Proceedings of the Pakistan Academy of Sciences 52 (1): 67–73 (2015)

ISSN: 0377 - 2969 (print), 2306 - 1448 (online)

Pakistan Academy of Sciences

Research Article

On the Stability of Recursive Least Squares and QR Decomposition Algorithm for Adaptive Filtering Application

F. Z. Okwonu[1,2], and N. A. Atinah[1]

[1] School of Mathematical Sciences, University Sains Malaysia, 11800, Pinang, Malaysia
[2] Department of Mathematics and Computer Science, Delta State University, P.M.B.1, Abraka, Nigeria

Abstract: This paper investigates the stability and tractability of the recursive least squares (RLS) and the QR decomposition for the recursive least squares (QRD-RLS) techniques for adaptive filtering application using the mean weight error norm (MWEN) instead of the conventional mean square error (MSE). The analysis is based on the comparative performance with respect to numerical stability and tractability of incoming signals. The analysis depends on the effect of the different values of the filter coefficient α to these techniques. The simulation indicate that the RLS procedure reduces tractability and is numerically unstable as the value of α increases while the QR decomposition for the recursive least squares showed numerical stability and tractability of incoming signals. The simulation also revealed that the RLS technique produces high misadjument as the filter coefficient increases; the contrary is true for the QRD-RLS technique. In general, the RLS technique converges faster than the QRD-RLS technique.
Keywords: Adaptive Filter, RLS, QRD-RLS

1. INTRODUCTION

Filtering is a signal processing operations with diverse objectives [9]. From mathematical point of view, filtering is a function approximation technique. Adaptive filter may be understood as self modifying digital filter that adjust its coefficients in order to minimize a predefined error function. Adaptive filters are time varying since their parameters are continually changing in order to meet performance specification. Adaptive algorithm minimizes error function which includes data matrix, desired signal and adaptive filter output signal. Adaptive algorithm is applied to adapt the coefficients of the used filter to nonstationary process; the coefficient of the filter is adapted in a process that the error signal is minimized.

Recursive least squares (RLS) is a modified version of conventional least squares problem. When solutions to least squares problem are computed and updated each time new input samples arrive the solution to the system becomes recursive. RLS updates the estimate of least squares minimization problem. The computational procedure of RLS begins with unknown data value or initial condition and applies the new data sample to update the previous data value. RLS is often described as time varying process since its parameters are recursively updated when new sample arrives. RLS recursively update solution to linear least squares filter in which the inverse of the autocorrelation matrix is recursively updated via matrix inversion lemma. The recursiveness of recursive least squares corresponds to adaptive filtering application. RLS solve adaptive filtering problem in order to compute coefficient vector and associated errors recursively. RLS depend heavily on input signal vector. RLS has excellent performance when working in time varying environment than stationary environment. The acceptance of the RLS algorithm has been impeded by unacceptable numerical performance in limited precision environment [13]. For RLS algorithm to operate in time varying environment the forgetting factor λ should be less than one, this allows the RLS algorithm to utilize finite memory. In this regard, RLS has the capability to

Received, March 2013; Accepted, February 2015
*Corresponding author: F. Z. Okwonu; E-mail: fzokwonu_delsu@yahoo.com

track signal variation slowly. When the forgetting factor is less than unity, the adaptive filter coefficient is inconsistent [12]. This process causes noise in the coefficient of the adaptive filter, with result that they become misadjusted from their optimum setting. Due to numerical instability associated with recursive least squares, QR decomposition was proposed [13]. In adaptive filtering, QR decomposition apply time recursive in order to accept input data and desired signal at time instant k. Conventional QR decomposition transform data matrix to orthogonal and upper triangular matrix and also transform desired signal vector. QR decomposition decomposes the data matrix into orthogonal matrix $\mathbf{Q}(k)$ and upper triangular matrix $\mathbf{R}(k)$. In other words we say QR decomposition transform triangular system of equation to a triangular system that uses only transformed matrix [8, 13]. The transformation process via QR decomposition reduces the original data matrix to a reduced form. QR decomposition recursive least squares (QRD-RLS) is a modified version of conventional QR technique for solving conventional least squares problem. QR decomposition technique is numerically robust and stable because of the norm preserving property for the 2- norm [3]. Unlike RLS algorithms and its variant which are highly computationally intensive and also sometimes show some properties of numerical instability [7, 13]. QR decomposition based on Givens rotations approaches the solution to recursive least squares problem by decomposing the data matrix to upper triangular matrix. The upper triangular matrix and the desired signal vector are recursively updated via sequence of Givens rotations in order to compute coefficient vector via backward substitution and subsequently perform rank one update when new input is appended to the upper triangular matrix.

The rest of this paper is organized as follows; Section Two describes the recursive least squares procedure while Section Three contains the modification of QR decomposition for adaptive filtering. The QR decomposition for recursive least squares algorithm (QRD-RLS) is explained in Section Four. Simulation and conclusions are contained in Sections Five and Six, respectively.

2. RECURSIVE LEAST SQUARES

Recursive least squares is a modified version of the conventional least squares problem. RLS updates the estimate of least squares problem, this is done by defining initial conditions and apply the new data value to update previous data value [12]. When solutions to least squares problem are computed and updated each time new input samples arrive the solution to the system becomes recursive. RLS converges fast with low condition number however as the condition number increases RLS produce high misadjustment. RLS has excellent performance when working in time varying environment than stationary environment. The acceptance of the RLS algorithm is sometimes impeded by unacceptable numerical performance in limited precision environment [13]. Our aim is to select the coefficient of the adaptive filter such that the adaptive filter output during the period of observation will correspond to the desired signal, since the minimization process requires information from the input signal to be available. The constant λ is the forgetting factor, this parameter gives more attention to present data values than previous data values during adaptation process [1, 5]. In formulating recursive least squares cost function it is natural to include forgetting factor to the cost function to ensure that the previous data are given less attention in other to concentrate on the new input data. In other word, if the forgetting factor is small the previous data sample contribute very infinitesimal to the system because the forgetting factor is very sensitive to the present data sample than previous data sample [2]. In the context of linear least squares filter the coefficient vector $\mathbf{w}(k)$ is the least squares solution which can be updated recursively. The computation of the inverse autocorrelation matrix $\mathbf{U}^{-1}(k)$ is computationally intensive as such computing the least squares solution directly by computing the inverse of the autocorrelation matrix is not a good practical option [4]. An alternative approach is to apply matrix inversion lemma to update and recursively compute the inverse of the autocorrelation matrix in order to reduce the operations to order $O(n^2)$. Ordinarily, as in conventional least squares by inverting the data matrix $\mathbf{X}$ and multiplying by the desired signal vector $\mathbf{d}$ one obtain the required solution. However, in recursive least squares sense we have to recursively update the inverse of the autocorrelation matrix via matrix inversion lemma [6; 10]. To implement the RLS algorithm, we have to initialize the autocorrelation matrix and coefficient vector before the algorithm

start. In a situation where the time index is smaller than the filter order, this means that the inverse of the autocorrelation matrix does not exist as such we intend to initialize the autocorrelation matrix to obtain

$$\mathbf{U}(i) = \sum_{k=0}^{i} \lambda^{i-k} \mathbf{x}^T(k)\mathbf{x}(k) + \tau\lambda^i \mathbf{I}, \tag{1}$$

where τ is a constant and $\mathbf{I}$ is an identity matrix. Suppose $i = 0$ this implies that equation (1) becomes

$$\mathbf{U}(0) = \tau\mathbf{I}. \tag{2}$$

The constant term in equation (2) is introduced to avoid division by zero and is the only parameter required during initialization [12]. The initialization of the autocorrelation matrix is infinitesimal to the steady state behavior of the recursive least squares algorithm. The algorithm is summarized as follow:

Initialize autocorrelation matrix and coefficient vector

$$\mathbf{U}(0) = \tau\mathbf{I},$$

$$\mathbf{w}(0) = 0.$$

For $k = 1, \quad 2, \quad 3, \quad \cdots,$

compute gain vector

$$\mathbf{h}(k) = \frac{\lambda^{-1}\mathbf{U}^{-1}(k-1)\mathbf{x}(k)}{1+\lambda^{-1}\mathbf{x}^T(k)\mathbf{U}^{-1}(k-1)\mathbf{x}(k)}$$

compute a priori error

$$e(k) = d(k) - \mathbf{w}^T(k-1)\mathbf{x}(k)$$

update coefficient vector

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{h}(k)e(k)$$

compute filter output

$$y(k) = \mathbf{w}^T(k)\mathbf{x}(k)$$

compute a posteriori error

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{w}^T(k)\mathbf{x}(k)$$

update the inverse autocorrelation matrix

$$\mathbf{U}^{-1}(k) = \lambda^{-1}(\mathbf{U}^{-1}(k-1) -$$
$$\mathbf{h}(k)\mathbf{x}^T(k)\mathbf{U}^{-1}(k-1)).$$

3. MODIFICATION OF QR DECOMPOSITION FOR ADAPTIVE FILTERING

In this subsection, we decompose the data matrix with respect to time so that the triangularization process is recursive. Adaptive filtering parameters are continually changing in order to meet specified requirement where such requirement is based on autocorrelation matrix or cross correlation vector between data matrix and desired signal vector. As mentioned earlier, since time varying is recursive in nature we modify conventional least squares problem (LSP) with respect to time such that the data matrix, unknown vector and the desired signal is defined with respect to time. The modify LSP with respect to time translate to

$$\min_{\mathbf{w}(k)} \| \mathbf{X}(k)\mathbf{w}(k) - \mathbf{d}(k) \|_2^2 . \tag{3}$$

Since we understand that $\mathbf{Q}(k)$ and $\mathbf{R}(k)$ are the QR factor of $\mathbf{X}(k)$ and also note that in adaptive filtering problem the data matrix $\mathbf{X}(k)$ and $k \times k$ orthogonal matrix $\mathbf{Q}(k)$ are updated for every system update. For any time instant k with respect to QR decomposition of the data matrix we obtain

$$\mathbf{Q}^T(k)\mathbf{X}(k) = \mathbf{R}(k) = \begin{bmatrix} \mathbf{R}_1(k) \\ \mathbf{0} \end{bmatrix}. \tag{4}$$

Where $\mathbf{R}_1(k)$ and $\mathbf{Q}(k)$ are the upper triangular and orthogonal matrices, respectively. Both the orthogonal and upper triangular matrices are obtained due to the transformation of the input matrix. The objective is to ensure that the input matrix is stable when applied recursively. As new data is received into the system the data matrix becomes $\mathbf{X}(k+1)$ with dimension $(k+1)\times n$ with additional row vector $\mathbf{x}^T(k+1)$ added to the data matrix. As such the new data matrix can be define as

$$\mathbf{X}(k+1) = \begin{bmatrix} \mathbf{X}(k) \\ \mathbf{x}^T(k+1) \end{bmatrix}.$$

If we define

$$\mathbf{Q}(k+1) = \begin{bmatrix} \mathbf{Q}(k) & 0 \\ 0 & 1 \end{bmatrix}, \tag{5}$$

and multiplying the above (5) with the new data matrix we obtain

$$\mathbf{Q}^T(k+1)\mathbf{X}(k+1)=\begin{bmatrix}\mathbf{Q}^T(k) & 0\\ 0 & 1\end{bmatrix}\mathbf{X}(k+1) \\ =\begin{bmatrix}\mathbf{R}(k)\\ \mathbf{Q}^T(k)\mathbf{x}^T(k+1)\end{bmatrix}. \quad (6)$$

In order to obtain upper triangular matrix $\mathbf{R}(k+1)$, the last row of equation (6) needs to be annihilated. Let $\mathbf{T}(k+1)$ be an orthogonal matrix that annihilate the last row of

$$\mathbf{T}(k+1)\mathbf{Q}^T(k+1)\mathbf{X}(k+1)=\begin{bmatrix}\mathbf{R}(k+1)\\ \mathbf{0}^T\end{bmatrix},$$

where $\mathbf{Q}^T(k+1)=\mathbf{T}(k+1)\mathbf{Q}^T(k+1)$ is orthogonal because both $\mathbf{T}(k+1)$ and $\mathbf{Q}^T(k+1)$ are orthogonal. The matrix $\mathbf{T}(k+1)$ can be formed using orthogonalization procedure such as Givens rotation and Householder transformation. It is vital to state that the upper triangular matrix $\mathbf{R}(k+1)$, is henceforth the input matrix and is updated as new input data enter the plant. From here onward we will use Givens rotations for $\mathbf{T}(k+1)$. Updating the dimension of the data matrix from $k\times n$ to $(k+1)\times n$ at the $(k+1)$ th system update means that the $(k+1)$ th orthogonal matrix $\mathbf{Q}(k+1)$ has dimension $(k+1)\times(k+1)$. The above analysis shows the procedure for transforming data matrix to upper triangular matrix with respect to time.

4. QR DECOMPOSITION FOR RECURSIVE LEAST SQUARES ALGORITHM (QRD-RLS)

In previous section, we have described fundamental steps to enhance efficient computation of coefficient vector by forming triangular system of equation thereby paving way for the solution to the triangular system of equation which can be solved using backward substitution [4; 11]. Appending input vector and applying sequence of Givens rotations to annihilate appended input vector except the last row we obtain

$$\mathbf{Q}^T(k)\mathbf{X}(k)=\mathbf{Q}^T(k)\begin{bmatrix}\lambda^{1/2}\mathbf{X}(k-1)\\ \cdots\cdots\cdots\cdots\\ \mathbf{x}^T(k)\end{bmatrix}$$

$$=\mathbf{T}(k)\begin{bmatrix}\mathbf{Q}^T(k-1) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\lambda^{1/2}\mathbf{X}(k-1)\\ \cdots\cdots\cdots\cdots\\ \mathbf{x}^T(k)\end{bmatrix}$$

$$=\mathbf{T}(k)\begin{bmatrix}\lambda^{1/2}\mathbf{R}_1(k-1)\\ \mathbf{0}\\ \mathbf{x}^T(k)\end{bmatrix}=\begin{bmatrix}\lambda^{1/2}\mathbf{R}_1(k-1)\\ \mathbf{0}\\ \mathbf{0}^T\end{bmatrix}. \quad (7)$$

The matrix $\lambda^{1/2}\mathbf{R}(k-1)$ is the upper triangular matrix scaled by the square root of the forgetting factor.

QRD-RLS Algorithm

For each k,

Initialize:

$\mathbf{w}(0)=\mathbf{0}$,

$\mathbf{X}(0)=\delta\mathbf{I}$.

Compute the transformation matrix

$$\mathbf{T}_\theta(k)= \quad \mathbf{T}'_{\theta n}(k) \quad \mathbf{T}'_{\theta n-1}(k)= \\ \cdots \quad \mathbf{T}'_{\theta 2}(k) \quad \mathbf{T}'_{\theta 1}(k)=\prod_{i=1}^{n}\mathbf{T}_{\theta i}(k).$$

Apply $\mathbf{T}_\theta(k)$ to update $\mathbf{R}_1(k)$ recursively

$$\mathbf{T}_\theta(k)\begin{bmatrix}\lambda^{1/2}\mathbf{R}_1(k-1)\\ \mathbf{x}^T(k)\end{bmatrix}=\begin{bmatrix}\mathbf{R}_1(k)\\ \mathbf{0}^T\end{bmatrix}$$

compute and update the transformed desired signal vector

$$\begin{bmatrix}\mathbf{q}(k)\\ \mathbf{s}(k)\end{bmatrix}=\mathbf{T}_\theta(k)\begin{bmatrix}\lambda^{1/2}\mathbf{q}(k-1)\\ q(k)\end{bmatrix}$$

compute and update the coefficient vector recursively

via backward substitution

$$\mathbf{w}(k)=\frac{\mathbf{q}(k)-\sum_{i=1}^{n}\mathbf{R}_1(k)\mathbf{w}(k)}{\mathbf{R}_1(k)}$$

compute the cosine term

for each $i=1:n$

$$\psi(k) = \prod_{i=1}^{n} \cos\theta_i(k)$$

compute and update the a posteriori error and a priori error recursively

$$\varepsilon(k) = \mathbf{q}(k) \times \psi(k)$$

$$\hat{e}(k) = \frac{\mathbf{q}(k)}{\psi(k)}$$

end

If necessary compute and update the errors recursively using coefficient vector a priori error

$$e(k) = \mathbf{d}(k) - \mathbf{w}^T(k-1)\mathbf{x}(k)$$

a posteriori error

$$\mathbf{e}(k) = \mathbf{d}(k) - \mathbf{w}^T(k)\mathbf{x}(k)$$

5. SIMULATION

Simulation is performed using system identification with transfer function in which the plant is defined as the transfer function (see Figure 1. below) to identify the adaptive filter coefficient with the optimal coefficient. In this regards the length of both parameters are equal. The input signal $x(k)$ passes through the linear filter. Where

$$g(z) = (1-\alpha^2)^{1/2} / (1-\alpha z^{-1}),$$

is the transfer function and is applied to produce the desired signal vector $d(k)$; the signal to noise ratio (SNR) is 30dB. Based on Figure 1, the unknown plant is assumed linear. This expression revealed that α is an input to the system function, α is a real valued constant in the interval (0,1). The system function and the unit variance white noise $\omega_{\rho(k)}(z) = 1$ are applied as input to generate the autocorrelation function $\omega_{xx}(z) = g(z)g(z^{-1})$ $\omega_{\rho(k)}(z) = 1-\alpha^2 / (1-\alpha z^{-1})(1-\alpha z)$ The input signal $x(k)$ is defined as

$$\mathbf{U} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{K-1} \\ \alpha & 1 & \alpha & \ldots & \alpha^{K-2} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ \alpha^{K-1} & \alpha^{K-2} & \alpha^{K-3} & \ldots & 1 \end{bmatrix}.$$

The input signal $x(k)$ is generated based on the normal distributed random numbers (randn) which is further applied to the plant. The plant consist of the transfer function g(z). The adaptive filter is applied to identify the transfer function. Observe that the autocorrelation matrix consist of one in the diagonal and alpha α elsewhere. The α is applied to determine the condition number of the autocorrelation matrix. The performance of these techniques strictly depends on the values of the filter coefficient α. The filter coefficient is used to investigate the stability, tractability and adaptability of the techniques, which implies that the filter coefficient is user defined. The different values of the filter coefficient otherwise called forgetting factor are 0.01, 0.5, 0.75 and 0.99 [9].

To generate the data matrix, the following input signal generating procedure, say are applied. The mean used is zero with white Gaussian noise added to the desired signal; the variance is $\sigma = 0.001$. The filter coefficient α stated above were applied to investigate the performance of the algorithms. The mean weight error norm (MWEN) is obtained by taking the ensemble average of the weighted error norm. Instead of using the conventional MSE, the MWEN was applied to determine stability and tractability of the technqiues. The comparison between the two methods (QRD-RLS and RLS) showed that the QRD-RLS is numerically stable than the RLS. The Simulation also indicated that as the filter coefficient increases the RLS reduces tractability thereby producing high misadjustment. On the other hand, QRD-RLS remain stable even though the filter coefficient increases. The figures below showed that QRD-RLS algorithm using mean weight error norm is numerically stable and it's ability to track incoming signals is consistent and robust compared to the RLS procedure.

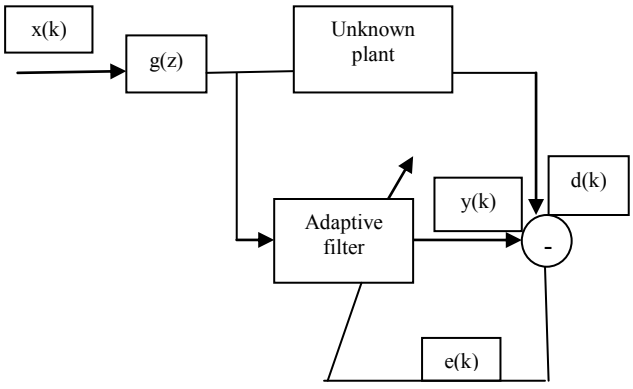

Fig. 1. System identification with transfer function.

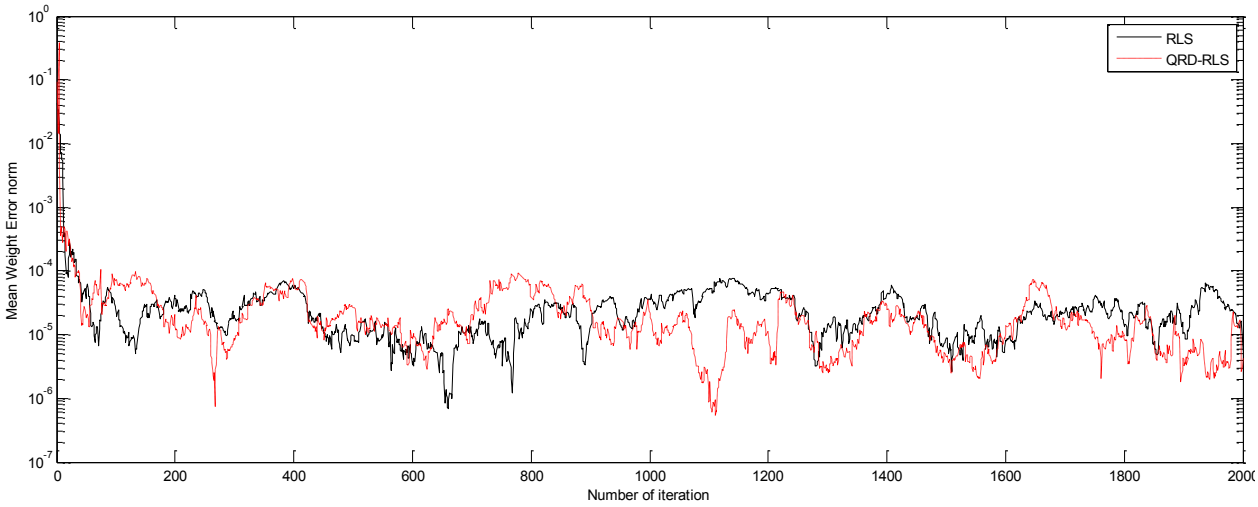

Fig. 2. $\alpha = 0.01$.

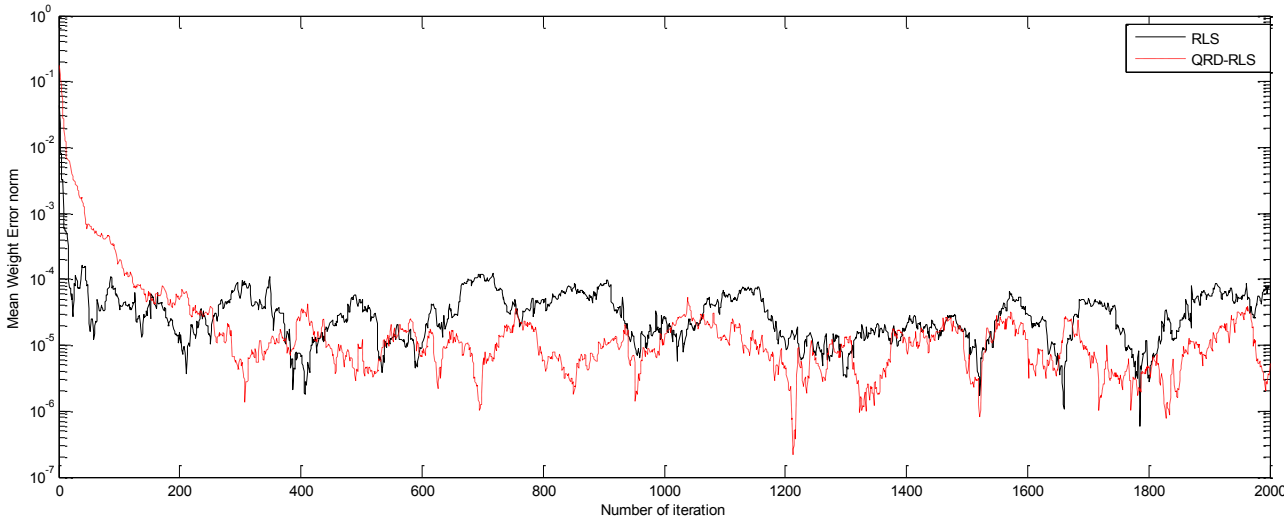

Fig. 3. $\alpha = 0.5$.

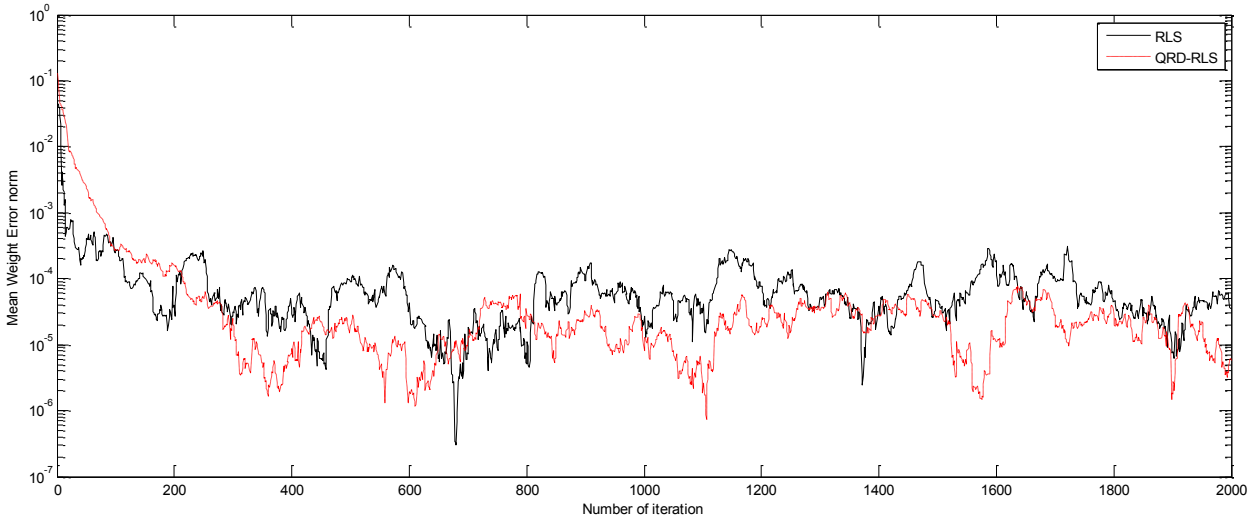

Fig. 4. $\alpha = 0.75$.

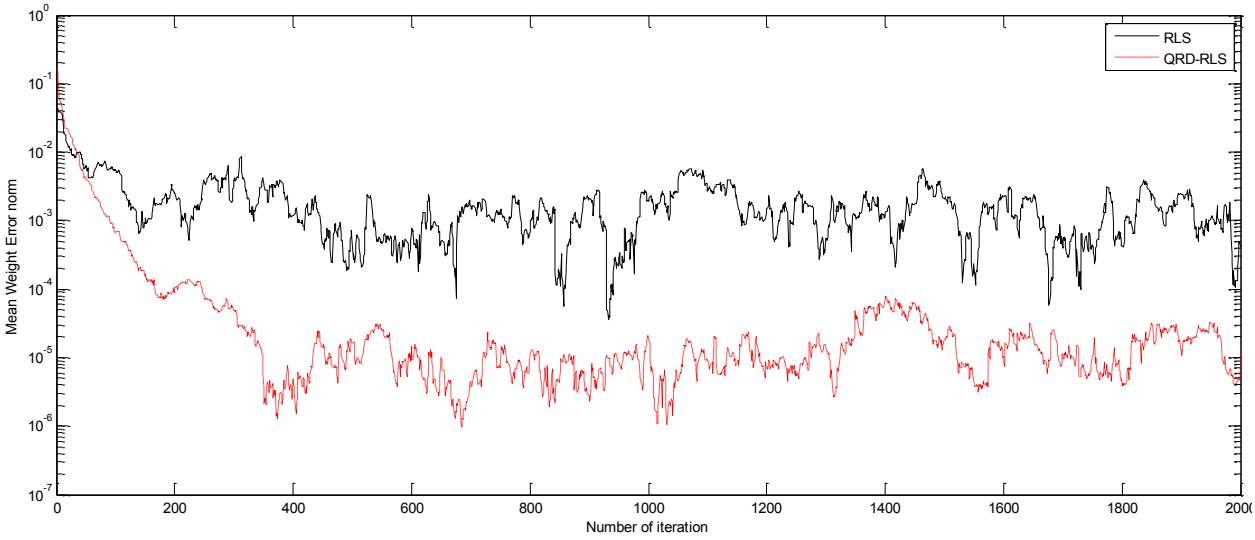

Fig. 5. $\alpha = 0.99$.

CONCLUSIONS

This study revealed the transformation of the data matrix to upper triangular matrix via sequence of Givens rotations. Unlike RLS approach the data matrix is updated via conventional matrix inversion lemma however for the QRD-RLS approach the triangularized upper triangular matrix is updated via sequence of Givens rotations. We performed Simulation to compare the stability and tractability of both algorithms. Result from simulations showed that the QRD-RLS is numerically stable, tractable with respect to incoming signal. In general, the QRD-RLS techniques is robust and has the capability of accepting incoming signals compared to RLS as shown in the figures above. The RLS approach converges faster, though as the condition number of the data autocorrelation matrix increases RLS become numerically unstable and is incapable of tracking incoming signals; on the other hand QRD-RLS is stable with large condition number.

6. REFERENCES

1. Apolinario, J.A., Jr., & D.M. Maria. *QRD-RLS Adaptive Filtering*. Springer book (2009).
2. Apolinario, J.A., & N.L. Sergio. *Introduction to Adaptive Filter*. Springer Science (2009).
3. Okwonu, F.Z. QR decomposition for adaptive filtering application. Research Work (2011).
4. Okwonu, F.Z., & N.A. Ahmad. Angle parameter loaded sample matrix inversion based on inverse QRD-RLS. *Global Journalo Pure and Applied Mathematics* 1: 17-29 (2010).
5. Farhang, B.B. Adaptive filters theory and applications, John Wiley & Sons (1999).
6. Woodbury, M.A. *Inverse Modified Matrices*. Memo Rep. 42, Statistical Research Group, Princeton, NJ (1950).
7. Chen, M.Q. A direction set based algorithm for least squares problems in adaptive signal processing. *Linear Algebra and Its Applications* 284: 73-94 (1998).
8. Ahmad, N.A., & F.Z. Okwonu. Least squares problem for adaptive filtering. *Australia Journal of Basic and Applied Sciences* 5: 69-74 (2011).
9. Okwonu, F.Z. Comparing the stability of recursive least squares and QR decomposition algorithm for adaptive filtering application. *Proceedings of 2nd Regional Conference on Applied and Engineering Mathematics* (RCAEM-II), p. 567-571 (2012).
10. Regalia, R.B.L. QRD-RLS *Adaptive Filtering*. Springer Science +Business Media, LL (2009).
11. P. Regalia, and Le Borne, R., QRD-RLS. *Adaptive Filtering*. Springer Science +Business Media, LL, (2009).
12. S. Haykin. *Adaptive Filtering Theory*. Prentice Hall, Inc. Englewood Cliffs, New Jersey (1991).
13. S. T. Alexander and A. L. Ghirnikar, A method for recursive least squares filtering based upon an inverse QR decomposition. *IEEE Transactions Signal Processing* s41: 20-30 (1993).