Consideration on the Performance of Kernel Adaptive Filters for the Mixture of Linear and Non-Linear Environments

Kiyoshi Nishikawa[*], Felix Albu[†]
[*]Department of Information and Communication Systems, Tokyo Metropolitan University
E-mail: knishikawa@m.ieice.org Tel: +81-42-585-8423
[†] Valahia University of Targoviste, Targoviste, Romania
E-mail: felix.albu@gmail.com

Abstract—**In this paper, we consider the characteristics of the kernel adaptive filters for the mixture of linear and non-linear environments. We first consider employing a linear kernel as one of the kernels in multi-kernel adaptive filters. It is pointed out that the convergence characteristics of the filter corresponding to the linear kernel is affected by the selection of the other kernels. Then, we propose a new structure which uses a linear and a multi-kernel adaptive filter simultaneously. In the proposed method, the a posteriori error of the linear filter is calculated before updating the multi-kernel adaptive filter. Then, the a posteriori error is used to calculate the update term for the multi-kernel adaptive filter. The effectiveness of the proposed method is confirmed by computer simulations.**

I. Introduction

So far, the linear adaptive filters are widely researched and utilized in a lot of applications, such as acoustic echo canceler, channel equalization, system identification[1], [2]. Recently, the concept of the kernel adaptive filters is proposed by applying the kernel method to the linear adaptive filters[3]. The kernel adaptive filters enable the autonomous learning of the non-linear systems. Several algorithms for kernel adaptive filters had been proposed, namely, the kernel least mean square (KLMS)[3], the kernel normalized least mean square (KNLMS)[4], the kernel proportionate NLMS[5], the kernel recursive least squares (KRLS)[6], the KRLS-DCD[7], [8] and so forth. These algorithms are derived by the ones for linear adaptive filters and have similar characteristics to their counterparts of the linear ones.

As an extended structure, the multi-kernel adaptive filter was proposed in [9]. It uses multiple kernels simultaneously in the structure, e.g., the Gaussian and Polynomial kernels. One of the advantages using the structure is that we can relax the problem of the selection of the kernel parameter. When we apply the kernel adaptive filters to applications, one of the practical problem is the selection of the kernel parameters. In the kernel adaptive filter, the Gaussian kernel is widely used, and it has an adjustable parameter, i.e., the bandwidth parameter. It is known that the convergence characteristics of a kernel filter depends on the selection of the parameter[4]. By using the multiple Gaussian kernels with different values of bandwidth parameter, we can relax the dependency of the convergence characteristics on the kernel parameter.

In this paper, we use the multi-kernel adaptive filter structure for considering the characteristics of the kernel adaptive filter in the mixture of linear and non-linear environments which can be modeled as a Volterra series. That is, we select, a linear kernel in the multi-kernel adaptive filter structure for effective learning of the linear components. The update equation for the filter corresponding to the linear kernel is examined and equivalent formula as a linear adaptive filter configuration is derived. From the formula, it is pointed out that the non-linear kernels affect the update of the linear-equivalent filter. Based on this consideration, we propose a modified structure which consists of linear and multi-kernel adaptive filters.

Different from the conventional multi-kernel structure, we propose to use a linear adaptive filter instead of a linear kernel in the multi-kernel structure. We show that, using the proposed structure, we can derive an update formula for independently update the filters without mutual affection. For that, we first update only the linear filter and calculate the a posteriori error using the updated filter. Then, using the a posteriori error, the update terms for the multi-kernel filter are obtained. As a result, we can update the linear and the multi-kernel filters independently so that the effect of multi-kernel filter on the linear one could be reduced. Note that the update in different timing in the proposed method is merely conceptual, and we show the update formula for the filters at once. Through computer simulations, we confirm the effectiveness of the proposed method.

II. Preparation

Here, we briefly review the kernel adaptive filters[3], especially the kernel normalized least mean square (KNLMS) algorithm and its multi-kernel implementation[9]. Then, we consider using a linear kernel as one of the kernels used in the multi-kernel KNLMS algorithm.

In the following, a matrix or a vector is indicated by bold letters, e.g., $\mathbf{R}$, or $\mathbf{r}$. The transpose of a matrix $\mathbf{R}$ is indicated as $\mathbf{R}^{\mathrm{T}}$, and variables at time n is expressed as $\mathbf{R}(n)$.

A. Non-linear system model

For deriving the proposed method, we assume the target environment can be modeled as a Volterra series[10], [11], [12]. Volterra series expansion is a well-known model of non-linear systems and is given as

$$\begin{aligned} y(n) &= h_0 + \sum_{m_1=0}^{\infty} h_1(m_1)x(n-m_1) \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_2=m_1}^{\infty} h_2(m_1,m_2)x(n-m_1)x(n-m_2) \\ &+ \sum_{m_1=0}^{\infty}\sum_{m_2=m_1}^{\infty}\cdots\sum_{m_p=m_{p-1}}^{\infty} h_m(m_1,m_2,\ldots,m_p) \\ &\times x(n-m_1)x(n-m_2)\cdots x(n-m_p) \\ &+ \ldots \end{aligned} \tag{1}$$

where $y(n)$ and $x(n)$ are the output and the input signals of the system whose p-th order coefficients are given as $h_p(m_1,\ldots,m_p)$. Usually the term h_0 is assumed to be zero, and in the following, we also assume $h_0 = 0$.

Under the Volterra model, we can assume that the input-output relation could be treated as a summation of linear terms and non-linear, or higher order, terms. Although the Volterra series expansion does not express all classes of non-linear systems, we assume the equation (1) for the rest of the paper.

B. Kernel adaptive filters

The concept of a kernel adaptive filter is derived by applying the kernel method to the linear adaptive filters[3]. To apply the kernel method, the input signal $\{x(n) \mid n = 0, 1, 2, \ldots\}$ is mapped onto a higher order characteristics space. We denote the input vector of the filter at time n as $\mathbf{x}(n)$, and its length is assumed to be S. Then, we express this mapping as $\phi(\mathbf{x}(n))$.

Using the input and output signals of an unknown system, the adaptive filter will estimate the unknown system itself[2]. For the kernel adaptive filter, we expand and approximate the coefficient vector of the adaptive filter $\mathbf{w}(n)$ in terms of $\{\phi(\mathbf{x}(m)) \mid m = 0,\ldots,n-1\}$ as $\mathbf{w}'(n)$:

$$\mathbf{w}'(n) = h_0\phi(\mathbf{x}(0)) + \cdots + h_{n-1}\phi(\mathbf{x}(n-1)) \tag{2}$$

where $\{h_i \mid i = 0,\ldots,n-1\}$ are the weights to be determined. The length of $\mathbf{w}(n)$ and $\mathbf{w}'(n)$ are same as that of $\mathbf{x}(n)$, or S. Using the kernel trick[13], the output signal $y(n)$ is expressed as

$$\begin{aligned} y(n) &= \phi(\mathbf{x}(n))^{\mathrm{T}}\mathbf{w}'(n) \\ &= \phi(\mathbf{x}(n))^{\mathrm{T}}(h_0\phi(\mathbf{x}(0)) + \cdots + h_{n-1}\phi(\mathbf{x}(n-1))) \\ &= h_0\phi(\mathbf{x}(n))^{\mathrm{T}}\phi(\mathbf{x}(0)) + \cdots + h_n\phi(\mathbf{x}(n)))^{\mathrm{T}}\phi(\mathbf{x}(n-1)) \\ &= h_0\kappa(\mathbf{x}(n),\mathbf{x}(0)) + \cdots + h_n\kappa(\mathbf{x}(n),\mathbf{x}(n-1)). \end{aligned} \tag{3}$$

where $\kappa(\cdot,\cdot)$ shows the kernel function. By defining the vectors $\mathcal{X}(n)$ and $\boldsymbol{h}(n)$ as

$$\begin{aligned} \mathcal{X}(n) &= [\kappa(\mathbf{x}(n),\mathbf{x}(0)) \quad \kappa(\mathbf{x}(n),\mathbf{x}(1)) \quad \cdots \quad \kappa(\mathbf{x}(n),\mathbf{x}(n-1))]^{\mathrm{T}} \\ \boldsymbol{h}(n) &= [h_0 \quad h_1 \quad \ldots \quad h_{n-1}]^{\mathrm{T}} \end{aligned} \tag{4}$$

and substituting into (3), we have a simpler expression of $y(n)$

$$y(n) = \mathcal{X}^{\mathrm{T}}(n)\boldsymbol{h}(n). \tag{5}$$

This equation can be interpreted as the input-output relation of a filter $\boldsymbol{h}(n)$. Based on this expression, $\boldsymbol{h}(n)$ is regarded as the filter coefficients of a kernel adaptive filter instead of $\mathbf{w}(n)$ and, for updating $\boldsymbol{h}(n)$, we could apply the linear adaptive algorithms with slight modifications[3].

As the kernel function, the Gaussian kernel in the following form is generally used.

$$\kappa(\mathbf{x},\mathbf{y}) = \exp\left(-\zeta\|\mathbf{x}-\mathbf{y}\|^2\right) \tag{6}$$

where the parameter ζ is called kernel parameter or kernel bandwidth.

C. Kernel normalized least square (KNLMS) algorithm

Here, we describe the KNLMS algorithm which is used in the proposed method.

The KNLMS algorithm is a kernel version of the well-known NLMS algorithm[2], [14] for linear adaptive filters. The filter coefficients $\boldsymbol{h}(n)$ are updated by

$$\boldsymbol{h}(n+1) = \boldsymbol{h}(n) + \eta\frac{e(n)\mathcal{X}(n)}{\epsilon + \mathcal{X}^{\mathrm{T}}(n)\mathcal{X}(n)} \tag{7}$$

$$e(n) = d(n) - \mathcal{X}^{\mathrm{T}}(n)\boldsymbol{h}(n-1) \tag{8}$$

where η and ϵ are a step size parameter and a stabilization parameter respectively. We should note that the order of $\boldsymbol{h}(n)$ and $\mathcal{X}(n)$ will be incremented as n advances. This feature does not exist in the linear adaptive filters.

We store the past input signals $\{\mathbf{x}(j) \mid j = 0, 1, \cdots, n-1\}$ as a dictionary and its size increases at each time. This means that the computational load is time varying and long learning time results in heavy load. To maintain the applicable amount of calculation, several sparsification methods of the input signal are proposed so far[3], [4], [15]. In this paper, we use the one proposed in [4] due to its simple structure. In that, the condition below is examined at each time

$$\max|\mathcal{X}(n)|_{j=1,\cdots,J} < \mathcal{T}_0 \tag{9}$$

where $\mathcal{T}_0$ is the predefined threshold value[4], and J shows the number of entries in the dictionary. Only when this condition holds, $\mathbf{x}(n)$ will be added to the dictionary as a new entry.

D. Multi-kernel adaptive filter

One of the practical problems of the kernel adaptive filters is to select a suitable value for kernel parameter. When we use the Gaussian kernel, it is known that the convergence characteristics are affected by the band width parameter ζ. The multi-kernel filter is proposed to relax this problem by using multiple kernels simultaneously[9].

In multi-kernel configuration, the filter output is expressed as

$$y_{\mathrm{k}}(n) = \sum_{m=1}^{M}\sum_{j=1}^{J_m} h_j^{(m)}(n)\kappa_m(\mathbf{x}(n),\mathbf{x}(j)) \tag{10}$$

where M shows the number of kernels used, and J_m the number of entries in the dictionary of m-th kernel. $h_j^{(m)}(n)$ shows the j-th coefficients of the adaptive filter of the m-th kernel at time n. In Fig. 1, we show a configuration of the multi-kernel adaptive filter[9].

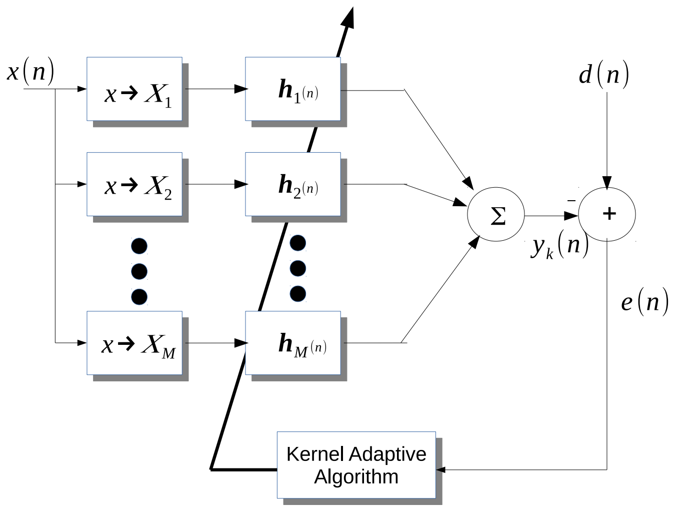

Fig. 1. Configuration of the multi-kernel adaptive filter.

Although we could select any combination of kernels in this configuration, e.g., the Gaussian and the polynomial, we only consider to use multiple Gaussian kernels with different values of the kernel parameters of each kernel, i.e., ζ_m. It is shown[9] that the configuration could relax the problem of selecting the optimum values for kernel parameters.

Let us consider an efficient implementation of the multi-kernel adaptive filter under the Volterra model of (1). Here, we consider using a linear kernel as one of the kernels in (10) to effectively learn the linear terms of (1). The linear kernel κ_ℓ is expressed as

$$\kappa_\ell(\mathbf{x}(n), \mathbf{x}(i)) = \mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(i) \tag{11}$$

or, simply the calculation of the inner product of two vectors.

Assuming the kernel function $\kappa_m(\cdot, \cdot)$ of $m = 1$ is the linear kernel, the equation (10) is rearranged as

$$y_{\mathrm{k}}(n) = \sum_{j=1}^{J_1} h_j^{(1)}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(j) + \sum_{m=2}^{M}\sum_{j=1}^{J_m} h_j^{(m)}(n)\kappa_m(\mathbf{x}(n), \mathbf{x}(j)) \tag{12}$$

We define the output of the linear kernel $y_{\mathrm{Lk}}(n)$ as

$$\begin{aligned} y_{\mathrm{Lk}}(n) &= \sum_{j=0}^{n} h_j^{(1)}(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(j) \\ &= \mathbf{x}^{\mathrm{T}}(n)\sum_{j=0}^{n} h_j^{(1)}(n)\mathbf{x}(j) \end{aligned} \tag{13}$$

where it is assumed that we do not use the sparsification method for the linear kernel, and hence, the upper limit of the summation is set as n. Also, that of the Gaussian kernels $y_{\mathrm{Gk}}(n)$ is defined as

$$y_{\mathrm{Gk}}(n) = \sum_{m=2}^{M}\sum_{j=1}^{J_m} h_j^{(m)}(n)\kappa_m(\mathbf{x}(n), \mathbf{x}(j)) \tag{14}$$

Then, we can write the error signal $e(n)$ as

$$\begin{aligned} e(n) =& d(n) - y_{\mathrm{k}}(n) \\ =& d(n) - y_{\mathrm{Lk}}(n) - y_{\mathrm{Gk}}(n) \end{aligned} \tag{15}$$

Each filter coefficient vector will be updated according to the equation

$$\begin{aligned} \boldsymbol{h}_m(n+1) =& \boldsymbol{h}_m(n) + \eta \frac{e(n)\mathcal{X}_m(n)}{\sum_{m=1}^{M}\mathcal{X}_m^{\mathrm{T}}(n)\mathcal{X}_m(n)} \\ & m = 1, 2, \ldots, M \end{aligned} \tag{16}$$

Note that, for the linear kernel filter, $\mathcal{X}_1(n)$ is given as

$$\mathcal{X}_1(n) = [\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(0) \quad \mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(1) \quad \ldots \quad \mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n-1)]^{\mathrm{T}} \tag{17}$$

In this configuration, the linear and kernel filters are updated using the same equation (16).

III. Proposed method

In this section, we first consider the effect of the kernel functions on the adaptive filter corresponding to the linear kernel when the multi-kernel filter of the previous section is used. Then, we propose a new structure to reduce the effect of kernel functions on the linear filter.

A. Dependency of linear kernel on the other kernels

Let us consider the effect of the kernel functions on the adaptation of the filter corresponding to the linear kernel in the configuration described in Section II-D.

From (16), it is known that the filter coefficient vector corresponding to the linear kernel is updated according to the equation below.

$$\boldsymbol{h}_1(n+1) = \boldsymbol{h}_1(n) + \eta \frac{e(n)\mathcal{X}_1(n)}{\sum_{k=1}^{M}\mathcal{X}_k^{\mathrm{T}}(n)\mathcal{X}_k(n)} \tag{18}$$

where we assumed $m = 1$ for the linear kernel. Also, from (13), we define the filter coefficients vector $\boldsymbol{w}_{\ell k}(n)$ as below

$$\boldsymbol{w}_{\ell k}(n) = \sum_{j=0}^{n-1} h_j^{(1)}(n)\mathbf{x}(j) \tag{19}$$

and it can be regarded as the coefficient vector of the linear adaptive filter.

Let us rewrite the equation (16) in terms of $\boldsymbol{w}_{\ell k}(n)$ instead of $\boldsymbol{h}_1(n)$. From (18), we obtain the update formula for each element of $\boldsymbol{h}(n)$ as

$$h_j^{(1)}(n+1) = h_j^{(1)}(n) + \eta \frac{e(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(j)}{\sum_{k=1}^{M}\mathcal{X}_k^{\mathrm{T}}(n)\mathcal{X}_k(n)}. \tag{20}$$

Then, the update equation of $\boldsymbol{w}_{\ell k}(n)$ is obtained as below:

$$\begin{aligned}\boldsymbol{w}_{\ell k}(n+1) &= \sum_{j=0}^{n-1} h_j^{(1)}(n+1)\mathbf{x}(j) \\ &= \sum_{j=0}^{n-1} h_j^{(1)}(n)\mathbf{x}(j) + \sum_{j=0}^{n-1} \eta \frac{e(n)\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(j)\mathbf{x}(j)}{\sum_{k=1}^{M} \mathcal{X}_k^T(n)\mathcal{X}_k(n)} \\ &= \boldsymbol{w}_{\ell k}(n) + \frac{\eta e(n)}{\sum_{k=1}^{M} \mathcal{X}_k^{\mathrm{T}}(n)\mathcal{X}_k(n)} \sum_{j=0}^{n-1} \mathbf{x}(j)\mathbf{x}^{\mathrm{T}}(j)\mathbf{x}(n).\end{aligned} \tag{21}$$

The last line of this equation resembles the formula of the standard linear NLMS algorithm[14]. However, there are differences between them. Namely, the denominator of the second term includes $\sum_{k=1}^{M} \mathcal{X}_k^{\mathrm{T}}(n)\mathcal{X}_k(n)$ which depends on the kernel functions $\{\kappa_m(\cdot,\cdot) \mid m = 1, 2, \cdots, M\}$. It means that the selection of kernel functions or that of kernel parameters could affect the convergence characteristics of $\boldsymbol{w}_{\ell k}(n)$.

B. *Proposed algorithm*

Here, we consider the structure of the multi-kernel adaptive filter in which the linear adaptive filter could be updated independently of the other kernels.

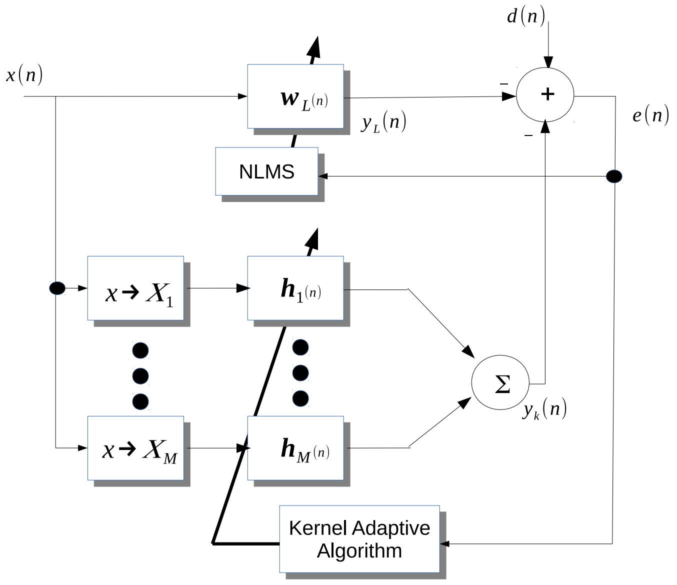

Fig. 2. Possible configuration of mixture of a linear and a multi-kernel adaptive filters.

In the proposed method, we employ a linear adaptive filter simultaneously with the multi-kernel adaptive filter. Note the difference from the structure considered in II-D where a linear kernel is used as one of the multiple kernels. A possible structure of the mixture of a linear and a kernel adaptive filters is shown in Fig. 2. Let us denote the filter coefficients of the linear filter as $\boldsymbol{w}_L(n)$ and is expressed as

$$\boldsymbol{w}_L(n) = [w_0(n) \quad w_1(n) \quad \ldots \quad w_{S-1}(n)]^{\mathrm{T}} \tag{22}$$

where S shows the number of coefficients. Its output $y_{\mathrm{L}}(n)$ is defined as

$$y_{\mathrm{L}}(n) = \mathbf{x}^{\mathrm{T}}(n)\boldsymbol{w}_L(n). \tag{23}$$

Also, in the proposed method, we employ a multi-kernel filter whose kernels are assumed to be Gaussian kernels with different values of the kernel parameter as in [9]. The output $y_{\mathrm{k}}(n)$ of the multi-kernel filters are given as (10).

Then, the error signal $\hat{e}(n)$ in the proposed method is given as

$$\hat{e}(n) = d(n) - y_{\mathrm{L}}(n) - y_{\mathrm{k}}(n) \tag{24}$$

After calculating the error signal using this equation, the adaptive filters are updated. However, if we use $\hat{e}(n)$ directly for updating both of a linear and a kernel filters, the update formula contains input signals of both filters as the one in the previous subsection. Hence, here, we consider another way to update each filters independently.

In contrast to (16) where all the filters are updated at once, we propose to divide the update process into two steps in the proposed method. Namely, first step, we propose to update only the coefficients of linear filter $\boldsymbol{w}_L(n)$ using the standard NLMS algorithm as

$$\boldsymbol{w}_L(n+1) = \boldsymbol{w}_L(n) + \alpha \frac{e(n)}{\epsilon + \mathbf{x}^T(n)\mathbf{x}(n)}\mathbf{x}(n) \tag{25}$$

where α is the step-size parameter. After this, we calculate the a posteriori error $\hat{e}_{\mathrm{p}}(n)$ of the linear filter using the updated $\boldsymbol{w}_L(n+1)$

$$\begin{aligned}\hat{e}_{\mathrm{p}}(n) &= d(n) - y_{\mathrm{L}}(n+1) - y_{\mathrm{k}}(n) \\ &= d(n) - \mathbf{x}^{\mathrm{T}}(n)\boldsymbol{w}_L(n+1) - y_{\mathrm{k}}(n) \\ &= d(n) - \mathbf{x}^{\mathrm{T}}(n)\left\{\boldsymbol{w}_L(n) + \alpha\frac{e(n)\mathbf{x}(n)}{\epsilon + \mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)}\right\} - y_{\mathrm{k}}(n) \\ &= d(n) - (y_{\mathrm{L}}(n) + \Delta y_{\mathrm{L}}(n)) - y_{\mathrm{k}}(n) \\ &= \hat{e}(n) - \Delta y_{\mathrm{L}}(n)\end{aligned} \tag{26}$$

where $\Delta y_{\mathrm{L}}(n)$ is defined as

$$\Delta y_{\mathrm{L}}(n) = \alpha \frac{e(n)}{\epsilon + \mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n)}\mathbf{x}^{\mathrm{T}}(n)\mathbf{x}(n). \tag{27}$$

Then, as the second step, the kernel filters are updated using $e(n)$ as

$$\begin{aligned}\boldsymbol{h}_m(n+1) =& \boldsymbol{h}_m(n) + \eta \frac{e(n)}{\sum_{m=2}^{M} \mathcal{X}_m^T(n)\mathcal{X}_m(n)}\mathcal{X}_m(n) \\ & m = 1, 2, \ldots, M\end{aligned} \tag{28}$$

The difference from the multi-kernel adaptive filter[9] is to use the a posteriori error to calculate the error signal $e(n)$ in the equation (26). In Fig. 3, we show a configuration of the proposed method.

Note that the term $\Delta y_{\mathrm{L}}(n)$ in (27) can be calculated before updating $\boldsymbol{w}_L(n)$. Hence, the two-step update described above is merely conceptual, and we could update all the filters at once.

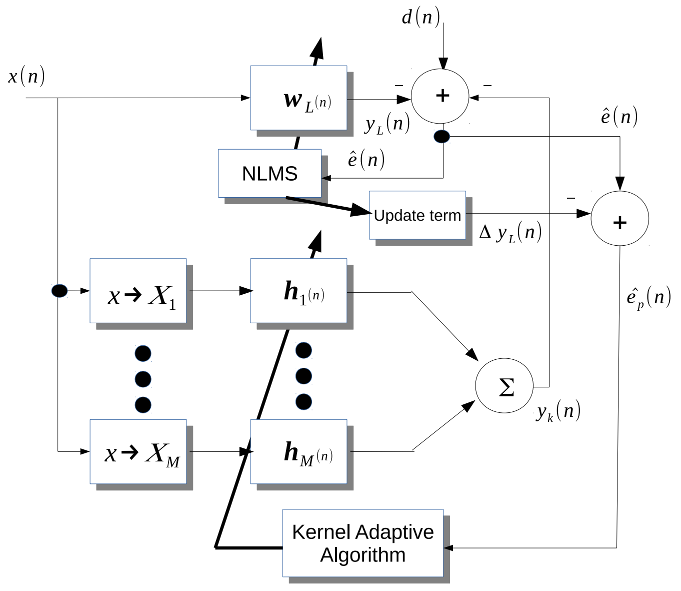

Fig. 3. Configuration of the proposed method. 'Update term' shows the calculation of the equation (27)

IV. Simulation Results

Finally, we show results of computer simulations using the proposed method. In the simulations, we compared four algorithms, namely, (i) the linear NLMS algorithm, (ii) the multi-kernel NLMS algorithm, (iii) the multi-kernel NLMS algorithm with a linear kernel, and (iv) the proposed method.

We applied these algorithms to three types of simulations and common conditions for them are as below. We set the step-size parameter in (i) as $\alpha = 0.1$. For (ii), we used two Gaussian kernels with different kernel bandwidth. The kernel NLMS algorithm was used and its step-size was set as $\eta = 0.1$ for all the kernel filters. In addition to these conditions for (ii), a linear kernel was added in (iii) and the NLMS algorithm was used for updating the linear filter with its step-size set as $\alpha = 0.1$. Also for the proposed method, the conditions for (ii) were used, namely two Gaussian kernels with different bandwidth and $\eta = 0.1$, and the linear filter was updated by the NLMS algorithm with $\alpha = 0.1$. The additive Gaussian noise was added to the desired signal whose mean was 0 and the variance was fixed as $\sigma_n^2 = 0.001$. The results shown in this section are ensemble averages of 500 independent simulations.

Note that, in the following description, we use the term 'multi kernel-linear NLMS' to express the algorithm (iii).

A. Non-linear prediction

First, we show simulation results of adaptive prediction[4], in which, the signal was generated by the equation

$$\begin{aligned} u(n) =& \left(c_1 - c_2 \exp\left(-u(n-1)^{-2}\right)\right) u(n-1) \\ &- \left(c_3 + c_4 \exp\left(-u(n-1)^2\right)\right) u(n-2) \\ &+ c_5 \sin\left(u(n-1)\pi\right) \qquad (29) \end{aligned}$$

$$[c_1 \;\; c_2 \;\; c_3 \;\; c_4 \;\; c_5] = [0.5 \;\; 0.2 \;\; 0.1 \;\; 0.8 \;\; 0.1] \qquad (30)$$

and the initial values of $u(-1)$ and $u(-2)$ were set as random numbers from a uniformly distributed random variable in the region $(0, 1)$, and the length of the signal was 1000. Besides, $u(n)$ was corrupted by additive noise as described before. The threshold value $\mathcal{T}_0$ was set as 0.8, and the kernel bandwidth parameters were set as -1.8 and -10 for the algorithms (ii), (iii), (iv).

The results of the simulations are shown in Fig. 4. From the figure, it is confirmed that the proposed method provides faster initial convergence rate which is inherited from the linear NLMS algorithm.

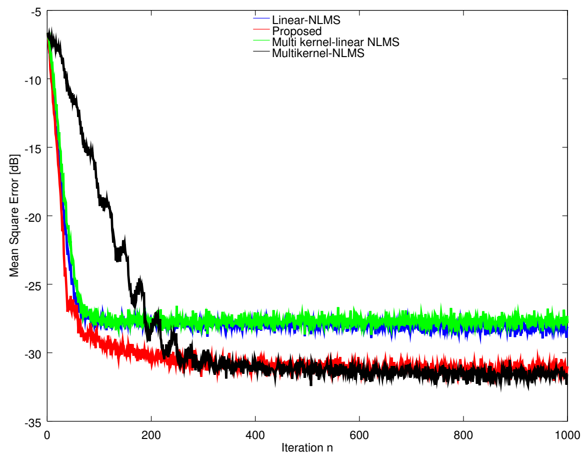

Fig. 4. Comparison of the algorithms in terms of MSE for prediction problem.

To confirm the tracking property of the proposed method, we simulated time-varying prediction. The values of coefficients in (29) were changed at $n = 500$ from (30) to the values below:

$$[c_1 \;\; c_2 \;\; c_3 \;\; c_4 \;\; c_5] = [0.8 \;\; 0.5 \;\; 0.3 \;\; 0.9 \;\; 0.1] \qquad (31)$$

The results are shown in Fig. 5. We can see that the tracking performance of the proposed method is almost same as other algorithms.

B. Linear-dominant model

Next, we applied the adaptive filters to the following rather artificial mixture model of linear and non-linear systems.

$$\begin{aligned} d(n) =& a_1 u(n) + a_2 u(n-1) + a_3(0.8 - 0.5\exp(-u^2(n)))u(n) \\ &- a_4(0.3 + 0.9\exp(-u^2(n))u(n-1) \\ &- 0.1 a_5 \sin(u(n)\pi) \qquad (32) \end{aligned}$$

where the coefficients $a_1, \ldots, a_5$ were set as

$$[a_1 \;\; a_2 \;\; a_3 \;\; a_4 \;\; a_5] = \begin{cases} [0.5 \;\; 0.5 \;\; 0.2 \;\; 0.2 \;\; 0.2] & (n \le 500) \\ [0.3 \;\; 0.0 \;\; 0.0 \;\; 0.5 \;\; 0.5] & (n > 500) \end{cases} \qquad (33)$$

We set $\mathcal{T}_0$ as 0.8, and ζ_m as -1.8 and -10 for the algorithms (ii), (iii), (iv).

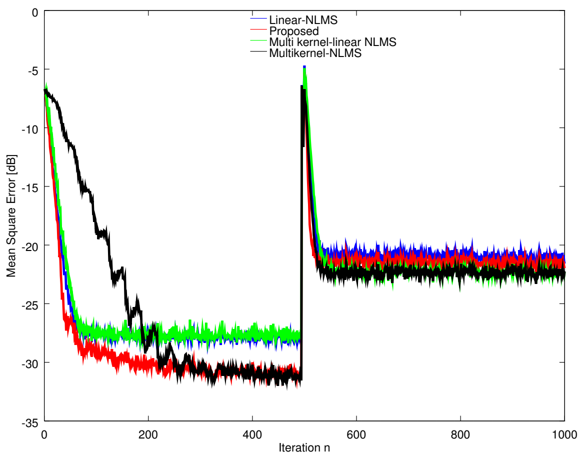

Fig. 5. Comparison of the algorithms in terms of MSE for prediction problem.

The results are shown in Fig. 6. From the figure, we can see that there are differences in the initial stage between the convergence characteristics of the multi-kernel-linear NLMS and the proposed method.

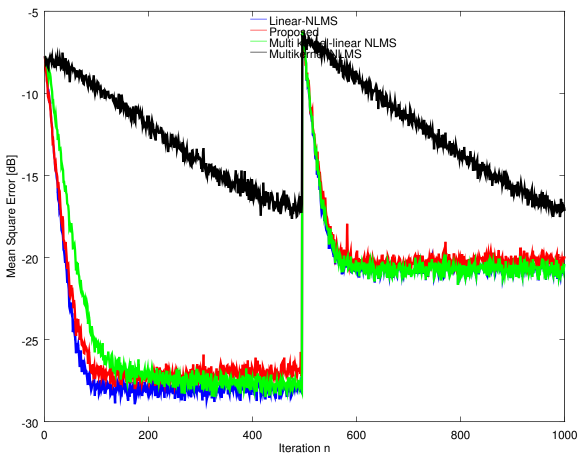

Fig. 6. Comparison of the algorithms in terms of MSE for linear dominant system identification.

C. Non linear system estimation

Finally, we examine the performance of the proposed method under the non-linear model studied in [16]:

$$\begin{cases} y(n) = \dfrac{y(n-1)}{1+y^2(n-1)} + u^3(n-1) \\ d(n) = y(n) + \xi(n) \end{cases} \tag{34}$$

where $u(n)$ and $y(n)$ are the input and the output signals respectively. The input signal $u(n)$ was generated from a zero-mean Gaussian process with the standard deviation $\sigma_u = 1$.

The results are shown in Fig. 7. From the figure, we can see that the linear NLMS filter results in a higher MSE than other algorithms. On the other hand, the proposed method achieves almost same MSE with other kernel based ones.

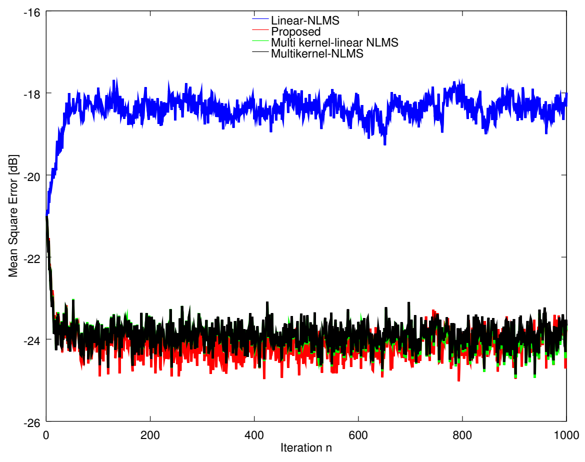

Fig. 7. Comparison of the algorithms in terms of MSE for the system defined by the equation (34).

The last model is also from [16], the signal $d(n)$ was generated by the equation below:

$$\phi(y(n)) = \begin{cases} \dfrac{y(n)}{3[0.1+0.9y^2(n)]^{1/2}} & \text{for } y(n) \le 0 \\ \dfrac{-y^2(n)[1-\exp(0.7y(n)]}{3} & \text{for } y(n) < 0 \end{cases} \tag{35}$$

$$d(n) = \phi(y(n)) + \xi(n) \tag{36}$$

where $\xi(n)$ is the same as the previous model, and $y(n)$ is given as

$$y(n) = \mathbf{a}^{\mathrm{T}}\mathbf{u}(n) - 0.2y(n-1) + 0.35y(n-2) \tag{37}$$

where $\mathbf{a} = [1 \quad 0.5]^{\mathrm{T}}$ and $\mathbf{u}(n)$ is

$$\mathbf{u}(n) = [u_1(n) \quad u_2(n)]^{\mathrm{T}}. \tag{38}$$

The results are shown in Fig. 8. From the figure, for this model, the mixture of the linear and kernel filters provide better convergence characteristics than the linear or kernel only configurations. Besides, the proposed method provides a faster rate of convergence at the first stage compared to that of the multi-kernel-linear NLMS.

V. Conclusions

In this paper, we considered a mixture structure of the linear and the multi-kernel adaptive filter for modeling the environments that can be modeled as a Volterra series. We derived an equivalent update equation when a linear kernel is used in the multi-kernel adaptive filter as (21), and showed that the update equation is affected by other kernels. Then, we proposed a new structure for the mixture usage of the kernel and linear adaptive filters. We proposed update equations of kernel filters using the a posteriori error of the liner adaptive filter. The results of computer simulation shows the possibility of better convergence characteristics of the proposed method.

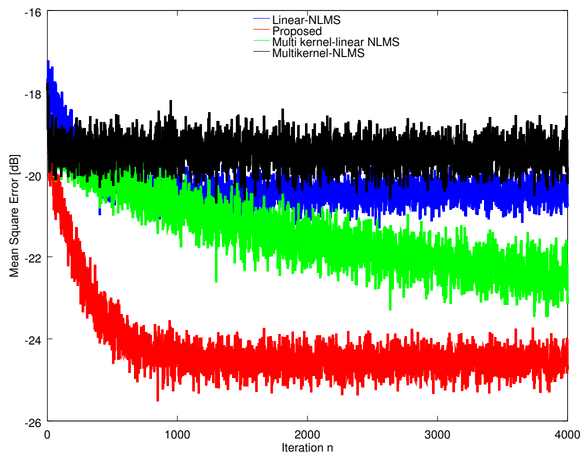

Fig. 8. Comparison of the algorithms in terms of MSE for the system defined by the equation (36).

We will focus our future work on the theoretical analysis of the proposed method.

Acknowledgment

References

[1] A. H. Sayed, *Fundamentals of Adaptive Filtering*. John Wiley & Sons, 2003.

[2] A. H. Sayed, *Adaptive Filters*. John Wiley & Sons, 2008.

[3] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering*. Wiley, 2010.

[4] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online Prediction of Time Series Data With Kernels," *IEEE Transactions on Signal Processing*, vol. 57, pp. 1058–1067, Mar. 2009.

[5] F. Albu and K. Nishikawa, "The Kernel Proportionate NLMS Algorithm," in *Proc. EUSIPCO 2013*, (Marrakech, Morocco), Sept. 2013.

[6] Y. Engel, S. Mannor, and R. Meir, "The Kernel Recursive Least-Squares Algorithm," *IEEE Transactions on Signal Processing*, vol. 52, pp. 2275–2285, Aug. 2004.

[7] Y. Ogawa and K. Nishikawa, "A Kernel Adaptive Filter based on ERLS-DCD Algorithm," in *Proc. of Intl Tech. Conf. Circuits Systems, Computer, Communications 2011*, (Gyeongju), pp. 1228–1231, June 2011.

[8] K. Nishikawa, Y. Ogawa, and F. Albu, "Fixed Order Implementation of Kernel RLS-DCD Adaptive Filters," in *Proc. APSIPA ASC 2013*, Oct. 2013.

[9] M. Yukawa, "Multikernel Adaptive Filtering," *Signal Processing, IEEE Transactions on*, vol. 60, pp. 4672–4682, Sept 2012.

[10] V. Mathews, "Adaptive polynomial filters," *Signal Processing Magazine, IEEE*, vol. 8, pp. 10–26, July 1991.

[11] T. Panicker and V. Mathew, "Parallel-cascade realizations and approximations of truncated volterra systems," *Signal Processing, IEEE Transactions on*, vol. 46, pp. 2829–2832, Oct 1998.

[12] F. Kuech and W. Kellermann, "Partitioned block frequency-domain adaptive second-order volterra filter," *Signal Processing, IEEE Transactions on*, vol. 53, pp. 564–575, Feb 2005.

[13] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[14] J. Nagumo and A. Noda, "A learning method for system identification," *IEEE Transactions on Automatic Control*, vol. 12, pp. 282–287, June 1967.

[15] W. Liu, I. Park, and J. C. Principe, "An information theoretic approach of designing sparse kernel adaptive filters.," *IEEE transactions on neural networks*, vol. 20, pp. 1950–61, Dec. 2009.

[16] W. Gao, J. Chen, C. Richard, and J. Huang, "Online Dictionary Learning for Kernel LMS," *IEEE Transactions on Signal Processing*, vol. 62, pp. 2765–2777, June 2014.