A Computationally Effcient Leaky and Regularized RLS Filter for Its Short Length

LETTER

A Computationally Efficient Leaky and Regularized RLS Filter for Its Short Length

Eisuke HORITA[†a)], ***Senior Member***

SUMMARY A Tikhonov regularized RLS algorithm with an exponential weighting factor, i.e., a leaky RLS (LRLS) algorithm was proposed by the author. A quadratic version of the LRLS algorithm also exists in the literature of adaptive filters. In this letter, a cubic version of the LRLS filter which is computationally efficient is proposed when the length of the adaptive filter is short. The proposed LRLS filter includes only a divide per iteration although its multiplications and additions increase in number. Simulation results show that the proposed LRLS filter is faster for its short length than the existing quadratic version of the LRLS filter.

key words: *adaptive filter, RLS, computational complexity, regularization, Tikhonov*

1. Introduction

Regularization in least-squares estimation provides a compromise between a bias of the solution and an algorithmic stability. The standard RLS algorithm with an exponential weighting factor has the regularization matrix which fades exponentially to zero [1]. As a result, numerically robust RLS variants, such as the Square-Root RLS (SR-RLS) [1], [2], are not suitable for applications where maintaining regularization throughout the adaptive process is important, such as adaptive beamforming [3]. The RLS filters by using a nonfading regularization matrix are known as a Levenberg-Marquardt regularization [4] or a Tikhonov [6] regularized RLS algorithm with an exponential weighting factor, i.e., a leaky RLS (LRLS) algorithm [7] proposed by the author, which was mentioned in [8] by Waterschoot et al. In [8], a quadratic version of the LRLS algorithm with the regularization method in [4] was introduced, and a leaky kernel affine projection algorithm was also proposed in [9]. In addition, the quadratic version of the LRLS algorithm was applied to an adaptive filter in [10] in order to track a time-varying regularization parameter of the correlation matrix of [10] in [11].

The quadratic version of the LRLS algorithm is more useful than a cubic version of the LRLS algorithm in [7]. However, it includes two divides per iteration although the standard RLS algorithm in [1], [2] has one divide per iteration.

In this letter, a cubic version of the LRLS filter which is computationally efficient is proposed when the length of the adaptive filter is short. The proposed LRLS filter includes only a divide per iteration although its multiplications and additions increase in number. Simulation results show that the proposed LRLS filter is faster for its short length than the quadratic version of the LRLS filter in [8].

Manuscript received June 22, 2017.
Manuscript revised August 8, 2017.

[†]The author is with the School of Electrical and Computer Engineering, College of Scienceand Engineering, Kanazawa University, Kanazawa-shi, 920-1192 Japan.
a) E-mail: horita@se.kanazawa-u.ac.jp

DOI: 10.1587/transfun.E100.A.3045

2. Existing Leaky RLS Filters

We define a cost function of the LRLS filter [7] which is a criterion of the ridge regression [12] as

$$J(k) = \sum_{i=1}^{k} \lambda^{k-i}\{d(i) - \boldsymbol{w}^T(k)\boldsymbol{u}(i)\}^2 + \alpha\|\boldsymbol{w}(k)\|_2^2 \quad (1)$$

where λ, $d(i), i = 1, 2, \ldots, k$ and α denote the exponential weighting factor, the desired response and the regularization parameter, respectively. The vectors $\boldsymbol{w}(k)$ and $\boldsymbol{u}(i), i = 1, 2, \ldots, k$ in (1) consist of coefficients and inputs of the LRLS filter, respectively defined by

$$\boldsymbol{w}^T(k) = [\ w_0(k), w_1(k), \ldots, w_{N-1}(k)\ ], \quad (2)$$

$$\boldsymbol{u}^T(i) = [\ u(i), u(i-1), \ldots, u(i-N+1)\ ]. \quad (3)$$

2.1 Conventional $O(N^3)$ LRLS Filter: LRLS1

The conventional LRLS filter [7] is expressed as follows: Initialize the filter by setting

$$\boldsymbol{\Phi}(0) = \alpha\boldsymbol{I},\ \boldsymbol{w}(0) = \boldsymbol{O} \quad (4)$$

For each instant of time $k = 1, 2, \ldots$, compute

$$\boldsymbol{\Phi}(k) = \lambda\boldsymbol{\Phi}(k-1) + \boldsymbol{u}(k)\boldsymbol{u}^T(k) + \alpha(1-\lambda)\boldsymbol{I} \quad (5)$$

$$\epsilon(k) = d(k) - \boldsymbol{w}^T(k-1)\boldsymbol{u}(k) \quad (6)$$

$$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \alpha(1-\lambda)\boldsymbol{\Phi}^{-1}(k)\boldsymbol{w}(k-1) + \boldsymbol{\Phi}^{-1}(k)\boldsymbol{u}(k)\epsilon(k) \quad (7)$$

where the correlation matrix $\boldsymbol{\Phi}(k)$ is defined by

$$\boldsymbol{\Phi}(k) = \sum_{i=1}^{k} \lambda^{k-i}\boldsymbol{u}(i)\boldsymbol{u}^T(i) + \alpha\boldsymbol{I}. \quad (8)$$

The above LRLS filter is not the Levenberg-Marquardt regularization [4] but the Tikhonov [6] regularized RLS algorithm with the exponential weighting factor, which was pointed out by Waterschoot et al. in [8] since (7) is not

Table 1 Estimated computational cost of the quadratic LRLS filter per iteration.

Equations	Divides	Multiplications	Additions
$\beta(k) = ((\alpha(1-\lambda)N)^{-1} + \lambda^{-1}[\boldsymbol{P}(k-1)]_{l,l}$	0	1	1
$\boldsymbol{R}(k) = \lambda^{-1}\boldsymbol{P}(k-1) - \beta^{-1}(k)\lambda^{-2}[\boldsymbol{P}(k-1)]_{:,l}[\boldsymbol{P}(k-1)]_{l,:}$	1	$2N^2+N$	N^2
$\boldsymbol{g}_1(k) = \boldsymbol{R}(k)\boldsymbol{u}(k)(1+\boldsymbol{u}^T(k)\boldsymbol{R}(k)\boldsymbol{u}(k))^{-1}$	1	$2N^2+2N$	$2N^2-N$
$\boldsymbol{P}(k) = \boldsymbol{R}(k) - \boldsymbol{g}_1(k)\boldsymbol{u}^T(k)\boldsymbol{R}(k)$	0	N^2	N^2
$\epsilon(k) = d(k) - \boldsymbol{w}^T(k-1)\boldsymbol{u}(k)$	0	N	N
$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \alpha(1-\lambda)\boldsymbol{P}(k)\boldsymbol{w}(k-1) + \boldsymbol{g}_1(k)\epsilon(k)$	0	N^2+2N	N^2+N
total	2	$6N^2+6N+1$	$5N^2+N+1$

Table 2 Estimated computational cost of the proposed LRLS filter per iteration.

Equations	Divides	Multiplications	Additions
$\boldsymbol{g}_\lambda(k) = \boldsymbol{P}(k)\boldsymbol{u}(k)(\lambda+\boldsymbol{u}^T(k)\boldsymbol{P}(k)\boldsymbol{u}(k))^{-1}$	1	$2N^2+2N$	$2N^2-N$
$\boldsymbol{C}(k) = \lambda^{-1}\boldsymbol{P}(k) - \lambda^{-1}\boldsymbol{g}_\lambda(k)\boldsymbol{u}^T(k)\boldsymbol{P}(k)$	0	$2N^2+N$	N^2
$\boldsymbol{P}(k) = \boldsymbol{C}(k) - \alpha(1-\lambda)\boldsymbol{C}^2(k)$	0	N^3+N^2	N^3
$\epsilon(k) = d(k) - \boldsymbol{w}^T(k-1)\boldsymbol{u}(k)$	0	N	N
$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \alpha(1-\lambda)\boldsymbol{P}(k)\boldsymbol{w}(k-1) + \boldsymbol{P}(k)\boldsymbol{u}(k)\epsilon(k)$	0	$2N^2+2N$	$2N^2$
total	1	N^3+7N^2+6N	N^3+5N^2

equivalent with $\lambda \neq 1$ to

$$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) + \boldsymbol{\Phi}^{-1}(k)\boldsymbol{u}(k)\epsilon(k). \tag{9}$$

2.2 Existing $O(N^2)$ LRLS Filter: LRLS2

We can obtain the quadratic version of the LRLS filter [8] by introducing the regularization method in [4] as follows: Initialize the filter by setting

$$\boldsymbol{P}(0) = \alpha^{-1}\boldsymbol{I},\ \boldsymbol{w}(0) = \boldsymbol{O} \tag{10}$$

For each instant of time $k = 1, 2, \ldots$, compute

$$\beta(k) = ((\alpha(1-\lambda)N)^{-1} + \lambda^{-1}\boldsymbol{\xi}_{k,N}^T\boldsymbol{P}(k-1)\boldsymbol{\xi}_{k,N} \tag{11}$$

$$\boldsymbol{R}(k) = \lambda^{-1}\boldsymbol{P}(k-1) - \beta^{-1}(k)\lambda^{-2}\boldsymbol{P}(k-1)\boldsymbol{\xi}_{k,N}\boldsymbol{\xi}_{k,N}^T\boldsymbol{P}(k-1) \tag{12}$$

$$\boldsymbol{g}_1(k) = \boldsymbol{R}(k)\boldsymbol{u}(k)(1+\boldsymbol{u}^T(k)\boldsymbol{R}(k)\boldsymbol{u}(k))^{-1} \tag{13}$$

$$\boldsymbol{P}(k) = \boldsymbol{R}(k) - \boldsymbol{g}_1(k)\boldsymbol{u}^T(k)\boldsymbol{R}(k) \tag{14}$$

$$\epsilon(k) = d(k) - \boldsymbol{w}^T(k-1)\boldsymbol{u}(k) \tag{15}$$

$$\boldsymbol{w}(k) = \boldsymbol{w}(k-1) - \alpha(1-\lambda)\boldsymbol{P}(k)\boldsymbol{w}(k-1) + \boldsymbol{g}_1(k)\epsilon(k) \tag{16}$$

where $\boldsymbol{P}(k)$ denotes the inverse matrix of $\boldsymbol{\Phi}(k)$ and $\boldsymbol{\xi}_{k,N}$ is an $N \times 1$ zeros vector except for its $((k-1)\ mod\ N)+1$ element which is 1. If we define the index of $\boldsymbol{\xi}_{k,N}$ as l, i.e., $l = ((k-1)\ mod\ N)+1$, we can obtain (17) and (18) in exchange for (11) and (12), respectively,

$$\beta(k) = ((\alpha(1-\lambda)N)^{-1} + \lambda^{-1}[\boldsymbol{P}(k-1)]_{l,l}, \tag{17}$$

$$\boldsymbol{R}(k) = \lambda^{-1}\boldsymbol{P}(k-1) - \beta^{-1}(k)\lambda^{-2}[\boldsymbol{P}(k-1)]_{:,l}[\boldsymbol{P}(k-1)]_{l,:}, \tag{18}$$

where the vectors $[\boldsymbol{P}(k-1)]_{:,l}$ and $[\boldsymbol{P}(k-1)]_{l,:}$ are respectively defined by

$$[\boldsymbol{P}(k-1)]_{:,l} = \begin{bmatrix} p(k-1)_{1l} \\ p(k-1)_{2l} \\ \vdots \\ p(k-1)_{Nl} \end{bmatrix}, \tag{19}$$

$$[\boldsymbol{P}(k-1)]_{l,:} = [p(k-1)_{l1}, p(k-1)_{l2}, \ldots, p(k-1)_{lN}]. \tag{20}$$

We show in Table 1 estimated numbers of real divisions, multiplications and additions that are required in computing specific equations of the quadratic LRLS filter in which we used parts of version II of the standard RLS algorithm summarized in Table 13.2 of [2] for preventing its explosive divergence [5]. The quadratic LRLS filter includes two divides per iteration although the standard RLS algorithm in [1], [2] has a divide per iteration.

3. Proposed LRLS Filter

We derive an LRLS filter that has only one divide per iteration in this section.

First, we rewrite (5) as the next two equations.

$$\boldsymbol{C}^{-1}(k) = \lambda\boldsymbol{\Phi}(k-1) + \boldsymbol{u}(k)\boldsymbol{u}^T(k) \tag{21}$$

$$\boldsymbol{\Phi}(k) = \boldsymbol{C}^{-1}(k) + \alpha(1-\lambda)\boldsymbol{I} = (\boldsymbol{I} + \alpha(1-\lambda)\boldsymbol{C}(k))\boldsymbol{C}^{-1}(k) \tag{22}$$

Moreover, we obtain (23) from the matrix inversion of the second line of (22),

$$\boldsymbol{\Phi}^{-1}(k) = \boldsymbol{C}(k)(\boldsymbol{I} + \alpha(1-\lambda)\boldsymbol{C}(k))^{-1} \tag{23}$$

where $\boldsymbol{C}(k)$ is given by using the matrix inversion lemma for (21) as

$$\boldsymbol{g}_{\lambda}(k) = \boldsymbol{P}(k-1)\boldsymbol{u}(k)(\lambda + \boldsymbol{u}^{T}(k)\boldsymbol{P}(k-1)\boldsymbol{u}(k))^{-1}, \quad (24)$$

$$\boldsymbol{C}(k) = \lambda^{-1}\boldsymbol{P}(k-1) - \lambda^{-1}\boldsymbol{g}_{\lambda}(k)\boldsymbol{u}^{T}(k)\boldsymbol{P}(k-1). \quad (25)$$

In addition, we accept Lemma 2.3.3 in [12] as follows:

Lemma 1: If $\boldsymbol{F} \in R^{n\times n}$ and $\|\boldsymbol{F}\|_p < 1$, then $\boldsymbol{I} - \boldsymbol{F}$ is nonsingular and

$$(\boldsymbol{I} - \boldsymbol{F})^{-1} = \sum_{l=0}^{\infty} \boldsymbol{F}^{l} \quad (26)$$

with

$$\|(\boldsymbol{I} - \boldsymbol{F})^{-1}\|_p \leq \frac{1}{1 - \|\boldsymbol{F}\|_p.} \quad (27)$$

If we accept Lemma 1 for the matrix $(\boldsymbol{I} + \alpha(1-\lambda)\boldsymbol{C}(k))^{-1}$ of (23) and neglect $\boldsymbol{F}^{l}, l \geq 2$ of (26), then we can obtain the matrix $\boldsymbol{P}(k) = \boldsymbol{\Phi}^{-1}(k)$ as

$$\begin{aligned}\boldsymbol{P}(k) &\approx \boldsymbol{C}(k)(\boldsymbol{I} - \alpha(1-\lambda)\boldsymbol{C}(k)) \quad (28)\\ &= \boldsymbol{C}(k) - \alpha(1-\lambda)\boldsymbol{C}^{2}(k),\end{aligned}$$

subject to

$$\|\alpha(1-\lambda)\boldsymbol{C}(k)\|_p < 1. \quad (29)$$

In this letter, we set the parameters α and λ to appropriate values in advance to satisfy (29) with $p = 1$.

We show in Table 2 estimated numbers of real divisions, multiplications and additions that are required in computing specific equations of the proposed LRLS filter. The proposed LRLS filter includes only one divide per iteration although it has more multiplications and additions than the quadratic version of the LRLS filter. In the next section, simulation results are shown that the proposed LRLS filter is faster for small N than the quadratic version of the LRLS filter.

4. Simulation Results

In this section, the proposed LRLS filter is compared with the existing LRLS algorithms in 2. I used Scilab 6.0.0 on an Intel(R) Core(TM)i5-7300U CPU @2.50 GHz processor with 8.00 GB RAM in all the experiments.

The unknown system was produced by using a first-order Markov model as [13]

$$\boldsymbol{w}_o(k) = \boldsymbol{w}_o(k-1) + \boldsymbol{v}_w(k) \quad (30)$$

where $\boldsymbol{v}_w(k)$ was a white Gaussian noise with variance $\sigma^2_{v_w}$ and the initial value $\boldsymbol{w}_o(0)$ was given as `10*h0` in the following Scilab command:

```
[h0,hm,fr]=wfir("lp",N,[.2 0],"hm",[0 0]);
```

The input signal was a colored noise, generated by filtering a white Gaussian signal of unit variance with an IIR filter which had a transfer function [13], [14]

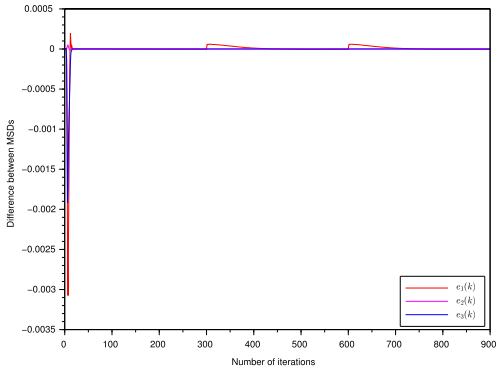

Fig. 1 Differences between learning curves with $\alpha = 0.001$ for a color Gaussian input signal (Experiment 1).

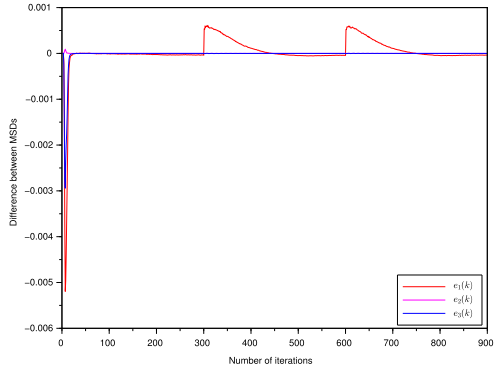

Fig. 2 Differences between learning curves with $\alpha = 0.01$ for a color Gaussian input signal (Experiment 2).

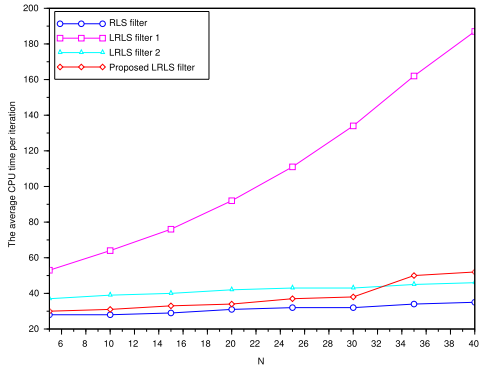

Fig. 3 Average CPU time per iteration of each adaptive filter in μs (Experiment 2).

$$H(z) = \frac{1}{1 - 0.95z^{-1} - 0.19z^{-2} - 0.09z^{-3} + 0.5z^{-4}}$$

In Experiment 1, I set $\sigma^2_{v_w} = 10^{-4}$ and $N = 5, 10, \ldots, 40$. The desired signal was corrupted by a white Gaussian measurement noise with variance $\sigma^2_v = 10^{-4}$. The weight vector of the unknown system was multiplied by -1 at iterations 300 and 600 so as to examine the convergence performance of each adaptive filter. The forgetting factor λ and the regularization parameter α were respectively set to 0.98 and 0.001.

In Experiment 2, I repeated Experiment 1 except that the regularization parameter α was set to 0.01 in order to check the stability performance of each adaptive filter.

I compared the performance of the proposed LRLS filter with that of the RLS and the existing LRLS filters by the mean-square-deviation (MSD) curves. The MSD was computed as $\|\boldsymbol{w}_o^{(i)}(k)-\boldsymbol{w}^{(i)}(k)\|_2/\|\boldsymbol{w}_o^{(i)}(k)\|_2$ averaged over $i = 1, 2, \ldots, 1000$ independent trials in each algorithm as follows:

$$MSD(k) = \frac{1}{1000}\sum_{i=1}^{1000}\frac{\|\boldsymbol{w}_o^{(i)}(k) - \boldsymbol{w}^{(i)}(k)\|_2}{\|\boldsymbol{w}_o^{(i)}(k)\|_2} \tag{31}$$

$$e_1(k) = MSD(k)_{RLS} - MSD(k)_{LRLS\,1} \tag{32}$$

$$e_2(k) = MSD(k)_{Proposed} - MSD(k)_{LRLS\,1} \tag{33}$$

$$e_3(k) = MSD(k)_{Proposed} - MSD(k)_{LRLS\,2} \tag{34}$$

The curves of $e_1(k)$, $e_2(k)$ and $e_3(k)$ with $N = 10$ for Experiment 1 and Experiment 2 were illustrated in Fig. 1 and Fig. 2, respectively. The average CPU time per iteration in microseconds of each adaptive filter for Experiment 2 was plotted in Fig. 3. Figs. 1 and 2 show that the stability performance of the proposed LRLS filter is as good as that of the existing LRLS filters and better than that of the RLS filter although the steady-state performance of the proposed filter with the larger α is inferior to that of the RLS filter. Fig. 3 clearly shows the increase in computational efficiency of the proposed filter when the length of the adaptive filter is short.

5. Conclusion

In this letter, we have obtained the cubic version of the LRLS filter which is computationally more efficient than the quadratic version of the LRLS filter in [8] when the length of the adaptive filter is short. The proposed LRLS filter includes only a divide per iteration although the quadratic version of the LRLS filter has two divides per iteration. Simulation results demonstrate that the proposed LRLS filter is faster for its short length than the quadratic version of the LRLS filter.

When a possible maximum length of the unknown system is not given in advance, however, we would select the quadratic LRLS algorithm of [8].

References

[1] A.H. Sayed, Adaptive Filters, IEEE Press, John Wiley & Sons, Inc., 2008.
[2] S. Haykin, Adaptive Filter Theory, 2nd ed., Prentice Hall, 1991.
[3] M.C. Tsakiris, C.G. Lopes, and V.H. Nascimento, "An array recursive least-squares algorithm with generic nonfading regularization matrix," IEEE Signal Process. Lett., vol.17, no.12, pp.1001–1004, Dec. 2010.
[4] L. Ljung, T. Söderström, Theory and Practice of Recursive Identification, MIT Press, Cambridge, MA, 1986.
[5] G.E. Bottomley and S.T. Alexander, "A novel approach for stabilizing recursive least squares filters," IEEE Trans. Signal Process., vol.39, no.8, pp.1770–1779, Aug. 1991.
[6] A. Tikhonov, V. Arsenin, Solutions of Ill-Posed Problems, Wiley, New York, 1977.
[7] E. Horita, K. Sumiya, H. Urakami, and S. Mitsuishi, "A leaky RLS algorithm: Its optimality and implementation," IEEE Trans. Signal Process., vol.52, no.10, pp.2924–2932, Oct. 2004.
[8] T. van Waterschoot, G. Rombouts, and M. Moonen, "Optimally regularized adaptive filtering algorithms for room acoustic signal enhancement," Elsevier Signal Process., vol.88, no.3, pp.594–611, 2008.
[9] J.M. Gil-Cacho, M. Signoretto, T .van Waterschoot, M. Moonen, and S.H. Jensen, "Nonlinear acoustic echo cancellation based on a sliding-window leaky kernel affine projection algorithm," IEEE Trans. Audio, Speech, Language Process., vol.21, no.9, pp.1867–1878, Sept. 2013.
[10] A.H. Sayed, A. Garulli, and S. Chandrasekaran, "A fast iterative solution for worst-case parameter estimation with bounded model uncertainties," Proc. Amer. Contr. Conf., pp.1499–1503, Albuquerque, NM, June 1997.
[11] E. Horita, "A fast algorithm for time-varying regularization parameter updates in an LRLS filter," IEICE Trans. Fundamentals (Japanese Edition), vol.J99-A, no.10, pp.399–407, Oct. 2016.
[12] G.H. Golub and C.F. Van Loan, Matrix Computations 3rd Edition, The Johns Hopkins University Press, 1996.
[13] Md. Zulfiquar Ali Bhotto, M. Omair Ahmad, and M.N. S. Swamy, "Robust shrinkage affine-projection sign adaptive-filtering algorithms for impulsive noise environments," IEEE Trans. Signal Process., vol.62, no.13, pp.3349–3359, July 2014.
[14] S. Werner and P.S.R. Diniz, "Set-membership affine projection algorithm," IEEE Signal Process. Lett., vol.8, no.8, pp.231–235, Aug. 2001.