# 4F7 Adaptive Filters (and Spectrum Estimation)

## Introduction, Wiener Filter and Steepest Descent

Sumeetpal Singh
Email : sss40@eng.cam.ac.uk

# 1 Preliminaries

As handouts you will receive

- All the lecture slides
- Examples sheets
  Some exercises require matlab files available at
  `www-sigproc.eng.cam.ac.uk/~sss40/teaching.html`

Course textbooks
`http://www.eng.cam.ac.uk/teaching`
`/courses/y4/Booklist-IIB-GrpF.pdf`

## 2 In Part 1 of this course we will

- Motivate the need for Filtering and Adaptive Filters
- Study the popular ones
  - Least Mean Square (LMS) and its variants
  - Recursive Least Squares
  - the Kalman filter
  - Hidden Markov Models (HMM)
- We will try to understand these filters
  - with motivating examples
  - by converting problem descriptions into filtering tasks
  - by simulating them
  - with some simple analysis
- Matlab code supplied in handouts and on website
  - Try them for practical experience (very important)

## 3 Course Requirements (Part 1)

- Revise linear algebra, probability, calculus for functions of several variables
- Appreciate the differences between the adaptive filters studied
  – when would you use a particular method
  – understand limitations
- Know how to formulate a given problem as a filtering task
- Understand the main steps in the simple analysis we do
- Acquire practical experience with the various filters using Matlab

# Filtering Defined

- Filtering is the process of removing "noise" from a measured signal in order to reveal or enhance information about some quantity of interest
- Any real data will include some degree of noise from various possible sources
- The noise could be introduced by the measuring system itself
- The noise could be due to the physical environment in which the source generating the signal of interest is immersed in
- Assume you have measurements about some signal of interest available to you at discrete time instances $n = 0, 1, 2, \ldots$
- *Filtering*, means extracting information about a quantity of interest at time $n$ using the data measured up to and including time $n$
- *Prediction*, means to derive information about some quantity of interest at some time $n+m$ in the future ($m > 0$) using the data measured up to and including time $n$

## 5 State-space Model

- A very useful statistical model may be described by the following set of equations:

$$X_{n+1} = AX_n + b_n + V_n$$
$$Y_n = CX_n + d_n + W_n$$

  where $A, C$ are matrices, $b_n, d_n$ are a known deterministic sequence of vectors, $\{V_n\}_{n\geq 0}$ is a i.i.d. zero mean Gaussian noise with variance $Q$, $\{W_n\}_{n\geq 0}$ is a i.i.d. zero mean Gaussian noise with variance $R$

- $\{X_n\}_{n\geq 0}$ is known as the **hidden** state sequence while $\{Y_n\}_{n\geq 0}$ is the **observation** sequence (Is there a model when $\{X_n\}_{n\geq 0}$ and/or $\{Y_n\}_{n\geq 0}$ is finite (or discrete) valued?)
- The filtering problem is: at time $n$, given $\{Y_1, \ldots, Y_n\}$, we wish to obtain an estimate of $X_n$

## 6 Application: Object Tracking

- Let

$$X_n = [\text{xPosition}, \quad \text{xVelocity}, \quad \text{yPosn}, \quad \text{yVel}]^T$$

of a target, if

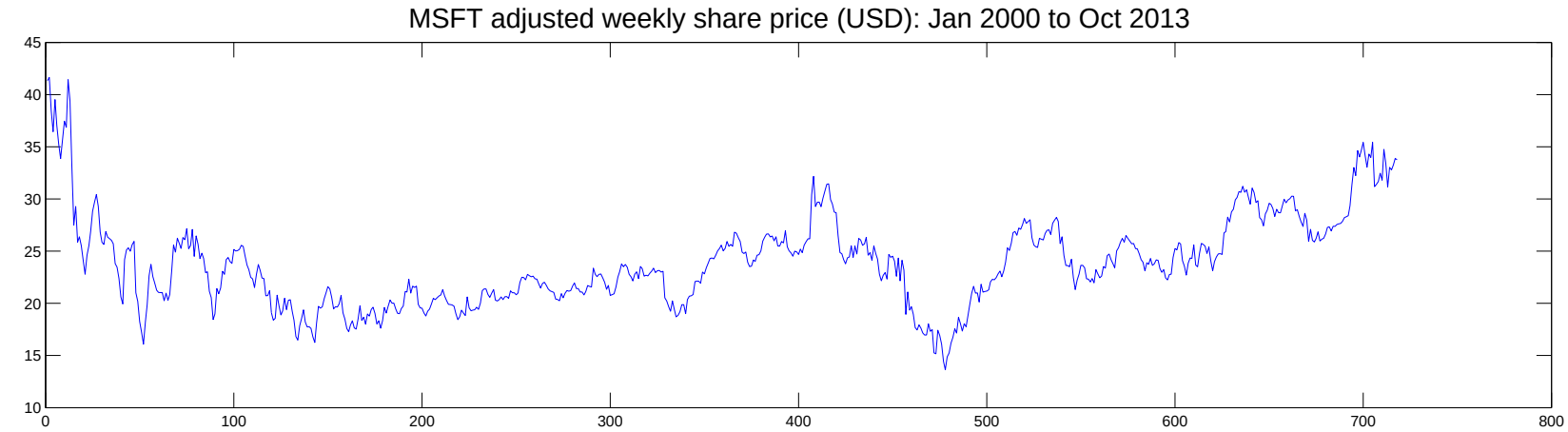
$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

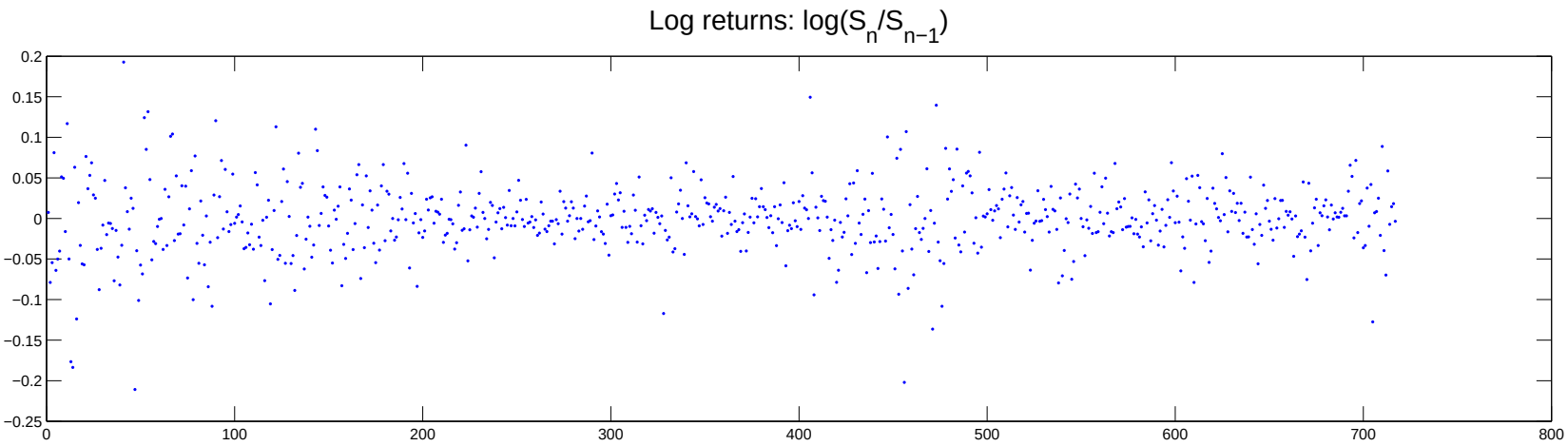
we observe only the targets postition (shown in Figure on next page)

- Important for military applications
- You can also consider measurements model where only the bearing and/or range of the target is measured

## 7 Application: Finance

- Consider the evolution of the price of a financial asset, like a share price or a foreign exchange rate, $S_n$
- The volatility of the price ratio, or of $\{\log(S_n/S_{n-1})\}_{n\geq 1}$ is of considerable interest to the financial analyst
- Shown on the next page is the adjusted values $S_n$ (weekly values) of the MSFT shares (Microsoft) from 13/03/2000–04/10/2007
  On the right-hand side is the plot of $\{\log(S_n/S_{n-1})\}_{n\geq 1}$


MSFT adjusted weekly share price (USD): Jan 2000 to Oct 2013
45
40
35
30
25
20
15
10
0
100
200
300
400
500
600
700
800


Log returns: log(S_n/S_n−1)
0.2
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−0.2
−0.25
0
100
200
300
400
500
600
700
800

- Notice how $\{Y_n = \log(S_n/S_{n-1})\}_{n\geq 1}$ looks like $Y_n = \beta_n W_n$ where $W_n$ is Gaussian with mean 0 and variance 1 while the variance of $Y_n$ is being modulated by $\beta_n$, which is known as the *volatility*
- We would like to estimate this volatility and we may do so by considering the following model

$$X_{n+1} = \phi X_n + \sigma V_n$$
$$Y_n = \beta \exp(X_n) W_n$$

  where $(\phi, \sigma, \beta)$ are known constants and $V_n$ is Gaussian with mean 0 and variance 1
- The filtering problem is: at time $n$, given $\{Y_1, \ldots, Y_n\}$, we wish to obtain an estimate of $X_n$
- We can approximately solve this problem by "linearising the model" and then use the Kalman filter which we learn in this course

- We will return to the state-space formulation later in the course. We now start with a more simple formulation that requires stationarity of the signals involved
- The state-space formulation arose later because of new applications where non-stationarity was intrinsic to the problems

## 8 More Mathematical notation

- Bold symbols will denote matrices or vectors
  – bold capital letter denotes a matrix, e.g., $\mathbf{A}$
  – bold lowercase letter denotes a vector, e.g., $\mathbf{p}$
- superscript "T" denotes transpose, "H" Hermitian transpose
- for a function $f : \mathbb{R}^M \rightarrow \mathbb{R}$, let $\nabla f(\mathbf{y})$ denote

$$\nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{y}} = \left[\frac{\partial f(\mathbf{y})}{\partial x_1}, \ldots, \frac{\partial f(\mathbf{y})}{\partial x_M}\right]^{\mathrm{T}}$$

## 9 Wiener model: the setup

- The **input** signal $\{u(n)\}$
- The **reference** signal $\{d(n)\}$
- The filter $\mathbf{h} = [h_0, h_1, \ldots, h_{M-1}]^{\mathsf{T}} \in \mathbb{R}^M$
- The **output** signal $\{y(n)\}$ is

$$y(n) = \sum_{k=0}^{M-1} h_k u(n-k) = \mathbf{u}^{\mathsf{T}}(n)\,\mathbf{h}$$

- The **error** signal $\{e(n)\}$

$$e(n) = d(n) - \mathbf{u}^{\mathsf{T}}(n)\,\mathbf{h}$$

## 10 Performance criterion

- Now that we have all the signals, we need a performance function – the Wiener filter is a solution to an optimization problem
- The **cost** function is

$$J_n(\mathbf{h}) \triangleq E\left\{e^2(n)\right\},$$

expanding gives

$$e^2(n) = d^2(n) + \mathbf{h}^{\mathsf{T}}\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\,\mathbf{h} - 2\mathbf{h}^{\mathsf{T}}\mathbf{u}(n)\,d(n),$$

taking the expectation yields

$$J_n(\mathbf{h}) = E\left\{d^2(n)\right\} + \mathbf{h}^{\mathsf{T}}E\left\{\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\right\}\mathbf{h} - 2\mathbf{h}^{\mathsf{T}}E\left\{\mathbf{u}(n)\,d(n)\right\}$$

- We call
  - $\sigma_d^2(n) = E\left\{d^2(n)\right\}$ reference signal power
  - $\mathbf{R}(n) = E\left\{\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\right\}$ input signal autocorrelation matrix
  - $\mathbf{p}(n) = E\left\{\mathbf{u}(n)\,d(n)\right\}$, crosscorrelation vector

## 11 Stationarity assumptions

- $\mathbf{R}(n) = E\left\{\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\right\}$ input signal autocorrelation matrix,
  $[\mathbf{R}(n)]_{i,j} = E\left\{u(n-i+1)\,u(n-j+1)\right\}$

  If $\{u(n)\}$ is **2nd-order stationary** then,
  $E\left\{u(n-i+1)\,u(n-j+1)\right\} = E\left[u(0)\,u(i-j)\right]$
  $= r_u(i-j)$

- $\mathbf{p}(n) = E\left\{\mathbf{u}(n)\,d(n)\right\}$, crosscorrelation of input and reference signal,
  $[\mathbf{p}(n)]_i = E\left\{u(n-i+1)\,d(n)\right\}$
  If $\{u(n)\}$ and $\{d(n)\}$ are **jointly stationary** then,
  $E\left\{u(n-i+1)\,d(n)\right\}$
  $= E\left\{u(-i+1)\,d(0)\right\}$
  $= r_{ud}(-i+1)$

- It is the **time-shift** that the statistics depend on

## 12 Wiener filter

- **Stationarity** assumptions imply a time-independent cost,

$$J(\mathbf{h}) = \sigma_d^2 + \mathbf{h}^{\mathrm{T}}\mathbf{R}\mathbf{h} - 2\mathbf{h}^{\mathrm{T}}\mathbf{p},$$

  which is also quadratic in $\mathbf{h}$

- To minimise, set the gradient to zero, i.e., $\bigtriangledown J(\mathbf{h}) = 0$, which gives

$$2\mathbf{R}\mathbf{h} - 2\mathbf{p} = 0 \quad (\textbf{Normal Equation})$$

$$\mathbf{h}_{\mathrm{opt}} = \mathbf{R}^{-1}\mathbf{p} \quad (\textbf{Wiener-Hopf filter})$$

- Normal equation has an **orthogonality** interpretation

$$E\{\mathbf{u}(n)\left[d(n) - \mathbf{u}^{\mathrm{T}}(n)\,\mathbf{h}_{\mathrm{opt}}\right]\} = 0$$

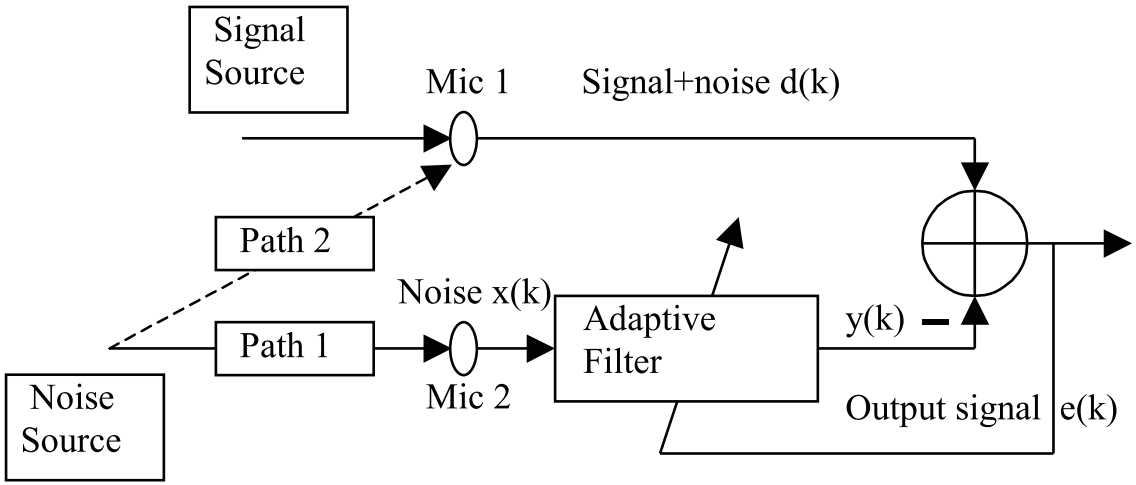
  $e(n)$ and $\mathbf{u}(n)$ are **orthogonal**

## 13 Remarks on Wiener filter

- We defined a time-independent cost function and the Wiener filter minimised it
- Didn't need to know $\{d(n)\}$ or $\{u(n)\}$ but only 2nd order statistics
  - $\sigma_d^2 = E\left\{d^2(n)\right\}$ reference signal power $\times$
  - $\mathbf{R} = E\left\{\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\right\}$ input signal autocorrelation matrix ✓
  - $\mathbf{p} = E\{\mathbf{u}(n)\,d(n)\}$, crosscorrelation ✓
- In practice we can solve for $\mathbf{h}_{\text{opt}} = \mathbf{R}^{-1}\mathbf{p}$ without requiring $\mathbf{R}$, $\mathbf{p}$, or inverting any matrices

## 14 Application: Noise Cancelation

- The principle behind this application is as follows. You have a signal of interest which is being corrupted by a source of interference. You can make measurements of this signal of interest but not without the corrupting interference
- However, you are also able to make measurements at the source of the interference itself where the interference dominates over the signal of interest
- The measured interference signal is filtered so that it becomes a replica of the interference signal that is present in the measurements of the signal of interest
- Now subtract this replica from the corrupted signal of interest to leave only the signal of interest. This is illustrated in following example




- Correction to Figure: $x(k)$ should be $u(k)$ in this figure
- The adaptive filter aims to minimise the error between the filtered noise and the output of Mic 1
- Recovered signal is $d(n) - \mathbf{h}^T\mathbf{u}(n)$

- Mic 1: Reference signal

$$d(n) = \underbrace{s(n)}_{\text{signal of interest}} + \underbrace{v(n)}_{\text{noise}}$$

  $s\,(n)$ and $v\,(n)$ statistically independent

- Aim: recover signal of interest
- Method: use another mic, Mic 2, to record only noise, $u(n)$
- Obviously $u\,(n) \neq v\,(n)$ but $u\,(n)$ and $v\,(n)$ are correlated
- Now **filter** recorded noise $u\,(n)$ with $\mathbf{h}$ to make it more like $v(n)$
- Recovered signal is $d(n) - \mathbf{h}^T\mathbf{u}(n)$

## 15 Application: Channel equalization

- Channel equalizers are important for reliable communication of digital data over non-ideal channels. Let $\{x(n)\}$ be the digital signal to be transmitted over the channel. This signal takes as values plus or minus 1.
- This signal is input to a pulse generator which produces a pulse of amplitude $A$ at time $n$ if $x(n) = 1$ or $-A$ otherwise. These pulses are modulated and transmitted over the channel.
- The receiver demodulates and samples the received waveform which produces the signal $\{u(n)\}$. The demodulated signal is distorted by the channel. The pulse shapes are distorted causing neighboring pulses to interfere with each other, which is known as Intersymbol Interference (ISI).
- A model for $\{u(n)\}$ is

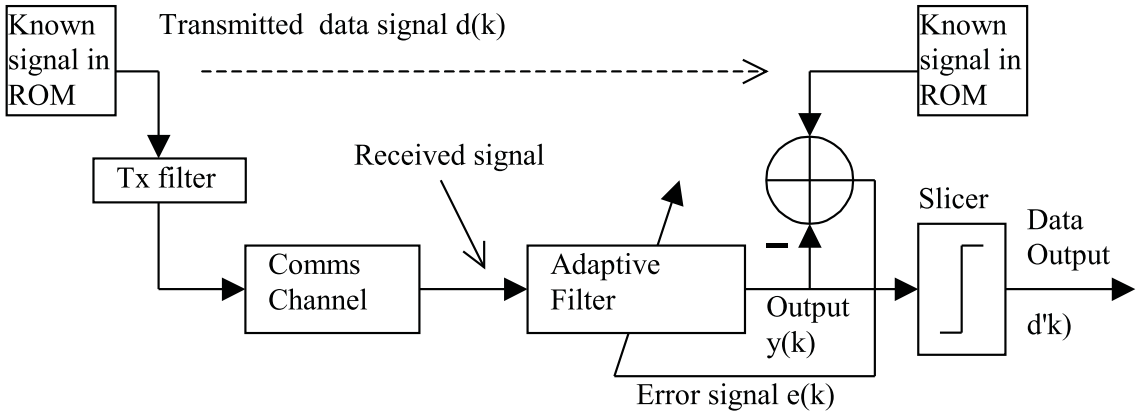
$$u(n) = \sum_{k=0}^{n} x(k)h'(n-k) + v(n).$$

- This model is motivated by physical reasons: the signal is subject to multi-path fading which means that the received signal is the sum of delayed scaled versions. Here $v(n)$ is additive noise.
- The decision on the transmitted bit is a simple threshold device:

$$\widehat{x}(n) = \begin{cases} 1 & \text{if } u(n) \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

- To improve the chances of correct decisions, an equalizer is used to minimise the channel distortion.

- A copy of $\{x(n)\}_{n=1}^{T}$ is available at the receiver
  – called a **training signal** in comms literature
  – this copy will be our **reference signal** though, $\{d(n)\}_{n=1}^{T} = \{x(n)\}_{n=1}^{T}$
- The receiver will
  – filter $\{u(n)\}_{n\geq 0}$ with $\mathbf{h}$ to give $y(n)$
  – adapt $\mathbf{h}$ to try to make $\{y(n)\}_{n=1}^{T}$ and $\{d(n)\}_{n=1}^{T}$ identical
  – the receiver **has only** $T$ **time** points to do so
- The designed $\mathbf{h}$ will then be used for time $n > T$
- $\mathbf{h}$ should invert the effect of the channel

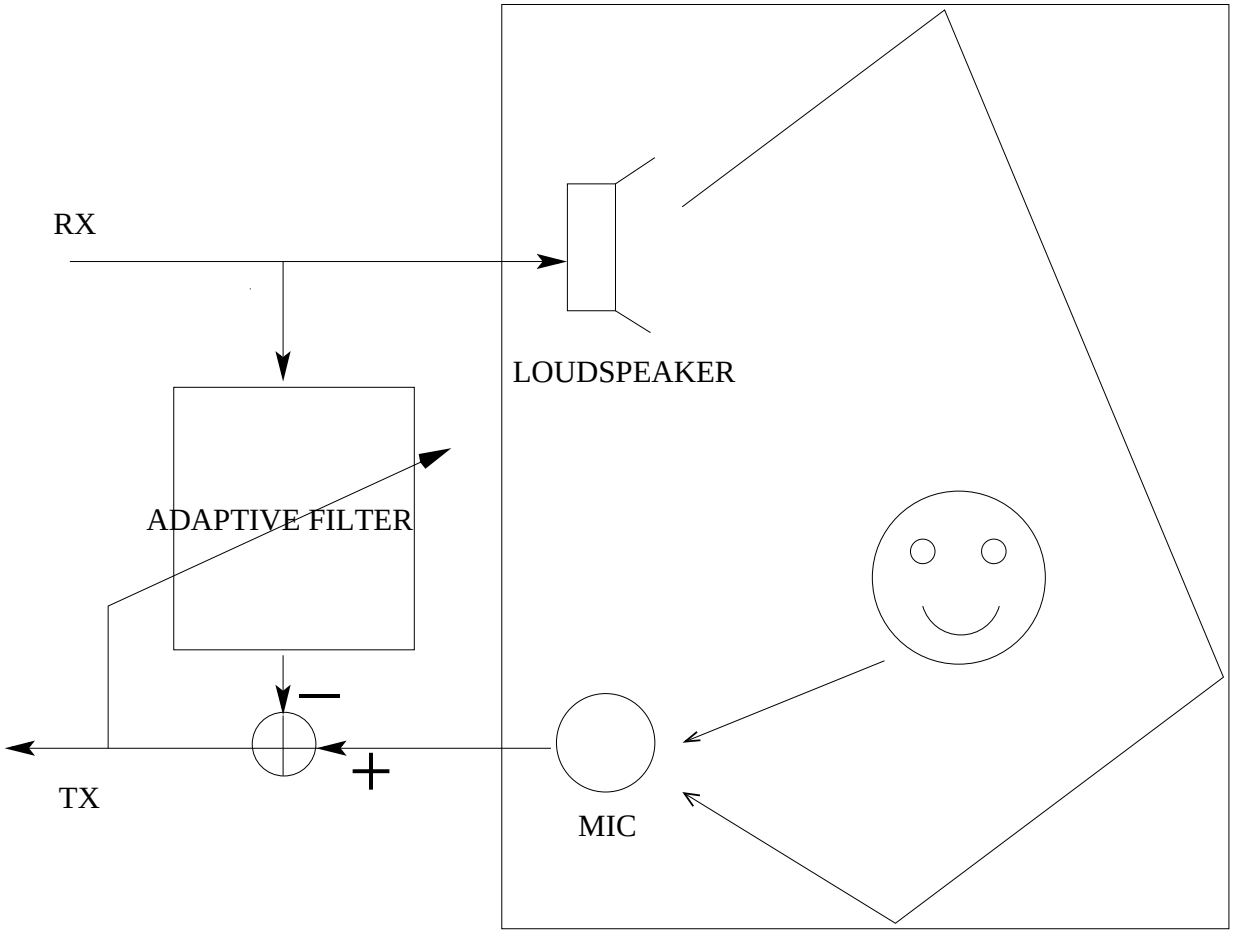



# Channel Equalization

## 16 Application: Echo Cancellation

- A hands-free unit includes a microphone and a loudspeaker
- Voice of far speaker, comes out of loudspeaker, reflected (the echo), and sent back through mic to the far speaker
- Echoes prohibit a normal conversation and must be cancelled. Mathematically,
  – $u(n)$ our of loudspeaker (speech signal of the far speaker)
  – $s(n)$ signal of near speaker
  – $H_{\text{room}}[u](n)$, the echo
- Call the signal into the mic $d(n)$,

$$d(n) = \underbrace{H_{\text{room}}[u](n)}_{\text{echo}} + \underbrace{s(n)}_{\text{near speaker speech}}$$

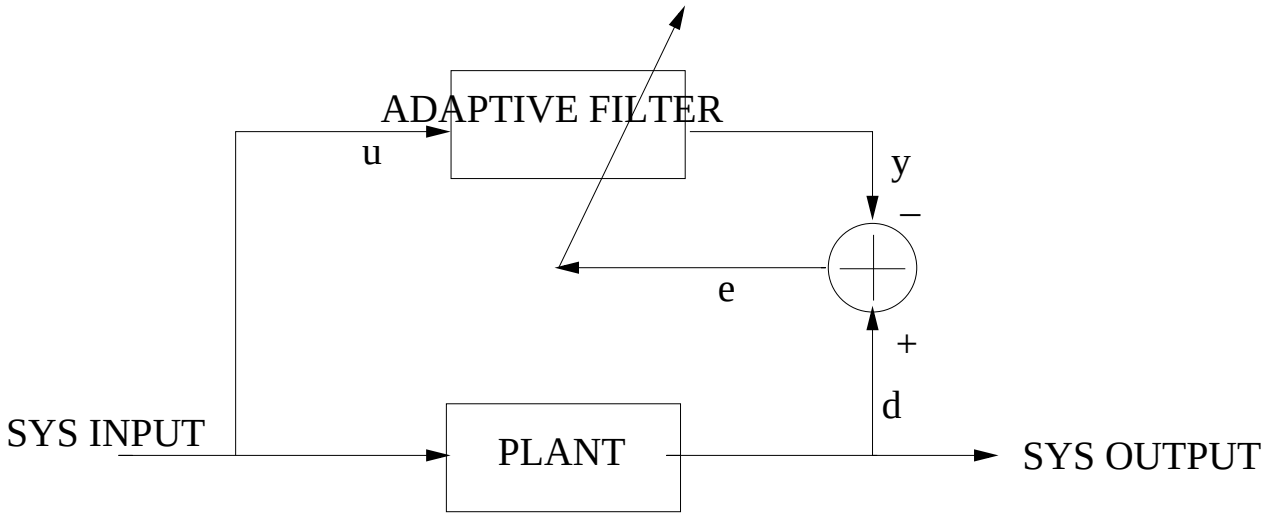



Echo Cancellation

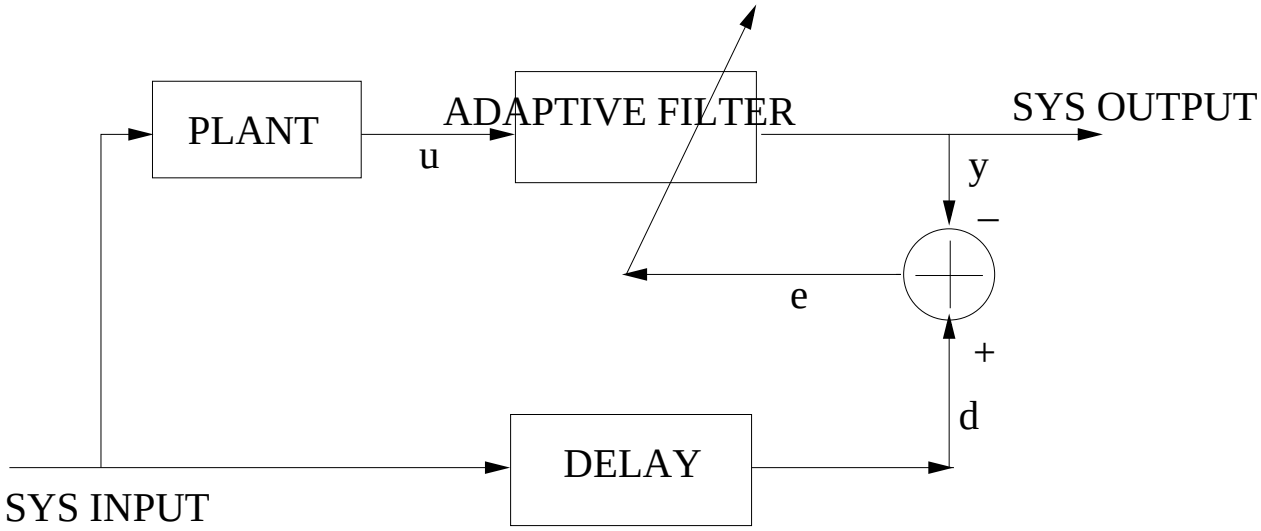
## 17 Four Classes of Applications

- Adaptive filters have been successfully applied in diverse fields such as communications, radar, seismology, biomedical engineering
- Although the applications are diverse, they share a basic common feature
- An input signal is filtered and compared to a desired response. This yields the error signal which is used to adjust the filter that is being applied to the input signal. The adjustable coefficients of the filter may take the form of tap weights
- The essential difference between the various applications of adaptive filtering is the manner in which the input signal and desired response are defined. There are four basic classes of adaptive filtering applications as discussed below

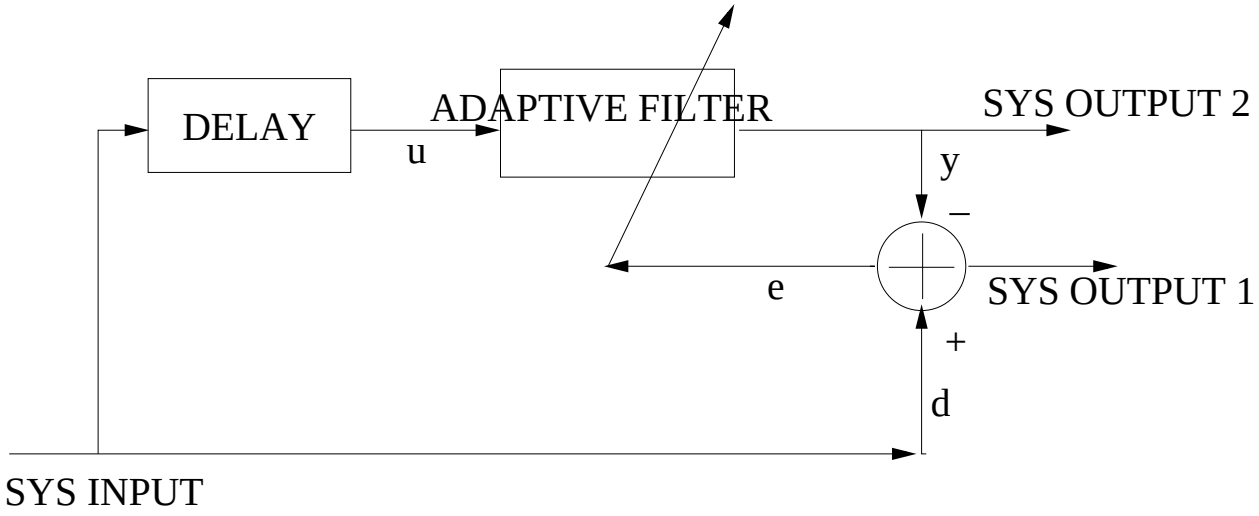
*Identification:* A mathematical model is essential component for analysis and control. An adaptive filter is used to provide a linear model that represents the best fit (in some sense) to an unknown *plant*. The plant and adaptive filter are driven by the same input and the plant output supplies the desired response. If the plant is dynamic in nature, the adaptive filter may be able to track the time-varying plant model provided it changes slower than the adaptation rate.



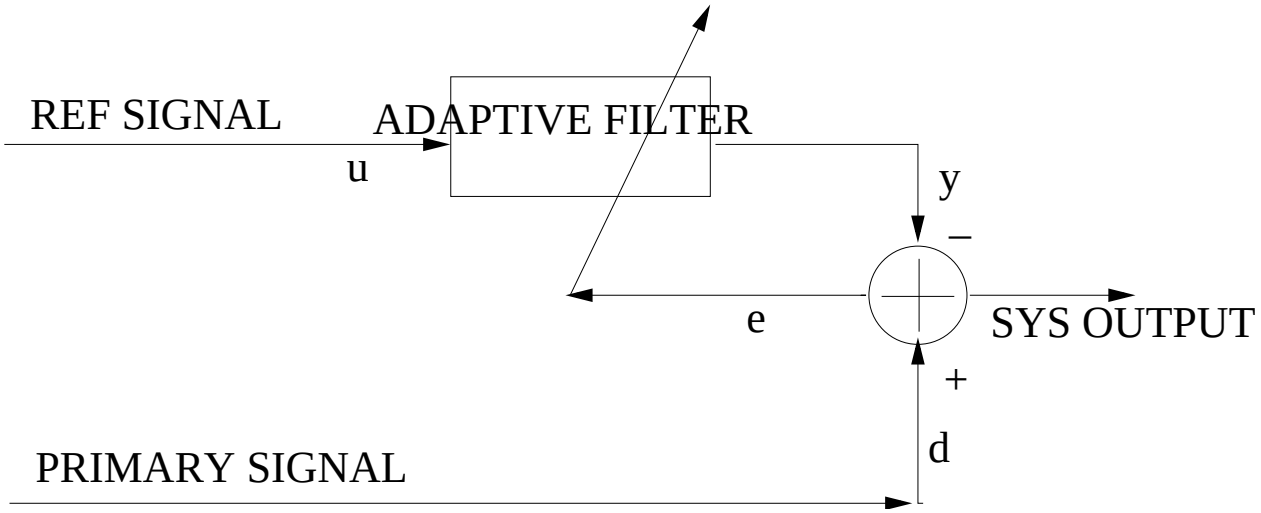
*Inverse modeling:* The function of the adaptive filter here is to provide the best fitting (in some sense) *inverse* model to the *unknown noisy plant*. Ideally the inverse model will have a transfer function equal to the inverse of the plant's transfer function. The desired response is a delayed version of the plant's input. In some instances a delay is not necessary.



*Prediction:* The function of the adaptive filter is to provide the best prediction (in some sense) of the present value of the input signal. The input of the filter are the past values, hence the delay. The desired response is the present value.



*Interference cancellation:* The adaptive filter is used to cancel interference which is present along side the signal of interest in the *primary* signal. The input to the filter is the *reference* signal. This signal is derived from a sensor located relative to the sensor supplying the primary signal in such a way that the signal of interest is undetectable. The adaptive filter will synthesise the interference in the primary signal using the reference signal which is then subtracted from the primary signal.



## 18 Gradient Descent

- Consider the real valued function $f : \mathbb{R}^M \to \mathbb{R}$ and we wish to minimise it
  – one way is to identify the set

  $$\{\mathbf{x} : \nabla f(\mathbf{x}) = 0\}$$

  and test each point to find the minima
- An iterative way: let $\mathbf{x}(k)$ be our current best solution. We improve it by

  $$\mathbf{x}(k+1) = \mathbf{x}(k) - \frac{\mu}{2}\nabla f(\mathbf{x}(k))$$

  where $\mu$ is the **stepsize**
- For example, if $M = 1$ and $f(x) = x^2$,

  $$\begin{aligned} x(k+1) &= x(k) - \mu x(k) \\ &= x(k)(1-\mu), \end{aligned}$$

  and $|1 - \mu| < 1$ implies $x(k) \to 0$

## 19 Steepest Descent

- $J(\mathbf{h})$ for Wiener filtering is a convex function and can be minimised by gradient descent
- Adaptation in the negative direction of the gradient

$$\mathbf{h}(n+1) = \mathbf{h}(n) - \frac{\mu}{2} \bigtriangledown J(\mathbf{h}(n))$$

- Gradient of $J$ is

$$\begin{aligned} \bigtriangledown J(\mathbf{h}(n)) &= 2\mathbf{R}\mathbf{h}(n) - 2\mathbf{p} \\ &= 2E\left\{\mathbf{u}(k)\,\mathbf{u}^{\mathrm{T}}(k)\right\}\mathbf{h}(n) \\ &\quad - 2E\{\mathbf{u}(k)\,d(k)\} \end{aligned}$$

Since true for any $k$, set $k = n$ and let $e(n) = d(n) - \mathbf{u}^{\mathrm{T}}(n)\,\mathbf{h}(n)$.

- The Steepest Descent (SD) algorithm is

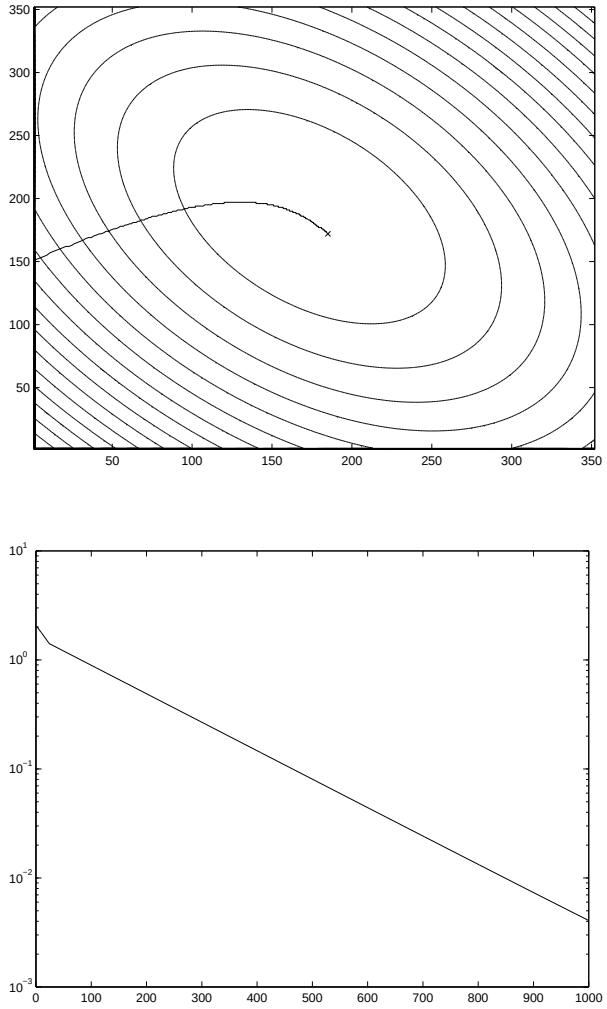
$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu E\{\mathbf{u}(n)\,e(n)\}, \quad n \geq 0$$

- Computational complexity: $O\left(M^2\right)$ (multiplications)

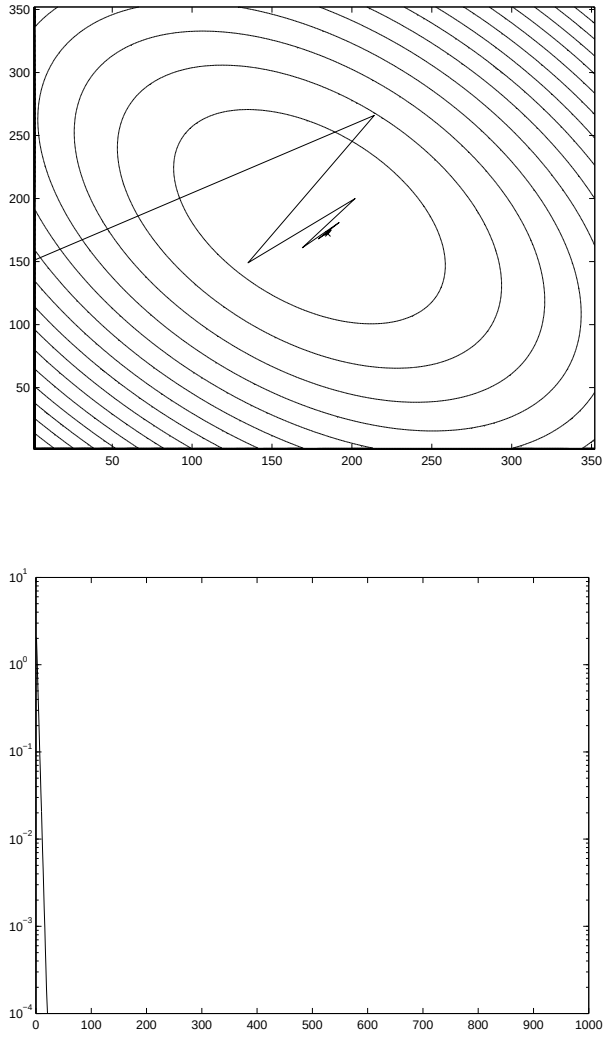
## 20 Matlab code for an example

```
% autocorrelation matrix
R=[1.1 0.5; 0.5 1.1];
% crosscorrelation matrix
p=[0.5271; -0.4458];
% initial filter value & stepsize
h=[-1;-1];
mu=0.001;
% record the evolution of the h
T=1000;
H=[];
% SD algorithm
for k=1:T
h=h-mu*(R*h-p);
H=[H,h];
end
```

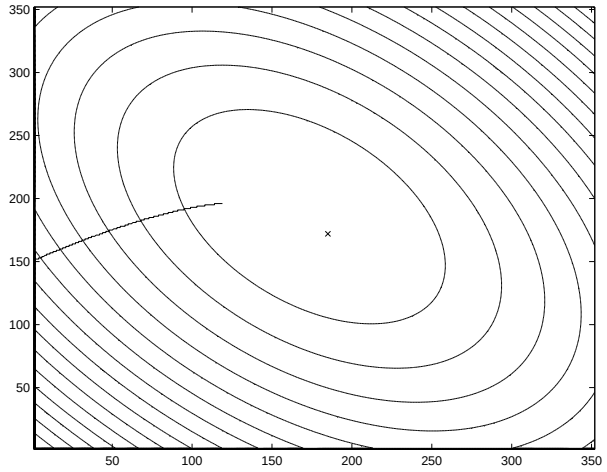
- $\mathbf{R} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix}$, $\mathbf{p} = \begin{bmatrix} 0.5272 \\ -0.4458 \end{bmatrix} \Rightarrow \mathbf{h}_{\text{opt}} = \begin{bmatrix} 0.8360 \\ -0.7853 \end{bmatrix}$
- Results obtained using 4 different stepsizes $\mu$
  – what will be the effect of the different stepsizes?
- Thus is just an example; we wouldn't use SD for such simple problems
  – when would we use SD though?
- Error in plots measured as $\left\|\mathbf{h}\left(n\right) - \mathbf{h}_{\text{opt}}\right\|_1$

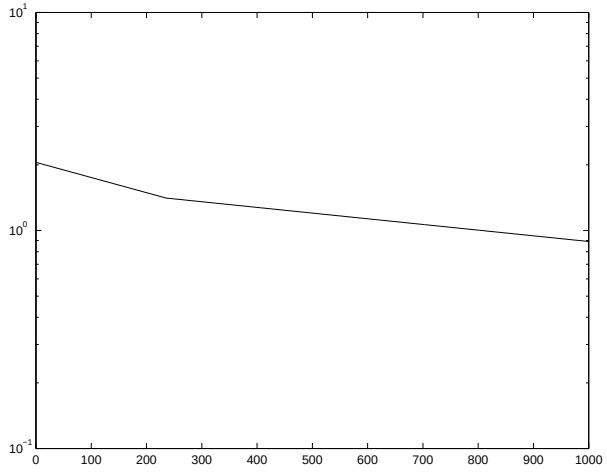



$\mathbf{h}(n)$ (above) and $\|\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\|_1$ against iteration number $n$ for $\mu = 0.01$

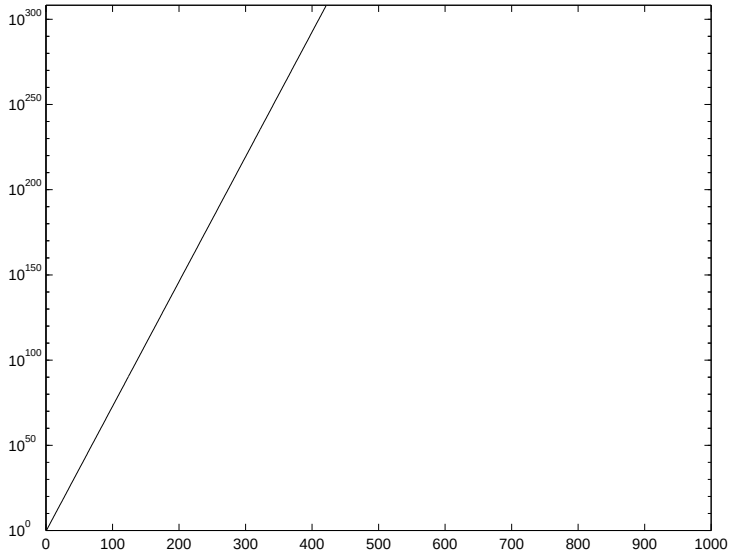



$\mathbf{h}(n)$ (above) and $\|\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\|_1$ against iteration number $n$ for $\mu = 1$




$\mathbf{h}(n)$ (above) and $\|\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\|_1$ against iteration number $n$ for $\mu = 0.001$




Evolution of $\left\|\mathbf{h}\left(n\right)-\mathbf{h}_{\text{opt}}\right\|_1$ against iteration number for $\mu=4$

## 21 Convergence of SD

- We have seen SD performance depends critically on the stepsizes – either converges slowly, fast or diverges
- $\mathbf{R} = E\left\{\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\right\}$ is symmetric and positive semidefinite,

$$\mathbf{v}^{\mathsf{T}}\mathbf{R}\mathbf{v} = \mathbf{v}^{\mathsf{T}}E\left\{\mathbf{u}(n)\,\mathbf{u}^{\mathsf{T}}(n)\right\}\mathbf{v} = E\left\{\left|\mathbf{v}^{\mathsf{T}}\mathbf{u}(n)\right|^2\right\} \geq 0$$

- Let $\lambda_k$ and $\mathbf{v}_k$ be respectively the $k^{\text{th}}$ eigenvalue and eigenvector of $\mathbf{R}$, i.e., $\mathbf{R}\mathbf{v}_k = \lambda_k\mathbf{v}_k$ (note $\lambda_k \geq 0$ (why?))

$$\mathbf{R}\underbrace{[\mathbf{v_1}, \cdots, \mathbf{v}_M]}_{\mathbf{Q}} = [\mathbf{v_1} \cdots \mathbf{v}_M]\underbrace{\operatorname{diag}(\lambda_1, \ldots, \lambda_M)}_{\mathbf{\Lambda}}$$

- and if $\mathbf{R}$ is non-singular (invertible), eigenvectors can be chosen to be orthonormal,

$$\mathbf{Q}^{\mathsf{T}}\mathbf{Q} = \mathbf{I}$$

- Note $\mathbf{Q}^{\mathsf{T}}\mathbf{Q}\mathbf{Q}^{-1} = \mathbf{I}\mathbf{Q}^{-1}$, we have $\mathbf{Q}^{-1} = \mathbf{Q}^{\mathsf{T}}$

- Thus,

$$\mathbf{R} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathsf{T}}$$

- SD update rule is $\mathbf{h}(n+1) = \mathbf{h}(n) + \mu(\mathbf{p} - \mathbf{R}\mathbf{h}(n))$, using $\mathbf{R}\mathbf{h}_{\text{opt}} = \mathbf{p}$,

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu\mathbf{R}\left(\mathbf{h}_{\text{opt}} - \mathbf{h}(n)\right),$$

subtracting $\mathbf{h}_{\text{opt}}$ from both sides,

$$\begin{aligned}\mathbf{h}(n+1) - \mathbf{h}_{\text{opt}} &= \mathbf{h}(n) - \mathbf{h}_{\text{opt}} - \mu\mathbf{R}\left(\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\right) \\ &= (\mathbf{I} - \mu\mathbf{R})\left(\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\right) \\ &= \left(\mathbf{I} - \mu\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathsf{T}}\right)\left(\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\right),\end{aligned}$$

where last line follows eigendecomposition

- Now multiply by $\mathbf{Q}^{\mathsf{T}}$

$$\begin{aligned}\mathbf{Q}^{\mathsf{T}}\left(\mathbf{h}(n+1) - \mathbf{h}_{\text{opt}}\right) &= \mathbf{Q}^{\mathsf{T}}\left(\mathbf{I} - \mu\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathsf{T}}\right) \\ &\quad \times \left(\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\right) \\ &= (\mathbf{I} - \mu\mathbf{\Lambda})\,\mathbf{Q}^{\mathsf{T}}\left(\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\right)\end{aligned}$$

- This expression is very convenient, setting $\nu(n) = \mathbf{Q}^{\mathsf{T}}\left(\mathbf{h}(n) - \mathbf{h}_{\text{opt}}\right)$ gives

$$\nu(n+1) = (\mathbf{I} - \mu\mathbf{\Lambda})\,\nu(n)$$

- $(\mathbf{I} - \mu\mathbf{\Lambda})$ is a diagonal matrix. In component form,

$$\nu(n) = [\nu_1(n), \ldots, \nu_M(n)]^{\mathsf{T}},$$

gives, for $k = 1, \ldots, M$,

$$\begin{aligned}\nu_k(n+1) &= (1 - \mu\lambda_k)\,\nu_k(n) \\ &= (1 - \mu\lambda_k)^{n+1}\,\nu_k(0)\end{aligned}$$

- We can now assert **stability**, $\nu_k(n) \to 0$ provided $|1 - \mu\lambda_k| < 1$, or

$$0 < \mu < 2/\lambda_k$$

- Since we want $\mathbf{h}(n) \to \mathbf{h}_{\text{opt}}$, we require $\nu(n) \to 0$, or equivalently $\nu_k(n) \to 0$ for all $k$
- Thus we get the **stability condition**

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

- In our example, $\mathbf{R} = \begin{bmatrix} 1.1 & 0.5 \\ 0.5 & 1.1 \end{bmatrix} \Rightarrow \lambda_{\max} = 1.6$ and $\lambda_{\min} = 0.6$

  $\Rightarrow \ \mu < 1.25$

- This explains observed divergence of $\mu = 4$