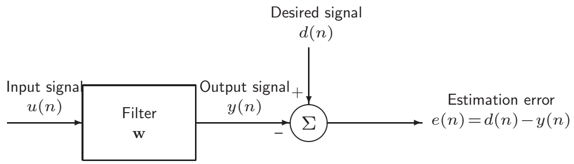
Lecture 1

This lecture deals with

- Repetition of the course Optimal signal processing (OSB)
- The method of the Steepest descent

Optimal Linear Filtrering



The filter $\mathbf{w} = [w_0 \quad w_1 \quad w_2 \ldots]^T$ which minimizes the estimation error $e(n)$, such that the output signal $y(n)$ resembles the desired signal $d(n)$ as much as possible is searched for.

Optimal Linear Filtrering

In order to determine the optimal filter a cost function J, which punish the deviation $e(n)$, is introduced. The larger $e(n)$, the higher cost.

From OSB you know some different strategies, e.g.,

- **The total squared error (LS)** Deterministic description of the signal.

$$J = \sum_{n_1}^{n_2} e^2(n)$$

- **Mean squared error (MS)** Stochastic description of the signal.

$$J = E\{|e(n)|^2\}$$

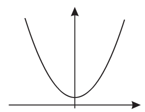
- **Mean squared error with extra contraint**

$$J = E\{|e(n)|^2\} + \lambda |u(n)|^2$$

Optimal Linear Filtrering

The cost function $J(n) = E\{|e(n)|^p\}$ can be used for any $p \geq 1$, but most oftenly for $p = 2$. This choice gives a convex cost function which is refered to as the *Mean Squared Error*.

$$J = E\{e(n)e^*(n)\} = E\{|e(n)|^2\} \quad MSE$$



The gradient operator yields

$$\nabla J(\mathbf{w}) = 2\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}^*} = 2\frac{\partial}{\partial \mathbf{w}^*}(\sigma_d^2 - \mathbf{w}^H\mathbf{p} - \mathbf{p}^H\mathbf{w} + \mathbf{w}^H\mathbf{R}\mathbf{w})$$
$$= -2\mathbf{p} + 2\mathbf{R}\mathbf{w}$$

If the gradient vector is set to zero, the Wiener-Hopf equation system results

$$\mathbf{R}\mathbf{w}_o = \mathbf{p} \quad \text{Wiener-Hopf}$$

which solution is the Wiener filter.

$$\mathbf{w}_o = \mathbf{R}^{-1}\mathbf{p} \quad \text{Wienerfilter}$$

In other words, the Wiener filter is optimal when the cost is controlled by *MSE*.

The cost function's dependence on the filter coefficients $\mathbf{w}$ can be made clear if written in canonical form

$$J(\mathbf{w}) = \sigma_d^2 - \mathbf{w}^H\mathbf{p} - \mathbf{p}^H\mathbf{w} + \mathbf{w}^H\mathbf{R}\mathbf{w}$$
$$= \boxed{\sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} + (\mathbf{w}-\mathbf{w}_o)^H\mathbf{R}(\mathbf{w}-\mathbf{w}_o)}$$

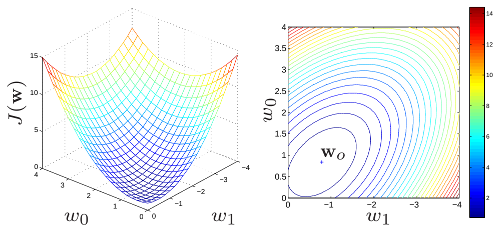
Here, Wiener-Hopf and the expression of the Wienerfilter have been used in addition to the fact that the following decomposition can be made

$$\mathbf{w}^H\mathbf{R}\mathbf{w} = (\mathbf{w}-\mathbf{w}_o)^H\mathbf{R}(\mathbf{w}-\mathbf{w}_o) - \mathbf{w}_o^H\mathbf{R}\mathbf{w}_o + \mathbf{w}_o^H\mathbf{R}\mathbf{w} + \mathbf{w}^H\mathbf{R}\mathbf{w}_o$$

With the optimal filter $\mathbf{w} = \mathbf{w}_o$ the minimal error J_{min} is achieved:

$$J_{min} \equiv J(\mathbf{w}_o) = \sigma_d^2 - \mathbf{p}^H\mathbf{R}^{-1}\mathbf{p} \quad MMSE$$

Error-Performance Surface för FIR-filter with two coefficients, $\mathbf{w} = [w_0,\ w_1]^T$



$$\mathbf{p} = [0.5272 \quad -0.4458]^T \qquad \mathbf{R} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \quad \sigma_d^2 = 0.9486$$

$$\mathbf{w}_o = [0.8360 \quad -0.7853]^T \quad J_{min} = 0.1579$$

The method of the Steepest descent is a recursive method that leads to the Wiener-Hopfs equations. The statistics are known ($\mathbf{R},\ \mathbf{p}$). The purpose is to avoid inversion of $\mathbf{R}$. (saves computations)

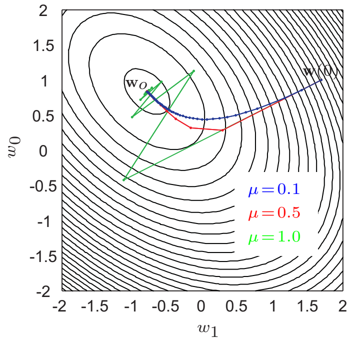
- Set start values for the filter coefficients, $\mathbf{w}(0)$ $(n=0)$
- Determine the gradient $\nabla J(n)$ that points in the direction in which the cost function increases the most. $\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$
- Adjust $\mathbf{w}(n+1)$ in the opposite direction to the gradient, but weight down the adjustment with the stepsize parameter μ

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)]$$

- Repete steps 2 and 3.

Convergence, error surface

The influence of the stepsize parameter on the convergence can be seen when analyzing $J(\mathbf{w})$. The example below illustrates the convergence towards J_{min} for different choices of μ.



$$\mathbf{p} = \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$$

$$\mathbf{w}_o = \begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix}$$

$$\mathbf{w}(0) = \begin{bmatrix} 1 \\ 1.7 \end{bmatrix}$$

Convergence analysis

With the observation that $\mathbf{w}(n) = \mathbf{Q}\boldsymbol{\nu}(n) + \mathbf{w}_o$ the update of the cost function can be derived:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu[\mathbf{p} - \mathbf{R}\mathbf{w}(n)]$$

$$\mathbf{Q}\boldsymbol{\nu}(n+1) + \mathbf{w}_o = \mathbf{Q}\boldsymbol{\nu}(n) + \mathbf{w}_o + \mu[\mathbf{p} - \mathbf{R}\mathbf{Q}\boldsymbol{\nu}(n) - \mathbf{R}\mathbf{w}_o]$$

$$\boldsymbol{\nu}(n+1) = \boldsymbol{\nu}(n) - \mu\mathbf{Q}^H\mathbf{R}\mathbf{Q}\boldsymbol{\nu}(n) = (\mathbf{I} - \mu\boldsymbol{\Lambda})\boldsymbol{\nu}(n)$$

$$\nu_k(n+1) = (1 - \mu\lambda_k)\nu_k(n) \qquad (\text{ Element } k \text{ i } \boldsymbol{\nu}(n) \text{ })$$

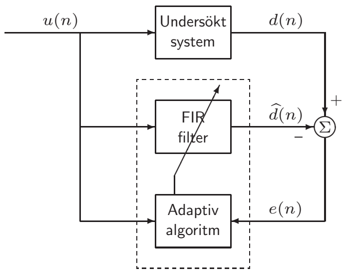
The latter is a 1:st order difference equation, with the solution

$$\nu_k(n) = (1 - \mu\lambda_k)^n \nu_k(0)$$

For this equation to converge it is required that $|1 - \mu\lambda_k| < 1$, which leads to the stability criterion of the method of the *Steepest Descent*:

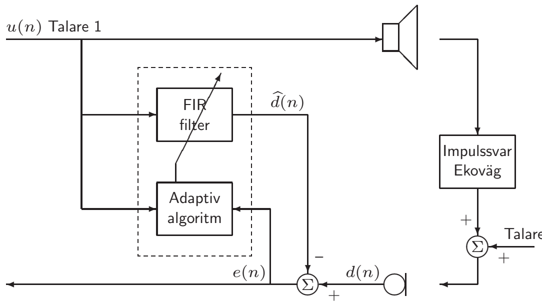
$$\boxed{0 < \mu < \frac{2}{\lambda_{max}}} \qquad \textit{Stabilitet, S.D.}$$

Exempel: Modellering/Identifiering



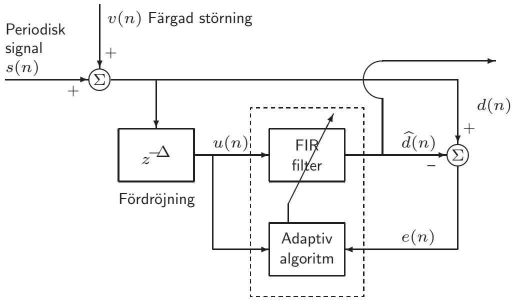
Vid modellering/identifiering kopplas det adaptiva filtret parallellt med det undersökta systemet. Om det undersökta systemet endast har nollställen, så är det lämpligt att använda motsvarande längd på filtret. Har systemet både poler och nollställen krävs i regel ett långt adaptivt FIR-filter.

Exempel: Ekosläckare II



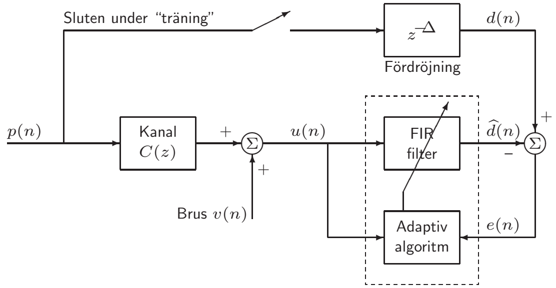
Denna struktur är tillämpbar på högtalartelefoner, videokonferenssystem och dylikt. Precis som vid telefonifallet hör Talare 1 sig själv i form av ett eko. Dock är denna effekt mer påtaglig här, eftersom mikrofonen fångar upp det som sänds ut av högtalaren. Adaptionen stoppas normalt när Talare 2 pratar, men filtreringen sker under hela tiden.

Exempel: Adaptive Line Enhancer



Information i form av en periodisk signal störs av ett färgat brus som är korrelerat med sig själv inom en viss tidsram. Genom att fördröja signalen så pass mycket att bruset (i $u(n)$ respektive $d(n)$) blir okorrelerat, kan bruset tryckas ned genom linjär prediktion.

Exempel: Kanalutjämnare (Equalizer)



En känd *pseudo noise*-sekvens används för att skatta en invers modell av kanalen ("träning"). Därefter stoppas adaptionen, men filtret fortsätter att verka på den översända signalen. Syftet är att ta bort kanalens inverkan på den översända signalen.

Lecture 2

During this lecture you will learn about

- The *Least Mean Squares* algorithm (LMS)
- Convergence analysis of the LMS
- Equalizer (Kanalutjämnare)

The Least Mean Square (LMS) algorithm

We want to create an algorithm that minimizes $E\{|e(n)|^2\}$, just like the SD, but based on unkown statistics.

A strategy that then can be used is to uses estimates of the autocorrelation matrix $\mathbf{R}$ and the cross correlationen vector $\mathbf{p}$. If instantaneous estimates are chosen,

$$\widehat{\mathbf{R}}(n) = \mathbf{u}(n)\mathbf{u}^H(n)$$

$$\widehat{\mathbf{p}}(n) = \mathbf{u}(n)d^*(n)$$

the resulting method is the *Least Mean Squares* algorithm.

The Least Mean Square (LMS) algorithm

For the SD, the update of the filter weights is given by

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{1}{2}\mu[-\nabla J(n)]$$

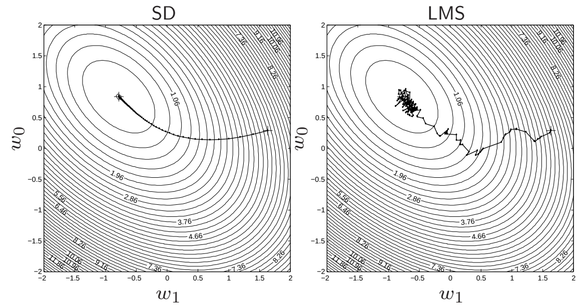
where $\boldsymbol{\nabla} J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n)$.

In the LMS we use the estimates $\widehat{\mathbf{R}}$ och $\widehat{\mathbf{p}}$ to calculate $\widehat{\boldsymbol{\nabla}} J(n)$. Thus, also the updated filter vector becomes an estimate. It is therefore denoted $\widehat{\mathbf{w}}(n)$;

$$\widehat{\boldsymbol{\nabla}} J(n) = -2\widehat{\mathbf{p}}(n) + 2\widehat{\mathbf{R}}(n)\widehat{\mathbf{w}}(n)$$

The Least Mean Square (LMS) algorithm

Illustration of *gradient noise*. $\mathbf{R}$ och $\mathbf{p}$ in the example from Lecture 1. $\mu = 0.1$.



Because of the noise in the gradientm the LMS never reaches J_{min}, which the SD does.

Properties of the LMS

- Convergence for the same interval as the SD:
$$0 < \mu < \frac{2}{\lambda_{max}}$$
- $J_{ex}(\infty) = J_{min} \sum_{i=1}^{M} \frac{\mu\lambda_i}{2 - \mu\lambda_i}$
- Misadjustment $\mathcal{M} = \frac{J_{ex}(\infty)}{J_{min}}$ is a measure of how close the the optimal solution the LMS (in mean-square sense).

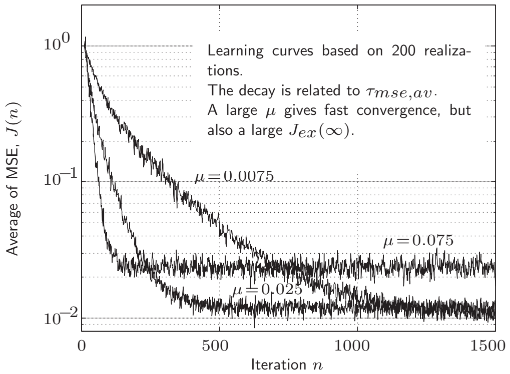
Rules of thumb LMS

The eigenvalues of $\mathbf{R}$ are rarely known. Therefore, a set of rules of thumbs is commonly used, that is based on the *tap-input power*, $\sum_{k=0}^{M-1} E\{|u(n-k)|^2\} = Mr(0) = \mathrm{tr}(\mathbf{R})$.

- The stepsize $0 < \mu < \frac{2}{\sum_{k=0}^{M-1} E\{|u(n-k)|^2\}}$
- Misadjustment $\mathcal{M} \approx \frac{\mu}{2} \sum_{k=0}^{M-1} E\{|u(n-k)|^2\}$
- Average time constant $\tau_{mse,av} \approx \frac{1}{2\mu\lambda_{av}}$ där $\lambda_{av} = \frac{1}{M}\sum_{i=1}^{M} \lambda_i$.
  When λ_i is unknown, the following relationship is useful.
- Approximate relationship $\mathcal{M}$ and $\tau_{mse,av}$ $\quad \mathcal{M} \approx \frac{M}{4\tau_{mse,av}}$

Here, $\tau_{mse,av}$ denotes the time it takes for $J(n)$ to decay a factor of e^{-1}.

Example, cont: Learning Curve



The curves illustrates learning curves for two different μ.

Lecture 3

Lecture 3 includes the following:

- Eigenvalue spread of $\mathbf{R}$ and its influence on the convergence speed for the LMS.
- Variants of the LMS:
  - The Normalized LMS
  - The Leaky LMS
  - The Sign LMS
- The Echo Canceller

LMS and eigenvalue spread

The convergence speed for the LMS depends on the spread of the eigenvalues of $\mathbf{R}$. This spread is normally written as the ratio between the largest and the smallest aigenvalues.

$$\chi(\mathbf{R}) = \frac{\lambda_{max}}{\lambda_{min}}$$

If $\chi(\mathbf{R}) > 1$, the contour plot of $J(n)$ for the 2-dimensional case $(M = 2)$ is elliptic.

The convergence is fastest along the short axis of the ellips since the cost function $J(n)$ has its fastes growth in this direction.

Along the long axis, the convergence is the slowest since the growth of $J(n)$ is minimal in this direction.

The Normalized LMS

For the standard LMS which we have looked at earlier, the step is proportial to $\mathbf{u}(n)$:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n) \ .$$

This means that the gradient noise is amplified when the signal is strong.

One solution to this problem is to normalize the update $\widehat{\mathbf{w}}(n)$ med $||\mathbf{u}(n)||^2 = \mathbf{u}^H(n)\mathbf{u}(n)$:

$$\widehat{\mathbf{w}}(n+1) = \widehat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a + ||\mathbf{u}(n)||^2}\mathbf{u}(n)e^*(n)$$

where $0 < \tilde{\mu} < 2$, and a is a positive protection constant.

The Leaky LMS

As for the standard LMS we start with the method of the Steepest descent,

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{2}[-\nabla J(n)] \ ,$$

but we now instead use the cost function

$$\begin{aligned} J(n) &= E\{|e(n)|^2\} + \alpha||\mathbf{w}(n)||^2 \\ &= E\{e^*(n)e(n)\} + \alpha\mathbf{w}^H(n)\mathbf{w}(n) \ . \end{aligned}$$

The gradient becomes

$$\nabla J(n) = -2\mathbf{p} + 2\mathbf{R}\mathbf{w}(n) + 2\alpha\mathbf{w}(n) \ .$$

Withe estimated statistics we get

$$\widehat{\nabla} J(n) = -2\widehat{\mathbf{p}}(n) + 2\widehat{\mathbf{R}}(n)\widehat{\mathbf{w}}(n) + 2\alpha\widehat{\mathbf{w}}(n) \ .$$

The Leaky LMS

The update equation for the Leaky LMS is given by

$$\widehat{\mathbf{w}}(n+1) = \underbrace{(1-\mu\alpha)}\widehat{\mathbf{w}}(n) + \mu\mathbf{u}(n)e^*(n) \ .$$

For the Leaky LMS it can be shown that

$$\lim_{n\to\infty} E\{\widehat{\mathbf{w}}(n)\} = (\mathbf{R} + \alpha\mathbf{I})^{-1}\mathbf{p}$$

i.e., it results in the same effect as if white noise with variance α was added at the input. This increases the uncertainty in the input signal, which hold the filter coefficients back.

The phenomenon is referred to as *prewhitening*, and α is denoted prewhitening parameter. The factor $(1-\mu\alpha)$ is called *Leakage* factor, where α is bounded by

$$0 \le \alpha < \frac{1}{N} \ .$$

The Sign LMS

Again, starting from the method of the Steepest descent

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{\mu}{2}[-\nabla J(n)] \ ,$$

and the cost function

$$J(n) = E\{|e(n)|\}$$

the gradient becomes

$$\widehat{\nabla} J(n) = -\mathrm{sign}[e(n)]\mathbf{u}(n) \ .$$

With this gradient, the update equation becomes

$$\widehat{\mathbf{w}}(n+1) = \widehat{\mathbf{w}}(n) + \alpha \cdot \mathrm{sign}[e(n)]\mathbf{u}(n) \ .$$

The Sign LMS is used in applications with extremely high requirements on computational complexity.

Lecture 4

Lecture 4 contains descriptions of

- Block-based LMS (7.1)
  (*edition 3:* 10.1)
- Frequency Domain LMS (FDAF, 7.2-7.3)
  (*edition 3:* 10.2-10.3)

Summary of the Block-LMS

1. $y(kL+i) = \widehat{\mathbf{w}}^T(k)\mathbf{u}(kL+i)$, L st
   i.e.,
   $y(k) = \widehat{\mathbf{w}}^T(k)\mathbf{u}(k)$.

2. $e(kL+i) = d(kL+i) - y(kL+i)$
   i.e.,
   $e(k) = d(k) - y(k)$.

3. $\widehat{\mathbf{w}}(k+1) = \widehat{\mathbf{w}}(k) + \mu \sum_{i=0}^{L-1} \mathbf{u}(kL+i)e(kL+i)$
   i.e.,
   $\widehat{\mathbf{w}}(k+1) = \widehat{\mathbf{w}}(k) + \mu \underset{\sim}{\mathbf{u}}(k)\mathbf{e}^T(k)$.

In applications with long filters, e.g. echo cancellation, the complexity becomes very high. The equations that take time are the filtering in (1) and the cross correlation in (3).

FDAF

In order to increase the calculation speed of the LMS algorithm, the filtering (convolution) and the gradient estimation (crosscorrelation) can be done in the frequency domain instead of the time domain.

Strategy:

1. FFT of input and error signals
2. Convolution and crosscorrelation corresponds to multiplication in the frequency domain
3. IFFT

Two advantages of the FDAF:

1. Faster calculation
2. Independent coefficients

Problem

Multiplication in the frequency domain corresponds to circular convolution, butr in order to maintain the properties of the LMS, linear convolution must be used.

- Filtering must be done with linear convolution.

- Gradient estimation should be done with linear convolution. Then the method is called Fast LMS. If linear convolution is not used here the method is called *Unconstrained FDAF*.

Properties of the Fast LMS, cont.

- The convergence speed for the Fast LMS can be optimized for each mode separately.
- The convergence speed for the i:th mode depends on $\mu\lambda_i$. A measure of λ_i is the average power in the frequency bin of the i:th mode,
  $P_i = |U_i|^2$.
  If $\mu_i = \frac{\alpha}{P_i}$, all modes will converge equally fast (WSS).
- If the imput signal is not WSS, P_i must be estimated recursively

$$P_i(k) = \gamma P_i(k-1) + (1-\gamma)|U_i(k)|^2$$

- The stepsize parameter μ is here substituted by a diagonal $2M \times 2M$ matrix $\boldsymbol{\mu} = \alpha\mathbf{D}(k)$, where $\mathbf{D}(k) = \text{diag}[P_0^{-1}, P_1^{-1}, \ldots, P_{2M-1}^{-1}]$.

Fast LMS, Update equations

$$\mathbf{U}(k) = \text{diag}(\text{FFT}[u((k-1)M)\ldots u(kM-1),\ u(kM)\ \ldots u((k+1)M-1)]^T)$$

$$\mathbf{y}(k) = \text{Last } M \text{ elements of } \text{IFFT}[\mathbf{U}(k)\widehat{\mathbf{W}}(k)]$$

$$\mathbf{d}(k) = \begin{bmatrix} d(kM) & d(kM+1) & \ldots & d((k+1)M-1) \end{bmatrix}^T$$

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{y}(k)$$

$$\mathbf{E}(k) = \text{FFT}\begin{bmatrix} \mathbf{0} \\ \mathbf{e}(k) \end{bmatrix}$$

$$\mathbf{P}(k) = \gamma\mathbf{P}(k-1) + (1-\gamma)\mathbf{U}^H(k)\mathbf{U}(k)$$

$$\mathbf{D}(k) = \mathbf{P}^{-1}(k) = \text{diag}\begin{bmatrix} P_0^{-1}(k) & P_1^{-1}(k) & \ldots & P_{2M-1}^{-1}(k) \end{bmatrix}$$

$$\boldsymbol{\phi}(k) = \text{First } M \text{ elements of } \text{IFFT}[\mathbf{D}(k)\mathbf{U}^H(k)\mathbf{E}(k)]$$

$$\widehat{\mathbf{W}}(k+1) = \widehat{\mathbf{W}}(k) + \alpha\text{FFT}\begin{bmatrix} \boldsymbol{\phi}(k) \\ \mathbf{0} \end{bmatrix}$$

Lecture 5

Lecture 5 presents

- Optimal LS filter
- Windowing of data
- Weighted LSfilter
- Recursive LS filter (RLS)

LS-filtret

The derivation of the normal equations for the LS filter is made in the same way as for the Wiener filter, but with the difference that the input signal now is known instead of that its statistics is known (Wiener filter).

We therefore choose to minimize the total error over a certain interval $[i_1, i_2]$,

$$\mathcal{E} = \sum_{i=i_1}^{i_2} |e(n)|^2$$

instead of the average error($E\{\bullet\}$).

LS-filtret

When minimizing the total error $\sum_{i=i_1}^{i_2} |e(n)|^2$ terms of the following type instead appears

$$\sum_{i=i_1}^{i_2} u(i-k)u^*(i-t) = \phi(t,k)$$

and

$$\sum_{i=i_1}^{i_2} u(i-k)d^*(i) = z(-k)\ .$$

$\phi(t,k)$ is called time-averaged autocorrelation and $z(-k)$ is called time-averaged crosscorrelation.

NNote that these are definitions and not estimates. In OSB the same notations were used both for ensamble-averaged and time-averaged correlation ($r_x(k)$ and $r_{dx}(k)$).

Time-averaged autocorrelation, matrix form

The corresponding time-averaged crosscorrelation vector (compare $\mathbf{p}$) is

$$\mathbf{z} = \begin{bmatrix} z(0) & z(1) & \ldots & z(-M+1) \end{bmatrix}^T$$

When minimizing $\mathcal{E} = \sum_{i=i_1}^{i_2} |e(i)|^2$ the normal equations for the LS filter will be:

$$\mathbf{\Phi}\widehat{\mathbf{w}} = \mathbf{z} \qquad \text{(jmf } \mathbf{R}\mathbf{w} = \mathbf{p}\text{)}$$

which solution is

$$\widehat{\mathbf{w}} = \mathbf{\Phi}^{-1}\mathbf{z} \qquad (= (\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H\mathbf{d}\ ) \ .$$

Weighted LS filter

The first step towards creating an adaptive LS filter is to introduce a weighting factor in the minimization criteria (prewindowing)

$$\mathcal{E}(n) = \sum_{i=1}^{n} \beta(n,i)|e(i)|^2 \ ,$$

where $\beta(n,i)$ determines how much the error for sample i shall be weighted into the cost functionat time n.

The weight must be in the interval

$$0 < \beta(n,i) \leq 1$$

and the most common choice of $\beta(n,i)$ is

$$\beta(n,i) = \lambda^{n-i} \ .$$

λ is called forgetting factor and the method is called *exponentially weighted LS*.

Weighted LS filter

In the same way as earlier the correlation matrix, $\mathbf{\Phi}$, and crosscorrelation vector, $\mathbf{z}$, can now be defined:

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^H(i)$$

and

$$\mathbf{z}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)d^*(i) \ ,$$

,i.e, with exponential weighting and forgetting factor.

Weighted LS filter

The optimal filter at time n, $\widehat{\mathbf{w}}(n)$, based on data since sample 1 (expontially weighted) can now be calculated from the time-dependent normal equation.

$$\mathbf{\Phi}(n)\widehat{\mathbf{w}}(n) = \mathbf{z}(n)$$

as

$$\widehat{\mathbf{w}}_o(n) = \mathbf{\Phi}^{-1}(n)\mathbf{z}(n) \ .$$

Problem: We cannot invert this big matrix for every new sample.

Exponentially weighted recursive LS (RLS)

The last expression can be written

$$\begin{aligned}\widehat{\mathbf{w}}(n) &= \widehat{\mathbf{w}}(n-1) + \mathbf{k}(n)\left[d^*(n) - \mathbf{u}^H(n)\widehat{\mathbf{w}}(n-1)\right] \\ &= \widehat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n)\ .\end{aligned}$$

The vector $\mathbf{k}(n)$ is often called *gain vector*.

The RLS algorithm

Initialize the algorithm with

$$\begin{aligned}\mathbf{P}(0) &= \delta^{-1}\mathbf{I}_{M\times M} \\ \widehat{\mathbf{w}}(0) &= \mathbf{0}_{M\times 1}\end{aligned}$$

Then for every sample update

$$\begin{aligned}\mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)} \\ \xi(n) &= d(n) - \widehat{\mathbf{w}}^H(n-1)\mathbf{u}(n) \\ \widehat{\mathbf{w}}(n) &= \widehat{\mathbf{w}}(n-1) + \mathbf{k}(n)\xi^*(n) \\ \mathbf{P}(n) &= \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)\ .\end{aligned}$$

Convergence analysis, RLS

- The ensamble-averaged *learning curve* for the RLS converges in approximately $2M$ iterations. This is considerably faster than for the LMS.
- When the number of iteration goes to infinity, $E\{|\xi(n)|^2\}$ goes to σ^2 in $e_o(n) = d(n) - \mathbf{w}_o^H\mathbf{u}(n)$ (for stationary signalsr). No misadjustment.
- The convergence is independent of the eigenvales to the correlation matrix of $\mathbf{u}(n)$.

Lecture 7

These lecture notes contain

- What to read for Haykin, edition 4.
- What to read for Haykin, edition 3.
- Exam information.

To read, Week1

Chapter	Section	Comment
0	1–4	General understanding
	5	General understanding, (ej square-root, FastRLS)
	7	Good understanding
2	2.1–2.7	Repetition, alt. Hayes
4	4.1–4.5	Good understanding of principles
+		Exercises, computer exercises

To read, Week2

Chapter	Section	Comment
5	5.1–5.2	Very important
	5.3	Good understanding, (ej *application* 1, 3, 6)
	5.4	The lecture substitutes pages 238–269. Pages 269–272are important. Pages 273–278 is not included in the course
	5.5	Very important except the part on *small-step theory*
	5.6	Good understanding
	5.7	Very important, the Equalizer example
+		Exercises, computer exercises

To read, Week3

Chapter	Section	Comment
5	5.9	Good understanding
	5.13	Good understanding
6	6.1	UInderstanding corresponding to the lecture. Extra material from lecture on the home page F3.
+		Material in lecture notes which is not in the chapter like, Leaky-LMS, etc.
+		Excitation of filter, hidden instability, see addition to F3.
+		Echo canceller – general understanding and principles
+		Exercises, computer exercises

To read, Week4

Chapter	Section	Comment
7	Intro.	Very important
	7.1	Important. Lecture notes cover the important part.
	7.2	Important. Lecture notes are enough + the relationship between linear and circular convolution.
	7.3	Good understanding, Lecture notes are enough.
	7.4	Lecture notes are enough.
	7.5	General understanding
	7.7–7.8	General understanding
+		All block diagrams, see addition to F4 on the web.
+		Exercises, computer exercises, laboration 1 (parameter drift)

To read, Week5

Chapter	*Section*	*Comment*
8	8.1–8.4	Good understanding, jmf OSB
	8.5	Important
	8.6	Good understanding
9	9.1–9.3	Very important. Lecture covers the important part.
	9.8	Very important
	9.10	Overview
+		Exercises, computer exercises, laboration 2

To read, Week6

Chapter	*Section*	*Comment*
13	Inled.+ 13.1–13.2	Overview, (ej parameterdrift). The important parts are in the lecture notes.
14	Inled.+ 14.1	Lecture notes are enough.
+		Parameterdrift, see excitation of filters in the add to F3.
+		Exercises, computer exercises

To read, Week1

Chapter	*Section*	*Comment*
Intro.	1–3	General understanding
	4	General understanding, (ej square-root, FastRLS)
	7	Good understanding, (ej lattice, blind eq., beamforming)
5	5.1–5.7	Repetition, alt. Hayes
8	8.1–8.5	Good principiell understanding
+		Exercises, computer exercises

To read, Week2

Chapter	*Section*	*Comment*
9	9.1–9.2	Mycket viktigt
	9.3	Good understanding, (ej exempel 1, 3, 6)
	9.4	understanding motsvarande genomgång på föreläsningen. Finns på nätet som tillägg till F2. Sidorna 400-405 är viktiga.
	9.5	Mycket viktigt
	9.6	Good understanding
	9.7	Mycket viktigt, Equalizer-exemplet
+		Exercises, computer exercises

To read, Week3

Chapter	*Section*	*Comment*
9	9.9	Good understanding
	9.11	understanding motsvarande genomgång på föreläsningen. Finns på nätet som tillägg till F3.
	9.12	Viktigt
+		Material i föreläsningsanteckningarna som ej finns i ovan nämnda Chapter, Leaky-LMS, etc.
+		Excitering av filter, dold instabilitet, se tillägg till F3.
+		Ekosläckare – allmän understanding och princip
+		Exercises, computer exercises

To read, Week4

Chapter	*Section*	*Comment*
10	Inled.	Mycket viktigt
	10.1	Viktigt, lite rörigt: M skall vara L i ekv. (10.9) och (10.11). Fel i ekv. (10.20). Föreläsningsanteckningarna räcker.
	10.2	Viktigt. Föreläsningsanteckningarna räcker + linjär resp. cirkulär faltning, samt overlap-save.
	10.3	Good understanding, föreläsningsanteckningarna räcker.
	10.4	Kursivt, föreläsningsanteckningarna räcker.
	10.5–10.6	General understanding
+		Olika strukturer, se tillägg till F4 på nätet.
+		Exercises, computer exercises, laboration 1 (parameterdrift)

To read, Week5

Chapter	*Section*	*Comment*
11	11.1–11.4	General understanding, jmf OSB
	11.5	Viktigt
	11.6	General understanding
13	13.1–13.4	Mycket viktigt. Föreläsningsanteckningarna täcker det viktiga.
	13.7	Mycket viktigt
	13.9	Generalt
+		Exercises, computer exercises, laboration 2

To read, Week6

Chapter	*Section*	*Comment*
16	Inled.	Föreläsningsanteckningarna räcker för övrigt.
17	17.1–17.2	Generalt, (ej parameterdrift). Det viktiga finns i föreläsningsanteckningarna.
+		Parameterdrift, se excitering av filter i tillägget till F3.
+		Exercises, computer exercises

Exam

Time and place:

Thursday 16/12 14.00-19.00, in room Sparta C.

The following is allowed to bring:

- Course book (Haykin). Dock ej anteckningar med lösningar till övningsuppgifter.
- Course book from OSB (Hayes). Dock ej anteckningar med lösningar till övningsuppgifter.
- Formulas (typ TeFyMa och Beta).
- Formulas from basic course.
- Calculator.