Experiment No. 8

AIM: Implementation of the recursive least squares (RLS) algorithm for adaptive filtering.

SODTWARE TO BE USED: PYTHON.

THEORY:

Method of Least Squares:

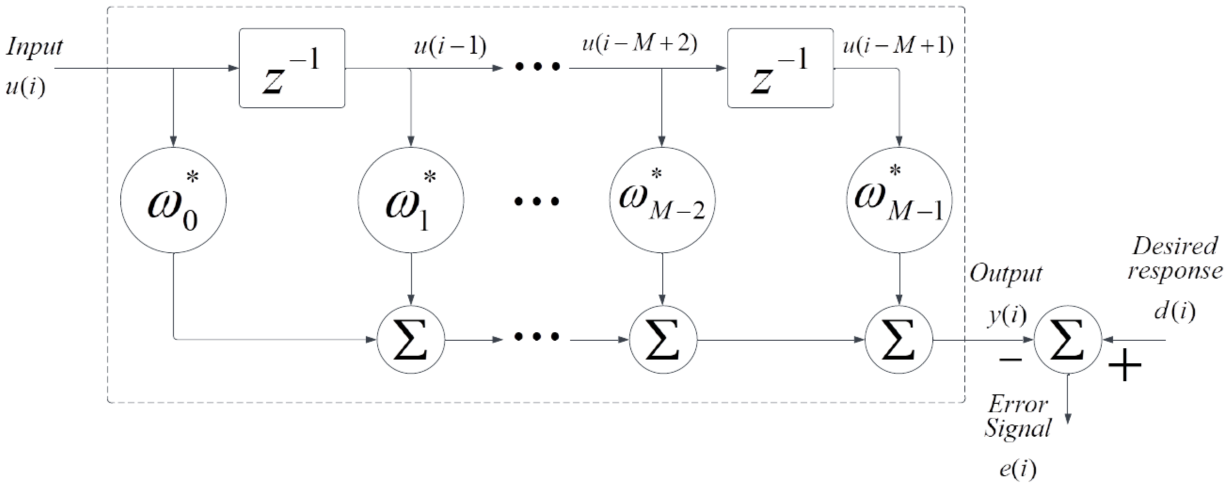

Figure 1: Filter model of least squares.

Method of least squares is deterministic in approach to minimize the sum of the squares of the difference between the desired signal $d(i)$ and filter output $y(i)$ for $i = 1, 2, 3, ..., N$.

$$y(i) = \sum_{k=0}^{M-1} \omega_k u(i-k) \quad \textbf{(1)}$$

$$e(i) = d(i) - y(i) \quad \textbf{(2)}$$

$u(i), u(i-1), u(i-2), ..., u(i-M+1)$ represents M-taped samples of input signal, ω_k; $k = [0, M-1]$ is the filter weight of length M and $e(i)$ is the estimation error. The tap weight ω_k is tuned to minimize the cost function ξ consisting sum of error squares given as

$$\xi(\omega_0, ..., \omega_{M-1}) = \sum_{i=i_1}^{i_2} |e(i)|^2 \quad \textbf{(3)}$$

where i_1 and i_2 defines the window for which the error minimization occurs. Gradient of ξ is given as

$$\nabla_k \xi = -2\sum_{i=i_1}^{i_2} u(i-k)e^*(i). \quad \textbf{(4)}$$

For minimizing ξ

$$\nabla_k \xi = 0. \quad \textbf{(5)}$$

Substituting eq. 4 in eq. 5 we get

$$\sum_{i=i_1}^{i_2} u(i-k)e^*_{\min}(i) = 0 \quad \textbf{(6)}$$

where $e_{\min}(i)$ denotes the value of $e(i)$ for which ξ is minimized. As per the *principle of orthogonality* the minimum-error time series $e_{\min}(i)$ is orthogonal to the time series $u(i-k)$ applied to tap k of a filter of length M when filter is operating in its least-squares condition. Substituting eq. 1 in eq. 2 we get

$$e_{\min}(i) = d(i) - \sum_{t=0}^{M-1} \hat{\omega}_t^* u(i-t) \quad \textbf{(7)}$$

Substituting eq. 7 in eq. 6 we get

$$\sum_{t=0}^{M-1} \hat{\omega}_t \sum_{i=i_1}^{i_2} u(i-k)u^*(i-t) = \sum_{i=i_1}^{i_2} u(i-k)d^*(i) \quad \textbf{(8)}$$

where $\phi(t,k) = \sum_{i=i_1}^{i_2} u(i-k)u^*(i-t)$ is the *time average auto-correlation function* of tap inputs,

$z(-k) = \sum_{i=i_1}^{i_2} u(i-k)d^*(i)$ is the *time average cross-correlation function* between the tap inputs and the desired response and $t = [0, M-1]$. We can rewrite eq. 8 as

$$\sum_{t=0}^{M-1} \hat{\omega}_t \phi(t,k) = z(-k)\,. \quad \textbf{(9)}$$

The eq. 9 can be represented in matrix form as

$$\mathbf{\Phi}\hat{\mathbf{w}} = \mathbf{z} \quad \textbf{(10)}$$

where Φ is the non-singular $M-by-M$ *time average auto-correlation matrix* of tap inputs, $\mathbf{z}$ is the $M-by-1$ *time average cross correlation vector* between tap inputs and desired response and $\hat{\mathbf{w}}$ is the $M-by-1$ *tap weight vector* of the least squares filter.

$$\mathbf{\Phi} = \begin{bmatrix} \phi(0,0) & \phi(1,0) & \cdots & \phi(M-1,0) \\ \phi(0,1) & \phi(1,1) & \cdots & \phi(M-1,1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(0,M-1) & \phi(1,M-1) & \cdots & \phi(M-1,M-1) \end{bmatrix} \quad \textbf{(10)}$$

$$\mathbf{z} = [z(0), z(-1), ..., z(-M+1)]^T \quad \textbf{(11)}$$

$$\hat{\mathbf{w}} = [\hat{\omega}_0, \hat{\omega}_1, ..., \hat{\omega}_{M-1}]^T \quad \textbf{(12)}$$

Finally, the tap-weight vector of the linear least-square filter can be derived as

$$\hat{\mathbf{w}} = \mathbf{\Phi}^{-1}\mathbf{z}\,. \quad \textbf{(13)}$$

Recursive Least Squares:

We extend to recursive least squares for use in real-time environments, where we estimate the weight prior to the availability of the input signal and update the weight with correction after the input is available.

Let introduce two parameters λ^n and δ. where λ is an exponential weighting factor; $0 < \lambda \le 1$ for large value of n, λ^n tends to zero hence, act as a forgetting factor to forget past values and δ is a positive real number called the regularization parameter which stabilizes RLS solution by smoothing it, where δ takes a large positive constant for small SNR and small positive constant for large SNR.

The time average auto-correlation matrix can be written as

$$\mathbf{\Phi}(n) = \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)\mathbf{u}^H(i) + \delta\lambda^n\mathbf{I} \tag{14}$$

$$\begin{aligned} \mathbf{\Phi}(n) &= \sum_{i=1}^{n-1} \lambda^{n-1-i}\mathbf{u}(i)\mathbf{u}^H(i) + \delta\lambda^{n-1}\mathbf{I} + \mathbf{u}(n)\mathbf{u}^H(n) \\ \mathbf{\Phi}(n) &= \lambda\mathbf{\Phi}(n-1) + \mathbf{u}(n)\mathbf{u}^H(n) \end{aligned} \tag{15}$$

Similarly, time average cross-correlation matrix $\mathbf{z}$ can be written as

$$\begin{aligned} \mathbf{z}(n) &= \sum_{i=1}^{n} \lambda^{n-i}\mathbf{u}(i)d^*(i) \\ \mathbf{z}(n) &= \lambda\mathbf{z}(n-1) + \mathbf{u}(n)d^*(n) \end{aligned} \tag{16}$$

By using matrix inversion lemma on eq. 15 we get

$$\begin{aligned} \mathbf{A} &= \mathbf{\Phi}(n) \\ \mathbf{B}^{-1} &= \lambda\mathbf{\Phi}(n-1) \\ \mathbf{C} &= \mathbf{u}(n) \\ \mathbf{D} &= 1 \end{aligned}$$

$$\mathbf{\Phi}^{-1}(n) = \lambda^{-1}\mathbf{\Phi}^{-1}(n-1) - \frac{\lambda^{-2}\mathbf{\Phi}^{-1}\mathbf{u}(n)\mathbf{u}^H(n)\mathbf{\Phi}^{-1}(n-1)}{1+\lambda^{-1}\mathbf{u}^H(n)\mathbf{\Phi}^{-1}(n-1)\mathbf{u}(n)}. \tag{17}$$

For convenience let's consider

$$\begin{aligned} \mathbf{P}(n) &= \mathbf{\Phi}^{-1}(n) \\ \mathbf{k}(n) &= \frac{\lambda^{-1}\mathbf{P}(n-1)\mathbf{u}(n)}{1+\lambda^{-1}\mathbf{u}^H\mathbf{P}(n-1)\mathbf{u}(n)} \end{aligned}$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\mathbf{k}(n)\mathbf{u}^H(n)\mathbf{P}(n-1). \tag{18}$$

By further simplifying it can be found that

$$\mathbf{k}(n) = \mathbf{P}(n)\mathbf{u}(n)$$

The estimate of weight of least-squares is given as

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{\Phi}^{-1}(n)\mathbf{z}(n)\\ &= \mathbf{\Phi}^{-1}(n)[\lambda\mathbf{z}(n-1)+\mathbf{u}(n)\mathbf{d}^{*}(n)]\\ &= \lambda\mathbf{P}(n)\mathbf{z}(n-1)+\mathbf{P}(n)\mathbf{u}(n)\mathbf{d}^{*}(n)\end{aligned}$$

Substituting the value of $\mathbf{P}(n)$ from eq. 18, we get

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \mathbf{P}(n-1)\mathbf{z}(n-1)-\mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{z}(n-1)+\mathbf{P}(n)\mathbf{u}(n)d^{*}(n)\\ &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)\mathbf{u}^{H}(n)\hat{\mathbf{w}}(n-1)+\mathbf{P}(n)\mathbf{u}(n)d^{*}(n)\\ &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)\mathbf{u}^{H}(n)\hat{\mathbf{w}}(n-1)+\mathbf{k}(n)d^{*}(n)\\ &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)[d^{*}(n)-\mathbf{u}^{H}(n)\hat{\mathbf{w}}(n-1)].\end{aligned}$$

So, the final weight updating equation for RLS algorithm is given as

$$\begin{aligned}\hat{\mathbf{w}}(n) &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)[d^{*}(n)-\mathbf{u}^{H}(n)\hat{\mathbf{w}}(n-1)]\\ &= \hat{\mathbf{w}}(n-1)-\mathbf{k}(n)\varepsilon^{*}(n).\end{aligned}$$

PROCEDURE:

1. Initialize $\delta, 0<\lambda\leq 1$, $\hat{\mathbf{w}}(0)=0$ and $\mathbf{P}(0)=\delta^{-1}\mathbf{I}$ where δ is a large positive constant for small SNR and small positive constant for large SNR.
2. Compute $\mathbf{k}$ matrix given as
$$\mathbf{k} = \frac{\mathbf{P}(n-1)\mathbf{u}(n)}{\lambda+\mathbf{u}^{H}(n)\mathbf{P}(n-1)\mathbf{u}(n)}$$
3. Compute apriori error $\varepsilon(n) = d(n)-\hat{\mathbf{w}}^{H}(n-1)\mathbf{u}(n)$.
4. Update weight matrix $\hat{\mathbf{w}}(n)=\hat{\mathbf{w}}(n-1)+\mathbf{k}(n)\varepsilon^{*}(n)$.
5. Update $\mathbf{P}$ matrix $\mathbf{P}(n)=\lambda^{-1}[\mathbf{P}(n-1)-\mathbf{k}(n)\mathbf{u}^{H}(n)\mathbf{P}(n-1)]$.
6. Repeat steps 2 to 5 for $n=1,2,3,...,N$ where N represents the number of iterations for training weight.
7. Filter the input signal with final obtained weight vector.

RESULTS:

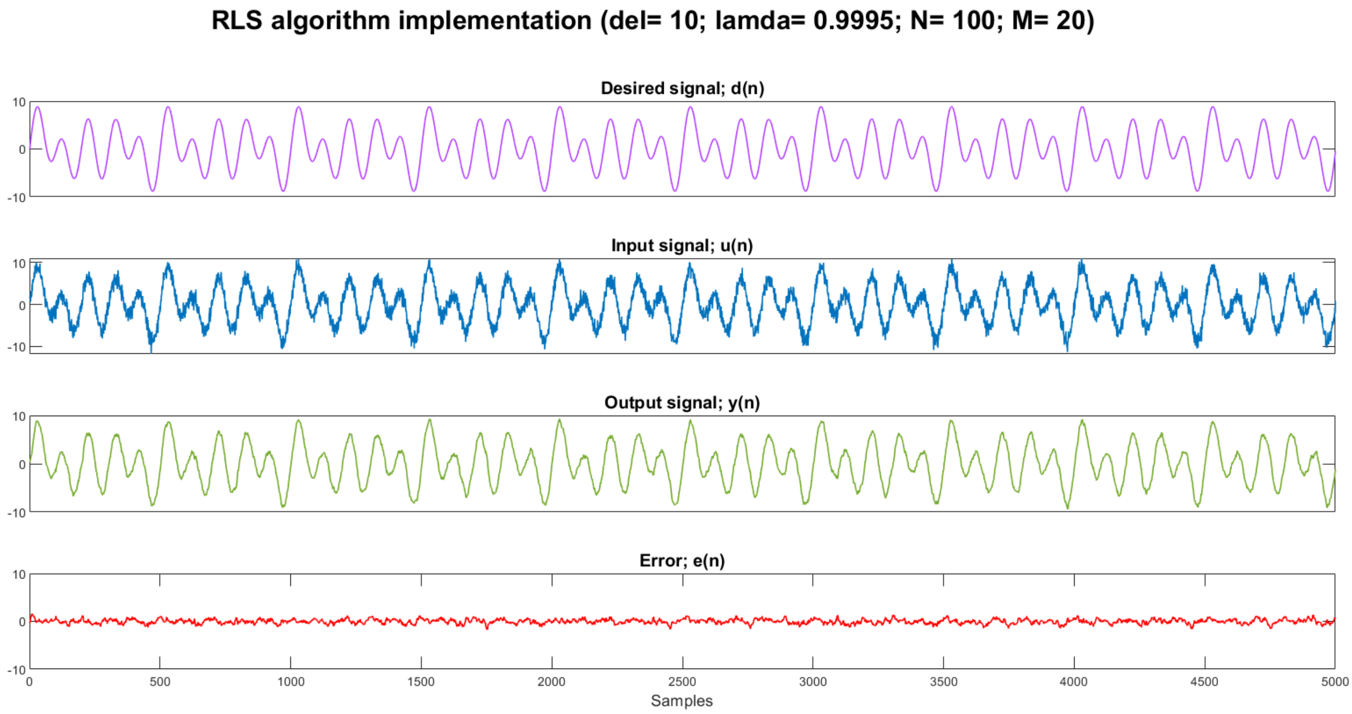

Figure 2: figure shows the desired signal, the desired signal with added noise, the output signal from RLS algorithm and the error between desired and output signal.

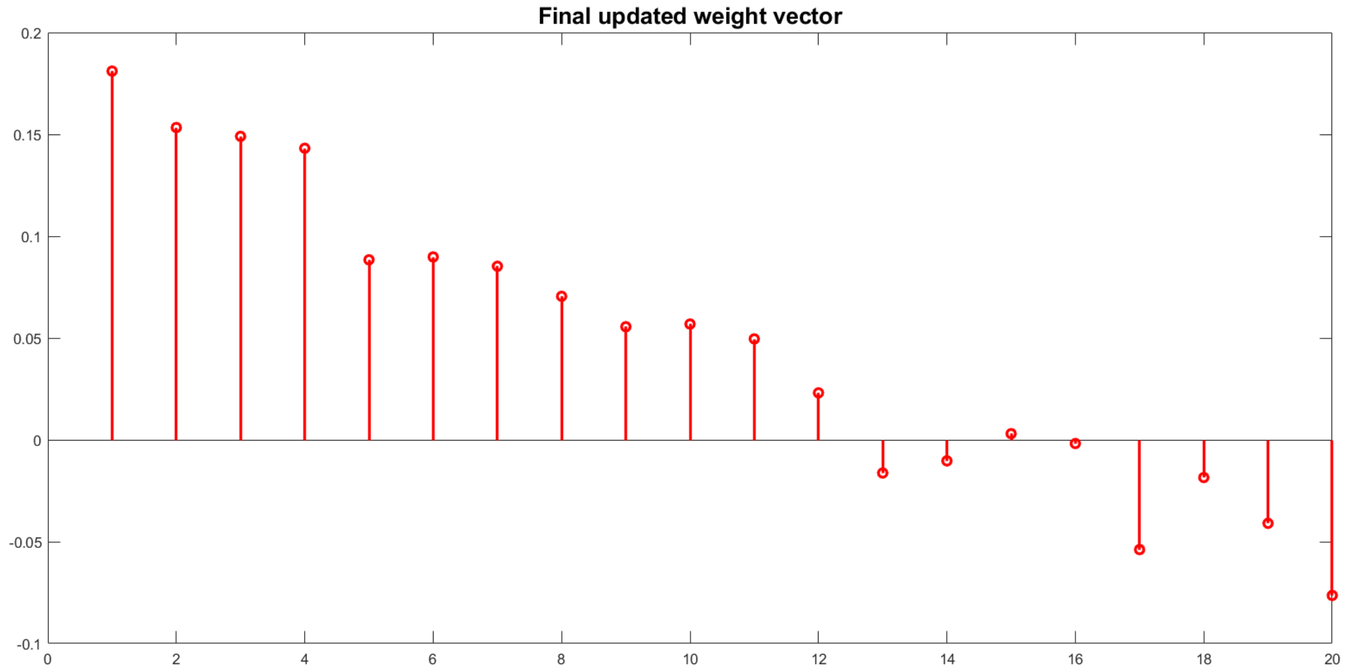

Figure 3: the figure shows the weight vector achieved at the final iteration of RLS algorithm.

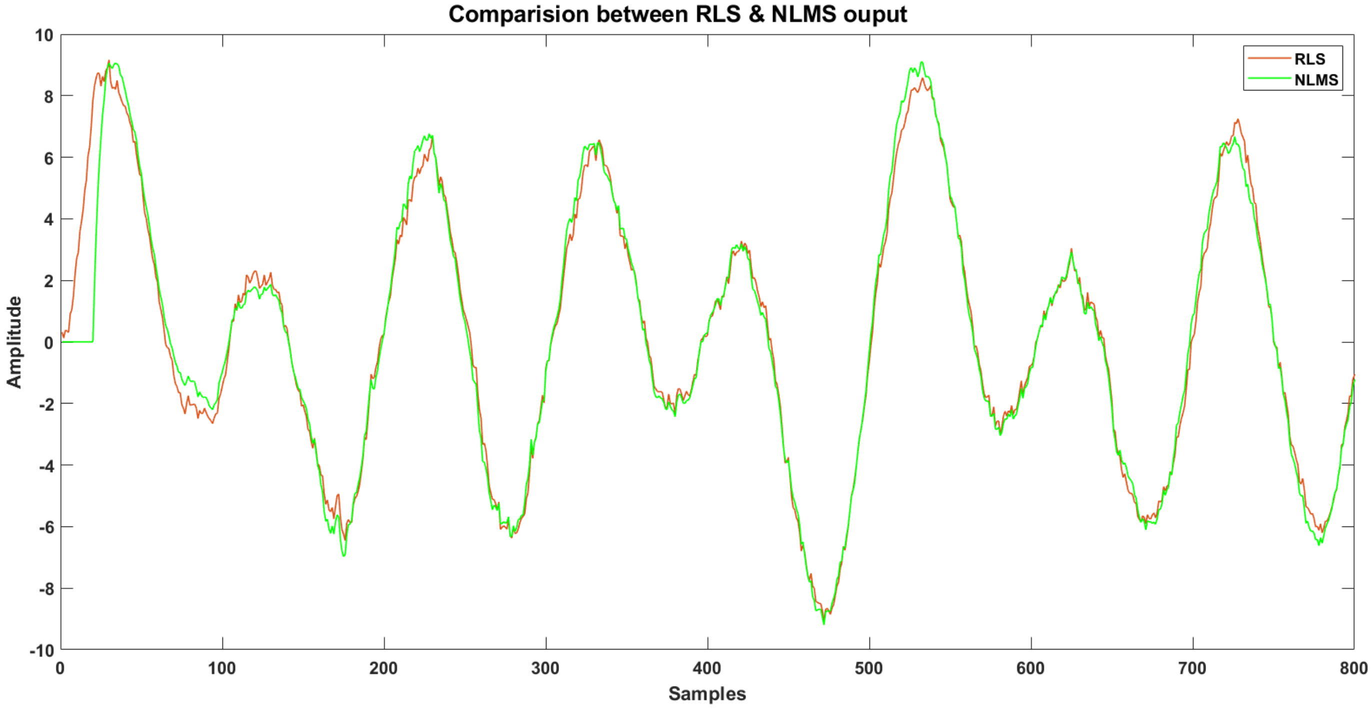

Figure 4: the figure shows the comparison between the output of NLMS and RLS algorithm.

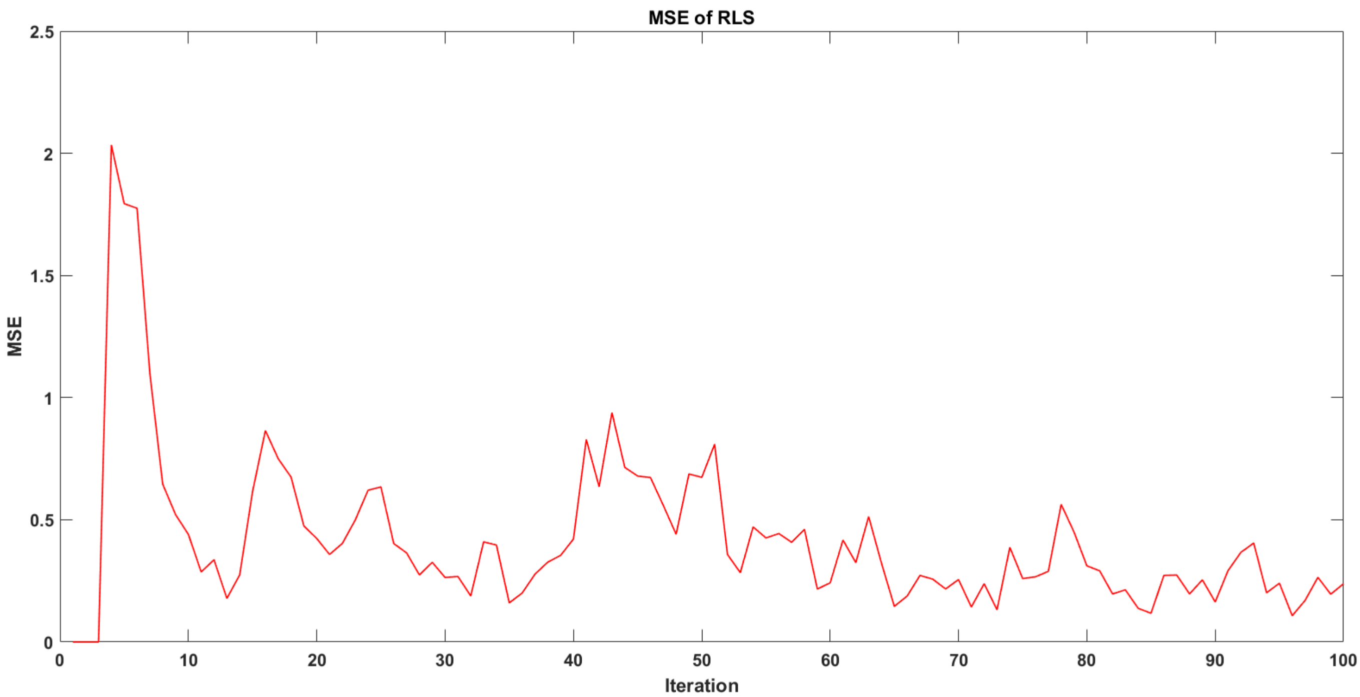

Figure 5: the figure shows the mean square error of RLS algorithm.