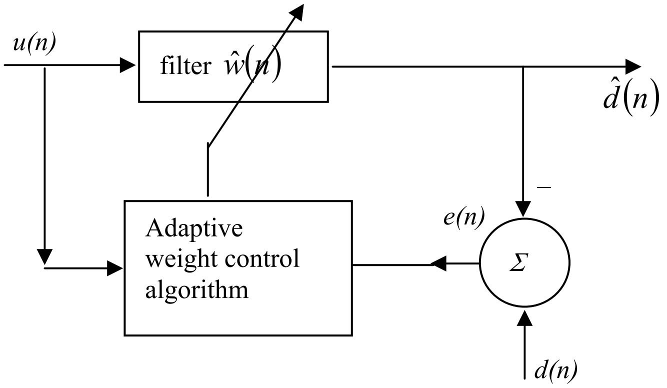
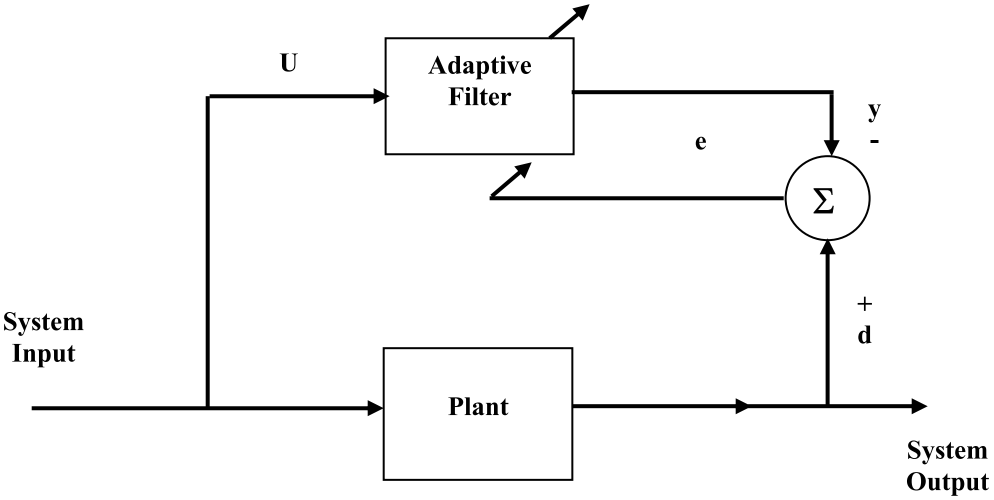
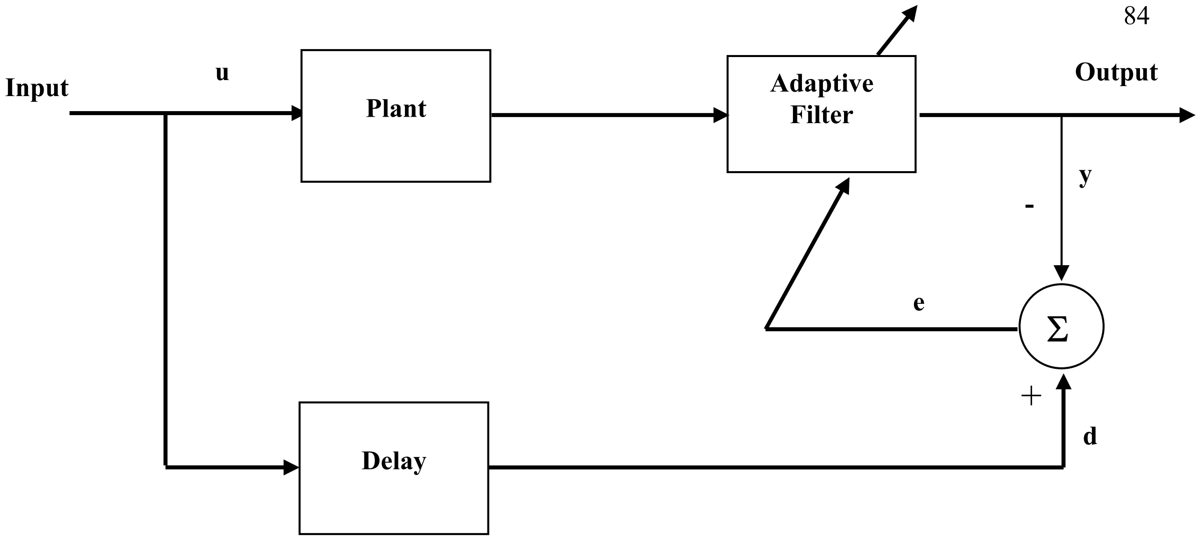
Adaptive Filter Theory



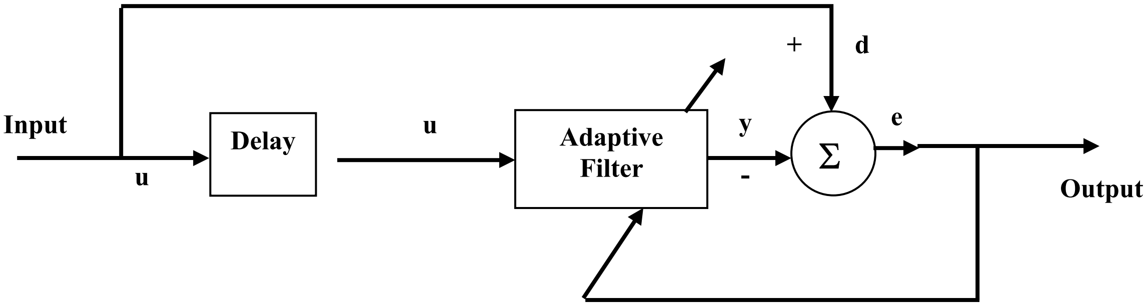
APPLICATION OF ADAPTIVE FILTERS



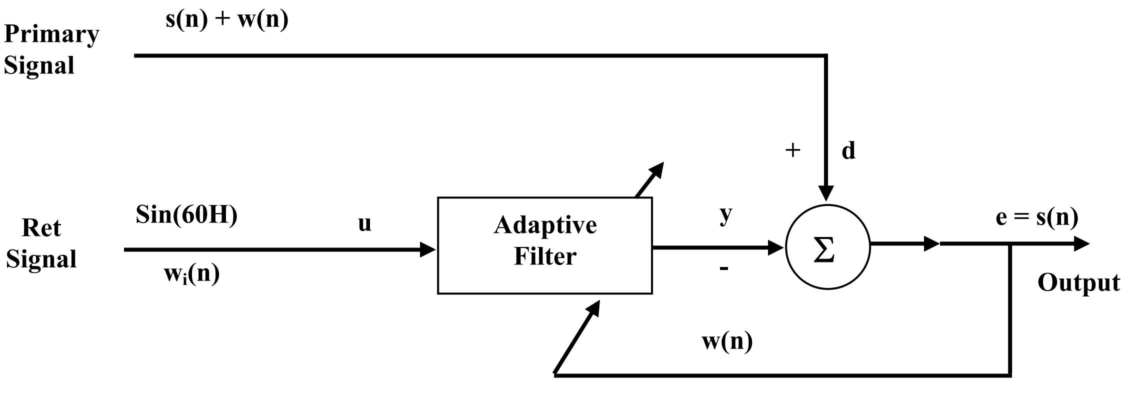
a) Identification



b) Inverse Modelling

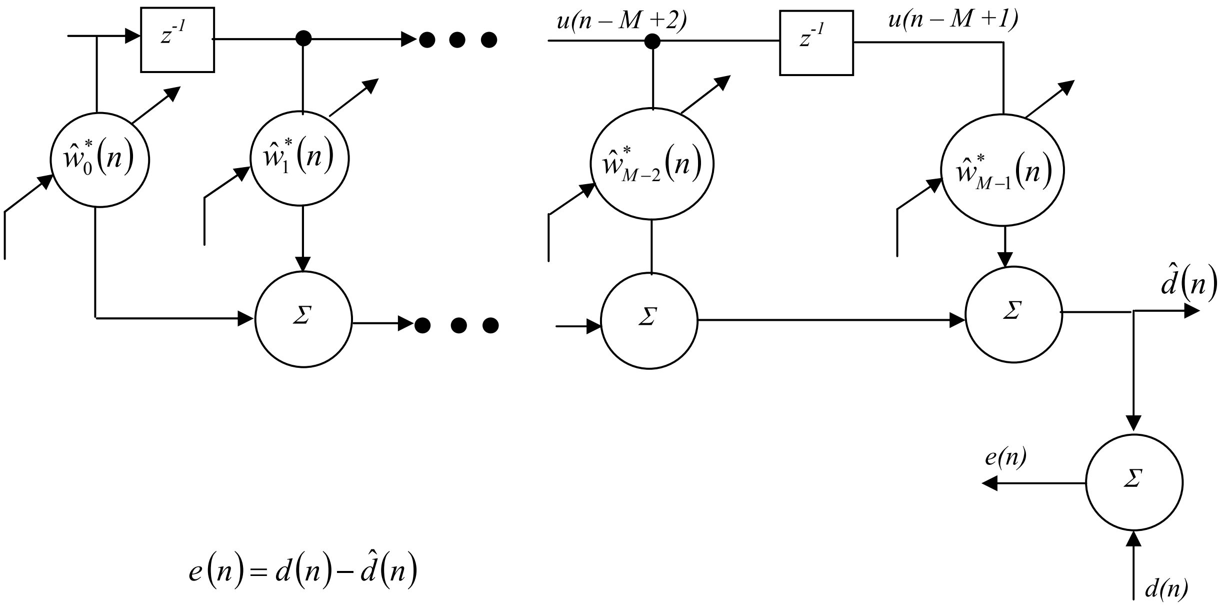


c) Prediction



d) Interference Cancellation

Least-Mean-Square (LMS) Adaptation Algorithm:



$$e(n) = d(n) - \hat{d}(n)$$

$$\hat{d}(n) = \underline{\hat{w}}^H(n)\cdot \underline{u}(n) \qquad \underline{\hat{w}}(n) = \left[\hat{w}_o(n), \hat{w}_1(n),\dots, \hat{w}_{M-1}(n)\right]^T$$

* H is the Hermitian Transposition $\qquad \underline{u}(n) = \left[u(n), u(n-1),\dots, u(n-M+1)\right]^T$

Cost function is defined as $J(n) = E\{|e(n)|^2\}$.

By taking the gradient vector (similar to Wiener-Hopf Equations) we will have:

$$\nabla J(n) = -2\underline{\gamma}_{ud} + 2\underline{\Gamma}_{uu}\cdot \underline{w}(n),$$

where $\begin{array}{ll} \underline{\gamma}_{ud} = E\{\underline{u}(n)\cdot d^*(n)\} & d(n)\ is\ a\ sample \\ \underline{\Gamma}_{uu} = E\{\underline{u}(n)\cdot \underline{u}^H(n)\} & \underline{u}(n) is\ a\ \underline{M}\times \underline{1}\ vector \end{array}$.

The simplest choice of estimators for Γ_{uu} and Υ_{du} is to use instantaneous estimates that are based on sample values of the tap-weight input vector (*u(n)*) and desired response as defined below.

$$\underline{\hat{\Gamma}}_{uu}(n) = \underline{u}(n)\cdot u^H(n) \text{ and } \hat{\gamma}_{du}(n) = \underline{u}(n)\cdot d^*(n)$$

$$\Rightarrow \hat{\nabla} J(n) = -2\underline{u}(n)\cdot d^*(n) + 2\underline{u}(n)u^H(n)\cdot \underline{\hat{w}}(n)$$

Minimizing this gradient vector by steepest-descent algorithm, leads to a recursive equation for updating the tap-weight vector:

$$\underline{\hat{w}}(n+1) = \underline{\hat{w}}(n) + \mu\,\underline{u}(n)\left[\underbrace{d^*(n) - \underline{u}^H(n)\cdot \underline{\hat{w}}(n)}_{e^*(n)\,(a\, scalar)}\right]$$

What is the steepest-Descent Algorithm?

Steepest-Descent Algorithm is one of the oldest methods of optimization. To find the minimum value of the Mean-Squared error, J_{min}, this algorithm suggests:

1) Begin with an initial value $\underline{w}(0)$ for the tap-weight vector, to provide an initial guess for minimum point of the error performance surface. $\underline{w}(0)$ is usually set equal to the null vector.
2) Using the initial guess, $\underline{w}(0)$, compute the gradient vector, $\nabla J(n)$ as the following

   let $\underline{w}(n) = [\alpha_0(n) + j\beta_0(n), \alpha_1(n) + j\beta_1(n), \ldots, \alpha_{M-1}(n) + j\beta_{M-1}(n)]^T$

   $$\text{Then } \nabla J(n) = \begin{bmatrix} \dfrac{\partial J(n)}{\partial \alpha_o(n)} + j\,\dfrac{\partial J(n)}{\partial \beta_o(n)} \\ \dfrac{\partial J(n)}{\partial \alpha_1(n)} + j\,\dfrac{\partial J(n)}{\partial \beta_1(n)} \\ \vdots \\ \dfrac{\partial J(n)}{\partial \alpha_{M-1}(n)} + j\,\dfrac{\partial J(n)}{\partial \beta_{M-1}(n)} \\ \\ \end{bmatrix} = -2\underline{\gamma}_{du}(n) + 2\underline{\Gamma}_{uu}(n) \cdot \underline{w}(n).$$

3) Compute the next guess at the tap-weight vector $\hat{\underline{w}}(n)$ by making a change in the initial or present guess in a direction opposite to that of the gradient vector.

   $\underline{w}(n+1) = \underline{w}(n) + \frac{1}{2}\mu \cdot (-\nabla J(n))$, where μ is a positive real-valued constant. The factor ½ is used merely for the purpose of canceling a factor *2* that appears in the formula for $\nabla J(n)$.

   $$\underline{w}(n+1) = \underline{w}(n) + \mu\lfloor \underline{\gamma}_{ud}(n) - \underline{\Gamma}_{uu}(n)\underline{w}(n) \rfloor \text{ or}$$

   $$\hat{\underline{w}}(n+1) = \hat{\underline{w}}(n) + \mu\lfloor \underline{u}(n) \cdot d^*(n) - \underline{u}(n)\, \underline{u}^H(n) \cdot \hat{\underline{w}}(n) \rfloor = \hat{\underline{w}}(n) + \mu \cdot \underline{u}(n) \cdot e^*(n)$$

We observe that μ controls the size of incremental correction applied to the tap-weight vector as we proceed from one iteration cycle to the next.

How μ should be chosen?

Stability and convergence analysis of the Steepest-Descent Algorithm, gives the criterion that μ must satisfy this condition: $o < \mu < \frac{2}{\lambda_{\max}}$, where λ_{max} is the largest eigen value of the correlation matrix $\underline{\Gamma}$ of the input. In practical applications that knowledge of $\underline{\Gamma}$ and λ_{max} is not available, μ is chosen as $o < \mu < \frac{2}{tr[\underline{\Gamma}]}$. $tr[\underline{\Gamma}]$ is the trace of matrix $\underline{\Gamma}$: $tr[\underline{\Gamma}] = \sum_{i=1}^{M} \lambda_i$. We may go one step further by noting that the auto-correlation matrix $\underline{\Gamma}$ is not only positive definite, but also a

Teoplitz matrix with its main diagonal equal to *Υ(0)*. Since $\gamma(0) = \sigma_u^2$ is itself equal to mean-squared value of the input at each of the *M* taps of the filter, then we have:

$$tr[\underline{\Gamma}] = M \cdot \gamma_{uu}(0) = \sum_{k=0}^{M-1} E\left[|u(n-k)|^2\right]$$
$$= tap - input \;\; power$$

Notes:

With a small value of μ, adaptation is slow, but the MSE after adaptation is small too. On the other hand, when μ is large, the adaptation is relatively fast but at the cost of an increase in MSE after adaptation. Thus, μ may be viewed as the "memory" of the LMS algorithm.

Summary of the LMS Algorithm

Parameters: M = # of taps

μ = step-size parameter

$$o < \mu < \frac{2}{tap - input \;\; power} = \frac{2}{\sum_{k=0}^{M-1} E\left[|u(n-k)|^2\right]}$$

Initialization: $\hat{\underline{w}}(0) = \underline{0}$ unless there is a prior knowledge.

Given: $\underline{u}(n) = M$ - by -1 tap input vector at time *n*.

d(n) = desired response time at *n*

To be computed:

$\hat{\underline{w}}(n+1)$ = estimate of tap-weight vector at time *n* + *1*

Computation:

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

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

Normalized LMS Algorithm

The moralized LMS algorithm may be viewed as the solution to a constrained optimization (minimization) problem. Specifically the problem of interest may be stated as follows:

Given the tap-input vector *u(n)* and the desired response *d(n)*, determine the tap-weight vector $\hat{w}(n+1)$ so as to minimize the squared Euclidean norm of the change, $\delta\hat{\underline{w}}(n+1) = \hat{\underline{w}}(n+1) - \hat{\underline{w}}(n)$ subject to the constraint: $\hat{\underline{w}}^H(n+1)\underline{u}(n) = d(n)$.

Lagrange multipliers are used to solve this problem, and the result is called Normalized LMS Algorithm.

Summary of Normalized Algorithm

M = # of taps

$\hat{\mu}$ = adaptation constant, $o < \hat{\mu} < 2$

a = a small positive constant

Initialization: $\underline{\hat{w}}(0) = \underline{0}$

Given: *u(n)*: *M* by 1 tap-input vector at time *n*

d(n): desired response at time *n*

Compute:

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

$$\underline{\hat{w}}(n+1) = \underline{\hat{w}}(n) + \frac{\hat{\mu}}{a + \left\|\underline{u}(n)\right\|^{2}} \cdot \underline{u}(n) e^{*}(n)$$

As you see, $\hat{\mu}$ is divided by the norm of *u(n)* and hence, the name "Normalized". In case of having very small input, numerical difficulties may arise due to the division to $\|\, \underline{u}(n) \|^{2}$, then we may define $\mu(n) = \frac{\hat{\mu}}{\left\|\underline{u}(n)\right\|^{2}}$ and in this light Normalized LMS may be viewed as LMS with a time-varying step-size parameter. The rate of convergence of the Normalized LMS is faster than the conventional LMS Algorithm.

Recursive Least Squares Algorithm (RLS)

$$e(i) = d(i) - \underline{w}^H(n) \cdot \underline{u}(i) \qquad 1 \le i \le n$$

$$\underline{u}(i) = [u(i), u(i-1), \ldots, u(i-M+1)]^T$$

$$\underline{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T$$

Note that here the tap-weights of the filter remain fixed during the observation interval $1 \le i \le n$ for which the cost function $\varepsilon(n)$ is defined as: $\varepsilon(n) = \sum_{i=1}^{n} \lambda^{n-i} |e(i)|^2$ where λ is a positive constant close to but less than 1. When $\lambda = 1$, we have the ordinary method of least squares. λ is called the "forgetting factor". The optimum value of the tap-weight vector $\underline{\hat{w}}(n)$, for which the cost function $\varepsilon(n)$ attains its minimum value is defined by the normal equations as the following:

$$\underline{\phi}(n) \cdot \underline{\hat{w}}(n) = \underline{z}(n), \left(\Rightarrow \underline{\hat{w}}(n) = \underline{\phi}^{-1}(n) \cdot \underline{z}(n) \right)$$

where $\underline{\phi}(n) = \sum_{i=1}^{n} \lambda^{n-i} \underline{u}(i) \underline{u}^H(i)$ M by M correlation Matrix

and $\underline{z}(n) = \sum_{i=1}^{n} \lambda^{n-i} \underline{u}(i) d^*(i)$ M by 1 cross-correlation vector

Taking the term corresponding to $i = n$ from $\underline{\phi}(n)$ equation, it can be written as:

$$\underline{\phi}(n) = \lambda \left[\underbrace{\sum_{i=1}^{n-1} \lambda^{n-i-1} \underline{u}(i) \underline{u}^H(i)}_{\underline{\phi}(n-1)} \right] + \underline{u}(n) \underline{u}^H(n)$$

$$\Rightarrow \underline{\phi}(n) = \lambda \underline{\phi}(n-1) + \underline{u}(n) \underline{u}^H(n)$$

With the same method $\underline{z}(n)$ can also be written as

$$\underline{z}(n) = \lambda(n-1) + \underline{u}(n) d^*(n)$$

Matrix Inversion Lemma

Let A and B be positive definite M-by-M matrices related by $A = B^{-1} + CD^{-1}C^H$, where D is another positive definite N-by-M matrix and C is a M-by-N matrix. Then according to this, Lemma: $A^{-1} = B - BC(D + C^H BC)^{-1} C^H \cdot B$.

Now let $A = \phi(n), B = \lambda\phi(n-1), C = \underline{u}(n)\ D = I$, then the $\phi^{-1}(n)$ can be found. Let $P = \phi^{-1}(n)$. Then

$$\underline{P}(n) = \lambda^{-1}\underline{P}(n-1) - \lambda^{-1}\underline{K}(n)\underline{u}^H(n)\underline{P}(n-1),$$

where
$$\underline{K}(n) = \frac{\lambda^{-1}\underline{P}(n-1)\underline{u}(n)}{1+\lambda\,\underline{u}^H(n)\underline{P}(n-1)\underline{u}(n)}$$

Continuing working on these equations leads to RLS Algorithm:

RLS Algorithm

M = number of taps and λ = forgetting factor $\underline{<} 1$

Given $u(n)$: M-by-1 tap-input vector

$d(n)$: desired response.

Initialize: $\underline{P}(0) = \sigma^{-1} \cdot I$, σ = small positive constant (i.e. 0.25)

$\underline{\hat{w}}(0) = \underline{0}$

For each instant of time, $n = 1, 2, \ldots.,$ compute:

$$\underline{K}(n) = \frac{\lambda^{-1}\underline{P}(n-1)\underline{u}(n)}{1+\lambda^{-1}\underline{u}^H(n)\cdot\underline{P}(n-1)\underline{u}(n)}$$

$$e(n) = d(n) - \underline{\hat{w}}^H(n-1)\cdot\underline{u}(n)$$

$$\underline{\hat{w}}(n) = \underline{\hat{w}}(n-1) + \underline{K}(n)e^*(n)$$

$$\underline{P}(n) = \lambda^{-1}\,\underline{P}(n-1) - \lambda^{-1}\underline{K}(n)\cdot\underline{u}^H(n)\cdot\underline{P}(n-1)$$

A few key features of the RLS Algorithm

- The mean of the learning curve of the RLS algorithm converges in about $2M$ iterations, where M is the number of taps of the filter. Therefore, the RLS convergence rate is much faster than that of LMS Algorithm.
- As the number of iterations, n, approaches infinity, the mean-squared error of RLS approaches a final value equal to the variance σ^2 of the measurement error. In other words, in theory, RLS algorithm produce zero error as $n \to \infty$.
- Convergence of the RLS algorithm in the mean-squared sense is independent of the eigen value spread of the correlation matrix of the input vector.