Linear FIR Adaptive Filtering
Gradient based adaptation: Steepest Descent Method

Adaptive filtering: Problem statement

- Consider the family of variable parameter FIR filters, computing their output according to

$$\begin{aligned} y(n) &= w_0(n)u(n) + w_1(n)u(n-1) + \ldots w_{M-1}(n)u(n-M+1) \\ &= \sum_{k=0}^{M-1} w_k(n)u(n-k) = \underline{w}(n)^T \underline{u}(n), \quad n = 0, 1, 2, \ldots, \infty \end{aligned}$$

  where parameters $\underline{w}(n)$ are allowed to change at every time step, $u(t)$ is the input signal, $d(t)$ is the desired signal and $\{u(t)$ and $d(t)\}$ are jointly stationary.

- Given the parameters $\underline{w}(n) = [w_0(n), w_1(n), w_2(n), \ldots, w_{M-1}(n)]^T$, find an adaptation mechanism $\underline{w}(n+1) = \underline{w}(n) + \delta\underline{w}(n)$, or written componentwise

$$\begin{aligned} w_0(n+1) &= w_0(n) + \delta w_0(n) \\ w_1(n+1) &= w_1(n) + \delta w_1(n) \\ &\ldots \\ w_{M-1}(n+1) &= w_{M-1}(n) + \delta w_{M-1}(n) \end{aligned}$$

  such as the adaptation process converges to the parameters of the optimal Wiener filter, $\underline{w}(n+1) \to \underline{w}_o$, no matter where the iterations are initialized (i.e. $\forall \ \ \underline{w}(0)$).

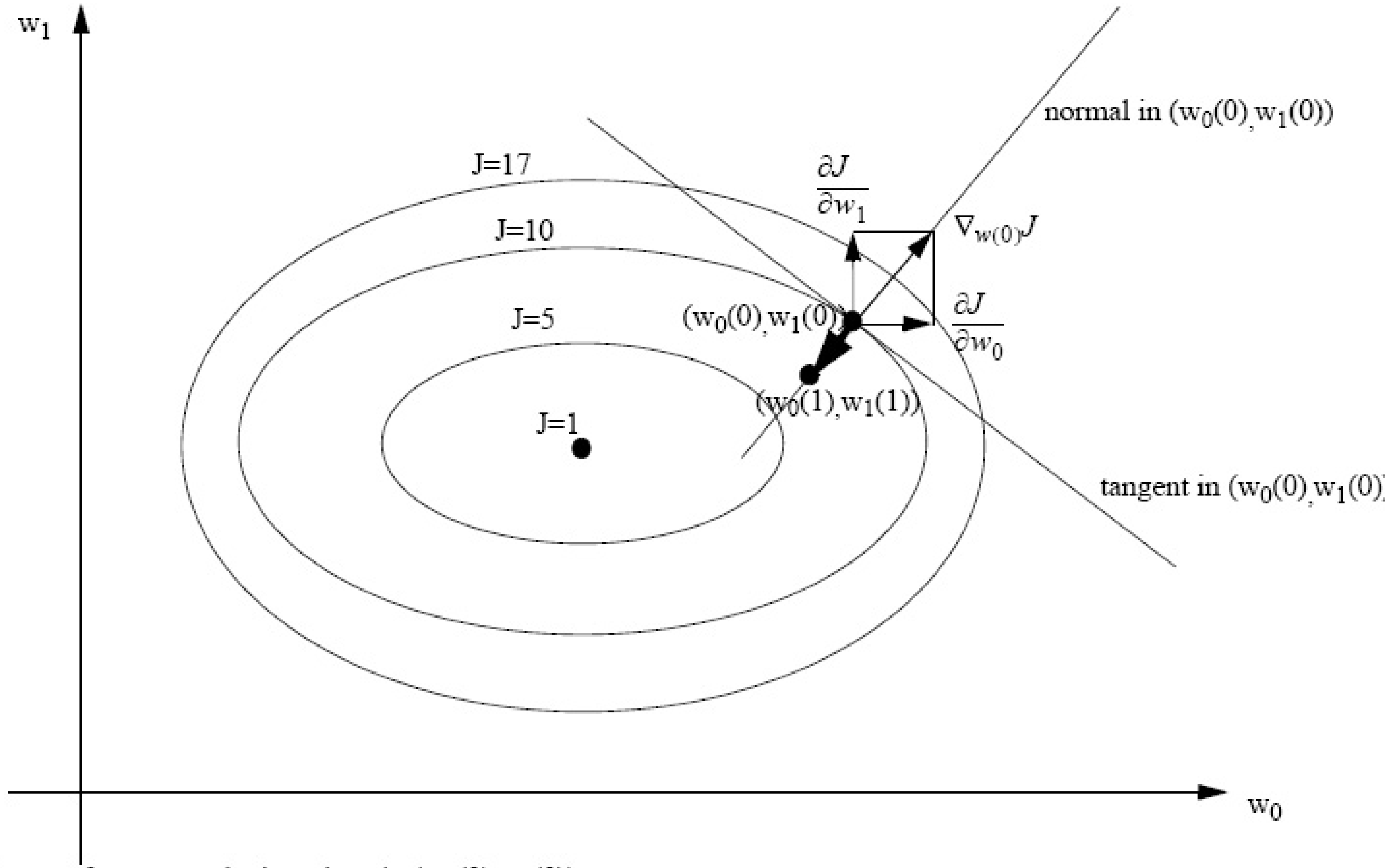
□

The key of adaptation process is the existence of an error between the output of the filter $y(n)$ and the desired signal $d(n)$:

$$e(n) = d(n) - y(n) = d(n) - \underline{w}(n)^T \underline{u}(n)$$

If the parameter vector $\underline{w}(n)$ will be used at all time instants, the performance criterion will be (see last Lecture)

$$\begin{aligned}
J(n) = J_{\underline{w}(n)} &= E[e(n)e(n)] = E[(d(n) - \underline{w}^T(n)\underline{u}(n))(d(n) - \underline{u}^T(n)\underline{w}(n))] \\
&= E[d^2(n)] - 2E[d(n)\underline{u}^T(n)]\underline{w}(n) + \underline{w}^T(n)E[\underline{u}(n)\underline{u}^T(n)]\underline{w}(n) \\
&= \sigma_d^2 - 2\underline{p}^T\underline{w}(n) + \underline{w}^T(n)R\underline{w}(n) \\
&= \sigma_d^2 - 2\sum_{i=0}^{M-1} p(-i)w_i + \sum_{l=0}^{M-1}\sum_{i=0}^{M-1} w_l w_i R_{i,l} \qquad (1)
\end{aligned}$$



At time n= 0, we start the iterations in $(w_0(0), w_1(0))$.

We will move at time n=1 to the next point, $(w_0(1), w_1(1))$ in with a step μ in a direction opposite to that of gradient vector:

$$-\mu \nabla_{w(0)} J$$

Expressions of the gradient vector

The gradient of the criterion $J(n)$ with respect to the parameters $\underline{w}(n)$ is

$$\nabla_{\underline{w}(n)} J(n) = \begin{bmatrix} \frac{\partial J(n)}{\partial w_0(n)} \\ \frac{\partial J(n)}{\partial w_1(n)} \\ . \\ . \\ . \\ \frac{\partial J(n)}{\partial w_{M-1}(n)} \end{bmatrix} = \begin{bmatrix} -2p(0) + 2\sum_{i=0}^{M-1} R_{0,i} w_i(n) \\ -2p(-1) + 2\sum_{i=0}^{M-1} R_{1,i} w_i(n) \\ . \\ . \\ . \\ -2p(-M+1) + 2\sum_{i=0}^{M-1} R_{1,M-1} w_i(n) \end{bmatrix} = -2\underline{p} + 2R\underline{w}(n) \tag{2}$$

$$\nabla_{\underline{w}(n)} J(n) = \begin{bmatrix} -2Ed(n)u(n) + 2\sum_{i=0}^{M-1} Eu(n)u(n-i)w_i(n) \\ -2Ed(n)u(n-1) + 2\sum_{i=0}^{M-1} Eu(n-1)u(n-i)w_i(n) \\ . \\ . \\ . \\ -2Ed(n)u(n-M+1) + 2\sum_{i=0}^{M-1} Eu(n-M+1)u(n-i)w_i(n) \end{bmatrix}$$

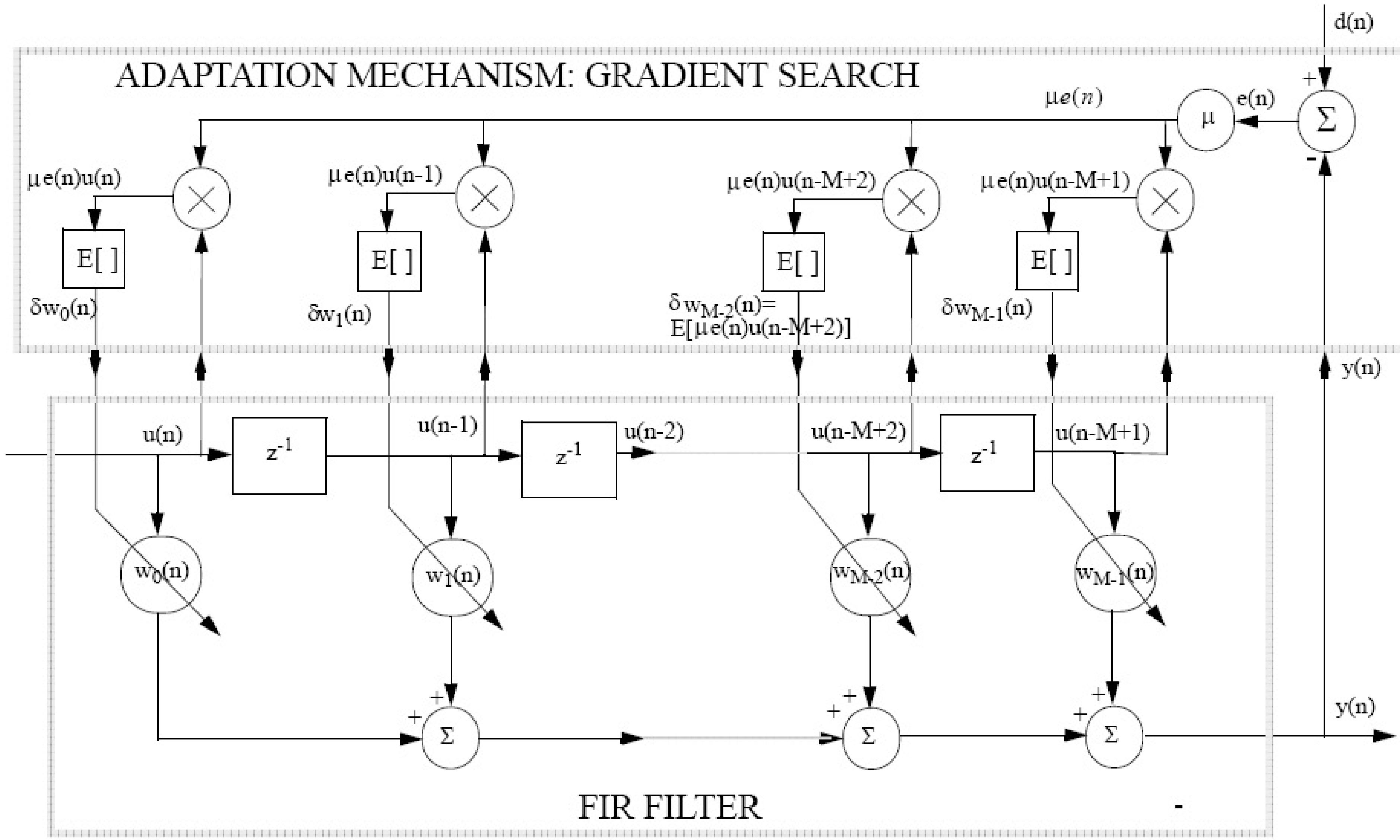
$$= \begin{bmatrix} -2Eu(n)\left(d(n) - \sum_{i=0}^{M-1} u(n-i)w_i(n)\right) \\ -2Eu(n-1)\left(d(n) - \sum_{i=0}^{M-1} u(n-i)w_i(n)\right) \\ . \\ . \\ . \\ -2Eu(n-M+1)\left(d(n) - \sum_{i=0}^{M-1} u(n-i)w_i(n)\right) \end{bmatrix}$$

$$= \begin{bmatrix} -2E\left(u(n)e(n)\right) \\ -2E\left(u(n-1)e(n)\right) \\ \cdot \\ \cdot \\ \cdot \\ -2E\left(u(n-M+1)e(n)\right) \end{bmatrix} = -2E \begin{bmatrix} u(n) \\ u(n-1) \\ \cdot \\ \cdot \\ \cdot \\ u(n-M+1) \end{bmatrix} e(n) = -2E\underline{u}(n)e(n) \tag{3}$$

((2) and (3) give the solution of exercise 7, page 229 [Haykin 2002]).

Steepest descent algorithm

$$\begin{aligned} \underline{w}(n+1) &= \underline{w}(n) - \tfrac{1}{2}\mu \nabla_{\underline{w}(n)} J(n) \\ &= \underline{w}(n) + \mu E \underline{u}(n)e(n) \end{aligned}$$



ADAPTATION MECHANISM: GRADIENT SEARCH
d(n)
+
Σ
-
e(n)
μ
$\mu e(n)$
μ e(n)u(n)
μ e(n)u(n-1)
μ e(n)u(n-M+2)
μ e(n)u(n-M+1)
E[ ]
E[ ]
E[ ]
E[ ]
$\delta w_0(n)$
$\delta w_1(n)$
$\delta w_{M-2}(n)=$
E[μ e(n)u(n-M+2)]
$\delta w_{M-1}(n)$
y(n)
u(n)
z^{-1}
u(n-1)
z^{-1}
u(n-2)
u(n-M+2)
z^{-1}
u(n-M+1)
$w_0(n)$
$w_1(n)$
$w_{M-2}(n)$
$w_{M-1}(n)$
+
+
Σ
+
+
Σ
+
+
Σ
y(n)
FIR FILTER
-

SD ALGORITHM

Steepest descent search algorithm for finding the Wiener FIR optimal filter

Given

- the autocorrelation matrix $R = E\underline{u}(n)\underline{u}^T(n)$
- the cross-correlation vector $\underline{p}(n) = E\underline{u}(n)d(n)$

Initialize the algorithm with an arbitrary parameter vector $\underline{w}(0)$.

Iterate for $n = 0, 1, 2, 3, \ldots, n_{max}$

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

Stop iterations if $\|\underline{p} - R\underline{w}(n)\| < \varepsilon$

$\square$

Designer degrees of freedom: $\mu, \varepsilon, n_{max}$

Equivalent forms of the adaptation equations We have the following equivalent forms of adaptive equations, each showing one facet (or one interpretation) of the adaptation process.

1. Solving $\underline{p} = R\underline{w}_o$ for $\underline{w}_o$ using iterative schemes:
   $\underline{w}(n+1) = \underline{w}(n) + \mu[\underline{p} - R\underline{w}(n)]$
2. or componentwise
   $w_0(n+1) = w_0(n) + \mu(p(0) - \sum_{i=0}^{M-1} R_{0,i} w_i(n))$
   $w_1(n+1) = w_1(n) + \mu(p(-1) - \sum_{i=0}^{M-1} R_{1,i} w_i(n))$
   $\ldots$
   $w_{M-1}(n+1) = w_{M-1}(n) + \mu(p(-M+1) - \sum_{i=0}^{M-1} R_{M-1,i} w_i(n))$
3. Error driven adaptation process (See Figure at page 2'):
   $\underline{w}(n+1) = \underline{w}(n) + \mu[Ee(n)\underline{u}(n)]$
4. or componentwise
   $w_0(n+1) = w_0(n) + \mu(Ee(n)u_0(n))$
   $w_1(n+1) = w_1(n) + \mu(Ee(n)u_1(n))$
   $\ldots$
   $w_{M-1}(n+1) = w_{M-1}(n) + \mu(Ee(n)u_{M-1}(n))$

Example: Running SD algorithm to solve for the Wiener filter derived in the example of Lecture 2.

$$R\underline{w} = \underline{p};$$

$$\begin{bmatrix} r_u(0) & r_u(1) \\ r_u(1) & r_u(0) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} Ed(n)u(n) \\ Ed(n)u(n-1) \end{bmatrix}$$

$$\begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0.5271 \\ -0.4458 \end{bmatrix}$$

with the solution

$$\begin{bmatrix} w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0.8362 \\ -0.7854 \end{bmatrix}$$

We will use the Matlab code

```
R_u=[1.1 0.5  ;               % Define autocovariance matrix
     0.5 1.1 ];
p= [0.5271 ; -0.4458]         % Define cross-correlation vector
W=[-1;-1]; mu= 0.01;          % Initial values for the adaptation algorithm
Wt=W;                         % Wt will record the evolution of vector W
 for k=1:1000                 % Iterate 1000 times the adaptation step
   W=W +mu*(p-R_u*W);         %    Adaptation Equation ! Quite simple!
   Wt=[Wt W];                 % Wt records the evolution of vector W
 end                          % Is W the Wiener Filter?
```

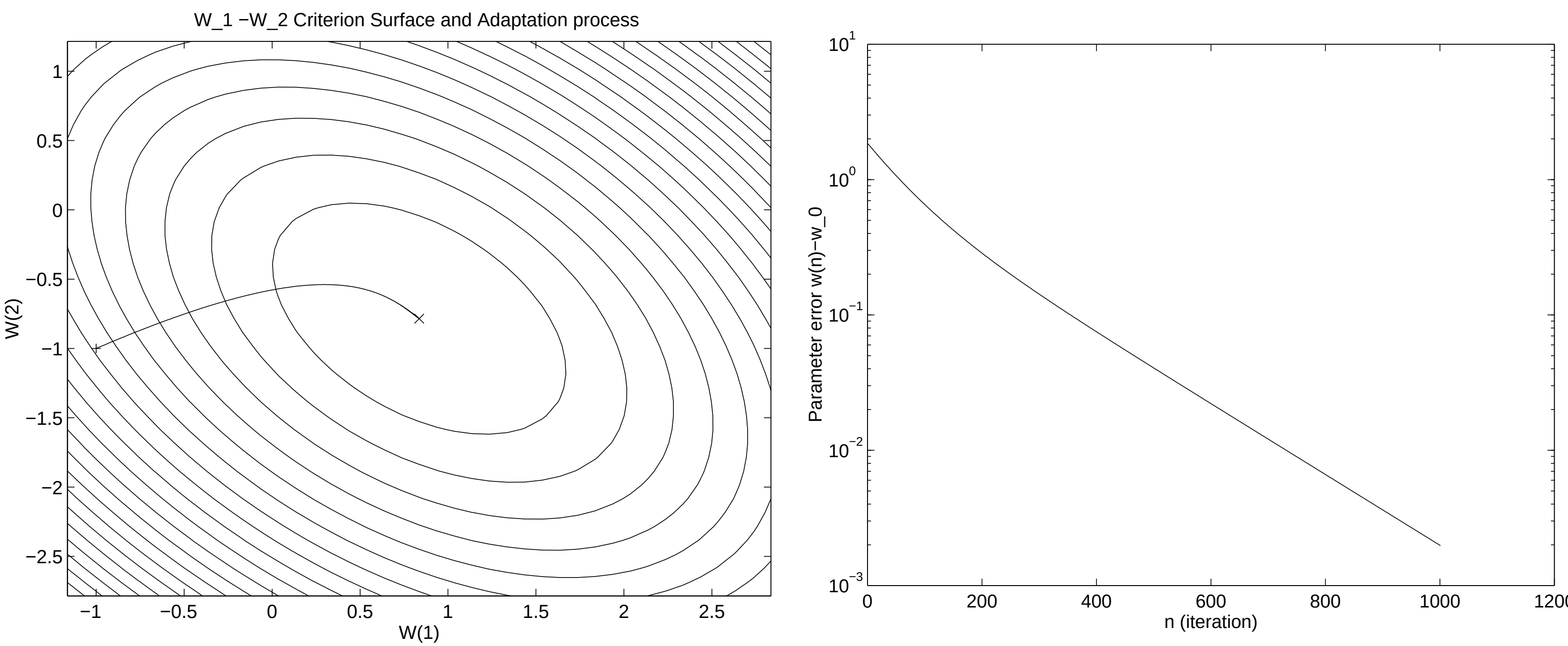
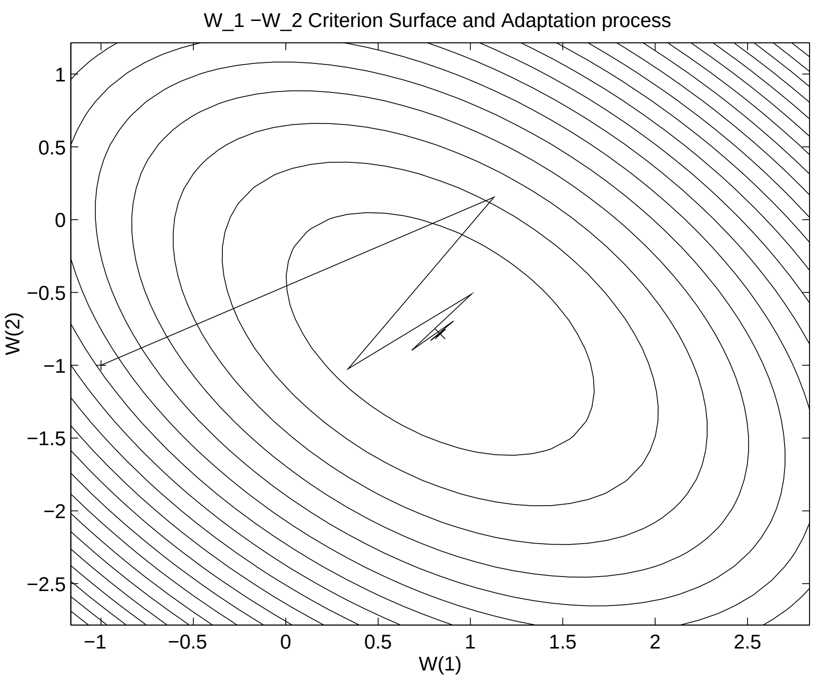


Figure 1: **Adaptation Step** $\mu = 0.01$. *Left*: Adaptation process starts from an arbitary point $\underline{w}(0) = [-1,\ -1]^T$ – marked with a cross – and converges to Wiener filter parameters $\underline{w}(0) = [0.8362\ \ -0.7854]^T$ ("x" point, in the center of ellipses). *Right* The convergence rate is exponential (note the logarithmic scale for the parametric error). In 1000 iterations the parameter error reaches 10^{-3}.



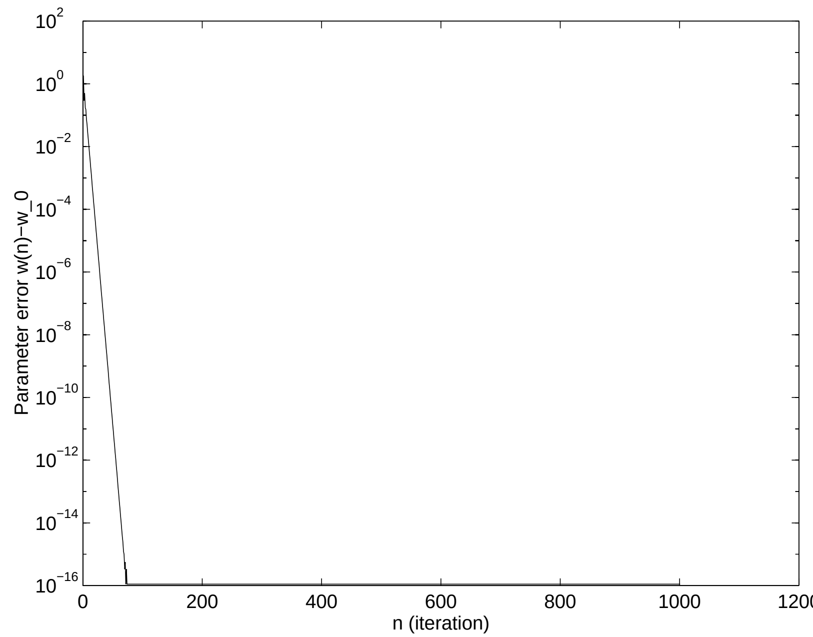


Figure 2: **Adaptation Step** $\mu = 1$. *Left*: Stable oscillatory adaptation process starting from an arbitary point $\underline{w}(0) = [-1,\ -1]^T$ – marked with a cross – and **convergence** to Wiener filter parameters $\underline{w}(0) = [0.8362\ \ -0.7854]^T$ ("x" point, in the center of ellipses). *Right* The convergence rate is exponential . In about 70 iterations the parameter error reaches 10^{-16}.

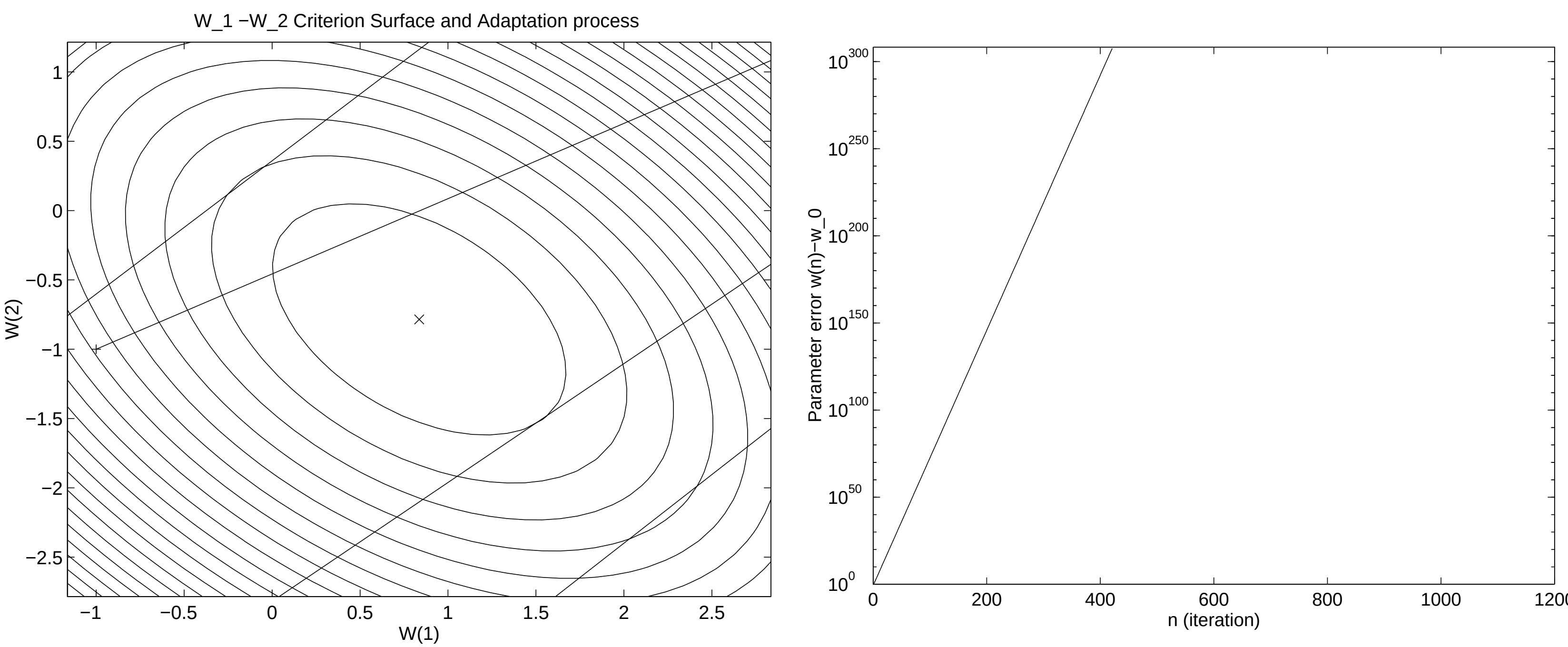


Figure 3: **Adaptation Step** $\mu = 4$. *Left*: Unstable oscillatory adaptation process starting from an arbitary point $\underline{w}(0) = [-1,\ -1]^T$ – marked with a cross – and **divergence** from Wiener filter parameters $\underline{w}(0) = [0.8362\ \ -0.7854]^T$ ("x" point, in the center of ellipses). *Right* The divergence rate is exponential . In about 400 iterations the parameter error reaches 10^{30}.

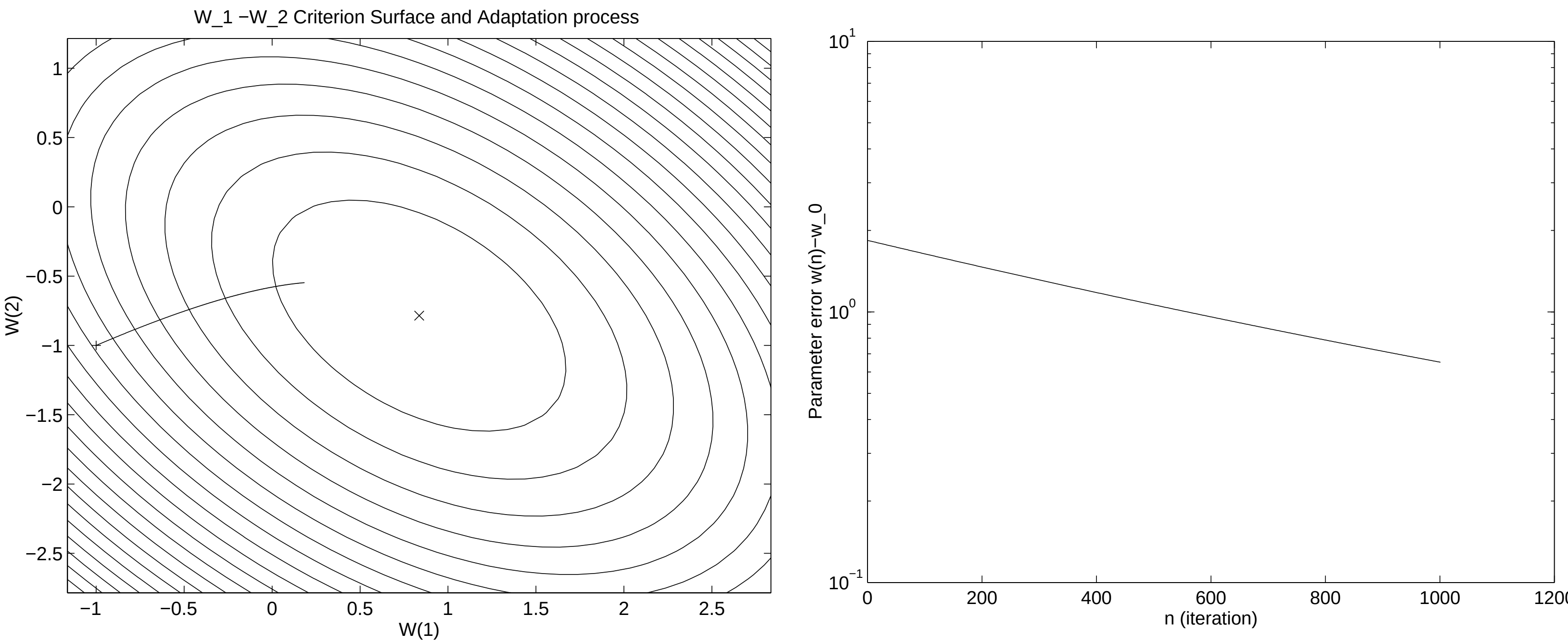


Figure 4: **Adaptation Step** $\mu = 0.001$ *Left*: Very slow adaptation process, starting from an arbitary point $\underline{w}(0) = [-1,\ -1]^T$ – marked with a cross. It will eventually converge to Wiener filter parameters $\underline{w}(0) = [0.8362\ \ -0.7854]^T$ ("x", point in the center of ellipses). *Right* The convergence rate is exponential (note the logarithmic scale for the parametric error). In 1000 iterations the parameter error reaches 0.7.

Stability of Steepest – Descent algorithm

Write the adaptation algorithm as

$$\underline{w}(n+1) = \underline{w}(n) + \mu[\underline{p} - R\underline{w}(n)] \tag{4}$$

But from Wiener-Hopf equation $\underline{p} = R\underline{w}_o$ and therefore

$$\underline{w}(n+1) = \underline{w}(n) + \mu[R\underline{w}_o - R\underline{w}(n)] = \underline{w}(n) + \mu R[\underline{w}_o - \underline{w}(n)] \tag{5}$$

Now subtract Wiener optimal parameters,$\underline{w}_o$, from both members

$$\underline{w}(n+1) - \underline{w}_o = \underline{w}(n) - \underline{w}_o + \mu R[\underline{w}_o - \underline{w}(n)] = (I - \mu R)[\underline{w}(n) - \underline{w}_o]$$

and introducing the vector $\underline{c}(n) = \underline{w}(n) - \underline{w}_o$, we have

$$\underline{c}(n+1) = (I - \mu R)\underline{c}(n)$$

Let $\lambda_1, \lambda_2, \ldots, \lambda_M$ be the (real and positive) eigenvalues and let $\underline{q}_1, \underline{q}_2, \ldots, \underline{q}_M$ be the (generally complex) eigenvectors of the matrix R, thus satisfying

$$R\underline{q}_i = \lambda_i \underline{q}_i \tag{6}$$

Then the matrix $Q = [\underline{q}_1 \ \underline{q}_2 \ \ldots \ \underline{q}_M]$ can transform R to diagonal form $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_M)$

$$R = Q\Lambda Q^H \tag{7}$$

where the superscript H means complex conjugation and transposition.

$$\underline{c}(n+1) = (I - \mu R)\underline{c}(n) = (I - \mu Q\Lambda Q^H)\underline{c}(n)$$

Left- multiplying by Q^H

$$\underline{Q}^H c(n+1) = Q^H(I - \mu Q \Lambda Q^H)\underline{c}(n) = (Q^H - \mu Q^H Q \Lambda Q^H)\underline{c}(n) = (I - \mu\Lambda)\underline{Q}^H c(n)$$

We notate $\underline{\nu}(n)$ the rotated and translated version of $\underline{w}(n)$

$$\underline{\nu}(n) = Q^H \underline{c}(n) = Q^H(\underline{w}(n) - \underline{w}_o) \tag{8}$$

and now we have

$$\underline{\nu}(n+1) = (I - \mu\Lambda)\underline{\nu}(n)$$

where the initial value $\nu(0) = Q^H(\underline{w}(0) - \underline{w}_o)$. We can write componentwise the recursions for $\underline{\nu}(n)$

$$\nu_k(n+1) = (1 - \mu\lambda_k)\nu_k(n) \quad k = 1, 2, \ldots, M$$

which can be solved easily to give

$$\nu_k(n) = (1 - \mu\lambda_k)^n \nu_k(0) \quad k = 1, 2, \ldots, M$$

For the stability of the algorithm

$$-1 < 1 - \mu\lambda_k < 1 \quad k = 1, 2, \ldots, M$$

or

$$0 < \mu < \frac{2}{\lambda_k} \quad k = 1, 2, \ldots, M$$

or

$$0 < \mu < \frac{2}{\lambda_{max}} \quad STABILITY CONDITION!$$

where λ_{max} is the maximum eigenvalue of autocovariance matrix R.

If the stability condition is respected, this will result in

$$\nu_k(n) \to 0, \text{ i.e. } w_k(n) \to w_{ok} \text{ when } n \to \infty, \qquad k = 1, 2, \ldots, M$$

Notating $\tau_k = \frac{-1}{\log(|1-\mu\lambda_k|)}$, we have

$$|1 - \mu\lambda_k| = exp(-\frac{1}{\tau_k})$$

and therefore

$$|\nu_k(n)| = |1 - \mu\lambda_k|^n |\nu_k(0)| = exp(-\frac{n}{\tau_k})|\nu_k(0)|$$

which is a decreasing exponential, which needs approximately $4\tau_k$ steps to reduce $\nu_k(0)$ to 2% of its value.

Thus, we have obtained

* a condition of stability,

* information about the transient behavior and speed of convergence of parameters $\nu_k(n)$.

We can transfer back to the original parameters the results of analysis: The matrix $Q = [\underline{q}_1 \ \underline{q}_2 \ \ldots \ \underline{q}_M]$ obeys $Q^H Q = QQ^H = I$ and multiplying by Q the extreme terms of the equalities

$$\underline{\nu}(n) = Q^H \underline{c}(n) = Q^H(\underline{w}(n) - \underline{w}_o)$$

$$Q\underline{\nu}(n) = \underline{w}(n) - \underline{w}_o$$

$$\underline{w}(n) = \underline{w}_o + Q\underline{\nu}(n) = \underline{w}_o + [\underline{q}_1 \; \underline{q}_2 \; \ldots \; \underline{q}_M]\underline{\nu}(n) = \underline{w}_o + \sum_{k=1}^{M} \underline{q}_k \nu_k(n)$$

Writing componentwise the equality, we have

$$w_i(n) = w_{o,i} + \sum_{k=1}^{M} q_{k,i}\nu_k(0)(1-\mu\lambda_k)^n$$

The fastest rate of convergence is obtained for the eigenvalue λ_k which gives the least term $|1-\mu\lambda_k|$, while the slowest rate of convergence results for the eigenvalue λ_k which gives the largest term $|1-\mu\lambda_k|$.

If we limit μ such that $0 < 1-\mu\lambda_k < 1$, then

$$\frac{-1}{\log(1-\mu\lambda_{max})} \leq \tau_a \leq \frac{-1}{\log(1-\mu\lambda_{min})}$$

Transient behavior of the Mean -squared error

In Lecture 2 we obtained the canonical form of the quadratic form which expresses the mean square error:

$$J_{\underline{w}(n)} \;\; = \;\; J_{\underline{w}_o} + \sum_{i=1}^{M} \lambda_i |\nu_i|^2$$

where $\underline{\nu}$ was defined as

$$\underline{\nu}(n) = Q^H \underline{c}(n) = Q^H(\underline{w}(n) - \underline{w}_o) \tag{9}$$

Taking into account that

$$\nu_k(n) = (1-\mu\lambda_k)^n \nu_k(0) \qquad k = 1, 2, \ldots, M$$

we have finally

$$J_{\underline{w}(n)} = J_{\underline{w}_o} + \sum_{i=1}^{M} \lambda_i (1 - \mu\lambda_i)^{2n} |\nu_i(0)|^2$$

The convergence

$$J_{\underline{w}(n)} \rightarrow J_{\underline{w}_o} \tag{10}$$

takes place under the same conditions as parameter convergence, and the rate of convergence is bounded, similarly, by

$$\frac{-1}{2\log(1 - \mu\lambda_{max})} \leq \tau_a \leq \frac{-1}{2\log(1 - \mu\lambda_{min})} \tag{11}$$