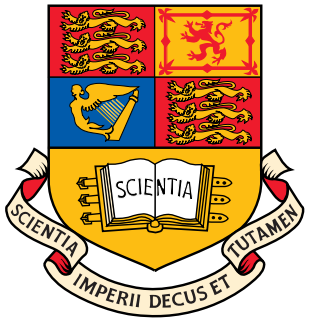
Spectral Estimation & Adaptive SP
Introduction to Adaptive Filters

Danilo Mandic

room 813, ext: 46271



Department of Electrical and Electronic Engineering
Imperial College London, UK

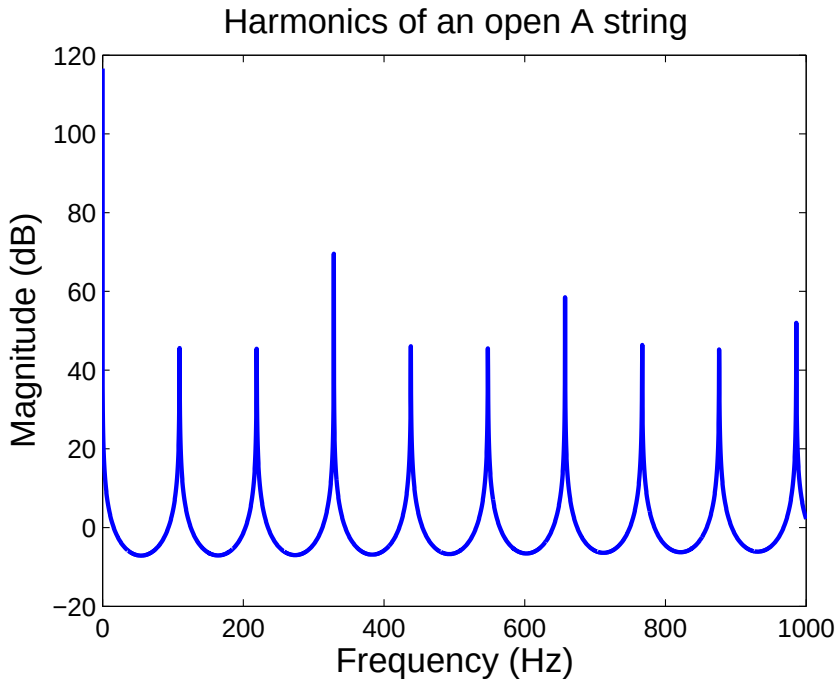
d.mandic@imperial.ac.uk, URL: www.commsp.ee.ic.ac.uk/∼mandic

Aims

- To introduce the concept of adaptive filtering
- Parallels (duality) between spectrum estimation and adaptive filtering
- To introduce adaptive filtering architectures
- Supervised and blind adaptive filtering
- The concept of steepest descent
- To introduce the Least Mean Square (LMS) algorithm
- Error surface, learning rate and convergence
- Practical applications

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



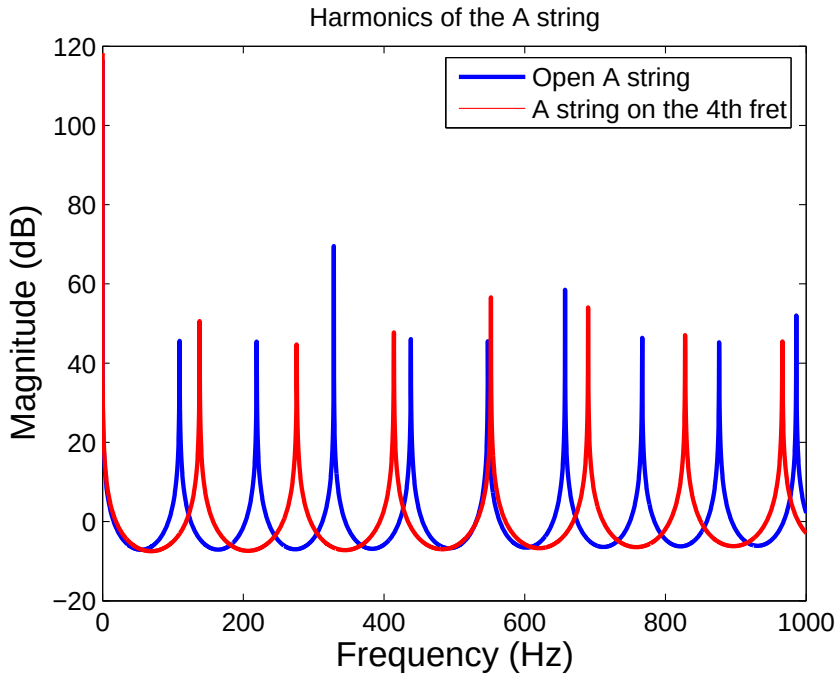
Basic frequency and Harmonics: Two ways to generate these

- ☐ 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

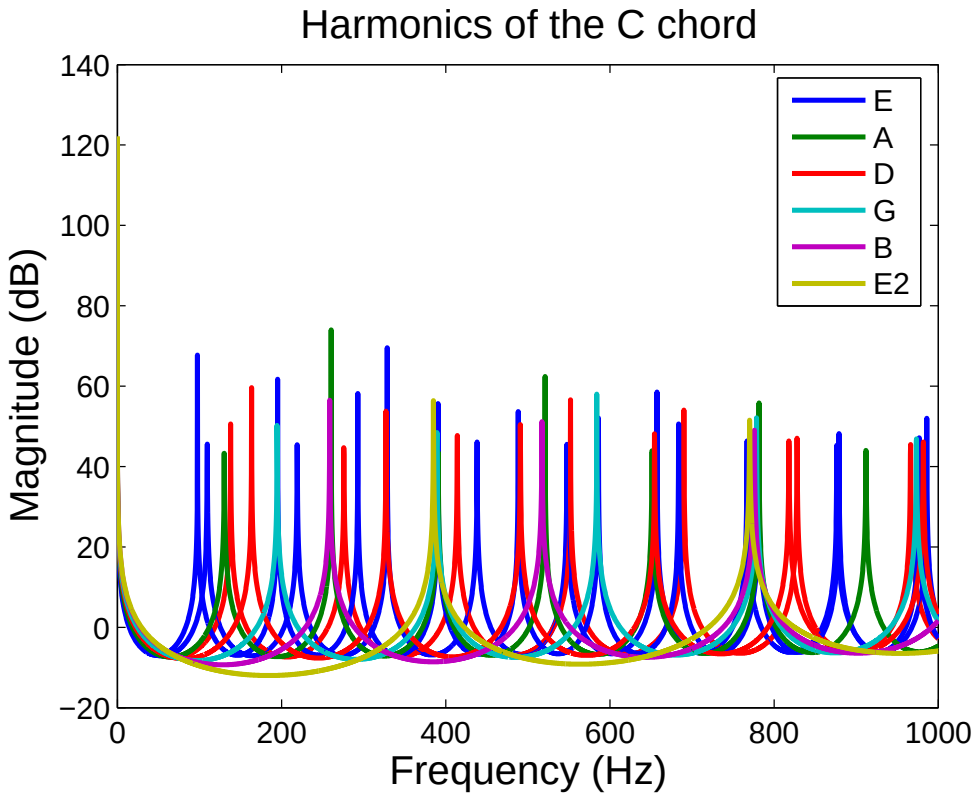


Basic frequencies and Harmonics

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

Things getting more complicated: Chords

Generate_Chord_and_Play.m

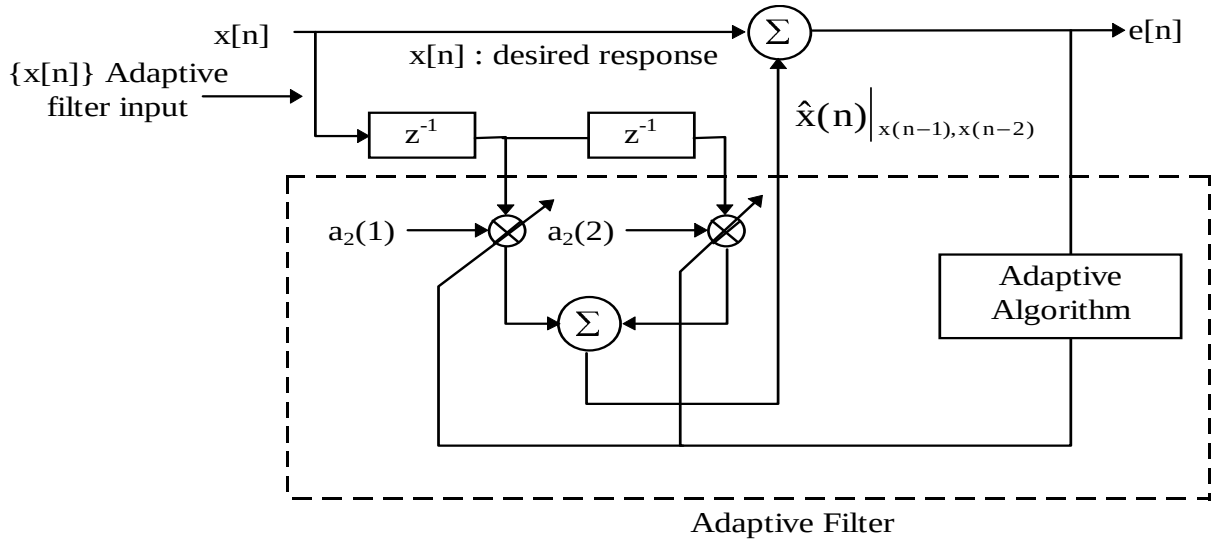


Many more parameters to estimate: Basis for music synthesis

- Spectrum based – resolution problems
- Digital filters based – filter order may become prohibitive

Can we make the spectrum estimate adaptive?

Supervised adaptive filters



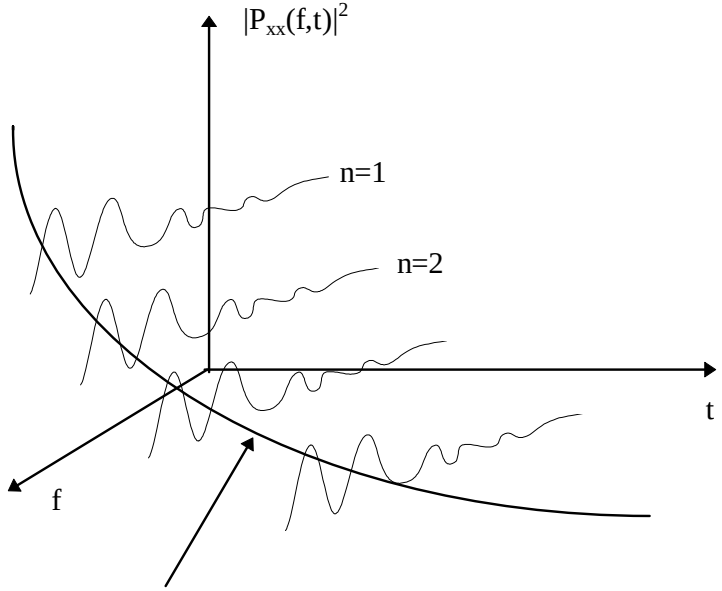
- The input signal is $x[n]$, and the coefficients of the second order linear predictor, $a_2(1), a_2(2)$, are controlled by an adaptive algorithm
- The adaptive algorithm adjusts these coefficients so as to minimise the prediction error power $E\{e^2[n]\}$

Clearly, this structure performs sequential AR spectral estimation

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

Adaptive SP vs. Spectral Estimation

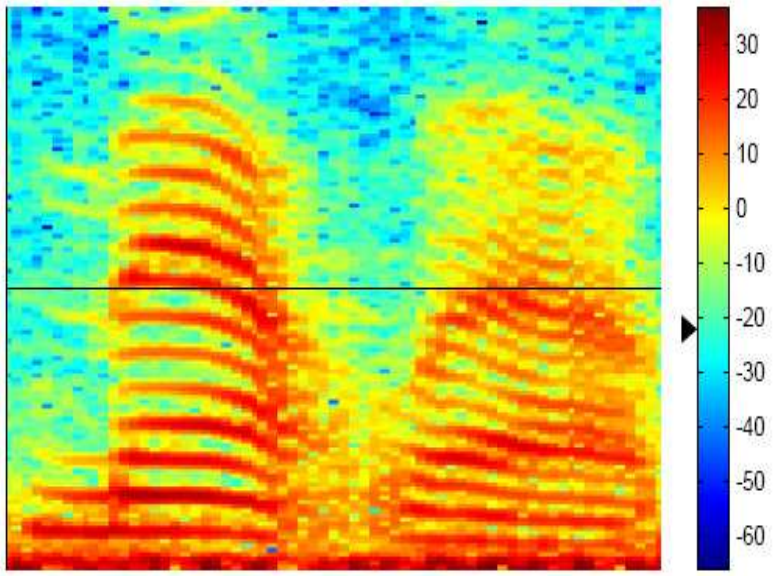
- An adjustment to the coefficient values can be made as each new sample arrives in $\mathbf{x}[n] = [x(n-1), x(n-2), \ldots, x(n-N)]^T$
- Therefore, it is possible to estimate the shape of the input power spectral density, at every iteration, based upon these estimated parameters $\mathbf{a}(n) = [a_1(n), \ldots, a_N(n)]^T$
- 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



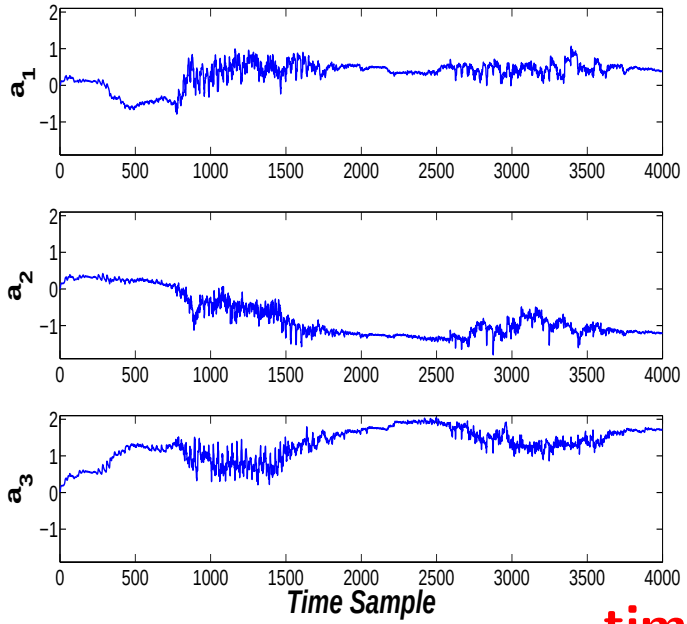
time

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

PSD from the coefficients of AFs

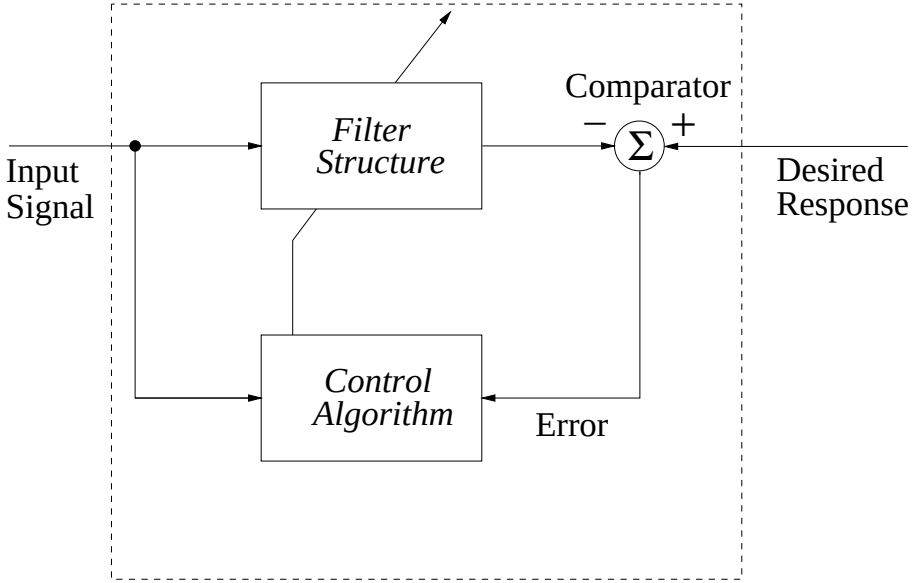
The shape of the PSD is estimated from:

$$\hat{P}_{xx}(f,n) = \frac{1}{|1 + a_1(n)e^{-\jmath 2\pi f} + \cdots + a_p(n)e^{-\jmath 2p\pi f}|^2}$$

Again:

- An adaptive filter: architecture + learning algorithm
- it has a certain structure, for example finite impulse response (FIR) in direct or lattice form or infinite impulse response (IIR)
- and a particular algorithm which adjusts the parameters of the filter so as to minimise some function of the output error.

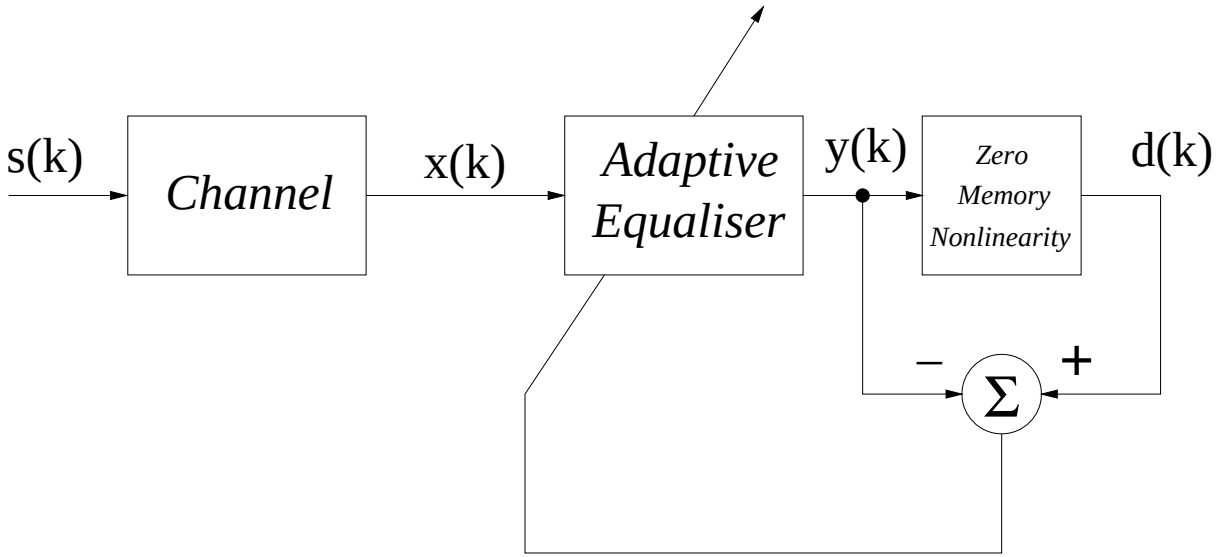
Generic adaptive filter (supervised, unsupervised)



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

Blind (unsupervised) adaptive filter

One possible structure:



- Unlike **supervised** filters (previous slide), no desired signal readily available (hence **unsupervised or blind**)
- We need to employ some "prior knowledge" (symbols ± 1 in comms)
- In general, slower convergence than supervised filters
- In this course we will focus on supervised adaptive filters

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

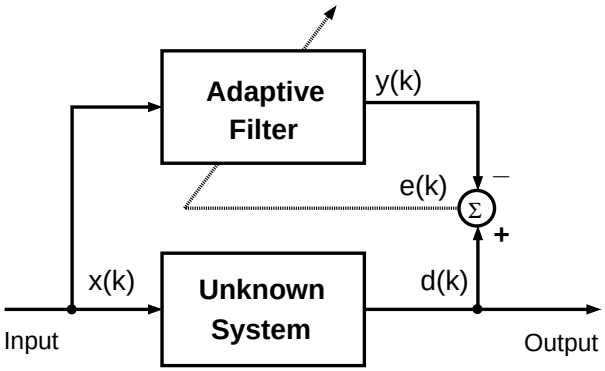
2. **System identification** (both the adaptive filter and the unknown system are fed with the same input signal $x(k)$), as in acoustic echo cancellation

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

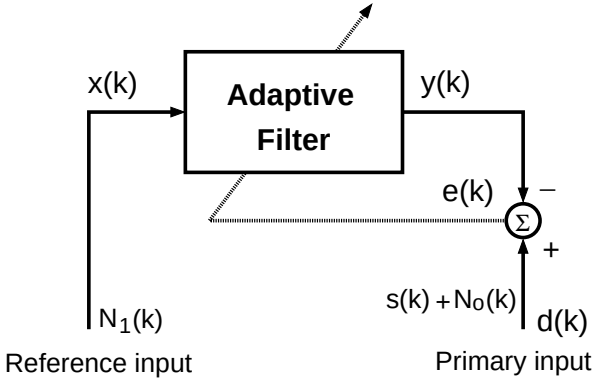
4. **Noise cancellation** (the only requirement is that the noise in the primary input and the reference noise are correlated), as in speech denoising

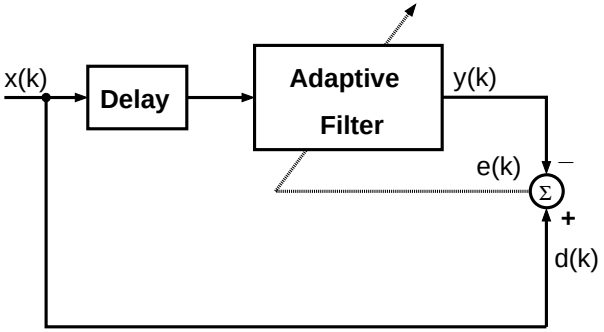
Applications of adaptive filters – Block diagrams

System Identification

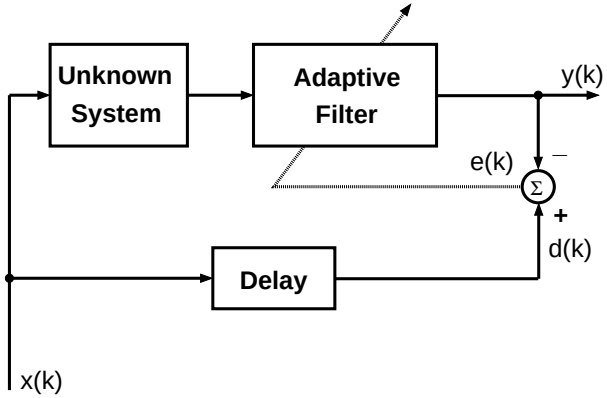


Noise Cancellation





Adaptive Prediction

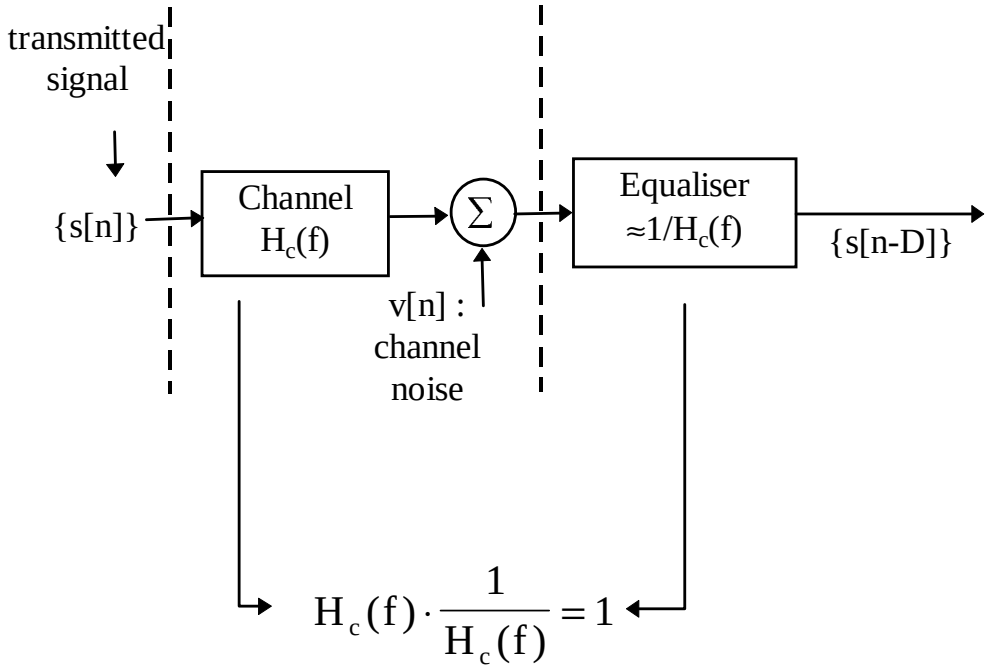


Inverse System Modelling

Example: Channel equalisation

- A transmitted signal $s[n]$ whose values belong to some finite alphabet, e.g. ± 1, is passed through a channel with frequency response $H(f)$
- This channel models the dispersion introduced by a physical or wireless link
- The channel output is also corrupted by additive measurement noise which might be due to temperature effects at the receiver
- The job of the equaliser or inverse filter is to model approximately the inverse of the channel
- The equaliser output will then be approximately $s[n - D]$, where D is a fixed delay

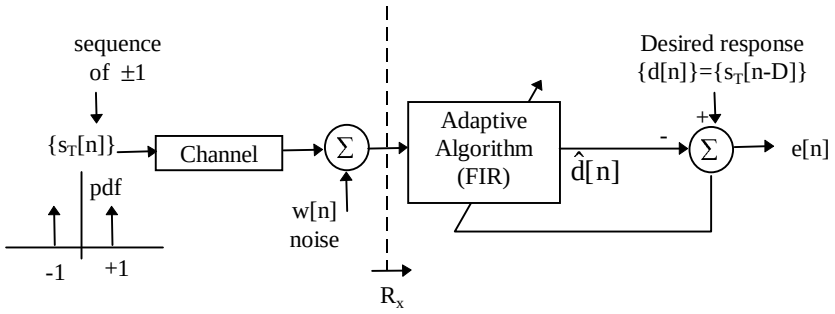
Block diagram – channel equalisation



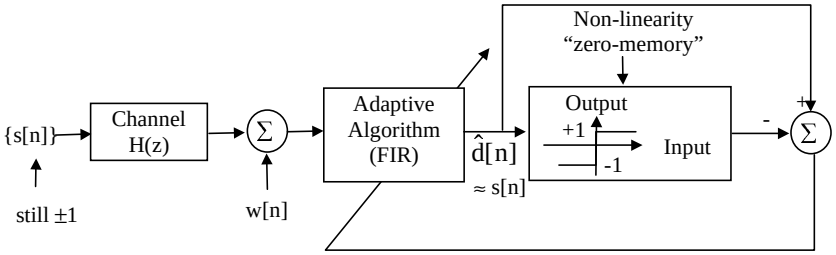
- In practice, an adaptive solution is required to design the equaliser because the channel is **time-varying**
- Such time-variation may be due to variations of the characteristics of some transmission path or the movement of the mobiles in a wireless environment

Training an adaptive equaliser

- Such an adaptive equaliser can operate in two modes, **training** or **blind** modes
- The adaptive equaliser is assumed to have an FIR structure and the desired response is assumed to be known at the receiver
- The disadvantage with this approach is that useful transmission is interrupted whilst the training information is transmitted



Training mode

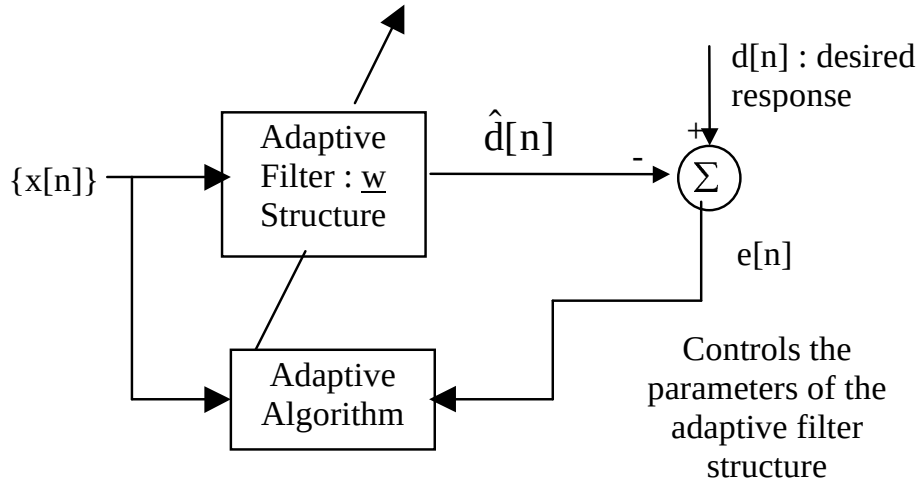


Blind mode

Operation of an adaptive equaliser

- In the blind mode of operation the adaptive filter operates without an explicit desired response and hence the term "blind"
- A desired response is generated from the prior knowledge that if equalisation is achieved, the equaliser output should either be ± 1
- Any difference between the closest one of these values and the output of the equaliser is used as an error signal to drive the adaptation algorithm
- This blind approach to equalisation has considerable advantage because it is not necessary to interrupt transmission and hence bandwidth efficiency is maximised
- The convergence properties of blind equalisation algorithms is an open issue, however, and remains as a subject of on-going research
- Researchers within the Communications and Signal Processing Research Group at Imperial College have contributed much to this area since 1994

Zoom in into adaptive filters



- The 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 in the applications below
- To measure the performance of an adaptive filter we can consider how functions of the error $J(e[n])$ behave as time increases, or whether the filter coefficient (weight) vector $\mathbf{w}(n)$ approaches some optimal setting

Adaptive algorithm

- All algorithms are based on minimising some function of the error:

$$f[e[n]] = |e|, \quad f[e[n]] = e^2, \quad f[e[n]] = e^4, \quad f[e[n]] = |e^3|$$

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

- The error squared form will be found to be most analytically tractable and appropriate for measurements corrupted by Gaussian noise
- When the measurement noise is sub-Gaussian higher power errors are preferred whilst for super-Gaussian measurement noise distributions, lower power errors are more useful
- 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
- The function we wish to minimise is the Mean Square Error (MSE):

$$J \equiv \frac{1}{2} E\{e^2[n]\} \quad \text{where} \quad e[n] = d[n] - \hat{d}[n] = d[n] - \mathbf{x}^T[n]\mathbf{w}[n]$$

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:

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

Definitions of the cross correlation vector and autocorrelation matrix

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

Error performance surface

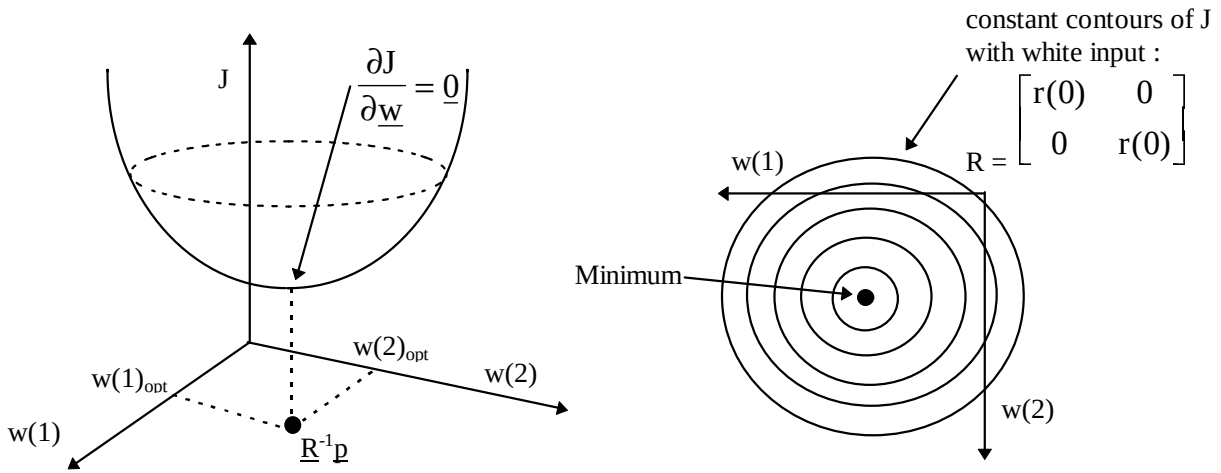
Thus, (let us make $\mathbf{w}$ constant for the time being)

$$J = \sigma_d^2 - 2\mathbf{w}^T\mathbf{p} + \mathbf{w}^T\mathbf{R}\mathbf{w}$$

$\Rightarrow$ quadratic function of $\mathbf{w}$ and provided $\mathbf{R}$ is full rank, J will have **one unique minimum.**

Consider a filter with two parameters such that $\mathbf{w} = [w(1), w(2)]^T$.

The input $\{x[n]\}$ is assumed to be zero mean, white noise, so that the contours of constant J, as in the figure on the right below, are circular.



Wiener–Hopf solution

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

$$\frac{\partial J}{\partial \mathbf{w}} = -\mathbf{p} + \mathbf{R} \cdot \mathbf{w} = \mathbf{0} \Rightarrow -\mathbf{p} + \mathbf{R} \cdot \mathbf{w}_{opt} = \mathbf{0}$$

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

- 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 when $\mathbf{w} = \mathbf{w}_{opt}$, i.e. $J_{min} = \sigma_d^2 - \mathbf{w}_{opt}^T\mathbf{p}$.

Role of eigen-analysis in Wiener solution

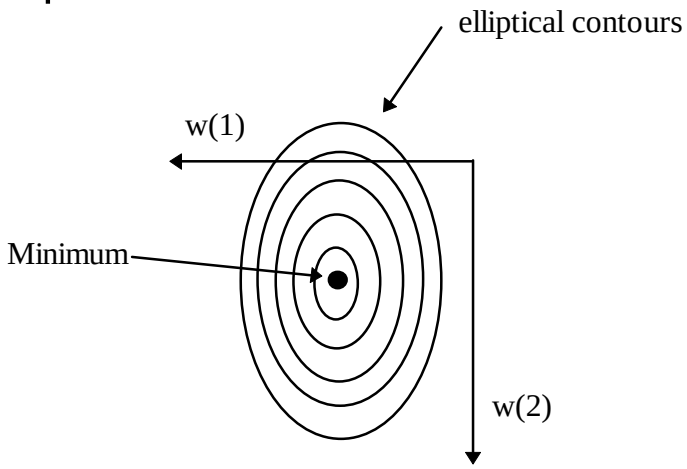
The shape of the error performance surface is related directly to the eigen–structure of the autocorrelation matrix $\mathbf{R}$.

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**.

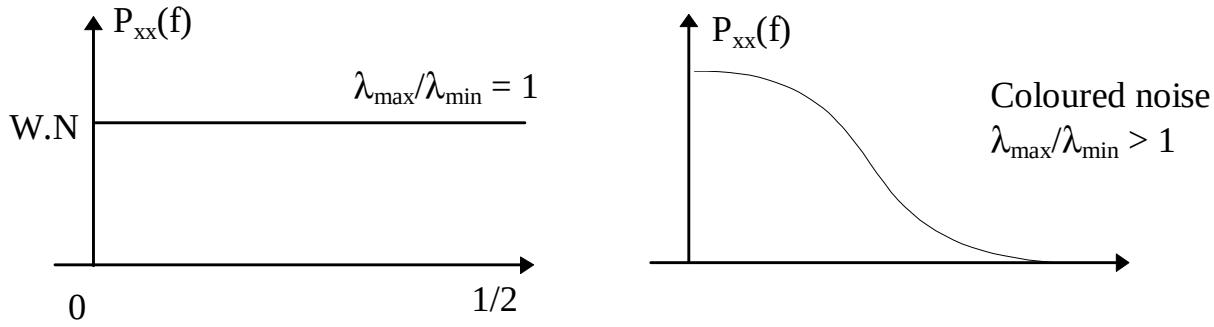
Therefore the **contours of J are circles when projected onto the $(w(1), w(2))$ plane.**

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.

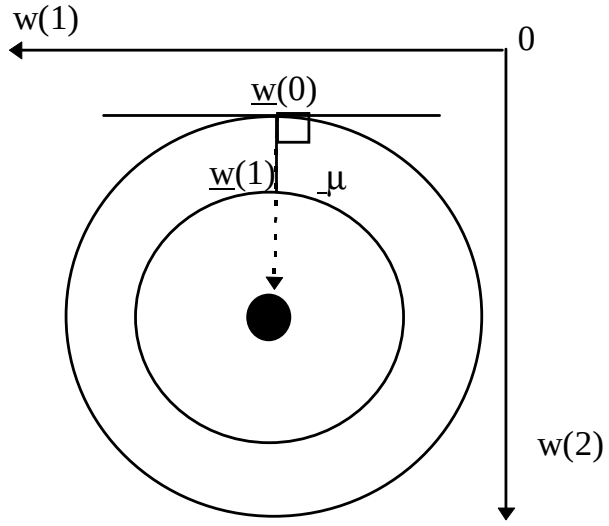
Steepest Descent (SD) methods

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 some function J is given by

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

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

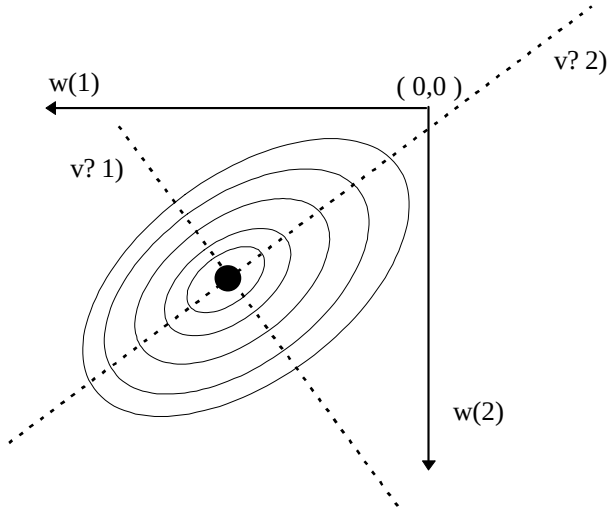


The convergence of the SD algorithm from the initial point $\mathbf{w}[0]$ toward the optimum.[1]

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

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 $\mathbf{R}_{xx}$

$$\begin{aligned} \mathbf{v}[n+1] &= [\mathbf{I} - \mu\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T]\mathbf{v}[n] \\ \mathbf{S}^T\mathbf{v}[n+1] &= \mathbf{S}^T(\mathbf{I} - \mu\mathbf{S}\mathbf{\Lambda}\mathbf{S}^T)\mathbf{v}[n] \end{aligned}$$

we define $\mathbf{v}'[n] = \mathbf{S}^T\mathbf{v}[n]$, *then*

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

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

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

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

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.

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

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]:

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu\mathbf{x}[n]\left(d[n] - \mathbf{x}^T[n]\mathbf{w}[n]\right) = \mathbf{w}[n] + \mu e[n]\mathbf{x}[n]$$

$$\mathbf{w}[0] = \mathbf{0} \qquad \text{where} \qquad d[n] - \mathbf{x}^T[n]\mathbf{w}[n] = e[n]$$

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

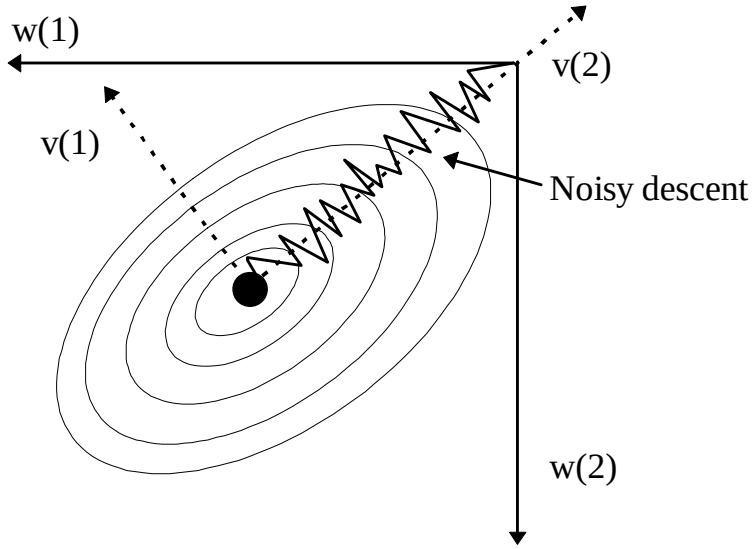
- together with its robust performance, is the reason why it finds extensive use in channel equalisation and echo cancellation in modems, and coding in speech (ADPCM) codecs.

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.

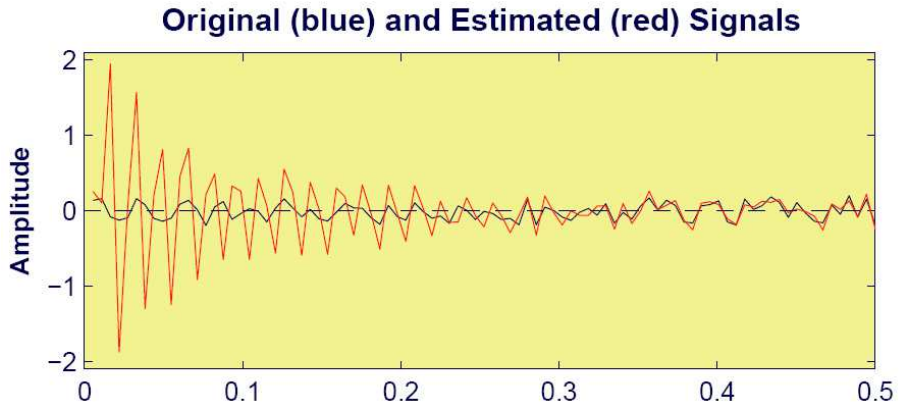
Only on the average will the LMS algorithm follow the direction of SD

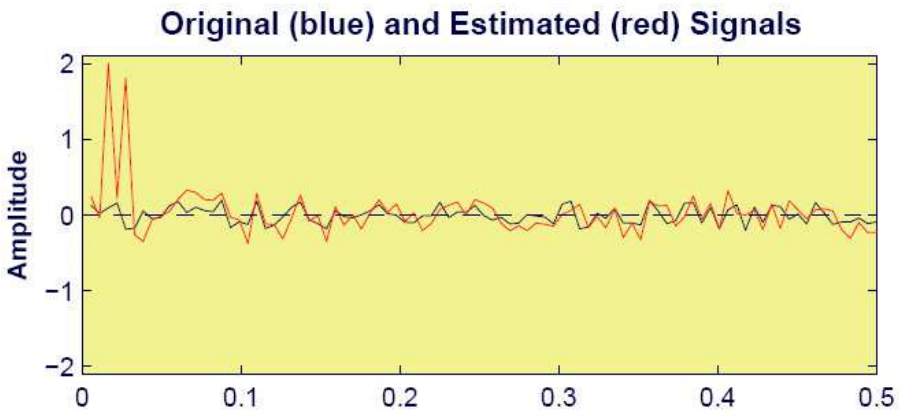
We wish to determine the value so that the average value of $\mathbf{w}[n]$ tends to the Wiener solution - this does not mean that the actual value of $\mathbf{w}[n]$ will equal the Wiener solution at anytime.



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

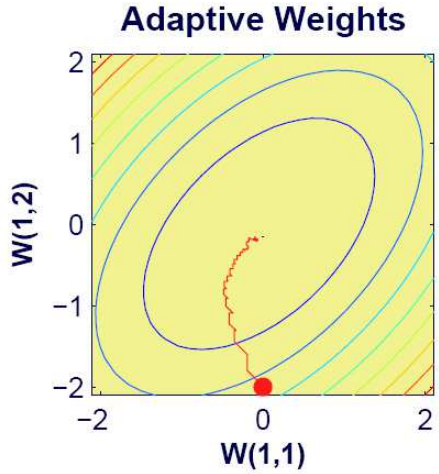
Signal and Prediction

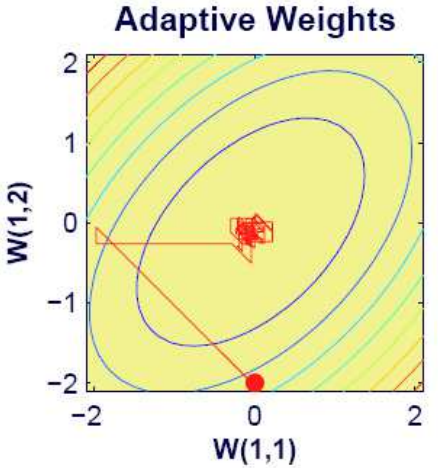




Top panel - learning rate 0.1

Error surface





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 $E\{\mathbf{w}[n+1]\} \approx E\{\mathbf{w}(n)\} \quad as \quad n \to +\infty$

- **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 $n \to \infty$

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

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

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 $\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$;
- $\mathbf{Q}$ is a matrix of eigenvectors and $\mathbf{\Lambda} = diag(\lambda_{max}, \ldots, \lambda_{min})$;
- $\mathbf{Q}$ can therefore rotate $\mathbf{R}$ into the diagonal matrix $\mathbf{\Lambda}$, that is, $\mathbf{Q}\mathbf{R}\mathbf{Q}^T \rightsquigarrow \mathbf{\Lambda}$;

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

$$\mathbf{v}'(n+1) = \big(\mathbf{I} - \mu\mathbf{\Lambda}\big)\mathbf{v}'(n)$$

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

Convergence of LMS in the mean – continued

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

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

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

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

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

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

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

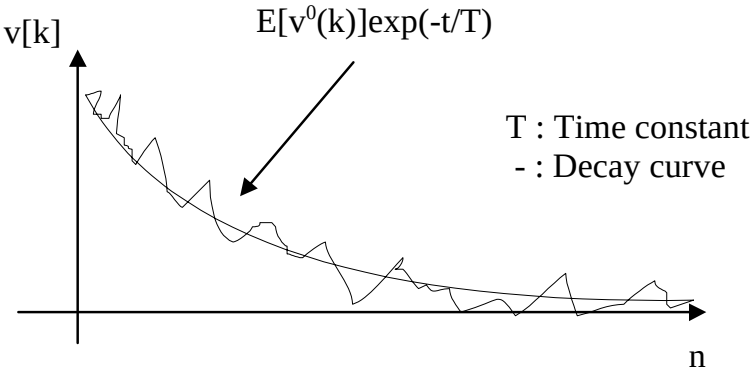
A dimensionless quantity used to quantify the accuracy of the convergence of the LMS algorithm is the misadjustment

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

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

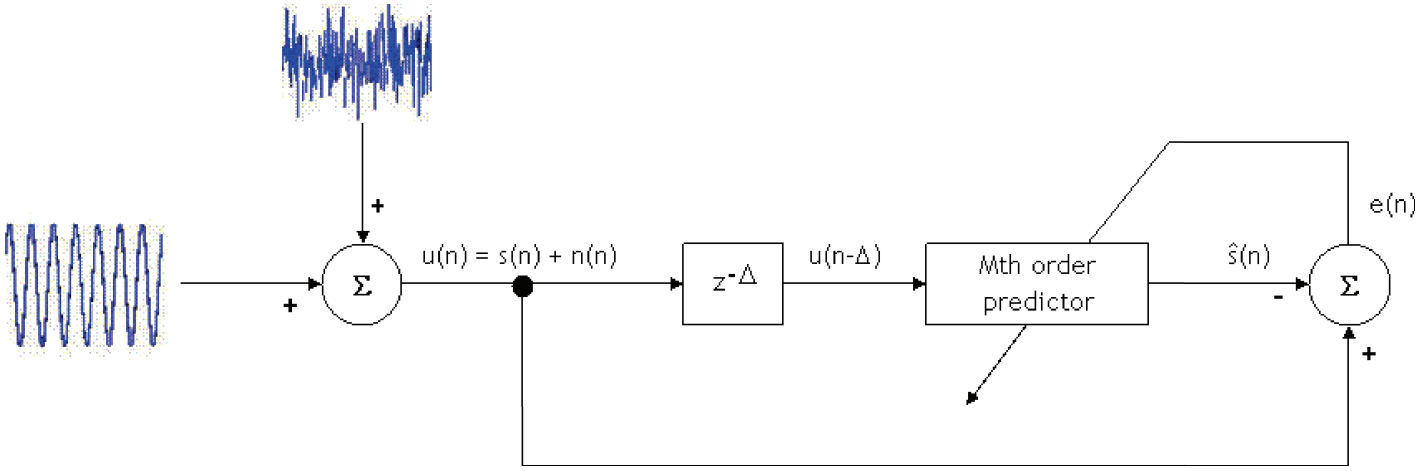
$$\tau_k \equiv -\frac{1}{ln(|1 - 2\mu\lambda_k|)}$$

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

Adaptive Line Enhancement (no reference)

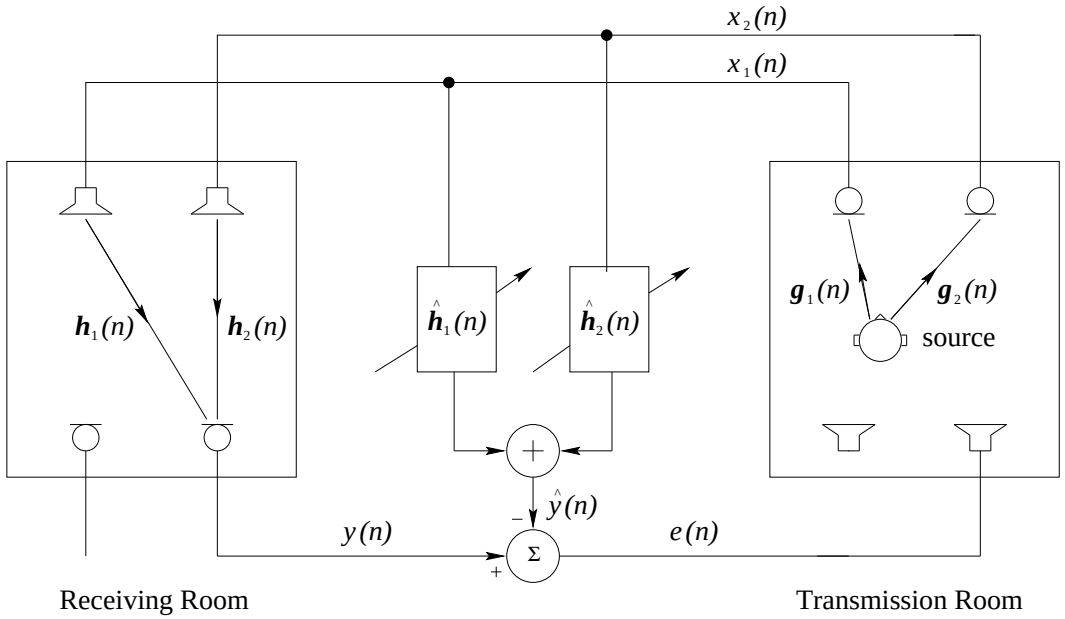
'lms_fixed_demo' (more detail soon)



Enhancement of a 100Hz signal in band limited WN, with a $N = 30$ FIR LMS filter.

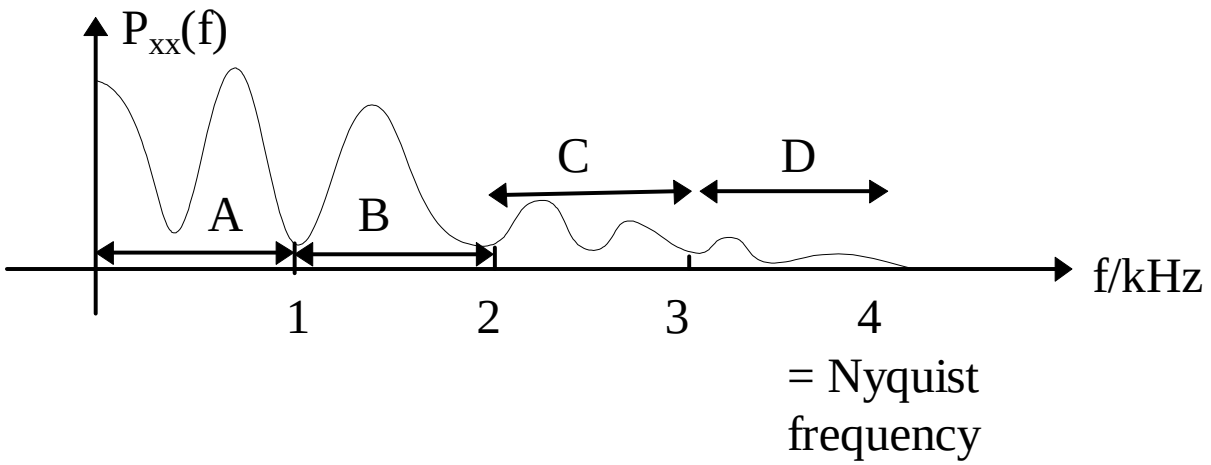
- Adaptive line enhancement (ALE) refers to the case where a noisy signal, $u(n) = `sin(n)' + `w(n)'$
- ALE consists of a de-correlation stage, symbolised by $z^{-\Delta}$ and an adaptive predictor
- The de-correlation stage attempts to remove any correlation that may exist between the samples of noise, by shifting them Δ samples apart
- A phase shift introduced (input Δ steps behind)

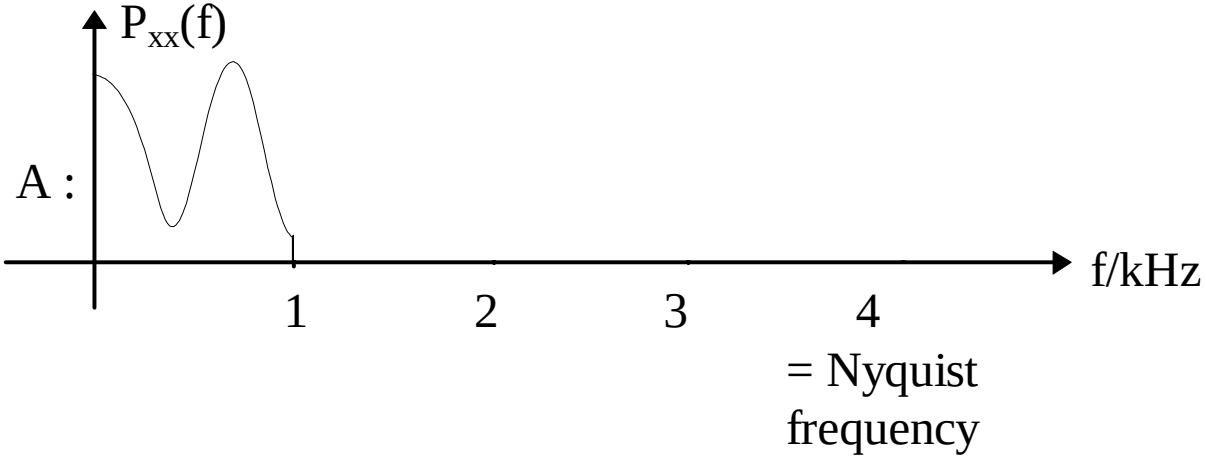
Acoustic Echo Cancellation (AEC) problem



- 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

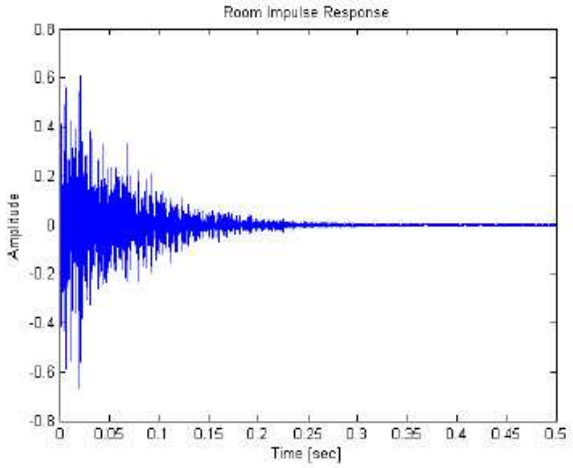
In terms of the spectrum



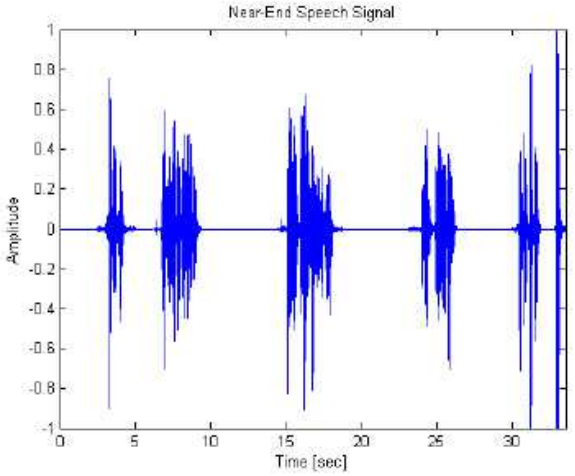


Acoustic echo cancellation problem: Signals

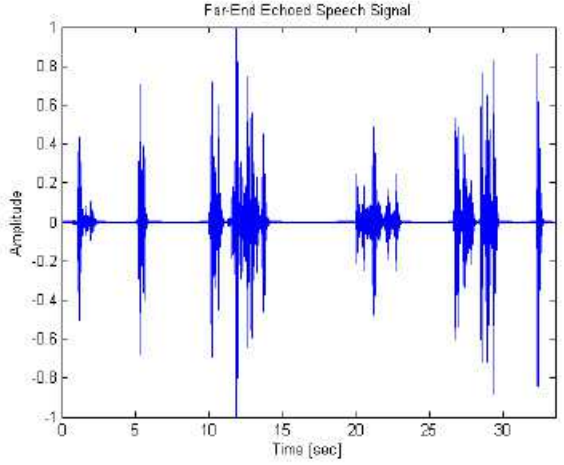
Room Impulse Response



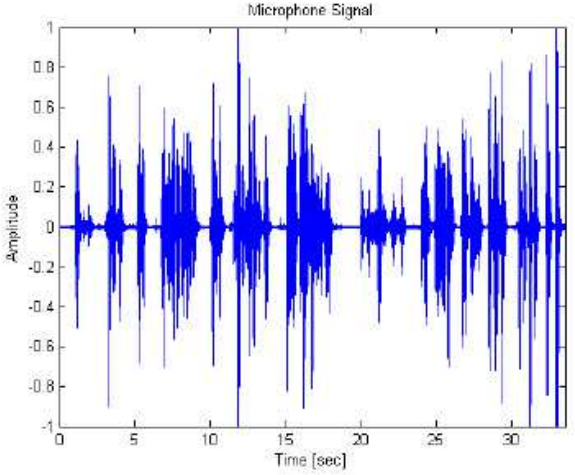
Near–end speech



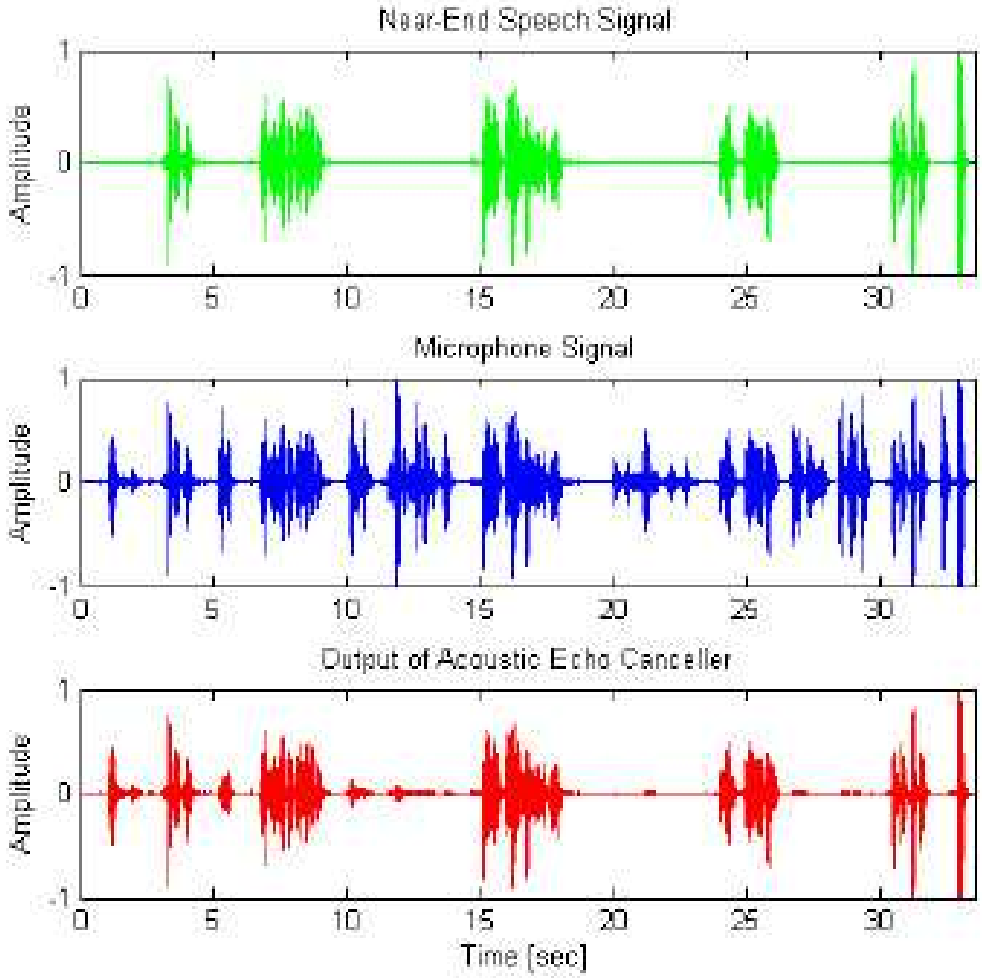
Far–end echoed speech



Microphone signal

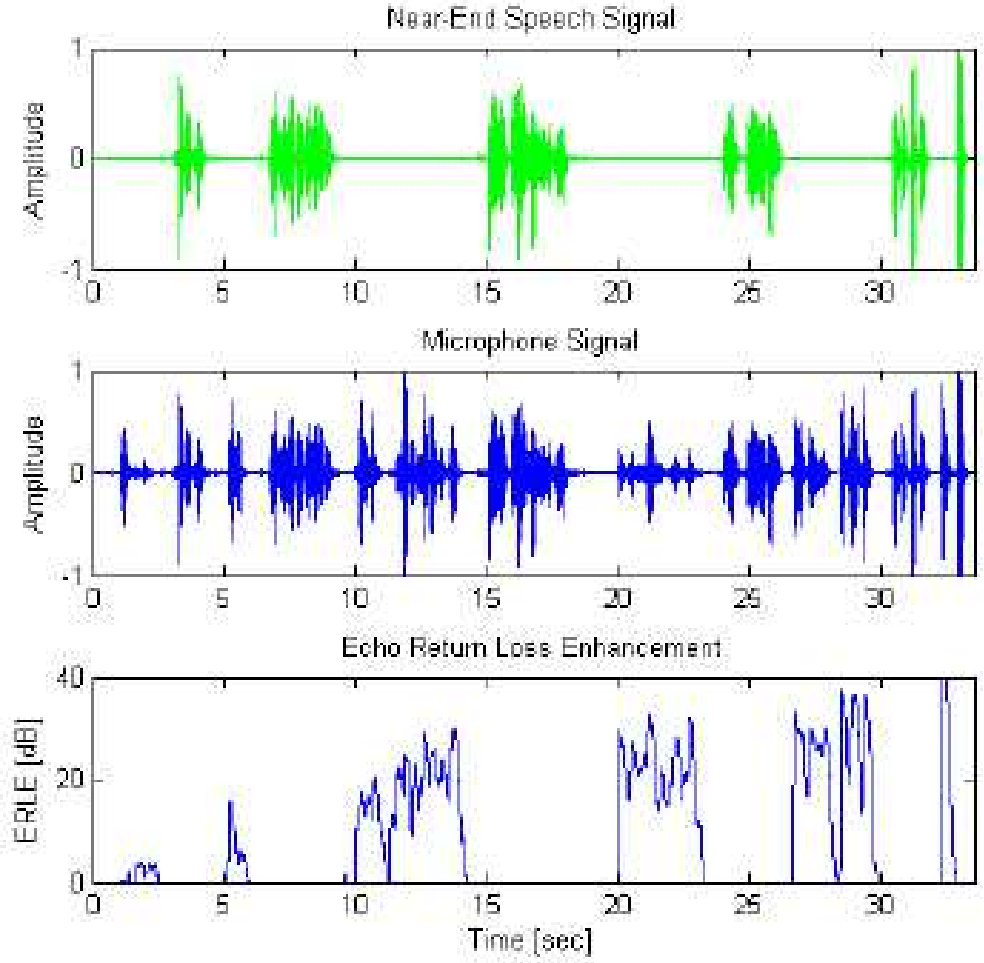


AEC – Cancellation results



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