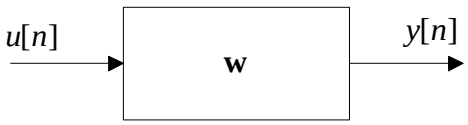
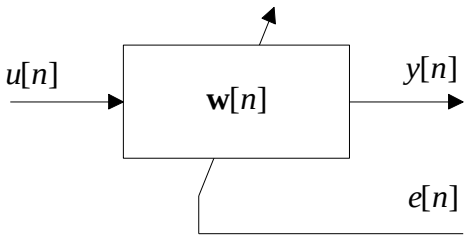
Adaptive Filters

- Statistical digital signal processing: in many problems of interest, the signals exhibit some inherent variability plus additive noise and we use probabilistic laws to model the statistical variability.
- In adaptive filtering, statistics are not known and must be inferred from data itself.
- Fixed filter (FIR):



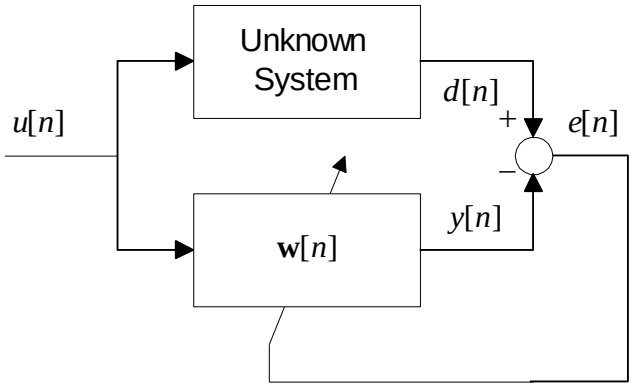
$$y[n] = \mathbf{w}^H \mathbf{u}[n] = \sum_{k=0}^{M-1} w_k^* u[n-k]$$

- Adaptive filter (FIR):

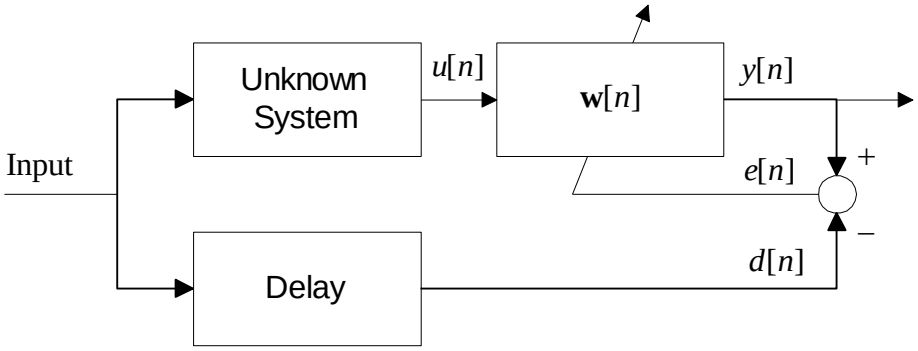


$$y[n] = \mathbf{w}^H[n]\mathbf{u}[n] = \sum_{k=0}^{M-1} w_k^*[n] u[n-k]$$

(1) Identification

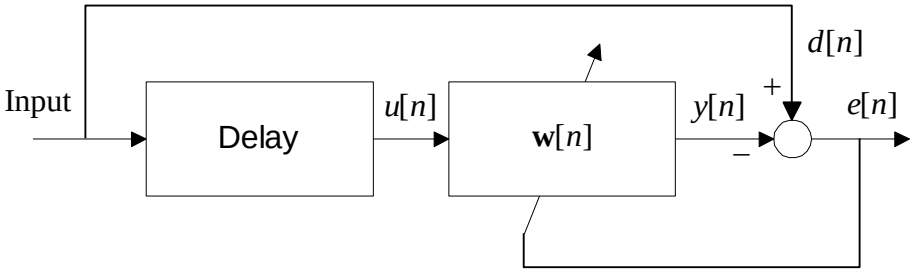


(2) Inverse modeling



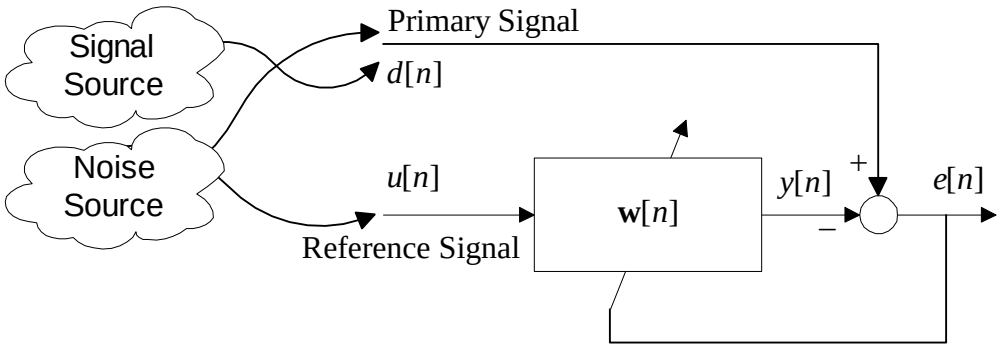
- Predictive deconvolution
- Adaptive equalization

(3) Prediction



- Linear predictive coding
- Spectrum estimation
- Signal detection

(4) Interference cancellation

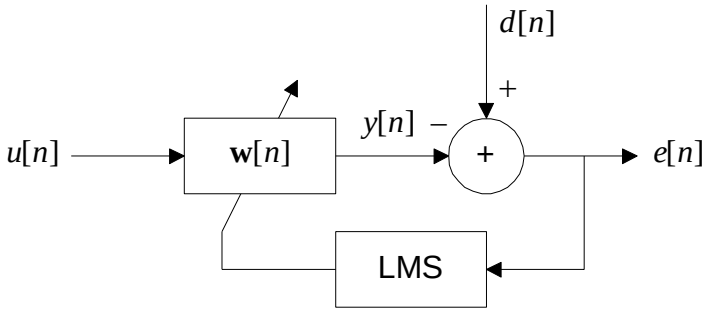


- Adaptive noise cancellation
- Echo cancellation
- Adaptive beam forming

Adaptive Filter using LMS (Least Mean Square) Algorithm

Unknown statistics ⇒ adaptive (learning) algorithm, iterative algorithm
Stochastic steepest descent algorithm

(1) Formulation



Let $\mathbf{w}[n]=\left[w_0[n]w_1[n]\cdots w_{M-1}[n]\right]^T\in\mathbf{C}^M$, $\mathbf{u}[n]=\left[u[n]u[n-1]\cdots u[n-M+1]\right]^T$,

and $y[n]=\mathbf{w}^H[n]\mathbf{u}[n]$ and $e[n]=d[n]-y[n]$. Define

$$J(\mathbf{w}[n])=E\left\{\left|e[n]\right|^2\right\}=E\left\{\left|d[n]-\mathbf{w}^H[n]\mathbf{u}[n]\right|^2\right\} \quad \text{and} \quad \mathbf{w}_o=\arg\min_{\mathbf{w}[n]\in\mathbf{C}^M} J(\mathbf{w}[n])$$

Since we do not know the statistics, we take the estimate of $J(\mathbf{w}[n])$ by its instantaneous value as follows.

$$\begin{aligned}\hat{J}(n;\mathbf{w}[n]) &= \left|e[n]\right|^2 \\ &= \left(d[n]-\mathbf{w}^H[n]\mathbf{u}[n]\right)\left(d[n]-\mathbf{w}^H[n]\mathbf{u}[n]\right)^H \\ &= d^2[n]-d[n]\mathbf{u}^H[n]\mathbf{w}[n]-d^*[n]\mathbf{w}^H[n]\mathbf{u}[n]+\mathbf{w}^H[n]\mathbf{u}[n]\mathbf{u}^H[n]\mathbf{w}[n]\end{aligned}$$

The goal is to minimize $\hat{J}(n;\mathbf{w}[n])$ by a suitable choice of $\mathbf{w}[n]$.

In LMS algorithm, we use the steepest descent algorithm.

$$\nabla\hat{J}(n;\mathbf{w}[n])=\frac{\partial\hat{J}(n;\mathbf{w}[n])}{\partial\mathbf{w}[n]}=2\left(\mathbf{u}[n]\mathbf{u}^H[n]\mathbf{w}[n]-\mathbf{u}[n]d^*[n]\right)$$

and

$$\begin{aligned}\mathbf{w}[n+1] &= \mathbf{w}[n] - \frac{1}{2}\mu\nabla\hat{J}(n;\mathbf{w}[n]) \\ &= \mathbf{w}[n] + \mu\left(\mathbf{u}[n]d^*[n] - \mathbf{u}[n]\mathbf{u}^H[n]\mathbf{w}[n]\right) \\ &= \mathbf{w}[n] + \mu\mathbf{u}[n]\left(d^*[n] - \mathbf{u}^H[n]\mathbf{w}[n]\right) \\ &= \mathbf{w}[n] + \mu\mathbf{u}[n]e^*[n]\end{aligned}$$

(2) Algorithm

- Filter output, $y[n] = \mathbf{w}^H\mathbf{u}[n]$
- Estimate error, $e[n] = d[n] - y[n]$
- Update filter, $\mathbf{w}[n+1] = \mathbf{w}[n] + \mu\mathbf{u}[n]e^*[n]$
- Repeat

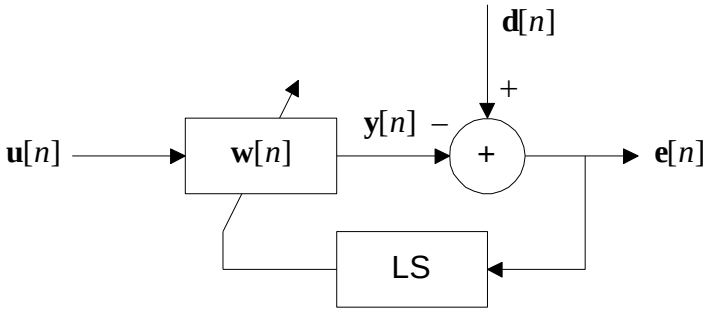
(3) Implementation

- Number of taps, M
- Step size parameter, μ
  - $0 < \mu < \dfrac{2}{\text{tap-input power}}$ with tap-input power $= \sum_{k=0}^{M-1} E\left\{\left|u[n-k]\right|^2\right\}$
  - $0 < \mu < \dfrac{2}{\lambda_{\max}}$ with $\lambda_{\max}$ is the largest eigenvalue of $\mathbf{R}$
  - $0 < \mu < \dfrac{2}{\text{tr}(\mathbf{R})}$

Adaptive Filter using LS (Least Square) Algorithm

We approximate ensemble average with time average.

(1) Formulation



Let $\mathbf{w}[n]=\left[w_0[n]w_1[n]\cdots w_{M-1}[n]\right]^T\in\mathbf{C}^M$, $\mathbf{u}[n]=\left[u[n]u[n-1]\cdots u[n-M+1]\right]^T$, $y[n]=\mathbf{w}^H[n]\mathbf{u}[n]$, and $e[n]=d[n]-y[n]$. Define

$$J(\mathbf{w}[n])=E\left\{\left|e[n]\right|^2\right\}=E\left\{\left|d[n]-\mathbf{w}^H[n]\mathbf{u}[n]\right|^2\right\} \quad \text{and} \quad \mathbf{w}_o=\arg\min_{\mathbf{w}[n]\in\mathbf{C}^M}J(\mathbf{w}[n])$$

We take the estimate of $J(\mathbf{w}[n])$ by its time average value as follows.

$$\hat{J}(n;\mathbf{w}[n])=\ \mathcal{E}(N)=\sum_{n=M}^{N}\left|e[n]\right|^2$$

where $N>M$. For stationary process, $\frac{\mathcal{E}(N)}{N}\xrightarrow[N\to\infty]{}E\left\{\left|e[n]\right|^2\right\}$. Assume $\mathbf{w}[n]=\mathbf{w}=\mathbf{w}_o$ for all n.

We vectorize the signals as follows. Let $\mathbf{d}=\left[d[M]d[M+1]\cdots d[N]\right]^H$ and $\mathbf{e}=\left[e[M]\ e[M+1]\cdots e[N]\right]^T$. Let

$$\mathbf{y}=\left[y[M]y[M+1]\cdots y[N]\right]^H=\left[\mathbf{w}^H\mathbf{u}[M]\mathbf{w}^H\mathbf{u}[M+1]\cdots\mathbf{w}^H\mathbf{u}[N]\right]^H,$$

then

$$\mathbf{y}^H=\mathbf{w}^H\left[\mathbf{u}[M]\mathbf{u}[M+1]\cdots\mathbf{u}[N]\right]=\mathbf{w}^H\mathbf{A}^H$$

with $\mathbf{y}=\mathbf{A}\mathbf{w}$ where $\mathbf{y}$ is $N\times1$, $\mathbf{w}$ is $M\times1$, and $\mathbf{A}$ is $N\times M$. Assume $\mathbf{A}$ is full rank matrix (i.e. rank($\mathbf{A}$) = M). Then, $\mathbf{y}\in Span(\mathbf{A})$.

In order to minimize $\mathcal{E}(N) = \mathbf{e}^H\mathbf{e} = \|\mathbf{e}\|^2 = \langle \mathbf{e}, \mathbf{e} \rangle$ where $\mathbf{e} = \mathbf{d} - \mathbf{y} = \mathbf{d} - \mathbf{A}\mathbf{w}$, $\mathbf{y}$ must be the projection of $\mathbf{d}$ onto $Span(\mathbf{A})$. From the orthogonality principle (OP), $\forall \mathbf{a} \in Span(\mathbf{A})$ and $\mathbf{a} \neq \mathbf{0}$, $\mathbf{a} = \mathbf{A}\mathbf{b}$ and $\langle \mathbf{e}, \mathbf{a} \rangle = 0$. Therefore, $(\mathbf{d}^H - \mathbf{w}^H\mathbf{A}^H)\mathbf{A}\mathbf{b} = 0$. Since $\mathbf{b} \neq \mathbf{0}$, $\mathbf{w}^H\mathbf{A}^H\mathbf{A} = \mathbf{d}^H\mathbf{A}$ and $\mathbf{A}^H\mathbf{A}\mathbf{w} = \mathbf{A}^H\mathbf{d}$. Finally,

$$\mathbf{w} = \left(\mathbf{A}^H\mathbf{A}\right)^{-1}\mathbf{A}^H\mathbf{d}$$

and

$$\mathbf{y} = \mathbf{A}\left(\mathbf{A}^H\mathbf{A}\right)^{-1}\mathbf{A}^H\mathbf{d}.$$

This is the least square solution.

(2) Algorithm

- Construct the data matrix, **A**.
- Solve $\mathbf{A}^H\mathbf{A}\mathbf{w} = \mathbf{A}^H\mathbf{d}$ for **w**.
- Compute **y** = **A****w**

(3) Implementation

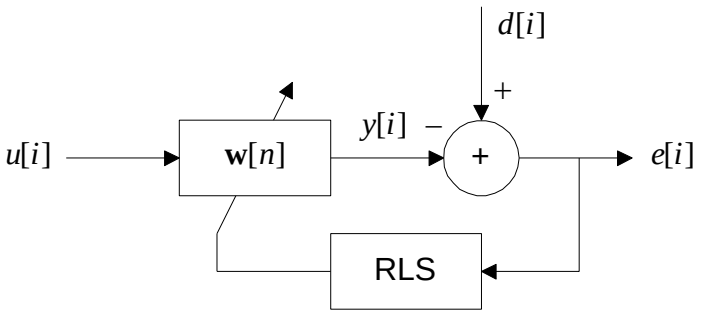
- Number of taps, *M*
- Number of data point, *N*

* What if **A** is rank deficient? What if $N < M$? In this case, we have infinitely many solutions and we want to find the minimum norm solution.

Adaptive Filter using RLS (Recursive Least Square) Algorithm

Recursive form of LS (least square) algorithm.

(1) Formulation



We take the estimate of $J(\mathbf{w}[n])$ by its time average value of the time interval $1 \le i \le n$ as follows.

$$\hat{J}(n;\mathbf{w}[n]) = \mathcal{E}(n) = \sum_{i=1}^{n} \beta(n,i) \left|e[i]\right|^2$$

where $n > M$. Let $\mathbf{w}[n] = \left[w_0[n] w_1[n] \cdots w_{M-1}[n]\right]^T \in \mathbf{C}^M$ and $\mathbf{w}[n]$ is fixed for time $1 \le i \le n$. Let $\mathbf{u}[i] = \left[u[i] u[i-1] \cdots u[i-M+1]\right]^T$, $y[i] = \mathbf{w}^H[n]\mathbf{u}[i]$, and $e[i] = d[i] - y[i]$.

The weighting factor $\beta(n,i)$ satisfies $0 < \beta(n,i) \le 1$ for $i = 1, 2, ..., n$. Especially, the exponential weighting factor or forgetting factor is defined by $\beta(n,i) = \lambda^{n-i}$ where λ is a positive constant close to 1 but less than 1. Then,

$$\mathcal{E}(n) = \sum_{i=1}^{n} \lambda^{n-i} \left|e[i]\right|^2 .$$

We define $M \times M$ correlation matrix $\Phi(n)$ at time n as

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

and define $M \times 1$ cross-correlation vector $\mathbf{z}(n)$ at time n as

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

Then, by the LS method, the solution $\hat{\mathbf{w}}(n)$ of the normal equation $\Phi(n)\hat{\mathbf{w}}(n) = \mathbf{z}(n)$

minimizes $\mathcal{E}(n)$. However, we want to recursively compute $\hat{\mathbf{w}}(i)$ for $i \geq n$.

Note that

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

and

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

Therefore, given $\Phi(n-1)$ and $\mathbf{z}(n-1)$, at time n, we update $\Phi(n)$ and $\mathbf{z}(n)$ using $\mathbf{u}(n)$ and $d^*(n)$. Then, we can compute $\hat{\mathbf{w}}(n)$ from $\Phi(n)\hat{\mathbf{w}}(n) = \mathbf{z}(n)$. The essence of RLS algorithm is to avoid matrix inversion using the matrix inversion lemma.

Matrix inversion lemma (or Woodbury's identity) is as follows. Let **A** and **B** be two positive definite $M \times M$ matrices related by

$$\mathbf{A} = \mathbf{B}^{-1} + \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^H$$

where **D** is another positive definite $N \times M$ matrix, and **C** is an $M \times N$ matrix. Then,

$$\mathbf{A}^{-1} = \mathbf{B} - \mathbf{B}\mathbf{C}\left(\mathbf{D} + \mathbf{C}^H\mathbf{B}\mathbf{C}\right)^{-1}\mathbf{C}^H\mathbf{B}.$$

We set up as follows.

$$\mathbf{A} = \Phi(n), \mathbf{B}^{-1} = \lambda\Phi(n-1), \mathbf{C} = \mathbf{u}(n), \mathbf{D} = 1.$$

Then,

$$\Phi^{-1}(n) = \lambda^{-1}\Phi^{-1}(n-1) - \frac{\lambda^{-2}\Phi^{-1}(n-1)\mathbf{u}(n)\mathbf{u}^H(n)\Phi^{-1}(n-1)}{1 + \lambda^{-1}\mathbf{u}^H(n)\Phi^{-1}(n-1)\mathbf{u}(n)}.$$

Let $\mathbf{P}(n) = \Phi^{-1}(n)$ and $\mathbf{k}(n) = \dfrac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1 + \lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)}$, then the Riccati equation for the RLS algorithm is

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1).$$

The $M \times M$ matrix $\mathbf{P}(n)$ is the inverse correlation matrix and $M \times 1$ vector $\mathbf{k}(n)$ is the gain vector. We also know

$$\begin{aligned}\mathbf{k}(n) &= \lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n) \\ &= \left[\lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)\right]\mathbf{u}(n) \\ &= \mathbf{P}(n)\mathbf{u}(n) \\ &= \Phi^{-1}(n)\mathbf{u}(n)\end{aligned}.$$

Now, at time n, $\hat{\mathbf{w}}(n) = \Phi^{-1}(n)\mathbf{z}(n) = \mathbf{P}(n)\mathbf{z}(n) = \lambda\mathbf{P}(n)\mathbf{z}(n-1) + \mathbf{P}(n)\mathbf{u}(n)d^*(n)$. Therefore,

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

where $\xi(n)$ is the a priori estimation error defined by

$$\xi(n) = d(n) - \mathbf{u}^T(n)\mathbf{w}^*(n-1) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n).$$

A posteriori estimation error is

$$e(n) = d(n) - \hat{\mathbf{w}}^H(n)\mathbf{u}(n).$$

(2) Algorithm

- Initialize the algorithm by $\mathbf{P}(0) = \delta^{-1}\mathbf{I}$ and $\mathbf{w}(0) = \mathbf{0}$ for a small positive constant δ.
- For each time n = 1, 2, ..., compute

$$\mathbf{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$

$$\xi(n) = d(n) - \hat{\mathbf{w}}^H(n-1)\mathbf{u}(n)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n)\xi^*(n)$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1)$$

(3) Implementation

- Number of taps, M
- Initialization
- Convergence in about $2M$ iterations
- Signal distortion