

Statistical Signal Processing
Prof. Prabin Kumar Bora
Department of Electronics and Electrical Engineering
Indian Institute of Technology – Guwahati

Lecture – 33
Recursive Least Squares (RLS) Adaptive Filter 2

(Refer Slide Time: 00:41)

Let us recall

- ❖ The LS estimation problem for signal estimation is

$$\text{Minimize } \varepsilon(n) = \sum_{k=0}^n \lambda^{n-k} (d(k) - \mathbf{y}'(k) \mathbf{h}(n))^2 \text{ with respect to } \mathbf{h}(n)$$

which gives the **normal equation** for the LS estimation

$$\left(\sum_{k=0}^n \lambda^{n-k} \mathbf{y}(k) \mathbf{y}'(k) \right) \mathbf{h}(n) = \sum_{k=0}^n \lambda^{n-k} d(k) \mathbf{y}(k).$$

- ❖ The normal equation is written as

$$\hat{\mathbf{R}}_{\mathbf{y}}(n) \mathbf{h}(n) = \hat{\mathbf{r}}_{\mathbf{y}}(n)$$

- ❖ The solution to the normal equation is given by

$$\mathbf{h}(n) = \hat{\mathbf{R}}_{\mathbf{y}}^{-1}(n) \hat{\mathbf{r}}_{\mathbf{y}}(n)$$

The RLS algorithm finds the above inverse recursively.

Hello students. Welcome to this lecture on Recursive Least Squares, Adaptive Filter. Let us recall the. The LS estimation problem for signal estimation by adaptive filter is, this is the problem, minimize the weighted sum square error given by summation lambda to the power n - k into dk - y transpose k into h n whole square, k going from 0 to n with respect to the filter parameter h n. So we have to minimize this weighted sum square error with respect to the filter parameters. And here dk is the desired signal and yk is the input signal.

The minimization gives the normal equation for the LS estimation and this is given by this equation, summation lambda to the power n - k y vector into yk transpose, k going from 0 to n into h n vector is equal to summation lambda to the power n - k into dk into yk vector k going from 0 to n. And this we write in usual notation that is R y hat n for this into h n rd y hat for this expression so is equal to rd y hat n.

This solution, this is the normal equation and we can solve it by matrix inversion. $\hat{\mathbf{h}}_n$ is given by $\hat{\mathbf{r}}_n^{-1} \mathbf{r}_n$, so this is the inverse of the estimated autocorrelation function and this is the estimated cross correlation function. The RLS algorithm finds the above inverse this inverse recursively.

(Refer Slide Time: 02:51)

Let us recall...

RLS algorithm

$\mathbf{P}(-1) = \frac{1}{\delta} \mathbf{I}_{MM}$, δ a small positive number

$\mathbf{h}(-1) = \mathbf{0}$ and choose λ

For $n = 1, 2, \dots$ do

1. Get $d(n), \mathbf{y}(n)$
2. Compute the filter output $\hat{d}(n) = \mathbf{h}'(n-1)\mathbf{y}(n)$
3. Get $e(n) = d(n) - \hat{d}(n)$
4. Calculate gain vector $\mathbf{k}(n) = \frac{\mathbf{P}(n-1)\mathbf{y}(n)}{\lambda + \mathbf{y}'(n)\mathbf{P}(n-1)\mathbf{y}(n)}$
5. Update the filter parameters
 $\mathbf{h}(n) = \mathbf{h}(n-1) + \mathbf{k}(n)e(n)$
6. Update the inverse of the autocorrelation (\mathbf{P}) matrix
 $\mathbf{P}(n) = \frac{1}{\lambda} (\mathbf{P}(n-1) - \mathbf{k}(n)\mathbf{y}'(n)\mathbf{P}(n-1))$

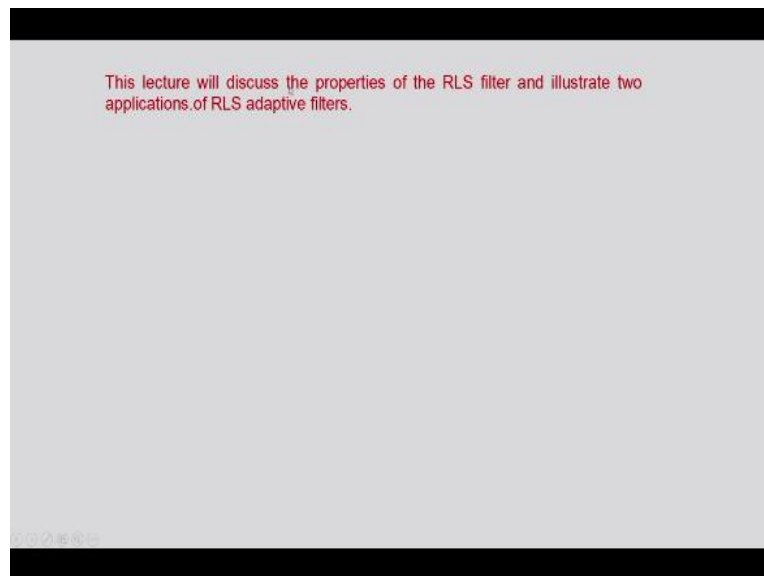
Let us recall the RLS algorithm. It is initialized this P matrix $\mathbf{P}(-1) = 1/\delta$ into identity matrix where delta is a small positive number and filter coefficients are initialize 0, we have to choose a proper value of lambda around 195.98 .99 like that. Now for $n = 1$ or 0 we will carry out these computations; first get d_n , \mathbf{y}_n get d_n and \mathbf{y}_n . This is the desired signal and this is the input vector.

Compute the filter output \hat{d}_n is equal to that is previous filter estimate into \mathbf{y}_n vector. Then I will compute e_n is equal to $d_n - \hat{d}_n$ and \hat{d}_n we have computed d_n is here, so we have to find out the error. Next step is to calculate the gain vector \mathbf{k}_n , \mathbf{k}_n is given by $\mathbf{P}(n-1)$ into \mathbf{y}_n that is the current data, P matrix into current data divided by $\lambda + \mathbf{y}_n^T \mathbf{P}(n-1) \mathbf{y}_n$. So this is the current data, current data and past estimate of the P matrix.

Then we will update the filter parameters by this relationship \mathbf{h}_n is equal to $\mathbf{h}_{n-1} + \mathbf{k}_n e_n$, so \mathbf{k}_n is the gain vector. Once we have calculated the filter coefficient we have to update the P

matrix and this is given by $P_n = (1 - \lambda) P_{n-1} + y_n y_n^T$. So this is the update equation for the inverse of the autocorrelation matrix.

(Refer Slide Time: 05:16)



This lecture will discuss the properties of the RLS filter and illustrate two applications of RLS adaptive filters.

(Refer Slide Time: 05:26)

Properties of the RLS filters

Relation with Wiener filter

We have the optimality condition for the RLS filters

$$\hat{\mathbf{R}}_y(n) \mathbf{h}(n) = \hat{\mathbf{r}}_{dy}(n)$$

where $\hat{\mathbf{R}}_y(n) = \sum_{k=0}^n \lambda^{n-k} \mathbf{y}(k) \mathbf{y}'(k)$ and

$$\hat{\mathbf{r}}_{dy}(n) = \sum_{k=0}^n \lambda^{n-k} d(k) \mathbf{y}(k)$$

Under stationary condition, we take $\lambda = 1$. Dividing by $n+1$, we get

$$\frac{\hat{\mathbf{R}}_y(n)}{n+1} = \frac{\sum_{k=0}^n \mathbf{y}(k) \mathbf{y}'(k)}{n+1}$$

If we consider the elements of $\frac{\hat{\mathbf{R}}_y(n)}{n+1}$, we see that each is an estimator for the autocorrelation of specific lag.

$$\frac{1}{n+1} \sum_{k=0}^n y(k) y'(k) = \hat{r}_{yy}^l$$

Let us first see the relation of the RLS adaptive filter with Wiener filter. We have the optimality condition for RLS filters given by the normal equation that is $\hat{\mathbf{R}}_y(n)$ matrix multiplied by $\mathbf{h}(n)$ vector is equal to cross correlation vector that is $\hat{\mathbf{r}}_{dy}(n)$. So where $\hat{\mathbf{R}}_y(n)$ is given by this

relationship with a sum of y_k into $y_k^T y_k$ transpose is a matrix, so this matrix for different instant k are summed up.

Similarly, $\hat{r}_{dy}(n)$ is equal to summation λ to the power $n - k$ into d_k into y_k , k going from 0 to n . Here d_k into y_k is a vector and this vector summed up for different values of k and we get this estimate. Under stationary condition, we take λ is equal to 1 dividing by $n + 1$ this expression if we divide this expression by $n + 1$ we get $R \hat{y}(n)$ divided by $n + 1$ is equal to summation y_k into y_k^T k going from 0 to n divided by $n + 1$ so this is the expression for $R \hat{y}(n)$ divided by $n + 1$.

Now if we consider the elements of $R \hat{y}(n)$ by $n + 1$ we see that it is an estimator for the autocorrelation of specific lag. So for example the first element of this summation will be y_k into y_k , this will be the first element, summation k going from 0 to n and then divided by $n + 1$. So this will be the estimate for the variance. So this will be σ_y^2 , so that way all the elements of this $R \hat{y}(n)$ matrix divided by $n + 1$ will be the corresponding autocorrelation function at specific lag. So this is important observation.

(Refer Slide Time: 08:20)

Relation with Wiener filter

- Under stationarity assumption,

$$\lim_{n \rightarrow \infty} \frac{\hat{R}_y(n)}{n+1} = R_y$$

- Because, the sample autocorrelation function is a consistent estimator of the true autocorrelation function.
- Similarly

$$\lim_{n \rightarrow \infty} \frac{\hat{r}_{dy}(n)}{n+1} = r_{dy}$$

- Hence as $n \rightarrow \infty$, optimality condition can be written as

$$R_y h = r_{dy}$$

Thus, if the data is stationary, the algorithm will converge to the Wiener solution

Under stationarity assumption this matrix divided $n + 1$ as limit $R \hat{y}(n)$ divided by $n + 1$ as n tends to infinity will be R_y . This is because the sample autocorrelation function is a consistent estimator of the true autocorrelation function under stationarity assumption. Similarly limit of r

that n divided by $n + 1$ as n tends to infinity will be the cross correlation vector r . Hence as n tends to infinity optimality condition can be written as R_y matrix into $h = r$ vector. That is, this is the Wiener equation for the Wiener filter. Thus if the data is stationary the algorithm will converge to the Wiener solution. So this is important of the result.

(Refer Slide Time: 09:25)

Dependence on the initial values of P matrix

Consider the recursive relation

$$\hat{R}_y(n) = \lambda \hat{R}_y(n-1) + y(n)y'(n)$$

Corresponding to

$$\hat{R}_y^{-1}(-1) = \delta I \quad P(-1) = \delta I$$

we have $\hat{R}_y(-1) = \frac{1}{\delta}$

With this initial condition the matrix difference equation has the solution:

$$\begin{aligned} \tilde{R}_y(n) &= \lambda^{n+1} \hat{R}_y(-1) + \sum_{k=0}^n \lambda^{n-k} y(k)y'(k) \\ &= \lambda^{n+1} \hat{R}_y(-1) + \hat{R}_y(n) \\ &= \lambda^{n+1} \frac{1}{\delta} + \hat{R}_y(n) \end{aligned}$$

Next we will see dependence on the initial values of P matrix. Because we initialize P matrix here P of -1 is equal to 1 by delta into identity matrix where delta is a small positive number. Consider the recursive relation that is R_y hat n is equal to lambda times R_y hat $n - 1$ + y_n into y_n transpose corresponding to the initialization R_y hat inverse, - 1 is equal to delta I that is the; that is P of + - 1 is equal to delta I. This is P of - 1 is equal to delta into identity matrix.

So corresponding to R_y hat inverse at point -1 is equal to delta I we have, so inverse is delta I, so R_y hat at point - 1 will be I by delta. With this initial condition the matrix difference equation that is the difference equation corresponding to the recursive estimation of autocorrelation function. So the matrix differential equation has this solution like this; \tilde{R} tilde n because we are assuming this initial condition is equal to lambda to the power $n + 1$ into R_y hat - 1.

So this is the initial condition plus summation lambda to the power $n - k$ into y_k into y_k transpose k going from 0 to n , so these we can write like lambda to the power $n + 1$ into R_y hat - 1 plus this is by definition R_y hat n . So this will give us lambda to the power $n + 1$ into I by

delta into R y hat n. So because of the; this initialization we have an additional factor like this. Now our autocorrelation at different instant will be given by this expression lambda to the power n + 1 into I / delta plus actual estimation R y hat n.

(Refer Slide Time: 12:19)

Dependence....

Hence the optimality condition is modified as

$$\left(\lambda^{n+1} \frac{I}{\delta} + \hat{\mathbf{R}}_r(n)\right) \tilde{\mathbf{h}}(n) = \hat{\mathbf{r}}_{dr}(n)$$

where $\tilde{\mathbf{h}}(n)$ is the modified solution due to assumed initial value of the P-matrix.

$$\frac{\lambda^{n+1} \hat{\mathbf{R}}_r^{-1}(n) \tilde{\mathbf{h}}(n)}{\delta} + \tilde{\mathbf{h}}(n) = \mathbf{h}(n)$$

pre-multiplying by $\hat{\mathbf{R}}_r^{-1}(n)$

❖ If we take λ as less than 1, then the bias term in the left-hand side of the above equation will be eventually die down and we will get

$$\tilde{\mathbf{h}}(n) = \mathbf{h}(n)$$

We note

❖ Unlike the LMS filter, the convergence is less sensitive to eigen value spread. This is a remarkable feature of the RLS algorithm.

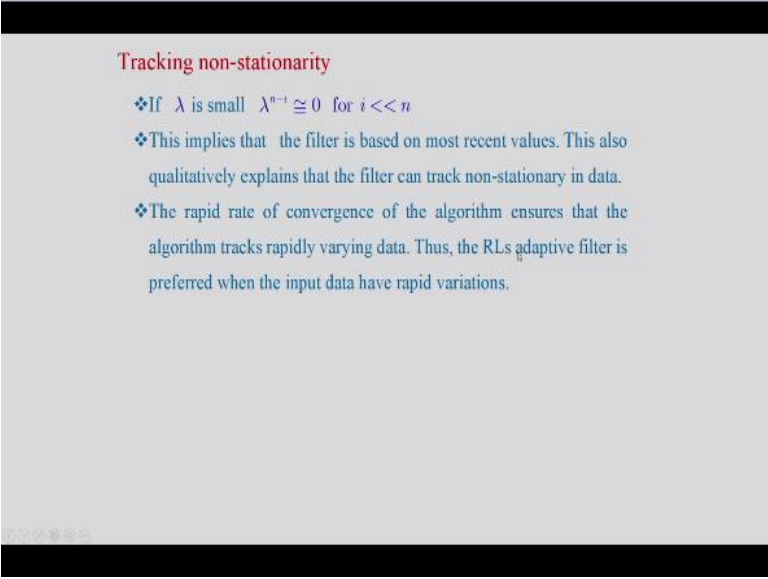
Hence the optimality condition of normal equation is modified to this; that is lambda to the power n + 1 into I / by delta + R y hat n into h tilde n must be equal to R dy hat n. So this is the modification in the left hand side because of the initialization of the P matrix as delta into I. So this is the modified normal equation where h tilde n is the modified solution due to the assumed initial value of the P matrix.

Now taking the inverse, so this we get pre-multiplying R y inverse n. So we are pre-multiplying. Here, so R y hat inverse n; here we are pre-multiplying this will be now identity so h tilde n will be there and here pre-multiplied by R y hat inverse then we will get h n. So that way this relationship we get. Now if we take lambda less than 1, lambda is less than 1 then the bias term this is the bias term in the left hand side of the above equation will be eventually die down as n tends to infinity.

This term will become 0 because lambda to the power n+1 will become 0 as n tends to infinity. And we will get h tilde n is equal to h n. So therefore, because of this initialization we have taken that is R y hat inverse n - 1 is equal to delta I. So this will not affect the solution. So ultimately,

we will get $\tilde{h}_n = h_n$. Unlike the LMS filter, the convergence is less sensitive to Eigen value spread; we will not; one more thing that unlike the LMS filter the convergence is less sensitive to Eigen value spread, this can be soon but we are; we take this as a result. This is a remarkable feature of the RLS algorithm. So we also note this, we note. Unlike the LMS filter the convergence is less sensitive to Eigen value spread. And this is one of the remarkable property of RLS algorithm.

(Refer Slide Time: 15:30)



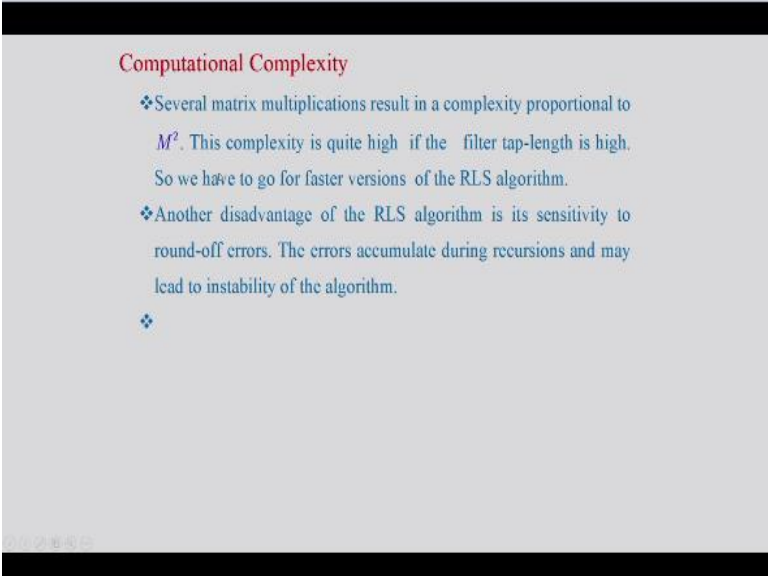
Tracking non-stationarity

- ❖ If λ is small $\lambda^{n-i} \approx 0$ for $i \ll n$
- ❖ This implies that the filter is based on most recent values. This also qualitatively explains that the filter can track non-stationary in data.
- ❖ The rapid rate of convergence of the algorithm ensures that the algorithm tracks rapidly varying data. Thus, the RLS adaptive filter is preferred when the input data have rapid variations.

Next we will consider taking up non-stationarity. If λ is small λ to the power $n - i$ will be approximately equal to 0 for i must less than n , suppose for smaller value of i compared to n λ^{n-i} will be approximately 0. This implies that the filter is based on the most recent values only; because we are weighing every error by λ^{n-i} therefore the filter will be based on the most recent values.

This also qualitatively explains that the filter can track non-stationarity in data. So this is one important factor that because of this forgetting factor RLS filter can track non-stationarity in data. The rapid rate of convergence of the algorithm ensures that the algorithm tracks rapidly varying data. This is another observation; the rapid rate of convergence of the algorithm ensures that the algorithm tracks rapidly varying data. Thus, the RLS adaptive filter is preferred when the input data have rapid variations. So under this situation where input data have rapid variations RLS adaptive filters are prepared.

(Refer Slide Time: 17:07)



Computational Complexity

- ❖ Several matrix multiplications result in a complexity proportional to M^2 . This complexity is quite high if the filter tap-length is high. So we have to go for faster versions of the RLS algorithm.
- ❖ Another disadvantage of the RLS algorithm is its sensitivity to round-off errors. The errors accumulate during recursions and may lead to instability of the algorithm.

❖

Next let us see the computational complexity of RLS filter. Several matrix multiplications result in a complexity proportional to M squared. This is the complexity; it is proportional to M squared because of different matrix multiplications involved in the algorithm. This complexity is quite high if the filter tap-length is high because it is M square so this complexity is high. So we have to go for faster versions of RLS algorithms. There are several faster RLS algorithms like the latest RLS adaptive filter.

Another disadvantage of the RLS algorithm is its sensitivity to round-off errors. So because of the recursive nature of estimation the errors accumulate during recursions and may lead to instability of the algorithm. So this is one drawback and another drawback is the computational complexity.

(Refer Slide Time: 18:19)

Example 1

Adaptive linear prediction – Consider the AR(2) signal

$$Y(n) = 1.72Y(n-1) - 0.81Y(n-2) + W(n)$$

where $W(n)$ is a zero-mean unity variance white noise. The linear predictor for the signal is given by

$$\hat{Y}(n) = h_1Y(n-1) - h_2Y(n-2)$$

The optimal linear predictor is shown to be given by

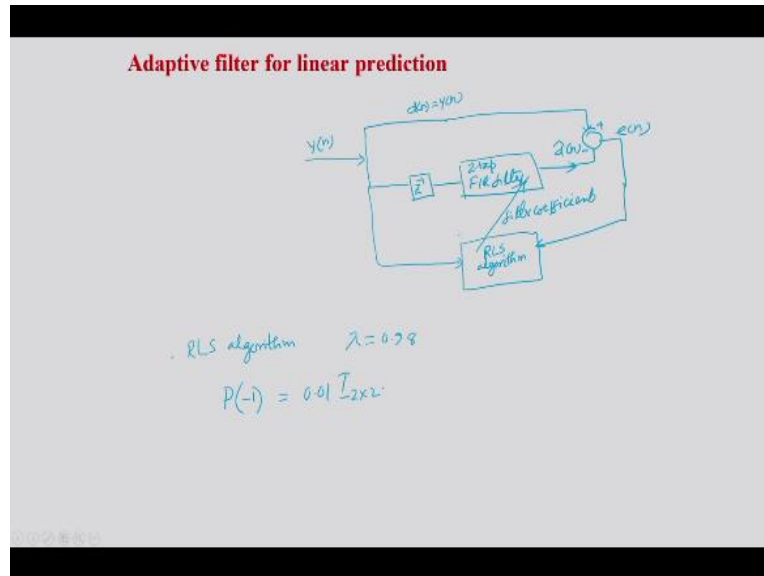
$$\hat{Y}(n) = 1.72Y(n-1) - 0.81Y(n-2)$$

The LP coefficients can be adaptively computed.

Thus, we have seen that RLS adaptive filter has a better convergence characteristics compared to LMS filters but it has higher computational complexity. Let us consider one example. Adaptive linear prediction; consider the AR 2 signal decision AR 2 signal; $Y_n = 1.72Y_{n-1} - 0.81Y_{n-2} + W_n$ where W_n is a zero-mean unity variance white noise process. This is a auto regressive process of order 2 and the second-order linear predictor for this signal is given by this; $\hat{Y}_n = h_1Y_{n-1} - h_2Y_{n-2}$.

So this is the linear predictor. And under optimality condition we can show that this linear predictor is equal to same as this AR process terms, so that way $\hat{Y}_n = 1.71Y_{n-1} - 0.81Y_{n-2}$. Here the LP coefficients are to be computed adaptively.

(Refer Slide Time: 19:52)

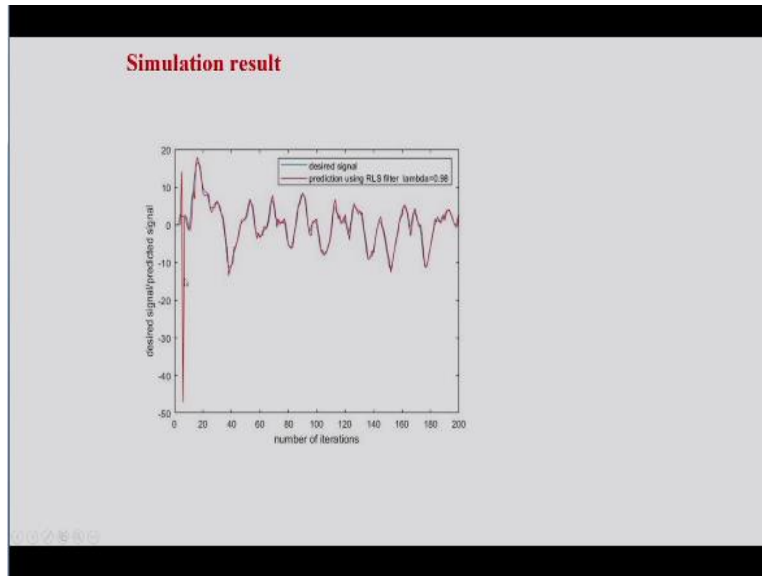


And now for adaptive filtering this is y_n . Now for linear prediction we have to consider the delayed version of this signal, suppose this is delay and this is the delayed version of the signal. And this delayed version of the signal is passed through the 2-tap FIR filter for linear prediction. So we will get suppose this is \hat{d}_n . Now this y_n is the desired signal here, so this is the desired signal $d_n = y_n$.

And this desired signal will be compared with the \hat{d}_n and the error e_n will be computed. Now we have the adaptive filter algorithm that is RLS algorithm here, RLS algorithm. So this error will be passed here and this will be passed to here also and the filter coefficients will be updated and the filter coefficients; these are the filter coefficients. So this filter coefficients will be used to filter this signal delayed version of the signal to get \hat{d}_n .

So this is the implementation of the second-order linear prediction. This is a 2-tap filter and this is the delayed version so that it will compute \hat{d}_n is equal to some h_1 times $y_{n-1} + h_2$ times y_{n-2} . So we use RLS algorithm and we have to initialize λ , λ we use 0.98. And similarly the P matrix is initialize suppose P of -1 is equal to δ , δ we can take about 0.01 into identity matrix. This is a $2/2$ identity matrix.

(Refer Slide Time: 22:43)



So with this if we run the RLS algorithm then the solution will look like this. So this is the desired signal is this blue is the desired signal and red is the prediction using RLS filter of lambda is equal to 0.98. And we see that initially there is mismatch but as n increases the predicted value and the desired value of the signal are very close, they are almost matching.

(Refer Slide Time: 23:23)

Example System identification:

You are given a 6-length FIR system with unknown parameters. Identify them.

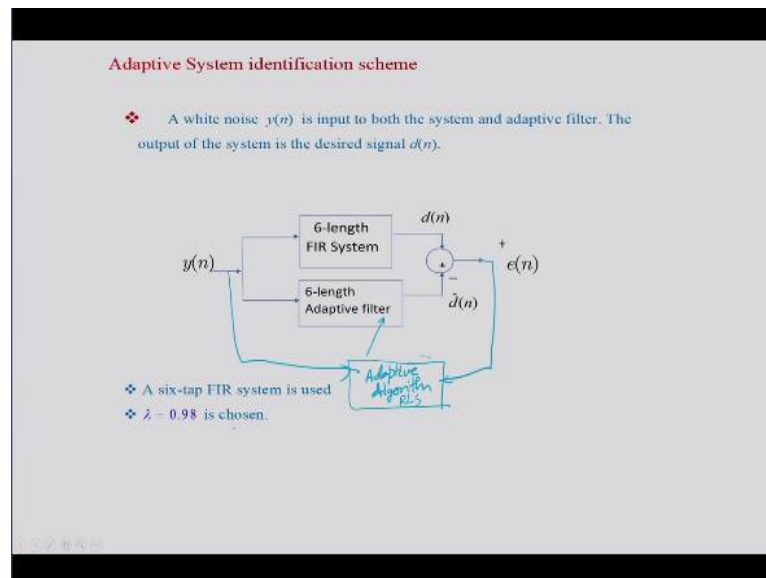
Suppose the system is given by

$$H(z) = -0.0078 + 0.064z^{-1} + 0.4433z^{-2} + 0.4433z^{-3} + 0.064z^{-4} - 0.0078z^{-5}$$

We will consider second example, example 2; this is the example 2. System identification, you are given a 6 length FIR system with unknown parameters. We know it is FIR system but parameters are unknown, identify them; we have to identify them. Suppose this system is given by this, we are considering system, FIR system $H(z)$ is given $-0.00078 + 0.064$ into z to the power $-1 + 0.4433$ into z to the power $-2 + 0.4433$ into z to the power $-3 + 0.064$ into z to the

power - 4 - 0.0078 into z to the power -5. This is the transfer function of the unknown system which we want to identify.

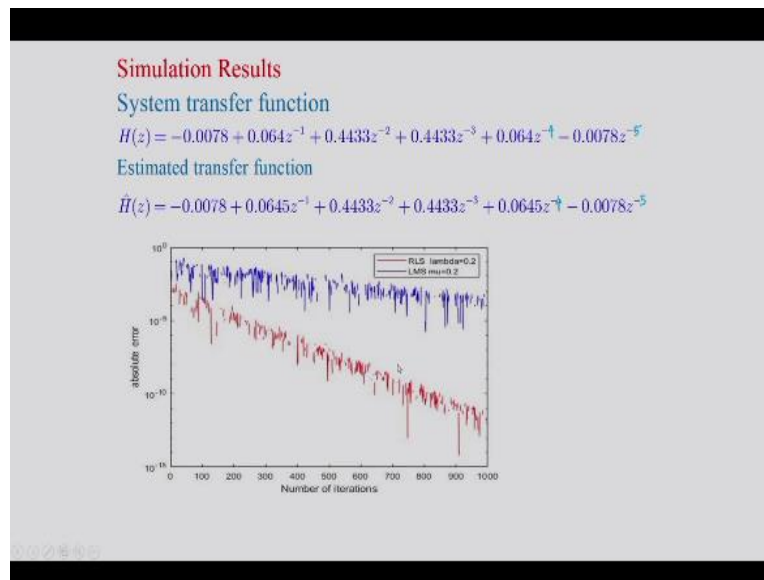
(Refer Slide Time: 24:33)



So this adaptive system is like this. This is the; we have to send a broadband signal usually a white noise signal $y(n)$ is input to both the system and any adaptive filter. Here we consider a 6-length adaptive filter. This is the 6 length FIR system and if white noise $y(n)$ is input to both. Now the output of the FIR system is $d(n)$ this is the desired signal and we want to mimic the system by this adaptive filter and this adaptive filter output is $\hat{d}(n)$.

And this $d(n)$ and $\hat{d}(n)$ are compared and the $e(n)$ will be passing through the adaptive algorithm. Suppose this is the adaptive algorithm, so this error is sent here and similarly $y(n)$ will be sent here so the adaptive filter will be operating and then this error; this filter coefficient whatever are estimated that is paid here. So these filter coefficients are used to filter this signal and if there is error this error is again used to update the filter parameters. So this is the system identification, system we are using here this 6 length adaptive filter will model 6 length FIR system. Again we apply the RLS algorithm here, RLS and we take λ is equal to 0.98.

(Refer Slide Time: 26:45)



So these are the simulation result or original system is like this. This is the transfer function of the original system and the estimated transfer function using the RLS algorithm we get this $H(z) = -0.0078 + 0.064z^{-1} + 0.4433z^{-2} + 0.4433z^{-3} + 0.064z^{-4} - 0.0078z^{-5}$ this is same here also; this is a symmetric filter because of the linear phase so here also -0.0078 this is same, similarly this is 0.064 here 0.0645 , only difference in the four decimal place.

Similarly, these are equal, this also equal so that way at this RLS filter identify the given system. We also examine the convergence characteristics. This is the error, absolute error after each iteration and we see that in the case of LMS algorithm with $\mu = 0.2$ this blue values are the error values and these are the error values when RLS with λ is equal to 0.2 is used. So this is RLS so we see that this error is gradually decreasing at a very fast rate. So this is one important observation in the case of RLS filter.

(Refer Slide Time: 28:26)

Summary

- ❖ The RLS adaptive filter has several attractive properties.
- ❖ Under stationarity of the data the RLS solution converges to the Wiener solution
- ❖ Unlike the LMS filter, the convergence is less sensitive to eigen value spread.
This is a remarkable feature of the RLS algorithm
- ❖ Unlike the LMS algorithm, RLS algorithm is computationally complex. Several matrix multiplications result in a complexity proportional to M^2 .

• May also suffer instability because of the accumulation of round-off error during recursion.

Kalman Filter:

Let us summarize the lecture; the RLS adaptive filter has several attractive properties. Under stationarity of the data the RLS solution converges to the Wiener solution, this is very important. Unlike the LMS filter, the convergence is less sensitive to Eigen value spread. This is a remarkable feature of the RLS algorithm. Unlike the LMS algorithm, RLS algorithm is computationally complex; this is the disadvantage of the RLS filters.

Several matrix multiplications result in a complexity proportional to M square. So this algorithm may also suffer instability because of the accumulation of round-off error during recursion. In the next lecture we will discuss one more important filtering technique that is known as the Kalman Filter. Thank you.