Introduction To Equalization

Presented By :

Guy Wolf

Roy Ron

Guy Shwartz

(Adavanced DSP, Dr.Uri Mahlab)

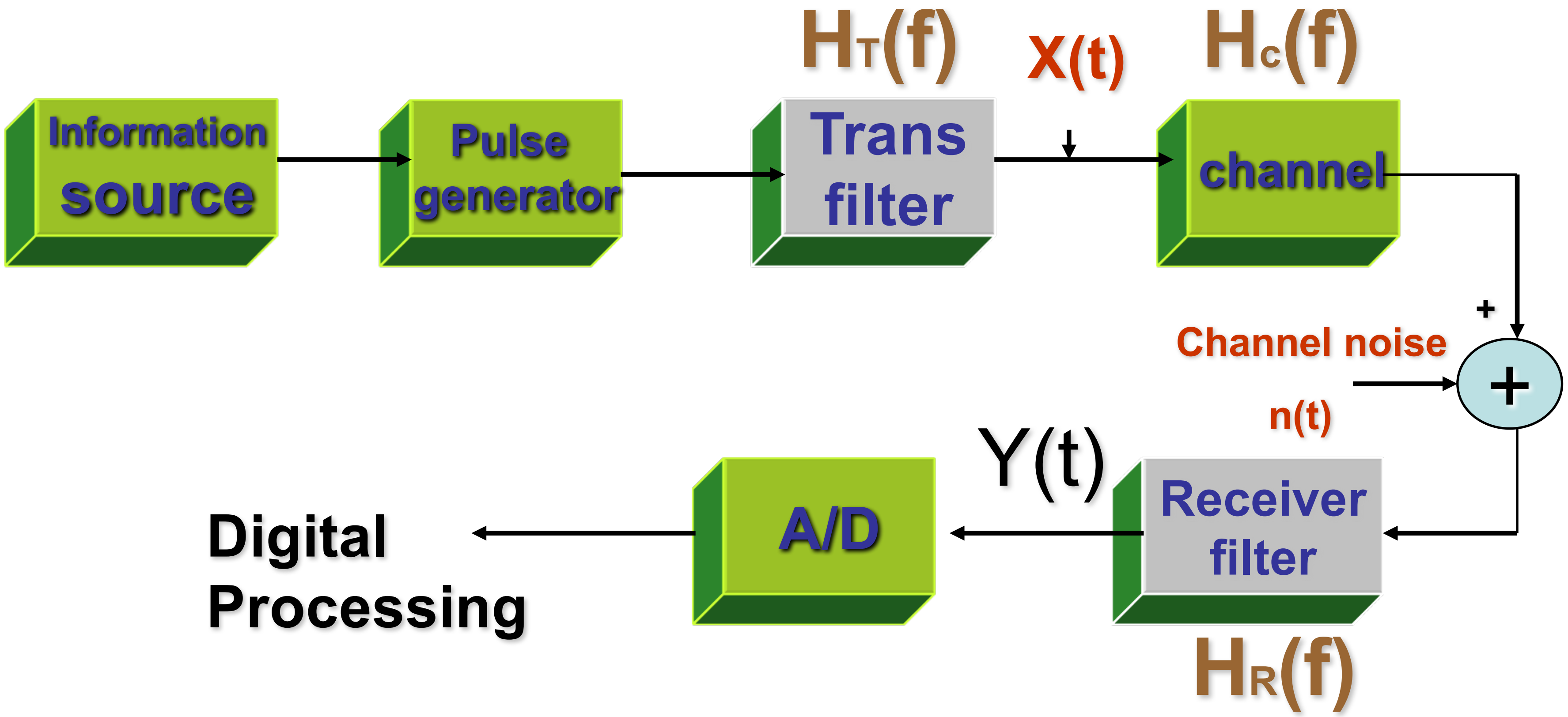
HAIT

14.5.04

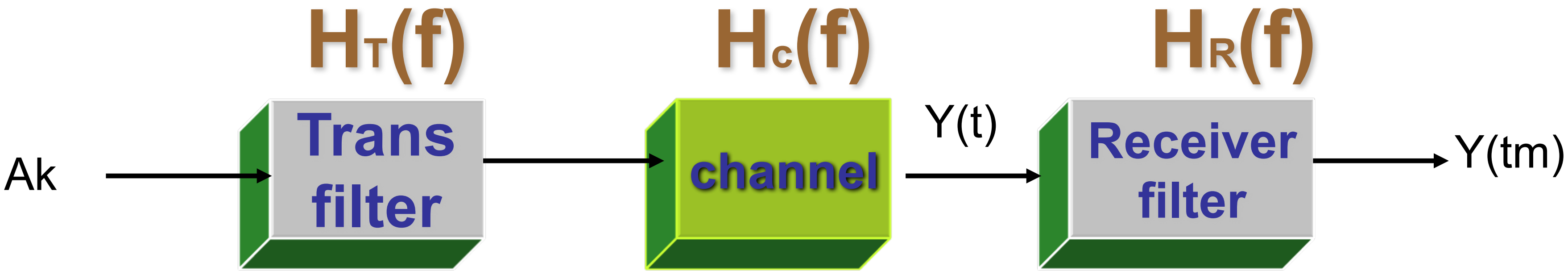
TOC

- Communication system model
- Need for equalization
- ZFE
- MSE criterion
- LS
- LMS
- Blind Equalization – Concepts
- Turbo Equalization – Concepts
- MLSE - Concepts

Basic Digital Communication System



Basic Communication System

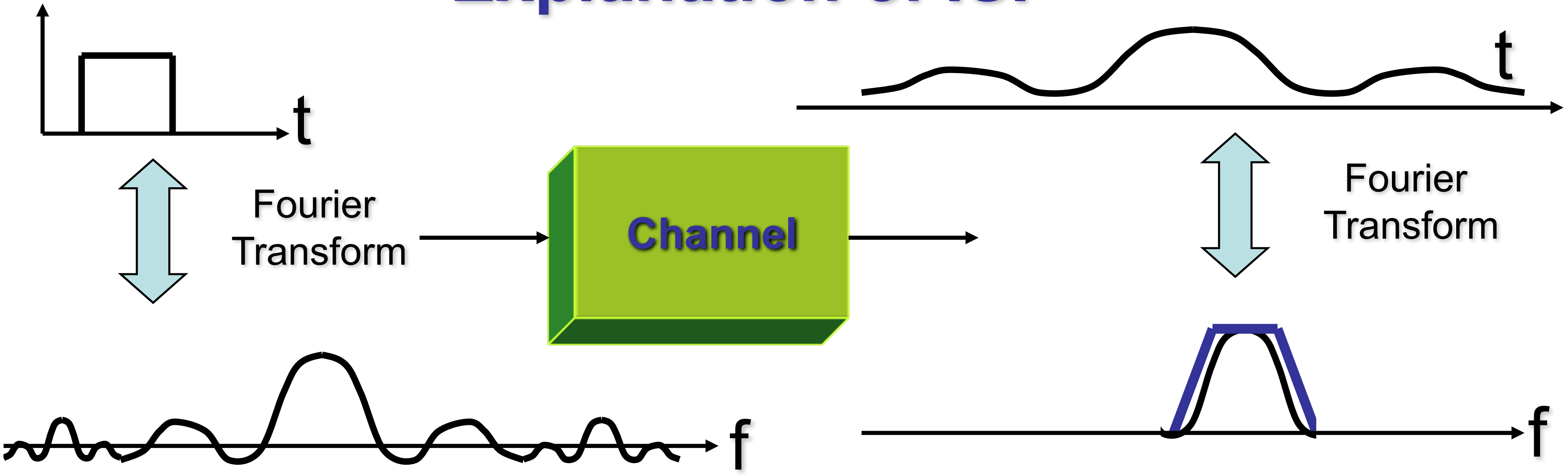


$$Y(t) = \sum_k A_k h_c(t - t_d - kT_b) + n_0(t)$$

The received Signal is the transmitted signal, convolved with the channel And added with AWGN (Neglecting HTx,HRx)

$$Y(t_m) = A_m + \underbrace{\sum_{K \neq m} A_k h_c[(m-k)T_b]}_{\text{ISI - Inter Symbol Interference}} + n_0(t_m)$$

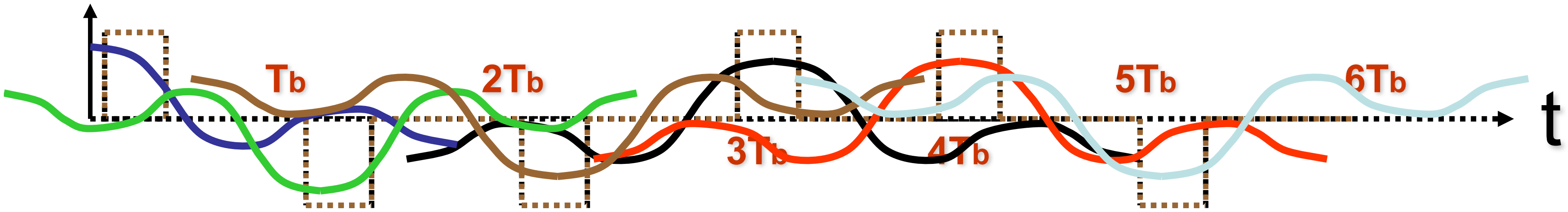
Explanation of ISI



Fourier Transform

Channel

Fourier Transform



Reasons for ISI

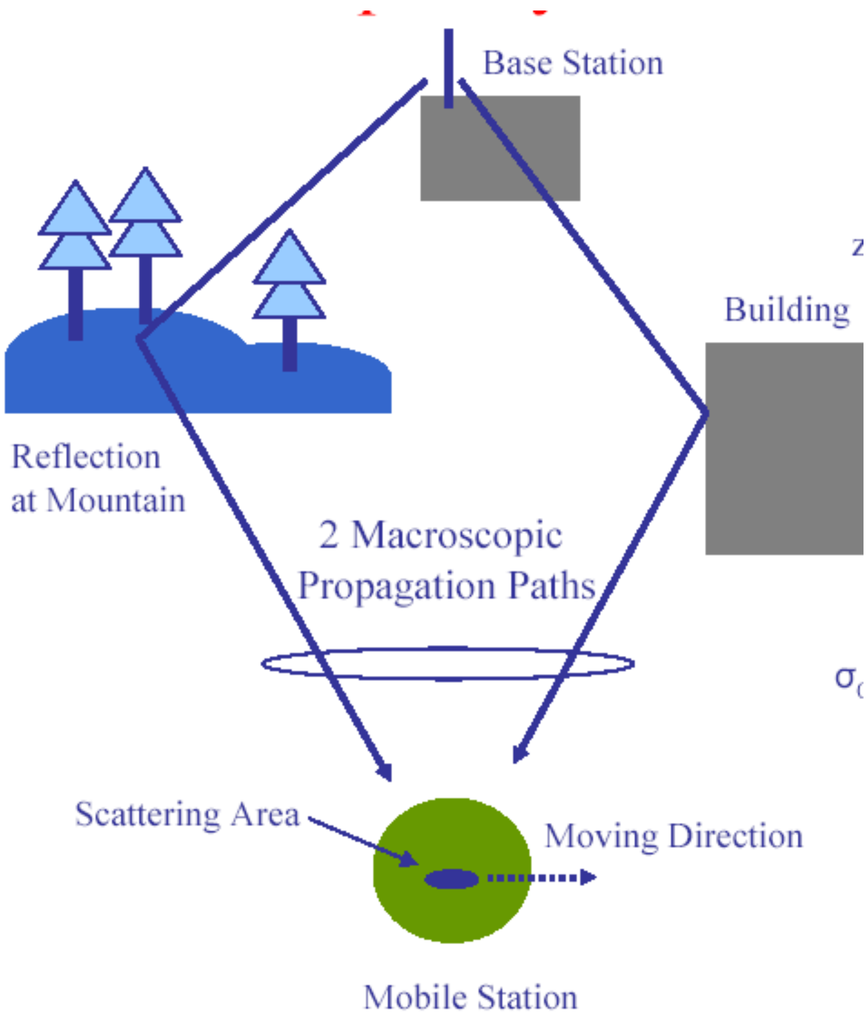
- Channel is band limited in nature

  Physics – e.g. parasitic capacitance in twisted pairs

  – limited frequency response

  – → unlimited time response

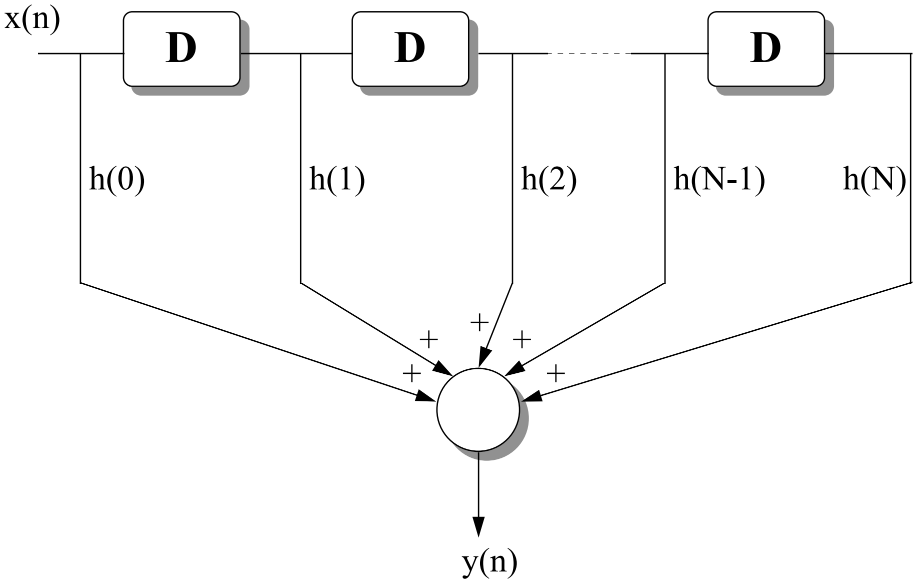
- Channel has multi-path reflections



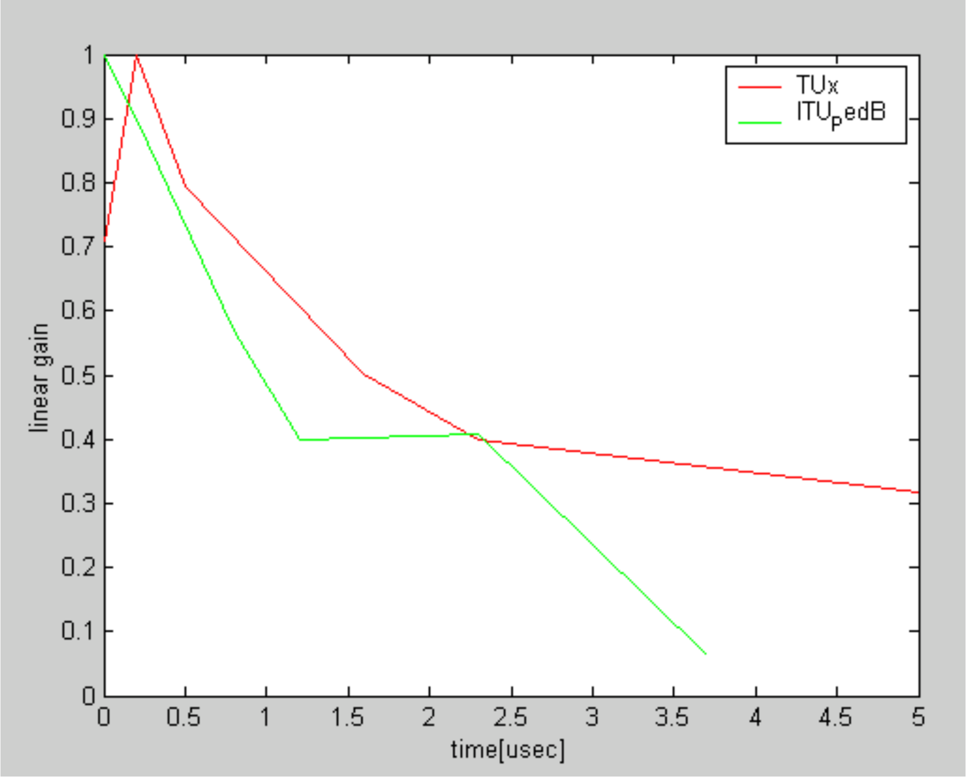
•Tx filter might add ISI when channel spacing is crucial.

Channel Model

- Channel is unknown
- Channel is usually modeled as Tap-Delay-Line (FIR)

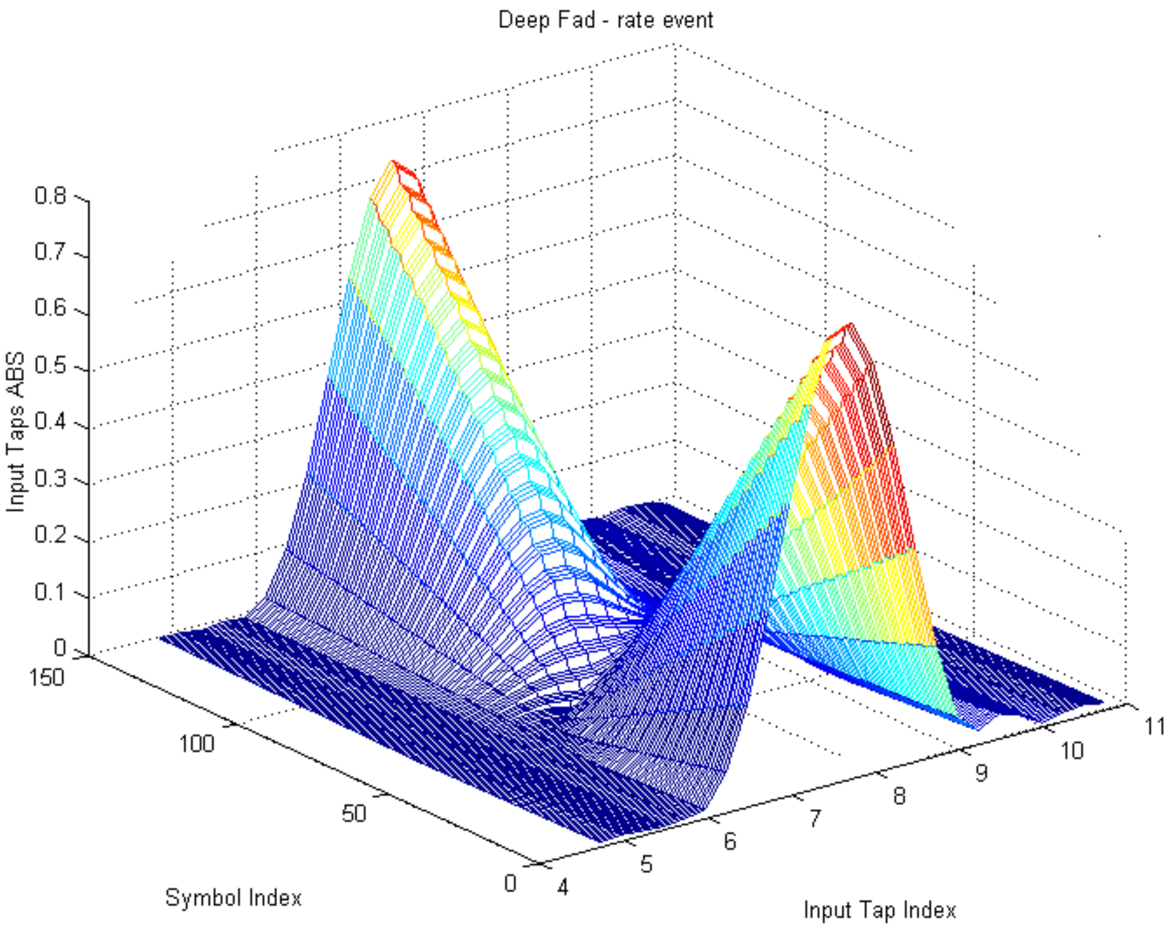


Example for Measured Channels

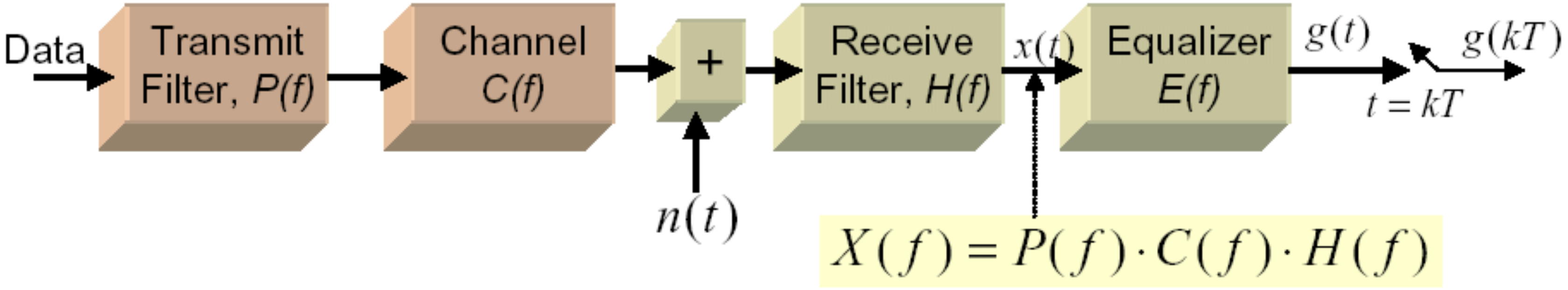


The Variation of the Amplitude of the Channel Taps is Random (changing Multipath)and usually modeled as Railegh distribution in Typical Urban Areas

Example for Channel Variation:



Equalizer: equalizes the channel – the received signal would seen like it passed a delta response.



$$|G_E(f)| = \frac{1}{|G_C(f)|} \Rightarrow |G_E(f)| \cdot |G_C(f)| = 1 \Rightarrow h_{total}(t) = \delta(t)$$

$$\arg(G_E(f)) = -\arg(G_C(f))$$

Need For Equalization

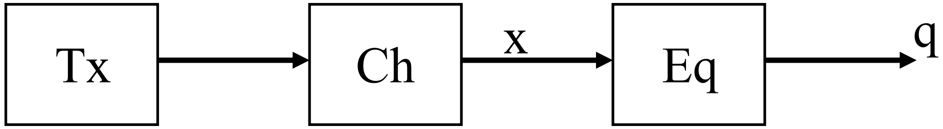
- Need For Equalization:
  - Overcome ISI degradation
- Need For Adaptive Equalization:
  - Changing Channel in Time

- => Objective:

Find the Inverse of the Channel Response to reflect a ‘delta channel to the Rx

* Applications (or standards recommend us the channel types for the receiver to cope with).

Zero forcing equalizers (according to Peak Distortion Criterion)



$$q(mT) = \sum_{n=-2}^{2} Cn \cdot X(mT - n \cdot \tau) = \begin{cases} 1, m = 0 \\ 0, m = \pm 1, \pm 2 \ldots \end{cases}$$

No ISI :Force

Equalizer taps

Example: 5tap-Equalizer, 2/T sample rate:

$x(mT - nT/2)$ is described as matrix

$$X = \begin{bmatrix} x(0) & x(-0.5T) & x(-1T) & x(-1.5T) & x(-2T) \\ x(1T) & x(0.5T) & x(0) & x(-0.5T) & x(-1T) \\ x(2T) & x(1.5T) & x(1T) & x(0.5T) & x(0) \\ x(3T) & x(2.5T) & x(2T) & x(1.5T) & x(1T) \end{bmatrix}$$

$$C = \begin{bmatrix} c_{-2} \\ c_{-1} \\ c_0 \\ c_1 \\ c_2 \end{bmatrix} \longrightarrow$$ Equalizer taps as vector

$$q = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \longrightarrow$$ Desired signal as vector

$$\Rightarrow \quad XC=q \quad \Rightarrow \quad C_{opt}=X^{-1}q$$

Disadvantages: Ignores presence of additive noise (noise enhancement)

MSE Criterion

UnKnown Parameter
(Equalizer filter response)

Desired Signal

Received Signal

$$J[\theta] = \sum_{n=0}^{N-1} (x[n] - \theta h[n])^2$$

Mean Square Error between the received signal and the desired signal, filtered by the equalizer filter

LS Algorithm

LMS Algorithm

LS

- Least Square Method:
  - Unbiased estimator
  - Exhibits minimum variance (optimal)
  - No probabilistic assumptions (only signal model)
  - Presented by Guass (1795) in studies of planetary motions)

LS - Theory

1. $s[n] = \sum h[n-m]\theta[m]$

2. $s[n] = \theta H$

3. $J[\theta] = \sum_{n=0}^{N-1} (x[n] - \theta h[n])^2$:MSE

Derivative according to θ :

4. $\hat{\theta} = \frac{\sum_{n=0}^{N-1} x[n]h[n]}{\sum_{n=0}^{N-1} h^2[n]}$

Back-Up

The minimum LS error would be obtained by substituting 4 to 3:

$$J_{\min} = J[\theta] = \sum_{n=0}^{N-1} (x[n] - \hat{\theta}h[n])^2 = \sum_{n=0}^{N-1} (x[n] - \hat{\theta}h[n])(x[n] - \hat{\theta}h[n])$$

$$= \sum_{n=0}^{N-1} x[n](x[n] - \hat{\theta}h[n]) - \hat{\theta}\underbrace{\sum_{n=0}^{N-1} h[n](x[n] - \hat{\theta}h[n])}_{0(BySubstitutinh\,\hat{\theta})}$$

$$= \sum_{n=0}^{N-1} x^2[n] - \hat{\theta}\sum_{n=0}^{N-1} x[n]h[n]$$

$$=> J_{\min} = \sum_{n=0}^{N-1} x^2[n] - \frac{(\sum_{n=0}^{N-1} x[n]h[n])^2}{\sum_{n=0}^{N-1}]h^2[n]}$$

⇧ Energy Of Original Signal

⇧ Energy Of Fitted Signal

$x[n] = Signal + w[n]$ ⇨ If Noise Small enough (SNR large enough): Jmin~0

Finding the LS solution

$s[n] = H\theta$ (H: observation matrix (Nxp) and $s[n] = (s[0], s[1], ...s[N-1])^T$

$$J[\theta] = \sum_{n=0}^{N-1}(x[n] - \hat{\theta}h[n])^2 = \sum_{n=0}^{N-1}(x[n] - \hat{\theta}h[n])(x[n] - \hat{\theta}h[n])$$

$$= (x[n] - H\theta])^T (x[n] - H\theta])$$

$$J[\theta] = x^T x - x^T H\theta - \theta^T H^T x + \theta^T H^T H\theta$$

$$= x^T z - \underbrace{2x^T H\theta}_{scalar} + \theta^T H^T H\theta$$

$$\frac{\partial J(\theta)}{\partial \theta} = -\underbrace{2H^T x}_{scalar} + 2H^T H\theta$$

$$\boxed{\hat{\theta} = (H^T H)^{-1} H^T x}$$

LS : Pros & Cons

- **Advantages**:
  - Optimal approximation for the Channel- once calculated it could feed the Equalizer taps.

- **Disadvantages**:
  - heavy Processing (due to matrix inversion which by It self is a challenge)
  - Not adaptive (calculated every once in a while and is not good for fast varying channels

- **Adaptive** Equalizer is required when the Channel is time variant (changes in time) in order to adjust the equalizer filter tap Weights according to the instantaneous channel properties.

LEAST-MEAN-SQUARE ALGORITHM

Contents:

- Introduction - approximating steepest-descent algorithm
- Steepest descend method
- Least-mean-square algorithm
- LMS algorithm convergence stability
- Numerical example for channel equalization using LMS
- Summary

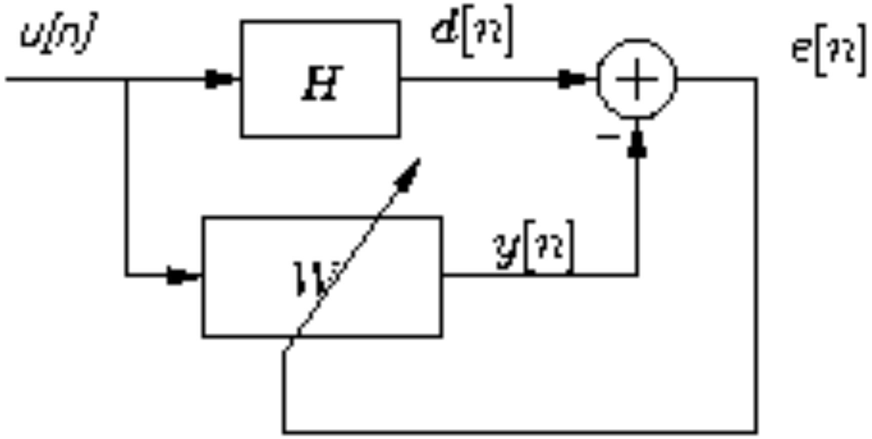
INTRODUCTION

- **Introduced by Widrow & Hoff in 1959**
- Simple, no matrices calculation involved in the adaptation
- In the family of stochastic gradient algorithms
- Approximation of the steepset – descent method
- Based on the MMSE criterion.(Minimum Mean square Error)
- Adaptive process containing two input signals:
- ✒✎✉ Filtering process, producing output signal.
- 2.) Desired signal (Training sequence)
- **Adaptive process: recursive adjustment of filter tap weights**

NOTATIONS

- Input signal (vector): **u(n)**
- Autocorrelation matrix of input signal: **Ruu = E[u(n)u^{H}(n)]**
- Desired response: **d(n)**
- Cross-correlation vector between **u(n) and d(n)**: **Pud = E[u(n)d*(n)]**
- Filter tap weights: **w(n)**
- Filter output: y(n) = **w^{H}(n)u(n)**
- Estimation error: ❋✈■✉ † ❄✈■✉ – ▮✈■✉
- ✯❈❁■ ✳❑◆❁❒❈ ✤❒❒❑❒✚ ✪ † ✤✱❟❋✈■✉❟^{2}✽ † **E[e(n)e*(n)]**

SYSTEM BLOCK USING THE LMS



U[n] = Input signal from the channel ; d[n] = Desired Response

H[n] = Some training sequence generator

e[n] = Error feedback between :

A.) desired response.

B.) Equalizer FIR filter output

W = Fir filter using tap weights vector

STEEPEST DESCENT METHOD

- Steepest decent algorithm is a gradient based method which employs recursive solution over problem (cost function)
- The current equalizer taps vector is W(n) and the next sample equalizer taps vector weight is W(n+1), We could estimate the W(n+1) vector by this approximation:

$$W[n] = W[n+1] + 0.5\mu(-\nabla J[n])$$

- The gradient is a vector pointing in the direction of the change in filter coefficients that will cause the greatest increase in the error signal. Because the goal is to minimize the error, however, the filter coefficients updated in the direction opposite the gradient; that is why the gradient term is negated.
- The constant μ is a step-size. After repeatedly adjusting each coefficient in the direction opposite to the gradient of the error, the adaptive filter should converge.

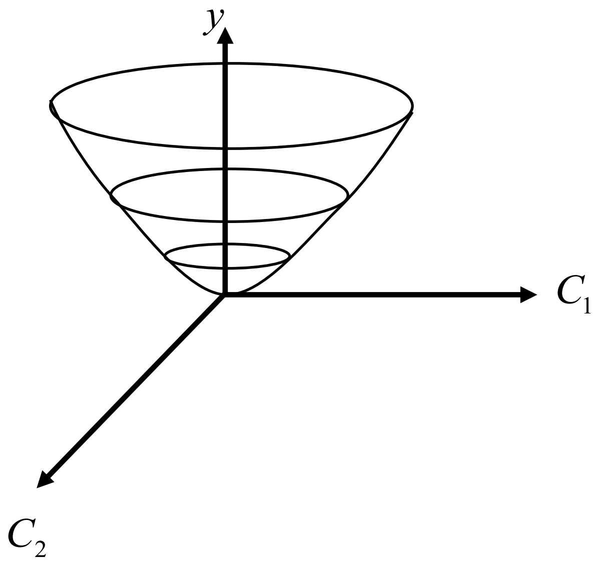
STEEPEST DESCENT EXAMPLE

- Given the following function we need to obtain the vector that would give us the absolute minimum.

$$Y(c_1, c_2) = C_1^2 + C_2^2$$

- It is obvious that $C_1 = C_2 = 0,$

give us the minimum.



Now lets find the solution by the steepest descend method

STEEPEST DESCENT EXAMPLE

- We start by assuming ($C_1 = 5, C_2 = 7$)
- We select the constant μ. If it is too big, we miss the minimum. If it is too small, it would take us a lot of time to het the minimum. I would select $\mu = 0.1$.
- The gradient vector is:

$$\nabla y = \begin{bmatrix} \frac{dy}{dc_1} \\ \frac{dy}{dc_2} \end{bmatrix} = \begin{bmatrix} 2C_1 \\ 2C_2 \end{bmatrix}$$

- So our iterative equation is:

$$\begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{[n+1]} = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{[n]} - 0.2 * \nabla y = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{[n]} - 0.1 \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{[n]} = 0.9 \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{[n]}$$

STEEPEST DESCENT EXAMPLE

$$Iteration1: \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

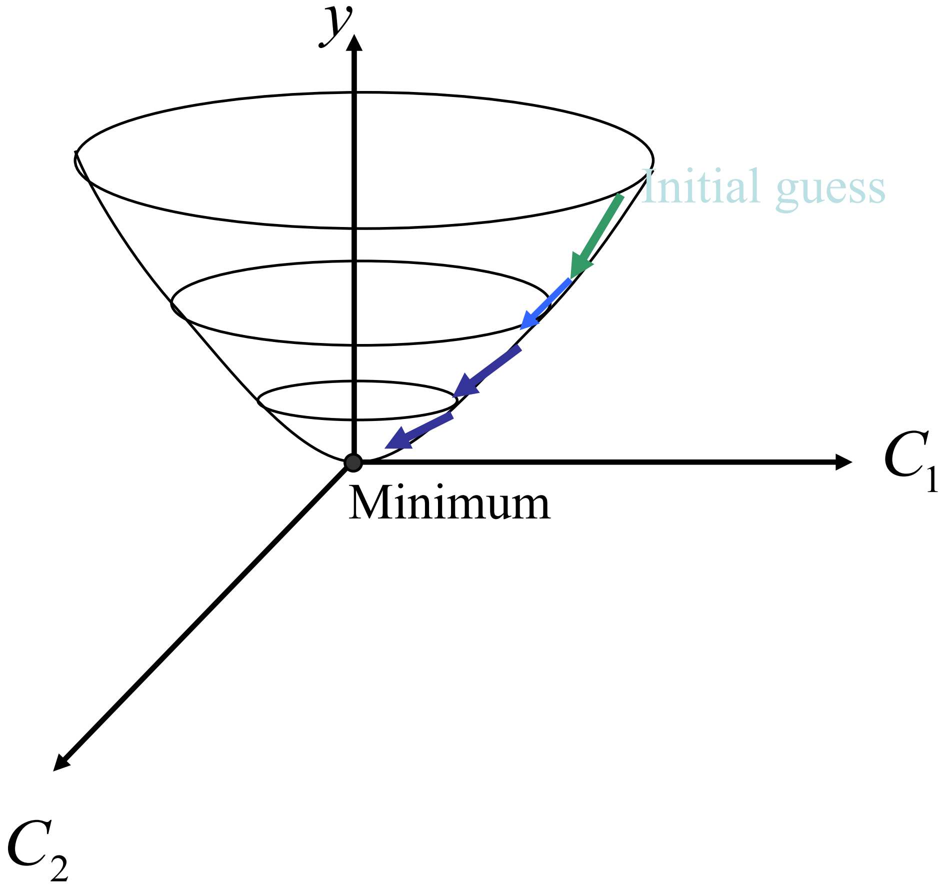
$$Iteration2: \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 4.5 \\ 6.3 \end{bmatrix}$$

$$Iteration3: \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0.405 \\ 0.567 \end{bmatrix}$$

......

$$Iteration\ 60: \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 0.01 \\ 0.013 \end{bmatrix}$$

$$\lim_{n\to\infty} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}_{[n]} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$



As we can see, the vector [c1,c2] convergates to the value which would yield the function minimum and the speed of this convergence depends on μ.

MMSE CRITERIA FOR THE LMS

- MMSE – Minimum mean square error
- MSE = $E\{[(d(k)-y(k)]^2\} = E\{[(d(k)-\sum_{n=-N}^{N} w(n)u(k-n)]^2\}$

$$E\{[(d(k)-\sum_{n=-N}^{N} w(n)u(k-n)]^2\} = E\{d(k)^2\} - 2\sum_{n=-N}^{N} w(n)P_{du}(n) + \sum_{n=-N}^{N}\sum_{m=-N}^{N} w(n)w(m)R(n-m)$$

$$P_{du}(n) = E\{d(k)u(n-k)\}$$

$$R_{uu}(n-m) = E\{u(m-k)u(n-k)\}$$

- To obtain the LMS MMSE we should derivative the MSE and compare it to 0:
- $$\frac{d(MSE)}{dW(k)} = \frac{d(E\{d(k)^2\} - 2\sum_{n=-N}^{N} w(n)P_{du}(n) + \sum_{n=-N}^{N}\sum_{m=-N}^{N} w(n)w(m)R(n-m))}{dW(k)}$$

MMSE CRITERION FOR THE LMS

And finally we get:

$$\nabla J(n) = \frac{d(MSE)}{dW(k)} = -2P_{du}(k) + 2\sum_{n--N}^{N} w[n]R_{uu}(n-k), k = 0, \pm 1, \pm 2, ...$$

By comparing the derivative to zero we get the MMSE:

$$w_{opt} = R^{-1} \bullet P$$

This calculation is complicated for the DSP (calculating the inverse matrix), and can cause the system to not being stable cause if there are NULLs in the noise, we could get very large values in the inverse matrix. Also we could not always know the Auto correlation matrix of the input and the cross-correlation vector, so we would like to make an approximation of this.

LMS – APPROXIMATION OF THE STEEPEST DESCENT METHOD

W(n+1) = W(n) + 2*[P – Rw(n)] <= According the MMSE criterion

We assume the following assumptions:

- Input vectors :$u(n), u(n-1),\ldots,u(1)$ statistically independent vectors.
- Input vector $u(n)$ and desired response $d(n)$, are statistically independent of $d(n), d(n-1),\ldots,d(1)$
- Input vector $u(n)$ and desired response $d(n)$ are *Gaussian-distributed R.V.*
- Environment is wide-sense stationary;

In LMS, the following estimates are used:

Ruu^ = $u(n)u^{H}(n)$ – Autocorrelation matrix of input signal

Pud^ = u(n)d*(n) - Cross-correlation vector between U[n] and d[n].

*** Or we could calculate the gradient of $|e[n]|^2$ instead of $E\{|e[n]|^2\}$

LMS ALGORITHM

$$W[n+1] \cong W[n] + \mu\{P^{\wedge} - R^{\wedge} w[n]\}$$

$$= w(n) + \mu\{u[n]d^{*}[n] - u[n]u^{H}[n]w[n]\}$$

$$= w(n) + \mu\{u[n]\{d^{*}[n] - y^{*}[n]\}$$

We get the final result:

$$W[n+1] \cong W[n] + \mu\{u[n]e^{*}[n]\}$$

LMS STABILITY

The size of the step size determines the algorithm convergence rate. Too small step size will make the algorithm take a lot of iterations. Too big step size will not convergence the weight taps.

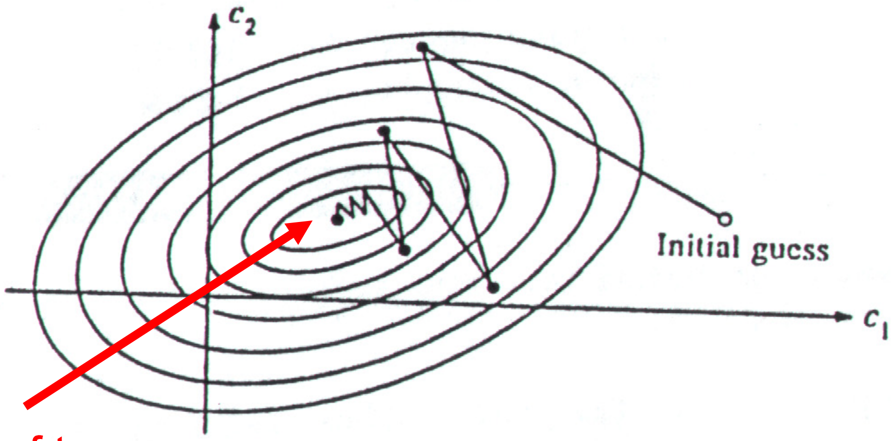
Rule Of Thumb:

$$\mu = \frac{1}{5(2N+1)P_R}$$

Where, N is the equalizer length
Pr, is the received power (signal+noise)
that could be estimated in the receiver.

LMS – CONVERGENCE GRAPH

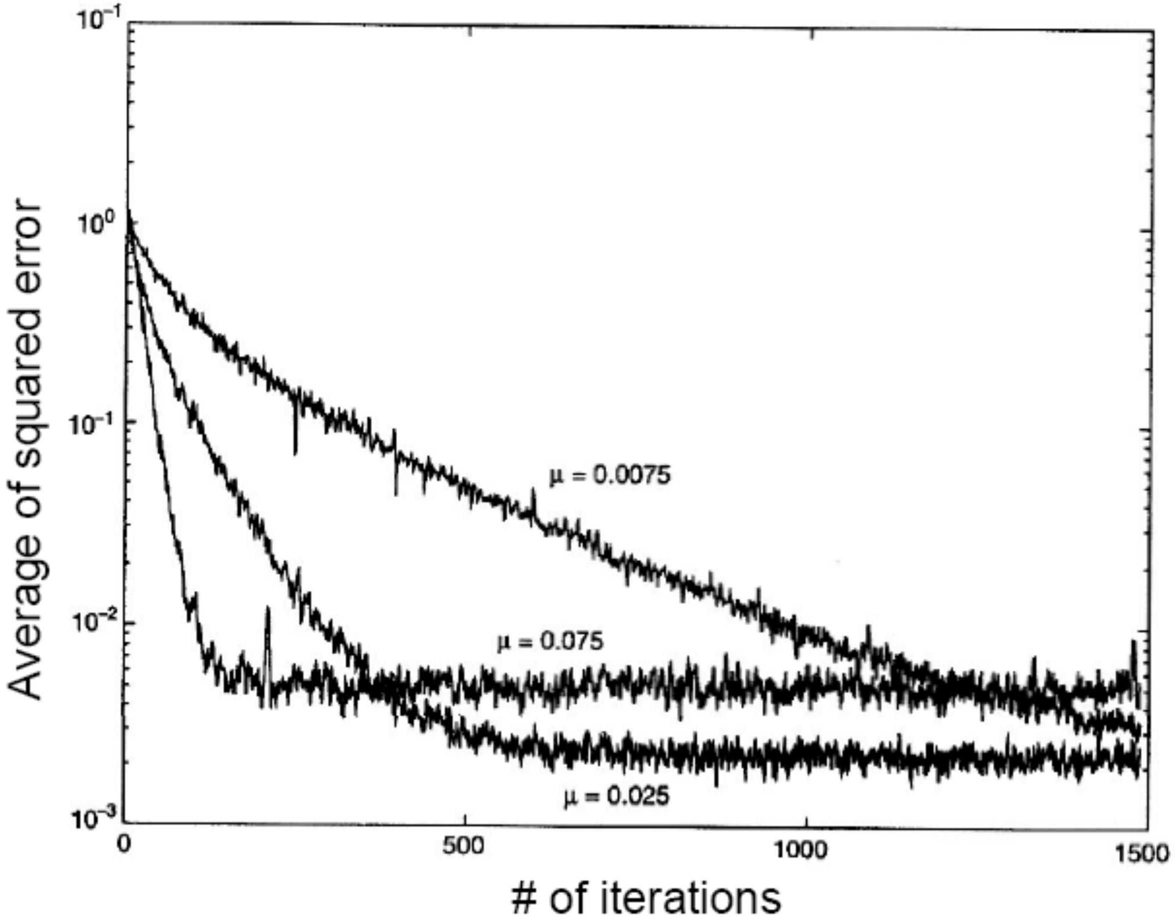
Example for the Unknown Channel of 2nd order:



Desired Combination of taps

This graph illustrates the LMS algorithm. First we start from guessing the TAP weights. Then we start going in opposite the gradient vector, to calculate the next taps, and so on, until we get the MMSE, meaning the MSE is 0 or a very close value to it.(In practice we can not get exactly error of 0 because the noise is a random process, we could only decrease the error below a desired minimum)

LMS Convergence Vs u



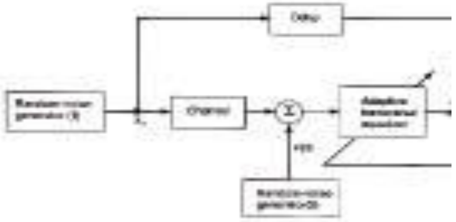
LMS – EQUALIZER EXAMPLE

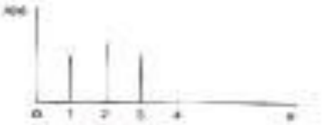
Channel equalization example:

- Transmitted signal: random sequence of ±1's.
- The transmitted signal is corrupted by a channel.
- Channel impulse response:

$$h_n = \begin{cases} \frac{1}{2}\left[1+\cos\left(\frac{2\pi}{W}(n-2)\right)\right], & n=1,2,3 \\ 0, & \text{otherwise} \end{cases}$$

- To the output of channel, white Gaussian noise with $\sigma_v^2 = 0.001$ is added.
- The received signal is processed by a linear, 11-tap FIR equalizer adapted with the LMS algorithm

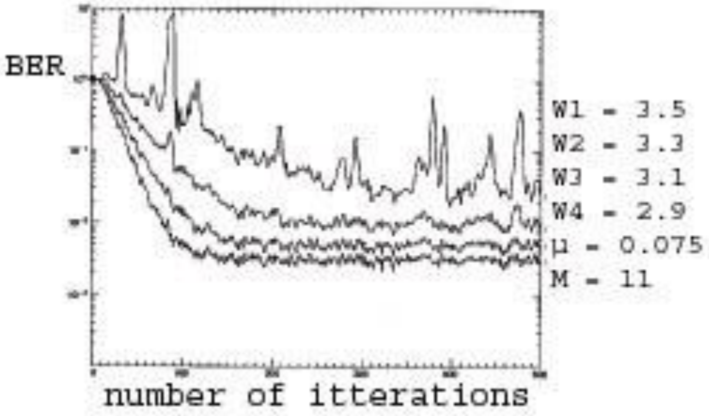


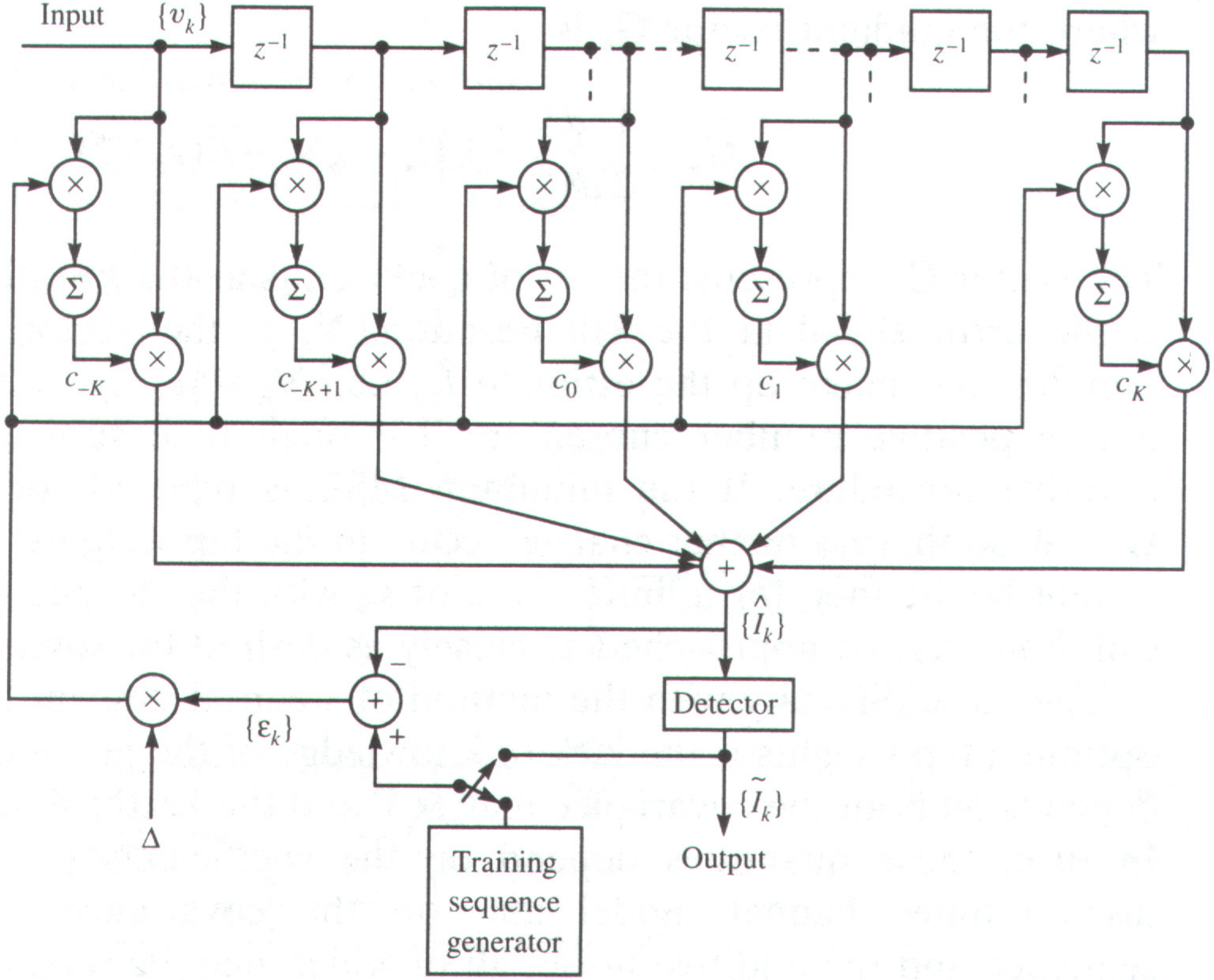


Channel response

Average Square Error as a function of iterations number using different channel transfer function

(change of W)




Input
$\{v_k\}$
z^{-1}
z^{-1}
z^{-1}
z^{-1}
z^{-1}
c_{-K}
c_{-K+1}
c_0
c_1
c_K
$\{\hat{I}_k\}$
−
+
Detector
$\{\varepsilon_k\}$
Δ
$\{\tilde{I}_k\}$
Output
Training sequence generator

LMS : Pros & Cons

LMS – Advantage:

- **Simplicity of implementation**
- **Not neglecting the noise like Zero forcing equalizer**
- **By pass the need for calculating an inverse matrix.**

LMS – Disadvantage:

Slow Convergence

Demands using of training sequence as reference ,thus decreasing the communication BW.

Non linear equalization

Linear equalization (reminder):

- *Tap delayed equalization*
- *Output is linear combination of the equalizer input*

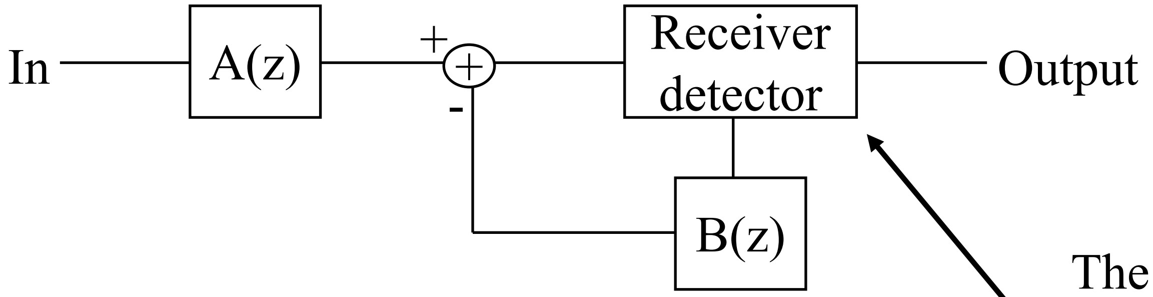
$$G_E = \frac{1}{G_C}$$

$$G_E = \prod_i (a_i - z^{-1})$$ → *as FIR*

$$\frac{Y(z)}{X(z)} = C_E = a_0 + a_1 z^{-1} + a_3 z^{-2} + \ldots$$

$$y(n) = a_0 \cdot x(n) + a_1 \cdot x(n-1) + a_2 \cdot x(n-2) + \ldots$$

Non linear equalization – DFE (Decision feedback Equalization)



The nonlinearity is due the detector characteristics that is fed back (MAPPER)

$$y(n) = \sum a_i \cdot x(n-i) - \sum b_i \cdot y(n-i)$$

$$\frac{Y(z)}{X(z)} = G_E = \frac{\prod_i (a_i - z^{-1})}{\prod_i (b_i - z^{-1})}$$

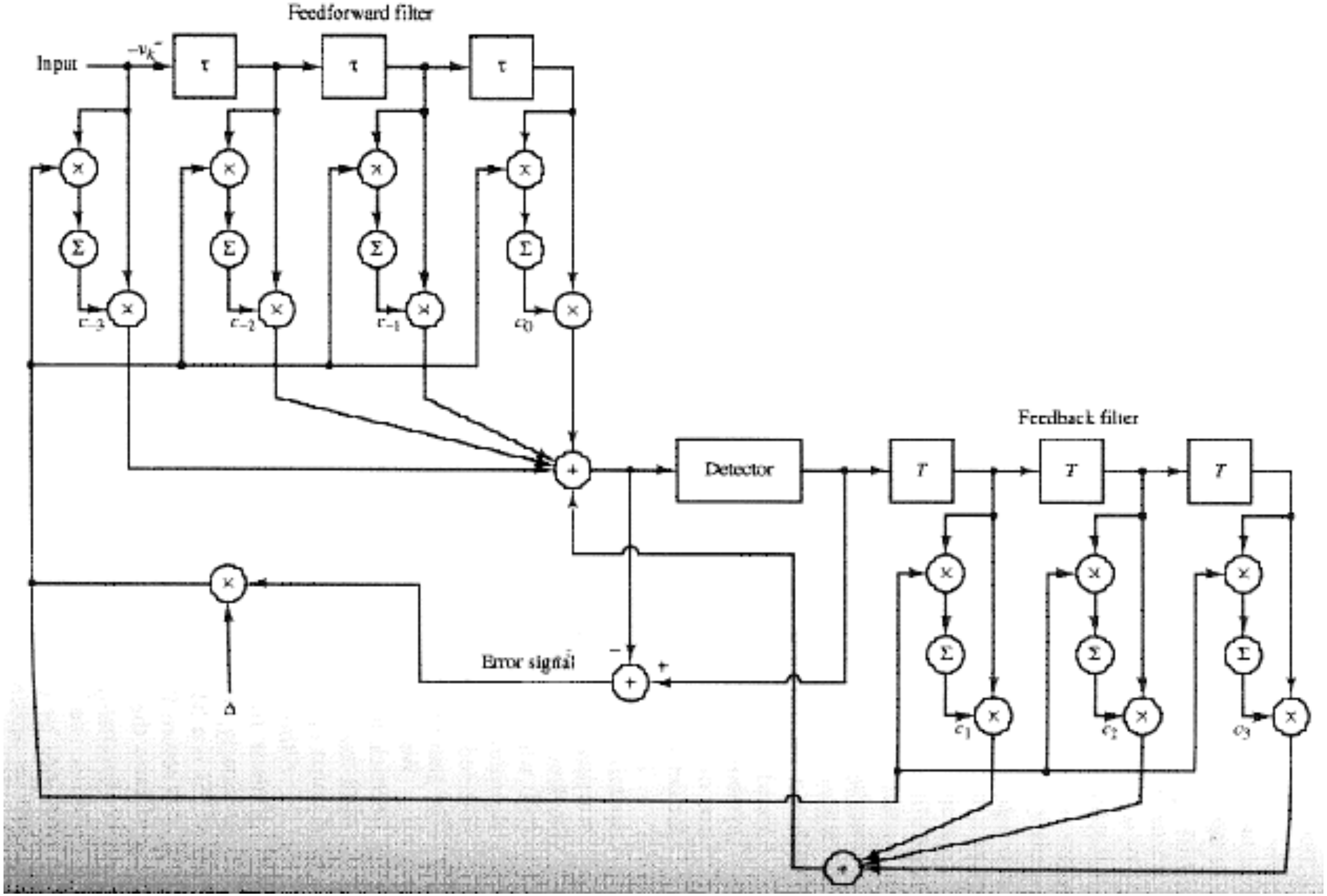
→ as IIR

The Decision feedback leads poles in z domain

Advantages: copes with larger ISI

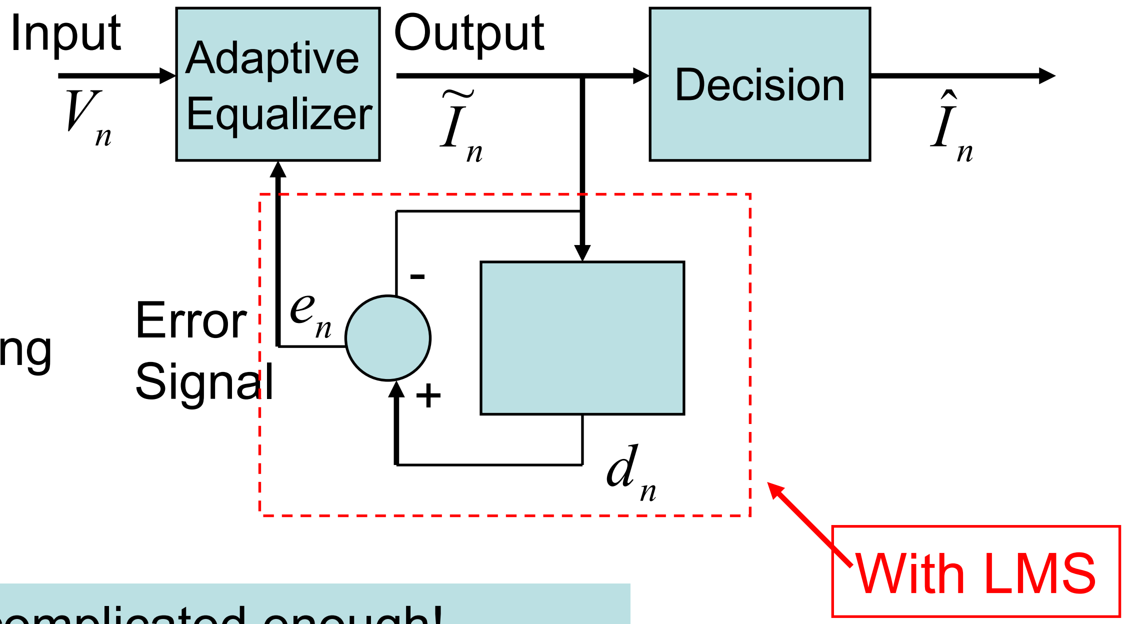
Disadvantages: instability danger

Non linear equalization - DFE



Blind Equalization

- ZFE and MSE equalizers assume option of training sequence for learning the channel.
- What happens when there is none?
  - **Blind Equalization**



But Usually employs also :

Interleaving\DeInterleaving

Advanced coding

ML criterion

Why? Blind Eq is hard and complicated enough!
So if you are going to implement it, use the best blocks
For decision (detection) and equalizing

Turbo Equalization

Iterative :

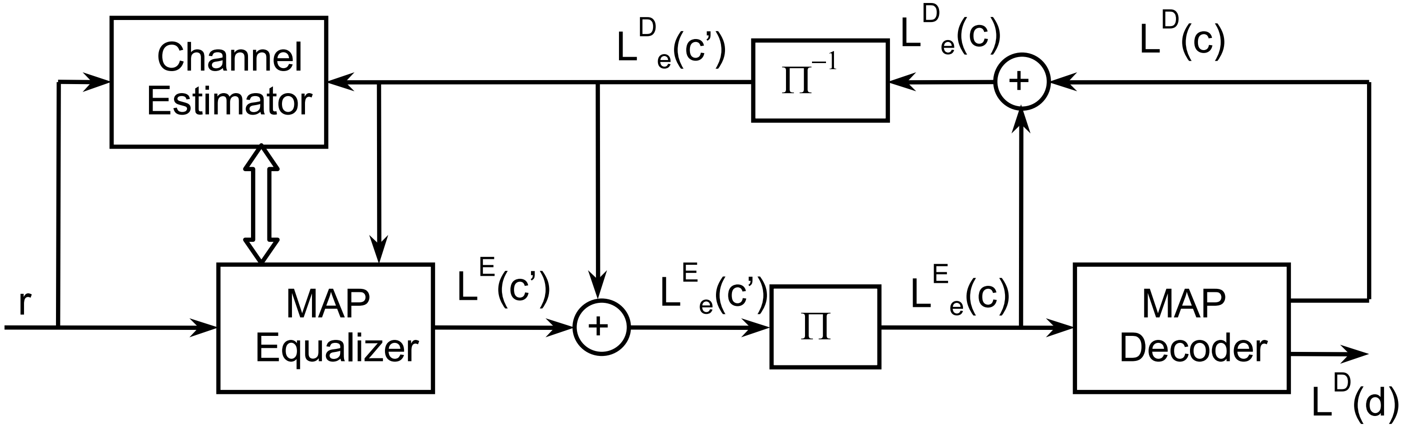
Estimate
Equalize
Decode
ReEncode

Next iteration would rely on better estimation therefore would lead more precise equalization

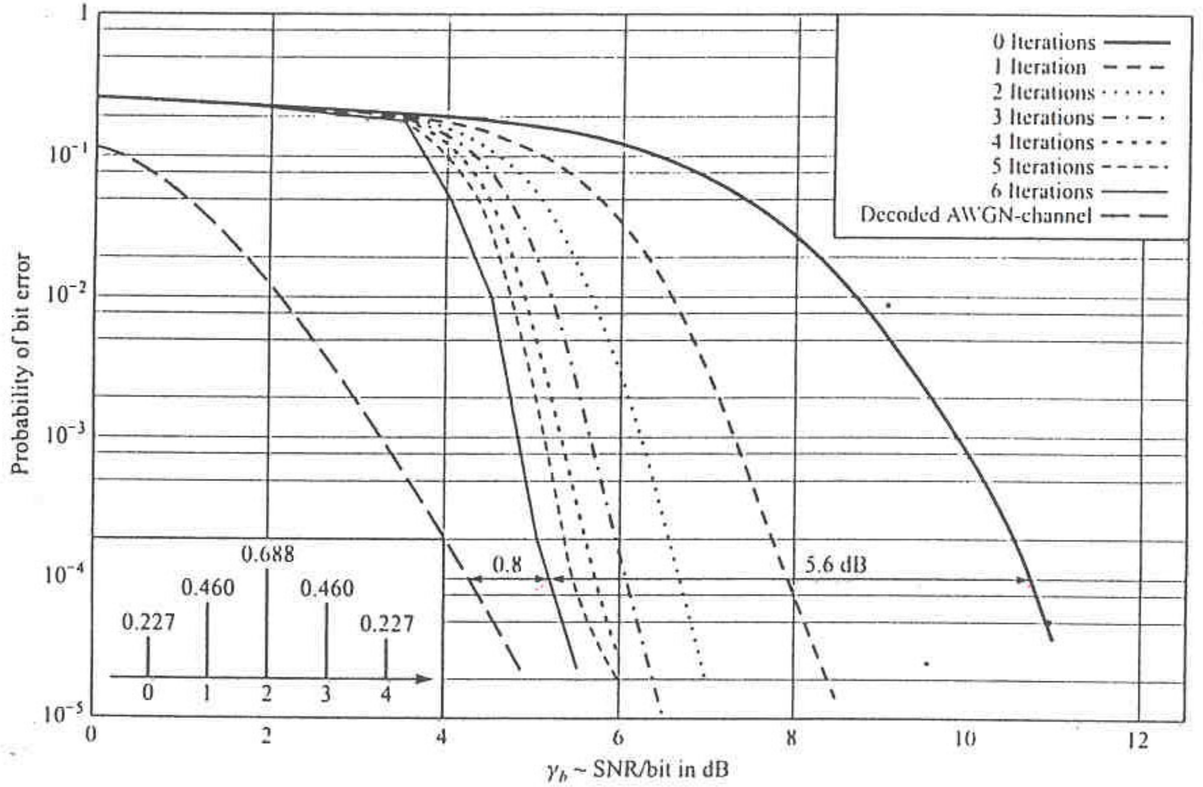
Usually employs also :

Interleaving\DeInterleaving
TurboCoding (Advanced iterative code)
MAP (based on ML criterion)

Why? It is complicated enough!
So if you are going to implement it, use the best blocks



Performance of Turbo Eq Vs Iterations



ML criterion

- MSE optimizes detection up to 1st/2nd order statistics.
- In Uri's Class:
  - Optimum Detection:
    - Strongest Survivor
    - Correlation (MF)

    (allow optimal performance for Delta ch and Additive noise.

    → Optimized Detection maximizes prob of detection (minimizes error or Euclidean distance in Signal Space)
- Lets find the Optimal Detection Criterion while in presence of memory channel (ISI)

ML criterion –Cont.

- Maximum Likelihood :

Maximizes decision probability for the received trellis

Example BPSK (NRZI)

$$S_1 = -S_0 = \sqrt{E_b}$$

Energy Per Bit

2 possible transmitted signals

$$r_k = \pm\sqrt{E_b} + n_k$$

Received Signal occupies AWGN

Conditional PDF (prob of correct decision on r1 pending s1 was transmitted…)

$$p(r_k \mid s_1) = \frac{1}{\sqrt{2\pi\sigma_n}} \exp\left[-\frac{(r_k - \sqrt{E_b})^2}{2\sigma_n{}^2} \right]$$

N0/2

$$p(r_k \mid s_0) = \frac{1}{\sqrt{2\pi\sigma_n}} \exp\left[-\frac{(r_k + \sqrt{E_b})^2}{2\sigma_n{}^2} \right]$$

Prob of correct decision on a $\underbrace{\text{sequence of symbols}}$

optimal

$$p(r_1, r_2, \ldots, r_k \mid s^{(m)}) = \prod_{k=1}^{K} p(r_k \mid s_k{}^{(m)})$$

Transmitted sequence

ML – Cont.

With logarithm operation, it could be shown that this is equivalent to :

Minimizing the Euclidean distance metric of the sequence:

$$D(r, s^{(m)}) = \sum_{k=1}^{K} (r_k - s_k{}^{(m)})^2$$

→ (Called Metric)

Looks Similar?

while MSE minimizes Error (maximizes Prob) for decision on certain Sym,
MLSE minimizes Error (maximizes Prob) for decision on certain Trellis ofSym,

How could this be used?

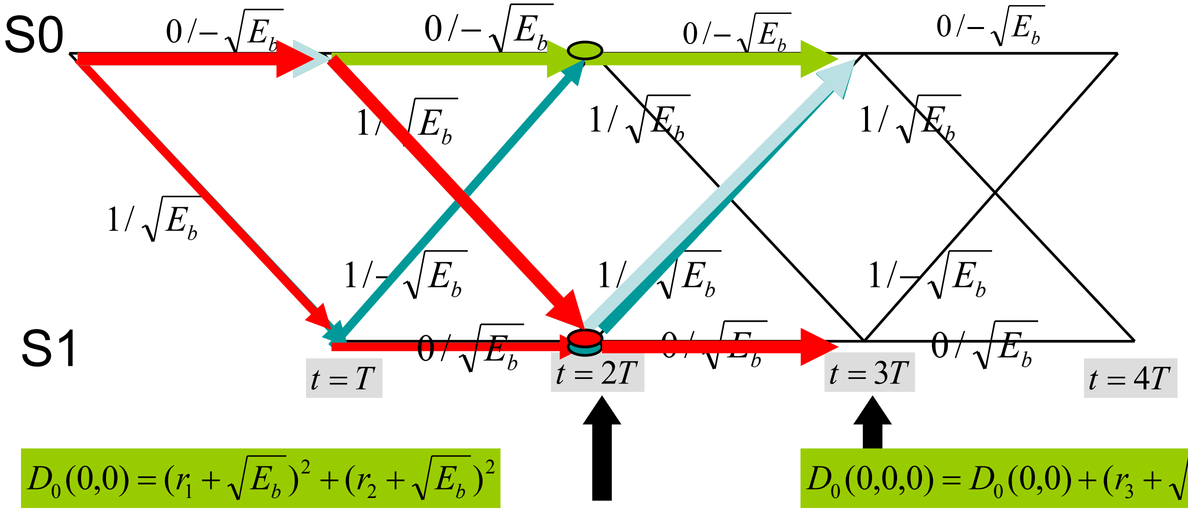
Viterbi Equalizer
(On the tip of the tongue)

Example for NRZI:

Trasmit Symbols: $\sqrt{E_b}$ $-\sqrt{E_b}$

(0=No change in transmitted Symbol

(1=Alter Symbol)



Metric (Sum of Euclidean Distance)

$D_0(0,0)=(r_1+\sqrt{E_b})^2+(r_2+\sqrt{E_b})^2$

$D_0(1,1)=(r_1-\sqrt{E_b})^2+(r_2+\sqrt{E_b})^2$

$D_0(0,1)=(r_1+\sqrt{E_b})^2+(r_2-\sqrt{E_b})^2$

$D_0(1,0)=(r_1-\sqrt{E_b})^2+(r_2-\sqrt{E_b})^2$

$D_0(0,0,0)=D_0(0,0)+(r_3+\sqrt{E_b})^2$

$D_0(0,1,1)=D_0(0,1)+(r_3+\sqrt{E_b})^2$

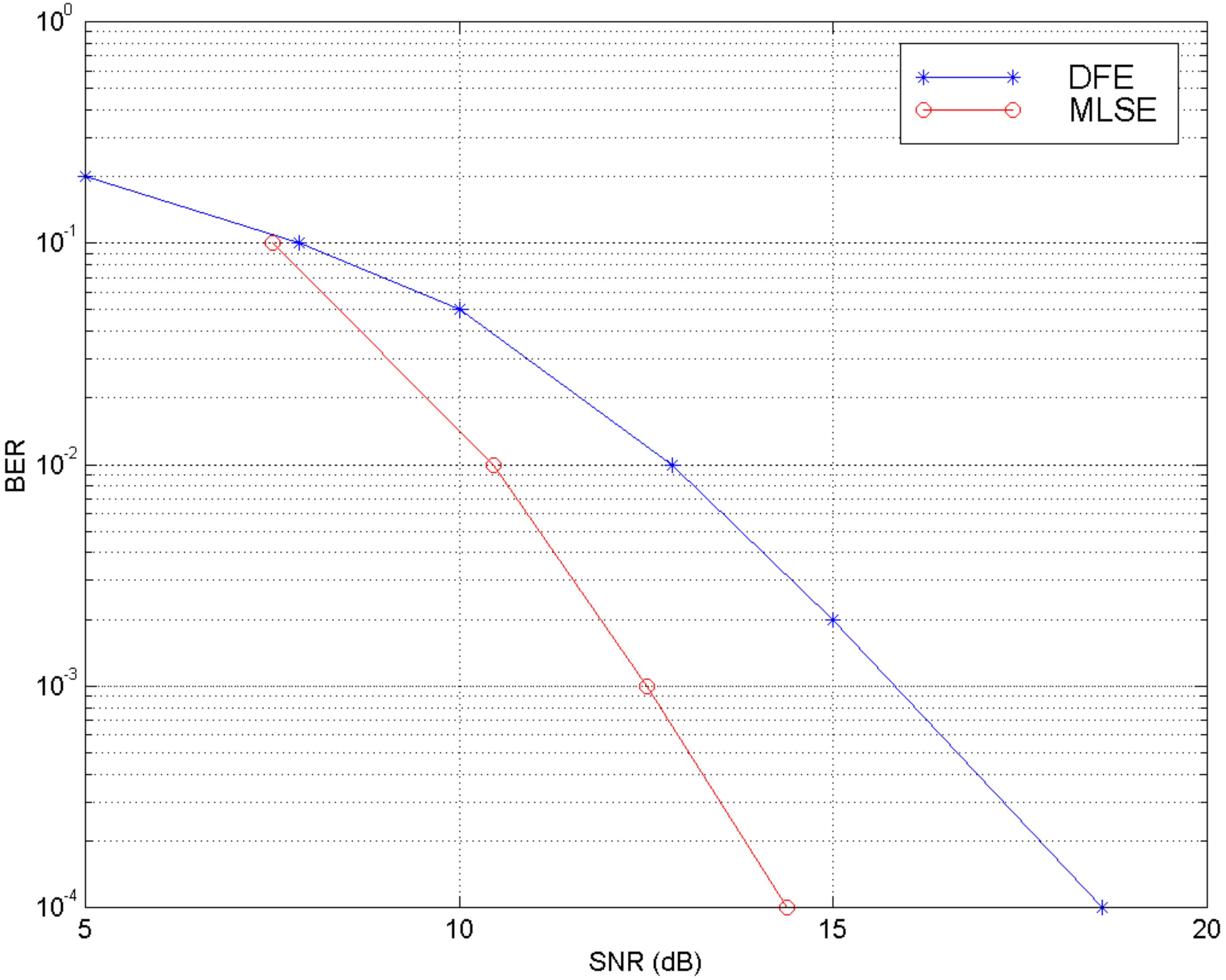
$D_0(0,0,1)=D_0(0,0)+(r_3-\sqrt{E_b})^2$

$D_0(0,1,0)=D_0(0,1)+(r_3-\sqrt{E_b})^2$

We Always disqualify one metric for possible S0 and possible S1.

Finally we are left with 2 options for possible Trellis.

Finally are decide on the correct Trellis with the Euclidean
 Metric of each or with Apost DATA


DFE
MLSE
BER
SNR (dB)
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
5
10
15
20

References

- John G.Proakis – Digital Communications.
- John G.Proakis –Communication Systems Eng.
- Simon Haykin - Adaptive Filter Theory
- K Hooli – Adaptive filters and LMS
- S.Kay – Statistical Signal Processing – Estimation Theory