Spectrum Estimation & Adaptive SP
Applications and Modifications of LMS

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

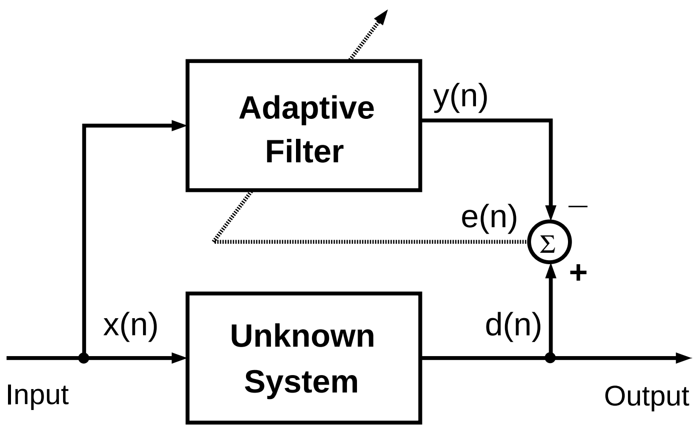
Motivation

- Applications of adaptive filters
- Faster initial convergence and enhanced stability (NLMS)
- Regularisation of Error Surface (NLMS, DR)
- A posteriori mode of learning $\looparrowright$ data reusing
- Boorrowing the concepts from physics $\looparrowright$ simulated annealing
- Reduced computational complexity $\looparrowright$ sign algorithms
- Regularisation and constrained optimisation $\looparrowright$ leaky algorithms
- Sub–band/frequency–domain adaptive filtering
- Stability consideration

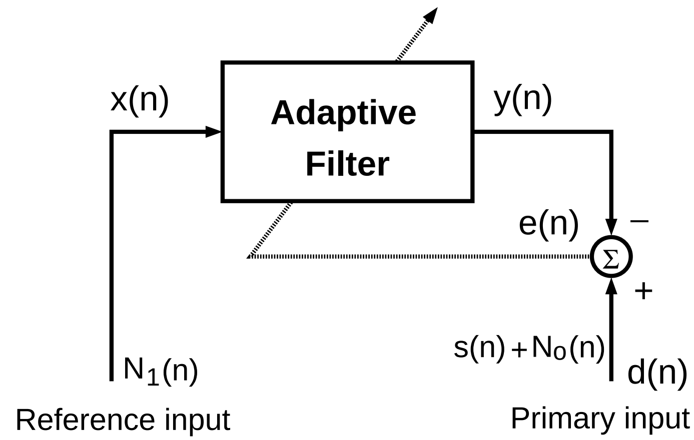
Recall: Adaptive filtering configurations

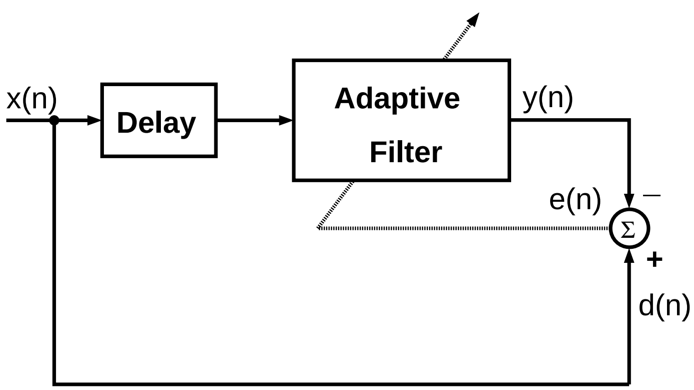
the same learning algorithm, e.g. the LMS, operates for any configuration

System identification

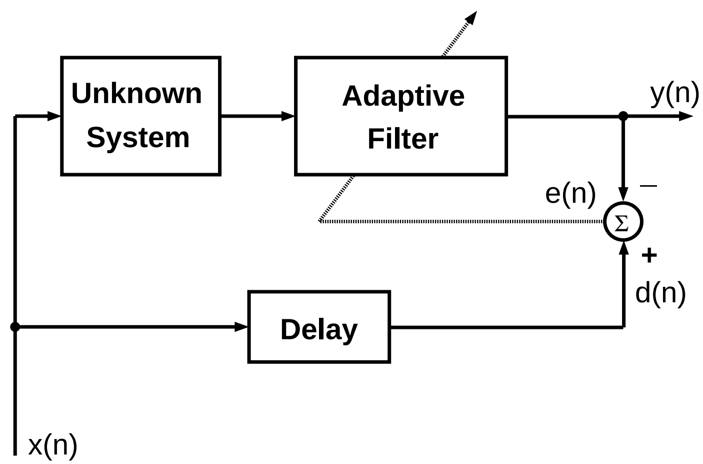


Noise cancellation





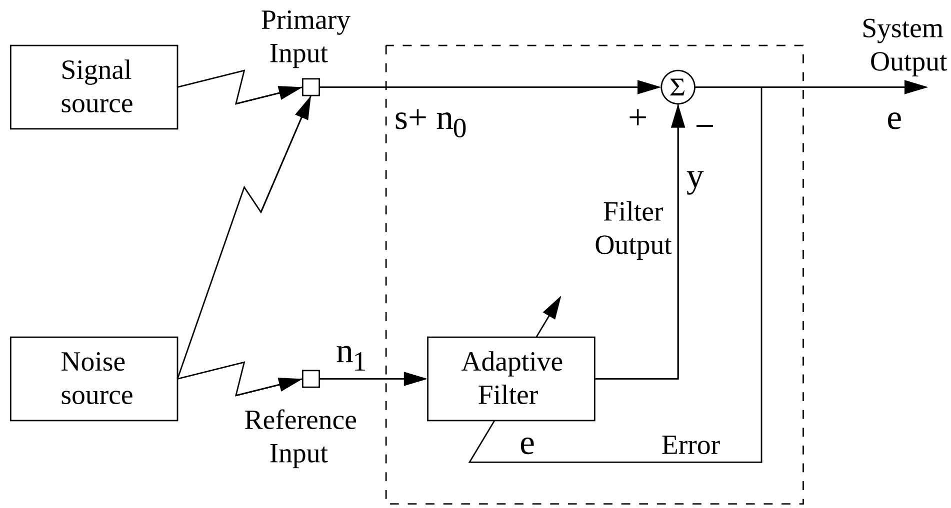
Adaptive prediction

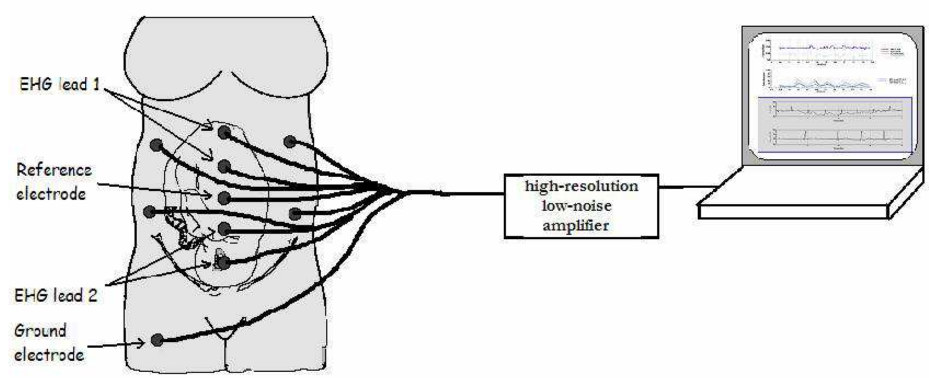


Inverse system modelling

Foetal ECG: Data Acquisition

ANC with Reference

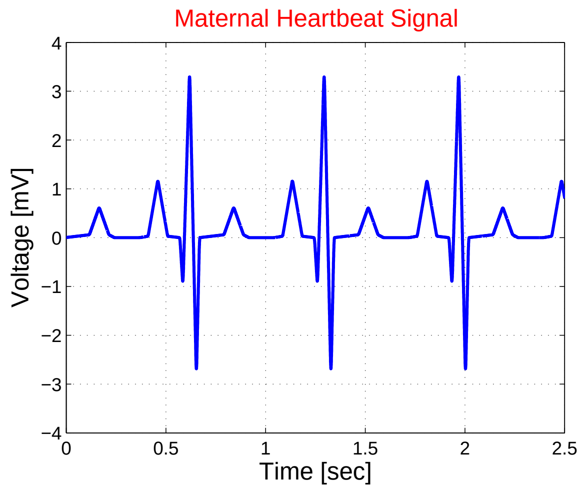




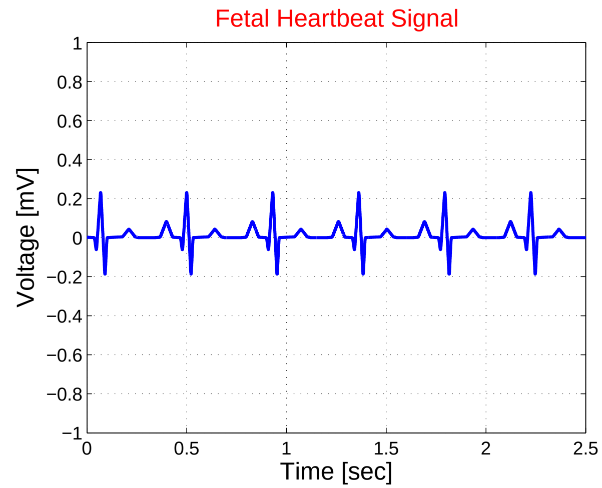
ECG recording (Reference electrode $\neq$ Reference input)

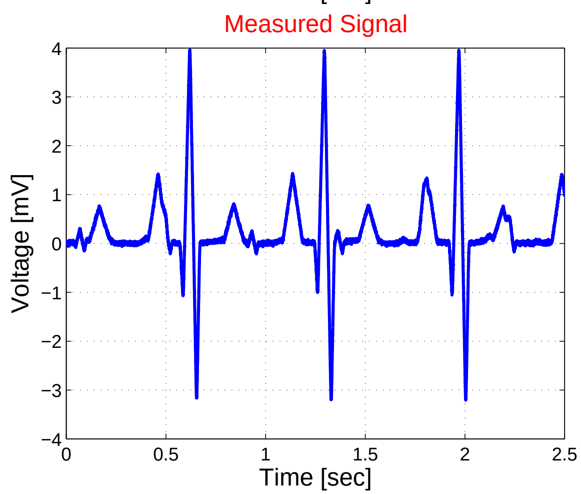
Foetal ECG Recovery

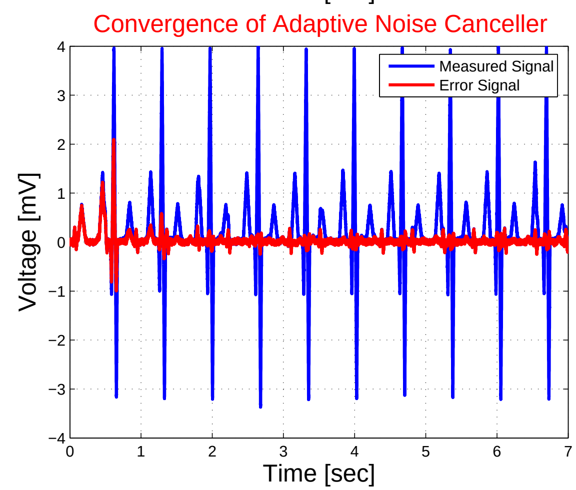
Maternal ECG Signal



Fetal Heartbeat







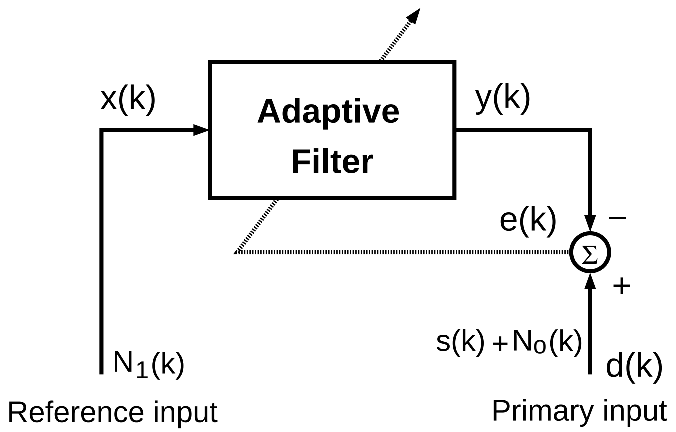
Measured Fetal ECG

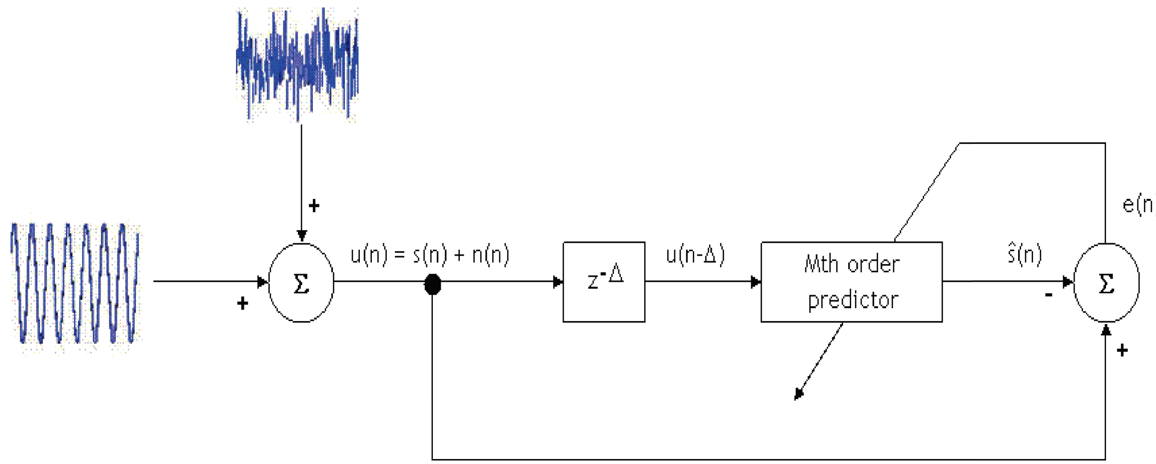
Maternal and Foetal ECG

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)





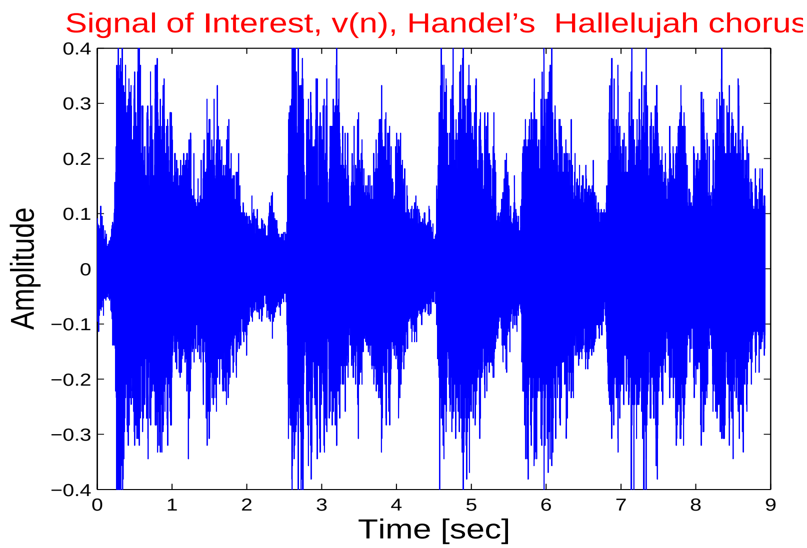
- Adaptive line enhancement (ALE) refers to the case where we want to clean a noisy signal,e.g. a noisy sinewave $u(n) = `sin(n)' + `wn(n)'$
- ALE is effectively an adaptive predictor equipped with a de-correlation stage, symbolised by $z^{-\Delta}$. The autocorrelation of noise is narrow, so

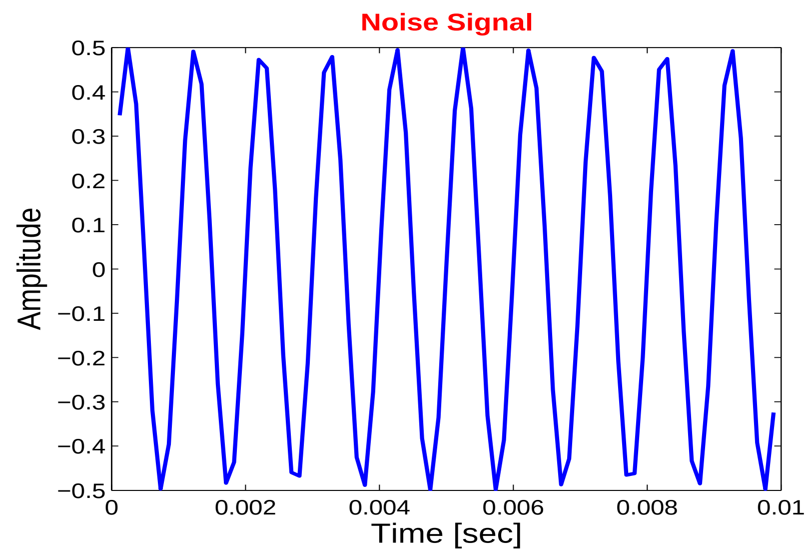
$$E\{u(n)u(n-\Delta)\} \approx E\{s(n)s(n-\Delta)\}$$

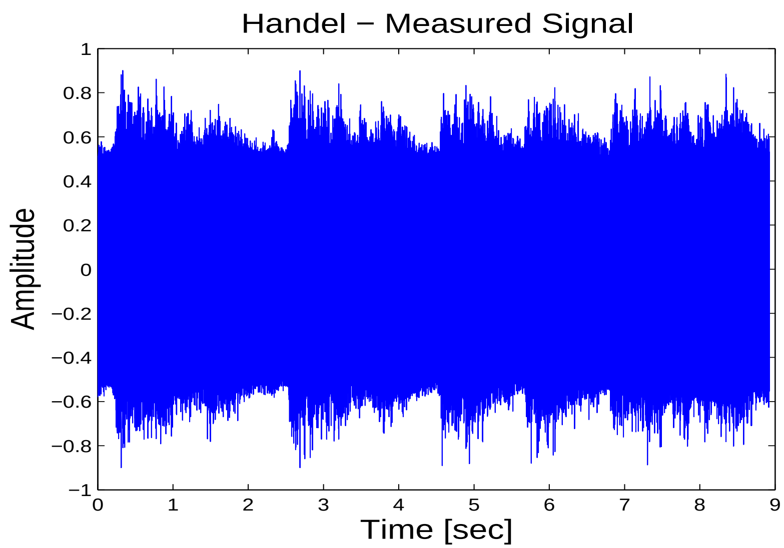
- By shifting $u(n)$ by Δ samples apart we aim to remove any correlation between the noise contribution in the samples $u(n)$ and $u(n-\Delta)$
- A small delay (phase shift) of Δ samples is introduced at the output

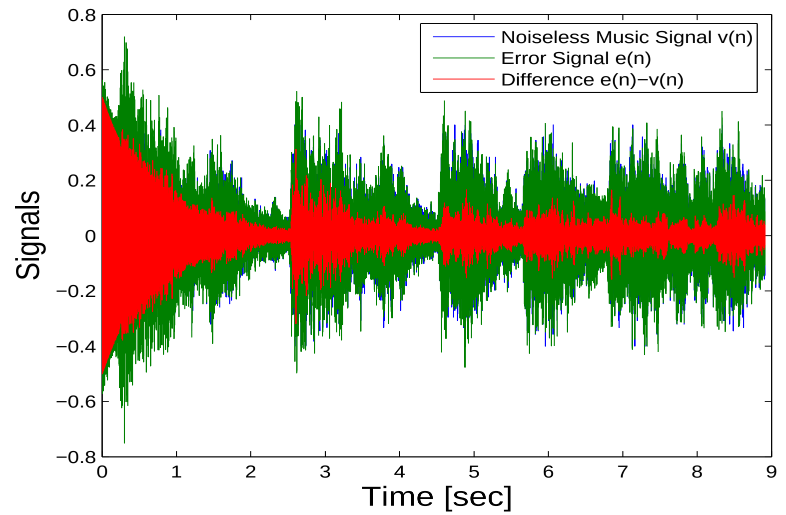
ALE - interference removal in music perform.

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









Quantitative performance assessment $\looparrowright$ 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 $e(n) = d(n) - \mathbf{x}^T(n)\mathbf{w}(n), \quad \mathbf{p} = E\{d(n)\mathbf{x}(n)\}$):

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

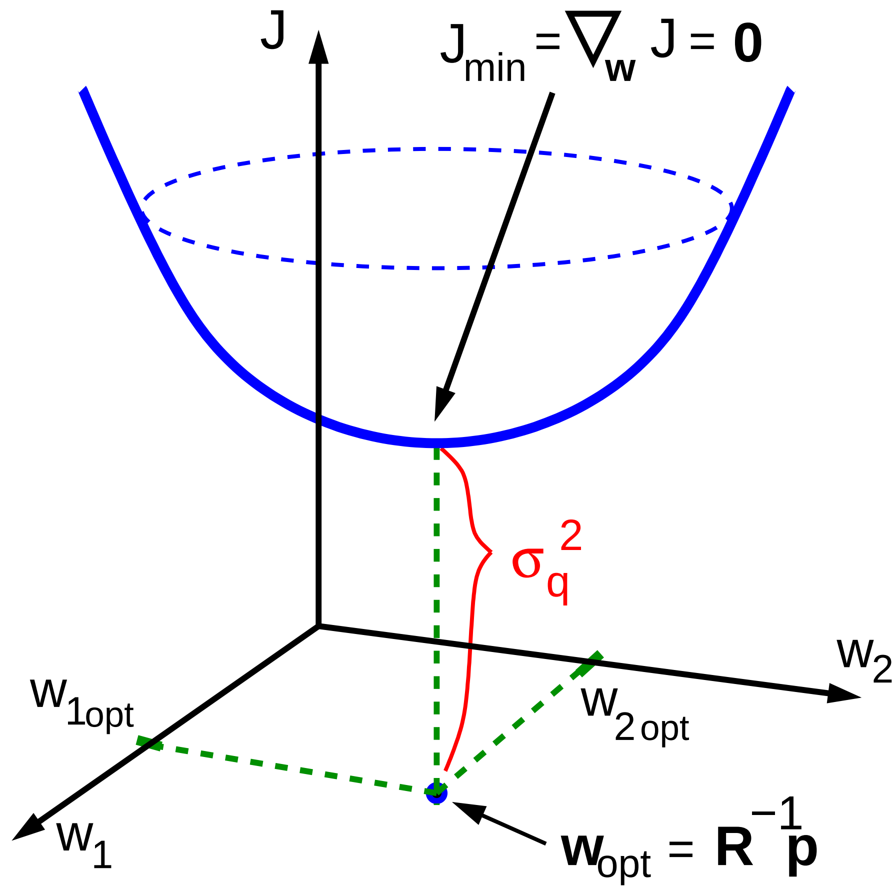
So, what is the value of J_{min}?
Assume without loss in generality that the teaching signal $d(n)$ is the output of a system with coefficients $\mathbf{w}_{opt}$

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

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



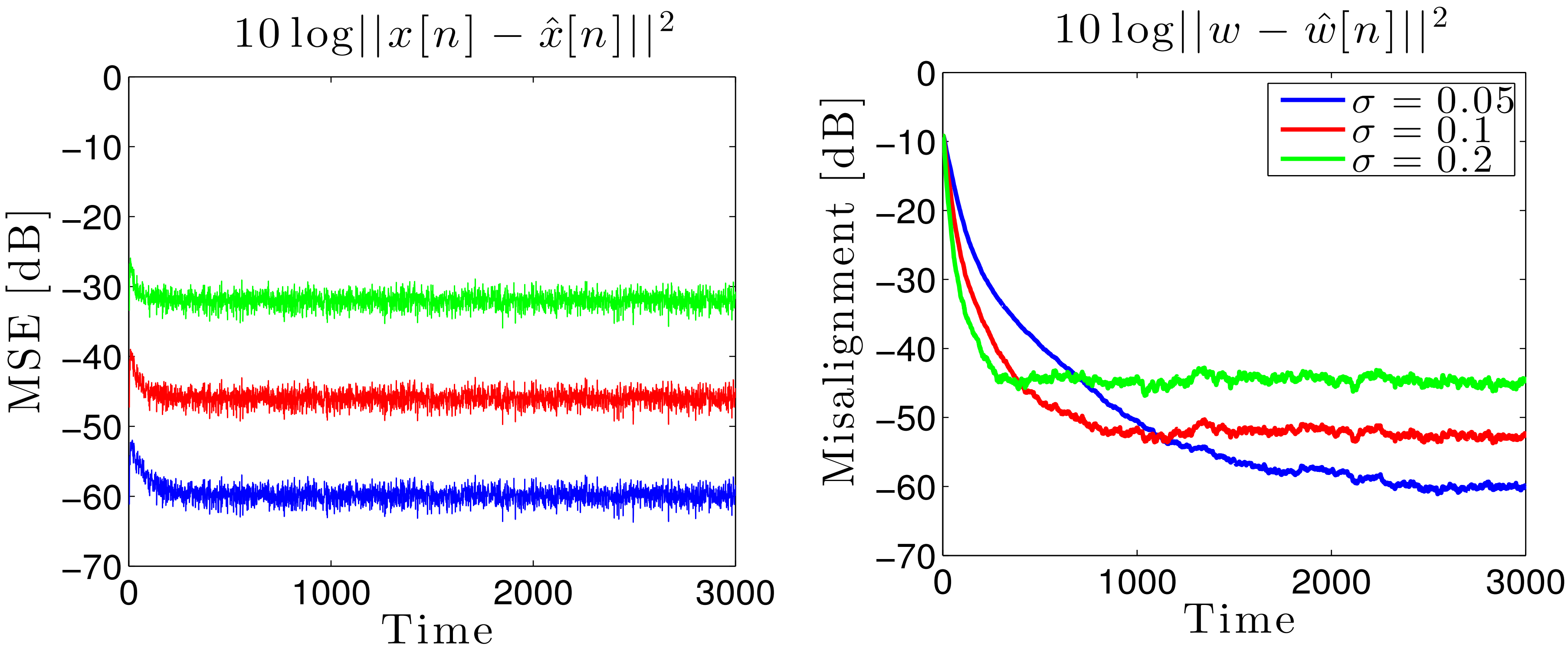
Learning curves: behaviour of MSE $\hookrightarrow$ plot of $10log|e(n)|^2$

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)

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

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, $\mathcal{M} = J_{ex}(\infty)/J_{min}$

Geometric insight into the LMS

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

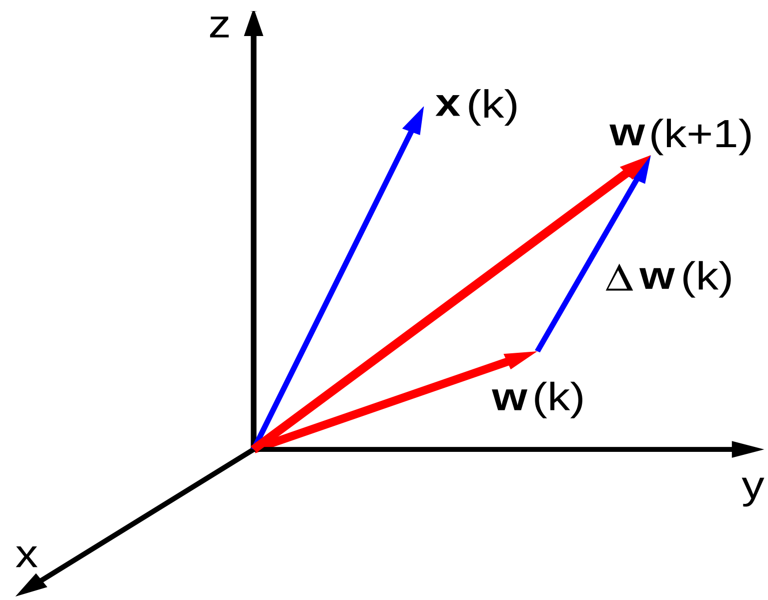
Recap: Let us derive LMS directly from the instantaneous cost function

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

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.

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 $sign(\mathbf{x}(n))$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n) sign(\mathbf{x}(n))$$

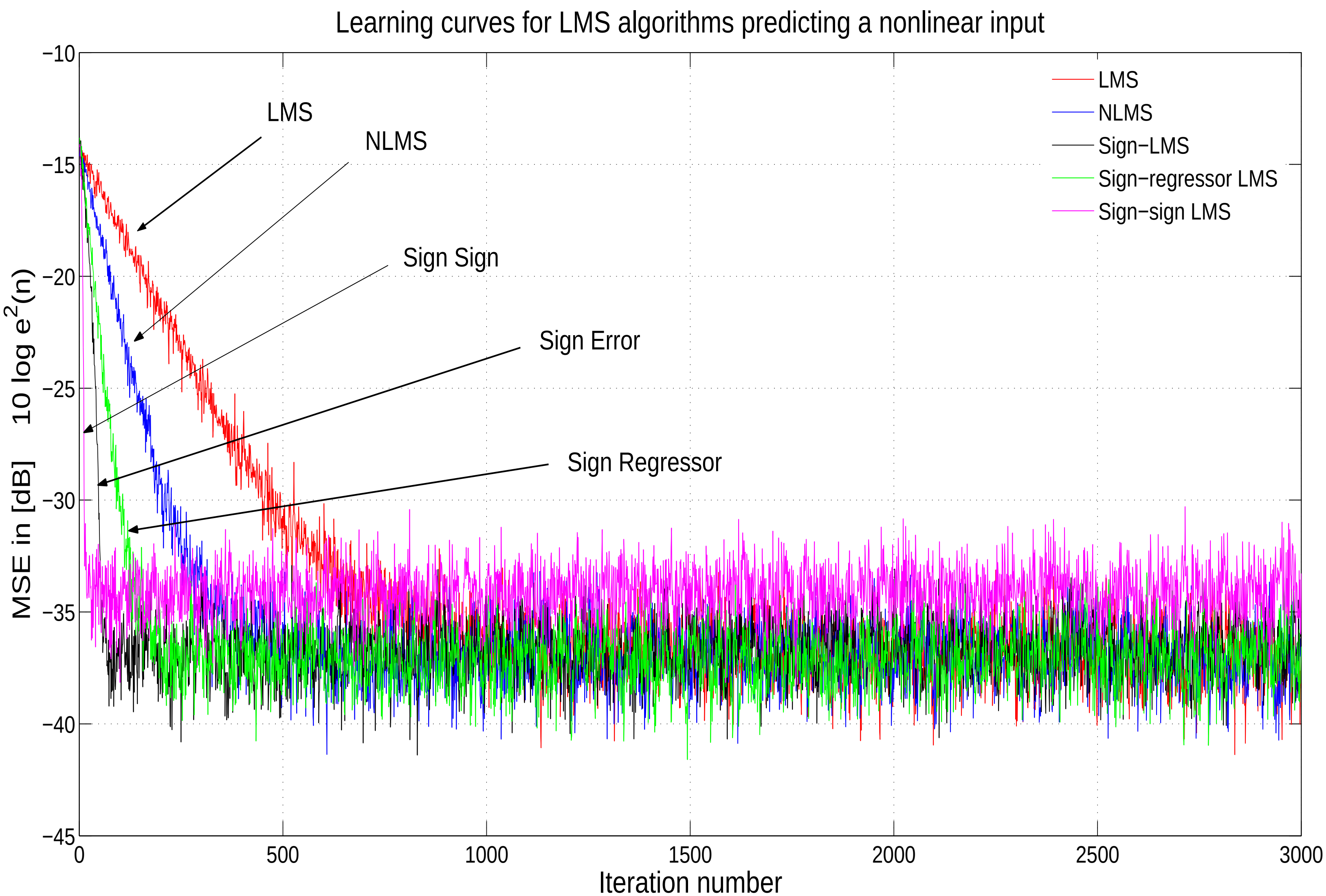
  Performs much better than the sign algorithm.

- **The Sign-Sign Algorithm**

  Combines the above two algorithms

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu sign(e(n)) sign(\mathbf{x}(n))$$

Performance of sign algorithms



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

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

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 $\looparrowright$ also run 'nnd10nc in Matlab'

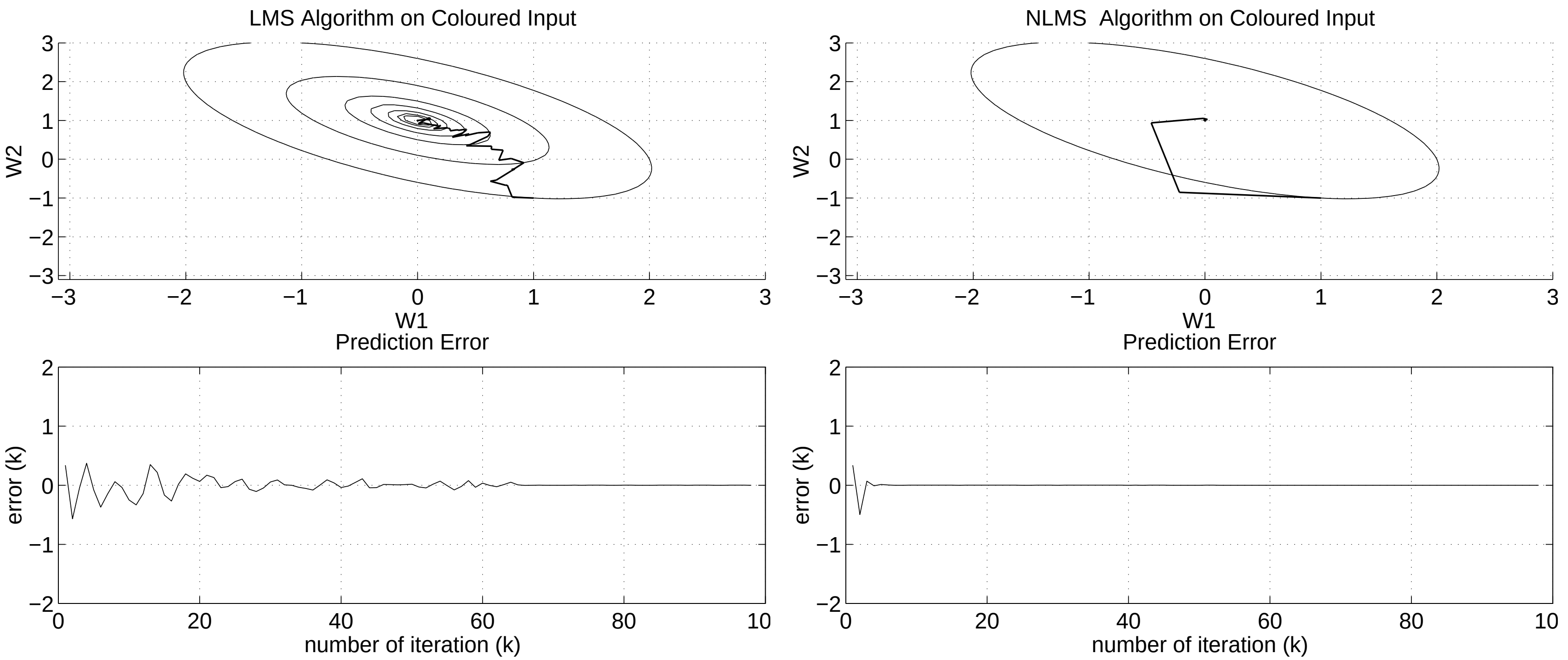
NLMS is independent of signal power $\rightsquigarrow$ suitable for real-world changing environ.

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

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

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

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



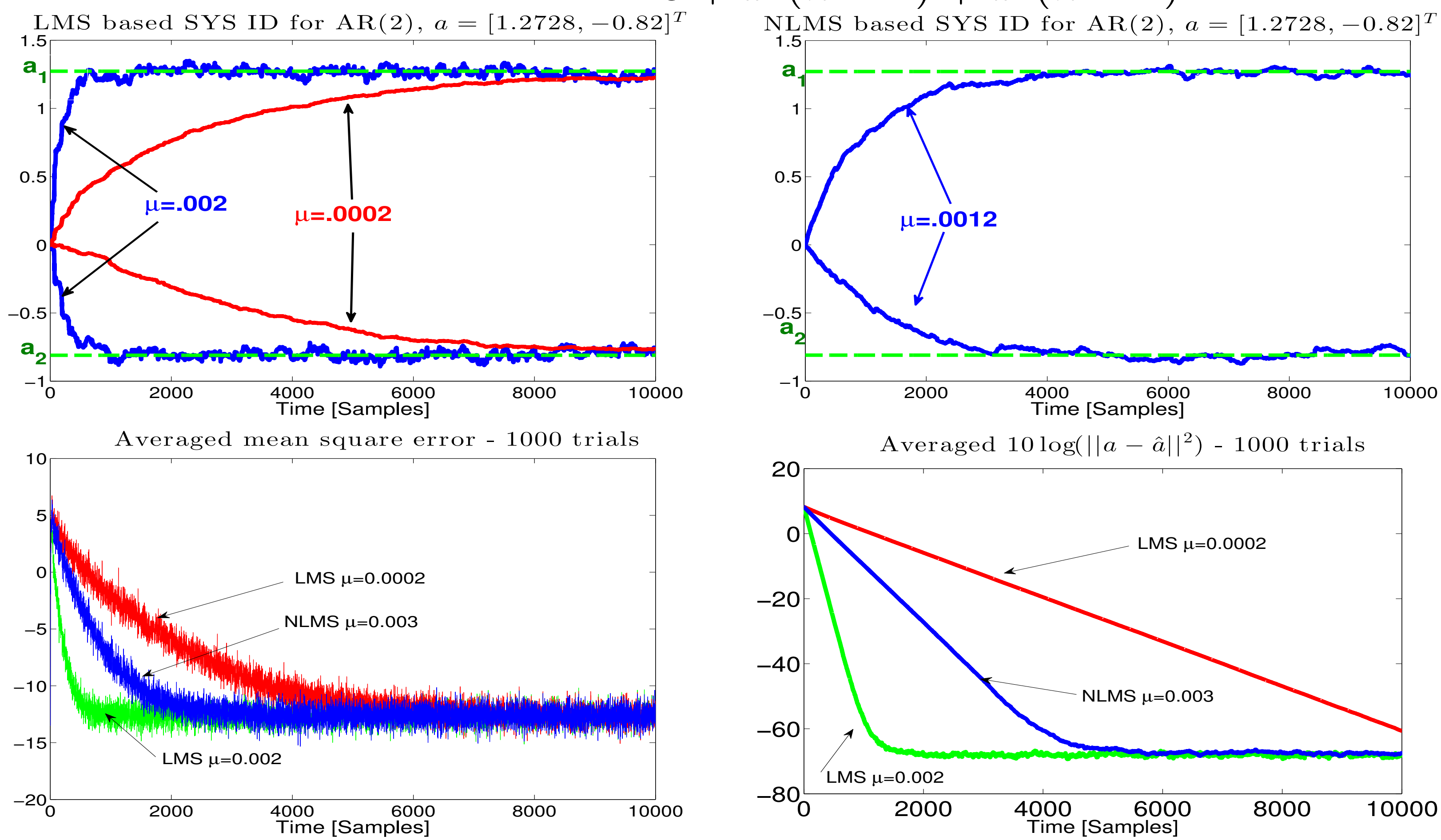
Example 1: Learning curves and performance measures

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 $(\tau = const)$

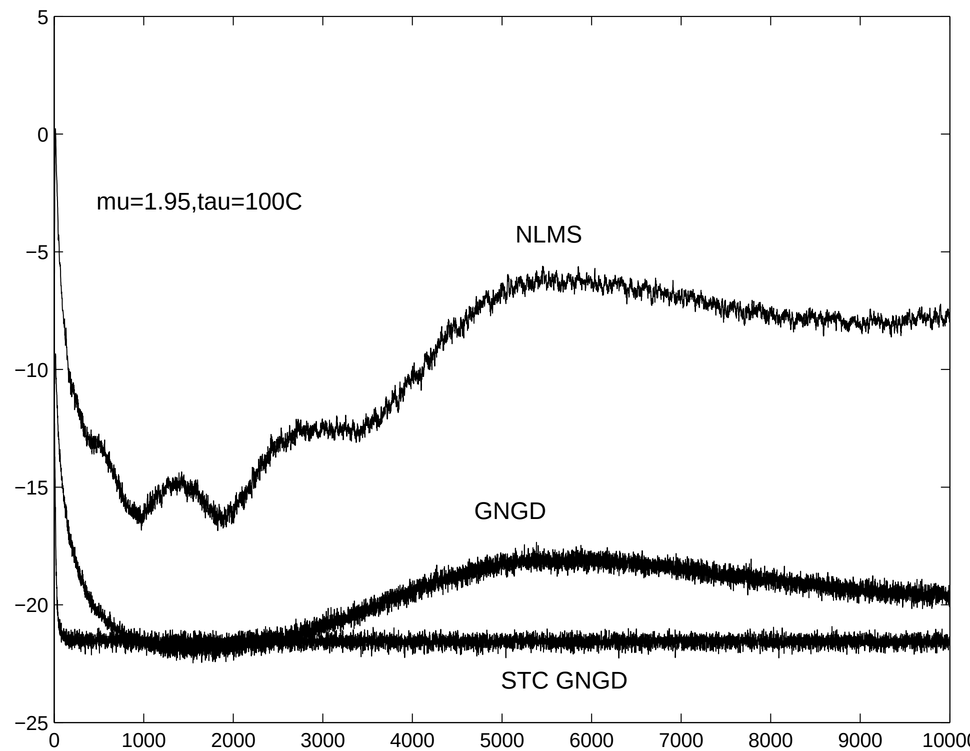
$$\eta(k) = \frac{\mu}{1 + k/\tau}$$

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

- 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

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

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

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

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

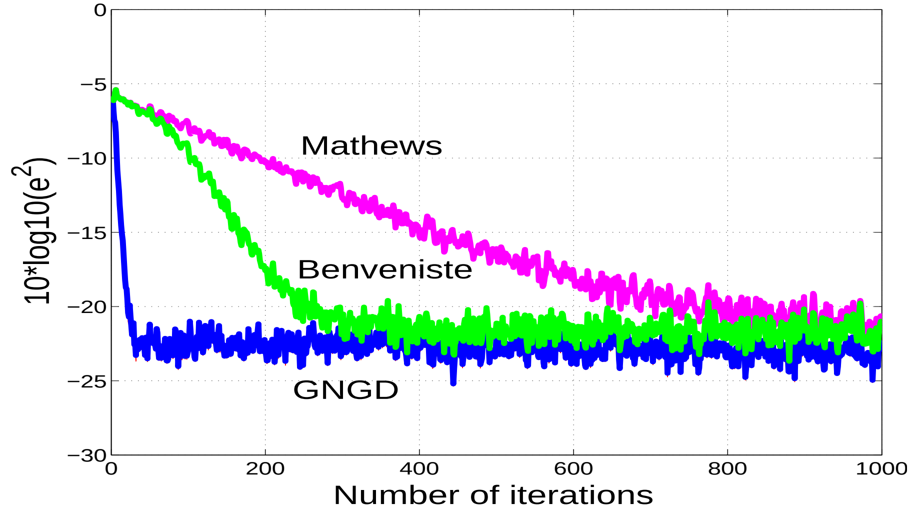
○ **The GNGD** $\looparrowright$ ***"nonlinear"*** update of $\mu(n)$ (gradient adaptive regularisation factor $\varepsilon(n)$ in NLMS), $\mu(n) \sim \nabla_{\varepsilon} J(n)$

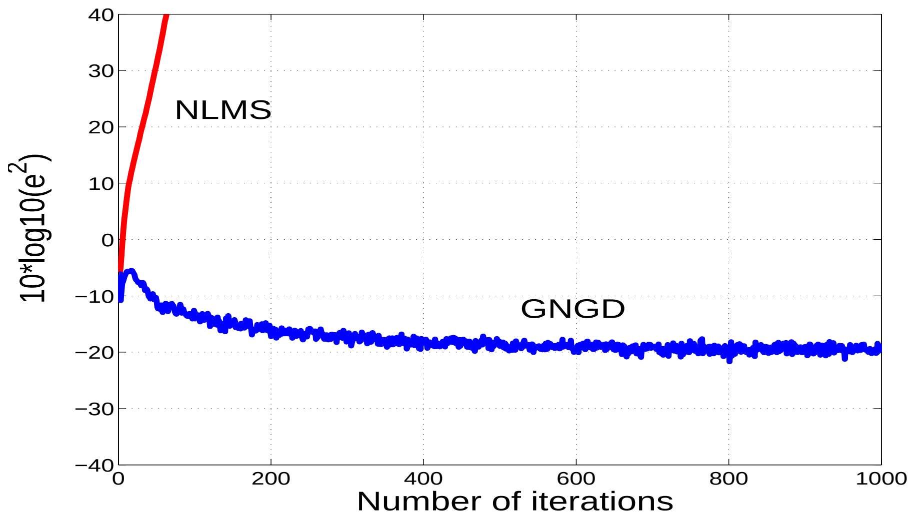
○ **GASS** algorithms $\looparrowright$ ***"linear"*** updates of $\mu(n)$, $\mu(n) \sim \nabla_{\mu} J(n)$

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

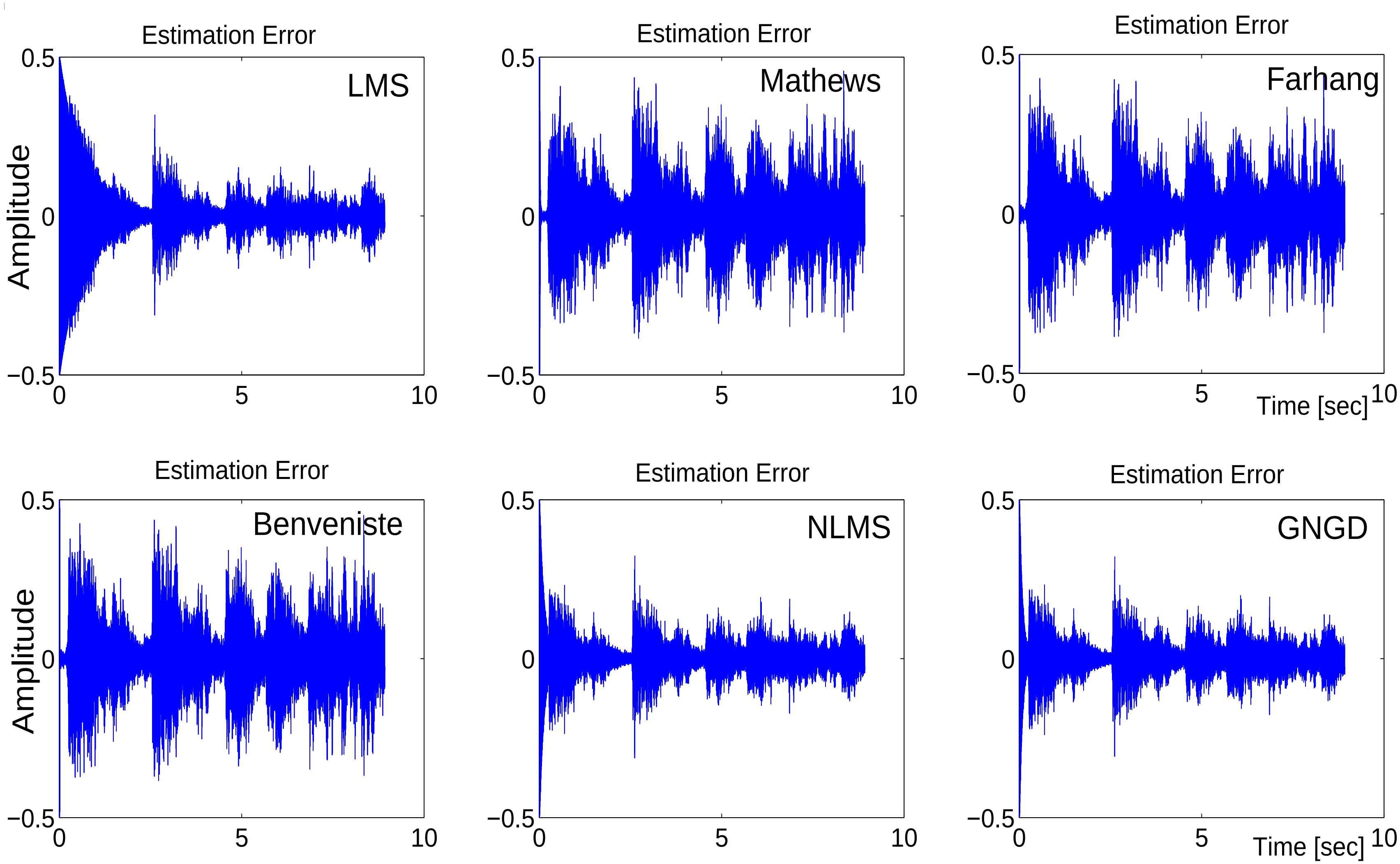
Bottom: convergence curves for $\mu = 2.1$





ALE for music, variable stepsize algs. All_in_One_ALE_Sin_Noise

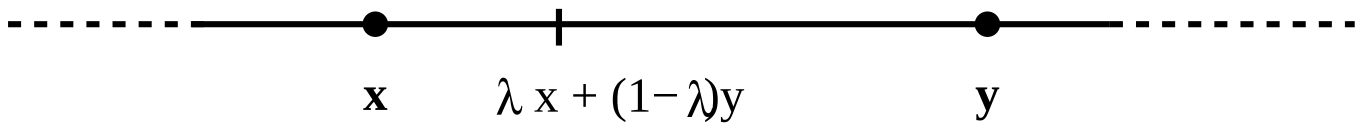
ALE parameters: $\Delta = 100$, filter length $N = 32$ (both can be varied)



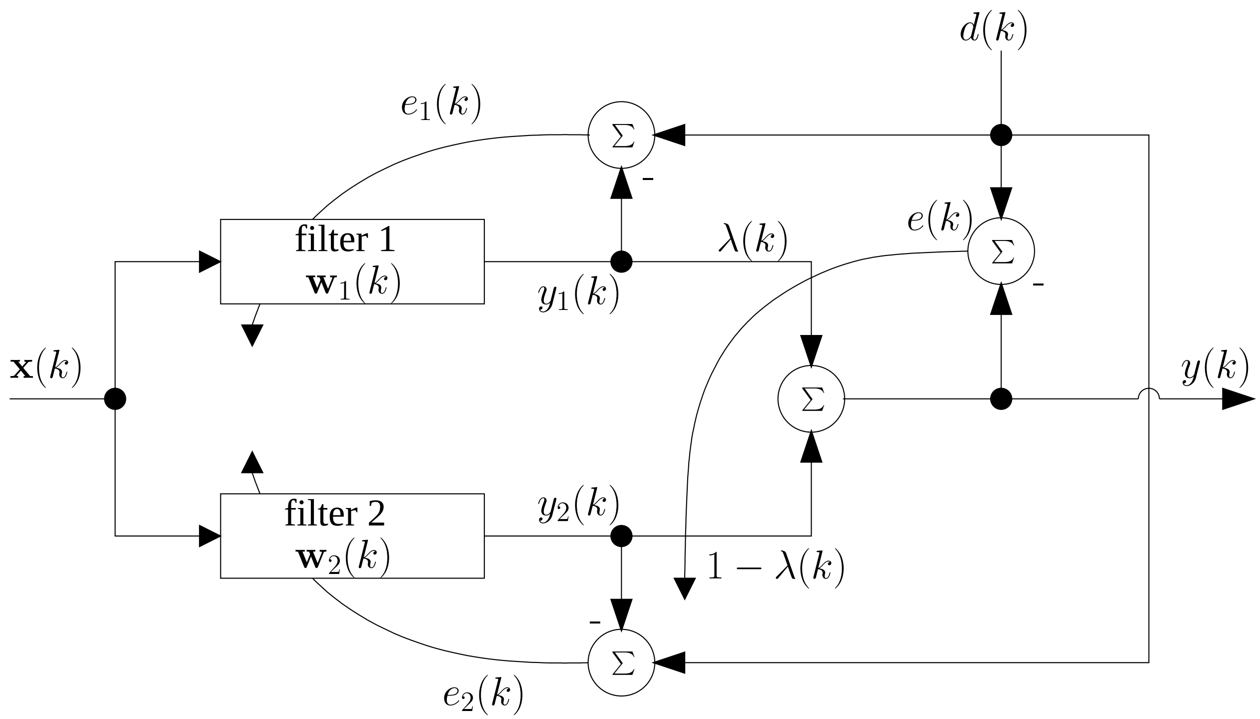
All the algorithms suppress the line noise, some better than other

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

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

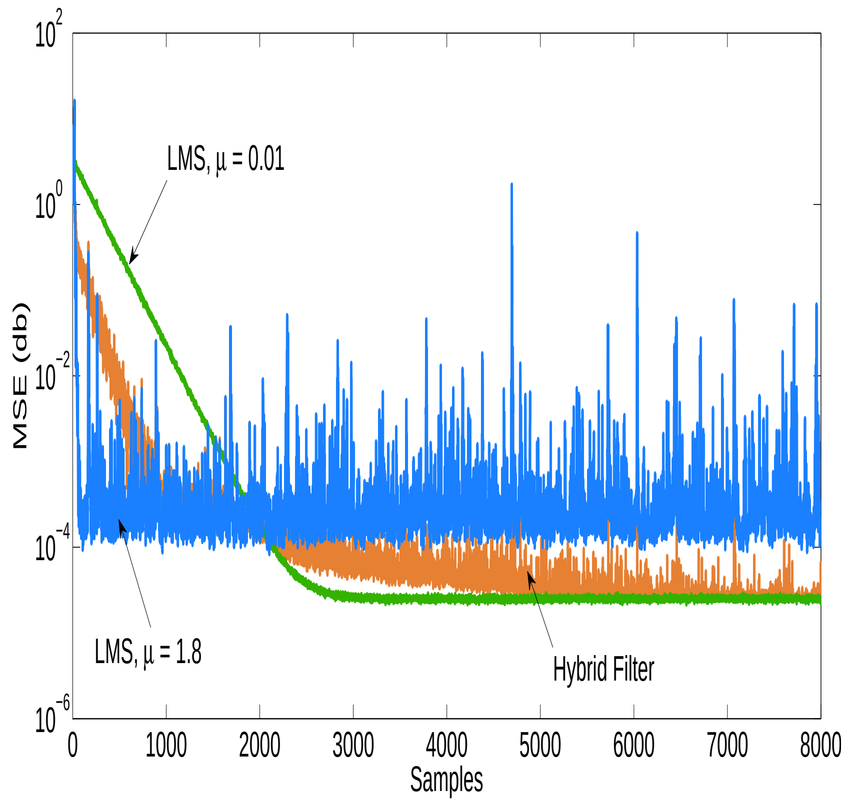
To ensure the combination of adaptive filters remains a convex function it is critical λ remains within the range $0 \le \lambda(k) \le 1$, a hard limit on the set of allowed values for $\lambda(k)$ was therefore implemented.

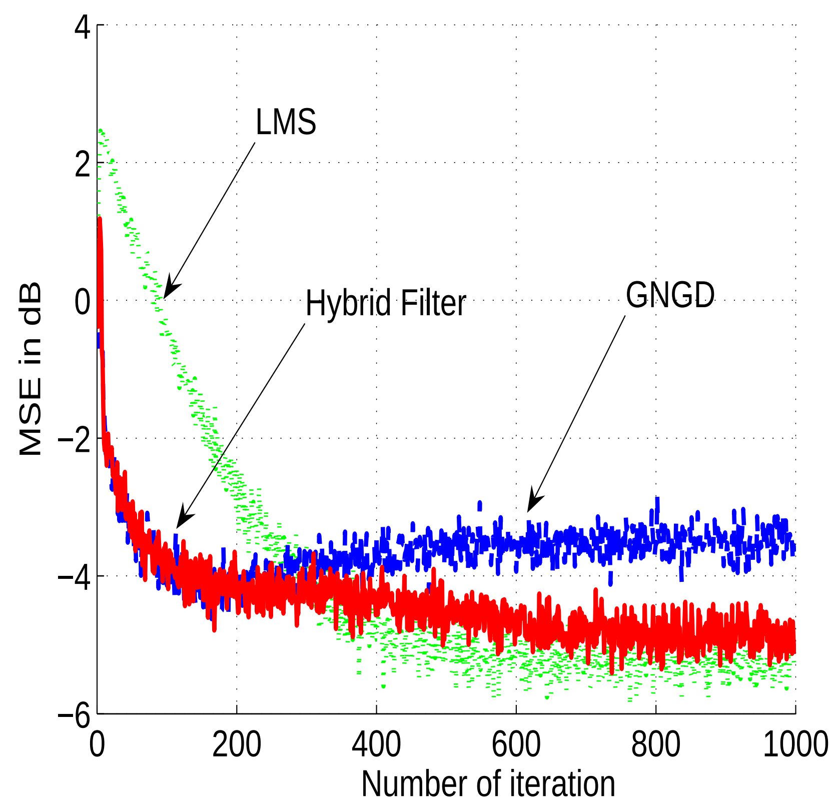
Performance of hybrid filters – prediction setting

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

Hybrid Filters: Summary

- Collaborative adaptive signal processing – lends itself to distributed estimation from multiple sensors
- Distributed estimation – fault tolerance and lower computational complexity
- Also used in communicationss (e.g. real–time allocation of best communications channel for communication with the probe on Mars)
- Usually one fast filter and one slow filter: Fast filter for convergence speed (μ_1 $large$) and slow filter (μ_2 $small$) for good steady state miadjustment
- The learning curve of a hybrid filter should follow the fast filter in the beginning of adaptation and then follow slow filter in the steady state – an optimal "gear shifting" for the learning rate achieved through the architecture
- Possibility of detecting the changes in signal nature

Conclusions

- The LMS is a workhorse in adaptive filtering applications – you can find it virtually everywhere, from channel equalisation in mobile phones, to audio systems, robotics, and biomedical equipment
- Several modifications improve its tracking ability in various scenarios
  - ⊛ To make the LMS independent to the power variations in data and adaptive step size algorithms (NLMS, GASS)
  - ⊛ Various regularisations (GNGD and “leaky” algorithms)
  - ⊛ Collaborative and distributed architectures to increase robustness to sensor failure and enhance stability
  - ⊛ The list is not exhaustive and is ever growing ... some of the future contributions in adaptive systems will be due to you ☞
- Gradient descent algorithms – first order algorithms, we can also use second order algorithms, e.g. the quasi-Newton algorithm
- Some emerging areas, like smart grid or bodysensor networks heavily rely on adaptive signal processing as a mathematical backbone for the analysis of weak signals in drifting noise ☺

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

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

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

Appendix: 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[1] for the gradient $\nabla_{\mu}E(k)$

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

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)

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

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

Notes

○

Notes

○