# EXTENDED RECURSIVE LEAST SQUARES IN RKHS

*Weifeng Liu and José C. Príncipe*

Electrical and Computer Engineering, University of Florida

## ABSTRACT

In this paper, a kernelized version of the extended recursive least squares (Ex-RLS) algorithm, along with its Kalman filter interpretation will be presented. The center piece of this development is a reformulation of the Ex-RLS algorithm which only requires inner product operations between input vectors. Thus, the kernel trick can be readily applied to obtain nonlinear versions in reproducing kernel Hilbert spaces (RKHS). In so doing, we arrive at extended RLS algorithms with kernels that are better suited for tracking the state-vector of general linear state-space models in the feature space, when compared with a fixed state model in the standard recursive least squares. The proposed kernel Ex-RLS is applied to a nonlinear Rayleigh multipath channel tracking problem. We show that the proposed algorithm is able to outperform the standard kernel RLS in a fading environment.

***Index Terms***— Extended recursive least squares, Kalman filter, kernel RLS, kernel Kalman filter.

## 1. INTRODUCTION

In recent years, there has been increasing interest to combine kernel methods and adaptive filtering techniques to achieve computationally efficient online extensions with nonlinear modeling capabilities, such as kernel least-mean-square (LMS) [1], kernel leaky LMS [2], kernel recursive least squares (RLS) [3] and sliding-window kernel RLS [4].

In this paper, we present a general variant of the kernel RLS, namely the kernel extended RLS (Ex-KRLS) with a very appealing feature. Besides providing a generalization of the KRLS and exponentially-weighted KRLS, the state-vector model of the Ex-KRLS allows tracking variations in the weight vector with linear state-space recursions. Due to the close relationship between the Ex-RLS [5] and the Kalman filter [6], it may be possible to derive a kernel Kalman filter, which corresponds to a nonlinear state filter in the input space. This possibility opens a new research line in the area of nonlinear Kalman filtering [7, 8, 9, 10].

This work is partially supported by NSF Grant ECS-0601271.

## 2. A REVIEW ON RLS AND KERNEL RLS

With a sequence of training data $\{\mathbf{u}(t), d(t)\}_{t=1}^{i-1}$ up to time $i-1$, the recursive least squares algorithm estimates the optimal weight $\mathbf{w}(i-1)$ by minimizing the following cost

$$\min_{\mathbf{w}(i-1)} \sum_{j=1}^{i-1} |d(j) - \mathbf{u}(j)^T \mathbf{w}(i-1)|^2 + \lambda ||\mathbf{w}(i-1)||^2 \quad (1)$$

Here $\mathbf{u}(t)$ is the $L \times 1$ regressor input, $d(t)$ is the desired response, and $\lambda$ is the regularization parameter. When a new input-output pair $\{\mathbf{u}(i), d(i)\}$ becomes available, the optimal weight $\mathbf{w}(i)$ which is the minimizer of

$$\min_{\mathbf{w}(i)} \sum_{j=1}^{i} |d(j) - \mathbf{u}(j)^T \mathbf{w}(i)|^2 + \lambda ||\mathbf{w}(i)||^2 \quad (2)$$

can be calculated recursively from the previous estimate $\mathbf{w}(i-1)$ without solving (2) directly. The standard recursive least squares (RLS) is as follows [7]:

**Algorithm 1** The RLS algorithm (RLS)

Start with $\mathbf{w}(0) = 0$, $\mathbf{P}(0) = \lambda^{-1}\mathbf{I}$
iterate for $i \geq 1$

$$\begin{aligned} r_e(i) &= 1 + \mathbf{u}(i)^T \mathbf{P}(i-1)\mathbf{u}(i) \\ \mathbf{k}_p(i) &= \mathbf{P}(i-1)\mathbf{u}(i)/r_e(i) \\ e(i) &= d(i) - \mathbf{u}(i)^T \mathbf{w}(i-1) \\ \mathbf{w}(i) &= \mathbf{w}(i-1) + \mathbf{k}_p(i)e(i) \\ \mathbf{P}(i) &= [\mathbf{P}(i-1) - \mathbf{P}(i-1)\mathbf{u}(i) \\ &\qquad \mathbf{u}(i)^T \mathbf{P}(i-1)/r_e(i)] \end{aligned} \quad (3)$$

Notice that the RLS distributes the computation load evenly into each iteration, which is very appealing in applications like channel equalization [5] where data is available sequentially over time.

The conventional kernel methods such as regularization networks [11], support vector machines [12] are formulated in a batch mode and are computationally expensive. Therefore efficient online alternatives are much needed to better address the issue especially in applications of nonlinear adaptive filtering and digital communications. Here, we mainly focus on

the kernel recursive least squares [3] and give a brief review on it to better distinguish and appreciate our contribution in this paper. Only the main idea is presented for simplicity and please refer to [3] for details.

A kernel [13] is a continuous, symmetric, positive-definite function $\kappa : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$. $\mathbb{U}$ is the input domain, a compact subset of $\mathbb{R}^L$. The commonly used kernels include the Gaussian kernel (4) and the polynomial kernel (5):

$$\kappa(\mathbf{u}, \mathbf{u}') = \exp(-a||\mathbf{u} - \mathbf{u}'||^2) \tag{4}$$

$$\kappa(\mathbf{u}, \mathbf{u}') = (\mathbf{u}^T\mathbf{u}' + 1)^p \tag{5}$$

The Mercer theorem [13], [14] states that any kernel $\kappa(\mathbf{u}, \mathbf{u}')$ can be expanded as follows:

$$\kappa(\mathbf{u}, \mathbf{u}') = \sum_{i=1}^{\infty} \varsigma_i \phi_i(\mathbf{u})\phi_i(\mathbf{u}') \tag{6}$$

where $\varsigma_i$ and $\phi_i$ are the eigenvalues and the eigenfunctions respectively. The eigenvalues are non-negative. Therefore, a mapping $\varphi$ can be constructed as

$$\begin{aligned} &\varphi : \mathbb{U} \rightarrow \mathbb{F} \\ &\varphi(\mathbf{u}) = [\sqrt{\varsigma_1}\phi_1(\mathbf{u}), \sqrt{\varsigma_2}\phi_2(\mathbf{u}), ...] \end{aligned}$$

such that

$$\kappa(\mathbf{u}, \mathbf{u}') = \varphi(\mathbf{u})^T\varphi(\mathbf{u}') \tag{7}$$

By construction, the dimensionality of $\mathbb{F}$ is determined by the number of strictly positive eigenvalues, which can be infinite in the case of the Gaussian kernel.

We utilize this theorem to transform the data $\mathbf{u}(i)$ into the feature space $\mathbb{F}$ as $\varphi(\mathbf{u}(i))$ and interpret (7) as the usual dot product. The feature space $\mathbb{F}$ is a reproducing kernel Hilbert space (RKHS). Denoting $\varphi(i) = \varphi(\mathbf{u}(i))$, we now formulate the recursive least squares algorithm on the example sequence $\{d(1), d(2), \dots\}$ and $\{\varphi(1), \varphi(2), \dots\}$. At each iteration, the weight vector $\boldsymbol{\omega}(i)$, which is the minimizer of

$$\min_{\boldsymbol{\omega}(i)} \sum_{j=1}^{i} |d(j) - \boldsymbol{\omega}(i)^T\varphi(j)|^2 + \lambda||\boldsymbol{\omega}(i)||^2 \tag{8}$$

needs to be solved recursively as in (3). However, (3) can not be directly applied here because the dimensionality of $\varphi(j)$ is so high (can be even infinite) that it is not feasible in practice.

Introducing

$$\begin{aligned} \mathbf{d}(i) &= [d(1), ..., d(i)]^T \\ \Phi(i) &= [\varphi(1), ..., \varphi(i)] \end{aligned} \tag{9}$$

one has

$$\boldsymbol{\omega}(i) = [\lambda\mathbf{I} + \boldsymbol{\Phi}(i)\boldsymbol{\Phi}(i)^T]^{-1}\boldsymbol{\Phi}(i)\mathbf{d}(i) \tag{10}$$

further by the matrix inversion lemma [5],

$$\boldsymbol{\omega}(i) = \boldsymbol{\Phi}(i)[\lambda\mathbf{I} + \boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i)]^{-1}\mathbf{d}(i) \tag{11}$$

We have to emphasize the significance of the change from (10) to (11) here. First, $\boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i)$ is computable by the kernel trick (7) and second the weight is explicitly expressed as a linear combination of the input data $\boldsymbol{\omega}(i) = \boldsymbol{\Phi}(i)\mathbf{a}(i)$. Denote

$$\mathbf{Q}(i) = (\lambda\mathbf{I} + \boldsymbol{\Phi}(i)^T\boldsymbol{\Phi}(i))^{-1} \tag{12}$$

It is easy to see that

$$\mathbf{Q}(i)^{-1} = \begin{bmatrix} \mathbf{Q}(i-1)^{-1} & \mathbf{h}(i-1,i) \\ \mathbf{h}(i-1,i)^T & \lambda + \varphi(i)^T\varphi(i) \end{bmatrix} \tag{13}$$

where $\mathbf{h}(i-1,i) = \boldsymbol{\Phi}(i-1)^T\varphi(i)$. Using this sliding-window structure, the updating of the inversion of this growing matrix can be quite efficient [5]

$$\mathbf{Q}(i) = \begin{bmatrix} \mathbf{Q}(i-1) + \mathbf{z}(i)\mathbf{z}(i)^T s(i) & -\mathbf{z}(i)s(i) \\ -\mathbf{z}(i)^T s(i) & s(i) \end{bmatrix} \tag{14}$$

where

$$\begin{aligned} \mathbf{z}(i) &= \mathbf{Q}(i-1)\mathbf{h}(i-1,i) \\ s(i) &= 1/(\lambda + \varphi(i)^T\varphi(i) - \mathbf{z}(i)^T\mathbf{h}(i-1,i)) \end{aligned} \tag{15}$$

Therefore the expansion coefficients of the weight are

$$\begin{aligned} &\mathbf{a}(i) = \mathbf{Q}(i)\mathbf{d}(i) \\ &= \begin{bmatrix} \mathbf{Q}(i-1) + \mathbf{z}(i)\mathbf{z}(i)^T s(i) & -\mathbf{z}(i)s(i) \\ -\mathbf{z}(i)^T s(i) & s(i) \end{bmatrix} \begin{bmatrix} \mathbf{d}(i-1) \\ d(i) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{a}(i-1) - \mathbf{z}(i)s(i)e(i) \\ s(i)e(i) \end{bmatrix} \end{aligned} \tag{16}$$

The KRLS is summarized in Algorithm 2.

**Algorithm 2** The kernel RLS algorithm (KRLS)

Start with
$\mathbf{Q}(1) = (\lambda + \kappa(\mathbf{u}(1), \mathbf{u}(1))^{-1}$
$\mathbf{a}(1) = \mathbf{Q}(1)d(1)$
iterate for $i > 1$:

$$\begin{aligned} &\mathbf{h}(i-1,i) = \boldsymbol{\Phi}(i-1)^T\varphi(i) \\ &\mathbf{z}(i) = \mathbf{Q}(i-1)\mathbf{h}(i-1,i) \\ &s(i) = 1/(\lambda + \varphi(i)^T\varphi(i) - \mathbf{z}(i)^T\mathbf{h}(i-1,i)) \\ &\mathbf{Q}(i) = \begin{bmatrix} \mathbf{Q}(i-1) + \mathbf{z}(i)\mathbf{z}(i)^T s(i) & -\mathbf{z}(i)s(i) \\ -\mathbf{z}(i)^T s(i) & s(i) \end{bmatrix} \\ &e(i) = d(i) - \mathbf{h}(i-1,i)^T\mathbf{a}(i-1) \\ &\mathbf{a}(i) = \begin{bmatrix} \mathbf{a}(i-1) - \mathbf{z}(i)s(i)e(i) \\ s(i)e(i) \end{bmatrix} \end{aligned} \tag{17}$$

Given an input $\mathbf{u}'$, the output of the system can be calculated by

$$\boldsymbol{\omega}(i)^T\varphi(\mathbf{u}') = \sum_{j=1}^{i} \mathbf{a}_j(i)\kappa(\mathbf{u}(j), \mathbf{u}')$$

which assumes a radial basis function network structure [15]. Here $\mathbf{a}_j(i)$ is the $jth$ component of $\mathbf{a}(i)$.

## 3. THE EXTENDED RECURSIVE LEAST-SQUARES ALGORITHM

The problem for the RLS (and the KRLS) is that it has a poor tracking performance. From the viewpoint of state-space model, the RLS implicitly assumes that the data satisfy [7]

$$\begin{aligned} \mathbf{x}(i+1) &= \mathbf{x}(i) \\ d(i) &= \mathbf{u}(i)^T\mathbf{x}(i) + v(i) \end{aligned} \tag{18}$$

i.e. the state $\mathbf{x}(i)$ is fixed over time and with no surprise the optimal estimate of the state $\mathbf{w}(i)$ can not track variations.

To improve its tracking ability, several techniques can be employed. For example, the data can be exponentially-weighted over time or a truncated window can be applied on the training data (which also can be viewed as a special weighting). The work for a sliding-window KRLS has already be done in [4] and this paper tries to derive an exponentially-weighted KRLS and a step further.

For the exponentially-weighted scheme, the assumed state-space model is[1]

$$\begin{aligned} \mathbf{x}(i+1) &= \alpha\mathbf{x}(i) \\ d(i) &= \mathbf{u}(i)^T\mathbf{x}(i) + v(i) \end{aligned} \tag{19}$$

where $\alpha$ is a scaling factor. As is known from the extended RLS method, the most general state-space model would be

$$\begin{aligned} \mathbf{x}(i+1) &= \mathbf{A}\mathbf{x}(i) + \mathbf{n}(i) \\ d(i) &= \mathbf{u}(i)^T\mathbf{x}(i) + v(i) \end{aligned} \tag{20}$$

with $\mathbf{A}$ as the state transition matrix. While it would be most desirable to have such a general state-space model in the RKHS, it turns out to be at best very difficult. In this paper we focus on a special case of (20), i.e.,

$$\begin{aligned} \mathbf{x}(i+1) &= \alpha\mathbf{x}(i) + \mathbf{n}(i) \\ d(i) &= \mathbf{u}(i)^T\mathbf{x}(i) + v(i) \end{aligned} \tag{21}$$

Kalman [6] proposed a two step sequential estimation algorithm to update the state estimate. At the core of this procedure is the recursive least square update of the observation model. Indeed, the solution of this state-space estimation problem amounts to solving the following least squares cost function [5]:

$$\begin{aligned} \min_{\{\mathbf{x}(1),\mathbf{n}(1),\ldots,\mathbf{n}(i)\}} [&\sum_{j=1}^{i} \beta^{i-j}|d(j) - \mathbf{u}(j)^T\mathbf{x}(j)|^2 \\ &+\lambda\beta^i||\mathbf{x}(1)||^2 + q^{-1}\sum_{j=1}^{i}\beta^{i-j}||\mathbf{n}(j)||^2] \end{aligned} \tag{22}$$

subject to $\mathbf{x}(j+1) = \alpha\mathbf{x}(j) + \mathbf{n}(j)$. $\beta$ is introduced to have exponential weighting on the past data, $\lambda$ is the regularization parameter to control the initial state-vector norm and $q$ provides some trade-off between the modeling variation and measurement disturbance. Observe that if $q = 0$, $\alpha = 1$ , (22) reduces to the exponentially-weighted RLS. Further if $\beta = 1$, it reduces to the standard RLS. We have to emphasize that (22) is a much harder quadratic optimization problem with linear constraints compared to the constraint-free least squares problem (2). When the input data are transformed into a high dimensional feature space via a kernel mapping, this problem gets just much harder.

The extended RLS recursions are given by [5]:

**Algorithm 3** The extended RLS algorithm (Ex-RLS)

Start with $\mathbf{w}(0) = 0$, $\mathbf{P}(0) = \lambda^{-1}\beta^{-1}\mathbf{I}$
iterate for $i \geq 1$

$$\begin{aligned} r_e(i) &= \beta^i + \mathbf{u}(i)^T\mathbf{P}(i-1)\mathbf{u}(i) \\ \mathbf{k}_p(i) &= \alpha\mathbf{P}(i-1)\mathbf{u}(i)/r_e(i) \\ e(i) &= d(i) - \mathbf{u}(i)^T\mathbf{w}(i-1) \\ \mathbf{w}(i) &= \alpha\mathbf{w}(i-1) + \mathbf{k}_p(i)e(i) \\ \mathbf{P}(i) &= |\alpha|^2[\mathbf{P}(i-1) - \mathbf{P}(i-1)\mathbf{u}(i) \\ &\qquad \mathbf{u}(i)^T\mathbf{P}(i-1)/r_e(i)] + \beta^i q\mathbf{I} \end{aligned} \tag{23}$$

## 4. THE KERNEL EXTENDED RLS ALGORITHM

In the feature space $\mathbb{F}$, the model becomes

$$\begin{aligned} \mathbf{x}(i+1) &= \alpha\mathbf{x}(i) + \mathbf{n}(i) \\ d(i) &= \varphi(i)^T\mathbf{x}(i) + v(i) \end{aligned} \tag{24}$$

which is similar to (21) except that the input is $\varphi(i)$ instead of $\mathbf{u}(i)$.

We can not use (23) directly because the input data and the state vector now lie in a possibly infinite dimensional space. We can not use the matrix inversion lemma either like in the KRLS because of the complicated constrained least squares cost function (22). The application of the kernel trick requires the reformulation of the recursion solely in terms of inner product operations between input vectors.

By carefully observing the recursion (23), one can conclude that all the calculations are based on $\mathbf{u}(j)^T\mathbf{P}(k)\mathbf{u}(i)$ for any $k$, $i$, $j$.

**Theorem 4.1** *The matrices* $\mathbf{P}(j)$ *in* (23) *assume the following form*

$$\mathbf{P}(j) = \rho(j)\mathbf{I} - \mathbf{H}(j)\mathbf{Q}(j)\mathbf{H}(j)^T \tag{25}$$

*where* $\rho(j)$ *is a scalar and* $\mathbf{H}(j) = [\mathbf{u}(1), ..., \mathbf{u}(j)]$ *and* $\mathbf{Q}(j)$ *is an* $j \times j$ *matrix, for all* $j > 0$.

**Proof** First notice that by (23)

[1]This fact is nontrivial and please refer to [5] for details.

$$\mathbf{P}(0) = \lambda^{-1}\beta^{-1}\mathbf{I},$$
$$\mathbf{P}(1) = |\alpha|^2[\lambda^{-1}\beta^{-1}\mathbf{I} - \frac{\lambda^{-2}\beta^{-2}\mathbf{u}(1)\mathbf{u}(1)^T}{r_e(i)}] + \beta q\mathbf{I}$$
$$= [\frac{|\alpha|^2}{\lambda\beta} + \beta q]\mathbf{I} - \mathbf{u}(1)[\frac{|\alpha|^2\lambda^{-2}\beta^{-2}}{\beta + \lambda^{-1}\beta^{-1}\mathbf{u}(1)^T\mathbf{u}(1)}]\mathbf{u}(1)^T$$

so the claim is valid for $j = 1$, namely,

$$\rho(1) = |\alpha|^2\lambda^{-1}\beta^{-1} + \beta q,$$
$$\mathbf{Q}(1) = \frac{|\alpha|^2\lambda^{-2}\beta^{-2}}{\beta + \lambda^{-1}\beta^{-1}\mathbf{u}(1)^T\mathbf{u}(1)}$$

Then using the mathematical induction, the proof for all $j$ follows. Assume it is true for $j = i - 1$, i.e.,

$$\mathbf{P}(i-1) = \rho(i-1)\mathbf{I} - \mathbf{H}(i-1)\mathbf{Q}(i-1)\mathbf{H}(i-1)^T \quad (26)$$

By substituting it into the last equation of (23), one has

$$\mathbf{P}(i) = |\alpha|^2[\mathbf{P}(i-1) - \tfrac{\mathbf{P}(i-1)\mathbf{u}(i)\mathbf{u}(i)^T\mathbf{P}(i-1)}{r_e(i)}] + \beta^i q\mathbf{I}$$
$$= (|\alpha|^2\rho(i-1) + \beta^i q)\mathbf{I} - |\alpha|^2 r_e^{-1}(i)\mathbf{H}(i)$$
$$\begin{bmatrix} \mathbf{Q}(i-1)r_e(i) + \mathbf{z}(i)\mathbf{z}(i)^T & -\rho(i-1)\mathbf{z}(i) \\ -\rho(i-1)\mathbf{z}(i)^T & \rho^2(i-1) \end{bmatrix}\mathbf{H}(i)^T$$

Therefore

$$\rho(i) = |\alpha|^2\rho(i-1) + \beta^i q$$
$$\mathbf{Q}(i) = \frac{|\alpha|^2}{r_e(i)}\begin{bmatrix} \mathbf{Q}(i-1)r_e(i) + \mathbf{z}(i)\mathbf{z}(i)^T & -\rho(i-1)\mathbf{z}(i) \\ -\rho(i-1)\mathbf{z}(i)^T & \rho^2(i-1) \end{bmatrix}$$
$$\mathbf{P}(i) = \rho(i) - \mathbf{H}(i)\mathbf{Q}(i)\mathbf{H}(i)^T$$

where $\mathbf{z}(i) = \mathbf{Q}(i-1)\mathbf{H}(i-1)^T\mathbf{u}(i)$. ∎

By theorem 4.1, the calculation $\mathbf{u}(j)^T\mathbf{P}(k)\mathbf{u}(i)$ only involves inner product operations between the input vectors, which is the prerequisite to using the kernel trick.

**Theorem 4.2** *The optimal state estimate in* (23) *is a linear combination of the past input vector, namely*

$$\mathbf{w}(j) = \mathbf{H}(j)\mathbf{a}(j) \quad (27)$$

**Proof** Notice that by (23)

$$\mathbf{w}(0) = 0$$
$$\mathbf{w}(1) = \frac{\alpha\lambda^{-1}\beta^{-1}d(1)\mathbf{u}(1)}{\beta + \lambda^{-1}\beta^{-1}\mathbf{u}(1)^T\mathbf{u}(1)}$$

so

$$\mathbf{a}(1) = \frac{\alpha\lambda^{-1}\beta^{-1}d(1)}{\beta + \lambda^{-1}\beta^{-1}\mathbf{u}(1)^T\mathbf{u}(1)}$$

Thus the claim is valid for $j = 1$. Then we use the mathematical induction to prove it is valid for any $j$. Assume it is true for $i - 1$. By the recursion (23) and the result from theorem 4.1, we have

$$\mathbf{w}(i) = \alpha\mathbf{w}(i-1) + k_p(i)e(i)$$
$$= \alpha\mathbf{H}(i-1)\mathbf{a}(i-1) + \alpha\mathbf{P}(i-1)\mathbf{u}(i)e(i)/r_e(i)$$
$$= \alpha\mathbf{H}(i-1)\mathbf{a}(i-1) + \alpha\rho(i-1)\mathbf{u}(i)e(i)/r_e(i)$$
$$-\alpha\mathbf{H}(i-1)\mathbf{z}(i)e(i)/r_e(i)$$
$$= \mathbf{H}(i)\begin{bmatrix} \alpha\mathbf{a}(i-1) - \alpha\mathbf{z}(i)e(i)r_e^{-1}(i) \\ \alpha\rho(i-1)e(i)r_e^{-1}(i) \end{bmatrix}$$

where $\mathbf{z}(i) = \mathbf{Q}(i-1)\mathbf{H}(i-1)^T\mathbf{u}(i)$. ∎

Therefore, the recursion (23) is equivalent to the following:

**Algorithm 4** A novel variant of extended RLS (Ex-RLS-2)

Start with

$$\mathbf{a}(1) = \frac{\alpha\lambda^{-1}\beta^{-1}d(1)}{\beta + \lambda^{-1}\beta^{-1}\mathbf{u}(1)^T\mathbf{u}(1)},$$
$$\rho(1) = |\alpha|^2\lambda^{-1}\beta^{-1} + \beta q,$$
$$\mathbf{Q}(1) = \frac{|\alpha|^2\lambda^{-2}\beta^{-2}}{\beta + \lambda^{-1}\beta^{-1}\mathbf{u}(1)^T\mathbf{u}(1)}$$

iterate for $i > 1$:

$$\mathbf{h}(i) = \mathbf{H}(i-1)^T\mathbf{u}(i)$$
$$\mathbf{z}(i) = \mathbf{Q}(i-1)\mathbf{h}(i)$$
$$r_e(i) = \beta^i + \rho(i-1)\mathbf{u}(i)^T\mathbf{u}(i) - \mathbf{h}(i)^T\mathbf{z}(i)$$
$$e(i) = d(i) - \mathbf{h}(i)^T\mathbf{a}(i-1)$$
$$\mathbf{a}(i) = \alpha\begin{bmatrix} \mathbf{a}(i-1) - \mathbf{z}(i)r_e^{-1}(i)e(i) \\ \rho(i-1)r_e^{-1}(i)e(i) \end{bmatrix}$$
$$\rho(i) = |\alpha|^2\rho(i-1) + \beta^i q$$
$$\mathbf{Q}(i) = \frac{|\alpha|^2}{r_e(i)}\begin{bmatrix} \mathbf{Q}(i-1)r_e(i) + \mathbf{z}(i)\mathbf{z}(i)^T & -\rho(i-1)\mathbf{z}(i) \\ -\rho(i-1)\mathbf{z}(i)^T & \rho^2(i-1) \end{bmatrix}$$

Notice that throughout the iteration, the input vector $\mathbf{u}(i)$ is only in the calculation of $\mathbf{h}(i)$ and $r_e(i)$, both in the form of inner product. The significance of this reformulation is its independence on the data dimensionality. Comparing with Algorithm 3, we replace the recursion on $\mathbf{w}(i)$ with the one on $\mathbf{a}(i)$ in Algorithm 4. Further, we replace the recursion on $\mathbf{P}(i)$ with the ones on $\rho(i)$ and $\mathbf{Q}(i)$ where $\rho(i)$ is a scalar and $\mathbf{Q}(i)$ is $i \times i$. In a word, the dimension of $\mathbf{a}(i)$, $\rho(i)$ and $\mathbf{Q}(i)$ only depends on the size of training data $i$ regardless of the dimension of the input $\mathbf{u}$.

Now the nonlinear extension of Algorithm 4 is straightforward by replacing $\mathbf{u}(i)^T\mathbf{u}(j)$ with $\kappa(\mathbf{u}(i), \mathbf{u}(j))$ (See Algorithm 5).

**Algorithm 5** The kernel extended RLS for model (21) (Ex-KRLS)

Start with

$$\mathbf{a}(1) = \frac{\alpha\lambda^{-1}\beta^{-1}d(1)}{\beta + \lambda^{-1}\beta^{-1}\kappa(\mathbf{u}(1),\mathbf{u}(1))},$$
$$\rho(1) = |\alpha|^2\lambda^{-1}\beta^{-1} + \beta q,$$
$$\mathbf{Q}(1) = \frac{|\alpha|^2\lambda^{-2}\beta^{-2}}{\beta + \lambda^{-1}\beta^{-1}\kappa(\mathbf{u}(1),\mathbf{u}(1))}$$

iterate for $i > 1$:

$$\mathbf{h}(i) = [\kappa(\mathbf{u}(i),\mathbf{u}(1)),...,\kappa(\mathbf{u}(i),\mathbf{u}(i-1))]^T$$
$$\mathbf{z}(i) = \mathbf{Q}(i-1)\mathbf{h}(i)$$
$$r_e(i) = \beta^i + \rho(i-1)\kappa(\mathbf{u}(i),\mathbf{u}(i)) - \mathbf{h}(i)^T\mathbf{z}(i)$$
$$e(i) = d(i) - \mathbf{h}(i)^T\mathbf{a}(i-1)$$
$$\mathbf{a}(i) = \alpha \begin{bmatrix} \mathbf{a}(i-1) - \mathbf{z}(i)r_e^{-1}(i)e(i) \\ \rho(i-1)r_e^{-1}(i)e(i) \end{bmatrix}$$
$$\rho(i) = |\alpha|^2\rho(i-1) + \beta^i q$$
$$\mathbf{Q}(i) = \frac{|\alpha|^2}{r_e(i)} \begin{bmatrix} \mathbf{Q}(i-1)r_e(i) + \mathbf{z}(i)\mathbf{z}(i)^T & -\rho(i-1)\mathbf{z}(i) \\ -\rho(i-1)\mathbf{z}(i)^T & \rho^2(i-1) \end{bmatrix}$$

## 5. SIMULATIONS

We consider the problem of tracking a nonlinear Rayleigh fading multipath channel and compare the performance of the proposed Ex-KRLS algorithm to the original KRLS. Also performance of the normalized LMS, RLS and Ex-RLS are included for comparison.

The nonlinear Rayleigh fading multipath channel employed here is the cascade of a traditional Rayleigh fading multipath channel from [5] and a saturation nonlinearity. In the Rayleigh multipath fading channel, the number of the paths is chosen as $M = 5$, the maximum Doppler frequency $f_D = 50Hz$ and the sampling rate $T_s = 0.8\mu s$ (so it is a slow fading channel with the same fading rate for all the paths). All the tap coefficients are generated according to the Rayleigh model but only the real part is used in this experiment. A white Gaussian distributed time series (with unit power) is sent through this channel, corrupted with the additive white Gaussian noise (with variance $\sigma^2 = 0.001$) and then the saturation nonlinearity $y = \tanh(x)$ is applied on it, where $x$ is the output of the Rayleigh channel. The whole nonlinear channel is treated as a black box and only the input and output are known.

The tracking task is tested on 5 methods. The first one is the normalized LMS (regularization parameter $\epsilon = 10^{-3}$, step size $\eta = 0.25$); the second is the RLS (with regularization parameter $\epsilon = 10^{-3}$); the third one is the Ex-RLS ($\alpha = 0.999999984208$, $q = 3.158 \times 10^{-8}$, $\beta = 0.995$, $\epsilon = 10^{-3}$ according to [5] (on page 759). The last two are nonlinear methods, namely the KRLS (regularization parameter $\lambda = 0.1$) and the proposed Ex-KRLS ($\alpha = 0.999998$, $q = 10^{-5}$, $\beta = 0.995$, $\lambda = 0.1$). We use the Gaussian kernel in both cases with kernel parameter $a = 1$. Notice that $\alpha$ is very close to 1 and $q$ very close to 0 since the fading of the channel is very slow.

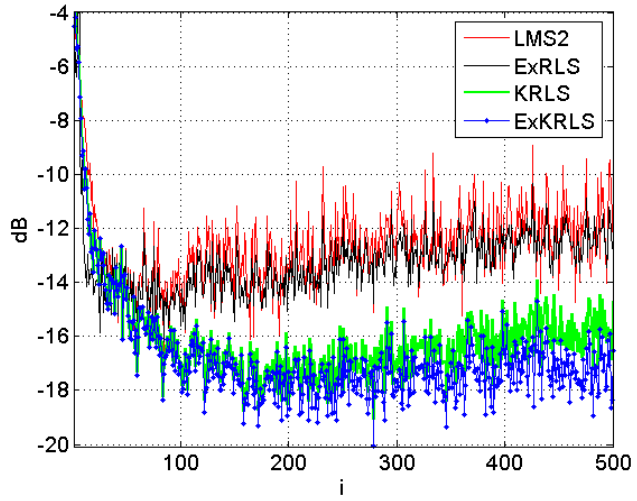


**Fig. 1**. The ensemble-average learning curves for $\epsilon$-NLMS, Ex-RLS, KRLS and Ex-KRLS in tracking a Rayleigh fading multipath channel (noise variance 0.001 and $f_D = 50Hz$).

We generate 300 symbols for every experiment and perform independently 100 Monte Carlo experiments. The ensemble learning curves are plotted in Figure 1. The last 100 values in the learning curves are used to calculate the final mean square error (MSE), which is listed in Table 1.

In Table 1, it is seen that the nonlinear methods outperform the linear methods significantly, since the channel model we use here is nonlinear. The Rayleigh channel is a slow fading channel in this problem, so the Ex-RLS performs almost identically as the RLS. Nevertheless, we enjoy 0.33dB margin by using the Ex-KRLS model. Further if we increase the maximum Doppler frequency to $200Hz$ and increase the noise variance to 0.01, the performance gap is enlarged to 1.3dB between the Ex-KRLS and KRLS and the difference between the Ex-RLS and RLS is also more visible.

## 6. DISCUSSION AND CONCLUSION

A kernel based version of the Ex-RLS (for tracking model) was presented. Comparing with the existing KRLS algorithms, it provides a more general state-space model, which is a step closer to the possible kernel Kalman filters. Preliminary results of this algorithm are promising, which suggests it can have a wide applicability in nonlinear extensions of most problems which is dealt with by the Ex-RLS.

The proposed algorithm is very appealing in applications from digital communications where the training sequence is

**Table 1**. Performance Comparison in Rayleigh Channel Tracking

| Algorithm | MSE (dB) ($\sigma^2 = 0.001$ and $f_D = 50Hz$ ) | MSE (dB) ($\sigma^2 = 0.01$ and $f_D = 200Hz$ ) |
|---|---|---|
| $\epsilon$-NLMS | -13.51 | -11.93 |
| RLS | -14.25 | -11.99 |
| Ex-RLS | -14.26 | -12.43 |
| KRLS | -20.36 | -15.79 |
| Ex-KRLS | -20.69 | -17.09 |

usually finite and within each transmission frame. However it is clear that Algorithms 2 and 5 have a growing network structure. The size of the network (also the dimension of $\mathbf{a}(i)$ and $\mathbf{Q}(i)$) increases linearly with the number of the training data. Several sparsification methods exist to address this shortcoming, for example, the novelty criterion introduced in [16] and the approximate linear dependency test (ALD) in [3]. A common feature of these methods is a basis dictionary is created instead of storing all the training data. Both the novelty criterion and the ALD test ease the growing problem to some extent. With the assumption of compact input domain, the size of the dictionary is guaranteed to be finite. However, in a nonstationary scenario, pruning methods with a fixed budget are more suitable [17, 18]. The integration of the novelty criterion and ALD into the Ex-KRLS is straightforward whereas a fixed-budget Ex-KRLS will be part of the future work.

The other important future work includes testing the proposed algorithm on more problems and generalizing the state-space model to a full-blown Kalman filter, which if possible might have a deep impact on the nonlinear Kalman filter research field.

## 7. REFERENCES


[1] P. Pokharel, Weifeng Liu, and José Príncipe, "Kernel lms," in *Proc. International Conference on Accoustics, Speech and Signal Processing 2007*, 2007, pp. 1421–1424.

[2] J. Kivinen, A. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. on Signal Processing*, vol. 52, pp. 2165–2176, Aug. 2004.

[3] Yaakov Engel, Shie Mannor, and Ron Meir, "The kernel recursive least-squares algorithm," *IEEE Trans. on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[4] S. Van Vaerenbergh, J. Via, and I. Santamaría, "A sliding-window kernel rls algorithm and its application to nonlinear channel identification," in *Proc. International Conference on Accoustics, Speech and Signal Processing 2006*, May 2006, pp. 789–792.

[5] A. Sayed, *Fundamentals of Adaptive Filtering*, Wiley, New York, 2003.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME - Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[7] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, NJ, 2002.

[8] E. A. Wan and R. van der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proc. of IEEE Symposium 2000 (AS-SPCC)*. ake Louise, Alberta, Canada, 2000.

[9] S. Arulampalam, S. Maskell, N. J. Gordon, and T. Clapp, "A tutorial on particle filters for on-line nonlinear/non-gaussian bayesian tracking," *IEEE Transactions of Signal Processing*, vol. 50(2), pp. 174–188, 2002.

[10] L. Ralaivola and F. d'Alche Buc, "Time series filtering, smoothing and learning using the kernel kalman filter," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 2005, pp. 1449–1454.

[11] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, pp. 219–269, 1995.

[12] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, New York, 1995.

[13] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.

[14] Christopher J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[15] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, second edition, 1998.

[16] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[17] Ofer Dekel, Shai Shalev-Shwartz, and Yoram Singer, "The forgetron: A kernel-based perceptron on a fixed budget," in *Advances in Neural Information Processing Systems 18*, Cambridge, MA, 2006, pp. 1342–1372, MIT Press.

[18] Y. Sun, P. Saratchandran, and N. Sundararajan, "A direct link minimal resource allocation network for adaptive noise cancellation," *Neural processing letters*, vol. 12, no. 3, pp. 255–265, 2000.