Adaptive SP & Machine Intelligence
Linear Adaptive Filters and Applications

Danilo Mandic

room 813, ext: 46271

Department of Electrical and Electronic Engineering
Imperial College London, UK

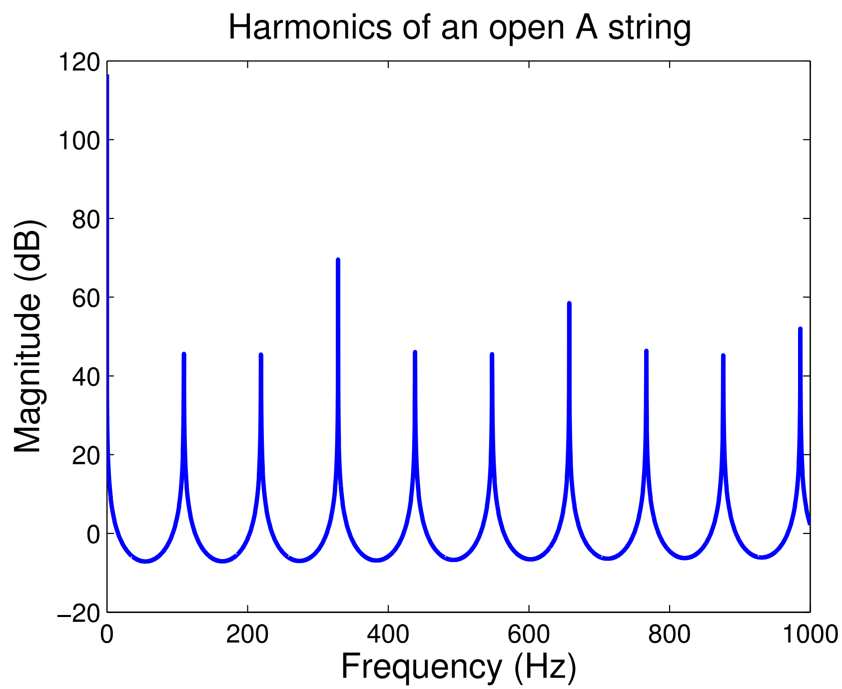
d.mandic@imperial.ac.uk, URL: www.commsp.ee.ic.ac.uk/∼mandic

Aims

- To introduce the concept of adaptive signal processing
- Parallels (duality) between spectrum estimation and adaptive filtering
- To introduce adaptive filtering architectures
- Supervised and blind adaptive filtering
- The concept of steepest descent and the Least Mean Square (LMS) algorithm
- Error surface, performance metrics, learning rate and convergence
- Fast convergence, Normalized NLMS, Generalized Normalized Gradient Descent (GNGD), gradient adaptive step-size (GASS) algorithms
- Practical applications (prediction, denoising, system ID, equalisation)

Spectrum Estimation or Digital Filtering? Let us play guitar and see Generate_Open_A_and_Play.m

Open string A has the frequency of 110 Hz, and harmonics at $k \times 110$ Hz



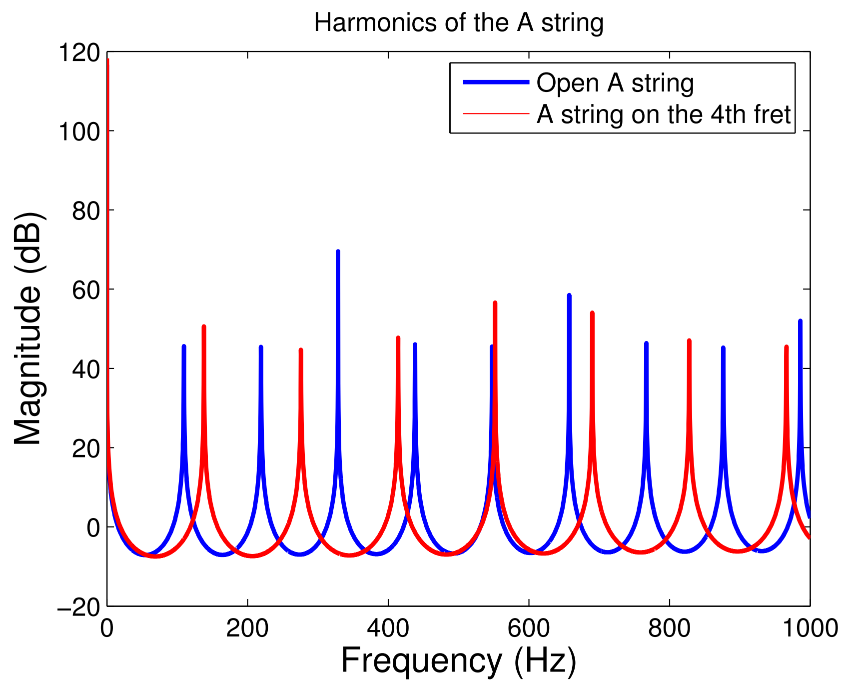
Basic frequency and Harmonics: Two ways to generate these

- ☐ Approximate the spectrum (AR Spectrum estimation, MUSIC)
- ☐ Output of an IIR filter (shown above)

Sound of string for fret four – C#

Generate_Open_A_and_Fret_Four_Play.m

C# has the frequency of $Freq_{C\#} = 2^{\frac{1}{12}} \times 110$ and harmonics at $(k \times Freq_{C\#})$ Hz

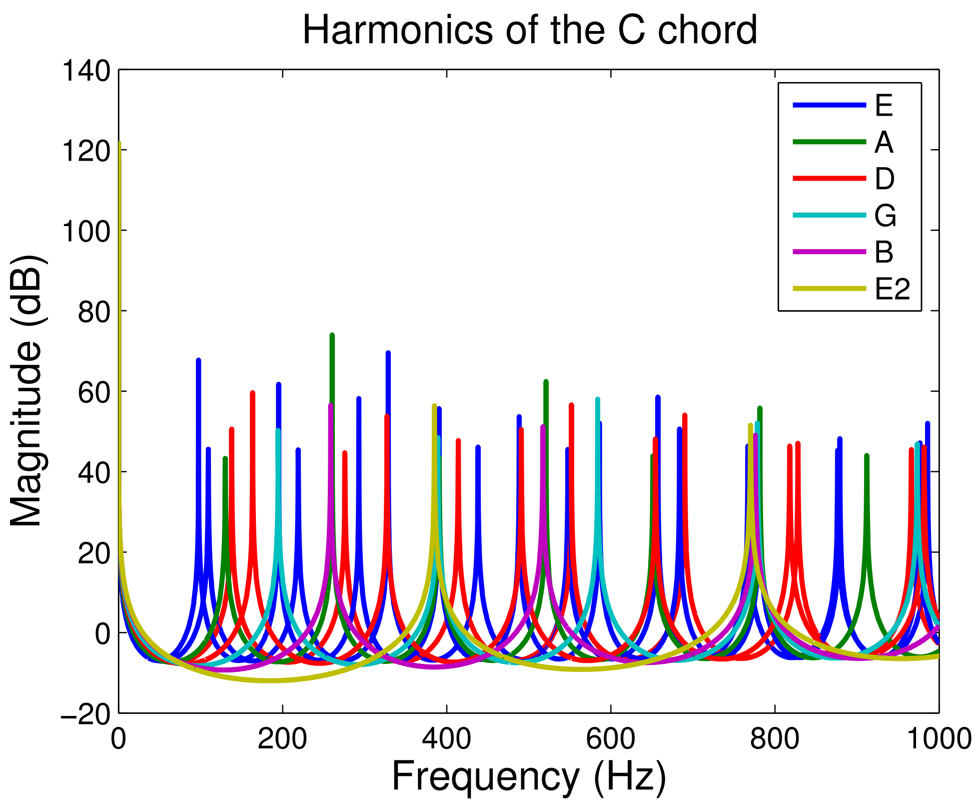


Basic frequencies and Harmonics:

- ☐ Approximate the spectrum (AR Spectrum estimation, MUSIC)?
- ☐ Output of an IIR filter?

Things getting more complicated: Chords

Generate_Chord_and_Play.m



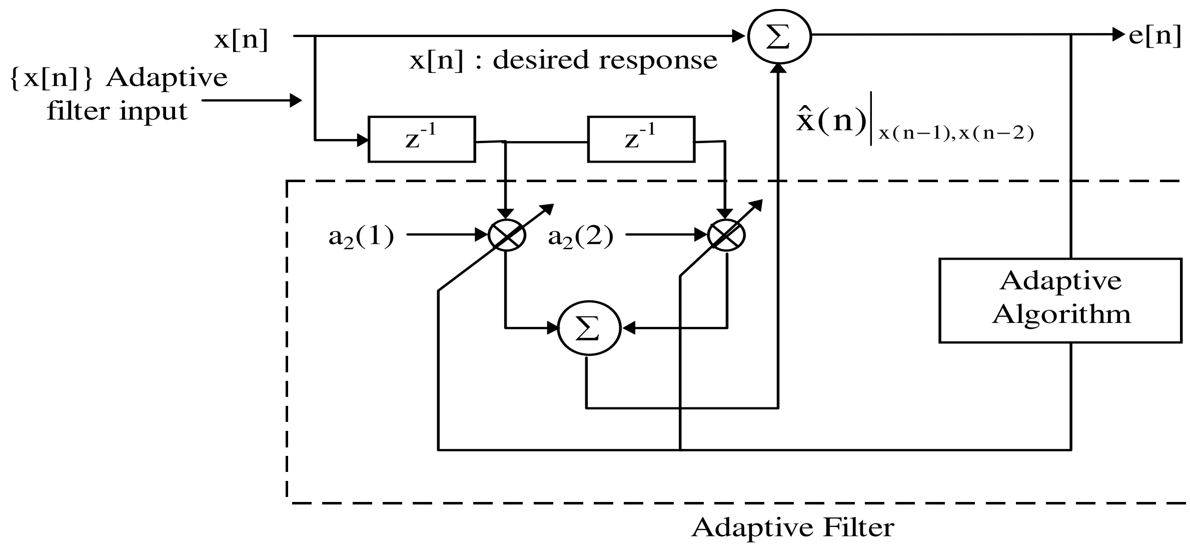
Many more parameters to estimate: Basis for music synthesis

☐ Spectrum based – resolution problems

☐ Digital filters based – filter order may become prohibitive

☞ **Can we make the spectrum estimate adaptive?**

Supervised adaptive filters



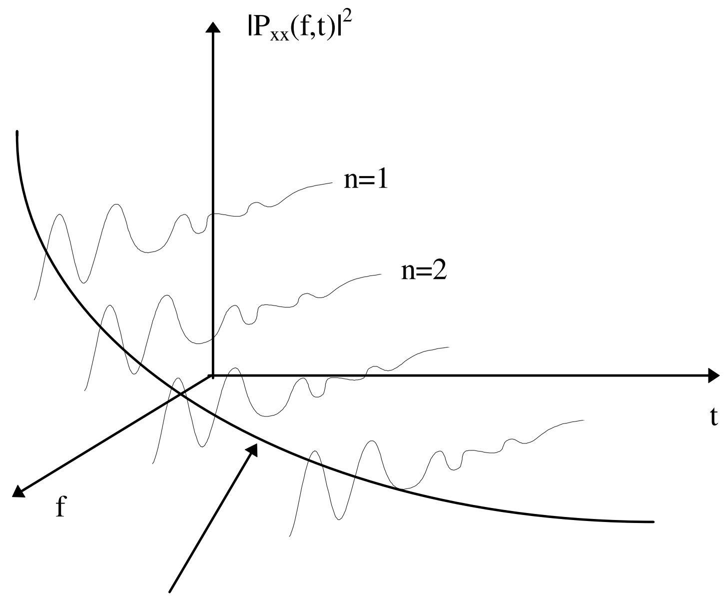
- The input signal is $x[n]$, and the coefficients of the second order linear predictor, $a_2(1), a_2(2)$, are controlled by an adaptive algorithm
- The adaptive algorithm adjusts these coefficients so as to minimise the prediction error power $E\{e^2[n]\}$

Clearly, this structure performs sequential AR spectral estimation

where for each n $\rightsquigarrow$ $$P_x(\omega, n) = \frac{1}{\left|1 + a_1(n)\exp(-\jmath\omega) + a_2(n)\exp(-2\jmath\omega)\right|^2}$$

Adaptive SP vs. Spectral Estimation

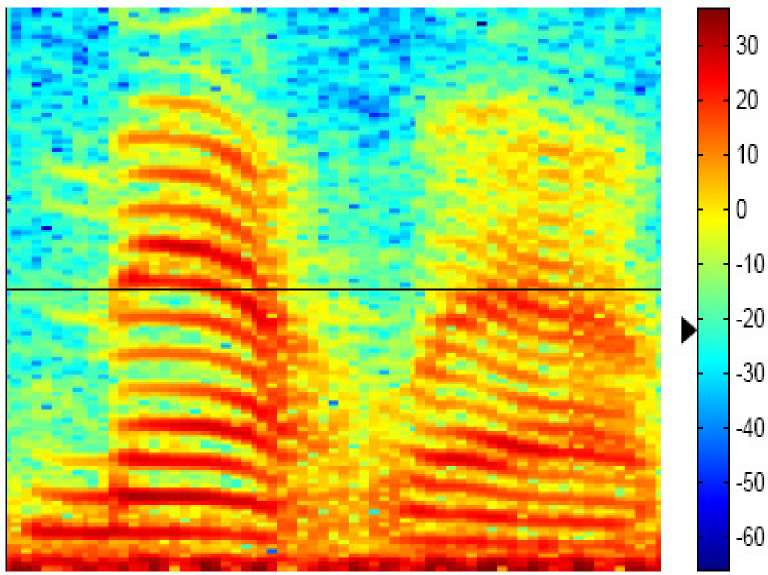
- An adjustment to the coefficient values can be made as each new sample arrives in $\mathbf{x}[n] = [x(n-1), x(n-2), \ldots, x(n-N)]^T$
- Therefore, it is possible to estimate the shape of the input power spectral density, at every iteration, based upon these estimated parameters $\mathbf{a}(n) = [a_1(n), \ldots, a_N(n)]^T$
- This provides a form of **time-frequency analysis** and is the **link between spectral estimation and adaptive signal processing**
- The figure shows the evolution of the PSD estimates



Speech example – saying "Matlab" – 'specgramdemo'

In Matlab: S = SPECTROGRAM(X,WINDOW,NOVERLAP,NFFT,Fs)

Frequency



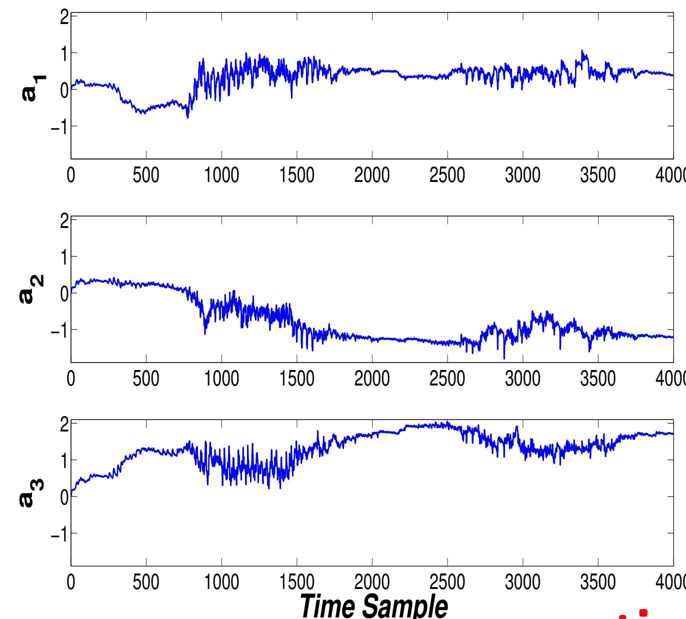
time

M aaa t l aaa b

Time-frequency spectrogram: *Stack PSDs together, $\forall n$*

Darker areas: higher magnitude

Filter coeff. **mtlb_filtercoeffs.m**



time

M aaa t l aaa b

Evolution of AR coefficients: *They follow the signal statistics*

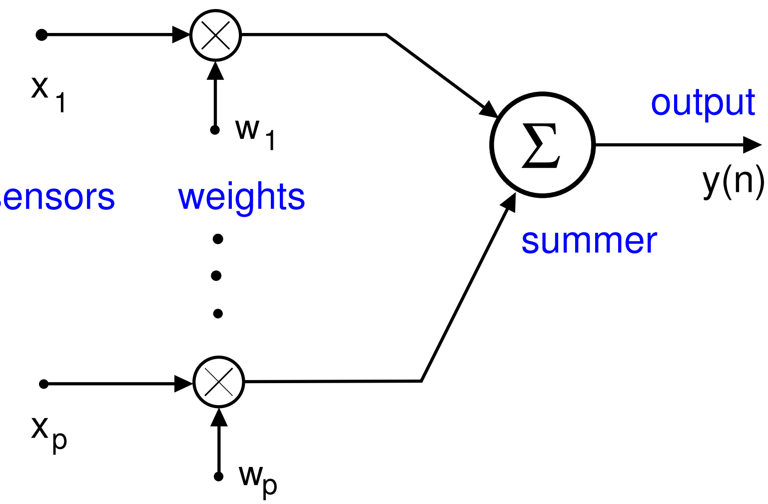
Vowels: more dynamics

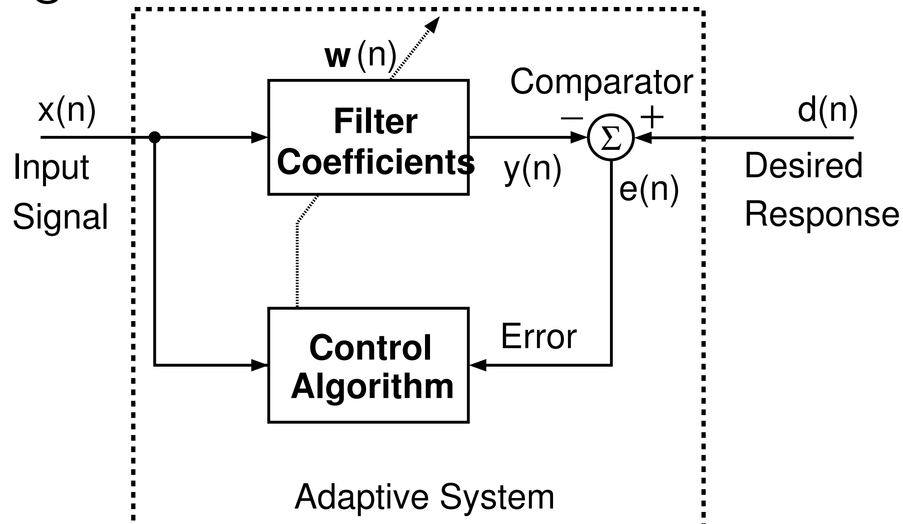
Adaptive learning: Problem formulation

from a fixed h in digital filters to a time–varying $\mathrm{w}(n)$ in adaptive filters

Consider a set of p sensors at different points in space (filter order p)

Let $x_1, x_2, \ldots, x_p$ be the individual signals from the sensors





- The sensor signals are weighted by the corresponding set of **time–varying** filter parameters $\mathbf{w}(n) = [w_1(n), \ldots, w_p(n)]^T$ (weights)
- The weighted signals are then summed to produce the output

$$y(n) = \sum_{i=1}^{p} w_i(n) x_i(n) = \mathbf{x}^T(n)\mathbf{w}(n) = \mathbf{w}^T(n)\mathbf{x}(n), \qquad n = 0, 1, 2, \ldots$$

where $\mathbf{x}^T(n) = [x_1(n), \ldots, x_p(n)], \quad \mathbf{w}^T(n) = [w_1(n), \ldots, w_p(n)]$

Applications of adaptive filters

Adaptive filters have found application in many problems.

We shall concentrate upon their application in four classes of problems:

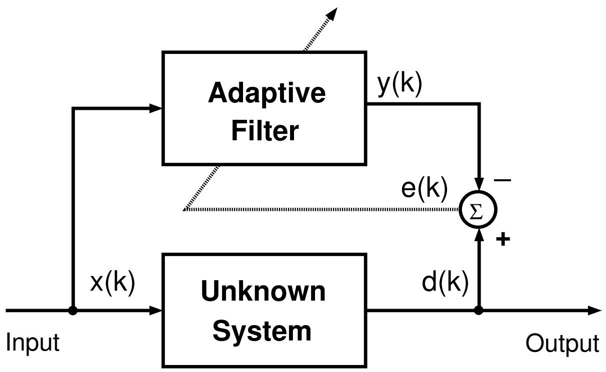
1. **Forward prediction** (the desired signal is the input signal advanced relative to the input of the adaptive filter), as we have seen in sequential AR modelling before

2. **System identification** (both the adaptive filter and the unknown system are fed with the same input signal $x(k)$), as in acoustic echo cancellation

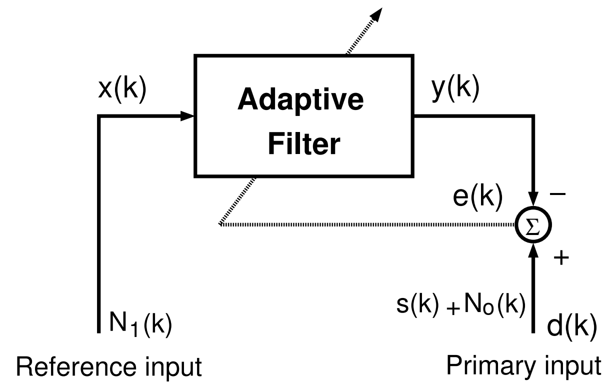
3. **Inverse Modelling** (an adaptive system cascaded with the unknown system), as in channel equalisation

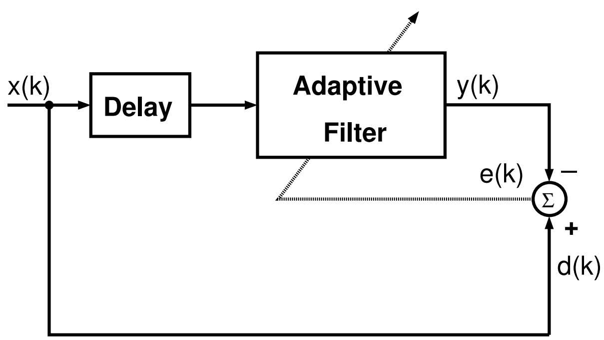
4. **Noise cancellation** (the only requirement is that the noise in the primary input and the reference noise are correlated), as in speech denoising

System Identification

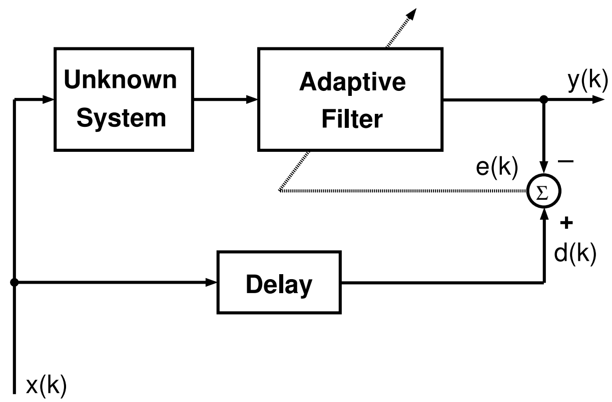


Noise Cancellation





Adaptive Prediction



Inverse System Modelling

Zoom in into adaptive filters

- **Filter architecture** (FIR, IIR, linear, nonlinear)
- **Filter variables:** Input $\{x\}$, output $\{y\}$, and desired $\{d\}$ signal
- **Filter function:** Prediction, system identification, inverse system modelling, noise cancellation
- **Adaptation strategy:** Based on some appropriate function of the output error, $e(n) = d(n) - y(n)$

☞ An adaptive filter operates on the input $x[n]$ to produce an estimate of the desired response $d[n]$

- The generation of the desired response is an important issue and will be described for several concrete applications below
- To measure the performance of an adaptive filter we can consider how some "cost/objective" function of the error $J(e[n])$ evolves with time, or whether the filter coefficient (weight) vector $\mathbf{w}(n)$ converges

Adaptive filtering algorithms

- All are based on the minimisation of some function of the output error:

$$f_1\big(e[n]\big) = |e|, \quad f_2[e[n]] = e^2, \quad f_3\big(e[n]\big) = e^4, \quad f_4\big(e[n]\big) = |e^3|$$

with $e[n] = d[n] - y[n] = d[n] - \mathbf{x}^T[n]\mathbf{w}[n] = d[n] - \mathbf{w}^T[n]\mathbf{x}[n]$

- The "error square" form will be found to be most analytically tractable and appropriate for measurements corrupted by Gaussian noise
- As a rule of thumb, when the measurement noise is sub-Gaussian higher power errors are preferred, and vice versa for super-Gaussian noise
- To derive the optimal setting of an adaptive FIR filter we shall assume that the input and derived response signals are zero-mean and WSS

☞ **It is natural to aim to minimise the expected output error power**, that is the Mean Square Error (MSE):

$$J \equiv \frac{1}{2}\,E\{e^2[n]\} \;\equiv\; \frac{1}{2}\,E\{f\big(\mathbf{w}[n]\big)\}$$

Variables in the algorithm

$$
\begin{aligned}
\mathbf{w}(n) &= \text{the } p\times 1 \text{ column weight (parameter) vector of the filter}\\
&= [w_1(n), w_2(n), \ldots, w_p(n)]^T\\
\mathbf{x}[n] &= [x[n], x[n-1], \ldots, x[n-p+1]]^T \text{ the input vector}
\end{aligned}
$$

Thus, the cost (error,objective) function becomes:

$$
\begin{aligned}
J &= \tfrac{1}{2}\, E\{e(n)\ \ e^T(n)\} = E\{(d[n]-\mathbf{w}^T\mathbf{x}[n])(d[n]-\mathbf{w}^T\mathbf{x}[n])^T\}\\
&= \tfrac{1}{2}\, E\{d^2(n) - d[n]\mathbf{x}^T[n]\mathbf{w} - d[n]\mathbf{w}^T\mathbf{x}[n] + \mathbf{w}^T\mathbf{x}[n]\mathbf{x}^T[n]\mathbf{w}\}\\
&= \tfrac{1}{2}\, E\{d^2(n) - 2d[n]\mathbf{x}^T[n]\mathbf{w} + \mathbf{w}^T\mathbf{x}[n]\mathbf{x}^T[n]\mathbf{w}\}\\
&= \tfrac{1}{2}\, E\{d^2(n)\} \text{ - } \tfrac{1}{2}\, 2\mathbf{w}^T E\{\mathbf{x}[n]d[n]\} + \tfrac{1}{2}\ \mathbf{w}^T E\{\mathbf{x}[n]\mathbf{x}^T[n]\}\mathbf{w}
\end{aligned}
$$

Therefore, $J(\mathbf{w}) = E\{|e(n)|^2\} = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w}$

☞ Definitions of the cross–correlation vector, $\mathbf{p}$ and autocorrelation matrix, $\mathbf{R}$:

$$
\mathbf{p} \equiv E[\mathbf{x}[n]d[n]] \qquad \mathbf{R} \equiv E[\mathbf{x}[n]\mathbf{x}^T[n]]
$$

Wiener–Hopf solution

The optimal **minimum mean square error** (MMSE) solution corresponds to the zero–gradient point of J and is found from

$$\nabla J(\mathbf{w}) = \frac{\partial J}{\partial \mathbf{w}} = -\mathbf{p} + \mathbf{R} \cdot \mathbf{w} \qquad \Rrightarrow \qquad \text{for} \quad J_{min} \qquad -\mathbf{p} + \mathbf{R} \cdot \mathbf{w}_{opt} = \mathbf{0}$$

$$\Rightarrow \quad \mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{p} \qquad \textbf{the Wiener–Hopf equation}$$

- The inverse autocorrelation matrix, $\mathbf{R}^{-1}$, effectively acts as a conditioning matrix (pre-whitening structure)
- The key ingredient is the cross–correlation vector $\mathbf{p}$
- The Wiener filter is designed based upon the degree of correlation between the desired response and the input to the filter, namely second order cross–correlation information
- The minimum MSE is given by the value of $J(\mathbf{w})$ when $\mathbf{w} = \mathbf{w}_{opt}$, i.e.

$$J_{min} = J(\mathbf{w}_{opt}) = \sigma_d^2 - 2\mathbf{w}_{opt}^T\mathbf{p} + \mathbf{w}_{opt}^T\mathbf{R}\mathbf{w}_{opt} = \sigma_d^2 - \mathbf{w}_{opt}^T\mathbf{p}$$

Quantitative performance assessment $\hookrightarrow$ error surface

Recall that $J(\mathbf{w}) = E\{|e(n)|^2\} = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w}$

Therefore (we also had $e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n), \quad \mathbf{p} = E\{d(n)\mathbf{x}(n)\}$):

$$\mathbf{w}_{opt} = \arg\min_{\mathbf{w}} J(\mathbf{w}) = \mathbf{R}^{-1}\mathbf{p} \quad \hookrightarrow \quad J_{min} = J(\mathbf{w}_{opt}) = \sigma_d^2 - \mathbf{w}_{opt}^T\mathbf{p}$$

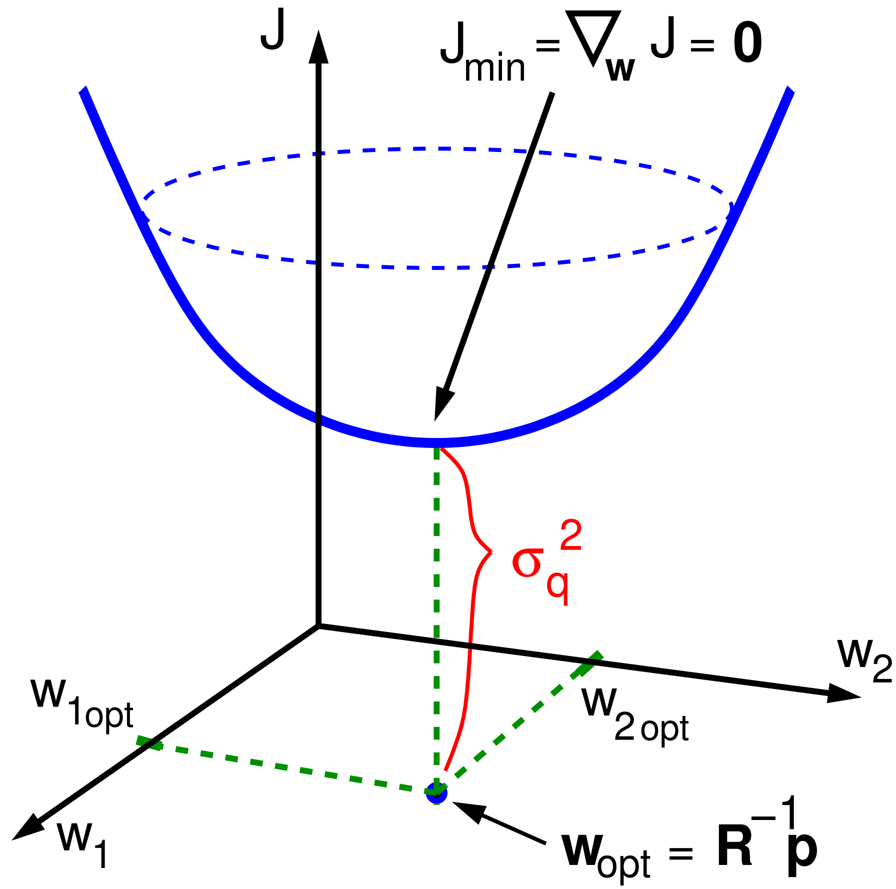
So, what is the value of J_{min}?
Assume without loss in generality that the teaching signal $d(n)$ is the output of a system with coefficients $\mathbf{w}_{opt}$

$$d(n) = \mathbf{x}^T(n)\mathbf{w}_{opt} + q(n), \quad q \sim \mathcal{N}(0, \sigma_q^2)$$

Then

$$\begin{aligned} \sigma_d^2 &= E\Big\{\big[\mathbf{w}_{opt}^T\mathbf{x}(n) + q(n)\big]d(n)\Big\} \\ &= \mathbf{w}_{opt}^T\mathbf{p} + \sigma_q^2 \end{aligned}$$

and $\quad J_{min} = \sigma_d^2 - \mathbf{w}_{opt}^T\mathbf{p} = \sigma_q^2$



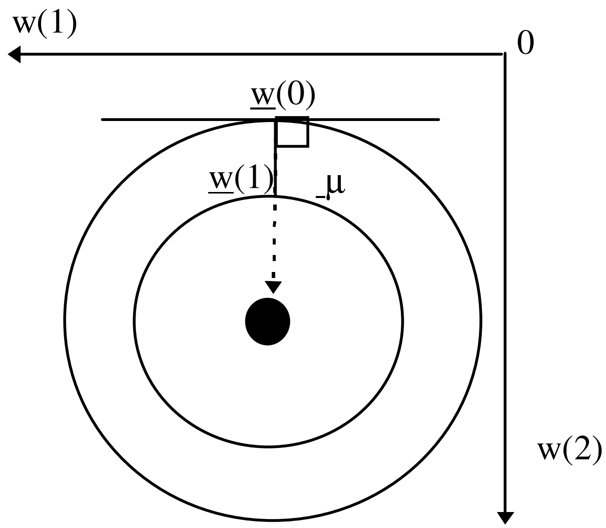
Steepest Descent (SD) methods

an iterative solution that does not require an inverse of the correlation matrix

The update equation for the steepest descent method to find the minimum of the cost function J is given by

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu(-\nabla J|_{\mathbf{w}[n]}) = \mathbf{w}[n] + \mu\big[\underbrace{\mathbf{p} - \mathbf{R}\mathbf{w}[n]}_{-\nabla J_{\mathbf{w}[n]}}\big]$$

∘ The parameter μ is termed the adaptation gain (learning rate, step size) and controls the speed of convergence



The convergence of the SD algorithm from the initial point $\mathbf{w}[0]$ toward the optimum.[1]

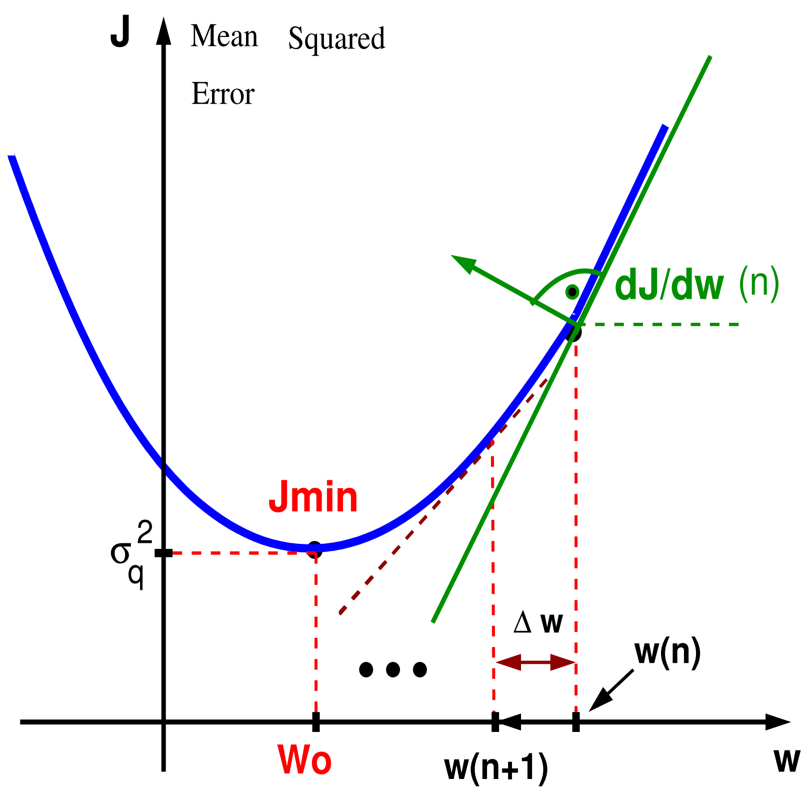
[1] This diagram is for WGN input ⇒ the direction of steepest descent always point to the minimum.

Method of steepest descent: Iterative Wiener solution

we reach $\mathbf{w}_o$ through iterations $\mathbf{w}(n+1) = \mathbf{w}(n) + \Delta\mathbf{w}(n) = \mathbf{w}(n) - \mu\nabla_{\mathbf{w}} J(n)$

Problem with the Wiener filter: It is computationally demanding to calculate the inverse of a possibly large correlation matrix $\mathbf{R}_{xx}$.

Solution: Allow the weights to have a **time–varying** form, so that they can be adjusted in an **iterative** fashion along the error surface.



This is achieved in the direction of steepest descent of error surface, that is, in a **direction opposite to the gradient vector** whose elements are defined by $\nabla_{w_k} J, \quad k = 1, 2, \ldots, p$.

For a teaching signal, assume

$$d(n) = \mathbf{x}^T(n)\mathbf{w}_o + q(n),$$

where $q \in \mathcal{N}(0, \sigma_q^2)$, so that we have $J_{min} = \sigma_q^2$

Role of eigen-analysis in Wiener solution

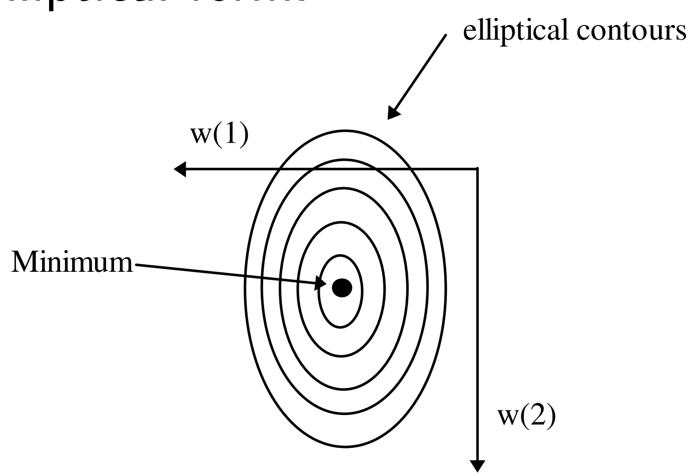
The shape of the error performance surface is related directly to the eigen–structure of the autocorrelation matrix $\mathbf{R}$.

The condition number of $\mathbf{R}$, that is, $\lambda_{max}/\lambda_{min}$, is particularly important.

For a **white input**, $\mathbf{R} = \begin{bmatrix} r_{xx}(0) & 0 \\ 0 & r_{xx}(0) \end{bmatrix} \rightarrow$ **condition number = 1**.

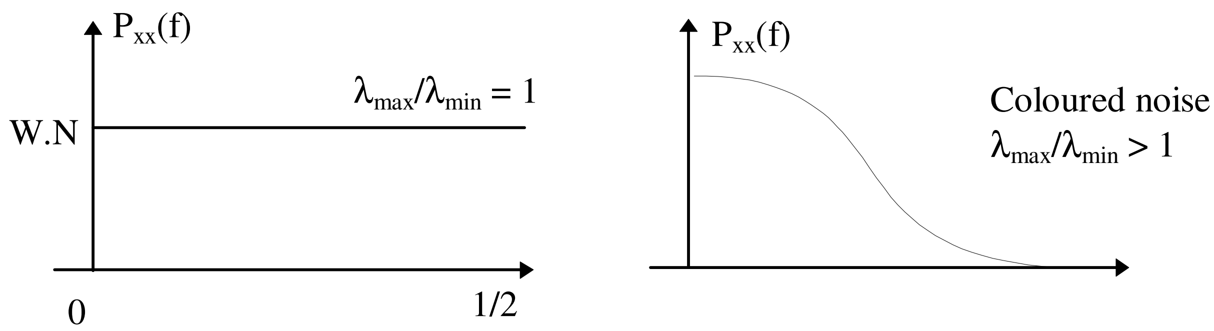
Therefore the **contours of J are circles when projected onto the $(w(1), w(2))$ plane.**

When the input is coloured, the condition number increases, and the contours will take an elliptical form.



Eigenvalues vs PSD (a useful rule of thumb)

When the condition number $\lambda_{max}/\lambda_{min} > 1$ the power spectral density of the input to the Wiener filter will depart from the flat case of white noise.



A very important bound for the condition number is given by

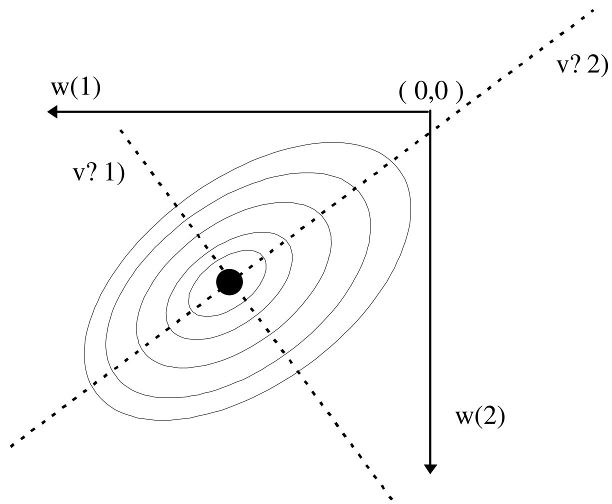
$$1 \leq \frac{\lambda_{max}}{\lambda_{min}} \leq \frac{P_{xx}^{max}(f)}{P_{xx}^{min}(f)}$$

which shows that as the spread of the input PSD increases so too will the elliptical form of the contours of J.

This will affect the convergence of gradient-based adaptive algorithms.

Coloured Input – Convergence

For the coloured case, as depicted in the figure below, the direction of steepest descent does not necessarily point at the minimum, it depends on the starting point we are taking.



To analyse the convergence of the method of steepest descent, replace the $[w(1), w(2)]$ axis by moving the origin to $\mathbf{w}_{opt}$ and replacing $\mathbf{w}$ by $\mathbf{v} = (\mathbf{w} - \mathbf{w}_{opt})$ and then rotating the axes by a new matrix $\mathbf{S}$, to align with the principal axes denoted $\mathbf{v}'$ in the diagram above.

Eigenvalues and convergence

The matrix $\mathbf{S}$ corresponds to a component of the spectral factorisation of the autocorrelation matrix, i.e.

$$\mathbf{R}_{xx} = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^T \quad where \quad \mathbf{\Lambda} = Diag(\lambda_1, \lambda_2, \ldots, \lambda_p)$$

and $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_p]$, $\mathbf{s}_i$ is a normalised eigenvector.

Therefore $\mathbf{S} \cdot \mathbf{S}^T = \mathbf{I}$ the $p \times p$ identity matrix.

The purpose of redefining the axes is to "decouple" the learning modes of the adaptive filter.

Proceeding with the analysis of $\mathbf{w}[n+1]$, we have

$$\begin{aligned}
\mathbf{w}[n+1] &= \mathbf{w}[n] + \mu(\mathbf{p} - \mathbf{R}\mathbf{w}[n]) \\
&= \mathbf{w}[n] + \mu(\mathbf{R}\mathbf{w}_{opt} - \mathbf{R}\mathbf{w}[n]) \\
&= \mathbf{w}[n] + \mu\mathbf{R}\left(\mathbf{w}_{opt} - \mathbf{w}[n]\right)
\end{aligned}$$

Convergence analysis – weight error vector $\mathbf{v}(n)$

Subtract from both sides $\mathbf{w}_{opt}$

$$\mathbf{v}[n+1] = \mathbf{w}[n+1] - \mathbf{w}_{opt} = \underbrace{\mathbf{w}[n] - \mathbf{w}_{opt}}_{\mathbf{v}[n]} - \mu\mathbf{R}(\underbrace{\mathbf{w}_{opt} - \mathbf{w}[n]}_{\mathbf{v}[n]})$$

Using the spectral factorisaton of $\mathbf{R}_{xx}$

$$\begin{aligned} \mathbf{v}[n+1] &= [\mathbf{I} - \mu\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T]\mathbf{v}[n] \\ \mathbf{S}^T\mathbf{v}[n+1] &= \mathbf{S}^T(\mathbf{I} - \mu\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T)\mathbf{v}[n] \\ \text{we define } \mathbf{v}'[n] = \mathbf{S}^T\mathbf{v}[n], & \quad then \end{aligned}$$

$$\begin{aligned} \mathbf{v}'(n+1) &= [\mathbf{S}^T - \mu\mathbf{S}^T\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T]\mathbf{v}(n) \\ &= [\mathbf{S}^T - \mu\mathbf{\Lambda}\mathbf{S}^T]\mathbf{v}(n) \\ &= [\mathbf{I} - \mu\mathbf{\Lambda}]\mathbf{v}'(n) \end{aligned}$$

Modes of convergence

Finally,

$$\mathbf{v}'(n+1) = [\mathbf{I} - \mu\mathbf{\Lambda}]\,\mathbf{v}'(n) \quad \text{where } \mathbf{I} - \mu\mathbf{\Lambda} \text{ is diagonal matrix}$$

and we have the so-called **modes of convergence**

$$v^j[n+1] = (1 - \mu\lambda_j)\,v^j(n) \qquad \text{where } j = 1, 2, \ldots, p$$

For each mode, at adaptation sample number n, we have:

$$v^j[n+1] = (1 - \mu\lambda_j)^n\,v^j(0)$$

For convergence, we require that

$$|1 - \mu\lambda_j| < 1$$

then the algorithm is guaranteed to converge to the Wiener-Hopf

Solution:

$$|1 - \mu\lambda_j| < 1 \qquad \Rightarrow \qquad -1 < 1 - \mu\lambda_j < 1$$

Convergence requirement

Now: $\quad 0 < \mu < 2/\lambda_i \qquad \forall \lambda_i$

Generally, the eigenvalues are not equal ($\lambda_j = \sigma_N^2$ if white noise $\forall i$), therefore we take the worst case

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

This condition is also sufficient for convergence of the steepest descent algorithm in **the mean square**.

This is easily seen from the following expression for the mean square error as a function of discrete time n

$$J[n] = J_{min} + \sum_{k=0}^{p} \lambda_k \left(1 - \mu \lambda_k\right)^{2n} \left|v'_k[0]\right|^2$$

The Least Mean Square (LMS) algorithm

From the steepest descent algorithm (N.B. $J = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w}$)

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu(-\nabla J|_{\mathbf{w}[n]}) = \mathbf{w}[n] + \mu\big(\mathbf{p} - \mathbf{R}\mathbf{w}[n]\big)$$

In practice, we must estimate the statistics to form the search direction, i.e.

$$\mathbf{p} = E\{\mathbf{x}[n]d[n]\} \qquad\qquad \mathbf{R} = E\{\mathbf{x}[n]\mathbf{x}^T[n]\}$$

In adaptive filtering we use an **instantaneous estimate**

$$\hat{\mathbf{p}} = \mathbf{x}[n]d[n] \qquad \text{and similarly} \qquad \hat{\mathbf{R}} = \mathbf{x}[n]\mathbf{x}^T[n] \qquad \text{to yield}$$

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu\big(\mathbf{p} - \mathbf{R}\mathbf{w}[n]\big) \approx \mathbf{w}[n] + \mu\big(\mathbf{x}[n]d[n] - \mathbf{x}[n]\mathbf{x}^T[n]\mathbf{w}(n)\big)$$

Least Mean Square algorithm [Widrow,1960]:

$$\begin{aligned} \mathbf{w}[n+1] &= \mathbf{w}[n] + \mu\mathbf{x}[n]\left(d[n] - \mathbf{x}^T[n]\mathbf{w}[n]\right) = \mathbf{w}[n] + \mu e[n]\mathbf{x}[n] \\ \mathbf{w}[0] &= \mathbf{0} \qquad \text{where} \qquad d[n] - \mathbf{x}^T[n]\mathbf{w}[n] = e[n] \end{aligned}$$

Computational requirement for the LMS algorithm

- To calculate $e[n]$

  p multiplications + p additions

- For weight update
  - 1 multiplication (for $2\mu e[n]$) + p multiplications (for $\mu \mathbf{x}[n]e[n]$) $\Rightarrow$ (p + 1) multiplications
  - p additions (updating $\mathbf{w}[n]$)

$\Rightarrow$ the LMS algorithm is an $\mathcal{O}(2N)$ algorithm

- only twice the complexity of a fixed filter

- together with its robust performance, is the reason why it finds extensive use in channel equalisation and echo cancellation in modems, and coding in speech (ADPCM) codecs.

Geometric insight into the LMS

direction of the weight update vector is parallel to the input vector

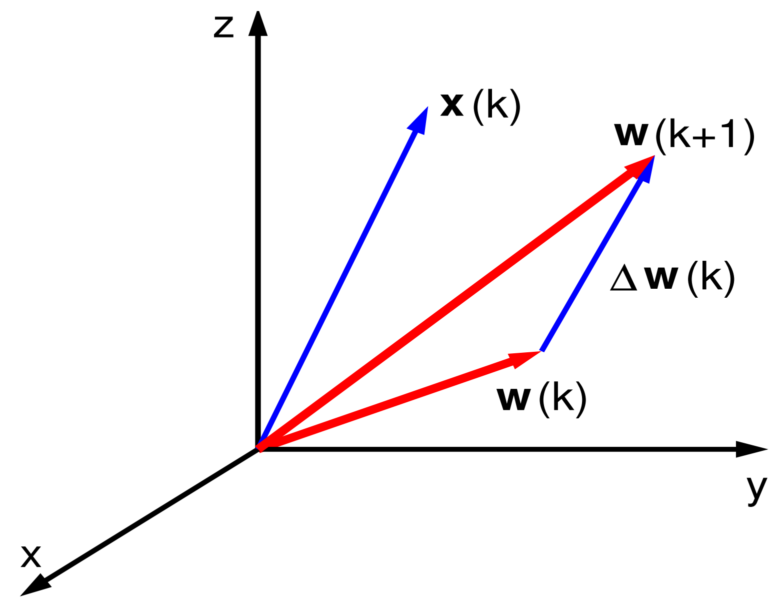
Recap: Let us derive LMS directly from the instantaneous cost function

$$J(k) = \frac{1}{2}e^2(k)$$

Then

$$\begin{aligned} e(k) &= d(k) - y(k) \\ y(k) &= \mathbf{x}^T(k)\mathbf{w}(k) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) - \mu\nabla_{\mathbf{w}}J(k) \\ \nabla_{\mathbf{w}}J(k) &= \frac{1}{2}\underbrace{\frac{\partial e^2(k)}{\partial e(k)}}_{e(k)}\underbrace{\frac{\partial e(k)}{\partial y(k)}}_{-1}\underbrace{\frac{\partial y(k)}{\partial \mathbf{w}(k)}}_{\mathbf{x}(k)} \end{aligned}$$

LMS: $\mathbf{w}(k+1) = \mathbf{w}(k) + \underbrace{\mu e(k)\mathbf{x}(k)}_{\Delta\mathbf{w}(k)}$



Geometry of learning. Weight update $\Delta\mathbf{w}(k)$ is parallel to the tap-input in filter memory $\mathbf{x}(k)$
$\looparrowright$ $\Delta\mathbf{w}(k)$ follows statistics of $\mathbf{x}$.

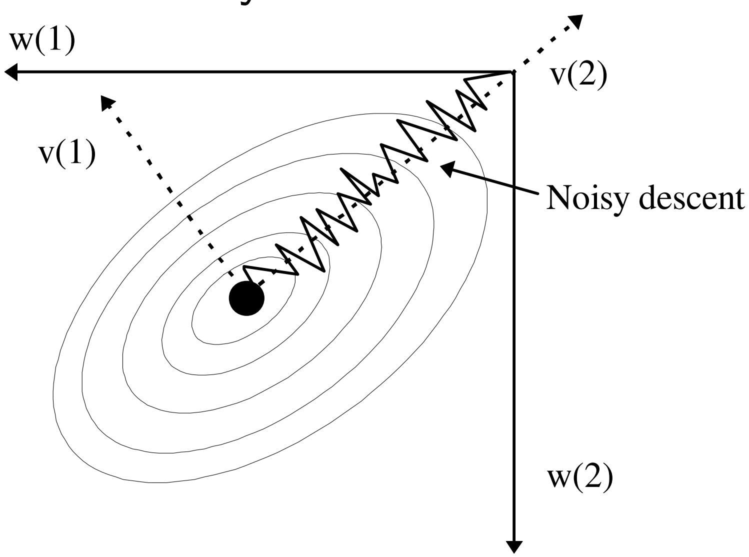
The weight update is dominated by the largest element $x_{max}(k)$ of $\mathbf{x}(k)$, which can be true behaviour or an artefact.

Error performance surface for LMS

The actual LMS algorithm follows a **noisy descent direction** due to the **approximate gradient expression** used in the update equation.

Only on the average will the LMS algorithm follow the direction of SD

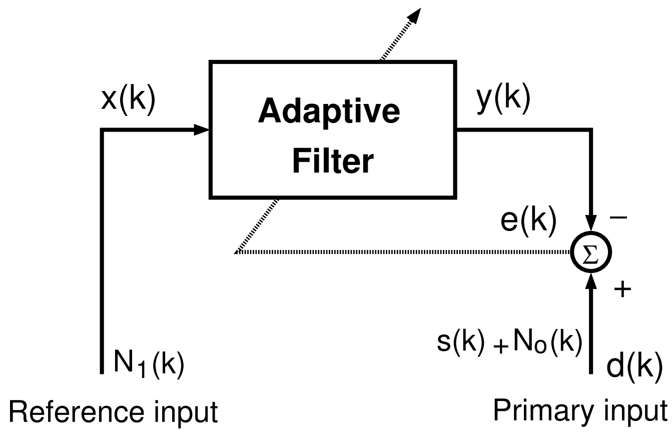
We wish to determine the value so that the average value of $\mathbf{w}[n]$ tends to the Wiener solution - this does not mean that the actual value of $\mathbf{w}[n]$ will equal the Wiener solution at anytime.

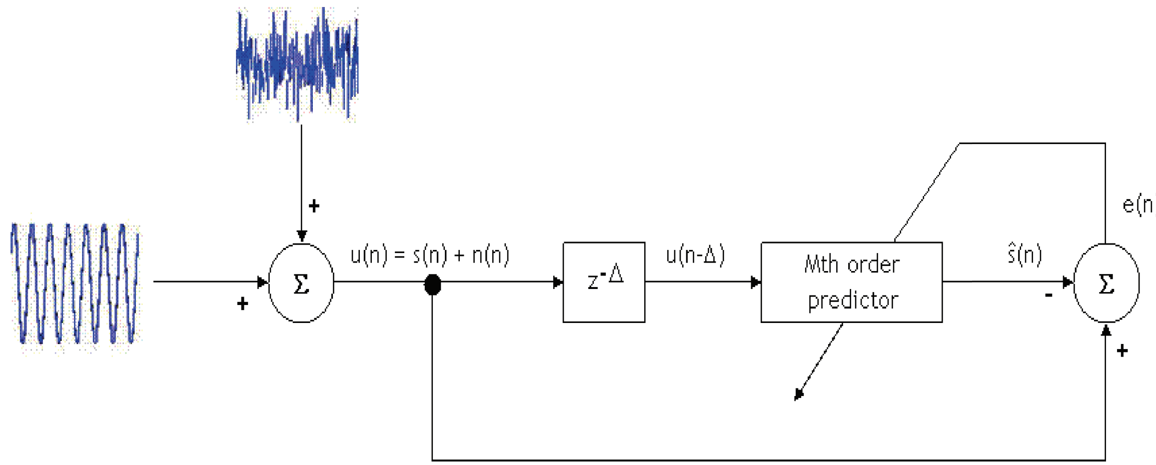


Adaptive line enhancement (no reference) **'lms_fixed_demo'**

Enhancement of a 100Hz signal in band-limited WGN, with a $N = 30$ LMS filter

From the configuration with reference (left) to self-tuning configuration (right)





- Adaptive line enhancement (ALE) refers to the case where we want to clean a noisy signal,e.g. a noisy sinewave $u(n) = `sin(n)' + `wn(n)'$
- ALE is effectively an adaptive predictor equipped with a de-correlation stage, symbolised by $z^{-\Delta}$. The autocorrelation of noise is narrow, so

$$E\{u(n)u(n-\Delta)\} \approx E\{s(n)s(n-\Delta)\}$$

- By shifting $u(n)$ by Δ samples apart we aim to remove any correlation between the noise contribution in the samples $u(n)$ and $u(n-\Delta)$
- A small delay (phase shift) of Δ samples is introduced at the output

The operation of adaptive noise cancellers

The physics behind the operation of ANC is as follows

☞ The objective is to produce the "system output", $e = s + N_0 - y$, that is the best fit (in the least squares sense) to the signal of interest, s.

Upon squaring, we have

$$e^2 = s^2 + (N_0 - y)^2 + 2s(N_0 - y)$$

Since s is uncorrelated with N_0 and y, we have

$$E[e^2] = E[s^2] + E\big[(N_0 - y)^2\big] + 2\,\underbrace{E\big[s(N_0 - y)\big]}_{=0,\ uncorrelated} = E[s^2] + E\big[(N_0 - y)^2\big]$$

☞ The signal power, $E[s^2]$, will be unaffected as the filter minimises $E[e^2]$.

$\leadsto$ minimum output power $E_{min}[e^2] = E[s^2] + E_{min}\big[(N_0 - y)^2\big]$

In other words, when $E[e^2]$ is minimised, so too is $E[(N_0 - y)^2]$

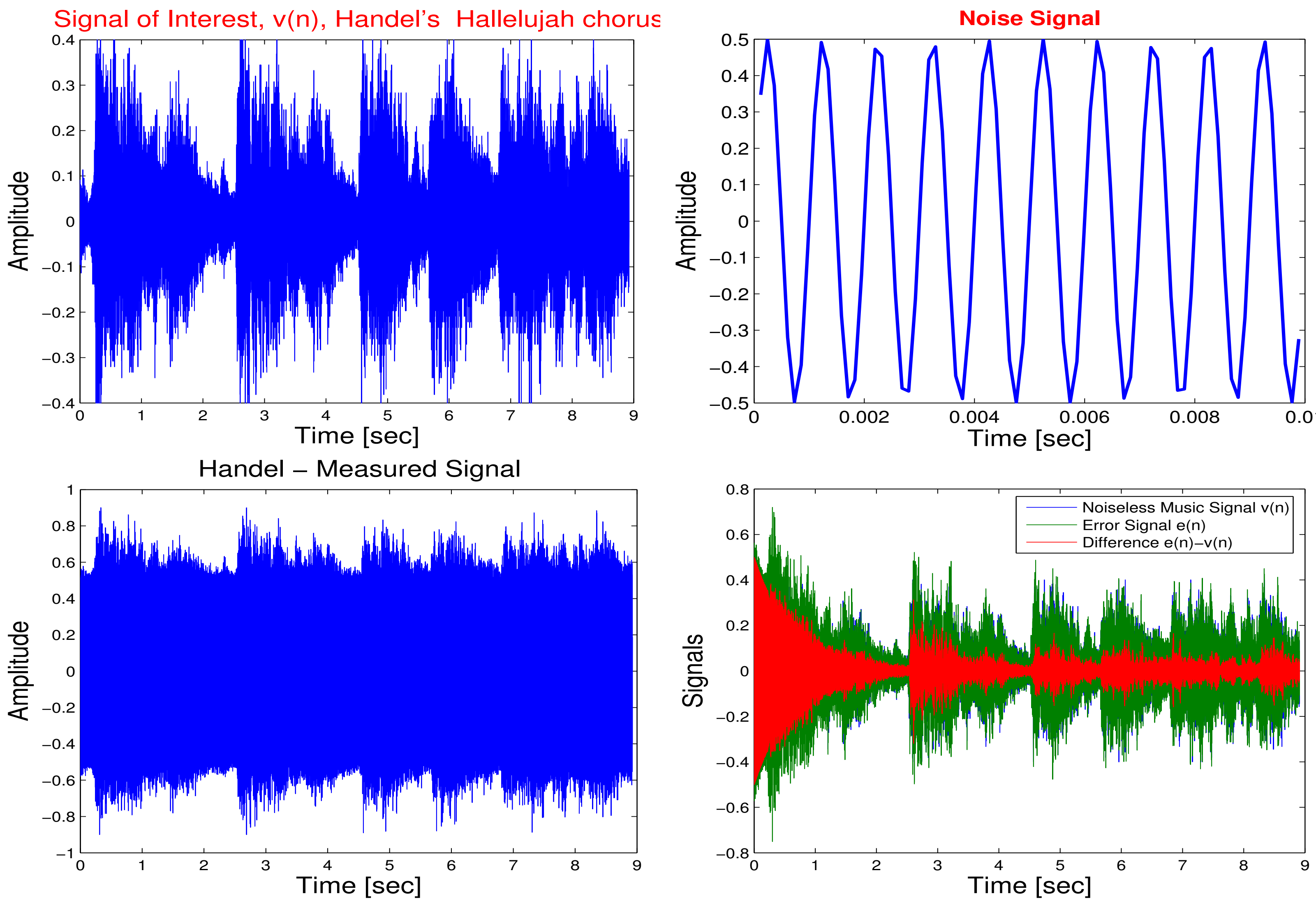
☞ The "filter output", y, is the best LS estimate of the primary noise, N_0.

Moreover, when $E[(N_0 - y)^2]$ is minimised, so too is $E[(e - s)^2]$

☞ When the filter output is adjusted to minimise to total output power, then the "system output", e is a best LS estimate of the signal of interest, s.

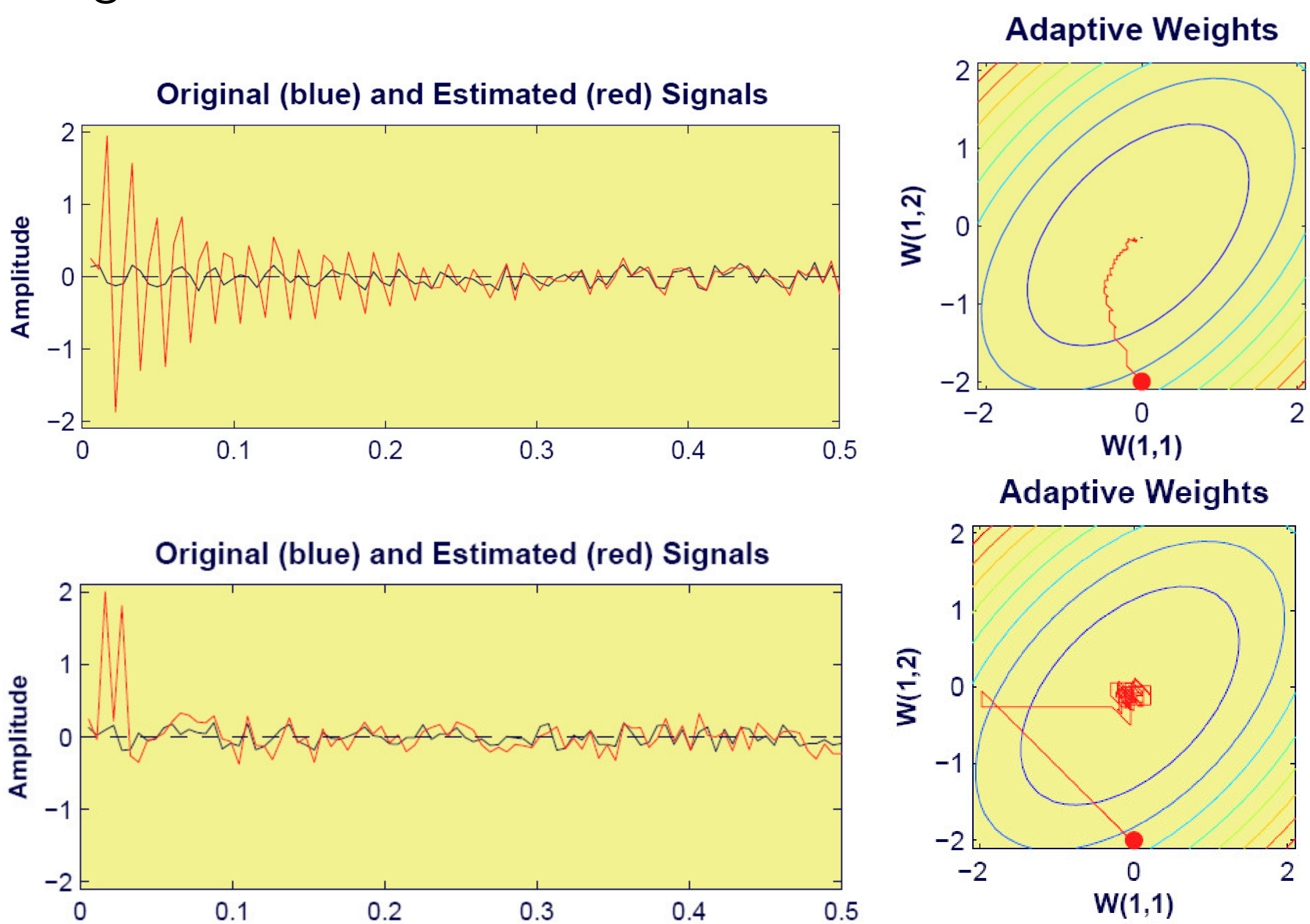
ALE - interference removal in music perform. ‘ALE_Handel’

Handel’s Hallelujah chorus with $1000Hz$ interference, N=32, $\Delta = 100$



Error surface for an echo cancellation example – ('nnd10nc in Matlab')

Signal and Prediction | Error surface



Top panel - learning rate 0.1 | Bottom panel - learning rate 0.9

Convergence of LMS

how fast and how well do we approach the steady state

- **Convergence in the mean**, to establish whether
  $\mathbf{w}(n) \overset{n\to\infty}{\longrightarrow} \mathbf{w}_{opt}$, that is $E\{\mathbf{w}[n+1]\} \approx E\{\mathbf{w}(n)\} \quad as \quad n \rightarrow +\infty$

- **Convergence in the mean square** to establish whether the variance of the weight error vector $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_{opt}$ approaches J_{min} as $n \rightarrow \infty$

The analysis of convergence in the mean is straightforward, whereas the analysis of convergence in the mean square is more mathematically involved.

It is convenient to analyse convergence for a **white** input $\mathbf{x}(n)$ and using the **independence assumptions** (that is, all the data in the filter memory are jointly Gaussian)

i) the sequence of $\{\mathbf{x}\}$ are statistically independent;

ii) $\{\mathbf{x}\}$ indepdendent of $\{d\}$;

iii) $\{d\}$ is independent identically distributed (iid);

iv) $\mathbf{w}(n) \perp e(n) \perp d(n) \perp \mathbf{x}(n) \perp \mu$

Convergence of LMS – continued

Based on the cost function $J(n) = \frac{1}{2}E\{e^2(n)\} = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T R\mathbf{w}$.
Without loss in generality assume that

$$\begin{aligned} d(n) &= \mathbf{x}^T(n)\mathbf{w}_{opt} + q(n), \qquad\qquad q(n) \sim \mathcal{N}(0, \sigma_q^2) \quad \text{so that} \\ e(k) &= \mathbf{x}^T(k)\mathbf{w}_{opt} + q(k) - \mathbf{x}^T(k)\mathbf{w}(k) \qquad \text{and the LMS update} \\ \mathbf{w}(n+1) &= \mathbf{w}(n) + \mu\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}_{opt} - \mu\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{w}(n) + \mu q(n)\mathbf{x}(n) \end{aligned}$$

so that the minimum achievable mean square error becomes $J_{min} = \sigma_q^2$.
Subtract the optimal weight vector $\mathbf{w}_{opt}$ from both sides, and knowing that $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_{opt}$, gives

$$\begin{aligned} \mathbf{v}(n+1) &= \mathbf{v}(n) - \mu\mathbf{x}(n)\mathbf{x}^T(n)\mathbf{v}(n) + \mu q(n)\mathbf{x}(n) \qquad \text{and} \\ E\{\mathbf{v}(n+1)\} &= \Big(\mathbf{I} - \mu \underbrace{E\{\mathbf{x}(n)\mathbf{x}^T(n)\}}_{=\ \text{corr. matrix}\ \mathbf{R}}\Big)E\{\mathbf{v}(n)\} + \mu \underbrace{E\{q(n)\mathbf{x}(n)\}}_{=\mathbf{0}\ \text{as}\ q\perp\mathbf{x}} \end{aligned}$$

Convergence of LMS in the Mean

Observations:

- To analyse the "modes of convergence", they should be decoupled;
- In other words $\mathbf{R}$ should be diagonal (or the input should be white);
- As $\mathbf{R}$ is Toeplitz, there is a unitary matrix $\mathbf{Q}$ so that $\mathbf{R} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$;
- $\mathbf{Q}$ is a matrix of eigenvectors and $\boldsymbol{\Lambda} = diag(\lambda_{max}, \ldots, \lambda_{min})$;
- $\mathbf{Q}$ can therefore rotate $\mathbf{R}$ into the diagonal matrix $\boldsymbol{\Lambda}$, that is, $\mathbf{QRQ}^T \rightsquigarrow \boldsymbol{\Lambda}$;

We can therefore multiply the equation for convergence modes by $\mathbf{Q}$ to have "rotated coordinates" $\mathbf{v}'(n) = \mathbf{Qv}(n)$, and a diagonal $\mathbb{R}$ so that

$$\begin{aligned} \mathbf{v}'(n+1) &= \big(\mathbf{I} - \mu\boldsymbol{\Lambda}\big)\mathbf{v}'(n) \end{aligned}$$

$$\begin{bmatrix} v_1'(n+1) \\ \vdots \\ v_p'(n+1) \end{bmatrix} = \begin{bmatrix} 1-\mu\lambda_{max} & 0 & \cdots & 0 \\ 0 & 1-\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-\lambda_{min} \end{bmatrix} \begin{bmatrix} v_1'(n) \\ \vdots \\ v_p'(n) \end{bmatrix}$$

Convergence of LMS in the mean – continued

Determine the value of μ to guarantee convergence in the mean, i.e.

$$\text{Wiener solution}: \qquad \mathbf{w}_{opt} = E\{\mathbf{w}(n)\} = \mathbf{R}_{xx}^{-1}\mathbf{p} \qquad \text{as} \quad n \rightarrow +\infty$$

The mode of convergence corresponding to the largest eigenvalue is

$$v'(n+1) = (1 - \mu\lambda_{max})v'(n)$$

which converges to zero for $|1 - \lambda_{max}| < 1$. Thus, for convergence

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

In practice, calculation of eigenvalues is too computationally complex, so we use the relationship that

$$trace(\mathbf{R}_{xx}) = \mathrm{p}\ r_{xx}(0) = \sum_{i=1}^{p} \lambda_i$$

and because $\lambda_i \geq 0 \quad \forall i$ then $\sum \lambda_i > \lambda_{max}$, thus a practical bound is

$$0 < \mu < \frac{2}{\mathrm{p}\ r_{xx}(0)} = \frac{2}{p\,\sigma_x^2} \qquad \text{Depends on signal power} \quad \sigma_x^2$$

Convergence in the Mean Square: Some practical results

For the approximation in the independence assumption we use

$$0 < \mu < 1/[3\,\mathrm{p}\; r_{xx}(0)]$$

To find the condition on for convergence in the mean square is involved [Haykin 1996].

However, the key result is that the mean square of the LMS algorithm converges to a **steady state value**

$$J[\infty] = J_{min} + J_{ex}[\infty]$$

if and only if

$$0 < \mu < \frac{1}{\lambda_{max}} \quad \text{and} \quad \mu \sum_{k=1}^{p} \frac{\lambda k}{1 - \mu \lambda_k} < 1$$

where $J_{ex}[\infty]$ is the **excess mean squared error** (due to gradient noise).

LMS – Misadjustment & time constants

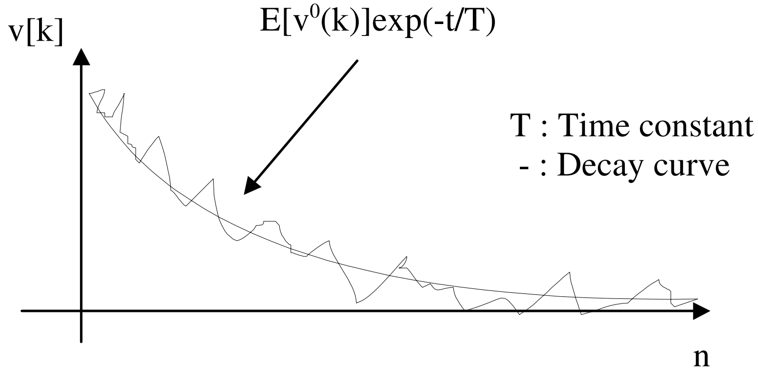
A dimensionless quantity used to quantify the accuracy of the convergence of the LMS algorithm is the misadjustment

$$\mathcal{M} = J_{ex}[\infty]/J_{min}$$

The misadjustment can be approximated as

$$\mathcal{M} \approx \frac{1}{2}\,\mu\; trace\{\mathbf{R}\} \quad \text{and for white input} \quad \approx \frac{1}{2}\mu\, p\, \sigma_x^2$$

To quantify the speed of convergence of the LMS algorithm, **time constants** are used.



Convergence in the mean of the k-th mode of the LMS algorithm

Time constants – analytical form

This can be represented as

$$E[v^{n+1}[k]] = (1 - 2\mu\lambda_k)^n E\{v^0[k]\}$$

where the superscript $n + 1$ denotes discrete time, and $(1 - 2\mu\lambda_k)$ is the factor affecting the rate of decay of $v[2]$.

The parameter τ indicates when the value of $E\{v(k)\}$ has fallen by the factor of e^{-1} of its initial value

$$\tau_k \equiv -\frac{1}{ln(|1 - 2\mu\lambda_k|)}$$

For a length p adaptive filter there will be N time constants and the slowest mode will correspond to the smallest eigenvalue.

- $\Rightarrow$ there is a trade-off in terms of selecting the adaptation gain
- it must satisfy the conditions for convergence in the mean and mean square, and be small enough to provide acceptable steady state error, whilst being large enough to ensure the convergence modes are not too long
- In a practical situation where the input statistics are changing, there will also be another constraint upon the adaptation gain to ensure good tracking performance

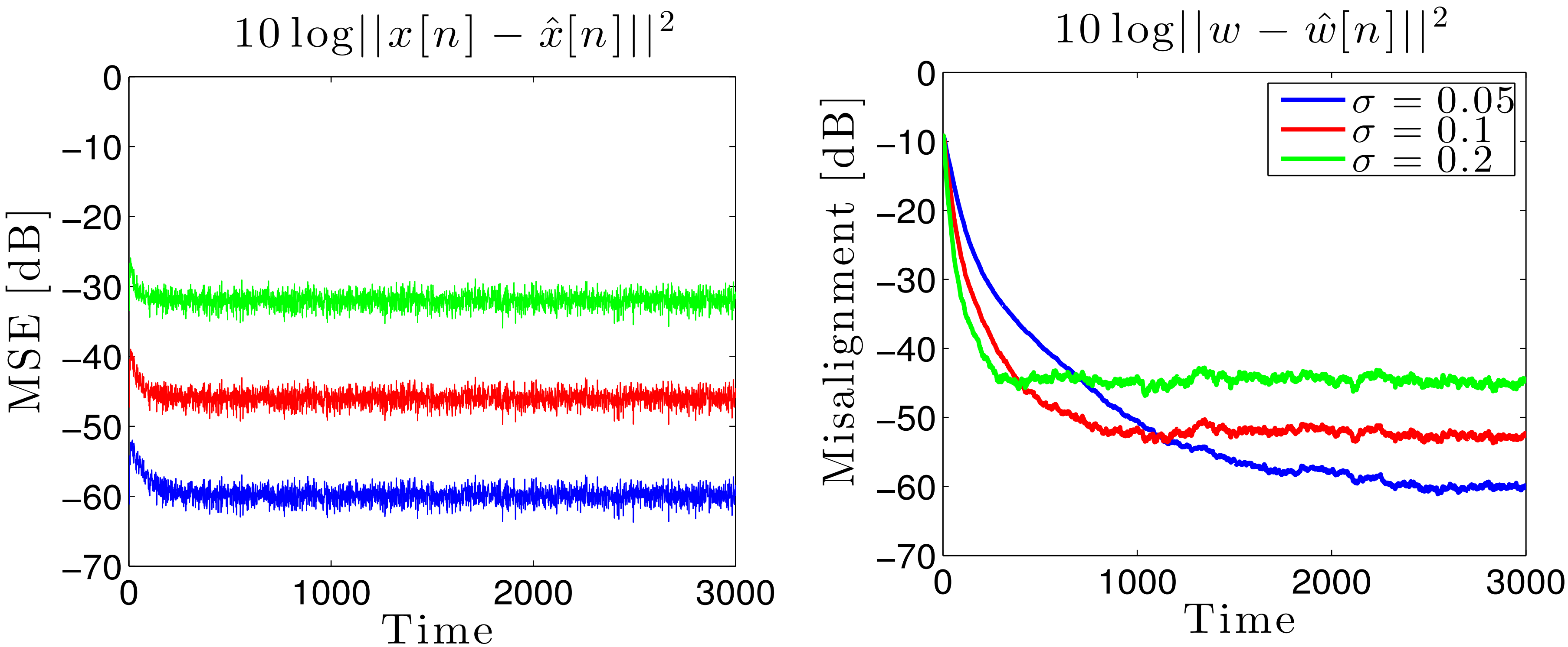
Learning curves: behaviour of MSE $\hookrightarrow$ plot of $10log|e(n)|^2$

evolution of mean square error along the adaptation

For illustration, consider the AR(2) process

$$x[n] = 0.6x[n-1] + 0.2x[n-2] + q[n], \quad q[n] \sim \mathcal{N}(0, \sigma_q^2)$$

Our task is prediction, so $\hat{x}[n] = 0.6x[n-1] + 0.2x[n-2]$



Left: Learning curves for varying σ_q^2. The best we can do is $J_{min} = \sigma_q^2$

Right: Evolution of weight error vector (misalignment) $\mathbf{v}(n) = \mathbf{w}(n) - \mathbf{w}_o$

Summary of performance measures

Prediction gain: (a cumulative measure - no notion of time)

$$R_p = 10\log\frac{\hat{\sigma}_x^2}{\hat{\sigma}_e^2} \qquad \text{ratio of signal and error powers}$$

We may calculate R_p for the whole signal, or just in the steady state.

Mean square error: MSE is evaluated over time (learning curve)

$$MSE(k) = 10\log e^2(k) = 10log|e(k)|^2$$

Misalignment: that is "mean square weight error" $\mathbf{v}^T(k)\mathbf{v}(k)$, given by

$$10\log \parallel \mathbf{w}(k) - \mathbf{w}_{opt} \parallel_2^2 = 10\log\mathbf{v}^T(k)\mathbf{v}(k), \qquad \text{where } \mathbf{v}(k) = \mathbf{w}(k) - \mathbf{w}_{opt}(k)$$

Normalised versions of MSE and misalignment: for example

$$10\log\frac{\parallel \mathbf{w}(k) - \mathbf{w}_{opt} \parallel_2^2}{\parallel \mathbf{w}(k) \parallel_2^2}$$

Excess MSE, J_{ex}. As $J[\infty] = J_{min} + J_{ex}[\infty] \;\Rightarrow\; J_{ex}[\infty] = J[\infty] - J_{min}$

Misadjustment: ratio of excess MSE and minimum MSE, $\mathcal{M} = J_{ex}(\infty)/J_{min}$

Improving the convergence and stability of LMS: The Normalised Least Mean Square (NLMS)

Uses an adaptive step size by normalising μ by the signal power in the filter memory, that is

from fixed $\mu \quad \rightsquigarrow \quad$ data adaptive $\mu(n) = \dfrac{\mu}{\mathbf{x}^T(n)\mathbf{x}(n)} = \dfrac{\mu}{\| \mathbf{x}(n) \|_2^2}$

Can be derived from the Taylor Series Expansion of the output error

$$e(n+1) = e(n) + \sum_{k=1}^{p} \frac{\partial e(n)}{\partial w_k(n)} \Delta w_k(n) + \underbrace{\text{higher order terms}}_{=0,\ since\ the\ filter\ is\ linear}$$

Since $\partial e(n)/\partial w_k(n) = -x_k(n)$ and $\Delta w_k(n) = \mu e(n) x_k(n)$, we have

$$e(n+1) = e(n)\Big[1 - \mu \sum_{k=1}^{p} x_k^2(n)\Big] = \big[1 - \mu \| \mathbf{x}(n) \|_2^2\big] \quad \text{as } \Big(\sum_{k=1}^{p} x_k^2 = \| \mathbf{x} \|_2^2\Big)$$

Set $e(n+1) = 0$, to arrive at the step size which minimizes the error:

$\mu = \dfrac{1}{\| \mathbf{x}(n) \|_2^2} \qquad$ however, in practice we use $\qquad \mu(n) = \dfrac{\mu}{\| \mathbf{x}(n) \|_2^2 + \varepsilon}$

where $0 < \mu < 2$, $\mu(n)$ is time-varying, and ε is a small "regularisation" constant, added to avoid division by 0 for small values of input

Effects of normalisation ↬ also run 'nnd10nc in Matlab'

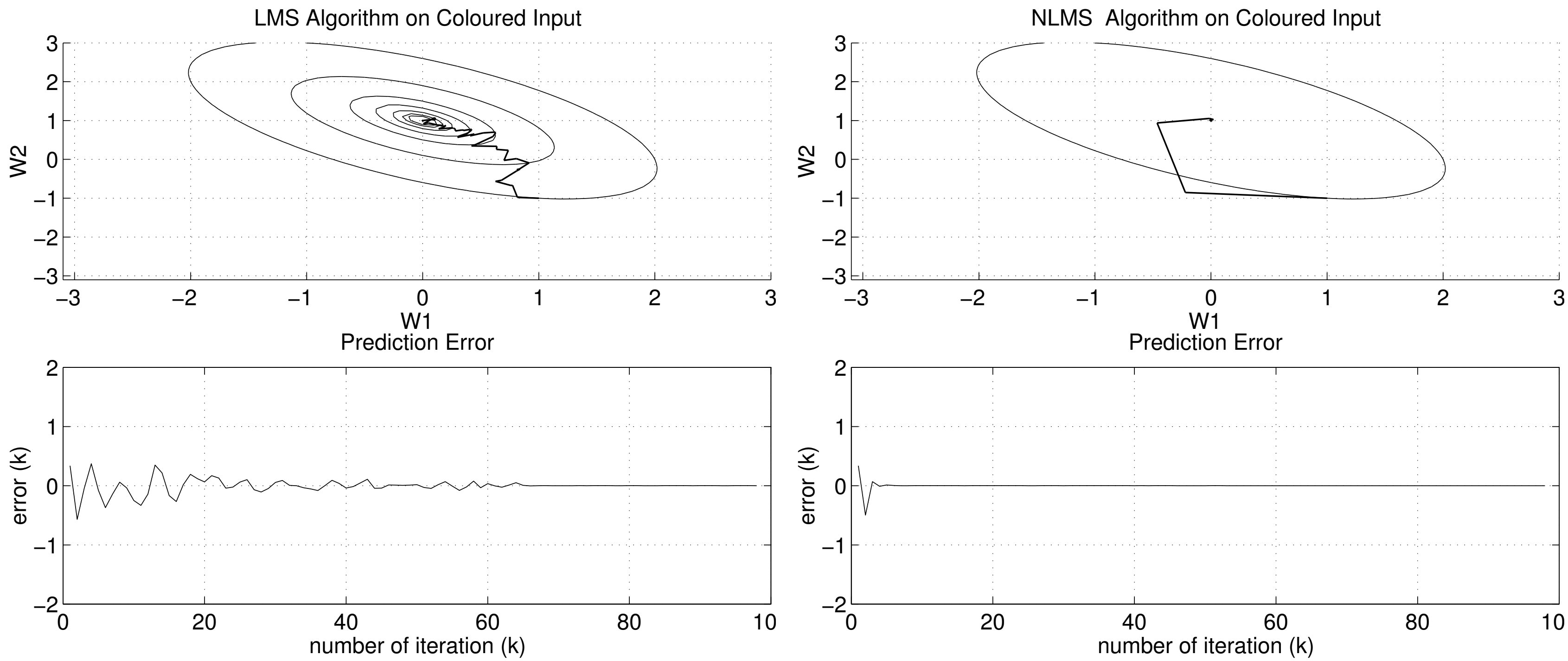
NLMS is independent of signal power ⇝ suitable for real-world changing environ.

○ **"Regularises"** the error surface by dividing μ by the tap input power

$$\mathbf{x}_{NLMS}(k) = \frac{\mathbf{x}_{LMS}(k)}{\| \mathbf{x}_{LMS}(k) \|_2^2} \qquad 1/ \| \mathbf{x}_{LMS}(k) \|_2^2 \ \text{is a primitive} \ \mathbf{R}^{-1}$$

☞ Conditioning of the tap input correlation matrix $\mathbf{R}_{xx}$ ⇝ the error surface becomes parabolic ⇝ faster convergence

○ **Both LMS and NLMS converge to the same Wiener solution**



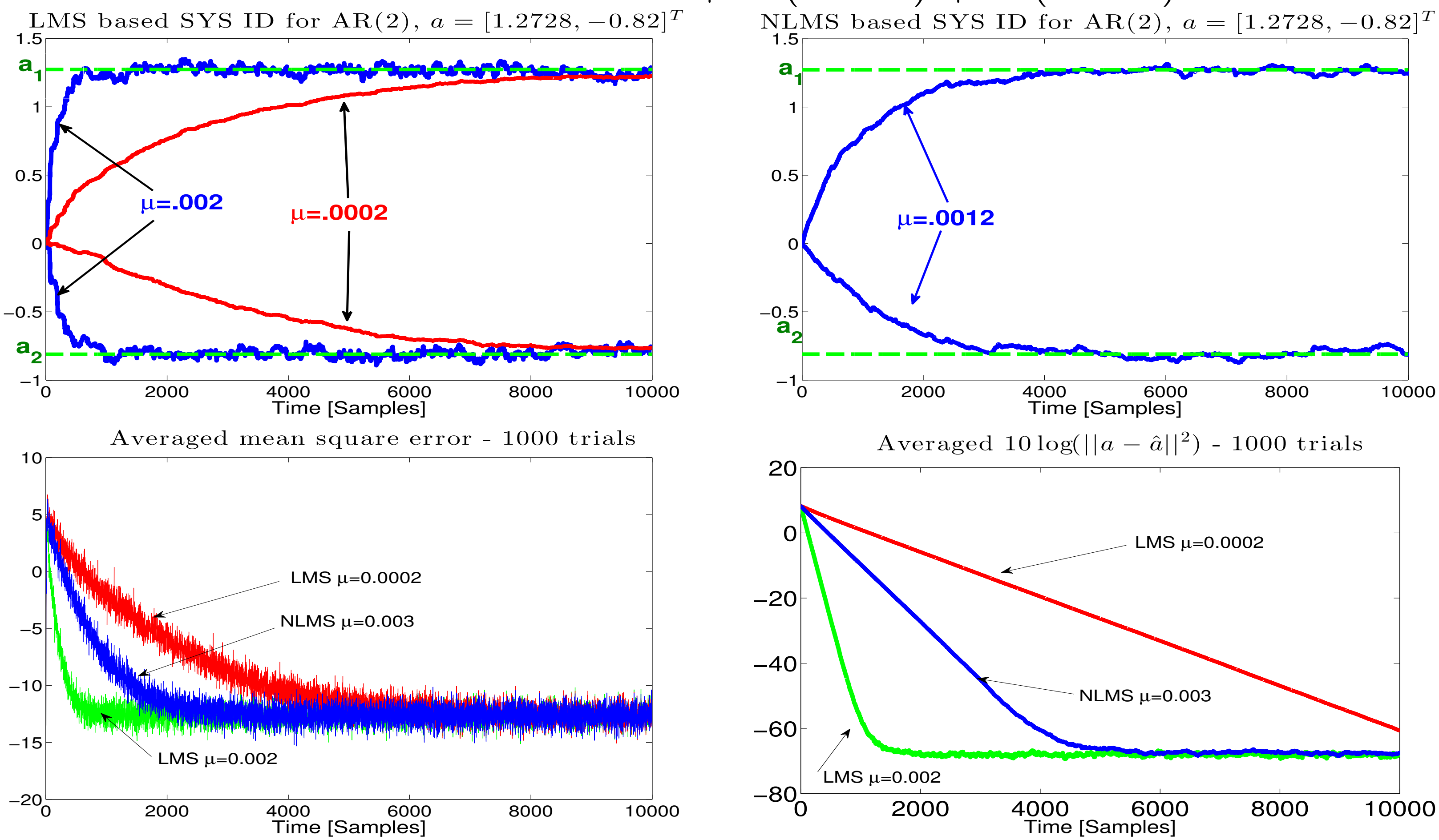
Example 1: Learning curves and performance measures

Task: Adaptively identify an AR(2) system given by

$x(n) = 1.2728x(n-1) - 0.81x(n-2) + q(n), \quad q \sim \mathcal{N}(0, \sigma_q^2)$

LMS and NLMS: $\hat{x}(n) = w_1(n)x(n-1) + w_2(n)x(n-2)$ **system model**

NLMS weights (i=1,2): $w_i(n+1) = w_i(n) + \dfrac{\mu}{\varepsilon + x^2(n-1) + x^2(n-2)} e(n)x(n-i)$



Some rules of thumb in LMS parameter choice

The **steady state the misadjustment** for the LMS algorithms is given by

$$\mathcal{M} \approx \frac{1}{2}\mu\, N\, \sigma_x^2$$

- It is proportional to learning rate μ, so the smaller the μ the lower the $\mathcal{M}$; however for fast initial convergence we need a relatively large μ in the beginning of adaptation;
- It is proportional to filter length N, so the shorter the filter the better; however, a short N may not be able to capture the dynamics of the input;
- It depends on signal power σ_x^2; however, the signal power in filter memory (tap input power) changes from sample to sample.

To make the adaptive filter independent of the power in the tap input we use the Normalized LMS (NLMS)

To have an optimal stepsize in nonstationary environments we may employ adaptive learning rates within LMS

Algorithms with an Adaptive Stepsize

We will study three classes of such algorithms:

- **Determinisic,** which provide large learning rate in the beginning of adaptation for fast convergence, and small learning rate at the end of adaptation for good steady state properties (remember $\mathcal{M} \sim \mu N \sigma_x^2$), such as **simulated annealing algorithms.**
- **Stochastic** based on $\frac{\partial J}{\partial \mu}$, that is "gradient adaptive stepsize" **(GASS)**;
- **Stochastic** based on the adaptive regularization factor ε within the NLMS, such as the Generalized Normalized Gradient Descent **(GNGD)**;

The general form of such LMS updates with an adaptive stepsize then becomes

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \eta(k)e(k)\mathbf{x}(k)$$

where $\eta(k)$ is the adaptive learning rate, and $\eta(k) = \mu(k)$ for GASS algorithms and $\eta(k) = \frac{\mu}{\|\mathbf{x}(k)\|_2^2 + \epsilon(k)}$ for GNGD.

Deterministic learning rate update: Simulated annealing

(also knows as "search then converge" (STC) algorithms

As the misadjustment $\mathcal{M} \sim \mu$, select an automatic scheme to choose μ initially large for fast convergence and then to reduce along the iterations it for small misadjustment.

- "Cooling schedule" (think iron)
- A STC stepsize $(\tau = const)$

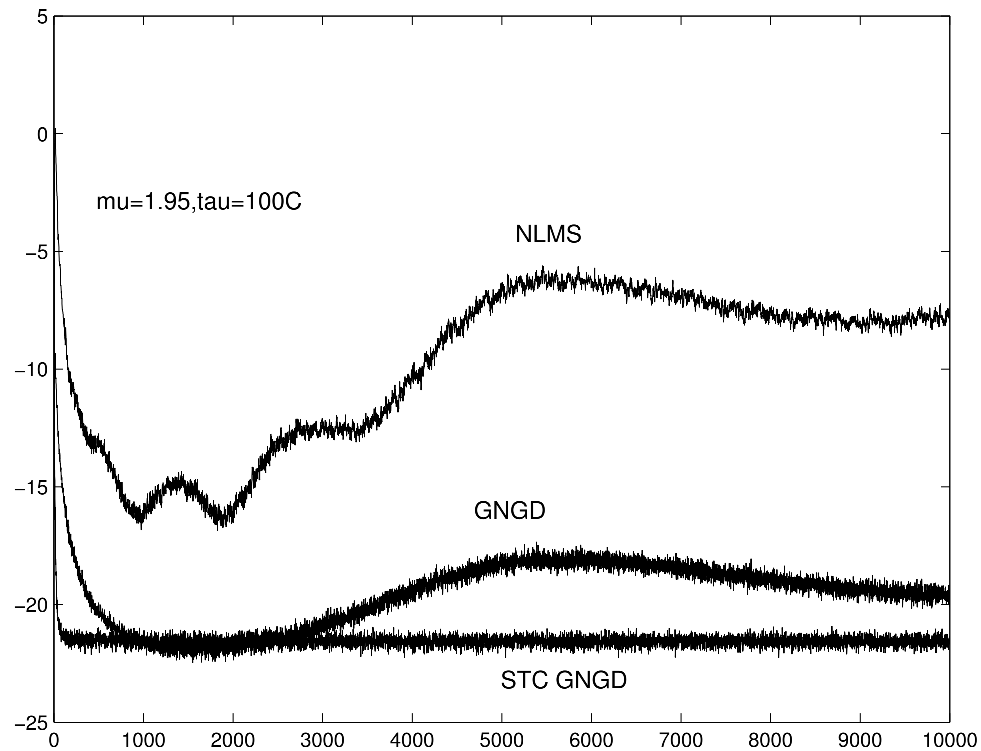
$$\eta(k) = \frac{\mu}{1 + k/\tau}$$

$$\eta(k) \rightarrow 0 \quad \text{when} \quad n \rightarrow \infty$$

- A second order cooling schedule

$$\eta(k) = \eta_0 \frac{1 + \frac{c}{\eta_0}\frac{k}{\tau}}{1 + \frac{c}{\eta_0}\frac{k}{\tau} + \tau\frac{k^2}{\tau^2}}$$

- **Small misadjustment** as compared with LMS
- Not suitable for nonstationary environments



Learning curves: pred. of a nonlin. signal

Gradient Adaptive Stepsize Algorithms (GASS)

Start from $\mu(k+1) = \mu(k) - \rho \nabla_\mu E(k)_{|\mu=\mu(k-1)}$ where ρ is a stepsize.

$$\nabla_\mu E(k) = \frac{1}{2}\frac{\partial e^2(k)}{\partial e(k)}\frac{\partial e(k)}{\partial y(k)}\frac{\partial y(k)}{\partial \mathbf{w}(k)}\frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)} = -e(k)\mathbf{x}^T(k)\frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)}$$

Denote $\boldsymbol{\gamma}(k) = \frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)}$ **to obtain** $\mu(k+1) = \mu(k) + \rho e(k)\mathbf{x}^T(k)\boldsymbol{\gamma}(k)$

Recall that $\quad \mathbf{w}(k) = \mathbf{w}(k-1) + \mu(k-1)e(k-1)\mathbf{x}(k-1)$

$$\begin{aligned}\frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)} &= \frac{\partial \mathbf{w}(k-1)}{\partial \mu(k-1)} + e(k-1)\mathbf{x}(k-1) + \mu(k-1)\frac{\partial e(k-1)}{\partial \mu(k-1)}\mathbf{x}(k-1) \\ &+ \mu(k-1)e(k-1)\underbrace{\frac{\partial \mathbf{x}(k-1)}{\partial \mu(k-1)}}_{=0 \quad as \quad \mathbf{x} \neq f(\mu)}\end{aligned}$$

$$\frac{\partial e(k-1)}{\partial \mu(k-1)} = \frac{\partial\left(d(k-1) - \mathbf{x}^T(k-1)\mathbf{w}(k-1)\right)}{\partial \mu(k-1)} = -\mathbf{x}^T(k-1)\frac{\partial \mathbf{w}(k-1)}{\partial \mu(k-1)}$$

GASS $\looparrowright$ Benveniste, Farhang, Mathews

Start from $\nabla_{\mu(k-1)}E(k) = -e(k)\mathbf{x}^T(k)\boldsymbol{\gamma}(k)$

Benveniste algorithm: The correct expression[2] for the gradient $\nabla_{\mu}E(k)$

$$\boldsymbol{\gamma}(k) \quad = \big[\underbrace{\mathbf{I} - \mu(k-1)\mathbf{x}(k-1)\mathbf{x}^T(k-1)}_{filtering \quad term}\big]\boldsymbol{\gamma}(k-1) + e(k-1)\mathbf{x}(k-1)$$

Farhang-Ang algorithm: use a low pass filter with a fixed coefficient α

$$\boldsymbol{\gamma}(k) = \alpha\boldsymbol{\gamma}(k-1) + e(k-1)\mathbf{x}(k-1), \quad 0 \le \alpha \le 1$$

Mathews' algorithm: assume $\alpha = 0$ (we now only have a noisy gradient)

$$\boldsymbol{\gamma}(k) = e(k-1)\mathbf{x}(k-1), \quad 0 \le \alpha \le 1$$

[2]For a small value of μ, assume $\mu(k-1) \approx \mu(k)$ and therefore $\frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)} \approx \frac{\partial \mathbf{w}(k)}{\partial \mu(k)} = \boldsymbol{\gamma}(k)$.

Introducing robustness into NLMS: The GNGD

- For close to zero $\mathbf{x}(k)$, instability of NLMS as $\eta \sim 1/\parallel \mathbf{x} \parallel_2^2$
- Therefore, we need to add a regularistion factor ε, as

$$\eta(k) = \frac{\mu}{\parallel \mathbf{x}(k) \parallel_2^2 + \varepsilon(k)}$$

- This regularisation factor can be either fixed or made gradient adaptive

$$\begin{aligned}
\varepsilon(k+1) &= \varepsilon(k) - \rho \nabla_{\varepsilon} J(k) \\
\frac{\partial J(k)}{\partial \varepsilon(k-1)} &= \frac{\partial J(k)}{\partial e(k)} \frac{\partial e(k)}{\partial y(k)} \frac{\partial y(k)}{\partial \mathbf{w}(k)} \frac{\partial \mathbf{w}(k)}{\partial \eta(k-1)} \frac{\partial \eta(k-1)}{\partial \varepsilon(k-1)} \\
\varepsilon(k) &= \varepsilon(k-1) - \rho\, \mu \frac{e(k)e(k-1)\mathbf{x}^T(k)\mathbf{x}(k-1)}{\left(\parallel \mathbf{x}(k-1) \parallel_2^2 + \varepsilon(k-1)\right)^2}
\end{aligned}$$

The NLMS with an adaptive regularisation factor $\varepsilon(k)$ is called the Generalised Normalised Gradient Descent (GNGD)

Simulations: Linear adaptive prediction

Learning curves for GSS algorithms – GNGD very fast and robust to μ values

Learning curves, $10\log|e(n)|^2$, used for performance evaluation

Learning curves were produced by "Monte Carlo" simulations (averaging 100 independent trials) – to make them smooth

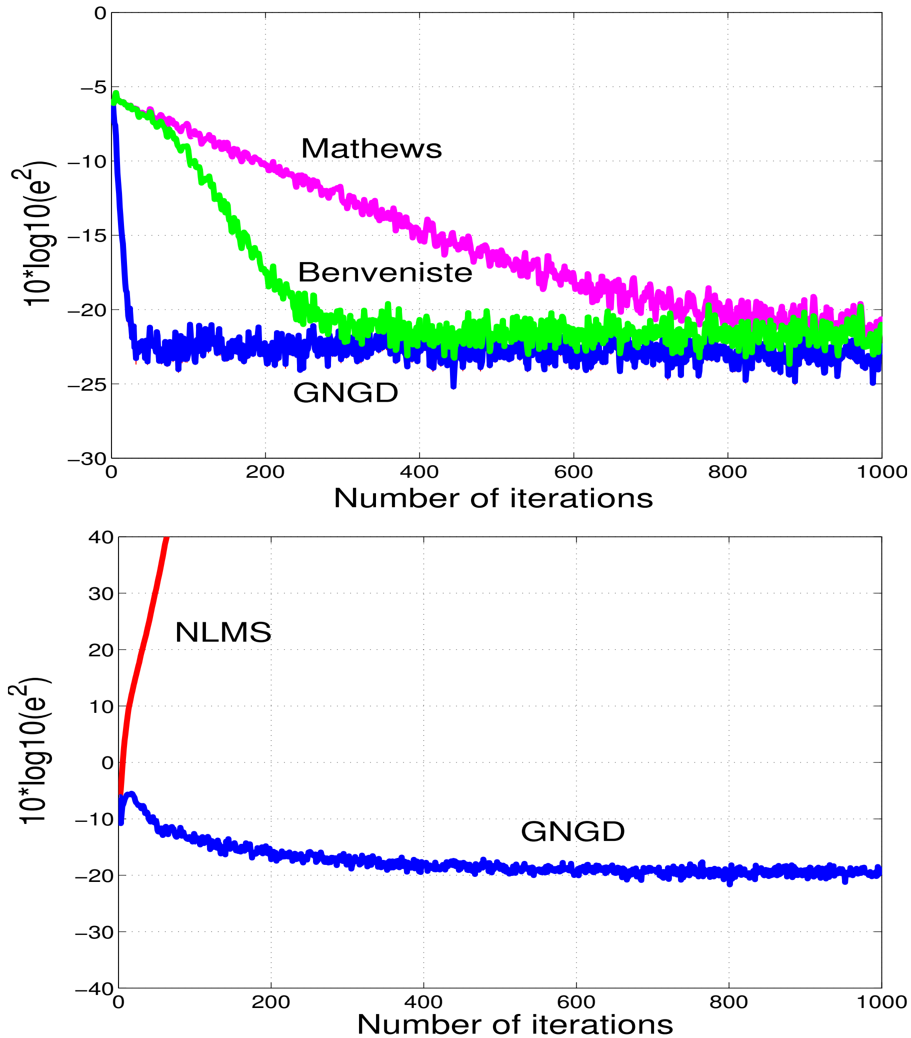
○ **The GNGD** $\looparrowright$ ***"nonlinear"*** update of $\mu(n)$ (gradient adaptive regularisation factor $\varepsilon(n)$ in NLMS), $\mu(n) \sim \nabla_{\varepsilon} J(n)$

○ **GASS** algorithms $\looparrowright$ ***"linear"*** updates of $\mu(n)$, $\mu(n) \sim \nabla_{\mu} J(n)$

GNGD was stable even for $\mu = 2.1$ $\looparrowright$ outside stability bounds of NLMS and LMS (bottom). GASS algorithms may have good steady state properties.

Top: convergence curves for a linear signal

Bottom: convergence curves for $\mu = 2.1$

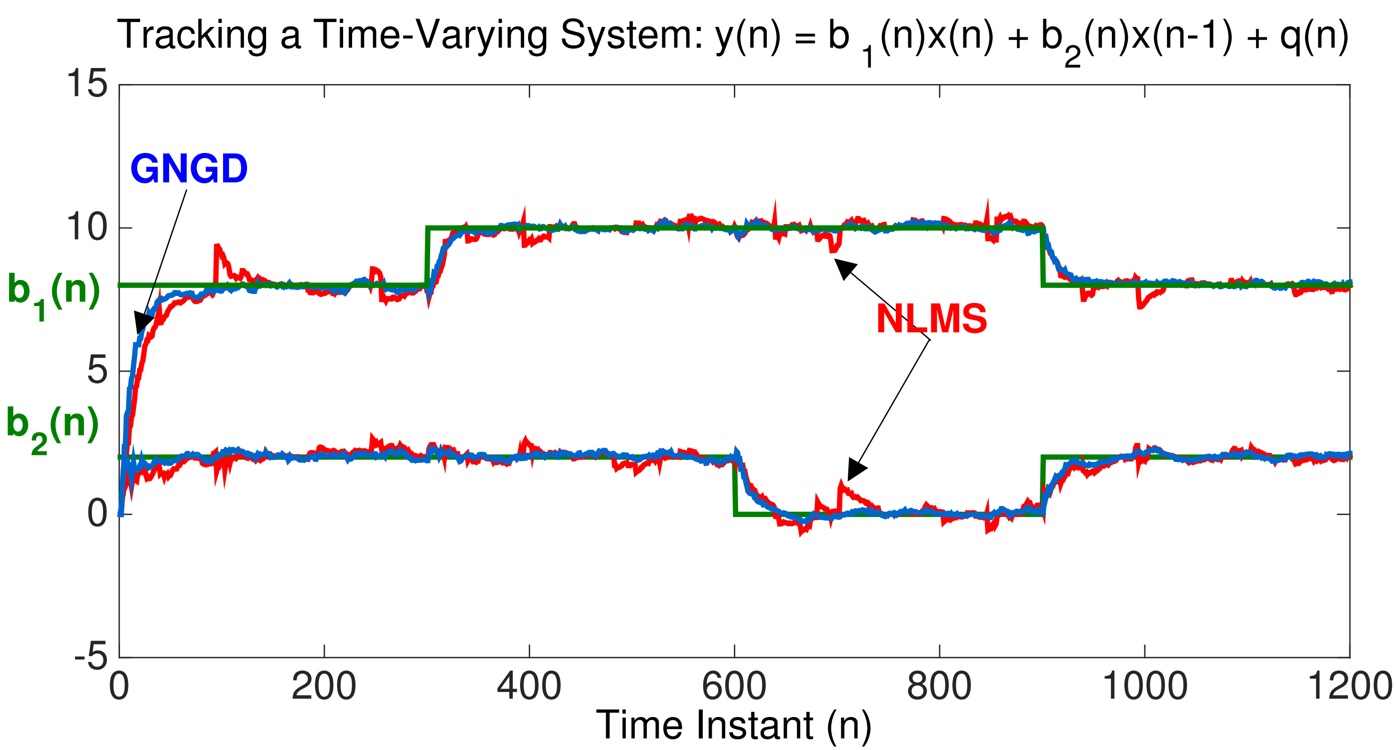


Performance in nonstationary environments

System to be identified: An AR model with time-varying coefficients b_1 and b_2. The driving noise $q \sim \mathcal{N}(0, \sigma_q^2)$

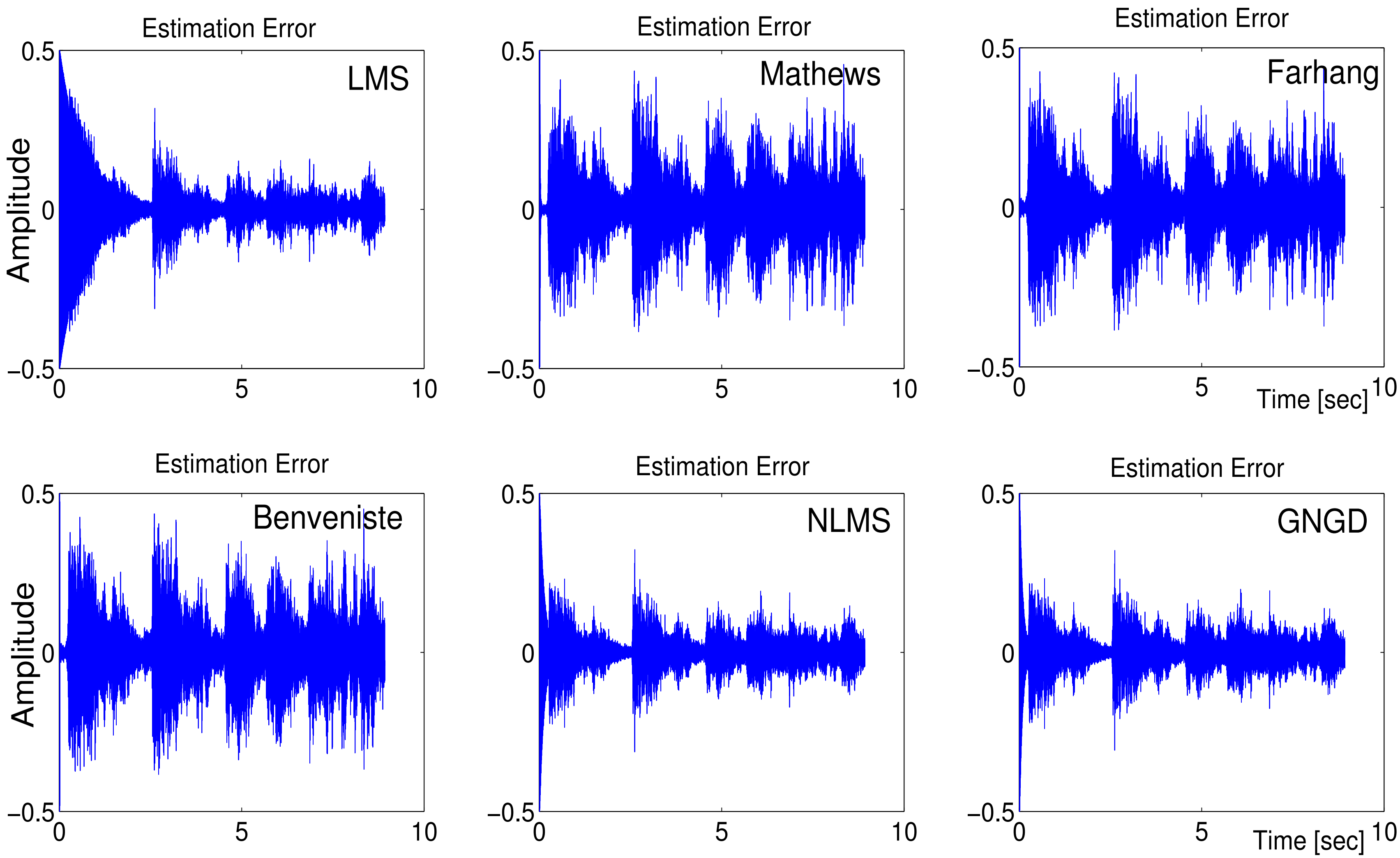
Wiener solution: Considers the whole 1200 data points, does not capture changes in b_1 and b_2, and gives an "average" solution $\hat{\mathbf{b}} = [1.5,\ 9.0]^T$

Learning algorithms: GNGD improved on the performance of NLMS



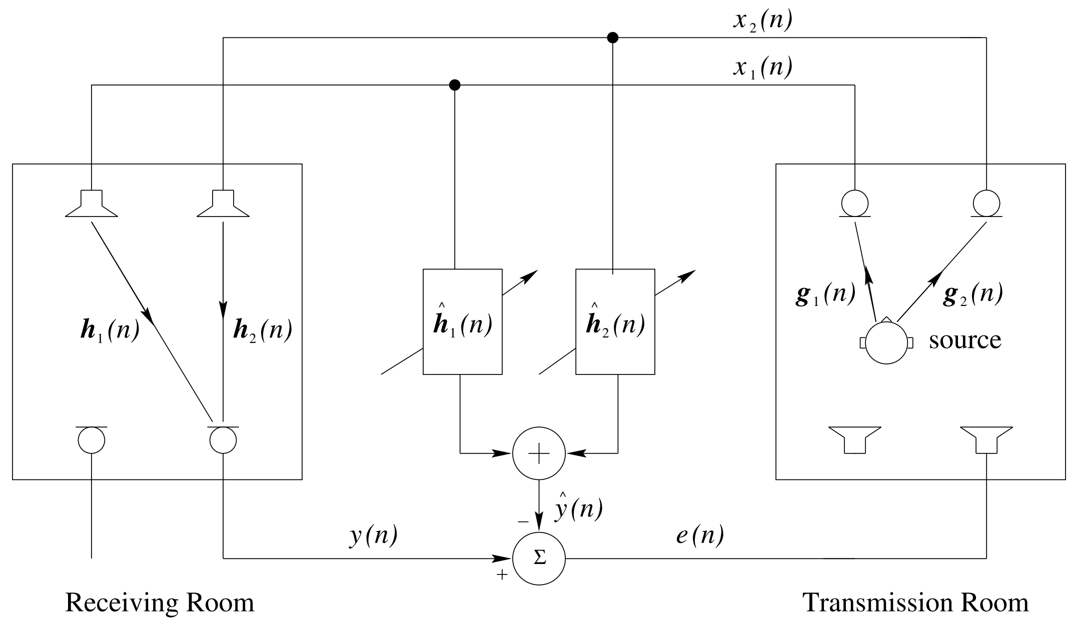
ALE for music, variable stepsize algs. All_in_One_ALE_Sin_Noise

ALE parameters: $\Delta = 100$, filter length $N = 32$ (both can be varied)



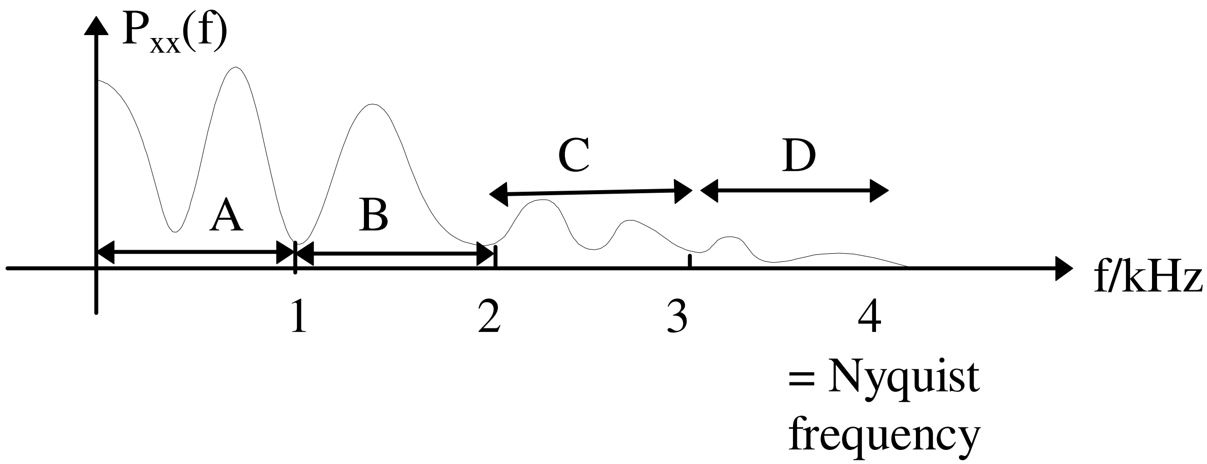
All the algorithms suppress the line noise, some better than other

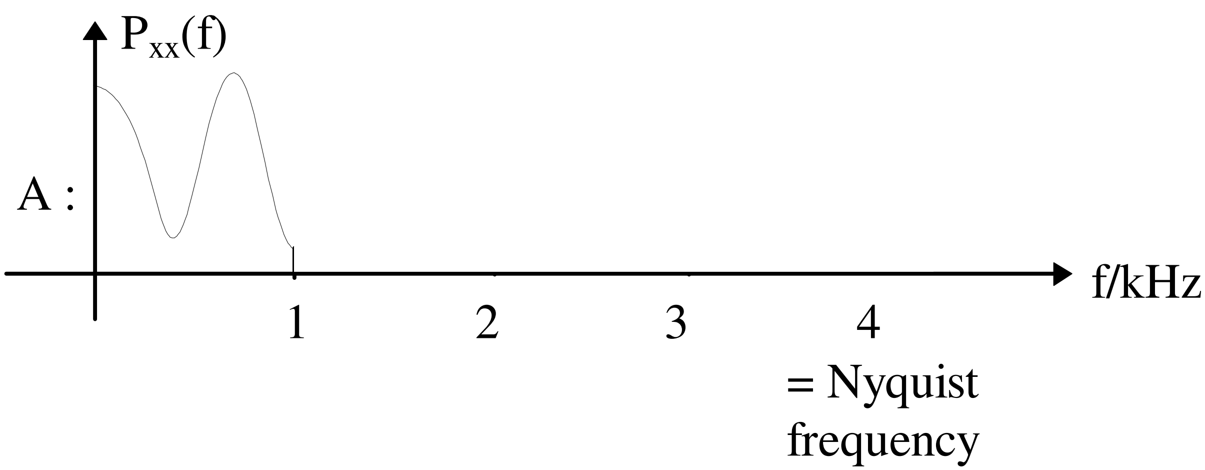
Acoustic Echo Cancellation (AEC) problem



- A measured microphone signal contains two signals: the near–end speech signal and the far–end echoed signal
- The goal is to remove the far–end echoed speech signal from the microphone so that only the near–end speech signal is transmitted
- To that end, we need the knowledge of the room impulse response

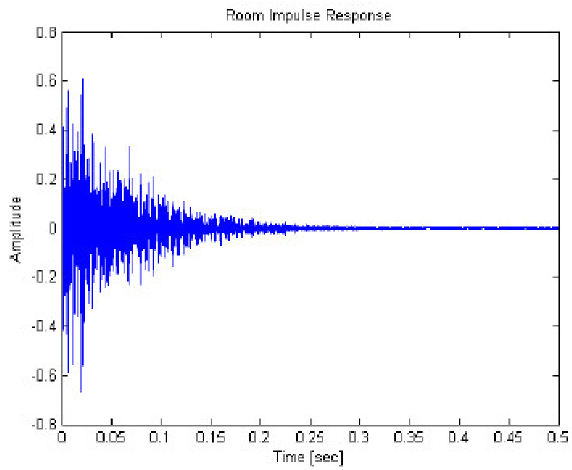
In terms of the spectrum



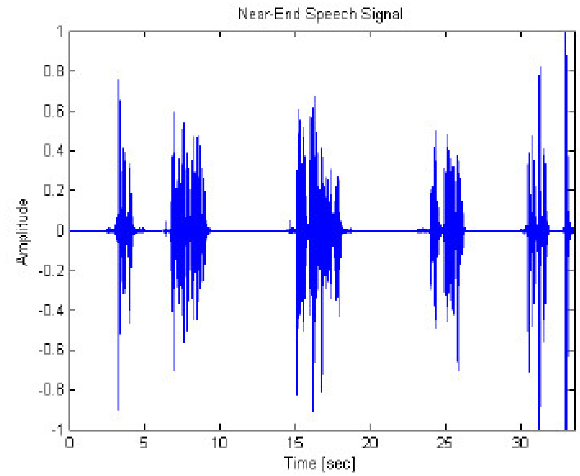


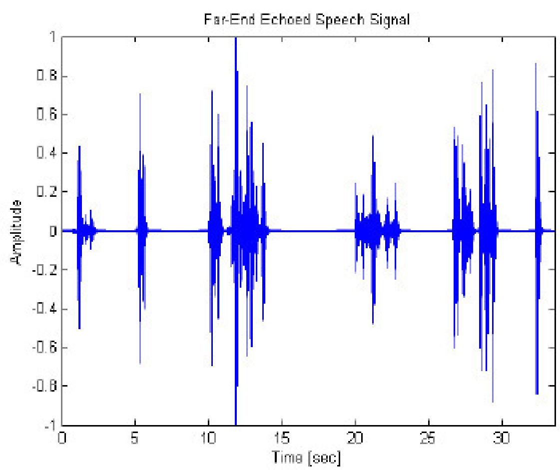
Acoustic echo cancellation problem: Signals

Room Impulse Response

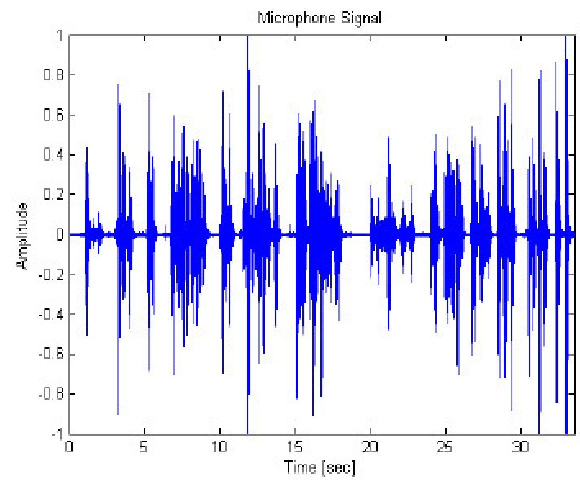


Near–end speech



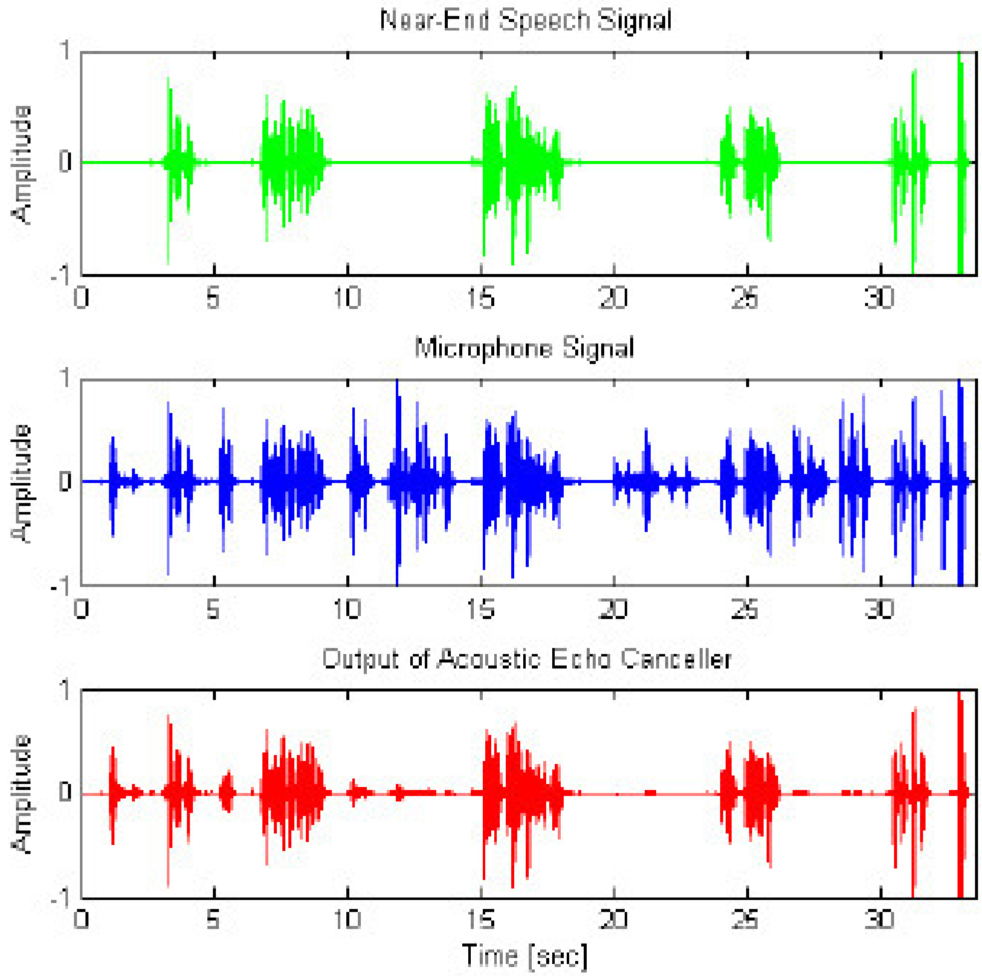


Far–end echoed speech



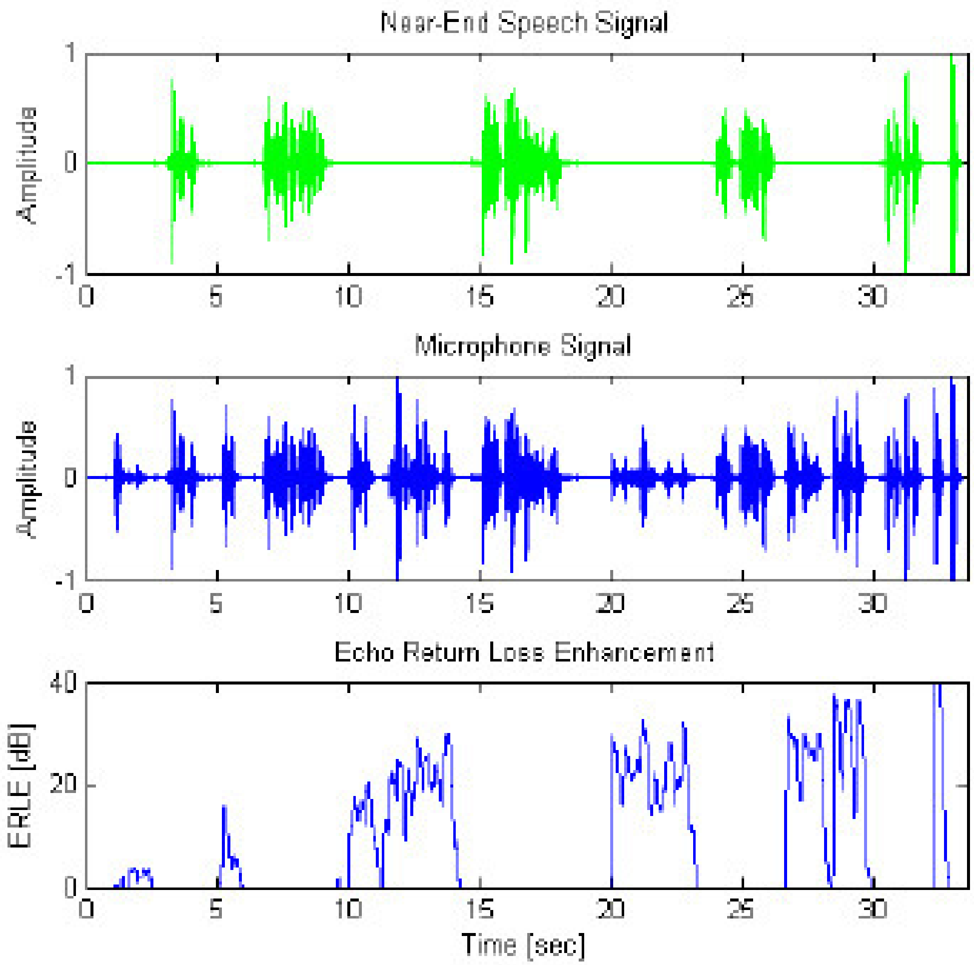
Microphone signal

AEC – Cancellation results



Clearly, the echo has been removed

AEC – Echo Return Loss Enhancement (ERLE)



ERLE: a smoothed measure of echo attenuation ($10 * log\frac{var(loudspeaker)}{var(error)}$ dB)

Summary

- Basics of adaptive filtering
- Duality with Spectrum Estimation
- Principle of Stepest Descent – Gradient learning
- LMS - the workhorse of adaptive filtering
- Convergence in the mean, mean square and steady state
- Error surfaces and divergence
- Prediction application
- Acoustic echo cancellation application

Appendix: Reducing computational complexity: Sign algorithms

Simplified LMS, derived based on $sign(e) = |e|/e$ and $\nabla|e| = sign(e)$.

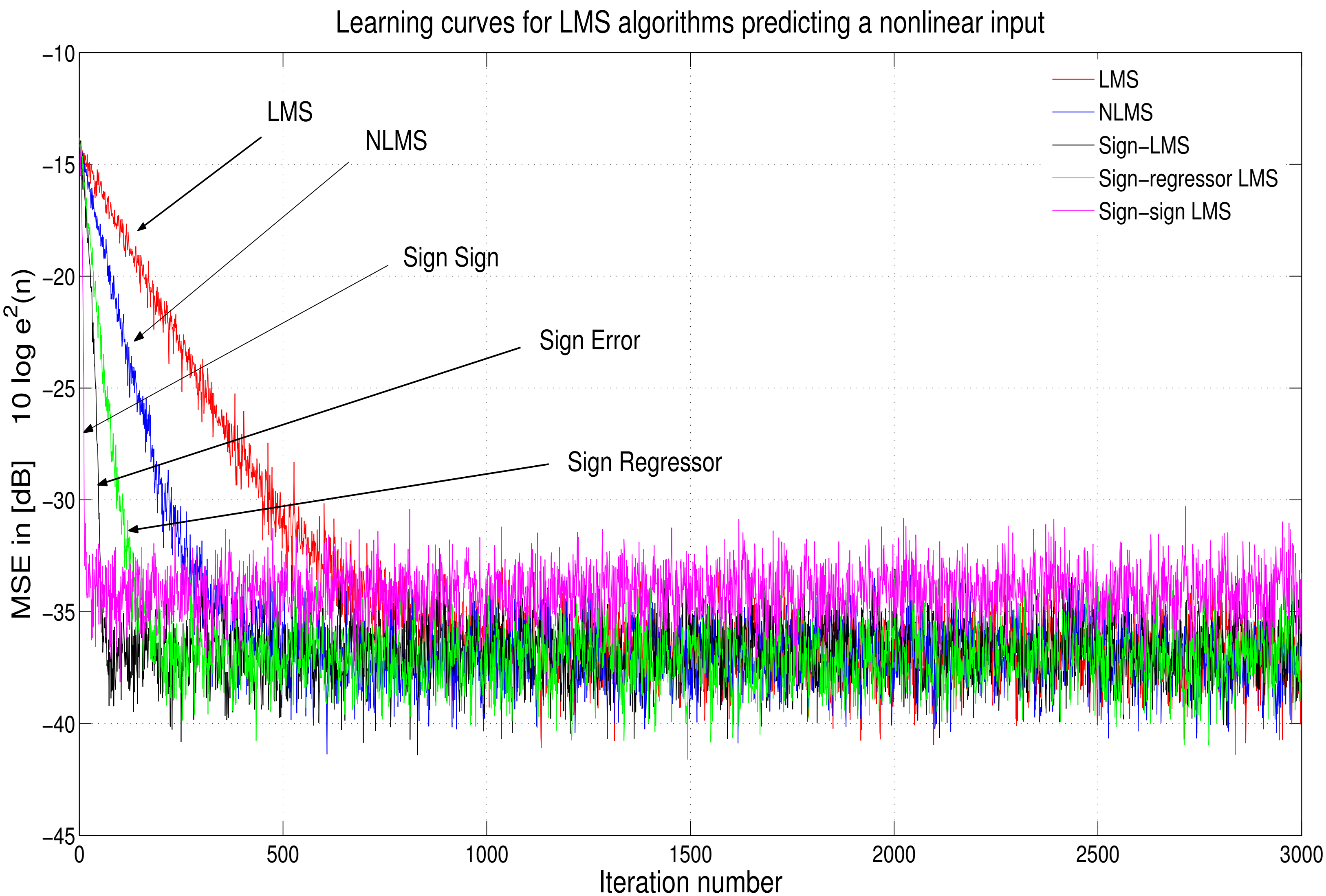
Good for hardware and high speed applications.

- **The Sign Algorithm** (The cost function here is $J[n] = |e[n]|$)
  Replace $e(n)$ by its sign to obtain
  $$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu sign(e(n))\mathbf{x}(n)$$

- **The Signed Regressor Algorithm**
  Replace $\mathbf{x}(n)$ by $sign(\mathbf{x}(n))$
  $$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n) sign(\mathbf{x}(n))$$
  Performs much better than the sign algorithm.

- **The Sign-Sign Algorithm**
  Combines the above two algorithms
  $$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu sign(e(n)) sign(\mathbf{x}(n))$$

Appendix: Performance of sign algorithms



Appendix: A simple derivation of Mathews' GASS algorithm

A gradient adaptive learning rate $\mu(k)$ can be introduced into the LMS as

$$\mu(k+1) \quad = \quad \mu(k) - \rho \nabla_{\mu} J(k)_{|\mu=\mu(k-1)}$$

where parameter ρ denotes the stepsize. Thus, we have

$$\nabla_{\mu} J(k) = \frac{1}{2} \frac{\partial e^2(k)}{\partial e(k)} \frac{\partial e(k)}{\partial y(k)} \frac{\partial y(k)}{\partial \mathbf{w}(k)} \frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)} = -e(k)\mathbf{x}^T(k) \frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)}$$

Since

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu(k-1)e(k-1)\mathbf{x}(k-1) \quad \Rightarrow \quad \frac{\partial \mathbf{w}(k)}{\partial \mu(k-1)} = e(k-1)\mathbf{x}(k-1)$$

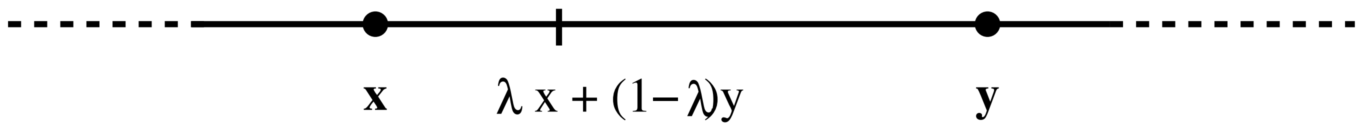
The GASS variant of the LMS algorithm thus becomes

$$\begin{aligned} \mathbf{w}(k+1) &= \mathbf{w}(k) + \mu(k)e(k)\mathbf{x}(k) \\ \mu(k+1) &= \mu(k) + \rho e(k)e(k-1)\mathbf{x}^T(k)\mathbf{x}(k) \end{aligned}$$

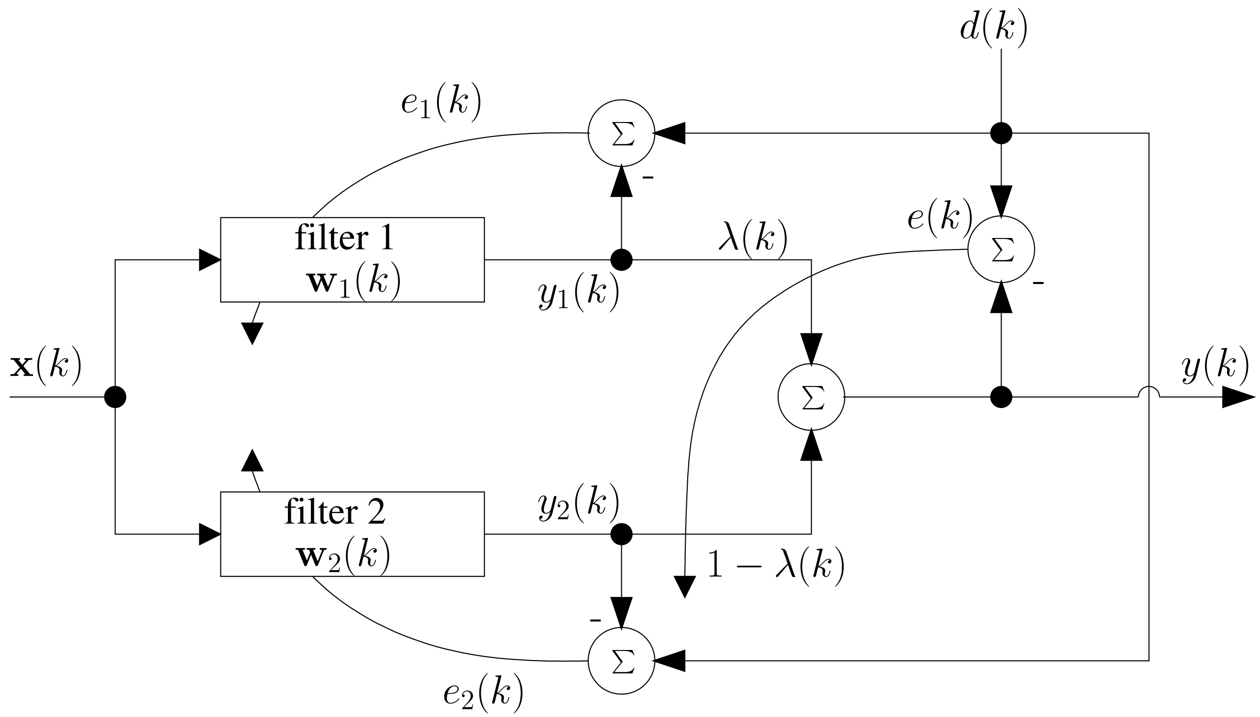
For the derivation of other members of the GASS class, see the Appendix.

Appendix: Collaborative adaptive filters: A hybrid filtering configuration

Virtues of Convex Combination ($\lambda \in [0, 1]$)



Can we have both fast convergence and small steady state error automatically?



Typically two LMS algorithms, one fast (large μ) and one slow (small μ)

Adaptation of Mixing Parameter λ

To preserve the inherent characteristics of the subfilters, the constituent subfilters are each updated independently using their own errors $e_1(k)$ and $e_2(k)$, while the parameter λ is updated based on the overall error $e(k)$.

The convex mixing parameter $\lambda(k)$ is updated using the standard gradient adaptation

$$\lambda(k+1) = \lambda(k) - \mu_\lambda \nabla_\lambda E(k)_{|\lambda=\lambda(k)}$$

where μ_λ is the adaptation step-size. The λ update can be shown to be

$$\begin{aligned} \lambda(k+1) &= \lambda(k) - \frac{\mu_\lambda}{2}\frac{\partial e^2(k)}{\partial \lambda(k)} \\ &= \lambda(k) + \mu_\lambda e(k)\big(y_1(k) - y_2(k)\big) \end{aligned}$$

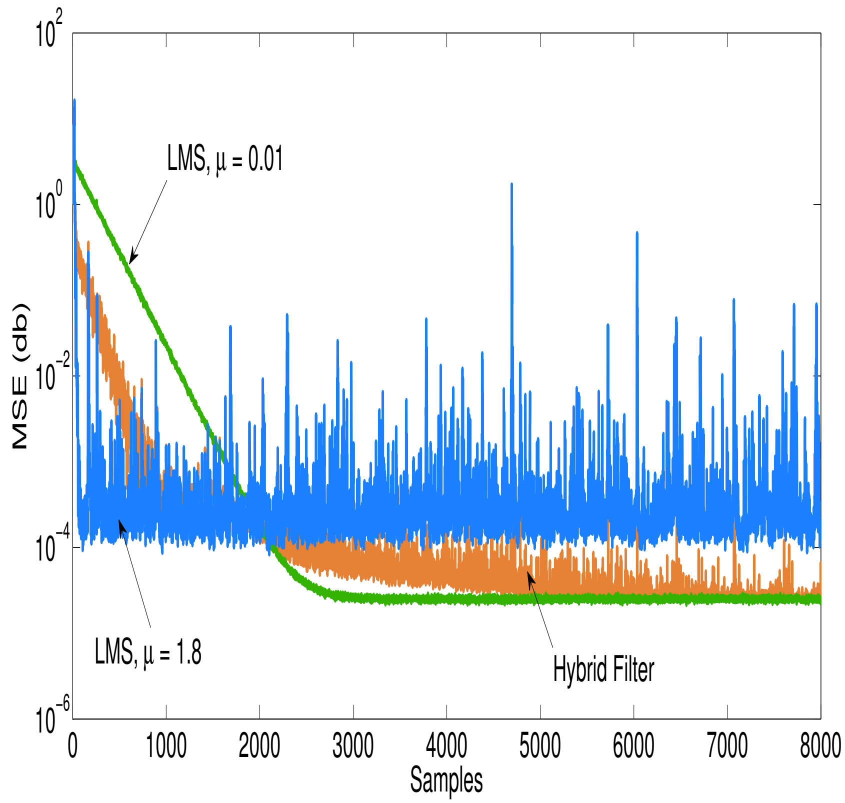
To ensure the combination of adaptive filters remains a convex function it is critical λ remains within the range $0 \leq \lambda(k) \leq 1$, a hard limit on the set of allowed values for $\lambda(k)$ was therefore implemented.

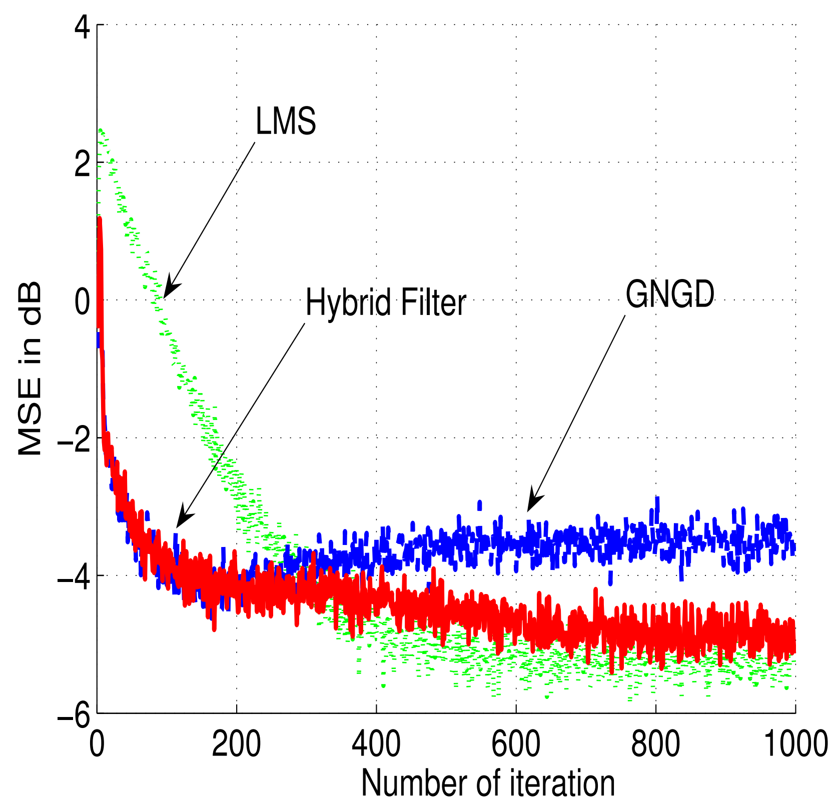
Performance of hybrid filters – prediction setting

consider an LMS/GNGD hybrid – GNGD is fast, LMS with small μ has good $\mathcal{M}$

Hybrid attempts to follow the subfilter with better performance.

If one of the subfilters diverges, hybrid filters still converges.





Learn. curves for pred.: Left $\looparrowright$ linear signal   Right $\looparrowright$ nonlinear signal