ESTIMATION, FILTERING
AND ADAPTIVE PROCESSES

COURSE NOTES FOR GRADUATE STUDENTS

2009
ESTIMATION, FILTERING AND ADAPTIVE PROCESSES

ADAPTIVE FILTERING

REFERENCES


PRECEDENCES


EXAMINATION

\[
\text{Final score} = \sum_{n=1}^{6} \text{home \_ work \_ score}(n) + \text{one \_ day \_ project \_ score}
\]

CONTENT OF THE COURSE

<table>
<thead>
<tr>
<th>No.</th>
<th>Subjects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Stochastic Processes and Models. Introduction</td>
</tr>
<tr>
<td>2.</td>
<td>Estimation theory and examples: mean, correlation, spectral density.</td>
</tr>
<tr>
<td>3.</td>
<td>Linear Optimum Filtering. Wiener Filtering.</td>
</tr>
<tr>
<td>4.</td>
<td>Least Mean Square Adaptive Filters</td>
</tr>
<tr>
<td>5.</td>
<td>Method of Least Squares</td>
</tr>
<tr>
<td>6.</td>
<td>Recursive Least-Squares Adaptive Filters</td>
</tr>
<tr>
<td>7.</td>
<td>Kalman Filters</td>
</tr>
</tbody>
</table>
Chapter 1 – Background

The filtering problem

The term estimator or filter is commonly used to refer to a system that is designed to extract information about a prescribed quantity of interest from noisy data.

In any communication system the signal received at the channel output is a noisy and distorted version of the signal that is transmitted. The function of the receiver is to operate on the received signal and deliver a reliable estimate of the original message signal to a user at the output of the system.

Estimation theory is statistical in nature because of the unavoidable presence of the noise or system errors, by contaminating the operation of the communication system.

Three basic kinds of estimation

There are three kinds of estimation: filtering, smoothing and prediction, each of which may be performed by an estimator. The difference between these operations is illustrated in Figure 2. Filtering is an operation that involves the extraction of information about a quantity of interest at time $t$, by using data measured up to and including $t$. Smoothing is an a posteriori (post datum) form of estimation, in that data measured after the time of interest are used in the estimation. The smoothed estimated time $t'$ is obtained by using data measured over the interval $[0, t]$, where $t'<t$. The benefit gained by waiting for more data to accumulate is that smoothing can yield a more accurate estimate than filtering.
**Prediction** is an estimation of an interest quantity at the time \((t+\tau)\) (future, for \(\tau > 0\)) by using data measured up to and including time \(t\).

We may classify filters as *linear* or *nonlinear*. A filter is said to be *linear* if the filtered, smoothed or predicted quantity at the output of the filter is a linear function of the observations applied to the filter input. Otherwise, the filter is nonlinear.

**Linear Filter Structures**

The impulse response of a linear filter determines the filter’s memory. On this basis, we may classify linear filters into finite-duration impulse response (FIR) and infinite-duration impulse response (IIR). The former one is called also with finite memory and the second one with infinite long memory (but fading).

A FIR filter is described by a relation

\[ y(n) = \sum_{k=0}^{M} w_k^* \cdot u(n - k) \]  

where the asterisk means *complex conjugation*.

This filter is called also transversal filter referred to as tapped-delay line filter or finite-duration impulse response or moving average filter.

The filter has an impulse response of finite duration. An FIR filter is always stable, which explains the popular use of FIR filters in practice.

In the \(z\) domain, if \(Y(z)\) and \(U(z)\) are the \(Z\) transforms of \(y(n)\) and \(u(n)\) respectively, \(W(z)\) is the transfer function of the FIR filter given by:

\[ W(z) = \frac{Y(z)}{U(z)} = w_0^* + w_1^* \cdot z^{-1} + \ldots + w_M^* \cdot z^{-M} \]  

Equation (1) could be write equivalently as

\[ y(n) = w^T \otimes u = h^T \otimes u \]  

Figure 3 – Transversal filter
that means is a (finite) convolution sum between the impulse response \( h = w \) and the filter input, \( u \). The vectors\(^*\) are defined by

\[
w = \begin{bmatrix} w_0 & w_1 & \ldots & w_M \end{bmatrix}^T \tag{3.a}
\]

and

\[
u(n) = \begin{bmatrix} u(n) & u(n-1) & \ldots & u(n-M) \end{bmatrix}^T \tag{3.b}
\]

**Linear Filters with Infinite Memory**

The discrete equation of such a filter is

\[
y(n) = \sum_{k=0}^{M} b_k \cdot u(n-k) - \sum_{k=1}^{N} a_k \cdot y(n-k) \tag{4}
\]

The structures of IIR filters contain feedback paths, which introduce a new problem: potential instability. In particular, it is possible for an IIR filter to become unstable (oscillation), unless special precaution is taken in the choice of feedback coefficients.

The transfer function of the IIR filter is given by

\[
W(z) = \frac{Y(z)}{U(z)} = \frac{b_0 + b_1 \cdot z^{-1} + \ldots + b_M \cdot z^{-M}}{1 + a_1 \cdot z^{-1} + \ldots + a_N \cdot z^{-N}} \tag{5}
\]

This type of structure is also called as the Autoregressive Moving Average (ARMA) filter. The structure of such a filter is presented is figure 4.

**Need for adaptive filters**

- Digital filters with fixed coefficients (i.e. fixed frequency response) are useful when the signal and noise spectral characteristics are known and they do not change with time.

- When the signal and noise characteristics are unknown and/or time-varying, then filters that adapt their frequency response appropriately are required. Such filters are called adaptive.

**A quick summary**

- Adaptive filters are useful when signal and noise spectral characteristics are unknown and time-varying.

- Adaptive filters are characterized by
  - the filter structure
  - the coefficient update algorithm.

\(^*\) Column vectors are commonly used
Figure 4 – The structure of the IIR filter, real value data
CHAPTER 2 - RANDOM DISCRETE PROCESSES

The term **random process**, or **stochastic process**, is used to describe the time evolution of a statistical phenomenon according to probabilistic laws.

The **time evolution** of the phenomenon means that a stochastic process is a function of time, defined on some observations interval.

The **statistical nature** of the phenomenon means that before conducting an experiment, it is not possible to define exactly the way it evolves in time. Examples of a stochastic process include speech signals, television signals, radar signals, digital computer data, the output of a communication channel, seismological data, noise, etc.

The type of interest is a **discrete** and uniformly spaced instants of time.

A random process has an infinite number of different realizations of the process. One particular realization of discrete-time stochastic process is called a time series.

For convenience of notation, we normalize time with respect to the sampling period. For example, the sequence \( u(n), u(n-1), \ldots, u(n-M) \) represents a time series that consists of the present observations \( u(n) \) made at time \( n \) and \( M \) past observations of the process made at times \( n-1, \ldots, n-M \).

1. Partial characterization of a discrete-time random process

Consider a discrete-time random process represented by the time series \( u(n), u(n-1), \ldots, u(n-M) \), which may be complex valued. To simplify the terminology we use \( u(n) \) to denote such a process instead of \( U(n) \).

The **mean-value function** of the process is defined as

\[
\mu(n) = E[u(n)]
\] (1)

where \( E[ ] \) denotes the statistical expectation operator.

The **autocorrelation function** of the process is defined as

\[
r_{uu}(n, n-k) = E[u(n) \cdot u^*(n-k)], \quad k = 0, \pm 1, \pm 2, \ldots
\] (2)

where the asterisk denotes complex conjugation.

The **autocovariance function** of the process is defined as

\[
c_{uu}(n, n-k) = E[(u(n) - \mu(n)) \cdot (u(n-k) - \mu(n-k))^*], \quad k = 0, \pm 1, \pm 2, \ldots
\] (3)

From (1), (2) and (3) the following relation is valid:

\[
c_{uu}(n, n-k) = r_{uu}(n, n-k) - \mu(n) \cdot \mu^*(n-k)
\] (3.a)

For a partial characterization (second-order) we need to specify the mean-value and the autocorrelation (or autocovariance) function for various values of \( n \) and \( k \) that are of interest.

This form of partial characterization offers two important advantages:

1. It lends itself to practical measurements.
2. It is well suited to linear operations on random processes.
For a discrete-time random process that is strictly stationary (or stationary to the second order, or wide-sense stationary (WSS)), all three quantities previously defined assume simple forms:

1. The mean-value function of the process is constant:

\[ \mu(n) = \mu, \text{ for } \forall n \]  

(4)

2. The autocorrelation (and thus the autocovariance) function depends only on the difference between the observation times \( (n) \) and \( (n-k) \):

\[ r_{uu}(n, n-k) = r_{uu}(k) \]  

(5.a)

\[ c_{uu}(n, n-k) = c_{uu}(k) \]  

(5.b)

There it is a condition for a strict stationary process. \{u(n)\} or \( u(n) \) – for short, is stationary in wide sense if and only if

\[ E[u(n)^2] < \infty, \text{ } \forall n \]  

(6)

This condition is ordinarily satisfied by random processes encountered in the physical sciences and engineering.

1.2. Mean Ergodic Theorem

The expectations, or ensemble averages, of a random process are averages “across the process”. We may also define long-term sample averages, or time averages that are averages “along the process”.

Time averages may be used to build a stochastic model of a physical process by estimating unknown parameters of the model. For such an approach to be rigorous we have to show that time averages converge to corresponding ensemble averages of the process in some statistically sense, for example on mean-square-error.

Let a constant \( \mu \) denote the mean of the process and \( c_{uu}(k) \) denote its autocovariance function of lag \( k \). For an estimate of the mean \( \mu \) we may use the time average

\[ \hat{\mu}(N) = \frac{1}{N} \sum_{n=0}^{N-1} u(n) \]  

(7)

where \( N \) is the total number of sample used in estimation. Note that the estimate \( \hat{\mu}(N) \) is a random variable with a mean and variance of its own. In particular, we readily find from (7) that a mean (expectation) of \( \hat{\mu}(N) \) is

\[ E[\hat{\mu}(N)] = E \left[ \frac{1}{N} \sum_{n=0}^{N-1} u(n) \right] = \frac{1}{N} \sum_{n=0}^{N-1} E[u(n)] = \frac{1}{N} \cdot N \cdot \mu = \mu, \text{ } \forall N \]  

(8)

\footnote{Not mandatory}
We say the time average $\hat{\mu}(N)$ is an unbiased estimator of the ensemble mean average of the process.

We say that the process $u(n)$ is mean ergodic in the mean-square error sense if the mean-square value of the error between the ensemble average $\mu$ and time average $\hat{\mu}(N)$ approaches zero as the number of samples, $N$, approaches infinity:

$$\lim_{N \to \infty} E \left[ \mu - \hat{\mu}(N) \right] = 0$$  \hspace{1cm} (9)

Using the time average formula we may write

$$E \left[ \mu - \hat{\mu}(N) \right]^2 = E \left[ \mu - \frac{1}{N} \sum_{n=0}^{N-1} u(n) \right]^2 = \frac{1}{N^2} \cdot E \left[ \sum_{n=0}^{N-1} (u(n) - \mu)^2 \right] = \frac{1}{N^2} \cdot \sum_{n=0}^{N-1} \sum_{k=0}^{N-1} E\{(u(n) - \mu)(u(k) - \mu)^*\}$$

$$= \frac{1}{N^2} \cdot \sum_{n=0}^{N-1} \sum_{k=0}^{N-1} c(n-k)$$

Let $l = n - k$ then

$$E \left[ \mu - \hat{\mu}(N) \right]^2 = \ldots = \frac{1}{N} \cdot \sum_{l=-\infty}^{N-1} \left(1 - \frac{|l|}{N}\right) \cdot c(l)$$  \hspace{1cm} (11)

We can state that the necessary and sufficient condition for the process $u(n)$ to be mean ergodic in the mean-square-error is that

$$\lim_{N \to \infty} \frac{1}{N} \sum_{l=-\infty}^{N-1} \left(1 - \frac{|l|}{N}\right) \cdot c(l) = 0$$  \hspace{1cm} (12)

In other words, if the process $u(n)$ is asymptotically uncorrelated in the sense of Eq. (10) then the time average $\hat{\mu}(N)$ of the process converges to the ensemble average $\mu$ in the mean square error sense. This is the statement of a particular form of the mean ergodic theorem.

The use of the theorem may be extended to other time averages of the process. If the following time averages is used to estimate the autocorrelation function of a WSS process:

$$\hat{r}_{uu}(k, N) = \frac{1}{N} \sum_{n=0}^{N-1} u(n)u(n-k), \; 0 \leq k \leq N-1$$  \hspace{1cm} (13)

the process $u(n)$ is ergodic in correlation in the mean square error sense if

$$\lim_{N \to \infty} E \left[ \hat{r}(k) - \hat{r}(k, N) \right]^2 = 0$$  \hspace{1cm} (14)
where \( r(k) \) is the real value and \( \hat{r}(k, N) \) is the estimated value.

**The correlation matrix**

Let the \( M \)-by-1 observation vector \( \mathbf{u}(n) \) represent the elements of the zero-mean time series: \( u(n), u(n-1), \ldots, u(n-M+1) \). The vectorial representation is

\[
\mathbf{u}(n) = \begin{bmatrix} u(n) & u(n-1) & \ldots & u(n-M+1) \end{bmatrix}^T
\tag{15}
\]

where the superscript \( T \) denotes transposition.

We define the autocorrelation matrix of a stationary discrete-time stochastic process as the expectation of the outer product of the observation vector \( \mathbf{u}(n) \) with itself. Thus we may write

\[
\mathbf{R}_{uu} = E\left[\mathbf{u}(n) \cdot \mathbf{u}^H(n)\right]
\tag{16}
\]

where \( H \) denotes the Hermitian transposition (i.e., the operation of transposition combined with complex conjugation). Substituting (13) in (12), we express the correlation matrix in the expanded form:

\[
\mathbf{R}_{uu} = \begin{bmatrix}
    r_{uu}(0) & r_{uu}(1) & \ldots & r_{uu}(M-1) \\
    r_{uu}(-1) & r_{uu}(0) & \ldots & r_{uu}(M-2) \\
    \vdots & \vdots & \ddots & \vdots \\
    r_{uu}(-M+1) & r_{uu}(-M+2) & \ldots & r_{uu}(0)
\end{bmatrix}
\tag{17}
\]

The elements of the main diagonal are always real valued. For complex-valued data, the remaining elements of \( \mathbf{R} \) assume complex values.

**Properties of the correlation matrix**

**P1:** The correlation matrix of a stationary discrete-time random process is Hermitian. (A complex valued matrix is Hermitian if it is equal to its transpose).

\[
\mathbf{R}^H = \mathbf{R}
\tag{18.a}
\]

or

\[
r(-k) = r^*(k), \quad \forall k = 0, \pm 1, \pm 2, \ldots
\tag{18.b}
\]

For a WSS process, we only need \( M \) values of the autocorrelation function \( r_{uu}(k) \), \( k=0,1,2,\ldots,M-1 \), in order to completely define the correlation matrix \( \mathbf{R}_{uu} \). We can rewrite the matrix as:
For real-valued data, the autocorrelation function \( r(k) \) is real for all \( k \) and the correlation matrix \( \mathbf{R} \) is symmetric.

**P2:** The correlation matrix of a stationary discrete-time random process is **Toeplitz**. (A square matrix is Toeplitz if all elements on its main diagonal are equal and if the elements on any other diagonal parallel to the main diagonal are also equal)

\[
\mathbf{R}_{uu}^{H} = \mathbf{R}_{uu} \quad (17.\text{a})
\]

or

\[
r_{uu}(-k) = r_{uu}^{*}(k), \quad \forall k = 0,\pm1,\pm2,\ldots \quad (17.\text{b})
\]

We can say also the inverse: if the correlation matrix is Toeplitz then \( \mathbf{u}(n) \) must be wide-sense stationary.

**P3:** The correlation matrix of a stationary discrete-time random process is always nonnegative definite and almost always positive definite.

\[
\text{Let } \mathbf{a} \text{ be an arbitrary (nonzero) } M\text{-by-1 complex valued vector. If } \mathbf{a}^{H} \cdot \mathbf{R} \cdot \mathbf{a} > 0 \text{ for every } \mathbf{a} \neq \mathbf{0} \text{ then } \mathbf{R} \text{ is positive semidefinite.}
\]

**P4:** The correlation matrix \( \mathbf{R}_{uu} \) of a WSS process is nonsingular due the unavoidable presence of additive noise.

The matrix \( \mathbf{R}_{uu} \) is said to be nonsingular if its determinant, denoted by \( \det(\mathbf{R}_{uu}) \), in nonzero. The implication is that always the inverse exists:

\[
\mathbf{R}_{uu}^{-1} = \frac{\text{adj}(\mathbf{R}_{uu})}{\det(\mathbf{R}_{uu})} \quad (18)
\]

where \( \text{adj}(\mathbf{R}_{uu}) \) is the adjoint of \( \mathbf{R}_{uu} \). (By definition, \( \text{adj}(\mathbf{R}_{uu}) \) is the transpose of a matrix whose entries are cofactors of the elements in \( \det(\mathbf{R}_{uu}) \).)

**Complex Gaussian Processes**

Let \( \mathbf{u}(n) \) denotes a complex Gaussian process consisting of \( N \) samples. For the first and second-order statistics of this process, we assume the following:

1. A mean of zero, as shown by

\[
\mu = E[\mathbf{u}(n)] = 0, \quad \text{for } 1,2,\ldots,N \quad (1)
\]
2). An autocorrelation function denoted by:

\[ r_{uu}(k) = E[u(n) \cdot u^*(n-k)] \quad \text{for} \quad k = 0,1,2,...,N-1 \]  

(2)

The set of the autocorrelation values \( \{r_{uu}(k), k = 0,1,2,3,...,N-1\} \) defines the correlation matrix \( R_{uu} \) of the Gaussian process \( u(n) \).

Equations (1) and (2) imply wide-sense stationarity of the process. Knowledge of the mean and the autocorrelation function for varying values of lag \( k \) are sufficient for a complete characterization of the complex Gaussian process \( u(n) \).

Properties of the zero-mean complex Gaussian process \( u(n) \) that is WSS:

**P1**). The process \( u(n) \) is stationary in the strict sense.

**P2**). The process \( u(n) \) is circularly complex, in the sense that any two different sample \( u(n) \) and \( u(k) \) of the process satisfy the condition

\[ E[u(n) \cdot u(k)] = 0, \quad \text{for} \quad n \neq k \]  

(3)

Power spectral density

The autocorrelation function is a time-domain description of the second-order statistics of a stochastic process. The frequency-domain counterpart of this statistical parameter is the power spectral density, which is also referred to as power spectrum or, simply, spectrum.

Let a windowed time series as

\[ u_N(n) = \begin{cases} u(n), & n = 0, \pm 1, \pm 2, ..., \pm N \\ 0, & |n| > N \end{cases} \]  

(4)

By definition, the discrete-time Fourier transform of the windowed time series \( u_N(n) \) is given by:

\[ U_N(\omega) = \sum_{n=-N}^{N} u_N(n) \cdot e^{-j \omega n} \]  

(5)

where \( \omega \) is the angular frequency, lying in the interval \(( -\pi, \pi )\). In general, \( U_N(\omega) \) is complex valued; specifically, its complex conjugate is given by

\[ U_N^*(\omega) = \sum_{n=-N}^{N} u_N^*(n) \cdot e^{j \omega n} \]  

(6)

By multiplication of the last two relations we obtain

\[ |U_N(\omega)|^2 = \sum_{n=-N}^{N} \sum_{k=-N}^{N} u_N(n) \cdot u_N^*(k) e^{-j \omega (n-k)} \]  

(7)
The quantity

\[
S(\omega) = \lim_{N \to \infty} \frac{1}{N} E\left[|U_N(\omega)|^2\right]
\]

is “the spectral density of expected power”, which is abbreviated as the power spectral density of the process.

The quantity

\[
\frac{|U_N(\omega)|^2}{N}
\]

is called periodogram of the windowed time series \(u_N(n)\).

When the limit (7) exists, the quantity \(S(\omega)\) has the following interpretation: “\(S(\omega)\)\(d\omega\) = average of the contribution to the total power from components of a wide-sense stationary random process with angular frequencies located between \(\omega\) and \(\omega + d\omega\). The average is taken over all possible realizations of the process”.

Some properties are useful in many real applications:

**P1:** The Wiener-Hinchin theorem:

\[
S(\omega) = \sum_{l = -\infty}^{+\infty} r(l) \cdot e^{-j\omega l}, \quad -\pi < \omega \leq \pi
\]

(9.a)

\[
r(l) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega) \cdot e^{j\omega l} \cdot d\omega, \quad l = 0, \pm 1, \pm 2, \ldots
\]

(9.b)

**P2.** The frequency support of the power spectral density \(S(\omega)\) is the interval \(-\pi < \omega \leq \pi\). Outside this interval, \(S(\omega)\) is periodic:

\[
S(\omega + 2k\pi) = S(\omega), \quad k = \pm 1, \pm 2, \ldots
\]

(10)

**P3:** The power spectral density of a stationary discrete-time stochastic process is real.

**P4:** The power spectral density of a real-valued discrete time stochastic process is even (i.e., symmetric), \(S(\omega) = S(-\omega)\); if the process is complex valued, its power spectral density is not necessarily even.

**P5:** The mean-square value of a stationary discrete-time stochastic process equals the area under the power spectral density curve, for \(-\pi < \omega \leq \pi\), except for a scaling factor of \(1/2\pi\).

\[
r(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\omega) d\omega
\]

(11)
NUMERICAL EXAMPLES

\[ v = \begin{bmatrix} 0.4230 & -3.4762 & 2.1872 & 2.7006 & -1.8346 \end{bmatrix}; \]
\[ u = \begin{bmatrix} 1.9021 & 1.6180 & 1.1756 & 0.6180 & 0.0000 \end{bmatrix}; \]
\[ y = \begin{bmatrix} 2.3251 & -1.8581 & 3.3628 & 3.3186 & -1.8346 \end{bmatrix}; \]
\[ r_{yy} = \begin{bmatrix} -0.8531 & 2.2250 & -0.9034 & -1.0995 & 6.9092 & -1.0995 & -0.9034 & 2.2250 & -0.8531 \end{bmatrix}; \]
\[ r_{uu} = \begin{bmatrix} 0.00 & 0.2351 & 0.6472 & 1.1413 & 1.6000 & 1.1413 & 0.6472 & 0.2351 & 0.00 \end{bmatrix}; \]
\[ r_{vv} = \begin{bmatrix} -0.1552 & 1.5039 & -2.4950 & -1.6242 & 5.5411 & -1.6242 & -2.4950 & 1.5039 & -0.1552 \end{bmatrix}; \]
\[
xcorr(u,v) = [-3.4896, 2.1684, \textbf{6.3732}, -1.0322, -0.5797, -2.0503, -1.6512, 0.2614, 0.00];
\]
\[
xcorr(u,y) = [-3.4896, 3.3439, \textbf{9.6093}, 4.6741, 7.4203, 3.6560, 1.5849, 1.4370, 0.0];
\]
\[
xcorr(v,u) = [0.0, 0.2614, -1.6512, -2.0503, -0.5797, -1.0322, \textbf{6.3732}, 2.1684, -3.4896];
\]
\[
xcorr(y,u) = [0.0, 1.4370, 1.5849, 3.6560, 7.4203, 4.6741, \textbf{9.6093}, 3.3439, -3.4896];
\]

\[
\begin{align*}
R_{yy} &= [6.9092, -1.0995, -0.9034, 2.2250, -0.8531] \\
&[\quad -1.0995, \textbf{6.9092}, -1.0995, -0.9034, 2.2250] \\
&[-0.9034, -1.0995, \textbf{6.9092}, -1.0995, -0.9034] \\
&[\quad 2.2250, -0.9034, -1.0995, \textbf{6.9092}, -1.0995] \\
&[-0.8531, 2.2250, -0.9034, -1.0995, \textbf{6.9092}]
\end{align*}
\]
MORE SAMPLES
Course 2

Estimating the parameters of random processes from data

1. **Introduction**

A stochastic discrete process has a model and a model has parameters. The model, like structure and order, is the result of the identification process. The parameters of the model are the result of an estimation process.

In such problems of estimating unknown parameters are two important questions:
1. What is a good method of using the data to estimate the unknown parameters.
2. How “good” is the resulting estimate.

The basic premise of estimation is to determine the value of an unknown quantity using a statistic, that is, using a function of measurements. The estimator \( g(X_1, X_2, \ldots, X_n) \) is a random variable. A specific set of measurements will result in \( X_i = x_i, \ i=1,2,\ldots, N \) and the resulting value \( g(x_1, x_2, \ldots, x_n) \) will be called an estimate or an estimated value.

2. **Definition of a statistic**

Let \( X_1, X_2, \ldots, X_n \) be \( n \) i.i.d\(^1\) random variables from a given distribution function \( F_X \). Then \( Y = g(x_1, x_2, \ldots, x_n) \) is called a statistic if the function \( g \) does not depend on any unknown parameter. For example

\[
\bar{X} = \frac{1}{N} \sum_{i=1}^{N} x_i
\]

(1)

is a statistic. The quantity

\[
\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)^2
\]

(2)

is not a statistic because it depends upon the unknown parameter, \( \mu \).

3. **Parametric and non-parametric estimators**

There are two classes of estimation techniques: parametric and non-parametric. For a parametric method, we might assume, for example, that \( f_X(x) \) is Gaussian with parameters \( \mu_X \) and \( \sigma_X^2 \), whose values are not known. We estimate the values of these parameters from data, and substitute these estimated values in \( f_X(x) \) to obtain an estimate of \( f_X(x) \) for all values of \( x \).

In the non-parametric approach, we do not assume a functional form for \( f_X(x) \) directly from data for all values of \( x \).

4. **Point estimators of parameters**

---

1. i.i.d. = independent and identically distributed
The basic purpose of point estimation is to estimate an unknown parameter with a statistic, that is, with a function of the i.i.d. measurements. Assume that the unknown parameter is \( \theta \) and there is a set of \( N \) samples, i.i.d. measurements \( X_1, X_2, ..., X_n \). We then form a statistic \( g(x_1, x_2, ..., x_n) = \hat{\theta} \) which is called the point estimator of \( \theta \). A specific sample will result in \( X_i = x_i, \ i = 1, 2, ..., N \) and the resulting value of \( \hat{\theta} \) will be called the point estimate of \( \theta \).

The estimator is a random variable that will take on different values depending on the values of the measurements, whereas the estimate is a number.

**Estimation of the mean**

The mean \( \mu_X \) of a random variable is usually estimated by the (arithmetic) average of the samples:

\[
\hat{\mu}_X = \overline{X} = \frac{1}{N} \sum_{i=1}^{N} x_i
\]

where the \( X_i \)'s are i.i.d. measurements or observations from the population with the distribution \( X_F \). \( \overline{X} \) is the most familiar estimator of \( \mu_X \). However, the following two estimators of \( \mu_X \) are sometimes (not often) used:

\[
\hat{\mu}_X = \frac{1}{2} (X_{\text{max}} + X_{\text{min}})
\]

The value \( x \) such that

\[
\hat{F}_X(x_{1}, x_{2}, ..., x_{N}) = \frac{1}{2}
\]

this is called the empirical median.

**Estimation of the variance**

The variance \( \sigma_X^2 \) of a random variable \( X \) is commonly estimated by:

\[
\hat{\sigma}_X^2 = \frac{1}{N} \sum_{i=1}^{N} (X_i - \overline{X})^2
\]

or by the estimator

\[
S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \overline{X})^2
\]

**An estimator of probability**

Let \( P \) be the probability of an event \( A \). Then a statistic that can be used to estimate \( P \) is:
where \( N_A \) is the random variable that represents the number of times that event \( A \) occurs in \( N \) independent trials.

**Estimators of the covariance**

The covariance \( \sigma_{XY} \) is usually estimated by

\[
\hat{\sigma}_{XY} = \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y})
\]

or by

\[
\hat{\sigma}_{XY} = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \bar{X})(Y_i - \bar{Y})
\]

5. **Notation for estimators**

Our interest in estimating the unknown parameter \( \theta \) will be reflected by writing the (probability) distribution function of the random variable \( X \) by writing as a function of \( \theta \):

\[
F_X(x; \theta) = F(x; \theta)
\]

**Example:** Consider a normal (gaussian) random variable with unit variance, \( \sigma^2 = 1 \), and unknown mean \( \mu \). Then

\[
f(x; \mu) = f_X(x; \mu) = f_X(x; \theta = \mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2}}
\]

The only change is that we changed the notation in order to reflect the fact that we now change the model to include a family of distributions. Each value of \( \theta \) corresponds with one member of the family.

The purpose of the experiments and the resulting estimators (one for each experiment) is to select one member of the family as being the “best”.

6. **Maximum likelihood estimators**

It is the most common method of deriving estimators. As before, we use the notation \( F(x; \theta) \) to indicate the distribution of the random variable \( X \) given the parameter \( \theta \). Thus, if \( X_1, X_2, ..., X_n \) are i.i.d measurements of \( X \), then \( f(x_1, x_2, ..., x_n; \theta) \) is the joint pdf\(^2\) of \( X_1, X_2, ..., X_n \) given \( \theta \), and because of independence it can be written:

\(^2\) pdf = probability density function
\[
f(x_1, x_2, \ldots, x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)
\]  

(8)

Now, if the values \(X_1, X_2, \ldots, X_n\) are considered fixed and \(\theta\) is an unknown parameter, then \(f(x_1, x_2, \ldots, x_n; \theta)\) is called a likelihood function and is usually denoted by:

\[
L(\theta) = f(x_1, x_2, \ldots, x_n; \theta) = \prod_{i=1}^{n} f(x_i; \theta)
\]  

(9)

Example 1: Let \(X\) a random variable with an exponential distribution, that is

\[
f(x; \theta) = \begin{cases} \frac{1}{\theta} e^{-\frac{x}{\theta}}, & x \geq 0, \theta > 0 \\ 0, & x < 0 \end{cases}
\]

If we have five independent and identically distributed (i.i.d) measurements of \(X\) that are 10, 11, 8, 12, and 9, find the likelihood function of \(\theta\).

Solution:

\[
L(\theta) = \prod_{i=1}^{5} f(x_i; \theta) = \prod_{i=1}^{5} \frac{1}{\theta} e^{-\frac{x_i}{\theta}} = \theta^{-5} \cdot e^{-\frac{50}{\theta}}, \quad \theta > 0
\]

The value \(\hat{\theta}\) that maximizes the likelihood function is called a maximum likelihood estimator of \(\theta\). That is, a value \(\hat{\theta}\) such that for all values \(\theta\)

\[
L(\hat{\theta}) = f(x_1, x_2, \ldots, x_n; \hat{\theta}) \geq f(x_1, x_2, \ldots, x_n; \theta) = L(\theta)
\]  

(10)

is called a maximum likelihood estimate of \(\theta\). Such an estimate is justified on the basis that \(\hat{\theta}\) is the value that maximizes the joint probability density (likelihood), given the sample of observations or measurements that was obtained.

Example 2: Find the maximum likelihood estimate of \(\theta\) from the previous example.

Solution:

\[
L(\theta) = \theta^{-5} \cdot e^{-\frac{50}{\theta}}, \quad \theta > 0
\]

\[
\frac{dL(\theta)}{d\theta} = -5 \cdot \theta^{-6} \cdot e^{-\frac{50}{\theta}} + \theta^{-5} \cdot \frac{50}{\theta^2} \cdot e^{-\frac{50}{\theta}}
\]

Solving the derivative equal to zero and solving of \(\hat{\theta}\), the value of \(\theta\) that causes the derivative to equal zero yields the estimate

\[\text{likelihood} = \text{plausible}\]
Estimation, Filtering and Adaptive Processes

\[
\frac{dL(\theta)}{d\theta} = 0, \quad e^{-\left(\frac{50}{\theta}\right)} \cdot \theta^{-6} \cdot \left(-5 + \frac{50}{\theta}\right) = 0
\]

and

\[
\hat{\theta} = \frac{50}{5} = 10
\]

Example 3: Let \( N \) samples of i.i.d measurements from a normal distribution with known variance \( \sigma^2 \). Find the maximum likelihood estimate of the mean.

Solution:

\[
L(\mu) = f(x_1, x_2, ..., x_n; \mu) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \mu)^2}{2\sigma^2}}
\]

Finding the value that maximize \( \ln[L(\mu)] \) is equivalent to find the value of \( \mu \) that maximizes \( L(\mu) \). Thus:

\[
g(\mu) = \ln[L(\mu)] = N \cdot \ln \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2
\]

\[
\frac{dg}{d\mu} = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} \left(2(x_i - \mu) \bigg| \mu = \hat{\mu}\right) = 0
\]

or

\[
N \cdot \hat{\mu} = \sum_{i=1}^{N} x_i \rightarrow \hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} x_i
\]

Note that in this case the maximum likelihood estimator is simply \( \bar{x} \).

Example 4: Let \( X \) a uniformly distributed between 0 and \( \hat{\theta} \). Find the maximum likelihood estimator of \( \theta \) based on a sample of \( N \) observations.

Solution: With \( f(x; \theta) = \begin{cases} \frac{1}{\theta}, & 0 \leq x \leq \theta, \theta < 0 \\ 0, & \text{elsewhere} \end{cases} \) we have \( L(\theta) = \frac{1}{\theta^N}, 0 \leq x_i \leq \theta, i = 1,2, ..., N \), where \( x_i, i = 1,2, ..., N \) represent the values of the \( N \) i.i.d. measurements. The maximum of \( \frac{1}{\theta^N} \) cannot be found by the differentiation because the smallest possible value of \( \theta > 0 \)

\[\text{Because the natural logarithms is a strictly increasing function, the value of } \theta \text{ that maximizes the likelihood function is the same value that maximizes the logarithm of the likelihood function.}\]
maximizes $L(\theta)$. The smallest possible value of $\theta$ is $\max(x_i)$ because of the constraint $x_i \leq \theta$. Thus, the maximum likelihood estimator of $\theta$ is $\hat{\theta} = \max(x_i)$.

7. Measures of the quality of estimators

It seems natural to wish $\hat{\theta} = \theta$, but $\hat{\theta}$ is a random variable. Thus, we must adopt some probabilistic criteria for measuring how close $\hat{\theta}$ is to $\theta$.

7.1. Bias

An estimator $\hat{\theta}$ of $\theta$ is called unbiased if

$$E\{\hat{\theta}\} = \theta$$

(11)

If

$$E\{\hat{\theta}\} = a \neq \theta$$

(11.a)

then $\hat{\theta}$ is said to be biased and the bias or bias error is:

$$b\{\hat{\theta}\} = \text{estimated value} - \text{real value} = E\{\hat{\theta}\} - \theta$$

(11.b)

**Example 1:** If $X_1, X_2, \ldots, X_N$ are i.i.d with mean $\mu$ then

$$\bar{X} = \hat{\mu} = g(x_1, x_2, \ldots, x_N) = \frac{X_1 + X_2 + \ldots + X_N}{N}$$

is an unbiased estimator of the mean $\mu$ because:

$$E\{\bar{X}\} = E\{\hat{\mu}\} = \frac{E\{X_1 + X_2 + \ldots + X_N\}}{N} = \frac{N \cdot \mu}{N} = \mu.$$  

**Example 2:** Compute the expected value of $\hat{\sigma}^2 = \frac{1}{N} \sum_{i=1}^{N} (X_i - \bar{X})^2$ and determine whether it is an unbiased estimator of variance $\sigma^2$.

**Solution:** The mean of the estimator is $E\{\hat{\sigma}^2\} = \ldots = \sigma^2 \cdot \frac{N-1}{N}$ thus $\hat{\sigma}^2$ is a biased estimator of $\sigma^2$. However it is easy to see that:

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \bar{X})^2$$

is an unbiased estimator of $\sigma^2$. For large $N$, $\hat{\sigma}^2$ and $S^2$ are nearly the same. The intuitive idea is that one degree of freedom is used in determining $\bar{X}$. That is, $\bar{X}, X_1, X_2, \ldots, X_{N-1}$ determine $X_N$. So, the sum should be divided by $(N-1)$ rather than by $N$.

---

Because it is a function of random variables
There are also other estimators. Thus, some other measure(s) of estimators is needed to separate good estimators from those not so well.

If the estimator $\hat{\theta}$ has a mean of $\theta$, then we also desire $\hat{\theta}$ to have a small variation from one sample to the next. This variation can be measured in various ways. For instance, measures of variations are:

1. $E|\hat{\theta} - \theta|$
2. $[\max(\hat{\theta}) - \min(\hat{\theta})]$
3. $E[(\hat{\theta} - \theta)^2]$

Although any of these or other measures might be used, the most common measure is the mean squared error (MSE):

$$MSE = E\left\{ (\hat{\theta} - \theta)^2 \right\}$$

If $\hat{\theta}$ is unbiased then the MSE is simply the variance of $\hat{\theta}$. If $E[\hat{\theta}] = m$ then

$$MSE = E\left\{ (\hat{\theta} - \theta)^2 \right\} = (\theta - m)^2 + \sigma^2_{\theta}$$

The important result may be stated as:

$$MSE(\hat{\theta}) = Bias(\hat{\theta})^2 + var(\hat{\theta})$$

The variance of $\bar{X}$ is

$$var(\bar{X}) = var\left( \frac{\sum_{i=1}^{N} X_i}{N} \right) = \frac{\sigma^2}{N}$$

If we have only one measurement, then:

$$var(X_1) = \sigma^2$$

The average $\bar{X}$ has a lower variance and by criterion of minimum variance of minimum MSE, $\bar{X}$ is a better estimator than a single measurement.

The positive square root of the variance of $\hat{\theta}$ is often called the standard error of $\hat{\theta}$. The positive square root of the MSE is called the root mean square error or the RMS error.

For a given sample size $N$, $\hat{\theta}_N = g(X_1, X_2, ..., X_N)$ will be called an unbiased minimum variance estimator of the parameter $\theta$ if $\hat{\theta}_N$ is unbiased and if the variance of $\hat{\theta}_N$ is less than or equal to the variance of every other unbiased estimator of $\theta$ that uses a sample of size $N$.

If $\theta \neq 0$, the normalized bias, normalized standard deviation (also called the coefficient of variation), and the normalized RMS errors are defined as:

- normalized bias:
\[
\varepsilon_b = \frac{\hat{\theta} - \theta}{\sigma}\quad (18.a)
\]
- normalized standard error:
\[
\varepsilon_r = \frac{\sigma \hat{\theta}}{\theta}\quad (18.b)
\]
- normalized RMS error
\[
\varepsilon = \sqrt{\frac{E[(\hat{\theta} - \theta)^2]}{\theta^2}}\quad (18.c)
\]
- normalized MSE
\[
\varepsilon^2 = \frac{E[(\hat{\theta} - \theta)^2]}{\theta^2}\quad (18.d)
\]

9. Consistent estimators

Any statistic or estimator that converges in probability to the parameter being estimated is called a consistent estimator of that parameter. For example:
\[
\hat{X}_N = \frac{1}{N} \sum_{i=1}^{N} X_i\quad (19)
\]
has mean \(\mu\) and variance \(\frac{\sigma^2}{N}\). Thus, as \(N \to \infty\), \(\bar{X}_N\) has mean \(\mu\) and a variance that approaches to 0. Thus, \(\bar{X}_N\) converges in probability to \(\mu\) and \(\bar{X}_N\) is a consistent estimator of \(\mu\). Note that also:
\[
\frac{1}{N} \sum_{i=1}^{N} (X_i - \hat{X}_N)^2\quad (20)
\]
and
\[
\frac{1}{N-1} \sum_{i=1}^{N} (X_i - \hat{X}_N)^2\quad (21)
\]
are consistent estimators of variance, \(\sigma^2\).

10. Efficient estimators

Let \(\hat{\theta}_1\) and \(\hat{\theta}_2\) be unbiased estimators of \(\theta\). Then we can define the (relative) efficiency of \(\hat{\theta}_1\) with respect to \(\hat{\theta}_2\) as:
In some cases it is possible to find among the unbiased estimators one that has the minimum variance, \( V \). In such a case, the absolute efficiency of an unbiased estimator \( \hat{\theta}_1 \) is

\[
\frac{V}{\text{var}(\hat{\theta}_1)}
\]

11. Tests for stationarity and ergodicity

In order to estimate any of the unknown parameters of a random process, the usual practice is to estimate these parameters from one sample function of the random process. So, ergodicity is assumed and thus, stationarity is also assumed. If

\[
\int_{-\infty}^{\infty} |R_{XX}(\tau)| d\tau < \infty
\]

then the process is ergodic. Thus, if we assume that the process is Gaussian and that this condition is met, then testing for stationarity is equivalent to testing for ergodicity.

Stationarity can and should be tested when estimating the parameters of a random process. Random sequences encountered in practice may be classified into three categories:

1). Those that are stationary over long periods of time. The underlying process seems stationary, for example, as with thermal noise and the resulting data do not fail the standard stationarity tests. No series will be stationary indefinitely.

2). Those that may be considered stationary for short periods of time.

3). Sequences that are obviously non-stationary: such series possibly may be transformed into (quasi)-stationary series.

One requirement of stationarity is that \( f_X \) (probability density function) does not vary with \( N \). In particular, the mean and the variance should not vary with \( N \). A reasonable method of determining whether this is true is to divide the data into two or more sequential sections and calculate the sample mean and the sample variance from each section. If there are not big variations of the mean and of the sample for different sections, then the random process is stationary.

12. Autocorrelation function estimation

With \( N \) data points we can only estimate \( R_{XX}(k) \) for values of \( k \) less than \( N \), that is:

\[
\hat{R}_{XX}(k) = \begin{cases} 
\frac{1}{N-k} \sum_{i=0}^{N-k-1} X(i) \cdot X(i+k), & k < N \\
0, & k \geq N \\
\hat{R}_{XX}(-k), & k < 0
\end{cases}
\]

(25)
This is equivalent to truncating the estimator for \(|k| \geq N\).

As \(k \to N\), we are using fewer and fewer points to obtain the estimate of \(R_{XX}(k)\). This leads to larger variances in the estimated value of \(R_{XX}(k)\) for \(k \to N\). (see Fig. 1).

![Figure 1 - Truncation in the estimation of the autocorrelation](image)

It is easy to show that

\[
E\left\{ \hat{R}_{XX}(k) \right\} = R_{XX}(k), \quad k < N
\]

that is, the estimator is unbiased for \(k < N\).

**Example:** Estimate the autocorrelation function of a zero-mean white Gaussian random process with \(R_{XX}(k) = \begin{cases} \sigma^2_X, & k = 0 \\ 0, & k \neq 0 \end{cases}\). Find the variance of the \(\hat{R}_{XX}(k)\).

**Solution:** For \(k = 0\) we can write:

\[
\hat{R}_{XX}(0) = \frac{1}{N} \sum_{i=0}^{N-1} [X(i)]^2
\]

\[
E\left\{ \hat{R}_{XX}(0) \right\} = \frac{1}{N} \sum_{i=0}^{N-1} E\left\{X(i)^2\right\} = R_{XX}(0)
\]

\[
var\left\{ \hat{R}_{XX}(0) \right\} = E\left\{ [\hat{R}_{XX}(0)]^2 \right\} - [R_{XX}(0)]^2 = \ldots = \frac{N + 2}{N} \sigma^4_X - \left[ \sigma^2_X \right]^2 = \frac{2 \sigma^4_X}{N}
\]

Similarly, it can be shown that:

\[
var\left\{ \hat{R}_{XX}(k) \right\} = \frac{1}{N - |k|} \sigma^4_X, \quad 1 \leq k \leq N
\]

Note that as \(k \to N\) the variance increases rapidly.
13. Estimation of Power Spectral Density Functions

The psd function of a stationary random process (continue time) is defined as:

\[ S_{XX}(f) = \int_{-\infty}^{\infty} R_{XX}(\tau) e^{-j2\pi f \cdot \tau} d\tau \] (27)

We can define an estimator for the psd as:

\[ \hat{S}_{XX}(f) = \int_{-\infty}^{\infty} \hat{R}_{XX}(\tau) e^{-j2\pi f \cdot \tau} d\tau \] (28)

where \( \hat{R}_{XX}(\tau) \) is an estimator of \( R_{XX}(\tau) \).

In the discrete case we can estimate \( \hat{S}_{XX}(f) \) using the estimator:

\[
\hat{S}_{XX}(f) = \frac{1}{N-k} \sum_{i=0}^{N-k-1} X(i) \cdot X(i+k), \quad k = 0,1,2,\ldots,N-1
\]

where

\[
\hat{R}_{XX}(k) = \frac{1}{N-k} \sum_{i=0}^{N-k-1} X(i) \cdot X(i+k), \quad k = 0,1,2,\ldots,N-1
\] (29.a)

is the estimation of the autocorrelation function defined in the preceding section.

The estimator given in Equation (29) has some major drawbacks. First, Equation (29) is computationally intensive. Second, the variance of the estimator is large and there is no guarantee that Equation (29) will yield a nonnegative value for the discrete estimate of the psd. These problems can be overcome by using a DFT-based estimator of the psd called periodogram.

---

Appendix – DFT

The sampling theorem says the sampling in the time domain should be at

\[ T_S < \frac{1}{2B}, \]

where \( B \) is the bandwidth of the random process, that is, \( S_{XX}(f) = 0, |f| > B \). It is a common fact to normalize the sampling interval \( T_S \), to be 1, which requires that \( B < \frac{1}{2} \). More, this imposes a restriction, \( |f| < \frac{1}{2} \), for Fourier transform of sequences.
If \( x(t) = 0 \) for \( t < t_1 \) or \( t > t_1 + T_M \), then by an identical type of argument, we can sample in the frequency domain at an interval \( f_s < \frac{1}{T_M} \). If we have normalized, \( T_s = 1 \), and if \( x(n) = 0 \) for \( n < 0 \) or \( n > N - 1 \), then we choose \( f_s = \frac{1}{N} \), then we completely represent the signal \( x(t) \). If this is the case, then we have the usual Fourier transform of a sequence

\[
X_F(f) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi n f}, \quad 0 \leq f \leq 1
\]

where we have now taken the principle part of the cyclical \( X_F(f) \) to be \( 0 \leq f \leq 1 \) (rather than \( |f| < \frac{1}{2} \) as before).

Now set \( f = \frac{k}{N} \), \( k = 0, 1, 2, ..., N - 1 \) and we have

\[
X_F\left(\frac{k}{N}\right) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi \frac{n k}{N}}, \quad k = 0, 1, 2, ..., N - 1
\]

and \( x(n) \) can be recovered from \( X_F(k/n) \) by the inverse transform

\[
x(n) = \sum_{k=0}^{N-1} X_F\left(\frac{k}{N}\right) \cdot e^{j2\pi \frac{n k}{N}}, \quad n = 0, 1, 2, ..., N - 1
\]

The last two equations define the Discrete Fourier Transform DFT. If \( N \) is chosen to be a power of 2, then Fast Fourier Transform (FFT) algorithms, which are computationally efficient, can be used to implement Eqs. of DFT.

### 14. Periodogram estimator of the psd

In order to define this estimator, let us introduce another estimator for the autocorrelation function:

\[
\hat{R}_{XX}(k) = \frac{1}{N} \sum_{i=0}^{N-1} X(i) \cdot X(i + k), \quad k = 0, 1, ..., N - 1
\]

(30)

This estimator differs from \( \hat{R}_{XX}(k) \) given in Equation (25) by the factor \( (N - k) / N \) that is

\[
\hat{R}_{XX}(k) = \frac{N - k}{N} \cdot \hat{R}_{XX}(k), \quad k = 0, 1, ..., N - 1
\]

(31)

We define
\[
d(n) = \begin{cases} 
1, & n = 0, 1, \ldots, N - 1 \\
0, & \text{elsewhere}
\end{cases}
\]  
(32)

then it can be shown that:

\[
\hat{R}_{XX}(k) = \frac{1}{N} \sum_{i=0}^{N-k-1} [d(n)X(n)][d(n+k)X(n+k)]_p, \quad k = 0, 1, \ldots, N - 1
\]  
(30)

The Fourier transform of \( \hat{R}_{XX}(k) \) is

\[
\hat{S}_{XX}(f) = \sum_{k=-\infty}^{\infty} \hat{R}_{XX}(k)e^{-j2\pi kf}
\]  
(31)

\[
= \ldots = \frac{1}{N} X_F^*(f) \cdot X_F(f) = \frac{1}{N} \left| X_F(f) \right|^2, \quad |f| < \frac{1}{2}
\]

where \( X_F(f) \) is the Fourier transform of the data sequence,

\[
X_F(f) = \sum_{n=-\infty}^{\infty} d(n) \cdot X(n) \cdot e^{-j2\pi nf} = \sum_{n=0}^{N-1} X(n) \cdot e^{-j2\pi nf}
\]  
(32)

The estimator \( \hat{S}_{XX}(f) \) defined in Equation (31) is called the periodogram of the data sequence \( X(0), X(1), \ldots, X(N - 1) \)

**Bias of the periodogram**

The periodogram estimator is a biased estimator of the psd, and we can evaluate the bias by calculating \( E\{\hat{S}_{XX}(f)\} \):

\[
E\{\hat{S}_{XX}(f)\} = E\left\{ \sum_{k=-\infty}^{\infty} \hat{R}_{XX}(k)e^{-j2\pi kf} \right\} = \ldots = \sum_{k=-\infty}^{\infty} q_N(k) \cdot R_{XX}(k) \cdot e^{-j2\pi kf}, \quad |f| < \frac{1}{2}
\]  
(33)

where

\[
q_N(k) = \frac{1}{N} \sum_{n=-\infty}^{\infty} d(n)d(n+k) = \begin{cases} 
1 - \frac{|k|}{N}, & |k| < N \\
0, & \text{elsewhere}
\end{cases}
\]  
(34)

The sum on the right-hand of Equation (33) is the Fourier transform of the product of \( q_N(k) \) and \( R_{XX}(k) \); thus,
\[ E\left\{ \hat{S}_{XX}(f) \right\} = \frac{1}{2} \int_{-1/2}^{1/2} S_{XX}(f)Q_N(f - \alpha) d\alpha \] (35.a)

where \( Q_N(f) \) is the Fourier transform of \( q_N(k) \), that is

\[ Q_N(f) = \frac{1}{N} \left[ \frac{\sin(\pi fN)}{\sin(\pi f)} \right]^2, \quad |f| < \frac{1}{2} \] (35.b)

Equations (34) and (35) show that \( \hat{S}_{XX}(f) \) is a biased estimator and the bias results from the truncation and the triangular window function (as a result of using \( 1/N \) instead of \( 1/(N - |k|) \)) in the definition of \( \hat{R}_{XX}(k) \).

Equations (35) show that \( S_{XX} \) is convolved with a \((\sin c)^2\) function in the frequency domain.

**Home work:** Plot of \( q_N(k) \) and its transform \( Q_N(f) \) to see the windowing effect.

The convolution given in Equation (35) has an “averaging” effect and it produces a “measured” estimate of \( S_{XX}(f) \). The effect of this smearing is known as spectral leakage. If \( S_{XX}(f) \) has two closely spaced spectral peaks, the periodogram estimate will smooth these peaks together.

**Home work:** Imagine a simulation scenario in order to present the spectral leakage.

However, as \( N \) is increased, \( Q_N(f) \to \delta(f) \) resulting in:

\[ E\left\{ \hat{S}_{XX}(f) \right\} = \frac{1}{2} \int_{-1/2}^{1/2} S_{XX}(\alpha) \cdot \delta(f - \alpha) d\alpha = S_{XX}(f) \] (36)

Thus \( \hat{S}_{XX}(f) \) is asymptotically unbiased.

**Variance of the periodogram**

In order to compute the variance of the periodogram we need to make assumptions about the psd that is being estimated. Assuming \( X(n) \) to be a zero-mean white Gaussian sequence with variance \( \sigma^2 \), we can compute the variance of the periodogram at \( f = p/N \), which is given by

\[^6\text{ (see for details Shanmugan, p.574)}\]
\[ \hat{S}_{XY} \left( \frac{p}{N} \right) = \frac{1}{N} \left| \sum_{n=0}^{N-1} X(n)e^{-j \frac{2\pi np}{N}} \right|^2 = \sigma^4, \quad p \neq 0, \left\lfloor \frac{N}{2} \right\rfloor \]
\[ = \ldots = \begin{cases} \sigma^4, & p \neq 0, \left\lfloor \frac{N}{2} \right\rfloor \\ 2\sigma^4, & p = 0, \left\lfloor \frac{N}{2} \right\rfloor \end{cases} \quad (37) \]

Equation (37) shows that, for most values of \( f \), \( \hat{S}_{XX} (f) \) has a variance of \( \sigma^4 \). Since we have assumed that \( S_{XX} (f) = \sigma^2 \), the normalized standard error, \( \varepsilon_r \), of the periodogram estimator is

\[ \varepsilon_r = \sqrt{\frac{\text{var} \left( \hat{S}_{XX} (f) \right)}{S_{XX} (f)}} = \frac{\sigma^2}{\sigma^2} = 100\% \quad (38) \]

This is a relatively poor estimator. In addition, the variance does not depend on the sample size \( N \). Unlike most estimation problems, where the variance of the estimator is reduced as the sample size is increased, the variance of the periodogram cannot be reduced by increasing the sample size. However, increasing the sample size, \( N \), will produce better resolution in the frequency domain.

**Home work:** Write a program to show/present the invariance of the periodogram with sample set size.

The periodogram estimator can be improved by averaging or smoothing. Two (weighted) averaging techniques are widely used:

- averaging of the estimates obtained from non-overlapping sections of the data;
- averaging the estimates in the frequency domain. Appropriate weighting (or window) functions are applied to control the bias and variance of the averaged estimators.

**Trade-off between bias and variance**

Since the periodogram may be viewed as the transform of \( \hat{R}_{XX} (k) \), let examine the estimator \( \hat{R}_{XX} (k) \). We estimate \( R_{XX} (k) \) for \( |k| = 0,1,\ldots,M \) and use the estimated values of \( R_{XX} (k) \) to form an estimate of \( S_{XX} (f) \).

When \( M << N \) and \( N >> 1 \) we obtain “good” estimators of \( R_{XX} (k) \) for \( |k| < M \). The bias of \( \hat{S}_{XX} (f) \) will be larger since the estimated autocorrelation function is truncated.
beyond \( k > M \). As we increase \( M \to N \), the bias of \( \hat{S}_{XX}(f) \) will become smaller, but the variance of the estimator of \( R_{XX}(k) \) will be larger as \( k \to N \) since fewer and fewer points are used in the estimator. Thus, for a finite sample size, we cannot completely control both bias and variance; when we attempt to reduce one, the other one increase.

When the sample size is very large, we can reduce both bias and variance to acceptable levels by using appropriate “windowing” (or averaging) techniques as explained in the following sections.

### Smoothing of Spectral Estimates

We can take the \( N \) measurements \( X(0), X(1), ..., X(N-1) \) divide them into \( n \) sections, each of which contains \( N/n \) points, form \( n \) different estimators of the psd and average the \( n \) estimators to form an averaged spectral estimator of the form

\[
\overline{S}_{XX}(f) = \frac{1}{n} \sum_{k=1}^{n} \hat{S}_{XX}(f)_k
\]

where \( \hat{S}_{XX}(f)_k \) is the spectral estimate obtained from the \( k \)-th segment of data. If we assume that the estimators \( \hat{S}_{XX}(f)_k \) are independent\(^7\), the variance of the averaged estimator will be reduced by the factor \( n \). However, since the fewer and fewer points are used to obtain the estimator \( \hat{S}_{XX}(f)_k \), the function \( Q_{N/n}(f) \) will be wider than \( Q_N(f) \) in the frequency domain, and thus, the bias will be larger.

A similar form of averaging can be also be done by averaging spectral estimates in the frequency domain. The averaging can be done simply as:

\[
\overline{S}_{XX}\left(\frac{p}{N}\right) = \frac{1}{2m+1} \sum_{i=-m}^{m} \hat{S}_{XX}\left(\frac{p+i}{m}\right)
\]

The Equation represents a running average in the frequency domain using a sliding rectangular window of width \((2m+1)\) points. The variance is reduced while the bias increases.

By using non-uniformly weighted window functions, we can control the trade-off between bias and variance and produce asymptotically unbiased and consistent estimators for the psd.

### Windowing procedure

Windowed or smoothed estimators of power spectral density functions are implemented using the following three steps:

**Step 1:** Computes \( \hat{R}_{XX}(k) \) using the estimator 1 or \( \hat{R}_{XX}(k) \) with the following DFT/FFT operations:

\(^7\) which is not completely true if the data segments are adjacent
1.a.) Pad $X(n)$ with $N$ zeroes and create a padded sequence $X_p(n)$ whose length is at least $2N$ points. The padding is necessary to avoid the circular (periodic) nature of the DFT, which cause error in convolution and correlation operations.

1.b). Compute

$$X_{p,F}\left(\frac{m}{2N}\right) = \sum_{n=0}^{2N-1} X_p(n) \cdot \exp\left(-\frac{j \cdot 2\pi \cdot m \cdot n}{2N}\right), \quad m = 0,1,2,\ldots,2N-1$$

1.c) Obtain:

$$\hat{R}_{XX}(k) = \frac{1}{N} \left[ \sum_{m=0}^{2N-1} X_{p,F}\left(\frac{m}{2N}\right)^2 \cdot \exp\left(\frac{j \cdot 2\pi \cdot k \cdot m}{2N}\right) \right], \quad k = 0,1,2,\ldots,N-1$$

1.d) Compute

$$\hat{R}_{XX}(k) = \begin{cases} \frac{N}{N-k} \hat{R}_{XX}(k), & k = 0,1,\ldots,N-1 \\ 0, & k > N-1 \end{cases}$$

$$\hat{R}_{XX}(-k) = \hat{R}_{XX}(k)$$

**Step 2:** Apply a weighted window and truncate $\hat{R}_{XX}(k)$ to $2M+1$ points:

$$\overline{R}_{XX}(k) = \hat{R}_{XX}(k) \cdot \hat{\lambda}(k), \quad |k| = 0,1,2,\ldots,M << N$$

where $\hat{\lambda}(k)$ is a window function and will be discussed later.

**Step 3:** Pad $\overline{R}_{XX}(k)$ with zeroes for $|k| > M$ and take the DFT to obtain the smoothed estimate $\overline{S}_{XX}(f)$ as

$$\overline{S}_{XX}\left(\frac{p}{N}\right) = \sum_{k=-(N-1)}^{N-1} \overline{R}_{XX}(k) \cdot \exp\left(-\frac{j \cdot 2\pi \cdot k \cdot p}{N}\right), \quad |p| = 0,1,2,\ldots,\frac{N}{2}$$

Multiplying with $\hat{\lambda}(k)$ and taking the Fourier transform in step 2 has the effect:

$$E\left[\overline{S}_{XX}(f)\right] = \int_{-0.5}^{0.5} S_{XX}(\alpha) \cdot w_m(f - \alpha) \cdot d\alpha$$

34
where \( w_m(f) \) is the Fourier transform of the window function \( \lambda(k) \). In order to reduce the bias (and spectral leakage), \( \lambda(k) \) should be chosen such that \( w_m(f) \) has most of its energy in a narrow “main lobe” and has smaller “side lobes”. This reduces the amount of “leakage”. Several window functions have been proposed and are presented below. It should be noted that most of these windows introduce a scale factor in the estimator of the power spectral density.

**Rectangular window**

\[
\lambda(k) = \begin{cases} 
1, & |k| \leq M, \quad M < N \\
0, & \text{otherwise}
\end{cases} \quad (47.a)
\]

\[
w_M(f) = \sin \left[ \left( M + \frac{1}{2} \right) \cdot 2\pi \cdot f \right] \sin(\pi f) \quad (47.b)
\]

Since \( w_M(f) \) is negative for some \( f \), this window might produce a negative estimate for psd. This window is seldom used in practice.

**Bartlett window**

\[
\lambda(k) = \begin{cases} 
1 - \frac{|k|}{M}, & |k| \leq M, \quad M < N \\
0, & \text{otherwise}
\end{cases} \quad (48.a)
\]

\[
w_M(f) = \frac{1}{M} \left[ \frac{\sin(\pi f \cdot M)}{\sin(\pi f)} \right]^2 \quad (48.b)
\]

Since \( w_M(f) \) is always positive, the estimated value is always positive. When \( M = N - 1 \), this window produces the unsmoothed periodogram estimator given in Equation (31).

**Blackman – Tukey window**

\[
\lambda(k) = \begin{cases} 
\frac{1}{2} \left[ 1 + \cos \left( \frac{\pi k}{M} \right) \right], & |k| \leq M, \quad M < N \\
0, & \text{otherwise}
\end{cases} \quad (49.a)
\]

\[
w_M(f) = \frac{1}{4} \left[ D_M \left( 2\pi \cdot f - \frac{\pi}{M} \right) + D_M \left( 2\pi \cdot f + \frac{\pi}{M} \right) \right] + \frac{1}{2} D_M (2\pi \cdot f) \quad (49.b)
\]

\[
D_M (2\pi \cdot f) = \sin \left[ \left( M + \frac{1}{2} \right) \cdot 2\pi \cdot f \right] \sin(\pi f)
\]
Parzen window

\[
\lambda(k) = \begin{cases} 
1 - 6 \left( \frac{k}{M} \right)^2 + 6 \left( \frac{|k|}{M} \right)^3, & |k| \leq \frac{M}{2} \\
2 \left( 1 - \frac{|k|}{M} \right)^3, & \frac{M}{2} < |k| \leq M < N \\
0, & \text{otherwise}
\end{cases}
\]  

(50.a)

\[
w_M(f) = \frac{3}{4M^3} \left[ \sin \left( \frac{\pi \cdot f \cdot M}{2} \right) \right]^4 \left[ \frac{1}{2 \sin(\pi f)} \right]^{\frac{1}{2}} \left[ 1 - \frac{2}{3} \sin^2(f) \right]
\]  

(50.b)

\(w_M(f)\) is nonnegative and these Parzen’s window produces a non-negative estimate of the psd.

**Bias and variances of smoothed estimators**

Expressions for the bias and variances of smoothed estimators are derived in [1]. The results are presented below. For the Bartlett, Blackman-Tukey and Parzen windows, the asymptotic bias and variance are given by:

\[
\text{Bias}\{S_{XX}(f)\} = C_1 \cdot \frac{S''_{XX}(f)}{M^2}
\]  

(51.a)

and

\[
\text{var}\{\widehat{S}_{XX}(f)\} = C_2 \cdot \frac{M}{N} \cdot S_{XX}^2(f)
\]  

(51.b)

where \(C_1, C_2\) constants with are different values for the different windows and \(S''_{XX}(f)\) is the second derivative of \(S_{XX}(f)\).

A common choice is \(M = 2\sqrt{N}\) and then as the sample size \(N \to \infty\), both the bias and variance of the smoothed estimators approach zero, as Eqs. (51) show. Thus, we have a family of asymptotically unbiased and consistent estimators for the psd.
Conclusions

The estimator is a random variable that will take on different values depending on the values of the measurements, whereas the estimate is a number.

The periodogram estimator can be improved by averaging or smoothing. Two (weighted) averaging techniques are widely used:

• averaging of the estimates obtained from non-overlapping sections of the data;
• averaging the estimates in the frequency domain. Appropriate weighting (or window) functions are applied to control the bias and variance of the averaged estimators.

References


N=5

\[ u = \begin{bmatrix} -0.9672 & -3.4153 & 0.8680 & 1.4523 & -1.6125 \end{bmatrix}; \]

\[ \text{rxx\_biased} = \begin{bmatrix} 3.6126 & -0.1485 & -1.4399 & 0.8205 & 0.3119 \end{bmatrix}; \]

\[ \text{rxx\_unbiased} = \begin{bmatrix} 3.6126 & -0.1856 & -2.3998 & 2.0513 & 1.5597 \end{bmatrix}; \]

\[ \text{rxx2\_biased} = \begin{bmatrix} 0.3119 & 0.8205 & -1.4399 & -0.1485 & 3.6126 & -0.1485 & -1.4399 & 0.8205 & 0.3119 \end{bmatrix}; \]

\[ \text{rxx2\_unbiased} = \begin{bmatrix} 1.5597 & 2.0513 & -2.3998 & -0.1856 & 3.6126 & -0.1856 & -2.3998 & 2.0513 & 1.5597 \end{bmatrix}; \]

\[ \text{rxx2\_mat\_unbiased} = \begin{bmatrix} 1.5597 & 2.0513 & -2.3998 & -0.1856 & 3.6126 & -0.1856 & -2.3998 & 2.0513 & 1.5597 \end{bmatrix}; \]

\[ \text{rxx2\_mat\_biased} = \begin{bmatrix} 0.3119 & 0.8205 & -1.4399 & -0.1485 & 3.6126 & -0.1485 & -1.4399 & 0.8205 & 0.3119 \end{bmatrix}; \]
Course 3 - Parameter estimation

1. Stochastic models of the signals

The term model is used for any hypothesis that may be applied to explain or describe the hidden laws that are supposed to govern or constrain the generation of physical data of interest.

The idea is that a time series \( u(n) \) consisting of highly correlated observations may be generated by applying a series of statistically independent “shocks” to a linear filter, as in Figure 1.

![Figure 1 – Generation of time series](image)

The shocks are random variables drawn from a fixed distribution that is usually assumed to be Gaussian with zero mean and constant variance. Such a series of random variables constitutes a purely random process, commonly referred to as white Gaussian noise.

In general, the time-domain description of the input-output relation for the stochastic model may be described as follows:

\[
\text{Present value of the model output} + \text{Linear combination of past values of model output} = \text{Linear combination of present and past values of model input}
\]

Such a description is referred to as a linear process. We may identify three popular types of linear stochastic models:

1). **AR model** (no past values of the model input are used);
2). **MA model** (no past values of the model output);
3). Mixed AR with MA = **ARMA model**.

---

8 The representation of a stochastic process by a model dates back to an idea of Yule (1927).
1.1. Autoregressive (AR) models

We say that the time series \( u(n), u(n-1), \ldots, u(n-M) \) represents the realization of an autoregressive process (AR) of order \( M \) if it satisfies the difference equation:

\[
u(n) + a_1^* u(n-1) + \ldots + a_M^* u(n-M) = v(n)\]

(1.a)

or, equivalently,

\[
\sum_{i=0}^{M} a_i^* u(n-i) = v(n), \quad a_0 = 1.
\]

(1.b)

or

\[
u(n) = -\sum_{i=1}^{M} a_i^* u(n-i) + v(n)
\]

(1.c)

where \( a_n \) are constants called AR parameters and “\(^*\)” means complex conjugation, \( v(n) \) means white noise with zero mean and variance \( \sigma_v^2 \).

The filter which generates an AR model has transfer function of form

\[
H(z) = \frac{U(z)}{V(z)} = \frac{1}{1 + a_1^* z^{-1} + \ldots + a_M^* z^{-M}}
\]

(1.d)

and is all-pole transfer function.

The matrix Eq. for an AR model is:

\[
u(n) = -a^T u + v(n)
\]

(1.e)

where

\[
a = [a_1, a_2, \ldots, a_M]^T
\]

(1.e.1)

and

\[
u = [u(n-1), a(n-2), \ldots, a(n-M)]^T
\]

(1.e.2)

For asymptotic stationarity of the AR process, we require that all poles of the filter in the AR model lie inside the unit circle in the \( z \)-plane.

The name “autoregressive” is coming from the fact that we can rewrite (1.c) as:

\[
u(n) = \sum_{i=1}^{M} w_i^* u(n-i) + v(n)
\]

So, the present value of the process, \( u(n) \) equals a finite linear combination of past values of the process, \( u(n-1), u(n-2), \ldots, u(n-M) \), plus an error term \( v(n) \). We can see now the reason for “autoregressive”.

A linear model

\[
y = \sum_{i=1}^{M} w_i^* x_i + v
\]

relating a dependent variable \( y \) to a set of independent variables \( x_1, x_2, \ldots, x_M \), plus an error term \( v \) is often referred to as a regression model, and \( y \) is said to be “regressed” on \( x_1, x_2, \ldots, x_M \). In
the first expression, the variable \( u(n) \) is regressed on previous values of itself; hence the term “autoregressive”.

### 1.2. Moving Average (MA) Models

The process \( u(n) \), produced at the filter output, is described by the difference equation:

\[
 u(n) = b_0 v(n) + b_1 v(n-1) + b_2 v(n-2) + ... + b_K v(n-K) \tag{2.a}
\]

or,

\[
 u(n) = v(n) + \sum_{i=1}^{K} b_i \cdot v(n-i) \tag{2.b}
\]

where \( b_1, b_2, ..., b_K \) are constants called the MA parameters and \( v(n) \) is white noise of zero mean and variance \( \sigma_v^2 \). The order of the MA process equals \( K \).

The transfer function is with all-zero (has only zeros..)

\[
 H(z) = \frac{U(z)}{V(z)} = 1 + b_1 z^{-1} + ... + b_K z^{-K} \tag{19.b}
\]

The matrix equation is

---

The term “moving average” is coming from the fact that if we are given a complete temporal realization of the white noise \( v(n) \), we may compute \( u(n) \) by constructing a weighted average of the sample values \( v(n), v(n-1), ..., v(n-K) \).

### 1.3. Autoregressive Moving Average (ARMA) models

Autoregressive-moving-average (ARMA) models are mathematical models of the persistence, or autocorrelation, in a time series. There are several possible reasons for fitting ARMA models to data.

- Modeling can contribute to understanding the physical system by revealing something about the physical process that builds persistence into the series.
- ARMA models can also be used to predict behavior of a time series from past values alone. Such a prediction can be used as a baseline to evaluate possible importance of other variables to the system.
- ARMA models are widely used for prediction of economic and industrial time series.
- ARMA models can also be used to remove persistence and to generate residual samples – time series with no dependence on past values. This operation, called prewhitening, is meant to remove persistence from the series.

The ARMA process is described by:
\[ u(n) + \sum_{i=1}^{M} a_i^* u(n-i) = v(n) + \sum_{j=1}^{K} b_j^* v(n-j) \]  
(3.a)

where \(a_1, a_2, ..., a_M\) and \(b_1, b_2, ..., b_K\) are called ARMA-parameters. The order of the ARMA process equals \((M+K)\). The AR and MA models are special cases of an ARMA model.

The transfer function of the filter is called a pole-zero transfer function

\[
H(z) = \frac{U(z)}{V(z)} = \frac{1 + b_1^* z^{-1} + \ldots + b_K^* z^{-K}}{1 + a_1^* z^{-1} + \ldots + a_M^* z^{-M}}, \quad M \geq K
\]  
(3.b)

In practice, the AR model is the most popular then MA or ARMA because the computation of the AR coefficients involves a system of linear equation in opposite with nonlinear equations in the case of MA or ARMA.

2. Would decomposition (1938)

Any stationary discrete-time stochastic MA process is decomposed into the sum of a general linear process and a predictable process, with these two processes being uncorrelated with each other.

**Theorem:** Any stationary discrete-time stochastic process \(x(n)\) may be expressed in the form:

\[ x(n) = u(n) + s(n) \]  
(4)

where:

1). \(u(n)\) and \(s(n)\) are uncorrelated processes;

2). \(u(n)\) is a general linear process represented by the MA model:

\[ u(n) = \sum_{j=0}^{\infty} b_j^* v(n-j), \quad b_0 = 1 \text{ and } \sum_{j=0}^{\infty} |b_j|^2 < \infty. \]

3). \(v(n)\) is uncorrelated white noise with \(s(n)\): \(E\{v(n) \cdot s^*(k)\} = 0, \quad \forall(n,k)\)

4). \(s(n)\) is a predictable process, that is, the process can be predicted from its own past with zero prediction variance.

3. Correlation function of an asymptotically stationary AR process

If multiplying both sides of Eq. (1.b) by \(u^*(n-l)\)

\[ \sum_{i=0}^{M} a_i^* \cdot u(n-i) = v(n) \]  
(1.b)

and then apply the expectation operator, we obtain:

\[ E\left\{ \sum_{i=0}^{M} a_i^* \cdot u(n-i) \cdot u^*(n-l) \right\} = E\{v(n) \cdot u^*(n-l)\} \]  
(5)

By interchanging the expectation and summation in the left-hand and by recognizing that the expectation \(E\{u(n-i) \cdot u^*(n-l)\}\) equals the autocorrelation function of the AR process for a
lag of $(l-i)$. Then we simplify the right-hand side by observing that the expectation

$$E\{v(n) \cdot u^*(n-l)\}$$

is zero for $l>0$ since $u(n-l)$ involves only samples of white noise, which are

uncorrelated with the white – noise sample $v(n)$. It results:

$$\sum_{i=0}^{M} a_i \cdot r_{uu}(l-i) = 0, \ l > 0, \ a_0 = 1$$

(6)

We thus see that the autocorrelation function of the AR process satisfies the difference equation:

$$r_{uu}(l) = w_1 \cdot r_{uu}(l-1) + w_2 \cdot r_{uu}(l-2) + ... + w_M \cdot r_{uu}(l-M), \ l > 0, a_0 = 1$$

(7)

where $w_i = -a_i, \ i = 1,2,...,M$. We may express the general solution of (7) as:

$$r_{uu}(m) = \sum_{i=0}^{M} c_i \cdot p_i^m$$

(8)

where $c_1,c_2,...,c_M$ are constants and $p_1,p_2,...,p_M$ are roots of the characteristic equation (7).

Note that when the AR model of the process satisfies the condition for asymptotic stationarity, $|p_i| < 1$ for all $i$, in which case the autocorrelation function $r_{uu}(m)$ approaches zero as the $\log(m)$ approaches infinity.

### 4. YULE-WALKER Equations

In order to uniquely define the AR model of order $M$ we need to specify two sets of model parameters:

1. The AR coefficients: $a_1,a_2,...,a_M$

2. The variance $\sigma_v^2$ of the white noise used as excitation.

By writing Eq. (6) for $l = 1,2,...,M$ we get a set of $M$ simultaneous equations with the values of $r(0),r(1),...,r(M) = r_{uu}(0),r_{uu}(1),...,r_{uu}(M)$, that means the autocorrelation function of the AR process as the known quantities and the AR parameters $a_1,a_2,...,a_M$ as the unknowns. The set of transformations is

$$\sum_{i=0}^{M} a_i \cdot r_{uu}(l-i) = 0, \ l > 0, \ a_0 = 1$$

(6)

$$\begin{align*}
    r(1) + a_1^* \cdot r(0) + ... + a_M^* \cdot r(-(M-1)) = 0 \\
    r(2) + a_1^* \cdot r(1) + ... + a_M^* \cdot r(-(M-2)) = 0 \\
    \vdots \\
    r(M) + a_1^* \cdot r(M-1) + ... + a_M^* \cdot r(0) = 0
\end{align*}$$

(9.a)
and the final general form is

\[
\begin{bmatrix}
  r(0) & r(-1) & \ldots & r(-(M-1)) \\
  r(1) & r(0) & \ldots & r(-(M-2)) \\
  \vdots & \vdots & \ddots & \vdots \\
  r(M-1) & r(M-2) & \ldots & r(0)
\end{bmatrix}
\begin{bmatrix}
a_1^* \\
a_2^* \\
\vdots \\
a_M^*
\end{bmatrix}
= \begin{bmatrix}
  -r(1) \\
  -r(2) \\
  \vdots \\
  -r(M)
\end{bmatrix}
\tag{9.b}
\]

\[
\begin{bmatrix}
r(0) & r(-1) & \ldots & r(-(M-1)) \\
r(1) & r(0) & \ldots & r(-(M-2)) \\
\vdots & \vdots & \ddots & \vdots \\
r(M-1) & r(M-2) & \ldots & r(0)
\end{bmatrix}
\begin{bmatrix}
a_1^* \\
a_2^* \\
\vdots \\
a_M^*
\end{bmatrix}
= \begin{bmatrix}
  r(1) \\
r(2) \\
\vdots \\
r(M)
\end{bmatrix}
\tag{9.c}
\]

\[
\begin{bmatrix}
r(0) & r^*(-1) & \ldots & r^*(-(M-1)) \\
r^*(1) & r(0) & \ldots & r^*(-(M-2)) \\
\vdots & \vdots & \ddots & \vdots \\
r^*(M-1) & r^*(M-2) & \ldots & r(0)
\end{bmatrix}
\begin{bmatrix}
a_1^* \\
a_2^* \\
\vdots \\
a_M^*
\end{bmatrix}
= \begin{bmatrix}
r^*(1) \\
r^*(2) \\
\vdots \\
r^*(M)
\end{bmatrix}
\tag{9.d}
\]

\[
\begin{bmatrix}
r(0) & r(1) & \ldots & r(M-1) \\
r^*(1) & r(0) & \ldots & r(M-2) \\
\vdots & \vdots & \ddots & \vdots \\
r^*(M-1) & r^*(M-2) & \ldots & r(0)
\end{bmatrix}
\begin{bmatrix}
w_1^* \\
w_2^* \\
\vdots \\
w_M^*
\end{bmatrix}
= \begin{bmatrix}
r^*(1) \\
r^*(2) \\
\vdots \\
r^*(M)
\end{bmatrix}
\tag{9.e}
\]

where \( w_k = -a_k \). The set of equations (9) is called the Yule-Walker equations.

We may express the Yule-Walker equations in the compact matrix form:

\[
R_{uu} \cdot w = r_{uu}
\tag{9.a}
\]

with the solution

\[
w = R^{-1}_{uu} \cdot r_{uu}
\tag{10}
\]

were

\[
w = [w_1 \quad w_2 \quad \ldots \quad w_M]^T = -[a_1 \quad a_2 \quad \ldots \quad a_M]^T
\tag{10.a}
\]

So, we can compute the AR coefficients, \( a_k = -w_k \). In other words, there is a unique relationship between the coefficients \( a_1, a_2, \ldots, a_M \) of the AR model and of the AR process \( u(n) \) as shown by

\[
\{a_1, a_2, \ldots, a_M\} \leftrightarrow \{\rho_1, \rho_2, \ldots, \rho_M\}
\tag{11}
\]

where the \( k \)-th correlation coefficient is defined by

\[
\rho_i = \frac{r(i)}{r(0)}, i = 1, 2, \ldots, M
\tag{11.a}
**Example:** Let \( u(n) \) be a first order AR process \( u(n) + a_1 \cdot u(n-1) = v(n) \), with the variance of the noise \( \sigma_v^2 = 1 \). The Yule-Walker equations are \( [r_{uu}(0)] \cdot [a_1^*] = [r_{uu}^*(1)] \) and the parameter is then \( a_1^* = -\frac{r_{uu}^*(1)}{r_{uu}(0)} \).

\[
\begin{align*}
\text{5. Variance of the white noise}\end{align*}
\]

For \( l=0 \), we find that the expectation on the right-hand side of Eq (5)

\[
E\left\{ \sum_{i=0}^{M} a_i^* \cdot u(n-i) \cdot u^*(n-l) \right\} = E\{v(n) \cdot u^*(n-l)\}
\]

assumes the special form (see also Eq. 1.a):

\[
E\{v(n) \cdot u^*(n)\} = E\{v(n) \cdot v^*(0)\} = r_{vv}(0) = \sigma_v^2
\]

where \( \sigma_v^2 \) is the variance of the zero-mean white noise, \( v(n) \). Setting \( l=0 \) in Eq (5) and performing a complex conjugation on both sides, we get the formula:

\[
\sigma_v^2 = \sum_{i=0}^{M} a_i^* \cdot r_{uu}(i)
\]

with \( a_0 = 1 \), for the variance of the white noise. Hence, given the autocorrelation \( r_{uu}(0) \), \( r_{uu}(1) \), ..., \( r_{uu}(M) \), we may determine the white-noise variance \( \sigma_v^2 \).

\[
\begin{align*}
\text{6. Selecting the model order}\end{align*}
\]

The representation of a stochastic process by a linear model may be used for synthesis or analysis. In synthesis, we generate a desired time series by assigning a prescribed set of values to the parameters of the model and feeding it with white noise of zero mean and prescribed variance.

In analysis, we estimate the parameters of the model by processing a given time series of finite length. Except the values of the parameters we need the order of the model. In practice, there are two criteria:


Let \( u(i) = u_i, i = 1, 2, ..., N \) denote the data obtained by \( N \) independent observations of a stationary discrete-time process and \( g(u_i) \) denote the probability density function of \( u_i \). Let \( f_{u_i}(u_i / \hat{\theta}) \) the conditional probability density function of \( u_i \), given \( \hat{\theta} \), the estimated vector of parameters that model the process. Let \( M \) be the model order, so that:
\[ \hat{\theta} = [\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_M]^T \] (14)

The criterion of AKAIKE selects the model for which the quantity

\[ AIC(M) = -2L(\hat{\theta}) + 2M \] (15)

is a minimum. The function

\[ L(M) = \max_{i=1}^{N} \ln f_y(u_i / \hat{\theta}) \] (15.a)

The first term is the logarithm of the maximum – likelihood estimates of the parameters in the model. The second term, \(2M\), represent a model complexity penalty.

A practical criterion: A different way of identifying ARMA models is by trial and error and use of a goodness-of-fit statistic. Akaike’s Final Prediction Error (FPE) and Information Theoretic Criterion (AIC) are two closely related alternative statistical measures of goodness-of-fit of an ARMA(p,q) model.

Goodness of fit might be expected to be measured by some function of the variance of the model residuals: the fit improves as the residuals become smaller. Both the FPE and AIC are functions of the variance of residuals.

Another factor that must be considered, however, is the number of estimated parameters \(n = p + q\). This is so because by including enough parameters we can force a model to perfectly fit any data set.

Measures of goodness of fit must therefore compensate for the artificial improvement in fit that comes from increasing complexity of model structure.

The FPE is given by

\[ FPE = \frac{1 + n/N}{1 - n/N} \cdot V \] (15.b)

where \(V\) is the variance of model residuals, \(N\) is the length of the time series. In application, the FPE is computed for various candidate models, and the model with the lowest FPE is selected as the best-fit model.

The AIC (Akaike Information Criterion) is another widely used goodness-of-fit measure, and is given by

\[ AIC = \log V + \frac{2n}{N} \] (15.c)

As with the FPE, the best-fit model has minimum value of AIC.

Neither the FPE nor the AIC directly addresses the question whether the model residuals are white noise. A strategy for model identification by the FPE is to iteratively fit several different models and find the model that gives approximately minimum FPE AND does a good job of producing random residuals. The checking of residuals is described in the next section.


The criterion is

\[ MDL(M) = -L(\hat{\theta}) + \frac{1}{2} m \cdot \ln(N) \] (15.d)
where $N$ is the sample size, i.e., the numbers of observations. The attributes of MDL criterion are:

a). The model permits the shortest encoding of the observed data and captures all of the learnable properties of the observed data in the best possible manner.
b). The MDL criterion is a consistent model-order estimator, in the sense that it converges to the true model order as the sample size increases.
c). The model is optimal in the context of linear regression problems as well as ARMA models.

---

**Steps in modeling**

ARMA modeling proceeds by a series of well-defined steps.

1). The first step is to identify the model. Identification consists of specifying the appropriate structure (AR, MA or ARMA) and order of model.

Identification is sometimes done by looking at plots of the acf. Each structure has a pattern which could guide experimented people to guess the right structure of the model. The classical method of model identification as described by Box and Jenkins (1970) is judge the appropriate model structure and order from the appearance of the plotted acf and partial autocorrelation functions. As an example let look to the pattern from figure 1, which show the acf of AR(3) and MA(3) models. The identification of ARMA models from the acf and pacf plots is difficult and requires much experience for all but the simplest models.

![Figure 2 – Acf patterns for AR and MA model](image)

Sometimes identification is done by an automated iterative procedure -- fitting many different possible model structures and orders and using a goodness-of-fit statistic to select the best model.

2). The second step is to estimate the coefficients of the model. Coefficients of AR models can be estimated by least-squares regression. Estimation of parameters of MA and ARMA models usually requires a more complicated iteration procedure (Chatfield 1975). In practice, estimation is fairly transparent to the user, as it accomplished automatically by a computer.
program with little or no user interaction.

3). The third step is to **check the model**. This step is also called diagnostic checking, or verification (Anderson 1976). Two important elements of checking are to ensure that the residuals of the model are random, and to ensure that the estimated parameters are statistically significant. Usually the fitting process is guided by the principal of parsimony, by which the best model is the simplest possible model that adequately describes the data. The simplest model is the model with the fewest parameters.

**Checking the model – are the residuals random?** A key question in ARMA modeling is does the model effectively describe the persistence? If so, the model residuals should be random—or uncorrelated in time—and the autocorrelation function (acf) of residuals should be zero at all lags except lag zero. Of course, for sample series, the acf will not be exactly zero, but should fluctuate close to zero.

The acf of the residuals can be examined in two ways. First, the acf can be scanned to see if any individual coefficients fall outside some specified confidence interval around zero. Approximate confidence intervals can be computed. The correlogram of the true residuals (which are unknown) is such that $r_k$ is normally distributed with mean

$$E[r(k)] = 0$$  \hspace{1cm} (1)

and variance

$$\text{var}(r(k)) = \frac{1}{N}$$  \hspace{1cm} (2)

where $r_k$ is the autocorrelation coefficient of the ARMA residuals at lag $k$.

**Checking the model – are the estimated coefficients significantly different from zero?** Besides the randomness of the residuals, we are concerned with the statistical significance of the model coefficients. The estimated coefficients should be significantly different than zero. If not, the model should probably be simplified, say, by reducing the model order. For example, an AR(2) model for which the second-order coefficient is not significantly different from zero might better be fit with an AR(1) model. Significance of the ARMA coefficients can be evaluated from the approximate variances of the parameters. The estimated parameters should be compared with their standard deviations to check that the parameters are “significantly” different from zero.

**Practical vs statistical significance of persistence**

Note that the variance of the estimated autoregressive coefficient for an AR(1) model is inversely proportional to the sample length. For long time series (e.g., many hundreds of observations), ARMA modeling may yield a model whose estimated parameters are significantly different from zero but very small. The persistence described by such a model might actually account for a negligibly small percentage of the variance of the original time series.

A measure of the practical significance of the autocorrelation, or persistence, in a time series is the percentage of the series variance that is reduced by fitting the series to an ARMA model. If the variance of the model residuals is much smaller than the variance of the original series, the ARMA model accounts for a large fraction of the variance, and a large part of the
variance of the series is “due to persistence.” In contrast, if the variance of the residuals is almost as large as the original variance, then little variance has been removed by ARMA modeling, and the variance due to persistence is small.

A simple measure of fractional variance due to persistence:

\[ R_p^2 = 1 - \frac{\text{var}(e)}{\text{var}(u)} \]  

where \( \text{var}(u) \) is the variance of the original series, and \( \text{var}(e) \) is the variance of the residuals of the ARMA model. Whether any given value of \( R_p^2 \) is practically significant is a matter of subjective judgment and depends on the problem. For example, in a time series of tree-ring index, \( R_p^2 = 0.5 \) would likely be considered practically significant, as half the variance of the original time series is explained by the modeled persistence. On the other hand, \( R_p^2 = 0.01 \) might well be dismissed as practically insignificant.

**Extension to nonstationary time series**

ARMA modeling assumes the time series is weakly stationarity. With the appropriate modification, nonstationary series can also be studied with ARMA modeling. Periodic time series is a special case. An example of a periodic series is a monthly time series of air temperature, which has a superposed annual cycle. The mean is clearly nonstationarity in that it varies in a regular pattern from month to month. One way of handling such a series with ARMA modeling is to remove the annual cycle – for example, by transforming the monthly series to departures from the long-term monthly means. Another way is by applying periodic ARMA models, in which separate parameters are simultaneously estimated for each month of the year.

Another way of handling nonstationarity is by autoregressive-integrated-moving-average (ARIMA) models. ARIMA modeling essentially begin with first-differencing to convert the original, nonstationary, series to a stationary series. A series with shifts in level over time is amenable to the approach. The nonstationarity in a random walk can effectively be removed by first-differencing.

**Remark**

Estimating the moving average (MA) parameters is usually more difficult than estimating the autoregressive (AR) part, especially if the zeros are located close to the unit circle. Four linear methods for MA parameter estimation are presented and discussed by (Niclas and Stoica, 2006)

**7. Parametric methods of power spectrum estimation**

Depending on the specified model adopted, we may identify three different parametric approaches for spectrum estimation. The basic idea is presented in Fig. 2.
The method is part of a field called *model identification*. In this class of parametric methods, a rational function or a polynomial in $e^{-j\omega}$ is assumed for the transfer function of the model, and a white-noise source is used to drive the model. The power spectrum of the resulting model provides the desired spectrum estimate.

The resulting power spectra measured at the outputs of these models are referred to as AR, MA and ARMA spectra, respectively.

The input-output relation is

$$S_0(\omega) = |H(e^{j\omega})|^2 \cdot S_I(\omega)$$

where $H(e^{j\omega})$ is the frequency response of the filter. The frequency response $H(e^{j\omega})$ equals the discrete transfer function $H(z)$ evaluated on the unit circle in the $z$-plane.

The important feature of this result is that the value of the output spectral density at angular frequency $\omega$ depends purely on the squared amplitude response of the filter and the input power spectral density at the same angular frequency $\omega$.

If the input is white noise with power $\sigma_v^2$, then

$$S_0(\omega) = |H(e^{j\omega})|^2 \cdot \sigma_v^2$$

The problem thus becomes one of estimating the model parameters of the stochastic process under study. Because the parameters of the model are also the parameters of the transfer function of the filter, such an approach to power spectrum estimation may indeed be viewed as a problem in model (system) identification.

**Example:** See the program “Lab_33.m”.

**Conclusions**

Simple models of stochastic signals could be considered by defining a filter. The input of the filter is drawn by white noise and the output of the filter is the desired signal.

In practice, the AR model is more popular than MA or ARMA because the computation of the AR coefficients involves a system of linear equations in opposite with nonlinear equations in the case of MA or ARMA.

The equations of models are somewhat simpler if the time series is first reduced to
zero-mean by subtracting the sample mean. Therefore, we will work with the mean-adjusted series

\[ u_{\text{new}}(n) = u(n) - \overline{u(n)}, \quad n = 1, 2, \ldots \]  

(1)

where \( u(n) \) is the original time series.

An AR model expresses a time series as a linear function of its past values. The order of the AR model tells how many lagged past values are included.

The noise also goes by various other names: the error, the random-shock, and the residual. The residuals \( e_t \) are assumed to be random in time (not autocorrelated), and normally distributed.

The autoregressive model includes lagged terms on the time series itself, and that the moving average model includes lagged terms on the noise or residuals.

There are criteria for the selection of the model order, as information based criterion of Akaike and minimum length description (MDL) of Rissanen.

For an AR model there are direct linear relations to estimate the parameters of the model, by using the set of Yule-Walker equations.

Model a signal is equivalent with estimation of coefficients of some filters. From here the name of system (the filter) identification.

References


Figure 1 - Results of Lab_31.m – Conversion of models from MA to AR
Figure 2 - Results of Lab_32.m – AR model estimation

Figure 4 - Results of Lab_33.m – AR Spectrum estimation
Figure 5 - Analysis of ‘a’ – voice signal with AR model
Figure 6 - Analysis of 'u' – voice signal
LINEAR OPTIMAL FILTERING
(Wiener Filtering)

1. Statement of the problem

Consider a linear discrete-time filter as it is presented in Fig. 1. The filter input consists of a time series \( u(0), u(1), u(2), \ldots \); the filter is characterized by the impulse response \( w = [w_0, w_1, w_2, \ldots] \); the output of the filter is \( y(n) \). The job is to design a filter in order to provide an estimate of a desired response, \( d(n) \), as close as possible, in some statistical sense.

The estimation error, \( e(n) \), is defined as the difference between the desired response \( d(n) \) and the filter output \( y(n) \). The design of the filter is made by optimizing a cost function as, for example, the mean-square value of the estimation error. This is a statistical optimization problem.

The mathematical solution can be developed by following entirely two different approaches that are complementary: 1). Principle of orthogonality and 2). The error-performance surface.

2. Filter design by the principle of orthogonality

With reference of the statistical filtering problem described in the previous section, the filter output at a discrete time is defined by the linear convolution sum:

\[
y(n) = w \ast u = \sum_{i=0}^{\infty} w_i^* \cdot u(n-i), \quad n = 0, 1, 2, \ldots
\]

We assume that the filter input and the desired response are single realizations of jointly wide-sense stationary stochastic processes, both with zero mean.

The estimation of \( d(n) \) is naturally accompanied by an error, which is defined by the difference

\[
e(n) = d(n) - y(n)
\]
The estimation error \( e(n) \) is the sample value of a random variable. To optimize the filter design, we choose to minimize the mean-square value of \( e(n) \). We thus define the cost function as the mean-square-error (MSE):

\[
J = E[e(n) \cdot e^*(n)] = E\left[|e(n)|^2\right]
\]

where \( E \) denotes the statistical expectation operator. For complex input data, the filter coefficients are, in general, complex, too. If the filter coefficient \( w_k \) is defined as

\[
w_k = a_k + j \cdot b_k, \quad k = 0,1,2,...
\]

We define a gradient operator as

\[
\nabla_k = \frac{\partial J}{\partial a_k} + j \cdot \frac{\partial J}{\partial b_k}, \quad k = 0,1,2,...
\]

Applying the gradient operator to the cost function \( J \), a multidimensional complex gradient vector \( \nabla J \) is obtained as

\[
\nabla_k J = \frac{\partial J}{\partial a_k} + j \cdot \frac{\partial J}{\partial b_k}, \quad k = 0,1,2,...
\]

where \( J \) is considered as real value.

The minimization of the cost function \( J \) requires that all elements of the gradient vector \( \nabla J \) to be simultaneously equal to zero:

\[
\nabla_k J = 0, \quad k = 0,1,2,...
\]

Under this set of conditions, the filter is said to be optimum in the mean-square-error sense.

According to Eq. (3), the cost function \( J \) is a scalar that is independent of time \( n \); by substituting the first derivatives of that equation, we get:

\[
\nabla_k J = E\left[\frac{\partial}{\partial a_k}(e(n) \cdot e^*(n)) + j \cdot \frac{\partial}{\partial b_k}(e(n) \cdot e^*(n))\right] =
\]

\[
= E\left[\frac{\partial e(n)}{\partial a_k} e^*(n) + \frac{\partial e^*(n)}{\partial a_k} e(n) + j \cdot \frac{\partial e(n)}{\partial b_k} e^*(n) + j \cdot \frac{\partial e^*(n)}{\partial b_k} e(n)\right], \quad k = 0,1,2,...
\]

Based on the relations

\[
e(n) = d(n) - y(n) = d(n) - \sum_{k=0}^{\infty} w_k^* \cdot u(n-k) = d(n) - \sum_{k=0}^{\infty} (a_k + j b_k)^* \cdot u(n-k)
\]

\[
e^*(n) = [d(n) - y(n)]^* = d(n) - \sum_{k=0}^{\infty} w_k \cdot u^*(n-k) = d(n) - \sum_{k=0}^{\infty} (a_k + j b_k) \cdot u^*(n-k)
\]

the following equations are obtained:
\[
\frac{\partial e(n)}{\partial a_k} = -u(n-k), \quad \frac{\partial e^*(n)}{\partial a_k} = -u^*(n-k)
\] (10.a)

\[
\frac{\partial e(n)}{\partial b_k} = j\mu^*(n-k), \quad \frac{\partial e^*(n)}{\partial b_k} = -j\mu^*(n-k)
\] (10.b)

By substituting these partial derivatives into Eq. (8) we obtain

\[
\nabla_k J = -2 \cdot E[u(n-k) \cdot e^*(n)] \quad k = 0,1,2,...
\] (11)

Let \(e_0\) denote the value of the estimator error that results when the filter operates in its optimum condition. Then:

\[
\nabla_k J = 0 \rightarrow E[u(n-k) \cdot e_0^*(n)] = r_{ue0}(k) = 0, \quad k = 0,1,2,...
\] (12)

So, the necessary and sufficient condition for the cost function \(J\) to reach its minimum value is for the corresponding value of the estimation error \(e_0(n)\) to be orthogonal to each input sample that enters into estimation of the desired response at time \(n\).

The above statement constitutes the principle of orthogonality. It provides the mathematical basis of a procedure for testing whether the linear filter is operating in its optimum condition or not.

**Note 1 (Corollary to the principle of orthogonality):** Let us compute the correlation between the filter output \(y(n)\) and the estimation error :

\[
r_{ye}(0) = E[y(n) \cdot e^*(n)] = E\left[\sum_{k=0}^{\infty} w_k^* \cdot u(n-k) \cdot e^*(n)\right] = \sum_{k=0}^{\infty} w_k^* \cdot E[u(n-k) \cdot e^*(n)]
\] (13)

Let \(y_0(n)\) denote the output produced by the filter optimized in the mean-square-error sense and \(e_0(n)\) denoting the corresponding estimation error. Hence, the principle of orthogonality gets the result:

\[
\nabla_k J = 0 \rightarrow E[y_0(n) \cdot e_0^*(n)] = r_{y0e0}(0) = 0
\] (14)

So, when the filter operates in its optimum condition, the estimate of the desired response, defined by the filter output \(y_0(n)\) and the corresponding estimation error \(e_0(n)\), are also orthogonal to each other.

**Note 2 (geometric interpretation):** The vectors representing the estimation error is normal (i.e., perpendicular) to the vector representing the filter output, as in Fig. 2., when the filter is optimal.
3. Wiener-Hopf Equations

The principle of orthogonality specifies the necessary and sufficient condition for the optimum operation of the filter. We may reformulate this condition by substituting Eqs. (1) and (2) into (12):

\[
E \left[ u(n-k) \cdot \left( d^*(n) - \sum_{i=0}^{\infty} w_{oi} \cdot u^*(n-i) \right) \right] = 0, \quad k = 0,1,2,\ldots
\]

(15)

where \( w_{oi} \) is the \( i \)-th coefficient in the impulse response of the optimum filter. Expanding this equation and rearranging terms, we get:

\[
\sum_{i=0}^{\infty} w_{oi} \cdot E[u(n-k) \cdot u^*(n-i)] = E[u(n-k) \cdot d^*(n)] \quad k = 0,1,2,\ldots
\]

(16.a)

or

\[
\sum_{i=0}^{\infty} w_{oi} \cdot r_{uu}(i-k) = p(-k) = r_{ud}(-k), \quad k = 0,1,2,\ldots
\]

(16.b)

where

\[
r_{uu}(i-k) = E[u(n-k) \cdot u^*(n-i)] \quad k = 0,1,2,\ldots
\]

(16.c)

is the autocorrelation function of the filter input \( u \) for a lag of \( (i-k) \), and

\[
r_{ud}(-k) = p(-k) = E[u(n-k) \cdot d^*(n)] \quad k = 0,1,2,\ldots
\]

(16.d)

is the correlation function between the filter input \( u \) and the desired response \( d \) for a lag of \(-k\). Accordingly, the necessary and sufficient condition for the optimality of the filter is:

\[
\sum_{i=0}^{\infty} w_{oi} \cdot r_{uu}(i-k) = p(-k) = r_{ud}(-k), \quad k = 0,1,2,\ldots
\]

(17.a)

or

\[
\sum_{i=0}^{\infty} w_{oi} \cdot r_{uu}(i-k) = p(-k) = r_{du}(k), \quad k = 0,1,2,\ldots
\]

(17.b)
The system of equations (17) defines the optimum filter coefficients, in terms of two correlation functions. These equations are called Wiener-Hopf equations.

The matrix form of the Wiener-Hopf equations is considered now. Let $R$ denote the $M$-by-$M$ correlation matrix of the input vector

$$
\mathbf{u}(n)_{M \times 1} = [u(n) \ u(n-1) \ ... \ u(n-(M-1))]^T
$$

(18)

so

$$
R_{uu} = E[\mathbf{u}(n) \cdot \mathbf{u}^H(n)]
$$

(19)

Let $\mathbf{p} = \mathbf{r}_{ud}$ denote the $M$-by-1 the cross-correlation vector between the input vector $\mathbf{u}(n)$ and the desired response $d(n)$:

$$
\mathbf{p} = \mathbf{r}_{ud} = E[\mathbf{u}(n) \cdot d^*(n)] = [p(0) \ p(-1) \ ... \ p(-(M-1))]^T
$$

(20)

The matrix form of the Wiener-Hopf equation is then:

$$
[\mathbf{R}_{uu} \cdot \mathbf{w}_o = \mathbf{r}_{ud}]
$$

(21)

where

$$
\mathbf{w}_o = [w_{o,0} \ w_{o,1} \ ... \ w_{o,M-1}]^T
$$

(21.a)

denote the $M$-by-1 optimum – impulse-response coefficients of the optimal filter. The solution of the equation (21) is

$$
\mathbf{w}_o = \text{inv}(\mathbf{R}_{uu}) \cdot \mathbf{r}_{ud} = \mathbf{R}_{uu}^{-1} \cdot \mathbf{r}_{ud}
$$

(22)

4. Solution of the Wiener-Hopf equations for Linear Transversal Filters (FIR)

The transversal filter, or FIR, has an impulse response defined by the finite set of weights $w_0,w_1,...,w_{M-1}$, as in Fig. 3.. The Wiener-Hopf equations reduce to the system of $M$ simultaneous equations:

$$
\sum_{i=0}^{M-1} w_{o,i} \cdot r_{uu}(i-k) = p(-k) = r_{ud}(-k), \ k = 0,1,2,\ldots,M-1
$$

(22)

where $\{w_{o,i}\}$ are the optimum values of the tap weights of the filter. The structure of FIR filter is presented in Fig. 3. The transfer function is

$$
H(z) = \frac{Y(z)}{U(z)} = w_0^* + w_1^* \cdot z^{-1} + w_2^* \cdot z^{-2} + \ldots + w_{M-1}^* \cdot z^{-(M-1)}
$$

(23)

and the discrete-time equation is:
5. Minimum - Mean - Square Error (MMSE)

When the linear discrete-time operates in its optimum condition, the equation error takes the form

\[ e_o(n) = d(n) - y_o(n) \]  
\[ d(n) = e_o(n) + y_o(n) \]

and the minimum value of the error criterion is

\[ J_{min} = E\left[ e_o(n)^2 \right] \]

Evaluating the mean-square values on both sides of Eq. (4.b) we have:

\[ \sigma_d^2 = J_{min} + \sigma_y^2 \]

and the minimum mean-square error is

\[ J_{min} = \sigma_d^2 - \sigma_y^2 \]

where both signals are zero mean considered.

The output of the optimal filter is

\[ y_o(n) = \sum_{k=0}^{M-1} w_{o,k}^* u(n-k) = w_o^H \cdot u(n) \]
We assume that the input signal vector \( u(n) \) has zero mean, which makes the estimate \( y(n) \) have zero mean, too. The variance of the estimation is

\[
\sigma_y^2 = E\left[|y(n)|^2\right] = E\left[\left(w_o^H \cdot u(n)\right)\left(u^H(n) \cdot w_o\right)\right] = w_o^H \cdot E\left[u(n)\right] \cdot u^H(n) \cdot w_o = w_o^H \cdot R_{uu} \cdot w_o
\]

(30)

By using the WH equations in matrix form we obtain

\[
\sigma_y^2 = (R_{uu}^{-1} \cdot p)^H \cdot R_{uu} \cdot R_{uu}^{-1} \cdot p = p^H \cdot R_{uu}^{-1} \cdot p
\]

(31.a)

or

\[
\sigma_y^2 = p^H \cdot R_{uu}^{-1} \cdot p = r_{ud}^H \cdot R_{uu}^{-1} \cdot r_{ud}
\]

(31.b)

To evaluate the minimum mean-square error produced by the transversal filter, and according with Eq. (28), we have

\[
J_{\text{min}} = \sigma_d^2 - \sigma_y^2 = \sigma_d^2 - w_o^H \cdot R_{uu} \cdot w_o =
\]

\[
= \sigma_d^2 - p^H \cdot R_{uu}^{-1} \cdot p = \sigma_d^2 - r_{ud}^H \cdot R_{uu}^{-1} \cdot r_{ud} = \sigma_d^2 - r_{ud}^H \cdot w_o
\]

(32)

**Example 1:** Consider a Wiener filtering problem, having a correlation matrix \( R_{uu} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix} \), the crosscorrelation vector between the tap-input vector \( u(n) \) and the desired response \( d(n) \) is \( p = r_{ud} = [0.5 \ 0.25]^T \). Compute the tap-weights of the Wiener filter (FIR implementation) and the minimum mean square error.

**Solution:** The solution of the Wiener-Hopf equation is

\[
w_o = R_{uu}^{-1} \cdot p = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0.5 \\ 0.25 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}
\]

and the minimum MSE is

\[
J_{\text{min}} = \sigma_d^2 - p^H \cdot w_o = \sigma_d^2 - [0.5 \ 0.25] \cdot \begin{bmatrix} 0.5 \\ 0 \end{bmatrix} = \sigma_d^2 - 0.25
\]

6. **Multiple linear regression model**

The configuration of the FIR filter, in the sense of length, \( M \), implies the minimum value of the error criterion, \( J_{\text{min}} \).

Given the possibility of adjusting the length of the transversal filter, as a design parameter, it is expected to change the value of the \( J_{\text{min}} \).

The following hypotheses are considered valid:
1. The model of the signal is linear;
2. The observable (measurable) data are noisy.
3. The noise is additive and white.

The (signal) model of Fig. 3 is called the **multiple linear regression model** and is described by:

\[
d(n) = \mathbf{a}^H \cdot \mathbf{u}_m(n) + \nu(n) \tag{33.a}
\]

\[
d(n) = \sum_{k=1}^{m} a^*(k) \cdot u(n-k+1) + \nu(n) = y(n) + \nu(n) \tag{33.b}
\]

where \( \mathbf{a} = [a_1, a_2, \ldots, a_m]^T \) denotes an unknown parameter vector of the model, of size \( m \), \( \mathbf{u}_m(n) = [u(n), u(n-1), \ldots, u(n-m+1)]^T \) denotes the input vector or **regressor vector**, and \( \nu(n) \) accounts for the additive white noise. The noise \( \nu(n) \) is statistically independent of \( u(n) \).

![Multiple Linear Regression Model](image)

**Figure 3 – Multiple Linear Regression Model**

Let \( \sigma_v^2 \) denote the variance of the noise \( \nu(n) \). Then the variance of the **observable** data \( d(n) \) supplying the desired response is given by:

\[
\sigma_d^2 = E[d(n) \cdot d^*(n)] = \sigma_v^2 + \left[ \mathbf{a}^H \right]_{1 \times m} \cdot \left[ \mathbf{R}_{uu} \right]_{m \times m} \cdot \left[ \mathbf{a} \right]_{m \times 1} \tag{34}
\]

where \( \mathbf{R}_{uu} = E[\mathbf{u}_m(n) \cdot \mathbf{u}_m^H(n)] \) is the \( m \)-by-\( m \) correlation matrix of the input vector.

Consider now a Wiener filter that operates on an input vector \( \mathbf{u}_M(n) = [u(n), u(n-1), \ldots, u(n-M+1)] \) and desired response \( d(n) \) to produce a minimum mean-square-error \( J_{\text{min}}(M) \), which is adjustable by varying the filter length \( M \). Substituting Eq. (34) in the expression of \( J_{\text{min}} \), we have:

\[
J_{\text{min}} = \sigma_d^2 - \mathbf{w}_0^H \cdot \mathbf{R}_{uu} \cdot \mathbf{w}_0 = \sigma_v^2 + \left[ \mathbf{a}^H \right]_{1 \times m} \cdot \left[ \mathbf{R}_{uu} \right]_{m \times m} \cdot \left[ \mathbf{a} \right]_{m \times 1} - \left[ \mathbf{w}_0^H \right]_{1 \times M} \cdot \left[ \mathbf{R}_{uu} \right]_{M \times M} \cdot \left[ \mathbf{w}_0 \right]_{M \times 1} \tag{35}
\]

The only adjustable term in this expression is \( \mathbf{w}_0^H \cdot \mathbf{R}_{uu} \cdot \mathbf{w}_0 \) which is quadratic in \( M \). We may identify three regimes of model selection:

1. **Underfitted model**, \( M < m \). The Wiener filter has improved performance with increasing \( M \) for a prescribed \( m \). The MSE decreases quadratically, starting from the worst possible condition:
\[ J_{\text{min}}(0) = \sigma_v^2 + a^H \cdot R_{uu} \cdot a \]  

(36.a)

2). Critically fitted model, \( M=m \). At the critical point \( M=m \), the Wiener filter is perfectly matched to the regression model, in that \( \mathbf{w}_0 = a \). The minimum MSE of the Wiener filter attains the lowest value:

\[ J_{\text{min}}(0) = \sigma_v^2 \]  

(36.b)

3). Overfitted model, \( M>m \). When the length of the Wiener filter is greater than the model of order \( m \), the tail end of the tap-weight vector is zero: \( \mathbf{w}_o = \left[ \mathbf{a}_{1xm} \right] \). The tap-input vector of the Wiener filter takes the form \( \mathbf{u}(n) = \left[ \mathbf{u}_m(n) \mathbf{u}_{M-m}(n) \right]^T \), where \( \mathbf{u}_{M-m}(n) \) is an \( (M-m) \)-by-1 vector made up of the past data samples immediately preceding the \( m \)-by-1 vector \( \mathbf{u}_m(n) \). The same value of MSE is obtained as in the case (b) but with a longer filter length.

From this discussion, the preferred design strategy is to match the length \( M \) of the Wiener filter to the order \( m \) of the regression model. In this critical case, the estimation error \( e_o(n) \) produced by the Wiener filter is white with variance \( \sigma_v^2 \), inheriting the statistical characterization of the additive noise \( v(n) \) in the regression model of Eq. (33).

Example 2: Consider a Wiener filtering problem, having a correlation matrix \( R_{uu} = \begin{bmatrix} 1,1 & 0,5 & 0,1 & -0,1 \\ 0,5 & 1,1 & 0,5 & 0,1 \\ 0,1 & 0,5 & 1,1 & 0,5 \\ -0,1 & 0,1 & 0,5 & 1,1 \end{bmatrix} \), the crosscorrelation vector between the tap-input vector \( \mathbf{u}(n) \) and the desired response \( d(n) \) is \( \mathbf{p} = \mathbf{r}_{ud} = \left[ 0,5 \ -0,4 \ -0,2 \ -0,1 \right]^T \), \( \sigma_d^2 = 1 \) and \( \sigma_v^2 = 1 \). Compute the tap-weights of the Wiener filter and the minimum mean square error for \( M=0,1,2,3,4 \).

Solution:

\[ \mathbf{M} = [0 \ 1 \ 2 \ 3 \ 4] \rightarrow \mathbf{J}_{\text{min}} = [1 \ 0.9773 \ 0.3219 \ 0.3141 \ 0.3141] \]

8. Practical issues in Wiener Filter Implementation

The linear optimal (Wiener) filter, \( \mathbf{w}_0 = \mathbf{R}^{-1} \cdot \mathbf{p} \), is ideal for many application. Several issues must be addressed to use it properly in practice.

1). In practice, one usually won’t know exactly the statistics of \( u_k \) and \( d_k \) (i.e., \( R_{uu} \) and \( \mathbf{p} = \mathbf{r}_{ud} \)) needed to compute the optimal filter. The problem is solved by estimating the statistics:

\[ \hat{R}_{uu}(l) \equiv \frac{1}{N} \sum_{k=0}^{N-1} u(k) \cdot u(k+l) \]  

(39.a)
\[
\hat{r}_{du}(l) \equiv \frac{1}{N} \sum_{k=0}^{N-1} d(k) \cdot u(k-l)
\]
(39.b)

\[
\hat{r}_{ud}(l) = \hat{r}_{du}(-l)
\]
(39.c)

and then solve

\[
\hat{w}_0 \approx \hat{R}^{-1} \cdot \hat{p}
\]
(39.c)

2). In many applications, the statistics of \( u(k) \) or \( d(k) \) vary slowly with time. How does one develop an adaptive system which tracks these changes over time to keep the system near optimal at all times? The solution is on using short-time windowed estimates of the correlation functions:

\[
\hat{r}_{uu}^k(l) \equiv \frac{1}{N} \sum_{i=0}^{N-1} u(k-i) \cdot u(k-i-l)
\]
(40.a)

\[
\hat{r}_{du}^k(l) \equiv \frac{1}{N} \sum_{i=0}^{N-1} d(k-i) \cdot u(k-i-l)
\]
(40.b)

and, finally,

\[
\hat{w}_0^k \approx \left( \hat{R}^k \right)^{-1} \cdot \hat{p}^k
\]
(40.c)

so each window has a Winer solution.

3). How can compute efficiently the autocorrelation function \( \hat{r}_{uu}^k(l) \)? The solution is to use a recursive relation as

\[
r_{uu}^k(l) \equiv r_{uu}^{k-1}(l) + u(k) \cdot u(k-l) - u(k-N) \cdot u(k-N-l)
\]
(41.a)

This is critically stable, so people usually do:

\[
(1-\alpha) \cdot r_{uu}^k(l) \equiv \alpha \cdot r_{uu}^{k-1}(l) + u(k) \cdot u(k-l)
\]
(41.b)

4). How does one choose the number of samples, \( N \)? The solution is on a trade-off. Larger \( N \) means more accurate estimates of the correlation values and better \( \hat{w}_0 \). However, larger \( N \) leads to slower adaptation. In fact, the success of adaptive system depends on \( u, d \) being roughly stationary over \( N \) samples, so \( N>M \).

Summary

The importance of the Wiener filter lies in the fact that it provides a frame of reference for the linear filtering of stochastic signals, assuming wide-sense stationarity.

The Wiener filter has two important properties:

1). The principle of orthogonality: the error signal (estimation error) is orthogonal to its tap inputs.

2). Statistical characterization of the error signal as white noise: the condition is attained when the filter length matches the order of the multiple regression model describing the generation of the observable data (i.e., the desired response).
Multiple regression model identification

In figure 3 we recognize a first block to generate a signal with \( v_1(n) = u(n) \) as input and \( y(n) \) as output. The input-output relation is

\[
y(n) = \sum_{i=1}^{m} a(i) \cdot u(n-i+1) = a^T \cdot u
\]

with

\[
a = [a_1 \ a_2 \ ... \ a_M]^T \quad \text{and} \quad u(n) = [u(n) \ u(n-1) \ ... \ u(n-M+1)]^T.
\]

and is called a **Multiple Linear Regression (MLR)** model, which is slightly different of an AR model which has the generation equation as:

\[
y(n) = -\sum_{i=1}^{m} a(i) \cdot y(n-i) + u(n)
\]

or

\[
y(n) = -a \cdot y + u(n)
\]

The considerations made in the section 6 are made only for MLR model. Below are the results of simulation with “Lab_43.m”. The are two MLR models, of order \( m=3 \) and \( m=4 \) respectively. In each case there are good estimation results, in the sense that the results are close the theory. There are still some differences, i.e. between theory and results of simulation and try to explain why?

Q1). If \( J(M = 10) \leq J(M = 3) \) which value of \( M \) is considered?

Q2). Why the coefficients \( w_0(i,i) > m \) are not exactly zero?

\[
W_0 \cdot v3 = [10 \ 0.3444 \ -0.1282 \ 0.1851 \ -0.0178 \ 0.0001 \ -0.0003 \ -0.0007 \ -0.0210 \ -0.0040 \ 0.0006];
\]
Adaptive Wiener Filtering

Using Adaptive Filtering

The wiener2 function applies a Wiener filter (a type of linear filter) to an image adaptively, tailoring itself to the local image variance. Where the variance is large, wiener2 performs little smoothing. Where the variance is small, wiener2 performs more smoothing.

This approach often produces better results than linear filtering. The adaptive filter is more selective than a comparable linear filter, preserving edges and other high-frequency parts of an image. In addition, there are no design tasks.

The wiener2 function handles all preliminary computations and implements the filter for an input image.

The wiener2, however, does require more computation time than linear filtering. wiener2 works best when the noise is constant-power ("white") additive noise, such as Gaussian noise.

Algorithm

Wiener2 estimates the local mean and variance around each pixel,

\[ \mu = \frac{1}{NM} \sum_{n_1=1}^{N} \sum_{n_2=1}^{M} A(n_1, n_2) \]  

\[ \sigma^2 = R_{AA}(0,0) - \mu^2 = \frac{1}{NM} \sum_{n_1=1}^{N} \sum_{n_2=1}^{M} A^2(n_1, n_2) - \mu^2 \]  

where \( A \) is the \( N \)-by-\( M \) local neighborhood of each pixel in the image \( A \).

Wiener2 then creates a pixelwise Wiener filter using these estimates:

\[ N(n_1, n_2) = \mu + \frac{\sigma^2 - \nu^2}{\sigma^2} [A(n_1, n_2) - \mu] \]  

where \( \nu^2 \) is the noise variance. If the noise variance is not given, wiener2 uses the average of all the local estimated variances.

Syntax

\[ J = \text{wiener2}(I, [m n], \text{noise}) \]

Description

wiener2 lowpass-filters a grayscale image that has been degraded by constant power additive noise.
wiener2 uses a pixelwise adaptive Wiener method based on statistics estimated from a local neighborhood of each pixel.

\[ J = \text{wiener2}(I, [m \ n], \text{noise}) \]

filters the image \( I \) using pixelwise adaptive Wiener filtering, using neighborhoods of size \( m \)-by-\( n \) to estimate the local image mean and standard deviation. If you omit the \([m \ n]\) argument, \( m \) and \( n \) default to 3.

\[ [J, \text{noise}] = \text{wiener2}(I, [m \ n]) \]

also estimates the additive noise power before doing the filtering. wiener2 returns this estimate in noise.

**Removing Noise By Adaptive Filtering. Example of code source.**

The example below applies wiener2 to an image of Saturn that has had Gaussian noise added. For an interactive demonstration of filtering to remove noise, try running `nrfiltdemo`.

1. **Read in an image.** If the image is in other format then RGB or gray converts it to grayscale.

   RGB = imread('saturn.png');
   I = rgb2gray(RGB);

2. **Add Gaussian noise to the image and then displays the image.** Because the image is quite large, the figure only shows a portion of the image.

   J = imnoise(I,'gaussian',0,0.025);
   imshow(J)

3. **Remove the noise, using the wiener2 function.** Again, the figure shows a portion of the image

   K = wiener2(J, [5 5]);
   Figure, imshow(K);
Figure 1: An example of adaptive Wiener filtering

**Reference**


\[
W_0v4 = [ 10 \ 0.3435 \ -0.1154 \ 0.1850 \ 0.3355 \ -0.0010 \ 0.0121 \ 0.0002 \ -0.0099 \\
0.0075 \ -0.0034 ];
\]
Figure 2: Results of Wiener filter coefficients estimation for MLR model
Course 5 - Method of Steepest Descend\(^1\)

Content
Method of Steepest Descend
Basic idea of the steepest-descent algorithm
The steepest-descent algorithm applied to the Wiener filter
Examples
Summary

We study a gradient-based adaptation by describing an old optimization technique, known as the method of steepest descent.

The method is recursive in the sense that its formulation is represented by a feedback system whereby the computation of the filter proceeds iteratively in a step-by-step manner.

When the method is applied to the Wiener filter, its provides us with an algorithm solution that allows the tracking of time variations in the signal’s statistics without having to solve the Wiener-Hopf equations each time the statistics change.

In the particular case of a stationary environment, we find that, starting from an arbitrary initial value of the tap-weight vector, the solution improves with the increased number of iterations. The important note is that the solution obtained converges to the Wiener solution without to invert the correlation matrix of the input vector.

1. Basic idea of the steepest-descent algorithm

Consider a cost function \( J(w) \) that is continuously differentiable function of some unknown weight vector \( w \). The function \( J(w) \) maps the elements of \( w \) into real numbers. We want to find an optimal solution \( w_0 \) that satisfies the condition

\[
J(w_0) \leq J(w), \quad \forall w
\]

The idea of the (unconstrained) optimization of (1) is based on the idea of local iterative descent: starting with an initial guess denoted by \( w(0) \), generate a sequence of weights vectors \( w(1), w(2),..., \) such that the cost function \( J(w) \) is reduced at each iteration of the algorithm:

\[
J(w(n+1)) \leq J(w(n)), \quad \forall w
\]

where \( w(n) \) is the old value of the weight vector and \( w(n+1) \) is the updated value.

In a simple form of iterative descent, known as method of steepest descent, the successive adjustments applied to the weight vector \( w \) are in the direction of steepest descend, that is, in opposite direction to the gradient vector of the cost function, which is denoted by \( \nabla J(w) \). Some times a notation is used for the gradient as

\[
g = \nabla J(w) = \frac{\partial J(w)}{\partial w}
\]

\(^1\) Metoda celei mai bune coborari (scaderi).
The steepest-descent algorithm is formally described by:

\[
w(n+1) = w(n) - \frac{1}{2} \cdot \mu \cdot \nabla J(w) = w(n) - \frac{1}{2} \cdot \mu \cdot \frac{\partial J(w(n))}{w(n)} = w(n) - \frac{1}{2} \cdot \mu \cdot g(n)
\]  

(4)

where \( n \) denotes the iteration (i.e., the time step in the iterative process), \( \mu \) is a positive constant called step-size parameter, and the factor \( 1/2 \) is introduced for mathematical convenience. Thus, the basic relation is:

\[
w(n+1) = w(n) - \frac{1}{2} \cdot \mu \cdot g(n)
\]  

(4.a)

In going from iteration \( n \) to \( n+1 \), the algorithm applies the weight adjustment:

\[
\delta w(n) = w(n+1) - w(n) = \frac{1}{2} \cdot \mu \cdot g(n)
\]  

(5)

Note: By using a first order Taylor series expansion around \( w \) of \( J(w(n+1)) \) we obtain the approximation

\[
J(w(n+1)) \approx J(w(n)) + \left[ \frac{\partial J(w(n))}{w(n)} \right]^H \cdot \delta w(n) =
\]

\[
= J(w(n)) + \left[ \frac{\partial J(w(n))}{w(n)} \right]^H \cdot \frac{1}{2} \cdot \mu \cdot \left[ \frac{\partial J(w(n))}{w(n)} \right] =
\]

\[
= J(w(n)) - \frac{1}{2} \cdot \mu \cdot \left\| \frac{\partial J(w(n))}{w(n)} \right\|^2 \leq J(w(n))
\]  

(6)

which shows that \( J(w(n+1)) \) is smaller than \( J(w(n)) \) if the step size \( \mu \) is positive.

2. The steepest-descent algorithm applied to the Wiener filter

Consider a transversal filter with tap inputs \( u(n), u(n-1),...,u(n-M+1) \) and a corresponding set of tap weights \( w_0(n), w_1(n),...,w_{M-1}(n) \). As Fig.1 shows, the tap inputs represent samples drawn from a wide-sense stationary stochastic process of zero mean. In addition to these inputs, the filter is supplied with a desired response \( d(n) \), that provides a frame of reference for the optimum filtering action.

The estimation error is

\[
e(n) = d(n) - \hat{d}(n) = d(n) - w^H(n) \cdot u(n)
\]  

(7)

with

\[
w(n) = [w_0(n) \quad w_1(n) \quad ... \quad w_{M-1}(n)]^T
\]  

(7.a)

\[
u(n) = [u(n) \quad u(n-1) \quad ... \quad u(n-M+1)]^T
\]  

(7.b)
If the tap-input vector \( u(n) \) and the desired response \( d(n) \) are jointly stationary, then the *mean square error function* \( J(w(n)) \) is a quadratic function of the tap-weight vector:

\[
J(w(n)) = E[e(n) \cdot e^*(n)] = E\left[ (d(n) - w^H(n) \cdot u(n))^\text{T} \cdot (d(n) - w^H(n) \cdot u(n)) \right] = \\
E[d(n) \cdot d^*(n)] - E[d(n) \cdot u^H(n) \cdot w(n)] - E[w^H(n) \cdot u(n) \cdot d^*(n)] + \\
E[w^H(n) \cdot u(n) \cdot u^H(n) \cdot w(n)]
\]

and by reducing

\[
J(w(n)) = \sigma^2_d - E[d(n) \cdot u^H(n)] \cdot w(n) - w^H(n) \cdot E[u(n) \cdot d^*(n)] + \\
w^H(n) \cdot E[u(n) \cdot u^H(n)] \cdot w(n)
\]  

and finally:

\[
J(w(n)) = \sigma^2_d - r_{du} \cdot w(n) - w^H(n) \cdot r_{ud} \cdot +w^H(n) \cdot R_{uu} \cdot w(n)
\]

**Figure 1 – Structure of adaptive transversal filter**

We have obtained:

\[
J(w(n)) = \sigma^2_d - \mathbf{w}^H(n) \cdot \mathbf{r}_{ud} - \mathbf{r}_{ud}^H \cdot \mathbf{w}(n) + \mathbf{w}^H(n) \cdot \mathbf{R}_{uu} \cdot \mathbf{w}(n)
\]  

where \( \sigma^2_d \) is the variance of the desired response \( d(n) \), \( \mathbf{p} = \mathbf{r}_{ud} \) is the cross-correlation between the tap-input vector \( u(n) \) and the desired response \( d(n) \).

The gradient of the weights (parameters) is given by:
\[
\n\nabla J(w(n)) = \begin{bmatrix}
\frac{\partial J(w(n))}{\partial a_1(n)} & \cdots & \frac{\partial J(w(n))}{\partial a_M(n)}
\end{bmatrix} = -2 \cdot r_{ud} + 2 \cdot R_{uu} \cdot w(n) \tag{9}
\]

So, substituting (9) in (4) we obtain:

\[

w(n + 1) = w(n) + \mu \cdot [r_{ud} - R_{uu} \cdot w(n)]
\tag{10}
\]

which describes the mathematical formulation of the steepest-descent algorithm for Wiener filtering.

**Note 1:** The **necessary and sufficient condition** for the convergence or stability of the steepest descent algorithm is that the step-size parameter \( \mu \) satisfy the double inequality:

\[
0 < \mu < \frac{2}{\lambda_{\text{max}}}
\tag{11}
\]

where \( \lambda_{\text{max}} \) is the largest eigenvalue of the autocorrelation matrix \( R_{uu} \). Depending on the value assigned to the step-size parameter \( \mu \), the transient response of steepest descent exhibit one of three forms of behavior:

a) underdumped response, for \( \mu \) large, when the trajectory exhibits oscillations;

b) overdamped response, for \( \mu \) small, with a nonoscillatory behavior;

c) critically damped response, which is the line dividing underdamped and overdamped condition.

**Note 2:** The curve obtained by plotting the mean-square error \( J(w(n)) \) versus the number of iterations, \( n \), is called a **learning curve**.

**Note 3:** Limitation of the steepest descend algorithm. The important feature of the algorithm is the simplicity of its implementation. However, we may require a large number of iterations for the algorithm to converge to a point sufficiently close to the optimum solution, \( w_0 \). The performance limitation is generated by the fact that the algorithm is based on a straight – line (i.e. first order) approximation of the error-performance around the current point.

**Example:** Consider an AR process of order one, described by the difference equation:

\[

u(n) = -a \cdot u(n-1) + v(n),
\]

where \( a \in \mathbb{R} \) is the parameter of the process and \( v(n) \) is a zero-mean white noise of variance \( \sigma_v^2 \).

1). Set up a linear predictor of order one to compute the parameter \( a \), by using the steepest descend algorithm for the recursive computation of the Wiener solution of parameter \( a \).

2). Plot the error-performance curve for this problem, identifying the minimum point of the curve in terms of known parameters.

3). What is the condition on the step-size parameter \( \mu \) to ensure stability?

4). The structure of the filter.

**Solution:** 1). The general relation of the algorithm is

\[
\]
\[ w(n+1) = w(n) + \mu \cdot [r_{ud} - R_{uu} \cdot w(n)] \]

For the one-step predictor of order one we have \( a(n) = -w(n) \), and hence
\[ a(n+1) = a(n) - \mu \cdot [r_{ud} + R_{uu} \cdot a(n)] \]

The cross-correlation between the input tap \( u(n-1)_{Mx1} = [u(n-1)]_{1x1} \) and the desired response \( u(n) \) is
\[ r_{ud} = E[u(n-1) \cdot u(n)] = r_{uu}(-1) = r_{uu}(1) = r(1) \]

The autocorrelation matrix of the input taps is
\[ R_{uu} = E[u(n-1) \cdot u^H(n-1)] = E[u(n-1) \cdot u(n-1)] = E[u(n-1)^2] = r(0) \]

It results
\[ a(n+1) = a(n) - \mu \cdot [r(1) + r(0) \cdot a(n)] \]

2). The error-performance surface is a curve in \( xOy \) axis because we have only one parameter, \( a \). Starting from the general expression we have
\[ J(w(n)) = \sigma_d^2 - w^H(n) \cdot r_{ud} - r_{ud}^H \cdot w(n) + w^H(n) \cdot R_{uu} \cdot w(n) \]

and taking into account the fact
\[ \sigma_d^2 = E[|u(n)|^2] = r(0) \]

the following expression is obtained
\[ J(a(n)) = r(0) + a(n) \cdot r(1) + r(1) \cdot a(n) + a(n) \cdot r(0) \cdot a(n) = r(0) + 2 \cdot a(n) \cdot r(1) + r(0) \cdot a(n)^2 \]

First derivative is indicating the optimum value of \( a \):
\[ \frac{dJ}{da} = +2 \cdot r(1) + 2 \cdot r(0) \cdot a = 0 \rightarrow a_0 = -\frac{r(1)}{r(0)} \]

The minimum value of the error – criterion is then
\[ J_{\text{min}} = r(0) - \frac{r^2(1)}{r(0)} \]
\begin{equation*}
J(a_0) = r(0) - 2 \cdot \frac{r(1)}{r(0)} \cdot r(1) + r(0) \cdot \left( \frac{r(1)}{r(0)} \right)^2 = r(0) - \frac{r^2(1)}{r(0)}
\end{equation*}

The corresponding plot of the error surface is therefore presented below, for \( r(0) = 1, \ r(1) = -0.3 \). Other values are: \( a_0 = 0.3, \ \min(J) = 0.91 \).

c). The conditions for the step-size parameter are \( 0 < \mu < \frac{2}{\lambda_{\max}} = \frac{2}{r(0)} \)

d). The structure of the filter is presented now:

Example 2: Let an autoregressive process described by the difference equation:
\begin{equation*}
u(n) = -0.5 \cdot u(n - 1) + 0.1 \cdot u(n - 2) + v(n), \end{equation*}
where \( v(n) \) is a zero-mean white noise of unit variance. The method of steepest descend is used for the recursive computation of the optimum weight vector of the forward linear predictor applied to the process \( u(n) \). Find the bounds of the step-size parameter \( \mu \) that ensure stability of the steepest-descent algorithm.

Solution: The structure of the filter is presented now. It has two tap delays.

For the one step predictor the input vector \( u(n) \) must be considered by a delay, as \( u(n-1) \).
The correlation matrix is
\[
\begin{pmatrix}
r_{uu}(0) & r_{uu}(1) \\
r_{uu}(1) & r_{uu}(0)
\end{pmatrix}
\]

The autocorrelation matrix is
\[
\begin{pmatrix}
r_{uu}(0) & r_{uu}(1) \\
r_{uu}(1) & r_{uu}(0)
\end{pmatrix}
\]

The elements of the autocorrelation matrix could be computed by solving the Yule-Walker equations, as:
\[
\begin{pmatrix}
r(0) & r(1) \\
r(1) & r(0)
\end{pmatrix}
\begin{pmatrix}
-a_1 \\
-a_2
\end{pmatrix}
= \begin{pmatrix}
r(1) \\
r(2)
\end{pmatrix} \quad \text{or} \quad \begin{pmatrix}
r(0) & r(1) \\
r(1) & r(0)
\end{pmatrix}
\begin{pmatrix}
-0.5 & 0.1 \\
0.1 & 0.1
\end{pmatrix}
= \begin{pmatrix}
r(1) \\
r(2)
\end{pmatrix}
\]

which is a system of two equations with three unknowns, \( r(0), r(1), r(2) \). The third equation is obtained from the relation between the variance of the noise:
\[
\sigma_v^2 = \sum_{k=0}^{2} a_k \cdot r(k) \quad \text{or} \quad \sigma_v^2 = r(0) + a_1 \cdot r(1) + a_2 \cdot r(2)
\]

The set the three equations and the solution are
\[
\begin{cases}
-0.5r(0) + 0.1r(1) = r(1) \\
-0.5r(1) + 0.1r(0) = r(2) \\
r(0) + 0.5r(1) - 0.1r(2) = 1
\end{cases}
\rightarrow
\begin{cases}
r(0) = 0.8065 \\
r(1) = 0.4480 \\
r(2) = 0.3047
\end{cases}
\]

The autocorrelation matrix is
\[
\begin{pmatrix}
r_{uu}(0) & r_{uu}(1) \\
r_{uu}(1) & r_{uu}(0)
\end{pmatrix}
= \begin{pmatrix}
0.8065 & 0.4480 \\
0.4480 & 0.8065
\end{pmatrix}
\]

The eigenvalues of \( R \) are \{0.3585, 1.2545\} and the condition for stability is \( 0 < \mu < 1.59 \).

**Example 3:** For the previous formulation of the problem write the relations to update the weights of the filter in order to estimate the “unknowns” coefficients, \( a_1 \) and \( a_2 \), by the method of steepest descend.

**Solution:**

The general relation of the algorithm is
\[
w(n+1) = w(n) + \mu [r_{ud} - R_{uu} \cdot w(n)]
\]

For the one-step predictor of order one we have \( a(n) = -w(n) \), and hence
\[
a(n+1) = a(n) - \mu [r_{ud} + R_{uu} \cdot a(n)]
\]

The cross-correlation between the input tap \( u(n-1) \) and the desired response \( u(n) \) is
\[
r_{ud} = [r(1) \quad r(2)]^T
\]

The autocorrelation matrix of the input taps is
\[
\mathbf{R}_{uu} = E \left[ \mathbf{u}(n-1) \cdot \mathbf{u}^H(n-1) \right] = \begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix}
\]

It results
\[
\begin{bmatrix}
    a_1(n+1) \\
    a_2(n+1)
\end{bmatrix} = \begin{bmatrix}
    a_1(n) \\
    a_2(n)
\end{bmatrix} - \mu \cdot \begin{bmatrix}
    r(1) \\
    r(2)
\end{bmatrix} + \begin{bmatrix}
    r(0) & r(1) \\
    r(1) & r(0)
\end{bmatrix} \cdot \begin{bmatrix}
    a_1(n) \\
    a_2(n)
\end{bmatrix}
\]
or
\[
\begin{cases}
    a_1(n+1) = a_1(n) - \mu \cdot [r(1) + r(0) \cdot a_1(n) + r(1) \cdot a_2(n)] \\
    a_2(n+1) = a_2(n) - \mu \cdot [r(2) + r(1) \cdot a_1(n) + r(0) \cdot a_2(n)]
\end{cases}
\]

The error-performance surface has an expression as
\[
J(\mathbf{a}) = r(0) + \begin{bmatrix}
    a_1 & a_2
\end{bmatrix} \cdot \begin{bmatrix}
    r(1) \\
    r(2)
\end{bmatrix} + \begin{bmatrix}
    r(1) & r(2)
\end{bmatrix} \cdot \begin{bmatrix}
    a_1 \\
    a_2
\end{bmatrix} + \begin{bmatrix}
    a_1 & a_2
\end{bmatrix} \cdot \begin{bmatrix}
    r(0) & r(1) \\
    r(1) & r(0)
\end{bmatrix} \cdot \begin{bmatrix}
    a_1 \\
    a_2
\end{bmatrix}
\]
\[
= r(0) + 2r(1)a_1 + 2r(2)a_2 + r(0)a_1^2 + 2r(1)a_1a_2 + r(0)a_2^2
\]
The corresponding plot of the error surface is therefore presented below, for \(r(0)=0.8065, \ r(1)=0.4480, \ r(2)=0.3047\). Other values are: \(\min(J) = 0.4259\).
Summary

The steepest descent method is a method of unconstrained linear optimization.

The optimization criterion is the mean-square of the estimation error, i.e. the difference between desired and obtained response of the filter.

The method is deterministic, in the sense that the recursion formula is based on the (deterministic) minimization of the error criterion.

The method is commonly used in the parameter estimation problems, by using a linear transversal parameter filter and a structure of forward predictor.

The advantage of the method, comparing with e.g. Wiener method, is the recursion formula in the computation of the weights of the filter, which avoid the computation of the inverse matrix of correlation.
Course 5

Least-Mean-Square Adaptive Filters\(^{10}\)

Content
The structure and operation of the Least-Mean-Squared Algorithm
Least-Mean-Square Adaptation Algorithm
Learning curves
Normalized Least-Mean-Square Adaptive Filters (nLMS)

Summary

The LMS algorithm is an important member of stochastic gradient algorithms, in opposite of the method of steepest descent, which uses a deterministic gradient in a recursive computation of the Wiener filter for stochastic inputs.

A significant feature of the LMS algorithm is its simplicity. Moreover, it does not require measurements of the pertinent correlation functions, nor does it require matrix inversion.

1. The structure and operation of the Least-Mean-Squared Algorithm

The LMS algorithm is a linear adaptive filtering algorithm, which consists of two basic processes:
1). A filtering process, which involves a) computation the output of a linear filter in response to an input signal, and b). generation of the estimation error by comparing this output with a desired response.
2). An adaptive process, which involves the automatic adjustment of the parameters of the filter in accordance with the estimation error.

The combination of these two processes working together constitutes a feedback loop, as illustrated in Fig. 1. First, we have a transversal filter, around which the LMS algorithm is built; this component is responsible for performing the filtering process. Second, we have a mechanism for performing the adaptive control process on the loop weights of the transversal filter.

\[ u(n) \rightarrow \hat{w}(n) \rightarrow \hat{d}(n) \]

\[ e(n) = d(n) - \hat{d}(n) \]

Figure 1 - Block diagram of the adaptive transversal filter

\(^{10}\) Widrow and Hoff, 1960
LMS = “Cea mai mica Media a patratelor”
The used vectors are

- the input vector at time $n$: $\mathbf{u}(n) = [u(n) \ u(n-1) \ ... \ u(n-M+1)]^T$
- the estimated weight vector at time $n$: $\hat{\mathbf{w}}(n) = [w_0(n) \ w_1(n) \ ... \ w_{M-1}(n)]^T$

The value computed for this vector using LMS algorithm represents an estimate whose expected value may come close to the Wiener solution, $\mathbf{w}_0$, as the number of iterations, $n$, approaches infinity. The tap-input vector $\mathbf{u}(n)$ and the desired response $d(n)$ are drawn from a jointly wide-sense stationary environment. The desired response $d(n)$ is linearly related to the input vector (i.e., regressor) $\mathbf{u}(n)$ by a multiple regression model whose parameter vector is unknown.

2. Least-Mean-Square Adaptation Algorithm

The LMS algorithm comes to correct the steepest-descent algorithm, in the sense that the exact measurements of the gradient vector $\nabla J(n)$ at iteration $(n)$ are not possible, since that would require prior knowledge of both correlation matrix $R_{uu}$ and the cross-correlation vector $p = r_{ud}$. Consequently, the gradient vector must be estimated from the available data when we operate in an unknown environment.

The gradient formula of SDA (Steepest Descent Algorithm) was:

$$\nabla J(w(n)) = -2p + 2R \cdot w(n) = -2 \cdot r_{ud} + 2R_{uu} \cdot w(n) \quad (1)$$

The simplest choice of estimators is to use instantaneous estimates for $R_{uu}$ and $p=r_{ud}$ that are based on sample values of the tap-input vector and desired response, defined by:

$$\hat{R}(n) = \mathbf{u}(n) \cdot \mathbf{u}^H(n) \quad (2)$$

and

$$\hat{p}(n) = \hat{r}_{ud} = \mathbf{u}(n) \cdot d^*(n) \quad (3)$$

The “instantaneous” estimate of the gradient vector is:

$$\hat{\nabla} J(n) = -2\mathbf{u}(n) \cdot d^*(n) + 2 \cdot \mathbf{u}(n) \cdot \mathbf{u}^H(n) \cdot \hat{w}(n) \quad (4)$$

Substituting the Eq. (4) for the gradient vector in the SDA equation,

$$w(n+1) = w(n) - \frac{1}{2} \cdot \mu \cdot \nabla J(w)$$

we obtain a new recursive relation for updating the tap-weight vector.

$$\hat{w}(n+1) = \hat{w}(n) + \mu \cdot \mathbf{u}(n) \cdot [d^*(n) - \mathbf{u}^H(n) \cdot \hat{w}(n)] = \hat{w}(n) + \mu \cdot \mathbf{u}(n) \cdot e^*(n) \quad (5)$$

11 Steepest = cea mai buna..
Note 1: We have used a hat symbol over the symbol for the weight vector to distinguish it from the value obtained by using SDA.

Note 2: Although the derivation of the LMS algorithm is entirely based on wide-sense stationary stochastic signals, the algorithm is also applicable to the following other situations:
- A deterministic environment, in which the tap input vector \( \mathbf{u}(n) \) and desired response \( d(n) \) are both deterministic signals.
- A nonstationary environment, in which we have an additional requirement, namely, the tracking of statistical variations in the environment.

3. Learning curves

It is a common practice to use ensemble-average learning curves to study the statistical performance of adaptive filters. We may identify two kinds of learning curves.

1). The mean-square-error (MSE) learning curve, which is based on ensemble averaging of the squared estimation error (signal), \( | e(n) |^2 \). This learning curve is thus a plot of the mean-square error:

\[
J(n) = E\left[ |e(n)|^2 \right] = E\left[ |d(n) - \hat{d}(n)|^2 \right] = E\left[ |d(n) - \mathbf{w}^H(n) \cdot \mathbf{u}(n)|^2 \right]
\]

(6)

2). The mean-square deviation (MSD) learning curve, which is based on ensemble averaging of the squared (weight) error deviation, \( \| \mathbf{e}(n) \|^2 \). This learning curve is thus a plot of the mean-square deviation:

\[
D(n) = E\left[ \|\mathbf{e}(n)\|^2 \right]
\]

(7)

where

\[
\mathbf{e}(n) = \mathbf{w}_0 - \hat{\mathbf{w}}(n)
\]

(7.a)

is the weight-error vector, and \( \mathbf{w}_0 \) denotes the optimum Wiener solution for the tap-weights vector, and \( \hat{\mathbf{w}}(n) \) is the estimate produced by the LMS filter at iteration \( n \).

Unlike the case of steepest-descent algorithm, both the mean-square error \( J(n) \) and the mean-square deviation \( D(n) \) in the LMS algorithm depend on iteration, \( n \), because the estimation error, \( e(n) \), and weight-error vector \( \mathbf{e}(n) \) are both nonstationary processes. In fact, the learning curves of LMS are noisy, in opposition of SDA.

4. Normalized Least-Mean-Square Adaptive Filters (nLMS)

The adjustment in LMS algorithm is direct proportional to the tap-input vector \( \mathbf{u}(n) \), i.e.:

\[
\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) \cdot e^*(n) = \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) \cdot [d^*(n) - \mathbf{u}^H(n) \cdot \hat{\mathbf{w}}(n)]
\]

(8)
Therefore, when \( u(n) \) is large, the LMS filter suffers from a gradient noise multiplication (amplification) problem\(^\text{12}\).

To overcome this difficulty, a normalized LMS filter is used. The idea is to “normalize” the adjustment applied to the tap-weight vector at iteration \((n+1)\) by the squared Euclidean norm of the tap-input vector \( u(n) \) at iteration \( n \).

The formula for computing the \( M \)-by-1 tap-weight vector is now

\[
\hat{w}(n+1) = \hat{w}(n) + \frac{\tilde{\mu}}{\|u(n)\|^2} \cdot u(n) \cdot e^*(n) = \hat{w}(n) + \frac{\tilde{\mu}}{\|u(n)\|^2} \cdot u(n) \cdot \left( d^*(n) - u^H(n) \cdot \hat{w}(n) \right)
\]

(9)

where \( \tilde{\mu} \) is a real positive scaling factor and

\[
\|u(n)\|^2 = \sum_{k=1}^{M} u_k^2
\]

(9.a)

Comparing Eq. (5) with Eq.(9) we may make some observations:

1). The adaptation constant \( \tilde{\mu} \) for nLMS filter is dimensionless, whereas the adaptation constant \( \mu \) for the LMS filter has the dimension of inverse power.

2). Setting

\[
\mu(n) = \frac{\tilde{\mu}}{\|u(n)\|^2}
\]

(10)

we may view the nLMS filter as an LMS filter with a time varying step-size parameter.

3). The nLMS algorithm has a rate of convergence potentially faster than that of the standard LMS algorithm.

**Note 1:** nLMS filter has a problem for small tap-input vector \( u(n) \), when numerical difficulties may arise because then we have to divide by a small value for the squared norm \( \|u(n)\|^2 \). To overcome this problem, we modify the Eq. (9) slightly to produce

\[
\hat{w}(n+1) = \hat{w}(n) + \frac{\tilde{\mu}}{\delta + \|u(n)\|^2} \cdot u(n) \cdot e^*(n), \quad \delta > 0
\]

(11)

For \( \delta = 0 \), Eq. (11) reduces to the form given in (9).

**Note 2:** The upper bound of the normalized step-size parameter \( \tilde{\mu} \) and for real data is

\[
0 < \delta < 2 \frac{D(n) \cdot E[u^2(n)]}{E[e^2(n)]}
\]

(12)

\(^{12}\) Output Noise in rising with input noise
where \( E[e^2(n)] \) is the error signal energy, \( E[u^2(n)] \) is the input signal energy and \( D(n) = E[\|e(n)\|^2] = E[\|w - \hat{w}(n)\|^2] \) is the mean-square deviation of estimations. For the computation of these two first powers see below. The estimation of \( D(n) \) is difficult in general case but not so difficult in real applications with some helpful hypothesis, e.g. in speech based applications.

**Note 3:** For short –term power estimation of a signal denoted by \( x(n) \), we may use the idea of convex optimization to formulate a first-order recursive procedure described by:

\[
x^2(n+1) = (1 - \gamma) \cdot x^2(n) + \gamma \cdot x^2(n)
\]  

(13)

where \( x^2(n) \) is the power estimate at time \( n \) and \( \gamma \in [0.9; 0.99] \) is a smoothing constant. For a prescribed value of \( \gamma \), we may thus use (13) to estimate the error signal power \( E[e^2(n)] \) and input signal power \( E[u^2(n)] \) by setting \( x(n) = e(n) \) and \( x(n) = u(n) \), respectively.

### Summary

**Summary of the LMS algorithm**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>( M ) - the number of taps (i.e., the filter length)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \mu ) - step size parameter ( \in \left[ 0; 2 \right] \frac{2}{M \cdot S_{\text{max}}} )</td>
</tr>
<tr>
<td></td>
<td>( S_{\text{max}} ) is the maximum value of the power spectral density of the tap input ( u(n) ).</td>
</tr>
<tr>
<td></td>
<td>( M ) - the filter length, as a moderate to large value</td>
</tr>
<tr>
<td><strong>Initialization</strong></td>
<td>( \mathbf{w}(0) = 0 ) or ( \mathbf{w}(0) = \text{randn}(M,1) ) if no prior knowledge of ( \hat{\mathbf{w}}(n) ) is available.</td>
</tr>
<tr>
<td></td>
<td>( J_{\text{min}} ), the minimum value of the error criterion</td>
</tr>
<tr>
<td><strong>Data</strong></td>
<td>( \mathbf{u}(n) = [u(n) \quad u(n-1) \quad \ldots \quad u(n-M+1)]^T ), if ( d(n) = u(n+1) )</td>
</tr>
<tr>
<td></td>
<td>( \mathbf{u}(n) = [u(n-1) \quad u(n-2) \quad \ldots \quad u(n-M)]^T ), if ( d(n) = u(n) )</td>
</tr>
<tr>
<td></td>
<td>( d(n) ), desired response at time ( n );</td>
</tr>
<tr>
<td><strong>To compute</strong></td>
<td>( \hat{\mathbf{w}}(n+1) ) estimated weights at time ( (n+1) )</td>
</tr>
<tr>
<td><strong>Computation</strong></td>
<td>( n = 1; )</td>
</tr>
<tr>
<td></td>
<td>DO</td>
</tr>
<tr>
<td></td>
<td>( e(n) = d(n) - \hat{\mathbf{w}}^H(n) \cdot \mathbf{u}(n) )</td>
</tr>
<tr>
<td></td>
<td>( \hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) \cdot e^*(n) )</td>
</tr>
<tr>
<td></td>
<td>( n = n + 1; )</td>
</tr>
<tr>
<td></td>
<td>UNTIL ( J &lt; J_{\text{min}} )</td>
</tr>
</tbody>
</table>
A standard form of least-mean-square (LMS) filters was presented, which is the "workhorse" of linear adaptive filtering.

The practical importance of LMS filters is due largely to two unique attributes:

- simplicity of implementation;
- a model independent, and therefore robust, performance.

The main limitation of LMS filters is their relatively slow rate of convergence. Two factors are in charge: the step-size parameter $\mu$ and the eigenvalues of the correlation matrix, $R_{uu}$.

The effects are:

- if the step-size parameter $\mu$ is small, the adaptation is slow, which is equivalent to the LMS filter having a long "memory". If $\mu$ is big, the adaptation is relatively fast, but at the expense of an increase in the average excess mean-square-error after adaptation;

- The convergence depends on the spreading of the eigenvalues of the correlation matrix. The excess of the mean-square-error produced by the LMS filter is imposed primarily by the largest eigenvalues, and the time convergence is limited by the spread of the eigenvalues of $R$. When the eigenvalue spread is large, the convergence of the LMS filter may slow down. For example:

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Matrix and eigenvalues</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$A =$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.0000 2.0000 -3.0000 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10.0000 -2.5000 -2.0000 -1.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9.0000 2.0000 -1.0000 -6.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0000 3.0000 -2.3000 9.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$x =$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-6.0707</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1190 + 5.2343i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1190 - 5.2343i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10.3327</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>$A =$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.0000 2.0000 -3.0000 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0000 2.5000 5.0000 -1.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0000 -2.0000 -1.0000 -6.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.0000 -3.0000 -2.0000 -1.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$x =$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.9869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-2.1430</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.6720 + 2.9835i</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-1.6720 - 2.9835i</td>
<td></td>
</tr>
</tbody>
</table>
The applications are oriented to the identification of models of systems (impulse response, mainly) and signals (AR models, mainly). The adaptive behavior is matching/considering tracking problems, so in applications where the stationarity is changing slowly.
Course 5 – Applications of adaptive filtering

1. Identification

In such applications it is clear what the desired signal is. The general identification mechanism is presented in Fig. 1. The adaptive filter is used to provide a model that represents the best fit to an unknown channel. The channel and the adaptive filter are driven by the same input. The channel output is the desired filter response. The quantities of interest are the filter’s parameters.

![Figure 1 – The basic structure of the identification](image)

Examples of such an identification process are related to identification of the echo path in application of echo canceling and identification of radio channels in channel equalization applications.

Note: In the real cases applications a noise must be added to the output of the channel. So the desired channel will be \( d(n) = y(n) + v(n). \)

2. Echo phenomenon

Echo is a phenomenon in which a delayed and distorted replica of an original sound or electrical signal when is reflected back to the source. Acoustic echo is caused by the reflection of the loudspeaker’s sound from ceiling, wall, window, floor and other objects in an enclosed environment, e.g. in a room, back to the microphone as shown in Fig.2. In addition, acoustic echo may also result from a feedback path set up between the speaker and the microphone in teleconference and hearing aid systems.

The effects of an echo depend upon the time delay between the incident and the reflected waves, the strength of the reflected waves, and the number of echo paths through which the waves are reflected. In general, acoustic echo is reflected from many different surfaces and propagates through different paths. If the time delay is not too long, the acoustic echo can be viewed as a soft reverberation which enhances the artistic quality of a musical performance in a concert hall or a church hall. However, the acoustic echo arriving several tens of ms after the direct waves is undesirable and annoying. The echo cancellation system is therefore required when the effect of an echo exceeds an acceptable level.
Naturally, the problem of acoustic echo is quite complicated for a number of reasons:

- The time delay of acoustic echo is too long (up to a second). This implies that a large number of estimated parameters, probably in the order of a few thousands, is needed in the transversal FIR filter to obtain a reasonable approximation.

- The characteristic of an acoustic echo path is rather non-stationary, e.g. by opening or closing of a door, or a moving person.

- The acoustic echo results from reflection of the loudspeaker’s sound from many different paths, e.g. from walls, windows, floors and other objects in an enclosed environment. This is called multipath propagation.

- The echo path is not well approximated by a FIR and IIR linear filter because it has a mixture of linear and non-linear characteristics. The reflection of acoustic signals inside a room is almost linearly distorted, but the loudspeaker produces non-linearity.

![Figure 2 – Echo generation scheme](image)

### 3. Echo Canceller (EC)

An effective way of removing the acoustic echo signal is to use an echo canceller. The EC first estimates the characteristics (or transfer function) of the true echo path, and then generates a replica of the true echo signal that is used to eliminate the true echo signal by subtraction. Since the transfer function of the echo path is generally unknown, an estimate of it must be determined.

The synthetic echo, \( \hat{r}(t) \), is generated by passing the far-end signal \( u(t) \) through the estimated model of the true echo path, \( He \), represents the impulse response function of the true echo path, \( r(t) \) is the true echo signal produced from the true echo path, i.e. \( r(t) = He * u(t) \) where * denotes the convolution sum, and \( x(t) \) is the desired signal from the echo canceller. Therefore, the combined signal \( y(t) = r(t) + x(t) \) is subtracted from the synthetic echo, \( \hat{r}(t) \), to obtain the canceller error signal.
Typically, a good echo canceller yields \( e(t) = x(t) \). Note that, \( x(t) \) can be perceived as the near-end speaker’s speech signal and/or unknown disturbance. To model the echo path correctly, one must avoid to interpret \( x(t) \) as a part of the true echo signal. Generally, \( x(t) \) is assumed for simplicity to be low-level disturbance and uncorrelated with the true echo signal.

In theory, the number of parameters employed in the filter depends upon the echo path delay and the length of the impulse response of an echo path. For the echo canceller to work suitably, the number of parameters should have a length greater than the combined effect of the true echo path’s impulse response length and the echo path delay. Let \( T_s \) be the sampling period of the digitized speech signal, \( M \) be the number of parameters used in the filter, and \( \tau \) be the combined effect to be considered. Then

\[
M \cdot T_s > \tau
\]  

Since the typical value of \( T_s \) is 125 μs (Fs = 8000 Hz) for a standard digital communication system, it is obvious that, if \( \tau = 500 \text{ ms} \), \( M > 4000 \) parameters are required in the filter.

The prototype scheme is presented in Fig. 3. The echo path is to be identified. The goal is to subtract a synthesized version of the echo from another signal (here, picked up by a microphone), so that the resulting signal is “free of echo” and contains only the signal of interest.

![Figure 3 – Echo cancellation scheme](image)

This scheme applies also to hands-free telephony inside a car or teleconferencing in a conference room.

It is assumed that the near-end signal is statistically independent of the far-end signal, which results in an adaptive filter trying to model the echo path as if there were no near-end signals.

**Note:** When this is not the case, the filter weights are adjusted only in those periods when only the far-end party is talking. Such a scheme requires an extra circuit that can detect when the near-end talker is speaking.
4. Interference/noise cancellation

Echo cancellation may be viewed as a special case of the general interference/noise cancellation problem, as it is shown in Figure 4. Roughly speaking, the primary input (desired signal) consists of signal plus noise, and the reference input consists of noise alone.

The noise in the reference signal is generally different from the noise in the primary signal (i.e. the transfer functions from the noise source to the different sensors are not identical), but it is assumed that the noise in the different sensors are correlated. The objectives to use the reference sensor are to reduce the noise in the primary sensor – assuming that the signal and noise are uncorrelated as in the echo cancellation problem.

In the identification context, the adaptive filter may be viewed here a model for the “noise-source-to primary-sensor” transfer function, times the inverse of the “noise-source-to-reference-sensors” transfer function.

An example is shown in Fig. 5, where the aim is to reduce acoustic noise in a speech recording. The reference microphones are placed in a location where there is a sufficient isolation from the source of speech.

Acoustic noise reduction is particularly useful when low bit rate coding scheme (e.g. LPC) are used for digital representation of the speech signal. Such audiocodecs are very sensitive to the presence of background noise, often leading to unintelligible digitized speech.

Another important application of this scheme is main electricity interference cancellation (i.e. removal of 50-60 Hz sin waves), where the references signal is taken from a wall outlet.
Figure 5 – Acoustic interference cancellation example
Course 6 – Least-Squares Adaptive Filters

Content

Method of Least Squares. Problem formulation.
The principle of orthogonallity
Normal equations
Application: Adaptive Equalizer

Summary


The method of Least Squares (LS) may be viewed as an alternative to Wiener filter theory. Wiener Filters are derived from ensemble averages, with the result that one filter (optimum) is obtained for all realizations of the operational environment.

The LS method is deterministic in approach. The method is solving the linear filtering problem without invoking assumptions on the statistics of the input applied to the filter. The LS method is based on the minimization of the sum of the squares of the difference between two signals, \( u(i) \) and \( d(i) \), \( i = 1, 2, \ldots, N \).

Let a multiregressive linear model of a signal as

\[
y(i) = \sum_{k=0}^{M-1} w_o^*(k) \cdot u(i-k) + e_o(i)
\]  

(1)

where \( \{w_{ok} = w_o(k)\} \) are unknown parameters of the model and \( e_0(i) \) represents the measurement error to which the statistical nature of the phenomenon is described. The measurement error \( e_0(i) \) is an unobservable random variable that is introduced into the model to account for its accuracy. It is assumed that \( e_0(i) \) is white with zero mean and variance \( \sigma^2 \). That is

\[
E\{e_0(i)\} = 0, \quad \forall i
\]

(2.a)

\[
E\{e_0(i) \cdot e_0^*(k)\} = \begin{cases} 
\sigma^2, & i = k \\
0, & i \neq k 
\end{cases}
\]

(2.b)

The effect of the considerations made on measurement noise is that we may write:

\[
E[y(i)] = \sum_{k=0}^{M-1} w_o^* \cdot E[u(i-k)] + E[e_o(i)] = \sum_{k=0}^{M-1} w_o^* \cdot E[u(i-k)]
\]

(3)

The model (1) could be also considered as a model for a linear transversal filter, with \( w_{ok} \) the weights of the taps (i.e. the sample of the impulse response of the filter) and with an addition of a signal \( d(i) \) as desired signal.
The problem to solve is to estimate the unknown parameters \( \{w_{ok}\} \) of the multiple linear regression model (1), given two sets of variables: \( \{u(i)\} \) and \( \{y(i)\} \), \( i=1,2,\ldots,N \). The estimation error is

\[
e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} w_k^* \cdot u(i - k)
\]

(4)

In the method of LS, we choose the weights \( \{w_k\} \) of the transversal filter so as to minimize a cost function that consists of the sum of error squares:\footnote{also called error energy}:

\[
J(w) = \sum_{i=1}^{i_2} |e(i)|^2
\]

(5)

where \( i_1 \) and \( i_2 \) define the index limits at which the error minimization occurs. The values assigned to these limits depend on data windowing employed. For the minimization of criterion \( J \), the tap-weights of the filter \( w = [w_0 \ w_1 \ \ldots \ w_{M-2} \ w_{M-1}]^T \) are held constant during the interval \( i_1 \leq i \leq i_2 \). The filter resulting from the minimization is termed a linear least-squares filter.

2. The principle of orthogonality

We make the assumptions that \( i_1 = M, i_2 = N \) and no assumptions about the data outside the interval \([M, N]\). By such considerations, we make sure that, for each value of \( i \), all the tap inputs of the transversal filter have nonzero values. The cost function, or the sum of the error squares, is:

\[
J(w) = \sum_{i=M}^{N} |e(i)|^2 = \sum_{i=M}^{N} e(i) \cdot e^*(i)
\]

(6)

In the general case, we must write complex values for weights:

\[
w_k = a_k + jb_k, \quad k = 0,1,2,\ldots,M - 1
\]

(7)

and the estimation error is:

\[
e(i) = d(i) - y(i) = d(i) - \sum_{k=0}^{M-1} (a_k - jb_k) \cdot u(i - k)
\]

(8)

We define the \( k \)th component of the gradient vector \( \nabla J \) as the derivative of the cost function \( J(w) = J(w_0, w_1, \ldots, w_{M-1}) \), with respect to the real and imaginary part of tap-weight \( w_k \):
\[ \nabla_k J = \frac{\partial J}{\partial a_k} + j \frac{\partial J}{\partial b_k}, \quad k = 0,1,2,\ldots,M - 1 \quad (9) \]

Then, we get:
\[ \nabla_k J = -\sum_{i=M}^{N} \left[ e(i) \cdot \frac{\partial e^*(i)}{\partial a_k} + e^*(i) \cdot \frac{\partial e(i)}{\partial a_k} + je(i) \cdot \frac{\partial e^*(i)}{\partial b_k} + je^*(i) \cdot \frac{\partial e(i)}{\partial b_k} \right] = \ldots = \]
\[ = -2 \cdot \sum_{i=M}^{N} u(i-k) \cdot e^*(i) \quad (10) \]

For the minimization of the cost function \( J(w) = J(w_0, w_1, \ldots, w_{M-1}) \) with respect to the tap weights of the transversal filter, we require that the following conditions be satisfied simultaneously:
\[ \nabla_k J = 0, \quad k = 0,1,2,\ldots,M - 1 \quad (11) \]

Let \( e_{\text{min}}(i) \) denote the value of the estimation error \( e(i) \) that results when the cost function \( J(w) \) is minimized in accordance with Eq. (11). From Eq. 10, the set of conditions (11) is equivalent to the formula:
\[ \sum_{i=M}^{N} u(i-k) \cdot e_{\text{min}}^*(i) = 0, \quad k = 0,1,2,\ldots,M - 1 \quad (12) \]

By considering the time-average of the left-hand side of Eq. (12) we obtain the cross-correlation between the tap input \( u(i-k) \) and the minimum estimation error \( e_{\text{min}}(i) \) over the values of time \( (i) \) in the interval \([M,N]\), for a fixed value of \( k \). Accordingly, we may state the principle of orthogonality as follows:

The minimum-error time series \( e_{\text{min}}(i) \) is orthogonal to the time series \( u(i-k) \) applied to tap \( k \) of a transversal filter of length \( M \), for \( k=0,1,\ldots,M-1 \), when the filter is operating in its least-squares conditions.

This principle provides the basis of a simple test that we can carry out in practice to check whether or not the transversal filter is operating in its least-squares condition. We merely have to determine the time-average cross-correlation between the estimation error and the time series applied to each tap input of the filter. It is only when all these \( M \) cross-correlation functions are identically zero that we find that the cost function \( J(w) = J(w_0, w_1, \ldots, w_{M-1}) \) is minimal.

**Corollary**

Let \( \{\hat{w}_j\} \) denote the special values of the tap-weights \( \{w_j\} \) that result when the transversal filter is optimized to operate in its least-squares condition. The filter output is
This output provides a least-squares estimate of the desired response \( d(i) \).

Let \( U_M(i) \) denote the vector (sequence) of \([u(i), u(i-1), \ldots, u(i-M-1)]\). Let \( \hat{d}(i/U_M(i)) \) denote the least-squares estimate of the desired response \( d(i) \), given the tap inputs described by \( U_M(i) \). We may then write:

\[
\hat{d}(i/U_M(i)) = y_{\min}(i) = \sum_{k=0}^{M-1} \hat{w}_k^* \cdot u(i-k)
\]

Multiplying both sides of Eq. (12) by \( \hat{w}_k^* \) and then sum the result over the values of \( k \) in the interval \([-M, 0]\), we get:

\[
\sum_{i=M}^{N} \sum_{k=0}^{M-1} \hat{w}_k^* \cdot u(i-k) \cdot e_{\min}^*(i) = 0
\]

The summation term inside the square brackets on the left-hand side of (15) is recognized to be the least-squares estimate \( \hat{d}(i/U_M(i)) \). So,

\[
\sum_{i=M}^{N} \hat{d}(i/U_M(i)) \cdot e_{\min}^*(i) = 0
\]

The last equation is the mathematical description of the corollary to the principle of orthogonality:

When a transversal filter operates in its least-squares condition, the least-squares estimate of the desired response, produced at the filter output and represented by time series, \( \hat{d}(i/U_M(i)) \), and the minimum estimation error time series \( e_{\min}(i) \) are orthogonal to each other over time \( i \).

3. Normal Equations

The principle of orthogonality (Eq.12) is formulated in terms of a set of tap-inputs and the minimum estimation error, \( e_{\min}(i) \). Setting the tap weights in Eq. (4) to their least-squares values, we get:

\[
e(i) = d(i) - y(i) = d(i) - \sum_{t=0}^{M-1} w_t^* \cdot u(i-t)
\]

Substituting (17) in (12) and rearranging terms, we get a system of \( M \) simultaneous equations:
The two summations in (18) involving the index \( i \) represent time averages, except for a scaling factor. They have the following interpretations:

1). The time average (over \( i \)) on the left-hand side of the equation represent the time-average autocorrelation function of the tap inputs in the linear transversal filter. In particular, we may write:

\[
\Phi(t,k) = \sum_{i=M}^{N} u(i-k) \cdot u^*(i-t), \quad 0 \leq (t,k) \leq M - 1
\]  

(19)

2). The time average (also over \( i \)) on the right-hand side of Eq. (18) represents the time average cross-correlation between the tap inputs and the desired response. In particular, we may write:

\[
z(-k) = \sum_{i=M}^{N} u(i-k) \cdot d^*(i), \quad 0 \leq (t,k) \leq M - 1
\]  

(20)

We may write the system of simultaneous equations as:

\[
\sum_{t=0}^{M-1} w^*_t \cdot \Phi(t,k) = z(-k), \quad 0 \leq k \leq M - 1
\]  

(21)

The system of Eq. (21) represents the expanded system of the normal equations for linear least-squares filters.

The matrix formulation

We introduce first some definitions:

1). The \( M \)-by-\( M \) time-average correlation matrix of the tap inputs \( \{u(i), u(i-1), \ldots, u(i-M+1)\} \) is:

\[
\Phi = \begin{bmatrix}
\Phi(0,0) & \Phi(0,1) & \ldots & \Phi(0,M-1) \\
\Phi(1,0) & \Phi(1,1) & \ldots & \Phi(1,M-1) \\
\vdots & \vdots & \ddots & \vdots \\
\Phi(M-1,0) & \Phi(M-1,1) & \ldots & \Phi(M-1,M-1)
\end{bmatrix}_{M \times M}
\]  

(22)

2). The \( M \)-by-1 time average cross-correlation vector between the tap inputs \( \{u(i), u(i-1), \ldots, u(i-M+1)\} \) and the desired response \( d(i) \):

\[
z = r_{ud} = E[u(i) \cdot d^*(i)] = [z(0) \quad z(-1) \quad \ldots \quad z(-M+1)]^T
\]  

(23)

3). The \( M \)-by-1 tap weight vector of the least-squares filter:

\[
\hat{w} = [\hat{w}_0 \quad \hat{w}_1 \quad \ldots \quad \hat{w}_{M-1}]^T
\]  

(24)
We may rewrite the system of $M$ simultaneous equations (21) in terms of these matrix definitions simply as:

$$\Phi \cdot \hat{w} = z$$  

(25)

which is the matrix form of the normal equations for linear-squares filters. Assuming that $\Phi$ is nonsingular, we may solve (25) for the tap-weight vector of the linear least-squares filter:

$$\hat{w} = \Phi^{-1} \cdot z$$  

(26)

Eq. (26) is a very important result: it is the linear least-squares counter-part to the solution of the matrix form of the Wiener-Hopf equations. More, this equation is fundamental to the development of all recursive formulations of the linear least-squares filters.

**Note 1:** The time average correlation matrix has as generic elements $\Phi(t,k)$. The index $k$ refers to the row number in the matrix $\Phi$ and the index $t$ refers to the column number. Let $M$-by-1 tap-input vector:

$$u(i) = [u(i), u(i-1), ..., u(i-M + 1)]^T$$  

(27)

Then, we may write\(^{15}\):

$$\Phi = \sum_{i=M}^{N} u(i) \cdot u^H(i)$$  

(28)

In the statistics literature, the scaled form of $\Phi$ is referred to as the sample correlation matrix. The matrix $\Phi$ has some useful properties:

1). The correlation matrix $\Phi$ is Hermitian:

$$\Phi^H = \Phi$$  

(28)

2). The correlation matrix $\Phi$ is nonnegative definite, that is

$$x^H \cdot \Phi \cdot x \geq 0$$  

(29)

for any $M$-by-1 vector $x$.

3). The correlation matrix is the product of two rectangular Toeplitz matrices that are the Hermitian transpose of each other:

\(^{15}\) The summation (28) must be divided by the scaling factor $(N-M+1)$ for the correlation matrix $\Phi$ to be a time average in precise terms.
The matrix $A$ is a Toeplitz matrix and is called data matrix.

**Note 2:** We may reformulate the normal equations in terms of data matrices. We introduce also the desired vector, $d$, consisting of the desired response $d(i)$ for values of $i$ in the interval $[M,N]$:

$$d^H = [d(M) \ d(M+1) \ \ldots \ d(N)] \quad (31)$$

Based on relations:

$$z = A^H \cdot d \quad (32)$$

and

$$A^H \cdot A \cdot \hat{w} = A^H \cdot d \quad (33)$$

we obtain:

$$\hat{w} = (A^H \cdot A)^{-1} \cdot A^H \cdot d \quad (34)$$

**Note 3:** Properties of least-squares estimates.

**P1:** The least-squares estimate $\hat{w}$ is unbiased, if the measurement error process $e_0(i)$ has zero mean:

$$E[\hat{w}] = w_0 \quad (35)$$

**P2:** When the measurement error process $e_0(i)$ is white with zero-mean and variance $\sigma^2$, the covariance matrix of the least-squares estimate $\hat{w}$ equals $\sigma^2 \cdot \Phi^{-1}$:

$$\text{cov}(\hat{w}) = E[(\hat{w} - w_0) \cdot (\hat{w} - w_0)^H] = \sigma^2 \cdot \Phi^{-1} \quad (36)$$

**P3:** When the measurement error process $e_0(i)$ is white with zero-mean, the least-squares estimate $\hat{w}$ is the best linear unbiased estimate (BLUE).

**Note 4:** The criterion (5) has a minimum value – that is, the minimum sum of error squares, $J_{\text{min}}$. Based on relation

$$e_{\text{min}}(i) = d(i) - \hat{d}(i / U_M(i)) \quad (37)$$
and by considering the principle of orthogonality we may write

\[ E_{\min} = E_d - E_{est} \]  

where

\[ E_d = \sum_{i=M}^{N} |d(i)|^2 \]  

is the energy of the desired response.

\[ E_{est} = \sum_{i=M}^{N} |\hat{d}(i/U_M(i))|^2 \]  

is the energy of the estimated desired response

\[ E_{\min} = \sum_{i=M}^{N} |e_{\min}(i)|^2 \]  

is the energy minimum estimation error.

The following relations are valid:

\[ E_{\min} = d^H \cdot d - z^H \cdot \Phi^{-1} \cdot z \]  

\[ E_{\min} = d^H \cdot d - d^H \cdot A \cdot \left(A^H \cdot A\right)^{-1} \cdot A^H \cdot d \]  

**Summary**

The method of least-squares for solving the linear adaptive filtering problem through the use of a batch (block) processing approach was considered.

The distinguishing features of this approach are:

- It is a **model-dependent** that operates on the input data on a block-by-block basis; the model on which the approach is based is a multi-parameter linear regression model.

- It yields a solution for the tap-weight vector of a transversal filter that is the **best linear unbiased estimate (BLUE)**, assuming that the measurements error process in the multi-parameter regression model is white with zero mean.

The method of least-squares is well suited for solving **spectrum estimation problems**, such as those based on autoregressive (AR) models.
Course 6 – Recursive Least-Squares Adaptive Filters

Content
Preliminaries
Reformulation of the normal equations
Recursive computation of correlation matrices
Recursive Least-Squares Adaptive Filters
Application: Adaptive Equalizer

Summary

1. Preliminaries

Let us consider the problem of design of a transversal filter, given input data consisting of a tap-input vector \( \mathbf{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T \) and the corresponding desired response \( d(n) \) for varying \( n \). The requirement is to estimate the unknown parameter vector \( \mathbf{w} \) of a multiple regression model that relates \( d(n) \) to \( \mathbf{u}(n) \).

In order to use some (prior) information about the input-output mapping, the cost function is expanded with some information about \( \mathbf{w} \). Thus, it has two components:

\[
J(n) = \sum_{i=1}^{n} \beta(n,i) \cdot |e(i)|^2 + \delta \cdot \lambda^n \cdot \| \mathbf{w}(n) \| \quad (1)
\]

The two components of the cost function are as follows:

1). The sum of weighted error squares:

\[
\sum_{i=1}^{n} \beta(n,i) \cdot |e(i)|^2 = \sum_{i=1}^{n} \beta(n,i) \cdot |d(i) - \mathbf{w}^H(n) \cdot \mathbf{u}(i)|^2 \quad (2)
\]

which is data dependent. There is a weighting factor

\[
0 < \beta(n,i) \leq 1, \quad i = 1,2,\ldots,n \quad (3)
\]

which is used to ensure that data in the distant past are “forgotten” in order to afford the possibility of following the statistical variations when the filter operates in a non-stationary environment.

A special case of weighting that is commonly used is the exponential weighting factor, or forgetting factor, defined by

\[
\beta(n,i) = \lambda^{n-i}, \quad i = 1,2,\ldots,n \quad (4)
\]

where \( \lambda \) is a positive constant close to, but less than, unity. The inverse of \( \frac{1}{1-\lambda} \) is a measure of the memory of the algorithm. When \( \lambda = 1 \) we have the ordinary method of least squares and correspond to infinite memory case. This component measures the exponentially weighted error between the desired response \( d(i) \) and the actual response of the filter, \( y(i) \).
2). The second term is called regularization term:

\[ \delta \cdot \lambda^n \cdot \|w(n)\| = \delta \cdot \lambda^n \cdot w^H (n) \cdot w(n) \]  \hspace{1cm} (5)

where \( \delta \) is a positive real number called the regularization parameter. Except for the factor, the regularizing term depends only on the tap weight vector \( w(n) \).

The term is included to stabilize the solution to the recursive least-squares by smoothing the solution.

2. Reformulation of the normal equations

Considering Eq. (1), the effect of including the regularizing term \( \delta \cdot \lambda^n \cdot \|w(n)\| \) in the cost function \( E(n) \) is equivalent to a reformulation of the \( M \text{-by-} M \) time average correlation matrix\(^{16} \) of the tap-input vector \( u(i) \):

\[ \Phi(n) = \sum_{i=1}^{n} \lambda^{n-i} \cdot u(i) \cdot u^H (i) + \delta \cdot \lambda^n \cdot I \]  \hspace{1cm} (6)

where \( I \) is the \( M \text{-by-} M \) identity matrix.

The \( M \text{-by-} 1 \) time-average cross-correlation vector \( z(n) \) between the tap inputs of the transversal filter and the desired response is unaffected by the use of regularization:

\[ z(n) = \sum_{i=1}^{n} \lambda^{n-1} \cdot u(i) \cdot d * (i) \]  \hspace{1cm} (7)

According to the method of least-squares, the optimum value of the \( M \text{-by-} 1 \) tap-weight vector \( \hat{w}(n) \), for which the cost function \( J(n) \) of Eq. (1) attains its minimum value, is defined by the normal equations. For the recursive least-squares problem, the normal equations are written in matrix form as:

\[ \Phi(n) \cdot \hat{w}(n) = z(n) \]  \hspace{1cm} (8)

where \( \Phi(n) \) and \( z(n) \) are now defined by Eqs. (6) and (7), respectively.

\textbf{Note}: Addition of the regularizing term has the effect of making the correlation matrix \( \Phi(n) \) nonsingular at all stages of the computation, starting from \( n = 0 \).

3. Recursive computation of \( \Phi(n) \) and \( z(n) \)

Isolating the term corresponding to \( i = n \) from the rest of the summation on the right-hand side of Eq. (6), we may write:

\[^{16} \text{A correlation matrix modified as in Eq. (6) is said to be diagonally loaded.} \]
\[ \Phi(n) = \lambda \sum_{i=1}^{n-1} \lambda^{n-1-i} \cdot u(i) \cdot u^H(i) + \delta \cdot \lambda^{n-1} \cdot I \] + u(n) \cdot u^H(n) = \]

\[ \Phi(n) = \lambda \cdot \Phi(n-1) + u(n) \cdot u^H(n) \] (9)

Similarly, we may write

\[ z(n) = \lambda \cdot z(n-1) + u(n) \cdot d^*(n) \] (10)

The computation of the LS estimate \( \hat{w}(n) \) for the tap-weight vector in accordance with (6), involves the decomposition of the inverse of the correlation matrix \( \Phi(n) \), which could be difficult, particularly if the number of tap weights, \( M \), is high.

**The matrix inversion lemma**

Let \( A \) and \( B \) two positive-definite \( M \)-by-\( M \) matrices related by:

\[ A = B^{-1} + C \cdot D^{-1} \cdot C^H \] (11)

where \( D \) is a positive-definite \( N \)-by-\( M \) matrix and \( C \) is an \( M \)-by-\( N \) matrix. According to the matrix inversion lemma, we may express the inverse of the matrix \( A \) as:

\[ A^{-1} = B - B \cdot C \cdot \left( D + C^H \cdot B \cdot C \right) \cdot C^H \cdot B \] (12)

The exponentially weighted recursive least-squares algorithm uses the correlation matrix \( \Phi(n) \), assumed non-singular and therefore invertible; we may apply the matrix inversion lemma to the recursive equation (9). First, we must make some identification:

\[ A = \Phi(n) \] (13.a)

\[ B^{-1} = \lambda \cdot \Phi(n-1) \] (13.b)

\[ C = u(n) \] (13.c)

\[ D = 1 \] (13.d)

Then, by substituting these definitions into the matrix inversion lemma, we obtain the following recursive equation for the inverse of the correlation matrix:

\[ \Phi^{-1}(n) = \lambda^{-1} \cdot \Phi^{-1}(n-1) - \frac{\lambda^{-2} \cdot \Phi^{-1}(n-1) \cdot u(n) \cdot u^H(n) \cdot \Phi^{-1}(n-1)}{1 + \lambda^{-1} \cdot u^H(n) \cdot \Phi^{-1}(n-1) \cdot u(n)} \] (14)

For the convenience of the computation, let

\[ P(n) = \Phi^{-1}(n) \] (15.a)

and
Then, we may rewrite Eq. (14) as:

\[ P(n) = \lambda^{-1} \cdot P(n-1) - \lambda^{-1} \cdot k(n) \cdot u^H(n) \cdot P(n-1) \]

The \( M \)-by-\( M \) matrix \( P(n) \) is referred as the inverse correlation matrix. The \( M \)-by-1 vector \( k(n) \) is referred to as gain vector. Eq. (16) is the Riccati equation for the RLS algorithm.

### 4. Time update of the tap-weight vector

The LS estimate of the tap-weight vector at iteration \( n \) is:

\[
\hat{w}(n) = \Phi^{-1}(n) \cdot z(n) = P(n) \cdot z(n) = \lambda \cdot P(n) \cdot z(n-1) + P(n) \cdot u(n) \cdot d^*(n)
\]

By making some simple processing step, we get:

\[
\hat{w}(n) = \hat{w}(n-1) + k(n) \cdot [d^*(n) - u^H(n) \cdot \hat{w}(n-1)] = \\
= \hat{w}(n-1) + k(n) \cdot \xi^*(n)
\]

where

\[
\xi(n) = d(n) - u^T(n) \cdot \hat{w}^*(n-1) = d(n) - \hat{w}^H(n-1) \cdot u(n)
\]

is the a priori estimation error and

\[
k(n) = \frac{\lambda^{-1} \cdot P(n-1) \cdot u(n)}{1 + \lambda^{-1} \cdot u^H(n) \cdot P(n-1) \cdot u(n)} = \frac{\pi(n)}{\lambda + u^H(n) \cdot \pi(n)}
\]

is the gain matrix and

\[
P(n) = \lambda^{-1} \cdot P(n-1) - \lambda^{-1} \cdot k(n) \cdot u^H(n) \cdot P(n-1)
\]

is the estimation of the inverse of the correlation matrix at step \( n \).

Eq. (18) represents the Recursive Least-Square algorithm with a block-diagram computation depicted in Fig. 1. The a priori estimation error \( \xi(n) \) is, in general, different from the a posteriori estimation error:

\[
e(n) = d(n) - \hat{w}^H(n) \cdot u(n)
\]

\( \xi(n) \) may be viewed as a “tentative” value of \( e(n) \) before updating the tap-weight vector.
**Summary**

An important (main) feature of the RLS algorithm is that the inversion of the correlation matrix $\Phi(n)$ is replaced at each step by a simple scalar division.

The RLS algorithm could be summarized as:

$$
\begin{align*}
\mathbf{w}(0) &= \mathbf{0} \quad \text{or} \quad \mathbf{w}(0) = \text{random small values}; & \text{Size:} & \quad M \times 1 \\
\mathbf{P}(0) &= \delta^{-1} \cdot \mathbf{I}; & & \quad M \times M \\
\text{#1. Initializations:} & \\
0 < \delta &= \begin{cases} 
\text{small for high SNR} \\
\text{large for low SNR}
\end{cases} & \text{1 x 1} \\
\delta \text{ is called regularization parameter.} \\
\text{Example: RSZ < 10 dB ---&gt; delta = 10} \\
n = 0; & \quad \lambda = 0.1...0.9 \\
\text{#2. LOOP n:} & \\
& \quad n = n+1; \\
\text{#2.1. Compute:} & \quad \pi(n) = \mathbf{P}(n-1) \cdot \mathbf{u}(n) & \quad M \times 1 \\
\text{#2.2. Compute gain:} & \quad \mathbf{k}(n) = \pi(n) \left( \lambda + \mathbf{u}^H(n) \cdot \pi(n) \right) & \quad M \times 1 \\
\text{#2.3. Estimate error:} & \quad \xi(n) = d(n) - \hat{w}^H(n-1) \cdot \mathbf{u}(n) & \quad 1 \times 1 \\
\text{#2.4. Adaptation:} & \quad \hat{w}(n) = \hat{w}(n-1) + \mathbf{k}(n) \cdot \xi^*(n) & \quad M \times 1 \\
\text{#2.5. Compute:} & \quad \mathbf{P}(n) = \lambda^{-1} \cdot \mathbf{P}(n-1) - \lambda^{-1} \cdot \mathbf{k}(n) \cdot \mathbf{u}^H(n) \cdot \mathbf{P}(n-1) & \quad M \times M
\end{align*}
$$
UNTIL $J(n) < J_{\text{min}}$

Two-stage computation of the gain $k(n)$ is preferred over the direct computation of $k(n)$, from arithmetic point of view.

The recursive least-squares (RLS) algorithm is a natural extension of the method of least squares. The derivation was based on a lemma in matrix algebra known as the matrix inversion lemma.

The fundamental difference between the RLS algorithm and the LMS algorithm may be stated as follows:

- the step-size parameter $\mu$ in the LMS algorithm is replaced by $\Phi^{-1}(n)$, that is, the inverse of the correlation matrix of the input vector $u(n)$, which has the effect of whitening the tap inputs. This modification has a strong impact on the convergence behavior for the RLS algorithm for a stationary environment:
  - The rate convergence of the RLS algorithm is typically an order of magnitude faster than that of LMS algorithm.
  - The rate of convergence of the RLS algorithm is invariant to the eigenvalue spread (i.e., condition number) of the ensemble-average correlation matrix $R$ of the input vector $u(n)$.
  - The excess mean-square error of the RLS algorithm converges to zero as the number of iterations, $n$, approaches infinity, if the exponential weighting factor $\lambda = 1$. 


Course 6 – Applications of adaptive filtering

= Channel equalization =

1. Description of the problem

In mobile communications, channel dispersion introduces intersymbol interference, which means that the signal which is received (by a mobile phase, e.g.) is a filtered version of the original transmitted symbol sequence. It is then the receivers job to figure out what signal was transmitted, and to turn it into understandable information. If everything goes well, the information the receiver delivers should coincide with the information fed into the transmitter.

To correct the channel distortions, equalizations are used at the receiver. A solution for this situation requires a (FIR) model of the channel distortion. Such a model is obtained by adaptive identification. In this case, a training (fixed) sequence is used. The receiver thus sees the channel output and knows the corresponding channel input, so it has everything it needs to do the system (channel) identification.

Because a radio-channel is often highly time-varying, the channel identification has to be repeated continuously. For example, the GSM system is based on burst mode communication, where burst of 148 bits each are transmitted. Each burst has a 26 bit training sequence, which is used to update the channel model. The training bits thus constitute a significant portion (17.5 %) of the transmitted sequence.

Note: There are also identification algorithms which are working without training bits. Such class of algorithms is referred to as “blind identification/equalization”.

Figure 1 – Radio channel identification in mobile communications

\[\text{Training sequence } (a_k) \ldots 01100100 \ldots\]

\[\text{Radio channel}\]

\[\text{Base Station Antenna}\]

\[\text{Mobile receiver}\]

\[\text{Training sequence } \begin{bmatrix} a_k \end{bmatrix} \ldots 01100100 \ldots\]

\[\text{Adaptive filter}\]

\[\text{What to do with the training bits}\]

\[\text{Decision-directed mode}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]

\[\text{What to do with the training bits}\]
2. Problem formulation

Let consider a linear dispersive channel that produces (unknown) distortions. We assume real valued data. Figure 2 shows the block diagram of the system used in this experiment.

The adaptive equalizer has the task of correcting for the distortion produced by the channel in the presence of additive white noise.

The first random number generator provides the test signal \( x(n) \) used for probing the channel. This is a common choice in system identification, in order to excite all the dynamics of the considered system. The signal \( \{x(n)\} \) is a random binary signal with values from the set \{-1, +1\}. Generator #1 supplies the desired sequence applied to the adaptive equalizer in the form of training sequence.

The second random generator serves as the source of additive white noise \( v(n) \) that corrupts the channel output. This generator has lower values then the previous one and has continuous values.

The two generators are independent of each other.

\[
\begin{align*}
\text{Data Source} & \rightarrow U(z) \rightarrow Y(z) \rightarrow X(z) \rightarrow \text{End User} \\
& \rightarrow \text{Channel} \rightarrow H_c(z) \rightarrow H_e(z) \rightarrow X(z) \\
\end{align*}
\]

If \( X(z) = U(z) \), then \( H_e(z) = \frac{1}{H_c(z)} = H_c(z)^{-1} \)

Figure 1 – The principle of the channel equalization

The random sequence \( x(n) \) consists of a Bernoulli sequence\(^{18} \), with values on \{-1, +1\} set and with zero mean and unit variance.

The sequence \( v(n) \) has zero mean and the variance \( \sigma_v^2 = 0.001 \). The impulse response of the channel is described by the raised cosine impulse response function\(^{19} \):

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

with the parameter \( W \) controls the amount of amplitude distortion produced by the channel, with the distortion increasing with \( W \). Equivalently, \( W \) controls the eigenvalue spread \( \chi(R) \) of the correlation matrix of the tap inputs in the equalizer, with the eigenvalues spread increasing with \( W \).

\(^{18}\) A special sequence used in system identification

\(^{19}\) Functie in cos ridicat.
Figure 2: Block-diagram of adaptive equalizer experiment: Structure and basic signals

Figure 3: Results of the equalization
About the symmetry of the impulse response function

The equalizer has $M=11$ taps. Since the channel has an impulse response $h(n)$ that is symmetric about time $n=2$, as depicted in Fig.3, it follows that the optimum tap weights $w_{on}$ of the equalizer are likewise symmetric about time $n=5$. Accordingly, the channel input $x(n)$ is delayed by $\Delta = 2 + 5 = 7$ samples to provide the desired response for the equalizer. By selecting the delay $\Delta$ to match the midpoint of the transversal equalizer, the algorithm is enabled to provide an approximate inversion of both minimum-phase and non-minimum-phase component of the channel response.

![Figure 3 – Impulse responses of the channel](image)

![Figure 4 – Impulse response functions](image)

3. Preliminary analysis: Correlation matrix of the Equalizer Input

$^{20}$ The group delay of a filter is a measure of the average delay of the filter as a function of frequency. It is defined as the first negative derivative of a filter’s phase response.
The first input of the equalizer at time $n$ is

$$u(n) = (h \otimes x)(n) + v(n) = 
\sum_{k=1}^{3} h(k) \cdot x(n-k) + v(n)$$  \hspace{1cm} (2)

The correlation matrix of the input of the equalizer is a symmetric 11-by-11 matrix. Since the impulse response $h(n)$ has nonzero values for $n=1,2,3$, and the noise process $v(n)$ is white with zero mean and variance $\sigma_v^2$, the correlation matrix $R$ is \textit{quint-diagonal}. That is, the only nonzero elements of $R$ are on the main diagonal and the four diagonals directly above and below it, two on either side, as shown by the special structure:

$$R = \begin{bmatrix}
    r(0) & r(1) & r(2) & 0 & \ldots & 0 \\
    r(1) & r(0) & r(1) & r(2) & \ldots & 0 \\
    r(2) & r(1) & r(0) & r(1) & \ldots & 0 \\
    0 & r(2) & r(1) & r(0) & \ldots & 0 \\
    \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & 0 & \ldots & r(0)
\end{bmatrix}$$  \hspace{1cm} (3)

where

$$r(0) = h^2(1) + h^2(2) + h^2(3) + \sigma_v^2$$  \hspace{1cm} (3.a)

$$r(1) = h(1) \cdot h(2) + h(2) \cdot h(3)$$  \hspace{1cm} (3.b)

$$r(2) = h(1) \cdot h(3)$$  \hspace{1cm} (3.c)

In table 1, a list of the autocorrelation functions are presented, the smallest and the biggest eigenvalue$^{21}$ and the \textit{eigenvalue spread} $\chi(R) = \lambda_{\text{max}} / \lambda_{\text{min}}$.

---

$^{21}$ Eigenvalues are a special set of scalars associated with a linear system of equations (i.e., a matrix equation) that are sometimes also known as characteristic roots, characteristic values, proper values, or latent roots. The determination of the eigenvalues and eigenvectors of a system is extremely important in physics and engineering, where it is equivalent to matrix diagonalization. Each eigenvalue is paired with a corresponding so-called eigenvector (or, in general, a corresponding right eigenvector and a corresponding left eigenvector; there is no analogous distinction between left and right for eigenvalues).

The decomposition of a square matrix $A$ into eigenvalues and eigenvectors is known in this work as \textit{eigen decomposition}, and the fact that this decomposition is always possible as long as the matrix consisting of the eigenvectors of $A$ is \textit{square} is known as the \textit{eigen decomposition theorem}.

Let $A$ be a linear transformation represented by a matrix $A$. If there is a vector $X \in \mathbb{R}^n \neq 0$ such that $A \cdot X = \lambda \cdot X$ for some scalar $\lambda$, then $\lambda$ is called the eigenvalue of $A$ with corresponding (right) eigenvector $X$. 
<table>
<thead>
<tr>
<th>$W$</th>
<th>2.9</th>
<th>3.1</th>
<th>3.3</th>
<th>3.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r(0)$</td>
<td>1.0963</td>
<td>1.1568</td>
<td>1.2264</td>
<td>1.3022</td>
</tr>
<tr>
<td>$r(1)$</td>
<td>0.4388</td>
<td>0.5596</td>
<td>0.6729</td>
<td>0.7774</td>
</tr>
<tr>
<td>$r(2)$</td>
<td>0.0481</td>
<td>0.0783</td>
<td>0.1132</td>
<td>0.1511</td>
</tr>
<tr>
<td>$\lambda_{\text{min}}$</td>
<td>0.3339</td>
<td>0.2136</td>
<td>0.1256</td>
<td>0.0656</td>
</tr>
<tr>
<td>$\lambda_{\text{max}}$</td>
<td>2.0295</td>
<td>2.3761</td>
<td>2.7263</td>
<td>3.0707</td>
</tr>
<tr>
<td>$\chi(R) = \lambda_{\text{max}} / \lambda_{\text{min}}$</td>
<td>6.0782</td>
<td>11.1238</td>
<td>21.7132</td>
<td>46.8216</td>
</tr>
</tbody>
</table>
Course Notes - 2009

Course 7 - Kalman Filters

Content

Recursive minimum mean –square estimation for scalar random variables
Statement of the Kalman filtering problem
The innovations process
Correlation matrix of the innovation process
Estimation of the state using the innovation process
Filtering
Filtered Estimation Error
Initial conditions
Summary

The Kalman filter is a set of mathematical equations that provides an efficient computational (recursive) means to estimate the state of a process, in a way that minimizes the mean of the squared error.

The filter is very powerful in several aspects: it supports estimations of past, present, and even future states, and it can do so even when the precise nature of the modeled system is unknown.

A distinctive feature of the Kalman filter is that its mathematical formulation is described in terms of the state-space concepts.

Another novel feature of the Kalman filter is that its solution is computed recursively, applying without modification to stationary as non-stationary environments.

1. Recursive minimum mean –square estimation for scalar random variables

Let assume that, on the basis of a complete set of observed random variables

\[ Y(n-1) = \{y(1), y(2), \ldots, y(n-1)\} \],

starting with the first observation at time 1 and extending up to and including time \((n-1)\), we have found the minimum mean-square estimate \(x(n-1/Y(n-1))\) of a related zero-mean random variable \(x(n-1)\).

We are assuming that the observation at (or before) \(n=0\) is zero. The space spanned by the observations \(\{y(1), y(2), \ldots, y(n-1)\}\) is denoted by \(Y(n-1)\).

Let \(y(n)\) an additional observation at time \((n)\) and let be the requirement to compute an updated estimate \(\hat{x}(n/Y(n))\) of the random variable \(x(n)\).

**Innovation.** The forward prediction error

\[ e_{n-1}(n) = y(n) - \hat{y}(n/Y(n-1)), \quad n = 1, 2, \ldots \] (1)

where \(\hat{y}(n/Y(n-1))\) is the one step prediction of the observed variable \(y(n)\) at time \(n\) using all past observations available up to and include time \((n-1)\). The order of the prediction equals \((n-1)\) because the number of the past observations used in this estimation is \((n-1)\).

According to the principle of orthogonality, the prediction error \(e_{n-1}(n) = f_{n-1}(n)\) is orthogonal to all past observations \(\{y(1), y(2), \ldots, y(n-1)\}\) and may therefore be regarded as a
measure of the new information in the random variable $y(n)$ observed at time $(n)$, hence the name “innovation”.

The fact is that the observation $y(n)$ does not convey completely new information, since the predictable part, $\hat{y}(n/Y(n-1))$, is already completely determined by the past observations $\{y(1), y(2), \ldots, y(n-1)\}$. So, the part of the observation $y(n)$ that is really new is contained in the forward prediction error $f_{n-1}(n)$. The prediction error is referred as the innovation

$$\alpha(n) = f_{n-1}(n), \quad n = 1, 2, \ldots$$

The innovations have some useful properties:

**P1. Orthogonality**: The innovation $\alpha(n)$ associated with the observed random variable $y(n)$ is orthogonal to the past observations $\{y(1), y(2), \ldots, y(n-1)\}$:

$$E[\alpha(n) \cdot y^*(k)] = 0, \quad 1 \leq k \leq n - 1$$

**P2. Whiteness**: The innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$ are orthogonal to each other, as shown by:

$$E[\alpha(n) \cdot \alpha^*(k)] = 0, \quad 1 \leq k \leq n - 1$$

i.e, the innovations process is white.

**P3. Correspondence**: There is one-to-one correspondence between the observed data $\{y(1), y(2), \ldots, y(n)\}$ and the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$, in that the one sequence may be obtained from the other by means of a causal and causally invertible filter without loss of information.

$$\{y(1), y(2), \ldots, y(n)\} \Leftrightarrow \{\alpha(1), \alpha(2), \ldots, \alpha(n)\}$$

Based on this property we may write

$$\hat{x}(n/Y(n)) = \hat{x}(n/A(n))$$

where

$$Y(n) = \{y(1), y(2), \ldots, y(n)\}$$

$$A(n) = \{\alpha(1), \alpha(2), \ldots, \alpha(n)\}$$

The hat notation denotes the minimum mean-square estimate of $x(n)$.

We can write the estimate $\hat{x}(n/Y(n))$ as a linear combination of the innovations $\alpha(1), \alpha(2), \ldots, \alpha(n)$:

$$\hat{x}(n/Y(n)) = \sum_{k=1}^{n} b_k \cdot \alpha(k)$$

where $\{b_k\}$ are to be determined as:
\[ b_k = \frac{E[x(n) \cdot \alpha^*(k)]}{E[\alpha(k) \cdot \alpha^*(k)]}, \quad 1 \leq k \leq n \] (10)

Isolating the term corresponding to \( k=n \), we rewrite (9) in the form:

\[ \hat{x}(n / Y(n)) = \sum_{k=0}^{n-1} b_k \cdot \alpha(k) + b_n \cdot \alpha(n) \] (11)

with

\[ b_n = \frac{E[x(n) \cdot \alpha^*(n)]}{E[\alpha(n) \cdot \alpha^*(n)]} \] (12)

Because, by definition, the summation term on the right-hand side of Eq. 11 equals the previous estimate, \( \hat{x}(n-1 / Y(n-1)) \), we may express the recursive estimation algorithm that are seeking as:

\[ \hat{x}(n / Y(n)) = \hat{x}(n-1 / Y(n-1)) + b_n \cdot \alpha(n) \] (13)

Equations (1), (3) (11) and (13) show that the underlying structure of a recursive mean-square-error estimator is in the form of a predictor-corrector, as depicted in Fig. 1.

The structure consists of two basic steps:
1). The use of observations to compute a forward prediction term. So, the a priori estimations are:

\[ \hat{x}(n / Y(n-1)) \rightarrow \hat{y}(n / Y(n-1)) \]

2). The use of the new measurements to update (i.e., correct) the minimum mean-square estimate of a random variable related linearly to the observations.
The Kalman filter estimates a process by using a form of feedback control: the filter estimates the process state at some time and then obtains feedback in the form of (noisy) measurements. As such, the equations for the Kalman filter fall into two groups: time update equations and measurement update equations.

The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step.

The measurement update equations are responsible for the feedback—i.e. for incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

The time update equations can also be thought of as predictor equations, while the measurement update equations can be thought of as corrector equations. Indeed, the final estimation algorithm resembles that of a predictor-corrector algorithm for solving numerical problems as shown above in Figure 1.

2. Statement of the Kalman filtering problem

Consider a linear, discrete-time dynamical system described by the signal-flow graph shown in Fig.2. The concept of state is fundamental to this formulation. The state vector, or simply state, \( x(n) \), is defined as the minimal set of data that is sufficient to uniquely describe the unforced dynamical behavior of the system. In other words, the state comprises the fewest data on the past behavior of the system that are needed to predict its behavior.

Typically, the state \( x(n) \), assured to be of dimension \( M \) is unknown. To estimate it, we use a set of observed data, denoted by the vector \( y(n) \) in the figure. The observation vector, or simply observation, \( y(n) \), is assumed to be of dimension \( N \).

In mathematical terms, the signal-flow graph of Fig.2 embodies the following pair of equations:

**A). A process equation:**

\[
\alpha(n) = y(n) - \hat{y}(n/Y(n-1))
\]

\[
\hat{x}(n/Y(n)) = \hat{x}(n-1/Y(n-1)) + \alpha(n)
\]
\[ \mathbf{x}(n + 1) = \mathbf{F}(n + 1, n) \cdot \mathbf{x}(n) + \mathbf{v}_1(n) \]  \hfill (14)

The \( M \)-by-1 vector \( \mathbf{v}_1(n) \) represents process noise modeled as a zero-mean, white-noise process whose correlation matrix is defined as:

\[
E[\mathbf{v}_1(n) \cdot \mathbf{v}_1^H(k)] = \begin{cases} \mathbf{Q}_1(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases}
\hfill (15)

The process equation (14) models unknown physical stochastic phenomenon denoted by the state \( \mathbf{x}(n) \) as the output of a linear dynamical system excited by the white noise \( \mathbf{v}_1(n) \).

The linear dynamical system is uniquely characterized by the feedback connection of two units: the transition matrix, denoted by \( \mathbf{F}(n + 1, n) \), and the memory unit denoted by \( z^{-1} \cdot \mathbf{I} \), where \( z^{-1} \) is the unit delay and \( \mathbf{I} \) is the \( M \)-by-\( M \) identity matrix. The transition matrix \( \mathbf{F}(n + 1, n) \) indicates a transition of the system from time \( n \) to time \( n + 1 \) and has the following properties:

1. Product rule:
   \[ \mathbf{F}(n, m) \cdot \mathbf{F}(m, l) = \mathbf{F}(n, l), \quad m, n, l \in \mathbb{Z} \]  \hfill (16)

2. Inverse rule:
   \[ \mathbf{F}^{-1}(n, m) = \mathbf{F}(m, n), \quad m, n \in \mathbb{Z} \]  \hfill (17)

From these two rules, we readily see that:

\[ \mathbf{F}(n, n) = \mathbf{I} \]  \hfill (18)

Also, if the system described in Fig.2 is stationary, then the transition matrix is a constant.

B. A measurement equation, which describes the observation vector as:

\[ \mathbf{y}(n) = \mathbf{C}(n) \cdot \mathbf{x}(n) + \mathbf{v}_2(n) \]  \hfill (19)

where \( \mathbf{C}(n) \) is a known \( N \)-by-\( M \) measurement matrix. The \( N \)-by-1 vector \( \mathbf{v}_2(n) \) is called measurement noise modeled as a zero-mean, white noise process whose correlation matrix is:

\[
E[\mathbf{v}_2(n) \cdot \mathbf{v}_2^H(k)] = \begin{cases} \mathbf{Q}_2(n), & n = k \\ \mathbf{0}, & n \neq k \end{cases}
\hfill (20)

The measurement equation (19) relates the observable output of the system \( \mathbf{y}(n) \) to the state \( \mathbf{x}(n) \). It is assumed that:

- \( \mathbf{x}(0) \), the initial values of the state, is uncorrelated with both \( \mathbf{v}_1(n) \) and \( \mathbf{v}_2(n) \), for \( n \geq 0 \).
- the noise vectors \( \mathbf{v}_1(n) \) and \( \mathbf{v}_2(n) \) are statistically independent, so we may write

\[ E[\mathbf{v}_1(n) \cdot \mathbf{v}_2^H(k)] = \mathbf{0}, \quad \forall n \quad \text{and} \quad k \]  \hfill (21)
The Kalman filtering problem, namely, the problem of jointly solving the process and measurement equations for the unknown state in an optimal manner, may now be formally stated as follow:

Use the entire observed data, consisting of observations \( y(1), y(2), \ldots, y(n) \) to find, for each \( n \geq 1 \), the minimum mean-square-estimate of the state \( x(i) \). The problem is called filtering if \( n = i \), prediction if \( i > n \) and smoothing if \( 1 \leq i < n \).

3. The innovations process

In light of Eqs. (1) and (2), we define the innovations process associated with \( y(n) \) as:

\[
a(n) = y(n) - \hat{y}(n / Y(n-1)), \quad n = 1, 2, \ldots
\]

The \( M \)-by-1 vector \( a(n) \) represents the new information in the observed data \( y(n) \). The innovations process \( a(n) \) has the following properties:

1). The innovations process \( a(n) \), associated with the observed data \( y(n) \) at time \( n \), is orthogonal to all past observations \( \{y(1), y(2), \ldots, y(n-1)\} \), as shown by:

\[
E[a(n) \cdot y^H(k)] = 0, \quad 1 \leq k \leq n - 1
\]

2). The innovations process consists of a sequence of vector random variables that are orthogonal to each other as shown by:

\[
E[a(n) \cdot a^H(k)] = 0, \quad 1 \leq k \leq n - 1
\]

3). There is one-to-one correspondence between the sequence of vector random variables \( \{y(1), y(2), \ldots, y(n)\} \) representing the observed data and the sequence of vector random variables \( \{a(1), a(2), \ldots, a(n)\} \) representing the innovations process, in that the one sequence may be obtained from the other by means of linear stable operators without loss of information. Thus, we may state that:

\[
\{y(1), y(2), \ldots, y(n)\} \iff \{a(1), a(2), \ldots, a(n)\}
\]

4. Correlation matrix of the innovation process

The correlation matrix of the innovations process \( a(n) \) is defined by

\[
R(n) = R_{aa}(n) = E[a(n) \cdot a^H(n)]
\]

The innovation process may express in the form:
\( \mathbf{a}(n) = \mathbf{y}(n) - \mathbf{C}(n) \cdot \hat{\mathbf{x}}(n / Y(n-1)) = \mathbf{C}(n) \cdot \mathbf{x}(n) + \mathbf{v}_2(n) - \mathbf{C}(n) \cdot \hat{\mathbf{x}}(n / Y(n-1)) = \mathbf{C}(n) \cdot \varepsilon(n, n-1) + \mathbf{v}_2(n) \) \tag{27}

where \( \varepsilon(n, n-1) \) is the predicted state-error vector at time \( n \), using data up to time \( (n-1) \). In other words, \( \varepsilon(n, n-1) \) is the difference between the state \( \mathbf{x}(n) \) and the one-step prediction \( \hat{\mathbf{x}}(n / Y(n-1)) \):

\( \varepsilon(n, n-1) = \mathbf{x}(n) - \hat{\mathbf{x}}(n / Y(n-1)) \) \tag{28}

By substituting Eq. (27) into (26) and using the fact that the vectors \( \varepsilon(n, n-1) \) and \( \mathbf{v}_2(n) \) are orthogonal, we obtain the result:

\( \mathbf{R}(n) = \mathbf{C}(n) \cdot \mathbf{K}(n, n-1) \cdot \mathbf{C}^H(n) + \mathbf{Q}_2(n) \) \tag{29}

where \( \mathbf{Q}_2(n) \) is the correlation matrix of the measurement noise vector \( \mathbf{v}_2(n) \). The \( M \)-by-\( M \) matrix \( \mathbf{K}(n, n-1) \) is called the predicted state-error correlation matrix and is defined by:

\( \mathbf{K}(n, n-1) = E[\varepsilon(n, n-1) \cdot \varepsilon^H(n, n-1)] \) \tag{30}

This matrix is used as the statistical description of the error in the predicted estimate \( \hat{\mathbf{x}}(n / Y(n-1)) \).

5. Estimation of the state using the innovation process

Consider the problem of deriving the minimum-mean-square estimate of the state \( \mathbf{x}(i) \) from the innovations process. As was presented in section 1, we may deduce that this estimate may be expressed as a linear combination of the sequence of innovations processes \( \{\mathbf{a}(1), \mathbf{a}(2), ..., \mathbf{a}(n)\} \):

\( \hat{\mathbf{x}}(i / Y(n)) = \sum_{k=1}^{n} \mathbf{B}_i(k) \cdot \mathbf{a}(k) \) \tag{31}

where \( \{\mathbf{B}_i(k), k = 1, 2, ..., n\} \) is a set of \( M \)-by-\( N \) matrices to be determined. The matrix \( \mathbf{B}_i(m) \) is given by:

\( \mathbf{B}_i(m) = E[\mathbf{x}(i) \cdot \mathbf{a}^H(m)] \cdot \mathbf{R}^{-1}(m) \) \tag{32}

Then:

\( \hat{\mathbf{x}}(i / Y(n)) = \sum_{k=1}^{n} E[\mathbf{x}(i) \cdot \mathbf{a}^H(k)] \cdot \mathbf{R}^{-1}(k) \cdot \mathbf{a}(k) = \sum_{k=1}^{n-1} E[\mathbf{x}(i) \cdot \mathbf{a}^H(k)] \cdot \mathbf{R}^{-1}(k) \cdot \mathbf{a}(k) + E[\mathbf{x}(i) \cdot \mathbf{a}^H(n)] \cdot \mathbf{R}^{-1}(n) \cdot \mathbf{a}(n) \) \tag{33}

For \( i = n + 1 \) we may write
\[ \hat{x}(n+1/Y(n)) = \sum_{k=1}^{n-1} E\left[ x(n+1) \cdot a^H(k) \right] \cdot R^{-1}(k) \cdot a(k) + E\left[ x(n+1) \cdot a^H(n) \right] \cdot R^{-1}(n) \cdot a(n) \] (34)

The state \( x(n+1) \) at time \( (n+1) \) is related to the state \( x(n) \) at time \( (n) \) by Eq. (14). Therefore, using this relation, we may write for \( 0 \leq k \leq n \):

\[
E[x(n+1) \cdot a^H(k)] = E\left[ F(n+1,n) \cdot x(n) + v_1(n) \cdot a^H(k) \right] = F(n+1,n) \cdot E[x(n) \cdot a^H(k)]
\]

(35)

where we have made use the fact that \( a(k) \) depends only on the observed data \( \{x(1), y(2), ..., y(k)\} \) and, therefore, \( v_1(n) \) and \( a(k) \) are orthogonal for \( 0 \leq k \leq n \). We may write the summation term on the right-hand side of Eq. (34) as:

\[
\sum_{k=1}^{n-1} E[x(n+1) \cdot a^H(k)] \cdot R^{-1}(k) \cdot a(k) = F(n+1,n) \cdot \hat{x}(n/Y(n-1))
\]

(36)

**Definition:** (Kalman gain): The \( M \)-by-\( N \) matrix

\[
G(n) = E\left[ x(n+1) \cdot a^H(n) \right] \cdot R^{-1}(n)
\]

(37)

where \( E[x(n+1) \cdot a^H(n)] \) is the cross-correlation matrix between the state vector \( x(n+1) \) and the innovation process \( a(n) \).

Using this definition and the result of Eq.(34), we may write:

\[
\hat{x}(n+1/Y(n)) = F(n+1,n) \cdot \hat{x}(n/Y(n-1)) + G(n) \cdot a(n)
\]

(38)

The last equation is fundamental and it shows that we may compute the minimum mean-square estimate \( \hat{x}(n+1/Y(n)) \) of the state of a linear dynamical system by adding to the previous estimate \( \hat{x}(n/Y(n-1)) \), pre-multiplied by the transition matrix \( F(n+1,n) \), a correction term which equals the innovations process \( a(n) \) pre-multiplied by the matrix \( G(n) \), the Kalman gain matrix.

The Kalman gain matrix can be reformulated as:

\[
G(n) = F(n+1,n) \cdot K(n,n-1) \cdot C^H(n) \cdot R^{-1}(n)
\]

(39)

**Riccati Equation.** Computation of the Kalman gain by (39) is difficult because it requires that the predicted state-error correlation matrix \( K(n,n-1) \) be known. A more efficient way will be a recursive formula. The following Eq. could be derived:

\[
K(n+1,n) = F(n+1,n) \cdot K(n,n-1) \cdot F^H(n+1,n) + Q_1(n)
\]

(40)
and is called Riccati difference equation for the recursive computation of the predicted state-error correlation matrix. The new matrix, $M$-by-$M$ matrix $K(n)$ is described by the recursion

$$K(n) = K(n, n-1) - F(n, n+1) \cdot G(n) \cdot C(n) \cdot K(n, n-1)$$  

(41)

The Eqs. (38), (39), (40) define the Kalman’s one-step prediction algorithm.

Figure 3 represents a signal flow of Eqs. (40) and (41) and represent the Riccati equation solver in that, given $K(n, n-1)$, it computes the updated value $K(n+1, n)$.

Figure 3 – One-step state predictor: given the old estimate $\hat{x}(n / Y(n-1))$ and the observation $y(n)$, the predictor computes the new state $\hat{x}(n+1 / Y(n))$

**Discrete Kalman filter time update equations (Prediction)**

<table>
<thead>
<tr>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial estimates:</td>
<td>$\hat{x}(n)$ and $K(n, n-1)$</td>
</tr>
<tr>
<td>Estimate the state (a priori) (prediction)</td>
<td>$\hat{x}(n+1/Y(n)) = F(n+1,n) \cdot \hat{x}(n/y(n-1)) + G(n) \cdot a(n)$</td>
</tr>
<tr>
<td>Estimate the covariance of the error (prediction)</td>
<td>$K(n+1,n) = F(n+1,n) \cdot K(n,n-1) \cdot F^H(n+1,n) + Q_1(n)$</td>
</tr>
</tbody>
</table>

The first task during the measurement update is to compute the Kalman gain, $G$. The next step is to actually measure the process to obtain $y(n)$ and then to generate the a posteriori state estimate by incorporating the measurement. The final step is to obtain an a posteriori error covariance $K(n+1,n)$.

**Discrete Kalman filter measurement update equations**

<table>
<thead>
<tr>
<th>Description</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute the Kalman gain</td>
<td>$G(n) = F(n+1,n) \cdot K(n,n-1) \cdot C^H(n) \cdot R^{-1}(n)$</td>
</tr>
<tr>
<td>Measure the process</td>
<td>$y(n)$</td>
</tr>
</tbody>
</table>
Generate the *a posteriori* state estimate
\[
\alpha(n) = y(n) - \hat{y}(n/Y(n-1))
\]
\[
\hat{y}(n/Y(n-1)) = C(n) \cdot \hat{x}(n/Y(n-1))
\]

Compute *a posteriori* error covariance
\[
K(n+1,n) = K(n,n-1) - F(n+1,n) \cdot G(n) \cdot C(n) \cdot K(n,n-1)
\]

**Filter parameters and tuning**

The measurement noise correlation matrix \( Q_2 \) is usually measured *a priori* to operation of the filter. Measuring the measurement error covariance \( Q_2 \) is generally practical (possible) because we need to be able to measure the process anyway (while operating the filter) so we would generally be able to take some off-line sample measurements in order to determine the variance of the measurement noise.

The determination of the process noise correlation matrix \( Q_1 \) is generally more difficult as we typically do not have the ability to directly observe the process we are estimating. Sometimes a relatively simple (poor) process model can produce acceptable results if one “injects” enough uncertainty into the process via the selection of \( Q_1 \). Certainly in this case one would hope that the process measurements are available.

In either case, whether or not we have a rational basis for choosing the parameters, often times superior filter performance (statistically speaking) can be obtained by tuning the filter parameters \( Q_1 \) and \( Q_2 \). The tuning is usually performed off-line.

We note that under conditions where \( Q_1 \) and \( Q_2 \) are in fact constants, both estimation error covariance \( K \) and the Kalman gain \( G \) will stabilize quickly and then remain constant. If this is the case, these parameters can be pre-computed by either running the filter off-line, or for example by determining the steady-state value of \( K \).

### 6. Filtering

We wish to compute the filtered estimate \( \hat{x}(n/Y(n)) \) by using the *one-step prediction algorithm* described previously. Starting from the state equation,

\[
x(n+1) = F(n+1,n) \cdot x(n) + v_1(n)
\]

we may write that the minimum mean-square estimate of the state \( \hat{x}(n+1) \) at time \( (n+1) \) given data up to time \( n \), equals:

\[
\hat{x}(n+1/Y(n)) = F(n+1,n) \cdot \hat{x}(n/Y(n)) + \hat{v}_1(n/Y(n))
\]

Because the noise \( v_1(n) \) is independent of the observed data \( y(1),...,y(n) \), it follows that the corresponding minimum mean-square estimate \( v_1(n/Y(n)) \) is zero. Then (42) simplifies to

\[
\hat{x}(n+1/Y(n)) = F(n+1,n) \cdot \hat{x}(n/Y(n))
\]

To find the filtered estimate \( x(n/Y(n)) \) we pre-multiply both sides of Eq. (43) by the transition matrix \( F(n,n+1) \). By doing so, we obtain:

\[
F(n,n+1) \cdot \hat{x}(n+1/Y(n)) = F(n,n+1) \cdot F(n+1,n) \cdot \hat{x}(n/Y(n))
\]
Based on property
\[ F(n+1,n) \cdot F(n,n+1) = I \] (46)
we deduce
\[ F(n,n+1) = F^{-1}(n+1,n) \] (47)
and
\[ F(n,n+1) \cdot F(n+1,n) = F^{-1}(n+1,n) \cdot F(n+1,n) = I \] (48)

Then, from Eq. (44), we obtain
\[ \hat{x}(n/Y(n)) = F(n,n+1) \cdot \hat{x}(n+1/Y(n)) = F^{-1}(n+1,n) \cdot \hat{x}(n+1/Y(n)) \] (49)

This equation shows that knowing the solution of the one-step prediction problem (i.e. the minimum mean-square estimate \( \hat{x}(n+1/Y(n)) \)) we may determine the corresponding filtered estimate \( \hat{x}(n/Y(n)) \) simply by multiplying \( \hat{x}(n+1/Y(n)) \) by the transition matrix \( F(n,n+1) \).

7. Filtered Estimation Error

In a filtering framework, it is natural to define the filtered estimation error vector in terms of the filtered estimate of the state as:
\[ e(n) = y(n) - C(n) \cdot \hat{x}(n/Y(n)) \] (50)

This definition is similar to that of Eq. (31) for the innovations vector \( a(n) \), except that we have substituted the filtered estimate \( \hat{x}(n/Y(n)) \) for the predicted estimate \( \hat{x}(n/Y(n-1)) \). By making some computations, we have:
\[ e(n) = y(n) - C(n) \cdot \hat{x}(n/Y(n-1)) - C(n) \cdot F(n,n+1) \cdot G(n) \cdot a(n) = a(n) - C(n) \cdot F(n,n+1) \cdot G(n) \cdot a(n) = [I - C(n) \cdot F(n,n+1) \cdot G(n)] \cdot a(n) \] (51)

The matrix-valued quantity inside the square brackets is called the conversion factor, which provides a formula for converting the innovations vector \( a(n) \) into the filtered estimation error vector \( e(n) \).

An equivalent expression of filtered estimation error vector is:
\[ e(n) = Q_2(n) \cdot R^{-1}(n) \cdot a(n) \] (52)

where \( Q_2(n) \) is the correlation matrix of the measurement noise process, \( v_2(n) \), and the matrix \( R(n) \) is the correlation matrix of the innovations process \( a(n) \).

Excepting a pre-multiplication by \( Q_2(n) \), the last equation shows that the inverse matrix \( R^{-1}(n) \) plays the role of a conversion factor in the Kalman filter theory.
Note: The $M$-by-$M$ matrix $K(n)$ in the formulation of the Riccati difference equation (40) equals the correlation matrix of the error inherent in the filtered estimate $\hat{x}(n|Y(n))$:

$$K(n) = E[e(n)\cdot e^H(n)].$$

7. Initial conditions

The initial state of the process equation is not known precisely. Relatively, it is usually described by its mean and correlation matrix. In the absence of any observed data at time $n=0$, we may choose the initial predicted estimate as

$$\hat{x}(1|Y(0)) = E[x(1)]$$

and its correlation matrix:

$$K(1,0) = E[(x(1) - E[x(1)])(x(1) - E[x(1)])^H] = \Pi_0$$

The choice for the initial condition has the advantage of a filtered estimate of the state $\hat{x}(n|Y(n))$ that is unbiased. Assuming that the state vector $x(n)$ has zero mean, we may simplify:

$$\hat{x}(1|Y(0)) = 0$$

and

$$K(1,0) = E[x(1)\cdot x^H(1)] = \Pi_0$$

8. Kalman filter as the unifying basis for RLS filters

In order to represent a RLS filter as a Kalman filter, we need to formulate the state – space description for the underlying dynamics of RLS filters.

The RLS filter is a linear estimator of the multiple regression model, as:

$$d(n) = w_0^H \cdot u(n) + e_0(n)$$

where $w_0$ is the unknown parameter vector of the model and $e_0(n)$ is the measurement error modeled as white noise.

Using the Kalman filter notations we may postulate the following pair of equations for the state-space model of RLS filters with $\lambda = 1$:

$$x(n+1) = x(n)$$

$$y(n) = C(n) \cdot x(n) + v(n)$$

A logical choice for $x(n)$ is the parameter vector $w_0$. Then, taking the complex conjugate of both sides of (57) and comparing with the measurement equation of Kalman, we deduce for $\lambda = 1$:
\[ x(n) = w_0 \]
\[ y(n) = d^*(n) \]
\[ C(n) = u^H(n) \]
\[ v(n) = e^*_0(n) \]

We may conclude that the basic relationship between the Kalman filter and the RLS filter is based on the unforced state-space model:

\[ x(n+1) = \lambda^{-\frac{1}{2}} \cdot x(n), \quad 0 < \lambda \leq 1 \]  
\[ y(n) = u^H(n) \cdot x(n) + v(n) \]  

Table 1 shows a summary of the correspondence between Kalman variables and RLS variables.

<table>
<thead>
<tr>
<th>Kalman description</th>
<th>(Unforced dynamical model)</th>
<th>RLS</th>
<th>Multiple regression model description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial value of state</td>
<td>x(0)</td>
<td>w_0</td>
<td>Parameter vector</td>
</tr>
<tr>
<td>State</td>
<td>x(n)</td>
<td>( \lambda^{-\frac{n}{2}} \cdot w_0 )</td>
<td>Exponentially weighted parameter vector</td>
</tr>
<tr>
<td>Reference (observation) signal</td>
<td>y(n)</td>
<td>( \lambda^{-\frac{n}{2}} \cdot d^*(n) )</td>
<td>Desired response</td>
</tr>
<tr>
<td>Measurement noise</td>
<td>v(n)</td>
<td>( \lambda^{-\frac{n}{2}} \cdot e_0^*(n) )</td>
<td>Measurement error</td>
</tr>
<tr>
<td>One-step prediction of state vector</td>
<td>( \hat{x}(n+1/Y(n)) )</td>
<td>( \lambda^{-\frac{n+1}{2}} \cdot \hat{w}(n) )</td>
<td>Estimation of the parameter vector</td>
</tr>
<tr>
<td>Correlation matrix of error in state prediction</td>
<td>K(n)</td>
<td>( \lambda^{-1} \cdot P(n) )</td>
<td>Inverse of the correlation matrix of input vector</td>
</tr>
<tr>
<td>Kalman gain</td>
<td>g(n)</td>
<td>( \lambda^{-\frac{1}{2}} \cdot k(n) )</td>
<td>Gain vector</td>
</tr>
<tr>
<td>Innovation</td>
<td>a(n)</td>
<td>( \lambda^{-\frac{n}{2}} \cdot \xi^*(n) )</td>
<td>A priori estimation error</td>
</tr>
<tr>
<td>Conversion factor</td>
<td>( r^{-1}(n) )</td>
<td>( \gamma(n) )</td>
<td>Conversion factor</td>
</tr>
<tr>
<td>Initial conditions</td>
<td>( \hat{x}(1/Y(0)) = 0 )</td>
<td>( \hat{w}(0) = 0 )</td>
<td>Initial conditions</td>
</tr>
</tbody>
</table>
Summary

Table 2 below presents a summary of the variables and parameters used to formulate the solution to the Kalman filtering problem. The input of the filter is the vector process \( y(n) \), represented by the vector space \( Y(n) \) and the output is the filtered estimate \( \hat{x}(n/Y(n)) \) of the state vector.

We have used \( \hat{x}(n/Y(n-1)) \) and \( \hat{x}(n/Y(n)) \) to distinguish between the \textit{a priori} (before including the observation \( y(n) \)) and \textit{a posteriori} (after including the observation \( y(n) \)) estimates of the state \( x(n) \), respectively. In the literature of the Kalman filters, the alternative notations \( x_\textit{a priori} \) and \( x_\textit{a posteriori} \) are often used for the \textit{a priori} and \textit{a posteriori} estimates of the state, respectively. By the same token, \( K_\textit{a priori} \) and \( K_\textit{a posteriori} \) are often used in place of \( K(n,n-1) \) and \( K(n) \), respectively.

Table 3 presents a summary of the Kalman filter (including initial conditions) based on the one-step prediction algorithm.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>The number of outputs</td>
<td></td>
</tr>
<tr>
<td>( M )</td>
<td>The number of states</td>
<td></td>
</tr>
<tr>
<td>( x(n) )</td>
<td>State at time ( n )</td>
<td>( M ) by 1</td>
</tr>
<tr>
<td>( y(n) )</td>
<td>Observation at time ( n )</td>
<td>( N ) by 1</td>
</tr>
<tr>
<td>( F(n+1,n) )</td>
<td>Transition matrix from time ( n ) to time ( n+1 )</td>
<td>( M ) by ( M )</td>
</tr>
<tr>
<td>( C(n) )</td>
<td>Measurement matrix at time ( n )</td>
<td>( N ) by ( M )</td>
</tr>
<tr>
<td>( Q_1(n) )</td>
<td>Correlation matrix of process noise ( v_1(n) )</td>
<td>( M ) by ( M )</td>
</tr>
<tr>
<td>( Q_2(n) )</td>
<td>Correlation matrix of measurement noise ( v_2(n) )</td>
<td>( N ) by ( N )</td>
</tr>
<tr>
<td>( \hat{x}(n/Y(n-1)) )</td>
<td>\textit{Predicted (a priori) estimate} of the state at time ( n ) given the observations ( y(1), y(2), \ldots, y(n-1) )</td>
<td>( M ) by 1</td>
</tr>
<tr>
<td>( \hat{x}(n/Y(n)) )</td>
<td>\textit{Predicted (a posteriori) estimate} of the state at time ( n ) given the observations ( y(1), y(2), \ldots, y(n) )</td>
<td>( M ) by 1</td>
</tr>
<tr>
<td>( G(n) )</td>
<td>Kalman gain at time ( n )</td>
<td>( M ) by ( N )</td>
</tr>
<tr>
<td>( a(n) )</td>
<td>Innovations vector at time ( n )</td>
<td>( N ) by 1</td>
</tr>
<tr>
<td>( R(n) )</td>
<td>Correlation matrix of the innovations vector ( a(n) )</td>
<td>( N ) by ( N )</td>
</tr>
<tr>
<td>( K(n,n-1) )</td>
<td>\textit{(A priori) (Predicted) correlation matrix} of the state error in ( \hat{x}(n/Y(n-1)) )</td>
<td>( M ) by ( M )</td>
</tr>
<tr>
<td>( K(n,n) )</td>
<td>\textit{(A posteriori) correlation matrix} of the state error in ( \hat{x}(n/Y(n)) )</td>
<td>( M ) by ( M )</td>
</tr>
</tbody>
</table>
Table 3 – Summary of the Kalman Filter based on One-Step Prediction

**Input vector process**
- Observations = \{y(1), y(2), ..., y(N)\}

**Known parameters:**
- Transition matrix = \(F(n+1, n)\)
- Measurement matrix = \(C(n)\)
- Correlation matrix of process noise = \(Q_1(n)\)
- Correlation matrix of measurement noise = \(Q_2(n)\)

**Initial Conditions:**
- \(n = 0;\)
- \(\hat{x}(1/Y(0)) = E[x(1)]\)
- \(K(1,0) = E[(x(1) - E[x(1)])(x(1) - E[x(1)])^H] = \Pi_0\)

REPEAT:
- \(n = n + 1;\)

// Correction (A posteriori estimation)
- \(R(n) = C(n) \cdot K(n, n-1) \cdot C^H(n) + Q_2(n)\)  
  **Innovation Correlation**
- \(G(n) = F(n+1, n) \cdot K(n, n-1) \cdot C^H(n) \cdot R^{-1}(n)\)  
  **Kalman gain**
- \(\hat{y}(n/Y(n-1)) = C(n) \cdot \hat{x}(n/Y(n-1))\)
- \(\alpha(n) = y(n) - \hat{y}(n/Y(n-1))\)  
  **Innovation**
- \(K(n, n) = K(n, n-1) - F(n, n+1) \cdot G(n) \cdot C(n) \cdot K(n, n-1)\)  
  **Filtered state-error correlation**

// Prediction (A priori estimation)
- \(\dot{x}(n+1/Y(n)) = F(n+1, n) \cdot \dot{x}(n/Y(n-1)) + G(n) \cdot \alpha(n)\)
- \(K(n+1, n) = F(n+1, n) \cdot K(n, n) \cdot F^H(n+1, n) + Q_1(n)\)  
  **Predicted state-error correlation**

// Filtering
- \(\hat{x}(n/Y(n)) = F(n, n+1) \cdot \dot{x}(n+1/Y(n)) = F^{-1}(n+1, n) \cdot \hat{x}(n+1/Y(n))\)  
  **Filtered state estimate**

END.

The Kalman filter is a linear, discrete-time, finite – dimensional system endowed with a recursive structure that makes a digital computer well suited for its implementation. A key property of the Kalman filter is that it is the minimum mean-square (variance) estimator of the state of a dynamical system.

In the context of the family of linear adaptive filtering algorithms rooted in deterministic least-squares estimation, Kalman filter theory is of profound theoretical as well as practical importance for two reasons:

1). The underlying dynamics of recursive least-squares RLS filters are described by an unforced linear dynamical model, i.e. a Kalman’s model.
2). The Kalman filter is more general and can be used to develop variants of the RLS filters, namely, square-root RLS filters and order-recursive RLS filters.

Many problems in signal processing and control theory are mathematically equivalent. The Kalman filter theory is rooted in the control literature, but we established the link between Kalman filters and linear adaptive filters, which is one heart of the signal-processing field.

References

Some tutorials, references, and research related to the Kalman filter. The site is maintained by Greg Welch and Gary Bishop, faculty members of the Department of Computer Science at the University of North Carolina at Chapel Hill. [http://www.cs.unc.edu/~welch/kalman/](http://www.cs.unc.edu/~welch/kalman/)
Tracking of time-varying systems

Content

Modeling of the non-stationary environment
Degree of nonstationarity
Criteria for tracking assessment

Summary

Two families of adaptive filtering algorithms were considered: the LMS family and the RLS family. We considered the average behavior of the standard LMS and RLS algorithm operating in a stationary environment. In such a case, the error performance surface is fixed and the essential requirement is to seek the minimum point of that surface and thereby assure optimum or near optimum performance.

Tracking is a steady-state phenomenon, to be considered with convergence, which is a transient phenomenon. Moreover, we may state that, in general, the rate of the convergence and the tracking capability are two different properties of the algorithm.

1. Modeling of the non-stationary environment

An environment may become nonstationary in one of two basic ways:

a). The desired response has a statistics which is time varying. An example is system identification problem when a FIR filter is used to model a time-varying system.

b). The stochastic process supplying the tap inputs of the adaptive filter is non-stationary. As example: a FIR filter used to equalize a time varying channel.

A popular time-varying model for a non-stationary environment is considered now. It is governed by two basic processes:

1. **A first order Markov process.** The unknown dynamic equations of the environment are modeled by a transversal filter whose tap-weight vector \( \mathbf{w}_0(n) \) undergoes a first-order Markov process, written in vector form as:

\[
\mathbf{w}_0(n+1) = a \cdot \mathbf{w}_0(n) + \mathbf{\omega}(n)
\]

where \( a \) is a fixed parameter of the model and \( \mathbf{\omega}(n) \) is the process noise vector assumed to be of zero mean and correlation matrix \( \mathbf{R}_{\omega\omega} \). In physical terms, the tap-weight vector \( \mathbf{w}_0(n) \) may be viewed as originating from the process noise \( \mathbf{\omega}(n) \), whose individual elements are applied to a bank of one-pole low-pass filter. Each such filter has a transfer function equal to \( \frac{1}{1-a \cdot z^{-1}} \). It is assumed that the value of the parameter \( a \) is very close to unity. The effect is that the bandwidth of the low pass filter is very smaller than the incoming data rate.

2. **Multiple regression** The observable feature of the environment is the desired response, \( d(n) \), is governed by the linear regression model:
where \( v(n) \) is the measurement noise, assumed to be white noise zero mean and variance \( \sigma_v^2 \).
Thus, even, through both input vector \( u(n) \) and the noise \( v(n) \) are stationary, the model output \( d(n) \) is nonstationary random process by virtue of the fact that \( w_0(n) \) varies in time.

\[
d(n) = w^H(n) \cdot u(n) + v(n)
\]  

(2)

The system identification problem is depicted in Fig. 2. The problem of system identification described in terms of adaptive filtering could be presented as:

*Given the observable \( d(n) \) as the desired response for an input vector \( u(n) \), design an adaptive filter that tracks statistical variations in the Markov's impulse response vector \( w_0(n) \).*

Figure 1 – A block diagram of Eqs. (1) and (2)

Figure 2 – System identification using an adaptive filter.
Both \( w_0(n) \) and \( \hat{w}(n) \) are assumed to be of length \( M \)
The error signal involved in the adaptive process is defined by:

\[ e(n) = d(n) - y(n) = w_H^0(n) \cdot u(n) + v(n) - w_H^0(n) \cdot u(n) \]  \hspace{2cm} (3)

where \( \hat{w}(n) \) is the tap-weight of the adaptive filter, assumed to have both a transversal structure and the same number of taps. The unknown system is represented by \( w_o(n) \). The tap weight vector \( w_o(n) \) represents the target to be tracked by the filter. Whenever \( \hat{w}(n) \) equals \( w_o(n) \), the minimum mean-square error produced by the adaptive filter equals the irreducible error variance \( \sigma_v^2 \).

**Notes:**
1). The time variations of the process noise vector \( \omega(n) \) are slow (i.e. bounded), which makes it possible for the adaptive transversal filter to track the statistical variations in the dynamic behavior of the environment.

2). For a given linear model of a time-varying model, the Kalman filter is the optimum-tracking algorithm assuming Gaussian statistics.

2. Degree of nonstationarity

In order to provide a clear definition of the rather ambiguous concepts of “slow” and “fast” statistical variations of the model, some variable to measure the degree of the nonstationarity are introduced.

In the context of the Markov model, the degree of nonstationarity, denoted by \( \alpha \), is formally defined as the square root of the ratio of two quantities: the expectation of the squared magnitude of the inner product of the process noise \( \omega(n) \) and the vector \( u(n) \), and the average power of the measurement noise, \( v(n) \). That is:

\[
\alpha = \left( \frac{E \left[ \omega^H(n) \cdot u(n) \right]^2}{E \left[ v(n)^2 \right]} \right)^{1/2}
\]  \hspace{2cm} (4)

The degree of nonstationarity is a characteristic of time-varying system alone and has nothing to do with the adaptive filter.

If:

(i) the process noise \( \omega(n) \) is white with zero mean and correlation matrix \( R_{\omega\omega}(n) \);

(ii) the measurement noise \( v(n) \) is white with zero mean and variance \( \sigma_v^2 \);

(iii) the process noise \( \omega(n) \) and measurement noise \( v(n) \) are independent of each other;
the measurement matrix, i.e. $\mathbf{u}^H(n)$, is independent of both the measurement noise $v(n)$ and the process noise $\phi(n)$; then

$$
\alpha = \frac{1}{\sigma_v} \sqrt{\text{tr}(\mathbf{R}_{uu} \cdot \mathbf{R}_{\phi\phi})}
$$

where $\text{tr}[]$ denotes the trace\(^{22}\) of the matrix enclosed inside the square brackets.

The second variable used in the description of nonstationarity is called the \textit{misadjustment}:

$$
M = J_{\text{ex}}(\infty) \geq \frac{\text{tr}(\mathbf{R}_{uu} \cdot \mathbf{R}_{\phi\phi})}{\sigma_v^2} = \alpha^2
$$

where $J_{\text{min}}$ is the minimum mean square error which the adaptive filter may equal the variance of the measurement noise, $\sigma_v^2$.

3. \textbf{Criteria for tracking assessment}

If $\mathbf{w}_0(n)$ represent the parameters of the unknown dynamical system and $\hat{\mathbf{w}}(n)$ represent the tap-weight vector of the adaptive transversal filter, we define the weight error vector as:

$$
\varepsilon(n) = \mathbf{w}_0(n) - \hat{\mathbf{w}}(n)
$$

The first criterion is \textit{mean square deviation} (MSD) between the actual weight vector $\mathbf{w}_0(n)$ of the unknown dynamical system and the adjusted weight vector $\hat{\mathbf{w}}(n)$ of the adaptive filter:

$$
D(n) = E[\|\mathbf{w}_0(n) - \hat{\mathbf{w}}(n)\|^2] = E[\|\varepsilon(n)\|^2]
$$

where $n$ is the number of iterations, assumed to be large enough for the adaptive filter’s transient mode of operation to have finished.

Because the weight-error vector may be expressed as:

$$
\varepsilon(n) = \varepsilon_1(n) + \varepsilon_2(n) = (E[\hat{\mathbf{w}}(n)] - \mathbf{w}(n)) + (\mathbf{w}_0(n) - E[\hat{\mathbf{w}}(n)])
$$

where $\varepsilon_1(n)$ is the weight vector noise and $\varepsilon_2(n)$ is the weight vector lag, we may express the mean-square deviations as

---

\(^{22}\) Trace($\mathbf{A}$) is the sum of the diagonal elements of $\mathbf{A}$, which is also the sum of the eigenvalues of $\mathbf{A}$. 

132
\[ D(n) = D_1(n) + D_2(n) = E\left[\|e_1(n)\|^2\right] + E\left[\|e_2(n)\|^2\right] \] (10)

where the first term is the estimation variance and the second one is the lag variance. The presence of the \( D_2(n) \) is testimony to the nonstationary nature of the environment.

The second criterion is defined by

\[ M(n) = J_{\text{ex}}(n) / \sigma_v^2 \] (11)

where \( J_{\text{ex}}(n) \) is the excess (residual) mean-square error of the adaptive filter, measured with respect to the variance \( \sigma_v^2 \) of the white noise component \( v(n) \) at the output of the model. For good tracking performance, the misadjustment \( M(n) \) should be small compared with unity.

**Summary**

The problem of tracking was presented, i.e. the problem of adaptive filtering in nonstationary environment.

Tracking is problem specific. Thus, the tracking performance of an adaptive filter used in system identification is quite different from that of an adaptive filter used in channel equalization.

To assess the tracking capability of an adaptive filter, we may use the mean-square deviation \( D(n) \) or the misadjustment, \( M(n) \). In each case, we may identify two contributions, one representative of the stationary case and the other attributed to nonstationarity of the environment.

The LMS algorithm exhibits a more robust tracking behavior than the RLS algorithm does. Therefore, it should not be surprising to find that, in data transmission over time-varying communication channels, the LMS algorithm is preferred over the RLS algorithm, not only of its simplicity but also because of its better tracking properties.
### COURSE 3 – BASIC FORMULAS

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Model</th>
<th>Time description</th>
<th>Transfer Function (Real case only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>AR</td>
<td>$u(n) + a_1^*u(n-1) + \ldots + a_M^*u(n-M) = v(n)$</td>
<td>$H(z) = \frac{U(z)}{V(z)} = \frac{1}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}}$</td>
</tr>
<tr>
<td></td>
<td>AR</td>
<td>$\sum_{i=0}^{M} a_n^* \cdot u(n-i) = v(n), \quad a_0 = 1.$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AR</td>
<td>$u(n) = -\sum_{i=1}^{M} a_n^* \cdot u(n-i) + v(n)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Property</td>
<td>$r_{uu}(l) = w_1^* \cdot r_{uu}(l-1) + w_2^* \cdot r_{uu}(l-2) + \ldots + w_M^* \cdot r_{uu}(l-M), \quad l &gt; 0$</td>
<td>$\sigma_v^2 = \sum_{i=0}^{M} a_i^* \cdot r_{uu}(i)$</td>
</tr>
<tr>
<td>2.</td>
<td>MA</td>
<td>$u(n) = b_0^* \cdot v(n) + b_1^* \cdot v(n-1) + b_2^* \cdot v(n-2) + \ldots + b_K^* \cdot v(n-K)$</td>
<td>$H(z) = \frac{U(z)}{V(z)} = b_0 + b_1 z^{-1} + \ldots + b_K z^{-K}$</td>
</tr>
<tr>
<td></td>
<td>MA</td>
<td>$u(n) = v(n) + \sum_{i=1}^{K} b_i^* \cdot v(n-i)$</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>ARMA</td>
<td>$u(n) + \sum_{i=1}^{M} a_i^* \cdot u(n-i) = v(n) + \sum_{j=1}^{K} b_i^* \cdot v(n-j)$</td>
<td>$H(z) = \frac{U(z)}{V(z)} = \frac{1+b_1 z^{-1} + \ldots + b_K z^{-K}}{1 + a_1 z^{-1} + \ldots + a_M z^{-M}}$</td>
</tr>
<tr>
<td>4.</td>
<td>Yule-Walker</td>
<td>$\begin{bmatrix} r(0) &amp; r^<em>(-1) &amp; \ldots &amp; r^</em>(-(M-1)) \ r^<em>(1) &amp; r(0) &amp; \ldots &amp; r^</em>(-(M-2)) \ \vdots &amp; \vdots &amp; \ddots &amp; \vdots \ r^<em>(M-1) &amp; r^</em>(M-2) &amp; \ldots &amp; r(0) \end{bmatrix} \begin{bmatrix} -a_1^* \ -a_2^* \ \vdots \ -a_M^* \end{bmatrix} = \begin{bmatrix} r^<em>(1) \ r^</em>(2) \ \vdots \ r^*(M) \end{bmatrix}$</td>
<td>$w = -a = R_{uu}^{-1} \cdot r_{uu}$</td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>$S_0(\omega) = \left</td>
<td>H(e^{i\omega})\right</td>
</tr>
</tbody>
</table>
## Basic relation of course 5

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Name of the algorithm</th>
<th>Basic relations</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Steepest Descent (SD)</td>
<td>( \mathbf{w}(n+1) = \mathbf{w}(n) + \mu \cdot [\mathbf{r}<em>{ud} - \mathbf{R}</em>{uu} \cdot \mathbf{w}(n)] )</td>
<td>( 0 &lt; \mu &lt; \frac{2}{\lambda_{\text{max}}} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( e(n) = d(n) - \hat{d}(n) = d(n) - \mathbf{w}^H(n) \cdot \mathbf{u}(n) )</td>
<td>( \lambda_{\text{max}} ) is the largest eigenvalue of the autocorrelation matrix ( \mathbf{R}_{uu} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \mathbf{w}(n) = [w_0(n) \quad w_1(n) \quad \ldots \quad w_{M-1}(n)]^T )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \mathbf{u}(n) = [u(n) \quad u(n-1) \quad \ldots \quad u(n-M+1)]^T )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( J(\mathbf{w}(n)) = \sigma_d^2 - \mathbf{w}^H(n) \cdot \mathbf{r}<em>{ud} - \mathbf{r}</em>{ud}^H \cdot \mathbf{w}(n) + \mathbf{w}^H(n) \cdot \mathbf{R}_{uu} \cdot \mathbf{w}(n) )</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Least-Mean-Square (LMS)</td>
<td>( \hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) \cdot [\mathbf{d}^\ast(n) - \mathbf{H}^H(n) \cdot \hat{\mathbf{w}}(n)] = \hat{\mathbf{w}}(n) + \mu \cdot \mathbf{u}(n) \cdot e^\ast(n) )</td>
<td>( \mu \in \left( 0; \frac{2}{M \cdot S_{\text{max}}} \right) )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( S_{\text{max}} ) is the maximum value of the power spectral density of the tap input ( u(n) ).</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( M )- the filter length, as a moderate to large value</td>
<td>( \hat{\mu} ) is a real positive scaling factor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \mu )</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Normalized Least-Mean-Square (nLMS)</td>
<td>( \hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\hat{\mu}}{|\mathbf{u}(n)|^2} \cdot \mathbf{u}(n) \cdot e^\ast(n) )</td>
<td>( 0 &lt; \delta &lt; 2 \frac{D(n) \cdot E[u^2(n)]}{E[e^2(n)]} )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( D(n) = E[|\mathbf{e}(n)|^2] = E[|\mathbf{w} - \hat{\mathbf{w}}(n)|^2] )</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Convex optimization for recursive mean square estimations</td>
<td>( x^2(n+1) = (1 - \gamma) \cdot x^2(n+1) + \gamma \cdot x^2(n) )</td>
<td>( \gamma \in [0.9; 0.99] ) a smoothing constant</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( x(n) = e(n) \rightarrow E[e^2(n)] = e^2(n) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>( x(n) = u(n) \rightarrow E[u^2(n)] = u^2(n) )</td>
<td></td>
</tr>
</tbody>
</table>