STATE SPACE APPROACH TO ADAPTIVE FILTERS
EEE 606: ADAPTIVE SIGNAL PROCESSING

Kaushik Iyer [1] (12236396175)
MS RAS (EE) On-Campus
[1]School of Electrical and Computer Engineering, Arizona State University, Tempe

Finite impulse response (FIR) filters normally use gradient-based techniques to adapt FIR filter coefficients for applications such as system identification. Although they are effective, they barely make use of the knowledge of a system. Moreover, they do not account for noises accurately, making them less robust. On the other hand, a state space (SS) model uses differential equations to estimate parameters of interest of the system concerned. SS models use a set of external observations to estimate the system parameters – in this case, FIR filter coefficients. Moreover, the error is adjusted dynamically to ensure better steady state properties and faster convergence. This work provides an insight into how an SS-based approach for system identification can outperform conventional adaptive FIR filtering algorithms.

***Index Terms*—FIR, LMS, SSLMS, SSLMSWAM, KF-LMS.**

I. INTRODUCTION

ADAPTIVE FIR filters have been used for applications such as system identification, noise cancellation, channel estimation etc. They are preferred over infinite impulse response (IIR) filters due to better stability properties. One of the best known adaptive FIR filters is the LMS filter [1]. LMS filter minimises a quadratic cost function to minimise the mean squared error to update the filter coefficients. A typical representation of an FIR adaptive filter is given in fig. 1.

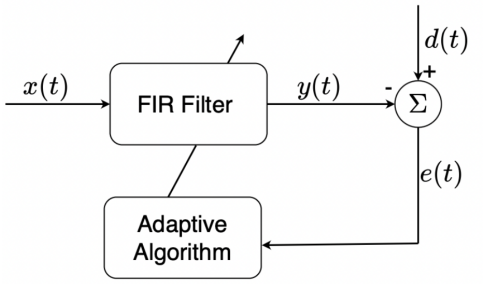

Fig. 1: FIR System representation

As shown in fig. 1, the input $x(t)$ is convolved with the FIR filter of a desired filter order, which provides the output $y(t)$. This is compared with a desired output, $d(t)$, to generate an error, $e(t)$. The error is used in the adaptive feedback algorithm to update the filter coefficients. These algorithms have been proven to be quite effective for system identification, but they have a couple of drawbacks:

1) First, as seen from the fig. 1, such adaptive FIR filters do not make explicit use of any knowledge of the system. Using some knowledge of the system can certainly improve the performance of the algorithm.
2) Second, there is no incorporation of noise in the model, which may make such algorithms less robust to impulsive and/or correlated inputs.
3) Third, the filter coefficients are adapted using a scaled feedback error. This scaling is usually done using a step size parameter. The error effects are not propagated every iteration, which can cause the steady state error to have more noise.
4) Last, the algorithm requires the desired output to be available, which may not be the case always.

A state space (SS) approach can help mitigate the shortcomings of FIR adaptive filters. A state space model is used to estimate the unknown states of a system given a set of observations available. The states of a system are a set of unknown parameters of interest, that need to be estimated to fully characterise the system. For instance, a sinusoid can be fully characterised using the amplitude, phase and frequency. So these can form the states of the sinusoid. In the case of a FIR filter, the states comprise the filter coefficients. An SS model in the context of adaptive FIR filters is shown in fig. 2.

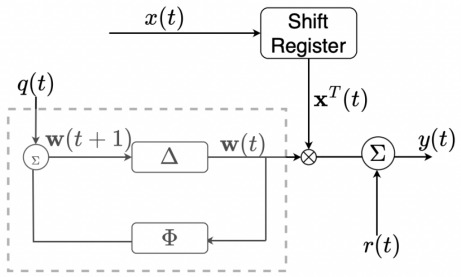

Fig. 2: State space representation of an adaptive filter [2].

The figure shows an input passing through a shift register to store the past samples. The states of the system, $\mathbf{w}(t)$ are propagated to the next iteration using a state transition matrix, $\mathbf{\Phi}$. $\mathbf{\Phi}$ is known if knowledge of the system is known. Since there could be modelling errors, a noise $q(t) \sim \mathcal{N}(0, \mathbf{Q})$ is introduced. The states are convolved with the input to produce the output. This output is compared against an observed output (say, $y_{\text{obs}}(t)$). Since this observed output can be noisy,

the computed output, $y(t)$ is appended with some noise, $r(t) \sim \mathcal{N}(0, \mathbf{R})$. $\mathbf{Q}, \mathbf{R}$ are known are the process and observation/measurement noise co-variance matrix respectively.

This work studies such SS approaches. Three such algorithms studied here are the state space least mean squares (SSLMS) [3], state space least mean squares with adaptive memory (SSLMSWAM) [4] and the Kalman filter based LMS (KF-LMS) [5]. There are three different tests considered:

1) First test is to the track a sinusoid of a known frequency, by estimating the phase and amplitude. The SSLMS is used for this test and compared with the LMS, NLMS and RLS.
2) Second test is to track Van der Pol oscillations. The SSLMSWAM is used for this test and compared with the same three algorithms as in test 1.
3) Third test is to track an electrocardiograph (ECG) signal since the observed ECG signal can have instrument induced noise. The KF-LMS is used for this test and compared against the other three adaptive filters.

The work is organised as follows: Section II describes the SSLMS, SSLMSWAM and KF-LMS algorithms. Section III details the tests and discusses the results. Finally, Section IV concludes the work. An appendix is attached showing the codes used for the algorithms.

II. SS-based Algorithms

A. SSLMS

The SS-LMS algorithm has been described in [3]. The idea is to estimate the unknown filter coefficients $\mathbf{w}(k)$ using a set of observed signals ($y_{\text{obs}}(k)$). The theory remains the same for continuous and discrete time systems, hence only one of them is presented. There are two notations used that will be consistent throughout the three algorithms. $\bar{a}$ denotes the predicted value of a, which is computed using the system dynamics model. $\hat{a}$ denotes the updated/estimated value that is obtained using the external observation y_{obs}. The system model or the state propagation equation for the SSLMS is given by:

$$\bar{\mathbf{w}}(k) = \mathbf{\Phi}_{k|k-1}\hat{\mathbf{w}}(k-1) \tag{1}$$

For the case of SSLMS, $\mathbf{\Phi}_{k|k-1} = \mathbf{I}_L$, where $\mathbf{I}_L$ is the identity matrix of dimension L for an L^{th} order filter. There are two types of errors and two types of outputs defined for SSLMS:

$$\begin{aligned} \epsilon(k) &= y_{obs}(k) - \bar{y}(k) \;\text{ where, }\; \bar{y}(k) = \mathbf{C}(k)\bar{\mathbf{w}}(k) \\ e(k) &= y_{obs}(k) - \hat{y}(k) \;\text{ where, }\; \hat{y}(k) = \mathbf{C}(k)\hat{\mathbf{w}}(k) \\ e(k) &= \epsilon(k) - \mathbf{C}(k)\left(\hat{\mathbf{w}}(k) - \bar{\mathbf{w}}(k)\right) \end{aligned} \tag{2}$$

where $\epsilon(k)$ and $e(k)$ denote the prediction and estimation error respectively. $\mathbf{C}(k)$ is called the observation matrix, which relates the observed outputs to the states of the system. For the case of FIR filters, $\mathbf{C}(k) \implies \mathbf{x}^T(k) = \begin{bmatrix} x(k) & x(k-1) & \dots & x(k-L) \end{bmatrix}$.

The state update equation uses the error in the feedback which is scaled by what is called the "observer gain" $\mathbf{K}(k)$. This gain is to be found by optimising a cost function.

$$\hat{\mathbf{w}}(k) = \bar{\mathbf{w}}(k) + \mathbf{K}(k)\epsilon(k) \tag{3}$$

The solution for $\mathbf{K}(k)$ in [3] use the minimum norm solution to find $\mathbf{K}(k)$, which is given by:

$$\mathbf{K}(k) = \mathbf{C}^T(k)\left(\mathbf{C}(k)\mathbf{C}^T(k)\right)^{-1} \tag{4}$$

For the state space model to be controllable, i.e., the the input can help the states achieve the output for any initial condition, another matrix $\mathbf{G}(k)$ is added to ensure the controllability condition. $\mathbf{G}(k)$ ensures that $\left[\mathbf{\Phi}_{k|k-1} - \mathbf{K}(k)\mathbf{C}(k)\mathbf{\Phi}_{k|k-1}, \mathbf{K}(k)\right]$ is full rank, which is desirable. Finally, a step size μ is also added to $\mathbf{K}(k)$, to achieve faster convergence. Adding this to (4) gives:

$$\mathbf{K}(k) = \mu\mathbf{G}\mathbf{C}^T(k)\left(\mathbf{C}(k)\mathbf{C}^T(k)\right)^{-1} \tag{5}$$

Alternate forms of (5) exist for better numerical stability and/or reducing computation. A drawback of this method is that it still does not incorporate noise effectively and μ is a constant.

B. SSLMSWAM

This is presented in both [3], [4]. SSLMSWAM builds on the limitation of SSLMS by adding a time-varying μ. The only changes in the equations are [3]:

$$\begin{aligned} \mu(k) &= \mu(k-1) + \alpha\psi^T(k-1)\mathbf{\Phi}_{k|k-1}^T\mathbf{C}^T(k)\epsilon(k) \\ \psi(k) &= \left(\mathbf{\Phi}_{k|k-1} - \mathbf{K}(k)\mathbf{C}(k)\mathbf{\Phi}_{k|k-1}\right)\psi(k-1) + \\ &\quad \mathbf{G}(k)\mathbf{C}^T(k)\epsilon(k) \end{aligned} \tag{6}$$

where ψ is the gradient for μ and α is a scaling factor. SSLMSWAM requires an initialisation for μ and ψ. $\psi = \mathbf{0}$ is usually a good initialisation.

C. KF-LMS

The KF-LMS builds over the limitations of SSLMS and SSLMSWAM by incorporating noise into to the SS model. The idea for KF-LMS is taken from [5]. The KF is a special case of the SS. It incorporates noise co-variances from the system model and the noisy observations in the adaptive gain. This helps account for the gradient noise or linearisation errors that remain unaccounteed for in LMS, NLMS, RLS etc. The KF-LMS minimises the state error covariance matrix defined as follows:

$$\mathbf{P}(k) = E\left[\left(\mathbf{w}(k) - \mathbf{w}_{opt}(k)\right)^T\left(\mathbf{w}(k) - \mathbf{w}_{opt}(k)\right)\right] \tag{7}$$

where $\mathbf{w}_{\text{opt}}$ denotes the optimal filter coefficients. The KF-LMS is divided into two stages:

$$\begin{aligned} &\text{Prediction Steps} \\ &\quad \bar{\mathbf{w}}(k) = \mathbf{I}\hat{\mathbf{w}}(k-1) \\ &\quad \bar{\mathbf{P}}(k) = \hat{\mathbf{P}}(k-1) + \mathbf{Q} \\ &\text{Update Steps} \\ &\quad \mathbf{K}(k) = \bar{\mathbf{P}}(k)\mathbf{C}^T(k)\left(\mathbf{C}(k)\bar{\mathbf{P}}(k)\mathbf{C}^T(k) + \mathbf{R}\right)^{-1} \\ &e_{meas}(k) = y_{obs}(k) - \mathbf{C}(k)\bar{\mathbf{w}}(k) \\ &\quad \hat{\mathbf{w}}(k) = \bar{\mathbf{w}}(k) + \mathbf{K}(k)e_{meas}(k) \\ &\quad \hat{\mathbf{P}}(k) = \left(\mathbf{I} - \mathbf{K}(k)\mathbf{C}(k)\right)\bar{\mathbf{P}}(k-1) \end{aligned} \tag{8}$$

where $\mathbf{K}$ is called the Kalman gain, $\mathbf{Q}, \mathbf{R}$ are as described previously. KF-LMS requires an initialisation for $\mathbf{P}(\mathbf{0})$, $\mathbf{w}(0)$, $\mathbf{Q}$ and $\mathbf{R}$. These can be initialised as diagonal matrices using knowledge of the system.

III. Simulation Results and Discussion

The algorithms explained in Section II are tested through simulations.

A. Tracking Sinusoid

The first simulation is to track a sinusoid of a known frequency by estimating the amplitude and phase. In this case, since the structure of a sinusoid is known , the states do not contain the FIR filter coefficients, but the amplitude and phase. The sinusoid is defined by the equation:

$$y(k) = \sigma_s \cos(\omega_0 k T_s + \phi) + \nu(kT_s) \tag{9}$$

where $\sigma_s = a^2/2$ is the signal power and a is the amplitude. ϕ is the phase and ν is a zero-mean Gaussian noise of variance 0.4. The clean sinusoid and the observed (corrupted by noise) is shown in fig. 3.

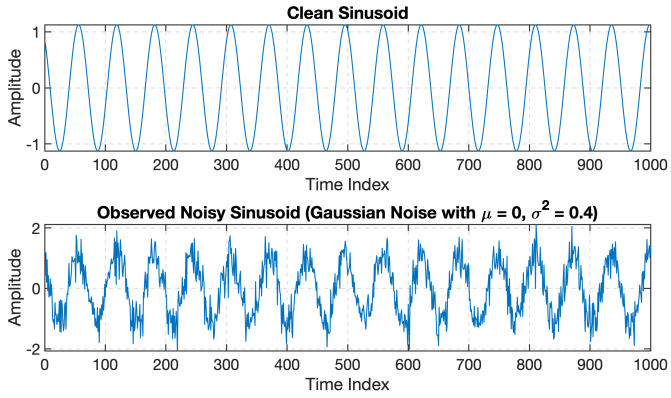

Fig. 3: Clean and observed (noisy) sinusoid.

The sinusoid parameters have been tracked using:

1) SS-LMS ($\mathbf{G} = \mathbf{I}$, $\mu = 0.2$)
2) LMS ($L = 5, \mu = 0.005$)
3) NLMS ($L = 5, \mu = 0.1$)
4) RLS ($L = 5, \beta = 0.98, \delta = 0.01$)

The matrices Φ and $\mathbf{C}$ for this case are given by:

$$\begin{aligned} \mathbf{\Phi} &= \begin{bmatrix} \cos(\omega_0 T_s) & \sin(\omega_0 T_s) \\ -\sin(\omega_0 T_s) & \cos(\omega_0 T_s) \end{bmatrix} \\ \mathbf{C} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned} \tag{10}$$

The mean square error (MSE) and the absolute error plots are shown in figs. 5 and 6 respectively.

The mean values of each algorithm in each case is shown on the plot. It is evident that the SSLMS outperforms the other algorithms. The absolute error plot in fig. 5 gives more insight into the transient and steady state behaviour of the four algorithms compared here. The SSLMS is seen to have the shortest transient behaviour and the least number of spikes. This is a desirable property for any algorithm and shows how fast the algorithm adapts to the input.

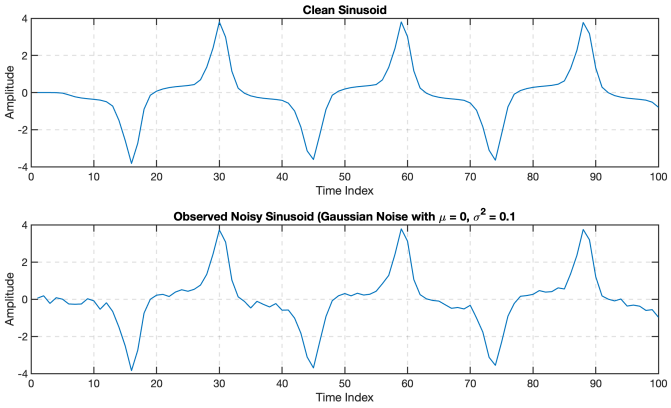

Fig. 4: Clean and observed (noisy) Van der Pol oscillation signal x_2.

It may be argued that the order of the FIR filter for LMS, NLMS and RLS algorithms is really low. Although it is true that by increasing the order of the filter, the performance of the algorithms improves, it is necessary to note that the SSLMS deals with matrices and vectors of dimension 2, which is smaller than that for the FIR adaptive algorithms. This also highlights the fact that SS based algorithms are more efficient when the system model is known.

B. Tracking Van der Pol Oscillations

The second simulation is to track the Van der Pol oscillations. These oscillations are governed by a second order differential equation. This can be broken down into two first-order differential equations given as [3], [4]:

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -x_1 + m\left(1 - x_1^2\right) x_2 \end{aligned} \tag{11}$$

where m is a damping coefficient. There are two signals x_1 and x_2, that can be observed, but in this work, only x_2 is tracked. The Van der Pol oscillator is corrupted by a zero-mean Gaussian noise of variance 0.1. The clean and noisy oscillations are shown in fig. 4.

The damping factor m was set to 2 for the simulation. The initialisation for this simulation are:

1) SS-LMSWAM ($\mu(0) = 0.1, \psi(0) = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \alpha = 0.01$)
2) LMS ($L = 5, \mu = 0.01$)
3) NLMS ($L = 5, \mu = 0.2$)
4) RLS ($L = 5, \beta = 0.9, \delta = 0.001$)

The matrices $\mathbf{\Phi}, \mathbf{G}, \mathbf{C}$ are given by [3]:

$$\begin{aligned} \mathbf{\Phi} &= \begin{bmatrix} 1 & T_s & \frac{T_s^2}{2} \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{G} &= \begin{bmatrix} 1 & 0 & 0 \\ 0.3 & 0 & 0 \\ 0.3 & 0 & 0 \end{bmatrix} \\ \mathbf{C} &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \end{aligned} \tag{12}$$

The plots of the MSE and estimation error are given in figs. 9 and 10. The mean values of teh MSE and the estimation error

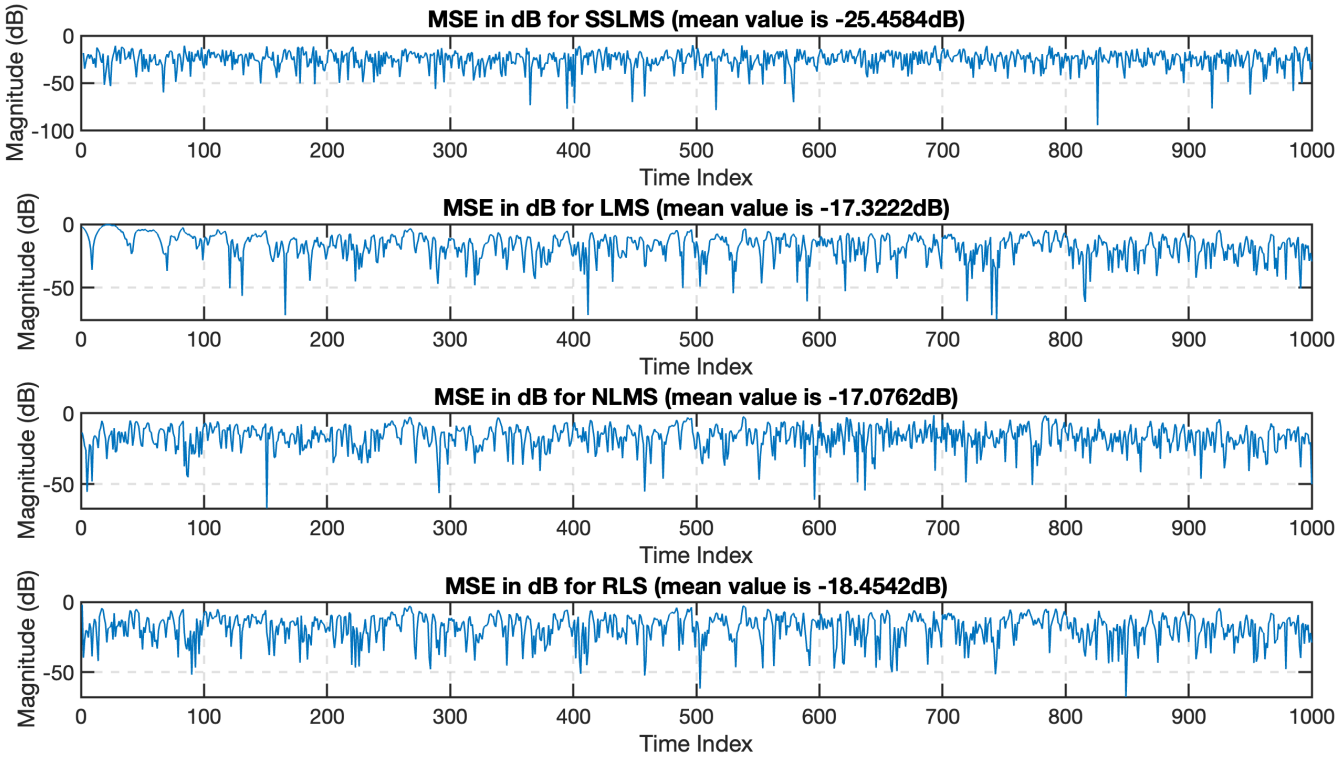

Fig. 5: MSE plot in dB for (a) SSLMS, (b) LMS, (c) NLMS, (d), RLS.

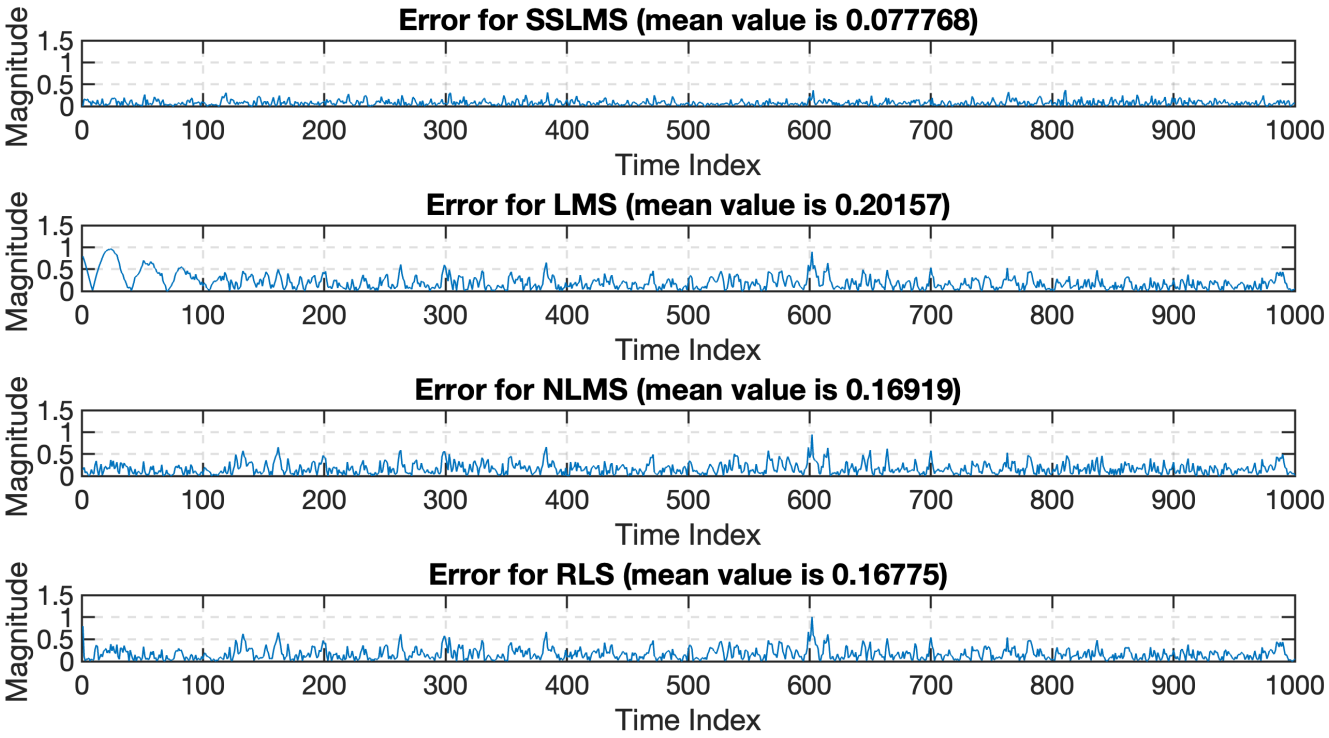

Fig. 6: Absolute error plot for (a) SSLMS, (b) LMS, (c) NLMS, (d), RLS.

for each algorithm is given as well. The results in this case are comparable. However, they are very noisy in all the cases. Moreover, the SSLMSWAM shows some peaks in the error. The learning rate or the step size (μ) adapts to the algorithm. The learning curve is shown in fig. 7. The learning curve in this case steadily increases til it saturates at about 1.2, which is quite high. This is the possible reason for peaks in the results of the SSLMSWAM. This also highlights the fact that that

the algorithm is very sensitive to variations in mu as well as initialisation.

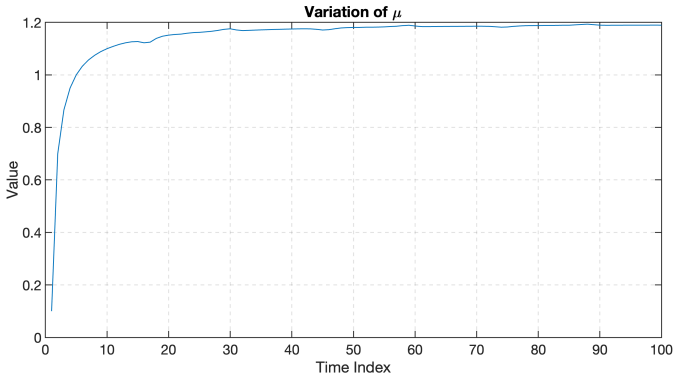

Fig. 7: Learning curve for SSLMSWAM

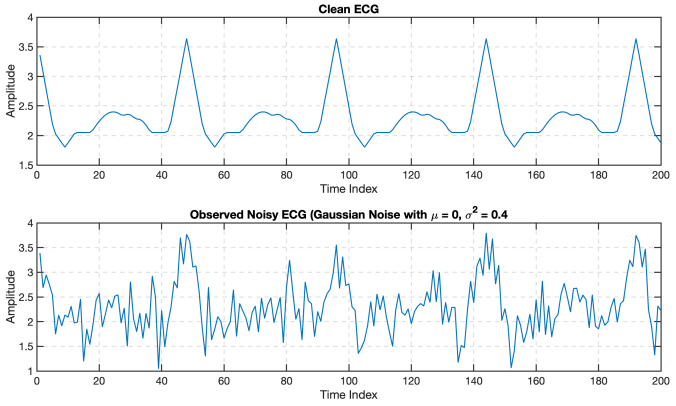

Fig. 8: Clean and observed (noisy) ECG signal.

C. Tracking ECG Signal

The third simulation is to track an ECG signal to test the performance of the KF-LMS. The ECG signal plays an important role in biomedical applications and it becomes necessary to have the ECG signal free of noise for accurate diagnosis. Since the ECG signal does not have a specific mathematical structure, it is hard to have a system model for it. Hence, the KF-LMS has its states as the filter coefficients of an L^{th} order FIR filter. The benefit of KF-LMS is due to the incorporation of noises and decent propogation of errors. The clean and noisy ECG signals are shown in fig. 8. The ECG signal has been taken from the MIT BIH Arrhythmia database [6]. The signal here has been corrupted by a zero-mean Gaussian noise of variance 0.4.

The initialisation for this simulation are:

1) KF-LMS ($L = 15, P = 10\mathbf{I}_{L+1}, \mathbf{Q} = 2\mathbf{I}_{L+1}, \mathbf{R} = \left(\frac{1}{\sigma^2}\right), \mathbf{w}(0) = \mathbf{0}$)
2) LMS ($L = 15, \mu = 0.01$)
3) NLMS ($L = 15, \mu = 0.25$)
4) RLS ($L = 15, \beta = 0.99, \delta = 0.001$)

The MSE and estimation error plots are given in figs. 11 and 12 respectively.

The KF-LMS clearly outperforms the other algorithms. Moreover, the transient response dies out in about 3 iterations. Moreover, the results are very smooth, indicating a very stable and robust algorithm.

IV. Conclusion

This work presented different state space approaches and compared them with the FIR adaptive filters. The KF-LMS was seen to be the best among SSMLMS and SSLMSWAM.

The idea of the project was to introduce concepts of state space models that are heavily used in control theory for applications in robotics, target tracking, navigation etc. If a system model is known, such algorithms are very efficient and robust even for applications of signal processing such as system identification, noise cancellation, channel estimation etc. Even for the case when the system model is unknown, the KF-LMS showed superior results while tracking an ECG signal. This is largely due to the incorporation of noise models in the adaptation. Another benefit of such algorithms is that the desired signal need not be used. The observed noisy signal is enough to track the system/signal, if noise characteristics are modeled accurately.

I have used SS models extensively for my research in navigation based algorithms. However, I had no idea they were so effective in such applications. My key learning was getting a deeper insight into how the domains of control theory and signal processing can be linked and worked upon.

Appendix

The MATLAB codes are attached for SSLMS, SSLMSWAM and KF-LMS.

References

[1] S. S. Haykin, *Adaptive filter theory*. Pearson Education India, 2002.
[2] D. Lippuner and G. S. Moschytz, "The kalman filter in the context of adaptive filter theory," *International journal of circuit theory and applications*, vol. 32, no. 4, pp. 223–253, 2004.
[3] M. B. Malik and M. Salman, "State-space least mean square," *Digital Signal Processing*, vol. 18, no. 3, pp. 334–345, 2008.
[4] M. SALMAN, "Adaptive estimation using state-space methods," Ph.D. dissertation, National University of Sciences and Technology, Pakistan, 2009.
[5] P. A. Lopes and J. B. Gerald, "New normalized lms algorithms based on the kalman filter," in *2007 IEEE International Symposium on Circuits and Systems*. IEEE, 2007, pp. 117–120.
[6] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

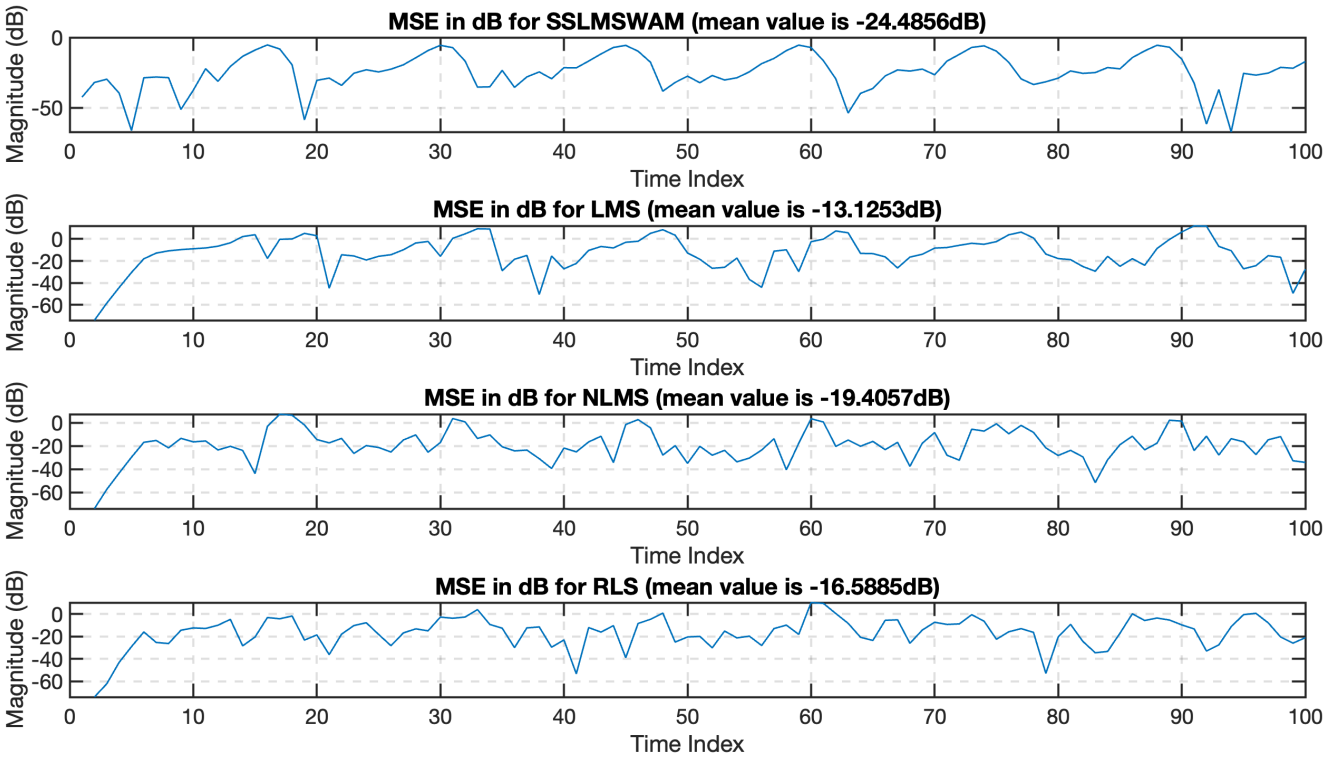

Fig. 9: MSE plot in dB for (a) SSLMSWAM, (b) LMS, (c) NLMS, (d), RLS.

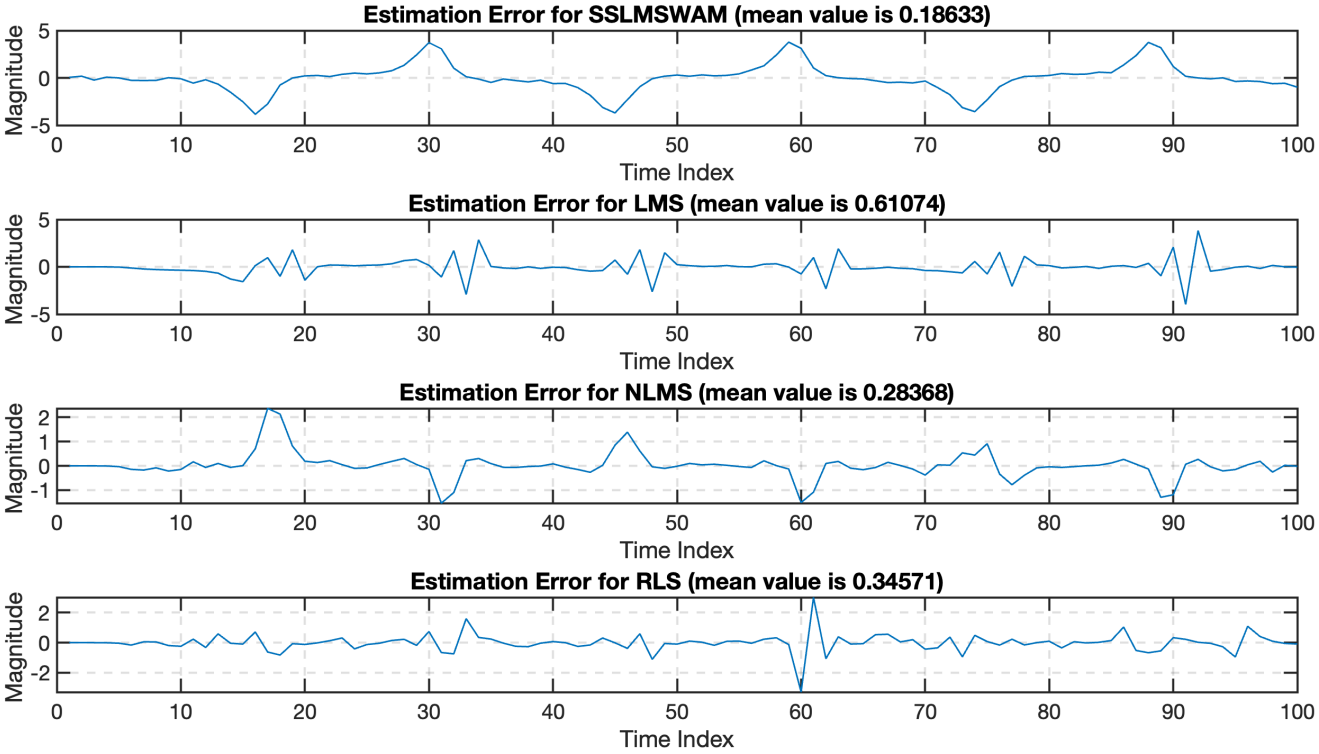

Fig. 10: Estimation error plot for (a) SSLMSWAM, (b) LMS, (c) NLMS, (d), RLS.

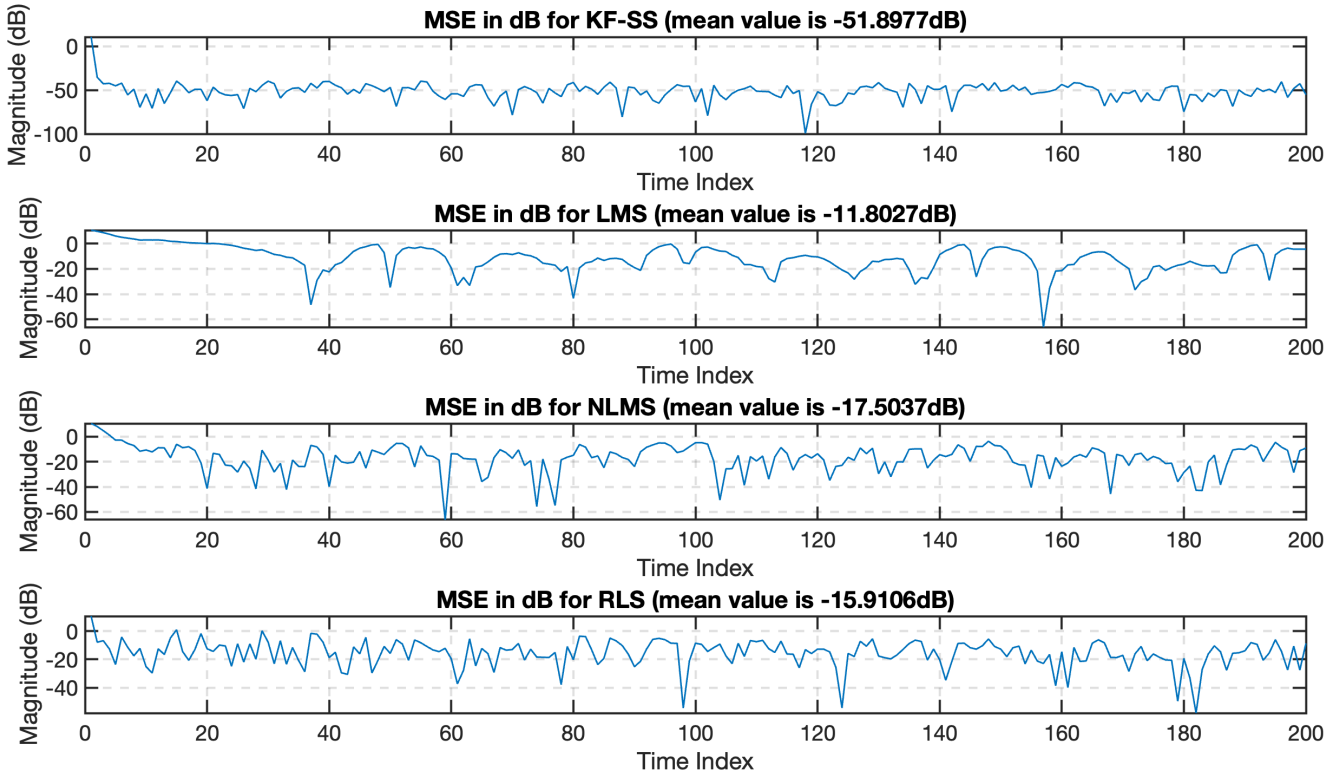

Fig. 11: MSE plot in dB for (a) KF-LMS, (b) LMS, (c) NLMS, (d), RLS.

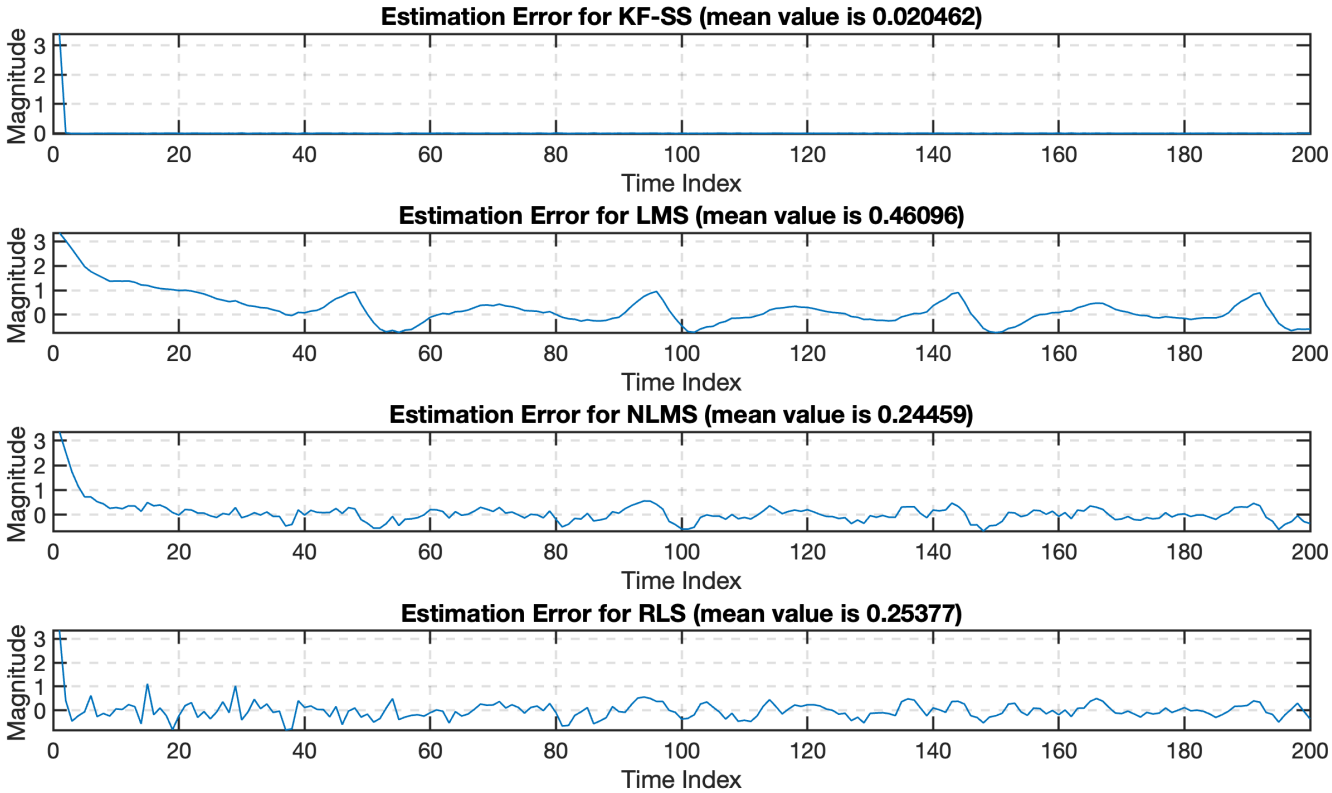

Fig. 12: Estimation error plot for (a) KF-LMS, (b) LMS, (c) NLMS, (d), RLS.

```
%% State Space Least Mean Squares (SSLMS) Algorithm
% Application to track a sinusoid with unknown amplitude and phase but known
% frequency. Such a case case arises in case of Power Line Interference (PLI)
% for ECG signals.  Signal model: y[n] = σ sin(ωnT + φ) + ν(nT)
% where σ = (a^2)/2 is the signal power and a is the amplitude
%       ω is the known frequency (60 Hz)
%       T is the sampling time (s)
%       φ is the singal phase (rad)
%       ν is the observation noise
%       y is the observed signal corrupted by noise
% Definitions and Initialisations
n     = 2;      % Dimension of state space
x     = [1 0]'; % State of the system (initialised to [1 0]' -> [amp, phase]')
Ts    = 1;      % Sampling time
N     = 1000;   % Number of iterations
mu    = 0.8;
% Parameters of the actual and observed signal
phi    =  pi/4;                            % Phase
a      = 1.5;                              % Amplitude
wo     = 0.1;                              % Known frequency
t      = 0:1:N-1;      t = t';             % Time vector
sigma2 = 0.4;
nu     = sigma2*randn(N,1);                % Noise
yobs   = ((a^2)/2)*cos(wo*t*Ts + phi) + nu; % Observation signal
y      = yobs - nu;                        % Clean signal
% System matrices for SSLMS
A = [cos(wo*Ts) sin(wo*Ts);
     -sin(wo*Ts)  cos(wo*Ts)];
C = [1 0];
G = eye(2);
K = [1;1];
% Initialise vectors for plots
eSSLMS  = zeros(N,1);  yhat    = zeros(N,1);   xhat = x;
% Loop for SSLMS
for k = 1:N
    xbar = A*xhat;                  % State vector preditcion
    ybar = C*xbar;                  % Output prediction
    epsilon = yobs(k) - ybar;       % Output prediction error
    xhat = xbar + K*epsilon;        % State vector update using output error
    yhat(k)   = C*xhat;             % Output estimation
    eSSLMS(k) = yobs(k) - yhat(k);  % Output estimation error
    K = mu*G*C'*inv(C*C');          % Compute the observer gain
end
```

Fig. 13: MATLAB code for SSLMS.

```
% Final Project EEE 606
% Kaushik Iyer (1223696175)

%% State Space Least Mean Squares With Adaptive Memory (SSLMSWAM) Algorithm
% Application to track a Van der Pol oscillations defined by the equations:
% x1' = x2
% x2' = −x1 + m(1 − x1^2)x2
% The state space is given by the constant acceleration model and the
% signal x2 is observed


% Definitions and Initialisations
n             = 3;          % Dimension of state space
x             = [0 0 0]';   % State of the system
Ts            = 0.01;       % Sampling time
N             = 2500;       % Number of iterations
muSSLMSWAM(1) = 0.1;        % Initial step size
alpha         = 0.01;       % Learning parameter
psi           = zeros(n,1); % Initial gradient

% System and Gain matrices
A = [1 Ts 0.5*Ts^2; 0 1 Ts; 0 0 1]; % State propagation matrix
C = [1 0 0];                        % Observation matrix
G = [1 0 0; 0.3 0 0; 0.3 0 0];      % Constant Gain

% Generate Van der Pol oscillation
x0   = [2;0];
y    = GenVanderPolOsc(N,2,x0);
y    = y.y(end,1:100)';         % extract only the last column as we observe x2
Nvar = 0.1;                     % Noise variance
nu   = Nvar*randn(length(y),1); % Generate noise
yobs = y + nu;                  % Observed oscillation (noisy)

epsilon                = zeros(length(y),1); % Store Output error
muSSLMSWAM(2:length(y)) = 0;                 % Store mu
% Loop for SSLMSWAM
for i = 1:length(y)
    K             = muSSLMSWAM(i)*G*C'/(C*C');                    % Compute Observer Gain
    epsilon(i)    = yobs(i) - C*A*x;                              % Compute Output Error
    x             = A*x + K*epsilon(i);                           % Update state
    muSSLMSWAM(i) = muSSLMSWAM(i) + alpha*psi'*A'*C'*epsilon(i);  % Update mu
    psi           = (A - K*C*A)*psi + G*C'*epsilon(i);            % Update gradient
end
```

Fig. 14: MATLAB code for SSLMSWAM.

```
%% State Space / KF Approach to track Noisy ECG Signals
% State Space/ KF approach for systems where the dynamical model is not
% known and/or the system is an FIR filter. Helps incorporate noise models
% for the observed noisy input signal and also incorporates the gradient
% noise effects that conventional Adaptive filtering algorithms (LMS, NLMS,
% RLS, etc) fail to account for.

% Initialisations and Definitions for the system
L       = 15;           % Order of the FIR filter
x       = zeros(L+1,1); % State vector (contains filter weights)
P0      = 10;            % State error variance
P       = P0*eye(L+1);  % State error covariance matrix
Q0      = 2e0;          % Gradient noise
Q       = Q0*eye(L+1);  % Process noise co-variance matrix
F       = eye(L+1);     % State transition matrix
xBuffer = zeros(L+1,1); % Convolution buffer
muSS    = 0.001;

% Generate ECG signal (clean and noisy)
N = 2;
y = genECG(N)';
Nvar = 0.4;
nu = Nvar*randn(length(y),1);
yobs = y + nu;

R = (1/Nvar)*eye(1);   % Observation noise co-variance matrix
ySS = zeros(length(y),1);
e_meas = zeros(length(y),1);
for i = 1:length(y)
    xBuffer = [yobs(i); xBuffer(1:end-1)];
    H       = xBuffer';
    e_meas(i)  = yobs(i) - H*x;
    % Predict
    x = F*x + muSS*e_meas(i)*xBuffer;
    P = F*P*F + Q;
    % Update
    K = P*H'/(H*P*H' + R);
    x = x + K*(e_meas(i));
    P = (eye(1) - K*H)*P;

    ySS(i) = x'*xBuffer;
end
```

Fig. 15: MATLAB code for KF-LMS.