Lecture 5: Variants of the LMS algorithm

Standard LMS Algorithm

FIR filters:

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

Error between filter output $y(t)$ and a desired signal $d(t)$:

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

Change the filter parameters according to

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n) e(n)$$

1. Normalized LMS Algorithm

Modify at time n the parameter vector from $\underline{w}(n)$ to $\underline{w}(n+1)$

- fulfilling the constraint

$$\underline{w}^T(n+1)\underline{u}(n) = d(n)$$

- with the “least modification” of $\underline{w}(n)$, i.e. with the least Euclidian norm of the difference

$$\underline{w}(n+1) - \underline{w}(n) = \delta \underline{w}(n+1)$$

Thus we have to minimize

$$\|\delta \underline{w}(n+1)\|^2 = \sum_{k=0}^{M-1} (w_k(n+1) - w_k(n))^2$$

under the constraint

$$\underline{w}^T(n+1)\underline{u}(n) = d(n)$$

The solution can be obtained by Lagrange multipliers method:

$$\begin{aligned} J(\underline{w}(n+1), \lambda) &= \|\delta \underline{w}(n+1)\|^2 + \lambda \left(d(n) - \underline{w}^T(n+1)\underline{u}(n)\right) \\ &= \sum_{k=0}^{M-1} (w_k(n+1) - w_k(n))^2 + \lambda \left(d(n) - \sum_{i=0}^{M-1} w_i(n+1)u(n-i)\right) \end{aligned}$$

To obtain the minimum of $J(\underline{w}(n+1), \lambda)$ we check the zeros of criterion partial derivatives:

$$\begin{aligned} \frac{\partial J(\underline{w}(n+1), \lambda)}{\partial w_j(n+1)} &= 0 \\ \frac{\partial}{\partial w_j(n+1)} \left[\sum_{k=0}^{M-1} (w_k(n+1) - w_k(n))^2 + \lambda \left(d(n) - \sum_{i=0}^{M-1} w_i(n+1)u(n-i)\right)\right] &= 0 \\ 2(w_j(n+1) - w_j(n)) - \lambda u(n-j) &= 0 \end{aligned}$$

We have thus

$$w_j(n+1) = w_j(n) + \frac{1}{2}\lambda u(n-j)$$

where λ will result from

$$d(n) = \sum_{i=0}^{M-1} w_i(n+1)u(n-i)$$

$$\begin{aligned}
d(n) &= \sum_{i=0}^{M-1}(w_i(n)+\frac{1}{2}\lambda u(n-i))u(n-i)\\
d(n) &= \sum_{i=0}^{M-1} w_i(n)u(n-i)+\frac{1}{2}\lambda\sum_{i=0}^{M-1}(u(n-i))^2\\
\lambda &= \frac{2(d(n)-\sum_{i=0}^{M-1} w_i(n)u(n-i))}{\sum_{i=0}^{M-1}(u(n-i))^2}\\
\lambda &= \frac{2e(n)}{\sum_{i=0}^{M-1}(u(n-i))^2}
\end{aligned}$$

Thus, the minimum of the criterion $J(\underline{w}(n+1),\lambda)$ will be obtained using the adaptation equation

$$w_j(n+1)=w_j(n)+\frac{2e(n)}{\sum_{i=0}^{M-1}(u(n-i))^2}u(n-j)$$

In order to add an extra freedom degree to the adaptation strategy, one constant, $\tilde{\mu}$, controlling the step size will be introduced:

$$w_j(n+1)=w_j(n)+\tilde{\mu}\frac{1}{\sum_{i=0}^{M-1}(u(n-i))^2}e(n)u(n-j)=w_j(n)+\frac{\tilde{\mu}}{\|\underline{u}(n)\|^2}e(n)u(n-j)$$

To overcome the possible numerical difficulties when $\|\underline{u}(n)\|$ is very close to zero, a constant $a>0$ is used:

$$\boxed{w_j(n+1)=w_j(n)+\frac{\tilde{\mu}}{a+\|\underline{u}(n)\|^2}e(n)u(n-j)}$$

This is the updating equation used in the Normalized LMS algorithm.

The principal characteristics of the Normalized LMS algorithm are the following:

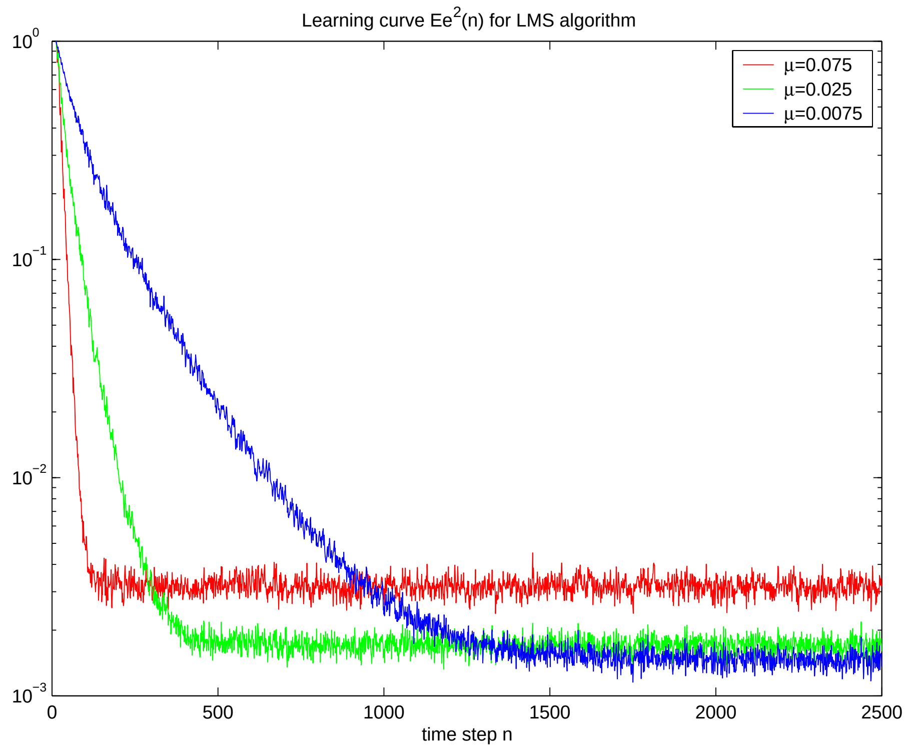
- The adaptation constant $\tilde{\mu}$ is dimensionless, whereas in LMS, the adaptation has the dimensioning of a inverse power.
- Setting
$$\mu(n) = \frac{\tilde{\mu}}{a + \|\underline{u}(n)\|^2}$$
we may vue Normalized LMS algorithm as a LMS algorithm with *data- dependent adptation step size.*
- Considering the approximate expression
$$\mu(n) = \frac{\tilde{\mu}}{a + ME(u(n))^2}$$
the normalization is such that :

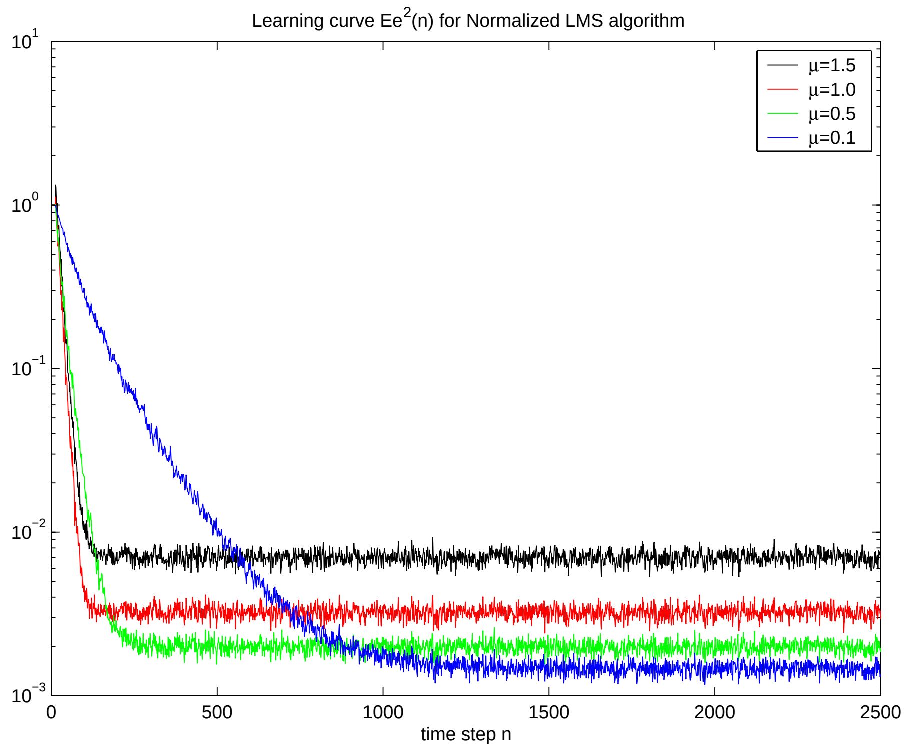
  * the effect of large fluctuations in the power levels of the input signal is compensated at the adaptation level.

  * the effect of large input vector length is compensated, by reducing the step size of the algorithm.
- This algorithm was derived based on an intuitive principle:

  > *In the light of new input data, the parameters of an adaptive system should only be disturbed in a minimal fashion.*
- The Normalized LMS algorithm is convergent in mean square sense if
$$0 < \tilde{\mu} < 2$$

Comparison of LMS and NLMS within the example from Lecture 4 (channel equalization)





Comparison of LMS and NLMS within the example from Lecture 4 (channel equalization):

- The LMS was run with three different step-sizes: $\mu = [0.075; 0.025; 0.0075]$
- The NLMS was run with four different step-sizes: $\tilde{\mu} = [1.5; 1.0; 0.5; 0.1]$
  - With the step-size $\tilde{\mu} = 1.5$, NLMS behaved definitely worse than with step-size $\tilde{\mu} = 1.0$ (slower, and with a higher steady state square error). So $\tilde{\mu} = 1.5$ is further ruled out
  - Each of the three step-sizes $\tilde{\mu} = [1.0; 0.5; 0.1]$ was interesting: on one hand, the larger the step-size, the faster the convergence. But on the other hand, the smaller the step-size, the better the steady state square error. So each step-size may be a useful tradeoff between convergence speed and stationary MSE (not both can be very good simultaneously).
- LMS with $\mu = 0.0075$ and NLMS with $\tilde{\mu} = 0.1$ achieved a similar (very good) average steady state square error. However, NLMS was faster.
- LMS with $\mu = 0.075$ and NLMS with $\tilde{\mu} = 1.0$ had a similar convergence speed (very good). However, NLMS achieved a lower steady state average square error.
- To conclude: NLMS offers better trade-offs than LMS.
- The computational complexity of NLMS is slightly higher than that of LMS.

2. LMS Algorithm with Time Variable Adaptation Step

Heuristics of the method: We combine the benefits of two different situations:

- The convergence time constant is small for large μ.
- The mean-square error in steady state is low for small μ.

Therefore, in the initial adaptation stages μ is kept large, then it is monotonically reduced, such that in the final adaptation stage it is very small.

There are many receipts of cooling down an adaptation process.

- Monotonically decreasing the step size

  $$\mu(n) = \frac{1}{n+c}$$

  Disadvantage for non-stationary data: the algorithm will not react anymore to changes in the optimum solution, for large values of n.

- Variable Step algorithm:

  $$\underline{w}(n+1) = \underline{w}(n) + M(n)\underline{u}(n)e(n)$$

  where

  $$M(n) = \begin{bmatrix} \mu_0(n) & 0 & . & 0 \\ 0 & \mu_1(n) & . & 0 \\ 0 & 0 & . & 0 \\ 0 & 0 & . & \mu_{M-1}(n) \end{bmatrix}$$

or componentwise

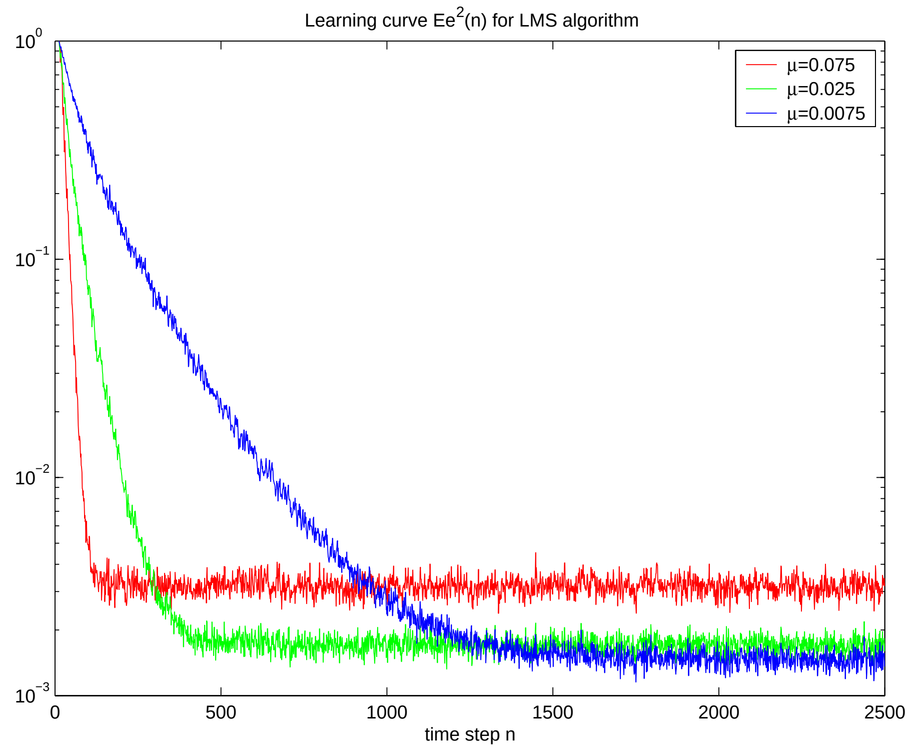
$$w_i(n+1) = w_i(n) + \mu_i(n)u(n-i)e(n) \quad i = 0, 1, \ldots, M-1$$

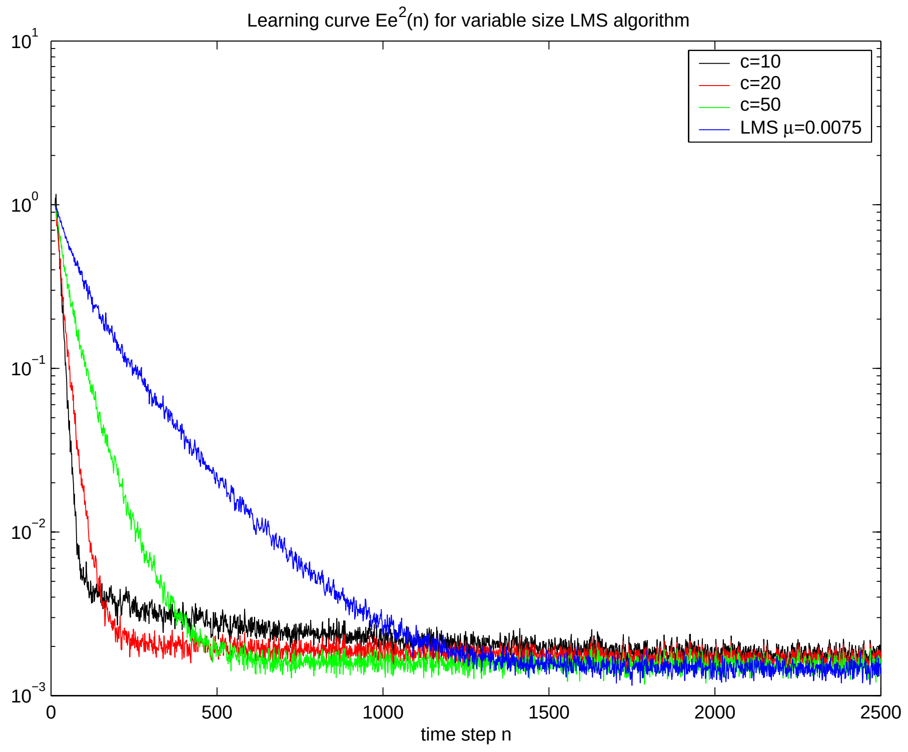
* each filter parameter $w_i(n)$ is updated using an independent adaptation step $\mu_i(n)$.

* the time variation of $\mu_i(n)$ is ad-hoc selected as

- if m_1 successive identical signs of the gradient estimate, $-e(n)u(n-i)$,are observed, then $\mu_i(n)$ is increased $\mu_i(n) = c_1\mu_i(n-m_1)$ $(c_1 > 1)$(the algorithm is still far of the optimum, is better to accelerate)
- if m_2 successive changes in the sign of gradient estimate, $-e(n)u(n-i)$, are observed, then $\mu_i(n)$ is decreased $\mu_i(n) = \mu_i(n-m_2)/c_2$ $(c_2 > 1)$ (the algorithm is near the optimum, is better to decelerate; by decreasing the step-size, so that the steady state error will finally decrease).

Comparison of LMS and variable size LMS ($\tilde{\mu} = \frac{\mu}{0.01n+c}$)within the example from Lecture 4 (channel equalization) with $c = [10; 20; 50]$





3. Sign algorithms

In high speed communication the time is critical, thus faster adaptation processes is needed.

$$sgn(a) = \begin{cases} 1; & a > 0 \\ 0; & a = 0 \\ -1; & a < 0 \end{cases}$$

- The Sign algorithm (other names: pilot LMS, or Sign Error)

$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{u}(n)\ sgn(e(n))$$

- The Clipped LMS (or Signed Regressor)

$$\underline{w}(n+1) = \underline{w}(n) + \mu\ sgn(\underline{u}(n))e(n)$$

- The Zero forcing LMS (or Sign Sign)

$$\underline{w}(n+1) = \underline{w}(n) + \mu\ sgn(\underline{u}(n))\ sgn(e(n))$$

The Sign algorithm can be derived as a LMS algorithm for minimizing the Mean absolute error (MAE) criterion

$$J(\underline{w}) = E[|e(n)|] = E[|d(n) - \underline{w}^T \underline{u}(n)|]$$

(I propose this as an exercise for you).

Properties of sign algorithms:

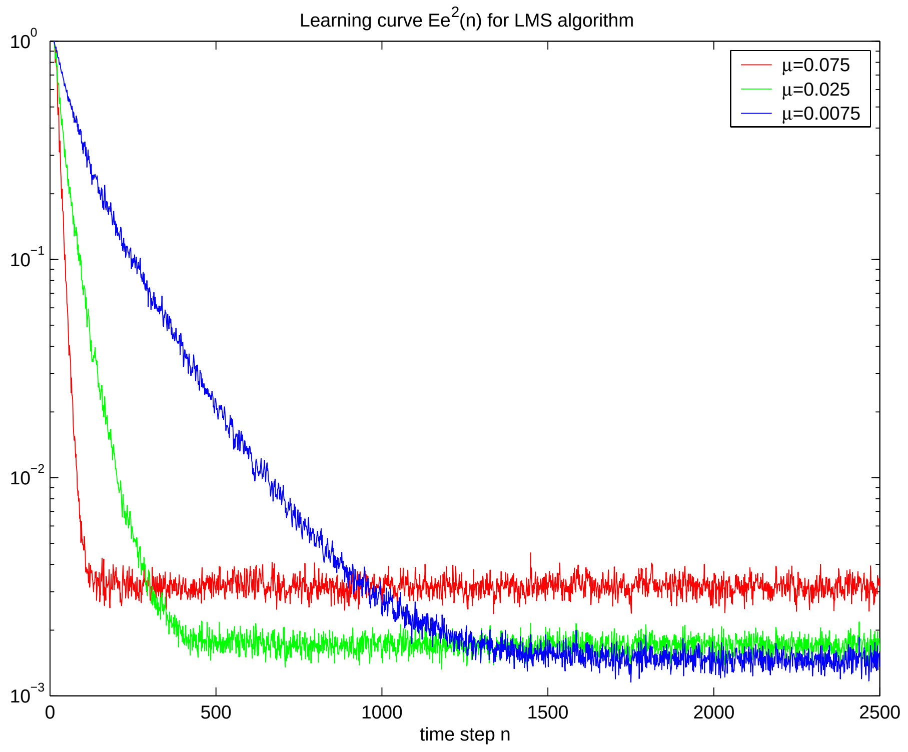
- Very fast computation : if μ is constrained to the form $\mu = 2^m$, only shifting and addition operations are required.
- Drawback: the update mechanism is degraded, compared to LMS algorithm, by the crude quantization of gradient estimates.

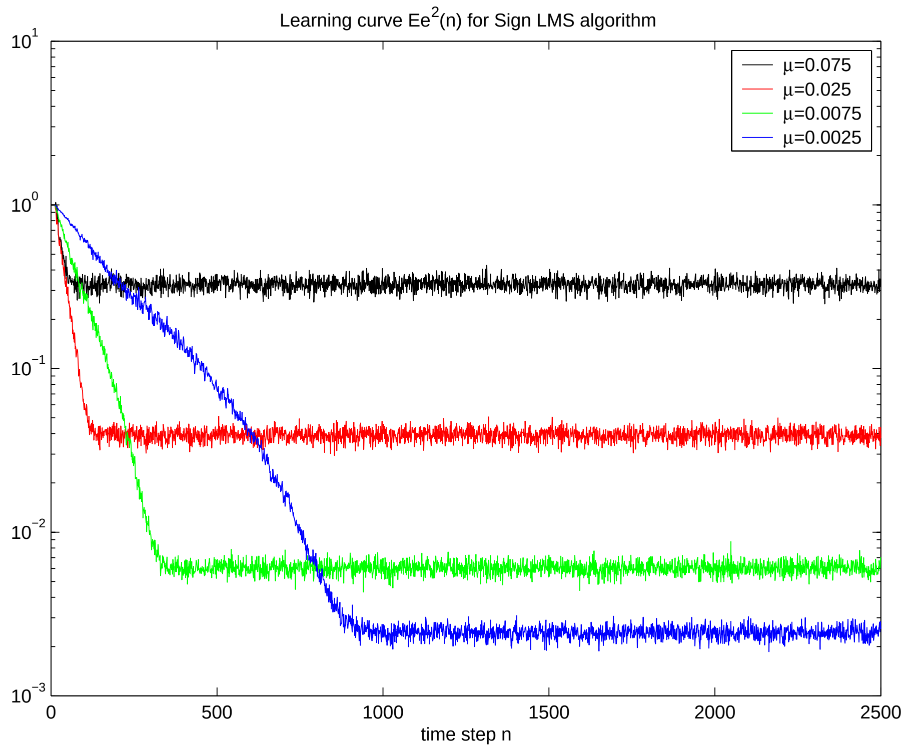
  * The steady state error will increase

  * The convergence rate decreases

The fastest of them, Sign-Sign, is used in the CCITT ADPCM standard for 32000 bps system.

Comparison of LMS and Sign LMS within the example from Lecture 4 (channel equalization)





Sign LMS algorithm should be operated at smaller step-sizes to get a similar behavior as standard LMS algorithm.

4. Linear smoothing of LMS gradient estimates

- Lowpass filtering the noisy gradient

  Let us rename the noisy gradient $\underline{g}(n) = \hat{\nabla}_{\underline{w}} J$

$$\begin{aligned} \underline{g}(n) &= \hat{\nabla}_{\underline{w}} J = -2\underline{u}(n)e(n) \\ g_i(n) &= -2e(n)u(n-i) \end{aligned}$$

  Passing the signals $g_i(n)$ through low pass filters will prevent the large fluctuations of direction during adaptation process.

$$b_i(n) = LPF(g_i(n))$$

  where LPF denotes a low pass filtering operation. The updating process will use the filtered noisy gradient

$$\underline{w}(n+1) = \underline{w}(n) - \mu \underline{b}(n)$$

  The following versions are well known:

- Averaged LMS algorithm

  When LPF is the filter with impulse response $h(0) = \frac{1}{N}, \ldots, h(N-1) = \frac{1}{N}, h(N) = h(N+1) = \ldots = 0$ we obtain simply the average of gradient components:

$$\underline{w}(n+1) = \underline{w}(n) + \frac{\mu}{N} \sum_{j=n-N+1}^{n} e(j)\underline{u}(j)$$

- Momentum LMS algorithm

  When LPF is an IIR filter of first order $h(0) = 1 - \gamma, h(1) = \gamma h(0), h(2) = \gamma^2 h(0), \ldots$ then,

$$\begin{aligned} b_i(n) &= LPF(g_i(n)) = \gamma b_i(n-1) + (1-\gamma) g_i(n) \\ \underline{b}(n) &= \gamma \underline{b}(n-1) + (1-\gamma)\underline{g}(n) \end{aligned}$$

  The resulting algorithm can be written as a second order recursion:

$$\begin{aligned} \underline{w}(n+1) &= \underline{w}(n) - \mu \underline{b}(n) \\ \gamma \underline{w}(n) &= \gamma \underline{w}(n-1) - \gamma \mu \underline{b}(n-1) \\ \underline{w}(n+1) - \gamma \underline{w}(n) &= \underline{w}(n) - \gamma \underline{w}(n-1) - \mu \underline{b}(n) + \gamma \mu \underline{b}(n-1) \\ \underline{w}(n+1) &= \underline{w}(n) + \gamma(\underline{w}(n) - \underline{w}(n-1)) - \mu(\underline{b}(n) - \gamma \underline{b}(n-1)) \\ \underline{w}(n+1) &= \underline{w}(n) + \gamma(\underline{w}(n) - \underline{w}(n-1)) - \mu(1-\gamma)\underline{g}(n) \\ \underline{w}(n+1) &= \underline{w}(n) + \gamma(\underline{w}(n) - \underline{w}(n-1)) + 2\mu(1-\gamma) e(n) \underline{u}(n) \end{aligned}$$

$$\boxed{\underline{w}(n+1) - \underline{w}(n) = \gamma(\underline{w}(n) - \underline{w}(n-1)) + \tilde{\mu}(1-\gamma) e(n) \underline{u}(n)}$$

  Drawback: The convergence rate may decrease.

  Advantages: The momentum term keeps the algorithm active even in the regions close to minimum.

  For nonlinear criterion surfaces this helps in avoiding local minima (as in neural network learning by backpropagation)

5. Nonlinear smoothing of LMS gradient estimates

- If there is an impulsive interference in either $d(n)$ or $u(n)$, the performances of LMS algorithm will drastically degrade (sometimes even leading to instabilty).

  Smoothing the noisy gradient components using a nonlinear filter provides a potential solution.

- The Median LMS Algorithm

  Computing the median of window size $N+1$, for each component of the gradient vector, will smooth out the effect of impulsive noise. The adaptation equation can be implemented as

$$w_i(n+1) \;=\; w_i(n) - \mu \cdot med\left((e(n)u(n-i)), (e(n-1)u(n-1-i)), \ldots, (e(n-N)u(n-N-i))\right)$$

  * Experimental evidence shows that the smoothing effect in impulsive noise environment is very strong.

  * If the environment is not impulsive, the performances of Median LMS are comparable with those of LMS, thus the extra computational cost of Median LMS is not worth.

  * However, the convergence rate must be slower than in LMS, and there are reports of instabilty occurring whith Median LMS.

6. Double updating algorithm

As usual, we first compute the linear combination of the inputs

$$y(n) = \underline{w}(n)^T \underline{u}(n) \qquad (1)$$

then the error

$$e(n) = d(n) - y(n)$$

The parameter adaptation is done as in Normalized LMS:

$$\underline{w}(n+1) = \underline{w}(n) + \frac{\mu}{\|\underline{u}(n)\|^2} e(n)\underline{u}(n)$$

Now the new feature of double updating algorithm is the output estimate as

$$\hat{y}(n) = \underline{w}(n+1)^T \underline{u}(n) \qquad (2)$$

Since we use at each time instant, n, two different computations of the filter output ((1) and (2)), the name of the method is "double updating".

- One interesting situation is obtained when $\mu = 1$. Then

$$\begin{aligned}\hat{y}(n) &= \underline{w}(n+1)^T \underline{u}(n) = (\underline{w}(n)^T + \frac{\mu}{\underline{u}(n)^T \underline{u}(n)} e(n)\underline{u}(n)^T)\underline{u}(n) \\ &= \underline{w}(n)^T \underline{u}(n) + e(n) = \underline{w}(n)^T \underline{u}(n) + d(n) - \underline{w}(n)^T \underline{u}(n) = d(n)\end{aligned}$$

  Finaly, the output of the filter is equal to the desired signal $\hat{y}(n) = d(n)$, and the new error is zero $\hat{e}(n) = 0$.

  This situation is not acceptable (sometimes too good is worse than good). Why?

7. LMS Volterra algorithm

We will generalize the LMS algorithm, from linear combination of inputs (FIR filters) to quadratic combinations of the inputs.

$$\begin{aligned} y(n) &= \underline{w}(n)^T \underline{u}(n) = \sum_{k=0}^{M-1} w_k(n) u(n-k) \qquad \text{Linear combiner} \\ y(n) &= \sum_{k=0}^{M-1} w_k^{[1]}(n) u(n-k) + \sum_{i=0}^{M-1} \sum_{j=i}^{M-1} w_{i,j}^{[2]}(n) u(n-i) u(n-j) \qquad (3) \\ & \qquad \text{Quadratic combiner} \end{aligned}$$

We will introduce the input vector with dimension $M + M(M+1)/2$:

$$\underline{\psi}(n) = \begin{bmatrix} u(n) & u(n-1) & \dots & u(n-M+1) & u^2(n) & u(n)u(n-1) & \dots & u^2(n-M+1) \end{bmatrix}^T$$

and the parameter vector

$$\underline{\theta}(n) = \begin{bmatrix} w_0^{[1]}(n) & w_1^{[1]}(n) & \dots & w_{M-1}^{[1]}(n) & w_{0,0}^{[2]}(n) & w_{0,1}^{[2]}(n) & \dots & w_{M-1,M-1}^{[2]}(n) \end{bmatrix}^T$$

Now the output of the quadratic filter (3) can be written

$$y(n) = \underline{\theta}(n)^T \underline{\psi}(n)$$

and therefore the error

$$e(n) = d(n) - y(n) = d(n) - \underline{\theta}(n)^T \underline{\psi}(n)$$

is a linear function of the filter parameters (i.e. the entries of $\underline{\theta}(n)$)

Minimization of the mean square error criterion

$$J(\underline{\theta}) = E(e(n))^2 = E(d(n) - \underline{\theta}(n)^T \underline{\psi}(n))^2$$

will proceed as in the linear case, resulting in the LMS adptation equation

$$\underline{\theta}(n+1) = \underline{\theta}(n) + \mu \underline{\psi}(n) e(n)$$

Some proprties of LMS for Volterra filters are the following:

- The mean sense convergence of the algorithm is obtained when

  $$0 < \mu < \frac{2}{\lambda_{max}} \tag{4}$$

  where λ_{max} is the maximum eigenvalue of the correlation matrix

  $$R = E\underline{\psi}(n)\underline{\psi}(n)^T$$

- If the distribution of the input is symmetric (as for the Gaussian distribution), the corelation matrix has the block structure

  $$R = \begin{bmatrix} R_2 & 0 \\ 0 & R_4 \end{bmatrix}$$

  corresponding to the partition in linear and quadratic coefficients

  $$\underline{\theta}(n) = \begin{bmatrix} \underline{w}^{[1]}(n) \\ \underline{w}^{[2]}(n) \end{bmatrix} \qquad \underline{\psi}(n) = \begin{bmatrix} \underline{u}^{[1]}(n) \\ \underline{u}^{[2]}(n) \end{bmatrix}$$

and the adaptation problem decouples in two problems, i.e. the Wiener optimal filters will be solutions of

$$\begin{aligned} R_2\underline{w}^{[1]} &= Ed(n)\underline{u}^{[1]}(n) \\ R_4\underline{w}^{[2]} &= Ed(n)\underline{u}^{[2]}(n) \end{aligned}$$

and similarly, the LMS solutions can be analyzed separately, for linear, $\underline{w}^{[1]}$, and quadratic, $\underline{w}^{[2]}$, coefficients, according to the eigenvalues of R_2 and R_4.