Implementation of RLS Algorithm

H.V.Ajmera and M.Z.Ahmed

Fixed and Mobile Communications, University of Plymouth, Plymouth, UK
e-mail: M.Ahmed@plymouth.ac.uk

Abstract

An algorithm for recursively calculating the least square solution to adaptive filter, for its implementation on a dsPIC is described in this paper. It shows an unified approach for deriving the fixed-point Recursive Least Square (RLS) algorithm to estimate the optimal set of filter coefficients. The objective is to study implementation effects on the Recursive Least Square algorithm used in the mean-square manner for adaptive filter for a dsPIC. Using the matrix operation as well as the round-off error model, the effects of machine implementation on RLS algorithm has being found out theoretically confirmed by simulation results. The main sources of errors can be thought of: division operation involved in updating filter variables. Due to such errors, the digitally implemented algorithm shows a divergence rather than convergence. A couple of solutions are provided to give better performance, so that the simulated results for filter coefficients for fixed and floating point are found to be similar.

Keywords

RLS algorithm, Finite Precision Effect, Error Analysis

1 Introduction

In recent years, recursive least square (RLS) filters have emerge as a powerful tool for adaptive filtering, prediction and identifying (Haykin, 1996). However, when implemented in a finite precision environment, RLS algorithm can suddenly become unstable and also divergence becomes a problem. A number of papers have being published that deals with the effects that finite word length has on the RLS algorithm (Adali and Ardalan, 1987; Ardalan, 1986; Bottomley and Alexander, 1991). Its always being a difficult issue to analyses such algorithm due to recursive nature of this filter. Consequently, certain simplifications have being made in order to yield useful results. One simplifying approach is to assume that kalman gain is available with finite precision range (Adali and Ardalan, 1987; Ardalan, 1986). A second approach was to bias the round-off error in each filter quantity to achieve a more stable performance (Bottomley and Alexander, 1991). It has also being shown that the prewindowed growing memory RLS algorithm is unstable (Adali and Ardalan, 1987), while other assume that some of the term didn't contribute to the error. All the elements of matrices and vectors in the RLS algorithm will deviate from their correct values due to quantization effects, giving three separate effects: 1) The inverse autocorrelation matrix may become indefinite due to accumulation of the error, 2) Propagation of the error due to recursive use of the algorithm and 3) With use of exponentially forgetting factor, the errors also grows exponentially. The output signal is computed by convolving the input sample with tap coefficients and the weight vector are updated by taking the product of kalman gain with the predicted

error. In this paper, there are no simplifications made. Specifically, 1) Quantized input signal and desires signal are available, 2) Neglect the second order noise term if the magnitude is smaller than 1. The paper is organised as: In section II, the infinite precision RLS algorithm is being discussed, while in section III, fixed-point error are injected into the algorithm with results being reviewed in section IV.

2 RLS Algorithm

The basic aim of the least square algorithm is to minimize the sum of the squares of the difference between the desired signal r(i) and the filter output y(i). An important characteristic of RLS is the computation of the estimate of the inverse correlation matrix from the input data u(i), which helps the minimization process. The adapted coefficient $h_n(i)$, n=0,1,2…N aim at minimizing the given objective function. In the case of the least square method, the objective function is given by

$$\varepsilon(n) = \sum_{i=0}^{n} \lambda^{n-i} \left| e(i) \right|^2 \tag{2.1}$$

$$\text{where, } e(i) = r(i) - y(i) \tag{2.2}$$

and λ is forgetting factor. To find optimum values of the tap-weight vector h(n), it necessary that $\boldsymbol{\varepsilon}(n)$ achieves its minimum value. This is done by taking partial derivative with respect to the tap coefficients h_n. The optimum value for the filter coefficient is obtained when the partial derivative is set to zero. The resulting expression is given by:

$$hn = R_u^{-1}(n) .rdu(n) \tag{2.3}$$

where, $R_u(n)$ and $r_{du}(n)$ are auto correlation of the u(n) and cross correlation matrix between u(n) and r(n) respectively. Using the matrix inversion lemma, and defining term kalman gain, the inverse of the autocorrelation matrix is given by

$$P(n+1) = R_u^{-1}(n+1) = \frac{1}{\lambda} [P(n) - k(n+1)\, uT(n+1).P(n)] \tag{2.4}$$

Using eq. (2.2), (2.3) and (2.4) we develop a recursive solution for updating the least square estimate h(n) for the tap weight at iteration n as follows:

$$hn+1 = h(n) + k(n+1)\, e(n+1) \tag{2.5}$$

Thus, we can conclude that the new estimates of the tap coefficients are calculated based on the inner product of the old estimate and the current input sample.

3 Fixed-Point Implementation of RLS Algorithm

In this section, a fixed point analysis approach is developed to analyze RLS algorithm. The approach develop is clear- model all the round-off errors resulting from fixed-point implementation and depending on this model, build up exact recursive equation for total error in the algorithm. A typical approach towards the modeling is the assumption that addition and subtraction do not introduce any round-off error and this holds true as long as there is no overflow. Thus, for most of cases, the culprit left are multiplication and division, as the source of round-off error in fixed implementation of RLS algorithm. The round-off error in the product of a multiplication can be expressed as

$$f[x \cdot y] = xy + \xi_{xy}$$

where xy is the infinite precision product, ξ_{xy} is relative error and is independent of x,y and also of x·y. Similar result do exist in case of division.

The error term e'(i) consists of the difference between a desired filter output r(i) and the fixed-point output of a filter y'(i), where e'(i) is the fixed-point error term. Here h'(n) denotes fixed-point weight coefficients.

$$e'(i) = r(i) - y'(i) = r(i) - h'_n \, uT(i) - \mu(i) \tag{3.1}$$

The optimum value of weight tap coefficients is found out in a same way as done for conventional RLS algorithm and for fixed-point RLS is given by:

$$rdu(n) - \mu du(n) = Ru(n).h'n \tag{3.2}$$

where, $\mu_{du}(n)$ is cross correlation matrix between u(i) and error μ(n). As from the fixed-point model defined previously, the multiplication and division may result in quantization error, denoting the fixed-point kalman gain by **k**'(n + 1) while β(n) as the fixed-point error, the kalman gain is expressed as:

$$k'(n + 1) = k(n + 1) + \beta(n) \tag{3.3}$$

Similarly, the error introduced in the calculation of the inverse autocorrelation matrix is ω(n) and P'(n + 1) as the inverse auto correlation matrix using fixed-point RLS algorithm, then

$$P'(n + 1) = \frac{1}{\lambda} [P(n) - k'(n + 1) \, uT(n + 1).P(n) + \beta(n) \, uT(n + 1).P(n) + \omega(n)] \tag{3.4}$$

Once the fixed-point inverse autocorrelation is calculated, we can then use eq. (2.5), (3.2), (3.3) and (3.4) to update the weight vector by recursive sequence.

$$h'(n + 1) = h'(n) + k(n + 1)e'(n + 1) - \alpha NN(n)h'(n) \tag{3.5}$$

where, $\alpha_{NN}(n) = \beta(n)\, u^T(n+1).P(n) + \omega(n)$. This is the theoretical expression for the fixed-point weight error vector.

4 Experimental Analysis

The section below provides the effect of fixed-point errors on the performance of the RLS algorithm through simulation results. The programs were stimulated in C and MPLAB. The algorithm is based on the equations (3.1), (3.3), (3.4) and (3.5). In the calculation, the inverse autocorrelation matrix is initialized as $P(0) = \delta^{-1}I$, where, $\delta = 0.005$ which ensure that $P^{-1}(n)$ is a positive definite matrix. All the other vectors except the input and the desired response are set to zero. The desired response to the algorithm was a sine wave, while the input response consists of delayed version of the sine wave added with a sine wave of different frequency. The table shown below is calculated for different forgetting factor lambda λ.

Iteration	λ=0.1	λ =0.5
0	0.0000	0.0000
1	0.0000	0.0000
2	0.0371	-0.0413
3	-0.013	0.4645
4	-0.069	0.3786
5	-0.2171	0.4446
6	-0.0514	-0.328
7	-0.0706	0.2280
8	-0.7693	-0.1528
9	-0.395	0.3817
10	-1.233	-0.1596

Table 1: Comparison of μ(i) for different λ

With larger value of λ, the magnitude of error μ remains less than 0.6 and its value remains more stable, although the filter continues to be unstable due to the higher value of μ. The smaller value of λ makes the filter more unstable as compared to higher values of λ, and also sets the maximum value of μ(i). This smaller value of λ has benefit of offering the least value of μ(i). With the smaller value of forgetting factor, the maximum value shown is almost the double of its counterpart.

One of the most important filter quantity is the kalman gain, as the value of k(n) affects all the filter quantities directly. This can be seen from eq. (3.4) used to calculate the inverse of auto correlation matrix and from eq. (3.5), where current tap coefficient vector uses the value of kalman gain. Hence, quantization in kalman gain k(n), would amplify the total error in the filter and severely affect the stability of the algorithm. To see how the errors are manifested in practice, we show the results of computer experiment.

Table 2 shows results for different value of λ at iteration 2. As observed from above, the error vector β(n) increases as the value of the λ is being increased from 0.1 to 0.5.

The variation in the error for the case when λ=0.1 is very small, while the simulation results shows a tremendous growth in the error β(n) for the latter case. This means the magnitude of β could be said to be proportional to λ.

With the presence of inverse autocorrelation, the successive taps becomes decorrelated in RLS algorithm, making the algorithm self-orthogonalizing. The update value of P(n) is calculated using the previous value of P(n), the product of kalman gain and the tap inputs. Table 3 below shows the error ω(n).

λ=0.1	λ=0.5
Iteration 2	
$\begin{bmatrix} 1.9711 \\ -0.8482 \\ 0.0673 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 3.2636 \\ 5.3929 \\ 12.7649 \\ 0 \end{bmatrix}$

Table 2: Comparison of β(n) for different λ

λ=0.1	λ=0.5
$\begin{bmatrix} -2.4592 & -0.5371 & 2.218 & 0 \\ -0.6029 & -0.0188 & 3.8376 & 0 \\ 2.5396 & 4.6352 & 9.7435 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -3.0142 & -1.5050 & 0.2129 & 0 \\ -1.4353 & -1.4719 & 0.8287 & 0 \\ 0.3748 & 0.7289 & 3.7256 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

Table 3: Comparison of ω(n) for different λ

As the autocorrelation matrix in symmetric, so is the case with the error in it. The values of ω(n) also follows this symmetric property. As seen from the table, ω(n) shows maximum value for the minimum value of forgetting factor λ.

Having calculated all the error in each filter variables, the final step will be to calculate the total error in the tap coefficients affected due to the errors present in the filter variable. In this sense, the errors generated in the filter vectors and matrices are multiplied by different filter variables to produce additional errors. In the eq. (3.5), there is an error $\alpha_{NN}(n)$ which is given as:

$$\begin{bmatrix} 4.078 & 2.5076 & -5.5667 & 0 \\ 10.4329 & 5.1587. & -9.7086 & 0 \\ 27.2169 & 13.9655 & -17.2931 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 4: Error $\alpha_{NN}(n)$

The magnitude of the error produce is of order of 10. An error with such magnitude definitely tends to make the filter highly unstable and also losses the convergence property for which the adaptive filter are famous for. This error is then multiplied by the current tap coefficients.

Table 5 shows the amount of the error which is added to the previous tap coefficients to give the new coefficients. In other words, this is the error by which the new taps coefficient will be shifted. One peculiar characteristics of this error is the dependence on the previous tap coefficient as seen in eq. (3.5). In short, the tap coefficients are updated as addition of previous tap coefficients with the product of kalman gain and error e(n), with the addition of the product $\alpha_{NN}(n)h'(n)$.

$$\begin{bmatrix} 3.0379 \\ 4.6637 \\ 6.0451 \\ 0 \end{bmatrix}$$

Table 5: Total Error in Tap Coefficient

There are two problems which can be associated with the error result found out above: Firstly, a fixed-point implementation does not guarantee to prevent the overflow and underflow, a problem which results in nothing but divergence. Secondly, the stability may not be guaranteed. The problem can be solved by preventing the overflows. This can be done either by using floating-point, not possible in our case or by modifying the code. From a designer point of view, a careful fixed-point implementation of the filter is to be developed, so that overflow and underflow are minimized. Secondly, the filter is to be made stable. The following are the way to minimize the error:

- Changing the value of forgetting factor λ

The minimum value for error in β can be obtained when the value of λ is taken to be minimum shown in table 2. Successively, as seen from table 3, with minimum value of forgetting factor the error ω reaches a value of 10 in the first few iteration only. The minimum value for this error can be obtained when the value of the λ is maximum. Successively, as a designer there has to be a trade-off made in selecting the value of the forgetting factor.

- Reducing the value of k by factor of $\frac{1}{2}$

Changing the value of k to $\frac{k}{2}$ and by keeping the value λ=0.5, has the advantage of decreasing the error β and also compensates the error in the auto correlation matrix.

With the above solution in the mind, let us recalculate the errors in different term. Figure 1 below shows the stimulation result for the average mean square error μ(i) against the number of iteration for the original value of kalman gain k(n). This situation sets the maximum value of the average mean square error μ(i), which comes to be more than 2. However, for most of the iteration the error value remains close to 1. Accordingly, this indicates a huge amount of error in the least square solution, which results in divergence. As seen form the graph, there is continuously rise and fall in the value, with a sudden rise reaching a peak value of about 2.4, indicating the instability of the filter.

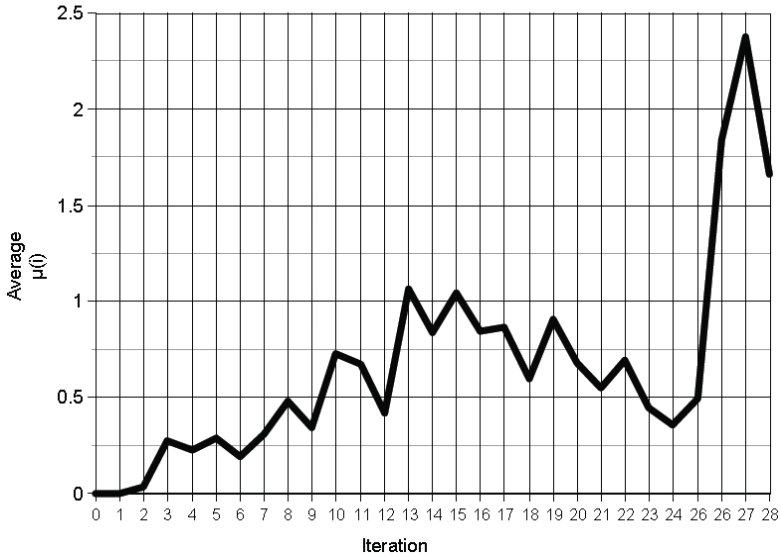

Figure 1: Average Mean Square Error μ(i)

Now let us analyze the graph when the value of the kalman gain is reduced by 2, so that the input to the filter is now $\frac{k(n)}{2}$. It can clearly seen in figure 2 that the value of the average mean square error μ(i) has decreased from the maximum value of 2.4 to a value just less than 1.77. Consequently, the graph seen is more stable and tries to maintain a similar value of 0.75, but at certain iteration the value is doubled then its previous iteration. There is peculiar similarity about both the graphs- at a point there is sudden linear rise followed by liner fall and again linear rise.

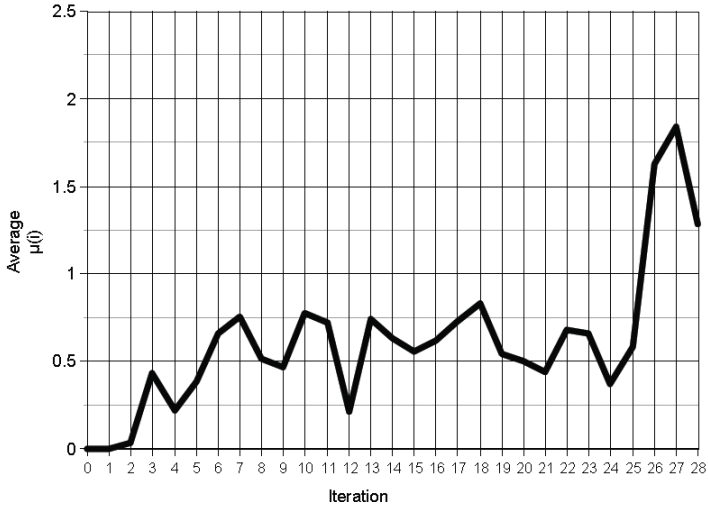

Figure 2: Average Mean Square Error μ(i) for scaled kalman gain

As seen from Figure 2, the error in the μ(i) has decreased. The error in kalman gain β(n) as shown decreases due to scaling and as shown in table 3, with increase in value of forgetting factor the error ω(n) decreases. With these details, the error $\alpha_{NN}(n)$ is calculated as shown in table 6.

When this calculated value of $\alpha_{NN}(n)$ is compared with the previous value, one can easily state that the error had reduced to almost half of the previous one. Now let us compute the total error due to which the current estimate tap coefficients are shifted. This is achieved in the same way as earlier by multiplying the error shown below with the previous tap coefficients.

$$\begin{bmatrix} -0.235 & 3.9352 & -1.867 & 0 \\ 7.8091 & 1.4737 & -3.1074 & 0 \\ 15.7651 & 10.0565 & -7.4675 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Table 6: Error $\alpha_{NN}(n)$ for scaled kalman gain

There is tremendous decrease in the total error as seen in table 7, when matched against the outcome of the simulation performed without the correction in the program. This may due to the reason that the error in the previous coefficient vector may be less as well as the other error has decreased. Performing similar scaling of various filter quantities may help the designer to obtain convergence or convergence may be possible with the above solution after may be 100's of iteration or the error value will be so less that the magnitude may be less than 1.

$$\begin{bmatrix} 0.4504 \\ 0.9807 \\ 1.1111 \\ 0 \end{bmatrix}$$

Table 7: Total Error in Tap Coefficient for scaled kalman gain

5 Conclusion

A surprising divergence is observed on implementing the algorithm on a fixed-point processor. From the mathematical expressions, we can say that the least square solution includes a huge amount of error, confirmed with the stimulated results. From the theoretical expression, it is shown that in equation for optimum filter coefficients, the only error present is the correlation matrix of μ(i) and tap inputs u(i). The expression for least square solution is same as that for infinite precision with finite precision least square solution being the result due to arithmetic operation carried on the finite precision filter quantities plus the error multiplied by the old estimate of tap coefficient. It was shown that the error contribution of certain fixed-point operations increased as the forgetting factor increases while errors due to other operations decreased. The cause of error in some of the variables has being identified, pointing towards the division operation carried out to calculate these

terms. With the above facts, it forces the designer to make trade-off in the value of forgetting factor λ. The value of lambda could be taken equal to 0.5. The β(n) will still show a higher error value. Consequently, the values in kalman gain k(n) should be then reduced by a factor of ½. With this solutions, the error decreases which is shown in table 7.

6 References

Adali, T. and Ardalan, S. H. (1987). “Fixed-point roundoff error analysis of the exponentially windowed RLS algorithm for time-varying systems.” Proceedings of the Acoustics, Speech, and Signal Processing, 1987. ICASSP-87., 1987 International Conference on.

Ardalan, S. (1986). "Floating-point error analysis of recursive least-squares and least-mean-squares adaptive filters." Circuits and Systems, IEEE Transactions on 33(12): 1192-1208.

Bottomley, G. E. and Alexander, S. T. (1991). “A novel approach for stabilizing recursive least squares filters.” Signal Processing, . IEEE Transactions on 39(8): 1770-1779.

Haykin, S. S. (1996) *Adaptive filter theory,* Upper Saddle River, N.J., Prentice Hall.