Time Varying Communication Channel Estimation Using Kalman Filters

Submitted by: Vijay Surya Vempati(16EC01012)

Abstract

In this report, a time varying channel estimation problem was realized by using Kalman Filters. In the first part of the study, the introduction and some definitions were given. In the second part, the problem was analysed and some useful theoretical and practical information were given. In the third part of the study, the method Kalman Filters were explained and the simulation algorithm was given. In the last part of the study the simulation results were given and these results were explained and commented.

1.Introduction

Channel: in general terms can be referred as everything that is present between the transmitter and the receiver. They are the medium of information transmission. The information transmitted is generally affected by the channel properties.

Channel Model: Is a mathematical representation of the transfer characteristics of the physical medium. Channel models are formulated by observing the characteristics of the received signal. The one that best explains the received signal behaviour is used to model the channel.

Channel Estimation: The process of characterizing the effect of the physical medium on the input sequence.

Why Channel Estimation?

Allows to receiver to approximate effect of the channel on the signal.

The channel estimate is essential for removing inter symbol interference, noise rejection techniques etc.

Also used in diversity combining, ML detection, angle of arrival estimation etc.

Many transmission channels can be characterised as being linear but not time invariant. Examples of these are fading dispersive channel or fading multipath channels. Here in we deal with a similar problem formulated in the next section.

2.Defining channel model

If we sample the output of the channel, then it can be shown that a good model is the low-pass tapped delay line model.

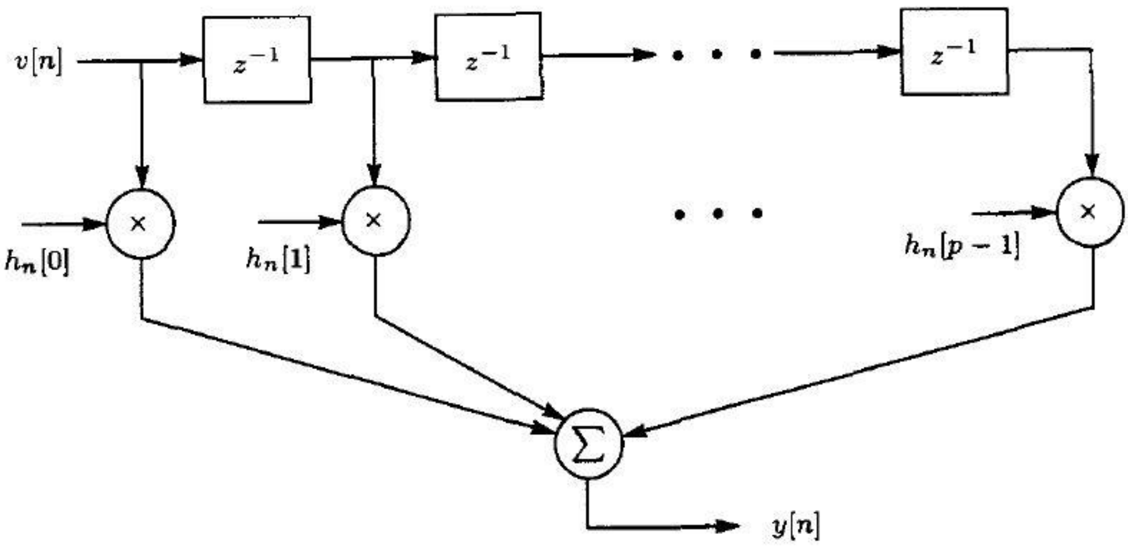

Figure 1:Tapped delay line channel model

The input-output relation is as follows

$$y[n] = \sum_{k=0}^{p-1} h_n[k]v[n-k]$$

This is really nothing more than an FIR filter with time – varying coefficients. To design effective communication or sonar systems it is necessary to have knowledge of these coefficients. Estimating these will be our primary focus.

3.Problem Statement

The problem can be stated as to estimate the time varying coefficients $h_n[k]$ based on the noise corrupted output of the channel.

$$x[n] = \sum_{k=0}^{p-1} h_n[k]v[n-k] + w[n]$$

Where $w[n]$ is termed as observation noise.

We fail to apply system identification linear model to determine the parameters because the channel is not time invariant.

A way out to realize this problem is the weights won't change abruptly from sample to sample but rather slowly. Statistically, the slow variation can be interpreted as high correlation between samples of the same tap weights. This observation naturally leads us to model the tap weights as random variables whose time variation is described by a Gauss – Markov model.

$$\mathbf{h}[n] = \mathbf{A}\mathbf{h}[n-1] + \mathbf{u}[n]$$

Where $\mathbf{h}[n] = \left[h_n[0]\ h_n[1] \ldots h_n[p-1]\right]'$,**A** is a p x p matrix and $\mathbf{u}[n]$ is vector WGN with covariance matrix **Q.** To simplify the modelling the uncorrelated scattering can be used. It assumes that the tap weights are uncorrelated with each other and hence independent due to the jointly Gaussian assumption. As a result, we can assume **A**, **Q** and $\boldsymbol{C}_h$ the covariance matrix of $\mathbf{h}[-1]$, be diagonal matrices.

The vector Gauss – Markov model then becomes p independent scalar models. The measurement model becomes,

$$x[n] = \left[v[n]\ v[n-1] \ldots v[n-(p-1)]\right]\mathbf{h}[n] + w[n]$$

where $w[n]$ is assumed to be WGN with variance σ^2 and the $v[n]$ sequence is assumed to known (since it is provided the input to the channel).

4.Kalman Filter Equations

We can now form the MMSE estimator for the tapped delay line weights recursively in time using the Kalman Filter equations for a vector state and scalar observations.

1)Prediction equation - $\hat{\mathbf{h}}[n|n-1] = \mathbf{A}\hat{\mathbf{h}}[n-1|n-1]$

2) Minimum Prediction MSE - $\mathbf{M}[n|n-1] = A\mathbf{M}[n-1|n-1]A^T + Q$

3)Kalman Gain - $\mathbf{K}[n] = \dfrac{\mathbf{M}[n|n-1]\mathbf{v}[n]}{\sigma^2 + \mathbf{v}[n]^T\mathbf{M}[n|n-1]\mathbf{v}[n]}$

4)Correction - $\hat{\mathbf{h}}[n|n] = \hat{\mathbf{h}}[n|n-1] + \mathbf{K}[n](x[n] - \mathbf{v}[n]^T\hat{\mathbf{h}}[n|n-1])$

5)Minimum MSE - $\mathbf{M}[n|n] = (\mathbf{I} - \mathbf{K}[n]\mathbf{v}[n]^T)\mathbf{M}[n|n-1]$

5.Simulations

Kalman filter is initialized as $\hat{\mathbf{h}}[-1|-1] = \boldsymbol{\mu_h} = \mathbf{0}$ & $\mathbf{M}[-1|-1] = \boldsymbol{C}_h = 100\mathbf{I}$.

We implemented the Kalman Filter estimator for a tapped delay line having $p = 2$ weights. We assume a state model with

$A = \begin{bmatrix} 0.99 & 0 \\ 0 & 0.999 \end{bmatrix}$ and $Q = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$.

The two weights of $\mathbf{h}[n]$ with time are as follows from the graph.

Simulation results-

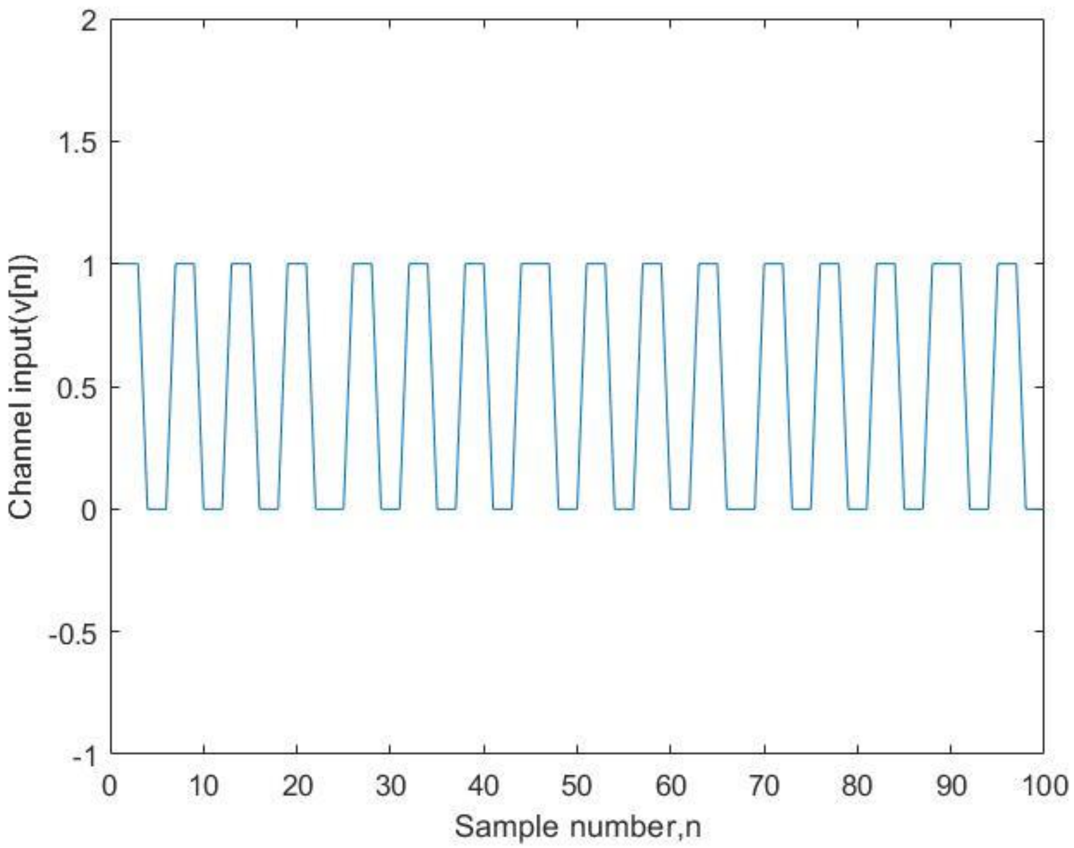

Fig 2: Channel Input

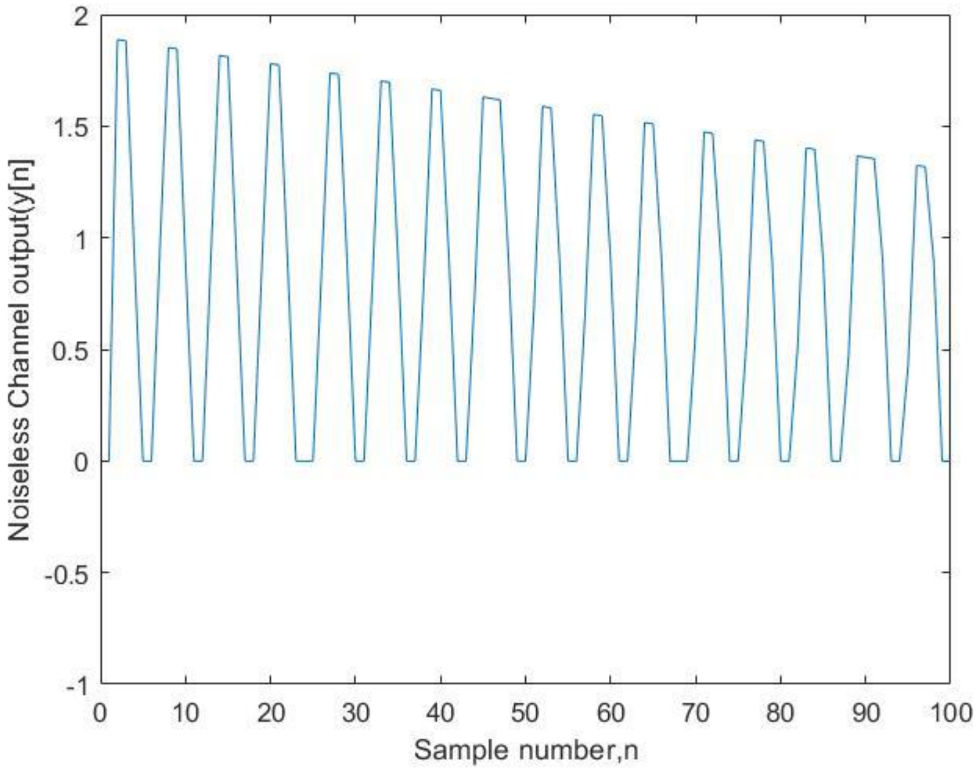

Fig 3: Noiseless channel output

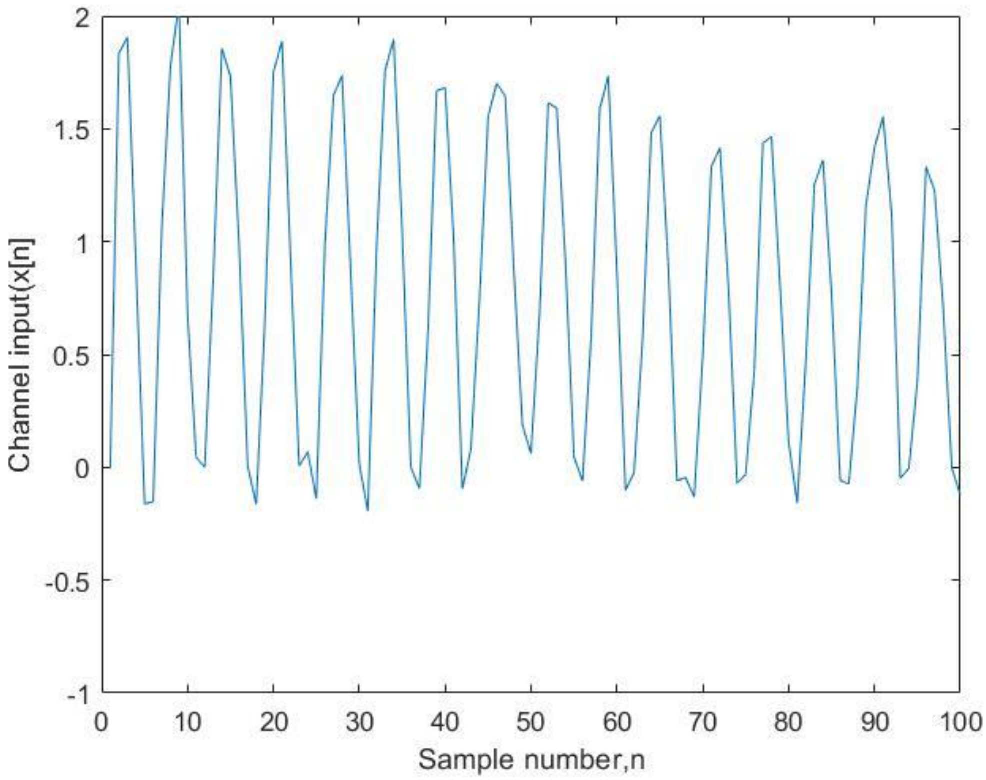

Fig 4: Channel output

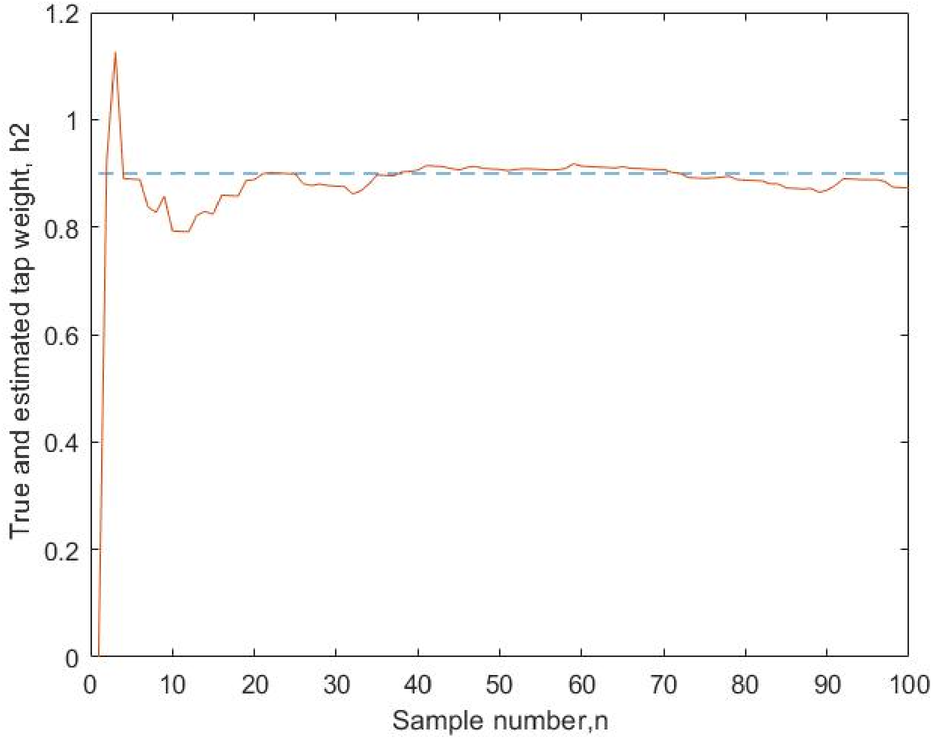

Fig 5: Kalman filter estimate of h_2

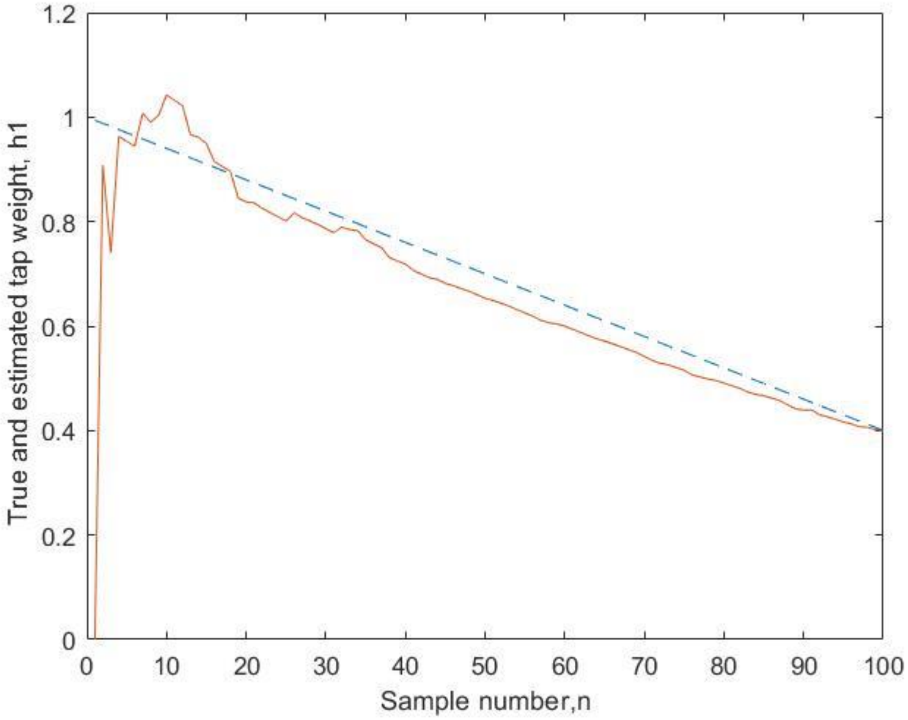

Fig 6: Kalman filter estimate of h_1

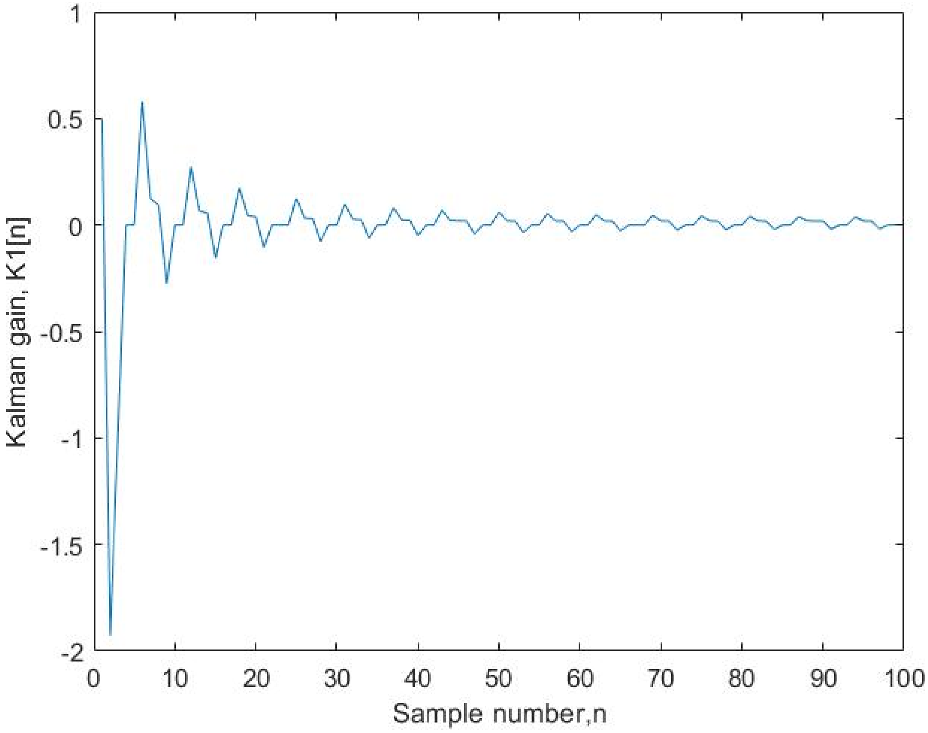

Fig 7: Kalman Gain K_{11}

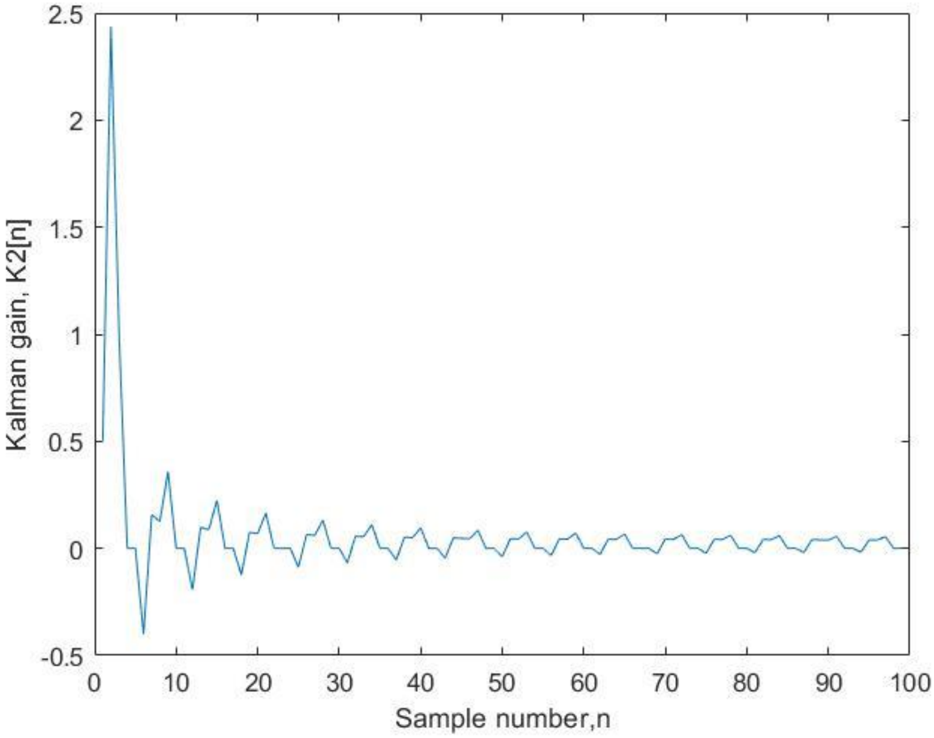

Fig 8: Kalman Gain K_{22}

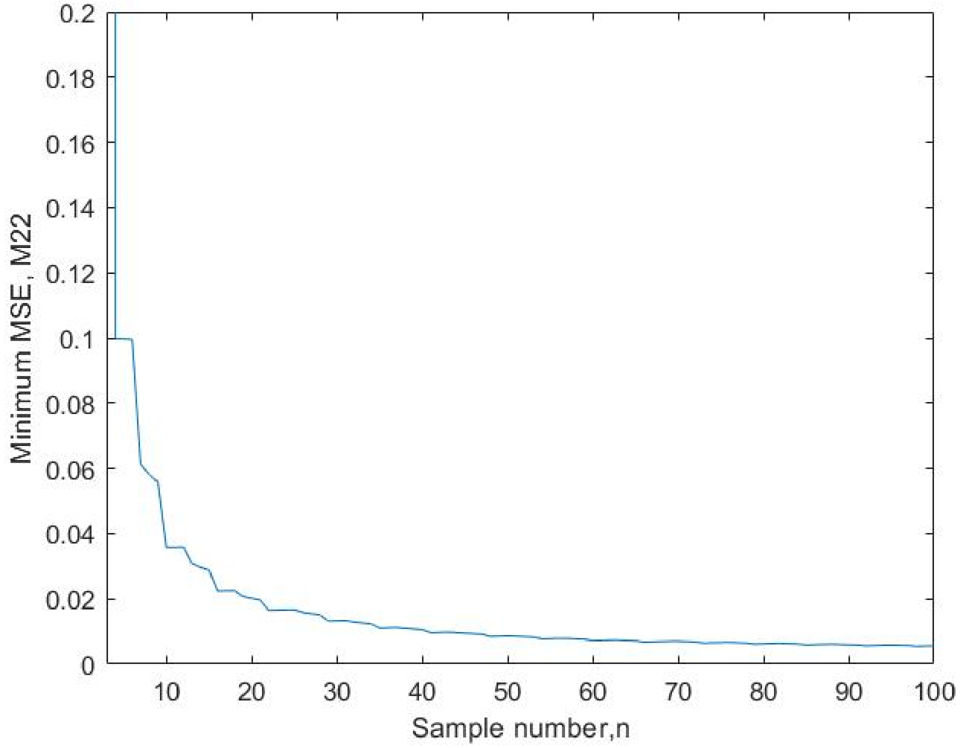

Fig 9: Kalman Filter minimum MSE_{22}

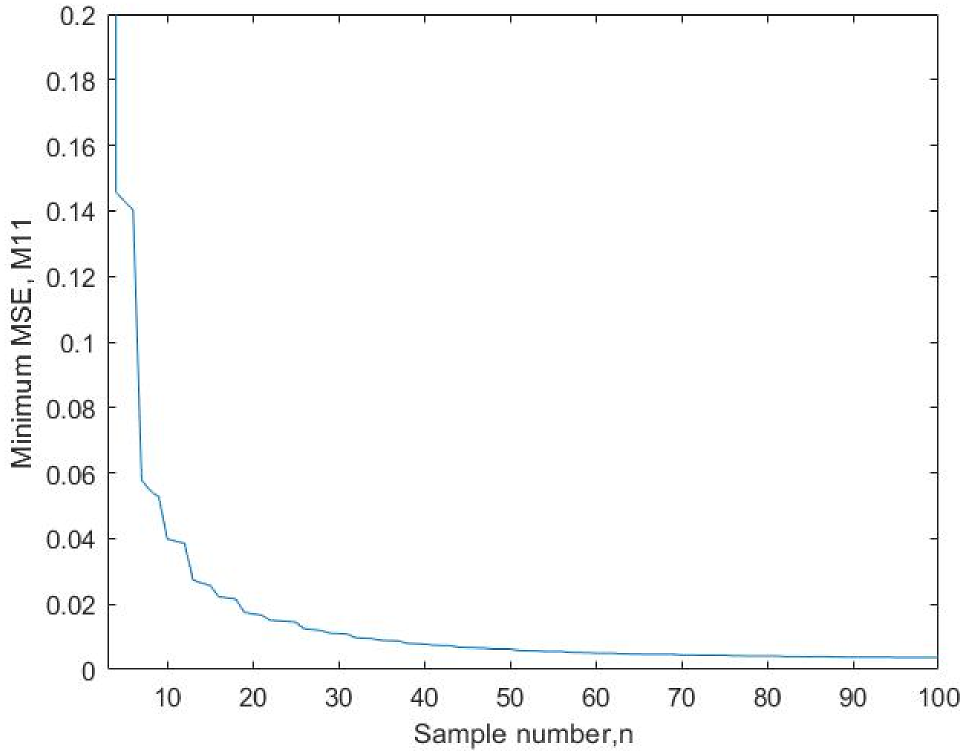

Fig 10: Kalman Filter minimum MSE_{11}

Observations:

- A particular realization is shown in fig 5 and fig 6. In which $h_n[0]$ is decaying to zero while $h_n[1]$is fairly constant. This is because the mean of the weights will be zero in steady – state. Due to the smaller value of $[A]_{11}$, $h_n[0]$ will decay more rapidly.
- When observation noise is added with $\sigma^2 = 0.1$, we obtain channel output shown in fig 4.
- The Kalman filter is applied with $\hat{\mathbf{h}}[-1|-1] = \mathbf{0}$ and $\mathbf{M}[-1|-1] = 100\mathbf{I}$., which were chosen to reflect little knowledge about the initial state. In the theoretical development of the Kalman filter the estimate of the initial state is given by the mean of s[-1]. In practice this is seldom known, so that we usually just choose an arbitrary initial state estimate with a large initial MSE matrix to avoid 'biasing' the Kalman filter towards that assumed state. The estimated tap weights are shown in fig 5 and 6. After an initial transient, the Kalman filter locks on to the true weights and tracks them closely.
- The Kalman filter gains are shown in fig 8 and fig 9. They appear to attain a periodic steady – state, although this behaviour is different than the usual steady – state which is discussed previously since $\mathbf{v}[n]$ varies with time and real steady – state is never attained. This is because at these times $\mathbf{v}[n]$ is zero due to the zero input and thus the observations contain only noise. The Kalman filter ignores these data samples by forcing the gain to be zero.
- The minimum MSEs are shown in figure 8 and are seen to decrease monotonically, although this generally will not be the case for a Kalman filter.

6.Conclusion

In this paper, we have thoroughly described the channel estimation and the problem statement followed by Kalman filtering to solve the problem .
To conclude that for time varying coefficients of a channel , we cannot use linear model, we need Kalman Filters because it is not possible to extend linear approach to this problem since there are too many parameters to estimate in a single go. If we assumed the filter coefficients are time invariant, the linear model could be applied to estimate the deterministic parameters.

The channel is modelled as a FIR filter with time varying coefficients. The observation model is assumed to be Gauss-Markov for tap weights. Vector-state Scalar-observation Kalman filter is used to estimate the time varying coefficients of the channel.

7.References

[1]. Fundamentals of Statistical Signal Processing: Estimation Theory by Steven M. Kay (ISBN 0-13-345711-7).

Appendix

MATLAB code

```
clear all;close all;
%initial conditions
h_(:,1)=[0;0];%initial weights
M_=100*eye(2);%Mmse initial
R=0.1;%observation noise varience
A=[0.99 0;0 0.999];%State matrix constant with time
Q=0.0001*eye(2);%process noise covariance
n=1:1:100;%sampling time
v= 0.5*(1 + square(n));%input to the channel
p=2;%number of weights modelled by the channel
x=zeros(1,100);
y=zeros(1,100);
K=zeros(2,100);
h1=zeros(1,100);
h2=zeros(1,100);
h_1=zeros(1,100);
h_2=zeros(1,100);
K1=zeros(1,99);
K2=zeros(1,99);
%modelling channel coeffiecients to generate channel coeffiecients with
%time
for i=1:100
    h1(i)=-0.006*i+1+randn*(0.0001);
    h2(i)=0.9+randn*(0.0001);
end
h=[h1;h2];
M1=zeros(1,100);
M2=zeros(1,100);
for i= 2:100
    h_(:,i)=A*h_(:,i-1);%process state equation
    y(i)=[v(i) v(i-2+1)]*h(:,i);%actual output
    x(i)=[v(i) v(i-2+1)]*h(:,i)+ R*randn;%measuremnet equation
    %prediction
    h_(:,i)=A*h_(:,i-1);
    M_=A*M_*A'+Q;
    den=R+[v(i) v(i-2+1)]*M_*[v(i) v(i-2+1)]';
    K(:,i)=(M_*[v(i) v(i-2+1)]')/den;
    h_(:,i)=h_(:,i)+K(:,i)*(x(i)-[v(i) v(i-2+1)]*h_(:,i));
    M_=(eye(2)-K(:,i)*[v(i) v(i-2+1)])*M_;
```

```
    M1(i)=M_(1,1);
    M2(i)=M_(2,2);
    h_1(i)=h_(1,i);
    h_2(i)=h_(2,i);
    K1(i-1)=K(1,i);
    K2(i-1)=K(2,i);
end
plot(v)
ylim([-1 2])
ylabel('Channel input(v[n])')
xlabel('Sample number,n')
figure
plot(y)
ylim([-1 2])
ylabel('Noiseless Channel output(y[n]')
xlabel('Sample number,n')
figure
plot(x)
ylim([-1 2])
ylabel('Channel input(x[n]')
xlabel('Sample number,n')
figure
plot(h1,'--')
hold on
plot(h_1,'-')
ylabel(' True and estimated tap weight, h1')
xlabel('Sample number,n')
hold off
figure
plot(h2,'--')
hold on
plot(h_2,'-')
ylabel(' True and estimated tap weight, h2')
xlabel('Sample number,n')
hold off
figure
plot(K1)
ylabel('Kalman gain, K1[n]')
xlabel('Sample number,n')
figure
plot(K2)
ylabel('Kalman gain, K2[n]')
xlabel('Sample number,n')
figure
plot(M1)
ylim([0 0.20])
xlim([3 100])
ylabel('Minimum MSE, M11')
xlabel('Sample number,n')
figure
plot(M2)
ylim([0 0.20])
xlim([3 100])
ylabel('Minimum MSE, M22')
xlabel('Sample number,n')
```