Aalto University

T61.3040

Adaptive filtering

Amaury Lendasse and
Yoan Miche

Diagram of content of the final part of the course

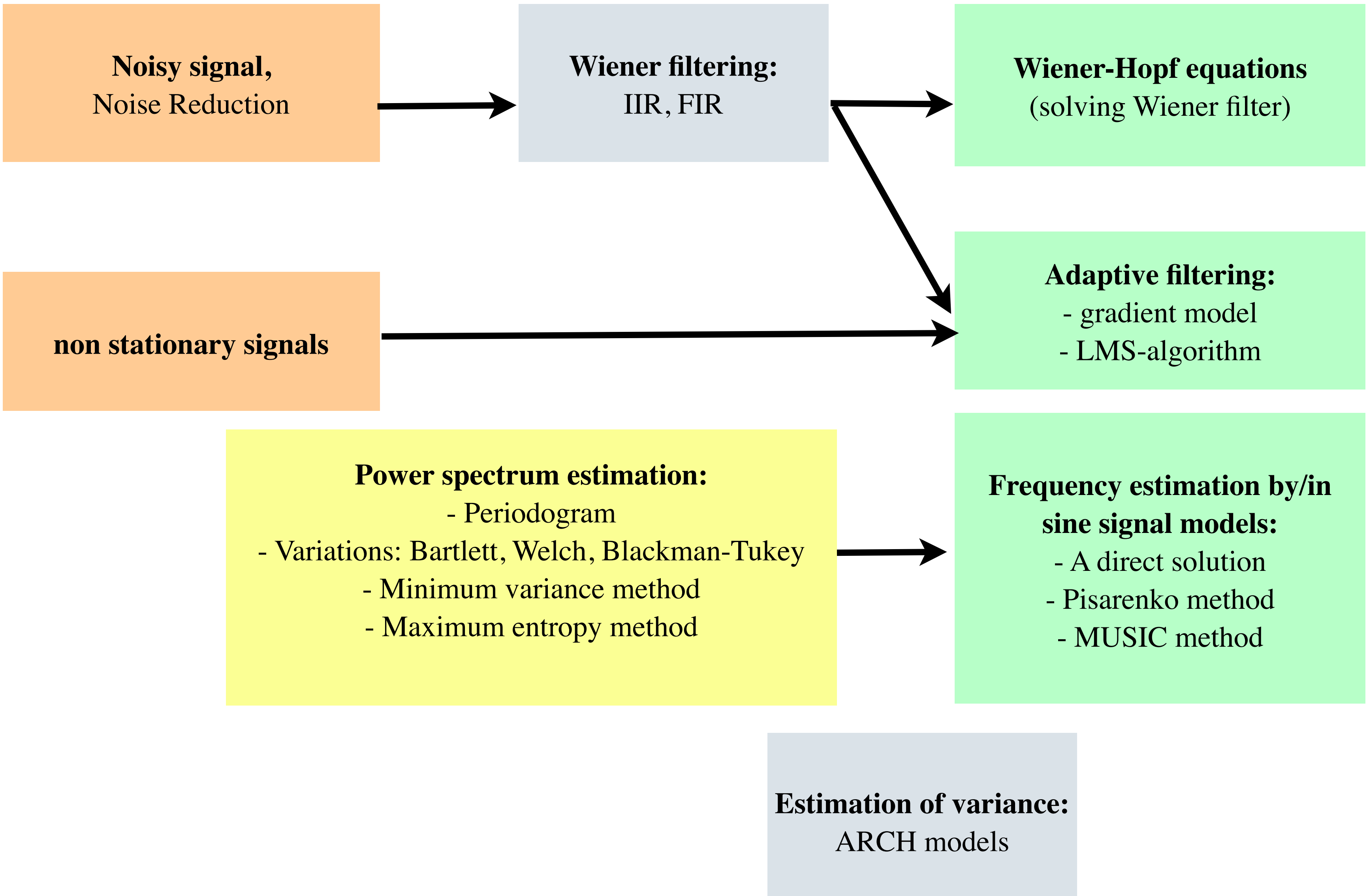

Diagram of content of the final part of the course

non stationary signals $\longrightarrow$ **Adaptive filtering:**
- gradient model
- LMS-algorithm

Estimation of variance:
ARCH models

- Methods discussed so far require that the process is stationary (WSS) and ergodic
- Estimation can then be performed by averaging over time and using all observations $x(0)$, $x(1)$,... , $x(N-1)$
- In many applications, stationarity can not always be assumed

- Examples:
- Speech signal: for example, the frequency and/or intensity of a vowel can change during the pronunciation
- For a signal measured from an electric motor, the frequency changes if the engine speed changes

- The default "process is non-stationary" is by itself far too weak
- We need more accurate assumptions, which are usually application-specific
- "Almost WSS": statistical properties change slowly over time
- Simple solution: look at the process in pieces and assume each piece is a WSS process

- Even if piecewise estimation were successful, estimated values change abruptly
- In many applications this may be an unwanted property
- However, if the parameters of interest do not change over time, then cutting the process may be sufficient

- Example: $x(n) = -a(1,n)\, x(n-1) - a(2)\, x(n-2) + v(n)$
- $a(1,n)$ is a function of time, but $a(2)$ is constant
- Previous methods do not work, because $a(1)$ does not remain constant
- Piecewise modeling provides several estimates of $a(2)$, for which the average can be calculated
- demo: nonwss.R

- In general, we are interested in all the parameters
- In addition, often you want to track slow changes smoothly so cutting is not a satisfactory solution
- Consider, therefore, how to deal with processes which are assumed to be non stationary, but changes are slow

- Stationary Wiener filter

$$\hat{d}(n) = \sum_{k=0}^{p} w(k)x(n-k)$$

- Solved by the Wiener-Hopf equations

$$R_x w = r_{dx}$$

- (Note: this filter has p+1 coefficients, while in lecture 8 the filter had p coefficients. This is only a difference in notation.)
- $x(n)$ and $d(n)$ were assumed to be jointly WSS

- From the WSS assumption, it follows that the parameters $d(n)$, $x(n)$, $x(n-1)$,... , $x(n-p)$ correlations do not depend on the time n: then the solution neither depends on the time n
- Get rid of the WSS assumption: the solution may change when n changes
- In theory, we can solve the wiener filter for every moment n

- Let's denote the dependence on *n* explicitly:

$$\hat{d}(n) = w_n^T x_n$$

- where

$$w_n = [w_n(0), w_n(1), \ldots, w_n(p)]^T$$

$$x_n = [x(n), x(n-1), \ldots, x(n-p)]^T$$

- WH-solution $w_n = R_x^{-1} r_{dx}$ is the same as previously, but it is separately defined for each *n*
- The solution is not feasible to calculate, because time averages can not be used

- We try to solve the Wiener filter adaptively $w_{n+1} = w_n + \Delta w_n$, where w_n is a solution at the time *n*
- We require that
  - When the process is WSS, then we must have

$$\lim_{n \to \infty} w_n = R_x^{-1} r_{dx}$$

  - Δw_n determined only from observations
  - We can monitor the changes caused by the nonstationarity

- Minimize the error

$$s(n) = \mathrm{E}(|e(n)|^2),\ e(n) = d(n) - \hat{d}(n)$$

- As in the case of WSS, we get WH equations

$$R_x(n)w_n = r_{dx}(n)$$

- Where all variables depend on the time *n*
- Autocorrelation matrix $R_x(n)$ with elements

$$R_x(n)_{ij} = \mathrm{E}[x(n-j+1)x^*(n-i+1)]$$

Gradient method

- We solve the correction term Δw_n via error gradient
- Assume that w_n is a solution at the time n
- We calculate w_{n+1} by adding one term to w_n, which reduces the error $s(n) = \mathrm{E}(|e(n)|^2)$
- The error gradient

$$\nabla s(n) = \left[\frac{\partial}{\partial w_n(0)} s(n), \quad \frac{\partial}{\partial w_n(1)} s(n), \quad \ldots, \quad \frac{\partial}{\partial w_n(p)} s(n)\right]^T$$

shows the direction where the error for which the error is growing the fastest

Gradient method

- We move the vector w_n in the opposite direction of the gradient, i.e. in the direction where the error decreases most rapidly:

$$w_{n+1} = w_n - \mu \nabla s(n)$$

- The positive step length μ determines how far to go to the negative gradient direction

Gradient method

- Summary of the gradient method:
  1. select the initial value of w_n and step length $\mu > 0$
  2. calculate the gradient $\nabla s(n)$ using the vector w_n
  3. Compute the next vector:

$$w_{n+1} = w_n - \mu \nabla s(n)$$

  4. increase n by one and go to step 2

Gradient method

- The gradient can be calculated by differentiating with respect to the complex conjugate of *w*

$$
\begin{aligned}
\nabla s(n) &= \nabla\,\mathrm{E}(|\,d(n) - w_n^T x_n\,|^2) \\
&= \mathrm{E}(\nabla[|\,d(n) - w_n^T x_n\,|^2]) \\
&= \mathrm{E}(e(n)\nabla[(d(n) - w_n^T x_n)^*]) \\
&= -\mathrm{E}(e(n)x_n^*)
\end{aligned}
$$

- By substitution, we obtain the update rule

$$
w_{n+1} = w_n + \mu\,\mathrm{E}(e(n)x_n^*)
$$

Gradient method equilibrium point

- Assuming WSS, we obtain

$$-\nabla s(n) = \mathrm{E}(e(n)x_n^*) = r_{dx} - R_x w_n$$

- and

$$w_{n+1} = w_n + \mu(r_{dx} - R_x w_n)$$

- If $w_n = R_x^{-1} r_{dx}$, i.e. WH-solution, then $w_{n+1} = w_n$

- So we remain at the solution, if we can get there...

Gradient method convergence

- Subject to certain conditions, the algorithm really reaches the Wiener-Hopf solution:
- If the step length satisfies

$$0 < \mu < \frac{2}{\lambda_{max}}$$

- Where λ_{max} is the largest eigenvalue of the autocorrelation matrix R_x, then

$$\lim_{n \to \infty} w_n = R_x^{-1} r_{dx}$$

Gradient method convergence

- So, for sufficiently small step size μ, we reach the Wiener-Hopf solution
- The result is true only when $x(n)$ and $d(n)$ are jointly WSS
- For a non stationary process, convergence can not be demonstrated

Gradient method convergence

- Significance of the gradient method is that it can be demonstrated to converge
- Correction term $\Delta w_n = \mu \, \mathrm{E}(e(n)x_n^*)$ is however not possible to calculate from the observations in the nonstationary case
- Previously we required that the adaptive filter correction term can be calculated from the observations
- We replace the expectation by its estimate, which is formed as the time average of the values already detected

Gradient method convergence

- By selecting *L* observations, we can estimate

$$\mathrm{E}(e(n)x_n^*) = \frac{1}{L}\sum_{l=0}^{L-1} e(n-l)x_{n-l}^*$$

- By substituting in to the gradient method we obtain

$$w_{n+1} = w_n + \frac{\mu}{L}\sum_{l=0}^{L-1} e(n-l)x_{n-l}^*$$

LMS algorithm

- When $L = 1$, we get the LMS algorithm:

$$w_{n+1} = w_n + \mu e(n) x_n^*$$

- For each component separately we get

$$w_{n+1}(k) = w_n(k) + \mu e(n) x^*(n-k)$$

- Which shows the simplicity of the LMS algorithm. Each iteration requires only the following calculations: for calculating the scalar $\mu e(n) = \mu(d(n) - w_n^T x_n)$ we need p+1 multiplications, p+1 additions and one multiplication by the constant μ.

LMS algorithm

- The difference with the gradient method:

$$\nabla s(n) = -\mathrm{E}(e(n)x_n^*)$$

$$\hat{\nabla} s(n) = -e(n)x_n^*$$

- LMS does not always proceed in the right direction: on the other hand

$$\mathrm{E}(\hat{\nabla} s(n)) = -\mathrm{E}(e(n)x_n^*) = \nabla s(n)$$

- so on average LMS is progressing in the direction of negative gradient

LMS algorithm

- Consider how the LMS algorithm progresses for different coefficients μ
- Set the initial values for the parameters as zeros
- The correct solution is $w(1) = 1.5$ and $w(2) = -0.6$
- Compare the step lengths $\mu1 = 0.002$ and $\mu2 = 0.01$
- Demos: lms1.R and lms2.R

LMS algorithm

- Judging by demos, the LMS algorithm converges towards the correct values
- Convergence rate seems to depend on the length of the step
- After convergence, with different step sizes, the LMS algorithm "wobbles" more or less around the true value

LMS algorithm

- In the LMS algorithm w_n is a random vector (depending on $e(n)$ and the process $x(n)$)
- Assume that $x(n)$ and $d(n)$ are jointly WSS
- We want to know the conditions under which the LMS algorithm on average converges towards of the Wiener-Hopf solution

$$w = R_x^{-1} r_{dx}$$

- i.e. when

$$\lim_{n \to \infty} \mathrm{E}(w_n) = w$$

LMS algorithm

- Taking the expectation of the update rule:

$$
\begin{aligned}
\mathrm{E}(w_{n+1}) &= \mathrm{E}(w_n + \mu e(n) x_n^*) \\
&= \mathrm{E}(w_n) + \mu\,\mathrm{E}\left(d(n) x_n^*\right) - \mu\,\mathrm{E}\left(x_n^* x_n^T w_n\right) \\
&= \mathrm{E}(w_n) + \mu r_{dx} - \mu\,\mathrm{E}\left(x_n^* x_n^T w_n\right)
\end{aligned}
$$

- Assume that w_n and x_n are independent, then

$$\mathrm{E}\left(x_n^* x_n^T w_n\right) = R_x\,\mathrm{E}(w_n)$$

LMS algorithm

- We get

$$\mathrm{E}(w_{n+1}) = \mathrm{E}(w_n) + \mu(r_{dx} - R_x \mathrm{E}(w_n))$$

- This is now the gradient method for the vector $\mathrm{E}(w_n)$
- Then we get the LMS algorithm convergence result: with WSS and independence assumptions, the LMS algorithm converges in expectation, $0 < \mu < 2/\lambda_{max}$
- Convergence in expectation means that $\mathrm{E}(w_n)$ converges towards the correct value of w

LMS algorithm

- The convergence result is difficult because the largest eigenvalue λ_{max} of the matrix R_x should be calculated
- Replace λ_{max} with a larger value, then the coefficient μ is certainly between the required values
- Matrix trace is easy to calculate: it can be shown

$$\mathrm{tr}(R_x) = \sum_i \lambda_i \geq \lambda_{max}$$

- eigenvalues therefore not required to be calculated.
- Replace λ_{max} with the trace of R_x

LMS algorithm

- For a WSS process $x(n)$, R_x is a Toeplitz matrix and its trace is then

$$(p+1)r_x(0) = (p+1)\,\mathrm{E}(|x(n)|^2)$$

- We get

$$0 < \mu < \frac{2}{(p+1)\,\mathrm{E}(|x(n)|^2)}$$

- • the variance $\mathrm{E}(|x(n)|^2)$ is much simpler to estimate than λ_{max}

Aalto University

T61.3040

Variance prediction

Amaury Lendasse and
Yoan Miche

Variance prediction

- Let's return to the WSS process:

  1. expectation is time-independent constant $m_x = \mathrm{E}(x(n))$

  2. autocorrelations are independent, i.e. can be written in the form $r_x(k)$

  3. variance is finite: $c_x(0) < \infty$

Variance prediction

- WSS assumptions concern the expectations $E(x(n))$ and $E(x(n)\, x^*(n))$
- An important application of modeling is the prediction of future values from the observed values
- Then we need conditional statistics, such as $E(x(n) \mid x(n-1), x(n-2), \ldots)$
- Conditionality means that the expected value is calculated while assuming that the values of the variables to the right of the vertical line are known

Variance prediction

- We previously saw that the conditional expectation of an ARMA process depends on the observed values
- This was used to predict future values
- But for a normally distributed ARMA process, the conditional variance is constant, i.e. the variance of the prediction is always the same regardless of the values observed

Variance prediction

$$\text{AR(1)}: \; x(n) = -ax(n-1) + b(0)v(n), \quad v(n) \sim N(0,1)$$

- Conditional statistics:

$$\mathrm{E}(x(n)|x(n-1)) = -ax(n-1)$$

$$\mathrm{var}(x(n)|x(n-1)) = b^2(0)$$

$$\mathrm{E}(x(n)) = 0$$

$$\mathrm{var}(x(n)) = b^2(0)/(1-a^2)$$

- We see in particular that the conditional variance does not depend on the observations!

Variance prediction

- The conditional variance is thus a constant: is this a feature of WSS processes, or only of the normally distributed ARMA processes?
- Example: define a process

$$x(n) \sim \begin{cases} N(0,1), & x(n-1) \geq 0 \\ N(0,2), & x(n-1) < 0 \end{cases}$$

Variance prediction

- Properties (in the exercises):

$$\begin{aligned}
\mathrm{E}(x(n)) &= 0 \\
\mathrm{E}(x(n)|x(n-1)) &= 0 \\
\mathrm{var}(x(n)) &= 1.5 \\
\mathrm{var}(x(n)|x(n-1)) &= 0.5 * (-\operatorname{sgn}(x(n-1)) + 3)
\end{aligned}$$

- $x(n)$ is WSS

Variance prediction

- It is therefore possible that for a WSS process, the conditional variance depends on the observations

- In the previous example, the process is white noise, but its variance can be predicted

ARCH model

- In the ARCH model, the conditional variance is dependent on the observations
- The variance is modeled parametrically on the previous values
- ARCH process is obtained by multiplying the white noise $v(n)$ by the time-dependent standard deviation $[h(n)]^{1/2}$
- The variance $h(n)$ is defined as a function of the observations $x(n-1)$, $x(n-2)$,. . .

ARCH model

- ARCH(1)-model $$x(n) = v(n)[h(n)]^{1/2}$$ $$h(n) = a(0) + a(1)x^2(n-1), \quad a(0) > 0, a(1) \geq 0$$
- Noise $v(n)$ values are independent and identically distributed, with expected value of zero and variance 1
- Let's calculate the conditional expectation and variance:

$$\mathrm{E}(x(n)|x(n-1)) = 0$$

$$\mathrm{var}(x(n)|x(n-1)) = h(n)$$

ARCH model

- The conditional variance is thus directly $h(n)$ which is not independent of the observations
- Because $h(n) = a(0) + a(1)x^2(n-1)$ then a high value of $x^2(n\text{-}1)$ causes a large variance at time n, and vice versa
- In financial applications, they talk about volatility clustering (the process gets large values in some time range(s), and small in others)

ARCH model

- Let's see how the ARCH process behaves
- Simulate the above process $x(n)$ with positive coefficients a(0) = 0.03, a(1) = 1.0
- Demo: archex.R, order $q = 1$
- Demo: archexp.R, order $q = 10$,
- a (0) = 0.01, a (1) = ... = a (10) = 0.1

ARCH model

- More generally, we can define an ARCH(q) process

$$x(n) = v(n)[h(n)]^{1/2}$$

$$h(n) = a(0) + \sum_{k=1}^{q} a(k)x^2(n-k)$$

$$a(0) > 0, a(1) \geq 0, \ldots, a(q) \geq 0$$

- Noise $v(n)$ defined as in the ARCH(1)-model
- Noise assumption means that the conditional statistics are determined only by $h(n)$

ARCH model

- Interpretation: If you know the previous findings, the observation $x(n)$ depends only on the value of $h(n)$ calculated from them (and of course the value of the noise $v(n)$)
- The process $h(n)$ is a variance process, because it determines the conditional variance of the ARCH(q) process at time n
- If the parameters $a(k)$ are known, then the variance of $h(n)$ can be calculated from the observations at time n-1
- The ARCH model therefore allows prediction of the variance

ARCH model

- The ARCH model can be generalized in several ways
- The variance function $h(n)$ may depend on the observations in other ways than the ones presented above
- The modeling of the conditional expectation can be included in the model (now the expectation is zero)