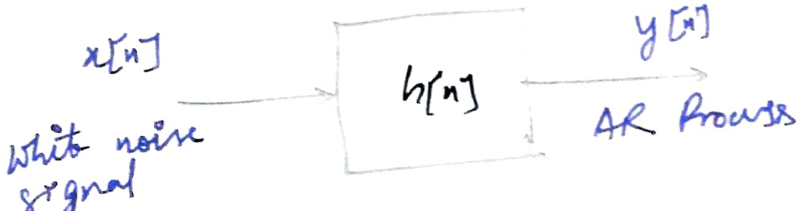
Assignment-4

Submitted By:
18EE35014

Levinson Durbin Algorithm to calculate LPC:

LD algorithm is used most commonly to estimate the all-pole AR model parameters, because the design equations used to obtain the best-fit AR model are simpler than those used for MA or ARMA modelling.



Consider a general all-pole filter : $H(z) = \dfrac{Y(z)}{X(z)}$ (order P)

$$\frac{Y(z)}{X(z)} = \frac{G}{1 - \sum_{k=1}^{P} \alpha_k^{(P)} z^{-k}}$$

applying inverse z transform, we get

$$y[n] = Gx[n] + \sum_{k=1}^{P} \alpha_k^{(P)} y[n-k]$$

Now we want to obtain a filter T.F to an arbitrary desired filter T.F $H_d(z)$. This is done by minimizing the average square error between magnitudes of frequency response of desired filter $H_d(e^{j\omega})$ and all pole filter $H(e^{j\omega})$

$$e^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) - H_d(e^{j\omega}) \right|^2 d\omega$$

applying parseval's theorem.

$$e^2 = \sum_{n=0}^{N-1} \left(h[n] - h_d[n] \right)^2$$

Since $h[n]$ is impulse response : system response to $\delta[n]$.

$$h[n] = G \cdot \delta[n] + \sum_{k=1}^{P} \alpha_k^{(P)} h[n-k]$$

$$\Rightarrow e^2 = \sum_{n=0}^{N-1} \left(G\delta[n] + \sum_{k=1}^{P} \alpha_k^{(P)} h[n-k] - h_d[n] \right)^2$$

for error to be minimum,

$$\frac{de^2}{d\alpha_k} = 0$$

$$\sum_{n=0}^{N-1} \left(G\delta[n] \cdot h[n-k] + \sum_{l=1}^{P} \alpha_l^{P} h[n-l] h[n-k] \right) = \sum_{n=0}^{N-1} h_d[n] h[n-k]$$

Since system is causal, G won't enter the solution.

$$\sum_{l=1}^{P} \alpha_l^{(P)} \cdot \frac{\left(\sum_{n=0}^{N-1} h[n-l] h[n-k] \right)}{N} = \frac{\sum_{n=0}^{N-1} h_d[n] \cdot h[n-k]}{N}$$

let $\phi_{yy}[m] = \frac{1}{N} \sum_{n=0}^{N-1} h[n] \cdot h[n-m] = \phi[m]$

$$\boxed{\sum_{l=1}^{p} \alpha_l^{(p)} \phi[k-l] = \phi[k]}$$

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

p set of linear equations

$$\boxed{R\alpha = P}$$

where $R = \begin{bmatrix} \phi[0] & \phi[1] & \ldots & \phi[p-1] \\ \phi[1] & \phi[0] & \ldots & \phi[p-2] \\ \vdots & \vdots & & \vdots \\ \phi[p-1] & \phi[p-2] & \ldots & \phi[0] \end{bmatrix}$

$\alpha = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_p \end{bmatrix}$, $P = \begin{bmatrix} \phi[1] \\ \phi[2] \\ \vdots \\ \phi[p] \end{bmatrix}$

* a direct solution is given by [complexity $O(N^3)$]

$\alpha = R^{-1} P$ but it is tough to compute inverse for the large number. So instead we find solution in a recursive method to reduce the complexity.

The basic idea of the recursion is to find the solution α_{p+1} for the $(p+1)^{th}$ order case from the solution α_p for the p^{th} order case.

$$R_{p+1} = \left[\begin{array}{c|c} R_p & \begin{matrix}\phi[p] \\ \phi[p-1] \\ \vdots \\ \phi[1]\end{matrix} \\ \hline \phi[p] \; \phi[p-1] \dots \phi[1] & \phi[0] \end{array}\right]$$

let $p = q$

$$q_{p+1} = \left[\begin{array}{c} q_p \\ \vdots \\ \hline \phi[p+1] \end{array}\right] \quad \text{and} \quad f_p = \begin{bmatrix} \phi[p] \\ \phi[p-1] \\ \vdots \\ \phi[1] \end{bmatrix}$$

$$R_{p+1} = \left[\begin{array}{c|c} R_p & f_p \\ \hline f_p^T & \phi[0] \end{array}\right]$$

$$\alpha_{p+1} = \left[\begin{array}{c} \gamma_p \\ \hline \alpha_{p+1} \end{array}\right] = \left[\begin{array}{c} \alpha_p \\ \hline 0 \end{array}\right] + \left[\begin{array}{c} \varepsilon_p \\ \hline k_{p+1} \end{array}\right]$$

where ε_p is correction term

k_{p+1} is new $\alpha_{p+1} \rightarrow$ reflection coefficients

$\therefore \quad R_{p+1}\alpha_{p+1} = q_{p+1}$

$$\begin{bmatrix} R_p & \rho_p \\ \rho_p^T & \phi[0] \end{bmatrix} \left(\begin{bmatrix} \alpha_p \\ 0 \end{bmatrix} + \begin{bmatrix} \epsilon_p \\ k_{p+1} \end{bmatrix} \right) = \begin{bmatrix} q_p \\ \phi[p+1] \end{bmatrix}$$

$$R_p\alpha_p + R_p\epsilon_p + \rho_p k_{p+1} = q_p$$

and

$$\rho_p^T\alpha_p + \rho_p^T\epsilon_p + \phi[0]\, k_{p+1} = \phi[p+1]$$

On simplifying and approximately,

$$k_{p+1} \doteq \frac{-\phi[p+1] - \rho_p^T\alpha_p}{E_p}$$

where $E_p = (1-k_p)^2 E_{p-1}$

$E_0 = \phi[0]$

$\therefore$ Recursion algorithm is as follows:

$$E_0 = \phi[0]$$

$$k_i = \frac{-\phi[i] - \sum_{j=1}^{i-1} \alpha_j^{(i-1)}\phi[i-j]}{E_{i-1}} \quad \text{for } 1 \le i \le p$$

$$\alpha_i^{(i)} = k_i$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} + k_i\alpha_{i-j}^{(i-1)} \quad \text{for } j = 1, 2, \ldots, i-1$$

$$E_i = (1-k_i^2)E_{i-1}$$

after P steps, we arrive at pth order estimate

here

$$e_p = \left[\prod_{j=1}^{p} (1-k_j^2)\right] E_0 = \left[\prod_{j=1}^{p} (1-k_j^2)\right] \phi[0]$$

gives the estimate of variance of $x[n]$

$$\begin{array}{ccc} & \xrightarrow{\text{Levinson}} & \\ \text{Reflection coefficients} & \xleftrightarrow{\text{interconversion}} & \text{AR Parameters} \\ \phi[0], k_1, k_2 \dots k_p & & a[1], a[2], \dots a[p], \sigma^2 \\ & \xleftarrow{\text{step down}} & \end{array}$$

Note:

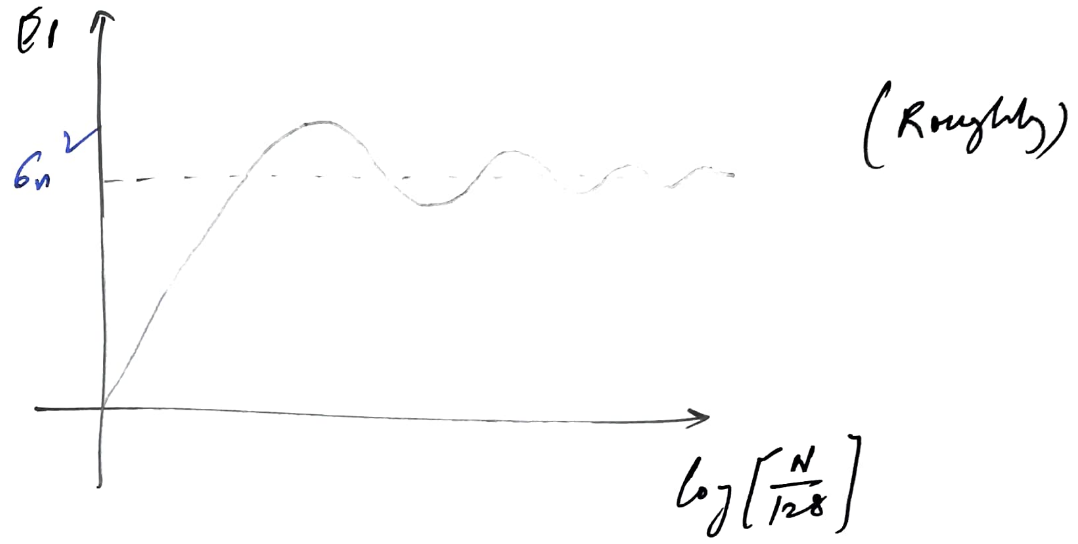
→ for all pole AR process generator to be stable, the poles must all lie inside unit circle in the z-plane.

→ Time complexity of LD algorithm is $O(N^2)$

Space " " " " is $O(N)$.

Results

→ for lower length, the estimation is too weak; h
as length increases the estimation is becoming
more perfect.



Comparision of LD with other algorithms:

→ LD v/s Cholesky decomposition:

* The cholesky decomposition is a method used to find inverse of matrix which has Hermitian Symmetry.

Computational Complexity	, Space
LD – $O(N^2)$	$O(N)$
Cholesky – $O(N^3)$	$O(N^2)$

* LD, exploits the fact that LPC analysis has Toeplitz Symmetry.

→ LD v/s LMS algorithm

* LMS algo is an adaptive filter techniques, But it does not gurantee minimum phase systems & stability while LD does. Although LMS is more elegant and accurate method of prediction, It is considerably slower than LD algorithm.