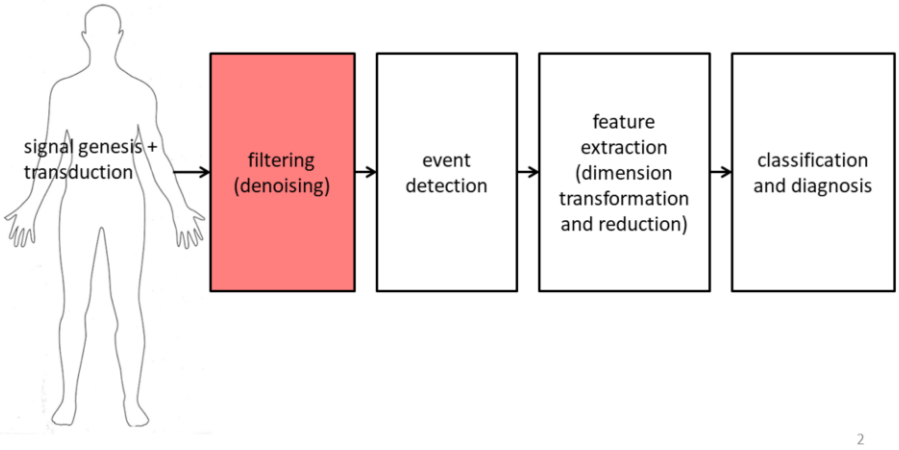
Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Biomedical Signal Processing
2018-2019 Autumn

Noise Filtering

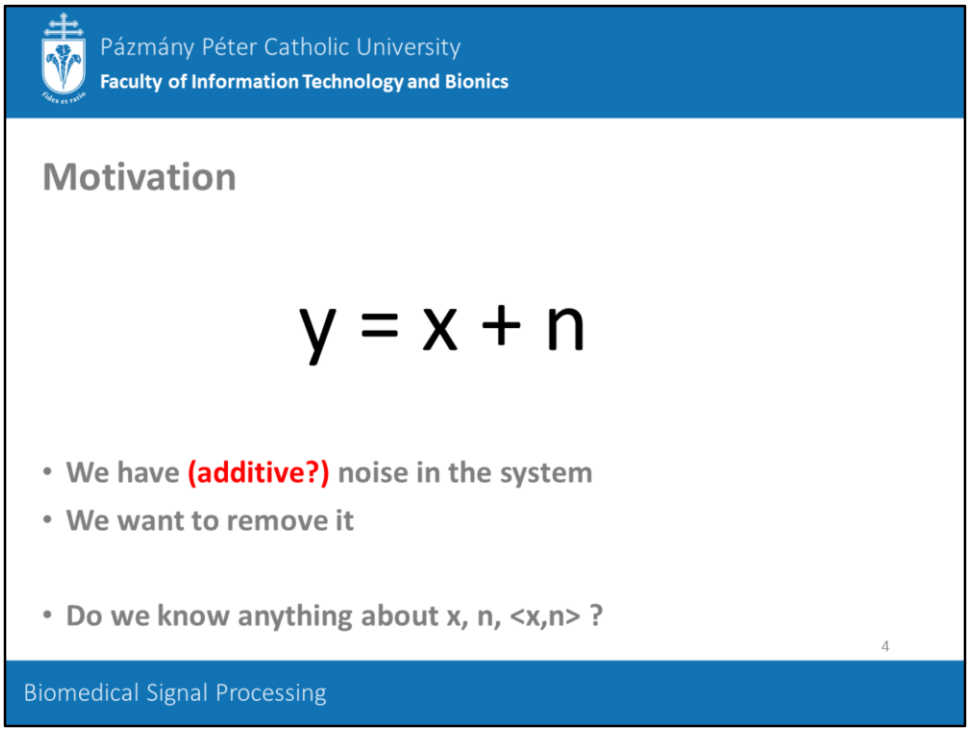
Lecturer: Janka Hatvani
Responsible lecturer: dr. Miklós Gyöngy

The BSP Flow Chart



Today's goal

- **What kind of noises are possible?**
- **Learning the theory of the following filters:**
  - Synchronized averaging
  - Ad hoc filters (moving average, pole-zero, band filters)
  - Wiener filtering
  - Adaptive filtering
- **Hopefully getting an idea, what kind of filter to use for a given task**



The goal is to eliminate the additive noise from the system.
We have to know, whether the noise is independent from the signal, and whether it is random.
If not, what is their relation? It has to be included into the model.

If noise is multiplicative, take log, then filter (homomorphic filtering)

ECG noise

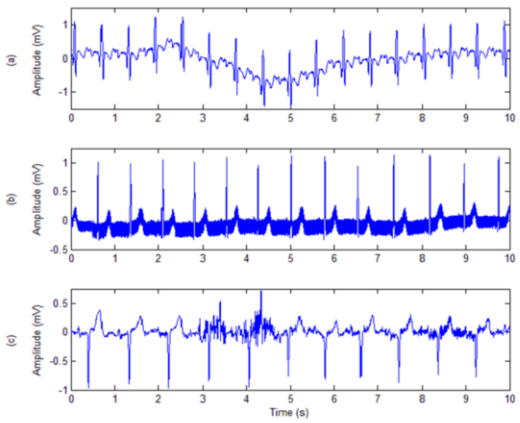
~<1 Hz: baseline wander (cable movement, loose electrode)

<1 Hz: respiration-induced changes in beat morphology and amplitude

1-10 Hz: electrode motion artefacts

0-300 Hz: electromyographic (EMG) noise

50/60 Hz: power-line interference



Common types of noise in ECG recordings. (a) Baseline wander, (b) 50 Hz power line interference, and (c) Electromyographic noise.

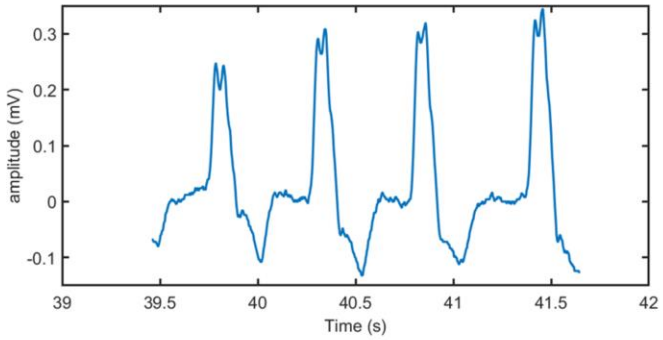
An example is the ECG signal. It has multiple noise components, as listed above.

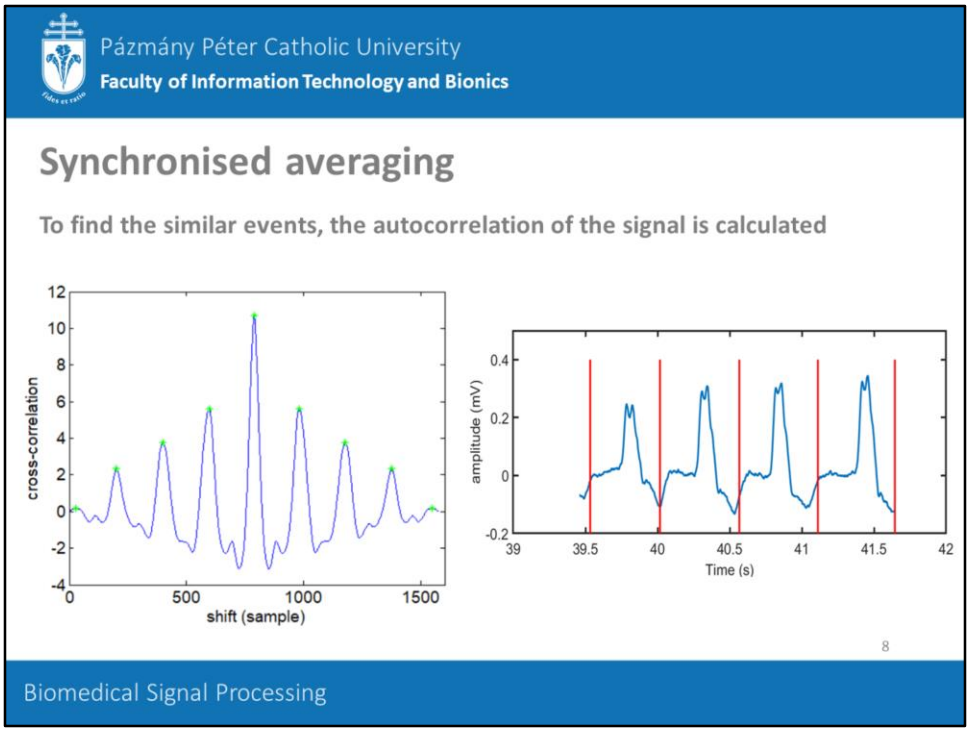
Techniques

- **Synchronised averaging**
- **Ad hoc filters**
  - moving average
  - pole-zero design
  - filter classes (Butterworth, Chebyshev)
- **Wiener filtering**
- **Adaptive filtering**

Synchronised averaging

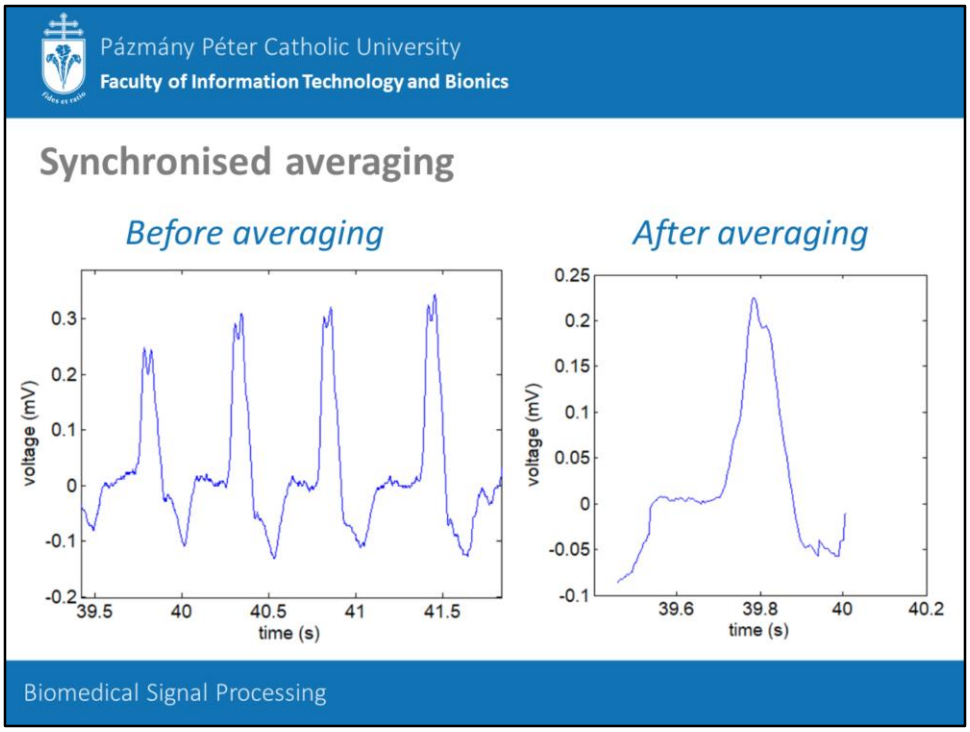
In this method similar segments are cut out from the signal, they are aligned, and averaged. In this case the cardiac cycles are collected





To find the similar parts of the signal, the autocorrelation is calculated. This plot shows, that with each shift, how big the correlation between the original and its shifted version is. Note, that this plot is symmetric, because shifts are calculated in both + and – directions.
At the peak shifts the cycles overlap, there the signal can be cut. (red lines)

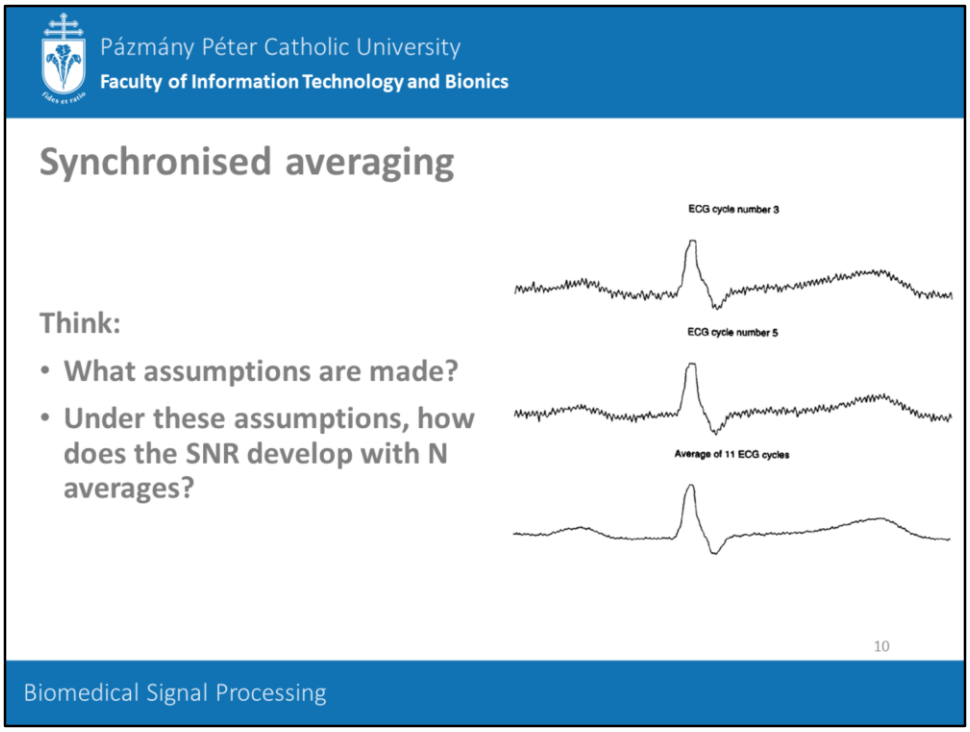
If you have annoted data, like the QRS location of ECG signals, the autocorrelation step can be skipped, and the data can be cut using these reference points.



When is it good?

It will take away important informations from a cardiac signal.

It can be useful when we have multiple measurements of the same event (from multiple detectors, or if the signal is assumed to be constant. This can be an ultrasound measurements of the tissue. Another example is evoked potentials: in the oddball paradigm a negative potential can be observed at 300 ms upon the the sound of the deviant stimulus. From one EEG measurement this would be very noisy, but using synvhronized averaging this can be avoided.



When you do time domain averaging on the vibration signal from a real machine, the averaged time record gradually accumulates those portions of the signal that are synchronized with the trigger, and other parts of the signal, such as noise and any other components such as other rotating parts of the machine, etc., are effectively averaged out.
The method assumes that **similar events** build up the signal. We also assume, that the **noise is not correlated** (with itself or with the signal)!!!
If we have an important anomaly, it will disappear.

Linear filters (wiener filter, pole-zero filter, butterworth-chebyshev-elliptic filters) fail to work when the spectrum of the noise and that of the signal overlap. With synchronized averaging this problem can be solved.

http://azimadli.com/vibman/synchronousaveraging.htm

Why is it working?
The useful signal is assumed to be the same through cycles, so its average is the same. The white noise, on the other hand, flattens out with more and more average. The SNR improves with the number of samples, N.

$$y = x + n$$

The SNR can be calculated as:

$SNR = \frac{S}{\sigma^2}$, signal power ($E(x^2)$) over noise variance

The variance is the square of the standard deviation, which is constant for the noise:
Var(X) = $E[(X-\mu)^2] = E[X^2] - E[X]^2 = Cov(X,X)$
$E(n^2) = var(n) = \sigma^2$

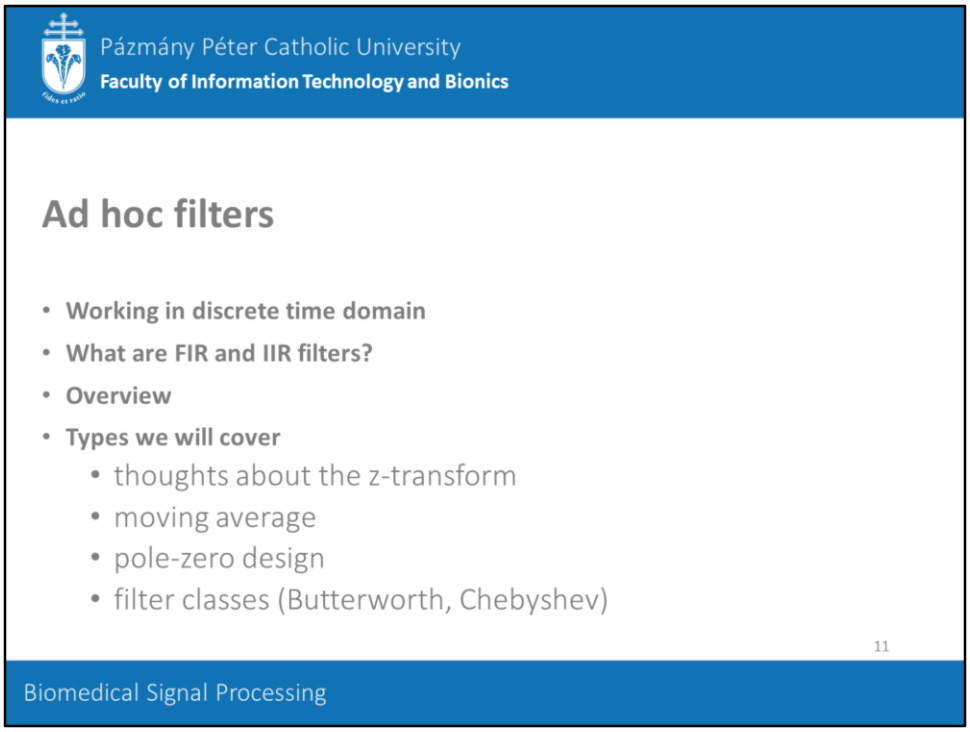
Rule for variances: for two **uncorrelated** variables
$var(a+b) = var(a) + var(b)$
For the same variable, however, there is a square **scaling** if a scalar is present:
$var(ka)$ = $k^2\ var\ (a)$

Variance of averaged noise:

$$var\left(\frac{1}{N}\sum_{i=1}^{N} n_i\right) = \frac{1}{N^2}\sum_{i=1}^{N} var(n_i) = \frac{1}{N^2} N\,\sigma^2 = \frac{1}{N}\sigma^2$$

Power of averaged signal is the same, as that of 1 signal, S (as we assume that x is identical through cycles)

$$SNR_{avg} = \frac{S}{\frac{1}{N}\sigma^2} = N \cdot SNR_1$$

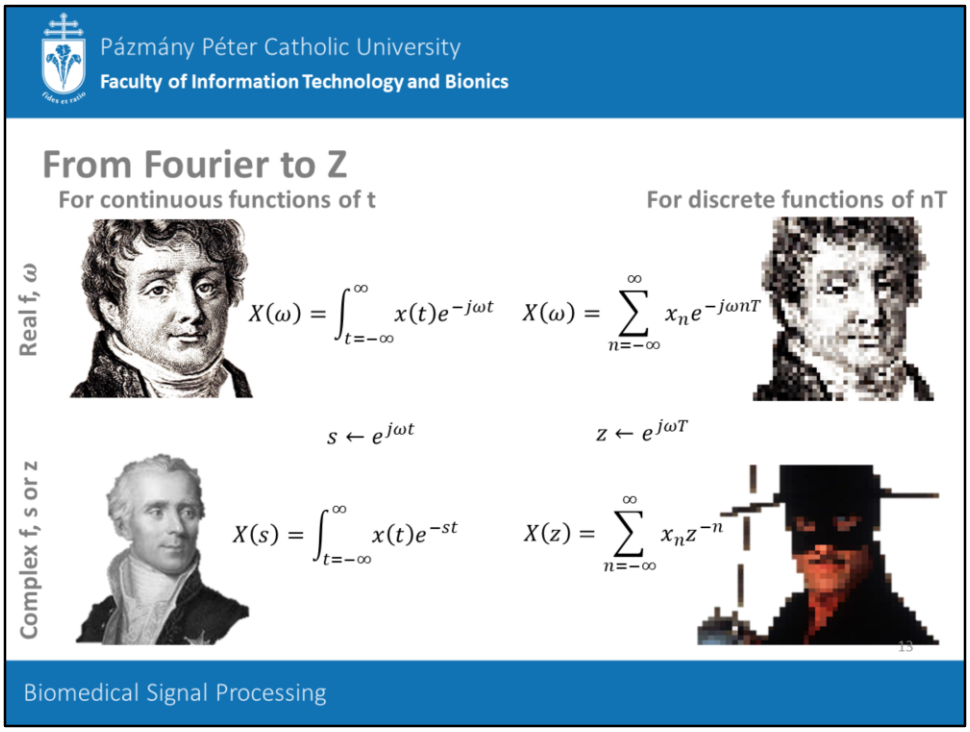


„Ad hoc is a Latin phrase meaning literally "for this." In English, it generally signifies a solution designed for a specific problem or task, non-generalizable, and not intended to be able to be adapted to other purposes”

https://en.wikipedia.org/wiki/Ad_hoc

These filters can be written in the form of a transfer function of a system. (See slides of previous lecture for the definition of a transfer function.

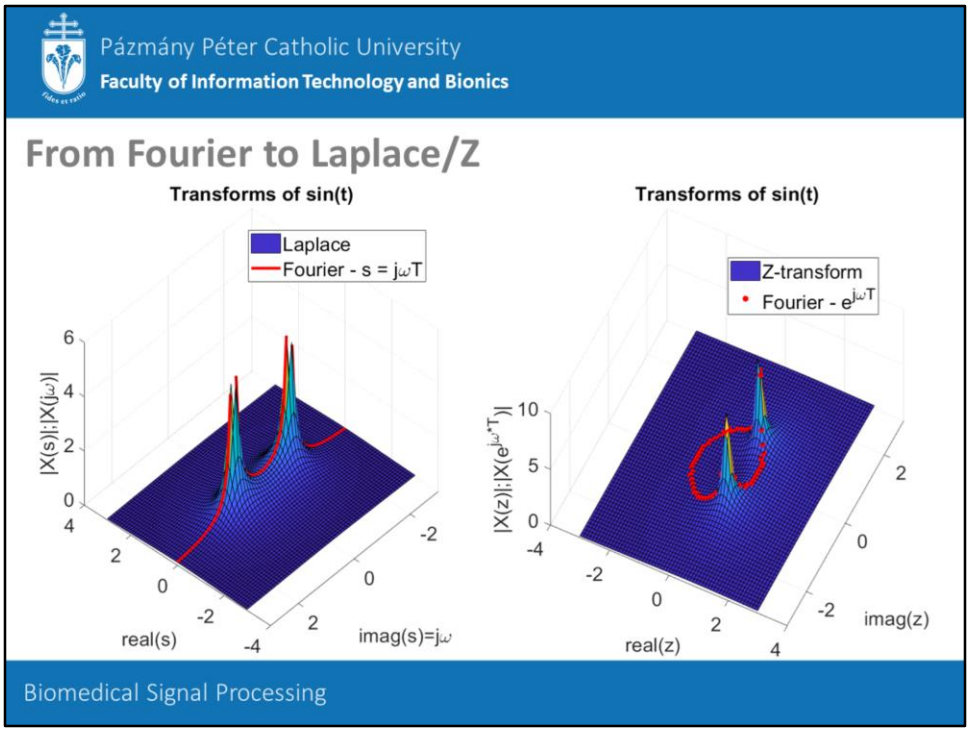
Pole: a substitution value s of the transferfunction $G(s)$, where its denominator is zero, causing a a division by 0 → infinite value of $G(s)$ (going to the sky)
Zero: a substitution value of $G(s)$, where the nominator is zero → zero value of $G(s)$ (by the ground)



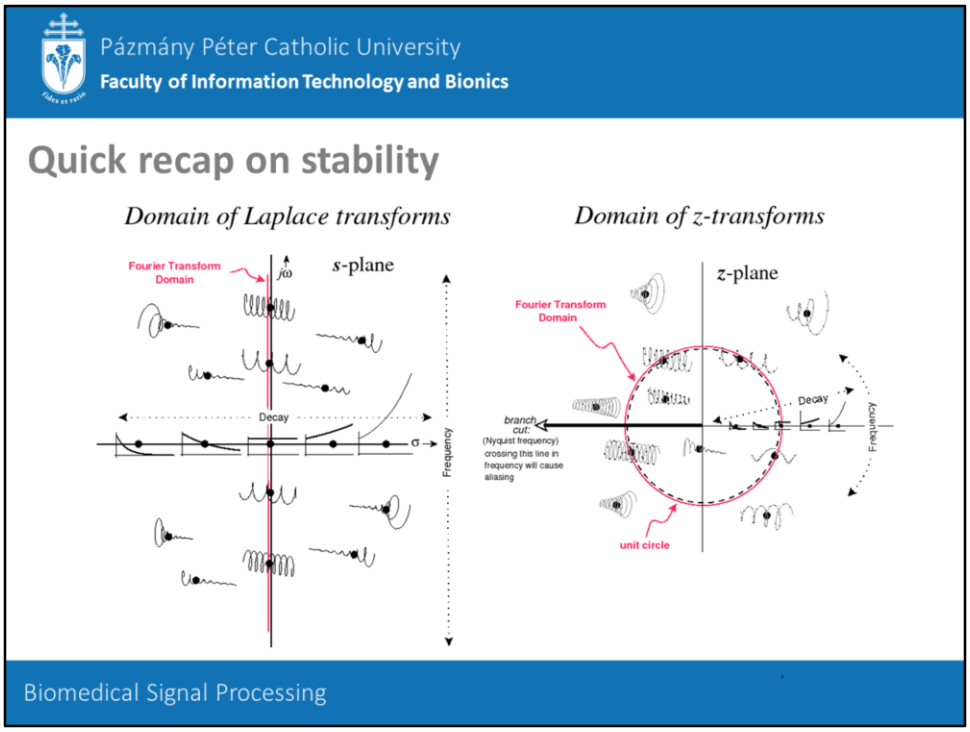
The continuous Fourier transform maps a continuous function x(t) to the real-valued function X(ω), $\omega \in \mathbb{R}$ (if $X(\omega)$ is calculated only for discrete ωs, than it is a fourier discrete function).
The Laplace transform maps a continuous function x(t) to the complex-valued function X(s) , $s \in \mathbb{C}$
The discrete Fourier transform maps a discrete function x(n) to the real-valued function X(ω), $\omega \in \mathbb{R}$
The discrete Z-transform maps a discrete function x(n) to the complex-valued function X(z) , $z \in \mathbb{C}$

The discrete form of the Fourier transform converts into the discrete Z-transform with a simple substitution of $z \leftarrow e^{j\omega T}$
„It gives a tractable way to solve linear, constant-coefficient difference equations."

https://en.wikipedia.org/wiki/Z-transform



The Laplace transform is a complex-valued function, it takes frequency values on the whole frequency plane. The Fourier transform is real valued, it is drawn along the imaginary axis of the Laplace trasform.

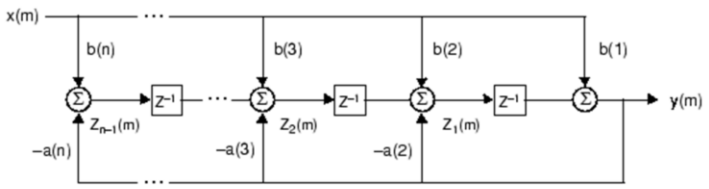


Laplace transform:
If this **pole is on the negative half-plane**, the solution is stable (the exponential is not exploding, as discussed in the previous lecture)

Z-transform ($z \quad e^{j\omega T}$)
In the **discrete form**, using the **Z-transform**, the **poles have to be within the unit circle** to have a stable solution.

The doamin of the Fourier transform is the imaginary axis of the Lablace transorm (or unit circle of the Z-transform, no damping). s = $j\omega$

Filters in the z-domain

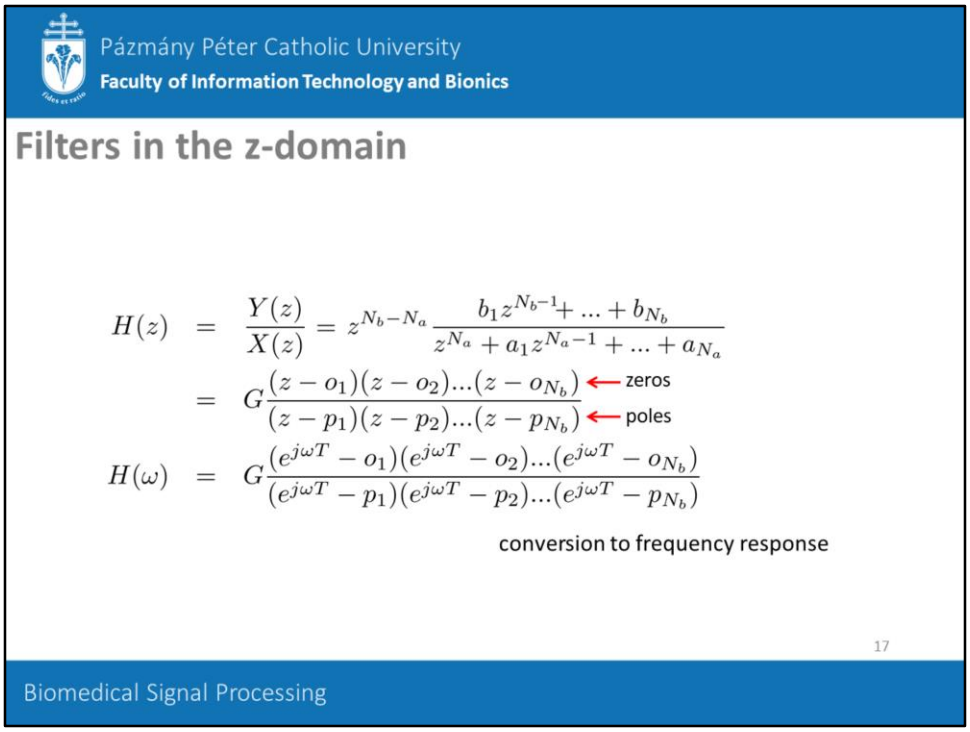


$$
\begin{aligned}
y_n + a_1 y_{n-1} + \ldots + a_{N_a} y_{n-N_a} &= b_1 x_n + \ldots + b_{N_b} x_{n-N_b} \\
\frac{Y(z)}{X(z)} &= \frac{b_1 + b_2 z^{-1} + \ldots + b_{N_b} z^{-N_b}}{1 + a_1 z^{-1} + \ldots + a_{N_a} z^{-N_a}} \\
&= z^{N_b - N_a} \frac{b_1 z^{N_b} + \ldots + b_{N_b}}{z^{N_a} + a_1 z^{N_a - 1} + \ldots + a_{N_a}}
\end{aligned}
$$

16

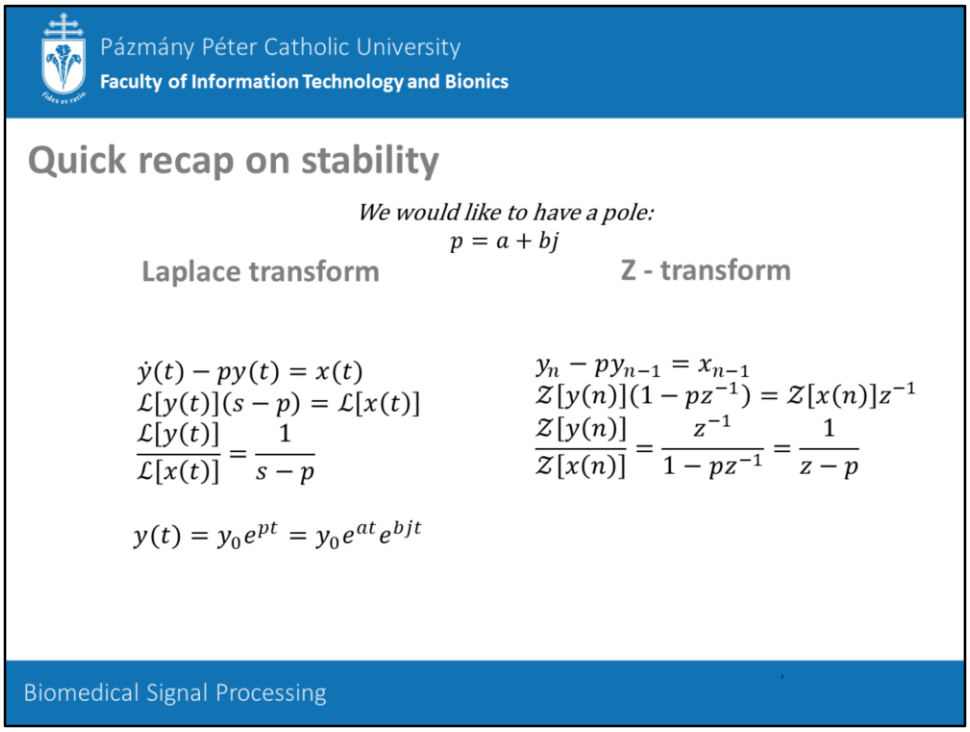
From up to down:

- Signal flow diagram of the system
- I/O model of the system. x_i is the input datapoint, y_i is the output datapoint at time i. The order of the equation is N_a.
- Z-transform of the system, in the form of the transfer function



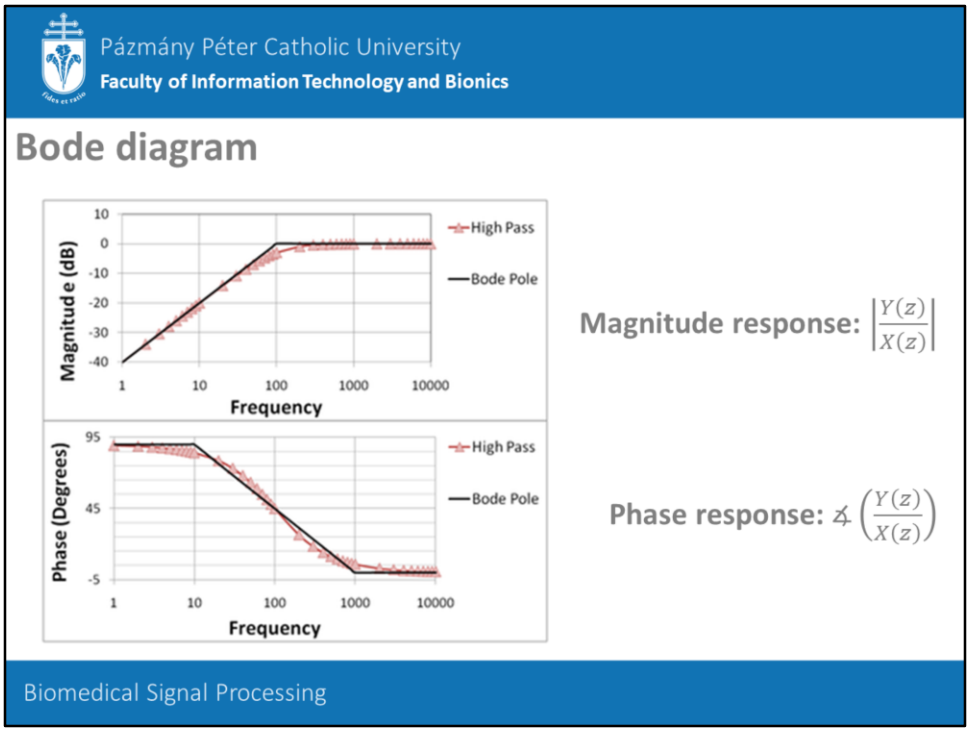
As described previously, the zeros can be calculated as the roots of the nominator, the poles are the roots of the denominator.

If we go from the Z-domain to the Fourier domain, only a simple substitution has to be done: $z \leftarrow e^{j\omega T}$
Calculating in the Z-domain is easier, in the Fourier form the physical frequency response can be read.



$p \in \mathbb{C}$ is a complex number.
A simple **continuous** diff.eq. is $\dot{y} = py + x$. The solution is in the form of $y_0 e^{at} e^{bjt}$ using the **Laplace transform**, the pole is p.

If this **pole is on the negative half-plane**, the solution is stable (the exponential is not exploding, as discussed in the previous lecture)
In the **discrete form**, using the **Z-transform**, the **poles have to be within the unit circle** to have a stable solution.



Bode diagram of a low-pass filter

To see the magnitude and phase response of a filter, the Bode diagam should be drawn. It will tell us, how the filter modifies signals with different frequencies.

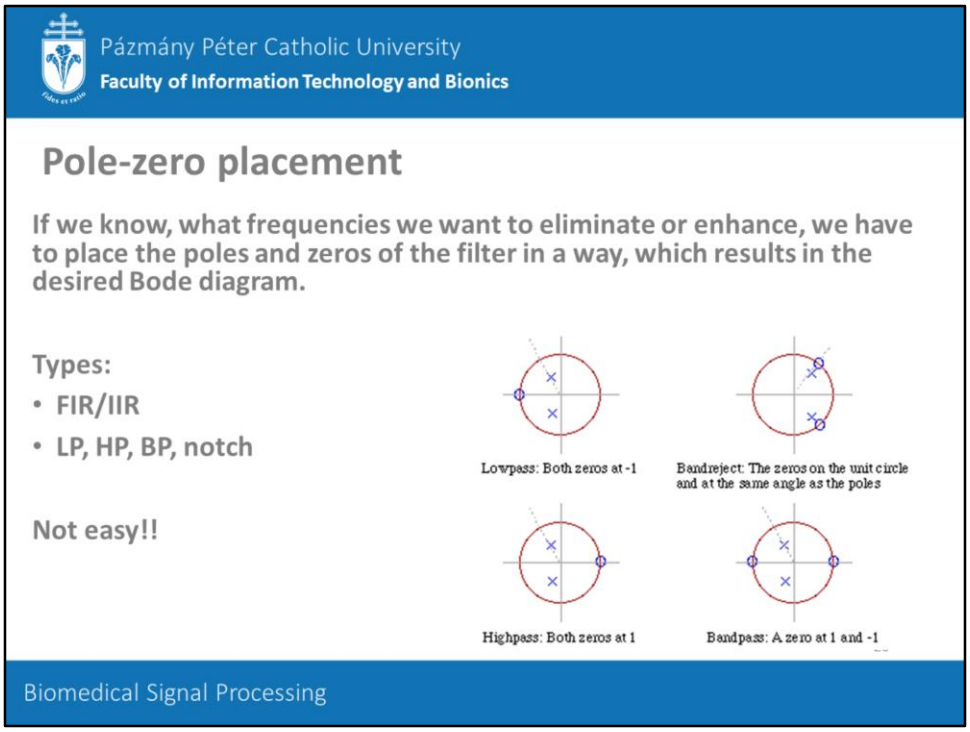
Magnitude response: $\left|\frac{Y(z)}{X(z)}\right|$, absolute value of the transfer function

Phase response: $\measuredangle\left(\frac{Y(z)}{X(z)}\right)$, the angle of the transfer function

If i know, what frequencies I want to eliminate or enhance, I have to place the poles and zeros of the filter in a way, which results in the desired Bode diagram.

Note that magnitude is in dB!! $-20\log_{10}(\frac{output}{input})$

- if output = input , log(1) = 0
- If output = input*0.01), $-20\log_{10}(100) = -40$



Location of poles and zeros: influences magnitude
Location of poles: influences stability
Location of zeros: determines phase linearity

If we have zeros in the system, we are talking about an IIR (infinite impulse response, as the output depends on previous output(s)). Otherwise, we are talking about a FIR filter. FIR filters are usually easier to design, but for nice properties a high order (many electronic components) might be needed. On the other hand, IIR filers can be computationally more efficient, as the complex networks can be implemented with a lower order → lower number of components. The same response function can usually be implemented by both FIR and IIR structure.

From the perspective of the frequency response we can design lowpass, highpass, single/multi bandpass and notch filters

Coming up: MATLAB demo on wiki zfiltdes.m
Wolfram demo (e.g. http://demonstrations.wolfram.com/TransferFunctionAnalysisByManipulationOfPolesAndZeros/)
FilterDesign pdf (http://faculty.ksu.edu.sa/ghulam/Documents/CEN352/DSP_CEN352_FilterDesign.pd

f)
dsp_iir.pdf (http://staff.neu.edu.tr/~fahri/dsp_iir.pdf)

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Moving-average filters

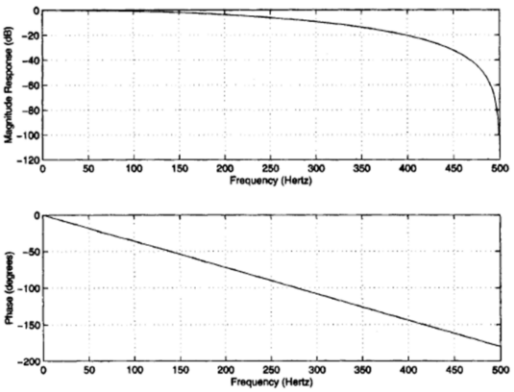
$$y_n = \frac{1}{2}(x_n + x_{n-1})$$

$$y_n = \frac{1}{4}(x_n + 2x_{n-1} + x_{n-2})$$

$$y_n = \frac{1}{8}(x_n + \ldots + x_{n-7})$$

Causality: only ‚older samples' are used

response of Hanning [1 2 1] filter



21

Biomedical Signal Processing

In moving average filters a number of previous inputs is averaged to obtain the i[th] output of the filter. These filters in the above implementation have no poles, only zeros.

Moving average filters in the I/O form on the left, and the frequency and phase response of the Hanning-filter (framed in red on the left).

Poles : $\emptyset$ → FIR filter

Zeros: $z^2 + 2z + 1 = 0 \rightarrow \frac{-2 \pm \sqrt{4-4}}{2} = 1$, zero is at „full frequency" (the amplitude at this frequency is attenuated heaviliy)

The magnitude is in dB!!! 0 means no difference (log(1)), very negative numbers mean big attenuation

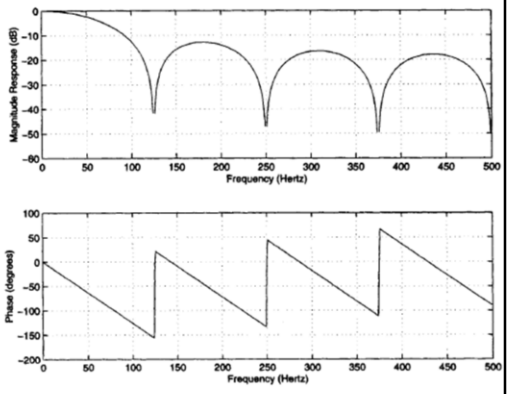
Moving-average filters

$$y_n = \frac{1}{2}(x_n + x_{n-1})$$

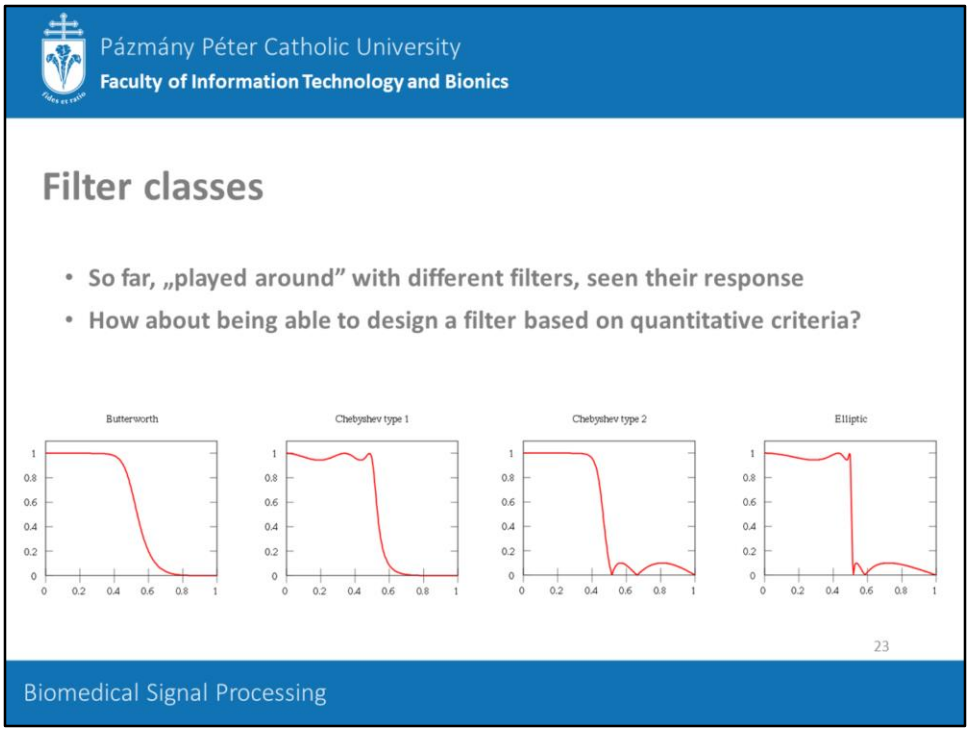
$$y_n = \frac{1}{4}(x_n + 2x_{n-1} + x_{n-2})$$

$$y_n = \frac{1}{8}(x_n + \ldots + x_{n-7})$$

response of 8-point moving average filter



Poles: ∅ → FIR filter
Zeros: at ¼, ½ , ¾ , 1 normalized frequency (the amplitude at this frequency is 0)



Magnitude response of the filters

- Butterworth filter: maximally flat magnitude filter, with a mild slope after the cutoff frequency. It means, that the frequencies which are passed are preserved smoothly, but after the cutoff frequency many frequencies remain present with some amplitude.
- Chebysev type I: it has ripples in the pass-band, but has a sharp slope, and the eliminated frequencies are cut smoothly.
- Chebysev type II: it has a smooth pass band, sharp rolloff, but ripples in the eliminated frequency.
- Elliptic: ripples in both pass and cut band, but very sharp rolloff.

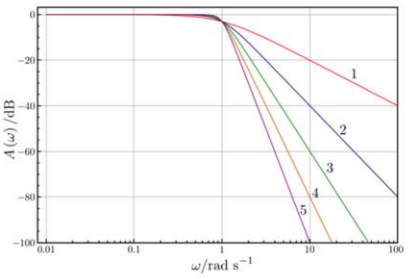
Our choise of filter depends on the task. Do we want to keep the passed frequencies perfectly, or eliminate the cut-band completely? Do we need a sharp cutoff?

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Butterworth filter

- LP: $|H(j\omega)|^2 = \frac{1}{1+(\omega/\omega_c)^{2N}}$
- Filter completely defined by cut-off frequency ω_c and filter order N
- Completely flat (no ripples)
- $\left.\frac{\partial^i |H(\omega)|}{\partial \omega^i}\right|_{\omega=0} = 0, i \leq 2N-1$
- Matlab:

```
[b,a] = butter(N,Wn)
[b,a] = butter(1,0.5)
b=[0.5 0.5]
a=[1   1  ]
```



note 20*N* dB/decade rolloff

24

Biomedical Signal Processing

When the Butterworth filter is designed as a lowpass filter, its magnitude response is: $|H(j\omega)|^2 = \frac{1}{1+\left(\frac{\omega}{\omega_c}\right)^{2N}}$, where ω_c is the **cutting frequency**, and N is the **order** of the filter.

These are the 2 parameters that have to be defined also in matlab. (N, W_n)

The magnitude response is $(2N-1)$-times differentiable. The derivate in the pass-band is zero → completey flat, there are no ripples. It can be seen in the plot, that the higher the filter order, the steeper the rolloff is.

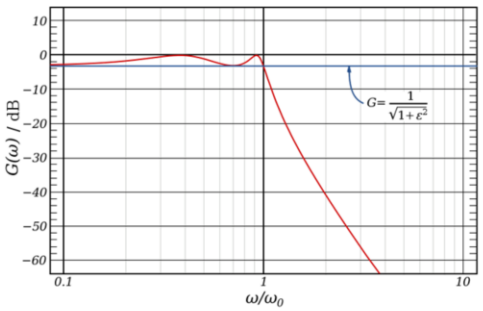
Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

Chebyshev filter (Type 1)

- Very similar to Butterworth
- Chebyshev polynomial function T_n modulates response, causing ripples in the pass-band but also a steeper roll-off

$$|H_n(j\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 T_n^2\left(\frac{\omega}{\omega_0}\right)}}$$

```
% Matlab:
[b,a] = cheby1 (n,eps,Wc)
% eps is in dB
```



25

Biomedical Signal Processing

The magnitude response is

$$|H_n(j\omega)| = \frac{1}{\sqrt{1+\epsilon^2 T_n^2\left(\frac{\omega}{\omega_c}\right)}},$$

where ϵ is the ripple factor, ω_c is the cutoff frequency and T_n is a Chebyshev polynomial of the nth order.

The Chebyshev polynomials type I:

$T_0(x) = 1$

$T_1(x) = x$

$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$

The Chebyshev polynomials type I:

$U_0(x) = 1$

$U_1(x) = 2x$

$U_{n+1}(x) = 2xU_n(x) - U_{n-1}(x)$

Wiener filter: Introduction

How about designing an *optimal filter* using knowledge of the signal and noise spectral (or autocorrelation) characteristics?

Pázmány Péter Catholic University
Faculty of Information Technology and Bionics

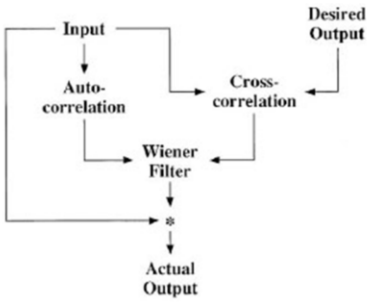
Wiener filter

The Wiener filter statistically minimizes the mean square error between the observed process (y) containing noise (n) and the unknown desired process (x).

$$y = x + n$$
$$S_x = \mathcal{F}(R_x(\tau))$$
$$S_n = \mathcal{F}(R_n(\tau))$$

The filter:

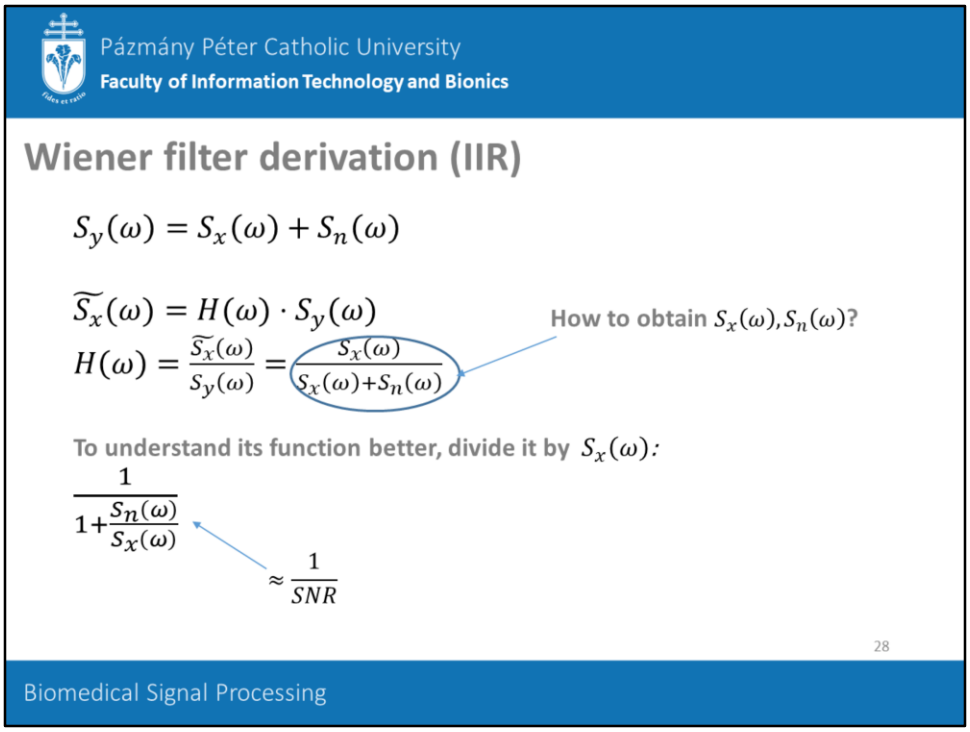
$$\boldsymbol{H(\omega) = \frac{Sx}{Sx+Sn}}$$



Biomedical Signal Processing

S_n and S_y is the power spectral density (PSD) of the related signal. It is calculated as the Fourier transform of the (auto)correlation functions. The autocorrelation $R_x(\tau)$ is $E[x(t)x(t+\tau)]$.

The Wiener filter is linear. It is assumed, that the signal is stationary (its statistical parameters are not changing with time) and the noise is independent, additive.



In line 1: orthogonality (independence) between the signal and noise is assumed
In line 2 we assume that the noise can be filtered out from the observed signal by $H(\omega)$ *to obtain the original one.*
In line 3 the power spectrum of the estimated signal $\widetilde{S_x}(\omega)$ *is assumed to equal the real original signal* $S_x(\omega)$*, again because of signal orthogonality.*
In line 4 $\frac{S_n(\omega)}{S_x(\omega)}$ *is the inverse of the SNR: with high SNR (low noise), the filter will give back the observed signal y, with weak signals the filter will go to 0.*

Obtaining $S_x(\omega)$ can be problematic. The signal can be assumed to have a parametric shape, like exponential or Gaussian (method#0). It can also be estimated by averaging similar signals(method#1), or a mathematical model (method#2, last lecture) of the signal.
$S_n(\omega)$ for white noise is its variance, square of standard deviation. But if we do not know it, $S_y(\omega)$ can be directly calculated from the observation, and $S_n(\omega) = S_y(\omega) - S_x(\omega)$

for details, see *Sörnmo and Laguna (2005): Bioelectric Signal Processing, p 245-250*

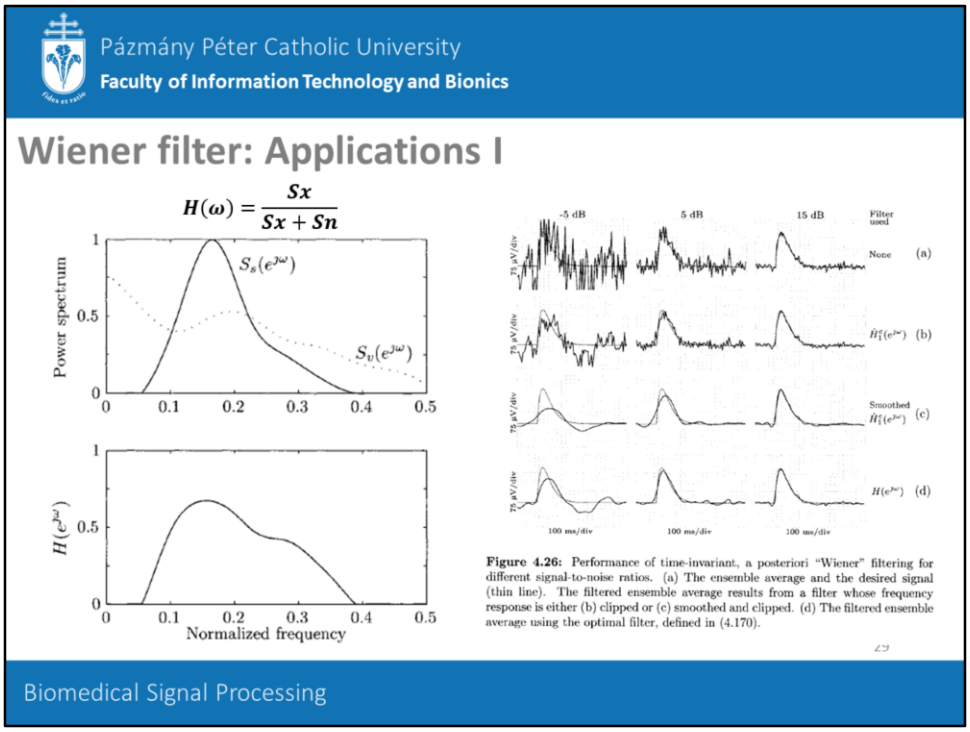


Figure 4.26: Performance of time-invariant, a posteriori "Wiener" filtering for different signal-to-noise ratios. (a) The ensemble average and the desired signal (thin line). The filtered ensemble average results from a filter whose frequency response is either (b) clipped or (c) smoothed and clipped. (d) The filtered ensemble average using the optimal filter, defined in (4.170).

The first plot shows the power spectrum of the signal and noise, and the ideal wiener filter calculated from them.

On the right hand side it can be seen, how the filter acts under different SNR values (columns).
The thin line represents the ideal signal. Here it is estimated as the average of the ensemble (method#1).
In the first row the original observation y is plotted. The second uses a LP filter, the third a smoothed LP filter. The last row is the ideal Wiener filter.

Note that here we are talking about power spectra – some textbooks may use $S_y^2(\omega), S_x^2(\omega), S_n^2(\omega)$ where Sx etc refers to magnitude spectra

Wiener filter: Applications II

Another example…

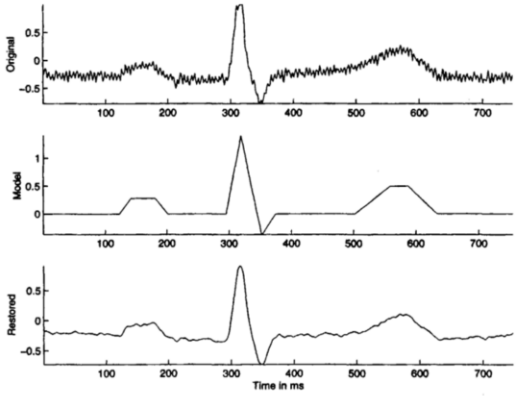
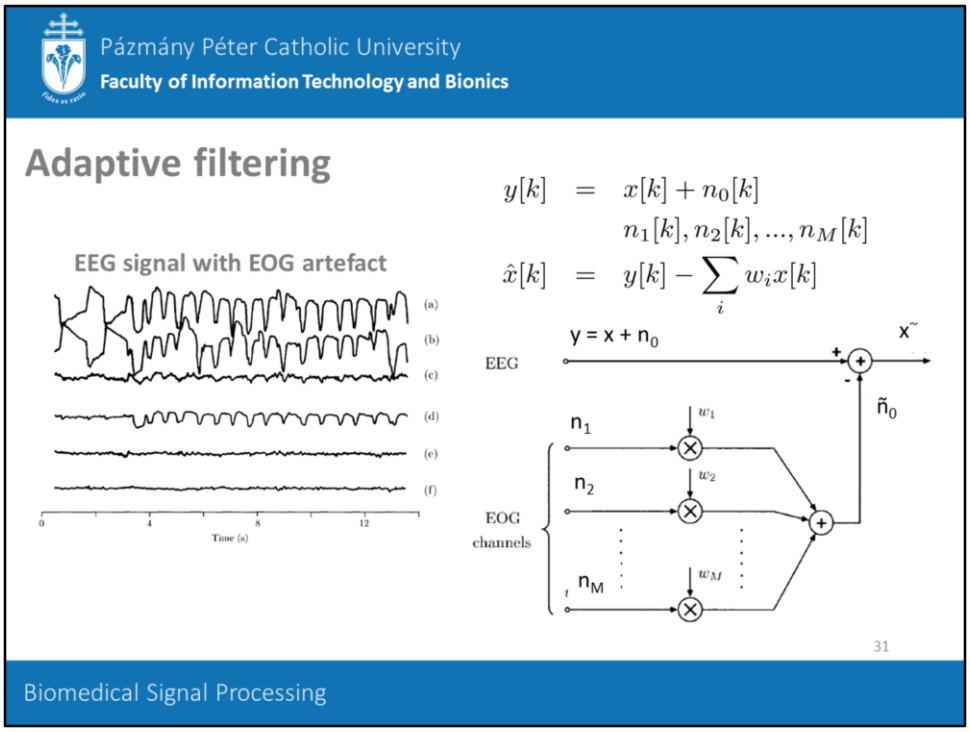


Figure 3.47 From top to bottom: one cycle of the noisy ECG signal in Figure 3.5 (labeled as Original); a piece-wise linear model of the desired noise-free signal (Model); and the output of the Wiener filter (Restored).

In this case the signal power spectrum is estimated from a mathematical model of the ECG (method#2).



The signal is **not** always **stationary**, when previous filters are not efficient. The frequency, mean value, variance, etc. can change with time, when new methods are necessary. A solution would be to clip the signal dependng on these properties, and filter the segments separatedly. But it is not always efficient.

We can try to remove noise based on reference signal.
Adaptive: signal-dependent; **adapt with time**
Examples: EOG interference in EEG (eye muscle movements) or maternal interference in fetal ECG.

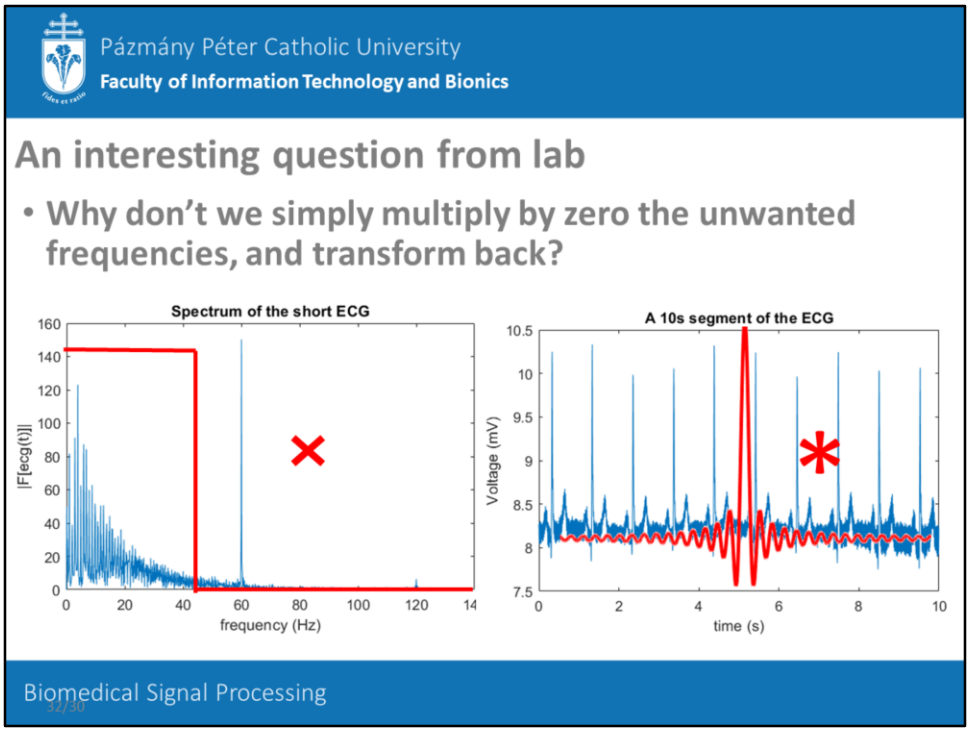
In the image an EEG signal with EOG artefact can be observed. The first two lines are the EMG recording from the eye movement. The second two rows are EEG signals with the artefact present. The last two rows are the filtered timelines.

In the system model the observation y is composed of x, the EEG signal with some noise n0. This n0 is the EOG noise – in this case only n1 and n2, the two channels. With some weight we would like to substract these EOG recording from the signal. These weights can change with time, as in each instant (k) EOG can corrut the EEG differently.
How are weights w_1,…,w_M updated?

Idea: minimize norm of the estimated signal $\tilde{x}$.

Assume: n1,..nM are correlated with n0 (as it is composed of the eye movements), and x is uncorrelated with n0,n1,..nM.
LMS (least mean square, to have minimum norm of $\tilde{x}$) filter:
$w_{l,k+1} = w_{l,k} + 2\mu\, \tilde{x}_k \cdot n_{l,k}$, where μ is the convergence factor (https://en.wikipedia.org/wiki/Adaptive_filter)

Concept: very simple! If error positive, subtract more from signal, in proportion to previous reference data point and error signal data point. Likewise if error is negative. If correct weight is not changing, eventually, applied weight will “lock onto” correct weight. If slowly changing, it will follow it.



We multiply by a rectangular window in the frequency domain == convolution with a sinc function in the time domain

Summary on Classmarker ☺

THE END

References

ECG noise (5): SL p. 442, R p. 88

Techniques (6): R Ch 3: Filtering for Removal of Artefacts

Synchronised averaging I (7): R p. 98

Synchronised averaging II (8): R p. 94

http://www.canvashome.com/circus-tents.html

The z-transform (11): http://en.wikipedia.org/wiki/Z-transform

Relation to discrete-time Fourier transform (12): http://en.wikipedia.org/wiki/Discrete-time_Fourier_transform; note the slight difference in definitions owing to wikipedia's implicit assumption that fs = 1

Filters in the z-dimain (13): Matlab, doc filter

Moving-average filters (16-17): R p. 99-104

Filter classes (20): http://en.wikipedia.org/wiki/File:Electronic_linear_filters.svg

Butterworth filter (21): R p. 118-119; http://en.wikipedia.org/wiki/Butterworth_filter

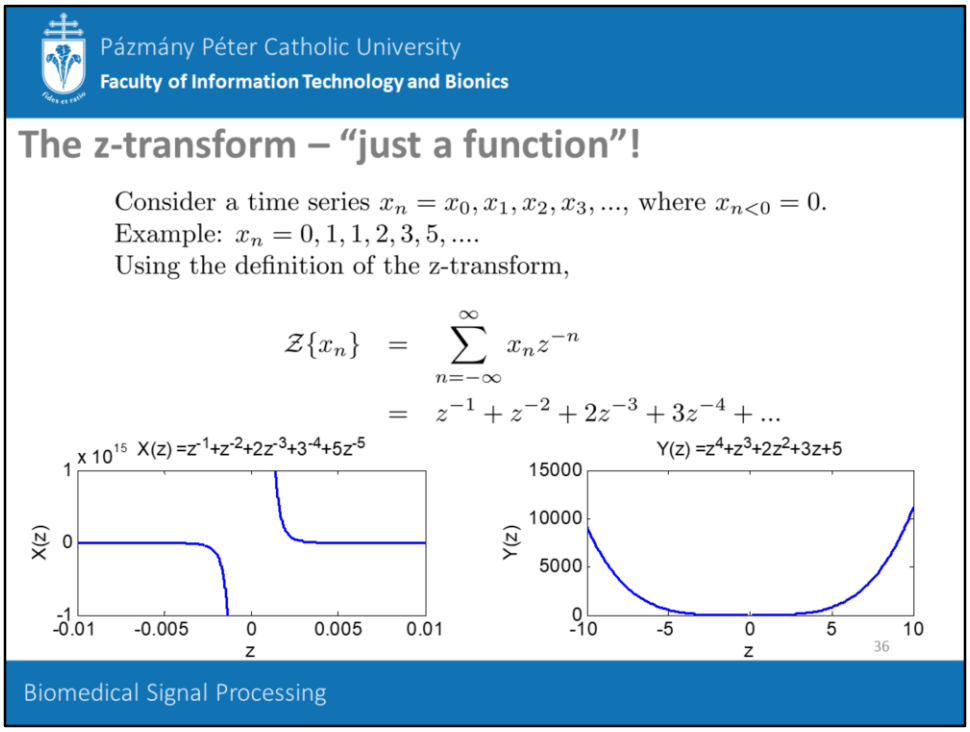
Chebyshev filter (Type 1) (22): http://en.wikipedia.org/wiki/Chebyshev_filter

Wiener filter: Introduction (23): SL p. 245-247

Wiener filter derivation (IIR) (24): SL p. 245-247; R p. 137-144

Wiener filter derivation (FIR) (25): SL p. 247, 250

Wiener filter Applications II (26): R p. 144



generating functions, first introduced by de Moivre.
Take (from Wolfram's Mathworld) example of generating function for Fibonacci numbers, f(x) = x/(1-x-x^2) = x + x^2 + 2x^3 + 3x^4
So in other words, a generating function has the property that its polynomial coefficients gives a number series that is of interest to us.