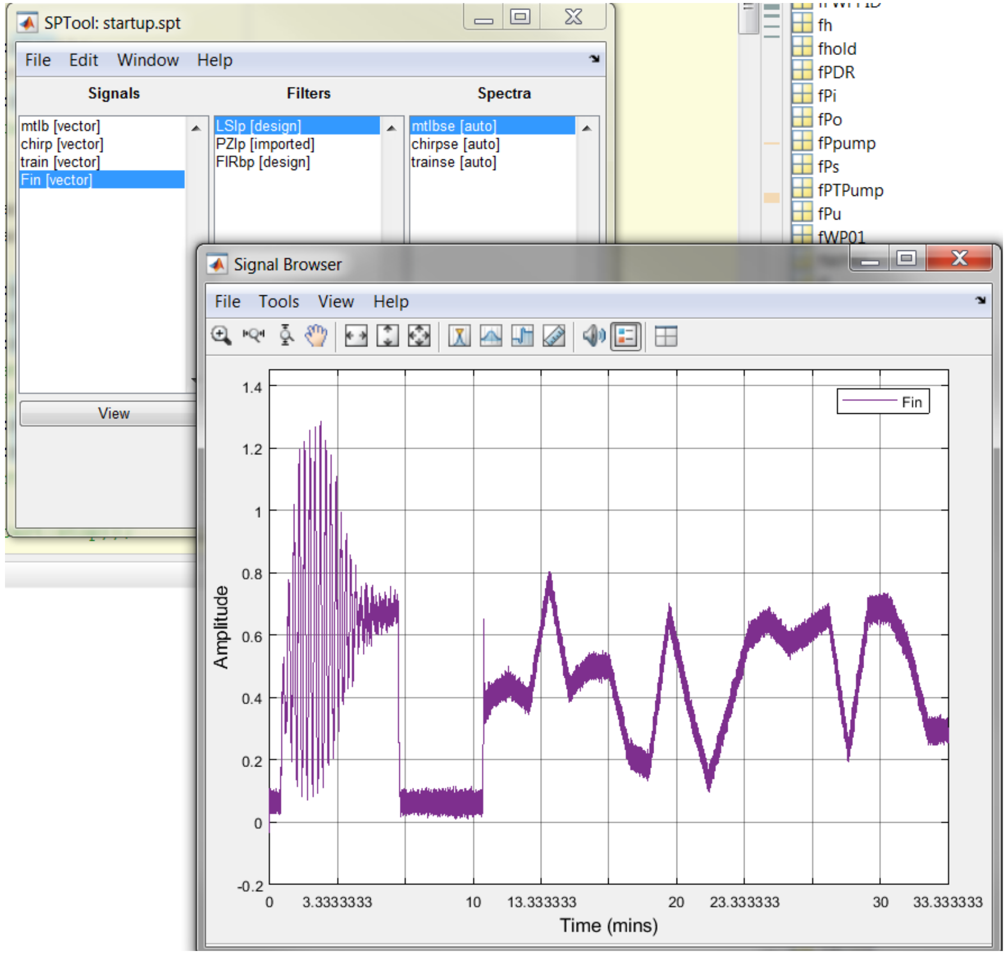
Practical session: introduction to SPTOOL

- Import signals
- Filter design
- Spectral analysis



Practical session: introduction to SPTOOL

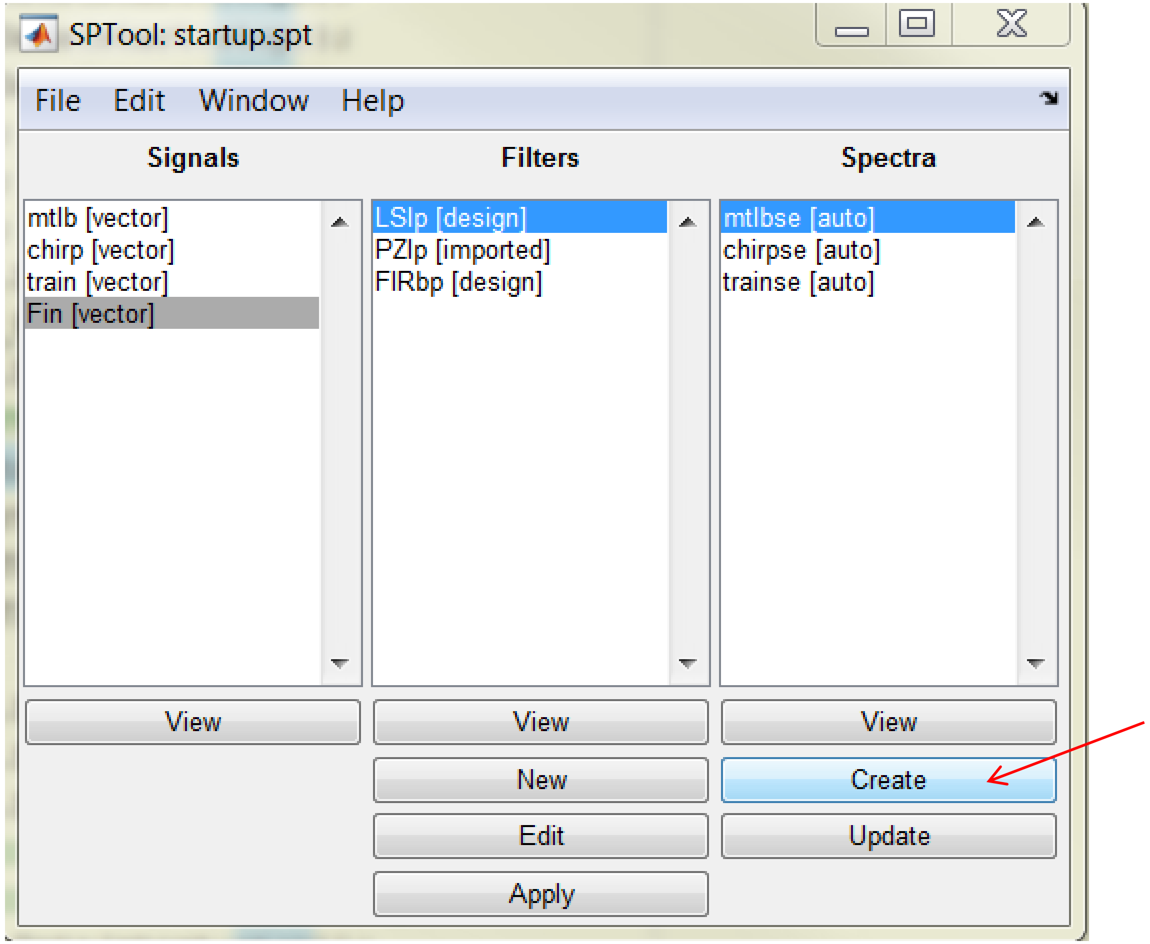
- Import signals
- Filter design
- Spectral analysis



AALBORG UNIVERSITY
DENMARK

Practical session: introduction to SPTOOL

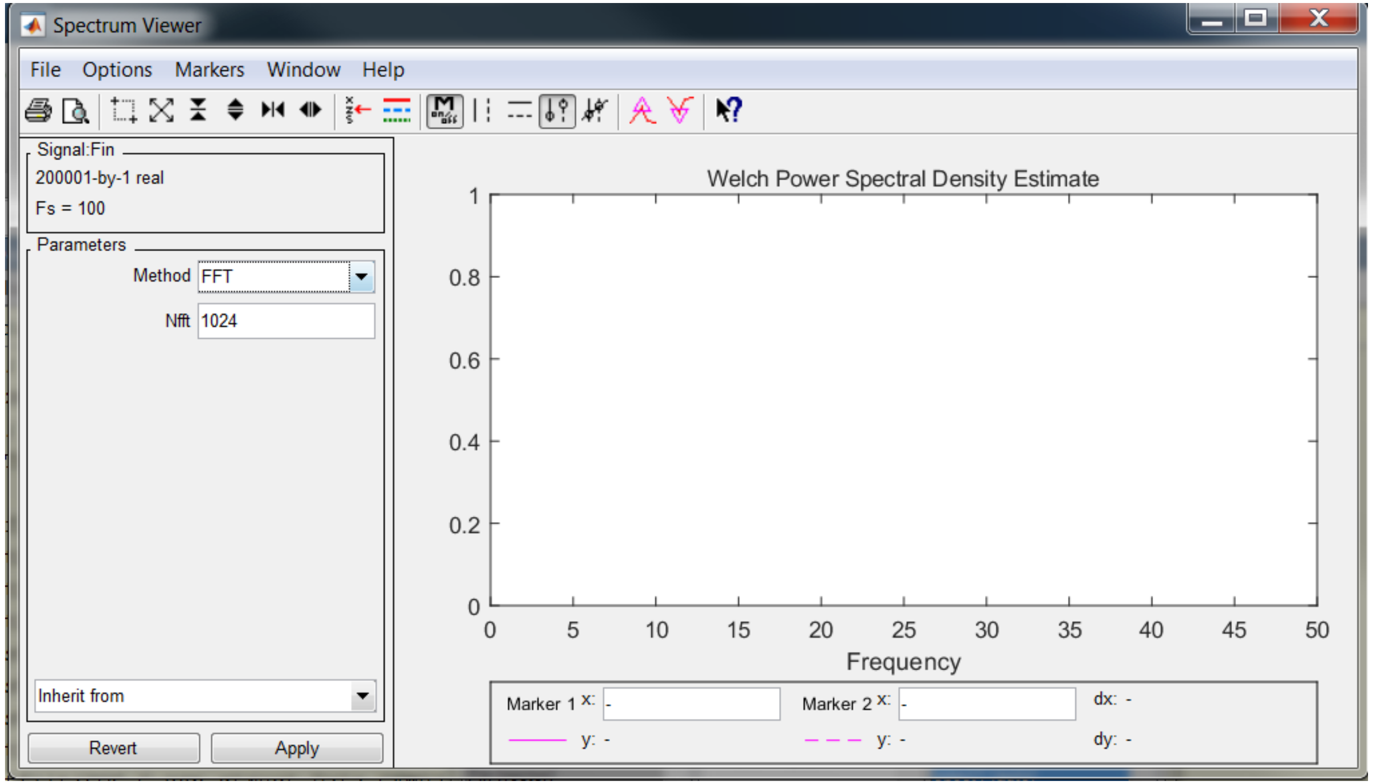
- Spectral analysis



AALBORG UNIVERSITY
DENMARK

Practical session: introduction to SPTOOL

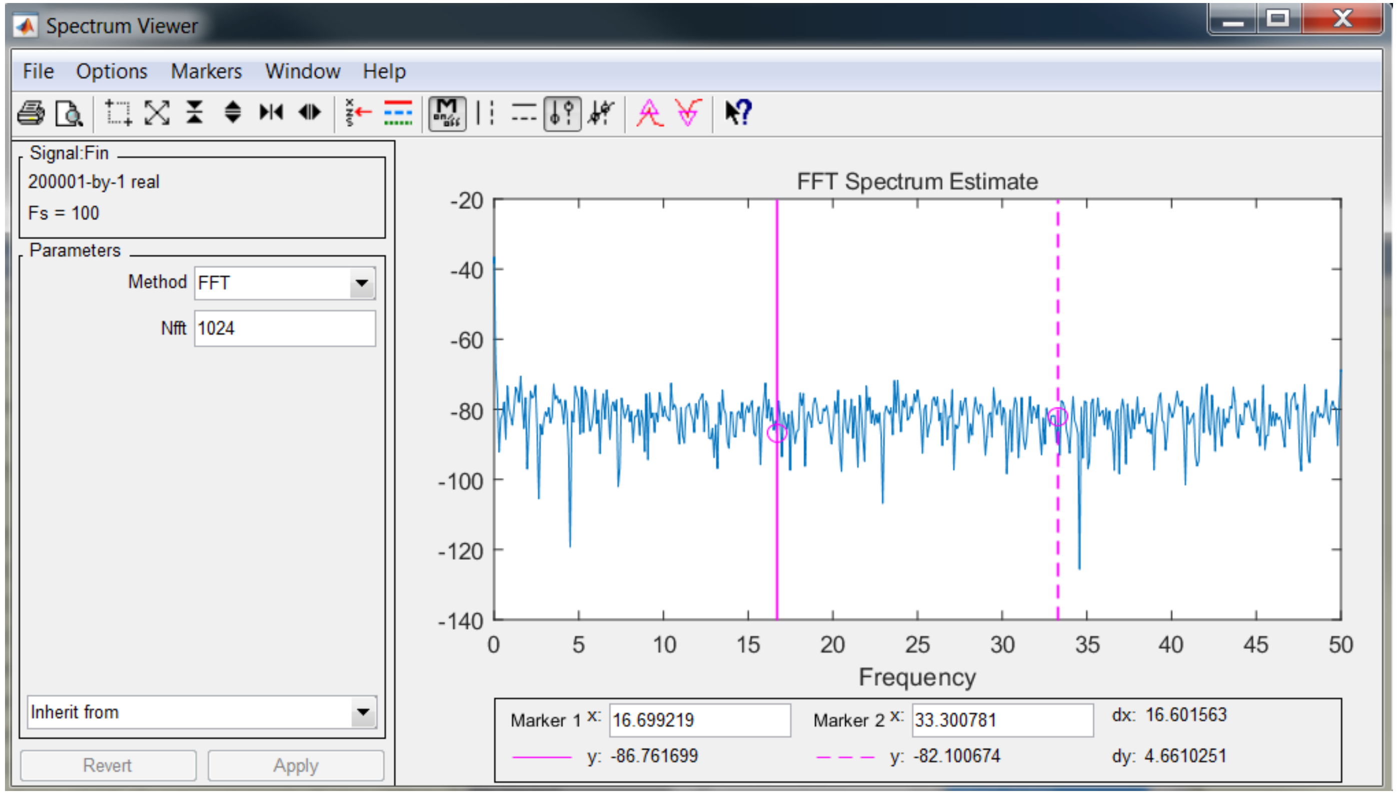
- Spectral analysis
  - Method – There are different methods, use FFT in this course
  - Nfft – resolution for the FFT plot.
  - Nwind – size of the window
  - Window – Window type.



AALBORG UNIVERSITY
DENMARK

Practical session: introduction to SPTOOL

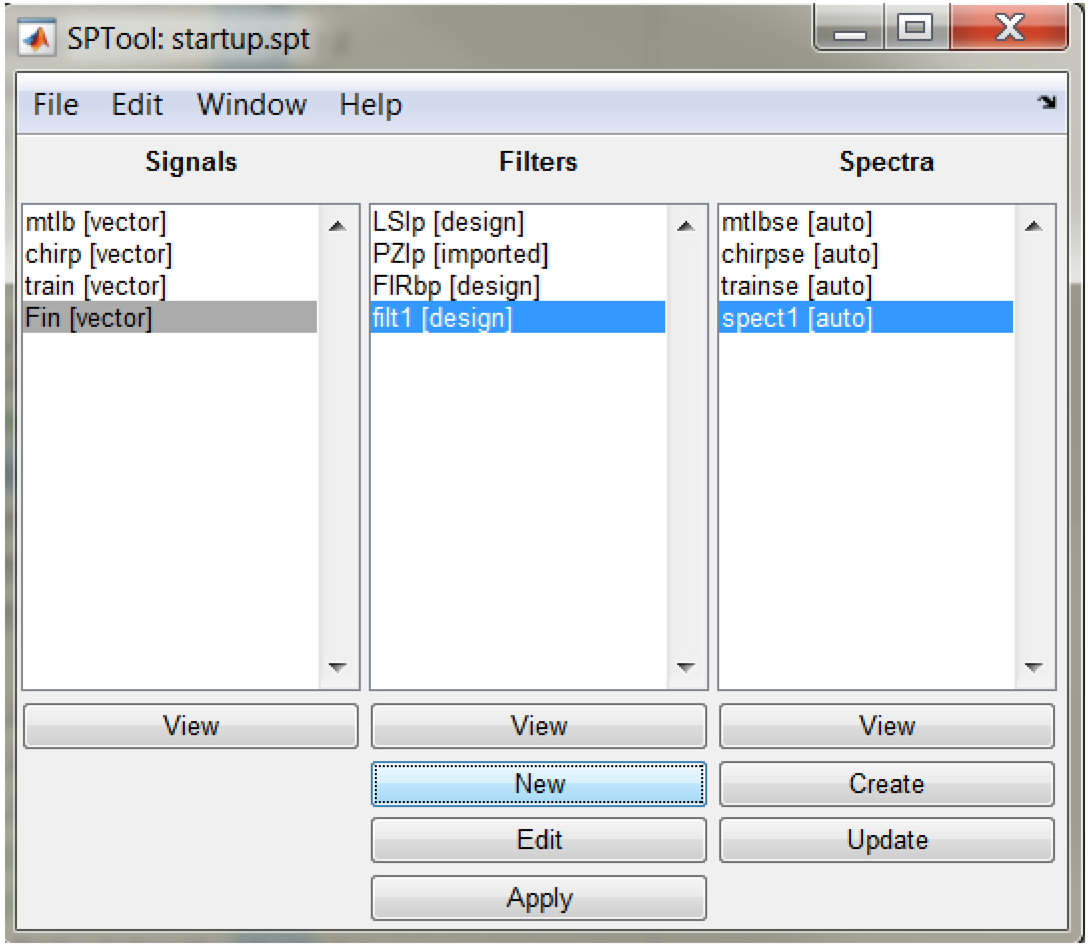
- Spectral analysis
  - Method – There are different methods, use FFT in this course
  - Nfft – resolution for the FFT plot.
  - Nwind – size of the window
  - Window – Window type.



AALBORG UNIVERSITY
DENMARK

Practical session: introduction to SPTOOL

- Filter design
    - A link to the filter design toolbox

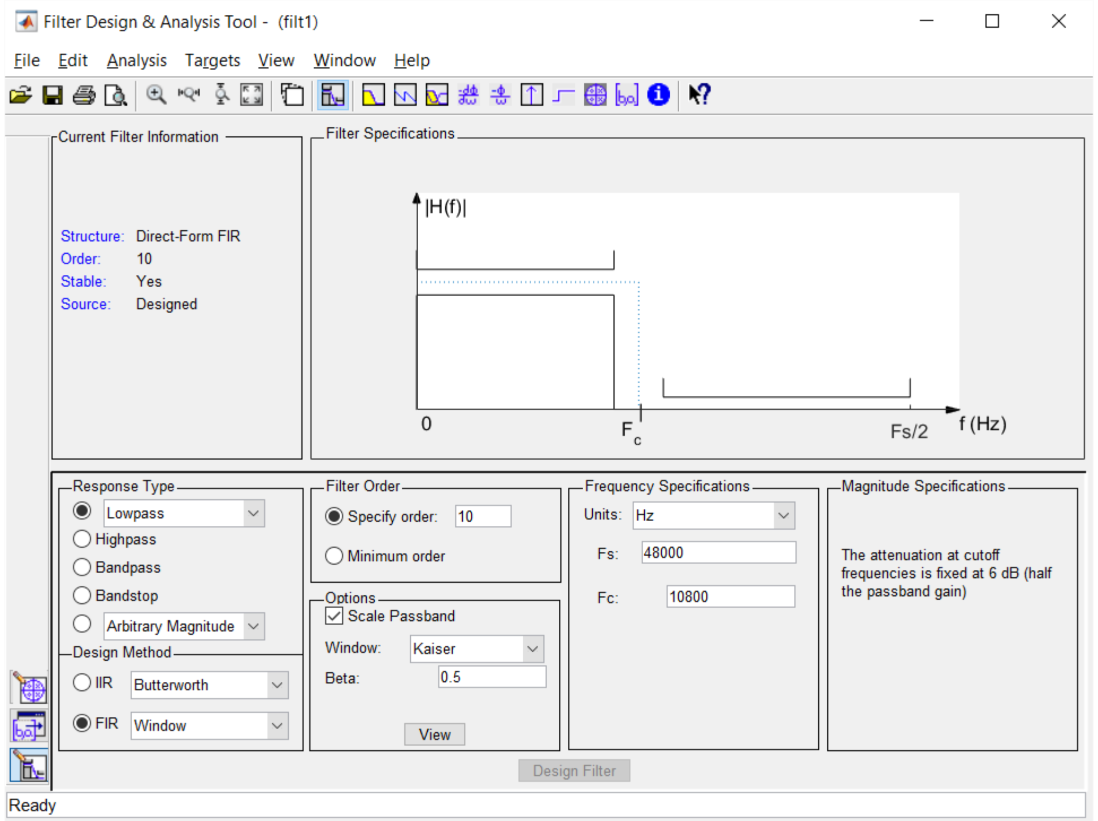


FILTER DESIGN USING FDATOOL

Practical session: introduction to FDATOOL

- Design of the filter
  - Response type
  - Design method
    - Filter order
    - Other options (this changes for differnet filter types)

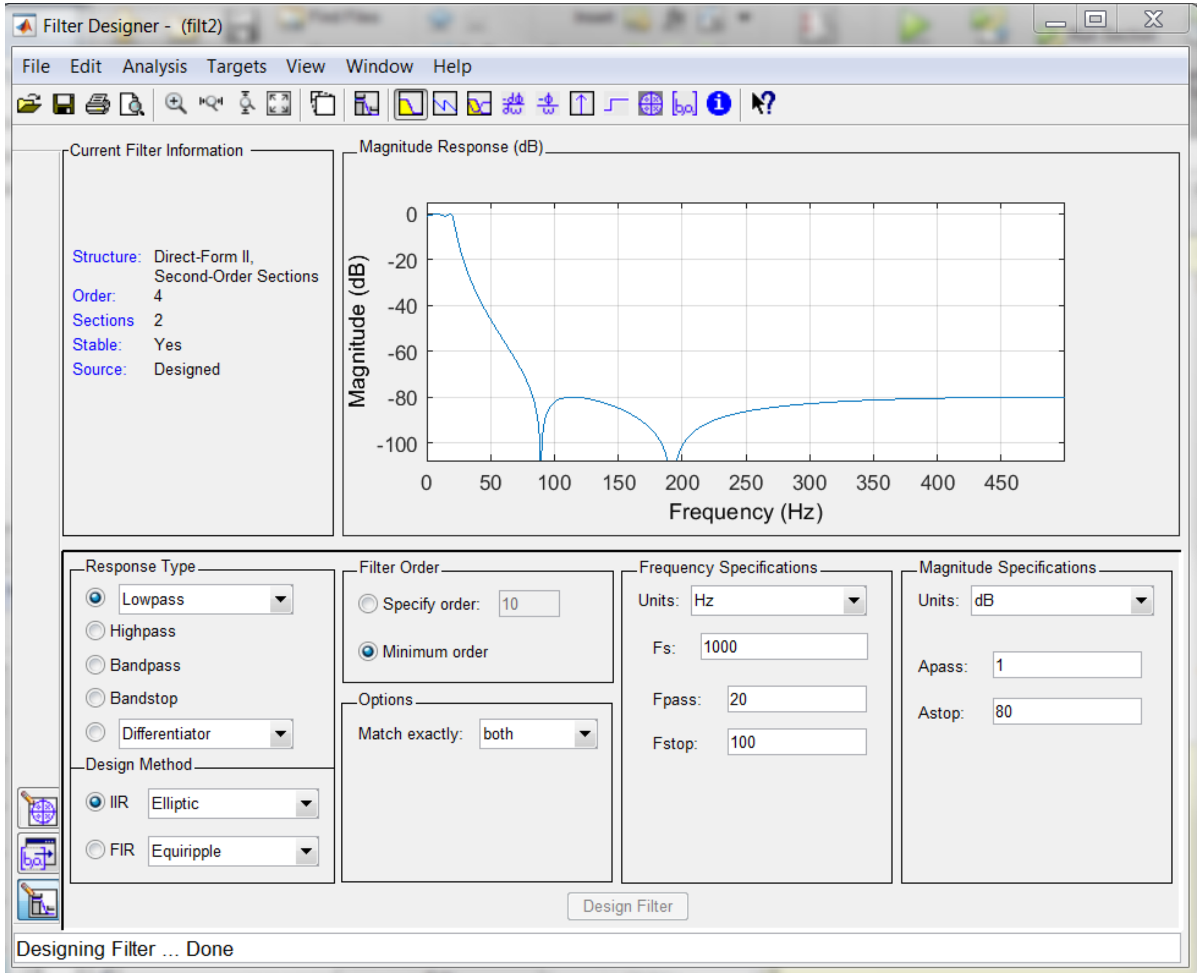
- Observe the following charactetristis of the filter
  - Magnitude/phase respons
  - Group delay
  - Passband/stopband ripple if any



Practical session: introduction to FDATOOL

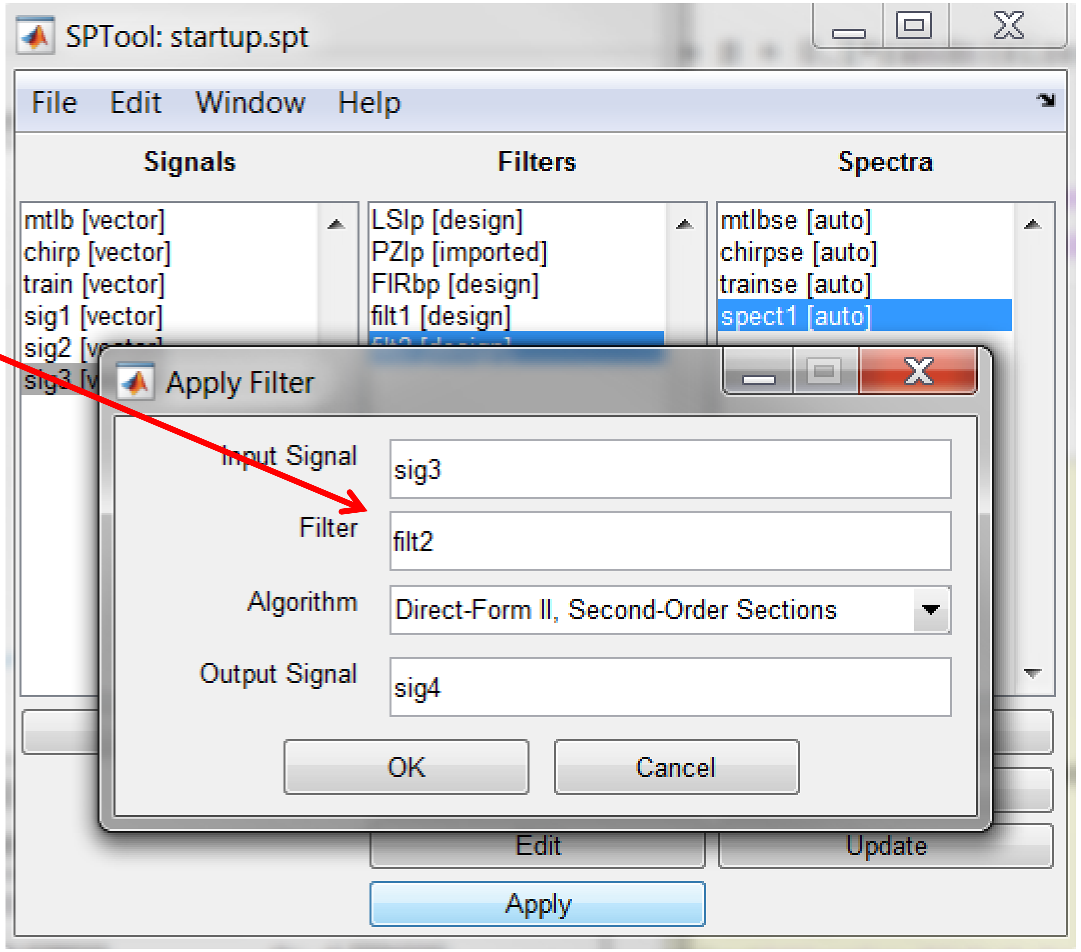
- Design of the filter
  - Response type
  - Design method
    - Filter order
    - Other options (this changes for different filter types)

- Observe the following characteristic of the filter
  - Magnitude/phase response
  - Group delay
  - Passband/stopband ripple if any



Practical session: introduction to FDATOOL

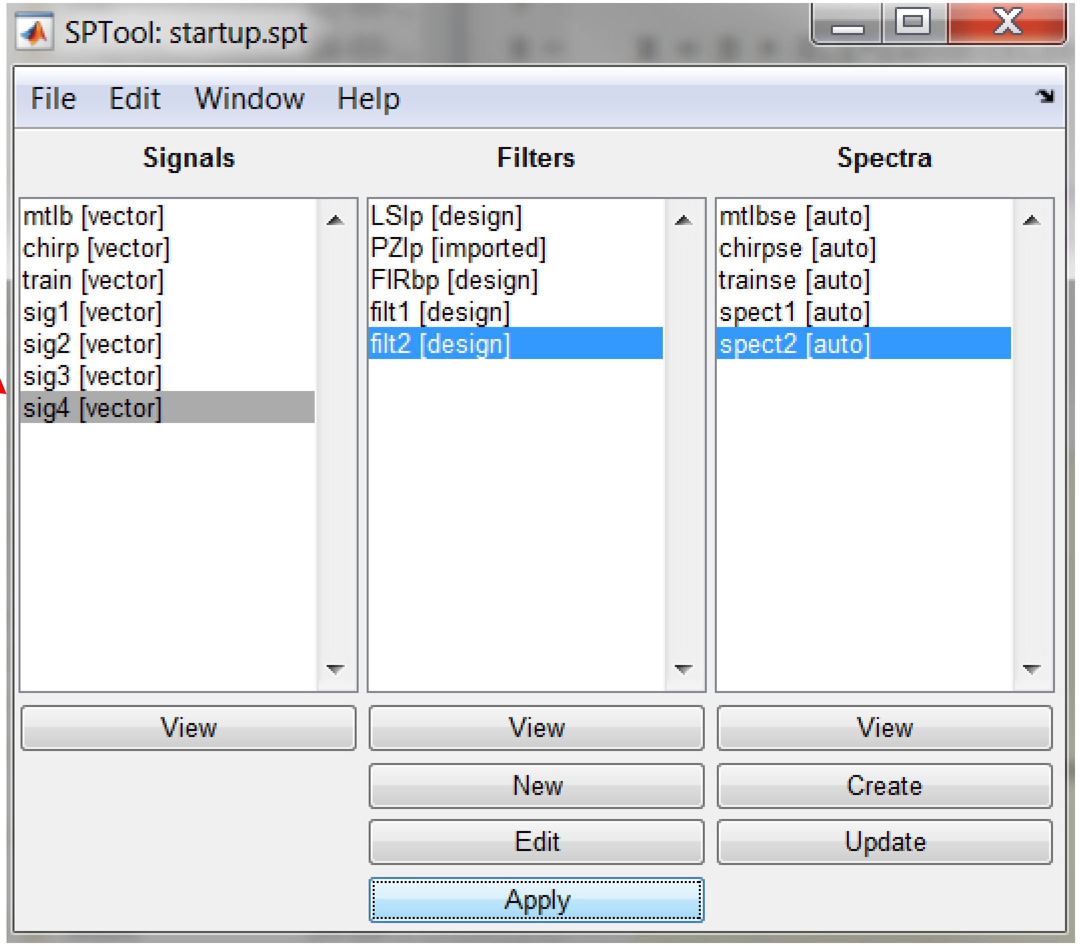
- Apply the filter using SPTOOL



AALBORG UNIVERSITY DENMARK

Practical session: introduction to FDATOOL

- Apply the filter using SPTOOL
- New filtered signal now appears in Signals



Practical session: introduction to FDATOOL

- Apply the filter using SPTOOL
- New filtered signal now appears in Signals
- Analyze spectrum of the filtered signal