Statistical Signal Processing
Theory

Book : Adaptive Filter, by Simon Haykin

For Random Process - Papoulis
Peebles
Yates

- Probability Theory based Axioms -
  sets, Fields and Events
- Sample Description space - Set of all outcomes
- Random Experiment → H
- Subsets of Ω are Events
  $\Omega \in \{ \xi_i \}$

$\Omega \rightarrow$ Countably Finite or Infinite outcomes
$\hookrightarrow$ Uncountably infinite

If, $\Omega = \{ \xi_1, \dots \xi_N \}$ then the number of subsets is 2^N

- If Ω has a countable number of elements then any subset of Ω maybe assigned a probability according to axioms.
- Then the class of all subsets make up a σ-field and each subset is an event.
- σ-field is denoted by F
- We talk of σ-field, F, of events

- When $\Omega = \mathbb{R}$ = the real line, then not every subset of Ω can be assigned a probability consistent with the axioms.

- Only those subsets to which a probability can be assigned consistent with the axioms will be called events.

- This smaller collection forms a σ-field, F

- On the real line, $\mathbb{R}$, the σ-field is sometimes called the BOREL field.

$P \rightarrow$ Probability Space
$\rightarrow (\Omega, F, P)$

Axiomatic Definition

Probability is a set function $P[\cdot]$ that assigns to every event $E \in F$ a number $P[E]$ called the probability of E s.t.

i) $P[E] \geq 0$
ii) $P[\Omega] = 1$
iii) $P[E \cup F] = P[E] + P[F]$ if $E \cdot F = \phi$

Random Process

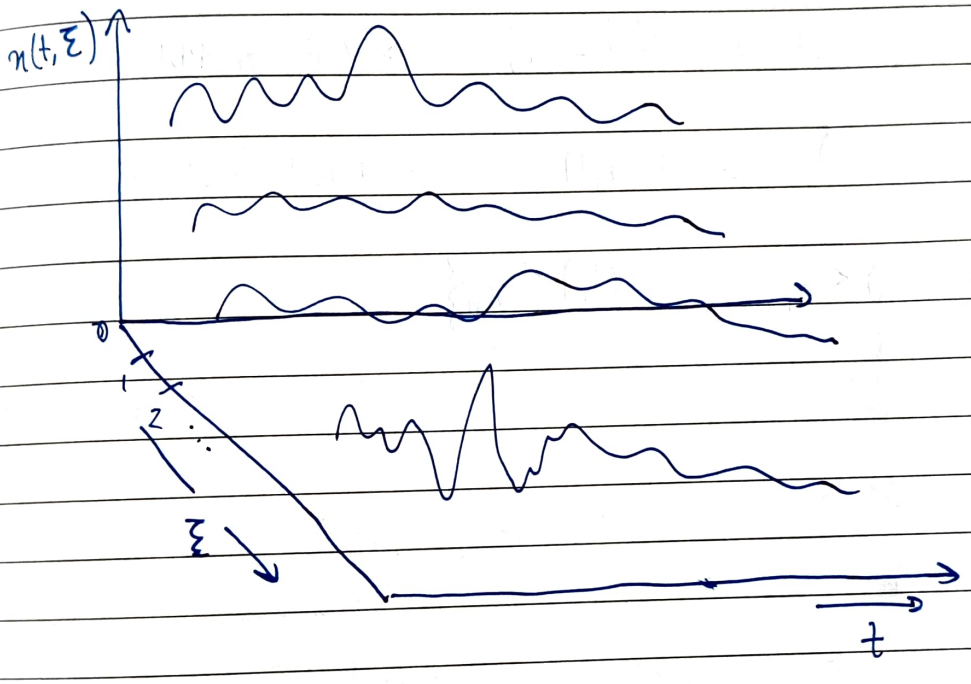
Definition

Let (Ω, F, P) be a probability space.
Then define a mapping X from the sample space Ω to a space of continuous time functions.
The elements in this space will be called sample functions

This mapping is called a random process if at each fixed time, the mapping is a random variable.

$$X(A, \zeta) \in F \rightsquigarrow \{\xi : X(\xi) \leq x\} \subset F \quad \forall x$$

for each fixed 't' on the real line $-\infty < t < \infty$



Definition (Random Process)

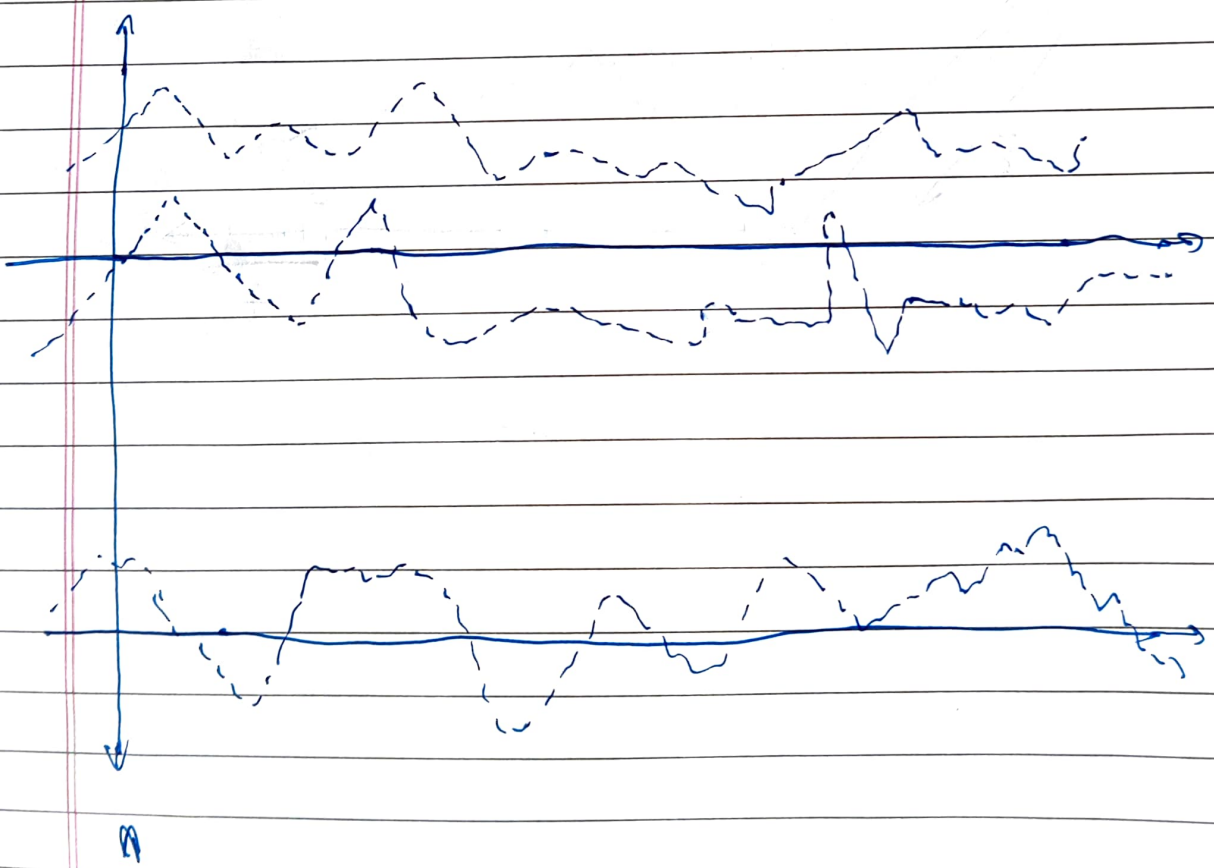
A random process $X(t)$ is statistically specified by its complete set of n^{th} order PDFs for all positive integers 'n',

$F_X(x_1, x_2, \ldots, x_n ; t_1, t_2, \ldots, t_n)$ $\forall$ $x_1, x_2, \ldots, x_n$ and for all $t_1, t_2, \ldots, t_n$

$F(x,t) = P\{X(t) \leq x\}$ → CDF or PDF

$f(x,t) = \frac{d}{dx} F(x,t)$ → Probability density function

$= P\{x(t_1) \leq x_1, \ldots, X(t_n) \leq x_n\}$



Definition

Let $n_t(x)$ denote the no. of trials st at time 't' the ordinates of the observed functions do not exceed 'x'

$$F(x,t) \triangleq \frac{n_t(x)}{n}$$

$$F(x_1, x_2; t_1, t_2) = P\{X(t_1) \leq x_1 , X(t_2) \leq x_2\}$$

Strict Sense Stationary Process

$$f(x_1, x_2, \ldots, x_n; t_1, t_2, \ldots, t_n) = f(x_1, x_2, \ldots, x_n; t_1+c, t_2+c, \ldots, t_n+c)$$

n-dimensional

When you sample the random process with a comb at time t and t+c, the same random variables are obtained.

Joint Probability Distribution will remain constant over time.

Wide Sense Stationary

Mean and Autocorrelation are time invariant

Mean Ergodicity

Find $E\{X(t)\}$ from a single sample function.

Q When can we substitute time averages in place of ensemble averages?

$$\hat{\mu}_X(t_k) = \frac{1}{n}\sum_i X(t_k, \xi_i)$$

(Ensemble Average, over same time stamps in different ensembles)

↑ Capital X

- Consider one sample function $x(t)$ of a stationary with the observation interval $-T \le t \le T$

- The DC value of $x(t)$ (small x) is defined as the time average

$$\mu_x(T) = \frac{1}{2T}\int_{-T}^{T} x(t)\,dt$$

$$E[\mu_x(T)] = \frac{1}{2T}\int_{-T}^{T} E[x(t)]\,dt$$

(μ_x: small x)

$$= \frac{1}{2T}\int_{-T}^{T} \mu_X\,dt$$

$$= \mu_X = \text{Ensemble Mean}$$

(μ_X: capital X)

- A random process $X(t)$ is ergodic in the mean if

i) The time average $\mu_x(T)$ approaches the ensemble average μ_X in the limit

$$\lim_{T\to\infty} \mu_x(T) = \mu_X$$

ii) The variance of $\mu_x(T)$ treated as a random variable approaches zero in the limit as $T \to \infty$

$$\lim_{T\to\infty} \text{var}\left[\mu_x(T)\right] = 0$$

Ergodicity in Auto-Correlation Function

The time averaged auto-correlation function is defined as

Time auto correlation → $R_x(\tau, T) = \frac{1}{2T}\int_{-T}^{T} x(t+\tau)\,x(t)\,dt$

i) $\lim_{T\to\infty} R_x(\tau, T) = R_x(\tau)$ → Ensemble Auto-correlation

ii) $\lim_{T\to\infty} \text{var}\left[R_x(\tau, T)\right] = 0$

Gaussian Random Process

Consider a set of r.v.s $X(t_1) \ldots X(t_n)$ obtained by observing a random process $X(t)$.

If this set of r.v.s is jointly Gaussian for any 'n' with the n-fold being completely specified by

$$\mu_x(t_i) = E[X(t_i)]$$

and the of covariance function $C_{xx}(t_k, t_i)$:-

$$C_{xx}(t_k, t_i) = E\left[\left\{X(t_k) - \mu_x(t_k)\right\} \cdot \left\{X(t_i) - \mu_x(t_i)\right\}\right]$$

where $k, i = 1, 2, \ldots, n$

then the random process is a Gaussian Process

If $\underline{X} = \begin{bmatrix} X(t_1) \\ \vdots \\ X(t_n) \end{bmatrix}$ and $\underline{x}$ denote a realization of $\underline{X}$

then $f_{X(t_1)\ldots X(t_n)}(x_1, x_2, \ldots x_n)$

$$= \frac{1}{(2\pi)^{n/2}\,\Delta^{\frac{1}{2}}} \exp\left[-\frac{1}{2}(\underline{x}-\underline{\mu_x})^T \Sigma^{-1} (\underline{x}-\underline{\mu_x})\right]$$

where $\underline{\mu_x} = \begin{bmatrix} E[X(t_1)] \\ \vdots \\ E[X(t_n)] \end{bmatrix}$, $\Sigma \{C_{XX}(t_k, t_i)\}_{n=1}^{n}$

and Δ is determinant of the covariance matrix Σ

H.V. Show 2-dimensional joint pdf of Gaussian random variable

Read Markovian Process

Multivariate Gaussian PDF :-

$$f_X(x_1, x_2, \ldots, x_K) = \frac{1}{\sqrt{(2\pi)^K |\Sigma|}}\, e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Where $\sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$ is the Mahalanobis Distance of x from μ.

Bivariate Gaussian PDF :-

$$f(x,y) = \frac{1}{2\pi\,\sigma_x \sigma_y \sqrt{1-\rho^2}}\, e^{-\frac{1}{2(1-\rho^2)}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right) + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]}$$

$$\mu = \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$$

Markov Process

A random process whose future probabilities are independent of all past values, and depends only on its present state.

A stochastic process $x(t)$ is called Markov if for every n and $t_1 < t_2 < \dots < t_n$, we have

$$P(x(t_n) < x_n \mid x(t_{n-1}), \dots x(t_1)) = P(x(t_n) \leq x_n \mid x(t_{n-1}))$$

eg→ Coin Toss, Throwing a dice

Non-Markov Process example - Weather

Markov Random Process

a) A continuous valued first order random process satisfies

$$f_X(x_n \mid x_{n-1}, x_{n-2}, \ldots, x_1; t_n, t_{n-1}, \ldots t_1)$$

$$= f_X(x_n \mid x_{n-1}; t_n, t_{n-1})$$

b) Discrete valued (first order) Markov random process satisfies

$$P_X(x_n \mid x_{n-1}, \ldots, x_1; t_n, \ldots t_1)$$

↓ Probability Mass Function

$$= p_X(x_n \mid x_{n-1}; t_n, t_{n-1})$$

$\forall\, x_1, x_2, \ldots, x_n$ and for all $t_1 < \ldots < t_n$ and for all integers n.

$$p_X(x_1, \ldots, x_n; t_1, t_2, \ldots t_n) = p_X(x_1) \cdot p_X(x_2 \mid x_1) \cdot p_X(x_3 \mid x_2) \ldots p_X(x_n \mid x_{n-1})$$

for a 1st order Markov Process

Chapman Kolmogorov Equations

- Consider $t_3 > t_2 > t_1$ and the Markov Process random variable at these time instants
  $X(t_3)$, $X(t_2)$ and $X(t_1)$

- We wish to complete the conditional density of $X(t_3)$ given $X(t_1)$

$$f_p(x_3 \mid x_1) = \int_{-\infty}^{\infty} f_X(x_3 \mid x_2) \cdot f(x_2 \mid x_1)\, dx_2$$

which is known as the Chapman-Kolmogorov equation for the transition density $f_X(x_3 \mid x_1)$ of a Markov process.

Time Series

- The particular realization of a random process is called a time series.
- The sequence $v(n), v(n-1), \ldots, v(n-M)$ represents a time series.
- Mean value function of the process $v(n)$ is defined as

$$\mu(n) = E[v(n)]$$

- Autocorrelation function of the process is defined as

$$r(n, n-k) = E[v(n)\, v^*(n-k)] \quad \text{for } k = 0, \pm 1, \pm 2, \ldots$$

- Autocovariance function of the random process $v(n)$ is

$$c(n, n-k) = E[(v(n) - \mu(n)) \cdot (v(n-k) - v(n-k))^*]$$

for $k = 0, \pm 1, \pm 2$

Therefore $c(n, n-k) = r(n, n-k) - \mu(n)\mu^*(n-k)$

For a Wide Sense Stationary Process :-

$$\mu(n) = \mu \text{ for all } n$$
$$r(n, n-k) = r(k)$$
$$c(n, n-k) = c(k)$$

- Note that when $k=0$, corresponding to time lag of zero

$$r(0) = E[|u(n)|^2]$$
and $c(0) = \sigma_u^2 \rightarrow$ Variance of $u(n)$

Correlation Matrix

Let the $M \times 1$ observation vector $u(n)$ denote the elements of the zero mean time series

$$u(n), u(n-1), \ldots, u(n-M+1)$$

So, $\underline{u}(n) = [u(n), u(n-1), \ldots u(n-M+1)]^T$

matrix

We define the correlation of a stationary discrete-time stochastic process represented by this time-series as

$$R = E[\underline{u}(n)\, \underline{u}(n)^H]$$

'H' denote transpose + complex conjugate

$$R = \begin{bmatrix} r(0) & r(1) & \ldots & r(M-1) \\ r(-1) & r(0) & \ldots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(-M+1) & r(-M+2) & \ldots & r(0) \end{bmatrix}$$

- The element $r(0)$ on the main diagonal is always real valued

Properties of Correlation Matrix

Assuming a discrete time stochastic process.

1. Correlation matrix of a stationary discrete-time random process is Hermitian : $R = R^H$

Another way of ~~stationary~~ stating the Hermitian property

$$r(-k) = r^*(k)$$

where $r(k)$ is the auto-correlation function of the random process $u(n)$ for a lag of 'k'.

- For a WSS process $u(n)$

$$R = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix}$$

2. Correlation matrix of a stationary process discrete time stochastic process is TOEPLITZ

3. Correlation matrix of a DTSP is always positive semi definite and almost always +ve definite.

Let 'a' be an arbitrary (non-zero) $M \times 1$ complex-valued vector. Define the scalar random variable $y = \underline{a}^H \underline{u}(n)$

Taking the Hermitian transpose of both sides and noting that y is a scalar, we get

$$y^* = \underline{u}^H(n)\,\underline{a}$$

The mean square value of the random variable y is

$$E[|y|^2] = E[yy^*] = E[\underline{a}^H \underline{u}(n)\,\underline{u}^H \underline{a}]$$

$$= \underline{a}^H E[\underline{u}(n)\,\underline{u}^H(n)]\,\underline{a}$$

$$= \underline{a}^H R\,\underline{a}$$

Since $E[|y|^2] \geq 0$, it follows that $\underline{a}^H R\,\underline{a} \geq 0$

$$\underline{u}^B(n) = [u(n-M+1),\ u(n-M+2),\ \ldots,\ u(n)]^T \qquad [B \leftarrow \text{Backwards}]$$

$$E[\underline{u}^B(n)\,\underline{u}^B(n)^H] = \begin{bmatrix} r(0) & r^*(1) & \cdots & r^*(M-1) \\ r(1) & r(0) & \cdots & r^*(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r(M-1) & r(M-2) & \cdots & r(0) \end{bmatrix} = R^T$$

Correlation Matrix of Sine Wave Plus Noise

- Let α denote the complex amplitude of the complex sinusoid and
- ω denote its angular frequency
- $v(n)$ denote a sample of the noise assumed to have zero mean
- A sample of the time series consisting of the complex sinusoid plus noise can be written as $u(n) = \alpha e^{j\omega n} + v(n)$

$n = 0, 1, \ldots, N-1$

Note: The sources of the complex sinusoid and the noise are independent of each other.

- By assumption, the noise component $v(n)$ has zero mean, therefore

$$E[u(n)] = \alpha \exp(j\omega n)$$

- To find the auto-correlation function of $u(n)$, we need to ~~keep~~ know the auto-correlation function of $v(n)$.
- We assume that the noise has a specialized auto-correlation function

$$E[v(n)\, v^*(n-k)] = \begin{cases} \sigma_v^2, & k = 0 \\ 0, & k \neq 0 \end{cases}$$

- Therefore $v(n)$ is White Noise.

- Since we have assumed that sources of complex sinusoid and thermal noise are independent, therefore the auto-correlation function of $v(n)$ equals the sum of the autocorrelation functions of its two individual components.

- Therefore, the auto-correlation function of the process $v(n)$ for lag 'k'

$$r(k) = E[v(n)\, v^*(n-k)]$$

$$= \begin{cases} |\alpha|^2 + \sigma_v^2 & , k = 0 \\ |\alpha|^2 \exp(j\omega k) & , k \neq 0 \end{cases}$$

Where $|\alpha|$ is the magnitude of the complex amplitude α

- Given the series of samples $v(n), v(n-1), \ldots, v(n-M+1)$ we may express the correlation matrix of $v(n)$ as

$$R = |\alpha|^2 \begin{bmatrix} 1 + 1/p & e^{j\omega} & \cdots & e^{j\omega(M-1)} \\ e^{-j\omega} & 1 + 1/p & \cdots & e^{j\omega(M-2)} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-j\omega(M-1)} & e^{-j\omega(M-2)} & \cdots & 1 + 1/p \end{bmatrix}_{M \times M}$$

Where $p = \dfrac{|\alpha|^2}{\sigma_v^2}$ is the signal-to-noise ratio (SNR)

Example

Consider the idealized case of a noiseless sinusoid of angular frequency ω.

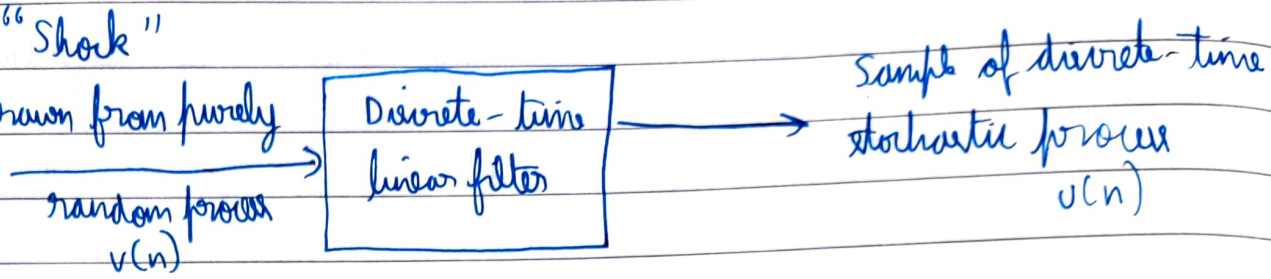
Assume that the time series consists of 3 uniformly spaced samples from this sinusoid.

Hence, setting SNR $\rightarrow \rho \rightarrow \infty$ and the number of samples $M = 3$

$$R = |\alpha|^2 \begin{bmatrix} 1 & e^{j\omega} & e^{j2\omega} \\ e^{-j\omega} & 1 & e^{j\omega} \\ e^{-j2\omega} & e^{-j\omega} & 1 \end{bmatrix}$$

Note: $|R| = 0$

Stochastic Models



- The representation of a stochastic process by a model dates back to an idea by Yule (1972)

- 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.

- The shocks are random variables drawn from a fixed distribution that is usually assumed to be Gaussian with zero mean and constant variance.

- Input is white Gaussian noise

$$E[v(n)] = 0 \quad \forall\ n$$

and $$E[v(n)\,v^*(k)] = \begin{cases} \sigma_v^2, & k = n \\ 0, & \text{otherwise} \end{cases}$$

where σ_v^2 is the noise variance

In general, the Time domain input-output relation for the stochastic model shown in the Figure may be described as

[Present value of model output] + [Linear combination of past values of model output] = [Linear combination of present and past values of model input]

- We may identify 3 types of linear stochastic models

1. Auto-regressive (AR) model, in which NO PAST VALUES of the model input are used.

2. Moving average (MA) models in which NO PAST VALUES of the model output are used

Auto-Regressive Models (AR)

The time series $v(n), v(n-1), \ldots, v(n-M)$ represents the realization of an AR process of order M if it satisfies the difference equation -

$$v(n) + a_1^* v(n-1) + \ldots + a_M^* v(n-M) = v(n)$$

- Reason for calling this auto-regressive -

$$v(n) = w_1^* v(n-1) + w_2^* v(n-2) + \ldots + w_M^* v(n-M) + v(n)$$

where $w_k = -a_k$

Thus, $y = \sum_{k=1}^{M} w_k^* x_k + v$

- LHS of the first equation represents the convolution of the input sequence $\{v(n)\}$ and the sequence of parameters $\{a_n^*\}$

$$\sum_{k=0}^{M} a_k^* \, v(n-k) = v(n) \qquad , (a_0 = 1)$$

Taking Z-transform on both sides, we get

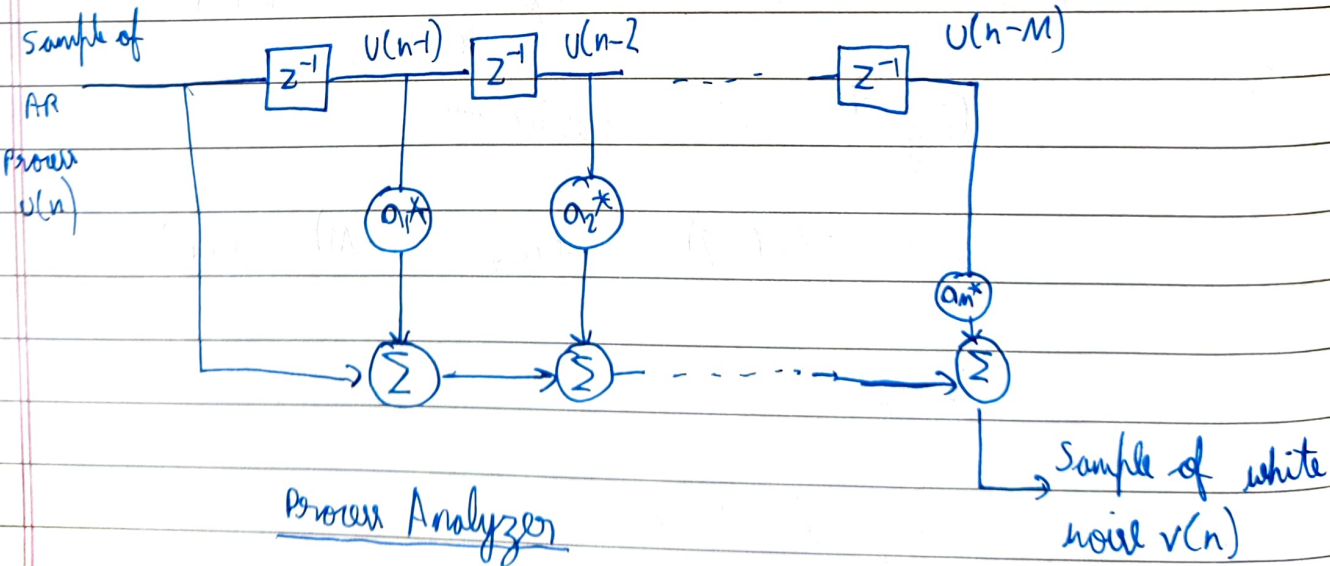
$$H_A(z) \, V(z) = V(z)$$

where $V(z) = \sum_{n=0}^{\infty} v(n) z^{-n}$

Let $H_A(z) = \sum_{n=0}^{M} a_n^* z^{-n}$ denote the Z-transform of the sequence $\{a_n^*\}$

Let $U(z) = \sum_{n=0}^{\infty} u(n) z^{-n}$ denote the Z-transform of the input sequence $\{u(n)\}$

Case I

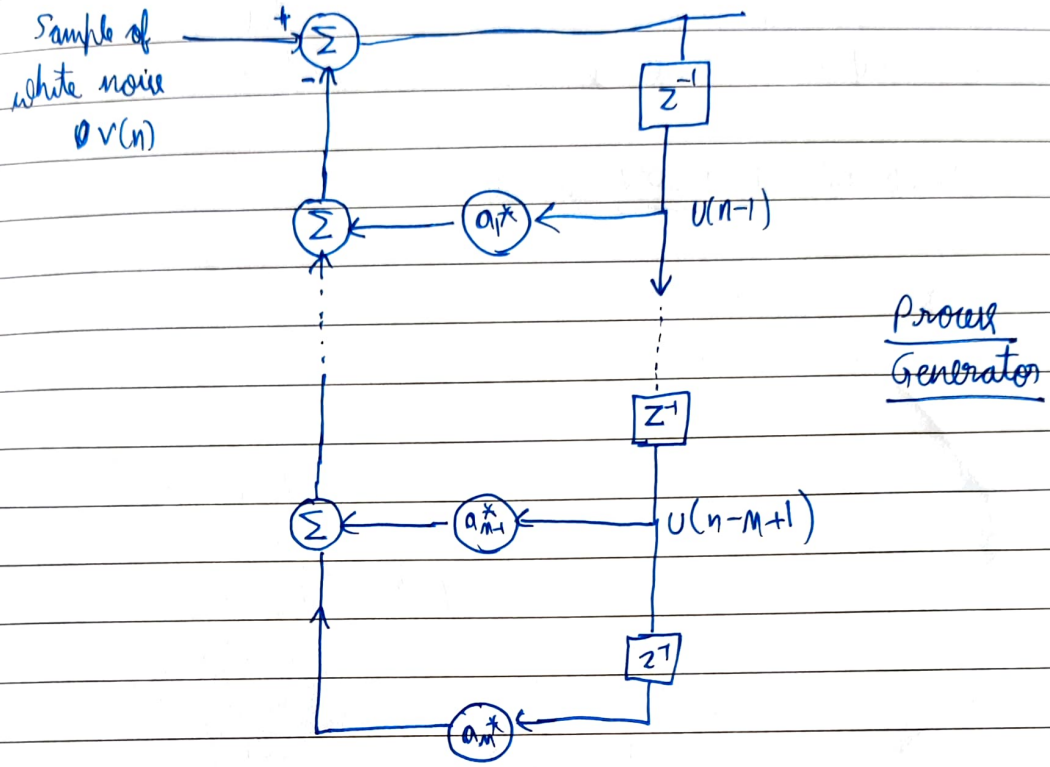


Process Analyzer

The above filter has a transfer function $H_A(z) = \frac{V(z)}{U(z)}$

Impulse response of the AR process analyzer has finite duration.

Case II



- This filter represents a process generator whose TF:

$$H_g(z) = \frac{U(z)}{V(z)} = \frac{1}{H_a(z)} = \frac{1}{\sum_{n=0}^{M} a_n^* z^{-n}}$$

- The impulse response of an AR process generator has infinite duration.

- This AR process generator is an All-Pole filter since $H_g(z)$ can be completely defined by specifying the location of its poles.

$$H_g(z) = \frac{1}{(1-p_1 z^{-1})(1-p_2 z^{-1}) \dots (1-p_M z^{-1})}$$

Where $p_1, p_2, \dots, p_M$ are the poles of $H_g(z)$

- The poles are defined by the roots of the characteristic equation

$$1 + a_1^* z^{-1} + a_2^* z^{-2} + \ldots + a_M^* z^{-M} = 0$$

- For the all-pole AR process generation to be stable, the roots of the above equation must all lie inside the unit circle in the z-pl

Study WOLD DECOMPOSITION yourself

Moving Average Model

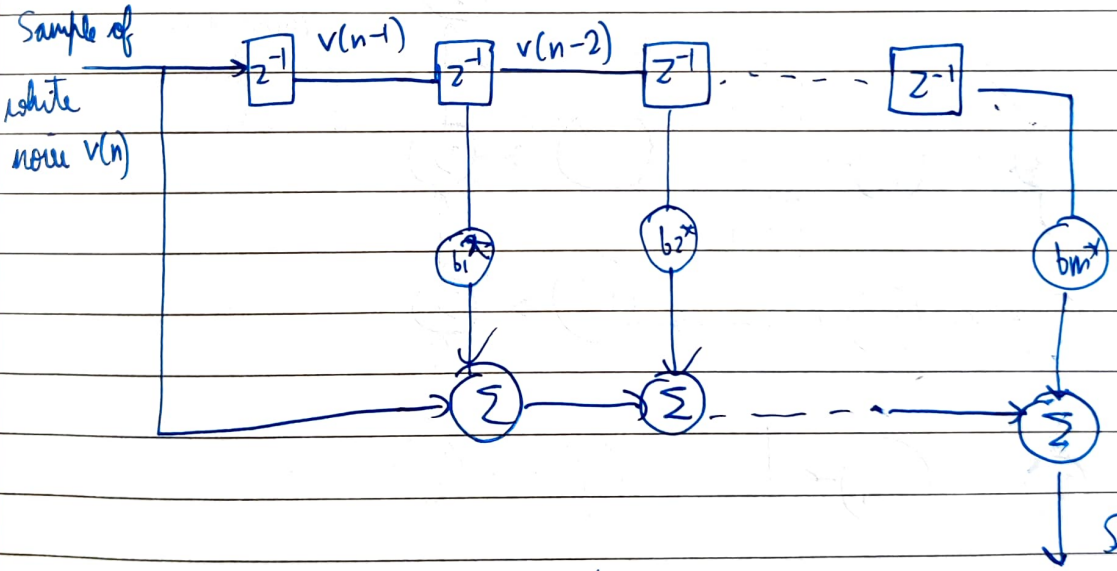
In the MA model, the time linear filter consists of an all-zero filter driven by white noise.

The resulting random $v(n)$ produced at the output can be written as –

$$v(n) = v(n) + b_1 * v(n-1) + \dots + b_K * v(n-K)$$

Where $b_1, b_2, \dots, b_K$ are called MA parameters and $v(n)$ is white noise of zero mean and variance σ_v^2.

- The order of the MA process is K



A Process Generator

- Filters used in the generation and analysis of an MA process are the opposite of those used in the case of an AR process

Auto-Regressive Moving Average (ARMA) Model

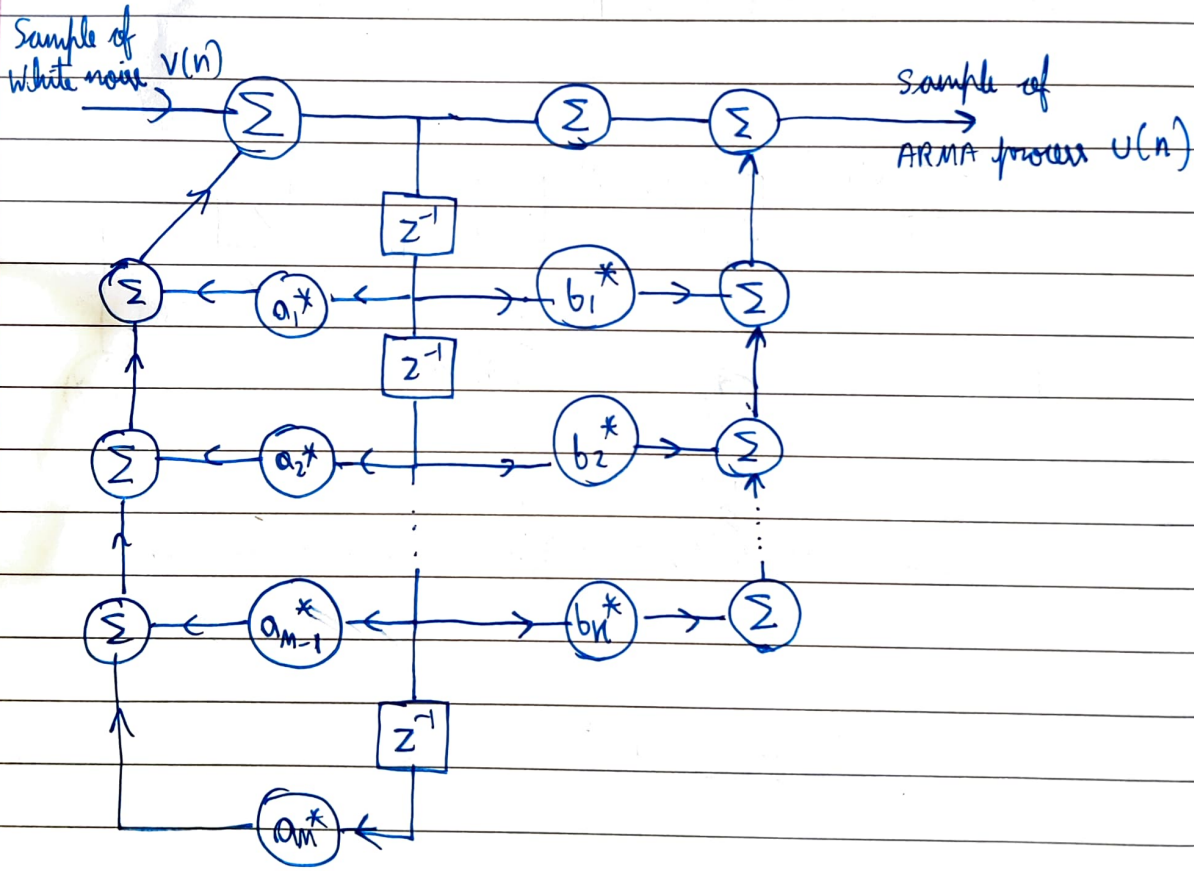
$$u(n) + a_1^* u(n-1) + \dots + a_M^* (u(n-M)) = v(n) + b_1^* v(n-1) + \dots + b_K^* v(n-K)$$

When $a_1, a_2, \dots, a_M$ and $b_1, b_2, \dots, b_K$ are called ARMA parameters

Order of ARMA process = (M, K)

Assuming $M > K$, ARMA model of order (M, K)

Process Generator



- From a computational point of view, the AR models have an advantage over MA and ARMA models.
- Computation of the AR coefficients involves a system of linear equations known as YULE-WALKER equations.

On the other hand, computation of MA and ARMA coefficients require solving of

WOLD Decomposition (1938)

The wide application of AR model is fixed, by a fundamental theorem of time-series analysis.

Theorem: Any stationary discrete-time stochastic process may be decomposed into a sum of a general linear process and a regular predictable process with these two process being uncorrelated with each other.

$$x(n) = u(n) + s(n)$$

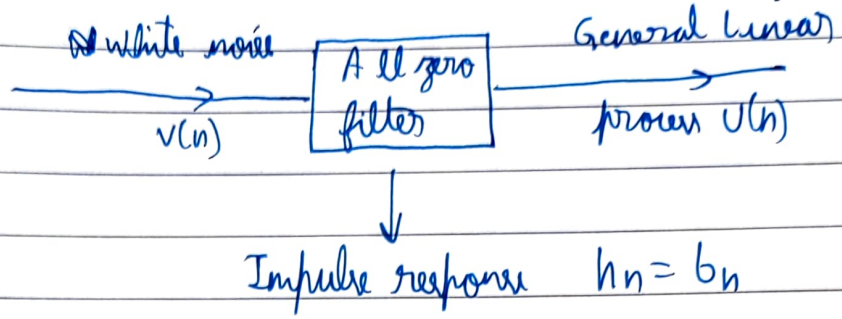
Where $u(n)$ is a general linear process represented by the MA model.

$$u(n) = \sum_{k=0}^{\infty} b_k^* v(n-k)$$

with $b_0 = 1$ and $\sum_{k=0}^{\infty} |b_k|^2 < \infty$ where $v(n)$ is white noise

uncorrelated with $s(n)$ i.e., $E[v(n)s^*(k)] = 0 \; \forall (n,k)$

$s(n)$ is a predictable process, that means, it can be predicted from its own past with zero prediction variance.



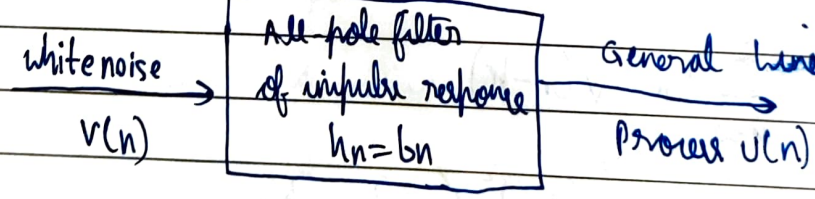
- The general linear process $u(n)$ may be generated by feeding an all-zero filter with white noise $v(n)$.

- The zeroes of the transfer function of this filter equal the roots of the equation

$$B(z) = \sum_{n=0}^{\infty} b_n^* z^{-n} = 0$$

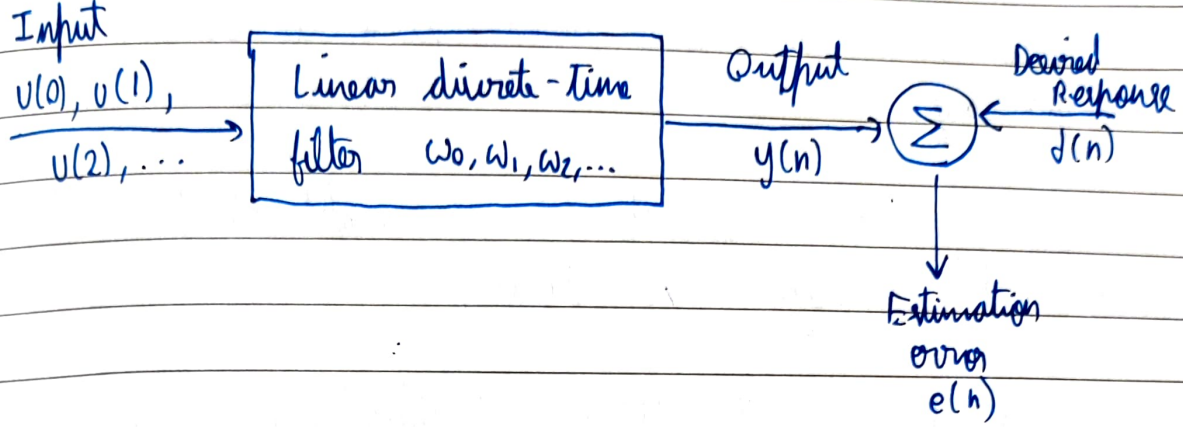
- A solution of particular interest is an all-zero filter that is 'minimum phase', i.e., all the zeroes of the polynomial $B(z)$ lie inside the unit circle.

- In such a case, we may replace the all-zero filter with an equivalent all-pole filter that has the same impulse response $h_n = b_n$



Therefore except for a predictable component, a stationary, discrete-time stochastic process may also be represented as an AR process of the appropriate order subject to the restriction on $B(z)$

Wiener Filter



Linear Optimum Filtering

- Filter input consists of a time series u(0), u(1), u(2), ... and the linear discrete time filter is itself characterised by the impulse response w(0), w(1), ...

- At time instant 'n', the filter produces an output y(n). This output provides an estimate of the desired response y(n).

- Since the filter input and the desired response are single realizations of respective stochastic processes, the estimation is accompanied by an error with its own statistical characterization

$$e(n) = d(n) - y(n)$$

- Requirement is to make the estimation error as small as possible in some statistical sense.

Two restrictions are placed on the filter :

1. The filter is assumed to be Linear to make mathematical analysis easy.

2. The filter operates in discrete-Time to enable the filter to be implemented in software / hardware.

The final details of the filter specification depends upon -

1. Whether the impulse response has finite or infinite duration.
   ↳ FIR or IIR filter?
2. The type of statistical criterion used for optimization.
   ↳ Mathematical tractability

Criterion to choose for Statistical Optimization

We consider optimizing the filter design by minimizing a cost function or index of performance selected from -

1. Mean-square value of the estimation error.
2. Expectation of the absolute value of the estimation error.
3. Expectation of third or higher powers of the estimation error.

This ~~does does~~ choice results in a second order dependance for the cost function on the unknown coefficients in the impulse response of the filter.

Also, the cost function has a single minimum that UNIQUELY defines the optimum statistical design of the filter.

We summarize the problem as -

Design a linear discrete-time filter whose output $y(n)$ provides an estimate of the desired response $d(n)$, given a set of input samples $u(0), u(1), \ldots$ such that the mean-square value of the estimation error $e(n) = d(n) - y(n)$ is minimized.

Principle of Orthogonality

The input is denoted by the time series $u(0), u(1), u(2), \ldots$ and the impulse response of the filter is ~~denoted~~ denoted by $w_0, w_1, w_2, \ldots$ both of which are assumed to be complex valued and of infinite ~~for~~ duration.

The filter output at discrete-time 'n' is defined by the linear ~~con~~ convolution sum -

$$y(n) = \sum_{k=0}^{\infty} w_k^* \, u(n-k) \qquad , n = 0, 1, 2, \ldots$$

- In the complex terminology $w_k^* \, u(n-k)$ represents the scalar version of an inner product
- The purpose of the filter is to produce an estimate of the desired response $d(n)$.
- We assume that the filter input and the desired response ~~are~~ are single realizations of jointly WSS process, both with zero mean.

Estimation of $d(n)$ is naturally accompanied by an error defined as $e(n) = d(n) - y(n)$

- We define the cost function as the mean squared error.

$$J = E[e(n)\, e^*(n)]$$
$$= E[|e(n)|^2]$$

- We have to minimize the above cost function.

- For example input data, the filter coefficients are in general complex functions.

- Let the k^{th} filter coefficient $w_k = a_k + j b_k$, $k = 0, 1, 2, \ldots$

- We define a gradient operator the k^{th} element of which is written in terms of first order partial derivatives

$$\nabla_k = \frac{\partial}{\partial a_k} + j \frac{\partial}{\partial b_k}, \quad k = 0, 1, 2$$

- Applying this gradient operator ∇ to the cost function J, we obtain a multi-dimensional complex gradient vector ∇J, the k^{th} element of which is

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

- For the cost function J to attain its minimum value, all the elements of the gradient vector ∇J must be simultaneously set to zero

$$\nabla_k J = 0, \quad k = 0, 1, 2, \ldots$$

- Under this set of conditions, the filter is said to be optimum in the mean square sense. MMSE - Minimum Mean Squared Error Optimization

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

H.W. Derive these 4 terms ? and report the answers

Answers in next page →

MIT Linear Algebra Gilbert*Strein*

$$\frac{\partial}{\partial a_k} e(n) = -u(n-k)$$

$$\frac{\partial}{\partial b_k} e(n) = j\,u(n-k)$$

$$\frac{\partial}{\partial a_k} e^*(n) = -u^*(n-k)$$

$$\frac{\partial}{\partial b_k} e^*(n) = -j\,u^*(n-k)$$

Substituting the above partial derivatives into $\nabla_k J$ and cancelling common terms.

$$\nabla_k J = -2E[u(n-k)\,e^*(n)]$$

- Let e_o denote the optimal value of the estimation error that results when the filter operates in the optimum condition.

  Therefore $E[u(n-k)\,e_o^*(n)] = 0, \quad k = 0, 1, 2, \ldots$

- In other words, the necessary and sufficient condition for the cost function J to attain its minimum value is for the corresponding value of the estimation error $e_o(n)$ to be orthogonal to each input sample that enters into the estimation of the desired response at time 'n'.

- This constitutes the <u>Principle of Orthogonality</u>.

Corollary to Principle of Orthogonality

- Examining the correlation between filter output $y(n)$ and the estimation error $e(n)$.

$$E[y(n)\, e^*(n)] = E\left[\sum_{k=0}^{\infty} w_k^* u(n-k) e^*(n)\right]$$

$$= \sum_{k=0}^{\infty} w_k^* E[u(n-k) e^*(n)]$$

$$= 0$$

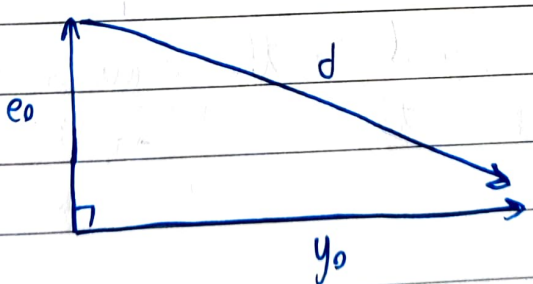
- Let $y_o(n)$ denote the output produced by the filter optimized in the MSE sense, with $e_o(n)$ denoting the corresponding estimation error.

- Hence, using the principle of orthogonality $E[y_o(n) e^*(n)] = 0$

- Let $\hat{d}(n|U_n)$ denote the estimate of the desired response in the MSE sense given the input data that span the space U_n upto the time 'n', then

$$\hat{d}(n|U_n) = y_o(n)$$

Geometric Interpretation



Statistician's Pythagorean Theorem

Wiener - Hopf Equations

- The principle of orthogonality specifies the necessary and sufficient condition for the optimum operation of the filter.

We may write : $E\left[u(n-k)\left(d^*(n) - \sum_{i=0}^{\infty} w_{o,i}\, u^*(n-i)\right)\right] = 0$

for $k = 0, 1, 2, \ldots$

where $w_{o,i}$ is the i^{th} coefficient in the impulse response of the optimum filter.

Re-arranging the terms, we get $\sum_{i=0}^{\infty} w_{o,i}\, E[u(n-k)\, u^*(n-i)]$

$= E[u(n-k)\, d^*(n)]$ for $k = 0, 1, 2, \ldots$

The two expectations in the above equation can be interpreted as -

1. $E[u(n-k)\, u^*(n-i)]$ is equal to the auto-correlation function of the filter input by a lag of $i-k$

$$r(i-k) = E[u(n-k)\, u^*(n-i)]$$

2. $E[u(n-k)\, d^*(n)]$ is equal to the cross-correlation between the filter input $u(n-k)$ and the desired ~~input~~ response $d(n)$ for a lag of $-k$.

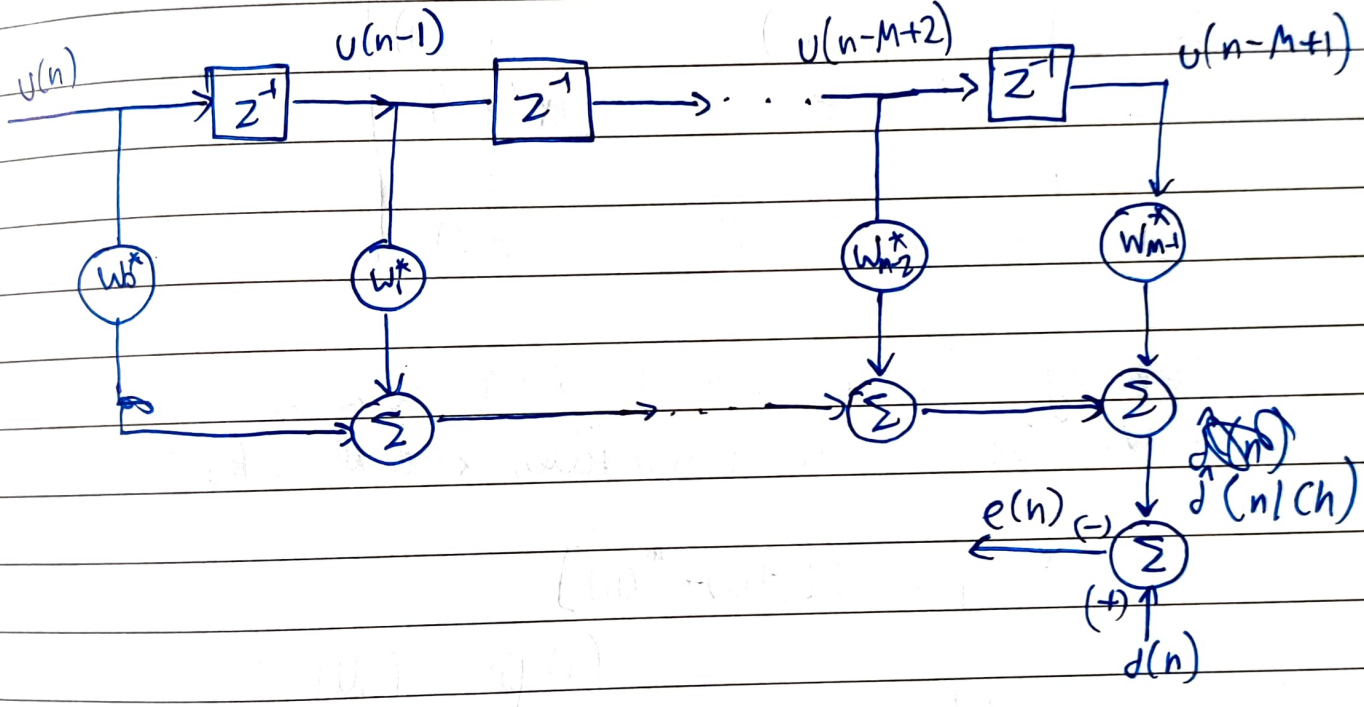
We may thus express $p(-k) = E[u(n-k)\, d^*(n)]$

Therefore, we get an infinitely large system of equations as the necessary and sufficient condition for the optimality of

$$\sum_{i=0}^{\infty} w_{oi}\, r(i-k) = p(-k) \quad , k=0,1,2,\ldots$$

These equations are called Wiener - Hopf ~~eqn~~ equations

Transversal Filter



Accordingly, the Wiener - Hopf equations reduce to a system of M simultaneous linear equations.

$$\sum_{i=0}^{M-1} w_{oi}\, r(i-k) = p(-k) \quad , k=0,1,2,\ldots,M-1$$

where w_{oi} are the optimum weights of the filter.

Matrix Formulation

Let R denote the M×M correlation matrix of the tap inputs
$u(n), u(n-1), \ldots, u(n-M+1)$

in the transversal filter $R = E[\underline{u}(n)\,\underline{u}^H(n)]$

where $\underline{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$

$$R = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \cdots & r(0) \end{bmatrix}$$

Correspondingly, let p denote the M×1 cross-correlation vector between the tap inputs of the filter and the desired response $d(n)$

$$\underline{p} = E[\underline{u}(n)\,d^*(n)]$$

In the expanded form,

$$\underline{p} = [p(0), p(-1), \ldots, p(1-M)]$$

Therefore, the Wiener-Hopf equations can be written in the matrix form as

$$R\,\underline{w}_o = \underline{p}$$

where $\underline{w}_o = [w_{o_0}, w_{o_1}, \ldots, w_{o_{M-1}}]^T$

$$\therefore \underline{w}_o = R^{-1}\,\underline{p}$$

Error Performance Surface

- We can also derive the Wiener-Hopf equations by examining the dependence of the cost function J on the tap weights of the ~~trans~~ transversal filter.

- We write the estimation error as

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

- The cost function of the transversal filter is

$$J = E[e(n)e^*(n)]$$

$$= E[|d(n)|^2] - \sum_{k=0}^{M-1} w_k^* E[u(n-k)d^*(n)]$$

$$- \sum_{k=0}^{M-1} w_k E[u^*(n-k)d(n)] + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i E[u(n-k)u(n-i)^*]$$

1. For the first expectation we have $\sigma_d^2 = E[|d(n)|^2]$ where σ_d^2 is the variance of the desired response $d(n)$, assumed to be of zero mean.

2. For the 2nd and 3rd terms in RHS we have

$$p(-k) = E[u(n-k)d^*(n)]$$
$$p^*(-k) = E[u^*(n-k)d(n)]$$

3. For the 4th expectation, $r(i-k) = E[u(n-k)u^*(n-i)]$

Thus we may write the cost function as

P.T.O.

$$J = \sigma_d^2 - \sum_{k=0}^{M-1} w_k^* p(-k) - \sum_{k=0}^{M-1} w_k p^*(-k) + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i r(i-k)$$

- When the tap inputs of the transversal filter and the desired response are jointly stationary, the cost function or mean-squared error is precisely a second-order function of the tap weights for the filter.

- We may visualize the dependency of J on the tap weights $w_0, w_1, \ldots, w_{M-1}$ as a bowl-shaped $(M+1)$ dimensional surface with M degrees of freedom represented by tap weights.

- This is known as error performance surface (EPS) which has a UNIQUE minimum.

- At the bottom or minimum point of the EPS, the cost J attains its minimum value J_{min}.

- At this point, the gradient vector ∇J is identically zero.

$$\nabla_k J = 0, \quad k = 0, 1, \ldots, M$$

- We write $w_k = a_k + j b_k$

$$\nabla_k J = \frac{\partial J}{\partial a_k} + j \frac{\partial J}{\partial b_k}$$

$$= -2p(-k) + 2\sum_{i=0}^{M-1} w_i r(i-k)$$

- Putting $\nabla_k J = 0$, we obtain $\sum_{i=0}^{M-1} w_{o,i} r(i-k) = p(-k), \quad k = 0, 1, \ldots, M$

Minimum Mean-Square Error

Let $\hat{d}(n|U_n)$ denote the estimate of the desired ~~signal~~ response $d(n)$, produced at the output of the transversal filter that is optimum in the mean square sense, given the tap inputs $u(n), u(n-1), \ldots u(n-M+1)$, that span the space U_n.

$$\hat{d}(n|U_n) = \sum_{k=0}^{M-1} w_{o_k}^{*} u(n-k)$$

$$= \underline{w}_o^H \underline{u}(n)$$

where $\underline{w}_o = [w_{o_0}, w_{o_1}, \ldots, w_{o_{M-1}}]^T$

and $\underline{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$

Hence, the variance of $\hat{d}(n|U_n)$

$$\sigma_{\hat{d}}^2 = E[\underline{w}_o^H \underline{u}(n) \underline{u}^H(n) \underline{w}_o]$$

$$= \underline{w}_o^H E[\underline{u}(n) \underline{u}^H(n)] \underline{w}_o$$

$$= \underline{w}_o^H R \underline{w}_o$$

$$\sigma_{\hat{d}}^2 = \underline{p}^H \underline{w}_o = \underline{p}^H R^{-1} \underline{p}$$

$$J_{min} = \sigma_d^2 - \sigma_{\hat{d}}^2$$

$$= \sigma_d^2 - \underline{w}_o^H R \underline{w}_o$$

$$= \sigma_d^2 - \underline{p}^H \underline{w}_o$$

$$= \sigma_d^2 - \underline{p}^H R^{-1} \underline{p}$$ → Error for optimal weights

Canonical Form of the Error Performance Surface

$$J = \sigma_d^2 - \sum_{k=0}^{M-1} w_k^* p(-k) - \sum_{k=0}^{M-1} w_k p^*(-k) + \sum_{k=0}^{M-1}\sum_{i=0}^{M-1} w_k^* w_i r(i-k)$$

$$J(\underline{w}) = \sigma_d^2 - \underline{w}^H \underline{p} - \underline{p}^H \underline{w} + \underline{w}^H R \underline{w}$$

Express $J(\underline{w})$ as a "perfect square" in $\underline{w}$ by

$$J(\underline{w}) = \sigma_d^2 - \underline{p}^H R^{-1} \underline{p} + (\underline{w} - R^{-1}\underline{p})^H \cdot R(\underline{w} - R^{-1}\underline{p})$$

Therefore, $\min J(\underline{w}) = \sigma_d^2 - \underline{p}^H R^{-1} \underline{p}$ for $\underline{w}_0 = R^{-1}\underline{p}$

We can also write $J(\underline{w}) = J_{min} + (\underline{w} - \underline{w}_0)^H R (\underline{w} - \underline{w}_0)$

Therefore $J(\underline{w}_0) = J_{min}$

- Using eigen-decomposition, express $R = Q \Lambda Q^H$ where Λ is a diagonal matrix consisting of eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ of R and the matrix Q has for its columns the eigenvectors $q_1, q_2, \ldots, q_M$ associated with these eigenvalues.
- Therefore, $J = J_{min} + (\underline{w} - \underline{w}_0)^H Q \Lambda Q^H (\underline{w} - \underline{w}_0)$
- Define a transformed version of the difference between the optimum solution $\underline{w}_0$ and the tap weight vector $\underline{w}$ as

$$\underline{v} = Q^H (\underline{w}_0 - \underline{w})$$

- Hence we can now write $J = J_{min} + \underline{v}^H \Lambda \underline{v}$

This new formulation of the mean-square error contains no cross-product becomes

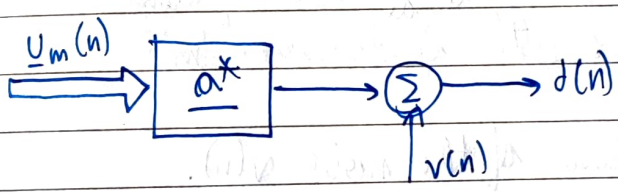
$$J = J_{min} + \sum_{k=1}^{M} \lambda_k v_k v_k^*$$

$$= J_{min} + \sum_{k=1}^{M} \lambda_k |v_k|^2$$

where v_k is the k^{th} component of $\underline{v}$

This representation of the error performance surface is useful because the components of the transformed coefficients vector $\underline{v}$ constitute the Principal Axes of the surface.

Multiple Linear Regression Model



- The maximum mean-square error of the Wiener filter

$$J_{min} = \sigma_d^2 - \underline{p}^H \underline{R}^{-1} \underline{p}$$

applies to a transversal filter configuration of a prescribed length (i.e., no. of tap weights)

- Suppose, it is possible to adjust the length of the transversal filter as a design parameter, can we reduce J_{min} to an irreducible level?

- The answer is to this fundamental question depends on the model describing the relationship between $d(n)$ and the input vector $\underline{u}(n)$.

- We make the following 3 assumptions -

i) The model is linear
ii) The observable data are noisy (measurable)
iii) The noise is additive and white.

$$d(n) = \underline{a}^H \underline{u}_m(n) + v(n)$$

Where $\underline{a}$ denotes an unknown parameter vector of the model, $\underline{u}_m(n)$ denotes the input vector (regressor) and $v(n)$ is the additive white noise that is statistically independent of $\underline{u}_m(n)$

Parameter $\underline{a}$ is also referred to as the regression vector.
Vectors $\underline{u}_m(n)$ and $\underline{a}$ are both of dimension 'm'.

Let σ_v^2 denote the variance of the noise $v(n)$.

Then the variance of the observable data $d(n)$ supplying the desired response is given by

$$\sigma_d^2 = E[d(n)\, d^*(n)]$$
$$= \sigma_v^2 + \underline{a}^H R_m \underline{a}$$

where $R_m = E[\underline{u}_m(n)\ \underline{u}_m^H(n)]$

is the $m \times m$ correlation matrix of the I/P vector.

Now consider a Wiener filter that operates on an input vector $\underline{u}(n)$ and desired response $d(n)$ to produce a minimum mean square error $J_{min}(M)$ which is adjustable by varying filter length 'M'.

The input vector $\underline{u}(n)$ and the corresponding tap-weight vector $\underline{w}_o$ are both $M \times 1$ vectors.

Substituting $\sigma_J^2 = \sigma_v^2 + \underline{a}^H R_m \underline{a}$

into $J_{min} = \sigma_J^2 - \underline{w}_o^H R \underline{w}_o$ we get

$$J_{min}(M) = \sigma_v^2 + (\underline{a}^H R_m \underline{a} - \underline{w}_o^H R \underline{w}_o)$$

Therefore we have 3 regimes of model selection

1. Underfitted model ($M < m$)

In this regime, the Wiener filter exhibits better performance with increasing M for a prescribed value of 'm'.

In particular, the minimum MSE ~~does~~ decreases quadratically with filter length M and

$$J_{min}(0) = \sigma_v^2 + \underline{a}^H R_m \underline{a}$$

is the worst possible value for J_{min} from which the MMSE falls as M is increased.

2. Critically fitted model ($M = m$)

At the critical point, $M = m$, Wiener filter is perfectly matched to the regression model in the sense

$$\underline{w}_o = \underline{a} \text{ and } R = R_m$$

Correspondingly, the MMSE of the Wiener filter attains the irreducible value.

$$J_{min}(M) = \sigma_v^2$$

3. Overfitted Model : $(M > m)$

When the length of the Wiener filter is greater than the model order 'm', the tail end of the tap-weight vector of the Wiener filter is zero.

Therefore, $\underline{w}_o = \begin{bmatrix} \underline{a} \\ \underline{0} \end{bmatrix}$

Correspondingly, the tap-~~weights~~ input vector of the Wiener filter takes the form

$$\underline{u}(n) = \begin{bmatrix} \underline{u}_m(n) \\ \underline{u}_{M-m}(n) \end{bmatrix}$$

Where $\underline{u}_{M-m}(n)$ is an $(M-m) \times 1$ vector made up of past data samples immediately preceding the $m \times 1$ vector $\underline{u}_m(n)$.

- The net result is that, in theory, the same MMSE performance as the critically fitted case is achieved but with a longer filter.

Example

When a regression

$$\underline{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$

Statistical characterization of the assumed to be real valued is as follows.

a) The correlation matrix of input vector is

$$R_4 = \begin{bmatrix} 1.1 & 0.5 & 0.1 & -0.05 \\ 0.5 & & & \\ 0.1 & & & \\ -0.05 & & & \end{bmatrix}$$

b) The cross-correlation vector between the input vector $\underline{u}(n)$ and observable ~~d~~ $d(n)$ is

$$\underline{P} = [0.5272, -0.4458,$$

c) The variance of the observable data $\sigma_d^2 = 0.9486$

d) Variance of the additive white noise $\sigma_v^2 = 0.1066$

Method of Steepest Descent

- Consider a cost function $J(\underline{w})$ that is continuously differentiable function of some unknown weight vector $\underline{w}$
- We want to find an optimal solution $\underline{w}_o$ that satisfies the condition

$$J(\underline{w}_o) \leq J(\underline{w}) \text{ for all } \underline{w}$$

Local Iterative Descent

Starting with an initial guess denoted by $\underline{w}(0)$, generate a sequence of weights $\underline{w}(1), \underline{w}(2), \ldots$ such that the cost function $J(\underline{w})$ is reduced at each iteration of the algorithm

$$J(\underline{w}(n+1)) < J(\underline{w}(n))$$

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

We "hope" that this algorithm will eventually converge on the optimal value $\underline{w}_o$ although there is a ~~distant~~ distinct possibility that the algorithm will diverge unless certain precautions are taken.

- In a simple form of iterative descent known as the "method of steepest descent" successive adjustments made to the weight vector $\underline{w}$ are in the ~~discrete~~ direction of steepest descent - opposite to gradient vector of $J(\underline{w})$

$$\text{Let } \underline{g} = \nabla J(\underline{w}) = \frac{\partial}{\partial \underline{w}} J(\underline{w})$$

Therefore, the steepest descent algorithm is formally described as

$$\underline{w}(n+1) = \underline{w}(n) - \frac{1}{2}\mu \underline{g}(n)$$

where 'n' denotes the iteration, μ is a positive constant called the 'step-size' parameter.

- To go from iteration 'n' to 'n+1', the algorithm

Taylor Series

- Taylor series of a function is an infinite sum of terms that are expressed in terms of the function's derivatives at a single point.

- If zero is the point where the derivatives are considered a Taylor series is also called Maclaurin series.

Definition

- The Taylor series of a real or complex valued function $f(x)$ that is infinitely differentiable ...

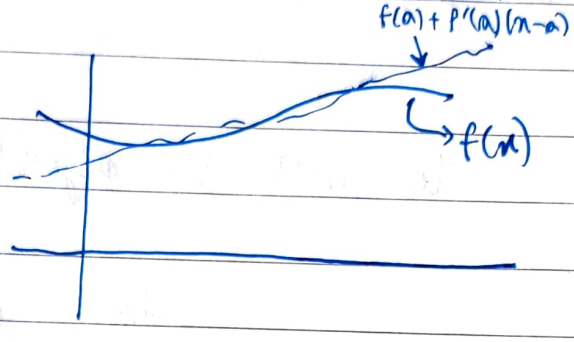
$$f(x) \cong f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \dots$$

$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n$$

Visualization

Best linear approximation for $f(x)$ at $x = a$ is

$$f(x) \approx f(a) + f'(a)(x-a)$$



Multiple Variables

$$f(x,y) \approx f(a,b) + (x-a) f_x(a,b) + (y-b) f_y(a,b)$$

$$+ \frac{1}{2!} \left\{ (x-a)^2 f_{xx}(a,b) + 2(x-a)(y-b) f_{xy}(a,b) + (y-b)^2 f_{yy}(a,b) \right\}$$

$$+ \dots$$

For more than 2 variables, Taylor series of scalar valued func is

$$f(\underline{x}) \approx f(\underline{a}) + (\underline{x}-\underline{a})^T Df(\underline{a}) + \frac{1}{2!} (\underline{x}-\underline{a})^T D^2 f(\underline{a}) (\underline{x}-\underline{a})$$

$$+ \dots$$

Where $Df(\underline{a})$ is the gradient evaluated at $\underline{x} = \underline{a}$ and $D^2 f(\underline{a})$ is the Hessian matrix

Missed Filter & diagram between this page and previous

Steepest Descent technique approach of Wiener filter

The estimation error $e(n) = d(n) - \hat{d}(n|U_n)$

$= d(n) - \underline{w}^H(n)\, u(n)$

Where $\underline{w}(n) = [w_0(n), w_1(n), \ldots, w_{M-1}(n)]^T$

and $\underline{u}(n) = [u(n), u(n-1), \ldots, u(n-M+1)]^T$

If the tap input $\underline{u}(n)$ and the desired response $d(n)$ are jointly stationary, then the MSE or cost function $J(\underline{w}(n))$ or simply $J(n)$ at time 'n' is a quadratic function of the tap weight vectors.

$$J(n) = \sigma_d^2 - \underline{w}^H(n)\,\underline{p} - \underline{p}^H\,\underline{w}(n) + \underline{w}^H(n)\,R\,\underline{w}$$

(from Canonical form of Error Performance Surface)

$$\nabla J(\underline{w}(n)) = \frac{d}{d\underline{w}} J(\underline{w}(n))$$

$$\nabla J(n) = \begin{bmatrix} \frac{\partial J(n)}{\partial a_0(n)} + j\frac{\partial J(n)}{\partial b_0(n)} \\ \frac{\partial J(n)}{\partial a_1(n)} + j\frac{\partial J(n)}{\partial b_1(n)} \\ \vdots \\ \frac{\partial J(n)}{\partial a_{M-1}(n)} + j\frac{\partial J(n)}{\partial b_{M-1}(n)} \end{bmatrix} = -2\underline{p} + 2R\,\underline{w}(n)$$

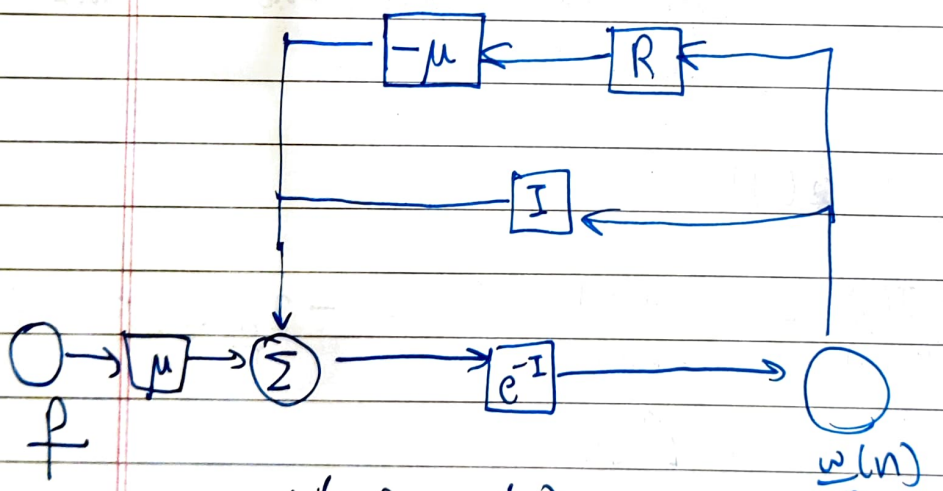
- We assume that the correlation matrix R and the cross-correlation vector $\underline{p}$ are known, so that we may compute the gradient vector $\underline{\nabla} J(n)$ for a given value of the tap weight vector $\underline{w}(n)$

- Thus, we may compute the updated value of the tap weight vector $\underline{w}(n+1)$ by using the recursive relation

$$\underline{w}(n+1) = \underline{w}(n) + \mu [\underline{p} - R\, \underline{w}(n)] \quad \text{for } n = 0, 1, 2, \ldots$$

which describes the mathematical ~~for~~ of the steepest descent algorithm for Wiener filtering

Signal Flow Graph Representation



$$\underline{w}(n+1) = \underline{w}(n) + \mu \underline{p} - \mu R \underline{w}(n)$$

Stability of Steepest Descent

- To determine the condition for the stability of the gradient descent algorithm, we examine Natural Modes of it.

- In particular, we use the representation of the correlation matrix, R

We know $\underline{p} = R\underline{w}_o$ and

$$\underline{w}(n+1) = \underline{w}(n) + \mu[\underline{p} - R\underline{w}(n)]$$

which can be written as $\underline{w}(n+1) = \underline{w}(n) + \mu[R\underline{w}_o - R\underline{w}(n)]$

$$= \underline{w}(n) + \mu R\,\underline{c}(n)$$

and
$$\begin{aligned}\underline{c}(n+1) &= \underline{w}_o - \underline{w}(n+1)\\ &= \underline{w}_o - \underline{w}(n) - \mu R\,\underline{c}(n)\\ &= \underline{c}(n) - \mu R\,\underline{c}(n)\\ &= [I - \mu R]\,\underline{c}(n)\end{aligned}$$

Expressing $R = Q\Lambda Q^H$

Q is called the UNITARY matrix of the orthogonal set of eigenvalues associated with eigen values of R.

Λ is a diagonal matrix containing $\lambda_1, \lambda_2, \ldots, \lambda_M$ eigenvalues of the correlation matrix R.
These eigenvalues are positive and real.

- We can write $\underline{c}(n+1) = (I - \mu Q\Lambda Q^H)\,\underline{c}(n)$ since Q is unitary, $Q^H = Q^{-1}$

Therefore, $Q^H\underline{c}(n+1) = (I - \mu\Lambda)\,Q^H\underline{c}(n)$

Define a new set of coordinates

$$\underline{v}(n) = Q^H \underline{c}(n) = Q^H [\underline{w}_o - \underline{w}(n)]$$

So, $\underline{v}(n+1) = (I - \mu\Lambda)\,\underline{v}(n)$

The initial value of $\underline{v}(n)$ is

$$\underline{v}(0) = Q^H [\underline{w}_o - \underline{w}(0)]$$

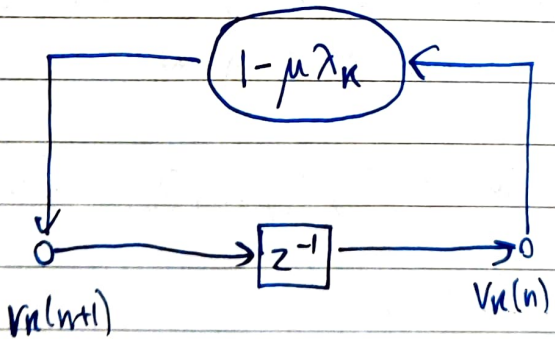
Assuming that the initial tap weight vector is zero, i.e., $\underline{w}(0) = 0$
We get

$$\underline{v}(0) = Q^H [\underline{w}_o]$$

- For the k^{th} natural mode of the steepest descent algorithm we have

$$v_k(n+1) = (1 - \mu\lambda_k)\, v_k(n)$$

where $k = 0, 1, 2, \ldots, M$
and λ_k is the k^{th} eigenvalue of R



Signal flow graph representation of the k^{th} natural mode of the steepest descent algorithm.

- The above equation is a first order homogeneous difference equation

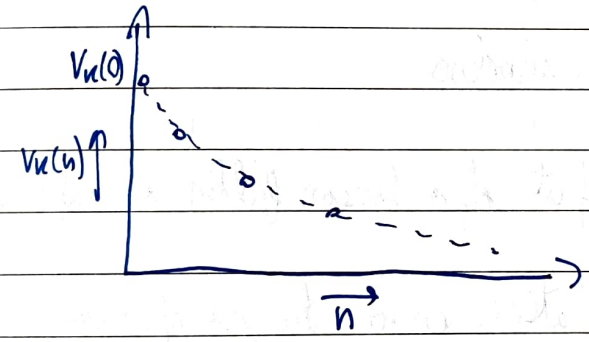
Assuming that $v_k(n)$ has the initial value $v_k(0)$, we readily obtain the solution

$$v_k(n) = (1-\mu\lambda_k)^n v_k(0)$$

where $k = 0, 1, 2, \ldots, M$

For stability of convergence of the steepest descent algorithm, the magnitude of the geometric ratio must be less than UNITY for all values of 'k'.

If $-1 < 1-\mu\lambda_k < 1$ for all k then as the number of iterations 'n' approaches infinity, the natural modes of the steepest descent algorithm die out irrespective of the initial conditions



Referring to $\underline{v}(n) = Q^H(\underline{w}_o - \underline{w}(n))$ it is equivalent to saying that $\underline{w}(n)$ approaches the optimal solution $\underline{w}_o$ as $n \to \infty$

Since eigenvalues of R are real and positive, it follows that a necessary and sufficient condition for the convergence or stability of the steepest descent algorithm is

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

CT - 1 / 10%
Assignments - 10% (might be more)

LMS Adaptive Filter

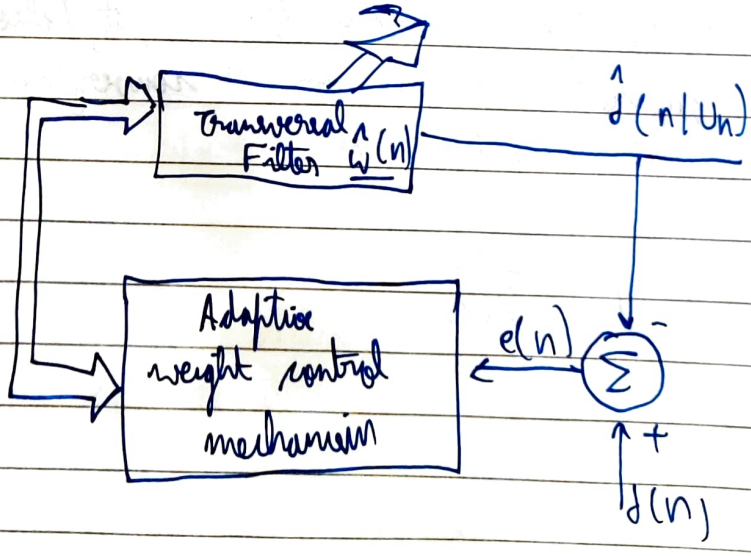
- Does not require measurement of R and p and also does not require any matrix inversion.

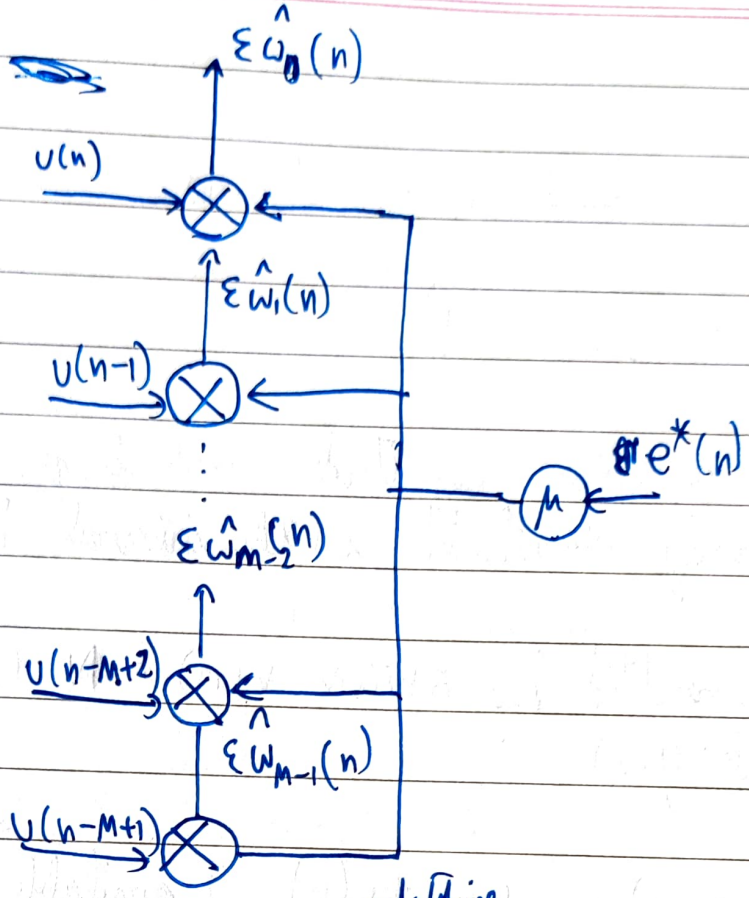
Overview

LMS algorithm is a linear adaptive filtering algo which has two processes -

i) A filtering process which involves
   a) computing the output of a linear filter in response to an input signal
   b) generating an estimation error by comparing this output with a desired response.

ii) An adaptive process, which involves automatic adjustment of the parameters of the filter in accordance with the estimation error





Detailed ~~Determined~~ structure of the ~~filter~~ adaptive weight control mechanism

- The tap weights $u(n), u(n-1), \ldots u(n-M+1)$ form the ~~components~~ components of the $M \times 1$ tap ~~vector~~ input vector $\underline{u}(n)$.
- Then inputs span a multi ~~dimensional~~ dimensional array space denoted by U_n

NOTE The value computed for the vector using the LMS algorithm represents an estimate whose expected value may come close to the Wiener-Solution w_o (for a wss measurement) as the no. of iterations 'n' approaches infinity.

- The estimation error is defined by $e(n) = d(n) - \hat{i}(n|U_n)$

- The estimation error $e(n)$ are the tap input vector & $u(n)$ are applied to the control mechanism and the feedback loop around the tap weights is then closed.

- In the adaptive weight-control mechanism module, a scalar version of the inner product of the estimation error $e(n)$ and the tap input $u(n-k)$ computed for $n = 0, 1, \ldots, M-1$

- The scaling factor used in this computation is called the step size

- The result of the scalar inner-product the correlation $\delta w_k(n)$ applied to the tap-weights $w_k(n)$ at the iteration $(n+1)$

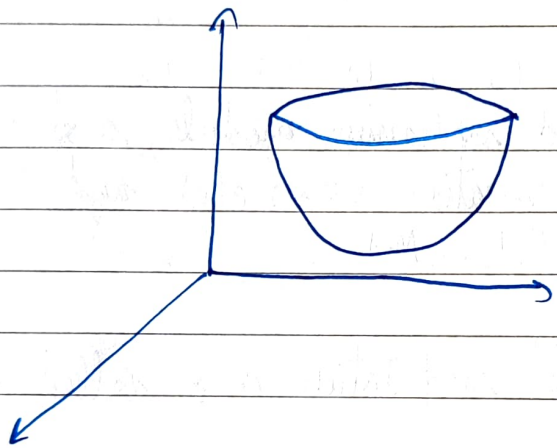
$x(n)$

$\downarrow$

$(n-M+1)$

- Comparing the control mechanism for the LMS algorithm with the method of steepest descent, we observe that the LMS algorithm uses the product $u(n-k)\,e^*(n)$ as an estimate of descent in the gradient vector $\nabla J(n)$ that characterizes the method of steepest descent.

- Now there is NO expectation operator in all the paths. Therefore, the recursive computation of each tap weight in the LMS algorithm suffers from a gradient noise



For such an environment, we know that the method of steepest descent computes a tap vector $\underline{w}(n)$ that moves down the ensemble average error performance surface along a deterministic trajectory that terminates on the Wiener soln $\underline{w}_o$

The LMS algo behaves differently due to presence of gradient noise. Rather than terminating on Wiener sol, the tap weight vector $\underline{\hat{w}}(n)$ computed by LMS algorithm executes a random motion around the minimum point of the Error Performance Surface. It can be shown that this motion is a form of Brownian motion for small values of μ.

Date
Page

LMS Algorithm

- If it were possible to make exact measurements of the gradient vector $\nabla J(n)$ at each iteration 'n' and if 'μ' is suitably chosen, then the tap-weight vector computed using the steepest descent algorithm would indeed converge to the optimum Wiener soln
- However in practice, exact measurements of the gradient vector are not possible as this requires prior knowledge of R and p.
- The simplest choice of estimation is to use instantaneous estimates of R and p that are based on sample values of the tap-input vector and the desired response

$$\hat{R}(n) = \underline{u}(n)\,\underline{u}^{H}(n)$$
$$\hat{p}(n) = \underline{u}(n)\,d^{*}(n)$$

- Therefore the instantaneous estimate of the gradient vector is

$$\hat{\nabla} J(n) = -2\hat{p}(n) + 2\hat{R}\,\hat{\underline{w}}(n)$$
$$= -2\,\underline{u}(n)\,d^{*}(n) + 2\,\underline{u}(n)\,\underline{u}^{H}(n)\,\hat{w}(n)$$

- Substituting this estimate of the gradient vector we get a new recursive solution for updating the tap weight vector

$$\hat{\underline{w}}(n+1) = \hat{w}(n) + \mu\,\underline{u}(n)\left[d^{*}(n) - \underline{u}^{H}(n)\,\hat{w}(n)\right]$$

- A that we have a hat operator over the symbol for to tap weight vector to distinguish it from the value obtained from steepest descent algorithm.

- We may write the above result in the form of 3 basic relations

1. Filter output $y(n) = \underline{\hat{w}}^H(n)\, \underline{u}(n)$

2. Estimation error or error $e(n) = d(n) - y(n)$

$$e^*(n) = d^*(n) - y^*(n)$$
$$= d^*(n) - \underline{u}^H(n)\, \underline{\hat{w}}(n)$$

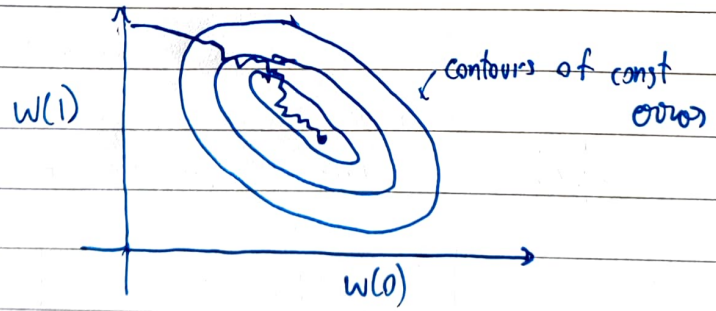
Note: $\langle \underline{a}, \underline{b} \rangle = \overline{\langle \underline{b}, \underline{a} \rangle}$

So $\underline{y}^H \underline{x} = (\underline{x}^H \underline{y})^* = \underline{y}^H \underline{x}$

3. Tap-weight adjustment

$$\hat{w}(n+1) = \underline{\hat{w}}(n) + \mu\, \underline{u}(n)\, e^*(n)$$

- The term $\mu\, \underline{u}(n)\, e^*(n)$ represents the adjustment that is applied to the current estimate of the tap weight vector $\underline{\hat{w}}(n)$.



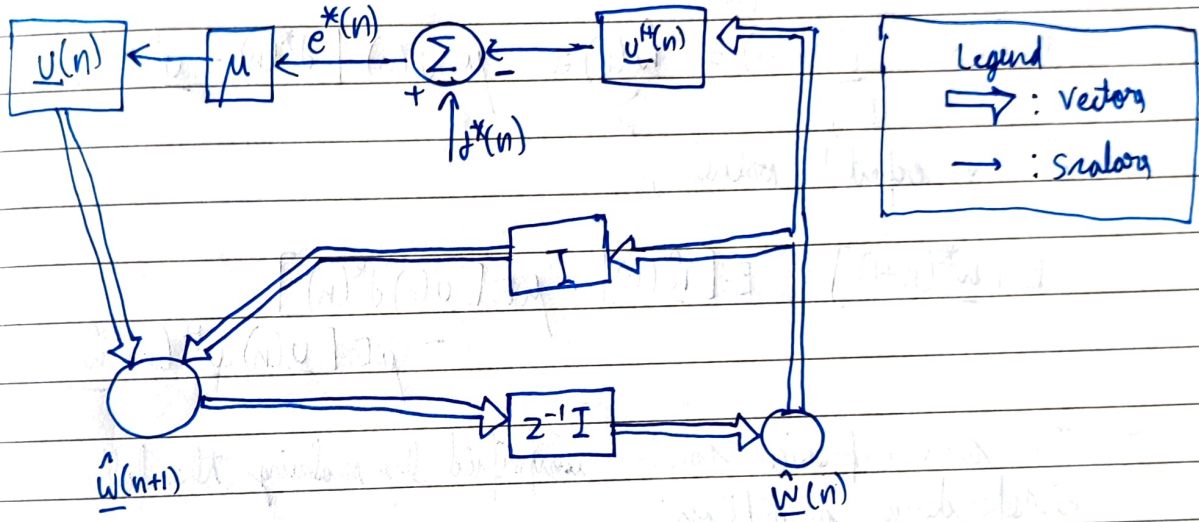
- The LMS algorithm is a member of the family of stochastic gradient algorithms

- In particular, when the LMS Algorithm operates on stochastic inputs, the allowed set of directions along which we step

from one iteration cycle to the next is quite random and cannot therefore be thought of consisting of true gradient direction.

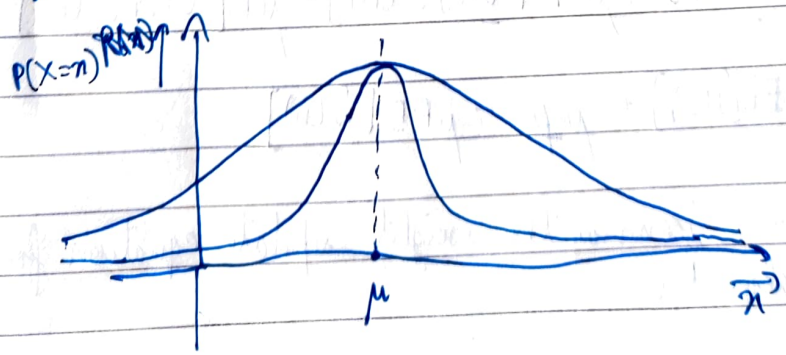
- Although the correction (update) that is applied to $\hat{\underline{w}}(n)$ is generally not aligned with the direction of steepest descent, the correction is on the average in the direction of the steepest descent.

Signal Flow Graph Representation



- The computational complexity of the LMS Algorithm is $O(M)$.

- The instantaneous estimates of R and p have relatively large variances but since LMS algorithm is recursive in nature, the algorithm itself effectively averages these estimates in some sense during the course of adaptation.



Convergence of LMS Algorithm

We begin by assuming that $\underline{u}(n)$ and $d(n)$ are jointly stationary processes.

We will determine when the coefficients $\hat{\underline{w}}(n)$ converge in the mean to $\underline{w}_0 = R^{-1}p$

$$\lim_{n\to\infty} E[\hat{\underline{w}}(n)] = \underline{w}_0 = R^{-1}p$$

- We have $\hat{\underline{w}}(n+1) = \hat{\underline{w}}(n) + \mu\,\underline{u}(n)\,[d^*(n) - \underline{u}^H(n)\,\hat{\underline{w}}(n)]$

Taking expected values,

$$E[\underline{w}^*(n+1)] = E[\hat{\underline{w}}(n)] + \mu E[\underline{u}(n)d^*(n)] - \mu E[\underline{u}(n)\,\underline{u}^H(n)\,\hat{\underline{w}}(n)]$$

- The above expression can be simplified by making the following independence assumption:

* The data $\underline{u}(n)$ and the weight vector $\hat{\underline{w}}(n)$ are statistically independent

The above equation can now be simplified as -

$$E[\hat{\underline{w}}(n+1)] = E[\hat{\underline{w}}(n)] + \mu E[\underline{u}(n)d^*(n)] - \mu E[\underline{u}(n)\,u^H(n)]\,E[\hat{\underline{w}}(n)]$$

$$= E[\hat{\underline{w}}(n)] + \mu p - \mu R E[\hat{\underline{w}}(n)]$$

which is in the same form as the weight update equation for the steepest descent algorithm

- Therefore, the analysis for the steepest descent algorithm is applicable to $E[\hat{w}(n+1)]$

- We already know that $\underline{v}(n) = (I - \mu\Lambda)^n \underline{v}(0)$

where $\underline{v}(n) = Q^H [\underline{w}_o - \underline{w}(n)]$

- If $E[\underline{v}(n)]$ converges to 0, then $\hat{\underline{w}}(n)$ will converge to mean.

- Thus we have the following property:

For jointly WSS processes, the LMS algo converges in the mean if

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

and the independence assumption is satisfied.

Note that although this bound on μ ensures that $E[\hat{w}(n)]$ converges it places no constraints on how large the variance of $\hat{w}(n)$ may become.

- The upper bound on μ is expressed in terms of the highest eigenvalue of R but if this matrix is unknown, the λ_{max} has to be estimated as

$$\lambda_{max} \leq \sum_{k=0}^{M-1} \lambda_k = tr[R]$$

- If $\underline{u}(n)$ is WSS, then R is Toeplitz and the trace becomes

$$tr[R] = M r_u(0) = ME[|u(n)|^2]$$

Therefore the bounds on μ can be replaced as

$$0 < \mu < \frac{2}{ME\{|u(n)|^2\}}$$

Where $E[|u(n)|^2]$ represents the power in $\underline{u}(n)$

- $$\hat{E}[|u(n)|^2] = \frac{1}{M} \sum_{n=0}^{M-1} |u(n-n)|^2$$

- Therefore the following bound step size parameter, μ, for mean-square convergence

$$0 < \mu < \frac{2}{\underline{u}^H(n)\,\underline{u}(n)}$$

Normalized LMS Algorithm

- A convenient way to incorporate this bound into the LMS adaptive filter is to use a time varying step size

$$\text{Then } \mu(n) = \frac{\beta}{\underline{u}^H(n)\,\underline{u}(n)} = \frac{\beta}{\|u(n)\|^2}$$

where $0 < \beta < 2$

- The weight update relationship is given as

$$\hat{\underline{w}}(n+1) = \hat{\underline{w}}(n) + \frac{\beta \underline{u}(n)}{\|u(n)\|^2} \cdot e^*(n)$$

- Note: The effect of the normalization by $\|u(n)\|^2$ is to alter the magnitude but not the direction of the estimated gradient vector.

- we know that in the LMS algorithm the correction which is applied to $\hat{w}(n)$ is proportional to the input vector $u(n)$.

- Therefore, when $u(n)$ is large, the LMS algorithm experiences a problem called "gradient noise amplification".

- With the normalization of the step-size by $\|u(n)\|^2$ in the LMS algorithm, this gradient noise amplification problem is diminished.

- When $\|u(n)\|$ becomes too small,

$$\hat{w}(n+1) = \hat{w}(n) + \beta \frac{u(n)}{\epsilon + \|u(n)\|^2} \cdot e^*(n)$$

where ϵ is a small +ve no number

Linear Prediction

- Consider a time series $u(n)$, $u(n-1)$ representing $(M+1)$ samples of a stationary discrete-time stochastic process.

- The operation of prediction involves using $u(n-1), u(n-2), \ldots, u(n-M)$ to make an estimate of $u(n)$

- Let U_{n-1} denote the M-dimensional space spanned by the samples $u(n-1), u(n-2), \ldots, u(n-M)$.
  Then $\hat{u}(n, U_{n-1})$ denotes the predicted value of $u(n)$ given these samples.

- In linear prediction, we express this predicted value as a linear combination of $u(n-1), u(n-2), \ldots, u(n-M)$.

- The operation corresponds to one-step prediction of the future measured with respect to time $(n-1)$.

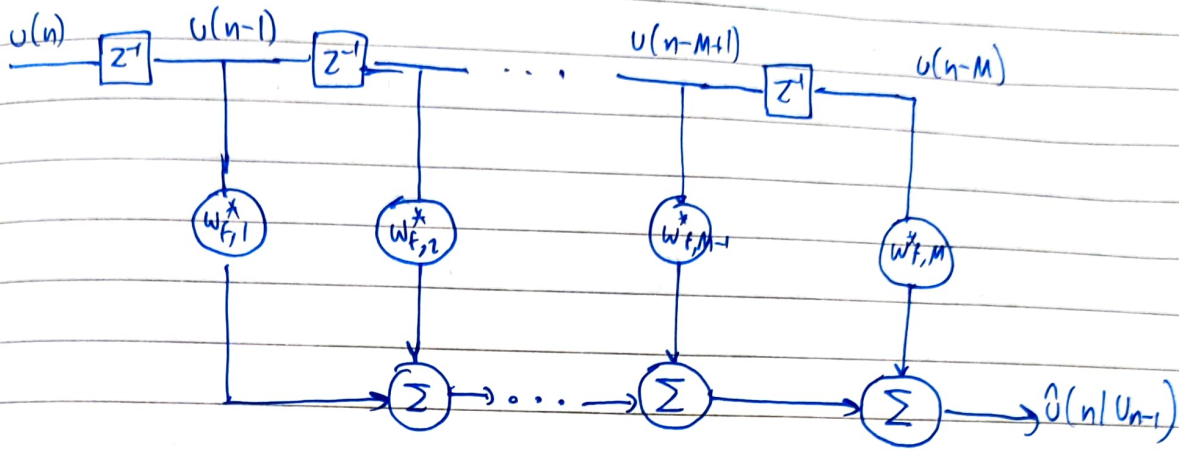
- Accordingly, we refer to this form of prediction as one-step linear prediction, simply, forward linear prediction.

- In another form of prediction, we use the samples $u(n), u(n-1), \ldots u(n-M+1)$ to make a prediction of past sample $u(n-M)$.

- We refer to the second form of prediction as backward linear prediction

- We will use the Wiener filter theory to optimize the design of a forward or backward predictor in the mean-square error sense, for the case of WSS discrete-time stochastic process.

Forward Linear Prediction



- The figure shows a forward prediction that consists of a linear transversal filter with M tap weights $w_{f,1}, w_{f,2}, \ldots w_{f,M}$ and tap inputs $u(n-1), u(n-2), \ldots, u(n-M)$ respectively.

- Assume that the tap inputs are drawn from a WSS process with zero mean.

- We further assume that the tap-weights are optimized in the mean-square error sense in accordance with Wiener filter theory.

The predicted value is $\hat{u}(n|U_{n-1}) = \sum_{k=1}^{M} w^*_{f,k}\, u(n-k)$

Note that for the situation described above the desired response $d(n)$ equals $u(n)$

$$d(n) = u(n)$$

The forward prediction error equals

$$f_M(n) = u(n) - \hat{u}(n|U_{n-1})$$

- The subscript M for the forward prediction error signifies the order of the predictor defined as the number of unit-delay elements needed to store the given set of samples used to make the prediction.

- Let $P_m = E[|f_{f,m}(n)|^2]$ for all n denote the minimum mean square prediction error.

- With the tap inputs assumed to have zero mean, the forward prediction error $f_{f,m}(n)$ will likewise have 0 mean.

- Under this condition, P_m equals the variance of the forward prediction error.

- Let w_f denote the $M \times 1$ optimum tap-weight vector of the forward prediction

$$\underline{w}_f = [w_{f,1}, w_{f,2}, \ldots, w_{f,m}]^T$$

- Let the $M \times 1$ tap-input vector

$$\underline{u}(n-1) = [u(n-1), u(n-2), \ldots, u(n-M)]^T$$

Hence the correlation matrix of the tap inputs equals

$$R = E[u(n-1)\, u^{*H}(n-1)]$$

$$= \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r^*(1) & r(0) & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ r^*(M-1) & \cdots & \cdots & r(0) \end{bmatrix}$$

where $r(k)$ denotes the auto-correlation of the input process for lag 'k' where $k = 0, 1, \ldots, (M-1)$

- The cross correlation vector b/w these top inputs $u(n-1), u(n-2), \ldots u(n-M)$ and the desired response $u(n)$ is

$$\underline{r} = E[\underline{u}(n-1)\, u^*(n)]$$

$$= \begin{bmatrix} r^*(1) \\ r^*(2) \\ \vdots \\ r^*(M) \end{bmatrix} = \begin{bmatrix} r(-1) \\ r(-2) \\ \vdots \\ r(-M) \end{bmatrix}$$

- The variance of $u(n)$ equals $r(0)$ since $u(n)$ has 0 mean.

- We may ~~adopt~~ adapt the Wiener ~~Hotter~~-Hopf equations to solve the forward linear prediction problem for stationary inputs.

$$R\, \underline{w}_f = \underline{r}$$

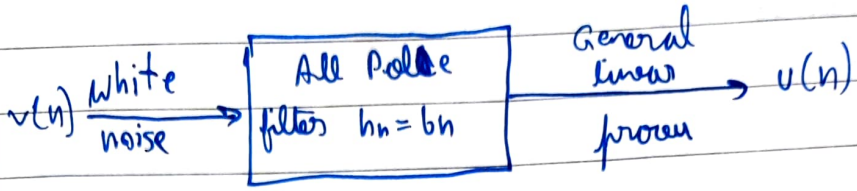
- Using $J_{min} = \sigma_d^2 - \underline{w}_f^H R\, \underline{w}_f = \sigma_d^2 - \underline{p}^H \underline{w}_f$

$$= \sigma_d^2 - p^H R p$$

we get the forward prediction error power

$$P_{f,M} = r(0) - \underline{r}^H \underline{w}_f$$

Relation between Linear Prediction and Auto-Regressive modeling



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

1. The AR coefficients $a_1, a_2, \ldots, a_M$
2. The variance σ_v^2 of the white noise $v(n)$ used as excitation.

The time series $u(n), u(n-1), \ldots, u(n-M)$ represents the realization of an auto-regressive process of order M if it satisfies

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

or

$$\sum_{k=0}^{M} a_k^* u(n-k) = v(n)$$

where $a_0 = 1$

Multiply the above equation by $u^*(n-l)$ and taking expectation

$$E\left[\sum_{k=0}^{M} a_k^* u(n-k) u^*(n-l)\right] = E[v(n) u^*(n-l)]$$

We may simplify the above equation to

$$\sum_{k=0}^{M} a_k^* r(l-k) = 0, \quad l > 0, \quad \text{where } a_0 = 1$$

Thus we see that the auto-correlation function of the AR process satisfies the difference eq^n

$$r(l) = w_1^* r(l-1) + w_2^* r(l-2) + \dots + w_M^* r(l-M)$$

where $l > 0$

and $w_k = -a_k, \quad k = 1, 2, \dots, M$

Yule-Walker Equations

Writing the above equation for $l = 1, 2, \dots, M$ we get a set of M simultaneous equations with the values $r(0), r(1), \dots, r(M)$ of the auto-correlation function of the AR process as known quantities and the AR parameters $a_1, a_2, \dots, a_M$ as the unknowns.

These set of equations can be expanded -

$$\begin{bmatrix} r(0) & r(1) & \dots & r(M-1) \\ r^*(1) & r(0) & \dots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \dots & r(0) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} = \begin{bmatrix} r^*(1) \\ r^*(2) \\ \vdots \\ r^*(M) \end{bmatrix}$$

where $w_k = -a_k$

In a compact form,

$$R\underline{w} = \underline{r}$$

Assuming that the auto-correlation matrix is non-singular,

$$\underline{w} = R^{-1} \underline{r}$$

We may uniquely determine the matrix R and the vector $\underline{r}$ given the auto-correlation sequence $r(0), \ldots, r(M)$.

- Hence, using the above equation, we can compute the AR coefficients,

$$a_k = -w_k, \qquad k = 1, 2, \ldots, M$$

For $l = 0$, we can re-write

$$E\left[\sum_{k=0}^{M} a_k^* u(n-k)\, u^*(n-l)\right] = E\left[v(n)\, u^*(n-l)\right]$$

or, $E\left[v(n) u^*(n)\right] = E\left[v(n)\, v^*(n)\right] = \sigma_v^2$

Using the fact that

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

where σ_v^2 is the variance of the zero mean white noise $v(n)$.

Setting $l = 0$ and performing complex conjugation on both sides

$$\sigma_v^2 = \sum_{k=0}^{M} a_k\, r(k) \qquad \text{with } a_0 = 1$$