# Statistical Signal Processing

*Lecture 9*

chapter 3: Optimal Filtering

Wiener filtering

- signal in noise
- equalization
- causal Wiener filtering
- FIR Wiener filtering
  - iterative solution: steepest-descent algorithm

chapter 4: Adaptive Filtering

- LMS algorithm
- Normalized LMS (NLMS) algorithm

## FIR Wiener Filtering

- LMMSE estimation of $x_k$ from $y_k$ using a causal FIR filter
- $\hat{x}_k = \sum_{i=0}^{N-1} h_i y_{k-i} = Y_k^T H = H^T Y_k$ where $Y_k = \begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-N+1} \end{bmatrix}$ , $H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix}$
- $$\begin{aligned} \text{MSE} = \xi(H) &= E\,(x_k - \hat{x}_k)^2 = E\left(x_k - H^T Y_k\right)^2 \\ &= E\left(x_k^2 - 2H^T Y_k x_k + H^T Y_k Y_k^T H\right) \\ &= \sigma_x^2 - 2H^T E\,(Y_k x_k) + H^T E\left(Y_k Y_k^T\right) H \\ &= \sigma_x^2 - 2H^T R_{Yx} + H^T R_{YY} H \end{aligned}$$
- MSE $\xi(H)$ is a quadratic cost function in $H$. Minimization $\Rightarrow$ gradient = 0

$$\frac{\partial\,\xi}{\partial\,H} \triangleq \begin{bmatrix} \frac{\partial \xi}{\partial h_0} \\ \vdots \\ \frac{\partial \xi}{\partial h_{N-1}} \end{bmatrix} = 2(R_{YY}H^o - R_{Yx}) = 0 \quad \Rightarrow \quad H^o = R_{YY}^{-1} R_{Yx}$$

- extremum = minimum because the Hessian $\left[\frac{\partial^2 \xi}{\partial h_i\,\partial h_j}\right]_{i,j=0}^{N-1} = 2R_{YY} > 0$

## FIR Wiener Filtering (2)

- The minimum MSE (MMSE) can be found to be:

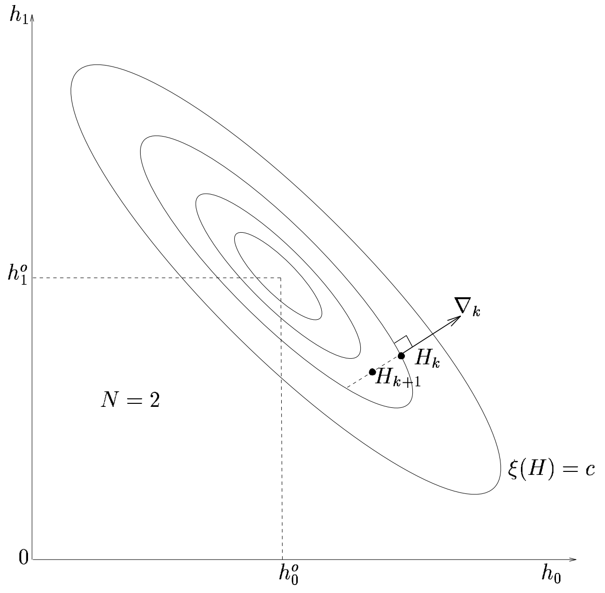
$$\begin{aligned}\xi^o \;=\; \xi(H^o) \;&=\; \sigma_x^2 - 2R_{Yx}^T H^o + H^{o\,T} R_{YY} H^o \\ &=\; \cdots = \; \sigma_x^2 - R_{xY}\, R_{YY}^{-1} R_{Yx} \;=\; \sigma_x^2 - R_{xY} H^o\end{aligned}$$

- alternatively, the criterion can be written as

$$\xi(H) \;=\; \xi^o + (H - H^o)^T\, R_{YY}\, (H - H^o)$$

which brings out clearly the parabolic character.

## FIR Wiener Filtering (3)



## Steepest Descent Algorithm

- iterative algorithm to solve the normal equations $R_{YY}\ H^o\ =\ R_{Yx}$, updates in the direction of steepest descent Let $\mu$ be the stepsize parameter. It determines how far we shall go in the direction of steepest descent. If $H_k$ represents the approximate solution at iteration step $k$, then we get the following recursion:

$$H_{k+1}\ =\ H_k - \frac{\mu}{2}\nabla_k$$

where

$$\nabla_k\ =\ \left.\frac{\partial\xi}{\partial H}\right|_{H=H_k}\ =\ 2R_{YY}H_k - 2R_{Yx}\ =\ 2R_{YY}\left(H_k - H^o\right)\ =\ -2R_{YY}\widetilde{H}_k$$

where we have introduced the filter approximation error $\widetilde{H}_k\ =\ H^o - H_k$ .

- So the iterative algorithm becomes

$$H_{k+1}\ =\ H_k - \mu\left(R_{YY}H_k - R_{Yx}\right)\ =\ \left(I - \mu R_{YY}\right)H_k + \mu R_{Yx}\ .$$

note: $\frac{1}{2}\nabla_k = R_{YY}H_k - R_{Yx}$, the error in the satisfaction of the system of normal equations of which we are seeking the solution, is used to adjust the approximation to the solution.

- Complexity: multiplying $R_{YY}$ by a vector.

## Steepest Descent Algorithm (2)

- Question: will the sequence $H_k$ converge, and if so, to $H^o$?

- For the analysis of the convergence process, we shall assume knowledge of $H^o\ =\ R_{YY}^{-1}R_{Yx}$ but we shall see that the conditions for convergence do not depend on $H^o$. The conditions for convergence are conditions on the stepsize $\mu$. From the derivation of the algorithm, it appears intuitively clear that the algorithm should work for a small positive $\mu$ since in that case we can be sure to make descending steps. If the steps we take are too big however, it is clear that we pass the valley and climb up the hill on the other side of the valley. So for large $\mu$ we do not expect convergence. Let us see what the analysis gives.

- Translation $\Rightarrow$ set of homogeneous equations

$$\widetilde{H}_{k+1}\ =\ \left(I - \mu R_{YY}\right)\widetilde{H}_k$$

This is a set of coupled equations since $R_{YY}$ is not a diagonal matrix in general.

## Steepest Descent Algorithm (3)

- Eigen decomposition of the covariance matrix $R$:

$$R_{YY} \;=\; EY_kY_k^T \;=\; V\Lambda V^T \;=\; \sum_{i=1}^{N} \lambda_i V_i V_i^T$$

where $\Lambda \;=\;$ diag $\{\lambda_1 \cdots \lambda_N\}$ , $V \;=\; [V_1\, V_2 \cdots V_N]$ are matrices containing the eigenvalues and eigenvectors respectively. Note : $V^TV \;=\; I \;=\; VV^T$ or in other words $V_i^TV_j \;=\; \delta_{ij}$. We also assume that the eigenvalues are ordered: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N > 0$.

- Rotating the principal axes using $V$: $\widetilde{H}_k \;=\; V\left(V^T\,\widetilde{H}_k\right) \;=\; \sum_{i=1}^{N} V_i\; v_i(k)$

$$V^T\widetilde{H}_{k+1} = V^T\left(I - \mu R_{YY}\right)\widetilde{H}_k = \left(V^T - \mu\Lambda V^T\right)\widetilde{H}_k = \left(I - \mu\Lambda\right)V^T\widetilde{H}_k$$

which is now a system of decoupled equations ($I - \mu\Lambda$ is a diagonal matrix).

- If we introduce the error components $v_i(k)$ as

$$\left(V^T\widetilde{H}_k\right) \;\triangleq\; \left[v_1(k)\;\; v_2(k) \cdots v_N(k)\right]^T$$

then we find the following decoupled dynamics for the $i$th *natural mode*:

$$v_i(k+1) \;=\; \left(1 - \mu\lambda_i\right)v_i(k)\;,\quad i = 1, \ldots, N\;.$$

## Steepest Descent Algorithm (4)

- The solution of these homogeneous difference equations of first order is readily obtained as

$$v_i(k) \;=\; \left(1 - \mu\lambda_i\right)^k v_i(o)\;,\quad i = 1, \ldots, N\;.$$

$v_i(k)\;$ = *geometric series* with geometric ratio = $1 - \mu\lambda_i$.

For *stability* or *convergence* of the steepest-descent algorithm:

$$-1 < 1 - \mu\lambda_i < 1\;,\quad i = 1, \ldots, N\;,\;\Rightarrow\; \lim_{k\to\infty}\left(1 - \mu\lambda_i\right)^k \;=\; 0\;,\quad i = 1, \ldots, N$$

Then $H_k \to H^o$, irrespective of $H_0$.

- necessary and sufficient condition for convergence: $0 < \mu < \dfrac{2}{\lambda_1} = \min\limits_{i\in\{1,\ldots,n\}} \dfrac{2}{\lambda_i}$

- In absence of knowledge of the $\lambda_i$, the following reasoning leads to a safe operating region: $\lambda_1 \;<\; \Sigma_{i=1}^{N}\lambda_i \;=\;$ tr $R_{YY} \;=\; N\sigma_y^2$ , hence sufficient condition

$$0 \;<\; \mu \;<\; \frac{2}{N\sigma_y^2} \;<\; \frac{2}{\lambda_1}$$

In practice, it will be much easier to determine $\sigma_y^2 = \left(R_{YY}\right)_{ii}$ than $\lambda_1$.

## Steepest Descent Algorithm (5)

- By taking the unit of time to be the duration of one iteration cycle, we can associate a time constant $\tau_i$ with the $i$th natural mode (for what follows, we assume $1-\mu\lambda_i > 0$):

$$|1-\mu\lambda_i|^k = e^{-\frac{k}{\tau_i}} \;,\quad |1-\mu\lambda_i| = e^{-\frac{1}{\tau_i}} \quad\Rightarrow\quad \tau_i = \frac{-1}{\ln|1-\mu\lambda_i|}$$

  which can be approximated, for small values of $\mu$, as $\tau_i \approx \frac{1}{\mu\lambda_i}\;, \quad \mu \ll 1$ . This shows that the smaller the stepsize parameter $\mu$, the slower will be the convergence of the steepest-descent algorithm.

- To see how the natural modes determine the evolution of the filter estimate $H_k$ in the original coordinate system, consider

$$H = H^o - \widetilde{H} = H^o - V\left(V^T\widetilde{H}\right) = H^o - \sum_{i=1}^{N} V_i v_i \;.$$

  So, when represented in the basis composed of the eigen vectors, the coordinates of $\widetilde{H}$ are $[v_1 \cdots v_N]$. The evolution of $H_k$ in terms of the natural modes becomes

$$\begin{aligned} H_k &= H^o - \widetilde{H}_k = H^o - V\left(V^T\widetilde{H}_k\right) = H^o - \sum_{i=1}^{N} V_i v_i(k) \\ &= H^o - \sum_{i=1}^{N} V_i v_i(0)\,(1-\mu\lambda_i)^k \;. \end{aligned}$$

## Steepest Descent Algorithm (6)

- MSE:

$$\begin{aligned} \xi(H) &= \xi^o + \widetilde{H}^T R_{YY}\widetilde{H} = \xi^o + \widetilde{H}^T V\Lambda V^T\widetilde{H} \\ &= \xi^o + [v_1 \cdots v_N]\begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_N \end{bmatrix}\begin{bmatrix} v_1 \\ \vdots \\ v_N \end{bmatrix} = \xi^o + \sum_{i=1}^{N} \lambda_i v_i^2 \;. \end{aligned}$$

- Hence, the set of points $H$ of constant cost function, $\xi(H) = c'$, is given by

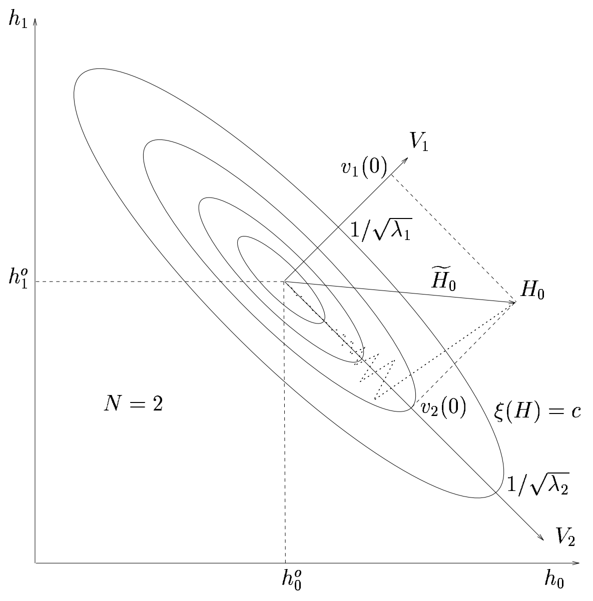
$$\sum_{i=1}^{N} \frac{v_i^2}{1/\lambda_i} = c = c' - \xi^o$$

  This is an ellipsoid (ellips when $N = 2$) centered at $H^o$ and with principal axes coinciding with the eigenvectors of $R_{YY}$. In the next figure, the intersections with the principal axes are indicated for the ellips corresponding to $c = 1$. The convergence of the $H_k$ is traced for a case in which $1-\mu\lambda_1 < 0$, $1-\mu\lambda_2 > 0$ and $|1-\mu\lambda_1| \ll |1-\mu\lambda_2|$. So for the component $v_1(k)$ along the $V_1$-axis, we get alternating signs and a fairly fast convergence. For the component $v_2(k)$ along the $V_2$-axis, we get a more slowly decaying exponential of constant sign.

- The evolution of the MSE :

$$\begin{aligned} \xi_k = \xi(H_k) &= \xi^o + \widetilde{H}_k^T R_{YY}\widetilde{H}_k = \xi^o + \widetilde{H}_k^T V\Lambda V^T\widetilde{H}_k = \xi^o + \sum_{i=1}^{N} \lambda_i v_i^2(k) \\ &= \xi^o + \sum_{i=1}^{N} \lambda_i\,(1-\mu\lambda_i)^{2k}\, v_i^2(0) \overset{\triangle}{=} \xi^o + \xi_k^e \geq \xi^o \;. \end{aligned}$$

## Steepest Descent Algorithm (7)



## Steepest Descent Algorithm (8)

- The MSE is the sum of the minimum MSE (MMSE) $\xi^o$ and what is called the *Excess MSE* (EMSE) $\xi^e$.
- The curve obtained by plotting the MSE $\xi_k$ as a function of iteration number $k$ is called the *learning curve*.
- In general, the learning curve of the steepest-descent algorithm consists of a sum of $N$ exponentials, each one of which corresponds to a natural mode of the algorithm. Again, when the stepsize satisfies the convergence condition, we get that irrespective of the initial conditions

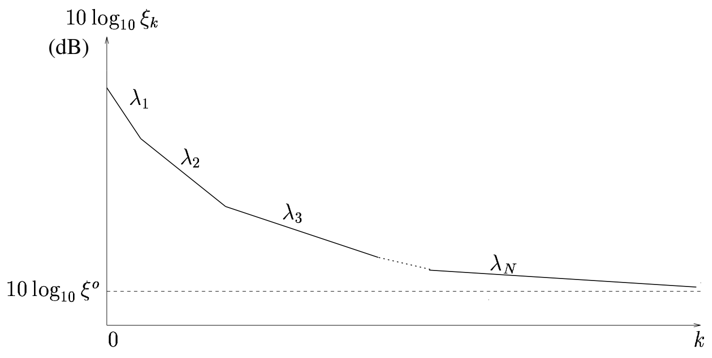
  $$\lim_{k\to\infty} \xi_k = \xi^o , \quad \lim_{k\to\infty} \xi_k^e = 0 .$$

  Note that the time constants for the MSE are half of the corresponding time constants for the natural modes in the convergence of the filter estimate.
- The learning curve is dominated by different modes at different stages (if the eigenvalues are well separated). The (approximate) piecewise linear form of the learning curve holds when $\lambda_1 \gg \lambda_2 \gg \cdots \gg \lambda_N$ and $1-\mu\lambda_1 > 0$. In that case $\tau_1 \ll \tau_2 \ll \cdots \ll \tau_N$.
- When the MSE is plotted in dB, an exponential decay of $\xi_k$ corresponds to a linear decay of $10\log_{10}\xi_k$.

## Steepest Descent Algorithm (9)

learning curve

- In the beginning, the learning curve follows the decay of the fastest mode, associated with $\lambda_1$. After about $2\tau_1 = 4\frac{\tau_1}{2}$ (the time constants for $\xi_k$ are half those for $H_k$), the first mode has died out while the other modes have hardly decreased in this short time span. Now the decay associated with mode two dominates the learning curve until about $2\tau_2$ etc.
- If $1-\mu\lambda_1 < 0$, then a similar reasoning still applies. But then the time constants are not ordered according to the $\lambda_i$ but according to the magnitudes $|1-\mu\lambda_i|$.



## Steepest Descent Algorithm (10)

- fastest convergence : $\min_\mu \max_i |1-\mu\lambda_i|$

  which leads to the following condition

$$1-\mu^o\lambda_1 = -\,(1-\mu^o\lambda_N) \;\Rightarrow\; \mu^o \;=\; \frac{2}{\lambda_1+\lambda_N} \;<\; \frac{2}{\lambda_1}$$

- (optimal) slowest mode : $\min_\mu \max_i |1-\mu\lambda_i| \;=\; \frac{\lambda_1-\lambda_N}{\lambda_1+\lambda_N}$

  This shows that in general the convergence of the steepest-descent algorithm slows down as the eigenvalue spread (the ratio $\lambda_1/\lambda_N$) increases.

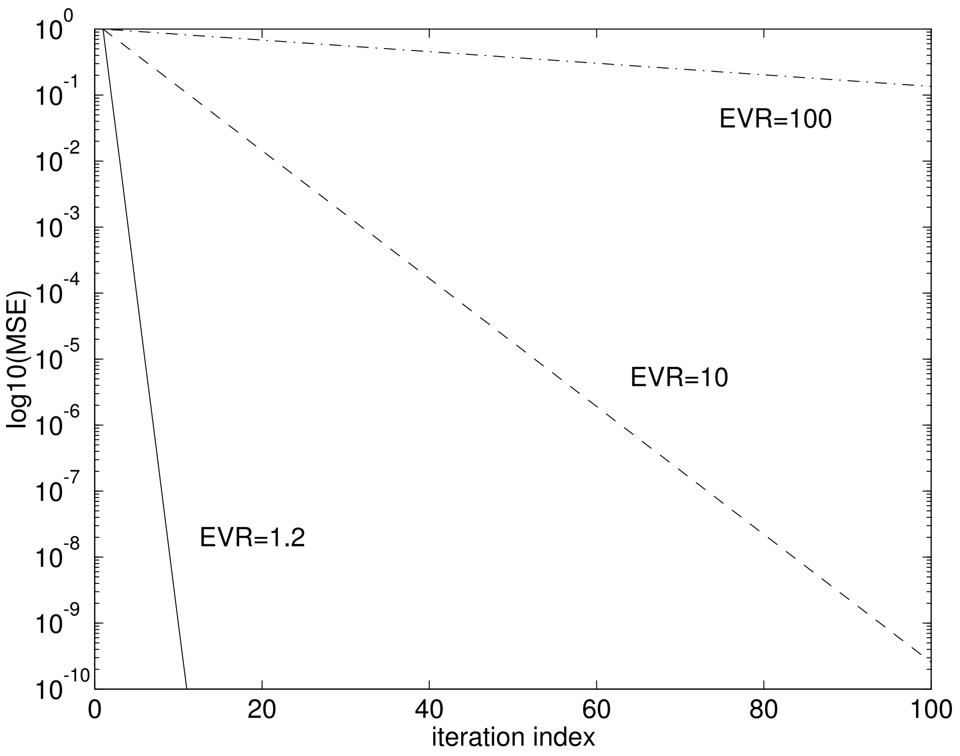
- Convergence is fastest when all eigenvalues are equal. In that case

$$\min_\mu \max_i |1-\mu\lambda_i| \;=\; 0$$

  for $\mu^o = 1/\lambda_1$. This means that convergence occurs in one iteration! Indeed, when all eigenvalues are equal, then the ellipsoids become spheres. In this case, all negative gradients (at any point in $H$ space) point towards $H^o$. So it suffices to take the right stepsize to end up at $H^o$ in one step.

## Steepest Descent Algorithm (11)

EVR=$\frac{\lambda_1}{\lambda_N}$, decay as $\left(\frac{\lambda_1-\lambda_N}{\lambda_1+\lambda_N}\right)^{2k}$



## LMS Algorithm

- *Adaptive Filtering*: $R_{YY}$ and $R_{Yx}$ not available but a realization $\{x_k, y_k, k \geq 0\}$ is available. From these samples, we could of course try to estimate $R_{YY}$ and $R_{Yx}$, and below we shall see that the *Recursive Least-Squares* (RLS) algorithm effectively does just that. However, we may also find that such an approach becomes too complicated, too computationally demanding. The *Least Mean Square* (LMS) algorithm takes the following alternative approach (invented by Widrow and Hopf).
- So actually: not $x_k$ of interest but $\hat{x}_k$, $\tilde{x}_k$, $H$ or $x_k$ at other time instants (training).
- Let's just drop the mathematical expectation operator $E$ in the MSE criterion, and instead consider the instantaneous squared error. So the new cost function is simply
$$J_k(H) \;=\; \epsilon_k^2(H) \;=\; (x_k - H^T Y_k)^2 \;.$$
Apply steepest-descent on this criterion, iteration index = time index now,
1 iteration per sample.
So we have dropped the statistical averaging operation, but we hope to replace it by some amount of time domain averaging inherent in the low-pass filtering nature of the adaptation process.
- Just as this criterion can be considered an instantaneous estimate of the MSE, the true criterion of interest, the gradient of $J_k$ can be considered to be an estimate of the true gradient.

## LMS Algorithm (2)

- $$\widehat{\nabla}_k = \left.\frac{\partial J_k}{\partial H}\right|_{H=H_{k-1}} = \left.\begin{bmatrix} \frac{\partial \epsilon_k^2}{\partial h_0} \\ \frac{\partial \epsilon_k^2}{\partial h_1} \\ \vdots \\ \frac{\partial \epsilon_k^2}{\partial h_{N-1}} \end{bmatrix}\right|_{H=H_{k-1}} = 2\epsilon_k^p \left.\begin{bmatrix} \frac{\partial \epsilon_k}{\partial h_0} \\ \frac{\partial \epsilon_k}{\partial h_1} \\ \vdots \\ \frac{\partial \epsilon_k}{\partial h_{N-1}} \end{bmatrix}\right|_{H=H_{k-1}} = -2\epsilon_k^p Y_k$$

  where $\epsilon_k^p \triangleq \epsilon_k(H_{k-1}) = x_k - H_{k-1}^T Y_k$ is the *predicted* or *a priori* error signal, obtained as the difference of the desired response and the filter output at time $k$, but using the filter estimate at time $k-1$.

- With this simple estimate of the gradient, we can specify a steepest-descent type of adaptive algorithm. We get

$$H_k = H_{k-1} - \frac{\mu}{2}\widehat{\nabla}_k = H_{k-1} + \mu\epsilon_k^p Y_k$$

  The last 2 equations specify the LMS algorithm.

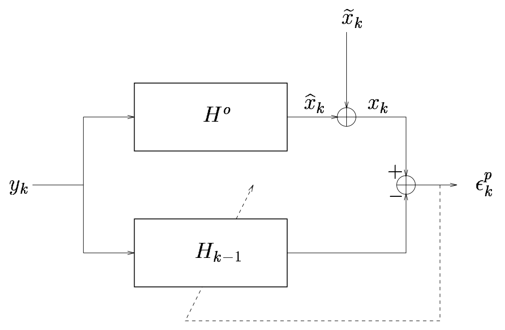
- Note: the LMS algorithm does one iteration per sample period.

- computational simplicity: $2N$ operations per sample

- The main question now is, does $H_k$ converge to the Wiener solution $H^o$ in any sense?

## LMS Algorithm (3)

a model for the purpose of analysis



- $x_k = \hat{x}_k + \tilde{x}_k = Y_k^T H^o + \tilde{x}_k$ where $\tilde{x}_k$ is independent white noise with variance $\xi^o$

- If we now consider estimating $x_k$ from $Y_k$ by some arbitrary FIR filter $H$, then the associated MSE becomes

$$\begin{aligned} \xi(H) &= E(x_k - H^T Y_k)^2 = E(\tilde{x}_k + H^{oT} Y_k - H^T Y_k)^2 \\ &= E(\widetilde{H}^T Y_k + \tilde{x}_k)^2 \ , \quad \widetilde{H} = H^o - H \\ &= E\tilde{x}_k^2 + 2\widetilde{H}^T E Y_k \tilde{x}_k + \widetilde{H}^T (E Y_k Y_k^T)\widetilde{H} \\ &= \xi^o + \widetilde{H}^T R_{YY} \widetilde{H} \end{aligned}$$

  This shows among other things that $H = H^o$ is indeed the Wiener solution.

## LMS Algorithm (4)

- *independence assumption* : treat $H_{k-1}$ and $Y_k$ as if they were independent
- *learning curve* :

$$\begin{aligned} E(\epsilon_k^p)^2 &= E(x_k - H_{k-1}^T Y_k)^2 \;=\; E(\tilde{x}_k + H^{oT} Y_k - H_{k-1}^T Y_k)^2 \\ &= E(\widetilde{H}_{k-1}^T Y_k + \tilde{x}_k)^2 \;\;,\;\; \widetilde{H}_k = H^o - H_k \\ &= E\tilde{x}_k^2 + 2E\widetilde{H}_{k-1}^T Y_k \tilde{x}_k + E\widetilde{H}_{k-1}^T (Y_k Y_k^T) \widetilde{H}_{k-1} \\ &= E\tilde{x}_k^2 + 2(E\widetilde{H}_{k-1}^T Y_k)(E\tilde{x}_k) + \text{tr}\left(E(Y_k Y_k^T)(\widetilde{H}_{k-1}\widetilde{H}_{k-1}^T)\right) \\ &= E\tilde{x}_k^2 + 2(E\widetilde{H}_{k-1}^T Y_k)0 + \text{tr}\left((EY_k Y_k^T)(E\widetilde{H}_{k-1}\widetilde{H}_{k-1}^T)\right) \\ &= \xi^o + \text{tr}(R_{YY} C_{k-1}) \;=\; E\xi(H_{k-1}) \end{aligned}$$

  where we used the independence assumption and introduced $C_k \triangleq E\widetilde{H}_k \widetilde{H}_k^T$.

- If $C_k \to 0$ then $\widetilde{H}_k \to 0$ or $H_k \to H^o$ in mean square.
- Note that convergence of the correlation matrix implies convergence of the mean and covariance of the estimation error:

$$C_k = E\widetilde{H}_k\widetilde{H}_k^T = (E\widetilde{H}_k)(E\widetilde{H}_k)^T + E(\widetilde{H}_k - E\widetilde{H}_k)(\widetilde{H}_k - E\widetilde{H}_k)^T$$

- In summary, in order to investigate the learning curve (MSE) of the LMS algorithm (the curve of $E\xi(H_k)$), we need to investigate how $C_k$ evolves.

## LMS Algorithm (5)

- From the LMS algorithm we get

$$\begin{aligned} \widetilde{H}_k \;=\; H^o - H_k &= H^o - H_{k-1} - \mu\epsilon_k^p Y_k \\ &= \widetilde{H}_{k-1} - \mu(Y_k^T \widetilde{H}_{k-1} + \tilde{x}_k) Y_k \\ \widetilde{H}_k &= \left(I - \mu Y_k Y_k^T\right) \widetilde{H}_{k-1} - \mu \tilde{x}_k Y_k \end{aligned}$$

- Compared to the steepest-descent algorithm, we now have a driving term (input).
- Also, in the system transition matrix, $R_{YY}$ got replaced by $Y_k Y_k^T$. The system matrix $I - \mu Y_k Y_k^T$ is now stochastic. This makes the system very hard to analyze. However, it is possible to introduce a simplification when the stepsize is small.
- independence assumption $\Rightarrow$ analysis of first order moment $E\widetilde{H}_k$

$$E\widetilde{H}_k \;=\; (I - \mu\, R_{YY})\; E\widetilde{H}_{k-1}$$

  The mean $E\widetilde{H}_k$ in LMS converges like $\widetilde{H}_k$ in steepest-descent. However, it is not because the mean of $\widetilde{H}_k$ converges to zero that $\widetilde{H}_k$ converges to zero. To see how $E(\epsilon_k^p)^2$ converges, we have to analyze the second-order moments of $\widetilde{H}_k$.

## LMS Algorithm (6)

*Averaging Theorem*

- Consider a (stationary) stochastic dynamic ($F_k$) system

$$\zeta_k \;=\; (I-\mu F_k)\zeta_{k-1} + W_k$$

- Now consider the averaged system $\overline{\zeta}_k \;=\; (I-\mu EF_k)\overline{\zeta}_{k-1} + W_k$

- difference between the two systems:

$$\begin{aligned} \zeta_k &= (I-\mu EF_k + \mu(EF_k - F_k))\zeta_{k-1} + W_k \\ \Rightarrow\quad (\zeta_k - \overline{\zeta}_k) &= (I-\mu EF_k)(\zeta_{k-1} - \overline{\zeta}_{k-1}) + \mu(EF_k - F_k))\zeta_{k-1} \end{aligned}$$

  and hence $\zeta_k - \overline{\zeta}_k \;\sim\; \mu$

- The averaging theorem consists of the following weak convergence result:

$$distribution\ of\ \{\zeta_k\} \;\rightarrow\; distribution\ of\ \{\overline{\zeta}_k\} \quad as\ \mu \rightarrow 0$$

  if the averaged system is exponentially stable.

- So if we want to study first and second order moments of $\zeta_k$, we may as well study those of $\overline{\zeta}_k$ which is generated by the simpler averaged system. The results become correct in the limit as the stepsize becomes very small.

## LMS Algorithm (7)

- Averaged LMS system: $\widetilde{H}_k = (I-\mu R_{YY})\,\widetilde{H}_{k-1} - \mu\tilde{x}_k Y_k$

- mean: $E\widetilde{H}_k = (I-\mu R_{YY})\,E\widetilde{H}_{k-1}$ same as steepest-descent!

- learning curve:

$$\begin{aligned} \xi_k \;\overset{\triangle}{=}\; E\xi(H_k) = E(\epsilon^p_{k+1})^2 &= \xi^o + \mathrm{tr}(R_{YY}C_k) \\ &= \xi^o + \mathrm{tr}(V\Lambda V^T C_k) \;=\; \xi^o + \mathrm{tr}(\Lambda V^T C_k V) \\ &= \xi^o + \Sigma_{i=1}^N \lambda_i (V^T C_k V)_{ii} \end{aligned}$$

  but

$$V^T C_k V \;=\; V^T E\widetilde{H}_k\widetilde{H}_k^T\,V \;=\; E\left(V^T\widetilde{H}_k\right)\left(V^T\widetilde{H}_k\right)^T \;=\; E\begin{bmatrix} v_1(k) \\ \vdots \\ v_N(k)\end{bmatrix}\begin{bmatrix} v_1(k) \\ \vdots \\ v_N(k)\end{bmatrix}^T$$

  and hence $(V^T C_k V)_{ii} \;=\; Ev_i^2(k)$ which leads to

$$\xi_k \;=\; \xi^o + \sum_{i=1}^N \lambda_i E v_i^2(k)\;.$$

## LMS Algorithm (8)

- in transformed coordinates: $V^T\widetilde{H}_k = (I - \mu\Lambda)\, V^T\widetilde{H}_{k-1} - \mu\tilde{x}_k V^T Y_k$
- put

$$V^T\widetilde{H}_k = \begin{bmatrix} v_1(k) \\ \vdots \\ v_N(k) \end{bmatrix} , \quad V^T Y_k = \begin{bmatrix} w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix} ,$$

$$E \begin{bmatrix} w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix} \begin{bmatrix} w_1(k) \\ \vdots \\ w_N(k) \end{bmatrix}^T = E V^T Y_k Y_k^T V = V^T R_{YY} V = \Lambda ,$$

then

$$v_i(k) = (1 - \mu\lambda_i)\, v_i(k{-}1) - \mu\tilde{x}_k w_i(k) \;.$$

- Taking the expected value of the square of both sides, we get

$$E v_i^2(k) = (1 - \mu\lambda_i)^2\, E v_i^2(k{-}1) + \mu^2 E\tilde{x}_k^2 w_i^2(k) = (1 - \mu\lambda_i)^2\, E v_i^2(k{-}1) + \mu^2 \xi^o \lambda_i$$

- Since we have the same dynamics as in the steepest-descent algorithm, we shall again have stability (convergence) when $0 < \mu < \frac{2}{\lambda_1}$.

## LMS Algorithm (9)

- However, the results here are based on the averaging theorem which assumes that $\mu$ is small. If based on such a theory, we derive a condition for the maximum possible value for $\mu$ to have convergence, then we can expect that result to be erroneous. Indeed, the range of stable stepsize values for convergence of the second-order moments in practice turns out to be quite a bit more conservative than $\frac{2}{\lambda_1}$.
- In any case, for $\mu$ within the stable range, $E v_i^2(k)$ will converge exponentially fast to some value $E v_i^2(\infty)$:

$$E v_i^2(\infty) \;=\; \frac{\mu}{2 - \mu\lambda_i} \xi^o \;.$$

- The learning curve converges to the value

$$\xi_\infty \;=\; \xi^o + \sum_{i=1}^{N} \lambda_i E v_i^2(\infty) \;=\; \xi^o + \xi^e \;=\; \xi^o + \xi^o \mu \sum_{i=1}^{N} \frac{\lambda_i}{2 - \mu\lambda_i} = \xi^o (1 + M)$$

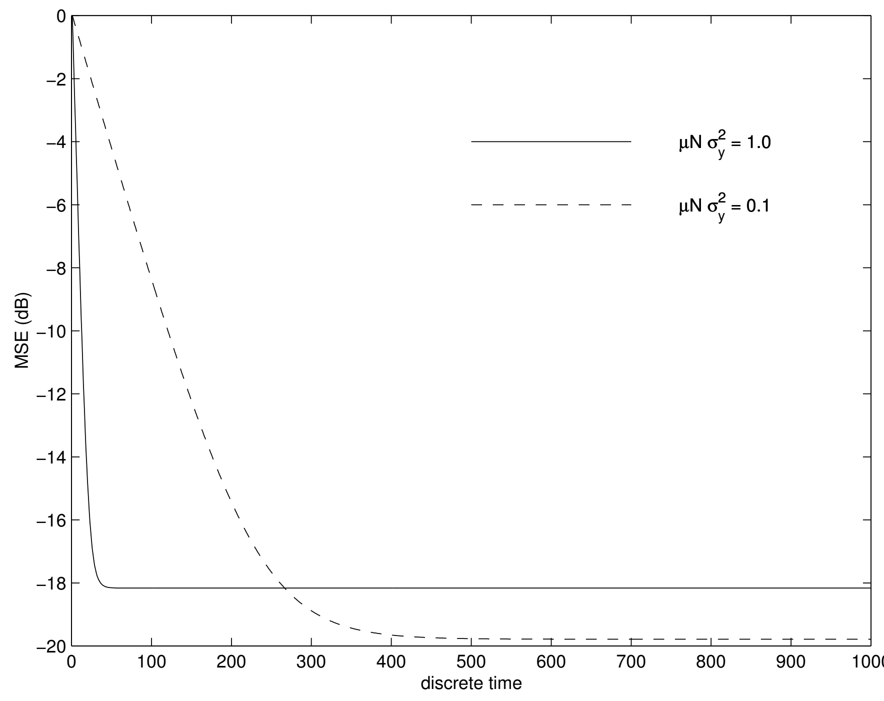
where

$$M \;=\; \frac{\xi^e}{\xi^o} \;=\; \mu \sum_{i=1}^{N} \frac{\lambda_i}{2 - \mu\lambda_i} \;\approx\; \frac{\mu}{2} \sum_{i=1}^{N} \lambda_i = \frac{\mu N \sigma_y^2}{2}$$

is called the *misadjustment* factor, $\xi^e$ the *excess* MSE, and the indicated approximation holds for small $\mu$.

## LMS Algorithm (11)

$$\xi_k = \underbrace{\xi^o}_{MMSE} + \underbrace{\sum_{i=1}^{N} \lambda_i (1-\mu\lambda_i)^{2k} \, E v_i^2(0)}_{\text{mean} \to 0} + \underbrace{\xi^o \, \mu \sum_{i=1}^{N} \frac{1-(1-\mu\lambda_i)^{2k}}{2-\mu\lambda_i} \lambda_i}_{\text{variance} \to \text{EMSE}} \,, \quad k \geq 1$$



## LMS Algorithm (10)

- Although $EH_\infty = H^o$ (the estimate is asymptotically unbiased), the estimate $H_k$ of $H^o$ is not *consistent* because its variance does not decrease to zero. Due to the use of a finite nonzero stepsize, the LMS algorithm will, even in steady-state, continue taking steps in the noisy gradient direction (even though the true gradient would be zero at convergence). This residual amount of variance is measured by the misadjustment factor $M$ or by the Excess MSE $\xi^e_\infty = \xi^o M$. The design of a constant stepsize $\mu$ results from a compromise: small values for $\mu$ lead to low misadjustment but slow convergence dynamics, large values lead to the opposite.

- *conditions for exact convergence* *gear shifting*
What if we require $M = 0$. Then we need $\mu = 0$ from our previous steady-state considerations. However, $\mu = 0$ will clearly not allow any adaptation. So what we need to do is to vary $\mu$ with $k$. The time-varying $\mu_k$ will have some finite value initially and decrease to zero as $k \to \infty$. One can show that under the following conditions

$$\mu_k \geq 0 \,, \; \sum_{k=1}^{\infty} \mu_k = \infty \,, \; \sum_{k=1}^{\infty} \mu_k^2 < \infty \quad \Rightarrow \quad \mu_k \overset{k\to\infty}{\longrightarrow} 0$$

the misadjustment converges to zero and $H_k \to H^o$, the Wiener solution, in mean square. A typical stepsize sequence : $\mu_k = \frac{c}{k}$

## Normalized LMS Algorithm

- LMS: applies steepest descent approach to instantaneous squared filtering error $\epsilon_k^2(H) = \left(x_k - H^T Y_k\right)^2$

$$\begin{cases} \epsilon_k^p &= x_k - H_{k-1}^T Y_k \\ H_k &= H_{k-1} + \mu \epsilon_k^p Y_k \end{cases} ,$$

  where $\epsilon_k^p = \epsilon_k\left(H_{k-1}\right)$ is the *a priori* (or *predicted*) filtering error.

- From our previous analysis: $\mu_{\max} \sim \frac{1}{\sigma_y^2}$. Therefore in order to be robust w.r.t. possible variations in the level of the input signal $y_k$, it is desirable to normalize the stepsize, to divide the stepsize by a quantity that behaves roughly as the variance $\sigma_y^2$. The so-called Normalized LMS (NLMS) algorithm takes

$$\mu_k = \frac{\overline{\mu}}{\|Y_k\|^2} . \quad \text{Note: } E\,\|Y_k\|^2 = N\sigma_y^2$$

- NLMS:

$$\begin{cases} \epsilon_k^p &= x_k - H_{k-1}^T Y_k \\ H_k &= H_{k-1} + \frac{\overline{\mu}}{\|Y_k\|^2}\, \epsilon_k^p\, Y_k \end{cases}$$

## Normalized LMS Algorithm (2)

- However, the consequences of the stepsize normalization reach much farther.
- Consider the *a posteriori* filtering error: $\epsilon_k = \epsilon_k\left(H_k\right)$.

$$\begin{cases} \text{LMS}: & \epsilon_k &= \left(1 - \mu \|Y_k\|^2\right) \epsilon_k^p \\ \text{NLMS}: & \epsilon_k &= \left(1 - \overline{\mu}\right) \epsilon_k^p \end{cases} .$$

  NLMS allows a precise control of the magnitude of the *a posteriori* filtering error $\epsilon_k$ , and in particular $\overline{\mu} = 1$ makes $\epsilon_k \equiv 0$. LMS only allows to make $\epsilon_k$ small on the average, with possibly large values occuring at certain time instants, depending on the variations of $\|Y_k\|^2$ with time.

- remember model $x_k = H^{oT} Y_k + \tilde{x}_k$ , ($H^o$ = FIR Wiener filter), $\widetilde{H}_k = H^o - H_k$ ,

$$\begin{cases} & \epsilon_k^p &= \widetilde{H}_{k-1}^T Y_k + \tilde{x}_k & \\ \text{LMS}: & \widetilde{H}_k &= \Phi_k^{LMS} \widetilde{H}_{k-1} - \mu \tilde{x}_k Y_k \ , & \Phi_k^{LMS} = I - \mu Y_k Y_k^T \\ \text{NLMS}: & \widetilde{H}_k &= \Phi_k^{NLMS} \widetilde{H}_{k-1} - \frac{\overline{\mu}}{\|Y_k\|^2} \tilde{x}_k Y_k \ , & \Phi_k^{NLMS} = I - \overline{\mu} \frac{Y_k Y_k^T}{Y_k^T Y_k} . \end{cases}$$

  state transition matrix $\Phi_k$ determines the dynamics and in particular the stability of the error system $\widetilde{H}_k$.

## Normalized LMS Algorithm (3)

- *long-term dynamics*: we used the averaging theorem, leading to the approximations
$$\begin{cases} \Phi_k^{LMS} & \approx & I - \mu R_{YY} \\ \Phi_k^{NLMS} & \approx & I - \dfrac{\overline{\mu}}{\mathrm{tr} R_{YY}} R_{YY} \end{cases} ,$$
where we have used for NLMS an additional approximation that is valid for large $N$. The analysis based on the averaging theorem has revealed the dependence of the dynamics on the eigenvalue spread $\lambda_1/\lambda_N$ of $R_{YY}$.

- *instantaneous dynamics*: $\Phi_k$ can be shown to have the following eigenvalues
$$\begin{cases} \nu_1(k) & = & \nu_2 \;=\cdots=\; \nu_{N-1}(k) = 1 \quad \text{for LMS and NLMS} \\ \nu_N^{LMS}(k) & = & 1 - \mu \|Y_k\|^2 \\ \nu_N^{NLMS}(k) & = & 1 - \overline{\mu} \end{cases} ,$$
and the eigenvector $W_N(k)$ corresponding to $\nu_N(k)$ is proportional to $Y_k$ (what are the eigenvectors corresponding to $\nu_1(k), \nu_2(k), \cdots, \nu_{N-1}(k)$?).

- Remark that we can write for both algorithms
$$\epsilon_k = \nu_N(k) \epsilon_k^p \;\; .$$

## Normalized LMS Algorithm (4)

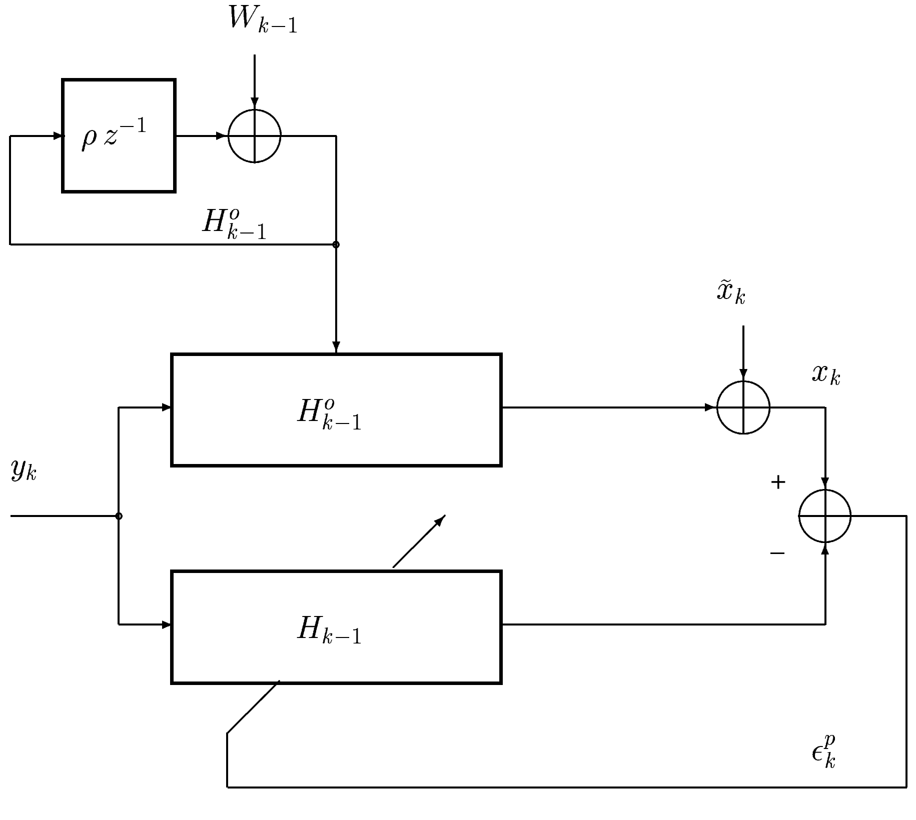
- $W_i(k)$ = eigenvector of $\Phi_k$ associated with eigenvalue $\nu_i(k)$. The eigenvectors form an orthonormal basis for $\mathcal{R}^N$ and we can always write $\widetilde{H}_k = W W^T \widetilde{H}_k = \sum_{i=1}^{N} W_i \left( W_i^T \widetilde{H}_k \right)$. We can write for both algorithms
$$\begin{cases} W_i^T \widetilde{H}_k & = & W_i^T \widetilde{H}_{k-1} \;\; , \; i = 1, 2, \cdots, N-1 \\ W_N^T \widetilde{H}_k & = & \nu_N(k) W_N^T \widetilde{H}_{k-1} - \mu_k \tilde{x}_k W_N^T Y_k \end{cases} , \quad \mu_k = \begin{cases} \mu & \text{for LMS,} \\ \dfrac{\overline{\mu}}{\|Y_k\|^2} & \text{for NLMS.} \end{cases}$$
Hence, for the components of $\widetilde{H}_{k-1}$ along directions orthogonal to $Y_k$, nothing changes. The component of $\widetilde{H}_{k-1}$ along the direction of $Y_k$ however gets multiplied by $\nu_N(k)$.

- For NLMS, $\nu_N^{NLMS}(k)$ can be controlled very precisely by $\overline{\mu}$. The choice for $\overline{\mu}$: trade-off between fast dynamics = small $|1-\overline{\mu}|$ = big $\overline{\mu}$, and a small noise amplification factor $\overline{\mu}$ (second (driving) term). In particular, $\overline{\mu} = 1$ results in complete elimination of the accumulated error component of $\widetilde{H}_{k-1}$ in the direction of $Y_k$ (also, $\epsilon_k = 0$ for $\overline{\mu} = 1$). However, with $\tilde{x}_k \neq 0$, some noise gets injected in the component of $\widetilde{H}_k$ along $Y_k$.

## Normalized LMS Algorithm (5)

- For LMS, if we want to make $\left|\nu_N^{LMS}(k)\right|$ small on the average , we need to choose $\mu$ fairly big. However, due to the statistical fluctuations of $\|Y_k\|$, $\left|\nu_N^{LMS}(k)\right|$ may get quite a bit bigger than 1 at certain times $k$, if $\mu$ is big. That means that even though LMS is converging on the average, it may actually take diverging steps at certain isolated time instants. This happens especially when $\mu$ is relatively large, when we want the average convergence to be as fast as possible.
- This problem of instantaneous diverging steps does not occur in the NLMS algorithm. Therefore, NLMS always converges faster than LMS when both algorithms have a stepsize that is optimized for fastest convergence speed.

## Tracking Time-Varying Filters



$W_{k-1}$
$\rho\, z^{-1}$
$H^o_{k-1}$
$H^o_{k-1}$
$\tilde{x}_k$
$x_k$
$y_k$
+
$-$
$H_{k-1}$
$\epsilon^p_k$