Recursive Least Squares Estimation

Overview

- Recursive Least squares estimation;
  - The exponentially weighted Least squares
  - Recursive-in-time solution
  - Initialization of the algorithm
  - Recursion for MSE criterion
- Examples: Noise canceller, Channel equalization, Echo cancellation

Reference : Chapter 9 from *S. Haykin- Adaptive Filtering Theory - Prentice Hall, 2002.*

Recursive Least Squares Estimation

Problem statement

- Given the set of input samples $\{u(1), u(2), \ldots, u(N)\}$ and the set of desired response $\{d(1), d(2), \ldots, d(N)\}$
- In the family of linear filters computing their output according to

$$y(n) = \sum_{k=0}^{M} w_k u(n-k), \quad n = 0, 1, 2, \ldots \tag{1}$$

- Find *recursively in time* the parameters $\{w_0(n), w_1(n), \ldots, w_{M-1}(n)\}$ such as to minimize the sum of error squares

$$\mathcal{E}(n) = \mathcal{E}(w_0(n), w_1(n), \ldots, w_{M-1}(n)) = \sum_{i=i_1}^{n} \beta(n,i)[e(i)^2] = \sum_{i=i_1}^{n} \beta(n,i)[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2$$

where the error signal is

$$e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)$$

and the forgetting factor or weighting factor reduces the influence of old data

$$0 < \beta(n,i) \leq 1, \qquad i = 1, 2, \ldots, n$$

usually taking the form $(0 < \lambda < 1)$

$$\beta(n,i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n$$

□

The exponentially weighted Least squares solution Writing the criterion with an exponential forgetting factor

$$\mathcal{E}(n) = \mathcal{E}(w_0(n), w_1(n), \ldots, w_{M-1}(n)) = \sum_{i=i_1}^{n} \lambda^{n-i}[e(i)^2] = \sum_{i=i_1}^{n} \lambda^{n-i}[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2$$

Make the following variable changes:

$$u'(i) = \sqrt{\lambda^{n-i}}u(i); \qquad d'(i) = \sqrt{\lambda^{n-i}}d(i) \tag{2}$$

Then the criterion rewrites

$$\mathcal{E}(n) = \sum_{i=i_1}^{n} \lambda^{n-i}[d(i) - \sum_{k=0}^{M-1} w_k(n)u(i-k)]^2 = \sum_{i=i_1}^{n} [d'(i) - \sum_{k=0}^{M-1} w_k(n)u'(i-k)]^2$$

which is the standard LS criterion, in the new variables $u'(i), d'(i)$. The LS solution can be obtained as

$$\underline{w}(n) = (\sum_{i=i_1}^{n} \underline{u}'(i)\underline{u}'(i)^T)^{-1} \sum_{i=i_1}^{n} \underline{u}'(i)d'(i) = (\sum_{i=i_1}^{n} \lambda^{n-i}\underline{u}(i)\underline{u}(i)^T)^{-1} \sum_{i=i_1}^{n} \lambda^{n-i}\underline{u}(i)d(i) = [\Phi(n)]^{-1}\underline{\psi}(n) =$$

where we will denote (making use of Pre-windowing assumption, that data before $i = 1$ is zero)

$$\Phi(n) = \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)\underline{u}(i)^T$$

$$\underline{\psi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)d(i)$$

Recursive in time solution

We want to find a recursive in time way to compute

$$\underline{w}(n) = [\Phi(n)]^{-1}\underline{\psi}(n)$$

using the information already available at time $n-1$, i.e.

$$\underline{w}(n-1) = [\Phi(n-1)]^{-1}\underline{\psi}(n-1)$$

We therefore will rewrite the variables $\Phi(n)$ and $\underline{\psi}(n)$ as functions of $\Phi(n-1)$ and $\underline{\psi}(n-1)$

$$\begin{aligned}
\Phi(n) &= \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)\underline{u}(i)^T = \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i}\underline{u}(i)\underline{u}(i)^T + \underline{u}(n)\underline{u}(n)^T = \lambda\Phi(n-1) + \underline{u}(n)\underline{u}(n)^T \\
\underline{\psi}(n) &= \sum_{i=1}^{n} \lambda^{n-i}\underline{u}(i)d(i) = \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i}\underline{u}(i)d(i) + \underline{u}(n)d(n) = \lambda\underline{\psi}(n-1) + \underline{u}(n)d(n)
\end{aligned}$$

The matrix inversion formula

If A and B are $M \times M$ positive definite matrices, D is a $N \times N$ matrix, and C is a $M \times N$ matrix which are related by

$$A = B^{-1} + CD^{-1}C^T$$

then

$$A^{-1} = B + BC(D + C^T BC)^{-1}C^T B$$

Proof Exercise.

Derivation of the algorithm

Applying the matrix inversion formula to

$$\Phi(n) \;=\; \lambda\Phi(n-1) + \underline{u}(n)\underline{u}(n)^T$$

we obtain

$$\Phi^{-1}(n) \;=\; \lambda^{-1}\Phi^{-1}(n-1) - \frac{\lambda^{-2}\Phi^{-1}(n-1)\underline{u}(n)\underline{u}^T(n)\Phi^{-1}(n-1)}{1+\lambda^{-1}\underline{u}^T(n)\Phi^{-1}(n-1)\underline{u}(n)}$$

Denoting

$$P(n) = \Phi^{-1}(n)$$

and

$$\underline{k}(n) \;=\; \frac{\lambda^{-1}P(n-1)\underline{u}(n)}{1+\lambda^{-1}\underline{u}^T(n)P(n-1)\underline{u}(n)} = \text{Exercise} = P(n)\underline{u}(n)$$

we obtain

$$P(n) \;=\; \lambda^{-1}P(n-1) - \lambda^{-1}\underline{k}(n)\underline{u}^T(n)P(n-1)$$

We are now able to derive the main time-update equation, that of $\underline{w}(n)$

$$\begin{aligned}
\underline{w}(n) \;&=\; [\Phi(n)]^{-1}\underline{\psi}(n) = P(n)\underline{\psi}(n) = P(n)(\lambda\underline{\psi}(n-1) + \underline{u}(n)d(n)) = P(n)(\lambda\Phi(n-1)\underline{w}(n-1) + \underline{u}(n)d(n)) \\
&=\; P(n)((\Phi(n) - \underline{u}(n)\underline{u}(n)^T)\underline{w}(n-1) + \underline{u}(n)d(n)) = \underline{w}(n-1) - P(n)\underline{u}(n)\underline{u}(n)^T\underline{w}(n-1) + P(n)\underline{u}(n)d(n) \\
&=\; \underline{w}(n-1) + P(n)\underline{u}(n)(d(n) - \underline{u}(n)^T\underline{w}(n-1)) = \underline{w}(n-1) + P(n)\underline{u}(n)\alpha(n) = \underline{w}(n-1) + \underline{k}(n)\alpha(n)
\end{aligned}$$

where

$$\alpha(n) \;=\; d(n) - \underline{u}(n)^T \underline{w}(n-1)$$

is the innovation process (apriori errors). Now we can collect all necessary equations to form the RLS algorithm:

$$\begin{aligned}
\underline{k}(n) &= \frac{\lambda^{-1} P(n-1)\underline{u}(n)}{1 + \lambda^{-1}\underline{u}^T(n)P(n-1)\underline{u}(n)} \\
\alpha(n) &= d(n) - \underline{u}(n)^T \underline{w}(n-1) \\
\underline{w}(n) &= \underline{w}(n-1) + \underline{k}(n)\alpha(n) \\
P(n) &= \lambda^{-1}P(n-1) - \lambda^{-1}\underline{k}(n)\underline{u}^T(n)P(n-1)
\end{aligned}$$

Initialization of RLS algorithm

In RLS algorithm there are two variables involved in the recursions (those with time index $n-1$): $\underline{\hat{w}}(n-1)$, P_{n-1}. We must provide initial values for these variables in order to start the recursions :

- $\underline{w}(0)$

  If we have some apriori information about the parameters $\underline{\hat{w}}$ this information will be used to initialize the algorithm.

  Otherwise, the typical initialization is

$$\underline{w}(0) = 0$$

- P_0

  1. Recalling the significance of $P(n)$

$$P(n) = \Phi^{-1}(n) = [\sum_{i=i_1}^{n} \lambda^{n-i} \underline{u}(i)\underline{u}(i)^T]^{-1}$$

  the *exact initialization* of the recursions uses a small initial segment of the data $u(i_1), u(i_1+1), \ldots, u(0)$ to compute

$$P(0) = \Phi^{-1}(0) = [\sum_{i=i_1}^{0} \lambda^{-i} \underline{u}(i)\underline{u}(i)^T]^{-1}$$

  However, it is not a simple matter to select the length of data required for ensuring invertibility of $\Phi(0)$!

2. The *approximate initialization* is commonly used, it don't require matrix inversion:

$$P(0) = \delta I$$

There is an intuitive explanation of this initialization. The significance

$$P(n) = \Phi^{-1}(n) \approx const. \cdot E(\underline{w}(n) - \underline{\hat{w}})(\underline{w}(n) - \underline{\hat{w}})^T$$

can be proven. Thus, $P(n)$ is proportional to the covariance matrix of the parameters $\underline{w}(n)$. Since our knowledge of these parameters at $n = 0$ is very vague, a very high covariance matrix of the parameters is to be expected, and thus we must assign a high value to δ.

The recommended value for δ is

$$\delta > 100\sigma_u^2$$

For large data length, the initial values assigned at $n = 0$ are not important, since they are forgotten due to exponential forgetting factor λ.

Summary of the RLS algorithm

Given data $u(1), u(2), u(3), \ldots, u(N)$ and $d(1), d(2), d(3), \ldots, d(N)$

1. Initialize $\underline{w}(0) = 0, \quad P_0 = \delta I$
2. For each time instant, $n = 1, \ldots, N$, Compute

2.1 $\underline{\pi} = \underline{u}^T(n) P(n-1)$	M^2 *flops*
2.2 $\gamma = \lambda + \underline{\pi}\ \underline{u}$	M *flops*
2.3 $\underline{k}(n) = \frac{\underline{\pi}^T}{\gamma}$	M *flops*
2.4 $\alpha(n) = d(n) - \underline{w}^T(n-1)\underline{u}(n)$	M *flops*
2.5 $\underline{w}(n) = \underline{w}(n-1) + \underline{k}(n)\alpha(n)$	M *flops*
2.5 $P' = \underline{k}(n)\underline{\pi}$	M^2 *flops*
2.5 $P(n) = \frac{1}{\lambda}(P(n-1) - P')$	M^2 *flops*

We used flop as an abbreviation for one addition (subtraction) + one multiplication (floating point operations).

The overall complexity of the algorithm is $\mathcal{O}(M^2)$ operations (flops) per time iteration.

Recursion for MSE criterion

Suppose we have at moment $n-1$ the MSE criterion value

$$\mathcal{E}(\underline{w}(n-1)) \;=\; \sum_{i=i_1}^{n-1} \lambda^{n-1-i}(d(i)-\underline{w}(n-1)^T\underline{u}(i))^2$$

and we want to know what is the MSE of the new filter $\underline{w}(n)$

$$\mathcal{E}(\underline{w}(n)) \;=\; \sum_{i=i_1}^{n} \lambda^{n-i}(d(i)-\underline{w}(n)^T\underline{u}(i))^2 = \sum_{i=i_1}^{n}(d(i))^2 - \underline{w}(n)^T\psi(n)$$

One remarkable recursion holds:

$$\mathcal{E}(\underline{w}(n)) \;=\; \lambda\mathcal{E}(\underline{w}(n-1)) + \alpha(n)e(n)$$

where $\alpha(n)$ is the apriori error

$$\alpha(n) = d(n) - \underline{w}(n-1)^T\underline{u}(n)$$

and $e(n)$ is the aposteriori error

$$e(n) = d(n) - \underline{w}(n)^T\underline{u}(n)$$

Applications of Recursive LS filtering

1. Adaptive noise canceller

Single weight, dual-input adaptive noise canceller

The filter order is $M = 1$ thus the filter output is

$$y(n) = \underline{w}(n)^T \underline{u}(n) = w(n)u(n)$$

Denoting $P^{-1}(n) = \sigma^2(n)$, the Recursive Least Squares filtering algorithm can be rearranged as follows:

RLS

Given data

$u(1), u(2), u(3), \ldots, u(N)$ and
$d(1), d(2), d(3), \ldots, d(N)$

1. Initialize $w(0) = 0, \quad P_0 = \delta$
2. For each time instant, $n = 1, \ldots, N$
   - 2.1 $k(n) = \dfrac{1}{\lambda\sigma(n-1)^2 + u(n)^2} u(n)$
   - 2.2 $\alpha(n) = d(n) - w(n-1)u(n)$
   - 2.3 $w(n) = w(n-1) + \alpha(n)k(n)$
   - 2.4 $\sigma^2(n) = \lambda\sigma(n-1)^2 + u(n)^2$

Normalized LMS

Given data

$u(1), u(2), u(3), \ldots, u(N)$ and
$d(1), d(2), d(3), \ldots, d(N)$

1. Initialize $w(0) = 0$
2. For each time instant, $n = 1, \ldots, N$
   - 2.1 $k(n) = \dfrac{1}{a + u(n)^2} u(n)$
   - 2.2 $\alpha(n) = d(n) - w(n-1)u(n)$
   - 2.3 $w(n) = w(n-1) + \alpha(n)k(n)$

The normalized LMS algorithms can be obtained from RLS algorithm replacing the time varying term $\lambda\sigma(n-1)^2$ with the constant a.

2. Adaptive Equalization

Modelling the communication channel

We assume the impulse response of the channel in the form

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

The filter input signal will be

$$u(n) = (h * a)(n) = \sum_{k=1}^{3} h(k)a(n-k) + v(n)$$

where $\sigma_v^2 = 0.001$

Selecting the filter structure

The filter has $M = 11$ delays units (taps).

$$y(n) = \underline{w}(n)^T \underline{u}(n) = w_0(n)u(n) + w_1(n)u(n-1) + \ldots + w_{10}(n)u(n-10)$$

The channel input is delayed 7 units to provide the desired response to the equalizer.

$$d(n) = a(n-7)$$

Two recursive (adaptive) filtering algorithms are compared: Recursive Least Squares (RLS) and (LMS).

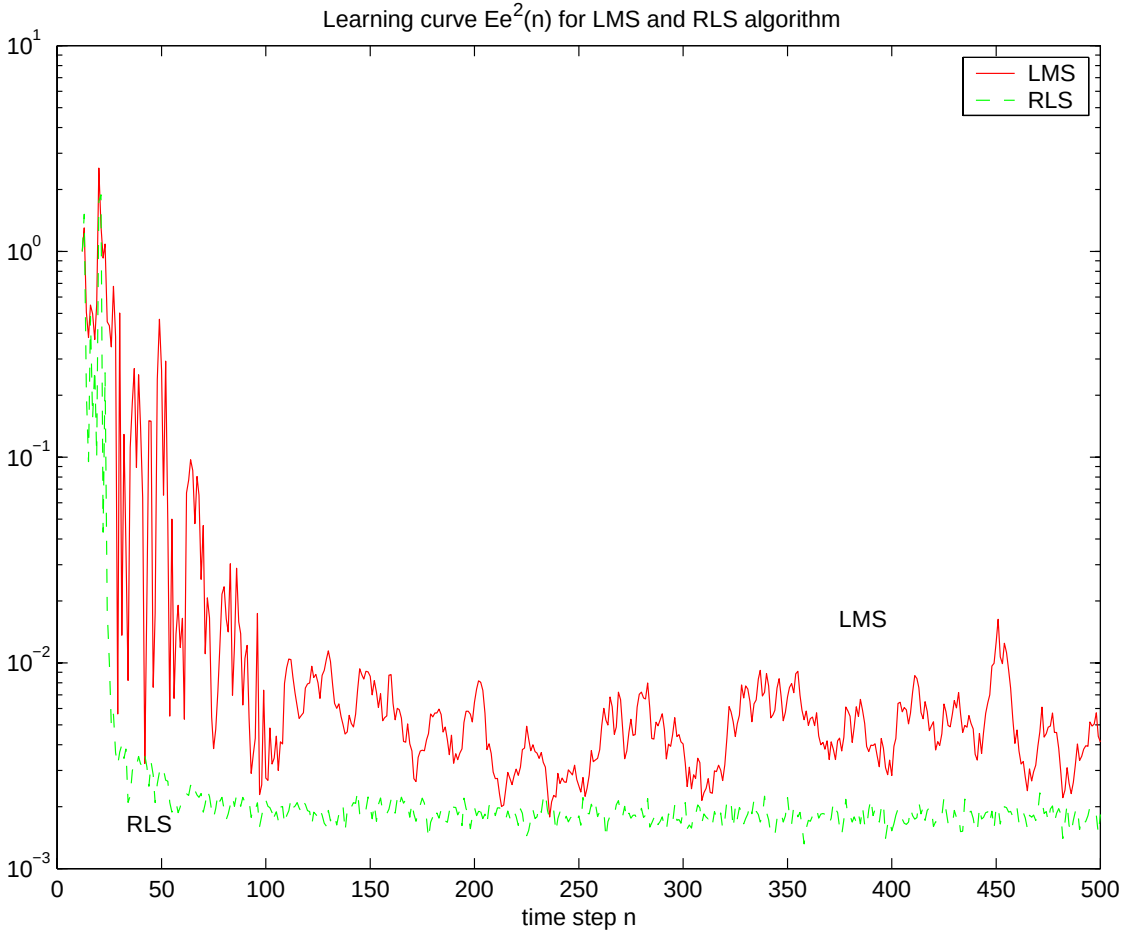
RLS algorithm has higher computational requirement than LMS , but behaves much better in terms of steady state MSE and transient time. For a picture of major differences between RLS and LMS, the main recursive equation are rewritten:

RLS algorithm

1. Initialize $\underline{w}(0) = 0, \quad P_0 = \delta I$
2. For each time instant, $n = 1, \ldots, N$
   2.1 $\underline{w}(n) = \underline{w}(n-1) + \underline{P}(n)\underline{u}(n)(d(n) - \underline{w}^T(n-1)\underline{u}(n))$
   2.2 $P(n) = \frac{1}{\lambda + \underline{u}(n)^T P(n-1)\underline{u}(n)}(P(n-1) - P(n-1)\underline{u}(n)\underline{u}(n)^T P(n-1))$

LMS algorithm

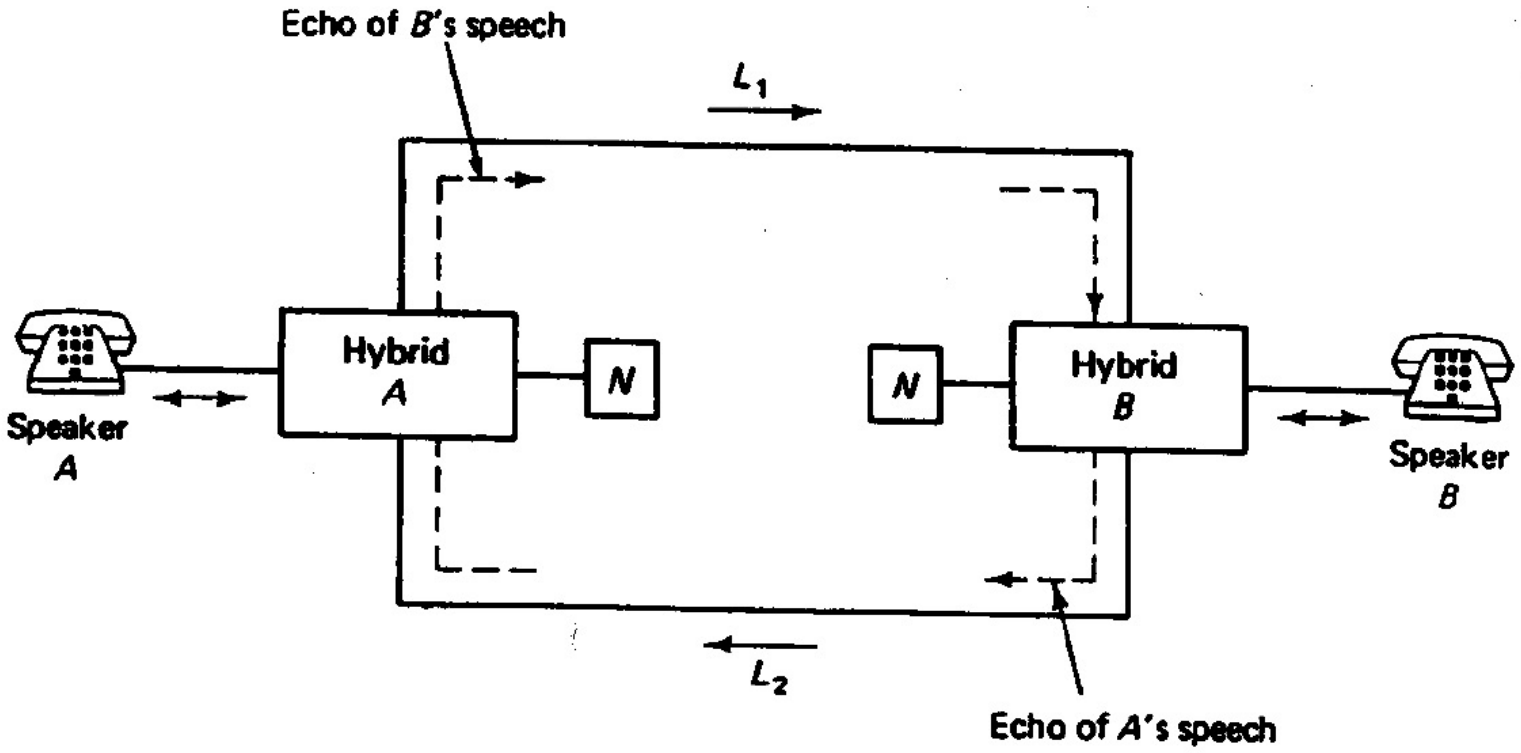
1. Initialize $\underline{w}(0) = 0$
2. For each time instant, $n = 1, \ldots, N$
   2.1 $\underline{w}(n) = \underline{w}(n-1) + \mu\underline{u}(n)(d(n) - \underline{w}^T(n-1)\underline{u}(n))$

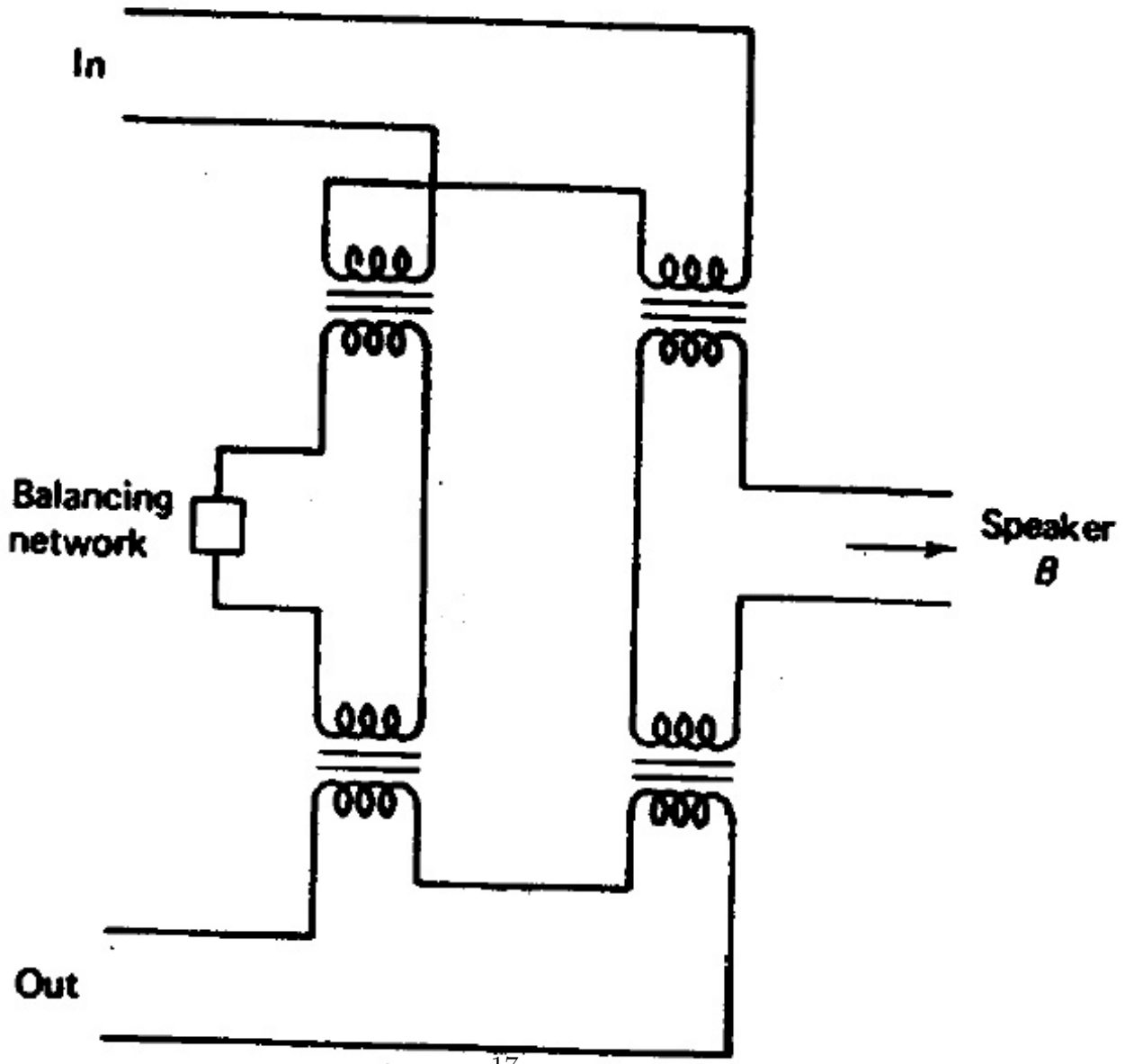


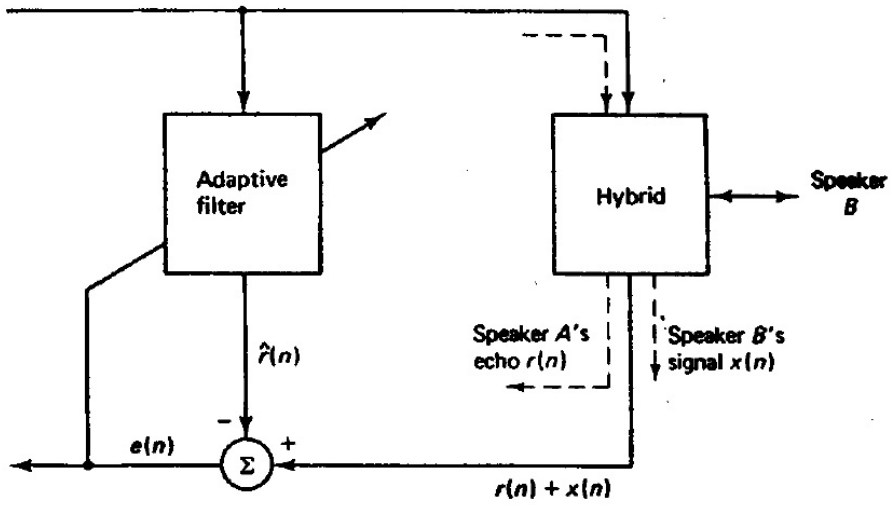
The parameters: $W = 3.1$ (i.e eigenvalue spread $\xi(R) = 11.1$). RLS: $\lambda = 1$, $\delta = 250$. LMS: $\mu = 0.075$

Echo Cancellation

- Echo in telephone networks (PSTN = public switched telephone network);
- Subjectively echo is extremely annoying, the same as a too low volume or too high noise;
- The echo is produced at the connection of four-wire circuits with two-wire circuits;
  - The customer loop (two-wire line, maximum 60 km) is connected to the central office by four-wire lines.
  - In the central office a hybrid transformer connects to four-wire lines.
  - A hybrid transformer has three ports: IN, OUT and the two-wire line.
  - If the bridge is not balanced, the signal from IN port will appear at the OUT port (distorted and delayed).
  - Speaker A will receive its own speech, do to the leakage at Speaker B hybrid, delayed by the round trip time.
  - if the round trip time is very high (in satellite communications it may be 600msec) the echo is very annoying.
  - The larger the delay, the attenuation must be more important, to alleviate the subjective discomfort.


Echo of B's speech
L1
Hybrid A
N
N
Hybrid B
Speaker A
Speaker B
L2
Echo of A's speech


In
Balancing
network
Speaker
B
Out



- Cancelling the effect of noise: Adaptive filter
  - The input signal in the adaptive filter: $u(n)$ speech signal from Speaker A
  - The output signal in the adaptive filter: a replica of speech signal from Speaker A, $y(n) = \hat{r}(n)$
  - The desired signal for the adaptive filter: the signal coming from OUT port of hybrid, i.e the sum $d(n) = r(n) + x(n)$ of the echo of speaker A $r(n)$ and of the speech from speaker B, $x(n)$
  - The error signal in the adaptive filter is $e(n) = d(n) - y(n) = r(n) + x(n) - \hat{r}(n)$

– The adaptive filter changes its parameters such that the variance of $e(n)$ is minimized. This variance is composed of two terms:
  * The variance of $r(n) - \hat{r}(n)$, which depends on the weight vector, and therefore can be reduced.
  * The variance of $x(n)$, which don't depend on the weight vector, and therefore can't be reduced.
– the length M of adaptive filter impulse response must be larger than the assumed echo path length, e.g.
  * The sampling rate in ADPCM is $8kHz$ and then sampling period is $T_s = 0.125ms$
  * If the echo path has a delay $\tau = 30ms$, then M must be selected such that $M > 240$

So far the echo canceller was placed at the near-end, to compensate for the echo produced in the hybrid. If the echo is not compensated near the hybrid, the echo canceller at the far end will have a much more difficult task, the impulse response of the filter has to be nonzero for hundreds of milliseconds, see below.